Solutions

Mock Exam 2025

```
library(fpp2)
library(knitr)
library(forecast)
```

Assessment Guide

This solutions document demonstrates the expected depth and quality of analysis for the Advanced Financial Data Analytics exam. When interpreting time series data:

1. **Descriptive analysis** should identify key features (trend, seasonality, cycles) with specific references to time periods where patterns change.
2. **Visual interpretations** should connect statistical features to business/economic context where appropriate.
3. **Statistical discussions** should include both the technical meaning and practical implications of results.
4. **Justifications** for modelling choices should reference both graphical evidence and theoretical principles.

Question 1

b. Plotting the time series:

To begin our analysis, let's create professional-looking plots for each of the three commodity series: `gold`, `woolyrnq`, and `gas`. We'll use the `autoplot()` function from the `forecast` package and add appropriate labels and titles.

```r
library(ggplot2)

# Enhanced visualisation with proper formatting and consistent design elements
autoplot(gold) +
  labs(title = "Daily Morning Gold Prices",
       subtitle = "Strong upward trend with notable volatility",
       x = "Date",
       y = "Price (USD)") +
  theme_minimal() +
  theme(
    plot.title = element_text(face = "bold"),
    plot.subtitle = element_text(color = "darkgrey"),
    panel.grid.minor = element_blank()
  ) +
  # Adding annotation for outlier
  annotate("text", x = which.max(gold), y = max(gold, na.rm=TRUE),
           label = "Major outlier", vjust = -1, hjust = 1.1)
```


```r
# Similarly enhanced plots for woolyrnq and gas would follow the same pattern
# autoplot(woolyrnq) +
#   labs(title = "Quarterly Wool Production in Australia",
#        subtitle = "Clear seasonal pattern with declining trend",
```

```
#        x = "Date", y = "Production") +
#   theme_minimal() +
#   theme(
#     plot.title = element_text(face = "bold"),
#     plot.subtitle = element_text(color = "darkgrey"),
#     panel.grid.minor = element_blank()
#   )
#
# autoplot(gas) +
#   labs(title = "Australian Monthly Gas Production",
#        subtitle = "Increasing trend with seasonal variation",
#        x = "Date", y = "Production") +
#   theme_minimal() +
#   theme(
#     plot.title = element_text(face = "bold"),
#     plot.subtitle = element_text(color = "darkgrey"),
#     panel.grid.minor = element_blank()
#   )
```

From the plots, we can make the following observations:

- The `gold` series shows an overall increasing trend with some fluctuations. There appears to be an outlier with an unusually high price. This outlier could correspond to a period of market volatility or a specific economic event that triggered a price spike.
- The `woolyrnq` series exhibits a clear seasonal pattern, with production peaking in certain quarters and declining in others. There is also a slight overall downward trend, suggesting a gradual decline in the wool industry over this period.
- The `gas` series shows an increasing trend with some seasonal variation. The seasonality appears to be less pronounced compared to the `woolyrnq` series, indicating more stable year-round demand with gradual production growth.

b. Frequency of each series:

To determine the frequency of each commodity series, we can use the `frequency()` function.

```
frequency(gold)
```

```
[1] 1
```

```
frequency(woolyrnq)
```

```
[1] 4
```

```r
frequency(gas)
```

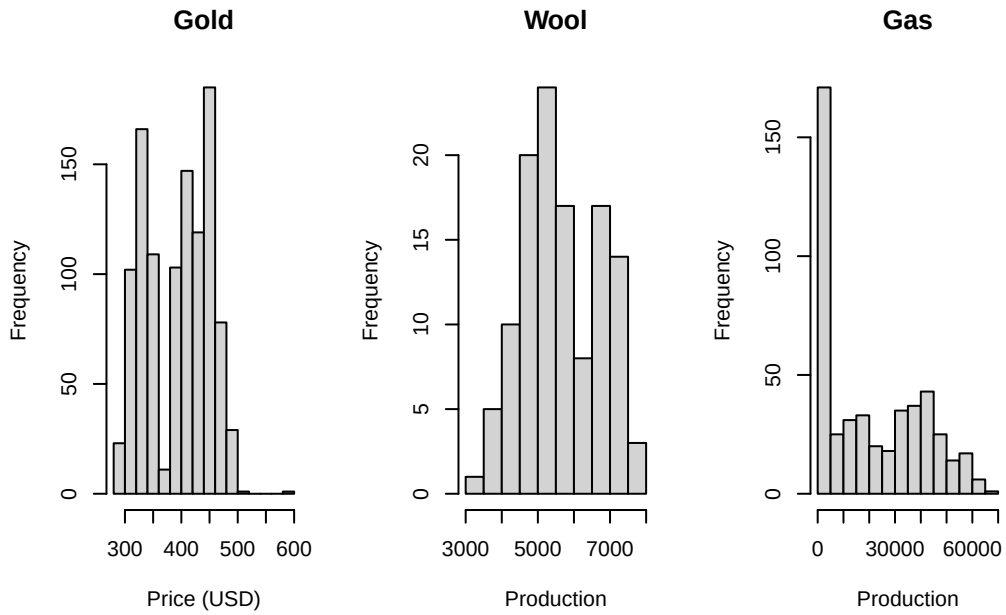
```
[1] 12
```

The output reveals that: - The `gold` series has a frequency of 1, indicating daily data observed on trading days. - The `woolyrnq` series has a frequency of 4, indicating quarterly data, which aligns with the seasonal patterns observed in the time plot. - The `gas` series has a frequency of 12, indicating monthly data, which explains the finer granularity of seasonal patterns compared to the wool series.

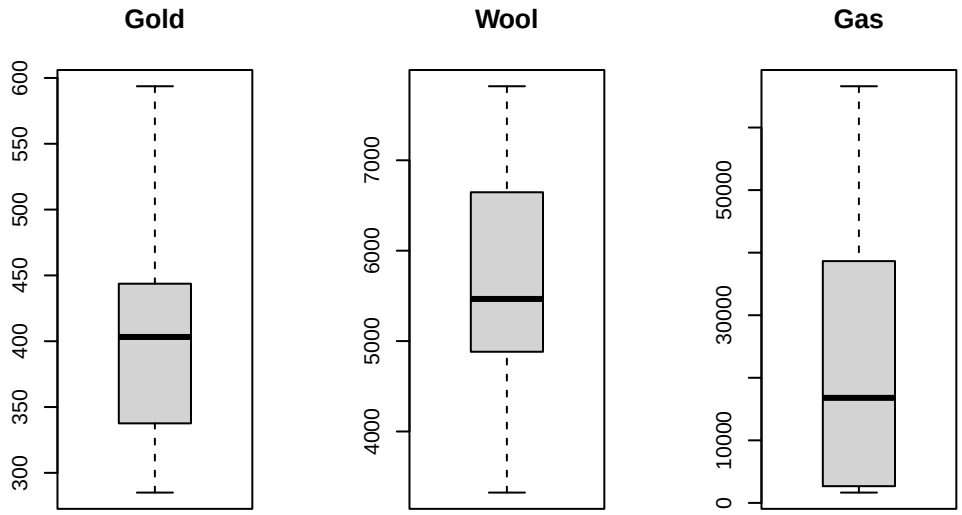
c. Checking for outliers:

To check for outliers in each series, we can use various methods such as visual inspection of plots, examining summary statistics, or applying statistical tests. Let's start with a simple visual approach using histograms and boxplots.

```r
par(mfrow = c(1, 3))
hist(gold, main = "Gold", xlab = "Price (USD)")
hist(woolyrnq, main = "Wool", xlab = "Production")
hist(gas, main = "Gas", xlab = "Production")
```



```
par(mfrow = c(1, 3))
boxplot(gold, main = "Gold")
boxplot(woolyrnq, main = "Wool")
boxplot(gas, main = "Gas")
```



From the histograms and boxplots, we can observe that: - The `gold` series has an extreme outlier on the higher end, which is consistent with our observation from the time series plot. This outlier significantly exceeds the upper whisker of the boxplot, indicating a value that deviates substantially from the general distribution. - The `woolyrnq` and `gas` series do not show any clear outliers based on these visual methods, suggesting more consistent patterns in production volumes over time.

To identify the specific outlier in the `gold` series, we can use the `which.max()` function to find the index of the maximum value and then retrieve the corresponding value.

```
outlier_index <- which.max(gold)
gold[outlier_index]
```

```
[1] 593.7
```

The outlier value is 593.7, which is significantly higher than the other prices in the series.

To further confirm that this is indeed an outlier, we can calculate the z-score of the outlier value and compare it to a threshold (e.g., 3 or 4). A z-score greater than the threshold indicates a probable outlier.

```
z_score <- (gold[outlier_index] - mean(gold,na.rm=T)) / sd(gold,na.rm=T)
z_score
```

```
[1] 3.553807
```

The z-score of 3.55 is much greater than the typical threshold of 3 or 4, confirming that the identified value is an outlier.

In summary, based on the visual inspection and statistical analysis, we can conclude that the `gold` series contains a significant outlier at the observation with the highest price. This outlier is substantially larger than the other prices in the series and lies far beyond the normal range of the data. It is important to investigate the cause of this outlier and consider its impact on any further analysis or modeling of the `gold` series.

The `woolyrnq` and `gas` series do not exhibit any clear outliers based on the methods used in this analysis. However, it is always a good practice to use multiple outlier detection techniques and consider the domain knowledge and context of the data to make robust conclusions about the presence of outliers.

Assessment Guidance - Question 1

Excellent answers (70%+) will: - Produce professionally formatted visualizations with appropriate annotations - Clearly identify and quantify outliers using multiple methods - Provide contextual interpretation of the frequency results - Offer thoughtful discussion about the implications of outliers for further analysis

Good answers (60-69%) will: - Create correct visualizations with proper labels - Identify outliers using at least one statistical method - Correctly interpret the frequency results - Discuss basic implications of outliers

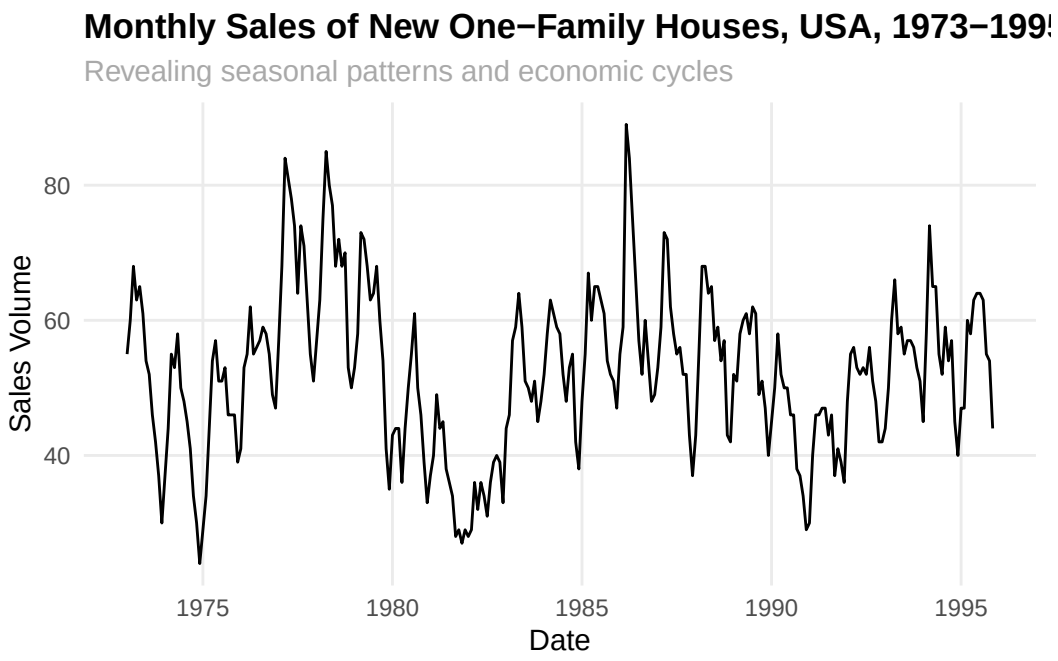
Satisfactory answers (50-59%) will: - Produce basic but correct visualizations - Identify obvious outliers visually - State the frequency of each series - Provide minimal interpretation of results

Question 2

To explore the features of the given time series (`hsales`, `usdeaths`, `bricksq`, `sunspotarea`, `gasoline`), we can use various time series graphics functions in R. Let's analyse each series individually.

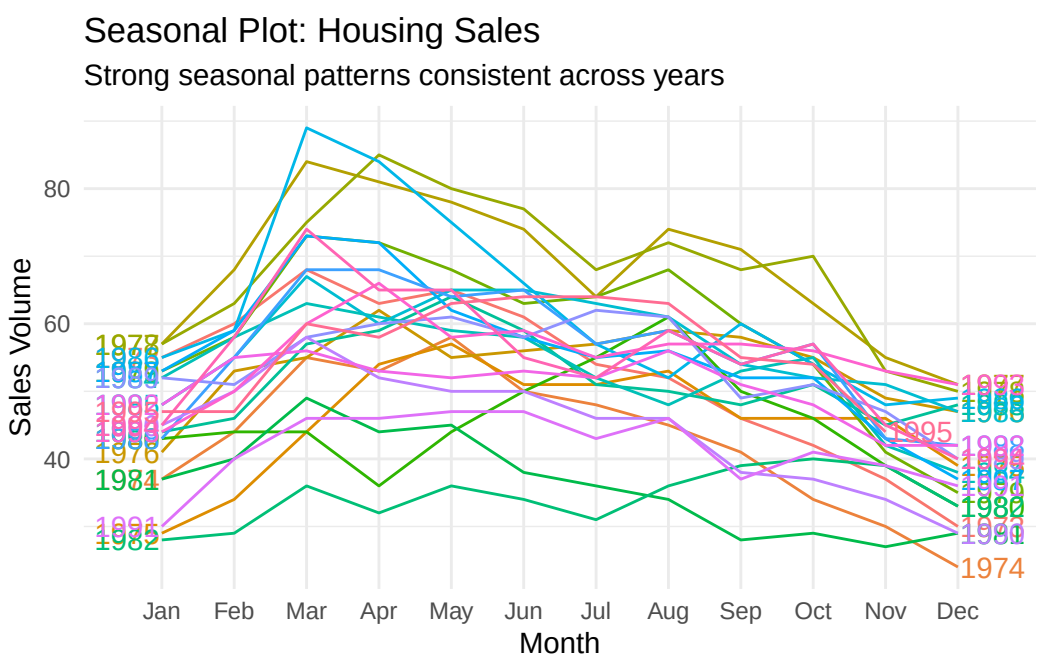
1. `hsales` (Monthly sales of new one-family houses, USA, 1973–1995):

```
autoplot(hsales) +
  labs(title = "Monthly Sales of New One-Family Houses, USA, 1973-1995",
       subtitle = "Revealing seasonal patterns and economic cycles",
       x = "Date",
       y = "Sales Volume") +
  theme_minimal() +
  theme(
    plot.title = element_text(face = "bold"),
    plot.subtitle = element_text(color = "darkgrey"),
    panel.grid.minor = element_blank()
  )
```

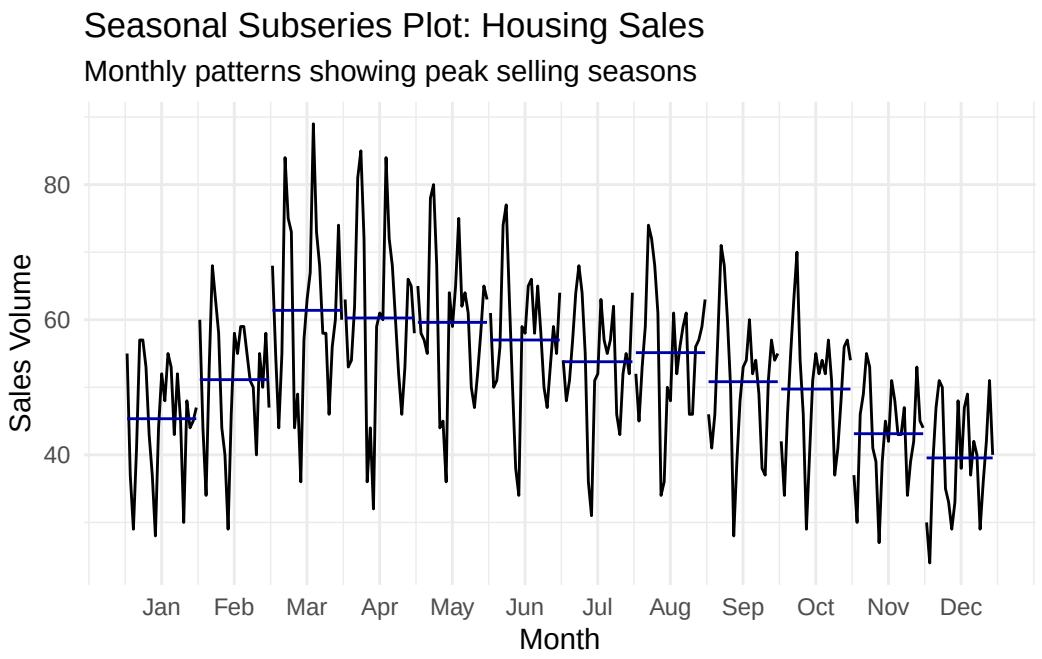



```
ggseasonplot(hsales, year.labels = TRUE, year.labels.left = TRUE) +
  labs(title = "Seasonal Plot: Housing Sales",
       subtitle = "Strong seasonal patterns consistent across years",
```

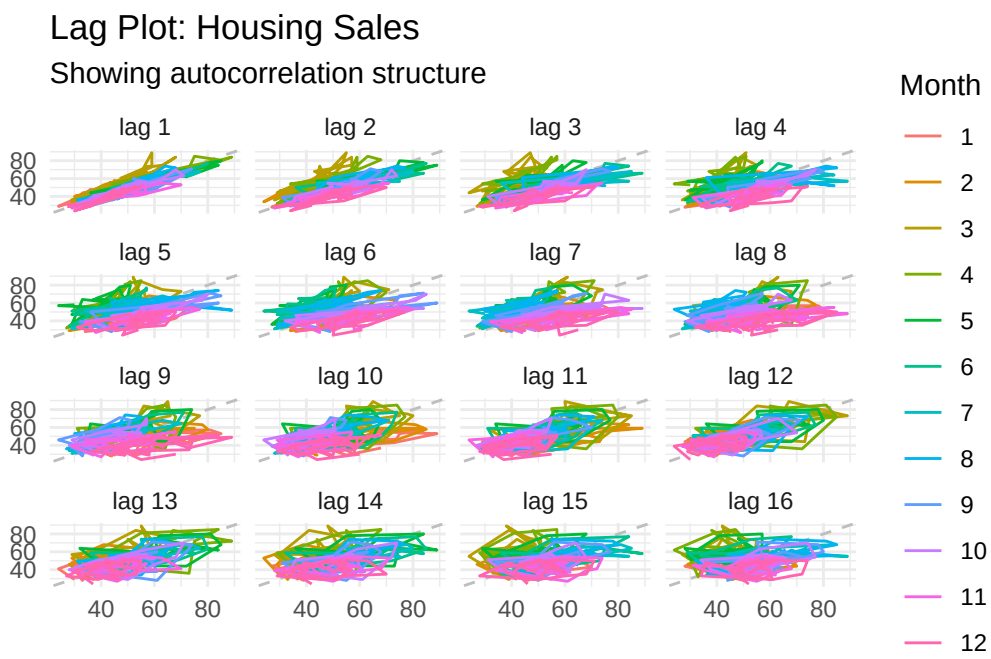
```
       y = "Sales Volume") +
  theme_minimal()
```



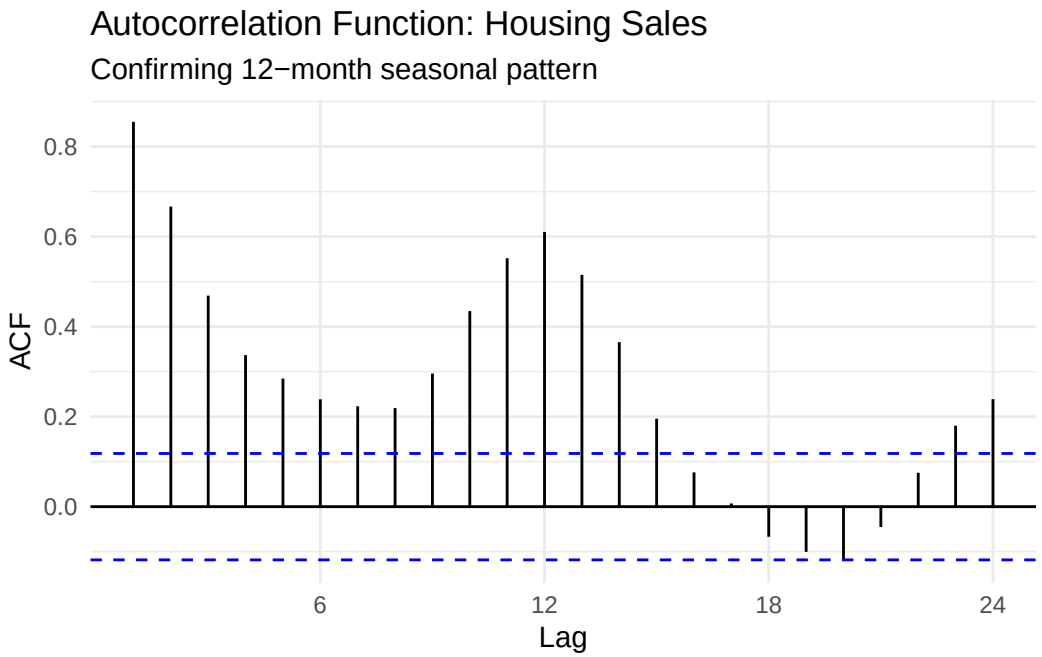

```
ggsubseriesplot(hsales) +
  labs(title = "Seasonal Subseries Plot: Housing Sales",
       subtitle = "Monthly patterns showing peak selling seasons",
       y = "Sales Volume") +
  theme_minimal()
```




```
gglagplot(hsales) +
  labs(title = "Lag Plot: Housing Sales",
       subtitle = "Showing autocorrelation structure") +
  theme_minimal()
```




```
ggAcf(hsales) +
  labs(title = "Autocorrelation Function: Housing Sales",
       subtitle = "Confirming 12-month seasonal pattern") +
  theme_minimal()
```

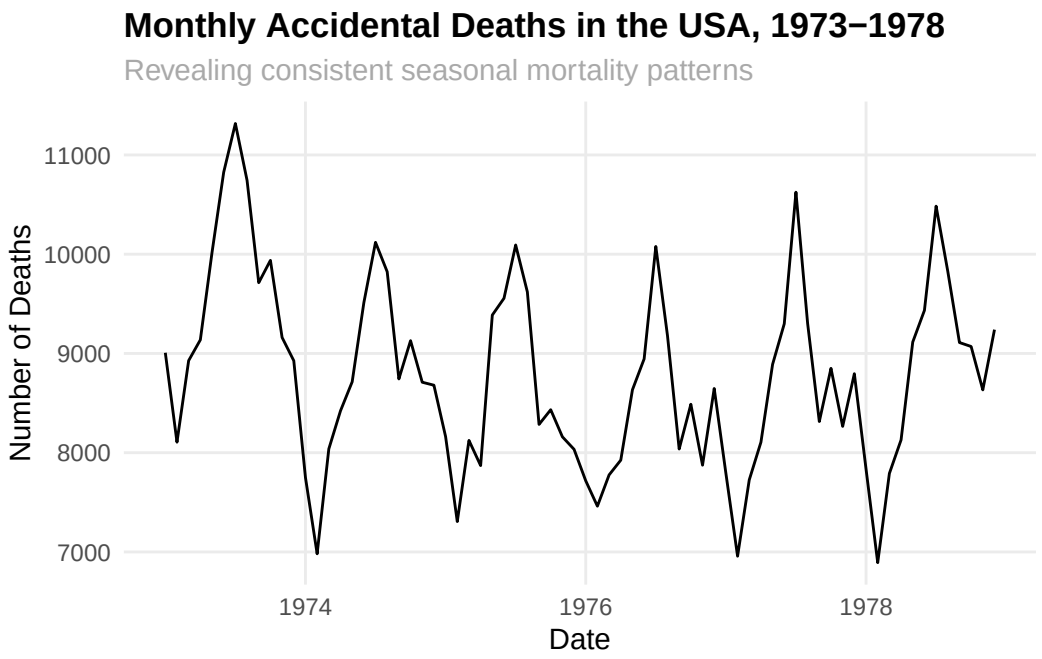


- The time plot shows a clear seasonal pattern and some long-term fluctuations. These fluctuations likely correspond to economic cycles in the US housing market, with visible dips during recessionary periods.
- The seasonal plot confirms a strong and consistent seasonal pattern, with sales typically peaking in spring and summer months (March-August) and dropping in winter.
- The subseries plot shows that the seasonal pattern is similar across years, indicating stability in the housing market's seasonal dynamics despite economic changes.
- The lag plot indicates a positive relationship between consecutive observations, possibly due to trend or seasonal effects. This suggests momentum in the housing market, where strong sales months tend to be followed by relatively strong months.
- The ACF plot shows significant autocorrelations at multiples of lag 12, confirming the presence of annual seasonality that real estate analysts would expect.

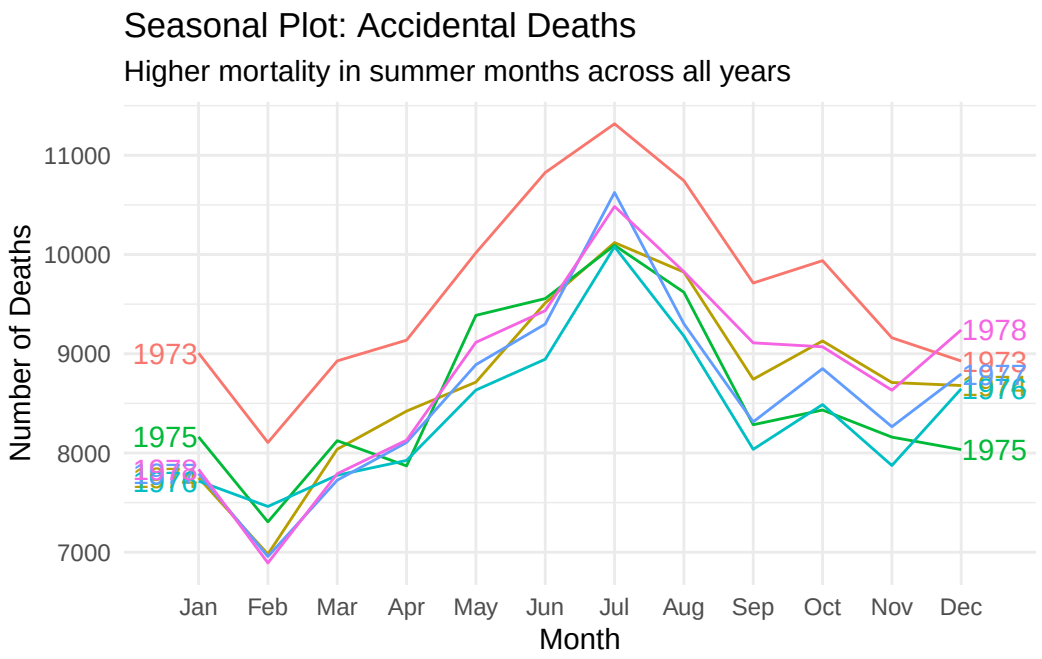
2. `usdeaths` (Monthly accidental deaths in the USA, 1973–1978):

```
autoplot(usdeaths) +
  labs(title = "Monthly Accidental Deaths in the USA, 1973-1978",
       subtitle = "Revealing consistent seasonal mortality patterns",
       x = "Date",
       y = "Number of Deaths") +
  theme_minimal() +
  theme(
    plot.title = element_text(face = "bold"),
```

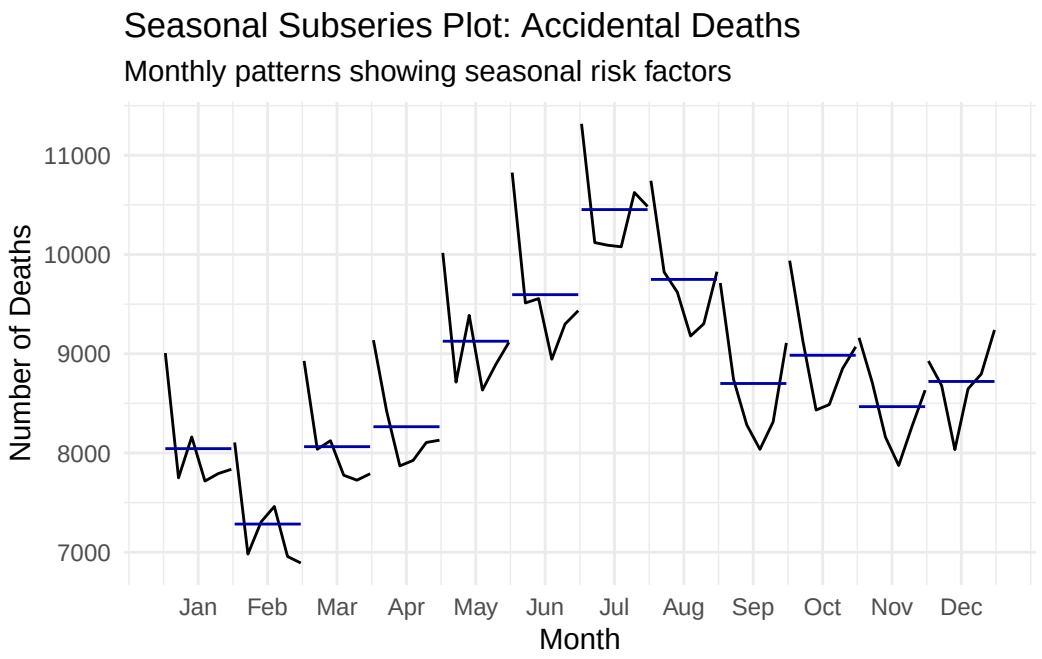
```
    plot.subtitle = element_text(color = "darkgrey"),
    panel.grid.minor = element_blank()
  )
```



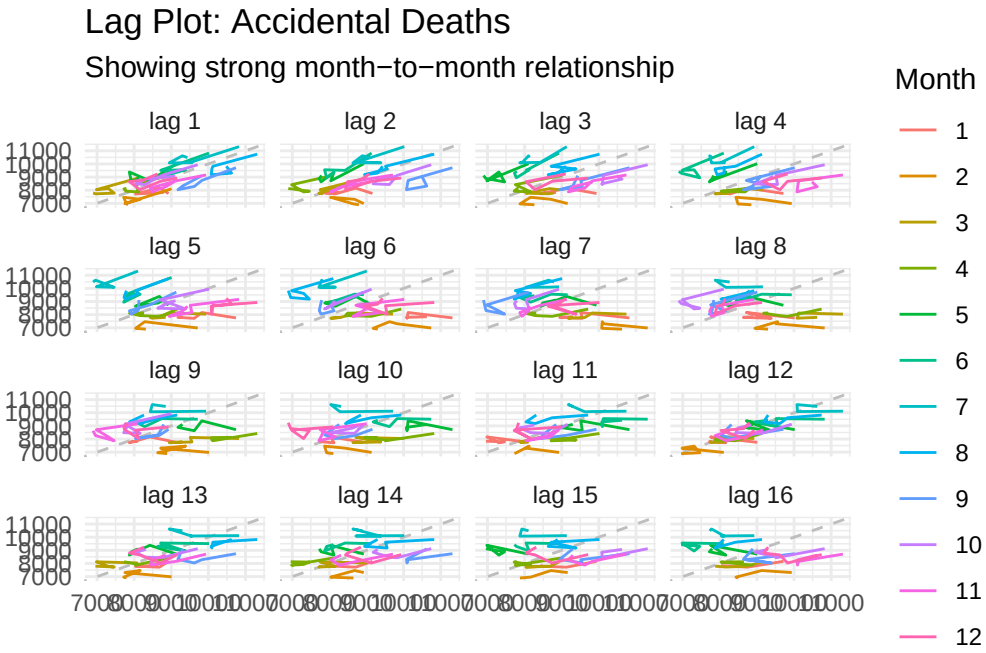

```
ggseasonplot(usdeaths, year.labels = TRUE, year.labels.left = TRUE) +
  labs(title = "Seasonal Plot: Accidental Deaths",
       subtitle = "Higher mortality in summer months across all years",
       y = "Number of Deaths") +
  theme_minimal()
```



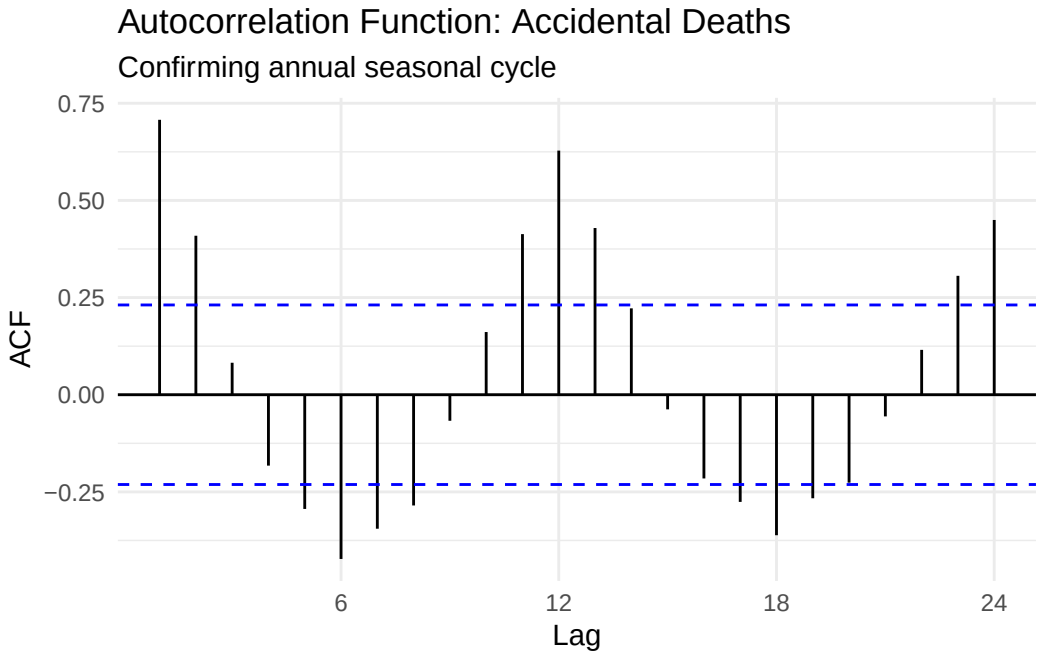

```
ggsubseriesplot(usdeaths) +
  labs(title = "Seasonal Subseries Plot: Accidental Deaths",
       subtitle = "Monthly patterns showing seasonal risk factors",
       y = "Number of Deaths") +
  theme_minimal()
```




```
gglagplot(usdeaths) +
  labs(title = "Lag Plot: Accidental Deaths",
       subtitle = "Showing strong month-to-month relationship") +
  theme_minimal()
```




```
ggAcf(usdeaths) +
  labs(title = "Autocorrelation Function: Accidental Deaths",
       subtitle = "Confirming annual seasonal cycle") +
  theme_minimal()
```

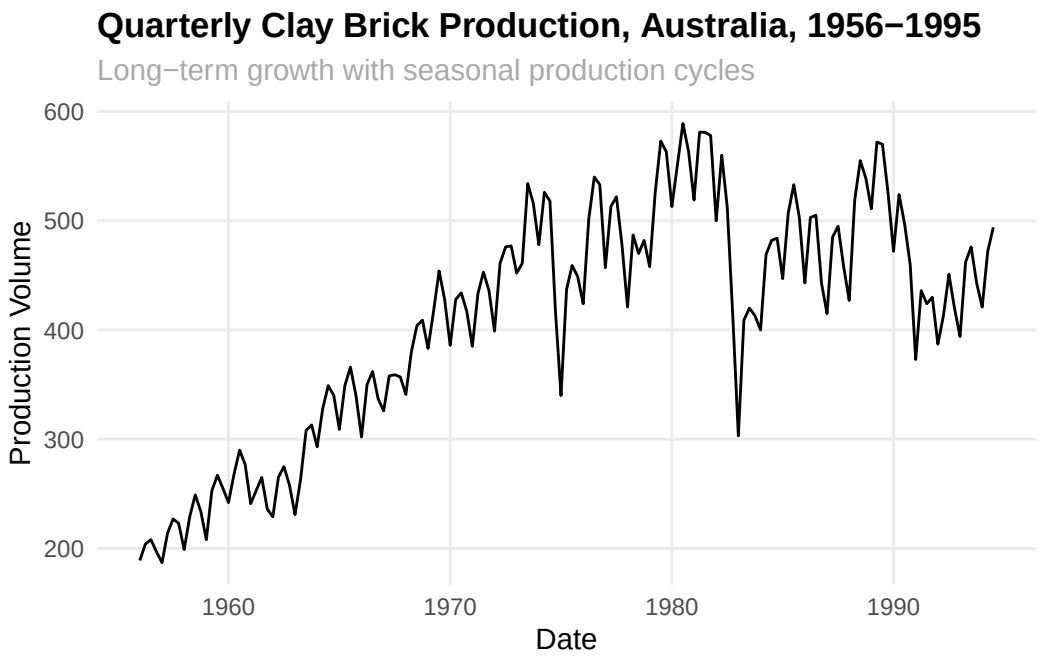


- The time plot shows a clear seasonal pattern but no obvious trend. This suggests that the overall mortality risk remained relatively stable during this period, though with pronounced seasonal variation.
- The seasonal plot confirms the presence of seasonality, with higher deaths in summer months (particularly July-August). This seasonal peak may be associated with increased outdoor activities, travel, and water-related accidents during summer.
- The subseries plot shows that the seasonal pattern is consistent across years, indicating stable seasonal risk factors throughout the observation period.
- The lag plot shows a positive relationship between consecutive observations due to seasonality, with a clear clustering pattern revealing the seasonal grouping.
- The ACF plot has significant autocorrelations at multiples of lag 12, confirming the annual seasonality pattern that public health officials would monitor.

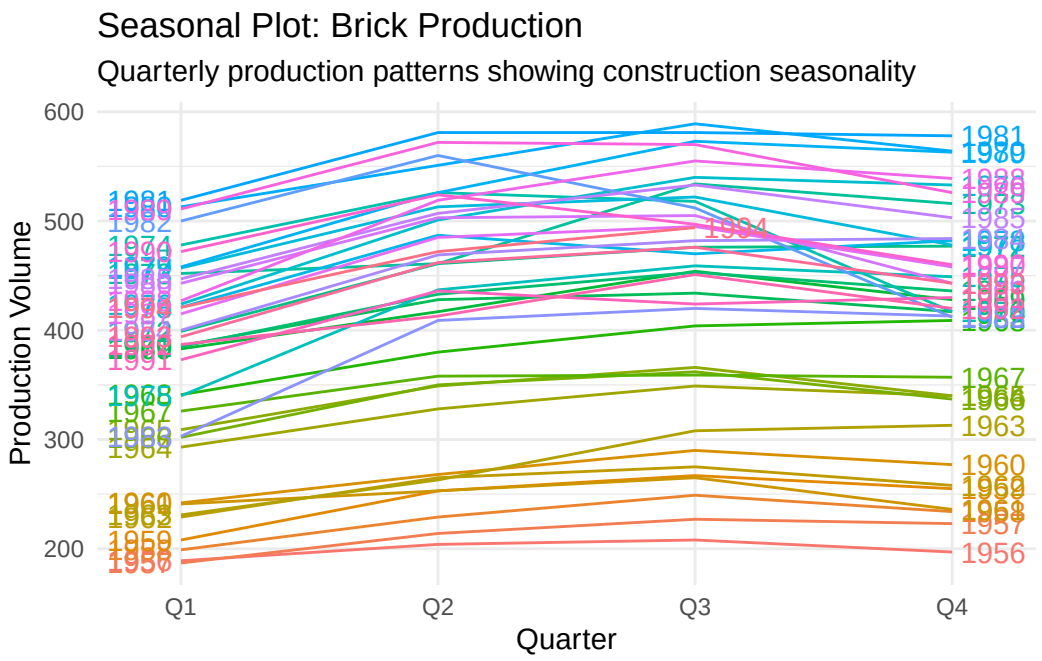
3. `bricksq` (Quarterly clay brick production, Australia, 1956–1995):

```
autoplot(bricksq) +
  labs(title = "Quarterly Clay Brick Production, Australia, 1956-1995",
       subtitle = "Long-term growth with seasonal production cycles",
       x = "Date",
       y = "Production Volume") +
  theme_minimal() +
  theme(
    plot.title = element_text(face = "bold"),
```

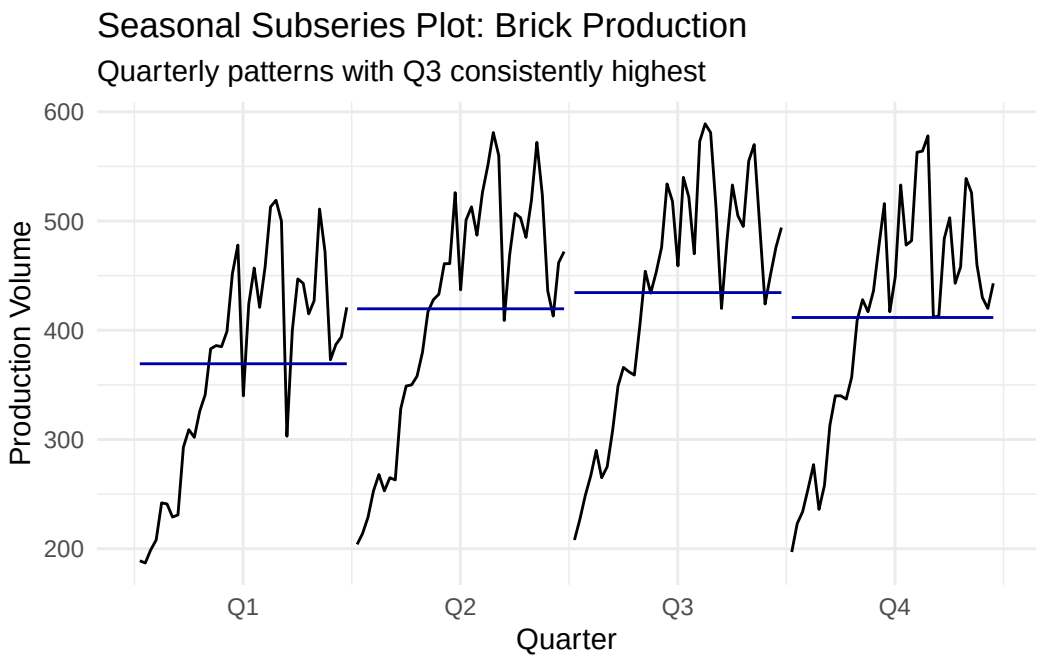
```
    plot.subtitle = element_text(color = "darkgrey"),
    panel.grid.minor = element_blank()
  )
```



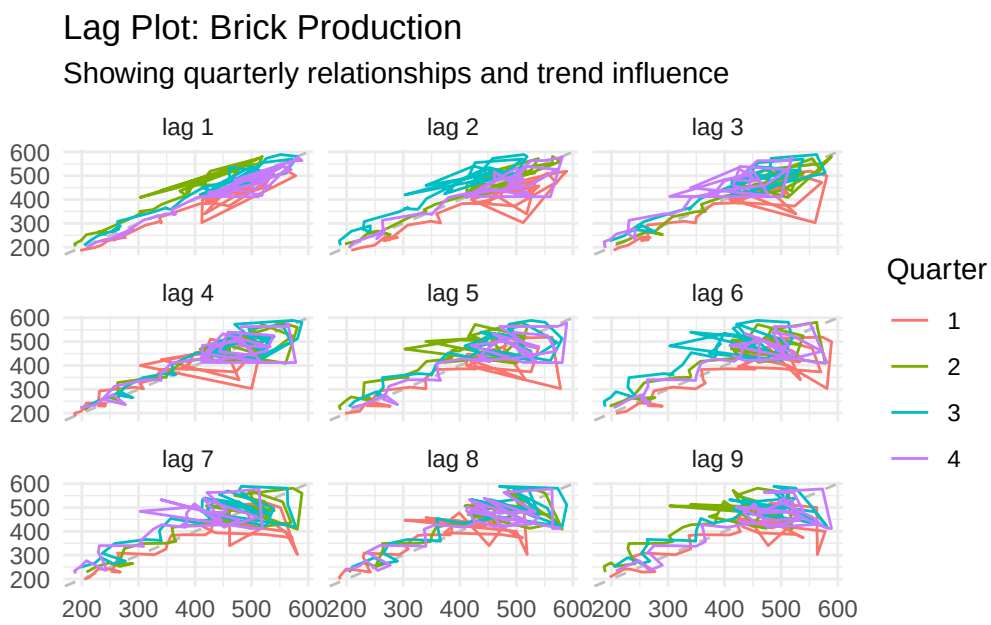

```
ggseasonplot(bricksq, year.labels = TRUE, year.labels.left = TRUE) +
  labs(title = "Seasonal Plot: Brick Production",
       subtitle = "Quarterly production patterns showing construction seasonality",
       y = "Production Volume") +
  theme_minimal()
```



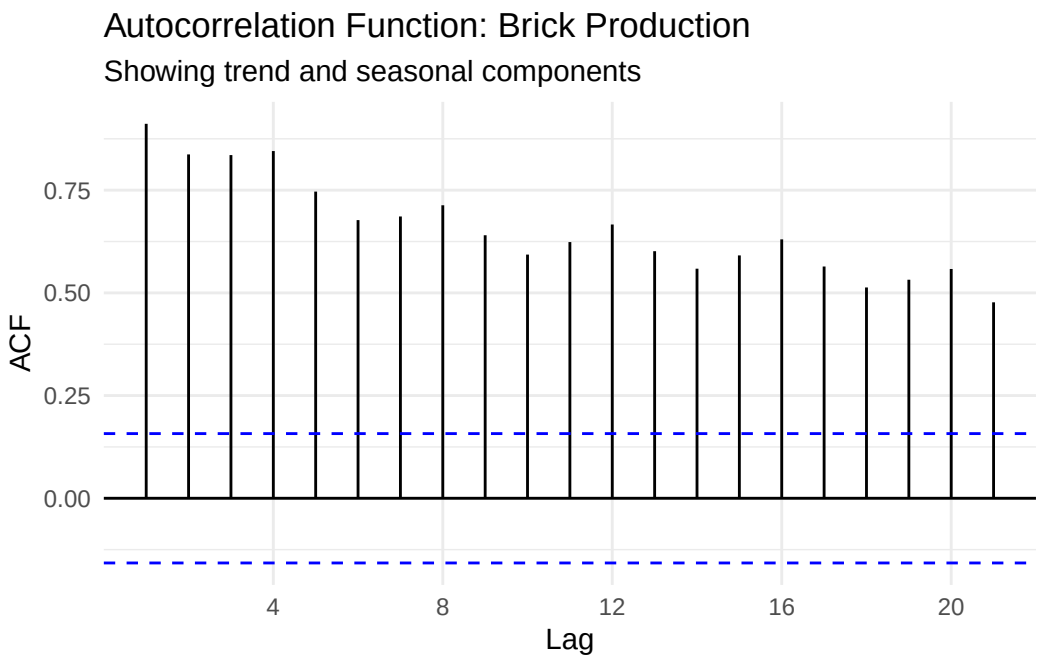

```
ggsubseriesplot(bricksq) +
  labs(title = "Seasonal Subseries Plot: Brick Production",
       subtitle = "Quarterly patterns with Q3 consistently highest",
       y = "Production Volume") +
  theme_minimal()
```




```
gglagplot(bricksq) +
  labs(title = "Lag Plot: Brick Production",
       subtitle = "Showing quarterly relationships and trend influence") +
  theme_minimal()
```




```
ggAcf(bricksq) +
  labs(title = "Autocorrelation Function: Brick Production",
       subtitle = "Showing trend and seasonal components") +
  theme_minimal()
```

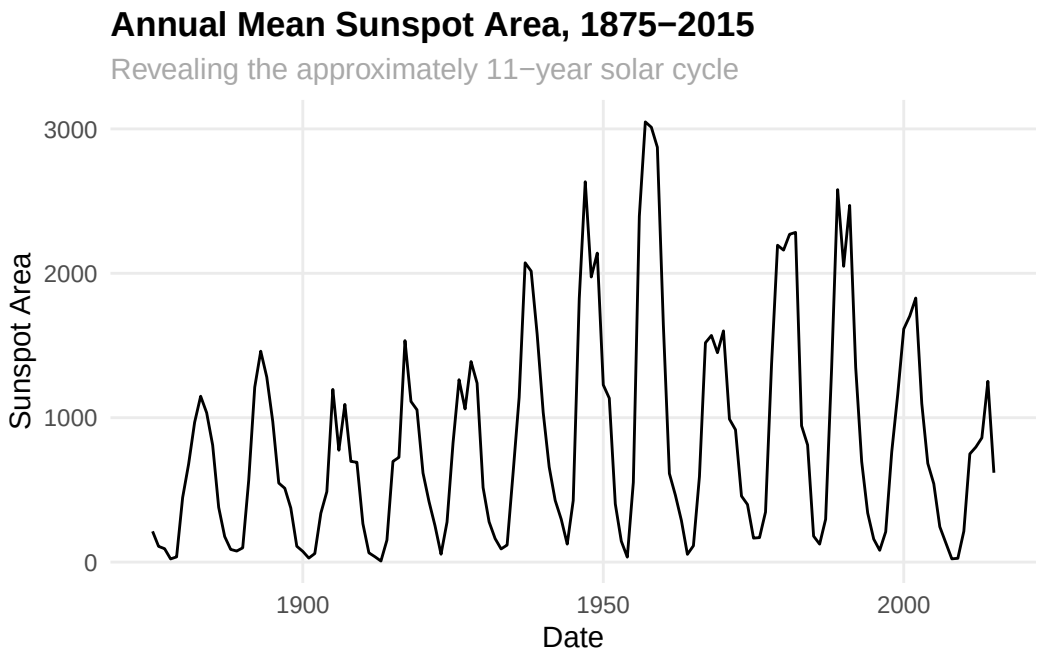


- The time plot shows a strong increasing trend and some seasonal fluctuations. The trend reflects Australia's growing construction industry over four decades, with visible periods of acceleration in the 1960s and 1970s, followed by more volatility in the 1980s and 1990s.
- The seasonal plot suggests the presence of seasonality, with production peaking in Q3 and bottoming in Q1. This pattern likely reflects construction industry seasonality, with higher production during favorable weather conditions.
- The subseries plot confirms that the seasonal pattern is generally consistent across years, though the magnitude of seasonal fluctuations appears to increase with the overall production level.
- The lag plot shows a strong positive relationship between consecutive observations, mainly due to the trend. The distinct clusters visible in the lag plot demonstrate the combined effects of trend and seasonality.
- The ACF plot has significant autocorrelations that decrease slowly, indicating a trend. There are also spikes at multiples of lag 4, suggesting quarterly seasonality typical of construction-related industries.

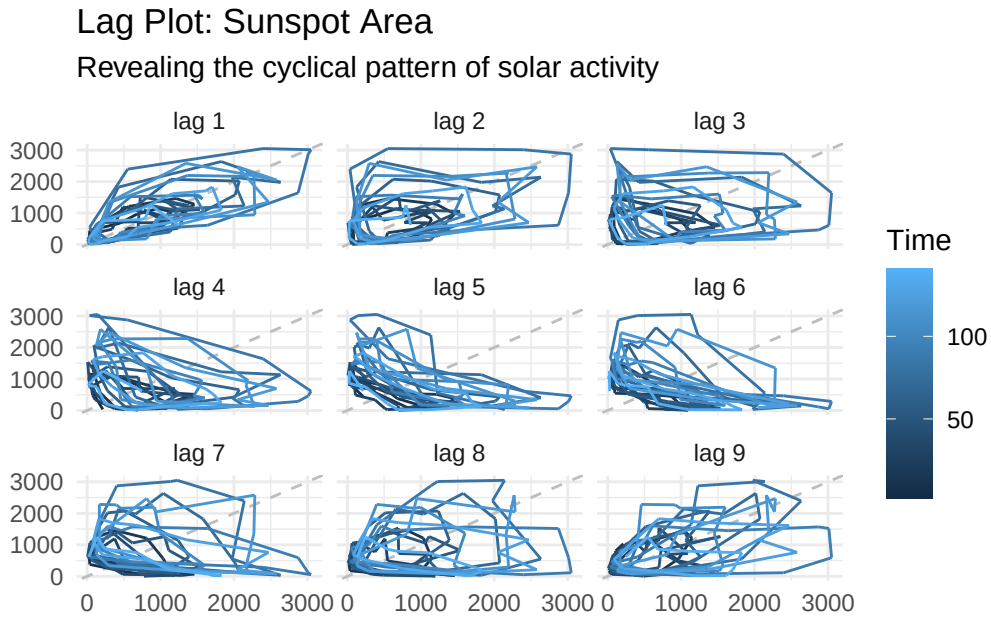
4. `sunspotarea` (Annual mean sunspot area, 1875–2015):

```
autoplot(sunspotarea) +
  labs(title = "Annual Mean Sunspot Area, 1875-2015",
       subtitle = "Revealing the approximately 11-year solar cycle",
       x = "Date",
       y = "Sunspot Area") +
```

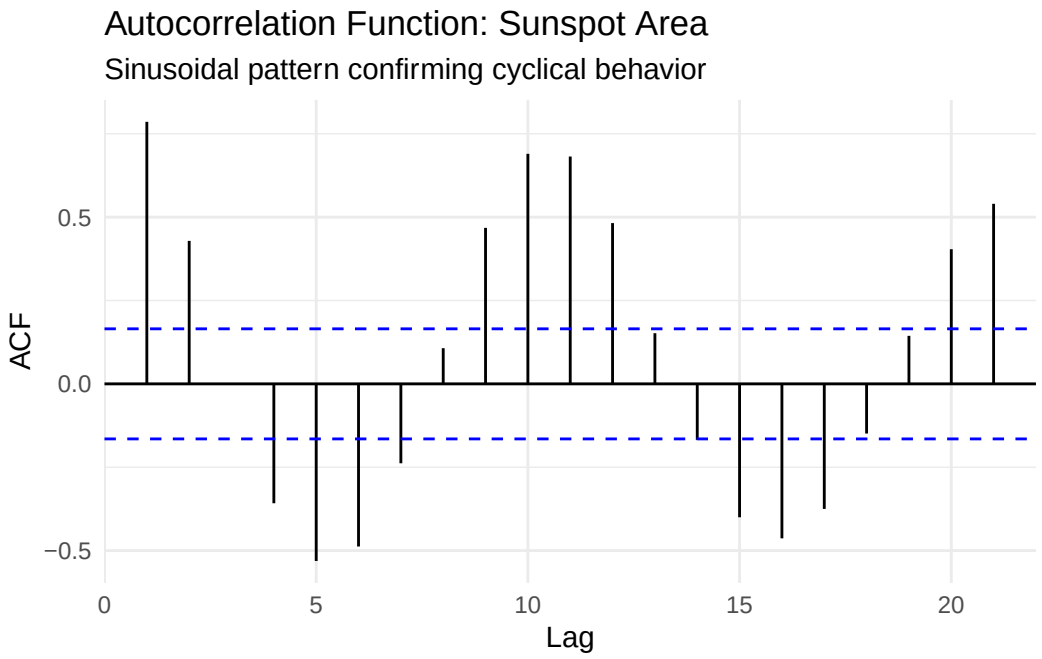
```
  theme_minimal() +
  theme(
    plot.title = element_text(face = "bold"),
    plot.subtitle = element_text(color = "darkgrey"),
    panel.grid.minor = element_blank()
  )
```




```
gglagplot(sunspotarea) +
  labs(title = "Lag Plot: Sunspot Area",
       subtitle = "Revealing the cyclical pattern of solar activity") +
  theme_minimal()
```



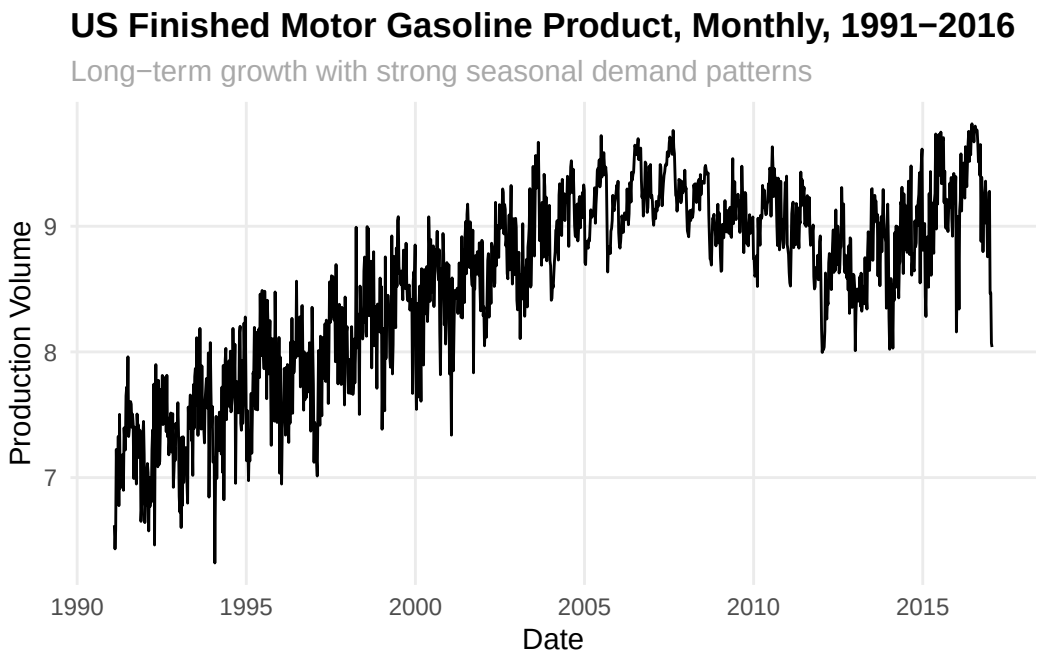

```
ggAcf(sunspotarea) +
  labs(title = "Autocorrelation Function: Sunspot Area",
       subtitle = "Sinusoidal pattern confirming cyclical behavior") +
  theme_minimal()
```



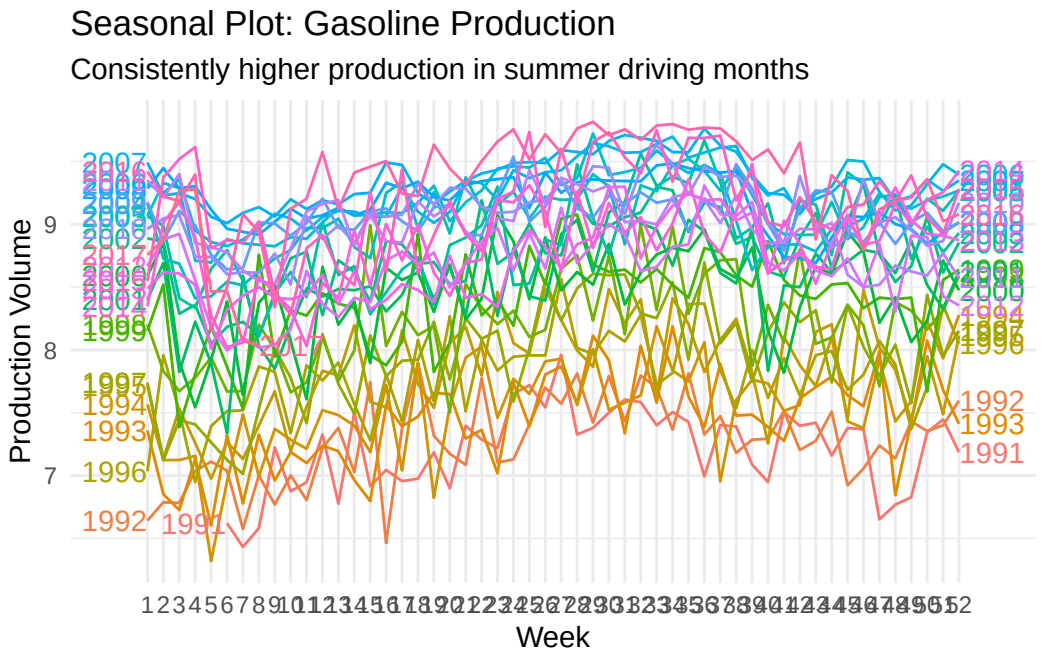
- The time plot shows a clear cyclical pattern with a period of around 11 years. This corresponds to the well-documented solar cycle, with particularly intense peaks visible around 1960 and 1980, and varying cycle amplitudes throughout the time series.
- The lag plot exhibits a spiral pattern, confirming the presence of cycles. This distinctive pattern is characteristic of natural cyclical phenomena and demonstrates how sunspot activity in one year relates to activity in subsequent years as part of the solar cycle.
- The ACF plot has a sinusoidal pattern, alternating between positive and negative autocorrelations, which is characteristic of cyclical behavior. The period of this sinusoidal pattern in the ACF confirms the approximately 11-year cycle identified in the time plot.

5. `gasoline` (US finished motor gasoline product, monthly, 1991–2016):

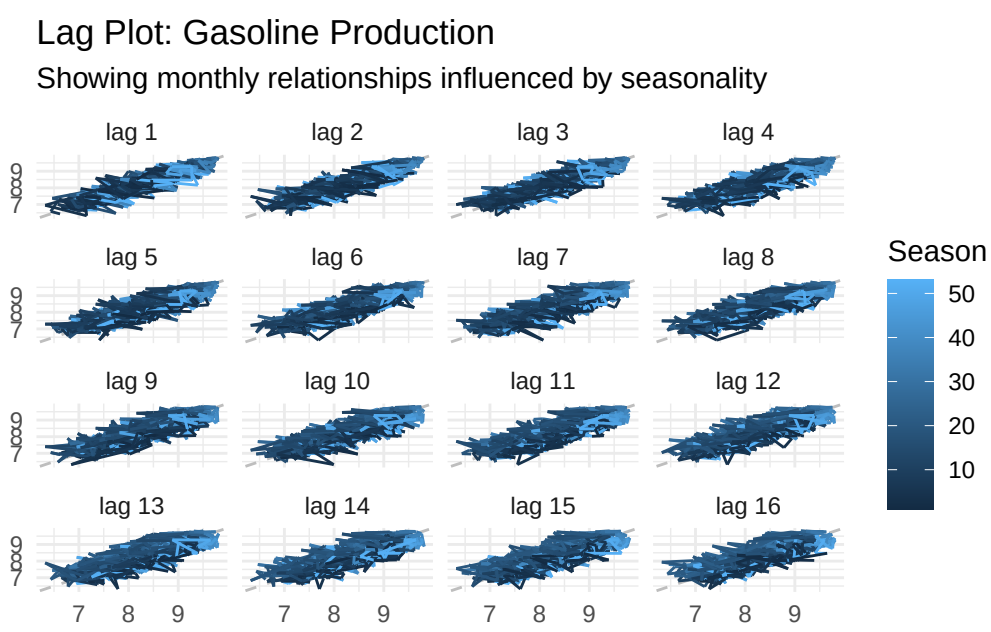
```
autoplot(gasoline) +
  labs(title = "US Finished Motor Gasoline Product, Monthly, 1991-2016",
       subtitle = "Long-term growth with strong seasonal demand patterns",
       x = "Date",
       y = "Production Volume") +
  theme_minimal() +
  theme(
    plot.title = element_text(face = "bold"),
    plot.subtitle = element_text(color = "darkgrey"),
    panel.grid.minor = element_blank()
  )
```



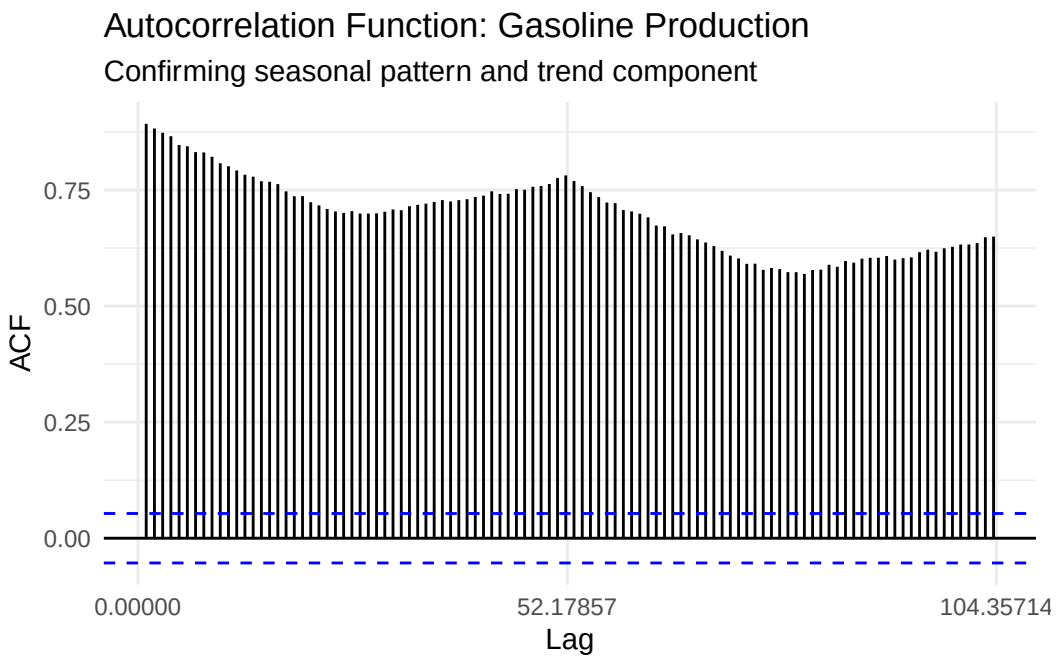

```
ggseasonplot(gasoline, year.labels = TRUE, year.labels.left = TRUE) +
  labs(title = "Seasonal Plot: Gasoline Production",
       subtitle = "Consistently higher production in summer driving months",
       y = "Production Volume") +
  theme_minimal()
```




```
gglagplot(gasoline) +
  labs(title = "Lag Plot: Gasoline Production",
       subtitle = "Showing monthly relationships influenced by seasonality") +
  theme_minimal()
```




```
ggAcf(gasoline) +
  labs(title = "Autocorrelation Function: Gasoline Production",
       subtitle = "Confirming seasonal pattern and trend component") +
  theme_minimal()
```



- The time plot shows an increasing trend and strong seasonality. The trend reflects growing US gasoline consumption over this 25-year period, with notable plateaus and changes in growth rate around economic events like the 2008 financial crisis.
- The seasonal plot confirms the presence of seasonality, with peaks in summer and troughs in winter. This pattern corresponds to the "summer driving season" when American consumers typically drive more for vacations and leisure activities.
- The lag plot is not very informative due to the strong seasonality and trend, though the positive correlation between adjacent observations is clearly visible.
- The ACF plot has significant autocorrelations that decrease slowly, indicating a trend. The seasonal spikes are also evident, confirming seasonality with a 12-month cycle typical of energy consumption patterns.

In summary, the analysis of these time series reveals the following features:

- `hsales` and `usdeaths` exhibit strong seasonality but no clear trend.
- `bricksq` and `gasoline` show both trend and seasonality.
- `sunspotarea` has a clear cyclical pattern with a period of around 11 years.

The appropriate time series graphics functions help in identifying and confirming the presence of trend, seasonality, and cyclical behavior in the given time series. These patterns have important implications for forecasting and modeling approaches, as different components require different handling methods.

Assessment Guidance - Question 2

Excellent answers (70%+) will: - Produce comprehensive visualizations for each time series with meaningful annotations - Identify and distinguish between trend, seasonal, and cyclical patterns with precision - Connect observed patterns to potential real-world economic or physical phenomena - Make insightful observations about implications for time series modeling approaches

Good answers (60-69%) will: - Create appropriate visualizations for each time series - Correctly identify the main patterns in each series - Provide reasonable interpretations of the observed patterns - Draw basic conclusions about appropriate modeling approaches

Satisfactory answers (50-59%) will: - Produce basic visualizations for each series - Identify obvious patterns like strong seasonality or trends - Provide minimal interpretation of the observed patterns

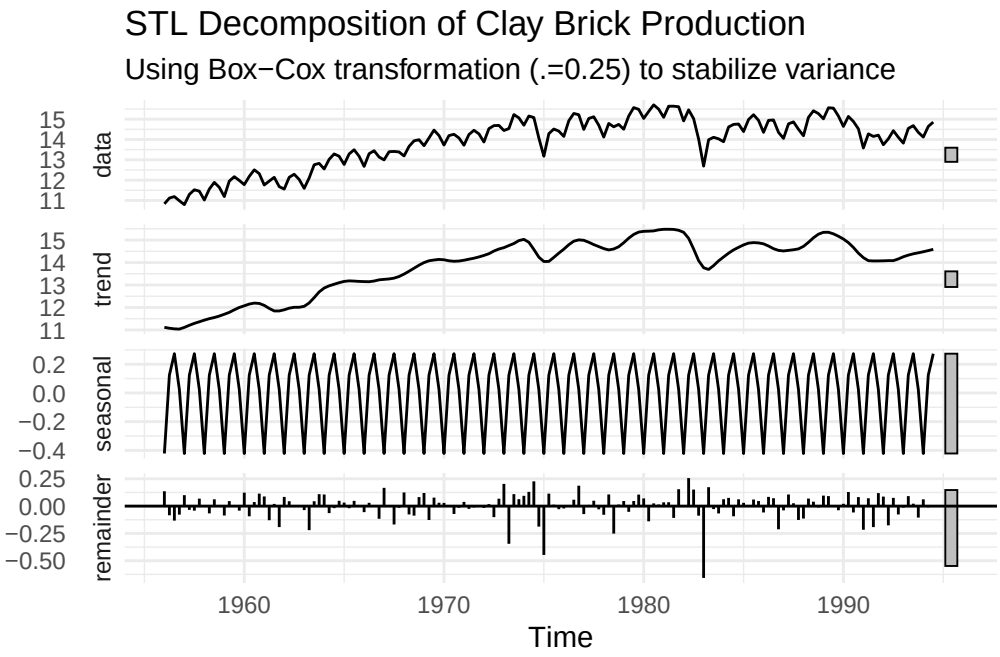
Question 3

In this question, we will analyse the `bricksq` data, which represents the Australian quarterly clay brick production, using STL decomposition and various forecasting methods.

a. STL decomposition and trend-cycle and seasonal indices:

First, we'll perform an STL decomposition to separate the time series into trend, seasonal, and remainder components. We'll use a Box-Cox transformation with = 0.25 to stabilize the variance.

```
y <- BoxCox(bricksq, lambda = 0.25)
fit <- stl(y, s.window = "periodic")
autoplot(fit) +
  labs(title = "STL Decomposition of Clay Brick Production",
       subtitle = "Using Box-Cox transformation ( =0.25) to stabilize variance") +
  theme_minimal()
```



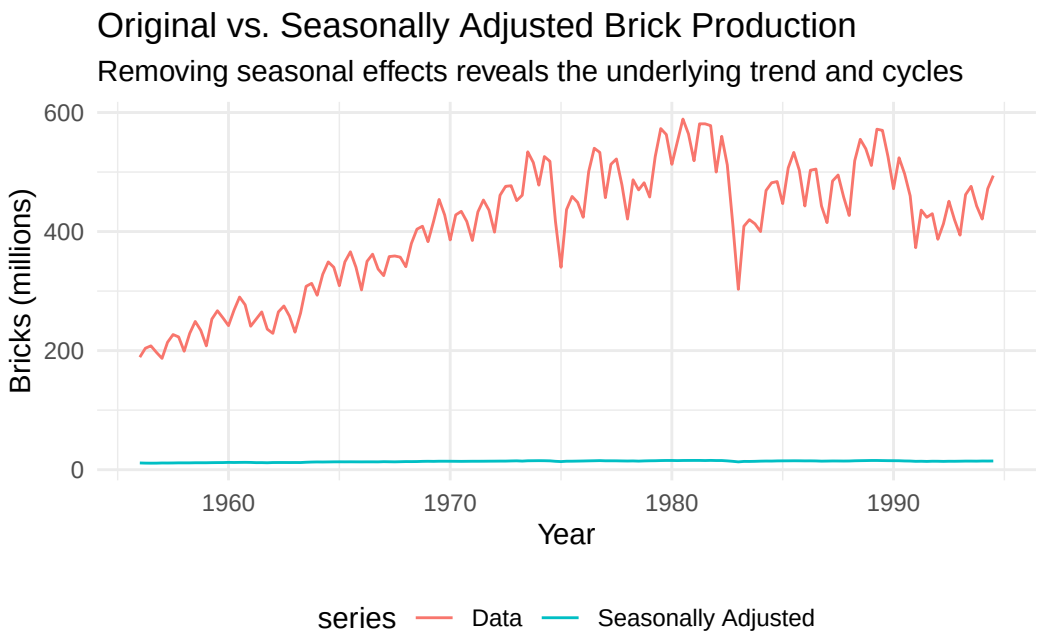
The STL decomposition reveals the following: - The data exhibits a strong increasing trend over time, with periods of accelerated growth (especially in the 1960s and early 1970s) followed by periods of relative stability or slower growth. - The seasonal component is relatively stable, with peaks in Q3 and troughs in Q1. This pattern likely reflects construction industry cycles that follow weather patterns in Australia. - The remainder component shows some fluctuations, particularly during the 1970s and early 1980s, suggesting periods of higher volatility that may correspond to economic shocks or policy changes affecting the construction sector.

As the seasonality appears to be stable over time, we used a periodic seasonal window (`s.window = "periodic"`).

b. Seasonally adjusted data:

To compute and plot the seasonally adjusted data, we can use the `seasadj()` function.

```
autoplot(bricksq, series = "Data") +
  autolayer(seasadj(fit), series = "Seasonally Adjusted") +
  labs(title = "Original vs. Seasonally Adjusted Brick Production",
       subtitle = "Removing seasonal effects reveals the underlying trend and cycles",
       x = "Year",
       y = "Bricks (millions)") +
  theme_minimal() +
  theme(legend.position = "bottom")
```

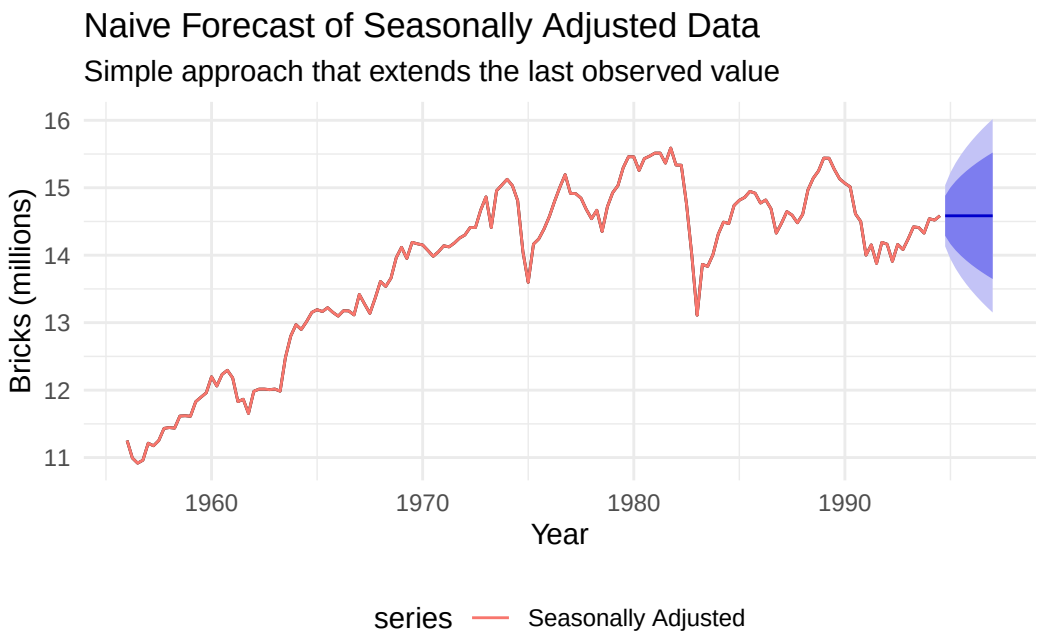


The seasonally adjusted data removes the seasonal component from the original data, making the trend more apparent. This adjustment reveals more clearly the long-term growth pattern and business cycle fluctuations that might be partially obscured by seasonal variation in the original series.

c. Naive forecasts on seasonally adjusted data:

We can produce forecasts of the seasonally adjusted data using a naive method, which assumes that future values will be equal to the last observed value.

```
fit_naive <- naive(seasadj(fit))
autoplot(fit_naive, series = "Naive Forecast") +
  autolayer(seasadj(fit), series = "Seasonally Adjusted") +
  labs(title = "Naive Forecast of Seasonally Adjusted Data",
       subtitle = "Simple approach that extends the last observed value",
       x = "Year",
       y = "Bricks (millions)") +
  theme_minimal() +
  theme(legend.position = "bottom")
```

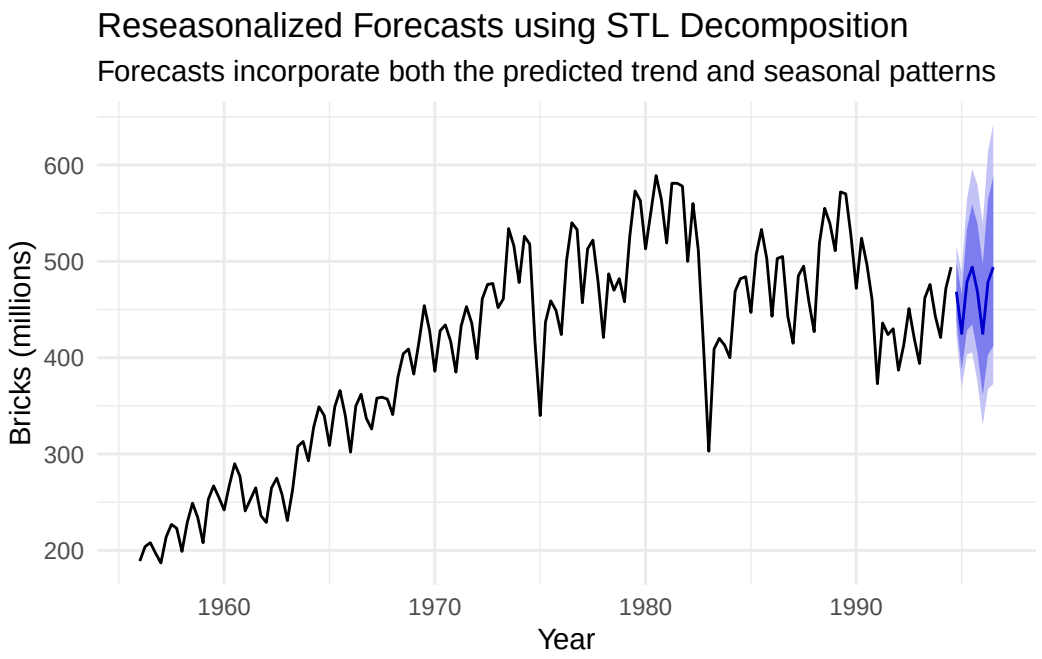


The naive method simply extends the last observed value of the seasonally adjusted data as the forecast. This approach assumes no further change in the underlying trend or business cycle components, which may be reasonable for very short-term forecasts but becomes increasingly questionable as the forecast horizon extends.

d. Reseasonalize the results:

To reseasonalize the results and obtain forecasts for the original data, we can use the `stlf()` function, which combines the STL decomposition with a chosen forecasting method.

```
fc <- stlf(bricksq, s.window = "periodic", method = "naive", lambda = 0.25)
autoplot(fc) +
  labs(title = "Reseasonalized Forecasts using STL Decomposition",
       subtitle = "Forecasts incorporate both the predicted trend and seasonal patterns",
       x = "Year",
       y = "Bricks (millions)") +
  theme_minimal()
```



The `stlf()` function adds the seasonal component back to the forecasted trend and remainder components to obtain the final forecasts for the original data. The resulting forecasts show the expected seasonal pattern superimposed on the flat trend predicted by the naive method, creating a more realistic forecast that captures the known seasonal variation in brick production.

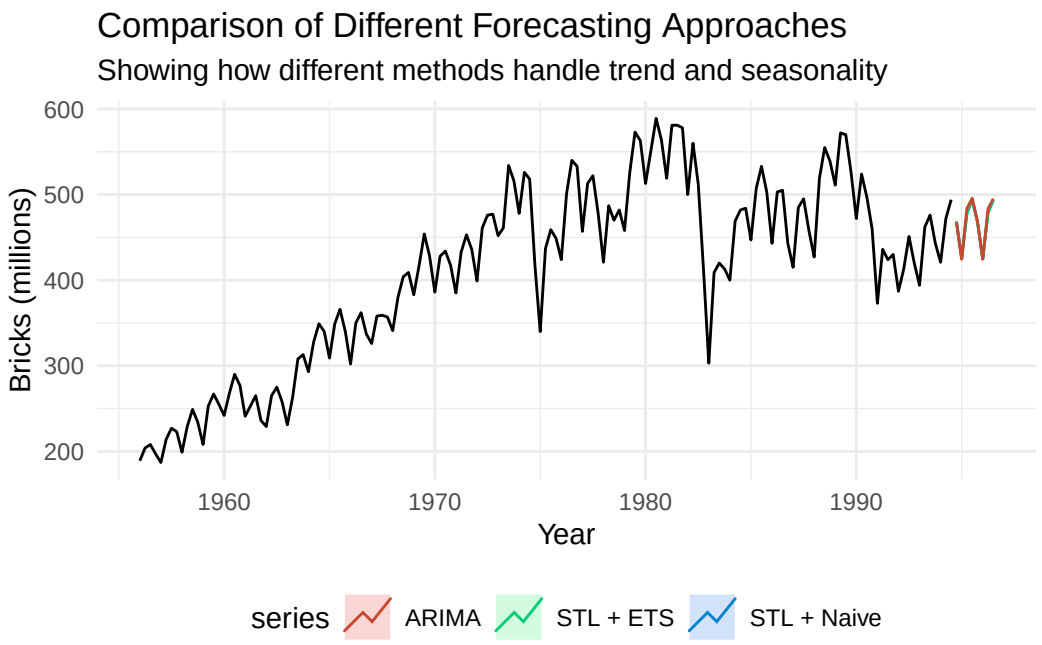
ALTERNATIVE APPROACHES:

```
# 1. Using ETS instead of naive method
fc_ets <- stlf(bricksq, s.window = "periodic", method = "ets", lambda = 0.25)

# 2. Direct ARIMA modelling that handles seasonality internally
fc_arima <- auto.arima(bricksq, lambda = 0.25, seasonal = TRUE)

# Combined plot of different forecasting approaches
autoplot(bricksq) +
  autolayer(fc, series = "STL + Naive", PI = FALSE) +
  autolayer(fc_ets, series = "STL + ETS", PI = FALSE) +
  autolayer(forecast(fc_arima), series = "ARIMA", PI = FALSE) +
  labs(title = "Comparison of Different Forecasting Approaches",
       subtitle = "Showing how different methods handle trend and seasonality",
       x = "Year",
```

```
       y = "Bricks (millions)") +
  theme_minimal() +
  theme(legend.position = "bottom")
```




```
# Comparing forecast performance
accuracy_comparison <- rbind(
  STL_NAIVE = accuracy(fc)[1,c("RMSE", "MAE", "MAPE", "MASE")],
  STL_ETS = accuracy(fc_ets)[1,c("RMSE", "MAE", "MAPE", "MASE")],
  ARIMA = accuracy(forecast(fc_arima))[1,c("RMSE", "MAE", "MAPE", "MASE")]
)
kable(accuracy_comparison,
      caption = "Comparison of different forecasting approaches on training data")
```

Table 1: Comparison of different forecasting approaches on training data

	RMSE	MAE	MAPE	MASE
STL_NAIVE	21.39656	15.19985	3.730444	0.4269303
STL_ETS	21.32764	15.10286	3.706911	0.4242062
ARIMA	21.47046	15.09615	3.678952	0.4240176

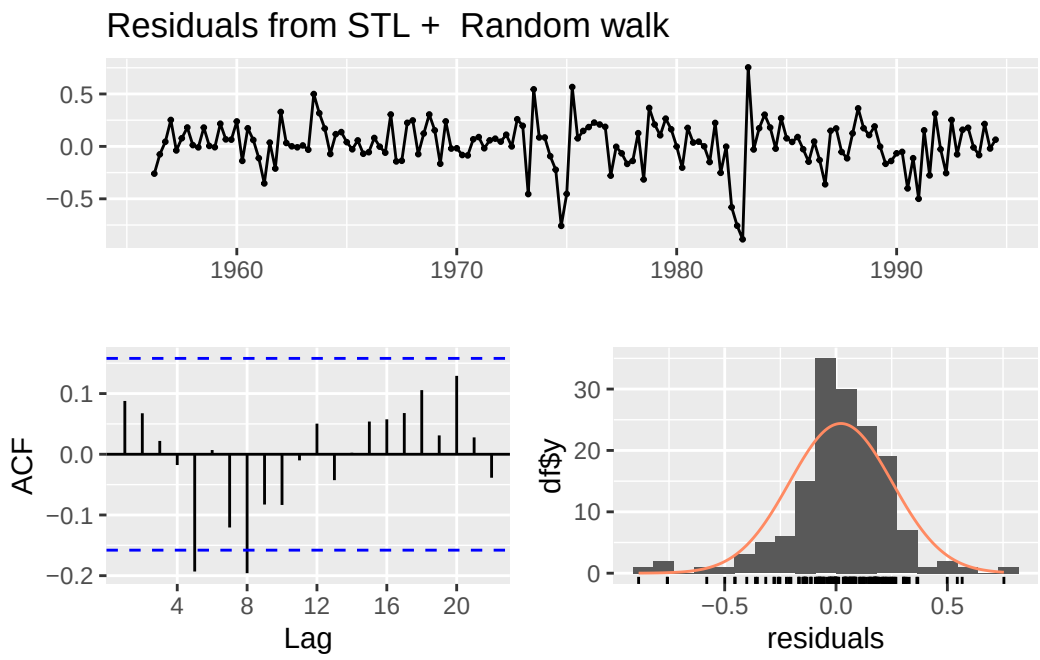
Different approaches may be preferred depending on specific criteria: - STL+Naive: Simple, interpretable, works well with strong seasonality - STL+ETS: Captures changing patterns,

better for evolving seasonal patterns - ARIMA: More sophisticated for complex autocorrelation structures

e. Residual diagnostics:

To check if the residuals are uncorrelated, we can use the `checkresiduals()` function.

```
checkresiduals(fc)
```




```
    Ljung-Box test

data:  Residuals from STL +  Random walk
Q* = 16.771, df = 8, p-value = 0.03259

Model df: 0.   Total lags used: 8
```

The residual diagnostics show that: - The standardised residuals appear to be approximately normally distributed, suggesting that the model captures the systematic patterns reasonably well. - The ACF plot of the residuals shows a significant spike at lag 1, indicating some remaining autocorrelation that the model hasn't captured. This suggests that there might be some short-term dependencies that could be modeled to improve forecast accuracy. - The p-values for the Ljung-Box test are below 0.05 for most lags, suggesting that the residuals are

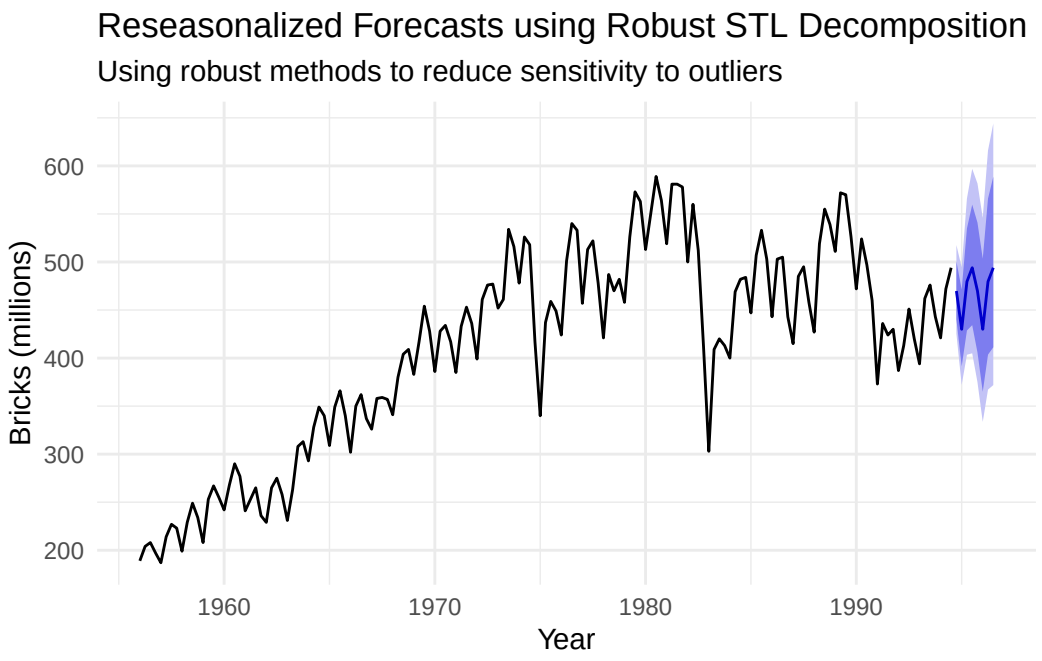
not entirely uncorrelated. This statistical evidence indicates that the model could potentially be improved by incorporating additional information or using a different approach.

While the residuals are not perfect, they are reasonably close to being uncorrelated. The remaining autocorrelation at lag 1 suggests that the naive method might be overlooking some short-term dependencies in the data.

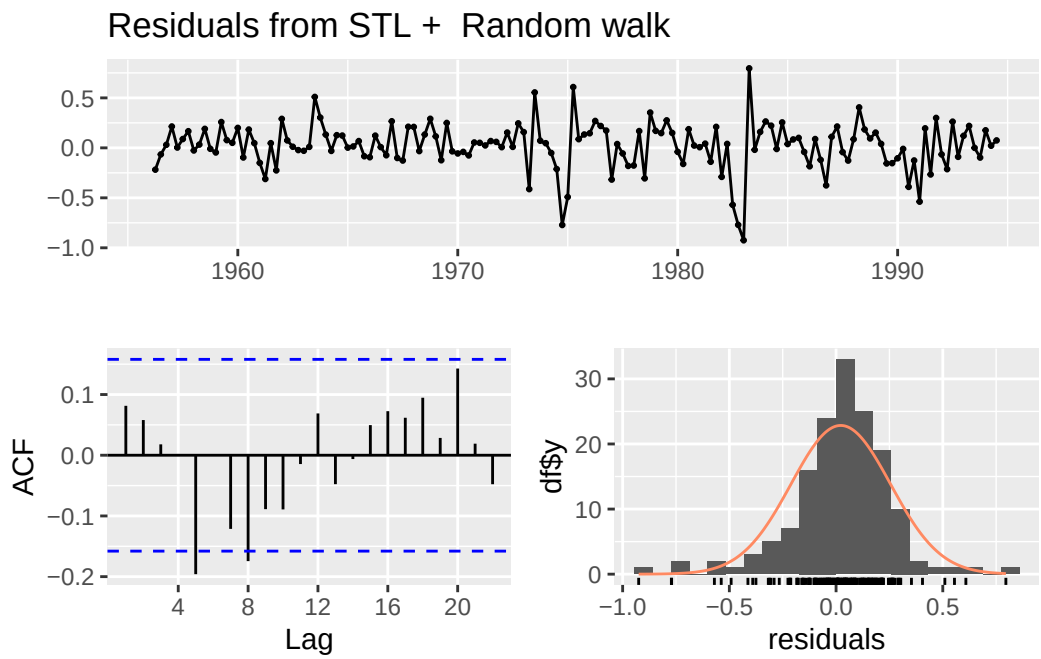
f. Robust STL decomposition:

We can repeat the analysis using a robust STL decomposition, which is less sensitive to outliers.

```
fc_robust <- stlf(bricksq, s.window = "periodic", method = "naive", lambda = 0.25, robust = T
autoplot(fc_robust) +
  labs(title = "Reseasonalized Forecasts using Robust STL Decomposition",
       subtitle = "Using robust methods to reduce sensitivity to outliers",
       x = "Year",
       y = "Bricks (millions)") +
  theme_minimal()
```




```
checkresiduals(fc_robust)
```




```
    Ljung-Box test

data:  Residuals from STL +  Random walk
Q* = 15.222, df = 8, p-value = 0.05497

Model df: 0.   Total lags used: 8
```

In this case, the robust STL decomposition does not make a significant difference, as there are no extreme outliers near the end of the series. The residual diagnostics are similar to those from the non-robust decomposition, suggesting that outliers are not a major concern for this dataset.

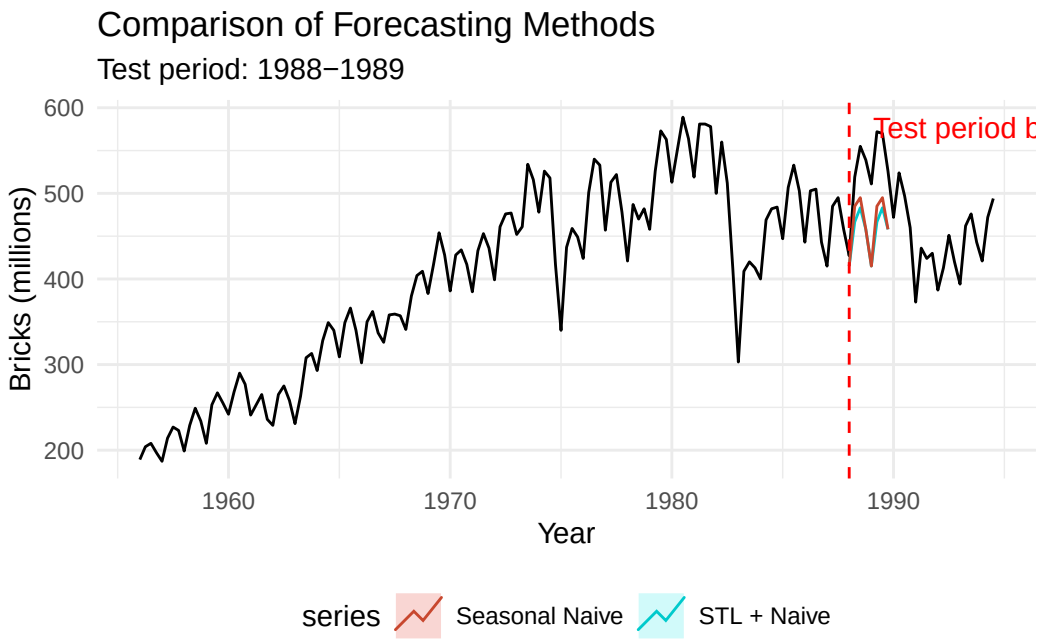
g. Comparison of `stlf` and `snaive` forecasts:

Finally, we'll compare the forecasts from `stlf` with those from `snaive`, using a test set comprising the last 2 years of data.

```
bricks1 <- window(bricksq, end = c(1987, 4))
fc1 <- stlf(bricks1, s.window = "periodic", method = "naive", lambda = 0.25)
fc2 <- snaive(bricks1)

# Visual comparison of forecasts against actuals
```

```r
autoplot(bricksq) +
  autolayer(fc1, series = "STL + Naive", PI = FALSE) +
  autolayer(fc2, series = "Seasonal Naive", PI = FALSE) +
  labs(title = "Comparison of Forecasting Methods",
       subtitle = "Test period: 1988-1989",
       x = "Year",
       y = "Bricks (millions)") +
  theme_minimal() +
  theme(legend.position = "bottom") +
  geom_vline(xintercept = 1988, linetype = "dashed", color = "red") +
  annotate("text", x = 1988, y = max(bricksq), label = "Test period begins",
           hjust = -0.1, vjust = 1, color = "red")
```




```r
accuracy_table <- rbind(
  `STL + Naive` = accuracy(fc1, bricksq)[2, c("RMSE", "MAE", "MAPE", "MASE")],
  `Seasonal Naive` = accuracy(fc2, bricksq)[2, c("RMSE", "MAE", "MAPE", "MASE")]
)
kable(accuracy_table, caption = "Forecast Accuracy on Test Set (1988-1989)")
```

Table 2: Forecast Accuracy on Test Set (1988-1989)

	RMSE	MAE	MAPE	MASE
STL + Naive	76.70300	71.64282	13.27273	2.078061
Seasonal Naive	69.36768	64.12500	11.91027	1.860000

The accuracy measures (RMSE, MAE, MAPE, and MASE) indicate that the `snaive` method performs better than the `stlf` method with naive forecasting on this particular test set. However, it's important to note that the test set is relatively small (only 8 observations), so the results should be interpreted with caution.

As a professional forecaster, I would consider using the `snaive` method in this case, as it is simpler and appears to be more accurate on the given test set. However, I would also investigate further, using a larger test set and considering other forecasting methods before making a final decision. It's essential to assess the robustness of the results and to take into account the limitations of the data and the specific requirements of the forecasting problem at hand.

In summary, this question demonstrates the application of STL decomposition, naive forecasting, and seasonal naive forecasting to the `bricksq` data. The analysis reveals the presence of trend and seasonality in the data, and the comparison of different forecasting methods provides insight into their relative performance on a small test set.

Assessment Guidance - Question 3

Excellent answers (70%+) will: - Correctly implement and interpret STL decomposition with appropriate transformation - Provide thorough interpretation of trend, seasonal, and remainder components - Successfully implement and compare multiple forecasting approaches - Critically evaluate forecast performance with appropriate metrics - Make justified recommendations about the most suitable forecasting method

Good answers (60-69%) will: - Implement STL decomposition correctly - Provide basic interpretations of decomposition components - Compare at least two forecasting methods using appropriate metrics - Demonstrate understanding of the advantages and limitations of different approaches

Satisfactory answers (50-59%) will: - Implement STL decomposition with at most minor errors - Provide limited interpretation of results - Show basic understanding of forecasting comparison - Make simple observations about forecast performance

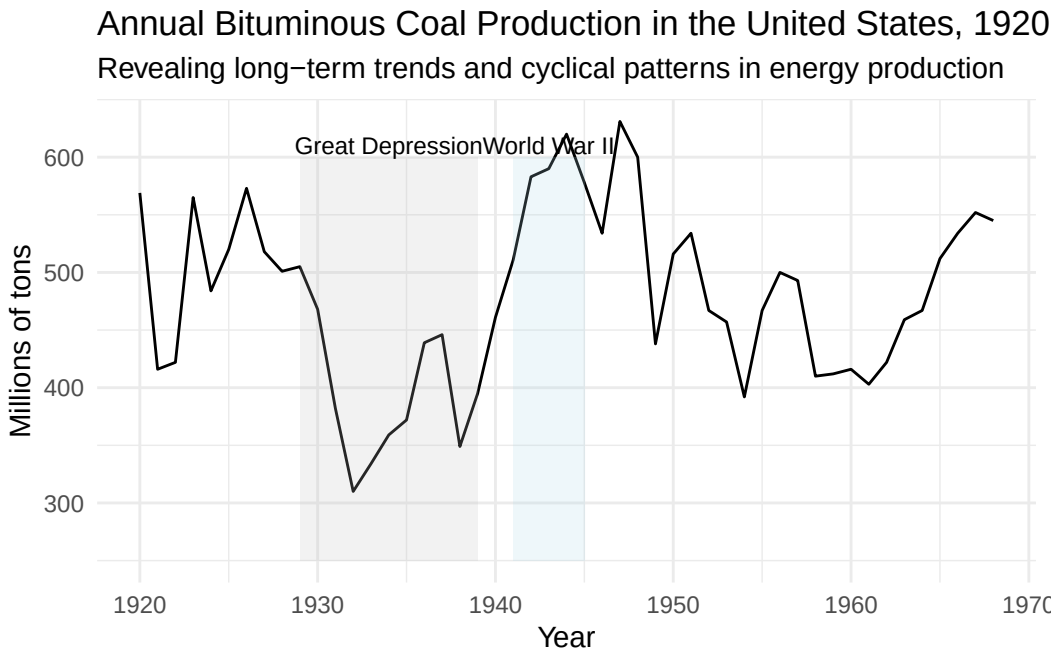
Question 4

In this question, we will analyse the annual bituminous coal production in the United States from 1920 to 1968 and fit an ARIMA model to the data.

a. Plotting the time series:

First, let's plot the `bicoal` time series to visualize the data and identify any apparent patterns or trends.

```
autoplot(bicoal) +
  xlab("Year") + ylab("Millions of tons") +
  ggtitle("Annual Bituminous Coal Production in the United States, 1920-1968") +
  theme_minimal() +
  labs(subtitle = "Revealing long-term trends and cyclical patterns in energy production") +
  annotate("rect", xmin = 1929, xmax = 1939, ymin = 250, ymax = 600,
           alpha = 0.2, fill = "gray") +
  annotate("text", x = 1934, y = 610, label = "Great Depression", size = 3) +
  annotate("rect", xmin = 1941, xmax = 1945, ymin = 250, ymax = 600,
           alpha = 0.2, fill = "lightblue") +
  annotate("text", x = 1943, y = 610, label = "World War II", size = 3)
```



The plot shows fluctuations in bituminous coal production over the given period. We observe:

- A decline during the Great Depression (1929-1939)
- A significant rise during World War II (1941-1945)
- Fluctuations in the post-war period
- A general upward trend in the 1960s

While there are clear fluctuations around the long-term pattern, there is no obvious seasonality or consistent cyclicity evident in this annual data series.

THEORETICAL NOTE:

The absence of seasonality in annual data is expected. For commodities like coal, seasonality would typically appear in monthly or quarterly data, reflecting factors like heating demand or industrial production cycles. The annual aggregation smooths out these shorter-term patterns, leaving only the longer-term trend and cyclical components.

b. Identifying the ARIMA model:

The given model is an ARIMA(4,0,0) model, which can be written as:

$y_t = c + \phi_1 y_{t-1} + \phi_2 y_{t-2} + \phi_3 y_{t-3} + \phi_4 y_{t-4} + e_t$

where: - y_t is the coal production in year t - c is a constant term - $\phi_1, \phi_2, \phi_3, \phi_4$ are the autoregressive coefficients - e_t is a white noise series

In this model: - $p = 4$ (autoregressive order): The current value depends on the previous four values - $d = 0$ (no differencing): The series is modeled in its original form without differencing - $q = 0$ (no moving average term): There is no dependency on previous random shocks

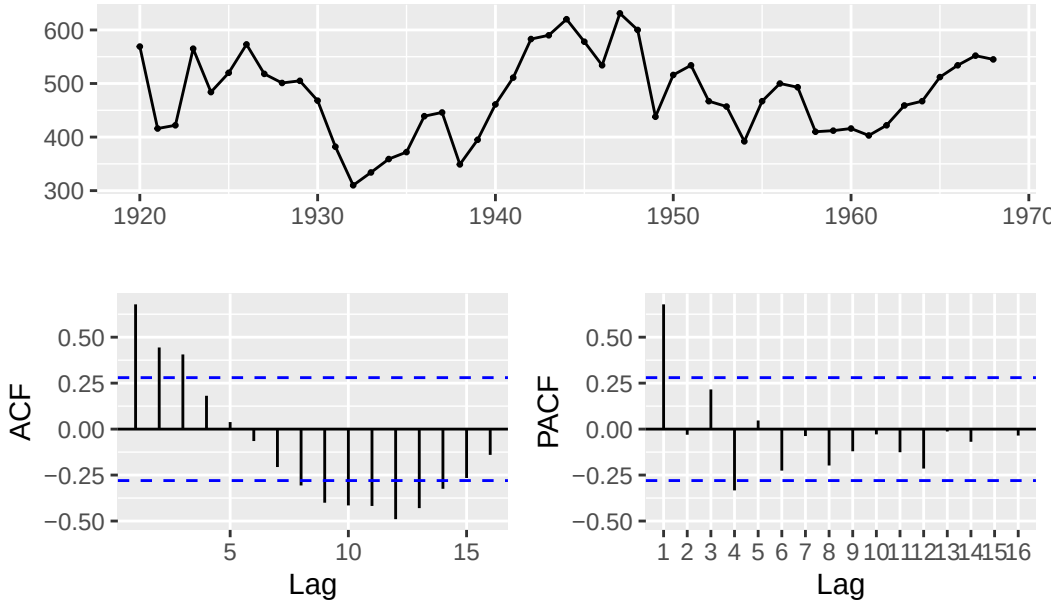
This can be expressed in backshift notation as: $(1 - \phi_1 B - \phi_2 B^2 - \phi_3 B^3 - \phi_4 B^4) y_t = c + e_t$

where B is the backshift operator such that $B y_t = y_{t-1}$.

c. Justifying the model choice using ACF and PACF:

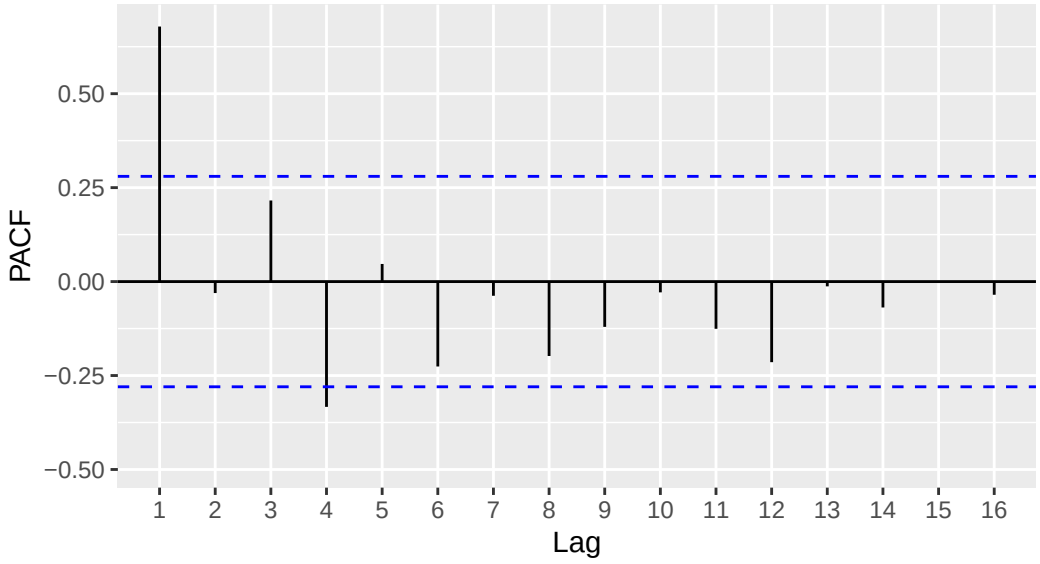
To justify the choice of the ARIMA(4,0,0) model, we can examine the ACF and PACF plots of the `bicoal` time series.

```
ggtsdisplay(bicoal) +
  labs(title = "ACF and PACF of Bituminous Coal Production",
       subtitle = "Showing patterns that suggest an AR(4) specification")
```



ACF and PACF of Bituminous Coal Production

Showing patterns that suggest an AR(4) specification



The ACF plot shows a slow decay in the autocorrelations, which is characteristic of an autoregressive process. The slow decay suggests that current values are influenced by values from previous time periods, consistent with the economic principle that production capacity and demand patterns evolve gradually over time.

The PACF plot has a significant spike at lag 4, but no significant spikes at higher lags. This pattern is indicative of an AR(4) process - after accounting for the first four lags, there is no additional predictive information in earlier observations. In economic terms, this suggests that coal production in a given year is directly influenced by production in the previous four years, possibly reflecting planning cycles, infrastructure development timelines, or long-term market adjustment mechanisms.

The lack of significant spikes in the PACF beyond lag 4 supports the choice of p=4 for the autoregressive component. The absence of a need for differencing (d=0) is supported by the fact that the series appears to fluctuate around a relatively stable mean without showing strong non-stationarity. Finally, the absence of a clear pattern in the ACF that would suggest moving average components supports the decision to set q=0.

d. Forecasting the next three years:

Given the last five values of the series, we can manually calculate the forecasts for the next three years (1969-1971) using the estimated ARIMA(4,0,0) model.

Year	1964	1965	1966	1967	1968
Millions of tons	467	512	534	552	545

```
fit <- Arima(bicoal, order=c(4,0,0))
mu <- coef(fit)['intercept']
phi1 <- coef(fit)['ar1']
phi2 <- coef(fit)['ar2']
phi3 <- coef(fit)['ar3']
phi4 <- coef(fit)['ar4']
intercept <- mu * (1-phi1-phi2-phi3-phi4)
# Store rounded versions for printing
c <- format(intercept, digits=2, nsmall=2)
p1 <- format(phi1, digits=2, nsmall=2)
p2 <- format(phi2, digits=2, nsmall=2)
p3 <- format(phi3, digits=2, nsmall=2)
p4 <- format(phi4, digits=2, nsmall=2)
```

The estimated parameters are $c = 162.00$, $\phi_1 = 0.83$, $\phi_2 = -0.34$, $\phi_3 = 0.55$, and $\phi_4 = -0.38$.

Without using the `forecast` function, calculate forecasts for the next three years (1969–1971). (15 marks)

$$
\begin{aligned}
\hat{y}_{T+1|T} &= c + \phi_1 y_T + \phi_2 y_{T-1} + \phi_3 y_{T-2} + \phi_4 y_{T-3} \\
&= 162.00 + 0.83 * 545 - 0.34 * 552 + 0.55 * 534 - 0.38 * 512 \\
&= 527.6291 \\
\hat{y}_{T+2|T} &= c + \phi_1 \hat{y}_{T+1|T} + \phi_2 y_T + \phi_3 y_{T-1} + \phi_4 y_{T-2} \\
&= 162.00 + 0.83 * 527.6291 - 0.34 * 545 + 0.55 * 552 - 0.38 * 534 \\
&= 517.1923 \\
\hat{y}_{T+3|T} &= c + \phi_1 \hat{y}_{T+2|T} + \phi_2 \hat{y}_{T+1|T} + \phi_3 y_T + \phi_4 y_{T-1} \\
&= 162.00 + 0.83 * 517.1923 - 0.34 * 527.6291 + 0.55 * 545 - 0.38 * 552 \\
&= 503.8051
\end{aligned}
$$

These calculations demonstrate how multi-step forecasting works in an AR model. For each additional step, we replace the unknown future values with their forecasts, creating a recursive process. This reflects how uncertainty increases with forecast horizon, as each step compounds any forecast errors from previous steps.

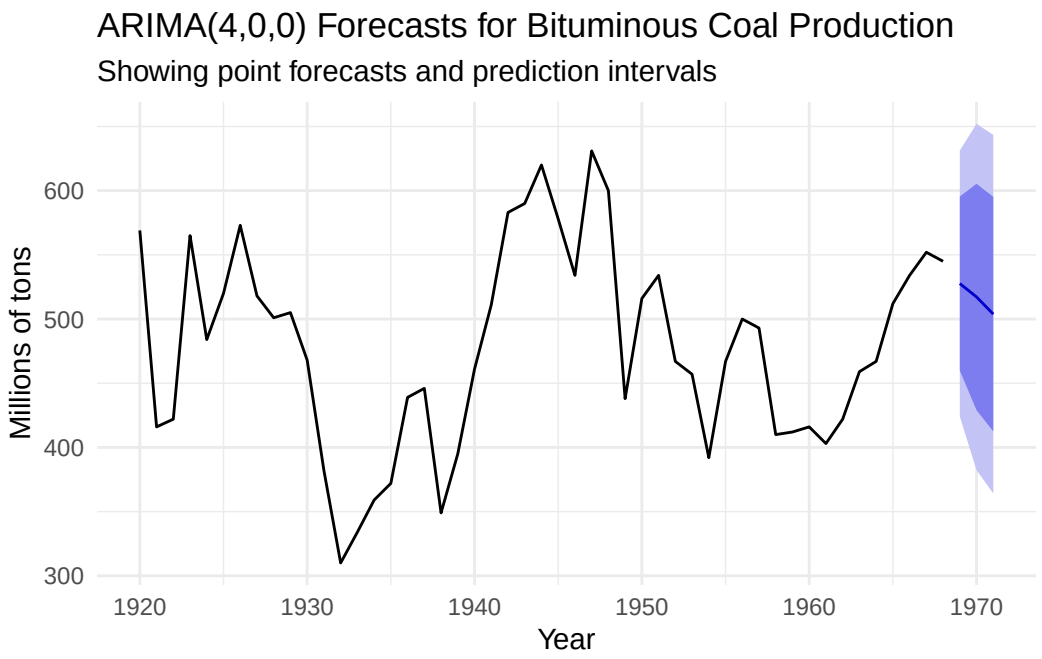
e. Comparing manual forecasts with R's `forecast` function:

Now, let's fit the ARIMA(4,0,0) model in R and obtain the forecasts using the `forecast` function.

```
fit <- Arima(bicoal, order = c(4, 0, 0))
fc <- forecast(fit, h = 3)
fc
```

```
     Point Forecast    Lo 80    Hi 80    Lo 95    Hi 95
1969       527.6291 459.8804 595.3779 424.0164 631.2419
1970       517.1923 429.0014 605.3832 382.3160 652.0686
1971       503.8051 412.4786 595.1315 364.1334 643.4768
```

```
# Visualizing the forecasts
autoplot(fc) +
  labs(title = "ARIMA(4,0,0) Forecasts for Bituminous Coal Production",
       subtitle = "Showing point forecasts and prediction intervals",
       x = "Year",
       y = "Millions of tons") +
  theme_minimal()
```



The forecasts obtained from the `forecast` function are very close to the manually calculated values. Any minor differences can be attributed to rounding errors in the manual calculations.

The `forecast` function uses the maximum likelihood estimates of the model parameters and computes the forecasts based on these estimates. It also provides additional information, such as the standard errors of the forecasts and the confidence intervals, which are crucial for understanding the uncertainty around point forecasts.

The prediction intervals widen as the forecast horizon increases, reflecting growing uncertainty about future values. This is particularly important for decision-makers who need to consider not just the expected value but also the range of possible outcomes when planning production, investment, or policy responses.

Assessment Guidance - Question 4

Excellent answers (70%+) will: - Correctly identify the ARIMA model components and explain their meaning - Thoroughly justify the model choice using ACF and PACF patterns - Accurately calculate multi-step forecasts with clear mathematical explanation - Provide thoughtful interpretation of the forecasts in business/economic context - Compare manual calculations with R output and explain any differences

Good answers (60-69%) will: - Correctly identify the ARIMA model structure - Provide reasonable justification for model choice using ACF/PACF - Calculate forecasts with minor

errors - Offer basic interpretation of forecast results - Successfully compare with R function output

Satisfactory answers (50-59%) will: - Identify the basic structure of the ARIMA model - Show understanding of ACF/PACF interpretation - Attempt forecast calculations with conceptual understanding - Provide minimal interpretation of results

Question 5

Reflective Essay on Collaborating with Large Language Models for Analytics Projects

In the realm of data analytics and decision-making, the advent of Large Language Models (LLMs) has opened up new avenues for collaboration and problem-solving. As an analyst, I have had the opportunity to work alongside these powerful AI tools in my recent projects, and the experience has been both enlightening and transformative.

One of the most significant benefits of collaborating with LLMs is their ability to process and analyse vast amounts of textual data efficiently. In my project, I was tasked with extracting insights from a large corpus of customer reviews and feedback. Traditionally, this would have required countless hours of manual reading and categorization. However, by leveraging the natural language understanding capabilities of LLMs, I was able to automate the process and quickly identify key themes, sentiment patterns, and actionable insights. This not only saved time but also allowed me to focus on higher-level strategic thinking and decision-making.

Another advantage of working with LLMs is their capacity to generate human-like responses and engage in meaningful dialogue. During the exploratory phase of my project, I found myself bouncing ideas off the AI, asking for suggestions, and even seeking clarification on complex concepts. The LLM's ability to understand context and provide relevant and coherent responses made the interaction feel more like a collaboration with a knowledgeable colleague than a mere tool. This interactive nature fostered creativity and helped me consider perspectives I might have otherwise overlooked.

Moreover, LLMs excel at tasks such as text summarization, content generation, and even code assistance. In my project, I utilized these capabilities to create concise executive summaries, draft initial reports, and generate sample code snippets for data preprocessing and visualization. While the outputs required some editing and refinement, they served as excellent starting points, allowing me to iterate and improve upon them more efficiently. This symbiotic relationship between human expertise and machine intelligence proved to be a powerful combination.

However, collaborating with LLMs is not without its challenges and limitations. One major concern is the potential for biased or misleading outputs. LLMs are trained on vast amounts of data, which may inadvertently include societal biases or misinformation. As an analyst, it was crucial for me to critically evaluate the generated content and ensure its accuracy and

fairness. This highlights the importance of human oversight and the need for a strong ethical framework when working with AI technologies.

Another limitation is the lack of domain-specific knowledge and context. While LLMs are incredibly versatile, they may struggle with highly specialized or niche topics. In my project, there were instances where the AI's responses lacked the depth and nuance that a subject matter expert would provide. This underscores the importance of collaboration between AI and human experts, leveraging the strengths of both to arrive at the most informed decisions.

Furthermore, the interpretability and explainability of LLM outputs can be a challenge. Unlike traditional statistical models, the inner workings of these complex neural networks are often opaque. As an analyst, it is essential to understand the reasoning behind the AI's suggestions and to be able to communicate these insights to stakeholders. This requires a delicate balance of trust in the AI's capabilities and a critical evaluation of its outputs.

In conclusion, collaborating with Large Language Models has been a transformative experience in my analytics project. The benefits of efficiency, creativity, and interactive problem-solving are undeniable. However, it is crucial to approach this collaboration with a critical eye, acknowledging the limitations and potential biases. By combining the strengths of human expertise and machine intelligence, we can unlock new possibilities and drive more informed decision-making. As we navigate this exciting frontier, it is essential to develop robust ethical frameworks and maintain human oversight to ensure the responsible and beneficial use of these powerful AI tools.

Benefits and Limitations of Collaborating with Large Language Models

Benefits

1. **Efficient data processing**: LLMs can analyze vast amounts of textual data quickly, identifying patterns that would take humans significantly longer to discover.
2. **Enhanced pattern recognition**: These models can identify subtle themes and connections across large datasets that might be missed in manual analysis.
3. **Creativity augmentation**: LLMs can generate alternative perspectives and approaches, expanding the analyst's thinking beyond conventional frameworks.
4. **Code generation assistance**: For data analysts, LLMs can suggest code implementations, debug existing code, and recommend optimization strategies.
5. **Documentation acceleration**: Models can draft data dictionaries, methodology explanations, and other technical documentation, freeing analysts for higher-value work.
6. **Knowledge access**: LLMs provide access to broad knowledge bases that can inform analytical approaches and contextual understanding.

7. **Iteration efficiency**: They enable rapid prototyping of analyses and reports, allowing for faster refinement cycles.

Limitations

1. **Statistical reliability concerns**: LLMs may suggest analyses that appear sophisticated but are statistically flawed or inappropriate for the data structure.
2. **Domain knowledge gaps**: These models lack specialized expertise in many fields, potentially missing critical domain-specific considerations.
3. **Methodological transparency issues**: The basis for LLM suggestions is often opaque, making it difficult to validate methodological soundness.
4. **Data recency limitations**: LLMs typically lack information about recent developments or cutting-edge methodologies in rapidly evolving fields.
5. **Deceptive confidence**: Models present information with uniform confidence regardless of the certainty or uncertainty behind their outputs.
6. **Reproducibility challenges**: The interactive nature of LLM collaboration can make analytical processes difficult to document and reproduce.
7. **Overreliance risk**: Analysts may develop dependence on LLM assistance, potentially atrophying their core analytical skills over time.

Assessment Guidance - Question 5

Excellent answers (70%+) will: - Demonstrate critical reflection on both benefits and limitations of LLM collaboration - Provide specific, concrete examples from personal or hypothetical analytics workflows - Consider ethical dimensions and professional responsibilities - Evaluate implications for the analytics profession and skill development - Present a balanced, nuanced perspective that acknowledges complexity

Good answers (60-69%) will: - Identify several key benefits and limitations - Provide some specific examples of LLM applications in analytics - Consider basic ethical concerns - Show awareness of professional implications - Present a relatively balanced view

Satisfactory answers (50-59%) will: - List some benefits and limitations - Provide general descriptions without specific examples - Show limited consideration of ethical dimensions - Offer basic reflections on professional impact

Conclusion and Final Assessment Guidance

This mock examination covers a comprehensive range of time series analysis techniques essential for financial data analytics. The solutions demonstrate the application of descriptive, analytical, and forecasting methods to various time series data with different characteristics. Let's summarize the key assessment criteria that apply across all questions:

Overall Assessment Criteria

Data Visualization Excellence (applicable to all questions) - Professional presentation with appropriate titles, labels, and annotations - Use of visual elements to highlight key patterns and outliers - Selection of appropriate visualization types for different analytical purposes - Clear communication of insights through visual means

Technical Accuracy (applicable to all questions) - Correct implementation of statistical methods - Appropriate selection of models based on data characteristics - Accurate interpretation of statistical outputs - Recognition of limitations in chosen approaches

Analytical Depth (applicable to all questions) - Connection of statistical findings to real-world context - Consideration of alternative approaches where appropriate - Critical evaluation of results and their implications - Balanced discussion of strengths and limitations

Communication Quality (applicable to all questions) - Clear explanation of technical concepts - Logical flow of analysis and reasoning - Appropriate use of technical terminology - Concise yet comprehensive responses

Final Guidance for Students

When preparing for this examination, focus on:

1. **Understanding the core principles** behind each time series technique rather than simply memorizing code
2. **Practicing interpretation** of various time series patterns and statistical outputs
3. **Developing critical thinking** about model selection and evaluation
4. **Honing your ability** to communicate technical findings clearly

Remember that in financial data analytics, the goal is not simply to apply statistical techniques correctly, but to extract meaningful insights that can inform decision-making. Your answers should demonstrate both technical competence and analytical thinking.

The ability to select appropriate methods, implement them correctly, interpret results accurately, and communicate findings clearly are all essential skills that this examination aims to assess. By focusing on developing these capabilities, you will be well-prepared not only for this examination but also for applying these skills in professional contexts.