Package 'echos'

October 22, 2026

Type Package

Title Echo State Networks for Time Series Modeling and Forecasting

Version 1.0.5

Description Provides a lightweight implementation of functions and methods for fast and fully automatic time series modeling and forecasting using Echo State Networks (ESNs).

License GPL-3

URL https://github.com/ahaeusser/echos, https://ahaeusser.github.io/echos/

BugReports https://github.com/ahaeusser/echos/issues

Depends R (>= 4.0.0), fabletools (>= 0.3.0)

Imports Rcpp (>= 1.0.3), tsibble, dplyr, tidyr, rlang, distributional

LinkingTo Rcpp, RcppArmadillo

Encoding UTF-8

LazyData true

Suggests knitr, rmarkdown, tidyverse, fable, testthat (>= 3.0.0)

VignetteBuilder knitr

Config/testthat/edition 3

Config/roxygen2/version 8.1.1

NeedsCompilation yes

Author Alexander Häußer [aut, cre, cph] (ORCID: <https://orcid.org/0009-0000-5419-8479>)

Maintainer Alexander Häußer <alexander-haeusser@gmx.de>

Repository CRAN

Date/Publication 2026-10-22 07:10:02 UTC

Contents

ESN . . . 2
filter_esn . . . 3
fitted.ESN . . . 4
forecast.ESN . . . 5
forecast_esn . . . 6
glance.ESN . . . 8
is.esn . . . 9
is.forecast_esn . . . 9
is.tune_esn . . . 10
m4_data . . . 11
m4_monthly_subset . . . 11
model_sum.ESN . . . 12
plot.esn . . . 13
plot.forecast_esn . . . 14
plot.tune_esn . . . 15
print.esn . . . 16
report.ESN . . . 16
reservoir . . . 17
residuals.ESN . . . 18
run_reservoir . . . 19
summary.esn . . . 19
summary.tune_esn . . . 20
synthetic_data . . . 21
tidy.ESN . . . 22
train_esn . . . 23
tune_esn . . . 26

Index **29**

`ESN` *Train an Echo State Network*

Description

Train an Echo State Network (ESN) to a univariate time series. The function automatically manages data pre-processing, reservoir generation (i.e., internal states) and model estimation and selection. The function is a wrapper for `train_esn()` and intended to be used in combination with `fabletools::model()`.

Usage

```
ESN(formula, ...)
```

Arguments

formula	A model formula used by the fabletools model specification interface. The left-hand side identifies the univariate response variable to be modeled. The current implementation does not interpret additional right-hand-side terms, because echos currently supports univariate ESN models without exogenous regressors. The formula interface is retained for consistency with fabletools::model() and to allow future extensions, such as exogenous variables or richer model specifications.
...	Further arguments passed to train_esn(), including ESN architecture, preprocessing, and model-selection parameters such as lags, n_diff, n_states, n_models, alpha, rho, tau, density, lambda, scale_win, scale_wres, and scale_inputs.

Value

An object of class ESN.

See Also

Other tidy functions: filter_esn(), fitted.ESN(), forecast.ESN(), glance.ESN(), model_sum.ESN(), report.ESN(), reservoir(), residuals.ESN(), tidy.ESN()

Examples

```
library(tsibble)
library(fable)
AirPassengers %>%
as_tsibble() %>%
model("ESN" = ESN(value))
```

---

filter_esn *Filter ESN models*

Description

Filter an object of class mdl_df ("mable") to include ESN models only, i.e., other models like ARIMA or ETS are excluded from the mable.

Usage

```
filter_esn(object)
```

Arguments

object	An object of class mdl_df, containing an ESN model.

Value

An object of class `mdl_df` in long-format.

See Also

Other tidy functions: `ESN()`, `fitted.ESN()`, `forecast.ESN()`, `glance.ESN()`, `model_sum.ESN()`, `report.ESN()`, `reservoir()`, `residuals.ESN()`, `tidy.ESN()`

Examples

```
library(tsibble)
library(fable)
AirPassengers %>%
as_tsibble() %>%
model("ESN" = ESN(value)) %>%
filter_esn()
```

---

fitted.ESN *Extract fitted values from a trained ESN*

Description

Extract fitted values from a trained ESN as `tsibble`.

Usage

```
## S3 method for class 'ESN'
fitted(object, ...)
```

Arguments

`object`	An object of class `mdl_df`, containing an ESN model.
`...`	Currently not in use.

Value

Fitted values extracted from the object.

See Also

Other tidy functions: `ESN()`, `filter_esn()`, `forecast.ESN()`, `glance.ESN()`, `model_sum.ESN()`, `report.ESN()`, `reservoir()`, `residuals.ESN()`, `tidy.ESN()`

Examples

```
library(tsibble)
library(fable)
AirPassengers %>%
as_tsibble() %>%
model("ESN" = ESN(value)) %>%
fitted()
```

`forecast.ESN` *Forecast an Echo State Network*

Description

Forecast an Echo State Network (ESN) from a trained model via recursive forecasting. Forecast intervals are generated by simulating future sample path based on a moving block bootstrap of the residuals and estimating the quantiles from the simulations. The function is a wrapper for `forecast_esn()` and intended to be used in combination with `fabletools::model()`.

Usage

```
## S3 method for class 'ESN'
forecast(
  object,
  new_data,
  normal = TRUE,
  n_sim = 200,
  specials = NULL,
  xreg = NULL,
  ...
)
```

Arguments

`object`	An object of class `mdl_df`, containing an ESN model.
`new_data`	Forecast horizon (n-step ahead forecast).
`normal`	Logical value. If `TRUE`, `dist_normal()` is used, otherwise `dist_sample()`.
`n_sim`	Integer value. The number of future sample path generated during simulation.
`specials`	Currently not in use.
`xreg`	A `tsibble` containing exogenous variables.
`...`	Currently not in use.

Value

An object of class `fbl_ts` ("fable").

See Also

Other tidy functions: ESN(), filter_esn(), fitted.ESN(), glance.ESN(), model_sum.ESN(), report.ESN(), reservoir(), residuals.ESN(), tidy.ESN()

Examples

```
library(tsibble)
library(fable)
AirPassengers %>%
as_tsibble() %>%
model("ESN" = ESN(value)) %>%
forecast(h = 18)
```

`forecast_esn` *Forecast an Echo State Network*

Description

Forecast an Echo State Network (ESN) from a trained model via recursive forecasting. Forecast intervals are generated by simulating future sample path based on a moving block bootstrap of the residuals and estimating the quantiles from the simulations.

Usage

```
forecast_esn(
  object,
  n_ahead = 18,
  levels = c(80, 95),
  n_sim = 100,
  n_seed = 42
)
```

Arguments

`object`	An object of class `esn`. The result of a call to `train_esn()`.
`n_ahead`	Integer value. The number of periods for forecasting (i.e. forecast horizon).
`levels`	Numeric vector. The levels of the forecast intervals (in percent), e.g., `c(80, 95)`. Values must lie between 0 and 100.
`n_sim`	Integer value. The number of simulated future paths used to compute forecast intervals via a moving block bootstrap of the (demeaned) in-sample residuals. If `NULL`, no intervals are computed.
`n_seed`	Integer value. The seed for the random number generator (for reproducibility).

Value

A `list` containing:

- `point`: Numeric vector containing the point forecasts.
- `interval`: Numeric matrix containing the forecast intervals.
- `sim`: Numeric matrix containing the simulated future sample path.
- `std`: Numeric vector with standard deviations.
- `levels`: Integer vector. The levels of the forecast intervals.
- `actual`: Numeric vector containing the actual values.
- `fitted`: Numeric vector containing the fitted values.
- `n_ahead`: Integer value. The number of periods for forecasting (forecast horizon).
- `model_spec`: Character value. The model specification as string.

References

- Häußer, A. (2027). Echo State Networks for Time Series Forecasting: Hyperparameter Sweep and Benchmarking. Applied Soft Computing, 204, 116346. doi:10.1016/j.asoc.2026.116346
- Häußer, A. (2026). echos: An R Package for Automatic Time Series Forecasting using Echo State Networks. The R Journal, 18(3), 120–143. doi:10.32614/RJ2026040
- Jaeger, H. (2001). The "echo state" approach to analysing and training recurrent neural networks. GMD Report 148, German National Research Center for Information Technology.
- Jaeger, H. (2002). Tutorial on training recurrent neural networks, covering BPTT, RTRL, EKF and the "echo state network" approach. GMD Report 159, German National Research Center for Information Technology.
- Lukosevicius, M. (2012). A practical guide to applying echo state networks. In Neural Networks: Tricks of the Trade, 2nd edition, pages 659–686. Springer. doi:10.1007/9783642-352898_36
- Lukosevicius, M. and Jaeger, H. (2009). Reservoir computing approaches to recurrent neural network training. Computer Science Review, 3(3):127–149. doi:10.1016/j.cosrev.2009.03.005

See Also

Other base functions: `is.esn()`, `is.forecast_esn()`, `is.tune_esn()`, `plot.esn()`, `plot.forecast_esn()`, `plot.tune_esn()`, `print.esn()`, `summary.esn()`, `summary.tune_esn()`, `train_esn()`, `tune_esn()`

Examples

```
xdata <- as.numeric(AirPassengers)
xmodel <- train_esn(y = xdata)
xfcst <- forecast_esn(xmodel, n_ahead = 12)
plot(xfcst)
```

glance.ESN *Summary of trained models during random search*

Description

Return summary statistics from trained ESN models during random search as `tibble`.

- `model`: Model identifier.
- `loglik`: Log-likelihood.
- `nobs`: Number of observations.
- `df`: Effective degrees of freedom.
- `lambda`: Regularization parameter.
- `aic`: Akaike Information Criterion.
- `aicc`: Corrected Akaike Information Criterion.
- `bic`: Bayesian Information Criterion.
- `hqc`: Hannan-Quinn Information Criterion.
- `mse`: Mean Squared Error.
- `mae`: Mean Absolute Error.

Usage

```
## S3 method for class 'ESN'
glance(x, ...)
```

Arguments

`x`	An object of class `mdl_df`, containing an ESN model.
`...`	Currently not in use.

Value

Summary statistics extracted from the object.

See Also

Other tidy functions: `ESN()`, `filter_esn()`, `fitted.ESN()`, `forecast.ESN()`, `model_sum.ESN()`, `report.ESN()`, `reservoir()`, `residuals.ESN()`, `tidy.ESN()`

Examples

```
library(tsibble)
library(fable)
AirPassengers %>%
as_tsibble() %>%
model("ESN" = ESN(value)) %>%
glance()
```

is.esn *Checks if object is of class "esn"*

Description

Returns TRUE if the object is of class esn.

Usage

```
is.esn(object)
```

Arguments

object	object to be tested.

Value

Logical value. If TRUE, the object is of class esn.

See Also

Other base functions: forecast_esn(), is.forecast_esn(), is.tune_esn(), plot.esn(), plot.forecast_esn(), plot.tune_esn(), print.esn(), summary.esn(), summary.tune_esn(), train_esn(), tune_esn()

Examples

```
xdata <- as.numeric(AirPassengers)
xmodel <- train_esn(y = xdata)
is.esn(xmodel)
```

is.forecast_esn *Checks if object is of class "forecast_esn"*

Description

Returns TRUE if the object is of class forecast_esn.

Usage

```
is.forecast_esn(object)
```

Arguments

object	object to be tested.

Value

Logical value. If TRUE, the object is of class forecast_esn.

See Also

Other base functions: forecast_esn(), is.esn(), is.tune_esn(), plot.esn(), plot.forecast_esn(), plot.tune_esn(), print.esn(), summary.esn(), summary.tune_esn(), train_esn(), tune_esn()

Examples

```
xdata <- as.numeric(AirPassengers)
xmodel <- train_esn(y = xdata)
xfcst <- forecast_esn(xmodel, n_ahead = 12)
is.forecast_esn(xfcst)
```

---

is.tune_esn *Checks if object is of class "tune_esn"*

Description

Returns TRUE if the object is of class tune_esn.

Usage

```
is.tune_esn(object)
```

Arguments

object	object to be tested.

Value

Logical value. If TRUE, the object is of class tune_esn.

See Also

Other base functions: forecast_esn(), is.esn(), is.forecast_esn(), plot.esn(), plot.forecast_esn(), plot.tune_esn(), print.esn(), summary.esn(), summary.tune_esn(), train_esn(), tune_esn()

Examples

```
xdata <- as.numeric(AirPassengers)
fit <- tune_esn(
  y = xdata,
  n_ahead = 12,
  n_split = 5,
  alpha = c(0.5, 1),
  rho   = c(1.0),
```

```
  tau   = c(0.4),
  inf_crit = "bic"
)
is.tune_esn(fit)
```

`m4_data` *Deprecated name for* `m4_monthly_subset`

Description

`m4_data` has been renamed to `m4_monthly_subset`. The old name is retained for backward compatibility with earlier versions of `echos`, but users should prefer `m4_monthly_subset` in new code.

Usage

```
data(m4_data)
```

Format

See `m4_monthly_subset`.

See Also

`m4_monthly_subset`

Examples

```
data(m4_data)
head(m4_data)
```

`m4_monthly_subset` *Monthly subset of the M4 competition data*

Description

A `tsibble` with six monthly time series from the M4 Forecasting Competition. The dataset is intended for examples, documentation, and tests in the `echos` package. It is not the full M4 dataset.

The dataset contains the following time series:

- M21655 (Demographic), 1995 Jan - 2015 Mar
- M21683 (Demographic), 2000 Jan - 2023 Apr
- M2717 (Macro), 1996 Jan - 2016 Nov
- M28597 (Industry), 1996 Jan - 2016 Dec
- M42529 (Finance), 2001 Jan - 2009 Apr
- M4813 (Macro), 1994 Apr - 2006 May

Usage

```
data(m4_monthly_subset)
```

Format

A time series object of class `tsibble` with 1,152 rows and 4 columns:

- `series`: Unique identifier as `character` (key variable).
- `category`: Category, e.g. Demographic or Macro, as `factor`.
- `index`: Monthly date as `yearmonth` (index variable).
- `value`: Observed value as `numeric` (measurement variable).

Details

This dataset contains a small monthly subset of the M4 Forecasting Competition data. It is included to demonstrate the tidy forecasting interface of `echos` with `tsibble`, `fable`, and `fabletools`.

Source

M4 Forecasting Competition

Examples

```
data(m4_monthly_subset)
head(m4_monthly_subset)
```

`model_sum.ESN` *Model specification of a trained ESN model*

Description

Provides a compact overview of the model specification in the format `ESN({n_states, alpha, rho}, {n_models, df})`.

Usage

```
## S3 method for class 'ESN'
model_sum(x)
```

Arguments

`x`	An object of class `mdl_df`, containing an ESN model.

Value

Model summary extracted from the object.

See Also

Other tidy functions: ESN(), filter_esn(), fitted.ESN(), forecast.ESN(), glance.ESN(), report.ESN(), reservoir(), residuals.ESN(), tidy.ESN()

Examples

```
library(tsibble)
library(fable)
AirPassengers %>%
as_tsibble() %>%
model("ESN" = ESN(value))
```

`plot.esn` *Plot internal states of a trained ESN model*

Description

Plot internal states (i.e., the reservoir) of a trained ESN model as line chart.

Usage

```
## S3 method for class 'esn'
plot(x, ...)
```

Arguments

`x`	An object of class `esn`. The result of a call to `train_esn()`.
`...`	Further arguments passed to `matplot()`.

Value

Line chart of internal states.

See Also

Other base functions: forecast_esn(), is.esn(), is.forecast_esn(), is.tune_esn(), plot.forecast_esn(), plot.tune_esn(), print.esn(), summary.esn(), summary.tune_esn(), train_esn(), tune_esn()

Examples

```
xdata <- as.numeric(AirPassengers)
xmodel <- train_esn(y = xdata)
plot(xmodel)
```

`plot.forecast_esn` *Plot forecasts of a trained ESN model*

Description

Plot point forecasts and forecast intervals, actual values of a trained ESN model. Optionally, test data (out-of-sample) and fitted values can be added to the plot.

Usage

```
## S3 method for class 'forecast_esn'
plot(x, test = NULL, fitted = TRUE, interval = TRUE, n_obs = NULL, ...)
```

Arguments

`x`	An object of class `forecast_esn`. The result of a call to `forecast_esn()`.
`test`	Numeric vector. Test data, i.e., out-of-sample actual values.
`fitted`	Logical value. If `TRUE`, fitted values are added.
`interval`	Logical value. If `TRUE`, forecast intervals are added.
`n_obs`	Integer value. If `NULL`, all in-sample values are shown, otherwise only the last `n_obs`.
`...`	Further arguments passed to `plot()`.

Value

Line chart of point forecast and actual values.

See Also

Other base functions: `forecast_esn()`, `is.esn()`, `is.forecast_esn()`, `is.tune_esn()`, `plot.esn()`, `plot.tune_esn()`, `print.esn()`, `summary.esn()`, `summary.tune_esn()`, `train_esn()`, `tune_esn()`

Examples

```
xdata <- as.numeric(AirPassengers)
xmodel <- train_esn(y = xdata)
xfcst <- forecast_esn(xmodel, n_ahead = 12)
plot(xfcst)
```

`plot.tune_esn` *Plot forecasts from a tuned ESN object*

Description

Plot actual values and the point forecasts from the best hyperparameter combination selected via `tune.esn()` using the selected accuracy metric. Forecasts are shown as separate line segments for each test split, with vertical dashed lines marking the starts of test windows.

Usage

```
## S3 method for class 'tune_esn'
plot(x, metric = "mse", ...)
```

Arguments

`x`	An object of class `tune_esn`. The result of a call to `tune_esn()`.
`metric`	Character value. The metric used to select the best hyperparameter combination (`metric = c("mse", "mae")`.
`...`	Further arguments passed to `plot()`.

Value

Line chart of point forecast and actual values.

See Also

Other base functions: `forecast_esn()`, `is.esn()`, `is.forecast_esn()`, `is.tune_esn()`, `plot.esn()`, `plot.forecast_esn()`, `print.esn()`, `summary.esn()`, `summary.tune_esn()`, `train_esn()`, `tune_esn()`

Examples

```
xdata <- as.numeric(AirPassengers)
fit <- tune_esn(
  y = xdata,
  n_ahead = 12,
  n_split = 5,
  alpha = c(0.5, 1),
  rho   = c(1.0),
  tau   = c(0.4),
  inf_crit = "bic"
)

plot(fit)
```

print.esn — *Print model specification of the trained ESN model*

Description

Provides a compact overview of the model specification in the format ESN({n_states, alpha, rho}, {n_models, df}).

Usage

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

Arguments

x	An object of class esn. The result of a call to train_esn().
...	Currently not in use.

Value

Print specification of the trained ESN model.

See Also

Other base functions: forecast_esn(), is.esn(), is.forecast_esn(), is.tune_esn(), plot.esn(), plot.forecast_esn(), plot.tune_esn(), summary.esn(), summary.tune_esn(), train_esn(), tune_esn()

Examples

```
xdata <- as.numeric(AirPassengers)
xmodel <- train_esn(y = xdata)
print(xmodel)
```

report.ESN — *Provide a detailed summary of the trained ESN model*

Description

Provide a detailed summary of the trained ESN model. The function is a wrapper for summary.esn().

Usage

```
## S3 method for class 'ESN'
report(object, ...)
```

Arguments

object	An object of class mdl_df, containing an ESN model.
...	Currently not in use.

Value

Print detailed model summary.

See Also

Other tidy functions: ESN(), filter_esn(), fitted.ESN(), forecast.ESN(), glance.ESN(), model_sum.ESN(), reservoir(), residuals.ESN(), tidy.ESN()

Examples

```
library(tsibble)
library(fable)
AirPassengers %>%
as_tsibble() %>%
model("ESN" = ESN(value)) %>%
report()
```

---

reservoir *Return the reservoir from a trained ESN as tibble*

Description

Return the reservoir (internal states) from a trained ESN as tibble. The function works only for models of class ESN.

Usage

```
reservoir(object)
```

Arguments

object	An object of class mdl_df, containing an ESN model.

Value

A tibble containing the reservoir (internal states).

See Also

Other tidy functions: ESN(), filter_esn(), fitted.ESN(), forecast.ESN(), glance.ESN(), model_sum.ESN(), report.ESN(), residuals.ESN(), tidy.ESN()

Examples

```
library(tsibble)
library(fable)
AirPassengers %>%
as_tsibble() %>%
model("ESN" = ESN(value)) %>%
reservoir()
```

`residuals.ESN` *Extract residuals from a trained ESN*

Description

Extract residuals from a trained ESN as `tsibble`.

Usage

```
## S3 method for class 'ESN'
residuals(object, ...)
```

Arguments

`object`	An object of class `mdl_df`, containing an ESN model.
`...`	Currently not in use.

Value

Residuals extracted from the object.

See Also

Other tidy functions: `ESN()`, `filter_esn()`, `fitted.ESN()`, `forecast.ESN()`, `glance.ESN()`, `model_sum.ESN()`, `report.ESN()`, `reservoir()`, `tidy.ESN()`

Examples

```
library(tsibble)
library(fable)
AirPassengers %>%
as_tsibble() %>%
model("ESN" = ESN(value)) %>%
residuals()
```

run_reservoir *Run reservoir*

Description

Run reservoir creates the internal states for the ESN.

Usage

```
run_reservoir(inputs, win, wres, alpha)
```

Arguments

inputs	Numeric matrix containing the input features.
win	Numeric matrix. The input weight matrix.
wres	Numeric matrix. The reservoir weight matrix.
alpha	Numeric value. The leakage rate (smoothing parameter).

Value

Numeric matrix with the internal states.

summary.esn *Provide a detailed summary of the trained ESN model*

Description

Provide a detailed summary of the trained ESN model.

Usage

```
## S3 method for class 'esn'
summary(object, ...)
```

Arguments

object	An object of class esn. The result of a call to train_esn().
...	Currently not in use.

Value

Print detailed model summary.

See Also

Other base functions: forecast_esn(), is.esn(), is.forecast_esn(), is.tune_esn(), plot.esn(), plot.forecast_esn(), plot.tune_esn(), print.esn(), summary.tune_esn(), train_esn(), tune_esn()

Examples

```
xdata <- as.numeric(AirPassengers)
xmodel <- train_esn(y = xdata)
summary(xmodel)
```

---

summary.tune_esn *Provide a summary of the hyperparameter tuning*

Description

Provide a summary of the tuned hyperparameters alpha, rho and tau.

Usage

```
## S3 method for class 'tune_esn'
summary(object, metric = "mse", ...)
```

Arguments

object	An object of class tune_esn. The result of a call to tune_esn().
metric	Character value. The metric used to select the best hyperparameter combination (metric = c("mse", "mae").
...	Currently not in use.

Value

Print detailed model summary.

See Also

Other base functions: forecast_esn(), is.esn(), is.forecast_esn(), is.tune_esn(), plot.esn(), plot.forecast_esn(), plot.tune_esn(), print.esn(), summary.esn(), train_esn(), tune_esn()

Examples

```
xdata <- as.numeric(AirPassengers)
fit <- tune_esn(
  y = xdata,
  n_ahead = 12,
  n_split = 5,
  alpha = c(0.5, 1),
  rho   = c(1.0),
  tau   = c(0.4),
  inf_crit = "bic"
)

summary(fit)
```

---

synthetic_data *Synthetic time series data*

Description

A `tibble` with ten synthetic univariate time series. The dataset is intended for examples, documentation, and tests in the `echos` package. It contains deterministic wave patterns as well as stochastic benchmark processes.

The dataset contains the following time series:

- Square Wave
- Sawtooth Wave
- Harmonic Wave
- Harmonic Wave w/ Trend
- Amplitude Modulated Wave
- Frequency Modulated Wave
- AR(1) Process
- MA(2) Process
- White Noise Process
- Random Walk Process

Usage

```
data(synthetic_data)
```

Format

An object of class `tibble` with 2,000 rows and 3 columns:

- `variable`: Unique identifier of the synthetic time series as `character`.
- `index`: Time index as `integer`.
- `value`: Observed value as `numeric`.

Details

Each synthetic series contains 200 observations. The deterministic series provide examples of common nonlinear and periodic patterns, while the stochastic series provide simple autoregressive, moving-average, white-noise, and random-walk benchmark processes.

Examples

```
data(synthetic_data)
head(synthetic_data)
```

`tidy.ESN` *Estimated coefficients*

Description

Return the estimated coefficients from a trained ESN as `tibble`.

Usage

```
## S3 method for class 'ESN'
tidy(x, ...)
```

Arguments

`x`	An object of class `mdl_df`, containing an ESN model.
`...`	Currently not in use.

Value

Coefficients extracted from the object.

See Also

Other tidy functions: `ESN()`, `filter_esn()`, `fitted.ESN()`, `forecast.ESN()`, `glance.ESN()`, `model_sum.ESN()`, `report.ESN()`, `reservoir()`, `residuals.ESN()`

Examples

```
library(tsibble)
library(fable)
AirPassengers %>%
as_tsibble() %>%
model("ESN" = ESN(value)) %>%
tidy()
```

`train_esn` *Train an Echo State Network*

Description

Train an Echo State Network (ESN) to a univariate time series. The function automatically manages data pre-processing, reservoir generation (i.e., internal states) and model estimation and selection.

Usage

```
train_esn(
  y,
  lags = 1,
  inf_crit = "bic",
  n_diff = NULL,
  n_states = NULL,
  n_models = NULL,
  n_initial = NULL,
  n_seed = 42,
  alpha = 1,
  rho = 1,
  tau = 0.4,
  density = 0.5,
  lambda = c(1e-04, 2),
  scale_win = 0.5,
  scale_wres = 0.5,
  scale_inputs = c(-0.5, 0.5)
)
```

Arguments

`y`	Numeric vector containing the response variable (no missing values).
`lags`	Integer vector with the lag(s) associated with the input variable.
`inf_crit`	Character value. Information criterion used for model selection among the `n_models` candidate ridge fits (different random `lambda` values). The candidate with the smallest criterion value is selected. One of `c("aic", "aicc", "bic", "hqc")`.
`n_diff`	Integer value. The nth-differences of the response variable. If `n_diff = NULL`, the number of differences required to achieve stationarity is determined automatically via a KPSS-test.
`n_states`	Integer value. The number of internal states of the reservoir. If `n_states = NULL`, the reservoir size is determined by `min(floor(n_total * tau), 200)`, where `n_total` is the time series length.
`n_models`	Integer value. The maximum number of (random) models to train for model selection. If `n_models = NULL`, the number of models is defined as `n_states*2`.

n_initial	Integer value. The number of observations of internal states for initial drop out (throw-off). If n_initial = NULL, the throw-off is defined as n_total*0.05, where n_total is the time series length.
n_seed	Integer value. The seed for the random number generator (for reproducibility).
alpha	Numeric value. The leakage rate (smoothing parameter) applied to the reservoir (value greater than 0 and less than or equal to 1).
rho	Numeric value. The spectral radius for scaling the reservoir weight matrix (value often between 0 and 1, but values above 1 are possible).
tau	Numeric value. The reservoir scaling parameter to determine the reservoir size based on the time series length (value greater than 0 and less than or equal to 1).
density	Numeric value. The connectivity of the reservoir weight matrix (dense or sparse) (value greater than 0 and less than or equal to 1).
lambda	Numeric vector. Lower and upper bound of lambda range for ridge regression (numeric vector of length 2 with both values greater than 0 and lambda[1] < lambda[2]).
scale_win	Numeric value. The lower and upper bound of the uniform distribution for scaling the input weight matrix (value greater than 0, weights are sampled from U(-scale_win, scale_win)).
scale_wres	Numeric value. The lower and upper bound of the uniform distribution for scaling the reservoir weight matrix (value greater than 0, weights are sampled from U(-scale_wres, scale_wres) before applying rho and density).
scale_inputs	Numeric vector. The lower and upper bound for scaling the time series data (numeric vector of length 2 with scale_inputs[1] < scale_inputs[2], often symmetric, e.g., c(-0.5, 0.5) or c(-1, 1)).

Details

The implemented Echo State Network consists of an input layer, a randomly generated recurrent reservoir, and a linear readout layer. The input and reservoir weight matrices are randomly initialized and then kept fixed. Only the readout weights are estimated, using ridge regression.

Before the ESN is trained, the input series is optionally differenced using n_diff; if n_diff = NULL, the number of differences is selected automatically using a KPSS-based procedure. The transformed series is scaled to scale_inputs. Lagged values specified by lags are used as inputs to the reservoir.

If n_states = NULL, the reservoir size is set to min(floor(n_total * tau), 200), where n_total is the length of the input series. If n_models = NULL, n_states * 2 candidate readout models are estimated. For these candidate models, regularization parameters are drawn from the interval specified by lambda, and the best model is selected according to inf_crit.

The most influential ESN hyperparameters are n_states, alpha, rho, density, scale_win, scale_wres, and lambda. Larger reservoirs can represent richer dynamics but increase runtime and overfitting risk. The leakage rate alpha controls the speed of reservoir state updates. The spectral radius rho affects the memory and stability of the reservoir. The density controls the sparsity of recurrent connections. The lambda interval controls the amount of ridge regularization considered during readout selection.

Value

A `list` containing:

- `actual`: Numeric vector containing the actual values.
- `fitted`: Numeric vector containing the fitted values.
- `resid`: Numeric vector containing the residuals.
- `states_train`: Numeric matrix containing the internal states.
- `method`: A `list` containing several objects and meta information of the trained ESN (weight matrices, hyperparameters, model metrics, etc.).

References

- Häußer, A. (2027). Echo State Networks for Time Series Forecasting: Hyperparameter Sweep and Benchmarking. Applied Soft Computing, 204, 116346. doi:10.1016/j.asoc.2026.116346
- Häußer, A. (2026). echos: An R Package for Automatic Time Series Forecasting using Echo State Networks. The R Journal, 18(3), 120–143. doi:10.32614/RJ2026040
- Jaeger, H. (2001). The “echo state” approach to analysing and training recurrent neural networks. GMD Report 148, German National Research Center for Information Technology.
- Jaeger, H. (2002). Tutorial on training recurrent neural networks, covering BPTT, RTRL, EKF and the "echo state network" approach. GMD Report 159, German National Research Center for Information Technology.
- Lukosevicius, M. (2012). A practical guide to applying echo state networks. In Neural Networks: Tricks of the Trade, 2nd edition, pages 659–686. Springer. doi:10.1007/9783642-352898_36
- Lukosevicius, M. and Jaeger, H. (2009). Reservoir computing approaches to recurrent neural network training. Computer Science Review, 3(3):127–149. doi:10.1016/j.cosrev.2009.03.005

See Also

Other base functions: `forecast_esn()`, `is.esn()`, `is.forecast_esn()`, `is.tune_esn()`, `plot.esn()`, `plot.forecast_esn()`, `plot.tune_esn()`, `print.esn()`, `summary.esn()`, `summary.tune_esn()`, `tune_esn()`

Examples

```
xdata <- as.numeric(AirPassengers)
xmodel <- train_esn(y = xdata)
summary(xmodel)
```

tune_esn *Tune hyperparameters of an Echo State Network*

Description

Tune hyperparameters of an Echo State Network (ESN) based on time series cross-validation (i.e., rolling forecast). The input series is split into `n_split` expanding-window train/test sets with test size `n_ahead`. For each split and each hyperparameter combination (`alpha`, `rho`, `tau`) an ESN is trained via `train_esn()` and forecasts are generated via `forecast_esn()`.

Usage

```
tune_esn(
  y,
  n_ahead = 12,
  n_split = 5,
  alpha = seq(0.1, 1, by = 0.1),
  rho = seq(0.1, 1, by = 0.1),
  tau = c(0.1, 0.2, 0.4),
  min_train = NULL,
  ...
)
```

Arguments

`y`	Numeric vector containing the response variable (no missing values).
`n_ahead`	Integer value. The number of periods for forecasting (i.e. forecast horizon).
`n_split`	Integer value. The number of rolling train/test splits.
`alpha`	Numeric vector of candidate leakage rates. Smaller values produce slower reservoir state updates; larger values make the reservoir react more strongly to recent inputs.
`rho`	Numeric vector of candidate spectral radii. This parameter scales the recurrent reservoir weights and affects reservoir memory and stability.
`tau`	Numeric vector of candidate reservoir scaling values. Used to determine the reservoir size when `n_states = NULL` in `train_esn()`.
`min_train`	Integer value. Minimum training sample size for the first split.
`...`	Further arguments passed to `train_esn()` (except `alpha`, `rho`, and `tau`, which are set by the tuning grid).

Details

`tune_esn()` performs grid-based hyperparameter tuning using expanding-window time series cross-validation. The tuning grid is formed from all combinations of `alpha`, `rho`, and `tau`. For each split, the model is trained on observations from the beginning of the series up to the split-specific training endpoint and evaluated on the following `n_ahead` observations.

For every candidate configuration and split, `train_esn()` is called with the corresponding `alpha`, `rho`, and `tau`; all other arguments supplied through `...` are passed to `train_esn()`. Forecasts are generated with `forecast_esn()`, and the mean squared error (`mse`) and mean absolute error (`mae`) are stored. The accompanying `summary()` and `plot()` methods use these stored errors to identify and display the best-performing hyperparameter configuration.

Runtime increases approximately linearly with the number of grid combinations and the number of validation splits. Users should start with coarse grids and increase the grid resolution only where needed.

Value

An object of class `"tune_esn"` (a list) with:

- `pars`: A `tibble` with one row per hyperparameter combination and split. Columns include `alpha`, `rho`, `tau`, `split`, `train_start`, `train_end`, `test_start`, `test_end`, `mse`, `mae`, and `id`.
- `fcst`: A numeric matrix of point forecasts with `nrow(fcst) == nrow(pars)` and `ncol(fcst) == n_ahead`.
- `actual`: The original input series `y` (numeric vector), returned for convenience.

References

- Häußer, A. (2027). Echo State Networks for Time Series Forecasting: Hyperparameter Sweep and Benchmarking. Applied Soft Computing, 204, 116346. doi:10.1016/j.asoc.2026.116346
- Häußer, A. (2026). echos: An R Package for Automatic Time Series Forecasting using Echo State Networks. The R Journal, 18(3), 120–143. doi:10.32614/RJ2026040
- Jaeger, H. (2001). The “echo state” approach to analysing and training recurrent neural networks. GMD Report 148, German National Research Center for Information Technology.
- Jaeger, H. (2002). Tutorial on training recurrent neural networks, covering BPTT, RTRL, EKF and the "echo state network" approach. GMD Report 159, German National Research Center for Information Technology.
- Lukosevicius, M. (2012). A practical guide to applying echo state networks. In Neural Networks: Tricks of the Trade, 2nd edition, pages 659–686. Springer. doi:10.1007/9783642-352898_36
- Lukosevicius, M. and Jaeger, H. (2009). Reservoir computing approaches to recurrent neural network training. Computer Science Review, 3(3):127–149. doi:10.1016/j.cosrev.2009.03.005

See Also

Other base functions: `forecast_esn()`, `is.esn()`, `is.forecast_esn()`, `is.tune_esn()`, `plot.esn()`, `plot.forecast_esn()`, `plot.tune_esn()`, `print.esn()`, `summary.esn()`, `summary.tune_esn()`, `train_esn()`

Examples

```
xdata <- as.numeric(AirPassengers)
fit <- tune_esn(
```

```
  y = xdata,
  n_ahead = 12,
  n_split = 5,
  alpha = c(0.5, 1),
  rho   = c(1.0),
  tau   = c(0.4),
  inf_crit = "bic"
)
summary(fit)
plot(fit)
```

Index

∗ **base functions**
  forecast_esn, 6
  is.esn, 9
  is.forecast_esn, 9
  is.tune_esn, 10
  plot.esn, 13
  plot.forecast_esn, 14
  plot.tune_esn, 15
  print.esn, 16
  summary.esn, 19
  summary.tune_esn, 20
  train_esn, 23
  tune_esn, 26
∗ **datasets**
  m4_data, 11
  m4_monthly_subset, 11
  synthetic_data, 21
∗ **tidy functions**
  ESN, 2
  filter_esn, 3
  fitted.ESN, 4
  forecast.ESN, 5
  glance.ESN, 8
  model_sum.ESN, 12
  report.ESN, 16
  reservoir, 17
  residuals.ESN, 18
  tidy.ESN, 22

ESN, 2
ESN(), *4*, *6*, *8*, *13*, *17, 18*, *22*

filter_esn, 3
filter_esn(), *3, 4*, *6*, *8*, *13*, *17, 18*, *22*
fitted.ESN, 4
fitted.ESN(), *3, 4*, *6*, *8*, *13*, *17, 18*, *22*
forecast.ESN, 5
forecast.ESN(), *3, 4*, *8*, *13*, *17, 18*, *22*
forecast_esn, 6
forecast_esn(), *9, 10*, *13–16*, *20*, *25*, *27*

glance.ESN, 8
glance.ESN(), *3, 4*, *6*, *13*, *17, 18*, *22*

is.esn, 9
is.esn(), *7*, *10*, *13–16*, *20*, *25*, *27*
is.forecast_esn, 9
is.forecast_esn(), *7*, *9, 10*, *13–16*, *20*, *25*, *27*
is.tune_esn, 10
is.tune_esn(), *7*, *9, 10*, *13–16*, *20*, *25*, *27*

m4_data, 11
m4_monthly_subset, *11*, 11
model_sum.ESN, 12
model_sum.ESN(), *3, 4*, *6*, *8*, *17, 18*, *22*

plot.esn, 13
plot.esn(), *7*, *9, 10*, *14–16*, *20*, *25*, *27*
plot.forecast_esn, 14
plot.forecast_esn(), *7*, *9, 10*, *13*, *15, 16*, *20*, *25*, *27*
plot.tune_esn, 15
plot.tune_esn(), *7*, *9, 10*, *13, 14*, *16*, *20*, *25*, *27*
print.esn, 16
print.esn(), *7*, *9, 10*, *13–15*, *20*, *25*, *27*

report.ESN, 16
report.ESN(), *3, 4*, *6*, *8*, *13*, *17, 18*, *22*
reservoir, 17
reservoir(), *3, 4*, *6*, *8*, *13*, *17, 18*, *22*
residuals.ESN, 18
residuals.ESN(), *3, 4*, *6*, *8*, *13*, *17*, *22*
run_reservoir, 19

summary.esn, 19
summary.esn(), *7*, *9, 10*, *13–16*, *20*, *25*, *27*
summary.tune_esn, 20
summary.tune_esn(), *7*, *9, 10*, *13–16*, *20*, *25*, *27*
synthetic_data, 21

tidy.ESN, 22
tidy.ESN(), *3*, *4*, *6*, *8*, *13*, *17*, *18*
train_esn, 23
train_esn(), *7*, *9*, *10*, *13–16*, *20*, *27*
tune_esn, 26
tune_esn(), *7*, *9*, *10*, *13–16*, *20*, *25*