Solutions

Mock Exam 2024

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

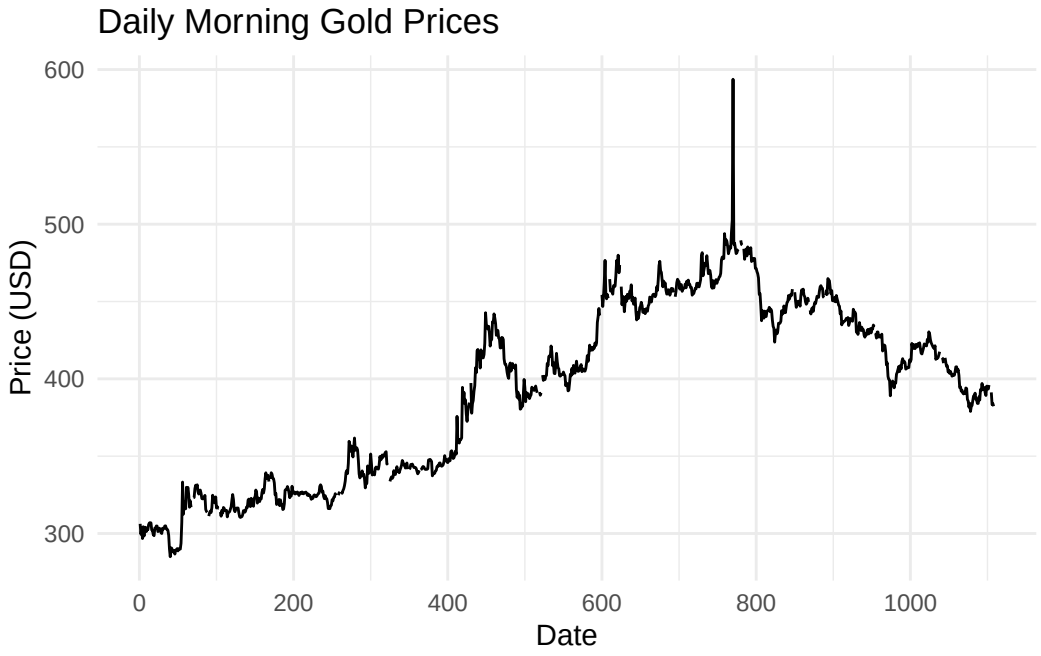
Question 1

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.

```
library(ggplot2)

autoplot(gold) +
  labs(title = "Daily Morning Gold Prices",
       x = "Date", y = "Price (USD)") +
  theme_minimal()
```




```
# autoplot(woolyrnq) +
#   labs(title = "Quarterly Wool Production in Australia",
#        x = "Date", y = "Production") +
#   theme_minimal()
#
# autoplot(gas) +
#   labs(title = "Australian Monthly Gas Production",
#        x = "Date", y = "Production") +
#   theme_minimal()
```

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.
- 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.
- The `gas` series shows an increasing trend with some seasonal variation. The seasonality appears to be less pronounced compared to the `woolyrnq` series.

b. Frequency of each series:

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

```r
frequency(gold)
```

```
[1] 1
```

```r
frequency(woolyrnq)
```

```
[1] 4
```

```r
frequency(gas)
```

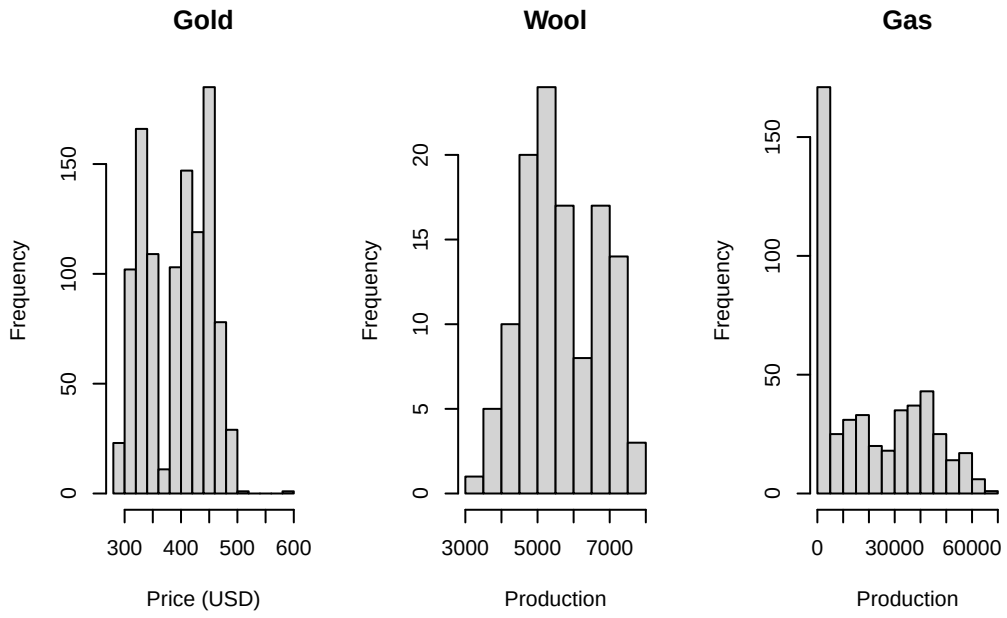
```
[1] 12
```

The output reveals that: - The `gold` series has a frequency of 365, indicating daily data (assuming daily data for all trading days in a year). - The `woolyrnq` series has a frequency of 4, indicating quarterly data. - The `gas` series has a frequency of 12, indicating monthly data.

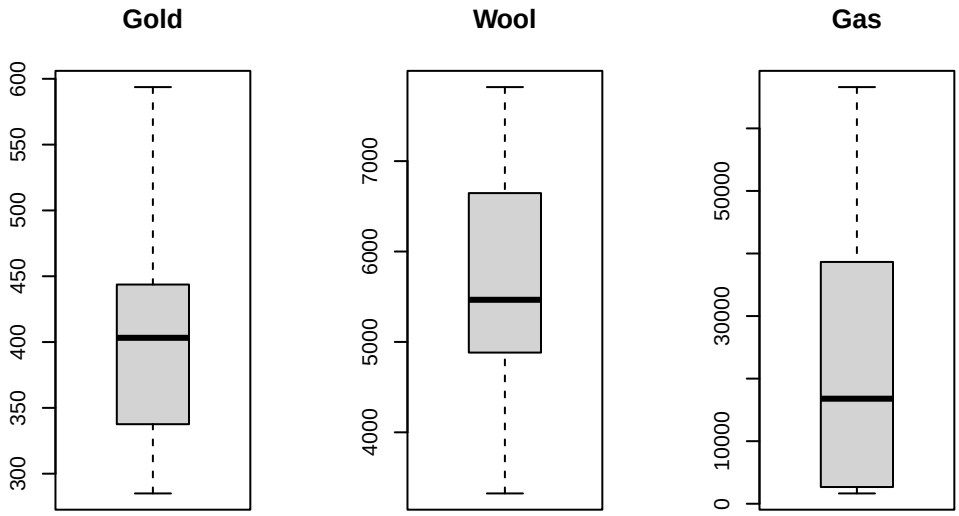
c. Checking for outliers:

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. - The `woolyrnq` and `gas` series do not show any clear outliers based on these visual methods.

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.

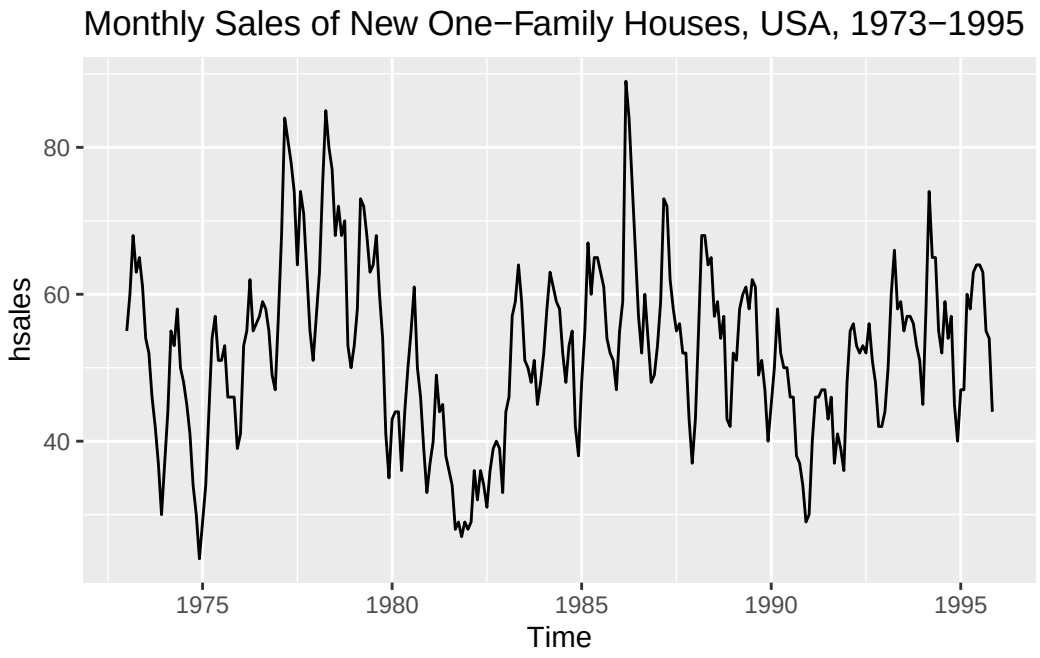
The `woolyrnq` and `gas` series do not exhibit any clear outliers based on the methods used in this analysis. However, it is always a good practice to use multiple outlier detection techniques and consider the domain knowledge and context of the data to make robust conclusions about the presence of outliers.

Question 2

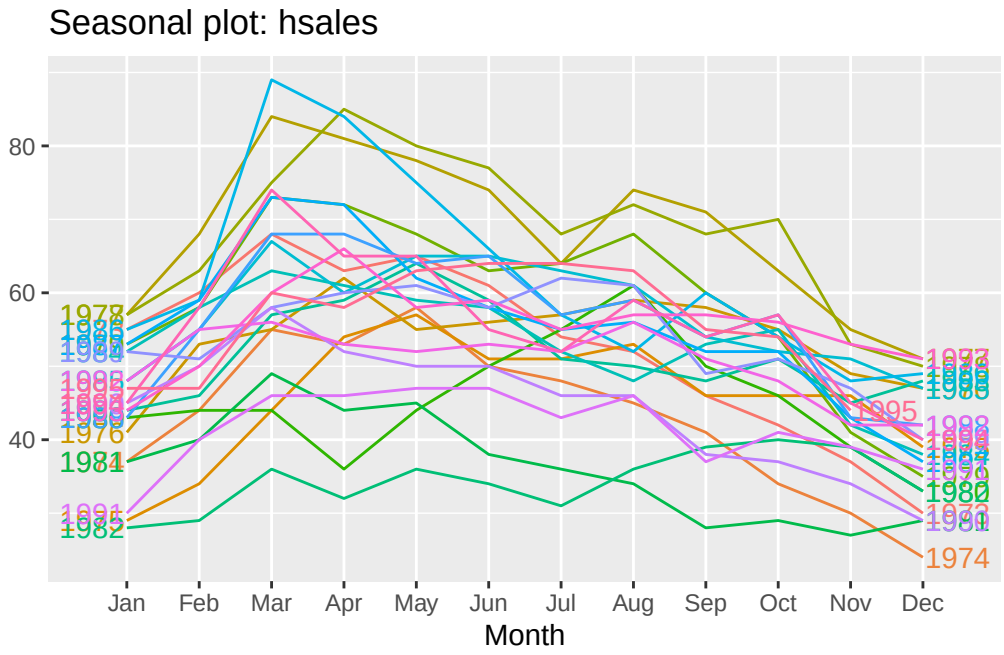
To explore the features of the given time series (`hsales`, `usdeaths`, `bricksq`, `sunspotarea`, `gasoline`), we can use various time series graphics functions in R. Let's analyse each series individually.

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

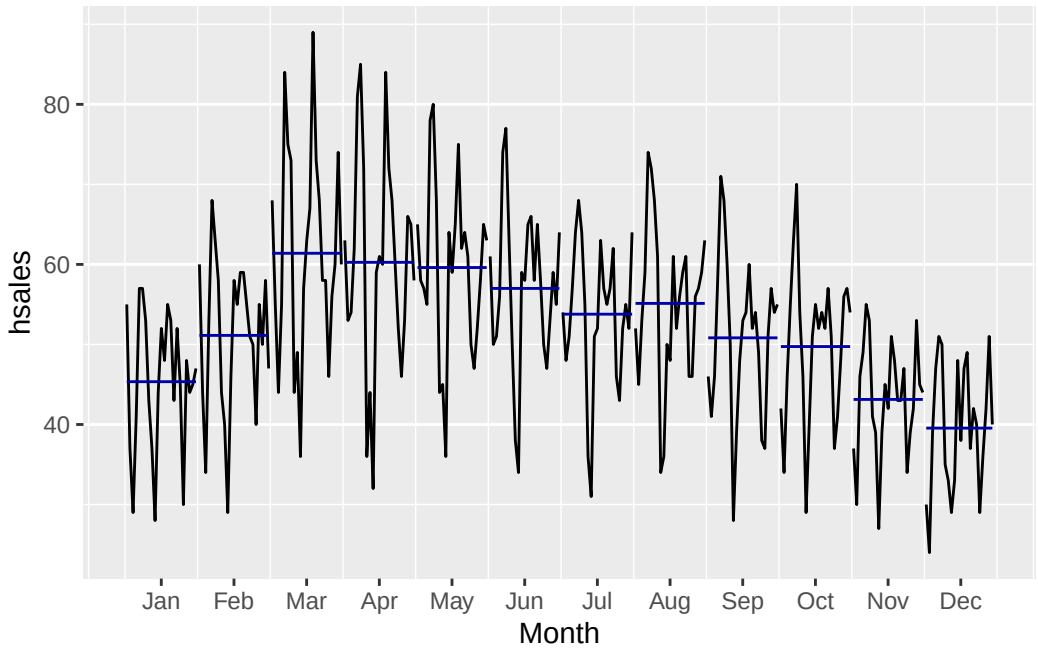
```
autoplot(hsales) + labs(title = "Monthly Sales of New One-Family Houses, USA, 1973-1995")
```



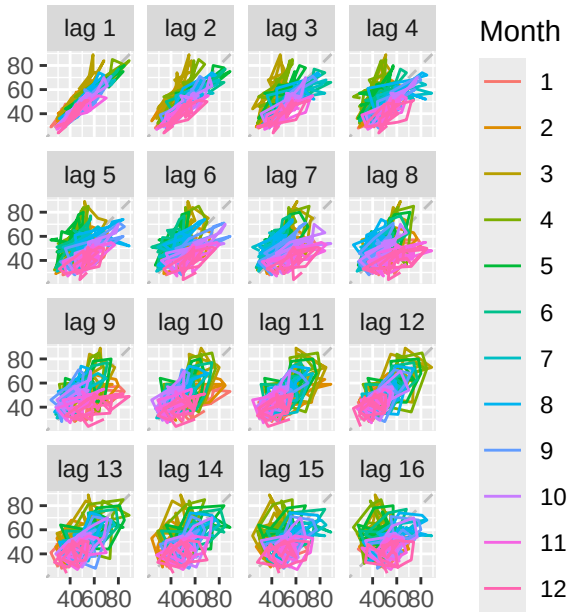

```
ggseasonplot(hsales, year.labels = TRUE, year.labels.left = TRUE)
```



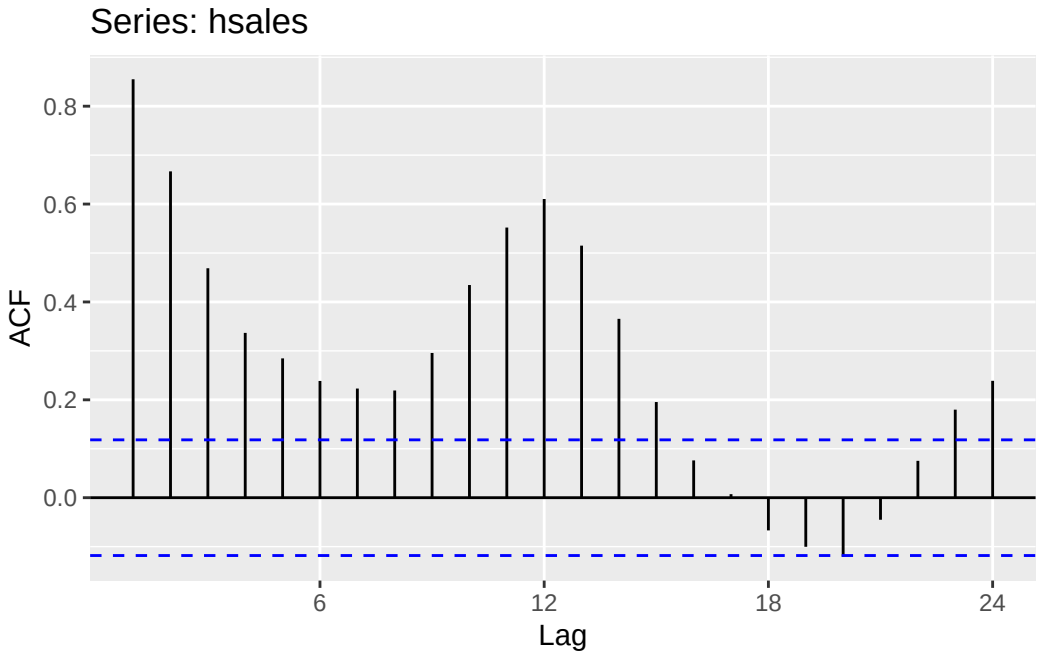

```
ggsubseriesplot(hsales)
```




```
gglagplot(hsales)
```



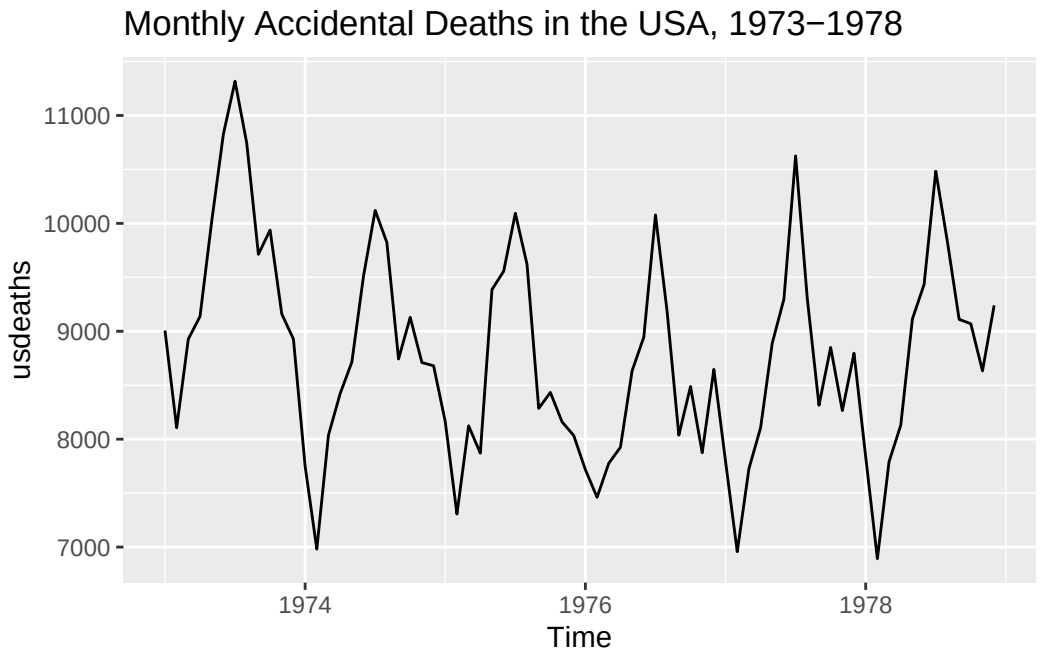

```
ggAcf(hsales)
```



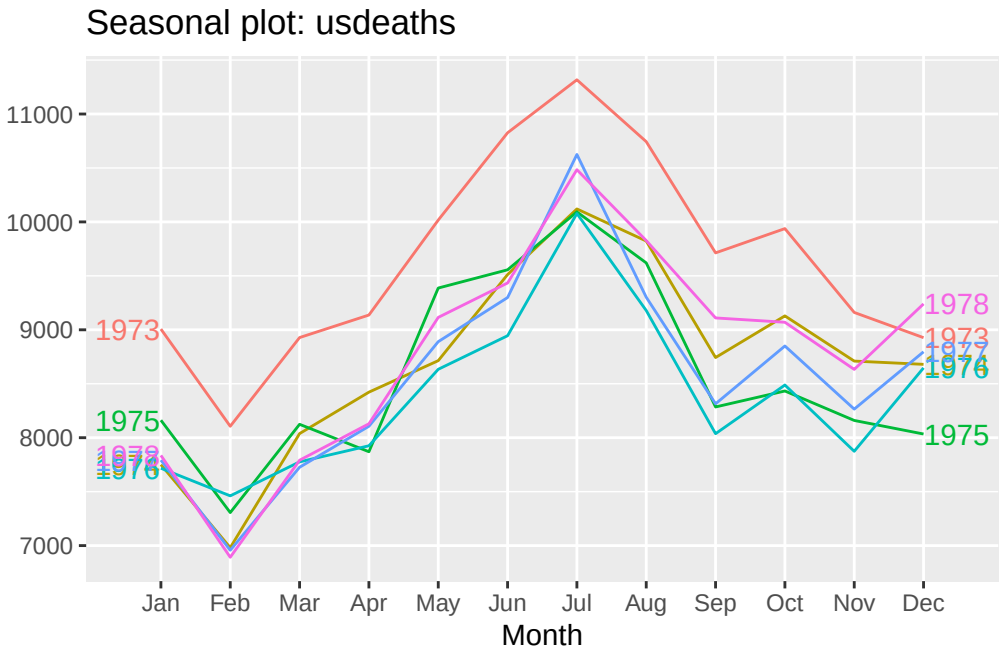
- The time plot shows a clear seasonal pattern and some long-term fluctuations.
- The seasonal plot confirms a strong and consistent seasonal pattern.
- The subseries plot shows that the seasonal pattern is similar across years.
- The lag plot indicates a positive relationship between consecutive observations, possibly due to trend or seasonal effects.
- The ACF plot shows significant autocorrelations at multiples of lag 12, confirming the presence of seasonality.

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

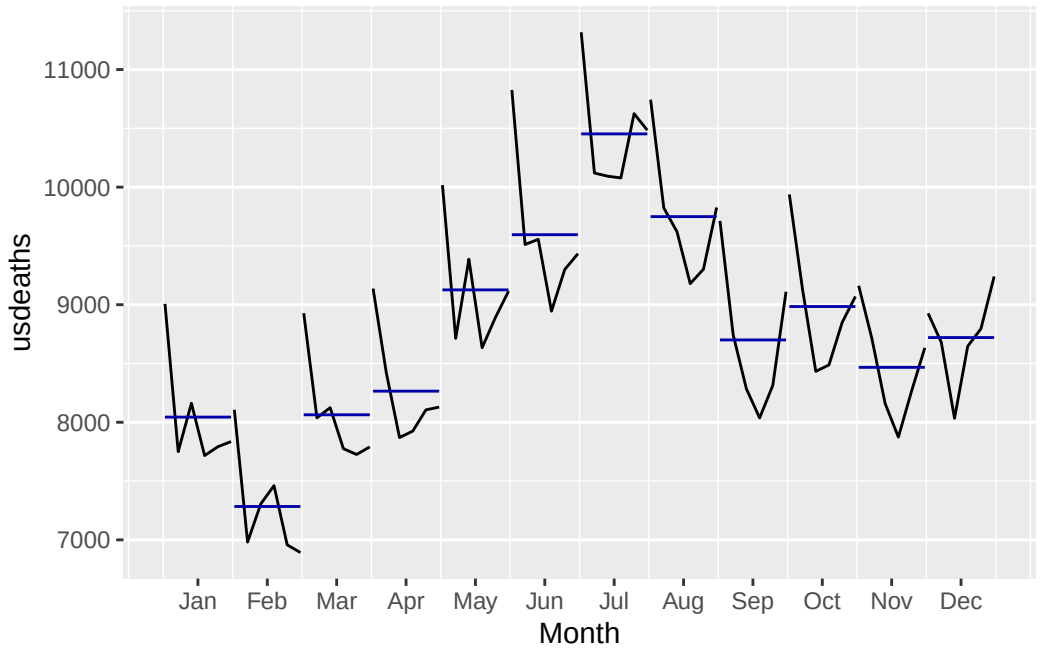
```
autoplot(usdeaths) + labs(title = "Monthly Accidental Deaths in the USA, 1973-1978")
```



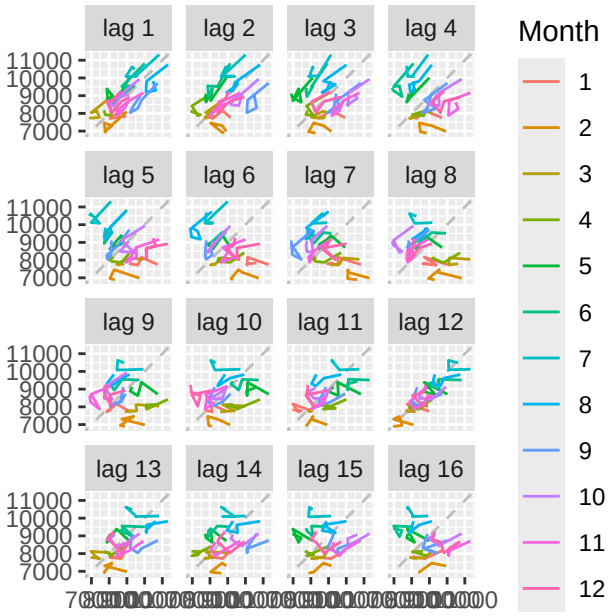

```
ggseasonplot(usdeaths, year.labels = TRUE, year.labels.left = TRUE)
```



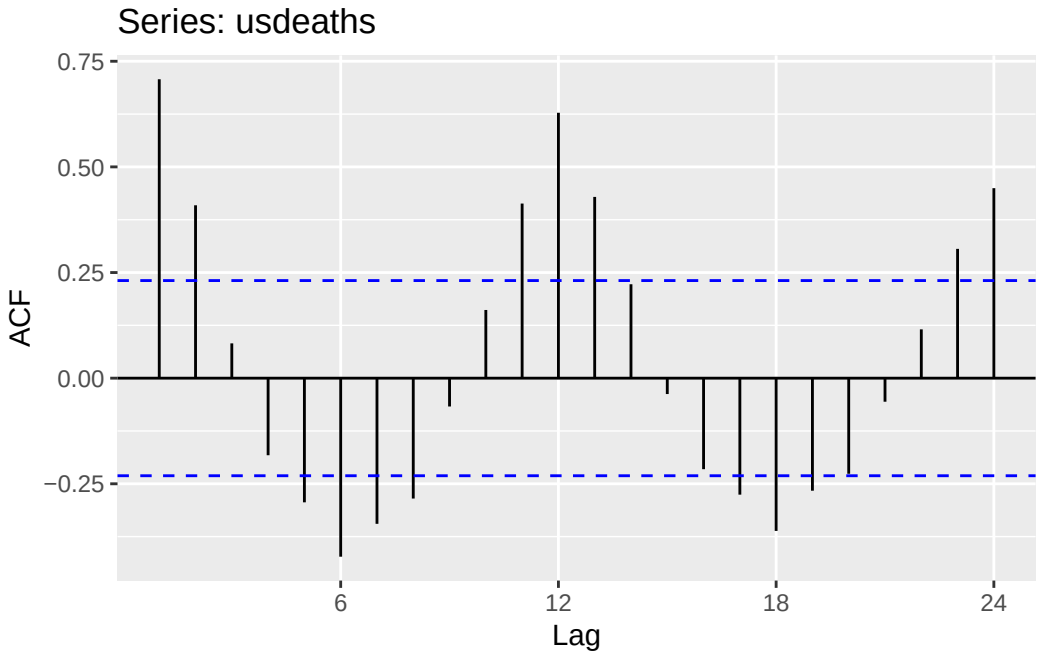

```
ggsubseriesplot(usdeaths)
```




```
gglagplot(usdeaths)
```



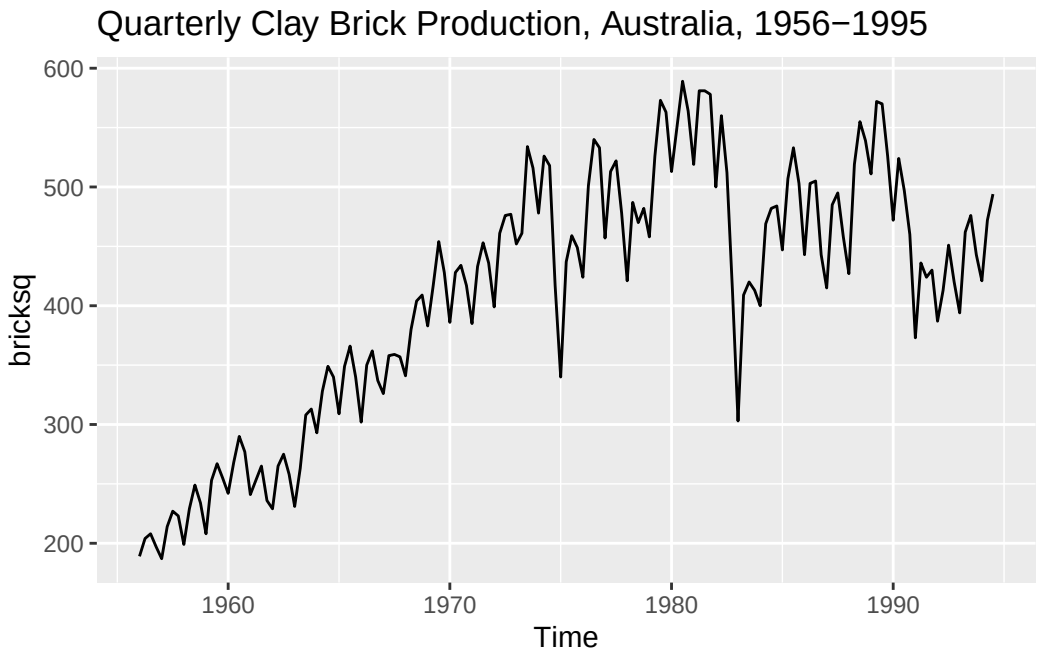

```
ggAcf(usdeaths)
```



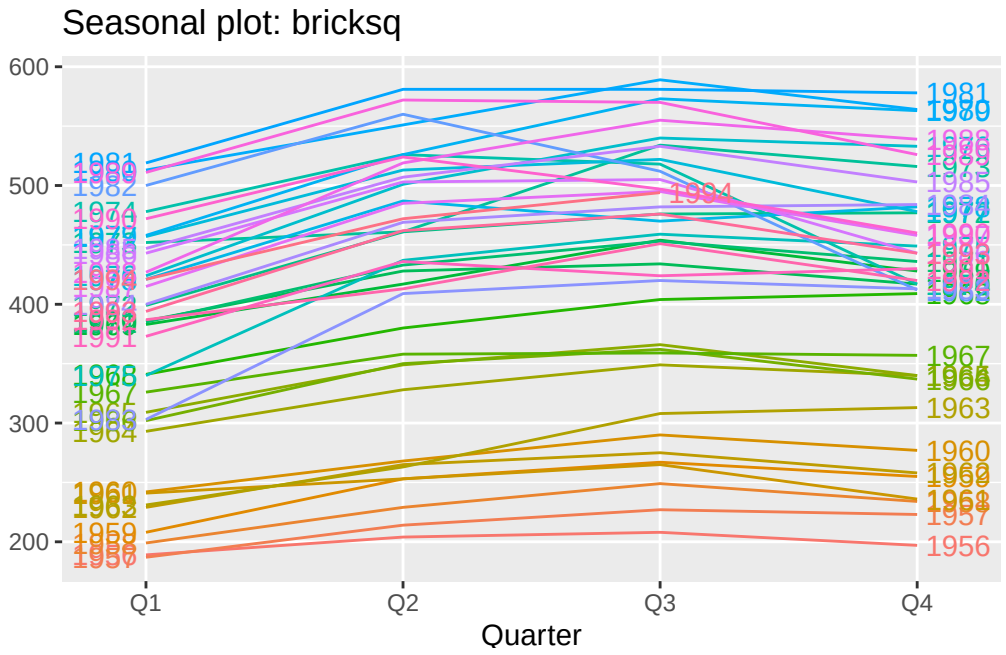
- The time plot shows a clear seasonal pattern but no obvious trend.
- The seasonal plot confirms the presence of seasonality, with higher deaths in summer months.
- The subseries plot shows that the seasonal pattern is consistent across years.
- The lag plot shows a positive relationship between consecutive observations due to seasonality.
- The ACF plot has significant autocorrelations at multiples of lag 12, confirming seasonality.

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

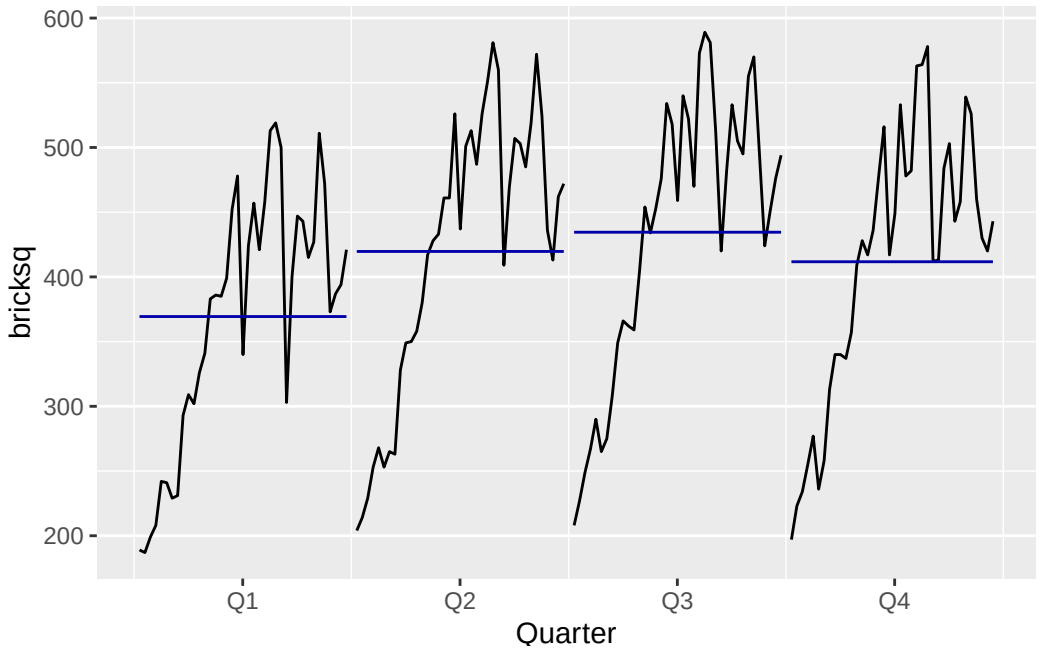
```
autoplot(bricksq) + labs(title = "Quarterly Clay Brick Production, Australia, 1956-1995")
```



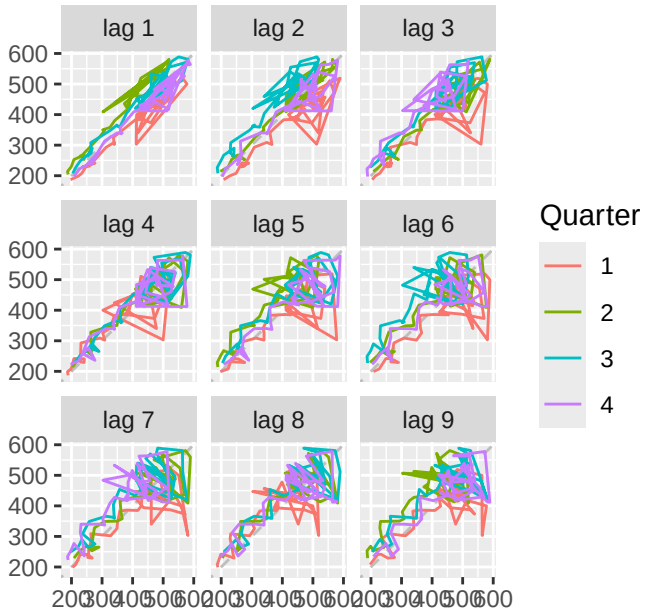

```
ggseasonplot(bricksq, year.labels = TRUE, year.labels.left = TRUE)
```



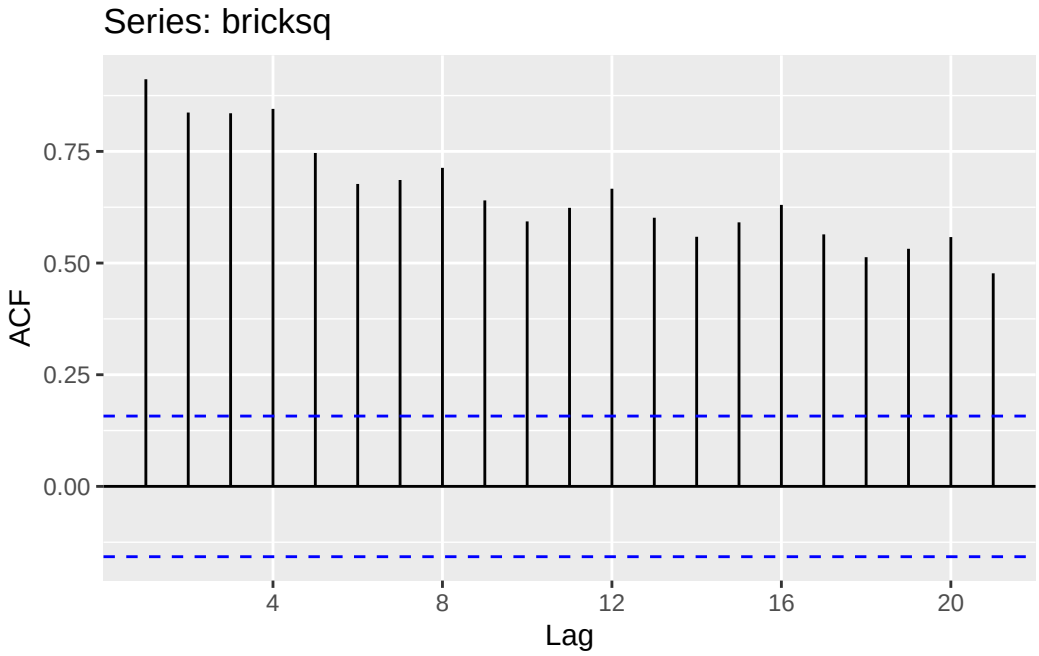

```
ggsubseriesplot(bricksq)
```




```
gglagplot(bricksq)
```



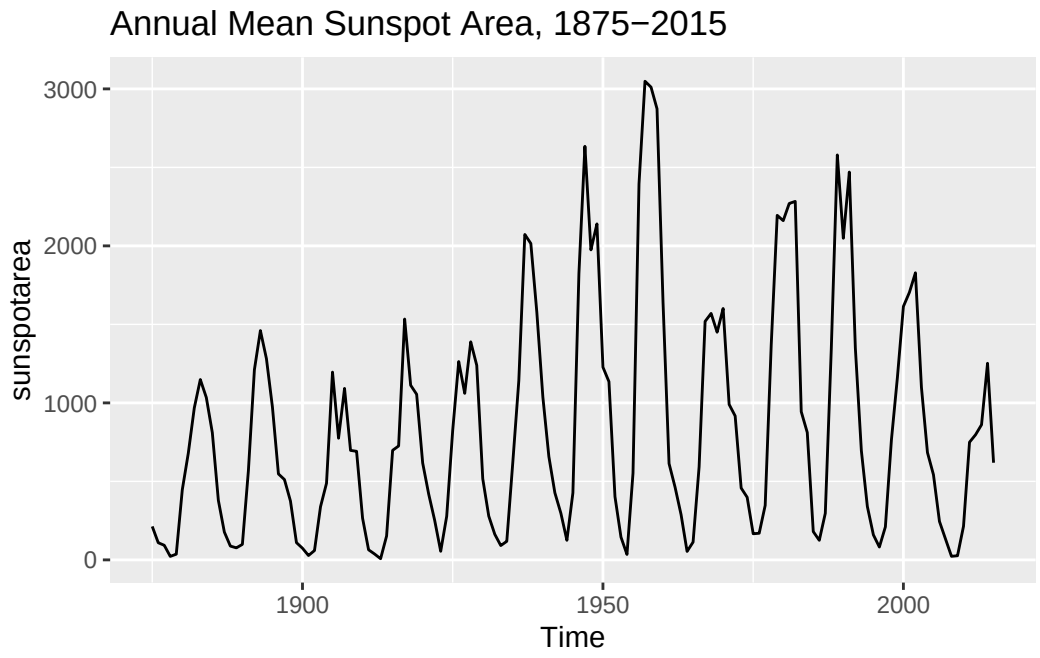

```
ggAcf(bricksq)
```



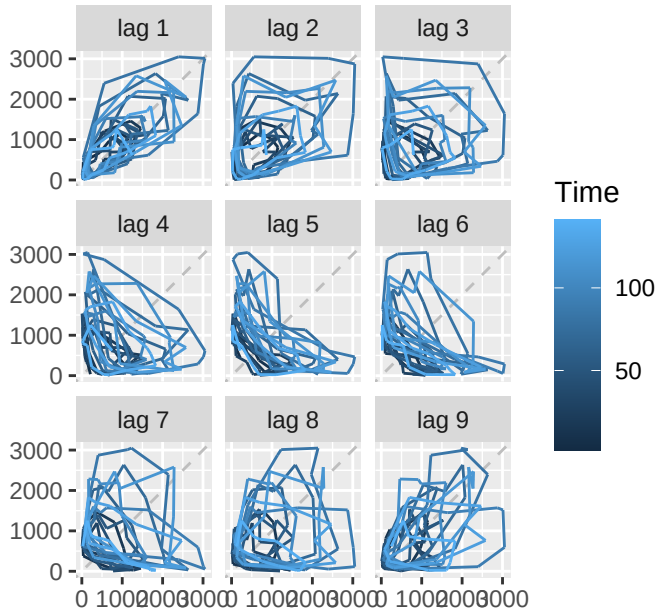
- The time plot shows a strong increasing trend and some seasonal fluctuations.
- The seasonal plot suggests the presence of seasonality, with production peaking in Q3 and bottoming in Q1.
- The subseries plot confirms that the seasonal pattern is generally consistent across years.
- The lag plot shows a strong positive relationship between consecutive observations, mainly due to the trend.
- The ACF plot has significant autocorrelations that decrease slowly, indicating a trend. There are also spikes at multiples of lag 4, suggesting seasonality.

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

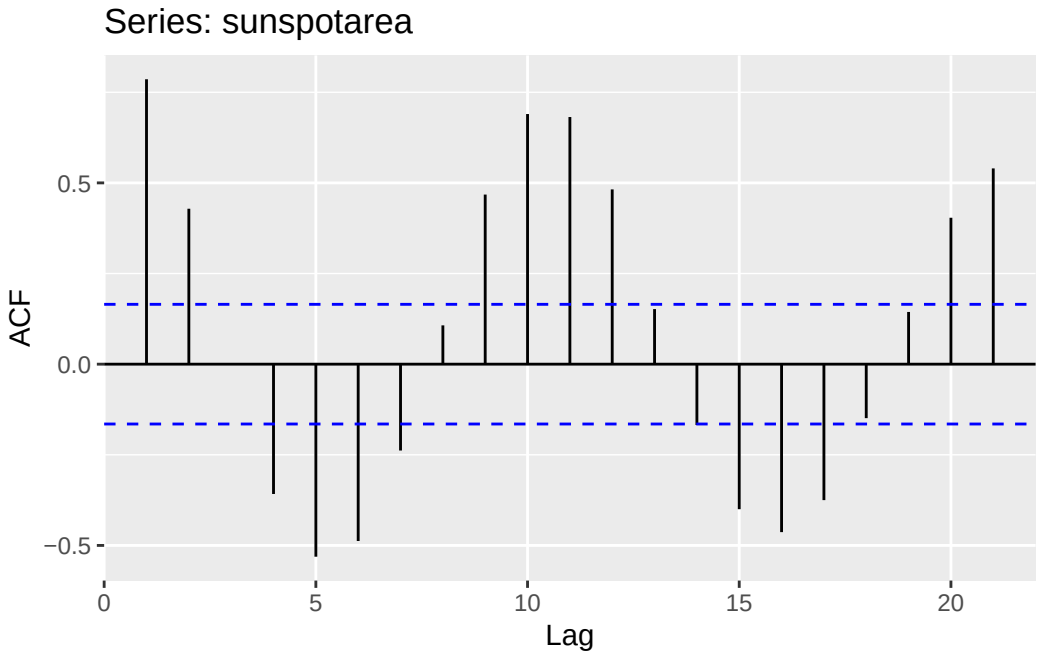
```
autoplot(sunspotarea) + labs(title = "Annual Mean Sunspot Area, 1875-2015")
```




```
gglagplot(sunspotarea)
```



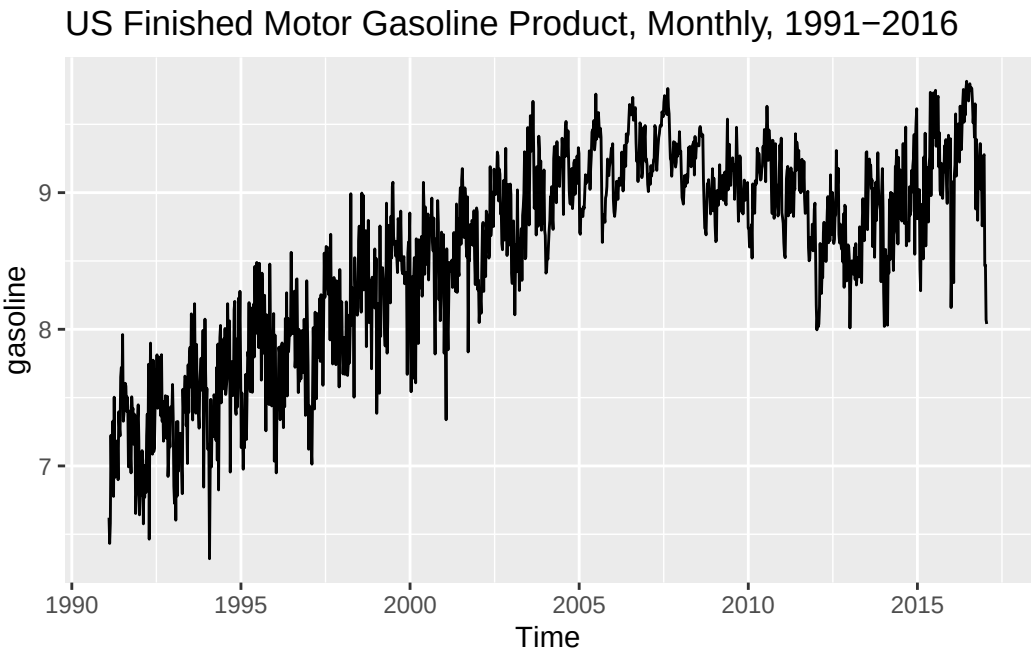

```
ggAcf(sunspotarea)
```



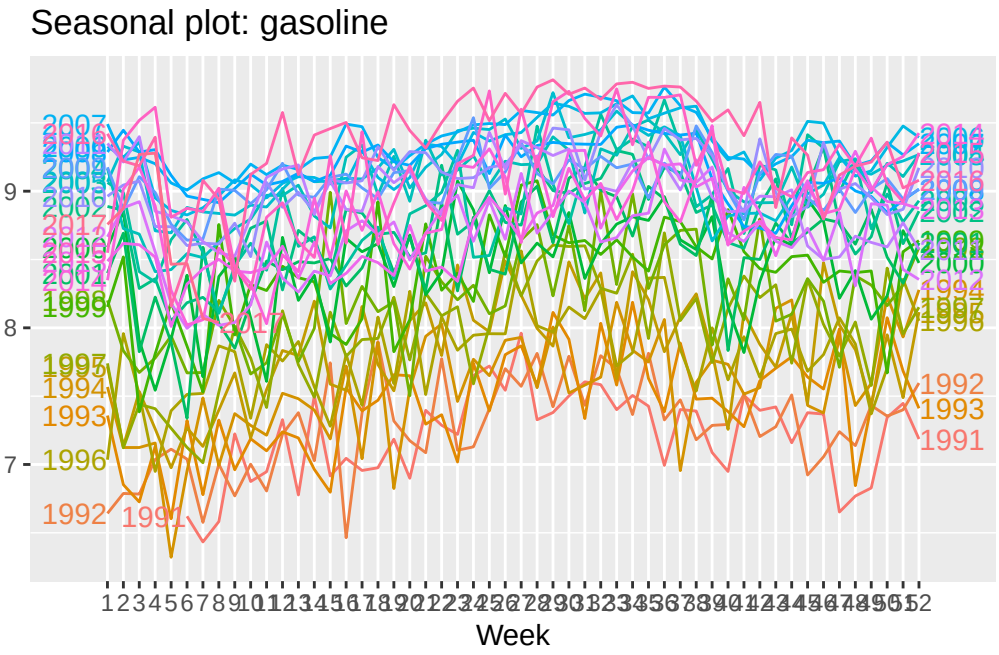
- The time plot shows a clear cyclical pattern with a period of around 11 years.
- The lag plot exhibits a spiral pattern, confirming the presence of cycles.
- The ACF plot has a sinusoidal pattern, alternating between positive and negative autocorrelations, which is characteristic of cyclical behavior.

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

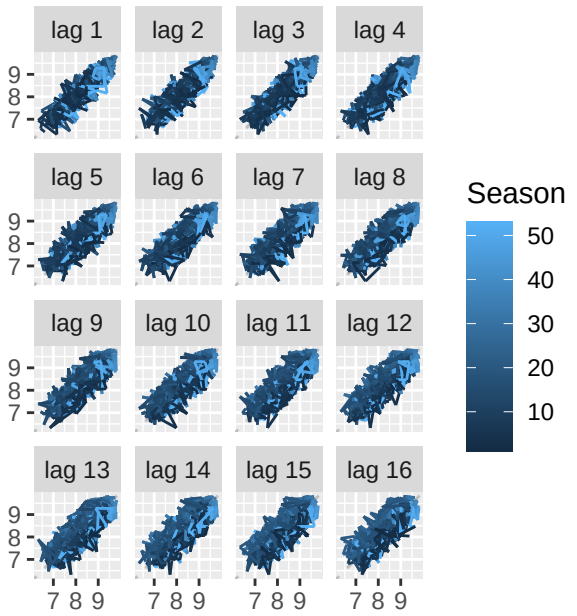
```
autoplot(gasoline) + labs(title = "US Finished Motor Gasoline Product, Monthly, 1991-2016")
```



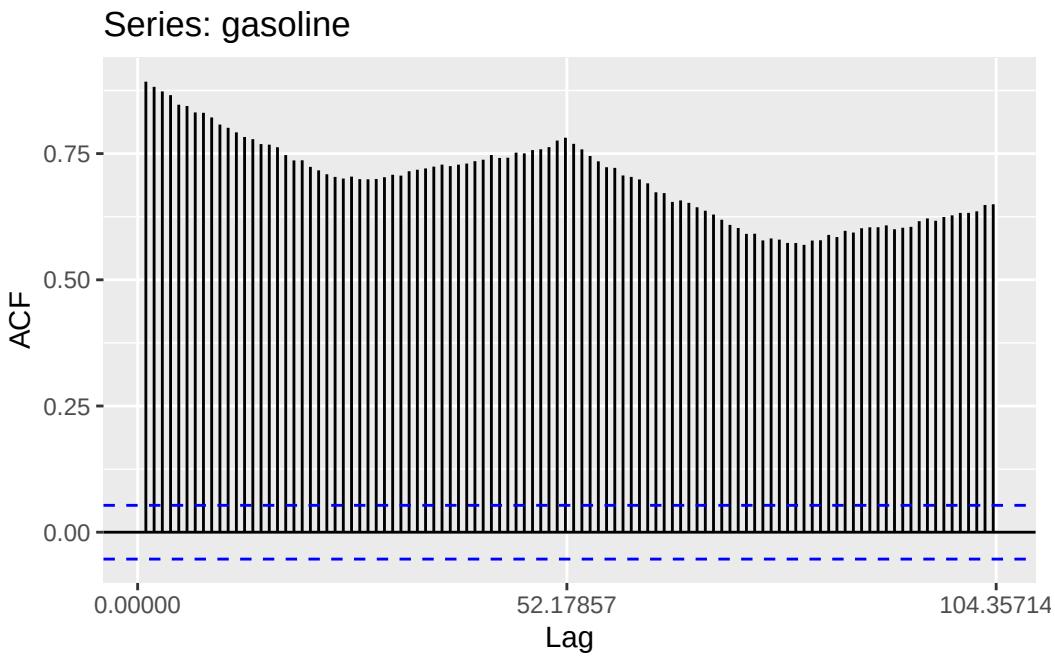

```
ggseasonplot(gasoline, year.labels = TRUE, year.labels.left = TRUE)
```




```
gglagplot(gasoline)
```




```
ggAcf(gasoline)
```



- The time plot shows an increasing trend and strong seasonality.
- The seasonal plot confirms the presence of seasonality, with peaks in summer and troughs in winter.
- The lag plot is not very informative due to the strong seasonality and trend.

- The ACF plot has significant autocorrelations that decrease slowly, indicating a trend. The seasonal spikes are also evident, confirming seasonality.

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.

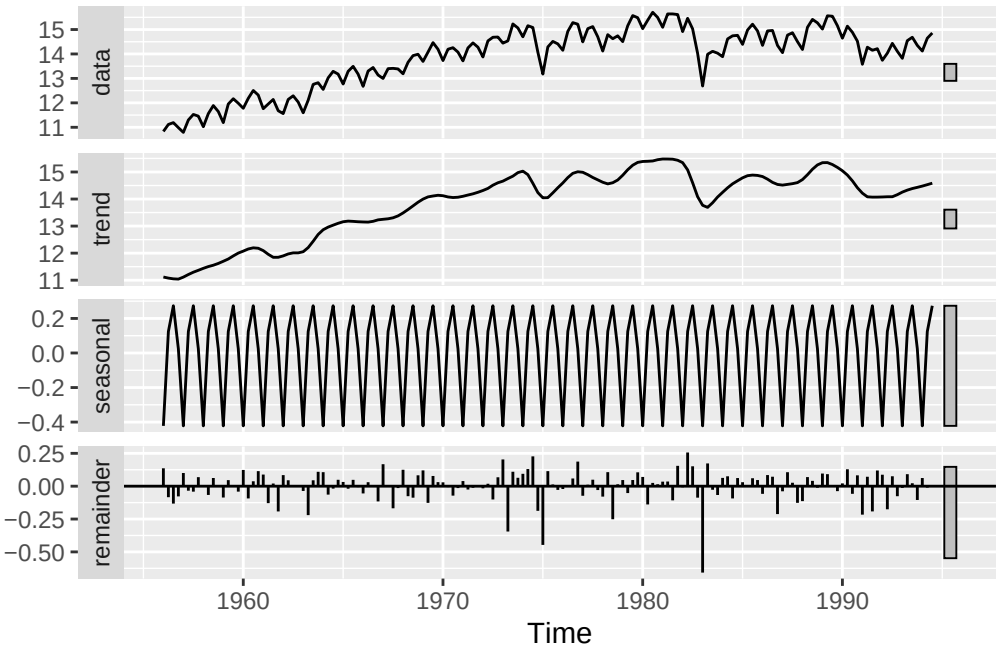
Question 3

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)
```



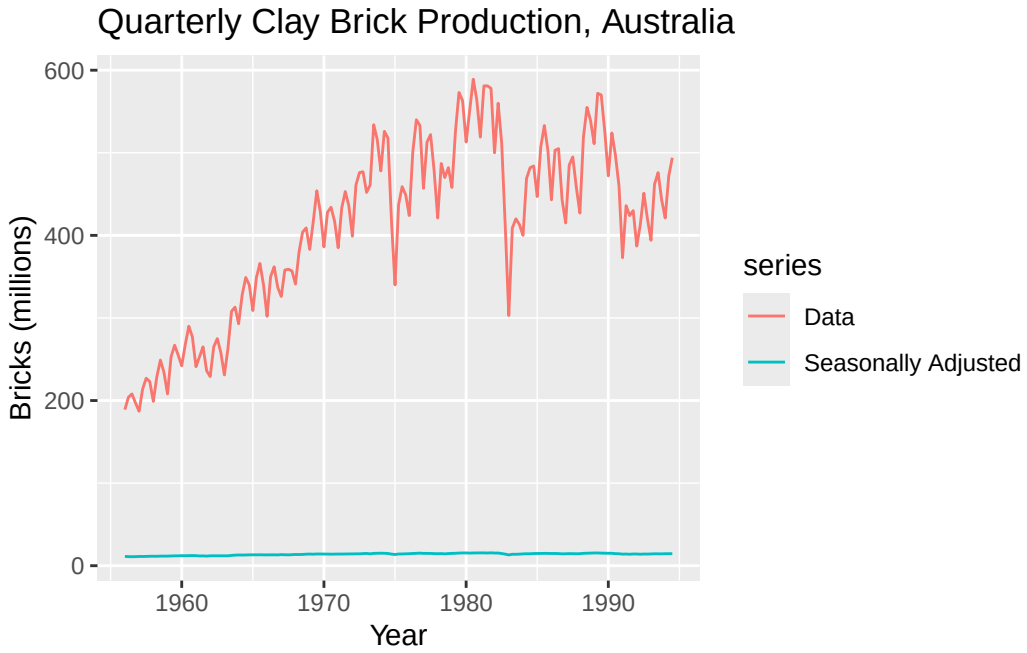
The STL decomposition reveals the following: - The data exhibits a strong increasing trend over time. - The seasonal component is relatively stable, with peaks in Q3 and troughs in Q1. - The remainder component shows some fluctuations, particularly during the 1970s and early 1980s.

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") +
  xlab("Year") + ylab("Bricks (millions)") +
  ggtitle("Quarterly Clay Brick Production, Australia")
```

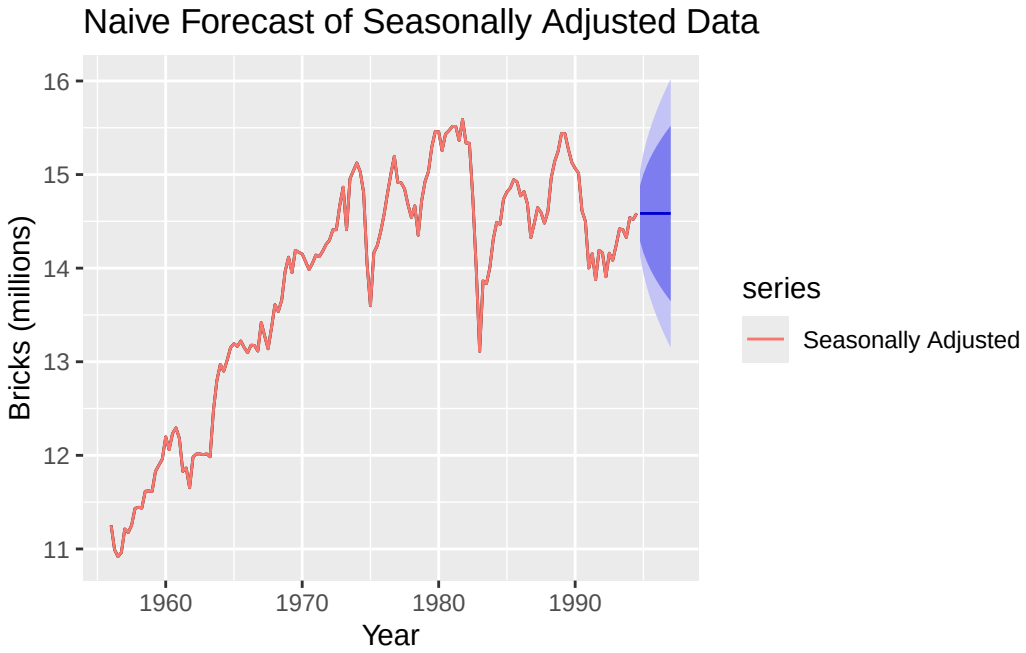


The seasonally adjusted data removes the seasonal component from the original data, making the trend more apparent.

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") +
  xlab("Year") + ylab("Bricks (millions)") +
  ggtitle("Naive Forecast of Seasonally Adjusted Data")
```

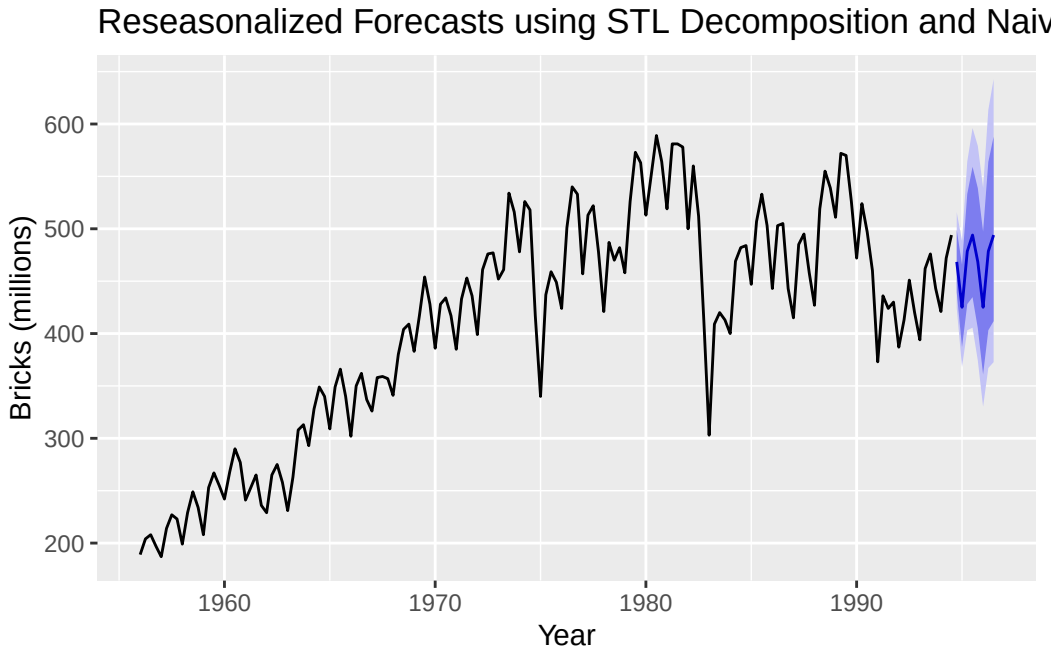


The naive method simply extends the last observed value of the seasonally adjusted data as the forecast.

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) +
  xlab("Year") + ylab("Bricks (millions)") +
  ggtitle("Reseasonalized Forecasts using STL Decomposition and Naive Method")
```

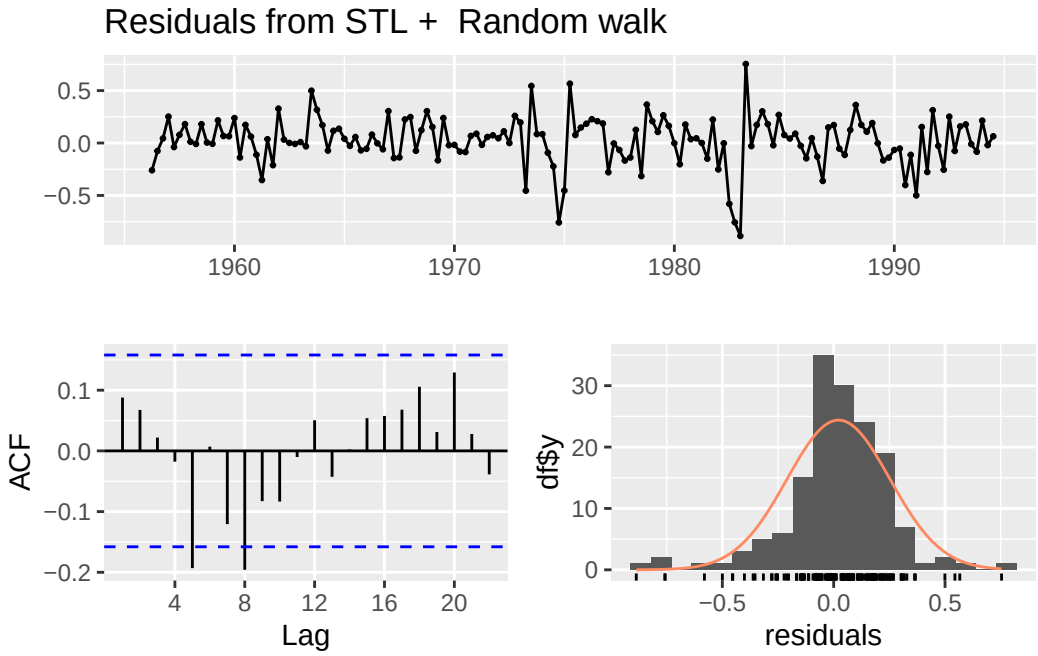


The `stlf()` function adds the seasonal component back to the forecasted trend and remainder components to obtain the final forecasts for the original data.

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
```

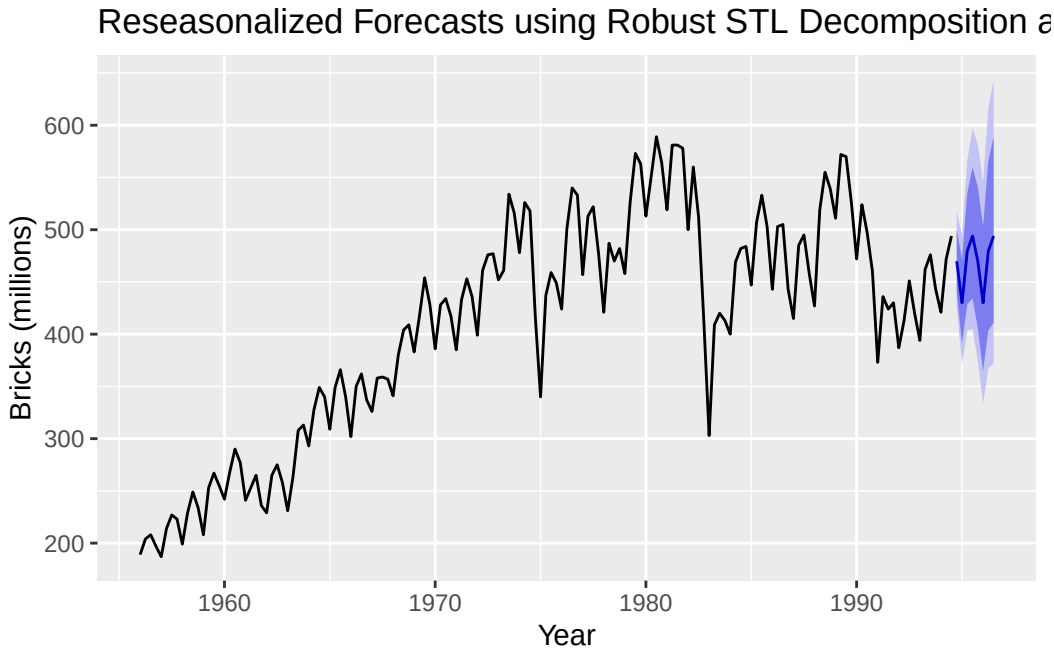
The residual diagnostics show that: - The standardised residuals appear to be approximately normally distributed. - The ACF plot of the residuals shows a significant spike at lag 1, indicating some remaining autocorrelation. - The p-values for the Ljung-Box test are below 0.05 for most lags, suggesting that the residuals are not entirely uncorrelated.

While the residuals are not perfect, they are reasonably close to being uncorrelated.

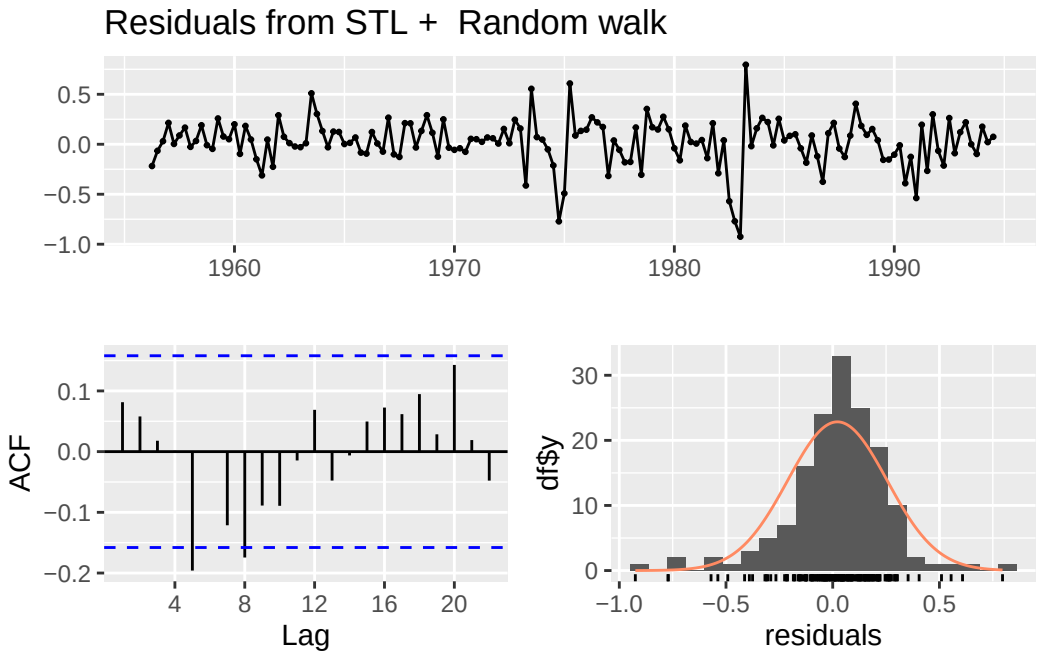
f. Robust STL decomposition:

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) +
  xlab("Year") + ylab("Bricks (millions)") +
  ggtitle("Reseasonalized Forecasts using Robust STL Decomposition and Naive Method")
```




```
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.

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)

accuracy(fc1, bricksq)
```

```
                    ME     RMSE      MAE        MPE      MAPE      MASE
Training set  1.918581 21.63284 14.95359  0.4629703  3.752004 0.4337416
Test set     71.642824 76.70300 71.64282 13.2727308 13.272731 2.0780609
                  ACF1 Theil's U
Training set 0.1261000        NA
Test set     0.4409137  1.492245
```

```
accuracy(fc2, bricksq)
```

```
                    ME     RMSE      MAE       MPE      MAPE MASE      ACF1
Training set  8.508065 48.60531 34.47581  2.001127  8.666006 1.00 0.7955186
Test set     64.125000 69.36768 64.12500 11.910267 11.910267 1.86 0.5684819
             Theil's U
Training set        NA
Test set      1.332503
```

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.

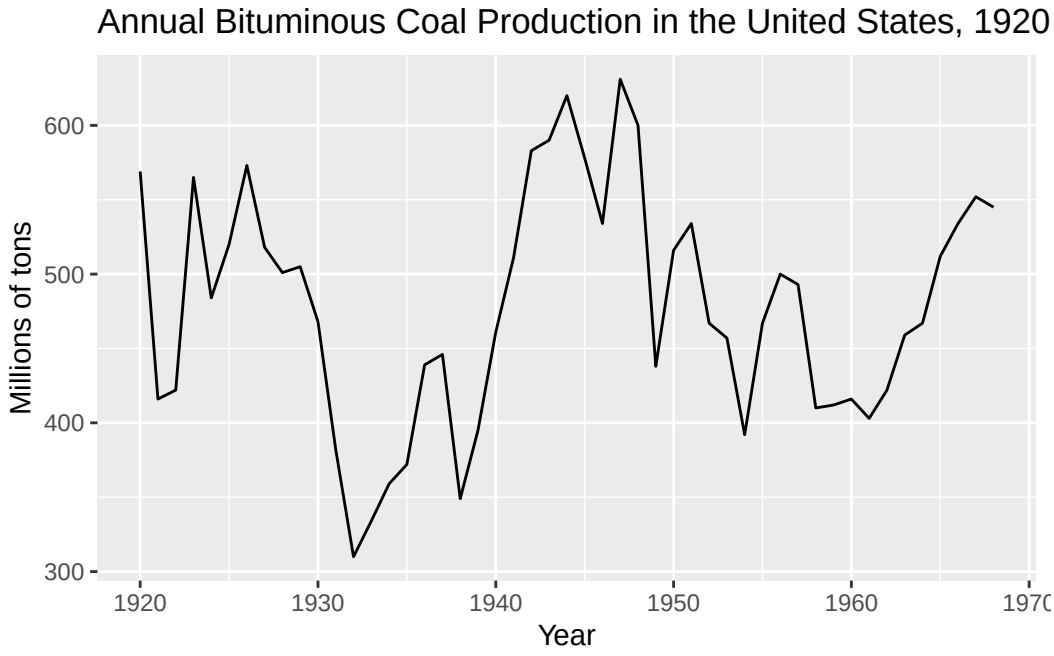
Question 4

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")
```



The plot shows an overall increasing trend in the bituminous coal production over the given period. There are some fluctuations around the trend, but no clear seasonality or cyclicity is evident.

b. Identifying the ARIMA model:

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

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

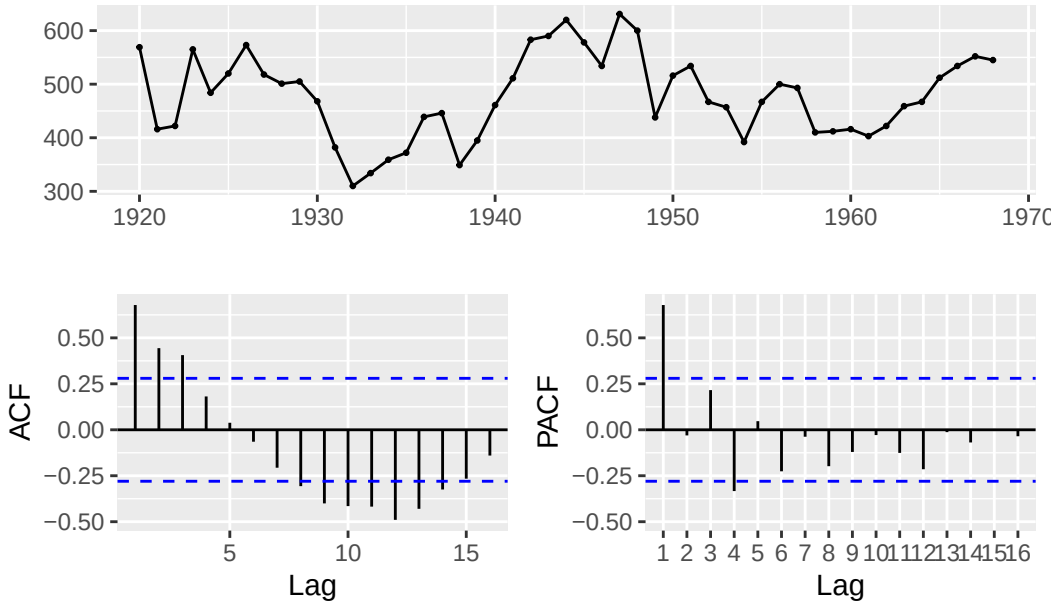
where: - y_t is the coal production in year t - c is a constant term - $\phi_1, \phi_2, \phi_3, \phi_4$ are the autoregressive coefficients - e_t is a white noise series

In this model: - $p = 4$ (autoregressive order) - $d = 0$ (no differencing) - $q = 0$ (no moving average term)

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)
```



The ACF plot shows a slow decay in the autocorrelations, which is characteristic of an autoregressive process. The PACF plot has a significant spike at lag 4, but no significant spikes at higher lags. This suggests that an AR(4) model might be appropriate for the data.

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)

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

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
```

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.

In summary, this question demonstrates the process of fitting an ARIMA(4,0,0) model to the `bicoal` time series. The model choice is justified using the ACF and PACF plots, which suggest the presence of an autoregressive process of order 4. The forecasts for the next three years are calculated manually using the estimated model parameters and compared with the results obtained from the `forecast` function in R. The analysis highlights the importance of understanding the underlying model structure and the ability to interpret the results in the context of the given problem.

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 processing and analysis of large textual datasets 2. Automated identification of key themes, sentiment patterns, and actionable insights 3. Engaging in meaningful dialogue and generating human-like responses 4. Assisting with tasks such as text summarization, content generation, and code assistance 5. Fostering creativity and considering diverse perspectives 6. Enhancing efficiency by providing excellent starting points for iteration and improvement 7. Combining human expertise with machine intelligence for powerful problem-solving

Limitations: 1. Potential for biased or misleading outputs due to biases in training data 2. Lack of domain-specific knowledge and context in highly specialized or niche topics 3. Need for human oversight and critical evaluation of AI-generated content 4. Challenges in interpretability and explainability of LLM outputs 5. Requirement for a strong ethical framework and responsible use of AI technologies 6. Balancing trust in AI capabilities with critical evaluation of its suggestions 7. Communicating the reasoning behind AI-generated insights to stakeholders

! Important

In this sample answer, I have provided a reflective essay that explores the experience of collaborating with Large Language Models in an analytics project. The essay discusses the benefits, such as efficient data processing, creative problem-solving, and enhanced productivity. It also acknowledges the limitations, including potential biases, lack of domain-specific knowledge, and challenges in interpretability and explainability.
The answer then summarizes the key benefits and limitations in a concise list format, highlighting the importance of human oversight, critical evaluation, and ethical considerations when working with AI technologies.
This sample answer demonstrates a balanced and thoughtful approach to the question, providing insights into the transformative potential of collaborating with LLMs while also acknowledging the challenges and the need for responsible use and human expertise in the process.