Package ‘epitrix’

January 14, 2023

Title  Small Helpers and Tricks for Epidemics Analysis
Version  0.4.0

Description  A collection of small functions useful for epidemics analysis and infectious disease modelling. This includes computation of basic reproduction numbers from growth rates, generation of hashed labels to anonymize data, and fitting discretized Gamma distributions.

Depends  R (>= 3.3.0)
License  MIT + file LICENSE
Encoding  UTF-8

Suggests  testthat, roxygen2, outbreaks, incidence (>= 1.4.1), knitr, rmarkdown, magrittr, ggplot2, tibble, covr
Imports  sodium, distcrete, stringi, dplyr, purrr, rlang, tidyr
RoxygenNote  7.2.3

Config/testthat.edition  3

URL  http://www.repidemicsconsortium.org/epitrix/

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

VignetteBuilder  knitr

NeedsCompilation  no

Author  Thibaut Jombart [aut, cre],
        Anne Cori [aut],
        Zhian N. Kamvar [ctb],
        Dirk Schumacher [ctb],
        Flavio Finger [aut],
        Charlie Whittaker [ctb]

Maintainer  Thibaut Jombart <thibautjombart@gmail.com>

Repository  CRAN

Date/Publication  2023-01-13 23:40:02 UTC
**R topics documented:**

AR2R0 ................................................................. 2
clean_labels ..................................................... 3
empirical_incubation_dist ................................. 4
fit_disc_gamma .................................................. 6
fit_gamma_incubation_dist ................................. 7
gamma_shapescale2mucv .................................... 8
hash_names ...................................................... 10
R02AR ............................................................. 12
R02herd_immunity_threshold ............................... 13
r2R0 .............................................................. 13
sim_linelist ..................................................... 15

**Index** 16

---

**AR2R0**  
*Title Calculate basic reproduction number from attack rate*

**Description**

Title Calculate basic reproduction number from attack rate

**Usage**

`AR2R0(AR)`

**Arguments**

`AR`  
the attack rate; a value or vector of values between 0 and 1

**Value**

`R0`, the basic reproduction number, calculated as `-log(1-AR)/AR`

**Examples**

```r
## Calculate R0 for an attack rate of 50%
AR2R0(0.5)

## plot the relationship between R0 and attack rate
x <- seq(0.01, 1, 0.01)
plot(AR2R0(x), x, type = "l", xlab = "R0", ylab = "Attack rate")
```
Description

This function standardises labels e.g. used as variable names or character string values, removing non-ascii characters, replacing diacritics (e.g. é, ô) with their closest ascii equivalents, and standardises separating characters. See details for more information on label transformation.

Usage

clean_labels(
  x,
  sep = "_",
  transformation = "Any-Latin; Latin-ASCII",
  protect = ""
)

Arguments

x A vector of labels, normally provided as characters.
sep A character string used as separator, defaulting to '_'.
transformation a string to be passed on to stringi::stri_trans_general() for conversion. Default is "Any-Latin; Latin-ASCII", which will convert any non-latin characters to latin and then converts all accented characters to ASCII characters. See stringi::stri_trans_list() for a full list of options.
protect a character string defining the punctuation that should be protected. This helps prevent meaningful symbols like > and < from being removed.

Details

The following changes are performed:

- all non-ascii characters are removed
- all diacritics are replaced with their non-accentuated equivalents, e.g. 'é', 'ê' and 'è' become 'e'.
- all characters are set to lower case
- separators are standardised to the use of a single character provided in sep (defaults to '_'); heading and trailing separators are removed.

Note

Because of differences between the underlying transliteration engine (ICU), the default transiliterations will not transiliterate German umlaute correctly. You can add them by specifying "de-ASCII" in the transformation string after "Any-Latin".
empirical_incubation_dist

Extract empirical incubation period distribution from linelist data

Description

This function takes in a linelist data frame and extracts the empirical incubation period distribution and can take into account uncertainty in the dates of exposure.

Usage

```r
empirical_incubation_dist(x, date_of_onset, exposure, exposure_end = NULL)
```
Arguments

- **x**: the linelist data (data.frame or linelist object) containing at least a column containing the exposure dates and one containing the onset dates.
- **date_of_onset**: the name of the column containing the onset dates (bare variable name or in quotes)
- **exposure**: the name of the column containing the exposure dates (bare variable name or in quotes)
- **exposure_end**: the name of a column containing dates representing the end of the exposure period. This is ‘NULL’ by default, indicating all exposures are known and in the 'exposure' column.

Value

A data frame containing a column with the different incubation periods and a column containing their relative frequency.

Note

For exposure dates, each element can be a vector containing several possible exposure dates. Note that if the same exposure date appears twice in the list it is given twice as much weight.

Author(s)

Flavio Finger, `<flavio.finger@lshtm.ac.uk>`, Zhian N. Kamvar

Examples

```r
if (require(tibble)) {
  random_dates <- as.Date("2020-01-01") + sample(0:30, 50, replace = TRUE)
  x <- tibble(date_of_onset = random_dates)

  # Linelist with a list column of potential exposure dates ------------------
  mkexposures <- function(x) x - round(rgamma(sample.int(5, size = 1), shape = 12, rate = 3))
  exposures <- sapply(x$date_of_onset, mkexposures)
  x$date_exposure <- exposures

  incubation_period_dist <- empirical_incubation_dist(x, date_of_onset, date_exposure)
  incubation_period_dist

  # Linelist with exposure range -------------------------------------------
  start_exposure <- round(rgamma(nrow(x), shape = 12, rate = 3))
  end_exposure <- round(rgamma(nrow(x), shape = 12, rate = 7))
  x$exposure_end <- x$date_of_onset - end_exposure
  x$exposure_start <- x$exposure_end - start_exposure

  incubation_period_dist <- empirical_incubation_dist(x, date_of_onset, exposure_start, exposure_end)
  incubation_period_dist

  plot(incubation_period_dist,
       type = "h", lwd = 10, lend = 2, col = "#49D193",
       xlab = "Days since exposure",
       ylab = "Probability")
}
```
fit_disc_gamma

fit_disc_gamma(x, mu_ini = NULL, cv_ini = NULL, interval = 1, w = 0, ...)

Arguments

x A vector of numeric data to fit; NAs will be removed with a warning.
mu_ini The initial value for the mean 'mu', defaulting to the empirically calculated value.
cv_ini The initial value for the coefficient of variation 'cv', defaulting to the empirically calculated value.
interval The interval used for discretisation; see distcrete.
w The centering of the interval used for discretisation; see distcrete.
... Further arguments passed to optim.

Value

The function returns a list with human-readable parametrisation of the discretised Gamma distribution (mean, sd, cv), convergence indicators, and the discretised Gamma distribution itself as a distcrete object (from the distcrete package).

Author(s)

Thibaut Jombart <thibautjombart@gmail.com>
Charlie Whittaker <charles.whittaker16@imperial.com>

See Also

The distcrete package for discretising distributions, and optim for details on available optimisation procedures.
Examples

```r
## generate data

mu <- 15.3 # days
sigma <- 9.3 # days
cv <- sigma / mu
cv
param <- gamma_mucv2shapescale(mu, cv)

if (require(distcrete)) {
w <- distcrete("gamma", interval = 1,
    shape = param$shape,
    scale = param$scale, w = 0)

x <- w$r(100)
x

fit_disc_gamma(x)
}
```

---

**Fit discrete gamma distribution to incubation periods**

**Description**

A wrapper around fit_disc_gamma to fit a discrete gamma distribution to incubation periods derived from exposure and onset dates. Can take into account uncertain dates of exposure.

**Usage**

```r
fit_gamma_incubation_dist(
    x,  # the linelist data (data.frame or linelist object) containing at least a column containing the exposure dates and one containing the onset dates.
    date_of_onset,  # the name of the column containing the onset dates (bare variable name or in quotes)
    exposure,  # exposure dates
    exposure_end = NULL,  # exposure end dates
    nsamples = 1000,  # number of samples to fit
    ...  # additional arguments
)
```

**Arguments**

- `x`
- `date_of_onset`
exposure the name of the column containing the exposure dates (bare variable name or in quotes)

exposure_end the name of a column containing dates representing the end of the exposure period. This is 'NULL' by default, indicating all exposures are known and in the 'exposure' column.

nsamples The number of samples to draw from the empirical distribution to fit on (defaults to 1000)

... passed to fit_disc_gamma

Value see [fit_disc_gamma()]

Author(s)

Flavio Finger, <flavio.finger@lshtm.ac.uk>

Examples

```r
random_dates <- as.Date("2020-01-01") + sample(0:30, 50, replace = TRUE)
x <- data.frame(date_of_onset = random_dates)

mkexposures <- function(x) x - round(rgamma(sample.int(5, size = 1), shape = 12, rate = 3))
exposures <- sapply(x$date_of_onset, mkexposures)
x$date_exposure <- exposures

fit <- fit_gamma_incubation_dist(x, date_of_onset, date_exposure)
plot(0:20, fit$distribution$d(0:20),
    type = "h", lwd = 10, lend = 2, col = "#49D193",
    xlab = "Days since exposure",
    ylab = "Probability",
    main = "Incubation time distribution")
```

---

gamma_shapescale2mucv Reparameterise Gamma distributions

description

These functions permit to use alternate parametrisations for Gamma distributions, from 'shape and scale' to 'mean (mu) and coefficient of variation (cv), and back. gamma_shapescale2mucv does the first conversion, while gamma_mucv2shapescale does the second. The function gamma_log_likelihood is a shortcut for computing Gamma log-likelihood with the alternative parametrisation (mean, cv). See 'details' for a guide of which parametrisation to use.
gamma_shapescale2mucv

Usage

gamma_shapescale2mucv(shape, scale)

gamma_mucv2shapescale(mu, cv)

gamma_log_likelihood(
  x,
  mu,
  cv,
  discrete = TRUE,
  interval = 1,
  w = 0,
  anchor = 0.5
)

Arguments

shape      The shape parameter of the Gamma distribution.
scale      The scale parameter of the Gamma distribution.
mu         The mean of the Gamma distribution.
cv         The coefficient of variation of the Gamma distribution, i.e. the standard deviation divided by the mean.
x          A vector of data treated as observations drawn from a Gamma distribution, for which the likelihood is to be computed.
discrete   A logical indicating if the distribution should be discretised; TRUE by default.
interval   The interval used for discretisation; see distcrete.
w          The centering of the interval used for discretisation, defaulting to 0; see distcrete.
anchor     The anchor used for discretisation, i.e. starting point of the discretisation process; defaults to 0; see distcrete.

Details

The gamma distribution is described in ?dgamma is parametrised using shape and scale (or rate). However, these parameters are naturally correlated, which makes them poor choices whenever trying to fit data to a Gamma distribution. Their interpretation is also less clear than the traditional mean and variance. When fitting the data, or reporting results, it is best to use the alternative parametrisation using the mean (mu) and the coefficient of variation (cv), i.e. the standard deviation divided by the mean.

Value

A named list containing 'shape' and 'scale', or mean ('mean') and coefficient of variation ('cv').

Author(s)

Code by Anne Cori <a.cori@imperial.ac.uk>, packaging by Thibaut Jombart <thibautjombart@gmail.com>
Examples

```r
## set up some parameters
mu <- 10
cv <- 1

## transform into shape scale
tmp <- gamma_mucv2shapescale (mu, cv)
shape <- tmp$shape
scale <- tmp$scale

## recover original parameters when applying the revert function
gamma_shapescale2mucv(shape, scale) # compare with mu, cv

## empirical validation:
## check mean / cv of a sample derived using rgamma with
## shape and scale computed from mu and cv
gamma_sample <- rgamma(n = 10000, shape = shape, scale = scale)
mean(gamma_sample) # compare to mu
sd(gamma_sample) / mean(gamma_sample) # compare to cv
```

## hash_names

Anonymise data using scrypt

### Description

This function uses the scrypt algorithm from libsodium to anonymise data, based on user-indicated data fields. Data fields are concatenated first, then each entry is hashed. The function can either return a full detailed output, or short labels ready to use for ‘anonymised data’. Before concatenation (using "_" as a separator) to form labels, inputs are modified using [clean_labels()]

### Usage

```
hash_names(
  ..., 
  size = 6, 
  full = TRUE, 
  hashfun = "secure", 
  salt = NULL, 
  clean_labels = TRUE
)
```
hash_names

Arguments

- **...**
  Data fields to be hashed.
- **size**
  The number of characters retained in the hash.
- **full**
  A logical indicating if the a full output should be returned as a data.frame, including original labels, shortened hash, and full hash.
- **hashfun**
  This defines the hashing function to be used. If you specify "secure" (default), it will use [sodium::scrypt()], which will be secure, but will be slow for large data sets. For fast hashing with no collisions, you can sepecify "fast", and it will use [sodium::sha256()], which is several orders of magnitude faster than [sodium::scrypt()]. You can also specify a hashing function that takes and returns a [raw][base::raw] vector of bytes that can be converted to character with [rawToChar()].
- **salt**
  An optional object that can be coerced to a character to be used to 'salt' the hashing algorithm (see details). Ignored if ‘NULL’.
- **clean_labels**
  A logical indicating if labels of variables should be standardized; defaults to ‘TRUE’

Details

The argument 'salt' should be used for salting the algorithm, i.e. adding an extra input to the input fields (the 'salt') to change the resulting hash and prevent identification of individuals via pre-computed hash tables.

It is highly recommend to choose a secret, random salt in order make it harder for an attacker to decode the hash.

Author(s)

Thibaut Jombart <thibautjombart@gmail.com>, Dirk Schumacher <mail@dirk-schumacher.net>, Zhian N. Kamvar <zkamvar@gmail.com>

See Also

[clean_labels()], used to clean labels prior to hashing
[sodium::hash()] for available hashing functions.

Examples

```r
first_name <- c("Jane", "Joe", "Raoul")
last_name <- c("Doe", "Smith", "Dupont")
age <- c(25, 69, 36)

# secure hashing
hash_names(first_name, last_name, age, hashfun = "secure")

# fast hashing
hash_names(first_name, last_name, age, 
           size = 8, full = FALSE, hashfun = "fast")
```
## salting the hashing (more secure!)

hash_names(first_name, last_name) # unsalted - less secure  
hash_names(first_name, last_name, salt = 123) # salted with an integer  
hash_names(first_name, last_name, salt = "foobar") # salted with a character  

## using a different hash algorithm if you want things to run faster  
hash_names(first_name, last_name, hashfun = "fast") # use sha256 algorithm

---

**R02AR**  
*Title Calculate attack rate from basic reproduction number*

### Description

Title Calculate attack rate from basic reproduction number

### Usage

R02AR(R0, tol = 0.01)

### Arguments

- **R0**: a value or vector of values representing the basic reproduction number, must be \( \geq 0 \)
- **tol**: a single \( \geq 0 \) value giving the tolerance for the calculated attack rate

### Value

AR, the attack rate, calculated using the relationship: \( R_0 = \frac{-\log(1-AR)}{AR} \)

### Examples

```r
## Calculate the attack rate for a specific value of the reproduction number  
R02AR(2) # returns the AR for an R0 of 2

## plot the relationship between R0 and attack rate  
x <- seq(1.01, 5, 0.01)  
plot(x, R02AR(x), type = "l", xlab = "R0", ylab = "Attack rate")
```
Title Calculate herd immunity threshold from basic reproduction number

Usage

R02herd_immunity_threshold(R0)

Arguments

R0 a value or vector of values representing the basic reproduction number, must be >=0

Value

The herd immunity threshold, calculated as 1 - 1 / R0

Examples

## Calculate the herd immunity threshold for a specific value of the ## reproduction number (here 2)
R02herd_immunity_threshold(2)

## plot the relationship between R0 and herd immunity threshold
x <- seq(1.01, 15, 0.01)
plot(x, R02herd_immunity_threshold(x), type = "l",
xlab = "R0", ylab = "Herd immunity threshold")

Transform a growth rate into a reproduction number

Description

The function r2R0 can be used to transform a growth rate into a reproduction number estimate, given a generation time distribution. This uses the approach described in Wallinga and Lipsitch (2007, Proc Roy Soc B 274:599–604) for empirical distributions. The function lm2R0_sample generates a sample of R0 values from a log-linear regression of incidence data stored in a lm object.
Usage

\[ r2R0(r, w, \text{trunc} = 1000) \]

\[ \text{lm2R0}\_\text{sample}(x, w, n = 100, \text{trunc} = 1000) \]

Arguments

- **r**: A vector of growth rate values.
- **w**: The serial interval distribution, either provided as a `distcrete` object, or as a numeric vector containing probabilities of the mass functions.
- **trunc**: The number of time units (most often, days), used for truncating \( w \), whenever a `distcrete` object is provided. Defaults to 1000.
- **x**: A `lm` object storing a linear regression of log-incidence over time.
- **n**: The number of draws of R0 values, defaulting to 100.

Details

It is assumed that the growth rate ('r') is measured in the same time unit as the serial interval ('w' is the SI distribution, starting at time 0).

Author(s)

Code by Anne Cori &lt;a.cori@imperial.ac.uk&gt;, packaging by Thibaut Jombart &lt;thibautjombart@gmail.com&gt;

Examples

```r
## Ebola estimates of the SI distribution from the first 9 months of
## West-African Ebola outbreak

mu <- 15.3  # days
sigma <- 9.3  # days
param <- gamma_mucv2shapescale(mu, sigma / mu)

if (require(distcrete)) {
  w <- distcrete("gamma", interval = 1,
                 shape = param$shape,
                 scale = param$scale, w = 0)

  r2R0(c(-1, -0.001, 0, 0.001, 1), w)
}

## Use simulated Ebola outbreak and 'incidence' to get a log-linear
## model of daily incidence.

if (require(outbreaks) && require(incidence)) {
  i <- incidence(ebola_sim$linelist$date_of_onset)
  plot(i)
  f <- fit(i[1:100])
  f
```
`sim_linelist`  

Simulate simple linelist data

**Description**

This function simulates a simple linelist data including dates of epidemiological events and basic patient information. No underlying epidemiological model is used.

**Usage**

```r
sim_linelist(
  n = 1,
  onset_from = as.Date("2020-01-01"),
  onset_span = 60,
  report_delay = 7,
  cfr = 0.1
)
```

**Arguments**

- **n**: Number of entries to simulate.
- **onset_from**: The earliest date of symptom onset which can be generated.
- **onset_span**: The time span over which to generate dates of onset.
- **report_delay**: The mean delay between onset and reporting, using a Poisson distribution.
- **cfr**: The case fatality ratio, i.e. the proportion of patient dying from the infection (used to generate the 'outcome' variable).

**Author(s)**

Thibaut Jombart <thibautjombart@gmail.com>

**Examples**

```r
sim_linelist(10)
```
Index

AR2R0, 2

clean_labels, 3

distcrete, 6, 9

distcrete, 6, 9

empirical_incubation_dist, 4

fit_disc_gamma, 6

fit_discrete(fit_disc_gamma), 6

fit_disc_gamma, 6

fit_discrete(fit_disc_gamma), 6

fit_disc_gamma, 6

fit_discrete(fit_disc_gamma), 6

fit_discrete(fit_disc_gamma), 6

fit_discrete(fit_disc_gamma), 6

fit_discrete(fit_disc_gamma), 6

fit_discrete(fit_disc_gamma), 6

fit_discrete(fit_disc_gamma), 6

fit_discrete(fit_disc_gamma), 6

fit_discrete(fit_disc_gamma), 6

fit_discrete(fit_disc_gamma), 6

fit_discrete(fit_disc_gamma), 6

fit_discrete(fit_disc_gamma), 6

fit_discrete(fit_disc_gamma), 6

fit_discrete(fit_disc_gamma), 6

fit_discrete(fit_disc_gamma), 6

fit_discrete(fit_disc_gamma), 6

fit_discrete(fit_disc_gamma), 6

fit_discrete(fit_disc_gamma), 6

fit_discrete(fit_disc_gamma), 6

fit_discrete(fit_disc_gamma), 6

fit_discrete(fit_disc_gamma), 6

fit_discgamma, 6

fit_discgamma, 6

gamma_log_likelihood

gamma_shapescale2mucv, 8

gamma_mucv2shapescale

gamma_shapescale2mucv, 8

gamma_shapescale2mucv, 8

hash_names, 10

lm2R0_sample(r2R0), 13

optim, 6

R02AR, 12

R02herd_immunity_threshold, 13

r2R0, 13

sim_linelist, 15

stringi::stri_trans_general(), 3

stringi::stri_trans_list(), 3