Final_2020_Solutions

May 30, 2020

1 Final 2020 Solutions

This is a general guide to the solutions only. It does not provide detailed answers. It focus on the core parts of the answers.

1.0.1 Exercise 1)

[1]:
```
library(readxl)
library(urca)
library(tseries)
```

```
Registered S3 method overwritten by 'quantmod':
  method            from
  as.zoo.data.frame zoo
```

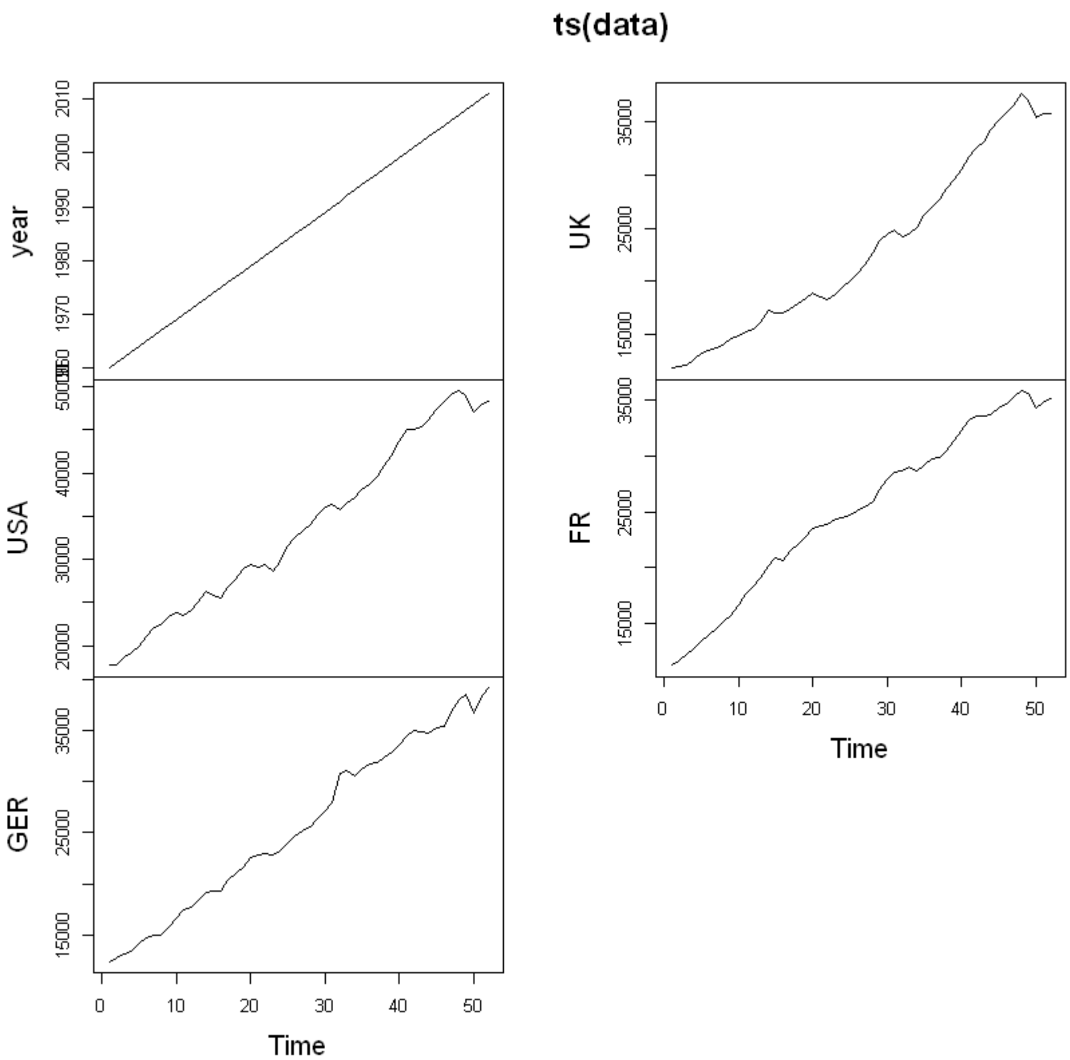
[2]:
```
data = read_excel("C:/Users/Fabio/Dropbox/Lecture_Macroeconometrics/Data/
↪data_final_2020_Q1.xls")
#data = read_excel("/Users/joaobduarte/Dropbox/Lecture_Macroeconometrics/Data/
↪data_final_2020_Q1.xls")
```

1 a) Pretesting the variables for their order of integration:

[3]:
```
head(data)
```

year	USA	GER	UK	FR
1960	17747	12352	11879	11272
1961	17862	12753	12107	11688
1962	18655	13193	12172	12268
1963	19192	13433	12662	12805
1964	20023	14184	13316	13485
1965	21044	14779	13571	14000

[4]:
```
plot(ts(data))
```



[5]:
```
adf.test(log(data$USA))
adf.test(log(data$GER))
adf.test(log(data$UK))
adf.test(log(data$FR))
```

```
Augmented Dickey-Fuller Test

data:  log(data$USA)
Dickey-Fuller = -2.1328, Lag order = 3, p-value = 0.521
alternative hypothesis: stationary



Augmented Dickey-Fuller Test
```

```
data:  log(data$GER)
Dickey-Fuller = -1.1427, Lag order = 3, p-value = 0.9075
alternative hypothesis: stationary



Augmented Dickey-Fuller Test

data:  log(data$UK)
Dickey-Fuller = -2.2745, Lag order = 3, p-value = 0.464
alternative hypothesis: stationary



Augmented Dickey-Fuller Test

data:  log(data$FR)
Dickey-Fuller = -1.9966, Lag order = 3, p-value = 0.5758
alternative hypothesis: stationary
```

[6]:

```
adf.test(diff(log(data$USA)))
adf.test(diff(log(data$GER)))
adf.test(diff(log(data$UK)))
adf.test(diff(log(data$FR)))
```

```
Augmented Dickey-Fuller Test

data:  diff(log(data$USA))
Dickey-Fuller = -4.1498, Lag order = 3, p-value = 0.01001
alternative hypothesis: stationary



Warning message in adf.test(diff(log(data$GER))):
"p-value smaller than printed p-value"

Augmented Dickey-Fuller Test

data:  diff(log(data$GER))
Dickey-Fuller = -4.3711, Lag order = 3, p-value = 0.01
alternative hypothesis: stationary



Augmented Dickey-Fuller Test

data:  diff(log(data$UK))
```

```
Dickey-Fuller = -2.9327, Lag order = 3, p-value = 0.1994
alternative hypothesis: stationary


	Augmented Dickey-Fuller Test

data:  diff(log(data$FR))
Dickey-Fuller = -3.3368, Lag order = 3, p-value = 0.0755
alternative hypothesis: stationary
```

[7]:
```
adf.test(diff(diff(log(data$USA))))
adf.test(diff(diff(log(data$GER))))
adf.test(diff(diff(log(data$UK))))
adf.test(diff(diff(log(data$FR))))
```

```
Warning message in adf.test(diff(diff(log(data$USA)))):
"p-value smaller than printed p-value"

	Augmented Dickey-Fuller Test

data:  diff(diff(log(data$USA)))
Dickey-Fuller = -6.1968, Lag order = 3, p-value = 0.01
alternative hypothesis: stationary


Warning message in adf.test(diff(diff(log(data$GER)))):
"p-value smaller than printed p-value"

	Augmented Dickey-Fuller Test

data:  diff(diff(log(data$GER)))
Dickey-Fuller = -5.4902, Lag order = 3, p-value = 0.01
alternative hypothesis: stationary


Warning message in adf.test(diff(diff(log(data$UK)))):
"p-value smaller than printed p-value"

	Augmented Dickey-Fuller Test

data:  diff(diff(log(data$UK)))
Dickey-Fuller = -5.7332, Lag order = 3, p-value = 0.01
alternative hypothesis: stationary
```

```
Warning message in adf.test(diff(diff(log(data$FR)))):
"p-value smaller than printed p-value"
```

```
Augmented Dickey-Fuller Test

data:  diff(diff(log(data$FR)))
Dickey-Fuller = -4.9542, Lag order = 3, p-value = 0.01
alternative hypothesis: stationary
```

The variables with exception of the UK and France seem to be integrated of order one. The other two seem to be of order two.

[8]:
```
vecm = ca.jo(log(data[,2:5]), K = 2, spec = "transitory", type = "trace")
```

[9]:
```
summary(vecm)
```

```
######################
# Johansen-Procedure #
######################

Test type: trace statistic , with linear trend

Eigenvalues (lambda):
[1] 0.50825523 0.27196935 0.19777598 0.04793494

Values of teststatistic and critical values of test:

          test 10pct  5pct  1pct
r <= 3 |  2.46  6.50  8.18 11.65
r <= 2 | 13.47 15.66 17.95 23.52
r <= 1 | 29.35 28.71 31.52 37.22
r = 0  | 64.83 45.23 48.28 55.43

Eigenvectors, normalised to first column:
(These are the cointegration relations)

             USA.l1     GER.l1    UK.l1      FR.l1
USA.l1  1.000000000  1.0000000  1.000000  1.0000000
GER.l1  0.436967968 -0.3065273  1.328313  1.0039269
UK.l1  -1.014610727 -0.4776611 -1.267007  0.2005609
FR.l1   0.005283738 -0.1240861 -1.003626 -2.0527199

Weights W:
(This is the loading matrix)

           USA.l1     GER.l1      UK.l1        FR.l1
USA.d -0.13388793 -0.2402984  0.06278952 -0.018899250
```

```
GER.d -0.10017801  0.3013174 -0.08677278 -0.025310102
UK.d  -0.02963721  0.1061705  0.12916666 -0.020862541
FR.d  -0.11736909  0.2577538  0.03907042 -0.003775357
```

[10]:
```
vecm1 = ca.jo(log(data[,2:5]), spec = "transitory", K = 2, type = "eigen")
```

[11]:
```
summary(vecm1)
```

```
######################
# Johansen-Procedure #
######################

Test type: maximal eigenvalue statistic (lambda max) , with linear trend

Eigenvalues (lambda):
[1] 0.50825523 0.27196935 0.19777598 0.04793494

Values of teststatistic and critical values of test:

          test 10pct  5pct  1pct
r <= 3 |  2.46  6.50  8.18 11.65
r <= 2 | 11.02 12.91 14.90 19.19
r <= 1 | 15.87 18.90 21.07 25.75
r = 0  | 35.49 24.78 27.14 32.14

Eigenvectors, normalised to first column:
(These are the cointegration relations)

             USA.l1     GER.l1     UK.l1      FR.l1
USA.l1  1.000000000  1.0000000  1.000000  1.0000000
GER.l1  0.436967968 -0.3065273  1.328313  1.0039269
UK.l1  -1.014610727 -0.4776611 -1.267007  0.2005609
FR.l1   0.005283738 -0.1240861 -1.003626 -2.0527199

Weights W:
(This is the loading matrix)

           USA.l1     GER.l1      UK.l1        FR.l1
USA.d -0.13388793 -0.2402984  0.06278952 -0.018899250
GER.d -0.10017801  0.3013174 -0.08677278 -0.025310102
UK.d  -0.02963721  0.1061705  0.12916666 -0.020862541
FR.d  -0.11736909  0.2577538  0.03907042 -0.003775357
```

Both approaches lead to the same result. We reject that there is no cointegration relationship and cannot reject that there is 1 cointegration relationship. Therefore, we conclude that there is one statistically significant cointegration relationship. The log series are cointegrated, since we have a statistically significant eigenvalue of the π matrix, which is evidence for this.

Note that since we found out that some variables are I(2), it must be the case that we have multicointegration. I.e. the linear combination of the I(2) with other variables produce I(1), which in turn have a linear combination with other I(1) variables that is I(0). You could have tried including a constant and a trend. But in this case, the results are similar and hence we go with the smaller model.

1 b)

[12]:
```
coint_V = attr(vecm, "V")
print(coint_V[,1])
```

```
     USA.l1       GER.l1        UK.l1        FR.l1
1.000000000  0.436967968 -1.014610727  0.005283738
```

As expected, given the strong international trade links between US and UK, their GDP are tied together in the long-run almost 1 to 1. At the same time France's GDP is not so interconnected with the other three countries.

1 c) Since we have a cointegration relationship, following the lecture note we know that there exists a VECM representation of the VAR. And moreover, we know that if all series are I(1), the first difference will be I(0) and since the coitegrated vector is also I(0), the resulting residuals would be I(0) as well. However, in this question that may not be the case since not all the variables are I(1). Hence, lets test this by translating the VECM into a VAR and testing its resulting residuals after running OLS:

[13]:
```
library(vars)
svecm = vec2var(vecm, r = 1)
```

```
Warning message:
"package 'vars' was built under R version 3.6.3"Loading required package: MASS
Loading required package: strucchange
Warning message:
"package 'strucchange' was built under R version 3.6.3"Loading required package:
zoo

Attaching package: 'zoo'

The following objects are masked from 'package:base':

    as.Date, as.Date.numeric

Loading required package: sandwich
Warning message:
"package 'sandwich' was built under R version 3.6.3"Loading required package:
```

```
lmtest
Warning message:
"package 'lmtest' was built under R version 3.6.3"
```

[14]:
```
resid = svecm$resid
length(resid)

adf.test(resid[,1])
adf.test(resid[,2])
adf.test(resid[,3])
adf.test(resid[,4])
```

200

```
Warning message in adf.test(resid[, 1]):
"p-value smaller than printed p-value"
```

```
Augmented Dickey-Fuller Test

data:  resid[, 1]
Dickey-Fuller = -4.2257, Lag order = 3, p-value = 0.01
alternative hypothesis: stationary
```

```
Warning message in adf.test(resid[, 2]):
"p-value smaller than printed p-value"
```

```
Augmented Dickey-Fuller Test

data:  resid[, 2]
Dickey-Fuller = -4.2428, Lag order = 3, p-value = 0.01
alternative hypothesis: stationary
```

```
Augmented Dickey-Fuller Test

data:  resid[, 3]
Dickey-Fuller = -3.3894, Lag order = 3, p-value = 0.06782
alternative hypothesis: stationary
```

```
Augmented Dickey-Fuller Test

data:  resid[, 4]
Dickey-Fuller = -4.0932, Lag order = 3, p-value = 0.0128
alternative hypothesis: stationary
```

We need to test the residuals against the Engle-Granger critical values. Using the critical value of -4.154 and -3.853 for the model without a trend at the 5% and 10% level, we would conclude that most of the residuals with exception to the third series are stationary. Doing a regression of the long-run equation using the Engle-Granger method is also fine but a particular application. Its like testing our first equation of the VAR only. Not surprisingly, students that did this will find that the residuals are stationary.

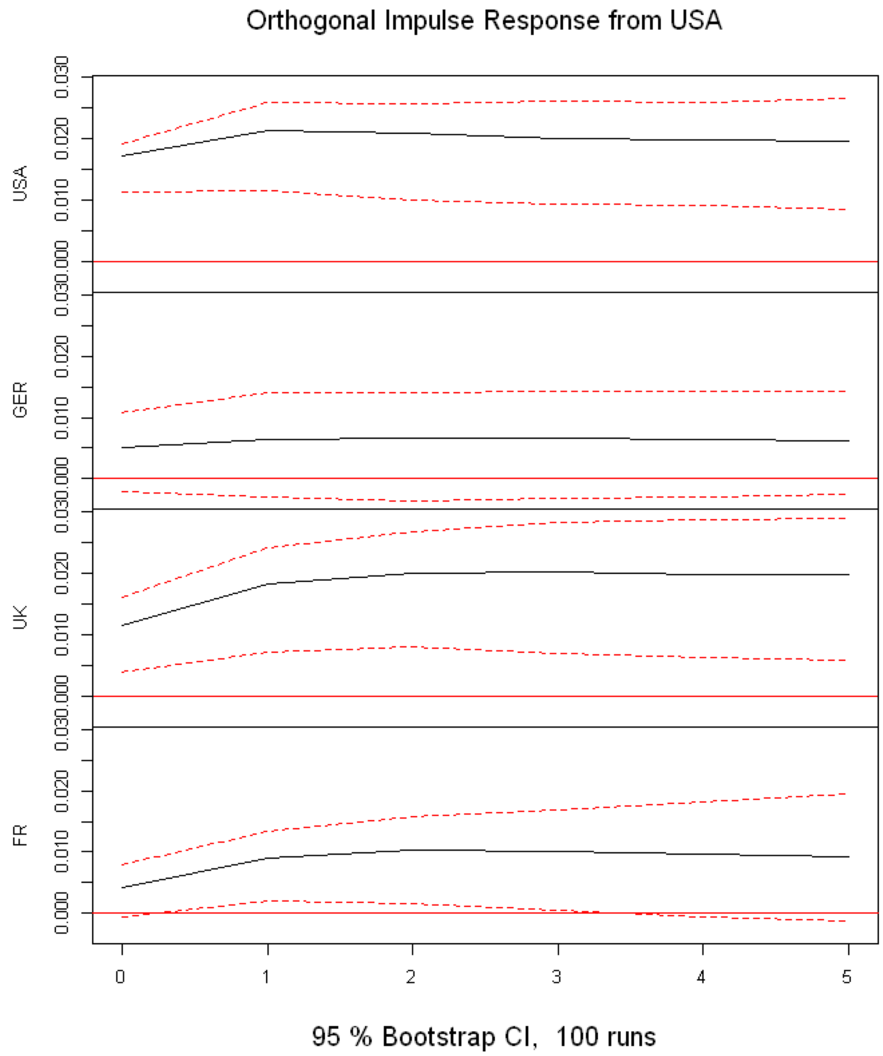
1 d)

[15]:
```
W=attr(vecm, "W")
print(W[,1])
```

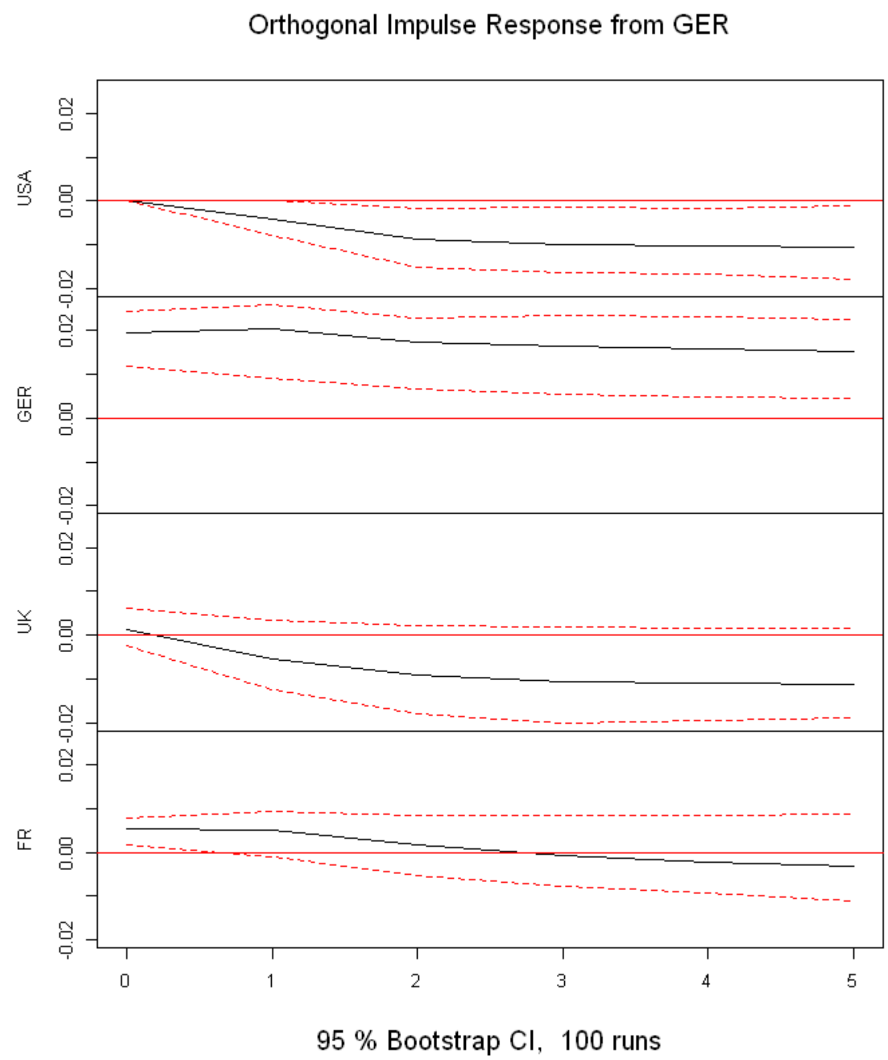
```
      USA.d       GER.d        UK.d        FR.d
-0.13388793 -0.10017801 -0.02963721 -0.11736909
```

The speed of adjustment coefficients give us a sense on how each of the variables adjusts relative to deviations in the long-run equilibrium. If there are positive deviations to the long-run equilibrium, all variables adjust negatively in order to restore equilibrium. The one that ajudjust faster is the US GDP, followed by France, Germany and lastly by the UK. Each unit of deviation in the equilibrum makes the US decrease the GDP by 13% which is very high. Hence, the US will adjust strongly to any small deviations to the long-run equilibrium.

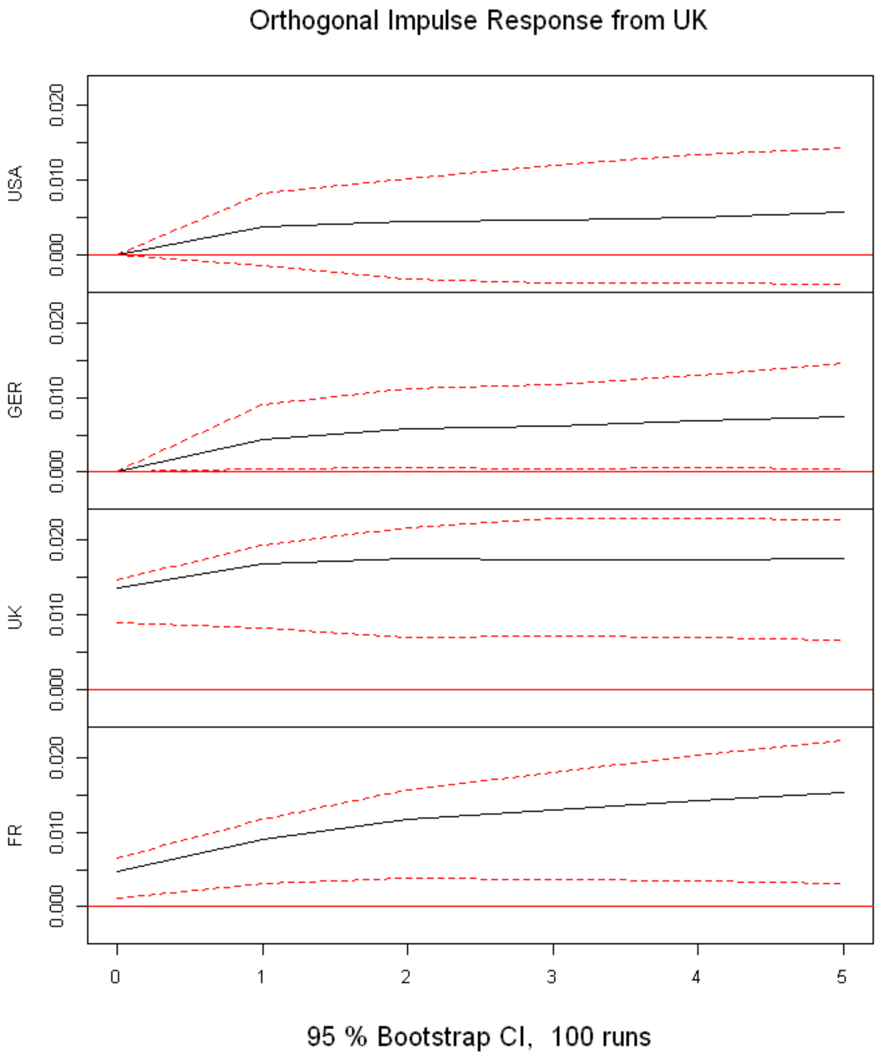
[16]:
```
irf = irf(svecm, n = 5)
plot(irf)
```

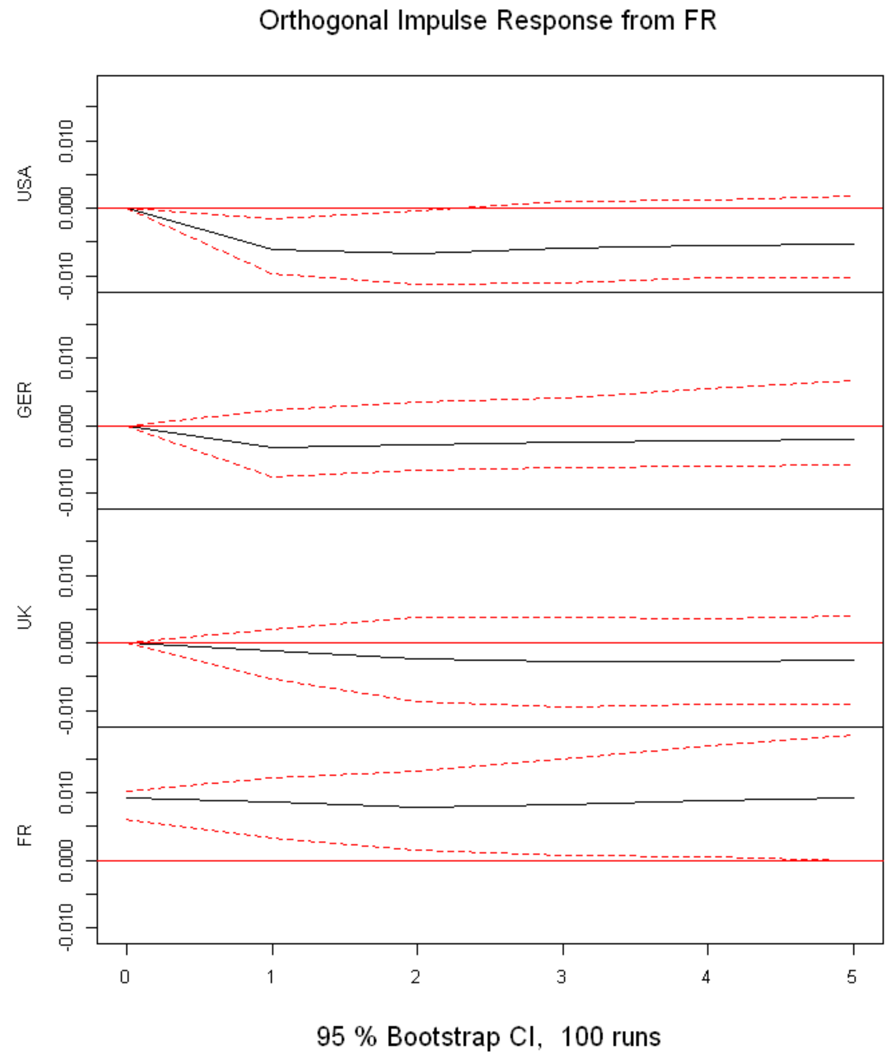


Orthogonal Impulse Response from USA
USA
GER
UK
FR
0.000
0.010
0.020
0.030
0
1
2
3
4
5
95 % Bootstrap CI, 100 runs



Orthogonal Impulse Response from GER
USA
GER
UK
FR
0.02
0.00
-0.02
0
1
2
3
4
5
95 % Bootstrap CI, 100 runs


Orthogonal Impulse Response from UK
USA
GER
UK
FR
0.000
0.010
0.020
0
1
2
3
4
5
95 % Bootstrap CI, 100 runs

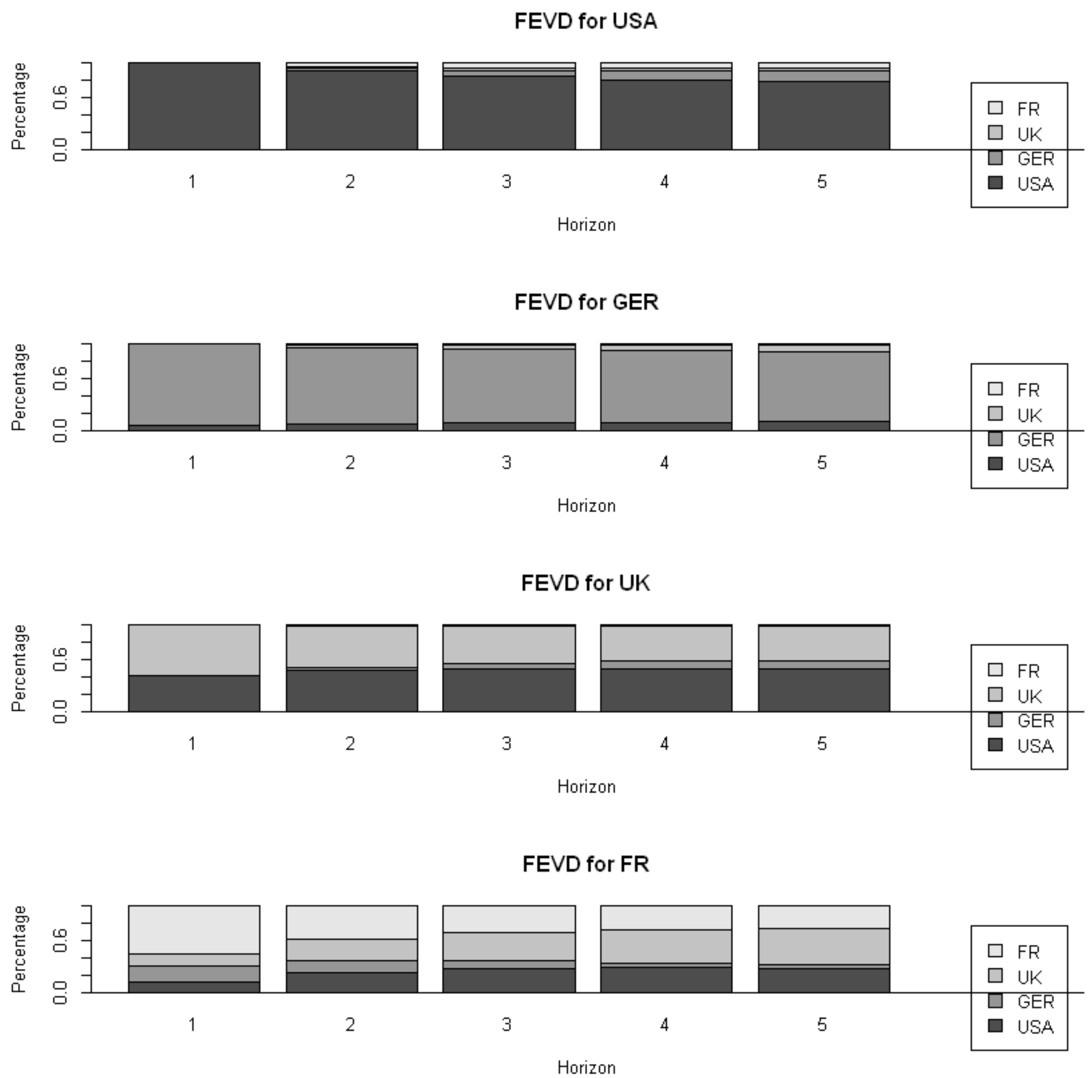


Positive shocks in the US economic activity have significant impact mainly in the UK economy but also in France. It has however, no significant impact on Germany. Shocks in the Germany economy on the other hand, only affect marginally positively France on impact and the first year only. Moreover, it has a tiny negative impact on the US economy, which is perhaps driven by product competition. Positive shocks in the UK economy mostly benefits European countries and not the US. Finally, France has basically a marginal effect on the US and not effect on others.

[17]:
```
fevd = fevd(svecm, n = 5)
print(fevd)
plot(fevd)
```

```
$USA
           USA        GER         UK         FR
[1,] 1.0000000 0.00000000 0.00000000 0.00000000
```

```
[2,] 0.9150784 0.02211728 0.01739191 0.04541243
[3,] 0.8485107 0.06860224 0.02483085 0.05805625
[4,] 0.8119658 0.10010656 0.02839224 0.05953542
[5,] 0.7888915 0.11985673 0.03231454 0.05893725

$GER
            USA       GER         UK         FR
[1,] 0.06173793 0.9382621 0.00000000 0.00000000
[2,] 0.07596388 0.8899785 0.02240153 0.01165605
[3,] 0.08717490 0.8565538 0.04138840 0.01488295
[4,] 0.09457376 0.8336479 0.05634312 0.01543517
[5,] 0.09884922 0.8152718 0.07053265 0.01534636

$UK
           USA        GER        UK          FR
[1,] 0.4168309 0.00435148 0.5788176 0.000000000
[2,] 0.4799572 0.03265853 0.4858075 0.001576811
[3,] 0.4916271 0.06596722 0.4382027 0.004202965
[4,] 0.4904641 0.08882130 0.4149490 0.005765663
[5,] 0.4872697 0.10304236 0.4032435 0.006444449

$FR
           USA        GER        UK        FR
[1,] 0.1167100 0.18951750 0.1464885 0.5472839
[2,] 0.2366566 0.13215521 0.2500106 0.3811775
[3,] 0.2826274 0.08008172 0.3320751 0.3052157
[4,] 0.2887352 0.05510862 0.3855695 0.2705867
[5,] 0.2788007 0.04399631 0.4244223 0.2527808
```



The FEVD, confirm the relative high importance of the US economy in all other economies, except for Germany. Note, however, that even in Germany the US economy explain almost 10% of the variation after 5 years. For the UK, it is clear the importance of the US economy on its own performance.

1.0.2 Exercise 2)

[2]:
```r
# library(xlsx)
library(tsDyn)
library(vars)
library(repr)
```

```
Loading required package: MASS

Loading required package: strucchange

Loading required package: zoo


Attaching package: ‘zoo’


The following objects are masked from ‘package:base’:

    as.Date, as.Date.numeric


Loading required package: sandwich

Loading required package: lmtest
```

[4]:
```
# Importing the data
data2 = read.csv("/Users/joaobduarte/Dropbox/Lecture_Macroeconometrics/Data/
 ↪data_final_2020_Q2.csv")

head(data2)
```

A data.frame: 6 × 129

	sasdate <chr>	RPI <dbl>	W875RX1 <dbl>	DPCERA3M086SBEA <dbl>	CMRMTSPLx <dbl>	RETA <dbl
1	Transform:	5.000	5.0	5.000	5.0	5.00
2	1/1/1959	2437.296	2288.8	17.302	292258.8	18235.
3	2/1/1959	2446.902	2297.0	17.482	294429.5	18369.
4	3/1/1959	2462.689	2314.0	17.647	293425.4	18523.
5	4/1/1959	2478.744	2330.3	17.584	299331.7	18534.
6	5/1/1959	2493.228	2345.8	17.796	301373.0	18679.

2 a)

[5]:
```
# Create a function that transforms the data

transform = function(data){

    for(j in 1:length(data[1,])){
      if(data[1,j] == 1){
        data[2:nrow(data),j] = data[2:nrow(data),j]
      }
      if(data[1,j] == 2){
        data[(2+1):nrow(data),j] = diff(data[2:nrow(data),j])
      } # remember that when you use diff you loose one observation!
```

```
      if(data[1,j] == 3){
        data[(2+2):nrow(data),j] = diff(diff(data[2:nrow(data),j]))
      } # remember that when you use diff (diff) you loose TWO observation!
      if(data[1,j] == 4){
        data[2:nrow(data),j] = log(data[2:nrow(data),j])
      }
      if(data[1,j] == 5){
        data[(2+1):nrow(data),j] = diff(log(data[2:nrow(data),j]))
      } # remember that when you use diff you loose one observation!
      if(data[1,j] == 6){
        data[(2+2):nrow(data),j] = diff(diff(log(data[2:nrow(data),j])))
      } # remember that when you use diff (diff) you loose TWO observation!
      if(data[1,j] == 7){
        data[(2+2):nrow(data),j] = diff((data[3:nrow(data),j])/(data[2:
↪(nrow(data)-1),j])-1)
      }
    }
    return(data)
}
```

[6]:
```
data = ts(data2[,-1])
data_t = transform(data)
```

[7]:
```
data_t = data_t[4:nrow(data_t),] # eliminate the first three obs: 1 trans code,
↪2 first diff, 3 second diff
```

[8]:
```
head(data_t)
```

A matrix: 6×128 of type dbl

RPI	W875RX1	DPCERA3M086SBEA	CMRMTSPLx	RETAI
0.0064311078	0.0073737051	0.009394017	-3.416370e-03	0.00832
0.0064981379	0.0070193859	-0.003576400	1.992879e-02	0.00061
0.0058262762	0.0066294805	0.011984315	6.796409e-03	0.00780
0.0031079972	0.0030221148	0.003645852	-2.693377e-05	0.00906
-0.0005855398	-0.0008078403	-0.003364929	1.210440e-02	-0.0003
-0.0056952663	-0.0057160075	0.005992905	-5.253123e-02	0.00636

2 b)

[9]:
```
data_t = data_t[-c(734), ]
```

[10]:
```
any(is.na(data_t))
```

TRUE

[11]:
```
data_s = scale(data_t, center = TRUE, scale = TRUE)
```

[12]:
```
pc_all = prcomp(na.omit(data_s),
                center=FALSE,
                scale.=FALSE,
                rank. = 3)
```

[13]:
```
C = pc_all$x
head(C)
```

A matrix: 6×3 of type dbl

PC1	PC2	PC3
0.5788056	0.48090277	-2.6855674
1.4607900	0.65516037	-2.6252382
1.8365851	0.35968506	-1.2418608
3.0009912	0.04016417	-0.0944485
1.7200672	0.84806899	-1.9390267
2.5650027	1.12500815	0.7164771

[14]:
```
summary(pc_all)
```

```
Importance of first k=3 (out of 128) components:
                          PC1    PC2     PC3
Standard deviation     3.8747 3.7527 2.91243
Proportion of Variance 0.1277 0.1198 0.07216
Cumulative Proportion  0.1277 0.2475 0.31970
```

[15]:
```
RPI = data_s[(nrow(data_s)+1 - nrow(C)):nrow(data_s), 1]
UNRATE = data_s[(nrow(data_s)+1 - nrow(C)):nrow(data_s), 24]
```

[16]:
```
reg1 = lm(RPI ~ C)
reg2 = lm(UNRATE ~ C)
```

[17]:
```
summary(reg1)
summary(reg2)
```

```
Call:
lm(formula = RPI ~ C)

Residuals:
    Min      1Q  Median      3Q     Max
-9.4706 -0.3550  0.0228  0.3691  7.1256

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) -0.03097    0.07194  -0.430   0.6671
CPC1        -0.03825    0.01818  -2.104   0.0361 *
CPC2        -0.03643    0.01798  -2.026   0.0435 *
CPC3        -0.03937    0.02335  -1.686   0.0927 .
---
```

```
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 1.222 on 331 degrees of freedom
Multiple R-squared:  0.03392,Adjusted R-squared:  0.02516
F-statistic: 3.873 on 3 and 331 DF,  p-value: 0.009579



Call:
lm(formula = UNRATE ~ C)

Residuals:
    Min      1Q  Median      3Q     Max
-2.4561 -0.4998 -0.0329  0.4582  4.8416

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) -0.06611    0.04663  -1.418  0.15717
CPC1         0.05608    0.01179   4.759 2.92e-06 ***
CPC2         0.03422    0.01165   2.937  0.00355 **
CPC3         0.10949    0.01513   7.234 3.28e-12 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.7924 on 331 degrees of freedom
Multiple R-squared:  0.2092,Adjusted R-squared:  0.202
F-statistic: 29.19 on 3 and 331 DF,  p-value: < 2.2e-16
```

The three factors explain approximately 19% of UNRATE, but only 4% of RPI.

2 c)

[18]:
```
# We follow BBE(2005) to define the sets of slow and fast variables
names = colnames(data_s)
```

[19]:
```
names
```

1. 'RPI' 2. 'W875RX1' 3. 'DPCERA3M086SBEA' 4. 'CMRMTSPLx' 5. 'RETAILx' 6. 'INDPRO' 7. 'IPFPNSS' 8. 'IPFINAL' 9. 'IPCONGD' 10. 'IPDCONGD' 11. 'IPNCONGD' 12. 'IPBUSEQ' 13. 'IPMAT' 14. 'IPDMAT' 15. 'IPNMAT' 16. 'IPMANSICS' 17. 'IPB51222S' 18. 'IPFUELS' 19. 'CUMFNS' 20. 'HWI' 21. 'HWIURATIO' 22. 'CLF16OV' 23. 'CE16OV' 24. 'UNRATE' 25. 'UEMPMEAN' 26. 'UEMPLT5' 27. 'UEMP5TO14' 28. 'UEMP15OV' 29. 'UEMP15T26' 30. 'UEMP27OV' 31. 'CLAIMSx' 32. 'PAYEMS' 33. 'USGOOD' 34. 'CES1021000001' 35. 'USCONS' 36. 'MANEMP' 37. 'DMANEMP' 38. 'NDMANEMP' 39. 'SRVPRD' 40. 'USTPU' 41. 'USWTRADE' 42. 'USTRADE' 43. 'USFIRE' 44. 'USGOVT' 45. 'CES0600000007' 46. 'AWOTMAN' 47. 'AWHMAN' 48. 'HOUST' 49. 'HOUSTNE' 50. 'HOUSTMW' 51. 'HOUSTS' 52. 'HOUSTW' 53. 'PERMIT' 54. 'PERMITNE' 55. 'PER-

MITMW' 56. 'PERMITS' 57. 'PERMITW' 58. 'ACOGNO' 59. 'AMDMNOx' 60. 'ANDENOx' 61. 'AMDMUOx' 62. 'BUSINVx' 63. 'ISRATIOx' 64. 'M1SL' 65. 'M2SL' 66. 'M2REAL' 67. 'BOGMBASE' 68. 'TOTRESNS' 69. 'NONBORRES' 70. 'BUSLOANS' 71. 'REALLN' 72. 'NONREVSL' 73. 'CONSPI' 74. 'S.P.500' 75. 'S.P..indust' 76. 'S.P.div.yield' 77. 'S.P.PE.ratio' 78. 'FEDFUNDS' 79. 'CP3Mx' 80. 'TB3MS' 81. 'TB6MS' 82. 'GS1' 83. 'GS5' 84. 'GS10' 85. 'AAA' 86. 'BAA' 87. 'COMPAPFFx' 88. 'TB3SMFFM' 89. 'TB6SMFFM' 90. 'T1YFFM' 91. 'T5YFFM' 92. 'T10YFFM' 93. 'AAAFFM' 94. 'BAAFFM' 95. 'TWEXAFEGSMTHx' 96. 'EXSZUSx' 97. 'EXJPUSx' 98. 'EXUSUKx' 99. 'EXCAUSx' 100. 'WPSFD49207' 101. 'WPSFD49502' 102. 'WPSID61' 103. 'WPSID62' 104. 'OILPRICEx' 105. 'PPICMM' 106. 'CPIAUCSL' 107. 'CPIAPPSL' 108. 'CPITRNSL' 109. 'CPIMEDSL' 110. 'CUSR0000SAC' 111. 'CUSR0000SAD' 112. 'CUSR0000SAS' 113. 'CPIULFSL' 114. 'CUSR0000SA0L2' 115. 'CUSR0000SA0L5' 116. 'PCEPI' 117. 'DDURRG3M086SBEA' 118. 'DNDGRG3M086SBEA' 119. 'DSERRG3M086SBEA' 120. 'CES0600000008' 121. 'CES2000000008' 122. 'CES3000000008' 123. 'UMCSENTx' 124. 'MZMSL' 125. 'DTCOLNVHFNM' 126. 'DTCTHFNM' 127. 'INVEST' 128. 'VXOCLSx'

[20]:
```
fast = c("HOUST", "HOUSTNE", "HOUSTMW", "HOUSTS", "HOUSTW", "PERMIT",␣
↪"PERMITNE", "PERMITMW",
         "PERMITS", "PERMITW", "CMRMTSPLx", "RETAILx", "ACOGNO", "AMDMNOx",␣
↪"ANDENOx", "AMDMUOx",
         "BUSINVx", "ISRATIOx", "UMCSENTx", "M1SL", "M2SL", "M2REAL",␣
↪"BOGMBASE", "TOTRESNS",
         "NONBORRES", "BUSLOANS", "REALLN", "NONREVSL", "CONSPI", "S.P.500", "S.
↪P..indust",
         "S.P.div.yield", "S.P.PE.ratio", "FEDFUNDS", "CP3Mx", "TB3MS",␣
↪"TB6MS", "GS1",
         "GS5", "GS10", "AAA", "BAA", "COMPAPFFx", "TB3SMFFM", "TB6SMFFM",␣
↪"T1YFFM", "T5YFFM",
         "T10YFFM", "AAAFFM", "BAAFFM", "TWEXAFEGSMTHx", "EXSZUSx", "EXJPUSx",␣
↪"EXUSUKx", "EXCAUSx",
         "VXOCLSx")
```

[21]:
```
data_s = na.omit(data_s)
```

[22]:
```
data_s = as.data.frame(data_s)
```

[23]:
```
slow = rep(1,128)
j = 1
for(i in names){
    if(i %in% fast)
        slow[j]=0
        j = j+1
}
```

[24]:
```
slow
```

1. 1 2. 1 3. 1 4. 0 5. 0 6. 1 7. 1 8. 1 9. 1 10. 1 11. 1 12. 1 13. 1 14. 1 15. 1 16. 1 17. 1 18. 1 19. 1

20. 1 21. 1 22. 1 23. 1 24. 1 25. 1 26. 1 27. 1 28. 1 29. 1 30. 1 31. 1 32. 1 33. 1 34. 1 35. 1 36. 1 37. 1 38. 1 39. 1 40. 1 41. 1 42. 1 43. 1 44. 1 45. 1 46. 1 47. 1 48. 0 49. 0 50. 0 51. 0 52. 0 53. 0 54. 0 55. 0 56. 0 57. 0 58. 0 59. 0 60. 0 61. 0 62. 0 63. 0 64. 0 65. 0 66. 0 67. 0 68. 0 69. 0 70. 0 71. 0 72. 0 73. 0 74. 0 75. 0 76. 0 77. 0 78. 0 79. 0 80. 0 81. 0 82. 0 83. 0 84. 0 85. 0 86. 0 87. 0 88. 0 89. 0 90. 0 91. 0 92. 0 93. 0 94. 0 95. 0 96. 0 97. 0 98. 0 99. 0 100. 1 101. 1 102. 1 103. 1 104. 1 105. 1 106. 1 107. 1 108. 1 109. 1 110. 1 111. 1 112. 1 113. 1 114. 1 115. 1 116. 1 117. 1 118. 1 119. 1 120. 1 121. 1 122. 1 123. 0 124. 1 125. 1 126. 1 127. 1 128. 0

[25]:
```
data_slow = data_s[, slow == 1]
```

[26]:
```
pc_slow = prcomp(data_slow, center=FALSE, scale.=FALSE, rank. = 3)
F_slow = pc_slow$x
```

[27]:
```
# Next clean the PC of all space from the observed Y
reg = lm(C ~ F_slow + data_s[,"FEDFUNDS"])
#summary(reg)
F_hat = C - data.matrix(data_s[,"FEDFUNDS"])%*%reg$coefficients[5,] # cleaning␣
↪and saving F_hat
```

[28]:
```
data_var = data.frame(F_hat, "FYFF" = data_s[,"FEDFUNDS"])
var = VAR(data_var, p = 13)
#summary(var)

irf_point = irf(var, n.ahead = 60, impulse = "FYFF", response = "FYFF", boot =␣
↪FALSE)

# Shock size of 25 basis points
impulse_sd = 0.25/sd(as.data.frame(data_t)$FEDFUNDS)
scale = impulse_sd/(irf_point$irf$FYFF[1]) # position of FYFF response at step 0


# Computing Loading Factors
reg_loadings = lm(ts(data_s) ~ F_hat + data_s[,"FEDFUNDS"])
loadings = reg_loadings$coefficients
# head(reg_loadings$coefficients)
#summary(reg_loadings)


#### BOOTSTRAPING ########

R = 500 # Number of simulations
nvars = 128 # Number of variables
nsteps = 61 # numbers of steps

IRFs = array(c(0,0,0), dim = c(nsteps,nvars,R))

var = lineVar(data_var, lag = 13, include = "const")
```

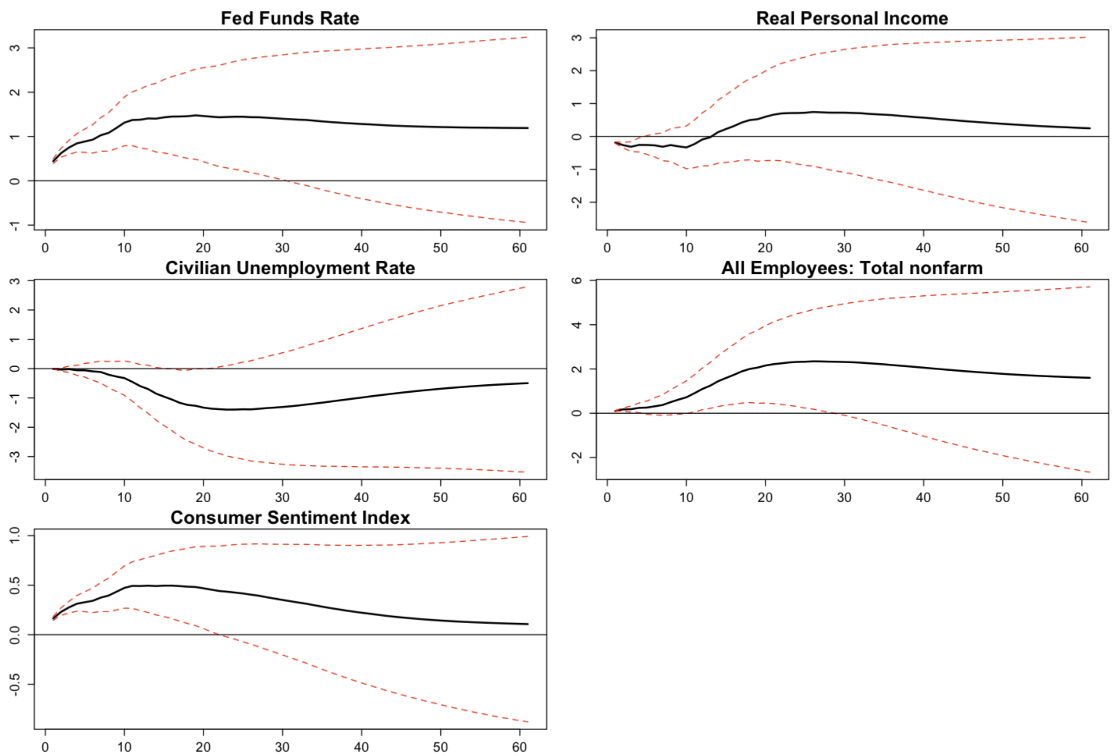
```
for(j in 1:R){
    data_boot = VAR.boot(var, boot.scheme ="resample")
    var_boot = VAR(data_boot, lag = 13)
    irf1 = irf(var_boot, n.ahead = 60, impulse = "FYFF", boot = FALSE)
    for(i in 1:nvars){
        IRFs[,i,j] = (irf1$irf$FYFF %*% matrix(loadings[2:5, i]))*scale
        }
} ## Boot simulations done

# Extract the quantiles of IRFs we are interested: 90% confidence intervals in␣
↪BBE
Upper = array(c(0,0), dim = c(nsteps, nvars))
for(k in 1:nsteps){
    for(i in 1:nvars){
        Upper[k,i] = quantile(IRFs[k,i,], probs = c(0.95))[1]
        }
}
Lower = array(c(0,0), dim = c(nsteps, nvars))
for(k in 1:nsteps){
    for(i in 1:nvars){
        Lower[k,i] = quantile(IRFs[k,i,], probs = c(0.05))[1]
        }
}
IRF = array(c(0,0), dim = c(nsteps, nvars))
for(k in 1:nsteps){
    for(i in 1:nvars){
        IRF[k,i] = quantile(IRFs[k,i,], probs = c(0.5))[1]
        }
}
rm(var_boot)
rm(IRFs)
```

[29]:

```
# Select the Variables you are Interested in
# List of variables we are interested: FYFF, IP, CPI
variables = c(grep("^FEDFUNDS$", colnames(data_s)), grep("^RPI$",␣
↪colnames(data_s)), grep("^UNRATE$", colnames(data_s)),
            grep("^PAYEMS$", colnames(data_s)), grep("^UMCSENTx$",␣
↪colnames(data_s))
             )

transf_code = c(2, 5, 2,
             5, 2
               )

variable_names = c("Fed Funds Rate", "Real Personal Income", "Civilian␣
↪Unemployment Rate",
                "All Employees: Total nonfarm", "Consumer Sentiment Index"
```

```
                )
```

[30]:
```
# 3 Factors and Y = FEDFUNDS
# Change plot size to 15 x 10
options(repr.plot.width=12, repr.plot.height=8)

par(mfrow=c(3,2),
   mar = c(2, 2, 2, 2))

for(i in variables){
    index = which(variables == i)
    if(transf_code[index] == 2 | transf_code[index] == 5){
        plot(cumsum(IRF[,i]), type ='l',lwd=2, main = variable_names[index],
             ylab= "", xlab="Steps",␣
↪ylim=range(cumsum(Lower[,i]),cumsum(Upper[,i])),
             cex.main=1.8, cex.axis=1.3)
        lines(cumsum(Upper[,i]), lty=2, col="red")
        lines(cumsum(Lower[,i]), lty=2, col="red")
        abline(h=0)
    }
    else{
        plot(IRF[,i], type ='l',lwd=2, main = variable_names[index],
             ylab= "", xlab="Steps", ylim=range((Lower[,i]),(Upper[,i])),
             cex.main=1.8, cex.axis=1.3)
        lines((Upper[,i]), lty=2, col="red")
        lines((Lower[,i]), lty=2, col="red")
        abline(h=0)
    }
}
```



The identification of monetary policy shocks seems to be off, since we find that consumer sentiment improves and that employment increases after a monetary contraction. This may be due to the zero lower bound period or the great recession, or because we are including only 3 factors, which explain only 30% of total variation.

2 d)

[31]:
```
library(forecast)
```

[32]:
```
# First we use the FAVAR to forecast the factors
favar_p = predict(var,n.ahead = 1)
favar_p
```

A matrix: 1 × 4 of type dbl

	PC1	PC2	PC3	FYFF
336	-0.5505035	-0.04017985	-0.3932873	-0.1794336

[33]:
```
variables
```

1. 78 2. 1 3. 24 4. 32 5. 123

[34]:
```
# Now we use the factor loadings to make prediction about the variables of␣
↪interest:
rpi_load = loadings[,1]
unrate_load = loadings[,24]
```

[35]:
```
rpi_load
```

(Intercept) -0.00525421594190863 **F_hatPC1** -0.0999186286204694 **F_hatPC2** 0.0253649051573023 **F_hatPC3** -0.0885568359557552 **data_s{[}, "FEDFUNDS"{]}** -0.419064501794218

[36]:
```
rpi_march = rpi_load[1]+ rpi_load[2:5]%*%favar_p[1,]
unrate_march = unrate_load[1]+ unrate_load[2:5]%*%favar_p[1,]
```

[37]:
```
# transform back to original units
unrate_march = unrate_march*sqrt(var(data_t[,24])) + mean(data_t[,24])
```

[39]:
```
print(unrate_march)
```

```
            [,1]
[1,] -0.03489754
```

The FAVAR model predicts a fall of 0.03 p.p. in march. This very off the mark as expected given the rare event of COVID-19. According to BLS data and our own dataset, unemployment jumped 0.9 p.p. in March.

1.0.3 Exercise 3)

[54]:
```
library(quantmod)
getSymbols("DJIA", src="FRED")
```

```
Loading required package: xts
Loading required package: TTR
Version 0.4-0 included new data defaults. See ?getSymbols.
'getSymbols' currently uses auto.assign=TRUE by default, but will
use auto.assign=FALSE in 0.5-0. You will still be able to use
'loadSymbols' to automatically load data. getOption("getSymbols.env")
and getOption("getSymbols.auto.assign") will still be checked for
alternate defaults.

This message is shown once per session and may be disabled by setting
options("getSymbols.warning4.0"=FALSE). See ?getSymbols for details.
```

'DJIA'

[55]:
```
plot(DJIA)
```

[56]:
```
DJIA = window(DJIA, start = "2010-05-28", end = "2020-05-25")
```

[57]:
```
tail(DJIA)
```

```
                DJIA
2020-05-18 24597.37
2020-05-19 24206.86
2020-05-20 24575.90
2020-05-21 24474.12
2020-05-22 24465.16
2020-05-25       NA
```

3 a)

[58]:
```
library(tseries)

dow = DJIA[!is.na(DJIA)]

adf.test(dow)
```

```
Augmented Dickey-Fuller Test

data:  dow
```

```
Dickey-Fuller = -3.7723, Lag order = 13, p-value = 0.02036
alternative hypothesis: stationary
```

We reject nonstationarity. The reason being that the test includes a drift and a constant. Since the COVID period makes the stock prices fall, a test will trend will get us to the conclusion that the series is stationary around a trend. We are faced with a couple of choices. We can difference the data anyways and work with stock returns for the COVID period, we can exclude the COVID period or we work with the data in levels but have to include a time trend.

3 b)

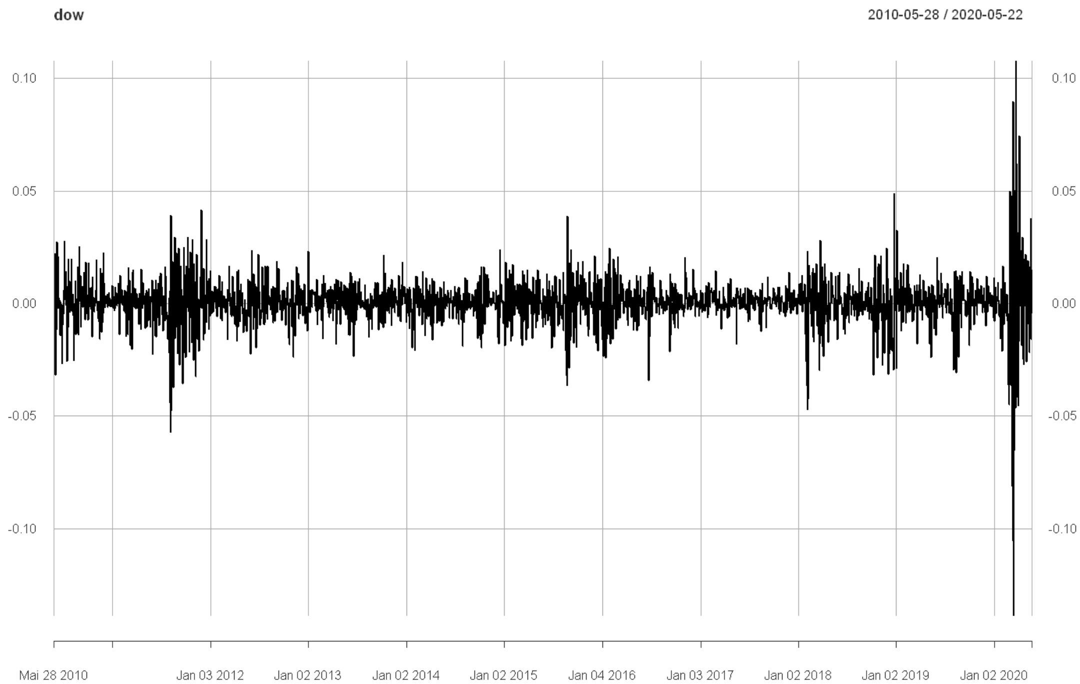
[59]:
```
# Here, we will work with stock returns.

dow = diff(log(dow))
```

[60]:
```
adf.test(na.omit(dow))
```

```
Warning message in adf.test(na.omit(dow)):
"p-value smaller than printed p-value"
```

```
Augmented Dickey-Fuller Test

data:  na.omit(dow)
Dickey-Fuller = -13.595, Lag order = 13, p-value = 0.01
alternative hypothesis: stationary
```

[61]:
```
plot(dow)
```



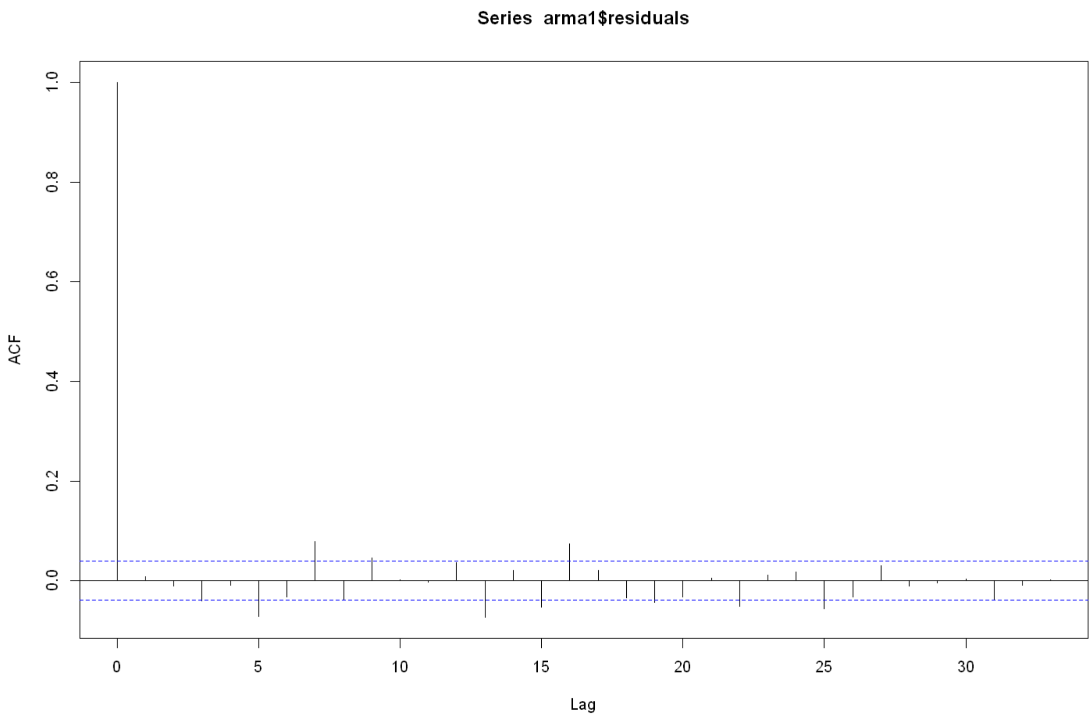
We are left with a series that is stationary for the mean but that clealy exhibits non constant variance. Notice that even if we do not include the COVID period, that data would still display suggestive evidence of GARCH effects.

[62]:
```
library(forecast)
arma1 = auto.arima(dow)
summary(arma1)
```

```
Series: dow
ARIMA(3,0,1) with non-zero mean

Coefficients:
          ar1      ar2     ar3     ma1   mean
      -0.9939  -0.0118  0.1542  0.8588  4e-04
s.e.   0.0344   0.0285  0.0200  0.0293  2e-04

sigma^2 estimated as 0.0001127:  log likelihood=7852.75
AIC=-15693.5   AICc=-15693.47   BIC=-15658.53

Training set error measures:
                       ME       RMSE         MAE MPE MAPE      MASE        ACF1
Training set 5.797915e-08 0.01060628 0.006735374 NaN  Inf 0.6740577 0.008251366
```

[63]:
```
# Check the residuals visually
acf(arma1$residuals)

# Check it statistically, as well
Box.test(arma1$residuals, type = "Ljung-Box")
```

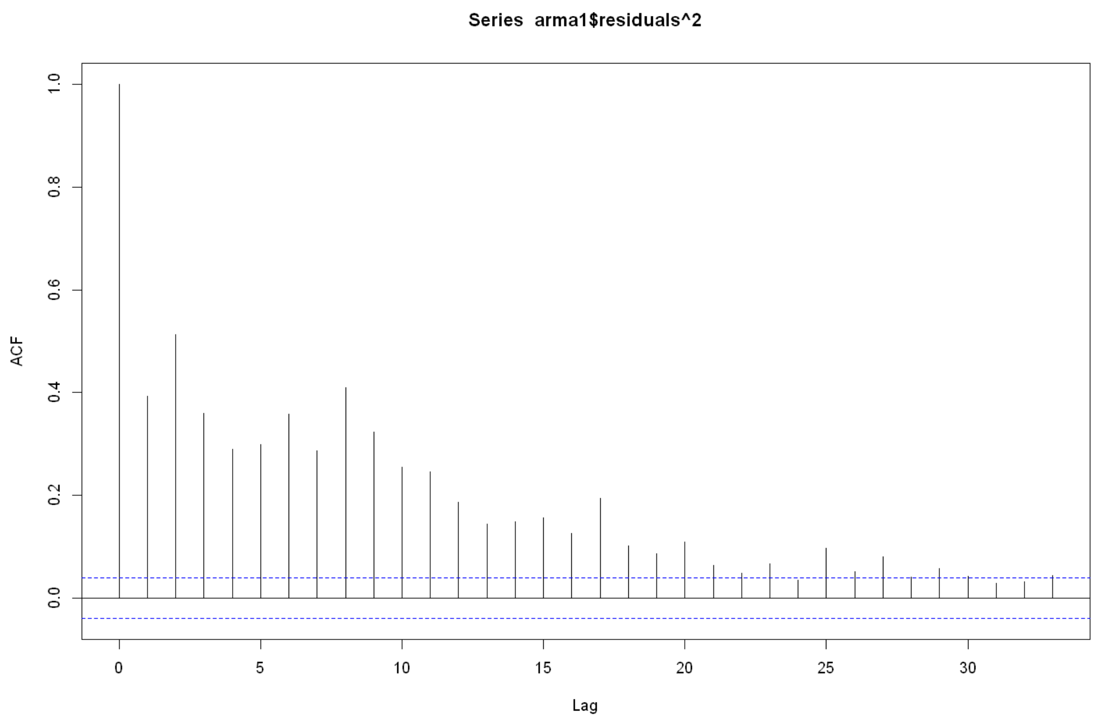
```
Box-Ljung test

data:  arma1$residuals
X-squared = 0.17117, df = 1, p-value = 0.6791
```



We do not find statistically significant autocorrelation, therefore we consider the model to be valid

[64]:
```
# Check the squared residuals
acf(arma1$residuals^2)     # Generate the squared residuals and check them for␣
↪autocorrelation

# Checking statistically for autocorrelation
Box.test(arma1$residuals^2,lag=10) #Testing for this statistically we find␣
↪significant evidence
#Changing the number of lags does not change the results here
```

```
Box-Pierce test

data:  arma1$residuals^2
X-squared = 3186.4, df = 10, p-value < 2.2e-16
```



The regression estimates show us that we find significant autocorrelation for the squared residuals. Consequently, we can conclude that the process has conditional heteroscedastic variance.

3 c)

[65]:
```
library(fGarch)
```

```
Warning message:
"package 'fGarch' was built under R version 3.6.3"Loading required package:
timeDate
Loading required package: timeSeries
Warning message:
"package 'timeSeries' was built under R version 3.6.3"
Attaching package: 'timeSeries'

The following object is masked from 'package:zoo':
```

```
    time<-

Loading required package: fBasics
Warning message:
"package 'fBasics' was built under R version 3.6.3"
Attaching package: 'fBasics'

The following object is masked from 'package:TTR':

    volatility
```

[66]:
```
# Estimate the ARMA-GARCH model together:
garch= garchFit(~arma(3,1)+garch(1,1), data=dow[-1])
```

```
Series Initialization:
 ARMA Model:                arma
 Formula Mean:              ~ arma(3, 1)
 GARCH Model:               garch
 Formula Variance:          ~ garch(1, 1)
 ARMA Order:                3 1
 Max ARMA Order:            3
 GARCH Order:               1 1
 Max GARCH Order:           1
 Maximum Order:             3
 Conditional Dist:          norm
 h.start:                   4
 llh.start:                 1
 Length of Series:          2511
 Recursion Init:            mci
 Series Scale:              0.01089823

Parameter Initialization:
 Initial Parameters:          $params
 Limits of Transformations:   $U, $V
 Which Parameters are Fixed?  $includes
 Parameter Matrix:
                     U           V      params includes
    mu     -0.32197321   0.3219732  0.03237528     TRUE
    ar1    -0.99999999   1.0000000 -0.99766042     TRUE
    ar2    -0.99999999   1.0000000 -0.01279596     TRUE
    ar3    -0.99999999   1.0000000  0.15385223     TRUE
    ma1    -0.99999999   1.0000000  0.86115518     TRUE
    omega   0.00000100 100.0000000  0.10000000     TRUE
    alpha1  0.00000001   1.0000000  0.10000000     TRUE
    gamma1 -0.99999999   1.0000000  0.10000000    FALSE
    beta1   0.00000001   1.0000000  0.80000000     TRUE
```

```
    delta   0.00000000   2.0000000  2.00000000    FALSE
    skew    0.10000000  10.0000000  1.00000000    FALSE
    shape   1.00000000  10.0000000  4.00000000    FALSE
 Index List of Parameters to be Optimized:
    mu    ar1    ar2    ar3    ma1  omega alpha1  beta1
     1      2      3      4      5      6      7      9
 Persistence:                  0.9


--- START OF TRACE ---
Selected Algorithm: nlminb

R coded nlminb Solver:

  0:     2961.9323: 0.0323753 -0.997660 -0.0127960 0.153852 0.861155 0.100000
0.100000 0.800000
  1:     2875.7593: 0.0323762 -0.997266 -0.0110658 0.150027 0.861149 0.0707961
0.0967667 0.782741
  2:     2845.1341: 0.0323787 -0.996005 -0.00697141 0.140769 0.861541 0.0427406
0.113569 0.780433
  3:     2830.7764: 0.0323834 -0.993012 -0.00227658 0.128793 0.863721 0.0551888
0.141679 0.787825
  4:     2802.5382: 0.0323946 -0.980764 -2.34655e-05 0.112398 0.875470 0.0365441
0.154067 0.777063
  5:     2793.7149: 0.0324158 -0.970699 0.00936118 0.0839928 0.884279 0.0395198
0.157874 0.786084
  6:     2791.1323: 0.0324411 -0.949301 0.00197953 0.0759122 0.904104 0.0291688
0.158110 0.796108
  7:     2785.4937: 0.0324522 -0.953598 0.00926653 0.0650971 0.896625 0.0315812
0.159325 0.800961
  8:     2784.3567: 0.0324528 -0.953401 0.00933538 0.0647242 0.896785 0.0289926
0.158961 0.800080
  9:     2783.8937: 0.0324610 -0.951159 0.0102563 0.0602908 0.898597 0.0263849
0.161568 0.800847
 10:     2782.7851: 0.0324977 -0.944878 0.0107770 0.0506929 0.902097 0.0301398
0.160860 0.797348
 11:     2782.6174: 0.0325360 -0.941109 0.00931096 0.0476227 0.903215 0.0291904
0.161849 0.794579
 12:     2782.1975: 0.0326310 -0.939427 0.00795546 0.0437703 0.900654 0.0297377
0.162836 0.795973
 13:     2781.9908: 0.0328823 -0.932738 0.00562592 0.0416987 0.896111 0.0273037
0.165430 0.797788
 14:     2781.6825: 0.0331231 -0.925721 0.00821466 0.0380921 0.894385 0.0298285
0.167348 0.794037
 15:     2781.6188: 0.0331952 -0.924461 0.00946867 0.0360759 0.893610 0.0291269
0.166914 0.792996
 16:     2781.5424: 0.0332883 -0.922393 0.00901685 0.0356846 0.892495 0.0296342
0.167037 0.793262
```

17: 2781.5355: 0.0334768 -0.918564 0.00755687 0.0360689 0.890056 0.0288132
0.167756 0.793060
18: 2781.4559: 0.0335743 -0.917273 0.00779098 0.0353901 0.888521 0.0296023
0.167989 0.793507
19: 2781.3939: 0.0336740 -0.915814 0.00798223 0.0347355 0.887074 0.0290606
0.167817 0.793447
20: 2781.3575: 0.0337764 -0.914344 0.00815701 0.0342250 0.885692 0.0293391
0.167984 0.793551
21: 2781.3244: 0.0338803 -0.912897 0.00824046 0.0339654 0.884340 0.0291073
0.168148 0.793233
22: 2780.3182: 0.0392813 -0.841565 0.00974098 0.0294077 0.814018 0.0316696
0.168407 0.787373
23: 2777.7475: 0.0580046 -0.613448 0.0262516 0.0485350 0.568791 0.0258786
0.153743 0.811625
24: 2777.6829: 0.0580054 -0.613125 0.0260161 0.0483847 0.569080 0.0253791
0.153804 0.811205
25: 2777.6221: 0.0580062 -0.612786 0.0257710 0.0482140 0.569383 0.0259188
0.154126 0.811150
26: 2777.5548: 0.0580445 -0.612039 0.0254219 0.0480414 0.569187 0.0254003
0.154316 0.810592
27: 2777.4773: 0.0581347 -0.610854 0.0250258 0.0479348 0.568180 0.0260058
0.154853 0.810396
28: 2777.3810: 0.0583192 -0.608641 0.0243584 0.0478243 0.565929 0.0255486
0.155418 0.809538
29: 2777.2502: 0.0586915 -0.604616 0.0232981 0.0478300 0.560989 0.0261987
0.156639 0.808761
30: 2777.0724: 0.0594396 -0.596844 0.0212103 0.0478525 0.550944 0.0257515
0.158372 0.806908
31: 2776.8426: 0.0609563 -0.582216 0.0165562 0.0471058 0.531328 0.0266525
0.160732 0.805616
32: 2774.8798: 0.0734077 -0.470048 0.0239623 0.0365025 0.445702 0.0245824
0.158297 0.812424
33: 2774.8030: 0.0734081 -0.470105 0.0239047 0.0363673 0.445626 0.0240780
0.158003 0.811711
34: 2774.7440: 0.0734180 -0.470084 0.0238397 0.0362094 0.445440 0.0248619
0.158020 0.811384
35: 2774.6857: 0.0734679 -0.469672 0.0237857 0.0360923 0.444856 0.0245102
0.157897 0.810707
36: 2774.6256: 0.0735753 -0.468716 0.0237291 0.0359852 0.443674 0.0251671
0.158109 0.810324
37: 2774.5515: 0.0737941 -0.466698 0.0236499 0.0359206 0.441349 0.0248661
0.158283 0.809572
38: 2774.4467: 0.0742335 -0.462623 0.0233950 0.0358211 0.436715 0.0254337
0.158901 0.808995
39: 2773.6029: 0.0815351 -0.395161 0.0147523 0.0332777 0.359884 0.0244969
0.164924 0.805824
40: 2773.5692: 0.0815355 -0.395140 0.0147529 0.0329813 0.359888 0.0258757
0.164788 0.805446

```
 41:     2773.4579: 0.0815629 -0.394966 0.0152555 0.0331758 0.359924 0.0252335
0.164765 0.804613
 42:     2773.3981: 0.0816300 -0.394803 0.0162292 0.0332846 0.360194 0.0263824
0.165064 0.803273
 43:     2773.3307: 0.0816668 -0.398020 0.0131585 0.0310340 0.362291 0.0257456
0.163398 0.804099
 44:     2773.2804: 0.0819318 -0.396836 0.0189884 0.0340864 0.363314 0.0265416
0.166710 0.800555
 45:     2773.1152: 0.0820082 -0.403384 0.0136036 0.0285724 0.368059 0.0271070
0.164110 0.800166
 46:     2773.0897: 0.0820104 -0.403174 0.0134452 0.0275217 0.368167 0.0276259
0.165341 0.799542
 47:     2773.0260: 0.0820481 -0.402916 0.0139476 0.0273421 0.368269 0.0271392
0.166117 0.798655
 48:     2773.0008: 0.0820921 -0.402462 0.0143131 0.0271238 0.368130 0.0274251
0.166938 0.798083
 49:     2772.9698: 0.0821917 -0.401318 0.0148815 0.0270158 0.367413 0.0272205
0.167876 0.797182
 50:     2772.9392: 0.0823950 -0.398637 0.0152423 0.0270205 0.365080 0.0275359
0.168696 0.796596
 51:     2772.7199: 0.0863368 -0.348030 0.0141713 0.0283559 0.315471 0.0277788
0.171540 0.793453
 52:     2772.5469: 0.0908483 -0.356774 0.00964283 0.0143879 0.318788 0.0273293
0.180116 0.788827
 53:     2772.4886: 0.0925286 -0.356264 0.00465879 0.0118550 0.315565 0.0279338
0.176936 0.790637
 54:     2772.4119: 0.0958465 -0.411753 0.00715312 0.0145342 0.376029 0.0291095
0.178335 0.787000
 55:     2772.3724: 0.0997258 -0.449559 0.00376307 0.0145760 0.413772 0.0289982
0.176875 0.786758
 56:     2772.3599: 0.102484 -0.488555 0.00261927 0.0160291 0.452375 0.0289867
0.177129 0.786788
 57:     2772.3392: 0.110594 -0.599808 -0.00100739 0.0178454 0.562801 0.0289104
0.177276 0.787062
 58:     2772.3054: 0.126297 -0.816124 -0.00898394 0.0199659 0.778349 0.0288754
0.177563 0.787057
 59:     2772.3030: 0.127648 -0.834882 -0.00971779 0.0201747 0.797105 0.0288382
0.177568 0.787026
 60:     2772.3017: 0.128444 -0.845291 -0.0103007 0.0202038 0.807802 0.0288661
0.177511 0.786954
 61:     2772.3014: 0.129255 -0.855291 -0.0107682 0.0199899 0.817794 0.0288997
0.177510 0.786959
 62:     2772.3009: 0.129292 -0.853452 -0.0109454 0.0194803 0.816115 0.0288773
0.177462 0.786919
 63:     2772.3007: 0.129225 -0.852110 -0.0109808 0.0194244 0.814779 0.0288873
0.177479 0.786925
 64:     2772.3007: 0.129161 -0.850741 -0.0110671 0.0193622 0.813364 0.0288860
0.177486 0.786930
```

```
 65:     2772.3006: 0.129110 -0.849447 -0.0111967 0.0192879 0.811979 0.0288847
0.177489 0.786939
 66:     2772.3006: 0.129116 -0.849498 -0.0112067 0.0192857 0.812025 0.0288842
0.177489 0.786939

Final Estimate of the Negative LLH:
 LLH:  -8575.297    norm LLH:  -3.415092
           mu            ar1            ar2            ar3            ma1
 1.407138e-03 -8.494985e-01 -1.120665e-02  1.928573e-02  8.120247e-01
        omega         alpha1          beta1
 3.430619e-06  1.774886e-01  7.869391e-01

R-optimhess Difference Approximated Hessian Matrix:
                  mu            ar1            ar2            ar3            ma1
mu     -1.699391e+07  -16175.78066 -1.257378e+04  -15582.95464 -3.070235e+03
ar1    -1.617578e+04   -6576.94641  5.535219e+03   -4579.25357 -6.324963e+03
ar2    -1.257378e+04    5535.21897 -6.739160e+03    5639.61363  5.251957e+03
ar3    -1.558295e+04   -4579.25357  5.639614e+03   -6951.66027 -4.306800e+03
ma1    -3.070235e+03   -6324.96320  5.251957e+03   -4306.80033 -6.102680e+03
omega  -6.242467e+08 6787230.22073 -8.732013e+06 4351699.82053  6.820032e+06
alpha1  9.362175e+03      10.54738 -7.837135e+01      82.74352  1.497318e+00
beta1  -1.944328e+04     271.56299 -4.534898e+02     439.17625  2.642325e+02
               omega         alpha1          beta1
mu     -6.242467e+08  9.362175e+03 -1.944328e+04
ar1     6.787230e+06  1.054738e+01  2.715630e+02
ar2    -8.732013e+06 -7.837135e+01 -4.534898e+02
ar3     4.351700e+06  8.274352e+01  4.391763e+02
ma1     6.820032e+06  1.497318e+00  2.642325e+02
omega  -1.552208e+13 -3.728083e+08 -6.332513e+08
alpha1 -3.728083e+08 -1.949364e+04 -2.403237e+04
beta1  -6.332513e+08 -2.403237e+04 -3.609826e+04
attr(,"time")
Time difference of 0.2368159 secs

--- END OF TRACE ---


Time to Estimate Parameters:
 Time difference of 1.004639 secs
```

[67]:
```r
# Look at the results
summary(garch)
```

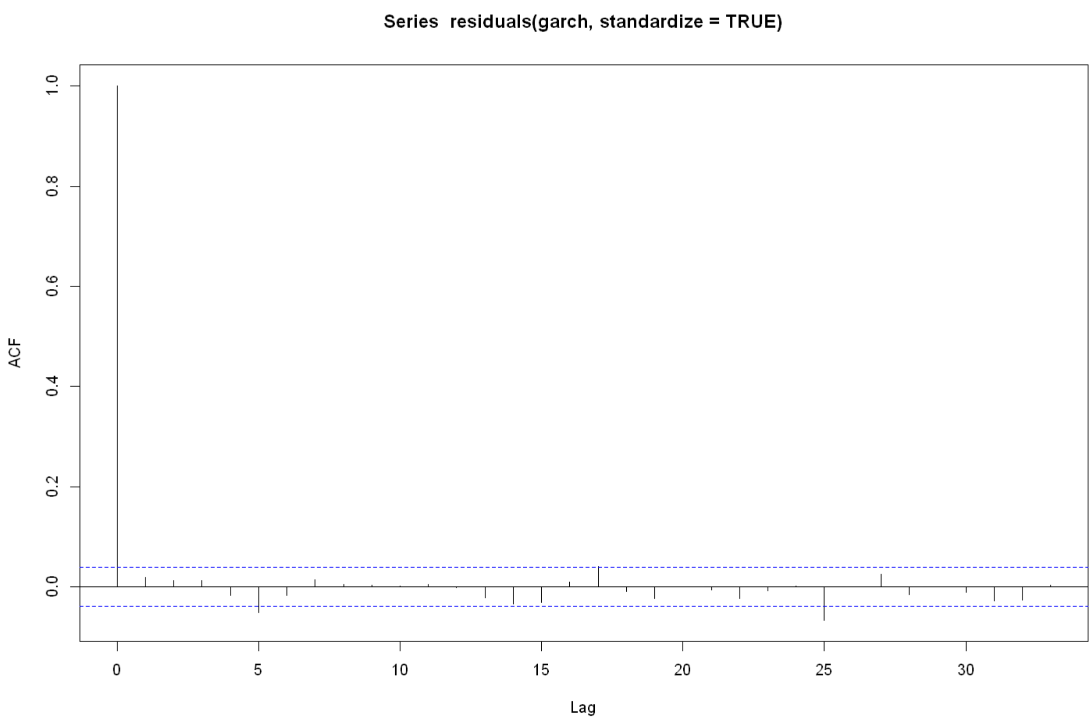
```
Title:
 GARCH Modelling

Call:
```

```
 garchFit(formula = ~arma(3, 1) + garch(1, 1), data = dow[-1])

Mean and Variance Equation:
 data ~ arma(3, 1) + garch(1, 1)
<environment: 0x000000003f610b38>
 [data = dow[-1]]

Conditional Distribution:
 norm

Coefficient(s):
         mu          ar1          ar2          ar3          ma1        omega
 1.4071e-03  -8.4950e-01  -1.1207e-02   1.9286e-02   8.1202e-01   3.4306e-06
     alpha1        beta1
 1.7749e-01   7.8694e-01

Std. Errors:
 based on Hessian

Error Analysis:
         Estimate  Std. Error  t value Pr(>|t|)
mu      1.407e-03   3.948e-04    3.564 0.000365 ***
ar1    -8.495e-01   3.861e-01   -2.200 0.027795 *
ar2    -1.121e-02   3.262e-02   -0.344 0.731181
ar3     1.929e-02   2.156e-02    0.895 0.370984
ma1     8.120e-01   3.852e-01    2.108 0.035020 *
omega   3.431e-06   5.183e-07    6.618 3.63e-11 ***
alpha1  1.775e-01   1.850e-02    9.594  < 2e-16 ***
beta1   7.869e-01   1.873e-02   42.021  < 2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Log Likelihood:
 8575.297    normalized:  3.415092

Description:
 Thu May 28 11:43:40 2020 by user: Fabio


Standardised Residuals Tests:
                                Statistic p-Value
 Jarque-Bera Test   R    Chi^2  442.0722  0
 Shapiro-Wilk Test  R    W      0.9765522 0
 Ljung-Box Test     R    Q(10)  10.3663   0.4089656
 Ljung-Box Test     R    Q(15)  17.14963  0.3099984
 Ljung-Box Test     R    Q(20)  23.16727  0.2806563
 Ljung-Box Test     R^2  Q(10)  7.211495  0.705339
 Ljung-Box Test     R^2  Q(15)  9.317555  0.8603368
```

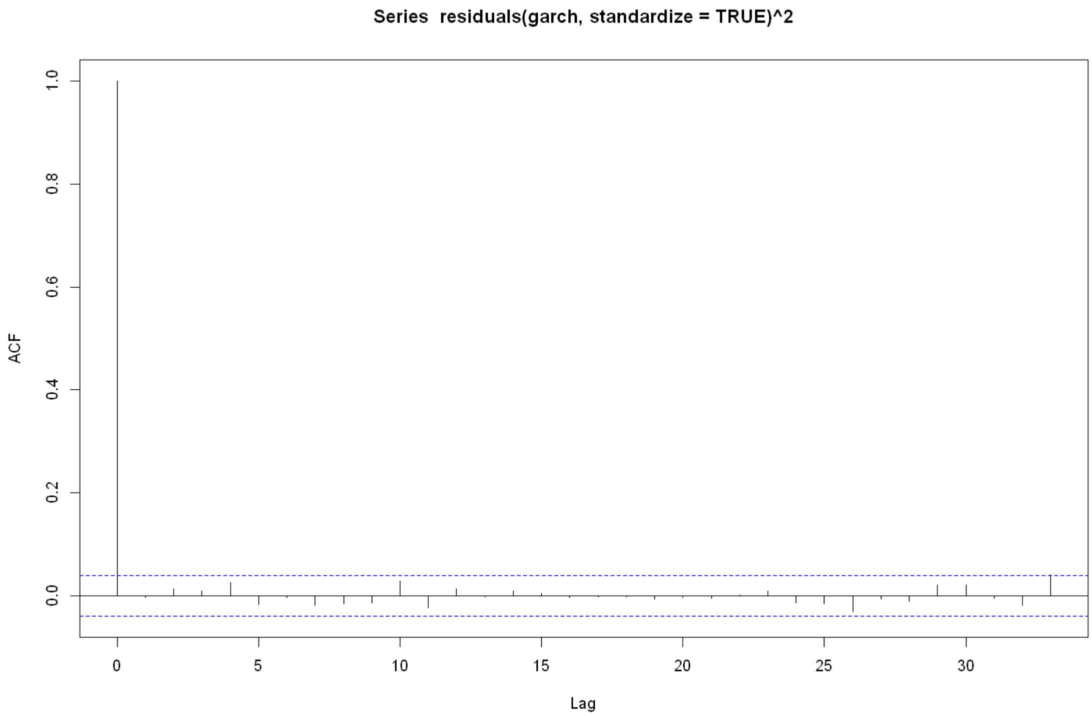
```
 Ljung-Box Test     R^2  Q(20)  9.483282  0.9766037
 LM Arch Test       R    TR^2   8.783067  0.7213393

Information Criterion Statistics:
      AIC       BIC       SIC      HQIC
-6.823813 -6.805243 -6.823833 -6.817073
```

[68]:
```
# Test the standardize residuals
sd(residuals(garch, standardize=TRUE))^2
acf(residuals(garch, standardize=TRUE)) #Looks good, we need to standardize the␣
 ↪errors
acf(residuals(garch, standardize=TRUE)^2) #We have a spike, but we do capture␣
 ↪most of the autocorrelated movement
Box.test(residuals(garch, standardize=TRUE)^2, lag=20) #And it is statistically␣
 ↪not significant
```

0.999865136035962

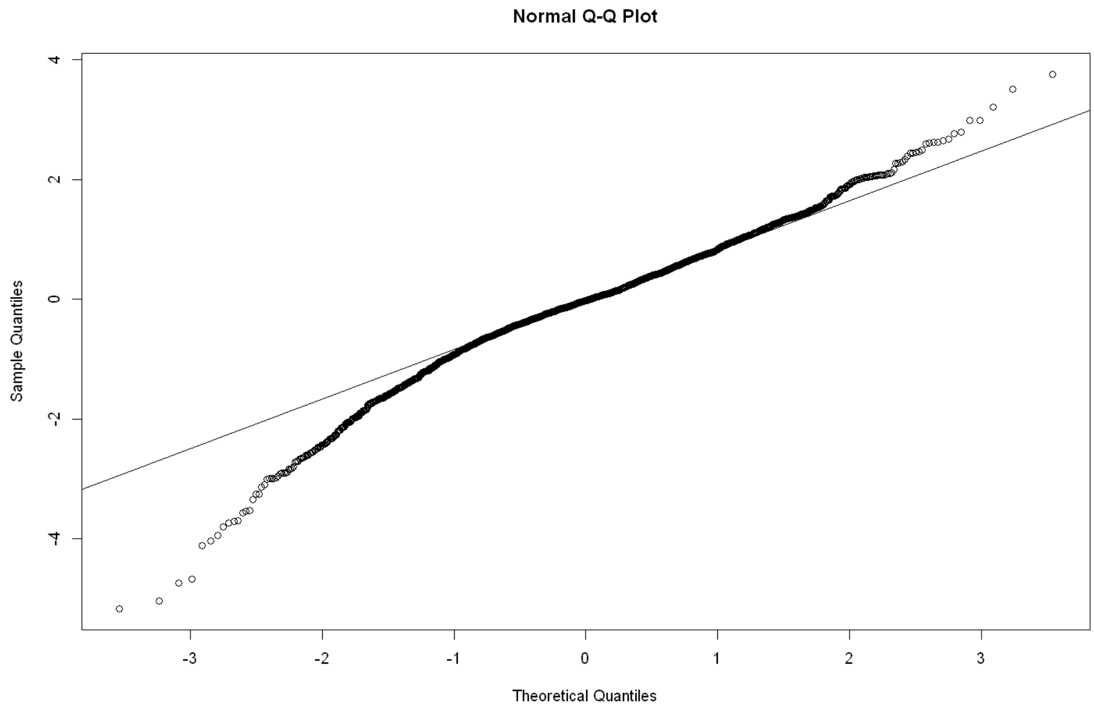



```
Box-Pierce test

data:  residuals(garch, standardize = TRUE)^2
X-squared = 9.4456, df = 20, p-value = 0.9771
```



The residuals do not have any further garch effects. The only issue in terms of model validity is that the residuals appear to be non-normal distributed. Hence, the normality assumption is violated and the approximation of the model estiamted as if the residuals are normal may be far off.

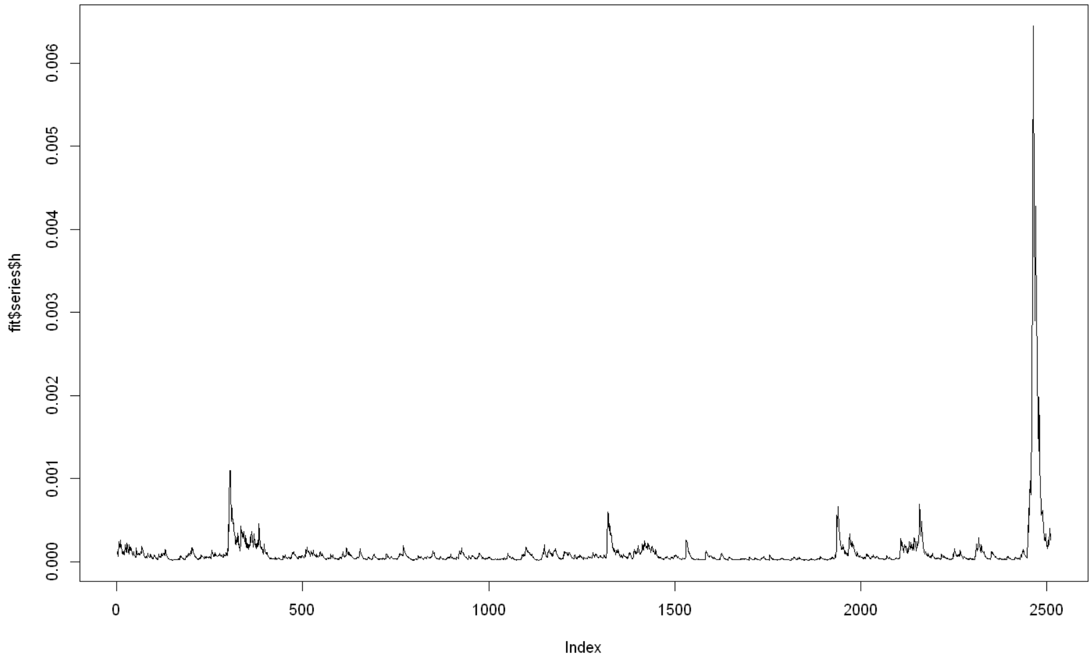
[69]:
```
# Plotting the distribution of the residuals against the theoretical␣
↪distribution
qqnorm(residuals(garch, standardize=TRUE))
qqline(residuals(garch, standardize=TRUE))
```



We tried different specifications of lags for the GARCH model, but the the non-normality did not go away.

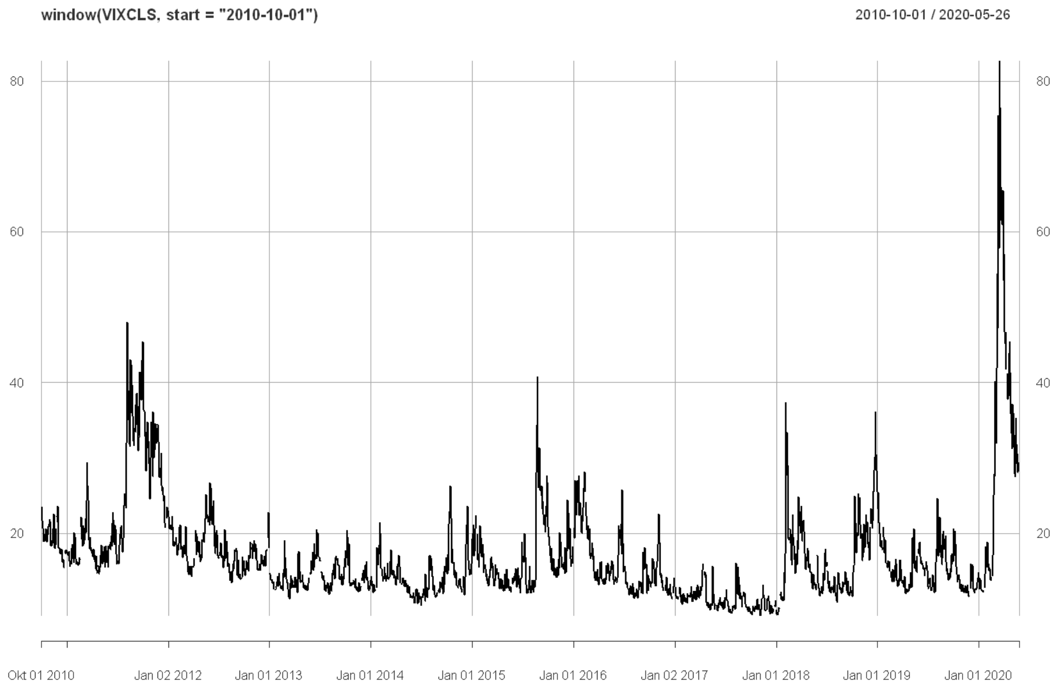
3 d)

[70]:
```
fit = attr(garch, "fit")
plot(fit$series$h, type = "l", col = "black", lty = 1)
```



[71]:
```
getSymbols(Symbols = "VIXCLS", src = "FRED")
plot(window(VIXCLS, start = "2010-10-01"))
```

'VIXCLS'



[72]:
```
x = as.numeric(fit$series$h)
y = as.numeric(VIXCLS[(7928+1-length(x)):7928])
cor(x, y, use="complete.obs")
```

0.645245303803502

[73]:
```
cor.test(x,y)
```

```
Pearson's product-moment correlation

data:  x and y
t = 41.548, df = 2420, p-value < 2.2e-16
alternative hypothesis: true correlation is not equal to 0
95 percent confidence interval:
 0.6213855 0.6679094
sample estimates:
      cor
0.6452453
```

Clearly, there is a strong significant correlation between VIX and the estimated conditional variance

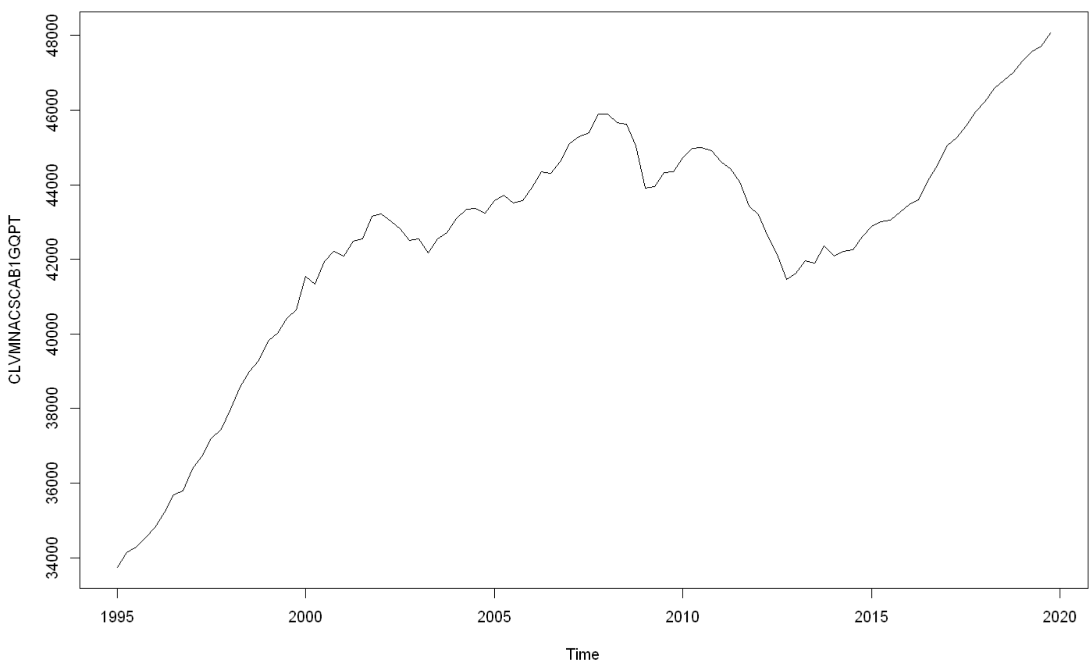
1.0.4 Exercise 4)

[74]:
```
library(quantmod)
getSymbols(Symbols = "CLVMNACSCAB1GQPT", src = "FRED")
pt = CLVMNACSCAB1GQPT
head(pt)
pt = ts(pt, start = 1995, deltat = 1/4)
```

'CLVMNACSCAB1GQPT'

```
           CLVMNACSCAB1GQPT
1995-01-01          33747.2
1995-04-01          34145.4
1995-07-01          34293.6
1995-10-01          34526.0
1996-01-01          34798.6
1996-04-01          35216.0
```

[75]:
```
plot(pt)
```



4 a)

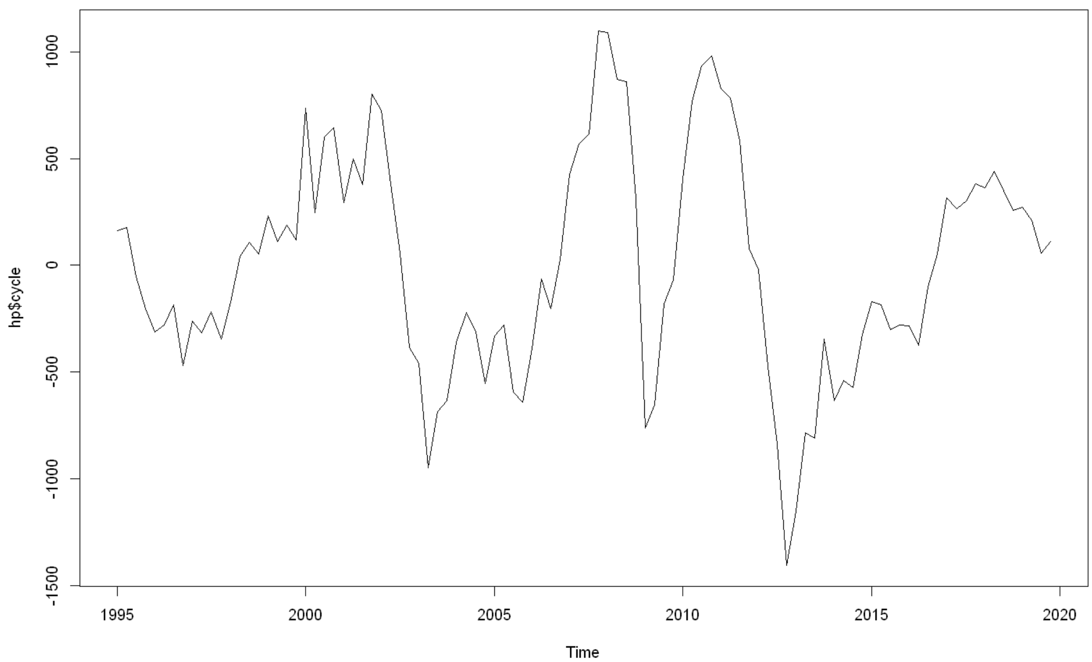
[76]:
```
adf.test(pt)
```

```
Augmented Dickey-Fuller Test

data:  pt
Dickey-Fuller = -2.5132, Lag order = 4, p-value = 0.364
alternative hypothesis: stationary
```

Not stationary.

[77]:
```
library(mFilter)
hp = hpfilter(pt, freq = 1600)
plot(hp$cycle, type = "l")
```

```
Warning message:
"package 'mFilter' was built under R version 3.6.3"
```



[78]:
```
cycle = ts(hp$cycle, start = 1995, deltat = 1/12)
adf.test(cycle)
```

```
Augmented Dickey-Fuller Test

data:  cycle
Dickey-Fuller = -3.7714, Lag order = 4, p-value = 0.02307
alternative hypothesis: stationary
```

Stationary.

4 b)

[79]:
```
library(NTS)
```

```
Warning message:
"package 'NTS' was built under R version 3.6.3"
```

[80]:
```
mod = MSM.fit(cycle, p = 4, nregime = 2)
summary(mod)
```

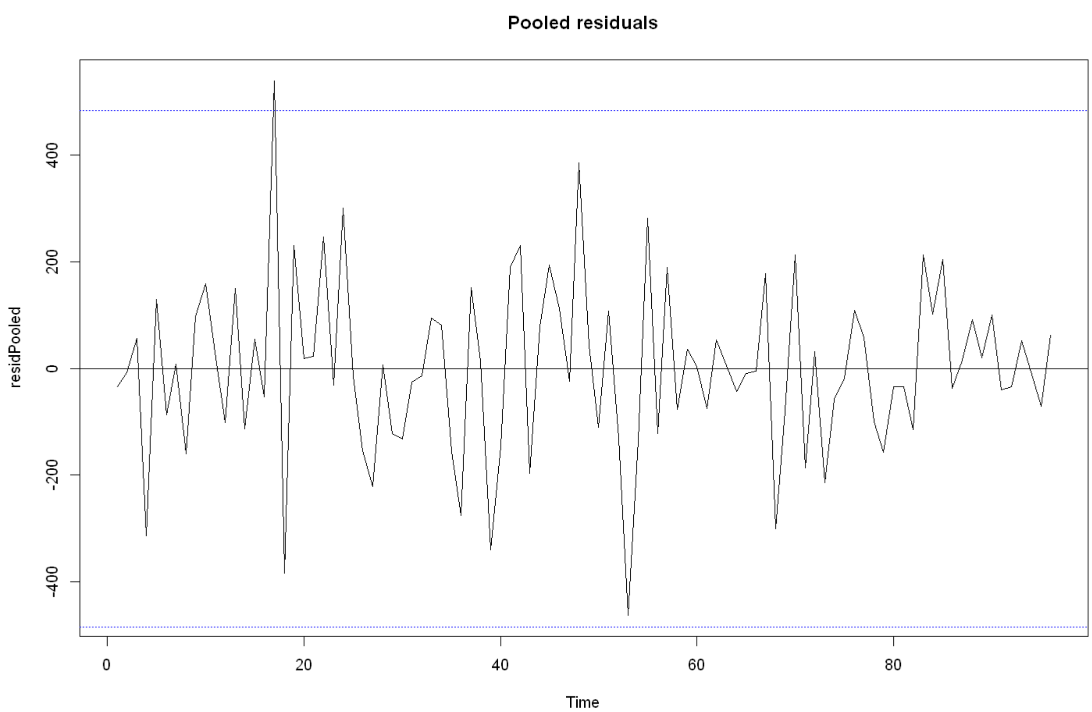
```
Markov Switching Model

Call: msmFit(object = mo, k = nregime, sw = sw)

       AIC      BIC    logLik
  1331.025 1402.312 -655.5125

Coefficients:

Regime 1
---------
         Estimate Std. Error t value  Pr(>|t|)
cnst(S)   49.3178    31.4158  1.5698    0.1165
lag.1(S)   0.7870     0.1156  6.8080 9.897e-12 ***
lag.2(S)   0.1598     0.1422  1.1238    0.2611
lag.3(S)  -0.2589     0.1650 -1.5691    0.1166
lag.4(S)   0.2363     0.1546  1.5285    0.1264
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 178.9584
Multiple R-squared: 0.8623

Standardized Residuals:
       Min          Q1         Med          Q3         Max
-373.92436  -93.38561  -18.91740   92.35258  536.60970

Regime 2
---------
         Estimate Std. Error t value  Pr(>|t|)
cnst(S)  -57.1512    69.8268 -0.8185 0.4130717
lag.1(S)   1.2690     0.3368  3.7678 0.0001647 ***
lag.2(S)  -0.1481     0.3101 -0.4776 0.6329349
```

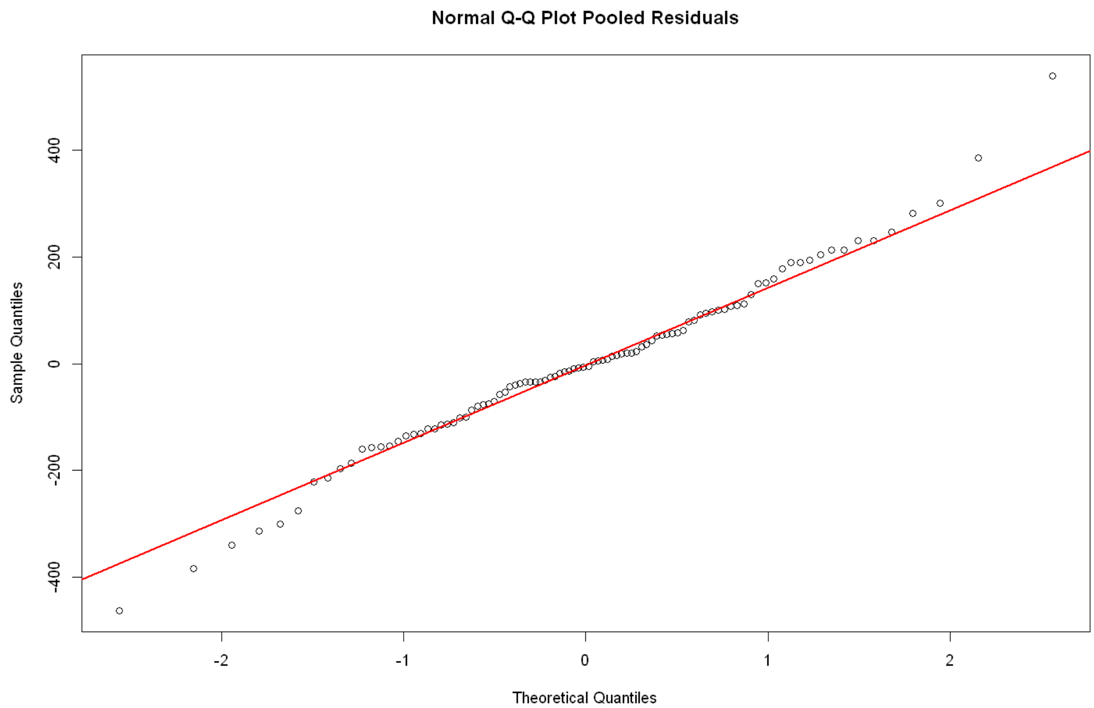
```
lag.3(S)    0.2299      0.2251   1.0213 0.3071123
lag.4(S)   -0.6545      0.1994 -3.2823 0.0010296 **
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 187.648
Multiple R-squared: 0.8939

Standardized Residuals:
       Min          Q1         Med          Q3         Max
-462.45008  -84.72949   11.68128  102.12086  366.45948

Transition probabilities:
          Regime 1  Regime 2
Regime 1 0.6632076 0.4297292
Regime 2 0.3367924 0.5702708
```

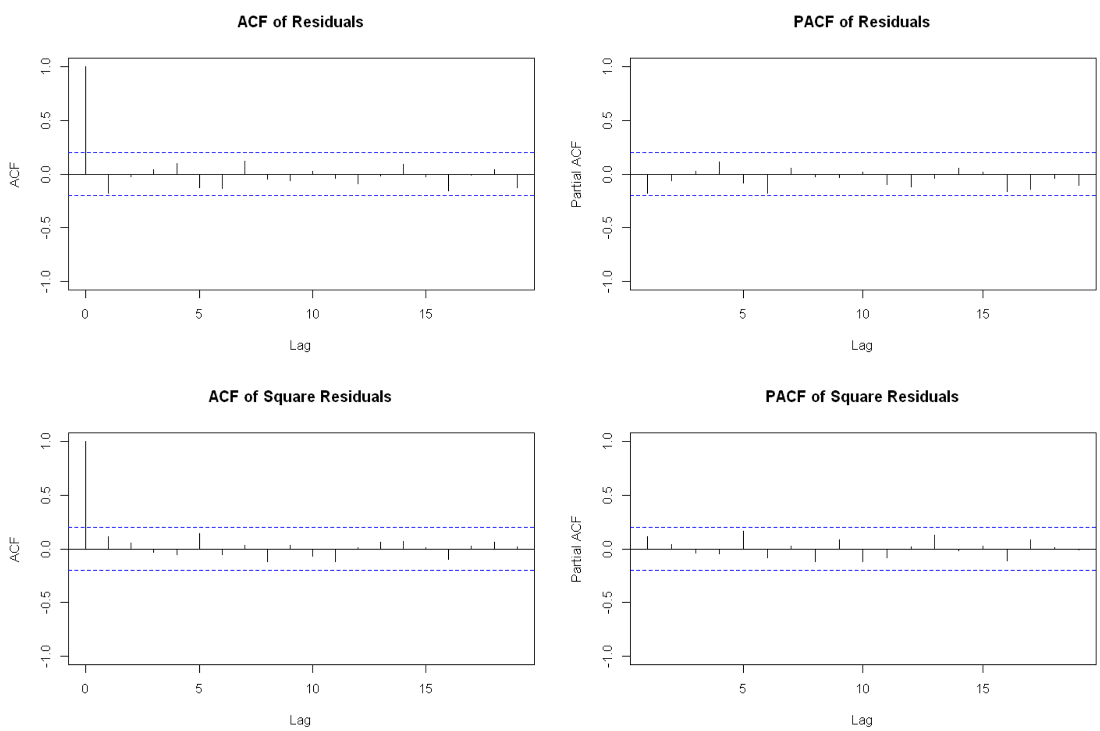
4 c)

[81]:
```
library(MSwM)
plotDiag(mod)
```

```
Warning message:
"package 'MSwM' was built under R version 3.6.3"Loading required package:
parallel
```




Normal Q-Q Plot Pooled Residuals
Sample Quantiles
Theoretical Quantiles
-400
-200
0
200
400
-2
-1
0
1
2


ACF of Residuals
PACF of Residuals
ACF of Square Residuals
PACF of Square Residuals
ACF
Partial ACF
Lag

[82]:
```
resid = msmResid(mod)
adf.test(resid)
Box.test(resid)
shapiro.test(resid)
```

```
Warning message in adf.test(resid):
"p-value smaller than printed p-value"
```

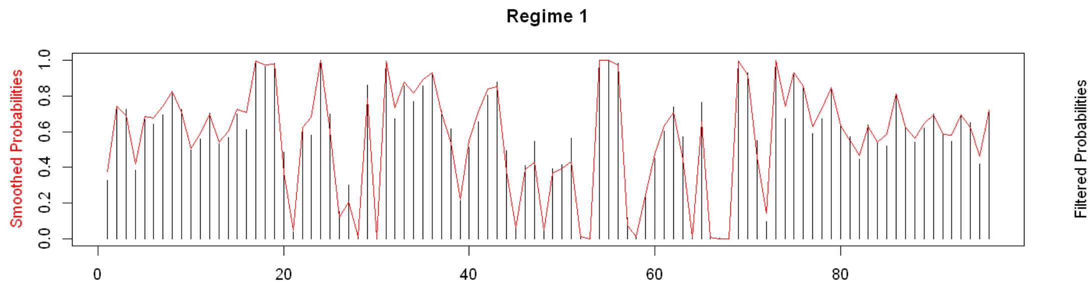
```
Augmented Dickey-Fuller Test

data:  resid
Dickey-Fuller = -4.2623, Lag order = 4, p-value = 0.01
alternative hypothesis: stationary



Box-Pierce test

data:  resid
X-squared = 2.9673, df = 1, p-value = 0.08496



Shapiro-Wilk normality test

data:  resid
W = 0.98488, p-value = 0.3388
```

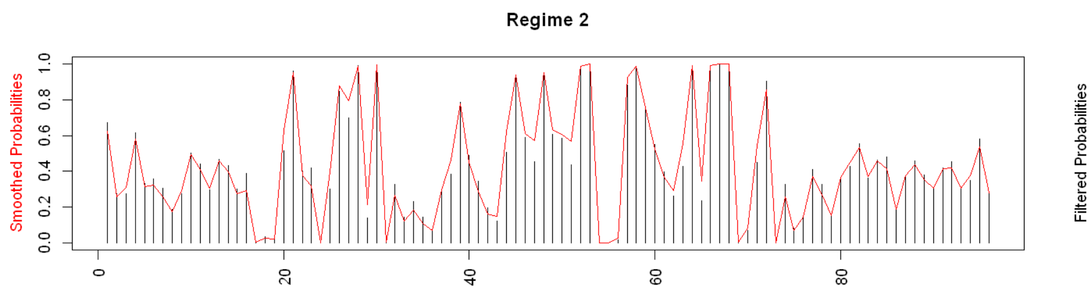
We cannot reject the zero hypothesis of stationary residuals, neither can we reject the zero hypothesis of no autocorrelation, nor the zero hypothesis of normal residuals. Therefore, the residuals are stationary, and normal, as well as the model adequate.

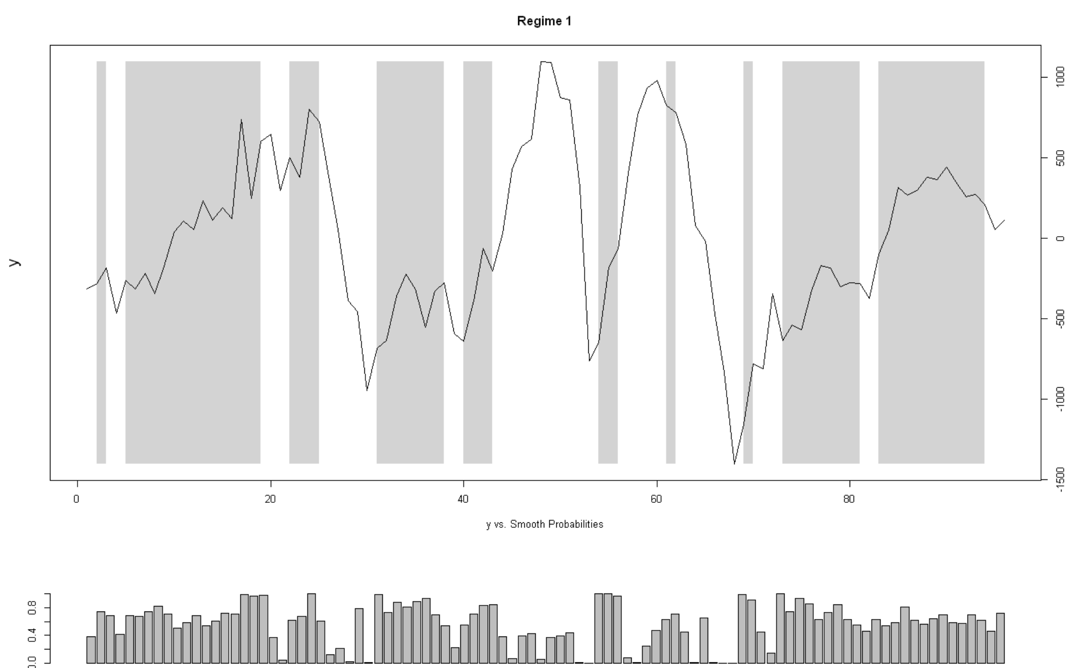
In terms of interpretation, we can check that the intercept is different as well as the lag coefficients. Here students could investigate what his means in terms of unconditional mean in each Regime, as well as the persistency in each Regime looking at the eigen values.

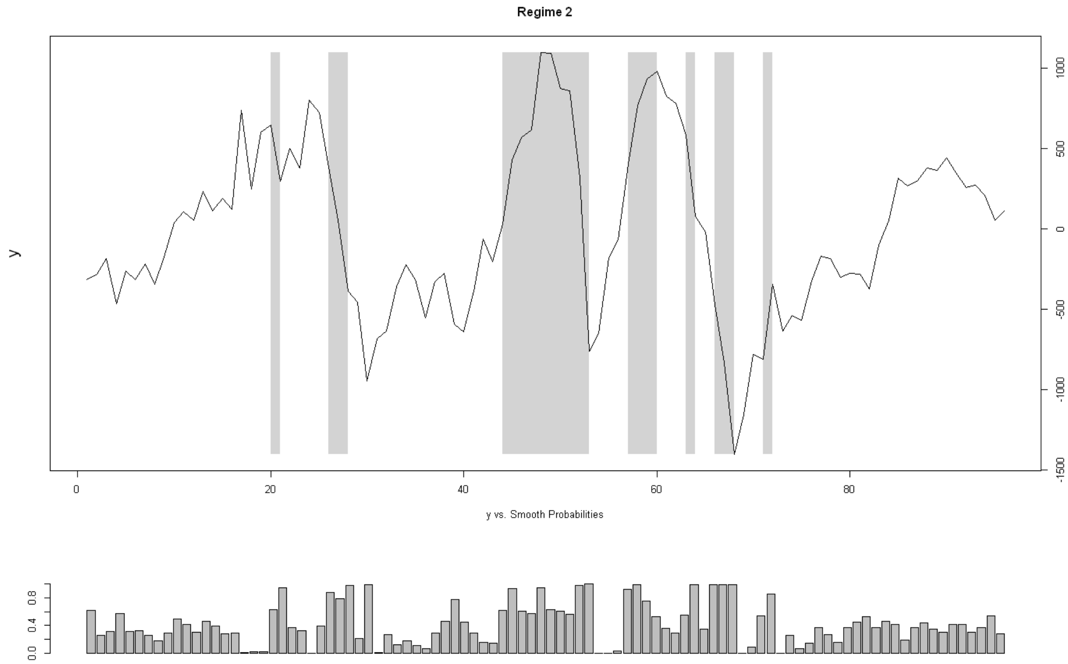
4 d)

[83]:
```
library(MSwM)
plotProb(mod)
```


Regime 1
Smoothed Probabilities
Filtered Probabilities
0.0 0.2 0.4 0.6 0.8 1.0
0 20 40 60 80


Regime 2
Smoothed Probabilities
Filtered Probabilities
0.0 0.2 0.4 0.6 0.8 1.0
0 20 40 60 80


Regime 1
y
1000 500 0 -500 -1000 -1500
0 20 40 60 80
y vs. Smooth Probabilities
0.0 0.4 0.8



There is a great deal of switching between regimes in Portugal. The model is basically capturing the two massive episodes, the great recession and the public debt crises as Regime 2.

```
[ ]:
```