# Future-Proofing TSA Operations: Data-Driven Resource Optimization

```r
library(readxl)
#install.packages('fpp3')
#install.packages("urca")
library(fpp3)
```

```
Registered S3 method overwritten by 'tsibble':
  method               from
  as_tibble.grouped_df dplyr
```

```
-- Attaching packages -------------------------------------------- fpp3 1.0.2 --
```

```
v tibble      3.2.1     v tsibble     1.1.6
v dplyr       1.1.4     v tsibbledata 0.4.1
v tidyr       1.3.1     v feasts      0.4.2
v lubridate   1.9.4     v fable       0.4.1
v ggplot2     3.5.1
```

```
-- Conflicts ------------------------------------------------- fpp3_conflicts --
x lubridate::date()    masks base::date()
x dplyr::filter()      masks stats::filter()
x tsibble::intersect() masks base::intersect()
x tsibble::interval()  masks lubridate::interval()
x dplyr::lag()         masks stats::lag()
x tsibble::setdiff()   masks base::setdiff()
x tsibble::union()     masks base::union()
```

```r
library(tidyverse)
```

```
-- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
v forcats 1.0.0     v readr   2.1.5
v purrr   1.0.2     v stringr 1.5.1
```

```
-- Conflicts ------------------------------------------ tidyverse_conflicts() --
x dplyr::filter()     masks stats::filter()
x tsibble::interval() masks lubridate::interval()
x dplyr::lag()        masks stats::lag()
i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to becom
```

```r
library(fpp3)
library(slider)
library(urca)
library(scales)
```

```
Attaching package: 'scales'

The following object is masked from 'package:purrr':

    discard

The following object is masked from 'package:readr':

    col_factor
```

### Exploratory Data Analysis

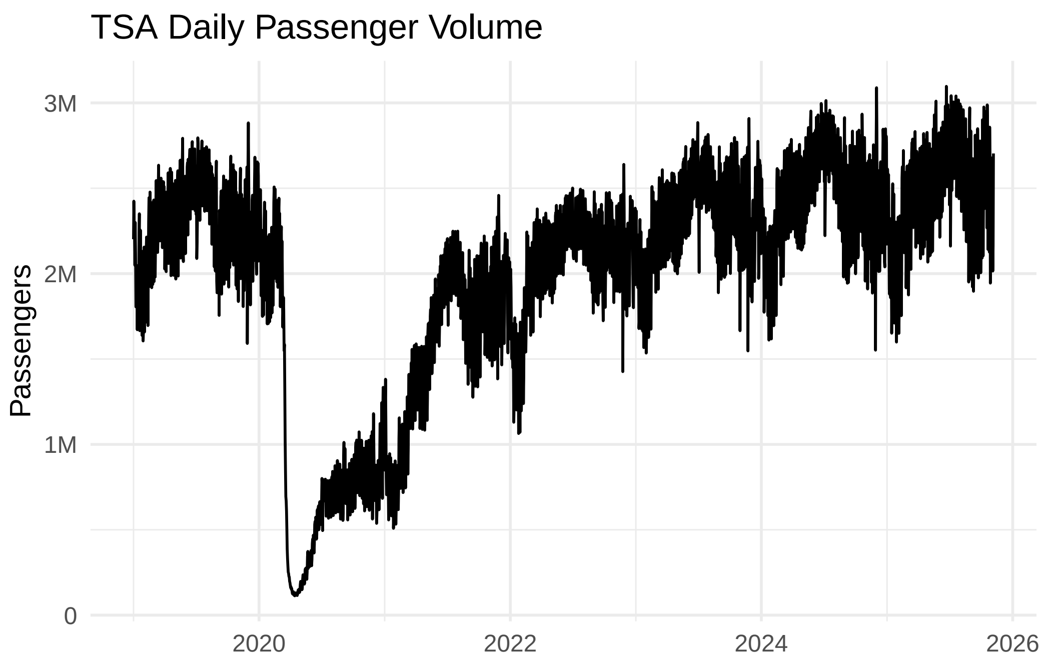
```r
tsa_clean <- read_excel(here::here("Data", "TSA.xlsx"))
```

```r
head(tsa_clean)
```

```
# A tibble: 6 x 2
  Date                Numbers
  <dttm>                <dbl>
1 2025-11-06 00:00:00 2703787
2 2025-11-05 00:00:00 2162183
3 2025-11-04 00:00:00 2015297
4 2025-11-03 00:00:00 2492532
```

```
5 2025-11-02 00:00:00 2279000
6 2025-11-01 00:00:00 2029517
```

```r
library(ggplot2)
autoplot(tsa_clean |> mutate(Date = as.Date(Date)) |> as_tsibble(index = Date) , Numbers) +
  scale_y_continuous(labels = label_number(scale_cut = cut_short_scale())) +
  labs(
    title = "TSA Daily Passenger Volume",
    y = "Passengers",
    x = NULL
  ) +
  theme_minimal()
```




```r
# Glimpse at dataset
glimpse(tsa_clean)
```

```
Rows: 2,502
Columns: 2
$ Date    <dttm> 2025-11-06, 2025-11-05, 2025-11-04, 2025-11-03, 2025-11-02, 2~
$ Numbers <dbl> 2703787, 2162183, 2015297, 2492532, 2279000, 2029517, 2678000,~
```

```r
# Number Column summary
summary(tsa_clean$Numbers)
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
 113147 1811221 2188235 2024965 2498146 3096797 
```

```r
# Missing data check
sum(is.na(tsa_clean$Numbers))
```

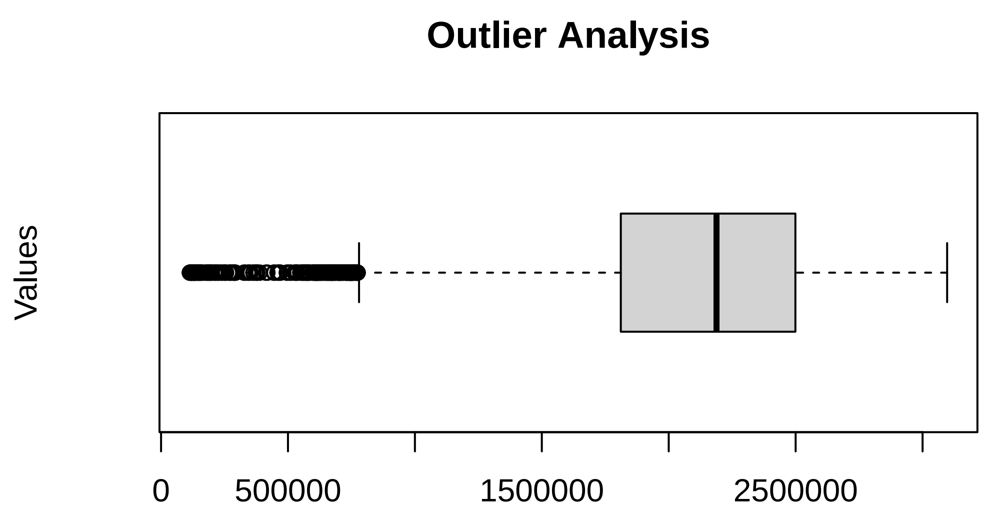
```
[1] 0
```

```r
# Range of the Dates
min(tsa_clean$Date); max(tsa_clean$Date)
```

```
[1] "2019-01-01 UTC"
```

```
[1] "2025-11-06 UTC"
```

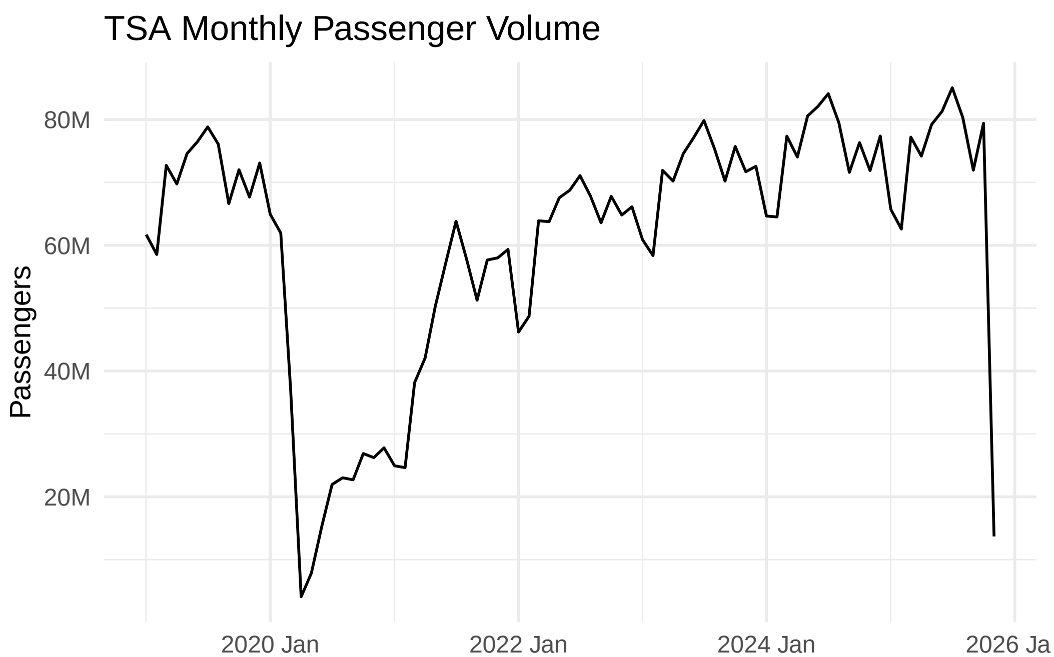
```r
# Outlier analysis
boxplot(tsa_clean$Numbers, main = "Outlier Analysis", ylab = "Values", horizontal = TRUE)
```



```r
# List of outliers -- 2020-2021
Q1 <- quantile(tsa_clean$Numbers, 0.25, na.rm = TRUE)
Q3 <- quantile(tsa_clean$Numbers, 0.75, na.rm = TRUE)
IQR_value <- IQR(tsa_clean$Numbers, na.rm = TRUE)

lower_bound <- Q1 - (1.5 * IQR_value)
upper_bound <- Q3 + (1.5 * IQR_value)

outliers <- tsa_clean |> filter(Numbers < lower_bound | Numbers > upper_bound)

head(outliers)
```

```
# A tibble: 6 x 2
  Date                Numbers
  <dttm>                <dbl>
1 2021-01-06 00:00:00  705249
2 2021-01-09 00:00:00  750419
3 2021-01-11 00:00:00  750407
4 2021-01-12 00:00:00  557517
5 2021-01-13 00:00:00  605887
6 2021-01-16 00:00:00  729703
```

```r
tsa_monthly <- tsa_clean |>
  mutate(Month = yearmonth(Date)) |>
  group_by(Month) |>
  summarise(Total = sum(Numbers, na.rm = TRUE)) |>
  ungroup() |>
  as_tsibble(index = Month)

autoplot(tsa_monthly, Total) +
  scale_y_continuous(labels = label_number(scale_cut = cut_short_scale())) +
  labs(
    title = "TSA Monthly Passenger Volume",
    y = "Passengers",
    x = NULL
  ) +
  theme_minimal()
```

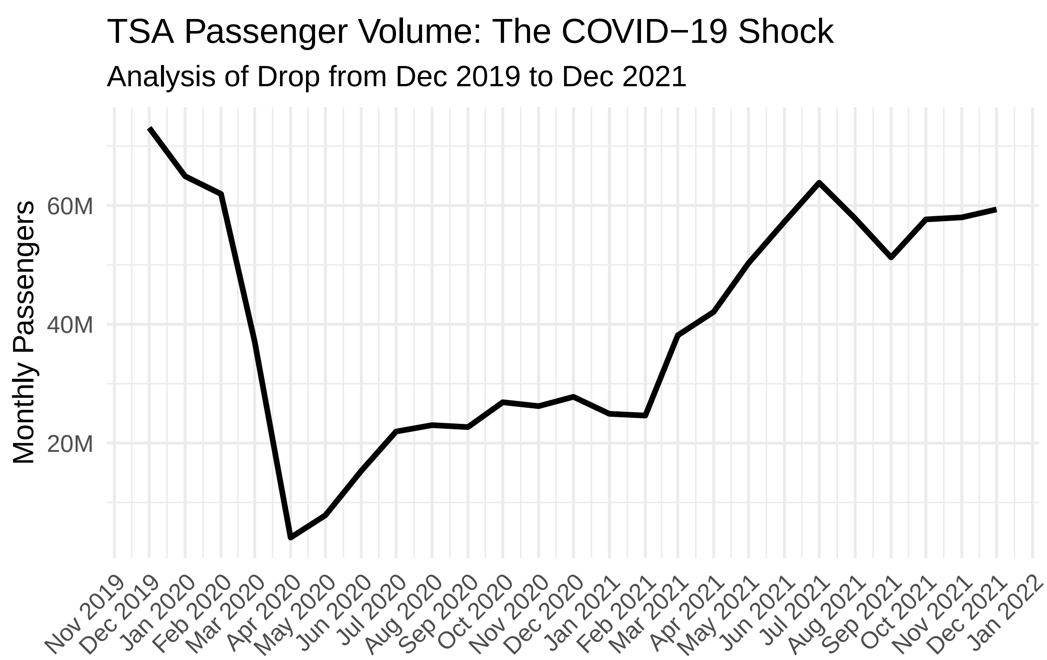



## Training/Test

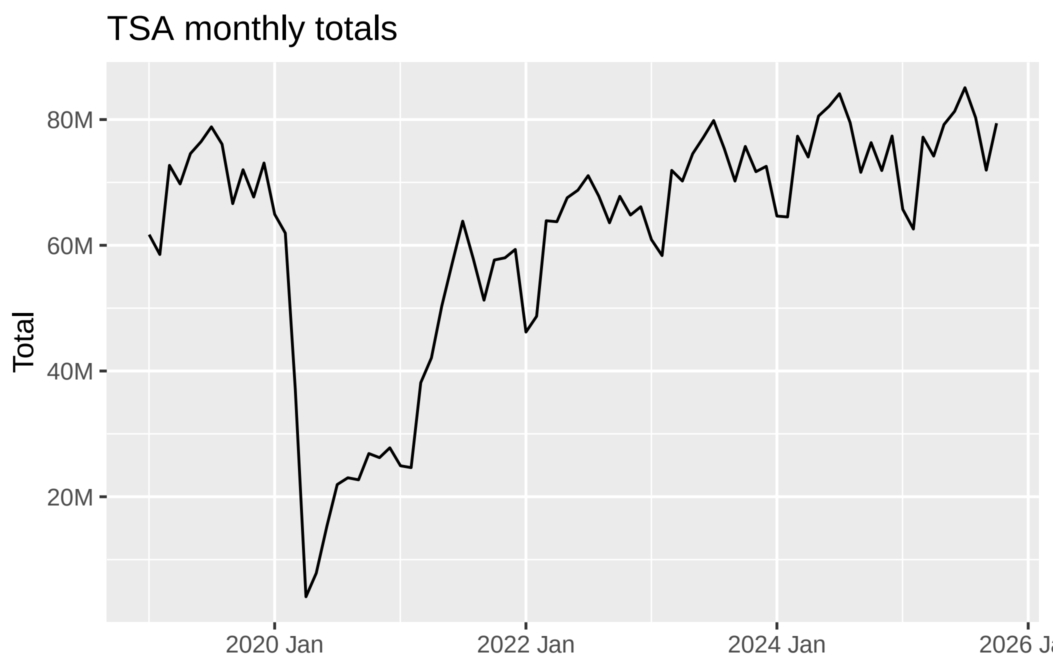
Convert to TSIBBLE object, filter the months, and create a Totals Column.

```
covid_window <- tsa_monthly %>% # Changed from tsa_clean to tsa_monthly
  filter_index("2019 Dec" ~ "2021 Dec")

# Plot the Drop
covid_window %>%
  autoplot(Total, linewidth = 1) +

  # Formatting axes
  scale_y_continuous(labels = label_number(scale_cut = cut_short_scale())) +
  scale_x_yearmonth(date_breaks = "1 month", date_labels = "%b %Y") +

  labs(
    title = "TSA Passenger Volume: The COVID-19 Shock",
    subtitle = "Analysis of Drop from Dec 2019 to Dec 2021",
    y = "Monthly Passengers",
    x = NULL
  ) +
```

```
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```




```
# Expect columns: Date, Numbers (daily)
tsa_clean <- tsa_monthly %>%
  filter(Month < max(Month)) %>%
  filter(year(Month) >= 2019) %>%
  mutate(
    covid_shock = if_else(Month >= yearmonth("2020 Feb") & Month <= yearmonth("2021 Dec"), 1
  )

autoplot(tsa_clean, Total) + labs(title = "TSA monthly totals", x = NULL, y = "Total")+
scale_y_continuous(labels = label_number(scale_cut = cut_short_scale()))
```



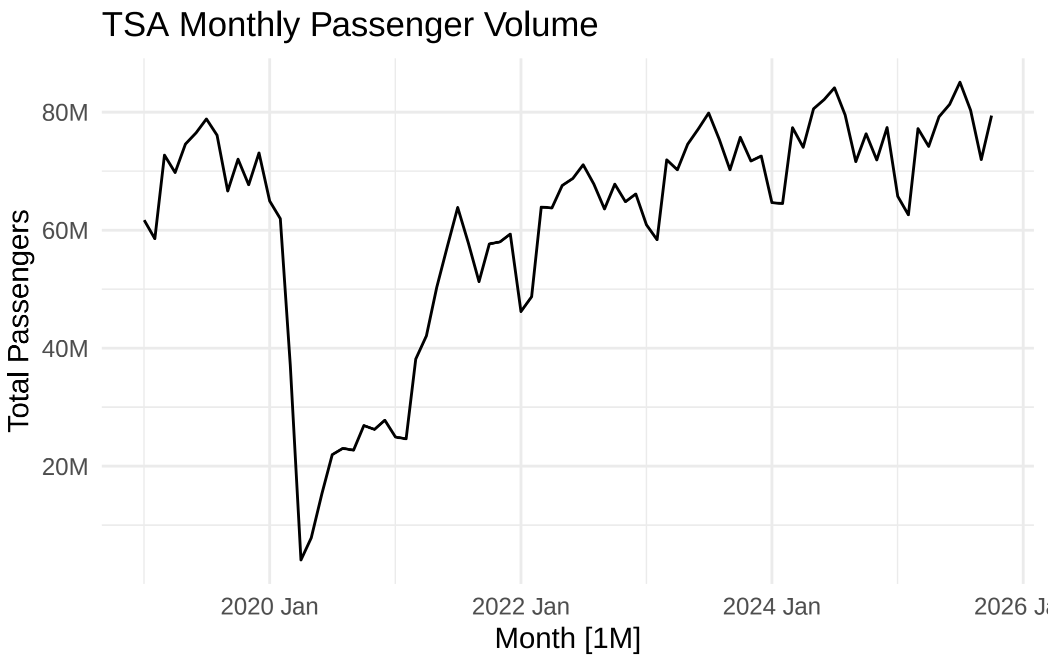

```
head(tsa_clean)
```

```
# A tsibble: 6 x 3 [1M]
     Month    Total covid_shock
     <mth>    <dbl>       <dbl>
1 2019 Jan 61694899           0
2 2019 Feb 58535547           0
3 2019 Mar 72714771           0
4 2019 Apr 69754997           0
5 2019 May 74582398           0
6 2019 Jun 76489355           0
```

#### Monthly Timeseries Plot

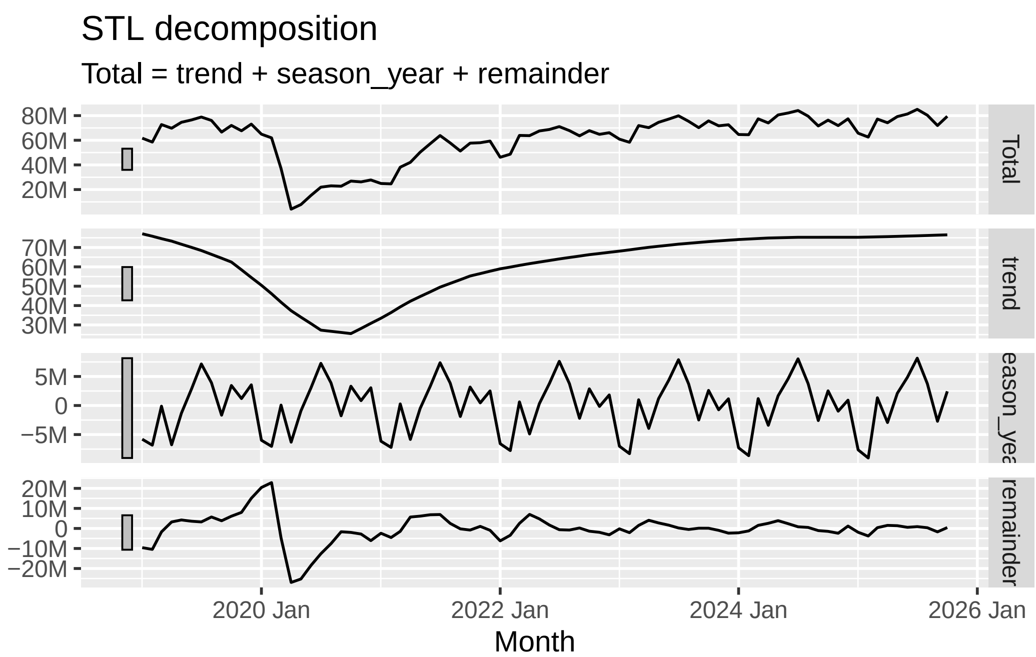
```
tsa_clean %>%
  autoplot(Total) +
  scale_y_continuous(labels = label_number(scale_cut = cut_short_scale()))+
    labs(
    title = "TSA Monthly Passenger Volume",
    y = "Total Passengers"
```

```
  ) +
  theme_minimal()
```




#### STL Decomposition

```
tsa_clean %>%
  model(STL(Total)) %>%
  components() %>%
  autoplot()+
  scale_y_continuous(labels = label_number(scale_cut = cut_short_scale()))
```




**Training/Testing Phase**

```
h <- 12 # goo for evalaution seasonality

# train test, fit
n_total <- nrow(tsa_clean)
train <- tsa_clean %>% filter(year(Month) <= 2023)
test  <- tsa_clean %>% filter(year(Month) > 2023)
```

```
max(test$Month) ; min(test$Month)
```

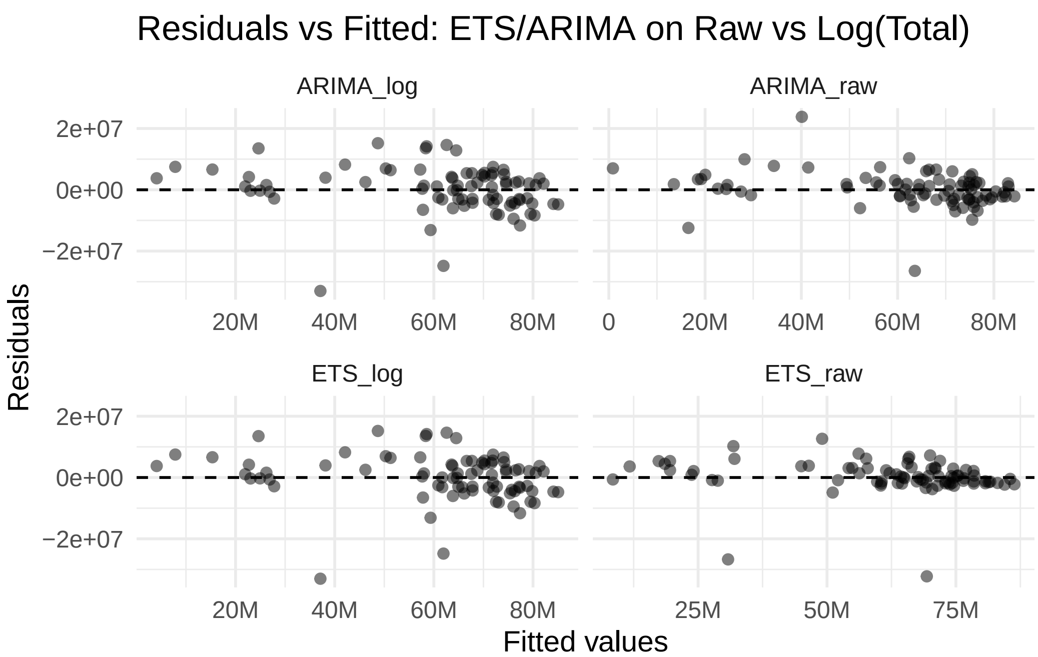
```
<yearmonth[1]>
[1] "2025 Oct"
```

```
<yearmonth[1]>
[1] "2024 Jan"
```

```
max(train$Month) ; min(train$Month)
```

```
<yearmonth[1]>
[1] "2023 Dec"

<yearmonth[1]>
[1] "2019 Jan"
```

```r
# Simple diagnostic models (not your final comparison yet)
diag_fit <- tsa_clean %>%
  model(
    ETS_raw   = ETS(Total),
    ETS_log   = ETS(log(Total)),
    ARIMA_raw = ARIMA(Total),
    ARIMA_log = ARIMA(log(Total))
  )

# Residuals vs fitted for both
diag_resids <- diag_fit %>%
  augment()

diag_resids %>%
  ggplot(aes(x = .fitted, y = .resid)) +
  geom_point(alpha = 0.5) +
  geom_hline(yintercept = 0, linetype = "dashed") +
  facet_wrap(~ .model, scales = "free_x") +
  labs(
    title = "Residuals vs Fitted: ETS/ARIMA on Raw vs Log(Total)",
    x = "Fitted values",
    y = "Residuals"
  ) +
  scale_x_continuous(
    labels = scales::label_number(scale_cut = scales::cut_short_scale())
  ) +
  theme_minimal()
```

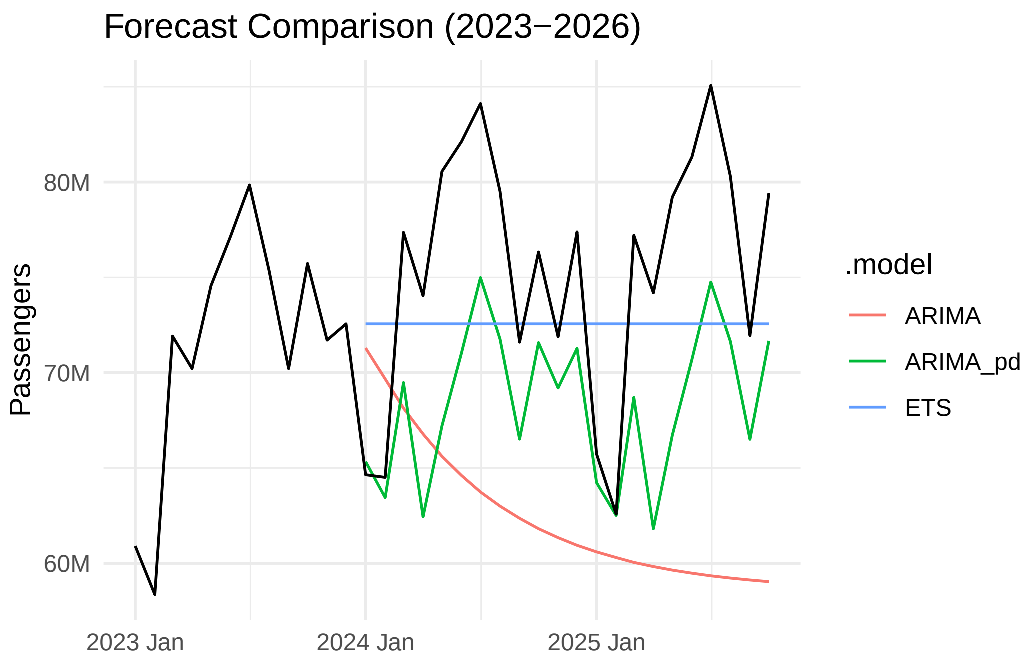



Residuals vs fitted plots for ETS and ARIMA on both the original and log-transformed scales show that the raw-scale models (ETS(Total) and ARIMA(Total)) have tighter, more homoscedastic residuals centered around zero. In contrast, the log-transformed models exhibit larger and more structured residuals. Together with the poorer RMSE/MAE for the log models, this indicates that a log transformation does not improve the fit and is not appropriate for this series. We therefore proceed using models on the original scale only.

### Fitting Stage

```
fit <- train %>%
  model(
    ETS    = ETS(Total),
    ARIMA = ARIMA(Total),
    ARIMA_pd = ARIMA((Total) ~ covid_shock + season() + trend())
  )

fc <- new_data(train, n = nrow(test)) %>%
  mutate(covid_shock = 0)

fc <- fit %>%
  forecast(new_data = fc)
```

```r
fc %>%
  filter(year(Month) >= 2023) %>%
  autoplot(tsa_clean %>% filter(year(Month) >= 2023), level = NULL) +
    scale_y_continuous(labels = label_number(scale_cut = cut_short_scale())) +

  labs(
    title = "Forecast Comparison (2023-2026)",
    y = "Passengers",
    x = NULL
  ) +
  theme_minimal()
```




```r
acc_baseline_models <- accuracy(fc, test) %>%
  select(.model:MAPE)

acc_baseline_models %>%
  arrange(RMSE) %>%
  knitr::kable()
```

```
Warning in attr(x, "align"): 'xfun::attr()' is deprecated.
Use 'xfun::attr2()' instead.
See help("Deprecated")
```

```
Warning in attr(x, "format"): 'xfun::attr()' is deprecated.
Use 'xfun::attr2()' instead.
See help("Deprecated")
```

| .model | .type | ME | RMSE | MAE | MPE | MAPE |
|---|---|---|---|---|---|---|
| ETS | Test | 2939451 | 7018582 | 6124212 | 3.160949 | 8.088764 |
| ARIMA_pd | Test | 7160362 | 8266266 | 7223109 | 9.146159 | 9.243218 |
| ARIMA | Test | 12958056 | 15346419 | 14031963 | 16.394367 | 18.057124 |

## ARIMA Set-Up

```r
print('Stationary Test')
```

```
[1] "Stationary Test"
```

```r
train |> fabletools::features(Total, c(unitroot_kpss, unitroot_ndiffs, unitroot_nsdiffs)) |>
```

```
Warning in attr(x, "align"): 'xfun::attr()' is deprecated.
Use 'xfun::attr2()' instead.
See help("Deprecated")
```

```
Warning in attr(x, "format"): 'xfun::attr()' is deprecated.
Use 'xfun::attr2()' instead.
See help("Deprecated")
```

| kpss_stat | kpss_pvalue | ndiffs | nsdiffs |
|---|---|---|---|
| 0.3902161 | 0.0813724 | 0 | 0 |

Observation: Null hypothesis cannot be rejected 0.05. Therefore, series is stationary. 1 regular and 1 seasonal differentiation is required.

```r
print('Stationary test after applying a season and regular differentiation')
```

```
[1] "Stationary test after applying a season and regular differentiation"
```

```
train |> fabletools::features(Total |>  difference(lag=1) , c(unitroot_kpss, unitroot_ndiffs
```

```
Warning in attr(x, "align"): 'xfun::attr()' is deprecated.
Use 'xfun::attr2()' instead.
See help("Deprecated")

Warning in attr(x, "format"): 'xfun::attr()' is deprecated.
Use 'xfun::attr2()' instead.
See help("Deprecated")
```

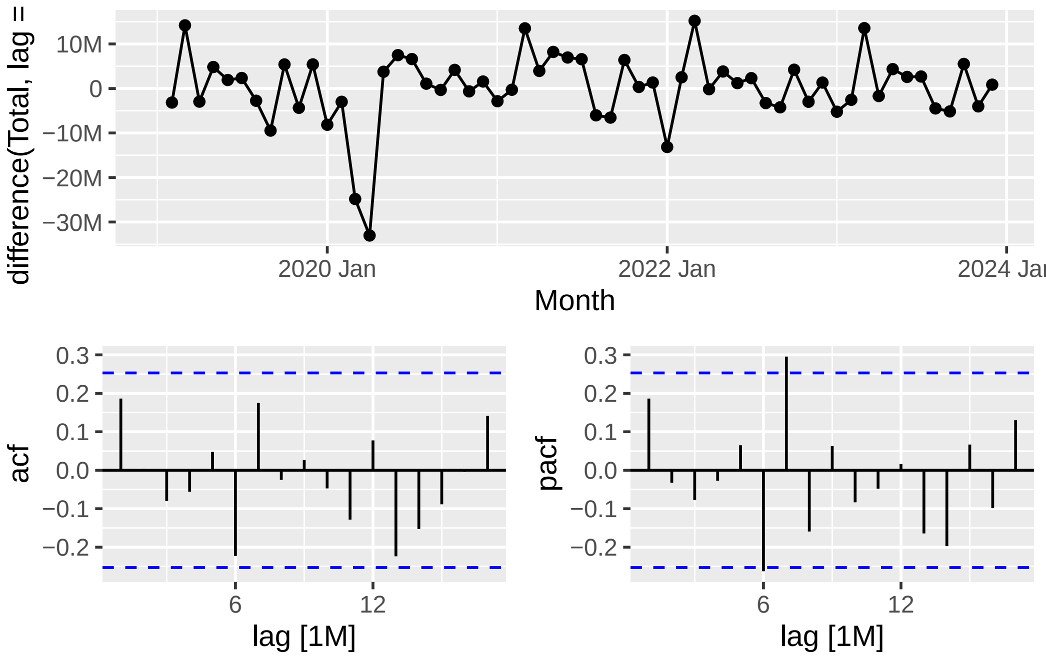
| kpss_stat | kpss_pvalue | ndiffs | nsdiffs |
|---|---|---|---|
| 0.1058268 | 0.1 | 0 | 0 |

```
#orders1 <- tsa_clean |>
#  mutate(Total_d = difference(difference(Total, lag = 12))) |>
#  features(Total_d, unitroot_kpss)
#print(orders1)
```

## Differenced series and ACF/PACF plots

```
# plot differenced series and ACF/PACF plots with non-seasonal and seasonal differencing
train |>
  feasts::gg_tsdisplay(
    Total |> difference(lag = 1),
    plot_type = "partial"
  ) +
  ggplot2::scale_y_continuous(labels = label_number(scale_cut = cut_short_scale()))
```

```
Warning: `gg_tsdisplay()` was deprecated in feasts 0.4.2.
i Please use `ggtime::gg_tsdisplay()` instead.

Warning: Removed 1 row containing missing values or values outside the scale range
(`geom_line()`).

Warning: Removed 1 row containing missing values or values outside the scale range
(`geom_point()`).
```




```r
# model fit
fitted_manual_arima <- train |>
  model(
    ma1_manual = ARIMA(Total ~ pdq(0, 1, 1) + PDQ(0, 1, 1)),
    ar1_manual = ARIMA(Total ~ pdq(1, 1, 0) + PDQ(1, 1, 0))
  )

h_test <- nrow(test)

# Holds both forecast models
fc_manual_arima <- fitted_manual_arima |>
  forecast(h = h_test)
```

```r
# Manual ARIMA scores from Training Data
fitted_manual_arima |> glance() |> select(.model, AICc) |> arrange(AICc)
```

```
# A tibble: 2 x 2
  .model      AICc
  <chr>      <dbl>
1 ma1_manual 1626.
2 ar1_manual 1632.
```

```r
all_models <- train |>
  model(
    ETS        = ETS(Total),
    auto_arima = ARIMA(Total),
    ma1_manual = ARIMA(Total ~ pdq(0, 1, 1) + PDQ(0, 1, 1)),
    ar1_manual = ARIMA(Total ~ pdq(1, 1, 0) + PDQ(1, 1, 0))
  )

all_models |> glance() |> select(.model, AICc) |> arrange(AICc)
```

```
# A tibble: 4 x 2
  .model      AICc
  <chr>      <dbl>
1 ma1_manual 1626.
2 ar1_manual 1632.
3 auto_arima 2080.
4 ETS        2158.
```

All performance metrics combined

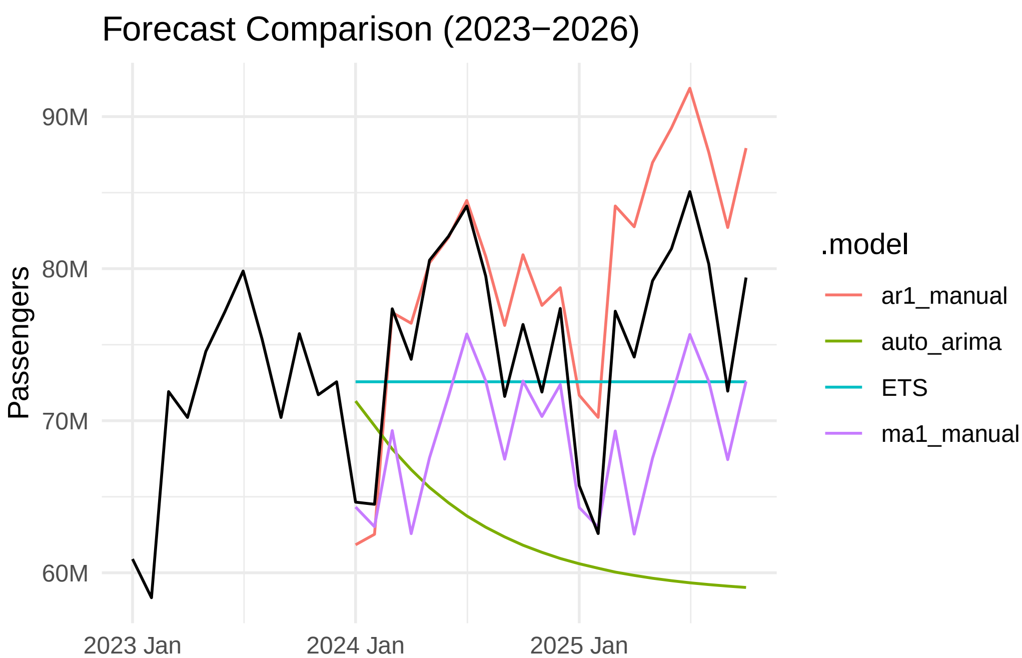
```r
forecast_all_models <- all_models |>
  forecast(h = nrow(test))

accuracy_combined <- forecast_all_models |>
  accuracy(tsa_clean) |>
  select(.model, RMSE, MAE, MAPE) |>
  arrange(RMSE)
accuracy_combined |> knitr::kable()
```

```
Warning in attr(x, "align"): 'xfun::attr()' is deprecated.
Use 'xfun::attr2()' instead.
See help("Deprecated")
```

```
Warning in attr(x, "format"): 'xfun::attr()' is deprecated.
Use 'xfun::attr2()' instead.
See help("Deprecated")
```

| .model | RMSE | MAE | MAPE |
|---|---|---|---|
| ar1_manual | 5734254 | 4719541 | 6.348599 |
| ETS | 7018582 | 6124212 | 8.088764 |
| ma1_manual | 7687646 | 6627659 | 8.481189 |
| auto_arima | 15346419 | 14031963 | 18.057124 |

```
# Plot forecasts from all models, restricted to 2023 onward
forecast_all_models %>%
  filter(year(Month) >= 2023) %>%
  autoplot(tsa_clean %>% filter(year(Month) >= 2023), level = NULL) +
  scale_y_continuous(labels = label_number(scale_cut = cut_short_scale())) +
  labs(
    title = "Forecast Comparison (2023-2026)",
    y = "Passengers",
    x = NULL
  ) +
  theme_minimal()
```




Based on test-set RMSE, the ARIMA(1,1,0)(1,1,0) model (ar1_manual) achieved the best out-of-sample performance (RMSE 5.7M), outperforming both ETS and the alternative ARIMA specifications. Although ma1_manual had slightly better AICc on the training data, ar1_manual provided superior accuracy on the holdout period, so we select ar1_manual as our final model. Log-transformed and pandemic-adjusted models all produced substantially larger RMSE values and were discarded.

## Accuracy Comparison with Cross-validation

```
n_total <- nrow(tsa_clean)
train <- tsa_clean |> slice_head(n = n_total - h)
test  <- tsa_clean |> slice_tail(n = h)

cv_train <- tsa_clean |> slice_head(n = n_total - h) |> stretch_tsibble(.init=48, .step = 12)
```

```
# new cv fits
cv_fits <- cv_train |>
  model(
    ETS        = ETS(Total),
    ARIMA      = ARIMA(Total),
    ma1_manual = ARIMA(Total ~ pdq(0,1,1) + PDQ(0,1,0)),
    ar1_manual = ARIMA(Total ~ pdq(1,1,0) + PDQ(0,0,1))
  )
```

```
fc_cv_models <- cv_fits |> forecast(h = "12 months")


accuracy <- accuracy(fc_cv_models, tsa_clean) |> select(.model, .type, ME, RMSE, MAE, MAPE)
accuracy |> knitr::kable()
```

```
Warning in attr(x, "align"): 'xfun::attr()' is deprecated.
Use 'xfun::attr2()' instead.
See help("Deprecated")
```

```
Warning in attr(x, "format"): 'xfun::attr()' is deprecated.
Use 'xfun::attr2()' instead.
See help("Deprecated")
```

| .model | .type | ME | RMSE | MAE | MAPE |
|---|---|---:|---:|---:|---:|
| ma1_manual | Test | -1145959 | 2985843 | 2177955 | 3.054853 |
| ar1_manual | Test | 3354766 | 6594746 | 5904169 | 8.039470 |

| .model | .type | ME | RMSE | MAE | MAPE |
|---|---|---|---|---|---|
| ETS | Test | 4095700 | 7399181 | 6646001 | 9.014394 |
| ARIMA | Test | 11219474 | 13945564 | 13005485 | 17.332255 |

Although ETS produced the best accuracy on the single 12-month hold-out set, its performance dropped considerably under rolling-origin cross-validation. This difference occurs because the simple train/test split evaluates the model on only one recent period (a relatively stable year), where ETS performs well. In contrast, time-series cross-validation tests the model across multiple historical forecasting scenarios, including volatile periods such as the COVID-19 shock and the recovery phase. In these more challenging conditions, ETS is less stable, while the manual ARIMA(0,1,1)(0,1,0) model remains consistently accurate. Therefore, cross-validation indicates that the ARIMA model is more robust overall and is the preferred final model.

**Workforce Prediction**

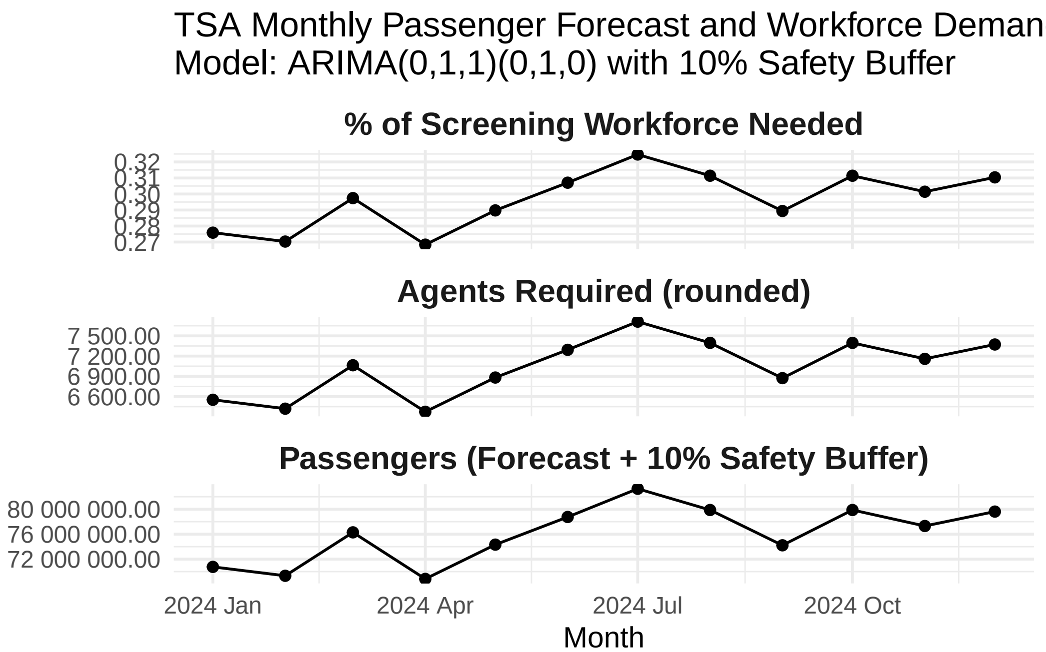
```r
# Workforce Assumptions (Based on TSA lane throughput)
current_agents <- 47500

# Lane capacity per hour (assumed)
pax_per_lane_hour <- 180

# Daily Operating hours per lane
hours_per_day_lane <- 16

# Days per month
days_per_month <- 30

# TSOs required to operate one lane
tsos_per_lane <- 8

# Monthly lane capacity
pax_per_lane_month <- pax_per_lane_hour * hours_per_day_lane * days_per_month

# Share of total workforce assigned to screen (assumption)
screening_duty_fraction <- 0.5

# Active checkpoint workforce
active_screening_tsos <- current_agents * screening_duty_fraction

# Passengers each TSO effectively supports/month
passengers_per_agent_month <- pax_per_lane_month / tsos_per_lane
```

```r
# ma1_manual is your chosen final model
fc_final <- all_models |>
  select(ma1_manual) |>
  forecast(h = nrow(test))
```

```r
buffer_factor <- 1.10  # 10% safety buffer

tsa_agents_forecast <- fc_final |>
  mutate(
    pax_forecast_mean = as.numeric(.mean),
    pax_forecast_high = pax_forecast_mean * buffer_factor,
    agents_required_exact   = pax_forecast_high / passengers_per_agent_month,
    agents_required_ceiling = ceiling(agents_required_exact),
    pct_of_screening_staff  = agents_required_ceiling / active_screening_tsos
  )
```

```r
tsa_dual <- tsa_agents_forecast |>
  as_tibble() |>
  transmute(
    Month,
    `Passengers (Forecast + 10% Safety Buffer)` = pax_forecast_high,
    `Agents Required (rounded)`                 = agents_required_ceiling,
    `% of Screening Workforce Needed`           = pct_of_screening_staff
  ) |>
  pivot_longer(
    cols      = -Month,
    names_to  = "metric",
    values_to = "value"
  ) |>
  drop_na(value)

ggplot(tsa_dual, aes(x = Month, y = value)) +
  geom_line() +
  geom_point() +
  facet_wrap(~ metric, scales = "free_y", ncol = 1) +
  labs(
    title = "TSA Monthly Passenger Forecast and Workforce Demand\nModel: ARIMA(0,1,1)(0,1,0)
    x = "Month",
    y = NULL
  ) +
  scale_y_continuous(
    labels = scales::label_number(accuracy = 0.01)   # <-- removed scale_cut
```

```
) +
theme_minimal() +
theme(
  strip.text = element_text(size = 12, face = "bold")
)
```




**Insight:** Nearly 200 TSA Officers are paid by the government but work full-time on union matters. These people do not retain certification to perform screening functions. Additionally, in a recent TSA employee survey, over 60% said poor performers are allowed to stay employed and, not surprisingly, continue to not perform.

**Source:** https://www.dhs.gov/news/2025/03/07/dhs-ends-collective-bargaining-tsas-transportation-security-officers-enhancing

Recommendation We recommend that TSA adopt a Monthly Forecasting Cadence anchored by the ARIMA(0,1,1)(0,1,0) model to guide checkpoint workforce planning. This forecasting approach provides TSA with a reliable month-ahead view of expected passenger volumes, enabling proactive and data-driven staffing decisions.

Using TSA's own operational throughput standards (180 passengers per lane per hour, ~8 TSOs required per lane across shifts) and assuming that approximately 50% of the agency's 47,500 officers rotate through screening duties, our workforce model shows that TSA will need only 7,000–8,500 officers per month on screening duty throughout 2024. Even after applying a 10% safety buffer to account for forecast uncertainty, this represents only 25–36% of the available screening workforce.

This reveals a significant and consistent capacity margin that TSA can strategically leverage. Monthly forecasting enables TSA to:

Adjust screening deployments by month, rather than relying on static annual templates

Reallocate 15–25% of screeners during low-volume periods to training, leave, administrative roles, or airport support

Reduce unnecessary overstaffing during shoulder travel months

Prepare earlier for predictable surges, decreasing reliance on overtime and last-minute staffing during holiday and summer peaks

Enhance TSA well-being by smoothing workload intensity and improving shift predictability

By aligning staffing to model-based demand—and doing so monthly—TSA can maintain its <30-minute wait-time service level while operating more efficiently, reducing operational strain, and improving workforce readiness without reducing headcount.

```r
library(dplyr)
library(lubridate)
library(tsibble)
library(scales)
library(knitr)

# Prep data
tsa_yearly <- tsa_clean %>%
  as_tibble() %>%
  ungroup() %>%
  mutate(Year = year(Month)) %>%
  group_by(Year) %>%
  summarise(
    yearly_total = sum(Total, na.rm = TRUE),
    mean_monthly = mean(Total, na.rm = TRUE),
    median_monthly = median(Total, na.rm = TRUE),
    .groups = "drop"
  ) %>%
  arrange(Year) %>%
  mutate(
    yoy_growth = (yearly_total / lag(yearly_total) - 1),
    Year = as.character(Year)
  )
```

```r
tsa_overall <- tsa_clean %>%
  as_tibble() %>%
  ungroup() %>%
  summarise(
    Year           = "All years",
    yearly_total = sum(Total, na.rm = TRUE),
    mean_monthly = mean(Total, na.rm = TRUE),
    median_monthly = median(Total, na.rm = TRUE),
    yoy_growth     = NA_real_
  )

tsa_raw_data <- bind_rows(tsa_overall, tsa_yearly)

# Format and display table
tsa_raw_data %>%
  mutate(
    `Total Passengers` = comma(yearly_total),  # Adds commas: 1,000,000
    `Avg Monthly`      = comma(mean_monthly, accuracy = 1),
    `Median Monthly`   = comma(median_monthly, accuracy = 1),
    `YoY Growth`       = if_else(is.na(yoy_growth), "-", percent(yoy_growth, accuracy = 0.1))
  ) %>%
  select(Year, `Total Passengers`, `YoY Growth`, `Avg Monthly`, `Median Monthly`) %>%
  kable(
    align = "r",
    caption = "TSA Passenger Throughput (2019-2025)"
  )
```

```
Warning in attr(x, "align"): 'xfun::attr()' is deprecated.
Use 'xfun::attr2()' instead.
See help("Deprecated")
```

```
Warning in attr(x, "format"): 'xfun::attr()' is deprecated.
Use 'xfun::attr2()' instead.
See help("Deprecated")
```

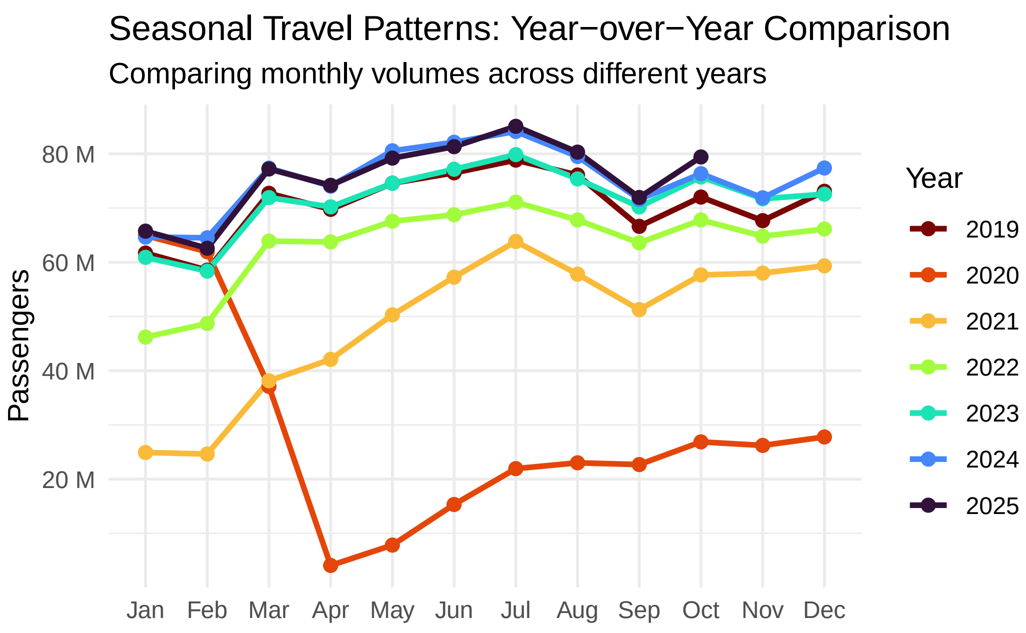
Table 6: TSA Passenger Throughput (2019-2025)

| Year | Total Passengers | YoY Growth | Avg Monthly | Median Monthly |
|---:|---:|---:|---:|---:|
| All years | 5,052,780,460 | — | 61,619,274 | 67,616,844 |
| 2019 | 848,102,043 | — | 70,675,170 | 72,367,869 |

| Year | Total Passengers | YoY Growth | Avg Monthly | Median Monthly |
|---|---|---|---|---|
| 2020 | 339,774,756 | -59.9% | 28,314,563 | 24,618,172 |
| 2021 | 585,250,987 | 72.2% | 48,770,916 | 54,258,962 |
| 2022 | 760,071,362 | 29.9% | 63,339,280 | 65,469,214 |
| 2023 | 858,548,196 | 13.0% | 71,545,683 | 72,240,801 |
| 2024 | 904,068,577 | 5.3% | 75,339,048 | 76,847,030 |
| 2025 | 756,964,539 | -16.3% | 75,696,454 | 78,205,370 |

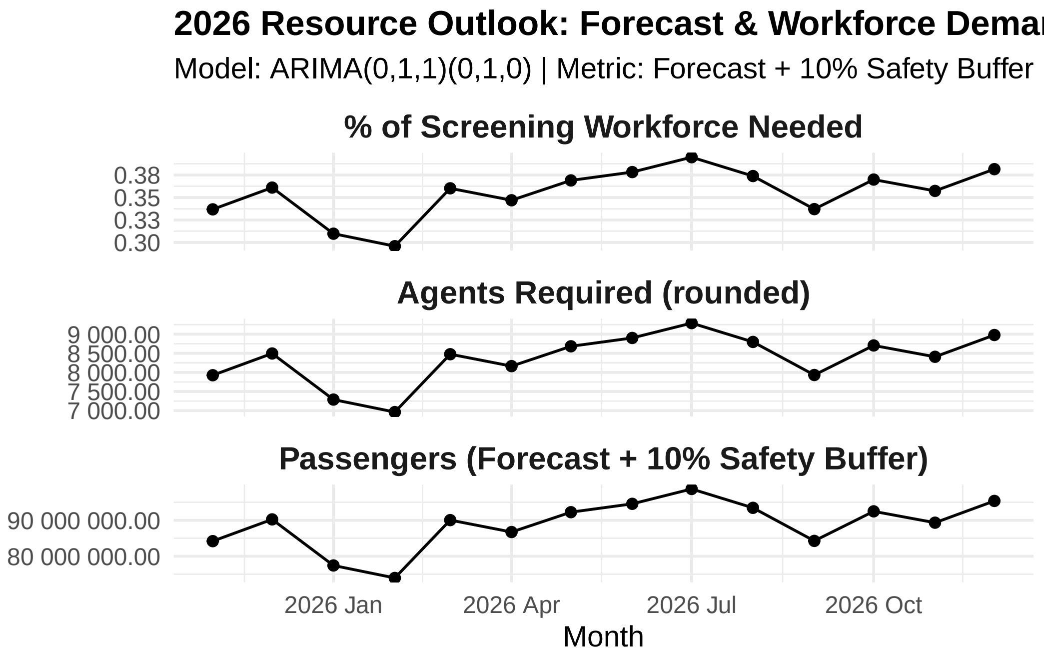
```r
tsa_clean %>%
  as_tibble() %>%
  mutate(
    Month_Label = month(Month, label = TRUE, abbr = TRUE),
    Month_Num = month(Month),
    Year_Factor = as.factor(year(Month))
  ) %>%
  ggplot(aes(x = Month_Label, y = Total, group = Year_Factor, color = Year_Factor)) +
  geom_line(size = 1) +
  geom_point(size = 2) +

  # Highlight 2019 (Baseline) and 2025 (Current)
  scale_color_viridis_d(option = "turbo", direction = -1) +

  # Format chart
  scale_y_continuous(labels = label_number(scale = 1e-6, suffix = " M")) +

  theme_minimal() +
  labs(
    title = "Seasonal Travel Patterns: Year-over-Year Comparison",
    subtitle = "Comparing monthly volumes across different years",
    x = "",
    y = "Passengers",
    color = "Year"
  )
```

```
Warning: Using `size` aesthetic for lines was deprecated in ggplot2 3.4.0.
i Please use `linewidth` instead.
```




```r
# Max passengers - 85M pax ~= 8000 agents
pax_per_agent_ratio <- 10625
total_staff_available <- 23530

# Refit ARIMA(0,1,1)(0,1,0) to full dataset
fit_final <- tsa_clean %>%
  model(
   ma1_manual = ARIMA(Total ~ pdq(0,1,1) + PDQ(0,1,0))
  )

# Forecast forward (covering remaining 2025 and all of 2026)
fc_2026 <- fit_final %>%
  forecast(h = "14 months")

# Calculate the Metrics
tsa_agents_forecast <- fc_2026 %>%
  as_tibble() %>%
  mutate(
    pax_forecast_high = .mean * 1.10,
    agents_required_ceiling = ceiling(pax_forecast_high / pax_per_agent_ratio),
    pct_of_screening_staff = agents_required_ceiling / total_staff_available
  ) %>%
  select(Month, pax_forecast_high, agents_required_ceiling, pct_of_screening_staff)
```

```
tsa_dual <- tsa_agents_forecast %>%
  transmute(
    Month,
    `Passengers (Forecast + 10% Safety Buffer)` = pax_forecast_high,
    `Agents Required (rounded)`                 = agents_required_ceiling,
    `% of Screening Workforce Needed`           = pct_of_screening_staff
  ) %>%
  pivot_longer(
    cols = -Month,
    names_to = "metric",
    values_to = "value"
  ) %>%
  drop_na(value)

ggplot(tsa_dual, aes(x = Month, y = value)) +
  geom_line() +
  geom_point() +
  facet_wrap(~ metric, scales = "free_y", ncol = 1) +
  labs(
    title = "2026 Resource Outlook: Forecast & Workforce Demand",
    subtitle = "Model: ARIMA(0,1,1)(0,1,0) | Metric: Forecast + 10% Safety Buffer",
    x = "Month",
    y = NULL
  ) +
  scale_y_continuous(labels = scales::label_number(accuracy = 0.01)) +  # <-- safer
  theme_minimal() +
  theme(
    strip.text = element_text(size = 12, face = "bold"),
    plot.title = element_text(face = "bold")
  )
```


2026 Resource Outlook: Forecast & Workforce Demar
Model: ARIMA(0,1,1)(0,1,0) | Metric: Forecast + 10% Safety Buffer
% of Screening Workforce Needed
0.38
0.35
0.33
0.30
Agents Required (rounded)
9 000.00
8 500.00
8 000.00
7 500.00
7 000.00
Passengers (Forecast + 10% Safety Buffer)
90 000 000.00
80 000 000.00
2026 Jan
2026 Apr
2026 Jul
2026 Oct
Month