Time-series Project

Zheng Luo (Davies Luo)

2023-12-15

Introduction and Data Exploratory

This report analyses a decade of monthly sales data (January 2011-December 2020), focusing on patterns, trends, and seasonal fluctuations in sales. To achieve a robust forecast and glean actionable insights, I employed a triad of sophisticated statistical methods: Seasonal ARIMA model, Time-Series Regression, and Dynamic Linear Model (DLM). Each method is chosen for its unique strengths and ability to model different aspects of time series data. I scrutinized the stationarity of the series, explored seasonal decomposition, and fitted the models, carefully calibrating them to capture the intrinsic properties of the data. The diagnostics of the models were rigorously evaluated to ensure the adequacy of the fit, and their forecasts were compared to deliver the most reliable prediction for the future sales from January to June 2021, with the testig prediction from January to December 2020 as well. This report documents my methodical approach, from preliminary data analysis to the selection of appropriate models, diagnostics, comparison, and final forecasting, providing a comprehensive narrative of my analytical journey.

To initiate my comprehensive analysis, I began by loading the dataset from a text file, specifically selecting the fourth column which represents the dataset of interest. This dataset encapsulates the monthly sales figures over a ten-year period, providing a lengthy temporal snapshot essential for robust time series analysis.

With the data successfully loaded, the next step is to visualize the time series. Plotting the time series serves as a preliminary diagnostic tool, allowing me to observe patterns, trends, outliers, or any irregularities present in the data. The visualization lays the groundwork for all subsequent analyses by providing an initial impression of the data's characteristics.

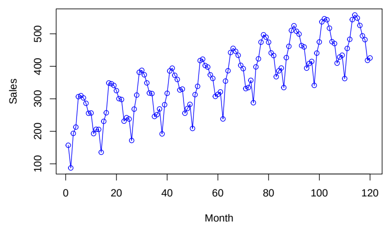


Figure 1: Monthly Sales Time Series

Modelling and Diagnostic Checks

Addressing Seasonality in Time Series Forecasting

In the realm of time series analysis, ensuring data stationarity is pivotal for the accuracy of forecasting models. Stationarity implies that the statistical properties of the series—such as mean, variance, and autocorrelation—are constant over time. As the monthly sales data exhibi, where sales ebb and flow in a predictable annual cycleted clear seasonal patterns from the previus time series plot.

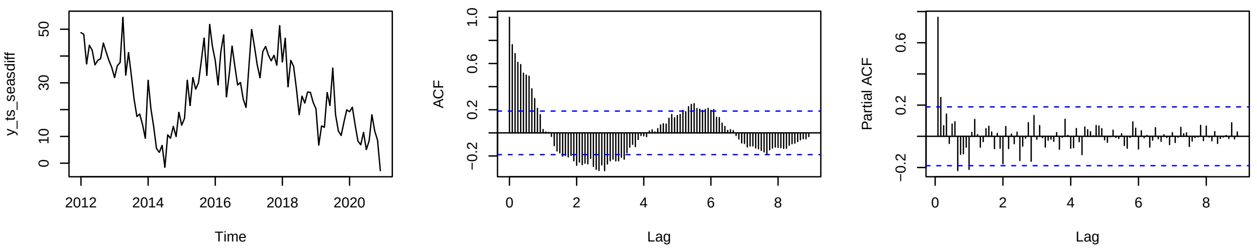


Figure 2: Seasonally Differenced Time Series(left) with ACF(middle) and PACF(right)

After the initial seasonal differencing, the resulting time series plot suggested that further differencing might be necessary. The series did not yet resemble the "white noise" characteristic indicative of stationarity, as patterns were still discernible in the data. Consequently, I proceeded to apply a first-order difference to the already seasonally differenced data to remove any lingering trends or non-seasonal patterns.

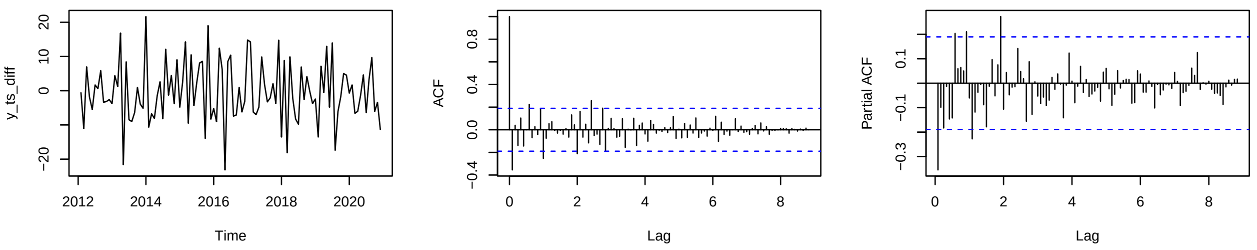


Figure 3: Seasonally and Non-seasonally Differenced Time Series(left) with ACF(middle) and PACF(right)

The plot of the first and seasonally differenced time series provided a more promising outlook, with the appearance of the data now more closely aligning with that of stationary white noise. Further analysis with Autocorrelation (ACF) and Partial Autocorrelation (PACF) functions confirmed these observations.

The ACF plot for the non-seasonally and seasonally differenced data indicated that the autocorrelations at most lags fall within the confidence bounds, suggesting no significant autocorrelations remain. Similarly, the PACF plot did not display significant spikes, reinforcing the impression of an adequately differenced series.

Ensuring Stationarity: The Augmented Dickey-Fuller Test

The Augmented Dickey-Fuller test(Appendix a.1) result affirmed the series' stationarity. The test yielded a Dickey-Fuller statistic of -6.134 with a lag order of 4. The reported p-value of 0.01 strongly rejects the null hypothesis of a unit root being present.

Decomposing the Time Series: STL Method

Beyond testing for stationarity, understanding the behavior of the time series components is also essential for accurate modeling and forecasting. The STL decomposition offers a flexible approach to separate a time series into trend, seasonal, and irregular components.

STL decomposition plot(Appendix a.2) revealed clear seasonal and trend components in the data: an annual pattern and a general upward trajectory over ten years. The remainder component, representing the noise after accounting for seasonality and trend. The decomposition supports the choice of a Seasonal ARIMA model by confirming the presence of both trend and seasonality, which were subsequently accounted for in the model specification.

Seasonal ARIMA model

Given the confirmed seasonality from prior analysis and the plots showing no significant spikes in the ACF and PACF for the seasonally and non-seasonally differenced series, an ARIMA model with no additional autoregressive or moving average terms for the non-seasonal part may be suitable by my estimation. Tthe lack of significant spikes in both the ACF and PACF plots suggests that an ARIMA model with no further autoregressive or moving average components is needed for the differenced series. This leads to a model of ARIMA(0,1,1)(0,1,1)[12], capturing the prior differencing steps and the absence of significant autocorrelation after differencing. In order to confirm, I will conduct a grid search using R's loop function to identify the optimal ARIMA model by iterating over parameters, the model selection was empirically based on AIC and BIC. A lower AIC and BIC indicate a model that efficiently captures the data trends without unnecessary complexity.

p	d	q	P	D	Q	AIC	BIC
0	1	1	1	1	1	739.4804	750.1718
0	1	1	0	1	1	739.5887	747.6072
1	1	1	1	1	1	741.3306	754.6947
0	1	2	1	1	1	741.3707	754.7348
1	1	1	0	1	1	741.5287	752.2200

My grid search results highlighted the best time series model fitting the data's underlying patternn. The ARIMA(0,1,1)(1,1,1)[12] model emerges as the frontrunner, boasting the lowest AIC and BIC values, suggesting it as the most parsimonious fit among the contenders. This model adeptly captures the essence of the time series, incorporating both non-seasonal and seasonal components without overcomplicating the structure. These results highlight the balance between model complexity and data accuracy, emphasizing systematic model selection.

The Box-Ljung test (Appendix a.3) gives a p-value of 0.9293, which is well above the conventional alpha level of 0.05. This indicates that there is no significant autocorrelation in the residuals at lag 1. It suggests that the model residuals are independent across time lags, which is a good sign that the model is adequately capturing the time series data's structure.

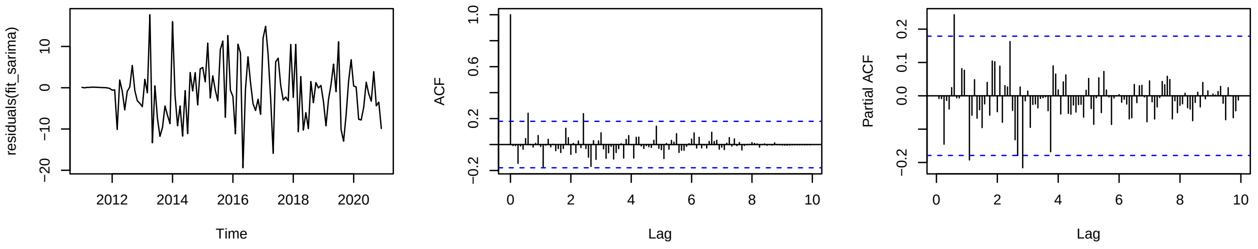


Figure 4: Residuals of ARIMA Model(left) with ACF(middle) and PACF(right)

The ACF plot of residuals shows that all autocorrelations are within the confidence bounds, which implies that there is no significant autocorrelation left in the residuals. This is a strong indicator that the model has accounted for the autocorrelations in the data properly. And the PACF plot similarly indicates that there are no significant spikes outside the confidence bounds, suggesting that the model has captured the partial autocorrelations effectively. There's no evidence of any overlooked autoregressive behavior in the data.

Diagnostic checks confirmed the ARIMA model's adequacy in capturing time series patterns with minimal residual structure. The normality of the residuals is satisfactory, and the lack of autocorrelation suggests that the model is not omitting any significant information in the data. These diagnostics suggest a well-specified model for reliable forecasts.

Time-Series Regression

In the pursuit of creating a more robust forecasting model for the time series data, I adapt a two-pronged analytical approach. First, I use linear regression to capture deterministic trends and time-related cyclical fluctuations. Subsequently, to refine the model's predictive power, I address the stochastic components—the random fluctuations and noise that the regression may not account for - by fitting an ARMA model to the regression residuals. This dual strategy clearly models deterministic elements and captures stochastic processes using ARMA.

I initiate by fitting a linear regression model to the time series(the summary and fitted value plot can be found in Appendix a.4), which serves to explain any trend or systematic changes in the data with respect to time. This isolates the series' predictable aspects linked to time progression.

After regression, I extract residuals unexplained by the time trend. These residuals are expected to contain the intrinsic, stochastic properties of the time series—essentially the noise that isn't accounted for by the linear component. I then engage in fitting an array of ARMA models to these residuals. Each model endeavors to characterize the autocorrelation within the residuals, thereby capturing any lingering patterns in the noise.

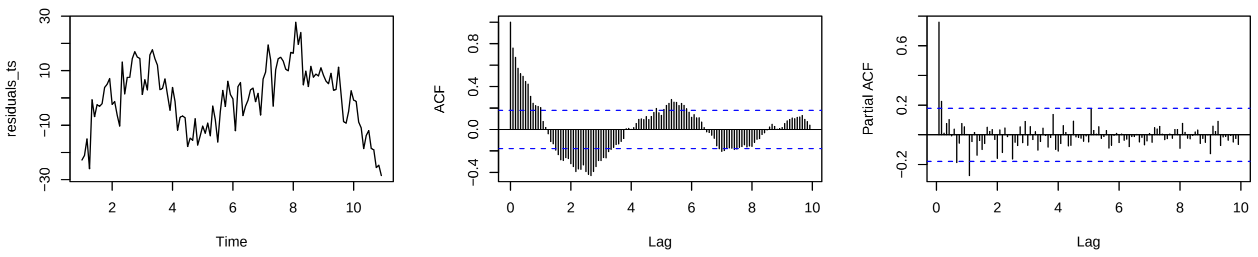


Figure 5: Residuals of Linear Model(left) with ACF(middle) and PACF(right)

The ACF and PACF plots of the linear model residuals suggest a potential ARMA(1,1) model, as indicated by a sharp cut-off after the first lag in the PACF and a slow decay in the ACF, despite the non-stationary appearance of the residuals, due to the complexity of further differencing.

To further identify the best-fitting ARMA model, I utilized the Akaike Information Criterion (AIC) and Bayesian Information Criterion (BIC), both of which reward model fit while penalizing excessive complexity:

p	q	AIC	BIC
1	1	806.7802	817.9301
2	0	808.1135	819.2634
2	1	808.7440	822.6815
1	2	808.7544	822.6919
2	2	810.6454	827.3703

The grid search for the optimal ARMA model to fit the residuals of the linear regression has yielded a clear preference for an ARMA(1,1) model, denoted by the lowest AIC and BIC values, which meets my estimation. Then I will start the process of diagnostic checking, which is integral to the validation of ARMA models fitted to the residuals of a linear regression. This step is pivotal as it ensures that the ARMA model effectively captures any autocorrelation structures in the regression residuals, which may contain patterns not explained by the initial regression.

The Ljung-Box test(Appendix a.5) is applied again to check for the independence of residuals. A p-value of 0.9826 suggests that there is no significant autocorrelation at the first lag, implying that the residuals from the ARMA model are independent over time and the model has adequately captured the autocorrelation structure of the time series.

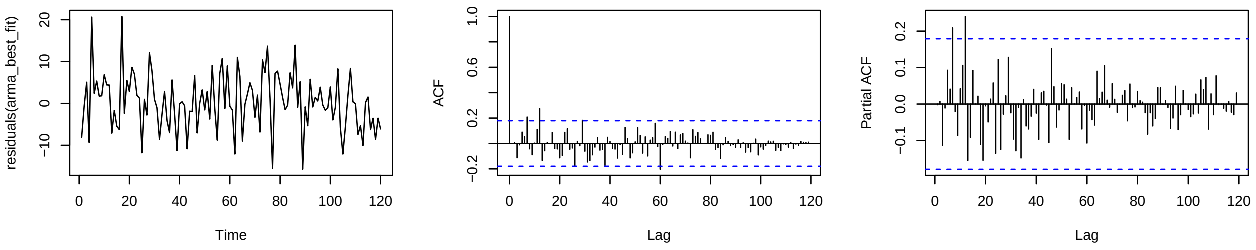


Figure 6: Residuals of ARMA Model(left) with ACF(middle) and PACF(right)

The ACF plot of the residuals shows that autocorrelations are mostly within the confidence bounds, suggesting an adequate model fit with minimal residual autocorrelation. The PACF plot also exhibits a similar pattern, with most of the partial autocorrelations lying within the confidence bounds. These observations suggest that while the ARMA model is generally well-fitted. In summary, the diagnostics indicate that the best ARMA model for the residuals of the linear regression is ARMA(1,1). This model has shown promise in capturing the underlying autocorrelation structures present in the residuals.

Dynamic Linear Model

Then, I extended my analysis to include Dynamic Linear Models (DLMs). DLMs offer a flexible approach to time series modeling by accommodating changes in both level and trend over time. This adaptability is particularly beneficial when dealing with non-stationary data that exhibits evolving dynamics. By fitting a DLM, I aimed to leverage its state-space representation to provide a comprehensive understanding of the underlying process generating the sales data, potentially improving forecast accuracy and insight into the time series structure.

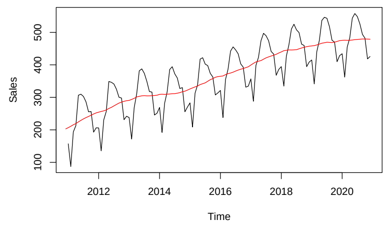
Before delving into the detailed analysis of the DLM residuals, it is imperative to ensure the robustness of the model fitting process. A crucial step in this endeavor is to evaluate the convergence of the parameter optimization routine. Convergence is a marker of a successful estimation process, indicating that the model's parameters are estimated reliably:

```
## [1] 0
```

The result of 0 signifies successful convergence, indicating that the optimization algorithm has found a stable set of parameters and that the model is well-calibrated to the data.

Smoothing within a Dynamic Linear Model is an analytical preparation for forecasting. It serves to clarify the underlying trend by minimizing the impact of random fluctuations and seasonal variations. This step is not about business strategy but about enhancing the model's predictive accuracy. The smoother the foundation, the clearer I can project into the future. By smoothing the data, I aim to create a stable platform from which to extend the forecast horizon and ascertain the DLM's predictive prowess against other models.

The smoothed trend line in the plot shows a continuous curve, indicative of long-term sales behavior. This trend, highlighted in red, contrasts sharply with the raw data's peaks and troughs, offering a simplified and more interpretable view of growth or decline:



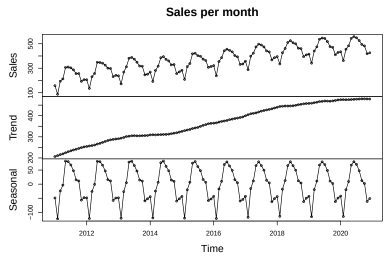


Figure 7: Monthly sales data with smoothed values and Decomposation of the sales data using DLM

Decomposing the time series into trend and seasonal components using the DLM's smoothed states vectors allows for a granular analysis of the underlying patterns. The trend component reflects the general sales trajectory, abstracted from the cyclical influences, while the seasonal component captures the repetitive, predictable changes within each period. With decomposition plot reveals the dual forces in the sales time series, the steady ascent or descent of the trend line reveals the core direction of sales over time, while the seasonal plot demonstrates the ebb and flow within each year.

Analysing with Seasonal ARIMA, Time-Series Regression, and DLMs provides a comprehensive understanding of the sales data. Each model, with its unique strengths and perspectives, has contributed valuable insights into the data's behavior. The Seasonal ARIMA model highlighted the significance of the seasonal pattern, while the Time-Series Regression illuminated the specific monthly effects on sales. The DLM, with its state-space approach, provided a dynamic view of the series' evolution over time. The convergence and residual analyses across these models have reinforced confidence in their collective ability to capture the essential characteristics of the time series.

Forecasting and Summary

Prior to delving into forecasting, I initiate with a critical testing phase, using the 10th year of data as an independent test set. This approach is instrumental in assessing the performance of the models - Seasonal ARIMA, Time-Series Regression, and Dynamic Linear Model. The testing phase evaluates the models' predictive accuracy and robustness beyond historical data, ensuring relevance and reliability for future scenarios.

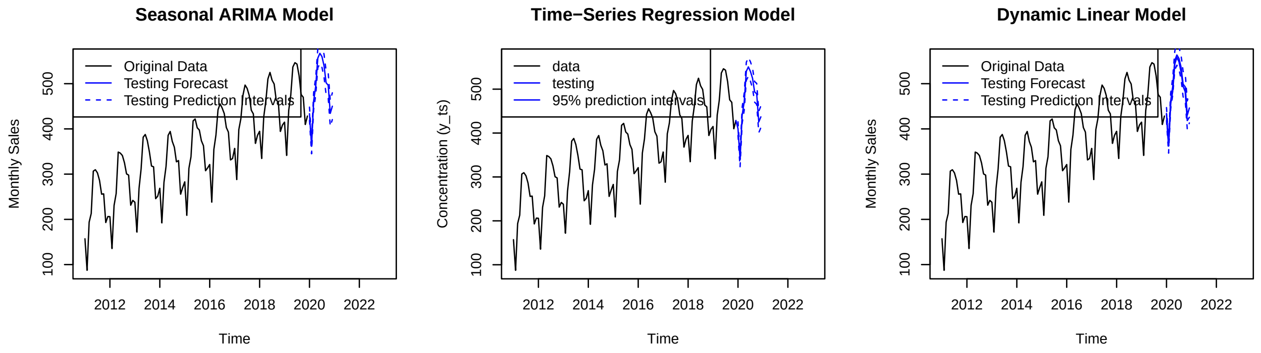


Figure 8: First 9 years' data and the testing forecast (10th year) with prediction intervals of Three Models

Having completed the testing phase, I now advance to the forecasting step. My focus shifts to predicting future trends based on the insights gained from the models' performance during testing. At this stage, I expect to observe how each model, with its unique analytical approach, projects the future months:

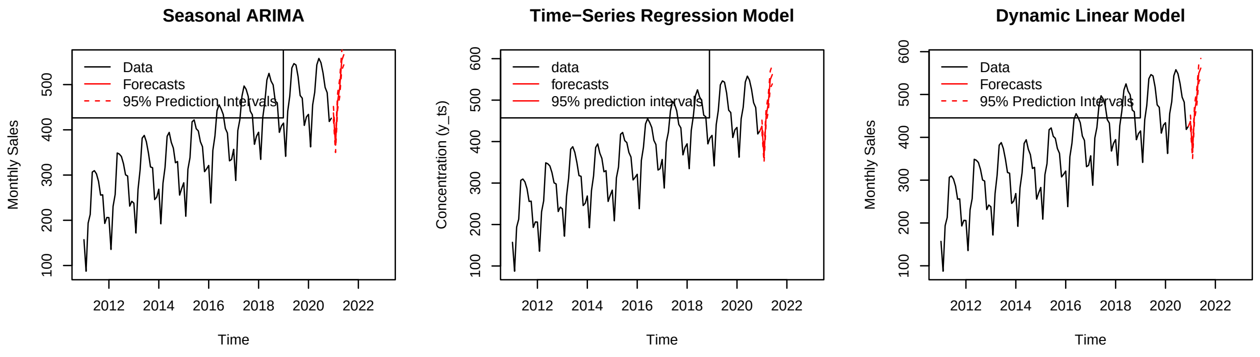


Figure 9: The Forecasts and Prediction Intervals of Three Models

In the forecasting phase, all models consistently reflected the data pattern. This uniformity across the Seasonal ARIMA, Time-Series Regression, and Dynamic Linear Model, suggests a reliable representation of the time series trends. However, to determine the optimal model in terms of efficiency and accuracy, I will next consider a more granular analysis using metrics.

In my final step, I use AIC and BIC to determine the best model. These metrics help me find a balance between model complexity and fit. A lower AIC and BIC indicate a model that efficiently captures the data trends without unnecessary complexity.

Table 3: Model Comparison for AIC and BIC

Model	AIC	BIC
Seasonal ARIMA	739.4804	750.1718
Time-Series Regression	830.7802	875.3801
Dynamic Linear Model	783.6795	797.6169

Analysing the AIC and BIC values, it's evident that the Seasonal ARIMA model emerges as the superior choice with the lowest AIC (739.4804) and BIC (750.1718) scores, indicating a more efficient balance between fit and complexity, particularly for short-term forecasting. The Time-Series Regression model, while insightful and potentially better suited for longer-term forecasts, appears to be comparatively more complex as reflected in its higher AIC and BIC values. The Dynamic Linear Model, adept at adapting to changing trends and thus potentially more effective for long-term forecasting, still falls short of the Seasonal ARIMA's performance in terms of efficiency.

Summarising the process, my journey through this analytical endeavor spanned from initial data exploration, testing for stationarity, model fitting and diagnostics, to forecasting and rigorous model comparison. The consistent alignment in forecasts across models during the testing phase provided a solid base, while the final AIC and BIC comparison conclusively pointed towards the Seasonal ARIMA model as the most fitting choice for my short-term time series forecasting needs.

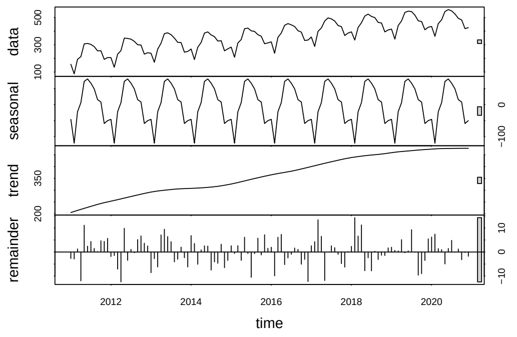
Appendix

a. Supplementary tabular and graphical output

1. Augmented Dickey-Fuller test on original time-series

```
##
##  Augmented Dickey-Fuller Test
##
## data:  y_ts_diff
## Dickey-Fuller = -6.134, Lag order = 4, p-value = 0.01
## alternative hypothesis: stationary
```

2. STL decomposition plot

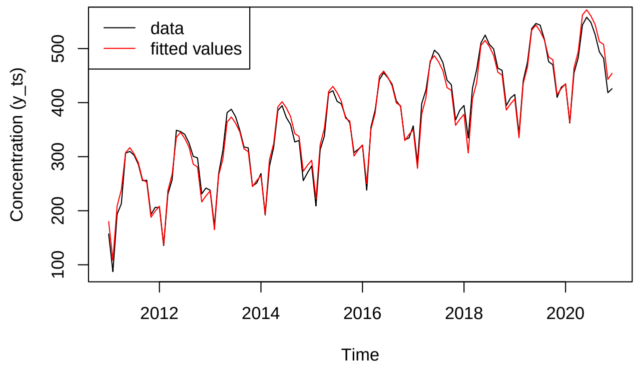


3. The Box-Ljung test on the residuals of Seaonal ARIMA Model

```
##
##  Box-Ljung test
##
## data:  residuals(fit_sarima)
## X-squared = 0.0078633, df = 1, p-value = 0.9293
```

4. The summary and fitted value plot of the linear regression model

```
##
## Call:
## lm(formula = y_ts ~ ., data = df)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -28.482  -7.718   1.314   8.119  27.770
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 177.57050    4.27336  41.553  < 2e-16 ***
## t             2.36168    0.03251  72.650  < 2e-16 ***
## monthFeb    -73.84568    5.48917 -13.453  < 2e-16 ***
## monthMar     24.07164    5.48946   4.385 2.72e-05 ***
## monthApr     52.02095    5.48994   9.476 7.85e-16 ***
## monthMay    117.89127    5.49061  21.471  < 2e-16 ***
## monthJun    124.89759    5.49148  22.744  < 2e-16 ***
## monthJul    111.12591    5.49254  20.232  < 2e-16 ***
## monthAug     92.80523    5.49379  16.893  < 2e-16 ***
## monthSep     58.92455    5.49523  10.723  < 2e-16 ***
## monthOct     51.37386    5.49687   9.346 1.54e-15 ***
## monthNov    -15.58382    5.49869  -2.834  0.00549 **
## monthDec     -6.77050    5.50071  -1.231  0.22108
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 12.27 on 107 degrees of freedom
## Multiple R-squared:  0.9871, Adjusted R-squared:  0.9856
## F-statistic: 681.9 on 12 and 107 DF,  p-value: < 2.2e-16
```



5. The Box-Ljung test on the ARMA Model within the time-series regression model

```
##
##  Box-Ljung test
##
## data:  residuals(arma_best_fit)
## X-squared = 0.00047536, df = 1, p-value = 0.9826
```

b. R Code for Reproduce

1. Load the data

```
filename = "projectdata.txt"
data = read.table(filename, header=FALSE)

y = data[,4]
```

2. Plot the original time series

```
plot(y, type = 'o', col = 'blue', xlab = 'Month', ylab = 'Sales')
```

3. Seasonally difference the original time series

```r
library(tseries)
library(forecast)

y_ts <- ts(y, frequency = 12, start = c(2011, 1))
y_ts_seasdiff <- diff(y_ts, lag = frequency(y_ts), differences = 1)
```

4. Plot residuals and acf, pacf of the residuals

```r
par(mfrow=c(1,3))
plot(y_ts_seasdiff)
acf(y_ts_seasdiff, main=" ", lag.max = 120)
pacf(y_ts_seasdiff, main=" ", lag.max = 120)
```

5. Apply a first-order difference to the already seasonally differenced data

```r
y_ts_diff <- diff(y_ts_seasdiff)
```

6. Plot residuals and acf, pacf of the residuals

```r
par(mfrow=c(1,3))
plot(y_ts_diff)
acf(y_ts_diff, main=" ", lag.max = 120)
pacf(y_ts_diff, main=" ", lag.max = 120)
```

7. Grid search using R's for loop to identify the optimal ARIMA model

```r
p_range <- 0:2
d_range <- 1  # Based on prior differencing
q_range <- 0:2
P_range <- 0:1
D_range <- 1  # Based on prior seasonal differencing
Q_range <- 0:1
period <- 12

results <- data.frame(p = integer(), d = integer(), q = integer(),
                      P = integer(), D = integer(), Q = integer(),
                      AIC = numeric(), BIC = numeric())

for (p in p_range) {
  for (d in d_range) {
    for (q in q_range) {
      for (P in P_range) {
        for (D in D_range) {
          for (Q in Q_range) {
            model <- arima(y_ts, order = c(p, d, q), seasonal = list(order = c(P, D, Q), period = perio
            results <- rbind(results, data.frame(p = p, d = d, q = q, P = P, D = D, Q = Q,
                                                 AIC = AIC(model), BIC = BIC(model)))
          }
        }
      }
    }
  }
}

results <- results[order(results$AIC),]

library(knitr)
kable(head(results, 5),row.names=FALSE)
```

8. Fit the ARIMA Model with the result

```
fit_sarima <- arima(y_ts, order = c(0,1,1), seasonal = list(order = c(1,1,1), period = 12))
```

9. Plot residuals and acf, pacf of the residuals

```
par(mfrow=c(1,3))
plot(residuals(fit_sarima), main=" ")
acf(residuals(fit_sarima), main = " ", lag.max = 120)
pacf(residuals(fit_sarima), main = " ", lag.max = 120)
```

10. Fit the Linear Model and plot residuals with acf, pacf of the residuals

```
month = rep(factor(month.abb, levels=month.abb), length.out=length(y))
time_index <- seq_along(y_ts)
df <- data.frame(y_ts=y_ts, t=time_index, month=month)

linear_fit <- lm(y_ts ~., data = df)

linear_residuals <- residuals(linear_fit)

residuals_ts <- ts(linear_residuals, frequency = 12)

par(mfrow=c(1,3))
plot(residuals_ts, main="Residuals of Linear Model")
acf(residuals_ts, main = "ACF of Residuals", lag.max = 120)
pacf(residuals_ts, main = "PACF of Residuals", lag.max = 120)
```

11. Grid search using R's for loop to identify the optimal ARMA model

```
p_range <- 0:2  # Autoregressive order
q_range <- 0:2  # Moving average order


arma_results <- data.frame(p = integer(), q = integer(), AIC = numeric(), BIC = numeric())


for (p in p_range) {
  for (q in q_range) {
    arma_fit <- arima(linear_residuals, order = c(p, 0, q))

    arma_results <- rbind(arma_results, data.frame(p = p, q = q, AIC = AIC(arma_fit), BIC = BIC(arma_fi
  }
}

arma_results <- arma_results[order(arma_results$AIC),]

kable(head(arma_results, 5),row.names=FALSE)
```

12. Fit the ARMA Model with the result and plot residuals with acf, pacf of the residuals

```
arma_best_fit <- arima(linear_residuals, order = c(1, 0, 1))
par(mfrow=c(1,3))
plot(residuals(arma_best_fit), main=" ")
acf(residuals(arma_best_fit), main = " ", lag.max = 120)
pacf(residuals(arma_best_fit), main = " ", lag.max = 120)
```

13.Fit the Dynamic Linear Model and check convergence

```
library(dlm)
build_dlm = function(params) {
  dlmModPoly(order=2, dV=exp(params[1]), dW=exp(params[2:3])) +
    dlmModTrig(s=12, dV=0, dW=exp(rep(params[4], 11)))
}

dlm_fit = dlmMLE(y_ts, parm=c(0,0,0,0), build=build_dlm)
dlm_fit$convergence
```

14. Draw the smoothed trend line

```
dlm_mod = build_dlm(dlm_fit$par)

dlm_smooth = dlmSmooth(y_ts, dlm_mod)
dlm_level = dlm_smooth$s[,1]
```

```
plot(y_ts, ylab="Sales")
lines(dlm_level, col="red")
```

15. Decompose the time series into trend and seasonal components using the DLM's smoothed states vectors

```
x = cbind(y_ts, dropFirst(dlm_smooth$s[,1]),
          dropFirst(dlm_smooth$s[,-(1:2)]) %*% t(dlm_mod$FF)[-(1:2)])
colnames(x) = c("Sales", "Trend", "Seasonal")

plot(x, type="o", main="Sales per month")
```

16. Generate the forecast for the testing year

```
#Seasonal ARIMA Model
train_data <- window(y_ts, end = c(2019, 12))
test_data <- window(y_ts, start = c(2020, 1))

fit_sarima_train <- arima(train_data, order = c(0,1,1), seasonal = list(order = c(1,1,1), period = 12))

forecast_test <- forecast(fit_sarima_train, h = 12)

#Time-Series Regression Model
train_set <- window(y_ts, end = c(2019, 12))
test_set <- window(y_ts, start = c(2020, 1))

test_month = month.abb[1:12]
test_data = data.frame(t=109:120, month=test_month)

test = predict(linear_fit, test_data, interval="prediction")

test_naive = ts(test[,1], frequency=12, start=c(2020, 1))

testing_resids = predict(arma_best_fit, n.ahead=12)

test_resids = as.numeric(testing_resids$pred)
testvar_resids = as.numeric(testing_resids$se^2)

confvar = predict(linear_fit, test_data, interval="confidence", se.fit=TRUE)$se.fit^2

testing = test_naive + test_resids

testvar = confvar + testvar_resids
lower = testing - 1.96 * sqrt(testvar)
upper = testing + 1.96 * sqrt(testvar)

y_ts_f = ts(c(train_set, upper), start=start(train_set), frequency=frequency(train_set))
test_f = ts(c(train_set[length(train_set)], testing), start=end(train_set), frequency=frequency(train_s
# DLM
train_data_dlm <- window(y_ts, end = c(2019, 12))
test_data_dlm <- window(y_ts, start = c(2020, 1))

dlm_fit_train <- dlmFilter(train_data_dlm, dlm_mod)

dlm_test_preds <- dlmForecast(dlm_fit_train, nAhead=12)

test_preds <- dlm_test_preds$f[,1]
test_lower <- test_preds - 1.96 * sqrt(unlist(dlm_test_preds$Q))
test_upper <- test_preds + 1.96 * sqrt(unlist(dlm_test_preds$Q))
```

17. Plotting the forecast for the testing year

```
par(mfrow=c(1,3))
# Seasonal ARIMA Model - Plot the original data (first 9 years) and the testing forecast (10th year) wi
plot(y_ts, col = "black", ylab = "Monthly Sales", type = "n", xlim = c(2011, 2023), main = "Seasonal AR
lines(train_data, col = "black")
lines(test_data, col = "blue")
lines(forecast_test$mean, col = "blue")
lines(forecast_test$lower[,2], lty = 2, col = "blue")
lines(forecast_test$upper[,2], lty = 2, col = "blue")

legend("topleft",
       c("Original Data", "Testing Forecast", "Testing Prediction Intervals"),
       col = c("black", "blue", "blue"),
       lty = c(1, 1, 2))

# Time-Series Regression Model - Plot the original data (first 9 years) and the testing forecast (10th
plot(y_ts_f, ylab="Concentration (y_ts)", type="n", xlim = c(2011, 2023), main = "Time-Series Regressio
lines(train_set, col="black")
lines(test_f, col="blue")

lines(lower, lty=2, col="blue")
lines(upper, lty=2, col="blue")
legend("topleft", c("data", "testing", "95% prediction intervals"), col=c("black", "blue", "blue"),
       lty=c(1, 1, 1, 2))


# DLM  - Plot the original data (first 9 years) and the testing forecast (10th year) with prediction in
plot(y_ts, col = "black", ylab = "Monthly Sales", type = "n", xlim = c(2011, 2023), main = "Dynamic Lin
lines(train_data_dlm, col = "black")
lines(test_data_dlm, col = "blue")
lines(test_preds, col = "blue")
lines(test_lower, lty = 2, col = "blue")
lines(test_upper, lty = 2, col = "blue")

legend("topleft",
       c("Original Data", "Testing Forecast", "Testing Prediction Intervals"),
       col = c("black", "blue", "blue"),
       lty = c(1, 1, 2))
```

18. Generate the forecast for the future 6 months

```r
# Seasonal ARIMA model
forecast_future <- forecast(fit_sarima, h = 6)

# Time-Series Regression Model
new_month = month.abb[1:6]
new_data = data.frame(t=121:126, month=new_month)

forecast = predict(linear_fit, new_data, interval="prediction")

preds_naive = ts(forecast[,1], frequency=12, start=c(2021, 1))

forecast_resids = predict(arma_best_fit, n.ahead=6)

preds_resids = as.numeric(forecast_resids$pred)
predvar_resids = as.numeric(forecast_resids$se^2)

confvar = predict(linear_fit, new_data, interval="confidence", se.fit=TRUE)$se.fit^2

preds = preds_naive + preds_resids

predvar = confvar + predvar_resids
lower = preds - 1.96 * sqrt(predvar)
upper = preds + 1.96 * sqrt(predvar)

y_ts_f = ts(c(y_ts, upper), start=start(y_ts), frequency=frequency(y_ts))
preds_f = ts(c(y_ts[length(y_ts)], preds), start=end(y_ts), frequency=frequency(y_ts))
preds_naive_f = ts(c(y_ts[length(y_ts)], preds_naive), start=end(y_ts),
                   frequency=frequency(y_ts))

# DLM
dlm_filt = dlmFilter(y_ts, dlm_mod)
dlm_preds = dlmForecast(dlm_filt, nAhead=6)

preds = dlm_preds$f[,1]
lower = preds - 1.96 * sqrt(unlist(dlm_preds$Q))
upper = preds + 1.96 * sqrt(unlist(dlm_preds$Q))

dlm_f = ts(c(y_ts, upper), start=start(y_ts), frequency=frequency(y_ts))
preds_f = ts(c(y_ts[length(y_ts)], preds), start=end(y_ts), frequency=frequency(y_ts))
```

19. Plotting the forecast for the future 6 months

```r
par(mfrow=c(1,3))
# Plot Seasonal Arima Model data, forecasts and prediction intervals
plot(y_ts, col="black", ylab="Monthly Sales", type="n", xlim = c(2011, 2023), main = "Seasonal ARIMA")
lines(y_ts, col="black")
lines(forecast_future$mean, col="red")
lines(forecast_future$lower[,2], lty=2, col="red")
lines(forecast_future$upper[,2], lty=2, col="red")
```

```
legend("topleft", c("Data", "Forecasts", "95% Prediction Intervals"),
       col=c("black", "red", "red"), lty=c(1, 1, 2))

# Plot Time-Series Regression Model data, forecasts and prediction intervals
plot(y_ts_f, ylab="Concentration (y_ts)", type="n", xlim = c(2011, 2023), main = "Time-Series Regressio
lines(y_ts, col="black")
lines(preds_f, col="red")
lines(lower, lty=2, col="red")
lines(upper, lty=2, col="red")
legend("topleft", c("data", "forecasts", "95% prediction intervals"), col=c("black", "red", "red"),
       lty=c(1, 1, 1, 2))

# Plot DLM data, forecasts and prediction intervals
plot(dlm_f, col="red", ylab="Monthly Sales", type="n", xlim = c(2011, 2023), main = "Dynamic Linear Mod
lines(y_ts, col="black")
lines(preds_f, col="red")
lines(lower, lty=2, col="red")
lines(upper, lty=2, col="red")
legend("topleft", c("Data", "Forecasts", "95% Prediction Intervals"),
       col=c("black", "red", "red"), lty=c(1, 1, 2))
```

20. Caculate AIC and BIC for three models and compare

```
# AIC and BIC for Seasonal ARIMA Model
aic_sarima <- AIC(fit_sarima)
bic_sarima <- BIC(fit_sarima)

# AIC and BIC for Regression with ARMA Errors Model
n <- length(residuals_ts)
k <- length(coef(linear_fit)) + length(coef(arma_best_fit))

loglik <- logLik(arma_best_fit)

aic_combined <- -2 * loglik + 2 * k
bic_combined <- -2 * loglik + log(n) * k

# AIC and BIC for Dynamic Linear Model
logLik_dlm <- -dlm_fit$value
num_params_dlm <- length(dlm_fit$par) + 1
aic_dlm <- -2 * logLik_dlm + 2 * num_params_dlm
bic_dlm <- -2 * logLik_dlm + num_params_dlm * log(length(y_ts))

#Comparison table
model_comparison <- data.frame(
  Model = c("Seasonal ARIMA", "Time-Series Regression", "Dynamic Linear Model"),
  AIC = c(aic_sarima, aic_combined, aic_dlm),
  BIC = c(bic_sarima, bic_combined, bic_dlm)
)

kable(model_comparison, format = "markdown", col.names = c("Model", "AIC", "BIC"), caption = "Model Com
```

Appendix 1. Augmented Dickey-Fuller test on original time-series

```
adf_result <- adf.test(y_ts_diff, alternative = "stationary")
adf_result
```

Appendix 2. STL decomposition plot

```
y_ts <- ts(y, frequency = 12, start = c(2011, 1))
decomposed <- stl(y_ts, s.window = "periodic")
plot(decomposed)
```

Appendix 3. The Box-Ljung test on the residuals of Seaonal ARIMA Model

```
Box.test(residuals(fit_sarima), type = "Ljung-Box")
```

Appendix 4. The Summary and fitted value plot of the linear regression model

```
month = rep(factor(month.abb, levels=month.abb), length.out=length(y))
time_index <- seq_along(y_ts)
df <- data.frame(y_ts=y_ts, t=time_index, month=month)

linear_fit <- lm(y_ts ~., data = df)

linear_residuals <- residuals(linear_fit)

kable(coef(summary(linear_fit)))
```

Appendix 5. The Box-Ljung test on the ARMA Model within the time-series regression model

```
Box.test(residuals(arma_best_fit), type = "Ljung-Box")
```

Theoretical Part

Question 1

Identify the model

For the given model $X_t = \varepsilon_t + \theta \cdot \varepsilon_{t-2}$, where $\varepsilon_t \sim N(0, \sigma^2)$, I identify it as a Moving Average process of order 2 (MA(2)). This means the model forecasts future values based on past errors, specifically the error two periods ago, without considering past values of X_t itself. There are no components that take into account the past values of X_t (autoregressive) or differences between them (integrated), and no indication of a seasonal component. Therefore, I classify this model as ARIMA$\{(0, 0, 2) \times (0, 0, 0)s\}$, as a non-seasonal moving average model.

Stationarity and invertibility

Stationarity is the constancy of a time series' statistical properties over time. For the given model, this concept is straightforward due to the absence of autoregressive (AR) and differencing components. Such elements typically introduce time-dependent changes in a series, affecting its mean, variance, and other statistical properties. However, in this case, the model's reliance solely on the moving average terms at specific lags means that it maintains consistent statistical properties over time, qualifying it as stationary.

Invertibility is a desirable property in moving average models, indicating that the model can be expressed in terms of past values instead of past errors. For the given MA(2) model, invertibility hinges on the magnitude of the coefficient θ. Specifically, the model is invertible when the absolute value of θ, denoted as $|\theta|$, is less than 1. This mathematical constraint ensures that the moving average process can be restructured in a way that past errors can be uniquely deduced from the current and past values of X_t. Essentially, if $|\theta| < 1$, the model's past error terms can be inferred from its observed values, lending to easier interpretation and understanding.

Infinite Order Autoregression Coefficients (π_k)

In this model, the presence of ε_{t-2} indicates that X_t is influenced by the error term from two periods ago. This influence is quantified by the coefficient θ. The absence of any other error terms or lags in the model suggests that the influence of past values on X_t is limited to this specific lag. Therefore, in the infinite AR representation, $\pi_2 = \theta$.

For all other lags, the coefficients π_k are zero because the model does not incorporate any influence from error terms at these lags. Therefore, the coefficient $\pi_1 = 0$ (since there is no influence from the immediate past error term), and similarly, $\pi_k = 0$ for k>2. This pattern of coefficients distinctly reflects the model's structural emphasis on the second lag.

Forecasting Equations and Error Variance

The given model $X_t = \varepsilon_t + \theta \cdot \varepsilon_{t-2}$, where ε_t is a white noise process with a normal distribution N(0, σ^2). It is a MA(2) model (Moving Average model of order 2), which means it uses the error terms from the current and two periods ago to predict the value at time t.

For the one-step ahead forecast(k=1), $\hat{x}_{n+1} = 0$. This is because the model does not utilize information from the immediate past, either values or errors, to predict the next value. The structure of the model is focusing on the error term two periods ago, making the immediate next forecast independent of any past information.

For the two-step ahead forecast(k=2), $\hat{x}n + 2 = \theta \cdot \varepsilon_n$. The last known error term, ε_n, plays an important role. Because of the model's structure, this forecast incorporates the influence of the error term from two periods ago, which is the most recent error term available at the time of forecasting.

For beyond two steps($k \geq 3$), the forecasts $\hat{x}_{n+k} = 0$ again. This pattern highlights the diminishing influence of the model for longer-term forecasts, as the model's structure does not account for error terms beyond the second lag in its predictions.

The forecast error variance is an indicator of the expected accuracy of the forecasts. It can also be varied depending on how far ahead the forecast is made:

For the one-step ahead forecast(k=1), the error variance is equal to σ^2. This is because the forecast value is zero, and the only source of error is the immediate next period's error term, ε_{n+1}, which has a variance of σ^2.

For the two-step ahead forecast(k=2), the error variance is $(1 + \theta^2)\sigma^2$. The forecast incorporates ε_n, but the actual value at $n + 2$ will also be influenced by the then-unknown error term ε_{n+2}. The error variance in this case is the variance of the sum of two independent error terms: $\theta \cdot \varepsilon_n$ (already known at the time of forecasting) and ε_{n+2} (unknown at the time of forecasting). Since ε_n and ε_{n+2} are independent and each has a variance of σ^2, the total variance is $(1+\theta^2)\sigma^2$.

For forecasts beyond two steps($k \geq 3$), the forecast error variance stabilizes at σ^2. This is because the forecasts for these horizons are zero, and the only source of error is the error term corresponding to the forecast period, which has a variance of σ^2.

Question 2

The model provided for the seasonal effects is expressed as $s_t = \sum_{k=1}^{K} \left\{ a_k \cos\left(\frac{2\pi kt}{p}\right) + b_k \sin\left(\frac{2\pi kt}{p}\right) \right\}$. This model essentially breaks down the seasonal effect, s_t, into a series of sine and cosine waves. Each wave, characterized by coefficients a_k and b_k, represents different seasonal patterns, where p denotes the period of the cycle (such as 12 months in a year for monthly data).

Condition 1: Repetition of the Seasonal Pattern

The first condition requires that the seasonal effect s_t repeats at regular intervals, such as every p periods. This condition is satisfied by the periodic nature of the sine and cosine functions. These functions complete a full cycle over an interval of 2π, which means for each k, the terms $\cos\left(\frac{2\pi kt}{p}\right)$ and $\sin\left(\frac{2\pi kt}{p}\right)$ repeat their values every p periods. Therefore, the seasonal effect modeled by the Fourier series will demonstrate this required periodicity, effectively capturing the repeating nature of seasonal patterns.

Condition 2: Zero Sum Over a Complete Cycle

The second condition is that the sum of the seasonal effects over one complete cycle should be zero. This is a critical condition to ensure that the model does not imply a long-term increasing or decreasing trend in the seasonal effect. Due to the properties of sine and cosine functions, the sum of these functions over one complete cycle (from 0 to 2π) is zero. This is because these functions are symmetric about the horizontal axis over their respective cycles. Therefore, when the Fourier series is summed over a complete period of p, the overall sum of s_t becomes zero. This ensures that the seasonal effects, as modeled, do not bias the data towards a long-term trend.