Package 'BTWAR'

March 19, 2026

Type Package

Title Butterworth-Induced Autoregressive Model

Version 1.0.1

Description Implements the Butterworth-Induced Autoregressive ('BTWAR') model, where autoregressive coefficients are obtained from analog Butterworth filter prototypes mapped into the discrete-time domain using the Matched Z-Transform. The framework establishes a structured connection between frequency-domain filter design and time-domain autoregressive modeling. Model order selection is performed via nested rolling-origin cross-validation. Method described in Bras-Geraldes, Rocha and Martins (2026) <doi:10.3390/math14030479>.

Depends R (>= 3.5.0)

License GPL-3

Encoding UTF-8

Language en-US

NeedsCompilation no

Imports ggplot2, pracma, tseries, scales

Suggests knitr, rmarkdown, spelling

VignetteBuilder knitr

URL https://doi.org/10.3390/math14030479, https://github.com/cgeraldes/BTWAR

BugReports https://github.com/cgeraldes/BTWAR/issues

RoxygenNote 7.3.3

Author Carlos Bras-Geraldes [aut, cre, cph], J. Leonel Rocha [aut, cph]

Maintainer Carlos Bras-Geraldes <cgeraldes@gmail.com>

Repository CRAN

Date/Publication 2026-03-19 14:40:12 UTC

Contents

`apply_stationarity` *Apply Stationarity Transformation to a Train/Test Split*

Description

Determines the differencing order required to stationarise the training series using `make_stationary()` and applies the same order to the test series, ensuring both sets are on a comparable scale.

Usage

```
apply_stationarity(y_tr_raw, y_te_raw, max_d = 3L, alpha = 0.05)
```

Arguments

`y_tr_raw`	Numeric vector. Raw training time series. Must contain at least 4 observations.
`y_te_raw`	Numeric vector. Raw test (hold-out) time series. Must contain at least 1 observation.
`max_d`	Non-negative integer. Maximum differencing order passed to `make_stationary()`. Default `3`.
`alpha`	Numeric in $(0, 1)$. Significance level for the Augmented Dickey-Fuller test. Default `0.05`.

Details

This function is a convenience wrapper intended for use before fitting a `btwar_fit` model when the user wants to inspect or control the stationarity step explicitly. Internally, `btwar_fit` calls `make_stationary()` directly.

Value

A named list with four elements:

`y_tr` Numeric vector. Stationarised training series of length $n_{tr} - d$.

`y_te` Numeric vector. Differenced test series of length $n_{te} - d$.

`d` Integer. Number of differences applied.

`stationary` Logical. `TRUE` if the ADF test declared the training series stationary.

See Also

`btwar_fit`, `adf.test`

Examples

```
set.seed(1)
y     <- cumsum(rnorm(200))          # random walk (non-stationary)
n_tr  <- 160
split <- apply_stationarity(
  y_tr_raw = y[seq_len(n_tr)],
  y_te_raw = y[seq.int(n_tr + 1L, length(y))]
)
split$d           # differencing order applied
split$stationary  # was stationarity achieved?
```

---

btwar_fit *Fit a Butterworth-Induced Autoregressive Model*

Description

Fits a Butterworth-Induced Autoregressive (BTWAR) model using nested rolling-origin cross-validation for order and attenuation parameter selection.

Usage

```
btwar_fit(
  y_tr_raw,
  y_te_raw,
  fs = 2,
  method = "ls",
  As_vec = seq(20, 60, by = 5),
```

```
  N_vec = 2:20,
  max_d = 3L,
  alpha_stationarity = 0.05,
  min_train = NULL,
  verbose = TRUE
)
```

Arguments

`y_tr_raw`	Numeric vector. Training time series.
`y_te_raw`	Numeric vector. Test (hold-out) time series.
`fs`	Positive numeric. Sampling frequency. Default `2`.
`method`	Character string. Scaling estimation method, one of `"ls"` (least squares), `"lad"` (least absolute deviations), or `"huber"` (Huber M-estimator). Default `"ls"`.
`As_vec`	Numeric vector. Grid of stopband attenuation values (dB) to search over. Default `seq(20, 60, by = 5)`.
`N_vec`	Integer vector. Grid of AR orders to search over. Default `2:20`.
`max_d`	Non-negative integer. Maximum differencing order allowed during the stationarity transformation step. Default `3`.
`alpha_stationarity`	Numeric in $(0, 1)$. Significance level for the Augmented Dickey-Fuller stationarity test. Default `0.05`.
`min_train`	Non-negative integer or `NULL`. Minimum number of observations used for training in the rolling-origin cross-validation and for the final predictions. If `NULL` (default), the criterion `max(20 * p, 100)` is applied automatically.
`verbose`	Logical. If `TRUE` (default), prints progress messages during cross-validation.

Value

An object of class `"btwar"`, which is a named list containing:

`parameters` List with `d_used`, `N_opt`, `A_opt`, `fc_opt`, `alpha`, `phi`, and `poles_z`.

`performance` List with `rmse_train` and `rmse_test`.

`train` List with `y_real` and `yhat` for the training set.

`test` List with `y_real` and `yhat` for the test set.

`mu` Training mean used for centering.

`call` The matched call.

See Also

`predict.btwar`, `summary.btwar`

Examples

```
set.seed(42)
y <- cumsum(rnorm(900))
train_series <- y[1:600]
test_series  <- y[601:900]

result <- btwar_fit(
  y_tr_raw = train_series,
  y_te_raw = test_series,
  fs       = 2,
  method   = "ls",
  N_vec    = 2:5,
  As_vec   = c(20, 40),
  verbose  = FALSE
)
summary(result)
```

`coef.btwar` *Extract AR Coefficients from a BTWAR Model*

Description

Extract AR Coefficients from a BTWAR Model

Usage

```
## S3 method for class 'btwar'
coef(object, ...)
```

Arguments

`object`	Object of class `"btwar"`.
`...`	Additional arguments (currently unused).

Value

Named numeric vector of AR coefficients.

`compute_spectrum` *Compute the One-Sided Power Spectrum*

Description

Estimates the one-sided power spectrum of a time series using the Fast Fourier Transform (FFT). The series is mean-centred prior to transformation to remove the DC component.

Usage

```
compute_spectrum(x, fs = 1)
```

Arguments

`x`	Numeric vector. The input time series. Must contain at least 2 observations.
`fs`	Positive numeric. Sampling frequency in Hz (or consistent units). Default 1.

Details

Let X_k denote the k-th coefficient of the DFT of $x_1, \ldots, x_n$. The raw periodogram is defined as

$$P_k = \frac{|X_k|^2}{n}, \quad k = 0, 1, \ldots, \lfloor n/2 \rfloor - 1$$

and the corresponding frequencies are

$$f_k = \frac{k \cdot f_s}{n}$$

Only the non-redundant (one-sided) portion of the spectrum is returned, i.e. frequencies from 0 up to (but not including) the Nyquist frequency $f_s/2$.

Value

A named list with two elements:

`f` Numeric vector of frequencies (0 to $f_s/2$, exclusive), length $\lfloor n/2 \rfloor$.

`P` Numeric vector of power spectral estimates, same length as `f`.

See Also

`fft`, `spectrum`

Examples

```
set.seed(1)
x   <- arima.sim(n = 256, model = list(ar = 0.8))
sp  <- compute_spectrum(x, fs = 1)
plot(sp$f, sp$P, type = "l",
     xlab = "Frequency (Hz)", ylab = "Power",
     main = "One-sided power spectrum")
```

`fitted.btwar` *Fitted Values from a BTWAR Model*

Description

Fitted Values from a BTWAR Model

Usage

```
## S3 method for class 'btwar'
fitted(object, ...)
```

Arguments

`object`	Object of class `"btwar"`.
`...`	Additional arguments (currently unused).

Value

Numeric vector of fitted values on the training set.

`mse` *Mean Squared Error*

Description

Computes the mean squared error (MSE) between observed and predicted values. MSE is commonly used for optimisation and theoretical analysis, and is the square of `rmse`.

Usage

```
mse(y_true, y_pred)
```

Arguments

`y_true`	Numeric vector of observed (true) values.
`y_pred`	Numeric vector of predicted values. Must have the same length as `y_true`.

Details

$$\mathrm{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

Value

A single non-negative numeric value.

See Also

rmse

Examples

```
y    <- c(1, 2, 3)
yhat <- c(1.1, 1.9, 3.2)
mse(y, yhat)
```

`plot.btwar` *Plot Predictions from a BTWAR Model*

Description

Generates a **ggplot2** line chart of fitted or test-set predictions from a `"btwar"` object. Optionally overlays the observed series and/or an external comparison series (e.g., a benchmark model).

Usage

```
## S3 method for class 'btwar'
plot(
  x,
  dataset = c("train", "test"),
  fs = 1,
  show_observed = TRUE,
  external = NULL,
  external_name = "External",
  title = NULL,
  colour_btwar = "#2166AC",
  colour_observed = "#333333",
  colour_external = NULL,
  lwd_btwar = 0.9,
  lwd_observed = 0.55,
  lwd_external = 0.7,
  alpha = 0.9,
  base_size = 11,
```

```
  palette = "Dark 2",
  layer_order = NULL,
  ...
)
```

Arguments

x	Object of class "btwar", as returned by btwar_fit.
dataset	Character string. Which dataset to display: "train" (default) or "test".
fs	Positive numeric. Sampling frequency used to construct the time axis as $t = 0, 1/f_s, 2/f_s, \ldots$. Default 1.
show_observed	Logical. If TRUE (default), the observed series is overlaid on the predicted values.
external	Optional numeric vector of the same length as the selected dataset. If supplied, it is plotted as an additional comparison series. Default NULL.
external_name	Character string. Legend label for the external series. Default "External".
title	Character string. Plot title. If NULL (default), a title is generated automatically from the dataset name.
colour_btwar	Character string. Colour for the BTWAR prediction line. Default "#2166AC".
colour_observed	Character string. Colour for the observed series line. Default "#333333".
colour_external	Character string. Colour for the external series line. If NULL (default), a colour is assigned automatically from the palette.
lwd_btwar	Positive numeric. Line width for the BTWAR prediction line. Default 0.9.
lwd_observed	Positive numeric. Line width for the observed series line. Default 0.55.
lwd_external	Positive numeric. Line width for any external series line. Default 0.7.
alpha	Numeric in $(0, 1]$. Transparency of all lines. Default 0.9.
base_size	Positive numeric. Base font size passed to theme_minimal. Default 11.
palette	Character string. Name of the hcl.colors palette used for additional external series when colour_external is NULL. Default "Dark 2".
layer_order	Character vector. Controls the drawing order (back to front) and legend order of the series. Must contain the names of all active series: "Observed", "BTWAR", and the value of external_name if an external series is supplied. If NULL (default), the fixed order Observed $\rightarrow$ BTWAR $\rightarrow$ external is used.
...	Additional arguments (currently unused).

Details

Series are drawn in the following fixed order (back to front): Observed, BTWAR, external.

Value

A ggplot object. The plot is also printed as a side effect when called interactively.

See Also

btwar_fit, plot_zpoles, fitted.btwar, residuals.btwar

Examples

```
sim <- simulate_ar_split(phi_real = c(0.34, -0.22, 0.16),
                         n = 1000, burn = 200, prop_train = 0.7)
fit <- btwar_fit(y_tr_raw = as.numeric(sim$train),
                 y_te_raw = as.numeric(sim$test),
                 fs = 12000, method = "ls", N_vec = 3:3)

plot(fit, dataset = "train", fs = 12000)
plot(fit, dataset = "test",  fs = 12000, show_observed = TRUE)
```

---

plot_bode *Plot the Bode Magnitude Diagram of a BTWAR Model*

Description

Draws the Butterworth filter magnitude response (in dB) associated with the selected BTW-AR model, using the optimal order N and cutoff frequency f_c stored in the fitted object. Reference lines are drawn at the cutoff frequency, the Nyquist frequency, and their corresponding magnitude levels.

Usage

```
plot_bode(
  x,
  fs = 2,
  n_freq = 2000L,
  colour_magnitude = "blue",
  colour_cutoff = "orange",
  colour_nyquist = "red",
  lwd_magnitude = 1.2,
  lwd_ref = 1,
  base_size = 11,
  title = NULL
)
```

Arguments

x	Object of class "btwar", as returned by btwar_fit. The parameters N_opt, A_opt, and fc_opt are read from x$parameters.
fs	Positive numeric. Sampling frequency (Hz) used to derive the Nyquist frequency as $f_s/2$. Must match the value passed to btwar_fit.

n_freq	Positive integer. Number of frequency points used to evaluate the magnitude response. Default 2000L.
colour_magnitude	Character string. Colour for the magnitude response curve. Default "blue".
colour_cutoff	Character string. Colour for the cutoff frequency reference lines (vertical dashed and horizontal dotted). Default "orange".
colour_nyquist	Character string. Colour for the Nyquist frequency reference lines (vertical dashed and horizontal dotted). Default "red".
lwd_magnitude	Positive numeric. Line width for the magnitude response curve. Default 1.2.
lwd_ref	Positive numeric. Line width for all reference lines. Default 1.0.
base_size	Positive numeric. Base font size passed to theme_minimal. Default 11.
title	Character string. Plot title. If NULL (default), a title is generated automatically from the fitted model parameters.

Details

The Butterworth filter magnitude response is

$$|H(f)| = \frac{1}{\sqrt{1 + \beta^2 \, (2\pi f)^{2N}}}, \quad \beta = \frac{1}{(2\pi f_c)^N},$$

evaluated on a log-spaced frequency axis from 10^{-3} Hz to the Nyquist frequency. The magnitude at the cutoff frequency equals -3 dB ($1/\sqrt{2}$), and the magnitude at the Nyquist frequency reflects the actual stopband attenuation achieved by the selected filter.

Value

A ggplot object. The plot is also printed as a side effect when called interactively.

See Also

btwar_fit, plot.btwar, plot_zpoles

Examples

```
sim <- simulate_ar_split(phi_real = c(0.34, -0.22, 0.16),
                         n = 1000, burn = 200, prop_train = 0.7)
fit <- btwar_fit(y_tr_raw = as.numeric(sim$train),
                 y_te_raw = as.numeric(sim$test),
                 fs = 12000, method = "ls", N_vec = 3:3)

plot_bode(fit, fs = 12000)
```

`plot_freq_amplitude` *Plot the Frequency Amplitude Spectrum of a Time Series*

Description

Displays the one-sided amplitude spectrum of a numeric series, computed via the Fast Fourier Transform (FFT), with reference lines at the cutoff frequency and the Nyquist frequency.

Usage

```
plot_freq_amplitude(
  x,
  fs,
  fc,
  colour_spectrum = "blue",
  colour_cutoff = "red",
  colour_nyquist = "black",
  lwd_spectrum = 0.6,
  lwd_ref = 1,
  base_size = 11,
  title = NULL
)
```

Arguments

`x`	Numeric vector. The time series to analyze. The mean is removed internally before computing the FFT.
`fs`	Positive numeric. Sampling frequency (Hz). Must match the value passed to `btwar_fit`.
`fc`	Positive numeric. Cutoff frequency (Hz) to display as a reference line. Typically read from `fit$parameters$fc_opt`.
`colour_spectrum`	Character string. Colour for the amplitude spectrum segments. Default `"blue"`.
`colour_cutoff`	Character string. Colour for the cutoff frequency reference line. Default `"red"`.
`colour_nyquist`	Character string. Colour for the Nyquist frequency reference line. Default `"black"`.
`lwd_spectrum`	Positive numeric. Line width for the amplitude spectrum segments. Default `0.6`.
`lwd_ref`	Positive numeric. Line width for the reference lines. Default `1.0`.
`base_size`	Positive numeric. Base font size passed to `theme_minimal`. Default `11`.
`title`	Character string. Plot title. Default `NULL` (no title).

Value

A `ggplot` object. The plot is also printed as a side effect when called interactively.

See Also

btwar_fit, plot.btwar, plot_bode

Examples

```
sim <- simulate_ar_split(phi_real = c(0.34, -0.22, 0.16),
                         n = 1000, burn = 200, prop_train = 0.7)
fit <- btwar_fit(y_tr_raw = as.numeric(sim$train),
                 y_te_raw = as.numeric(sim$test),
                 fs = 12000, method = "ls", N_vec = 3:3)

plot_freq_amplitude(as.numeric(sim$train),
                fs = 12000,
                fc = fit$parameters$fc_opt)
```

---

plot_zpoles *Plot Z-Plane Poles from a BTWAR Model*

Description

Displays the poles of the selected BTW-AR model in the complex Z-plane, together with the unit circle. Optionally overlays poles from one or more external sources (e.g., a fitted ARIMA model or the true data-generating poles).

Usage

```
plot_zpoles(
  x,
  external_list = NULL,
  colour_selected = "green",
  external_colours = c("red", "darkgreen", "purple", "orange", "brown", "navy"),
  external_shapes = c(17L, 15L, 18L, 8L, 10L, 12L),
  lim = 1.5,
  base_size = 10,
  title = NULL
)
```

Arguments

x	Object of class "btwar", as returned by btwar_fit. The selected model poles are read from x$parameters$poles_z.
external_list	Named list of external pole sets to overlay. Each element must be a data.frame with columns Re and Im giving the real and imaginary parts of the poles. The element names are used as legend labels. Example: list("ARIMA Z-Poles" = df_arima, "True Signal" = df_true). Default NULL (no external poles).

`colour_selected`
: Character string. Colour for the BTW-AR (selected) poles. Default `"green"`.

`external_colours`
: Character vector. Colours assigned to external pole sets in order. Recycled if fewer colours than sets are provided. Default `c("red", "darkgreen", "purple", "orange", "brown", "navy")`.

`external_shapes`
: Integer vector. shape codes assigned to external pole sets in order. Recycled if needed. Default `c(17L, 15L, 18L, 8L, 10L, 12L)`.

`lim`
: Positive numeric. Half-width of the plot window on both axes. Default `1.5`.

`base_size`
: Positive numeric. Base font size passed to `theme_minimal`. Default `10`.

`title`
: Character string. Plot title. If `NULL` (default), no title is added.

Details

Poles of the selected BTW-AR model are extracted from `x$parameters$poles_z`, which is populated by `btwar_fit` via the internal cross-validation routine. The unit circle is drawn as a reference: poles inside the circle correspond to stable filters.

Colours and shapes for external pole sets are assigned sequentially from `external_colours` and `external_shapes`, with circular recycling when the number of sets exceeds the palette length.

Value

A `ggplot` object. The plot is also printed as a side effect when called interactively.

See Also

`btwar_fit`, `plot.btwar`

Examples

```
sim <- simulate_ar_split(phi_real = c(0.34, -0.22, 0.16),
                         n = 1000, burn = 200, prop_train = 0.7)
fit <- btwar_fit(y_tr_raw = as.numeric(sim$train),
                 y_te_raw = as.numeric(sim$test),
                 fs = 12000, method = "ls", N_vec = 3:3)

plot_zpoles(fit)

df_true <- data.frame(Re = Re(sim$poles), Im = Im(sim$poles))
plot_zpoles(fit, external_list = list("True Signal" = df_true))
```

poles_AR *Compute Z-Plane Poles of an AR Model*

Description

Returns the poles of an autoregressive (AR) model in the complex Z-plane by finding the roots of the characteristic polynomial $1 - \phi_1 z^{-1} - \cdots - \phi_p z^{-p}$.

Usage

```
poles_AR(phi)
```

Arguments

phi	Numeric vector of AR coefficients $(\phi_1, \phi_2, \ldots, \phi_p)$, ordered from lag 1 to lag p.

Details

The characteristic polynomial is

$$A(z) = 1 - \phi_1 z^{-1} - \cdots - \phi_p z^{-p},$$

which, after multiplication by z^p, becomes the polynomial whose roots are computed by `polyroot`. Coefficients are reversed internally so that `polyroot` receives them in ascending degree order.

Value

A complex vector of length p containing the Z-plane poles. A model is stable if and only if all poles lie strictly inside the unit circle, i.e., `all(Mod(poles_AR(phi)) < 1)`.

See Also

`yhat_ar`, `yhat_arma`, `plot_zpoles`, `polyroot`

Examples

```
# AR(2) with phi = c(0.6, -0.3)
phi  <- c(0.6, -0.3)
poles <- poles_AR(phi)
print(poles)
all(Mod(poles) < 1)  # TRUE => stable


# Use with plot_zpoles
sim <- simulate_ar_split(phi_real = c(0.34, -0.22, 0.16),
                         n = 1000, burn = 200, prop_train = 0.7)
fit <- btwar_fit(y_tr_raw = as.numeric(sim$train),
                 y_te_raw = as.numeric(sim$test),
                 fs = 12000, method = "ls", N_vec = 3:3)
```

```
df <- data.frame(Re = Re(poles), Im = Im(poles))
plot_zpoles(fit, external_list = list("AR(2) poles" = df))
```

`predict.btwar` *Predict from a BTWAR Model*

Description

Generates one-step-ahead predictions for a new time series using a fitted `"btwar"` model.

Usage

```
## S3 method for class 'btwar'
predict(object, newdata, ...)
```

Arguments

`object`	Object of class `"btwar"`.
`newdata`	Numeric vector. New time series to predict from.
`...`	Additional arguments (currently unused).

Value

Numeric vector of predicted values.

`print.btwar` *Print a BTWAR Model*

Description

Compact console display of a fitted `"btwar"` object.

Usage

```
## S3 method for class 'btwar'
print(x, ...)
```

Arguments

`x`	Object of class `"btwar"`.
`...`	Additional arguments (currently unused).

Value

Invisibly returns `x`.

print.summary.btwar *Print a BTWAR Model Summary*

Description

Print a BTWAR Model Summary

Usage

```
## S3 method for class 'summary.btwar'
print(x, ...)
```

Arguments

x	Object of class "summary.btwar".
...	Additional arguments (currently unused).

Value

Invisibly returns x.

residuals.btwar *Residuals from a BTWAR Model*

Description

Residuals from a BTWAR Model

Usage

```
## S3 method for class 'btwar'
residuals(object, ...)
```

Arguments

object	Object of class "btwar".
...	Additional arguments (currently unused).

Value

Numeric vector of residuals from the training set.

`rmse` *Root Mean Squared Error*

Description

Computes the root mean squared error (RMSE) between observed and predicted values. RMSE penalises large deviations more strongly than MAE due to squaring, and is expressed in the same units as the response variable.

Usage

```
rmse(y_true, y_pred)
```

Arguments

`y_true`	Numeric vector of observed (true) values.
`y_pred`	Numeric vector of predicted values. Must have the same length as `y_true`.

Details

$$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}$$

Value

A single non-negative numeric value.

See Also

`mse`

Examples

```
y    <- c(1, 2, 3)
yhat <- c(1.1, 1.9, 3.2)
rmse(y, yhat)
```

`simulate_ar_split` *Simulate an AR(p) Series and Split into Train/Test Sets*

Description

Generates a stationary AR(p) time series via `arima.sim`, discards a burn-in period, and partitions the result into a training set and a hold-out test set.

Usage

```
simulate_ar_split(
  phi_real,
  n = 2000L,
  burn = 200L,
  prop_train = 0.8,
  seasonal = FALSE,
  freq = NA
)
```

Arguments

`phi_real`	Numeric vector of true AR coefficients $\phi_1, \ldots, \phi_p$. The process must be stationary (all roots of the characteristic polynomial outside the unit circle).
`n`	Integer. Number of observations to retain after burn-in. Default `2000`.
`burn`	Integer. Number of initial observations discarded as burn-in. Default `200`.
`prop_train`	Numeric in $(0, 1)$. Proportion of observations allocated to the training set. Default `0.8`.
`seasonal`	Logical. If `TRUE` and `freq` is supplied, the retained series is coerced to a `ts` object with the given frequency. Default `FALSE`.
`freq`	Integer or `NA`. Seasonal frequency passed to `ts` when `seasonal = TRUE`. Default `NA`.

Details

Random number generation is controlled externally by the user via `set.seed` when reproducibility is required.

The function does not modify the global random number generator state. For reproducible results, users should call `set.seed()` prior to invoking this function.

Value

A named list with four elements:

`series` The full simulated series (length `n`).

`train` Training observations (first `floor(n * prop_train)` values).

`test` Hold-out test observations (remaining values).

`poles` Complex vector of true AR poles in the Z-plane.

See Also

arima.sim, ts

Examples

```
# Reproducible AR(3) with phi = (0.6, -0.3, 0.2)
set.seed(123)
result <- simulate_ar_split(phi_real = c(0.6, -0.3, 0.2))
length(result$train)  # 1600
length(result$test)   # 400

# AR(3) as a monthly time series, 90/10 split
set.seed(123)
result2 <- simulate_ar_split(
  phi_real   = c(0.6, -0.3, 0.2),
  n          = 1200,
  prop_train = 0.9,
  seasonal   = TRUE,
  freq       = 12
)
```

---

summary.btwar *Summarise a BTWAR Model*

Description

Returns a "summary.btwar" object with model parameters and performance metrics. The associated print method displays a formatted summary including the AR coefficients.

Usage

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

Arguments

object	Object of class "btwar".
...	Additional arguments (currently unused).

Value

An object of class "summary.btwar" (invisibly).

yhat_ar *One-Step-Ahead AR Predictions*

Description

Computes one-step-ahead predictions for a time series under a pure autoregressive model of order p.

Usage

```
yhat_ar(x, phi)
```

Arguments

x	Numeric vector. The observed time series.
phi	Numeric vector of AR coefficients $(\phi_1, \phi_2, \ldots, \phi_p)$, ordered from lag 1 to lag p.

Details

No intercept is included. Center x before calling this function if the series has a non-zero mean.

Value

A numeric vector of the same length as x. The first p elements are NA (insufficient history); element t ($t > p$) contains $\hat{x}_t = \phi_1 x_{t-1} + \cdots + \phi_p x_{t-p}$.

See Also

poles_AR, yhat_arma, btwar_fit

Examples

```
x   <- as.numeric(arima.sim(list(ar = c(0.6, -0.3)), n = 200))
phi <- c(0.6, -0.3)
yh  <- yhat_ar(x, phi)

# First p values are NA
head(yh, 5)

# RMSE on the predictable portion
obs <- x[(length(phi) + 1):length(x)]
hat <- yh[(length(phi) + 1):length(x)]
sqrt(mean((obs - hat)^2))
```

yhat_arma *One-Step-Ahead ARMA Predictions*

Description

Computes one-step-ahead predictions for a time series under an autoregressive moving-average (ARMA) model of orders (p, q).

Usage

```
yhat_arma(x, ar, ma)
```

Arguments

x	Numeric vector. The observed time series.
ar	Numeric vector of AR coefficients $(\phi_1, \ldots, \phi_p)$, ordered from lag 1 to lag p. Supply `numeric(0)` for a pure MA model.
ma	Numeric vector of MA coefficients $(\theta_1, \ldots, \theta_q)$, ordered from lag 1 to lag q. Supply `numeric(0)` for a pure AR model.

Details

Residuals prior to the start index are initialised to zero. No intercept is included; center `x` before calling if the series has a non-zero mean. For a pure AR model, prefer `yhat_ar`, which is slightly more efficient.

Value

A numeric vector of the same length as `x`. Elements $1, \ldots, \max(p, q)$ are `NA`; element t contains

$$\hat{x}_t = \sum_{i=1}^{p} \phi_i x_{t-i} + \sum_{j=1}^{q} \theta_j \varepsilon_{t-j},$$

where residuals $\varepsilon_s = x_s - \hat{x}_s$ are accumulated recursively and initialised to zero.

See Also

`yhat_ar`, `poles_AR`, `btwar_fit`

Examples

```
x  <- as.numeric(arima.sim(list(ar = 0.6, ma = 0.4), n = 200))
yh <- yhat_arma(x, ar = 0.6, ma = 0.4)

# First max(p, q) values are NA
head(yh, 5)

# RMSE on the predictable portion
```

```
    start <- max(length(0.6), length(0.4)) + 1L
    sqrt(mean((x[start:length(x)] - yh[start:length(x)])^2))
```

Index