Package 'CRAFT'

August 8, 2026

Type Package

Title Conditional Regime Analog Forecasting with Trajectories

Version 0.1.0

Maintainer Giancarlo Vercellino <giancarlo.vercellino@gmail.com>

Description Tools for Conditional Regime Analog Forecasting with Trajectories. The package builds lag and lead trajectory embeddings, detects regimes with singular value decomposition and changepoint analysis, estimates transition probabilities between regimes, samples future-trajectory analogues, and fits smooth empirical forecast distributions.

License GPL-3

RoxygenNote 7.3.3

Depends R (>= 4.1.0)

Imports changepoint, graphics, grDevices, parallel, stats, utils

Encoding UTF-8

Suggests knitr, rmarkdown, testthat (>= 3.0.0)

VignetteBuilder knitr

Config/testthat/edition 3

URL https://rpubs.com/giancarlo_vercellino/craft

NeedsCompilation no

Author Giancarlo Vercellino [aut, cre, cph]

Repository CRAN

Date/Publication 2026-08-08 12:50:20 UTC

Contents

CRAFT-package 2
craft_fit 2
craft_predict 5
denoiser 6
regime_sampler 8

smooth_quantile_distribution 9
svd_changepoint_regimes 10
time_volume_weights 13
trajectory_embedding 14
trajectory_forecast 14
trajectory_reframer 16

Index **18**

`CRAFT-package` *Conditional Regime Analog Forecasting with Trajectories*

Description

CRAFT provides tools for building lag and lead trajectory embeddings from multivariate time series, detecting regimes with singular value decomposition and changepoint analysis, estimating regime-transition probabilities, sampling future-trajectory analogues, and fitting smooth empirical forecast distributions.

Details

The main workflow is implemented by `craft_fit()` and `craft_predict()`. Lower-level helpers are also exported for users who want to build custom workflows: `trajectory_embedding()`, `svd_changepoint_regimes()`, `regime_sampler()`, `trajectory_forecast()`, `smooth_quantile_distribution()`, `denoiser()`, and `time_volume_weights()`.

Author(s)

Giancarlo C.

See Also

`craft_fit()`, `craft_predict()`, `trajectory_embedding()`, `svd_changepoint_regimes()`

`craft_fit` *Fit a Trajectory-Aware Regime Forecast*

Description

Runs the CRAFT end-to-end workflow: trajectory embedding, past and future regime detection, transition probability estimation, future-trajectory sampling, correlation diagnostics, distribution fitting, and optional internal backtesting.

Usage

```
craft_fit(
  series,
  volumes = NULL,
  window = 20,
  n_draws = 1000,
  n_factors = 3,
  var_threshold = 0.95,
  horizon = NULL,
  seed = 42,
  transition_alpha = 1,
  transition_min_count = 1,
  transition_key_sep = "\r",
  valid_frac = 0.2,
  valid_split = c("last", "random", "blocked"),
  n_testing = 0L,
  backtest_parallel = TRUE,
  backtest_workers = NULL,
  backtest_min_train = NULL,
 sampler_weight = c("uniform", "weighted_bal_acc", "bal_acc", "weighted_acc", "acc"),
  tail_penalty_lambda = 0,
  tail_penalty_threshold = 3,
  tail_penalty_power = 2,
  tail_penalty_aggregation = c("mean", "max", "sum"),
  tail_penalty_scope = c("forecast", "all"),
  volume_alpha = 1,
  max_regimes_per_factor = 6,
  factor_selection = "fixed",
  min_segment = 30L,
  past_regime_args = list(),
  future_regime_args = list(),
  return_train_probs = TRUE,
  verbose = TRUE
)
```

Arguments

series	A data frame or matrix of numeric time series in columns.
volumes	Optional matrix or data frame of volumes with the same dimensions as series.
window	Trajectory window length used for lag and lead embeddings.
n_draws	Number of future-trajectory rows to sample.
n_factors	Number of SVD factors when fixed factor selection is used.
var_threshold	Cumulative variance threshold for variance-based factor selection.
horizon	Optional forecast horizon. If NULL, all horizons from 1 to window are retained.
seed	Optional random seed.

`transition_alpha`
: Additive smoothing parameter for transition probabilities.

`transition_min_count`
: Minimum count for memorized transition rows before fallback to global probabilities.

`transition_key_sep`
: Internal separator used for transition-state keys.

`valid_frac`
: Fraction of rows reserved for transition-model validation.

`valid_split`
: Validation split strategy.

`n_testing`
: Number of internal backtest origins. Use zero to skip backtesting.

`backtest_parallel`
: Logical. If `TRUE`, run internal backtests in parallel when possible.

`backtest_workers`
: Optional number of parallel workers.

`backtest_min_train`
: Optional minimum training length for backtest origins.

`sampler_weight`
: Metric used to weight target regimes during future-trajectory sampling.

`tail_penalty_lambda`
: Strength of the optional tail penalty applied to candidate rows.

`tail_penalty_threshold`
: Robust z-score threshold for tail penalties.

`tail_penalty_power`
: Power applied to excess robust z-scores.

`tail_penalty_aggregation`
: Aggregation used for row-level tail penalties.

`tail_penalty_scope`
: Whether the tail penalty uses forecast-horizon columns or all forward-trajectory columns.

`volume_alpha`
: Exponent for volume-derived time weights.

`max_regimes_per_factor`
: Maximum regimes per SVD factor.

`factor_selection`
: SVD factor selection strategy passed to `svd_changepoint_regimes()`.

`min_segment`
: Minimum segment length for changepoint regime detection.

`past_regime_args`
: Additional arguments passed to past-trajectory regime detection.

`future_regime_args`
: Additional arguments passed to future-trajectory regime detection.

`return_train_probs`
: Logical. If `TRUE`, include training transition probabilities.

`verbose`
: Logical. If `TRUE`, emit progress messages.

Value

An object of class `craft_fit`, represented as a list containing trajectories, regime models, labels, transition probabilities, sampled trajectory draws, fitted return and level distributions, diagnostics, classifier details, optional backtest results, validation metrics, and the information needed by `craft_predict()`, including `trajectory_draws`.

Examples

```
set.seed(1)
series <- data.frame(
  a = cumprod(1 + rnorm(80, 0.001, 0.01)),
  b = cumprod(1 + rnorm(80, 0.0005, 0.012))
)
fit <- craft_fit(
  series,
  window = 5,
  n_draws = 50,
  n_factors = 1,
  min_segment = 5,
  max_regimes_per_factor = 3,
  n_testing = 0,
  verbose = FALSE
)
names(fit$return_dists)
```

---

craft_predict *Predict from a Fitted CRAFT Model*

Description

Uses a fitted CRAFT model to transform the latest trajectory, estimate future regime probabilities, sample future-trajectory analogues, and fit smooth forecast distributions.

Usage

```
craft_predict(
  object,
  newdata = NULL,
  n_draws = NULL,
  seed = NULL,
  type = c("all", "trajectory_draws", "distributions", "probabilities", "labels")
)
```

Arguments

`object`	A fitted CRAFT object returned by `craft_fit()`.

newdata	Optional matrix or data frame containing the most recent series history. If supplied, it must contain the fitted asset columns and more rows than the fitted trajectory window. If NULL, the latest trajectory stored in object is reused.
n_draws	Optional number of sampled future-trajectory draws. Defaults to the number of draws stored in object.
seed	Optional random seed for sampling.
type	Prediction output type. "all" returns the full prediction object; "trajectory_draws" returns sampled future-trajectory rows; "distributions" returns return and level forecast distributions; "probabilities" returns transition probabilities; "labels" returns latest regime labels.

Value

For type = "all", an object of class craft_prediction, a list with latest_regime_labels, latest_regime_probabilities, trajectory_draws, sampler_trace, corr_by_horizon, return_dists, level_dists, and diagnostics. Other type values return the requested subset.

Examples

```
set.seed(1)
series <- data.frame(
  a = cumprod(1 + rnorm(80, 0.001, 0.01)),
  b = cumprod(1 + rnorm(80, 0.0005, 0.012))
)
fit <- craft_fit(
  series,
  window = 5,
  n_draws = 50,
  n_factors = 1,
  min_segment = 5,
  max_regimes_per_factor = 3,
  n_testing = 0,
  verbose = FALSE
)
pred <- craft_predict(fit, n_draws = 25, seed = 1)
names(pred$return_dists)
```

---

denoiser *Select Informative Numeric Features*

Description

Ranks numeric features using completeness, variation, temporal structure, PCA contribution, signal-to-noise, and optional supervised association, then returns a reduced feature set.

Usage

```
denoiser(
  x,
  target = NULL,
  min_non_missing = 0.7,
  min_unique = 3L,
  min_mad = 1e-08,
  max_cor = 0.95,
  keep_quantile = 0.7,
  min_score = NULL,
  min_features = 1L,
  max_features = NULL,
  top_n = NULL,
  supervised_weight = if (!is.null(target)) 0.6 else 0,
  pca_weight = 0.45,
  temporal_weight = 0.25,
  snr_weight = 0.15,
  info_weight = 0.15,
  lag_max = 5L,
  impute = c("median", "zero"),
  return_report = FALSE,
  verbose = TRUE
)
```

Arguments

x	A matrix or data frame of candidate features.
target	Optional target vector, or the name of a column in x, used for supervised scoring.
min_non_missing	Minimum non-missing fraction required before a feature can be selected.
min_unique	Minimum number of unique finite values required.
min_mad	Minimum median absolute deviation required.
max_cor	Maximum absolute pairwise correlation allowed among selected features.
keep_quantile	Score quantile used as an adaptive selection threshold.
min_score	Optional absolute minimum score threshold.
min_features	Minimum number of features to retain when possible.
max_features	Optional maximum number of retained features.
top_n	Optional number of top-scoring features to retain.
supervised_weight	Weight assigned to supervised target association.
pca_weight	Weight assigned to PCA loading contribution.
temporal_weight	Weight assigned to temporal autocorrelation structure.
snr_weight	Weight assigned to signal-to-noise proxy scoring.

info_weight	Weight assigned to information and completeness scoring.
lag_max	Maximum lag used for temporal-structure scoring.
impute	Imputation method used during scoring.
return_report	Logical. If TRUE, return selected data, feature names, report, and settings.
verbose	Logical. If TRUE, emit a summary message.

Value

By default, a matrix or data frame with selected features and attributes denoiser_report, selected_features, and removed_features. If return_report = TRUE, a list with data, features, report, and settings.

Examples

```
set.seed(1)
x <- data.frame(signal = cumsum(rnorm(30)), noise = rnorm(30), const = 1)
out <- denoiser(x, min_features = 1, verbose = FALSE, return_report = TRUE)
out$features
```

---

regime_sampler *Sample Candidate Rows from Regime Probabilities*

Description

Combines per-target regime probability vectors into row-level sampling probabilities and samples candidate future-trajectory rows accordingly.

Usage

```
regime_sampler(
  trafo_df,
  cluster_df,
  prob_vectors,
  weights = NULL,
  size = 1000L,
  return_type = c("stack", "index"),
  seed = NULL,
  eps = 1e-12,
  ...
)
```

Arguments

trafo_df	Data frame of candidate transformed observations to sample.
cluster_df	Data frame of regime labels for each candidate row. Must have the same number of rows as trafo_df.
prob_vectors	Named list of probability vectors or one-row matrices/data frames, one per target regime column. Names should correspond to columns of cluster_df.
weights	Optional numeric weights for target regime columns.
size	Number of rows or indices to sample with replacement.
return_type	Return sampled rows as "stack" or sampled row indices as "index".
seed	Optional random seed.
eps	Small positive probability floor.
...	Additional arguments. The legacy argument name return is accepted as an alias for return_type.

Value

If return_type = "stack", a data frame sampled from trafo_df. If return_type = "index", an integer vector of sampled row indices.

Examples

```
trafo <- data.frame(x = 1:4)
clusters <- data.frame(r = c("low", "low", "high", "high"))
probs <- list(r = c(low = 0.2, high = 0.8))
regime_sampler(trafo, clusters, probs, size = 5, seed = 1)
```

---

smooth_quantile_distribution

Fit a Smooth Empirical Distribution from Quantiles

Description

Fits a smooth empirical quantile function and derives an interpolated density, cumulative distribution, quantile function, and random sampler.

Usage

```
smooth_quantile_distribution(
  x,
  probs = seq(0.001, 0.999, length.out = 1000),
  spar = NULL,
  density_spar = NULL,
  eps = 1e-06,
  enforce_monotone = TRUE,
  plot_result = FALSE
)
```

Arguments

x	Numeric vector of observations. Non-finite values are removed.
probs	Probability grid strictly inside (0, 1).
spar	Optional smoothing parameter passed to stats::smooth.spline() for the quantile function.
density_spar	Optional smoothing parameter passed to stats::smooth.spline() for the implied density.
eps	Small positive lower bound used in derivative and probability calculations.
enforce_monotone	Logical. If TRUE, enforce monotonicity of the smoothed quantile curve.
plot_result	Logical. If TRUE, plot the fitted quantile curve and implied density.

Value

A list with data, dfun, pfun, qfun, rfun, q_fit, d_fit, and original_x.

Examples

```
set.seed(1)
fit <- smooth_quantile_distribution(rnorm(100), probs = seq(0.05, 0.95, length.out = 50))
fit$qfun(c(0.25, 0.5, 0.75))
fit$dfun(0)
```

---

svd_changepoint_regimes

Detect Regimes from SVD Scores and Changepoints

Description

Projects multivariate time-series columns into SVD scores, detects changepoints in selected score dimensions, merges or filters regimes, and returns regime labels, profiles, diagnostics, and helper functions for new data transformation and plotting.

Usage

```
svd_changepoint_regimes(
  X,
  series_cols = NULL,
  denoise = TRUE,
  denoiser_args = NULL,
  reduction_criteria = c(
    "variance", "elbow", "kaiser", "bic", "fixed",
    "regime_strength", "hybrid"
  ),
  n_dim = NULL,
```

```
  var_threshold = 0.95,
  max_dim = NULL,
  center = TRUE,
  scale = TRUE,
  na_action = c("impute", "stop"),
  cpt_method = "PELT",
  cpt_penalty = "MBIC",
  minseglen = 5,
  filter_weak_dims = FALSE,
  min_regime_strength = 0.1,
  max_selected_dims = NULL,
  max_regimes_per_dim = 6,
  min_regime_size = 30,
  merge_similar_regimes = TRUE,
  merge_threshold = 0.35,
  stability_bootstrap = FALSE,
  n_bootstrap = 100,
  bootstrap_noise = 0.05,
  cpt_tolerance = NULL,
  min_cpt_stability = 0.6,
  newdata_temperature = 1,
  regime_prefix = "regime",
  return_scores = TRUE,
  return_probabilities = TRUE,
  plot = FALSE,
  plot_type = c("scores", "heatmap", "summary"),
  seed = NULL
)
```

Arguments

X	A data frame or matrix containing numeric time-series columns.
series_cols	Optional names of columns to use. If NULL, all numeric columns are used.
denoise	Logical. If TRUE, select features with denoiser() before fitting.
denoiser_args	Optional list of arguments passed to denoiser().
reduction_criteria	Criterion used to select SVD dimensions.
n_dim	Number of dimensions when reduction_criteria = "fixed".
var_threshold	Cumulative explained variance threshold for variance-based selection.
max_dim	Optional maximum SVD dimension considered.
center	Logical. If TRUE, center selected columns.
scale	Logical. If TRUE, scale selected columns.
na_action	How to handle missing values: impute column means or stop.
cpt_method	Changepoint method passed to changepoint::cpt.meanvar().
cpt_penalty	Penalty passed to changepoint::cpt.meanvar().

`minseglen` Minimum segment length for changepoint fitting.

`filter_weak_dims`
: Logical. If `TRUE`, filter dimensions with weak regime strength.

`min_regime_strength`
: Minimum regime-strength threshold.

`max_selected_dims`
: Optional maximum number of selected dimensions.

`max_regimes_per_dim`
: Maximum regimes retained per SVD dimension after merging.

`min_regime_size`
: Minimum desired regime segment size.

`merge_similar_regimes`
: Logical. If `TRUE`, merge short or similar adjacent regimes.

`merge_threshold`
: Distance threshold used when merging similar regimes.

`stability_bootstrap`
: Logical. If `TRUE`, assess changepoint stability by bootstrapping.

`n_bootstrap` Number of bootstrap replicates.

`bootstrap_noise`
: Noise scale used for bootstrap perturbations.

`cpt_tolerance` Maximum distance for matching bootstrapped changepoints.

`min_cpt_stability`
: Minimum bootstrap stability for retaining changepoints.

`newdata_temperature`
: Temperature used when transforming new observations to regime probabilities.

`regime_prefix` Prefix used for generated regime-label columns.

`return_scores` Logical. If `TRUE`, include selected SVD scores.

`return_probabilities`
: Logical. If `TRUE`, include regime probabilities.

`plot` Logical. If `TRUE`, produce a plot.

`plot_type` Plot type for `plot = TRUE`.

`seed` Optional random seed.

Value

An object of class `svd_changepoint_regimes_v2`, represented as a list with regime labels, selected dimensions, diagnostics, profiles, SVD metadata, changepoint results, and helper functions.

Examples

```
set.seed(1)
x <- data.frame(a = c(rnorm(40, 0), rnorm(40, 2)), b = c(rnorm(40, 0), rnorm(40, -1)))
fit <- svd_changepoint_regimes(
  x,
  denoise = FALSE,
```

```
  reduction_criteria = "fixed",
  n_dim = 1,
  minseglen = 5,
  min_regime_size = 5,
  max_regimes_per_dim = 3,
  seed = 1
)
head(fit$regimes)
```

`time_volume_weights` *Compute Time Weights from Trading Volumes*

Description

Aggregates numeric volume columns row-wise and converts the aggregate volume profile into non-negative time weights.

Usage

```
time_volume_weights(
  volume_df,
  date_col = NULL,
  alpha = 1,
  normalize = TRUE,
  eps = 1e-08,
  winsorize = FALSE,
  winsor_q = 0.99
)
```

Arguments

`volume_df`	A data frame containing one or more numeric volume columns.
`date_col`	Optional name of a date column to exclude from the numeric volume aggregation and use as output names.
`alpha`	Non-negative exponent applied to aggregate volume scores. Values below 1 compress high-volume observations; values above 1 emphasize them.
`normalize`	Logical. If `TRUE`, return weights summing to one.
`eps`	Small positive offset added before exponentiation.
`winsorize`	Logical. If `TRUE`, cap aggregate volumes at `winsor_q`.
`winsor_q`	Quantile used as the cap when `winsorize = TRUE`.

Value

A numeric vector of weights, named by `date_col` when supplied.

Examples

```
vol <- data.frame(day = as.Date("2024-01-01") + 0:2, a = c(10, 20, 30), b = c(5, 5, 10))
time_volume_weights(vol, date_col = "day")
```

`trajectory_embedding` *Build Past and Future Trajectory Matrices*

Description

Builds past and future cumulative-trajectory matrices for all columns in a multivariate time-series data set. Rows containing unavailable lag or lead values are removed from the aligned training matrices, while `latest_trajectory` retains the latest past trajectory row for forecasting.

Usage

```
trajectory_embedding(ts_set, trajectory_window)
```

Arguments

`ts_set`	A data frame or matrix of numeric time series in columns.
`trajectory_window`	Positive integer number of lag and lead horizons.

Value

A list with `past_trajectories`, `future_trajectories`, `latest_trajectory`, and `time_index`. The `time_index` vector indicates rows retained in the aligned past and future trajectory matrices.

Examples

```
prices <- data.frame(a = c(100, 101, 103, 104, 106), b = c(50, 52, 51, 53, 54))
trajectories <- trajectory_embedding(prices, trajectory_window = 2)
dim(trajectories$past_trajectories)
dim(trajectories$future_trajectories)
```

`trajectory_forecast` *Fit Smooth Forecast Distributions for Trajectory Draws*

Description

Fits one smoothed empirical forecast distribution per column in a data frame of sampled future-trajectory draws. Optionally reconstructs returns or differences into forecast levels.

Usage

```
trajectory_forecast(
  trajectory_draws,
  last_level = NULL,
  recon_mode = NULL,
  probs = seq(0.001, 0.999, length.out = 1000),
  spar = NULL,
  density_spar = NULL,
  eps = 1e-06,
  enforce_monotone = TRUE,
  plot_result = FALSE,
  min_obs = 10L,
  min_unique = 10L,
  jitter_if_needed = TRUE,
  jitter_scale = 1e-08,
  zero_density_outside = TRUE,
  seed = 1L,
  verbose = FALSE,
  return_diagnostics = TRUE
)
```

Arguments

`trajectory_draws`	Data frame of sampled future-trajectory draws, with one variable per column.
`last_level`	Optional scalar or vector of latest observed levels used to reconstruct forecast levels.
`recon_mode`	Optional reconstruction mode: `"differences"`, `"scaled_growth"`, `"log_growth"`, or `NULL`.
`probs`	Probability grid passed to `smooth_quantile_distribution()`.
`spar`	Optional quantile smoothing parameter.
`density_spar`	Optional density smoothing parameter.
`eps`	Small positive numerical tolerance.
`enforce_monotone`	Logical. If `TRUE`, enforce monotone quantile fits.
`plot_result`	Logical. If `TRUE`, plot fitted distributions.
`min_obs`	Minimum observations used for each fitted distribution. Shorter samples are bootstrapped.
`min_unique`	Minimum unique values before jittering is considered.
`jitter_if_needed`	Logical. If `TRUE`, add small jitter to discrete or constant samples.
`jitter_scale`	Scale of jitter relative to the sample scale.
`zero_density_outside`	Logical. If `TRUE`, return zero density outside the fitted support.

seed	Optional random seed used for augmentation and jitter.
verbose	Logical. If TRUE, emit fitting progress messages.
return_diagnostics	Logical. If TRUE, attach diagnostics as an attribute.

Value

A named list of distribution objects. Diagnostics and settings are attached as attributes when requested.

Examples

```
set.seed(1)
trajectory_draws <- data.frame(a = rnorm(40, 0.01, 0.02), b = rnorm(40, -0.01, 0.03))
fc <- trajectory_forecast(
  trajectory_draws,
  probs = seq(0.05, 0.95, length.out = 30),
  verbose = FALSE
)
fc$a$qfun(0.5)
```

---

trajectory_reframer *Build Lag or Lead Trajectory Features for One Series*

Description

Creates cumulative scaled lag or lead trajectory features for a single time series. For a backward trajectory, horizon h at time t is x[t] / x[t - h] - 1. For a forward trajectory, horizon h is x[t + h] / x[t] - 1.

Usage

```
trajectory_reframer(ts, k, mode = c("backward", "forward"), ts_name = NULL)
```

Arguments

ts	A numeric vector or time-series-like object.
k	Positive integer number of lag or lead horizons to create.
mode	Direction of the trajectory. "backward" creates cumulative lag changes; "forward" creates cumulative lead changes.
ts_name	Optional name used as the prefix for output columns.

Value

An object of class trajectory_reframer, represented as a list with trajectory, ts, k, and mode.

Examples

```
x <- c(100, 102, 105, 103, 108)
traj <- trajectory_reframer(x, k = 2, mode = "backward", ts_name = "price")
head(traj$trajectory)
```

Index

∗ **package**
CRAFT-package, 2

CRAFT *(*CRAFT-package*)*, 2
CRAFT-package, 2
craft_fit, 2
craft_predict, 5

denoiser, 6

regime_sampler, 8

smooth_quantile_distribution, 9
svd_changepoint_regimes, 10

time_volume_weights, 13
trajectory_embedding, 14
trajectory_forecast, 14
trajectory_reframer, 16