

Package ‘EconCausal’

September 17, 2026

Type Package

Title Causal Analysis for Macroeconomic Time Series (ECM-MARS, BSTS, Bayesian GLM-AR(1))

Version 1.0.4

Description

Implements three complementary pipelines for causal analysis on macroeconomic time series:

- (1) Error-Correction Models with Multivariate Adaptive Regression Splines (ECM-MARS),
- (2) Bayesian Structural Time Series (BSTS), and
- (3) Bayesian GLM with AR(1) errors validated with Leave-Future-Out (LFO).

Heavy backends (Stan) are optional and never used in examples or tests.

VignetteBuilder knitr

License GPL (>= 3)

Encoding UTF-8

Depends R (>= 4.1)

Imports brms, readxl, dplyr, tidyr, tibble, purrr, tseries, urca,
vars, earth, lmtest, sandwich, progressr, future, future.apply,
rlang, magrittr, parallel, stats, utils

Suggests bsts, BoomSpikeSlab, knitr, cmdstanr, rstan, RhpcBLASctl,
rmarkdown, testthat (>= 3.2.0)

URL <https://github.com/IsadoreNabi/EconCausal>

BugReports <https://github.com/IsadoreNabi/EconCausal/issues>

Additional_repositories <https://stan-dev.r-universe.dev>

Config/testthat/edition 3

Language en-US

NeedsCompilation no

Author José Mauricio Gómez Julián [aut, cre]

Maintainer José Mauricio Gómez Julián <isadore.nabi@pm.me>

Config/roxygen2/version 8.0.0

Repository CRAN

Date/Publication 2026-09-17 13:10:02 UTC

Contents

bglmar1	2
bsts_model	5
ecm_mars	8

Index	13
--------------	-----------

bglmar1	<i>Bayesian generalized linear model with AR(1) errors and leave-future-out validation</i>
---------	--

Description

Evaluates every directed pair between six circulation variables and seven production variables, in both directions, by comparing a Bayesian regression of the response on a linear time trend with and without lagged values of the predictor, both with first-order autoregressive errors. Each comparison is repeated over successive forecast windows, and a direction counts as supported in a window only when adding the lags improves both the predictive density and the point forecast. Use it to rank directed relationships by how consistently the predictor improves out-of-sample forecasts.

Usage

```
bglmar1(  
  data_path,  
  circ_vars,  
  prod_vars,  
  max_lag = 3,  
  initial_frac = 0.7,  
  initial_min = 90,  
  test_h = 12,  
  step_h = 12,  
  lfo_window = "sliding",  
  chains = 4,  
  parallel_chains = 4,  
  iter = 1500,  
  warmup = 750,  
  adapt_delta = 0.95,  
  trees = 12,  
  seed = 2025,  
  support_min = 0.6,  
  folds_min = 5,  
  sup_hi = 0.7,  
  sup_lo = 0.6,  
  backend = c("auto", "rstan", "cmdstanr")  
)
```

Arguments

data_path	Character scalar. Path to an Excel file with a date column and the series named in circ_vars and prod_vars. Column names are cleaned before matching: a leading as.numeric., trailing .NEW. or .1.588. and trailing dots are removed, and remaining dots become underscores. A date column named Month is used, or the first date-time column is renamed to Month.
circ_vars	Character vector of exactly six circulation variable names, after cleaning.
prod_vars	Character vector of exactly seven production variable names, after cleaning.
max_lag	Integer scalar ≥ 1 , default 3. Number of lags of the predictor used as regressors.
initial_frac	Numeric scalar in (0, 1), default 0.7. Fraction of the complete observations that sizes the first training window; the size actually used is the larger of that fraction and initial_min.
initial_min	Integer scalar ≥ 1 , default 90. Lower bound of the first training window.
test_h	Integer scalar ≥ 1 , default 12. Number of observations in the test block of each validation window; one year with monthly data.
step_h	Integer scalar ≥ 1 , default 12. Advance of the end of the training sample between windows.
lfo_window	"sliding" (default) or "expanding". With "sliding" the training sample keeps at most the initial number of rows and moves forward; with "expanding" it grows from the first observation.
chains	Integer scalar ≥ 1 , default 4. Number of MCMC chains per model.
parallel_chains	Integer scalar ≥ 1 , default 4. Number of chains run in parallel, passed as the number of cores of the sampler.
iter	Integer scalar $>$ warmup, default 1500. Total iterations per chain, warmup included.
warmup	Integer scalar ≥ 0 and $<$ iter, default 750. Warmup iterations per chain.
adapt_delta	Numeric scalar in (0, 1), default 0.95. Target acceptance rate of the sampler.
trees	Integer scalar ≥ 1 , default 12. Maximum tree depth of the sampler.
seed	Integer scalar, default 2025. Base seed; the model without lags is fitted with seed + 101 and the model with lags with seed + 202, in every window and for every pair.
support_min	Numeric scalar, default 0.6. Not used by the function; the support thresholds applied to the returned rankings are sup_hi and sup_lo.
folds_min	Integer scalar ≥ 0 , default 5. Minimum number of evaluated windows required by winners_070 and winners_060.
sup_hi	Numeric scalar in [0, 1], default 0.7. Support threshold of winners_070.
sup_lo	Numeric scalar in [0, 1], default 0.6. Support threshold of winners_060.
backend	Character scalar, one of "auto" (default), "rstan" or "cmdstanr". Engine used to fit the models. With "auto" the function uses rstan when it is installed and otherwise a working cmdstanr. The option EconCausal.backend, when set, overrides this argument, and the function stops when neither engine is available.

Details

For each direction the predictor is lagged `max_lag` times and incomplete rows are dropped. A direction is skipped when the remaining rows are fewer than `initial_min + test_h + max_lag + 5`. The first training sample holds the larger of `initial_min` rows and `initial_frac` of the remaining rows, each test block holds the next `test_h` observations, and the end of the training sample advances by `step_h` until the last observation is reached. In each window the response, the time index and the lags are standardised with the training mean and standard deviation, the response is returned to its original scale before the metrics are computed, and a lag whose training standard deviation is zero is dropped from that window.

Two models are fitted to the standardised response with Gaussian errors, a normal prior on the regression coefficients, a Student *t* prior on the intercept and an exponential prior on the residual scale: a model on the standardised time index alone and a model that adds the standardised lags of the predictor. Both carry a first-order autoregressive term on a single series indexed by time, so the comparison isolates the contribution of the lags. For the test block the window records the expected log predictive density of each observation, obtained as the log of the mean of the posterior likelihood draws and summed over the block, together with RMSE, MAE, symmetric MAPE and the coefficient of determination of the posterior mean forecast. A window is a win when the model with the lags has both a higher predictive density and a lower RMSE, and support is the share of wins among the evaluated windows.

Value

A list with four elements. `bench_bayes` is a tibble with one row per direction and the columns `pair` (written as "X -> Y"), `folds`, `folds_pass`, `support` (`folds_pass / folds`), `ELPD_diff_mean`, `RMSE_diff_mean`, `RMSE_full_mean`, `RMSE_base_mean`, `MAE_full_mean`, `MAE_base_mean`, `sMAPE_full_mean`, `sMAPE_base_mean`, `R2_full_mean` and `R2_base_mean`, sorted by decreasing `support`, decreasing `ELPD_diff_mean` and increasing `RMSE_diff_mean`. `winners_070` and `winners_060` are the rows of that tibble with at least `folds_min` evaluated windows, finite support of at least `sup_hi` or `sup_lo`, positive `ELPD_diff_mean` and negative `RMSE_diff_mean`; their names do not change with `sup_hi` and `sup_lo`. `rank_out` is a list with all, the sorted benchmark, `winners_hi` and `winners_lo`, the two tibbles just described, and `ratios`, with `pair`, `support`, `ELPD_diff_mean`, `RMSE_diff_mean`, `RMSE_ratio` and `MAE_ratio`, sorted by increasing `RMSE_ratio`. A direction with too few observations, without any validation window or without any evaluated window returns a row with `folds` equal to zero and missing metrics.

Methodological notes

`support` requires improvement in probabilistic fit and in point error at the same time; a predictor that improves only one of them in a window does not count. The predictive density is computed from the posterior likelihood draws of the test observations, with group-level terms excluded, and not from a Gaussian approximation to the forecast. Standardising the response, the time index and the lags with training statistics only keeps test-window information out of the fit. The linear time trend enters both models, so a direction cannot be supported by the trend itself. Windows in which the response does not vary, or in which a fit or a predictive evaluation fails, are dropped and do not count in `folds`, so `support` is a proportion over evaluated windows and not over possible ones. Both seeds are derived from `seed` and not from the window or the `pair`, which makes a run reproducible for a given seed. The 84 directions are ranked under a common criterion and without any correction

for multiplicity: support describes how consistent a single direction is across windows, and is not a joint test.

Dependencies

brms builds the formulas and the priors, fits the models, and produces the pointwise likelihood and the posterior expectation of the test block; the fitting engine is rstan or cmdstanr, which are suggested packages chosen at run time and not imported. readxl reads the data; dplyr, tidyr and tibble build lags, windows and summaries; magrittr supplies the pipe and rlang the data pronoun; stats provides the standard deviation, the Gaussian family and the formula built for each window; parallel counts the available cores for the duration of the call; and utils writes the ranking tables when the internal ranking helper is given output paths, which the defaults do not do.

References

- Bürkner, P.-C. (2017). brms: An R package for Bayesian multilevel models using Stan. *Journal of Statistical Software*, 80(1), 1–28. doi:10.18637/jss.v080.i01
- Bürkner, P.-C., Gabry, J., & Vehtari, A. (2020). Approximate leave-future-out cross-validation for Bayesian time series models. *Journal of Statistical Computation and Simulation*, 90(14), 2499–2523. doi:10.1080/00949655.2020.1783262

See Also

`bsts_model()`, `ecm_mars()`; the vignettes `bglmar1-eng` and `bglmar-esp`.

Examples

```
## Not run:
result <- bglmar1(
  data_path = file.path(tempdir(), "data.xlsx"),
  circ_vars = c("TC_SPOT_CAN_US", "TC_SPOT_US_CAN", "TC_SPOT_US_REMB",
    "IPC", "TdI_LdelT", "TasaDescuento"),
  prod_vars = c("ValorExportaciones", "Real_Net_Profit",
    "RealSocialConsumptionPerWorker2017", "RealWage_PPP2017",
    "CapitalStock_PPP2017", "LaborProductivity_PPP2017",
    "InvestmentPerWorker_PPP2017"),
  backend = "auto"
)

## End(Not run)
```

Description

Evaluates every directed pair between six circulation variables and seven production variables, in both directions, by comparing a structural time series model of the response with and without lagged values of the predictor. Each comparison is repeated over successive forecast windows, and a direction counts as supported in a window only when adding the predictor improves both the predictive density and the point forecast. Use it to rank directed relationships by how consistently the predictor improves out-of-sample forecasts.

Usage

```
bsts_model(
  data_path,
  circ_vars,
  prod_vars,
  max_lag = 6,
  lfo_init_frac = 0.8,
  lfo_h = 6,
  lfo_step = 6,
  niter = 2000,
  burn = 500,
  seed = 123,
  seasonality = NULL,
  support_min = 0.6,
  folds_min = 5,
  sup_hi = 0.7,
  sup_lo = 0.6,
  out_dir = NULL
)
```

Arguments

<code>data_path</code>	Character scalar. Path to an Excel file with a date column and the series named in <code>circ_vars</code> and <code>prod_vars</code> . Column names are cleaned before matching: a leading <code>as.numeric.</code> , trailing <code>.NEW.</code> or <code>.1.588.</code> and trailing dots are removed, and remaining dots become underscores. A date column named <code>Month</code> is used, or the first date-time column is renamed to <code>Month</code> .
<code>circ_vars</code>	Character vector of exactly six circulation variable names, after cleaning.
<code>prod_vars</code>	Character vector of exactly seven production variable names, after cleaning.
<code>max_lag</code>	Integer scalar ≥ 1 , default 6. Number of lags of the predictor used as regressors.
<code>lfo_init_frac</code>	Numeric scalar in (0, 1), default 0.8. Fraction of the complete observations in the first training window (at least 30 rows).
<code>lfo_h</code>	Integer scalar ≥ 1 , default 6. Forecast horizon of each validation window.
<code>lfo_step</code>	Integer scalar ≥ 1 , default 6. Advance between validation windows.
<code>niter</code>	Integer scalar $> \text{burn}$, default 2000. Number of MCMC iterations per model.
<code>burn</code>	Integer scalar ≥ 0 and $< \text{niter}$, default 500. MCMC iterations discarded before forecasting.

seed	Integer scalar, default 123. Base seed; model fits and forecasts in window i use $\text{seed} + i$ (model without predictor) and $\text{seed} + 1000 + i$ (model with predictor).
seasonality	NULL (default) or an integer scalar ≥ 2 giving the number of seasons of a seasonal state component.
support_min	Numeric scalar in $[0, 1]$, default 0.6. Minimum support for pass_support.
folds_min	Integer scalar ≥ 0 , default 5. Minimum number of evaluated windows for pass_support.
sup_hi	Numeric scalar in $[0, 1]$, default 0.7. Support threshold of winners_ss_070.
sup_lo	Numeric scalar in $[0, 1]$, default 0.6. Support threshold of winners_ss_060.
out_dir	NULL (default) or a character scalar. If given, the directory is created and the three ranking tables are written there as CSV files.

Details

For each direction the predictor is lagged `max_lag` times and incomplete rows are dropped. Validation windows use an expanding training sample and a test block of `lfo_h` observations that always lies after the training sample. In each window the lags are standardised with the training mean and standard deviation, and two models are fitted to the training response with the same state specification: a model without regressors and a model that adds a regression on the standardised lags with a spike-and-slab prior. The prior is built on the design matrix including the intercept, with an expected model size equal to the number of columns of that matrix capped at 5, prior information weight 0.01 and diagonal shrinkage 0.5.

Forecasts use the posterior predictive draws after burn. For each model the window records the Gaussian log predictive density of the test observations given the forecast mean and standard deviation, RMSE, MAE and the coverage of central 80 and 95 percent Gaussian intervals built from the same mean and standard deviation. A window is a win when the model with the predictor has both higher log predictive density and lower RMSE. Two state specifications are evaluated, local level ("LL") and local linear trend ("LLT"), each with the optional seasonal component, and the one with the highest dELPD_mean, then support, then dRMSE_mean, is reported for the direction.

Value

A list with four tibbles, or NULL with a warning when no direction produced a result: `summaries_ss`, one row per direction with the selected specification `spec` ("LL" or "LLT"), `folds`, `wins`, `support` (`wins / folds`), `dELPD_mean`, `dRMSE_mean`, `dMAE_mean`, `RMSE_base_mean`, `RMSE_full_mean`, `MAE_base_mean`, `MAE_full_mean`, `cover80_mean`, `cover95_mean`, `Y` and `X`; `rank_ss_all`, the same rows with `pair` and `pass_support`, sorted by decreasing `support`, `dELPD_mean` and `dRMSE_mean`; `winners_ss_070` and `winners_ss_060`, the rows of `rank_ss_all` with `pass_support`, `support` at least `sup_hi` or `sup_lo`, and positive `dELPD_mean` and `dRMSE_mean`. The names `winners_ss_070` and `winners_ss_060` do not change with `sup_hi` and `sup_lo`.

Methodological notes

support requires improvement in probabilistic fit and in point error at the same time; a predictor that improves only one of them in a window does not count. The log predictive density is a Gaussian approximation computed from the forecast mean and standard deviation, not the exact predictive density of the draws. Standardising lags with training statistics only keeps test-window information

out of the fit. Models without regressors are fitted on the response vector, as `bsts` documents for models with no regression component. Tying each forecast seed to the seed of its fit makes the results reproducible for a given seed; with a different seed, support can change, especially with few MCMC iterations.

Dependencies

`bsts` builds the state specification, runs the MCMC and produces the forecasts; `BoomSpikeSlab` builds the spike-and-slab prior. Both are suggested packages that this function requires; it stops with an informative message when either is missing. `readxl` reads the data; `dplyr`, `tidyr`, `tibble` and `purrr` build lags, windows and summaries; `stats` and `utils` provide the metrics and the CSV export.

References

- Bürkner, P.-C., Gabry, J., & Vehtari, A. (2020). Approximate leave-future-out cross-validation for Bayesian time series models. *Journal of Statistical Computation and Simulation*, 90(14), 2499–2523. doi:10.1080/00949655.2020.1783262
- Scott, S. L., & Varian, H. R. (2014). Predicting the present with Bayesian structural time series. *International Journal of Mathematical Modelling and Numerical Optimisation*, 5(1/2), 4–23. doi:10.1504/IJMMNO.2014.059942

See Also

`ecm_mars()`, `bg1mar1()`; the vignettes `bsts-eng` and `bsts-esp`.

Examples

```
## Not run:
result <- bsts_model(
  data_path = file.path(tempdir(), "data.xlsx"),
  circ_vars = c("TC_SPOT_CAN_US", "TC_SPOT_US_CAN", "TC_SPOT_US_REMB",
    "IPC", "TdI_LdelT", "TasaDescuento"),
  prod_vars = c("ValorExportaciones", "Real_Net_Profit",
    "RealSocialConsumptionPerWorker2017", "RealWage_PPP2017",
    "CapitalStock_PPP2017", "LaborProductivity_PPP2017",
    "InvestmentPerWorker_PPP2017")
)

## End(Not run)
```

Description

Evaluates every directed pair between a set of circulation variables and a set of production variables, in both directions, and measures how often a non-linear error-correction forecast is both statistically admissible and predictive across successive temporal windows. Each window first screens the pair for integration, cointegration and a significant error-correction term; only windows that pass that screen fit a Multivariate Adaptive Regression Splines (MARS) forecaster. Use it to rank directed relationships by the temporal stability of their error-correction structure rather than by a single full-sample fit.

Usage

```
ecm_mars(
  data_path,
  circ_vars,
  prod_vars,
  cointeg_rule = "either",
  eg_p_cutoff = 0.05,
  ecm_p_cutoff = 0.05,
  lag_max_ecm = 4,
  min_tr = 20,
  min_te = 8,
  rolling_cv_enable = TRUE,
  rolling_cv_window = "sliding",
  rolling_cv_initial_frac = 0.8,
  rolling_cv_initial_min = 40,
  rolling_cv_test = 12,
  rolling_cv_step = 12,
  nested_tune = TRUE,
  nested_initial_f = 0.6,
  nested_test = 6,
  nested_step = 3,
  mars_grid = expand.grid(degree = c(1, 2), nk = c(15, 25, 35, 50, 65)),
  support_min = 0.75,
  folds_min_abs = 5,
  parallel_enable = TRUE,
  parallel_workers = max(1, parallel::detectCores() - 1)
)
```

Arguments

<code>data_path</code>	Character scalar. Path to an Excel file with exactly 14 columns in this order: a date column followed by the 13 series ER.SPOT.CAN.US, ER.SPOT.US.CAN, ER.SPOT.US.REMB, CPI, TreasuryBonds10y, FedDiscountRate, Exports, RealNetProfit, RealSocialConsumptionPerWorker2017, RealWagePPP2017, CapitalStockPPP2017, LaborProductivityPPP2017, InvestmentPerWorkerPPP2017. The columns are renamed to these names by position, whatever their names in the file.
<code>circ_vars</code>	Character vector of circulation variable names, a subset of the 13 names above. Names not present are dropped.

prod_vars	Character vector of production variable names, a subset of the 13 names above. Names not present are dropped.
cointeg_rule	Character scalar, "either" (default) or "both": whether cointegration requires the Engle-Granger/Phillips-Ouliaris screen or the Johansen screen, or both of them.
eg_p_cutoff	Numeric scalar in (0, 1), default 0.05. Significance level of the residual unit-root test and of the Phillips-Ouliaris test.
ecm_p_cutoff	Numeric scalar in (0, 1), default 0.05. One-sided significance level for the error-correction coefficient being negative.
lag_max_ecm	Integer scalar ≥ 1 , default 4. Largest lag order of the differenced series considered in the linear error-correction model.
min_tr	Integer scalar ≥ 1 , default 20. Minimum number of complete training rows required to fit MARS in a window.
min_te	Integer scalar ≥ 1 , default 8. Minimum number of complete test rows required to evaluate a window.
rolling_cv_enable	Logical scalar, default TRUE. If FALSE, a single split with the first 75 percent of observations for training is used instead of rolling windows.
rolling_cv_window	Character scalar, "sliding" (default) or "expanding". Training window type of the outer validation.
rolling_cv_initial_frac	Numeric scalar in (0, 1), default 0.8. Fraction of the sample in the first training window.
rolling_cv_initial_min	Integer scalar ≥ 1 , default 40. Minimum size of the first training window.
rolling_cv_test	Integer scalar ≥ 1 , default 12. Test horizon of each outer window.
rolling_cv_step	Integer scalar ≥ 1 , default 12. Advance between outer windows.
nested_tune	Logical scalar, default TRUE. If TRUE, MARS degree and number of terms are chosen in each outer training window by an inner validation; if FALSE, degree 2 and 25 terms are used.
nested_initial_f	Numeric scalar in (0, 1), default 0.6. Fraction of the outer training window in the first inner training window (at least 30 rows).
nested_test	Integer scalar ≥ 1 , default 6. Test horizon of each inner window.
nested_step	Integer scalar ≥ 1 , default 3. Advance between inner windows.
mars_grid	Data frame with numeric columns degree and nk, default <code>expand.grid(degree = c(1, 2), nk = c(15, 25, 35, 50, 65))</code> . Candidate MARS settings.
support_min	Numeric scalar in (0, 1], default 0.75. Minimum proportion of outer windows that must pass the screen for <code>pass_support</code> .
folds_min_abs	Integer scalar ≥ 0 , default 5. Minimum absolute number of outer windows that must pass the screen for <code>pass_support</code> .

`parallel_enable`

Logical scalar, default TRUE. Evaluate directions in parallel R sessions.

`parallel_workers`

Integer scalar ≥ 1 , default `max(1, parallel::detectCores() - 1)`. Number of parallel sessions when `parallel_enable = TRUE`.

Details

In every training window the response Y and the predictor X must both be integrated of order one: augmented Dickey-Fuller tests with lag order chosen by AIC fail to reject a unit root in levels (with drift or with trend, 10 percent) and reject it in first differences. The VAR lag order is chosen by the Schwarz criterion. Cointegration is assessed with the Johansen trace test (constant, then trend, 5 percent) and with the Engle-Granger residual test or the Phillips-Ouliaris test at `eg_p_cutoff`, combined by `cointeg_rule`. A linear error-correction model is then fitted with the lag order that minimises BIC, preferring the first order whose residuals pass a Ljung-Box test at lag 12, and the error-correction coefficient must be negative with a one-sided p-value below `ecm_p_cutoff` using Newey-West standard errors. The screen is deliberately strict because MARS is flexible enough to fit spurious non-linear structure between non-cointegrated series.

Windows that pass fit MARS to the first difference of Y on the lagged error-correction term, the first difference of X , the first lags of both differences and the second lag of the difference of Y , and forecast the level as the previous level plus the forecast difference. With `nested_tune = TRUE`, the MARS setting with the lowest mean RMSE across inner expanding windows is used; the inner validation never sees the outer test window.

`pass_support` requires `folds_proceed  $\geq$  max(folds_min_abs, ceiling(support_min * folds))`, so both a relative and an absolute number of admissible windows are needed.

Value

A data.frame with one row per direction and columns pair (" $X \rightarrow Y$ "), Y , X , folds (outer windows), folds_proceed (windows that passed the screen), RMSE, MAE, MAPE, SMAPE, R^2 , TheilU, bias_prop, var_prop, cov_prop (forecast metrics averaged over the windows that passed), support (folds_proceed / folds), pass_support (logical), R^2_{stab} ($R^2 * support$) and U_{stab} (TheilU / support), sorted by decreasing R^2 and then by pair.

Methodological notes

support measures temporal stability of the screen, not forecast accuracy: a direction can forecast well in the few windows where it passes and still have low support. R^2 and the other metrics are averaged only over windows that passed, so they must be read together with folds_proceed. The error-correction term uses the Engle-Granger long-run coefficients estimated in the training window, so no test-window information enters the forecaster. Parallel evaluation uses reproducible random streams (`future.seed = TRUE`). The function restores the caller's future plan and the BLAS/OpenMP thread settings of the calling R session on exit, also when it stops with an error. The thread limit applies to the calling session only: the parallel worker sessions are separate R processes and keep their own BLAS and OpenMP settings. A thread count that cannot be read (for example OpenMP when R_{hpc}BLASct1 was built without it) is left untouched.

Dependencies

readxl reads the data; dplyr orders observations and builds lags; urca provides the augmented Dickey-Fuller and Johansen tests; vars selects the VAR lag order; tseries provides the Phillips-Ouliaris test; lmtest and sandwich give the Newey-West test of the error-correction coefficient; earth fits MARS; future and future.apply evaluate directions in parallel; progressr reports progress; RhpCBLASctl, if installed, keeps BLAS and OpenMP single-threaded in the calling R session during the evaluation.

References

- Engle, R. F., & Granger, C. W. J. (1987). Co-integration and error correction: Representation, estimation, and testing. *Econometrica*, 55(2), 251–276. doi:10.2307/1913236
- Friedman, J. H. (1991). Multivariate adaptive regression splines. *The Annals of Statistics*, 19(1), 1–67. doi:10.1214/aos/1176347963
- Johansen, S. (1988). Statistical analysis of cointegration vectors. *Journal of Economic Dynamics and Control*, 12(2–3), 231–254. doi:10.1016/01651889(88)900413
- 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
- Phillips, P. C. B., & Ouliaris, S. (1990). Asymptotic properties of residual based tests for cointegration. *Econometrica*, 58(1), 165–193. doi:10.2307/2938339

See Also

`bsts_model()`, `bglmar1()`; the vignettes `ecm-mars-eng` and `ecm-mars-esp`.

Examples

```
## Not run:
result <- ecm_mars(
  data_path = file.path(tempdir(), "data.xlsx"),
  circ_vars = c("ER.SPOT.CAN.US", "ER.SPOT.US.CAN", "ER.SPOT.US.REMB",
    "CPI", "TreasuryBonds10y", "FedDiscountRate"),
  prod_vars = c("Exports", "RealNetProfit", "RealSocialConsumptionPerWorker2017",
    "RealWagePPP2017", "CapitalStockPPP2017",
    "LaborProductivityPPP2017", "InvestmentPerWorkerPPP2017")
)

## End(Not run)
```

Index

bglmar1, [2](#)
bglmar1(), [8](#), [12](#)
bsts_model, [5](#)
bsts_model(), [5](#), [12](#)

ecm_mars, [8](#)
ecm_mars(), [5](#), [8](#)