The Innovations State Space Model

Alexios Galanos

2025-04-26

Contents

1 Introduction

The `tsissm` package implements the linear innovations state space model described by De Livera et al. (2011)[1] and incorporating trigonometric seasonality as used in the `tbats` function from the `forecast` package of Hyndman et al. (2024). However, `tsissm` differs significantly in both implementation and features. Key enhancements include:

1. **Automatic differentiation and robust inference**:
   Estimation leverages automatic differentiation (autodiff), with multiple sandwich estimators available for standard error calculation. System forecastability and ARMA constraints are handled exactly via the nloptr solver using autodiff-based Jacobians.

2. **Flexible error distributions**:
   In addition to Gaussian errors, the model supports heavy-tailed and skewed alternatives, including the Student's t distribution and the Johnson's SU distribution.

3. **Heteroscedasticity modeling**:
   Conditional heteroscedasticity is supported via a GARCH specification on the innovation variance.

4. **Automatic model selection and ensembling**:
   Users may select the best model based on an information criterion (e.g., AIC or BIC) or retain the

---

[1]Originally proposed by Anderson and Moore (2012)

top N models for ensembling. This feature is also available in the `backtest` method, providing a more robust approach to evaluation and forecasting.

5. **Simulated predictive distributions**:
   The `predict` method always returns a simulated forecast distribution, alongside the analytic mean, allowing for richer uncertainty quantification.

6. **Handling of missing data**:
   Missing values in the response variable are permitted and automatically handled via the state space formulation.

7. **Support for external regressors**:
   Regressors are supported in the mean.[2]

2   Model Formulation

Given an initial state vector of unobserved components (such as level, slope, and seasonality), the proposed model evolves the states over time using a linear transition equation, incorporating the effect of the most recent observation error. At each time step, the observed (Box-Cox transformed) value is modeled as a linear combination of the previous state, lagged external regressors, and a normally[3] distributed random error. This structure allows the model to capture complex patterns in the data—such as trends, seasonal cycles, and the influence of exogenous variables—while dynamically updating its internal state based on new information.

Consider the following Single Error Model (SEM) with trigonometric seasonality:

$$
\begin{aligned}
y_t^{(\lambda)} &= \mathbf{w}'\mathbf{x}_{t-1} + \mathbf{c}'\mathbf{u}_{t-1} + \varepsilon_{\mathbf{t}}, \quad \varepsilon_{\mathbf{t}} \sim \mathbf{N}\left(\mathbf{0}, \sigma^{\mathbf{2}}\right), \\
\mathbf{x}_t &= \mathbf{F}\mathbf{x}_{t-1} + \mathbf{g}\varepsilon_t,
\end{aligned}
\tag{1}
$$

where λ represents the Box Cox parameter, $\mathbf{w}$ the observation coefficient vector, $\mathbf{x_t}$ the unobserved state vector, and $\mathbf{c}$ a vector of coefficients on the external regressor set $\mathbf{u}$.[4]

Define the state vector[5] as:

$$
\mathbf{x_t} = \left(\mathbf{l_t}, \mathbf{b_t}, \mathbf{s_t^{(1)}}, \ldots, \mathbf{s_t^{(T)}}, \mathbf{d_t}, \mathbf{d_{t-1}}, \ldots, \mathbf{d_{t-p-1}}, \varepsilon_{\mathbf{t}}, \varepsilon_{\mathbf{t-1}}, \ldots, \varepsilon_{\mathbf{t-q-1}}\right)',
\tag{2}
$$

where $\mathbf{s_t^{(i)}}$ is the row vector $\left(s_{1,t}^{(i)}, s_{2,t}^{(i)}, \ldots, s_{k_i,t}^{(i)}, s_{1,t}^{*(i)}, s_{2,t}^{*(i)}, \ldots, s_{k_i,t}^{*(i)}\right)$ for the trigonometric seasonality. Also define $\mathbf{1_r}$ and $\mathbf{0_r}$ as a vector of ones and zeros, respectively, of length r, $\mathbf{O_{u,v}}$ a $u \times v$ matrix of zeros and $\mathbf{I_{u,v}}$ a $u \times v$ diagonal matrix of ones; let $\gamma = \left(\gamma^{(1)}, \ldots, \gamma^{(T)}\right)$ be a vector of seasonal parameters with $\gamma^{(i)} = \left(\gamma_1^{(i)}\mathbf{1}_{k_i}, \gamma_2^{(i)}\mathbf{1}_{k_i}\right)$ (with k harmonics); $\theta = (\theta_1, \theta_2, \ldots, \theta_p)$ and $\psi = (\psi_1, \psi_2, \ldots, \psi_q)$ as the vector of AR(p) and MA(q) parameters, respectively. The observation transition vector $\mathbf{w} = (1, \phi, \mathbf{a}, \theta, \psi)'$, where $\mathbf{a} = \left(\mathbf{a}^{(1)}, \ldots, \mathbf{a}^{(T)}\right)$ with $\mathbf{a}^{(i)} = (\mathbf{1}_{k_i}, \mathbf{0}_{k_i})$. The state error adjustment vector $\mathbf{g} = (\alpha, \beta, \gamma, 1, \mathbf{0}_{p-1}, 1, \mathbf{0}_{q-1})'$. Further, let $\mathbf{B} = \gamma'\theta$, $\mathbf{C} = \gamma'\psi$ and $\mathbf{A} = \oplus_{i=1}^{T}\mathbf{A}_i$, with

$$
\mathbf{A}_i = \begin{bmatrix} \mathbf{C}^{(i)} & \mathbf{S}^{(i)} \\ -\mathbf{S}^{(i)} & \mathbf{C}^{(i)} \end{bmatrix},
\tag{3}
$$

and with $\mathbf{C^{(i)}}$ and $\mathbf{S^{(i)}}$ representing the $k_i \times k_i$ diagonal matrices with elements $cos(\lambda_j^{(i)})$ and $sin(\lambda_j^{(i)})$[6] respectively.[7]

---

[2]A future enhancement may incorporate regularization.

[3]In our formulation we relax this to allow for other choices as well.

[4]In the package, it is expected that the regression matrix is already pre-lagged.

[5]The following equations apply for the case when all components are present (level, slope, seasonal and ARMA).

[6]λ here should not be confused with the Box Cox lambda.

[7]For $j = 1, \ldots, k_i$ and $i = 1, \ldots, T$, representing the number of harmonics k per seasonal period i.

Finally, the state transition matrix $\mathbf{F}$ is composed as follows:

$$\mathbf{F} = \begin{bmatrix} 1 & \phi & \mathbf{0}_\tau & \alpha\theta & \alpha\psi \\ 0 & \phi & \mathbf{0}_\tau & \beta\theta & \beta\psi \\ \mathbf{0}'_\tau & \mathbf{0}'_\tau & \mathbf{A} & \mathbf{B} & \mathbf{C} \\ 0 & 0 & \mathbf{0}_\tau & \theta & \psi \\ \mathbf{0}'_{p-1} & \mathbf{0}'_{p-1} & \mathbf{O}_{p-1,\tau} & \mathbf{I}_{p-1,p} & \mathbf{O}_{p-1,q} \\ 0 & 0 & \mathbf{0}_\tau & \mathbf{0}_p & \mathbf{0}_q \\ \mathbf{0}'_{q-1} & \mathbf{0}'_{q-1} & \mathbf{O}_{q-1,\tau} & \mathbf{O}_{q-1,p} & \mathbf{I}_{q-1,q} \end{bmatrix} \tag{4}$$

where $\tau = 2 \sum_{i=1}^{T} k_i$. The model has the feature of allowing for multiple seasonal trigonometric components.

3   State Initialization

A key innovation of the De Livera et al. (2011) paper is in providing the exact initialization of the non-stationary component's seed states, the exponential smoothing analogue of the De Jong (1991) method for augmenting the Kalman filter to handle seed states with infinite variances. The proof, based on De Livera et al. (2011) and expanded here is as follows, let:

$$\mathbf{D} = \mathbf{F} - \mathbf{g}\mathbf{w}'. \tag{5}$$

We eliminate ε_t in 1 to give:

$$\mathbf{x}_t = \mathbf{D}\mathbf{x}_{t-1} + \mathbf{g}\mathbf{y}_t. \tag{6}$$

Next, we proceed by backsolving the equation for the error, given a given value of λ:[8]

$$\begin{aligned} \varepsilon_t &= y_t - \mathbf{w}\hat{\mathbf{x}}_{t-1}, \\ \varepsilon_t &= y_t - \mathbf{w}'\left(\mathbf{D}\hat{\mathbf{x}}_{t-2} + \mathbf{g}y_{t-1}\right). \end{aligned} \tag{7}$$

Starting with $t = 4$ and working backwards:

$$\begin{aligned} \varepsilon_4 &= y_4 - \mathbf{w}'\left(\mathbf{D}\hat{\mathbf{x}}_2 + \mathbf{g}y_3\right) \\ &= y_4 - \mathbf{w}'\left(\mathbf{D}\left(\mathbf{D}\hat{\mathbf{x}}_1 + \mathbf{g}y_2\right) + \mathbf{g}y_3\right) \\ &= y_4 - \mathbf{w}'\left(\mathbf{D}\left(\mathbf{D}\left(\mathbf{D}\hat{\mathbf{x}}_0 + \mathbf{g}y_1\right) + \mathbf{g}y_2\right) + \mathbf{g}y_3\right) \\ &= y_4 - \mathbf{w}'\left(\mathbf{D}\left(\mathbf{D}^2\mathbf{x}_0 + \mathbf{D}\mathbf{g}y_1 + \mathbf{g}y_2\right) + \mathbf{g}y_3\right) \\ &= y_4 - \mathbf{w}'\left(\mathbf{D}^3\mathbf{x}_0 + \mathbf{D}^2\mathbf{g}y_1 + \mathbf{D}\mathbf{g}y_2 + \mathbf{g}y_3\right) \\ &= y_4 - \mathbf{w}' \sum_{j=1}^{3} \mathbf{D}^{j-1}\mathbf{g}y_{4-j} - \mathbf{w}'\mathbf{D}^3\mathbf{x}_0 \end{aligned} \tag{8}$$

and generalizing to ε_t:

$$\begin{aligned} \varepsilon_t &= y_t - \mathbf{w}'\left(\sum_{j=1}^{t-1} \mathbf{D}^{j-1}\mathbf{g}y_{t-j}\right) - \mathbf{w}'\mathbf{D}^{t-1}\mathbf{x}_0 \\ &= y_t - \mathbf{w}'\tilde{\mathbf{x}}_{t-1} - \mathbf{w}'_{t-1}\mathbf{x}_0 \\ &= \tilde{y}_t - \mathbf{w}'_{t-1}\mathbf{x}_0, \end{aligned} \tag{9}$$

where $\tilde{y}_t = y_t - \mathbf{w}'\tilde{\mathbf{x}}_{t-1}$, $\tilde{\mathbf{x}}_t = \mathbf{D}\tilde{\mathbf{x}}_{t-1} + \mathbf{g}y_t$, $\mathbf{w}'_t = \mathbf{D}\mathbf{w}'_{t-1}$, $\tilde{\mathbf{x}}_0 = 0$ and $\mathbf{w}'_0 = \mathbf{w}'$, so that $\mathbf{x}_0$ are the coefficients from the regression of $\mathbf{w}$ on $\boldsymbol{\varepsilon}$. The way this is implemented, is to calculate the seed states based

---

[8]For simplicity of exposition, y_t is equivalent to $y_t^{(\lambda)}$.

3

on the initial parameter vector and λ parameter and then use those same seed states without re-calculating during the optimization. However, when λ is also part of the parameter estimation set, we have chosen instead to re-calculate the seed state during the optimization as part of the autodiff tape. This differs from the approach adopted in the `forecast` package of Hyndman et al. (2024) which simply re-transforms the initial seed states based on the value of λ during estimation.

4   Constraints

A number of constraints are implemented during estimation, including a system forecastability constraint and a constraint on the ARMA parameters when present.

4.1   System Forecastability Constraint

The forecastability constraint necessitates that the characteristic roots of the matrix D (see (5)) lie within the unit circle, which means that the maximum of the modulus of the eigenvalues of D are less than 1. In the `forecast` package this constraint is directly checked and returns Inf when the condition is violated which is a type of infinite penalty method. It is known that this type of approach can lead to numerical instabilities and discontinuities as well as difficulties in convergence, though it often "works" in practice to some degree. Instead, in the `tsissm` package, we model the constraint using RTMB[9] to obtain the autodiff based Jacobian of the constraint and pass the output to the `nloptr` solver using the `eval_g_ineq` and `eval_jac_g_ineq` arguments. Specifically, in order to avoid the use of the non-differentiable max operator, we impose that the modulus of all the eigenvalues is less than 1 and the Jacobian is then a $C \times N$ matrix where C are the constraints and N the number of parameters.[10]

4.2   ARMA Constraint

It is typical to check for stationarity in the AR component and invertibility in the MA component when those are present, by solving for the characteristic roots of the system polynomials. In R, one way to solve for this is to use the `polyroot` function such that `Mod(polyroot(c(1, -ar)))` and `Mod(polyroot(c(1, ma)))` are greater than 1. In the `forecast` package this is how they are checked and an infinite penalty applied, similar to the system forecastability constraint. As in our previous approach, we instead code this up to make use of automatic differentiation in order to obtain a reasonable set of constraints and their Jacobians.

Consider an autoregressive (AR) model specified by the polynomial

$$1 - a_1 z - a_2 z^2 - \cdots - a_n z^n = 0. \tag{10}$$

The stationarity condition requires that the roots z of this polynomial satisfy $|z| > 1$.

That is, the roots must lie outside the unit circle.

Step 1: Rewriting in Monic Form

Multiply the equation by -1 to obtain:

$$-1 + a_1 z + a_2 z^2 + \cdots + a_n z^n = 0. \tag{11}$$

Rearrange this as:

$$a_n z^n + a_{n-1} z^{n-1} + \cdots + a_1 z - 1 = 0. \tag{12}$$

---

[9]RTMB instead of TMB was used for the constraint due to issues discussed here.

[10]In practice we could have just imposed the constraint on the first eigenvalue, but since D is non-symmetric, we have no *guarantees*, from what I have read, that the LAPACK routine (`dgeev`) will always return the eigenvalues in decreasing order by modulus.

Dividing through by a_n (assuming $a_n \neq 0$) yields the monic polynomial:

$$z^n + \frac{a_{n-1}}{a_n} z^{n-1} + \cdots + \frac{a_1}{a_n} z - \frac{1}{a_n} = 0. \tag{13}$$

Step 2: Constructing the Companion Matrix

For a monic polynomial

$$z^n + c_{n-1} z^{n-1} + \cdots + c_1 z + c_0 = 0, \tag{14}$$

the standard companion matrix is defined as:

$$C = \begin{pmatrix} -c_{n-1} & -c_{n-2} & \cdots & -c_1 & -c_0 \\ 1 & 0 & \cdots & 0 & 0 \\ 0 & 1 & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & 1 & 0 \end{pmatrix}. \tag{15}$$

In our case, by comparing coefficients, we have:

$$c_{n-1} = \frac{a_{n-1}}{a_n}, \quad c_{n-2} = \frac{a_{n-2}}{a_n}, \quad \ldots, \quad c_1 = \frac{a_1}{a_n}, \quad c_0 = -\frac{1}{a_n}. \tag{16}$$

Thus, the companion matrix becomes:

$$C = \begin{pmatrix} -\dfrac{a_{n-1}}{a_n} & -\dfrac{a_{n-2}}{a_n} & \cdots & -\dfrac{a_1}{a_n} & \dfrac{1}{a_n} \\ 1 & 0 & \cdots & 0 & 0 \\ 0 & 1 & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & 1 & 0 \end{pmatrix}. \tag{17}$$

Step 3: Stationarity Constraint via the Companion Matrix

Since the eigenvalues of the companion matrix C are precisely the roots z of the polynomial

$$1 - a_1 z - a_2 z^2 - \cdots - a_n z^n = 0, \tag{18}$$

the stationarity condition $|z| > 1$ can be reformulated in terms of the companion matrix as

$$|\lambda_i(C)| > 1 \quad \text{for } i = 1, \ldots, n. \tag{19}$$

That is, the moduli of the eigenvalues of the companion matrix must be greater than 1. A detailed exposition can be found in Lütkepohl (2005) (see Chapter 3).

A similar constraint is imposed in the presence of an MA component, by flipping the sign of the coefficients. It should be noted that these constraints are only applied when the order of either the AR or MA components is greater than 1, else the normal parameter bounds ({-1, 1}) are sufficient.

5 Estimation

The estimation is carried out by minimizing the negative of the log-likelihood, subject to various constraints discussed in 4, and parameter bounds. The log-likelihood (to be maximized), jointly with the Box-Cox transformation (parameter λ) is derived as follows:

$$
\begin{aligned}
p(y_t \mid x_0, \vartheta, \sigma^2) &= p\left(y_t^{(\lambda)} \mid x_0, \vartheta, \sigma^2\right) \det\left(\frac{\partial y_t^{(\lambda)}}{\partial y}\right) \\
&= p\left(y_t^{(\lambda)} \mid x_0, \vartheta, \sigma^2\right) \prod_{t=1}^{n} y_t^{\lambda-1} \\
&= \ln p\left(y_t^{(\lambda)} \mid x_0, \vartheta, \sigma^2\right) + (\lambda - 1) \sum_{t=1}^{n} \ln y_t.
\end{aligned}
\tag{20}
$$

where $p(\cdot)$ represents the probability density function, given the initial state observations x_0, the parameter vector θ and the variance σ^2.

The Box-Cox transformation, represented by parameter λ is a power transformation applied to the dependent variable to stabilize its variance and to make its distribution closer to normal. This adjustment often leads to improved model estimation and inference. Since we are jointly estimating lambda, it is necessary to adjust the likelihood by the Jacobian of the transformation, which in the likelihood expression appears as the determinant term

$$
\det\left(\frac{\partial y_t^{(\lambda)}}{\partial y}\right),
$$

which is equivalently expressed as

$$
\prod_{t=1}^{n} y_t^{\lambda-1}.
$$

This term scales the probability density correctly back to the original data scale, ensuring that the transformation is properly accounted for during estimation. Including the parameter λ in the estimation process allows the model to choose the optimal transformation that best normalizes the data and stabilizes its variance.

Beyond the Gaussian distribution, the package also implements the Student's t and Johnson's SU, details of which can be found in the tsdistributions package. Additionally, the variance can follow GARCH dynamics such that:

$$
\sigma_t^2 = \hat{\sigma}^2 (1 - P) + \sum_{j=1}^{q} \alpha_j \varepsilon_{t-j}^2 + \sum_{j=1}^{p} \beta_j \sigma_{t-j}^2
\tag{21}
$$

where we impose a variance targeting intercept instead of estimating ω, with P being the persistence and $\hat{\sigma}^2$ the unconditional variance. Initialization of the recursion can either use the full information set or a subsample (for more details see the online documentation of tsgarch).

The optimization is undertaken using the nloptr solver, making use of an autodiff based gradient for the parameters and autodiff based Jacobian for the constraints. The solver defaults to the use of the SQP variant, but other options are available via the `issm_control` function.

In order to calculate sandwich estimators[11] for the standard errors, the scores (Jacobian) of the log-likelihood function need to be calculated. Due to the expense of this operation on large datasets, we have implemented asynchronous evaluation which requires the use of a parallel worker set up using a future plan. The user can also turn off the evaluation of scores, in which case the autodiff based Hessian is used for the calculation of the standard errors.

---

[11]We make use of the sandwich package by exporting `estfun` and `bread` methods.

6 Prediction

The h-step ahead analytic mean and variance of the model, in the Box-Cox transformed space, are given by:

$$E\left(y_{n+h|n}^{(\lambda)}\right) = \mu_{t+h} = \mathbf{w}'\mathbf{F}^{h-1}\mathbf{x}_n \tag{22}$$

$$V\left(y_{n+h|n}^{(\lambda)}\right) = \sigma_{t+h}^2 = \begin{cases} \hat{\sigma}^2, & \text{if } h = 1, \\ \hat{\sigma}^2\left[1 + \sum\limits_{j=1}^{h-1} c_j^2\right], & \text{if } h \geq 2. \end{cases} \tag{23}$$

where $c_j = w'F^{j-1}g$. For the mean, a reasonable approximation in back-transformed space, using a second-order Taylor expansion, is given by:

$$y_{t+h} = \begin{cases} \exp(\mu_{t+h})\left(1 + \frac{\sigma_{t+h}^2}{2}\right), & \text{if } \lambda = 0, \\ (\lambda\mu_{t+h} + 1)^{\frac{1}{\lambda}}\left(1 + \frac{\sigma_{t+h}^2(1-\lambda)}{2(\lambda\mu_{t+h}+1)^2}\right), & \text{if } \lambda \neq 0. \end{cases} \tag{24}$$

For the variance in the back transformed space, one should use the simulated distribution to approximate this as it was not possible to find a reasonable approximation, even using higher order Taylor expansions, which would be good enough, particularly with increasing h.

7 Automatic Selection and Ensembling

The `forecast` package of Hyndman et al. (2024) takes a smart, automated approach to identifying the best model from a set of candidate specifications—such as whether to include a slope, the number of harmonics, and the number of AR or MA terms.

In contrast, the `tsissm` package supports **complete enumeration** of all model configurations, including multiple seasonalities and multiple harmonics per seasonal frequency. It also allows testing for constant versus dynamic variance, though only one distributional assumption can be tested at a time. Users can return either the best model based on an information criterion (AIC or BIC), or the top N models ranked by the selected criterion. This flexibility enables **model ensembling** for filtering, prediction, and simulation.

The backtesting function (`tsbacktest`) supports this functionality directly, allowing automatic selection of the top N models. Three weighting schemes are available:

- **User-supplied fixed weights.**
- **AIC-based weights:**
  Models are weighted by the Akaike Information Criterion, favoring those with lower AIC:
  $$w_i = \frac{\exp(-0.5\,\Delta_i)}{\sum_j \exp(-0.5\,\Delta_j)}, \quad \text{where } \Delta_i = AIC_i - \min_j AIC_j.$$
- **BIC-based weights:**
  Similar to AIC-based weights, but using the Bayesian Information Criterion, which penalizes complexity more heavily.

The final ensemble prediction is computed as a weighted average:

$$\hat{y}_{\text{ensemble}} = \sum_i w_i\,\hat{y}_i,$$

where $\hat{y}_i$ are individual model predictions and w_i are the corresponding model weights. AIC-based weighting is typically preferred when models vary in complexity but are fit on the same dataset, whereas BIC-based weighting may be more appropriate for large sample sizes due to its stronger complexity penalty.

Beyond point forecasts, we also ensemble **simulated forecasts**, accounting for error dependencies across models. Rather than assuming independence, we explicitly model the dependence structure of model residuals using a **Gaussian copula**. This provides more accurate risk quantification, especially when the top N models are highly correlated.

First, the correlation of transformed residuals across retained models is computed using **Kendall's tau**[12]. This is then converted into a correlation matrix $\mathbf{R}$ suitable for a Gaussian copula:

$$R_{k\ell} = \sin\left(\frac{\pi}{2}\tau^{(k,\ell)}\right).$$

From this, correlated quantile samples are drawn from a multivariate normal distribution and passed into the prediction function via the `innov` argument with `innov_type = "q"` (quantiles). These quantiles are then transformed back into residuals using each model's error distribution, allowing for simulated forecasts with cross-model error dependence.

Formally, for each model $k = 1, \ldots, K$, simulation j, and forecast horizon i, we define the simulation equations:

$$y_{j,i}^{(k)} = \left(x_{i-1}^{(k)}\right)^\top w^{(k)} + \left(X_i^{(k)}\right)^\top \kappa^{(k)} + E_{j,i}^{(k)}$$
$$x_i^{(k)} = F^{(k)} x_{i-1}^{(k)} + g^{(k)} E_{j,i}^{(k)}$$

The simulated error term $E_{j,i}^{(k)}$ is generated as:

$$E_{j,i}^{(k)} = F_k^{-1}\left(\Phi\left(z_{j,i}^{(k)}\right)\right), \quad \text{where } \mathbf{z}_{j,i} \sim \mathcal{N}(\mathbf{0}, \mathbf{R}).$$

Here:

- $\Phi(\cdot)$ is the standard normal cumulative distribution function (CDF),
- $F_k^{-1}(\cdot)$ is the quantile function (inverse CDF) of the error distribution for model k,
- $z_{j,i}^{(k)}$ is the k-th element of the multivariate Gaussian sample,
- $\mathbf{R}$ is the copula correlation matrix derived from Kendall's tau between model residuals.

This approach preserves both marginal error distributions and cross-model error dependence in simulation-based forecasting.

---

[12] Because the transformed residuals may be in different scales (different Box-Cox λ), include outliers, etc, we adopt the approach of first calculating the dependence using Kendall's tau and then transform to Pearson's correlation.

8 Methods and Functions

Table 1 provides a summary of the methods and functions implemented in the package by input specification.

Table 1: Workflow of ISSM Package Methods by Specification Type

Step	Specification Type	Operation	Resulting Object/Class
Manual Specification (auto = FALSE)			
Model Specification	Manual (auto = FALSE)	`issm_modelspec(auto = FALSE)`	`tsissm.spec`
Estimation	Manual (auto = FALSE)	`estimate()`	`tsissm.estimate`
Backtesting	Manual (auto = FALSE)	`tsbacktest()`	Backtest output
Automatic Specification (top_n = 1)			
Model Specification	Auto (top_n = 1)	`issm_modelspec(auto = TRUE, top_n = 1)`	`tsissm.autospec`
Estimation	Auto (top_n = 1)	`estimate()`	`tsissm.estimate`
Diagnostics & Summary	Auto (top_n = 1)	`summary(), AIC(), BIC(), estfun(), bread(), residuals(), fitted(), tsdecompose(), tsmetrics(), vcov(), sigma(), logLik(), coef(), hresiduals(), tsequation(), tsdiagnose(), plot(), tsmoments(), tsprofile()`	Diagnostic outputs
Filtering	Auto (top_n = 1)	`tsfilter()`	Updated `tsissm.estimate`
Prediction & Simulation	Auto (top_n = 1)	`predict() & simulate()`	`tsissm.predict` and `tsissm.simulate`
Prediction Diagnostics	Auto (top_n = 1)	`tsmetrics(), plot(), tsdecompose()`	Prediction diagnostic outputs
Automatic Specification (top_n > 1)			
Model Specification	Auto (top_n > 1)	`issm_modelspec(auto = TRUE, top_n = 2)`	`tsissm.autospec`
Estimation	Auto (top_n > 1)	`estimate()`	`tsissm.selection`
Filtering	Auto (top_n > 1)	`tsfilter()`	Updated `tsissm.selection`
Prediction & Simulation	Auto (top_n > 1)	`predict() & simulate()`	`tsissm.selection_predict` and `tsissm.selection_simulate`
Backtesting	Auto (top_n > 1)	`tsbacktest()`	Backtest output
Selection Diagnostics	Auto (top_n > 1)	`AIC(), BIC(), logLik(), tsensemble()`	Diagnostic and ensemble outputs

9 Conclusion

The tsissm package makes use of the methods implemented in tsmethods and shared across all packages in the tsmodels framework. It provides an enhanced version of the **tbats** implementation in (Hyndman et al., 2024), based on suggestions in (Hyndman et al., 2008). A number of demo vignettes are provided showcasing the functionality of the package, and longer demos are available at nopredict.com.

Future work may look at regularization of regressors, automatic anomaly handing and revision of the still experimental vector ETS (tsvets) package.

References

B Anderson and J Moore. *Optimal Filtering (Dover Books on Electrical Engineering)*. 2012.

Piet De Jong. The diffuse kalman filter. *The Annals of Statistics*, 19(2):1073–1083, 1991.

Alysha M De Livera, Rob J Hyndman, and Ralph D Snyder. Forecasting time series with complex seasonal patterns using exponential smoothing. *Journal of the American statistical association*, 106(496):1513–1527, 2011.

Rob Hyndman, Anne B Koehler, J Keith Ord, and Ralph D Snyder. *Forecasting with exponential smoothing: the state space approach*. 2008.

Rob Hyndman, George Athanasopoulos, Christoph Bergmeir, Gabriel Caceres, Leanne Chhay, Mitchell O'Hara-Wild, Fotios Petropoulos, Slava Razbash, Earo Wang, and Farah Yasmeen. *forecast: Forecasting functions for time series and linear models*, 2024. URL https://pkg.robjhyndman.com/forecast/. R package version 8.23.0.

Helmut Lütkepohl. *New introduction to multiple time series analysis*. 2005.