Package: flowz (via r-universe)

October 18, 2026

Type Package

Title Local Latent Flow Forecasting with Empirical Calibration

Version 0.1.0

Description Fits conditional flow-matching models to univariate time series. Combines a selectable exponentially weighted moving average (EWMA), autoregressive integrated moving average (ARIMA), autoregressive fractionally integrated moving average (ARFIMA), or structural time-series baseline with direct multi-step residual paths and a random-feature extreme learning machine (ELM) velocity field. Spectral dispersion calibration uses a chronological calibration block, with a separate test block for forecast scoring. Provides simulated forecast paths and smooth marginal distributions. Methodological background is given by Lipman et al. (2023) <doi:10.48550/arXiv.2210.02747>, Huang et al. (2006) <doi:10.1016/j.neucom.2005.12.126>, Hyndman and Khandakar (2008) <doi:10.18637/jss.v027.i03>, and Haslett and Raftery (1989) <doi:10.2307/2347679>.

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

License GPL-3

Encoding UTF-8

Depends R (>= 4.1.0)

Imports forecast, fracdiff, graphics, grDevices, parallel, stats, utils

Suggests testthat (>= 3.0.0), pkgload

Config/testthat/edition 3

RoxygenNote 7.3.3

NeedsCompilation no

Author Giancarlo Vercellino [aut, cre]

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

Repository https://cran.r-universe.dev

Date/Publication 2026-10-18 16:20:21 UTC

RemoteUrl https://github.com/cran/flowz

RemoteRef HEAD

RemoteSha 2cdbc4a06b68fc94c9350c149611e6db0cc0635a

Contents

flowz-package 2
flowz 2
flowz_forecast 6
predict.flowz 8

Index **10**

`flowz-package` *Conditional Flow Forecasts with Spectral Calibration*

Description

Fits univariate time series using a selectable EWMA, ARIMA, ARFIMA, or structural baseline and a conditional random-feature velocity field. Produces joint multi-step paths, smooth marginal distributions, spectral dispersion calibration, and rolling-origin diagnostics.

Details

Start with `flowz` to fit a model, `predict.flowz` to forecast an updated history, and `plot.flowz` to inspect the four-panel diagnostic display. Read `vignette("flowz-introduction", package = "flowz")` for an executable workflow, parameter guidance, and evaluation limitations.

Author(s)

Giancarlo Vercellino

`flowz` *Fit a Conditional Flow Model for a Univariate Time Series*

Description

Fits a joint distribution of future levels using a selectable EWMA, ARIMA, ARFIMA, or structural baseline, an extreme learning machine (ELM) velocity field, and spectral dispersion calibration. Returns simulated paths, marginal distribution functions, and rolling-origin evaluation diagnostics.

Usage

```
flowz(x, horizon = 10L, transform = "none", baseline = "ewma",
  baseline_control = list(), autotune = TRUE,
  drift_lambda = 0.90, vol_lambda = 0.94, min_train = NULL,
  innovation_lags = 8L,
  n_hidden = 300L, lambda = 1e-3, flow_aug = 5L, n_steps = 30L,
  n_sims = 1000L, cal_fraction = 0.20, test_fraction = 0.20,
  cal_sims = 200L, test_sims = 200L, spectral_modes = NULL,
  spectral_spar = 0.65, spectral_shrink_to_one = 0.10,
  spectral_tau_bounds = c(0.15, 2.50),
  distribution_bw_factor = 0.50,
  seed = 123L, n_origins = NULL, parallel = TRUE, n_cores = NULL,
  plot = FALSE, verbose = FALSE)
```

Arguments

`x` Finite numeric vector or univariate `ts`, ordered from oldest to newest observation. Missing values are not allowed. Time-series attributes are not retained; horizons count observations, not calendar periods.

`horizon` Positive integer number of future observations to simulate.

`transform` Either `"none"` or `"log"`. Log transformation requires strictly positive input. Fitting and evaluation use log values; forecast samples and marginal functions are returned on the original scale.

`baseline` Baseline engine: `"ewma"`, `"arima"`, `"arfima"`, or `"structural"`. It supplies the causal trajectory and horizon-specific uncertainty around which Flow residuals are learned.

`baseline_control` Named list of engine controls. ARIMA accepts `order`, `include.drift`, `max.p`, `max.q`, `max.d`, `stepwise`, and `approximation`. ARFIMA accepts `drange`, `estim`, and a fixed `order = c(p, q)` when autotuning is disabled. Structural models accept `type = "level"` or `"trend"`.

`autotune` Logical. Select the ARIMA, ARFIMA, or structural specification using only the initial training window. The selected specification is then refitted causally at every historical origin. EWMA continues to use `drift_lambda` and `vol_lambda`.

`drift_lambda` EWMA persistence for local first differences, strictly between zero and one. Larger values give slower adaptation.

`vol_lambda` EWMA persistence for squared innovations, strictly between zero and one. Larger values give slower volatility adaptation.

`min_train` First eligible forecast origin. `NULL` uses the largest of 50, five times `horizon`, and six times `innovation_lags + 5L`. Enough subsequent origins must remain for training, calibration, and testing.

`innovation_lags` Positive integer number of lagged standardized innovations used in the conditioning context, alongside drift, volatility, and momentum features.

`n_hidden` Number of random hidden units in the ELM; integer at least five. Increasing this expands model capacity and computational cost.

`lambda` Nonnegative ridge penalty for the ELM output weights. This is distinct from the two EWMA persistence parameters.

`flow_aug` Positive integer number of random noise/time training augmentations per training origin.

`n_steps` Positive integer number of fourth-order Runge–Kutta steps used to integrate the learned velocity field from time zero to one.

`n_sims` Number of current forecast paths; integer at least 20.

`cal_fraction` Positive fraction of selected origins allocated to spectral calibration.

`test_fraction` Positive fraction allocated to final evaluation. The sum of `cal_fraction` and `test_fraction` must be below 0.70. Each block receives at least ten origins; at least twenty training origins must remain.

`cal_sims` Paths simulated per calibration origin; integer at least 20.

`test_sims` Paths simulated per test origin; integer at least 20.

`spectral_modes` Number of low-frequency discrete cosine transform modes to calibrate, from one to `horizon`. `NULL` uses the larger of six and twice the square root of the horizon (rounded up), capped at `horizon`.

`spectral_spar` Smoothing parameter in $[0, 1]$ for the spectral dispersion curve. When smoothing is applicable, the engine uses `stats::smooth.spline`.

`spectral_shrink_to_one`
Shrinkage towards no dispersion correction, in $[0, 1)$. Larger values damp the estimated correction.

`spectral_tau_bounds`
Two positive increasing numbers bounding spectral dispersion multipliers.

`distribution_bw_factor`
Gaussian-kernel bandwidth multiplier in $(0, 2]$. Affects Flow marginal distribution functions, not the stored paths or their empirical summary quantiles.

`seed` Integer random seed. The function sets and advances the session's random-number generator state.

`n_origins` Number of historical origins to select approximately evenly. `NULL` uses a heuristic based on horizon, context size, split fractions, and available history, capped at 500. `Inf` selects all available origins. Small explicit values may leave too few origins for the split.

`parallel` Logical; use parallel calibration and test simulations when more than one worker is available. Defaults to `TRUE`.

`n_cores` Worker count. `NULL` uses at most two detected physical cores, leaving one core free where possible. Explicit values allow more workers; `parallel = FALSE` forces serial execution.

`plot` Logical; draw the four-panel diagnostic plot after fitting.

`verbose` Logical; print progress messages.

Details

The selected baseline supplies a causal mean trajectory and forecast scale. Training targets are joint, standardized multi-step level errors. Random tanh features and ridge-regression output weights approximate a conditional velocity field. Gaussian starting points are integrated to produce joint residual paths. Calibration adjusts dispersion in a discrete cosine basis around the Flow sample mean, preserving that mean on the fitting scale.

Origins are split chronologically, but forecast windows can overlap within and across split boundaries: the implementation does not enforce a horizon embargo. Test scores should therefore not be interpreted as fully independent holdout estimates. Calibration is empirical and offers no exact nominal coverage guarantee. Baseline specification selection uses only the initial training window. If an alternative model fails at an isolated origin, the failure is recorded and Drift/EWMA is used at that origin.

With `transform = "log"`, forecast means, standard deviations, medians, and quantiles are computed from back-transformed samples. Model parameters, kernel bandwidths, and evaluation diagnostics remain on the fitting (log) scale. Density functions include the change-of-variable Jacobian.

Parallel execution uses PSOCK workers on Windows and forked processes elsewhere. Small workloads may run faster with `parallel = FALSE`.

Value

An object of class `flowz`, containing:

`forecast`	A `flowz_forecast` object with three model components: `calibrated`, `baseline`, and `uncalibrated`. Each contains `summary`, `samples`, and `pred_fun`.
`evaluation`	Chronological test results. `metrics` contains `aggregate` and `per_horizon` tables; `pit` contains PIT values for all three models; `metadata` records the scale, split, and observed test values. Forecast windows may overlap.
`calibration`	Spectral calibration settings and estimates, including the applied multipliers, selected modes, variances, and bias diagnostics.
`model`	Fitted weights, scaling constants, calibration, and settings.
`metadata`	The public call, transformation, training size, horizon, output and model scales, and fitting diagnostics such as baseline state, origin selection, parallel settings, and timings.

Practical checks

Use `n_origins = NULL` initially. An explicit origin count below 40 cannot leave twenty training origins and two blocks of ten. Check the selected origin count and spacing in `metadata$diagnostics$origin_selection`. For a change in units, compare diagnostics only between fits made on the same scale. In particular, log-scale RMSE is not a price-unit error. See the vignette for examples of chronological evaluation and joint events.

See Also

`predict.flowz`, `plot.flowz`. For a worked tutorial, run `vignette("flowz-introduction")`.

Examples

```
t <- seq_len(260)
x <- cumsum(sin(t / 4) + cos(t / 11))
fit <- flowz(x, horizon = 3, n_hidden = 10, flow_aug = 1,
  n_steps = 3, n_sims = 30, cal_sims = 30, test_sims = 30,
  parallel = FALSE)
fit$forecast$calibrated$summary
fit$forecast$calibrated$pred_fun$qfun$h3(c(0.05, 0.50, 0.95))
pr <- predict(fit, x, n_sims = 30, seed = 42)
dim(pr$forecast$calibrated$samples)
```

---

flowz_forecast *Forecast and Prediction Output Structures*

Description

`flowz_forecast` stores the three comparable forecasts in a uniform layout. It is available as `fit$forecast`. `predict()` returns a compact `flowz_prediction` object containing `forecast` and `metadata`.

Usage

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

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

Arguments

`x`	A forecast or prediction object.
`...`	Unused additional arguments.

Details

A forecast has exactly three named components:

`calibrated` The final spectrally calibrated Flow forecast.

`baseline` The selected EWMA, ARIMA, ARFIMA, or structural baseline forecast. Its exact label is recorded in metadata.

`uncalibrated` The Flow forecast before spectral correction.

Every component has the same structure:

`summary` A data frame with one row per horizon and columns `horizon`, `mean`, `median`, `sd`, `q05`, `q25`, `q75`, and `q95`. Values describe the stored samples on the original data scale.

`samples` A numeric matrix with simulated paths in rows and horizons named `h1`, `h2`, etc. in columns. Complete rows retain cross-horizon dependence within that model. Rows from different models are not paired scenarios.

`pred_fun` Lists named `dfun`, `pfun`, `qfun`, and `rfun`; each contains functions named `h1`, `h2`, etc. Their interfaces follow R's density, distribution, quantile, and random generation conventions. For example:

```
fit$forecast$calibrated$pred_fun$qfun$h3(c(.05, .95))
```

Flow functions use smooth kernel-mixture marginals; baseline functions use Gaussian marginals around the selected engine trajectory and scale. All operate on original-scale levels, including for log-transformed fits. Separate marginal random draws do not preserve dependence across horizons.

The `flowz_prediction` returned by `predict()` contains:

`forecast` A `flowz_forecast` object with the layout above.

`metadata` The origin index, horizon, simulation count, transformation, selected baseline and label, forecast scale, and model scale. The origin is the number of observations in the supplied history, not a date.

Summary quantiles are empirical sample quantiles. The smooth marginal functions can return slightly different quantiles. The interface has no duplicated aliases: choose the model first, then its summary, paths, or marginal functions.

Value

The print methods display the calibrated summary and return `x` invisibly. Objects are created by `flowz()` and `predict()`.

See Also

`flowz`, `predict.flowz`

Examples

```
t <- seq_len(260)
x <- cumsum(sin(t / 4) + cos(t / 11))
fit <- flowz(x, horizon = 3, n_hidden = 10, flow_aug = 1,
  n_steps = 3, n_sims = 30, cal_sims = 30, test_sims = 30,
  parallel = FALSE)
fit$forecast$calibrated$summary
fit$forecast$calibrated$pred_fun$qfun$h3(c(.05, .5, .95))
dim(fit$forecast$calibrated$samples)
next_prediction <- predict(fit, x, n_sims = 30, seed = 42)
next_prediction$metadata
next_prediction$forecast$baseline$summary
```

`predict.flowz` *Predict, Plot, and Print a Fitted Flow Model*

Description

Generate paths from an updated history, draw diagnostic plots, or print a summary of a fitted `flowz` object.

Usage

```
## S3 method for class 'flowz'
predict(object, newdata, n_sims = 1000L,
  seed = NULL, plot = FALSE, ...)
## S3 method for class 'flowz'
plot(x, ...)
## S3 method for class 'flowz'
print(x, ...)
```

Arguments

`object`, `x`	An object returned by `flowz`.
`newdata`	Full finite numeric history, in chronological order, ending at the new forecast origin. Must satisfy the fitted minimum-history length. Log-transformed models require strictly positive values.
`n_sims`	Number of new paths. Use at least 20; the method defaults to 1000.
`seed`	Optional integer simulation seed. `NULL` continues from the current random-number generator state. An explicit seed makes repeated predictions reproducible for the same history and settings.
`plot`	Logical; draw the new forecast for an untransformed model. Currently this prediction-time plot is suppressed for log-transformed models; use the returned original-scale samples or summary to draw a new forecast.
`...`	Additional arguments, currently unused.

Details

Prediction updates drift, volatility, and conditioning context using the supplied full history. The fitted velocity field, horizon, and spectral calibration are held fixed; no retraining occurs.

The diagnostic plot uses the history and evaluation results from fitting. It shows the current forecast with a 90 percent band, test RMSE, test coverage, and a probability integral transform (PIT) quantile comparison. Calibrated Flow is a thick solid black line; the selected baseline is dashed gray and identified by its fitted label. Reference lines are dotted. For log-transformed models the forecast panel uses original levels, while RMSE remains on the log scale. PIT values are pooled over test origins and horizons, so the panel may mask horizon-specific deviations. `plot(x)` does not update after `predict(x, newdata)`.

Value

predict returns a flowz_prediction object with two fields: forecast, which has the same three-model structure as object$forecast, and metadata. Each model owns its summary, samples, and pred_fun; see flowz_forecast. Forecast values use the original data scale. plot and print invisibly return the fitted object.

See Also

flowz

Examples

```
t <- seq_len(260)
x <- cumsum(sin(t / 4) + cos(t / 11))
fit <- flowz(x, horizon = 3, n_hidden = 10, flow_aug = 1,
  n_steps = 3, n_sims = 30, cal_sims = 30, test_sims = 30,
  parallel = FALSE)
t_new <- seq_len(265)
x_new <- cumsum(sin(t_new / 4) + cos(t_new / 11))
prediction <- predict(fit, x_new, n_sims = 30, seed = 42)
prediction$forecast$calibrated$summary
prediction$forecast$calibrated$pred_fun$qfun$h3(c(0.05, 0.95))
plot(fit)
```

Index

flowz, 2, *2*, *7–9*
flowz-package, 2
flowz_forecast, *5*, 6, *9*
flowz_prediction *(*flowz_forecast*)*, 6

plot.flowz, *2*, *5*
plot.flowz *(*predict.flowz*)*, 8
predict.flowz, *2*, *5*, *7*, 8
print.flowz *(*predict.flowz*)*, 8
print.flowz_forecast *(*flowz_forecast*)*, 6
print.flowz_prediction
*(*flowz_forecast*)*, 6