# Assignment 4

Rachel Montgomery

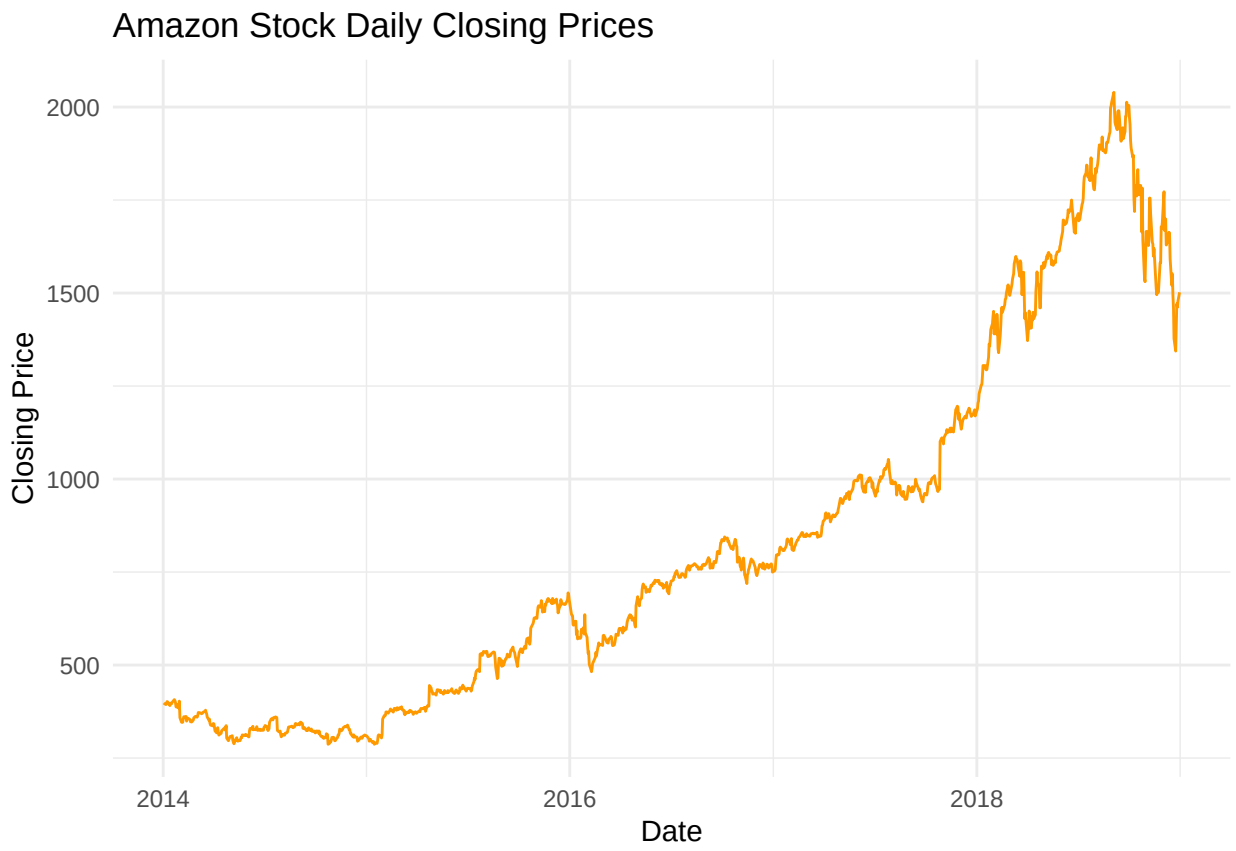
2023-10-03

## FPP3 9.11: 2 and 6

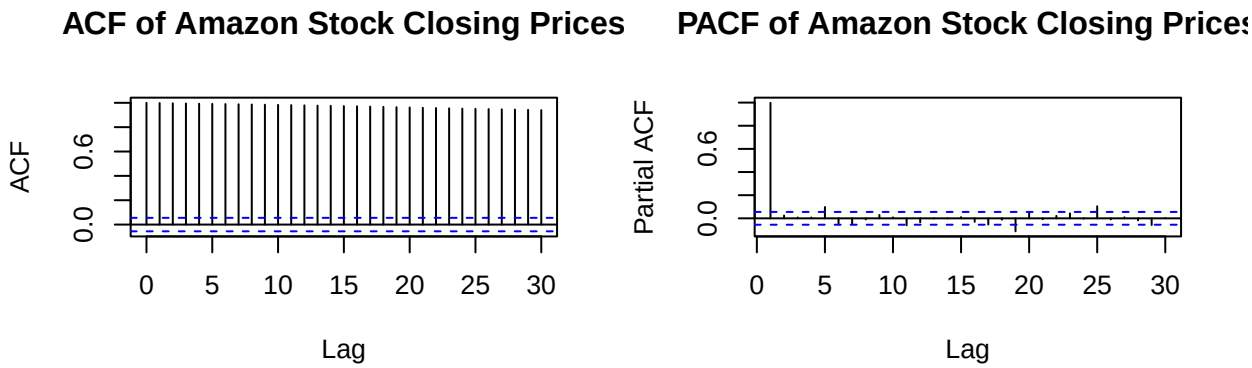
### 2. A classic example of a non-stationary series are stock prices.

**a. Plot the daily closing prices for Amazon stock (contained in `gafa_stock`), along with the ACF and PACF.**

```
# Plot time series, ACF, and PACF
## Use Amazon's stock symbol ("AMZN")

##gafa_stock is built into R

# Filter the dataset for Amazon (AMZN)
amazon_stock <- subset(gafa_stock, Symbol == 'AMZN')

# Plot the daily closing prices
ggplot(data = amazon_stock, aes(x = Date, y = Close)) +
  geom_line(color = "#ff9900") +  # Set line color to #ff9900, which is amazon's color
  labs(title = "Amazon Stock Daily Closing Prices", x = "Date", y = "Closing Price")+
  theme_minimal()
```




```r
##plotting acf and pacf
# Create a multi-panel plot
par(mfrow = c(2, 2))

# Plot the ACF of Amazon's closing prices
acf(amazon_stock$Close, lag.max = 30, main = "ACF of Amazon Stock Closing Prices")

# Plot the PACF of Amazon's closing prices
pacf(amazon_stock$Close, lag.max = 30, main = "PACF of Amazon Stock Closing Prices")
```




**b. Explain how each plot shows that the series is non-stationary.**

The closing prices plot shows a clear upward trend over time, which suggests that the series is non-stationary, and that the mean of the time series is not constant.

In the ACF plot, almost each value is significant, which indicates that the series is non-stationary.

**c. Perform KPSS unit root test and interpret the test statistic (i.e., *p*-value).**

```
# KPSS test

# amazon_stock %>%
#   features(amazon_stock$Close, unitroot_kpss()) %>%
#   as.matrix()

## that didn't work, trying tseries package now

# KPSS unit root test with tseries package
kpss_test <- kpss.test(amazon_stock$Close)
print(kpss_test)
```

```
    KPSS Test for Level Stationarity
```

```
data:  amazon_stock$Close
KPSS Level = 14.04, Truncation lag parameter = 7, p-value = 0.01
```

Answer "Interpret the test statistic (i.e., *p*-value)"

The p-value measures the probability of observing a test statistic as extreme as the one calculated (14.04) if the null hypothesis (stationarity) were true.

In this case, the p-value is 0.01, which is less than 0.05, so we reject the null hypothesis and conclude that the Amazon stock closing prices are non-stationary, indicating the presence of a trend or other non-constant characteristics in the data (which we saw the upward trend in the plot before.)

**d. Perform unit root tests to determine whether seasonal differencing or differencing are necessary.**

```
## Number of seasonal differencings
amazon_stock %>%
  features(Close, unitroot_nsdiffs) %>%
  as.matrix()
```

**Seasonal differencing should be performed before differencing (best pratice).**

```
     Symbol nsdiffs
[1,] "AMZN" "0"
```

```
## Number of differencing
amazon_stock %>%
  features(Close, unitroot_ndiffs) %>%
  as.matrix()
```

```
     Symbol ndiffs
[1,] "AMZN" "1"
```

Based on the output, seasonal differencing is not needed, and regular(non-seasonal) differencing of 1 is needed. This follows with our conclusions from the original plot.

**e. Perform KPSS unit root test with your *differenced* data and interpret the test statistic (i.e., *p*-value).**

```
## differenced data
# apply first-order differencing to the 'Close'
amazon_stock_diff <- diff(amazon_stock$Close, differences = 1)
```

```
# KPSS test

# KPSS unit root test with tseries package
kpss_test2 <- kpss.test(amazon_stock_diff)
print(kpss_test2)
```

```
    KPSS Test for Level Stationarity

data:  amazon_stock_diff
KPSS Level = 0.1495, Truncation lag parameter = 7, p-value = 0.1
```

The KPSS test indicates that after applying first-order differencing to the original data, the resulting series is level stationary!

Since our p-value is large ($0.10 > 0.05$) we fail to reject the null-hypothesis and conclude that the differenced amazon stock data is level stationary at the 0.05 significance level.

## 6. Simulate and plot some data from ARIMA models.

**Ask some generative AI to write a function that will simulate data from an ARIMA model**

**The model should have arguments for `time_points`, `constant`, `phi`, `theta`, and `sigma2`**

**Use `arima_simulate` function to generate data from an AR(1) model with $\phi_1 = 0.6$ and $\sigma^2 = 1$. The process starts with $y_1 = 0$.**

```r
# Set seed for reproducibility
set.seed(123) # Don't change
```

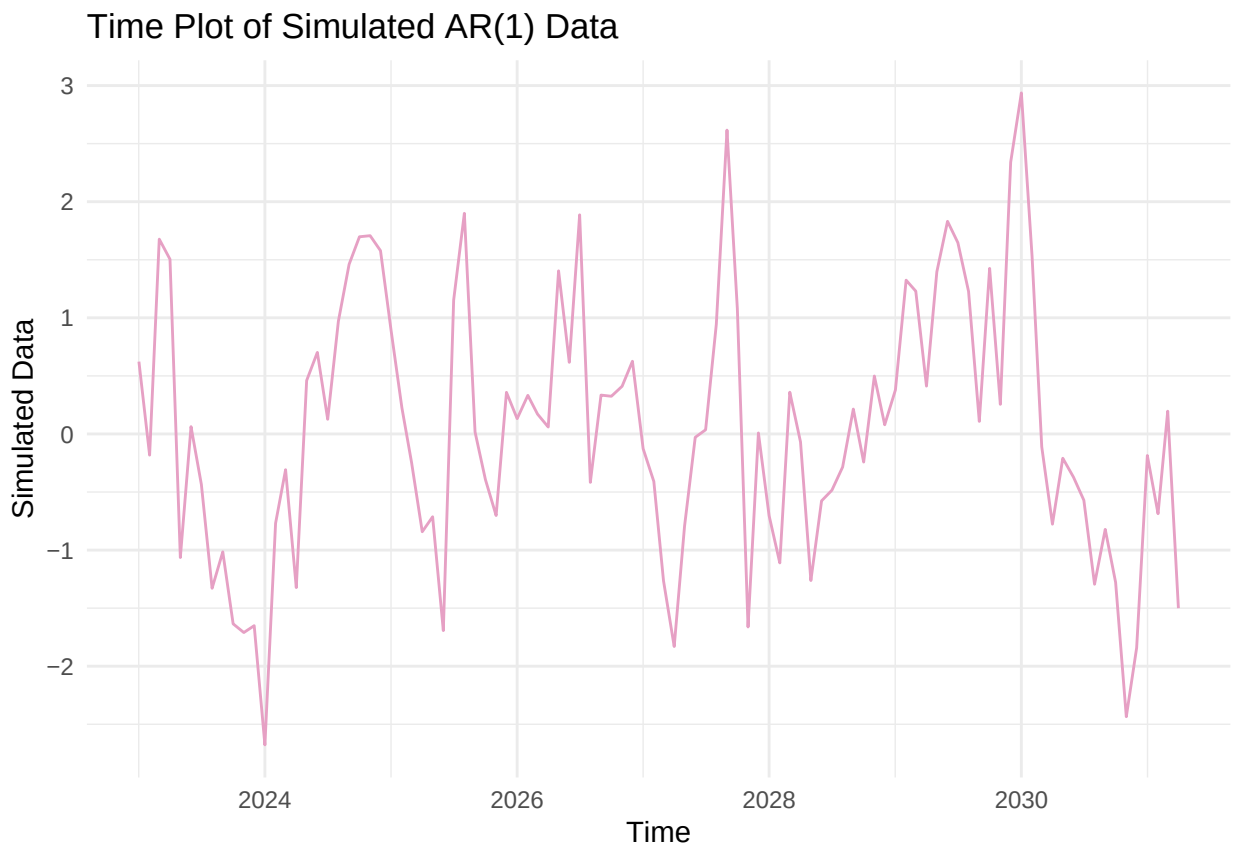
```r
## Asked Miss Chat GPT to write a function that will simulate data from an ARIMA model

simulate_arima <- function(time_points, constant = 0, phi = 0, theta = 0, sigma2 = 1) {
  # Define the ARIMA specification
  arima_order <- c(1, 0, 0)  # AR(1) model

  # Set the AR(1) coefficient and variance
  arima_params <- list(ar = phi)

  # Simulate data using arima.sim
  simulated_data <- arima.sim(model = list(order = arima_order, ar = arima_params),
                              n = time_points, innov = rnorm(time_points, mean = 0, sd = sqrt(sigma2)))

  # Add a constant if specified
  if (constant != 0) {
    simulated_data <- simulated_data + constant
  }

  # Return the simulated data
  return(simulated_data)
}
```

```r
# Use `arima_simulate` function to generate data from an AR(1) model with $\phi_1 = 0.6$ and $\sigma^2

# Define the ARIMA specification
arima_order <- c(1, 0, 0)  # AR(1) model

# Set the AR(1) coefficient and variance
```

```r
phi <- 0.6
sigma2 <- 1

# Simulate data using arima.sim
simulated_data <- arima.sim(model = list(order = arima_order, ar = phi), n = 100, sd =
                            sqrt(sigma2))
```

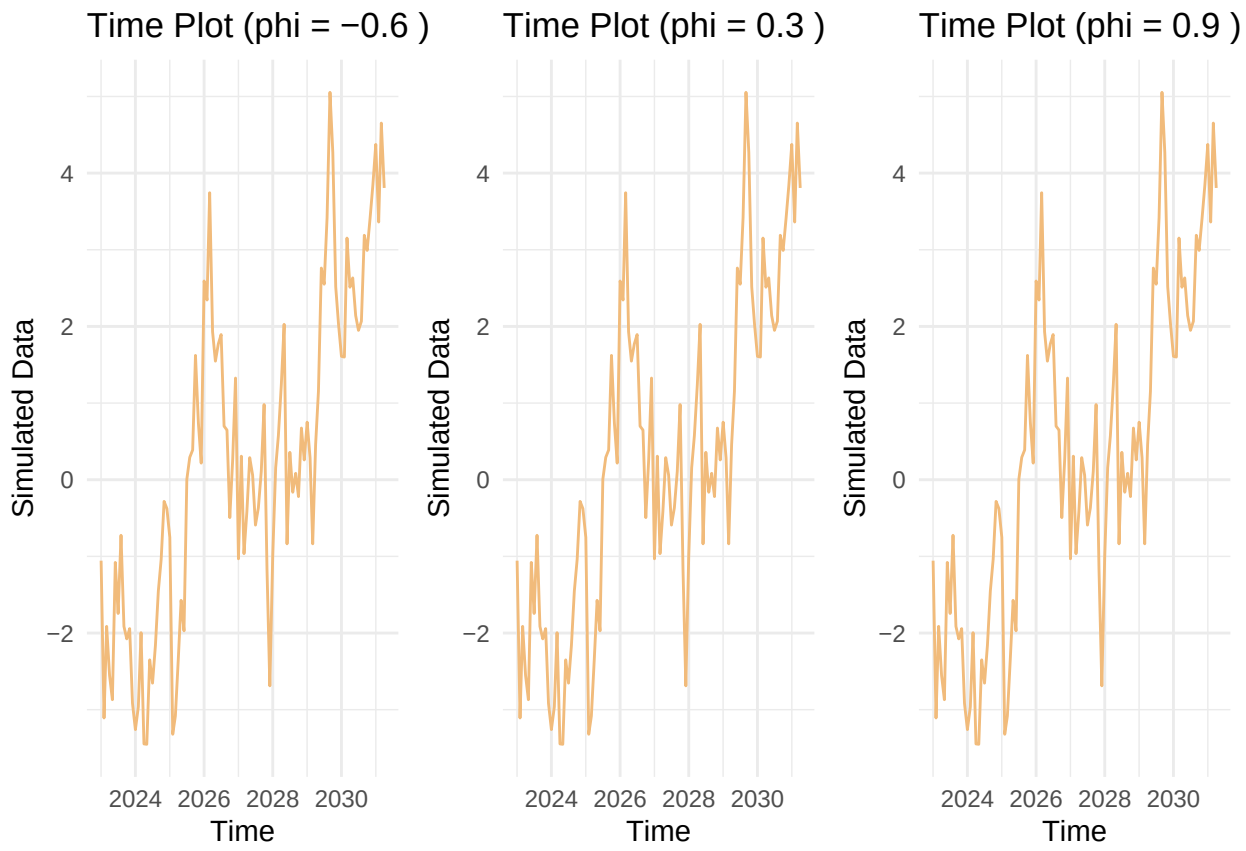
**a. Produce a time plot for the series.**

```r
# Plot original y

library(wesanderson) ## using wes anderson color palette b/c it makes me happy

# Create a data frame with time and simulated data
time_sequence <- seq(as.Date("2023-01-01"), by = "months", length.out = length(simulated_data))
data_df <- data.frame(Time = time_sequence, Simulated_Data = simulated_data)

#plotting
ggplot(data = data_df, aes(x = Time, y = Simulated_Data)) +
  geom_line(color = wes_palette("GrandBudapest2")[1]) +  #using Wes Anderson palette
  labs(x = "Time", y = "Simulated Data", title = "Time Plot of Simulated AR(1) Data") +
  theme_minimal()
```



**b. Plot three other $\phi$ values (use some negative and positive values)**

```r
# Values of phi to explore
phi_values <- c(-0.6, 0.3, 0.9)

# Create a data frame with time and simulated data
data_df <- data.frame(Time = time_sequence)

# Create and save time plots for different phi values
for (phi in phi_values) {
  simulated_data <- arima.sim(model = list(order = arima_order, ar = phi), n = 100, sd = sqrt(sigma2))
  data_df[[paste0("Simulated_Data_", phi)]] <- simulated_data
}

# Create a separate time plot for each phi value using ggplot2
plots <- list()
for (phi in phi_values) {
  plot <- ggplot(data = data_df, aes(x = Time, y = data_df[[paste0("Simulated_Data_", phi)]])) +
    geom_line(color = wes_palette("GrandBudapest1")[1]) +  # Set color using Wes Anderson palette
    labs(x = "Time", y = "Simulated Data", title = paste("Time Plot (phi =", phi, ")")) +
    theme_minimal()
  plots[[as.character(phi)]] <- plot
}

# Combine and display the time plots using grid.arrange
grid.arrange(grobs = plots, ncol = 3)
```




What does `phi` seem to affect in the data generating process?

In theory, the negative values of phi should lead to oscillations or mean-reverting behavior around a central value, positive values of phi should lead to a gradual upward or downward trend, and values that are approximately zero should be stationary.

In the plots of the phi values explored, the plots all appear relatively similar. The phi value of -0.6 has somewhat more of a dramatic downward trend, but it isn't radically different than the others.

**c. Simulate ARIMA data using the best fitting parameters of the Egyptian exports data. Discuss the similarities and differences between the parameters.**

```
## Obtain the Egyptian exports data
actual_data <- global_economy %>%
  filter(Code == "EGY")

## find best fitting arima model
auto.arima(actual_data$Exports, trace=TRUE) ##forecast package auto.arima function
```

```
 ARIMA(2,0,2) with non-zero mean : 296.2322
 ARIMA(0,0,0) with non-zero mean : 369.069
 ARIMA(1,0,0) with non-zero mean : 300.9472
 ARIMA(0,0,1) with non-zero mean : 321.8672
```

```
 ARIMA(0,0,0) with zero mean     : 517.6216
 ARIMA(1,0,2) with non-zero mean : 301.047
 ARIMA(2,0,1) with non-zero mean : 294.2861
 ARIMA(1,0,1) with non-zero mean : 298.9197
 ARIMA(2,0,0) with non-zero mean : 297.6059
 ARIMA(3,0,1) with non-zero mean : 302.1247
 ARIMA(3,0,0) with non-zero mean : 298.3894
 ARIMA(3,0,2) with non-zero mean : 298.5836
 ARIMA(2,0,1) with zero mean     : 308.4412

 Best model: ARIMA(2,0,1) with non-zero mean

Series: actual_data$Exports
ARIMA(2,0,1) with non-zero mean

Coefficients:
         ar1      ar2      ma1     mean
      1.6764  -0.8034  -0.6896  20.1790
s.e.  0.1111   0.0928   0.1492   0.9142

sigma^2 = 8.046:  log likelihood = -141.57
AIC=293.13   AICc=294.29   BIC=303.43
```

```
##best model is ARIMA(2,0,2) with non-zero mean : 296.2322

## Model the data with the best fit

# Fit ARIMA(2,0,2) model
fit <- actual_data %>%
  mutate(diff_export = difference(Exports)) %>%
  na.omit() %>%
  model(arima_best = ARIMA(diff_export ~ PDQ(2,0,2)))

# Report fit
glance(fit) %>%
  select(AIC, AICc, BIC)
```

```
# A tibble: 1 x 3
    AIC  AICc   BIC
  <dbl> <dbl> <dbl>
1  277.  277.  279.
```

The AIC, AICc, and BIC are all similar to eachother. The main difference between the critera is how they penalize complexity and their sensitivity to sample size, so because there are small differences between the critera, we can infer that the model is a good fit.

```
# Set seed for reproducibility
set.seed(1234) # Don't change

# Simulate ARIMA data using the same parameters as the Egyptian export data
## You will need to set all parameters!
## time_points = nrow(actual_data)
## constant = one value
```

```r
## phi = two values!
## theta = one value
## sigma2 = one value


## defining ARIMA parameters
arima_params <- list(
  order = c(2, 0, 1),  # ARIMA(2,0,1) order
  ar = c(0.2, 0.3),    # Autoregressive coefficients (phi1 and phi2) 2 total!!
  ma = -0.4,           # Moving average coefficient (theta)
  sd = sqrt(0.1)       # Standard deviation (sigma)
)

# Simulate ARIMA data using the specified parameters
simulated_data <- arima.sim(model = arima_params, n = nrow(actual_data))

# Fit ARIMA(2,0,1) model
arima_model <- Arima(simulated_data, order = c(2, 0, 1))

# Report fit
summary(arima_model)
```

```
Series: simulated_data
ARIMA(2,0,1) with non-zero mean

Coefficients:
         ar1     ar2      ma1     mean
      0.4051  0.4412  -0.5551  -0.0933
s.e.  0.1927  0.1311   0.1985   0.3604

sigma^2 = 1.094:  log likelihood = -83.17
AIC=176.34   AICc=177.49   BIC=186.64

Training set error measures:
                     ME     RMSE       MAE      MPE     MAPE      MASE
Training set -0.0341576 1.009421 0.8133641 55.98996 149.5742 0.6271061
                     ACF1
Training set -0.005931065
```

The AIC, AICc, and BIC are similar, which suggests a good fit.

**d. Using your data that's supposed to be similar to the Egyptian exports data, check for whether the data are stationary using the KPSS test. If not, then report on whether seasonal differencing and/or differencing are necessary.**

```r
# KPSS

# KPSS unit root test with tseries package
kpss_test <- kpss.test(simulated_data)
print(kpss_test)
```

```
    KPSS Test for Level Stationarity

data:  simulated_data
KPSS Level = 0.52573, Truncation lag parameter = 3, p-value = 0.03587
```

Because our p-value (0.03587) is less than alpha (0.05), we reject the null hypothesis and conclude that our simulated data is not level stationary. Differencing is most likely necessary, and we can use the nsdiffs function to determine what differencing is needed.