Modelling TSA Checkpoint Data

Abstract

Following an analysis of TSA checkpoint data from 2019 to 2026 in the frequency domain and using STL decomposition, we find strong evidence of both annual and weekly seasonality in the data. Aggregating by week to predict annual trends, we fit an ARIMA(3,1,0) model to the seasonally adjusted log data, and a SARIMA(2,1,1) model to the log data. We find that the ARIMA(3,1,0) model on the seasonally adjusted data outperforms the SARIMA(2,1,1) model on the log data, with better forecasting ability.

1 Introduction

Airports are responsible for much of the long-distance commercial travel foundational for the function of a healthy economy. As such, the commercial throughput of a country's airports stands as a quantity of interest.

When it comes to planning airport logistics, flight tracking, or security adjustments, detecting repeating patterns in travel and aiport activity is paramount to ensuring optimal operational adjustments can be made.

This begs the natural question - what sort of data can one use to analyze such a quantity? Data on the number of flights alone is likely insufficient, as noted in past project comments ("Pre-Covid u.s. Airline Traffic Analysis" 2025a), as it is important to consider passenger flights independently of cargo flights. Thus, we focus on the number of people passing through Transportation Security Administration (TSA) checkpoints daily. The TSA checkpoint data serves as a very close proxy for the number of daily airline passengers, which may or may not double-count those with layovers, depending on the airport's policies regarding bag checks.

In this project, we analyze daily TSA checkpoint data ranging from 1/1/2019 through 2/10/2026, taken from the TSA's website (Administration 2026). We analyze this data using power spectral density estimation and STL decomposition. From here, we determine that the weekly fluctuations are strong and periodic, and thus employ weekly aggregation to explore the annual trends specifically. We fit an ARIMA model to the de-seasonalized log data as well as a SARIMA model to the log data. We attempt to answer questions regarding seasonality and trends present. Practically, we set out to answer the question of whether an ARIMA model on the seasonally adjusted log data can outperform a SARIMA model on the log data.

2 Methods

2.1 Power Spectral Density Estimation (PSD)

We estimated the PSD by squaring the absolute value of the FFT of the data (Ionides 2026, Lecture 7). To mitigate spectral leakage and increase visual resolution, we detrended the data, applied a Hanning window, and zero-padded to 2^{16} points. The PSD was only taken for post-pandemic data (after 2022-01-01) to eliminate the disruptive effects of the COVID-19 pandemic on the spectrum. We identified peaks using the `scipy.signal.find_peaks` function, with a minimum height threshold of 10% of the maximum PSD value to focus on the most prominent peaks. We then computed the corresponding frequencies and periods for these peaks to interpret the periodicities in the data.

2.2 Short-Time Fourier Transform (STFT)

To analyze how the frequency content of the data evolves over time, we implemented a STFT ("Short-Time Fourier Transform" 2026). We used a window size of 91 days (13 weeks) to capture seasonal patterns, with a step size of 7 days to ensure weekly resolution. Each window was detrended, windowed with a Hanning window, and zero-padded to 2048 points. The PSD was estimated by squaring the absolute value of the FFT of each window. The results were visualized as a heatmap, or spectrogram, with time on the x-axis, frequency on the y-axis, and color representing the magnitude of the FFT (square root of the PSD). This was done for the entire dataset to capture the evolution of frequency content over the full time span, including the pandemic period.

2.3 Spectral Entropy

To quantify the complexity of the frequency content over time, we computed the spectral entropy for each window. We divided the power of each frequency bin by the total power to get a probability distribution, then computed the Shannon entropy of this distribution. We then normalized the entropy by dividing by $\log(N)$, where N is the number of frequency bins, to ensure the spectral entropy ranges from 0 to 1 (Wikipedia contributors 2025).

$$p_k = \frac{PSD_k}{\sum_{j=1}^{N} PSD_j} \qquad \hat{H} = -\frac{1}{\log(N)} \sum_{k=1}^{N} p_k \log(p_k)$$

$\hat{H} = 0$ indicates that all power in concentrated in a single frequency bin, while $\hat{H} = 1$ indicates that the power is uniformly distributed across all frequency bins. A higher spectral entropy indicates a more complex frequency content, while a lower value indicates a more regular, periodic signal.

2.4 STL decomposition (trend + seasonality + remainder)

STL, shorthand for (Seasonal-Trend decomposition using LOESS ("Seasonal-Trend Decomposition Using LOESS (STL)" 2026)) is a framework that decomposes a time series into three components: trend (smooth long-run movement), seasonality (recurring periodic pattern), and the remainder / Residual (irregular component left after removing trend and seasonality).

In additive form, STL decomposition is written as $y_t = T_t + S_t + R_t$ where y_t is the observed series, T_t is the trend component, S_t is the seasonal component, and R_t is the remainder (residual) component. In the `statsmodels` STL implementation, these are returned as `trend`, `seasonal`, and `resid`. The method uses **LOESS** (locally estimated scatterplot smoothing) to obtain smooth estimates of the components. After STL decomposition, a common object for downstream ARIMA modeling is the seasonally adjusted series, obtained by removing the seasonal component:

$$y_t^{(\mathrm{sa})} = y_t - S_t = T_t + R_t$$

We applied STL decomposition to the weekly aggregated data, as the PSD analysis revealed strong seasonality. We used a 52 week period to capture annual seasonality.
Once deseasonalized, we fit an ARIMA model to the seasonally adjusted log data, and a SARIMA model to the log data, to compare the performance of these two approaches.

3 Likelihood Ratio Test for ARIMA Models

To compare nested ARIMA models, we used a likelihood ratio test (Ionides 2026, chap. 5). The test statistic is calculated as:

$$\Lambda = 2\left(\log L_{\text{larger}} - \log L_{\text{smaller}}\right)$$

where L is the likelihood of the model. Under the null hypothesis that the smaller model is correct, Λ follows a χ^2 distribution with degrees of freedom equal to the difference in degrees of freedom between the two models. We used this test to compare the best ARIMA(3,1,0) model with ARIMA(2,1,0), ARIMA(4,1,0), and ARIMA(3,1,1) to ensure that we were not overfitting by including unnecessary parameters.

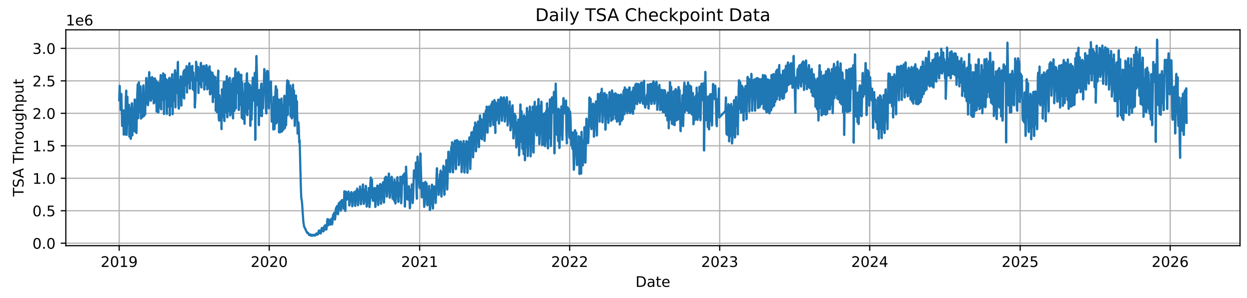
4 Ljung-Box Test

The Ljung-Box test checks whether a time series (or model residuals) has autocorrelation across a set of lags ("Monthly Air Passengers(1949 - 1960) Time Series Analysis and Modelling" 2016). It tests the null hypothesis that the autocorrelations up to lag h are jointly zero. In ARIMA, it is typically applied to the residuals to assess whether the fitted model has removed serial dependence. The Null hypothesis is that no serial correlation up to lag h, while the alternative hypothesis is that at least one autocorrelation up to lag h is nonzero. The Ljung–Box statistic is

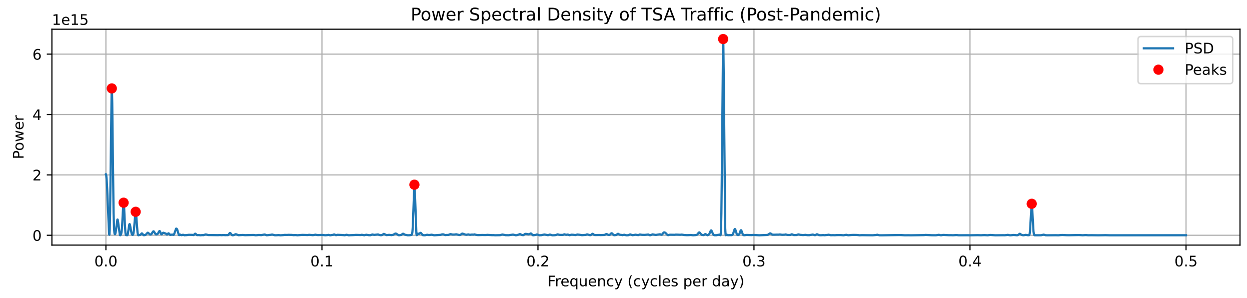
$$Q = n(n+2)\sum_{k=1}^{h} \frac{\hat{\rho}_k^2}{n-k},$$

where n is the sample size, $\hat{\rho}_k$ is the sample autocorrelation at lag k, and h is the maximum lag being tested. Under the null hypothesis (asymptotically), $Q \sim \chi^2_h$. So for significance level α, we reject the null when $Q > \chi^2_{1-\alpha,\,h}$.

5 Exploratory Data Analysis

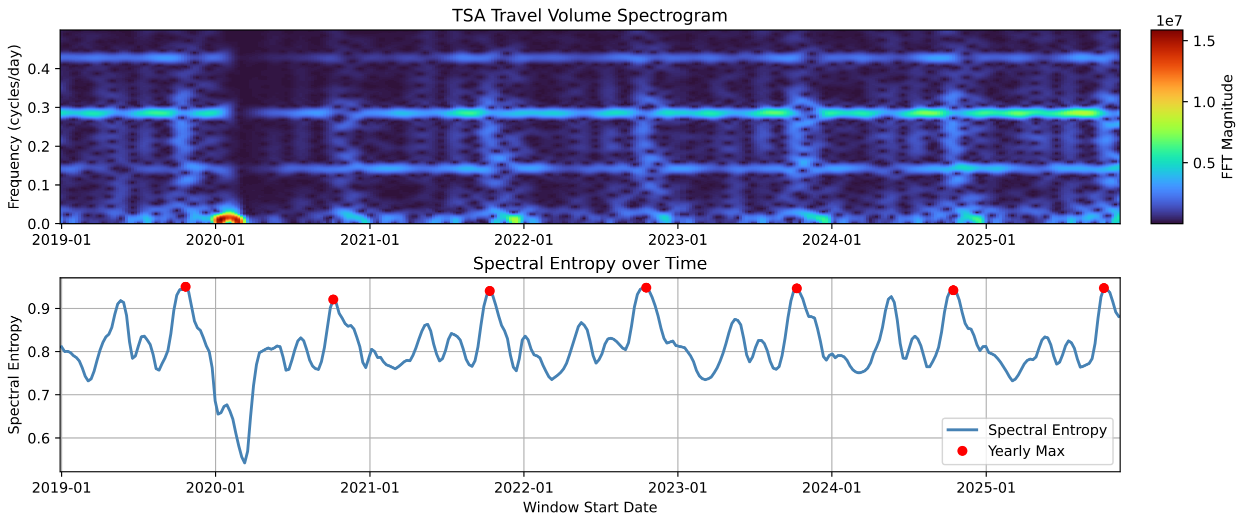


The data appears highly seasonal, with a clear annual pattern that is interrupted by the COVID-19 pandemic in early 2020. Post-pandemic, there appears to be a repeating annual pattern that slowly trends upwards. This inuitively makes sense, as we would have expected the pandemic to have disrupted normal travel patterns, with travel gradually recovering over time.



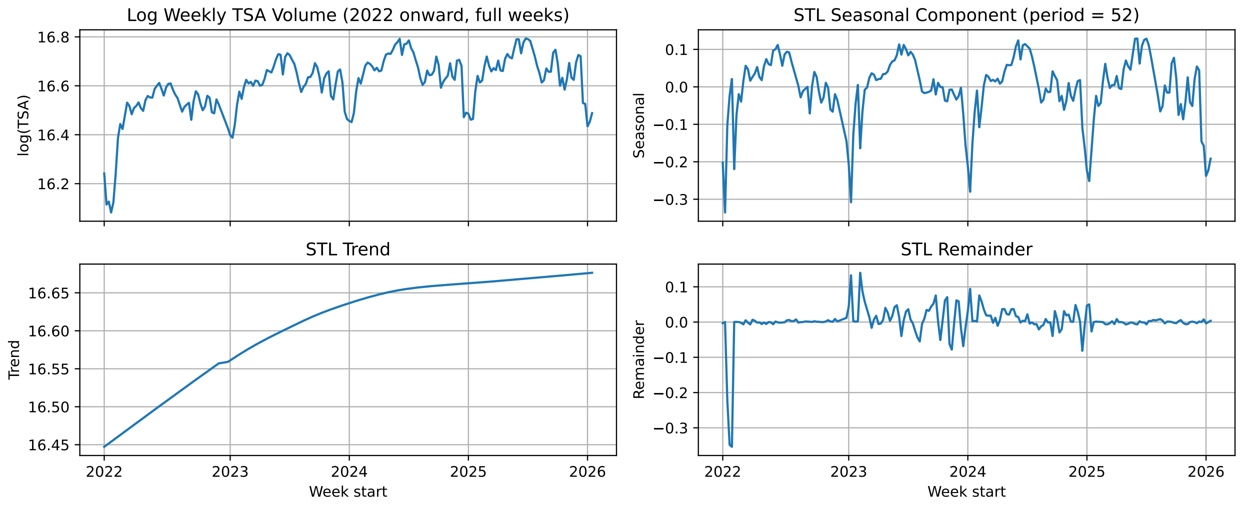
	0	1	2	3	4	5
Frequency (cycles/day)	0.002731	0.008194	0.013779	0.142807	0.285721	0.428574
Period (days)	366.122905	122.040968	72.575858	7.002458	3.499920	2.333321

The power spectral density (PSD) of the post-pandemic data is exceptionally clean. Peaks 0, 1, and 2 correspond to the annual cycle, with 1 and 2 being the 1/3 and 1/5 harmonics, respectively. Peak 3 corresponds to the weekly cycle, with peaks 4 and 5 being the 1/2 and 1/3 harmonics, respectively. The presence of these peaks confirms the strong seasonality in the data, with both annual and weekly patterns clearly present. Of particular note is the height of peak 4, which is the highest peak in the spectrum. This indicates a significant non-sinusoidal component to the weekly pattern ("Pre-Covid u.s. Airline Traffic Analysis" 2025a).



```
Max Spectral Entropy Dates (one per year):
 10-22; 10-06; 10-12; 10-18; 10-10; 10-15; 10-07;
```

The spectrogram is broadly very consistent with the PSD analysis. We see strong horizontal bands corresponding to the weekly cycles, with the annual cycles missing due to the short window length. There is a vertical blank band corresponding to the COVID-19 pandemic. On the spectral entropy plot, we see an annual cycle with maximum entropy in mid-October. Correspondingly, we see fuzzy vertical bands in the spectrogram. These windows with higher entropy encompass the holiday season, spanning until mid-January. Both of these patterns are likely the result of increased variability in travel patterns during the holiday season, with people flying to visit family or take vacations. The complexity of the frequency domain appears to fluctuate consistently and periodically.



To explore the annual trends in the time domain, we can examine the STL decomposition of the weekly TSA data post 2022. Here, we can see that the STL decomposition demonstrates the strong annual trend as well as an overall upwards trend. We see that the remainders are not white noise, and still contain some sort of serial dependence, as shown by the Ljung-box test in the supplementary material (under STL residuals). Next, we will fit an ARIMA model to the seasonally adjusted log data, and a SARIMA model to the log data, and compare the two.

6 Results

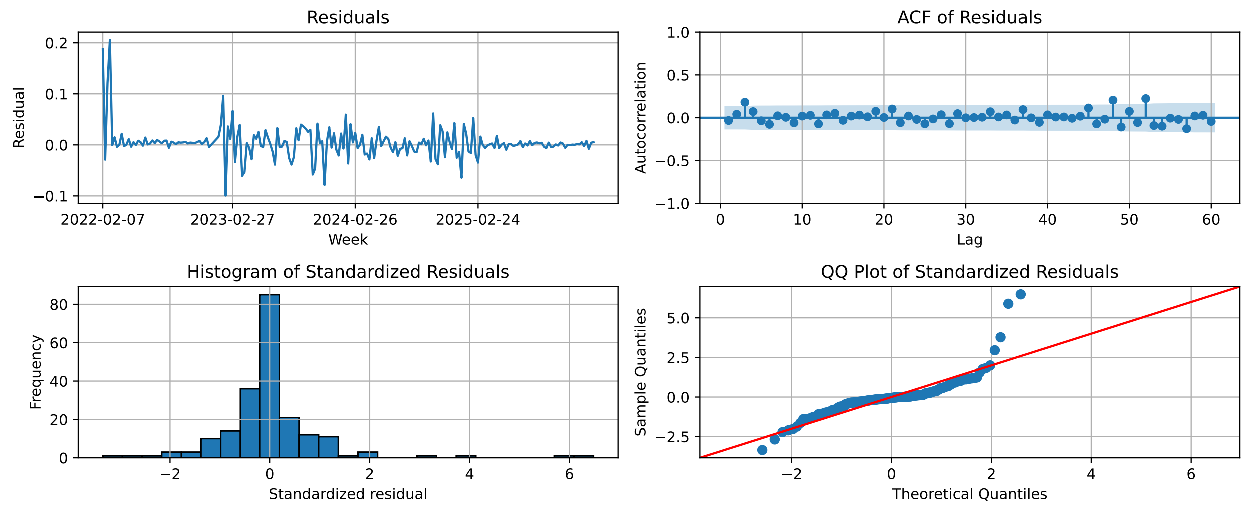
6.0.1 ARIMA and SARIMA Modeling

We begin by building an AIC table for ARIMA ("Autoregressive Integrated Moving Average" 2025) to determine the best ARIMA model for the data.

```
        MA0     MA1     MA2     MA3     MA4     MA5
AR0 -729.01 -727.67 -762.35 -799.67 -803.20 -811.06
AR1 -727.34 -726.42 -768.79 -800.40 -807.06 -807.67
AR2 -746.63 -786.66 -792.56 -809.49 -810.44 -809.44
AR3 -813.60 -811.71 -810.16 -810.11 -808.54 -807.06
AR4 -811.69 -809.74 -808.96 -808.67 -806.84 -805.18
AR5 -810.05 -808.20 -806.96 -798.00 -805.13 -802.69
```

Here we can see that the best ARIMA(p,d,q) for the seasonal-adjusted log data is ARIMA(3,1,0) with an AIC of -813.60. However, to avoid overfitting, we will also use a likelihood ratio test to compare this model with ARIMA(2,1,0), ARIMA(4,1,0), and ARIMA(3,1,1).

We note that in ARIMA(2,1,0) vs ARIMA(3,1,0): (LR = 68.98), (p < 10^{-16}). This strongly supports including the third AR term. Additionally, in ARIMA(3,1,0) vs ARIMA(4,1,0): (LR = 0.087), (p = 0.768) This provides no evidence that adding a fourth AR term improves fit. Finally, in ARIMA(3,1,0) vs ARIMA(3,1,1): (LR = 0.102), (p = 0.75). This provides no evidence that adding an MA(1) term improves fit. Thus, we proceed with fitting ARIMA(3,1,0), and a residual analysis.



The residuals visually appear to be uncorrelated, and the QQ plot of standardized ARIMA(3,1,0) residuals shows only rough alignment with the reference line in the middle quantiles, with clear departures in both tails, especially the upper tail. This indicates substantial nonnormality in the residual distribution, with heavy tails and right skewed extreme values.

There are several unusually large positive shocks and a smaller number of large negative shocks. This pattern is consistent with occasional abrupt changes in TSA checkpoint volume that are not fully captured by a Gaussian error assumption. In fact, we observe that 19 of the top 25 largest absolute residuals fall within the high-entropy holiday season portion of the year (October through February). This suggests that the increased variability in travel patterns during the holiday season may be contributing to these large residuals, and that the ARIMA(3,1,0) model may struggle to capture this heightened complexity during these periods. There are several unusually large positive shocks and a smaller number of large negative shocks. This pattern is consistent with occasional abrupt changes in TSA checkpoint volume that are not fully captured by a Gaussian error assumption.

This result does not contradict the Ljung-Box diagnostics. The Ljung-Box tests (see below) indicate that the ARIMA(3,1,0) model adequately removes serial dependence in the mean, while the QQ plot shows that the residuals remain non-normal in distribution.

Next, we fit a SARIMA model (Ionides 2026, chap. 6) to the log data, to see if we can capture the seasonality better than STL decomposition. We find that the optimal SARIMA model is SARIMA(2,1,1). The full grid search, alongside other diagnostics such as the QQ plot, can be found in the supplemental materials.

Confirming what we observed visually with the ARIMA model, the Ljung-Box test results show that the SARIMA model residuals have significant autocorrelation, while the STL+ARIMA model residuals do not. The Ljung-Box test at lag 52 for the STL+ARIMA model yields a p-value of 0.1477. For the SARIMA model, the p-value is 0.0. This is a strong indication that the STL+ARIMA model is better at capturing the underlying structure of the data, and that the SARIMA model is missing some important patterns. We can examine this in more practical terms by looking at the forecasts of these two models.

6.1 Forecasting

Ultimately, the primary question raised by our project involved finding the most reliable model to perform forecasting with. ARIMA and SARIMA models are heavily used in general forecasting applications, and given that TSA data is extremely seasonal, we benchmark our ARIMA(3,1,0) and SARIMA(2,1,1) models with a naive yearly and weekly forecast.

There are four forecasting Time-Series models used for the weekly forecasting test.

1. ARIMA(3,1,0): Our STL+ARIMA(3,1,0) depends on the de-seasonalized logged TSA checkpoint values, before reverting back our log-forecasts back into level values.
2. SARIMA(2,1,1): The SARIMA(2,1,1)x(0,0,0,52) model is trained on logged seasonal TSA data. Before evaluating more complex time-series models, we include simple benchmark forecasts. These are useful because they provide a baseline that any advanced model should outperform to justify its added complexity.

3. Naive last-week forecast: This benchmark predicts the current week's TSA total using the previous week's observed total, [$\hat{y}t = yt - 1$]. This is a short-memory benchmark. It assumes week-to-week conditions remain similar, but it does not explicitly account for annual seasonal patterns.

4. Seasonal naive (last-year) forecast: This benchmark predicts the current week's TSA total using the same week from the previous year, [$\hat{y}t = yt - 52$]. Since TSA checkpoint volume has strong annual seasonality, this is a rather strong benchmark. However, in the end it's still naive, and extreme events can always shift forecast results. Nevertheless, we use this to benchmark against the STL ARIMA and SARIMA models (To find the code for these forecasts and run it, please consult the appendix)

Model	MAE	RMSE	MAPE_pct
Naive (last year)	342877.17	444850.62	2.19
STL+ARIMA(3,1,0)	423640.41	490322.35	2.70
Naive (last week)	1497874.67	1635643.32	9.29
SARIMA(2,1,1)	1536459.75	1676742.33	9.44

The results show that while the naive annual forecast had the smallest Mean Absolute Error, the ARIMA(3,1,0) on de-seasonalized log data performed much better than the weekly naive forecast and the SARIMA(2,1,1). Interestingly, the SARIMA(2,1,1) performed the worst, despite being a time-series model designed to handle seasonal data. This indicates that the Seasonality and Trend Decomposition by LOESS did capture several patterns in the data that the SARIMA(2,1,1) (which has the lowest AIC/BIC values among all SARIMA combinations) failed to capture. We also note that the residuals of the ARIMA(3,1,0), while exhibiting heavy tails, were closer to the normal QQ-plot than the SARIMA residuals.

Conclusion

We analyze TSA checkpoint data from 2019 onwards, with a focus on the post-pandemic period beginning in 2022. We find a strong seasonal component to the data, alongside a general upwards trend. Through spectral analysis, we confirm the presence of strong annual and weekly seasonality, along with their harmonics. We find that the STL decomposition captures the seasonality well, but the residuals still show some serial dependence. We find that an ARIMA(3,1,0) model on the seasonally adjusted data captures the remaining structure well, and better than the SARIMA model.

In this setting, the key result is that forecasting performance depends more on separating deterministic seasonal structure from stochastic dependence before modeling. The TSA series is a good example of why that matters. It contains recurring patterns that are highly predictable, but also abrupt deviations that distort standard parametric seasonal fits. Hence, the STL ARIMA remains a reliable choice of model when it comes to decomposing and forcasting seasonal data.

Acknowledgments

This project is related to the work done in W25P02 (“Pre-Covid u.s. Airline Traffic Analysis” 2025b). However, while that project explored pre-Covid flight numbers, we explore beyond the pandemic, and focus on TSA checkpoint data, which is a more direct proxy for passenger volume than flight numbers, which may be confounded by cargo flights and changes in average passengers per flight. This is noted in comments 1 and 2 in the W25P02 comments (“Pre-Covid u.s. Airline Traffic Analysis” 2025a). Similarly, following the note made W25P02 comment 6 regarding the use of SARIMA vs ARIMA, we explore both ARIMA and SARIMA models in our work.

We also examined the work done in W20P37 (“Monthly Air Passengers(1949 - 1960) Time Series Analysis and Modelling” 2020). While our work has some similarities, we note the same distinction as with W25P02, where rather than using Kaggle data, we use data straight from the TSA. Of course, the time period we explore is also much more modern. While with some other data this may not matter much, specifically in the case of airline data, the trends observed during the popularization of air travel differ significantly from current trends. We also explore forecasting not only with our ARIMA model as in W20P37, but also using our SARIMA model.

We made use of various LLMs (Gemini, Github Copilot) to assist in particular with code debugging and generation, and to a lesser extent to verify interpretations of results. Some code blocks were generated with LLM assistance, and all code blocks were reviewed/verified by us and LLM assistance.

Bibliography

Administration, Transportation Security. 2026. “TSA Checkpoint Travel Numbers.” 2026. https://www.tsa.gov/travel/passenger-volumes.

“Autoregressive Integrated Moving Average.” 2025. 2025. https://en.wikipedia.org/wiki/Autoregressive_integrated_moving_average.

Ionides, Edward. 2026. “Notes for STATS 531, Modeling and Analysis of Time Series Data.” https://ionides.github.io/531w26/.

“Monthly Air Passengers(1949 - 1960) Time Series Analysis and Modelling.” 2016. 2016. https://en.wikipedia.org/wiki/Ljung%E2%80%93Box_test.

———. 2020. 2020. https://ionides.github.io/531w20/midterm_project/project37/Midterm_project.html.

“Pre-Covid u.s. Airline Traffic Analysis.” 2025b. 2025. https://ionides.github.io/531w25/midterm_project/project02/blinded.html.

———. 2025a. 2025. https://ionides.github.io/531w25/midterm_project/project02/comments.html.

“Seasonal-Trend Decomposition Using LOESS (STL).” 2026. 2026. https://www.statsmodels.org/dev/examples/notebooks/generated/stl_decomposition.html#Seasonal-Trend-decomposition-using-LOESS-(STL).

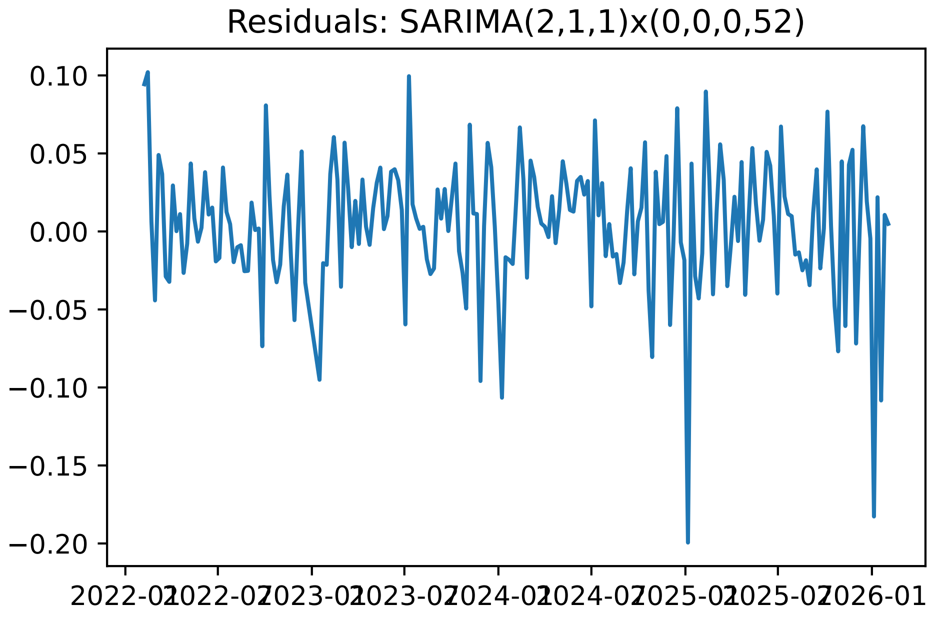
“Short-Time Fourier Transform.” 2026. 2026. https://en.wikipedia.org/wiki/Short-time_Fourier_transform.

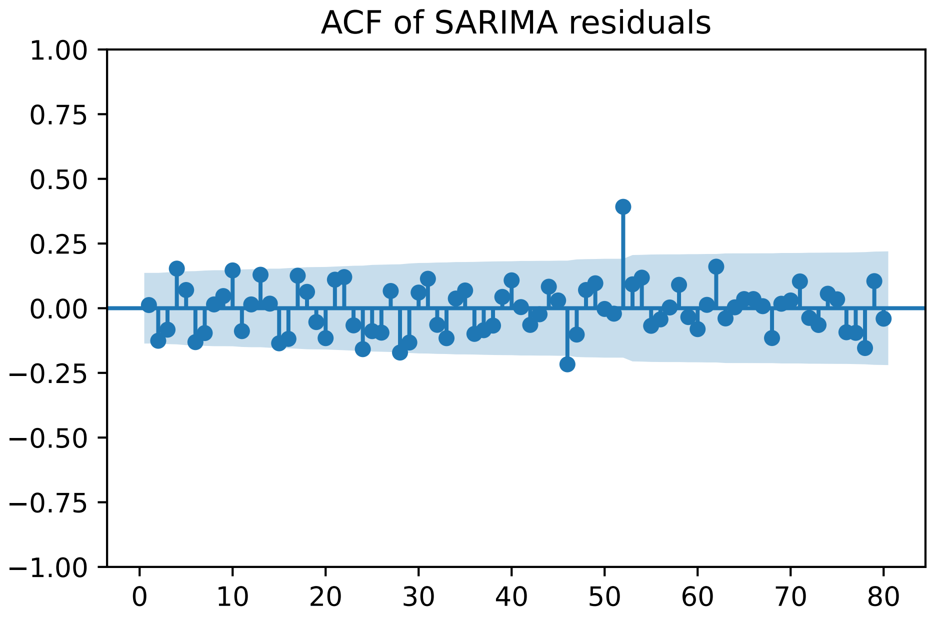
Wikipedia contributors. 2025. “Entropy (Information Theory) — Wikipedia, the Free Encyclopedia.” https://en.wikipedia.org/wiki/Entropy_(information_theory).

6.1 Supplementary material

[1]

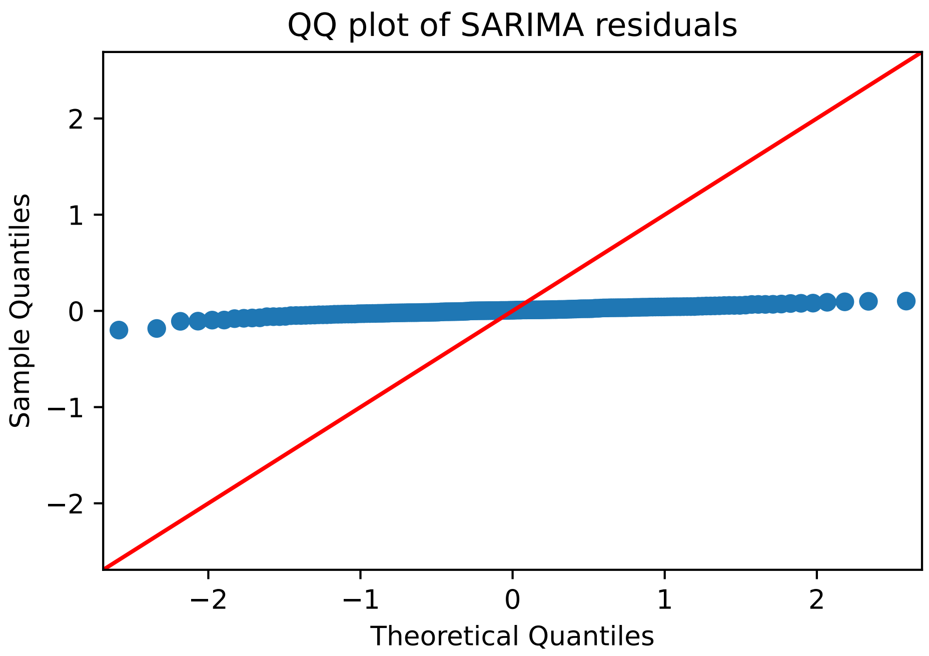
Residual Plots for SARIMA(2,1,1)





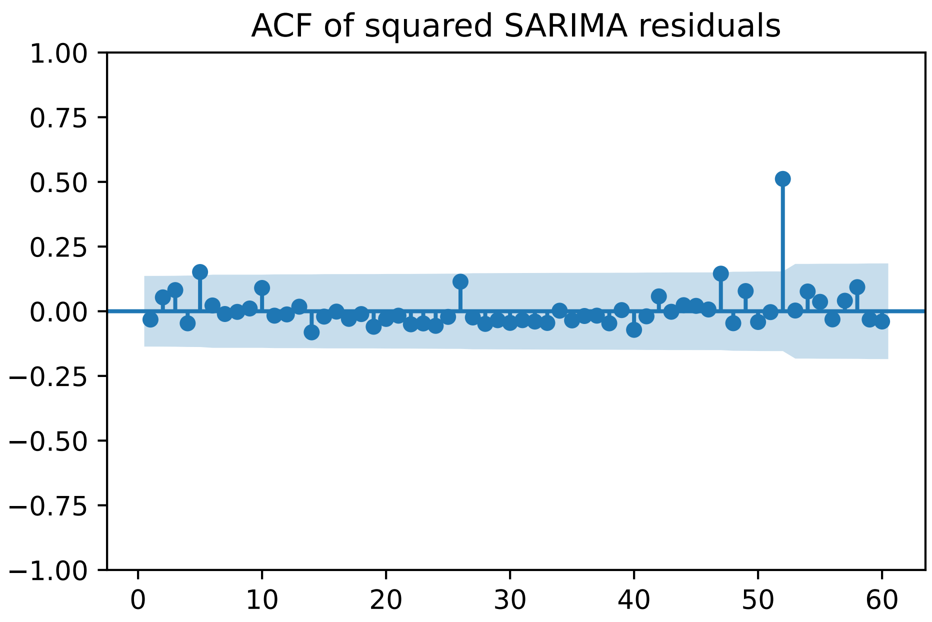
```
Ljung-Box (SARIMA residuals)
       lb_stat     lb_pvalue
```

```
10   21.695055  1.673575e-02
20   42.782372  2.184065e-03
52  156.584404  2.060084e-12
```



```
Jarque-Bera: {'JB': np.float64(163.1767522012266), 'pvalue': np.float64(3.6865359388490484e-
36)}
```

```
ARCH LM: {'LM_stat': np.float64(11.76215170019649), 'LM_pvalue': np.float64(0.30129019454110034
```



Residual Analyses: Comparison between STL+ARIMA(3,1,0) and SARIMA(2,1,1)

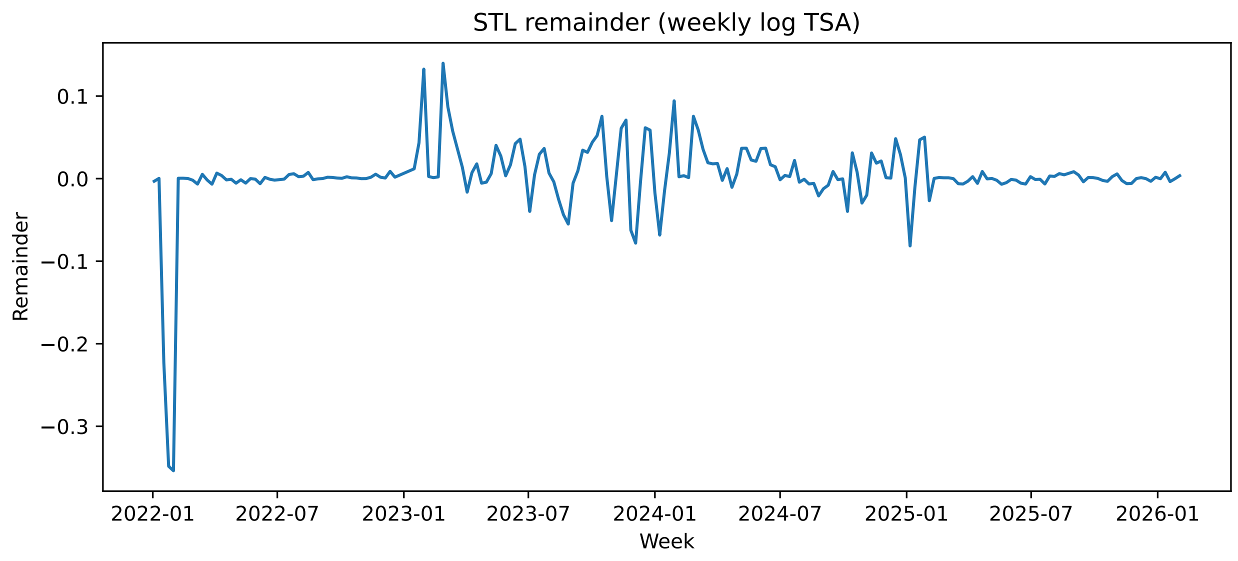
```
STL+ARIMA residual head:
week_start
2022-02-07    5.496918
2022-02-14   -0.848358
2022-02-21    3.570105
2022-02-28    6.037890
2022-03-07   -0.014626
2022-03-14    0.437775
2022-03-21   -0.111911
2022-03-28   -0.000478
2022-04-04    0.639885
2022-04-11   -0.075912
Name: STL+ARIMA(3,1,0), dtype: float64

Largest abs residuals:
week_start
2022-02-28    6.037890
2022-02-07    5.496918
2022-02-21    3.570105
2023-02-06    2.914924
2023-01-30    2.825747
2023-11-27    2.305508
2023-02-27    1.953519
```

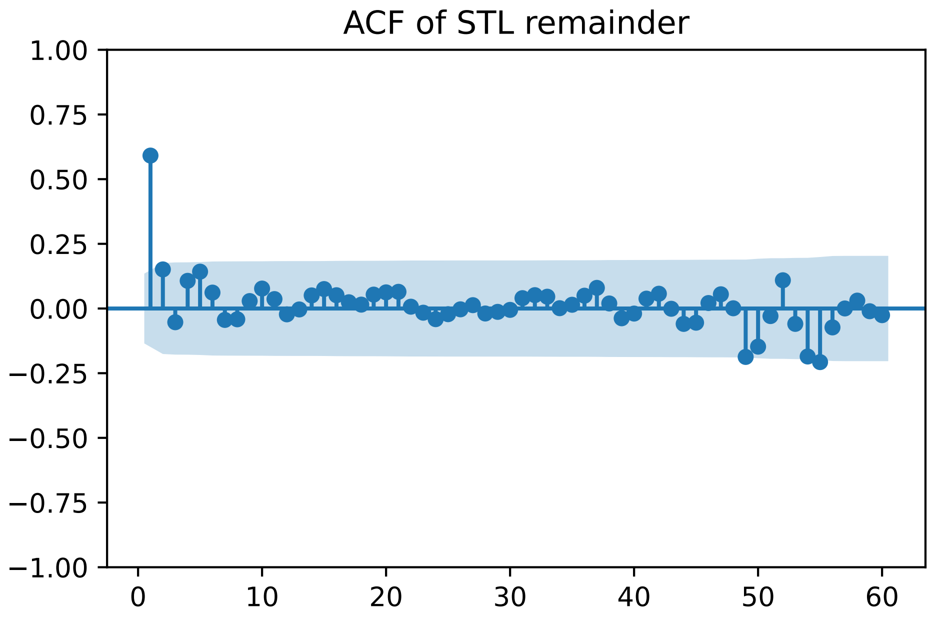
```
2025-01-06    1.886917
2024-10-14    1.810232
2023-03-27    1.781781
Name: STL+ARIMA(3,1,0), dtype: float64

Summary:
count    206.000000
mean       0.132116
std        0.913114
min       -2.914924
25%       -0.157270
50%        0.082763
75%        0.322056
max        6.037890
Name: STL+ARIMA(3,1,0), dtype: float64
```

[STL remainder diagnostics]

```python
# STL remainder diagnostics

#1 Time plot
plt.figure(figsize=(10,4))
plt.plot(remainder.index, remainder.values)
plt.title("STL remainder (weekly log TSA)")
plt.xlabel("Week")
plt.ylabel("Remainder")
plt.show()

#2 ACF
plot_acf(remainder.dropna(), lags=60, zero=False)
plt.title("ACF of STL remainder")
plt.show()

#3 Ljung-Box test
lb_rem = acorr_ljungbox(remainder.dropna(), lags=[10,20,52], return_df=True)
print("Ljung-Box on STL remainder")
print(lb_rem)

#4 differenced remainder check
dr = remainder.dropna().diff().dropna()

plot_acf(dr, lags=60, zero=False)
plt.title("ACF of differenced STL remainder")
plt.show()
```

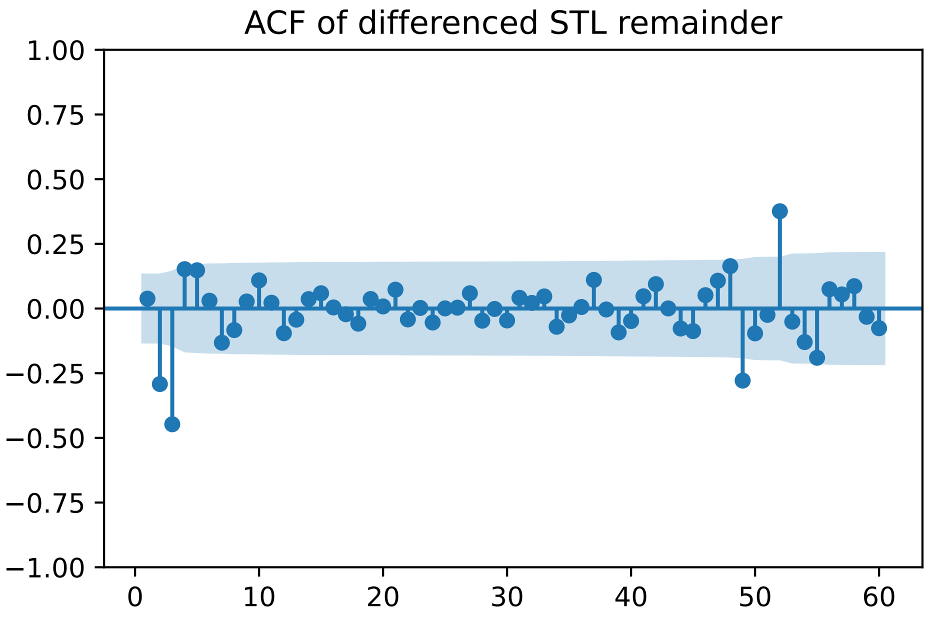
```
lb_dr = acorr_ljungbox(dr, lags=[10,20,52], return_df=True)
print("\nLjung-Box on differenced STL remainder")
print(lb_dr)
```





```
Ljung-Box on STL remainder
       lb_stat     lb_pvalue
10   90.369282  4.524751e-15
```

```
20   95.054172  9.564408e-12
52  124.677307  6.613433e-08
```



```
Ljung-Box on differenced STL remainder
       lb_stat     lb_pvalue
10   79.707927  5.727693e-13
20   84.616397  6.374443e-10
52  179.298690  6.734613e-16
```