Package 'epigrowthfit'

July 22, 2025

Version 0.15.4

Date 2025-02-14

License GPL-3

Encoding UTF-8

URL https://github.com/davidearn/epigrowthfit

BugReports https://github.com/davidearn/epigrowthfit/issues

Title Nonlinear Mixed Effects Models of Epidemic Growth

Description Maximum likelihood estimation of nonlinear mixed effects models of epidemic growth using Template Model Builder ('TMB'). Enables joint estimation for collections of disease incidence time series, including time series that describe multiple epidemic waves. Supports a set of widely used phenomenological models: exponential, logistic, Richards (generalized logistic), subexponential, and Gompertz. Provides methods for interrogating model objects and several auxiliary functions, including one for computing basic reproduction numbers from fitted values of the initial exponential growth rate.
Preliminary versions of this software were applied in Ma et al. (2014) <doi:10.1007/s11538-013-9918-2> and in Earn et al. (2020) <doi:10.1073/pnas.2004904117>.

Depends R (>= 4.3)

LinkingTo RcppEigen (>= 0.3.4.0.0), TMB

Imports Matrix (>= 1.6-2), TMB, grDevices, graphics, methods, nlme, stats, utils

LazyData true

NeedsCompilation yes

Author Mikael Jagan [aut, cre] (ORCID: <https://orcid.org/0000-0002-3542-2938>), Ben Bolker [aut] (ORCID: <https://orcid.org/0000-0002-2127-0443>), Jonathan Dushoff [ctb] (ORCID: <https://orcid.org/0000-0003-0506-4794>), David Earn [ctb] (ORCID: <https://orcid.org/0000-0003-3597-617X>), Junling Ma [ctb]

Maintainer Mikael Jagan <jaganmn@mcmaster.ca>

Repository CRAN

Date/Publication 2025-02-19 14:50:02 UTC

Contents

`epigrowthfit-package` R *Package* **epigrowthfit**

Description

An R package for estimating nonlinear mixed effects models of epidemic growth.

Details

The "main" model estimating function is `egf`.

To render a list of available help topics, use `help(package = "epigrowthfit")`. Many of these document methods for the class of objects returned by `egf`.

To report a bug or request a change, use `bug.report(package = "epigrowthfit")`.

Author(s)

Mikael Jagan <jaganmn@mcmaster.ca>

`coef.egf` *Extract Coefficients and Random Effect Covariance Parameters*

Description

Extracts the bottom level parameter vector `c(beta, theta, b)` or a subset. Segments `beta`, `theta`, and `b` contain (respectively) fixed effect coefficients, random effect covariance parameters, and random effect coefficients.

Usage

```
## S3 method for class 'egf'
coef(object, random = FALSE, full = FALSE, ...)
```

Arguments

`object`	an `egf` object.
`random`	a logical. If `FALSE`, then segment `b` is excluded.
`full`	a logical. If `FALSE`, then mapped elements are excluded, and the result is called "condensed".
`...`	unused optional arguments.

Value

A numeric vector concatenating `beta`, `theta`, and `b`, without `b` if `random = FALSE` and without mapped elements if `full = FALSE`.

Attribute `len` is a named integer vector partitioning the result by segment.

Attribute `map` is a named list of integer vectors `i` such that that a full segment `y` and its condensed counterpart `x` are related by `y = x[i]`, with the exception that `i[j]` is `NA` if `y[j]` is mapped to an initial value. `NULL` is used in place of an integer vector where `x` and `y` are identical.

The result inherits from class `coef.egf`, which has methods for `print`, `as.list`, and `labels`.

See Also

The generic function `coef`.

Examples

```
example("egf", package = "epigrowthfit")

for (random in c(FALSE, TRUE)) {
    for (full in c(FALSE, TRUE)) {
        cat(sprintf("random = %s, full = %s :\n\n", random, full))
        str(coef(m1, random = random, full = full))
        cat("\n")
    }
}
```

`confint.egf` *Confidence Intervals*

Description

Computes confidence intervals on fixed effect coefficients, random effect covariance parameters, and linear combinations thereof, including population fitted values. Intervals on individual fitted values accounting for random effects are supported, but only by `method = "wald"`.

Usage

```
## S3 method for class 'egf'
confint(object, parm, level = 0.95,
        A = seq_along(par), method = c("wald", "profile", "uniroot"), scale = 7,
        parallel = egf_parallel(), trace = FALSE,
        top = egf_top(object), subset = NULL, select = NULL,
        class = FALSE, link = TRUE, random = FALSE, ...)

## S3 method for class 'confint.egf'
plot(x, by = 12L,
     subset = NULL, order = NULL, label = NULL, main = NULL, ...)
```

Arguments

object	an egf object.
parm	unused argument, for consistency with the generic function.
level	a number in the interval $(0, 1)$ indicating a confidence level.
A	a numeric matrix with 1+p columns, where p = length(coef(object)), in which case each row specifies a linear combination of the elements of c(1, coef(fitted)) for which intervals are computed; or a valid index vector for coef(fitted), in which case intervals are computed for the indexed elements; or NULL, in which case intervals on fitted values are computed.
method	a character string indicating *how* intervals are computed.
scale	a positive number, for method = "uniroot". tmbroot will search for roots between value-scale*se and value+scale*se, where value and se are the estimate and standard error.
parallel	an egf_parallel object defining options for R level parallelization.
trace	a logical. If TRUE, then basic tracing messages indicating progress are printed. These may be mixed with optimizer output depending on object[["control"]][["trace"]].
top	a subset of egf_top(object) naming top level nonlinear model parameters for which intervals on fitted values should be computed.
subset, select	index vectors for the rows and columns of model.frame(object, "combined") or language objects evaluating to such vectors. subset indicates fitting windows for which intervals should be computed; the default indicates all. select indicates variables that should be appended to the result; the default indicates none. Evaluation of language objects follows the implementation of subset.data.frame. For the plot method: an index vector for seq_len(nrow(x)) or a language object evaluating in x to such a vector. subset indicates which intervals are plotted; the default indicates all.
class	a logical. If TRUE and if A = NULL, then the value of the confint call is a confint.egf object, not a matrix.
link	a logical. If FALSE and if A = NULL and class = TRUE, then fitted values and confidence limits are returned on the "natural" (inverse link) scale.
random	a logical, affecting only method = "wald". If TRUE, then intervals are computed for individual fitted values, which count random effects, rather than population fitted values, which do not.
...	additional arguments passed from or to other methods.
x	a confint.egf object.
by	a positive integer indicating the number of intervals displayed in one plot.
order	a permutation of seq_len(nrow(x)) or a language object evaluating in x to such a vector. order indicates the order in which intervals are plotted; the default indicates the original order.
label	a character or expression vector of length nrow(x) or a language object evaluating in x to such a vector. label indicates y-axis labels for intervals; the default is to use as.character(x[["window"]]).

main	a character or expression vector of length 1 indicating a plot title, to be recycled for all plots.

Details

Three methods for computing confidence intervals are available:

"wald" confidence limits are calculated as

```
value + c(-1, 1) * sqrt(q) * se
```

where q = qchisq(level, df = 1).

"profile", "uniroot" confidence limits are calculated as approximate solutions of the equation

```
2 * (f(x) - f(value)) = q
```

where q = qchisq(level, df = 1) and f is the negative log marginal likelihood function expressed as a function of the parameter x in question. Solutions are approximated by interpolating a likelihood profile ("profile") or by rootfinding ("uniroot").

"wald" assumes asymptotic normality of the maximum likelihood estimator. "profile" and "uniroot" avoid this contraint but are typically expensive, requiring estimation of many restricted models. They are parallelized at the C++ level when there is OpenMP support and object[["control"]][["omp_num_threads"]] is set to an integer greater than 1. When there is no OpenMP support, they can still be parallelized at the R level with appropriate setting of argument parallel.

Value

A numeric array in 2 or 3 dimensions containing the lower and upper confidence limits in the last dimension.

When confidence intervals on fitted values are desired, the user will set A = NULL and in that case have the option of passing class = TRUE to obtain an augmented result. Thus, alternatively:

A data frame inheriting from class confint.egf, with variables:

top	top level nonlinear model parameter, from egf_top(object).
ts	time series, from levels(model.frame(object)[["ts"]]).
window	fitting window, from levels(model.frame(object)[["window"]]).
value	fitted value.
ci	a numeric matrix with two columns giving the lower and upper confidence limits.
...	further variables from model.frame(object, "combined") specified by argument select.

The confidence level level is preserved as an attribute.

See Also

The generic function confint.

Examples

```
example("egf", package = "epigrowthfit")

zz1 <- confint(m1, A = NULL, method =    "wald", class = TRUE,
               random = TRUE)
str(zz1)

op <- par(mar = c(4.5, 4, 2, 1), oma = c(0, 0, 0, 0))
plot(zz1)
par(op)

zz2 <- confint(m1, A = NULL, method = "profile", class = TRUE,
               top = "log(r)", subset = quote(country == "A" & wave == 1))
zz3 <- confint(m1, A = NULL, method = "uniroot", class = TRUE,
               top = "log(r)", subset = quote(country == "A" & wave == 1))
```

`cov2theta` *Compute a Packed Representation of a Covariance Matrix*

Description

Transform covariances matrices to a "packed" representation or compute the inverse transformation.

Usage

```
cov2theta(Sigma)
theta2cov(theta)
```

Arguments

`Sigma`	an n-by-n real, symmetric positive definite matrix. Only the upper triangle is "seen".
`theta`	a numeric vector of length $n(n+1)/2$ whose first n elements are positive.

Details

An n-by-n real, symmetric, positive definite matrix Σ can be factorized as

$$\Sigma = R'R\,.$$

The upper triangular Cholesky factor R can be written as

$$R = R_1 D^{-1/2} D_\sigma^{1/2}\,,$$

where R_1 is a unit upper triangular matrix and $D = \mathrm{diag}(\mathrm{diag}(R_1' R_1))$ and $D_\sigma = \mathrm{diag}(\mathrm{diag}(\Sigma))$ are diagonal matrices.

`cov2theta` takes Σ and returns the vector θ of length $n(n+1)/2$ containing the log diagonal entries of D_σ followed by (in column-major order) the strictly upper triangular entries of R_1. `theta2cov` computes the inverse transformation.

Value

A vector like `theta` (`cov2theta`) or a matrix like `Sigma` (`theta2cov`); see 'Details'.

`covid19.ontario` *COVID-19 in Ontario, Canada*

Description

Time series of COVID-19 cases and tests in Ontario, Canada, daily from February 8, 2020 to May 1, 2022.

Usage

```
data(covid19.ontario, package = "epigrowthfit")
```

Format

A data frame with 814 rows and 3 variables:

`date` a `Date` vector.

`cases` an integer vector. `cases[i]` is the number of cases confirmed by Ontario public health units prior to `date[i]`. This number includes resolved and fatal cases as well as reinfections.

`tests` an integer vector. `tests[i]` is the number of tests completed prior to `date[i]`. This number includes repeated tests by individuals except prior to April 15, 2020, when individuals were counted at most once.

Source

This data set is a processed subset of a larger data set downloaded on 2024-01-10 from the It is updated using an installed script:

```
\link{system.file}("scripts", "covid19.ontario.R", package = "epigrowthfit")
```

Examples

```
data(covid19.ontario, package = "epigrowthfit")
plot(1 + diff(c(NA, cases)) ~ date, data = covid19.ontario, log = "y")
```

df.residual.egf *Extract the Residual Degrees of Freedom*

Description

Extracts from a model object the number of observations (see nobs) minus the number of estimated parameters (fixed effect coefficients and random effect covariance parameters).

Usage

```
## S3 method for class 'egf'
df.residual(object, ...)
```

Arguments

object	an egf object.
...	unused optional arguments.

Value

An integer.

See Also

The generic function df.residual.

egf *Fit Nonlinear Mixed Effects Models of Epidemic Growth*

Description

Fits nonlinear mixed effects models of epidemic growth to collections of one or more disease incidence time series.

Usage

```
egf(model, ...)

## S3 method for class 'egf_model'
egf(model,
    formula_ts,
    formula_windows,
    formula_parameters = list(),
    formula_priors = list(),
    data_ts,
```

```
    data_windows,
    subset_ts = NULL,
    subset_windows = NULL,
    select_windows = NULL,
    na_action_ts = c("fail", "pass"),
    na_action_windows = c("fail", "omit"),
    control = egf_control(),
    init = list(),
    map = list(),
    fit = TRUE,
    se = FALSE,
    ...)
```

Arguments

`model`	an R object specifying a top level nonlinear model, typically of class `egf_model`.
`formula_ts`	a formula of the form `cbind(time, x) ~ ts` specifying one or more disease incidence time series in long format. `ts` must evaluate to a factor (insofar as `as.factor(ts)` is a factor) grouping the data by time series. `time` must evaluate to a numeric vector that is increasing within levels of `ts`. `Date`, `POSIXct`, and `POSIXlt` vectors are supported and coerced to numeric with `julian(time)`. Finally, `x` must evaluate to a non-negative numeric vector with `x[i]` equal to the number of cases observed over the interval `(time[i-1], time[i]]`. Edge cases like `x[1]` are ignored internally. Elements of `x` that are not integer-valued are rounded with a warning. `formula_ts = cbind(time, x) ~ 1` can be supplied when there is only one time series; it is equivalent to `formula_ts = cbind(time, x) ~ ts` with `ts` evaluating to `rep(factor(1), length(x))`.
`formula_windows`	a formula of the form `cbind(start, end) ~ ts` specifying disjoint fitting windows `(start, end]` in long format. If `formula_ts = cbind(time, x) ~ ts1` and `formula_windows = cbind(start, end) ~ ts2`, then observation `x[i]` is associated with window `(start[j], end[j]]` if and only if `time[i-1] >= start[j]`, `time[i] <= end[j]`, and `ts1[i] == ts2[j]`.
`formula_parameters`	a list of formulae of the form `parameter ~ terms` specifying mixed effects models for top level nonlinear model parameters using **lme4**-like syntax (see, e.g., `help("lmer", package = "lme4")`). Alternatively, a formula of the form `~terms` to be recycled for all parameters. A list of parameters for which formulae may be specified can be retrieved with `egf_top`. Specifically, `deparse(parameter)` must be an element of `egf_top(model)`. The default for parameters not assigned a formula is `~1`.
`formula_priors`	a list of formulae of the form `parameter ~ prior` defining priors on: (i) top level nonlinear model parameters, (ii) fixed effect coefficients and random effect covariance parameters (elements

of segments beta and theta of the bottom level parameter vector), or
(iii) random effect covariance matrices (elements of a list Sigma containing the matrices).

prior must be a call to a prior function with arguments specifying suitable hyperparameters. In case (i), deparse(parameter) must be an element of egf_top(model), and hyperparameters supplied on the right hand side must have length 1. In cases (ii) and (iii), parameter must be beta, theta, or Sigma or a call to [or [[referring to a subset or element of beta, theta, or Sigma (e.g., beta[index], where index is a valid index vector for beta), and hyperparameters are recycled to the length of the indicated subset.

Expressions prior and index are evaluated in the corresponding formula environment.

data_ts, data_windows
: data frames, lists, or environments to be searched for variables named in the corresponding formula_* and subset_* arguments. (formula_parameters uses data_windows.) Formula environments are searched for variables not found here.

subset_ts, subset_windows
: expressions to be evaluated in the corresponding data_* data frames. The value should be a valid index vector for the rows of the data frame. Rows that are not indexed are discarded. Rows that are indexed are filtered further (e.g., time series with zero associated fitting windows are discarded regardless of subset_ts). The default is to preserve all rows for further filtering.

select_windows
: an expression indicating additional variables in data_windows (if it is a data frame) to be preserved in the returned object for use by methods. The default is to preserve nothing. A dot '.' is to preserve all variables not occurring in formula_windows or formula_parameters. Outside of these two special cases, evaluation of select follows the implementation of subset.data.frame.

na_action_ts
: a character string affecting the handling of NA in x if formula_ts = cbind(time, x) ~ ts. "fail" is to throw an error. "pass" is to ignore NA when fitting and replace NA when predicting. NA in time and ts are always an error.

na_action_windows
: a character string affecting the handling of NA in formula_windows and formula_parameters variables. "fail" is to throw an error. "omit" is to discard incomplete rows of data.

control
: an egf_control object specifying control parameters.

init
: a named list of numeric vectors with possible elements beta, theta, and b, specifying values to be used in the first likelihood evaluation for the so-named segments of the bottom level parameter vector. The default value of each segment is a zero vector, with the exception that "(Intercept)" coefficients in beta have default values computed from supplied time series. Use NA to indicate elements that should retain their default value.

map
: a named list of factors with possible elements beta, theta, and b, each as long as the so-named segment of the bottom level parameter. Elements of a segment <name> indexed by is.na(map[["<name>"]]) are fixed at their initial values,

rather than estimated, and elements corresponding to a common factor level are constrained to have a common value during estimation. `map[["<name>"]]` can be an index vector for segment `<name>`, instead of a factor. In this case, the indexed elements of that segment are fixed at their initial values.

`fit` a logical. If `FALSE`, then `egf` returns early (*before* fitting) with a partial model object. The details of the partial result are subject to change and therefore sparsely documented, on purpose ...

`se` a logical. If `TRUE`, then the Hessian matrix of the negative log marginal likelihood function is computed and inverted to approximate the joint covariance matrix of segments `beta` and `theta` of the bottom level parameter vector. Standard errors on the fitted values of all top level nonlinear model parameters are computed approximately using the delta method. Computations are preserved in the model object for reuse by methods.

`...` additional arguments passed from or to other methods.

Details

Users attempting to set arguments `formula_priors`, `init`, and `map` should know the structure of the bottom level parameter vector. It is described under topic `egf-class`.

If

```
formula_ts = cbind(time, x) ~ ts1
formula_windows = cbind(start, end) ~ ts2
```

then it is expected that `time`, `start`, and `end` (after coercion to numeric) measure time on the same scale. To be precise, numeric times should have a common unit of measure and, at least within time series, represent displacements from a common reference time. These conditions will always hold if `time`, `start`, and `end` all evaluate to `Date`, `POSIXct`, or `POSIXlt` vectors.

When day of week effects are estimated, numeric times are interpreted as numbers of days since midnight on January 1, 1970, so that time points can be mapped unambiguously to days of week. Furthermore, in this case, `time` (after coercion to numeric) is required to be integer-valued with one day spacing in all time series. This means that

```
isTRUE(all.equal(time, round(time))) &&
    all(range(diff(round(time))) == 1)
```

must be `TRUE` in each level of `ts1`. These conditions ensure that intervals between successive time points represent exactly one day of week.

Value

A list inheriting from class `egf`. See topic `egf-class` for class documentation.

See Also

The many methods for class `egf`, listed by `methods(class = "egf")`.

Examples

```
## Simulate 'N' incidence time series exhibiting exponential growth
set.seed(180149L)
N <- 10L
f <- function(time, r, c0) {
    lambda <- diff(exp(log(c0) + r * time))
    c(NA, rpois(lambda, lambda))
}
time <- seq.int(0, 40, 1)
r <- rlnorm(N, -3.2, 0.2)
c0 <- rlnorm(N, 6, 0.2)
data_ts <-
    data.frame(country = gl(N, length(time), labels = LETTERS[1:N]),
               time = rep.int(time, N),
               x = unlist(Map(f, time = list(time), r = r, c0 = c0)))
rm(f, time)

## Define fitting windows (here, two per time series)
data_windows <-
    data.frame(country = gl(N, 1L, 2L * N, labels = LETTERS[1:N]),
               wave = gl(2L, 10L),
               start = c(sample(seq.int(0, 5, 1), N, TRUE),
                         sample(seq.int(20, 25, 1), N, TRUE)),
               end = c(sample(seq.int(15, 20, 1), N, TRUE),
                       sample(seq.int(35, 40, 1), N, TRUE)))

## Estimate the generative model
m1 <-
    egf(model = egf_model(curve = "exponential", family = "pois"),
        formula_ts = cbind(time, x) ~ country,
        formula_windows = cbind(start, end) ~ country,
        formula_parameters = ~(1 | country:wave),
        data_ts = data_ts,
        data_windows = data_windows,
        se = TRUE)

## Re-estimate the generative model with:
## * Gaussian prior on beta[1L]
## * LKJ prior on all random effect covariance matrices
##   (here there happens to be just one)
## * initial value of 'theta' set explicitly
## * theta[3L] fixed at initial value
m2 <-
    update(m1,
           formula_priors = list(beta[1L] ~ Normal(mu = -3, sigma = 1),
                                 Sigma ~ LKJ(eta = 2)),
           init = list(theta = c(log(0.5), log(0.5), 0)),
           map = list(theta = 3L))
```

---

egf-class *Description of Objects of Class* egf

Description

Class `egf` designates models estimated by function `egf`. Objects of this class hold information about an estimated model. Components can be accessed directly. However, as the components are subject to change without notice, portable code will rely on exported methods for interrogation.

Details

Currently, a legitimate `egf` object is a list with elements:

`model` a copy of the so-named argument of `egf`.

`frame` a list of the form `list(ts, windows, parameters, extra)`. `ts` and `windows` are data frames preserving time series and fitting window endpoints. `parameters` is a list of mixed effects model frames, with one element for each top level nonlinear model parameter. `extra` is a data frame preserving additional variables specified in `call[["select_windows"]]`. `windows`, the model frames listed in `parameters`, and `extra` all correspond rowwise.

`priors` a list of the form `list(top, bottom = list(beta, theta, Sigma))`, where `top`, `beta`, `theta`, and `Sigma` are all lists of `egf_prior` objects.

`control` a copy of the so-named argument of `egf`.

`tmb_out` the list output of `MakeADFun`.

`optimizer_out` the list output of the optimizer specified by `control[["optimizer"]]`.

`init, best` numeric vectors giving the values of the condensed bottom level parameter vector used in the first and maximal likelihood evaluations.

`random` a logical vector indexing the elements of the condensed bottom level parameter vector that are *not* arguments of the negative log marginal likelihood function. It indexes all elements of segment `b` (random effect coefficients) and (but only if `control[["profile"]] = TRUE`) all elements of segment `beta` (fixed effect coefficients).

`value, gradient` numeric vectors giving the value and gradient of the negative log marginal likelihood function at `best[!random]`.

`hessian` a logical flag indicating whether the Hessian matrix of the negative log marginal likelihood function is positive definite at `best[!random]`. `NA` means that the matrix has not been computed.

`coefficients` a list of the form `list(fixed, random)`, where `fixed` and `random` are data frames preserving interpretive information about fixed and random effect coefficients.

`contrasts` a list of the form `list(fixed, random)`, where `fixed` and `random` are lists preserving contrasts used to construct the fixed and random effects design matrices.

`call` the call to `egf`, enabling updates to the object by the default method of generic function `update`.

Bottom Level Parameter Vector

An estimated model is specified by a bottom level parameter vector that is the concatenation of three segments:

`beta` the result of `unlist(lbeta)`, where `lbeta` is a list of numeric vectors of fixed effect coefficients, with one vector for each top level nonlinear model parameter. The order of top level parameters is specified by `egf_top(model)`.

`theta` the result of `unlist(ltheta)`, where `ltheta` is a list of numeric vectors of random effect covariance parameters, with one vector for each distinct random effect term in `formula_parameters`. Each vector parametrizes a random effect covariance matrix via `theta2cov` and its inverse `cov2theta`.

The list `Sigma` mentioned in the description of `egf` argument `formula_priors` is precisely `lapply(ltheta, theta2cov)`.

`b` the result of `unlist(lb)`, where `lb` is a list of numeric matrices of scaled random effect coefficients, corresponding elementwise to `ltheta`. The columns of `lb[[i]]` (one per level of the grouping variable) are interpreted as samples from a zero mean, unit variance multivariate normal distribution with covariance matrix `cov2cor(theta2cov(ltheta[[i]]))`.

When elements of this vector are "mapped" via `egf` argument `map`, likelihood is defined as a function of the condensed vector that excludes mapped elements.

Methods are defined for generic functions `coef`, `fixef`, and `ranef` to allow users to interrogate the structure of the vector.

Examples

```
methods(class = "egf")
help.search("\\.egf$", fields = "alias", package = "epigrowthfit")
## less verbosely: alias??`\\.egf$`
```

`egf_control` *Define Control Parameters*

Description

Set parameters controlling the behaviour of `egf`.

Usage

```
egf_control(outer_optimizer = egf_optimizer(nlminb),
            inner_optimizer = egf_optimizer(newton),
            trace = FALSE, profile = FALSE, sparse_X = FALSE,
            omp_num_threads = getOption("egf.cores", 1L))
```

Arguments

`outer_optimizer, inner_optimizer`
: `egf_optimizer` objects specifying "outer" and "inner" optimization methods.

`trace`
: an integer determining the amount of tracing performed; see 'Details'.

`profile`
: a logical. If `TRUE`, then fixed effect coefficients are profiled out of the likelihood, which may stabilize optimization for models with many fixed effects. This "feature" should be considered experimental, and in fact it may *de*stabilize optimization, as it relies on assumptions about the optimization problem that are not necessarily satisfied by the nonlinear mixed effects models fit by `egf`.

`sparse_X` a logical. If `TRUE`, then the fixed effects design matrix is represented as a (sparse) dgCMatrix, rather than as a traditional (dense) matrix.

`omp_num_threads` an integer indicating a number of OpenMP threads to be used when evaluating the objective function, provided that **epigrowthfit** was compiled with OpenMP support.

Details

`trace` affects the amount of information printed during likelihood evaluations:

`0` likelihood evaluations are always silent.

`1` a message is printed whenever a negative log marginal likelihood term is `NaN` or exceeds `1e+09`.

`2` all negative log marginal likelihood terms are printed.

`egf` passes `silent = trace == 0L` to `MakeADFun`. A corollary is that nonzero values of `trace` have a number of additional side effects:

- error messages are printed during function and gradient evaluations;
- the maximum absolute gradient element is printed with each gradient evaluation; and
- trace flags set by `config` are turned on.

Value

A list inheriting from class `egf_control` containing the validated arguments.

Warning

Setting `trace > 0L` and `omp_num_threads > 0L` simultaneously should be avoided, because tracing messages are printed using R API functions that are not thread-safe.

Examples

```
control <- egf_control()
str(control)
```

---

egf_control_plot *Define Control Parameters for Plotting*

Description

Sets parameters controlling the graphical output of `plot` for objects of class `egf`. Supplied values override package defaults (retrievable as `defaults <- egf_control_plot()`), which in turn override global defaults set via `par`.

Below, `x`, `type`, `time_as`, and `delta` refer to the so-named arguments of `plot.egf`.

Usage

```
egf_control_plot(window, data, predict, asymptote, box, axis, title, doubling)
```

Arguments

window	a named list of arguments to rect affecting the appearance of fitting windows.
data	a named list of the form list(main, short, long). main is a named list of arguments to points affecting the appearance of observed data. short and long are alternatives to main used for counts over intervals shorter or longer than delta when type = "interval". short and long default to main (elementwise).
predict	a named list of the form list(value, ci). value and ci are named lists of arguments to lines and polygon affecting the appearance of predicted curves and corresponding confidence bands. ci[["col"]] defaults to value[["col"]] with added transparency.
asymptote	a named list of arguments to segments affecting the appearance of line segments drawn at y = <initial exponential growth rate> when type = "rt" and x[["model"]][["curve"]] = "logistic" or "richards".
box	a named list of arguments to box affecting the appearance of the box drawn around the plot region on the device.
axis	a named list of the form list(x, y). x and y are named lists of arguments to axis affecting the appearance of the bottom and left axes. When time_as = "Date", there are minor and major bottom axes. In this case, the major axis uses a modified version of x that tries to ensure that it is displayed below the minor axis in a slightly larger font.
title	a named list of the form list(main, sub, xlab, ylab). The elements are named lists of arguments to title affecting the appearance of plot (sub)titles and axis labels. sub[["adj"]] defaults to main[["adj"]].
doubling	a named list of the form list(legend, estimate, ci). The elements are named lists of arguments to mtext affecting the appearance of initial doubling times printed in the top margin.

Details

Setting an argument (or an element thereof in the case of nested lists) to NULL has the effect of suppressing the corresponding plot element.

Value

A named list containing the package defaults modified according to the arguments in the call.

egf_has_converged *Test for Convergence*

Description

Performs simple diagnostic tests to assess whether the optimizer that produced an estimated model actually converged to a local minimum point of the negative log marginal likelihood function.

Usage

```
egf_has_converged(object, check = TRUE, tol = 1)
```

Arguments

object	an egf object.
check	a logical. If TRUE, then an error is thrown if object does not actually inherit from class egf.
tol	a positive number. Convergence requires all gradient elements to be less than or equal to tol in absolute value.

Value

TRUE if all tests pass. FALSE if any test fails. NA if no test fails, but the test for a positive definite Hessian matrix is indeterminate because the matrix has not been computed.

egf_has_random *Test for Random Effects*

Description

Tests whether an object specifies a model with random effects.

Usage

```
egf_has_random(object, check = TRUE)
```

Arguments

object	an egf object.
check	a logical. If TRUE, then an error is thrown if object does not actually inherit from class egf.

Value

TRUE or FALSE.

egf_model *Define a Top Level Nonlinear Model*

Description

Sets flags defining a top level nonlinear model of epidemic growth to be estimated by egf.

Usage

```
egf_model(curve = c("logistic", "richards", "exponential",
                    "subexponential", "gompertz"),
          excess = FALSE,
          family = c("nbinom", "pois"),
          day_of_week = FALSE)
```

Arguments

curve	a character string specifying a model for expected cumulative incidence as a function of time.
excess	a logical flag. If TRUE, then a constant baseline mortality rate is estimated.
family	a character string specifying a family of discrete probability distributions assigned to observations, which are the first order differences of observed cumulative incidence.
day_of_week	an integer flag. If positive, then day of week effects are estimated as offsets relative to the indicated day of week (1=Sunday, 2=Monday, and so on).

Value

A list inheriting from class egf_model containing the validated arguments.

See Also

simulate.egf_model.

Examples

```
model <- egf_model()
str(model)
```

egf_optimizer *Define an Optimization Method*

Description

Utilities for linking an optimizer with optional arguments and control parameters to define an optimization method for use by egf.

Usage

```
egf_optimizer(f = nlminb, args = list(), control = list())
```

Arguments

f	a function performing optimization. Supported are newton, optim, nlminb, nlm, and any optim-like function.
args	a list of optional arguments to f not including control. If f = optim and args does not have method as an element, then method = "BFGS" is appended.
control	a list of control parameters to be passed to f.

Details

An optim-like function is a function f such that:

- the first three arguments of f specify an initial parameter vector, an objective function, and a gradient function, respectively;
- f accepts control as a fourth (or later) argument; and
- f returns a named list with elements par, value, convergence, and message.

Value

A list inheriting from class egf_optimizer containing the validated arguments, wherein f may be a new function wrapping the supplied one to make it optim-like.

Examples

```
optimizer <- egf_optimizer(nlminb)
str(optimizer)
```

egf_parallel *Define a Parallelization Method*

Description

Defines instructions for parallelization by linking a method with options.

Usage

```
egf_parallel(method = c("serial", "multicore", "snow"),
             outfile = "", cores = getOption("egf.cores", 1L),
             args = list(), cl = NULL)
```

Arguments

method	a character string indicating a method of parallelization. "serial" indicates no parallelization. "multicore" indicates R level forking. It is intended for use from a terminal rather than from a GUI. "snow" indicates socket clusters. On Windows, "multicore" is equivalent to "serial". "snow" is supported on both Unix-alikes and Windows.
outfile	a character string indicating a file path where console output should be diverted. An empty string indicates no diversion. If method = "snow", then diversion may be necessary to view output.
cores	a positive integer indicating a number of threads/processes to fork/spawn when parallel != "serial". detectCores can be called to detect the theoretical maximum.
args	a list of optional arguments to mclapply (method = "multicore") or makePSOCKcluster (method = "snow").
cl	an existing socket cluster (method = "snow"). The default is to create a new cluster stop it upon job completion.

Value

A list inheriting from class "egf_parallel" containing the arguments (after possible matching and coercion).

See Also

vignette("parallel", "parallel").

Examples

```
parallel <- egf_parallel()
str(parallel)
```

egf_prior *Prior Distributions*

Description

Functions used by egf to specify prior distributions of bottom level mixed effects model parameters.

Usage

```
Normal(mu = 0, sigma = 1)
LKJ(eta = 1)
Wishart(df, scale, tol = 1e-06)
InverseWishart(df, scale, tol = 1e-06)
```

Arguments

mu	a numeric vector listing means.
sigma	a positive numeric vector listing standard deviations.
eta	a positive numeric vector listing values for the shape parameter, with 1 corresponding to a uniform distribution over the space of real, symmetric, positive definite matrices with unit diagonal elements. Lesser (greater) values concentrate the probability density around such matrices whose determinant is nearer to 0 (1).
df	a numeric vector listing degrees of freedom. df must be greater than nrow(scale) - 1. If either df or scale has length greater than 1, then this condition is checked elementwise after recycling.
scale	a list of real, symmetric, positive definite matrices or a matrix to be placed in a list of length 1.
tol	a non-negative number specifying a tolerance for indefiniteness of scale. All eigenvalues of scale must exceed -tol * rho, where rho is the spectral radius of scale. However, regardless of tol, diag(scale) must be positive, as standard deviations are stored on a logarithmic scale.

Value

A list inheriting from class egf_prior, with elements:

family	a character string specifying a family of distributions.
parameters	a named list of numeric vectors specifying parameter values.

Examples

```
Normal(mu = 0, sigma = 1)
Normal(mu = -5:5, sigma = c(0.1, 1))

LKJ(eta = 2)
```

```
u <- matrix(rnorm(9L), 3L, 3L)
utu <- crossprod(u)
uut <- tcrossprod(u)
Wishart(df = 6, scale = utu)
InverseWishart(df = 6, scale = list(utu, uut))
```

`egf_top` *Top Level Nonlinear Model Parameter Names*

Description

Retrieves the names used internally for top level nonlinear model parameters.

Usage

```
egf_top(object, ...)

## S3 method for class 'egf_model'
egf_top(object, link = TRUE, ...)
## S3 method for class 'egf'
egf_top(object, link = TRUE, ...)
```

Arguments

`object`	an R object specifying a top level nonlinear model, typically of class `egf_model` or `egf`.
`link`	a logical flag. If `TRUE`, then `"<link>(<name>)"` is returned instead of `"<name>"`.
`...`	unused optional arguments.

Value

A character vector listing names relevant to `object`.

`epigrowthfit-defunct` *Defunct Functions in Package* **epigrowthfit**

Description

The functions and other objects listed here are no longer part of **epigrowthfit** as they are no longer needed.

Usage

```
## Nothing yet!
```

Details

These either are stubs reporting that they are defunct or have been removed completely (apart from being documented here).

See Also

Deprecated, base-deprecated, epigrowthfit-deprecated, epigrowthfit-notyet.

---

epigrowthfit-deprecated *Deprecated Functions in Package* **epigrowthfit**

Description

The functions and other objects listed here are provided only for compatibility with older versions of **epigrowthfit** and may become defunct as soon as the next release.

Usage

```
## Nothing yet!
```

See Also

Defunct, base-defunct, epigrowthfit-defunct, epigrowthfit-notyet.

---

epigrowthfit-notyet *Not Yet Implemented Functions in Package* **epigrowthfit**

Description

The functions listed here are defined but not yet implemented. Use `bug.report(package = "epigrowthfit")` to request an implementation.

Usage

```
## Nothing yet!
```

See Also

NotYetImplemented, epigrowthfit-deprecated, epigrowthfit-defunct.

extractAIC.egf *Extract the (Generalized) AIC*

Description

Extracts from a model object the generalized Akaike Information Criterion (AIC).

Usage

```
## S3 method for class 'egf'
extractAIC(fit, scale, k = 2, ...)
```

Arguments

`fit`	an `egf` object.
`scale`	unused argument, for generic consistency.
`k`	a number giving a weight for the equivalent degrees of freedom. `k=2` and `k=log(nobs(fit))` give the standard Akaike Information Criterion and Bayesian Information Criterion.
`...`	unused optional arguments.

Value

An numeric vector of length 2 giving the equivalent degrees of freedom and criterion value.

See Also

The generic function `extractAIC`.

finalsize *Compute the Expected Epidemic Final Size*

Description

Computes the proportion of a population expected to be infected over the course of an epidemic, as a function of the basic reproduction number.

Usage

```
finalsize(R0, S0, I0)
```

Arguments

`R0`	a numeric vector listing non-negative values for the basic reproduction number.
`S0, I0`	numeric vectors listing values in the interval $[0, 1]$ for the proportions of the population that are susceptible and infected, respectively, at the start of the epidemic. Hence `S0 + I0` must be less than or equal to 1.

Details

At least one of `S0` and `I0` must be supplied. If `S0` (`I0`) is supplied but not `I0` (`S0`), then the latter is assigned the value of one minus the former.

`R0`, `S0`, and `I0` are recycled to a common length (the maximum of their lengths).

Value

A numeric vector listing values in the interval $[0, 1]$ for the expected epidemic final size.

Computation

The basic reproduction number `R0` defines the expected epidemic final size `Z` through an implicit equation,

```
Z = S0 * (1 - exp(-R0 * (Z + I0))) ,
```

which admits an explicit solution

```
Z = S0 + (1/R0) * W(-R0 * S0 * exp(-R0 * (S0 + I0))) .
```

Here, `W` denotes the Lambert W function. `finalsize` computes this solution, relying on function `lambertW` from package **emdbook**.

References

Ma, J. & Earn, D. J. D. (2006). Generality of the final size formula for an epidemic of a newly invading infectious disease. *Bulletin of Mathetmatical Biology*, *68*(3), 679-702. doi:10.1007/s11538-00590477

See Also

`timescale`, `R0`.

Examples

```
R0 <- 10^seq(-3, 1, length.out = 151L)
plot(R0, finalsize(R0, S0 = 1, I0 = 0), type = "l", las = 1,
     xlab = "basic reproduction number",
     ylab = "final size")
```

`fitted.egf` *Fitted Values*

Description

Retrieves fitted values of top level nonlinear model parameters. The fitted value of a given parameter for a given fitting window is obtained by adding (i) the population fitted value computed as a linear combination of fixed effect coefficients and (ii) all applicable random effects, with random effects set equal to their conditional modes.

Usage

```
## S3 method for class 'egf'
fitted(object,
       top = egf_top(object), subset = NULL, select = NULL,
       class = FALSE, se = FALSE, ...)

## S3 method for class 'fitted.egf'
confint(object, parm = seq_len(nrow(object)), level = 0.95,
        class = FALSE, ...)
```

Arguments

`object`	an `egf` or `fitted.egf` object.
`top`	a subset of `egf_top(object)` naming top level nonlinear model parameters whose fitted values should be retrieved.
`subset, select`	index vectors for the rows and columns of `model.frame(object, "combined")` or language objects evaluating to such vectors. `subset` indicates fitting windows for which fitted values should be retrieved; the default indicates all. `select` indicates variables that should be appended to the result; the default indicates none. Evaluation of language objects follows the implementation of `subset.data.frame`.
`class`	a logical. If `TRUE`, then the value of the method call is a `fitted.egf` or `confint.egf` object, not a matrix.
`se`	a logical. If `TRUE` and if `class = TRUE`, then the result is augmented with approximate delta method standard errors.
`...`	additional arguments passed from or to other methods.
`parm`	a valid index vector for the rows of `object` indicating a subset of the fitted values.
`level`	a number in the interval $(0, 1)$ indicating a confidence level.

Value

A numeric matrix containing fitted values.

Users can pass `class = TRUE` to obtain an augmented result. Thus, alternatively:

A data frame inheriting from class `fitted.egf`, with variables:

top	top level nonlinear model parameter, from egf_top(object).
ts	time series, from levels(model.frame(object)[["ts"]]).
window	fitting window, from levels(model.frame(object)[["window"]]).
value	fitted value.
se	approximate delta method standard error (only if requested).
...	further variables from model.frame(object, "combined") specified by argument select.

See Also

The generic function fitted.

Examples

```
example("egf", package = "epigrowthfit")

zz <- fitted(m1, class = TRUE, se = TRUE)
str(zz)

confint(zz, class = TRUE)
```

---

fixef.egf *Details about Fixed Effect Coefficients*

Description

Extracts from a model object details about the fixed effect coefficients, namely segment beta of the bottom level parameter vector.

Usage

```
## S3 method for class 'egf'
fixef(object, ...)
```

Arguments

object	an egf object.
...	unused optional arguments.

Value

A data frame with one row per coefficient and variables:

bottom	label for a bottom level mixed effects model parameter, in this case for a fixed effect coefficient. This is a string with format "beta[%d]".
top	name of the top level nonlinear model parameter whose fitted value is a function of bottom, from egf_top(object).
term	term from the fixed effects component of the mixed effects model formula for parameter top.
colname	column name in the fixed effects design matrix model.matrix(object, "fixed").
value	coefficient estimate, from segment beta of coef(object, full = TRUE).

See Also

The generic function fixef.

---

formula.egf *Extract Model Formulae*

Description

Extracts from a model object the mixed effects model formula corresponding to a top level nonlinear model parameter.

Usage

```
## S3 method for class 'egf'
formula(x, top = egf_top(x), split = FALSE, ...)
```

Arguments

x	an egf object.
top	a character string specifying a top level nonlinear model parameter.
split	a logical flag. If TRUE, then random effect terms are deleted from the formula and preserved as an attribute.
...	unused optional arguments.

Value

By default, the mixed effects model formula corresponding to top. If split = TRUE, then the same formula without random effect terms. The deleted terms are stored in an expression vector and preserved as attribute random of the result.

See Also

The generic function formula.

getCall.egf *Extract Model Calls*

Description

Extracts from a model object the call to egf that produced it. This method exists mainly to enable compatibility with the default method of generic function update.

Usage

```
## S3 method for class 'egf'
getCall(x, ...)
```

Arguments

x	an egf object.
...	unused optional arguments.

Value

A call to egf.

See Also

The generic function getCall.

gi *Generation Interval Distribution*

Description

Generation interval density function (dgi), distribution function (pgi), quantile function (qgi), and sampling (rgi). Results are conditional on supplied latent and infectious period distributions. It is assumed

- that the latent period and infectious waiting time are independent,
- that infectiousness is constant over the infectious period, and
- that the latent and infectious periods are positive and integer-valued (in arbitrary but common units of time).

Usage

```
dgi(x, latent, infectious)
pgi(q, latent, infectious)
qgi(p, latent, infectious)
rgi(n, latent, infectious)
```

Arguments

x, q	a numeric vector listing generation intervals.
p	a numeric vector listing probabilities.
n	a non-negative integer indicating a sample size. If length(n) > 1, then length(n) is taken to be the sample size.
latent, infectious	numeric vectors such that latent[i] and infectious[i] are the probabilities that the latent and infectious periods, respectively, are i units of time. It is sufficient to supply probability weights, as internally both vectors are divided by their sums.

Value

A numeric vector with length equal to the that of the first argument or length n in the case of rgi.

References

Svensson, Å. (2007). A note on generation times in epidemic models. *Mathematical Biosciences*, *208*(1), 300-311. doi:10.1016/j.mbs.2006.10.010

Examples

```
latent <- c(0.026, 0.104, 0.182, 0.246, 0.318, 0.104,
            0.013, 0.004, 0.003)
m <- length(latent)

infectious <- c(0.138, 0.462, 0.256, 0.078, 0.041, 0.007,
                0.004, 0.004, 0.006, 0.004)
n <- length(infectious)

## Histogram of samples
y <- rgi(1e06, latent, infectious)
hist(y, breaks = seq(0, m + n + 1), freq = FALSE, las = 1,
     ylab = "relative frequency",
     main = "")

## Density and distribution functions
x <- seq(0, m + n + 1, by = 0.02)
fx <- dgi(x, latent, infectious)
Fx <- pgi(x, latent, infectious)
plot(x, fx, type = "l", las = 1, # consistent with histogram
     xlab = "generation interval",
     ylab = "density function")
plot(x, Fx, type = "l", las = 1,
     xlab = "generation interval",
     ylab = "distribution function")

## Quantile function
p <- seq(0, 1, by = 0.001)
qp <- qgi(p, latent, infectious)
```

```
plot(p, qp, type = "l", las = 1,
     xlab = "probability",
     ylab = "quantile function")
```

`logLik.egf` *Extract the Log (Marginal) Likelihood*

Description

Extracts from a model object the value of the log marginal likelihood. Whether the result represents a local maximum depends on, among other things, convergence of the optimizer.

Usage

```
## S3 method for class 'egf'
logLik(object, ...)
```

Arguments

`object`	an `egf` object.
`...`	unused optional arguments.

Value

A numeric vector of length 1 inheriting from class `logLik`. Attribute `df` is the number of estimated parameters (fixed effect coefficients and random effect covariance parameters). Attribute `nobs` is the number of observations of disease incidence used in estimation.

See Also

The generic function `logLik`.

`model.frame.egf` *Extract Model Frames*

Description

Extracts from a model object any of several data frames used to specify the model, including the mixed effects model frames.

Usage

```
## S3 method for class 'egf'
model.frame(formula,
            which = c("ts", "windows", "parameters", "extra", "combined"),
            top = egf_top(formula), full = FALSE, ...)
```

Arguments

formula	an egf object.
which	a character string controlling what is returned:
	"ts" disease incidence time series.
	"windows" fitting window endpoints.
	"parameters" the mixed effects model frame corresponding to top.
	"extra" variables preserved in formula due to setting of egf argument select_windows.
	"combined" the result of concatenating (in the sense of cbind) all mixed effects model frames and the data frame corresponding to "extra", then deleting any duplicated variables.
top	a character string specifying a top level nonlinear model parameter, for which = "parameters".
full	a logical, for which = "ts". If TRUE, then complete time series are returned. Otherwise, only observations belonging to fitting windows are returned.
...	unused optional arguments.

Value

A data frame.

See Also

The generic function model.frame.

---

model.matrix.egf *Extract Design Matrices*

Description

Extracts from a model object fixed and random effects design matrices.

Usage

```
## S3 method for class 'egf'
model.matrix(object, which = c("fixed", "random"),
             top = NULL, random = NULL, ...)
```

Arguments

object	an egf object.
which	a character string controlling what is returned:
	"fixed" the fixed effects design matrix X corresponding to top.
	"random" the random effects design matrix Z corresponding to top and random.

top	a character string specifying a top level nonlinear model parameter. NULL indicates "all of them"; see 'Details'.
random	a random effect term, which is a call to binary operator |. NULL indicates "all of them"; see 'Details'.
...	unused optional arguments.

Details

model.matrix(which = "fixed", top = NULL) returns the result of combining (in the sense of cbind) all fixed effects design matrices.

model.matrix(which = "random", top = "<name>", random = NULL) returns the result of combining all random effects design matrices associated with parameter top.

model.matrix(which = "random", top = NULL, random = NULL) returns the result of combining all random effects design matrices *and* permuting the columns to obtain a convenient ordering of random effect coefficients. (Coefficients are sorted by relation to a common random vector. Random vectors are sorted by relation to a common covariance matrix.)

None of these "combined" design matrices possesses attributes assign and contrasts.

Value

A (sparse) dgCMatrix or a traditional (dense) matrix, with attributes assign and contrasts except in special cases; see 'Details'.

See Also

The generic function model.matrix.

---

nobs.egf *Extract the Number of Observations*

Description

Returns the number of observations of disease incidence that were used in estimation of a model. This number excludes missing values and observations not belonging to a fitting window, which, despite being preserved in model objects, do not affect estimation.

Usage

```
## S3 method for class 'egf'
nobs(object, ...)
```

Arguments

object	an egf object.
...	unused optional arguments.

Value

An integer.

See Also

The generic function nobs.

---

plot.egf *Plot Nonlinear Mixed Effects Models of Epidemic Growth*

Description

A method for printing objects of class egf.

Usage

```
## S3 method for class 'egf'
plot(x, type = c("interval", "cumulative", "rt"),
     time_as = c("Date", "numeric"), delta = 1, log = TRUE, zero = NA,
     show_predict = TRUE, show_doubling = FALSE, level = 0.95,
     control = egf_control_plot(), cache = NULL, plot = TRUE,
     subset = NULL, order = NULL, xlim = NULL, ylim = NULL,
     main = NULL, sub = NULL, xlab = NULL, ylab = NULL, ...)
```

Arguments

x	an egf object.
type	a character string indicating a type of plot. The options are: interval incidence ("interval"), cumulative incidence ("cumulative"), and per capita growth rate ("rt").
time_as	a character string indicating how numeric times are displayed on the bottom axis. The options are: as is ("numeric") and with a calendar ("Date"). In the latter case, horizontal user coordinates on measure time in days since midnight on January 1, 1970.
delta	a positive number indicating a step size on the time axis. Predicted curves are evaluated on a grid with this spacing. When type = "interval", counts observed over shorter or longer intervals delta0 are scaled by a factor of delta/delta0 so that their scale matches that of predicted incidence. Scaled counts can be highlighted via control. If x specifies a model with day of week effects, then delta = 1 is used unconditionally.
log	a logical. If TRUE, then the dependent variable is plotted on a logarithmic scale.
zero	a positive number indicating a line on which to plot zeros when log = TRUE and type = "interval" or "cumulative". NA is to place zeros on the bottom axis. NULL is to suppress zeros.

`show_predict` an integer flag: 2 is to draw predicted curves with confidence bands, 1 is draw predicted curves only, 0 is to draw neither.

`show_doubling` an integer flag: 2 is to print initial doubling time estimates in the top margin with confidence intervals, 1 is to print estimates only, 0 is to print neither. Nothing is printed for models without a well-defined initial exponential growth rate.

`level` a number in the interval $(0, 1)$ indicating confidence level, used when `show_predict = 2` or `show_doubling = 2`.

`control` an `egf_control_plot` object controlling the appearance of most plot elements.

`cache` a `plot.egf` object returned by a previous evaluation of `plot(x, ...)`. Fitted and predicted values and standard errors stored in `cache` are reused to avoid recomputation.

`plot` a logical. If `FALSE`, then plotting does not occur. Useful when only the returned `plot.egf` object is desired.

`subset` an index vector for the rows of `mf = model.frame(object, "combined")` or a language object evaluating to such a vector. Only time series corresponding to indexed rows are plotted and only fitting windows corresponding to indexed rows are highlighted; the default is to plot all series and to highlight all windows. Evaluation of language objects follows the implementation of `subset.data.frame`.

`order` a permutation of `seq_len(nrow(mf))` or a language object evaluating in `mf` to such a vector. `order` indicates the order in which time series are plotted; the default indicates the original order.

`xlim, ylim` numeric vectors of length 2 specifying axis limits, which are recycled for all plots. If `time_as = "Date"`, then `xlim` can instead be a `Date`, `POSIXct`, or `POSIXlt` vector.

`main, sub, xlab, ylab` character or expression vectors or (`main`, `sub`) language objects evaluating in `mf` to such vectors. These are used to generate plot (`main`, `sub`) and axis (`xlab`, `ylab`) labels.

`...` unused optional arguments.

Details

Computation of fitted and predicted values and standard errors is performed before any plots are created. To avoid waste of computation time, cached computations are returned *even if* an error is thrown during plotting. To ensure that the cache is preserved, assign the result of the function call to a name:

```
cache <- plot(x, \dots)
```

. Caching functionality must be used with care, as mismatch between `x` and `cache` will not be detected. Constructions such as `plot(y, cache = plot(x, ...), ...)`, where `x` and `y` are different `egf` objects, should not be expected to produce correct results.

Value

A data frame inheriting from class `plot.egf`.

If argument `cache` was supplied in the function call, then this data frame is the result of augmenting `cache` with any new computations.

See Also

The generic function `plot`.

Examples

```
example("egf", package = "epigrowthfit")

l <- list(legend = list(cex = 0.8),
          value = list(cex = 0.8, font = 2),
          ci = list(cex = 0.8))
control <- egf_control_plot(doubling = l)

op <- par(mar = c(3.5, 5, 5, 1))
plot(m1,
     type = "interval",
     show_predict = 2L,
     show_doubling = 2L,
     control = control)
plot(m1,
     type = "cumulative",
     main = "Fitted exponential model",
     sub = quote(paste("Country", country)))
par(op)

op <- par(mar = c(3.5, 9.5, 5, 1))
plot(m1, type = "rt", subset = quote(country %in% LETTERS[4:6]))
par(op)
```

---

predict.egf *Predicted Values*

Description

Computes predicted values of top level nonlinear model parameters. These are conditional on an estimated nonlinear mixed effects model and, optionally, new data.

Usage

```
## S3 method for class 'egf'
predict(object, newdata = NULL, class = FALSE, se = FALSE, ...)
```

Arguments

object	an egf object.
newdata	a data frame containing variables to replace those in the model frame. The default is to use the model frame as is, and currently that is the only implemented behaviour.
class	a logical. If TRUE, then the value of the method call call is a predict.egf object, not a matrix.
se	a logical. If TRUE and if class = TRUE, then the result is augmented with approximate delta method standard errors.
...	additional arguments passed from or to other methods.

Value

A numeric matrix containing predicted values.

Users can pass class = TRUE to obtain an augmented result. Thus, alternatively:

A data frame inheriting from class predict.egf, with variables:

top	top level nonlinear model parameter, from egf_top(object).
ts	time series, from levels(model.frame(object)[["ts"]]).
window	fitting window, from levels(model.frame(object)[["window"]]).
value	predicted value.
se	approximate delta method standard error (only if requested).

See Also

The generic function predict.

---

print.egf *Printing Model Objects*

Description

A method for printing objects of class egf.

Usage

```
## S3 method for class 'egf'
print(x, width = 0.9 * getOption("width"), indent = 2L, ...)
```

Arguments

x	an egf object.
width	an integer width for header text.
indent	an integer indent for body text.
...	unused optional arguments.

Value

The argument x, unchanged but invisible.

See Also

The generic function print.

---

profile.egf *Univariate Likelihood Profiles*

Description

Computes univariate likelihood profiles of fixed effect coefficients, random effect covariance parameters, and linear combinations thereof, including population fitted values.

Usage

```
## S3 method for class 'egf'
profile(fitted, level = 0.95,
        A = seq_along(par), grid = 12L,
        parallel = egf_parallel(), trace = FALSE,
        top = egf_top(fitted), subset = NULL, select = NULL, ...)

## S3 method for class 'profile.egf'
confint(object, parm = seq_along(object), level = attr(object, "level"),
        class = FALSE, ...)

## S3 method for class 'profile.egf'
plot(x, parm = seq_along(x), level = attr(x, "level"),
     type = c("z", "abs(z)", "z^2"), ...)
```

Arguments

fitted	an egf object.
level	a number in the interval $(0, 1)$ indicating a confidence level. Profiles are computed up to a change in deviance equal to qchisq(level, df = 1).
A	a numeric matrix with 1+p columns, where p = length(coef(fitted)), in which case each row specifies a linear combination of the elements of c(1, coef(fitted)) to be profiled; or a valid index vector for coef(fitted), in which case the indexed elements are profiled; or NULL, in which case population fitted values are profiled.
grid	a positive integer. Step sizes chosen adaptively by tmbprofile will generate approximately this many points on each side of a profile's minimum point.
parallel	an egf_parallel object defining options for R level parallelization.
trace	a logical. If TRUE, then basic tracing messages indicating progress are printed. These may be mixed with optimizer output depending on fitted[["control"]][["trace"]].

top	a subset of egf_top(fitted) naming top level nonlinear model parameters for which profiles on population fitted values should be profiled.
subset, select	index vectors for the rows and columns of model.frame(fitted, "combined") or language objects evaluating to such vectors. subset indicates fitting windows for which profiles should be computed; the default indicates all. select indicates variables that should be appended to the result; the default indicates none. Evaluation of language objects follows the implementation of subset.data.frame.
...	additional arguments passed from or to other methods.
object, x	a profile.egf object.
parm	a valid index vector for object or x indicating a subset of the profiles.
class	a logical. If TRUE and if object was created by profile(A = NULL), then the value of the method call is a confint.egf object, not a matrix.
type	a character string indicating which of z, $\|z\|$, and z^2 is plotted.

Details

Computation of likelihood profiles is typically expensive, requiring estimation of many restricted models. It is parallelized at the C++ level when there is OpenMP support and fitted[["control"]][["omp_num_threads"] is set to an integer greater than 1. When there is no OpenMP support, it can still be parallelized at the R level with appropriate setting of argument parallel.

Value

A list of length nrow(A) inheriting from classes profile.egf and profile. Each element is a data frame specifying a profile, with two variables:

z	a numeric vector containing profile z-statistics. The profile z-statistic is the appropriately signed square root of the change in deviance under the restricted model.
par.vals	a numeric matrix with one column containing values of the linear combination specified by A[i,].

The confidence level level is preserved as an attribute.

See Also

The generic function profile. The more basic "next" method for generic function plot, namely plot.profile.

Examples

```
example("egf", package = "epigrowthfit")

zz <- profile(m1, A = NULL,
              top = "log(r)", subset = quote(country == "A" & wave == 1))
str(zz)

confint(zz, class = TRUE)
```

```
pty <- c("z", "abs(z)", "z^2")
bty <- c("l", "u", "u")
for (i in 1:3)
    plot(zz, type = pty[i], bty = bty[i], las = 1)
```

R0 *Compute the Basic Reproduction Number*

Description

Computes the basic reproduction number as a function of the initial exponential growth rate, conditional on a binned generation interval distribution.

Usage

```
R0(r, breaks, probs)
```

Arguments

r	a non-negative numeric vector listing initial exponential growth rates.
breaks	an increasing numeric vector of length 2 or greater listing break points in the support of the generation interval distribution, in reciprocal units of r.
probs	a numeric vector of length length(breaks)-1. probs[i] is the probability that the generation interval is between breaks[i] and breaks[i+1]. It is sufficient to supply probability weights, as internally the vector is divided by its sum.

Value

A numeric vector listing basic reproduction numbers.

Computation

For an initial exponential growth rate r, the basic reproduction number is computed as

```
r / sum(probs * (exp(-r * breaks[-n]) - exp(-r * breaks[-1L])) / (breaks[-1L] - breaks[-n])) ,
```

where n = length(breaks).

References

Wallinga, J. & Lipsitch M. (2007). How generation intervals shape the relationship between growth rates and reproductive numbers. *Proceedings of the Royal Society B: Biological Sciences*, *274*(1609), 599-604. doi:10.1098/rspb.2006.3754

See Also

timescale, finalsize.

Examples

```
r <- seq(0, 1, 0.02)
breaks <- seq(0, 20, 1)
probs <- diff(pgamma(breaks, shape = 1, scale = 2.5))

plot(r, R0(r, breaks, probs), las = 1,
     xlab = "initial exponential growth rate",
     ylab = "basic reproduction number")
```

---

ranef.egf *Details about Random Effect Coefficients*

Description

Extracts from a model object details about the random effect coefficients, namely segment b of the bottom level parameter vector.

Usage

```
## S3 method for class 'egf'
ranef(object, makeSigma = FALSE, ...)
```

Arguments

object	an egf object.
makeSigma	a logical flag. If TRUE, then random effect covariance matrices are constructed from segment theta of coef(object, full = TRUE) and preserved as an attribute of the result.
...	unused optional arguments.

Value

A data frame with one row per coefficient and variables:

cov	label for a covariance matrix. This is the interaction of term and group, but using levels with format "Sigma[%d]".
vec	label for a random vector. This is the interaction of term, group, and level, but using levels with format "u[%d]".
bottom	label for a bottom level mixed effects model parameter, in this case for a random effect coefficient; this is a string with format "b[%d]".
top	name of the top level nonlinear model parameter whose fitted value is a function of bottom, from egf_top(object).
term, group	term from the random effects component of the mixed effects model formula for parameter top. term and group give the left and right hand sides of the term, which is a call to binary operator |.

level	level of the factor or interaction indicated by group.
colname	column name in the random effects design matrix model.matrix(object, "random").
value	random effect conditional mode (unit variance scale), from segment b of coef(object, full = TRUE).

If makeSigma = TRUE, then the result has attribute Sigma, a list of covariance matrices corresponding to the levels of variable cov.

See Also

The generic function ranef.

---

simulate.egf *Simulation and Parametric Bootstrapping*

Description

Simulates incidence data conditional on a fitted nonlinear mixed effects model of epidemic growth. Optionally re-estimates the model given the simulated data, thus generating samples from the conditional distribution of the bottom level parameter vector.

Usage

```
## S3 method for class 'egf'
simulate(object, nsim = 1, seed = NULL,
         bootstrap = FALSE,
         control = list(), parallel = egf_parallel(),
         trace = FALSE, ...)
```

Arguments

object	an egf object.
nsim	a positive integer indicating a number of replications.
seed	an integer used to set the RNG state before simulation or, otherwise, NULL; see simulate.
bootstrap	a logical. If TRUE, then a bootstrapping step is performed.
control	passed to nlminb.
parallel	an egf_parallel object defining options for R level parallelization.
trace	a logical. If TRUE, then basic tracing messages indicating progress are printed. These may be mixed with optimizer output depending on object[["control"]][["trace"]].
...	additional arguments passed from or to other methods.

Details

Bootstrap optimizations are typically expensive for nontrivial models. They are parallelized at the C++ level when there is OpenMP support and `object[["control"]][["omp_num_threads"]]` is set to an integer greater than 1. When there is no OpenMP support, they can still be parallelized at the R level with appropriate setting of argument `parallel`.

Arguments `control`, `parallel`, and `trace` are unused when `bootstrap = FALSE`.

Value

A list inheriting from class `simulate.egf`, with elements:

`simulation`	a data frame containing simulated incidence data. It has variables `ts`, `window`, `time`, and `X`, where `X` is a numeric matrix with `nsim` columns. It corresponds rowwise to `model.frame(object)`.
`bootstrap`	a numeric matrix with `nsim` columns, each a sample from the conditional distribution of the parameter vector represented by `coef(object)`. `NULL` if the method call did not set `bootstrap = TRUE`.

Attribute `RNGstate` preserves the RNG state prior to simulation, making the result reproducible.

See Also

The generic function `simulate`.

Examples

```
example("egf", package = "epigrowthfit")

zz <- simulate(m2, nsim = 6L, seed = 181952L, bootstrap = TRUE)
str(zz)

matplot(t(zz[["bootstrap"]][!m2[["random"]], ]),
        type = "o", las = 1, xlab = "simulation", ylab = "value")
```

`simulate.egf_model` *Simulating Incidence Time Series*

Description

Simulates equally spaced incidence time series according to a specified nonlinear model. Top level nonlinear model parameters vary between time series according to a fixed intercept model `~ts` or random intercept model `~(1 | ts)`.

Usage

```
## S3 method for class 'egf_model'
simulate(object, nsim = 1, seed = NULL,
         mu, Sigma = NULL, tol = 1e-06, cstart = 0, tend = 100, ...)
```

Arguments

object	an egf_model object specifying a top level nonlinear model to be simulated.
nsim	a positive integer indicating a number of time series.
seed	an integer used to set the RNG state before simulation or, otherwise, NULL; see simulate.
mu	a numeric vector listing means across time series of top level nonlinear model parameters (link scale). It is assumed that elements are ordered as egf_top(object).
Sigma	a real, symmetric positive definite matrix to be used as the covariance matrix corresponding to mu. The default is equivalent to a zero matrix and is handled specially.
tol	a non-negative number indicating a tolerance for indefinite Sigma. Eigenvalues of Sigma must exceed -tol times its spectral radius. diag(Sigma) must be positive regardless of tol, as standard deviations are handled on a logarithmic scale.
cstart	a number indicating a threshold value of cumulative incidence. Left endpoints of suggested fitting windows are those times when cumulative incidence first exceeds this threshold.
tend	a positive number. Simulated time series run from time 0 to time tend with unit spacing. For nonlinear models of expected cumulative incidence with inflections, this argument is ignored and set to tinfl+1, where tinfl is the time of inflection.
...	unused optional arguments.

Value

A list inheriting from class simulate.egf_model, with elements:

model	copy of argument object.
formula_ts	a formula, always cbind(time, x) ~ ts, expressing how simulated time series are stored in data_ts.
formula_windows	a formula, always cbind(start, end) ~ ts, expressing how fitting window endpoints are stored in data_windows.
formula_parameters	a formula specifying the generative model. If Sigma = NULL, then the formula is ~1 if nsim = 1 and ~ts if nsim > 1. Otherwise, it is ~(1 | ts).
data_ts	a data frame with variables ts, time, and x storing nsim simulated time series in long format.
data_windows	a data frame with nsim rows and variables ts, start, and end suggesting fitting window endpoints for each simulated time series. Start times are determined by cstart. End times are always the last time point in the corresponding time series.
init	a named list of the form list(beta, theta) giving the full bottom level parameter vector of the generative model.

Y	a numeric matrix with nsim rows and length(mu) columns listing top level nonlinear model parameter values for each time series. If Sigma = NULL, then the row vectors of are all mu. Otherwise, Y is (conceptually) the result of MASS::mvrnorm(nsim, mu, Sigma, tol).
call	the call to simulate, allowing for updates to the simulate.egf_model object via update.

Attribute RNGstate preserves the RNG state prior to simulation, making the result reproducible.

See Also

The generic function simulate.

Examples

```
r <- 0.04
c0 <- 400
s <- 0.2

mu <- log(c(r, c0))
Sigma <- diag(rep.int(s^2, length(mu)))

zz <- simulate(object = egf_model(curve = "exponential", family = "pois"),
               nsim = 20L,
               seed = 202737L,
               mu = mu,
               Sigma = Sigma,
               cstart = 10)
str(zz)

mm <- egf(zz)
(pp <- cbind(actual = coef(zz), fitted = coef(mm)))
```

---

summary.egf *Model Summaries*

Description

Summarizes fitted values of top level nonlinear model parameters and gathers diagnostic information that can be used to quickly assess convergence of the optimizer.

Usage

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

Arguments

object	an egf object.
...	additional arguments passed from or to other methods.

Value

A list inheriting from class `summary.egf`, with elements:

`fitted`	a numeric matrix. Each column is the result of applying `summary.default` to a numeric vector listing the fitted values of a top level nonlinear model parameters. Fitted values are retrieved by `fitted.egf`.
`convergence`	an integer code returned by the optimizer, with 0 indicating successful convergence within the specified absolute or relative tolerance.
`value`, `gradient`	numeric vectors giving the value and gradient of the negative log marginal likelihood function at the parameter vector returned by the optimizer.
`hessian`	a logical flag indicating whether the Hessian matrix of the negative log marginal likelihood function is positive definite at the parameter vector returned by the optimizer. `NA` means that the matrix was not computed by `egf`, either because `se = TRUE` was not passed in the function call or because an error was thrown during computation.

See Also

The generic function `summary`.

Examples

```
example("egf", package = "epigrowthfit")

zz <- summary(m1)
str(zz)
```

---

terms.egf *Model Terms*

Description

Extracts the `terms` object corresponding to a top level nonlinear model parameter.

Usage

```
## S3 method for class 'egf'
terms(x, top = egf_top(x), ...)
```

Arguments

`x`	an `egf` object.
`top`	a character string specifying a top level nonlinear model parameter.
`...`	unused optional arguments.

Value

A terms object.

See Also

The generic function terms.

`timescale` *Compute the Characteristic Time Scale*

Description

Computes characteristic time scales corresponding to exponential growth rates.

Usage

```
timescale(r, units)
```

Arguments

`r`	a non-negative numeric vector listing exponential growth rates.
`units`	a character string indicating units for the result. If missing, then the result is "unitless".

Value

`1/r`, as a `difftime` if `units` is not missing.

See Also

`R0`, `finalsize`.

Examples

```
r <- 10^(-2:0)
units <- "days"
stopifnot(all.equal(timescale(r), 1 / r),
          all.equal(timescale(r, units), .difftime(1 / r, units)))
```

vcov.egf *Model Covariance Matrices*

Description

Extracts (or, if necessary, computes) the covariance matrix of bottom level parameters `beta` and `theta`, corresponding to the output of `coef(object)`.

Usage

```
## S3 method for class 'egf'
vcov(object, ...)
```

Arguments

`object`	an `egf` object.
`...`	unused optional arguments.

Details

If the resulting matrix is not finite or not positive definite, then the fit specified by `object` should be investigated, as the optimizer that produced the fit may have failed to converge.

Value

A real, symmetric matrix.

See Also

The generic function `vcov`.

Index