OM 286 Analytics for Supply Chain and Distribution Planning Project:
Forecasting Restaurant Ingredient Usage with Bayesian Modeling, Exponential Smoothing, and ARIMA

Prepared For:

Dr. Genaro Gutierrez
McCombs School of Business
The University of Texas at Austin
2110 Speedway Austin, TX 78705

Prepared by:

Jenna Ferguson
Carissa Ing
Haden Loveridge
Vishwa Patel
Ethan Wong

Thursday, December 12th, 2024

EXECUTIVE SUMMARY

This report evaluates three advanced time series forecasting methods - Bayesian Modeling, Exponential Smoothing (ETS), and ARIMA - to address La Casita's critical need for accurate demand forecasting of perishable ingredients. As a traditional-style Mexican restaurant in Springville, Utah, La Casita faces increasing challenges in aligning supply with demand due to changing consumption patterns post-COVID. Accurate forecasts for key ingredients - shrimp, steak, and tomatoes - are essential to minimize food waste, prevent stockouts, and improve inventory management.

Bayesian Modeling utilized a Poisson likelihood framework with Fourier terms to account for weekly and seasonal patterns. While this model demonstrated strong performance for shrimp and steak with relatively low RMSE values, it struggled with the variability inherent in tomato sales, resulting in wide confidence intervals. Despite these limitations, the Bayesian approach effectively captured long-term trends and provided robust uncertainty quantification.

Exponential Smoothing (ETS) models, selected based on AICc values, significantly improved short-term forecasting accuracy over the Bayesian models. The optimal ETS configurations - MNM for shrimp, MAA for steak, and ANA for tomatoes - strongly aligned with observed sales patterns, achieving lower RMSE values for shrimp and steak. However, the tomato model exhibited larger errors due to its high variability. Notably, ETS models provided flexibility in capturing both additive and multiplicative seasonality, making them highly adaptable to the specific characteristics of each ingredient.

ARIMA, particularly the *auto.arima()* implementation emerged as the most effective method for all three ingredients. Including weekly seasonality and optimized parameter selection allowed ARIMA to achieve the lowest RMSE values and tighter confidence intervals compared to Bayesian and ETS models. Shrimp and steak models effectively captured sales trends and seasonality, while the tomato model demonstrated superior handling of variability, outperforming other methods significantly. Cross-validation further validated ARIMA's robustness, showing consistent accuracy improvements across forecast horizons.

Compared to Naive Seasonal Forecasting, all three advanced methods demonstrated substantial reductions in RMSE. However, ARIMA consistently outperformed Bayesian and ETS models in short and long-term forecasts. This superiority was particularly evident for tomatoes, where the auto.arima() model effectively handled the high variance and complex seasonality of sales data.

In summary, ARIMA offers the most robust and reliable forecasting solution for La Casita's needs. By leveraging its ability to capture intricate seasonal patterns and minimize residual variance, ARIMA provides La Casita with a practical tool to optimize inventory management, reduce waste, and improve customer satisfaction.

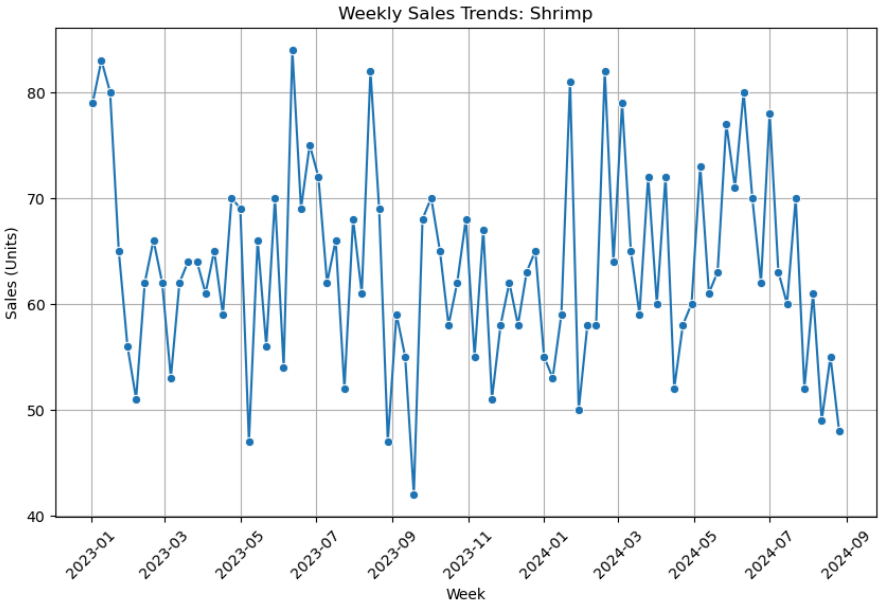
I. TABLE OF CONTENTS

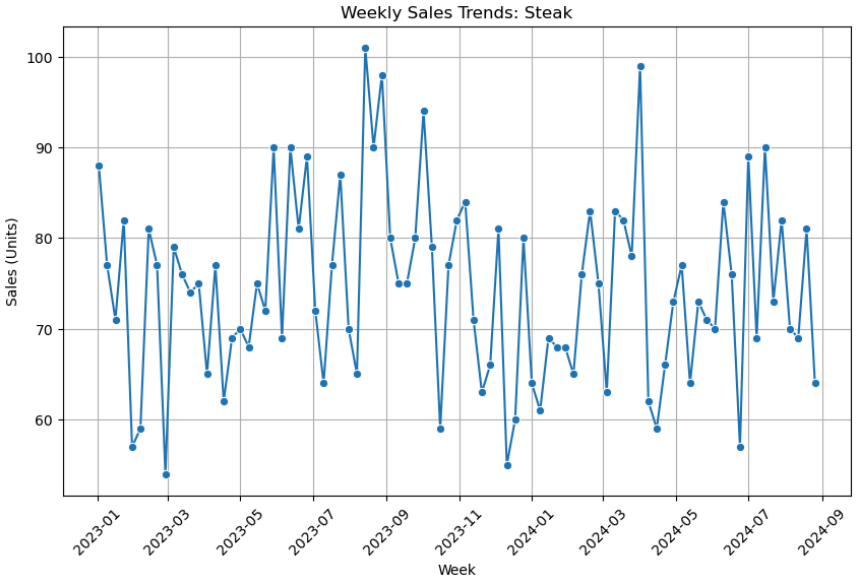
GitHub Repository: https://github.com/ethanrwong/Supply-Chain-Analytics-Group-Project

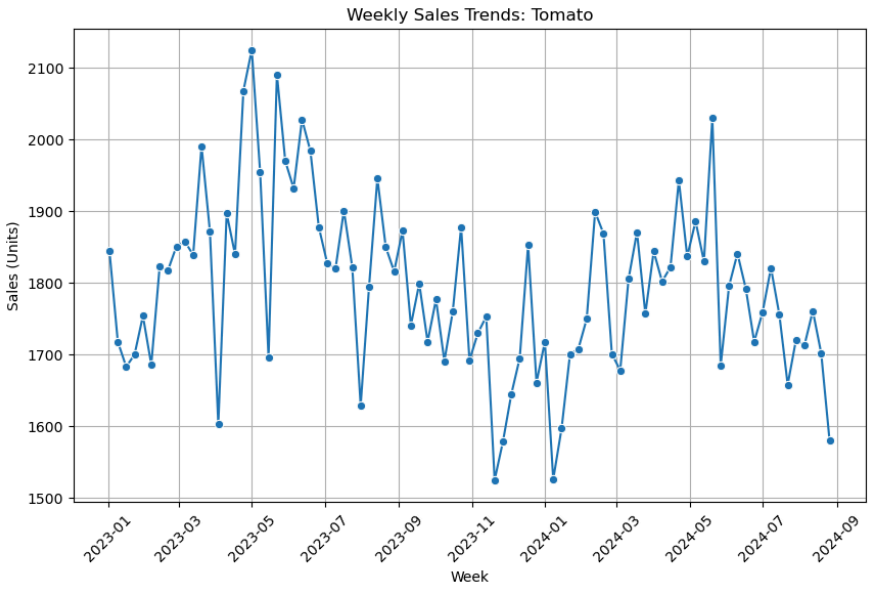
II. DEFINING THE BUSINESS PROBLEM

For any restaurant, being able to forecast demand for perishable goods is a critical challenge. These items have a limited shelf life, and any mismatch between supply and demand carries financial implications. Overstocking leads to waste, while understocking runs the risk of dissatisfied customers and missed sales opportunities. Effective demand forecasting allows for better inventory management and increased operational efficiency. By leveraging precise forecasting techniques, we can help La Casita improve its purchasing strategy, minimize costs, and align its inventory with anticipated demand patterns. This is especially important in the restaurant industry, where the price of perishable goods directly impacts the bottom line.

La Casita is a restaurant that serves traditional-style Mexican food. It is located in Springville, Utah, and has operated for over 46 years. During that time, their key to success was a consistent customer base that showed consistent patterns in purchasing, leading to a relatively simple pattern of supply and demand. However, the consistency has shifted due to increased online demand post-Covid and other unknown factors. From the dozens of possible daily ingredients, we focused our analysis on three main ingredients: tomatoes, shrimp, and steak. We chose shrimp and steak as ingredients because 1) they have a relatively short shelf life compared to items such as alcohol, rice, or beans; 2) we expected to see weekly, monthly, or quarterly seasonality; and 3) they pose the most significant risk /reward of poorly calculated forecasts due to their high price to purchase and the high price of the dishes sold to customers (high loss for understocking as well as overstocking). Tomatoes were selected as an ingredient because they still pose a risk of under or over-stocking due to short shelf life and their frequent use in multiple plates, but they also offer a more consistent pattern relative to other ingredients due to their versatility and inclusion in the majority of dishes sold. The data was manually extracted from La Casita's point-of-sale platform, Heartland, by retrieving daily sales records from January 1, 2023, through August 31, 2024. We recorded the sales quantities for each day's three items of interest and compiled them into a CSV file. This manual extraction process ensured the data was clean from the outset, eliminating the need for additional preprocessing. Notably, La Casita is closed on Sundays. Time plots of the sales of these three ingredients each week are shown below.


Weekly Sales Trends: Shrimp
Sales (Units)
Week
40
50
60
70
80
2023-01
2023-03
2023-05
2023-07
2023-09
2023-11
2024-01
2024-03
2024-05
2024-07
2024-09


Weekly Sales Trends: Steak
Sales (Units)
Week
60
70
80
90
100
2023-01
2023-03
2023-05
2023-07
2023-09
2023-11
2024-01
2024-03
2024-05
2024-07
2024-09



III. EXPLORING DIFFERENT TIME SERIES TECHNIQUES

Bayesian Modeling

We implemented a Bayesian time series model using Stan and R to forecast daily sales of shrimp, steak, and tomatoes at a Mexican restaurant. In the Stan model, we began by defining the data structure, which included the number of observations, forecast days, day-of-week indices, time indices, and Fourier terms to capture seasonality. We specified parameters such as the intercept, day-of-week effects, trend coefficient, and Fourier coefficients. We transformed the day-of-week effects to include a baseline effect for a reference day. We defined priors for these parameters and used a Poisson likelihood to model the sales counts. We produced posterior predictive samples in the generated quantities block to forecast future sales. We arrived at this "final" model through experimentation by testing the accuracy of various model configurations. Below is a snapshot of the Stan code to create the data and initial parameters:

```
  data {
    int<lower=1> N;              // number of observations
    int<lower=1> N_forecast;     // number of days to forecast
    int<lower=1> K_dow;          // number of day-of-week categories
    int<lower=1,upper=K_dow> dow[N];    // day-of-week index for historical data
    int<lower=1,upper=K_dow> dow_forecast[N_forecast]; // day-of-week index for forecasts
    int<lower=0> y[N];           // observed counts
    real t[N];                   // scaled time index
    real t_forecast[N_forecast]; // scaled future time points
    int<lower=1> S;              // number of Fourier terms
    matrix[N, 2 * S] fourier_terms; // Fourier terms for historical data
    matrix[N_forecast, 2 * S] fourier_terms_forecast; // Fourier terms for forecast
  }


  parameters {
    real alpha;                          // intercept
    vector[K_dow-1] beta_raw;            // day-of-week effects (except baseline)
    real delta;                          // trend coefficient
    vector[2 * S] gamma;                 // Fourier coefficients
  }

transformed parameters {
  vector[K_dow] beta;
  beta[1] = 0; // baseline day of week (e.g., Monday)
  for (i in 2:K_dow) {
    beta[i] = beta_raw[i-1];
  }
}

model {
  // Priors
  alpha ~ normal(0, 5);
  beta_raw ~ normal(0, 2);
  delta ~ normal(0, 2); //was 1, changed to 2
  gamma ~ normal(0, 1); // Prior for Fourier coefficients, was 0.5, changed to 1
  // Likelihood
  for (n in 1:N) {
    real log_lambda = alpha + beta[dow[n]] + delta * t[n] + dot_product(gamma, fourier_terms[n]);
    target += poisson_log_lpmf(y[n] | log_lambda);
  }
}

generated quantities {
  vector[N_forecast] y_pred;
  for (f in 1:N_forecast) {
    real log_lambda_forecast = alpha + beta[dow_forecast[f]] + delta *
    t_forecast[f] + dot_product(gamma, fourier_terms_forecast[f]);
    y_pred[f] = poisson_rng(exp(log_lambda_forecast));
  }
}
```

The final model contains an intercept, 6 coefficients for the days of the week (one day is the baseline, so it's not included), a trend coefficient, and Fourier coefficients for seasonality.

In R, we prepared the data by creating day-of-week indices and generating Fourier terms for annual seasonality. Although the restaurant is closed on Sundays, we left them in the dataset. The model perfectly accounted for the 0 unit sales on those days, so altering the data was unnecessary. After splitting the data into training and validation sets and scaling the time indices to improve numerical stability, we bundled the processed data into a list. We passed it to the Stan model for fitting. We had 20 months of data available, so we chose an 80/20 split with Jan 2023-Apr 2024 as the training data and May-Aug 2024 as the validation data. After extracting posterior samples, we generated predictions and evaluated them using Mean Absolute Error (MAE) and Root Mean Square Error (RMSE).

The likelihood of the model was defined using a Poisson distribution, where the logarithm of the expected sales count was expressed as a combination of the intercept, seasonality, trend, and

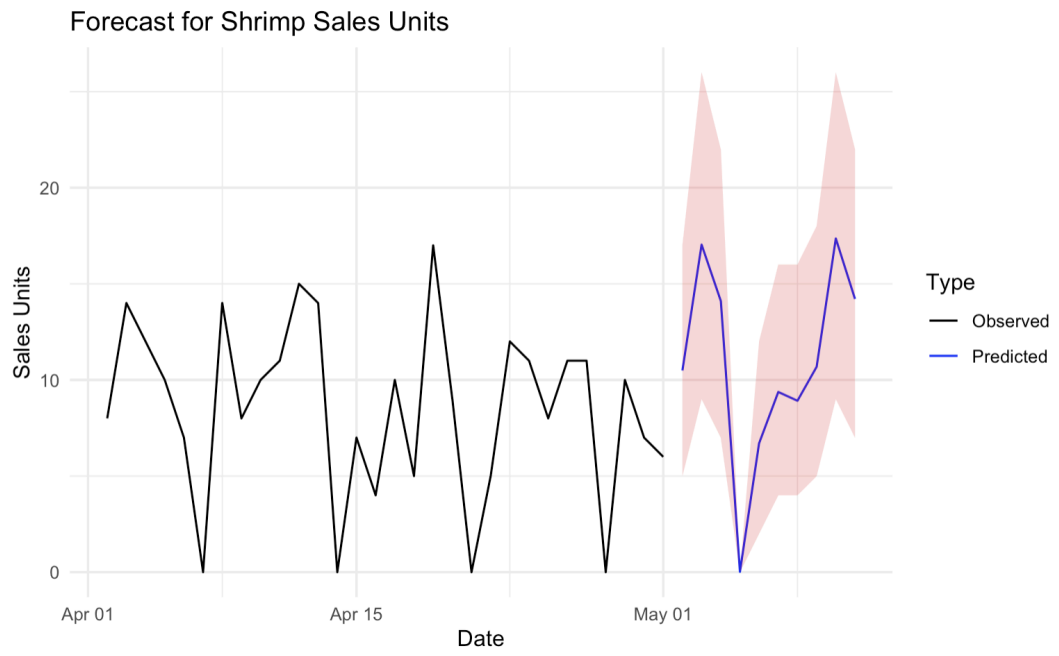
day-of-week effects. We applied priors to regularize parameter estimates, preventing overfitting and ensuring the model remained interpretable. We conducted posterior predictive checks using the generated quantities block to evaluate the model, providing forecasts for the validation period and uncertainty quantification.

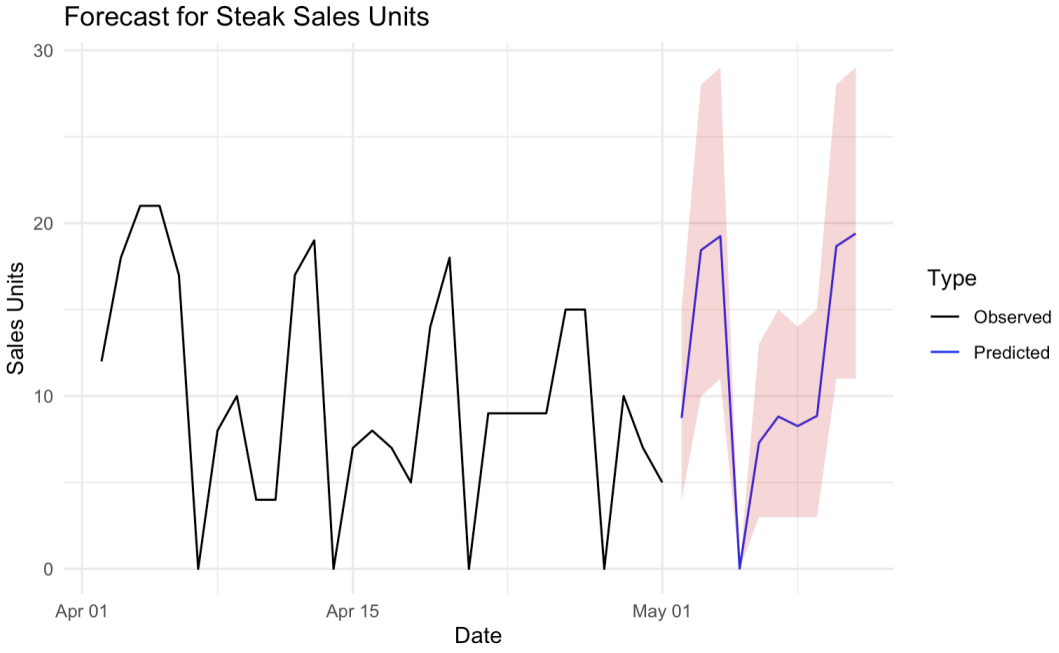
We found an extensive range in the lower and upper bounds of the 95% confidence interval for all three ingredients. For example, the 95% confidence interval bounds for the predicted value of shrimp sales on May 2, 2024, were 5 and 17, respectively. However, our dataset's overall range of shrimp sales (ignoring Sundays when the restaurant is closed) is 1 to 28. Similarly, large confidence intervals existed for the steak and tomato data. However, despite this, the RMSE values were relatively low, and the model performed somewhat well. For example, the standard deviation of shrimp sales is 5.00, while our model achieved a 4.04 RMSE value.

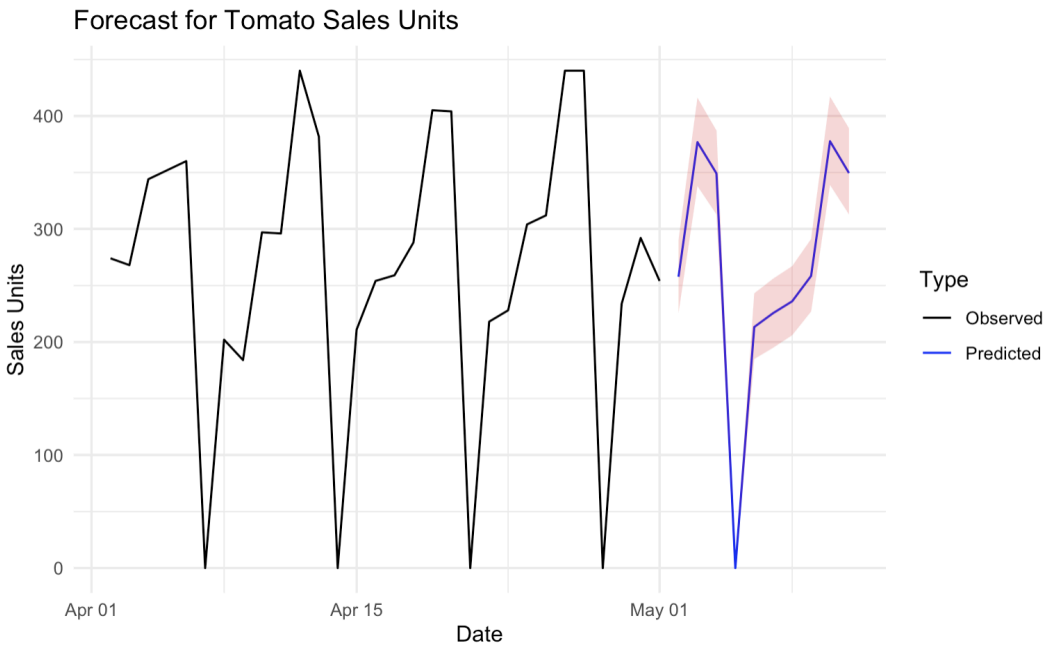
Below are the RMSE and MAE values for all three ingredients:

Ingredient	**MAE**	**RMSE**
Shrimp	3.061405	4.039198
Steak	2.545723	3.785547
Tomato	30.7557	41.1707

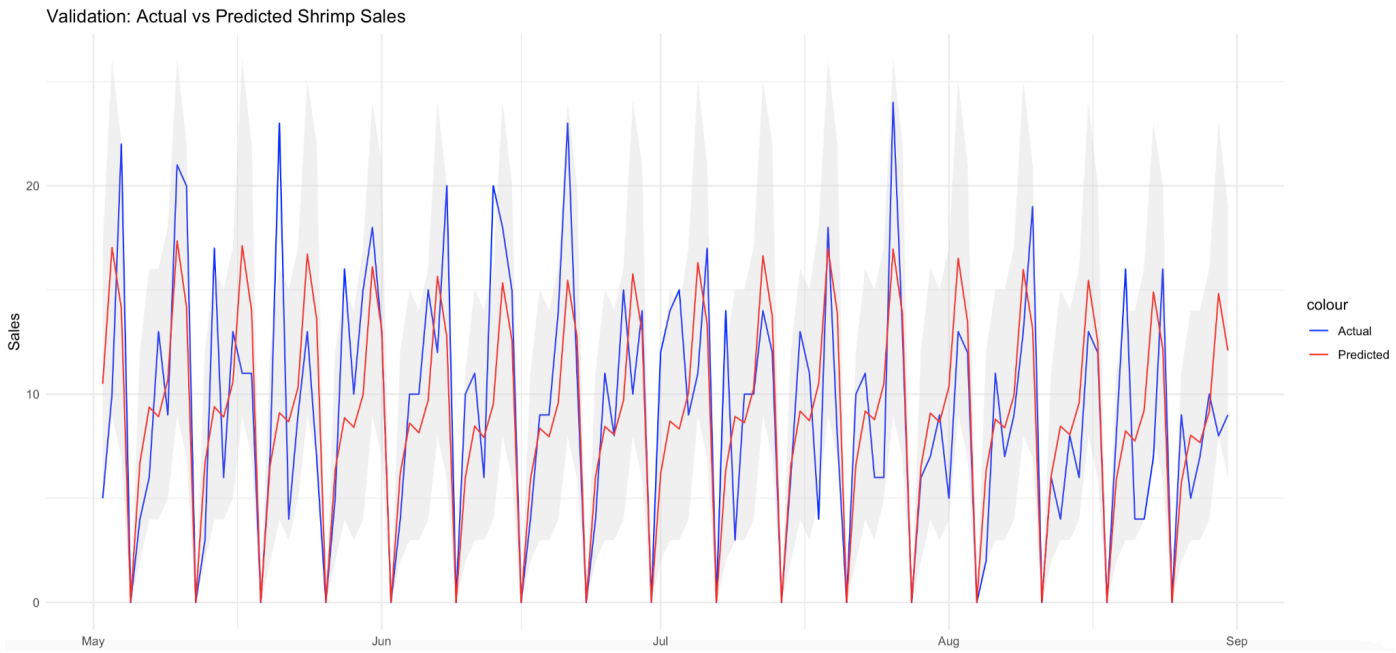
Below are 14-day forecasts for shrimp, steak, and tomato from the validation data set. The red region shows the previously mentioned large confidence interval.


Forecast for Shrimp Sales Units
Sales Units
20
10
0
Apr 01
Apr 15
May 01
Date
Type
Observed
Predicted


Forecast for Steak Sales Units
Sales Units
30
20
10
0
Apr 01
Apr 15
May 01
Date
Type
Observed
Predicted



Below is a more long-term forecast and comparison of the predicted versus actual sales unit values for shrimp using the Bayesian model. This was done on the validation set of May-Aug 2024. The gray region in this graph represents the confidence interval, the red line represents the predicted values, and the blue line represents the actuals. We can see that the model lags in certain areas, overpredicting the number of sales 1-2 days after the spike occurred. Although there is room for improvement, we had challenges making this model more accurate, as it's clear that the blue spikes (actual sales) are somewhat irregular and unpredictable when viewed over a longer timeframe.



Exponential Smoothing

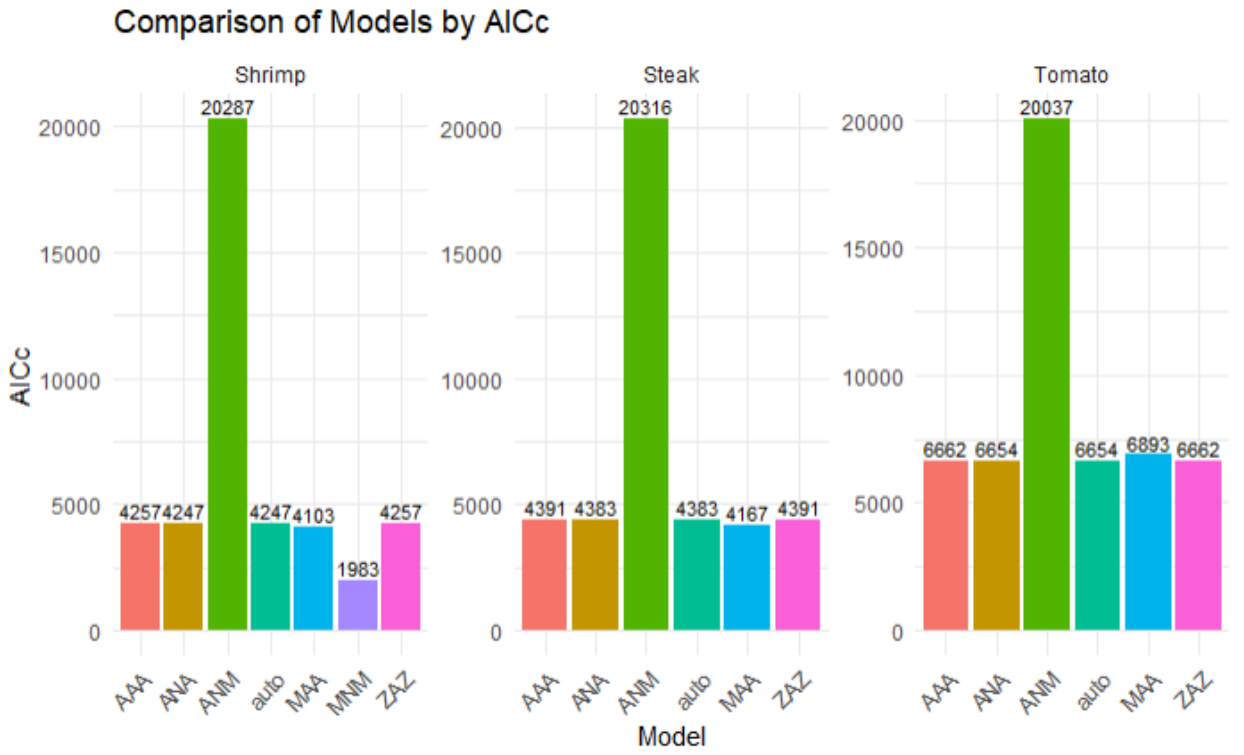
To better capture weekly and monthly trends in sales, we decided to test exponential smoothing (ETS) models for each ingredient. This approach aims to account for seasonality and trends that may affect sales patterns over time. Using the training period from January 2023 to April 2024, we evaluated various ETS models to determine the best fit for shrimp, steak, and tomato demand. Each model was assessed based on its ability to minimize forecast errors, using metrics such as AICc, RMSE, and MAE, and its capacity to align with observed sales patterns during the training period.

```
m_shrimp <- train %>%
  filter(Ingredient == "Shrimp") %>%
  model(m.auto = ETS(Sales_Units),
        m.ANA = ETS(Sales_Units ~ error("A") + trend("N") + season("A")),
        m.ANM = ETS(Sales_Units ~ error("A") + trend("N") + season("M")),
        m.MNM = ETS(Sales_Units ~ error("M") + trend("N") + season("M")),
        m.AAA = ETS(Sales_Units ~ error("A") + trend("A") + season("A")),
        m.MAA = ETS(Sales_Units ~ error("M") + trend("A") + season("A")),
        m.ZAZ = ETS(Sales_Units ~ trend("A")),
        m.ZMZ = ETS(Sales_Units ~ trend("M")))
```

* Repeated for “Steak” and “Tomato”

Testing these ETS models for tracking restaurant sales helps identify the factors influencing sales, such as seasonality, trends, and error types. Models like `*m.ANA*` and `*m.ANM*` analyze whether sales follow seasonal patterns, while models like `*m.ZAZ*` and `*m.ZMZ*` focus on longer-term trends without considering seasonal changes. The letter codes in these models represent the components of the ETS framework: error (A for additive or M for multiplicative), trend (A for additive, M for multiplicative, or N for none), and seasonality (A for additive, M for multiplicative, or N for none). Models such as `*m.AAA*` and `*m.MAA*` compare different error structures to determine whether sales variability is better explained by constant or proportional changes, improving forecasts' reliability. Including an automated model (`*m.auto*`) provides a baseline for evaluating whether manually selected models offer better accuracy based on specific knowledge of shrimp sales.

The AICc (Corrected Akaike Information Criterion) provides a reliable metric for selecting the best ETS model for forecasting ingredient sales. The AICc balances model fit with complexity, penalizing models that overfit data by incorporating too many parameters. This ensures the selected model explains the observed data well and effectively generalizes unseen data. Among competing ETS models, the one with the lowest AICc is considered the best, representing the optimal trade-off between simplicity and accuracy.



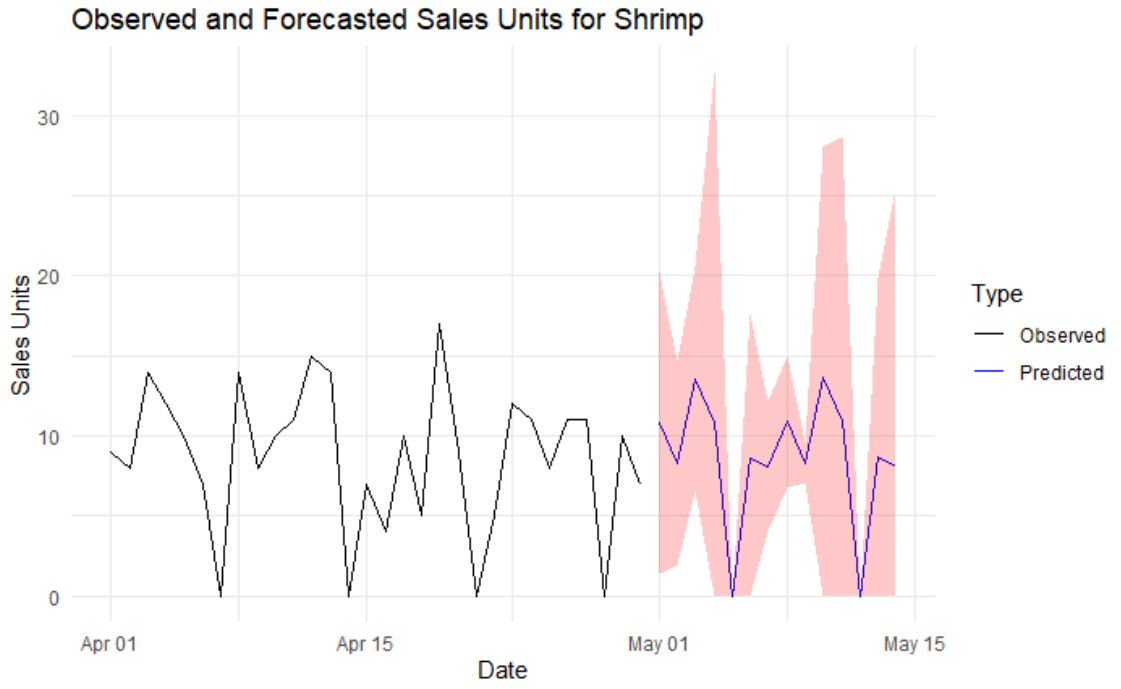
Based on the graph, the optimal models were identified as MNM for Shrimp, MAA for Steak, and ANA for Tomato since these had the lowest AICc values. Interestingly, only the model for Tomato aligned with the automatic selection (‘*m.auto*’), highlighting that for Shrimp and Steak, the automated model selection might not have chosen the best-fit model. This suggests that while automated methods provide a good baseline, manual inspection of AICc values can offer more tailored and accurate model choices for specific cases, like when modeling shrimp and steak dish sales.

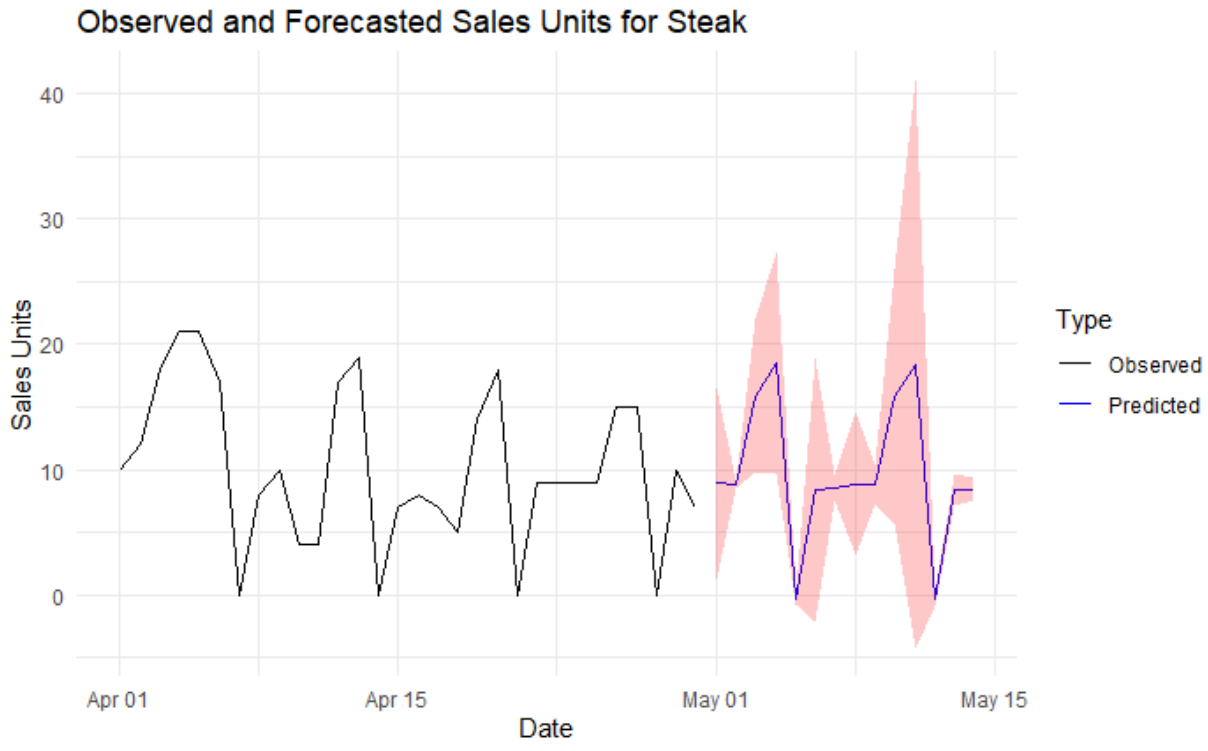
Using the selected models, we forecasted sales for the next 120 days (4 months) for shrimp, steak, and tomato dishes and compared these predictions to the actual sales data. The table shows the accuracy metrics - MAE and RMSE - for each ingredient's model. Shrimp and steak models performed well, with low error values (e.g., Shrimp RMSE = 4.511 and Steak RMSE = 3.970), indicating strong alignment between forecasted and actual sales. However, the tomato model exhibited significant errors (MAE = 29.001 and RMSE = 38.739), which is particularly concerning given that tomato sales typically range between 200 and 400 units. This indicates that the model's errors are proportionally large and suggest a poor fit.

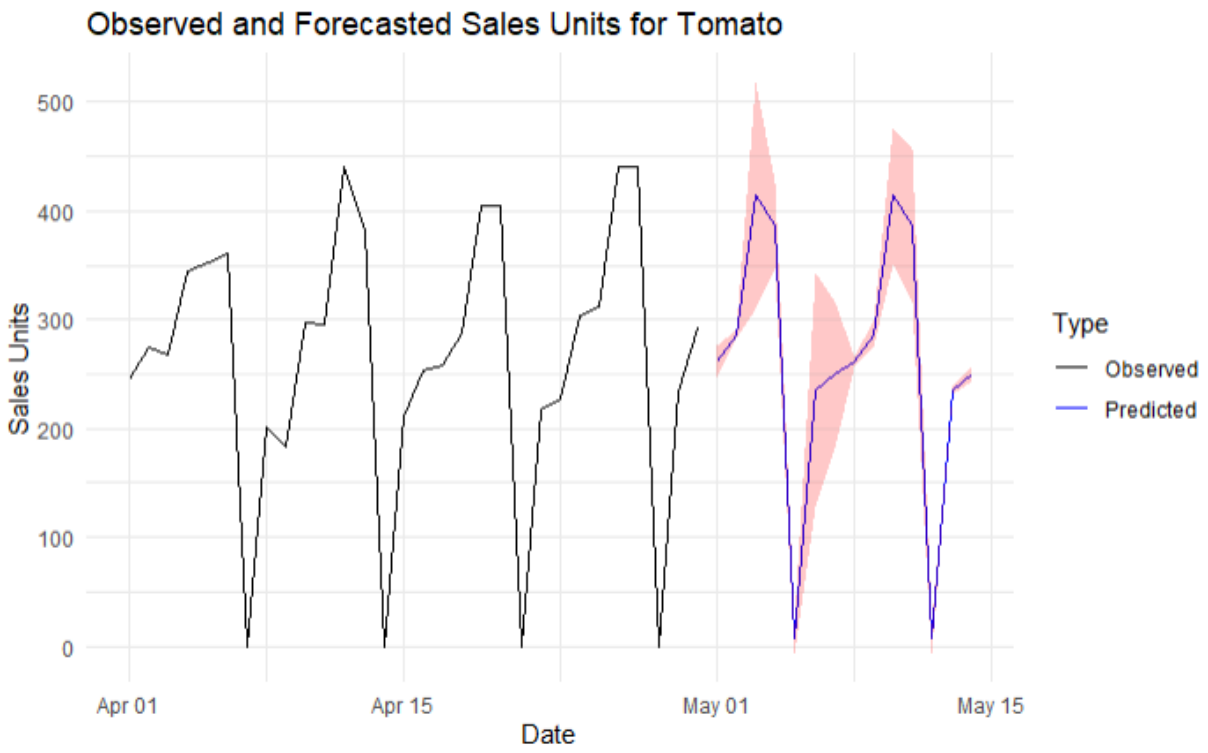
Ingredient	**MAE**	**RMSE**
Shrimp	3.437245	4.510853

Steak	2.791849	3.969921
Tomato	29.00137	38.73852

The following graphs show 14-day forecast graphs for shrimp, steak, and tomato show observed versus predicted sales units, with red shading representing the confidence intervals:

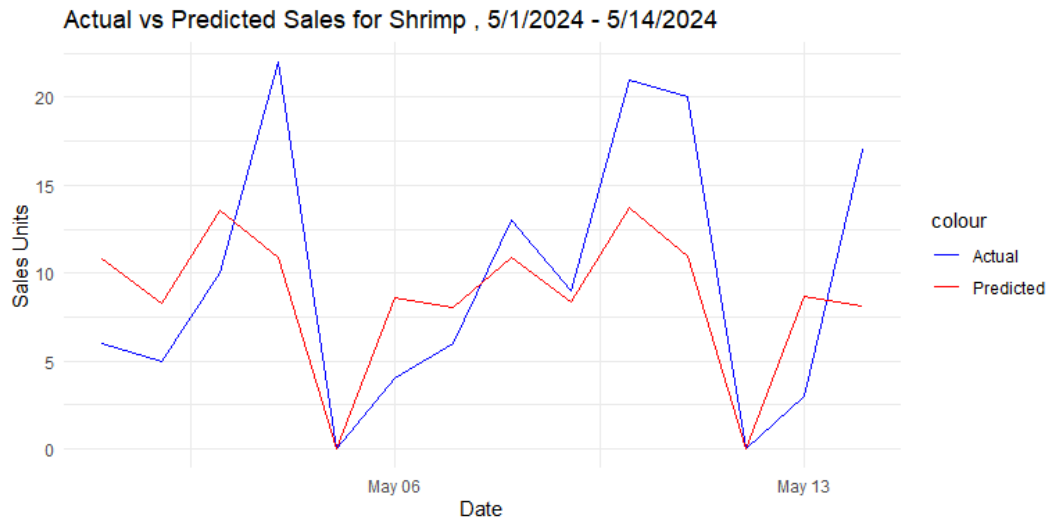


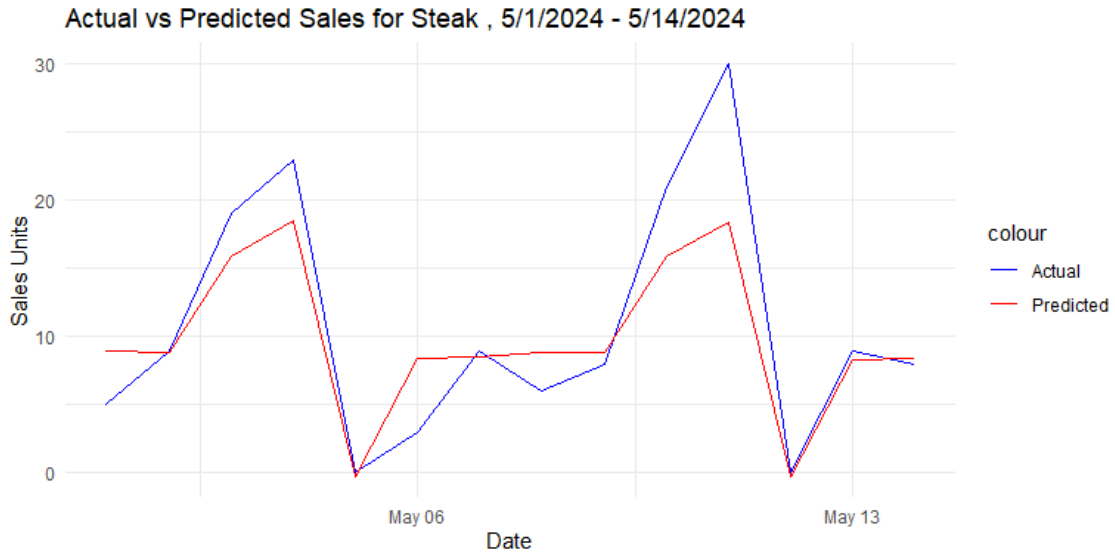


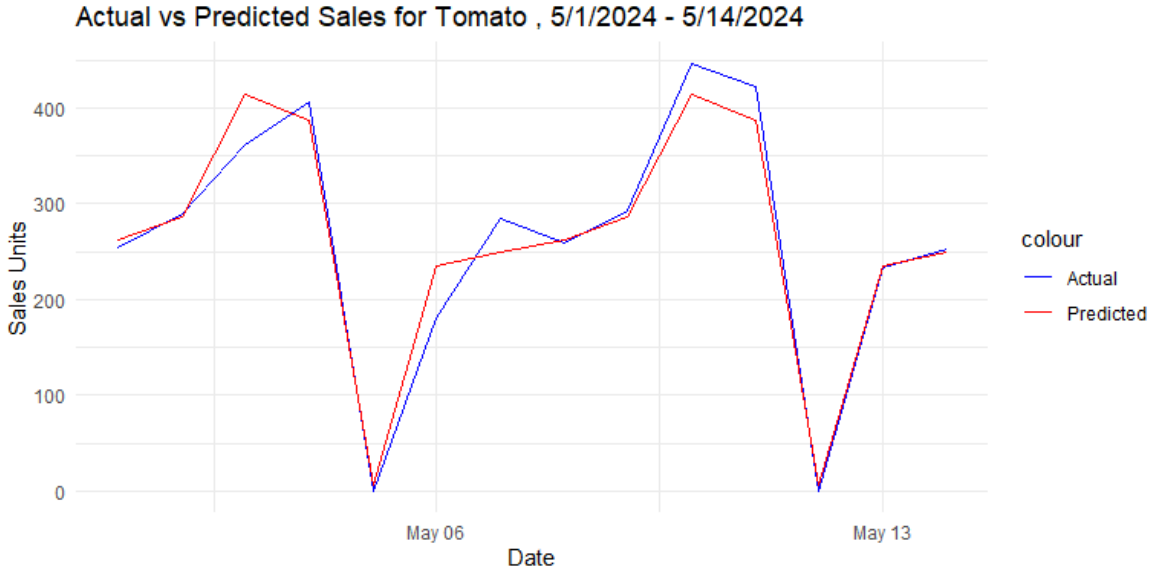


A notable feature is the large confidence intervals, particularly for tomato, which reached a maximum range of 214.23739 - far exceeding those for shrimp (32.68499) and steak (41.09970). These wide intervals suggest substantial uncertainty in the model's predictions, especially for Tomato, which aligns with its high RMSE and MAE values (38.739 and 29.001, respectively). This further demonstrates the model's difficulty in capturing the variability of tomato sales.

When comparing the actual and predicted sales for the first two weeks following the known demand period, the models successfully capture the overall trends in sales but struggle to predict peaks accurately. For instance, on May 4, 2024, actual shrimp dish sales were 22 units, while the forecast predicted only 11 dishes, illustrating a significant underestimation of peak sales. This pattern is particularly pronounced for shrimp and steak, where the models consistently fail to match high-demand days accurately.


Actual vs Predicted Sales for Shrimp , 5/1/2024 - 5/14/2024
Sales Units
20
15
10
5
0
May 06
May 13
Date
colour
Actual
Predicted

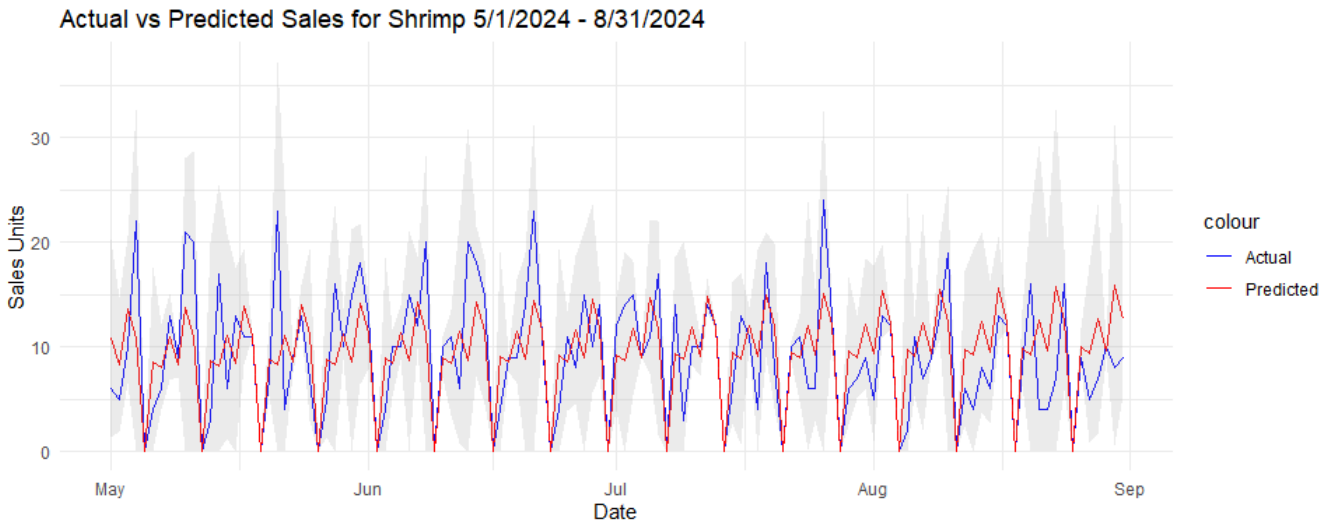

Actual vs Predicted Sales for Steak , 5/1/2024 - 5/14/2024
Sales Units
30
20
10
0
May 06
May 13
Date
colour
Actual
Predicted

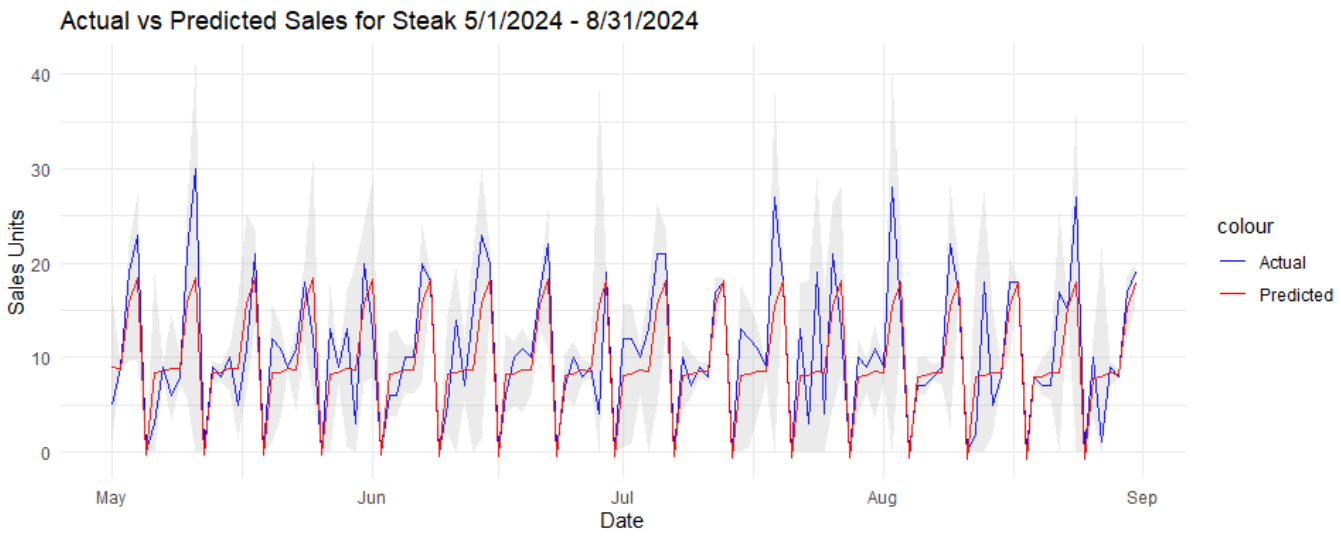

Actual vs Predicted Sales for Tomato , 5/1/2024 - 5/14/2024
Sales Units
400
300
200
100
0
May 06
May 13
Date
colour
Actual
Predicted

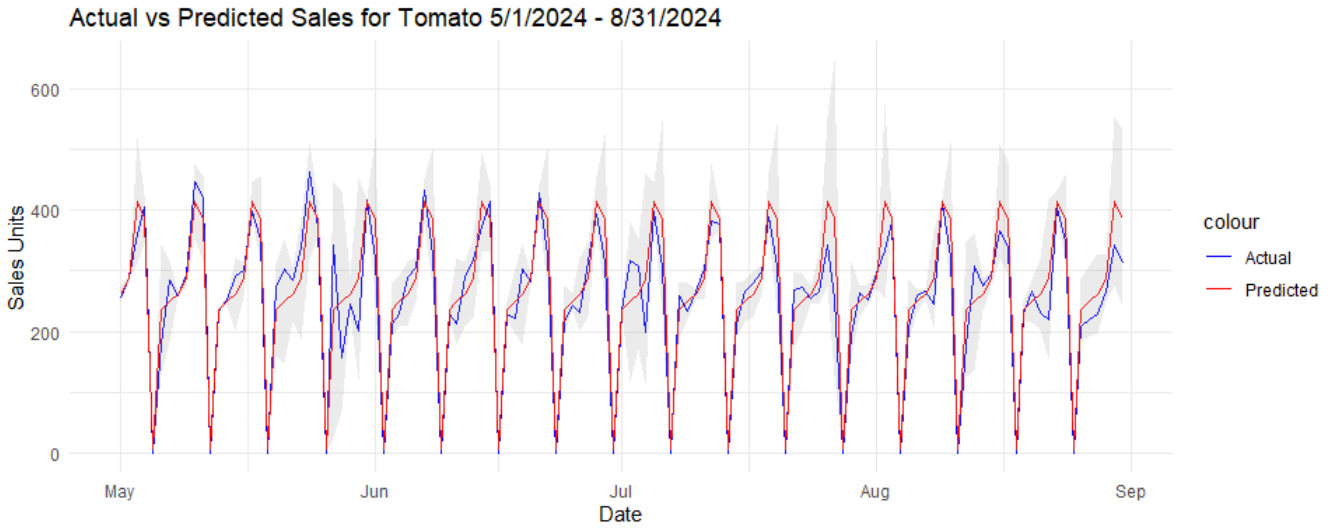
Surprisingly, the tomato model performs relatively well regarding trend alignment despite its high RMSE, MAE, and the wide confidence intervals observed. This suggests that while the model struggles with precision, especially in estimating the exact sales values, it still provides reasonable directional forecasts. This performance discrepancy indicates that while Shrimp and Steak models may need further refinement to address peak mismatches, the Tomato model's ability to approximate trends despite inherent uncertainties could still offer some value for long-term sales planning.

When the models are extended to forecast sales for a longer period of up to four months, the patterns observed in short-term forecasting persist. The shrimp model frequently overestimates sales on low-demand days but needs to be more accurate in predicting peak sales during high-demand periods. This leads to a lack of reliability in predicting peak sales. This could present significant challenges for inventory and supply chain management, particularly during peak periods when underestimations could result in unmet demand.

The steak model performs slightly better on days with low to mid-level sales than the shrimp model but still struggles with underestimating sales on high-demand days. This suggests that while the model can more effectively capture moderate trends, it requires further adjustments to handle peaks. Conversely, the tomato model continues to perform well, maintaining its ability to align forecasts closely with actual sales even over the extended forecasting horizon. Despite high overall errors and confidence intervals, this consistency makes it a relatively dependable tool for long-term sales planning for Tomato dishes.







ARIMA

To build an ARIMA model, we first decided to determine if differencing was necessary to achieve stationarity. Visually, based on the time series plot in ***Defining the Business Problem***, we did not think differencing was required. However, we first wanted to confirm this with a Kwiatkowski - Phillips - Schmidt - Shin (KPSS) test. We did this based on the first and second differences in our data along with the original series.

```
data_tsibble %>%
  features(Sales_Units, unitroot_kpss)
```

```
## # A tibble: 3 × 3
##   Ingredient kpss_stat kpss_pvalue
##   <chr>          <dbl>       <dbl>
## 1 Shrimp        0.0877         0.1
## 2 Steak         0.123          0.1
## 3 Tomato        0.685          0.0149
```

```
data_diff %>%
  features(diff_1, unitroot_kpss)
```

```
## # A tibble: 3 × 3
##   Ingredient kpss_stat kpss_pvalue
##   <chr>          <dbl>       <dbl>
## 1 Shrimp        0.0295         0.1
## 2 Steak         0.0278         0.1
## 3 Tomato        0.0938         0.1
```

```
data_diff %>%
  features(diff_2, unitroot_kpss)
```

```
## # A tibble: 3 × 3
##   Ingredient kpss_stat kpss_pvalue
##   <chr>          <dbl>       <dbl>
## 1 Shrimp        0.0197         0.1
## 2 Steak         0.0219         0.1
## 3 Tomato        0.0584         0.1
```

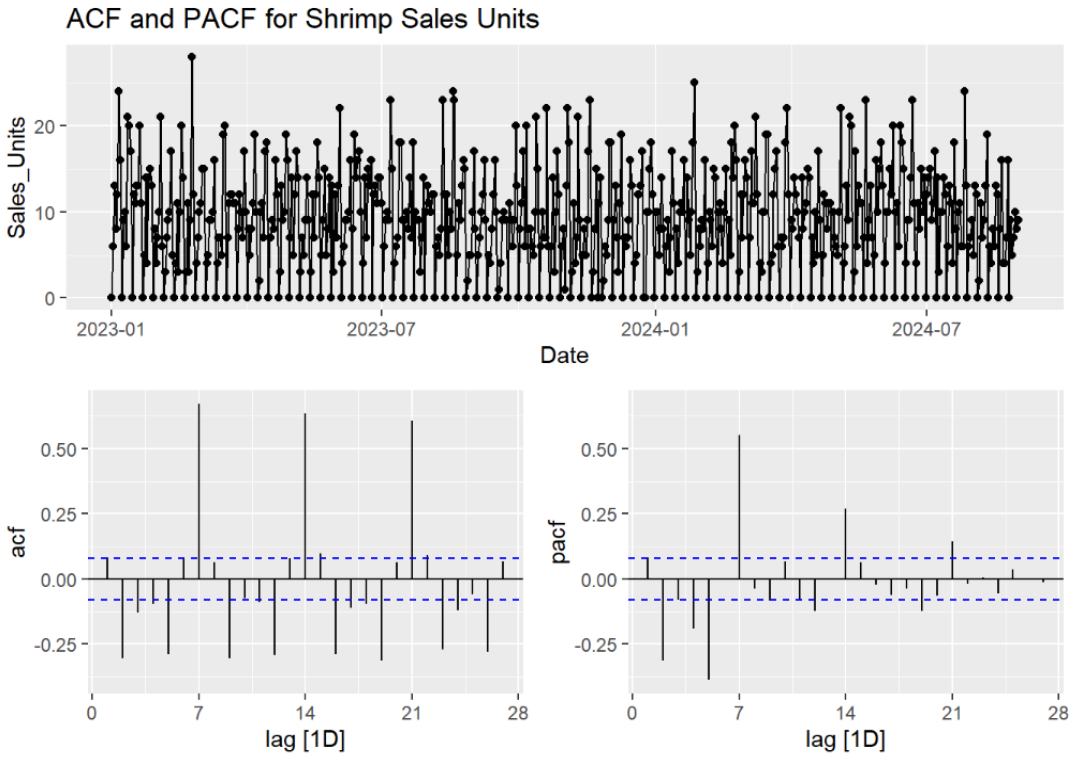
The KPSS test's null hypothesis is that the series is stationary. Since the shrimp and steak data on the original series have p-values> 0.05, we cannot reject stationarity. However, non-stationarity for the tomato data is possible, so we also decided to perform an Augmented Dickey-Fuller (ADF) test.

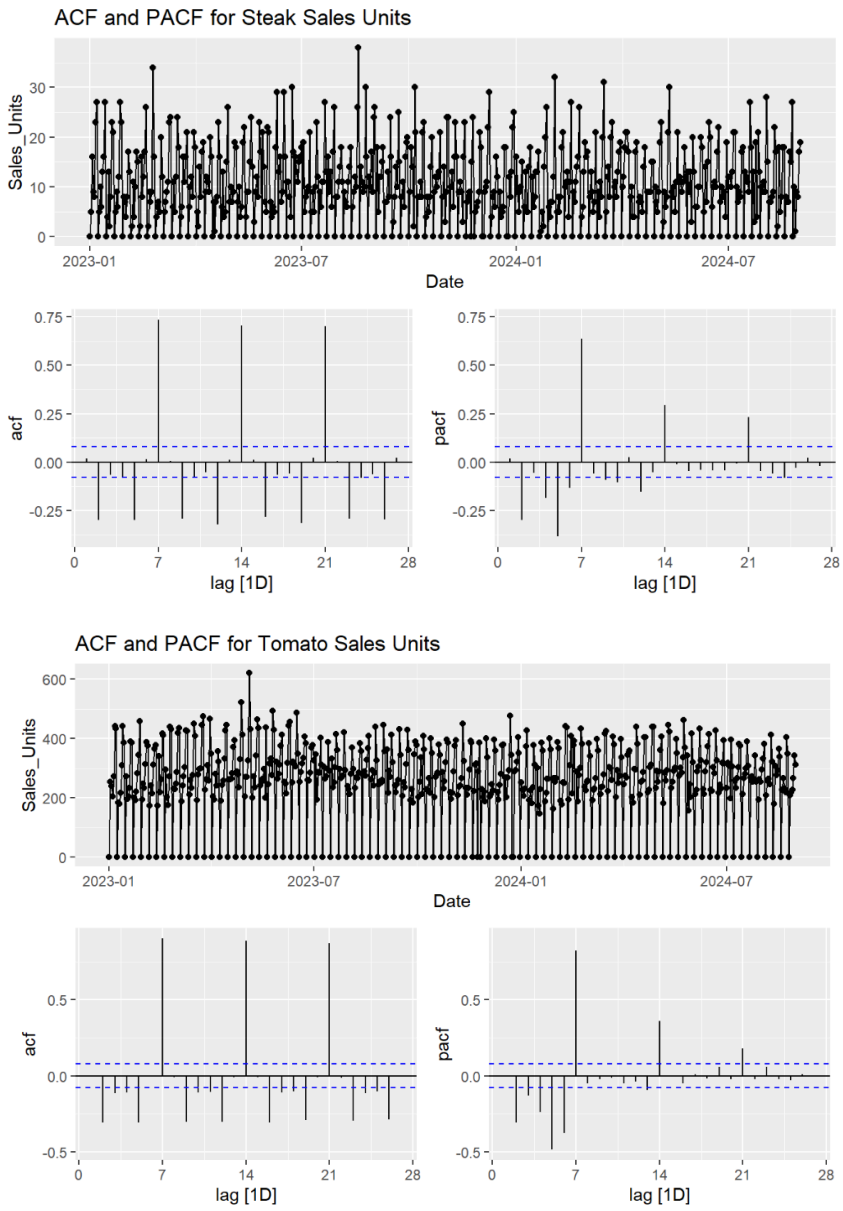
```
adf_results <- data_diff %>%
  group_by(Ingredient) %>%
  group_modify(~ {
    # ADF test on original series
    adf_original <- adf.test(.x$Sales_Units, alternative = "stationary")

    # ADF test on first difference
    adf_diff_1 <- adf.test(na.omit(.x$diff_1), alternative = "stationary")

    # ADF test on second difference
    adf_diff_2 <- adf.test(na.omit(.x$diff_2), alternative = "stationary")

    # Return results as a tibble
    tibble(
      adf_p_value_original = adf_original$p.value,
      adf_p_value_diff_1 = adf_diff_1$p.value,
      adf_p_value_diff_2 = adf_diff_2$p.value
    )
  })

print(adf_results)
```

```
## # A tibble: 3 × 4
## # Groups:   Ingredient [3]
##   Ingredient adf_p_value_original adf_p_value_diff_1 adf_p_value_diff_2
##   <chr>                     <dbl>              <dbl>              <dbl>
## 1 Shrimp                     0.01               0.01               0.01
## 2 Steak                      0.01               0.01               0.01
## 3 Tomato                     0.01               0.01               0.01
```

For the ADF test, the null hypothesis is that the series is not stationary. Since the p-values are < 0.05, we can reject the null hypothesis, even at the original series level. After this ADF test, we believed that differencing was likely unnecessary for the steak and shrimp data and that the original series for those data was stationary. However, since the KPSS and ADF results disagreed for tomato, we suspected that the *auto.arima()* function may use d = 1 for that series. Our goal was to compare the *auto.arima()* results to the ARIMA models we hypothesize will be a good fit for each of the three ingredients' data.

Next, we examined the ACF and PACF for the shrimp, steak, and tomato sales units to determine the order of appropriate ARIMA models.





Based on these three sets of plots, all three ingredients show strong weekly seasonality in their sales data, indicated by the periodic spikes at multiples of 7 in the ACF plots. Significant spikes at lag 1 in the PACF plots across all three ingredients indicated that we should include an AR(1) component in the models. We hypothesized the following model fits for each ingredient:

Shrimp: SARIMA(p=1, d=0, q=0)(P=1, D=0, Q=0)[7]
Steak: SARIMA(p=1, d=0, q=0)(P=1, D=0, Q=0)[7]
Tomato: SARIMA(p=1, d=1, q=0)(P=1, D=1, Q=0)[7]

Due to the inherent seasonality, which appears to be weekly, we now use *auto.arima()* to fit models for Shrimp, Steak, and Tomato.

```
ingredients <- unique(data_tsibble$Ingredient)

models <- list()

for (ingredient in ingredients) {
  # Filter the data for the specific ingredient
  ingredient_data <- data_tsibble %>%
    filter(Ingredient == ingredient) %>%
    as_tsibble(index = Date)

  # Convert to time series object for auto.arima
  ts_data <- ts(ingredient_data$Sales_Units, frequency = 7) # Weekly seasonality

  # Fit seasonal ARIMA model
  model <- auto.arima(ts_data, seasonal = TRUE, stepwise = FALSE, approximation = FALSE)

  # Store the model in a list
  models[[ingredient]] <- model

  # Print model summary
  cat(paste("Seasonal ARIMA model for", ingredient, ":\n"))
  print(model)
  cat("\n")
}
```

```
## Seasonal ARIMA model for Shrimp :
## Series: ts_data
## ARIMA(3,0,0)(2,1,0)[7]
##
## Coefficients:
##           ar1      ar2      ar3     sar1     sar2
##       -0.0101  -0.0244  -0.0997  -0.5781  -0.2692
## s.e.   0.0408   0.0406   0.0408   0.0396   0.0397
##
## sigma^2 = 16.53:  log likelihood = -1697.36
## AIC=3406.71   AICc=3406.85   BIC=3433.11
##
## Seasonal ARIMA model for Steak :
## Series: ts_data
## ARIMA(0,0,0)(2,1,0)[7]
##
## Coefficients:
##          sar1     sar2
##       -0.6216  -0.3407
## s.e.   0.0385   0.0386
##
## sigma^2 = 20.52:  log likelihood = -1764.31
## AIC=3534.63   AICc=3534.67   BIC=3547.83
##
## Seasonal ARIMA model for Tomato :
## Series: ts_data
## ARIMA(0,0,0)(0,1,1)[7]
##
## Coefficients:
##          sma1
##       -0.8583
## s.e.   0.0250
##
## sigma^2 = 1708:  log likelihood = -3098.67
## AIC=6201.34   AICc=6201.36   BIC=6210.14
```

We also manually fit our hypothesized ARIMA configurations for these three ingredients for comparison.

```
# Fit SARIMA model for Shrimp
model_shrimp <- data_tsibble %>%
  filter(Ingredient == "Shrimp") %>%
  model(SARIMA = ARIMA(Sales_Units ~ 0 + pdq(1,0,0) + PDQ(1,0,0)))

# Fit SARIMA model for Steak
model_steak <- data_tsibble %>%
  filter(Ingredient == "Steak") %>%
  model(SARIMA = ARIMA(Sales_Units ~ 0 + pdq(1,0,0) + PDQ(1,0,0)))

# Fit SARIMA model for Tomato
model_tomato <- data_tsibble %>%
  filter(Ingredient == "Tomato") %>%
  model(SARIMA = ARIMA(Sales_Units ~ 0 + pdq(1,1,0) + PDQ(1,1,0)))

model_shrimp %>% report()
```

```
## Series: Sales_Units
## Model: ARIMA(1,0,0)(1,0,0)[7]
##
## Coefficients:
##          ar1    sar1
##       0.0682  0.8928
## s.e.  0.0438  0.0190
##
## sigma^2 estimated as 21.7:  log likelihood=-1805.76
## AIC=3617.52   AICc=3617.56   BIC=3630.76
```

```
model_steak %>% report()
```

```
## Series: Sales_Units
## Model: ARIMA(1,0,0)(1,0,0)[7]
##
## Coefficients:
##          ar1    sar1
##       0.0622  0.9064
## s.e.  0.0427  0.0174
##
## sigma^2 estimated as 28.27:  log likelihood=-1886.7
## AIC=3779.39   AICc=3779.43   BIC=3792.63
```

```
model_tomato %>% report()
```

```
## Series: Sales_Units
## Model: ARIMA(1,1,0)(1,1,0)[7]
##
## Coefficients:
##           ar1     sar1
##       -0.5026  -0.4826
## s.e.   0.0352   0.0357
##
## sigma^2 estimated as 3386:  log likelihood=-3295.19
## AIC=6596.37   AICc=6596.41   BIC=6609.57
```

When comparing the *auto.arima()* to the manually fitted ARIMA models, the auto-fitted models perform better regarding AIC, AICc, and BIC along with residual variance (sigma^2). The models selected by *auto.arima()* are:

Shrimp: SARIMA(3,0,0)(2,1,0)[7]
Steak: SARIMA(0,0,0)(2,1,0)[7]
Tomato: SARIMA(0,0,0)(0,1,1)[7]

While these coefficients are more complex and harder to interpret, we believe they are a better fit and have better explanatory power.

With our ARIMA models fit, we performed a residual analysis to validate independence assumptions. To start, we conducted a Ljung-Box test on the *auto.arima()* selected models.

```
# Refit models with optimal parameters identified with auto.arima()
model_shrimp <- data_tsibble %>%
  filter(Ingredient == "Shrimp") %>%
  model(SARIMA = ARIMA(Sales_Units ~ 0 + pdq(3,0,0) + PDQ(2,1,0)))
model_steak <- data_tsibble %>%
  filter(Ingredient == "Steak") %>%
  model(SARIMA = ARIMA(Sales_Units ~ 0 + pdq(0,0,0) + PDQ(2,1,0)))
model_tomato <- data_tsibble %>%
  filter(Ingredient == "Tomato") %>%
  model(SARIMA = ARIMA(Sales_Units ~ 0 + pdq(0,0,0) + PDQ(0,1,1)))

# Perform a ljungbox test for each model
model_shrimp %>% augment() %>%
  features(.resid, ljung_box, lag = 10)
```

```
## # A tibble: 1 × 4
##   Ingredient .model lb_stat lb_pvalue
##   <chr>      <chr>    <dbl>     <dbl>
## 1 Shrimp     SARIMA    11.7     0.307
```

```
model_steak %>% augment() %>%
  features(.resid, ljung_box, lag = 10)
```

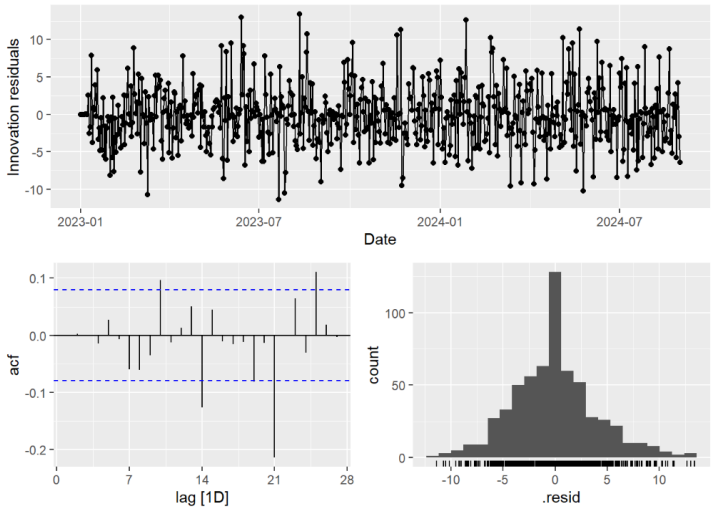
```
## # A tibble: 1 × 4
##   Ingredient .model lb_stat lb_pvalue
##   <chr>      <chr>    <dbl>     <dbl>
## 1 Steak      SARIMA    12.6     0.245
```

```
model_tomato %>% augment() %>%
  features(.resid, ljung_box, lag = 10)
```

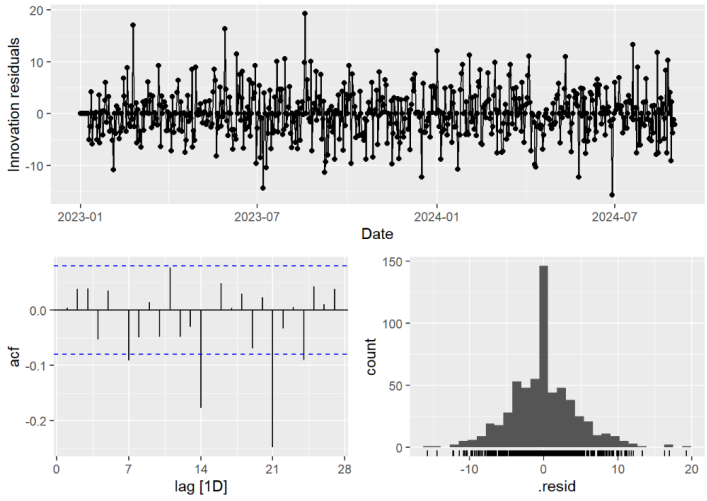
```
## # A tibble: 1 × 4
##   Ingredient .model lb_stat lb_pvalue
##   <chr>      <chr>    <dbl>     <dbl>
## 1 Tomato     SARIMA    11.0     0.359
```

The null hypothesis is that residuals are uncorrelated. Since all three models have a p-value > 0.05, we concluded that we cannot reject residual independence. We also decided to analyze the residual plots for these ARIMA models.

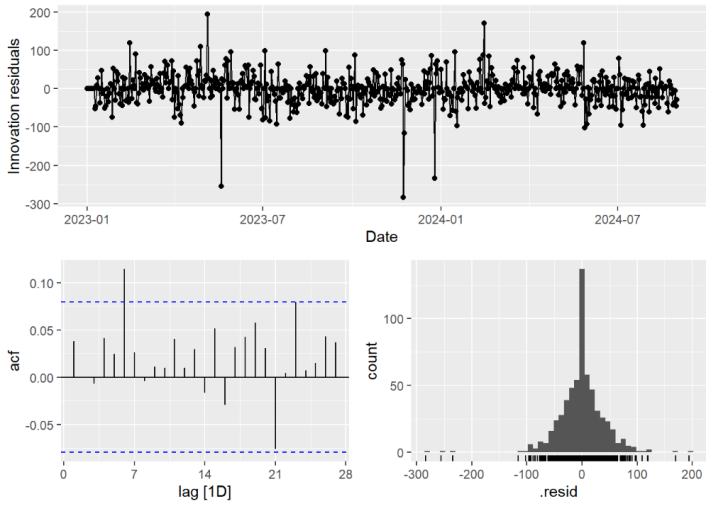
```
model_shrimp %>% gg_tsresiduals()
```




```
model_steak %>% gg_tsresiduals()
```




```
model_tomato %>% gg_tsresiduals()
```



Based on these residual outputs, we see small spikes in ACF at seasonal lags (7, 14, etc.), suggesting that the weekly seasonality might need to be fully captured in the models for shrimp and steak. All three models show extreme residuals, possibly due to a systematic effect the models are not capturing. However, these residuals are all approximately symmetric (though heavy tails exist for shrimp and steak).

To conclude our model validation steps, we examined the information criteria for each model selected by *auto.arima()*.

```
glance(model_shrimp)
```

```
## # A tibble: 1 × 9
##   Ingredient .model sigma2 log_lik   AIC  AICc   BIC ar_roots   ma_roots
##   <chr>      <chr>   <dbl>   <dbl> <dbl> <dbl> <dbl> <list>     <list>
## 1 Shrimp     SARIMA   16.5  -1697. 3407. 3407. 3433. <cpl [17]> <cpl [0]>
```

```
glance(model_steak)
```

```
## # A tibble: 1 × 9
##   Ingredient .model sigma2 log_lik   AIC  AICc   BIC ar_roots   ma_roots
##   <chr>      <chr>   <dbl>   <dbl> <dbl> <dbl> <dbl> <list>     <list>
## 1 Steak      SARIMA   20.5  -1764. 3535. 3535. 3548. <cpl [14]> <cpl [0]>
```

```
glance(model_tomato)
```

```
## # A tibble: 1 × 9
##   Ingredient .model sigma2 log_lik   AIC  AICc   BIC ar_roots  ma_roots
##   <chr>      <chr>   <dbl>   <dbl> <dbl> <dbl> <dbl> <list>    <list>
## 1 Tomato     SARIMA  1708.  -3099. 6201. 6201. 6210. <cpl [0]> <cpl [7]>
```

Based on these results, the Shrimp model has the lowest AIC and BIC among the candidate models, with tomato having significantly higher AIC/BIC due to higher variance (sigma^2).

The final procedure we performed on the ARIMA models was cross-validation. To cross-validate a time-series forecasting model, we repeatedly extended our data to fit the model. We started by assuming that the initial data consists of 70% of the data in the code below. Then, we recalculated the model sequentially, assuming we observed seven additional points (datums). Finally, we calculated the out-of-sample accuracy metrics by comparing each forecast on the horizon with the available data. The cross-validation procedure was also performed on our hypothesized (manual) ARIMA models before using *auto.arima()*.

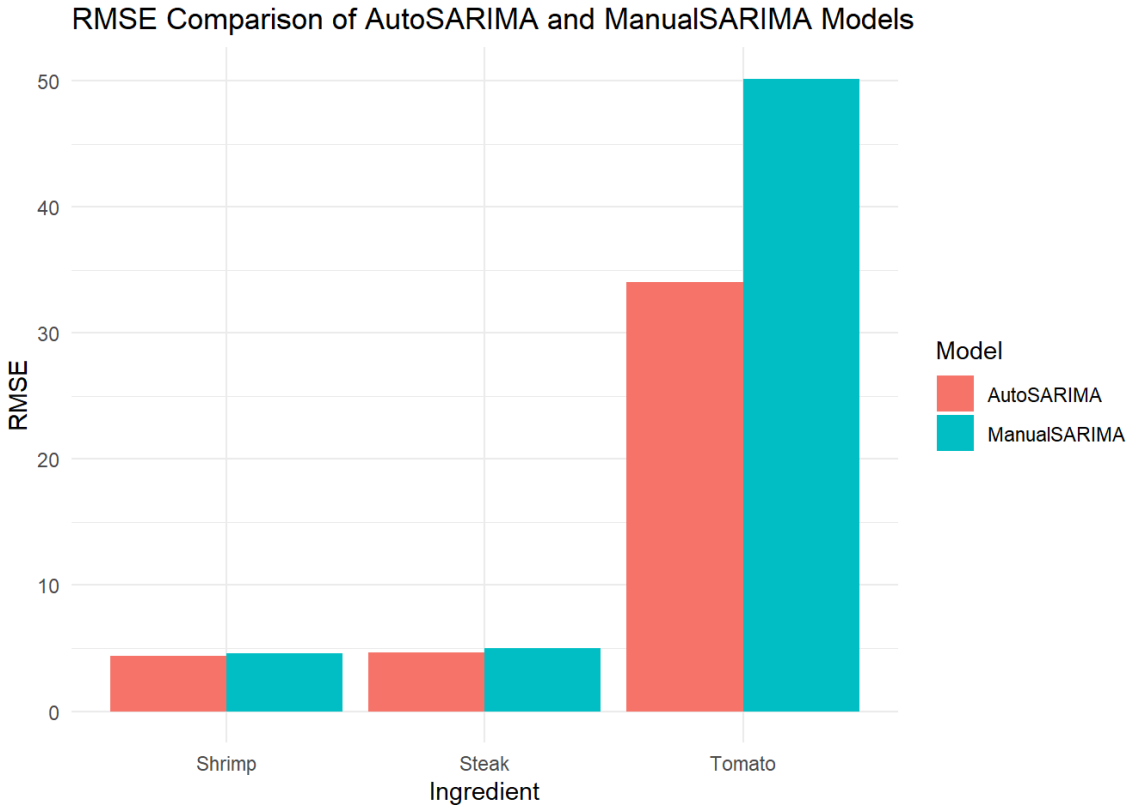
```
all_accuracy_metrics <- tibble()

# Define a function to calculate accuracy for a given ingredient
calculate_accuracy <- function(data, ingredient, auto_formula, manual_formula) {
  # Filter the data for the specific ingredient
  ingredient_data <- data %>%
    filter(Ingredient == !!ingredient)

  # Stretch tsibble for cross-validation
  cv_data <- ingredient_data %>%
    stretch_tsibble(.init = 426, .step = 7)

  # Fit the models
  cv_models <- cv_data %>%
    model(
      AutoSARIMA = ARIMA(auto_formula),
      ManualSARIMA = ARIMA(manual_formula)
    )

  # Forecast and compute accuracy
  cv_forecasts <- cv_models %>%
    forecast(h = 14)

  # Compute accuracy and add the ingredient as a column
  cv_forecasts %>%
    accuracy(ingredient_data) %>%
    mutate(Ingredient = !!ingredient) %>%
    return()
}

# Add results for Shrimp
all_accuracy_metrics <- bind_rows(
  all_accuracy_metrics,
  calculate_accuracy(
    data_tsibble, "Shrimp",
    auto_formula = Sales_Units ~ pdq(3, 0, 0) + PDQ(2, 1, 0),
    manual_formula = Sales_Units ~ pdq(1, 0, 0) + PDQ(1, 0, 0)
  )
)

# Add results for Steak
all_accuracy_metrics <- bind_rows(
  all_accuracy_metrics,
  calculate_accuracy(
    data_tsibble, "Steak",
    auto_formula = Sales_Units ~ pdq(0, 0, 0) + PDQ(2, 1, 0),
    manual_formula = Sales_Units ~ pdq(1, 0, 0) + PDQ(1, 0, 0)
  )
)

# Add results for Tomato
all_accuracy_metrics <- bind_rows(
  all_accuracy_metrics,
  calculate_accuracy(
    data_tsibble, "Tomato",
    auto_formula = Sales_Units ~ pdq(0, 0, 0) + PDQ(0, 1, 1),
    manual_formula = Sales_Units ~ pdq(1, 1, 0) + PDQ(1, 1, 0)
  )
)

print(all_accuracy_metrics)
```

```
## # A tibble: 6 × 11
##   .model Ingredient .type      ME  RMSE   MAE     MPE  MAPE  MASE RMSSE     ACF1
##   <chr>  <chr>      <chr>   <dbl> <dbl> <dbl>   <dbl> <dbl> <dbl> <dbl>    <dbl>
## 1 AutoS… Shrimp     Test  -0.278   4.40  3.28  NaN    Inf   0.915 0.922 -0.0474
## 2 Manua… Shrimp     Test  -0.114   4.60  3.77 -Inf    Inf   1.05  0.964 -0.0208
## 3 AutoS… Steak      Test  -0.0814  4.66  3.35  -24.6   48.9 0.864 0.859 -0.0963
## 4 Manua… Steak      Test  -0.0565  5.00  3.91 -Inf    Inf   1.01  0.921  0.00787
## 5 AutoS… Tomato     Test  -0.312  34.0  24.6    -1.40  10.2 0.687 0.633  0.0386
## 6 Manua… Tomato     Test   0.705  50.2  39.1   NaN    Inf   1.09  0.934  0.409
```

As we expected, the models *auto.arima()* selected have lower RMSE and MAE across all ingredients, confirming better predictive accuracy. MPE/MAPE is Inf/NAN for several models

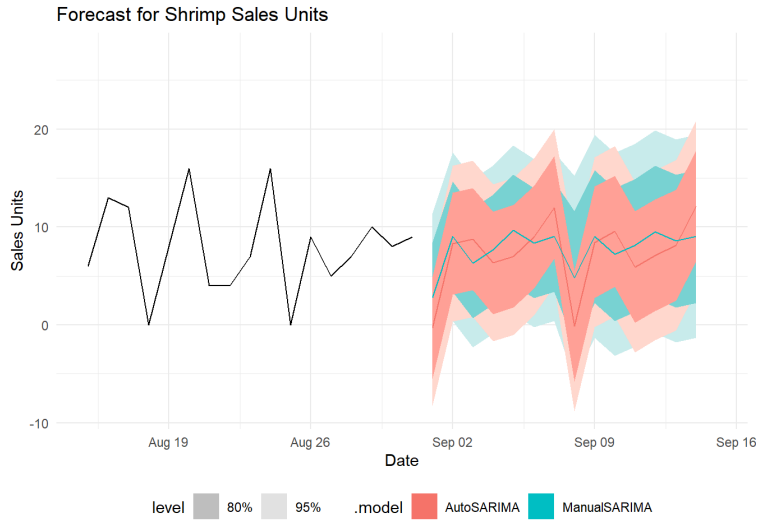
due to the 0 values on Sundays. ACF1 values (autocorrelation of residuals at lag 1) are close to zero for all models in Shrimp and Steak, indicating minimal autocorrelation in residuals. The tomato shows higher ACF1 for the Manual ARIMA model we hypothesized (0.409), suggesting significant residual autocorrelation and potential model misspecification.

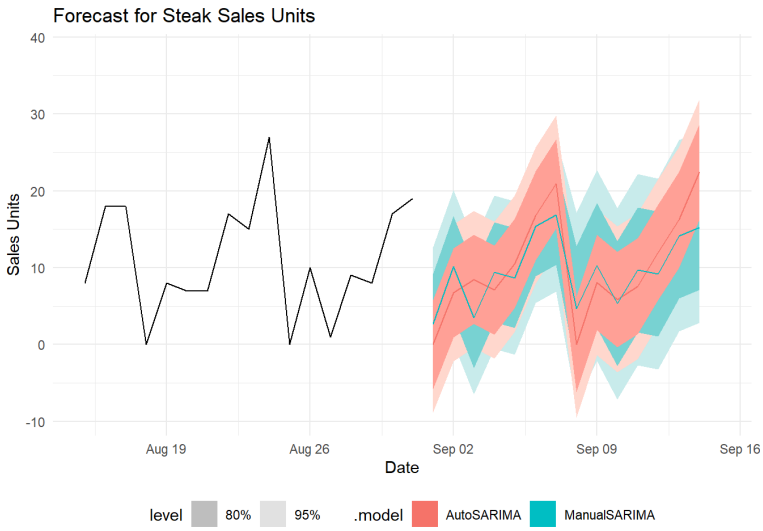
Below, we compared the RMSE for each model and ingredient.

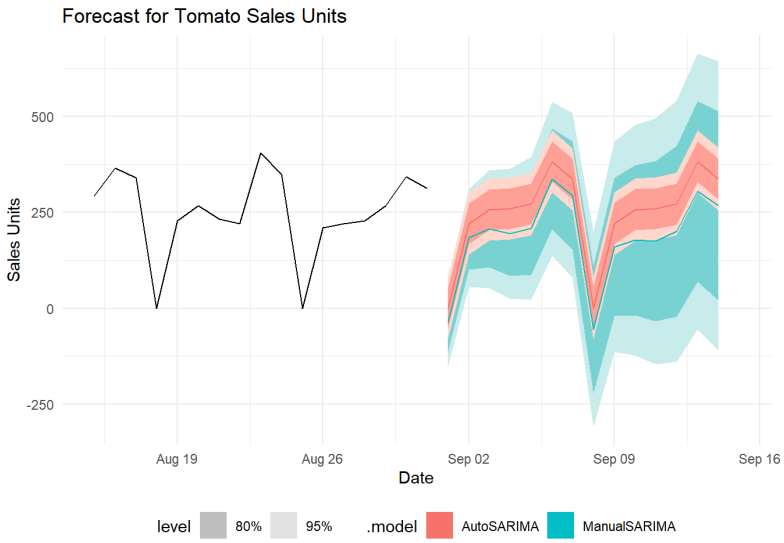


From the above graph, we see that the *auto.arima()* models perform slightly better than the manual, hypothesized ARIMA models for steak and shrimp. However, this improvement is much more dramatic for the Tomato *auto.arima()* model than the Tomato Manual ARIMA model. Tomato sales were generally more volatile, so this was expected.

Lastly, we looked at the forecast plots of the Auto ARIMA and Manual ARIMA plots below.


Forecast for Shrimp Sales Units
Sales Units
20
10
0
-10
Aug 19
Aug 26
Sep 02
Sep 09
Sep 16
Date
level
80%
95%
.model
AutoSARIMA
ManualSARIMA


Forecast for Steak Sales Units
Sales Units
40
30
20
10
0
-10
Aug 19
Aug 26
Sep 02
Sep 09
Sep 16
Date
level
80%
95%
.model
AutoSARIMA
ManualSARIMA


Forecast for Tomato Sales Units
Sales Units
500
250
0
-250
Aug 19
Aug 26
Sep 02
Sep 09
Sep 16
Date
level
80%
95%
.model
AutoSARIMA
ManualSARIMA

The *auto.arima() model* shows tighter confidence intervals for shrimp, indicating higher certainty than the Manual ARIMA model. The same can be said for the steak model, as the *auto.arima()* model has smoother predictions. Lastly, despite the uncertainty in tomato sales, the Auto ARIMA model still shows tighter confidence intervals, indicating a better model fit.

Overall, tomato sales are the most challenging to model, likely due to higher variability or unique seasonal patterns. However, the ARIMA models perform very well in modeling and forecasting the sales of these ingredients. We could further evaluate the models with comparable data for the forecast period and more historical data. We also considered aggregating the data for each week for each ingredient and fitting ARIMA models this way. However, ARIMA determined the results to be white noise and, thus, unsuitable for forecasting usage by ARIMA.

IV: COMPARISON WITH NAÌVE (SEASONAL) FORECASTING

Bayesian Modeling

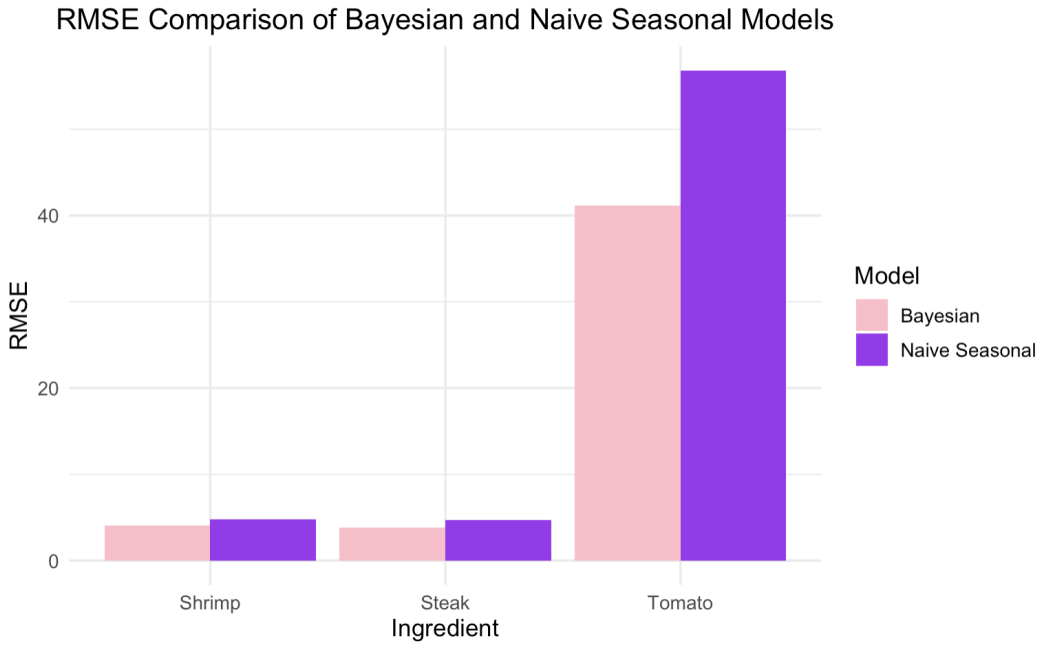
When comparing the Bayesian model to the naive seasonal forecast, we observed that the Bayesian model consistently outperformed the naive approach in terms of RMSE across all three ingredients, demonstrating a better ability to reduce error. The most significant improvement was seen in tomato sales, where the Bayesian model reduced the RMSE from 56.80 in the naive forecast to 41.17, indicating its capacity to handle variability in this data better.

However, it was challenging to outperform the naive method for shrimp, as the RMSE for the naive and Bayesian models differed slightly. Additionally, the standard deviation (whose formula is very similar to RMSE) is very close to the RMSE values for shrimp, suggesting that forecasting approaches struggled with the inherent irregularities.

The reduction from a standard deviation of 77.77 for tomato sales to an RMSE of 41.17 demonstrates significant improvement despite the inherent irregularities in the data. Even a naive seasonal approach is a vast improvement over the standard deviation.
Below is a table comparing the RMSEs with the standard deviation, which we can interpret as the “RMSE” we would get if we just forecasted a simple mean for future periods.

Ingredient	**Naive Seasonal Forecasting RMSE**	**Bayesian Modeling RMSE**	**Mean of Data**	**Standard Deviation of Data**
Shrimp	4.808633	4.039198	10.63	5.00
Steak	4.707859	3.785547	12.43	6.68

Tomato	56.79919	41.1707	300.13	77.77



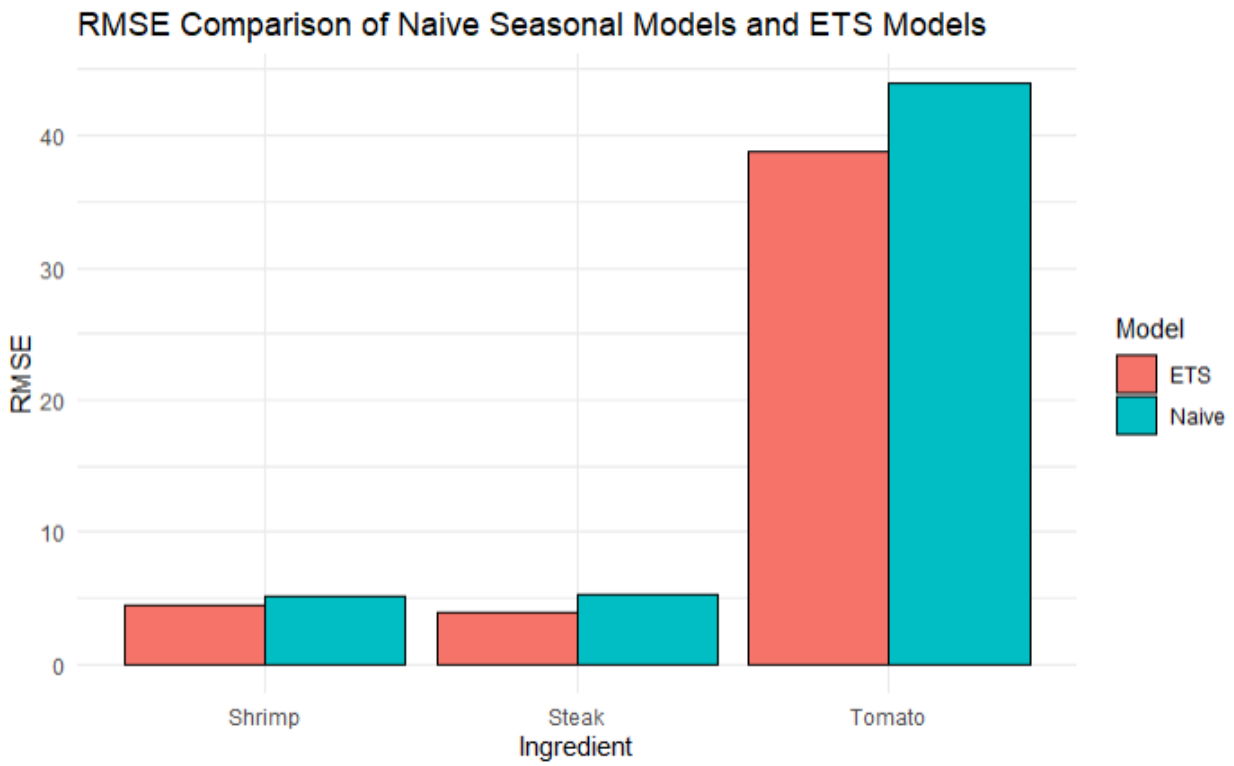
Exponential Smoothing

The ETS model (MNM) reduced the RMSE for shrimp from 5.102 to 4.511, indicating better accuracy in capturing sales patterns and trends. Similarly, for steak, the ETS model (MAA) decreased the RMSE from 5.327 to 3.970, demonstrating a notable improvement in forecasting precision, particularly for mid-demand sales days.

The ETS model (ANA) significantly reduced the RMSE for tomato from 43.931 to 38.739. Although the improvement is smaller than that for shrimp and steak, it still represents an enhanced ability to model tomato sales variability. This suggests that ETS models are more effective at accounting for the underlying seasonal and trend components in sales data, resulting in forecasts that better align with observed patterns across all ingredients. The reductions in RMSE validate the choice of ETS models as a superior forecasting approach compared to the simpler Naive Seasonal method.

Ingredient	**Naive Seasonal Forecasting RMSE**	**ETS Model**	**ETS Modeling RMSE**
Shrimp	5.102296	MNM	4.510853

Steak	5.327178	MAA	3.969921
Tomato	43.930624	ANA	38.73852

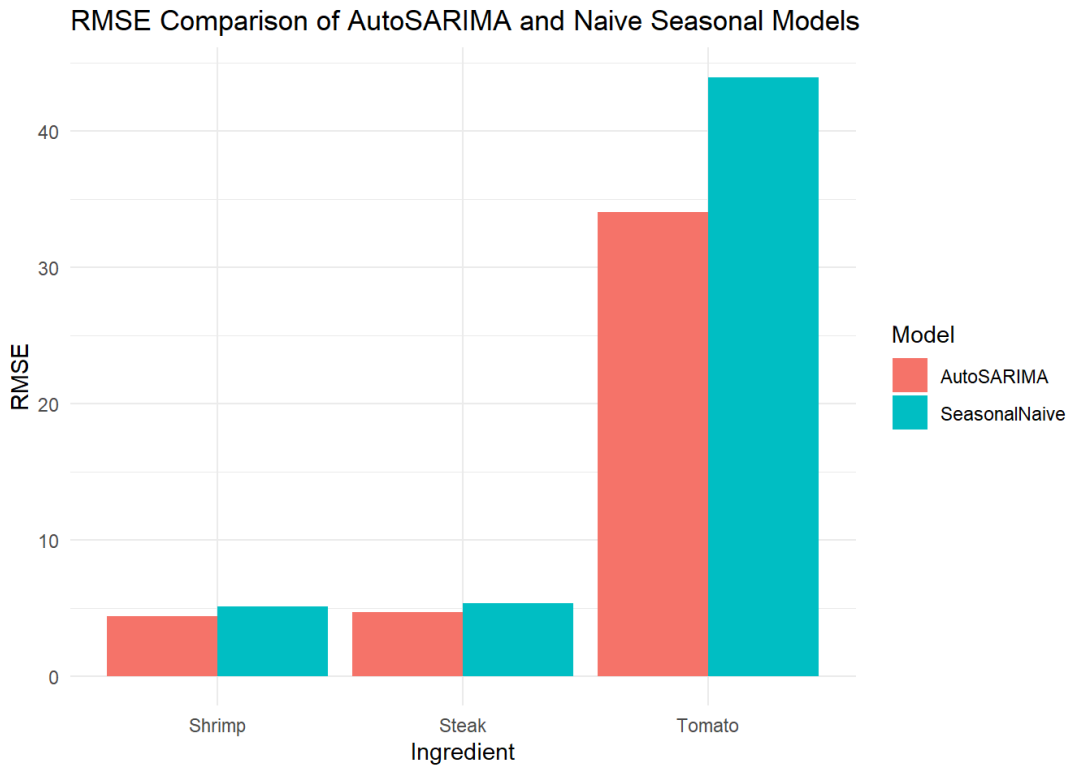


ARIMA

The ARIMA models we selected included seasonal components. We evaluated them against Naïve seasonal forecasting models (automatic fit) for a fair comparison. We also created a plot to visually compare the RMSE between the *auto.arima()* and Naive seasonal models.

```
compare_models_with_accuracy <- function(data, ingredient, auto_formula) {
  # Filter data for the specific ingredient
  ingredient_data <- data %>%
    filter(Ingredient == !!ingredient)

  # Use stretch_tsibble for time-series cross-validation
  cv_data <- ingredient_data %>%
    stretch_tsibble(.init = 426, .step = 7)

  # Fit Auto SARIMA model and Seasonal Naive model
  cv_models <- cv_data %>%
    model(
      AutoSARIMA = ARIMA(auto_formula),
      SeasonalNaive = SNAIVE(Sales_Units ~ lag("week"))
    )

  # Generate forecasts for both models
  cv_forecasts <- cv_models %>%
    forecast(h = 14)

  # Align actual data and compute accuracy
  accuracy_metrics <- cv_forecasts %>%
    accuracy(ingredient_data) %>%
    mutate(Ingredient = !!ingredient)

  return(list(forecasts = cv_forecasts, accuracy = accuracy_metrics))
}

# Initialize lists to store results for all ingredients
all_forecasts <- list()
all_accuracy_metrics <- tibble()

# Compare models for Shrimp
shrimp_results <- compare_models_with_accuracy(
  data_tsibble, "Shrimp",
  auto_formula = Sales_Units ~ pdq(3, 0, 0) + PDQ(2, 1, 0)
)
all_forecasts$Shrimp <- shrimp_results$forecasts
all_accuracy_metrics <- bind_rows(all_accuracy_metrics, shrimp_results$accuracy)

# Compare models for Steak
steak_results <- compare_models_with_accuracy(
  data_tsibble, "Steak",
  auto_formula = Sales_Units ~ pdq(0, 0, 0) + PDQ(2, 1, 0)
)
all_forecasts$Steak <- steak_results$forecasts
all_accuracy_metrics <- bind_rows(all_accuracy_metrics, steak_results$accuracy)

# Compare models for Tomato
tomato_results <- compare_models_with_accuracy(
  data_tsibble, "Tomato",
  auto_formula = Sales_Units ~ pdq(0, 0, 0) + PDQ(0, 1, 1)
)
all_forecasts$Tomato <- tomato_results$forecasts
all_accuracy_metrics <- bind_rows(all_accuracy_metrics, tomato_results$accuracy)

# Print accuracy metrics
print(all_accuracy_metrics)
```

```
## # A tibble: 6 × 11
##   .model   Ingredient .type      ME  RMSE   MAE    MPE  MAPE  MASE RMSSE    ACF1
##   <chr>    <chr>      <chr>   <dbl> <dbl> <dbl>  <dbl> <dbl> <dbl> <dbl>   <dbl>
## 1 AutoSAR… Shrimp     Test  -0.278   4.40  3.28 NaN    Inf   0.915 0.922 -0.0474
## 2 Seasona… Shrimp     Test  -0.173   5.10  3.83 -20.0   52.5 1.07  1.07  -0.0343
## 3 AutoSAR… Steak      Test  -0.0814  4.66  3.35 -24.6   48.9 0.864 0.859 -0.0963
## 4 Seasona… Steak      Test  -0.0501  5.33  3.77 -24.6   54.8 0.971 0.982  0.0305
## 5 AutoSAR… Tomato     Test  -0.312  34.0  24.6   -1.40  10.2 0.687 0.633  0.0386
## 6 Seasona… Tomato     Test  -0.780  43.9  31.9   -1.80  13.3 0.891 0.818  0.0172
```



The *auto.arima()* models consistently outperform the Seasonal Naive approach across all ingredients regarding RMSE and MAE. The *auto.arima()* model achieves an RMSE of 4.4 for shrimp compared to 5.1 for Seasonal Naive, indicating better accuracy in capturing patterns. Similarly, the *auto.arima()* exhibits an RMSE of 4.66 for steak versus 5.33 for Seasonal Naive. The most significant difference appears in tomato, where the *auto.arima()* model reduces the RMSE from 43.93 (Seasonal Naive) to 34.01, demonstrating its superior ability to handle the high variance and complex seasonality of tomato sales. Despite Seasonal Naive's simplicity, its higher RMSE and MAE suggest that it struggles to account for intricate seasonal and autoregressive patterns, especially for volatile series like the tomato data. Overall, the *auto.arima()* models offer a more robust and reliable forecasting method for all three ingredients, capturing trends and seasonality more effectively.

V: CONCLUSION AND RECOMMENDATIONS

In conclusion, ARIMA is the most reliable and accurate forecasting method for La Casita's ingredient demand. Its ability to integrate seasonality, trend, and residual independence outperformed Bayesian and ETS models across all evaluation metrics, particularly in handling the high variability of tomato sales. We recommend implementing *auto.arima()* for shrimp, steak, and tomato forecasting, complemented by regular cross-validation to refine the model as new data becomes available. This approach will empower La Casita to optimize inventory, reduce waste, and respond effectively to fluctuating customer demand.