UNIVERSITÄT PADERBORN
Die Universität der Informationsgesellschaft

Center For International Economics

Working Paper Series

Working Paper No. 2026-10

Time series forecasting in SAP using a data-driven seasonal semiparametric ARMA model

Li Chen, Yuanhua Feng

June 2026

CIE Center For International Economics

Time series forecasting in SAP using a data-driven seasonal semiparametric ARMA model

Li Chen* Yuanhua Feng†

Abstract

Building upon our previous work that integrated a semi-parametric ARMA model into the SAP ecosystem, this paper introduces an enhanced forecasting application for SAP Analytics Cloud (SAC), termed *deseatsForecast*. The application leverages a data-driven seasonal semiparametric ARMA (S-Semi-ARMA) algorithm and novelly addresses two critical gaps in the practical deployment of advanced semiparametric models within enterprise environments. Specifically, the proposed *deseatsForecast* application enables robust estimation of slowly-changing seasonal patterns jointly with trend components through a data-driven Iterative Plug-In (IPI) algorithm for bandwidth selection. Secondly, the application provides native support for panel data structures, thereby extending its applicability to multidimensional business datasets commonly encountered in enterprise settings. The paper begins with a review of the data-driven S-Semi-ARMA model and the estimation procedures for trend, seasonal, and residual components. Subsequently, forecasting techniques based on the S-Semi-ARMA framework are presented, followed by a brief description of the architecture and design of the *deseatsForecast* application, with particular emphasis on its extensions relative to the *smootsForecast* application. Finally, the forecasting application is empirically validated using OECD passenger car registration data for multiple countries, and a comparative study against SAP's *autoML*-based forecasting approach is conducted. The empirical results demonstrate consistently strong forecast performance of the *deseatsForecast* application and highlight its superior forecast accuracy and transparency compared with the current *autoML* approach in SAP.

Keywords : Time series forecasting, semiparametric algorithm, forecasting accuracy, *deseats*, SAP Analytics Cloud, enterprise adoption

*Corresponding author. E-mail: lichen@campus.uni-paderborn.de. University of Paderborn, Faculty of Business Administration and Economics

†University of Paderborn, Faculty of Business Administration and Economics

1 Introduction

The digitalization of enterprise operations has led to a rapid increase in both the volume and velocity of time series data available for analytical purposes. From demand planning to financial revenue projections, the ability to accurately forecast future business developments has become a critical competitive differentiator. Traditionally, enterprise forecasting has been dominated by parametric modeling approaches. However, recent advances in nonparametric and semiparametric statistics provide compelling alternatives with strong empirical support (Chen and Feng, 2025; Fildes et al., 2022; Aburto and Weber, 2007; Kuo, 2001; Žliobaitė et al., 2012). These methods avoid imposing rigid functional forms on trend and seasonal components and are therefore better suited to adapt to structural breaks and evolving economic conditions (Ghysels et al., 2006). Building on the work of Schulz et al. (2021), forecast intervals for semiparametric models can be constructed using an iterative bootstrap approach.

Despite their methodological advantages, the adoption of such systematic forecasting methods (SFM) in enterprise environments remains limited. A survey by Goodwin et al. (2023) identifies three primary barriers to adoption: limited in-house expertise in advanced statistical and machine learning methods, the high costs associated with implementing specialized software solutions, and a prevailing lack of trust in the perceived "black-box" nature of complex algorithms. Against this backdrop, a business-user-friendly, well-integrated, and cost-efficient forecasting application such as *smootsForecast* (Chen and Feng, 2025) represents a meaningful step toward practical enterprise deployment.

Although the *smootsForecast* application successfully demonstrates the technical feasibility of integration, cost efficiency, and usability, two important limitations remain. First, most business-relevant time series, such as sales or demand data, exhibit pronounced seasonal patterns that require explicit modeling (Box et al., 1978a; Hylleberg, 1992; Ghysels et al., 2006). Second, enterprise data are inherently multidimensional and are typically organized as panel data across products, regions, or business units, whereas many advanced forecasting algorithms are designed primarily for univariate time series.

To address these limitations, the R package *deseats* (Schulz, 2024) is integrated into a SAP Analytics Cloud application named *deseatsForecast*. The package implements a seasonal semiparametric ARMA (S-Semi-ARMA) model that enables the simultaneous estimation of a smooth trend and a slowly evolving seasonal component using locally weighted regression

techniques (Heiler and Feng, 2000; Feng, 2013; Schulz, 2024). Similar to the semi-ARMA framework proposed by Feng et al. (2020); Schulz et al. (2021), the S-Semi-ARMA model incorporates an automatic, data-driven bandwidth selection procedure based on an Iterative Plug-In (IPI) algorithm inspired by Bühlmann (1996). This approach substantially lowers the barrier to model specification by eliminating the need for manual bandwidth selection, which is often a major challenge for business users with limited statistical expertise.

Furthermore, the *deseatsForecast* application extends the original framework by providing native support for panel time series data within its custom R logic. A dynamic encoding and decoding mechanism is implemented to generate forecasts for multiple time series simultaneously and to consolidate the results into a single object that can be written back to the SAP Analytics Cloud application layer (See Section 5.3 and Appendix B). The frontend user interface is redesigned accordingly, for example to support the joint visualization of multiple time series and their corresponding forecasts.

The remainder of this paper is structured as follows. Section 2 reviews the foundational concepts of time series decomposition, locally weighted regression, and trigonometric regression. Sections 3 and 4 introduce the seasonal semiparametric ARMA (S-Semi-ARMA) model and the associated estimation and forecasting methodologies. Section 5 describes the architecture and design of the *deseatsForecast* application, with particular emphasis on the extensions relative to the *smootsForecast* application. An empirical study based on publicly available passenger car registration data for five selected countries is then conducted to evaluate forecasting performance in Section 6. This includes a comparative analysis against SAP's *autoML* approach, demonstrating the superior forecast accuracy of the *deseatsForecast* application. Section 7 concludes.

2 Review of previous results on modeling seasonality in times series

2.1 Time series decomposition and seasonal adjustment

Time series decomposition is a fundamental technique in times series analysis and forecasting predicated on the assumption that a time series y_t can be separated into unobservable components including trend, cycles, seasonality and residuals. While cycles captures fluctuations with not fixed frequency, seasonality describes patterns with a fixed frequency (Hylleberg,

1992; Chatfield, 2003). Due to the fact that most external influencers like natural temperature changes in different seasons or recurring bank holidays are of fixed frequency, this paper ignores the difference of cycles and seasonality. Assuming an additive model, a times series y_t can be modeled as the sum of a trend-cycle g_t, a seasonal component s_t, and an irregular remainder ξ_t.

The history of modeling seasonal component has been evolving from classic univariate linear models, including methods based on SARIMA models, seasonally integrated models and deterministic seasonality models, via periodic models with seasonally varying parameter, to nonlinear nonlinear seasonal models (Ghysels et al., 2006; Mazzi et al., 2018). More sophisticated procedures like X-11 and its successors like X-12-ARIMA and X-13 ARIMA-SEATS, developed by the US Census Bureau, employ iterative moving averages and ARIMA modeling to handle diverse calendar effects and outliers (U.S. Census Bureau, 2017; Findley et al., 1998). Similarly, the Berlin Procedure (BV4.1), used by the German Federal Statistical Office, utilizes fixed filters based on polynomial and trigonometric approximation (Heiler, 1970; Speth, 2004; Cieplik, 2006). Upon this foundation, Feng (1999); Feng et al. (2020); Feng (2013) proposed multiple daten-driven approaches for extracting trend and seasonality using a automatically selected asymptotically optimal bandwidth. Together with a parametric autoregressive moving average (ARMA) component, semiparametric models for eastimating and forecasting time series are constructed. These algorithms are further developed and implemented as public R package *smoots* and *deseats* by Schulz et al. (2021); Schulz (2024); Feng et al. (2020); Feng (2013).

2.2 The locally weighted regression

Most nonparametric methods for decomposing time series utilize local methods like local regression, among which locally weighted regression is one the most useful approaches (Feng, 1999). LWR provides a flexible, nonparametric method for estimating the underlying mean function of a time series without assuming a global parametric form. Unlike global polynomial regression, which fits a single polynomial to the entire dataset, LWR fits a separate polynomial for each target point x_0 using only the data in its local neighborhood.

The fundamental premise of LWR is that the unknown trend function $g(x)$ can be approximated locally by a polynomial of low order r (typically $r = 1$ for local linear or $r = 3$ for local cubic regression) via a Taylor series expansion. For a target time point x_0, the estimator

minimizes a weighted sum of squared residuals $Q(x_0) = \sum_{t=1}^{n} [Y_t - \tilde{g}(x_t) - \tilde{s}(x_t)]^2 W\left(\frac{x_t - x_0}{h}\right)$. Here, $W(\cdot)$ is a kernel weighting function (e.g., Epanechnikov, Bisquare, or Triweight) that assigns maximum weight to observations near x_0 and decreasing weight to those further away. The bandwidth h controls the size of the local neighborhood and effectively determines the smoothness of the estimate (Feng, 1999; Heiler, 1970).

LWR offers several distinct advantages over kernel-based smoothing as described by Feng (2004). First, it possesses automatic boundary correction mechanism proposed and further developed by Fan and Gijbels (1992); Hastie and Loader (1993); Ruppert and Wand (1994). Unlike simple moving averages or standard kernel smoothers which suffer from high bias at the boundaries due to the asymmetry of the data window, local polynomial regression adapts the effective kernel shape at the endpoints. This adaptation maintains the same order of bias convergence at the boundaries as in the interior, a critical feature for forecasting applications where the most recent trend estimate is paramount. Second, LWR allows for the direct estimation of derivatives (Feng and Heiler, 2009; Feng et al., 2020; Chen and Feng, 2025). The coefficients $\hat{\beta}_j(x_0)$ obtained from the minimization problem provide direct estimates of the j-th derivative of the trend, $g^{(j)}(x_0)$. These derivatives are crucial for identifying turning points in economic cycles. For instance, the first derivative indicates the growth rate, while the second derivative can signal acceleration or deceleration of the trend.

However, the efficacy of LWR is strongly dependent on the selection of the bandwidth h. A bandwidth that is too large leads to oversmoothing, obscuring significant features of the trend such as cyclical turning points. Conversely, a bandwidth that is too small results in undersmoothing, causing the trend estimate to track the noise rather than the signal. Thanks the nice properties of LWR estimator, especially its asymptotical equivalence to kernel estimators, data-driven approaches for automatically selecting a optimal bandwidth using the iterative plug-in (IPI) algorihtms are developed as described by Müller (1987); Feng (2004); Feng et al. (2020).

2.3 Trigonometric regression

While local polynomial regression is highly effective for estimating smooth trends, trigonometric regression are often used for capturing seasonality. Many scholars leverge trigonometric functions to estimate the seasonality (Cleveland and Tiao, 1976; Box et al., 1978b; Hillmer and Tiao, 1982). Heiler (1966, 1970) proposed to model seasonal pattern using trigonometric

functions as local regressors, which was later built in the Berlin Procedure.

Instead of projecting the data solely onto a polynomial basis $1, (x - x_0), \ldots, (x - x_0)^r$, a fixed seasonal pattern may be captured by a set of trigonometric terms at the seasonal frequencies. This method is often referred to as local trigonometric regression. The local approximation for the seasonal component $s(x_t)$ at a target point x_0 is given by $s(x_t) \approx \tilde{s}(x_t) = \sum_{i=1}^{q} \{\alpha_{1,i}(x_0) \cos[\lambda_i n(x_t - x_0)] + \alpha_{2,i}(x_0) \sin[\lambda_i n(x_t - x_0)]\}$, where $\lambda_i = 2\pi i / p_s$ are the seasonal frequencies, p_s is the seasonal period, and $q = [p_s/2]$ is the number of harmonics required to fully represent the seasonality. For even periods (e.g., $p_s = 12$), the sine term at the Nyquist frequency is omitted to prevent singularity.

This approach essentially performs a local Fourier analysis. Because the coefficients $\alpha_{1,i}$ and $\alpha_{2,i}$ are estimated locally at each time point x_0, they can vary over time. This allows the amplitude and phase of the seasonal cycle to evolve gradually, providing a flexible representation of slowly changing seasonality as described by Feng (1999); Schulz (2024). This is a more realistic assumption for economic data than fixed seasonal dummies, as seasonal patterns often drift due to structural shifts in the economy.

3 The seasonal semiparametric ARMA model

While the semi-ARMA model (Feng et al., 2020; Schulz et al., 2021) focues more on the extraction of trend component, a seasonal semiparametric ARMA (S-Semi-ARMA) model was proposed to include a seasonal component for time series analyses (Feng, 2013; Schulz, 2024). The S-Semi-ARMA model also represents a hybrid approach, combining the flexibility of nonparametric smoothing with the structural precision of parametric residual modeling.

3.1 Definition of the seasonal semiparametric ARMA model

Following Schulz (2024), we assume an univariate, equidistant time series $y_{t\,t=1}^{n}$ observed at time points $t = 1, \ldots, n$. The model is defined as:

$$y_t = m(x_t) + \xi_t = g(x_t) + s(x_t) + \xi_t \tag{1}$$

Where $x_t = t/n$ represents the rescaled time on the interval. Rescaling allows for asymptotic analysis as $n \to \infty$. $g(x_t)$ is the smooth and deterministic trend function. It is assumed to be k-times continuously differentiable. $s(x_t)$ refers to a slowly changing seasonal component

with period p_s, for example 12 for monthly data. ξ_t is a stationary stochastic error process with $E(\xi_t) = 0$. Furthermore, ξ_t allows for deterministic short-range dependence, meaning the autocovariance $\gamma(\tau) = Cov(\xi_t, \xi_{t+\tau})$ is not zero for all $\tau \neq 0$, but decays sufficiently fast such that the spectral density at frequency zero $c_f = f(0) = (2\pi)^{-1} \sum_{\tau=-\infty}^{\infty} \gamma(\tau) < \infty$.

3.2 Estimation of $g(x_t)$ and $s(x_t)$ via locally weighted regression (LWR)

Obviously, model (1) is an extension of model proposed by Feng et al. (2020) with only one component for trend and seasonality $m(x_t)$, where now the trend and seasonal component are more specifically and seperately estimated. To avoid bias that can occur in estimating trend and seasonality sequentially, it's important to estimate $g(x_t)$ and $s(x_t)$ simultaneously, so that the first step might absorb variance belonging to the second component (Hoffman et al., 2011). Based on ideas and example of Heiler and Feng (2000), $g(x_t)$ and $s(x_t)$ can be estimated using a unified locally weighted least squares optimization as described below.

For a target time point x_0, the local trend component $g(x_t)$ can be approximated by a polynomial of order r as below:

$$g(x_t) \approx \tilde{g}(x_t) = \sum_{j=0}^{r} \beta_j(x_0)(x_t - x_0)^j \tag{2}$$

where typically $r = 1$ for local linear or $r = 3$ for local cubic, $\beta_j(x_0), j = 0, \ldots, r$ are real-valued coefficients in the approximation of $g(x_t)$ in the proximity of x_0.

For a target time point x_0, the local seasonal component $s(x_t)$ can be approximated by a trigonometric polynomial containing $q = [p_s/2]$ harmonic frequencies:

$$s(x_t) \approx \tilde{s}(x_t) = \sum_{i=1}^{q} \{\alpha_{1,i}(x_0) \cos[\lambda_i n(x_t - x_0)] + \alpha_{2,i}(x_0) \sin[\lambda_i n(x_t - x_0)]\} \tag{3}$$

Here, $\lambda_1 = 2\pi/p_s$ is the fundamental frequency, and $\lambda_i = i\lambda_1$ for $i = 2, \ldots, q$, are the harmonics. For even periods (e.g., $p_s = 12$), the sine term at the Nyquist frequency is omitted to avoid singularity (Feng, 2013). Similarly, $\alpha_{1,i}(x_0)$ and $\alpha_{2,i}(x_0)$ for $i = 1, \ldots, q$ are real-valued coefficients for approximating$s(x_t)$ around x_0.

Following the approximation (2) and (3), estimating trend and seasonal component becomes a problem of estimating the coefficients β and α. An obvious approach is to minimize

the locally weighted sum of squared errors $Q(x_0)$ as defined below:

$$Q(x_0) = \sum_{t=1}^{n} \left[Y_t - \tilde{g}(x_t) - \tilde{s}(x_t)\right]^2 W\left(\frac{x_t - x_0}{h}\right) \tag{4}$$

where the weighting function $W(\cdot)$ is a kernel density with compact support $[-1, 1]$. Following the idea of Schulz (2024), weighting functions of the form $W_\mu(u) = C_\mu(1-u^2)^\mu I_{[-1,1]}(u)$ can be used, where μ controls smoothness. For simplicity, $\mu = 0, 1, 2, 3$ representing popular kernel functions are implemented by Schulz (2024) in his algorithm and R package.

Following the example by Feng (2013), the minimization problem can be solved using a design matrix. Let $y = (y_1, \ldots, y_n)'$ be the vector of observations. A design matrix $X = (X_1 \vdots X_2)$ can be crafted with X_1 containing the polynomial regressors for the trend-cyclical component, X_2 being the trigonometric regressors for the seasonal component and their definition like below:

$$X_1 = \begin{pmatrix} 1 & x_1 - x_0 & \cdots & (x_1 - x_0)^r \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_n - x_0 & \cdots & (x_n - x_0)^r \end{pmatrix} \tag{5}$$

and

$$X_2 = \begin{pmatrix} \cos[\lambda_1(1-t_0)] & \sin[\lambda_1(1-t_0)] & \cdots & \cos[\lambda_q(1-t_0)] & \sin[\lambda_q(1-t_0)] \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ \cos[\lambda_1(n-t_0)] & \sin[\lambda_1(n-t_0)] & \cdots & \cos[\lambda_q(n-t_0)] & \sin[\lambda_q(n-t_0)] \end{pmatrix} \tag{6}$$

Let $W = diag(w_1, \ldots, w_n)$ be the diagonal weight matrix where $w_t = W(\frac{x_t - x_0}{h})$. Define the j-th $(r+1) \times 1$ unit vector by $i_{g,j}$ and let i_s be an $(p_s - 1) \times 1$ vector having 1 in its odd entries and 0 elsewhere. The estimator for the combined trend seasonal component $m(x_0)$ can be written as

$$\hat{m}(x_0) = (i'_{g,j}, i'_s)(X'WX)^{-1}X'Wy \tag{7}$$

Partitioning the design matrix reversely, the estimate for the components trend $\hat{g}(x_0)$ and the seasonal component $\hat{s}(x_0)$ can then be extracted as below:

$$\hat{g}(x_0) = (i'_{g,j}, 0')(X'WX)^{-1}X'Wy \tag{8}$$

$$\hat{s}(x_0) = (0', i'_s)(X'WX)^{-1}X'Wy \tag{9}$$

where $i_{g,j}$ selects the j-th trend coefficient and i_s aggregates the seasonal coefficients.

3.3 Estimation of the asymptotically optimized bandwidth h_A

To find out the estimate for trend $\hat{g}(x_0)$ and $\hat{s}(x_0)$, the asymptotic mean integrated squared error (AMISE) for this estimator under short-range dependence can be leveraged following Feng (2013). Similarly, let $k = r + 1$ be the order of the asymptotically equivalent kernel $K_{r,\mu}$, $I[g^{(k)}]$ be the integrated squared k-th derivative of the trend and c_f be the spectral density at frequency zero, the AMISE can then be written as

$$AMISE(h) = h^{2k} \frac{I[g^{(k)}] b_{(k)}^2}{(k!)^2} + \frac{2\pi c_f (d_b - c_b) R(K_{r,\mu}) + (p_s - 1) R(W_\mu)}{nh} \tag{10}$$

with $b_{(k)} = \int_{-1}^{1} u^k K_{r,\mu}(u)\, du$ and $R(K) = \int_{-1}^{1} K^2(u)\, du$. c_b and d_b are boundary constraints between $(0, 1)$ to reduce the boundary effect. To minimize AMISE, we can put the first derivative of it be zero and solve the equation for h to get the asymptotically optimized bandwidth h_A, which can be written as

$$h_A = \left(\frac{(k!)^2}{2k} \frac{2\pi c_f (d_b - c_b) R(K_{r,\mu}) + (p_s - 1) R(W_\mu)}{I[g^{(k)}] b_{(k)}^2} \right)^{\frac{1}{2k+1}} n^{-\frac{1}{2k+1}} \tag{11}$$

This equation makes it clear that the asymptotically optimized bandwidth h_A is dependent from the variance factor c_f and the integrated squared k-th derivative of the trend $I[g^{(k)}]$. Furthermore, it explains mathematically why standard methods assuming an independently identically distributed (*i.i.d.*) white noise process with zero mean and variance σ^2 often fail. Since typically $c_f > \sigma^2$ for economic data, assuming *i.i.d.* errors leads to an underestimation of the variance term, resulting in a theoretically optimal bandwidth that is actually too small.

Following the idea of Feng (2013), the DeSeaTS iterative plug-In (IPI) method similar to Feng et al. (2020) can be employed for bandwidth selection. The IPI algorithm basically solves the "chicken-and-egg" problem by using a pilot bandwidth h_0 for obtaining initial estimates of the trend curvature $I[g^{(k)}]$ and the error factor c_f and subsquently plug these initial estimates into the formula above to update the bandwidth h until convergence or a maximal number of iterations is reached. To be specific, the the asymptotically optimized bandwidth h_A can be selected with following steps:

1. Step 1: Set a starting bandwidth h_0 for example $h_0 = 0.1$.
2. Step 2: Starting the j-th iteration:
   (a) Using bandwidth h_{j-1}, estimate the trend $\hat{g}_j$ and seasonality $\hat{s}_j$.

(b) Calculating residuals using $\hat{\xi}_{t,j} = Y_t - \hat{g}_j(x_t) - \hat{s}_j(x_t)$.

(c) Estimating the variance factor c_f based on calculated residuals $\hat{\xi}_{t,j}$. This needs to apply a data-driven lag-window spectral density estimator to $\hat{\xi}_{t,j}$, which in the fact uses a separate IPI sub-algorithm following the idea of Bühlmann (1996) to select a window width M for the lag-window estimator $\hat{c}_f = (2\pi)^{-1} \sum_{l=-M}^{M} w_l \hat{\gamma}(l)$ with w_l being Bartlett weights. More details for this step can be found in previous works by Feng et al. (2020).

(d) Estimating the curvature $I[g^{(k)}]$ using a pilot bandwidth $h_{pilot} = h_{j-1}^{\nu}$, where ν is an inflation factor, typically 5/7 for local linear regression. More details for this step can be found in previous work by Feng et al. (2020).

(e) Finally, plugging the estimated $\hat{c}_f$ and $\hat{I}[g^{(k)}]$ into the AMISE minimization formula (11) to calculate a new bandwidth h_j.

(f) Increase the iteration counter j by 1.

3. Step 3: Iterations in step 2 should continue until the bandwidth stabilizes ($|h_j - h_{j-1}| < \epsilon$) or a maximum iteration count is reached. The final bandwidth $\hat{h}$ is selected for the decomposition.

As shown in formula (11), boundary problems are corrected in the locally weighted regression. Specificly, this was enhanced by shortening the window or extending the weights, such ensuring that the trend estimates at the most recent data points are robust. Furthermore, the algorithm involves repeated inversion of matrices $(X'WX)$ at every time point for every iteration. For a time series of length n, this implies a computational complexity of $O(n^2)$ per iteration. To mitigate computation issues, the *deseats* package implements the core loops in C++ using Rcpp and RcppArmadillo linear algebra libraries, ensuring that the algorithm is performant enough for interactive use in SAP (Schulz, 2024).

3.4 Estimation of residual component using ARMA model

The approximated residual component can be calculated after extraction of trend and seasonal component using $\tilde{\xi}_t = y_t - \hat{g}(x_t) - \hat{s}(x_t)$, which is often called as the trend- and seasonal adjusted time series. Following the definition in (1) and approaches by Feng et al. (2020) and Schulz (2024), this component can be estimated by assuming an auto regressive moving

average (ARMA) process with order p and q written as below:

$$\xi_t = \phi_1 \xi_{t-1} + \phi_2 \xi_{t-2} + \cdots + \phi_p \xi_{t-p} + \theta_1 u_{t-1} + \theta_2 u_{t-2} + \cdots + \theta_q u_{t-q} + u_t \tag{12}$$

where u_t fulfills the assumptions of an independently identically distributed ($i.i.d.$) white noise process with zero mean and a constant variance of σ_u^2. Furthermore, θ and ϕ are coefficients of the underlying AR and MA processes respectively. For estimating the parametric ARMA model, there are well established approaches like quasi-maximum-likelihood estimation (QMLE), where the optimal parameters are selected by comparing statistical criteria like the Bayesian information criterion (Schwarz, 1978).

4 Forecasting techniques for the seasonal semiparametric ARMA model

After decomposition $y_t = g(x_t) + s(x_t) + \xi_t$ with a automatically selected optimal bandwidth, forecasts can be made by projecting each component forward into the next k forecasting periods. This section describes the specific methods used for forecasting each component.

4.1 Extrapolation of trend

Forecasting the nonparametric trend component requires an assumption about its future evolution. The *deseats* package assumes a linear extrapolation of the local trend estimate at the boundary following Schulz (2024) and Beran and Ocker (1999). Mathematically, the point forecast for the trend component at time point $t = n + k$ can then be written as

$$\hat{g}(x_{n+k}) = \hat{g}(x_n) + k \cdot \hat{g}^{(1)}(x_n) \tag{13}$$

where $\hat{g}^{(1)}(x_n)$ is the estimated first derivative of the trend at the last observation x_n. While simple, this is consistent with the local linearity assumption of the model. For local cubic trends, curvature could theoretically be extrapolated, but linear projection is generally more stable for forecasting (Schulz, 2024) .

4.2 Extrapolation of seasonality

For the seasonal component $s(x_t)$, we assume it to be slowly changing. For forecasting purposes, it is further assumed to be locally periodic. The forecast repeats simply the seasonal

pattern estimated from the most recent full cycle as below:

$$\hat{s}(x_{n+k}) = \hat{s}(x_{n+k-p_s(\delta+1)}) \tag{14}$$

where $\delta = [k/p_s]$ denotes the integral part of k/p_s, which points to the corresponding period of last seasonal cycle in the historical data.

4.3 Forecast of residual component using parameters

According to model defition, the residuals component is modeled as a auto regressive moving average (ARMA) process with order p and q, which follows a parametric design. Consequently, forecasting of this component can simply use the parameters obtained from model fitting. However, it's important to point out here that the possible effect of $\hat{g}(x_{n+k})$ and $\hat{s}(x_{n+k})$ on $\hat{\xi}_{t+k}$ is neglected for simplicity as suggested by Schulz (2024); Feng and Zhou (2015). Following Fritz et al. (2018) and Schulz et al. (2021), an approximately best linear predictions from an ARMA model fitted to $\hat{\xi}$ can be computed by

$$\hat{\xi}_{n+k} = \sum_{i=1}^{p} \hat{\phi}_i \hat{\xi}_{n+k-i} + \sum_{j=1}^{q} \hat{\theta}_j \hat{u}_{n+k-j} \tag{15}$$

where $\hat{\xi}_{n+k-i}$ are forecast values for $k - i > 0$. However, if $k - i \leq 0$, $\hat{\xi}_{n+k-i}$ should be the actual values of the detrended series. The same applies also for the innovation component $\hat{u}_{n+k-j}$ with $E(u_t) = 0$ for $k - i > 0$.

Ultimately, the total point forecast $\hat{y}_{t+k}$ at a future time point $t+k$ can be approximately calculated by aggregating estimation of all three components together as below:

$$\hat{y}_{t+k} = \hat{g}(x_{n+k}) + \hat{s}(x_{n+k}) + \hat{\xi}_{t+k} \tag{16}$$

4.4 Calculation of forecast intervals

Constructing prediction intervals for semiparametric models is complex for the nonparametric components trend and seasonality. However, the error in $\hat{g}(x_{n+k}) + \hat{s}(x_{n+k})$ is proven to be of order $O(n^{-2/5})$, can therefore be omitted for simplicity as proposed by Fritz et al. (2018). This means, the forecast interval of $\hat{y}_{t+k}$ is only dependent on the error the parametric ARMA component $\hat{\xi}_{t+k}$. In analogy to approaches of Chen and Feng (2025), forecast intervals can be obtained under normal error or unknown error. Under assumption of normality, i.e.

$u_t \sim N(0, \sigma_u^2)$ and omission of error in point forecasts of trend and seasonal components, the forecast intervals for point forecast $\hat{y}_{n+k}$ at a confidence level α can be obtained by

$$[\hat{y}_{n+k} - \Phi(1-\alpha^*/2)\tilde{\sigma}_k, \hat{y}_{n+k} + \Phi(1-\alpha^*/2)\tilde{\sigma}_k] \tag{17}$$

with $\alpha^* = 1 - \alpha$ and $\Phi(i)$ is the $(i \times 100)\%$-quantile of the standard normal distribution.

Assuming unknown errors, forecast intervals of a time series ξ_t with n observations can be calculated using a iterative bootstrap procedure as implemented by Schulz et al. (2021); Schulz (2024). Let $s = 1, 2, \ldots, M$ be the number of iterations and M be the maximal number of iterations, the iterative procedure includes specifically steps below:

1. Estimate the coefficients ϕ_i and θ_j of the underlying ARMA process and calculate the point forecasts $\hat{\xi}_{n+1}, \hat{\xi}_{n+2}, \ldots, \hat{\xi}_{n+k}$. Thereafter, calculate the residuals $\hat{u}_t$ and define $\hat{F}_u$ as the empirical distribution of the residual series.

2. Start an iteration counter $s = 1$. Draw a sample set $u_{t,s}^*$ with n observations from $\hat{F}_u$ to form a new series $\xi_{1,s}^*, \xi_{2,s}^*, \ldots, \xi_{n,s}^*$ using sampled residual $u_{t,s}^*$ and estimated coefficients $\hat{\phi}_i$ and $\hat{\theta}_j$ obtained in step 1.

3. Estimate the coefficients $\hat{\phi}_{i,s}^*$ and $\hat{\theta}_{j,s}^*$ of the new simulated series in step 2 and calculate a new residual series $\hat{u}_{t,s}^{**}$ from the original process ξ_t based on these coefficients.

4. Obtain the point forecasts $\hat{\xi}_{n+1,s}^*, \hat{\xi}_{n+2,s}^*, \ldots, \hat{\xi}_{n+k,s}^*$ using Formula **??** based on ξ_t, $\hat{u}_{t,s}^{**}$, $\hat{\phi}_{i,s}^*$ and $\hat{\theta}_{j,s}^*$.

5. Calculate the future bootstrap observations $\xi_{n+k,s}^*$ based on $\xi_t, \hat{u}_t, \hat{\phi}_i, \hat{\theta}_j$ and H bootstrapped future innovations $u_{t,s}^*$.

6. Calculate the forecasting error by $\xi_{n+k,s}^* - \hat{\xi}_{n+k,s}^*$ for each $k = 1, 2, \ldots, H$.

7. Increase s by 1 and repeat step 2 to 6 M times and obtain the empirical distribution of $\xi_{n+k,s}^* - \hat{\xi}_{n+k,s}^*$ for each k with the respective $(\alpha^*/2)100\%$-quantiles, denoted as $q_k(\alpha^*/2)$.

The forecasting intervals can then be described as

$$[\hat{\xi}_{n+k} + q_k(\alpha^*/2), \hat{\xi}_{n+k} - q_k(\alpha^*/2)] \tag{18}$$

Under omission of the error in $\hat{g}(x_{n+k}) + \hat{s}(x_{n+k})$, the final forecasting intervals for point forecast $\hat{y}_{n+k}$ at a confidence level α can be given by

$$[\hat{y}_{n+k} + q_k(\alpha^*/2), \hat{y}_{n+k} - q_k(\alpha^*/2)] \tag{19}$$

4.5 Evaluation of forecast accuracy

In context forecast, a natural inevitable topic is evaluating forecast quality. According to summarization and suggestion by Tashman (2000), the method with spliting dataset into train and test dataset is convenient and reliable. Based on the point forecasts and the real values within the test data set, it's possible to calculate some statistical measures. Following the idea by Chen and Feng (2025), mean absolute scaled error (MASE), root mean squared scaled error (RMSSE) and mean absolute percentage error (MAPE) are used in this paper. MASE is a scale-independent metric that compares the forecast error to the error of a naïve random walk forecast. It is ideal for comparing accuracy across different products or time series with different magnitudes like in our case for panel data with different products or regions. RMSSE is similar to MASE, but penalizes large errors more heavily. MAPE is calculated and displayed because it remains the language of business forecasting in the SAP ecosystem. Providing MAPE ensures that business users can easily interpret the results and comparing them with other SAP applications.

5 The *deseatsForecast* application in SAP Analytics Cloud

In the enterprise context, forecasting time series, such as sales quantities across various markets or regions, is a fundamental requirement for effective financial planning. To organize these recurring planning and forecasting processes efficiently, organizations employ enterprise software solutions like SAP Analytics Cloud (SAC) to ensure consistency and scalability. Following the initial architectural concept proposed by Chen and Feng (2025) with the *smootsForecast* application, this section focuses on the practical integration of the seasonal semiparametric ARMA model (S-Semi-ARMA) into a forecasting application named *deseatsForecast*. Furthermore, recognizing that enterprise data typically exists as multi-dimensional panel data rather than isolated univariate time series, we propose a specific approach for handling panel data structures to facilitate real-world adoption.

5.1 The *deseats* package with seasonal semiparametric ARMA model

The seasonal semiparametric model, as described in Section 3 and Section 4, has been implemented in the R package *deseats* by Schulz (2024). This package serves as the computational core for the integration into SAP Analytics Cloud. For the development of an enterprise-grade

forecasting application, the following functions within the *deseats* package are of particular importance:

- *read_ts*: This function facilitates the reading and conversion of raw time series data into compatible time series objects of class *ts*, ensuring the data is correctly structured for subsequent processing steps. Without this class type, time series properties like seasonality might be ignored due to missing frequency information.

- *deseats*: As one of the core algorithms, this function performs the decomposition of the time series into trend, seasonal, and residual components using locally weighted regression with a automatically selected optimal bandwidth.

- *plot*: This function provides graphical representations of the decomposition results, allowing the estimated trend and seasonal components alongside the original data to be visualized on a same plot. It's used for initial quick check for seasonality in the *deseatsForecast* application.

- *s_semiarma*: This function fits the seasonalsemiparametric ARMA (S-Semi-ARMA) model to a univariate time series. It executes the iterative plug-in (IPI) algorithm to estimate the nonparametric components while simultaneously modeling the error structure.

- *predict*: This function generates both point forecasts and forecast intervals (confidence bands) based on the fitted S-Semi-ARMA models, extending the trend and seasonal components into the future as described in section 4. This funtion is crucial for the *deseatsForecast* application.

- *measures*: To ensure reliability, this function calculates statistical quality measures, such as the mean absolute scaled error (MASE) or root mean squared scaled error (RMSSE), which are written back to SAC for users' assessment of forecast accuracy. This builds the fundament for transparency of forecast quality.

Furthermore, many other functions are provided in the *deseats* package, which can be used for different purposes.

5.2 Architecture and design of *deseatsForecast* application

Following the industry standard for enterprise architecture TOGAF by The Open Group (2022) and the conceptual foundation of *smootsForecast* application laid by Chen and Feng (2025), the solution architecture for *deseatsForecast* is designed across four primary domains to ensure robustness and alignment with business requirements:

Business architecture: Such a *deseatsForecast* application is often used as starting point for integrated financial planning in enterprises, in which a huge number of users and following steps are dependent on the forecast results. Therefore, it's critical to ensure an automated, high-quality forecasting which saves leading time and necessitates minimal statistical intervention from end users at the same time. The *deseatsForecast* keeps the *simplemode* and *advancedmode* design, providing users flexibility for complicated use cases and at the same time benefits from the data-driven bandwidth selection of S-Semi-ARMA algorithm as much as possible. A further key requirement in the enterprise environment is the capability to parallelize forecasting process for multi-dimensional data (panel data) representing forecasts across diverse products and market segments.

Data architecture: Consistent with the *smootsForecast* application described by Chen and Feng (2025), the data architecture utilizes SAP HANA Cloud underlying the planning model in SAC as the single source of truth. This centralized repository stores historical training data, generated point forecasts, and prediction intervals. To manage panel data effectively, multi-dimensional records are aggregated via a unique identifier within the account dimension of the SAC planning model. This dimension is enriched with semantic attributes like region, product and so on, facilitating the logical disaggregation of point forecasts subsequent to the generation phase. To store statistical measures like MASE, MAPE, RMSSE values, this values are defined as an array stored as attributes of account dimension.

Application architecture: Similar to the *smootsForecast* application, the user interface is constructed as an SAP Analytics Cloud Story, leveraging standard widgets such as charts and tables for data interaction and visualization. The application features a dual-mode UI to accommodate varying levels of user expertise. The *simplemode* requires only minimal input for forecast horizon and desired confidence level from end users, while the *advancedmode* exposes more parameters of the algorithm like the polynomial order r, kernel function type, and ARMA constraints, allowing expert users to flexibly analyse deeper in some use cases.

However, the *deseatsForecast* application introduces several extensions with respect to

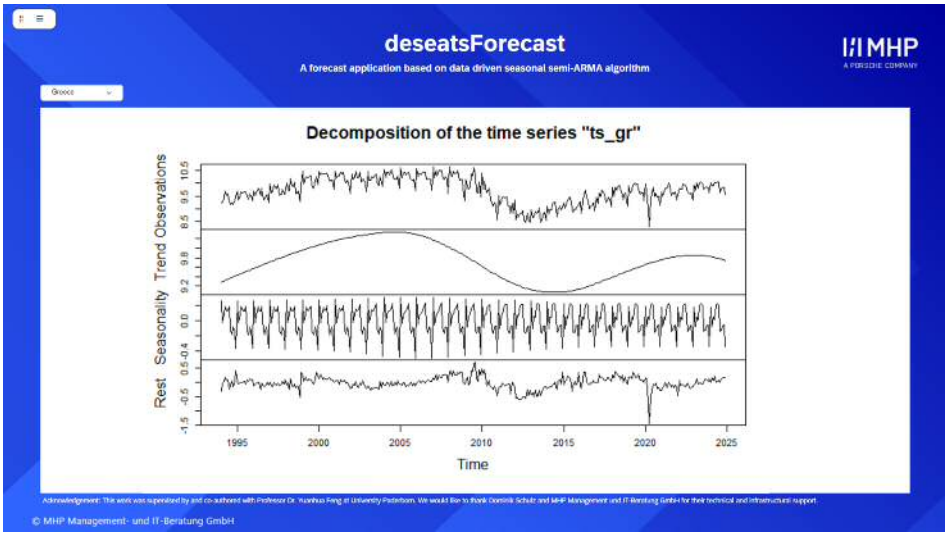


Figure 1: Page Seasonality Check

user interaction. First, an additional application page for seasonality check is implemented. On this page, users can select a single available time series and decompose it using the decomposition algorithm provided by the *deseats* package. The custom R code on R server side for decomposition of selected time series is attached as Appendix A. The corresponding results are displayed directly within the SAP application, as illustrated in Figure 1.

Another major difference compared to the *smootsForecast* application concerns the panel for forecast evaluation. To provide business users with immediate visual feedback on overall forecast reliability, a traffic light system is implemented as an overall quality indicator, similar to the approach used in the *smootsForecast* application. However, since each individual time series within the panel data is associated with its own performance indicators, such as MASE, MAPE, and RMSSE, which may vary substantially across series, the logic of the overall quality indicator has been redesigned to operate at the panel level.

Specifically, the overall forecast quality is classified as *green*, if at least 85% of all individual time series achieve a MASE value smaller than 1, indicating superior performance relative to a naïve benchmark forecast. The indicator is set to *yellow* if at least 50%, but less than 85%, of the time series obtain a MASE smaller than 1. A *red* traffic light indicates that fewer than 50% of the time series within the panel achieve a MASE below 1. In this case, the forecast quality is considered inadequate for the majority of the time series, thereby motivating a more detailed investigation of data quality and model assumptions.

Furthermore, since each time series is now associated with its own performance indicators (MASE, MAPE, RMSSE), the value display on the right-hand side of the application is

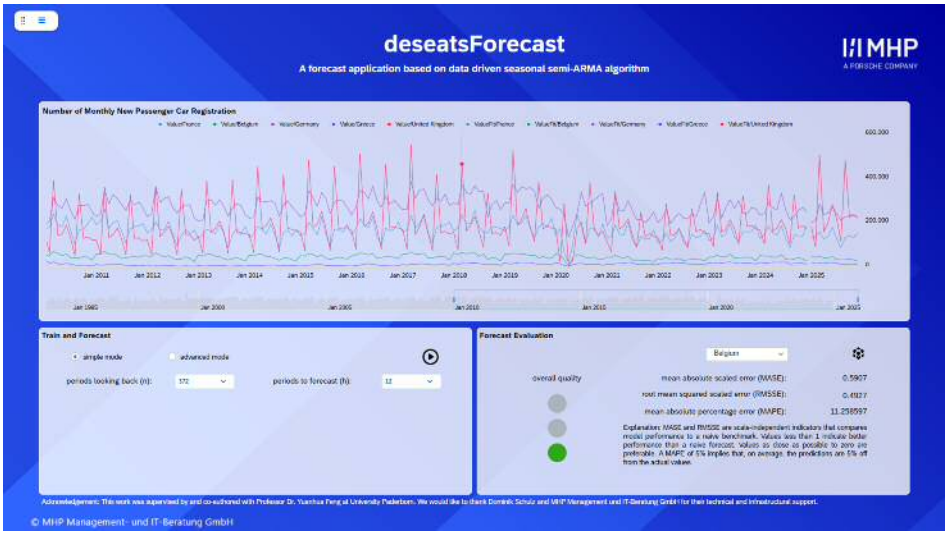


Figure 2: Page Seasonality Check

extended by a dropdown menu that allows users to select a specific time series. The corresponding performance indicators for the selected series are then displayed, as shown in Figure 2.

Technology architecture: As illustrated on technical architecture 3, the SAC environment is connected to an external R server environment via the secure SAP R Connector. The runtime environment is provisioned with the necessary libraries, including *deseats* for the core algorithm, *Rcpp* for performance optimization, and other dependencies. Upon execution, raw data are extracted from the SAC Planning Model via the R Connector to the R server, where they are transformed and fed into for example the *s_semiarma* and *predict* function from the *deseats* package. Results for the multiple time series (panel data) including point forecasts, forecast intervals and statistical measures are concatenated into one single string which is written back to the SAC for further decoding and write-back into the SAP HANA Cloud underlying the planning model in SAC.

5.3 Support for panel data

A critical technical constraint identified during the integration process involves the data transfer mechanism of the SAP standard R-widget. This component, acting as the bridge to the R engine, is restricted to passing data back to SAP solely as single numeric values or character strings. This limitation hinders the direct write-back of complex vector objects, such as multiple point forecasts, confidence intervals, and statistical quality measures, which are essential for panel data applications.

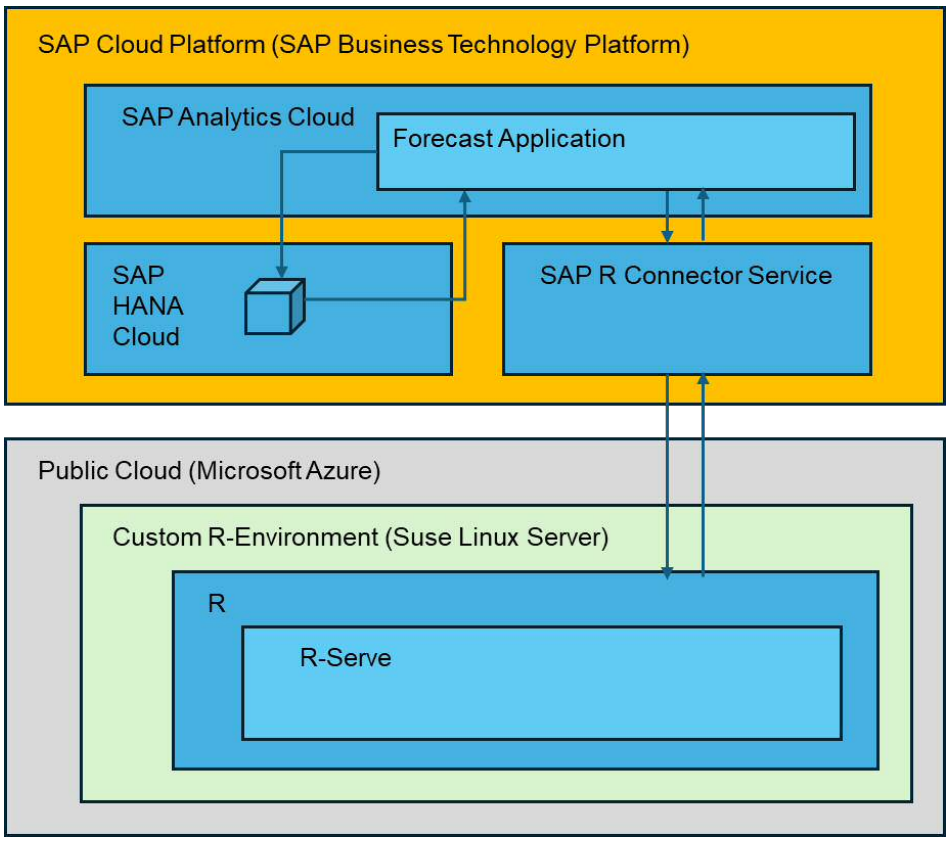


Figure 3: Architecture of forecast application

To resolve this challenge, a technical workaround with a encoding and decoding step is employed. Multiple numeric return values are converted and concatenated into a single, long character string using semantic separators within the R environment. Upon return to the SAP layer, this string is parsed and transformed back into discrete values, which are then mapped and written to the appropriate fields in the SAC data model. As the number of time series within the panel data can change, a dynamic handling of multiple time series in R algorithm, the encoding and decoding step is necessary. For this, the custom R code on R server is designed to be able to process varying number of time series as shown in Appendix B.

While this approach introduces encoding and decoding overhead, it enables a lightweight, cost-efficient forecasting application that operates entirely within the standard SAC infrastructure. Alternative solutions, such as establishing an external API server or utilizing an intermediate application layer like SAP Datasphere, would resolve the data transfer limitations more elegantly but would incur significantly higher implementation and maintenance costs.

6 An empirical study including a comparative study

In this section, the *deseatsForecast* application is applied it to a real world data set for passenger car registrations from OECD (https://data-explorer.oecd.org/). This dataset is not directly from a specific enterprise, but it's to some extend an aggregated sales of new passenger cars per country, which implicitly contains patterns related to macroeconomic factors or market trends, seasonality and market noises. This makes it an ideal candidate for testing the S-Semi-ARMA algorithm.

6.1 Data selection

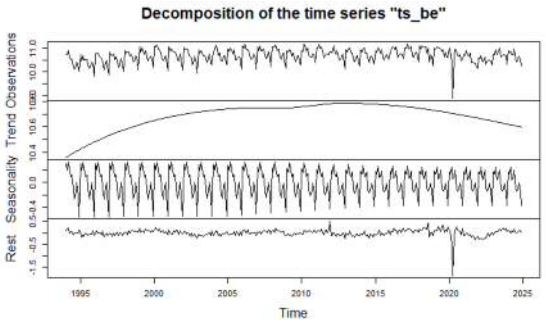
For this practical application, the data set "First registrations of brand new vehicles" from OECD (https://data-explorer.oecd.org/) with filter on passenger cars and time period from January 1994 to December 2024 is selected. As some data are missing during the whole time range, further filters are set on countries. To be specific, five countries Belgium, France, Germany, Greece and UK are selected, in order to demonstrate the forecast on panel data.

Furthermore, initial visual inspections are conducted on the selected data via plots to further validate the dataset's fit for research purpose. Sharp seasonal peaks are typically observed in March and September, which might be driven by license plate release cycles in markets like the UK. In August, the registration figures are mostly at a clearly lower level, which could be related to typical vacation periods. It's noticed that the amplitude of these seasonal swings is neither constant over time nor identical for all markets.

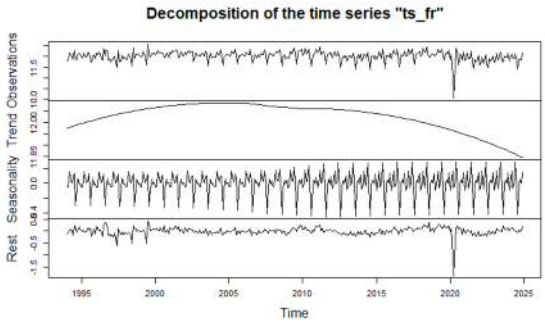
As for trend, most of the series shows non-linear trend behavior over time. For example, a clear decreasing trend is observed for the country Greece between 2010 und 2013, which might be related to the known economic recession in greece caused by european debt crisis (See Figure 1).

6.2 Visualization of decomposition in SAP

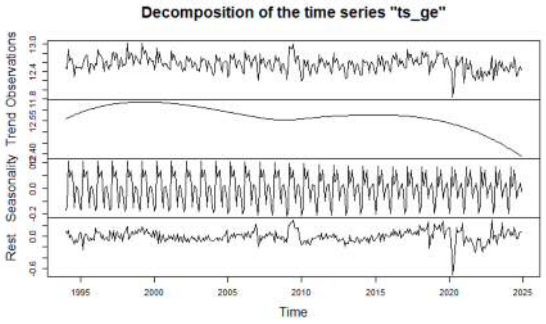
After data selection, the selected data are uploaded into the data model of *deseatsForecast* application, so that they can be decomposed and visualized on the page "Seasonality Check". In background, the selected single time series from the panel data, will be decomposed with local cubic trend ($r = 3$) and frequency of 12 months.



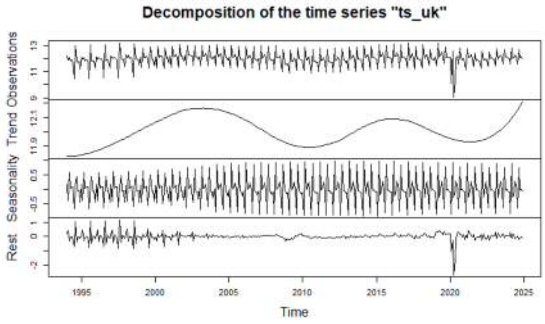
(a) Belgium



(b) France



(c) Germany



(d) UK

Figure 4: Decomposition results

Figure 1 and Figure 4 present the decomposition results for the passenger car registration time series of the five selected countries. All five time series in this panel dataset exhibit a clear and pronounced seasonal pattern. Notably, the four mainland European countries Belgium, France, Germany, and Greece share a broadly similar seasonal structure, although minor differences in the magnitude of the seasonal effects can be observed.

For these countries, passenger car registrations typically increase towards a peak around March, followed by a decline to a lower, yet still elevated, level around August. In October and November, registration figures rise again before decreasing sharply in December and January. In contrast, the United Kingdom exhibits a distinct seasonal pattern, characterized by pronounced peaks in March and September, which is probably driven by the biannual vehicle registration plate change system, which creates strong incentives for both private and corporate buyers to concentrate purchases around these months.

Another important observation is that the seasonal impact in all regions evolves gradually over time, which is consistent with the underlying assumption of the seasonal semiparametric ARMA model. Changes are evident not only in the intra-annual seasonal pattern but also in the overall strength of the seasonal component. As illustrated in Figure 1 and Figure 4,

the seasonal influence in Belgium, Germany, and Greece has weakened over the past decade compared to earlier periods. By contrast, the seasonal component appears to have become more pronounced in France and the United Kingdom.

6.3 Forecasting results in *deseatsForecast* application

After assessing the seasonal patterns, point forecasts together with forecast intervals for the period from January 2025 to December 2025 ($h = 12$) are generated using the *deseatsForecast* application in simple mode. All 372 observations from January 1994 to December 2024 are used for model estimation. Once the forecasting results are transferred to the application, the point forecasts are displayed alongside the historical observations in the time series chart at the top of the user interface. In addition, the numerical point forecasts for the year 2025 are shown on the right-hand side, as illustrated in Figure 2.

Table 1: Performance measures for n=372, h=12

	Belgium	France	Germany	Greece	UK
MASE	○ 0.59	● 0.42	○ 0.54	○ 0.76	● 0.25
MAPE	● 11.26	● 8.40	● 7.64	● 15.90	● 13.10

Table 2: Performance measures for n=372, h=6

	Belgium	France	Germany	Greece	UK
MASE	● 0.28	● 0.27	● 0.47	● 0.48	● 0.33
MAPE	● 6.36	● 5.30	● 7.30	● 11.11	● 17.29

The overall forecast quality indicator is classified as green, as the statistical accuracy measure MASE is smaller than 1 for all five countries. More specifically, Table 1 reports the forecast accuracy measures MASE and MAPE for all countries. A green indicator corresponds to a MASE smaller than 0.5 or a MAPE below 15%, a yellow indicator denotes a MASE between 0.5 and 1, and a red indicator indicates a MASE greater than 1 or a MAPE exceeding 15%. Overall, the results demonstrate strong forecast performance across almost all metrics. The only minor exception is the MAPE for Greece, which is slightly above the 15% threshold.

In addition, forecasts are generated for a shorter horizon of six periods ahead ($h = 6$). The overall quality indicator remains green, and the forecast accuracy measures improve further, as shown in Table 2. An exception is observed for the United Kingdom, where both MASE and MAPE deteriorate slightly compared to the twelve-month forecast horizon. This

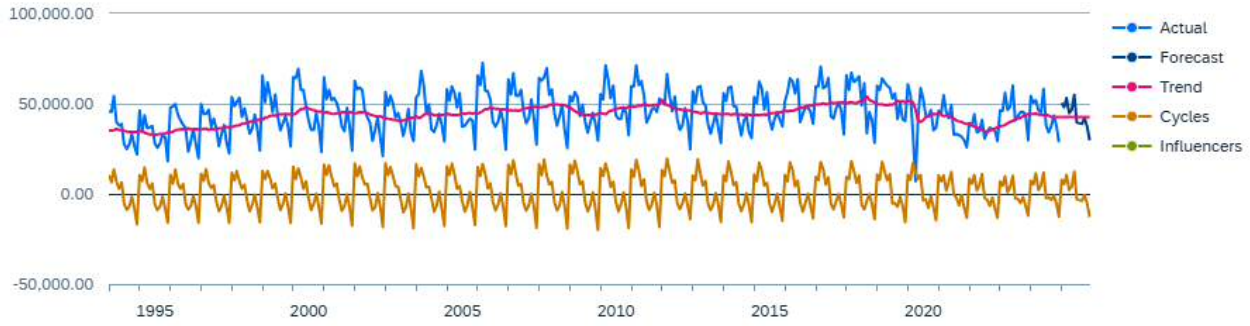


Figure 5: autoML forecast for Belgium in SAC

effect may be related to the exceptionally strong seasonal peak in September, which is not fully covered within the shorter forecast horizon.

6.4 Performance comparison between *deseatsForecast* application and *autoML* in SAP

As mentioned in the introduction, several forecasting algorithms are already available within the SAP ecosystem. In SAP Analytics Cloud (SAC), the built-in Predictive Scenarios feature enables time series forecasting based on an *autoML* approach. This framework decomposes each time series using multiple widely adopted methods, such as ARIMA and exponential smoothing, and subsequently selects the model with the best empirical performance.

To further validate the performance of the S-Semi-ARMA model, point forecasts are generated using the same panel dataset within the SAC Predictive Scenarios framework. For all five countries, the time series are decomposed into trend, annual seasonal component, and residuals, as illustrated in Figure 5 to 9. Compared with the smooth trend extracted by the S-Semi-ARMA model, the SAC forecasts yield more heterogeneous trend patterns. Specifically, non-linear trends are identified for Belgium, France, Germany, and Greece, whereas a linear trend is extracted for the United Kingdom. Moreover, the magnitude of the non-linear trends produced by SAC is substantially larger than that of the corresponding S-Semi-ARMA trends, suggesting potential overfitting.

Moreover, the predctive scenario feature in SAP Analytics Cloud (SAC) provides several statistical measures for assessing forecast accuracy. Using the same panel dataset, forecast performance measures are computed for horizons of $h = 12$ and $h = 6$ based on the SAC *autoML* models. The corresponding results are reported in Table 3 and Table 4 respectively.

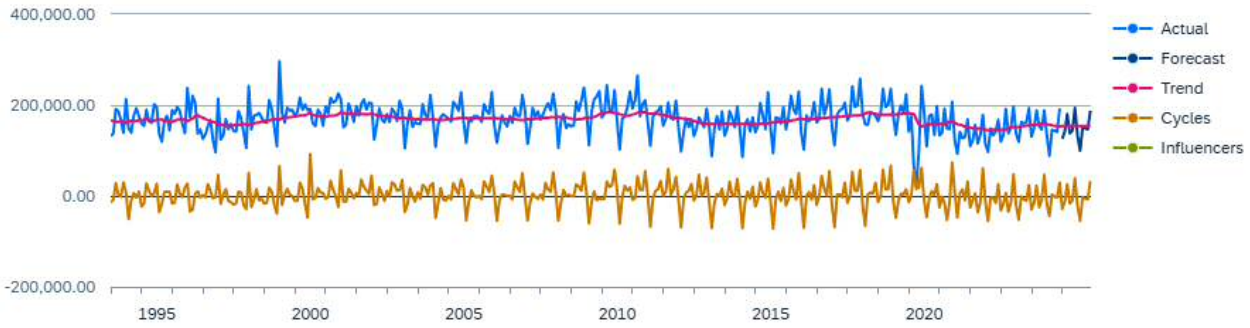


Figure 6: autoML forecast for France in SAC

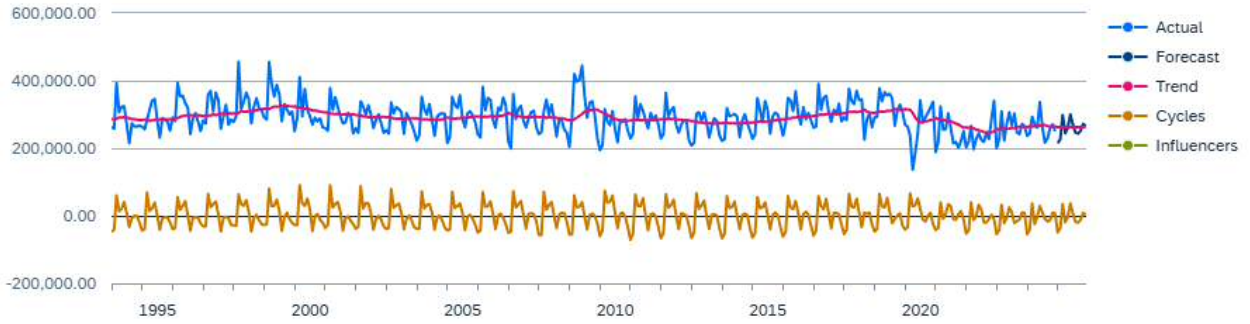


Figure 7: autoML forecast for Germany in SAC

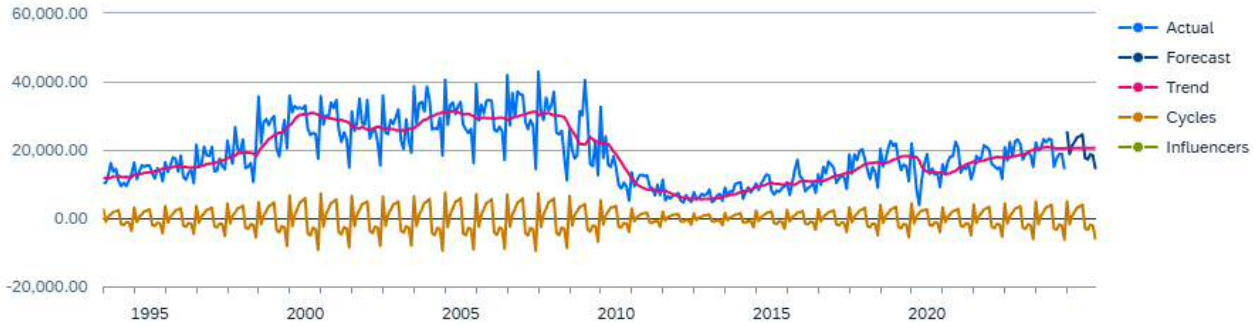


Figure 8: autoML forecast for Greece in SAC

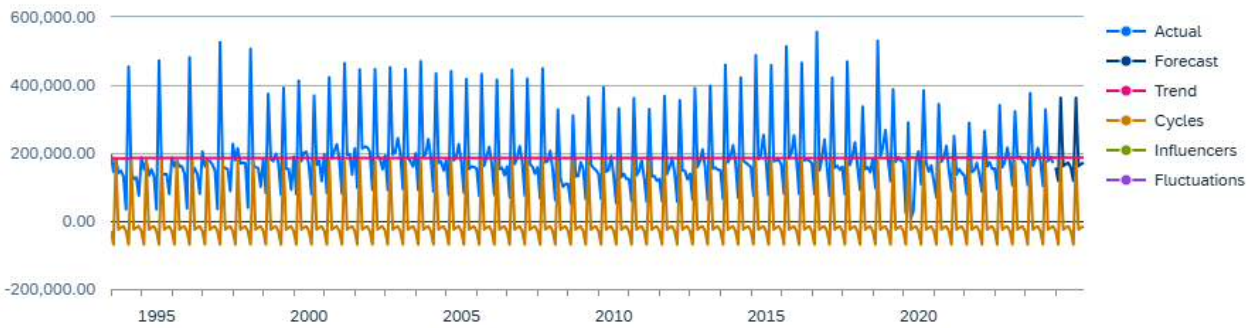


Figure 9: autoML forecast for UK in SAC

Table 3: autoML performance measures in SAC h=12

	Belgium	France	Germany	Greece	UK
Expected MASE	0.70	0.61	0.83	0.51	0.25
Expected MAPE	21.34	21.31	13.00	17.47	43.61

Table 4: autoML performance measures in SAC h=6

	Belgium	France	Germany	Greece	UK
Expected MASE	0.66	0.56	0.81	0.44	0.25
Expected MAPE	20.56	20.37	12.53	15.12	43.61

It should be noted that SAC reports so-called "Expected MASE" and "Expected MAPE" instead of accuracy measures derived from a simple train–test split. For instance, the Expected MAPE evaluates the average forecast error over the entire forecast horizon. For each observed value, the predictive model produces h forecasts, where h denotes the forecast horizon. Each forecasted value is compared with the corresponding realized observation, yielding a horizon-specific performance indicator. The expected performance measure is then computed as the mean of these per-horizon indicators. For simplicity, the Expected MASE and Expected MAPE reported by SAC are directly compared with the corresponding MASE and MAPE values obtained from the S-Semi-ARMA model.

The comparison reveals that, in most cases, the S-Semi-ARMA model outperforms the $autoML$-based forecasts generated by the SAC Predictive Scenarios, as reflected by consistently lower MASE and MAPE values. Minor exceptions are observed for Greece and the United Kingdom, which are likely attributable to pronounced trend changes and potential overfitting in the Greek series, as well as to extreme seasonal peaks in the UK time series.

Another advantage of the S-Semi-ARMA model over the $autoML$ approach lies in the methodological consistency it offers. In an enterprise context, such consistency is particularly beneficial for fostering trust and enhancing the explainability of forecasting results for non-technical business users. In contrast, generating sales forecasts for different regions using heterogeneous forecasting models can be difficult to explain and may appear confusing to end users. Moreover, when forecasts generated by different models are aggregated to higher hierarchical levels, such as from country to continental aggregates, interpretability may be substantially reduced. This loss of explainability arises from the combination of fundamentally different modeling approaches, each potentially relying on distinct assumptions and estimation principles.

7 Concluding remarks

This paper presents the design, implementation, and empirical validation of the *deseatsForecast* application, an enterprise-ready forecasting solution integrated into SAP Analytics Cloud (SAC) and based on a seasonal semiparametric ARMA (S-Semi-ARMA) model. Building upon earlier work on the integration of semiparametric ARMA models into enterprise systems, this study addresses two key limitations commonly encountered in real-world business forecasting: the explicit modeling of evolving seasonality and the systematic handling of panel time series data.

From a methodological perspective, the paper provides a comprehensive description of the seasonal semiparametric ARMA model, including the decomposition of time series into trend, seasonal, and residual components, as well as the associated forecasting procedures. A central advantage lies in the use of a data-driven Iterative Plug-In (IPI) algorithm for bandwidth selection, which eliminates the need for manual tuning of smoothing parameters. This feature substantially lowers the barrier to applying advanced semiparametric methods in practice and supports the broader goal of democratizing statistically sound forecasting techniques for non-expert business users.

On the application level, the *deseatsForecast* application demonstrates how advanced statistical models can be operationalized within a large-scale enterprise analytics platform. By extending the earlier *smootsForecast* application to include seasonality and native support for panel data, the proposed solution is substantially closer to real-world enterprise use cases, such as regional or product-level sales forecasting. The redesigned user interface, including the traffic-light-based quality indicators, provides immediate and actionable feedback on forecast reliability without requiring users to interpret complex statistical diagnostics. This design choice enhances transparency, trust, and usability which are critical for adoption in business environments.

An empirical study using OECD passenger car registration data for multiple countries confirms the strong forecast performance of the *deseatsForecast* application. In comparison with SAP's built-in *autoML*-based predictive scenarios, the S-Semi-ARMA approach consistently delivers competitive or superior forecast accuracy while maintaining methodological consistency across panel units. This consistency is particularly important in enterprise contexts, where forecasts are often aggregated across regions or business units and must remain interpretable at higher hierarchical levels.

Despite these advantages, several limitations remain. The lightweight architectural design based on SAC's standard *RVisualization* widget imposes technical constraints, particularly with respect to writing back forecast results into the application layer and supporting more complex interaction patterns. Moreover, although the IPI-based bandwidth selection significantly reduces the need for expert intervention, domain knowledge and statistical expertise may still be required in advanced configurations or in cases where forecast results are difficult to interpret.

Looking ahead, an important avenue for future research is the development of a more generalized and extensible forecasting framework that allows different algorithms to be integrated in a principled and transparent manner. While *autoML* approaches offer algorithmic flexibility, they often suffer from inconsistency and limited explainability in panel data settings, where different models may be selected for different units. This raises fundamental challenges for interpretation and aggregation of forecasts at higher organizational levels. Motivated by recent advances in AI agents, future work could explore the design of an intelligent forecasting agent that emulates the workflow of a human data scientist from assessing data quality and distributional properties, identifying seasonal and structural characteristics, selecting a unified forecasting approach for panel data, validating forecast performance to finally integrating results into enterprise analytics platforms.

Acknowledgments

This work was supervised by and co-authored with Professor Dr. Yuanhua Feng at University Paderborn. I am very grateful for his valuable guidance and support throughout this research. I also would like to thank Dominik Schulz at University Paderborn and MHP Management- und IT-Beratung GmbH for their technical and infrastructural support. Furthermore, I thank my family and friends in Paderborn for their mental and physical support during my research. During the preparation of this work, the author used ChatGPT to check for grammatical errors and refine some texts. The author reviewed and edited the texts thereafter and takes full responsibility for the content.

References

Luis Aburto and Richard Weber. Improved supply chain management based on hybrid demand forecasts. *Applied Soft Computing*, 7(1):136–144, 2007.

Jan Beran and Dirk Ocker. Semifar forecasts, with applications to foreign exchange rates. *Journal of Statistical Planning and Inference*, 80(1):137–153, 1999. ISSN 0378-3758. doi: https://doi.org/10.1016/S0378-3758(98)00247-X. URL https://www.sciencedirect.com/science/article/pii/S037837589800247X.

George E. P. Box, Steven C. Hillmer, and George C. Tiao. Analysis and modeling of seasonal time series. In Arnold Zellner, editor, *Seasonal Analysis of Economic Time Series*, pages 309–344. National Bureau of Economic Research, 1978a. URL http://www.nber.org/chapters/c4329.

George E. P. Box, Steven C. Hillmer, and George C. Tiao. Analysis and modeling of seasonal time series. In Arnold Zellner, editor, *Seasonal Analysis of Economic Time Series*, number ER-1 in Economic Research Report, pages 309–334. U.S. Bureau of the Census, 1978b.

P. Bühlmann. Locally adaptive lag-window spectral estimation. *Journal of Time Series Analysis*, 17(3):247–270, 1996.

Chris Chatfield. *The Analysis of Time Series - An Introduction*. Chapman and Hall/CRC, 2003.

Li Chen and Yuanhua Feng. Forecasting of trend stationary time series in sap using a data-driven semiparametric arma model. Working paper, Paderborn University, 2025.

Uwe Cieplik. *BV4.1 — Methodology and User-friendly Software for Decomposing Economic Time Series*. German Federal Statistical Office, 2006. URL https://ec.europa.eu/eurostat/documents/3888793/5841609/KS-DT-06-012-EN.PDF/8d4340d2-de7e-45f8-b1e9-c95a0ca1ee4b.

William S. Cleveland and George C. Tiao. Decomposition of seasonal time series: A model for the census X-11 program. *Journal of the American Statistical Association*, 71(355): 581–587, 1976. doi: 10.1080/01621459.1976.10480940.

J. Fan and I. Gijbels. Variable bandwidth and local linear regression smoothers. *The Annals of Statistics*, pages 2008–2036, 1992.

Y. Feng. *Non- and Semiparametric Regression With Fractional Time Series Errors: Theory and Applications to Financial Data*. PhD thesis, University of Konstanz, Germany, 2004.

Y. Feng and S. Heiler. A simple bootstrap bandwidth selector for local polynomial fitting. *Journal of Statistical Computation and Simulation*, 79(12):1425–1439, 2009.

Y. Feng and C. Zhou. Forecasting financial market activity using a semiparametric fractionally integrated log-acd. *International Journal of Forecasting*, 31(2):349–363, 2015.

Y. Feng, T. Gries, and M. Fritz. Data-driven local polynomial for the trend and its derivatives in economic time series. *Journal of Nonparametric Statistics*, 32(2):510–533, 2020.

Yuanhua Feng. *Kernel- and Locally Weighted Regression – with Application to Time Series Decomposition*. Verlag für Wissenschaft und Forschung, 1999.

Yuanhua Feng. An iterative plug-in algorithm for decomposing seasonal time series using the berlin method. *Journal of Applied Statistics*, 40(2):266–281, February 2013. doi: 10.1080/02664763.2012.740626. URL https://ideas.repec.org/a/taf/japsta/v40y2013i2p266-281.html.

Robert Fildes, Shaohui Ma, and Stephan Kolassa. Retail forecasting: Research and practice. *International Journal of Forecasting*, 38(4):1283–1318, 2022.

David F. Findley, Brian C. Monsell, William R. Bell, Mark C. Otto, and Bor-Chung Chen. New capabilities and methods of the X-12-ARIMA seasonal-adjustment program. *Journal of Business & Economic Statistics*, 16(2):127–152, 1998. doi: 10.1080/07350015.1998.10524743.

M. Fritz, S. Forstinger, Y. Feng, and T. Gries. Forecasting economic growth processes for developing economies. Preprint, Paderborn University, 2018.

Eric Ghysels, Denise R. Osborn, and Paulo M.M. Rodrigues. Chapter 13 forecasting seasonal time series. volume 1 of *Handbook of Economic Forecasting*, pages 659–711. Elsevier, 2006. doi: https://doi.org/10.1016/S1574-0706(05)01013-X. URL https://www.sciencedirect.com/science/article/pii/S157407060501013X.

P. Goodwin, J. Hoover, S. Makridakis, F. Petropoulos, and L. Tashman. Business forecasting methods: Impressive advances, lagging implementation. *PLoS One*, 18:e0295693, 12 2023.

T. Hastie and C. Loader. Local regression: Automatic kernel carpentry. *Statistical Science*, 8(2):120–129, 1993.

S. Heiler. *Analyse der Struktur wirtschaftlicher Prozesse durch Zerlegung von Zeitreihen*. Dissertation, Universität Tübingen, Tübingen, 1966.

S. Heiler. Theoretische grundlagen des "berliner verfahrens". In Wolfgang Wetzel, editor, *Neuere Entwicklungen Auf Dem Gebiet Der Zeitreihenanalyse*, volume 1 of *Sonderheft*, pages 67–93. Statistisches Bundesamt / Verlag, 1970.

Siegfried Heiler and Yuanhua Feng. Data-driven decomposition of seasonal time series. *Journal of Statistical Planning and Inference*, 91(2):351–363, 2000. doi: 10.1016/S0378-3758(00)00187-7.

Steven C. Hillmer and George C. Tiao. An ARIMA-based approach to seasonal adjustment. *Journal of the American Statistical Association*, 77(377):63–70, 1982. doi: 10.1080/01621459.1982.10477767.

Ross M. Hoffman, John H. Kagel, and Dan Levin. Simultaneous versus sequential information processing. *Economics Letters*, 112(1):16–18, 2011. ISSN 0165-1765. doi: https://doi.org/10.1016/j.econlet.2011.03.006. URL https://www.sciencedirect.com/science/article/pii/S0165176511000991.

S Hylleberg. *Modelling Seasonality*. Oxford University Press, 08 1992. ISBN 9780198773177. doi: 10.1093/oso/9780198773177.001.0001. URL https://doi.org/10.1093/oso/9780198773177.001.0001.

RJ Kuo. A sales forecasting system based on fuzzy neural network with initial weights generated by genetic algorithm. *European Journal of Operational Research*, 129(3):496–517, 2001.

Gian Luigi Mazzi, Dominique Ladiray, and Dirk A. Rieser. *Handbook on Seasonal Adjustment*. Publications Office of the European Union, Luxembourg, 2018. URL https://ec.europa.eu/eurostat/documents/3859598/8939616/KS-GQ-18-001-EN-N.pdf/7c4d120a-4b8a-441b-aefd-6afe81a7cf59.

H.-G. Müller. Weighted local regression and kernel methods for nonparametric curve fitting. *Journal of the American Statistical Association*, 82(397):231–238, 1987. doi: https://doi.org/10.2307/2289159.

D. Ruppert and M. P. Wand. Multivariate locally weighted least squares regression. *The Annals of Statistics*, pages 1346–1370, 1994.

Dominik Schulz. The r package deseats for data-driven trend and seasonality estimation in time series, 2024. working parper.

Dominik Schulz, Yuanhua Feng, Thomas Gries, Marlon Fritz, and Sebastian Letmathe. Diagnosing the trend and bootstrapping the forecasting intervals using a semiparametric arma. Technical report, Paderborn University, 2021.

Gideon Schwarz. Estimating the dimension of a model. *The Annals of Statistics*, 6(2): 461–464, 1978. URL http://www.jstor.org/stable/2958889.

H.-T. Speth. *Komponentenzerlegung und Saisonbereinigung ökonomischer Zeitreihen mit dem Verfahren BV4.1*. Statistisches Bundesamt, 2004. German Federal Statistical Office report.

Leonard J. Tashman. Out-of-sample tests of forecasting accuracy: an analysis and review. *International Journal of Forecasting*, 16(4):437–450, 2000. ISSN 0169-2070. The M3-Competition.

The Open Group. *The TOGAF® Standard, 10th Edition - Introduction and Core Concepts*. van Haren Publishing, 2022. URL https://pubs.opengroup.org/togaf-standard/index.html.

U.S. Census Bureau. *X-13ARIMA-SEATS Reference Manual*. U.S. Census Bureau, 2017. URL https://www2.census.gov/software/x-13arima-seats/x13as/unix-linux/documentation/docx13ashtml.pdf. Reference Manual for the X-13ARIMA-SEATS seasonal adjustment program.

Indrė Žliobaitė, Jorn Bakker, and Mykola Pechenizkiy. Beating the baseline prediction in food sales: How intelligent an intelligent predictor is? *Expert Systems with Applications*, 39(1):806–815, 2012.

A Appendix A: Custom R-script for seasonality check in *deseatsForecast* application

```
## import library
library(deseats)

## log transform raw data for each country
log_be <-log(TSModel$`New Passenger Car Registration Belgium`)
log_fr <-log(TSModel$`New Passenger Car Registration France`)
log_ge <-log(TSModel$`New Passenger Car Registration Germany`)
log_gr <-log(TSModel$`New Passenger Car Registration Greece`)
log_uk <-log(TSModel$`New Passenger Car Registration United Kingdom`)

## transform log data into time series class objects
ts_be <- ts(log_be, start = c(1994, 1), frequency = 12)
ts_fr <- ts(log_fr, start = c(1994, 1), frequency = 12)
ts_ge <- ts(log_ge, start = c(1994, 1), frequency = 12)
ts_gr <- ts(log_gr, start = c(1994, 1), frequency = 12)
ts_uk <- ts(log_uk, start = c(1994, 1), frequency = 12)

## decompose and plot decomposition based on filtered country
if(region == 1){
  ## Belgium
  est_be <- deseats(ts_be, set_options(order_poly = 3))
  plot(est_be, which = 1)
}else if(region == 2){
  ## France
  est_fr <- deseats(ts_fr, set_options(order_poly = 3))
  plot(est_fr, which = 1)
}else if(region == 3){
  ## Germany
  est_ge <- deseats(ts_ge, set_options(order_poly = 3))
  plot(est_ge, which = 1)
}else if(region == 4){
  ## Greece
  est_gr <- deseats(ts_gr, set_options(order_poly = 3))
  plot(est_gr, which = 1)
}else if(region == 5){
  ## UK
  est_uk <- deseats(ts_uk, set_options(order_poly = 3))
  plot(est_uk, which = 1)
}
```

B Appendix B: Custom R-script for forecast, intervals and measures in *deseatsForecast* application

```
library(deseats)

## Parameters coming from SAC
# startR <- 1                    # simple mode / advanced mode
# n <- 372                       # observations
# h <- 12                        # periods to be forecasted
# last_period_year <- 2024       # year of last observation period
# last_period_month <- 12        # month of last observation period
# TSModel                        # data frame containing panel data from SAC table

## assign data frame
df <- TSModel

if( startR == 0){
## do nothing on application initialization
} else if (startR == 1) {
## run simple mode if startR == 1

  ## ------------------------------------------------------------
  ## Helper functions
  ## ------------------------------------------------------------

  ## Encode numeric vector
  encode_vec <- function(x, sep = ",") {
    paste(ifelse(is.na(x), "NA", format(x, scientific = FALSE, trim = TRUE)),
    ↪ collapse = sep)
  }

  ## Transform into a matrix
  as_matrixish <- function(x, default_row_prefix = "t") {

    if (is.matrix(x)) return(x)
    if (is.data.frame(x)) return(as.matrix(x))

    if (inherits(x, "ts")) {
      m <- as.matrix(x)
      tt <- time(x)

      rlab <- paste0(default_row_prefix, "_", seq_along(tt))
      if (frequency(x) == 12) {
        yrs <- floor(tt)
        mons <- round((tt - yrs) * 12 + 1)
        rlab <- sprintf("%04d-%02d", yrs, mons)
      }
      rownames(m) <- rlab
      return(m)
```

```
  }

  if (is.atomic(x) && is.null(dim(x))) {
    m <- matrix(as.numeric(x), ncol = 1)
    rownames(m) <- paste0(default_row_prefix, "_", seq_len(nrow(m)))
    colnames(m) <- "V1"
    return(m)
  }

  stop("A matrix-like object (matrix/data.frame/ts/vector) is expected, got: ",
  ↪ paste(class(x), collapse = ", "))
}

## Encode a matrix with header + payload
encode_matrix <- function(m,
                          sec_sep = "",
                          vec_sep = ",",
                          name_sep = "") {
  m <- as_matrixish(m)

  dn <- dimnames(m)
  rn <- if (!is.null(dn) && length(dn) >= 1) dn[[1]] else rownames(m)
  cn <- if (!is.null(dn) && length(dn) >= 2) dn[[2]] else colnames(m)

  header <- paste(
    nrow(m),
    ncol(m),
    if (is.null(rn)) "" else paste(rn, collapse = name_sep),
    if (is.null(cn)) "" else paste(cn, collapse = name_sep),
    sep = sec_sep
  )

  payload <- encode_vec(as.numeric(m), sep = vec_sep)
  paste(header, payload, sep = sec_sep)
}

## Encode any R object robustly
encode_any <- function(x) {
  if (!requireNamespace("jsonlite", quietly = TRUE)) {
    stop("Package 'jsonlite' is required. Install via:
  ↪ install.packages('jsonlite')")
  }
  jsonlite::toJSON(x, auto_unbox = TRUE, null = "null")
}

## Main encoder: forecasts + intervals + measures into one long string
encode_deseats_fc_with_measures <- function(fc_list, measures_list = NULL,
                                            obj_sep  = "",
                                            part_sep = "",
                                            kv_sep   = "",
```

```
                                        sec_sep  = "",
                                        vec_sep  = ",",
                                        name_sep = "") {

if (methods::is(fc_list, "deseats_fc")) fc_list <- list(fc_list)

## Ensure names exist
if (is.null(names(fc_list))) names(fc_list) <- paste0("series_",
↪ seq_along(fc_list))
if (any(names(fc_list) == "")) names(fc_list)[names(fc_list) == ""] <-
↪ paste0("series_", which(names(fc_list) == ""))

## Get measures for this series_name robustly
get_measures_for_series <- function(series_name, fc_obj) {
  if (is.null(measures_list)) return(NULL)

  m_entry <- NULL

  ## Direct name lookup
  if (!is.null(measures_list[[series_name]])) {
    m_entry <- measures_list[[series_name]]
  } else {
    ## Fallback: try trimmed names
    s_trim <- trimws(series_name)
    n_trim <- trimws(names(measures_list))
    idx <- match(s_trim, n_trim)
    if (!is.na(idx)) m_entry <- measures_list[[idx]]
  }

  if (is.null(m_entry)) return(NULL)

  if (is.list(m_entry) && !is.null(m_entry$measures)) return(m_entry$measures)
  return(m_entry)
}

one_object <- function(fc, series_name) {

  ## S4 object
  if (methods::is(fc, "deseats_fc")) {
    tsn    <- tryCatch(fc@ts_name, error = function(e) "")
    pred   <- fc@pred
    interv <- fc@interv

  ## List with pred and intervals
  } else if (is.list(fc) && !is.null(fc$pred) && !is.null(fc$interv)) {
    tsn    <- if (!is.null(fc$ts_name)) fc$ts_name else ""
    pred   <- fc$pred
    interv <- fc$interv

  } else {
```

```
      stop("encode_deseats_fc_with_measures(): deseats_fc or list with
 ↪ $pred/$interv expected, got: ",
           paste(class(fc), collapse = ", "))
    }

    ## Extract measures safely (numeric vector preferred)
    m <- get_measures_for_series(series_name, fc)

    blocks <- c(
      paste0("series",  kv_sep, series_name),
      paste0("ts_name", kv_sep, tsn),
      paste0("pred",    kv_sep, encode_matrix(pred,   sec_sep, vec_sep,
 ↪ name_sep)),
      paste0("interv",  kv_sep, encode_matrix(interv, sec_sep, vec_sep,
 ↪ name_sep))
    )

    ## Add measures
    if (!is.null(m)) {
      blocks <- c(blocks, paste0("measures", kv_sep, encode_any(m)))
    }

    paste(blocks, collapse = part_sep)
  }

  blocks <- mapply(one_object, fc_list, names(fc_list), SIMPLIFY = TRUE,
 ↪ USE.NAMES = FALSE)
  paste(blocks, collapse = obj_sep)
}


## ------------------------------------------------------------
## Calculate start year / month
## ------------------------------------------------------------

last_date  <- as.Date(sprintf("%04d-%02d-01", last_period_year,
 ↪ last_period_month))
start_date <- seq(last_date, by = "-1 month", length.out = n)[n]
start_year  <- as.integer(format(start_date, "%Y"))
start_month <- as.integer(format(start_date, "%m"))


## ------------------------------------------------------------
## Forecast all countries from 2nd to last column
## ------------------------------------------------------------

country_cols <- names(df)[2:ncol(df)]

## Train & forecast for one column
fit_forecast_one <- function(colname) {
```

```
  x <- df[[colname]]

  log_x <- log(x)
  log_x <- tail(log_x, n)

  ts_x <- ts(log_x, start = c(start_year, start_month), frequency = 12)

  model <- s_semiarma(ts_x, set_options(order_poly = 3))
  predict(model, n.ahead = h, method = "boot", expo = TRUE)
}

## Generate forecasts for all columns
fc_list <- setNames(lapply(country_cols, fit_forecast_one), country_cols)


## ------------------------------------------------------
## Generate backtesting measures all countries
## ------------------------------------------------------

## Backtest for all columns
backtest_all_countries <- function(df,
                                   start_year, start_month,
                                   last_period_year, last_period_month,
                                   h = 12,
                                   order_poly = 3) {

  country_cols <- names(df)[2:ncol(df)]

  ## Backtest for one column
  backtest_one <- function(colname) {
    raw <- df[[colname]]

    ## build full ts on original scale
    ts_raw <- ts(raw, start = c(start_year, start_month), frequency = 12)

    ## define train end
    train_end <- c(last_period_year - 1, last_period_month)

    ## train window up to train_end
    ts_train <- window(ts_raw, end = train_end)

    ## train model on log scale
    log_train <- log(ts_train)
    est <- s_semiarma(log_train, set_options(order_poly = order_poly))
    fc_obj <- predict(est, n.ahead = h, expo = TRUE)
    pred <- fc_obj@pred

    ## calculate measures
    list(
      measures = measures(pred, ts_raw)
```

```
      )
    }
    ## Return measures
    out <- setNames(lapply(country_cols, backtest_one), country_cols)
  }

  ## Generate measures for all columns
  measures_list <- backtest_all_countries(
    df = df,
    start_year = start_year,
    start_month = start_month,
    last_period_year = last_period_year,
    last_period_month = last_period_month,
    h = h,
    order_poly = 3
  )


  ## ------------------------------------------------------------
  ## Encode into one long string and return only that
  ## ------------------------------------------------------------

  ## Call encoder
  result_string <- encode_deseats_fc_with_measures(fc_list, measures_list)

  ## For debug
  ## print(result_string)

}else if(startR == 2){
## run advanced mode if startR == 2
...
}
```