Introducing a cGAN-Based Approach for Seasonal Adjustment

Rouven Beiner

March 6, 2025

Abstract

This paper evaluates the performance of seasonal adjustment methods using Monte Carlo simulated synthetic time series. I introduce a new method for seasonal extraction based on a conditional Generative Adversarial Network (cGAN) and seasonal subseries smoothing using LOESS. Results suggest that cGAN-LOESS can outperform traditional methods, and an ensemble of different base methods can further improve performance.

Contents

1 Introduction

Many time series exhibit seasonal patterns that can bias the analysis if the primary interest lies in identifying the cycle or trend components. For example, a decrease in manufacturing production from June to July should not be immediately interpreted as an economic slowdown, as production reductions are common at the start of the summer holiday season. Adjusting for seasonality is therefore an essential preprocessing step for many time series analyses.

Various methods exist for filtering seasonality and statistical offices around the world have largely concentrated on X-13-ARIMA-SEATS (X13), Tramo/Seats, the Berliner Verfahren and STL. Previously, Cuevas and Quilis (2023), Franses et al. (2005) and Abeln and Jacobs (2015) analyzed the performance of TBATS, DSA, X13, CAMPLET and Tramo/Seats using Monte Carlo simulations. I evaluate the performance of X13, STL, and a newly introduced cGAN-LOESS method for seasonality prediction on a set of 2,500 Monte Carlo-simulated synthetic time series that mimic the output of economies. Additionally, a combined ensemble method that averages the seasonality extraction of different base learners is tested.

The newly introduced method for seasonal adjustment—cGAN-LOESS—is based on seasonality prediction using a cGAN followed by seasonal subseries smoothing using LOESS. A cGAN pix2pix model as described by Isola et al. (2016) can be used in a wide range of tasks, including removing noise from images, generating colorized photos from black-and-white images or transforming sketches into photos. They can be analogously used to isolate different components in a time series by treating the time series as a single-channel image. Unlike traditional smoothing or filtering methods for seasonal adjustment, cGAN-LOESS learns the underlying distribution of a data-generating process. For training, the analyst must define a data-generating process and generate corresponding samples. Once trained, the model requires only three parameters to adjust. Results show that cGAN-LOESS outperforms traditional methods, and ensemble methods can further improve performance.

This paper is organized as follows. Section 2 defines seasonality in theory. Section 3 presents conventional methods for seasonal adjustment, focusing on both the X-Family and STL. Section 4 provides an introduction to GANs. Section 5 describes the proposed cGAN–LOESS approach for seasonality extraction in detail. Section 6 explains how synthetic time series are generated. Section 7 evaluates the performance of seasonal adjustment methods.

2 Fundamentals of Seasonality

A key aspect of seasonality is that it has a *fixed frequency* (e.g., daily, weekly, or yearly). This requirement differentiates seasonality from other components, such as business cycles, which typically exhibit varying frequencies over time.

In mathematical terms, seasonality can be categorized along two dimensions: *deterministic* versus *stochastic* and *stationary* versus *non-stationary*. Consider the simple seasonal autoregressive process:

$$X_t = \Phi\, X_{t-s} + \varepsilon_t, \tag{1}$$

where s is the seasonal period (e.g., $s = 12$ for monthly data) and $\{\varepsilon_t\}$ is an i.i.d. noise sequence with mean zero and variance σ^2. Larger positive values of Φ reflect stronger seasonal dependence, and $\Phi = 1$ results in a seasonal unit root. Further let $\rho_h = Corr[X_{t+h}, X_t]$ be the autocorrelation function (ACF).

Seasonality can be categorized as follows:

1. **Deterministic Non-stationary**: $\sigma^2 = 0$, The ACF satisfies $\rho_h = 1$ whenever $h = n\,s$ for an integer n.
2. **Stochastic Non-stationary**: $\sigma^2 > 0$ and $\Phi = 1$.
3. **Stochastic Stationary**: $\sigma^2 > 0$ and $\Phi \in (-1, 1)$. The ACF satisfies $\rho_h = \Phi_1^n$ whenever $h = n\,s$ for an integer n.

I argue that random events—even if related to seasonal determinants—should not affect the pattern of seasonality. This speaks in favor for *deterministic* seasonality. For instance, tourist revenue in the Alps tends to peak in February due to winter sports. This yearly fluctuation follows a regular pattern and thus qualifies as seasonality. However, if an unseasonably warm February leads to melting snow and a one-time drop in revenue, that deviation does not change the existing seasonal pattern—unless warm February weather becomes a recurring event. In such a single occurrence, the revenue drop is attributable to a weather shock, an external influence rather than a fundamental shift in seasonality. Consequently, random disturbances should be distinguished from seasonality and treated as noise or as part of cyclical or trend components.

I further argue that seasonality may gradually shift over time—due, for example, to climate change—such shifts should exhibit a clear, structured pattern. Hence, even in deterministic models, one should allow for time-varying coefficients in the seasonal component to accommodate slow-moving changes. Methods such as STL decomposition capture these gradual shifts by smoothing trends in the seasonal subseries[1].

[1] A seasonal subseries with yearly seasonality contain values of a specific month across all years.

3 Conventional Methods for Seasonal Adjustment

If seasonality is purely *deterministic*, it can be addressed by including seasonal fixed effects or trigonometric functions in a regression (Harvey et al. (1997), De Livera et al. (2011)). When the seasonal component exhibits *stochastic* characteristics, it is necessary to estimate or extract the random seasonal effects, commonly using linear filters (McElroy and Roy (2022)). One drawback of filtering-based methods, such as the X-family and STL, is the endpoint bias, which I further discuss in Section A. Another concern is the possibility of overadjustment. Many filters over adjust by removing not only the seasonality but some of the non-seasonal content as well.

3.1 X-Family

The X-12 family of seasonal adjustment methods includes the original X-11 method and its successors, X-11-ARIMA, X-12-ARIMA, and X-13-ARIMA-SEATS. The Census Bureau's X-11 program for seasonal adjustment, introduced in 1978 (Shiskin (1978)), has become widely used worldwide. Key features are handling of extreme observations, weighted symmetric moving averages to separate trend-cycle and seasonal components, refined asymmetric moving averages for endpoints, and techniques for estimating trading-day effects.

The subsequent X-11-ARIMA program (Dagum (1980)) enhanced X-11 by extending the time series with ARIMA-based forecasts and backcasts, improving the stability of symmetric filters—especially at series endpoints.

The next development, X-12-ARIMA, was introduced by the Census Bureau and is detailed in Findley et al. (1998). The most important enhancements are the model-based calendar-effect adjustment using a RegARIMA model and several new diagnostic tools to test the quality of seasonal and calendar-effect adjustments. The RegARIMA model is defined as:

$$\ln(X_t) = \sum \alpha_l \cdot k_{lt} + \sum \beta_m \cdot LS_{mt} + \sum \gamma_n \cdot AO_{nt} + \sum \lambda_v \cdot TC_{vt} + Z_t \qquad (2)$$

where k_{lt} are calendar regressors (e.g., the number of working days per month), LS_{mt}, AO_{nt} and TC_{vt} are level shifts, additive outliers and temporary changes. Z_t is a seasonal ARIMA$(p,d,q)(P,D,Q)$ model component. Common ARIMA specifications, such as (011)(011), apply a difference operation both to the previous period ($d = 1$) and to the same period in the prior year ($D = 1$), with moving averages ($q = 1, Q = 1$) that depend on recent and seasonal influences. The RegARIMA modeling serves two key functions. First, it allows identification and removal of calendar effects. Second, it enables forward and backward projections to reduce distortions at the series boundaries in the filtering process. The second

stage in X-12-ARIMA involves applying a Henderson filter to estimate the trend-cycle component, followed by smoothing of the seasonal subseries. The seasonal subseries are smoothed with a series of moving averages.

The recent version X-13-ARIMA-SEATS uses the procedures of the predecessors and additionally uses the SEATS method developed by Gómez and Maravall (2001) at the bank of spain.

3.2 STL

STL is a filtering procedure for decomposing time series into trend, seasonal, and remainder components by applying a loess smoother and various moving averages. Proposed by Cleveland et al. (1990), the method allows for individualized specification of the seasonal period, provides robust estimates of the trend and seasonal components, and can handle time series with missing values. STL consists of an inner and an outer loop. Within each iteration of the inner loop, the seasonal and trend components are updated. First, the time series is detrended. Then, each seasonal subseries of the detrended series is smoothed by LOESS, and a low-pass filter is applied to the smoothed seasonal subseries. Each outer loop includes the inner loop followed by a calculation of robustness weights, which reduce the influence of transients in the next inner loop iteration.

4 A Short Introduction to Generative Adversarial Networks

In the context of supervised learning, models are often categorized as either *generative* or *discriminative*. Within the generative category, there is a further distinction between *explicit* and *implicit* approaches.

Explicit generative models learn a joint probability distribution $P(X, Y)$ for inputs X and outputs or labels Y. In unsupervised settings, this may reduce to $P(X)$. By modeling the full distribution, these models allow likelihood-based inference (e.g., via Bayes' theorem in classification tasks). In contrast, *implicit generative* models do not define a probability density function explicitly. Instead, they generate samples from the data distribution—typically from random noise—without computing a tractable likelihood.

Discriminative models learn the conditional distribution $P(Y \mid X)$ directly. They do not capture the full underlying data distribution, instead focusing on the decision boundary or functional mapping of inputs X to outputs Y.

Unconditional GANs, first proposed by Goodfellow et al. (2014), are *implicit generative* models that consist of two models trained simultaneously —a generator G and a discriminator D, both often implemented as multilayer perceptrons. The

generator aims to approximate the real data distribution p_{data} with its own distribution p_g. Specifically, it learns a mapping function $G(z;\theta_g)$ that transforms noise z into synthetic samples in the target data space y. The discriminator $D(y;\theta_d)$ outputs a single scalar representing the probability that a given input y originated from p_{data} rather than from p_g.

The training procedure is cast as a two-player minimax game in which G and D optimize their parameters θ_g and θ_d, respectively, as follows:

$$\min_G \max_D V(D,G) = \mathbb{E}_{y\sim p_{\text{data}}(y)}\left[\log D(y)\right] + \mathbb{E}_{z\sim p_z(z)}\left[\log(1-D(G(z)))\right] \quad (3)$$

Here, the generator G is trained to minimize $\log(1-D(G(z))$, which corresponds to maximizing the discriminator's probability $D(G(z))$ of mistakenly classifying a generated sample as real. Meanwhile, the discriminator D aims to distinguish true samples from generated ones as accurately as possible, maximizing $\log(D(y))$ (assigning high probabilities to real samples) and maximizing $\log(1-D(G(z))$ (assigning low probabilities $D(G(z))$ to generated samples). Through this adversarial process, G learns to produce samples that are increasingly difficult for D to distinguish from the real data, effectively bringing p_g closer to p_{data} as training progresses. This minimax game has a global optimum for $p_g = p_{\text{data}}$. Applied to time series data, a GAN can generate new, unobserved seasonality structures (or other time series components) from random noise z, by training the model on various seasonality patterns as model output y. Some researchers have used unconditional GANs to generate time-series data (Esteban et al. (2017); Smith and Smith (2020)).

Conditional GANs, first proposed by Mirza and Osindero (2014), extend the GAN by introducing additional input information x, enabling control over the characteristics of the generated data. The conditioning is performed by feeding x into both the discriminator and generator as an additional input layer. Therefore, in a cGAN, the discriminator learns to classify between fake tuples (y,G(z,x)) and real tuples (y,x). The generator learns a mapping from both x and the random noise z, to a target output y, $G((z,x);\theta_g)$. The objective of a cGAN is given by:

$$\min_G \max_D V(D,G) = \mathbb{E}_{y\sim p_{\text{data}}(y)}\left[\log D(y,x)\right] + \mathbb{E}_{z\sim p_z(z)}\left[\log(1-D(G(z,x),x))\right] \quad (4)$$

CGANs learn a structured loss that penalizes the joint configuration of output pixels, instead of penalizing each data point conditionally independent from all other data points, resulting in a per-pixel or per-data point regression (Isola et al. (2016)). This enables cGANs to capture complex patterns within the output space without requiring feature engineering, such as constructing interaction terms or lag variables. This is an important advantage in the time series context, particularly when the data-generating process is unknown or difficult to model. As a downside

of GANs, mode collapses can occur, where the generator produces limited variety in its outputs, often generating similar or identical samples regardless of input noise. However, various training strategies can help mitigate this issue.

The architecture of the GAN compared to a cGAN is illustrated in Figure 1 and Figure 2.

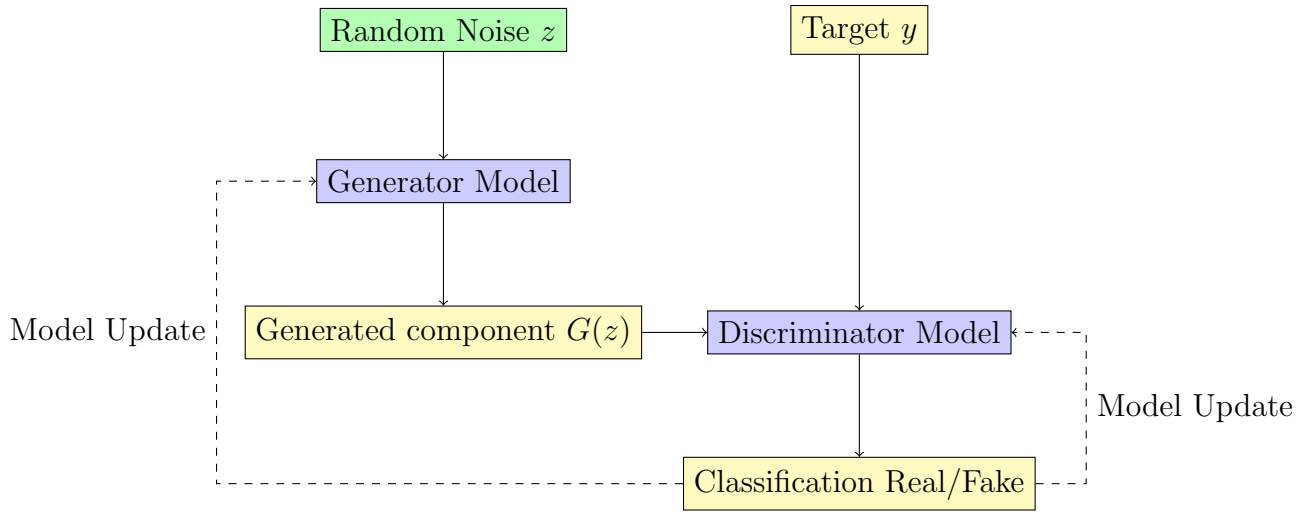


Figure 1: Flowchart of a GAN

5 A Conditional Generative Adversarial Network for Seasonal Adjustment

I base my cGAN implementation on the architecture proposed by Isola et al. (2016), which uses a U-net generator and a convolutional PatchGAN discriminator. I removed the "random jitter" from the original design and changed the input/output dimensions from 256×256×3 (an RGB image) to 256×1, matching a time series of 256 observations.

Because cGAN-based seasonality extraction requires a pre-trained model, its accuracy depends on the quality and suitability of the training data, defined by the researcher. Ideally, the training data should have a similar structure to the time series being deseasonalized. Furthermore some layers behave stochastically (e.g. dropout and batch normalization), and the predictions of the same input will differ slightly even after the finished training phase.

The cGAN–LOESS seasonality extraction procedure can be divided into two main phases:

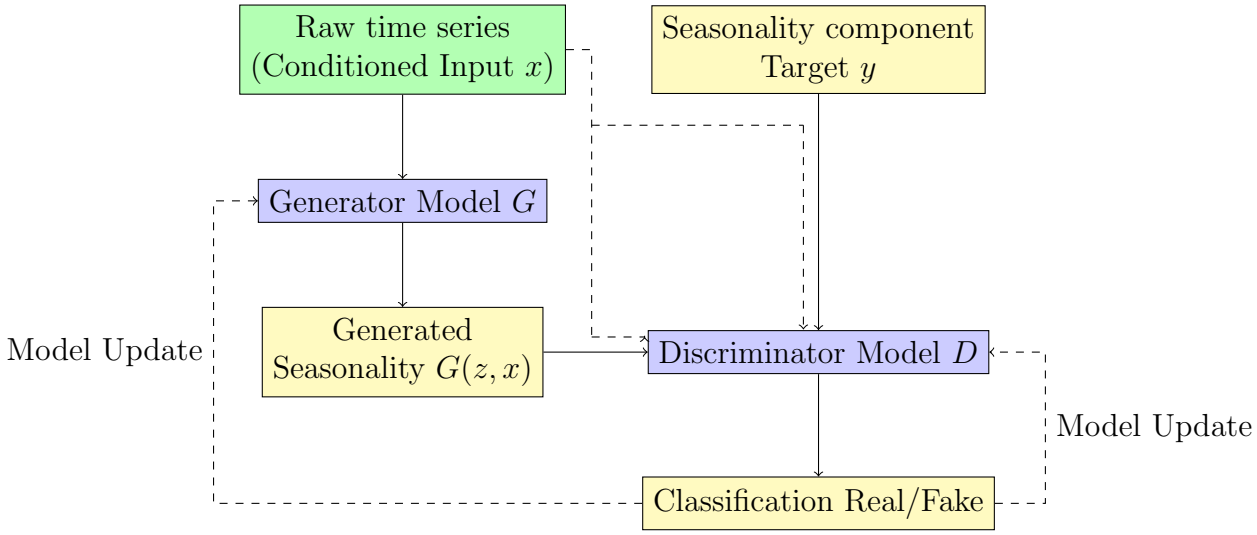


Figure 2: Flowchart of a Conditional GAN

I. Time Series Generation and Model Training

1. **Synthetic Time Series Generation**: Generate synthetic time series data that include a separate seasonality component.
2. **Time Series Pre-processing**: Detrend and normalize the synthetic time series.
3. **Model Training**: Train the cGAN on the pre-processed synthetic time series.

II. Seasonality Prediction

1. **Time Series Pre-processing**: Detrend and normalize the real time series using the same procedure as in training.
2. **Prediction**: Predict the seasonality component using the pre-trained cGAN.
3. **Inverse Scaling**: Revert the data to its original scale.
4. **LOESS Smoothing**: Apply LOESS smoothing to the predicted seasonal subseries, selecting the optimal bandwidth via cross-validation.

The generation of suitable synthetic time series is discussed in Section 6. I remove trends from the time series using the Hodrick and Prescott (1997) (HP) filter. The smoothing parameter (λ) of the HP filter is the first parameter to be specified in

the cGAN-LOESS procedure. The cutoff frequency should be set slightly above the seasonal period. For yearly seasonality, a 2-year cutoff works well. As shown in Section A, fore- and backcasting can help reduce endpoint bias in the HP filter. The HP filter minimizes the following objective function:

$$\min_{\{\tilde{y}_t\}_{t=1}^{T}} \left\{ \sum_{t=1}^{T} (y_t - \tilde{y}_t)^2 + \lambda \sum_{t=2}^{T-1} [(\tilde{y}_{t+1} - \tilde{y}_t) - (\tilde{y}_t - \tilde{y}_{t-1})]^2 \right\} \tag{5}$$

The filter computes a stochastic trend $\{\tilde{y}_t\}_{t=1}^{T}$ by minimizing the squared deviations from the trend $(y_t - \tilde{y}_t)^2$, constrained to keep the squared second differences small $[(\tilde{y}_{t+1} - \tilde{y}_t) - (\tilde{y}_t - \tilde{y}_{t-1})]^2$. The frequency response function of the Hodrick-Prescott filter is given by:

$$R(\omega) = \frac{1}{1 + 4\lambda[1 - \cos(\omega)]^2} \tag{6}$$

Now solving for the HP-filter parameter λ:

$$\lambda = \frac{1 - R(\omega)}{4R(\omega)[1 - \cos(\omega)]^2} \tag{7}$$

It can be shown that in the case of the HP filter, the gain function equals the frequency response function. Using a gain of $R(\omega) = 0.5$ and replacing the frequency in radians (ω) with the ordinary frequency (f) multiplied by 2π, the equation can be simplified as follows:

$$\lambda = \frac{1 - 0.5}{4 \cdot 0.5 \cdot [1 - \cos(\omega)]^2} = \frac{0.25}{[1 - \cos(\omega)]^2} = \frac{0.25}{[1 - \cos(2\pi f)]^2} \tag{8}$$

The lambda parameters are set so that the frequency cut-off occurs at frequencies higher than 2 years. Therefore, the corresponding ordinary frequencies of $f_1 = \frac{1}{24}$ is used:

$$\lambda_1 = \frac{0.25}{\left(1 - \cos\left(\frac{1}{24} 2\pi\right)\right)^2} = 215.3 \tag{9}$$

After detrending, the time series is normalized by dividing it by the largest absolute value. This step ensures a consistent $[-1, 1]$ scale for improved training and also simplifies recovery of the original scale.

For the cGAN training, the researcher has to specify the size of the convolution window in the (transposed) convolution layers. This integer parameter should be set equal to the periodicity of the seasonality, for example, 12 for monthly data and yearly seasonality.

Following the seasonality prediction, LOESS smoothing is applied to each seasonal subseries. This procedure removes high-frequency fluctuations from the seasonal component while allowing smooth transitions over time. The LOESS algorithm, introduced by Cleveland (1979), estimates a smooth curve through local linear regression. For a specific seasonal factor x, the local linear regression is defined as:

$$\min_{\alpha,\beta} \sum_{i}^{N} \left(Y_i - (\alpha + \beta X_i)\right)^2 K(\mathbf{X}_i, \mathbf{x}, \mathbf{H}), \tag{10}$$

Here, $i = 1, \ldots, N$ indexes the seasonal factors within a given group, and $K(\mathbf{X}_i, \mathbf{x}, \mathbf{H})$ is a kernel function that weights the proximity of other seasonal factors (e.g., January 2009, January 2011) $\mathbf{X}_i$ to the current seasonal factor (e.g., January 2010) $\mathbf{x}$. The minimization problem estimates α and β by minimizing the squared difference between the observed seasonality value Y_i and $(\alpha + \beta X_i)$, weighted by the kernel. Here, a tricube weighting function is used:

$$K\left(\frac{|X_i - x|}{h}\right) = \left(1 - \left(\frac{|X_i - x|}{h}\right)^3\right)^3, \tag{11}$$

where h is the bandwidth. The bandwidth is the third parameter to be specified. Bandwidth selection is crucial: a small h may closely fit the seasonal subseries but lead to overfitting and high variance, whereas a large h lowers variance but may fail to capture shifts in seasonality.

6 Synthetic Time Series generation

I generate synthetic time series using a multiple source of error state space framework that combines the following key components: A trend (T_t), two cycles (LC_t and SC_t) a time varying seasonality component (S_t), and an outlier component. The outlier component accounts for additive outliers (ao_t), temporary changes (tc_t) and structural breaks or level shifts (ls_t). The components are combined additively as follows:

$$Y_t = T_t + LC_t + SC_t + S_t + ao_t + tc_t + ls_t \tag{12}$$

In this type of model, the components are uncorrelated and shocks to different components are orthogonal. A zero correlation between the trend component and the cycle components is assumed in a standard text book RBC model where technology shocks drive stationary fluctuations around an exogenous growth trend. In contrast, endogenous growth models assume that shocks to the trend and the cyclical component are correlated.

The stochastic trend component T_t is a random walk with drift, defined by:

$$\Delta T_t = T_t - T_{t-1} = \delta + \epsilon_t, \tag{13}$$

Recursively substituting for T shows that this trend builds cumulatively over time as:

$$T_t = T_0 + t\delta + \sum_{i=1}^{t} \epsilon_{T,i} \tag{14}$$

where δ is the drift parameter, with $\delta \sim \mathcal{N}(0, 0.025^2)$ and $\epsilon_{T,i}$ is random noise with $\epsilon_{T,i} \sim \mathcal{N}(0, \sigma_T^2)$, and $\sigma_T \sim \mathcal{U}(0.01, 0.2)$.

The two cycle components are generated using two theories, namely RBC and Slutzky. In 1937, E.E. Slutsky's article (Slutzky (1937)), "The Summation of Random Causes as the Source of Cyclic Processes" appeared in Econometrica, following its earlier publication in the theoretical journal of the Moscow Conjuncture Institute. In that paper, Slutsky asked whether "a definite structure of a connection between random fluctuations could form them into a system of more or less regular waves" (p. 106). By examining the cumulative effects of random events—such as rainfall on crop yield—he showed that these could produce regular patterns. Specifically, he transformed a series of random numbers by applying various moving-average and differencing procedures sequentially.

Kuznets (1929) supported Slutsky's reasoning and highlights that a single, exceptionally large deviation can drive cyclical swings: "In a moving average or in a cumulation, this deviation, exceptional in its size, will raise or depress the level of all the members that include it, and instead of one high or low point we get a high or low plateau" (p. 267).

One year before Slutsky's publication, in 1926, G. U. Yule independently identified similar characteristics. He proposed that autoregressive (AR) processes can also generate persistent cyclical behavior, even when the underlying shocks are random. Later, Wold (1938) established the connection between Yule's AR(p) formulation and Slutsky's MA(q) approach.

A straightforward business cycle theory can emerge from Yule's and Slutsky's findings. The economy experiences a shock in each period, representing any factor influencing economic activity—such as productivity gains, weather events, or aggregated individual decisions. Under the first (Yule-type) assumption, shocks are correlated—one productivity gain can trigger a sequence of others, and current decisions often depend on previous outcomes. Under the second (Slutsky-type) assumption, the economy does not respond immediately but rather reflects an average of prior shocks, consistent with long-term contracts, imperfect information, sticky prices, or labor-market constraints.

This simple framework is used to generate two cycle components in the synthetic time series. First, a time series with autoregressive shocks is constructed;

then, moving averages are applied to capture the economy's gradual response to these shocks. The "long cycle" is simulated with a larger moving-average window, capturing more prolonged fluctuations, similar to a super-cycle or Kondratieff cycle. In contrast, the "short cycle" uses a smaller window, reproducing the briefer ups and downs characteristic of typical business cycles.

The cycle components, LC_t and SC_t, are constructed as AR(p) processes:

$$LC_t = \epsilon_{LC,t} + \sum_{i=1}^{p} \phi_i \cdot LC_{t-i}, \tag{15}$$

$$SC_t = \epsilon_{SC,t} + \sum_{i=1}^{p} \psi_i \cdot SC_{t-i}. \tag{16}$$

where $\epsilon_{LC,t} \sim \mathcal{N}(0, \sigma_{LC}^2)$ with $\sigma_{LC} \sim \mathcal{U}(2,5)$, and $\epsilon_{SC,t} \sim \mathcal{N}(0, \sigma_{SC}^2)$ with $\sigma_{SC} \sim \mathcal{U}(3,7)$. The autoregressive parameters decay geometrically as $\phi_i = \phi_{i,0} \cdot (0.5)^i$ with $\phi_{i,0} \sim \mathcal{N}(0, 0.5)$ and $\psi_i = \psi_{i,0} \cdot (0.5)^i$ with $\psi_{i,0} \sim \mathcal{N}(0, 0.5)$. If $\sum_{i=1}^{p} \psi_i > 1$ or $\sum_{i=1}^{p} \phi_i > 1$, the parameter series are scaled to stabilize the process. Moving averages with window sizes $W_{LC} \sim \mathcal{U}(200, 250)$ and $W_{SC} \sim \mathcal{U}(48, 72)$ are applied to smooth the cycle components:

$$LC_t = \frac{1}{W_{LC}} \sum_{j=0}^{W_{LC}-1} LC_{t-j}, \tag{17}$$

$$SC_t = \frac{1}{W_{SC}} \sum_{j=0}^{W_{SC}-1} SC_{t-j}. \tag{18}$$

The seasonality component S_t is composed of two *deterministic non-stationary* seasonal patterns, $S_{1,t}$ and $S_{2,t}$ which change in relative weight over time according to w_t:

$$S_t = S_{1,t} \cdot w_t + S_{2,t} \cdot (1 - w_t) \tag{19}$$

The two patterns and the weight ensure that the seasonality component randomly changes the weight of the two seasonality patterns. The weight w_t is itself a random walk and is bounded by two parameters, $w_{\min}$ and $w_{\max}$:

$$w_t = w_{t-1} + \epsilon_t \tag{20}$$

where $\epsilon_t \sim \mathcal{N}(0, 0.15)$. The bounds are given by $w_{\min} \sim \mathcal{U}(0, 0.5)$ and $w_{\max} \sim \mathcal{U}(0.5, 1)$, ensuring w_t remains within the interval $[w_{\min}, w_{\max}]$. Each seasonal pattern is a random walk with 12 periods. The seasonal pattern $S_{s,t}$ with $s \in (1, 2)$ is given by:

$$S_{s,t} = \sum_{i=1}^{12} \epsilon_{S_{s,i}} \tag{21}$$

where $\epsilon_{S_s} \sim \mathcal{N}(0, \sigma^2_{S_s})$ with $\sigma_{S_s} \sim \mathcal{N}(0, 0.1^2)$.

Additive outliers ao_t affect only a single observation or data point due to singular events like unusual holiday timing or extreme weather conditions.

Temporary changes represent transitory shocks that weaken over time, such as the impact of major events like the World Cup or the Olympics on employment. The duration of the temporary change tc_t is determined by $d_{tc,i} \sim \mathcal{U}(1, 20)$ with a linear decay effect over the duration.

The permanent level shift ls_t shifts all values from its occurrence onward.

The number of occurences for the three outlier types is determined by uniform distributions $n_{ao} \sim \mathcal{U}(0, 10), n_{tc} \sim \mathcal{U}(0, 5), n_{sb} \sim \mathcal{U}(0, 3)$. The strength of each outlier is normally distributed as $\mathcal{N}(0, 1)$.

This configuration can generate a wide range of patterns and defines how each time series component evolves through a set of distributional properties. Figure 3 shows the components of a sample synthetic time series.

7 Performance Evaluation

In real-world data, researchers typically have access only to the additive time series, not its individual components, making it difficult to distinguish between stylized facts and statistical artifacts. Grether and Nerlove (1970) suggest the following informal criteria for assessing the adequacy of seasonal adjustment procedures when the true seasonal component is unknown. First, the procedure should remove power at the seasonal frequencies while leaving the rest of the spectrum as intact as possible. Second, any phase shifts introduced by relying on past data should be minimized, especially at lower frequencies where most economic variation occurs. However, the first criterion is particularly challenging to measure because complex seasonal patterns include harmonic frequencies (or "overtones") of different power above the fundamental frequency.

Another approach is to interpret each seasonality-adjustment method as a filter in the frequency domain. It was proposed by Pedersen (2001) in the context of business cycle extraction. This perspective allows a direct comparison to an ideal high-pass or band-pass filter[2]. This ideal filter serves as a benchmark or "true filter," capturing the desired seasonal component and allowing any deviation from it to be viewed as distortion. However, two issues arise: first, assuming that this ideal filter adequately represents the true seasonal component may be debatable; second, an ideal seasonality high-pass (or band-pass) filter would also capture the harmonic frequencies of the cyclical or trend components, thus violating the first informal criterion.

[2]An ideal high-pass filter has a power transfer function of the form $H^*_{hp}(\omega) = 0$ if $|\omega| < \omega_1$, 1 if $|\omega| \geq \omega_1$

As a promising alternative, seasonal adjustment procedures can be evaluated through simulation, where the true seasonal component is known. In this setting, the MSE between the extracted seasonality and the ground truth seasonality provides a straightforward metric to quantify overall distortion.

To extract seasonality using the three tested methods, I used the following specifications. For X13, I used maximal SARIMA orders of (4,1,4)(1,1,1), enabled automatic outlier detection and testing, and included internal forecasting for five periods. Since the synthetic time series does not exhibit trading day effects, the automatic trading day adjustment was deactivated. For STL, I specified a periodicity of 12, a seasonal smoother length of 7, a trend smoother length of $1.5 * periodicity/(1 - 1.5/seasonal) = 23$ (following the suggestion in the original implementation), and a low pass filter length of 13. The cGAN-LOESS model training set consists of 2,500 synthetic time series, generated with stochastic properties as described in Section 6. The cGAN model training involves 75,000 steps and the kernel size is set to 12, matching the seasonal periodicity. For the seasonal factor smoothing in the cGAN-LOESS, the 6 nearest values are considered in the LOESS smoother.

Additionally, two combined ensemble models are constructed by taking the average and median of the extracted seasonality components from the three base methods. Ensemble methods are generally more reliable and robust, and they form decisions analogously to (rational) humans - by seeking opinions from multiple experts.

The MSE is estimated for each method on a set of 2,500 synthetic test time series. The distribution of the methods MSE across all test time series is shown in Figure 4(a), and the means and medians are displayed in Table 1.

In terms of mean MSE, the mean-ensemble approach achieves the lowest value (0.0046), followed by cGAN–LOESS (0.0049), the median ensemble (0.0050), and X13 (0.0055), whereas STL exhibits the highest mean MSE (0.0096). Similarly, the mean-ensemble method attains the lowest median MSE (0.0040), while STL again exhibits the highest mean MSE (0.0084). Both ensemble methods and X13 show relatively low variance, suggesting greater robustness, whereas cGAN–LOESS and STL display the highest variability. As illustrated by the violin plots in Figure 4(a), cGAN–LOESS produces several extreme outliers, whereas the ensemble methods do not, likely because averaging helps reduce outlier effects.

The mean and median values may differ but the MSE distributions have significant overlaps and do not provide sufficient clarity if differences are statistically significant. Therefore, bootstraps were used to determine if the differences in performance between methods are statistically significant. This was achieved by generating 10,000 bootstrap samples from the 2,500 time series and estimating the mean MSE for each bootstrap sample. The resulting bootstrap distributions

Table 1: Mean and Median MSE of Different Methods

	cGAN-LOESS	X13	STL	Mn. ensemble	Med. ensemble
Mean	0.0049	0.0055	0.0096	0.0046	0.005
Median	0.0038	0.0048	0.0084	0.0040	0.0044
St. dev.	0.0045	0.0034	0.0054	0.0026	0.0028

along with the 5% and 95% quantiles for each method are presented in Fig 4(b) and Table 2. The mean ensemble method has a significantly lower mean MSE than all other methods, as its 90% CI does not overlap with that of any other method. Both the X13 and STL have significantly higher mean MSEs than cGAN–LOESS.

Table 2: 5% and 95% Quantiles of Bootstrapped Mean MSE for Different Methods

	cGAN-LOESS	X13	STL	Mn. esemble	Med. ensemble
Q 5%	0.00472	0.00537	0.00944	0.00448	0.00486
Q 95%	0.00503	0.00558	0.0098	0.00465	0.00504

Figure 5 and Figure 6 show the seasonal subseries of the base and ensemble methods for an example synthetic time series. All methods seem to catch the general seasonal level quite well, as they have a similar level and roughly the same trend as the original seasonal subseries (the black line). But all methods struggle with identifying finer transitions and movements in the seasonal subseries. In general, the extracted seasonality patterns of X13 and STL are similar while the cGAN-LOESS has a different pattern. This reflects the different approaches of the methods, and shows that cGAN-LOESS could complement seasonal extraction. The methods are applied on real world example, namely the manufacturing orders (Auftragseingänge) which exhibit strong seasonality. For manufacturing orders Figure 7 shows the methods seasonal components and Figure 8 shows the seasonal subseries.

7.1 Meta Learner

I estimate the following regression to understand the relationship of time series characteristics and the performance of each method:

$$MSE_i = \alpha + \mathbf{parameter}_i \boldsymbol{\beta} + \epsilon_i, \tag{22}$$

where MSE_i is the mean squared error for time series i, $\mathbf{parameter}_i$ is the vector of parameters from the data-generating process, $\boldsymbol{\beta}$ is the corresponding vector

of coefficients, and ϵ_i is the error term. Table 3 presents the model summaries for each seasonal adjustment method. The R^2 values indicate that cGAN–LOESS depends only weakly on the parameters ($R^2 = 0.05$). This suggests that it captures the whole data distribution effectively and that it can handle different realizations of the simulated time series well. In contrast, the filter-based methods perform only under specific conditions as their R^2 values are much larger ($R^2 = 0.25$ and $R^2 = 0.4$).

The "trend scale" (T scale) parameter determines the variance of the random walk in the trend; its positive coefficients imply that higher noise in the trend increases errors for all methods. Notably, X13 and STL have larger coefficients, indicating that they filter out trend-related noise less effectively than cGAN–LOESS.

Cycle-related characteristics, such as "long cycle window size" (C_l win. size) and "short cycle window size" (C_s win. size) also play a role. A larger window size—corresponding to lower-frequency cycles—reduces MSE for all methods, likely because longer cycles are better to distinguish from the seasonality "cycle". Finally, zero seasonality ($\neq S$) lowers MSE, particularly for cGAN–LOESS, suggesting that it handles non-seasonal time series more effectively than the other approaches. This finding aligns with the additional MSE evidence for zero seasonality time series in Figure 9.

8 Discussion and Conclusion

CGAN-LOESS requires specifying only three parameters: the HP filter's smoothing parameter, the seasonal periodicity within the cGAN, and the LOESS smoothing parameter. Once trained, the model is straightforward to use, and these parameters are relatively easy to determine. However, the model's performance depends on the data-generating process chosen for training, which introduces considerable researchers degree of freedom regarding parameter selection.

Unlike STL and X13—both filter-based methods that directly extract seasonality—cGAN–LOESS learns the underlying data-generating process in order to predict the seasonal component. As a result, its extracted seasonality may differ categorically from that of traditional methods. In tests on synthetic time series, cGAN–LOESS outperformed both STL and X13. Furthermore, ensemble approaches improved performance and reduced variability and outlier occurrences.

More generally, cGANs can be used to disaggregate a time series into multiple subcomponents, making them suitable for a broad range of applications. In the future, similar methods could be employed to extract other components, such as business cycles, trends, or noise. In my analysis, the cGAN–LOESS model is trained on fixed time series of 256 observations. Future work should explore its performance for time series of varying lengths.

Table 3: OLS meta learner summary table

Variable	cGAN-LOESS	X13	STL	Mn. ensemble	Med. ensemble
const	0.00460***	0.00520***	0.00937***	0.00438***	0.00475***
T drift	0.00108	-0.00064	-0.00193	-0.00087	-0.00094
T scale	0.01358***	0.04610***	0.08050***	0.03462***	0.03952***
C_l win. size	-0.00002***	-0.00002***	-0.00002***	-0.00001***	-0.00001***
C_l amp	-0.00001	-0.00007	-0.00015**	-0.00006*	-0.00007*
C_s win. size	-0.00004***	-0.00011***	-0.00025***	-0.00010***	-0.00010***
C_s amp	0.00003	0.00024***	0.00069***	0.00022***	0.00024***
S_1 scale	0.00404***	-0.00027	-0.00154	0.00013	-0.00010
S_2 scale	0.00463***	0.00055	0.00006	0.00092*	0.00083
S_1 ar	0.00404***	-0.00027	-0.00154	0.00013	-0.00010
S_2 ar	0.00463***	0.00055	0.00006	0.00092*	0.00083
AO	0.00012	0.00002	0.00037**	0.00007	0.00007
TC	0.00013	0.00015	0.00025	0.00013	0.00012
LS	0.00008	-0.00000	0.00033*	0.00007	0.00007
$\neq S$	-0.00292***	-0.00041	-0.00035	-0.00088***	-0.00071***
R-squared	0.0519	0.2468	0.3950	0.2808	0.2924
N	2499	2499	2499	2499	2499

References

Abeln, B. and Jacobs, J. P. (2015). Seasonal adjustment with and without revisions: A comparison of x-13arima-seats and camplet. Technical Report 25/2015, CAMA Working Paper. Available at SSRN: `https://ssrn.com/abstract=2635289` or `http://dx.doi.org/10.2139/ssrn.2635289`.

Baxter, M. and King, R. G. (1999). Measuring Business Cycles: Approximate Band-Pass Filters for Economic Time Series. *The Review of Economics and Statistics*, 81(4):575–593.

Cleveland, R. B., Cleveland, W. S., McRae, J. E., and Terpenning, I. (1990). STL: A Seasonal-Trend Decomposition Procedure Based on LOESS. *Journal of Official Statistics*, 6:3–73.

Cleveland, W. S. (1979). Robust locally weighted regression and smoothing scatterplots. *Journal of the American Statistical Association*, 74(368):829–836.

Cuevas, A. and Quilis, E. M. (2023). Seasonal adjustment methods for daily time series: A comparison by a monte carlo experiment. *SSRN Electronic Journal*.

Dagum, E. B. (1980). *The X-11-ARIMA Seasonal Adjustment Method*. Statistics Canada, Ottawa.

De Livera, A. M., Hyndman, R. J., and Snyder, R. D. (2011). Forecasting time series with complex seasonal patterns using exponential smoothing. *Journal of the American Statistical Association*, 106(496):1513–1527.

Esteban, C., Hyland, S. L., and Rätsch, G. (2017). Real-valued (medical) time series generation with recurrent conditional gans.

Findley, D. F., Monsell, B. C., Bell, W. R., Otto, M. C., and Chen, B. (1998). New capabilities and methods of the x-12-arima seasonal-adjustment program. *Journal of Business & Economic Statistics*, 16(2):127–52.

Franses, P. H., Paap, R., and Fok, D. (2005). Performance of seasonal adjustment procedures: Simulation and empirical results. Technical report, Econometric Institute, Erasmus University Rotterdam.

Goodfellow, I. J., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A. C., and Bengio, Y. (2014). Generative adversarial nets. In *Neural Information Processing Systems*.

Grether, D. M. and Nerlove, M. (1970). Some properties of "optimal" seasonal adjustment. *Econometrica*, 38(5):682–703.

Gómez, V. and Maravall, A. (2001). Seasonal adjustment and signal extraction in economic time series. In Peña, D., Tiao, G. C., and Tsay, R. S., editors, *A Course in Time Series Analysis*, chapter 8. John Wiley & Sons, New York.

Hamilton, J. D. (2017). Why you should never use the hodrick-prescott filter. Working Paper 23429, National Bureau of Economic Research, Cambridge, MA.

Harvey, A., Koopman, S. J., and Riani, M. (1997). The modeling and seasonal adjustment of weekly observations. *Journal of Business & Economic Statistics*, 15(3):354–368.

Hodrick, R. J. and Prescott, E. C. (1997). Postwar u.s. business cycles: An empirical investigation. *Journal of Money, Credit and Banking*, 29(1):1–16.

Isola, P., Zhu, J., Zhou, T., and Efros, A. A. (2016). Image-to-image translation with conditional adversarial networks. *CoRR*, abs/1611.07004.

King, R. G. and Rebelo, S. T. (1993). Low frequency filtering and real business cycles. *Journal of Economic Dynamics and Control*, 17(1):207–231.

Kuznets, S. (1929). Random events and cyclical oscillations. *Journal of the American Statistical Association*, 24:258–275.

McElroy, T. and Roy, A. (2022). A review of seasonal adjustment diagnostics. *International Statistical Review*, 90(2):259–284.

Mirza, M. and Osindero, S. (2014). Conditional generative adversarial nets. *CoRR*, abs/1411.1784.

Pedersen, T. M. (2001). The hodrick–prescott filter, the slutzky effect, and the distortionary effect of filters. *Journal of Economic Dynamics and Control*, 25(8):1081–1101.

Shiskin, J. (1978). Seasonal adjustment of sensitive indicators. In Zellner, A., editor, *Seasonal Analysis of Economic Time Series*, pages 97–103. U.S. Department of Commerce, Bureau of the Census, Washington, DC.

Slutzky, E. (1937). The summation of random causes as the source of cyclic processes. *Econometrica*, 5(2):105–146.

Smith, K. E. and Smith, A. O. (2020). Conditional gan for timeseries generation.

A Handling the Hodrick–Prescott Filter Endpoint Bias

The HP filter is widely used due to its advantageous properties. It can generate stationary time series when the data are integrated up to order four King and Rebelo (1993), it does not introduce any phase shift because of its symmetric design, and it avoids spurious cycles such as those produced by the Slutzky or Kuznets filters (Pedersen (2001)). In fact, the HP filter closely approximates an ideal high-pass filter that sharply removes components at frequencies below and above the bandpass cutoffs.

However, as shown by Hamilton (2017), the HP filter has some drawbacks. For example, it tends to leak unwanted frequencies and suppress components it should preserve. Moreover, the filtered values at the sample endpoints often differ substantially from those in the middle, because the filter applies a symmetric infinite-dimensional moving average to a finite time series. Some authors even advise removing the first and last observations to minimize this endpoint bias Baxter and King (1999); however, this solution is not feasible for small time series, and particularly the last observations are of greatest interest.

To reduce the endpoint bias in the HP filter, the time series is extended using fore- and backcasting. To evaluate the performance of this approach, I generated synthetic time series Y_i (with $i \in \{1, \ldots, N\}$ and $N = 1,000$) that exhibit the same characteristics as those used for training the cGAN. Each series Y_i has a length of $t = 10,000$; from each series, a 120-observation middle segment y_i was extracted. The SARIMA(11,1,0)(1,1,0,12) model was then fitted to y_i to forecast the next 24 values. For backcasting, y_i was reversed and forecasted in a similar manner. The resulting forecasts and backcasts were appended to the end and beginning of y_i, respectively, to form the extended series y_i^*.

Next, the HP filter is applied to the extended series y_i^* to produce the adjusted cycle c_i^*, and it is applied to the original segment y_i to generate the unadjusted cycle c_i. A "true" cycle C_i is derived from the full series Y_i.

Figure 10 displays the average absolute error for both approaches, defined as

$$ME_t = \frac{1}{N} \sum_{i=1}^{N} |C_{it} - c_{it}| \tag{23}$$

for the standard method and

$$ME_t^{\text{ext}} = \frac{1}{N} \sum_{i=1}^{N} |C_{it} - c_{it}^*| \tag{24}$$

for the extended method. The effectiveness of this approach in reducing the HP filter's endpoint bias naturally depends on the predictability of the time series. In

the simulation, the mean error over all 120 time points for the standard method was 0.0151, whereas the extended method achieved a mean error of 0.0117, representing an error reduction of approximately 22%. Nonetheless, endpoint bias still exist even after applying the extended method.

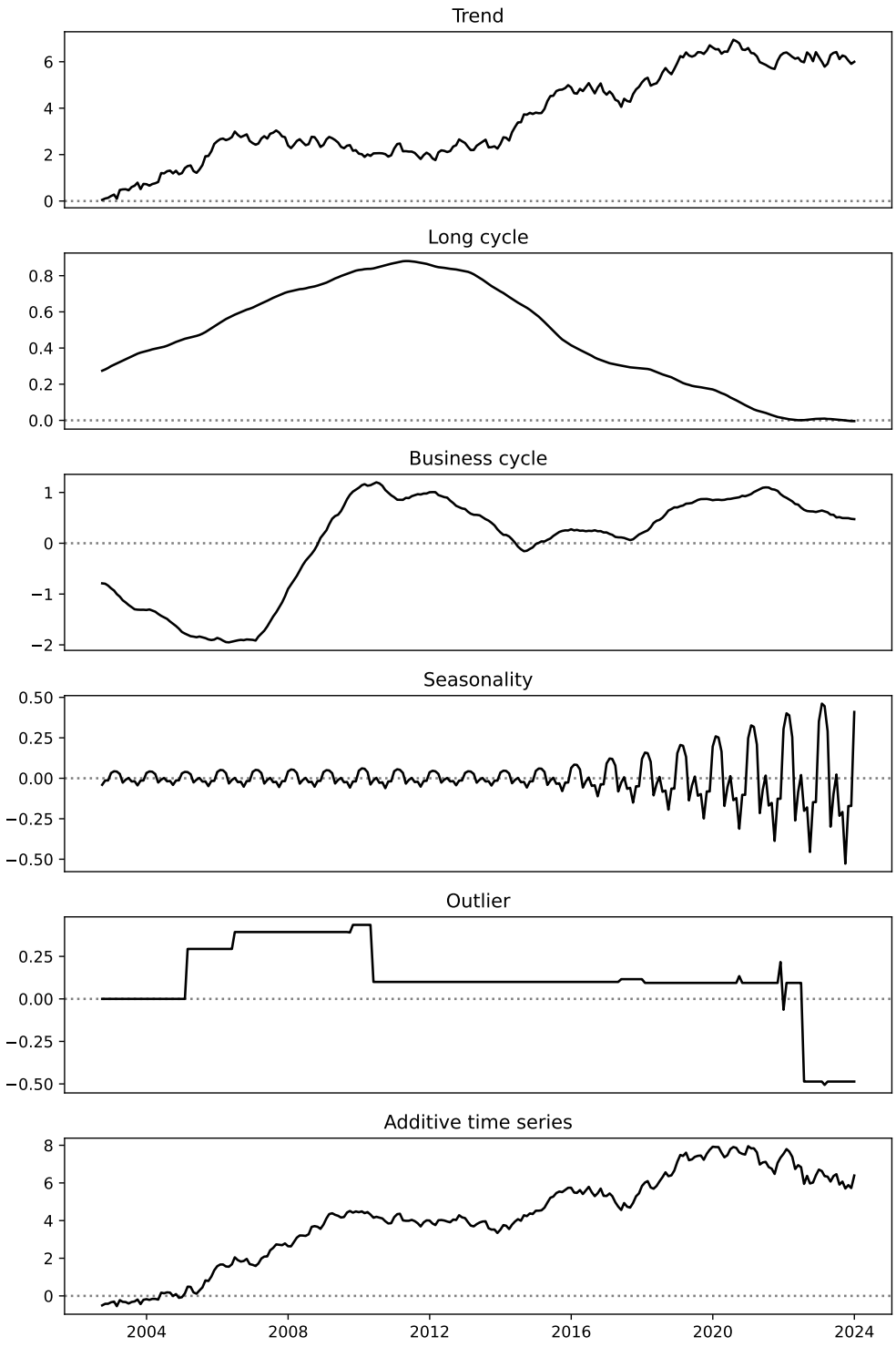
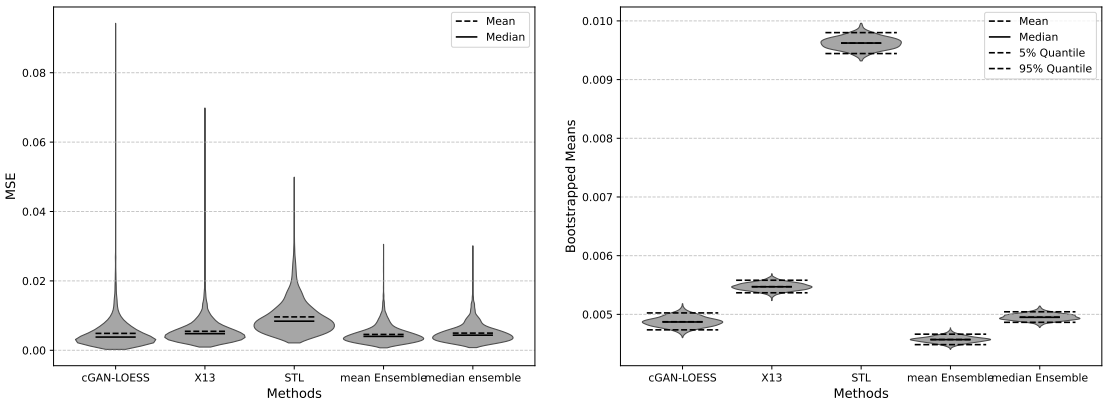


Figure 3: Synthetic time series components



(a) MSE (b) bootstrapped MSE means

Figure 4: Comparison of MSE distributions for different methods.

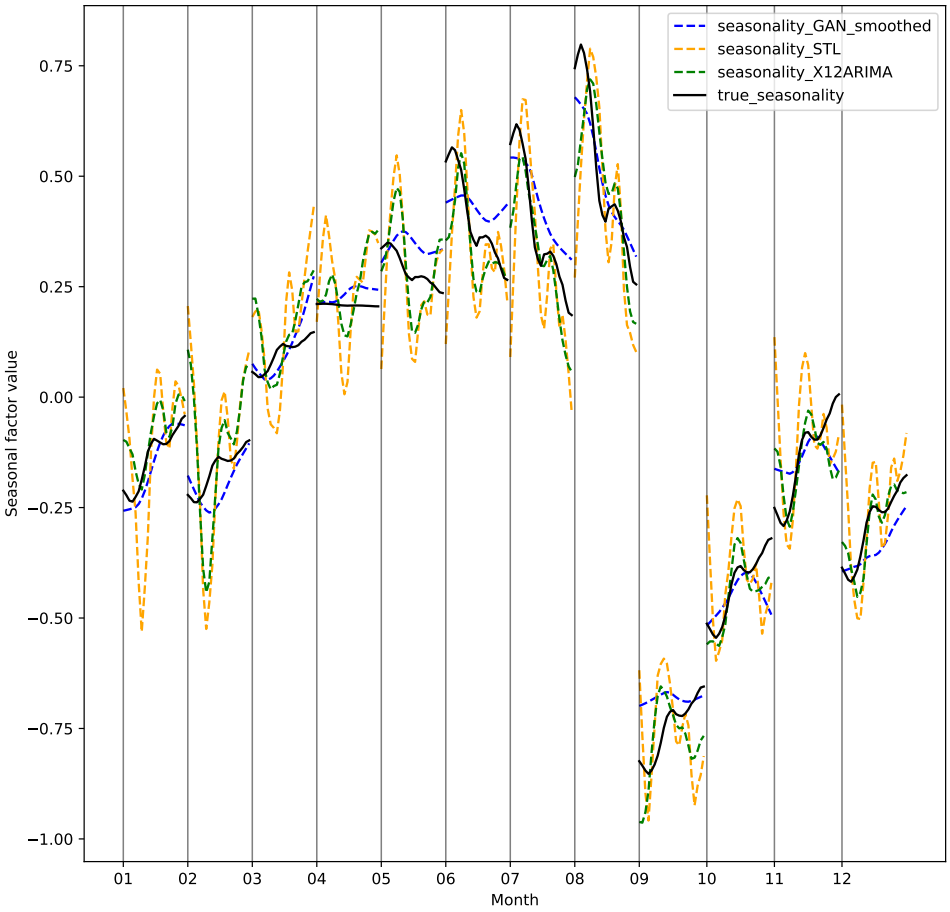


Figure 5: Seasonal subseries for base methods

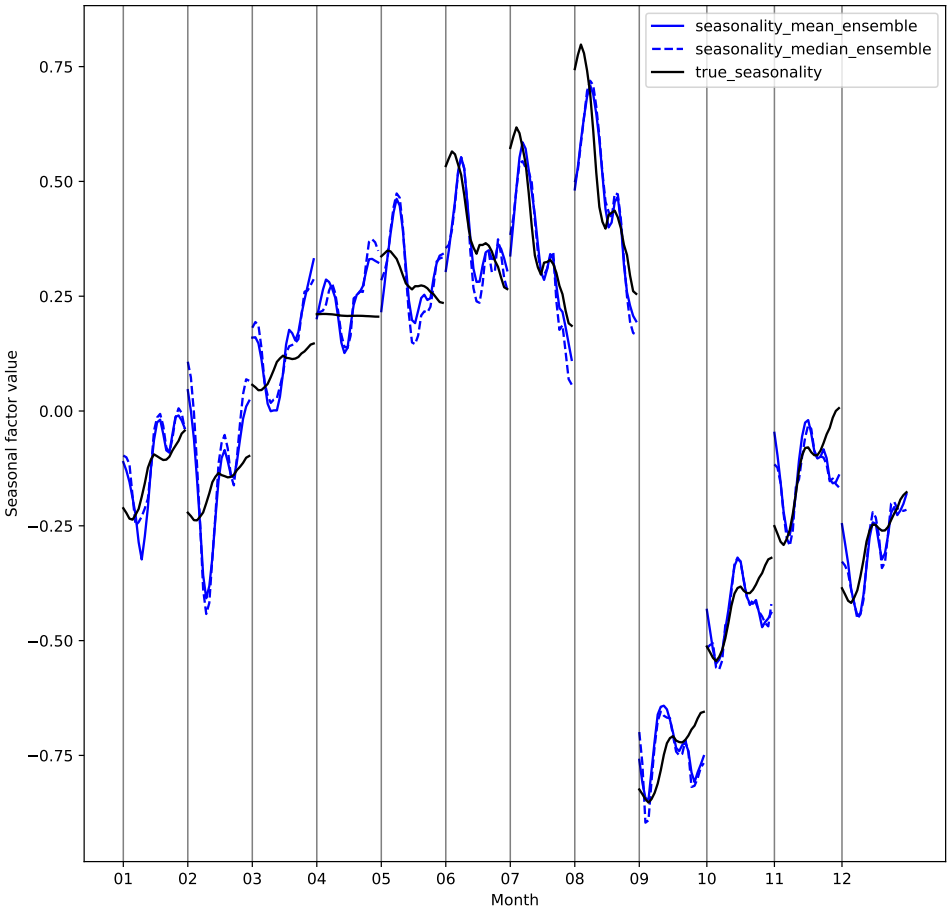


Figure 6: Seasonal subsseries for ensemble methods

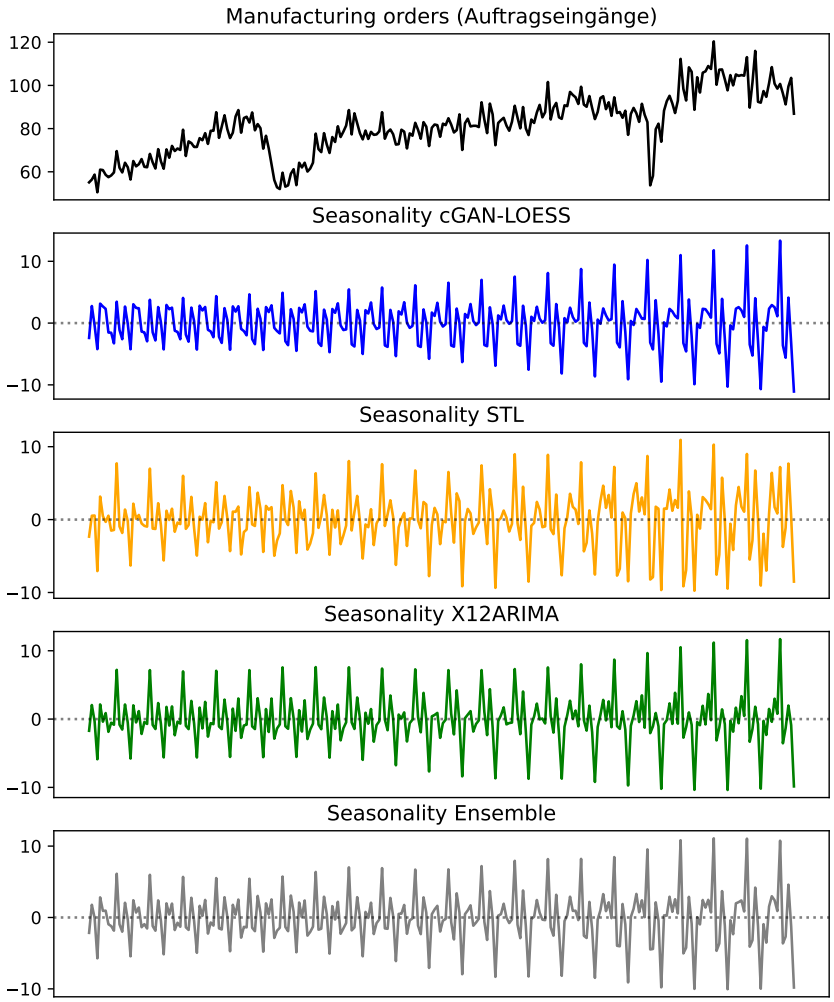


Figure 7: Time series for seasonal component (Auftragseingänge)

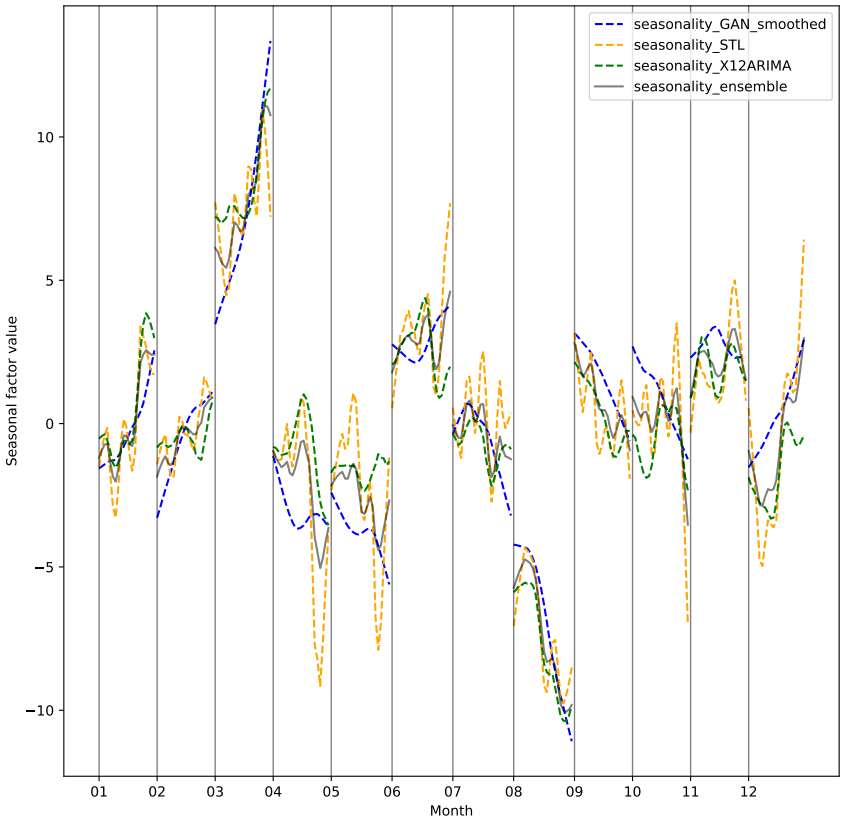


Figure 8: Seasonal subseries (Auftragseingänge)

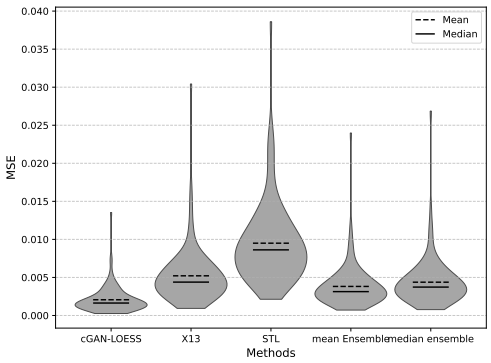


Figure 9: Violin plots of MSE for time series with zero seasonality

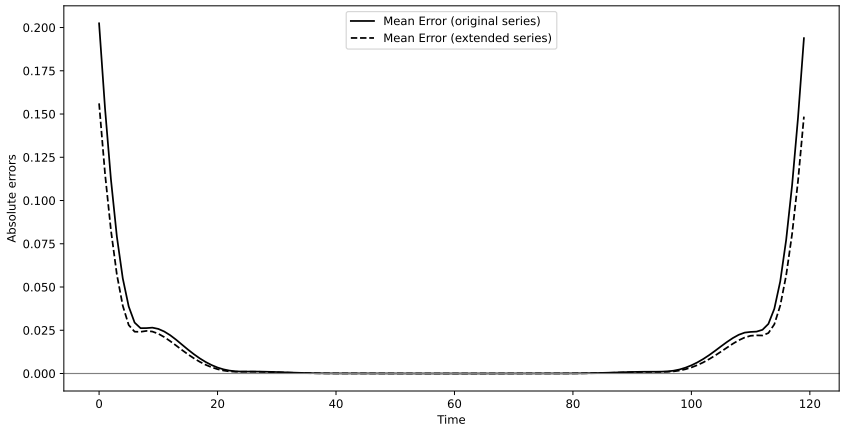


Figure 10: Hodrick-Prescott endpoint error simulation