Package 'coarseDataTools'

May 8, 2026

Version 0.7.2

Date 2025-04-09

Title Analysis of Coarsely Observed Data

Maintainer Nicholas G. Reich <nick@umass.edu>

License GPL (>= 2)

URL http://nickreich.github.io/coarseDataTools/

Encoding UTF-8

Depends R (>= 3.5.0)

Imports MCMCpack, graphics, stats, methods

Suggests markdown, knitr

Description Functions to analyze coarse data.
Specifically, it contains functions to (1) fit parametric accelerated failure time models to interval-censored survival time data, and (2) estimate the case-fatality ratio in scenarios with under-reporting. This package's development was motivated by applications to infectious disease: in particular, problems with estimating the incubation period and the case fatality ratio of a given disease. Sample data files are included in the package. See Reich et al. (2009) <doi:10.1002/sim.3659>, Reich et al. (2012) <doi:10.1111/j.1541-0420.2011.01709.x>, and Lessler et al. (2009) <doi:10.1016/S1473-3099(09)70069-6>.

Collate 'CFR_estimation.R' 'dic.fit.mcmc.R' 'dic.fit.R' 'get.obs.type.R' 'S4stuff.R' 'sampleSizeSimulation.R' 'coarseDataTools-package.R'

VignetteBuilder knitr

RoxygenNote 7.3.2

NeedsCompilation no

Author Nicholas G. Reich [aut, cre],
Justin Lessler [aut],
Andrew Azman [aut],
Zhian N. Kamvar [ctb],
Hugo Gruson [ctb]

Repository CRAN

Date/Publication 2025-04-11 21:00:02 UTC

Contents

`cd.fit` *An S4 Class that stores a fitted coarse data object*

Description

This is the output from `dic.fit()`, which contains the important bits of information about the model fit and key options used.

Slots

`ests`**:** Matrix of class `"numeric"`. This matrix summarizes the results of fitting the model. Rows correspond to the first parameter , the second parameter and then percentiles specified by the ptiles argument. Columns correspond to the point estimate, the lower and upper bounds on the 95% confidence interval and the standard error of the point estimate. If the maximization does not converge, this matrix is filled with NAs.

`conv`**:** Object of class `"numeric"`. A value of 1 indicates successful convergence; 0 indicates unsuccessful convergence.

`MSG`**:** Object of class `"character"`. The error message returned from `optim()` if the routine fails to converge.

`loglik`**:** Object of class `"numeric"`. Value of the estimated maximum log-likelihood.

samples: Object of class "data.frame". Data frame of bootstrap estimates of parameters (if bootstraps were performed).

data: Object of class "data.frame". Original data used to fit model.

dist: Object of class "character". Failure time distribution fit to data. "L" for log-normal, "G" for gamma, "W" for Weibull, and "E" for Erlang.

inv.hessian: Object of class "matrix". The inverse of the hessian matrix for the likelihood surface at the MLE. Used to determine the standard errors for the percentiles. Note that optimization is done on a transformed scale with all parameters logged for all distributions except the first parameter of the log-normal distribution.

est.method: Object of class "character". Method used for estimation.

ci.method: Object of class "character". Method used for estimation of confidence/credible intervals.

---

cd.fit.mcmc *An S4 Class that stores a MCMC fit coarse data object*

Description

This is the output from dic.fit.mcmc(), which contains the important bits of information about the model fit and key options used.

Slots

ests: Matrix of class "numeric". This matrix summarizes the results of fitting the model. Rows correspond to the first parameter , the second parameter and then percentiles specified by the ptiles argument. Columns correspond to the point estimate, the lower and upper bounds on the 95% credible interval and the standard error of the point estimate.

conv: Object of class "numeric". Not used in with dic.fit.mcmc

MSG: Object of class "character". The error message returned from optim() if the routine fails to converge.

loglik: Object of class "numeric". Not used in with dic.fit.mcmc.

samples: Object of class "data.frame". Data frame of posterior draws of parameters.

data: Object of class "data.frame". Original data used to fit model.

dist: Object of class "character". Failure time distribution fit to data. "L" for log-normal, "G" for gamma, "W" for Weibull, and "E" for Erlang.

inv.hessian: Object of class "matrix". Not used in with dic.fit.mcmc.

est.method: Object of class "character". Method used for estimation.

ci.method: Object of class "character". Method used for estimation of confidence/credible intervals.

dgammaOff1 *Function that calculates dgamma with a offset of 1 (i.e., 1 is equivalent to 0)*

Description

Function that calculates dgamma with a offset of 1 (i.e., 1 is equivalent to 0)

Usage

```
dgammaOff1(x, replace0 = FALSE, ...)
```

Arguments

x	value to calculate dgamma at
replace0	should we replace 0 with epsilon
...	other parameters to dgamma

Value

dgamma offset

dic.fit *censored survival data*

Description

`dic.fit` fits a parametric accelerated failure time model to survival data. It was developed with the application to estimating incubation periods of infectious diseases in mind but is applicable to many general problems. The data can be a mixture of doubly interval-censored, single interval-censored or exact observations from a single univariate distribution. Currently, three distributions are supported: log-normal, gamma, and Weibull. (The Erlang distribution is supported in the `dic.fit.mcmc` function, which implements an MCMC version of this code.) We use a consistent (par1, par2) notation for each distribution, they map in the following manner:

$$Log - normal(meanlog = par1, sdlog = par2)$$

$$Gamma(shape = par1, scale = par2)$$

$$Weibull(shape = par1, scale = par2)$$

Standard errors of parameters can be computed using closed-form asymptotic formulae or using a bootstrap routine for log-normal and gamma models. Currently, bootstrap SEs are the only option for the gamma models, which do not have a closed form for the percentiles. `dic.fit()` calculates asymptotic SEs by default, or whenever the `n.boots` option is set to 0. To compute bootstrap SEs, just set `n.boots` to be greater than zero. `dic.fit.mcmc` also allows for Markov Chain Monte Carlo fitting of these three parametric models and Erlang models as well.

Usage

```
dic.fit(
  dat,
  start.par2 = log(2),
  opt.method = "L-BFGS-B",
  par1.int = log(c(0.5, 13)),
  par2.int = log(c(1.01, log(5))),
  ptiles = c(0.05, 0.95, 0.99),
  dist = "L",
  n.boots = 0,
  ...
)
```

Arguments

dat	a matrix with columns named "EL", "ER", "SL", "SR", corresponding to the left (L) and right (R) endpoints of the windows of possible exposure (E) and symptom onset (S). Also, a "type" column must be specified and have entries with 0, 1, or 2, corresponding to doubly interval-censored, single interval-censored or exact observations, respectively.
start.par2	starting value for 2nd parameter of desired distribution
opt.method	method used by optim
par1.int	the log-scale interval of possible median values (in the same units as the observations in dat). Narrowing this interval can help speed up convergence of the algorithm, but care must be taken so that possible values are not excluded or that the maximization does not return a value at an endpoint of this interval.
par2.int	the log-scale interval of possible dispersion values
ptiles	percentiles of interest
dist	what distribution to use to fit the data. Default "L" for log-normal. "G" for gamma, and "W" for Weibull.
n.boots	number of bootstrap resamples (0 means that asymptotic results are desired)
...	additional options passed to optim

Value

a cd.fit S4 object.

References

Reich NG et al. Statistics in Medicine. Estimating incubation periods with coarse data. 2009. https://pubmed.ncbi.nlm.nih.gov/19598148/

See Also

cd.fit, dic.fit.mcmc

Examples

```
data(fluA.inc.per)
dic.fit(fluA.inc.per, dist="L")
```

`dic.fit.mcmc` *Fits the distribution to the passed-in data using MCMC as implemented in MCMCpack.*

Description

Similar to `dic.fit` but uses MCMC instead of a direct likelihood optimization routine to fit the model. Currently, four distributions are supported: log-normal, gamma, Weibull, and Erlang. See Details for prior specification.

Usage

```
dic.fit.mcmc(
  dat,
  prior.par1 = NULL,
  prior.par2 = NULL,
  init.pars = c(1, 1),
  ptiles = c(0.05, 0.95, 0.99),
  verbose = 1000,
  burnin = 3000,
  n.samples = 5000,
  dist = "L",
  seed = NULL,
  ...
)
```

Arguments

`dat`	the data
`prior.par1`	vector of first prior parameters for each model parameter. If `NULL` then default parameters are used (as described in Details section).
`prior.par2`	vector of second prior parameters for each model parameter. If `NULL` then default parameters are used (as described in Details section).
`init.pars`	the initial parameter values (vector length = 2)
`ptiles`	returned percentiles of the survival survival distribution
`verbose`	how often do you want a print out from MCMCpack on iteration number and M-H acceptance rate
`burnin`	number of burnin samples
`n.samples`	number of samples to draw from the posterior (after the burnin)
`dist`	distribution to be used (L for log-normal,W for weibull, G for Gamma, and E for erlang, off1G for 1 day right shifted gamma)

seed	seed for the random number generator for MCMC
...	additional parameters to MCMCmetrop1R

Details

The following models are used:

$$Log-normalmodel: f(x) = \frac{1}{x * \sigma\sqrt{2 * \pi}} exp\{-\frac{(\log x - \mu)^2}{2 * \sigma^2}\}$$

$$Log-normalDefaultPrior: \mu\ N(0, 1000), log(\sigma)\ N(0, 1000)$$

$$Weibullmodel: f(x) = \frac{\alpha}{\beta}(\frac{x}{\beta})^{\alpha-1} exp\{-(\frac{x}{\beta})^{\alpha}\}$$

$$WeibullDefaultPriorSpecification: log(\alpha)\ N(0, 1000), \beta\ Gamma(0.001, 0.001)$$

$$Gammamodel: f(x) = \frac{1}{\theta^k \Gamma(k)} x^{k-1} exp\{-\frac{x}{\theta}\}$$

$$GammaDefaultPriorSpecification: p(k, \theta) \propto \frac{1}{\theta} * \sqrt{k * TriGamma(k) - 1}$$

(Note: this is Jeffery's Prior when both parameters are unknown), and

$$Trigamma(x) = \frac{\partial}{\partial x^2} ln(\Gamma(x))$$

.)

$$Erlangmodel: f(x) = \frac{1}{\theta^k (k-1)!} x^{k-1} exp\{-\frac{x}{\theta}\}$$

$$ErlangDefaultPriorSpecification: k \sim NBinom(100, 1), log(\theta) \sim N(0, 1000)$$

(Note: parameters in the negative binomial distribution above represent mean and size, respectively)

Value

a cd.fit.mcmc S4 object

`EMforCFR` *A function to estimate the relative case fatality ratio when reporting rates are time-varying and deaths are lagged because of survival time.*

Description

This function implements an EM algorithm to estimate the relative case fatality ratio between two groups when reporting rates are time-varying and deaths are lagged because of survival time.

Usage

```
EMforCFR(assumed.nu, alpha.start.values, full.data, max.iter = 50,
  verb = FALSE, tol = 1e-10, SEM.var = TRUE)
```

Arguments

`assumed.nu`	a vector of probabilities corresponding to the survival distribution, i.e. nu[i]=Pr(surviving i days | fatal case)
`alpha.start.values`	a vector starting values for the reporting rate parameter of the GLM model. This must have length which corresponds to one less than the number of unique integer values of full.dat[,"new.times"].
`full.data`	A matrix of observed data. See description below.
`max.iter`	The maximum number of iterations for the EM algorithm and the accompanying SEM algorithm (if used).
`verb`	An indicator for whether the function should print results as it runs.
`tol`	A tolerance to use to test for convergence of the EM algorithm.
`SEM.var`	If TRUE, the SEM algorithm will be run in addition to the EM algorithm to calculate the variance of the parameter estimates.

Details

The data matrix full.data must have the following columns:

grp a 1 or a 2 indicating which of the two groups, j, the observation is for.

new.times an integer value representing the time, t, of observation.

R the count of recovered cases with onset at time t in group j.

D the count of deaths which occurred at time t in groupo j (note that these deaths did not have disease onset at time t but rather died at time t).

N the total cases at t, j, or the sum of R and D columns.

Value

A list with the following elements

naive.rel.cfr the naive estimate of the relative case fatality ratio

glm.rel.cfr the reporting-rate-adjusted estimate of the relative case fatality ratio

EM.rel.cfr the lag-adjusted estimate of the relative case fatality ratio

EM.re.cfr.var the variance for the log-scale lag-adjusted estimator taken from the final M-step

EM.rel.cfr.var.SEM the Supplemented EM algorithm variance for the log-scale lag-adjusted estimator

EM.rel.cfr.chain a vector of the EM algorithm iterates of the lag-adjusted relative CFR estimates

EMiter the number of iterations needed for the EM algorithm to converge

EMconv indicator for convergence of the EM algorithm. 0 indicates all parameters converged within max.iter iterations. 1 indicates that the estimate of the relative case fatality ratio converged but other did not. 2 indicates that the relative case fatality ratio did not converge.

SEMconv indicator for convergence of SEM algorithm. Same scheme as EMconv.

ests the coefficient estimates for the model

ests.chain a matrix with all of the coefficient estimates, at each EM iteration

DM the DM matrix from the SEM algorithm

DMiter a vector showing how many iterations it took for the variance component to converge in the SEM algorithm

Examples

```
    ## This is code from the CFR vignette provided in the documentation.

data(simulated.outbreak.deaths)
min.cases <- 10
N.1 <- simulated.outbreak.deaths[1:60, "N"]
N.2 <- simulated.outbreak.deaths[61:120, "N"]
first.t <- min(which(N.1 > min.cases & N.2 > min.cases))
last.t <- max(which(N.1 > min.cases & N.2 > min.cases))
idx.for.Estep <- first.t:last.t
new.times <- 1:length(idx.for.Estep)
simulated.outbreak.deaths <- cbind(simulated.outbreak.deaths, new.times = NA)
simulated.outbreak.deaths[c(idx.for.Estep, idx.for.Estep + 60), "new.times"] <- rep(new.times, + 2)
assumed.nu = c(0, 0.3, 0.4, 0.3)
alpha.start <- rep(0, 22)

## caution! this next line may take several minutes (5-10, depanding on
##    the speed of your machine) to run.
## Not run: cfr.ests <- EMforCFR(assumed.nu = assumed.nu,
                              alpha.start.values = alpha.start,
                              full.data = simulated.outbreak.deaths,
                              verb = FALSE,
                              SEM.var = TRUE,
                              max.iter = 500,
                              tol = 1e-05)
## End(Not run)
```

`exp.win.lengths` *Exposure window lengths from an influenza outbreak at a NYC school*

Description

A numeric vector of exposure window lengths taken from a dataset of doubly interval-censored incubation period observations. All observations came from a NYC public school. The outbreak has been described in full in Lessler et al. (see citation below).

Usage

```
data(exp.win.lengths)
```

Format

A numeric vector with 134 positive values. Each value represents an exposure window length from an observation of the incubation period for that individual. The exposure window length is the length of time during which exposure could have occurred. For example, if an individual could have been exposed anytime between 6am on Monday to 6am on Wednesday, her exposure window length would be 2 days.

Source

Lessler J et al. New England Journal of Medicine. Outbreak of 2009 Pandemic Influenza A (H1N1) at a New York City School. 2009. 361(27):2628-2636. doi:10.1056/NEJMoa0906089

Examples

```
data(exp.win.lengths)
summary(exp.win.lengths)
hist(exp.win.lengths)
```

---

fluA.inc.per *Coarse incubation period data for influenza A*

Description

These observations on the incubation period of influenza A come from a variety of sources, and were gathered for a literature review. They report doubly interval-censored, single interval-censored or exact observations for the incubation period.

Usage

```
data(fluA.inc.per)
```

Format

A data frame with 151 observations on the following 7 variables.

`author` the name of the primary author for the source of the observation

`year` the year of the study which is the source of the observation

`EL` the earliest possible time of infection

`ER` the latest possible time of infection

`SL` the earliest possible time of symptom onset

`SR` the latest possible time of symptom onset

`type` an indicator of the type of observation: 0 for doubly interval-censored, 1 for single-interval censored, 2 for exact

Source

Lessler J, Reich NG, Brookmeyer R, Perl TM, Nelson KE, Cummings DAT. (2009) A systematic review of the incubation periods of acute respiratory viral infections. Lancet Infectious Diseases. 9(5):291-300.

Examples

```
data(fluA.inc.per)
head(fluA.inc.per)
```

---

get.obs.type *Tries to guess the observation types (SIC, DIC, or exact).*

Description

Tries to guess the observation types (SIC, DIC, or exact).

Usage

```
get.obs.type(dat)
```

Arguments

dat	a matrix of data, similar to what needs to be passed to dic.fit().

Value

vector of guessed types

---

logLik,cd.fit-method *Get the log-likelihood value of a* cd.fit *or* cd.fit.mcmc *object*

Description

Get the log-likelihood value of a cd.fit or cd.fit.mcmc object

Usage

```
## S4 method for signature 'cd.fit'
logLik(object)
```

Arguments

object	A cd.fit or cd.fit.mcmc object.

Value

log-likelihood value

`loglikhd` *Negative log likelihood for a dataset of interval-censored data, given a distribution and its parameters.*

Description

Negative log likelihood for a dataset of interval-censored data, given a distribution and its parameters.

Usage

```
loglikhd(pars, dat, dist)
```

Arguments

`pars`	vector of the transformed (estimation scale) parameters
`dat`	a dataset, as in `dic.fit`
`dist`	a distribution, as in `dic.fit`

Details

This package uses two versions of each parameter, the estimation scale, or the scale that is used for numerical optimization, and the reporting scale, or the natural scale of the parameters. For all likelihood calculations, this `loglikhd` function expects parameters that are on the estimation scale, i.e. have range $(-\infty, \infty)$. Specifically, this translates into all parameters for all distributions being log-transformed except for the meanlog (i.e. "par1") for the log-normal distribution.

Value

negative log-likelihood for a given dataset, parameters, and distribution.

`mcmc.erlang` *Does a metropolis hastings for the Erlang distribution*

Description

Does a metropolis hastings for the Erlang distribution

Usage

```
mcmc.erlang(
  dat,
  prior.par1,
  prior.par2,
  init.pars,
  verbose,
  burnin,
  n.samples,
  sds = c(1, 1)
)
```

Arguments

dat	the data to fit
prior.par1	mean of priors. A negative binomial (for shape) and a normal for log(scale)
prior.par2	dispersion parameters for priors, dispersion for negative binomial, log scale sd for normal
init.pars	the starting parameters on the reporting scale
verbose	how often to print an update
burnin	how many burnin iterations to do
n.samples	the number of samples to keep and report back
sds	the standard deviations for the proposal distribution

Value

a matrix of n.samples X 2 parameters, on the estimation scale

---

mcmcpack.ll *posterior log likelihood function to pass to MCMCpack sampler*

Description

posterior log likelihood function to pass to MCMCpack sampler

Usage

```
mcmcpack.ll(pars, dat, prior.par1, prior.par2, dist)
```

Arguments

pars	the parameters to calculate the ll at
dat	the date to base it on
prior.par1	first parameter of each prior
prior.par2	second parameter of each prior
dist	the distribution the likelihood is being calculated for

Value

the posterior log likelihood

---

nycH1N1 *Incubation period data from New York City Public Schools, 2009 H1N1 influenza outbreak*

Description

These observations on the incubation period of influenza A come from the investigation of the H1N1 outbreak in NYC schools in the spring of 2009. They report doubly interval-censored observations for the incubation period.

Usage

```
data(nycH1N1)
```

Format

A data frame with 134 observations on the following 5 variables.

`EL` the earliest possible time of infection

`ER` the latest possible time of infection

`SL` the earliest possible time of symptom onset

`SR` the latest possible time of symptom onset

`type` an indicator of the type of observation: 0 for doubly interval-censored, 1 for single-interval censored, 2 for exact. All of these observations are doubly interval-censored.

Source

Lessler J, Reich NG, Cummings DAT and The DOHMH Swine Influenza Investigation Team. Outbreak of 2009 Pandemic Influenza A (H1N1) at a New York City School. New England Journal of Medicine. 2009. 361(27):2628-2636.

Examples

```
data(nycH1N1)
head(nycH1N1)
## Not run: dic.fit(nycH1N1)
```

pgammaOff1 *Function that calculates pgamma with a offset of 1 (i.e., 1 is equivalent to 0)*

Description

Function that calculates pgamma with a offset of 1 (i.e., 1 is equivalent to 0)

Usage

```
pgammaOff1(x, replace0 = FALSE, ...)
```

Arguments

x	value to calculate pgamma at
replace0	should we replace 0 with epsilon
...	other parameters to pgamma

Value

pgamma offset

precision.simulation *Simulate incubation period analyses with coarse data*

Description

These functions simulate coarse incubation period data sets and analyze them. The goal is for these simulations to provide evidence for how much information a given dataset contains about a characteristic of the incubation period distribution.

Usage

```
precision.simulation(
  N,
  med = 2,
  disp = 1.3,
  percentile = 0.5,
  nsim = 100,
  exact.data = FALSE,
  pct.type.A = 0.5,
  exp.win.dat = NULL,
  verb = FALSE
)
```

```
precision.simulation.exact(N, med, disp, percentile, nsim, verb)

precision.simulation.coarse(
  N,
  med,
  disp,
  percentile,
  nsim,
  pct.type.A,
  exp.win.dat,
  verb
)

generate.coarse.data(N, med, disp, pct.type.A, exp.win.dat)
```

Arguments

N	Overall sample size for the datasets to be simulated.
med	Median for the assumed log normal distribution of the incubation periods.
disp	Dispersion for the assumed log normal distribution of the incubation periods.
percentile	Percentile of the incubation period distribution which we want to estimate.
nsim	Number of datasets to analyze in the simulation.
exact.data	Either TRUE/FALSE. Incidates whether the data generated should be coarsened at all. If TRUE, pct.type.A and exp.win.dat are ignored.
pct.type.A	Percent of the N observations that are assumed to be type A data. If N*pct.type.A is not an integer, it will be rounded to the nearest integer.
exp.win.dat	A vector of exposure window lengths. Defaults to the observed window lengths from Lessler et al. (see below).
verb	If TRUE, a message with the system time and iteration number will be printed ten times during the simulation run.

Value

The `precision.simulation` functions return a matrix with four columns and nsim rows. The "ests" column gives the estimated percentiles for the incubation period distribution. The "SE" column gives the standard error for the estimate. The "conv" column is 1 if the doubly interval-censored likelihood maximization converged. Otherwise, it is 0. The "bias" column gives the estimated percentile - true percentile. The `generate.coarse.data` function returns a matrix with data suitable for analysis by the `dic.fit` function.

`simulated.outbreak.deaths` *Simulated case and death reports from a fictional outbreak*

Description

This dataset provides reported counts of cases and deaths occurring at different time points across a simulated outbreak. Details of the data simulation algorithm are provided in the manuscript "Estimating case fatality ratios from infectious disease surveillance data" (Reich et al., under review, available upon request).

Usage

```
data(simulated.outbreak.deaths)
```

Format

`time` time, t, after start of outbreak

`grp` an categorical variable indicating membership in one of two groups of a covariate, j

`R` number of recovered cases reported at the given t and j

`D` number of deaths reported at the given t and j

`N` total number of cases and deaths reported at t and j, or D+R

Source

Reich NG, Lessler J, Cummings DAT, Brookmeyer R. Estimating case fatality ratios from infectious disease surveillance data. Biometrics. 2012. 68(2): 598-606.

Examples

```
data(simulated.outbreak.deaths)
head(simulated.outbreak.deaths)
plot(simulated.outbreak.deaths[simulated.outbreak.deaths[,"grp"]==1,"D"], type="l")
```

Index