NAME
NATHAN CLARKE
STUDENT NUMBER
20387306

ECONOMETRICS OF FINANCIAL MARKETS

ARIMA FORECASTING

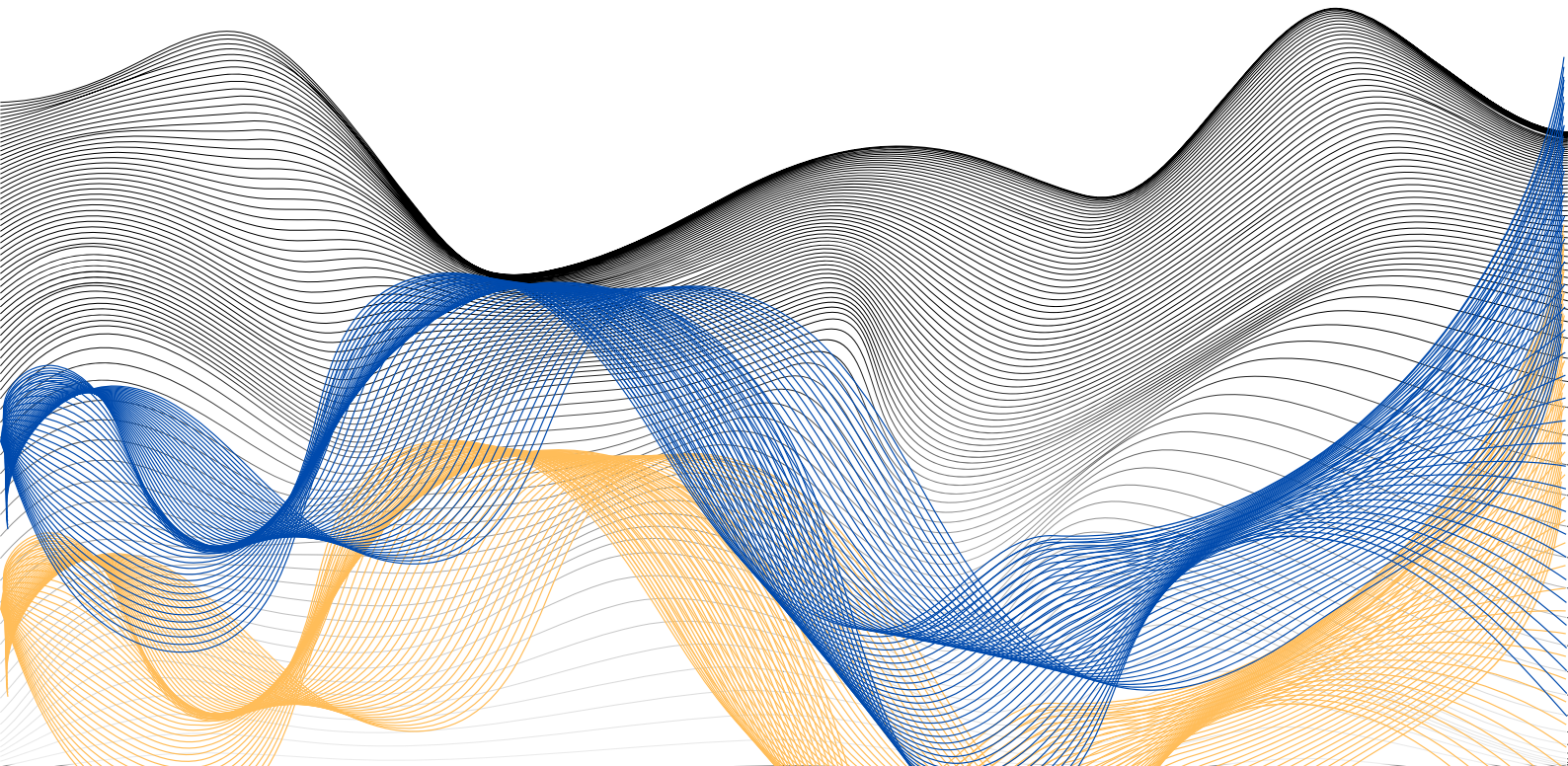


Table of Contents

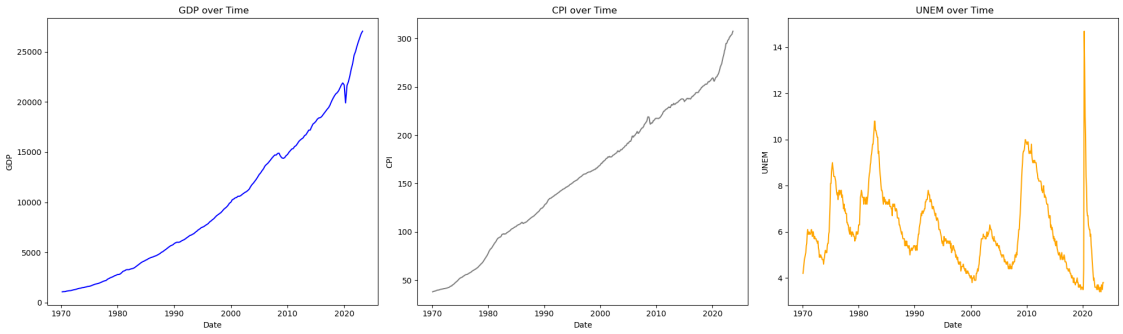
In this project, I will explore the theory and application of ARIMA forecasting by applying it to three Economic Indicators: GDP, CPI and Unemployment. I will use Python to carry out the forecasting, produce my visualisations and generate summary tables. I will then compile my results, discussion and analysis using Latex. All economic data is gathered using the FRED API.

Forecasting

November 5, 2023

1 Question A

For the first part of my assignment I will attempt to guess the next two values for GDP, CPI and Unemployment. In order to do this I have presented a plot of the time series of each economic indicator over time. I will guess the values by looking at the plots and motivating my answers with Macro Economic outlook.



1.1 Visual Trends

GDP

As you can see from the plot above GDP has been increasing steadily since the start of my dataset. While there have been a few dips around recessionary times such as 2008 and 2020 it has quickly recovered. Since the dip of 2020 GDP has been on a steep inclined trajectory. Barring any changes in the macroeconomic landscape, I would expected this trend to increase.

CPI

CPI is a metric used to determine inflation (price increases) over time. Historically it has followed a similar trajectory to GDP. There have been a few dips in the time series, such as around 2008 but more recetly CPI has experienced a sharp increase relating to the current period of high inflation. Given the increased volatility of CPI over the last few years it is difficult to guess what the next levels might be, so below I will use some macroeconomic factors to inform my guess.

Unemployment

Unemployment is a much more cyclical and volatile time series than GDP and CPI. Over the last 50 years it has gone through periods of sharp increase and sharp decline. More recently there was an extremely sharp increase and decline around the COVID pandemic in 2020, however it seems to have stabalised since that period. Given the volatitly of this time series it is more difficult to guess what it might do next, so below I will use some macroeconomic factors to inform my guess.

1.2 Marcoeconomic Landscape

GDP
Earlier this year there was increased risk of recession, however in recent months the overall sentiment of the Macro Economy has become more positive. In Q2 this year GDP increased by 2.1%, this was mainly driven by increases in international trade, personal spending and investment. As the rates environment also begins to stabalise I expect to see more postive growth in GDP going forward.
CPI
Over the past months, the Federal Reserve revised its inital Q4 inflation estimate down to 3.7%. This suggests a general consensus of inflationary cooling over the coming months. Therefore it seems that inflation has hit its short term peak and is expected to stabalise. Therefore I expect a small decrease in CPI over the next two periods.
Unemployment
As there is a strong Macro Economic link between inflation and unemployment, I would expect the general sentiment around inflation to mirrored in the labour market. Given that the labour market has been out-performing predictions of higher unemployment and remaining steady, I expect that as inflation cools Unemployment should stay pretty constant around its current levels (near the full capacity of employment).

1.3 Estimates

Period	GDP	Period	CPI	Period	UNEM
Q1 2023	26813.60	Aug-2023	306.27	Aug-2023	3.8%
Q2 2023	27063.01	Sep-2023	307.48	Sep-2023	3.8%
Q3 2023	27312.42	Oct-2023	307.24	Oct-2023	3.8%
Q4 2023	27561.83	Nov-2023	307.00	Nov-2023	3.8%

2 Question B

2.1 Intro

ARIMA forecasting is a popular technique used in time series analysis. An ARIMA model is composed of:
1. Autoregressive Process with parameter p (AR(p))

$$Y_t = \phi_1 Y_{t-1} + \phi_2 Y_{t-2} + ... + \phi_p Y_{t-p} + \epsilon_t$$

2. Moving Average Process with parameter q (MA(q))

$$Y_t = \epsilon_t + \theta_1 \epsilon_{t-1} + ... + \theta_q \epsilon_{t-q}$$

3. Integrated Component with parameter d, representing the order of differencing required to make the time series stationary.
Note: In the equations above ϵ_t represents a white noise process:

$$\epsilon_t \sim^{\text{iid}} N(0, \sigma_2)$$

An ARIMA process is a combination of AR and MA processes such that an ARIMA(p,d,q) process is defined as

$$Y_t = \phi_1 Y_{t-1} + \phi_2 Y_{t-2} + ... + \phi_p Y_{t-p} + \epsilon_t + \theta_1 \epsilon_{t-1} + ... + \theta_q \epsilon_{t-q}$$

Where the series $\{Y_t\}$ has been differenced at order d.
To estimate the possible orders for ARIMA models of GDP, CPI and Unemployment, I will first check if the time series are stationary (see below). If they are not, I will use differencing and detrending to remove

any trends; the level of differencing I will use will then be the parameter d of the ARIMA models. Once my time series' are stationary, I will follow the Box-Jenkins Framework to estimate the parameters of the AR and MA processes, p and q.

2.2 Model Estimation

2.2.1 Stationarity

What is Stationarity?
The definition of stationarity comes in two forms: strong stationarity and weak stationary. These are defined as follows:
Strong Stationarity:
A sequence of Random Variables $\{X_t\}$ is said to be strongly stationary if:

$$(X_0, X_1, ..., X_n) =^d (X_h, X_{1+h}, ..., X_{n+h})$$

$\forall h \in \mathbb{N}_0$
That is that the joint distribution of the variables equals the joint distribution of the variables shifted on in time by some value h.

Weak Stationarity:
A sequence of Random Variables $\{X_t\}$ is said to be weakly stationary if:
1. $E[X_{t+h}] = E[X_t]$
2. $Var(X_{t+h}) = Var(X_t)$
3. $Cov(X_{s+h}, X_{t+h}) = Cov(X_s, X_t) = \gamma(h)$
The mean, variance, and autocovariance do not depend on time. Covariance depends only on the lag h.

As I mentioned before, examining if a time series is stationary can be done by looking at the plot of the time series and checking for mean-reversion and time-independent variance. Looking back at the plots from part A, you can see that the time series plots of the three economic indicators do not appear to have constant mean and variance across time. Therefore, as I mentioned above, there may be trends present.

What kind of trends could be present in a time series, and how can they be removed?
A non-stationary time series can have three different forms of trends, these are:
1. Trend in Mean
2. Trend in Variance
3. Trend in Mean and Variance

Trend in Mean:
A trend in mean results in the mean or expectation of a time series being dependent on time. A common form of a trend in the mean is the relation:

$$Y_t = \alpha + \beta t + \epsilon_t$$

Where α and β are constants, t is time, and ϵ_t is the white noise term.
This is known as a linear trend and results in the mean of the time series increasing linearly with time ($E[Y_t] = \alpha + \beta t$).
Another example of a trend in the mean is a quadratic trend:

$$Y_t = \alpha + \beta t^2 + \epsilon_t$$

This example has the same parameters as my example of a linear trend; however, now, the mean increases quadratically with time($E[Y_t] = \alpha + \beta t^2$).
It is important to note that the trends I have mentioned so far only affect the mean and not the variance. Notice that the variance of Y_t in these examples does not depend on t:

$$Var(Y_t) = Var(\alpha + \beta t^2 + \epsilon_t) = Var(\alpha + \beta t + \epsilon_t) = Var(\epsilon_t) = \sigma_\epsilon^2$$

To remove a trend in the mean, we need to use regression; the residuals of this regression will have the trend removed.

Trend in Variance:
A trend in variance results in the variance of a time series being dependent on time. A common form of a process with a trend in variance is:

$$Y_t = Y_{t-1} + \epsilon_t$$

This is also known as a Random Walk or a Stochastic Trend.
In this process the value of Y_t is determined by the value of Y_{t-1} and the white noise ϵ_t.
To examine the variance of this process, it must first be rearranged using iteration into:

$$Y_t = Y_0 + \sum_{i=0}^{t-1} \epsilon_{t-i}$$

As you can see below, this process has a constant mean, but its variance depends on t:

$$E[Y_t] = E[Y_0 + \sum_{i=0}^{t-1} \epsilon_{t-i}] = Y_0$$

$$Var(Y_t) = Var(Y_0 + \sum_{i=0}^{t-1} \epsilon_{t-i}) = Var(\sum_{i=0}^{t-1} \epsilon_{t-i}) = t\sigma_\epsilon^2$$

To remove the trend in variance, we can use differencing of the time series ($\Delta_d Y_t = Y_t - Y_{t-d} = \epsilon_t$)

Trend in Mean and Variance:
A trend in mean and variance occurs when you have a combination of the components mentioned above in a model such that there is a trend present in both the mean and the variance
To remove both of the trends, both of the processes used in the previous parts must be used. Differencing can be used to remove the trend in variance, followed by a regression to remove the trend in the mean.

So, as you can see, the approach to removing trends depends on the type of trend or trends present in the time series. However, the particular type of trend is difficult to spot visually (In all of these cases, the ACF just displays slow decay). So, to check for these trends, I will use a statistical test known as the Dickey-Fuller test.

Dickey-Fuller test
This test is used to detect if a process is stationary or not, it can also be used to determine if there is a trend in mean, trend in variance or trend in mean and variance.
The Dickey-Fuller Test assumes an AR(1) process for our data ($Y_t = \phi Y_{t-1} + \epsilon_t$) (auxillary regression). If our process is non-stationary, then it will have $\phi = 1$, otherwise our data is considered weakly stationary
So we can use the following hypothesis test to check if our time series' are stationary:
$H_0 : \phi = 1$
$H_1 : \phi \neq 1$
To test these hypothesis' a test statistic must be used, so to check if $\phi = 1$, we can define a test statistic t:
$t = \frac{\hat{\phi}-1}{SE(\hat{\phi})}$
This is known as the Dickey-Fuller Test Statistic and has a t-distribution. However, since we will be working with non-stationary data, critical values must be drawn from a Dickey-Fuller Critical Value table.
The test I explained above does not test if there is a trend in mean; to test for this, another form of the auxiliary regression must be assumed.
We can assume the form of a random walk with drift. As I mentioned above, a random walk with drift has both a trend in mean and a trend in variance. So, by assuming this auxiliary regression, the Dickey-Fuller Test can be used to check for both trends in mean and the trend in variance.

First, I will check if our time series' are already stationary (no unit root) using the Dickey-Fuller test. However, like I mentioned above, due to the visible change in variance across time and no mean-reversion, I believe that they are non-stationary to begin with.

	GDP	CPI	UNEM
P-Value	1.0	0.999924	0.339368

As you can see from the table of p-values above, none of my time series have a statistically significant Dickey-Fuller Test statistic. So I fail to reject the null hypothesis and conclude that there is non-stationarity present in the time series of my three economic indicators.

Differencing

Like I mentioned above I will attempt to remove trends from my time series' by first using differencing to remove trend in variance and then regression to remove any remaining trend in mean. To find what order of differencing is needed to make each time series stationary, I will try first order differencing, then check my results using the Dickey-Fuller test. If the time series still displays a trend in variance, I will try second-order differencing and so on. Once I am happy that the trend in variance has been removed I will use regression to remove any remaining trend in mean, resulting in a fully stationary time series for each economic indicator. Following this process I will know my parameter d for my ARIMA models for each time series (d will be the order of differencing used).

Using python I have carried out first order differencing on each of my time series'. Once they were differenced I carried out the Dickey-Fuller Test on each one and once again presented the p-values in the table below.

	GDP	CPI	UNEM
Differenced P-Value	0.016356	0.065782	1.851872e-25

As you can see from the p-values I now get for GDP and Unemployment are statistically significant (** for GDP *** for Unemployment) Dickey-Fuller Statistics for each of my time series. Therefore I reject the null hypothesis that there is a unit root in favour of the alternative hypothesis of stationarity. However for CPI I see that it is only significant at a 10 percent level, this indicates that there may be a trend component to be removed. In order to check if there is still a trend in mean to be removed I have carried out the Augmented Dickey-Fuller Test for all possible trend components and presented the resulting p-values in the table below.

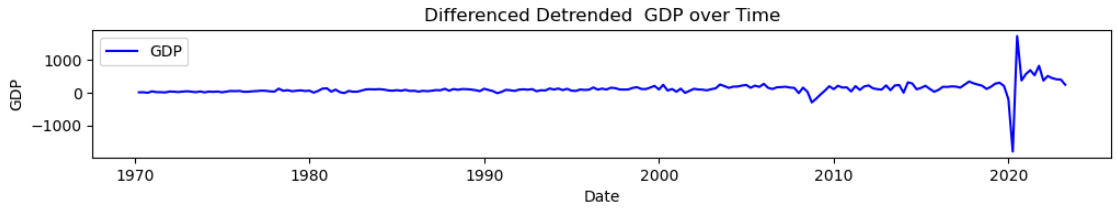
	GDP	CPI	UNEM
n	0.01636	0.06578	0.0
c	0.00130	0.00263	0.0
ct	0.00000	0.00844	0.0
ctt	0.00000	0.01280	0.0

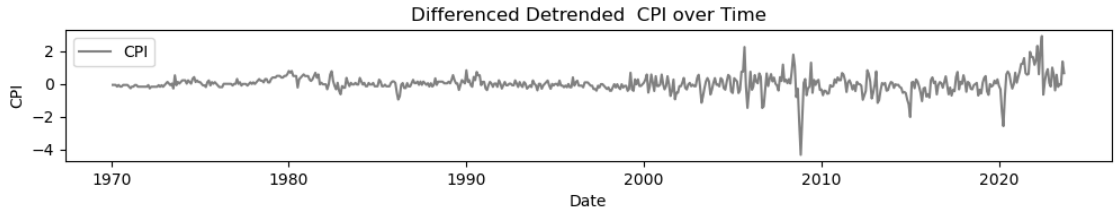
Since I have already concluded that GDP and Unemployment are stationary following differencing I am not surprised to see that it is also stationary for all levels of trends. However I can see that by adding a

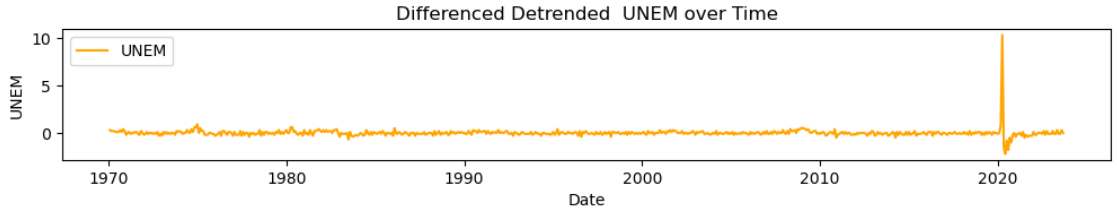
constant trend to CPI the time series can be made stationary at a 1 percent level of significance. Therefore I removed this trend and presented the final p-values below.

	GDP	CPI	UNEM
Differenced Detrended P-Value	0.016356	0.000067	1.851872e-25

I have plotted the transformed time series' below and as you can see they are now mean-reverting and appear to have constant variance. So I am happy to conclude that each of my time series are now stationary. I have discovered that each of my ARIMA models for each economic indicator should have their integrated parameter as d=1. I will now continue with the Box-Jenkins Methodology to estimate the order of the AR and MA parameters (p and q) my models.







2.2.2 ACF and PACF Inspection

In order to determine the parameters p and q we must inspect the Autocorrelation and Partial Autocorrelation functions for each of the time series'. Below I have introduced these functions:

Autocorrelation Function
In a stationary time series the Autocorrelation function or ACF is defined as:

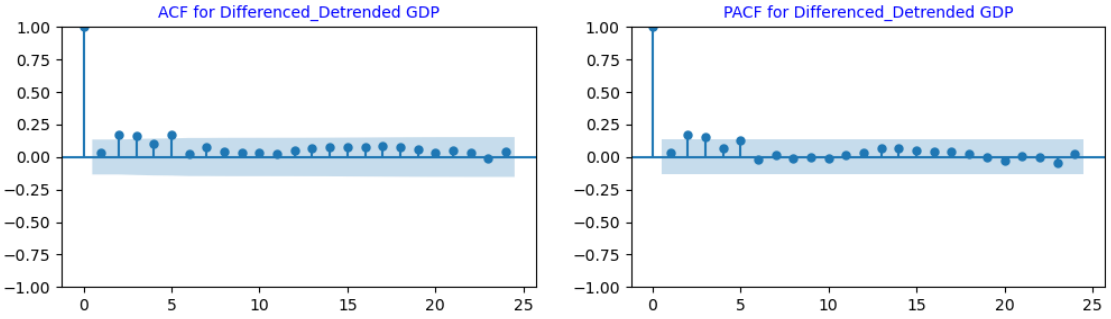
$$\rho_k = Corr(Y_{t+h}, Y_t) = \frac{Cov(Y_{t+h}, Y_t)}{\sqrt{Var(Y_{t+h})Var(Y_t)}} = \frac{Cov(Y_{t+h}, Y_t)}{VarY_t}$$

This function measures the correlation between the original time series and its lagged values (lagged by value h). It is helpful for identifying dependancies and relationships at different lags within a dataset

Partial Autocorrelation Function
$\theta_k = Corr(Y_{t+h}, Y_t | Y_{t+1}, ..., Y_{t+h-1})$
The PACF is similar to the ACF but it takes into account and removes the influence of shorter lags, providing a clearer view of the time series with its lagged values.

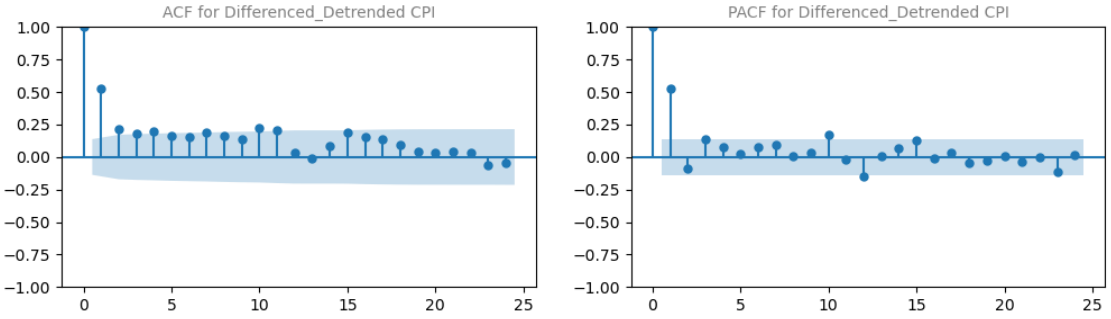
By inspecting the plots of these two functions, it is possible to identify different values for p and q in our ARIMA model for each time series.
More specifically, if the ACF plot of our stationary time series displays statistically significant autocorrelation at a certain lag, this is an indication that the order of the MA process (q) is the same as the number of the significant lag.
Similarly, by identifying statistically significant lags on the PACF it is possible to determine the order of the AR process (p).
Below, I have displayed the Correlogram (ACF and PCF plots) for each of the economic indicators. From these plots I will attempt to identify lags with statistically significant autocorrelation. Each of the ACF and PACF plots have a blue shaded area, indicating the region where I would fail to reject the null hypothesis that there is no autocorrelation or partial autocorrelation present in the respective plots. Therefore any lag with a value outside of the blue zone is considered statistically significant for autocorrelation/partial autocorrelation.
Note:
In the below section, there are a few lags which I could not distinguish between being inside or outside of the rejection region. So, I decided to include any lags that I was unsure of, knowing that if the parameter is not a good choice, its corresponding models will not be selected at the model selection step.

GDP



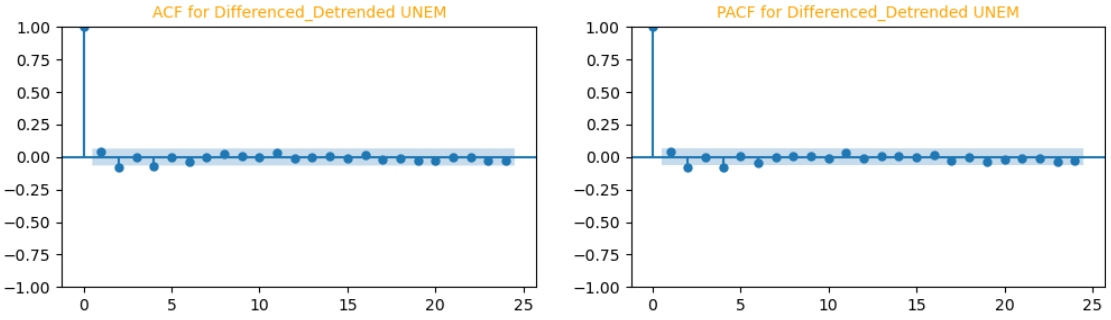
So, as you can see from the Correlogram, there are a few lags in our ACF and PACF, which appear to be statistically significant. Lags 2,3 and 5 on the ACF and lags 2 and 3 on the PACF appear to be significant. This indicates that any combination of these values for p and q, respectively, could lead to a useful ARIMA model. I will examine all of these models at the model selection step (the next step according to the Box-Jenkins Methodology).

CPI



So as you can see from the CPI Correlogram, there are a few lags on our ACF and PACF which appear to be statistically significant. Lags 1,2,3,4,10 and 11 on the ACF and lags 1,3 and 10 on the PACF appear to be significant.

Unemployment



As you can see from the Unemployemt Correlogram, there are a few lags on our ACF and PACF, which appear to be statistically significant; while they are not as pronounced as lags of GDP and CPI, they still take values outside of the rejection region. Lags 2 and 4 on the ACF and lags 2 and 4 on the PACF appear to be significant.

So follow inspection of the ACF and PACF I have identified the orders of possible models for each of my time series'. Next I will generate these models in Python and present some summary statistics in order to aid the model selection process.

2.2.3 Model Selection

In this section, I will generate all possible models given the significant lags found in the previous section and present a summary table to decide which model is best for each time series.

The objective of model selection is to choose a model which minimises the value of some statistical measures known as information criterion. These measures are used to choose the most appropriate model from various possible models. The primary purpose of information criteria is to find a balance between the fit and complexity of the model. They aim to maximise the fit with minimal complexity, as complexity may lead to overfitting and poor out-of-sample results. Below, I have provided the formula for three information criteria I will use for model selection.

AIC (Akaike Information Criterion):

$$T * ln(RSS) + 2n$$

BIC (Bayesian Information Criterion):

$$T * ln(RSS) + n * ln(T)$$

HQIC (Hannan-Quinn Information Criterion):

$$T * ln(RSS) + 2n * ln(ln(T))$$

Where:
T: Number of Observations
n: Number of Estimated Parameters
RSS: Residual Sum of Squares ($\sum_{i=1}^{T}(y_i - \hat{y}_i)$)
So, as you can see, all of these equations model a trade-off between model complexity (number of parameters) and model fit (RSS).

Below, I have presented tables summarising the information criteria of all possible models for each economic indicator. The columns of these tables are coloured such that the lowest value is the lightest shade and the highest value is the darkest, so I will attempt to spot the overall lightest rows.

CPI Model Selection Table

ARIMA_Order	AIC	BIC	HQIC
(1, 1, 1)	844.324939	857.723373	849.524419
(1, 1, 2)	757.823651	775.688230	764.756292
(1, 1, 3)	756.497515	778.828239	765.163316
(1, 1, 4)	757.881688	784.678557	768.280649
(1, 1, 10)	762.794302	816.388039	783.592224
(1, 1, 11)	764.503850	822.563731	787.034931
(3, 1, 1)	757.959107	780.289831	766.624908
(3, 1, 2)	758.595822	785.392691	768.994783
(3, 1, 3)	759.692528	790.955541	771.824649
(3, 1, 4)	752.995242	788.724400	766.860523
(3, 1, 10)	744.383531	806.909557	768.647773
(3, 1, 11)	745.849362	812.841533	771.846764
(10, 1, 1)	757.702806	811.296543	778.500728
(10, 1, 2)	745.643803	803.703684	768.174885
(10, 1, 3)	745.929548	808.455574	770.193789
(10, 1, 4)	746.314270	813.306441	772.311672
(10, 1, 10)	738.704765	832.493804	775.101127
(10, 1, 11)	745.508353	843.763537	783.637876

GDP Model Selection Table

ARIMA_Order	AIC	BIC	HQIC
(2, 1, 2)	2864.909020	2881.691952	2871.692289
(2, 1, 3)	2866.153888	2886.293405	2874.293810
(2, 1, 5)	2869.787734	2896.640424	2880.640964
(3, 1, 2)	2865.638369	2885.777887	2873.778292
(3, 1, 3)	2867.171136	2890.667240	2876.667712
(3, 1, 5)	2871.350076	2901.559353	2883.559960

Unemployment Model Selection Table

ARIMA_Order	AIC	BIC	HQIC
(2, 1, 2)	835.615501	857.946224	844.281301
(2, 1, 4)	844.437409	875.700422	856.569530
(4, 1, 2)	844.355903	875.618916	856.488023
(4, 1, 4)	847.726348	887.921650	863.324789

GDP Model Selection

From the table above, it is clear that the ARIMA(2,1,2) model appears to be the best in relation to the AIC, BIC and HQIC. While models such as (3,1,2), (3,1,3) and (2,1,3) also appear to have good performance among the information criteria, the (2,1,2) model appears to beat all of these on all criteria. Therefore, the ARIMA(2,1,2) model best describes the features of the GDP time series using the smallest number of parameters.

CPI Model Selection

For CPI, it is more difficult to choose the best model as no one model dominates all others across all information criteria. However, it does appear that the ARIMA(1,1,2) model has the best overall performance. While models such as (3,1,10) and (10,1,3) have lower AIC values than the (1,1,2) model, they both have higher BIC and HQIC. Therefore, the ARIMA(1,1,2) model best describes the features of the CPI time series using the smallest number of parameters.

UNEM Model Selection

I can see that the ARIMA(2,1,2) model is the best in relation to the AIC, BIC and HQIC information criteria. This model has the lowest AIC and HQIC and the 3rd best BIC, so overall, it outperforms the other possible models. Therefore, the ARIMA(2,1,2) model best describes the features of the Unemployment time series using the smallest number of parameters.

Now that I have decided on the best ARIMA models for each time series, it is helpful to check that the models' residuals are white noise. Without this property, the quality of forecasting for a given model will be poor. To check for this, I will use the Ljung-Box Test:.

Ljung-Box Test

This is a statistical test used to check if autocorrelation in the lags is different from zero. In order words, the null and alternative hypothesis are:

$H_0 : \rho_1 = \rho_2 = ... = \rho_n = 0$

H_1: At least one is $\neq 0$

The test statistic has a Chi-squared distribution and is given by:

$Q = h(h+2)\sum_{k=1}^{n} \frac{\hat{\rho_k^2}}{h-k} \sim \chi_n^2$

So for large $\hat{\rho_k^2}$, Q is large, so we are more likely to reject the null hypothesis.

But for small values of $\hat{\rho_k^2}$, we get a small Q and fail to reject H_0.

I carried out this test for 30 lags using Python and presented the results below:

Models	Q Stat	P-Value
GDP (2, 1, 2)	6.912665	0.999867
CPI (1, 1, 2)	0.536648	1.000000
UNEM (2, 1, 2)	4.823634	0.999996

So, as you can see, I get a p-value of near 1 for all models. Therefore, I fail to reject the null hypothesis that there is no autocorrelation in the lags of the residuals for each of my models. Consequently, I am confident that my residuals are white noise, so I am happy to use these models to forecast.

2.3 Forecasting

Now that I have selected the appropriate models for the time series of each of my economic indicators, I can use these models to provide a forecast for some upcoming observations. For each time series, I will forecast

two steps ahead:

GDP : Q3 and Q4 levels
CPI: October and November levels
Unemployment: October and November levels

To indicate how this process works mathematically, below, I have given the formula for calculating the H-steps ahead:

Given an AR(1) model of the form $y_t = \phi y_{t-1} + \epsilon_t$ the value 2-steps ahead is calculated as:

$$\hat{y}_{t+2|T} = E[y_{t+2}|T] = E[\phi y_{t+1} + \epsilon_{t+2}|T] = \phi E[y_{t+1}|T] = \phi^2 Y_T$$

So by recursion for H-steps ahead, we have:

$$\hat{y}_{t+h|T} = \phi^h Y_T$$

In our case, we are using ARIMA models, so the formulae are slightly different, but the process is the same. I have carried out this process in Python and presented the outputs in the table below.

Forecasts	Dates	Mean	Standard Errors	Lower Bound	Upper Bound
GDP Q3	2023-07-01	27409.719357	202.166763	27013.479782	27805.958931
GDP Q4	2023-10-01	27740.742251	278.853884	27194.198681	28287.285822
CPI October	2023-10-01	308.369822	0.432849	307.521454	309.218191
CPI November	2023-11-01	309.397400	0.793847	307.841488	310.953312
UNEM October	2023-10-01	3.860474	0.459205	2.960449	4.760499
UNEM November	2023-11-01	3.951084	0.661847	2.653887	5.248280

Above, you can see the forecasts for the next two steps for each economic indicator. In this table, I have presented the date corresponding to the forecasted step; I have kept this aligned with how FRED give their dates (i.e. for GDP Q3, the corresponding date is the first date in the 3rd quarter). I have also displayed the mean, standard deviation and upper/lower bounds of the 95% confidence interval for each forecasted step. Below, I have reproduced my guesses from part A to aid my comparison.

Period	GDP	Period	CPI	Period	UNEM
Q1 2023	26813.60	Aug-2023	306.27	Aug-2023	3.8%
Q2 2023	27063.01	Sep-2023	307.48	Sep-2023	3.8%
Q3 2023	27312.42	Oct-2023	307.24	Oct-2023	3.8%
Q4 2023	27561.83	Nov-2023	307.00	Nov-2023	3.8%

GDP

As you can see, my initial guesses for Q3 (27312.42) and Q4 (27561.83) GDP line up quite well with the values forecasted by my ARIMA model (27409.26 and 27739.898). My original guesses were quite naive in that I assumed that the rate of increase over the next two periods would be the same as the previous period. However, the ARIMA model forecasts a larger increase in GDP over the next two periods.

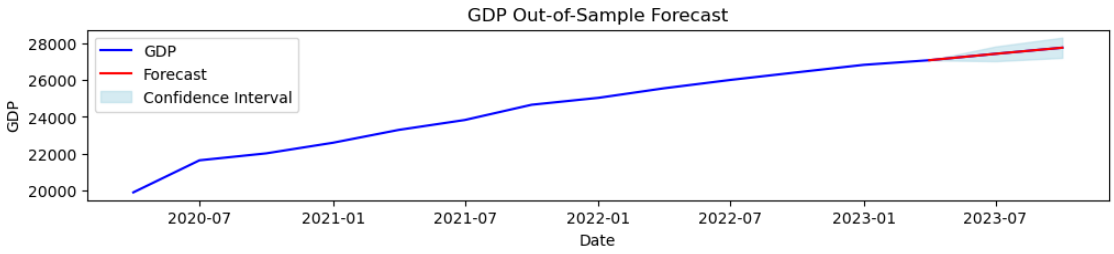
CPI

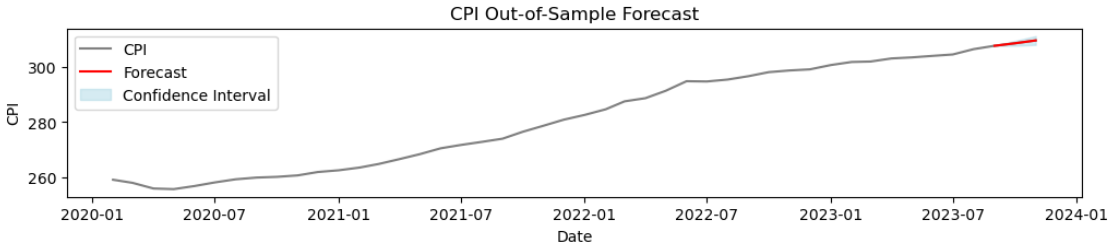
In part A, I decided that due to updated consensus for inflation and overall economic cooling, CPI would decrease over the next two periods (307.24,307.00). However, the ARIMA model forecasts a continued increase in CPI over the next two periods (308.37,309.397). Until the true values of CPI are released for the next two periods, it is difficult to say whether the current sentiment about inflation is correct and the model is performing poorly out-of-sample or if CPI is, in fact, still increasing.

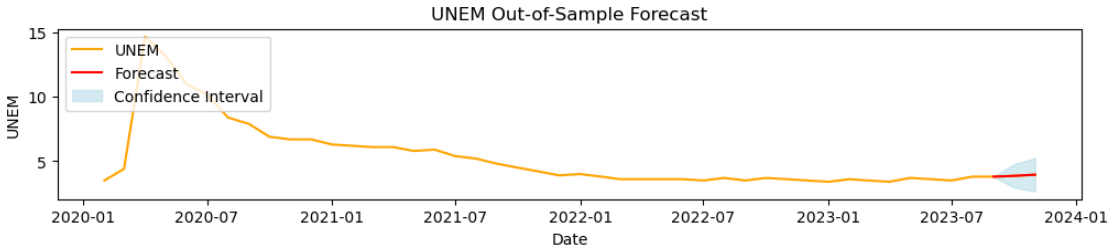
Unemployment

In part A, I predicted that Unemployment would remain steady for the next two periods (3.8%,3.8%). However, the ARIMA model forecasts a small increase in Unemployment over the next two periods (3.86%,3.95%). In my opinion, this small increase does indicate that unemployment will remain steady, which aligns with my guesses.

So overall, it appears that the ARIMA model forecasts for GDP and CPI follow a similar sentiment to my predictions. On the other hand, the CPI model predicts a continued increase in the following two periods. Below, I have presented the forecasts and confidence intervals (in blue) on the plots of the original time series over the last three years. As you can see, the total area of the confidence intervals varies across the time series; this is an indicator of the variance of the original time series. In part A, I observed that the plot of GDP and CPI were much less volatile than the plot of Unemployment.







3 Question C

3.1 Generating Forecasts

In this section, I will use the models I found in Question B to forecast one year (October 2022 to September 2023) worth of observations for both CPI and Unemployment. Once I have generated the 12m forecast, I will compare it to the actual values of CPI and Unemployment over that time period and carry out some tests to check the quality of my forecasts.
Below, I have presented summary tables for my forecasted values and plots comparing the forecast to the actual values.

CPI Pseudo Out-of-Sample Forecast

Dates	Means	Standard Errors	Lower Bound	Upper Bound
2022-10-01	3.558625	0.462889	2.651379	4.465870
2022-11-01	3.675699	0.667146	2.368117	4.983282
2022-12-01	3.745000	0.798481	2.180006	5.309994
2023-01-01	3.842764	0.911937	2.055400	5.630127
2023-02-01	3.914648	1.000782	1.953152	5.876145
2023-03-01	3.999748	1.080178	1.882638	6.116858
2023-04-01	4.070285	1.146928	1.822347	6.318222
2023-05-01	4.146328	1.207122	1.780413	6.512243
2023-06-01	4.213699	1.259593	1.744942	6.682457
2023-07-01	4.282722	1.307144	1.720767	6.844678
2023-08-01	4.346179	1.349465	1.701276	6.991082
2023-09-01	4.409399	1.387970	1.689027	7.129771

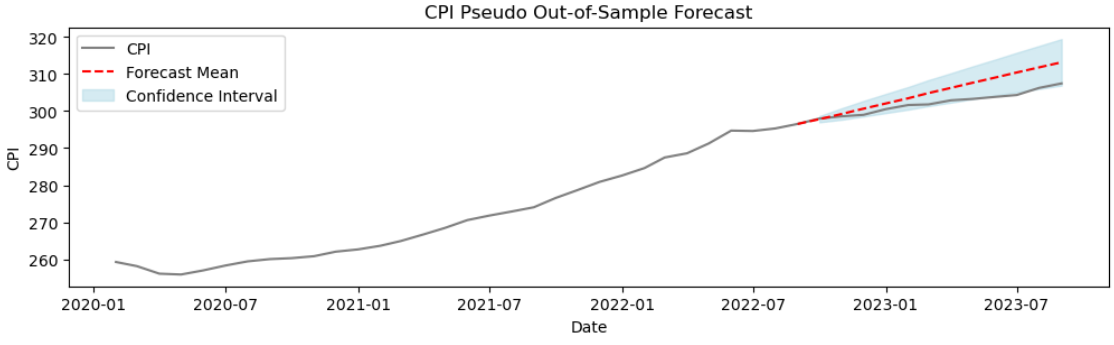
UNEM Pseudo Out-of-Sample Forecast

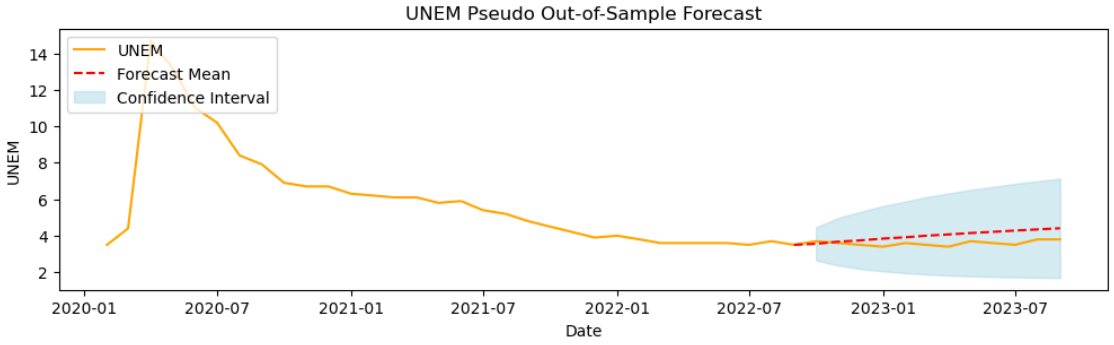
Dates	Means	Standard Errors	Lower Bound	Upper Bound
2022-10-01	297.817842	0.427386	296.980181	298.655503
2022-11-01	299.239312	0.788318	297.694237	300.784386
2022-12-01	300.655885	1.066545	298.565495	302.746275
2023-01-01	302.067579	1.317895	299.484552	304.650606
2023-02-01	303.474411	1.557349	300.422063	306.526759
2023-03-01	304.876397	1.791320	301.365475	308.387319
2023-04-01	306.273554	2.023140	302.308273	310.238835
2023-05-01	307.665898	2.254742	303.246686	312.085111
2023-06-01	309.053447	2.487323	304.178383	313.928511
2023-07-01	310.436216	2.721661	305.101858	315.770574
2023-08-01	311.814223	2.958274	306.016113	317.612332
2023-09-01	313.187482	3.197512	306.920474	319.454490

Above, you can see the forecasted values from September 2022 to September 2023 for CPI and Unemployment. Below, I have provided a visualisation (over the past three years) of these predicted values on the same plot as the in-sample values which the models are attempting to forecast. By observing the differences in the visualisation and by calculating a measure of fit statistics, I will comment on the overall fit of the predicted values.

3.2 Forecast Evaluation

3.2.1 Visually





As you can see from the plots of the pseudo out-of-sample forecast, the forecasted values have correctly captured the direction of the trend for CPI and Unemployment over the last year (both forecast increasing trends). At first glance, it appears that the magnitude of the trend is captured more effectively by the Unemployment forecast than by CPI. However, I noticed that the y-axis scale for CPI is much larger, so differences appear more pronounced. So, to compare the fit of both forecasts, I will use some statistics below. Both forecasts also failed to capture the month-on-month variations in the data and instead produced a linear forecast. This difficulty in correctly capturing the non-linearity of Out-of-Sample data is one of the main downfalls of ARIMA modelling.
On both plots, I have included the confidence interval of the forecast value as a blue shading. As you can see, the actual values of the time series over the past year lie within these bounds. Therefore, I believe the forecasted values for each time series represent a reliable estimate of the actual values on average. One interesting point to note is the increase in the width of these confidence intervals, the more steps we take from the last observation. This interval could be kept more constant if a process such as a rolling window (adding a forecasted value to the original dataset and retraining the model to predict the next value) was used.

3.2.2 Statistically

As I mentioned above, a forecast can be evaluated visually using the plots I displayed above; however, to accurately quantify the quality of a forecast, it is useful to run some statistical tests.
When forecasting, we would like to minimise a factor known as a forecasting error:

$$\tilde{\epsilon_t} = Y_{t+1} - \hat{Y}_{T+1|T}$$

This is the difference between the actual value of Y_{t+1} and the forecasted value.
This can be done using typical statistical measures of difference such as:
Mean Squared Error (known as mean squared forecast error in this case):

$$MSFE = \frac{1}{n}\sum_{i=1}^{n} \tilde{\epsilon_i}^2$$

Root Mean Squared Error (known as root mean squared forecast error in this case):

$$RMSFE = \sqrt{MSFE}$$

Mean Absolute Error (known as mean absolute forecast error in this case):

$$MAFE = \frac{1}{n}\sum_{i=1}^{n} |\tilde{\epsilon_i}|$$

Mean Absolute Percentage Error (known as mean absolute percentage forecast error in this case):

$$MAPFE = \frac{1}{n}\sum_{i=1}^{n} |\frac{\tilde{\epsilon_i}}{Y_i}|$$

where n is the period of the forecast
Below, I have presented a table of these values for each of my time series:

Forecasts	MSFE	RMSFE	MAFE	MAPFE
CPI	14.686191	3.832257	3.265214	1.074340
UNEM	0.244832	0.494805	0.448987	12.546895

For the CPI forecast, the MSFE is 14.69, and for Unemployment, it is 0.245. MSFE measures the average squared difference between a forecasted value and an actual value. A higher MSFE indicates less accurate forecasts; however, in this case, it is not comparable across the CPI and Unemployment values as the observations in the two datasets are not normalised on the same scale. So it appears that the CPI forecast has a much worse fit when, in reality, it just has larger values, so on average, larger errors. RMSFE is the square root of the MSFE and represents the average difference between the forecast and actual values. A similar scale effect is seen in this statistic. MAFE, on the other hand, measures the average absolute error between forecast and actual values, so like the MSFE and RMSFE, lower values are better. Still, they also cannot be used to compare the fit across the forecasts. However, my final statistic MAPFE can be used to compare fit across the forecasts. MAPFE measures the average percentage difference between the forecasted values and the actual values. Since this value is a percentage, it does not suffer from the scale issues the previous three statistics had, so it can be used to compare the fit of the forecasts. As you can see, the average percentage error for the CPI forecast is 1.07% and 12.55% for Unemployment. So it is clear that, on average, the CPI forecast is a much better fit than the Unemployment forecast.

4 Conclusion

To conclude my project:
In part A, I guessed the next two values for GDP, CPI and Unemployment by observing the historical trend and informing my guess with short-term macroeconomic news.
In part B, I made my time series stationary using differencing and detrending before following the Box-Jenkins Method to choose the correct model for each time series. First, I used the ACF and PACF to select the possible orders of my ARIMA models for each time series. Following this, I carried out model selection on all of my possible models using information criteria (AIC, BIC, HQIC). Finally, after choosing my model, I forecasted the following two values for each time series and compared them to my guesses in part A.
Lastly, in part C, I carried out a Pseudo Out-of-Sample forecast from September 2022 to September 2023 for CPI and Unemployment. Using visual and statistical tools, I checked the quality of the forecasts and compared the fit of the forecasts across the two Economic Indicators using a scaled statistic MAPFE.

Appendix

Module used to call FRED Price Data Through their API

In [1]:
```
from datetime import datetime

import pandas as pd
import requests

class price_data_generator:

    def __init__(self,start_date = None,end_date = datetime.now().strftime('%Y-%m-%d')):
        self.today_date : str = datetime.now().strftime('%Y-%m-%d')
        self.start_date : str = start_date
        self.fred_api_key : str = '###############' #hiding API key in accordance with FRED T&C's
        self.end_date : str = end_date


    def get_fred_data(self,inst):
        url : str = 'https://api.stlouisfed.org/fred/series/observations'

        params : dict = {
            'series_id': inst,
            'api_key': self.fred_api_key,
            'file_type': 'json',
            'observation_start': self.start_date,
            'observation_end': self.end_date,
        }

        response = requests.get(url, params=params)

        if response.status_code == 200:
            data = response.json()

        else:
            print(f"Error: {response.status_code}")

        price_data = []
        for day in data['observations']:
            date : str = day['date']
            if day['value'] != '.':
                price : float = float(day['value'])
                price_data.append([date,price])

        data : pd.DataFrame = pd.DataFrame(price_data,columns = ['date',f'{inst}_price'])
        data['date'] = pd.to_datetime(data['date'])

        return data
```

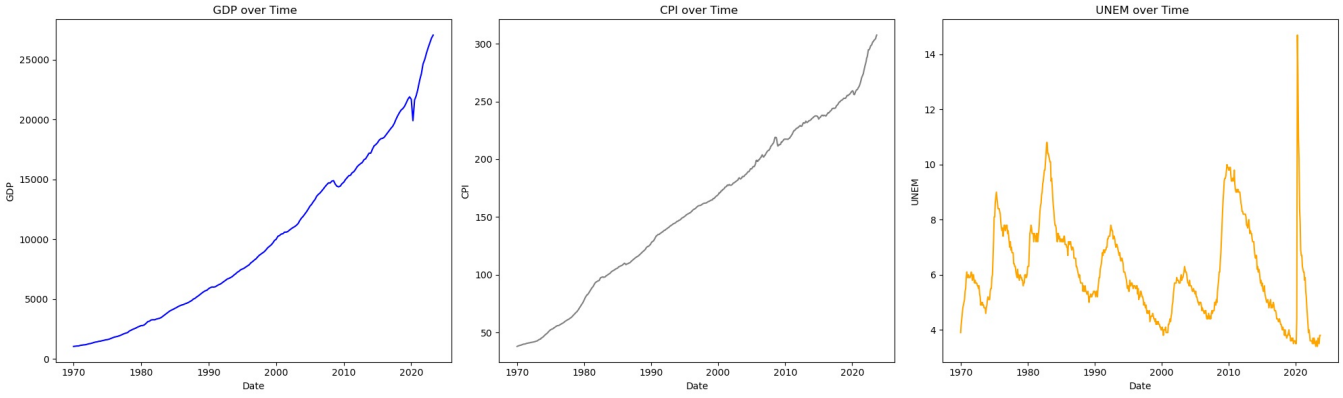
Project Code

In [2]:
```
from data_generator import price_data_generator as pdg
from datetime import datetime,timedelta
from statsmodels.graphics.tsaplots import plot_acf
from statsmodels.graphics.tsaplots import plot_pacf
from statsmodels.compat import lzip
from statsmodels.stats.diagnostic import acorr_ljungbox
from statsmodels.tsa.arima.model import ARIMA
from math import ceil
from statsmodels.tsa.tsatools import detrend
from scipy.stats import chi2
from IPython.display import HTML, display, Image

from statsmodels.tsa.stattools import adfuller
from arch.unitroot import ADF
from dateutil.relativedelta import relativedelta
from sklearn.metrics import mean_squared_error,mean_absolute_error

import pandas as pd
import numpy as np
import statsmodels.api as sm
import statsmodels.tsa as tsa
import matplotlib.pyplot as plt
import warnings
import itertools
import seaborn as sns


warnings.simplefilter(action='ignore', category=FutureWarning)
warnings.filterwarnings("ignore", category=UserWarning)
warnings.filterwarnings("ignore", category=pd.core.common.SettingWithCopyWarning)

%matplotlib inline
```

In [3]:
```
pdg = pdg(start_date = '1970-01-01')

inst_ref_data = {
    'GDP':{'code':'GDP','colour':'blue','palette':'Blues'},
    'CPI':{'code':'CPIAUCSL','colour':'grey','palette':'Greys'},
    'UNEM':{'code':'UNRATE','colour':'orange','palette':'Oranges'}
}

data = {}
for inst,ref_data in inst_ref_data.items():
    inst_data = pdg.get_fred_data(ref_data['code'])
    inst_data.columns = ['date','rate']
    data[inst] = inst_data
data['GDP'] = data['GDP'][:-1]
data['UNEM'] = data['UNEM'][:-1]

def generate_plot(inst,data=data,col='rate',min_date='1900-01-01',fig_size=(12,10),legend='Y',set_figure='Y',li
    if set_figure == 'Y':
        plt.figure(figsize = fig_size)
    data = data[inst]
    plot_data = data.loc[data['date']>datetime.strptime(min_date,'%Y-%m-%d')]
    plt.plot(plot_data['date'],plot_data[col],label = inst,color = inst_ref_data[inst]['colour'],linestyle = li
    plt.xlabel('Date')
    plt.ylabel(inst)
    plt.title(f"{col.replace('rate','').replace('_',' ')} {inst} over Time")
    if legend == 'Y':
        plt.legend(loc='upper left')
```

Question A

In [4]:
```
fig = plt.figure(figsize=(20,6))
for i,inst in enumerate(inst_ref_data.keys()):
    plt.subplot(1,3,i+1)
    generate_plot(inst,col = 'rate',set_figure = 'N',legend = 'N')

plt.tight_layout()

plt.show()
```



Question B

In [5]:
```
def dickey_fuller(inst,trend = 'n',col = 'rate'):
    data_ = data[inst][col]
    result_names = ['Dickey-Fuller Test Statistic','p-value',\
                    'Number of Lags Used', 'Number of Observations',\
                    'Critical Values','Maximized Information Criterion']
    results = adfuller(data_,regression = trend)
    return results[1],results[2]

def p_value_table(level=''):

    table = pd.DataFrame(index = [f"{level.replace('_',' ')} P-Value"])
    for inst in inst_ref_data.keys():
        table[inst] = [dickey_fuller(inst,trend = 'n',col = f'{level}rate')[0]]

    display(table)

def ADF_test(inst,trend = 'n',col = 'rate'):
    data_ = data[inst][col]
    test = ADF(data_,trend = trend)
    return test

def difference(data,order = 1):
    for _ in range(order):
        data = data.diff()
    return data
```

```
p_value_table()
```

	GDP	CPI	UNEM
P-Value	1.0	0.999924	0.339368

In [6]:
```
differencing_lags = {'GDP':1,'CPI':1,'UNEM':1}

def difference_time_series(lags):
    for inst,lag in differencing_lags.items():
        data[inst]['Differenced_rate'] = difference(data[inst]['rate'],\
                                                    order = lag)
        data[inst] = data[inst].iloc[lag:].reset_index(drop = True)

difference_time_series(differencing_lags)

p_value_table('Differenced_')
```

	GDP	CPI	UNEM
Differenced P-Value	0.016356	0.065782	1.851872e-25

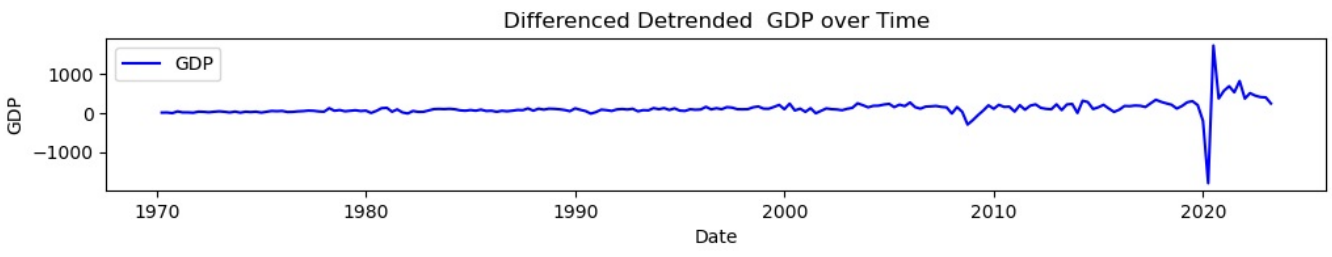
In [7]:
```
detrend_orders = {'n':0,'c':0,'ct':1,'ctt':2}

def generate_trend_summary_table():
    table = pd.DataFrame(index = detrend_orders.keys())
    for inst in inst_ref_data.keys():
        p_values = []
        for trend in detrend_orders.keys():
            p_value = round(dickey_fuller(inst,trend = trend,col = 'Differenced_rate')[0],5)
            p_values.append(p_value)
        table[inst] = p_values

    display(table)

generate_trend_summary_table()
```

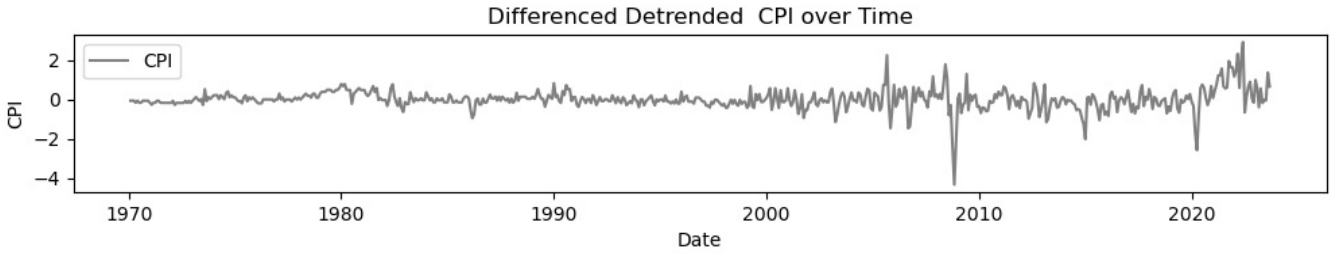
	GDP	CPI	UNEM
n	0.01636	0.06578	0.0
c	0.00130	0.00263	0.0
ct	0.00000	0.00844	0.0
ctt	0.00000	0.01280	0.0

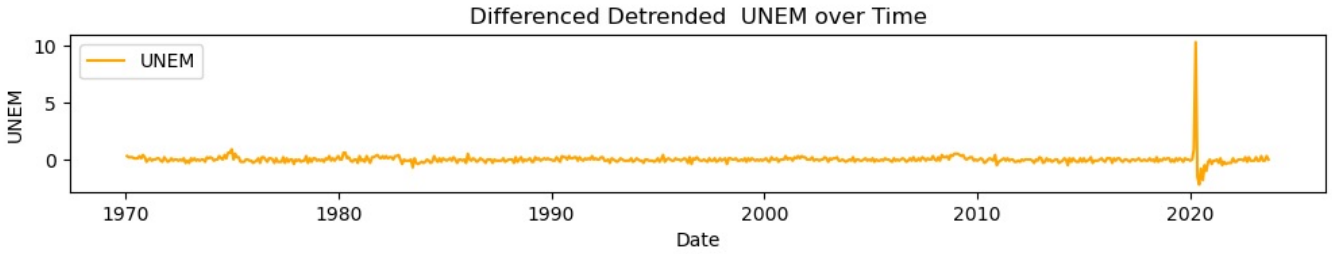
In [8]:
```
best_trends = {'GDP':'n','CPI':'c','UNEM':'n'}

def detrend_time_series(best_trends):
    for inst,trend in best_trends.items():
        if trend != 'n':
            data[inst]['Differenced_Detrended_rate'] = detrend(data[inst]['Differenced_rate'],order=detrend_ord
        else:
            data[inst]['Differenced_Detrended_rate'] = data[inst]['Differenced_rate']

detrend_time_series(best_trends)

p_value_table('Differenced_Detrended_')
```

	GDP	CPI	UNEM
Differenced Detrended P-Value	0.016356	0.000067	1.851872e-25

In [9]:
```
for inst in inst_ref_data.keys():
    generate_plot(inst,fig_size = (12,1.5),col = 'Differenced_Detrended_rate')
```



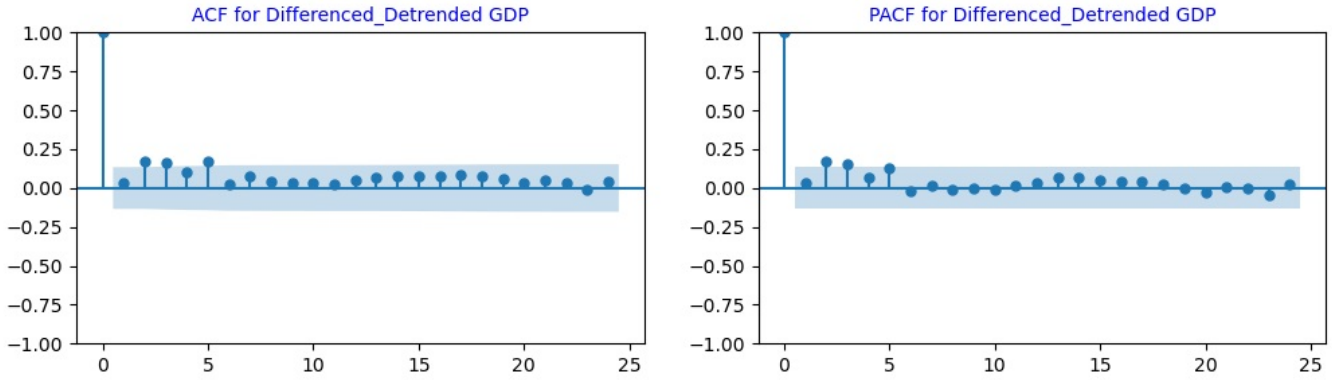




In [10]:
```
def plot_acf_pacf(inst,col = 'rate',size = 8,pacf_acf = 'Y',plot = 'Y',alpha = 0.05):

    lags = [i for i in range(25)]
    if plot == 'Y':
        generate_plot(inst,data = data,col = col,fig_size = (size*2,size*2/3))
        plt.show()

    if pacf_acf == 'Y':
        fig, axes = plt.subplots(1, 2, figsize=(size*3/2, size*3/8))


        plot_acf(data[inst][col], lags=lags, ax=axes[0],alpha = alpha)
        axes[0].set_title(f"ACF for {col[:-5]} {inst}",\
                          fontdict={'fontsize': size*5/4})\
        .set_color(inst_ref_data[inst]['colour'])

        # Plot PACF
        plot_pacf(data[inst][col], lags=lags, ax=axes[1],alpha = alpha)
        axes[1].set_title(f"PACF for {col[:-5]} {inst}",\
                          fontdict={'fontsize': size*5/4})\
        .set_color(inst_ref_data[inst]['colour'])

        plt.show()
```

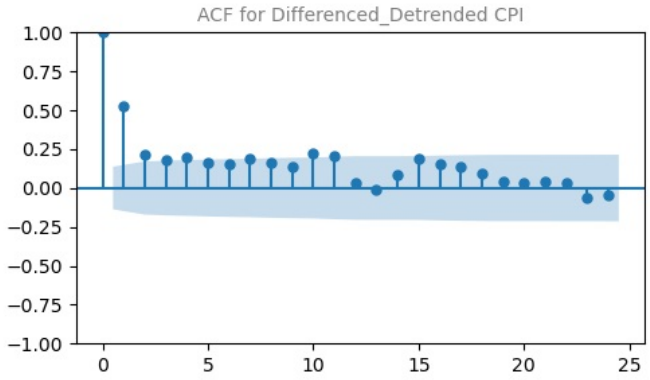
GDP

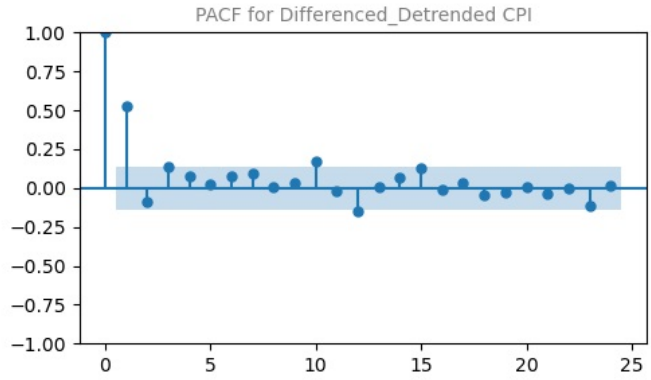
In [11]:
```
plot_acf_pacf('GDP',col = 'Differenced_Detrended_rate',plot = 'N')
```



CPI

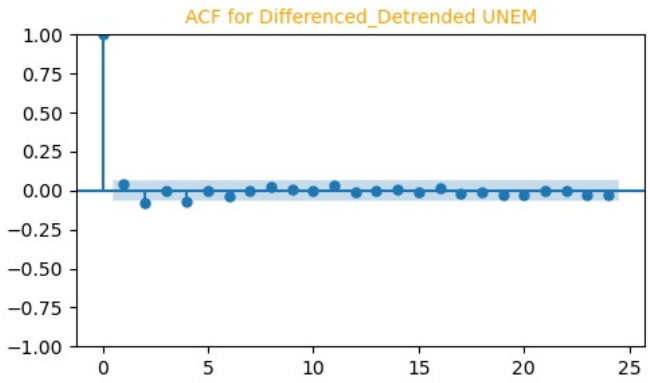
In [12]:
```
plot_acf_pacf('CPI',col = 'Differenced_Detrended_rate',plot = 'N')
```

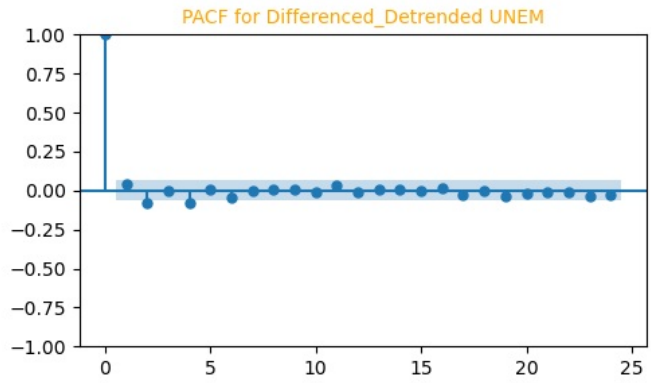




Unemployment

In [13]:
```
plot_acf_pacf('UNEM',col = 'Differenced_Detrended_rate',plot = 'N')
```





In [14]:
```
gdp_lags = {'ACF':[2,3,5],'PACF':[2,3]}
cpi_lags = {'ACF':[1,2,3,4,10,11],'PACF':[1,3,10]}
unem_lags = {'ACF':[2,4],'PACF':[2,4]}
lags_dict = {'GDP':gdp_lags,'CPI':cpi_lags,'UNEM':unem_lags}
```

Model Selection

In [15]:
```
def estimate_model(inst,order,col = 'rate'):
    order.insert(1,differencing_lags[inst])
    model = ARIMA(data[inst][col],order = order).fit()

    return model
```

In [16]:
```
def generate_summary(inst):
    ar_orders = lags_dict[inst]['PACF']
    ma_orders = lags_dict[inst]['ACF']
    combinations = [list(i) for i in list(itertools.product(ar_orders, ma_orders)) if list(i) != [0,0]]
    model_data = []
    for order in combinations:
        model = estimate_model(inst,order)
        aic = model.aic
        bic = model.bic
        hqic = model.hqic
        model_data.append([str(tuple(order)),aic,bic,hqic])
    model_data = pd.DataFrame(model_data,columns = ['ARIMA_Order','AIC','BIC','HQIC'])


    model_data.set_index('ARIMA_Order',inplace = True)

    colormap = sns.color_palette(inst_ref_data[inst]['palette'], as_cmap=True)
    styled_df = model_data.style.background_gradient(cmap=colormap, subset=['AIC','BIC','HQIC'])

    display(styled_df)
generate_summary('GDP')
generate_summary('CPI')
generate_summary('UNEM')
```

ARIMA_Order	AIC	BIC	HQIC
(2, 1, 2)	2864.909020	2881.691952	2871.692289
(2, 1, 3)	2866.153888	2886.293405	2874.293810
(2, 1, 5)	2869.787734	2896.640424	2880.640964
(3, 1, 2)	2865.638369	2885.777887	2873.778292
(3, 1, 3)	2867.171136	2890.667240	2876.667712
(3, 1, 5)	2871.350076	2901.559353	2883.559960

ARIMA_Order	AIC	BIC	HQIC
(1, 1, 1)	844.324939	857.723373	849.524419
(1, 1, 2)	757.823651	775.688230	764.756292
(1, 1, 3)	756.497515	778.828239	765.163316
(1, 1, 4)	757.881688	784.678557	768.280649
(1, 1, 10)	762.794302	816.388039	783.592224
(1, 1, 11)	764.503850	822.563731	787.034931
(3, 1, 1)	757.959107	780.289831	766.624908
(3, 1, 2)	758.595822	785.392691	768.994783
(3, 1, 3)	759.692528	790.955541	771.824649
(3, 1, 4)	752.995242	788.724400	766.860523
(3, 1, 10)	744.383531	806.909557	768.647773
(3, 1, 11)	745.849362	812.841533	771.846764
(10, 1, 1)	757.702806	811.296543	778.500728
(10, 1, 2)	745.643803	803.703684	768.174885
(10, 1, 3)	745.929548	808.455574	770.193789
(10, 1, 4)	746.314270	813.306441	772.311672
(10, 1, 10)	738.704765	832.493804	775.101127
(10, 1, 11)	745.508353	843.763537	783.637876

ARIMA_Order	AIC	BIC	HQIC
(2, 1, 2)	835.615501	857.946224	844.281301
(2, 1, 4)	844.437409	875.700422	856.569530
(4, 1, 2)	844.355903	875.618916	856.488023
(4, 1, 4)	847.726348	887.921650	863.324789

In [17]:
```
best_models = {'GDP':(2,1,2),'CPI':(1,1,2),'UNEM':(2,1,2)}
```

In [18]:
```
def ljung_box(lags,data = data):
    result_names = ['Ljung-Box Test Statistic', 'p-value']
    p_values = []
    statistics = []
    models = []
    for inst,order in best_models.items():
        model = ARIMA(data[inst]['rate'],order=order).fit()
        results = acorr_ljungbox(model.resid,lags=25).iloc[-1]
        statistic = results[0]
        p_value = results[1]
        statistics.append(statistic)
        p_values.append(p_value)
        models.append(f'{inst} {order}')
    table = pd.DataFrame()
    table['Q Stat'] = statistics
    table['P-Value'] = p_values
    table['Models'] = models
    table.set_index('Models',inplace = True)
    display(table)

ljung_box(30)
```

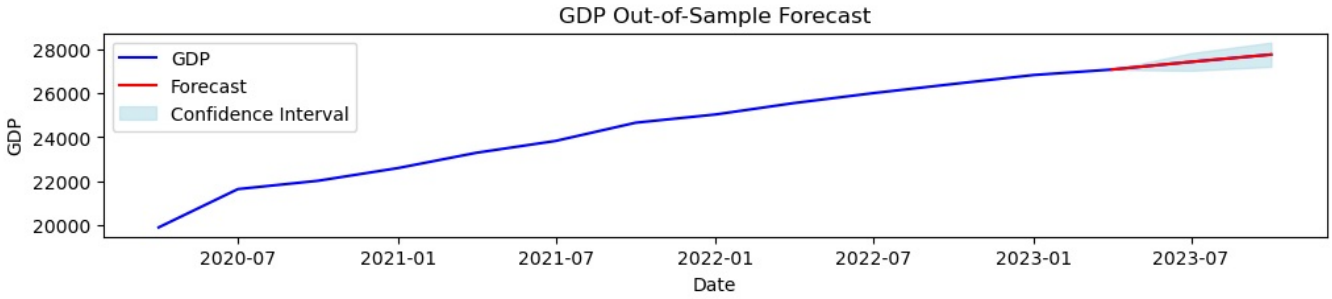
Models	Q Stat	P-Value
GDP (2, 1, 2)	6.912665	0.999867
CPI (1, 1, 2)	0.536648	1.000000
UNEM (2, 1, 2)	4.823634	0.999996

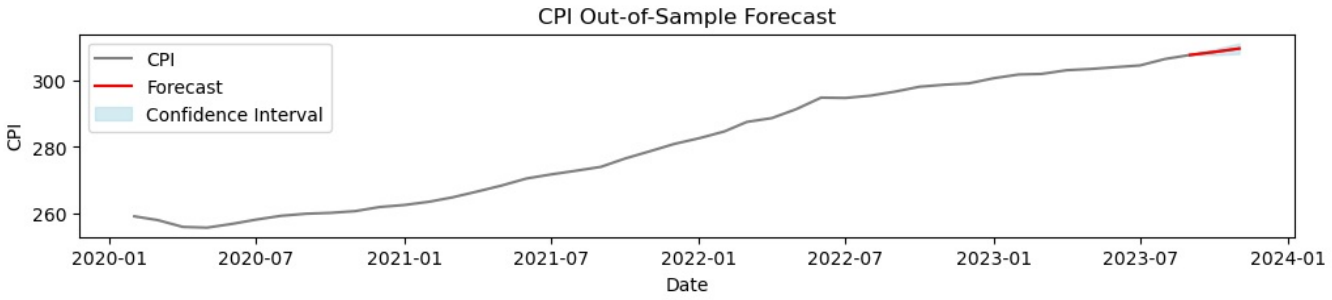
In [24]:
```
def generate_forecast(inst,steps,data=data,freq = None,dates = None):
    model = ARIMA(data[inst]['rate'],order = best_models[inst],freq = freq).fit()
    model.get_forecast(steps = steps)

    model_forecast = model.get_forecast(steps = steps)

    return model_forecast

step_labels_1 = {'GDP':['Q3','Q4'],'CPI':['October','November'],'UNEM':['October','November']}

periods = {'GDP':3,'UNEM':1,'CPI':1}

def generate_forecast_table(step_labels = step_labels_1,data = data):
    steps = 2
    table = pd.DataFrame()
    means,ses,lowers,uppers,index,dates = [],[],[],[],[],[]
    for inst in inst_ref_data.keys():
        for i in range(steps):
            forecast = generate_forecast(inst,steps = steps,data = data)
            mean = forecast.predicted_mean.iloc[i]
            se = forecast.se_mean.iloc[i]
            lower = forecast.conf_int(alpha = 0.05).iloc[i,0]
            upper = forecast.conf_int(alpha = 0.05).iloc[i,1]
            date = data[inst].iloc[-1]['date']  + relativedelta(months=+periods[inst]*(i+1))
            dates.append(date)
            means.append(mean)
            ses.append(se)
            lowers.append(lower)
            uppers.append(upper)
            index.append(f'{inst} {step_labels_1[inst][i]}')

    table['Dates'] = dates
    table['Mean'] = means
    table['Standard Errors'] = ses
    table['Lower Bound'] = lowers
    table['Upper Bound'] = uppers
    table['Forecasts'] = index
    table.set_index('Forecasts', inplace = True)
    display(table)
    return table


forecast_table = generate_forecast_table()
```

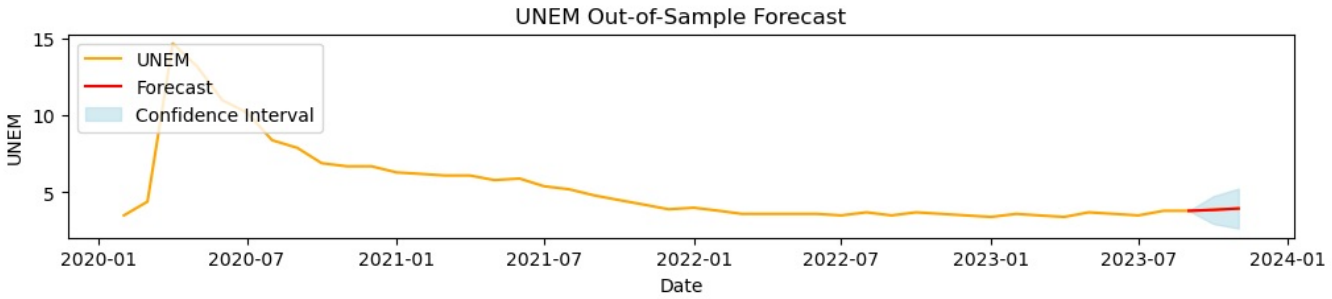
Forecasts	Dates	Mean	Standard Errors	Lower Bound	Upper Bound
GDP Q3	2023-07-01	27409.719357	202.166763	27013.479782	27805.958931
GDP Q4	2023-10-01	27740.742251	278.853884	27194.198681	28287.285822
CPI October	2023-10-01	308.369822	0.432849	307.521454	309.218191
CPI November	2023-11-01	309.397400	0.793847	307.841488	310.953312
UNEM October	2023-10-01	3.860474	0.459205	2.960449	4.760499
UNEM November	2023-11-01	3.951084	0.661847	2.653887	5.248280

In [25]:
```
forecast_data = data.copy()
forecast_data['lower_bound'] = ''
forecast_data['upper_bound'] = ''
for inst in inst_ref_data.keys():
    for index,row in forecast_table.iterrows():
        if inst in index:
            new_row_data = {'date':row['Dates'],'rate':row['Mean'],\
                            'lower_bound':row['Lower Bound'],'upper_bound':row['Upper Bound']}
            forecast_data[inst] = forecast_data[inst].append(new_row_data,ignore_index = True)
    forecast_data[inst].loc[len(forecast_data[inst])-3,'lower_bound'] = \
    forecast_data[inst].loc[len(forecast_data[inst])-3,'rate']
    forecast_data[inst].loc[len(forecast_data[inst])-3,'upper_bound'] = \
    forecast_data[inst].loc[len(forecast_data[inst])-3,'rate']

for inst in inst_ref_data.keys():
    generate_plot(inst,fig_size = (12,2),data = forecast_data,col = 'rate',min_date = '2020-01-01',legend = 'N'
    plt.plot(forecast_data[inst]['date'][-3:],forecast_data[inst]['rate'][-3:],\
             color = 'red',linestyle = '-',label = 'Forecast')
    plt.fill_between(forecast_data[inst]['date'],forecast_data[inst]['lower_bound'],\
                     forecast_data[inst]['upper_bound'],color = 'lightblue',alpha = 0.5,label = 'Confidence Int
```

```
    plt.title(f'{inst} Forecast')
    plt.legend(loc = 'upper left')
    plt.title(f'{inst} Out-of-Sample Forecast')
    plt.show()
```







Question C

Generating Forecasts

In [20]:

```
train_test_data = {}
train_test_data['train'] = {}
train_test_data['test'] = {}
for inst in ['CPI','UNEM']:

    split = 12
    train_test_data['train'][inst] = data[inst][:-int(split)]
    train_test_data['test'][inst] = data[inst][-int(split):]


def forecast_table(steps,data = train_test_data):
    forecasts = {}
    for inst in ['CPI','UNEM']:
        forecast_df = pd.DataFrame()
        training_data = data['train']
        forecast = generate_forecast(inst,steps = steps,data = training_data)
        means = forecast.predicted_mean.values
        dates = [training_data[inst].iloc[-1]['date'] + relativedelta(months=+(i+1)) for i in range(steps)]
        ses = [forecast.se_mean.iloc[i] for i in range(steps)]
        lower = [forecast.conf_int(alpha = 0.05).iloc[i,0] for i in range(steps)]
        upper = [forecast.conf_int(alpha = 0.05).iloc[i,1] for i in range(steps)]
        forecast_df['Dates'] = dates
        forecast_df['Means'] = means
        forecast_df['Standard Errors'] = ses
        forecast_df['Lower Bound'] = lower
        forecast_df['Upper Bound'] = upper
        forecast_df.set_index('Dates',inplace = True)
        display(forecast_df)
        first_row = pd.DataFrame()
        first_row['Dates'] = [training_data[inst].iloc[-1]['date']]
        first_row['Means'] = [training_data[inst].iloc[-1]['rate']]
        forecasts[inst] = forecast_df
        forecasts[inst] = pd.concat([first_row,forecasts[inst].reset_index()],ignore_index = True)
        forecasts[inst].set_index('Dates',inplace = True)
    return forecasts
```

```
forecasts = forecast_table(12)
```

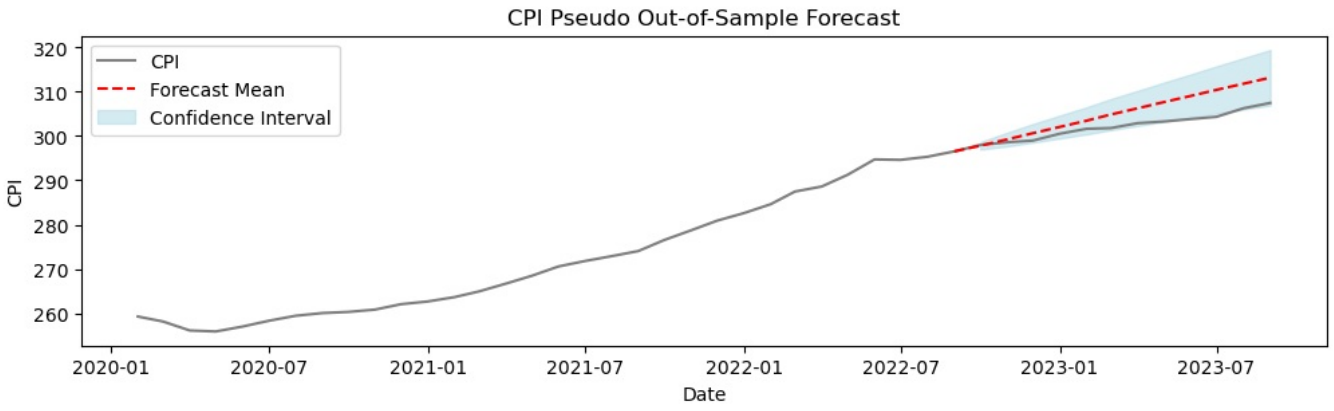
Dates	Means	Standard Errors	Lower Bound	Upper Bound
2022-10-01	297.817842	0.427386	296.980181	298.655503
2022-11-01	299.239312	0.788318	297.694237	300.784386
2022-12-01	300.655885	1.066545	298.565495	302.746275
2023-01-01	302.067579	1.317895	299.484552	304.650606
2023-02-01	303.474411	1.557349	300.422063	306.526759
2023-03-01	304.876397	1.791320	301.365475	308.387319
2023-04-01	306.273554	2.023140	302.308273	310.238835
2023-05-01	307.665898	2.254742	303.246686	312.085111
2023-06-01	309.053447	2.487323	304.178383	313.928511
2023-07-01	310.436216	2.721661	305.101858	315.770574
2023-08-01	311.814223	2.958274	306.016113	317.612332
2023-09-01	313.187482	3.197512	306.920474	319.454490

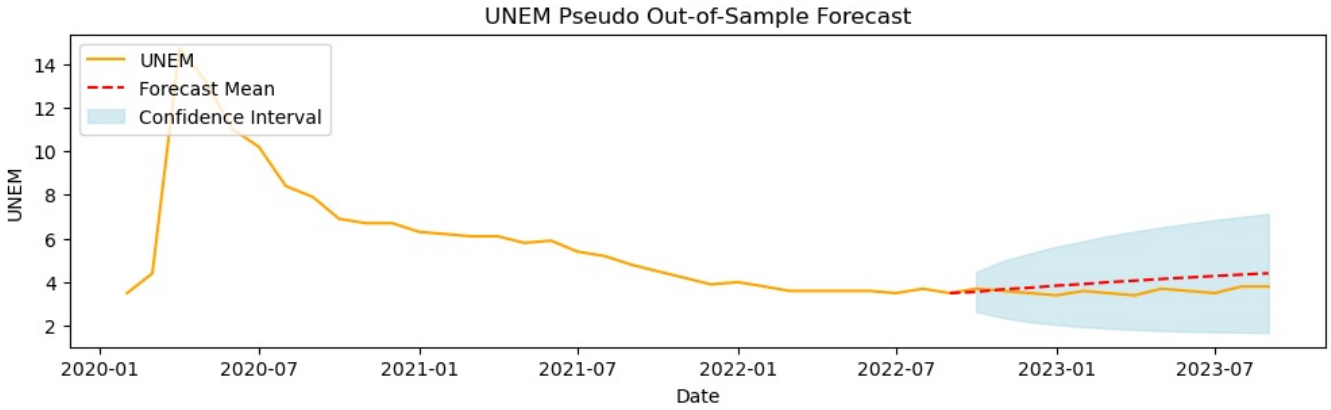
Dates	Means	Standard Errors	Lower Bound	Upper Bound
2022-10-01	3.558625	0.462889	2.651379	4.465870
2022-11-01	3.675699	0.667146	2.368117	4.983282
2022-12-01	3.745000	0.798481	2.180006	5.309994
2023-01-01	3.842764	0.911937	2.055400	5.630127
2023-02-01	3.914648	1.000782	1.953152	5.876145
2023-03-01	3.999748	1.080178	1.882638	6.116858
2023-04-01	4.070285	1.146928	1.822347	6.318222
2023-05-01	4.146328	1.207122	1.780413	6.512243
2023-06-01	4.213699	1.259593	1.744942	6.682457
2023-07-01	4.282722	1.307144	1.720767	6.844678
2023-08-01	4.346179	1.349465	1.701276	6.991082
2023-09-01	4.409399	1.387970	1.689027	7.129771

Forecast Evaluation

Visually

In [21]:
```
for inst in ['CPI','UNEM']:
    generate_plot(inst,fig_size = (12,3),data = data,col = 'rate',min_date = '2020-01-01',legend = 'N')
    plt.plot(forecasts[inst]['Means'],color = 'red',linestyle = '--',label = 'Forecast Mean')
    plt.fill_between(forecasts[inst].index,forecasts[inst]['Lower Bound'],\
                     forecasts[inst]['Upper Bound'],color = 'lightblue',alpha = 0.5,label = 'Confidence Interva
    plt.title(f'{inst} Pseudo Out-of-Sample Forecast')
    plt.legend(loc = 'upper left')
    plt.show()
```





In [22]:
```
evaluation_table = pd.DataFrame()
mses = []
maes = []
mapes = []
for inst,forecast in forecasts.items():
    merged = pd.merge(forecast.reset_index(),data[inst], left_on = 'Dates',right_on = 'date',how = 'inner')
    merged = merged[1:]
    mse = mean_squared_error(merged['rate'], merged['Means'])
    mae = mean_absolute_error(merged['rate'], merged['Means'])
    mape = np.mean(np.abs((merged['rate'] - merged['Means']) / merged['rate']))*100
    mses.append(mse)
    maes.append(mae)
    mapes.append(mape)
evaluation_table['MSFE'] = mses
evaluation_table['RMSFE'] = [i**(1/2) for i in mses]
evaluation_table['MAFE'] = maes
evaluation_table['MAPFE'] = mapes
evaluation_table['Forecasts'] = forecasts.keys()
evaluation_table.set_index('Forecasts',inplace = True)
display(evaluation_table)
```

Forecasts	MSFE	RMSFE	MAFE	MAPFE
CPI	14.686191	3.832257	3.265214	1.074340
UNEM	0.244832	0.494805	0.448987	12.546895