Financial Econometrics 1 - M2 FTD

Empirical Applications

Luis Miguel Fonseca
Stéphane Eloundou Mvondo
Natalia Cárdenas Frías

December 13, 2023

Contents

Introduction

This document compiles all our applications for the Financial Econometrics course. Each section represents a specific application, but we tried to make them coherent across them around a broad question:

1 Series Dynamics

1.1 Data

Note: Depending on each exercise along these applications we tried to use different series. We ended up testing more series than those taht we used therefore, this section encompasses more than the 3 series that were asked in the exercise.

In this work, we focus on the US market for the period 1990-2022 using monthly data. This time span allows us to encompass part of the Great Moderation and the last three crisis in the US economy: the dot com bubble, the Great Recession and the COVID crisis. We used monthly data in a hope to catch seasonal dynamics and a richer structure due to the rather high frequency. We use the following series retreived for the most part from FRED with its Python API (FRED tickers are in square brakets) :

GDP [GDP] This quaterly series is given by the US Bureau of Economic Analysis measures the US output at market value (nominal series). We apply the log transformation of this series from the get-go so one should consider the unit of the series to be in logs of billions of dollars (seasonally adjusted annual rate). Since the series has originally a quaterly frequency, we interpolate the missing months as midpoints. This implies that we assume that the changes in the series QoQ is equally distributed across all the months.

Real Disposable Personal Income [DSPIC96] This monthly series is made public by the US Bureau of Economic Analysis. As for the previous series, we applied the log transformation form the beggining so we consider it to be expressed in logs of billion of chained 2017 dollars[1]. The series is seasonally adjusted at an annual rate and we called it `dpi`.

Inflation Expectation [MICH] This data series is made public by the University of Michigan from their Survey of Consumers. The series represents the median expected value of the percent change in prices over the next year. The series is not seasonally adjusted. We called the series `infl_e`

GDP deflator [A191RI1Q225SBEA] As a measure of inflation, we decided to use the implicit price deflator of the US GDP. Unlike measures like the CPI deflators do not consider baskets of goods and therefore are broader measures of the price changes across the entire economy that measure the ratio of the GDP in value and volume. It is a measure produced by the US Bureau of Economic Analysis as a quarterly measure of percent change QoQ. The raw series is already seasonally adjusted at an annual rate. To get a monthly series, we interpolated the

[1] Unlike teh previous series that was nominal, this is a real measure due to the chained prices.

values within the same quarter using midpoints which implies that we assume that each month in a quarter contributes in the same way to the QoQ change.

FED fund rate [DFF] Its source is the Board of Governors of the Federal Reserve System and it is a mayor tools in conducting monetary policy as it is interest rate at which banks and other depository institutions trade federal funds with each other overnight. It is the main interest rate in the financial market and influences other interest rates. The series is daily and expressed in percent. To have monthly frequency, we took the average of the rate at every month. We called this series `rate` in our analysis.

S&P 500 price [SP500] It is the main index in the US stock market and the series retresents its value at market close. Its source is S&P Dow Jones Indices LLC and teh series is daily but the FRED series only has data from 2013. Since our analysis starts in the 1990s we complemented this series with Yahoo Finance's using its API. From the beggining of the analysis, we applied logs to this variable so all the information on its level is actually on the level of its log.

The series are plotted in Figure 1.

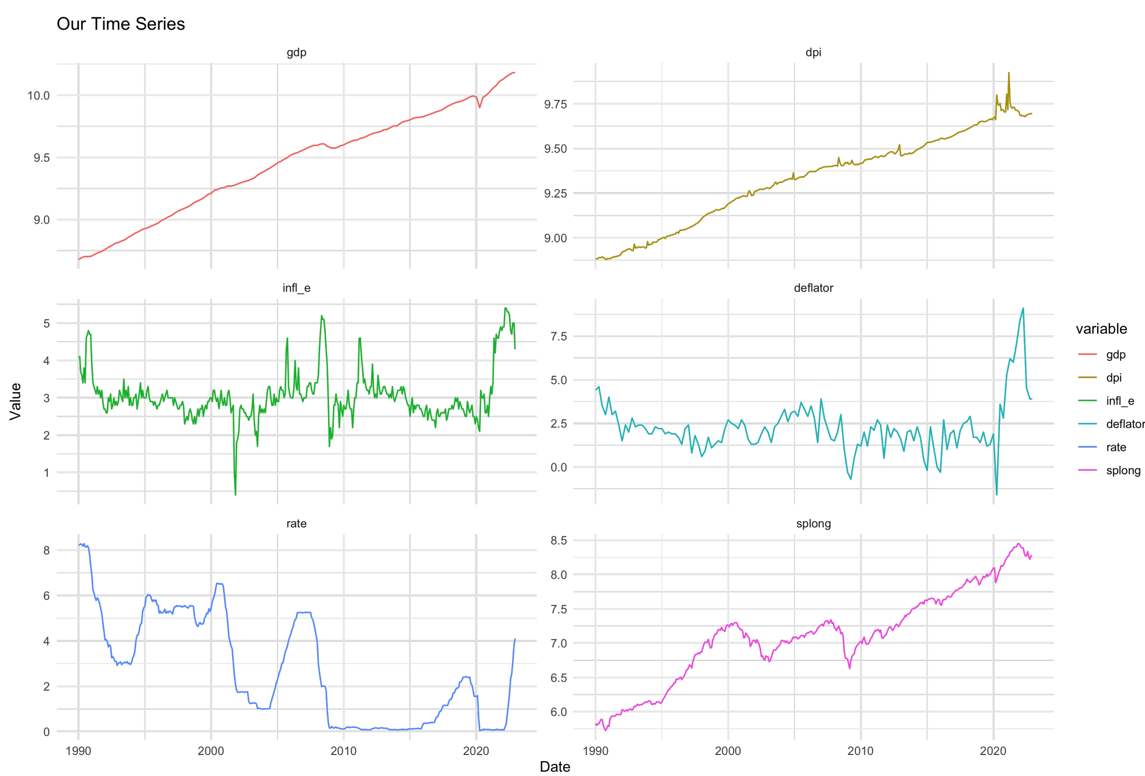


Figure 1: Plots of time series
Notes: The GDP, DPI and S&P500 series are in logs.

1.2 Unit root and trends

As for any time series analysis, the first analysis to perform is regarding the presence of unit roots in the series that would make them non-stationary. To do so, we perform the Augmented Dickey-Fuller tests that evaluate the presence of a stochastic trend (a unit root), a deterministic trend, and an intercept or drift. Importantly, this test requires estimating three equations/specifications because it requires investigating the joint presence of both types of trends and drift, for them to discard elements one by one. Each specification tests a different data-generating process of the series. For all specifications, the main null hypothesis H0 is that the series exhibits a UR. The inference with this test is non-standard and requires to use of corrected critical values to assess significance with the t-statistics.

1.2.1 ADF - Test jointly for deterministic and stochastic trend (with drift)

We first run the following specification to the ADF test to *jointly* investigate the presence of a stochastic and a determinist trend for each series $(X_t)_t$:

$$\Delta X_t = \alpha + \beta t + \gamma X_{t-1} + \sum_{i=1,2,..} \rho_i \Delta X_{t-i} + \varepsilon_t \tag{1}$$

As per usual, the ADF test assumes H0: $\gamma = 0$ i.e. a unit root exists and the series is non-stationary. We use R's built-in function `ur.df` with `type='trend'` to get this estimation. This function gives us (i) a regression table per series and (ii) a summary table with the following test statistics:

- tau3 refers to the t-statistic associated to H0: $\gamma = 0$ i.e. the presence of a UR. H1 refers to the absence of said UR.
- phi2 refers to the F-statistic associated with H0: $\alpha = \beta = \gamma = 0$ [2]
- phi3 is also an F-statistic, now associated to H0: $\beta = \gamma = 0$

Remark that the critical value in both tables can be a little different. This is because they are sensitive to the number of observations in each series. In Table 1, the critical values correspond to those provided directly by R and are associated with $N = 500$, while in Table 2 we give the values for $N = 250$. Since we have 396 data points per series we preferred to refer to the higher critical values but it does not change the analysis done.

Let us examine each series' results, summarized in the following tables:

Table 1: ADF test - 1st regression with drift, deterministic trend and stochastic trend

	gdp	dpi	infl_e	deflator	rate	splong	CV 1pct	CV 5pct	CV 10pct
tau3	-2.150	-2.694	-4.650	-5.197	-1.709	-1.975	-3.980	-3.420	-3.130
phi2	14.496	7.880	7.375	9.097	2.015	3.875	6.150	4.710	4.050
phi3	2.388	3.849	11.061	13.644	2.917	1.997	8.340	6.300	5.360

GDP We have $t_\gamma = -2.15 > -3.42$ we cannot reject the presence of a UR (H0). We shall note that phi2 ie that all coefficients are null is rejected since $F_{phi2} = 14.496 > 4.71$ while phi3 ($\gamma = \beta = 0$) is not ($F_{phi3} = 2.388 < 6.3$). This suggests the absence of a deterministic trend which is confirmed when assessing the significance of

[2] As with all F-tests, the alternative hypothesis is that at least one of these coefficients is non-null. Since this is general, we don't explicitly signal the H1 hereafter.

Table 2: ADF test - 1st regression t statistics

	gdp	dpi	infl_e	deflator	rate	splong
alpha	2.197	2.714	4.004	2.593	0.698	2.101
gamma	-2.150	-2.694	-4.650	-5.197	-1.709	-1.975
beta	2.094	2.577	1.265	1.166	-0.181	1.726
rho	16.326	-11.089	-0.255	14.255	15.719	4.099

Notes: With N=396, critical values at 5%: alpha = 3.09 ; gamma= -3.43 ; beta = 2.79

β using its non-standard critical value $|t_\beta| = 2.094 < 2.79$ (nullity, ie H0 cannot be rejected). These results require us to keep testing the series with the next specification of the test.

Disposable personal income Similarly as before we find $t_\gamma = -2.694 > -3.42$ and we fail to reject H0. Regarding the joint nullity tests as before we reject phi2 ($F_{phi2} = 7.88 > 4.71$) and cannot reject phi3 ($F_{phi3} = 3.849 < 6.3$). We also fail to reject the nullity of β as $|t_\beta| = 2.577 < 2.79$ and we shall test this series moving forward.

Inflation expectation We find $t_\gamma = -4.65 < -3.43$ we reject H0 ie we can't say that the series has a UR. The F-statistics of phi2 and phi3 lead us to reject their null hypothesis: $F_{phi2} = 7.375 > 4.71$, $F_{phi3} = 11.061 > 6.3$, leading us to believe that the series has either a drift and/or a deterministic trend. We, therefore, compare the t-statistics associated with α and β to the standard interest threshold (the critical values below Table 2 are conditional on having a UR). Since $|t_\alpha| = 4.004 > 1.96$ and $|t_\beta| = 1.265 < 1.96$, we fail to reject the presence of a deterministic trend while the drift term is significantly different from zero. We conclude that the series is *stationnary with a constant and without a deterministic trend*.

GDP deflator With $t_\gamma = -5.197 < -3.43$, as before we can reject H0 indicating that the series is stationary in levels. Since $|t_\alpha| = 2.593 > 1.96$, we reject the nullity of the drift. Finally, since $|t_\beta| = 1.166 < 1.96$ we cannot reject the absence of a deterministic trend. We conclude that the series is *stationary with a constant and without a deterministic trend*

Fed fund rate $t_\gamma = -1.709 > -3.43$, we are in the same situation as the previous series where we cannot reject the existence of a UR. Because we cannot reject phi2 nor phi3 ($F_{phi2} = 2.015 < 4.71,\ F_{phi3} = 2.917 < 6.3$) we need to continue testing this series with the other specifications as we don't reject the existence of a stochastic trend and we cannnot reject the nullity of the trend coefficient ($|t_\beta| = 0.181 < 2.79$).

S&P500 Without many surprises for price series, $t_\gamma = -1.975 > -3.43$ and we cannot reject the existence of a UR. Moreover, $F_{phi2} = 3.875 < 4.71$ and $F_{phi3} = 1.997 < 6.3$ (non-rejection of the null for both tests) leads to conclude that at least one of these coefficients is non-null (note that $|t_\beta| = 1.726 < 2.79$ and thus we cannot reject the nullity of β). We continue testing this series with the second specification of the test.

1.2.2 ADF - Test jointly for stochastic trend and drift

The second specification of the test models $\forall (X_t)_t$:

$$\Delta X_t = \alpha + \gamma X_{t-1} + \sum_{i=1,2,..} \rho_i \Delta X_{t-i} + \varepsilon_t \tag{2}$$

The null hypothesis still refers to H0: $\gamma = 0$ the presence of a unit root. We use now `type='trend'` in the `ur.df` function to get this estimation. The output of the test is similar to the previous specification and the same remarks on the critical values apply here. Now the test statistics reported refer to:

- tau2 refers to the t-statistic associated to $\gamma = 0$
- phi1 refers to the F-statistic associated to $\alpha = \gamma = 0$

Table 3: ADF test - 2nd regression with drift and stochastic trend

	gdp	dpi	rate	splong	CV 1pct	CV 5pct	CV 10pct
tau2	-0.621	-1.020	-2.412	-1.005	-3.440	-2.870	-2.570
phi1	19.382	8.378	3.014	4.300	6.470	4.610	3.790

Table 4: ADF test - 2nd regression t statistics

	gdp	dpi	rate	splong
alpha	0.983	1.127	1.508	1.258
gamma	-0.621	-1.020	-2.412	-1.005
rho	16.197	-12.106	15.984	3.977

Notes: With N=396, critical values at 5%: alpha = 2.53 ; gamma= -2.88

GDP We fail to reject the main null hypothesis (tau2) as $t_\gamma = -0.621 > -2.88$. Moreover, we do reject the joint nullity of α and γ as $F_{phi1} = 19.382 > 6.470$. Since we cannot reject the nullity of the drift term ($|t_\alpha| = 0.983 < 2.53$), we shall use the last specification of the test on this series.

Disposable personal income We fall in the same situation as with the previous series as tau2 is not rejected $t_\gamma = -1.02 > -2.88$ while phi1 is $F_{phi1} = 8.378 > 6.470$ and α is non-significant ($|t_\alpha| = 1.127 < 2.53$). We therefore also test this series with the last test specification.

Fed fund rate Given that $t_\gamma = -2.412 > -2.88$, we cannot reject the null hypothesis. We then check the F-statistic of the joint test *phi1*: $F_{phi1} = 3.014 < 4.61$: we cannot reject the null suggesting that the series has a UR and no drift. Supporting this, we also find that the drift term is not significantly different from zero as $|t_\alpha| = 1.508 < 2.53$. This leads us to use the third specification of the test.

S&P500 Since $t_\gamma = -1.005 > -2.87$, we cannot reject H0. By checking $F_{phi1} = 4.3 < 4.61$ and $|t_\alpha| = 1.258 < 2.53$, we fall in the same case as before where we need to continue testing the series as it seems to have a UR and no drift

1.2.3 ADF - Test for stochastic trend only

The last specification of the test keeps only the stochastic trend, $\forall (X_t)_t$:

$$\Delta X_t = \gamma X_{t-1} + \sum_{i=1,2,..} \rho_i \Delta X_{t-i} + \varepsilon_t \tag{3}$$

The null hypothesis still refers to H0: $\gamma = 0$ the presence of a unit root and we use `type='none'`. The output of the test is similar to the previous ones but now there is only one test statistic reported referring to the null (tau1). For this step, we only report the table with the t-statistics as its value for the "gamma" row is identical to the test statistics of tau1.

Table 5: ADF test - 3rd regression t statistics

	gdp	dpi	rate	splong
gamma	6.148	3.934	-1.935	2.647
rho	16.306	-12.122	15.950	3.976

Notes: With N=396, critical values at 5%: gamma= -1.95

GDP We have $t_\gamma = 6.148 > -1.95$. We clearly do not reject the presence of a UR (H0) and conclude that the *series has a unit root with no constant nor time trend*.

Disposable personal income Since $t_\gamma = 3.934 > -1.95$ we do not reject H0 and conclude that the *series has a unit root with no constant nor time trend*.

Fed fund rate We find $t_\gamma = -1.935 > -1.95$ thus we cannot reject the existence of a UR (this is a close call but seems adequate since we never rejected the UR and R's built-in `order_integration` also indicates a UR). We conclude that the *series has a unit root with no constant nor time trend*.

S&P500 Similarly, we find $t_\gamma = 2.647 > -1.95$ and we cannot reject H0 and conclude that the *series has a unit root with no constant nor time trend*.

1.2.4 Check stationarity of the series in deltas if UR in levels

Finally, to properly conclude that the previous series with UR are indeed I(1), we need to check that the series of their first differences are stationary (i.e. the series in deltas is I(0)). We perform the same ADF procedure to test these transformed series.

Table 6 reports the test results in the first specification of the ADF test. We easily see that all the t_γ (the statistic on tau3) are sufficiently negative to reject H0 and conclude that none of the differentiated series has a UR. This is sufficient to conclude that the series of GDP, of real disposable individual income, of the Fed fund rate, and of the returns of the S&P500 are indeed *integrated of order one*.

Table 6: ADF test - 1st regression with drift, deterministic trend and stochastic trend for series in deltas

	d_gdp	d_dpi	d_rate	d_splong	CV 1pct	CV 5pct	CV 10pct
tau3	-10.437	-19.070	-7.107	-13.317	-3.980	-3.420	-3.130
phi2	36.317	121.228	16.895	59.110	6.150	4.710	4.050
phi3	54.473	181.842	25.327	88.665	8.340	6.300	5.360

Moving forward, we will have to use these series in first-differences for models requiring statioanry series. Note that for most of the series, these first-differences have the advantage of being easily interpretable. Namely for the (log) GDP and (log) disposable personal income, their series in delatas is teh series of their growth rate. For the S&P500, the differenciated series are their rate of return. However, the interpretation of differenciated series of the Fed fund rate (in levels it was already a rate in percent) is not trivial as it does not have any intrisic meaning.

1.3 Decomposition of the series

All (monthly[3]) time series de can be decomposed into the following elements:

$$X_t = \underbrace{\alpha}_{\text{drift}} + \underbrace{\beta \times t}_{\text{deterministic trend}} + \underbrace{\gamma T t}_{\text{stochastic trend}} + \underbrace{\Sigma_{i=1}^{11} \rho_i \, \mathbb{1}_{\text{month = } i}}_{\text{seasonality}} + \underbrace{c_t}_{\text{cyclical component}}$$

Remark that the coefficients in the previous equation need not be the same as in the ADF test (the variable on the left-hand side is now in levels while AD tests the series in deltas).

We will apply this decomposition to stationary series. Therefore, the stochastic term $\gamma T t = 0$ for all series.

Using R's built-in function `stl`[4], we can decompose the time series. The results are plotted in Figure 2.

1.3.1 Estimation of the parameters in the stationary series

We perform we estimate the following regression to test the significance of the deterministic trend, drift and seasonal components:

$$X_t = \alpha + \beta \times t + \sum_{i=1}^{11} \rho_i \, \mathbb{1}_{\text{month}=i} + c_t$$

The results of this estimation are found in Table 7 for the series that are I(0) in levels, and in Table 8 for the series of first differences of the series that were not stationary

The coefficients on trend correspond to $\hat{\beta}$, those on the variables starting with M are $\hat{\rho}_i$. Since there is a monthly dummy less than the number of months, the coefficient on Contsnat encompasses both α and the coefficient on the last month. We tried to change the dummy that is ommitted and it does not change the results in terms of

[3]One should adapt the number of indicators in the seasonal components according to the data frequency. To avoid multicollinearity issues, the number of dummies must be one less than the frequency of the series: here we have only 11 indicators.

[4]Seasonal Decomposition of Time Series by Loess

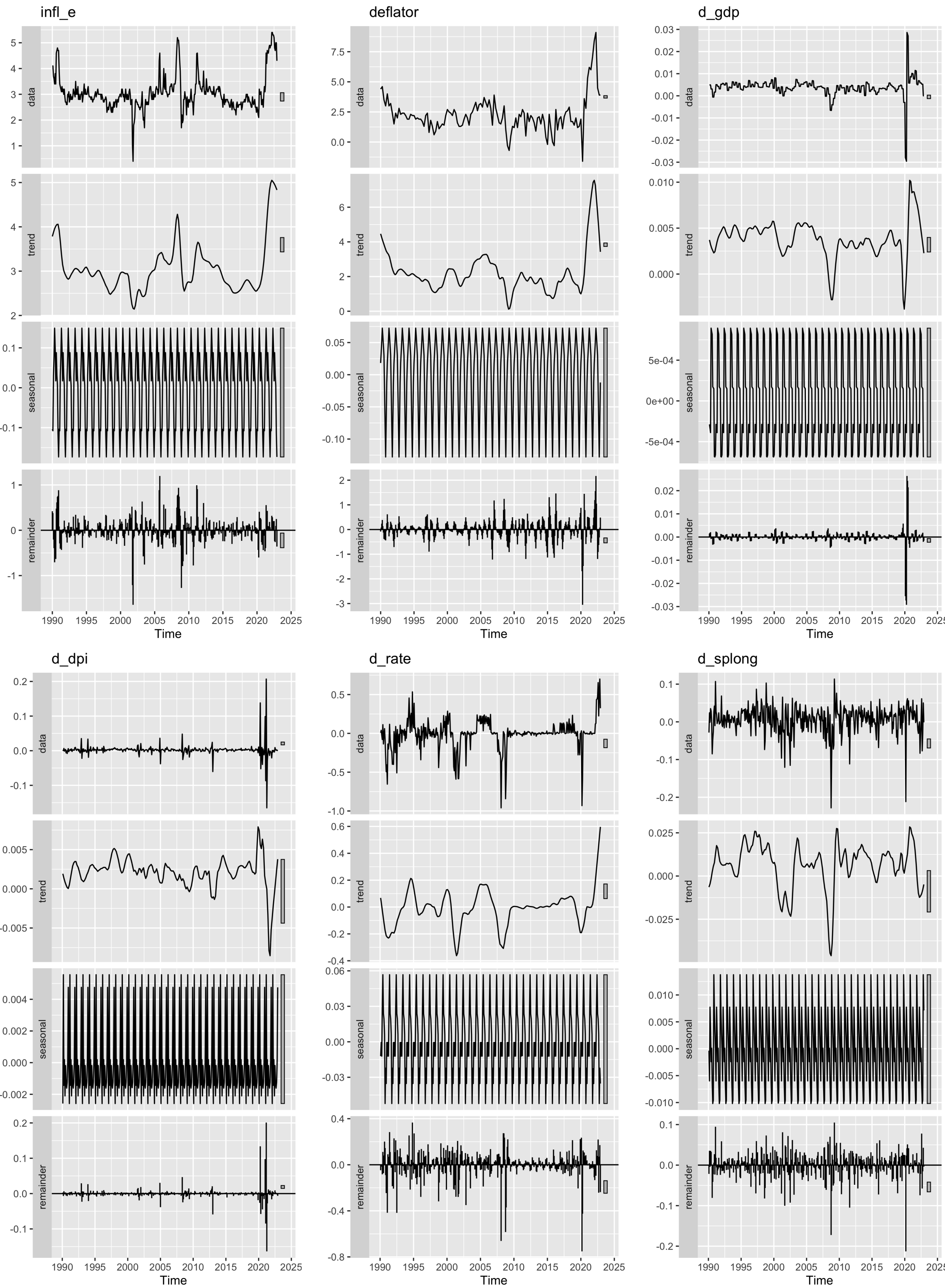


Figure 2: Time series decomposition of stationary series (levels and deltas)

significance.

Table 7: OLS decomposition of I(0) series

	Dependent variable:	
	infl_e	deflator
	(1)	(2)
trend	0.001**	0.002**
	(0.0003)	(0.001)
M1	0.051	0.083
	(0.166)	(0.347)
M2	0.050	0.097
	(0.166)	(0.347)
M3	0.192	0.112
	(0.166)	(0.347)
M4	0.239	0.127
	(0.166)	(0.347)
M5	0.311*	0.109
	(0.166)	(0.347)
M6	0.262	0.091
	(0.166)	(0.347)
M7	0.180	0.073
	(0.166)	(0.347)
M8	0.252	0.016
	(0.166)	(0.347)
M9	0.196	−0.041
	(0.166)	(0.347)
M10	0.174	−0.098
	(0.166)	(0.347)
M11	0.055	−0.049
	(0.166)	(0.347)
Constant	2.745***	1.899***
	(0.132)	(0.276)
Observations	396	396
R^2	0.036	0.018
Adjusted R^2	0.006	−0.012
Residual Std. Error (df = 383)	0.676	1.409
F Statistic (df = 12; 383)	1.208	0.600
Note:	*p<0.1; **p<0.05; ***p<0.01	

Table 8: OLS decomposition of the first difference of I(1) series

	Dependent variable:			
	d_gdp	d_dpi	d_rate	d_splong
	(1)	(2)	(3)	(4)
trend	−0.00000	−0.00000	0.0003***	−0.00000
	(0.00000)	(0.00001)	(0.0001)	(0.00002)
M1	0.0001	−0.005	0.019	−0.004
	(0.001)	(0.005)	(0.046)	(0.009)
M2	0.0004	−0.007	0.013	−0.008
	(0.001)	(0.005)	(0.045)	(0.009)
M3	0.0004	0.001	0.025	−0.013
	(0.001)	(0.005)	(0.045)	(0.009)
M4	0.0004	−0.006	0.014	0.001
	(0.001)	(0.005)	(0.045)	(0.009)
M5	0.002	−0.005	0.055	−0.004
	(0.001)	(0.005)	(0.045)	(0.009)
M6	0.002	−0.007	0.083*	−0.009
	(0.001)	(0.005)	(0.045)	(0.009)
M7	0.002	−0.005	0.051	−0.007
	(0.001)	(0.005)	(0.045)	(0.009)
M8	0.001	−0.006	0.048	−0.013
	(0.001)	(0.005)	(0.045)	(0.009)
M9	0.001	−0.006	0.036	−0.017*
	(0.001)	(0.005)	(0.045)	(0.009)
M10	0.001	−0.006	−0.021	−0.017*
	(0.001)	(0.005)	(0.045)	(0.009)
M11	0.00001	−0.005	0.011	0.007
	(0.001)	(0.005)	(0.045)	(0.009)
Constant	0.003***	0.007**	−0.091**	0.014**
	(0.001)	(0.004)	(0.036)	(0.007)
Observations	395	395	395	395
R^2	0.021	0.019	0.048	0.037
Adjusted R^2	−0.010	−0.012	0.018	0.007
Residual Std. Error (df = 382)	0.004	0.018	0.185	0.036
F Statistic (df = 12; 382)	0.668	0.602	1.598*	1.230
Note:				*p<0.1; **p<0.05; ***p<0.01

1.4 Cyclical component

To assess the behavior of the cyclical component, we detrend the series, substract the coefficient of the constant and remove the seasonal component when the coefficients are significant. After extracting these components, we would like to model them with appropiate $ARMA(p,q)$ models.

To do so, we first produce PACF (Figure 3) and ACF graphs (Figure 4) to know respectively the maximum values that p and q could take, $\bar{p}, \bar{q}$. The blue dotted lines onthe figures are the 95% confidence intervals. $\bar{p}, \bar{q}$ are given by the last period where the PACF or the ACF are significantly different from zero. We were surprised by the rather long memory of some of the processes, notably of the inflation expectation, the GDP deflator, and the first difference of the Fed fund rate series but we double checked and the series are stationary.

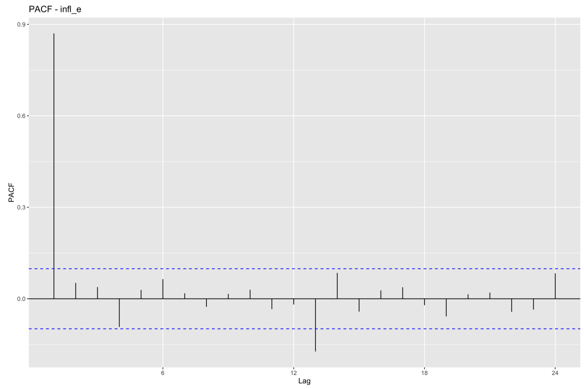
We then find the best parameters for each model by minimizing the BIC (an informational criteria) for each combination of $p \leq \bar{p}$ and $\leq \bar{q}$. This will allow us to find a combination of both types of models (AR and MA) such that we capture well the past with less parameters to estimate. To do so, we use R's `critMatrix` function with the option `criteria = 'bic'`. Importantly, this operation is slow in particular when $\bar{p}, \bar{q}$ are both large. In the case of the inflation expectation and the first difference of the Fed fund rate, their values were so large that R does not give any answer and reports an error. To solve this, we decided to take smaller values of $\bar{p}$. Indeed, by looking at the PACF graphs (Figure 3), we see that in both cases there is a single lag down the line that is significantly different from zero after several ones falling in the CI (lag 13 for *infl-e* and lag 15 for *d-rate*). We took the previous significant lag as $\bar{p}$ to be able to get the information criteria with the rationale that those "extreme" significant lags can be due to just luck, and that the combination of AR and MA models should allow to reduce the number of parameters.

We present the outputs of these comparitions tables per series in Appendix A because they are a lot and they were making hard to read this section. Remark that the palcement of the ps and qs can vary from one tabel to the next due to formatting issues. We find that the orders p and q that minimize the nformation criteria are:

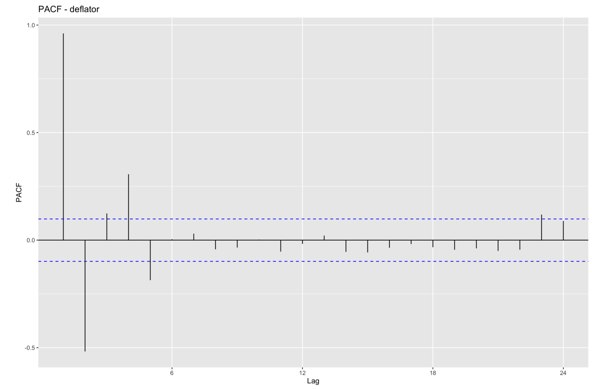
- Inflation expectation: $ARMA(1,0)$ (BIC = 236.04)
- GDP Deflator: $ARMA(2,1)$ (BIC = 164.23)
- GDP growth rate (*d-gdp*): $ARMA(0,2)$ (BIC = -3556.28)
- Real disposable income growth rate (*d-dpi*): $ARMA(2,1)$ (BIC=-2161.22)
- Fed Fund rate (*d-rate*): $ARMA(1,1)$ (BIC = -413.58)
- S&P500 rate (*d-splong*): $ARMA(0,1)$ (BIC = -1509.36)

With this information we can finally estimate these ARMA models using R's `arima` function. Since all the series are stationary, we just set $d = 0$ and the ARIMA models become ARMA. We get the following resuts:

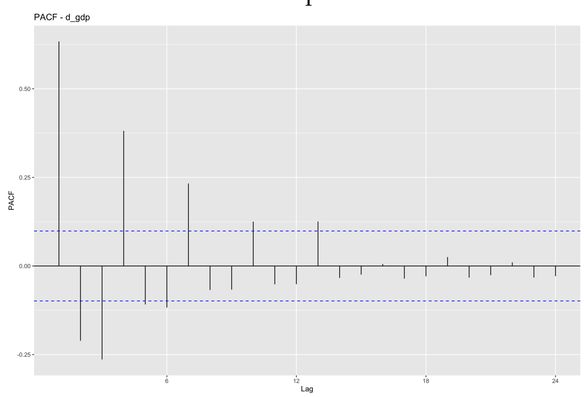
Finally, we check that the residuals of the models aren't serially correlated using the Ljung–Box test that is available with R's `Box.test` function. This test poses as a null hypothesis H0 that the data tested is independently



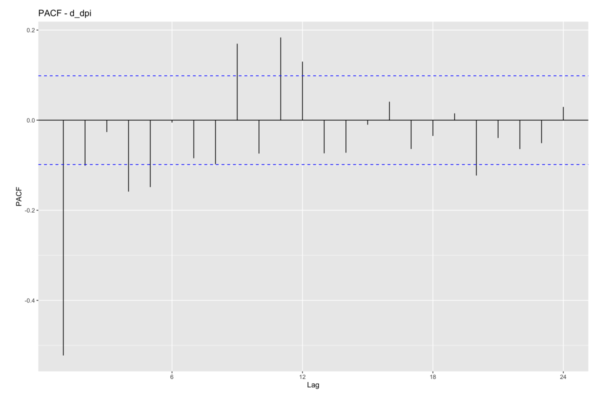
Inflation expectation



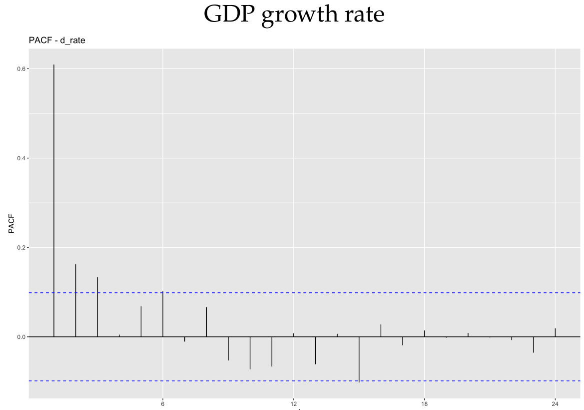
GDP deflator



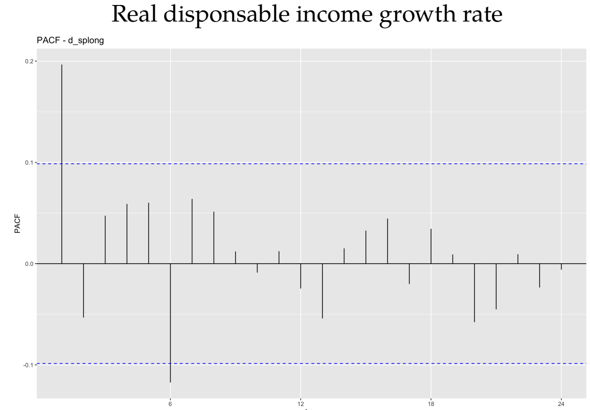
GDP growth rate



Real disponsable income growth rate



Fed fund rate



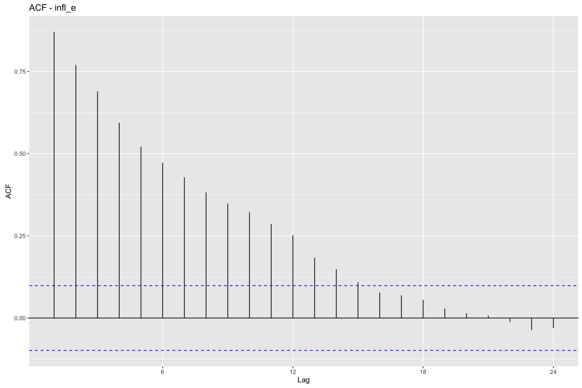
Returns of S&P500

Figure 3: PACFs

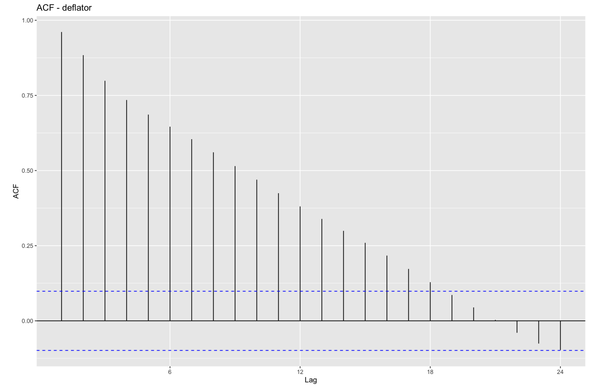
Notes: All the series plotted are stationary: inflation expectation and GDP deflator series are in levels while the rest are first differences.

distributed i.e. in this case, since we test the residuals, that there is no serial correlation between them. The test results are found in Table 10

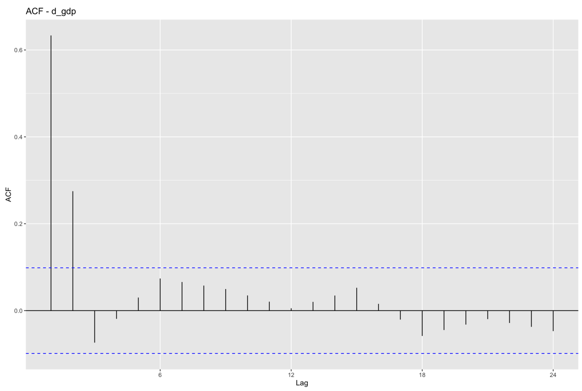
We cannot reject the null hypothesis in most of the models (p-values < 0.05) leading us to support the idea that for the inflation expectation, GDP growth, Fed fund rate and the return rate of the S&P500, the ARMA models that we proposed before correctly captures the auto-correlation structure of their cyclical components. However, we do reject H0 for the GDP deflator and the growth rate of disposable personal income suggesting



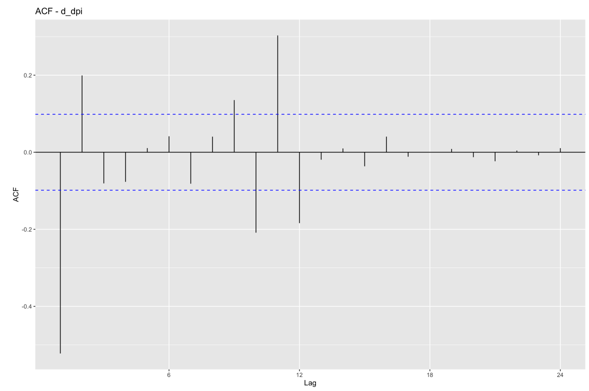
Inflation expectation



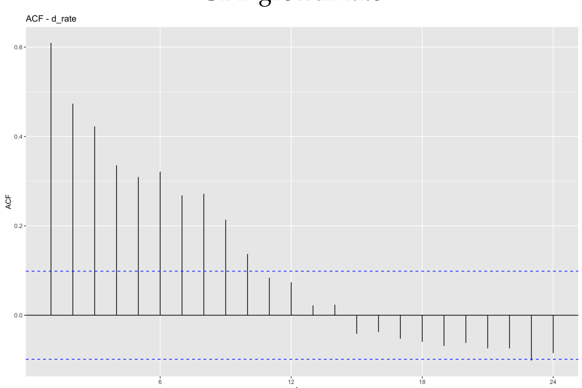
GDP deflator



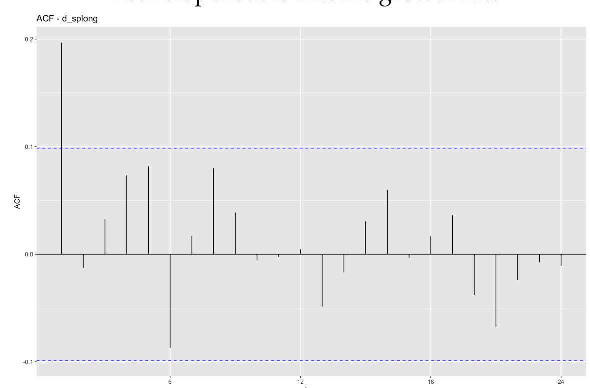
GDP growth rate



Real disponsable income growth rate



Fed fund rate



Returns of S&P500

Figure 4: ACFs

Notes: All the series plotted are stationary: inflation expectation and GDP deflator series are in levels while the rest are first differences.

that our models are not adequate. We couldn't find where the error might lie.

Table 9: ARMA model for the cyclical components

	Dependent variable:					
	infl_e	deflator	d_gdp	d_dpi	d_rate	d_splong
	(1)	(2)	(3)	(4)	(5)	(6)
ar1	0.881^{***}	-0.110^{***}		0.160	0.855^{***}	
	(0.024)	(0.024)		(0.107)	(0.045)	
ar2		0.886^{***}		0.268^{***}		
		(0.024)		(0.080)		
ma1		0.986^{***}	0.939^{***}	-0.770^{***}	-0.418^{***}	0.214^{***}
		(0.010)	(0.024)	(0.089)	(0.083)	(0.051)
ma2			0.881^{***}			
			(0.026)			
intercept	0.178	0.182	0.0004	-0.005^{***}	0.027	-0.005^{**}
	(0.134)	(0.140)	(0.0004)	(0.0003)	(0.028)	(0.002)
Observations	396	396	395	395	395	395
Log Likelihood	−115.027	−110.948	1,784.121	1,089.580	212.271	757.671
σ^2	0.104	0.102	0.00001	0.0002	0.020	0.001
Akaike Inf. Crit.	236.054	231.897	−3,560.241	−2,169.159	−416.542	−1,509.341
Note:						*p<0.1; **p<0.05; ***p<0.01

Table 10: Ljung-Box test: p-values

Model	p-value
infl_e	0.20
deflator	0.04
d_gdp	0.94
d_dpi	0.01
d_rate	0.08
d_splong	0.25

2 Canonical VAR model application

3 Cointegration theory

4 Impulse Response Analysis

4.1 Canonical IRF

4.2 Structural IRF

5 Introduce non-linearities

5.1 Markov-switching model

5.2 STR model

Appendix A Section 1 - Additional tables

Table 11: Information criteria on the parameters of ARMA for infl-e

	p=0	p=1	p=2	p=3	p=4
q=0	808.438	236.035	239.455	244.671	246.751
q=1	517.737	239.263	239.841	244.875	252.676
q=2	413.415	245.105	244.731	249.439	254.721
q=3	348.231	247.220	249.095	254.601	254.129
q=4	298.207	252.154	246.078	259.082	265.384
q=5	292.940	254.133	246.824	251.856	258.027
q=6	289.687	259.915	252.000	257.838	262.286
q=7	278.591	265.863	257.687	263.809	264.647
q=8	275.739	270.725	262.675	266.930	270.434
q=9	281.720	276.114	264.761	270.662	274.023
q=10	284.970	277.918	270.633	271.776	278.872
q=11	289.458	283.842	275.724	276.490	283.533
q=12	284.137	277.990	279.720	282.047	289.110
q=13	286.646	283.125	279.067	285.024	290.893
q=14	288.837	288.879	285.013	290.070	288.811
q=15	292.928	294.809	290.843	306.880	298.582

Table 12: Information criteria on the parameters of ARMA for GDP deflator

	p=0	p=1	p=2	p=3	p=4
q=0	808.438	236.035	239.455	244.671	246.751
q=1	517.737	239.263	239.841	244.875	252.676
q=2	413.415	245.105	244.731	249.439	254.721
q=3	348.231	247.220	249.095	254.601	254.129
q=4	298.207	252.154	246.078	259.082	265.384
q=5	292.940	254.133	246.824	251.856	258.027
q=6	289.687	259.915	252.000	257.838	262.286
q=7	278.591	265.863	257.687	263.809	264.647
q=8	275.739	270.725	262.675	266.930	270.434
q=9	281.720	276.114	264.761	270.662	274.023
q=10	284.970	277.918	270.633	271.776	278.872
q=11	289.458	283.842	275.724	276.490	283.533
q=12	284.137	277.990	279.720	282.047	289.110
q=13	286.646	283.125	279.067	285.024	290.893
q=14	288.837	288.879	285.013	290.070	288.811
q=15	292.928	294.809	290.843	306.880	298.582

Table 13: Information criteria on the parameters of ARMA for d-gdp

	q=0	q=1	q=2
p=0	-3, 196.342	-3, 335.595	-3, 556.284
p=1	-3, 393.003	-3, 395.734	-3, 550.426
p=2	-3, 404.863	-3, 405.710	-3, 544.540
p=3	-3, 427.848	-3, 456.107	-3, 540.254
p=4	-3, 484.507	-3, 481.070	-3, 534.314
p=5	-3, 483.226	-3, 478.455	-3, 528.379
p=6	-3, 483.225	-3, 488.833	-3, 523.609
p=7	-3, 500.495	-3, 495.556	-3, 517.633
p=8	-3, 496.418	-3, 490.948	-3, 511.656
p=9	-3, 492.678	-3, 490.332	-3, 507.291
p=10	-3, 493.990	-3, 488.534	-3, 501.325
p=11	-3, 489.174	-3, 483.437	-3, 495.360
p=12	-3, 484.524	-3, 482.220	-3, 489.569
p=13	-3, 485.664	-3, 480.011	-3, 483.611

Table 14: Information criteria on the parameters of ARMA for d-dpi

	q=0	q=1	q=2	q=3	q=4	q=5	q=6	q=7	
p=0	-2, 040.006	-2, 159.574	-2, 155.516	-2, 167.273	-2, 161.611	-2, 157.417	-2, 151.693	-2, 150.189	-2
p=1	-2, 159.554	-2, 158.813	-2, 154.793	-2, 161.479	-2, 159.258	-2, 155.433	-2, 145.472	-2, 149.050	-2
p=2	-2, 157.605	-2, 161.222	-2, 155.416	-2, 156.642	-2, 163.780	-2, 162.047	-2, 143.475	-2, 151.355	-2
p=3	-2, 151.898	-2, 155.614	-2, 152.945	-2, 164.239	-2, 159.070	-2, 154.684	-2, 151.312	-2, 146.238	-2
p=4	-2, 155.878	-2, 157.819	-2, 158.010	-2, 160.097	-2, 159.219	-2, 165.859	-2, 150.022	-2, 155.182	-2
p=5	-2, 158.597	-2, 153.039	-2, 147.634	-2, 154.626	-2, 166.740	-2, 161.024	-2, 155.178	-2, 151.351	-2
p=6	-2, 152.629	-2, 147.327	-2, 141.825	-2, 149.321	-2, 155.802	-2, 154.798	-2, 151.579	-2, 146.862	-2
p=7	-2, 149.433	-2, 144.239	-2, 152.828	-2, 143.108	-2, 148.209	-2, 142.850	-2, 155.557	-2, 150.118	-2
p=8	-2, 147.169	-2, 152.831	-2, 143.483	-2, 143.202	-2, 149.709	-2, 144.433	-2, 149.815	-2, 135.057	-2
p=9	-2, 152.503	-2, 145.737	-2, 147.194	-2, 141.698	-2, 148.990	-2, 138.511	-2, 132.540	-2, 140.389	-2
p=10	-2, 148.638	-2, 141.516	-2, 141.326	-2, 139.324	-2, 147.074	-2, 144.611	-2, 144.907	-2, 142.275	-2
p=11	-2, 155.935	-2, 153.152	-2, 154.028	-2, 149.088	-2, 143.284	-2, 137.544	-2, 132.018	-2, 126.041	-2
p=12	-2, 156.684	-2, 151.663	-2, 149.126	-2, 143.180	-2, 146.342	-2, 141.996	-2, 126.053	-2, 123.091	-2

Table 15: Information criteria on the parameters of ARMA for d-rate

	p=0	p=1	p=2	p=3	p=4	p=5	p=6
q=0	-220.437	-398.928	-405.474	-407.769	-401.798	-398.623	-397.967
q=1	-336.159	-412.584	-409.586	-403.731	-401.328	-395.918	-396.298
q=2	-357.789	-409.855	-404.200	-398.327	-395.988	-397.849	-392.506
q=3	-381.064	-403.959	-398.280	-398.236	-391.210	-385.390	-388.208
q=4	-385.108	-399.279	-393.396	-391.439	-385.366	-380.777	-386.024
q=5	-380.280	-393.931	-394.569	-386.948	-379.522	-379.390	-384.388
q=6	-381.165	-394.107	-394.053	-394.640	-389.631	-382.456	-378.467
q=7	-375.560	-388.690	-387.899	-389.048	-375.038	-371.914	-368.204
q=8	-376.117	-391.305	-386.325	-380.439	-374.871	-372.999	-367.089
q=9	-379.145	-386.439	-380.464	-374.571	-369.119	-368.622	-367.891
q=10	-381.673	-380.470	-380.584	-375.879	-369.271	-367.834	-357.236

Table 16: Information criteria on the parameters of ARMA for d-splong

	q=0	q=1
p=0	-1,498.418	-1,509.363
p=1	-1,508.029	-1,503.624
p=2	-1,503.196	-1,498.958
p=3	-1,498.133	-1,493.386
p=4	-1,493.534	-1,487.872
p=5	-1,488.927	-1,486.959
p=6	-1,488.534	-1,483.508

Appendix B Code - Data Cleaning

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
Financial Econometrics - Empirical Applications d
Data Gathering and Data Cleaning

@author: nataliacardenasf
"""

import pandas as pd
import os
import datetime

from fredapi import Fred

os.chdir('/Users/nataliacardenasf/Documents/GitHub/PROJECTS_AP_FE/FinancialEconometrics1')

### Initialize FRED API
fred = Fred(api_key='23edc2b1b61e17c07b83a97e7abfc02b')

### Import all the data

# S&P 500
sp500 = pd.DataFrame(fred.get_series('SP500')) #daily close, NSA, Index
sp500.columns= ['sp500']

#Inflation expectations from survey UMich
infl_e = pd.DataFrame(fred.get_series('MICH')) #monthly, NSA, median expected in % over next 12 mo
infl_e.columns= ['infl_e']


#ICE BofA US Corporate Index Total Return Index
corp_debt = pd.DataFrame(fred.get_series('BAMLCC0A0CMTRIV')) #daily, close, NSA, Index
```

```python
corp_debt.columns= ['corp_debt']


#MP rate
rate = pd.DataFrame(fred.get_series('DFF')) #daily, 7-Day, NSA, %
rate.columns = ['rate']

#Deflator
deflator = pd.DataFrame(fred.get_series('A191RI1Q225SBEA')) #Q, SA Annual Rate
deflator.columns = ['deflator']

#Unemployment
unempl = pd.DataFrame(fred.get_series('UNRATENSA')) #monthly, NSA, %
unempl.columns=['unempl']

# GDP
gdp = pd.DataFrame(fred.get_series('GDP'))  # quarterly, Billions of Dollars, SA Annual Rate
gdp.columns = ['gdp']

# RPI (Real Personal Income)
rpi = pd.DataFrame(fred.get_series('RPI'))  # monthly, SA rate, deflated
rpi.columns = ['rpi']

# Real Personal Disposable Income
dpi = pd.DataFrame(fred.get_series('DSPIC96'))  # monthly, SA annaul rate, chained 2017 USD
dpi.columns = ['dpi']

# Manufacturing Sector
manufacturing = pd.DataFrame(fred.get_series('MPU9900063'))  # annual, NSA => avoid this
manufacturing.columns = ['manufacturing']

fred.search("MPU9900063").T #this function gives of the info on every series


### Resample into monthly data
sp500 = sp500.resample('1M').mean(numeric_only=True)
infl_e = infl_e.resample('1M').mean(numeric_only=True)
corp_debt = corp_debt.resample('1M').mean(numeric_only=True)
rate = rate.resample('1M').mean(numeric_only=True)
deflator = deflator.resample('1M').mean(numeric_only=True)
unempl = unempl.resample('1M').mean(numeric_only=True)
gdp = gdp.resample('1M').mean(numeric_only=True)
rpi = rpi.resample('1M').mean(numeric_only=True)
dpi = dpi.resample('1M').mean(numeric_only=True)
manufacturing = manufacturing.resample('1M').mean(numeric_only=True)

dta = [infl_e, rate, sp500, corp_debt, deflator, unempl, gdp, rpi, dpi, manufacturing]

```

```
### Slice the df to relevant period
#Find common time span
min_date = max([min(i.index) for i in dta])
max_date = min([max(i.index) for i in dta])
print(min_date, max_date)

#Let us work on monthly data for the 1990-2022 period
start = datetime.datetime(1990,1,1)
end= datetime.datetime(2022,12,31)

## SP500 series is too short, I am taking it from Yahoo Finance
import yfinance as yf
splong = yf.download('^GSPC', start=start,end=end)['Adj Close'].resample('M').mean(
    numeric_only=True)
splong = pd.DataFrame(splong)
type(splong)
splong.rename(columns={"Adj Close":'splong'}, inplace=True)

##Get a single DF
dta.append(splong)
for i in range(len(dta)): #we had some indexes at end of month, others at 1st of month:
    harmonize to 1st each month
    df = dta[i]
    df.index = [pd.datetime(x.year, x.month, 1) for x in df.index.tolist()]
    dta[i] = df.loc[start:end,:]
dta# we're good now
#merge into 1 df, 1 series per column
monthly = pd.concat(dta, axis=1)
#interpolate missing months for deflatior data (Q): uses midpoints ie assumes that each month
    in the quarter contributes in the same fashion to the increase QoQ
m1 = monthly.interpolate(method ='linear', limit_direction ='forward')


m1.to_csv("DATA/data.csv")
```

Appendix C Code - Analysis 1

```r
##%% FE 1 v3
##%% @ncardenasfrias

# Load necessary packages and set personal path to documents
pacman::p_load(data.table, tseries, smoots, dplyr, bootUR, urca, gridExtra, tidyverse, gplots, xts, stargazer, forecast, ggplot2, vars)

setwd('/Users/nataliacardenasf/Documents/GitHub/PROJECTS_AP_FE/FinancialEconometrics1')

##########################
#0. Preprocessing
##########################

#Load the dataset, data gathering and cleaning done in Python
df = fread("DATA/data.csv")
as.data.table(df)
#change few names of columns to got something more clear
names(df)[1] = 'Date'


#Apply log to variables for which it makes sense ie prices and quantities
df$sp500 = log(df$sp500) #get the stock market return
df$splong = log(df$splong)
df$corp_debt = log(df$corp_debt) #return corporate debt (see index definition)
df$gdp = log(df$gdp)
df$rpi  = log(df$rpi)
df$dpi  = log(df$dpi)


#convert into TS
allts = list()
numeric_cols = df[, sapply(df, is.numeric)&names(df)!= 'Date'] #identifies the right columns
for(col in names(numeric_cols)){
  allts[[col]] = ts(df[[col]], start= c(year(df$Date[1]), month(df$Date[1])), frequency=12)
}

allts[['sp500']] = NULL #remove short SP500, use YahooFinance series that has the 90's and 00's data
allts[['Date']] = NULL # R was making regressions on the date column :/
#remove columns we ended up not using in the final version
allts[['manufacturing']] = NULL # rather not have data that was initially yearly
allts[['rpi']] = NULL # redundent with dpi
allts[['unempl']] = NULL # did not lead anywhere
allts[['corp_debt']] = NULL # we end up not using it and the UR test are weird


```

```
#Reorganise the list
desired_order = c('gdp', 'dpi', 'infl_e', 'deflator', 'rate', 'splong')
allts = allts[desired_order]
names = c('GDP', "Disposable Income", 'Inflation expectation', 'PIB deflator', 'Fed rate', '
    SP500')
rm(df, desired_order)

#Remove outliers => IT DOES NOT CHANGE ANYTHING
# tsclean(allts$infl_e, iterate = 2, lambda =NULL)
# tsclean(allts$deflator, iterate = 2, lambda =NULL)
# tsclean(allts$unempl, iterate = 2, lambda =NULL)
# tsclean(allts$rate, iterate = 2, lambda =NULL)
# tsclean(allts$splong, iterate = 2, lambda =NULL)
# tsclean(allts$corp_debt, iterate = 2, lambda =NULL)




#################################
#1.UR TEST - ADF, full procedure
#################################

### ADF 1st regression: deterministic trend + drift
results_adf_trend = list()
for (var_name in names(allts)) {
  result = ur.df(allts[[var_name]], type = "trend", selectlags = "BIC")
  results_adf_trend[[var_name]] = summary(result)
}
#export summary adf
trend_test =cbind(t(results_adf_trend$gdp@teststat), t(results_adf_trend$dpi@teststat),
                  t(results_adf_trend$infl_e@teststat),t(results_adf_trend$deflator@teststat),
                  t(results_adf_trend$rate@teststat), t(results_adf_trend$splong@teststat),
                  results_adf_trend$infl_e@cval)
colnames(trend_test) = c(names(results_adf_trend), "CV 1pct", "CV 5pct", "CV 10pct")
stargazer(trend_test, type='text')
# stargazer(trend_test, out='TABLES/adf_trend.tex', label= 'tab:adftrend_hyp',title = "ADF
    test - 1st regression with drift, deterministic trend and stochastic trend")

#export all t-values on coefficients, harder because output format
#create empty df with columns = rhs variable in ADF
t_values_table = data.frame(matrix(nrow = 0, ncol = length(c('Intercept', 'z.lag.1', 'tt', 'z.
    diff.lag'))))
colnames(t_values_table) = c('alpha', 'gamma', 'beta', 'rho')
#iterate over the summaries and extract the t stats for each series
for (i in names(results_adf_trend)){
  j = results_adf_trend[[i]]@testreg$coefficients[,'t value']
  tab = as.data.frame(j)
  row = c(tab$j[1], tab$j[2], tab$j[3], tab$j[4])
  t_values_table[nrow(t_values_table) + 1,] = row
```

```
}
row.names(t_values_table) = names(results_adf_trend) # call the rows as the series
t_values_table = t(t_values_table) #transpose the table, looks better
stargazer(t_values_table,type='text') #all ok
# stargazer(t_values_table,out="TABLES/adf_tstats_trend.tex", title="ADF test - 1st regression t statistics",
#          notes = '\\footnotesize Notes: With N=396, critical values at 5\\%: alpha = 3.09 ; gamma= -3.43 ; beta = 2.79',
#         label='tab:tstat_trend')


#extract series that need to keep being tested ie all but deflator and inflation expectation
adf_v2 = allts[c(1:2, 5:6)]
names(adf_v2) = names(allts)[c(1:2, 5:6)]



### ADF 2nd regression: drift
results_adf_drift = list()
for (var_name in names(adf_v2)) {
  result = ur.df(adf_v2[[var_name]], type = "drif", selectlags = "BIC")
  results_adf_drift[[var_name]] = summary(result)
}
#export summary adf
drift_test =cbind(t(results_adf_drift$gdp@teststat), t(results_adf_drift$dpi@teststat),
                  t(results_adf_drift$rate@teststat), t(results_adf_drift$splong@teststat),
                  results_adf_drift$splong@cval)
colnames(drift_test) = c(names(results_adf_drift), "CV 1pct", "CV 5pct", "CV 10pct")
stargazer(drift_test, type='text')
# stargazer(drift_test, out='TABLES/adf_drift.tex', label= 'tab:adfdrift_hyp',title = "ADF test - 2nd regression with drift and stochastic trend")

#export all t-values on coefficients, harder because output format
#create empty df with columns = rhs variable in ADF
t_values_table2 = data.frame(matrix(nrow = 0, ncol = length(c('Intercept', 'z.lag.1', 'z.diff.lag'))))
colnames(t_values_table2) = c('alpha', 'gamma', 'rho')
#iterate over the summaries and extract the t stats for each series
for (i in names(results_adf_drift)){
  j = results_adf_drift[[i]]@testreg$coefficients[,'t value']
  tab = as.data.frame(j)
  row = c(tab$j[1], tab$j[2], tab$j[3])
  t_values_table2[nrow(t_values_table2) + 1,] = row
}
row.names(t_values_table2) = names(results_adf_drift) # call the rows as the series
t_values_table2 = t(t_values_table2) #transpose the table, looks better
stargazer(t_values_table2,type='text') #all ok
# stargazer(t_values_table2,out="TABLES/adf_tstats_drift.tex", title="ADF test - 2nd
```

```
    regression t statistics",
#           notes = '\\footnotesize Notes: With N=396, critical values at 5\\%: alpha = 2.53 ;
     gamma= -2.88',
#           label='tab:tstat_drift')



adf_v3 = adf_v2 #series to keep testing (They are the same but oh well, I had a different
    pipeline with the other series)
names(adf_v3) = names(adf_v2)


### ADF 3rd regression: UR only
results_adf_none = list()
for (var_name in names(adf_v3)) {
  result = ur.df(adf_v3[[var_name]], type = "none", selectlags = "BIC")
  results_adf_none[[var_name]] = summary(result)
}
#export summary adf
none_test =cbind(t(results_adf_none$gdp@teststat),
                 t(results_adf_none$dpi@teststat), t(results_adf_none$rate@teststat),
                 t(results_adf_none$splong@teststat), results_adf_none$rate@cval)
colnames(none_test) = c(names(results_adf_none), "CV 1pct", "CV 5pct", "CV 10pct")
stargazer(none_test, type='text')
# stargazer(none_test, out='TABLES/adf_none.tex', label= 'tab:adfnone_hyp',title = "ADF test -
     3rd regression with stochastic trend")


#export all t-values on coefficients, harder because output format
#create empty df with columns = rhs variable in ADF
t_values_table3 = data.frame(matrix(nrow = 0, ncol = length(c('z.lag.1', 'z.diff.lag'))))
colnames(t_values_table3) = c('gamma', 'rho')
#iterate over the summaries and extract the t stats for each series
for (i in names(results_adf_none)){
  j = results_adf_none[[i]]@testreg$coefficients[,'t value']
  tab = as.data.frame(j)
  row = c(tab$j[1], tab$j[2], tab$j[3])
  t_values_table3[nrow(t_values_table3) + 1,] = row
}
row.names(t_values_table3) = names(results_adf_none) # call the rows as the series
t_values_table3 = t(t_values_table3) #transpose the table, looks better
stargazer(t_values_table3,type='text') #all ok
# stargazer(t_values_table3,out="TABLES/adf_tstats_none.tex", title="ADF test - 3rd regression
     t statistics",
#           notes = '\\footnotesize Notes: With N=396, critical values at 5\\%: gamma= -1.95',
#           label='tab:tstat_none')
```

```r

## Get deltas
deltas = list (diff(allts$gdp), diff(allts$dpi),
               diff(allts$rate),diff(allts$splong))
names(deltas) = list("d_gdp","d_dpi","d_rate", "d_splong")

### CHECK THAT DELTAS ARE I(0), I am using same code as in levels
results_adf_trend = list()
for (var_name in names(deltas)) {
  result = ur.df(deltas[[var_name]], type = "trend", selectlags = "BIC")
  results_adf_trend[[var_name]] = summary(result)
}
#export summary adf
trend_deltas =cbind(t(results_adf_trend$d_gdp@teststat), #t(results_adf_trend$d_rpi@teststat),
                    t(results_adf_trend$d_dpi@teststat), t(results_adf_trend$d_rate@teststat),
                    t(results_adf_trend$d_splong@teststat), results_adf_trend$d_rate@cval)
colnames(trend_deltas) = c(names(results_adf_trend), "CV 1pct", "CV 5pct", "CV 10pct")
stargazer(trend_deltas, type='text')
#stargazer(trend_deltas, out='TABLES/adf_deltas.tex', label= 'tab:adfdeltas_hyp',title = "ADF
    test - 1st regression with drift, deterministic trend and stochastic trend for series in
    deltas")




#################################
#2. Decompositions in levels
#################################
# use stl function to perform the decompositions and then OLS to estimate parameters

#### Series in levels that are stationary.
i0_levels = list(allts$infl_e, allts$deflator) #from adf!
names(i0_levels) = c('infl_e', "deflator")

decompositions = list() #store the decompositions
dec_graphs= list() #store the decomposition graphs
deseasonalized_ts = list() #store the deseasonalized ts, if seasonality ends up being
    important
for (ts in 1:length(i0_levels)) {
  #print(i0_levels[ts])}
  decomp = stl(i0_levels[[ts]], s.window='periodic') #additive seasonality seems right from
    graphs
  graph = autoplot(decomp) + labs(title = names(i0_levels)[ts])
  deseason = i0_levels[[ts]] - decomp$time.series[,'seasonal']
  decompositions[[length(decompositions)+1]] = decomp
  dec_graphs[[length(dec_graphs)+1]] = graph
  deseasonalized_ts[[length(deseasonalized_ts)+1]] = deseason
```

```r
}
names(deseasonalized_ts) = names(i0_levels) #name the series in deseasonalized_ts
names(decompositions) = names(i0_levels) #name the series in decompositions

rm(graph, decomp, deseason)

#Generate and export plots with the decomposition graphs
decom_i = grid.arrange(dec_graphs[[1]],dec_graphs[[2]], ncol=2)
# ggsave('IMAGES/decomposition_i.png', plot=decom_i, width = 12, height = 8)

### Run OLS regression on each component of the TS
#create a df with all the series and the monthly dummies
start_date = as.Date("1990-01-01")
end_date = as.Date("2022-12-01") # Assuming December 2022
all_dates = seq(start_date, end_date, by = "month")

df_i0level = data.frame(Date = all_dates)
for (i in seq_along(i0_levels)) {
  col_name = paste0("Series_", i)
  df_i0level[[col_name]] = i0_levels[[i]]
}
colnames(df_i0level) = c('Date', names(i0_levels))
rownames(df_i0level) = df_i0level$Date #date as index
df_i0level$MONTH=month(df_i0level$Date) #Get month dummies
df_i0level[paste0("M", 1:12)] = as.data.frame(t(sapply(df_i0level$MONTH, tabulate, 12)))

df_i0level$Date = NULL #no more need date column
df_i0level$MONTH = NULL #no more need MONTH column

df_i0level$trend = seq_along(df_i0level$infl_e) #deterministic trend

# Run OLS regression
infl_e_ols_dec = lm(infl_e ~ trend + M1 + M2 + M3 + M4 + M5 + M6 + M7 + M8 + M9 + M10 + M11,
    data = df_i0level)
deflator_ols_dec = lm(deflator ~ trend + M1 + M2 + M3 + M4 + M5 + M6 + M7 + M8 + M9 + M10 +
    M11, data = df_i0level)

stargazer(infl_e_ols_dec,deflator_ols_dec, type='text')
# stargazer(infl_e_ols_dec,deflator_ols_dec,
#           type='latex', out="TABLES/ols_decomp_levels", label='tab:ols_dec_levels',
#           title='OLS decomposition of I(0) series' )


## remove drifts and trend and store in new list
i0_levels_no_drift = list()

ct_infl = coef(infl_e_ols_dec)[1]
trend_infl = coef(infl_e_ols_dec)[2]
```

```
m5_infl = coef(infl_e_ols_dec)[7]
i0_levels_no_drift$infl_e = i0_levels$infl_e - ct_infl - (df_i0level$trend*trend_infl) - (df_
    i0level$M5*m5_infl)

ct_defl = coef(deflator_ols_dec)[1]
trend_defl = coef(deflator_ols_dec)[2]
i0_levels_no_drift$deflator = i0_levels$deflator - ct_defl- (df_i0level$trend*trend_defl)


### Find p and q with PACF and ACF

acf_infl = ggAcf(i0_levels_no_drift$infl_e, lag.max= 24) + labs(title = 'ACF - infl_e')
acf_defl= ggAcf(i0_levels_no_drift$deflator, lag.max= 24) + labs(title = 'ACF - deflator')

ggsave('IMAGES/acf_infl.png', plot=acf_infl, width = 12, height = 8)
ggsave('IMAGES/acf_defl.png', plot=acf_defl, width = 12, height = 8)

pacf_infl = ggPacf(i0_levels_no_drift$infl_e, lag.max= 24)+ labs(title = 'PACF - infl_e')
pacf_defl = ggPacf(i0_levels_no_drift$deflator, lag.max= 24) + labs(title = 'PACF - deflator')

ggsave('IMAGES/pacf_infl.png', plot= pacf_infl, width = 12, height=8)
ggsave('IMAGES/pacf_defl.png', plot= pacf_defl, width = 12, height=8)


### Minimize information criteria
#inflation expectation
arma_bic_infl = critMatrix(i0_levels_no_drift$infl_e, p.max = 4, q.max = 15, criterion='bic')
stargazer(arma_bic_infl, type='text', flip=T)
# starg azer(arma_bic_infl, type='latex', flip=T,
#            out= 'TABLES/BIC_arma_infl.tex', label="tab:bic_infl",
#            title= "Information criteria on the parameters of ARMA for infl-e")

#deflator
arma_bic_defl = critMatrix(i0_levels_no_drift$deflator, p.max = 5, q.max = 18, criterion='bic'
    )
stargazer(arma_bic_defl, type='text', flip=T)
# stargazer(arma_bic_infl, type='latex', flip=T,
#            out= 'TABLES/BIC_arma_deflator.tex', label="tab:bic_deflator",
#            title= "Information criteria on the parameters of ARMA for GDP deflator")


## fit ARMA model
arma_infl = arima(i0_levels_no_drift$infl_e, order= c(1,0,0))
stargazer(arma_infl, type='text')

arma_defl = arima(i0_levels_no_drift$infl_e, order= c(2,0,1))
stargazer(arma_defl, type='text')

```

```
###################################
#3. Decompositions in deltas
###################################
# Same idea for series in deltas

decompositions_d = list() #store the decompositions
dec_graphs_d= list() #store the decomposition graphs
deseasonalized_ts_d = list() #store the deseasonalized ts, if seasonality ends up being
    important
for (ts in 1:length(deltas)) {
  #print(deltas[ts])}
  decomp = stl(deltas[[ts]], s.window='periodic') #additive seasonality seems right from
    graphs
  graph = autoplot(decomp) + labs(title = names(deltas)[ts])
  deseason = deltas[[ts]] - decomp$time.series[,'seasonal']
  decompositions_d[[length(decompositions_d)+1]] = decomp
  dec_graphs_d[[length(dec_graphs_d)+1]] = graph
  deseasonalized_ts_d[[length(deseasonalized_ts_d)+1]] = deseason
}
names(deseasonalized_ts_d) = names(deltas) #name the series in deseasonalized_ts
names(decompositions_d) = names(deltas) #name the series in decompositions
rm(graph, decomp, deseason)

#Generate and export plots with the decomposition graphs
decom_ii = grid.arrange(dec_graphs_d[[1]],dec_graphs_d[[2]], dec_graphs_d[[3]],dec_graphs_d
    [[4]], ncol=2)
# ggsave('IMAGES/decomposition_ii.png', plot=decom_ii, width = 12, height = 16)

##Create single image with all decompositions, looks better on latex
all_dec = grid.arrange(dec_graphs[[1]], dec_graphs[[2]], dec_graphs_d[[1]],dec_graphs_d[[2]],
    dec_graphs_d[[3]],dec_graphs_d[[4]], ncol=3)
# ggsave('IMAGES/all_decompositions.png', plot=all_dec, width = 12, height = 16)



### Run OLS regression on each component of the TS
#create a df with all the series and the monthly dummies
start_date = as.Date("1990-02-01")
end_date = as.Date("2022-12-01")
all_dates = seq(start_date, end_date, by = "month")

df_deltas = data.frame(Date = all_dates)
for (i in seq_along(deltas)) {
  col_name = paste0("Series_", i)
  df_deltas[[col_name]] = deltas[[i]]
}
colnames(df_deltas) = c('Date', names(deltas))
```

```r
rownames(df_deltas) = df_deltas$Date #date as index
df_deltas$MONTH=month(df_deltas$Date) #Get month dummies
df_deltas[paste0("M", 1:12)] = as.data.frame(t(sapply(df_deltas$MONTH, tabulate, 12)))

df_deltas$Date = NULL #no more need date column
df_deltas$MONTH = NULL #no more need MONTH column

df_deltas$trend = seq_along(df_deltas$d_gdp) #deterministic trend

# Run OLS regression
d_gdp_ols_dec = lm(d_gdp ~ trend + M1 + M2 + M3 + M4 + M5 + M6 + M7 + M8 + M9 + M10 + M11,
    data = df_deltas)
d_dpi_ols_dec = lm(d_dpi ~ trend + M1 + M2 + M3 + M4 + M5 + M6 + M7 + M8 + M9 + M10 + M11,
    data = df_deltas)
d_rate_ols_dec = lm(d_rate ~ trend + M1 + M2 + M3 + M4 + M5 + M6 + M7 + M8 + M9 + M10 + M11,
    data = df_deltas)
d_splong_ols_dec = lm(d_splong ~ trend + M1 + M2 + M3 + M4 + M5 + M6 + M7 + M8 + M9 + M10 +
    M11, data = df_deltas)

stargazer(d_gdp_ols_dec,d_dpi_ols_dec,d_rate_ols_dec, d_splong_ols_dec, type='text')
# stargazer(d_gdp_ols_dec,d_dpi_ols_dec,d_rate_ols_dec, d_splong_ols_dec,
#           type='latex', out="TABLES/ols_decomp_deltas", label='tab:ols_dec_deltas',
#           title='OLS decomposition of the first difference of I(1) series' )

## remove drifts and store in new list (remove trend for rate only)
deltas_no_drift_trend = list()

ct_gdp = coef(d_gdp_ols_dec)[1]
deltas_no_drift_trend$d_gdp = deltas$d_gdp - ct_gdp

ct_dpi = coef(d_dpi_ols_dec)[1]
deltas_no_drift_trend$d_dpi = deltas$d_dpi - ct_dpi

ct_rate = coef(d_rate_ols_dec)[1]
trend_rate = coef(d_rate_ols_dec)[2]
m6_rate = coef(d_rate_ols_dec)[8]
deltas_no_drift_trend$d_rate = deltas$d_rate - ct_rate - (df_deltas$trend * trend_rate) - (m6_
    rate*df_deltas$M6)

ct_sp = coef(d_splong_ols_dec)[1]
m9_sp = coef(d_splong_ols_dec)[11]
m10_sp = coef(d_splong_ols_dec)[12]
deltas_no_drift_trend$d_splong = deltas$d_splong - ct_sp - (m9_sp*df_deltas$M9) - (m10_sp*df_
    deltas$M10)


### Find p and q with PACF and ACF

```

```
acf_gdp = ggAcf(deltas_no_drift_trend$d_gdp, lag.max= 24) + labs(title = 'ACF - d_gdp')
acf_dpi = ggAcf(deltas_no_drift_trend$d_dpi, lag.max= 24) + labs(title = 'ACF - d_dpi')
acf_rate = ggAcf(deltas_no_drift_trend$d_rate, lag.max= 24) + labs(title = 'ACF - d_rate')
acf_sp = ggAcf(deltas_no_drift_trend$d_splong, lag.max= 24) + labs(title = 'ACF - d_splong')

ggsave('IMAGES/acf_gdp.png', plot=acf_gdp, width = 12, height = 8)
ggsave('IMAGES/acf_dpi.png', plot=acf_dpi, width = 12, height = 8)
ggsave('IMAGES/acf_rate.png', plot=acf_rate, width = 12, height = 8)
ggsave('IMAGES/acf_sp.png', plot=acf_sp, width = 12, height = 8)

pacf_gdp = ggPacf(deltas_no_drift_trend$d_gdp, lag.max= 24)+ labs(title = 'PACF - d_gdp')
pacf_dpi = ggPacf(deltas_no_drift_trend$d_dpi, lag.max= 24)+ labs(title = 'PACF - d_dpi')
pacf_rate = ggPacf(deltas_no_drift_trend$d_rate, lag.max= 24)+ labs(title = 'PACF - d_rate')
pacf_sp = ggPacf(deltas_no_drift_trend$d_splong, lag.max= 24)+ labs(title = 'PACF - d_splong')

ggsave('IMAGES/pacf_gdp.png', plot= pacf_gdp, width = 12, height=8)
ggsave('IMAGES/pacf_dpi.png', plot= pacf_dpi, width = 12, height=8)
ggsave('IMAGES/pacf_rate.png', plot= pacf_rate, width = 12, height=8)
ggsave('IMAGES/pacf_sp.png', plot= pacf_sp, width = 12, height=8)


# minimize information criteria
#gdp
arma_bic_gdp = critMatrix(deltas_no_drift_trend$d_gdp, p.max = 13, q.max = 2, criterion='bic')
stargazer(arma_bic_gdp, type='text', flip=F)
# stargazer(arma_bic_gdp, type='latex', flip=F,
#           out= 'TABLES/BIC_arma_gdp.tex', label="tab:bic_gdp",
#           title= "Information criteria on the parameters of ARMA for d-gdp")


#dpi
arma_bic_dpi = critMatrix(deltas_no_drift_trend$d_dpi, p.max = 12, q.max = 12, criterion='bic'
    )
stargazer(arma_bic_dpi, type='text', flip=F)
# stargazer(arma_bic_dpi, type='latex', flip=F,
#           out= 'TABLES/BIC_arma_dpi.tex', label="tab:bic_dpi",
#           title= "Information criteria on the parameters of ARMA for d-dpi")


#rate
arma_bic_rate = critMatrix(deltas_no_drift_trend$d_rate, p.max = 6, q.max = 10, criterion='bic
    ')
stargazer(arma_bic_rate, type='text', flip=T)
# stargazer(arma_bic_rate, type='latex', flip=T,
#           out= 'TABLES/BIC_arma_rate.tex', label="tab:bic_rate",
#           title= "Information criteria on the parameters of ARMA for d-rate")


```

```r
#dpi
arma_bic_sp = critMatrix(deltas_no_drift_trend$d_splong, p.max =6, q.max = 1, criterion='bic')
stargazer(arma_bic_sp, type='text', flip=F)
# stargazer(arma_bic_sp, type='latex', flip=F,
#           out= 'TABLES/BIC_arma_sp.tex', label="tab:bic_sp",
#           title= "Information criteria on the parameters of ARMA for d-splong")


## fit ARMA model
arma_gdp = arima(deltas_no_drift_trend$d_gdp, order= c(0,0,2))
stargazer(arma_gdp, type='text')

arma_dpi = arima(deltas_no_drift_trend$d_dpi, order= c(2,0,1))
stargazer(arma_dpi, type='text')

arma_rate = arima(deltas_no_drift_trend$d_rate, order= c(1,0,1))
stargazer(arma_rate, type='text')

arma_sp = arima(deltas_no_drift_trend$d_splong, order= c(0,0,1))
stargazer(arma_sp, type='text')


#Export all to latex
arma_all = list(arma_infl, arma_defl,  arma_gdp, arma_dpi, arma_rate, arma_sp)
names(arma_all) = c('infl_e', 'deflator', 'd_gdp', 'd_dpi', 'd_rate', 'd_splong')

stargazer(arma_all, type='text')
# stargazer(arma_all, type='latex', out='TABLES/all_arma.tex',
#           title = 'ARMA model for the cyclical components',
#           label = 'tab:all_arma')


## Residuals serially correlated?
model_names = c()
p_values = c()
# Loop through each ARMA model, perform Ljung-Box test, and store results
for (i in seq_along(arma_all)) {
  residuals = residuals(arma_all[[i]])
  df = sum(arma_all[[i]]$arma)-12
  ljung_box_test = Box.test(residuals, lag = 10, type = "Ljung-Box")
  model_names = c(model_names, names(arma_all)[[i]])
  p_values = c(p_values, ljung_box_test$p.value)
}
results_df = data.frame(Model = model_names, P_Value = p_values)

latex_table = xtable::xtable(results_df)
print(latex_table, file="TABLES/ljuung_box.tex")

```

```
###############
#COINTEGRATION
###############

#Get df with all the I(1) series in levels
levels_var = data.frame(
  gdp = allts$gdp,
  dpi = allts$dpi,
  rate = allts$rate,
  splong = allts$splong)

start_date = as.Date("1990-01-01")
end_date = as.Date("2022-12-01") # Assuming December 2022
all_dates = seq(start_date, end_date, by = "month")
levels_var$date = all_dates
rownames(levels_var) = levels_var$date #date as index
levels_var$date = NULL

lag_order = VARselect(levels_var)
res = lag_order$criteria


johansen_test = ca.jo(levels_var, type='trace', ecdet='trend', K=10)
jo_sum = summary(johansen_test)
# r is rank of matrix == number of cointegration relationship.
#no cointegration
johansen_table = xtable::xtable(summary(johansen_test))
print(johansen_table, file="TABLES/cointegration.tex")
```