Beer B1

Predicting Beer Production for the next month

Jussi Juvonen, Niko Laurén, Isuru Mulle Gamage

Time series exploration with visualization and decomposition of trend, seasonality, and residuals

The time series data represents monthly beer production in Australia, with no missing months or duplicate timestamps. The dataset spans from January 1956 to August 1995. Figure 1 shows a Seasonal-Trend decomposition using LOESS (STL) analysis of the series.

The observed data shows annual seasonality and a gradual long-term increase in production until the late 1970s, followed by a slight decline. The trend component captures the long-term pattern, and both robust and non-robust decompositions produce nearly identical trends, indicating stable estimates with minimal outlier influence. The seasonal component shows a consistent, repeating yearly pattern with roughly constant amplitude. The residuals fluctuate around zero, which means that most systemic variation is explained by the trend and seasonal terms.

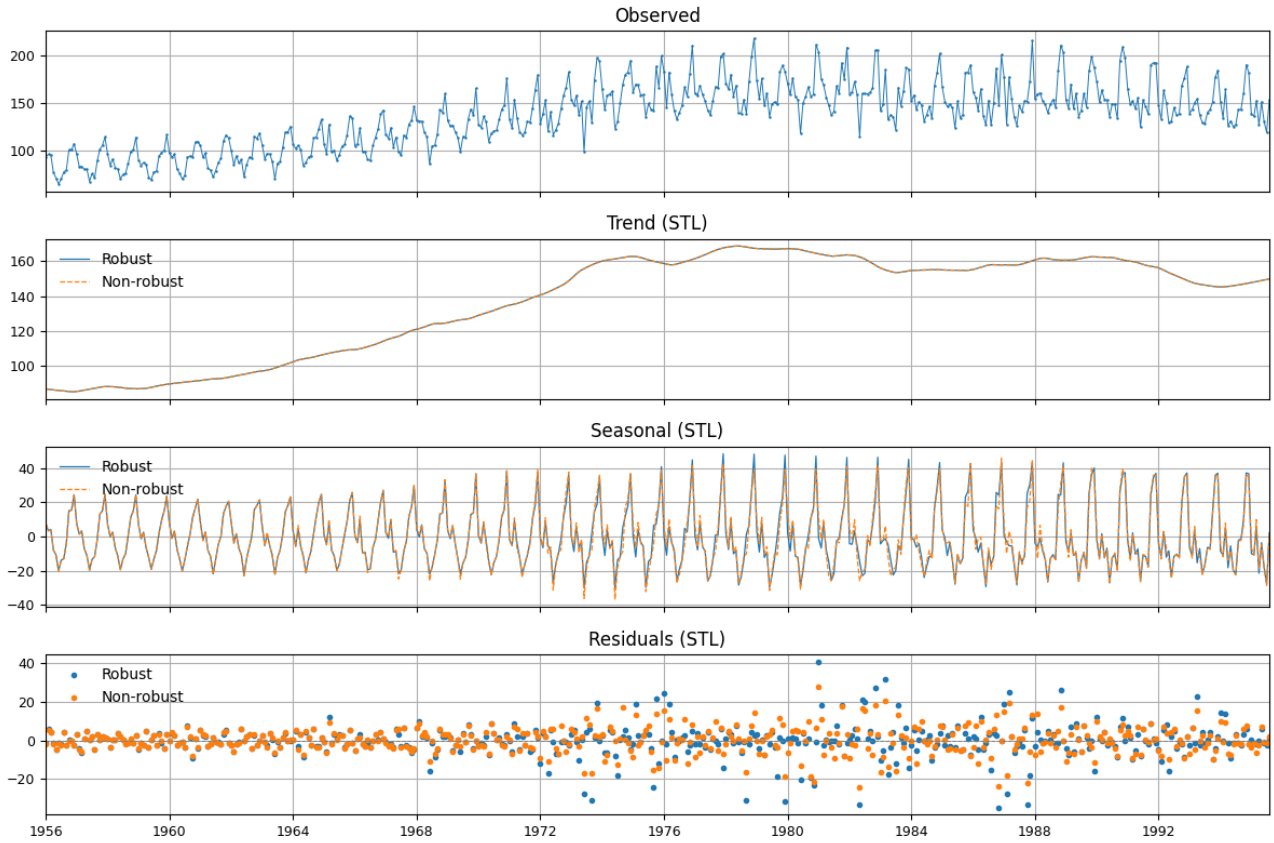

Figure 1: STL decomposition of monthly beer production time series showing the observed data, long-term trend, seasonal component, and residuals.

Figure 2 visualizes the seasonal component from the STL decomposition grouped by calendar month. The boxplots show a cyclical pattern in beer production, with the lowest seasonal values occurring during winter months (June - July) and the highest during summer (November - December). The relatively small spread of most boxes suggests that this seasonal pattern is stable across years, with only minor year-to-year variation.

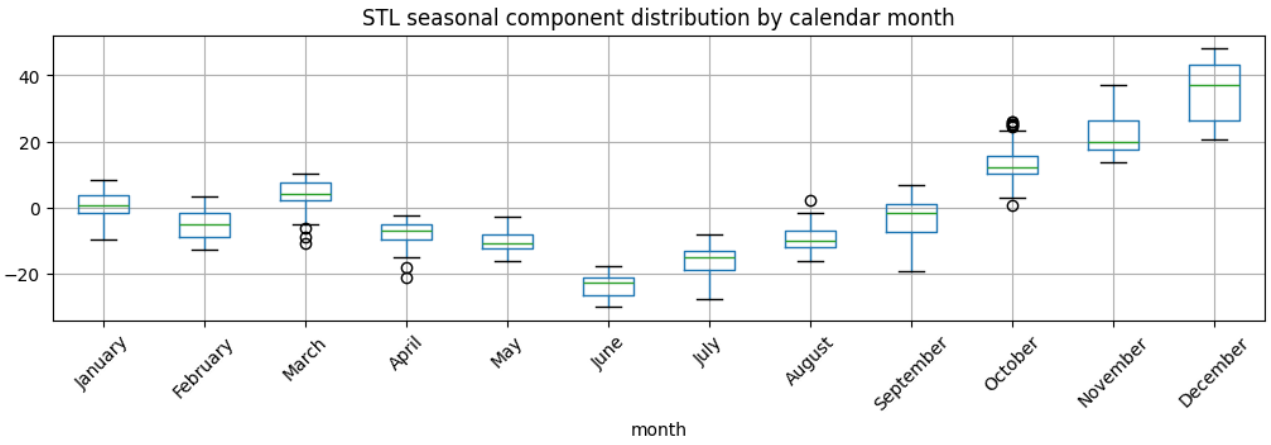

Figure 2: Distribution of the STL seasonal component by calendar month.

Autocorrelation analysis

Figure 3 shows the autocorrelation structure of the time series. The Autocorrelation (ACF) plot displays a strong positive correlation that gradually decays over many lags, with a noticeable seasonal pattern repeating approximately every 12 months, confirming yearly seasonality in the data. The Partial Autocorrelation (PACF) plot shows significant spikes at lag 1 and around lag 12. This suggests short-term dependence and annual cyclical effects. Together the plots confirm that beer production is highly autocorrelated and strongly seasonal over time.

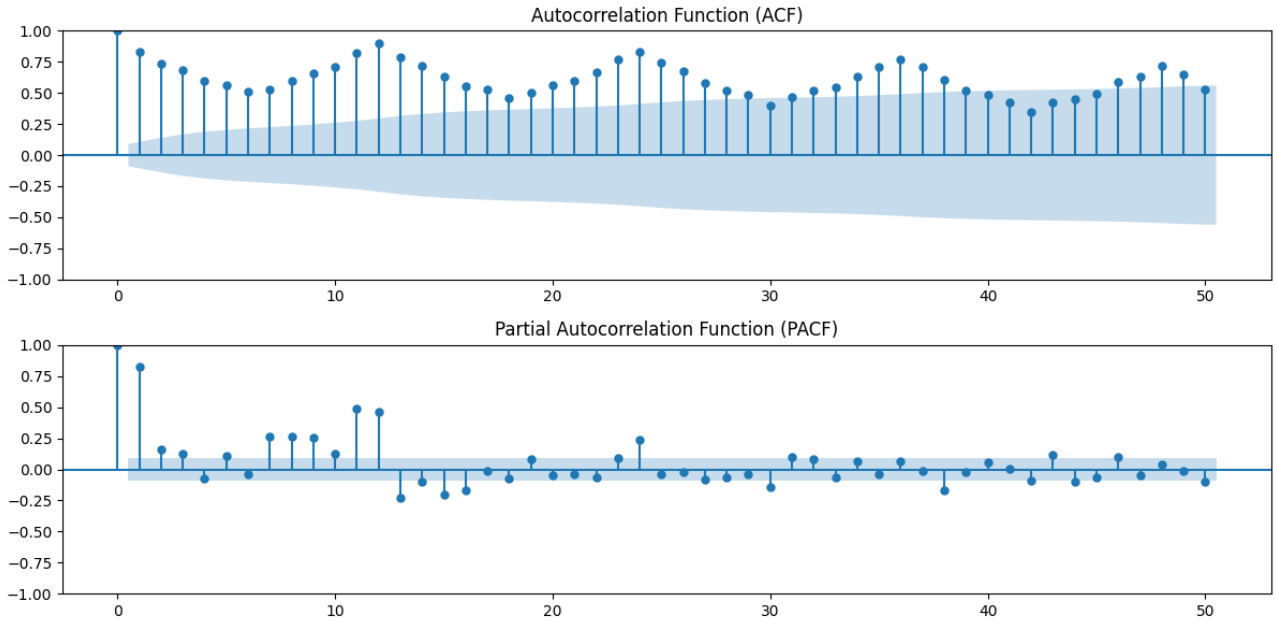

Figure 3: ACF and PACF plots of monthly beer production

Partitioning time series data

The dataset is partitioned chronologically into three subsets. The training subset is ~70% of the data covering period from January 1956 to September 1983 consisting of 333 samples. This subset is used to train the model and identify seasonal patterns in beer production. The validation subset is the following 71 samples or ~15% of the data from October 1983 to August 1989. The validation subset is used to adjust and fine-tune the parameters of the model. The remaining 72 samples (~15%) from September 1989 to August 1995 is for the test subset. The test subset is used to evaluate the model's predictive performance on unseen future data.

Data preprocessing and quality assessment

The dataset consists of uninterrupted monthly observations from January 1956 to August 1995, showing no missing or duplicate timestamps. The measurements are uniformly sampled at a constant monthly frequency. For irregular series, a uniform rate can be restored by resampling or interpolation. For missing data, short gaps can be interpolated, while longer gaps can be addressed through model-based smoothing or segment-wise estimation that preserves the temporal structure, such as Kalman filtering or spline-based local regression.

Only one variable, monthly beer production, is recorded, making temporal alignment across variables unnecessary. In multivariate settings, synchrony can be achieved by aligning all variables to a shared time index and interpolating values where data is missing or offset.

Outlier identification using STL decomposition

STL decomposition first separates the time series into trend, seasonal, and residual components. After removing the systematic trend and seasonality, the remaining residuals capture irregular variations in the data. Outliers can then be detected by identifying residuals

that deviate strongly from zero using statistical measures such as standard deviation or median absolute deviation. These points are usually flagged rather than removed to preserve the temporal structure.

Preparing Long Time Series for LSTM Forecasting: Sub Sequences, Seasonality, and Trend Handling

Dividing a long time series into shorter sub sequences is a standard preprocessing step for LSTM models. Instead of using the full sequence, the data is transformed into many fixed length sliding windows, each containing past observations and a future prediction target. This approach increases the number of training samples and helps the model learn local temporal patterns more effectively (Hewamalage et al., 2021). [1]

Seasonality strongly influences how these windows should be constructed. If the data contains annual or monthly seasonal cycles, the window length must be long enough to cover at least one full seasonal period; otherwise, the LSTM cannot learn recurring patterns (Lemke & Gabrys, 2019) [2]. Seasonal decomposition methods, such as STL, may also be used before windowing to simplify the series (Hyndman & Athanasopoulos, 2021). [3]

LSTMs can model trend and seasonality directly from raw sequences or through explicit preprocessing, such as deseasonalization or adding seasonal indicators as extra features (Bandara et al., 2020) [4].

LSTM standardization methods

Long Short Term Memory (LSTM) has two typical standardization methods called Min-Max normalization and Z-score standardization. Min-Max transforms data into a fixed range and it works well with sigmoid and tanh functions. Z-score centers the data around zero mean with unit variance and it's suitable for dealing with outliers and varying feature scales. Min-Max produces lower Round Mean Square Error (RMSE) in LSTM-based multivariate time series forecasting when compared to Z-score. Min-Max enhances training stability and prediction accuracy in LSTM models. [5]

Baseline model

A autoregressive baseline model was made by converting the series into sliding windows of length $T = 12$ months. Each sample uses the previous 12 monthly observations to predict the next month. The model itself is a single layer linear predictor mapping the 12-dimensional input to scalar forecast and trained to minimize mean squared error on the training set. The data was split so that all samples up to September 1983 were used for training and in the remaining months formed a single holdout set for validation.

In Figure 4, the training and validation losses of the baseline autoregressive model are shown across 40 epochs. The training loss decreases rapidly during the first 10 epochs and then gradually stabilizes. The validation loss follows a similar trajectory but begins to plateau and fluctuate after roughly 20-30 epochs, suggesting some mild overfitting. Both losses remain within the same order of magnitude, implying that the model generalizes reasonably well given its simplicity.

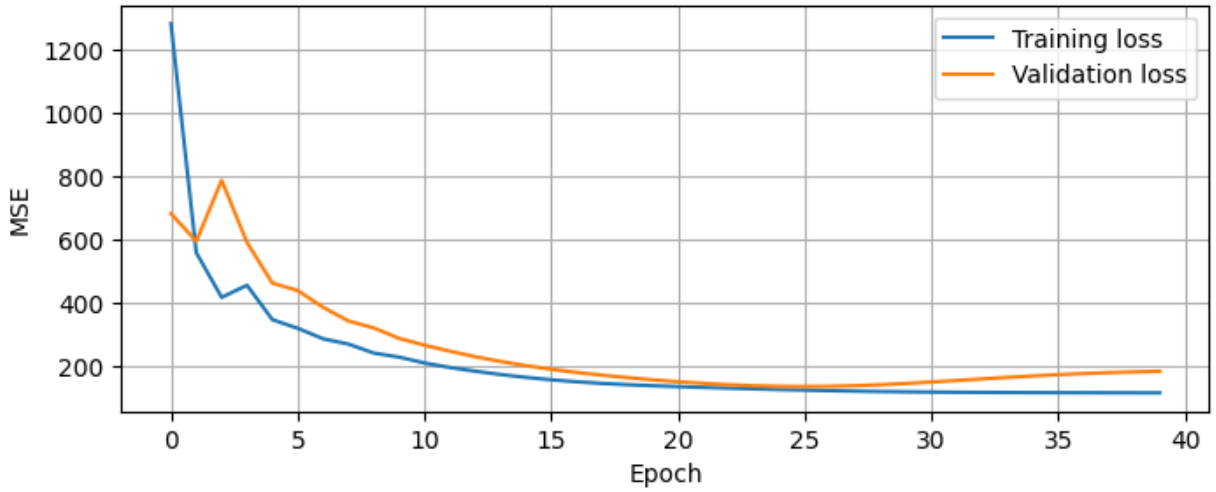

Figure 4: Training and validation MSE across epochs for the autoregressive baseline model.

In Figure 5, the iterative forecast is compared against the actual validation target over time. The iterative forecast shows moderate accuracy at the beginning of the validation data but tends to drift as the predictions progress. This drift might happen because each step's prediction is recursively fed back as input, allowing small errors to accumulate. The model occasionally underestimates or overestimates the overall trend, resulting in systematic bias near the end of the forecast. In this run the forecast overestimates the trend and achieved $MAE \approx 69.3$ and $RMSE \approx 82.3$, suggesting that while the baseline model captures the long-term pattern, its simplicity limits its ability to maintain stable trend estimation across multiple forecasted steps.

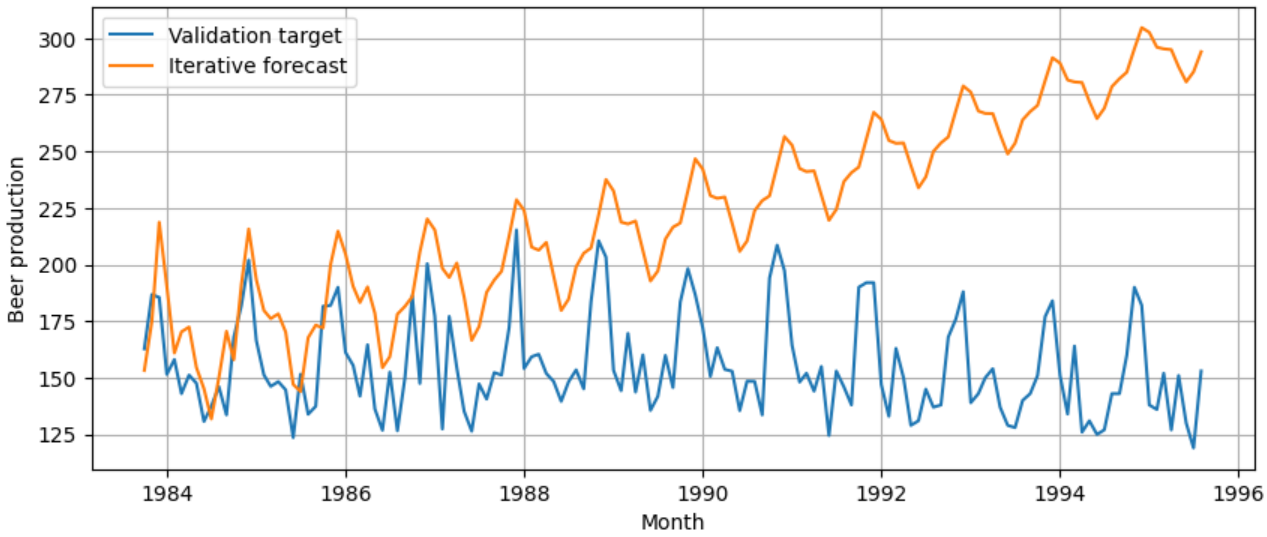

Figure 5: Iterative multi-step forecast compared with the validation target for the baseline autoregressive model.

References

Bandara, K., Bergmeir, C. & Smyl, S. (2020), "Forecasting across time series databases using recurrent neural networks on groups of similar series: A clustering approach", Expert Systems with Applications, 140, pp. 112896.

Bianchi, F.M., Livi, L., Rizzi, A. & Sadeghian, A. (2022), Recurrent Neural Networks for Short-Term Load Forecasting, Springer.

Fawaz, H.I., Forestier, G., Weber, J., Idoumghar, L. & Muller, P.A. (2019), "Deep learning for time series classification: A review", Data Mining and Knowledge Discovery, 33(4), pp. 917–963.

Hewamalage, H., Bergmeir, C. & Bandara, K. (2021), "Recurrent neural networks for time series forecasting: Current status and future directions", International Journal of Forecasting, 37(1), pp. 388–427.

Hyndman, R. & Athanasopoulos, G. (2021), Forecasting: Principles and Practice, 3rd edn., OTexts.

Lemke, C. & Gabrys, B. (2019), "Meta-learning for time series forecasting and forecast combination", Neurocomputing, 353, pp. 1–16.

Pranolo, A., Setyaputri, F.U., Paramarta, A.K.I., Triono, A.P.P., Fadhilla, A.F., Akbari, A.K.G., Putra Utama, A.B., Prasetya Wibawa, A. & Uriu, W. (2024) 'Enhanced Multivariate Time Series Analysis Using LSTM: A Comparative Study of Min-Max and Z-Score Normalization Techniques', ILKOM Jurnal Ilmiah, 16(2), pp. 210-220. DOI: 10.33096/ilkom.v16i2.2333.210-220.