

Package ‘rtForecastEval’

May 9, 2026

Title Evaluate the Discrepancy Between Two Real-Time Updated Probabilistic Forecasts

Version 0.1.0

Description

Methods from Yeh, Rice, and Dubin (2022) <[doi:10.1080/00031305.2021.1967781](https://doi.org/10.1080/00031305.2021.1967781)> for comparing two continuously updated probabilistic forecasts under squared (Brier) loss: pointwise loss and variance, a global delta test (Monte Carlo p-values), simulation designs, and a naive pointwise band plot.

URL <https://github.com/chikuang/rtForecastEval>

BugReports <https://github.com/chikuang/rtForecastEval/issues>

License GPL-3

Encoding UTF-8

RoxygenNote 7.3.3

Imports dplyr (>= 1.0.1), ggplot2 (>= 3.3.2), stats (>= 4.0.3), tidyr (>= 1.1.1), RSpectra (>= 0.16.0), rlist (>= 0.4.6.1), purrr (>= 0.3.4), MASS (>= 7.3-52)

Suggests pkgload (>= 1.2.0), knitr (>= 1.28.0), readr (>= 1.3.1), reshape2 (>= 1.4.4), rmarkdown (>= 2.1.0), gridExtra (>= 2.3.0), tibble (>= 3.0.3), sde

VignetteBuilder knitr

NeedsCompilation no

Author Chi-Kuang Yeh [aut, cre] (ORCID: <<https://orcid.org/0000-0001-7057-2096>>),
Gregory Rice [ctb, ths],
Joel A. Dubin [ctb, ths]

Maintainer Chi-Kuang Yeh <cye@gsu.edu>

Repository CRAN

Date/Publication 2026-04-28 18:30:02 UTC

Contents

rtForecastEval-package	2
calc_eig	3
calc_L_s2	4
calc_pval	5
calc_Z	6
df_gen	7
lin_interp	8
plot_pcb	9

Index	11
--------------	-----------

rtForecastEval-package

rtForecastEval: Compare real-time probabilistic forecasts

Description

Methods from Yeh, Rice, and Dubin (2022) [doi:10.1080/00031305.2021.1967781](https://doi.org/10.1080/00031305.2021.1967781) for comparing two continuously updated probabilistic forecasts under squared (Brier) loss: pointwise loss and variance, a global delta test (Monte Carlo p-values), simulation designs, and a naive pointwise band plot.

Author(s)

Maintainer: Chi-Kuang Yeh <cyehe@gsu.edu> ([ORCID](#))

Other contributors:

- Gregory Rice [contributor, thesis advisor]
- Joel A. Dubin [contributor, thesis advisor]

References

Yeh, C.-K., Rice, G., & Dubin, J. A. (2022). Evaluating real-time probabilistic forecasts with application to National Basketball Association outcome prediction. *The American Statistician*, 76, 214–223. [doi:10.1080/00031305.2021.1967781](https://doi.org/10.1080/00031305.2021.1967781)

Preprint: <https://arxiv.org/abs/2010.00781> (PDF: <https://arxiv.org/pdf/2010.00781.pdf>).

See Also

Useful links:

- <https://github.com/chikuang/rtForecastEval>
- Report bugs at <https://github.com/chikuang/rtForecastEval/issues>

calc_eig	<i>Leading eigenvalues for the delta test covariance</i>
----------	--

Description

Forms the empirical covariance matrix of either centered or non-centered forecast differences across time points (see `diff_cent` vs `diff_non_cent` in the paper), and returns the leading eigenvalues scaled by one over `nsamp`, for use with `calc_pval()`. This matches the construction in the replication utility.R (`calc_p_val / eigs_sym`).

Usage

```
calc_eig(df, n_eig = 10, ngame, nsamp, grid = "grid", cent = FALSE)
```

Arguments

<code>df</code>	Data frame containing <code>grid</code> and a column <code>diff_cent</code> or <code>diff_non_cent</code> (vector differences of forecasts across games, aligned within each time point).
<code>n_eig</code>	Number of leading eigenvalues to extract (dimension D in the paper).
<code>ngame</code>	Number of independent games (used for scaling inner products).
<code>nsamp</code>	Number of time grid points.
<code>grid</code>	Name of the time grid column.
<code>cent</code>	If TRUE, use <code>diff_cent</code> (centered differences); if FALSE, use <code>diff_non_cent</code> .

Value

A numeric vector of length `n_eig` of eigenvalues (descending).

References

Yeh, C.-K., Rice, G., & Dubin, J. A. (2022). *The American Statistician*, 76, 214–223. doi:10.1080/00031305.2021.1967781

Preprint: <https://arxiv.org/abs/2010.00781>

Examples

```
library(dplyr)
library(rtForecastEval)
ngame <- 8L
nsamp <- 6L
gr <- seq(0, 1, length.out = nsamp)
df <- tidyr::expand_grid(grid = gr, game = seq_len(ngame)) %>%
  mutate(
    phat_A = stats::runif(dplyr::n()),
    phat_B = stats::runif(dplyr::n()),
    diff_non_cent = phat_A - phat_B
  )
```

```
calc_eig(df, n_eig = 4L, ngame = ngame, nsamp = nsamp, cent = FALSE)
```

calc_L_s2

Pointwise mean loss difference and variance

Description

For each time grid point, estimates the pointwise difference in expected squared (Brier) loss between two probabilistic forecasts, and the variance factor used for inference (Yeh, Rice, and Dubin, 2022). With default loss $L(x, y) = (x - y)^2$, the function computes the mean over games of $L(Y, \text{phat}_1) - L(Y, \text{phat}_2)$ at each time, and the influence-style terms s_i used to form sigma-squared over four, matching the paper replication code.

Usage

```
calc_L_s2(
  df,
  pA = "phat_1",
  pB = "phat_2",
  Y = "Y",
  grid = "grid",
  L = function(x, y) (x - y)^2
)
```

Arguments

df	A data frame with one row per (game × time) in long format.
pA, pB	Column names for the two forecast vectors (probabilities between 0 and 1).
Y	Column name for the binary outcome (0/1).
grid	Column name for the normalized time grid between 0 and 1.
L	Loss function; default is squared error, $\backslash(L(x,y)=(x-y)^2\backslash)$.

Value

A tibble with one row per grid value and columns L (mean loss difference), sigma2 (variance factor), and n (number of rows).

References

Yeh, C.-K., Rice, G., & Dubin, J. A. (2022). *The American Statistician*, 76, 214–223. doi:10.1080/00031305.2021.1967781

Preprint: <https://arxiv.org/abs/2010.00781>

Examples

```

library(dplyr)
library(rtForecastEval)
ngame <- 8L
nsamp <- 6L
gr <- seq(0, 1, length.out = nsamp)
df <- tidyr::expand_grid(grid = gr, game = seq_len(ngame)) %>%
  mutate(
    phat_A = stats::runif(dplyr::n()),
    phat_B = stats::runif(dplyr::n()),
    Y = stats::rbinom(dplyr::n(), 1L, 0.5)
  )
calc_L_s2(df, "phat_A", "phat_B", "Y", "grid")

```

calc_pval

Monte Carlo p-value and quantiles for the delta test

Description

Given leading eigenvalues of the covariance operator used in the delta test (from `calc_eig()`) and the observed statistic $\lambda(Z)$ (from `calc_Z()`), draws a Monte Carlo sample from the weighted sum of chi-square(1) variables and returns the right-tail p -value and quantiles of the null distribution (as in the paper's replication code).

Usage

```
calc_pval(Z, eig, quan, n_MC = 5000)
```

Arguments

Z	Observed test statistic from <code>calc_Z()</code> .
eig	Numeric vector of leading eigenvalues (length <code>n_eig</code>).
quan	Probabilities for which to return quantiles of the simulated null statistic (e.g. <code>c(0.90, 0.95, 0.99)</code>).
n_MC	Number of Monte Carlo draws (default 5000).

Details

Monte Carlo draws use R's current random-number stream. The function does not call `set.seed`; call it yourself beforehand if you need reproducible `p_val` and quantiles.

Value

A list with `p_val` (one-sided p -value) and `quantile` (named vector of quantiles at `quan`).

References

Yeh, C.-K., Rice, G., & Dubin, J. A. (2022). *The American Statistician*, 76, 214–223. doi:10.1080/00031305.2021.1967781

Preprint: <https://arxiv.org/abs/2010.00781>

Examples

```
if (requireNamespace("sde", quietly = TRUE)) {
  library(dplyr)
  nsamp <- 20L
  ngame <- 25L
  df <- df_gen(N = nsamp, Ngame = ngame) %>%
    dplyr::group_by(grid) %>%
    dplyr::mutate(
      p_bar_12 = mean(phat_A - phat_B),
      diff_non_cent = phat_A - phat_B,
      diff_cent = phat_A - phat_B - p_bar_12
    ) %>%
    dplyr::ungroup()
  Z <- calc_Z(df, "phat_A", "phat_B", "Y", "grid", nsamp, ngame)
  eig <- calc_eig(df, n_eig = 5L, ngame = ngame, nsamp = nsamp, cent = FALSE)
  set.seed(1)
  calc_pval(Z, eig, quan = c(0.9, 0.95), n_MC = 300L)
}
```

calc_Z

Delta test statistic for comparing two forecasting methods

Description

Computes the global test statistic Z for comparing two real-time forecasters under squared loss: Z equals (n_game / n_samp) times the sum over grid points of the squared pointwise mean loss differences, matching the implementation in the paper replication code (utility.R).

Usage

```
calc_Z(
  df,
  pA = "phat_1",
  pB = "phat_2",
  Y = "Y",
  grid = "grid",
  nsamp,
  ngame,
  L = function(x, y) (x - y)^2
)
```

Arguments

df	Data frame containing forecasts, outcomes, and grid.
pA, pB	Names of columns with the two probability forecasts.
Y	Name of the binary outcome column.
grid	Name of the column with normalized times between 0 and 1.
nsamp	Number of distinct time points (length of the grid).
ngame	Number of independent replicates (e.g. games).
L	Loss function; default is squared error.

Value

A single numeric value of the test statistic Z .

References

Yeh, C.-K., Rice, G., & Dubin, J. A. (2022). *The American Statistician*, 76, 214–223. doi:10.1080/00031305.2021.1967781

Preprint: <https://arxiv.org/abs/2010.00781>

Examples

```
library(dplyr)
library(rtForecastEval)
ngame <- 8L
nsamp <- 6L
gr <- seq(0, 1, length.out = nsamp)
df <- tidyr::expand_grid(grid = gr, game = seq_len(ngame)) %>%
  mutate(
    phat_A = stats::runif(dplyr::n()),
    phat_B = stats::runif(dplyr::n()),
    Y = stats::rbinom(dplyr::n(), 1L, 0.5)
  )
calc_Z(df, "phat_A", "phat_B", "Y", "grid", nsamp = nsamp, ngame = ngame)
```

df_gen

Simulate real-time probabilistic forecasts (paper designs)

Description

Generates replicated “games” and two competing forecast trajectories on a uniform grid from 0 to 1, following the simulation constructions used in Yeh, Rice, and Dubin (2022) (see `sanity_generator` / simulation scripts in the replication repository). Ornstein–Uhlenbeck (`type = "OU"`) or Brownian motion (`type = "BM"`) noise can drive the latent processes; outputs include binary outcomes Y and forecast probabilities `phat_A`, `phat_B` suitable for `calc_Z()`, `calc_eig()`, etc.

Usage

```
df_gen(N, Ngame, type = c("OU", "BM"), a = 1, b = 0.27)
```

Arguments

N	Number of interior time steps (grid has N+1 points from 0 to 1).
Ngame	Number of independent replicates (games).
type	"OU" (default) or "BM" for the innovation process used in the bivariate construction.
a, b	Constants controlling the drift of the latent trajectory (see replication code).

Details

Requires the **sde** package (Suggests) for Brownian paths.

Value

A tibble with columns including game, grid, Y, phat_A, phat_B, and latent W1, W2, etc., in long format.

References

Yeh, C.-K., Rice, G., & Dubin, J. A. (2022). *The American Statistician*, 76, 214–223. doi:10.1080/00031305.2021.1967781

Preprint: <https://arxiv.org/abs/2010.00781>

Examples

```
if (requireNamespace("sde", quietly = TRUE)) {  
  d <- df_gen(N = 15L, Ngame = 12L)  
  head(d)  
}
```

lin_interp

Linear interpolation of a forecast trajectory onto a grid

Description

Maps a vector of probabilities prob observed at times grid onto an equally spaced grid of length length(grid) between 0 and 1 using linear interpolation (stats::approx). Useful when aligning ESPN or model outputs to the common grid used in the NBA analysis (see replication pre_process / utility.R).

Usage

```
lin_interp(prob, grid, outcome)
```

Arguments

prob Numeric vector of forecast probabilities.
 grid Numeric vector of time points (same length as prob) between 0 and 1.
 outcome Scalar binary outcome $\{Y\}$ to attach to every interpolated row.

Value

A tibble with columns phat_approx, grid, and Y.

References

Yeh, C.-K., Rice, G., & Dubin, J. A. (2022). *The American Statistician*, 76, 214–223. doi:10.1080/00031305.2021.1967781

Preprint: <https://arxiv.org/abs/2010.00781>

Examples

```
library(rtForecastEval)
g <- seq(0, 1, length.out = 9L)
lin_interp(prob = seq(0.1, 0.9, length.out = 9L), grid = g, outcome = 1L)
```

 plot_pcb

Naive pointwise confidence band for mean loss difference

Description

Plots the pointwise mean loss difference from `calc_L_s2()` (column L) against time, with a normal-theory band using `sigma2` and the point sample size `n`. This is the **naive** $\{t\}$ -style band from the paper; global inference uses `calc_Z()` / `calc_pval()` instead.

Usage

```
plot_pcb(
  df,
  grid = "grid",
  L = "L",
  var = "sigma2",
  title = "Pointwise mean loss difference (A vs B)",
  subtitle = "Naive normal band (95%); use calc_pval() for global test",
  caption = NULL,
  xlab = "Normalized time (grid)",
  ylab = "Mean squared loss difference"
)
```

Arguments

df	Output of <code>calc_L_s2()</code> (or compatible tibble with <code>grid</code> , <code>L</code> , <code>sigma2</code> , <code>n</code>).
grid	Name of the x-axis column (default "grid").
L	Name of the pointwise mean difference column (default "L").
var	Name of the variance column (default "sigma2").
title, subtitle, caption	Passed to <code>ggplot2::labs()</code> . Defaults describe the plot when NULL (some labels are omitted if NULL).
xlab, ylab	Axis labels (see <code>ggplot2::labs()</code>).

Details

This plot summarizes **relative skill** (loss difference) over time — not a classical **calibration** plot (forecast vs observed event rate). For a simple reliability-style view from the same long-format data, see the package vignette example that bins `phat` and plots mean `Y` vs mean forecast.

Value

A `ggplot2` object.

References

Yeh, C.-K., Rice, G., & Dubin, J. A. (2022). *The American Statistician*, 76, 214–223. doi:10.1080/00031305.2021.1967781

Preprint: <https://arxiv.org/abs/2010.00781>

Examples

```
library(dplyr)
library(rtForecastEval)
ngame <- 8L
nsamp <- 6L
gr <- seq(0, 1, length.out = nsamp)
df <- tidyr::expand_grid(grid = gr, game = seq_len(ngame)) %>%
  mutate(
    phat_A = stats::runif(dplyr::n()),
    phat_B = stats::runif(dplyr::n()),
    Y = stats::rbinom(dplyr::n(), 1L, 0.5)
  )
tmp <- calc_L_s2(df, "phat_A", "phat_B", "Y", "grid")
plot_pcb(tmp)
```

Index

`**delta` (rtForecastEval-package), 2

`additional` (rtForecastEval-package), 2

`and` (rtForecastEval-package), 2

`associated` (rtForecastEval-package), 2

`bands.` (rtForecastEval-package), 2

`blocks` (rtForecastEval-package), 2

`building` (rtForecastEval-package), 2

`calc_eig`, 3

`calc_eig()`, 5, 7

`calc_L_s2`, 4

`calc_L_s2()`, 9, 10

`calc_pval`, 5

`calc_pval()`, 3, 9

`calc_Z`, 6

`calc_Z()`, 5, 7, 9

`calibration` (rtForecastEval-package), 2

`Carlo` (rtForecastEval-package), 2

`companion` (rtForecastEval-package), 2

`comparing` (rtForecastEval-package), 2

`confidence` (rtForecastEval-package), 2

`continuously` (rtForecastEval-package), 2

`Core` (rtForecastEval-package), 2

`covariance` (rtForecastEval-package), 2

`data` (rtForecastEval-package), 2

`designs` (rtForecastEval-package), 2

`df_gen`, 7

`difference` (rtForecastEval-package), 2

`draws` (rtForecastEval-package), 2

`Dubin` (rtForecastEval-package), 2

`eigenvalues` (rtForecastEval-package), 2

`estimate` (rtForecastEval-package), 2

`exports` (rtForecastEval-package), 2

`for` (rtForecastEval-package), 2

`forecasts` (rtForecastEval-package), 2

`form` (rtForecastEval-package), 2

`from` (rtForecastEval-package), 2

`ggplot2::labs()`, 10

`Implements` (rtForecastEval-package), 2

`in` (rtForecastEval-package), 2

`in-game` (rtForecastEval-package), 2

`is` (rtForecastEval-package), 2

`its` (rtForecastEval-package), 2

`leading` (rtForecastEval-package), 2

`lin_interp`, 8

`loss` (rtForecastEval-package), 2

`mean` (rtForecastEval-package), 2

`Monte` (rtForecastEval-package), 2

`naive` (rtForecastEval-package), 2

`obtain` (rtForecastEval-package), 2

`of` (rtForecastEval-package), 2

`p-values` (rtForecastEval-package), 2

`package` (rtForecastEval-package), 2

`package;` (rtForecastEval-package), 2

`paper;` (rtForecastEval-package), 2

`plot_pcb`, 9

`plots` (rtForecastEval-package), 2

`pointwise` (rtForecastEval-package), 2

`probabilistic` (rtForecastEval-package), 2

`probabilities` (rtForecastEval-package), 2

`quantiles` (rtForecastEval-package), 2

`related` (rtForecastEval-package), 2

`replication` (rtForecastEval-package), 2

`repository` (rtForecastEval-package), 2

`reproduces` (rtForecastEval-package), 2

`Rice`, (rtForecastEval-package), 2

rtForecastEval
 (rtForecastEval-package), 2
rtForecastEval-package, 2

separate (rtForecastEval-package), 2
set.seed, 5
simulation (rtForecastEval-package), 2
squared (rtForecastEval-package), 2
statistic (rtForecastEval-package), 2
statistical (rtForecastEval-package), 2
steps: (rtForecastEval-package), 2
surfaces, (rtForecastEval-package), 2

test (rtForecastEval-package), 2
test** (rtForecastEval-package), 2
The (rtForecastEval-package), 2
the (rtForecastEval-package), 2
there. (rtForecastEval-package), 2
this (rtForecastEval-package), 2
tools (rtForecastEval-package), 2
two (rtForecastEval-package), 2

under (rtForecastEval-package), 2
updated (rtForecastEval-package), 2
used (rtForecastEval-package), 2

variance (rtForecastEval-package), 2

win (rtForecastEval-package), 2
workflow, (rtForecastEval-package), 2

Yeh, (rtForecastEval-package), 2