Evaluating probabilistic forecasts with scoringRules

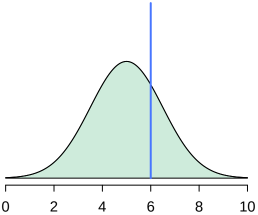
Alexander Jordan, Fabian Krüger, Sebastian Lerch

International Verification Methods Workshop, November 2020

Karlsruhe Institute of Technology

WAVES TO WEATHER

Evaluation of probabilistic forecasts: Proper scoring rules



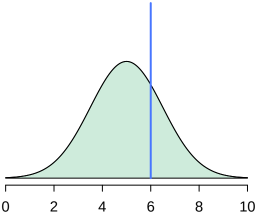
A (negatively oriented) proper scoring rule is any function

$$S(F, y)$$

such that for all F, G,

$$\mathbb{E}_{Y \sim G} S(G, Y) \leq \mathbb{E}_{Y \sim G} S(F, Y).$$

Evaluation of probabilistic forecasts: Proper scoring rules



A (negatively oriented) proper scoring rule is any function

$$S(F, y)$$

such that for all F, G,

$$\mathbb{E}_{Y \sim G} S(G, Y) \leq \mathbb{E}_{Y \sim G} S(F, Y).$$

Popular examples include

the logarithmic score

$$\mathrm{LogS}(F, y) = -\log(f(y))$$

the continuous ranked probability score

$$\mathrm{CRPS}(F, y) = \int_{-\infty}^{\infty} (F(z) - \mathbb{1}\{y \leq z\})^2 dz$$

Overview

The R package `scoringRules` provides functionality for comparative evaluation of probabilistic models based on proper scoring rules, covering a wide range of situations in applied work

- parametric predictive distributions
- simulated predictive distributions (e.g. ensemble forecasts)
- (simulated) multivariate predictive distributions

The package is available from Github and CRAN (`https://cran.r-project.org/package=scoringRules`).

Documenting paper with more details:

Jordan, A., Krüger, F. and Lerch, S. (2019) **Evaluating probabilistic forecasts with scoringRules**. *Journal of Statistical Software*, 90, 1–37.

Parametric predictive distributions

Essential functions for score computation follow the naming convention `[score]_[suffix]()`, for example

```
obs <- rnorm(5)
crps_norm(obs, mean = c(1:5), sd = c(1:5))

## [1] 0.288 1.625 1.570 2.003 2.744
```

Parametric predictive distributions

Essential functions for score computation follow the naming convention `[score]_[suffix]()`, for example

```
obs <- rnorm(5)
crps_norm(obs, mean = c(1:5), sd = c(1:5))

## [1] 0.288 1.625 1.570 2.003 2.744
```

Package developers may write S3 methods that hook into the S3 generic functions **crps**() and **logs**(). We reserve methods for the class `'numeric'`.

```
crps(obs, family = "normal", mean = c(1:5), sd = c(1:5))

## [1] 0.288 1.625 1.570 2.003 2.744
```

Examples

crps() and **logs**() functions with family argument are wrappers for the `[score]_[suffix]()` functions, but with meaningful error messages and input checks.

```
logs_norm(obs, mean = c(1:5), sd = c(1:4,-5))

## Warning in dnorm(y, location, scale, log = TRUE):
NaNs produced

## [1] 0.988 2.434 2.404 2.660   NaN

logs(obs, family = "normal", mean = c(1:5),
        sd = c(1:4,-5))

## Error in checkInput(input):  Parameter 'sd'
contains non-positive values.
```

Implemented parametric families

Closed-form expressions of the CRPS can be obtained for many parametric distributions and allow for efficient computation.

Distribution	Family	CRPS	LogS	Additional parameters
Distributions for variables on the real line				
Laplace	`"lapl"`	✓	✓	
Logistic	`"logis"`	✓	✓	
Normal	`"norm"`	✓	✓	
Mixture of normals	`"mixnorm"`	✓	✓	
Student's t	`"t"`	✓	✓	
Two-piece exponential	`"2pexp"`	✓	✓	
Two-piece normal	`"2pnorm"`	✓	✓	
Distributions for non-negative variables				
Exponential	`"exp"`	✓	✓	
Gamma	`"gamma"`	✓	✓	
Log-Laplace	`"llapl"`	✓	✓	
Log-logistic	`"llogis"`	✓	✓	
Log-normal	`"lnorm"`	✓	✓	

Implemented parametric families (continued)

Distribution	Family	CRPS	LogS	Additional parameters
Distributions with flexible support and/or point masses				
Beta	"beta"	✓	✓	limits
Uniform	"unif"	✓	✓	limits, point masses
Exponential	"exp2"		✓	location, scale
	"expM"	✓		location, scale, point mass
Gen. extreme value	"gev"	✓	✓	
Gen. Pareto	"gpd"	✓	✓	point mass (CRPS only)
Logistic	"tlogis"	✓	✓	limits (truncation)
	"clogis"	✓		limits (censoring)
	"gtclogis"	✓		limits, point masses
Normal	"tnorm"	✓	✓	limits (truncation)
	"cnorm"	✓		limits (censoring)
	"gtcnorm"	✓		limits, point masses
Student's t	"tt"	✓	✓	limits (truncation)
	"ct"	✓		limits (censoring)
	"gtct"	✓		limits, point masses
Distributions for discrete variables				
Binomial	"binom"	✓	✓	
Hypergeometric	"hyper"	✓	✓	
Negative binomial	"nbinom"	✓	✓	
Poisson	"pois"	✓	✓	

Simulated forecast distributions

In various applications (NWP ensembles, Bayesian forecasting models,...), the forecast distribution is only available through a discrete sample $X_1, \dots, X_m \sim F$.

The sample needs to be converted into an estimated distribution ($\hat{F}_m(z)$) to compute a proper scoring rule.

Simulated forecast distributions

In various applications (NWP ensembles, Bayesian forecasting models,...), the forecast distribution is only available through a discrete sample $X_1, \dots, X_m \sim F$.

The sample needs to be converted into an estimated distribution ($\hat{F}_m(z)$) to compute a proper scoring rule.

Using the empirical CDF as approximation the CRPS reduces to

$$\mathrm{CRPS}(\hat{F}_m, y) = \frac{1}{m}\sum_{i=1}^{m}|X_i - y| - \frac{1}{2m^2}\sum_{i=1}^{m}\sum_{j=1}^{m}|X_i - X_j|$$

or equivalently

$$\mathrm{CRPS}(\hat{F}_m, y) = \frac{2}{m^2}\sum_{i=1}^{m}(X_{(i)} - y)\left(m\mathbb{1}\{y < X_{(i)}\} - i + \frac{1}{2}\right).$$

Simulated forecast distributions

For the LogS, a predictive density is required and makes the use of kernel density estimation methods necessary.

crps_sample() and **logs_sample**(), provide implementations to compute CRPS and LogS for a vector of observations and a matrix with each row comprising one corresponding simulated sample.

Simulated forecast distributions

For the LogS, a predictive density is required and makes the use of kernel density estimation methods necessary.

crps_sample() and **logs_sample**(), provide implementations to compute CRPS and LogS for a vector of observations and a matrix with each row comprising one corresponding simulated sample.

The `'method'` argument controls which approximation method is used.

Implementation choices are based on theoretical considerations in

Krüger, F., Lerch, S., Thorarinsdottir, T.L. and Gneiting, T. (2020)
Predictive Inference Based on Markov Chain Monte Carlo Output.
International Statistical Review, in press.

Simulated forecast distributions

```
obs_n <- c(0, 1, 2)
sample_nm <- matrix(rnorm(3e4, mean = 2, sd = 3),
       nrow = 3)

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

## [1] 1.198 0.853 0.697

logs_sample(y = obs_n, dat = sample_nm, bw = NULL,
       show_messages = TRUE)

## Using the log score with kernel density estimation
tends to be fragile -- see KLTG (2019) for details.

## [1] 2.24 2.11 2.00
```

Usage example 1: Ensemble post-processing

Statistical post-processing is widely used to correct systematic errors of NWP ensemble predictions.

Here we illustrate how to evaluate post-processed ensemble forecasts of precipitation, based on data and methods from the `crch` package (Messner et al. 2016).

Usage example 1: Ensemble post-processing

Statistical post-processing is widely used to correct systematic errors of NWP ensemble predictions.

Here we illustrate how to evaluate post-processed ensemble forecasts of precipitation, based on data and methods from the `crch` package (Messner et al. 2016).

Using built-in functionality of the `crch` package we estimate a censored Gaussian regression model

$$\begin{aligned}
\mathbb{P}(Y = 0 | X_1, \ldots, X_m) &= F_\theta(0) \\
\mathbb{P}(Y \leq y | X_1, \ldots, X_m) &= F_\theta(y), \text{ for } y > 0, \\
\theta = (\mu, \sigma) &= (a_0 + a_1 \bar{X}, \exp(b_0 + b_1 s))
\end{aligned}$$

The example uses data for 3-day precipitation accumulations for Innsbruck, Austria from January 2000 to September 2013.

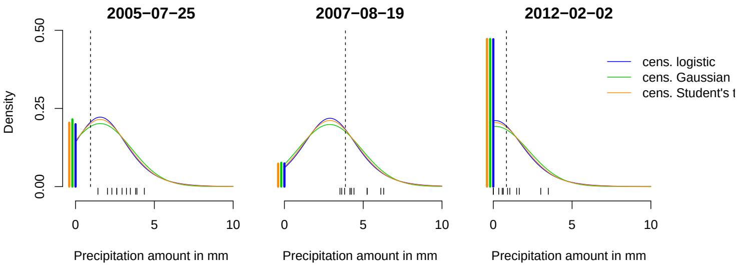
Usage example 1: Ensemble post-processing

```
library("crch"); data("RainIbk", package = "crch")
RainIbk <- sqrt(RainIbk)
ensfc <- RainIbk[, grep('^rainfc', names(RainIbk))]
RainIbk$ensmean <- apply(ensfc, 1, mean)
RainIbk$enssd <- apply(ensfc, 1, sd)
RainIbk <- subset(RainIbk, enssd > 0)
data_train <- subset(RainIbk,
        as.Date(rownames(RainIbk)) <= "2004-11-30")
data_eval <- subset(RainIbk,
        as.Date(rownames(RainIbk)) >= "2005-01-01")
ens_fc <- data_eval[, grep('^rainfc', names(RainIbk))]
```

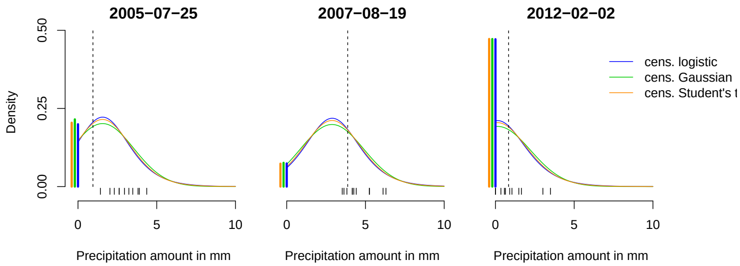
Estimation of censored regression models

```
CRCHgauss <- crch(rain ~ ensmean | log(enssd), data_train,
  dist = "gaussian", left = 0)
gauss_mu <- predict(CRCHgauss, data_eval, type = "location")
gauss_sc <- predict(CRCHgauss, data_eval, type = "scale")
```

Usage example 1: Ensemble post-processing – Results



Usage example 1: Ensemble post-processing – Results




```r
obs <- data_eval$rain
gauss_crps <- crps(obs, family = "cnorm", location = gauss_mu,
  scale = gauss_sc, lower = 0, upper = Inf)
ens_crps <- crps_sample(obs, dat = as.matrix(ens_fc))

scores <- data.frame(Postprocessed = gauss_crps, Ensemble = ens_crps)
sapply(scores, mean)

## Postprocessed      Ensemble
##         0.876         1.321
```

Usage example 2: Parameter estimation

Parameters of a model's forecast distribution can be determined by optimizing the value of a proper scoring rule, averaged over a training sample.

The computation functions `[score]_[family]()` entail little overhead in terms of input checks and are well suited for use in numerical optimization procedures such as `optim()`.

Functions to compute gradients and Hessian matrices of the CRPS have been implemented for a subset of parametric families, and can be supplied to assist numerical optimizers.

Usage example 2: Parameter estimation

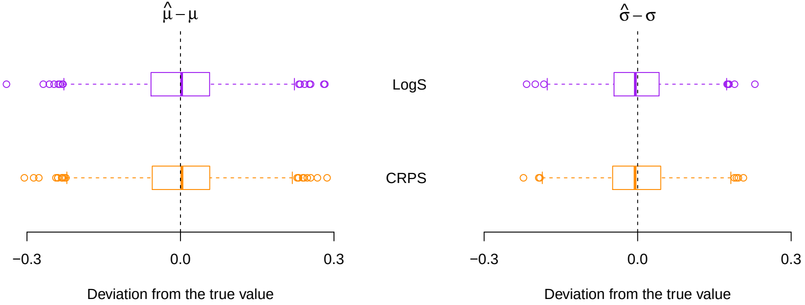
```
meancrps <- function(y_train, param){
  mean(crps_norm(y = y_train, mean = param[1], sd = param[2]))}
grad_meancrps <- function(y_train, param){
  apply(gradcrps_norm(y_train, param[1], param[2]), 2, mean)}

train_data <- rnorm(500, 1, -2)

estimates_crps <- optim(par = c(1, 1), fn = meancrps,
  gr = grad_meancrps, method = "BFGS", y_train = train_data)$par

estimates_ml <- c(mean(train_data),
  sd(train_data) * sqrt((n - 1) / n))
```

Usage example 2: Parameter estimation – Results



Boxplots of deviations from the true parameter values for estimates obtained via minimum CRPS and minimum LogS (i.e., maximum likelihood) estimation based on 1 000 independent samples of size 500.

Multivariate proper scoring rules: Background

Popular multivariate proper scoring rules for observations $\mathbf{y} \in \mathbb{R}^d$ and a sample $\mathbf{X}_1, \ldots, \mathbf{X}_m$ from a multivariate forecast distribution include the energy score

$$\mathrm{ES}(F, y) = \frac{1}{m} \sum_{i=1}^{m} \|\mathbf{X}_i - \mathbf{y}\| - \frac{1}{2m^2} \sum_{i=1}^{m} \sum_{j=1}^{m} \|\mathbf{X}_i - \mathbf{X}_j\|,$$

and the variogram score of order p

$$\mathrm{VS}^p(F, y) = \sum_{i=1}^{d} \sum_{j=1}^{d} w_{i,j} \left(\left| y^{(i)} - y^{(j)} \right|^p - \frac{1}{m} \sum_{k=1}^{m} \left| X_k^{(i)} - X_k^{(j)} \right|^p \right)^2.$$

Multivariate proper scoring rules: Background

Popular multivariate proper scoring rules for observations $\mathbf{y} \in \mathbb{R}^d$ and a sample $\mathbf{X}_1, \ldots, \mathbf{X}_m$ from a multivariate forecast distribution include the energy score

$$\mathrm{ES}(F, y) = \frac{1}{m} \sum_{i=1}^{m} \|\mathbf{X}_i - \mathbf{y}\| - \frac{1}{2m^2} \sum_{i=1}^{m} \sum_{j=1}^{m} \|\mathbf{X}_i - \mathbf{X}_j\|,$$

and the variogram score of order p

$$\mathrm{VS}^p(F, y) = \sum_{i=1}^{d} \sum_{j=1}^{d} w_{i,j} \left(\left| y^{(i)} - y^{(j)} \right|^p - \frac{1}{m} \sum_{k=1}^{m} \left| X_k^{(i)} - X_k^{(j)} \right|^p \right)^2.$$

Implementation in `scoringRules` (for single forecast cases only):

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

Summary and conclusions

- functionality to compute proper scoring rules for a wide range of situations prevalent in applications
- generally applicable and numerically efficient implementations based on theoretical considerations
- comprehensive collection of analytical expressions of the CRPS for parametric distributions

Summary and conclusions

- functionality to compute proper scoring rules for a wide range of situations prevalent in applications
- generally applicable and numerically efficient implementations based on theoretical considerations
- comprehensive collection of analytical expressions of the CRPS for parametric distributions
- Contributions are welcome! Examples include
  - parametric distributions relevant in your work
  - S3 methods for classes other than '`numeric`' (e.g., `crch` model objects)
- Possible future extensions include the addition of new scores such as weighted scoring rules.

Jordan, A., Krüger, F. and Lerch, S. (2019) **Evaluating probabilistic forecasts with scoringRules**. *Journal of Statistical Software*, 90, 1–37.