Package 'AdhereR'

June 23, 2022

Type Package

Title Adherence to Medications

Version 0.8.0

Author Dan Dediu [aut, cre],
Alexandra Dima [aut],
Samuel Allemann [aut]

Maintainer Dan Dediu <ddediu@gmail.com>

Description Computation of adherence to medications from Electronic Health care Data and visualization of individual medication histories and adherence patterns. The package implements a set of S3 classes and functions consistent with current adherence guidelines and definitions. It allows the computation of different measures of adherence (as defined in the literature, but also several original ones), their publication-quality plotting, the estimation of event duration and time to initiation, the interactive exploration of patient medication history and the real-time estimation of adherence given various parameter settings. It scales from very small datasets stored in flat CSV files to very large databases and from single-thread processing on mid-range consumer laptops to parallel processing on large heterogeneous computing clusters. It exposes a standardized interface allowing it to be used from other programming languages and platforms, such as Python.

URL https://github.com/ddediu/AdhereR

License GPL (>= 2)

LazyData TRUE

Imports lubridate (>= 1.5), parallel (>= 3.0), data.table (>= 1.9), rsvg (>= 1.3), jpeg (>= 0.1), png (>= 0.1), webp (>= 1.0), methods

Depends R (>= 3.0)

Suggests rmarkdown (>= 1.1), knitr (>= 1.20), R.rsp (>= 0.40), base64 (>= 2.0), viridisLite (>= 0.4), AdhereRViz (>= 0.2)

VignetteBuilder knitr, R.rsp

Encoding UTF-8

RoxygenNote 7.2.0

NeedsCompilation no

Repository CRAN

Date/Publication 2022-06-23 12:30:06 UTC

R topics documented:

`callAdhereR` *callAdhereR.*

Description

The function encapsulating all the logics that allows AdhereR to be called from any platform using the generic `shell` mechanism.

Usage

```
callAdhereR(shared.data.directory)
```

Arguments

`shared.data.directory`
: A *string* containing the path to the directory where all the exchanged (shared) data (both input and output) is. `AdhereR` needs read and write access to this directory.

Details

In most cases this should not be done directly by the user, but instead used by an appropriate `wrapper` on the client platform. It allows transparent use of `AdhereR` from virtually any platform or programming language for which an appropriate wrapper is provided. For more details see the vignette describing the included reference `Python 3` wrapper.

Value

This function displays any messages to the console, tries to also write them to the `Adherer-results.txt` file in the `shared.data.directory` directory, and, when finished, forces `R` to quit with a given shell error code:

- `0` The processing ended without major errors;
- `1` General error (hopefully there are messages in the `Adherer-results.txt` file;
- `10` The directory `shared.data.directory` does not exit;
- `11` `AdhereR` does not have read access to the `shared.data.directory` directory;
- `12` `AdhereR` does not have write access to the `shared.data.directory` directory;
- `13` issues with the parameters file `parameters.log`;
- `14` issues with the data file `dataset.csv`;

- 15 plotting issues;
- 16 interactive plotting issues;
- 17 issues exporting the results.

CMA0 *CMA0 constructor.*

Description

Constructs a basic CMA (continuous multiple-interval measures of medication availability/gaps) object.

Usage

```
CMA0(
  data = NULL,
  ID.colname = NA,
  event.date.colname = NA,
  event.duration.colname = NA,
  event.daily.dose.colname = NA,
  medication.class.colname = NA,
  medication.groups = NULL,
  flatten.medication.groups = FALSE,
  medication.groups.colname = ".MED_GROUP_ID",
  carryover.within.obs.window = NA,
  carryover.into.obs.window = NA,
  carry.only.for.same.medication = NA,
  consider.dosage.change = NA,
  followup.window.start = 0,
  followup.window.start.unit = c("days", "weeks", "months", "years")[1],
  followup.window.start.per.medication.group = FALSE,
  followup.window.duration = 365 * 2,
  followup.window.duration.unit = c("days", "weeks", "months", "years")[1],
  observation.window.start = 0,
  observation.window.start.unit = c("days", "weeks", "months", "years")[1],
  observation.window.duration = 365 * 2,
  observation.window.duration.unit = c("days", "weeks", "months", "years")[1],
  date.format = "%m/%d/%Y",
  summary = "Base CMA object",
  suppress.warnings = FALSE,
  suppress.special.argument.checks = FALSE,
  arguments.that.should.not.be.defined = NULL,
  ...
)
```

Arguments

data	A data.frame containing the medication events (prescribing or dispensing) used to compute the CMA. Must contain, at a minimum, the patient unique ID, the event date and duration, and might also contain the daily dosage and medication type (the actual column names are defined in the following four parameters).
ID.colname	A *string*, the name of the column in data containing the unique patient ID, or NA if not defined.
event.date.colname	A *string*, the name of the column in data containing the start date of the event (in the format given in the date.format parameter), or NA if not defined.
event.duration.colname	A *string*, the name of the column in data containing the event duration (in days), or NA if not defined.
event.daily.dose.colname	A *string*, the name of the column in data containing the prescribed daily dose, or NA if not defined.
medication.class.colname	A *string*, the name of the column in data containing the classes/types/groups of medication, or NA if not defined.
medication.groups	A *vector* of characters defining medication groups or the name of a column in data that defines such groups. The names of the vector are the medication group unique names, while the content defines them as logical expressions. While the names can be any string of characters except "}", it is recommended to stick to the rules for defining vector names in R. For example, c("A"="CATEGORY == 'medA'", "AA"="{A} & PERDAY < 4" defines two medication groups: *A* which selects all events of type "medA", and *B* which selects all events already defined by "A" but with a daily dose lower than 4. If NULL, no medication groups are defined. If medication groups are defined, there is one CMA estimate for each group; moreover, there is a special group *__ALL_OTHERS__* automatically defined containing all observations *not* covered by any of the explicitly defined groups.
flatten.medication.groups	*Logical*, if FALSE (the default) then the CMA and event.info components of the object are lists with one medication group per element; otherwise, they are data.frames with an extra column containing the medication group (its name is given by medication.groups.colname).
medication.groups.colname	a *string* (defaults to ".MED_GROUP_ID") giving the name of the column storing the group name when flatten.medication.groups is TRUE.
carryover.within.obs.window	*Logical*, if TRUE consider the carry-over within the observation window, or NA if not defined.
carryover.into.obs.window	*Logical*, if TRUE consider the carry-over from before the starting date of the observation window, or NA if not defined.

carry.only.for.same.medication
: *Logical*, if TRUE the carry-over applies only across medications of the same type, or NA if not defined.

consider.dosage.change
: *Logical*, if TRUE the carry-over is adjusted to reflect changes in dosage, or NA if not defined.

followup.window.start
: If a Date object, it represents the actual start date of the follow-up window; if a *string* it is the name of the column in data containing the start date of the follow-up window either as the numbers of followup.window.start.unit units after the first event (the column must be of type numeric) or as actual dates (in which case the column must be of type Date or a string that conforms to the format specified in date.format); if a *number* it is the number of time units defined in the followup.window.start.unit parameter after the begin of the participant's first event; or NA if not defined.

followup.window.start.unit
: can be either *"days"*, *"weeks"*, *"months"* or *"years"*, and represents the time units that followup.window.start refers to (when a number), or NA if not defined.

followup.window.start.per.medication.group
: a *logical*: if there are medication groups defined and this is TRUE, then the first event considered for the follow-up window start is relative to each medication group separately, otherwise (the default) it is relative to the patient.

followup.window.duration
: either a *number* representing the duration of the follow-up window in the time units given in followup.window.duration.unit, or a *string* giving the column containing these numbers. Should represent a period for which relevant medication events are recorded accurately (e.g. not extend after end of relevant treatment, loss-to-follow-up or change to a health care provider not covered by the database).

followup.window.duration.unit
: can be either *"days"*, *"weeks"*, *"months"* or *"years"*, and represents the time units that followup.window.duration refers to, or NA if not defined.

observation.window.start, observation.window.start.unit, observation.window.duration, observation.wir
: the definition of the observation window (see the follow-up window parameters above for details).

date.format
: A *string* giving the format of the dates used in the data and the other parameters; see the format parameters of the as.Date function for details (NB, this concerns only the dates given as strings and not as Date objects).

summary
: Metadata as a *string*, briefly describing this CMA.

suppress.warnings
: *Logical*, if TRUE don't show any warnings.

suppress.special.argument.checks
: *Logical* parameter for internal use; if FALSE (default) check if the important columns in the data have some of the reserved names, if TRUE this check is not performed.

arguments.that.should.not.be.defined
: a *list* of argument names and pre-defined valuesfor which a warning should be thrown if passed to the function.

... other possible parameters

Details

In most cases this should not be done directly by the user, but it is used internally by the other CMAs.

Value

An S3 object of class CMA0 with the following fields:

- data The actual event (prescribing or dispensing) data, as given by the data parameter.
- ID.colname the name of the column in data containing the unique patient ID, as given by the ID.colname parameter.
- event.date.colname the name of the column in data containing the start date of the event (in the format given in the date.format parameter), as given by the event.date.colname parameter.
- event.duration.colname the name of the column in data containing the event duration (in days), as given by the event.duration.colname parameter.
- event.daily.dose.colname the name of the column in data containing the prescribed daily dose, as given by the event.daily.dose.colname parameter.
- medication.class.colname the name of the column in data containing the classes/types/groups of medication, as given by the medication.class.colname parameter.
- carryover.within.obs.window whether to consider the carry-over within the observation window, as given by the carryover.within.obs.window parameter.
- carryover.into.obs.window whether to consider the carry-over from before the starting date of the observation window, as given by the carryover.into.obs.window parameter.
- carry.only.for.same.medication whether the carry-over applies only across medication of the same type, as given by the carry.only.for.same.medication parameter.
- consider.dosage.change whether the carry-over is adjusted to reflect changes in dosage, as given by the consider.dosage.change parameter.
- followup.window.start the beginning of the follow-up window, as given by the followup.window.start parameter.
- followup.window.start.unit the time unit of the followup.window.start, as given by the followup.window.start.unit parameter.
- followup.window.duration the duration of the follow-up window, as given by the followup.window.duration parameter.
- followup.window.duration.unit the time unit of the followup.window.duration, as given by the followup.window.duration.unit parameter.
- observation.window.start the beginning of the observation window, as given by the observation.window.start parameter.
- observation.window.start.unit the time unit of the observation.window.start, as given by the observation.window.start.unit parameter.
- observation.window.duration the duration of the observation window, as given by the observation.window.duration parameter.

- observation.window.duration.unit the time unit of the observation.window.duration, as given by the observation.window.duration.unit parameter.
- date.format the format of the dates, as given by the date.format parameter.
- summary the metadata, as given by the summary parameter.

Examples

```
cma0 <- CMA0(data=med.events,
             ID.colname="PATIENT_ID",
             event.date.colname="DATE",
             event.duration.colname="DURATION",
             event.daily.dose.colname="PERDAY",
             medication.class.colname="CATEGORY",
             followup.window.start=0,
             followup.window.start.unit="days",
             followup.window.duration=2*365,
             followup.window.duration.unit="days",
             observation.window.start=30,
             observation.window.start.unit="days",
             observation.window.duration=365,
             observation.window.duration.unit="days",
             date.format="%m/%d/%Y",
             summary="Base CMA");
```

CMA1 *CMA1 and CMA3 constructors.*

Description

Constructs a CMA (continuous multiple-interval measures of medication availability/gaps) type 1 or type 3 object.

Usage

```
CMA1(
  data = NULL,
  ID.colname = NA,
  event.date.colname = NA,
  event.duration.colname = NA,
  medication.groups = NULL,
  flatten.medication.groups = FALSE,
  medication.groups.colname = ".MED_GROUP_ID",
  followup.window.start = 0,
  followup.window.start.unit = c("days", "weeks", "months", "years")[1],
  followup.window.start.per.medication.group = FALSE,
  followup.window.duration = 365 * 2,
  followup.window.duration.unit = c("days", "weeks", "months", "years")[1],
  observation.window.start = 0,
```

```
  observation.window.start.unit = c("days", "weeks", "months", "years")[1],
  observation.window.duration = 365 * 2,
  observation.window.duration.unit = c("days", "weeks", "months", "years")[1],
  date.format = "%m/%d/%Y",
  summary = NA,
  event.interval.colname = "event.interval",
  gap.days.colname = "gap.days",
  force.NA.CMA.for.failed.patients = TRUE,
  parallel.backend = c("none", "multicore", "snow", "snow(SOCK)", "snow(MPI)",
    "snow(NWS)")[1],
  parallel.threads = "auto",
  suppress.warnings = FALSE,
  suppress.special.argument.checks = TRUE,
  arguments.that.should.not.be.defined = c(carryover.within.obs.window = FALSE,
    carryover.into.obs.window = FALSE, carry.only.for.same.medication = FALSE,
    consider.dosage.change = FALSE),
  ...
)

CMA3(
  data = NULL,
  ID.colname = NA,
  event.date.colname = NA,
  event.duration.colname = NA,
  medication.groups = NULL,
  flatten.medication.groups = FALSE,
  medication.groups.colname = ".MED_GROUP_ID",
  followup.window.start = 0,
  followup.window.start.unit = c("days", "weeks", "months", "years")[1],
  followup.window.start.per.medication.group = FALSE,
  followup.window.duration = 365 * 2,
  followup.window.duration.unit = c("days", "weeks", "months", "years")[1],
  observation.window.start = 0,
  observation.window.start.unit = c("days", "weeks", "months", "years")[1],
  observation.window.duration = 365 * 2,
  observation.window.duration.unit = c("days", "weeks", "months", "years")[1],
  date.format = "%m/%d/%Y",
  summary = NA,
  event.interval.colname = "event.interval",
  gap.days.colname = "gap.days",
  force.NA.CMA.for.failed.patients = TRUE,
  parallel.backend = c("none", "multicore", "snow", "snow(SOCK)", "snow(MPI)",
    "snow(NWS)")[1],
  parallel.threads = "auto",
  suppress.warnings = FALSE,
  suppress.special.argument.checks = TRUE,
  arguments.that.should.not.be.defined = c(carryover.within.obs.window = FALSE,
    carryover.into.obs.window = FALSE, carry.only.for.same.medication = FALSE,
```

```
    consider.dosage.change = FALSE),
  ...
)
```

Arguments

data	A data.frame containing the events used to compute the CMA. Must contain, at a minimum, the patient unique ID, the event date and duration, and might also contain the daily dosage and medication type (the actual column names are defined in the following four parameters).
ID.colname	A *string*, the name of the column in data containing the unique patient ID; must be present.
event.date.colname	A *string*, the name of the column in data containing the start date of the event (in the format given in the date.format parameter); must be present.
event.duration.colname	A *string*, the name of the column in data containing the event duration (in days); must be present.
medication.groups	A *vector* of characters defining medication groups or the name of a column in data that defines such groups. The names of the vector are the medication group unique names, while the content defines them as logical expressions. While the names can be any string of characters except "}", it is recommended to stick to the rules for defining vector names in R. For example, c("A"="CATEGORY == 'medA'", "AA"="{A} & PERDAY < 4" defines two medication groups: *A* which selects all events of type "medA", and *B* which selects all events already defined by "A" but with a daily dose lower than 4. If NULL, no medication groups are defined. If medication groups are defined, there is one CMA estimate for each group; moreover, there is a special group *__ALL_OTHERS__* automatically defined containing all observations *not* covered by any of the explicitly defined groups.
flatten.medication.groups	*Logical*, if FALSE (the default) then the CMA and event.info components of the object are lists with one medication group per element; otherwise, they are data.frames with an extra column containing the medication group (its name is given by medication.groups.colname).
medication.groups.colname	a *string* (defaults to ".MED_GROUP_ID") giving the name of the column storing the group name when flatten.medication.groups is TRUE.
followup.window.start	If a Date object, it represents the actual start date of the follow-up window; if a *string* it is the name of the column in data containing the start date of the follow-up window either as the numbers of followup.window.start.unit units after the first event (the column must be of type numeric) or as actual dates (in which case the column must be of type Date or a string that conforms to the format specified in date.format); if a *number* it is the number of time units defined in the followup.window.start.unit parameter after the begin of the participant's first event; or NA if not defined.

followup.window.start.unit
: can be either *"days"*, *"weeks"*, *"months"* or *"years"*, and represents the time units that followup.window.start refers to (when a number), or NA if not defined.

followup.window.start.per.medication.group
: a *logical*: if there are medication groups defined and this is TRUE, then the first event considered for the follow-up window start is relative to each medication group separately, otherwise (the default) it is relative to the patient.

followup.window.duration
: either a *number* representing the duration of the follow-up window in the time units given in followup.window.duration.unit, or a *string* giving the column containing these numbers. Should represent a period for which relevant medication events are recorded accurately (e.g. not extend after end of relevant treatment, loss-to-follow-up or change to a health care provider not covered by the database).

followup.window.duration.unit
: can be either *"days"*, *"weeks"*, *"months"* or *"years"*, and represents the time units that followup.window.duration refers to, or NA if not defined.

observation.window.start, observation.window.start.unit, observation.window.duration, observation.wir
: the definition of the observation window (see the follow-up window parameters above for details).

date.format
: A *string* giving the format of the dates used in the data and the other parameters; see the format parameters of the as.Date function for details (NB, this concerns only the dates given as strings and not as Date objects).

summary
: Metadata as a *string*, briefly describing this CMA.

event.interval.colname
: A *string*, the name of a newly-created column storing the number of days between the start of the current event and the start of the next one; the default value "event.interval" should be changed only if there is a naming conflict with a pre-existing "event.interval" column in event.info.

gap.days.colname
: A *string*, the name of a newly-created column storing the number of days when medication was not available (i.e., the "gap days"); the default value "gap.days" should be changed only if there is a naming conflict with a pre-existing "gap.days" column in event.info.

force.NA.CMA.for.failed.patients
: *Logical* describing how the patients for which the CMA estimation fails are treated: if TRUE they are returned with an NA CMA estimate, while for FALSE they are omitted.

parallel.backend
: Can be "none" (the default) for single-threaded execution, "multicore" (using mclapply in package parallel) for multicore processing (NB. not currently implemented on MS Windows and automatically falls back on "snow" on this platform), or "snow", "snow(SOCK)" (equivalent to "snow"), "snow(MPI)" or "snow(NWS)" specifying various types of SNOW clusters (can be on the local machine or more complex setups – please see the documentation of package snow for details; the last two require packages Rmpi and nws, respectively, not automatically installed with AdhereR).

parallel.threads
: Can be "auto" (for parallel.backend == "multicore", defaults to the number of cores in the system as given by options("cores"), while for parallel.backend == "snow", defaults to 2), a strictly positive integer specifying the number of parallel threads, or a more complex specification of the SNOW cluster nodes for parallel.backend == "snow" (see the documentation of package snow for details).

suppress.warnings
: *Logical*, if TRUE don't show any warnings.

suppress.special.argument.checks
: *Logical* parameter for internal use; if FALSE (default) check if the important columns in the data have some of the reserved names, if TRUE this check is not performed.

arguments.that.should.not.be.defined
: a *list* of argument names and pre-defined valuesfor which a warning should be thrown if passed to the function.

...
: other possible parameters

Details

CMA1 considers the total number of days with medication supplied in all medication events in the observation window, excluding the last event. CMA3 is identical to CMA1 except that it is capped at 100%.

The formula is

$$(numberofdayssupplyexcludinglast)/(firsttolastevent)$$

Thus, the durations of all events are added up, possibly resulting in an CMA estimate (much) bigger than 1.0 (100%).

CMA2 and CMA1 differ in the inclusion or not of the last event.

Value

An S3 object of class CMA1 (derived from CMA0) with the following fields:

- data The actual event data, as given by the data parameter.
- ID.colname the name of the column in data containing the unique patient ID, as given by the ID.colname parameter.
- event.date.colname the name of the column in data containing the start date of the event (in the format given in the date.format parameter), as given by the event.date.colname parameter.
- event.duration.colname the name of the column in data containing the event duration (in days), as given by the event.duration.colname parameter.
- event.daily.dose.colname the name of the column in data containing the prescribed daily dose, as given by the event.daily.dose.colname parameter.
- medication.class.colname the name of the column in data containing the classes/types/groups of medication, as given by the medication.class.colname parameter.

- followup.window.start the beginning of the follow-up window, as given by the followup.window.start parameter.
- followup.window.start.unit the time unit of the followup.window.start, as given by the followup.window.start.unit parameter.
- followup.window.duration the duration of the follow-up window, as given by the followup.window.duration parameter.
- followup.window.duration.unit the time unit of the followup.window.duration, as given by the followup.window.duration.unit parameter.
- observation.window.start the beginning of the observation window, as given by the observation.window.start parameter.
- observation.window.start.unit the time unit of the observation.window.start, as given by the observation.window.start.unit parameter.
- observation.window.duration the duration of the observation window, as given by the observation.window.duration parameter.
- observation.window.duration.unit the time unit of the observation.window.duration, as given by the observation.window.duration.unit parameter.
- date.format the format of the dates, as given by the date.format parameter.
- summary the metadata, as given by the summary parameter.
- event.info the data.frame containing the event info (irrelevant for most users; see compute.event.int.gaps for details).
- CMA the data.frame containing the actual CMA estimates for each participant (the ID.colname column).

Please note that if medication.groups are defined and flatten.medication.groups is FALSE, then the CMA and event.info are named lists, each element containing the CMA and event.info corresponding to a single medication group (the element's name), but if flatten.medication.groups is FALSE then they are data.frames with an extra column giving the medication group (the column's name is given by medication.groups.colname).

See Also

CMAs 1 to 8 are described in:

Vollmer, W. M., Xu, M., Feldstein, A., Smith, D., Waterbury, A., & Rand, C. (2012). Comparison of pharmacy-based measures of medication adherence. *BMC Health Services Research*, **12**, 155. doi: 10.1186/1472696312155.

Examples

```
cma1 <- CMA1(data=med.events,
             ID.colname="PATIENT_ID",
             event.date.colname="DATE",
             event.duration.colname="DURATION",
             followup.window.start=30,
             observation.window.start=30,
             observation.window.duration=365,
             date.format="%m/%d/%Y"
```

```
            );
cma3 <- CMA3(data=med.events,
             ID.colname="PATIENT_ID",
             event.date.colname="DATE",
             event.duration.colname="DURATION",
             followup.window.start=30,
             observation.window.start=30,
             observation.window.duration=365,
             date.format="%m/%d/%Y"
            );
```

CMA2 *CMA2 and CMA4 constructors.*

Description

Constructs a CMA (continuous multiple-interval measures of medication availability/gaps) type 2 or type 4 object.

Usage

```
CMA2(
  data = NULL,
  ID.colname = NA,
  event.date.colname = NA,
  event.duration.colname = NA,
  medication.groups = NULL,
  flatten.medication.groups = FALSE,
  medication.groups.colname = ".MED_GROUP_ID",
  followup.window.start = 0,
  followup.window.start.unit = c("days", "weeks", "months", "years")[1],
  followup.window.start.per.medication.group = FALSE,
  followup.window.duration = 365 * 2,
  followup.window.duration.unit = c("days", "weeks", "months", "years")[1],
  observation.window.start = 0,
  observation.window.start.unit = c("days", "weeks", "months", "years")[1],
  observation.window.duration = 365 * 2,
  observation.window.duration.unit = c("days", "weeks", "months", "years")[1],
  date.format = "%m/%d/%Y",
  summary = NA,
  event.interval.colname = "event.interval",
  gap.days.colname = "gap.days",
  force.NA.CMA.for.failed.patients = TRUE,
  parallel.backend = c("none", "multicore", "snow", "snow(SOCK)", "snow(MPI)",
    "snow(NWS)")[1],
  parallel.threads = "auto",
  suppress.warnings = FALSE,
  suppress.special.argument.checks = TRUE,
```

```
  arguments.that.should.not.be.defined = c(carryover.within.obs.window = FALSE,
    carryover.into.obs.window = FALSE, carry.only.for.same.medication = FALSE,
    consider.dosage.change = FALSE),
  ...
)

CMA4(
  data = NULL,
  ID.colname = NA,
  event.date.colname = NA,
  event.duration.colname = NA,
  medication.groups = NULL,
  flatten.medication.groups = FALSE,
  medication.groups.colname = ".MED_GROUP_ID",
  followup.window.start = 0,
  followup.window.start.unit = c("days", "weeks", "months", "years")[1],
  followup.window.start.per.medication.group = FALSE,
  followup.window.duration = 365 * 2,
  followup.window.duration.unit = c("days", "weeks", "months", "years")[1],
  observation.window.start = 0,
  observation.window.start.unit = c("days", "weeks", "months", "years")[1],
  observation.window.duration = 365 * 2,
  observation.window.duration.unit = c("days", "weeks", "months", "years")[1],
  date.format = "%m/%d/%Y",
  summary = NA,
  event.interval.colname = "event.interval",
  gap.days.colname = "gap.days",
  force.NA.CMA.for.failed.patients = TRUE,
  parallel.backend = c("none", "multicore", "snow", "snow(SOCK)", "snow(MPI)",
    "snow(NWS)")[1],
  parallel.threads = "auto",
  suppress.warnings = FALSE,
  suppress.special.argument.checks = TRUE,
  arguments.that.should.not.be.defined = c(carryover.within.obs.window = FALSE,
    carryover.into.obs.window = FALSE, carry.only.for.same.medication = FALSE,
    consider.dosage.change = FALSE),
  ...
)
```

Arguments

`data`	A `data.frame` containing the events used to compute the CMA. Must contain, at a minimum, the patient unique ID, the event date and duration, and might also contain the daily dosage and medication type (the actual column names are defined in the following four parameters).
`ID.colname`	A *string*, the name of the column in `data` containing the unique patient ID; must be present.

`event.date.colname`
: A *string*, the name of the column in `data` containing the start date of the event (in the format given in the `date.format` parameter); must be present.

`event.duration.colname`
: A *string*, the name of the column in `data` containing the event duration (in days); must be present.

`medication.groups`
: A *vector* of characters defining medication groups or the name of a column in `data` that defines such groups. The names of the vector are the medication group unique names, while the content defines them as logical expressions. While the names can be any string of characters except "}", it is recommended to stick to the rules for defining vector names in `R`. For example, `c("A"="CATEGORY == 'medA'", "AA"="{A} & PERDAY < 4"` defines two medication groups: *A* which selects all events of type "medA", and *B* which selects all events already defined by "A" but with a daily dose lower than 4. If `NULL`, no medication groups are defined. If medication groups are defined, there is one CMA estimate for each group; moreover, there is a special group *__ALL_OTHERS__* automatically defined containing all observations *not* covered by any of the explicitly defined groups.

`flatten.medication.groups`
: *Logical*, if `FALSE` (the default) then the `CMA` and `event.info` components of the object are lists with one medication group per element; otherwise, they are `data.frames` with an extra column containing the medication group (its name is given by `medication.groups.colname`).

`medication.groups.colname`
: a *string* (defaults to ".MED_GROUP_ID") giving the name of the column storing the group name when `flatten.medication.groups` is `TRUE`.

`followup.window.start`
: If a `Date` object, it represents the actual start date of the follow-up window; if a *string* it is the name of the column in `data` containing the start date of the follow-up window either as the numbers of `followup.window.start.unit` units after the first event (the column must be of type `numeric`) or as actual dates (in which case the column must be of type `Date` or a string that conforms to the format specified in `date.format`); if a *number* it is the number of time units defined in the `followup.window.start.unit` parameter after the begin of the participant's first event; or `NA` if not defined.

`followup.window.start.unit`
: can be either *"days"*, *"weeks"*, *"months"* or *"years"*, and represents the time units that `followup.window.start` refers to (when a number), or `NA` if not defined.

`followup.window.start.per.medication.group`
: a *logical*: if there are medication groups defined and this is `TRUE`, then the first event considered for the follow-up window start is relative to each medication group separately, otherwise (the default) it is relative to the patient.

`followup.window.duration`
: either a *number* representing the duration of the follow-up window in the time units given in `followup.window.duration.unit`, or a *string* giving the column containing these numbers. Should represent a period for which relevant

medication events are recorded accurately (e.g. not extend after end of relevant treatment, loss-to-follow-up or change to a health care provider not covered by the database).

`followup.window.duration.unit`
: can be either *"days"*, *"weeks"*, *"months"* or *"years"*, and represents the time units that `followup.window.duration` refers to, or `NA` if not defined.

`observation.window.start, observation.window.start.unit, observation.window.duration, observation.wir`
: the definition of the observation window (see the follow-up window parameters above for details).

`date.format`
: A *string* giving the format of the dates used in the `data` and the other parameters; see the `format` parameters of the `as.Date` function for details (NB, this concerns only the dates given as strings and not as `Date` objects).

`summary`
: Metadata as a *string*, briefly describing this CMA.

`event.interval.colname`
: A *string*, the name of a newly-created column storing the number of days between the start of the current event and the start of the next one; the default value "event.interval" should be changed only if there is a naming conflict with a pre-existing "event.interval" column in `event.info`.

`gap.days.colname`
: A *string*, the name of a newly-created column storing the number of days when medication was not available (i.e., the "gap days"); the default value "gap.days" should be changed only if there is a naming conflict with a pre-existing "gap.days" column in `event.info`.

`force.NA.CMA.for.failed.patients`
: *Logical* describing how the patients for which the CMA estimation fails are treated: if `TRUE` they are returned with an `NA` CMA estimate, while for `FALSE` they are omitted.

`parallel.backend`
: Can be "none" (the default) for single-threaded execution, "multicore" (using `mclapply` in package `parallel`) for multicore processing (NB. not currently implemented on MS Windows and automatically falls back on "snow" on this platform), or "snow", "snow(SOCK)" (equivalent to "snow"), "snow(MPI)" or "snow(NWS)" specifying various types of SNOW clusters (can be on the local machine or more complex setups – please see the documentation of package `snow` for details; the last two require packages `Rmpi` and `nws`, respectively, not automatically installed with `AdhereR`).

`parallel.threads`
: Can be "auto" (for `parallel.backend` == "multicore", defaults to the number of cores in the system as given by `options("cores")`, while for `parallel.backend` == "snow", defaults to 2), a strictly positive integer specifying the number of parallel threads, or a more complex specification of the SNOW cluster nodes for `parallel.backend` == "snow" (see the documentation of package `snow` for details).

`suppress.warnings`
: *Logical*, if `TRUE` don't show any warnings.

`suppress.special.argument.checks`
: *Logical* parameter for internal use; if `FALSE` (default) check if the important columns in the `data` have some of the reserved names, if `TRUE` this check is not performed.

`arguments.that.should.not.be.defined`
: a *list* of argument names and pre-defined valuesfor which a warning should be thrown if passed to the function.

`...`
: other possible parameters

Details

`CMA2` considers the total number of days with medication supplied in all medication events in the observation window, including the last event. `CMA4` is identical to `CMA2` except that it is capped at 100%.

The formula is

$$(numberofdayssupplyincludinglastevent)/(firsttolastevent)$$

Thus, the durations of all events are added up, possibly resulting in an CMA estimate (much) bigger than 1.0 (100%)

`CMA2` and `CMA1` differ in the inclusion or not of the last event.

Value

An `S3` object of class `CMA2` (derived from `CMA0`) with the following fields:

- `data` The actual event data, as given by the `data` parameter.
- `ID.colname` the name of the column in `data` containing the unique patient ID, as given by the `ID.colname` parameter.
- `event.date.colname` the name of the column in `data` containing the start date of the event (in the format given in the `date.format` parameter), as given by the `event.date.colname` parameter.
- `event.duration.colname` the name of the column in `data` containing the event duration (in days), as given by the `event.duration.colname` parameter.
- `event.daily.dose.colname` the name of the column in `data` containing the prescribed daily dose, as given by the `event.daily.dose.colname` parameter.
- `medication.class.colname` the name of the column in `data` containing the classes/types/groups of medication, as given by the `medication.class.colname` parameter.
- `followup.window.start` the beginning of the follow-up window, as given by the `followup.window.start` parameter.
- `followup.window.start.unit` the time unit of the `followup.window.start`, as given by the `followup.window.start.unit` parameter.
- `followup.window.duration` the duration of the follow-up window, as given by the `followup.window.duration` parameter.
- `followup.window.duration.unit` the time unit of the `followup.window.duration`, as given by the `followup.window.duration.unit` parameter.

- observation.window.start the beginning of the observation window, as given by the observation.window.start parameter.
- observation.window.start.unit the time unit of the observation.window.start, as given by the observation.window.start.unit parameter.
- observation.window.duration the duration of the observation window, as given by the observation.window.duration parameter.
- observation.window.duration.unit the time unit of the observation.window.duration, as given by the observation.window.duration.unit parameter.
- date.format the format of the dates, as given by the date.format parameter.
- summary the metadata, as given by the summary parameter.
- event.info the data.frame containing the event info (irrelevant for most users; see compute.event.int.gaps for details).
- CMA the data.frame containing the actual CMA estimates for each participant (the ID.colname column).

Please note that if medication.groups are defined, then the CMA and event.info are named lists, each element containing the CMA and event.info corresponding to a single medication group (the element's name).

See Also

CMAs 1 to 8 are defined in:

Vollmer, W. M., Xu, M., Feldstein, A., Smith, D., Waterbury, A., & Rand, C. (2012). Comparison of pharmacy-based measures of medication adherence. *BMC Health Services Research*, **12**, 155. doi: 10.1186/1472696312155.

Examples

```
## Not run:
cma2 <- CMA2(data=med.events,
             ID.colname="PATIENT_ID",
             event.date.colname="DATE",
             event.duration.colname="DURATION",
             followup.window.start=30,
             observation.window.start=30,
             observation.window.duration=365,
             date.format="%m/%d/%Y"
            );
cma4 <- CMA4(data=med.events,
             ID.colname="PATIENT_ID",
             event.date.colname="DATE",
             event.duration.colname="DURATION",
             followup.window.start=30,
             observation.window.start=30,
             observation.window.duration=365,
             date.format="%m/%d/%Y"
            );
## End(Not run)
```

CMA5 *CMA5 constructor.*

Description

Constructs a CMA (continuous multiple-interval measures of medication availability/gaps) type 5 object.

Usage

```
CMA5(
  data = NULL,
  ID.colname = NA,
  event.date.colname = NA,
  event.duration.colname = NA,
  event.daily.dose.colname = NA,
  medication.class.colname = NA,
  medication.groups = NULL,
  flatten.medication.groups = FALSE,
  medication.groups.colname = ".MED_GROUP_ID",
  carry.only.for.same.medication = FALSE,
  consider.dosage.change = FALSE,
  followup.window.start = 0,
  followup.window.start.unit = c("days", "weeks", "months", "years")[1],
  followup.window.start.per.medication.group = FALSE,
  followup.window.duration = 365 * 2,
  followup.window.duration.unit = c("days", "weeks", "months", "years")[1],
  observation.window.start = 0,
  observation.window.start.unit = c("days", "weeks", "months", "years")[1],
  observation.window.duration = 365 * 2,
  observation.window.duration.unit = c("days", "weeks", "months", "years")[1],
  date.format = "%m/%d/%Y",
  summary = NA,
  event.interval.colname = "event.interval",
  gap.days.colname = "gap.days",
  force.NA.CMA.for.failed.patients = TRUE,
  parallel.backend = c("none", "multicore", "snow", "snow(SOCK)", "snow(MPI)",
    "snow(NWS)")[1],
  parallel.threads = "auto",
  suppress.warnings = FALSE,
  suppress.special.argument.checks = TRUE,
  arguments.that.should.not.be.defined = c(carryover.within.obs.window = TRUE,
    carryover.into.obs.window = FALSE),
  ...
)
```

Arguments

data	A data.frame containing the medication events used to compute the CMA. Must contain, at a minimum, the patient unique ID, the event date and duration, and might also contain the daily dosage and medication type (the actual column names are defined in the following four parameters).
ID.colname	A *string*, the name of the column in data containing the unique patient ID; must be present.
event.date.colname	A *string*, the name of the column in data containing the start date of the event (in the format given in the date.format parameter); must be present.
event.duration.colname	A *string*, the name of the column in data containing the event duration (in days); must be present.
event.daily.dose.colname	A *string*, the name of the column in data containing the prescribed daily dose, or NA if not defined.
medication.class.colname	A *string*, the name of the column in data containing the medication type, or NA if not defined.
medication.groups	A *vector* of characters defining medication groups or the name of a column in data that defines such groups. The names of the vector are the medication group unique names, while the content defines them as logical expressions. While the names can be any string of characters except "}", it is recommended to stick to the rules for defining vector names in R. For example, c("A"="CATEGORY == 'medA'", "AA"="{A} & PERDAY < 4" defines two medication groups: *A* which selects all events of type "medA", and *B* which selects all events already defined by "A" but with a daily dose lower than 4. If NULL, no medication groups are defined. If medication groups are defined, there is one CMA estimate for each group; moreover, there is a special group *__ALL_OTHERS__* automatically defined containing all observations *not* covered by any of the explicitly defined groups.
flatten.medication.groups	*Logical*, if FALSE (the default) then the CMA and event.info components of the object are lists with one medication group per element; otherwise, they are data.frames with an extra column containing the medication group (its name is given by medication.groups.colname).
medication.groups.colname	a *string* (defaults to ".MED_GROUP_ID") giving the name of the column storing the group name when flatten.medication.groups is TRUE.
carry.only.for.same.medication	*Logical*, if TRUE, the carry-over applies only across medication of the same type.
consider.dosage.change	*Logical*, if TRUE, the carry-over is adjusted to also reflect changes in dosage.
followup.window.start	If a Date object, it represents the actual start date of the follow-up window; if a *string* it is the name of the column in data containing the start date of

the follow-up window either as the numbers of `followup.window.start.unit` units after the first event (the column must be of type `numeric`) or as actual dates (in which case the column must be of type `Date` or a string that conforms to the format specified in `date.format`); if a *number* it is the number of time units defined in the `followup.window.start.unit` parameter after the begin of the participant's first event; or `NA` if not defined.

`followup.window.start.unit`
: can be either *"days"*, *"weeks"*, *"months"* or *"years"*, and represents the time units that `followup.window.start` refers to (when a number), or `NA` if not defined.

`followup.window.start.per.medication.group`
: a *logical*: if there are medication groups defined and this is `TRUE`, then the first event considered for the follow-up window start is relative to each medication group separately, otherwise (the default) it is relative to the patient.

`followup.window.duration`
: either a *number* representing the duration of the follow-up window in the time units given in `followup.window.duration.unit`, or a *string* giving the column containing these numbers. Should represent a period for which relevant medication events are recorded accurately (e.g. not extend after end of relevant treatment, loss-to-follow-up or change to a health care provider not covered by the database).

`followup.window.duration.unit`
: can be either *"days"*, *"weeks"*, *"months"* or *"years"*, and represents the time units that `followup.window.duration` refers to, or `NA` if not defined.

`observation.window.start, observation.window.start.unit, observation.window.duration, observation.wir`
: the definition of the observation window (see the follow-up window parameters above for details).

`date.format`
: A *string* giving the format of the dates used in the `data` and the other parameters; see the `format` parameters of the `as.Date` function for details (NB, this concerns only the dates given as strings and not as `Date` objects).

`summary`
: Metadata as a *string*, briefly describing this CMA.

`event.interval.colname`
: A *string*, the name of a newly-created column storing the number of days between the start of the current event and the start of the next one; the default value "event.interval" should be changed only if there is a naming conflict with a pre-existing "event.interval" column in `event.info`.

`gap.days.colname`
: A *string*, the name of a newly-created column storing the number of days when medication was not available (i.e., the "gap days"); the default value "gap.days" should be changed only if there is a naming conflict with a pre-existing "gap.days" column in `event.info`.

`force.NA.CMA.for.failed.patients`
: *Logical* describing how the patients for which the CMA estimation fails are treated: if `TRUE` they are returned with an `NA` CMA estimate, while for `FALSE` they are omitted.

`parallel.backend`
: Can be "none" (the default) for single-threaded execution, "multicore" (using `mclapply` in package `parallel`) for multicore processing (NB. not currently

implemented on MS Windows and automatically falls back on "snow" on this platform), or "snow", "snow(SOCK)" (equivalent to "snow"), "snow(MPI)" or "snow(NWS)" specifying various types of SNOW clusters (can be on the local machine or more complex setups – please see the documentation of package `snow` for details; the last two require packages `Rmpi` and `nws`, respectively, not automatically installed with `AdhereR`).

`parallel.threads`
: Can be "auto" (for `parallel.backend` == "multicore", defaults to the number of cores in the system as given by `options("cores")`, while for `parallel.backend` == "snow", defaults to 2), a strictly positive integer specifying the number of parallel threads, or a more complex specification of the SNOW cluster nodes for `parallel.backend` == "snow" (see the documentation of package `snow` for details).

`suppress.warnings`
: *Logical*, if `TRUE` don't show any warnings.

`suppress.special.argument.checks`
: *Logical* parameter for internal use; if `FALSE` (default) check if the important columns in the `data` have some of the reserved names, if `TRUE` this check is not performed.

`arguments.that.should.not.be.defined`
: a *list* of argument names and pre-defined valuesfor which a warning should be thrown if passed to the function.

`...`
: other possible parameters

Details

`CMA5` assumes that, within the observation window, the medication is used as prescribed and new medication is "banked" until needed (oversupply from previous events is used first, followed new medication supply). It computes days of theoretical use by extracting the total number of gap days from the total time interval between the first and the last event, accounting for carry over for all medication events within the observation window. Thus, it accounts for timing within the observation window, and excludes the remaining supply at the start of the last event within the observation window.

The formula is

$$(numberofdaysoftheoreticaluse)/(firsttolastevent)$$

Observations:

- the `carry.only.for.same.medication` parameter controls the transmission of carry-over across medication changes, producing a "standard" `CMA5` (default value is FALSE), and an "alternative" `CMA5b`, respectively;
- the `consider.dosage.change` parameter controls if dosage changes are taken into account, i.e. if set as TRUE and a new medication event has a different daily dosage recommendation, carry-over is recomputed assuming medication use according to the new prescribed dosage (default value is FALSE).

Value

An S3 object of class CMA5 (derived from CMA0) with the following fields:

- data The actual event data, as given by the data parameter.
- ID.colname the name of the column in data containing the unique patient ID, as given by the ID.colname parameter.
- event.date.colname the name of the column in data containing the start date of the event (in the format given in the date.format parameter), as given by the event.date.colname parameter.
- event.duration.colname the name of the column in data containing the event duration (in days), as given by the event.duration.colname parameter.
- event.daily.dose.colname the name of the column in data containing the prescribed daily dose, as given by the event.daily.dose.colname parameter.
- medication.class.colname the name of the column in data containing the classes/types/groups of medication, as given by the medication.class.colname parameter.
- carry.only.for.same.medication whether the carry-over applies only across medication of the same type, as given by the carry.only.for.same.medication parameter.
- consider.dosage.change whether the carry-over is adjusted to reflect changes in dosage, as given by the consider.dosage.change parameter.
- followup.window.start the beginning of the follow-up window, as given by the followup.window.start parameter.
- followup.window.start.unit the time unit of the followup.window.start, as given by the followup.window.start.unit parameter.
- followup.window.duration the duration of the follow-up window, as given by the followup.window.duration parameter.
- followup.window.duration.unit the time unit of the followup.window.duration, as given by the followup.window.duration.unit parameter.
- observation.window.start the beginning of the observation window, as given by the observation.window.start parameter.
- observation.window.start.unit the time unit of the observation.window.start, as given by the observation.window.start.unit parameter.
- observation.window.duration the duration of the observation window, as given by the observation.window.duration parameter.
- observation.window.duration.unit the time unit of the observation.window.duration, as given by the observation.window.duration.unit parameter.
- date.format the format of the dates, as given by the date.format parameter.
- summary the metadata, as given by the summary parameter.
- event.info the data.frame containing the event info (irrelevant for most users; see compute.event.int.gaps for details).
- CMA the data.frame containing the actual CMA estimates for each participant (the ID.colname column).

Please note that if medication.groups are defined, then the CMA and event.info are named lists, each element containing the CMA and event.info corresponding to a single medication group (the element's name).

See Also

CMAs 1 to 8 are defined in:

Vollmer, W. M., Xu, M., Feldstein, A., Smith, D., Waterbury, A., & Rand, C. (2012). Comparison of pharmacy-based measures of medication adherence. *BMC Health Services Research*, **12**, 155. doi: 10.1186/1472696312155.

Examples

```
cma5 <- CMA5(data=med.events,
             ID.colname="PATIENT_ID",
             event.date.colname="DATE",
             event.duration.colname="DURATION",
             event.daily.dose.colname="PERDAY",
             medication.class.colname="CATEGORY",
             carry.only.for.same.medication=FALSE,
             consider.dosage.change=FALSE,
             followup.window.start=30,
             observation.window.start=30,
             observation.window.duration=365,
             date.format="%m/%d/%Y"
            );
```

CMA6 *CMA6 constructor.*

Description

Constructs a CMA (continuous multiple-interval measures of medication availability/gaps) type 6 object.

Usage

```
CMA6(
  data = NULL,
  ID.colname = NA,
  event.date.colname = NA,
  event.duration.colname = NA,
  event.daily.dose.colname = NA,
  medication.class.colname = NA,
  medication.groups = NULL,
  flatten.medication.groups = FALSE,
  medication.groups.colname = ".MED_GROUP_ID",
  carry.only.for.same.medication = FALSE,
  consider.dosage.change = FALSE,
  followup.window.start = 0,
  followup.window.start.unit = c("days", "weeks", "months", "years")[1],
  followup.window.start.per.medication.group = FALSE,
```

```
  followup.window.duration = 365 * 2,
  followup.window.duration.unit = c("days", "weeks", "months", "years")[1],
  observation.window.start = 0,
  observation.window.start.unit = c("days", "weeks", "months", "years")[1],
  observation.window.duration = 365 * 2,
  observation.window.duration.unit = c("days", "weeks", "months", "years")[1],
  date.format = "%m/%d/%Y",
  summary = NA,
  event.interval.colname = "event.interval",
  gap.days.colname = "gap.days",
  force.NA.CMA.for.failed.patients = TRUE,
  parallel.backend = c("none", "multicore", "snow", "snow(SOCK)", "snow(MPI)",
    "snow(NWS)")[1],
  parallel.threads = "auto",
  suppress.warnings = FALSE,
  suppress.special.argument.checks = TRUE,
  arguments.that.should.not.be.defined = c(carryover.within.obs.window = TRUE,
    carryover.into.obs.window = FALSE),
  ...
)
```

Arguments

`data`	A `data.frame` containing the events used to compute the CMA. Must contain, at a minimum, the patient unique ID, the event date and duration, and might also contain the daily dosage and medication type (the actual column names are defined in the following four parameters).
`ID.colname`	A *string*, the name of the column in `data` containing the unique patient ID; must be present.
`event.date.colname`	A *string*, the name of the column in `data` containing the start date of the event (in the format given in the `date.format` parameter); must be present.
`event.duration.colname`	A *string*, the name of the column in `data` containing the event duration (in days); must be present.
`event.daily.dose.colname`	A *string*, the name of the column in `data` containing the prescribed daily dose, or `NA` if not defined.
`medication.class.colname`	A *string*, the name of the column in `data` containing the medication type, or `NA` if not defined.
`medication.groups`	A *vector* of characters defining medication groups or the name of a column in `data` that defines such groups. The names of the vector are the medication group unique names, while the content defines them as logical expressions. While the names can be any string of characters except "}", it is recommended to stick to the rules for defining vector names in R. For example, `c("A"="CATEGORY == 'medA'", "AA"="{A} & PERDAY < 4"` defines two medication groups: *A* which

selects all events of type "medA", and *B* which selects all events already defined by "A" but with a daily dose lower than 4. If NULL, no medication groups are defined. If medication groups are defined, there is one CMA estimate for each group; moreover, there is a special group *__ALL_OTHERS__* automatically defined containing all observations *not* covered by any of the explicitly defined groups.

`flatten.medication.groups`
: *Logical*, if FALSE (the default) then the CMA and event.info components of the object are lists with one medication group per element; otherwise, they are data.frames with an extra column containing the medication group (its name is given by medication.groups.colname).

`medication.groups.colname`
: a *string* (defaults to ".MED_GROUP_ID") giving the name of the column storing the group name when flatten.medication.groups is TRUE.

`carry.only.for.same.medication`
: *Logical*, if TRUE, the carry-over applies only across medication of the same type.

`consider.dosage.change`
: *Logical*, if TRUE, the carry-over is adjusted to also reflect changes in dosage.

`followup.window.start`
: If a Date object, it represents the actual start date of the follow-up window; if a *string* it is the name of the column in data containing the start date of the follow-up window either as the numbers of followup.window.start.unit units after the first event (the column must be of type numeric) or as actual dates (in which case the column must be of type Date or a string that conforms to the format specified in date.format); if a *number* it is the number of time units defined in the followup.window.start.unit parameter after the begin of the participant's first event; or NA if not defined.

`followup.window.start.unit`
: can be either *"days"*, *"weeks"*, *"months"* or *"years"*, and represents the time units that followup.window.start refers to (when a number), or NA if not defined.

`followup.window.start.per.medication.group`
: a *logical*: if there are medication groups defined and this is TRUE, then the first event considered for the follow-up window start is relative to each medication group separately, otherwise (the default) it is relative to the patient.

`followup.window.duration`
: either a *number* representing the duration of the follow-up window in the time units given in followup.window.duration.unit, or a *string* giving the column containing these numbers. Should represent a period for which relevant medication events are recorded accurately (e.g. not extend after end of relevant treatment, loss-to-follow-up or change to a health care provider not covered by the database).

`followup.window.duration.unit`
: can be either *"days"*, *"weeks"*, *"months"* or *"years"*, and represents the time units that followup.window.duration refers to, or NA if not defined.

`observation.window.start, observation.window.start.unit, observation.window.duration, observation.wir`
: the definition of the observation window (see the follow-up window parameters above for details).

`date.format` A *string* giving the format of the dates used in the `data` and the other parameters; see the `format` parameters of the `as.Date` function for details (NB, this concerns only the dates given as strings and not as `Date` objects).

`summary` Metadata as a *string*, briefly describing this CMA.

`event.interval.colname`
A *string*, the name of a newly-created column storing the number of days between the start of the current event and the start of the next one; the default value "event.interval" should be changed only if there is a naming conflict with a pre-existing "event.interval" column in `event.info`.

`gap.days.colname`
A *string*, the name of a newly-created column storing the number of days when medication was not available (i.e., the "gap days"); the default value "gap.days" should be changed only if there is a naming conflict with a pre-existing "gap.days" column in `event.info`.

`force.NA.CMA.for.failed.patients`
Logical describing how the patients for which the CMA estimation fails are treated: if `TRUE` they are returned with an `NA` CMA estimate, while for `FALSE` they are omitted.

`parallel.backend`
Can be "none" (the default) for single-threaded execution, "multicore" (using `mclapply` in package `parallel`) for multicore processing (NB. not currently implemented on MS Windows and automatically falls back on "snow" on this platform), or "snow", "snow(SOCK)" (equivalent to "snow"), "snow(MPI)" or "snow(NWS)" specifying various types of SNOW clusters (can be on the local machine or more complex setups – please see the documentation of package `snow` for details; the last two require packages `Rmpi` and `nws`, respectively, not automatically installed with `AdhereR`).

`parallel.threads`
Can be "auto" (for `parallel.backend` == "multicore", defaults to the number of cores in the system as given by `options("cores")`, while for `parallel.backend` == "snow", defaults to 2), a strictly positive integer specifying the number of parallel threads, or a more complex specification of the SNOW cluster nodes for `parallel.backend` == "snow" (see the documentation of package `snow` for details).

`suppress.warnings`
Logical, if `TRUE` don't show any warnings.

`suppress.special.argument.checks`
Logical parameter for internal use; if `FALSE` (default) check if the important columns in the `data` have some of the reserved names, if `TRUE` this check is not performed.

`arguments.that.should.not.be.defined`
a *list* of argument names and pre-defined valuesfor which a warning should be thrown if passed to the function.

`...` other possible parameters

Details

CMA6 assumes that, within the observation window, the medication is used as prescribed and new medication is "banked" until needed (oversupply from previous events is used first, followed new medication supply). It computes days of theoretical use by extracting the total number of gap days from the total time interval between the first event and the end of the observation window, accounting for carry over for all medication events within the observation window. Thus, it accounts for timing within the observation window, and excludes the remaining supply at the end of the observation window.

The formula is

$$(number of days of theoretical use)/(first event to end of observation window)$$

Observations:

- the carry.only.for.same.medication parameter controls the transmission of carry-over across medication changes, producing a "standard" CMA6 (default value is FALSE), and an "alternative" CMA6b, respectively;
- the consider.dosage.change parameter controls if dosage changes are taken into account, i.e. if set as TRUE and a new medication event has a different daily dosage recommendation, carry-over is recomputed assuming medication use according to the new prescribed dosage (default value is FALSE).

Value

An S3 object of class CMA6 (derived from CMA0) with the following fields:

- data The actual event data, as given by the data parameter.
- ID.colname the name of the column in data containing the unique patient ID, as given by the ID.colname parameter.
- event.date.colname the name of the column in data containing the start date of the event (in the format given in the date.format parameter), as given by the event.date.colname parameter.
- event.duration.colname the name of the column in data containing the event duration (in days), as given by the event.duration.colname parameter.
- event.daily.dose.colname the name of the column in data containing the prescribed daily dose, as given by the event.daily.dose.colname parameter.
- medication.class.colname the name of the column in data containing the classes/types/groups of medication, as given by the medication.class.colname parameter.
- carry.only.for.same.medication whether the carry-over applies only across medication of the same type, as given by the carry.only.for.same.medication parameter.
- consider.dosage.change whether the carry-over is adjusted to reflect changes in dosage, as given by the consider.dosage.change parameter.
- followup.window.start the beginning of the follow-up window, as given by the followup.window.start parameter.
- followup.window.start.unit the time unit of the followup.window.start, as given by the followup.window.start.unit parameter.

- followup.window.duration the duration of the follow-up window, as given by the followup.window.duration parameter.
- followup.window.duration.unit the time unit of the followup.window.duration, as given by the followup.window.duration.unit parameter.
- observation.window.start the beginning of the observation window, as given by the observation.window.start parameter.
- observation.window.start.unit the time unit of the observation.window.start, as given by the observation.window.start.unit parameter.
- observation.window.duration the duration of the observation window, as given by the observation.window.duration parameter.
- observation.window.duration.unit the time unit of the observation.window.duration, as given by the observation.window.duration.unit parameter.
- date.format the format of the dates, as given by the date.format parameter.
- summary the metadata, as given by the summary parameter.
- event.info the data.frame containing the event info (irrelevant for most users; see compute.event.int.gaps for details).
- CMA the data.frame containing the actual CMA estimates for each participant (the ID.colname column).

Please note that if medication.groups are defined, then the CMA and event.info are named lists, each element containing the CMA and event.info corresponding to a single medication group (the element's name).

See Also

CMAs 1 to 8 are defined in:

Vollmer, W. M., Xu, M., Feldstein, A., Smith, D., Waterbury, A., & Rand, C. (2012). Comparison of pharmacy-based measures of medication adherence. *BMC Health Services Research*, **12**, 155. doi: 10.1186/1472696312155.

Examples

```
cma6 <- CMA6(data=med.events,
             ID.colname="PATIENT_ID",
             event.date.colname="DATE",
             event.duration.colname="DURATION",
             event.daily.dose.colname="PERDAY",
             medication.class.colname="CATEGORY",
             carry.only.for.same.medication=FALSE,
             consider.dosage.change=FALSE,
             followup.window.start=30,
             observation.window.start=30,
             observation.window.duration=365,
             date.format="%m/%d/%Y"
            );
```

CMA7 *CMA7 constructor.*

Description

Constructs a CMA (continuous multiple-interval measures of medication availability/gaps) type 7 object.

Usage

```
CMA7(
  data = NULL,
  ID.colname = NA,
  event.date.colname = NA,
  event.duration.colname = NA,
  event.daily.dose.colname = NA,
  medication.class.colname = NA,
  medication.groups = NULL,
  flatten.medication.groups = FALSE,
  medication.groups.colname = ".MED_GROUP_ID",
  carry.only.for.same.medication = FALSE,
  consider.dosage.change = FALSE,
  followup.window.start = 0,
  followup.window.start.unit = c("days", "weeks", "months", "years")[1],
  followup.window.start.per.medication.group = FALSE,
  followup.window.duration = 365 * 2,
  followup.window.duration.unit = c("days", "weeks", "months", "years")[1],
  observation.window.start = 0,
  observation.window.start.unit = c("days", "weeks", "months", "years")[1],
  observation.window.duration = 365 * 2,
  observation.window.duration.unit = c("days", "weeks", "months", "years")[1],
  date.format = "%m/%d/%Y",
  summary = NA,
  event.interval.colname = "event.interval",
  gap.days.colname = "gap.days",
  force.NA.CMA.for.failed.patients = TRUE,
  parallel.backend = c("none", "multicore", "snow", "snow(SOCK)", "snow(MPI)",
    "snow(NWS)")[1],
  parallel.threads = "auto",
  suppress.warnings = FALSE,
  suppress.special.argument.checks = TRUE,
  arguments.that.should.not.be.defined = c(carryover.within.obs.window = TRUE,
    carryover.into.obs.window = TRUE),
  ...
)
```

Arguments

data	A data.frame containing the events used to compute the CMA. Must contain, at a minimum, the patient unique ID, the event date and duration, and might also contain the daily dosage and medication type (the actual column names are defined in the following four parameters).
ID.colname	A *string*, the name of the column in data containing the unique patient ID; must be present.
event.date.colname	A *string*, the name of the column in data containing the start date of the event (in the format given in the date.format parameter); must be present.
event.duration.colname	A *string*, the name of the column in data containing the event duration (in days); must be present.
event.daily.dose.colname	A *string*, the name of the column in data containing the prescribed daily dose, or NA if not defined.
medication.class.colname	A *string*, the name of the column in data containing the medication type, or NA if not defined.
medication.groups	A *vector* of characters defining medication groups or the name of a column in data that defines such groups. The names of the vector are the medication group unique names, while the content defines them as logical expressions. While the names can be any string of characters except "}", it is recommended to stick to the rules for defining vector names in R. For example, c("A"="CATEGORY == 'medA'", "AA"="{A} & PERDAY < 4" defines two medication groups: *A* which selects all events of type "medA", and *B* which selects all events already defined by "A" but with a daily dose lower than 4. If NULL, no medication groups are defined. If medication groups are defined, there is one CMA estimate for each group; moreover, there is a special group *__ALL_OTHERS__* automatically defined containing all observations *not* covered by any of the explicitly defined groups.
flatten.medication.groups	*Logical*, if FALSE (the default) then the CMA and event.info components of the object are lists with one medication group per element; otherwise, they are data.frames with an extra column containing the medication group (its name is given by medication.groups.colname).
medication.groups.colname	a *string* (defaults to ".MED_GROUP_ID") giving the name of the column storing the group name when flatten.medication.groups is TRUE.
carry.only.for.same.medication	*Logical*, if TRUE, the carry-over applies only across medication of the same type.
consider.dosage.change	*Logical*, if TRUE, the carry-over is adjusted to also reflect changes in dosage.
followup.window.start	If a Date object, it represents the actual start date of the follow-up window; if a *string* it is the name of the column in data containing the start date of

the follow-up window either as the numbers of `followup.window.start.unit` units after the first event (the column must be of type `numeric`) or as actual dates (in which case the column must be of type `Date` or a string that conforms to the format specified in `date.format`); if a *number* it is the number of time units defined in the `followup.window.start.unit` parameter after the begin of the participant's first event; or `NA` if not defined.

`followup.window.start.unit`
: can be either *"days"*, *"weeks"*, *"months"* or *"years"*, and represents the time units that `followup.window.start` refers to (when a number), or `NA` if not defined.

`followup.window.start.per.medication.group`
: a *logical*: if there are medication groups defined and this is `TRUE`, then the first event considered for the follow-up window start is relative to each medication group separately, otherwise (the default) it is relative to the patient.

`followup.window.duration`
: either a *number* representing the duration of the follow-up window in the time units given in `followup.window.duration.unit`, or a *string* giving the column containing these numbers. Should represent a period for which relevant medication events are recorded accurately (e.g. not extend after end of relevant treatment, loss-to-follow-up or change to a health care provider not covered by the database).

`followup.window.duration.unit`
: can be either *"days"*, *"weeks"*, *"months"* or *"years"*, and represents the time units that `followup.window.duration` refers to, or `NA` if not defined.

`observation.window.start, observation.window.start.unit, observation.window.duration, observation.wir`
: the definition of the observation window (see the follow-up window parameters above for details).

`date.format`
: A *string* giving the format of the dates used in the `data` and the other parameters; see the `format` parameters of the `as.Date` function for details (NB, this concerns only the dates given as strings and not as `Date` objects).

`summary`
: Metadata as a *string*, briefly describing this CMA.

`event.interval.colname`
: A *string*, the name of a newly-created column storing the number of days between the start of the current event and the start of the next one; the default value "event.interval" should be changed only if there is a naming conflict with a pre-existing "event.interval" column in `event.info`.

`gap.days.colname`
: A *string*, the name of a newly-created column storing the number of days when medication was not available (i.e., the "gap days"); the default value "gap.days" should be changed only if there is a naming conflict with a pre-existing "gap.days" column in `event.info`.

`force.NA.CMA.for.failed.patients`
: *Logical* describing how the patients for which the CMA estimation fails are treated: if `TRUE` they are returned with an `NA` CMA estimate, while for `FALSE` they are omitted.

`parallel.backend`
: Can be "none" (the default) for single-threaded execution, "multicore" (using `mclapply` in package `parallel`) for multicore processing (NB. not currently

implemented on MS Windows and automatically falls back on "snow" on this platform), or "snow", "snow(SOCK)" (equivalent to "snow"), "snow(MPI)" or "snow(NWS)" specifying various types of SNOW clusters (can be on the local machine or more complex setups – please see the documentation of package `snow` for details; the last two require packages `Rmpi` and `nws`, respectively, not automatically installed with `AdhereR`).

`parallel.threads`
: Can be "auto" (for `parallel.backend` == "multicore", defaults to the number of cores in the system as given by `options("cores")`, while for `parallel.backend` == "snow", defaults to 2), a strictly positive integer specifying the number of parallel threads, or a more complex specification of the SNOW cluster nodes for `parallel.backend` == "snow" (see the documentation of package `snow` for details).

`suppress.warnings`
: *Logical*, if `TRUE` don't show any warnings.

`suppress.special.argument.checks`
: *Logical* parameter for internal use; if `FALSE` (default) check if the important columns in the `data` have some of the reserved names, if `TRUE` this check is not performed.

`arguments.that.should.not.be.defined`
: a *list* of argument names and pre-defined valuesfor which a warning should be thrown if passed to the function.

`...`
: other possible parameters

Details

`CMA7` assumes that, within and before the observation window, the medication is used as prescribed and new medication is "banked" until needed (oversupply from previous events is used first, followed new medication supply). It computes days of theoretical use by extracting the total number of gap days from the total time interval between the start and the end of the observation window, accounting for carry over for all medication events within and before the observation window. All medication events in the follow up window before observation window are considered for carry-over calculation. Thus, it accounts for timing within and before the observation window, and excludes the remaining supply at the end of the observation window.

The formula is

$$(number of days of theoretical use)/(start to end of observation window)$$

Observations:

- the `carry.only.for.same.medication` parameter controls the transmission of carry-over across medication changes, producing a "standard" `CMA7` (default value is FALSE), and an "alternative" `CMA7b`, respectively;
- the `consider.dosage.change` parameter controls if dosage changes are taken into account, i.e. if set as TRUE and a new medication event has a different daily dosage recommendation, carry-over is recomputed assuming medication use according to the new prescribed dosage (default value is FALSE).

Value

An S3 object of class CMA7 (derived from CMA0) with the following fields:

- data The actual event data, as given by the data parameter.
- ID.colname the name of the column in data containing the unique patient ID, as given by the ID.colname parameter.
- event.date.colname the name of the column in data containing the start date of the event (in the format given in the date.format parameter), as given by the event.date.colname parameter.
- event.duration.colname the name of the column in data containing the event duration (in days), as given by the event.duration.colname parameter.
- event.daily.dose.colname the name of the column in data containing the prescribed daily dose, as given by the event.daily.dose.colname parameter.
- medication.class.colname the name of the column in data containing the classes/types/groups of medication, as given by the medication.class.colname parameter.
- carry.only.for.same.medication whether the carry-over applies only across medication of the same type, as given by the carry.only.for.same.medication parameter.
- consider.dosage.change whether the carry-over is adjusted to reflect changes in dosage, as given by the consider.dosage.change parameter.
- followup.window.start the beginning of the follow-up window, as given by the followup.window.start parameter.
- followup.window.start.unit the time unit of the followup.window.start, as given by the followup.window.start.unit parameter.
- followup.window.duration the duration of the follow-up window, as given by the followup.window.duration parameter.
- followup.window.duration.unit the time unit of the followup.window.duration, as given by the followup.window.duration.unit parameter.
- observation.window.start the beginning of the observation window, as given by the observation.window.start parameter.
- observation.window.start.unit the time unit of the observation.window.start, as given by the observation.window.start.unit parameter.
- observation.window.duration the duration of the observation window, as given by the observation.window.duration parameter.
- observation.window.duration.unit the time unit of the observation.window.duration, as given by the observation.window.duration.unit parameter.
- date.format the format of the dates, as given by the date.format parameter.
- summary the metadata, as given by the summary parameter.
- event.info the data.frame containing the event info (irrelevant for most users; see compute.event.int.gaps for details).
- CMA the data.frame containing the actual CMA estimates for each participant (the ID.colname column).

Please note that if medication.groups are defined, then the CMA and event.info are named lists, each element containing the CMA and event.info corresponding to a single medication group (the element's name).

See Also

CMAs 1 to 8 are defined in:

Vollmer, W. M., Xu, M., Feldstein, A., Smith, D., Waterbury, A., & Rand, C. (2012). Comparison of pharmacy-based measures of medication adherence. *BMC Health Services Research*, **12**, 155. doi: 10.1186/1472696312155.

Examples

```
cma7 <- CMA7(data=med.events,
             ID.colname="PATIENT_ID",
             event.date.colname="DATE",
             event.duration.colname="DURATION",
             event.daily.dose.colname="PERDAY",
             medication.class.colname="CATEGORY",
             carry.only.for.same.medication=FALSE,
             consider.dosage.change=FALSE,
             followup.window.start=30,
             observation.window.start=30,
             observation.window.duration=365,
             date.format="%m/%d/%Y"
            );
```

CMA8 *CMA8 constructor.*

Description

Constructs a CMA (continuous multiple-interval measures of medication availability/gaps) type 8 object.

Usage

```
CMA8(
  data = NULL,
  ID.colname = NA,
  event.date.colname = NA,
  event.duration.colname = NA,
  event.daily.dose.colname = NA,
  medication.class.colname = NA,
  medication.groups = NULL,
  flatten.medication.groups = FALSE,
  medication.groups.colname = ".MED_GROUP_ID",
  carry.only.for.same.medication = FALSE,
  consider.dosage.change = FALSE,
  followup.window.start = 0,
  followup.window.start.unit = c("days", "weeks", "months", "years")[1],
  followup.window.start.per.medication.group = FALSE,
```

```
  followup.window.duration = 365 * 2,
  followup.window.duration.unit = c("days", "weeks", "months", "years")[1],
  observation.window.start = 0,
  observation.window.start.unit = c("days", "weeks", "months", "years")[1],
  observation.window.duration = 365 * 2,
  observation.window.duration.unit = c("days", "weeks", "months", "years")[1],
  date.format = "%m/%d/%Y",
  summary = NA,
  event.interval.colname = "event.interval",
  gap.days.colname = "gap.days",
  force.NA.CMA.for.failed.patients = TRUE,
  parallel.backend = c("none", "multicore", "snow", "snow(SOCK)", "snow(MPI)",
    "snow(NWS)")[1],
  parallel.threads = "auto",
  suppress.warnings = FALSE,
  suppress.special.argument.checks = TRUE,
  arguments.that.should.not.be.defined = c(carryover.within.obs.window = TRUE,
    carryover.into.obs.window = TRUE),
  ...
)
```

Arguments

`data`	A `data.frame` containing the events used to compute the CMA. Must contain, at a minimum, the patient unique ID, the event date and duration, and might also contain the daily dosage and medication type (the actual column names are defined in the following four parameters).
`ID.colname`	A *string*, the name of the column in `data` containing the unique patient ID; must be present.
`event.date.colname`	A *string*, the name of the column in `data` containing the start date of the event (in the format given in the `date.format` parameter); must be present.
`event.duration.colname`	A *string*, the name of the column in `data` containing the event duration (in days); must be present.
`event.daily.dose.colname`	A *string*, the name of the column in `data` containing the prescribed daily dose, or `NA` if not defined.
`medication.class.colname`	A *string*, the name of the column in `data` containing the medication type, or `NA` if not defined.
`medication.groups`	A *vector* of characters defining medication groups or the name of a column in `data` that defines such groups. The names of the vector are the medication group unique names, while the content defines them as logical expressions. While the names can be any string of characters except "}", it is recommended to stick to the rules for defining vector names in `R`. For example, `c("A"="CATEGORY == 'medA'", "AA"="{A} & PERDAY < 4"` defines two medication groups: *A* which

selects all events of type "medA", and *B* which selects all events already defined by "A" but with a daily dose lower than 4. If NULL, no medication groups are defined. If medication groups are defined, there is one CMA estimate for each group; moreover, there is a special group *__ALL_OTHERS__* automatically defined containing all observations *not* covered by any of the explicitly defined groups.

`flatten.medication.groups`
: *Logical*, if FALSE (the default) then the CMA and event.info components of the object are lists with one medication group per element; otherwise, they are data.frames with an extra column containing the medication group (its name is given by medication.groups.colname).

`medication.groups.colname`
: a *string* (defaults to ".MED_GROUP_ID") giving the name of the column storing the group name when flatten.medication.groups is TRUE.

`carry.only.for.same.medication`
: *Logical*, if TRUE, the carry-over applies only across medication of the same type.

`consider.dosage.change`
: *Logical*, if TRUE, the carry-over is adjusted to also reflect changes in dosage.

`followup.window.start`
: If a Date object, it represents the actual start date of the follow-up window; if a *string* it is the name of the column in data containing the start date of the follow-up window either as the numbers of followup.window.start.unit units after the first event (the column must be of type numeric) or as actual dates (in which case the column must be of type Date or a string that conforms to the format specified in date.format); if a *number* it is the number of time units defined in the followup.window.start.unit parameter after the begin of the participant's first event; or NA if not defined.

`followup.window.start.unit`
: can be either *"days"*, *"weeks"*, *"months"* or *"years"*, and represents the time units that followup.window.start refers to (when a number), or NA if not defined.

`followup.window.start.per.medication.group`
: a *logical*: if there are medication groups defined and this is TRUE, then the first event considered for the follow-up window start is relative to each medication group separately, otherwise (the default) it is relative to the patient.

`followup.window.duration`
: either a *number* representing the duration of the follow-up window in the time units given in followup.window.duration.unit, or a *string* giving the column containing these numbers. Should represent a period for which relevant medication events are recorded accurately (e.g. not extend after end of relevant treatment, loss-to-follow-up or change to a health care provider not covered by the database).

`followup.window.duration.unit`
: can be either *"days"*, *"weeks"*, *"months"* or *"years"*, and represents the time units that followup.window.duration refers to, or NA if not defined.

`observation.window.start, observation.window.start.unit, observation.window.duration, observation.wir`
: the definition of the observation window (see the follow-up window parameters above for details).

`date.format` A *string* giving the format of the dates used in the `data` and the other parameters; see the `format` parameters of the `as.Date` function for details (NB, this concerns only the dates given as strings and not as `Date` objects).

`summary` Metadata as a *string*, briefly describing this CMA.

`event.interval.colname`
A *string*, the name of a newly-created column storing the number of days between the start of the current event and the start of the next one; the default value "event.interval" should be changed only if there is a naming conflict with a pre-existing "event.interval" column in `event.info`.

`gap.days.colname`
A *string*, the name of a newly-created column storing the number of days when medication was not available (i.e., the "gap days"); the default value "gap.days" should be changed only if there is a naming conflict with a pre-existing "gap.days" column in `event.info`.

`force.NA.CMA.for.failed.patients`
Logical describing how the patients for which the CMA estimation fails are treated: if `TRUE` they are returned with an `NA` CMA estimate, while for `FALSE` they are omitted.

`parallel.backend`
Can be "none" (the default) for single-threaded execution, "multicore" (using `mclapply` in package `parallel`) for multicore processing (NB. not currently implemented on MS Windows and automatically falls back on "snow" on this platform), or "snow", "snow(SOCK)" (equivalent to "snow"), "snow(MPI)" or "snow(NWS)" specifying various types of SNOW clusters (can be on the local machine or more complex setups – please see the documentation of package `snow` for details; the last two require packages `Rmpi` and `nws`, respectively, not automatically installed with `AdhereR`).

`parallel.threads`
Can be "auto" (for `parallel.backend` == "multicore", defaults to the number of cores in the system as given by `options("cores")`, while for `parallel.backend` == "snow", defaults to 2), a strictly positive integer specifying the number of parallel threads, or a more complex specification of the SNOW cluster nodes for `parallel.backend` == "snow" (see the documentation of package `snow` for details).

`suppress.warnings`
Logical, if `TRUE` don't show any warnings.

`suppress.special.argument.checks`
Logical parameter for internal use; if `FALSE` (default) check if the important columns in the `data` have some of the reserved names, if `TRUE` this check is not performed.

`arguments.that.should.not.be.defined`
a *list* of argument names and pre-defined valuesfor which a warning should be thrown if passed to the function.

`...` other possible parameters

Details

CMA8 is similar to CMA6 in that it assumes that, within the observation window, the medication is used as prescribed and new medication is "banked" until needed (oversupply from previous events is used first, followed new medication supply). Unlike CMA6 it accounts for carry-over from before the window - but in a different way from CMA7: by adding a time lag at the start of the observation window equal to the duration of carry-over from before. It is designed for situations when an event with a hypothesized causal effect on adherence happens at the start of the observation window (e.g. enrolment in an intervention study); in this case, it may be that the existing supply is not part of the relationship under study (e.g. it delays the actual start of the study for that participant) and needs to be excluded by shortening the time interval examined. The end of the observation window remains the same. Thus, CMA8 computes days of theoretical use by extracting the total number of gap days from the total time interval between the lagged start and the end of the observation window, accounting for carry over for all medication events within the observation window. All medication events in the follow up window before observation window are considered for carry-over calculation. Thus, as CMA7, it accounts for timing within the observation window, as well as before (different adjustment than CMA7), and excludes the remaining supply at the end of the observation window.

The formula is

$$(number of days of theoretical use)/(lagged start to end of observation window)$$

Observations:

- the carry.only.for.same.medication parameter controls the transmission of carry-over across medication changes, producing a "standard" CMA8 (default value is FALSE), and an "alternative" CMA8b, respectively;
- the consider.dosage.change parameter controls if dosage changes are taken into account, i.e. if set as TRUE and a new medication event has a different daily dosage recommendation, carry-over is recomputed assuming medication use according to the new prescribed dosage (default value is FALSE).

Value

An S3 object of class CMA8 (derived from CMA0) with the following fields:

- data The actual event data, as given by the data parameter.
- ID.colname the name of the column in data containing the unique patient ID, as given by the ID.colname parameter.
- event.date.colname the name of the column in data containing the start date of the event (in the format given in the date.format parameter), as given by the event.date.colname parameter.
- event.duration.colname the name of the column in data containing the event duration (in days), as given by the event.duration.colname parameter.
- event.daily.dose.colname the name of the column in data containing the prescribed daily dose, as given by the event.daily.dose.colname parameter.
- medication.class.colname the name of the column in data containing the classes/types/groups of medication, as given by the medication.class.colname parameter.

- carry.only.for.same.medication whether the carry-over applies only across medication of the same type, as given by the carry.only.for.same.medication parameter.
- consider.dosage.change whether the carry-over is adjusted to reflect changes in dosage, as given by the consider.dosage.change parameter.
- followup.window.start the beginning of the follow-up window, as given by the followup.window.start parameter.
- followup.window.start.unit the time unit of the followup.window.start, as given by the followup.window.start.unit parameter.
- followup.window.duration the duration of the follow-up window, as given by the followup.window.duration parameter.
- followup.window.duration.unit the time unit of the followup.window.duration, as given by the followup.window.duration.unit parameter.
- observation.window.start the beginning of the observation window, as given by the observation.window.start parameter.
- observation.window.start.unit the time unit of the observation.window.start, as given by the observation.window.start.unit parameter.
- observation.window.duration the duration of the observation window, as given by the observation.window.duration parameter.
- observation.window.duration.unit the time unit of the observation.window.duration, as given by the observation.window.duration.unit parameter.
- date.format the format of the dates, as given by the date.format parameter.
- summary the metadata, as given by the summary parameter.
- event.info the data.frame containing the event info (irrelevant for most users; see compute.event.int.gaps for details).
- CMA the data.frame containing the actual CMA estimates for each participant (the ID.colname column).

Please note that if medication.groups are defined, then the CMA and event.info are named lists, each element containing the CMA and event.info corresponding to a single medication group (the element's name).

See Also

CMAs 1 to 8 are defined in:

Vollmer, W. M., Xu, M., Feldstein, A., Smith, D., Waterbury, A., & Rand, C. (2012). Comparison of pharmacy-based measures of medication adherence. *BMC Health Services Research*, **12**, 155. doi: 10.1186/1472696312155.

Examples

```
cma8 <- CMA8(data=med.events,
             ID.colname="PATIENT_ID",
             event.date.colname="DATE",
             event.duration.colname="DURATION",
             event.daily.dose.colname="PERDAY",
```

```
 medication.class.colname="CATEGORY",
 carry.only.for.same.medication=FALSE,
 consider.dosage.change=FALSE,
 followup.window.start=30,
 observation.window.start=30,
 observation.window.duration=365,
 date.format="%m/%d/%Y"
);
```

CMA9 *CMA9 constructor.*

Description

Constructs a CMA (continuous multiple-interval measures of medication availability/gaps) type 9 object.

Usage

```
CMA9(
  data = NULL,
  ID.colname = NA,
  event.date.colname = NA,
  event.duration.colname = NA,
  event.daily.dose.colname = NA,
  medication.class.colname = NA,
  medication.groups = NULL,
  flatten.medication.groups = FALSE,
  medication.groups.colname = ".MED_GROUP_ID",
  carry.only.for.same.medication = FALSE,
  consider.dosage.change = FALSE,
  followup.window.start = 0,
  followup.window.start.unit = c("days", "weeks", "months", "years")[1],
  followup.window.start.per.medication.group = FALSE,
  followup.window.duration = 365 * 2,
  followup.window.duration.unit = c("days", "weeks", "months", "years")[1],
  observation.window.start = 0,
  observation.window.start.unit = c("days", "weeks", "months", "years")[1],
  observation.window.duration = 365 * 2,
  observation.window.duration.unit = c("days", "weeks", "months", "years")[1],
  date.format = "%m/%d/%Y",
  summary = NA,
  event.interval.colname = "event.interval",
  gap.days.colname = "gap.days",
  force.NA.CMA.for.failed.patients = TRUE,
  parallel.backend = c("none", "multicore", "snow", "snow(SOCK)", "snow(MPI)",
    "snow(NWS)")[1],
  parallel.threads = "auto",
```

```
  suppress.warnings = FALSE,
  suppress.special.argument.checks = TRUE,
  arguments.that.should.not.be.defined = c(carryover.within.obs.window = TRUE,
    carryover.into.obs.window = TRUE),
  ...
)
```

Arguments

`data`	A `data.frame` containing the events used to compute the CMA. Must contain, at a minimum, the patient unique ID, the event date and duration, and might also contain the daily dosage and medication type (the actual column names are defined in the following four parameters).
`ID.colname`	A *string*, the name of the column in `data` containing the unique patient ID; must be present.
`event.date.colname`	A *string*, the name of the column in `data` containing the start date of the event (in the format given in the `date.format` parameter); must be present.
`event.duration.colname`	A *string*, the name of the column in `data` containing the event duration (in days); must be present.
`event.daily.dose.colname`	A *string*, the name of the column in `data` containing the prescribed daily dose, or `NA` if not defined.
`medication.class.colname`	A *string*, the name of the column in `data` containing the medication type, or `NA` if not defined.
`medication.groups`	A *vector* of characters defining medication groups or the name of a column in `data` that defines such groups. The names of the vector are the medication group unique names, while the content defines them as logical expressions. While the names can be any string of characters except "}", it is recommended to stick to the rules for defining vector names in `R`. For example, `c("A"="CATEGORY == 'medA'", "AA"="{A} & PERDAY < 4"` defines two medication groups: *A* which selects all events of type "medA", and *B* which selects all events already defined by "A" but with a daily dose lower than 4. If `NULL`, no medication groups are defined. If medication groups are defined, there is one CMA estimate for each group; moreover, there is a special group *__ALL_OTHERS__* automatically defined containing all observations *not* covered by any of the explicitly defined groups.
`flatten.medication.groups`	*Logical*, if `FALSE` (the default) then the `CMA` and `event.info` components of the object are lists with one medication group per element; otherwise, they are `data.frames` with an extra column containing the medication group (its name is given by `medication.groups.colname`).
`medication.groups.colname`	a *string* (defaults to ".MED_GROUP_ID") giving the name of the column storing the group name when `flatten.medication.groups` is `TRUE`.

`carry.only.for.same.medication`
: *Logical*, if `TRUE`, the carry-over applies only across medication of the same type.

`consider.dosage.change`
: *Logical*, if `TRUE`, the carry-over is adjusted to also reflect changes in dosage.

`followup.window.start`
: If a `Date` object, it represents the actual start date of the follow-up window; if a *string* it is the name of the column in `data` containing the start date of the follow-up window either as the numbers of `followup.window.start.unit` units after the first event (the column must be of type `numeric`) or as actual dates (in which case the column must be of type `Date` or a string that conforms to the format specified in `date.format`); if a *number* it is the number of time units defined in the `followup.window.start.unit` parameter after the begin of the participant's first event; or `NA` if not defined.

`followup.window.start.unit`
: can be either *"days"*, *"weeks"*, *"months"* or *"years"*, and represents the time units that `followup.window.start` refers to (when a number), or `NA` if not defined.

`followup.window.start.per.medication.group`
: a *logical*: if there are medication groups defined and this is `TRUE`, then the first event considered for the follow-up window start is relative to each medication group separately, otherwise (the default) it is relative to the patient.

`followup.window.duration`
: either a *number* representing the duration of the follow-up window in the time units given in `followup.window.duration.unit`, or a *string* giving the column containing these numbers. Should represent a period for which relevant medication events are recorded accurately (e.g. not extend after end of relevant treatment, loss-to-follow-up or change to a health care provider not covered by the database).

`followup.window.duration.unit`
: can be either *"days"*, *"weeks"*, *"months"* or *"years"*, and represents the time units that `followup.window.duration` refers to, or `NA` if not defined.

`observation.window.start, observation.window.start.unit, observation.window.duration, observation.wir`
: the definition of the observation window (see the follow-up window parameters above for details).

`date.format`
: A *string* giving the format of the dates used in the `data` and the other parameters; see the `format` parameters of the `as.Date` function for details (NB, this concerns only the dates given as strings and not as `Date` objects).

`summary`
: Metadata as a *string*, briefly describing this CMA.

`event.interval.colname`
: A *string*, the name of a newly-created column storing the number of days between the start of the current event and the start of the next one; the default value "event.interval" should be changed only if there is a naming conflict with a pre-existing "event.interval" column in `event.info`.

`gap.days.colname`
: A *string*, the name of a newly-created column storing the number of days when medication was not available (i.e., the "gap days"); the default value "gap.days" should be changed only if there is a naming conflict with a pre-existing "gap.days" column in `event.info`.

force.NA.CMA.for.failed.patients
: *Logical* describing how the patients for which the CMA estimation fails are treated: if TRUE they are returned with an NA CMA estimate, while for FALSE they are omitted.

parallel.backend
: Can be "none" (the default) for single-threaded execution, "multicore" (using mclapply in package parallel) for multicore processing (NB. not currently implemented on MS Windows and automatically falls back on "snow" on this platform), or "snow", "snow(SOCK)" (equivalent to "snow"), "snow(MPI)" or "snow(NWS)" specifying various types of SNOW clusters (can be on the local machine or more complex setups – please see the documentation of package snow for details; the last two require packages Rmpi and nws, respectively, not automatically installed with AdhereR).

parallel.threads
: Can be "auto" (for parallel.backend == "multicore", defaults to the number of cores in the system as given by options("cores"), while for parallel.backend == "snow", defaults to 2), a strictly positive integer specifying the number of parallel threads, or a more complex specification of the SNOW cluster nodes for parallel.backend == "snow" (see the documentation of package snow for details).

suppress.warnings
: *Logical*, if TRUE don't show any warnings.

suppress.special.argument.checks
: *Logical* parameter for internal use; if FALSE (default) check if the important columns in the data have some of the reserved names, if TRUE this check is not performed.

arguments.that.should.not.be.defined
: a *list* of argument names and pre-defined valuesfor which a warning should be thrown if passed to the function.

...
: other possible parameters

Details

CMA9 is similar to CMA7 and CMA8 in that it accounts for carry-over within and before the observation window assuming that new medication is "banked" until needed (oversupply from previous events is used first, followed new medication supply). Yet, unlike these previous CMAs, it does not assume the medication is used as prescribed; in longitudinal studies with multiple CMA measures, this assumption may introduce additional variation in CMA estimates depending on when the observation window starts in relation to the previous medication event. A shorter time distance from the previous event (and longer to the first event in the observation window) results in higher values even if the number of gap days is the same, and it may also be that the patient has had a similar use pattern for that time interval, rather than perfect adherence followed by no medication use. CMA9 applies a different adjustment: it computes a ratio of days supply over each interval between two prescriptions and considers this applies for each day of that interval, up to 100% (moving oversupply to the next event interval). All medication events in the follow up window before observation window are considered for carry-over calculation. The last interval ends at the end of the follow-up window. Thus, it accounts for timing within the observation window, as well as before (but differently from CMA7 and CMA8), and excludes the remaining supply at the end of the observation window, if any.

The formula is

$(number of days in the observation window, each weighted by the ratio of days supply applicable to their event interval)/($

Observations:

- the `carry.only.for.same.medication` parameter controls the transmission of carry-over across medication changes, producing a "standard" `CMA7` (default value is FALSE), and an "alternative" `CMA7b`, respectively;
- the `consider.dosage.change` parameter controls if dosage changes are taken into account, i.e. if set as TRUE and a new medication event has a different daily dosage recommendation, carry-over is recomputed assuming medication use according to the new prescribed dosage (default value is FALSE).

Value

An `S3` object of class `CMA9` (derived from `CMA0`) with the following fields:

- `data` The actual event data, as given by the `data` parameter.
- `ID.colname` the name of the column in `data` containing the unique patient ID, as given by the `ID.colname` parameter.
- `event.date.colname` the name of the column in `data` containing the start date of the event (in the format given in the `date.format` parameter), as given by the `event.date.colname` parameter.
- `event.duration.colname` the name of the column in `data` containing the event duration (in days), as given by the `event.duration.colname` parameter.
- `event.daily.dose.colname` the name of the column in `data` containing the prescribed daily dose, as given by the `event.daily.dose.colname` parameter.
- `medication.class.colname` the name of the column in `data` containing the classes/types/groups of medication, as given by the `medication.class.colname` parameter.
- `carry.only.for.same.medication` whether the carry-over applies only across medication of the same type, as given by the `carry.only.for.same.medication` parameter.
- `consider.dosage.change` whether the carry-over is adjusted to reflect changes in dosage, as given by the `consider.dosage.change` parameter.
- `followup.window.start` the beginning of the follow-up window, as given by the `followup.window.start` parameter.
- `followup.window.start.unit` the time unit of the `followup.window.start`, as given by the `followup.window.start.unit` parameter.
- `followup.window.duration` the duration of the follow-up window, as given by the `followup.window.duration` parameter.
- `followup.window.duration.unit` the time unit of the `followup.window.duration`, as given by the `followup.window.duration.unit` parameter.
- `observation.window.start` the beginning of the observation window, as given by the `observation.window.start` parameter.
- `observation.window.start.unit` the time unit of the `observation.window.start`, as given by the `observation.window.start.unit` parameter.

- observation.window.duration the duration of the observation window, as given by the observation.window.duration parameter.
- observation.window.duration.unit the time unit of the observation.window.duration, as given by the observation.window.duration.unit parameter.
- date.format the format of the dates, as given by the date.format parameter.
- summary the metadata, as given by the summary parameter.
- event.info the data.frame containing the event info (irrelevant for most users; see compute.event.int.gaps for details).
- CMA the data.frame containing the actual CMA estimates for each participant (the ID.colname column).

Please note that if medication.groups are defined, then the CMA and event.info are named lists, each element containing the CMA and event.info corresponding to a single medication group (the element's name).

Examples

```
cma9 <- CMA9(data=med.events,
             ID.colname="PATIENT_ID",
             event.date.colname="DATE",
             event.duration.colname="DURATION",
             event.daily.dose.colname="PERDAY",
             medication.class.colname="CATEGORY",
             carry.only.for.same.medication=FALSE,
             consider.dosage.change=FALSE,
             followup.window.start=30,
             observation.window.start=30,
             observation.window.duration=365,
             date.format="%m/%d/%Y"
            );
```

CMA_per_episode *CMA_per_episode constructor.*

Description

Applies a given CMA to each treatment episode and constructs a CMA_per_episode object.

Usage

```
CMA_per_episode(
  CMA.to.apply,
  data,
  treat.epi = NULL,
  ID.colname = NA,
  event.date.colname = NA,
  event.duration.colname = NA,
```

```
  event.daily.dose.colname = NA,
  medication.class.colname = NA,
  medication.groups = NULL,
  flatten.medication.groups = FALSE,
  medication.groups.colname = ".MED_GROUP_ID",
  carry.only.for.same.medication = NA,
  consider.dosage.change = NA,
  medication.change.means.new.treatment.episode = TRUE,
  dosage.change.means.new.treatment.episode = FALSE,
  maximum.permissible.gap = 180,
 maximum.permissible.gap.unit = c("days", "weeks", "months", "years", "percent")[1],
  maximum.permissible.gap.append.to.episode = FALSE,
  followup.window.start = 0,
  followup.window.start.unit = c("days", "weeks", "months", "years")[1],
  followup.window.start.per.medication.group = FALSE,
  followup.window.duration = 365 * 2,
  followup.window.duration.unit = c("days", "weeks", "months", "years")[1],
  observation.window.start = 0,
  observation.window.start.unit = c("days", "weeks", "months", "years")[1],
  observation.window.duration = 365 * 2,
  observation.window.duration.unit = c("days", "weeks", "months", "years")[1],
  return.inner.event.info = FALSE,
  date.format = "%m/%d/%Y",
  summary = "CMA per treatment episode",
  event.interval.colname = "event.interval",
  gap.days.colname = "gap.days",
  force.NA.CMA.for.failed.patients = TRUE,
  return.mapping.events.episodes = FALSE,
  parallel.backend = c("none", "multicore", "snow", "snow(SOCK)", "snow(MPI)",
    "snow(NWS)")[1],
  parallel.threads = "auto",
  suppress.warnings = FALSE,
  suppress.special.argument.checks = TRUE,
  ...
)
```

Arguments

`CMA.to.apply`	A *string* giving the name of the CMA function (1 to 9) that will be computed for each treatment episode.
`data`	A `data.frame` containing the events (prescribing or dispensing) used to compute the CMA. Must contain, at a minimum, the patient unique ID, the event date and duration, and might also contain the daily dosage and medication type (the actual column names are defined in the following four parameters).
`treat.epi`	A `data.frame` containing the treatment episodes. Must contain the patient ID (as given in `ID.colname`), the episode unique ID (increasing sequentially, `episode.ID`), the episode start date (`episode.start`), the episode duration in days (`episode.duration`), and the episode end date (`episode.end`).

`ID.colname` A *string*, the name of the column in `data` containing the unique patient ID; must be present.

`event.date.colname`
A *string*, the name of the column in `data` containing the start date of the event (in the format given in the `date.format` parameter); must be present.

`event.duration.colname`
A *string*, the name of the column in `data` containing the event duration (in days); must be present.

`event.daily.dose.colname`
A *string*, the name of the column in `data` containing the prescribed daily dose, or `NA` if not defined.

`medication.class.colname`
A *string*, the name of the column in `data` containing the medication type, or `NA` if not defined.

`medication.groups`
A *vector* of characters defining medication groups or the name of a column in `data` that defines such groups. The names of the vector are the medication group unique names, while the content defines them as logical expressions. While the names can be any string of characters except "}", it is recommended to stick to the rules for defining vector names in `R`. For example, `c("A"="CATEGORY == 'medA'", "AA"="{A} & PERDAY < 4"` defines two medication groups: *A* which selects all events of type "medA", and *B* which selects all events already defined by "A" but with a daily dose lower than 4. If `NULL`, no medication groups are defined. If medication groups are defined, there is one CMA estimate for each group; moreover, there is a special group *__ALL_OTHERS__* automatically defined containing all observations *not* covered by any of the explicitly defined groups.

`flatten.medication.groups`
Logical, if `FALSE` (the default) then the `CMA` and `event.info` components of the object are lists with one medication group per element; otherwise, they are `data.frames` with an extra column containing the medication group (its name is given by `medication.groups.colname`).

`medication.groups.colname`
a *string* (defaults to ".MED_GROUP_ID") giving the name of the column storing the group name when `flatten.medication.groups` is `TRUE`.

`carry.only.for.same.medication`
Logical, if `TRUE`, the carry-over applies only across medication of the same type; valid only for CMAs 5 to 9, in which case it is coupled (i.e., the same value is used for computing the treatment episodes and the CMA on each treatment episode).

`consider.dosage.change`
Logical, if `TRUE`, the carry-over is adjusted to also reflect changes in dosage; valid only for CMAs 5 to 9, in which case it is coupled (i.e., the same value is used for computing the treatment episodes and the CMA on each treatment episode).

`medication.change.means.new.treatment.episode`
Logical, should a change in medication automatically start a new treatment episode?

dosage.change.means.new.treatment.episode
: *Logical*, should a change in dosage automatically start a new treatment episode?

maximum.permissible.gap
: The *number* of units given by maximum.permissible.gap.unit representing the maximum duration of permissible gaps between treatment episodes (can also be a percent, see maximum.permissible.gap.unit for details).

maximum.permissible.gap.unit
: can be either *"days"*, *"weeks"*, *"months"*, *"years"* or *"percent"*, and represents the time units that maximum.permissible.gap refers to; if *percent*, then maximum.permissible.gap is interpreted as a percent (can be greater than 100%) of the duration of the current prescription.

maximum.permissible.gap.append.to.episode
: a *logical* value specifying of the maximum.permissible.gap should be append at the end of an episode with a gap larger than the maximum.permissible.gap; FALSE (the default) mean no addition, while TRUE mean that the full maximum.permissible.gap is added.

followup.window.start
: If a Date object, it represents the actual start date of the follow-up window; if a *string* it is the name of the column in data containing the start date of the follow-up window either as the numbers of followup.window.start.unit units after the first event (the column must be of type numeric) or as actual dates (in which case the column must be of type Date or a string that conforms to the format specified in date.format); if a *number* it is the number of time units defined in the followup.window.start.unit parameter after the begin of the participant's first event; or NA if not defined.

followup.window.start.unit
: can be either *"days"*, *"weeks"*, *"months"* or *"years"*, and represents the time units that followup.window.start refers to (when a number), or NA if not defined.

followup.window.start.per.medication.group
: a *logical*: if there are medication groups defined and this is TRUE, then the first event considered for the follow-up window start is relative to each medication group separately, otherwise (the default) it is relative to the patient.

followup.window.duration
: either a *number* representing the duration of the follow-up window in the time units given in followup.window.duration.unit, or a *string* giving the column containing these numbers. Should represent a period for which relevant medication events are recorded accurately (e.g. not extend after end of relevant treatment, loss-to-follow-up or change to a health care provider not covered by the database).

followup.window.duration.unit
: can be either *"days"*, *"weeks"*, *"months"* or *"years"*, and represents the time units that followup.window.duration refers to, or NA if not defined.

observation.window.start, observation.window.start.unit, observation.window.duration, observation.wir
: the definition of the observation window (see the follow-up window parameters above for details).

return.inner.event.info
: *Logical* specifying if the function should also return the event.info for all the individual events in each sliding window; by default it is FALSE as this informa-

tion is useful only in very specific cases (e.g., plotting the event intervals) and adds a small but non-negligible computational overhead.

`date.format` A *string* giving the format of the dates used in the `data` and the other parameters; see the `format` parameters of the `as.Date` function for details (NB, this concerns only the dates given as strings and not as `Date` objects).

`summary` Metadata as a *string*, briefly describing this CMA.

`event.interval.colname`
A *string*, the name of a newly-created column storing the number of days between the start of the current event and the start of the next one; the default value "event.interval" should be changed only if there is a naming conflict with a pre-existing "event.interval" column in `event.info`.

`gap.days.colname`
A *string*, the name of a newly-created column storing the number of days when medication was not available (i.e., the "gap days"); the default value "gap.days" should be changed only if there is a naming conflict with a pre-existing "gap.days" column in `event.info`.

`force.NA.CMA.for.failed.patients`
Logical describing how the patients for which the CMA estimation fails are treated: if `TRUE` they are returned with an `NA` CMA estimate, while for `FALSE` they are omitted.

`return.mapping.events.episodes`
A *Logical*, if `TRUE` then the mapping between events and episodes is returned as an extra component `mapping.episodes.to.events`, which is a `data.table` giving, for each episode, the events that belong to it (an event is given by its row number in the `data`). Please note that the episodes returned are specific to the particular simple CMA used, and should preferentially used over those returned by `compute.treatment.episodes()`. This component can also be accessed using the `getEventsToEpisodesMapping()` function.

`parallel.backend`
Can be "none" (the default) for single-threaded execution, "multicore" (using `mclapply` in package `parallel`) for multicore processing (NB. not currently implemented on MS Windows and automatically falls back on "snow" on this platform), or "snow", "snow(SOCK)" (equivalent to "snow"), "snow(MPI)" or "snow(NWS)" specifying various types of SNOW clusters (can be on the local machine or more complex setups – please see the documentation of package `snow` for details; the last two require packages `Rmpi` and `nws`, respectively, not automatically installed with `AdhereR`).

`parallel.threads`
Can be "auto" (for `parallel.backend` == "multicore", defaults to the number of cores in the system as given by `options("cores")`, while for `parallel.backend` == "snow", defaults to 2), a strictly positive integer specifying the number of parallel threads, or a more complex specification of the SNOW cluster nodes for `parallel.backend` == "snow" (see the documentation of package `snow` for details).

`suppress.warnings`
Logical, if `TRUE` don't show any warnings.

suppress.special.argument.checks
: *Logical* parameter for internal use; if FALSE (default) check if the important columns in the data have some of the reserved names, if TRUE this check is not performed.

...
: other possible parameters

Details

CMA_per_episode first identifies the treatment episodes for the whole follo-up window (using the compute.treatment.episodes function), and then computes the given "simple" CMA for each treatment episode that intersects with the observation window. NB: the CMA is computed for the period of the episode that is part of the observations window; thus, if an episode starts earlier or ends later than the observation window, CMA will be computed for a section of that episode. Thus, as opposed to the "simple" CMAs 1 to 9, it returns a set of CMAs, with possibly more than one element.

It is highly similar to CMA_sliding_window which computes a CMA for a set of sliding windows.

Value

An S3 object of class CMA_per_episode with the following fields:

- data The actual event data, as given by the data parameter.
- ID.colname the name of the column in data containing the unique patient ID, as given by the ID.colname parameter.
- event.date.colname the name of the column in data containing the start date of the event (in the format given in the date.format parameter), as given by the event.date.colname parameter.
- event.duration.colname the name of the column in data containing the event duration (in days), as given by the event.duration.colname parameter.
- event.daily.dose.colname the name of the column in data containing the prescribed daily dose, as given by the event.daily.dose.colname parameter.
- medication.class.colname the name of the column in data containing the classes/types/groups of medication, as given by the medication.class.colname parameter.
- carry.only.for.same.medication whether the carry-over applies only across medication of the same type, as given by the carry.only.for.same.medication parameter.
- consider.dosage.change whether the carry-over is adjusted to reflect changes in dosage, as given by the consider.dosage.change parameter.
- followup.window.start the beginning of the follow-up window, as given by the followup.window.start parameter.
- followup.window.start.unit the time unit of the followup.window.start, as given by the followup.window.start.unit parameter.
- followup.window.duration the duration of the follow-up window, as given by the followup.window.duration parameter.
- followup.window.duration.unit the time unit of the followup.window.duration, as given by the followup.window.duration.unit parameter.

- observation.window.start the beginning of the observation window, as given by the observation.window.start parameter.
- observation.window.start.unit the time unit of the observation.window.start, as given by the observation.window.start.unit parameter.
- observation.window.duration the duration of the observation window, as given by the observation.window.duration parameter.
- observation.window.duration.unit the time unit of the observation.window.duration, as given by the observation.window.duration.unit parameter.
- date.format the format of the dates, as given by the date.format parameter.
- summary the metadata, as given by the summary parameter.
- event.info the data.frame containing the event info (irrelevant for most users; see compute.event.int.gaps for details).
- computed.CMA the class name of the computed CMA.
- CMA the data.frame containing the actual CMA estimates for each participant (the ID.colname column) and treatment episode, with columns:
  - ID.colname the patient ID as given by the ID.colname parameter.
  - episode.ID the unique treatment episode ID (within patients).
  - episode.start the treatment episode's start date (as a Date object).
  - end.episode.gap.days the corresponding gap days of the last event in this episode.
  - episode.duration the treatment episode's duration in days.
  - episode.end the treatment episode's end date (as a Date object).
  - CMA the treatment episode's estimated CMA.

Please note that if medication.groups are defined, then the CMA and event.info are named lists, each element containing the CMA and event.info corresponding to a single medication group (the element's name).

See Also

CMA_sliding_window is very similar, computing a "simple" CMA for each of a set of same-size sliding windows. The "simple" CMAs that can be computed comprise CMA1, CMA2, CMA3, CMA4, CMA5, CMA6, CMA7, CMA8, CMA9, as well as user-defined classes derived from CMA0 that have a CMA component giving the estimated CMA per patient as a data.frame. If return.mapping.events.episodes is TRUE, then this also has a component mapping.episodes.to.events that gives the mapping between episodes and events as a data.table with the following columns:

- patid the patient ID.
- episode.ID the episode unique ID (increasing sequentially).
- event.index.in.data the event given by its row number in the data.

Examples

```
## Not run:
cmaE <- CMA_per_episode(CMA="CMA1",
                        data=med.events,
                        ID.colname="PATIENT_ID",
```

```
                        event.date.colname="DATE",
                        event.duration.colname="DURATION",
                        event.daily.dose.colname="PERDAY",
                        medication.class.colname="CATEGORY",
                        carry.only.for.same.medication=FALSE,
                        consider.dosage.change=FALSE,
                        followup.window.start=0,
                        observation.window.start=0,
                        observation.window.duration=365,
                        date.format="%m/%d/%Y"
                       );
## End(Not run)
```

CMA_polypharmacy *CMA constructor for polypharmacy.*

Description

Constructs a CMA (continuous multiple-interval measures of medication availability/gaps) object for polypharmacy.

Usage

```
CMA_polypharmacy(
  data = data,
  medication.groups = medication.class.colname,
  CMA.to.apply = NA,
  aggregate.first = TRUE,
  aggregation.method = NA,
  aggregation.method.arguments = NA,
  thresholds = NA,
  ID.colname = NA,
  event.date.colname = NA,
  event.duration.colname = NA,
  event.daily.dose.colname = NA,
  medication.class.colname = NA,
  carry.only.for.same.medication = NA,
  consider.dosage.change = NA,
  followup.window.start = 0,
  followup.window.start.unit = c("days", "weeks", "months", "years")[1],
  followup.window.duration = 365 * 2,
  followup.window.duration.unit = c("days", "weeks", "months", "years")[1],
  observation.window.start = 0,
  observation.window.start.unit = c("days", "weeks", "months", "years")[1],
  observation.window.duration = 365 * 2,
  observation.window.duration.unit = c("days", "weeks", "months", "years")[1],
  date.format = "%m/%d/%Y",
  summary = "CMA for polypharmacy",
```

```
  force.NA.CMA.for.failed.patients = TRUE,
  parallel.backend = c("none", "multicore", "snow", "snow(SOCK)", "snow(MPI)",
    "snow(NWS)")[1],
  parallel.threads = "auto",
  suppress.warnings = FALSE,
  ...
)
```

Arguments

`data`	A `data.frame` containing the events (prescribing or dispensing) used to compute the CMA. Must contain, at a minimum, the patient unique ID, the event date and duration, medication type, and might also contain the daily dosage (the actual column names are defined in the following four parameters).
`medication.groups`	A *string* with the name of the column containing the medication groups. If multiple medication classes should belong to the same treatment group, they can be differentiated here (important to investigate treatment switches)
`CMA.to.apply`	A *string* giving the name of the CMA function (1 to 9) that will be computed for each treatment group.
`aggregate.first`	*Logical*, if `TRUE`, aggregate across treatment groups before summarizing over time during OW.
`aggregation.method`	A *string* giving the name of the function to aggregate CMA values of medication group, or `NA` to return only raw CMA estimates per medication group. Accepts summary functions such as "mean", "sd", "var", "min", "max", and "median". Custom functions are possible as long as they take a numeric vector as input and return a single numeric value.
`aggregation.method.arguments`	optional, A *named list* of additional arguments to the function given in `aggregation method`, e.g. `na.rm = TRUE`.
`thresholds`	optional, a *number* to apply as threshold between aggregation and summarizing.
`ID.colname`	A *string*, the name of the column in `data` containing the medication type. Defaults to `medication.class.colname`.
`event.date.colname`	A *string*, the name of the column in `data` containing the start date of the event (in the format given in the `date.format` parameter); must be present.
`event.duration.colname`	A *string*, the name of the column in `data` containing the event duration (in days); must be present.
`event.daily.dose.colname`	A *string*, the name of the column in `data` containing the prescribed daily dose, or `NA` if not defined.
`medication.class.colname`	A *string*, the name of the column in `data` containing the medication type, or `NA` if not defined.

carry.only.for.same.medication
: *Logical*, if TRUE, the carry-over applies only across medication of the same type; valid only for CMAs 5 to 9, in which case it is coupled (i.e., the same value is used for computing the treatment episodes and the CMA on each treatment episode).

consider.dosage.change
: *Logical*, if TRUE, the carry-over is adjusted to also reflect changes in dosage; valid only for CMAs 5 to 9, in which case it is coupled (i.e., the same value is used for computing the treatment episodes and the CMA on each treatment episode).

followup.window.start
: If a Date object, it represents the actual start date of the follow-up window; if a *string* it is the name of the column in data containing the start date of the follow-up window either as the numbers of followup.window.start.unit units after the first event (the column must be of type numeric) or as actual dates (in which case the column must be of type Date); if a *number* it is the number of time units defined in the followup.window.start.unit parameter after the begin of the participant's first event; or NA if not defined.

followup.window.start.unit
: can be either *"days"*, *"weeks"*, *"months"* or *"years"*, and represents the time units that followup.window.start refers to (when a number), or NA if not defined.

followup.window.duration
: either a *number* representing the duration of the follow-up window in the time units given in followup.window.duration.unit, or a *string* giving the column containing these numbers. Should represent a period for which relevant medication events are recorded accurately (e.g. not extend after end of relevant treatment, loss-to-follow-up or change to a health care provider not covered by the database).

followup.window.duration.unit
: can be either *"days"*, *"weeks"*, *"months"* or *"years"*, and represents the time units that followup.window.duration refers to, or NA if not defined.

observation.window.start, observation.window.start.unit, observation.window.duration, observation.wir
: the definition of the observation window (see the follow-up window parameters above for details). Can be defined separately for each patient and treatment group.

date.format
: A *string* giving the format of the dates used in the data and the other parameters; see the format parameters of the as.Date function for details (NB, this concerns only the dates given as strings and not as Date objects).

summary
: Metadata as a *string*, briefly describing this CMA.

force.NA.CMA.for.failed.patients
: *Logical* describing how the patients for which the CMA estimation fails are treated: if TRUE they are returned with an NA CMA estimate, while for FALSE they are omitted.

parallel.backend
: Can be "none" (the default) for single-threaded execution, "multicore" (using mclapply in package parallel) for multicore processing (NB. not currently implemented on MS Windows and automatically falls back on "snow" on this

platform), or "snow", "snow(SOCK)" (equivalent to "snow"), "snow(MPI)" or "snow(NWS)" specifying various types of SNOW clusters (can be on the local machine or more complex setups – please see the documentation of package snow for details; the last two require packages Rmpi and nws, respectively, not automatically installed with AdhereR).

parallel.threads
: Can be "auto" (for parallel.backend == "multicore", defaults to the number of cores in the system as given by options("cores"), while for parallel.backend == "snow", defaults to 2), a strictly positive integer specifying the number of parallel threads, or a more complex specification of the SNOW cluster nodes for parallel.backend == "snow" (see the documentation of package snow for details).

suppress.warnings
: *Logical*, if TRUE don't show any warnings.

...
: other possible parameters

Value

An S3 object of class CMA_polypharmacy with the following fields:

- data The actual event data, as given by the data parameter.
- ID.colname the name of the column in data containing the unique patient ID, as given by the ID.colname parameter.
- event.date.colname the name of the column in data containing the start date of the event (in the format given in the date.format parameter), as given by the event.date.colname parameter.
- event.duration.colname the name of the column in data containing the event duration (in days), as given by the event.duration.colname parameter.
- event.daily.dose.colname the name of the column in data containing the prescribed daily dose, as given by the event.daily.dose.colname parameter.
- medication.class.colname the name of the column in data containing the classes/types/groups of medication, as given by the medication.class.colname parameter.
- carry.only.for.same.medication whether the carry-over applies only across medication of the same type, as given by the carry.only.for.same.medication parameter.
- consider.dosage.change whether the carry-over is adjusted to reflect changes in dosage, as given by the consider.dosage.change parameter.
- followup.window.start the beginning of the follow-up window, as given by the followup.window.start parameter.
- followup.window.start.unit the time unit of the followup.window.start, as given by the followup.window.start.unit parameter.
- followup.window.duration the duration of the follow-up window, as given by the followup.window.duration parameter.
- followup.window.duration.unit the time unit of the followup.window.duration, as given by the followup.window.duration.unit parameter.

- observation.window.start the beginning of the observation window, as given by the observation.window.start parameter.
- observation.window.start.unit the time unit of the observation.window.start, as given by the observation.window.start.unit parameter.
- observation.window.duration the duration of the observation window, as given by the observation.window.duration parameter.
- observation.window.duration.unit the time unit of the observation.window.duration, as given by the observation.window.duration.unit parameter.
- date.format the format of the dates, as given by the date.format parameter.
- summary the metadata, as given by the summary parameter.
- event.info the data.frame containing the event info (irrelevant for most users; see compute.event.int.gaps for details).
- aggregation.method the aggregation method to combine CMA values from different groups.
- computed.CMA the class name of the computed CMA.
- medication.groups a data.frame with medication groups and classes
- CMA the data.frame containing the actual CMA estimates for each participant (the ID.colname column) and sometimes treatment group, with columns:
  - ID.colname the patient ID as given by the ID.colname parameter.
  - medication.groups only when no aggregation method is used (aggregation.method = NA); the treatment group as given by the medication.groups parameter.
  - CMA the treatment episode's estimated CMA.

Examples

```
## Not run:
CMA_PP <- CMA_polypharmacy(data = med.events.pp,
medication.groups = med.groups,
CMA.to.apply = "CMA7",
aggregate.first = TRUE, # aggregate before summarizing
aggregation.method = "mean", # compute mean of CMAs
aggregation.method.arguments = list(na.rm = TRUE), # remove NA's during calculation
thresholds = NA, # don't apply threshold
ID.colname="PATIENT_ID",
event.date.colname="DATE",
event.duration.colname="DURATION",
event.daily.dose.colname="PERDAY",
medication.class.colname="CATEGORY",
followup.window.start=0,
observation.window.start=180,
observation.window.duration=365,
carry.only.for.same.medication = TRUE);
## End(Not run)
```

CMA_sliding_window *CMA_sliding_window constructor.*

Description

Applies a given CMA to each sliding window and constructs a CMA_sliding_window object.

Usage

```
CMA_sliding_window(
  CMA.to.apply,
  data,
  ID.colname = NA,
  event.date.colname = NA,
  event.duration.colname = NA,
  event.daily.dose.colname = NA,
  medication.class.colname = NA,
  medication.groups = NULL,
  flatten.medication.groups = FALSE,
  medication.groups.colname = ".MED_GROUP_ID",
  carry.only.for.same.medication = NA,
  consider.dosage.change = NA,
  followup.window.start = 0,
  followup.window.start.unit = c("days", "weeks", "months", "years")[1],
  followup.window.start.per.medication.group = FALSE,
  followup.window.duration = 365 * 2,
  followup.window.duration.unit = c("days", "weeks", "months", "years")[1],
  observation.window.start = 0,
  observation.window.start.unit = c("days", "weeks", "months", "years")[1],
  observation.window.duration = 365 * 2,
  observation.window.duration.unit = c("days", "weeks", "months", "years")[1],
  sliding.window.start = 0,
  sliding.window.start.unit = c("days", "weeks", "months", "years")[1],
  sliding.window.duration = 90,
  sliding.window.duration.unit = c("days", "weeks", "months", "years")[1],
  sliding.window.step.duration = 30,
  sliding.window.step.unit = c("days", "weeks", "months", "years")[1],
  sliding.window.no.steps = NA,
  return.inner.event.info = FALSE,
  date.format = "%m/%d/%Y",
  summary = "CMA per sliding window",
  event.interval.colname = "event.interval",
  gap.days.colname = "gap.days",
  force.NA.CMA.for.failed.patients = TRUE,
  return.mapping.events.sliding.window = FALSE,
  parallel.backend = c("none", "multicore", "snow", "snow(SOCK)", "snow(MPI)",
    "snow(NWS)")[1],
```

```
  parallel.threads = "auto",
  suppress.warnings = FALSE,
  suppress.special.argument.checks = FALSE,
  ...
)
```

Arguments

`CMA.to.apply`	A *string* giving the name of the CMA function (1 to 9) that will be computed for each treatment episode.
`data`	A `data.frame` containing the events used to compute the CMA. Must contain, at a minimum, the patient unique ID, the event date and duration, and might also contain the daily dosage and medication type (the actual column names are defined in the following four parameters).
`ID.colname`	A *string*, the name of the column in `data` containing the unique patient ID; must be present.
`event.date.colname`	A *string*, the name of the column in `data` containing the start date of the event (in the format given in the `date.format` parameter); must be present.
`event.duration.colname`	A *string*, the name of the column in `data` containing the event duration (in days); must be present.
`event.daily.dose.colname`	A *string*, the name of the column in `data` containing the prescribed daily dose, or `NA` if not defined.
`medication.class.colname`	A *string*, the name of the column in `data` containing the medication type, or `NA` if not defined.
`medication.groups`	A *vector* of characters defining medication groups or the name of a column in `data` that defines such groups. The names of the vector are the medication group unique names, while the content defines them as logical expressions. While the names can be any string of characters except "}", it is recommended to stick to the rules for defining vector names in R. For example, `c("A"="CATEGORY == 'medA'", "AA"="{A} & PERDAY < 4"` defines two medication groups: *A* which selects all events of type "medA", and *B* which selects all events already defined by "A" but with a daily dose lower than 4. If `NULL`, no medication groups are defined. If medication groups are defined, there is one CMA estimate for each group; moreover, there is a special group *__ALL_OTHERS__* automatically defined containing all observations *not* covered by any of the explicitly defined groups.
`flatten.medication.groups`	*Logical*, if `FALSE` (the default) then the `CMA` and `event.info` components of the object are lists with one medication group per element; otherwise, they are `data.frames` with an extra column containing the medication group (its name is given by `medication.groups.colname`).

`medication.groups.colname`
: a *string* (defaults to ".MED_GROUP_ID") giving the name of the column storing the group name when `flatten.medication.groups` is `TRUE`.

`carry.only.for.same.medication`
: *Logical*, if `TRUE`, the carry-over applies only across medication of the same type.

`consider.dosage.change`
: *Logical*, if `TRUE`, the carry-over is adjusted to also reflect changes in dosage.

`followup.window.start`
: If a `Date` object, it represents the actual start date of the follow-up window; if a *string* it is the name of the column in `data` containing the start date of the follow-up window either as the numbers of `followup.window.start.unit` units after the first event (the column must be of type `numeric`) or as actual dates (in which case the column must be of type `Date` or a string that conforms to the format specified in `date.format`); if a *number* it is the number of time units defined in the `followup.window.start.unit` parameter after the begin of the participant's first event; or `NA` if not defined.

`followup.window.start.unit`
: can be either *"days"*, *"weeks"*, *"months"* or *"years"*, and represents the time units that `followup.window.start` refers to (when a number), or `NA` if not defined.

`followup.window.start.per.medication.group`
: a *logical*: if there are medication groups defined and this is `TRUE`, then the first event considered for the follow-up window start is relative to each medication group separately, otherwise (the default) it is relative to the patient.

`followup.window.duration`
: either a *number* representing the duration of the follow-up window in the time units given in `followup.window.duration.unit`, or a *string* giving the column containing these numbers. Should represent a period for which relevant medication events are recorded accurately (e.g. not extend after end of relevant treatment, loss-to-follow-up or change to a health care provider not covered by the database).

`followup.window.duration.unit`
: can be either *"days"*, *"weeks"*, *"months"* or *"years"*, and represents the time units that `followup.window.duration` refers to, or `NA` if not defined.

`observation.window.start, observation.window.start.unit, observation.window.duration, observation.wir`
: the definition of the observation window (see the follow-up window parameters above for details).

`sliding.window.start, sliding.window.start.unit, sliding.window.duration, sliding.window.duration.uni`
: the definition of the first sliding window (see the follow-up window parameters above for details).

`sliding.window.step.duration, sliding.window.step.unit`
: if not missing (`NA`), these give the step (or "jump") to the right of the sliding window in time units.

`sliding.window.no.steps`
: a *integer* specifying the desired number of sliding windows to cover the observation window (if possible); trumps `sliding.window.step.duration` and `sliding.window.step.unit`.

`return.inner.event.info`
: *Logical* specifying if the function should also return the event.info for all the individual events in each sliding window; by default it is `FALSE` as this information is useful only in very specific cases (e.g., plotting the event intervals) and adds a small but non-negligible computational overhead.

`date.format`
: A *string* giving the format of the dates used in the `data` and the other parameters; see the `format` parameters of the `as.Date` function for details (NB, this concerns only the dates given as strings and not as `Date` objects).

`summary`
: Metadata as a *string*, briefly describing this CMA.

`event.interval.colname`
: A *string*, the name of a newly-created column storing the number of days between the start of the current event and the start of the next one; the default value "event.interval" should be changed only if there is a naming conflict with a pre-existing "event.interval" column in `event.info`.

`gap.days.colname`
: A *string*, the name of a newly-created column storing the number of days when medication was not available (i.e., the "gap days"); the default value "gap.days" should be changed only if there is a naming conflict with a pre-existing "gap.days" column in `event.info`.

`force.NA.CMA.for.failed.patients`
: *Logical* describing how the patients for which the CMA estimation fails are treated: if `TRUE` they are returned with an `NA` CMA estimate, while for `FALSE` they are omitted.

`return.mapping.events.sliding.window`
: A *Logical*, if `TRUE` then the mapping between events and sliding windows is returned as the component `mapping.windows.to.events`, which is a `data.table` giving, for each sliding window, the events that belong to it (an event is given by its row number in the `data`).

`parallel.backend`
: Can be "none" (the default) for single-threaded execution, "multicore" (using `mclapply` in package `parallel`) for multicore processing (NB. not currently implemented on MS Windows and automatically falls back on "snow" on this platform), or "snow", "snow(SOCK)" (equivalent to "snow"), "snow(MPI)" or "snow(NWS)" specifying various types of SNOW clusters (can be on the local machine or more complex setups – please see the documentation of package `snow` for details; the last two require packages `Rmpi` and `nws`, respectively, not automatically installed with `AdhereR`).

`parallel.threads`
: Can be "auto" (for `parallel.backend` == "multicore", defaults to the number of cores in the system as given by `options("cores")`, while for `parallel.backend` == "snow", defaults to 2), a strictly positive integer specifying the number of parallel threads, or a more complex specification of the SNOW cluster nodes for `parallel.backend` == "snow" (see the documentation of package `snow` for details).

`suppress.warnings`
: *Logical*, if `TRUE` don't show any warnings.

suppress.special.argument.checks
: *Logical* parameter for internal use; if FALSE (default) check if the important columns in the data have some of the reserved names, if TRUE this check is not performed.

...
: other possible parameters

Details

CMA_sliding_window first computes a set of fixed-size (possibly partly overlapping) sliding windows, each sliding to the right by a fixed timelag, and then, for each of them, it computes the given "simple" CMA. Thus, as opposed to the "simple" CMAs 1 to 9, it returns a set of CMAs, with possibly more than one element.

It is highly similar to CMA_per_episode which computes a CMA for a set of treatment episodes.

Value

An S3 object of class CMA_sliding_window with the following fields:

- data The actual event data, as given by the data parameter.
- ID.colname the name of the column in data containing the unique patient ID, as given by the ID.colname parameter.
- event.date.colname the name of the column in data containing the start date of the event (in the format given in the date.format parameter), as given by the event.date.colname parameter.
- event.duration.colname the name of the column in data containing the event duration (in days), as given by the event.duration.colname parameter.
- event.daily.dose.colname the name of the column in data containing the prescribed daily dose, as given by the event.daily.dose.colname parameter.
- medication.class.colname the name of the column in data containing the classes/types/groups of medication, as given by the medication.class.colname parameter.
- carry.only.for.same.medication whether the carry-over applies only across medication of the same type, as given by the carry.only.for.same.medication parameter.
- consider.dosage.change whether the carry-over is adjusted to reflect changes in dosage, as given by the consider.dosage.change parameter.
- followup.window.start the beginning of the follow-up window, as given by the followup.window.start parameter.
- followup.window.start.unit the time unit of the followup.window.start, as given by the followup.window.start.unit parameter.
- followup.window.duration the duration of the follow-up window, as given by the followup.window.duration parameter.
- followup.window.duration.unit the time unit of the followup.window.duration, as given by the followup.window.duration.unit parameter.
- observation.window.start the beginning of the observation window, as given by the observation.window.start parameter.

- observation.window.start.unit the time unit of the observation.window.start, as given by the observation.window.start.unit parameter.
- observation.window.duration the duration of the observation window, as given by the observation.window.duration parameter.
- observation.window.duration.unit the time unit of the observation.window.duration, as given by the observation.window.duration.unit parameter.
- date.format the format of the dates, as given by the date.format parameter.
- summary the metadata, as given by the summary parameter.
- event.info the data.frame containing the event info (irrelevant for most users; see compute.event.int.gaps for details).
- computed.CMA the class name of the computed CMA.
- CMA the data.frame containing the actual CMA estimates for each participant (the ID.colname column) and sliding window, with columns:
  - ID.colname the patient ID as given by the ID.colname parameter.
  - window.ID the unique window ID (within patients).
  - window.start the window's start date (as a Date object).
  - window.end the window's end date (as a Date object).
  - CMA the window's estimated CMA.

Please note that if medication.groups are defined, then the CMA and event.info are named lists, each element containing the CMA and event.info corresponding to a single medication group (the element's name).

See Also

CMA_per_episode is very similar, computing a "simple" CMA for each of the treatment episodes. The "simple" CMAs that can be computed comprise CMA1, CMA2, CMA3, CMA4, CMA5, CMA6, CMA7, CMA8, CMA9, as well as user-defined classes derived from CMA0 that have a CMA component giving the estimated CMA per patient as a data.frame. If return.mapping.events.sliding.window is TRUE, then this also has an attribute mapping.windows.to.events that gives the mapping between episodes and events as a data.table with the following columns:

- patid the patient ID.
- window.ID the episode unique ID (increasing sequentially).
- event.index.in.data the event given by its row number in the data.

Examples

```
## Not run:
cmaW <- CMA_sliding_window(CMA="CMA1",
                           data=med.events,
                           ID.colname="PATIENT_ID",
                           event.date.colname="DATE",
                           event.duration.colname="DURATION",
                           event.daily.dose.colname="PERDAY",
                           medication.class.colname="CATEGORY",
                           carry.only.for.same.medication=FALSE,
```

```
                              consider.dosage.change=FALSE,
                              followup.window.start=0,
                              observation.window.start=0,
                              observation.window.duration=365,
                              sliding.window.start=0,
                              sliding.window.start.unit="days",
                              sliding.window.duration=90,
                              sliding.window.duration.unit="days",
                              sliding.window.step.duration=7,
                              sliding.window.step.unit="days",
                              sliding.window.no.steps=NA,
                              date.format="%m/%d/%Y"
                             );
## End(Not run)
```

`compute.event.int.gaps` *Gap Days and Event (prescribing or dispensing) Intervals.*

Description

For a given event (prescribing or dispensing) database, compute the gap days and event intervals in various scenarious.

Usage

```
compute.event.int.gaps(
  data,
  ID.colname = NA,
  event.date.colname = NA,
  event.duration.colname = NA,
  event.daily.dose.colname = NA,
  medication.class.colname = NA,
  event.interval.colname = "event.interval",
  gap.days.colname = "gap.days",
  carryover.within.obs.window = FALSE,
  carryover.into.obs.window = FALSE,
  carry.only.for.same.medication = FALSE,
  consider.dosage.change = FALSE,
  followup.window.start = 0,
  followup.window.start.unit = c("days", "weeks", "months", "years")[1],
  followup.window.duration = 365 * 2,
  followup.window.duration.unit = c("days", "weeks", "months", "years")[1],
  observation.window.start = 0,
  observation.window.start.unit = c("days", "weeks", "months", "years")[1],
  observation.window.duration = 365 * 2,
  observation.window.duration.unit = c("days", "weeks", "months", "years")[1],
  date.format = "%m/%d/%Y",
```

```
  keep.window.start.end.dates = FALSE,
  remove.events.outside.followup.window = TRUE,
  keep.event.interval.for.all.events = FALSE,
  parallel.backend = c("none", "multicore", "snow", "snow(SOCK)", "snow(MPI)",
    "snow(NWS)")[1],
  parallel.threads = "auto",
  suppress.warnings = FALSE,
  suppress.special.argument.checks = FALSE,
  return.data.table = FALSE,
  ...
)
```

Arguments

`data`	A `data.frame` containing the events used to compute the CMA. Must contain, at a minimum, the patient unique ID, the event date and duration, and might also contain the daily dosage and medication type (the actual column names are defined in the following four parameters); the `CMA` constructors call this parameter `data`.
`ID.colname`	A *string*, the name of the column in `data` containing the unique patient ID; must be present.
`event.date.colname`	A *string*, the name of the column in `data` containing the start date of the event (in the format given in the `date.format` parameter); must be present.
`event.duration.colname`	A *string*, the name of the column in `data` containing the event duration (in days); must be present.
`event.daily.dose.colname`	A *string*, the name of the column in `data` containing the prescribed daily dose, or `NA` if not defined.
`medication.class.colname`	A *string*, the name of the column in `data` containing the classes/types/groups of medication, or `NA` if not defined.
`event.interval.colname`	A *string*, the name of a newly-created column storing the number of days between the start of the current event and the start of the next one; the default value "event.interval" should be changed only if there is a naming conflict with a pre-existing "event.interval" column in `event.info`.
`gap.days.colname`	A *string*, the name of a newly-created column storing the number of days when medication was not available (i.e., the "gap days"); the default value "gap.days" should be changed only if there is a naming conflict with a pre-existing "gap.days" column in `event.info`.
`carryover.within.obs.window`	*Logical*, if `TRUE` consider the carry-over within the observation window, or `NA` if not defined.

carryover.into.obs.window
: *Logical*, if TRUE consider the carry-over from before the starting date of the observation window, or NA if not defined.

carry.only.for.same.medication
: *Logical*, if TRUE the carry-over applies only across medication of the same type, or NA if not defined.

consider.dosage.change
: *Logical*, if TRUE the carry-over is adjusted to reflect changes in dosage, or NA if not defined.

followup.window.start
: If a Date object, it represents the actual start date of the follow-up window; if a *string* it is the name of the column in data containing the start date of the follow-up window either as the numbers of followup.window.start.unit units after the first event (the column must be of type numeric) or as actual dates (in which case the column must be of type Date); if a *number* it is the number of time units defined in the followup.window.start.unit parameter after the begin of the participant's first event.

followup.window.start.unit
: can be either *"days"*, *"weeks"*, *"months"* or *"years"*, and represents the time units that followup.window.start refers to (when a number), or NA if not defined.

followup.window.duration
: either a *number* representing the duration of the follow-up window in the time units given in followup.window.duration.unit, or a *string* giving the column containing these numbers. Should represent a period for which relevant medication events are recorded accurately (e.g. not extend after end of relevant treatment, loss-to-follow-up or change to a health care provider not covered by the database).

followup.window.duration.unit
: can be either *"days"*, *"weeks"*, *"months"* or *"years"*, and represents the time units that followup.window.duration refers to, or NA if not defined.

observation.window.start, observation.window.start.unit, observation.window.duration, observation.wir
: the definition of the observation window (see the follow-up window parameters above for details).

date.format
: A *string* giving the format of the dates used in the data and the other parameters; see the format parameters of the as.Date function for details (NB, this concerns only the dates given as strings and not as Date objects).

keep.window.start.end.dates
: *Logical*, should the computed start and end dates of the windows be kept?

remove.events.outside.followup.window
: *Logical*, should the events that fall outside the follo-wup window be removed from the results?

keep.event.interval.for.all.events
: *Logical*, should the computed event intervals be kept for all events, or NA'ed for those outside the OW?

parallel.backend
: Can be "none" (the default) for single-threaded execution, "multicore" (using mclapply in package parallel) for multicore processing (NB. not currently

implemented on MS Windows and automatically falls back on "snow" on this platform), or "snow", "snow(SOCK)" (equivalent to "snow"), "snow(MPI)" or "snow(NWS)" specifying various types of SNOW clusters (can be on the local machine or more complex setups – please see the documentation of package snow for details; the last two require packages Rmpi and nws, respectively, not automatically installed with AdhereR).

parallel.threads
: Can be "auto" (for parallel.backend == "multicore", defaults to the number of cores in the system as given by options("cores"), while for parallel.backend == "snow", defaults to 2), a strictly positive integer specifying the number of parallel threads, or a more complex specification of the SNOW cluster nodes for parallel.backend == "snow" (see the documentation of package snow for details).

suppress.warnings
: *Logical*, if TRUE don't show any warnings.

suppress.special.argument.checks
: *Logical* parameter for internal use; if FALSE (default) check if the important columns in the data have some of the reserved names, if TRUE this check is not performed.

return.data.table
: *Logical*, if TRUE return a data.table object, otherwise a data.frame.

...
: extra arguments.

Details

This should in general not be called directly by the user, but is provided as a basis for the extension to new CMAs.

Value

A data.frame or data.table extending the event.info parameter with:

- event.interval Or any other name given in event.interval.colname, containing the number of days between the start of the current event and the start of the next one.
- gap.days Or any other name given in gap.days.colname, containing the number of days when medication was not available for the current event (i.e., the "gap days").
- .FU.START.DATE,.FU.END.DATE if kept, the actual start and end dates of the follow-up window (after adjustments due to the various parameters).
- .OBS.START.DATE,.OBS.END.DATE if kept, the actual start and end dates of the observation window (after adjustments due to the various parameters).
- .EVENT.STARTS.BEFORE.OBS.WINDOW if kept, TRUE if the current event starts before the start of the observation window.
- .TDIFF1,.TDIFF2 if kept, various auxiliary time differences (in days).
- .EVENT.STARTS.AFTER.OBS.WINDOW if kept, TRUE if the current event starts after the end of the observation window.
- .CARRY.OVER.FROM.BEFORE if kept, the carry-over (if any) from the previous events.
- .EVENT.WITHIN.FU.WINDOW if kept, TRUE if the current event is within the follow-up window.

compute.treatment.episodes *Compute Treatment Episodes.*

Description

For a given event (prescribing or dispensing) database, compute the treatment episodes for each patient in various scenarious.

Usage

```
compute.treatment.episodes(
  data,
  ID.colname = NA,
  event.date.colname = NA,
  event.duration.colname = NA,
  event.daily.dose.colname = NA,
  medication.class.colname = NA,
  carryover.within.obs.window = TRUE,
  carry.only.for.same.medication = TRUE,
  consider.dosage.change = TRUE,
  medication.change.means.new.treatment.episode = TRUE,
  dosage.change.means.new.treatment.episode = FALSE,
  maximum.permissible.gap = 90,
 maximum.permissible.gap.unit = c("days", "weeks", "months", "years", "percent")[1],
  maximum.permissible.gap.append.to.episode = FALSE,
  followup.window.start = 0,
  followup.window.start.unit = c("days", "weeks", "months", "years")[1],
  followup.window.duration = 365 * 2,
  followup.window.duration.unit = c("days", "weeks", "months", "years")[1],
  event.interval.colname = "event.interval",
  gap.days.colname = "gap.days",
  return.mapping.events.episodes = FALSE,
  date.format = "%m/%d/%Y",
  parallel.backend = c("none", "multicore", "snow", "snow(SOCK)", "snow(MPI)",
    "snow(NWS)")[1],
  parallel.threads = "auto",
  suppress.warnings = FALSE,
  suppress.special.argument.checks = FALSE,
  return.data.table = FALSE,
  ...
)
```

Arguments

data	A data.frame containing the events used to compute the CMA. Must contain, at a minimum, the patient unique ID, the event date and duration, and might also

contain the daily dosage and medication type (the actual column names are defined in the following four parameters); the CMA constructors call this parameter data.

ID.colname
: A *string*, the name of the column in data containing the unique patient ID, or NA if not defined.

event.date.colname
: A *string*, the name of the column in data containing the start date of the event (in the format given in the date.format parameter), or NA if not defined.

event.duration.colname
: A *string*, the name of the column in data containing the event duration (in days), or NA if not defined.

event.daily.dose.colname
: A *string*, the name of the column in data containing the prescribed daily dose, or NA if not defined.

medication.class.colname
: A *string*, the name of the column in data containing the classes/types/groups of medication, or NA if not defined.

carryover.within.obs.window
: *Logical*, if TRUE consider the carry-over within the observation window, or NA if not defined.

carry.only.for.same.medication
: *Logical*, if TRUE the carry-over applies only across medication of the same type, or NA if not defined.

consider.dosage.change
: *Logical*, if TRUE the carry-over is adjusted to reflect changes in dosage, or NA if not defined.

medication.change.means.new.treatment.episode
: *Logical*, should a change in medication automatically start a new treatment episode?

dosage.change.means.new.treatment.episode
: *Logical*, should a change in dosage automatically start a new treatment episode?

maximum.permissible.gap
: The *number* of units given by maximum.permissible.gap.unit representing the maximum duration of permissible gaps between treatment episodes (can also be a percent, see maximum.permissible.gap.unit for details).

maximum.permissible.gap.unit
: can be either *"days"*, *"weeks"*, *"months"*, *"years"* or *"percent"*, and represents the time units that maximum.permissible.gap refers to; if *percent*, then maximum.permissible.gap is interpreted as a percent (can be greater than 100%) of the duration of the current prescription.

maximum.permissible.gap.append.to.episode
: a *logical* value specifying of the maximum.permissible.gap should be append at the end of an episode with a gap larger than the maximum.permissible.gap; FALSE (the default) mean no addition, while TRUE mean that the full maximum.permissible.gap is added.

followup.window.start
: If a Date object it is the actual start date of the follow-up window; if a *string* it is the name of the column in data containing the start date of the follow-up window; if a *number* it is the number of time units defined in the followup.window.start.unit parameter after the begin of the participant's first event; or NA if not defined.

followup.window.start.unit
: can be either *"days"*, *"weeks"*, *"months"* or *"years"*, and represents the time units that followup.window.start refers to (when a number), or NA if not defined.

followup.window.duration
: a *number* representing the duration of the follow-up window in the time units given in followup.window.duration.unit, or NA if not defined.

followup.window.duration.unit
: can be either *"days"*, *"weeks"*, *"months"* or *"years"*, and represents the time units that followup.window.duration refers to, or NA if not defined.

event.interval.colname
: A *string*, the name of a newly-created column storing the number of days between the start of the current event and the start of the next one; the default value "event.interval" should be changed only if there is a naming conflict with a pre-existing "event.interval" column in event.info.

gap.days.colname
: A *string*, the name of a newly-created column storing the number of days when medication was not available (i.e., the "gap days"); the default value "gap.days" should be changed only if there is a naming conflict with a pre-existing "gap.days" column in event.info.

return.mapping.events.episodes
: A *Logical*, if TRUE then the mapping between events and episodes is returned as the attribute mapping.episodes.to.events, which is a data.table giving, for each episode, the events that belong to it (an event is given by its row number in the data). Please note that the episodes returned are quite "generic" (e.g., they include all the events in the FUW), because which events will be actually used in the computation of a CMA_per_episode depend on which simple CMA is used (see also CMA_per_episode), and should be used with care (we recommend using the mappings given by CMA_per_episode instead).

date.format
: A *string* giving the format of the dates used in the data and the other parameters; see the format parameters of the as.Date function for details (NB, this concerns only the dates given as strings and not as Date objects).

parallel.backend
: Can be "none" (the default) for single-threaded execution, "multicore" (using mclapply in package parallel) for multicore processing (NB. not currently implemented on MS Windows and automatically falls back on "snow" on this platform), or "snow", "snow(SOCK)" (equivalent to "snow"), "snow(MPI)" or "snow(NWS)" specifying various types of SNOW clusters (can be on the local machine or more complex setups – please see the documentation of package snow for details; the last two require packages Rmpi and nws, respectively, not automatically installed with AdhereR).

parallel.threads
: Can be "auto" (for parallel.backend == "multicore", defaults to the number of cores in the system as given by options("cores"), while for parallel.backend

== "snow", defaults to 2), a strictly positive integer specifying the number of parallel threads, or a more complex specification of the SNOW cluster nodes for `parallel.backend` == "snow" (see the documentation of package `snow` for details).

`suppress.warnings`
: *Logical*, if `TRUE` don't show any warnings.

`suppress.special.argument.checks`
: *Logical* parameter for internal use; if `FALSE` (default) check if the important columns in the `data` have some of the reserved names, if `TRUE` this check is not performed.

`return.data.table`
: *Logical*, if `TRUE` return a `data.table` object, otherwise a `data.frame`.

`...`
: extra arguments.

Details

This should in general not be called directly by the user, but is provided as a basis for the extension to new CMAs.

For the last treatment episode, the gap is considered only when longer than the maximum permissible gap. Please note the following:

- episode starts at first medication event for a particular medication,
- episode ends on the day when the last supply of that medication finished or if a period longer than the permissible gap preceded the next medication event, or at the end of the FUW,
- end episode gap days represents either the number of days after the end of the treatment episode (if medication changed, or if a period longer than the permissible gap preceded the next medication event) or at the end of (and within) the episode, i.e. the number of days after the last supply finished (if no other medication event followed until the end of the FUW),
- the duration of the episode is the interval between the episode start and episode end (and may include the gap days at the end, in the latter condition described above),
- the number of gap days after the end of the episode can be computed as all values larger than the permissible gap and 0 otherwise,
- if medication change starts new episode, then previous episode ends when the last supply is finished (irrespective of the length of gap compared to a maximum permissible gap); any days before the date of the new medication supply are considered a gap; this maintains consistency with gaps between episodes (whether they are constructed based on the maximum permissible gap rule or the medication change rule).

Value

A `data.frame` or `data.table` with the following columns (or `NULL` if no treatment episodes could be computed):

- `patid` the patient ID.
- `episode.ID` the episode unique ID (increasing sequentially).
- `episode.start` the episode start date.

- end.episode.gap.days the corresponding gap days of the last event in this episode.
- episode.duration the episode duration in days.
- episode.end the episode end date.

If mapping.episodes.to.events is TRUE, then this also has an *attribute* mapping.episodes.to.events that gives the mapping between episodes and events as a data.table with the following columns:

- patid the patient ID.
- episode.ID the episode unique ID (increasing sequentially).
- event.index.in.data the event given by its row number in the data.

---

compute_event_durations

Computation of event durations.

Description

Computes event durations based on dispensing, prescription, and other data (e.g. hospitalization data) and returns a data.frame which can be used with the CMA constructors in AdhereR.

Usage

```
compute_event_durations(
  disp.data = NULL,
  presc.data = NULL,
  special.periods.data = NULL,
  ID.colname,
  medication.class.colnames,
  disp.date.colname,
  total.dose.colname,
  presc.date.colname,
  presc.daily.dose.colname,
  presc.duration.colname,
  visit.colname,
  split.on.dosage.change = TRUE,
  force.init.presc = FALSE,
  force.presc.renew = FALSE,
  trt.interruption = c("continue", "discard", "carryover")[1],
  special.periods.method = trt.interruption,
  carryover = FALSE,
  date.format = "%d.%m.%Y",
  suppress.warnings = FALSE,
  return.data.table = FALSE,
  progress.bar = TRUE,
  ...
)
```

Arguments

disp.data	A data.frame or data.table containing the dispensing events. Must contain, at a minimum, the patient unique ID, one medication identifier, the dispensing date, and total dispensed dose, and might also contain additional columns to identify and group medications (the actual column names are defined in the medication.class.colnames parameter).
presc.data	A data.frame containing the prescribing events. Must contain, at a minimum, the same unique patient ID and medication identifier(s) as the dispensing data, the prescription date, the daily prescribed dose, and the prescription duration. Optionally, it might also contain a visit number.
special.periods.data	Optional, NULL or a data.frame containing the information about special periods (e.g., hospitalizations or other situations where medication use may differ, e.g. during incarcerations or holidays). Must contain the same unique patient ID as dispensing and prescription data, the start and end dates of the special periods with the exact column names DATE.IN and DATE.OUT. Optional columns are TYPE (indicating the type of special situation), customized instructions how to handle a specific period (see special.periods.method), and any of those specified in medication.class.colnames.
ID.colname	A *string*, the name of the column in disp.data, presc.data, and special.periods.data containing the unique patient ID.
medication.class.colnames	A Vector of *strings*, the name(s) of the column(s) in disp.data and presc.data containing the classes/types/groups of medication.
disp.date.colname	A *string*, the name of the column in disp.data containing the dispensing date (in the format given in the date.format parameter).
total.dose.colname	A *string*, the name of the column in disp.data containing the total dispensed dose as numeric (e.g. 500 for 10 tablets of 50 mg).
presc.date.colname	A *string*, the name of the column in presc.data containing the prescription date (in the format given in the date.format parameter).
presc.daily.dose.colname	A *string*, the name of the column in presc.data containing the daily prescribed dose as numeric (e.g. 50 for 50 mg once per day, or 25 for 50 mg once ever 2 days).
presc.duration.colname	A *string*, the name of the column in presc.data containing the duration of the prescription as numeric or NA if duration is unknown.
visit.colname	A *string*, the name of the column in presc.data containing the number of the visit or a new column name if the prescribing data does not contain such a column.
split.on.dosage.change	*Logical* or *string*. If TRUE split the dispensing event on days with dosage change and create a new event with the new dosage for the remaining supply. If *string*,

the name of the column containing the `Logical` in *disp.data* for each medication class separatly. Important if carryover should be considered later on.

`force.init.presc`
: *Logical*. If `TRUE` advance the date of the first prescription event to the date of the first dispensing event, if the first prescription event is after the first dispensing event for a specific medication. Only if the first prescription event is not limited in duration (as indicated in the `presc.duration.colname`).

`force.presc.renew`
: *Logical* or *string*. If `TRUE` require a new prescription for all medications for every prescription event (visit), otherwise prescriptions end on the first visit without renewal. If *string*, the name of the column in *disp.data* containing the `Logical` for each medication class separatly.

`trt.interruption`
: can be either of *"continue"*, *"discard"*, *"carryover"*, or a *string*. It indicates how to handle durations during treatment interruptions (see `special.periods.method`). If *string*, the name of the (*character*) column in *disp.data* containing the information (*"continue"*, *"discard"*, or *"carryover"*) for each medication class separatly.

`special.periods.method`
: can be either of *continue*, *discard*, *carryover*, or *custom*. It indicates how to handle durations during special periods. With *continue*, special periods have no effect on durations and event start dates. With *discard*, durations are truncated at the beginning of special periods and the remaining quantity is discarded. With *carryover*, durations are truncated at the beginning of a special period and a new event with the remaining duration is created after the end of the end of the special period. With *custom*, the mapping has to be included in `special.periods.data`.

`carryover`
: *Logical*, if `TRUE` apply carry-over to medications of the same type (according to `medication.class.colnames`). Can only be used together with CMA7 and above in combination with `carry.only.for.same.medication = TRUE`.

`date.format`
: A *string* giving the format of the dates used in the `data` and the other parameters; see the `format` parameters of the `as.Date` function for details (NB, this concerns only the dates given as strings and not as `Date` objects).

`suppress.warnings`
: *Logical*, if `TRUE` don't show any warnings.

`return.data.table`
: *Logical*, if `TRUE` return a `data.table` object, otherwise a `data.frame`.

`progress.bar`
: *Logical*, if `TRUE` show a progress bar.

`...`
: other possible parameters.

Details

Computation of CMAs requires a supply duration for medications dispensed to patients. If medications are not supplied for fixed durations but as a quantity that may last for various durations based on the prescribed dose, the supply duration has to be calculated based on dispensed and prescribed doses. Treatments may be interrupted and resumed at later times, for which existing supplies may or

may not be taken into account. Patients may be hospitalized or incarcerated, and may not use their own supplies during these periods. This function calculates supply durations, taking into account the aforementioned situations and providing various parameters for flexible adjustments.

Value

A `list` with the following elements:

- `event_durations`: A `data.table` or `data.frame` with the following columns:
  - `ID.colname` the unique patient ID, as given by the `ID.colname` parameter.
  - `medication.class.colnames` the column(s) with classes/types/groups of medication, as given by the `medication.class.colnames` parameter.
  - `disp.date.colname` the date of the dispensing event, as given by the `disp.date.colnema` parameter.
  - `total.dose.colname` the total dispensed dose, as given by the `total.dose.colname` parameter.
  - `presc.daily.dose.colname` the prescribed daily dose, as given by the `presc.daily.dose.colname` parameter.
  - `DISP.START` the start date of the dispensing event, either the same as in `disp.date.colname` or a later date in case of dosage changes or treatment interruptions/hospitalizations.
  - `DURATION` the calculated duration of the supply, based on the total dispensed dose and the prescribed daily dose, starting from the `DISP.START` date.
  - `episode.start`: the start date of the current prescription episode.
  - `episode.end`: the end date of the current prescription episode. Can be before the start date of the dispensing event if dispensed during a treatment interruption.
  - `SPECIAL.DURATION` the number of days *during* the current duration affected by special durations or treatment interruptions of type "continue".
  - `CARRYOVER.DURATION` the number of days *after* the current duration affected by special durations or treatment interruptions of type "carryover".
  - `EVENT.ID`: in case of multiple events with the same dispensing date (e.g. for dosage changes or interruptions); a unique ID starting at 1 for the first event
  - `tot.presc.interruptions` the total number of prescription interruptions per patient for a specific medication.
  - `tot.dosage.changes` the total number of dosage changes per patient for a specific medication.
- `prescription_episodes`: A `data.table` or `data.frame` with the following columns:
  - `ID.colname`: the unique patient ID, as given by the `ID.colname` parameter.
  - `medication.class.colnames`: the column(s) with classes/types/groups of medication, as given by the `medication.class.colnames` parameter.
  - `presc.daily.dose.colname`: the prescribed daily dose, as given by the `presc.daily.dose.colname` parameter.
  - `episode.start`: the start date of the prescription episode.
  - `episode.duration`: the duration of the prescription episode in days.
  - `episode.end`: the end date of the prescription episode.
- `special_periods`: A `data.table` or `data.frame`, the `special.periods.data` with an additional column `SPECIAL.DURATION`: the number of days between `DATE.IN` and `DATE.OUT`

- ID.colname the name of the columns containing the unique patient ID, as given by the ID.colname parameter.
- medication.class.colnames the name(s) of the column(s) in disp.data and presc.data containing the classes/types/groups of medication, as given by the medication.class.colnames parameter.
- disp.date.colname the name of the column in disp.data containing the dispensing date, as given in the disp.date.colname parameter.
- total.dose.colname the name of the column in disp.data containing the total dispensed dose, as given by the total.dose.colname parameter.
- presc.date.colname the name of the column in presc.data containing the prescription date, as given in the presc.date.colname parameter.
- presc.daily.dose.colname the name of the column in presc.data containing the daily prescribed dose, as given by the presc.daily.dose.colname parameter.
- presc.duration.colname the name of the column in presc.data containing the duration of the prescription, as given by the presc.duration.colname parameter.
- visit.colname the name of the column containing the number of the visit, as given by the visit.colname parameter
- split.on.dosage.change whether to split the dispensing event on days with dosage changes and create a new event with the new dosage for the remaining supply, as given by the split.on.dosage.change parameter.
- force.init.presc whether the date of the first prescription event was set back to the date of the first dispensing event, when the first prescription event was after the first dispensing event for a specific medication, as given by the force.init.presc parameter.
- force.presc.renew whether a new prescription was required for all medications for every prescription event (visit), as given by the force.presc.renew parameter.
- trt.interruption how durations during treatment interruptions were handled, as given by the trt.interruption parameter.
- special.periods.method as given by the special.periods.method parameter.
- date.format the format of the dates, as given by the date.format parameter.

Examples

```
## Not run:
event_durations <- compute_event_durations(disp.data = durcomp.dispensing,
                                           presc.data = durcomp.prescribing,
                                         special.periods.data = durcomp.hospitalisation,
                                           ID.colname = "ID",
                                           presc.date.colname = "DATE.PRESC",
                                           disp.date.colname = "DATE.DISP",
                                           medication.class.colnames = c("ATC.CODE",
                                           "UNIT", "FORM"),
                                           total.dose.colname = "TOTAL.DOSE",
                                           presc.daily.dose.colname = "DAILY.DOSE",
                                           presc.duration.colname = "PRESC.DURATION",
                                           visit.colname = "VISIT",
                                           split.on.dosage.change = TRUE,
```

```
                                        force.init.presc = TRUE,
                                        force.presc.renew = TRUE,
                                        trt.interruption = "continue",
                                        special.periods.method = "continue",
                                        date.format = "%Y-%m-%d",
                                        suppress.warnings = FALSE,
                                        return.data.table = TRUE);

## End(Not run)
```

`cover_special_periods` *Cover special periods.*

Description

Identifies special periods that are in proximity to already covered durations and adds additional events for these durations.

Usage

```
cover_special_periods(
  events.data,
  special.periods.data,
  ID.colname,
  medication.class.colnames,
  disp.date.colname,
  disp.start.colname,
  episode.start.colname,
  episode.end.colname,
  duration.colname,
  days.before,
  days.after,
  date.format,
  suppress.warnings = FALSE,
  return.data.table = FALSE,
  ...
)
```

Arguments

`events.data`	A `data.frame` or `data.table` with the event durations.
`special.periods.data`	a `data.frame` or or `data.table` containing the information about special periods (e.g., hospitalizations or other situations where medication use may differ, e.g. during incarcerations or holidays). Must contain the same unique patient ID as dispensing and prescription data, the start and end dates of the special periods with the exact column names `DATE.IN` and `DATE.OUT`.

`ID.colname` A *string*, the name of the column in `events.data` and `special.periods.data` containing the unique patient ID.

`medication.class.colnames`
A `Vector` of *strings*, the name(s) of the column(s) in the `events.data` identify medication classes.

`disp.date.colname`
A *string*, the name of the column in `events.data` containing the dispensation start date (in the format given in the `date.format` parameter).

`disp.start.colname, episode.start.colname, episode.end.colname`
column names in `events.data`.

`duration.colname`
A *string*, the name of the column in `events.data` containing the duration of the medication event.

`days.before` an *integer*, the number of days before the start of a special period within which an event duration must end to consider the special period as covered.

`days.after` an *integer*, the number of days after a special period within which an event duration must start to consider the special period as covered.

`date.format` A *string* giving the format of the dates used in the `data` and the other parameters; see the `format` parameters of the `as.Date` function for details (NB, this concerns only the dates given as strings and not as `Date` objects).

`suppress.warnings`
Logical, if `TRUE` don't show any warnings.

`return.data.table`
Logical, if `TRUE` return a `data.table` object, otherwise a `data.frame`.

`...` other possible parameters.

Details

Special periods may appear as gaps, possibly leading to underestimation of implementation or even assumption of discontinuation and non-persistence. To consider such periods as covered, this function adds additional durations, for example when it is assumed that hospitalized patients are adherent during the hospitalization period. This function should be used after pruning with `prune_event_durations`.

Value

A `data.frame` or `data.table`, the `events.data` with the additional durations for special periods covered.

Examples

```
## Not run:
# select medication class of interest and compute event durations
disp_data <- durcomp.dispensing[ID == 3 & grepl("J01EE01", ATC.CODE)]
presc_data <- durcomp.prescribing[ID == 3 & grepl("J01EE01", ATC.CODE)]

event_durations_list <- compute_event_durations(disp.data = disp_data,
                                                presc.data = presc_data,
```

```
                                        special.periods.data = durcomp.hospitalisation,
                                               special.periods.method = "carryover",
                                               ID.colname = "ID",
                                               presc.date.colname = "DATE.PRESC",
                                               disp.date.colname = "DATE.DISP",
                                               date.format = "%Y-%m-%d",
                                              medication.class.colnames = c("ATC.CODE",
                                                                            "UNIT",
                                                                            "FORM"),
                                               total.dose.colname = "TOTAL.DOSE",
                                              presc.daily.dose.colname = "DAILY.DOSE",
                                             presc.duration.colname = "PRESC.DURATION",
                                               visit.colname = "VISIT",
                                               force.init.presc = TRUE,
                                               force.presc.renew = TRUE,
                                               split.on.dosage.change = TRUE,
                                               trt.interruption = "carryover",
                                               suppress.warnings = FALSE,
                                               return.data.table = TRUE,
                                               progress.bar = FALSE)

event_durations <- prune_event_durations(event_durations_list,
                                         include = c("special periods"),
                                         medication.class.colnames = "ATC.CODE",
                                         days.within.out.date.1 = 7,
                                         days.within.out.date.2 = 30,
                                         keep.all = TRUE)

# cover special periods
special_periods <- event_durations_list$special_periods
event_durations_covered <- cover_special_periods(events.data = event_durations,
                                                 special.periods.data = special_periods,
                                                 ID.colname = "ID",
                                                 medication.class.colnames = "ATC.CODE",
                                                 disp.start.colname = "DISP.START",
                                                 duration.colname = "DURATION",
                                                 days.before = 7,
                                                 days.after = 7,
                                                 date.format = "%Y-%m-%d")

## End(Not run)
```

`durcomp.dispensing` *Example dispensing events for 16 patients.*

Description

A sample dataset containing dispensing events (one per row) for 16 patients over a period of roughly 24 months (1794 events in total). This is the appropriate format to compute event durations with the `compute_event_durations` function. Each row represents an individual dispensing record for

a specific dose of a specific medication for a patient at a given date. More than one column to group medications can be supplied (such as ATC code, Form and Unit).

Usage

```
durcomp.dispensing
```

Format

A data frame with 1794 rows and 6 variables:

ID *integer* here; patient unique identifier. Can also be *string*.

DATE.DISP *Date* here;the dispensing event date, by default in the yyyy-mm-dd format. Can also be *string*.

ATC.CODE *character*; the medication type, according to the WHO ATC classification system. This can be a researcher-defined classification depending on study aims (e.g., based on therapeutic use, mechanism of action, chemical molecule, or pharmaceutical formulation). The `compute_event_durations` function will match prescribed medication to dispensed medications based on this variable.

UNIT *integer*; the unit of the dispensed dose. This is optional and can be used as a separate variable to match between prescription and dispensing events.

FORM *character*; the galenic form of the dispensed preparation. This is optional and can be used as a separate variable to match between prescription and dispensing events.

TOTAL.DOSE *numeric*; the total dispensed dose supplied at this medication event (e.g., `5000` for 10 tables of 500 mg).

---

durcomp.hospitalisation

Example special periods for 10 patients.

Description

A sample dataset containing special periods (one per row) for 10 patients over a period of roughly 18 months (28 events in total). This is the appropriate format to compute event durations with the `compute_event_durations` function. Each row represents an individual special period of type "hospitalization" of a patient for whom event durations should be calculated. Besides hospitalizations, this could cover other situations where medication use may differ, e.g. during incarcerations or holidays. All column names must match the format provided in this example.

Usage

```
durcomp.hospitalisation
```

Format

A data frame with 28 rows and 3 variables:

ID *Integer* here; patient unique identifier. Can also be *string*.

DATE.IN *Date* here;the start of the hospitalization period, by default in the yyyy-mm-dd format.Can also be *string*.

DATE.OUT *Date*;the end of the hospitalization period, by default in the yyyy-mm-dd format. Can also be *string*.

---

durcomp.prescribing *Example prescription events for 16 patients.*

Description

A sample dataset containing prescription events (one per row) for 16 patients over a period of roughly 15 months (1502 events in total). This is the appropriate format to compute event durations with the `compute_event_durations` function. Each row represents an individual prescription record for a specific dose of a specific medication for a patient at a given date. Visit number and Duration are optional, and more than one column to group medications can be supplied (such as ATC Code, Form or Unit).

Usage

```
durcomp.prescribing
```

Format

A data table with 1502 rows and 8 variables:

ID *integer* here; patient unique identifier. Can also be *string*.

DATE.PRESC *Date* here;the prescription event date, by default in the yyyy-mm-dd format. Can also be *string*.

VISIT *integer*; the consecutive number of the prescription instances. This column is optional and will be generated internally when not supplied. It is used to identify treatment interruptions.

ATC.CODE *character*; the medication type, according to the WHO ATC classification system. This can be a researcher-defined classification depending on study aims (e.g., based on therapeutic use, mechanism of action, chemical molecule, or pharmaceutical formulation). The `compute_event_durations` function will match prescribed medication to dispensed medications based on this variable.

FORM *character*; the galenic form of the prescribed preparation. This is optional and can be used as a separate variable to match between prescription and dispensing events.

UNIT *integer*; the unit of the prescribed dose. This is optional and can be used as a separate variable to match between prescription and dispensing events.

PRESC.DURATION *numeric*; the duration (in days) for which the prescription is intended. Can be `NA` if the prescription is continuous without a fixed end date.

DAILY.DOSE *numeric*; the daily dose prescribed during this event (e.g., `50` for 1 tablet of 50 mg per day or `25` for 1 tablet of 50 mg every two days).

get.event.plotting.area

Get the actual plotting area.

Description

Returns the actual plotting area rectangle in plotting coordinates.

Usage

```
get.event.plotting.area(
  plot.type = c("baseR", "SVG")[1],
  suppress.warnings = FALSE
)
```

Arguments

plot.type	Can be either "baseR" or "SVG" and specifies to which type of plotting the mapping applies.
suppress.warnings	*Logical*, if TRUE don't show any warnings.

Details

This is intended for advanced users only.

Value

A numeric vector with components *x.min*, *x.max*, *y.min* and *y.max*, or NULL in case of error.

get.legend.plotting.area

Get the legend plotting area.

Description

Returns the legend plotting area rectangle in plotting coordinates (if any).

Usage

```
get.legend.plotting.area(
  plot.type = c("baseR", "SVG")[1],
  suppress.warnings = FALSE
)
```

Arguments

plot.type	Can be either "baseR" or "SVG" and specifies to which type of plotting the mapping applies.
suppress.warnings	*Logical*, if TRUE don't show any warnings.

Details

This is intended for advanced users only.

Value

A numeric vector with components *x.min*, *x.max*, *y.min* and *y.max*, or NULL in case of error or no legend being shown.

---

get.plotted.events *Get info about the plotted events.*

Description

Returns a data.frame where each row contains info about one plotted event; the order of the rows reflects the y-axis (first row on bottom).

Usage

```
get.plotted.events(plot.type = c("baseR", "SVG")[1], suppress.warnings = FALSE)
```

Arguments

plot.type	Can be either "baseR" or "SVG" and specifies to which type of plotting the mapping applies.
suppress.warnings	*Logical*, if TRUE don't show any warnings.

Details

This is intended for advanced users only.

Value

A data.frame that, besides the info about each event, also contains info about:

- the corresponding follow-up and observation windows (and, for CMA8, the "real" observation window), given as the corners of the area *.X...START*, *.X...END*, *.Y...START* and *.Y...END* (where the mid dot stands for FUW, OW and ROW, respectively).
- the area occupied by the graphic representation of the event given by its four corners *.X.START*, *.X.END*, *.Y.START* and *.Y.END*, as well as the line width *.EV.LWD*.

- the dose text's (if any) position (*.X.DOSE*, *.Y.DOSE*) and font size *.FONT.SIZE.DOSE*.
- if event corvered and not covered are plotted, also give their areas as *.X.EVC.START*, *.X.EVC.END*, *.Y.EVC.START*, *.Y.EVC.END*, *.X.EVNC.START*, *.X.EVNC.END*, *.Y.EVNC.START* and *.Y.EVNC.END*.
- the continuation lines area as *.X.CNT.START*, *.X.CNT.END*, *.Y.CNT.START* and *.Y.CNT.END*.
- and the corresponding summary CMA (if any) given as the area *.X.SCMA.START*, *.X.SCMA.END*, *.Y.SCMA.START* and *.Y.SCMA.END*.

Please note that even if with follow-up and ("real") observation window, and the summary CMA info is repeated for each event, they really make sense at the level of the patient.

Examples

```
cma7 <- CMA7(data=med.events[med.events$PATIENT_ID %in% c(1,2),],
             ID.colname="PATIENT_ID",
             event.date.colname="DATE",
             event.duration.colname="DURATION",
             event.daily.dose.colname="PERDAY",
             medication.class.colname="CATEGORY",
             followup.window.start=0,
             followup.window.start.unit="days",
             followup.window.duration=2*365,
             followup.window.duration.unit="days",
             observation.window.start=30,
             observation.window.start.unit="days",
             observation.window.duration=365,
             observation.window.duration.unit="days",
             date.format="%m/%d/%Y",
             summary="Base CMA");
plot(cma7);
tmp <- get.plotted.events();
head(tmp);
# "Mask" the first event:
rect(tmp$.X.START[1], tmp$.Y.START[1]-0.5, tmp$.X.END[1], tmp$.Y.END[1]+0.5,
     col=adjustcolor("white",alpha.f=0.75), border="black");
# "Mask" the first patient's summary CMA:
rect(tmp$.X.SCMA.START[1], tmp$.Y.SCMA.START[1],
     tmp$.X.SCMA.END[1], tmp$.Y.SCMA.END[1],
     col=adjustcolor("white",alpha.f=0.75), border="black");
```

`get.plotted.partial.cmas`

Get info about the plotted partial CMAs.

Description

Returns a `data.frame` where each row contains info about one plotted partial CMA (partial CMAs make sense only for "complex" CMAs, i.e., per episode and sliding windows).

Usage

```
get.plotted.partial.cmas(
  plot.type = c("baseR", "SVG")[1],
  suppress.warnings = FALSE
)
```

Arguments

`plot.type`	Can be either "baseR" or "SVG" and specifies to which type of plotting the mapping applies.
`suppress.warnings`	*Logical*, if `TRUE` don't show any warnings.

Details

This is intended for advanced users only.

Value

A `data.frame` that contains info about:

- the patient ID (*pid*) to which the partial CMA belongs.
- the *type* of partial CMA (see the help for plotting "complex" CMAs).
- the corners of the whole area covered by the partial CMA plot given as *x.region.start*, *y.region.start*, *x.region.end* and *y.region.end*.
- for each element of the partial CMA plot, its area as *x.partial.start*, *y.partial.start*, *x.partial.end* and *y.partial.end*.

Please note that this contains one row per partial CMA element (e.g., if plotting stacked, one row for each rectangle).

`getCallerWrapperLocation` *getCallerWrapperLocation.*

Description

This function returns the full path to where the various `wrappers` that can call `AdhereR` are installed.

Usage

```
getCallerWrapperLocation(callig.platform = c("python3")[1], full.path = FALSE)
```

Arguments

`callig.platform`
: A *string* specifying the desired wrapper. Currently it can be "python3".

`full.path`
: A *logical* specifying if the returned path should also include the wrapper's main file name.

Details

In most cases, these wrappers are one or more files in the calling language that may be directly used as such. For more details see the vignette describing the included reference `Python 3` wrapper.

Value

The full path to the requested wrapper or NULL if none exists.

`getCMA` *Access the actual CMA estimate from a CMA object.*

Description

Retrieve the actual CMA estimate(s) encapsulated in a simple, per episode, or sliding window CMA object.

Usage

```
getCMA(
  x,
  flatten.medication.groups = FALSE,
  medication.groups.colname = ".MED_GROUP_ID"
)
```

Arguments

`x`
: a CMA object.

`flatten.medication.groups`
: *Logical*, if `TRUE` and there are medication groups defined, then the return value is flattened to a single `data.frame` with an extra column containing the medication group (its name is given by `medication.groups.colname`).

`medication.groups.colname`
: a *string* (defaults to ".MED_GROUP_ID") giving the name of the column storing the group name when `flatten.medication.groups` is `TRUE`.

Value

a *data.frame* containing the CMA estimate(s).

Examples

```
cma1 <- CMA1(data=med.events,
             ID.colname="PATIENT_ID",
             event.date.colname="DATE",
             event.duration.colname="DURATION",
             followup.window.start=30,
             observation.window.start=30,
             observation.window.duration=365,
             date.format="%m/%d/%Y"
            );
getCMA(cma1);
## Not run:
cmaE <- CMA_per_episode(CMA="CMA1",
                        data=med.events,
                        ID.colname="PATIENT_ID",
                        event.date.colname="DATE",
                        event.duration.colname="DURATION",
                        event.daily.dose.colname="PERDAY",
                        medication.class.colname="CATEGORY",
                        carry.only.for.same.medication=FALSE,
                        consider.dosage.change=FALSE,
                        followup.window.start=0,
                        observation.window.start=0,
                        observation.window.duration=365,
                        date.format="%m/%d/%Y"
                       );
getCMA(cmaE);
## End(Not run)
```

---

getEventInfo *Access the event info from a CMA object.*

Description

Retrieve the event info encapsulated in a simple, per episode, or sliding window CMA object.

Usage

```
getEventInfo(
  x,
  flatten.medication.groups = FALSE,
  medication.groups.colname = ".MED_GROUP_ID"
)
```

Arguments

x	a CMA object.

`flatten.medication.groups`
: *Logical*, if `TRUE` and there are medication groups defined, then the return value is flattened to a single `data.frame` with an extra column containing the medication group (its name is given by `medication.groups.colname`).

`medication.groups.colname`
: a *string* (defaults to ".MED_GROUP_ID") giving the name of the column storing the group name when `flatten.medication.groups` is `TRUE`.

Value

a *data.frame* containing the CMA estimate(s).

Examples

```
cma1 <- CMA1(data=med.events,
             ID.colname="PATIENT_ID",
             event.date.colname="DATE",
             event.duration.colname="DURATION",
             followup.window.start=30,
             observation.window.start=30,
             observation.window.duration=365,
             date.format="%m/%d/%Y"
            );
getEventInfo(cma1);
```

`getEventsToEpisodesMapping` *getEventsToEpisodesMapping*

Description

This function returns the event-to-episode mapping, if this information exists.

Usage

```
getEventsToEpisodesMapping(x)
```

Arguments

`x`
: Either a `data.frame` as returned by `compute.treatment.episodes()` or an `CMA_per_episode object`.

Details

There are cases where it is interesting to know which events belong to which episodes and which events have been actually used in computing the simple CMA for each episode. This information can be returned by `compute.treatment.episodes()` and `CMA_per_episode()` if the parameter `return.mapping.events.episodes` is set to `TRUE`.

Value

The mapping between events and episodes, if it exists either as the attribute `mapping.episodes.to.events` of the `data.frame` or as the `mapping.episodes.to.events` component of the `CMA_per_episode object` object, or `NULL` otherwise.

---

getEventsToSlidingWindowsMapping *getEventsToSlidingWindowsMapping*

Description

This function returns the event-to-sliding-window mapping, if this information exists.

Usage

```
getEventsToSlidingWindowsMapping(x)
```

Arguments

`x`	is an `CMA_sliding_window object`.

Details

There are cases where it is interesting to know which events belong to which sliding windows and which events have been actually used in computing the simple CMA for each sliding window This information can be returned by `CMA_sliding_window()` if the parameter `return.mapping.events.episodes` is set to `TRUE`.

Value

The mapping between events and episodes, if it exists as the `mapping.windows.to.events` component of the `CMA_sliding_window object` object, or `NULL` otherwise.

---

getInnerEventInfo *Access the inner event info from a complex CMA object.*

Description

Retrieve the event info encapsulated in a complex (i.e., per episode or sliding window) CMA object.

Usage

```
getInnerEventInfo(
  x,
  flatten.medication.groups = FALSE,
  medication.groups.colname = ".MED_GROUP_ID"
)
```

Arguments

x	a CMA object.
flatten.medication.groups	*Logical*, if TRUE and there are medication groups defined, then the return value is flattened to a single data.frame with an extra column containing the medication group (its name is given by medication.groups.colname).
medication.groups.colname	a *string* (defaults to ".MED_GROUP_ID") giving the name of the column storing the group name when flatten.medication.groups is TRUE.

Value

a *data.frame* containing the CMA estimate(s).

---

getMGs *Access the medication groups of a CMA object.*

Description

Retrieve the medication groups and the observations they refer to (if any).

Usage

```
getMGs(x)
```

Arguments

x	a CMA object.

Value

a *list* with two members:

- defs A data.frame containing the names and definitions of the medication classes; please note that there is an extra class *__ALL_OTHERS__* containing all the observations not selected by any of the explicitly given medication classes.
- obs A logical matrix where the columns are the medication classes (the last being *__ALL_OTHERS__*), and the rows the observations in the x's data; element (i, j) is TRUE iff observation j was selected by medication class i.

`last.plot.get.info` *Access last adherence plot info.*

Description

Returns the full info the last adherence plot, to be used to modify and/or to add new elements to this plot.

Usage

```
last.plot.get.info()
```

Details

This is intended for advanced users only. It may return `NULL` if no plotting was generated yet, but if one was, a list contaning one named element for each type of plot produced (currently only *baseR* and *SVG* are used). For all types of plots there are a set of *mapping* functions useful for transforming events in plotting coordinates: `.map.event.x(x)` takes a number of days `x`, `.map.event.date(d, adjust.for.earliest.date=TRUE)` takes a `Date d` (and implictely adjusts for the earilerst date plotted), and `.map.event.y(y)` takes a row ("event" number) `y`. Besides the shared elements (see the returned value), there are specific ones as well. For *baseR*, the members *old.par* and *used.par* contain the original (pre-plot) `par()` environment and the one used within `plot()`, respectively, in case these need restoring.

Value

A `list` (possibly empty) contaning one named element for each type of plot produced (currently only *baseR* and *SVG*). Each may contain shared and specific fields concerning:

- the values of the parameters with which `plot()` was invoked.
- actual plot size and other characteristics.
- actual title, axis names and labels and their position and size.
- legend size, position and size and position of its components.
- expanded `cma$data` contaning, for each event, info about its plotting, including the corresponding fullow-uo and observation windows, event start and end, dose text (if any) and other graphical elements.
- position, size of the partial CMAs (if any) and of their components.
- position, size of the plotted CMAs (if any) and of their components.
- rescaling function(s) useful for mapping events to plotting coordinates.

Examples

```
cma7 <- CMA7(data=med.events[med.events$PATIENT_ID %in% c(1,2),],
             ID.colname="PATIENT_ID",
             event.date.colname="DATE",
             event.duration.colname="DURATION",
             event.daily.dose.colname="PERDAY",
             medication.class.colname="CATEGORY",
             followup.window.start=0,
             followup.window.start.unit="days",
             followup.window.duration=2*365,
             followup.window.duration.unit="days",
             observation.window.start=30,
             observation.window.start.unit="days",
             observation.window.duration=365,
             observation.window.duration.unit="days",
             date.format="%m/%d/%Y",
             summary="Base CMA");
plot(cma7);
tmp <- last.plot.get.info();
names(tmp);
tmp$baseR$legend$box; # legend position and size
head(tmp$baseR$cma$data); # events + plotting info
# Add a transparent blue rect between days 270 and 900:
rect(tmp$baseR$.map.event.x(270), tmp$baseR$.map.event.y(1-0.5),
     tmp$baseR$.map.event.x(900), tmp$baseR$.map.event.y(nrow(tmp$baseR$cma$data)+0.5),
     col=adjustcolor("blue",alpha.f=0.5), border="blue");
# Add a transparent rect rect between dates 03/15/2036 and 03/15/2037:
rect(tmp$baseR$.map.event.date(as.Date("03/15/2036", format="%m/%d/%Y")),
     tmp$baseR$.map.event.y(1-0.5),
     tmp$baseR$.map.event.date(as.Date("03/15/2037", format="%m/%d/%Y")),
     tmp$baseR$.map.event.y(nrow(tmp$baseR$cma$data)+0.5),
     col=adjustcolor("red",alpha.f=0.5), border="blue");
```

`map.event.coords.to.plot` *Map from event to plot coordinates.*

Description

Maps the (x,y) coordinates in the event space to the plotting space.

Usage

```
map.event.coords.to.plot(
  x = NA,
  y = NA,
  x.is.Date = FALSE,
  x.date.format = "%m/%d/%Y",
  adjust.for.earliest.date = TRUE,
```

```
  plot.type = c("baseR", "SVG")[1],
  suppress.warnings = FALSE
)
```

Arguments

x	The *x* coordinate in the event space, either a number giving the number of days since the earliest plotted date, or a Date or a string in the format given by the *x.date.format* parameter giving the actual calendar date.
y	The *y* coordinate in the event space, thus a number giving the plot row.
x.is.Date	A logical, being TRUE if *x* is a string giving the date in the *x.date.format* format.
x.date.format	A string giving the format of the *x* date, if *x.is.Date* id TRUE.
adjust.for.earliest.date	A logical which is TRUE if *x* is a calendar date that must be adjusted for the earliest plotted date (by default TRUE).
plot.type	Can be either "baseR" or "SVG" and specifies to which type of plotting the mapping applies.
suppress.warnings	*Logical*, if TRUE don't show any warnings.

Details

This is intended for advanced users only. In the event space, the *x* coordinate can be either given as the number of days since the first plotted event, or as an actual calendar date (either as a Date object or a string with a given format; a date may or may not be corrected relative to the first displayed date). On the *y* coordinate, the plotting is divided in equally spaced rows, each row corresponding to a single event or an element of a partial CMA plot (one can specify in between rows using fractions). Any or both of *x* and *y* can be missing.

Value

A numeric vector with *x* and *y* components giving the plotting coordinates, or NULL in case of error.

Examples

```
cma7 <- CMA7(data=med.events[med.events$PATIENT_ID %in% c(1,2),],
             ID.colname="PATIENT_ID",
             event.date.colname="DATE",
             event.duration.colname="DURATION",
             event.daily.dose.colname="PERDAY",
             medication.class.colname="CATEGORY",
             followup.window.start=0,
             followup.window.start.unit="days",
             followup.window.duration=2*365,
             followup.window.duration.unit="days",
             observation.window.start=30,
             observation.window.start.unit="days",
             observation.window.duration=365,
             observation.window.duration.unit="days",
```

```
              date.format="%m/%d/%Y",
              summary="Base CMA");
plot(cma7);
# Add a transparent blue rect:
rect(map.event.coords.to.plot(x=270),
     get.event.plotting.area()["y.min"]-1,
     map.event.coords.to.plot(x="03/15/2037", x.is.Date=TRUE, x.date.format="%m/%d/%Y"),
     get.event.plotting.area()["y.max"]+1,
     col=adjustcolor("blue",alpha.f=0.5), border="blue");
```

`med.events` *Example medication events records for 100 patients.*

Description

An artificial dataset containing medication events (one per row) for 100 patients (1080 events in total). This is the dataset format appropriate for medication adherence analyses performed with the R package AdhereR. Medication events represent individual records of prescribing or dispensing a specific medication for a patient at a given date. Dosage and medication type is optional (only needed if calculation of adherence or persistence takes into account changes in dosage and type of medication).

Usage

```
med.events
```

Format

A data frame with 1080 rows and 5 variables:

PATIENT_ID *integer* here; patient unique identifier. Can also be *string*.

DATE *character*;the medication event date, by default in the mm/dd/yyyy format. It may represent a prescribing or dispensing date.

PERDAY *integer*; the daily dosage prescribed for the medication supplied at this medication event (i.e. how many doses should be taken daily according to the prescription). This column is optional, as it is not considered in all functions but may be relevant for specific research or clinical contexts. All values should be > 0.

CATEGORY *character*; the medication type, here two placeholder labels, 'medA' and 'medB'. This is a researcher-defined classification depending on study aims (e.g., based on therapeutic use, mechanism of action, chemical molecule, or pharmaceutical formulation). This column is optional, as it is not considered in all functions but may be relevant for specific research or clinical contexts.

DURATION *integer*; the medication event duration in days (i.e. how many days the mediation supplied would last if used as prescribed); may be available in the extraction or computed based on quantity supplied (the number of doses prescribed or dispensed on that occasion) and daily dosage. All values should be > 0.

med.events.ATC *Example of medication events with ATC codes.*

Description

An artificial dataset containing medication events (one per row) for 16 patients (1564 events in total), containing ATC codes. This dataset is derived from the durcomp datasets using the compute_event_durations function. See @med.events for more details.

Usage

```
med.events.ATC
```

Format

A data frame with 1564 rows and 7 variables:

PATIENT_ID the patient unique identifier.

DATE the medication event date.

DURATION the duration in days.

PERDAY the daily dosage.

CATEGORY the ATC code.

CATEGORY_L1 explicitation of the first field of the ATC code (e.g., "A"="ALIMENTARY TRACT AND METABOLISM").

CATEGORY_L2 explicitation of the first and second fields of the ATC code (e.g., "A02"="DRUGS FOR ACID RELATED DISORDERS").

med.groups *Example of medication groups.*

Description

An example defining 6 medication groups for med.events.ATC. It is a *named character vector*, where the names are the medication group unique *names* (e.g., "Vitamines") and the elements are the medication group *definitions* (e.g., "(CATEGORY_L2 == 'VITAMINS')"). The definitions are R logical expressions using *column names* and *values* that appear in the dataset, as well as references to other medication groups using the construction *"NAME"*.

Usage

```
med.groups
```

Format

An object of class character of length 6.

Details

In the above example, "CATEGORY_L2" is a column name in the `med.events.ATC` dataset, and 'VITAMINS' one of its possible values, and which selects all events that have prescribed ATC codes "A11" (aka "VITAMINS"). Another example is "NotVita" defined as "(!Vitamines)", which selects all events that do not have Vitamines prescribed.

For more details, please see the acompanying vignette.

`plot.CMA0` *Plot CMA0 objects.*

Description

Plots the events (prescribing or dispensing) data encapsulated in a basic CMA0 object.

Usage

```
## S3 method for class 'CMA0'
plot(
  x,
  ...,
  patients.to.plot = NULL,
  duration = NA,
  align.all.patients = FALSE,
  align.first.event.at.zero = FALSE,
  show.period = c("dates", "days")[2],
  period.in.days = 90,
  show.legend = TRUE,
  legend.x = "right",
  legend.y = "bottom",
  legend.bkg.opacity = 0.5,
  legend.cex = 0.75,
  legend.cex.title = 1,
  cex = 1,
  cex.axis = 0.75,
  cex.lab = 1,
  xlab = c(dates = "Date", days = "Days"),
  ylab = c(withoutCMA = "patient", withCMA = "patient (& CMA)"),
  title = c(aligned = "Event patterns (all patients aligned)", notaligned =
    "Event patterns"),
  col.cats = rainbow,
  unspecified.category.label = "drug",
  medication.groups.to.plot = NULL,
  medication.groups.separator.show = TRUE,
  medication.groups.separator.lty = "solid",
  medication.groups.separator.lwd = 2,
  medication.groups.separator.color = "blue",
```

```
  medication.groups.allother.label = "*",
  lty.event = "solid",
  lwd.event = 2,
  pch.start.event = 15,
  pch.end.event = 16,
  plot.events.vertically.displaced = TRUE,
  print.dose = FALSE,
  cex.dose = 0.75,
  print.dose.outline.col = "white",
  print.dose.centered = FALSE,
  plot.dose = FALSE,
  lwd.event.max.dose = 8,
  plot.dose.lwd.across.medication.classes = FALSE,
  col.continuation = "black",
  lty.continuation = "dotted",
  lwd.continuation = 1,
  col.na = "lightgray",
  highlight.followup.window = TRUE,
  followup.window.col = "green",
  highlight.observation.window = TRUE,
  observation.window.col = "yellow",
  observation.window.density = 35,
  observation.window.angle = -30,
  observation.window.opacity = 0.3,
  alternating.bands.cols = c("white", "gray95"),
  rotate.text = -60,
  force.draw.text = FALSE,
  bw.plot = FALSE,
  min.plot.size.in.characters.horiz = 0,
  min.plot.size.in.characters.vert = 0,
  suppress.warnings = FALSE,
  max.patients.to.plot = 100,
  export.formats = NULL,
  export.formats.fileprefix = "AdhereR-plot",
  export.formats.height = NA,
  export.formats.width = NA,
  export.formats.save.svg.placeholder = TRUE,
  export.formats.svg.placeholder.type = c("jpg", "png", "webp")[2],
  export.formats.svg.placeholder.embed = FALSE,
  export.formats.html.template = NULL,
  export.formats.html.javascript = NULL,
  export.formats.html.css = NULL,
  export.formats.directory = NA,
  generate.R.plot = TRUE,
  do.not.draw.plot = FALSE
)
```

Arguments

x	A CMA0 or derived object, representing the CMA to plot
...	other possible parameters
patients.to.plot	A vector of *strings* containing the list of patient IDs to plot (a subset of those in the cma object), or NULL for all
duration	A *number*, the total duration (in days) of the whole period to plot; in NA it is automatically determined from the event data such that the whole dataset fits.
align.all.patients	*Logical*, should all patients be aligned (i.e., the actual dates are discarded and all plots are relative to the earliest date)?
align.first.event.at.zero	*Logical*, should the first event be placed at the origin of the time axis (at 0)?
show.period	A *string*, if "dates" show the actual dates at the regular grid intervals, while for "days" (the default) shows the days since the beginning; if align.all.patients == TRUE, show.period is taken as "days".
period.in.days	The *number* of days at which the regular grid is drawn (or 0 for no grid).
show.legend	*Logical*, should the legend be drawn?
legend.x	The position of the legend on the x axis; can be "left", "right" (default), or a *numeric* value.
legend.y	The position of the legend on the y axis; can be "bottom" (default), "top", or a *numeric* value.
legend.bkg.opacity	A *number* between 0.0 and 1.0 specifying the opacity of the legend background.
cex, cex.axis, cex.lab, legend.cex, legend.cex.title	*numeric* values specifying the cex of the various types of text.
xlab	Named vector of x-axis labels to show for the two types of periods ("days" and "dates"), or a single value for both, or NULL for nothing.
ylab	Named vector of y-axis labels to show without and with CMA estimates, or a single value for both, or NULL for nonthing.
title	Named vector of titles to show for and without alignment, or a single value for both, or NULL for nonthing.
col.cats	A *color* or a *function* that specifies the single colour or the colour palette used to plot the different medication; by default rainbow, but we recommend, whenever possible, a colorblind-friendly palette such as viridis or colorblind_pal.
unspecified.category.label	A *string* giving the name of the unspecified (generic) medication category.
medication.groups.to.plot	the names of the medication groups to plot or NULL (the default) for all.
medication.groups.separator.show	a *boolean*, if TRUE (the default) visually mark the medication groups the belong to the same patient, using horizontal lines and alternating vertical lines.

`medication.groups.separator.lty, medication.groups.separator.lwd, medication.groups.separator.color`
: graphical parameters (line type, line width and colour describing the visual marking og medication groups as beloning to the same patient.

`medication.groups.allother.label`
: a *string* giving the label to use for the implicit `__ALL_OTHERS__` medication group (defaults to "*").

`lty.event, lwd.event, pch.start.event, pch.end.event`
: The style of the event (line style, width, and start and end symbols).

`plot.events.vertically.displaced`
: Should consecutive events be plotted on separate rows (i.e., separated vertically, the default) or on the same row?

`print.dose`
: *Logical*, should the daily dose be printed as text?

`cex.dose`
: *Numeric*, if daily dose is printed, what text size to use?

`print.dose.outline.col`
: If `NA`, don't print dose text with outline, otherwise a color name/code for the outline.

`print.dose.centered`
: *Logical*, print the daily dose centered on the segment or slightly below it?

`plot.dose`
: *Logical*, should the daily dose be indicated through segment width?

`lwd.event.max.dose`
: *Numeric*, the segment width corresponding to the maximum daily dose (must be >= lwd.event but not too big either).

`plot.dose.lwd.across.medication.classes`
: *Logical*, if `TRUE`, the line width of the even is scaled relative to all medication classes (i.e., relative to the global minimum and maximum doses), otherwise it is scale relative only to its medication class.

`col.continuation, lty.continuation, lwd.continuation`
: The style of the "continuation" lines connecting consecutive events (colour, line style and width).

`col.na`
: The colour used for missing event data.

`highlight.followup.window`
: *Logical*, should the follow-up window be plotted?

`followup.window.col`
: The follow-up window's colour.

`highlight.observation.window`
: *Logical*, should the observation window be plotted?

`observation.window.col, observation.window.density, observation.window.angle, observation.window.opac`
: Attributes of the observation window (colour, shading density, angle and opacity).

`alternating.bands.cols`
: The colors of the alternating vertical bands distinguishing the patients; can be `NULL` = don't draw the bandes; or a vector of colors.

`rotate.text`
: *Numeric*, the angle by which certain text elements (e.g., axis labels) should be rotated.

force.draw.text
: *Logical*, if TRUE, always draw text even if too big or too small

bw.plot
: *Logical*, should the plot use grayscale only (i.e., the gray.colors function)?

min.plot.size.in.characters.horiz, min.plot.size.in.characters.vert
: *Numeric*, the minimum size of the plotting surface in characters; horizontally (min.plot.size.in.characters.horiz) refers to the the whole duration of the events to plot; vertically (min.plot.size.in.characters.vert) refers to a single event. If the plotting is too small, possible solutions might be: if within RStudio, try to enlarge the "Plots" panel, or (also valid outside RStudio but not if using RStudio server start a new plotting device (e.g., using X11(), quartz() or windows(), depending on OS) or (works always) save to an image (e.g., jpeg(...); ...; dev.off()) and display it in a viewer.

suppress.warnings
: *Logical*: show or hide the warnings?

max.patients.to.plot
: *Numeric*, the maximum patients to attempt to plot.

export.formats
: a *string* giving the formats to export the figure to (by default NULL, meaning no exporting); can be any combination of "svg" (just an SVG file), "html" (SVG + HTML + CSS + JavaScript, all embedded within one HTML document), "jpg", "png", "webp", "ps" or "pdf".

export.formats.fileprefix
: a *string* giving the file name prefix for the exported formats (defaults to "AdhereR-plot").

export.formats.height, export.formats.width
: *numbers* giving the desired dimensions (in pixels) for the exported figure (defaults to sane values if NA).

export.formats.save.svg.placeholder
: a *logical*, if TRUE, save an image placeholder of type given by export.formats.svg.placeholder.type for the SVG image.

export.formats.svg.placeholder.type
: a *string*, giving the type of placeholder for the SVG image to save; can be "jpg", "png" (the default) or "webp".

export.formats.svg.placeholder.embed
: a *logical*, if TRUE, embed the placeholder image in the HTML document (if any) using base64 encoding, otherwise (the default) leave it as an external image file (works only when an HTML document is exported and only for JPEG or PNG images.

export.formats.html.template, export.formats.html.javascript, export.formats.html.css
: *character strings* or NULL (the default) giving the path to the HTML, JavaScript and CSS templates, respectively, to be used when generating the HTML+CSS semi-interactive plots; when NULL, the default ones included with the package will be used. If you decide to define new templates please use the default ones for inspiration and note that future version are not guaranteed to be backwards compatible!

export.formats.directory
: a *string*; if exporting, which directory to export to; if NA (the default), creates the files in a temporary directory.

`generate.R.plot`
: a *logical*, if `TRUE` (the default), generate the standard (base `R`) plot for plotting within `R`.

`do.not.draw.plot`
: a *logical*, if `TRUE` (*not* the default), does not draw the plot itself, but only the legend (if `show.legend` is `TRUE`) at coordinates (0,0) irrespective of the given legend coordinates. This is intended to allow (together with the `get.legend.plotting.area()` function) the separate plotting of the legend.

Details

The x-axis represents time (either in days since the earliest date or as actual dates), with consecutive events represented as ascending on the y-axis.

Each event is represented as a segment with style `lty.event` and line width `lwd.event` starting with a `pch.start.event` and ending with a `pch.end.event` character, coloured with a unique color as given by `col.cats`, extending from its start date until its end date. Consecutive events are thus represented on consecutive levels of the y-axis and are connected by a "continuation" line with `col.continuation` colour, `lty.continuation` style and `lwd.continuation` width; these continuation lines are purely visual guides helping to perceive the sequence of events, and carry no information about the availability of medication in this interval.

When several patients are displayed on the same plot, they are organized vertically, and alternating bands (white and gray) help distinguish consecutive patients. Implicitly, all patients contained in the `cma` object will be plotted, but the `patients.to.plot` parameter allows the selection of a subset of patients.

Examples

```
cma0 <- CMA0(data=med.events,
             ID.colname="PATIENT_ID",
             event.date.colname="DATE",
             event.duration.colname="DURATION",
             event.daily.dose.colname="PERDAY",
             medication.class.colname="CATEGORY",
             followup.window.start=0,
             followup.window.start.unit="days",
             followup.window.duration=2*365,
             followup.window.duration.unit="days",
             observation.window.start=30,
             observation.window.start.unit="days",
             observation.window.duration=365,
             observation.window.duration.unit="days",
             date.format="%m/%d/%Y",
             summary="Base CMA");
plot(cma0, patients.to.plot=c("1","2"));
```

plot.CMA1 *Plot CMA0-derived objects.*

Description

Plots the event data and estimated CMA encapsulated in objects derived from CMA0.

Usage

```
## S3 method for class 'CMA1'
plot(
  x,
  ...,
  patients.to.plot = NULL,
  duration = NA,
  align.all.patients = FALSE,
  align.first.event.at.zero = FALSE,
  show.period = c("dates", "days")[2],
  period.in.days = 90,
  show.legend = TRUE,
  legend.x = "right",
  legend.y = "bottom",
  legend.bkg.opacity = 0.5,
  legend.cex = 0.75,
  legend.cex.title = 1,
  cex = 1,
  cex.axis = 0.75,
  cex.lab = 1,
  show.cma = TRUE,
  col.cats = rainbow,
  unspecified.category.label = "drug",
  medication.groups.to.plot = NULL,
  medication.groups.separator.show = TRUE,
  medication.groups.separator.lty = "solid",
  medication.groups.separator.lwd = 2,
  medication.groups.separator.color = "blue",
  medication.groups.allother.label = "*",
  lty.event = "solid",
  lwd.event = 2,
  pch.start.event = 15,
  pch.end.event = 16,
  show.event.intervals = TRUE,
  col.na = "lightgray",
  print.CMA = TRUE,
  CMA.cex = 0.5,
  plot.CMA = TRUE,
  CMA.plot.ratio = 0.1,
```

```
  CMA.plot.col = "lightgreen",
  CMA.plot.border = "darkgreen",
  CMA.plot.bkg = "aquamarine",
  CMA.plot.text = CMA.plot.border,
  highlight.followup.window = TRUE,
  followup.window.col = "green",
  highlight.observation.window = TRUE,
  observation.window.col = "yellow",
  observation.window.density = 35,
  observation.window.angle = -30,
  observation.window.opacity = 0.3,
  show.real.obs.window.start = TRUE,
  real.obs.window.density = 35,
  real.obs.window.angle = 30,
  print.dose = FALSE,
  cex.dose = 0.75,
  print.dose.outline.col = "white",
  print.dose.centered = FALSE,
  plot.dose = FALSE,
  lwd.event.max.dose = 8,
  plot.dose.lwd.across.medication.classes = FALSE,
  alternating.bands.cols = c("white", "gray95"),
  bw.plot = FALSE,
  rotate.text = -60,
  force.draw.text = FALSE,
  min.plot.size.in.characters.horiz = 0,
  min.plot.size.in.characters.vert = 0,
  max.patients.to.plot = 100,
  export.formats = NULL,
  export.formats.fileprefix = "AdhereR-plot",
  export.formats.height = NA,
  export.formats.width = NA,
  export.formats.save.svg.placeholder = TRUE,
  export.formats.svg.placeholder.type = c("jpg", "png", "webp")[2],
  export.formats.svg.placeholder.embed = FALSE,
  export.formats.directory = NA,
  export.formats.html.template = NULL,
  export.formats.html.javascript = NULL,
  export.formats.html.css = NULL,
  generate.R.plot = TRUE,
  do.not.draw.plot = FALSE
)

## S3 method for class 'CMA2'
plot(...)

## S3 method for class 'CMA3'
plot(...)
```

```
## S3 method for class 'CMA4'
plot(...)

## S3 method for class 'CMA5'
plot(...)

## S3 method for class 'CMA6'
plot(...)

## S3 method for class 'CMA7'
plot(...)

## S3 method for class 'CMA8'
plot(...)

## S3 method for class 'CMA9'
plot(...)
```

Arguments

x	A CMA0 or derived object, representing the CMA to plot
...	other possible parameters
patients.to.plot	A vector of *strings* containing the list of patient IDs to plot (a subset of those in the cma object), or NULL for all
duration	A *number*, the total duration (in days) of the whole period to plot; in NA it is automatically determined from the event data such that the whole dataset fits.
align.all.patients	*Logical*, should all patients be aligned (i.e., the actual dates are discarded and all plots are relative to the earliest date)?
align.first.event.at.zero	*Logical*, should the first event be placed at the origin of the time axis (at 0)?
show.period	A *string*, if "dates" show the actual dates at the regular grid intervals, while for "days" (the default) shows the days since the beginning; if align.all.patients == TRUE, show.period is taken as "days".
period.in.days	The *number* of days at which the regular grid is drawn (or 0 for no grid).
show.legend	*Logical*, should the legend be drawn?
legend.x	The position of the legend on the x axis; can be "left", "right" (default), or a *numeric* value.
legend.y	The position of the legend on the y axis; can be "bottom" (default), "top", or a *numeric* value.
legend.bkg.opacity	A *number* between 0.0 and 1.0 specifying the opacity of the legend background.
cex, cex.axis, cex.lab, legend.cex, legend.cex.title, CMA.cex	*numeric* values specifying the cex of the various types of text.

show.cma *Logical*, should the CMA type be shown in the title?

col.cats A *color* or a *function* that specifies the single colour or the colour palette used to plot the different medication; by default rainbow, but we recommend, whenever possible, a colorblind-friendly palette such as viridis or colorblind_pal.

unspecified.category.label
A *string* giving the name of the unspecified (generic) medication category.

medication.groups.to.plot
the names of the medication groups to plot or NULL (the default) for all.

medication.groups.separator.show
a *boolean*, if TRUE (the default) visually mark the medication groups the belong to the same patient, using horizontal lines and alternating vertical lines.

medication.groups.separator.lty, medication.groups.separator.lwd, medication.groups.separator.color
graphical parameters (line type, line width and colour describing the visual marking og medication groups as beloning to the same patient.

medication.groups.allother.label
a *string* giving the label to use for the implicit __ALL_OTHERS__ medication group (defaults to "*").

lty.event, lwd.event, pch.start.event, pch.end.event
The style of the event (line style, width, and start and end symbols).

show.event.intervals
Logical, should the actual event intervals be shown?

col.na The colour used for missing event data.

print.CMA *Logical*, should the CMA values be printed?

plot.CMA *Logical*, should the CMA values be represented graphically?

CMA.plot.ratio A *number*, the proportion of the total horizontal plot space to be allocated to the CMA plot.

CMA.plot.col, CMA.plot.border, CMA.plot.bkg, CMA.plot.text
Strings giving the colours of the various components of the CMA plot.

highlight.followup.window
Logical, should the follow-up window be plotted?

followup.window.col
The follow-up window's colour.

highlight.observation.window
Logical, should the observation window be plotted?

observation.window.col, observation.window.density, observation.window.angle, observation.window.opac
Attributes of the observation window (colour, shading density, angle and opacity).

show.real.obs.window.start, real.obs.window.density, real.obs.window.angle
For some CMAs, the observation window might be adjusted, in which case should it be plotted and with that attributes?

print.dose *Logical*, should the daily dose be printed as text?

cex.dose *Numeric*, if daily dose is printed, what text size to use?

print.dose.outline.col
: If NA, don't print dose text with outline, otherwise a color name/code for the outline.

print.dose.centered
: *Logical*, print the daily dose centered on the segment or slightly below it?

plot.dose
: *Logical*, should the daily dose be indicated through segment width?

lwd.event.max.dose
: *Numeric*, the segment width corresponding to the maximum daily dose (must be >= lwd.event but not too big either).

plot.dose.lwd.across.medication.classes
: *Logical*, if TRUE, the line width of the even is scaled relative to all medication classes (i.e., relative to the global minimum and maximum doses), otherwise it is scale relative only to its medication class.

alternating.bands.cols
: The colors of the alternating vertical bands distinguishing the patients; can be NULL = don't draw the bandes; or a vector of colors.

bw.plot
: *Logical*, should the plot use grayscale only (i.e., the gray.colors function)?

rotate.text
: *Numeric*, the angle by which certain text elements (e.g., axis labels) should be rotated.

force.draw.text
: *Logical*, if TRUE, always draw text even if too big or too small

min.plot.size.in.characters.horiz, min.plot.size.in.characters.vert
: *Numeric*, the minimum size of the plotting surface in characters; horizontally (min.plot.size.in.characters.horiz) refers to the the whole duration of the events to plot; vertically (min.plot.size.in.characters.vert) refers to a single event. If the plotting is too small, possible solutions might be: if within RStudio, try to enlarge the "Plots" panel, or (also valid outside RStudio but not if using RStudio server start a new plotting device (e.g., using X11(), quartz() or windows(), depending on OS) or (works always) save to an image (e.g., jpeg(...); ...; dev.off()) and display it in a viewer.

max.patients.to.plot
: *Numeric*, the maximum patients to attempt to plot.

export.formats
: a *string* giving the formats to export the figure to (by default NULL, meaning no exporting); can be any combination of "svg" (just an SVG file), "html" (SVG + HTML + CSS + JavaScript, all embedded within one HTML document), "jpg", "png", "webp", "ps" or "pdf".

export.formats.fileprefix
: a *string* giving the file name prefix for the exported formats (defaults to "AdhereR-plot").

export.formats.height, export.formats.width
: *numbers* giving the desired dimensions (in pixels) for the exported figure (defaults to sane values if NA).

export.formats.save.svg.placeholder
: a *logical*, if TRUE, save an image placeholder of type given by export.formats.svg.placeholder.typ
for the SVG image.

`export.formats.svg.placeholder.type`
: a *string*, giving the type of placeholder for the SVG image to save; can be "jpg", "png" (the default) or "webp".

`export.formats.svg.placeholder.embed`
: a *logical*, if TRUE, embed the placeholder image in the HTML document (if any) using base64 encoding, otherwise (the default) leave it as an external image file (works only when an HTML document is exported and only for JPEG or PNG images.

`export.formats.directory`
: a *string*; if exporting, which directory to export to; if NA (the default), creates the files in a temporary directory.

`export.formats.html.template, export.formats.html.javascript, export.formats.html.css`
: *character strings* or NULL (the default) giving the path to the HTML, JavaScript and CSS templates, respectively, to be used when generating the HTML+CSS semi-interactive plots; when NULL, the default ones included with the package will be used. If you decide to define new templates please use the default ones for inspiration and note that future version are not guaranteed to be backwards compatible!

`generate.R.plot`
: a *logical*, if TRUE (the default), generate the standard (base R) plot for plotting within R.

`do.not.draw.plot`
: a *logical*, if TRUE (*not* the default), does not draw the plot itself, but only the legend (if `show.legend` is TRUE) at coordinates (0,0) irrespective of the given legend coordinates. This is intended to allow (together with the `get.legend.plotting.area()` function) the separate plotting of the legend.

Details

Please note that this function plots objects inheriting from `CMA0` but not objects of type `CMA0` itself (these are plotted by `plot.CMA0`).

The x-axis represents time (either in days since the earliest date or as actual dates), with consecutive events represented as ascending on the y-axis.

Each event is represented as a segment with style `lty.event` and line width `lwd.event` starting with a `pch.start.event` and ending with a `pch.end.event` character, coloured with a unique color as given by `col.cats`, extending from its start date until its end date. Superimposed on these are shown the event intervals and gap days as estimated by the particular CMA method, more precisely plotting the start and end of the available events as solid filled-in rectangles, and the event gaps as shaded rectangles.

The follow-up and the observation windows are plotted as an empty rectangle and as shaded rectangle, respectively (for some CMAs the observation window might be adjusted in which case the adjustment may also be plotted using a different shading).

The CMA estimates can be visually represented as well in the left side of the figure using bars (sometimes the estimates can go above 100%, in which case the maximum possible bar filling is adjusted to reflect this).

When several patients are displayed on the same plot, they are organized vertically, and alternating bands (white and gray) help distinguish consecutive patients. Implicitely, all patients contained in

the cma object will be plotted, but the patients.to.plot parameter allows the selection of a subset of patients.

Finally, the y-axis shows the patient ID and possibly the CMA estimate as well.

Examples

```
cma1 <- CMA1(data=med.events,
             ID.colname="PATIENT_ID",
             event.date.colname="DATE",
             event.duration.colname="DURATION",
             followup.window.start=30,
             observation.window.start=30,
             observation.window.duration=365,
             date.format="%m/%d/%Y"
            );
plot(cma1, patients.to.plot=c("1","2"));
```

---

plot.CMA_per_episode *Plot CMA_per_episode and CMA_sliding_window objects.*

Description

Plots the event data and the estimated CMA per treatment episode and sliding window, respectively.

Usage

```
## S3 method for class 'CMA_per_episode'
plot(
  x,
  patients.to.plot = NULL,
  duration = NA,
  align.all.patients = FALSE,
  align.first.event.at.zero = FALSE,
  show.period = c("dates", "days")[2],
  period.in.days = 90,
  show.legend = TRUE,
  legend.x = "right",
  legend.y = "bottom",
  legend.bkg.opacity = 0.5,
  legend.cex = 0.75,
  legend.cex.title = 1,
  cex = 1,
  cex.axis = 0.75,
  cex.lab = 1,
  show.cma = TRUE,
  xlab = c(dates = "Date", days = "Days"),
  ylab = c(withoutCMA = "patient", withCMA = "patient (& CMA)"),
  title = c(aligned = "Event patterns (all patients aligned)", notaligned =
```

```
  "Event patterns"),
 col.cats = rainbow,
 unspecified.category.label = "drug",
 medication.groups.to.plot = NULL,
 medication.groups.separator.show = TRUE,
 medication.groups.separator.lty = "solid",
 medication.groups.separator.lwd = 2,
 medication.groups.separator.color = "blue",
 medication.groups.allother.label = "*",
 lty.event = "solid",
 lwd.event = 2,
 pch.start.event = 15,
 pch.end.event = 16,
 show.event.intervals = FALSE,
 show.overlapping.event.intervals = c("first", "last", "min gap", "max gap",
  "average")[1],
 plot.events.vertically.displaced = TRUE,
 print.dose = FALSE,
 cex.dose = 0.75,
 print.dose.outline.col = "white",
 print.dose.centered = FALSE,
 plot.dose = FALSE,
 lwd.event.max.dose = 8,
 plot.dose.lwd.across.medication.classes = FALSE,
 col.na = "lightgray",
 col.continuation = "black",
 lty.continuation = "dotted",
 lwd.continuation = 1,
 print.CMA = TRUE,
 CMA.cex = 0.5,
 plot.CMA = TRUE,
 plot.CMA.as.histogram = TRUE,
 plot.partial.CMAs.as = c("stacked", "overlapping", "timeseries")[1],
 plot.partial.CMAs.as.stacked.col.bars = "gray90",
 plot.partial.CMAs.as.stacked.col.border = "gray30",
 plot.partial.CMAs.as.stacked.col.text = "black",
 plot.partial.CMAs.as.timeseries.vspace = 7,
 plot.partial.CMAs.as.timeseries.start.from.zero = TRUE,
 plot.partial.CMAs.as.timeseries.col.dot = "darkblue",
 plot.partial.CMAs.as.timeseries.col.interval = "gray70",
 plot.partial.CMAs.as.timeseries.col.text = "firebrick",
plot.partial.CMAs.as.timeseries.interval.type = c("none", "segments", "arrows",
  "lines", "rectangles")[2],
 plot.partial.CMAs.as.timeseries.lwd.interval = 1,
 plot.partial.CMAs.as.timeseries.alpha.interval = 0.25,
 plot.partial.CMAs.as.timeseries.show.0perc = TRUE,
 plot.partial.CMAs.as.timeseries.show.100perc = FALSE,
 plot.partial.CMAs.as.overlapping.alternate = TRUE,
```

```
  plot.partial.CMAs.as.overlapping.col.interval = "gray70",
  plot.partial.CMAs.as.overlapping.col.text = "firebrick",
  CMA.plot.ratio = 0.1,
  CMA.plot.col = "lightgreen",
  CMA.plot.border = "darkgreen",
  CMA.plot.bkg = "aquamarine",
  CMA.plot.text = CMA.plot.border,
  highlight.followup.window = TRUE,
  followup.window.col = "green",
  highlight.observation.window = TRUE,
  observation.window.col = "yellow",
  observation.window.opacity = 0.3,
  print.episode.or.sliding.window = FALSE,
  alternating.bands.cols = c("white", "gray95"),
  bw.plot = FALSE,
  rotate.text = -60,
  force.draw.text = FALSE,
  min.plot.size.in.characters.horiz = 0,
  min.plot.size.in.characters.vert = 0,
  max.patients.to.plot = 100,
  export.formats = NULL,
  export.formats.fileprefix = "AdhereR-plot",
  export.formats.height = NA,
  export.formats.width = NA,
  export.formats.save.svg.placeholder = TRUE,
  export.formats.svg.placeholder.type = c("jpg", "png", "webp")[2],
  export.formats.svg.placeholder.embed = FALSE,
  export.formats.html.template = NULL,
  export.formats.html.javascript = NULL,
  export.formats.html.css = NULL,
  export.formats.directory = NA,
  generate.R.plot = TRUE,
  do.not.draw.plot = FALSE,
  suppress.warnings = FALSE,
  ...
)

## S3 method for class 'CMA_sliding_window'
plot(
  x,
  patients.to.plot = NULL,
  duration = NA,
  align.all.patients = FALSE,
  align.first.event.at.zero = FALSE,
  show.period = c("dates", "days")[2],
  period.in.days = 90,
  show.legend = TRUE,
  legend.x = "right",
```

```
  legend.y = "bottom",
  legend.bkg.opacity = 0.5,
  legend.cex = 0.75,
  legend.cex.title = 1,
  cex = 1,
  cex.axis = 0.75,
  cex.lab = 1,
  show.cma = TRUE,
  xlab = c(dates = "Date", days = "Days"),
  ylab = c(withoutCMA = "patient", withCMA = "patient (& CMA)"),
  title = c(aligned = "Event patterns (all patients aligned)", notaligned =
    "Event patterns"),
  col.cats = rainbow,
  unspecified.category.label = "drug",
  medication.groups.to.plot = NULL,
  medication.groups.separator.show = TRUE,
  medication.groups.separator.lty = "solid",
  medication.groups.separator.lwd = 2,
  medication.groups.separator.color = "blue",
  medication.groups.allother.label = "*",
  lty.event = "solid",
  lwd.event = 2,
  pch.start.event = 15,
  pch.end.event = 16,
  show.event.intervals = FALSE,
  show.overlapping.event.intervals = c("first", "last", "min gap", "max gap",
    "average")[1],
  plot.events.vertically.displaced = TRUE,
  print.dose = FALSE,
  cex.dose = 0.75,
  print.dose.outline.col = "white",
  print.dose.centered = FALSE,
  plot.dose = FALSE,
  lwd.event.max.dose = 8,
  plot.dose.lwd.across.medication.classes = FALSE,
  col.na = "lightgray",
  col.continuation = "black",
  lty.continuation = "dotted",
  lwd.continuation = 1,
  print.CMA = TRUE,
  CMA.cex = 0.5,
  plot.CMA = TRUE,
  plot.CMA.as.histogram = TRUE,
  plot.partial.CMAs.as = c("stacked", "overlapping", "timeseries")[1],
  plot.partial.CMAs.as.stacked.col.bars = "gray90",
  plot.partial.CMAs.as.stacked.col.border = "gray30",
  plot.partial.CMAs.as.stacked.col.text = "black",
  plot.partial.CMAs.as.timeseries.vspace = 7,
```

```
  plot.partial.CMAs.as.timeseries.start.from.zero = TRUE,
  plot.partial.CMAs.as.timeseries.col.dot = "darkblue",
  plot.partial.CMAs.as.timeseries.col.interval = "gray70",
  plot.partial.CMAs.as.timeseries.col.text = "firebrick",
 plot.partial.CMAs.as.timeseries.interval.type = c("none", "segments", "arrows",
    "lines", "rectangles")[2],
  plot.partial.CMAs.as.timeseries.lwd.interval = 1,
  plot.partial.CMAs.as.timeseries.alpha.interval = 0.25,
  plot.partial.CMAs.as.timeseries.show.0perc = TRUE,
  plot.partial.CMAs.as.timeseries.show.100perc = FALSE,
  plot.partial.CMAs.as.overlapping.alternate = TRUE,
  plot.partial.CMAs.as.overlapping.col.interval = "gray70",
  plot.partial.CMAs.as.overlapping.col.text = "firebrick",
  CMA.plot.ratio = 0.1,
  CMA.plot.col = "lightgreen",
  CMA.plot.border = "darkgreen",
  CMA.plot.bkg = "aquamarine",
  CMA.plot.text = CMA.plot.border,
  highlight.followup.window = TRUE,
  followup.window.col = "green",
  highlight.observation.window = TRUE,
  observation.window.col = "yellow",
  observation.window.opacity = 0.3,
  print.episode.or.sliding.window = FALSE,
  alternating.bands.cols = c("white", "gray95"),
  bw.plot = FALSE,
  rotate.text = -60,
  force.draw.text = FALSE,
  min.plot.size.in.characters.horiz = 0,
  min.plot.size.in.characters.vert = 0,
  max.patients.to.plot = 100,
  export.formats = NULL,
  export.formats.fileprefix = "AdhereR-plot",
  export.formats.height = NA,
  export.formats.width = NA,
  export.formats.save.svg.placeholder = TRUE,
  export.formats.svg.placeholder.type = c("jpg", "png", "webp")[2],
  export.formats.svg.placeholder.embed = FALSE,
  export.formats.html.template = NULL,
  export.formats.html.javascript = NULL,
  export.formats.html.css = NULL,
  export.formats.directory = NA,
  generate.R.plot = TRUE,
  do.not.draw.plot = FALSE,
  suppress.warnings = FALSE,
  ...
)
```

Arguments

x	A CMA0 or derived object, representing the CMA to plot
patients.to.plot	A vector of *strings* containing the list of patient IDs to plot (a subset of those in the cma object), or NULL for all
duration	A *number*, the total duration (in days) of the whole period to plot; in NA it is automatically determined from the event data such that the whole dataset fits.
align.all.patients	*Logical*, should all patients be aligned (i.e., the actual dates are discarded and all plots are relative to the earliest date)?
align.first.event.at.zero	*Logical*, should the first event be placed at the origin of the time axis (at 0)?
show.period	A *string*, if "dates" show the actual dates at the regular grid intervals, while for "days" (the default) shows the days since the beginning; if align.all.patients == TRUE, show.period is taken as "days".
period.in.days	The *number* of days at which the regular grid is drawn (or 0 for no grid).
show.legend	*Logical*, should the legend be drawn?
legend.x	The position of the legend on the x axis; can be "left", "right" (default), or a *numeric* value.
legend.y	The position of the legend on the y axis; can be "bottom" (default), "top", or a *numeric* value.
legend.bkg.opacity	A *number* between 0.0 and 1.0 specifying the opacity of the legend background.
legend.cex, legend.cex.title	The legend and legend title font sizes.
cex, cex.axis, cex.lab	*numeric* values specifying the cex of the various types of text.
show.cma	*Logical*, should the CMA type be shown in the title?
xlab	Named vector of x-axis labels to show for the two types of periods ("days" and "dates"), or a single value for both, or NULL for nothing.
ylab	Named vector of y-axis labels to show without and with CMA estimates, or a single value for both, or NULL for nonthing.
title	Named vector of titles to show for and without alignment, or a single value for both, or NULL for nonthing.
col.cats	A *color* or a *function* that specifies the single colour or the colour palette used to plot the different medication; by default rainbow, but we recommend, whenever possible, a colorblind-friendly palette such as viridis or colorblind_pal.
unspecified.category.label	A *string* giving the name of the unspecified (generic) medication category.
medication.groups.to.plot	the names of the medication groups to plot or NULL (the default) for all.
medication.groups.separator.show	a *boolean*, if TRUE (the default) visually mark the medication groups the belong to the same patient, using horizontal lines and alternating vertical lines.

`medication.groups.separator.lty`, `medication.groups.separator.lwd`, `medication.groups.separator.color`
: graphical parameters (line type, line width and colour describing the visual marking og medication groups as beloning to the same patient.

`medication.groups.allother.label`
: a *string* giving the label to use for the implicit `__ALL_OTHERS__` medication group (defaults to "*").

`lty.event`, `lwd.event`, `pch.start.event`, `pch.end.event`
: The style of the event (line style, width, and start and end symbols).

`show.event.intervals`
: *Logical*, should the actual event intervals be shown? As per-episode and sliding windows might have overlapping intervals, it is better not to show them by default (`FALSE`).

`show.overlapping.event.intervals`
: specifies how to plot the event intervals that appear in multiple sliding windows or episodes. We can plot how thye look in the *first* sliding window or episode (the default), how they appear in the *last*, pick the one that minimizes the gap (*min gap*) or maximizes it (*max gap*), or compute their *average* across all sliding windows or episodes containing them.

`plot.events.vertically.displaced`
: Should consecutive events be plotted on separate rows (i.e., separated vertically, the default) or on the same row?

`print.dose`, `cex.dose`, `print.dose.outline.col`, `print.dose.centered`
: Print daily dose as a number and, if so, how (color, size, position...).

`plot.dose`, `lwd.event.max.dose`, `plot.dose.lwd.across.medication.classes`
: Show dose through the width of the event lines and, if so, what the maximum width should be, and should this maximum be by medication class or overall.

`col.na`
: The colour used for missing event data.

`col.continuation`, `lty.continuation`, `lwd.continuation`
: The color, style and width of the contuniation lines connecting consecutive events.

`print.CMA`
: *Logical*, should the CMA values be printed?

`CMA.cex`
: ... and, if printed, what cex (*numeric*) to use?

`plot.CMA`
: *Logical*, should the distribution of the CMA values across episodes/sliding windows be plotted? If `TRUE` (the default), the distribution is shown on the left-hand side of the plot, otherwise it is not.

`plot.CMA.as.histogram`
: *Logical*, should the CMA plot be a histogram or a (truncated) density plot? Please note that it is TRUE by deafult for CMA_per_episode and FALSE for CMA_sliding_window, because usually there are more sliding windows than episodes. Also, the density estimate cannot be estimated for less than three different values.

`plot.partial.CMAs.as`
: Should the partial CMAs be plotted? Possible values are "stacked", "overlapping" or "timeseries", or `NULL` for no partial CMA plots. Please note that `plot.CMA` and `plot.partial.CMAs.as` are independent of each other.

plot.partial.CMAs.as.stacked.col.bars, plot.partial.CMAs.as.stacked.col.border, plot.partial.CMAs.as
: If plotting the partial CMAs as stacked bars, define their graphical attributes.

plot.partial.CMAs.as.timeseries.vspace, plot.partial.CMAs.as.timeseries.start.from.zero, plot.partia
: If plotting the partial CMAs as imeseries, these are their graphical attributes.

plot.partial.CMAs.as.overlapping.alternate, plot.partial.CMAs.as.overlapping.col.interval, plot.part
: If plotting the partial CMAs as overlapping segments, these are their graphical attributes.

CMA.plot.ratio
: A *number*, the proportion of the total horizontal plot space to be allocated to the CMA plot.

CMA.plot.col, CMA.plot.border, CMA.plot.bkg, CMA.plot.text
: *Strings* giving the colours of the various components of the CMA plot.

highlight.followup.window
: *Logical*, should the follow-up window be plotted?

followup.window.col
: The follow-up window colour.

highlight.observation.window
: *Logical*, should the observation window be plotted?

observation.window.col, observation.window.opacity
: Attributes of the observation window (colour, transparency).

print.episode.or.sliding.window
: *Logical*, should we show which events belong to which episode or sliding window? To work, the CMA must have been constructed with return.mapping.events.episodes or return.mapping.events.sliding.window set to TRUE, respectively.

alternating.bands.cols
: The colors of the alternating vertical bands distinguishing the patients; can be NULL = don't draw the bandes; or a vector of colors.

bw.plot
: *Logical*, should the plot use grayscale only (i.e., the gray.colors function)?

rotate.text
: *Numeric*, the angle by which certain text elements (e.g., axis labels) should be rotated.

force.draw.text
: *Logical*, if TRUE, always draw text even if too big or too small

min.plot.size.in.characters.horiz, min.plot.size.in.characters.vert
: *Numeric*, the minimum size of the plotting surface in characters; horizontally (min.plot.size.in.characters.horiz) refers to the the whole duration of the events to plot; vertically (min.plot.size.in.characters.vert) refers to a single event. If the plotting is too small, possible solutions might be: if within RStudio, try to enlarge the "Plots" panel, or (also valid outside RStudio but not if using RStudio server start a new plotting device (e.g., using X11(), quartz() or windows(), depending on OS) or (works always) save to an image (e.g., jpeg(...); ...; dev.off()) and display it in a viewer.

max.patients.to.plot
: *Numeric*, the maximum patients to attempt to plot.

export.formats
: a *string* giving the formats to export the figure to (by default NULL, meaning no exporting); can be any combination of "svg" (just an SVG file), "html" (SVG + HTML + CSS + JavaScript, all embedded within one HTML document), "jpg", "png", "webp", "ps" or "pdf".

`export.formats.fileprefix`
: a *string* giving the file name prefix for the exported formats (defaults to "AdhereR-plot").

`export.formats.height, export.formats.width`
: *numbers* giving the desired dimensions (in pixels) for the exported figure (defaults to sane values if `NA`).

`export.formats.save.svg.placeholder`
: a *logical*, if TRUE, save an image placeholder of type given by `export.formats.svg.placeholder.type` for the `SVG` image.

`export.formats.svg.placeholder.type`
: a *string*, giving the type of placeholder for the `SVG` image to save; can be "jpg", "png" (the default) or "webp".

`export.formats.svg.placeholder.embed`
: a *logical*, if `TRUE`, embed the placeholder image in the HTML document (if any) using `base64` encoding, otherwise (the default) leave it as an external image file (works only when an `HTML` document is exported and only for `JPEG` or `PNG` images.

`export.formats.html.template, export.formats.html.javascript, export.formats.html.css`
: *character strings* or `NULL` (the default) giving the path to the `HTML`, `JavaScript` and `CSS` templates, respectively, to be used when generating the HTML+CSS semi-interactive plots; when `NULL`, the default ones included with the package will be used. If you decide to define new templates please use the default ones for inspiration and note that future version are not guaranteed to be backwards compatible!

`export.formats.directory`
: a *string*; if exporting, which directory to export to; if `NA` (the default), creates the files in a temporary directory.

`generate.R.plot`
: a *logical*, if `TRUE` (the default), generate the standard (base `R`) plot for plotting within `R`.

`do.not.draw.plot`
: a *logical*, if `TRUE` (*not* the default), does not draw the plot itself, but only the legend (if `show.legend` is `TRUE`) at coordinates (0,0) irrespective of the given legend coordinates. This is intended to allow (together with the `get.legend.plotting.area()` function) the separate plotting of the legend.

`suppress.warnings`
: *Logical*, if `TRUE` don't show any warnings.

`...`
: other parameters (to be passed to the estimation and plotting of the simple CMA)

Details

The x-axis represents time (either in days since the earliest date or as actual dates), with consecutive events represented as ascending on the y-axis.

Each event is represented as a segment with style `lty.event` and line width `lwd.event` starting with a `pch.start.event` and ending with a `pch.end.event` character, coloured with a unique color as given by `col.cats`, extending from its start date until its end date. Consecutive events are thus represented on consecutive levels of the y-axis and are connected by a "continuation" line

with `col.continuation` colour, `lty.continuation` style and `lwd.continuation` width; these continuation lines are purely visual guides helping to perceive the sequence of events, and carry no information about the avilability of medicine in this interval.

Above these, the treatment episodes or the sliding windows are represented in a stacked manner from the earlieast (left, bottom of the stack) to the latest (right, top of the stack), each showing the CMA as percent fill (capped at 100% even if CMA values may be higher) and also as text.

The follow-up and the observation windows are plotted as empty an rectangle and as shaded rectangle, respectively (for some CMAs the observation window might be adjusted in which case the adjustment may also be plotted using a different shading).

The kernel density ("smoothed histogram") of the CMA estimates across treatment episodes/sliding windows (if more than 2) can be visually represented as well in the left side of the figure (NB, their horizontal scales may be different across patients).

When several patients are displayed on the same plot, they are organized vertically, and alternating bands (white and gray) help distinguish consecutive patients. Implicitely, all patients contained in the `cma` object will be plotted, but the `patients.to.plot` parameter allows the selection of a subset of patients.

Finally, the y-axis shows the patient ID and possibly the CMA estimate as well.

Any not explicitely defined arguments are passed to the simple CMA estimation and plotting function; therefore, for more info about possible estimation parameters plese see the help for the appropriate simple CMA, and for possible aesthetic tweaks, please see the help for their plotting.

See Also

See the simple CMA estimation `CMA1` to `CMA9` and plotting `plot.CMA1` functions for extra parameters.

Examples

```
## Not run:
cmaW <- CMA_sliding_window(CMA=CMA1,
                        data=med.events,
                        ID.colname="PATIENT_ID",
                        event.date.colname="DATE",
                        event.duration.colname="DURATION",
                        event.daily.dose.colname="PERDAY",
                        medication.class.colname="CATEGORY",
                        carry.only.for.same.medication=FALSE,
                        consider.dosage.change=FALSE,
                        followup.window.start=0,
                        observation.window.start=0,
                        observation.window.duration=365,
                        sliding.window.start=0,
                        sliding.window.start.unit="days",
                        sliding.window.duration=90,
                        sliding.window.duration.unit="days",
                        sliding.window.step.duration=7,
                        sliding.window.step.unit="days",
                        sliding.window.no.steps=NA,
                        date.format="%m/%d/%Y"
```

```
                        );
plot(cmaW, patients.to.plot=c("1","2"));
cmaE <- CMA_per_episode(CMA=CMA1,
                        data=med.events,
                        ID.colname="PATIENT_ID",
                        event.date.colname="DATE",
                        event.duration.colname="DURATION",
                        event.daily.dose.colname="PERDAY",
                        medication.class.colname="CATEGORY",
                        carry.only.for.same.medication=FALSE,
                        consider.dosage.change=FALSE,
                        followup.window.start=0,
                        observation.window.start=0,
                        observation.window.duration=365,
                        date.format="%m/%d/%Y"
                       );
plot(cmaE, patients.to.plot=c("1","2"));
## End(Not run)
```

`plot_interactive_cma` *Interactive exploration and CMA computation.*

Description

Interactively plot a given patient's data, allowing the real-time exploration of the various CMAs and their parameters. It can use `Rstudio`'s `manipulate` library or `Shiny`.

Usage

```
plot_interactive_cma(...)
```

Arguments

`...`	Parameters to be passed to `plot_interactive_cma()` in package `AdhereRViz`.

Details

This is merely a stub for the actual implementation in package `AdhereRViz`: it just checks if this package is installed and functional, in which case it calls the actual implementation, otherwise warns the user that `AdhereRViz` must be instaled.

Value

Nothing

See Also

Function `plot_interactive_cma` in package `AdhereRViz`.

Examples

```
## Not run:
plot_interactive_cma(med.events,
                     ID.colname="PATIENT_ID",
                     event.date.colname="DATE",
                     event.duration.colname="DURATION",
                     event.daily.dose.colname="PERDAY",
                     medication.class.colname="CATEGORY");
## End(Not run)
```

`print.CMA0` *Print CMA0 (and derived) objects.*

Description

Prints and summarizes a basic CMA0, or derived, object.

Usage

```
## S3 method for class 'CMA0'
print(
  x,
  ...,
  inline = FALSE,
  format = c("text", "latex", "markdown"),
  print.params = TRUE,
  print.data = TRUE,
  exclude.params = c("event.info", "real.obs.windows"),
  skip.header = FALSE,
  cma.type = class(cma)[1]
)

## S3 method for class 'CMA1'
print(...)

## S3 method for class 'CMA2'
print(...)

## S3 method for class 'CMA3'
print(...)

## S3 method for class 'CMA4'
print(...)

## S3 method for class 'CMA5'
print(...)
```

```
## S3 method for class 'CMA6'
print(...)

## S3 method for class 'CMA7'
print(...)

## S3 method for class 'CMA8'
print(...)

## S3 method for class 'CMA9'
print(...)

## S3 method for class 'CMA_per_episode'
print(
  x,
  ...,
  inline = FALSE,
  format = c("text", "latex", "markdown"),
  print.params = TRUE,
  print.data = TRUE,
 exclude.params = c("event.info", "inner.event.info", "mapping.episodes.to.events"),
  skip.header = FALSE,
  cma.type = class(x)[1]
)

## S3 method for class 'CMA_sliding_window'
print(...)
```

Arguments

x	A CMA0 or derived object, representing the CMA to print.
...	other possible parameters
inline	*Logical*, should print inside a line of text or as a separate, extended object?
format	A *string*, the type of output: plain text ("text"; default), LaTeX ("latex") or R Markdown ("markdown").
print.params	*Logical*, should print the parameters?
print.data	*Logical*, should print a summary of the data?
exclude.params	A vector of *strings*, the names of the object fields to exclude from printing (usually, internal information irrelevant to the end-user).
skip.header	*Logical*, should the header be printed?
cma.type	A *string*, used to override the reported object's class.

Details

Can produce output for the console (text), R Markdown or LaTeX, showing various types of information.

Examples

```
cma0 <- CMA0(data=med.events,
             ID.colname="PATIENT_ID",
             event.date.colname="DATE",
             event.duration.colname="DURATION",
             event.daily.dose.colname="PERDAY",
             medication.class.colname="CATEGORY",
             followup.window.start=0,
             followup.window.start.unit="days",
             followup.window.duration=2*365,
             followup.window.duration.unit="days",
             observation.window.start=30,
             observation.window.start.unit="days",
             observation.window.duration=365,
             observation.window.duration.unit="days",
             date.format="%m/%d/%Y",
             summary="Base CMA");
cma0;
print(cma0, format="markdown");
cma1 <- CMA1(data=med.events,
             ID.colname="PATIENT_ID",
             event.date.colname="DATE",
             event.duration.colname="DURATION",
             followup.window.start=30,
             observation.window.start=30,
             observation.window.duration=365,
             date.format="%m/%d/%Y"
            );
cma1;
```

---

prune_event_durations *Prune event durations.*

Description

Flags or removes leftover supply durations after dosage changes, the end of a special period, or treatment interruption. The function accepts the raw list output of `compute_event_durations` and additional arguments to specify event durations that need to be removed.

Usage

```
prune_event_durations(
  data,
  include = c("special periods", "treatment interruptions", "dosage changes"),
  medication.class.colnames = data$medication.class.colnames,
  days.within.out.date.1,
  days.within.out.date.2,
  keep.all = TRUE,
  suppress.warnings = FALSE,
```

```
  return.data.table = FALSE,
  ...
)
```

Arguments

data	A list, the output of compute_event_durations.
include	A Vector of *strings* indicating whether to include dosage changes, special periods, and/or treatment interruptions.
medication.class.colnames	A Vector of *strings*, the name(s) of the column(s) in the event_durations element of data to identify medication classes. Defaults to the columns used in compute_event_durations.
days.within.out.date.1	event durations from before the dosage change, special period, or treatment interruptions are removed if there is a new dispensing event within the number of days specified as *integer* after the dosage change or end of the special period/treatment interruption.
days.within.out.date.2	event durations from before dosage change, special period, or treatment interruption are removed if there is *NO* new dispensing event within the number of days specified as *integer* after the dosage change or end of the special period/treatment interruption.
keep.all	*Logical*, should events be kept and marked for removal? If TRUE, a new column .prune.event will be added to event_durations, if FALSE the events will be removed from the output.
suppress.warnings	*Logical*, if TRUE don't show any warnings.
return.data.table	*Logical*, if TRUE return a data.table object, otherwise a data.frame.
...	other possible parameters.

Details

Dosage changes, special periods, and treatment interruptions may lead to overestimation of implementation, e.g. if patients get a refill after discharge from hospital and don't continue to use their previous supply. Likewise, it may also lead to overestimation of persistence, e.g. when patients discontinue treatments after the end of a special period or treatment interruption.

Value

A data.frame or data.table, the pruned event_durations.

Examples

```
## Not run:
# select medication class of interest and compute event durations
```

```
disp_data <- durcomp.dispensing[ID == 3 & grepl("J01EE01", ATC.CODE)]
presc_data <- durcomp.prescribing[ID == 3 & grepl("J01EE01", ATC.CODE)]

# compute event durations
event_durations_list <- compute_event_durations(disp.data = disp_data,
                                                presc.data = presc_data,
                                            special.periods.data = durcomp.hospitalisation,
                                                ID.colname = "ID",
                                                presc.date.colname = "DATE.PRESC",
                                                disp.date.colname = "DATE.DISP",
                                                date.format = "%Y-%m-%d",
                                               medication.class.colnames = c("ATC.CODE",
                                                                             "UNIT",
                                                                             "FORM"),
                                                total.dose.colname = "TOTAL.DOSE",
                                               presc.daily.dose.colname = "DAILY.DOSE",
                                              presc.duration.colname = "PRESC.DURATION",
                                                visit.colname = "VISIT",
                                                force.init.presc = TRUE,
                                                force.presc.renew = TRUE,
                                                split.on.dosage.change = TRUE,
                                                trt.interruption = "carryover",
                                                special.periods.method = "carryover",
                                                suppress.warnings = FALSE,
                                                return.data.table = TRUE,
                                                progress.bar = FALSE)

# prune event durations
event_durations <- prune_event_durations(event_durations_list,
                                         include = c("special periods"),
                                         medication.class.colnames = "ATC.CODE",
                                         days.within.out.date.1 = 7,
                                         days.within.out.date.2 = 30,
                                         keep.all = FALSE)

## End(Not run)
```

---

subsetCMA *Restrict a CMA object to a subset of patients.*

Description

Restrict a CMA object to a subset of patients.

Usage

```
subsetCMA(cma, patients, suppress.warnings)
```

Arguments

cma	a CMA object.
patients	a list of patient IDs to keep.
suppress.warnings	*Logical*, if TRUE don't show any warnings.

Value

a CMA object containing only the information for the given patients.

Examples

```
cma1 <- CMA1(data=med.events,
             ID.colname="PATIENT_ID",
             event.date.colname="DATE",
             event.duration.colname="DURATION",
             followup.window.start=30,
             observation.window.start=30,
             observation.window.duration=365,
             date.format="%m/%d/%Y"
            );
getCMA(cma1);
cma1a <- subsetCMA(cma1, patients=c(1:3,7));
cma1a; getCMA(cma1a);
```

---

time_to_initiation *Computation of initiation times.*

Description

Computes the time between the start of a prescription episode and the first dispensing event for each medication class.

Usage

```
time_to_initiation(
  presc.data = NULL,
  disp.data = NULL,
  ID.colname = NA,
  medication.class.colnames = NA,
  presc.start.colname = NA,
  disp.date.colname = NA,
  date.format = "%d.%m.%Y",
  suppress.warnings = FALSE,
  return.data.table = FALSE,
  ...
)
```

Arguments

presc.data	A data.frame or data.table containing the prescription episodes. Must contain, at a minimum, the patient unique ID, one medication identifier, and the start date of the prescription episode, and might also contain additional columns to identify and group medications (the actual column names are defined in the medication.class.colnames parameter).
disp.data	A data.frame or data.table containing the dispensing events. Must contain, at a minimum, the patient unique ID, one medication identifier, the dispensing date, and might also contain additional columns to identify and group medications (the actual column names are defined in the medication.class.colnames parameter).
ID.colname	A *string*, the name of the column in presc.data and disp.data containing the unique patient ID, or NA if not defined.
medication.class.colnames	A Vector of *strings*, the name(s) of the column(s) in data containing the classes/types/groups of medication, or NA if not defined.
presc.start.colname	A *string*, the name of the column in presc.data containing the prescription date (in the format given in the date.format parameter), or NA if not defined.
disp.date.colname	A *string*, the name of the column in disp.data containing the dispensing date (in the format given in the date.format parameter), or NA if not defined.
date.format	A *string* giving the format of the dates used in the data and the other parameters; see the format parameters of the as.Date function for details (NB, this concerns only the dates given as strings and not as Date objects).
suppress.warnings	*Logical*, if TRUE don't show any warnings.
return.data.table	*Logical*, if TRUE return a data.table object, otherwise a data.frame.
...	other possible parameters

Details

The period between the start of a prescription episode and the first dose administration may impact health outcomes differently than omitting doses once on treatment or interrupting medication for longer periods of time. Primary non-adherence (not acquiring the first prescription) or delayed initiation may have a negative impact on health outcomes. The function time_to_initiation calculates the time between the start of a prescription episode and the first dispensing event, taking into account multiple variables to differentiate between treatments.

Value

A data.frame or data.table with the following columns:

- ID.colname the unique patient ID, as given by the ID.colname parameter.
- medication.class.colnames the column(s) with classes/types/groups of medication, as given by the medication.class.colnames parameter.

- episode.start the date of the first prescription event.
- first.disp the date of the first dispensing event.
- time.to.initiation the difference in days between the first dispensing date and the first prescription date.

Examples

```
time_init <- time_to_initiation(presc.data = durcomp.prescribing,
                                disp.data = durcomp.dispensing,
                                ID.colname = "ID",
                               medication.class.colnames = c("ATC.CODE", "FORM", "UNIT"),
                                presc.start.colname = "DATE.PRESC",
                                disp.date.colname = "DATE.DISP",
                                date.format = "%Y-%m-%d",
                                suppress.warnings = FALSE,
                                return.data.table = TRUE);
```

Index