Package: epiflows (via r-universe)

September 14, 2024

Title Predicting Disease Spread from Flow Data

Version 0.2.1

Description Provides functions and classes designed to handle and visualise epidemiological flows between locations. Also contains a statistical method for predicting disease spread from flow data initially described in Dorigatti et al. (2017) <doi:10.2807/1560-7917.ES.2017.22.28.30572>. This package is part of the RECON (<https://www.repidemicsconsortium.org/>) toolkit for outbreak analysis.

License MIT + file LICENSE

Encoding UTF-8

LazyData true

Depends R (>= 3.4.0)

Imports epicontacts, leaflet, ggmap, geosphere, ggplot2, tibble, sp, stats, htmltools, visNetwork

Suggests testthat, roxygen2, knitr, outbreaks, vdiffr, curl, rmarkdown

RoxygenNote 7.2.3

URL https://www.repidemicsconsortium.org/epiflows/, https://github.com/reconhub/epiflows

BugReports https://github.com/reconhub/epiflows/issues

VignetteBuilder knitr

Repository https://reconhub.r-universe.dev

RemoteUrl https://github.com/reconhub/epiflows

RemoteRef HEAD

RemoteSha 2fdb4d22d82b4471872e2a805d3b3126eceed672

Contents

`add_coordinates` *Add/Retrieve location coordinates*

Description

Adds/Retrieves longitude/latitude values to location data within an epiflows object. Coordinates are added to object's locations slot as `lon` and `lat` columns.

Usage

```
add_coordinates(
  x,
  coordinates = c("lon", "lat"),
  loc_column = "id",
  overwrite = FALSE
)

get_coordinates(x, ...)

## S3 method for class 'epiflows'
get_coordinates(x, location = NULL, ...)
```

Arguments

x	An epiflows object.
coordinates	Either names of the appended columns with longitudes and latitudes, respectively (default: "lon" and "lat") or a data frame with longitude and latitude columns.
loc_column	Name of the column where location names are stored (default: "country").
overwrite	If TRUE, retrieves all geocodes, even those already retrieved. If FALSE (default), overwrites only NAs.
...	unused
location	a character specifying a single location to return as a vector of coordinates. You cannot specify multiple locations with this parameter. Defaults to 'NULL', indicating all locations.

Value

An updated epiflows object.

Author(s)

Pawel Piatkowski, Salla Toikkanen, Zhian Kamvar

See Also

map_epiflows(); plot.epiflows(); get_locations(); get_vars(); global_vars()

Examples

```
# Setting up the data
data("Brazil_epiflows")
data("YF_coordinates")
get_coordinates(Brazil_epiflows) # no coordinates yet
ef <- add_coordinates(Brazil_epiflows, YF_coordinates[-1])
get_coordinates(ef)
get_coordinates(ef, location = "Espirito Santo") # coordinates for Espirito Santo
if (interactive()) {
  # You can use google maps' geocode functionality if you have a decent
  # internet connection
  # NOTE: you may need to authenticate with Google Maps API beforehand
  # using ggmap::register_google()
  ef2 <- add_coordinates(Brazil_epiflows, loc_column = "id")
  ef2
}
```

as.SpatialLinesDataFrame *Convert to SpatialLinesDataFrame class*

Description

Convert to SpatialLinesDataFrame class

Usage

```
as.SpatialLinesDataFrame(x)

## S3 method for class 'epiflows'
as.SpatialLinesDataFrame(x)
```

Arguments

x	an epiflows object

Value

an object of class [sp::SpatialLinesDataFrame]

Examples

```
data("Brazil_epiflows")
data("YF_coordinates")
ef <- add_coordinates(Brazil_epiflows, YF_coordinates[-1])
ef2 <- epicontacts::thin(ef[j = c("Espirito Santo", "Italy"), contacts = "both"])
as.SpatialLinesDataFrame(ef2)
```

epiflows *epiflows*

Description

`epiflows` is a package for predicting and visualising spread of infectious diseases based on flows between geographical locations, e.g., countries. `epiflows` provides functions for calculating spread estimates, handling flow data, and visualization.

Details

The main functions of the package are `make_epiflows()` for constructing an epiflows object and `estimate_risk_spread()` to estimate potential spread. Visualization is available for maps, networks, and grids.

For details, see the vignettes:

```
vignette("introduction", package = "epiflows")
vignette("epiflows-class", package = "epiflows")
```

`estimate_risk_spread` *Travel-related disease cases spreaded to other locations from an infectious location*

Description

Calculates the mean and 95% confidence interval of the estimated number of disease cases that could potentially seed a disease outbreak in the locations they are travelling to, comprising exportations (infected residents of the infectious location travelling abroad during the incubation or infectious period), and importations (international tourists infected by the disease during their stay in the infectious location and returning to their home location). The mean and 95% confidence intervals are obtained by numerically sampling `n_sim` times from the incubation and infectious period distributions. If parameter `return_all_simulations` is set to `TRUE`, the function returns all simulations for each location.

Usage

```
estimate_risk_spread(...)

## Default S3 method:
estimate_risk_spread(
  location_code = character(0),
  location_population = numeric(0),
  num_cases_time_window = numeric(0),
  first_date_cases = character(0),
  last_date_cases = character(0),
  num_travellers_to_other_locations = numeric(0),
  num_travellers_from_other_locations = numeric(0),
  avg_length_stay_days = numeric(0),
  r_incubation = function(n) {
 },
  r_infectious = function(n) {
 },
  n_sim = 1000,
  return_all_simulations = FALSE,
  ...
)
```

```
## S3 method for class 'epiflows'
estimate_risk_spread(
  x,
  location_code = character(0),
  r_incubation = function(n) {
 },
  r_infectious = function(n) {
 },
  n_sim = 1000,
  return_all_simulations = FALSE,
  ...
)
```

Arguments

`...`	Arguments passed onto the default method.
`location_code`	a character string denoting the infectious location code
`location_population`	population of the infectious location
`num_cases_time_window`	cumulative number of cases in infectious location in time window
`first_date_cases`	string with the date of the first disease case in infectious location ("YYYY-MM-DD")
`last_date_cases`	string with the date of the last disease case in infectious location ("YYYY-MM-DD")
`num_travellers_to_other_locations`	number of travellers from the infectious location visiting other locations (T_D)
`num_travellers_from_other_locations`	number of travellers from other locations visiting the infectious location (T_O)
`avg_length_stay_days`	average length of stay in days of travellers from other locations visiting the infectious location. This can be a common number for all locations or a vector with different numbers for each location
`r_incubation`	a function with a single argument `n` generating 'n' random incubation periods
`r_infectious`	a function with a single argument `n` generating 'n' random durations of infectious period
`n_sim`	number of simulations from the incubation and infectious distributions
`return_all_simulations`	logical value indicating whether the returned object is a data frame with all simulations (`return_all_simulations = TRUE`) or a data frame with the mean and lower and upper limits of a 95% confidence interval of the number of cases spread to each location (`return_all_simulations = FALSE`)
`x`	an epiflows object

Value

if `return_all_simulations` is `TRUE`, data frame with all simulations. If `return_all_simulations` is `FALSE`, data frame with the mean and lower and upper limits of a 95% confidence interval of the number of cases spread to each location

Author(s)

Paula Moraga, Zhian Kamvar (epiflows class implementation)

References

Dorigatti I, Hamlet A, Aguas R, Cattarino L, Cori A, Donnelly CA, Garske T, Imai N, Ferguson NM. International risk of yellow fever spread from the ongoing outbreak in Brazil, December 2016 to May 2017. Euro Surveill. 2017;22(28):pii=30572. DOI: doi:10.2807/15607917.ES.2017.22.28.30572

See Also

Construction of epiflows object: `make_epiflows()`
Default variables used in the epiflows implementation: `global_vars()`
Access metadata from the epiflows object: `get_vars()`

Examples

```
## Using an epiflows object --------------------------------

data("YF_flows")
data("YF_locations")
ef <- make_epiflows(flows         = YF_flows,
                    locations     = YF_locations,
                    pop_size      = "location_population",
                    duration_stay = "length_of_stay",
                    num_cases     = "num_cases_time_window",
                    first_date    = "first_date_cases",
                    last_date     = "last_date_cases"
)
## functions generating incubation and infectious periods
incubation <- function(n) {
  rlnorm(n, 1.46, 0.35)
}

infectious <- function(n) {
  rnorm(n, 4.5, 1.5/1.96)
}

res <- estimate_risk_spread(ef,
                            location_code          = "Espirito Santo",
                            r_incubation           = incubation,
                            r_infectious           = infectious,
                            n_sim                  = 1e5,
                            return_all_simulations = TRUE)
boxplot(res, las = 3)
```

```
## Using other data ------------------------------------------------
data(YF_Brazil)
indstate <- 1 # "Espirito Santo" (indstate = 1),
              # "Minas Gerais" (indstate = 2),
              # "Southeast Brazil" (indstate = 5)

res <- estimate_risk_spread(
  location_code = YF_Brazil$states$location_code[indstate],
  location_population = YF_Brazil$states$location_population[indstate],
  num_cases_time_window = YF_Brazil$states$num_cases_time_window[indstate],
  first_date_cases = YF_Brazil$states$first_date_cases[indstate],
  last_date_cases = YF_Brazil$states$last_date_cases[indstate],
  num_travellers_to_other_locations = YF_Brazil$T_D[indstate,],
  num_travellers_from_other_locations = YF_Brazil$T_O[indstate,],
  avg_length_stay_days = YF_Brazil$length_of_stay,
  r_incubation = incubation,
  r_infectious = infectious,
  n_sim = 100000,
  return_all_simulations = FALSE
)
head(res)
```

`get_flows` *Access flow data*

Description

This accessor extract flow data from an `epiflows` object. `get_flows` is a generic with a method defined for `epiflows` objects.

Usage

```
get_flows(x, ...)

## S3 method for class 'epiflows'
get_flows(x, from = NULL, to = NULL, all = FALSE, ...)
```

Arguments

`x`	An ‘epiflows‘ object.
`...`	unused
`from`	a character string defining which regions should be included in the flows
`to`	a character string defining which regions should be included in the flows
`all`	when ‘TRUE‘, all the columns of the flows data frame will be returned. Defaults to ‘FALSE‘, which returns "from", "to", and "n".

Value

A `data.frame` with 3 columns:

- `from`: origin of the flow
- `to`: destination of the flow
- `n`: magnitude of the flow—can be a number of passengers per unit of time, a rate, a probability of migration

Author(s)

Zhian N. Kamvar

See Also

`get_n()`; For location metadata: `get_locations()`; `get_vars()`, `get_pop_size()`, `get_coordinates()`

Examples

```
data("Brazil_epiflows")
head(get_flows(Brazil_epiflows))
get_flows(Brazil_epiflows, from = "Minas Gerais")
get_flows(Brazil_epiflows, to = "Minas Gerais")
get_flows(Brazil_epiflows, from = "Italy", to = "Minas Gerais")
```

`get_id` *Access population identifiers in epiflows objects*

Description

this will return the unique population ID for your epiflows object.

Usage

```
get_id(x, ...)
```

Arguments

`x`	an epiflows object
`...`	arguments passed on to `epicontacts::get_id()`

Value

a character vector of population IDs

Author(s)

Zhian N. Kamvar

See Also

epicontacts::get_id(); get_vars(); get_pop_size(); global_vars()

Examples

```
data("Brazil_epiflows")
get_id(Brazil_epiflows)
```

`get_locations` *Access flow data*

Description

This accessor extract location data from an `epiflows` object. `get_locations` is a generic with a method defined for `epiflows` objects.

Usage

```
get_locations(x, ...)

## S3 method for class 'epiflows'
get_locations(x, ...)
```

Arguments

`x`	An `epiflows` object.
`...`	unused

Value

A `data.frame` with at least 1 column called `id`, specifying the id of the location used in the `flows` data frame.

Author(s)

Thibaut Jombart, Zhian Kamvar

See Also

get_flows(); get_id(); get_pop_size(); get_vars(); get_coordinates(); global_vars()

Examples

```
data("Brazil_epiflows")
get_locations(Brazil_epiflows)
```

get_n *get the number of cases per flow*

Description

This convenience function will return a named vector containing the number of cases flowing to or from a given region.

Usage

```
get_n(x, from = NULL, to = NULL, ...)

## S3 method for class 'epiflows'
get_n(x, from = NULL, to = NULL, ...)
```

Arguments

x	an epiflows object
from	a character vector of length one specifying the location from which the flows originate
to	a character vector of length one specifying the location to which the flows terminate
...	unused

Details

There are three possible outputs of this function:

- **no options specified**: an un-named vector, equivalent to `get_flows(x)$n`
- **from = X**: a named vector of cases flowing *from* **X**
- **to = X**: a named vector of cases flowing *to* **X**

Value

a character vector

See Also

`get_flows()`; For location metadata: `get_vars()`, `get_pop_size()`, `get_coordinates()`

Examples

```
data(Brazil_epiflows)
get_n(Brazil_epiflows, from = "Espirito Santo")
get_n(Brazil_epiflows, to   = "Espirito Santo")
```

get_pop_size *Get population size for each entry in locations*

Description

Get population size for each entry in locations

Usage

```
get_pop_size(x)

## S3 method for class 'epiflows'
get_pop_size(x)
```

Arguments

x	an epiflows object

Value

a named vector of population sizes

See Also

get_vars(), global_vars()

Examples

```
data("Brazil_epiflows")
get_pop_size(Brazil_epiflows)
```

get_vars *Access location metadata*

Description

This accessor extracts variables from the locations data frame in an epiflow object. get_vars is a generic with a method defined for epiflows objects.

Usage

```
get_vars(x, ...)

## S3 method for class 'epiflows'
get_vars(x, what = NULL, id = TRUE, vector = FALSE, ...)

set_vars(x, ...)

set_vars(x, name) <- value

## S3 method for class 'epiflows'
set_vars(x, ...)

## S3 replacement method for class 'epiflows'
set_vars(x, name) <- value
```

Arguments

x	An epiflows object.
...	For set_vars(), any number of variables defined in global_vars() that can be used for mapping or modelling. This is unused in get_vars()
what	a valid character string specifying the variable desired. If 'NULL' (default), the names of the available vars will be returned.
id	a logical. If 'TRUE' (default), the 'id' column of the locations will be the first column of the data frame. if 'FALSE', the variable will be returned with identifiers as row names.
vector	if 'TRUE' the result will be coerced into a vector (or a matrix in the case of coordinates)
name	the name of the variable in global_vars() to assign
value	the name of the column in the locations data

Value

A data frame with the variables requested

Author(s)

Thibaut Jombart, Zhian Kamvar

See Also

global_vars(); make_epiflows(); get_pop_size(); get_id()

Examples

```
data("Brazil_epiflows")
get_vars(Brazil_epiflows) # defined global variables pointint to column names
get_vars(Brazil_epiflows, "duration_stay")
get_vars(Brazil_epiflows, "duration_stay", vector = TRUE)
```

`global_vars` *Epiflow Global Variables*

Description

The metadata in **locations** such as population size, duration of stay in a given location, date of first and last cases, etc. can be useful in estimating the risk of spread, but not everyone will code their data with identical column names. To facilitate their use in the function `estimate_risk_spread()`, the epiflows object stores a dictionary of variables in a place called `$vars`. We can tell epiflows what variables are important when we create the object.

Usage

```
global_vars(..., set = FALSE, reset = FALSE)
```

Arguments

`...`	quoted varaibles to add to the default variables
`set`	when `TRUE`, the variables provided in `...` will be added to the global variables. Defaults to `FALSE`
`reset`	when `TRUE`, the global variables are reset to the default variables listed above. Defaults to `FALSE`

Details

The default varaibles are:

- `coordinates`: two columns specifying the lon and lat coordinates
- `pop_size`: population size of each location
- `duration_stay`: the average duration of stay for each location
- `first_date`: the date of first recorded case
- `last_date`: the date of the last recorded case
- `num_cases`: the number of cases between the first and last date

See Also

`make_epiflows()`, `get_locations()`, `get_vars()`, `set_vars()`, `get_coordinates()`

Examples

```
# see the default varaibles
global_vars()

# Equivalent
getOption("epiflows.vars")

# create an object, specifying these variables
data("YF_locations")
data("YF_flows")
ef <- make_epiflows(flows         = YF_flows,
                    locations     = YF_locations,
                    pop_size      = "location_population",
                    duration_stay = "length_of_stay",
                    num_cases     = "num_cases_time_window",
                    first_date    = "first_date_cases",
                    last_date     = "last_date_cases"
                   )
ef


# You will receive an error if a variable is specified incorrectly
YF_locations$random_variable <- runif(nrow(YF_locations))
try({
  ef <- make_epiflows(flows         = YF_flows,
                      locations     = YF_locations,
                      Pop_size      = "location_population",
                      duration_stay = "length_of_stay",
                      num_cases     = "num_cases_time_window",
                      first_date    = "first_date_cases",
                      last_date     = "last_date_cases",
                      random        = "random_variable"
                     )
   })


# If you create a new method and need other varaibles, or just want a shorter
# representation, they can be added to your options:

global_vars("random", set = TRUE)

YF_locations$random_variable <- runif(nrow(YF_locations))
ef <- make_epiflows(flows         = YF_flows,
                    locations     = YF_locations,
                    pop_size      = "location_population",
                    duration_stay = "length_of_stay",
                    num_cases     = "num_cases_time_window",
                    first_date    = "first_date_cases",
                    last_date     = "last_date_cases",
                    random        = "random_variable"
                   )
```

```
# You can also reset the variables
global_vars(reset = TRUE)
```

`grid_epiflows` *Visualise epidemic flows using a grid*

Description

This grid plot shows flows between locations by positioning origins and destination in y and x axes, respectively. This function is called when plotting `epiflows` with `type = "grid"`.

Usage

```
grid_epiflows(x, color_by = c("from", "to", "none"), ...)
```

Arguments

`x`	An `epiflows` object.
`color_by`	A character string indicating if flows should be colored by origin (`from`) or destination (`to`).
`...`	arguments passed on to `ggplot2::geom_point()`

Author(s)

Thibaut Jombart

`make_epiflows` *Create an epiflows object*

Description

An epiflows object contains a pair of data frames that provide information about locations and flows between locations.

Usage

```
make_epiflows(...)

## S3 method for class 'data.frame'
make_epiflows(
  flows,
  locations = NULL,
  from = 1L,
  to = 2L,
  n = 3L,
```

```
  id = 1L,
  ...
)

## S3 method for class 'integer'
make_epiflows(inflow, outflow, focus, locations, id = 1L, ...)

## S3 method for class 'numeric'
make_epiflows(inflow, outflow, focus, locations, id = 1L, ...)
```

Arguments

`...`	Any number of varaibles that can be used for mapping or modelling. See `global_vars()` and `get_vars()` for details.
`flows`	a data frame where each row represents a flow from one location to the next. This must have at least three columns: • Where the flow started (as specified in `from`, below) • Where the flow ended (as specified in `to`, below) • How many cases were involved (as specified in `n`, below)
`locations`	a data frame where each row represents a location. This can have any number of columns specifying useful metadata about the location, but it must contain at least one column specifying the location ID used in the `flows` data frame (as specified by the `id` argument, below).
`from`	the column in the `flows` data frame indicating where the flow started. This can be an integer or character. Default is the first column.
`to`	the column in the `flows` data frame indicating where the flow terminated. This can be an integer or character. Default is the second column.
`n`	the column in the `flows` data frame indicating how many cases were contained in the flow. This can be an integer or character. Default is the third column.
`id`	The column to use for the identifier in the `locations` data frame. This defaults to the first column.
`inflow`	a **named** integer or numeric vector specifying the number of cases flowing into a location specified by `focus`.
`outflow`	a **named** integer or numeric vector specifying the number of cases flowing from a location specified by `focus`.
`focus`	a character vector specifying the focal location for integer input. This is necessary for integer input to make clear what "outflow" and "inflow" are relative to.

Details

The `epiflows` object can be constructed using simply a list of locations with optional metadata (similar to a linelist) and a list of flows that describes the number of cases flowing from one location to another. Optional metadata such as coordinates and duration of stay can be included in the linelist for use in `estimate_risk_spread()` or `map_epiflows()`.

Developer note: object structure:

Because a flow of cases from one location to another can be thought of as a contact with a wider scope, the `epiflows` object inherits from the `epicontacts` object, constructed via `epicontacts::make_epicontacts()`. This means that all the methods for subsetting an object of class `epicontacts` also applies to `epiflows`, including the use of the function `epicontacts::thin()`. One caveat is that, internally, the names of the elements within the object do not match the terminology used in *epiflows*.

Value

An `epiflows` object in list format with four elements:

- **locations** (accessible via `get_locations()`): a data frame of locations with first column 'id' containing character vector of unique identifiers.
- **flows** (accessible via `get_flows()`): data.frame of flows with first two columns named 'from' and 'to' indicating directed flows between two locations, and a third column named 'n', specifying the number of cases in each
- **vars** (accessible via `get_vars()`). This contains a named list of available variables that can be used in further plotting and/or modelling. Default variables are found in `global_vars()`:
  - `coordinates`: two columns specifying the lon and lat coordinates
  - `pop_size`: population size of each location
  - `duration_stay`: the average duration of stay for each location
  - `first_date`: the date of first recorded case
  - `last_date`: the date of the last recorded case
  - `num_cases`: the number of cases between the first and last date

Author(s)

Zhian Kamvar, Thibaut Jombart

See Also

`global_vars()` for definitions of global variables, `estimate_risk_spread()` for modelling, `plot.epiflows()` for plotting, `add_coordinates()` for adding coordinates, `get_vars()` for accession of metadata, `get_locations()` to access the locations data frame, `get_flows()` to access the flows data frame.

Examples

```
## Load data
data(YF_flows)
data(YF_locations)
YF_flows
YF_locations
## Use both to create the epiflows object.
ef <- make_epiflows(flows         = YF_flows,
                    locations     = YF_locations,
                    pop_size      = "location_population",
                    duration_stay = "length_of_stay",
                    num_cases     = "num_cases_time_window",
                    first_date    = "first_date_cases",
```

```
                    last_date     = "last_date_cases"
                   )
ef
# Access variable information
get_pop_size(ef)
get_vars(ef, "duration_stay")
get_vars(ef, "num_cases")
data(YF_Brazil)
(inflows   <- YF_Brazil$T_O["Espirito Santo", , drop = TRUE])
(outflows  <- YF_Brazil$T_D["Espirito Santo", , drop = TRUE])
(locations <- subset(YF_Brazil$states, location_code == "Espirito Santo", drop = FALSE))
los <- data.frame(location_code    = names(YF_Brazil$length_of_stay),
                  length_of_stay   = YF_Brazil$length_of_stay,
                  stringsAsFactors = FALSE
                 )
locations <- merge(x   = locations,
                   y   = los,
                   by  = "location_code",
                   all = TRUE)
ef <- make_epiflows(inflow = inflows,
                    outflow = outflows,
                    focus = "Espirito Santo",
                    locations = locations,
                    pop_size = "location_population",
                    duration_stay = "length_of_stay",
                    num_cases = "num_cases_time_window",
                    first_date = "first_date_cases",
                    last_date = "last_date_cases"
                   )
ef
```

`map_epiflows` *Map flows of people between locations*

Description

The function `map_epiflows` uses `leaflet` to generate an interactive map displaying flows of people travelling between locations stored in a `epiflows` object. Note that the object needs to possess geographic coordinates.

Usage

```
map_epiflows(
  x,
  title = "",
  center = NULL,
  sort = TRUE,
  pal = "YlOrBr",
  adjust_width = TRUE,
  ...
)
```

Arguments

x	An epiflows object.
title	Plot title.
center	An optional set of coordinates or character string specifying ID to use as the center of the map
sort	a logical. When TRUE (default), the flows will be sorted in order of number of cases on the map so that the largest flows appear on top.
pal	a color palette to pass on to leaflet::colorQuantile(). This can be the name of a viridis or RColorBrewer palette, a vector of hex colors, or a color-generating functon.
adjust_width	a logical specifying if the width of the flows should be adjusted to reflect the number of flows between locations. Defaults to TRUE.
...	Additional parameters (not used).

Value

A leaflet object

Author(s)

Paula Moraga, Pawel Piatkowski, Salla Toikkanen, Zhian Kamvar

Examples

```
data("Brazil_epiflows")
data("YF_coordinates")
ef <- add_coordinates(Brazil_epiflows, YF_coordinates[-1])
plot(ef)
map_epiflows(ef, center = "Espirito Santo", title = "Flows to and from Brazil")
```

---

plot.epiflows *Various plots for epiflows objects*

Description

The plot method for epiflows objects offers types of graphics (argument type):

Usage

```
## S3 method for class 'epiflows'
plot(x, type = c("map", "network", "grid"), ...)
```

Arguments

x	an epiflows object.
type	The type of plot to produce (defaults to map).
...	arguments passed on to a given type

Details

- `map`: flows are displayed on an interactive map; see map_epiflows for more details
- `network`: flows are displayed as a network using the htmlwidget `visNetwork` and the plotting method for `epicontacts` objects; see vis_epiflows for more details
- `grid`: flows are displayed as a grid between origins and destinations; see grid_epiflows for more details

Author(s)

Salla Toikkanen, Thibaut Jombart, Zhian Kamvar

Examples

```
data("Brazil_epiflows")

# no coordinates, defaults to network
plot(Brazil_epiflows)

# grid bubbleplot
plot(Brazil_epiflows, "grid")

# adding coordinates defaults to map
data("YF_coordinates")
ef <- add_coordinates(Brazil_epiflows, YF_coordinates[-1])
plot(ef)
```

`print.epiflows` *Print method from epiflows objects*

Description

Displays a short summary of an `epiflows` object.

Usage

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

Arguments

`x`	An `epiflows` object.
`...`	Additional parameters (not used).

Author(s)

Zhian N. Kamvar, Thibaut Jombart

Examples

```
data("Brazil_epiflows")
print(Brazil_epiflows)
```

`vis_epiflows` *Visualise epidemic flows using visNetwork*

Description

This function shows flows between locations using a dynamic network visualisation implemented in the package **visNetwork**, calling the function `epicontacts::vis_epicontacts()` of the **epicontacts** package. The thickness of arrows/edges is proportional to the corresponding flows.

Usage

```
vis_epiflows(x, arrows = TRUE, max_width = 10, ...)
```

Arguments

`x`	An `epiflows` object.
`arrows`	A logical indicating if arrows should be used to show directionality of the flows.
`max_width`	A single number indicating the maximum width of an edge (corresponding to the largest flow).
`...`	Further arguments passed to `epicontacts::vis_epicontacts()`

Author(s)

Thibaut Jombart

`YF_Brazil` *Yellow Fever Data from Brazil; 2016-12 to 2017-05*

Description

This data set contains flows to and from five states in Brazil formatted in a list with the following items:

Usage

```
data("Brazil_epiflows")
  data("YF_coordinates")
  data("YF_locations")
  data("YF_flows")
  data("YF_Brazil")
```

Format

An object of class `list` of length 4.

Details

- `$states`: a data frame containing metadata for five Brazilian States: Espirito Santo, Minas Gerais, Rio de Janeiro, Sao Paulo, and Southeast Brazil
  - `$location_code` : names of the states
  - `$location_population` : population size for each state
  - `$num_cases_time_window` : number of cases recorded between 2016-12 and 2017-05
  - `$first_date_cases` : date of first disease case in the given location in ISO 8601 format
  - `$last_date_cases` : date of last disease case in the given location in ISO 8601 format
- `$T_D` A matrix containing the number of travellers from the infectious location visiting other locations
- `$T_O` A matrix containing the number of travellers visiting the infectious location
- `$length_of_stay` A named vector containing the average length of stay in days of travellers from other locations visiting the infectious locations.

References

Dorigatti I, Hamlet A, Aguas R, Cattarino L, Cori A, Donnelly CA, Garske T, Imai N, Ferguson NM. International risk of yellow fever spread from the ongoing outbreak in Brazil, December 2016 to May 2017. Euro Surveill. 2017;22(28):pii=30572. DOI: doi:10.2807/15607917.ES.2017.22.28.30572

See Also

`make_epiflows()` for transformation to an epiflows object `estimate_risk_spread()`

Examples

```
# This is an example of an epiflows object
data("Brazil_epiflows")
Brazil_epiflows

# The above data was constructed from a data frame containing flows and
# one containing location metadata
data("YF_flows")
data("YF_locations")
ef <- make_epiflows(flows         = YF_flows,
                    locations     = YF_locations,
                    pop_size      = "location_population",
                    duration_stay = "length_of_stay",
                    num_cases     = "num_cases_time_window",
                    first_date    = "first_date_cases",
                    last_date     = "last_date_cases"
                   )

# Both of the above data frames were constructed like so:
```

```
data("YF_Brazil")

# Create the flows data frame
from  <- as.data.frame.table(YF_Brazil$T_D, stringsAsFactors = FALSE)
to    <- as.data.frame.table(t(YF_Brazil$T_O), stringsAsFactors = FALSE)
flows <- rbind(from, to)
colnames(flows) <- c("from", "to", "n")

## Create the locations data frame
los <- data.frame(location_code    = names(YF_Brazil$length_of_stay),
                  length_of_stay   = YF_Brazil$length_of_stay,
                  stringsAsFactors = FALSE
                 )
locations <- merge(x   = YF_Brazil$states,
                   y   = los,
                   by  = "location_code",
                   all = TRUE)

## Use both to create the epiflows object.
ef <- make_epiflows(flows,
                    locations,
                    pop_size = "location_population",
                    duration_stay = "length_of_stay",
                    num_cases = "num_cases_time_window",
                    first_date = "first_date_cases",
                    last_date = "last_date_cases"
)
```

`[.epiflows` *Subset 'epiflows' objects*

Description

An epiflows object inherits the epicontacts class, so the subsetting mechanism is also inherited. The benefits is that it's extremely flexible. However, this also means that it's possible for the contacts to contain IDs that are not present in the locations metadata and vice versa. The best way to consistently subset an epiflows object is present in the examples.

Usage

```
## S3 method for class 'epiflows'
x[i, j, k = TRUE, l = TRUE, ...]
```

Arguments

x	An epiflows object.
i	An integer, logical, or character vector of one or more location.
j	An integer, logical, or character vector to subset the flows data frame.

k	A character vector of one or more columns to be retained in the location data.
l	A character vector of one or more columns to be retained in the flows data frame. Note: if using numbers, the first column stands for the first column after "n".
...	Additional parameters passed to [.epicontacts.

Details

Returns a subset of an epiflows object.

Value

An epiflows object.

Author(s)

Zhian N. Kamvar

Examples

```
data(Brazil_epiflows)
# You can subset, but the flows information will still be present
Brazil_epiflows[j = "Espirito Santo"]
# To help with this, use `thin` from epiflows
epicontacts::thin(Brazil_epiflows[j = "Espirito Santo"])
epicontacts::thin(Brazil_epiflows[j = c("Espirito Santo", "Rio de Jenerio")])
```

Index