Package 'incidence2'

September 2, 2024

Type Package

Title Compute, Handle and Plot Incidence of Dated Events

Version 2.4.0

Description Provides functions and classes to compute, handle and visualise incidence from dated events for a defined time interval. Dates can be provided in various standard formats. The class 'incidence2' is used to store computed incidence and can be easily manipulated, subsetted, and plotted. This package is part of the RECON (<https://www.repidemicsconsortium.org/>) toolkit for outbreak analysis (<https://www.reconverse.org>).

Encoding UTF-8

License MIT + file LICENSE

URL https://www.reconverse.org/incidence2/, https://github.com/reconverse/incidence2

BugReports https://github.com/reconverse/incidence2/issues

Depends grates (>= 1.0.0), R (>= 4.1.0)

Imports grDevices, data.table, pillar, utils, stats, tibble, tidyr, dplyr (>= 1.1.0), tidyselect, rlang, vctrs, ympes (>= 1.3.0)

RoxygenNote 7.3.2

Suggests outbreaks, ggplot2, scales, knitr, markdown, testthat (>= 3.0.0), ciTools, withr

VignetteBuilder knitr

Config/testthat/edition 3

LazyData true

NeedsCompilation no

Author Tim Taylor [aut, cre] (<https://orcid.org/0000-0002-8587-7113>), Thibaut Jombart [ctb]

Maintainer Tim Taylor <tim.taylor@hiddenelephants.co.uk>

Repository CRAN

Date/Publication 2024-09-02 11:50:02 UTC

Contents

`accessors` *Access various elements of an incidence object*

Description

Access various elements of an incidence object

Usage

```
get_date_index_name(x, ...)

## Default S3 method:
get_date_index_name(x, ...)

## S3 method for class 'incidence2'
get_date_index_name(x, ...)

get_dates_name(x, ...)

get_count_variable_name(x, ...)
```

```
## Default S3 method:
get_count_variable_name(x, ...)

## S3 method for class 'incidence2'
get_count_variable_name(x, ...)

get_count_value_name(x, ...)

## Default S3 method:
get_count_value_name(x, ...)

## S3 method for class 'incidence2'
get_count_value_name(x, ...)

get_group_names(x, ...)

## Default S3 method:
get_group_names(x, ...)

## S3 method for class 'incidence2'
get_group_names(x, ...)

get_date_index(x, ...)

## Default S3 method:
get_date_index(x, ...)

## S3 method for class 'incidence2'
get_date_index(x, ...)

get_dates(x, ...)

get_count_variable(x, ...)

## Default S3 method:
get_count_variable(x, ...)

## S3 method for class 'incidence2'
get_count_variable(x, ...)

get_count_value(x, ...)

## Default S3 method:
get_count_value(x, ...)

## S3 method for class 'incidence2'
get_count_value(x, ...)
```

```
get_groups(x, ...)

## Default S3 method:
get_groups(x, ...)

## S3 method for class 'incidence2'
get_groups(x, ...)
```

Arguments

x	An R object.
...	Not currently used.

Value

- `get_date_index_name()`: The name of the date_index variable of x.
- `get_dates_name()`: Alias for `get_date_index_name()`.
- `get_count_variable_name()`: The name of the count variable of x.
- `get_count_value_name()`: The name of the count value of x.
- `get_group_names()`: The name(s) of the group variable(s) of x.
- `get_date_index()`: The date_index variable of x.
- `get_dates()`: Alias for `get_date_index()`.
- `get_count_variable()`: The count variable of x.
- `get_count_value()`: The count value of x.
- `get_groups()`: List of the group variable(s) of x.

Examples

```
if (requireNamespace("outbreaks", quietly = TRUE)) {
    data(ebola_sim_clean, package = "outbreaks")
    dat <- ebola_sim_clean$linelist
    i <- incidence(
        dat,
        date_index = "date_of_onset",
        groups = c("gender", "hospital")
    )
    get_count_variable_name(i)
    get_group_names(i)
    get_dates_name(i)
}
```

as.data.frame.incidence2

Convert incident object to a data frame

Description

Convert incident object to a data frame

Usage

```
## S3 method for class 'incidence2'
as.data.frame(x, row.names, optional, ...)
```

Arguments

x	incidence2 object.
row.names	Not used.
optional	Not used.
...	Not used.

See Also

as.data.frame for the underlying generic.

Examples

```
dat <- data.frame(
    dates = Sys.Date() + 1:100,
    names = rep(c("Jo", "John"), 5)
)

dat <- incidence(dat, date_index = "dates", groups = "names")
as.data.frame(dat)
```

as.data.table.incidence2

Coerce to a data.table

Description

Coerce to a data.table

Usage

```
## S3 method for class 'incidence2'
as.data.table(x, keep.rownames, ...)
```

Arguments

x	An incidence2 object.
keep.rownames	Not used.
...	Passed to other methods.

Value

A data.table of the original input but with no additional attributes.

See Also

data.table::as.data.table for the underlying generic.

Examples

```
if (requireNamespace("outbreaks", quietly = TRUE)) {
    data(ebola_sim_clean, package = "outbreaks")
    dat <- ebola_sim_clean$linelist
    x <- incidence(dat, "date_of_onset")
    as.data.table(x)
}
```

---

as_incidence *Coerce to an incidence object*

Description

Generic for coercion to an <incidence2> object.

Usage

```
as_incidence(x, ...)

## Default S3 method:
as_incidence(x, ...)

## S3 method for class 'incidence2'
as_incidence(x, ...)
```

Arguments

x	An R object.
...	Additional arguments to be passed to or from other methods.

Value

An <incidence2> object.

---

as_tibble.incidence2 *Coerce to a tibble*

Description

Coerce to a tibble

Usage

```
## S3 method for class 'incidence2'
as_tibble(x, ..., .rows, .name_repair, rownames)
```

Arguments

x	An incidence2 object.
...	Unused, for extensibility.
.rows	The number of rows, useful to create a 0-column tibble or just as an additional check.
.name_repair	Treatment of problematic column names: • "minimal": No name repair or checks, beyond basic existence, • "unique": Make sure names are unique and not empty, • "check_unique": (default value), no name repair, but check they are unique, • "universal": Make the names unique and syntactic • a function: apply custom name repair (e.g., .name_repair = make.names for names in the style of base R). • A purrr-style anonymous function, see rlang::as_function() This argument is passed on as repair to vctrs::vec_as_names(). See there for more details on these terms and the strategies used to enforce them.
rownames	How to treat existing row names of a data frame or matrix: • NULL: remove row names. This is the default. • NA: keep row names. • A string: the name of a new column. Existing rownames are transferred into this column and the row.names attribute is deleted. No name repair is applied to the new column name, even if x already contains a column of that name. Use as_tibble(rownames_to_column(...)) to safeguard against this case. Read more in rownames.

Value

A tibble of the original input but with no additional attributes.

Examples

```
if (requireNamespace("outbreaks", quietly = TRUE)) {
    data(ebola_sim_clean, package = "outbreaks")
    dat <- ebola_sim_clean$linelist
    x <- incidence(dat, "date_of_onset")
    as_tibble(x)
}
```

`bootstrap_incidence` *Bootstrap incidence time series*

Description

This function can be used to bootstrap incidence2 objects. Bootstrapping is done by sampling with replacement the original input dates.

Usage

```
bootstrap_incidence(x, randomise_groups = FALSE)
```

Arguments

`x`	An incidence2 object.
`randomise_groups`	`bool`. Should groups be randomised as well in the resampling procedure; respective group sizes will be preserved, but this can be used to remove any group-specific temporal dynamics. If `FALSE` (default), data are resampled within groups.

Details

As original data are not stored in incidence2 objects, the bootstrapping is achieved by multinomial sampling of date bins weighted by their relative incidence.

Value

An incidence2 object.

Author(s)

Thibaut Jombart, Tim Taylor

Examples

```
if (requireNamespace("outbreaks", quietly = TRUE)) {
    data(fluH7N9_china_2013, package = "outbreaks")
    i <- incidence(
        fluH7N9_china_2013,
        date_index = "date_of_onset",
        groups = "gender"
   )
   bootstrap_incidence(i)
}
```

`complete_dates` *Complete dates for all group combinations*

Description

This function ensures that an incidence object has the same range of dates for each grouping. By default missing counts will be filled with `0L`.

Usage

```
complete_dates(x, expand = TRUE, fill = 0L, by = 1L, allow_POSIXct = FALSE)
```

Arguments

`x`	`<incidence2>` object.
`expand`	`logical`. Should a range of dates from the minimum to maximum value of the date index also be created. If `expand` is TRUE (default) then complete_dates will attempt to use `function(x) seq(min(x), max(x), by = 1)` to generate a complete sequence of dates.
`fill`	`numeric`. The value to replace missing counts by. Defaults to `0L`.
`by`	`Defunct`. Ignored.
`allow_POSIXct`	`logical`. Should this function work with POSIXct dates? Defaults to `FALSE`.

Value

An incidence2 object.

Examples

```
x <- data.frame(
    dates = Sys.Date() + c(1,3,4),
    groups = c("grp1","grp2", "grp1"),
    counts = 1:3
)

i <- incidence(x, date_index = "dates", groups = "groups", counts = "counts")
complete_dates(i)
```

`covidregionaldataUK` *Regional data for COVID-19 cases in the UK*

Description

A dataset containing the daily time-series of cases, tests, hospitalisations, and deaths for UK.

Usage

```
covidregionaldataUK
```

Format

A data frame with 6370 rows and 26 variables:

date the date that the counts were reported (YYYY-MM-DD)

region the region name

region_code the region code

cases_new new reported cases for that day

cases_total total reported cases up to and including that day

deaths_new new reported deaths for that day

deaths_total total reported deaths up to and including that day

recovered_new new reported recoveries for that day

recovered_total total reported coveries up to and including that day

hosp_new new reported hospitalisations for that day

hosp_total total reported hospitalisations up to and including that day (note this is cumulative total of new reported, not total currently in hospital).

tested_new tests for that day

tested_total total tests completed up to and including that day

Details

Extracted using the covidregionaldata package on 2021-06-03.

Source

https://CRAN.R-project.org/package=covidregionaldata

`cumulate` *Compute cumulative 'incidence'*

Description

`cumulate()` computes the cumulative incidence over time for an incidence2 object.

Usage

```
cumulate(x)
```

Arguments

`x`	incidence2 object.

Examples

```
dat <- data.frame(
  dates = as.integer(c(0,1,2,2,3,5,7)),
  groups = factor(c(1, 2, 3, 3, 3, 3, 1))
)

i <- incidence(dat, date_index = "dates", groups = "groups")
cumulate(i)
```

`estimate_peak` *Estimate the peak date of an incidence curve*

Description

This function can be used to estimate the peak of an epidemic curve using bootstrapped samples of the available data.

Usage

```
estimate_peak(x, n = 100L, alpha = 0.05, first_only = TRUE, progress = TRUE)
```

Arguments

x	An incidence2 object.
n	integer. The number of bootstrap datasets to be generated; defaults to 100. [double] vectors will be converted via as.integer(n).
alpha	numeric. The type 1 error chosen for the confidence interval; defaults to 0.05.
first_only	bool. Should only the first peak (by date) be kept. Defaults to TRUE.
progress	bool. Should a progress bar be displayed (default = TRUE)

Details

Input dates are resampled with replacement to form bootstrapped datasets; the peak is reported for each, resulting in a distribution of peak times. When there are ties for peak incidence, only the first date is reported.

Note that the bootstrapping approach used for estimating the peak time makes the following assumptions:

- the total number of event is known (no uncertainty on total incidence)
- dates with no events (zero incidence) will never be in bootstrapped datasets
- the reporting is assumed to be constant over time, i.e. every case is equally likely to be reported

Value

A data frame with the the following columns:

- observed_date: the date of peak incidence of the original dataset.
- observed_count: the peak incidence of the original dataset.
- estimated: the median peak time of the bootstrap datasets.
- lower_ci/upper_ci: the confidence interval based on bootstrap datasets.
- bootstrap_peaks: a nested tibble containing the the peak times of the bootstrapped datasets.

Author(s)

Thibaut Jombart and Tim Taylor, with inputs on caveats from Michael Höhle.

See Also

bootstrap_incidence() for the bootstrapping underlying this approach and keep_peaks() to get the peaks in a single incidence2 object.

Examples

```
if (requireNamespace("outbreaks", quietly = TRUE)) {

  # load data and create incidence
  data(fluH7N9_china_2013, package = "outbreaks")
  i <- incidence(fluH7N9_china_2013, date_index = "date_of_onset")

  # find 95% CI for peak time using bootstrap
  estimate_peak(i)
}
```

`incidence` *Compute the incidence of events*

Description

`incidence()` calculates the *incidence* of different events across specified time periods and groupings.

Usage

```
incidence(
  x,
  date_index,
  groups = NULL,
  counts = NULL,
  count_names_to = "count_variable",
  count_values_to = "count",
  date_names_to = "date_index",
  rm_na_dates = TRUE,
  interval = NULL,
  offset = NULL,
  complete_dates = FALSE,
  fill = 0L,
  ...
)
```

Arguments

`x`	A data frame object representing a linelist or pre-aggregated dataset.
`date_index`	`character`. The time index(es) of the given data. This should be the name(s) corresponding to the desired date column(s) in `x`. A named vector can be used for convenient relabelling of the resultant output. Multiple indices only make sense when `x` is a linelist.

groups
: character.

  An optional vector giving the names of the groups of observations for which incidence should be grouped.

  A named vector can be used for convenient relabelling of the resultant output.

counts
: character.

  The count variables of the given data. If NULL (default) the data is taken to be a linelist of individual observations.

  A named vector can be used for convenient relabelling of the resultant output.

count_names_to
: character.

  The column to create which will store the counts column names provided that counts is not NULL.

count_values_to
: character.

  The name of the column to store the resultant count values in.

date_names_to
: character.

  The name of the column to store the date variables in.

rm_na_dates
: bool.

  Should NA dates be removed prior to aggregation?

interval
: An optional scalar integer or string indicating the (fixed) size of the desired time interval you wish to use for for computing the incidence.

  Defaults to NULL in which case the date_index columns are left unchanged.

  Numeric values are coerced to integer and treated as a number of days to group.

  Text strings can be one of:

  ```
  * day or daily
  * week(s) or weekly
  * epiweek(s)
  * isoweek(s)
  * month(s) or monthly
  * yearmonth(s)
  * quarter(s) or quarterly
  * yearquarter(s)
  * year(s) or yearly
  ```

  More details can be found in the "Interval specification" section.

offset
: Only applicable when interval is not NULL.

  An optional scalar integer or date indicating the value you wish to start counting periods from relative to the Unix Epoch:

  - Default value of NULL corresponds to 0L.
  - For other integer values this is stored scaled by n (`offset <- as.integer(offset) %% n`).
  - For date values this is first converted to an integer offset (`offset <- floor(as.numeric(offset))`) and then scaled via n as above.

`complete_dates` `bool`.
Should the resulting object have the same range of dates for each grouping.
Missing counts will be filled with `0L` unless the `fill` argument is provided (and this value will take precedence).
Will attempt to use `function(x) seq(min(x), max(x), by = 1)` on the resultant date_index column to generate a complete sequence of dates.
More flexible completion is possible by using the `complete_dates()` function.

`fill` `numeric`.
Only applicable when `complete_dates = TRUE`.
The value to replace missing counts caused by completing dates.
If unset then will default to `0L`.

`...` Not currently used.

Details

`incidence2` objects are a sub class of data frame with some additional invariants. That is, an `incidence2` object must:

- have one column representing the date index (this does not need to be a `date` object but must have an inherent ordering over time);
- have one column representing the count variable (i.e. what is being counted) and one variable representing the associated count;
- have zero or more columns representing groups;
- not have duplicated rows with regards to the date and group variables.

Value

A `tibble` with subclass `incidence2`.

Interval specification

Where `interval` is specified, `incidence()`, predominantly uses the `grates` package to generate appropriate date groupings. The grouping used depends on the value of `interval`. This can be specified as either an integer value or a string corresponding to one of the classes:

- integer values: `<grates_period>` object, grouped by the specified number of days.
- day, daily: `<Date>` objects.
- week(s), weekly, isoweek: `<grates_isoweek>` objects.
- epiweek(s): `<grates_epiweek>` objects.
- month(s), monthly, yearmonth: `<grates_yearmonth>` objects.
- quarter(s), quarterly, yearquarter: `<grates_yearquarter>` objects.
- year(s) and yearly: `<grates_year>` objects.

For "day" or "daily" interval, we provide a thin wrapper around `as.Date()` that ensures the underlying data are whole numbers and that time zones are respected. Note that additional arguments are not forwarded to `as.Date()` so for greater flexibility users are advised to modifying your input prior to calling `incidence()`.

See Also

- `browseVignettes("grates")` for more details on the grate object classes.
- `incidence_()` for a version supporting tidy-select semantics in some arguments.

Examples

```
if (requireNamespace("outbreaks", quietly = TRUE)) {
    data(ebola_sim_clean, package = "outbreaks")
    dat <- ebola_sim_clean$linelist
    incidence(dat, "date_of_onset")
    incidence(dat, "date_of_onset", groups = c("gender", "hospital"))
}
```

---

incidence_ *Compute the incidence of events (tidyselect compatible)*

Description

`incidence_()` calculates the *incidence* of different events across specified time periods and groupings. It differs from `incidence()` only in its support for tidy-select semantics in some of its arguments.

Usage

```
incidence_(
  x,
  date_index,
  groups = NULL,
  counts = NULL,
  count_names_to = "count_variable",
  count_values_to = "count",
  date_names_to = "date_index",
  rm_na_dates = TRUE,
  interval = NULL,
  offset = NULL,
  complete_dates = FALSE,
  ...
)
```

Arguments

x	A data frame object representing a linelist or pre-aggregated dataset.

`date_index` tidyselect.

The time index(es) of the given data.

This should be the name(s) corresponding to the desired date column(s) in x.

A named vector can be used for convenient relabelling of the resultant output.

Multiple indices only make sense when x is a linelist.

`groups` tidyselect.

An optional vector giving the names of the groups of observations for which incidence should be grouped.

A named vector can be used for convenient relabelling of the resultant output.

`counts` tidyselect.

The count variables of the given data. If NULL (default) the data is taken to be a linelist of individual observations.

A named vector can be used for convenient relabelling of the resultant output.

`count_names_to` `character`.

The column to create which will store the `counts` column names provided that `counts` is not NULL.

`count_values_to`

`character`.

The name of the column to store the resultant count values in.

`date_names_to` `character`.

The name of the column to store the date variables in.

`rm_na_dates` `bool`.

Should `NA` dates be removed prior to aggregation?

`interval` An optional scalar integer or string indicating the (fixed) size of the desired time interval you wish to use for for computing the incidence.

Defaults to NULL in which case the date_index columns are left unchanged.

Numeric values are coerced to integer and treated as a number of days to group.

Text strings can be one of:

```
* day or daily
* week(s) or weekly
* epiweek(s)
* isoweek(s)
* month(s) or monthly
* yearmonth(s)
* quarter(s) or quarterly
* yearquarter(s)
* year(s) or yearly
```

More details can be found in the "Interval specification" section.

`offset` Only applicable when `interval` is not NULL.

An optional scalar integer or date indicating the value you wish to start counting periods from relative to the Unix Epoch:

- Default value of NULL corresponds to 0L.

- For other integer values this is stored scaled by `n` (`offset <- as.integer(offset) %% n`).
- For date values this is first converted to an integer offset (`offset <- floor(as.numeric(offset))`) and then scaled via `n` as above.

`complete_dates` `bool`.

Should the resulting object have the same range of dates for each grouping.

Missing counts will be filled with `0L`.

Will attempt to use `function(x) seq(min(x), max(x), by = 1)` on the resultant date_index column to generate a complete sequence of dates.

More flexible completion is possible by using the `complete_dates()`

`...` Not currently used.

Details

`<incidence2>` objects are a sub class of data frame with some additional invariants. That is, an `<incidence2>` object must:

- have one column representing the date index (this does not need to be a `date` object but must have an inherent ordering over time);
- have one column representing the count variable (i.e. what is being counted) and one variable representing the associated count;
- have zero or more columns representing groups;
- not have duplicated rows with regards to the date and group variables.

Value

A `tibble` with subclass `incidence2`.

Interval specification

Where `interval` is specified, `incidence_()`, predominantly uses the `grates` package to generate appropriate date groupings. The grouping used depends on the value of `interval`. This can be specified as either an integer value or a string corresponding to one of the classes:

- integer values: `<grates_period>` object, grouped by the specified number of days.
- day, daily: `<Date>` objects.
- week(s), weekly, isoweek: `<grates_isoweek>` objects.
- epiweek(s): `<grates_epiweek>` objects.
- month(s), monthly, yearmonth: `<grates_yearmonth>` objects.
- quarter(s), quarterly, yearquarter: `<grates_yearquarter>` objects.
- year(s) and yearly: `<grates_year>` objects.

For "day" or "daily" interval, we provide a thin wrapper around `as.Date()` that ensures the underlying data are whole numbers and that time zones are respected. Note that additional arguments are not forwarded to `as.Date()` so for greater flexibility users are advised to modifying your input prior to calling `incidence_()`.

See Also

- `browseVignettes("grates")` for more details on the grate object classes.
- `incidence()` for a the underlying function without support for tidyselect semantics. This may be preferable for programatic usage.

Examples

```
if (requireNamespace("outbreaks", quietly = TRUE)) {
    data(ebola_sim_clean, package = "outbreaks")
    dat <- ebola_sim_clean$linelist
    incidence_(dat, date_of_onset)
    incidence_(dat, date_of_onset, groups = c(gender, hospital))
}
```

---

keep *Keep first, last and peak occurences*

Description

`keep_first()` and `keep_last()` keep the first and last `n` rows to occur for each grouping when in ascending date order. `keep_peaks()` keeps the rows with the maximum count value for each group. `first_peak()` is a convenience wrapper around `keep_peaks()` with the `first_only` argument set to `TRUE`.

Usage

```
keep_first(x, n, complete_dates = TRUE, ...)

keep_last(x, n, complete_dates = TRUE, ...)

keep_peaks(x, complete_dates = TRUE, first_only = FALSE, ...)

first_peak(x, complete_dates = TRUE, ...)
```

Arguments

`x`	incidence2 object.
`n`	`integer`. Number of entries to keep. `double` vectors will be converted via `as.integer(n)`.
`complete_dates`	`bool`. Should `complete_dates()` be called on the data prior to keeping the first entries. Defaults to TRUE.

...	Other arguments passed to `complete_dates()`.
first_only	`bool`. Should only the first peak (by date) be kept. Defaults to `TRUE`.

Value

incidence2 object with the chosen entries.

Examples

```
if (requireNamespace("outbreaks", quietly = TRUE)) {
    data(ebola_sim_clean, package = "outbreaks")
    dat <- ebola_sim_clean$linelist
    inci <- incidence(dat, "date_of_onset")
    keep_first(inci, 3)
    keep_last(inci, 3)
}
```

---

mutate.incidence2 *Create, modify, and delete incidence2 columns*

Description

Method for dplyr::mutate that implicitly accounts for the inherent grouping structure of incidence2 objects.

Usage

```
## S3 method for class 'incidence2'
mutate(
  .data,
  ...,
  .by,
  .keep = c("all", "used", "unused", "none"),
  .before = NULL,
  .after = NULL
)
```

Arguments

.data	An incidence2 object.
...	<data-masking> Name-value pairs. The name gives the name of the column in the output. The value can be:

- A vector of length 1, which will be recycled to the correct length.
- A vector the same length as the current group (or the whole data frame if ungrouped).
- `NULL`, to remove the column.
- A data frame or tibble, to create multiple columns in the output.

`.by` Not used as grouping structure implicit.

`.keep` Control which columns from `.data` are retained in the output. Grouping columns and columns created by `...` are always kept.

- `"all"` retains all columns from `.data`. This is the default.
- `"used"` retains only the columns used in `...` to create new columns. This is useful for checking your work, as it displays inputs and outputs side-by-side.
- `"unused"` retains only the columns *not* used in `...` to create new columns. This is useful if you generate new columns, but no longer need the columns used to generate them.
- `"none"` doesn't retain any extra columns from `.data`. Only the grouping variables and columns created by `...` are kept.

`.before, .after` <`tidy-select`> Optionally, control where new columns should appear (the default is to add to the right hand side). See `relocate()` for more details.

Value

A modified incidence2 object if the necessary invariants are preserved, otherwise a tibble.

See Also

dplyr::mutate for the underlying generic.

Examples

```
if (requireNamespace("outbreaks", quietly = TRUE) && requireNamespace("ggplot2", quietly = TRUE)) {
    data(ebola_sim_clean, package = "outbreaks")
    ebola_sim_clean$linelist |>
        subset(!is.na(hospital)) |>
        incidence_(date_of_onset, hospital, interval = "isoweek") |>
        mutate(ave = data.table::frollmean(count, n = 3L, align = "right")) |>
        plot(border_colour = "white", angle = 45) +
        ggplot2::geom_line(ggplot2::aes(x = date_index, y = ave))
}
```

`nest.incidence2` *Nest rows into a list-column of data frames*

Description

Method for tidyr::nest that implicitly accounts for the inherent grouping structure of incidence2 objects.

Usage

```
## S3 method for class 'incidence2'
nest(.data, ..., .by, .key, .names_sep)
```

Arguments

`.data`	An incidence2 object.
`...`	Not used.
`.by`	Not used.
`.key`	The name of the resulting nested column. Only applicable when `...` isn't specified, i.e. in the case of `df %>% nest(.by = x)`. If `NULL`, then `"data"` will be used by default.
`.names_sep`	Not used.

Value

A nested tibble with rows corresponding to the count variable and (optionally) group columns of the input object.

See Also

tidyr::nest for the underlying generic.

Examples

```
if (requireNamespace("outbreaks", quietly = TRUE)) {
    data(ebola_sim_clean, package = "outbreaks")
    ebola_sim_clean$linelist |>
        subset(!is.na(hospital)) |>
        incidence_(date_of_onset, hospital, interval = "isoweek") |>
        nest()
}
```

plot.incidence2 *Plot an incidence object*

Description

plot() can be used to provide a bar plot of an incidence object. Due to the complexities with automating plotting it is some what experimental in nature and it may be better to use ggplot2 directly.

Usage

```
## S3 method for class 'incidence2'
plot(
  x,
  y,
  width = 1,
  colour_palette = vibrant,
  border_colour = NA,
  na_colour = "grey",
  alpha = 0.7,
  fill = NULL,
  legend = c("right", "left", "bottom", "top", "none"),
  title = NULL,
  angle = 0,
  size = NULL,
  nrow = NULL,
  n_breaks = 6L,
  show_cases = FALSE,
  ...
)
```

Arguments

x	incidence2 object.
y	Not used. Required for compatibility with the plot() generic.
width	numeric. Value between 0 and 1 indicating the relative size of the bars to the interval. Default 1.
colour_palette	function. The color palette to be used for the different count variables. Defaults to vibrant (see ?palettes).
border_colour	character. The color to be used for the borders of the bars. Use NA (default) for invisible borders.

na_colour	character. The colour to plot NA values in graphs. Defaults to grey.
alpha	numeric. The alpha level for color transparency, with 1 being fully opaque and 0 fully transparent Defaults to 0.7.
fill	character. Which variable to colour plots by. Must be a group or count variable and will mean that variable is not used for facetting. If NULL no distinction if made for plot colours.
legend	character. Position of legend in plot. Only applied if fill is not NULL. One of "right" (default), "left", "bottom", "top" or "none".
title	character. Optional title for the graph.
angle	numeric. Rotation angle for text.
size	numeric. text size in pts.
nrow	integer. Number of rows used for facetting if there are group variables present and just one count in the incidence object. Numeric values are coerced to integer via as.integer().
n_breaks	integer. Approximate number of breaks calculated using scales::breaks_pretty(). Numeric values are coerced to integer via as.integer(). Default 6L.
show_cases	logical. if TRUE, then each observation will be shown individually in a square format. Normally only used for outbreaks with a small number of cases. Defaults to FALSE.
...	Not currently used.

Details

- Faceting will occur automatically if either grouping variables or multiple counts are present.
- If there are multiple count variables, each count will occupy a different row of the resulting plot.
- Utilises ggplot2 so this must be installed to use.

Value

- A `ggplot2::ggplot()` object.

Examples

```
if (requireNamespace("outbreaks", quietly = TRUE) && requireNamespace("ggplot2", quietly = TRUE)) {
    data(ebola_sim_clean, package = "outbreaks")
    dat <- ebola_sim_clean$linelist

    inci <- incidence(dat, date_index = "date_of_onset", groups = "hospital")
    plot(inci, angle = 45)

    inci2 <- regroup(inci)
    plot(inci2)
}
```

---

regroup *Regroup 'incidence' objects*

Description

This function regroups an incidence2 object across the specified groups. The resulting incidence2 object will contains counts aggregated over the specified groups.

Usage

```
regroup(x, groups = NULL)
```

Arguments

`x`	`<incidence2>` object.
`groups`	`character`. The groups to sum over. If `NULL` (default) then the function returns the corresponding object with no groupings.

See Also

`regroup_()` for a version supporting

Examples

```
if (requireNamespace("outbreaks", quietly = TRUE)) {
    data(ebola_sim_clean, package = "outbreaks")
    dat <- ebola_sim_clean$linelist
    i <- incidence(
        dat,
        date_index = "date_of_onset",
        groups = c("gender", "hospital")
    )
    regroup(i)
    regroup(i, "hospital")
}
```

---

regroup_ *Regroup 'incidence' objects (tidyselect compatible)*

Description

This function regroups an <incidence2> object across the specified groups. The resulting <incidence2> object will contains counts summed over the groups present in the input. It differs from regroup() only in support for <tidy-select> semantics in the groups argument.

Usage

```
regroup_(x, groups = NULL)
```

Arguments

x	<incidence2> object.
groups	<tidyselect> The groups to sum over. If NULL (default) then the function returns the corresponding object with no groupings.

See Also

regroup() for a version without tidyselect semantics. This may be preferable for programatic usage.

Examples

```
if (requireNamespace("outbreaks", quietly = TRUE)) {
    data(ebola_sim_clean, package = "outbreaks")
    dat <- ebola_sim_clean$linelist
    i <- incidence_(
        dat,
```

```
        date_index = date_of_onset,
        groups = c(gender, hospital)
    )
    regroup_(i)
    regroup_(i, hospital)
}
```

`split.incidence2` *Divide an incidence2 object in to it's implicit groupings*

Description

Split divides and incidence2 object in to it's underlying groupings (count variable and optionally groups).

Usage

```
## S3 method for class 'incidence2'
split(x, f, drop, ...)
```

Arguments

`x`	An incidence2 object.
`f`	Not used. Present only for generic compatibility.
`drop`	Not used. Present only for generic compatibility.
`...`	Not used. Present only for generic compatibility.

Value

A list of tibbles contained the split data. This list also has a "key" attribute which is a tibble with rows corresponding to the grouping of each split.

See Also

`vctrs::vec_split()` on which `split.incidence2()` is built.

Examples

```
if (requireNamespace("outbreaks", quietly = TRUE)) {
    data(ebola_sim_clean, package = "outbreaks")
    ebola_sim_clean$linelist |>
        subset(!is.na(hospital)) |>
        incidence_(date_of_onset, hospital, interval = "isoweek") |>
        split()
}
```

`summarise.incidence2` *Summarise each grouping down to one row*

Description

Method for dplyr::summarise that implicitly accounts for the inherent grouping structure of incidence2 objects.

Usage

```
## S3 method for class 'incidence2'
summarise(.data, ..., .by, .groups)
```

Arguments

`.data`	An incidence2 object.
`...`	<data-masking> Name-value pairs. The name gives the name of the column in the output. The value can be: • A vector of length 1, which will be recycled to the correct length. • A vector the same length as the current group (or the whole data frame if ungrouped). • `NULL`, to remove the column. • A data frame or tibble, to create multiple columns in the output.
`.by`	Not used as grouping structure implicit.
`.groups`	Not used.

Value

A tibble.

See Also

dplyr::summarise for the underlying grouping.

Examples

```
if (requireNamespace("outbreaks", quietly = TRUE)) {
    data(ebola_sim_clean, package = "outbreaks")
    ebola_sim_clean$linelist |>
        subset(!is.na(hospital)) |>
        incidence_(date_of_onset, hospital, interval = "isoweek") |>
        summarise(model = list(glm(count ~ date_index, family = "poisson")))
}
```

summary.incidence2 *Summary of an incidence object*

Description

Summary of an incidence object

Usage

```
## S3 method for class 'incidence2'
summary(object, ...)
```

Arguments

object	An incidence2 object.
...	Not used.

Value

object (invisibly).

Examples

```
data(ebola_sim_clean, package = "outbreaks")
dat <- ebola_sim_clean$linelist
inci <- incidence(dat, "date_of_onset", groups = c("gender", "hospital"))
summary(inci)
```

vibrant *Color palettes used in incidence*

Description

These functions are color palettes used in incidence. The palettes come from https://personal.sron.nl/~pault/#sec:qualitative and exclude grey, which is reserved for missing data.

Usage

```
vibrant(n)

muted(n)
```

Arguments

n	integer. Number of colours. double vectors will be converted via as.integer(n).

Examples

```
vibrant(5)
muted(10)
```

Index