ECON 144 Proj 2

Leonard Zhu

2024-11-14

1. Modeling and Forecasting Trend, Seasonality, and Cycles

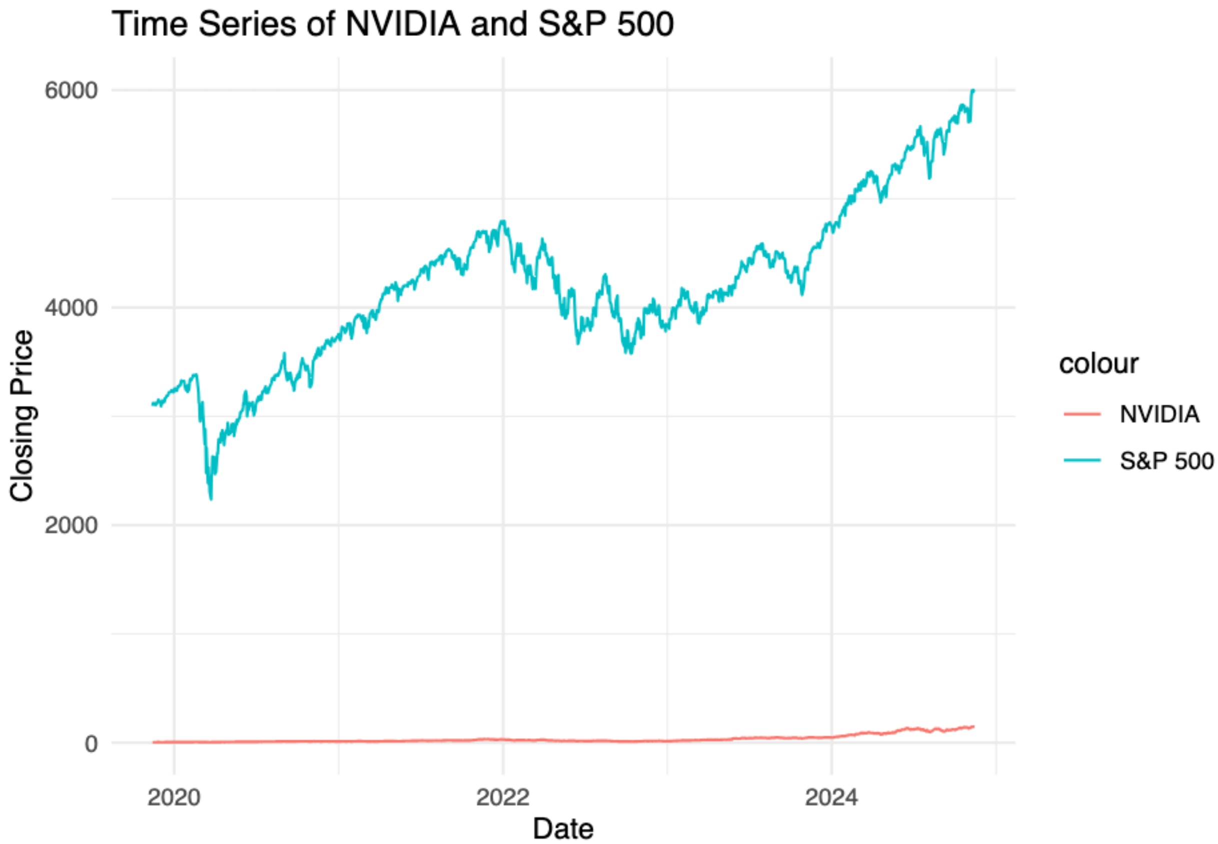
In this section, I load the data, ensure the date is formatted correctly, and merge the datasets by date to align both NVIDIA and S&P 500 data series.

```r
# Load NVIDIA data
nvidia_data <- read_excel("~/Downloads/NVIDIA_Data.xlsx")
# Load S&P500 data
sp500_data <- read_excel("~/Downloads/S&P500_Data.xlsx")

# Ensure date formatting and combine data into a time-series compatible format
nvidia_data$Date <- as.Date(nvidia_data$Date)
sp500_data$Date <- as.Date(sp500_data$Date)

# Merge datasets on date (if needed)
data_combined <- merge(nvidia_data, sp500_data, by = "Date")
```

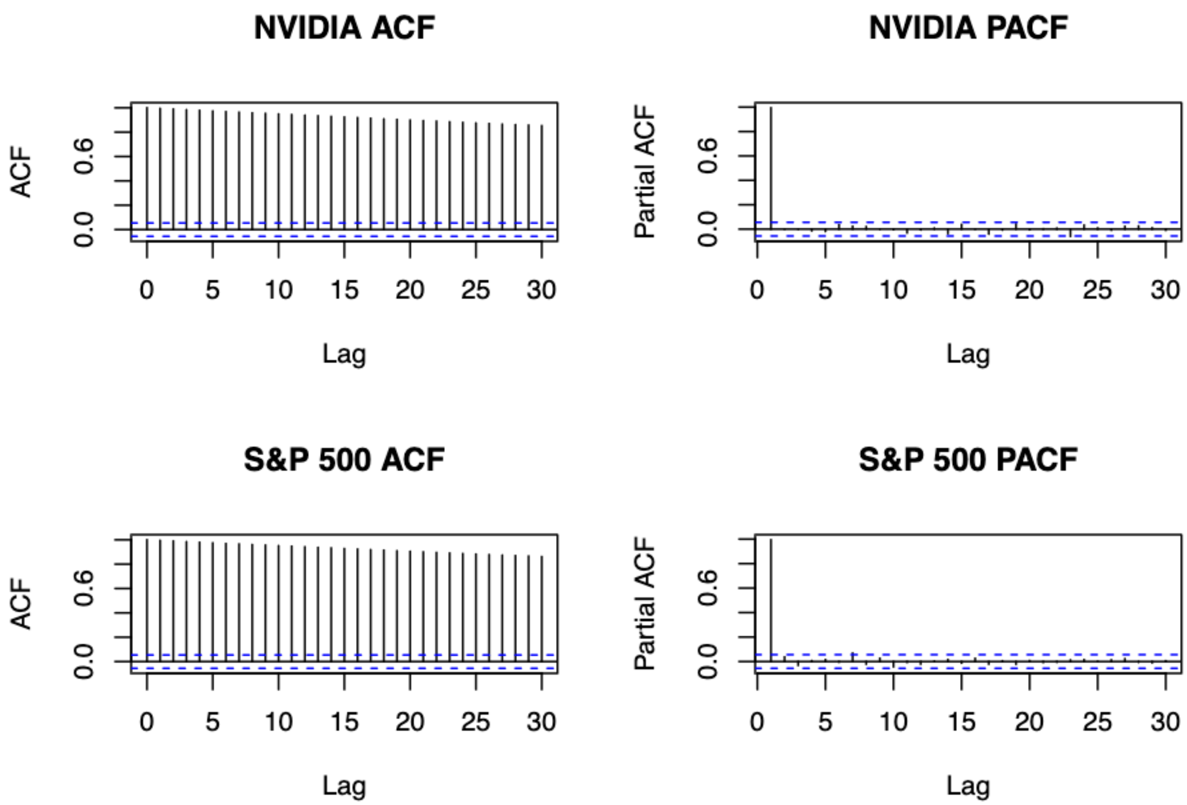
Plotting Time Series and Analyzing ACF/PACF

Here, I plot the time series for NVIDIA and S&P 500 closing prices to visually inspect any apparent relationship between the two. I also calculate and plot the ACF and PACF to analyze potential autocorrelations in each series.

```r
# Plot time-series
ggplot(data_combined, aes(x = Date)) +
  geom_line(aes(y = NVIDIA_Close, color = "NVIDIA")) +
  geom_line(aes(y = SP500_Close, color = "S&P 500")) +
  labs(title = "Time Series of NVIDIA and S&P 500", y = "Closing Price") +
  theme_minimal()
```



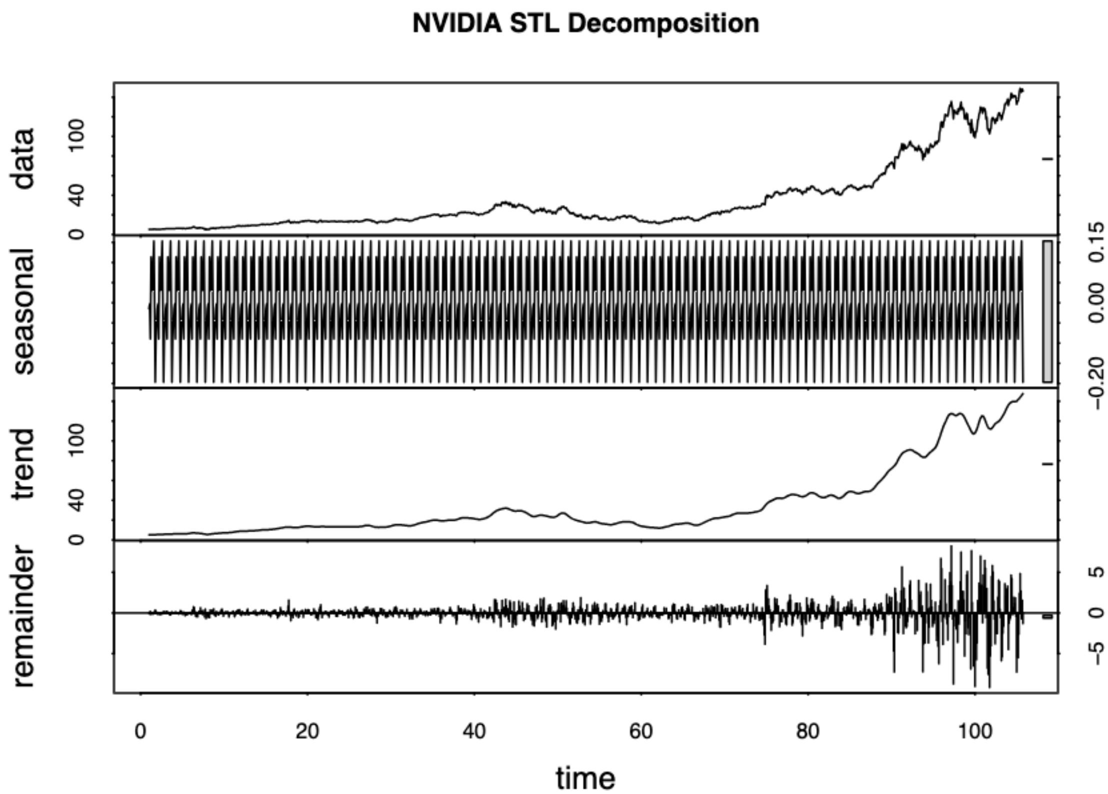

```r
# ACF and PACF
par(mfrow = c(2, 2))
acf(data_combined$NVIDIA_Close, main = "NVIDIA ACF")
pacf(data_combined$NVIDIA_Close, main = "NVIDIA PACF")
acf(data_combined$SP500_Close, main = "S&P 500 ACF")
pacf(data_combined$SP500_Close, main = "S&P 500 PACF")
```



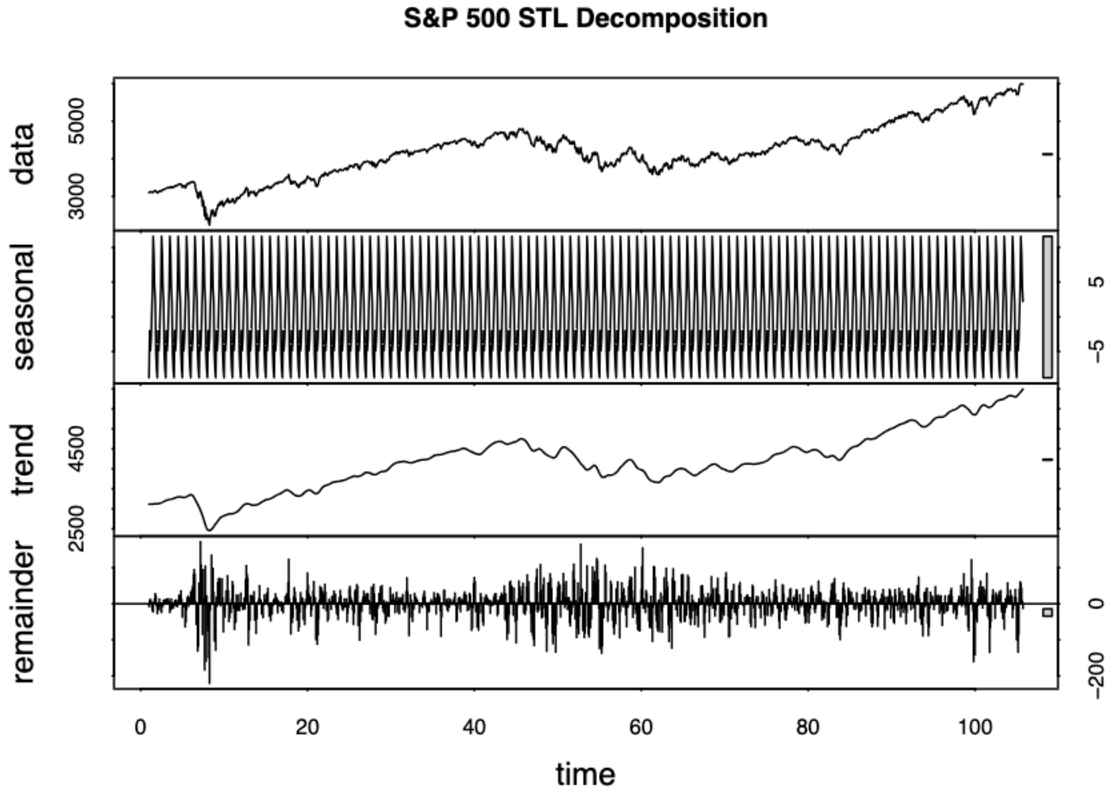
STL Decomposition

I perform STL decomposition on both time series to break down each into trend, seasonal, and remainder components. This helps identify underlying patterns in the data.

```
# Convert to time series and perform STL decomposition
nvidia_ts <- ts(data_combined$NVIDIA_Close, frequency = 12)
sp500_ts <- ts(data_combined$SP500_Close, frequency = 12)

nvidia_stl <- stl(nvidia_ts, s.window = "periodic")
sp500_stl <- stl(sp500_ts, s.window = "periodic")

plot(nvidia_stl, main = "NVIDIA STL Decomposition")
```




```r
plot(sp500_stl, main = "S&P 500 STL Decomposition")
```



Fitting ARIMA Models

Using auto.arima, I fit ARIMA models for NVIDIA and S&P 500 to capture trend, seasonality, and cycles, adjusting for differencing and periodic components.

```r
# Fit a model (e.g., ARIMA with trend and seasonality)
nvidia_arima <- auto.arima(nvidia_ts, seasonal = TRUE)
sp500_arima <- auto.arima(sp500_ts, seasonal = TRUE)

summary(nvidia_arima)
```

```
## Series: nvidia_ts
## ARIMA(5,2,0)(0,0,2)[12]
##
## Coefficients:
##           ar1      ar2      ar3      ar4      ar5     sma1     sma2
##       -0.9018  -0.6644  -0.5051  -0.3157  -0.2101  -0.0129  -0.0632
## s.e.   0.0277   0.0374   0.0389   0.0367   0.0281   0.0303   0.0303
##
## sigma^2 = 3.016:  log likelihood = -2472.61
## AIC=4961.21   AICc=4961.33   BIC=5002.3
##
## Training set error measures:
##                          ME     RMSE       MAE          MPE      MAPE       MASE
```

```
## Training set 0.001077871 1.730506 0.9296425 -0.00354818 2.690991 0.2843258
##                         ACF1
## Training set -0.03820278
```

Here we can see that the ARIMA of the NVIDA stock is displayed as ARIMA(5,2,0)(0,0,2). This ARIMA(5,2,0)(0,0,2)[12] model uses five autoregressive terms and two levels of differencing to capture non-seasonal patterns. The model includes two seasonal moving average terms with a yearly seasonality period (for monthly data), which helps in capturing any repeating annual pattern without additional differencing for the seasonal component.

```r
summary(sp500_arima)
```

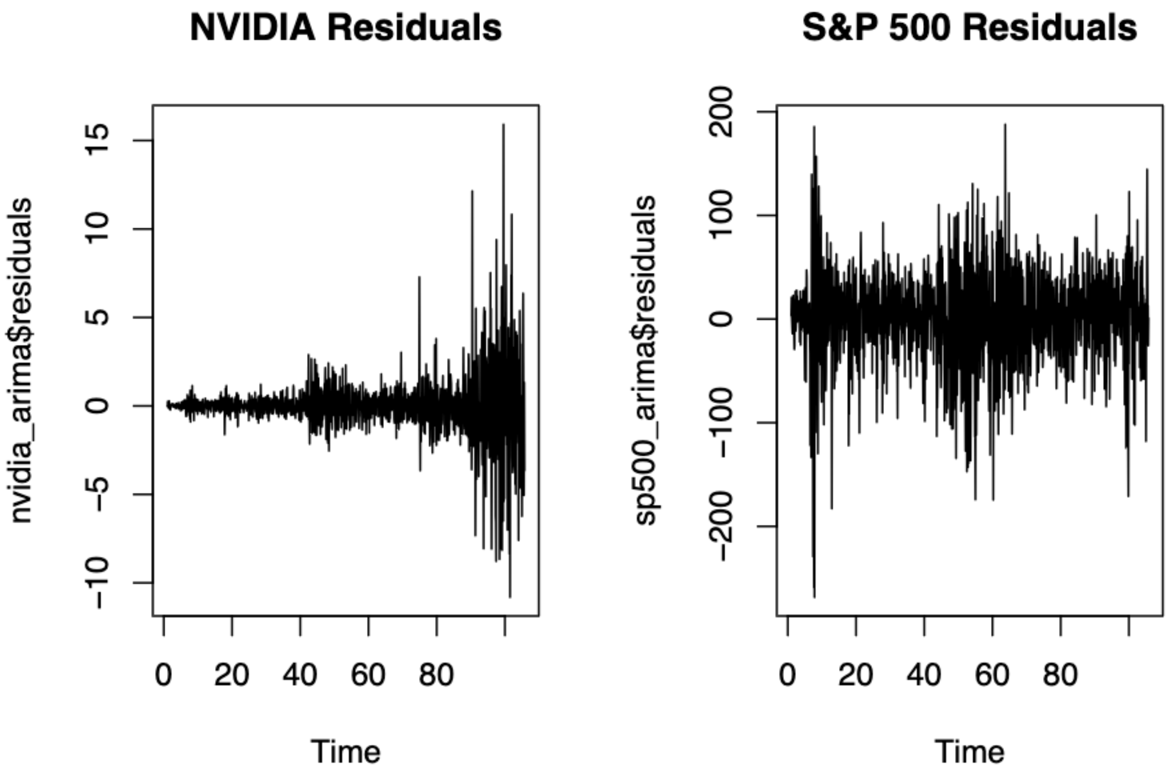
```
## Series: sp500_ts
## ARIMA(2,1,2)(0,0,1)[12] with drift
##
## Coefficients:
##           ar1      ar2     ma1     ma2     sma1   drift
##       -1.7628  -0.8837  1.6885  0.7908  -0.0097  2.3142
## s.e.   0.0386   0.0370  0.0505  0.0487   0.0294  1.2547
##
## sigma^2 = 2226:  log likelihood = -6625.27
## AIC=13264.53   AICc=13264.62   BIC=13300.49
##
## Training set error measures:
##                       ME     RMSE      MAE         MPE      MAPE      MASE
## Training set -0.01524049 47.05216 34.27405 -0.01369035 0.8665165 0.2802153
##                     ACF1
## Training set 0.01696899
```

The ARIMA(2,1,2)(0,0,2)[12] model uses two autoregressive and two moving average terms with one differencing step to handle non-seasonal patterns. The seasonal component has two moving average terms with a 12-month period, capturing yearly seasonality without extra seasonal differencing. This setup effectively models short-term dependencies and annual patterns in the data.

Residuals Analysis

I plot the residuals of each ARIMA model to evaluate the model's fit. Analyzing residual patterns helps assess if further adjustments are needed.

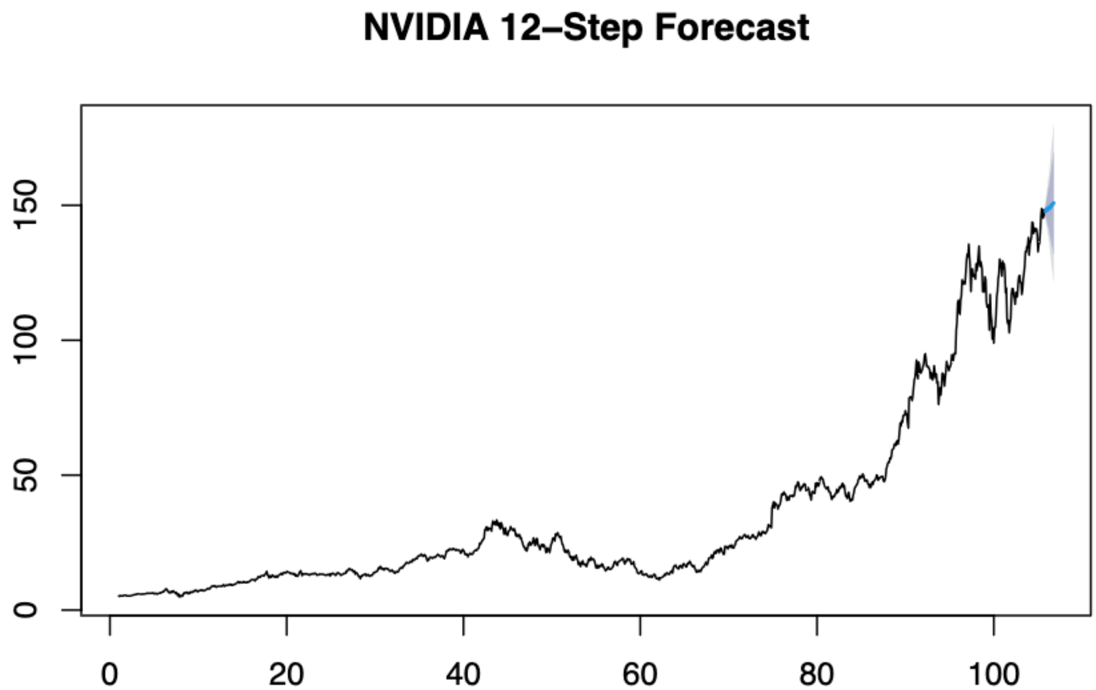
```r
# Residuals vs Fitted
par(mfrow = c(1, 2))
plot(nvidia_arima$residuals, main = "NVIDIA Residuals")
plot(sp500_arima$residuals, main = "S&P 500 Residuals")
```



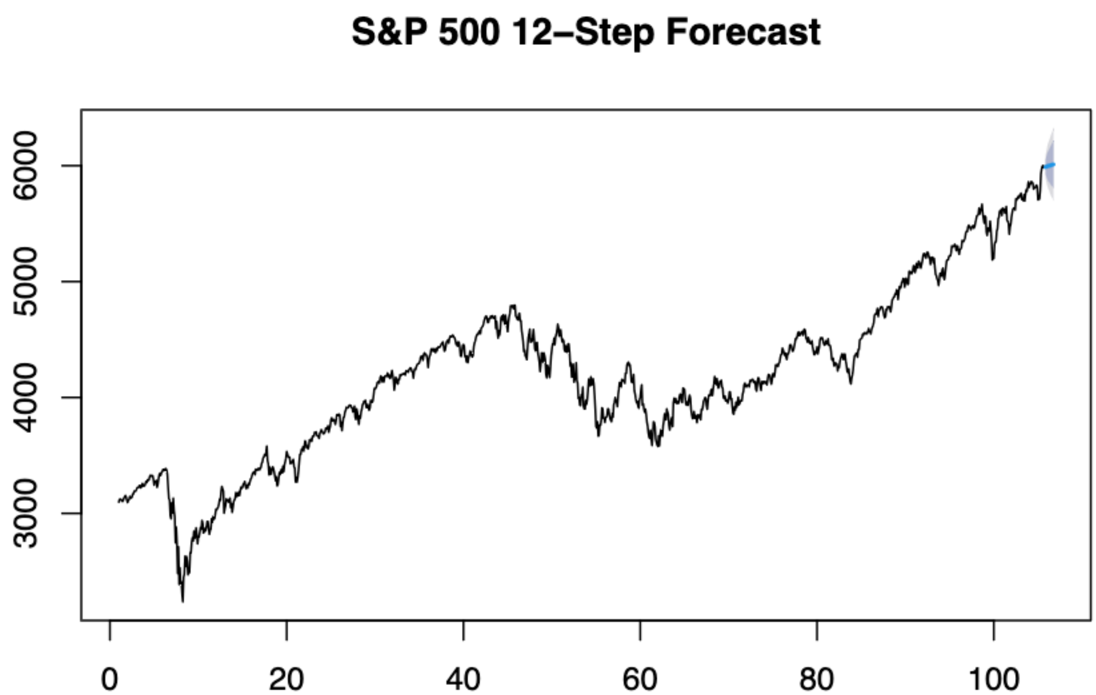
12-Step Forecast

Using the fitted ARIMA models, I forecast 12 steps ahead and plot the forecasts with error bands to visualize projected values and uncertainties.

```r
nvidia_forecast <- forecast(nvidia_arima, h = 12)
sp500_forecast <- forecast(sp500_arima, h = 12)

# Plot forecasts with error bands
plot(nvidia_forecast, main = "NVIDIA 12-Step Forecast")
```




```r
plot(sp500_forecast, main = "S&P 500 12-Step Forecast")
```



Fitting a VAR Model

I prepare the data for a VAR (Vector Autoregressive) model and fit it with lag order 2. This model allows us to analyze the relationship between NVIDIA and S&P 500 over time.

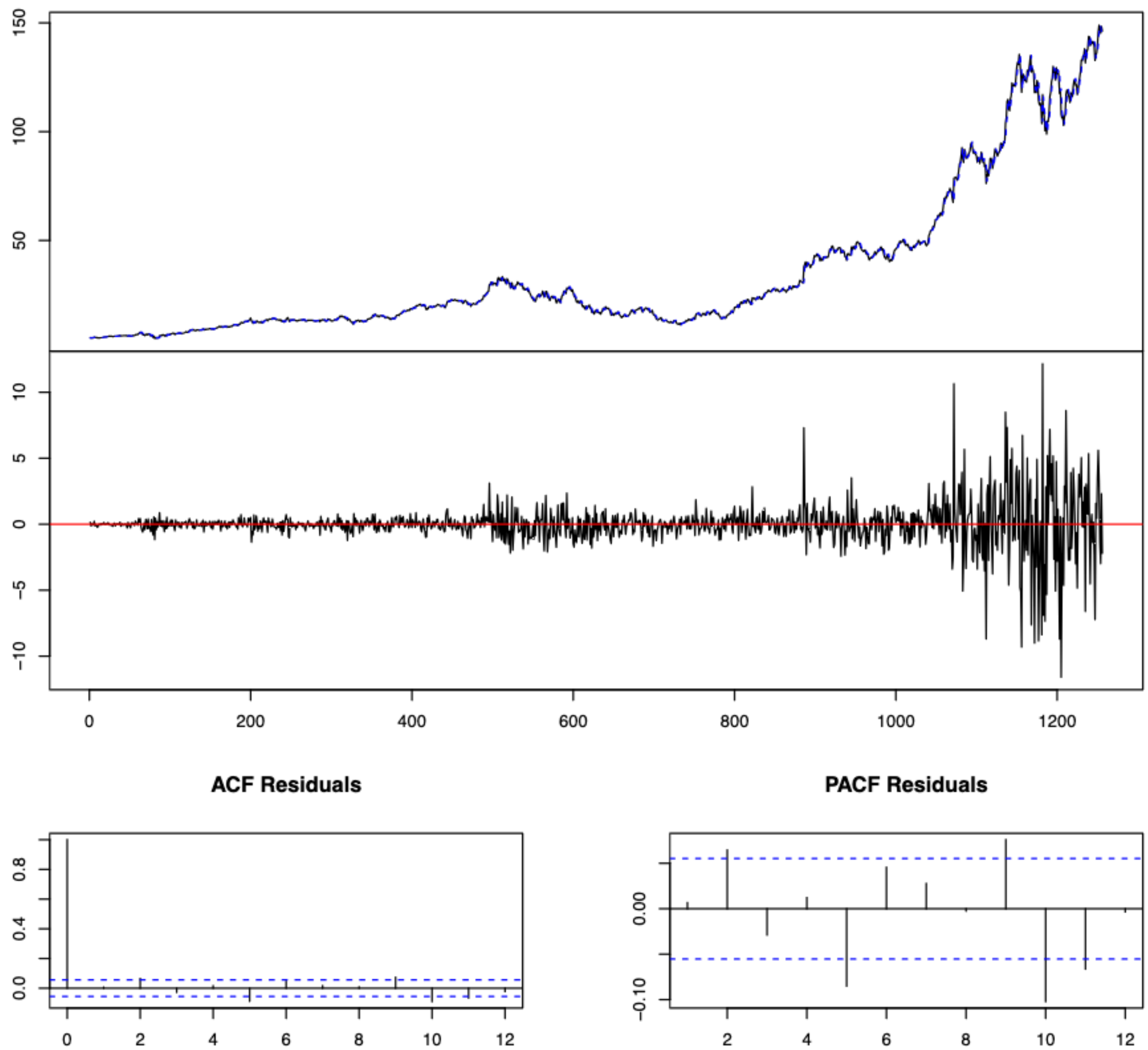
```r
# Prepare data for VAR
combined_ts <- cbind(NVIDIA = nvidia_ts, SP500 = sp500_ts)
var_model <- VAR(combined_ts, p = 2)

summary(var_model)
```

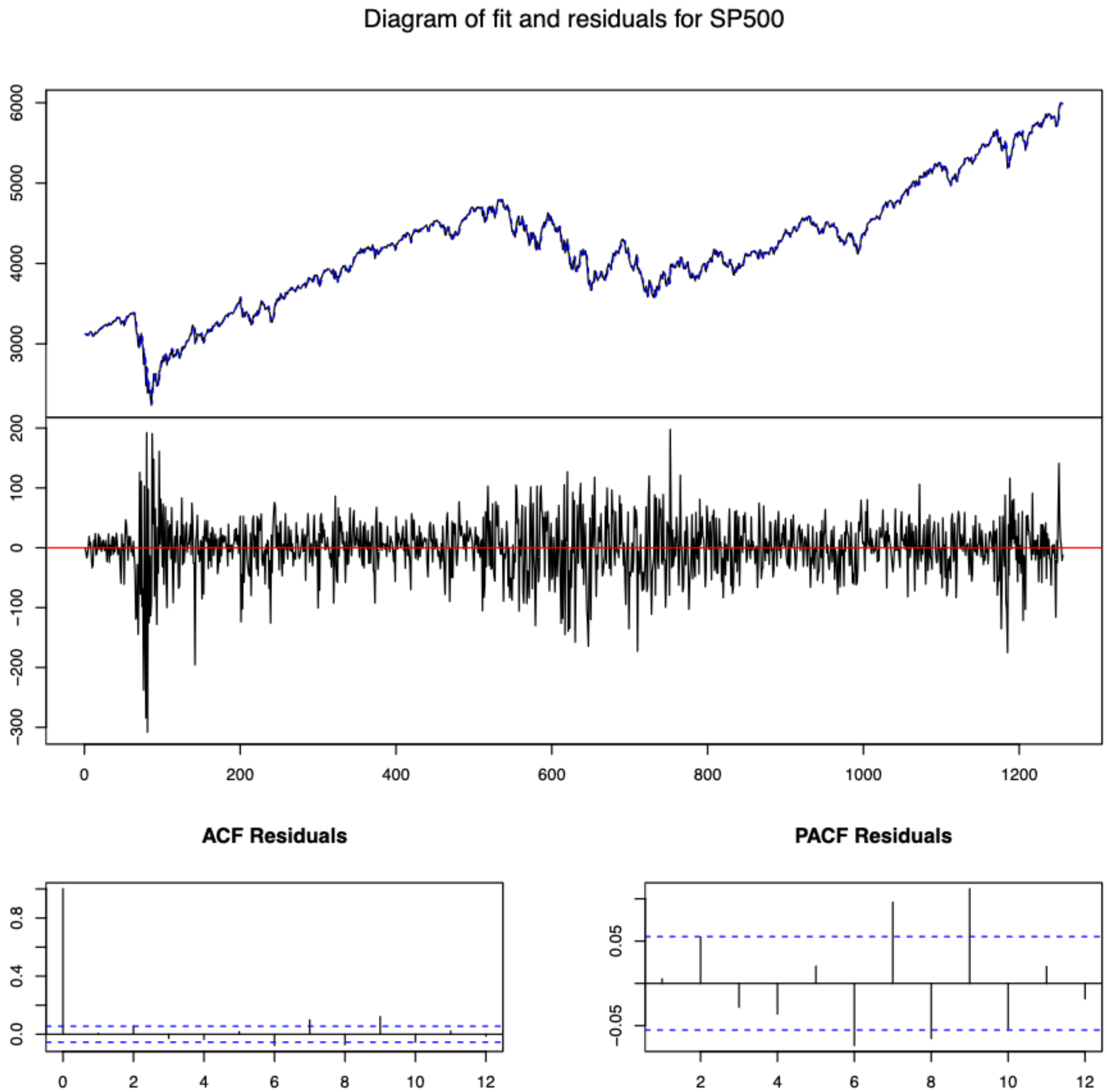
```
##
## VAR Estimation Results:
## =========================
## Endogenous variables: NVIDIA, SP500
## Deterministic variables: const
## Sample size: 1256
## Log Likelihood: -8884.376
## Roots of the characteristic polynomial:
## 1.003 0.9938 0.1264 0.08636
## Call:
## VAR(y = combined_ts, p = 2)
##
##
## Estimation results for equation NVIDIA:
```

```
## ========================================
## NVIDIA = NVIDIA.l1 + SP500.l1 + NVIDIA.l2 + SP500.l2 + const
##
##              Estimate Std. Error t value Pr(>|t|)
## NVIDIA.l1  0.9033088  0.0313667  28.798  < 2e-16 ***
## SP500.l1   0.0002608  0.0010570   0.247  0.80515
## NVIDIA.l2  0.1002450  0.0314877   3.184  0.00149 **
## SP500.l2  -0.0002823  0.0010559  -0.267  0.78925
## const      0.0887353  0.4420502   0.201  0.84094
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##
## Residual standard error: 1.617 on 1251 degrees of freedom
## Multiple R-Squared: 0.9979,  Adjusted R-squared: 0.9979
## F-statistic: 1.464e+05 on 4 and 1251 DF,  p-value: < 2.2e-16
##
##
## Estimation results for equation SP500:
## =======================================
## SP500 = NVIDIA.l1 + SP500.l1 + NVIDIA.l2 + SP500.l2 + const
##
##            Estimate Std. Error t value Pr(>|t|)
## NVIDIA.l1  1.43007    0.92953   1.538 0.124183
## SP500.l1   0.88062    0.03132  28.115  < 2e-16 ***
## NVIDIA.l2 -1.26380    0.93311  -1.354 0.175855
## SP500.l2   0.11214    0.03129   3.584 0.000352 ***
## const     26.88990   13.09982   2.053 0.040310 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##
## Residual standard error: 47.91 on 1251 degrees of freedom
## Multiple R-Squared: 0.9957,  Adjusted R-squared: 0.9957
## F-statistic: 7.313e+04 on 4 and 1251 DF,  p-value: < 2.2e-16
##
##
##
## Covariance matrix of residuals:
##        NVIDIA   SP500
## NVIDIA  2.613   34.41
## SP500  34.408 2295.09
##
## Correlation matrix of residuals:
##        NVIDIA  SP500
## NVIDIA 1.0000 0.4443
## SP500  0.4443 1.0000
```

```r
plot(var_model)
```

Diagram of fit and residuals for NVIDIA

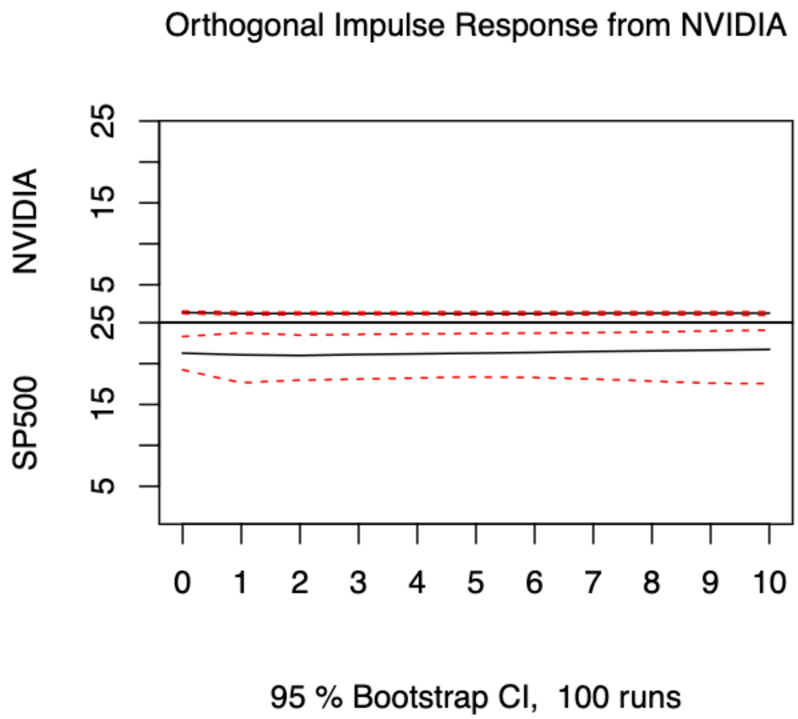




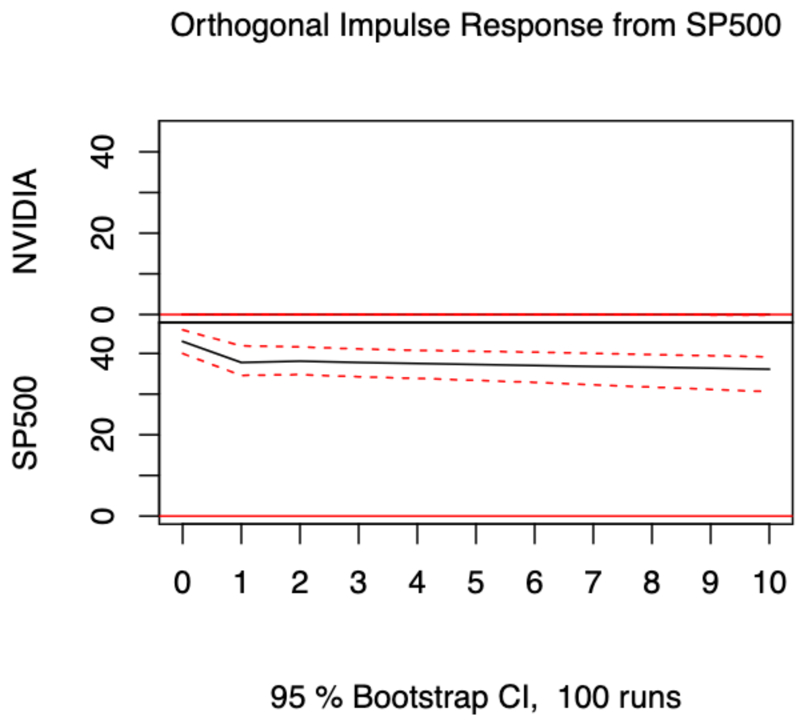
Impulse Response Function (IRF)

To understand the impact of shocks to NVIDIA on S&P 500 and vice versa, I compute and plot the impulse response functions (IRF) of the VAR model.

```
irf_results <- irf(var_model)
plot(irf_results)
```



Orthogonal Impulse Response from NVIDIA
NVIDIA
SP500
25
15
5
25
15
5
0
1
2
3
4
5
6
7
8
9
10
95 % Bootstrap CI, 100 runs



Granger Causality Test

I perform a Granger causality test to determine if changes in NVIDIA's series can predict changes in the S&P 500 series.

```r
granger_test <- causality(var_model, cause = "NVIDIA")
print(granger_test)
```

```
## $Granger
##
##  Granger causality H0: NVIDIA do not Granger-cause SP500
##
## data:  VAR object var_model
## F-Test = 3.5419, df1 = 2, df2 = 2502, p-value = 0.0291
##
##
## $Instant
##
##  H0: No instantaneous causality between: NVIDIA and SP500
##
## data:  VAR object var_model
## Chi-squared = 207.04, df = 1, p-value < 2.2e-16
```

CUSUM Test for Stability

Finally, I conduct the CUSUM test on the residuals of each series in the VAR model to check for structural stability.

```r
# Load necessary package
library(strucchange)

# Extract residuals for NVIDIA and S&P 500 from the VAR model
nvidia_resid <- residuals(var_model)[, "NVIDIA"]
sp500_resid <- residuals(var_model)[, "SP500"]

# Apply the CUSUM test on the residuals of each series
nvidia_cusum <- efp(nvidia_resid ~ 1, type = "OLS-CUSUM")
sp500_cusum <- efp(sp500_resid ~ 1, type = "OLS-CUSUM")

# Plot the CUSUM results for both residuals
par(mfrow = c(1, 2))
plot(nvidia_cusum, main = "CUSUM Test for NVIDIA Residuals")
plot(sp500_cusum, main = "CUSUM Test for S&P 500 Residuals")
```