Package ‘RCtest’

August 21, 2026

Title Reality Check and Predictive Ability Tests for Forecast Evaluation

Description Implements a comprehensive suite of statistical tests for evaluating the accuracy of forecasting models against a benchmark. The package is grounded in the reality check framework of White (2000) <doi:10.1111/1468-0262.00152>, extended by Hansen (2005) <doi:10.1198/073500105000000063> for Superior Predictive Ability (SPA), Giacomini & White (2006) <doi:10.1111/j.1468-0262.2006.00718.x> for Conditional Predictive Ability (CPA), and Corradi & Swanson (2006) <doi:10.1016/j.jeconom.2005.07.026> for predictive density evaluation via the Kullback-Leibler Information Criterion (KLIC) and ZP Quantile Loss test, the Continuous Ranked Probability Score (CRPS) (Gneiting & Raftery, 2007) <doi:10.1198/016214506000001437>, coverage tests (Kupiec, 1995) <doi:10.3905/jod.1995.407942>, HAC covariance estimation (Newey & West, 1987) <doi:10.2307/1913610>, and Moving Block Bootstrap resampling (Kunsch, 1989) <doi:10.1214/aos/1176347265>.

Imports ggplot2, gridExtra, ggrepel, rlang, stats

Encoding UTF-8

LazyData true

Note This research was funded in whole by National Science Centre, Poland, grant number 2022/45/B/HS4/00510.

RoxygenNote 7.3.3

Version 1.1

NeedsCompilation no

Author Joanna Jedrzejewska [aut, cre] (Faculty of Economic Sciences, University of Warsaw, Poland),
Krzysztof Drachal [ctb] (Faculty of Economic Sciences, University of Warsaw, Poland)

Maintainer Joanna Jedrzejewska `<j.jedrzejewska3@uw.edu.pl>`

Depends R (>= 3.5.0)

Language en-US

Repository CRAN

Date/Publication 2026-08-20 23:02:32 UTC

Contents

`compute_crps` *Compute Continuous Ranked Probability Score (CRPS)*

Description

Calculates the Continuous Ranked Probability Score (CRPS) using the energy score (Monte Carlo) approximation for a single forecast period.

Usage

```
compute_crps(forecast_density, target_realization)
```

Arguments

`forecast_density`
: `numeric` vector of simulated forecasts (density samples) representing the predictive distribution for a single time period.

`target_realization`
: `numeric` scalar representing the realized value against which the forecast density is evaluated.

Details

The CRPS is a strictly proper scoring rule that jointly rewards calibration and sharpness of a probabilistic forecast. It is computed via the energy score identity:

$$CRPS = E|X - y| - \frac{1}{2}E|X - X'|$$

where X, X' are independent draws from the forecast distribution and y is the realization. **Lower values are better**: a CRPS of 0 indicates a perfect point-mass forecast at the true realization.

Value

`numeric` scalar representing the CRPS loss, or `NA` if input is invalid. Lower values indicate better probabilistic forecast accuracy.

References

Gneiting, T., & Raftery, A. E. (2007). Strictly Proper Scoring Rules, Prediction, and Estimation. *Journal of the American Statistical Association*, 102(477), 359–378. doi:10.1198/016214506000001437

Examples

```
data(metals)
# metals: 165 x 15; columns 1-14 are competing forecasts, column 15 is the benchmark

# CRPS for forecast 1, period 1:
# Use the cross-sectional spread of all competing forecasts at period t=1 as the density
density_samples <- as.numeric(metals[1, 1:14])
realized_value <- metals[1, 15]
compute_crps(density_samples, realized_value)

# In practice, iterate over all forecasts and periods.
# For forecast k and period t, the predictive density is approximated by shifting the
# cross-sectional spread of all K competing forecasts so that it is centred at the
# cross-sectional mean of forecasts at period t. Specifically, for each forecast k:
#   density_samples_tk = (forecasts of all K forecast at t) - forecast_k(t) +
                       mean # (all K forecasts at t)
# This preserves the spread (diversity) across forecasts while recentring around the
# cross-sectional mean rather than around forecast k's own point forecast. It is an
# empirical approximation to the predictive distribution when no parametric density
# is available.
P <- nrow(metals)
```

```
K <- ncol(metals) - 1L # 14 competing forecasts
crps_matrix <- matrix(NA_real_, nrow = P, ncol = K,
                      dimnames = list(NULL, colnames(metals)[1:K]))
for (t in seq_len(P)) {
  for (k in seq_len(K)) {
   density_samples_tk <- as.numeric(metals[t, 1:K]) - metals[t, k] + mean(metals[t, 1:K])
    crps_matrix[t, k]  <- compute_crps(as.numeric(density_samples_tk),
                                       target_realization = metals[t, ncol(metals)])
  }
}
head(crps_matrix)
```

`compute_klic` *Compute Kullback-Leibler Information Criterion (KLIC) Negative Log-Likelihood Scores*

Description

Computes the per-period Negative Log-Likelihood Score (NLS) loss matrix under a Gaussian predictive density assumption. The NLS is the loss function corresponding to minimisation of the Kullback-Leibler Information Criterion (KLIC) distance from the true density (Corradi & Swanson, 2006).

Usage

```
compute_klic(
  forecast_matrix,
  forecast_sd_models,
  benchmark_col = ncol(forecast_matrix)
)
```

Arguments

`forecast_matrix`
: `matrix` of dimension `P x K_total`. The benchmark column supplies the realized values y_t.

`forecast_sd_models`
: `matrix` of dimension `P x (K_total - 1)`, containing time-varying forecast standard deviations, typically from `estimate_forecast_variance`.

`benchmark_col`
: Index or name of the benchmark column. Defaults to the last column.

Details

For each competing forecast `k` and period `t`:

$$NLS_{t,k} = -\log \phi(y_t \mid \hat{y}_{t,k},\, \hat{\sigma}_{t,k})$$

where ϕ denotes the Gaussian density, y_t is the realized value, $\hat{y}_{t,k}$ is the point forecast, and $\hat{\sigma}_{t,k}$ is the forecast standard deviation. Minimising the average NLS is equivalent to minimising the KLIC distance between the forecast's predictive density and the true density (Corradi & Swanson, 2006). **Lower NLS values are better.** The benchmark column in the returned matrix is set to zero.

Value

matrix of dimension P x K_total containing NLS values. Lower values indicate better density forecast accuracy. The benchmark column is set to zero.

References

Corradi, V., & Swanson, N. R. (2006). Predictive density and conditional confidence interval accuracy tests. *Journal of Econometrics*, 135(1–2), 187–228. doi:10.1016/j.jeconom.2005.07.026

Corradi, V., & Swanson, N. R. (2011). The White Reality Check and some of its recent extensions. In *Festschrift in honor of Halbert L. White*.

See Also

kullback_leibler_test, estimate_forecast_variance

Examples

```
data(metals)
benchmark_col      <- 15
K_total            <- ncol(metals)
comp_cols          <- setdiff(seq_len(K_total), benchmark_col)
realized           <- metals[, benchmark_col]
forecast_variance  <- estimate_forecast_variance(metals, realized = realized,
                        benchmark_col = benchmark_col)
forecast_sd_models <- sqrt(forecast_variance[, comp_cols])
klic_loss <- compute_klic(metals, forecast_sd_models,
                          benchmark_col = benchmark_col)
head(klic_loss)
```

---

compute_kupiec *Value-at-Risk (VaR) Unconditional Coverage Test (Kupiec)*

Description

Performs Kupiec's (1995) Unconditional Coverage (UC) test for evaluating Value-at-Risk (VaR) forecasts from competing forecast against realized values.

Hypotheses:

- **H0:** The forecast correctly captures VaR — violations occur with the expected frequency alpha.
- **H1:** The forecast fails to correctly capture VaR — the observed frequency of violations differs significantly from alpha.

Usage

```
compute_kupiec(
  forecast_matrix,
  forecast_sd_models,
  benchmark_col = ncol(forecast_matrix),
  alpha = 0.05
)
```

Arguments

`forecast_matrix`
: `matrix` of dimension `P x K_total`. Columns contain point forecasts for each model; the benchmark column supplies the realized values.

`forecast_sd_models`
: `matrix` of dimension `P x K`, where `K = K_total - 1`. Contains time-varying forecast standard deviations, typically from `estimate_forecast_variance`.

`benchmark_col`
: Index or name of the benchmark column. Defaults to the last column.

`alpha`
: `numeric` VaR significance level (e.g., `0.05` for 95% VaR). A violation occurs when the realized value falls below the estimated VaR.

Details

For each competing forecast `k`, the VaR at level `alpha` is:

$$VaR_{t,k} = \hat{y}_{t,k} + \Phi^{-1}(\alpha) \cdot \hat{\sigma}_{t,k}$$

where Φ^{-1} is the standard normal quantile function. A violation occurs when the realized value falls below $VaR_{t,k}$. The likelihood-ratio statistic LR_{UC} follows a $\chi^2(1)$ distribution under H0 (Kupiec, 1995). **Failing to reject H0** (large p-value) indicates correctly calibrated VaR; **rejecting H0** (small p-value) indicates the forecast under- or over-estimates tail risk.

Value

A named list (one element per competing forecast) of `htest` objects, each containing:

`statistic` The LR-UC test statistic (χ^2-distributed under H0).

`p.value` P-value from the $\chi^2(1)$ distribution. A large p-value indicates correctly calibrated VaR coverage.

`actual_exceedances` Observed number of VaR violations.

`expected` Expected number of violations (`P * alpha`).

References

Kupiec, P. H. (1995). Techniques for Verifying the Accuracy of Risk Measurement Models. *The Journal of Derivatives*, 3(2), 173–184. doi:10.3905/jod.1995.407942

See Also

`estimate_forecast_variance`, `run_comprehensive_erc_analysis`

Examples

```
data(metals)
benchmark_col      <- 15
K_total            <- ncol(metals)
comp_cols          <- setdiff(seq_len(K_total), benchmark_col)
realized           <- metals[, benchmark_col]
forecast_variance  <- estimate_forecast_variance(metals, realized = realized,
                        benchmark_col = benchmark_col, window_size = 20)
forecast_sd_models <- sqrt(forecast_variance[, comp_cols])
coverage_results   <- compute_kupiec(metals, forecast_sd_models,
                        benchmark_col = benchmark_col, alpha = 0.05)
print(coverage_results[[1]])
```

---

compute_per_model_statistics

Per-Model Diebold-Mariano Test (HAC + MBB Bootstrap)

Description

Performs a Diebold-Mariano (1995) test for each competing forecast separately, testing whether the predictive accuracy of that forecast differs significantly from the benchmark forecast. The test statistic is the sample mean loss differential standardised by a Newey-West HAC standard error; p-values are provided both analytically and via Moving Block Bootstrap (MBB). The direction of the alternative hypothesis is controlled by the H1 argument: "same" for a two-sided test ($H_1 : \bar{d}_k \neq 0$), "more" for the one-sided test that forecast k is more accurate than the benchmark ($H_1 : \bar{d}_k > 0$), and "less" for the one-sided test that forecast k is less accurate than the benchmark ($H_1 : \bar{d}_k < 0$).

Usage

```
compute_per_model_statistics(
  loss_differences,
  model_names,
  n_boot = 999,
  block_length = 5,
  alpha = 0.05,
  h = 1,
  H1 = "same"
)
```

Arguments

loss_differences
: matrix of dimension P x K, where P is the number of forecast periods and K is the number of competing forecasts. Each column k contains the loss differential series $d_{t,k} = g(e_{0,t}) - g(e_{k,t})$ for a generic loss function g. In the standard workflow of this package (run_comprehensive_erc_analysis), g is the squared error loss, so $d_{t,k} = (y_t - \hat{y}_{t,0})^2 - (y_t - \hat{y}_{t,k})^2$. A positive value of $d_{t,k}$ means the forecast from model k is more accurate than the benchmark forecast in period t.

model_names	character vector of length K with names of the competing model forecasts.
n_boot	integer number of MBB replications for P_Value_Boot. Default 999; see Davidson & MacKinnon (2000).
block_length	integer block length for HAC and MBB. Rule of thumb: $T^{1/3}$; for P = 165 approximately 5–6. Default is 5.
alpha	numeric significance level. Default 0.05.
h	integer forecast horizon (number of steps ahead). Default is 1 (one-step-ahead). Passed to the Harvey, Leybourne & Newbold (1997) small-sample correction; see *Details*.
H1	character alternative hypothesis: "same" (two-sided, default), "more" (one-sided, forecast k better), or "less" (one-sided, forecast k worse). See *Details*.

Details

For each forecast k, the loss differential series is:

$$d_{t,k} = g(e_{0,t}) - g(e_{k,t})$$

where $g(\cdot)$ is the loss function used to construct loss_differences. In the standard workflow of this package (run_comprehensive_erc_analysis), g is the squared error loss:

$$d_{t,k} = (y_t - \hat{y}_{t,0})^2 - (y_t - \hat{y}_{t,k})^2$$

The Diebold-Mariano test statistic is:

$$DM_k = \frac{\bar{d}_k}{\hat{SE}_{HAC,k}}$$

For multi-step forecasts ($h > 1$), the Harvey, Leybourne & Newbold (1997) small-sample correction is applied:

$$DM_k^* = DM_k \times \sqrt{\frac{T + 1 - 2h + \frac{1}{T}h(h-1)}{T}}$$

For $h = 1$ the correction reduces to $\sqrt{(T-1)/T}$, which approaches 1 as $T \to \infty$. For $h > 1$ the correction inflates the statistic, improving finite-sample size control. The corrected statistic DM_k^* is compared to a $t(T-1)$ distribution. The analytic p-value (P_Value) uses the t-distribution with $P - 1$ degrees of freedom. The bootstrap p-value (P_Value_Boot) uses MBB resampling (Kunsch, 1989) with recentering at the sample mean $\bar{d}_k$, placing the bootstrap distribution under H0. The Harvey correction is applied consistently to both the analytic and bootstrap statistics. The p-values are computed according to the alternative hypothesis specified by H1:

Alternative Hypothesis and P-values (H1):

"same" – **two-sided** $p = 2 \cdot P(t_{T-1} < -|DM_k^*|)$
"more" – **one-sided right** $p = P(t_{T-1} > DM_k^*)$
"less" – **one-sided left** $p = P(t_{T-1} < DM_k^*)$

The bootstrap analogue replaces the t-distribution probability with the empirical proportion of bootstrap statistics falling in the appropriate tail.

where T follows a t-distribution with $P - 1$ degrees of freedom. The bootstrap analogue replaces the t-distribution tail probability with the empirical proportion of bootstrap statistics falling in the appropriate tail.

This function performs K individual tests and does *not* control for multiple comparisons. For a joint test controlling the family-wise error rate, use `white_reality_check` or `superior_predictive_ability_test`.

Note on MASE: When `loss_differences` are constructed from Mean Absolute Scaled Errors, the scaling (division by the naive benchmark MAE, i.e. `mean(abs(diff(realizations)))`) must be applied *before* passing the loss differentials to this function. `compute_per_model_statistics` receives pre-computed loss differentials and applies no internal rescaling — the caller is responsible for ensuring that MASE-based `loss_differences` already contain scaled errors. In `run_comprehensive_erc_analysis` this is handled automatically.

Value

`data.frame` with one row per competing model forecast:

`Model`	Model name.
`Mean_Loss_Diff`	Sample mean of $d_{t,k}$.
`Frac_Better_Than_Benchmark`	Fraction of periods where $d_{t,k} > 0$.
`T_Stat`	Harvey-corrected DM statistic DM_k^*.
`P_Value`	Analytic p-value (t-distribution, $T - 1$ df).
`P_Value_Boot`	MBB bootstrap p-value.
`Significant`	`TRUE` if `P_Value <= alpha`.
`Significant_Boot`	`TRUE` if `P_Value_Boot <= alpha`.

References

Davidson, R., & MacKinnon, J. G. (2000). Bootstrap tests: How many bootstraps? *Econometric Reviews*, 19(1), 55–68. doi:10.1080/07474930008800459

Diebold, F. X., & Mariano, R. S. (1995). Comparing Predictive Accuracy. *Journal of Business & Economic Statistics*, 13(3), 253–263. doi:10.1080/07350015.1995.10524599

Harvey, D., Leybourne, S., & Newbold, P. (1997). Testing the equality of prediction mean squared errors. *International Journal of Forecasting*, *13*(2), 281–291. doi:10.1016/S01692070(96)007194

Kunsch, H. R. (1989). The jackknife and the bootstrap for general stationary observations. *The Annals of Statistics*, 17(3), 1217–1241. doi:10.1214/aos/1176347265

Newey, W. K., & West, K. D. (1987). A Simple, Positive Semi-Definite, Heteroskedasticity and Autocorrelation Consistent Covariance Matrix. *Econometrica*, 55(3), 703–708. doi:10.2307/1913610

See Also

`white_reality_check`, `superior_predictive_ability_test`, `estimate_long_run_covariance`

Examples

```
data(metals)
P       <- nrow(metals)
K_total <- ncol(metals)
```

```
K       <- K_total - 1
# A small offset (+0.5) avoids degenerate zero loss differences (illustration only).
realized         <- c(metals[-1, K_total], metals[P, K_total]) + 0.5
bench_loss       <- (metals[, K_total] - realized)^2
model_loss       <- (metals[, 1:K]     - realized)^2
loss_differences <- bench_loss - model_loss
model_names      <- colnames(metals)[1:K]
# Two-sided test (default)
result_df <- compute_per_model_statistics(loss_differences, model_names,
                                           n_boot = 10)
print(result_df)
# One-sided test: H1 = forecast is more accurate than benchmark
result_more <- compute_per_model_statistics(loss_differences, model_names,
                                             n_boot = 10, H1 = "more")
print(result_more)
```

---

compute_zp *Compute ZP Quantile Loss*

Description

Computes the per-period ZP quantile loss matrix based on the squared difference between the indicator of a tail event and the forecast's predicted probability of that event (Corradi & Swanson, 2006, eq. 7).

Usage

```
compute_zp(
  forecast_matrix,
  forecast_sd_models,
  threshold,
  benchmark_col = ncol(forecast_matrix)
)
```

Arguments

`forecast_matrix`
: `matrix` of dimension `P x K_total`. The benchmark column supplies the realized values y_t.

`forecast_sd_models`
: `matrix` of dimension `P x K`, where `K = K_total - 1`. Contains time-varying forecast standard deviations, typically from `estimate_forecast_variance`.

`threshold`
: `numeric` tail threshold τ. The ZP loss measures how well each model predicts the probability of $y_t \leq \tau$. Typically set to a low quantile of the realized series, e.g., `quantile(realized, 0.05)` for the 5th-percentile left tail. In `run_comprehensive_erc_analysis` this is computed automatically as `quantile(realizations, zp_quantile)`.

`benchmark_col`
: Index or name of the benchmark column. Defaults to the last column.

Details

For each competing forecast k and period t:

$$ZP_{t,k} = \left(\mathbf{1}(y_t \leq \tau) - \Phi\left(\frac{\tau - \hat{y}_{t,k}}{\hat{\sigma}_{t,k}}\right)\right)^2$$

where y_t is the realized value, τ is the threshold, $\hat{y}_{t,k}$ is the point forecast, and $\hat{\sigma}_{t,k}$ is the forecast standard deviation. **Lower ZP values are better.**

Choosing τ at the 5th percentile focuses evaluation on whether forecasts correctly predict the risk of falling into the worst 5% of outcomes. The benchmark column is assigned a point-mass predictive distribution ($\hat{\sigma} = 10^{-6}$), which approximates the Brier score for the tail indicator and serves as a conservative reference. When the benchmark is the Historical Average (HA), the ZP test thus evaluates whether any competing forecast's calibrated tail probability outperforms the HA's point prediction of the tail event.

Value

matrix of dimension P x K_total containing ZP loss values. Lower values indicate better left-tail probability calibration. The benchmark column uses $\hat{\sigma} = 10^{-6}$.

References

Corradi, V., & Swanson, N. R. (2006). Predictive density and conditional confidence interval accuracy tests. *Journal of Econometrics*, 135(1–2), 187–228. doi:10.1016/j.jeconom.2005.07.026

Corradi, V., & Swanson, N. R. (2011). The White Reality Check and some of its recent extensions. In *Festschrift in honor of Halbert L. White*.

See Also

reality_check_zp_test, estimate_forecast_variance

Examples

```
data(metals)
benchmark_col      <- 15
K_total            <- ncol(metals)
comp_cols          <- setdiff(seq_len(K_total), benchmark_col)
realized           <- metals[, benchmark_col]
forecast_variance  <- estimate_forecast_variance(metals, realized = realized,
                        benchmark_col = benchmark_col)
forecast_sd_models <- sqrt(forecast_variance[, comp_cols])
threshold_val      <- quantile(metals[, benchmark_col], 0.10)
zp_loss <- compute_zp(metals, forecast_sd_models,
                      threshold     = threshold_val,
                      benchmark_col = benchmark_col)
head(zp_loss)
```

create_unified_summary *Create Unified Summary*

Description

Creates a summary data frame consolidating p-values and conclusions for all statistical tests across all datasets and error metrics. Covers White's Reality Check (WRC), Superior Predictive Ability (SPA), Conditional Predictive Ability (CPA), ZP Quantile Loss test, and Kullback-Leibler (KLIC) test.

Usage

```
create_unified_summary(comprehensive_results, alpha = 0.05)
```

Arguments

comprehensive_results
: `list` output from `run_comprehensive_erc_analysis`.

alpha
: Numeric significance level for determining conclusions. Default is 0.05.

Value

`list` containing a data frame named `summary` with columns: `Dataset`, `Test`, `P_Value`, `Statistic`, `Conclusion` ("H0 rejected" or "H0 accepted").

References

White, H. (2000). A reality check for data snooping. *Econometrica*, 68(5), 1097–1126. doi:10.1111/14680262.00152

Hansen, P. R. (2005). A Test for Superior Predictive Ability. *Journal of Business & Economic Statistics*, 23(4), 365–380. doi:10.1198/073500105000000063

Giacomini, R., & White, H. (2006). Tests of Conditional Predictive Ability. *Econometrica*, 74(6), 1545–1578. doi:10.1111/j.14680262.2006.00718.x

Corradi, V., & Swanson, N. R. (2006). Predictive density and conditional confidence interval accuracy tests. *Journal of Econometrics*, 135(1–2), 187–228. doi:10.1016/j.jeconom.2005.07.026

See Also

`run_comprehensive_erc_analysis`, `generate_comprehensive_report`, `compute_per_model_statistics`

Examples

```
data(metals)
realizations <- list(M = metals[, ncol(metals)])
prep_list    <- list(M = list(R_start = 0))
f_hat        <- list(list(NULL, NULL, metals))
names(f_hat) <- "M"
res <- run_comprehensive_erc_analysis(
  data_list_prepared = prep_list,
  mods_matrix        = matrix(0),
  alpha_grid         = 0.05,
  window_size        = 20,
  y_hat_all          = f_hat,
  y_raw              = realizations,
  block_length       = 5,
  n_boot             = 10,
  zp_quantile        = 0.05
)
summary_table <- create_unified_summary(res$aggregate_results)
print(summary_table$summary)
```

---

estimate_forecast_variance

Estimate Forecast Variance via Rolling Window

Description

Estimates forecast variance from each competing model's own historical forecast errors relative to the realized outcome, using a rolling window. Used to approximate the time-varying predictive standard deviation for each competing model forecast, required by `compute_klic`, `compute_zp`, and `compute_kupiec`.

Usage

```
estimate_forecast_variance(
  forecast_matrix,
  realized,
  benchmark_col = ncol(forecast_matrix),
  window_size = 20
)
```

Arguments

`forecast_matrix`
: `matrix` of dimension `P` x `K_total`, where `P` is the number of forecast periods and `K_total` is the total number of columns (competing forecasts plus the benchmark).

realized — `numeric` vector of length `P` containing the realized (observed) outcome for each forecast period. Forecast errors are computed relative to this series, not relative to the benchmark forecast.

benchmark_col — Index or name of the benchmark column. Defaults to the last column. Retained for column bookkeeping (the benchmark column of the returned matrix is set to zero) and is no longer used to define forecast errors (see “Note” below).

window_size — `integer` rolling window size. For the first `window_size` periods, the full available history is used instead (expanding window). From period `window_size + 1` onwards, a rolling window of exactly `window_size` observations is used.

Details

For each competing forecast `k` and period `t`, the forecast error is defined as $e_{t,k} = y_t - \hat{y}_{t,k}$, where y_t is the realized outcome at period `t` and $\hat{y}_{t,k}$ is model `k`’s point forecast. The variance of these errors is estimated over a rolling window:

- If `t < window_size`: variance is computed over observations `1:t` (expanding window).
- If `t >= window_size`: variance is computed over observations `max(1, t - window_size):t` (rolling window of size `window_size`).

For `t = 1` the variance of a single observation is undefined (`NA`). Estimated variances that are `NA`, zero, or negative are replaced by `1e-6` to ensure numerical stability in downstream computations. The benchmark column in the returned matrix is set to zero throughout.

The resulting `forecast_variance[t, k]` is intended for use as the variance parameter of a Gaussian predictive density $N(\hat{y}_{t,k}, \text{forecast\_variance}[t,k])$ for model `k` at period `t`, as required by `compute_klic` (Negative Log-Likelihood Score) and `compute_zp` (tail probability calibration).

Value

`matrix` of dimension `P x K_total` containing estimated variances. Columns correspond to the same forecasts as `forecast_matrix`; the benchmark column contains zeros.

Note

Correction from earlier package versions. Prior versions of this function computed forecast errors as $e_{t,k} = \text{benchmark}_t - \hat{y}_{t,k}$, i.e., the discrepancy between the benchmark’s point forecast and model `k`’s point forecast, rather than each model’s own error relative to the realized outcome. This quantity does not represent a valid predictive-uncertainty proxy: a model that disagrees with the benchmark by a constant, systematic offset would show spuriously low variance under the previous construction (interpreted as high confidence) regardless of that model’s actual forecasting accuracy, while a model that agrees with the benchmark on average but fluctuates around it would show spuriously high variance. The current version instead computes each model’s own rolling forecast-error variance relative to the realized outcome, which is the standard and statistically appropriate quantity for approximating model-specific predictive uncertainty under a Gaussian density assumption. This changes the `realized` argument from absent to required; calling code must be updated accordingly (see `compute_klic`, `compute_zp`, and `compute_kupiec` examples, which have been updated to supply `realized` explicitly).

See Also

compute_klic, compute_zp, compute_kupiec

Examples

```
data(metals)
realized <- metals[, 15]
forecast_variance <- estimate_forecast_variance(metals, realized = realized,
                                                benchmark_col = 15,
                                                window_size = 20)
head(forecast_variance)
```

---

estimate_long_run_covariance
Long-Run Covariance Estimator via Bartlett Kernel (HAC)

Description

Estimates the long-run covariance matrix using the Newey-West (1987) approach with a Bartlett kernel. Provides Heteroskedasticity and Autocorrelation Consistent (HAC) variance estimates used for studentizing Reality Check test statistics.

Usage

```
estimate_long_run_covariance(loss_differences, block_length)
```

Arguments

`loss_differences`
: A numeric matrix (P x K) of loss differences (benchmark loss minus forecast loss), where P is the number of forecast periods and K is the number of competing forecasts.

`block_length`
: integer. The truncation lag l for the Bartlett kernel, numerically set equal to the MBB block length used elsewhere in this package for consistency. In HAC estimation this controls how many autocovariance lags are included; in MBB it controls block size – both capture the same dependence horizon. A commonly used rule of thumb is $l \approx T^{1/3}$ (Politis & Romano, 1994). For P = 165 this gives approximately 5–6.

Details

Implements the Newey-West (1987) HAC covariance matrix estimator with Bartlett kernel weights $w_j = 1 - j/(l+1)$ for lags $j = 1, \ldots, l$, where l denotes the truncation lag (following the notation of Newey & West, 1987, and Politis & Romano, 1994), here set equal to `block_length`. This is essential for accounting for serial dependence in time-series forecast evaluations.

Value

A symmetric positive semi-definite matrix of dimensions K x K representing the estimated long-run covariance.

References

Newey, W. K., & West, K. D. (1987). A Simple Positive Semi-Definite Heteroskedasticity and Autocorrelation Consistent Covariance Matrix. *Econometrica*, 55(3), 703–708. doi:10.2307/1913610

Politis, D. N., & Romano, J. P. (1994). The stationary bootstrap. *Journal of the American Statistical Association*, 89(428), 1303–1313. doi:10.1080/01621459.1994.10476870

Examples

```
data(metals)
# metals: 165 x 15; columns 1-14 are competing forecasts, column 15 is the benchmark
# A small offset (+0.5) is added to the lagged benchmark to avoid degenerate zero
# loss differences when forecasts equal the realized value exactly (illustration only).
P <- nrow(metals)
K_total <- ncol(metals)
K <- K_total - 1 # 14 competing forecasts
realized <- c(metals[-1, K_total], metals[P, K_total]) + 0.5
benchmark_loss <- (metals[, K_total] - realized)^2
model_loss     <- (metals[, 1:K] - realized)^2
loss_diff      <- benchmark_loss - model_loss
lrc_result <- estimate_long_run_covariance(loss_diff, block_length = 5)
print(round(lrc_result[1:3, 1:3], 6))
```

---

extract_and_flatten_results_aggregated

Flatten Results for Export

Description

Prepares the comprehensive results list for export (e.g., to Microsoft Excel). Converts all htest objects and per-model results into flat data.frame objects, one per dataset.

Usage

```
extract_and_flatten_results_aggregated(comprehensive_results, alpha = 0.05)
```

Arguments

comprehensive_results	list output from run_comprehensive_erc_analysis.
alpha	Numeric significance level used to determine Reject_H0. Default is 0.05.

Value

list of data frames, one per dataset, each with columns: Model, Test_Type, P_Value, Statistic, Actual_Violations, Reject_H0.

See Also

run_comprehensive_erc_analysis, create_unified_summary, generate_comprehensive_report

Examples

```
data(metals)
realizations <- list(M = metals[, ncol(metals)])
prep_list    <- list(M = list(R_start = 0))
f_hat        <- list(list(NULL, NULL, metals))
names(f_hat) <- "M"
res <- run_comprehensive_erc_analysis(
  data_list_prepared = prep_list,
  mods_matrix        = matrix(0),
  alpha_grid         = 0.05,
  window_size        = 20,
  y_hat_all          = f_hat,
  y_raw              = realizations,
  block_length       = 5,
  n_boot             = 10,
  zp_quantile        = 0.05
)
excel_data <- extract_and_flatten_results_aggregated(res$aggregate_results, alpha = 0.05)
head(excel_data[[1]])
```

---

generate_comprehensive_report
Generate Comprehensive Markdown Report

Description

Generates an automatic summary report in Markdown format covering all error metrics (MSE, MAE, MASE) and distributional tests (ZP, KLIC, Kupiec). For the ZP and KLIC sections, the report lists superior competing forecasts (i.e. those whose forecasts are found to be more accurate than the benchmark forecast) or states that no superior forecasts were found. For the Kupiec section, forecasts with correct VaR coverage (Reject_H0 == FALSE) are listed, or a message is printed if none passed.

Technical Abbreviations:

- **WRC:** White's Reality Check (White, 2000). Tests whether any competing forecast has lower expected loss than the benchmark forecast; controls family-wise error rate.
- **SPA:** Superior Predictive Ability test (Hansen, 2005). A studentized extension of WRC with improved power that corrects for irrelevant forecasts.

- **CPA:** Conditional Predictive Ability test (Giacomini & White, 2006). Tests whether loss differentials are predictable by a conditioning variable.
- **ZP:** Quantile Loss test (Corradi & Swanson, 2006). Evaluates whether any competing forecast better calibrates the probability of a left-tail event defined by the `zp_quantile` threshold.
- **KLIC:** Kullback-Leibler Information Criterion based density test (Corradi & Swanson, 2006). Selects the forecast whose predictive density is closest to the true density in terms of KLIC distance, evaluated via Negative Log-Likelihood Scores (NLS) under a Gaussian predictive density assumption.
- **CRPS:** Continuous Ranked Probability Score (Gneiting & Raftery, 2007). Jointly rewards calibration and sharpness of the predictive distribution.
- **UC:** Kupiec Unconditional Coverage test (Kupiec, 1995).
- **MSE:** Mean Squared Error.
- **MAE:** Mean Absolute Error.
- **MASE:** Mean Absolute Scaled Error.

Usage

```
generate_comprehensive_report(
  summary_df,
  zp_models_df,
  klic_models_df,
  kupiec_models_df,
  dataset_name,
  alpha = 0.05
)
```

Arguments

`summary_df`	`data.frame` unified summary from `create_unified_summary`.
`zp_models_df`	`data.frame` with columns `Model`, `P_Value`, and `Dataset`. Models with `P_Value` `<= alpha` are considered superior and listed in the report; if none are found, a message \"No superior models found\" is printed. Pass all competing models to ensure complete and unbiased reporting.
`klic_models_df`	`data.frame` with columns `Model`, `P_Value`, and `Dataset`. Models with `P_Value` `<= alpha` are considered superior and listed in the report; if none are found, a message \"No superior models found\" is printed. Pass all competing models to ensure complete and unbiased reporting.
`kupiec_models_df`	`data.frame` with columns `Model`, `Reject_H0`, and `Dataset`. Contains all models tested under the Kupiec UC test. Models with `Reject_H0 == FALSE` are considered to have correct VaR coverage and are listed in the report; models with `Reject_H0 == TRUE` are excluded. Pass all competing models to ensure complete and unbiased reporting.
`dataset_name`	`character` the name of the dataset to be used in the report header.
`alpha`	`numeric` significance level (default 0.05).

Value

character string in Markdown format.

References

White, H. (2000). A reality check for data snooping. *Econometrica*, 68(5), 1097–1126. doi:10.1111/14680262.00152

Hansen, P. R. (2005). A Test for Superior Predictive Ability. *Journal of Business & Economic Statistics*, 23(4), 365–380. doi:10.1198/073500105000000063

Giacomini, R., & White, H. (2006). Tests of Conditional Predictive Ability. *Econometrica*, 74(6), 1545–1578. doi:10.1111/j.14680262.2006.00718.x

Corradi, V., & Swanson, N. R. (2006). Predictive density and conditional confidence interval accuracy tests. *Journal of Econometrics*, 135(1–2), 187–228. doi:10.1016/j.jeconom.2005.07.026

Corradi, V., & Swanson, N. R. (2011). The White Reality Check and some of its recent extensions. In *Festschrift in honor of Halbert L. White*.

Gneiting, T., & Raftery, A. E. (2007). Strictly Proper Scoring Rules, Prediction, and Estimation. *Journal of the American Statistical Association*, 102(477), 359–378. doi:10.1198/016214506000001437

Kupiec, P. H. (1995). Techniques for Verifying the Accuracy of Risk Measurement Models. *The Journal of Derivatives*, 3(2), 173–184. doi:10.3905/jod.1995.407942

Politis, D. N., & Romano, J. P. (1994). The stationary bootstrap. *Journal of the American Statistical Association*, 89(428), 1303–1313. doi:10.1080/01621459.1994.10476870

Diebold, F. X., & Mariano, R. S. (1995). Comparing Predictive Accuracy. *Journal of Business & Economic Statistics*, 13(3), 253–263. doi:10.1080/07350015.1995.10524599

Examples

```
data(metals)
ds_name       <- "Dataset1"
prep_list     <- list(Dataset1 = list(R_start = 0))
realizations  <- list(Dataset1 = metals[, ncol(metals)])
f_hat         <- list(list(NULL, NULL, metals))
names(f_hat)  <- ds_name
res <- run_comprehensive_erc_analysis(
  data_list_prepared = prep_list,
  mods_matrix        = matrix(0),
  alpha_grid         = 0.05,
  window_size        = 20,
  y_hat_all          = f_hat,
  y_raw              = realizations,
  block_length       = 5,
  n_boot             = 10,
  zp_quantile        = 0.05
)
unified_summ <- create_unified_summary(res$aggregate_results)

zp_table     <- data.frame(Model   = colnames(metals)[1:(ncol(metals) - 1)],
                            P_Value = 0.1, Dataset = ds_name)
klic_table   <- zp_table
```

```
kupiec_table <- data.frame(Model     = colnames(metals)[1:(ncol(metals) - 1)],
                           Reject_H0 = rep(FALSE, ncol(metals) - 1),
                           Dataset   = ds_name)

report <- generate_comprehensive_report(
  summary_df       = unified_summ$summary,
  zp_models_df     = zp_table,
  klic_models_df   = klic_table,
  kupiec_models_df = kupiec_table,
  dataset_name     = ds_name
)
cat(report)
```

`kullback_leibler_test` *Kullback-Leibler Information Criterion (KLIC) Test*

Description

Implements the Reality Check using Negative Log-Likelihood Scores (NLS) to evaluate predictive densities in terms of their Kullback-Leibler divergence from the true density. Based on Corradi & Swanson (2006).

- **H0:** $\max_k E[\log f_1(y_t) - \log f_k(y_t)] \leq 0$ – no competing forecast achieves a higher average log-likelihood (lower KLIC distance) than the benchmark density f_1.
- **H1:** At least one competing forecast achieves strictly higher average log-likelihood than the benchmark.

Usage

```
kullback_leibler_test(
  log_likelihood_differences,
  block_length,
  num_bootstrap_replications,
  alpha = 0.05
)
```

Arguments

`log_likelihood_differences`
: A `numeric` matrix (`P x K`) of Negative Log-Likelihood Score (NLS) differences: benchmark NLS minus forecast NLS. A positive entry means the forecast's density assigns higher probability to the observed outcome than the benchmark density does.

`block_length`
: `integer`. The block length for MBB and HAC estimation. A commonly used rule of thumb is `block_length` $\approx T^{1/3}$, where T is the number of observations (Politis & Romano, 1994). For `P = 165`, this gives approximately 5–6.

num_bootstrap_replications
: integer number of MBB bootstrap replications. Default 999; see Davidson & MacKinnon (2000).

alpha
: numeric. The significance level (default 0.05).

Details

The KLIC between the true density f_0 and a forecast density f_k is $E[\log f_0(y) - \log f_k(y)]$. Minimising KLIC is equivalent to maximising expected log-likelihood. This test therefore selects the forecast with the smallest KLIC distance from the true density. **Lower NLS values are better**: a forecast with lower NLS assigns higher average probability to events that actually occurred (Corradi & Swanson, 2006).

The NLS loss matrix is constructed via compute_klic assuming normal predictive densities parameterised by a point forecast and a rolling-window standard deviation. The test statistic is the maximum studentized mean NLS differential, with p-values obtained via MBB following the SPA bootstrap of Hansen (2005).

Value

An object of class "htest". A small p-value (below alpha) leads to rejection of H0, indicating that at least one competing forecast has a lower Kullback-Leibler distance from the true density than the benchmark. Failure to reject H0 suggests no forecast provides significantly better density fit than the benchmark.

References

Corradi, V., & Swanson, N. R. (2006). Predictive density and conditional confidence interval accuracy tests. *Journal of Econometrics*, 135(1–2), 187–228. doi:10.1016/j.jeconom.2005.07.026

Corradi, V., & Swanson, N. R. (2011). The White Reality Check and some of its recent extensions. In *Festschrift in honor of Halbert L. White*.

Davidson, R., & MacKinnon, J. G. (2000). Bootstrap tests: How many bootstraps? *Econometric Reviews*, 19(1), 55–68. doi:10.1080/07474930008800459

Hansen, P. R. (2005). A Test for Superior Predictive Ability. *Journal of Business & Economic Statistics*, 23(4), 365–380. doi:10.1198/073500105000000063

Politis, D. N., & Romano, J. P. (1994). The stationary bootstrap. *Journal of the American Statistical Association*, 89(428), 1303–1313. doi:10.1080/01621459.1994.10476870

Examples

```
data(metals)
# metals: 165 x 15; columns 1-14 are competing forecasts, column 15 is the benchmark
benchmark_col      <- 15
P                  <- nrow(metals)
K_total            <- ncol(metals)
K                  <- K_total - 1 # 14 competing forecasts
realized           <- metals[, benchmark_col]
forecast_variance  <- estimate_forecast_variance(metals, realized = realized,
                        benchmark_col = benchmark_col, window_size = 20)
```

```
comp_cols          <- setdiff(seq_len(K_total), benchmark_col)
forecast_sd_models <- sqrt(forecast_variance[, comp_cols])
nls_loss  <- compute_klic(metals, forecast_sd_models, benchmark_col = K_total)
nls_diff  <- nls_loss[, K_total] - nls_loss[, comp_cols]
kullback_leibler_test(nls_diff, block_length = 5, num_bootstrap_replications = 50)
```

---

mbb_resample_data *Moving Block Bootstrap (MBB) Resampler*

Description

Generates a bootstrap resample of a time series matrix using the Moving Block Bootstrap (MBB) method of Kunsch (1989).

Usage

```
mbb_resample_data(data_series, block_length)
```

Arguments

`data_series`	`matrix` where rows are observations (`P`) and columns are variables (`K`).
`block_length`	`integer` block length for the resampler. Overlapping blocks of this length are sampled with replacement. A commonly used rule of thumb is `block_length` $\approx T^{1/3}$ (Politis & Romano, 1994). For `P = 165`, this gives approximately 5–6.

Details

Resamples overlapping blocks of `block_length` consecutive rows with replacement, then concatenates them to produce a bootstrap sample of the same length `P` as the original series. This preserves the short-run autocorrelation structure of the data, which is required for valid inference in the Reality Check and SPA-type tests.

Value

`matrix` of bootstrap resampled data with the same dimensions as `data_series`.

References

Kunsch, H. R. (1989). The jackknife and the bootstrap for general stationary observations. *The Annals of Statistics*, 17(3), 1217–1241. doi:10.1214/aos/1176347265

Politis, D. N., & Romano, J. P. (1994). The stationary bootstrap. *Journal of the American Statistical Association*, 89(428), 1303–1313. doi:10.1080/01621459.1994.10476870

Corradi, V., & Swanson, N. R. (2011). The White Reality Check and some of its recent extensions. In *Festschrift in honor of Halbert L. White*.

Liu, R. Y., & Singh, K. (1992). Moving blocks jackknife and bootstrap capture weak dependence. In R. LePage & L. Billard (Eds.), *Exploring the Limits of Bootstrap* (pp. 225–248). Wiley.

Examples

```
data(metals)
# metals: 165 x 15; columns 1-3 are the first three competing forecasts
mbb_resample_data(metals[, 1:3], block_length = 5)
```

`metals` *Forecasts of Base Metals Prices*

Description

A dataset of 165 observations of base metals price indices. It includes point forecasts from 15 predictive models (Bayesian and shrinkage-based). This dataset is based on the research methodology and benchmarks presented in Drachal and Jedrzejewska (2025).

Usage

```
metals
```

Format

A matrix with 165 rows and 15 columns:

- `metals[,1:14]`: Point forecasts from 14 competing models (including variations of Bayesian Dynamic Mixture Models (BDMM), Dynamic Model Averaging (DMA), Time-Varying Parameters (TVP), as well as Bayesian LASSO, Bayesian RIDGE, Autoregressive (AR1), and Linear Regression).
- `metals[,15]`: `HA` - Historical Average (Benchmark model).

Details

Monthly World Bank base metals price index (2010 = 100, USD), including aluminium, copper, lead, nickel, tin and zinc, forecasts for the period from 03/2011 to 11/2024.

Source

Drachal, K. (2025). Base metals price index forecasts. *figshare*. doi:10.6084/m9.figshare.28382480.v1

References

Drachal, K., & Jedrzejewska, J. (2025). Forecasting Base Metals Prices: A Comparison of Various Bayesian-Based Methods. *Sinteza 2025*, 175–183. doi:10.15308/Sinteza2025175183

World Bank. (2025). Commodity markets. `https://www.worldbank.org/en/research/commodity-markets`

Examples

```
data(metals)
head(metals)
plot(metals[, 1], type = "l", main = "Competing Model vs Benchmark")
lines(metals[, ncol(metals)], col = "red", lty = 2)
```

plot_cumulative_loss *Plot Cumulative Loss Differences*

Description

Generates a time-series plot of cumulative squared error (MSE) loss differences between each competing forecast and the benchmark. A positive value at time t means the competing forecast has accumulated lower squared errors than the benchmark up to that point (i.e., the forecast is outperforming the benchmark cumulatively).

Usage

```
plot_cumulative_loss(data_matrix, benchmark_col = ncol(data_matrix))
```

Arguments

`data_matrix` Matrix or data frame of dimension `P x K_total`, where rows are time periods and columns are **pre-computed forecast errors** $(y_t - \hat{y}_{k,t})$ for each forecast including the benchmark. **Do not pass raw forecasts**: the function squares column values directly, so passing raw forecasts produces cumulative sums of squared forecast levels, not cumulative MSE differences.

`benchmark_col` Index or name of the benchmark column. Defaults to the last column.

Details

The cumulative loss difference for forecast k at time t is:

$$\mathrm{CLD}_{k,t} = \sum_{s=1}^{t} \left(e^2_{\text{bench},s} - e^2_{k,s}\right)$$

where $e_{k,s} = y_s - \hat{y}_{k,s}$ is the forecast error of forecast k at period s. The function squares the values in `data_matrix` directly, so **pre-computed forecast errors** (not raw forecasts) must be passed for the result to represent cumulative MSE differences.

A positive $\mathrm{CLD}_{k,t}$ means forecast k has accumulated lower squared errors than the benchmark up to period t. A negative value means the benchmark has been more accurate up to that point.

Value

A `ggplot` object. The y-axis shows the cumulative MSE difference; forecasts above zero at the right edge have outperformed the benchmark over the full evaluation window. The top 2 and bottom 2 forecasts (by final cumulative loss difference) are labelled directly on the plot.

See Also

`white_reality_check`, `plot_performance_metrics`

Examples

```
data(metals)
P        <- nrow(metals)
K_total  <- ncol(metals)
realized <- c(metals[-1, K_total], metals[P, K_total])
errors   <- sweep(as.matrix(metals), 1, realized, "-")
p <- plot_cumulative_loss(errors, benchmark_col = K_total)
print(p)
```

`plot_density_forecast` *Plot Density Forecast*

Description

Generates a kernel density plot of a predictive distribution, highlighting the point forecast and a symmetric credible interval at a specified level. The predictive distribution can be constructed by treating the cross-sectional spread of model forecasts at a given period as a proxy for forecast uncertainty (see `compute_crps` for the centring convention).

Usage

```
plot_density_forecast(
  full_distribution,
  point_forecast,
  title = "Density Forecast",
  ci_level = 0.9
)
```

Arguments

`full_distribution`	Numeric vector of forecast samples drawn from the predictive distribution (e.g., forecasts from all models at one time period, optionally centred and shifted as in `compute_crps`).
`point_forecast`	Single numeric value: the point forecast to highlight (e.g., the mean or median of `full_distribution`, or one model's forecast).
`title`	Character string for the plot title. Default is `"Density Forecast"`.
`ci_level`	Numeric confidence interval level strictly between 0 and 1. Default is 0.90, producing lower and upper quantiles at `(1 - ci_level) / 2` and `1 - (1 - ci_level) / 2`.

Details

To build a pseudo-predictive distribution from the `metals` dataset, centre the cross-sectional model forecasts at period `t` around their mean, following the same convention used in `compute_crps`: `dist_t <- forecasts[t, ] - forecasts[t, k] + mean(forecasts[t, ])`. This preserves cross-sectional spread while recentring on the cross-sectional mean rather than on model `k`'s own point forecast.

Value

A `ggplot` object. The plot shows a kernel density curve (blue fill), a red dashed vertical line at `point_forecast`, and orange dotted vertical lines at the lower and upper quantiles of the credible interval. The subtitle reports the numeric bounds of the interval.

See Also

`compute_crps`, `run_comprehensive_erc_analysis`'

Examples

```
data(metals)
t       <- 100
K       <- ncol(metals) - 1
dist_t  <- as.numeric(metals[t, 1:K]) - metals[t, 7] +
            mean(as.numeric(metals[t, 1:K]))
pt_fcst <- mean(as.numeric(metals[t, 1:K]))
plot_density_forecast(dist_t, pt_fcst,
                      title    = "Predictive Density at t = 100",
                      ci_level = 0.90)
```

`plot_performance_metrics` *Plot Performance Metrics Comparison*

Description

Generates a grid of scatter plots for Root Mean Squared Error (RMSE), Normalized Root Mean Squared Error (N-RMSE), Mean Absolute Error (MAE), and Mean Absolute Scaled Error (MASE) plotted against the Equally Weighted Risk Contribution (ERC) Weight of each competing forecast. These metrics are used for visualization only and are computed directly from forecast errors; they are distinct from the loss functions used in the WRC/SPA/CPA hypothesis tests (see `run_comprehensive_erc_analysis`).

Each panel shows:

- **Points**: one per model. The radius (size) of each circle is proportional to the model's *Risk Contribution*, defined as $\text{RMSE}_k \times \text{ERC Weight}_k$. Larger circles indicate forecasts that contribute more to the portfolio-level risk.
- **Dashed line**: an OLS regression line of the error metric (x-axis) on the ERC Weight (y-axis), showing whether higher-weighted forecasts tend to have systematically lower or higher error.

Note on MASE scaling: The MASE denominator used here is `mean(abs(diff(benchmark)))`, where `benchmark` is the benchmark forecast column extracted from `forecast_matrix`. This differs from the scaling used in `run_comprehensive_erc_analysis`, where the denominator is `mean(abs(diff(realizations_ra` computed from the actual realised values. The two denominators coincide when the benchmark is a random-walk or historical-average forecast whose one-step-ahead forecasts equal the lagged realised value, but will diverge otherwise. The `plot_performance_metrics` function does not accept a separate realisation vector, so the benchmark forecast series is used as a proxy for the naive forecast scale. MASE values produced by this function are therefore intended for *visual comparison across forecasts only* and should not be directly compared to MASE values from the hypothesis tests in `run_comprehensive_erc_analysis`.

Usage

```
plot_performance_metrics(
  forecast_matrix,
  weights = NULL,
  benchmark_col = ncol(forecast_matrix)
)
```

Arguments

`forecast_matrix`
: Matrix or data frame of dimension `P` x `K_total`, where `P` is the number of forecast periods, columns `1` to `K_total - 1` are competing model forecasts, and the last column (or `benchmark_col`) is the benchmark.

`weights`
: Optional numeric vector of length `K_total - 1` giving ERC weights for each competing forecast. If `NULL` (default), equal weights $1/K$ are used.

`benchmark_col`
: Index or name of the benchmark column. Defaults to the last column.

Value

A `gtable` object produced by `grid.arrange` containing four panels arranged in a 2x2 grid: RMSE (top-left), N-RMSE (top-right), MAE (bottom-left), MASE (bottom-right). Each panel is a `ggplot` object and can be extracted individually if needed.

Equally Weighted Risk Contribution (ERC) Weight

ERC weights are portfolio weights assigned so that every competing forecast contributes an equal share to the total portfolio risk (measured here by forecast error dispersion). Formally, weights w_k are chosen so that $w_k \cdot \sigma_k = c$ for all k, where σ_k is a measure of forecast k's risk and $c = \frac{1}{K}\sum_{k=1}^{K} w_k \sigma_k$ is the common per-forecast risk budget determined endogenously by the equal-contribution constraint (Maillard et al., 2010). When `weights` are supplied by the user, they are treated as pre-computed ERC weights and normalised to sum to one. When `weights = NULL`, equal weights $1/K$ are used as a baseline.

References

Maillard, S., Roncalli, T., & Teïletche, J. (2010). The Properties of Equally Weighted Risk Contributions Portfolios. *The Journal of Portfolio Management*, 36(4), 60–70. doi:10.3905/jpm.2010.36.4.060

See Also

`run_comprehensive_erc_analysis`, `plot_cumulative_loss`

Examples

```
data(metals)
K_models       <- ncol(metals) - 1
custom_weights <- (K_models:1) / sum(K_models:1)
plot_performance_metrics(metals, weights = custom_weights, benchmark_col = 15)
```

`reality_check_zp_test` *ZP Quantile Loss Reality Check Test*

Description

Implements the studentized Reality Check test for comparing predictive densities based on the ZP quantile loss function of Corradi & Swanson (2006). The test evaluates whether any competing forecast more accurately predicts the probability of the outcome falling below a specified threshold than the benchmark forecast.

- **H0:** $\max_k E[\mu_1^2(u) - \mu_k^2(u)] \leq 0$ – no competing forecast has a lower expected squared probability forecast error at threshold u than the benchmark (Corradi & Swanson, 2006, eq. 7).
- **H1:** At least one competing forecast has lower expected ZP-loss than the benchmark.

Usage

```
reality_check_zp_test(
  zp_loss_differences,
  block_length,
  num_bootstrap_replications,
  alpha = 0.05
)
```

Arguments

`zp_loss_differences`
: A `numeric` matrix (`P x K`) of ZP-loss differences (benchmark ZP-loss minus forecast ZP-loss), where a positive entry means the competing forecast outperforms the benchmark forecast at that period.

`block_length`
: `integer`. The block length for MBB and HAC estimation. A commonly used rule of thumb is `block_length` $\approx T^{1/3}$, where T is the number of observations (Politis & Romano, 1994). For `P = 165`, this gives approximately 5–6.

`num_bootstrap_replications`
: `integer` number of MBB bootstrap replications. Default `999`; see Davidson & MacKinnon (2000).

`alpha`
: `numeric`. The significance level (default `0.05`).

Details

The ZP loss for forecast k at period t is

$$ZP_{t,k} = \left(\mathbf{1}(y_t \le \tau) - F_k(\tau \mid \hat{y}_{t,k}, \hat{\sigma}_{t,k})\right)^2$$

where τ is a threshold, $F_k(\cdot)$ is the forecast's predictive CDF at period t, and $\mathbf{1}(y_t \le \tau)$ is the indicator for a tail event. Interpretively, the threshold τ defines a tail event of interest (e.g., the 5th percentile of realizations). The ZP loss penalises the squared difference between the predicted probability of this event and whether it actually occurred. A forecast with **lower** ZP loss more accurately calibrates the left-tail probability. The threshold is typically set to a quantile of the realized series (e.g., `quantile(realized, 0.05)`); a lower threshold focuses the test more sharply on extreme left-tail events.

The benchmark is treated as a degenerate (point-mass) predictive distribution with $\hat{\sigma} = 10^{-6}$, which is a conservative choice ensuring the benchmark's ZP loss approximates the Brier score for the tail indicator.

Two p-values are returned: `p_consistent` and `p_conservative`, analogous to those in `superior_predictive_ability_te`

Value

An object of class `"htest"`. Lower p-values indicate that at least one competing forecast is significantly better calibrated in the left-tail than the benchmark forecast. Also contains `p_consistent` and `p_conservative`. Failure to reject H0 suggests no forecast provides significantly better density fit than the benchmark forecast.

References

Corradi, V., & Swanson, N. R. (2006). Predictive density and conditional confidence interval accuracy tests. *Journal of Econometrics*, 135(1–2), 187–228. doi:10.1016/j.jeconom.2005.07.026

Corradi, V., & Swanson, N. R. (2011). The White Reality Check and some of its recent extensions. In *Festschrift in honor of Halbert L. White*.

Davidson, R., & MacKinnon, J. G. (2000). Bootstrap tests: How many bootstraps? *Econometric Reviews*, 19(1), 55–68. doi:10.1080/07474930008800459

Hansen, P. R. (2005). A Test for Superior Predictive Ability. *Journal of Business & Economic Statistics*, 23(4), 365–380. doi:10.1198/073500105000000063

Politis, D. N., & Romano, J. P. (1994). The stationary bootstrap. *Journal of the American Statistical Association*, 89(428), 1303–1313. doi:10.1080/01621459.1994.10476870

Examples

```
data(metals)
# metals: 165 x 15; columns 1-14 are competing forecasts, column 15 is the benchmark
benchmark_col      <- 15
P                  <- nrow(metals)
K_total            <- ncol(metals)
K                  <- K_total - 1 # 14 competing forecasts
realized           <- metals[, benchmark_col]
forecast_variance  <- estimate_forecast_variance(metals, realized = realized,
                        benchmark_col = benchmark_col, window_size = 20)
```

```
comp_cols          <- setdiff(seq_len(K_total), benchmark_col)
forecast_sd_models <- sqrt(forecast_variance[, comp_cols])
threshold_val      <- quantile(metals[, K_total], 0.05)
zp_loss  <- compute_zp(metals, forecast_sd_models,
                       threshold = threshold_val, benchmark_col = K_total)
zp_diff  <- zp_loss[, K_total] - zp_loss[, comp_cols]
reality_check_zp_test(zp_diff, block_length = 5, num_bootstrap_replications = 50)
```

`run_comprehensive_erc_analysis`
Run Comprehensive Forecast Evaluation Analysis

Description

Runs a full suite of reality check and density forecast evaluation tests on one or more datasets. Tests included: White's Reality Check (WRC), Superior Predictive Ability (SPA), Conditional Predictive Ability (CPA), ZP Quantile Loss test, Kullback-Leibler Information Criterion (KLIC) test, Kupiec Unconditional Coverage (UC) test, CRPS-based CDF comparison, and per-model Diebold-Mariano statistics across four error metrics (MSE, MAE, MASE).

Technical Abbreviations:

- **WRC:** White's Reality Check (White, 2000). Tests whether any competing forecast has lower expected loss than the benchmark; controls family-wise error rate.
- **SPA:** Superior Predictive Ability test (Hansen, 2005). A studentized extension of WRC with improved power that corrects for irrelevant forecasts.
- **CPA:** Conditional Predictive Ability test (Giacomini & White, 2006). Tests whether loss differentials are predictable by a conditioning variable.
- **ZP:** Quantile Loss test (Corradi & Swanson, 2006). Evaluates whether any competing forecast better calibrates the probability of a left-tail event defined by the `zp_quantile` threshold.
- **KLIC:** Kullback-Leibler Information Criterion based density test (Corradi & Swanson, 2006). Selects the forecast whose predictive density is closest to the true density in terms of KLIC distance, evaluated via Negative Log-Likelihood Scores (NLS) under a Gaussian predictive density assumption.
- **CRPS:** Continuous Ranked Probability Score (Gneiting & Raftery, 2007). Jointly rewards calibration and sharpness of the predictive distribution.
- **UC:** Kupiec Unconditional Coverage test (Kupiec, 1995).
- **MSE:** Mean Squared Error.
- **MAE:** Mean Absolute Error.
- **MASE:** Mean Absolute Scaled Error.

Usage

```
run_comprehensive_erc_analysis(
  data_list_prepared,
  mods_matrix,
  alpha_grid,
  window_size,
  y_hat_all,
  y_raw,
  block_length = 5,
  n_boot = 999,
  zp_quantile = 0.05,
  n_crps_samples = 10,
  benchmark_col = NULL
)
```

Arguments

`data_list_prepared`
: Named list of prepared data frames, one per dataset. Each element must be a list containing at least a field `R_start`: a non-negative integer specifying how many observations to skip from the start of `y_raw` before aligning with the forecast matrix. Set `R_start = 0` to use all available observations. This is used as a warm-up offset when the raw series is longer than the forecast evaluation window.

`mods_matrix`
: A legacy placeholder matrix retained for interface compatibility with external pipelines in which forecasts were previously defined as a parameter matrix. Its contents are not read or used anywhere in this function — the actual forecast structure is derived entirely from the forecast matrices supplied in `y_hat_all`. Pass `matrix(0)` when calling the function directly.

`alpha_grid`
: Numeric scalar or vector of significance levels. Only the first element (`alpha_grid[1]`) is used as the significance level for all tests.

`window_size`
: Integer window size for rolling variance estimation passed to `estimate_forecast_variance`.

`y_hat_all`
: Named list of forecast results, one per dataset. Each element must be a list of length at least 3, where the third element (`[[3]]`) is a numeric matrix of dimension `P x K_total`: columns `1:(K_total-1)` are the competing model forecasts and column `K_total` (or the column indicated by `benchmark_col`) is the benchmark forecast.

`y_raw`
: Named list of raw realized value vectors, one per dataset. Each element is a numeric vector whose length must be at least `R_start + P`.

`block_length`
: Integer block length for Moving Block Bootstrap (MBB) used in WRC, SPA, CPA, ZP, and KLIC tests. Default is 5. A commonly used rule of thumb is $T^{1/3}$ (Politis & Romano, 1994); for `P = 165` this gives approximately 5–6.

`n_boot`
: `integer` number of MBB bootstrap replications. Default `999`; see Davidson & MacKinnon (2000).

`zp_quantile`
: Numeric quantile level used to define the left-tail threshold τ for the ZP test, computed as `quantile(realizations, zp_quantile)`. Default is 0.05 (5th percentile).

`n_crps_samples` Integer number of Monte Carlo samples drawn from the Gaussian predictive distribution $N(\hat{y}_{k,t}, \hat{\sigma}^2_{k,t})$ to approximate the Continuous Ranked Probability Score (CRPS) for each forecast and time period.Default is 10 (fast, suitable for examples only). For reliable results use at least 500; for publication-quality estimates use 1000 or more. Higher values reduce Monte Carlo variance but increase computation time linearly.

`benchmark_col` Index or name of the benchmark column in the forecast matrix. Defaults to the last column (`NULL`).

Value

`list` containing:

- `aggregate_results`: Named list of test results per dataset. Each dataset element contains named `htest` objects for each test and metric combination, plus `VaR_Backtests` (a list of per-model Kupiec `htest` objects).
- `per_model_results`: Named list of per-model Diebold-Mariano statistics per dataset, returned as `data.frame` objects from `compute_per_model_statistics`.

References

Davidson, R., & MacKinnon, J. G. (2000). Bootstrap tests: How many bootstraps? *Econometric Reviews*, 19(1), 55–68. doi:10.1080/07474930008800459

White, H. (2000). A reality check for data snooping. *Econometrica*, 68(5), 1097–1126. doi:10.1111/14680262.00152

Hansen, P. R. (2005). A Test for Superior Predictive Ability. *Journal of Business & Economic Statistics*, 23(4), 365–380. doi:10.1198/073500105000000063

Giacomini, R., & White, H. (2006). Tests of Conditional Predictive Ability. *Econometrica*, 74(6), 1545–1578. doi:10.1111/j.14680262.2006.00718.x

Corradi, V., & Swanson, N. R. (2006). Predictive density and conditional confidence interval accuracy tests. *Journal of Econometrics*, 135(1–2), 187–228. doi:10.1016/j.jeconom.2005.07.026

Corradi, V., & Swanson, N. R. (2011). The White Reality Check and some of its recent extensions. In *Festschrift in honor of Halbert L. White*.

Gneiting, T., & Raftery, A. E. (2007). Strictly Proper Scoring Rules, Prediction, and Estimation. *Journal of the American Statistical Association*, 102(477), 359–378. doi:10.1198/016214506000001437

Kupiec, P. H. (1995). Techniques for Verifying the Accuracy of Risk Measurement Models. *The Journal of Derivatives*, 3(2), 173–184. doi:10.3905/jod.1995.407942

Politis, D. N., & Romano, J. P. (1994). The stationary bootstrap. *Journal of the American Statistical Association*, 89(428), 1303–1313. doi:10.1080/01621459.1994.10476870

Diebold, F. X., & Mariano, R. S. (1995). Comparing Predictive Accuracy. *Journal of Business & Economic Statistics*, 13(3), 253–263. doi:10.1080/07350015.1995.10524599

See Also

`create_unified_summary`, `generate_comprehensive_report`, `white_reality_check`, `superior_predictive_ability`, `white_reality_check_conditional`, `reality_check_zp_test`, `kullback_leibler_test`, `compute_kupiec`

Examples

```
data(metals)
ds_name           <- "Dataset1"
realizations_list <- list(Dataset1 = metals[, ncol(metals)])
prep_list         <- list(Dataset1 = list(R_start = 0))
forecasts_list    <- list(list(NULL, NULL, metals))
names(forecasts_list) <- ds_name

res <- run_comprehensive_erc_analysis(
  data_list_prepared = prep_list,
  mods_matrix        = matrix(0),
  alpha_grid         = c(0.05),
  window_size        = 20,
  y_hat_all          = forecasts_list,
  y_raw              = realizations_list,
  n_boot             = 99,
  n_crps_samples     = 5
)
print(res$aggregate_results$Dataset1$White_Reality_Check_MSE)
```

`superior_predictive_ability_test`
Superior Predictive Ability (SPA) Test

Description

Implements the Hansen (2005) Superior Predictive Ability (SPA) test, a studentized extension of White's (2000) Reality Check that corrects for the inclusion of irrelevant (poor) forecasts to reduce conservatism.

Hypotheses:

- **H0:** $\max_k E[g(u_{0,t}) - g(u_{k,t})] \leq 0$ – no competing forecast produces strictly lower expected loss than the benchmark forecast.
- **H1:** At least one competing forecast has strictly lower expected loss than the benchmark forecast.

Usage

```
superior_predictive_ability_test(
  loss_differences,
  block_length,
  num_bootstrap_replications,
  alpha
)
```

Arguments

`loss_differences`
: A `numeric` matrix (P x K) of loss differences (benchmark loss minus forecast loss).

`block_length`
: `integer`. The block length for MBB and HAC estimation. A commonly used rule of thumb is $T^{1/3}$ (Politis & Romano, 1994). For P = 165, this gives approximately 5–6.

`num_bootstrap_replications`
: `integer` number of MBB bootstrap replications. Default 999; use at least 999 for reliable inference (Davidson & MacKinnon, 2000).

`alpha`
: `numeric`. The significance level (default 0.05).

Details

The SPA statistic studentizes each mean loss differential by its HAC standard deviation (estimated via `estimate_long_run_covariance`), then takes the maximum across forecasts. Two p-values are returned, corresponding to two choices of the null distribution (Hansen, 2005, Section 3):

- `p_consistent`: uses the sample-dependent null estimator $\hat{\mu}^c$, which recentres the bootstrap statistic at the sample mean $\bar{d}_k$ for each forecast. This is the **recommended** p-value.
- `p_conservative`: uses the Least Favourable Configuration (LFC) $\hat{\mu}^u = 0$ for all forecasts – equivalent to White's (2000) Reality Check bootstrap, where no recentring is applied. This provides an upper bound on the true p-value and is always $\geq$ `p_consistent`.

Value

An object of class "htest". Additionally contains `p_consistent` and `p_conservative` for the two SPA bootstrap variants.

References

Hansen, P. R. (2005). A Test for Superior Predictive Ability. *Journal of Business & Economic Statistics*, 23(4), 365–380. doi:10.1198/073500105000000063

Corradi, V., & Swanson, N. R. (2011). The White Reality Check and some of its recent extensions. In *Festschrift in honor of Halbert L. White*.

Davidson, R., & MacKinnon, J. G. (2000). Bootstrap tests: How many bootstraps? *Econometric Reviews*, 19(1), 55–68. doi:10.1080/07474930008800459

Kunsch, H. R. (1989). The jackknife and the bootstrap for general stationary observations. *The Annals of Statistics*, 17(3), 1217–1241. doi:10.1214/aos/1176347265

Examples

```
data(metals)
# metals: 165 x 15; columns 1-14 are competing forecasts, column 15 is the benchmark
# A small offset (+0.5) is added to the lagged benchmark to avoid degenerate zero
# loss differences when forecasts equal the realized value exactly (illustration only).
P <- nrow(metals)
K_total <- ncol(metals)
```

```
K <- K_total - 1 # 14 competing forecasts
realized       <- c(metals[-1, K_total], metals[P, K_total]) + 0.5
benchmark_loss <- (metals[, K_total] - realized)^2
model_loss     <- (metals[, 1:K] - realized)^2
loss_diff      <- benchmark_loss - model_loss
res <- superior_predictive_ability_test(loss_diff, block_length = 5,
                                        num_bootstrap_replications = 50,
                                        alpha = 0.05)
print(res)
```

---

white_reality_check *White's Reality Check (WRC)*

Description

Implements White's (2000) Reality Check (WRC) for comparing forecast accuracy of multiple competing forecasts against a benchmark forecast based on mean loss differences. The test controls the family-wise error rate across all forecast comparisons simultaneously, avoiding data-snooping bias.

Hypotheses:

- **H0:** $\max_k E[g(u_{0,t}) - g(u_{k,t})] \leq 0$ – no competing forecast produces a strictly lower expected loss than the benchmark forecast.
- **H1:** At least one competing forecast has strictly lower expected loss than the benchmark forecast.

Usage

```
white_reality_check(
  loss_differences,
  n_simulations = 999,
  block_length = 5,
  alpha = 0.05
)
```

Arguments

`loss_differences`
: A numeric matrix (P x K) of loss differences (benchmark loss minus forecast loss), where P is the number of forecast periods and K is the number of competing forecasts. A positive entry means the competing forecast outperforms the benchmark forecast in that period.

`n_simulations`
: integer. The number of MBB bootstrap replications. Default 999; see Davidson & MacKinnon (2000).

`block_length`
: integer. The block length for the Moving Block Bootstrap (MBB). A commonly used rule of thumb is $T^{1/3}$ (Politis & Romano, 1994). For P = 165, this gives approximately 5–6.

`alpha`
: numeric. The significance level (default 0.05).

Details

The test statistic is $\hat{S}_P = \max_k \overline{d}_k$, where $\overline{d}_k$ is the sample mean of the loss differential series for forecast k (White, 2000, eq. 2). Bootstrap p-values are obtained via the MBB of Kunsch (1989) by recentring each bootstrap statistic at the sample mean, following the procedure in Corradi & Swanson (2011). This is an *unstudentized* test; for a studentized version with improved power against irrelevant forecasts, see `superior_predictive_ability_test`.

Value

An object of class `"htest"`. The printed output shows the test statistic (maximum mean loss differential), the bootstrap p-value, and the test name. A small p-value (below `alpha`) leads to rejection of H0, indicating that at least one competing forecast is significantly more accurate than the benchmark forecast.

References

White, H. (2000). A reality check for data snooping. *Econometrica*, 68(5), 1097–1126. doi:10.1111/14680262.00152

Kunsch, H. R. (1989). The jackknife and the bootstrap for general stationary observations. *The Annals of Statistics*, 17(3), 1217–1241. doi:10.1214/aos/1176347265

Davidson, R., & MacKinnon, J. G. (2000). Bootstrap tests: How many bootstraps? *Econometric Reviews*, 19(1), 55–68. doi:10.1080/07474930008800459

Corradi, V., & Swanson, N. R. (2011). The White Reality Check and some of its recent extensions. In *Festschrift in honor of Halbert L. White*.

Examples

```
data(metals)
# metals: 165 x 15; columns 1-14 are competing forecasts, column 15 is the benchmark
# A small offset (+0.5) is added to the lagged benchmark to avoid degenerate zero
# loss differences when forecasts equal the realized value exactly (illustration only).
P <- nrow(metals)
K_total <- ncol(metals)
K <- K_total - 1 # 14 competing forecasts
realized       <- c(metals[-1, K_total], metals[P, K_total]) + 0.5
benchmark_loss <- (metals[, K_total] - realized)^2
model_loss     <- (metals[, 1:K] - realized)^2
loss_diff      <- benchmark_loss - model_loss
res <- white_reality_check(loss_diff, block_length = 5, n_simulations = 50)
print(res)
```

`white_reality_check_cdf_approx`

White's Reality Check via Expected Loss CDF Comparison (CDF-RC)

Description

Implements a studentized Reality Check test that compares competing **forecasts** against a benchmark across the entire distribution of loss differences, not only the mean. For each forecast k and each quantile threshold x_τ (derived from the pooled empirical distribution of loss differences), the test evaluates whether the empirical CDF of **forecast k's** loss differences, evaluated at x_{τ_j}, exceeds the nominal quantile level τ_j itself – i.e., whether the competing forecast falls below that threshold *more often than expected* under a correctly calibrated loss distribution, indicating stochastic dominance of the competing forecast's loss distribution over the benchmark's.

Hypotheses:

- **H0:** $\max_{k,j}\left(E[\mathbf{1}(d_{k,t} \leq x_{\tau_j})] - \tau_j\right) \leq 0$ for all $k = 1, \ldots, K$ and all quantile thresholds $x_{\tau_j},\ j = 1, \ldots, J$ – no competing forecast's empirical CDF of loss differences exceeds its nominal quantile level τ_j at any evaluation point, i.e., no competing forecast is stochastically dominant over the benchmark anywhere in the loss distribution.
- **H1:** At least one competing forecast's empirical CDF of loss differences significantly exceeds its nominal quantile level τ_j at some threshold x_{τ_j}, i.e., the benchmark is stochastically dominated in terms of loss differences at that point.

Note that because x_{τ_j} is itself defined as the τ_j-quantile of the *pooled* loss-difference distribution (across all K forecasts), each column's empirical CDF value $\bar{G}_{k,j}$ is centred at τ_j under a null of no systematic difference across forecasts – **not at zero**. The test statistic and its bootstrap null distribution are therefore constructed relative to τ_j, not zero; see "Details".

Usage

```
white_reality_check_cdf_approx(
  loss_differences,
  block_length,
  num_bootstrap_replications,
  alpha = 0.05
)
```

Arguments

`loss_differences`
: A `numeric` matrix (`P x K`) of loss differences (benchmark loss minus forecast forecast loss), where `P` is the number of forecast periods and `K` is the number of competing forecasts. A positive entry means the competing forecast is more accurate than the benchmark forecast in that period.

`block_length`
: `integer`. The block length for the Moving Block Bootstrap (MBB) and for HAC variance estimation via `estimate_long_run_covariance`. A commonly used rule of thumb is $T^{1/3}$ (Politis & Romano, 1994). For `P = 165`, this gives approximately 5–6.

`num_bootstrap_replications`
: `integer` number of MBB bootstrap replications. Default `999`; see Davidson & MacKinnon (2000).

`alpha`
: `numeric`. The significance level (default `0.05`).

Details

The test proceeds in four steps.

Step 1 – Quantile grid. A grid of $J = 9$ evaluation points $x_{\tau_1}, \ldots, x_{\tau_9}$ is constructed as the $\tau_j \in \{0.1, 0.2, \ldots, 0.9\}$ quantiles of the *pooled* empirical distribution of all loss differences (across all forecasts and all periods). Using quantiles of the data rather than a fixed grid ensures that the evaluation points are always in the support of the observed loss differences.

Step 2 – Indicator matrix. For each forecast k and each threshold x_{τ_j}, the binary indicator

$$G_{k,j,t} = \mathbf{1}(d_{k,t} \leq x_{\tau_j})$$

is formed, where $d_{k,t}$ is the loss difference for forecast k at period t. This yields a $P \times (K \cdot J)$ matrix `Gdata` with $K \times J = 14 \times 9 = 126$ columns (for the `metals` dataset). The column mean $\bar{G}_{k,j} = \frac{1}{P} \sum_t G_{k,j,t}$ estimates the empirical CDF of forecast k's loss differences evaluated at x_{τ_j}.

Step 3 – Null-centring (*critical for correct inference*). Because x_{τ_j} is defined as the τ_j-quantile of the *pooled* loss-difference sample, $\bar{G}_{k,j}$ is mechanically close to τ_j for every forecast k whenever forecasts are exchangeable under the null – it is **not** centred at zero. The relevant test quantity is therefore the *excess* empirical CDF over its nominal level,

$$\bar{G}_{k,j} - \tau_j,$$

which is genuinely centred at zero when no forecast is stochastically dominant. Studentizing the raw $\bar{G}_{k,j}$ (without subtracting τ_j) before comparing to a bootstrap distribution of *deviations* from the observed statistic produces a test statistic and a null distribution on incompatible scales, and causes the test to reject in virtually all samples regardless of the data – this was corrected in the current package version (see "Note").

Step 4 – Studentized test statistic. Each null-centred column mean is studentized by its HAC standard deviation (from `estimate_long_run_covariance`), and the test statistic is the maximum studentized excess CDF value across all $K \times J$ columns:

$$\hat{T} = \max_{k,j} \frac{\bar{G}_{k,j} - \tau_j}{\hat{\sigma}_{k,j}}$$

Bootstrap p-values are obtained via the MBB of Kunsch (1989) with recentring, following the WRC procedure of White (2000) and Corradi & Swanson (2011); the same τ_j offset is subtracted from each bootstrap replicate's column mean before studentizing, so that both the observed statistic and its bootstrap distribution are computed on the same, correctly null-centred scale.

Relationship to Corradi & Swanson (2006). This test is a loss-difference analogue of the predictive CDF comparison in Corradi & Swanson (2006, Section 4). Rather than comparing forecast CDFs against the true conditional distribution (as in the ZP test), it compares empirical CDFs of *loss differences* against their own nominal quantile levels, assessing stochastic dominance of a competing forecast over the benchmark in terms of loss. It complements `white_reality_check` (which tests only the mean) by detecting cases where one forecast is better in the tails but not on average.

Lower p-values are more informative: rejection of H0 indicates that at least one forecast stochastically dominates the benchmark at some point of the loss distribution. Failure to reject does not preclude dominance at specific quantiles – it means no single (k, j) combination is significant after controlling for multiple comparisons.

Value

An object of class "htest" with the following components:

statistic	Maximum studentized excess CDF value (empirical CDF minus nominal quantile level) across all $K \times J$ fore
p.value	Bootstrap p-value from the MBB procedure.
method	"Expected Loss CDF Comparison Test".
null.value	Named scalar "max studentized excess CDF value (empirical CDF minus nominal quantile level)" =
alternative	Description of the alternative hypothesis.

A small p-value (below alpha) leads to rejection of H0, indicating that at least one competing forecast's empirical CDF of loss differences significantly exceeds its nominal quantile level at some threshold – i.e., the competing forecast more frequently produces smaller losses than the benchmark forecast in some region of the loss distribution than would be expected by chance.

Note

Package versions prior to the current release contained two implementation errors in this function, both now corrected:

1. **Redundant variance division.** estimate_long_run_covariance already divides its output by the sample size internally; this function additionally divided diag(V_hat_full) by P_clean a second time before studentizing, inflating every studentized statistic by a factor of approximately $\sqrt{P}$. This has been removed.
2. **Missing null-centring.** As described in Step 3 above, the test statistic and bootstrap null distribution were computed on incompatible scales (raw empirical CDF values compared against a distribution of deviations from those same values), causing the test to reject in essentially 100% of samples regardless of the data – verified via an isolated Monte Carlo check under a pure i.i.d. null with a single forecast and a single quantile threshold. This function now subtracts the nominal quantile level τ_j from each column's empirical CDF before studentizing, restoring rejection rates near the nominal significance level under simulated null data.

Results obtained from this function in prior package versions should be treated as unreliable and are superseded by output from the current version.

References

Corradi, V., & Swanson, N. R. (2006). Predictive density and conditional confidence interval accuracy tests. *Journal of Econometrics*, 135(1–2), 187–228. doi:10.1016/j.jeconom.2005.07.026

Davidson, R., & MacKinnon, J. G. (2000). Bootstrap tests: How many bootstraps? *Econometric Reviews*, 19(1), 55–68. doi:10.1080/07474930008800459

White, H. (2000). A reality check for data snooping. *Econometrica*, 68(5), 1097–1126. doi:10.1111/14680262.00152

Corradi, V., & Swanson, N. R. (2011). The White Reality Check and some of its recent extensions. In *Festschrift in honor of Halbert L. White*.

Kunsch, H. R. (1989). The jackknife and the bootstrap for general stationary observations. *The Annals of Statistics*, 17(3), 1217–1241. doi:10.1214/aos/1176347265

Politis, D. N., & Romano, J. P. (1994). The stationary bootstrap. *Journal of the American Statistical Association*, 89(428), 1303–1313. doi:10.1080/01621459.1994.10476870

See Also

`white_reality_check` for the mean-based WRC test; `reality_check_zp_test` for a distributional test based on the true conditional CDF; `superior_predictive_ability_test` for the studentized mean-based SPA test.

Examples

```
data(metals)
# metals: 165 x 15; columns 1-14 are competing forecasts, column 15 is the benchmark
# A small offset (+0.5) is added to the lagged benchmark to avoid degenerate zero
# loss differences when forecasts equal the realized value exactly (illustration only).
P       <- nrow(metals)
K_total <- ncol(metals)
K       <- K_total - 1L  # 14 competing forecasts
realized       <- c(metals[-1, K_total], metals[P, K_total]) + 0.5
benchmark_loss <- (metals[, K_total] - realized)^2
model_loss     <- (metals[, 1:K]     - realized)^2
loss_diff      <- benchmark_loss - model_loss
res <- white_reality_check_cdf_approx(loss_diff,
                                      block_length               = 5,
                                      num_bootstrap_replications = 50)
print(res)
```

`white_reality_check_conditional`

Conditional Predictive Ability (CPA) Reality Check Test

Description

Implements the Conditional Predictive Ability (CPA) test of Giacomini & White (2006), extended to a multiple-forecast setting via a studentized Reality Check statistic. Tests whether any competing forecast's predictive advantage over the benchmark is state-dependent, i.e., predictable from a conditioning variable h_t known at the time the forecast is made.

Hypotheses:

- **H0:** $E[h_t \cdot (g(u_{0,t}) - g(u_{k,t}))] = 0$ for all $k = 1, \ldots, K$ – no competing forecast's loss differential with the benchmark is predictable using the conditioning information h_t.
- **H1:** At least one forecast's loss differential $d_{k,t} = g(u_{0,t}) - g(u_{k,t})$ is predictable by h_t, i.e., $E[h_t \cdot d_{k,t}] \neq 0$ for some k.

Usage

```
white_reality_check_conditional(
  loss_differences,
  weighting_vector,
  block_length,
  num_bootstrap_replications,
  alpha
)
```

Arguments

`loss_differences`
: A `numeric` matrix (`P x K`) of loss differences (benchmark loss minus forecast loss), where `P` is the number of forecast periods and `K` is the number of competing forecasts. A positive entry means the competing forecast outperforms the benchmark forecast in that period.

`weighting_vector`
: `numeric` vector of length `P` serving as the conditioning instrument h_t in the CPA test. At each period t, the test checks whether the loss differential $d_{k,t}$ covaries with h_t, i.e., whether $E[h_t \cdot d_{k,t}] \neq 0$. See the *Conditioning Instrument* section in Details for interpretation, requirements, and recommended choices.

`block_length`
: `integer`. The block length for MBB and HAC estimation. A commonly used rule of thumb is $T^{1/3}$ (Politis & Romano, 1994). For `P = 165`, this gives approximately 5–6.

`num_bootstrap_replications`
: `integer` number of MBB bootstrap replications. Default `999`; see Davidson & MacKinnon (2000).

`alpha`
: `numeric`. The significance level (default `0.05`).

Details

The test multiplies each column of `loss_differences` element-wise by `weighting_vector` to form the weighted loss differential series $h_t \cdot d_{k,t}$. The unconditional mean of this product, $E[h_t \cdot d_{k,t}]$, equals zero under H0 by the law of iterated expectations when h_t is a valid instrument. The test statistic is the maximum studentized mean across all K forecasts:

$$\hat{T}_{CPA} = \max_k \frac{\frac{1}{P}\sum_t h_t d_{k,t}}{\hat{\sigma}_{k,h}}$$

where $\hat{\sigma}_{k,h}$ is the HAC standard deviation of $h_t d_{k,t}$ estimated via `estimate_long_run_covariance`. Bootstrap p-values are obtained via the MBB of Kunsch (1989) with recentring, following the SPA-type procedure of Hansen (2005) applied to the weighted series.

Conditioning Instrument (`weighting_vector`):

A significant result means that knowing h_t allows one to predict which forecast will perform better in period t – the benchmark's advantage (or disadvantage) is state-dependent and potentially exploitable. This is a strictly stronger statement than the unconditional WRC: a forecast can fail the WRC (no unconditional improvement) yet pass the CPA test (conditional improvement in specific states).

h_t must be measurable with respect to the information set available at time t (Giacomini & White, 2006, Assumption 1) – it must not use information from period $t+1$ or later. The scale of h_t does not affect the test result because the statistic is studentized by its own HAC standard deviation.

Recommended choices:

`abs(realized)` **– absolute realised values** Tests whether forecast performance depends on outcome magnitude – a natural proxy for market volatility or economic uncertainty. Default in `run_comprehensive_erc_analysis`.

`c(realized[1], realized[-length(realized)])` **– lagged realised values** Tests whether the previous period's outcome predicts which forecast wins next period. Relevant when forecast errors are autocorrelated.

`rep(1, P)` **– constant vector** The product $h_t \cdot d_{k,t}$ reduces to $d_{k,t}$, making the CPA test equivalent to the unconditional WRC. Use as a sanity check: results should be consistent with `white_reality_check`.

External economic indicator E.g., a recession dummy, VIX level, lagged interest rate spread, or monetary policy stance dummy. Tests whether one forecast systematically outperforms during specific regimes. Must be lagged one period to ensure h_t is in the information set at the time of the forecast.

The scale of `weighting_vector` has no effect on inference because both the test statistic and its bootstrap distribution are studentized by the same $\hat{\sigma}_{k,h}$.

Value

An object of class `"htest"` with the following components:

`statistic`	Maximum studentized weighted mean loss differential across all K forecasts, labelled `"T-CPA"`.
`p.value`	Bootstrap p-value from the MBB procedure.
`method`	`"Conditional Predictive Ability (CPA) Test"`.
`null.value`	Named scalar `"max studentized weighted mean loss differential"` = 0.
`alternative`	Direction of the alternative hypothesis.
`reject_null`	Logical: `TRUE` if `p.value <= alpha`.

A small p-value indicates that at least one forecast's loss differential is predictable from the conditioning variable h_t. Failure to reject H0 means no evidence of state-dependent predictive ability for the chosen instrument.

References

Giacomini, R., & White, H. (2006). Tests of Conditional Predictive Ability. *Econometrica*, 74(6), 1545–1578. doi:10.1111/j.14680262.2006.00718.x

Hansen, P. R. (2005). A Test for Superior Predictive Ability. *Journal of Business & Economic Statistics*, 23(4), 365–380. doi:10.1198/073500105000000063

Corradi, V., & Swanson, N. R. (2011). The White Reality Check and some of its recent extensions. In *Festschrift in honor of Halbert L. White*.

Davidson, R., & MacKinnon, J. G. (2000). Bootstrap tests: How many bootstraps? *Econometric Reviews*, 19(1), 55–68. doi:10.1080/07474930008800459

Kunsch, H. R. (1989). The jackknife and the bootstrap for general stationary observations. *The Annals of Statistics*, 17(3), 1217–1241. doi:10.1214/aos/1176347265

Politis, D. N., & Romano, J. P. (1994). The stationary bootstrap. *Journal of the American Statistical Association*, 89(428), 1303–1313. doi:10.1080/01621459.1994.10476870

See Also

`white_reality_check` for the unconditional WRC test (equivalent to CPA with a constant `weighting_vector`); `superior_predictive_ability_test` for the studentized unconditional test.

Examples

```
data(metals)
# metals: 165 x 15; columns 1-14 are competing forecasts, column 15 is the benchmark
# A small offset (+0.5) is added to the lagged benchmark to avoid degenerate zero
# loss differences when forecasts equal the realized value exactly (illustration only).
P       <- nrow(metals)
K_total <- ncol(metals)
K       <- K_total - 1L  # 14 competing forecasts
realized       <- c(metals[-1, K_total], metals[P, K_total]) + 0.5
benchmark_loss <- (metals[, K_total] - realized)^2
model_loss     <- (metals[, 1:K]     - realized)^2
loss_diff      <- benchmark_loss - model_loss

# Example 1: absolute realised values as conditioning variable (volatility proxy)
res1 <- white_reality_check_conditional(
  loss_differences           = loss_diff,
  weighting_vector           = abs(realized),
  block_length               = 5,
  num_bootstrap_replications = 50,
  alpha                      = 0.05
)
print(res1)

# Example 2: constant vector - should give results consistent with white_reality_check()
res2 <- white_reality_check_conditional(
  loss_differences           = loss_diff,
  weighting_vector           = rep(1, P),
  block_length               = 5,
  num_bootstrap_replications = 50,
  alpha                      = 0.05
)
print(res2)
```

Index