Package 'arimasel'

September 17, 2026

Title Cartesian Product-Based ARIMA Model Identification and Selection

Version 0.2.0

Description Provides an alternative algorithm for ARIMA and seasonal ARIMA model identification based on Cartesian products of user-supplied parameter sets. Rather than relying on ACF/PACF plots or stepwise search (as in auto.arima()), the package exhaustively evaluates every candidate (p,d,q)(P,D,Q)[m] combination in the requested index sets, ranks all converged models by AIC, AICc, BIC, and HQIC simultaneously, computes Akaike weights for model uncertainty quantification, supports exogenous regressors, produces ensemble forecasts, evaluates candidate models by rolling-origin (expanding window) cross-validation, and provides publication-quality diagnostic and comparison plots. A feature-based exploratory data analysis suite computes scale-free time series characteristics (trend and seasonal strength, spectral entropy, autocorrelation, lumpiness, stability) in the spirit of Hyndman, Wang and Laptev (2015), and a feature-guided automatic search narrows the Cartesian product model space before the exhaustive search runs. The algorithm is flexible, transparent, and widely applicable for quick, reproducible ARIMA model selection in both academic research and industry forecasting pipelines. Applications are demonstrated with Nigerian macroeconomic time series data.

License GPL-3

Encoding UTF-8

LazyData true

Depends R (>= 4.0.0)

Imports stats, graphics, grDevices, utils, parallel

Suggests tseries, forecast, testthat (>= 3.0.0), knitr, rmarkdown

VignetteBuilder knitr

RoxygenNote 7.3.1

URL https://github.com/Olawaleawe/arimasel

BugReports https://github.com/Olawaleawe/arimasel/issues

NeedsCompilation no

Author Olushina Olawale Awe [aut, cre] (ORCID:
<https://orcid.org/0000-0002-0442-4519>)

Maintainer Olushina Olawale Awe <olawaleawe@gmail.com>

Repository CRAN

Date/Publication 2026-09-17 12:00:02 UTC

Contents

`arimasel-package` *arimasel: Cartesian Product-Based ARIMA Model Identification and Selection*

Description

Modelling and forecasting of time series data are essential components of data science and big data analytics. The Box-Jenkins methodology has gained tremendous popularity for modelling univariate time series, yet ACF/PACF plots often provide conflicting and biased pictures, making proper ARIMA model identification difficult.

arimasel proposes an alternative algorithm based on the mathematical principles of Cartesian products of sets. Given user-supplied index sets P, D, and Q (optionally combined with seasonal sets Ps, Ds, Qs at period m), the algorithm exhaustively fits every candidate (p,d,q)(P,D,Q)[m] combination, ranks all converged models by four information criteria simultaneously (AIC, AICc, BIC, HQIC), and quantifies model uncertainty via Akaike weights. The package additionally supports exogenous regressors, rolling-origin cross-validation for genuine out-of-sample evaluation (`arima_cv`), STL-based seasonal-strength diagnostics (`seasonal_strength`, `suggest_D`), feature-based exploratory data analysis (`ts_eda`, `ts_features`), and a feature-guided automatic search (`smart_arima`) that narrows the Cartesian product model space using time series characteristics before the exhaustive search runs. The algorithm is flexible, modular, and widely applicable – competing with and complementing stepwise approaches such as auto.arima().

Author(s)

Maintainer: Olushina Olawale Awe <olawaleawe@gmail.com> (ORCID)

See Also

Useful links:

- https://github.com/Olawaleawe/arimasel
- Report bugs at https://github.com/Olawaleawe/arimasel/issues

---

`arima_cv` *Rolling-Origin Cross-Validation for a Cartesian Product ARIMA Result*

Description

Evaluates the genuine out-of-sample forecasting accuracy of the model order selected by `cart_arima` using rolling-origin (expanding window) cross-validation – the time-series analogue of k-fold cross-validation, sometimes called "backtesting" in industry forecasting practice. Starting from an initial training window, the model is re-fitted with the same (p, d, q) (and seasonal, and `xreg`, if used) order at each origin, an `h`-step-ahead forecast is produced, and the forecast error is recorded. The window then expands by `step` observations and the process repeats to the end of the series. This

complements in-sample information criteria (AIC/BIC/HQIC) with a direct, model-agnostic measure of forecasting performance – standard practice in modern applied and industrial time series work.

Usage

```
arima_cv(object, h = 1L, initial = NULL, step = 1L)
```

Arguments

`object`	An object of class `"cartARIMA"` (the model order, seasonal specification, and `xreg` of `object$best_model` are reused at every origin).
`h`	Positive integer forecast horizon evaluated at each origin. Default `1L`.
`initial`	Integer; size of the first training window. Defaults to `max(20L, floor(0.5 * n_obs))`.
`step`	Integer; number of observations by which the training window expands between successive origins. Default `1L`.

Value

An object of class `"arimaCV"` with components:

`errors` A long-format `data.frame` with columns `Origin`, `Horizon`, `Actual`, `Forecast`, `Error`.

`accuracy` A `data.frame`, one row per horizon, with `RMSE`, `MAE`, `MAPE`, and `N` (number of folds contributing to that horizon).

`order` The (p, d, q) order reused at every origin.

`seasonal` The seasonal order/period reused, or `NULL`.

`n_folds` Number of rolling origins evaluated.

`h`, `initial`, `step` As supplied.

`call` The matched call.

See Also

`cart_arima`

Examples

```
set.seed(123)
x <- arima.sim(list(ar = 0.6), n = 120)
res <- cart_arima(x, p_set = 0:2, d_set = 0:1, q_set = 0:2)
cv <- arima_cv(res, h = 1, initial = 80)
print(cv)
```

`arima_diagnose` *Residual Diagnostics for a Cartesian Product ARIMA Result*

Description

Produces a six-panel residual diagnostic plot for the best ARIMA model identified by `cart_arima`, and returns an object of class `"arimaDiagnose"` (with its own `print` method reporting the Shapiro-Wilk and Ljung-Box test results) for further inspection. The panels are:

1. Residuals over time.
2. Sample ACF of residuals.
3. Sample PACF of residuals.
4. Histogram of residuals with fitted Normal density.
5. Normal Q-Q plot.
6. Ljung-Box test p-values for lags 1 to `lags`.

Usage

```
arima_diagnose(object, lags = NULL, ...)
```

Arguments

`object`	An object of class `"cartARIMA"`.
`lags`	Positive integer; maximum lag for the Ljung-Box plot. Default: `min(20L, floor(object$n_obs / 5))`.
`...`	Additional graphical parameters passed to base `plot()`.

Value

An object of class `"arimaDiagnose"` with components `ljung_box` (a `data.frame`) and `shapiro_wilk` (the result of `shapiro.test()`), printed by its own `print` method.

Examples

```
set.seed(3)
x <- arima.sim(list(ar = 0.5), n = 80)
res <- cart_arima(x, p_set = 0:2, d_set = 0:1, q_set = 0:2)
arima_diagnose(res, lags = 15)
```

`arima_forecast` *Forecast from a Cartesian Product ARIMA Selection Result*

Description

Produces point forecasts and prediction intervals from a fitted `"cartARIMA"` object, either from the single best model or as an Akaike-weight-averaged ensemble across the top `top_k` models.

Usage

```
arima_forecast(
  object,
  h = 10L,
  newxreg = NULL,
  ensemble = FALSE,
  top_k = NULL,
  level = c(80, 95),
  plot = TRUE,
  ...
)
```

Arguments

`object`	An object of class `"cartARIMA"`.
`h`	Positive integer; forecast horizon. Default: `10L`.
`newxreg`	Optional matrix of future external regressors with `h` rows, required when the model in `object` was fitted with `xreg`.
`ensemble`	Logical; if `TRUE`, compute an Akaike-weighted ensemble forecast overlaid on the best-model forecast. Default: `FALSE`.
`top_k`	Integer; number of top models to include in the ensemble. Default: `min(5L, object$n_converged)`.
`level`	Numeric vector of confidence levels for prediction intervals (values in (0,100)). Default: `c(80, 95)`.
`plot`	Logical; if `TRUE` (default), produce a forecast plot.
`...`	Additional graphical arguments passed to `plot()`.

Details

The ensemble point forecast is the weighted mean

$$\hat{y}_t^{\text{ens}} = \sum_{i=1}^{K} \tilde{w}_i \, \hat{y}_t^{(i)},$$

where $\tilde{w}_i$ are the re-normalised Akaike weights over the top K models and $\hat{y}_t^{(i)}$ is the point forecast from model i. Prediction intervals are taken from the best model.

Value

A list (invisibly) with components:

`mean` Numeric vector of best-model point forecasts.

`lower` Matrix of lower PI bounds (`nrow = h`, `ncol = length(level)`).

`upper` Matrix of upper PI bounds.

`ensemble_mean` Numeric vector of ensemble forecasts (or `NULL` if `ensemble = FALSE`).

`level` The confidence levels used.

`model` The best-model string.

`top_k` Number of ensemble members.

Examples

```
set.seed(7)
x <- arima.sim(list(ar = 0.6), n = 80)
res <- cart_arima(x, p_set = 0:2, d_set = 0:1, q_set = 0:2)
arima_forecast(res, h = 8, ensemble = TRUE, top_k = 3)
```

---

arima_table *Format and Re-rank the ARIMA Model Comparison Table*

Description

Extracts, optionally re-ranks, and truncates the model comparison table stored inside a `"cartARIMA"` object.

Usage

```
arima_table(object, criterion = NULL, top_n = NULL)
```

Arguments

`object`	An object of class `"cartARIMA"`.
`criterion`	Character; criterion to rank by. One of `"AIC"`, `"AICc"`, `"BIC"`, `"HQIC"`. If `NULL` (default), the criterion used when `cart_arima()` was called is retained.
`top_n`	Integer; number of rows to return. If `NULL`, all converged models are returned.

Value

A `data.frame` with columns Model, p, d, q, (P, D, Q if a seasonal search was used), LogLik, AIC, AICc, BIC, HQIC, Rank, Delta, and Weight.

Examples

```
set.seed(1)
x <- arima.sim(list(ar = 0.5), n = 80)
res <- cart_arima(x, p_set = 0:2, d_set = 0:1, q_set = 0:2)
arima_table(res, criterion = "BIC", top_n = 5)
```

`arima_weights` *Akaike Weights and Evidence Ratios*

Description

Given a numeric vector of information criterion values (lower is better), computes Akaike weights $w_i = \exp(-\Delta_i/2) / \sum_j \exp(-\Delta_j/2)$ and evidence ratios $ER_i = w_{\text{best}}/w_i$ relative to the best (lowest-IC) model.

Usage

```
arima_weights(criteria_values, names_vec = NULL)
```

Arguments

`criteria_values`	A numeric vector of IC values, length >= 2.
`names_vec`	Optional character vector of model names; if `NULL`, `names(criteria_values)` or `"Model_i"` labels are used.

Value

A `data.frame` with columns `Model`, `IC_value`, `Delta`, `Weight`, and `EvidenceRatio`, sorted by ascending `Delta`.

Examples

```
ic_vals <- c(ARIMA_110 = 200.1, ARIMA_011 = 198.4, ARIMA_210 = 201.7)
arima_weights(ic_vals)
```

`cart_arima` *Cartesian Product (Seasonal) ARIMA Model Identification and Selection*

Description

The flagship function of **arimasel**. Given sets P, D, and Q of non-negative integers (and, optionally, seasonal sets P_s, D_s, Q_s at period m), the algorithm:

1. Computes the Cartesian product $P \times D \times Q$ (optionally combined with $P_s \times D_s \times Q_s$).
2. Fits every candidate (S)ARIMA model, exhaustively rather than stepwise, optionally in parallel.
3. Ranks all converged models by AIC, AICc, BIC, and HQIC.
4. Computes Akaike weights and a criterion-vote table.
5. Returns the best model object and a full comparison table.

Usage

```
cart_arima(
  x,
  p_set = 0:2,
  d_set = 0:1,
  q_set = 0:2,
  seasonal = NULL,
  xreg = NULL,
  criterion = c("AIC", "AICc", "BIC", "HQIC"),
  top_n = 10L,
  parallel = FALSE,
  n_cores = NULL,
  ...
)
```

Arguments

`x`	A numeric vector or a `ts` object representing the time series. Must have at least 10 observations.
`p_set`	Integer vector of AR orders $p \geq 0$. Default: `0:2`.
`d_set`	Integer vector of differencing orders $d \geq 0$. Default: `0:1`.
`q_set`	Integer vector of MA orders $q \geq 0$. Default: `0:2`.
`seasonal`	Optional named list describing a seasonal search, with elements `P`, `D`, `Q` (integer index sets, each defaulting to `0:1`) and `period` (the seasonal period m; required). When supplied, every combination of (p, d, q) and $(P, D, Q)_m$ is fitted as a seasonal ARIMA model. If `NULL` (default), a non-seasonal search is performed.

xreg	Optional numeric vector or matrix of external regressors, with number of rows/length equal to length(x), passed to every candidate model fit (regression with ARIMA errors).
criterion	Primary ranking criterion: one of "AIC" (default), "AICc", "BIC", or "HQIC".
top_n	Integer; maximum number of models to include in the abbreviated comparison table (accessible via obj$table). Default: 10L. The full table is always stored in obj$full_table.
parallel	Logical; if TRUE, candidate models are fitted in parallel using mclapply (ignored, with a serial fallback, on Windows where forking is unavailable). Default FALSE.
n_cores	Integer number of worker processes to use when parallel = TRUE. Defaults to max(1L, parallel::detectCores() - 1L).
...	Additional arguments passed to arima, e.g. method = "ML", include.mean = FALSE.

Value

An object of class "cartARIMA" with components:

best_model The fitted Arima object for the top-ranked model.

best_model_str Character string, e.g. "ARIMA(1,1,1)".

criterion The primary ranking criterion.

table A data.frame of the top top_n models with LogLik, AIC, AICc, BIC, HQIC, Delta, Weight, Rank.

full_table The complete ranked table (all converged models).

vote_table A data.frame showing how many criteria each model wins.

failed_models Character vector of ARIMA strings that failed to converge.

n_total Total number of candidate models attempted.

n_converged Number of models that converged successfully.

p_set, d_set, q_set The index sets used.

seasonal The validated seasonal specification, or NULL.

xreg The external regressors used, or NULL.

data The original time series (as a ts object).

n_obs Length of the original series.

call The matched call.

See Also

arima_table, arima_forecast, arima_diagnose, compare_arima, cp_sets, arima_cv

Examples

```
set.seed(42)
x <- arima.sim(list(ar = 0.7, ma = -0.3), n = 100)
result <- cart_arima(x, p_set = 0:2, d_set = 0:1, q_set = 0:2)
print(result)


## Seasonal search on monthly data
data(inflation_ng)
res_s <- cart_arima(inflation_ng, p_set = 0:1, d_set = 0:1, q_set = 0:1,
                    seasonal = list(P = 0:1, D = 0:1, Q = 0:1, period = 12))
print(res_s)
```

`compare_arima` *Compare cart_arima with auto.arima()*

Description

Fits the Cartesian product model and, if the **forecast** package is available, compares it with `auto.arima()` on in-sample criteria and optionally on a hold-out test set. Returns an object of class `"compareArima"` with its own `print` method.

Usage

```
compare_arima(
  x,
  p_set = 0:2,
  d_set = 0:1,
  q_set = 0:2,
  seasonal = NULL,
  criterion = "AIC",
  holdout = 0L,
  ...
)
```

Arguments

`x`	A numeric vector or `ts` object.
`p_set, d_set, q_set`	Index sets for `cart_arima`.
`seasonal`	Optional seasonal specification passed to `cart_arima` (see its `seasonal` argument). When supplied, `auto.arima()` is also fitted with `seasonal = TRUE` for a like-for-like comparison.
`criterion`	Primary criterion for `cart_arima`. Default: `"AIC"`.

holdout	Integer; number of observations to reserve as a test set for out-of-sample evaluation. 0L (default) disables holdout.
...	Additional arguments passed to cart_arima.

Value

An object of class "compareArima", a list with components cart_arima_result, comparison_table, and (if **forecast** is available) auto_arima_model and winner, printed by its own print method.

Examples

```
set.seed(11)
x <- arima.sim(list(ar = 0.6), n = 80)
compare_arima(x, p_set = 0:2, d_set = 0:1, q_set = 0:2)
```

---

cp_sets *Cartesian Product of ARIMA (or Seasonal ARIMA) Parameter Sets*

Description

Displays and returns the Cartesian product $P \times D \times Q$ (optionally combined with a seasonal product $P_s \times D_s \times Q_s$) of user-supplied non-negative integer sets, enumerating all candidate ARIMA(p, d, q) or seasonal ARIMA$(p, d, q)(P, D, Q)_m$ models. This is the first step of the **arimasel** algorithm.

Usage

```
cp_sets(p_set = 0:2, d_set = 0:1, q_set = 0:2, seasonal = NULL)
```

Arguments

p_set	Integer vector of autoregressive orders $p \geq 0$. Default: 0:2.
d_set	Integer vector of differencing orders $d \geq 0$. Default: 0:1.
q_set	Integer vector of moving-average orders $q \geq 0$. Default: 0:2.
seasonal	Optional named list with integer elements P, D, Q (seasonal index sets, each defaulting to 0:1) and period (the seasonal period m, required). When supplied, the seasonal Cartesian product is combined with $P \times D \times Q$.

Value

An object of class "cartProduct" with components p_set, d_set, q_set, seasonal, product (a data.frame), n_models, and model_strings, printed by its own print method.

Examples

```
cp_sets(0:1, 0:1, 0:1)
cp_sets(p_set = c(0,1,2), d_set = 1, q_set = c(0,1))
cp_sets(0:1, 0:1, 0:1, seasonal = list(P = 0:1, D = 0:1, Q = 0:1, period = 12))
```

exchange_ng *Monthly USD/NGN Exchange Rate – Nigeria, 2010-2023*

Description

A univariate monthly time series of the US dollar to Nigerian Naira (NGN) exchange rate from January 2010 to December 2023. The series incorporates structural breaks reflecting major exchange-rate policy changes in Nigeria, including the 2016 devaluation and the 2023 unification of exchange-rate windows.

Usage

```
exchange_ng
```

Format

A `ts` object of length 168 with `frequency = 12`, starting in January 2010.

Source

Simulated to reflect documented CBN/FMDQ exchange-rate regimes. See `vignette("arimasel-intro")`.

Examples

```
data(exchange_ng)
plot(exchange_ng, main = "USD/NGN Exchange Rate", ylab = "NGN per USD")
```

gdp_ng *Annual GDP Growth Rate – Nigeria, 1990-2023*

Description

A univariate annual time series of Nigeria's GDP growth rate (percentage) from 1990 to 2023, based on publicly available World Bank and IMF data patterns. Suitable for demonstrating ARIMA modelling on short annual series.

Usage

```
gdp_ng
```

Format

A `ts` object of length 34 with `frequency = 1`, starting in 1990.

Source

Simulated to reflect documented Nigerian GDP growth patterns (World Bank WDI, IMF WEO). See `vignette("arimasel-intro")`.

Examples

```
data(gdp_ng)
plot(gdp_ng, main = "Nigeria Annual GDP Growth Rate (%)", ylab = "%", xlab = "Year")
abline(h = 0, lty = 2)
```

---

hqic *Hannan-Quinn Information Criterion*

Description

Computes the Hannan-Quinn Information Criterion (HQIC) for a fitted ARIMA model object. HQIC is defined as

$$\mathrm{HQIC} = -2\hat{\ell} + 2k \ln(\ln n),$$

where $\hat{\ell}$ is the maximised log-likelihood, k is the total number of free parameters (coefficients plus σ^2), and n is the number of observations used in fitting.

Usage

```
hqic(object, k = 2)
```

Arguments

`object`	A fitted model of class `"Arima"` (from `arima`).
`k`	Numeric penalty multiplier; default `2`.

Value

A single numeric value.

References

Hannan, E. J. and Quinn, B. G. (1979). The determination of the order of an autoregression. *Journal of the Royal Statistical Society, Series B*, **41**(2), 190–195.

Examples

```
x <- cumsum(rnorm(80))
fit <- arima(x, order = c(1, 1, 0))
hqic(fit)
```

inflation_ng *Monthly CPI Inflation Rate – Nigeria, 2000-2023*

Description

A univariate monthly time series of Nigeria's consumer price index (CPI) year-on-year inflation rate (percentage) from January 2000 to December 2023, simulated to reflect documented NBS/CBN inflationary regimes including low-inflation (2000-2014), the commodity-shock era (2015-2017), and the post-COVID acceleration (2020-2023).

Usage

```
inflation_ng
```

Format

A `ts` object of length 288 with `frequency = 12`, starting in January 2000.

Source

Simulated to reflect documented Nigerian CPI data patterns (NBS, CBN Annual Reports). See `vignette("arimasel-intro")`.

Examples

```
data(inflation_ng)
plot(inflation_ng, main = "Nigeria Monthly CPI Inflation (%)", ylab = "%")
```

plot.arimaCV *Plot Method for arimaCV Objects*

Description

Plot Method for arimaCV Objects

Usage

```
## S3 method for class 'arimaCV'
plot(x, type = c("rmse", "forecasts"), ...)
```

Arguments

`x`	An object of class `"arimaCV"`.
`type`	Character; `"rmse"` (default) plots RMSE/MAE by horizon; `"forecasts"` overlays every rolling-origin forecast against the actual series (horizon 1 only).
`...`	Additional graphical parameters passed to base plot functions.

Value

Invisibly returns `x`.

`plot.cartARIMA` *Plot Method for cartARIMA Objects*

Description

Produces one of several publication-quality diagnostic plots for a fitted `"cartARIMA"` object.

Usage

```
## S3 method for class 'cartARIMA'
plot(
  x,
  type = c("criteria", "weights", "surface", "fitted", "seasonal"),
  criterion = NULL,
  top_n = 10L,
  ...
)
```

Arguments

`x`	An object of class `"cartARIMA"`.
`type`	Character; plot type. One of:
	`"criteria"` Horizontal bar chart of IC values for top models.
	`"weights"` Bar chart of Akaike model weights.
	`"surface"` Criterion heat map over the $p \times q$ grid (for the most frequently occurring d value in the top models).
	`"fitted"` Time series overlay: observed vs. fitted values.
	`"seasonal"` STL decomposition of the original series (trend, seasonal, remainder); requires `frequency(x$data) > 1`.
`criterion`	Character; criterion to display. Defaults to the one used in `cart_arima()`.
`top_n`	Integer; number of models to include in `"criteria"` and `"weights"` plots. Default: 10.
`...`	Additional graphical parameters passed to base plot functions.

Value

Invisibly returns `x`.

print.arimaCV *Print Method for arimaCV Objects*

Description

Print Method for arimaCV Objects

Usage

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

Arguments

x	An object of class "arimaCV".
...	Currently unused.

Value

Invisibly returns x.

print.arimaDiagnose *Print Method for arimaDiagnose Objects*

Description

Print Method for arimaDiagnose Objects

Usage

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

Arguments

x	An object of class "arimaDiagnose".
...	Currently unused.

Value

Invisibly returns x.

`print.cartARIMA` *Print Method and Accessors for cartARIMA Objects*

Description

`print` and accessor methods (`coef`, `residuals`, `fitted`, `AIC`, `BIC`, `logLik`) for objects of class `"cartARIMA"`, all of which delegate to `object$best_model`.

Usage

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

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

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

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

## S3 method for class 'cartARIMA'
AIC(object, ..., k = 2)

## S3 method for class 'cartARIMA'
BIC(object, ...)

## S3 method for class 'cartARIMA'
logLik(object, ...)
```

Arguments

`x`, `object`	An object of class `"cartARIMA"`.
`...`	Currently unused (accessors pass `...` through to the corresponding method on `object$best_model` where applicable).
`k`	Numeric AIC penalty multiplier (see `AIC`); default `2`.

Value

Invisibly returns `x` for `print`; the accessors return the corresponding value from `object$best_model`.

print.compareArima *Print Method for compareArima Objects*

Description

Print Method for compareArima Objects

Usage

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

Arguments

x	An object of class "compareArima".
...	Currently unused.

Value

Invisibly returns x.

print.stationarityTest *Print Method for stationarityTest Objects*

Description

Print Method for stationarityTest Objects

Usage

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

Arguments

x	An object of class "stationarityTest".
...	Currently unused.

Value

Invisibly returns x.

print.tsEda *Print Method for tsEda Objects*

Description

Print Method for tsEda Objects

Usage

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

Arguments

x	An object of class "tsEda".
...	Currently unused.

Value

Invisibly returns x.

seasonal_strength *Trend and Seasonal Strength via STL Decomposition*

Description

Computes the trend and seasonal strength measures of Wang, Smith and Hyndman (2006) from an STL decomposition (stl):

$$F_S = \max\left(0, 1 - \frac{\mathrm{Var}(R_t)}{\mathrm{Var}(S_t + R_t)}\right)$$

for seasonal strength, and analogously for trend strength using $T_t + R_t$. Values close to 1 indicate a strong, well-defined trend or seasonal pattern; values close to 0 indicate little or none. These are modern, model-free diagnostics that complement formal unit-root tests when deciding whether to include seasonal terms in a model search.

Usage

```
seasonal_strength(x, period = NULL, s.window = "periodic")
```

Arguments

x	A numeric vector or ts object with frequency > 1.
period	Optional integer seasonal period, used when x is not already a ts object with a known frequency.
s.window	Passed to stl; default "periodic".

Value

A list with components `trend_strength`, `seasonal_strength`, `period`, and `stl` (the underlying `stl` decomposition object).

References

Wang, X., Smith, K. A. and Hyndman, R. J. (2006). Characteristic-based clustering for time series data. *Data Mining and Knowledge Discovery*, **13**(3), 335–364.

Examples

```
data(inflation_ng)
seasonal_strength(inflation_ng)
```

`smart_arima` *Feature-Guided Automatic (Seasonal) ARIMA Search*

Description

A "smart" wrapper around `cart_arima` that first runs feature-based exploratory analysis (`ts_features`, `suggest_d`, `suggest_D`) to narrow the search space automatically, then performs the usual exhaustive Cartesian-product search within that narrowed space. This mirrors modern automatic forecasting pipelines that use time series features to steer model selection (Hyndman, Wang and Laptev, 2015; Talagala, Hyndman and Athanasopoulos, 2023) while retaining full transparency: every decision the function makes is printed, and the underlying `cart_arima` call is always available for a fully manual search.

Usage

```
smart_arima(
  x,
  p_set = 0:3,
  q_set = 0:3,
  P_set = 0:1,
  Q_set = 0:1,
  period = NULL,
  criterion = "AIC",
  top_n = 10L,
  parallel = FALSE,
  n_cores = NULL,
  ...
)
```

Arguments

x	A numeric vector or ts object.
p_set, q_set	Non-seasonal AR/MA index sets to search (the differencing order d_set is chosen automatically). Defaults 0:3 and 0:3.
P_set, Q_set	Seasonal AR/MA index sets, used only when a seasonal search is triggered. Defaults 0:1 and 0:1.
period	Optional integer seasonal period, used when x is not already a seasonal ts object. If NULL and x has no usable frequency, the search is non-seasonal.
criterion, top_n, parallel, n_cores, ...	Passed through to cart_arima.

Details

Specifically: the differencing order d is restricted to suggest_d(x) (plus one higher order as a safety margin), and, when x has a seasonal frequency greater than 1 (or period is supplied), a seasonal search is automatically enabled with D restricted to suggest_D(x).

Value

An object of class "cartARIMA", identical in structure to the return value of cart_arima, with an additional eda component holding the ts_features output used to guide the search.

References

Hyndman, R. J., Wang, E. and Laptev, N. (2015). Large-scale unusual time series detection. *2015 IEEE International Conference on Data Mining Workshop*, 1616–1619.

Talagala, T. S., Hyndman, R. J. and Athanasopoulos, G. (2023). Meta-learning how to forecast time series. *Journal of Forecasting*, **42**(6), 1476–1501.

See Also

cart_arima, ts_features, ts_eda

Examples

```
data(gdp_ng)
res <- smart_arima(gdp_ng, p_set = 0:1, q_set = 0:1)
print(res)


## Seasonal monthly series: seasonal search is enabled automatically
data(inflation_ng)
res_seas <- smart_arima(inflation_ng, p_set = 0:1, q_set = 0:1)
print(res_seas)
```

stationarity_test *Battery of Stationarity Tests*

Description

Runs the Augmented Dickey-Fuller (ADF) test on the supplied time series. If the **tseries** package is installed, the Phillips-Perron (PP) and KPSS tests are also performed. Returns an object of class `"stationarityTest"` (a `data.frame` of test statistics, approximate p-values, and per-test conclusions, plus a consensus conclusion) with its own `print` method.

Usage

```
stationarity_test(x, lags = NULL, alpha = 0.05)
```

Arguments

`x`	A numeric vector or `ts` object.
`lags`	Integer; ADF lag order. If `NULL` (default), chosen as $\lfloor (n-1)^{1/3} \rfloor$.
`alpha`	Significance level for the consensus decision. Default: 0.05.

Value

An object of class `"stationarityTest"`, printed by its own `print` method. Underlying columns `Test`, `Statistic`, `p_value`, and `Conclusion`, plus attributes `consensus` and `suggested_d`.

Examples

```
set.seed(5)
x <- cumsum(rnorm(80))
stationarity_test(x)
```

suggest_D *Recommend a Seasonal Differencing Order*

Description

Recommends a seasonal differencing order $D \in \{0, 1\}$ using the seasonal-strength heuristic of Wang, Smith and Hyndman (2006) via `seasonal_strength`: a seasonal difference is recommended when the seasonal strength F_S exceeds `threshold` (default `0.64`, the rule used in `forecast::nsdiffs(test = "seas")`).

Usage

```
suggest_D(x, period = NULL, threshold = 0.64, verbose = FALSE)
```

Arguments

x	A numeric vector or ts object.
period	Optional integer seasonal period (required if x is not already a seasonal ts object).
threshold	Seasonal-strength threshold above which a seasonal difference is recommended. Default 0.64.
verbose	Logical; if TRUE, print the seasonal strength. Default FALSE.

Value

A single integer, 0L or 1L.

References

Wang, X., Smith, K. A. and Hyndman, R. J. (2006). Characteristic-based clustering for time series data. *Data Mining and Knowledge Discovery*, **13**(3), 335–364.

Examples

```
data(inflation_ng)
suggest_D(inflation_ng)
```

---

suggest_d *Recommend a Differencing Order*

Description

Applies the ADF test battery via stationarity_test and returns the recommended integer differencing order $d \in \{0, 1, 2\}$. Differencing is applied iteratively until stationarity is detected or the maximum d is reached.

Usage

```
suggest_d(x, max_d = 2L, alpha = 0.05, verbose = FALSE)
```

Arguments

x	A numeric vector or ts object.
max_d	Maximum differencing order to consider. Default: 2L.
alpha	Significance level. Default: 0.05.
verbose	Logical; if TRUE, print test summaries. Default: FALSE.

Value

A single non-negative integer.

Examples

```
set.seed(9)
x <- cumsum(rnorm(80))
suggest_d(x)
```

`summary.arimaCV` *Summary Method for arimaCV Objects*

Description

Summary Method for arimaCV Objects

Usage

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

Arguments

`object`	An object of class `"arimaCV"`.
`...`	Currently unused.

Value

Invisibly returns `object`.

`summary.cartARIMA` *Summary Method for cartARIMA Objects*

Description

Summary Method for cartARIMA Objects

Usage

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

Arguments

`object`	An object of class `"cartARIMA"`.
`...`	Currently unused.

Value

Invisibly returns `object`.

`ts_eda` *Exploratory Data Analysis for a Time Series*

Description

Produces a console summary and a multi-panel exploratory plot for a univariate time series before model identification: the time plot, distribution, autocorrelation structure, and (if applicable) seasonal pattern. Intended as the first step of the **arimasel** workflow, complementing formal stationarity testing (`stationarity_test`) and feature extraction (`ts_features`).

Usage

```
ts_eda(x, period = NULL, lag.max = NULL, ...)
```

Arguments

`x`	A numeric vector or `ts` object.
`period`	Optional integer seasonal period, used when `x` is not already a seasonal `ts` object.
`lag.max`	Maximum lag shown in the ACF/PACF panels. Default `min(24L, floor(length(x) / 4))`.
`...`	Additional graphical parameters passed to the time plot.

Value

An object of class `"tsEda"` with components `features` (the output of `ts_features`), `stationarity` (the output of `stationarity_test`), and `suggested_d` /`suggested_D` (recommended differencing orders), printed by its own `print` method.

See Also

`ts_features`, `stationarity_test`, `smart_arima`

Examples

```
data(inflation_ng)
eda <- ts_eda(inflation_ng)
eda$features
```

`ts_features` *Feature-Based Time Series Summary*

Description

Computes a compact vector of numeric characteristics of a time series in the spirit of feature-based time series analysis (Hyndman, Wang and Laptev, 2015): a scale-free description of a series' trend, seasonality, autocorrelation, entropy, and stability that can be used to compare many series, cluster them, or guide automatic model search (see `smart_arima`).

Usage

```
ts_features(x, period = NULL)
```

Arguments

`x`	A numeric vector or `ts` object.
`period`	Optional integer seasonal period, used when `x` is not already a seasonal `ts` object and trend/seasonal strength are requested. If omitted and `x` has no usable frequency, seasonal features are returned as `NA`.

Value

A one-row `data.frame` with columns:

`n` Series length.

`mean`, `sd` Sample mean and standard deviation.

`skewness`, `kurtosis` Standardised third and fourth moments (excess kurtosis).

`trend_strength`, `seasonal_strength` From `seasonal_strength`; `NA` if no seasonal period is available.

`entropy` Spectral entropy in $[0, 1]$; values near 0 indicate a strongly predictable (structured) series, values near 1 indicate near white noise.

`acf1` First-lag autocorrelation of `x`.

`lumpiness`, `stability` Variance of the variances / means across ten equal-sized blocks of the series – large values flag heteroscedasticity or level shifts.

`adf_stat`, `adf_pvalue` Augmented Dickey-Fuller test statistic and approximate p-value (see `stationarity_test`).

References

Hyndman, R. J., Wang, E. and Laptev, N. (2015). Large-scale unusual time series detection. *2015 IEEE International Conference on Data Mining Workshop*, 1616–1619.

See Also

`ts_eda`, `smart_arima`

Examples

```
data(gdp_ng)
ts_features(gdp_ng)
```

Index