Seminar Paper for the Course Structural Vector Autoregressive Analysis

Taught by Helmut Lütkepohl (DIW & FU-Berlin)

from 13^{th} May until 24^{th} May 2019

written by Patrick Brock (Humboldt-University)

Submitted on 2^{nd} June 2019

Preface

I replicated the main finding of the article

Technology, Employment, and the Business Cycle: Do Technology Shocks Explain Aggregate Fluctuations?

by Jordi Galí (1999) published in the AER, Vol. 89, No. 1.

On top of that, I provided some extension of Galí's work and conducted some analysis of my own inspired by Galí's work.

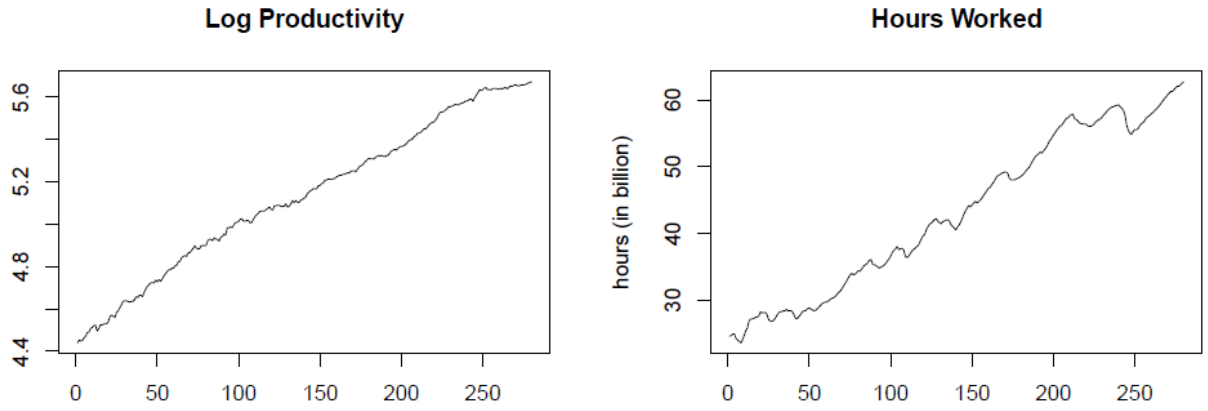
I almost exclusively used the programme *R* to conduct my analysis. All *R*-Studio scripts have been provided. The Appendix of this seminar paper provides the *R*-code, so that the reader does not necessarily have to go through all the *R*-Scripts, but can find the Code and the respective output neatly organised in the Appendix for each respective section. Some results have been computed using STATA and the Do-Files can be found in the Download Folder.

1 Data

Galí's VAR includes two variables: (Log) Productivity and hours worked, whereas productivity is defined as $log(GDP) - log(Hours\ worked)$. Hence, productivity is labour productivity and measures how much GDP could be generated by working for 1 hour. Galí drew his data from Citibase and used quarterly US data from 1948:1-1994:4. Unfortunately, this database is no longer available, so that my data differs from Galí's. I drew my data from FRED and used seasonally adjusted real GDP. Galí does not mention whether he used real GDP or whether it was seasonally adjusted, but it makes sense to remove seasonal effects and the increase of GDP due to higher prices. In addition, FRED and many other data sources only published yearly employment hours, but data on quarterly employment (i.e. the number of people in employment) was available. Using this data, I generated quarterly hours worked in the following manner: I combined the annual hours worked with the share of employment of a specific quarter for its respective year. For instance, if 1000h were worked in a year and 20% of all total employees of that year worked in the first quarter, then 200h were worked in the first quarter. This assumes that each employee within one year always worked the same amount of hours, i.e. a full time employee in 1980 worked 8h per day, no matter in which quarter he worked. Moreover, I used seasonally adjusted data for quarterly employees, so that my quarterly hours worked are also seasonally adjusted.

It is no surprise that productivity and hours worked both follow a linear trend as presented in Figure 1. It clearly shows that both variables are not stationary.

Figure 1: Evolution of Variables over Time



In order to make the variables stationary, Galí applies first differences to the log of productivity and the log of hours, i.e. $\Delta x_t = log(productivity_t) - log(productivity_{t-1})$ and $\Delta n_t = log(hours_t) - log(hours_{t-1})$, so that the variables can be interpreted as growth rates. Galí applies an augmented Dickey-Fuller Test with trend and drift to test for a unit root. Using the Schwert criterion, a number of $p = floor[12\{(T+1)/100\}^{0.25}] = 4$ lags for $T = 187$ observations is chosen. I ran the test myself and the Null is rejected for all lags from zero to 4 for all three specifications (none, drift, drift & trend) of the augmented Dickey-Fuller Test with all p-values smaller than 1%. Galí used a specification with drift & trend. For Δx_t, Galí obtained a test-value of -6.79 and I obtained a test-value of -7.02. For Δn_t, Galí obtained a test-value of -7.36 and I obtained a test-value of -7.79. Given that I drew my data from a different data set, a slight deviation in the test values is not surprising. In conclusion, the time series are stationary after the transformation, as the augmented DF-test rejects the presence of a unit root, so that the vector $y_t = [\Delta x_t, \Delta n_t] \sim I(0)$.

2 Estimating a reduced form VAR

In order to determine the lag order, I choose $p_{min} = 0$ and $p_{max} = 8$, as the data is quarterly data and use the LS-Estimator to estimate a VAR(p) with a constant. When computing the information criteria, I specifically do not rely on the individual information criteria being obtained from estimating models separately with different lags, but ensure that the variance of the error $\tilde{\Sigma}_u(m)$ for $m \in \{0, ..., p_{max}\}$ is always based on the same evaluation period $t = p_{max} + 1, ..., T$ for all m, so that the difference in the fit of the different models is not due to adding additional observations but rather due to additional lags[1]. When using the AIC and the HQC, the lag order is chosen to be 5, whereas when using the BIC the lag order is chosen to be 1. This is not surprising, since the AIC prefers larger models than the BIC. In order to determine which of the two lag orders to choose, I run a few model diagnostics. For the rest of this section, I will henceforth refer to the model $p^{AIC} = 5$ as the AIC-model and the model $p^{BIC} = 1$ as the Bayesian model, even though this model was explicitly not obtained by Bayesian estimates.

2.1 ARCH

I use the following standard model to test for ARCH-Effects

$$vech(\hat{u}_t\hat{u}_t') = \delta_0 + D_1 vech(\hat{u}_{t-1}\hat{u}_{t-1}') + \ldots + D_q vech(\hat{u}_{t-q}\hat{u}_{t-q}') + e_t$$

For the AIC-model, no ARCH-Effects are present when using $q = 1$ lag. However, when using more than 1 lag, the p-values are very small and even far below 1% which indicates the presence of ARCH-Effects in the model. Looking at the residual plot, it is not exactly clear why the test indicates ARCH-Effects, as the residuals are more or less evenly distributed across the zero-line. The residuals for the Bayesian model also have ARCH-Effects, but these are only present when including 4 or more lags.

Figure 2: Residuals for Δx_t for the AIC-model

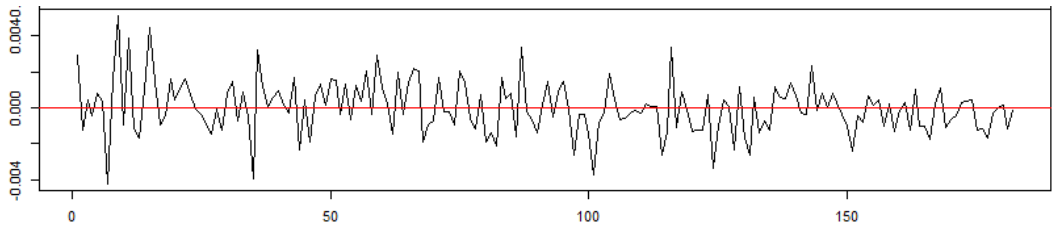
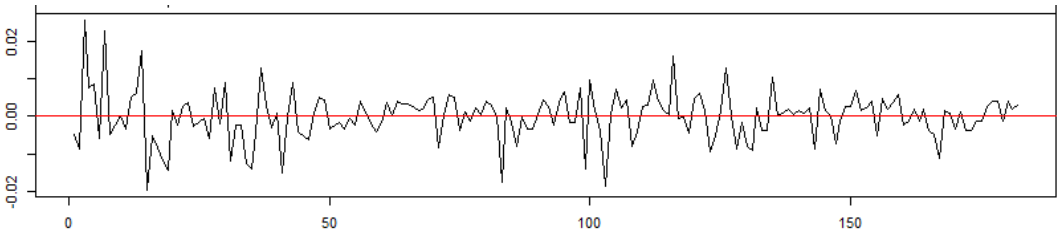


Figure 3: Residuals for Δn_t for the AIC-model



The residual plot for the Bayesian model is not reported, but can be replicated from the code in the Appendix. In Section 3, under a different specification, the Null for the AIC-model can no longer be rejected.

[1] Unfortunately, the *R*-packages for VAR by Bernhard Pfaff or Jae Kim do not support Maximum-Likelihood estimation for Σ_u which the procedure should be based on. This is why I ran diagnostics for all lags from 0 to 8 and found that 5 lags is the best, due to no serial correlation, which is also what my procedure based on LS finds.

2.2 Normality

The Null of the Jarque-Bera test for normality is rejected for both models. Even though the Skewness of the residuals is zero, the Kurtosis does not match the Kurtosis of a normal distribution.

2.3 Serial Correlation

The AIC-model does not exhibit any serial correlation. This is indicated by the Portmanteau test based on 16 lags, where the Null hypothesis for the AIC-model cannot be rejected, but the Null for the Bayesian model is strongly rejected. Due to the lack of serial correlation I choose the AIC-model over the Bayesian model.

2.4 Robustness Check

Since the data is quarterly data, frequently p^{max} is also set to 4. In such a scenario, the model based on the AIC has 4 lags, whereas the model based on the BIC has 2 lags. However, the diagnostics for both models only deteriorate and the Portmanteau-Test for the AIC-model rejects the Null of no serial correlation. As mentioned, I tested VAR's for all lags from zero to 8 and 5 lags is the only model that does not exhibit any serial correlation.

3 Forecasting

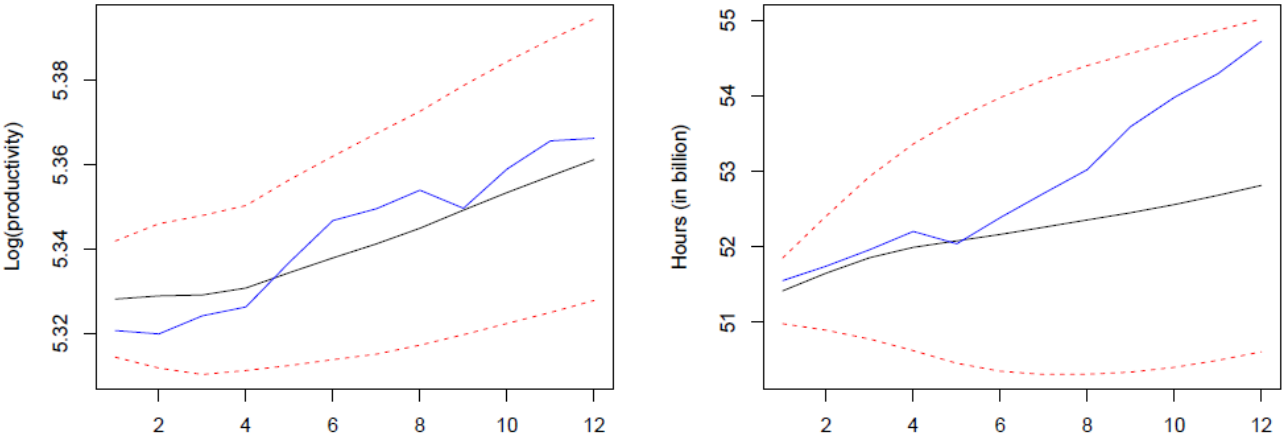
Before replicating Galí's main result, I turn my attention to forecasting which is not in the paper. For forecasting, I choose to estimate the VAR in levels with a linear trend to make it comparable to the forecast of a VECM (compare Figure 7). I also ran the diagnostics once again for the variables in levels with the trend component and the AIC still chooses 5 lags. Moreover, when estimating the VAR in levels with a trend, there is neither conditional heteroskedasticity nor serial autocorrelation, but the residuals are still non-normal. The estimates of the VAR(5), rounded to two decimal points, are given by

$$\begin{aligned}\begin{pmatrix} \widehat{log(prod_t)} \\ \hat{N}_t \end{pmatrix} = &\begin{pmatrix} 0.24 \\ 4.64 \end{pmatrix} + \begin{pmatrix} 0 \\ 0.01 \end{pmatrix} t + \begin{pmatrix} 0.74 & 0 \\ 10.24 & 1.4 \end{pmatrix} y_{t-1} + \begin{pmatrix} -0.01 & -0.01 \\ -2.92 & -0.28 \end{pmatrix} y_{t-2} \\ &+ \begin{pmatrix} -0.05 & 0 \\ -3.49 & -0.12 \end{pmatrix} y_{t-3} + \begin{pmatrix} 0.29 & -0.01 \\ -6.15 & 0.07 \end{pmatrix} y_{t-4} + \begin{pmatrix} -0.01 & 0.01 \\ 1.52 & -0.12 \end{pmatrix} y_{t-5}\end{aligned}$$

As the AIC is the information criteria that minimises the forecast error, the VAR(5) also seems to be the appropriate model for forecasting. Moreover, since Galí's time series ends in 1994:4, I can actually compute the true forecast errors by using the true values from 1995 and onwards to evaluate the forecasts. Figure 4 displays the result. The red line is the 95% confidence interval, the black line is the forecast and the blue line are the actual values. I forecasted 12 quarters.

As can be seen in the Figure, the forecast for productivity is close to the actual values throughout the forecast horizon. However, the forecast for hours worked diverges after 5 quarters. I also ran the forecast for a longer horizon and the true values even leave the confidence interval for the 13th quarter (not reported in paper).

Figure 4: Forecasts of VAR in levels vs. True Values



4 Structural Estimation

In this section I will replicate Galí's main result, so that I go back to using Galí's stationary variables Δx_t and Δn_t. Unfortunately, Galí does not report how many lags he used, but based on my analysis I choose a VAR(5). The reduced form can be written in the structural form by pre-multiplying it with the matrix B_0, such that $w_t = B_0 u_t$ and w_t are the structural shocks. Hence, the model is given by

$$B_0 y_t = B_0\nu + B_0 A_1 y_{t-1} + ... + B_0 A_5 y_{t-5} + B_0 u_t \Leftrightarrow B_0 A(L) y_t = B_0\nu + w_t$$
$$y_t = A(L)^{-1}\nu + [B_0 A(L)]^{-1} w_t = A(L)^{-1}\nu + \Theta(L) w_t \quad \text{with } \Theta(L) = A(L)^{-1} B_0^{-1}$$

Since $K = 2$, $\frac{K(K-1)}{2} = 1$ restriction is needed. Galí imposes the following long-run identifying restriction

$$\Theta(1) \begin{pmatrix} w_t^{technology} \\ w_t^{other} \end{pmatrix} = \begin{pmatrix} \theta_{11}(1) & 0 \\ \theta_{21}(1) & \theta_{22}(1) \end{pmatrix} \begin{pmatrix} w_t^{technology} \\ w_t^{other} \end{pmatrix}$$

so that in the long-run, only technology shocks have an effect on productivity. The other shocks are all non-technology shocks. Due to the Beveridge-Nelson decomposition, $\Theta(1)$ gives the long-run effects. The SVAR can be estimated by the following

$$\mathbb{E}[u_t u_t'] = \mathbb{E}[B_0^{-1} w_t w_t' [B_0^{-1}]'] \Leftrightarrow \Sigma_u = B_0^{-1} I_K B_0^{-1\prime} \Leftrightarrow \Sigma_u = A(1)\Theta(1)\Theta(1)' A(1)'$$
$$A(1)^{-1}\Sigma_u \left[A(1)^{-1}\right]' = \Theta(1)\Theta(1)'$$

where the left side only depends upon the reduced form parameters and $\Sigma_w = I_K$ was imposed by Galí. Since $\Theta(1)$ is a triangular matrix, a Cholesky-Decomposition of the left hand side is the solution to the equation. Since $B_0^{-1} = A(1)\Theta(1)$, this completes the estimation of the SVAR. The estimates are given by

$$\Theta(1) = \begin{pmatrix} 0.001098171 & 0 \\ -0.006387150 & 0.01091419 \end{pmatrix}$$
$$B_0^{-1} = \begin{pmatrix} 0.0009738645 & 0.001148000 \\ -0.0056362559 & 0.003845261 \end{pmatrix}$$

4.1 Impulse-Response functions

I have computed the cumulative and the non-cumulative Impulse-Response functions for a technology shock. When doing so, I specifically used Galí's assumption that the shocks are orthogonal to each other. The initial shock is a unit shock. As opposed to Galí, who used the asymptotic distribution, I constructed 86% Efron confidence intervals using 1000 bootstrap samples. I set a specific seed, so that the cumulative and non-cumulative confidence intervals were based on the same bootstrap sample. I believe that Galí himself reported the cumulative IRFs, as he put the variable in levels in his plot (p.261 Figure 2), even though the SVAR-model uses the variables in differences. Since summing up percentages differences roughly gives by how the level of the variable reacts, this does not only seem theoretically plausible[2], but also empirically the cumulative IRFs match Galí's much better than the non-cumulative ones. Figure 5 depicts the cumulative IRFs and for completeness Figure 6 depicts the non-cumulative IRFs. The variable on the left is Δx_t and the variable on the right is Δn_t. Galí also reported the change of GDP, which is however simply the sum of both IRFs, as $log(\text{productivity}_t) = log(GDP_t) - log(N_t) \Leftrightarrow log(GDP_t) = log(\text{productivity}_t) + log(N_t)$. I also believe that Galí multiplied the axis by 100. Since the variables are in logs, the change times 100 is the percentage change and as my values are roughly smaller by the factor 100, this seems plausible to me.

Figure 5: Cumulative Impulse-Response Functions for a Technology Shock

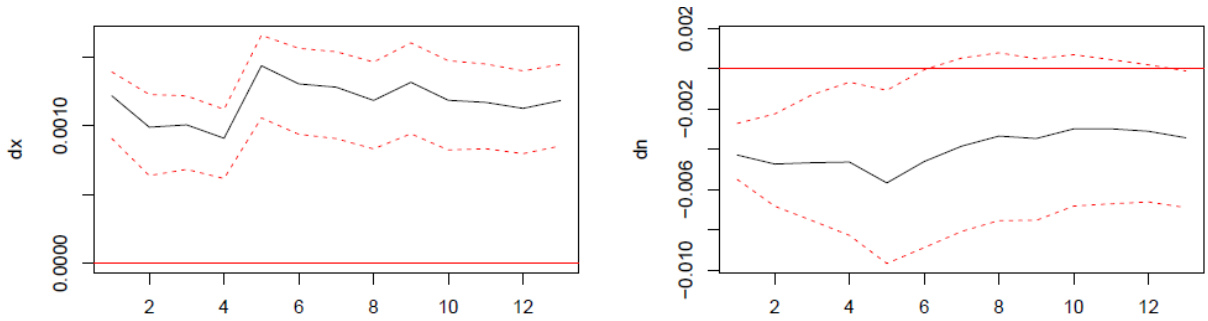
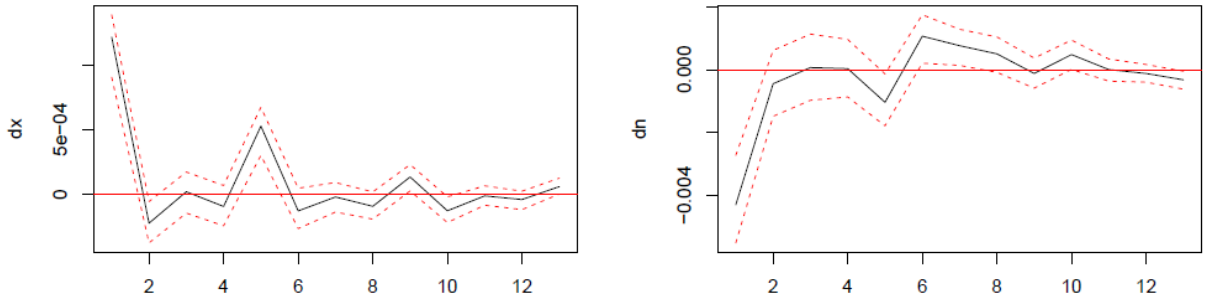


Figure 6: Impulse-Response Functions for a Technology Shock



On impact, when a technology shock hits, the hours worked decrease and in the first period this decline is strongly significant. In the second period, the next decrease is negative, but slightly insignificant. The effect of the technology shock is never significantly positive for hours worked. In conclusion, even though I was not able to directly replicate Galí's main finding (Figure 2 p.261), which is also due to the fact that I have to use different data than Galí, I was able to replicate his main finding and can show that the estimated response to a technology shock for labour is at odds with RBC models, since they predict a positive co-movement of technology shocks and labour, but the SVAR estimates a negative co-movement.

[2]For instance, if the wage rises by 1% and drops by 1%, then the wage is practically the same, as $1.01.0.99 \approx 1$, so that summing up the changes of $+1\%$ and -1% results in a zero change of the level.

5 SVECM

In order to expand Galí's work, I now include the interest rate in the model. Moreover, I will use US data from 1948:1-2017:4 and use the level of Productivity and Hours. Hence, I do not apply differences and productivity is the ratio of GDP and Hours. In order to see whether the interest rate is a relevant variable to include, I want to apply a Granger causality test. However, Productivity, Hours and the interest rate are not only all integrated of order 1 according to a DF-test[3] , but they are also likely to be cointegrated. Hence, before applying a Granger-causality test, I test for cointegration. In order to find the appropriate number of lags for the Johansen-cointegration test, I use a regular VAR model for $y_t = [Prod_t, N_t, i_t]$ and include a constant and a trend. I set the minimum lags to zero and the maximum lags to 8 and using the Akaike information criteria, the optimal number of lags is 6. Due to the trend of Productivity and Hours, I include a trend term in the Johansen cointegration test. The results are as follows

	Trace-test				Eigenvalue-test			
	test-value	$p = 10\%$	$p = 5\%$	$p = 1\%$	test-value	$p = 10\%$	$p = 5\%$	$p = 1\%$
$r = 0$	44.91	39.06	42.44	48.45	24.27	23.11	25.54	30.34
$r = 1$	20.64*	22.76	25.32	30.45	15.80*	16.85	18.96	23.65
$r = 2$	4.85	10.49	12.25	16.26	4.85	10.49	12.25	16.26

Hence, for a significance level of 10%, both tests choose a cointegration rank of $r = 1$. In order to test whether 6 lags are appropriate, I apply the adjusted Portmanteau-test due to the cointegration and cannot reject the Null for a significance level of 1%. Hence, 6 lags seem to be appropriate and avoid serial correlation. I also ran the same above analysis using the BIC, but the resulting model with 3 lags exhibited serial correlation.

To conduct a Granger-causality test for variables integrated of order 1, I use a VAR(7) instead of the VAR(6). Both the Null of the interest rate *not* granger causing Productivity & Hours and of *no* instantaneous causality are rejected even at a significance level of 1%. Hence, including the interest rate is a useful model expansion. Since the vector $y_t \sim I(1)$, I will estimate a VECM. Thus, the VECM, including an intercept and a trend, is given by

$$\Delta y_t - \mu_1 = \underbrace{\alpha}_{K\times r}\underbrace{\beta'}_{r\times K}\big(y_{t-1} - \mu_0 - \mu_1(t-1)\big) + \Gamma_1(\Delta y_{t-1} - \mu_1) + \ldots + \Gamma_5(\Delta y_{t-5} - \mu_1) + u_t$$

I estimated the VECM using Johansen's Maximum-Likelihood procedure and also used Johansen's normalisation restriction for the cointegration equation. The estimated cointegration equation is given by

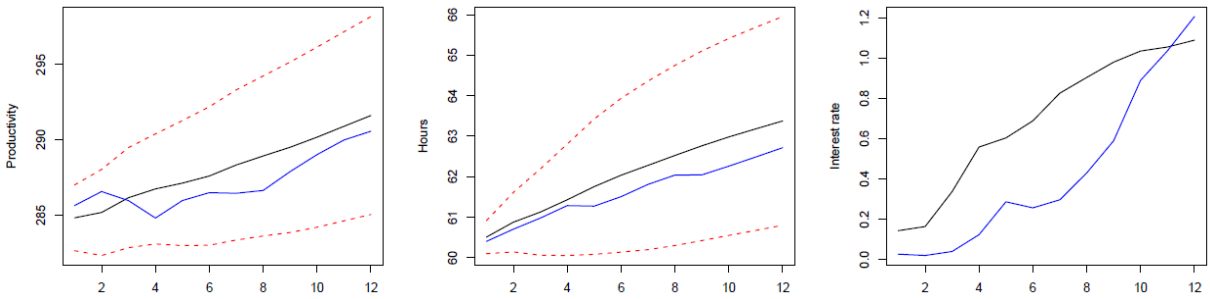
$$Productivity + 6.52 \cdot Hours + 3.07 \cdot irate - 1.61 \cdot t - 185.16 \sim I(0)$$

An augmented Dickey-Fuller test could not reject the Null hypothesis with drift or with drift and trend for a significance level of 5% for lags from 0 to 5 for this cointegration equation. For robustness, I also estimated the VECM without a trend and $r = 1$, but the resulting cointegration equation failed to reject the Null under the Dickey-Fuller test. I also estimated the VECM with trend & drift and drift only for $r = 2$, but no cointegration equation could reject the Null of the Dickey-Fuller test. These results can be generated from the code in the Appendix.

[3]For the levels, the DF-test could not reject the Null at all. For the differences, the DF-test rejected the Null for p-values $< 1\%$ for lags from 0 to 5 for models without drift, with drift and with drift & trend.

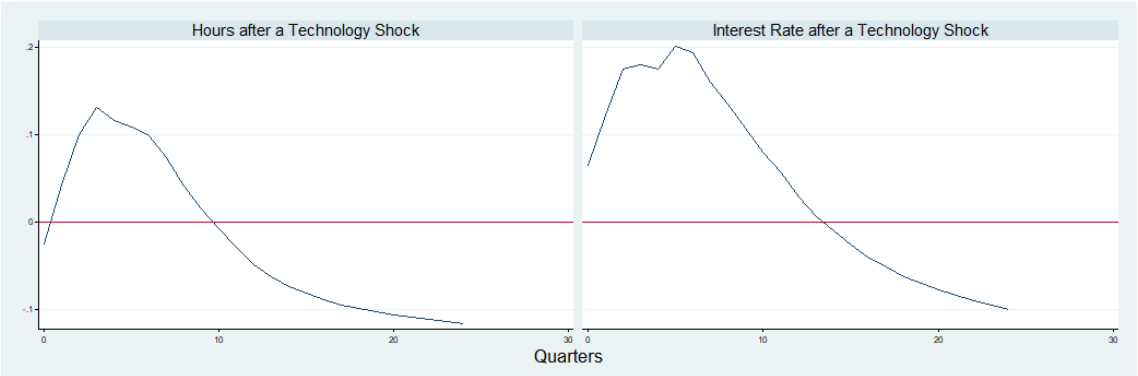
In order to put the VECM to use, I compute forecasts and (non-cumulative) impulse response functions based on it. As the last value in my dataset is 2017:4, I use all the data until 2014:4 and evaluate the forecast based on the last 12 quarters of my sample. The results are given in Figure 7. Unsurprisingly, the confidence bands (red) of the forecasts (black) for the VECM diverge as the forecast horizon grows, as the variance of the forecast errors for the levels of a cointegrated process diverges. It would make more sense to choose a smaller confidence band (i.e. 85%), but since I chose 95% confidence intervals in Figure 4, I use the same width for comparability. It can be seen when comparing the VECM-forecast with the VAR-forecast that the VECM-forecast is much more precise and captures the evolution of all variables, including labour, quite well. Naturally, the VECM-forecasts are more precise as they are based on a larger sample, i.e. 1948:1-2014:4, but I also estimated the VECM-forecast with the same sample used in Figure 4 for the VAR-forecast, i.e. 1948:1-1994:4, and the VECM-forecasts still performed better (Figure not reported in seminar paper). However, as the VECM includes another important variable, namely the interest rate, it is also somewhat easier for the VECM to produce a better forecast.

Figure 7: Forecasts & Confidence Intervals of the VECM vs. True Values



Note that the 95% confidence intervals are based on the asymptotic distribution. Hence, they are $\pm 1.96 \cdot S.E.$ of the respective variable. The confidence interval of the interest rate was so wide (range from -3 to +6) that it wasn't given, in order to not distort the Y-Axis. Note that productivity due to the data cannot really be interpreted. In my data set, the sum of quarterly hours equals to the yearly hours worked, whereas the sum of quarterly gdp is not the annual gdp. Hence, a value of 300 does not mean that a value of 300$ was generated in 1h of work. Next, to complete the augmentation of Galí's setting, I estimate Impulse-Response functions based on the VECM that are given in Figure 8. As can be seen for a technology shock, the effect of labour on impact is negative, but turns positive after one period and is negative again after some time. The interest rate evolves analogously. Hence, based on the VECM-IRFs, the effect of the technology shock is not as negative and not as at odds with the RBC-model as in Galí's setting. However, I only used a reduced form VECM and with a structural VECM the result might differ.

Figure 8: Impulse-Response Functions for a Technology Shock

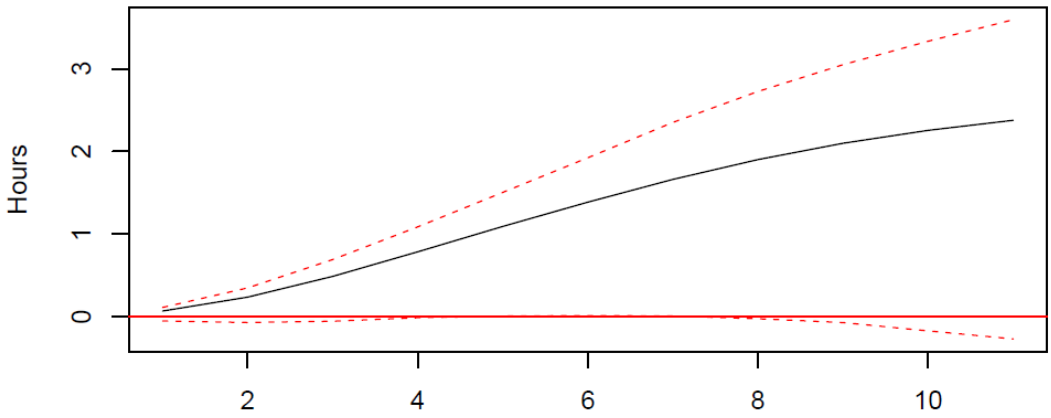


Consequently, I also computed the Impulse-Response function for a structural VECM and imposed the following long run triangular restriction

$$\Theta(1)\begin{pmatrix} w_t^{technology} \\ w_t^{interest\ rate} \\ w_t^{other} \end{pmatrix} = \begin{pmatrix} \cdot & 0 & 0 \\ \cdot & \cdot & 0 \\ \cdot & \cdot & \cdot \end{pmatrix}\begin{pmatrix} w_t^{technology} \\ w_t^{interest\ rate} \\ w_t^{other} \end{pmatrix}$$

so that only technology shocks have a long-run effect on productivity. Moreover, non-technology shocks do not have a long-run effect on the interest rate either. The estimated cumulative Impulse-Response function with 86% bootstrapped confidence intervals (200 samples) are given in Figure 9, whereas I consider orthogonal shocks.

Figure 9: Cumulative Impulse-Response of Labour for a Technology Shock for the SVECM



The cumulative effect for a technology shock is positive in the SVECM framework, even though it is slightly insignificant. Naturally, this SVECM cannot be directly compared to the SVAR by Galí, as I included a third variable and the identifying restriction I imposed must not necessarily be true. For instance, the long-run effect on productivity might not be zero for interest rate shocks. Nonetheless, this would act as a remedy for RBC-models.

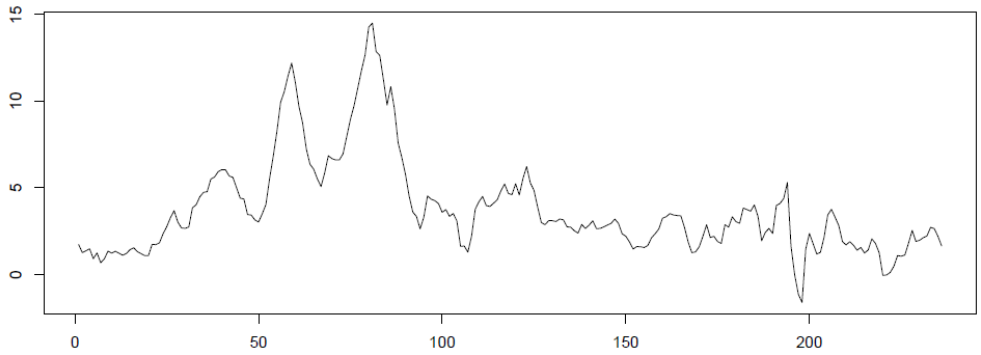
6 My Own Analysis

Galí used a simple bivariate SVAR to test the implications of RBC-models and their underlying economic theory. Inspired by this, I want to test the validity of monetarism based on a simple bivariate SVAR. Even though monetarism, just as RBC-models, is not as prominent anymore as it used to be, I deem this to be a good analysis of my own for this seminar paper. Hence, for the first VAR, I use US data for the monetary base $M1$ and the inflation rate based on the CPI. The monetary base is seasonally adjusted and I use data from 1960:1-2019:1 (no prior data available). Since the monetary base is not stationary, I take the logs and apply differences. The p-values of the augmented Dickey-Fuller test for both variables are given in the table below

	Δm_1			Inflation Rate		
lags	none	drift	drift & trend	none	drift	drift & trend
0	< 0.01	< 0.01	< 0.01	0.2596	0.3486	0.4505
1	< 0.01	< 0.01	< 0.01	0.0854	0.0458	0.0672
2	< 0.01	< 0.01	< 0.01	0.0831	0.0434	0.0617
3	< 0.01	< 0.01	< 0.01	0.0593	0.0196	0.0250
4	0.0341	0.01	0.0181	0.2320	0.2312	0.2635

Thus, the monetary base is clearly stationary. The results for inflation are not as clear cut. It makes the most sense to only take the specification with drift, as inflation does not exhibit a trend. For 1 to 3 lags the variable is stationary, but with 4 lags for conventional significance levels a unit root cannot be rejected. Looking at the evolution of inflation, given in Figure 10, I deem the variable as stationary and will continue my analysis.

Figure 10: Evolution of Inflation



As I have two variables, I require one identifying restriction. Contrary to Galí's analysis, I will impose a short run restriction, such that

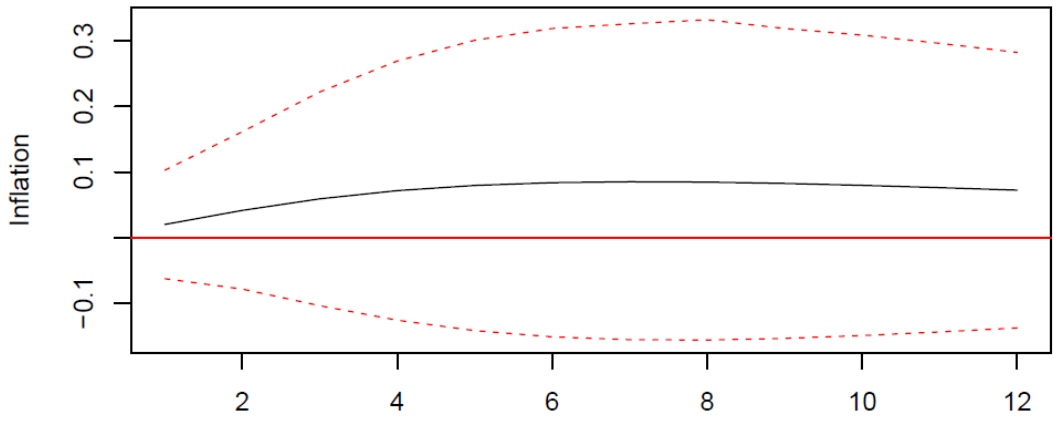
$$u_t = B_0^{-1} w_t \qquad \Leftrightarrow \qquad \begin{pmatrix} u_t^{inflation} \\ u_t^{money} \end{pmatrix} = \begin{pmatrix} b_0^{11} & 0 \\ b_0^{21} & b_0^{22} \end{pmatrix} \begin{pmatrix} w_t^{cost-push} \\ w_t^{money-supply} \end{pmatrix}$$

Hence, since prices are sticky, the inflation rate will not instantaneously react to changes in the money supply, so that prices are only adjusted at least one quarter after the money-supply shock. Cost-push shocks are all remaining shocks that influence the inflation rate. A very typical example of this would be an oil price shock that increases the price of almost all goods. The central bank as a policy maker can react to these shocks by changing the money-supply instantaneously according to my identification. It would also be possible to over-identify and set b_0^{21} to zero, so that the central bank cannot instantaneously react to inflationary shocks, since price data also takes some time to be measured and the central bank takes time for coordination and decision making. However, leaving out this restriction only marginally changes the IRF (graph not reported in paper).

The Impulse-Response function is given in Figure 11. Once again, the shocks are orthogonal to each other and I consider a unit shock. The 86% confidence interval is obtained from bootstrapped residuals with 200 bootstrap samples. After the initial shock, inflation only spikes at 0.08% and is not significance throughout. Note that inflation is already in percent, so that the Y-Axis does not need to be multiplied by 100. The results show that this is in line with New Keynesian theory. The New Keynesian inflation equation clearly shows that the inflation rate (if expectations are kept constant) are driven by steady-state deviations of marginal costs. The money $M1$ is however not what we usually think of as being a part of firms' marginal costs, so that especially large wage increases relative to productivity gains will drive the inflation rate up and not an expansion of the monetary base. Hence, this IRF does not confirm the theory of monetarism. Alternatively, I expand the model to include the interest rate into the model. A Granger-causality test rejects the Null of *no* granger-causality and *no* instantaneous causality at a significance level of 5%, so that the variable is relevant. I impose the following short-run restrictions.

$$\begin{pmatrix} u_t^{\pi} \\ u_t^{m} \\ u_t^{i} \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ b_0^{21} & 1 & b_0^{23} \\ 0 & b_0^{32} & 1 \end{pmatrix} \begin{pmatrix} w_t^{cost-push} \\ w_t^{money-supply} \\ w_t^{interestrate} \end{pmatrix}$$

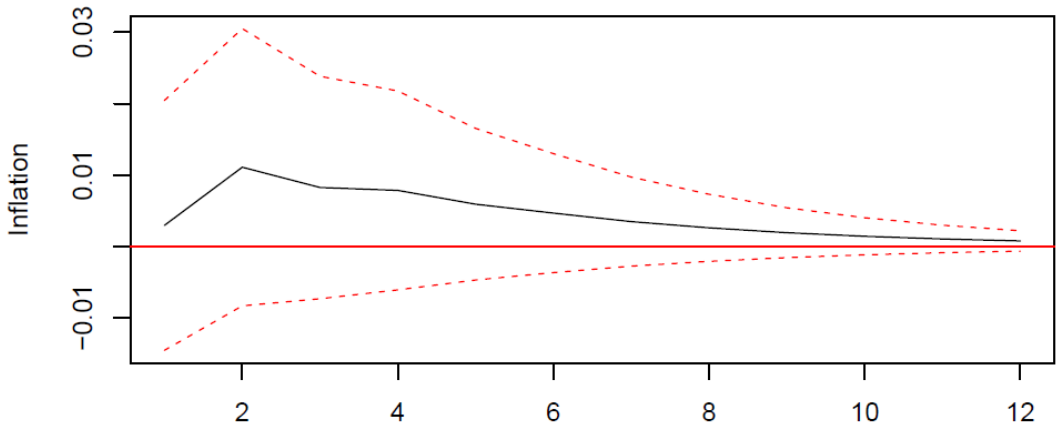
Figure 11: Impulse-Response of Inflation for a Money-Supply Shock



Once again, as prices are sticky, they do not instantly react to money-supply or interest rate shocks. On top of that, I impose that the central bank does not immediately adjust the interest rate to cost-push shocks due to inflation data not being available instantaneously and the central bank not acting immediately and observing the economy first before undertaking measures. I have three over-identifying restrictions in the model, which are unfortunately necessary for identification. Without the diagonal being equal to one, neither Maximum-Likelihood nor the scoring algorithm are able to compute any results due to singularity. It takes almost 1500 iterations for a solution of the above to be reached. The estimated impulse response functions are an almost exact copy of Figure 11. This is why I do not report this Impulse-Response, but it can be generated from the Do-File in the Download Folder.

Instead, I check for further robustness and see what happens when the data frequency is changed to monthly data. Since the model with the interest rate yielded the same results as without it, I omit the interest rate, so that the model is also easier to identify and less restrictions are required. I impose the same restrictions as on page 9. Hence, it might be too restrictive to impose rigid prices for one quarter, so that prices are now only rigid for one month on impact of the shock. Unfortunately, FRED only publishes the Index of the CPI on a monthly basis, but it is nonetheless possible to compute the percentage changes from this index to obtain the inflation rate. The results are given in Figure 12. Even though the IRFs have some more interesting dynamics, a money supply shock does not impact inflation significantly.

Figure 12: Impulse-Response of Inflation for a Money-Supply Shock (Monthly Data)



Appendix A Code

This Appendix provides the code for the respective section of the seminar paper. The Appendix is aimed to provide a better overview of the code, so that the reader does not need to look at the individual R-Scripts and can instead comfortably view it printed out on paper. As I don't expect the reader to be familiar with every single command, I did not suppress the output, so that it can be inferred from the output what the command does.

A.1 Data

```
#### Dickey-Fuller Test ####
library("aTSA")
data = read.csv2("C:/Users/Patrick/Data.csv")
dx = data[2:188,8] #Gali-Sample-Period
dn = data[2:188,9] #Gali-Sample-Period
adf.test(dx)
```

```
## Augmented Dickey-Fuller Test
## alternative: stationary
##
## Type 1: no drift no trend
##      lag    ADF p.value
## [1,]   0 -11.41    0.01
## [2,]   1  -6.90    0.01
## [3,]   2  -5.79    0.01
## [4,]   3  -3.89    0.01
## [5,]   4  -3.54    0.01
## Type 2: with drift no trend
##      lag    ADF p.value
## [1,]   0 -14.82    0.01
## [2,]   1  -9.97    0.01
## [3,]   2  -9.40    0.01
## [4,]   3  -6.60    0.01
## [5,]   4  -6.46    0.01
## Type 3: with drift and trend
##      lag    ADF p.value
## [1,]   0 -15.10    0.01
## [2,]   1 -10.33    0.01
## [3,]   2  -9.96    0.01
## [4,]   3  -7.05    0.01
## [5,]   4  -7.02    0.01 ## Value reported in Paper
## ----
## Note: in fact, p.value = 0.01 means p.value <= 0.01
```

```
adf.test(dn)
```

```
## Augmented Dickey-Fuller Test
## alternative: stationary
##
## Type 1: no drift no trend
##      lag   ADF p.value
```

```
## [1,]   0 -6.79    0.01
## [2,]   1 -5.50    0.01
## [3,]   2 -5.27    0.01
## [4,]   3 -4.89    0.01
## [5,]   4 -6.01    0.01
## Type 2: with drift no trend
##      lag   ADF p.value
## [1,]   0 -7.58    0.01
## [2,]   1 -6.24    0.01
## [3,]   2 -6.17    0.01
## [4,]   3 -6.08    0.01
## [5,]   4 -7.79    0.01
## Type 3: with drift and trend
##      lag   ADF p.value
## [1,]   0 -7.56    0.01
## [2,]   1 -6.22    0.01
## [3,]   2 -6.15    0.01
## [4,]   3 -6.07    0.01
## [5,]   4 -7.79    0.01 ## Value reported in Paper
## ----
## Note: in fact, p.value = 0.01 means p.value <= 0.01
```

A.2 Estimating a reduced form VAR

```
library("vars")
data = read.csv2("C:/Users/Patrick/Data.csv")
dx = data[2:188,8]
dn = data[2:188,9]

y = cbind(dx,dn)

            ### Choose lag order according to the AIC ####

var.aic = VAR(y,p=0,lag.max=8,type="const",ic="AIC") #5 lags
length(var.aic$varresult$dx$coefficients)
```

```
## [1] 11
```

```
#K*p+1 coefficients <--> 2*p+1 =11 implies 5 lags

            #### Choose lag order according to the BIC ###

var.bic =VAR(y,p=0,lag.max=8,type="const",ic="SC") #1 lag
length(var.bic$varresult$dx$coefficients)
```

```
## [1] 3
```

```
#K*p+1 coefficients <--> 2*p+1 =3 implies 1 lag

            #### Choose lag order according to HQ ####
```

```
var.hq = VAR(y,p=0,lag.max=8,type="const",ic="HQ") #5 lags
length(var.aic$varresult$dx$coefficients)
```

```
## [1] 11
```

```
#K*p+1 coefficients <--> 2*p+1 =11 implies 5 lags

            #### Test for ARCH in AIC ####

arch.aic = seq(1,4)
for(i in 1:4){
  arch.aic[i] = round(arch.test(var.aic,lags.multi=i)$arch.mul$p.value,4)
}
arch.aic #Displays p-value
```

```
## [1] 0.4215 0.0018 0.0009 0.0000
```

```
#Arch-Effects are present with 2 lags already


            #### Test for ARCH in BIC ####

arch.bic = seq(1,4)
for(i in 1:4){
  arch.bic[i] = round(arch.test(var.bic,lags.multi=i)$arch.mul$p.value,4)
}
arch.bic #Displays p-value
```

```
## [1] 0.8045 0.3359 0.4281 0.0000
```

```
#Arch-Effects only first present with 4 lags

            #### Normality Test ####

normality.test(var.aic)
```

```
## $JB
##
##  JB-Test (multivariate)
##
## data:  Residuals of VAR object var.aic
## Chi-squared = 49.556, df = 4, p-value = 4.469e-10
##
##
## $Skewness
##
##  Skewness only (multivariate)
##
## data:  Residuals of VAR object var.aic
## Chi-squared = 2.6277, df = 2, p-value = 0.2688
##
##
## $Kurtosis
```

```
##
##  Kurtosis only (multivariate)
##
## data:  Residuals of VAR object var.aic
## Chi-squared = 46.929, df = 2, p-value = 6.451e-11
```

```
normality.test(var.bic)
```

```
## $JB
##
##  JB-Test (multivariate)
##
## data:  Residuals of VAR object var.bic
## Chi-squared = 86.411, df = 4, p-value < 2.2e-16
##
##
## $Skewness
##
##  Skewness only (multivariate)
##
## data:  Residuals of VAR object var.bic
## Chi-squared = 1.4359, df = 2, p-value = 0.4878
##
##
## $Kurtosis
##
##  Kurtosis only (multivariate)
##
## data:  Residuals of VAR object var.bic
## Chi-squared = 84.976, df = 2, p-value < 2.2e-16
```

```
#Both models are non-normal, the Kurtosis is not equal to 3


            #### Serial Correlation ####
serial.test(var.aic)
```

```
##
##  Portmanteau Test (asymptotic)
##
## data:  Residuals of VAR object var.aic
## Chi-squared = 53.974, df = 44, p-value = 0.1441
```

```
serial.test(var.bic)
```

```
##
##  Portmanteau Test (asymptotic)
##
## data:  Residuals of VAR object var.bic
## Chi-squared = 133.94, df = 60, p-value = 1.437e-07
```

```
#Portmanteau test with 16 lags
#AIC not serially correlated, so take AIC
```

```r
serial.test(var.aic,type="PT.adjusted")
```

```
##
##  Portmanteau Test (adjusted)
##
## data:  Residuals of VAR object var.aic
## Chi-squared = 57.12, df = 44, p-value = 0.08863
```

```r
serial.test(var.bic, type="PT.adjusted")
```

```
##
##  Portmanteau Test (adjusted)
##
## data:  Residuals of VAR object var.bic
## Chi-squared = 139.85, df = 60, p-value = 2.544e-08
```

```r
#BIC still rejected, AIC not rejected


            #### Estimates ####
Bcoef(var.aic)
```

```
##         dx.l1     dn.l1      dx.l2       dn.l2      dx.l3       dn.l3
## dx -0.1348615 0.0147091 -0.1521087 -0.04145314 -0.1776738 -0.01562641
## dn  1.2015684 0.4449166  1.0098818  0.16076306  0.2725304  0.01101166
##         dx.l4       dn.l4       dx.l5       dn.l5        const
## dx  0.1064695 -0.07951504  0.07428187  0.03653065 1.564024e-03
## dn -0.3452543  0.11077123 -0.02083794 -0.22747553 6.981213e-05
```

A.3 Forecasting

```
library("vars")
data = read.csv2("C:/Users/Patrick/Data.csv")
#Forecast models based on levels. Include a trend term and a constant!

prod = data[1:188,4] #Gali-Sample to estimate VAR in levels
hours = data[1:188,3] #Gali-Sample to estimate VAR in levels

prod.new = data[189:200,4] #Actual values outside of sample
hours.new = data[189:200,3] #Actual values outside of sample

y = cbind(prod,hours)

            #### Estimating the VAR in levels with trends ####

var.aic = VAR(y,p=0,lag.max=8,type="both",ic="AIC") #5 lags + constant + trend

### Estimates ####
round(Bcoef(var.aic),2)
```

```
##       prod.l1 hours.l1 prod.l2 hours.l2 prod.l3 hours.l3 prod.l4 hours.l4
## prod     0.74      0.0   -0.01    -0.01   -0.05     0.00    0.29    -0.01
## hours   10.24      1.4   -2.92    -0.28   -3.49    -0.12   -6.15     0.07
##       prod.l5 hours.l5 const trend
## prod    -0.01     0.01  0.24  0.00
## hours    1.52    -0.12  4.64  0.01
```

```
#Diagnostics
serial.test(var.aic) #no serial correlation
```

```
##
##  Portmanteau Test (asymptotic)
##
## data:  Residuals of VAR object var.aic
## Chi-squared = 52.828, df = 44, p-value = 0.1699
```

```
arch.test(var.aic) #no ARCH-Effects
```

```
##
##  ARCH (multivariate)
##
## data:  Residuals of VAR object var.aic
## Chi-squared = 56.356, df = 45, p-value = 0.1194
```

```
normality.test(var.aic) #still non-normal errors
```

```
## $JB
##
##  JB-Test (multivariate)
##
## data:  Residuals of VAR object var.aic
## Chi-squared = 41.586, df = 4, p-value = 2.032e-08
##
```

```
##
## $Skewness
##
##  Skewness only (multivariate)
##
## data:  Residuals of VAR object var.aic
## Chi-squared = 3.5224, df = 2, p-value = 0.1718
##
##
## $Kurtosis
##
##  Kurtosis only (multivariate)
##
## data:  Residuals of VAR object var.aic
## Chi-squared = 38.064, df = 2, p-value = 5.427e-09
```

```
#### Compute forecasts

var.predict = predict(var.aic,n.ahead=12)#forecasting 3 years ahead

par(mfrow=c(1,2))

prod.predict = var.predict$fcst$prod
plot(prod.predict[,1], type="l",ylim=c(min(prod.predict[,2]),max(prod.predict[,3])),
ylab="Log(productivity)",xlab="")
lines(prod.predict[,2],type="l",col="red",lty="dashed")
lines(prod.predict[,3],type="l",col="red",lty="dashed")
lines(prod.new, type="l", col="blue")

hours.predict = var.predict$fcst$hours
plot(hours.predict[,1], type="l",ylim=c(min(hours.predict[,2]),max(hours.predict[,3])),
 ylab="Hours (in billion)",xlab="")
lines(hours.predict[,2],type="l",col="red",lty="dashed")
lines(hours.predict[,3],type="l",col="red",lty="dashed")
lines(hours.new, type="l", col="blue")
```

A.4 Structural Estimation / Impulse-Response Functions

```
library("vars")
data = read.csv2("C:/Users/Patrick/Data.csv")
dx = data[2:188,8]
dn = data[2:188,9]

y = cbind(dx,dn)

            ### Estimate VAR(5) ####

var.aic = VAR(y,p=5, type="cons") #5 lags

            #### Estimate Sigma_u ####
T = 187; K=2; p=5

Sigma.u = summary(var.aic)$covres
# or equivalently
dx.resid = as.matrix(var.aic$varresult$dx$residuals,ncol=1)
dn.resid = as.matrix(var.aic$varresult$dn$residuals,ncol=1) #$
U = rbind(t(dx.resid), t(dn.resid))
U %*%t(U)/((T-p)-K*p-1)
```

```
##                [,1]          [,2]
## [1,]  2.266317e-06 -1.074589e-06
## [2,] -1.074589e-06  4.655341e-05
```

```
            #### Obtain A(1) ####

#Step 1: Compute the Estimates
var.estimates = Bcoef(var.aic)
A1 = Bcoef(var.aic)[,1:2]
A2 = Bcoef(var.aic)[,3:4]
A3 = Bcoef(var.aic)[,5:6]
A4 = Bcoef(var.aic)[,7:8]
A5 = Bcoef(var.aic)[,9:10]
#Step 2: A(1) Polynomial
A.polyn = diag(2) - A1 - A2 - A3 - A4 - A5
A.polyn = unname(A.polyn,force=FALSE) # Remove names
A.polyn.inverse = solve(A.polyn)

#### Estimate Theta(1) ####
Theta.1 = t(chol(A.polyn.inverse%*%Sigma.u %*% t(A.polyn.inverse)))

B0.inverse = A.polyn%*%Theta.1

svar = BQ(var.aic)
#Uses the Blanchard-Quahn long-run triangular identification matrix, which
#is the same Gali used

            #### Compute cumulative IRFs ####
```

```
irf = irf(svar, ci=0.86, runs=1000,cumulative=T,ortho=TRUE, seed=528491,n.ahead=12)
plot(irf)

#### Make Graph for dx for Technology shock(dx-shock) ####
par(mfrow=c(1,2))
dx.irf = irf$irf$dx #IRF of dx-shock
dx.irf.ci.up = irf$Upper$dx #Upper C.I
dx.irf.ci.low = irf$Lower$dx #Lower C.I
plot(dx.irf[,1], type="l",ylim=c(0,max(dx.irf.ci.up[,1])),xlab="",ylab="dx")
lines(dx.irf.ci.up[,1],lty="dashed",col="red")
lines(dx.irf.ci.low[,1],lty="dashed",col="red")
abline(a=0,b=0,col="red")

#### Make Graph for dn for Technology shock ####
dn.irf = irf$irf$dx #IRF of dx-shock
dn.irf.ci.up = irf$Upper$dx #Upper C.I
dn.irf.ci.low = irf$Lower$dx #Lower C.I
plot(dn.irf[,2], type="l",xlab="",ylab="dn",
ylim=c(min(dn.irf.ci.low),max(dn.irf.ci.up)))
lines(dn.irf.ci.up[,2],lty="dashed",col="red")
lines(dn.irf.ci.low[,2],lty="dashed",col="red")
abline(a=0,b=0,col="red")
```

```
            #### Compute non-cumulative IRFs ####

irf.n = irf(svar, ci=0.86, runs=1000,cumulative=F,ortho=TRUE, seed=528491,n.ahead=12)
#### Make Graph for dx for Technology shock(dx-shock) ####
par(mfrow=c(1,2))
dx.irf.n = irf.n$irf$dx #IRF of dx-shock
dx.irf.n.ci.up = irf.n$Upper$dx #Upper C.I
dx.irf.n.ci.low = irf.n$Lower$dx #Lower C.I
plot(dx.irf[,1], type="l",
ylim=c(0,max(dx.irf.ci.up[,1])),xlab="",ylab="dx")
lines(dx.irf.ci.up[,1],lty="dashed",col="red")
lines(dx.irf.ci.low[,1],lty="dashed",col="red")
abline(a=0,b=0,col="red")

#### Make Graph for dn for Technology shock ####
dn.irf.n = irf.n$irf$dx #IRF of dx-shock
dn.irf.n.ci.up = irf.n$Upper$dx #Upper C.I
dn.irf.n.ci.low = irf.n$Lower$dx #Lower C.I
plot(dn.irf.n[,2], type="l",xlab="",ylab="dn",
ylim=c(min(dn.irf.n.ci.low),max(dn.irf.n.ci.up)))
lines(dn.irf.n.ci.up[,2],lty="dashed",col="red")
lines(dn.irf.n.ci.low[,2],lty="dashed",col="red")
abline(a=0,b=0,col="red")
```

A.5 SVECM

```
#Check that all  variables are utmost of order I(1). A VECM only works if
#Variables are not more than of order 1
library("vars")
library("aTSA")
data = read.csv2("C:/Users/Patrick/Data.csv")
gdp = data[1:280,2]
n = data[1:280,3]
irate = data[1:280,6]
prod = gdp/n

      #### Check if all Variables are at least I(1) if not I(0) ####

adf.test(prod) #Not stationary
```

```
## Augmented Dickey-Fuller Test
## alternative: stationary
##
## Type 1: no drift no trend
##      lag   ADF p.value
## [1,]   0 10.15    0.99
## [2,]   1  8.68    0.99
## [3,]   2  7.43    0.99
## [4,]   3  7.00    0.99
## [5,]   4  5.55    0.99
## [6,]   5  5.26    0.99
## Type 2: with drift no trend
##      lag   ADF p.value
## [1,]   0 1.134    0.99
## [2,]   1 1.205    0.99
## [3,]   2 1.096    0.99
## [4,]   3 1.130    0.99
## [5,]   4 0.983    0.99
## [6,]   5 0.994    0.99
## Type 3: with drift and trend
##      lag   ADF p.value
## [1,]   0 -1.49   0.792
## [2,]   1 -1.51   0.784
## [3,]   2 -1.45   0.806
## [4,]   3 -1.38   0.838
## [5,]   4 -1.53   0.774
## [6,]   5 -1.51   0.784
## ----
## Note: in fact, p.value = 0.01 means p.value <= 0.01
```

```
prod2 = rep(1,279)
for(i in 2:279){
  prod2[1] = NA
  prod2[i] = prod[i]-prod[i-1]
}
```

```
adf.test(prod2) #Stationary

## Augmented Dickey-Fuller Test
## alternative: stationary
##
## Type 1: no drift no trend
##      lag    ADF p.value
## [1,]   0 -12.68    0.01
## [2,]   1  -7.84    0.01
## [3,]   2  -6.12    0.01
## [4,]   3  -4.31    0.01
## [5,]   4  -3.76    0.01
## [6,]   5  -3.15    0.01
## Type 2: with drift no trend
##      lag    ADF p.value
## [1,]   0 -16.92    0.01
## [2,]   1 -11.60    0.01
## [3,]   2  -9.95    0.01
## [4,]   3  -7.44    0.01
## [5,]   4  -6.86    0.01
## [6,]   5  -5.92    0.01
## Type 3: with drift and trend
##      lag    ADF p.value
## [1,]   0 -17.01    0.01
## [2,]   1 -11.69    0.01
## [3,]   2 -10.05    0.01
## [4,]   3  -7.54    0.01
## [5,]   4  -6.97    0.01
## [6,]   5  -6.05    0.01
## ----
## Note: in fact, p.value = 0.01 means p.value <= 0.01

adf.test(n) #Not stationary

## Augmented Dickey-Fuller Test
## alternative: stationary
##
## Type 1: no drift no trend
##      lag  ADF p.value
## [1,]   0 7.31    0.99
## [2,]   1 3.30    0.99
## [3,]   2 2.87    0.99
## [4,]   3 2.96    0.99
## [5,]   4 3.01    0.99
## [6,]   5 3.76    0.99
## Type 2: with drift no trend
##      lag    ADF p.value
## [1,]   0  0.109   0.964
## [2,]   1 -0.261   0.923
## [3,]   2 -0.249   0.925
## [4,]   3 -0.314   0.915
```

```
## [5,]   4 -0.605   0.836
## [6,]   5 -0.466   0.885
## Type 3: with drift and trend
##      lag   ADF p.value
## [1,]   0 -1.59  0.7474
## [2,]   1 -2.95  0.1746
## [3,]   2 -3.34  0.0631
## [4,]   3 -3.20  0.0874
## [5,]   4 -3.12  0.1029
## [6,]   5 -2.45  0.3861
## ----
## Note: in fact, p.value = 0.01 means p.value <= 0.01

n2 = rep(1,279)
for(i in 2:279){
  n2[1] = NA
  n2[i] = n[i]-n[i-1]
}
adf.test(n2) #Stationary

## Augmented Dickey-Fuller Test
## alternative: stationary
##
## Type 1: no drift no trend
##      lag   ADF p.value
## [1,]   0 -7.52    0.01
## [2,]   1 -5.97    0.01
## [3,]   2 -5.61    0.01
## [4,]   3 -5.13    0.01
## [5,]   4 -5.96    0.01
## [6,]   5 -4.77    0.01
## Type 2: with drift no trend
##      lag   ADF p.value
## [1,]   0 -8.44    0.01
## [2,]   1 -6.79    0.01
## [3,]   2 -6.53    0.01
## [4,]   3 -6.20    0.01
## [5,]   4 -7.38    0.01
## [6,]   5 -6.07    0.01
## Type 3: with drift and trend
##      lag   ADF p.value
## [1,]   0 -8.42    0.01
## [2,]   1 -6.78    0.01
## [3,]   2 -6.52    0.01
## [4,]   3 -6.18    0.01
## [5,]   4 -7.36    0.01
## [6,]   5 -6.06    0.01
## ----
## Note: in fact, p.value = 0.01 means p.value <= 0.01

adf.test(irate)
```

```
## Augmented Dickey-Fuller Test
## alternative: stationary
##
## Type 1: no drift no trend
##      lag   ADF p.value
## [1,]   0 -1.08   0.293
## [2,]   1 -1.37   0.188
## [3,]   2 -1.07   0.295
## [4,]   3 -1.41   0.173
## [5,]   4 -1.27   0.224
## [6,]   5 -1.54   0.128
## Type 2: with drift no trend
##      lag   ADF p.value
## [1,]   0 -1.85  0.3882
## [2,]   1 -2.36  0.1855
## [3,]   2 -1.85  0.3851
## [4,]   3 -2.46  0.1464
## [5,]   4 -2.23  0.2388
## [6,]   5 -2.74  0.0727
## Type 3: with drift and trend
##      lag   ADF p.value
## [1,]   0 -1.95   0.597
## [2,]   1 -2.45   0.387
## [3,]   2 -1.96   0.591
## [4,]   3 -2.55   0.346
## [5,]   4 -2.32   0.441
## [6,]   5 -2.82   0.229
## ----
## Note: in fact, p.value = 0.01 means p.value <= 0.01
```

```
irate2 = rep(1,279)
for(i in 2:279){
  irate2[1] = NA
  irate2[i] = irate[i]-irate[i-1]
}
adf.test(irate2) #Stationary
```

```
## Augmented Dickey-Fuller Test
## alternative: stationary
##
## Type 1: no drift no trend
##      lag    ADF p.value
## [1,]   0 -13.05    0.01
## [2,]   1 -13.07    0.01
## [3,]   2  -7.77    0.01
## [4,]   3  -7.78    0.01
## [5,]   4  -5.78    0.01
## [6,]   5  -6.58    0.01
## Type 2: with drift no trend
##      lag    ADF p.value
## [1,]   0 -13.03    0.01
```

```
## [2,]   1 -13.05    0.01
## [3,]   2  -7.75    0.01
## [4,]   3  -7.77    0.01
## [5,]   4  -5.77    0.01
## [6,]   5  -6.57    0.01
## Type 3: with drift and trend
##      lag    ADF p.value
## [1,]   0 -13.04    0.01
## [2,]   1 -13.09    0.01
## [3,]   2  -7.78    0.01
## [4,]   3  -7.81    0.01
## [5,]   4  -5.80    0.01
## [6,]   5  -6.62    0.01
## ----
## Note: in fact, p.value = 0.01 means p.value <= 0.01
```

```
            #### Test for Cointegration ####
y = cbind(prod,n,irate)
var.aic=VAR(y,p=0,lag.max = 8, ic="AIC",type="both")
summary(ca.jo(y,K=6,ecdet="trend",type="trace"))
```

```
##
## ######################
## # Johansen-Procedure #
## ######################
##
## Test type: trace statistic , with linear trend in cointegration
##
## Eigenvalues (lambda):
## [1]  8.476513e-02  5.602288e-02  1.752751e-02 -3.469447e-18
##
## Values of teststatistic and critical values of test:
##
##           test 10pct  5pct  1pct
## r <= 2 |  4.85 10.49 12.25 16.26
## r <= 1 | 20.64 22.76 25.32 30.45
## r = 0  | 44.91 39.06 42.44 48.45
##
## Eigenvectors, normalised to first column:
## (These are the cointegration relations)
##
##            prod.l6       n.l6  irate.l6   trend.l6
## prod.l6   1.000000  1.0000000  1.000000  1.0000000
## n.l6      8.018620 -0.3806726  3.815333 -1.9863216
## irate.l6  3.263444  2.5855621 -1.534900 -0.4679434
## trend.l6 -1.864105 -0.7100683 -1.368620 -0.2072208
##
## Weights W:
## (This is the loading matrix)
##
##              prod.l6         n.l6      irate.l6      trend.l6
```

```
## prod.d   0.011070339 -0.044338618 -0.0017956079 -1.538321e-15
## n.d     -0.004334727 -0.002194706 -0.0007920478  5.900478e-16
## irate.d -0.005423616 -0.011995969  0.0048400927  2.784678e-16
```

```
summary(ca.jo(y,K=6,ecdet="trend",type="eigen"))
```

```
##
## ######################
## # Johansen-Procedure #
## ######################
##
## Test type: maximal eigenvalue statistic (lambda max) , with linear trend in cointegra
##
## Eigenvalues (lambda):
## [1]  8.476513e-02  5.602288e-02  1.752751e-02 -3.469447e-18
##
## Values of teststatistic and critical values of test:
##
##           test 10pct  5pct  1pct
## r <= 2 |  4.85 10.49 12.25 16.26
## r <= 1 | 15.80 16.85 18.96 23.65
## r = 0  | 24.27 23.11 25.54 30.34
##
## Eigenvectors, normalised to first column:
## (These are the cointegration relations)
##
##            prod.l6       n.l6  irate.l6   trend.l6
## prod.l6   1.000000  1.0000000  1.000000  1.0000000
## n.l6      8.018620 -0.3806726  3.815333 -1.9863216
## irate.l6  3.263444  2.5855621 -1.534900 -0.4679434
## trend.l6 -1.864105 -0.7100683 -1.368620 -0.2072208
##
## Weights W:
## (This is the loading matrix)
##
##              prod.l6         n.l6      irate.l6      trend.l6
## prod.d   0.011070339 -0.044338618 -0.0017956079 -1.538321e-15
## n.d     -0.004334727 -0.002194706 -0.0007920478  5.900478e-16
## irate.d -0.005423616 -0.011995969  0.0048400927  2.784678e-16
```

```
#Cointegration rank of 1


serial.test(var.aic, type="BG")
```

```
##
##  Breusch-Godfrey LM test
##
## data:  Residuals of VAR object var.aic
## Chi-squared = 70.524, df = 45, p-value = 0.00887
```

```
serial.test(var.aic,type="PT.adjusted")
```

```
##
##  Portmanteau Test (adjusted)
##
## data:  Residuals of VAR object var.aic
## Chi-squared = 118.33, df = 90, p-value = 0.02432

#Residuals are serially correlated
#Since this is for experimentation and augmentation,
#I accept the misspecification

#Step 1.1: Lags based on BIC
var.bic=VAR(y,p=0,lag.max = 8, ic="SC",type="both")
summary(ca.jo(y,K=2,type="eigen"))

##
## ######################
## # Johansen-Procedure #
## ######################
##
## Test type: maximal eigenvalue statistic (lambda max) , with linear trend
##
## Eigenvalues (lambda):
## [1] 0.039144058 0.028314736 0.003509263
##
## Values of teststatistic and critical values of test:
##
##           test 10pct  5pct  1pct
## r <= 2 |  0.98  6.50  8.18 11.65
## r <= 1 |  7.99 12.91 14.90 19.19
## r = 0  | 11.10 18.90 21.07 25.75
##
## Eigenvectors, normalised to first column:
## (These are the cointegration relations)
##
##            prod.l2       n.l2   irate.l2
## prod.l2   1.000000     1.0000  1.0000000
## n.l2     -5.576858  -285.0563 -3.7161694
## irate.l2  3.490491 -2069.4949  0.1692435
##
## Weights W:
## (This is the loading matrix)
##
##              prod.l2         n.l2      irate.l2
## prod.d  -0.016231361 1.947681e-06  0.0009672336
## n.d      0.001497368 2.824812e-06  0.0004788235
## irate.d -0.001090564 1.488560e-05 -0.0004682957

serial.test(var.bic, type="PT.adjusted")

##
##  Portmanteau Test (adjusted)
##
```

```
## data:  Residuals of VAR object var.bic
## Chi-squared = 231.67, df = 117, p-value = 1.469e-09

#Serial correlation, so reject BIC Model

#### Granger causality Test ####
var.granger = VAR(y,p=7,type="both")
causality(var.granger, cause="irate")

## $Granger
##
##  Granger causality H0: irate do not Granger-cause prod n
##
## data:  VAR object var.granger
## F-Test = 3.1355, df1 = 14, df2 = 750, p-value = 8.57e-05
##
##
## $Instant
##
##  H0: No instantaneous causality between: irate and prod n
##
## data:  VAR object var.granger
## Chi-squared = 12.579, df = 2, p-value = 0.001856

causality(var.granger, cause="irate",boot=T, boot.runs=3000)

## $Granger
##
##  Granger causality H0: irate do not Granger-cause prod n
##
## data:  VAR object var.granger
## F-Test = 3.1355, boot.runs = 3000, p-value = 0.01733
##
##
## $Instant
##
##  H0: No instantaneous causality between: irate and prod n
##
## data:  VAR object var.granger
## Chi-squared = 12.579, df = 2, p-value = 0.001856

#There is Granger-causality: Including irate makes sense

#IDENTIFICATION: Page 228 in the book

            #### Estimating the VECM ####
#Important notice: In R I cannot get the cointegration equation with trend
#So that the estimated cointegration equation with trend actually comes from
#STATA

library("tsDyn")
vecm = VECM(y, lag=5,r=1, include="both",estim="ML")
```

```
#toLatex(vecm) ; Get coefficients in LaTeX
t = seq(1,280)
coin.equation = prod + 6.52*n+3.07*irate-1.61*t-185
plot(t,coin.equation,type="l",ylab="Predicted Cointegrated Equation",
     xlab="")
```

```
adf.test(coin.equation)
```

```
## Augmented Dickey-Fuller Test
## alternative: stationary
##
## Type 1: no drift no trend
##      lag    ADF p.value
## [1,]   0 -0.331   0.549
## [2,]   1 -0.621   0.457
## [3,]   2 -0.532   0.489
## [4,]   3 -0.620   0.457
## [5,]   4 -0.459   0.512
## [6,]   5 -0.454   0.513
## Type 2: with drift no trend
##      lag   ADF p.value
## [1,]   0 -2.04  0.3109
## [2,]   1 -3.33  0.0156
## [3,]   2 -2.98  0.0404
## [4,]   3 -3.61  0.0100
## [5,]   4 -3.26  0.0189
## [6,]   5 -3.35  0.0148
## Type 3: with drift and trend
##      lag   ADF p.value
## [1,]   0 -2.38  0.4146
## [2,]   1 -3.88  0.0153
## [3,]   2 -3.49  0.0437
## [4,]   3 -4.23  0.0100
## [5,]   4 -3.77  0.0205
## [6,]   5 -3.89  0.0143
## ----
## Note: in fact, p.value = 0.01 means p.value <= 0.01
```

```
#With drift and drift+trend reject non-stationarity for alpha=5%


  #### Robustness of VECM ####
#Robustness check: r=1 and no trend in equation
coin.robust = prod -5.62*n+1.5*irate-1.08
plot(t,coin.robust, type="l")
```

```
adf.test(coin.robust) #Cannot reject Null, not stationary
```

```
## Augmented Dickey-Fuller Test
## alternative: stationary
##
## Type 1: no drift no trend
##      lag    ADF p.value
```

```
## [1,]   0 -0.117   0.610
## [2,]   1 -0.192   0.589
## [3,]   2 -0.287   0.561
## [4,]   3 -0.312   0.554
## [5,]   4 -0.308   0.555
## [6,]   5 -0.221   0.580
## Type 2: with drift no trend
##      lag   ADF p.value
## [1,]   0 -1.40   0.555
## [2,]   1 -1.70   0.447
## [3,]   2 -1.84   0.391
## [4,]   3 -1.99   0.331
## [5,]   4 -2.44   0.153
## [6,]   5 -2.28   0.217
## Type 3: with drift and trend
##      lag   ADF p.value
## [1,]   0 -1.51   0.783
## [2,]   1 -1.81   0.655
## [3,]   2 -2.01   0.572
## [4,]   3 -2.16   0.506
## [5,]   4 -2.55   0.342
## [6,]   5 -2.35   0.428
## ----
## Note: in fact, p.value = 0.01 means p.value <= 0.01
```

```
#### Robustness check: r=2 and trend in equation ####
coin.robust.1 = prod +2.65*irate-0.75*t-101
adf.test(coin.robust.1)
```

```
## Augmented Dickey-Fuller Test
## alternative: stationary
##
## Type 1: no drift no trend
##      lag    ADF p.value
## [1,]   0 -1.024   0.313
## [2,]   1 -1.120   0.279
## [3,]   2 -0.893   0.360
## [4,]   3 -0.948   0.340
## [5,]   4 -0.916   0.352
## [6,]   5 -0.938   0.344
## Type 2: with drift no trend
##      lag   ADF p.value
## [1,]   0 -3.20  0.0218
## [2,]   1 -3.57  0.0100
## [3,]   2 -2.90  0.0477
## [4,]   3 -3.09  0.0301
## [5,]   4 -3.06  0.0329
## [6,]   5 -3.15  0.0244
## Type 3: with drift and trend
##      lag   ADF p.value
## [1,]   0 -3.20  0.0872
```

```
## [2,]   1 -3.57  0.0365
## [3,]   2 -2.90  0.1952
## [4,]   3 -3.09  0.1181
## [5,]   4 -3.06  0.1308
## [6,]   5 -3.15  0.0964
## ----
## Note: in fact, p.value = 0.01 means p.value <= 0.01
```

```
#Cannot reject Null for trend: Higher p-values for drift only compared to
#baseline model
coin.robust.2 = n+0.06*irate-0.135*t-17
adf.test(coin.robust.2)
```

```
## Augmented Dickey-Fuller Test
## alternative: stationary
##
## Type 1: no drift no trend
##      lag    ADF p.value
## [1,]   0 -0.257   0.570
## [2,]   1 -0.610   0.461
## [3,]   2 -0.684   0.434
## [4,]   3 -0.618   0.458
## [5,]   4 -0.459   0.512
## [6,]   5 -0.317   0.553
## Type 2: with drift no trend
##      lag   ADF p.value
## [1,]   0 -1.36  0.5724
## [2,]   1 -2.76  0.0702
## [3,]   2 -2.93  0.0449
## [4,]   3 -2.91  0.0469
## [5,]   4 -2.98  0.0404
## [6,]   5 -2.36  0.1864
## Type 3: with drift and trend
##      lag   ADF p.value
## [1,]   0 -1.58  0.7520
## [2,]   1 -3.09  0.1189
## [3,]   2 -3.33  0.0660
## [4,]   3 -3.28  0.0750
## [5,]   4 -3.22  0.0853
## [6,]   5 -2.54  0.3470
## ----
## Note: in fact, p.value = 0.01 means p.value <= 0.01
```

```
#Cannot reject Null for second cointegration equation

#### Robustness check: r=2 and no trend in equation ####
coin.robust.3 = prod +47.12*irate-204.78
adf.test(coin.robust.3)
```

```
## Augmented Dickey-Fuller Test
## alternative: stationary
##
```

```
## Type 1: no drift no trend
##      lag   ADF p.value
## [1,]   0 -1.11   0.281
## [2,]   1 -1.44   0.165
## [3,]   2 -1.10   0.287
## [4,]   3 -1.46   0.158
## [5,]   4 -1.30   0.214
## [6,]   5 -1.59   0.112
## Type 2: with drift no trend
##      lag   ADF p.value
## [1,]   0 -2.21  0.2465
## [2,]   1 -2.63  0.0918
## [3,]   2 -2.21  0.2451
## [4,]   3 -2.69  0.0814
## [5,]   4 -2.50  0.1318
## [6,]   5 -2.93  0.0452
## Type 3: with drift and trend
##      lag   ADF p.value
## [1,]   0 -2.02   0.567
## [2,]   1 -2.53   0.353
## [3,]   2 -2.02   0.568
## [4,]   3 -2.61   0.319
## [5,]   4 -2.38   0.418
## [6,]   5 -2.88   0.203
## ----
## Note: in fact, p.value = 0.01 means p.value <= 0.01

#Not Stationary, p-values very high


            #### Forecasting ####
#Note that the Forecasts have been conducted in STATA, as the implementation
#is much more user friendly there. I took the Forecast Data from STATA
#to implement the plots

data.forecast = read.csv2("C:/Users/FORECASTDATA.csv")

t = seq(1,12)
prod.f = data.forecast[,3]
prod.lb = data.forecast[,4]
prod.up = data.forecast[,5]
prod.true= data.forecast[,12]
hours.f = data.forecast[,6]
hours.lb = data.forecast[,7]
hours.up = data.forecast[,8]
hours.true = data.forecast[,13]
irate.f = data.forecast[,9]
irate.lb = data.forecast[,10]
irate.up = data.forecast[,11]
irate.true =data.forecast[,14]
```

```
par(mfrow=c(1,3))

plot(t,prod.f, ylim=c(min(prod.lb),max(prod.up)),
     type="l",ylab="Productivity",xlab="")
lines(t,prod.lb, lty="dashed",col="red")
lines(t,prod.up, lty="dashed",col="red")
lines(t,prod.true,col="blue")

plot(t,hours.f, ylim=c(min(hours.lb),max(hours.up)),
     type="l",ylab="Hours",xlab="")
lines(t,hours.lb, lty="dashed",col="red")
lines(t,hours.up, lty="dashed",col="red")
lines(t,hours.true,col="blue")

plot(t,irate.f,type="l",ylab="Interest rate",
     xlab="",ylim=c(min(irate.true),max(irate.true)))
#ylim=c(min(irate.lb),max(irate.up))
lines(t,irate.lb, lty="dashed",col="red")
lines(t,irate.up, lty="dashed",col="red")
lines(t,irate.true, col="blue")
```

```
            #### Forecasting only using Gali's data ####

data.forecast.Gali = read.csv2("C:/Forecastdata-onlyuntil1994.csv")

t = seq(1,12)
prod.f = data.forecast.Gali[,3]
prod.lb = data.forecast.Gali[,4]
prod.up = data.forecast.Gali[,5]
prod.true= data.forecast.Gali[,12]
hours.f = data.forecast.Gali[,6]
hours.lb = data.forecast.Gali[,7]
hours.up = data.forecast.Gali[,8]
hours.true = data.forecast.Gali[,13]
irate.f = data.forecast.Gali[,9]
irate.lb = data.forecast.Gali[,10]
irate.up = data.forecast.Gali[,11]
irate.true =data.forecast.Gali[,14]

par(mfrow=c(1,3))

plot(t,prod.f, ylim=c(min(prod.lb),max(prod.up)),
     type="l",ylab="Produktivity",xlab="")
lines(t,prod.lb, lty="dashed",col="red")
lines(t,prod.up, lty="dashed",col="red")
lines(t,prod.true,col="blue")

plot(t,hours.f, ylim=c(min(hours.lb),max(hours.up)),type="l",
     ylab="Hours",xlab="")
lines(t,hours.lb, lty="dashed",col="red")
```

```
lines(t,hours.up, lty="dashed",col="red")
lines(t,hours.true,col="blue")

plot(t,irate.f,type="l",ylab="Interest rate",xlab="",
     ylim=c(min(irate.lb),max(irate.up)))
#ylim=c(min(irate.lb),max(irate.up))
lines(t,irate.lb, lty="dashed",col="red")
lines(t,irate.up, lty="dashed",col="red")
lines(t,irate.true, col="blue")
```

```
              ##### Structural VECM ####

LR = matrix(c(NA,0,0,NA,NA,0,NA,NA,NA),ncol=3,byrow=T)
SR = matrix(c(NA,NA,NA,NA,NA,NA,NA,NA,NA),ncol=3,byrow=T)

y = cbind(prod,irate,n)
vecm.s = SVEC(ca.jo(y,K=6,ecdet="trend",type="trace",spec="longrun"),
              LR=LR, SR=SR)

## Warning in SVEC(ca.jo(y, K = 6, ecdet = "trend", type = "trace", spec = "longrun"),
:  The SVEC is just identified.  No test possible.

irf.svecm = irf(vecm.s,cumulative=T,ci=0.86,ortho=T)

#Plot Impulse-Response of labour to a technology shock
plot(irf.svecm$irf$prod[,3],type="l",ylab="Hours",
     ylim=c(min(irf.svecm$Lower$prod[,3]),max(irf.svecm$Upper$prod[,3])))

lines(irf.svecm$Lower$prod[,3],type="l",col="red",lty="dashed")
lines(irf.svecm$Upper$prod[,3],type="l",col="red",lty="dashed")
abline(0,0,col="red")
```

A.6 My Own Analysis

```
library("vars")
library("aTSA")
data = read.csv2("C:/Users/Patrick/Data.csv")

M1 = data[,2]
dm1 = data[2:237,6]

cpi = data[2:237,4]

M2 = data[,3]
irate=data[2:237,7]

            #### Dickey-Fuller Test ####

adf.test(M1) #Not stationary
```

```
## Augmented Dickey-Fuller Test
## alternative: stationary
##
## Type 1: no drift no trend
##      lag   ADF p.value
## [1,]   0 18.73    0.99
## [2,]   1  5.07    0.99
## [3,]   2  4.28    0.99
## [4,]   3  3.44    0.99
## [5,]   4  2.76    0.99
## Type 2: with drift no trend
##      lag   ADF p.value
## [1,]   0 13.47    0.99
## [2,]   1  4.10    0.99
## [3,]   2  3.46    0.99
## [4,]   3  2.71    0.99
## [5,]   4  2.07    0.99
## Type 3: with drift and trend
##      lag   ADF p.value
## [1,]   0 6.485    0.99
## [2,]   1 1.793    0.99
## [3,]   2 1.382    0.99
## [4,]   3 0.824    0.99
## [5,]   4 0.257    0.99
## ----
## Note: in fact, p.value = 0.01 means p.value <= 0.01
```

```
adf.test(dm1) #Stationary: Small p-values for all lags and specifications
```

```
## Augmented Dickey-Fuller Test
## alternative: stationary
##
## Type 1: no drift no trend
##      lag   ADF p.value
```

```
## [1,]   0 -4.46  0.0100
## [2,]   1 -3.34  0.0100
## [3,]   2 -2.87  0.0100
## [4,]   3 -2.68  0.0100
## [5,]   4 -2.13  0.0341
## Type 2: with drift no trend
##      lag   ADF p.value
## [1,]   0 -7.09    0.01
## [2,]   1 -5.46    0.01
## [3,]   2 -4.93    0.01
## [4,]   3 -4.73    0.01
## [5,]   4 -3.85    0.01
## Type 3: with drift and trend
##      lag   ADF p.value
## [1,]   0 -7.09  0.0100
## [2,]   1 -5.46  0.0100
## [3,]   2 -4.92  0.0100
## [4,]   3 -4.72  0.0100
## [5,]   4 -3.83  0.0181
## ----
## Note: in fact, p.value = 0.01 means p.value <= 0.01

adf.test(cpi) #Even though rejected for some specifications, the plot shows

## Augmented Dickey-Fuller Test
## alternative: stationary
##
## Type 1: no drift no trend
##      lag   ADF p.value
## [1,]   0 -1.17  0.2596
## [2,]   1 -1.72  0.0854
## [3,]   2 -1.73  0.0831
## [4,]   3 -1.89  0.0593
## [5,]   4 -1.25  0.2320
## Type 2: with drift no trend
##      lag   ADF p.value
## [1,]   0 -1.95  0.3486
## [2,]   1 -2.93  0.0458
## [3,]   2 -2.95  0.0434
## [4,]   3 -3.25  0.0196
## [5,]   4 -2.25  0.2312
## Type 3: with drift and trend
##      lag   ADF p.value
## [1,]   0 -2.30  0.4505
## [2,]   1 -3.32  0.0672
## [3,]   2 -3.35  0.0617
## [4,]   3 -3.69  0.0250
## [5,]   4 -2.74  0.2635
## ----
## Note: in fact, p.value = 0.01 means p.value <= 0.01
```

```r
#That it doesn't exhibit a trend or any kind of non-stationarity
plot(cpi,type="l",main="Inflation Rate",ylab="",xlab="")

plot(dm1,type="l")


            #### Reduced Form VAR ####
y = cbind(cpi,dm1)
var.aic =VAR(y, p=0, lag.max=8, type="const", ic="AIC")
serial.test(var.aic)#Serial Correlation present

##
##  Portmanteau Test (asymptotic)
##
## data:  Residuals of VAR object var.aic
## Chi-squared = 76.499, df = 40, p-value = 0.0004498

#All vars from lags 0 to 8 express serial correlation.


            #### SVAR ####
B = matrix(c(NA, 0,NA,NA),byrow=T,ncol=2)
svar = SVAR(var.aic, estmethod="direct",Bmat=B,max.iter=500)

## Warning in SVAR(var.aic, estmethod = "direct", Bmat = B, max.iter = 500):  The
B-model is just identified.  No test possible.

summary(svar)

##
## SVAR Estimation Results:
## ========================
##
## Call:
## SVAR(x = var.aic, estmethod = "direct", Bmat = B, max.iter = 500)
##
## Type: B-model
## Sample size: 230
## Log Likelihood: 544.268
## Method: direct
## Number of iterations: 224
## Convergence code: 0
##
## Estimated A matrix:
##     cpi dm1
## cpi   1   0
## dm1   0   1
##
## Estimated B matrix:
##           cpi      dm1
## cpi  0.590995 0.000000
## dm1 -0.002395 0.009295
##
## Covariance matrix of reduced form residuals (*100):
##         cpi       dm1
```

```
## cpi 34.9275 -0.141542
## dm1 -0.1415  0.009213

#Since the programme writes it as Au_t = Bw_t, I need a restriction on B
#For the short-run restriction given in the paper to hold


            #### Impulse-Response Functions 2VAR ####
#The model has been estimated in STATA and the graph will be made in R
irf.2var = read.csv2("C:/Users/IRF_2VAR.csv")
plot(irf.2var[,2],type="l",ylim=c(min(irf.2var[,3]),max(irf.2var[,4]))
     ,ylab="Inflation",main="Quarterly Data")
lines(irf.2var[,3],type="l",lty="dashed",col="red")
lines(irf.2var[,4],type="l",lty="dashed",col="red")
abline(0,0,col="red")

            #### Impulse-Response Function 3VAR ####
y = cbind(cpi,dm1,irate)
var.granger = VAR(y,p=0,lag.max=8,type="both")
causality(var.granger, cause="irate")

##  Granger causality H0: irate do not Granger-cause cpi dm1
##
## data:  VAR object var.granger
## F-Test = 5.5313, df1 = 12, df2 = 630, p-value = 4.937e-09
##
## $Instant
##
##  H0: No instantaneous causality between: irate and cpi dm1
##
## data:  VAR object var.granger
## Chi-squared = 7.0447, df = 2, p-value = 0.02953

#The model has been estimated in STATA and the graph will be made in R
irf.3var = read.csv2("C:/Users/IRF_3VAR_Model.csv")
plot(irf.2var[,2],type="l",ylim=c(min(irf.2var[,3]),max(irf.2var[,4]))
     ,ylab="Inflation")
lines(irf.3var[,3],type="l",lty="dashed",col="red")
lines(irf.3var[,4],type="l",lty="dashed",col="red")
abline(0,0,col="red")

            #### Impulse-Response Function 2VAR with Monthly DATA ####
#The model has been estimated in STATA
irf.2var.monthly = read.csv2("C:/IRF_2VAR_monthly.csv")
plot(irf.2var.monthly[,2],type="l",ylim=c(min(irf.2var.monthly[,3]),max(irf.2var.monthly
     ,ylab="Inflation",main="Monthly Data")
lines(irf.2var.monthly[,3],type="l",lty="dashed",col="red")
lines(irf.2var.monthly[,4],type="l",lty="dashed",col="red")
abline(0,0,col="red")
```