

## CRAN Task View: Time Series Analysis

**Maintainer:** Rob J Hyndman, Rebecca Killick

**Contact:** Rob.Hyndman at monash.edu

**Version:** 2024-12-22

**URL:** <https://CRAN.R-project.org/view=TimeSeries>

**Source:** <https://github.com/cran-task-views/TimeSeries/>

**Contributions:** Suggestions and improvements for this task view are very welcome and can be made through issues or pull requests on GitHub or via e-mail to the maintainer address. For further details see the [Contributing guide](#).

**Citation:** Rob J Hyndman, Rebecca Killick (2024). CRAN Task View: Time Series Analysis. Version 2024-12-22. URL <https://CRAN.R-project.org/view=TimeSeries>.

**Installation:** The packages from this task view can be installed automatically using the [ctv](#) package. For example, `ctv::install.views("TimeSeries", coreOnly = TRUE)` installs all the core packages or `ctv::update.views("TimeSeries")` installs all packages that are not yet installed and up-to-date. See the [CRAN Task View Initiative](#) for more details.

Base R ships with a lot of functionality useful for time series, in particular in the stats package. This is complemented by many packages on CRAN, which are briefly summarized below. There is overlap between the tools for time series and those designed for specific domains including [Econometrics](#), [Finance](#) and [Environmetrics](#).

The packages in this view can be roughly structured into the following topics. If you think that some package is missing from the list, please let us know, either via e-mail to the maintainer or by submitting an issue or pull request in the GitHub repository linked above.

### Basics

- *Infrastructure* : Base R contains substantial infrastructure for representing and analysing time series data. The fundamental class is "ts" that can represent regularly spaced time series (using numeric time stamps). Hence, it is particularly well-suited for annual, monthly, quarterly data, etc.
- *Rolling statistics* : Moving averages are computed by `ma` from [forecast](#), and `rollmean` from [zoo](#). The latter also provides a general function `rollapply`, along with other specific rolling statistics functions. [slider](#) calculates a diverse and comprehensive set of type-stable running functions for any R data types. [tsibble](#) provides `slide_tsibble()` for rolling statistics, `tile_tsibble()` for non-overlapping sliding windows, and `stretch_tsibble()` for expanding windows. [tbrf](#) provides rolling functions based on date and time windows instead of n-lagged observations. [roll](#) provides parallel functions for computing rolling statistics. [runner](#) provides tools for running any R function in rolling windows or date windows. [runstats](#) provides fast computational methods for some running sample statistics. For [data.table](#), `froll()` can be used for high-performance rolling statistics.
- *Graphics* : Time series plots are obtained with `plot()` applied to `ts` objects. (Partial) autocorrelation functions plots are implemented in `acf()` and `pacf()`. Alternative versions are provided by `Acf()` and `Pacf()` in [forecast](#), along with a combination display using `tsdisplay()`. Seasonal displays are obtained using `monthplot()` in stats, `seasonplot` in [forecast](#), and `seasplot` in [tsutils](#). [feasts](#) provides various time series graphics for `tsibble` objects including time plots, season plots, subseries plots, ACF and PACF plots, and some combination displays. Interactive graphics for `tsibbles` using `htmlwidgets` are provided by [tsibbletalk](#). [dCovTS](#) computes and plots the distance covariance and correlation functions of time series. [ggseas](#) provides additional `ggplot2` graphics for seasonally adjusted series and rolling statistics. Calendar plots are implemented in [sugrants](#). [gravitas](#) allows for visualizing probability distributions conditional on bivariate temporal granularities. [dygraphs](#) provides an interface to the Dygraphs interactive time series charting library. [TSstudio](#) provides some interactive visualization tools for time series. [ZRA](#) plots forecast

objects from the [forecast](#) package using dygraphs. Basic fan plots of forecast distributions are provided by [forecast](#) and [vars](#). More flexible fan plots of any sequential distributions are implemented in [fanplot](#).

## Times and Dates

- Class "ts" can only deal with numeric time stamps, but many more classes are available for storing time/date information and computing with it. For an overview see *R Help Desk: Date and Time Classes in R* by Gabor Grothendieck and Thomas Petzoldt in [R News 4\(1\)](#), 29-32.
- Classes "yearmon" and "yearqtr" from [zoo](#) allow for more convenient computation with monthly and quarterly observations, respectively.
- Class "Date" from the base package is the basic class for dealing with dates in daily data. The dates are internally stored as the number of days since 1970-01-01.
- The [chron](#) package provides classes for `dates()`, `hours()` and date/time (intraday) in `chron()`. There is no support for time zones and daylight savings time. Internally, "chron" objects are (fractional) days since 1970-01-01.
- Classes "POSIXct" and "POSIXlt" implement the POSIX standard for date/time (intraday) information and also support time zones and daylight savings time. However, the time zone computations require some care and might be system-dependent. Internally, "POSIXct" objects are the number of seconds since 1970-01-01 00:00:00 GMT. Package [lubridate](#) provides functions that facilitate certain POSIX-based computations, while [clock](#) provides a comprehensive library for date-time manipulations using a new family of orthogonal date-time classes (durations, time points, zoned-times, and calendars). The [anytime](#) package converts various inputs into `POSIXct` or `Date` objects. Various recurrent calendar calculations are possibly using [almanac](#). [timechange](#) allows for efficient manipulation of date-times accounting for time zones and daylight saving times. [wktmo](#) converts weekly data to monthly data in several ways.
- Class "timeDate" is provided in the [timeDate](#) package (previously: fCalendar). It is aimed at financial time/date information and deals with time zones and daylight savings times via a new concept of "financial centers". Internally, it stores all information in "POSIXct" and does all computations in GMT only. Calendar functionality, e.g., including information about weekends and holidays for various stock exchanges, is also included. [qlcal](#) allows access to various financial exchange calendars via QuantLib.
- [partime](#) provides date time classes that allow for uncertainty and partially missing information.
- Datetimes with optional UTC offsets and/or heterogeneous time zones are provided by [datetimeoffset](#).
- To convert between the Gregorian and the Vedic calendars, use [VedicDateTime](#), while [jalcal](#) provides conversions between the Gregorian and Persian Jalali (or Solar Hijri) calendars. For year-based time series, [era](#) provides for many year numbering systems used in contemporary and historic calendars (e.g. Common Era, Islamic 'Hijri' years), as well as year-based time scales used in archaeology, astronomy, geology, and other palaeosciences. [aion](#) contains a toolkit for handling archaeological time series.
- The [tis](#) package provides the "ti" class for time/date information.
- The "mondate" class from the [mondate](#) package facilitates computing with dates in terms of months.
- The [CFtime](#) package encapsulates the CF Metadata Conventions "time" dimension, including all defined calendars. It facilitates the processing of climate change projection data.
- The [tempdisagg](#) package includes methods for temporal disaggregation and interpolation of a low frequency time series to a higher frequency series. Time series disaggregation is also provided by [TSdisaggregation](#), [tsdisagg2](#) and [disaggR](#).

## Time Series Classes

- As mentioned above, "ts" is the basic class for regularly spaced time series using numeric time stamps.
- The [zoo](#) package provides infrastructure for regularly and irregularly spaced time series using arbitrary classes for the time stamps (i.e., allowing all classes from the previous section). It is designed to be as consistent as possible with "ts".
- The package [xts](#) is based on [zoo](#) and provides uniform handling of R's different time-based data classes.
- Several packages aim to handle time-based tibbles: [tsibble](#) provides tidy temporal data frames and associated tools; [tsbox](#) contains tools for working with and coercing between many time series classes

including tsibble, ts, xts, zoo and more. [timetk](#) is another toolkit for converting between various time series data classes.

- Some manipulation tools for time series are available in [data.table](#) including `shift()` for lead/lag operations. Further basic time series functionalities are offered by [DTSg](#) which is based on [data.table](#). [dts](#) provides high-frequency time series support via [nanotime](#) and [data.table](#).
- [collapse](#) provides fast computation of several time series functions such as lead/lag operations, (quasi-, log-) differences and growth rates on time-series and panel data, and ACF/PACF/CCF estimation for panel data.
- Various packages implement irregular time series based on "POSIXct" time stamps, intended especially for financial applications. These include "irts" from [tseries](#).
- The class "timeSeries" in [timeSeries](#) (previously: fSeries) implements time series with "timeDate" time stamps.
- The class "tis" in [tis](#) implements time series with "ti" time stamps.
- The package [tframe](#) contains infrastructure for setting time frames in different formats.
- [timeseriesdb](#) manages time series for official statistics by mapping ts objects to PostgreSQL relations.

## Forecasting and Univariate Modeling

- The [fable](#) package provides tools for fitting univariate time series models to many series simultaneously including ETS, ARIMA, TSLM and other models. It also provides many functions for computing and analysing forecasts. The time series must be in the tsibble format. [fabletools](#) provides tools for extending the [fable](#) framework.
- The [forecast](#) package provides similar tools for ts objects, while [modeltime](#) provides time series forecasting tools for use with the 'tidymodels' ecosystem. Forecast resampling tools for use with [modeltime](#) are provided by [modeltime.resample](#).
- *Exponential smoothing* : `HoltWinters()` in stats provides some basic models with partial optimization, `ETS()` from [fable](#) and `ets()` from [forecast](#) provide a larger set of models and facilities with full optimization. [smooth](#) implements some generalizations of exponential smoothing. [legion](#) implements multivariate versions of exponential smoothing. The [MAPA](#) package combines exponential smoothing models at different levels of temporal aggregation to improve forecast accuracy. Some Bayesian extensions of exponential smoothing are contained in [Rlgt](#).
- [prophet](#) forecasts time series based on an additive model where nonlinear trends are fit with yearly and weekly seasonality, plus holidays. It works best with daily data. [fable.prophet](#) allows prophet models to be used in the [fable](#) framework.
- The theta method is implemented in the `THETA()` function from [fable](#), `thetaf()` function from [forecast](#), and `theta()` from [tsutils](#). An alternative and extended implementation is provided in [forecTheta](#).
- *Autoregressive models* : `ar()` in stats (with model selection).
- *ARIMA models* : `arima()` in stats is the basic function for ARIMA, SARIMA, RegARIMA, and subset ARIMA models. It is enhanced in the [fable](#) package via the `ARIMA()` function which allows for automatic modelling. Similar functionality is provided in the [forecast](#) package via the `auto.arima()` function. `arma()` in the [tseries](#) package provides different algorithms for ARMA and subset ARMA models. [arima2](#) provides a random-restart estimation algorithm to replace `stats:::arima()`. Other estimation methods including the innovations algorithm are provided by [itsmr](#). Package [gsarima](#) contains functionality for Generalized SARIMA time series simulation. [bayesforecast](#) fits Bayesian time series models including seasonal ARIMA and ARIMAX models. [BayesARIMAX](#) implements Bayesian estimation of ARIMAX models. Robust ARIMA modeling is provided in the [robustarima](#) package. The [mar1s](#) package handles multiplicative AR(1) with seasonal processes. [TSTutorial](#) provides an interactive tutorial for Box-Jenkins modelling. Improved prediction intervals for ARIMA and structural time series models are provided by [tsPI](#). ARIMA models with multiple seasonal periods can be handled with [tfarima](#) and [smooth](#).
- *Periodic ARMA models* : [partsm](#) provides for periodic autoregressive time series models, while [perARMA](#) and [pcts](#) implement periodic ARMA modelling and other procedures for periodic time series analysis.
  - *Long memory models* : Some facilities for fractional differenced ARFIMA models are provided in the [fracdiff](#) package. The [arfima](#) package has more advanced and general facilities for ARFIMA and ARIMA models, including dynamic regression (transfer function) models. Additional methods for

fitting and simulating non-stationary ARFIMA models are in [nsarfima](#). [LongMemoryTS](#) provides a collection of functions for analysing long memory time series. Fractionally differenced Gegenbaur ARMA processes are handled by [garma](#). [esemifar](#) provides tools for nonparametric smoothing of long-memory time series.

- *Transfer function* models are provided by the [arfima](#) function in the [arfima](#) and the [tfarima](#) packages.
- *Structural (or unobserved component) models* are implemented in [StructTS\(\)](#) in stats, while automatic modelling and forecasting are provided by [UComp](#) and [autostsm](#). [statespacer](#) implements univariate state space models including structural and SARIMA models. Bayesian structural time series models are implemented in [bsts](#) Robust Kalman filtering is provided by [RobKF](#).
- *Non-Gaussian time series* can be handled with GLARMA state space models via [glarma](#), and using Generalized Autoregressive Score models in the [GAS](#) and [gasmmodel](#) packages. [GlarmaVarSel](#) provides variable selection in high-dimensional sparse GLARMA models. Dynamic Generalized Linear Models are provided by [kDGLM](#), while Dynamic Generalized Additive Models are implemented in [mvgam](#). Conditional Efficient Bayesian inference for nonlinear and non-Gaussian state space models is provided in [bssm](#). [PTSR](#) includes functions to model and forecast a range of regression based dynamic models for positive time series.
- *Count time series* models are handled in the [tscount](#) and [acp](#) packages. [fableCount](#) provides a tidy interface to the INGARCH model from [tscount](#) and the GLARMA model from [glarma](#). [coconots](#) provides tools for convolution-closed time series models for low counts. [tsintermittent](#) implements various models for analysing and forecasting intermittent demand time series. [ZIM](#) provides for Zero-Inflated models for count time series. Zero-inflated INAR models can be handled with the [ZINARp](#) package. Semiparametric estimation and bootstrapping of INAR models is provided by the [spINAR](#) package.
- *GARCH models* : [garch\(\)](#) from [tseries](#) fits basic GARCH models. Many variations on GARCH models are provided by [rugarch](#) and [tsgarch](#). Other univariate GARCH packages include [fGarch](#) which implements ARIMA models with a wide class of GARCH innovations. [bayesforecast](#) fits Bayesian time series models including several variations of GARCH models. There are many more GARCH packages described in the [Finance](#) task view.
- *Stochastic volatility* models are handled by [stochvol](#) in a Bayesian framework.
- *Censored time series* can be modelled using [ARCensReg](#), which fits univariate censored regression models with autoregressive errors.
- *Diffusion models* such as Bass and Gompertz curves are provided by [diffusion](#) and [DIMORA](#). Dynamic Gompertz models for time series growth curves are implemented in [tsgc](#).
- *Portmanteau tests* are provided via [Box.test\(\)](#) in the stats package. Additional tests are given by [portes](#), [WeightedPortTest](#), and [testcorr](#) .
- Outlier detection following the Chen-Liu approach is provided by [tsoutliers](#).
- The [tsoutliers](#) and [tsclean](#) functions in the [forecast](#) package provide some simple heuristic methods for identifying and correcting outliers. [tsrobprep](#) provides methods for replacing missing values and outliers using a model-based approach. [ctbi](#) implements a procedure to clean, decompose and aggregate time series.
- *Change point detection* is provided in [strucchange](#) and [strucchangeRcpp](#) (using linear regression) and in [trend](#) (using nonparametric tests). The [changepoint](#) package provides many popular changepoint methods, and [ecp](#) does nonparametric changepoint detection for univariate and multivariate series. [changepoint.np](#) implements the nonparametric PELT algorithm, while [changepoint.geo](#) implements the high-dimensional changepoint detection method GeomCP. [mosum](#) provides a moving sum procedure for detecting multiple changepoints in univariate time series. [InspectChangepoint](#) uses sparse projection to estimate changepoints in high-dimensional time series. The nonparametric moving sum procedure for detecting multiple changepoints in multivariate time series is provided by [CptNonPar](#). [VARDetect](#) implements multiple change point detection in structural VAR models. [Rbeast](#) provides Bayesian change-point detection and time series decomposition. [breakfast](#) includes methods for fast multiple change-point detection and estimation. [fastcpd](#) provides flexible and fast change point detection for regression type data, time series (ARIMA, VAR and GARCH) and any other data with a custom cost function using Sequential Gradient Descent with PELT. [binsegRcpp](#) provides an efficient C++ implementation of the popular binary segmentation heuristic (univariate data, Gaussian/Poisson/L1/Laplace losses, computes sequence of models from 1 segment to a given max number of segments). [jointseg](#) provides [Fpsn\(\)](#) which implements

a “Functional pruning segment neighborhood” dynamic programming algorithm (univariate data, square loss, computes best model for a certain number of changes/segments). [fpop](#) provides `Fpop()` which implements a “Functional pruning optimal partitioning” dynamic programming algorithm (univariate data, square loss, computes best model for a certain penalty for each change), as well as `multiBinSeg()` which is an efficient implementation of the popular binary segmentation heuristic (multi-variate data, Gaussian loss, computes sequence of models from 1 segment to a given max number of segments). A tidy framework for several changepoint detection algorithms is implemented in [tidychange](#).

- Tests for possibly non-monotonic trends are provided by [funtimes](#).
- *Time series imputation* is provided by the [imputeTS](#) package. Some more limited facilities are available using `na.interp()` from the [forecast](#) package. [imputeTestbench](#) provides tools for testing and comparing imputation methods. [mtsdi](#) implements an EM algorithm for imputing missing values in multivariate normal time series, accounting for spatial and temporal correlations. Imputation methods for multivariate locally stationary time series are in [mvLSWimpute](#).
- The [seer](#) package implements a framework for feature-based forecast model selection.
- A standardized time series forecasting framework including many models is provided by [finnts](#), designed for financial time series.
- Forecasts can be combined in the [fable](#) package using simple linear expressions. [ForecastComb](#) supports many forecast combination methods including simple, geometric and regression-based combinations. [forecastHybrid](#) provides functions for ensemble forecasts, combining approaches from the [forecast](#) package. [opera](#) has facilities for online predictions based on combinations of forecasts provided by the user. [profoc](#) combines probabilistic forecasts using CRPS learning.
- Point forecast evaluation is provided in the `accuracy()` function from the [fable](#) and [forecast](#) packages. Distributional forecast evaluation using scoring rules is available in [fable](#), [scoringRules](#) and [scoringutils](#). The Diebold-Mariano test for comparing the forecast accuracy of two models is implemented in the `dm.test()` function in [forecast](#). [ForeComp](#) generates a size-power tradeoff plot for a given Diebold-Mariano test. A multivariate version of the Diebold-Mariano test is provided by [multDM](#). [tsutils](#) implements the Nemenyi test for comparing forecasts. [greybox](#) provides `ro()` for general rolling origin evaluation of forecasts. [tstests](#) implements several tests for time series goodness of fit and forecast evaluation.
- Tidy tools for forecasting are provided by [sweep](#), converting objects produced in [forecast](#) to “tidy” data frames.
- Multi-step-ahead direct forecasting with several machine learning approaches are provided in [forecastML](#).
- [onlineforecast](#) provides a framework for fitting adaptive forecasting models, allowing forecasts to be used as inputs to models, and models to be updated as new data arrives.
- Data leakage is a problem that can occur in forecasting competitions, and the [tsdataleaks](#) package provides tools for detecting data leakage in such settings.
- *Miscellaneous* : [ltsa](#) contains methods for linear time series analysis, [timsac](#) for time series analysis and control.

## Frequency analysis

- *Spectral density estimation* is provided by `spectrum()` in the stats package, including the periodogram, smoothed periodogram and AR estimates. Bayesian spectral inference is provided by [bspec](#), [beyondWhittle](#) and [regspec](#). [quantspec](#) includes methods to compute and plot Laplace periodograms for univariate time series. The Lomb-Scargle periodogram for unevenly sampled time series is computed by [lomb](#). [peacots](#) provides inference for periodograms using an Ornstein-Uhlenbeck state space model. [spectral](#) uses Fourier and Hilbert transforms for spectral filtering. [psd](#) produces adaptive, sine-multitaper spectral density estimates. [kza](#) provides Kolmogorov-Zurbenko Adaptive Filters including break detection, spectral analysis, wavelets and KZ Fourier Transforms. [multitaper](#) also provides some multitaper spectral analysis tools. Higher-order spectral analysis is implemented in [rhosa](#), including bispectrum, bicoherence, cross-bispectrum and cross-bicoherence.
- *Wavelet methods* : The [wavelets](#) package includes computing wavelet filters, wavelet transforms and multiresolution analyses. Multiresolution forecasting using wavelets is also implemented in [mrf](#). [WaveletComp](#) provides some tools for wavelet-based analysis of univariate and bivariate time series

including cross-wavelets, phase-difference and significance tests. [biwavelet](#) is a port of the WTC Matlab package for univariate and bivariate wavelet analyses. [mvLSW](#) provides tools for multivariate locally stationary wavelet processes. Local PACF estimation for locally stationary wavelet processes is provided by [lpacf](#). Locally stationary wavelet processes can be forecast using [forecastLSW](#). [LSWPlib](#) contains functions for simulation and spectral estimation of locally stationary wavelet packet processes. Tests of white noise using wavelets are provided by [hwwntest](#). Wavelet scalogram tools are contained in [wavScalogram](#). Further wavelet methods can be found in the packages [waveslim](#) and [wavethresh](#). Complex-valued wavelet spectral procedures are provided in [CNLTtsa](#).

- *Harmonic regression* using Fourier terms is implemented in [fable](#) and [forecast](#) packages via the `fourier` function.

## Decomposition and Filtering

- *Filters and smoothing* : `filter()` in stats provides autoregressive and moving average linear filtering of multiple univariate time series. The [robfilter](#) package provides several robust time series filters. `smooth()` from the stats package computes Tukey's running median smoothers, 3RS3R, 3RSS, 3R, etc. [sleekts](#) computes the 4253H twice smoothing method. [mFilter](#) implements several filters for smoothing and extracting trend and cyclical components including Hodrick-Prescott and Butterworth filters. Several filters are provided by [signal](#) including a Butterworth filter and a Savitsky-Golay filter. [hpfilter](#) implements one- and two-sided Hodrick-Prescott filters, while [corbouli](#) implements Corbae-Ouliaris frequency domain filtering. [smoots](#) provides nonparametric estimation of the time trend and its derivatives.
- *Decomposition* : Seasonal decomposition is discussed below. Autoregressive-based decomposition is provided by [ArDec](#). [tsdecomp](#) implements ARIMA-based decomposition of quarterly and monthly data.
- *Singular Spectrum Analysis* is implemented in [Rssa](#) and [ASSA](#).
- *Empirical Mode Decomposition* (EMD) and Hilbert spectral analysis is provided by [EMD](#). Additional tools, including ensemble EMD, are available in [hht](#). An alternative implementation of ensemble EMD and its complete variant are available in [Rlibeemd](#).

## Seasonality

- *Seasonal decomposition* : the stats package provides classical decomposition in `decompose()`, and STL decomposition in `stl()`. Enhanced STL decomposition is available in [stlplus](#). [stR](#) provides Seasonal-Trend decomposition based on Regression. [smooth](#) and [tsutils](#) implement extended versions of classical decomposition.
- X-13-ARIMA-SEATS binaries are provided in the [x13binary](#) package, with [seasonal](#) providing an R interface and [seasonalview](#) providing a GUI. An alternative interface is provided by [x12](#).
- An interface to the JDemetra+ seasonal adjustment software is provided by [RJDemetra](#). [ggdemetra](#) provides associated ggplot2 functions.
- [deseats](#) includes a locally weighted regression approach, and the Berlin method.
- Seasonal adjustment of daily time series, allowing for day-of-week, time-of-month, time-of-year and holiday effects is provided by [dsa](#). Seasonal adjustment of weekly data is provided by [boiwsa](#).
- [StructuralDecompose](#) decomposes a time series into trend, seasonality and residuals, allowing for level shifts.
- *Analysis of seasonality* : the [bfast](#) package provides methods for detecting and characterizing abrupt changes within the trend and seasonal components obtained from a decomposition.
- [season](#): Seasonal analysis of health data including regression models, time-stratified case-crossover, plotting functions and residual checks.
- [seas](#): Seasonal analysis and graphics, especially for climatology.
- [sazedR](#): Method to estimate the period of a seasonal time series.

## Stationarity, Unit Roots, and Cointegration

- *Stationarity and unit roots* : [tseries](#) provides various stationarity and unit root tests including Augmented Dickey-Fuller, Phillips-Perron, and KPSS. Alternative implementations of the ADF and KPSS tests are in

the [urca](#) package, which also includes further methods such as Elliott-Rothenberg-Stock, Schmidt-Phillips and Zivot-Andrews tests. [uroot](#) provides seasonal unit root tests. [CADFtest](#) provides implementations of both the standard ADF and a covariate-augmented ADF (CADF) test. [MultipleBubbles](#) tests for the existence of bubbles based on Phillips-Shi-Yu (2015). Simulation-based unit root tests are provided by [STSD](#).

- *Local stationarity* : [locits](#) provides a test of local stationarity and computes the localized autocovariance. Time series costationarity determination is provided by [costat](#). [LSTS](#) has functions for locally stationary time series analysis. Locally stationary wavelet models for nonstationary time series are implemented in [wavethresh](#) (including estimation, plotting, and simulation functionality for time-varying spectra).
- *Cointegration* : The Engle-Granger two-step method with the Phillips-Ouliaris cointegration test is implemented in [tseries](#) and [urca](#). The latter additionally contains functionality for the Johansen trace and lambda-max tests. [tsDyn](#) provides Johansen's test and AIC/BIC simultaneous rank-lag selection. Parameter estimation and inference in a cointegrating regression are implemented in [cointReg](#). Fractionally cointegrated VAR models are handled by [FCVAR](#).
- *Autoregressive distributed lag (ARDL) models* are provided by [ARDL](#), which constructs the underlying error correction model automatically. [nardl](#) and [ardl.nardl](#) both estimate nonlinear cointegrating autoregressive distributed lag models.

## Nonlinear Time Series Analysis

- *Nonlinear autoregression* : Tools for nonlinear time series analysis are provided in [NTS](#) including threshold autoregressive models, Markov-switching models, convolutional functional autoregressive models, and nonlinearity tests. Various forms of nonlinear autoregression are available in [tsDyn](#) including additive AR, SETAR and LSTAR models, threshold VAR and VECM. [EXPAR](#) provides exponential AR models, while [EXPARMA](#) provides exponential ARMA models. [bentableAR](#) implements Bent-Cable autoregression. [BAYSTAR](#) provides Bayesian analysis of threshold autoregressive models. Mixture AR models are implemented in [mixAR](#) and [uGMAR](#). [setartree](#) implements an SETAR tree algorithm, and a SETAR forest. [tseriesTARMA](#) provides routines for Threshold ARMA model testing fitting and forecasting.
- *Neural network autoregression* : Neural network forecasting based on lagged inputs are provided by [tsDyn](#), [GMDH](#) and [nnfor](#). [Nlnts](#) includes neural network VAR, and a nonlinear version of the Granger causality test based on feedforward neural networks. [TSLSTM](#) provides forecasts using a Long Short Term Memory (LSTM) model, while an enhanced version is implemented in [TSLSTMplus](#). [TSdeeplearning](#) implements LSTM and GRU networks. [TSANN](#) automatically identifies an artificial neural network based on forecasting accuracy.
- [tseriesChaos](#) provides an R implementation of the algorithms from the [TISEAN project](#). [DChaos](#) provides several algorithms for detecting chaotic signals inside univariate time series.
- Autoregression Markov switching models are provided in [MSwM](#), while dependent mixtures of latent Markov models are given in [depmix \(archived\)](#) and [depmixS4](#) for categorical and continuous time series.
- *Tests* : Various tests for nonlinearity are provided in [fNonlinear](#). [tseriesEntropy](#) tests for nonlinear serial dependence based on entropy metrics, while [tseriesTARMA](#) provides tests for nonlinearity based on threshold ARMA models.
- Additional functions for nonlinear time series are available in [nlts](#) and [nonlinearTseries](#).

## Entropy

- [RTransferEntropy](#) measures information flow between time series with Shannon and Renyi transfer entropy.
- An entropy measure based on the Bhattacharya-Hellinger-Matusita distance is implemented in [tseriesEntropy](#).
- Various approximate and sample entropies are computed using [TSEntropies](#).

## Dynamic Regression Models

- *Dynamic linear models* : A convenient interface for fitting dynamic regression models via OLS is available in [dynlm](#); an enhanced approach that also works with other regression functions and more time series classes is implemented in [dyn](#). Gaussian linear state space models can be fitted using [dlm](#) (via maximum likelihood, Kalman filtering/smoothing and Bayesian methods), or using [bsts](#) which uses MCMC. [dLagM](#) provides time series regression with distributed lags. Functions for distributed lag nonlinear modelling are provided in [dlnm](#). [fastTS](#) implements sparsity-ranked lasso methods for time series with exogenous features and/or complex seasonality. [crosslag](#) provides linear and nonlinear cross lag analysis. Distributed lag models based on Bayesian additive regression trees are implemented in [dlmtree](#). [sym.arma](#) will fit ARMA models with regressors where the observations follow a conditional symmetric distribution.
- *Time-varying parameter* models can be fitted using the [tpr](#) package.
- [greybox](#) provides several tools for modelling and forecasting with dynamic regression models.

## Pre-trained transformer models

- [nixtlar](#) allows users to interact with Nixtla's TimeGPT via the API.

## Multivariate Time Series Models

- *Vector autoregressive (VAR) models* are provided via [ar\(\)](#) in the basic stats package including order selection via the AIC. These models are restricted to be stationary. [MTS](#) is an all-purpose toolkit for analysing multivariate time series including VAR, VARMA, seasonal VARMA, VAR models with exogenous variables, multivariate regression with time series errors, and much more. Possibly non-stationary VAR models are fitted in the [mAr](#) package, which also allows VAR models in principal component space. Fractionally cointegrated VAR models are handled by [FCVAR](#). [sparsevar](#) allows estimation of sparse VAR and VECM models, [bigtime](#) estimates large sparse VAR, VARX and VARMA models, while [BigVAR](#) estimates VAR and VARX models with structured lasso penalties and [svars](#) implements data-driven structural VARs. [sstvars](#) provides a toolkit for reduced form and structural smooth transition VARs. Shrinkage estimation methods for VARs are implemented in [VARshrink](#). More elaborate models are provided in package [vars](#) and [tsDyn](#). Another implementation with bootstrapped prediction intervals is given in [VAR.etc](#). [bvartools](#) assists in the set-up of Bayesian VAR models, while [BVAR](#) and [bayesianVARs](#) provide toolkits for hierarchical Bayesian VAR models. [bsvars](#), [bsvarSIGNs](#), and [bvars](#) include efficient algorithms for estimating Bayesian Structural VAR models. [BMTAR](#) and [mtarm](#) implement Bayesian Multivariate Threshold AR models. Factor-augmented VAR (FAVAR) models are estimated by a Bayesian method with [FAVAR](#). [BGVAR](#) implements Bayesian Global VAR models. [mlVAR](#) provides multi-level vector autoregression. [gmvarkit](#) estimates Gaussian mixture VAR models. [GNAR](#) provides methods for fitting network AR models, while [graphicalVAR](#) and [tsnet](#) both estimate graphical VAR models. [gdpc](#) implements generalized dynamic principal components. [pcdpca](#) extends dynamic principal components to periodically correlated multivariate time series. [mgm](#) estimates time-varying mixed graphical models and mixed VAR models via regularized regression. [nets](#) provides estimation of sparse VARs using long run partial correlation networks for time series data. Factor-adjusted VARs using network estimation and forecasting for high-dimensional time series is implemented in [fnets](#).
- *Nonlinear VAR models* are provided by [NVAR](#).
- *Vector error correction models* are available via the [urca](#), [ecm](#), [vars](#), [tsDyn](#) packages, including versions with structural constraints and thresholding.
- *Vector exponential smoothing* is provided by [smooth](#).
- *Dynamic factor models* are available in the [dfms](#) package using EM or two-step estimation. Dynamic factor models with sparse loadings are implemented in [sparseDFM](#). Bayesian dynamic factor analysis is implemented in [bayesdfa](#) and [bvartools](#). [sufficientForecasting](#) implements a factor-based approach to forecasting with dimension reduction and a possibly nonlinear forecasting function.
- *Forecast Linear Augmented Projection (FLAP) methods* are implemented in [flap](#).
- *Time series component analysis* : [ForeCA](#) implements forecastable component analysis by searching for the best linear transformations that make a multivariate time series as forecastable as possible. [HDTSA](#) provides procedures for several high-dimensional time series analysis tools. Frequency-domain-based dynamic PCA is implemented in [freqdom](#). [tsBSS](#) provides blind source separation and supervised

dimension reduction for time series. [sdrt](#) estimates sufficient dimension reduction subspaces for time series.

- *Multivariate state space models* An implementation is provided by the [KFAS](#) package which provides a fast multivariate Kalman filter, smoother, simulation smoother and forecasting. [FKF](#) provides a fast and flexible implementation of the Kalman filter, which can deal with missing values. [FKF.SP](#) implements fast Kalman filtering through sequential processing. [kalmanfilter](#) provides an ‘Rcpp’ implementation of the multivariate Kalman filter for state space models that can handle missing values and exogenous data in the observation and state equations. Another implementation is given in the [dlm](#) package which also contains tools for converting other multivariate models into state space form. [MARSS](#) fits constrained and unconstrained multivariate autoregressive state-space models using an EM algorithm. [mbsts](#) provides tools for multivariate Bayesian structural time series models. All of these packages assume the observational and state error terms are uncorrelated.
- *Partially-observed Markov processes* are a generalization of the usual linear multivariate state space models, allowing non-Gaussian and nonlinear models. These are implemented in the [pomp](#) package.
- Multivariate stochastic volatility models (using latent factors) are provided by [factorstochvol](#). Multivariate ARCH models are implemented in [tsmarch](#).
- High-dimensional sparse multivariate GLARMA models are handled by [MultiGlarmaVarSel](#) including variable selection.
- Multivariate Dynamic Generalized Additive Models are implemented in [mgvam](#).

## Analysis of large groups of time series

- *Time series features* are computed in [feasts](#) for time series in tsibble format. They are computed using [tsfeatures](#) for a list or matrix of time series in ts format. In both packages, many built-in feature functions are included, and users can add their own. [Rcatch22](#) provides fast computation of 22 features identified as particularly useful. [theft](#) calculates time series features from various R and Python packages, while [theftdLc](#) is a companion package providing analysis and visualization functions. Feature extraction for ordinal time series is provided by [otsfeatures](#).
- *Time series clustering* is implemented in [TSclust](#), [dtwclust](#), [BNPTSclust](#) and [pdc](#).
- [TSdist](#) provides distance measures for time series data.
- [TSrepr](#) includes methods for representing time series using dimension reduction and feature extraction.
- Methods for plotting and forecasting collections of hierarchical and grouped time series are provided by [fable](#) and [hts](#). [thief](#) uses hierarchical methods to reconcile forecasts of temporally aggregated time series. [FoReco](#) provides various forecast reconciliation methods for cross-sectional, temporal, and cross-temporal constrained time series. Probabilistic reconciliation of hierarchical forecasts via conditioning is available in [bayesRecon](#). Degenerate hierarchical structures are handled by [htsDegenerateR](#).

## Dynamic time warping

- Dynamic time warping algorithms are provided by [dtw](#) for computing and plotting pairwise alignments between time series.
- Parametric time warping is implemented in [ptw](#).
- [rucrdtw](#) provides R bindings for functions from the UCR Suite to enable ultrafast subsequence search for the best match under Dynamic Time Warping and Euclidean Distance.
- [IncDTW](#) provides incremental calculation of dynamic time warping for streaming time series.
- Time-weighted dynamic time warping is provided by [twdtw](#).

## Functional time series

- Tools for visualizing, modeling, forecasting and analysing functional time series are implemented in `pkg("ftsa")`. [NTS](#) also implements functional autoregressive models. Seasonal functional autoregression models are provided by [Rsfar](#). [fpcb](#) implements predictive confidence bands for functional time series.
- [fdaACF](#) estimates the autocorrelation function for functional time series.
- [freqdom.fda](#) provides implements of dynamical functional principal components for functional time series.
- [STFTS](#) contains stationarity, trend and unit root tests for functional time series.

## Matrix and tensor-valued time series

- [MEFM](#) implements main effect matrix factor models for matrix time series.
- [tensorTS](#) provides functions for estimation, simulation and prediction of factor and autoregressive models for matrix and tensor valued time series.
- Time series tensor factor models are implemented in [TensorPreAve](#).
- [RTFA](#) provides robust factor analysis for tensor time series.

## Continuous time models

- [carfima](#) allows for continuous-time ARFIMA models.
- Simulation and inference for stochastic differential equations is provided by [sde](#) and [yuima](#).
- [Sim.DiffProc](#) simulates and models stochastic differential equations.
- [resde](#) provides maximum likelihood estimation for univariate reducible stochastic differential equation models.

## Resampling

- *Bootstrapping* : The [boot](#) package provides function `tsboot()` for time series bootstrapping, including block bootstrap with several variants. [blocklength](#) allows for selecting the optimal block-length for a dependent bootstrap. `tsbootstrap()` from [tseries](#) provides fast stationary and block bootstrapping. Maximum entropy bootstrap for time series is available in [meboot](#). [BootPR](#) computes bias-corrected forecasting and bootstrap prediction intervals for autoregressive time series. [bootUR](#) implements bootstrap unit root tests.

## Time Series Data

- Various data sets in [tsibble](#) format are provided by [tsibbledata](#).
- [gratis](#) generates new time series with diverse and controllable characteristics using mixture autoregression models.
- Data from Cryer and Chan (2010, 2nd ed) *Time series analysis with applications in R* are in the [TSA](#) package.
- Data from Hyndman and Athanasopoulos (2018, 2nd ed) *Forecasting: principles and practice* are in the [fpp2](#) package.
- Data from Hyndman and Athanasopoulos (2021, 3rd ed) *Forecasting: principles and practice* are in the [fpp3](#) package.
- Data from Hyndman, Koehler, Ord and Snyder (2008) *Forecasting with exponential smoothing* are in the [expsmooth](#) package.
- Data from Makridakis, Wheelwright and Hyndman (1998, 3rd ed) *Forecasting: methods and applications* are in the [fma](#) package.
- Data from Shumway and Stoffer (2017, 4th ed) *Time Series Analysis and Its Applications: With R Examples* are in the [astsa](#) package.
- Data from Tsay (2005, 2nd ed) *Analysis of Financial Time Series* are in the [FinTS](#) package.
- Data from Woodward, Gray, and Elliott (2016, 2nd ed) *Applied Time Series Analysis with R* are in the [tswge](#) package.
- [AER](#) and [Ecdat](#) both contain many data sets (including time series data) from many econometrics text books
- Data from the M and M3 forecasting competitions are provided in the [Mcomp](#) package. [Tcomp](#) provides data from the 2010 IJF Tourism Forecasting Competition. The M4 competition data are available from [M4comp2018](#). Data from the M5 forecasting competition can be downloaded using [m5](#).
- *National time series data*: [readabs](#) downloads, imports and tidies time series data from the [Australian Bureau of Statistics](#). [bbk](#) provides access to the German Deutsche Bundesbank and European Central Bank time series data. [bundesbank](#) also allows access to the time series databases of the Deutsche Bundesbank, while data from the European Central Bank can also be accessed via [ecb](#). [BETS](#) provides access to the most important economic time series in *Brazil*. Data from the Banque de France can be

downloaded using [rwebstat](#). Data from *Switzerland* via [dataseries.org](#) can be downloaded and imported using [dataseries](#). Macroeconomic time series for Africa can be obtained via [africamonitor](#). [ugatsdb](#) provides an API to access time series data for *Uganda*, while [samadb](#) does the same for *South Africa*. Economic time series and other data from FRED (the Federal Reserve Economic Data) can be retrieved using [fredr](#).

- *Time series databases*: [rdbnomics](#) provides access to hundreds of millions of time series from [DBnomics](#). [ifo](#) is a client for downloading time series data from the Ifo Institute. [influxdbr](#) provides an interface to the InfluxDB time series database. [pdfetch](#) provides facilities for downloading economic and financial time series from public sources. Data from the [Quandl](#) online portal to financial, economical and social datasets can be queried interactively using the [Quandl](#) package. [tsdb](#) implements a simple database for numerical time series.
- *Synthetic data* are produced by `simulate()` in [forecast](#) package or `generate()` in [fable](#), given a specific model. [gratis](#) generates new time series with diverse and controllable characteristics using mixture autoregression models. [synthesis](#) generates synthetic time series from commonly used statistical models, including linear, nonlinear and chaotic systems. [tssim](#) flexibly simulates daily or monthly time series using seasonal, calendar, and outlier components.

## Miscellaneous

- [complex](#) implements functions for complex-valued time series analysis and forecasting.
- [EBMAforecast](#): Ensemble Bayesian model averaging forecasts using Gibbs sampling or EM algorithms.
- [ensembleBMA](#): Bayesian Model Averaging to create probabilistic forecasts from ensemble forecasts and weather observations.
- [FeedbackTS](#): Analysis of fragmented time directionality to investigate feedback in time series.
- [gsignal](#) is an R implementation of the Octave package “signal”, containing a variety of signal processing tools.
- [paleoTS](#): Modeling evolution in paleontological time series.
- [pastecs](#): Regulation, decomposition and analysis of space-time series.
- [PSF](#): Forecasting univariate time series using pattern-sequences.
- [RGENERATE](#) provides tools to generate vector time series.
- [RMAWGEN](#) is set of S3 and S4 functions for spatial multi-site stochastic generation of daily time-series of temperature and precipitation making use of VAR models. The package can be used in climatology and statistical hydrology.
- [RSEIS](#): Seismic time series analysis tools.
- [rts](#): Raster time series analysis (e.g., time series of satellite images).
- [SLBDD](#): Functions for analysing large-scale time series, based on the book “Statistical Learning with Big Dependent Data” (Pena & Tsay, 2021).
- [spTimer](#): Spatio-temporal Bayesian modelling.
- [surveillance](#): Temporal and spatio-temporal modeling and monitoring of epidemic phenomena.
- [Tides](#): Functions to calculate characteristics of quasi periodic time series, e.g. observed estuarine water levels.
- [TSEAL](#): Multivariate time series classification based on a Discrete Wavelet Transform.
- [tsfknn](#): Time series forecasting with k-nearest-neighbours.
- [tsModel](#): Time series modeling for air pollution and health.

## CRAN packages

*Core:* [fable](#), [feasts](#), [forecast](#), [tseries](#), [tsibble](#), [zoo](#).

*Regular:* [acp](#), [AER](#), [africamonitor](#), [aion](#), [almanac](#), [anytime](#), [ARCensReg](#), [ArDec](#), [ARDL](#), [ardl.nardl](#), [arfima](#), [arima2](#), [ASSA](#), [astsa](#), [autostsm](#), [BayesARIMAX](#), [bayesdfa](#), [bayesforecast](#), [bayesianVARs](#), [bayesRecon](#), [BAYSTAR](#), [bbk](#), [bentcableAR](#), [BETS](#), [beyondWhittle](#), [bfast](#), [BGVAR](#), [bigtime](#), [BigVAR](#), [binsegRcpp](#), [biwavelet](#), [blocklength](#), [BMTAR](#), [BNPTSclust](#), [boiwsa](#), [boot](#), [BootPR](#), [bootUR](#), [breakfast](#), [bspec](#), [bssm](#), [bsts](#), [bsvars](#), [bsvarSIGNs](#), [bundesbank](#), [BVAR](#), [bvars](#), [bvartools](#), [CADFtest](#), [carfima](#), [CFtime](#), [changepoint](#), [changepoint.geo](#), [changepoint.np](#), [chron](#), [clock](#), [CNLTsfa](#),

[coconots](#), [cointReg](#), [collapse](#), [complex](#), [corbouli](#), [costat](#), [CptNonPar](#), [crosslag](#), [ctbi](#), [data.table](#), [dataseries](#), [datetimeoffset](#), [DChaos](#), [dCovTS](#), [depmixS4](#), [deseats](#), [dfms](#), [diffusion](#), [DIMORA](#), [disaggR](#), [dLagM](#), [dlm](#), [dlmtree](#), [dlnm](#), [dsa](#), [DTSG](#), [dtts](#), [dtw](#), [dtwclust](#), [dygraphs](#), [dyn](#), [dynlm](#), [EBMAforecast](#), [ecb](#), [Ecdat](#), [ecm](#), [ecp](#), [EMD](#), [ensembleBMA](#), [era](#), [esemifar](#), [EXPAR](#), [EXPARMA](#), [expsmooth](#), [fable.prophet](#), [fableCount](#), [fabletools](#), [factorstochvol](#), [fanplot](#), [fastcpd](#), [fastTS](#), [FAVAR](#), [FCVAR](#), [fdaACF](#), [FeedbackTS](#), [fGarch](#), [finnts](#), [FinTS](#), [FKF](#), [FKF.SP](#), [flap](#), [fma](#), [fnets](#), [fNonlinear](#), [ForeCA](#), [ForecastComb](#), [forecastHybrid](#), [forecastLSW](#), [forecastML](#), [FoReco](#), [ForeComp](#), [forecTheta](#), [fpch](#), [fpop](#), [fpp2](#), [fpp3](#), [fracdiff](#), [fredr](#), [freqdom](#), [freqdom.fda](#), [funtimes](#), [garma](#), [GAS](#), [gasmodel](#), [gdpc](#), [ggdemetra](#), [ggseas](#), [glarma](#), [GlarmaVarSel](#), [GMDH](#), [gmvarkit](#), [GNAR](#), [graphicalVAR](#), [gratis](#), [gravitas](#), [greybox](#), [gsarima](#), [gsignal](#), [HDTSA](#), [hht](#), [hpfilter](#), [hts](#), [htsDegenerateR](#), [hwwntest](#), [ifo](#), [imputeTestbench](#), [imputeTS](#), [IncDTW](#), [influxdb](#), [InspectChangepoint](#), [itsmr](#), [jalcal](#), [jointseg](#), [kalmanfilter](#), [kDGLM](#), [KFAS](#), [kza](#), [legion](#), [locits](#), [lomb](#), [LongMemoryTS](#), [lpacf](#), [LSTS](#), [LSWPlib](#), [ltsa](#), [lubridate](#), [m5](#), [MAPA](#), [mAr](#), [marls](#), [MARSS](#), [mbsts](#), [Mcomp](#), [meboot](#), [MEFM](#), [mFilter](#), [mgm](#), [mixAR](#), [mlVAR](#), [modeltime](#), [modeltime.resample](#), [mondate](#), [mosum](#), [mrf](#), [MSwM](#), [mtarm](#), [MTS](#), [mtsdi](#), [multDM](#), [MultiGlarmaVarSel](#), [MultipleBubbles](#), [multitaper](#), [mvgam](#), [mvLSW](#), [mvLSWimpute](#), [nanotime](#), [nardl](#), [nets](#), [nixtla](#), [NlinTS](#), [nlts](#), [nnfor](#), [nonlinearTseries](#), [nsarfima](#), [NTS](#), [NVAR](#), [onlineforecast](#), [opera](#), [otsfeatures](#), [paleoTS](#), [partsm](#), [parttime](#), [pastecs](#), [pcdpca](#), [pcts](#), [pdc](#), [pdfetch](#), [peacots](#), [perARMA](#), [pomp](#), [portes](#), [profoc](#), [prophet](#), [psd](#), [PSF](#), [PTSR](#), [ptw](#), [qlcal](#), [Quandl](#), [quantspec](#), [Rbeast](#), [Rcatch22](#), [rdbnomics](#), [readabs](#), [regspec](#), [resde](#), [RGENERATE](#), [rhosa](#), [RJDemetra](#), [Rlgt](#), [Rlibeemd](#), [RMAWGEN](#), [robfilter](#), [RobKF](#), [robustarma](#), [roll](#), [RSEIS](#), [Rsfar](#), [Rssa](#), [RTFA](#), [RTransferEntropy](#), [rts](#), [rucrdtw](#), [rugarch](#), [runner](#), [runstats](#), [rwebstat](#), [samadb](#), [sazedR](#), [scoringRules](#), [scoringutils](#), [sde](#), [sdrt](#), [seas](#), [season](#), [seasonal](#), [seasonalview](#), [seer](#), [setartree](#), [signal](#), [Sim.DiffProc](#), [SLBDD](#), [sleekts](#), [slider](#), [smooth](#), [smoots](#), [sparseDFM](#), [sparsevar](#), [spectral](#), [spINAR](#), [spTimer](#), [sstvars](#), [statespacer](#), [STFTS](#), [stlplus](#), [stochvol](#), [stR](#), [strucchange](#), [strucchangeRcpp](#), [StructuralDecompose](#), [sTSD](#), [sufficientForecasting](#), [sugrrants](#), [surveillance](#), [svars](#), [sweep](#), [sym.arma](#), [synthesis](#), [tbrf](#), [Tcomp](#), [tempdisagg](#), [TensorPreAve](#), [tensorTS](#), [testcorr](#), [tfarima](#), [tframe](#), [theft](#), [theftdlc](#), [thief](#), [Tides](#), [tidychange](#), [tidychange](#), [timechange](#), [timeDate](#), [timeSeries](#), [timeseriesdb](#), [timetk](#), [timsac](#), [tis](#), [tpr](#), [trend](#), [TSA](#), [TSANN](#), [tsbox](#), [tsBSS](#), [TSclust](#), [tscount](#), [tsdataleaks](#), [tsdb](#), [tsdecomp](#), [TSdeeplearning](#), [tsdisagg2](#), [TSdisaggregation](#), [TSdist](#), [tsDyn](#), [TSEAL](#), [TSEntropies](#), [tseriesChaos](#), [tseriesEntropy](#), [tseriesTARMA](#), [tsfeatures](#), [tsfknn](#), [tsgarch](#), [tsgc](#), [tsibbledata](#), [tsibbletalk](#), [tsintermittent](#), [TSLSTM](#), [TSLSTMplus](#), [tsmarch](#), [tsModel](#), [tsnet](#), [tsoutliers](#), [tsPI](#), [TSrepr](#), [tsrobprep](#), [tssim](#), [TSstudio](#), [tstests](#), [TSTutorial](#), [tsutils](#), [tswge](#), [twdtw](#), [UComp](#), [ugatsdb](#), [uGMAR](#), [urca](#), [uroot](#), [VAR.etc](#), [VARDetect](#), [vars](#), [VARshrink](#), [VedicDateTime](#), [WaveletComp](#), [wavelets](#), [waveslim](#), [wavethresh](#), [wavScalogram](#), [WeightedPortTest](#), [wktmo](#), [x12](#), [x13binary](#), [xts](#), [yuima](#), [ZIM](#), [ZINARp](#), [ZRA](#).

Archived: [depmix](#).

## Related links

- [TISEAN Project](#)

## Other resources

- CRAN Task View: [Econometrics](#)
- CRAN Task View: [Environmetrics](#)
- CRAN Task View: [Finance](#)
- GitHub Project: [M4comp2018](#)