# Package 'rollcast'

September 2, 2026

**Type** Package

**Title** Probabilistic Forecasting with Adaptive Mixtures of Rolling Statistics

**Version** 0.1.0

**Description** Implements a probabilistic time-series forecasting framework based on adaptive mixtures of rolling statistical anchors. Rolling means, medians, minimum and maximum values, regression endpoints, and user-specified quantiles define candidate forecast locations. A proper-score gating model assigns state-dependent mixture weights, optional state-conditional residual sampling adds local dispersion, and recursive simulation produces marginal and joint predictive distributions. Numeric hyperparameters can be supplied as scalars or candidate vectors for causal validation-based selection.

**License** MIT + file LICENSE

**Encoding** UTF-8

**Depends** R (>= 4.1.0)

**Imports** Rcpp (>= 1.0.12), stats, graphics, grDevices, utils

**LinkingTo** Rcpp

**Suggests** testthat (>= 3.0.0), knitr, rmarkdown

**VignetteBuilder** knitr

**Config/testthat/edition** 3

**NeedsCompilation** yes

**RoxygenNote** 7.3.3

**Author** Giancarlo Vercellino [aut, cre]

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

**Repository** CRAN

**Date/Publication** 2026-09-02 12:30:02 UTC

# Contents



---

`rollcast-package` *rollcast: Probabilistic Forecasting with Adaptive Mixtures of Rolling Statistics*

---

## Description

The **rollcast** package implements probabilistic forecasting using adaptive mixtures of rolling statistical anchors, causal proper-score gating, optional state-conditional residual sampling, and recursive predictive mixtures.

## Details

Use `rollcast` to fit a model and `predict()` to generate probabilistic forecasts.

## Author(s)

Giancarlo Vercellino

---

`plot.rollcast_prediction` *Plot Rollcast Probabilistic Forecasts and Diagnostics*

---

## Description

Plots either a predictive fan or a compact diagnostic dashboard containing forecast uncertainty and state-dynamics information.

## Usage

```
## S3 method for class 'rollcast_prediction'
plot(
  x,
  type = c("diagnostic", "fan"),
  ...
)

plot_rollcast_fan(
```

```
  prediction,
  levels = c(0.50, 0.80, 0.90, 0.95),
  history = TRUE,
  n_history = 150L,
  smooth = FALSE,
  spar = 0.60,
  enforce_nested = TRUE,
  col = NULL,
  history_col = "black",
  median_col = "black",
  history_lwd = 1.5,
  median_lwd = 2,
  median_lty = 1,
  boundary_lty = 3,
  main = "Rollcast probabilistic forecast",
  xlab = "Time",
  ylab = "Level",
  legend = TRUE,
  legend_outside = TRUE,
  legend_cex = 0.62,
  legend_inset = 0.20,
  legend_bty = "n",
  ...
)

plot_rollcast_diagnostic(
  prediction,
  levels = c(0.50, 0.80, 0.90, 0.95),
  history = TRUE,
  n_history = 180L,
  smooth = FALSE,
  spar = 0.60,
  enforce_nested = TRUE,
  dispersion_level = 0.90,
  transition_smooth = TRUE,
  transition_spar = 0.55,
  transition_min_segment = 3L,
  fan_palette = NULL,
  history_col = "#252525",
  median_col = "#111111",
  main = "Rollcast forecast diagnostics",
  ...
)
```

## Arguments

| | |
|---|---|
| x, prediction | A "rollcast_prediction" object. |
| type | For the S3 method, either "diagnostic" or "fan". |

| | |
|---|---|
| `levels` | Central predictive interval levels. |
| `history` | Logical; include recent observed history. |
| `n_history` | Number of historical observations to display. |
| `smooth` | Logical; optionally smooth displayed forecast quantiles. |
| `spar` | Smoothing parameter for displayed forecast curves. |
| `enforce_nested` | Logical; enforce nested displayed intervals. |
| `col` | Optional fan colors for `plot_rollcast_fan()`. |
| `history_col` | History line color. |
| `median_col` | Forecast median color. |
| `history_lwd` | History line width. |
| `median_lwd` | Forecast median line width. |
| `median_lty` | Forecast median line type. |
| `boundary_lty` | Predictive boundary line type. |
| `main` | Plot title. |
| `xlab` | X-axis label for the fan plot. |
| `ylab` | Y-axis label for the fan plot. |
| `legend` | Logical; draw the fan-plot legend. |
| `legend_outside` | Logical; place the fan-plot legend outside the data region when possible. |
| `legend_cex` | Fan-plot legend text size. |
| `legend_inset` | Fan-plot legend inset. |
| `legend_bty` | Fan-plot legend box type. |
| `dispersion_level` | Central interval used to measure uncertainty width. |
| `transition_smooth` | Logical; lightly smooth uncertainty width before transition estimation. |
| `transition_spar` | Smoothing parameter for transition detection. |
| `transition_min_segment` | Minimum horizons on each side of the segmented regression breakpoint. |
| `fan_palette` | Optional fan colors for the diagnostic dashboard. |
| `...` | Additional graphical arguments where supported. |

## Details

The uncertainty transition horizon fits the continuous segmented model

$$W_h = a + b_1 h + \Delta b (h - h^*)_+$$

to predictive interval width $W_h$. The breakpoint $h^*$ estimates a change in the rate of forecast-uncertainty expansion.

## Value

The plotting functions return their diagnostic quantities invisibly.

`predict.rollcast` *Generate Probabilistic Forecasts from a Rollcast Model*

## Description

Recursively simulates an explicit all-anchor predictive mixture and returns marginal and joint probabilistic forecasts.

## Usage

```
## S3 method for class 'rollcast'
predict(
  object,
  horizon = 20L,
  nsim = 3000L,
  probs = c(
    0.01, 0.025, 0.05, 0.10, 0.25, 0.50,
    0.75, 0.90, 0.95, 0.975, 0.99
  ),
  resampling = c("anchor_stratified", "systematic", "multinomial"),
  keep_components = FALSE,
  seed = NULL,
  ...
)
```

## Arguments

| | |
|---|---|
| `object` | A fitted `"rollcast"` object. |
| `horizon` | Positive integer forecast horizon. |
| `nsim` | Number of recursive particles. |
| `probs` | Probabilities included in the forecast summary. |
| `resampling` | Recursive resampling method. |
| `keep_components` | Logical; retain per-particle and per-anchor component matrices. |
| `seed` | Optional random-number seed. |
| `...` | Reserved for future use. |

## Value

An object of class `"rollcast_prediction"` containing recursive paths, anchor probabilities, explicit marginal mixture values and weights, forecast summaries, and predictive functions `dfun`, `pfun`, `qfun`, and `rfun`.

## Examples

```
set.seed(2)
y <- 100 + cumsum(rnorm(150, sd = 0.5))

fit <- rollcast(
  y,
  window = 30, tau = 0.25, lambda = 0.01,
  conditional_k = 10, state_bw = 1,
  residual_bw = 0.35, error_scale = 0,
  residual_smoothing = 0.03,
  rho_min = 0.05, rho_max = 0.90, rho_decay = 1,
  min_history = 10, maxit = 60
)

pred <- predict(fit, horizon = 5, nsim = 100, seed = 2)
pred$qfun(c(0.1, 0.5, 0.9), h = 1)
```

---

# print.rollcast *Print Rollcast Model and Prediction Summaries*

---

## Description

Compact print methods for fitted Rollcast models and probabilistic forecasts.

## Usage

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

## Arguments

| | |
|---|---|
| x | A fitted model or forecast object. |
| ... | Ignored. |

## Value

The input object, invisibly.

`rollcast` *Fit a Rollcast Probabilistic Forecasting Model*

## Description

Fits a probabilistic time-series forecasting model based on adaptive mixtures of rolling statistical anchors. Numeric hyperparameters use a unified interface: a scalar fixes the parameter, while a vector requests causal validation-based search over those candidate values.

## Usage

```
rollcast(
  ts,
  window = c(20L, 30L, 45L, 60L, 90L, 120L),
  tau = c(0.15, 0.25, 0.40),
  lambda = c(0.001, 0.01, 0.05),
  conditional_k = c(20L, 40L, 80L),
  state_bw = c(0.60, 1.00, 1.60),
  residual_bw = c(0.20, 0.35, 0.55),
  error_scale = c(0, 0.10, 0.25, 0.50, 0.75, 1),
  residual_smoothing = c(0.01, 0.03, 0.06),
  rho_min = c(0, 0.05, 0.10),
  rho_max = c(0.80, 0.90, 0.97),
  rho_decay = c(0.50, 1, 2),
  quantiles = c(0.05, 0.10, 0.25, 0.75, 0.90, 0.95),
  feature_type = c("relative", "raw"),
  min_history = 20L,
  score_floor_bw = 0.05,
  validation_fraction = 0.25,
  search_passes = 1L,
  search_maxit = 120L,
  maxit = 500L,
  tolerance = 1e-8,
  verbose = FALSE,
  na_action = c("fail", "omit")
)
```

## Arguments

| | |
|---|---|
| `ts` | Numeric univariate time series. |
| `window` | Rolling-window length or candidate vector. |
| `tau` | Positive temperature for soft anchor responsibilities. |
| `lambda` | Non-negative ridge penalty for the proper-score gating model. |
| `conditional_k` | Number of historical nearest states used by the conditional residual model. |
| `state_bw` | Positive state-similarity bandwidth. |

| | |
|---|---|
| `residual_bw` | Positive residual-kernel bandwidth. |
| `error_scale` | Residual-strength parameter in $[0, 1]$. Zero gives a pure anchor mixture; one applies the full learned residual correction. |
| `residual_smoothing` | Anchor-responsibility smoothing in $[0, 1]$. |
| `rho_min` | Lower bound for adaptive gating persistence. |
| `rho_max` | Upper bound for adaptive gating persistence. |
| `rho_decay` | Positive decay scale for adaptive persistence. |
| `quantiles` | Quantile probabilities used as rolling anchors. This vector defines the anchor dictionary and is not treated as a search grid. |
| `feature_type` | Either `"relative"` or `"raw"`. |
| `min_history` | Minimum causal history before conditional densities are evaluated. |
| `score_floor_bw` | Validation-only normalized density floor used when `error_scale = 0`. |
| `validation_fraction` | Fraction of common causal forecast origins used for validation. |
| `search_passes` | Number of cached coordinate-search passes. |
| `search_maxit` | Optimization budget for provisional gating fits during hyperparameter search. |
| `maxit` | Optimization budget for the final gating fit. |
| `tolerance` | Numerical convergence tolerance. |
| `verbose` | Logical; report search improvements. |
| `na_action` | How non-finite observations are handled. |

## Details

For rolling window $\mathcal{H}_t$, the model constructs an anchor vector containing the rolling mean, median, minimum, maximum, fitted regression endpoint, and user-specified quantiles. With robust scale $s_t$, the default state feature for anchor $j$ is

$$X_{t,j} = (S_{t,j} - y_t)/s_t.$$

Historical soft anchor responsibilities are based on

$$d_{t,j} = |y_{t+1} - S_{t,j}|/s_t$$

and a temperature-controlled softmax.

Forecast candidates are

$$Y_{h,b,j} = S_{h,b,j} + \gamma s_{h,b} E_{h,b,j},$$

where $\gamma$ is `error_scale`. Every particle and anchor contributes to the explicit predictive mixture before recursive resampling.

If a tunable numeric argument is supplied as a scalar, it is fixed. If it is supplied as a vector, candidate values are compared using cached coordinate search on common causal validation origins.

## Value

An object of class `"rollcast"` containing fitted gating parameters, selected hyperparameters, rolling states, residual information, current anchor probabilities, validation diagnostics, and search metadata.

## Examples

```
set.seed(1)
y <- 100 + cumsum(rnorm(160, sd = 0.6))

fit <- rollcast(
  y,
  window = 30,
  tau = 0.25,
  lambda = 0.01,
  conditional_k = 10,
  state_bw = 1,
  residual_bw = 0.35,
  error_scale = 0,
  residual_smoothing = 0.03,
  rho_min = 0.05,
  rho_max = 0.90,
  rho_decay = 1,
  min_history = 10,
  maxit = 60
)

fit
```

# Index