Package 'epitrix'

January 15, 2019

Title Small Helpers and Tricks for Epidemics Analysis

Version 0.2.2

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 anonymise data, and fitting discretised Gamma distributions.

Depends R (>= 3.3.0)

License MIT + file LICENSE

Encoding UTF-8

LazyData true

Suggests testthat, roxygen2, outbreaks, incidence (>= 1.4.1), knitr, rmarkdown

Imports sodium, distcrete, stringi

RoxygenNote 6.1.1

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]

Maintainer Thibaut Jombart <thibautjombart@gmail.com>

Repository CRAN

Date/Publication 2019-01-15 11:50:03 UTC

R topics documented:

`clean_labels` *Standardise labels*

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 = "_")
```

Arguments

`x`	A vector of labels, normally provided as characters.
`sep`	A character string used as separator, defaulting to '_'.

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', 'ö' becomes 'o', etc.
- 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.

Author(s)

Thibaut Jombart <thibautjombart@gmail.com>

Examples

```
clean_labels("-_-This is; A    WeÏrD**./sêntënce...")
clean_labels("-_-This is; A    WeÏrD**./sêntënce...", sep = ".")
input <- c("Peter and stëven", "peter-and.stëven", "pëtêr and stëven  _-")
input
clean_labels(input)
```

`fit_disc_gamma` *Fit discretised distributions using ML*

Description

These functions performs maximum-likelihood (ML) fitting of a discretised distribution. This is typically useful for describing delays between epidemiological events, such as incubation period (infection to onset) or serial intervals (primary to secondary onsets). The function `optim` is used internally for fitting.

Usage

```
fit_disc_gamma(x, mu_ini = 1, cv_ini = 1, 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 1.
`cv_ini`	The initial value for the coefficient of variation 'cv', defaulting to 1.
`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 distibution (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>

See Also

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

Examples

```
## generate data

mu <- 15.3 # days
sigma <- 9.3 # days
cv <- mu / sigma
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)
}
```

---

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.

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 make 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

```
## 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, salt = NULL)
```

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.
salt	An optional object that can be coerced to a character to be used to 'salt' the hashing algorithm (see details). Ignored if NULL (default).

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 Shchumacher <mail@dirk-schumacher.net>

See Also

clean_labels, used to clean labels prior to hashing.

Examples

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

hash_names(first_name, last_name, age)
```

```
hash_names(first_name, last_name, age,
           size = 8, full = FALSE)


## 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 an character
```

---

r2R0 *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, trunc = 1000)

lm2R0_sample(x, w, n = 100, 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 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 <a.cori@imperial.ac.uk>, packaging by Thibaut Jombart <thibautjombart@gmail.com>

Examples

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

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
    plot(i[1:150], fit = f)

    R0 <- lm2R0_sample(f$model, w)
    hist(R0, col = "grey", border = "white", main = "Distribution of R0")
    summary(R0)
  }
}
```

Index