TIME SERIES AUTO REGRESSION AND ERROR CORRECTION MODELS

Multi-Variate Econometric Model Configuration

Vector auto regression, Volatility, Granger causality & Error Correction

Sarit Maitra
Oct 23, 2020 · 8 min read ★

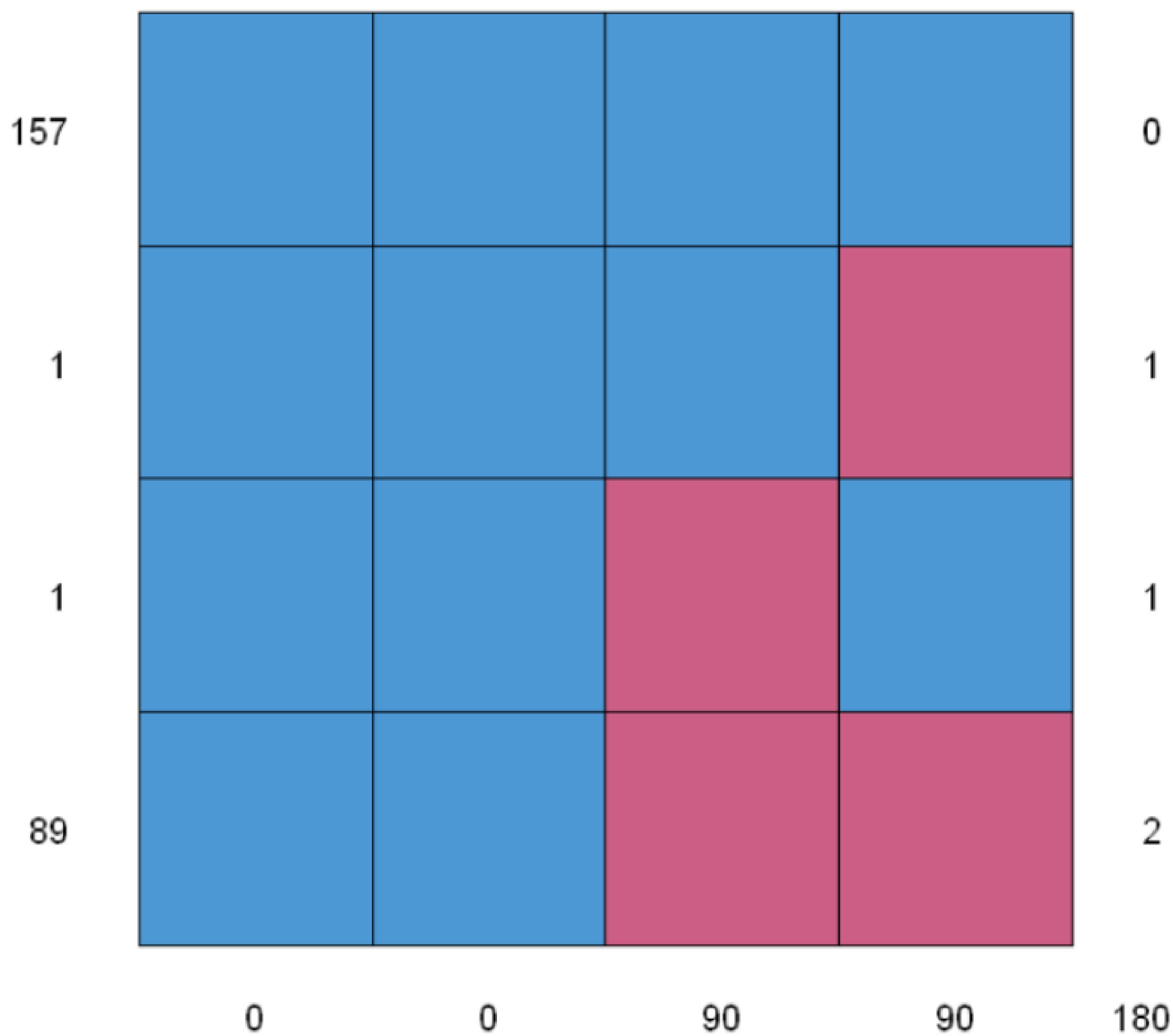


Image by author

Vector auto regression (VAR) to first difference generally creates integrated time-series (TS) models. But we may eliminate valuable information about the relationship among variables by differencing, where Vector Error Correction model (VECM) is applicable.

Granger causality:

VAR involves multiple exog variables which are important to predict future state of endog variable. Using Granger causality (GC) we can determine the importance of multiple variables and GC is only relevant with TS variables. We will use VAR to investigate GC here.

Here our use case is that, we have data of Western Texas Intermediate, Brent Crude oil and HenryHub Spot price and we shall forecast future 15 time steps of each. We shall use R program to solve this. Let us load the data :

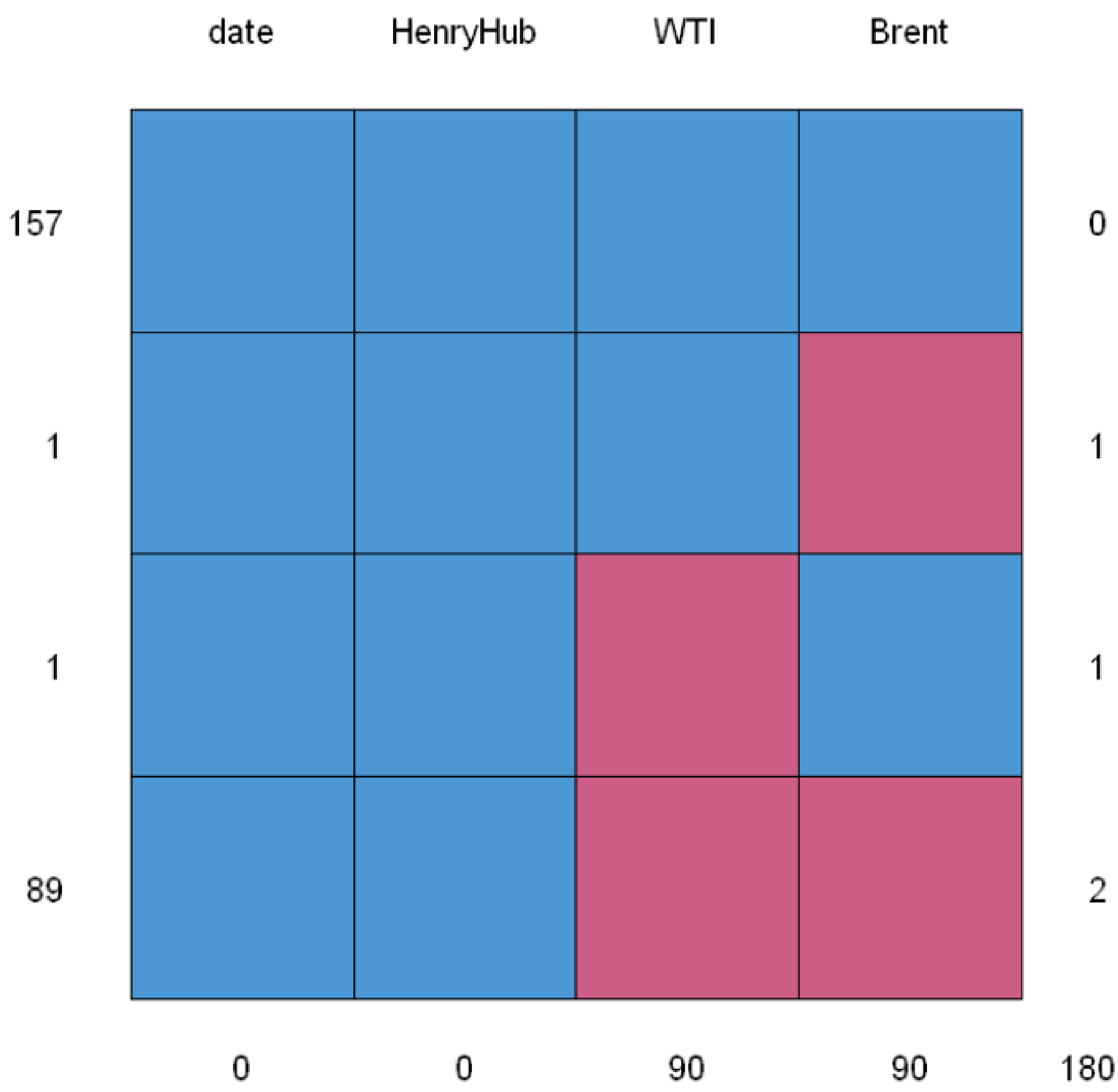
```
wti = as_tibble(get_fred_series("DCOILWTICO", "WTI",
observation_start = "2000-01-02"))
brent = as_tibble(get_fred_series("DCOILBRENTEU", "Brent",
observation_start = "2000-01-02"))
hh = as_tibble(get_fred_series("MHHNGSP", 'HenryHub',
observation_start = "2000-01-02"))
oil_prices = wti %>% left_join(brent, by="date")
prices = hh %>% left_join(oil_prices, by="date")
head(prices)
```

date	HenryHub	WTI	Brent
2000-02-01	2.66	28.28	27.35
2000-03-01	2.79	31.71	29.78
2000-04-01	3.04	NA	NA
2000-05-01	3.59	25.84	NA
2000-06-01	4.29	30.19	29.69
2000-07-01	3.99	NA	NA

Pre-processing data:

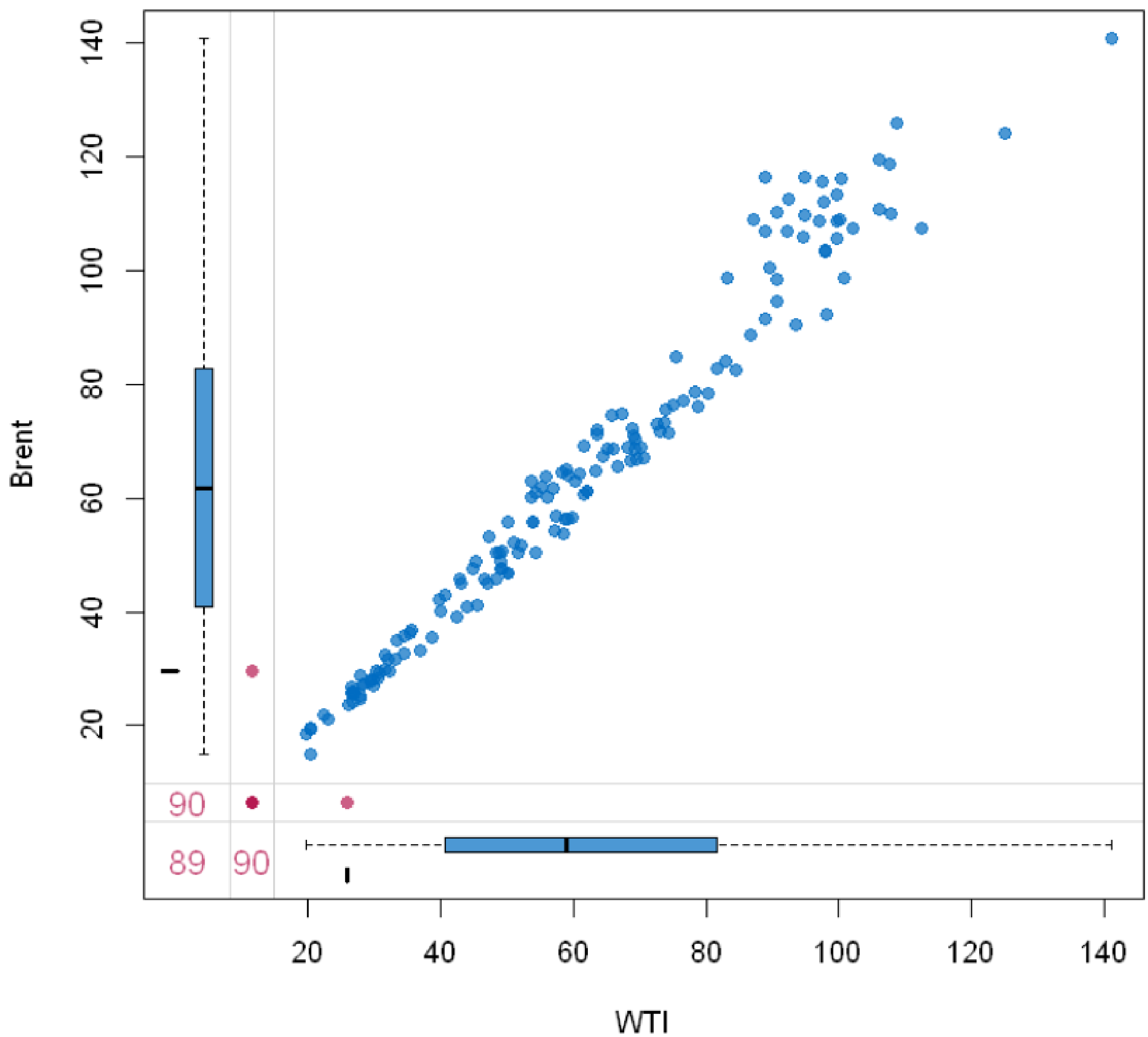
```
#understand the missing value pattern
md.pattern(prices)
```

	date	HenryHub	WTI	Brent	
157	1	1	1	1	0
1	1	1	1	0	1
1	1	1	0	1	1
89	1	1	0	0	2
	0	0	90	90	180



Margin plot to visualize missing values:

```
marginplot(prices[, c("WTI", "Brent")], col = mdc(1:2), cex.numbers =
1.2, pch = 19)
```



Imputing missing values:

```
imputes = mice(prices, m=5, maxit = 40)
# methods used for imputing
imputes
```

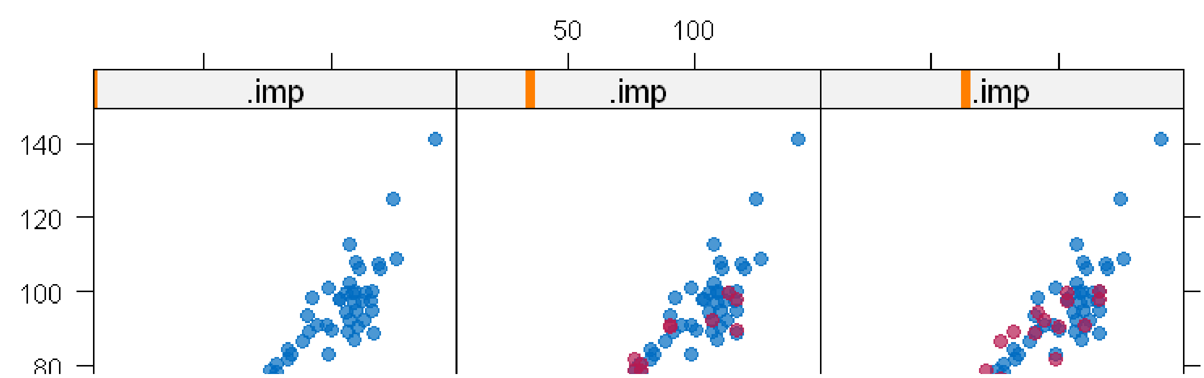
```
Class: mids
Number of multiple imputations:  5
Imputation methods:
    date HenryHub      WTI     Brent
      ""       ""    "pmm"     "pmm"
PredictorMatrix:
         date HenryHub WTI Brent
date        0        1   1     1
HenryHub    1        0   1     1
WTI         1        1   0     1
Brent       1        1   1     0
```

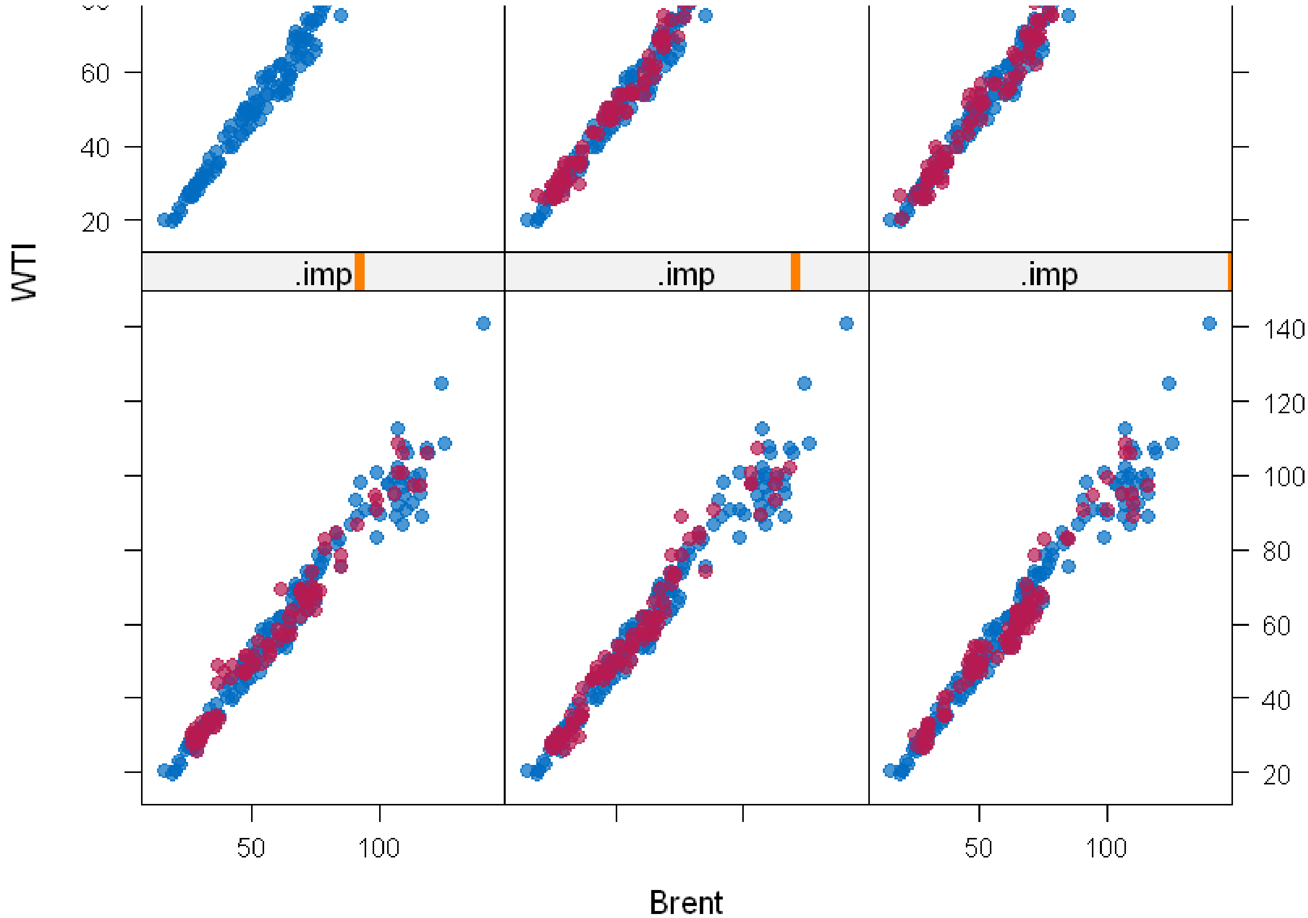
```
#Imputed dataset
data = complete(imputes,5)
head(data)
```

date	HenryHub	WTI	Brent
2000-02-01	2.66	28.28	27.35
2000-03-01	2.79	31.71	29.78
2000-04-01	3.04	29.89	29.42
2000-05-01	3.59	25.84	24.76
2000-06-01	4.29	30.19	29.69
2000-07-01	3.99	58.91	66.79

Goodness of fit:

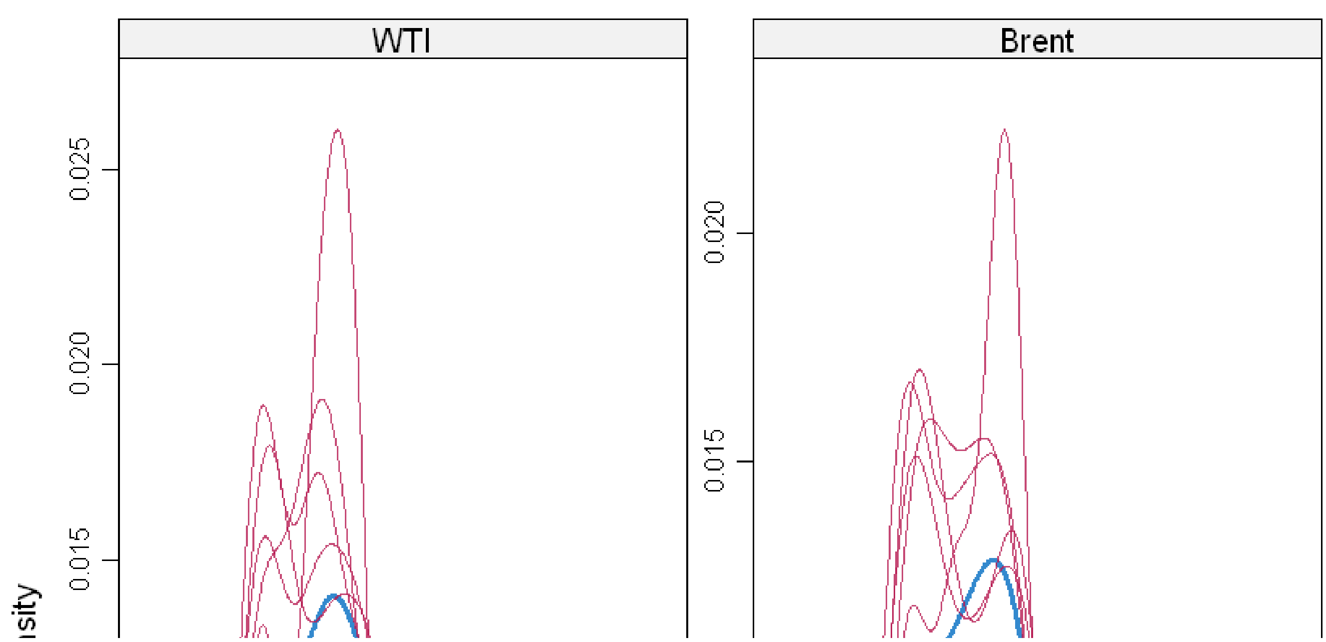
```
xyplot(imputes, WTI ~ Brent | .imp, pch = 20, cex = 1.4)
```

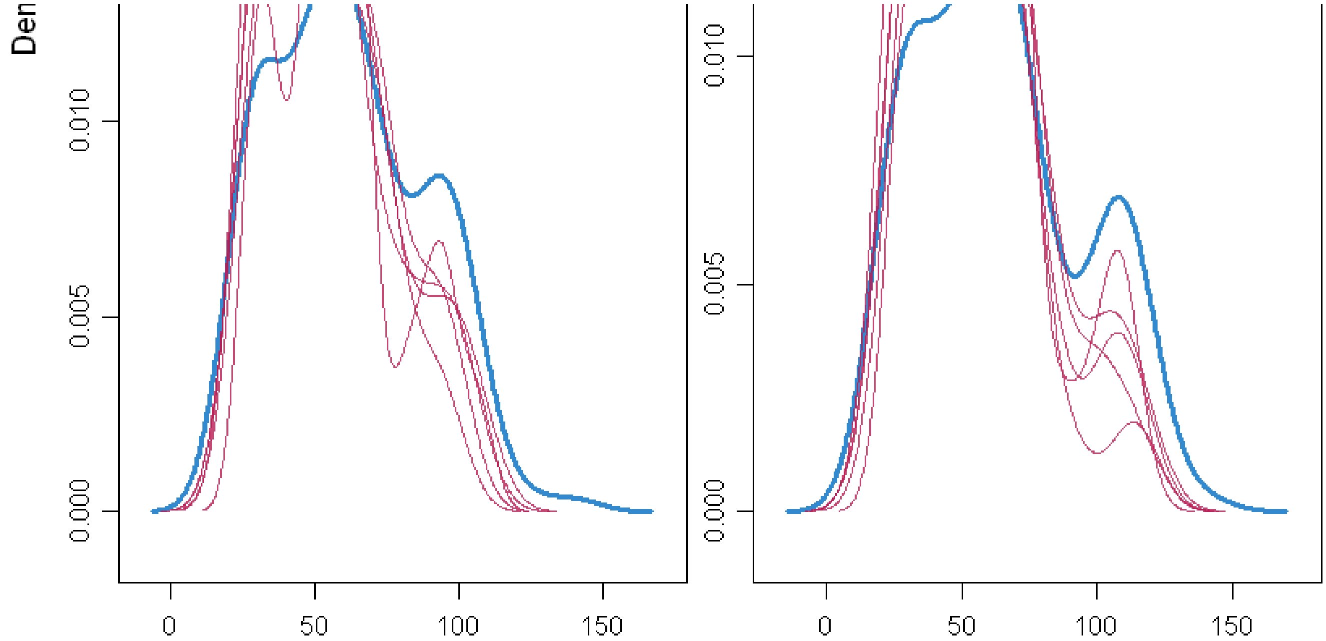




Density plot:

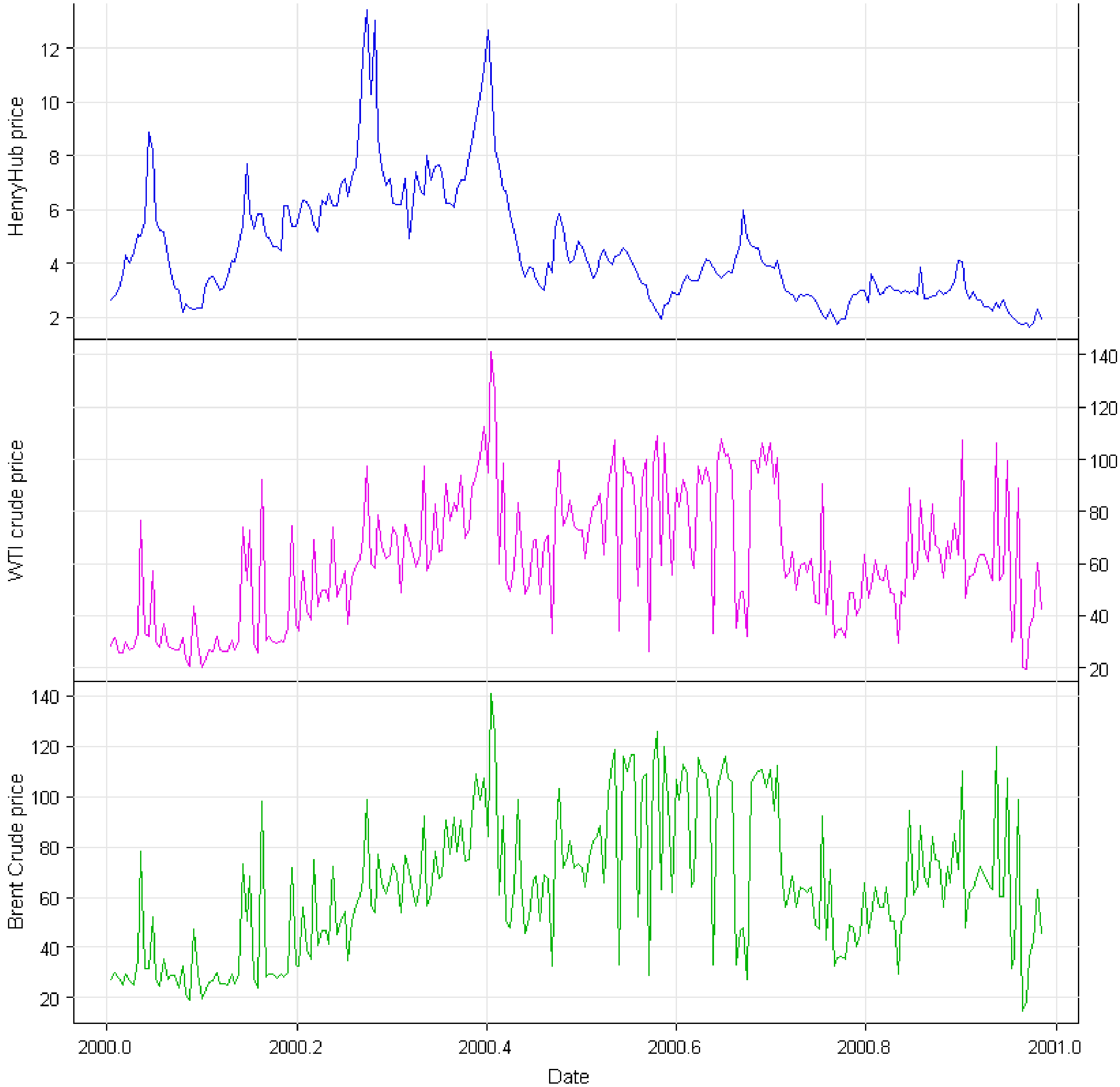
```
densityplot(imputes)
```





Time-series & Line plots:

```
# converting to time -series
henryhub = ts(data$HenryHub, start = c(2000,02,01), frequency = 252)
wti = ts(data$WTI, start = c(2000,02,01), frequency = 252)
brent = ts(data$Brent, start = c(2000,02,01), frequency = 252)

# time series line plot
par(mfrow=c(3,1), mar=c(0,3.5,0,3), oma=c(3.5,0,2,0), mgp=c(2,.6,0),
cex.lab=1.1, tcl=-.3, las=1)
plot(henryhub, ylab=expression('HenryHub price'), xaxt="no",
type='n')
  grid(lty=1, col=gray(.9))
  lines(henryhub, col=rgb(0,0,.9))

plot(wti, ylab=expression('WTI crude price'), xaxt="no", yaxt='no',
type='n')
  grid(lty=1, col=gray(.9))
  lines(wti, col=rgb(.9,0,.9))
  axis(4)
plot(brent, ylab=expression('Brent Crude price'))
  grid(lty=1, col=gray(.9))
  lines(brent, col=rgb(0,.7,0))
title(xlab="Date", outer=TRUE)

df = cbind(henryhub, wti, brent)
```



ADF Unit root test

Thumb rule: If calculated statistics > tabulated values, null hyposes can be rejected.

- ADF test (H0: series has unit root)
- PP test (H0: series has unit root)
- KPSS test (H0: series has no unit root)
- Zivot & Andrew test (H0: series has unit root)

```
adf.test(log(data[, "HenryHub"]))
adf.test(log(data[, "WTI"]))
adf.test(log(data[, "Brent"]))
```

```
        Augmented Dickey-Fuller Test

data:  log(data[, "HenryHub"])
Dickey-Fuller = -3.4604, Lag order = 6, p-value = 0.04712
alternative hypothesis: stationary


        Augmented Dickey-Fuller Test

data:  log(data[, "WTI"])
Dickey-Fuller = -2.5962, Lag order = 6, p-value = 0.3249
alternative hypothesis: stationary


        Augmented Dickey-Fuller Test

data:  log(data[, "Brent"])
Dickey-Fuller = -2.4523, Lag order = 6, p-value = 0.3854
alternative hypothesis: stationary
```

```
pp.test(log(data[, "HenryHub"]), type = "Z(t_alpha)")
pp.test(log(data[, "WTI"]), type = "Z(t_alpha)")
pp.test(log(data[, "Brent"]), type = "Z(t_alpha)")
```

```
        Phillips-Perron Unit Root Test

data:  log(data[, "HenryHub"])
Dickey-Fuller Z(t_alpha) = -3.4877, Truncation lag parameter = 5,
p-value = 0.04449
alternative hypothesis: stationary
```

```
Warning message in pp.test(log(data[, "WTI"]), type = "Z(t_alpha)"):
"p-value smaller than printed p-value"
```

```
        Phillips-Perron Unit Root Test
```

```
data:  log(data[, "WTI"])
Dickey-Fuller Z(t_alpha) = -9.248, Truncation lag parameter = 5,
p-value = 0.01
alternative hypothesis: stationary
```

```
Warning message in pp.test(log(data[, "Brent"]), type = "Z(t_alpha)"):
"p-value smaller than printed p-value"
```

```
	Phillips-Perron Unit Root Test

data:  log(data[, "Brent"])
Dickey-Fuller Z(t_alpha) = -9.3893, Truncation lag parameter = 5,
p-value = 0.01
alternative hypothesis: stationary
```

```
kpss.test(log(data[, "HenryHub"]))
kpss.test(log(data[, "WTI"]))
kpss.test(log(data[, "Brent"]))
```

```
Warning message in kpss.test(log(data[, "HenryHub"])):
"p-value smaller than printed p-value"
```

```
	KPSS Test for Level Stationarity

data:  log(data[, "HenryHub"])
KPSS Level = 1.899, Truncation lag parameter = 5, p-value = 0.01
```

```
Warning message in kpss.test(log(data[, "WTI"])):
"p-value smaller than printed p-value"
```

```
	KPSS Test for Level Stationarity

data:  log(data[, "WTI"])
KPSS Level = 1.3095, Truncation lag parameter = 5, p-value = 0.01
```

```
Warning message in kpss.test(log(data[, "Brent"])):
"p-value smaller than printed p-value"
```

```
	KPSS Test for Level Stationarity
```

```
data:  log(data[, "Brent"])
KPSS Level = 1.5159, Truncation lag parameter = 5, p-value = 0.01
```

Number of lags

```
VARselect(log(df), lag.max = 10, type="const")
```

$selection

AIC(n)	3
HQ(n)	3
SC(n)	1
FPE(n)	3

$criteria

	1	2	3	4	5	6	7	8	9	10
AIC(n)	-1.191455e+01	-1.195418e+01	-1.208738e+01	-1.203677e+01	-1.206986e+01	-1.205608e+01	-1.201783e+01	-1.202240e+01	-1.198340e+01	-1.196751e+01
HQ(n)	-1.184399e+01	-1.183071e+01	-1.191098e+01	-1.180745e+01	-1.178763e+01	-1.172094e+01	-1.162977e+01	-1.158142e+01	-1.148950e+01	-1.142069e+01
SC(n)	-1.173947e+01	-1.164781e+01	-1.164970e+01	-1.146778e+01	-1.136957e+01	-1.122449e+01	-1.105494e+01	-1.092820e+01	-1.075789e+01	-1.061070e+01
FPE(n)	6.692414e-06	6.432588e-06	5.630972e-06	5.924354e-06	5.733146e-06	5.815049e-06	6.045199e-06	6.022144e-06	6.267623e-06	6.375523e-06

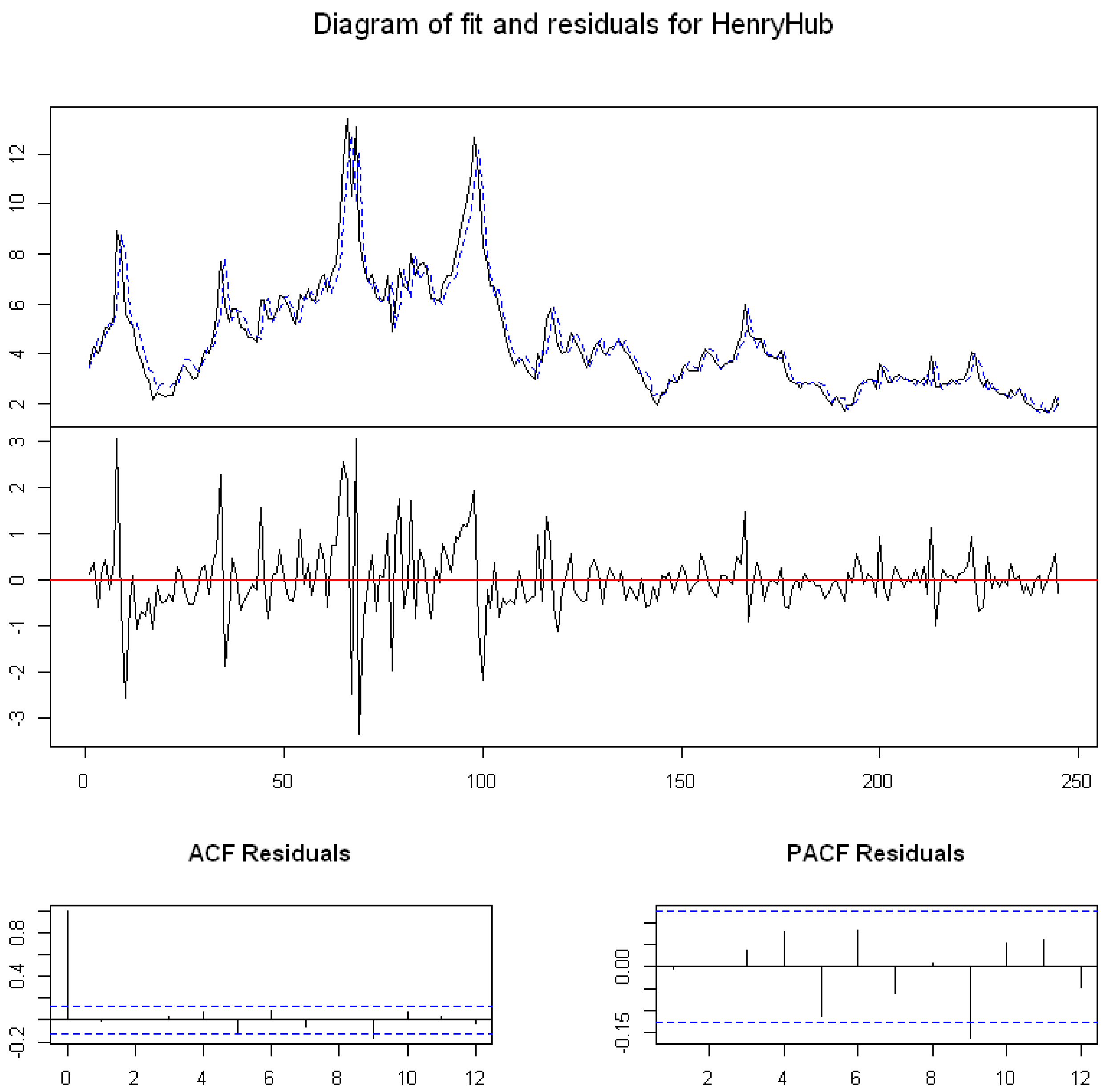
According to the AIC, HQ and FPE the optimal lag number is $p = 3$, whereas the SC criterion indicates an optimal lag length of $p = 1$. They estimated for all three lag orders a VAR including a constant and a trend as deterministic regressors and conducted diagnostic tests with respect to the residuals.

Estimating VAR Model:

```
var_mod <- VAR(df, p = 3, type = "both")
summary(var_mod)
plot(var_mod, names = "HenryHub")
```

```
VAR Estimation Results:
=========================
Endogenous variables: HenryHub, WTI, Brent
Deterministic variables: both
Sample size: 245
Log Likelihood: -2012.016
Roots of the characteristic polynomial:
0.902 0.8221 0.7166 0.6837 0.6837 0.4529 0.4529 0.3309 0.2936
Call:
VAR(y = data, p = 3, type = "both")


Estimation results for equation HenryHub:
=========================================
HenryHub = HenryHub.l1 + WTI.l1 + Brent.l1 + HenryHub.l2 + WTI.l2 + Brent.l2 + HenryHub.l3 + WTI.l3 + Brent.l3 + const + trend

             Estimate Std. Error t value Pr(>|t|)
HenryHub.l1  0.9222864  0.0655510  14.070  < 2e-16 ***
WTI.l1      -0.0011835  0.0143208  -0.083  0.93421
Brent.l1    -0.0012326  0.0125694  -0.098  0.92197
HenryHub.l2  0.0490920  0.0897412   0.547  0.58487
WTI.l2       0.0085102  0.0148313   0.574  0.56665
Brent.l2    -0.0031738  0.0131398  -0.242  0.80935
HenryHub.l3 -0.0830773  0.0688170  -1.207  0.22857
WTI.l3      -0.0016739  0.0142190  -0.118  0.90639
```

```
Brent.13     0.0007253 0.0125317   0.058 0.95389
const        0.6635724 0.2044061   3.246 0.00134 **
trend       -0.0021939 0.0010220  -2.147 0.03284 *
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```



Model Diagnostics:

Residuals:

Portmanteau goodness of-fit test to test the adequacy of the fitted model by checking whether the residuals are approximately white noise.

```
residuals = serial.test(var_mod, lags.pt=3, type="PT.asymptotic") # residuals
residuals$serial
```

```
	Portmanteau Test (asymptotic)

data:  Residuals of VAR object var_mod
Chi-squared = 5.4563, df = 0, p-value < 2.2e-16
```

The null hypothesis of no autocorrelation is rejected since the p-value < than the significance level of 0.05.

Normality test:

```
norm <- normality.test(var_mod)
norm$jb.mul
```

```
$JB

	JB-Test (multivariate)

data:  Residuals of VAR object var_mod
Chi-squared = 246.88, df = 6, p-value < 2.2e-16


$Skewness

	Skewness only (multivariate)

data:  Residuals of VAR object var_mod
Chi-squared = 9.9389, df = 3, p-value = 0.01909


$Kurtosis

	Kurtosis only (multivariate)

data:  Residuals of VAR object var_mod
Chi-squared = 236.94, df = 3, p-value < 2.2e-16
```

- The null hypothesis of the Jarque-Bera test is a joint hypothesis of the skewness being zero and the excess kurtosis being zero
- The result of the p-value shows the null hypothesis is rejected.

- Thus, we can conclude residuals does not follow a normal distribution.

Conditional volatility model:

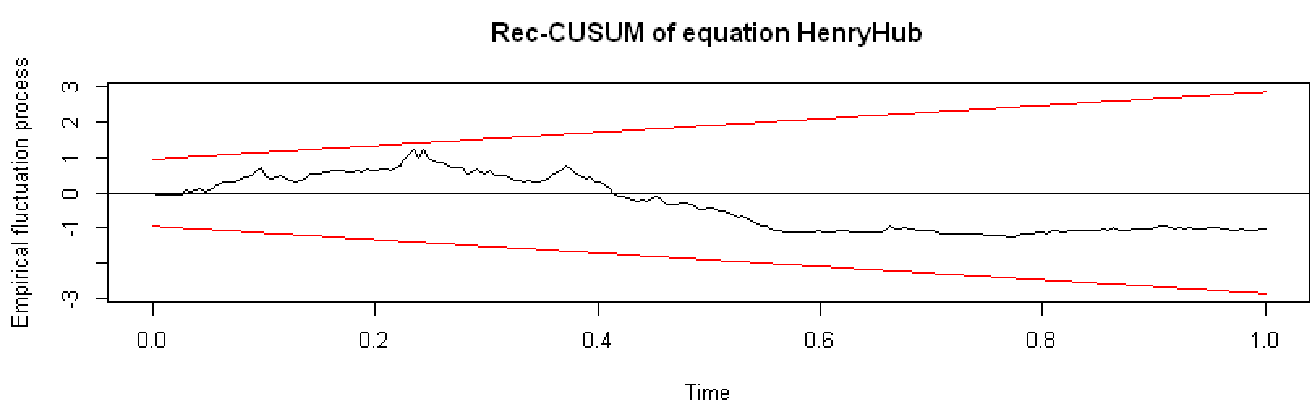
```
arch <- arch.test(var_mod, lags.multi = 5, multivariate.only = TRUE)
arch$arch.mul
```

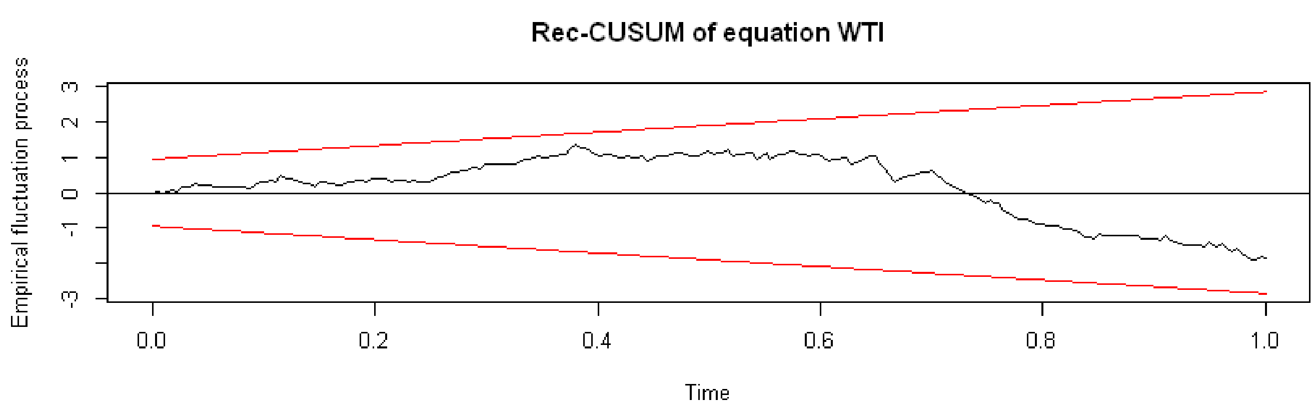
```
        ARCH (multivariate)

data:  Residuals of VAR object var_mod
Chi-squared = 375.04, df = 180, p-value = 7.772e-16
```

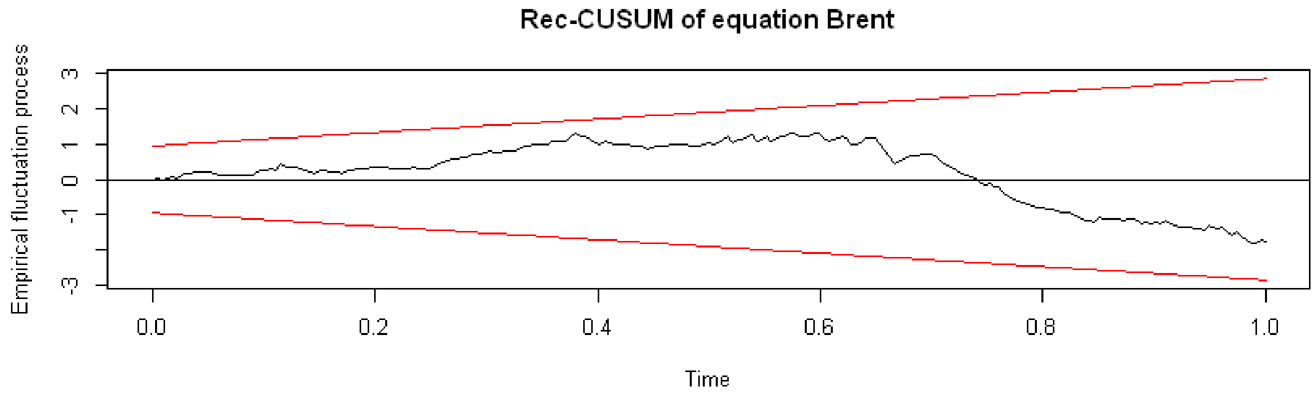
Here $p < 0.05$, therefore, so, we can assume that model suffers from heteroscedasticity.

Testing structural breaks:

```
## Stability
plot(stability(var_mod, type = "Rec-CUSUM"))
```







Here we find the system is stable because line plots do not exceed the red lines.

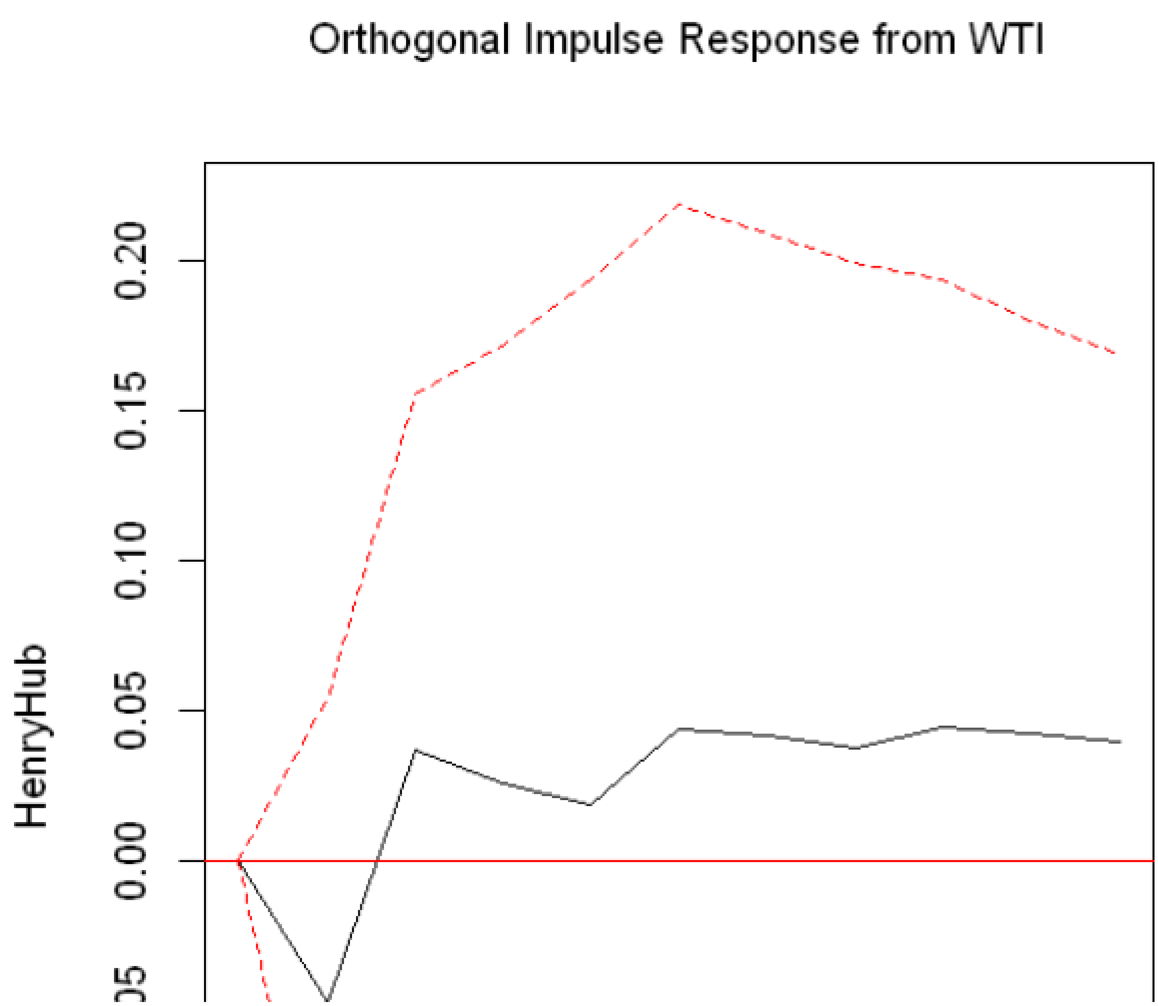
Granger Causality:

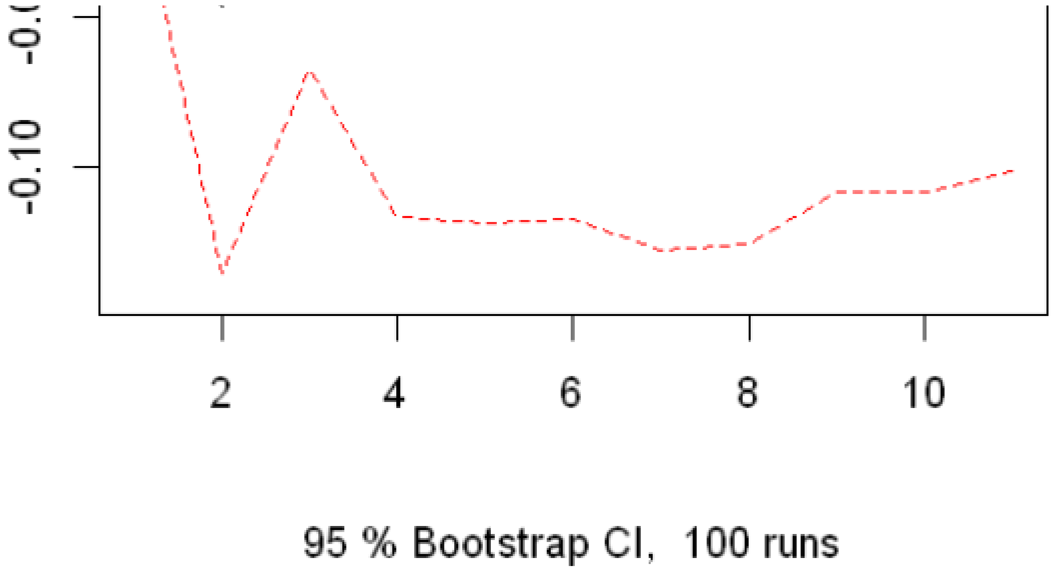
```
causality(var_mod, cause = 'wti')
causality(var_mod, cause = 'brent')
causality(var_mod, cause = 'henryhub')

$Granger

Granger causality H0: WTI do not Granger-cause HenryHub Brent

data:  VAR object var_mod
F-Test = 1.4459, df1 = 6, df2 = 702, p-value = 0.1944

$Instant
H0: No instantaneous causality between: WTI and HenryHub Brent

data:  VAR object var_mod
Chi-squared = 120.3, df = 2, p-value < 2.2e-16

$Granger

Granger causality H0: Brent do not Granger-cause HenryHub WTI

data:  VAR object var_mod
F-Test = 1.1576, df1 = 6, df2 = 702, p-value = 0.3273


$Instant

H0: No instantaneous causality between: Brent and HenryHub WTI

data:  VAR object var_mod
Chi-squared = 120.29, df = 2, p-value < 2.2e-16
```

```
$Granger

Granger causality H0: HenryHub do not Granger-cause WTI Brent

data:  VAR object var_mod
F-Test = 6.4644, df1 = 6, df2 = 702, p-value = 1.197e-06


$Instant

H0: No instantaneous causality between: HenryHub and WTI Brent

data:  VAR object var_mod
Chi-squared = 3.9472, df = 2, p-value = 0.139
```

Impulse response:

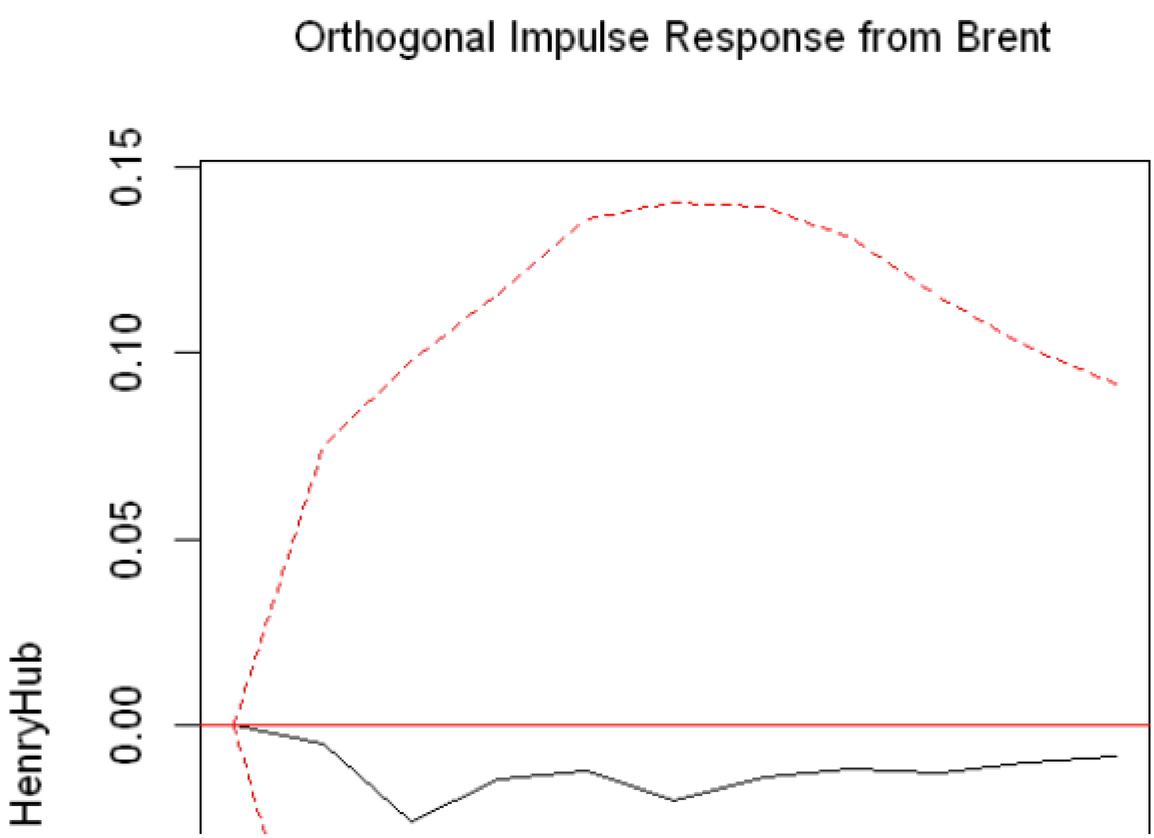
```
## IRF
plot(irf(var_mod, impulse = "wti", response = c("henryhub"),
n_ahead=15, boot = TRUE))
```

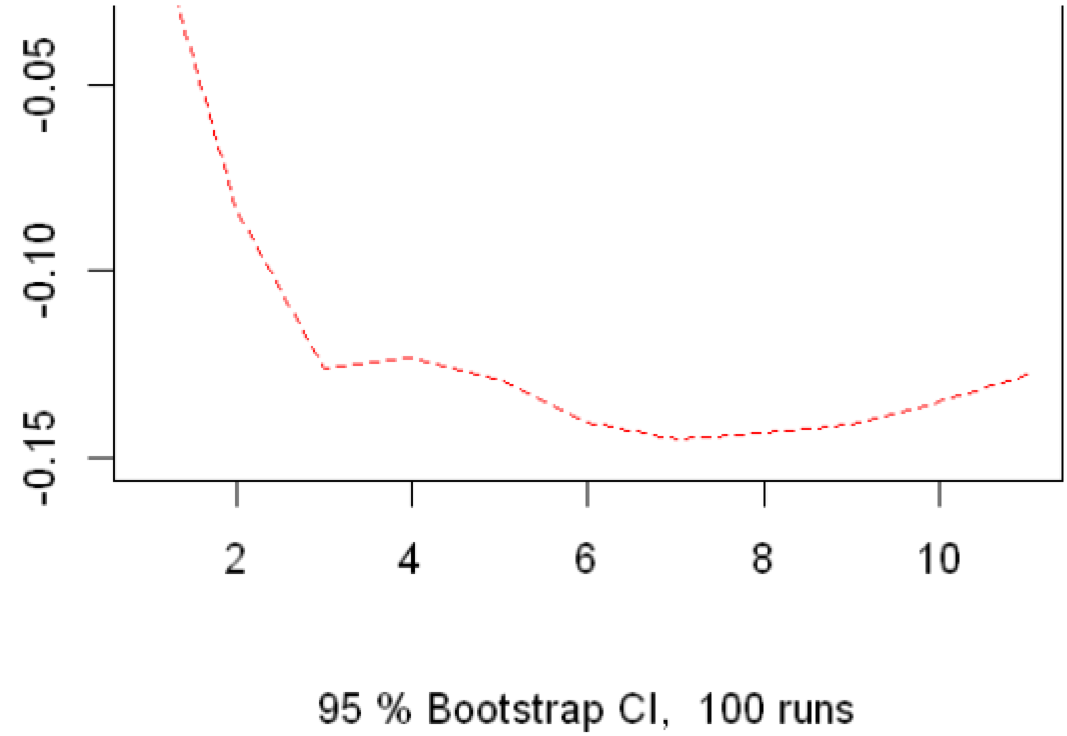




Confidence interval span is quite big indicating we have relatively bigger room for error; so, to effect there could be no effect between WTI & HenryHub; however, on the basis of point estimate, WTI will have positive effect on HenryHub.

```
## IRF
plot(irf(var_mod, impulse = "Brent", response = c("HenryHub"),
n_ahead=15, boot = TRUE))
```



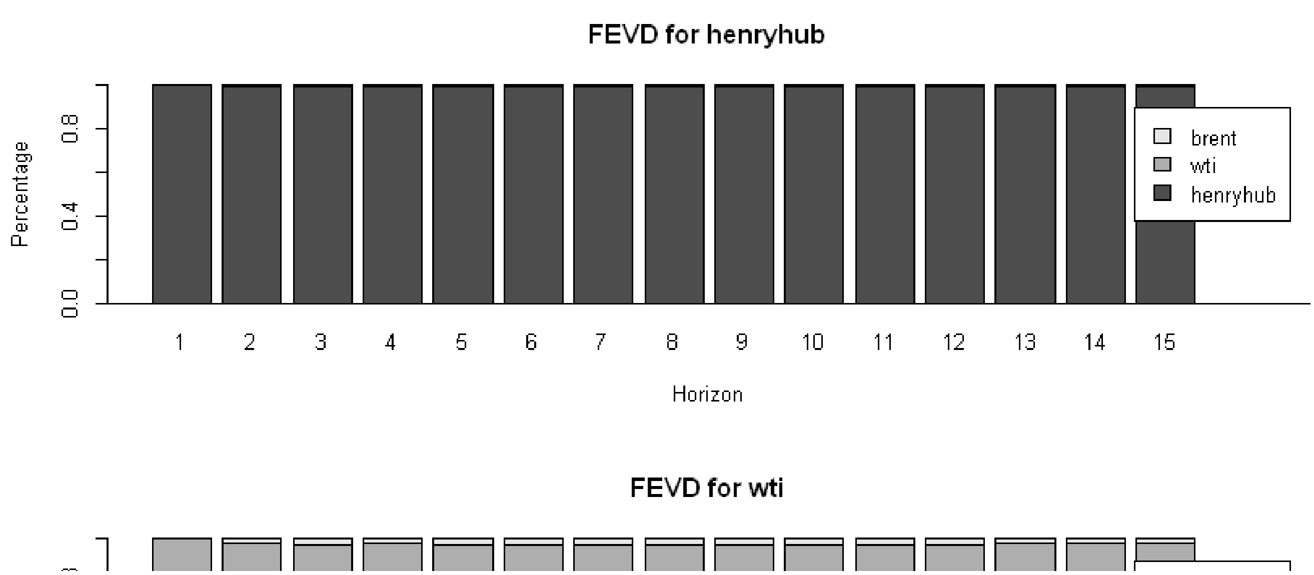


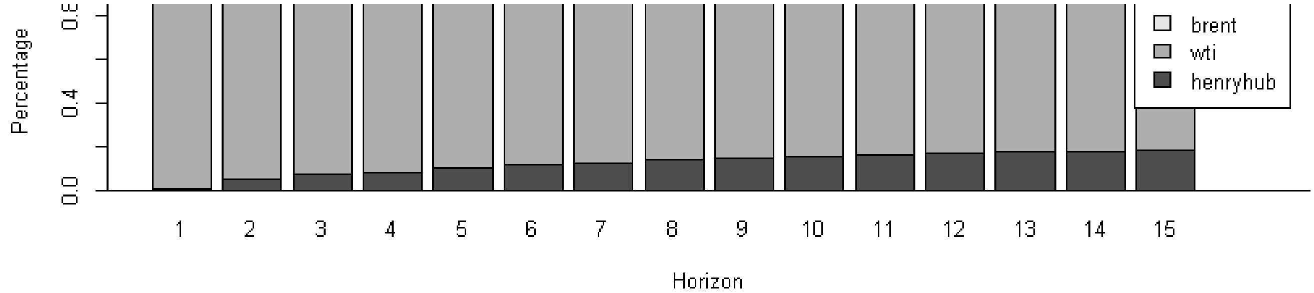
here too, we find there is no impact of Brent on HenryHub; we have a relatively bigger room for error; however, Brent having negative effect on HenryHub.

Forecast error variance decomposition:

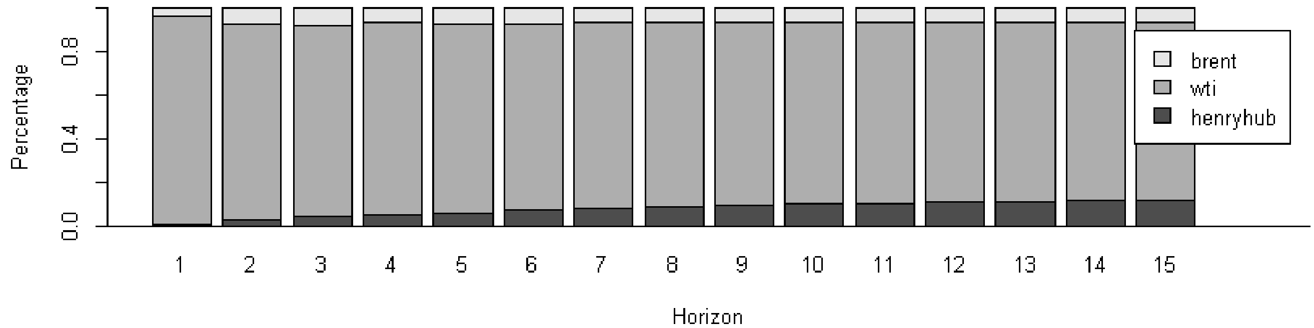
Forecast Error Variance Decomposition examines the impact of variables on one another.

```
plot(fevd(var_mod, n.ahead = 15))
```





FEVD for brent



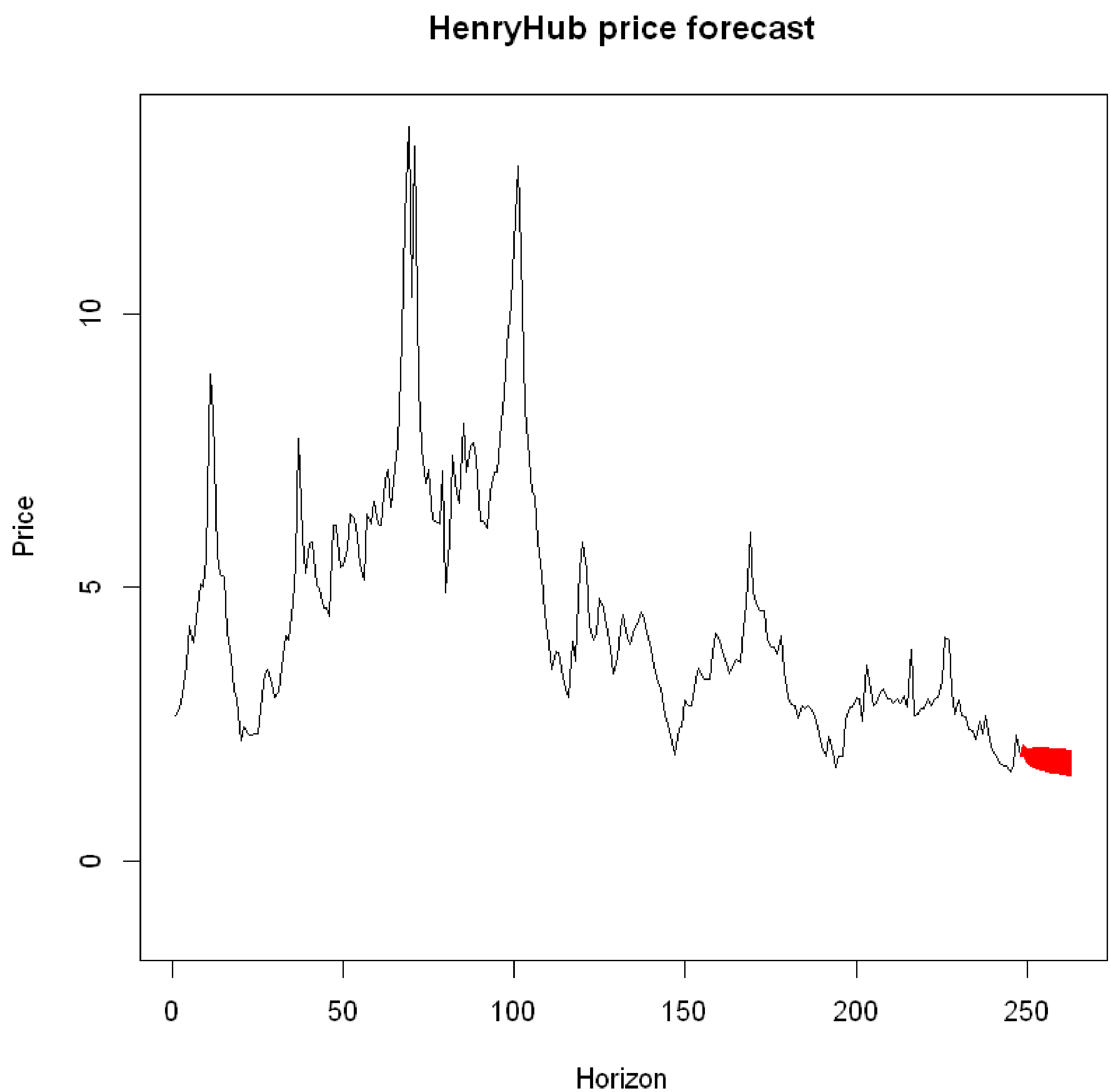
FEVD estimates the contribution of a shock in each variable to the response in both variables. Almost 100 % of the variance in HenryHub is caused by HenryHub itself, while only about 80 % in the variance of WTI and Brent caused by themselves and others.

Forecast VAR:

```
forecast <- predict(var_mod, n.ahead = 15, ci = 0.95)
fanchart(forecast, names = "henryhub", main = "HenryHub price
forecast", xlab = "Horizon", ylab = "Price")
forecast
```

```
$HenryHub
         fcst      lower    upper       CI
[1,] 2.023945  0.5085243 3.539366 1.515421
[2,] 1.918098 -0.1408333 3.977030 2.058932
[3,] 1.898657 -0.5686141 4.365928 2.467271
[4,] 1.906046 -0.8471416 4.659233 2.753187
[5,] 1.877952 -1.0853022 4.841206 2.963254
[6,] 1.865119 -1.2529830 4.983221 3.118102
[7,] 1.864559 -1.3717686 5.100886 3.236328
[8,] 1.851493 -1.4754498 5.178436 3.326943
```

```
 [9,] 1.842840 -1.5542984 5.239979 3.397139
[10,] 1.837991 -1.6141955 5.290177 3.452186
[11,] 1.828779 -1.6666943 5.324253 3.495474
[12,] 1.820372 -1.7093403 5.350085 3.529713
[13,] 1.812809 -1.7441342 5.369753 3.556944
[14,] 1.803387 -1.7752512 5.382025 3.578638
[15,] 1.793695 -1.8022861 5.389676 3.595981
```



Cointegration test:

If variables are found to be cointegrated, then we should work with an error correction model (ECM) involving these variables.

Eigen test:

```
coin = ca.jo(df, type = "eigen", ecdet = "none", K = 3, spec =
"transitory")
summary(coin)
```

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

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

Eigenvalues (lambda):
[1] 0.12310724 0.05612077 0.03222678

Values of teststatistic and critical values of test:

          test 10pct  5pct  1pct
r <= 2 |  8.03  6.50  8.18 11.65
r <= 1 | 14.15 12.91 14.90 19.19
r = 0  | 32.19 18.90 21.07 25.75

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

            henryhub.l1      wti.l1     brent.l1
henryhub.l1   1.0000000  1.00000000  1.000000000
wti.l1       -0.7789157 -0.94858843  0.008402311
brent.l1      0.6724960  0.05803292 -0.023625046

Weights W:
(This is the loading matrix)

            henryhub.l1      wti.l1    brent.l1
henryhub.d -0.002444656 0.004290066 -0.05996139
wti.d        0.385850853 0.233467332  0.54049420
brent.d    -0.252220780 0.266163009  0.65840636
```

- r=2 & r=1 : the critical value @ 95% are > test statistics, so we cannot reject H0; means our series are cointegrated and they have long run relations. Hence we can run ECM.

- r = 0: critical value @95% < test statistic, so we reject H0

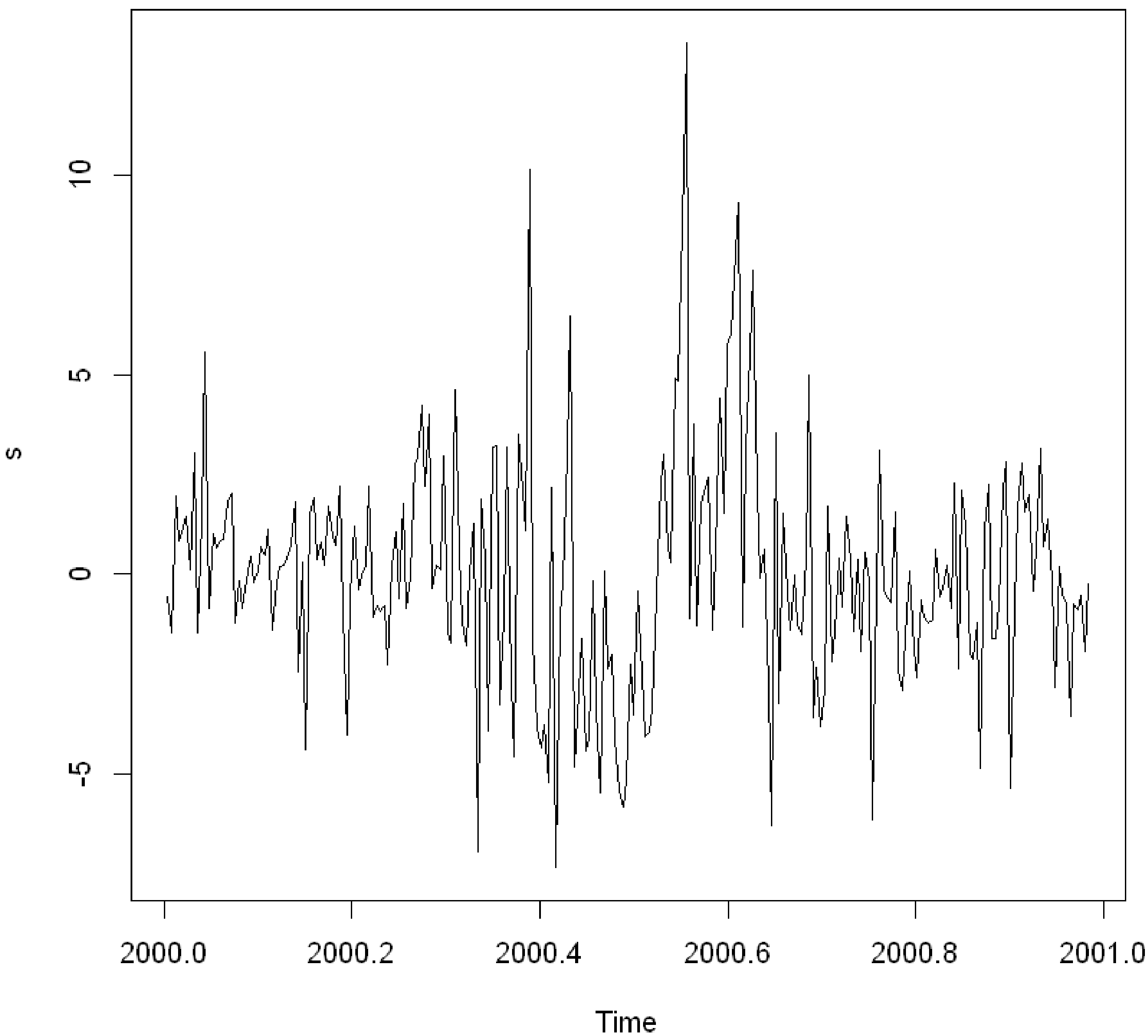
Trace test:

```
coin = ca.jo(df, type = "trace", ecdet = "none", K = 3, spec =
"transitory")
# endet = 'none' means there is a linear trend in data
summary(coin)
```

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

Test type: trace statistic , with linear trend

Eigenvalues (lambda):
[1] 0.12310724 0.05612077 0.03222678

Values of teststatistic and critical values of test:

          test 10pct  5pct  1pct
r <= 2 |  8.03  6.50  8.18 11.65
r <= 1 | 22.18 15.66 17.95 23.52
r = 0  | 54.36 28.71 31.52 37.22

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

            henryhub.l1      wti.l1     brent.l1
henryhub.l1   1.0000000  1.00000000  1.000000000
wti.l1       -0.7789157 -0.94858843  0.008402311
brent.l1      0.6724960  0.05803292 -0.023625046

Weights W:
(This is the loading matrix)

            henryhub.l1      wti.l1    brent.l1
henryhub.d -0.002444656 0.004290066 -0.05996139
wti.d       0.385850853 0.233467332  0.54049420
brent.d    -0.252220780 0.266163009  0.65840636
```

```
s = 1*henryhub - 0.7788110 * wti + 0.6723858*brent + 0.4075001
plot(s, type ='l')
```

```
	Augmented Dickey-Fuller Test

data:  s
Dickey-Fuller = -3.9344, Lag order = 6, p-value = 0.01283
alternative hypothesis: stationary
```



Error correction model:

```
model_vecm = VECM(df, lag=3, r=1, estim = 'ML')
summary(model_vecm)
```

```
#############
###Model VECM
#############
Full sample size: 248   End sample size: 244
Number of variables: 3  Number of estimated slope parameters 33
AIC 2032.957    BIC 2155.358    SSR 196301.3
Cointegrating vector (estimated by ML):
   henryhub        wti     brent
r1        1 -0.8056006 0.6861932


                  ECT                Intercept          henryhub -1
Equation henryhub 0.0056(0.0252)     -0.0005(0.0586)    -0.0341(0.0697)
Equation wti      0.6411(0.6071)     0.9615(1.4087)     4.0662(1.6759)*
Equation brent    0.0620(0.6956)     0.3144(1.6141)     4.1791(1.9202)*
                  wti -1              brent -1           henryhub -2
Equation henryhub 0.0031(0.0206)      -0.0039(0.0178)    0.0269(0.0700)
Equation wti      -1.0253(0.4962)*    0.3054(0.4271)     2.9330(1.6834).
Equation brent    -0.8660(0.5686)     0.0784(0.4893)     2.9353(1.9288)
                  wti -2              brent -2           henryhub -3
Equation henryhub 0.0090(0.0186)      -0.0045(0.0161)    -0.0352(0.0691)
Equation wti      -0.7466(0.4482).    0.2381(0.3865)     0.8158(1.6632)
Equation brent    -0.6502(0.5136)     0.0942(0.4428)     0.8632(1.9057)
                  wti -3             brent -3
Equation henryhub 0.0060(0.0149)     -0.0022(0.0131)
Equation wti      0.1549(0.3594)     -0.1605(0.3142)
Equation brent    0.1666(0.4118)     -0.1871(0.3600)
```

VAR representation of VECM

```
VARrep(model_vecm)
```

	constant	henryhub.l1	wti.l1	brent.l1	henryhub.l2	wti.l2	brent.l2	henryhub.l3	wti.l3	brent.l3	henryhub.
henryhub	-0.0004806127	0.9715251	-0.001415281	-7.319949e-05	0.06092865	0.005881203	-0.0006319013	-0.06205003	-0.002984156	0.002376428	0.035188
wti	0.9614912589	4.7073087	-0.541764120	7.453706e-01	-1.13314517	0.278697424	-0.0673222273	-2.11727341	0.901514891	-0.398614006	-0.81577
brent	0.3144215925	4.2411088	-0.915974879	1.120904e+00	-1.24383218	0.215857487	0.0158534913	-2.07206761	0.816784389	-0.281293110	-0.863201

Forecast VECM:

```
fcast = predict(model_vecm, n.ahead = 15)
fcast
```

	henryhub	wti	brent
249	2.051539	45.32164	48.27112
250	2.006353	52.23220	55.64675
251	1.950159	45.78249	49.17743
252	1.982784	46.95041	50.39440
253	1.969997	49.37090	53.00246
254	1.942418	47.18312	50.82095
255	1.950695	47.63156	51.29926
256	1.941458	48.50981	52.26203
257	1.927774	47.80119	51.57116
258	1.926214	48.00356	51.79891
259	1.918714	48.34895	52.18861
260	1.909685	48.15135	52.01215
261	1.904833	48.26792	52.15220
262	1.897930	48.43174	52.34589
263	1.890522	48.40989	52.34586

Summary:

Above we have shown a simplified approach. However, in real-life scenario building a VAR/ECM model involves several steps as stated below:

1. Information criterion (IC) to identify the optimal order; I have used AIC here; however HQ, FPE and SC can also be tried if the lag values are different in such cases.

2. Once optimal lag is identified, we need to estimate the model using OLS method and check the test statistic of the residuals to determine the goodness of fit o the model.

3. Forecast using fitted model.

4. Error correction model can be used if variables are found to be cointegrated.

Connect me here.

Note: The programs described here are experimental and should be used with caution for any commercial purpose. All such use at your own risk.

Econometrics Timeseries Forecasting Error Correction Vector Autoregression