

Package: RWInfoLSTM (via r-universe)

October 30, 2026

Type Package

Title Random Walk-Informed LSTM for Time Series Forecasting

Version 0.1.0

Maintainer Ranjit Kumar Paul <ranjitstat@gmail.com>

Description Implements a Random Walk (RW)-informed Long Short-Term Memory (LSTM) network for univariate time series forecasting. The LSTM is trained on the full series with a composite loss that blends the usual mean squared error (MSE) between predictions and observations with the MSE between predictions and the fitted Random Walk, weighted by a parameter lambda (0 gives a standard LSTM, 1 gives a fully RW-guided LSTM). The Random Walk, either simple or with drift, is selected automatically by its in-sample root mean squared error (RMSE), and the lambda value with the lowest training RMSE is retained. The LSTM hyperparameters can be supplied manually or tuned by grid search, and functions for out-of-sample forecasting, accuracy measures and plots are provided. The methodology is motivated by hybrid forecasting framework proposed by Yeasin and Paul (2024) <doi:10.1007/s11227-023-05542-3>.

License MIT + file LICENSE

Encoding UTF-8

Imports torch (>= 0.11.0), coro, ggplot2 (>= 3.4.0), cli (>= 3.6.0), stats, utils

Suggests testthat (>= 3.0.0)

Config/testthat/edition 3

Config/roxygen2/version 8.1.1

NeedsCompilation no

Author Md Yeasin [aut], Ranjit Kumar Paul [aut, cre], Pushkar Bora [aut]

Repository <https://ranjitstat.r-universe.dev>

Date/Publication 2026-10-29 16:16:30 UTC

RemoteUrl <https://github.com/cran/RWInfoLSTM>

RemoteRef HEAD
RemoteSha afc8339f300e3a5d975d9f580a38bf8815a35f83

Contents

AccuracyRWiL	2
plotlambda_rwil	3
plotRWiL	3
predictRWiL	4
RWiL	5
Index	8

AccuracyRWiL	<i>Out-of-sample accuracy for RWiL forecasts vs actual values</i>
--------------	---

Description

Out-of-sample accuracy for RWiL forecasts vs actual values

Usage

AccuracyRWiL(fit, actual, lambda = NULL, verbose = TRUE)

Arguments

- fit A RWiL object returned by RWiL().
- actual Numeric vector of actual values after training.
- lambda Lambda to use. NULL = best.
- verbose Logical. If TRUE (default), prints a formatted accuracy report to the console. Set to FALSE to suppress console output and only receive the returned data frame.

Value

Data frame with RMSE, MAE, SMAPE, MAPE.

Examples

```
# Runs only when the libtorch backend is installed (torch::install_torch())
if (torch::torch_is_installed()) {
  set.seed(1)
  y <- cumsum(rnorm(40, 0.5, 1))
  fit <- RWiL(y, mode = "manual", lag = 3, hidden_size = 4, lr = 0.01,
             epochs = 5, patience = 5, lambda_list = 0.5,
             verbose = FALSE, seed = 1)
  future <- tail(y, 1) + cumsum(rnorm(6, 0.5, 1))
  AccuracyRWiL(fit, actual = future)
}
```

plotlambda_rwil	<i>Plot Training RMSE Across All Lambda Values (RWiL)</i>
-----------------	---

Description

Plot Training RMSE Across All Lambda Values (RWiL)

Usage

```
plotlambda_rwil(fit)
```

Arguments

fit RWiL object.

Value

ggplot2 object (invisibly).

Examples

```
# Runs only when the libtorch backend is installed (torch::install_torch())
if (torch::torch_is_installed()) {
  set.seed(1)
  y <- cumsum(rnorm(40, 0.5, 1))
  fit <- RWiL(y, mode = "manual", lag = 3, hidden_size = 4, lr = 0.01,
             epochs = 5, patience = 5, lambda_list = 0.5,
             verbose = FALSE, seed = 1)
  plotlambda_rwil(fit)
}
```

plotRWiL	<i>Plot RWiL: Actual vs Predicted (in-sample) + Out-of-sample Forecast</i>
----------	--

Description

Plot RWiL: Actual vs Predicted (in-sample) + Out-of-sample Forecast

Usage

```
plotRWiL(
  fit,
  h = 12,
  lambda = NULL,
  actual = NULL,
  newdata = NULL,
  n_hist = NULL
)
```

Arguments

<code>fit</code>	RWiL object from <code>RWiL()</code> .
<code>h</code>	Forecast horizon. Default 12.
<code>lambda</code>	Lambda to use. <code>NULL</code> = best.
<code>actual</code>	Optional numeric vector of actual future values.
<code>newdata</code>	Optional new observations to update starting sequence.
<code>n_hist</code>	Number of historical points to show. <code>NULL</code> = all.

Value

ggplot2 object (invisibly).

Examples

```
# Runs only when the libtorch backend is installed (torch::install_torch())
if (torch::torch_is_installed()) {
  set.seed(1)
  y <- cumsum(rnorm(40, 0.5, 1))
  fit <- RWiL(y, mode = "manual", lag = 3, hidden_size = 4, lr = 0.01,
             epochs = 5, patience = 5, lambda_list = 0.5,
             verbose = FALSE, seed = 1)
  plotRWiL(fit, h = 6)
}
```

predictRWiL

Forecast h steps ahead from a fitted RWiL model

Description

Forecast h steps ahead from a fitted RWiL model

Usage

```
predictRWiL(fit, h = 1, lambda = NULL, newdata = NULL)
```

Arguments

<code>fit</code>	A RWiL object returned by <code>RWiL()</code> .
<code>h</code>	Number of steps to forecast. Default 1.
<code>lambda</code>	Lambda value to use. <code>NULL</code> = best (lowest Train RMSE).
<code>newdata</code>	Optional numeric vector of new observations after training.

Value

Data frame with columns: `h`, `time_idx`, `forecast`.

Examples

```
# Runs only when the libtorch backend is installed (torch::install_torch())
if (torch::torch_is_installed()) {
  set.seed(1)
  y <- cumsum(rnorm(40, 0.5, 1))
  fit <- RWiL(y, mode = "manual", lag = 3, hidden_size = 4, lr = 0.01,
             epochs = 5, patience = 5, lambda_list = 0.5,
             verbose = FALSE, seed = 1)
  predictRWiL(fit, h = 6)
}
```

RWiL

Random Walk-Informed LSTM (RWiL)

Description

Trains an LSTM on the FULL dataset using the Random Walk-informed composite loss:

$$\text{Loss} = (1-\lambda) \times \text{MSE}(\hat{y}, y) + \lambda \times \text{MSE}(\hat{y}, y_{\text{RW}})$$

Trains on ALL data – no train/test split. Use predictRWiL() for out-of-sample forecasting. An internal 20 percent validation split handles early stopping.

The Random Walk model is auto-selected between:

1. Simple RW: $\hat{y}_t = y_{t-1}$
2. Drift RW: $\hat{y}_t = y_{t-1} + d$, where $d = \text{mean}(\text{diff}(y))$

Selection is by lowest in-sample RMSE.

Usage

```
RWiL(
  data,
  mode = c("auto", "manual"),
  lag,
  hidden_size = 32,
  num_layers = 1,
  lr = 0.001,
  dropout = 0.0,
  epochs = 500,
  patience = 30,
  tune_epochs = 50,
  batch_size = 16,
  lambda_list = seq(0.1, 0.9, 0.1),
  hidden_grid = c(8, 16, 24, 32, 40),
  layers_grid = c(1L, 2L),
  lr_grid = c(0.001, 0.003, 0.005),
```

```

dropout_grid = c(0.0, 0.1, 0.2),
verbose = TRUE,
seed = NULL
)

```

Arguments

<code>data</code>	Numeric vector. Full time series – ALL observations used for training.
<code>mode</code>	"auto" runs grid-search LSTM tuning. "manual" uses user-supplied hyperparameters.
<code>lag</code>	Look-back window length. COMPULSORY – no default. Example: lag = 3 means model uses last 3 values to predict next.
<code>hidden_size</code>	LSTM hidden units (manual mode). Default 32.
<code>num_layers</code>	Stacked LSTM layers (manual mode). Default 1.
<code>lr</code>	Learning rate (manual mode). Default 0.001.
<code>dropout</code>	Dropout rate (manual mode). Default 0.0.
<code>epochs</code>	Maximum training epochs. Default 500.
<code>patience</code>	Early-stopping patience. Default 30.
<code>tune_epochs</code>	Epochs per grid-search combo (auto). Default 50.
<code>batch_size</code>	Mini-batch size. Default 16.
<code>lambda_list</code>	Lambda values to test. Default seq(0.1, 0.9, 0.1). lambda = 0.0: Standard LSTM (no RW guidance). lambda = 1.0: Fully RW-guided LSTM.
<code>hidden_grid</code>	Hidden-unit candidates (auto mode).
<code>layers_grid</code>	Layer-count candidates (auto mode).
<code>lr_grid</code>	Learning-rate candidates (auto mode).
<code>dropout_grid</code>	Dropout candidates (auto mode).
<code>verbose</code>	Print progress. Default TRUE.
<code>seed</code>	Optional integer random seed for reproducible results. Default NULL: no seed is set (call <code>set.seed()</code> and <code>torch::torch_manual_seed()</code> beforehand, or pass a value here).

Value

S3 object of class "RWiL" with elements:

results Named list per lambda. Each contains: `tr_rmse`, `tr_mae`, `tr_mape`, `tr_smape`, `tr_preds`, `tr_actuals`, `val_rmse`, `loss_hist`, `time`, `lambda`, `model`.

best_model Complete best lambda summary: `lambda`, `hyperparams`, `rw` (`type`, `drift`, `rmse_simple`, `rmse_drift`), `metrics`, `predictions`, `loss_history`, `model` object, `train_time`.

rw Random Walk results: `rw_type`, `drift`, `rmse_simple`, `rmse_drift`.

best_hp Best LSTM hyperparameters.

tuning_log Grid-search log (auto mode only).

meta Settings: `lag`, `lambda_list`, `mode`, `best_lam_nm`, `selection` = "Train RMSE".

data_info `data`, `N`, `scaler`, `last_seq_scaled`, `lag`.

Examples

```
# Small toy example (a few seconds)
# Runs only when the libtorch backend is installed (torch::install_torch())
if (torch::torch_is_installed()) {
  set.seed(1)
  y <- cumsum(rnorm(40, 0.5, 1))
  fit <- RWiL(y, mode = "manual", lag = 3, hidden_size = 4, lr = 0.01,
             epochs = 5, patience = 5, lambda_list = 0.5,
             verbose = FALSE, seed = 1)
  print(fit)
  summary(fit)
  predictRWiL(fit, h = 6)
  fit$rw$rw_type      # "simple" or "drift"
}

# Auto mode: grid search over the LSTM hyperparameters (takes longer)
if (torch::torch_is_installed()) {
  set.seed(1)
  y <- cumsum(rnorm(60, 0.5, 2))
  fit_auto <- RWiL(y, mode = "auto", lag = 3,
                  hidden_grid = c(4, 8), layers_grid = 1L,
                  lr_grid = 0.01, dropout_grid = 0,
                  tune_epochs = 5, epochs = 20, patience = 5,
                  lambda_list = c(0.3, 0.7), verbose = FALSE, seed = 1)
  summary(fit_auto)
  plotRWiL(fit_auto, h = 12)
  plotlambda_rwil(fit_auto)
}
```

Index

AccuracyRWiL, [2](#)

plotlambda_rwil, [3](#)

plotRWiL, [3](#)

predictRWiL, [4](#)

RWiL, [5](#)