Package ‘scoringRules’

October 5, 2020

Type Package

Title Scoring Rules for Parametric and Simulated Distribution Forecasts

Version 1.0.1

Date 2020-10-03

Author Alexander Jordan, Fabian Krueger, Sebastian Lerch

Maintainer Fabian Krueger <Fabian.Krueger83@gmail.com>

Description Dictionary-like reference for computing scoring rules in a wide range of situations. Covers both parametric forecast distributions (such as mixtures of Gaussians) and distributions generated via simulation.

URL https://github.com/FK83/scoringRules

License GPL (>= 2)

Imports Rcpp (>= 0.12.0), methods, MASS, knitr

Depends R (>= 3.00)

Suggests gsl (>= 1.8-3), hypergeo(>= 1.0), rmarkdown, testthat, crch

LinkingTo Rcpp, RcppArmadillo

RoxygenNote 7.1.1

VignetteBuilder knitr

Encoding UTF-8

NeedsCompilation yes

Repository CRAN

Date/Publication 2020-10-05 10:10:02 UTC

R topics documented:

  ar_ms ....................................................... 2
  crps.numeric ............................................. 5
  GDP data ................................................... 8
  logs.numeric .............................................. 9
ar_ms

Usage

ar_ms(
  y,
  nlag = 1,
  beta_switch = FALSE,
  variance_switch = TRUE,
  identification_constraint = "variance",
  n_burn = 5000,
  n_rep = 20000,
  forecast_periods = 5,
  printout = FALSE,
  Hm1_delta = 25,
  mu_delta = 0,
  s_ = 0.3,
  nu_ = 3,
  R = matrix(c(8, 2, 2, 8), nrow = 2)
)

Arguments

y numeric vector (time series to be analyzed).
nlag integer, number of autoregressive lags (defaults to one).
beta_switch, variance_switch logicals, indicating whether there should be Markovian state dependence in regression parameters and residual variance, respectively. Defaults to beta_switch = FALSE, variance_switch = TRUE.

identification_constraint character, indicating how to identify latent states. Possible values: "variance", "mean" and "persistence". Defaults to "variance".
n_burn, n_rep integers, number of MCMC iterations for burn-in and main analysis.
forecast_periods number of future periods for which forecasts are computed.
printout logical, whether to print progress report during MCMC (defaults to FALSE).
Hm1_delta, mu_delta, s_, nu_, R prior parameters as described in KLTG (2020, Appendix E and Table 4).

Details

The default parameters are as set by KLTG (2020, Section 5). The output matrices fcMeans and fcSds can be used to construct the mixture-of-parameters estimator analyzed by KLTG. While many of the model features can be changed as described above, the number of Markov regimes is always fixed at two.

ar_ms is an R/C++ implementation of Matlab code kindly shared by Gianni Amisano via his website (https://sites.google.com/site/gianniamisanowebsite/). See Amisano and Giacomini (2007) who analyze a similar model.
Value

List containing parameter estimates and forecasts, with the following elements:

- `pars`, matrix of posterior draws for parameters (rows are MCMC iterations, columns are parameters)
- `fcMeans` and `fcSds`, matrices of forecast means and standard deviations (rows are MCMC iterations, columns are forecast horizons)
- `probs`, matrix of filtered probabilities for first latent state (rows are MCMC iterations, columns are time periods, excluding the first `nlag` values for initialization).
- `count`, integer, counter for the number of states that were relabeled based on `identification_constraint`.

Author(s)

Fabian Krueger, based on Matlab code by Gianni Amisano (see details section)

References


See Also

`run_casestudy` uses `ar_ms` to replicate the results of KLTG (2020, Section 5).

Examples

```r
## Not run:
# Use GDP data from 2014Q4 edition
data(gdp)
dat <- subset(gdp, vint == "2014Q4")
y <- dat$val[order(dat$dt)]

# Fit model, using the default settings
set.seed(816)
fit <- ar_ms(y)

# Histograms of parameter draws
par(mfrow = c(2, 2))
hist(fit$pars[,1], main = "Intercept (state-invariant)", xlab = "")
hist(fit$pars[,2], main = "AR(1) term (state-invariant)", xlab = "")
hist(1/fit$pars[,3], main = "Residual variance in 1st state", xlab = "")
hist(1/fit$pars[,4], main = "Residual variance in 2nd state", xlab = "")

# By construction, the residual variance is smaller in the 1st than in the 2nd state:
print(mean(1/fit$pars[,3] < 1/fit$pars[,4]))

## End(Not run)
```
Description

Calculate the Continuous Ranked Probability Score (CRPS) given observations and parameters of a family of distributions.

Usage

```r
## S3 method for class 'numeric'
crps(y, family, ...)
```

Arguments

- `y`: vector of realized values.
- `...`: vectors of parameter values; expected input depends on the chosen family. See details below.

Details

Mathematical details are available in Appendix A of the vignette `Evaluating probabilistic forecasts with scoringRules` that accompanies the package.

The parameters supplied to each of the functions are numeric vectors:

1. Distributions defined on the real line:
   - "laplace" or "lapl": location (real-valued location parameter), scale (positive scale parameter); see `crps_lapl`
   - "logistic" or "logis": location (real-valued location parameter), scale (positive scale parameter); see `crps_logis`
   - "normal" or "norm": mean, sd (mean and standard deviation); see `crps_norm`
   - "normal-mixture" or "mixture-normal" or "mixnorm": m (mean parameters), s (standard deviations), w (weights); see `crps_mixnorm`; note: matrix-input for parameters
   - "t": df (degrees of freedom), location (real-valued location parameter), scale (positive scale parameter); see `crps_t`
   - "two-piece-exponential" or "2pexp": location (real-valued location parameter), scale1, scale2 (positive scale parameters); see `crps_2pexp`
   - "two-piece-normal" or "2pnorm": location (real-valued location parameter), scale1, scale2 (positive scale parameters); see `crps_2pnorm`

2. Distributions for non-negative random variables:
   - "exponential" or "exp": rate (positive rate parameter); see `crps_exp`
   - "gamma": shape (positive shape parameter), rate (positive rate parameter), scale (alternative to rate); see `crps_gamma`
3. Distributions with flexible support and/or point masses:
   • "beta": shape1, shape2 (positive shape parameters), lower, upper (lower and upper limits); see crps_beta
   • "uniform" or "unif": min, max (lower and upper limits), lmass, umass (point mass in lower or upper limit); see crps_unif
   • "expM": location (real-valued location parameter), scale (positive scale parameter), mass (point mass in location); see crps_expM
   • "gev": location (real-valued location parameter), scale (positive scale parameter), shape (real-valued shape parameter); see crps_gev
   • "gpd": location (real-valued location parameter), scale (positive scale parameter), shape (real-valued shape parameter), mass (point mass in location); see crps_gpd
   • "tlogis": location (location parameter), scale (scale parameter), lower, upper (lower and upper limits); see crps_tlogis
   • "clogis": location (location parameter), scale (scale parameter), lower, upper (lower and upper limits); see crps_clogis
   • "gtclogis": location (location parameter), scale (scale parameter), lower, upper (lower and upper limits); crps_gtclogis
   • "tnorm": location (location parameter), scale (scale parameter), lower, upper (lower and upper limits); see crps_tnorm
   • "cnorm": location (location parameter), scale (scale parameter), lower, upper (lower and upper limits); see crps_cnorm
   • "gtcnorm": location (location parameter), scale (scale parameter), lower, upper (lower and upper limits); crps_gtcnorm
   • "tt": df (degrees of freedom), location (location parameter), scale (scale parameter), lower, upper (lower and upper limits); see crps_tt
   • "ct": df (degrees of freedom), location (location parameter), scale (scale parameter), lower, upper (lower and upper limits); see crps_ct
   • "gtct": df (degrees of freedom), location (location parameter), scale (scale parameter), lower, upper (lower and upper limits); crps_gtct

4. Distributions of discrete variables:
   • "binom": size (number of trials (zero or more)), prob (probability of success on each trial); see crps_binom
   • "hyper": m (the number of white balls in the urn), n (the number of black balls in the urn), k (the number of balls drawn from the urn); see crps_hyper
   • "negative-binomial" or "nbinom": size (positive dispersion parameter), prob (success probability), mu (mean, alternative to prob); see crps_nbinom
   • "poisson" or "pois": lambda (positive mean); see crps_pois

All numerical arguments should be of the same length. An exception are scalars of length 1, which will be recycled.
Value

Vector of score values. *A lower score indicates a better forecast.*

Author(s)

Alexander Jordan, Fabian Krueger, Sebastian Lerch

References

*Closed form expressions of the CRPS for specific distributions:*


*Independent listing of closed-form solutions for the CRPS:*


See Also

logs.numeric
Examples

crps(y = 1, family = "normal", mean = 0, sd = 2)
crps(y = rnorm(20), family = "normal", mean = 1:20, sd = sqrt(1:20))

## Arguments can have different lengths:
crps(y = rnorm(20), family = "normal", mean = 0, sd = 2)
crps(y = 1, family = "normal", mean = 1:20, sd = sqrt(1:20))

## Mixture of normal distributions requires matrix input for parameters:
mval <- matrix(rnorm(20*50), nrow = 20)
sdval <- matrix(runif(20*50, min = 0, max = 2), nrow = 20)
weights <- matrix(rep(1/50, 20*50), nrow = 20)
crps(y = rnorm(20), family = "mixnorm", m = mval, s = sdval, w = weights)

GDP data

Data and forecasts for US GDP growth

Description

Historical data and forecast distributions for the growth rate of US gross domestic product (GDP). The forecasts are generated from a Bayesian Markov Switching model as described in Section 5 of Krueger et al (2020).

Format

gdp is a data frame which contains the real-time data set used in Section 5 of KLTG (2020), with the following columns:

- dt - date in question (e.g., "2013Q2" for the second quarter of 2013)
- vint - data vintage (i.e., the date at which the realization was recorded); same format as dt
- val - value of the GDP growth rate

gdp_mcmc is a list, whereby each element is a data frame. gdp_mcmc$forecasts contains the simulated forecast distributions. There are 20 columns (corresponding to quarters 2008:Q1 to 2012:Q4) and 5,000 rows (corresponding to simulation draws). gdp_mcmc$actuals contains the actual observations. There are 20 columns (again corresponding to quarterly dates) and a single row.

Details

The realizations in gdp_mcmc$actuals are also contained in gdp, based on the second available vintage for each date. For example, gdp_mcmc$actuals$X2008Q1 is the entry in gdp for which dt == "2008Q1" and vint == "2008Q3".
Source

The GDP growth rate is computed from real-time data provided by the Federal Reserve Bank of Philadelphia, [https://www.phil.frb.org/research-and-data/real-time-center/](https://www.phil.frb.org/research-and-data/real-time-center/) (series code “ROUTPUT”, second-vintage data). The same data also enters the model which is used to generate the forecast distribution. Disclaimer: The provider of the raw data takes no responsibility for the accuracy of the data posted here. Furthermore, the raw data may be revised over time, and the website linked above should be consulted for the official, most recent version.

The model from which the forecast draws are generated is described in Section 5 of KLTG (2020). Forecasts are one quarter ahead (that is, they are based on data until the previous quarter).

References


Examples

```r
## Not run:
# Load data
data(gdp_mcmc)

# Histogram of forecast draws for 2012Q4
fc_draws <- gdp_mcmc$forecasts[, "X2012Q4"]
hist(fc_draws, main = "Forecast draws for 2012:Q4", xlab = "Value")

# Add vertical line at realizing value
rlz <- gdp_mcmc$actuals[, "X2012Q4"]
abline(v = rlz, lwd = 3)

# Compute CRPS for this forecast case
crps_sample(y = rlz, dat = fc_draws)

## End(Not run)
```

logs.numeric  

*Logarithmic Score for Parametric Forecast Distributions*

Description

Calculate the logarithmic score (*LogS*) given observations and parameters of a family of distributions.

Usage

```r
## S3 method for class 'numeric'
logs(y, family, ...)
```
Arguments

- **y**: Vector of realized values.
- Vectors of parameter values; expected input depends on the chosen family. See details below.

Details

The parameters supplied to each of the functions are numeric vectors:

1. Distributions defined on the real line:
   - "laplace" or "lapl": location (real-valued location parameter), scale (positive scale parameter); see `logs_lapl`
   - "logistic" or "logis": location (real-valued location parameter), scale (positive scale parameter); see `logs_logis`
   - "normal" or "norm": mean, sd (mean and standard deviation); see `logs_norm`
   - "normal-mixture" or "mixture-normal" or "mixnorm": m (mean parameters), s (standard deviations), w (weights); see `logs_mixnorm`; note: matrix-input for parameters
   - "t": df (degrees of freedom), location (real-valued location parameter), scale (positive scale parameter); see `logs_t`
   - "two-piece-exponential" or "2pexp": location (real-valued location parameter), scale1, scale2 (positive scale parameters); see `logs_2pexp`
   - "two-piece-normal" or "2pnorm": location (real-valued location parameter), scale1, scale2 (positive scale parameters); see `logs_2pnorm`

2. Distributions for non-negative random variables:
   - "exponential" or "exp": rate (positive rate parameter); see `logs_exp`
   - "gamma": shape (positive shape parameter), rate (positive rate parameter), scale (alternative to rate); see `logs_gamma`
   - "log-laplace" or "llapl": location log (real-valued location parameter), scale log (positive scale parameter); see `logs_llapl`
   - "log-logistic" or "llogis": location log (real-valued location parameter), scale log (positive scale parameter); see `logs_llogis`
   - "log-normal" or "lnorm": location log (real-valued location parameter), scale log (positive scale parameter); see `logs_lnorm`

3. Distributions with flexible support and/or point masses:
   - "beta": shape1, shape2 (positive shape parameters), lower, upper (lower and upper limits); see `logs_beta`
   - "uniform" or "unif": min, max (lower and upper limits); see `logs_unif`
   - "exp2": location (real-valued location parameter), scale (positive scale parameter); see `logs_exp2`
   - "gev": location (real-valued location parameter), scale (positive scale parameter), shape (real-valued shape parameter); see `logs_gev`
   - "gpd": location (real-valued location parameter), scale (positive scale parameter), shape (real-valued shape parameter); see `logs_gpd`
• "tlogis": location (location parameter), scale (scale parameter), lower, upper (lower and upper limits); see logs_tlogis
• "tnorm": location (location parameter), scale (scale parameter), lower, upper (lower and upper limits); see logs_tnorm
• "tt": df (degrees of freedom), location (location parameter), scale (scale parameter), lower, upper (lower and upper limits); see logs_tt

4. Distributions of discrete variables:
• "binom": size (number of trials (zero or more)), prob (probability of success on each trial); see crps_binom
• "hyper": m (the number of white balls in the urn), n (the number of black balls in the urn), k (the number of balls drawn from the urn); see crps_hyper
• "negative-binomial" or "nbinom": size (positive dispersion parameter), prob (success probability), mu (mean, alternative to prob); see logs_nbinom
• "poisson" or "pois": lambda (positive mean); see logs_pois

All numerical arguments should be of the same length. An exception are scalars of length 1, which will be recycled.

Value

Vector of score values. A lower score indicates a better forecast.

Author(s)

Alexander Jordan, Fabian Krueger, Sebastian Lerch

See Also

crps.numeric

Examples

logs(y = 1, family = "normal", mean = 0, sd = 2)
logs(y = rnorm(20), family = "normal", mean = 1:20, sd = sqrt(1:20))

## Arguments can have different lengths:
logs(y = rnorm(20), family = "normal", mean = 0, sd = 2)
logs(y = 1, family = "normal", mean = 1:20, sd = sqrt(1:20))

## Mixture of normal distributions requires matrix input for parameters:
mval <- matrix(rnorm(20*50), nrow = 20)
sdval <- matrix(runif(20*50, min = 0, max = 2), nrow = 20)
weights <- matrix(rep(1/50, 20*50), nrow = 20)
logs(y = rnorm(20), family = "mixnorm", m = mval, s = sdval, w = weights)
plot.casestudy  

---

**plot.casestudy**  

*Plot the output of run_casestudy*

---

**Description**

Plot the output of run_casestudy

**Usage**

```r
## S3 method for class 'casestudy'
plot(x, ...)
```

**Arguments**

- `x`: object of class `casestudy`, generated by `run_casestudy`
- `...`: additional parameters, see details below.

**Details**

The plot is in the same format as Figure 3 in KLTG (2020). Its content (nr of MCMC chains, maximal sample size, etc) depends on the parameters used to generate `run_casestudy`. In terms of additional inputs (`...`), the following are currently implemented:

- `scoring_rule`, the scoring rule for which results are to be plotted, either "crps" or "logs". Defaults to "crps".
- `add_main_title`, logical, whether to add main title to plot. Defaults to `TRUE`.

**Value**

none, used for the effect of drawing a plot.

**Author(s)**

Fabian Krueger

**References**


**See Also**

`run_casestudy` produces the forecast results summarized by `plot.casestudy`
Description

Plot the output of run_mcstudy

Usage

```r
## S3 method for class 'mcstudy'
plot(x, ...)
```

Arguments

- `x` object of class mcstudy, generated by run_mcstudy
- `...` additional parameters, see details below.

Details

The plot is in the same format as Figure 1 or 2 in KLTG (2020), depending on the parameters set when running run_mcstudy. These parameters also determine the plot content (nr of MCMC chains, maximal sample size, etc). In terms of additional inputs (...), the following are currently implemented:

- `scoring_rule`, the scoring rule for which results are to be plotted, either "crps" or "logs". Defaults to "crps".
- `add_main_title`, logical, whether to add main title to plot. Defaults to TRUE.

Value

none, used for the effect of drawing a plot.

Author(s)

Fabian Krueger

References


See Also

run_mcstudy produces the simulation results summarized by plot.mcstudy
**print.casestudy**  
*Simple print method for object of class casestudy*

**Description**
Simple print method for object of class casestudy

**Usage**
```r
## S3 method for class 'casestudy'
print(x, ...)
```

**Arguments**
- `x`: Object of class casestudy, generated via `run_casestudy`
- `...`: Additional specifications (presently not in use)

---

**print.mcstudy**  
*Simple print function for object of class mcstudy*

**Description**
Simple print function for object of class mcstudy

**Usage**
```r
## S3 method for class 'mcstudy'
print(x, ...)
```

**Arguments**
- `x`: Object of class mcstudy, generated via `run_mcstudy`
- `...`: Additional specifications (presently not in use)
run_casestudy

Run the case study in KLTG (2020), or a smaller version thereof

Description

Run the case study in KLTG (2020), or a smaller version thereof

Usage

```r
case_study <- run_casestudy(
  data_df,  # data frame in the same format as the gdp data set in this package.
  burnin_size = 5000,  # length of the burn-in period used for each forecast.
  max_mcmc_sample_size = 5000,  # maximal number of MCMC draws to consider (integer, must equal either 1000, 5000, 10000, 20000 or 40000). Defaults to 5000.
  nr_of_chains = 3,  # number of parallel MCMC for each forecast date (integer, defaults to 3).
  first_vint = "1996Q2",  # first and last data vintage (= time point at which forecasts are made). Default to "1996Q2" and "2014Q3", respectively.
  last_vint = "2014Q3",
  forecast_horizon = 1,  # forecast horizon to be analyzed (integer, defaults to 1).
  random_seed = 816  # seed for random numbers used during the MCMC sampling process. Defaults to 816.
)
```

Arguments

- `data_df` data frame in the same format as the `gdp` data set in this package.
- `burnin_size` length of the burn-in period used for each forecast.
- `max_mcmc_sample_size` maximal number of MCMC draws to consider (integer, must equal either 1000, 5000, 10000, 20000 or 40000). Defaults to 5000.
- `nr_of_chains` number of parallel MCMC for each forecast date (integer, defaults to 3).
- `first_vint`, `last_vint` first and last data vintage (= time point at which forecasts are made). Default to "1996Q2" and "2014Q3", respectively.
- `forecast_horizon` forecast horizon to be analyzed (integer, defaults to 1).
- `random_seed` seed for random numbers used during the MCMC sampling process. Defaults to 816.

Details

The full results in Section 5 of KLTG (2020) are based on the following setup: `burnin_size = 10000, max_mcmc_sample_size = 50000, nr_of_chains = 16, data_df = gdp, first_vint = "1996Q2", last_vint = "2014Q3", and forecast_horizon = 1`. Since running this full configuration is very time consuming, the default setup offers the possibility to run a small-scale study which reproduces the qualitative outcomes of the analysis. Running the small-scale study implied by the defaults of `run_study` as well as the GDP data (`data_df = gdp`) takes about 40 minutes on an Intel i7 processor.
Value

Object of class "casestudy", containing the results of the analysis. This object can be passed to `plot` for plotting, see the documentation for `plot.casestudy`.

Author(s)

Fabian Krueger

References


See Also

`plot.casestudy` produces a summary plot of the results generated by `run_casestudy`. `run_casestudy` uses `ar_ms` to fit a Bayesian Markov Switching model, recursively for several time periods.

Examples

```r
## Not run:
data(gdp)
cs <- run_casestudy(data_df = gdp, last_vint = "1999Q4")
plot(cs)
## End(Not run)
```

---

**run_mcstudy**

*Run the Monte Carlo study by KLTG (2020), or a smaller version thereof*

Description

Run the Monte Carlo study by KLTG (2020), or a smaller version thereof

Usage

```r
run_mcstudy(
  s = 2,
  a = 0.5,
  n = 12,
  nr_iterations = 50,
  zoom = FALSE,
  random_seed = 816
)
```
scores

Arguments

s, a, n parameters characterizing the process from which data are simulated (see Section 4 and Table 4 of KLTG, 2019). Defaults to the values reported in the main text of the paper.

nr_iterations number of Monte Carlo iterations (defaults to 50).

zoom set to TRUE to produce results for a fine grid of small (MCMC) sample sizes, as in Figure 2 of KLTG (2020).

random_seed seed used for running the simulation experiment. Defaults to 816.

Details

The full results in Section 4 of KLTG (2020) are based on \( s = 2, a = 0.5, n = 12 \) and \( nr \_iterations = 1000 \). Producing these results takes about 140 minutes on an Intel i7 processor.

Value

Object of class "mcstudy", containing the results of the analysis. This object can be passed to plot for plotting, see the documentation for plot.mcstudy.

Author(s)

Fabian Krueger

References


See Also

plot.mcstudy produces a summary plot of the results generated by run_mcstudy

scores

Generic Scoring Rule Calculation

Description

Generic functions for calculating the Continuous Ranked Probability Score and the Logarithmic Score of R objects.

scoringRules provides default methods (crps.numeric, logs.numeric) to calculate scores of forecasts that are members of families of parametric distributions.
scores_2pexp

Usage

\texttt{crps(y, \ldots)}

\texttt{logs(y, \ldots)}

Arguments

\texttt{y} \quad \text{an object for which the score is to be calculated}

\texttt{\ldots} \quad \text{further arguments passed to or from other methods}

Details

The mean logarithmic score corresponds to the negative of the log-likelihood \texttt{logLik}.

Value

Returns a vector of scores. One for each forecast-observation pair.

References

General background and further references on scoring rules:

See Also

\texttt{crps.numeric, logs.numeric}

\begin{tabular}{ll}
\texttt{scores_2pexp} & Calculating scores for the two-piece-exponential distribution \\
\end{tabular}

Description

Calculating scores for the two-piece-exponential distribution

Usage

\texttt{crps_2pexp(y, scale1, scale2, location = 0)}

\texttt{logs_2pexp(y, scale1, scale2, location = 0)}

Arguments

\texttt{y} \quad \text{vector of observations.}

\texttt{scale1, scale2} \quad \text{vectors of positive scale parameters.}

\texttt{location} \quad \text{vector of location parameters.}
scores_2pnorm

Value

A vector of score values.

Description

Calculating scores for the two-piece-normal distribution

Usage

crps_2pnorm(y, scale1, scale2, location = 0)
logs_2pnorm(y, scale1, scale2, location = 0)

dss_2pnorm

Arguments

y vector of observations.
scale1, scale2 vectors of positive scale parameters.
location vector of location parameters.

Value

A vector of score values.

scores_beta

Calculating scores for the beta distribution

Description

Calculating scores for the beta distribution

Usage

crps_beta(y, shape1, shape2, lower = 0, upper = 1)
logs_beta(y, shape1, shape2, lower = 0, upper = 1)
dss_beta(y, shape1, shape2, lower = 0, upper = 1)

dss_beta

Arguments

y vector of observations.
shape1, shape2 vectors of positive shape parameters.
lower, upper vectors of lower and upper limits of the distribution. Must be finite.
scores_exp

Value

A vector of score values.

scores_binom

Calculating scores for the binomial distribution

Description

Calculating scores for the binomial distribution

Usage

crps_binom(y, size, prob)

logs_binom(y, size, prob)

Arguments

y vector of observations.
size number of trials (zero or more).
prob probability of success on each trial.

Value

A vector of score values.

scores_exp

Calculating scores for the exponential distribution

Description

Calculating scores (CRPS, LogS, DSS) for the exponential distribution, and the exponential distribution with location-scale transformation and point mass in location.

Usage

crps_exp(y, rate = 1)

crps_expM(y, location = 0, scale = 1, mass = 0)

logs_exp(y, rate = 1)

logs_exp2(y, location = 0, scale = 1)

dss_exp(y, rate = 1)
**scores_gamma**

**Arguments**
- **y**: vector of observations.
- **rate**: vector of rates.
- **location**: vector of location parameters.
- **scale**: vector of positive scale parameters.
- **mass**: vector of point masses in location.

**Value**
A vector of score values.

---

**Description**
Calculating scores for the gamma distribution

**Usage**
- `crps_gamma(y, shape, rate = 1, scale = 1/rate)`
- `logs_gamma(y, shape, rate = 1, scale = 1/rate)`
- `dss_gamma(y, shape, rate = 1, scale = 1/rate)`

**Arguments**
- **y**: vector of observations.
- **shape**: vector of positive shape parameters.
- **rate**: an alternative way to specify the scale.
- **scale**: vector of positive scale parameters.

**Value**
A vector of score values.
**scores_gev**  
*Calculating scores for the generalized extreme value distribution*

**Description**

Calculating scores for the generalized extreme value distribution

**Usage**

```r
crps_gev(y, shape, location = 0, scale = 1)
logs_gev(y, shape, location = 0, scale = 1)
dss_gev(y, shape, location = 0, scale = 1)
```

**Arguments**

- `y`: vector of observations.
- `shape`: vector of positive shape parameters.
- `location`: vector of location parameters.
- `scale`: vector of positive scale parameters.

**Value**

A vector of score values.

---

**scores_gpd**  
*Calculating scores for the generalized Pareto distribution*

**Description**

Calculating scores for the generalized Pareto distribution

**Usage**

```r
crps_gpd(y, shape, location = 0, scale = 1, mass = 0)
logs_gpd(y, shape, location = 0, scale = 1)
dss_gpd(y, shape, location = 0, scale = 1)
```
scores_hyper

Arguments

- \( y \) vector of observations.
- \( \text{shape} \) vector of positive shape parameters.
- \( \text{location} \) vector of location parameters.
- \( \text{scale} \) vector of positive scale parameters.
- \( \text{mass} \) vector of point masses in location.

Value

A vector of score values.

---

Calculating scores for the hypergeometric distribution

Description

Calculating scores for the hypergeometric distribution

Usage

\[
\text{crps_hyper}(y, m, n, k)
\]

\[
\text{logs_hyper}(y, m, n, k)
\]

Arguments

- \( y \) vector of observations / numbers of white balls drawn without replacement from an urn which contains both black and white balls.
- \( m \) the number of white balls in the urn.
- \( n \) the number of black balls in the urn.
- \( k \) the number of balls drawn from the urn, hence must be in \( 0, 1, \ldots, m + n \).

Value

A vector of score values.
**scores_lapl**  
*Calculating scores for the Laplace distribution*

**Description**  
Calculating scores for the Laplace distribution

**Usage**  
```r  
crps_lapl(y, location = 0, scale = 1)  
logs_lapl(y, location = 0, scale = 1)  
dss_lapl(y, location = 0, scale = 1)  
```

**Arguments**
- `y`  
  vector of observations.
- `location`  
  vector of location parameters.
- `scale`  
  vector of positive scale parameters.

**Value**

A vector of score values.

---

**scores_llapl**  
*Calculating scores for the log-Laplace distribution*

**Description**  
Calculating scores for the log-Laplace distribution

**Usage**  
```r  
crps_llapl(y, locationlog, scalelog)  
logs_llapl(y, locationlog, scalelog)  
dss_llapl(y, locationlog, scalelog)  
```

**Arguments**
- `y`  
  vector of observations.
- `locationlog`  
  vector of location parameters on the log scale.
- `scalelog`  
  vector of positive scale parameters on the log scale.
scores_llogis

Value

A vector of score values.

Description

Calculating scores for the log-logistic distribution

Usage

```
crps_llogis(y, locationlog, scalelog)
logs_llogis(y, locationlog, scalelog)
dss_llogis(y, locationlog, scalelog)
```

Arguments

- `y`: vector of observations.
- `locationlog`: vector of location parameters on the log scale.
- `scalelog`: vector of positive scale parameters on the log scale.

Value

A vector of score values.

scores_lnorm

Calculating scores for the log-normal distribution

Description

Calculating scores for the log-normal distribution

Usage

```
crps_lnorm(y, meanlog = 0, sdlog = 1, locationlog = meanlog, scalelog = sdlog)
logs_lnorm(y, meanlog = 0, sdlog = 1, locationlog = meanlog, scalelog = sdlog)
dss_lnorm(y, meanlog = 0, sdlog = 1, locationlog = meanlog, scalelog = sdlog)
```
Arguments

- **y**: vector of observations.
- **meanlog**: an alternative way to specify locationlog.
- **sdlog**: an alternative way to specify scalelog.
- **locationlog**: vector of location parameters on the log scale.
- **scalelog**: vector of positive scale parameters on the log scale.

Value

A vector of score values.

---

**scores_logis**

*Calculating scores for the logistic distribution*

Description

These functions calculate scores (CRPS, logarithmic score) and its gradient and Hessian with respect to the parameters of a location-scale transformed logistic distribution. Furthermore, the censoring transformation and the truncation transformation may be introduced on top of the location-scale transformed logistic distribution.

Usage

```r
## score functions
crps_logis(y, location = 0, scale = 1)
crps_clogis(y, location = 0, scale = 1, lower = -Inf, upper = Inf)
crps_tlogis(y, location = 0, scale = 1, lower = -Inf, upper = Inf)
crps_gtclogis(y, location = 0, scale = 1, lower = -Inf, upper = Inf, lmass = 0, umass = 0)
logs_logis(y, location = 0, scale = 1)
logs_tlogis(y, location = 0, scale = 1, lower = -Inf, upper = Inf)
dss_logis(y, location = 0, scale = 1)

## gradient (location, scale) functions
gradcrps_logis(y, location = 0, scale = 1)
gradcrps_clogis(y, location = 0, scale = 1, lower = -Inf, upper = Inf)
gradcrps_tlogis(y, location = 0, scale = 1, lower = -Inf, upper = Inf)

## Hessian (location, scale) functions
hesscrps_logis(y, location = 0, scale = 1)
```

scores_mixnorm

Arguments

- **y**: vector of observations.
- **location**: vector of location parameters.
- **scale**: vector of scale parameters.
- **lower, upper**: lower and upper truncation/censoring bounds.
- **lmass, umass**: vectors of point masses in lower and upper respectively.

Value

For the score functions: a vector of score values.
For the gradient and Hessian functions: a matrix with column names corresponding to the respective partial derivatives.

---

Calculating scores for a mixture of normal distributions.

Description

Calculating scores for a mixture of normal distributions.

Usage

```r
crps_mixnorm(y, m, s, w = NULL)
crps_mixnorm_int(y, m, s, w = NULL, rel_tol = 1e-06)
logs_mixnorm(y, m, s, w = NULL)
dss_mixnorm(y, m, s, w = NULL)
```

Arguments

- **y**: vector of observations.
- **m**: matrix of mean parameters; rows represent observations, columns represent mixture components.
- **s**: matrix of scale parameters; same structure as m.
- **w**: optional; matrix of non-negative weights; same structure as m.
- **rel_tol**: relative accuracy for numerical integration.

Details

`logs_mixnorm` and `crps_mixnorm` calculate scores via analytical formulas. `crps_mixnorm_int` uses numerical integration for the CRPS; this can be faster if there are many mixture components (i.e., if m, s and w have many columns). See examples below.
Value

A vector of score values.

Examples

```r
# Example 1: 100 observations, 15 mixture components
mval <- matrix(rnorm(100*15), nrow = 100)
sdval <- matrix(rgamma(100*15, shape = 2), nrow = 100)
weights <- matrix(rep(1/15, 100*15), nrow = 100)
y <- rnorm(100)
crps1 <- crps_mixnorm(y = y, m = mval, s = sdval, w = weights)
crps2 <- crps_mixnorm_int(y = y, m = mval, s = sdval, w = weights)

## Not run:
# Example 2: 2 observations, 10000 mixture components
mval <- matrix(rnorm(2*10000), nrow = 2)
sdval <- matrix(rgamma(2*10000, shape = 2), nrow = 2)
weights <- matrix(rep(1/10000, 2*10000), nrow = 2)
y <- rnorm(2)
# With many mixture components, numerical integration is much faster
system.time(crps1 <- crps_mixnorm(y = y, m = mval, s = sdval, w = weights))
system.time(crps2 <- crps_mixnorm_int(y = y, m = mval, s = sdval, w = weights))

## End(Not run)
```

---

**scores_moments**

*Scoring Rules for a Vector of Moments*

**Description**

Calculate scores (DSS, ESS) given observations and moments of the predictive distributions.

**Usage**

```r
dss_moments(y, mean = 0, var = 1)

ess_moments(y, mean = 0, var = 1, skew = 0)
```

**Arguments**

- `y` vector of realized values.
- `mean` vector of mean values.
- `var` vector of variance values.
- `skew` vector of skewness values.
Details

The skewness of a random variable $X$ is the third standardized moment

$$E[(\frac{X - \text{mean}}{\sqrt{\text{var}}})^3].$$

Value

Value of the score. A lower score indicates a better forecast.

Author(s)

Alexander Jordan, Sebastian Lerch

References

Dawid-Sebastiani score:
Dawid, A.P. and P. Sebastiani (1999): 'Coherent dispersion criteria for optimal experimental design’

Error-spread score:

scores_nbinom

Calculating scores for the negative binomial distribution

Description

Calculating scores for the negative binomial distribution

Usage

```r
crps_nbinom(y, size, prob, mu)
logs_nbinom(y, size, prob, mu)
dss_nbinom(y, size, prob, mu)
```

Arguments

- `y`: vector of observations.
- `size`: target for number of successful trials, or dispersion parameter (the shape parameter of the gamma mixing distribution). Must be strictly positive, need not be integer.
- `prob`: probability of success in each trial. $0 < \text{prob} <= 1$.
- `mu`: alternative parametrization via mean: see ‘Details’.
The mean of the negative binomial distribution is given by \( \mu = \text{size} \times (1 - \text{prob}) / \text{prob} \).

A vector of score values.

These functions calculate scores (CRPS, LogS, DSS) and their gradient and Hessian with respect to the parameters of a location-scale transformed normal distribution. Furthermore, the censoring transformation and the truncation transformation may be introduced on top of the location-scale transformed normal distribution.

## score functions
- `crps_norm(y, mean = 0, sd = 1, location = mean, scale = sd)`
- `crps_cnorm(y, location = 0, scale = 1, lower = -Inf, upper = Inf)`
- `crps_tnorm(y, location = 0, scale = 1, lower = -Inf, upper = Inf)`
- `crps_gtcnorm(y, location = 0, scale = 1, lower = -Inf, upper = Inf, lmass = 0, umass = 0)`
- `logs_norm(y, mean = 0, sd = 1, location = mean, scale = sd)`
- `logs_tnorm(y, location = 0, scale = 1, lower = -Inf, upper = Inf)`
- `dss_norm(y, mean = 0, sd = 1, location = mean, scale = sd)`

## gradient (location, scale) functions
- `gradcrps_norm(y, location = 0, scale = 1)`
- `gradcrps_cnorm(y, location = 0, scale = 1, lower = -Inf, upper = Inf)`
- `gradcrps_tnorm(y, location = 0, scale = 1, lower = -Inf, upper = Inf)`

## Hessian (location, scale) functions
- `hesscrps_norm(y, location = 0, scale = 1)`
- `hesscrps_cnorm(y, location = 0, scale = 1, lower = -Inf, upper = Inf)`
- `hesscrps_tnorm(y, location = 0, scale = 1, lower = -Inf, upper = Inf)`

### Arguments
- **y**: vector of observations.
- **mean**: an alternative way to specify location.
- **sd**: an alternative way to specify scale.
- **location**: vector of location parameters.
- **scale**: vector of scale parameters.
- **lower, upper**: lower and upper truncation/censoring bounds.
- **lmass, umass**: vectors of point masses in lower and upper respectively.
scores_pois

**Description**

Calculating scores for the Poisson distribution

**Usage**

```r
crps_pois(y, lambda)
logs_pois(y, lambda)
dss_pois(y, lambda)
```

**Arguments**

- `y`: vector of observations.
- `lambda`: vector of (non-negative) means.

**Value**

A vector of score values.

scores_sample_multiv

**Description**

Compute multivariate scores of the form \( S(y, dat) \), where \( S \) is a proper scoring rule, \( y \) is a \( d \)-dimensional realization vector and \( dat \) is a simulated sample of multivariate forecasts. Available are the energy score and the variogram score of order \( p \).

**Usage**

```r
es_sample(y, dat)
vs_sample(y, dat, w = NULL, p = 0.5)
```
Arguments

- **y**: realized values (numeric vector of length d).
- **dat**: numeric matrix of data (columns are simulation draws from multivariate forecast distribution).
- **w**: numeric matrix of weights for dat used in the variogram score. If no weights are specified, constant weights with \( w = 1 \) are used.
- **p**: order of variogram score. Standard choices include \( p = 1 \) and \( p = 0.5 \).

Details

In the input matrix dat each column is expected to represent a sample from the multivariate forecast distribution, the number of rows of dat thus has to match the length of the observation vector y, and the number of columns of dat is the number of simulated samples.

In vs_sample it is possible to specify a matrix \( w \) of non-negative weights that allow to emphasize or downweight pairs of component combinations based on subjective expert decisions. \( w \) is a square matrix with dimensions equal to the length of the input vector y, and the entry in the \( i \)-th row and \( j \)-th column of \( w \) corresponds to the weight assigned to the combination of the corresponding \( i \)-th and \( j \)-th component. For details and examples, see Scheuerer and Hamill (2015).

Value

Value of the score. A lower score indicates a better forecast.

Author(s)

Maximiliane Graeter, Sebastian Lerch, Fabian Krueger

References

- **Energy score**

- **Variogram-based proper scoring rules**

Examples

```r
  d <- 10  # number of dimensions
  m <- 50  # number of samples from multivariate forecast distribution

  mu0 <- rep(0, d)
  mu <- rep(1, d)
  Sigma0 <- S <- diag(d)
  Sigma[S==0] <- 0.1
```
scores_sample_univ

S0[S0==0] <- 0.2

# generate samples from multivariate normal distributions
obs <- drop(mu0 + rnorm(d) %*% chol(S0))
fc_sample <- replicate(m, drop(mu + rnorm(d) %*% chol(S)))

es_sample(y = obs, dat = fc_sample)

# weighting matrix for variogram score
w_vs <- matrix(NA, nrow = d, ncol = d)
for(d1 in 1:d){for(d2 in 1:d){w_vs[d1,d2] <- 0.5^abs(d1-d2)}}

vs_sample(y = obs, dat = fc_sample)
vs_sample(y = obs, dat = fc_sample, w = w_vs)
vs_sample(y = obs, dat = fc_sample, w = w_vs, p = 1)

scores_sample_univ

Scoring Rules for Simulated Forecast Distributions

Description

Calculate scores (CRPS, LogS, DSS) given observations and draws from the predictive distributions.

Usage

crps_sample(
  y,
  dat,
  method = "edf",
  w = NULL,
  bw = NULL,
  num_int = FALSE,
  show_messages = TRUE
)

logs_sample(y, dat, bw = NULL, show_messages = FALSE)

dss_sample(y, dat, w = NULL)

Arguments

y vector of realized values.
dat vector or matrix (depending on y; see details) of simulation draws from forecast distribution.
method string; approximation method. Options: "edf" (empirical distribution function) and "kde" (kernel density estimation).
scores_sample_univ

w optional; vector or matrix (matching dat) of weights for method "edf".
bw optional; vector (matching y) of bandwidths for kernel density estimation; see details.
num_int logical; if TRUE numerical integration is used for method "kde".
show_messages logical; display of messages (does not affect warnings and errors).

Details

For a vector y of length n, dat should be given as a matrix with n rows. If y has length 1, then dat may be a vector.

crps_sample employs an empirical version of the quantile decomposition of the CRPS (Laio and Tamea, 2007) when using method = "edf". For method = "kde", it uses kernel density estimation using a Gaussian kernel. The logarithmic score always uses kernel density estimation.

The bandwidth (bw) for kernel density estimation can be specified manually, in which case it must be a positive number. If bw == NULL, the bandwidth is selected using the core function bw.nrd. Numerical integration may speed up computation for crps_sample in case of large samples dat.

Value

Value of the score. A lower score indicates a better forecast.

Author(s)

Alexander Jordan, Fabian Krüeger, Sebastian Lerch

References

Evaluating simulation based forecast distributions:


Empirical quantile decomposition of the CRPS:


Examples

## Not run:

# y has length greater than 1
y <- 1:2
sample <- matrix(rnorm(20), nrow = 2)
crps_sample(y = y, dat = sample)
logs_sample(y = y, dat = sample)

y <- 1:2
sample <- rnorm(10)
scores_t

Calculating scores for Student's $t$-distribution

Description

These functions calculate scores (CRPS, logarithmic score) and their gradient and Hessian with respect to the parameters of a location-scale transformed Student's $t$-distribution. Furthermore, the censoring transformation and the truncation transformation may be introduced on top of the location-scale transformed normal distribution.

Usage

## score functions

```r
crps_t(y, df, location = 0, scale = 1)
crps_ct(y, df, location = 0, scale = 1, lower = -Inf, upper = Inf)
crps_tt(y, df, location = 0, scale = 1, lower = -Inf, upper = Inf)
crps_gtc(y, df, location = 0, scale = 1, lower = -Inf, upper = Inf, lmass = 0, umass = 0)
logs_t(y, df, location = 0, scale = 1)
logs_tt(y, df, location = 0, scale = 1, lower = -Inf, upper = Inf)
dss_t(y, df, location = 0, scale = 1)
```

## gradient (location, scale) functions

```r
gradcrps_t(y, df, location = 0, scale = 1)
gradcrps_ct(y, df, location = 0, scale = 1, lower = -Inf, upper = Inf)
gradcrps_tt(y, df, location = 0, scale = 1, lower = -Inf, upper = Inf)
```

## Hessian (location, scale) functions

```r
hesscrps_t(y, df, location = 0, scale = 1)
hesscrps_ct(y, df, location = 0, scale = 1, lower = -Inf, upper = Inf)
hesscrps_tt(y, df, location = 0, scale = 1, lower = -Inf, upper = Inf)
```
Arguments

- `y` vector of observations.
- `df` vector of degrees of freedom.
- `location` vector of location parameters.
- `scale` vector of scale parameters.
- `lower, upper` lower and upper truncation/censoring bounds.
- `lmass, umass` vectors of point masses in `lower` and `upper` respectively.

Value

For the CRPS functions: a vector of score values.

For the gradient and Hessian functions: a matrix with column names corresponding to the respective partial derivatives.

scores_unif Calculating scores for the uniform distribution

Description

Calculating scores for the uniform distribution

Usage

```r
scores_unif(y, min = 0, max = 1, lmass = 0, umass = 0)
```

```r
logs_unif(y, min = 0, max = 1)
```

```r
dss_unif(y, min = 0, max = 1)
```

Arguments

- `y` vector of observations.
- `min, max` lower and upper limits of the distribution. Must be finite.
- `lmass, umass` vectors of point masses in `min` and `max` respectively.

Value

A vector of score values.
**Summary.casestudy**  
Summary method for class casestudy

**Usage**

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

**Arguments**

- `object`: Object of class casestudy, generated via `run_casestudy`
- `...`: Additional specifications (presently not in use)

---

**Summary.mcstudy**  
Simple summary method for class mcstudy

**Usage**

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

**Arguments**

- `object`: Object of class mcstudy, generated via `run_mcstudy`
- `...`: Additional specifications (presently not in use)
**Supplementary distributions: Positive real line**

*Supplementary distributions (not in base R) supported on the positive real line.*

---

**Description**

We include the probability density functions of some distributions which are part of scoringRules, but are not part of base R. The parametrizations used here are identical to the ones used when calling `crps` and `logs`.

Here we document distributions on the positive real line: `fllapl` - log-Laplace distribution; `fllogis` - log-logistic distribution.

**Usage**

```
fllapl(x, locationlog, scalelog)
fllogis(x, locationlog, scalelog)
```

**Arguments**

- `x` vector of quantiles
- `locationlog` vector of location parameters on the log scale
- `scalelog` vector of scale parameters on the log scale

**Details**

To be added.

**Value**

Probability density function of the relevant distribution, evaluated at `x`.

**Author(s)**

Alexander Jordan

**See Also**

The documentation for `crps.numeric` contains the full list of distributions supported by scoringRules (includes the ones documented here, as well as many others).
Description

We include the probability density functions of some distributions which are part of scoringRules, but are not part of base R. The parametrizations used here are identical to the ones used when calling crps and logs.

Here we document distributions with support on the real line: flapl - Laplace distribution; f2pexp - two-piece exponential distribution; fmixnorm - mixture of normal distributions; f2pnorm - two-piece normal distribution.

Usage

flapl(x, location, scale)

f2pexp(x, location, scale1, scale2)
f2pnorm(x, location, scale1, scale2)

fmixnorm(x, m, s, w)

Arguments

x vector of quantiles
location vector of location parameters
scale, scale1, scale2 vector of scale parameters
m matrix of means (rows correspond to observations, columns correspond to mixture components)
s matrix of standard deviations (same structure as m)
w matrix of weights (same structure as m)

Details

The Laplace distribution (flapl) is described on https://en.wikipedia.org/wiki/Laplace_distribution. It is a special case of the two-piece exponential distribution (f2pexp), which allows for different scale parameters to the left and right of location.

The density function of a mixture of normal distributions (fmixnorm) is given by the weighted sum over the mixture components,

\[ f(x) = \sum w_i \phi((x - m_i)/s_i), \]

where \( \phi \) is the pdf of the standard normal distribution.

Value

Probability density function of the relevant distribution, evaluated at \( x \).

Author(s)

Alexander Jordan

See Also

The documentation for `crps.numeric` contains the full list of distributions supported by scoringRules (includes the ones documented here, as well as many others).

`fnorm`, `flogis`, `ft`

Examples

```r
# Plot PDF of Laplace distribution
ff <- function(x) flapl(x, location = 0, scale = 2)
curve(ff, from = -8, to = 8, bty = "n", xlab = "Value",
      ylab = "PDF",
      main = "Laplace distribution with location 0 and scale 2")
```

Supplementary distributions: Variable support

Supplementary distributions (not in base R) with variable support.

Description

We include the probability density functions of some distributions which are part of scoringRules, but are not part of base R. The parametrizations used here are identical to the ones used when calling `crps` and `logs`.

Here we document distributions with variable support: `fexp` - location-scale exponential distribution with a point mass on the lower boundary; `fgpd` - generalized Pareto distribution with a point mass on the lower boundary; `fgev` - generalized extreme value distribution; `fnorm`, `flogis`, `ft` - (normal/logistic/Student’s t)-distribution with flexible domain and point masses on the boundaries.

Usage

```r
fexp(x, location, scale, mass = 0, log = FALSE)
fexp(x, location, scale, mass = 0, log = FALSE)

fgpd(x, location, scale, shape, mass = 0, log = FALSE)

fgev(x, location, scale, shape)

fnorm(x, location, scale, lower = -Inf, upper = Inf, lmass = 0, umass = 0, log = FALSE)

ft(x, df, location, scale, lower = -Inf, upper = Inf, lmass = 0, umass = 0, log = FALSE)

flogis(x, location, scale, lower = -Inf, upper = Inf, lmass = 0, umass = 0, log = FALSE)
```
Supplementary distributions: Variable support

Arguments

- **x**: vector of quantiles
- **df**: vector of degrees of freedom parameters
- **location**: vector of location parameters
- **scale**: vector of scale parameters (positive)
- **shape**: vector of shape parameters
- **mass**: vector of point masses in `location`
- **lower**: vector of lower bounds
- **upper**: vector of upper bounds
- **lmass**: vector of point masses in `lower`, or strings "trunc" / "cens"
- **umass**: vector of point masses in `upper`, or strings "trunc" / "cens"
- **log**: logical; if TRUE, the log of the density is returned

Details


Sometimes truncated or censored versions of the normal distribution are used to model variables with a restricted domain (e.g., precipitation). We allow the flexible specification of lower and upper boundaries and point masses in those boundaries. The truncated normal distribution assumes no point masses (i.e. redistributes the cut-off) and can be specified using the string "trunc" instead of a numerical probability. In contrast, the censored distribution introduces a point mass at the bound in the amount of the cut-off. Here, the string "cens" may be used for `lmass` or `umass`. The most common use in practice lies in the context of non-negative quantities. For example, a truncated standard normal distribution (left truncation at zero) has pdf $f(x) = \phi(x)/(1 - \Phi(0))$, for $x \geq 0$ and 0 otherwise. A censored standard normal distribution (left censoring at zero) has point mass $\Phi(0)$ at zero, and density $\phi(x)$ for $x > 0$.

The location-scale family based on Student’s t-distribution (ft) has mean `location` for df > 1 and variance $df/(df-2) \times scale^2$ for df > 2. Note that the crps exists only for df > 1. For details, see https://en.wikipedia.org/wiki/Student’s_t-distribution#Non-standardized_Student.27s_t-distribution.

Value

Density function of the relevant distribution, evaluated at x. NOTE: For distributions involving a point mass (e.g., when `lmass` = "cens" in `fnorm`), the density functions do not integrate to one.

Author(s)

Alexander Jordan
See Also

The documentation for `crps.numeric` contains the full list of distributions supported by `scoringRules` (includes the ones documented here, as well as many others).
Index

* datasets
  GDP data, 8

* distributions
  Supplementary distributions: Positive real line, 38
  Supplementary distributions: Real line, 39
  Supplementary distributions: Variable support, 40

* replication
  plot.casestudy, 12
  plot.mcstudy, 13
  run_casestudy, 15
  run_mcstudy, 16

ar_ms, 2, 3, 4, 16

bw.nrd, 34

crps, 41
  crps(scores), 17
  crps.numeric, 5, 11, 17, 18, 38, 40, 42
  crps_2pexp, 5
  crps_2pexp(scores_2pexp), 18
  crps_2pnorm, 5
  crps_2pnorm(scores_2pnorm), 19
  crps_beta, 6
  crps_beta(scores_beta), 19
  crps_binom, 6, 11
  crps_binom(scores_binom), 20
  crps_clogis, 6
  crps_clogis(scores_logis), 26
  crps_cnorm, 6
  crps_cnorm(scores_norm), 30
  crps_ct, 6
  crps_ct(scores_t), 35
  crps_exp, 5
  crps_exp(scores_exp), 20
  crps_expM, 6
  crps_expM(scores_exp), 20
  crps_hyper, 6, 11
  crps_hyper(scores_hyper), 23
  crps_lapl, 5
  crps_lapl(scores_lapl), 24
  crps_llapl, 6
  crps_llapl(scores_llapl), 24
  crps_llogis, 6
  crps_llogis(scores_llogis), 25
  crps_lnorm, 6
  crps_lnorm(scores_lnorm), 30
  crps_logis, 5
  crps_logis(scores_logis), 26
  crps_mixnorm, 5
  crps_mixnorm(scores_mixnorm), 27
  crps_mixnorm_int(scores_mixnorm), 27
  crps_nbinom, 6
  crps_nbinom(scores_nbinom), 29
  crps_pois, 6
  crps_pois(scores_pois), 31
  crps_sample, 34
  crps_sample(scores_sample_univ), 33
  crps_t, 5
  crps_t(scores_t), 35
  crps_tlogis, 6
  crps_tlogis(scores_logis), 26
  crps_gamma, 5
  crps_gamma(scores_gamma), 21
  crps gev, 6
  crps gev(scores gev), 22
  crps_gpd, 6
  crps_gpd(scores gpd), 22
  crps_gtclogis, 6
  crps_gtclogis(scores_logis), 26
  crps_gtcnorm, 6
  crps_gtcnorm(scores norm), 30
  crps_gtct, 6
  crps_gtct(scores_t), 35
  crps_hyper(scores_hyper), 23
  crps_lapl, 5
  crps_lapl(scores_lapl), 24
  crps_llapl, 6
  crps_llapl(scores_llapl), 24
  crps_llogis, 6
  crps_llogis(scores_llogis), 25
  crps_lnorm, 6
  crps_lnorm(scores_lnorm), 30
  crps_logis, 5
  crps_logis(scores_logis), 26
  crps_mixnorm, 5
  crps_mixnorm(scores_mixnorm), 27
  crps_mixnorm_int(scores_mixnorm), 27
  crps_nbinom, 6
  crps_nbinom(scores_nbinom), 29
  crps_pois, 6
  crps_pois(scores_pois), 31
  crps_sample, 34
  crps_sample(scores_sample_univ), 33
  crps_t, 5
  crps_t(scores_t), 35
  crps_tlogis, 6
  crps_tlogis(scores_logis), 26
crps_tnorm, 6
crps_tnorm(scores_norm), 30
crps_tt, 6
crps_tt(scores_t), 35
crps_unif, 6
crps_unif(scores_unif), 36
dss_beta(scores_beta), 19
dss_exp(scores_exp), 20
dss_gamma(scores_gamma), 21
dss gev(scores gev), 22
dss_gpd(scores_gpd), 22
dss_lapl(scores_lapl), 24
dss_lnorm(scores_lnorm), 24
dss_llogis(scores_llogis), 25
dss_lnorm(scores_lnorm), 25
dss_logis(scores_logis), 26
dss_mixnorm(scores_mixnorm), 27
dss_moments(scores_moments), 28
dss_nbinom(scores_nbinom), 29
dss_norm(scores_norm), 30
dss_pois(scores_pois), 31
dss_sample(scores_sample_univ), 33
dss_t(scores_t), 35
dss_unif(scores_unif), 36
es_sample(scores_sample_multiv), 31
es_months(scores_months), 28

f2pexp

( Supplementary distributions: 
Real line), 39
f2pnorm( Supplementary distributions: 
Real line), 39
fexp( Supplementary distributions: 
Variable support), 40
fgev( Supplementary distributions: 
Variable support), 40
fgpd( Supplementary distributions: 
Variable support), 40
flapl( Supplementary distributions: 
Real line), 39
flapl( Supplementary distributions: 
Positive real line), 38
flogis( Supplementary distributions: 
Positive real line), 38
flogis, 40
flogis( Supplementary distributions: 
Variable support), 40

fmean

(g GDP data, 8
GDP data, 8
gdp_mcmc( GDP data), 8
gradcrps_logis(scores_logis), 26
gradcrps_cnorm(scores_norm), 30
gradcrps_ct(scores_t), 35
gradcrps_logis(scores_logis), 26
gradcrps_norm(scores_norm), 30
gradcrps_t(scores_t), 35
gradcrps_tlogis(scores_logis), 26
gradcrps_tnorm(scores_norm), 30
gradcrps_tt(scores_t), 35

hesscrps_logis(scores_logis), 26
hesscrps_cnorm(scores_norm), 30
hesscrps_ct(scores_t), 35
hesscrps_logis(scores_logis), 26
hesscrps_norm(scores_norm), 30
hesscrps_t(scores_t), 35
hesscrps_tlogis(scores_logis), 26
hesscrps_tnorm(scores_norm), 30
hesscrps_tt(scores_t), 35

loglik, 18
logs(scores), 17
logs.numeric, 7, 9, 17, 18
logs_2pexp, 10
logs_2pexp(scores_2pexp), 18
logs_2pnorm, 10
logs_2pnorm(scores_2pnorm), 19
logs_beta, 10
logs_beta(scores_beta), 19
logs_binom(scores_binom), 20
logs_exp, 10
logs_exp(scores_exp), 20
logs_exp2, 10
logs_exp2(scores_exp), 20
logs_gamma, 10
logs_gamma(scores_gamma), 21
logs_gev, 10
logs_gev(scores_gev), 22
logs_gpd, 10
logs_gpd(scores_gpd), 22
logs_hyper(scores_hyper), 23
logs_lapl, 10
logs_lapl(scores_lapl), 24
logs_llapl, 10
logs_llapl(scores_llapl), 24
logs_llogis, 10
logs_llogis(scores_llogis), 25
logs_lnorm, 10
logs_lnorm(scores_lnorm), 25
logs_logis, 10
logs_logis(scores_logis), 26
logs_mixnorm, 10
logs_mixnorm(scores_mixnorm), 27
logs_nbinom, 11
logs_nbinom(scores_nbinom), 29
logs_norm, 10
logs_norm(scores_norm), 30
logs_pois, 11
logs_pois(scores_pois), 31
logs_sample(scores_sample_univ), 33
logs_t, 10
logs_t(scores_t), 35
logs_tlogis, 11
logs_tlogis(scores_tlogis), 26
logs_tnorm, 11
logs_tnorm(scores_tnorm), 30
logs_tt, 11
logs_tt(scores_t), 35
logs_unif, 10
logs_unif(scores_unif), 36

plot.casestudy, 12, 12, 16
plot.mcstudy, 13, 13, 17
print.casestudy, 14
print.mcstudy, 14

run_casestudy, 4, 12, 14, 15, 16, 37
run_mcstudy, 13, 14, 16, 17, 37

scores, 17
scores_2pexp, 18
scores_2pnorm, 19
scores_beta, 19
scores_binom, 20
scores_exp, 20
scores_gamma, 21
scores_gev, 22
scores_gpd, 22
scores_hyper, 23
scores_lapl, 24
scores_llapl, 24
scores_llogis, 25
scores_lnorm, 25
scores_logis, 26
scores_mixnorm, 27
scores_moments, 28
scores_nbinom, 29
scores_norm, 30
scores_pois, 31
scores_sample_multiv, 31
scores_sample_univ, 33
scores_t, 35
scores_unif, 36
summary.casestudy, 37
summary.mcstudy, 37
Supplementary distributions: Positive real line, 38
Supplementary distributions: Real line, 39
Supplementary distributions: Variable support, 40
vs_sample, 32
vs_sample(scores_sample_multiv), 31