Oil Supply, Demand, and Prices Forecasting: Structural VAR Model (1973–2007)

Tony Au

2025-09-10

Introduction

This report analyzes the dynamics of global crude oil supply, demand, and prices using a Vector Autoregression (VAR) model. The variables considered are the percent change in global crude oil production (`dprod`), changes in freight rates deflated by US CPI (`rea`), and the real price of oil (`rpo`). The goal is to forecast oil price changes and understand the impact of structural shocks on macroeconomic indicators like real GDP and inflation.

```
rm(list=ls())
# install.packages("vars")  # Install once if needed
# install.packages("roll")
suppressPackageStartupMessages(library(quantmod))  # Data download (FRED) and xts helpers
suppressPackageStartupMessages(library(ggpubr))    # Plot arrangements
suppressPackageStartupMessages(library(lubridate)) # Date utilities
suppressPackageStartupMessages(library(dynlm))     # Time-series OLS with L() lag operator
suppressPackageStartupMessages(library(strucchange)) # Stability tests (CUSUM)
suppressPackageStartupMessages(library(forecast))  # fanchart and forecasting utils
library(tidyverse)  # ggplot2 and data wrangling
library(tseries)    # Time-series tests
library(vars)       # VAR estimation, IRF, FEVD, diagnostics
library(readxl)     # Read Excel input (oildata.xlsx)
```

3-variable VAR for (oil supply, global demand, oil price)

This section builds the monthly dataset, converts to time-series objects, and visualizes the series. We treat dprod as supply, rea as global demand proxy, and rpo as real oil price. Monthly frequency is set to 12 for consistent lagging and diagnostics.

```
# Load the data
data.oil <- read_xlsx("oildata.xlsx")

# dprod: percent change in global crude oil production
# rea: changes in freight rates deflated by US CPI (in logs)
# rpo: real price of oil (in logs)

data.oil <- ts(data.oil[,2:4], start = c(1973,2), end = c(2007,12), frequency = 12 )
head(data.oil,3)
```

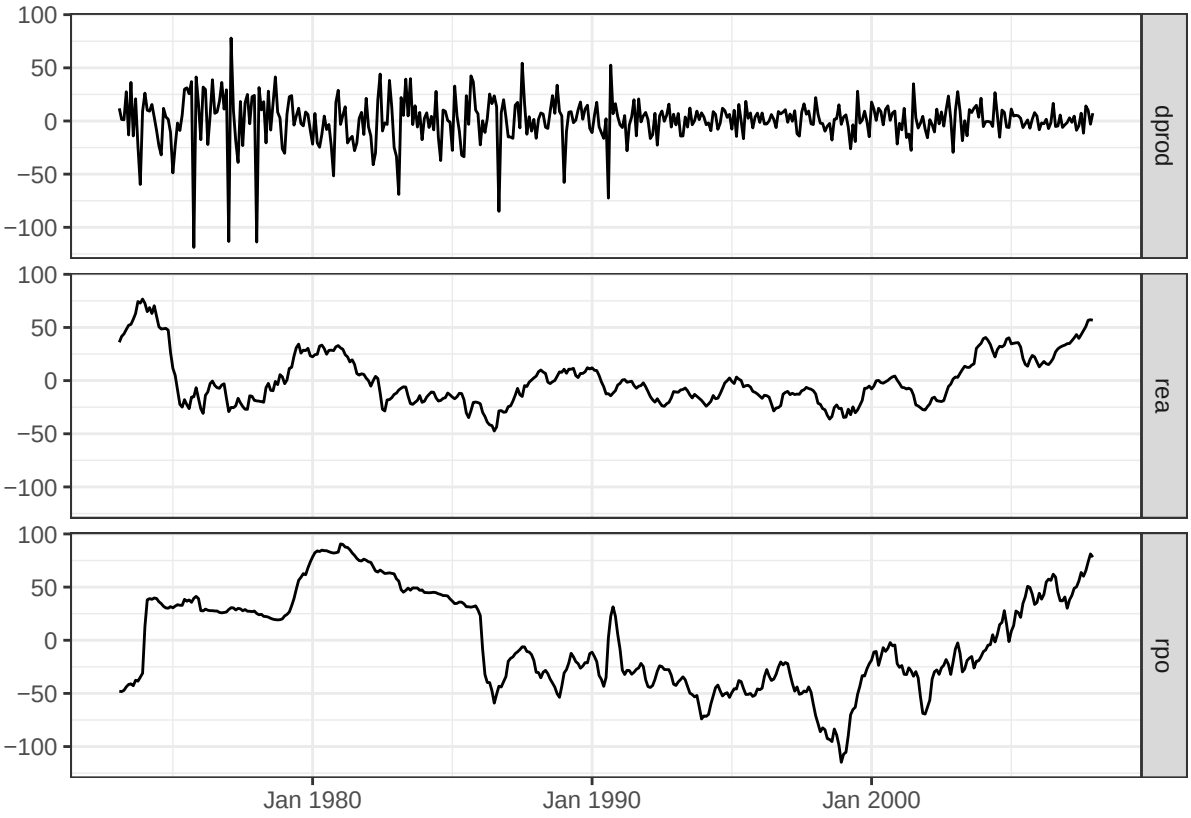
```
##              dprod      rea       rpo
## Feb 1973 11.877264 36.06423 -47.99953
```

```
## Mar 1973  1.419150 41.52702 -48.28647
## Apr 1973  1.177711 43.99125 -47.08250
```

```
tail(data.oil,3)
```

```
##              dprod      rea      rpo
## Oct 2007 10.104911 56.80675 73.67638
## Nov 2007 -3.038820 57.27422 81.20862
## Dec 2007  7.315859 56.82430 78.11541
```

```
data.oil <- as.xts(data.oil)

# Plot the series
autoplot(data.oil) + xlab("") + theme_bw()
```



Step 1. VAR lag selection

We choose the VAR lag length by minimizing information criteria (AIC, HQ, SC, FPE). Longer lags capture richer dynamics but reduce degrees of freedom.

```
var.info <- VARselect(data.oil, lag.max = 24, type = "const")  # IC-based lag selection up to 24
var.info$selection  # Shows optimal lags by AIC, HQ, SC, FPE
```

```
## AIC(n)  HQ(n)  SC(n) FPE(n)
##      3      2      2      3
```

Step 2. Estimate VAR(24) as in the class example

The VAR(p) system is Yt = c + A1Yt−1 + … + ApYt−p + t, with t ~ N(0, Σ). We estimate with OLS equation-by-equation, which is efficient under identical regressors.

```r
nlag    <- 24   # Chosen lag order (can compare with var.info results)
var.oil <- VAR(data.oil, p = nlag, type = "const")    # Estimate VAR(p) with constant
summary(var.oil)  # Obtain coefficients, SEs, R^2 per equation
```

```
##
## VAR Estimation Results:
## ========================
## Endogenous variables: dprod, rea, rpo
## Deterministic variables: const
## Sample size: 395
## Log Likelihood: -3975.113
## Roots of the characteristic polynomial:
## 0.9886 0.9733 0.9733 0.9693 0.9693 0.9681 0.9681 0.9614 0.9614 0.9598 0.9598 0.9556 0.9556 0.9545 0.9
## Call:
## VAR(y = data.oil, p = nlag, type = "const")
##
##
## Estimation results for equation dprod:
## ======================================
## dprod = dprod.l1 + rea.l1 + rpo.l1 + dprod.l2 + rea.l2 + rpo.l2 + dprod.l3 + rea.l3 + rpo.l3 + dprod
##
##            Estimate Std. Error t value Pr(>|t|)
## dprod.l1  -0.112467   0.055408  -2.030 0.043199 *
## rea.l1    -0.156742   0.253775  -0.618 0.537247
## rpo.l1    -0.372905   0.174891  -2.132 0.033745 *
## dprod.l2  -0.109896   0.055102  -1.994 0.046951 *
## rea.l2     0.415685   0.394676   1.053 0.293026
## rpo.l2     0.737053   0.301985   2.441 0.015197 *
## dprod.l3  -0.234035   0.055831  -4.192 3.58e-05 ***
## rea.l3    -0.435092   0.388297  -1.121 0.263329
## rpo.l3    -0.431526   0.320332  -1.347 0.178889
## dprod.l4  -0.088856   0.057813  -1.537 0.125282
## rea.l4     0.161676   0.385984   0.419 0.675592
## rpo.l4     0.125123   0.319485   0.392 0.695583
## dprod.l5  -0.209558   0.056387  -3.716 0.000238 ***
## rea.l5     0.087381   0.383971   0.228 0.820123
## rpo.l5    -0.275908   0.315437  -0.875 0.382398
## dprod.l6  -0.042377   0.057152  -0.741 0.458944
## rea.l6    -0.202976   0.380843  -0.533 0.594425
## rpo.l6     0.622631   0.316934   1.965 0.050327 .
## dprod.l7  -0.090661   0.056370  -1.608 0.108742
## rea.l7     0.673334   0.380061   1.772 0.077399 .
## rpo.l7    -0.368144   0.317467  -1.160 0.247059
## dprod.l8  -0.023336   0.056212  -0.415 0.678310
## rea.l8    -0.433399   0.381416  -1.136 0.256680
## rpo.l8    -0.183369   0.316796  -0.579 0.563112
## dprod.l9   0.109502   0.056062   1.953 0.051658 .
## rea.l9    -0.134027   0.377600  -0.355 0.722865
## rpo.l9     0.092109   0.317703   0.290 0.772063
## dprod.l10 -0.003386   0.055019  -0.062 0.950960
## rea.l10    0.046723   0.369740   0.126 0.899519
## rpo.l10    0.238444   0.317844   0.750 0.453687
## dprod.l11 -0.072375   0.054934  -1.317 0.188614
## rea.l11    0.039454   0.366040   0.108 0.914232
```

```
## rpo.l11   -0.318748   0.320664  -0.994 0.320958
## dprod.l12  0.215842   0.055153   3.913 0.000111 ***
## rea.l12   -0.087128   0.363126  -0.240 0.810530
## rpo.l12    0.330329   0.325375   1.015 0.310761
## dprod.l13  0.001946   0.055568   0.035 0.972085
## rea.l13    0.401447   0.360921   1.112 0.266846
## rpo.l13   -0.174269   0.323896  -0.538 0.590921
## dprod.l14 -0.057684   0.055671  -1.036 0.300909
## rea.l14   -0.996398   0.362616  -2.748 0.006338 **
## rpo.l14   -0.199736   0.314066  -0.636 0.525248
## dprod.l15  0.170134   0.055525   3.064 0.002368 **
## rea.l15    0.284906   0.363842   0.783 0.434174
## rpo.l15    0.076786   0.313836   0.245 0.806869
## dprod.l16 -0.054729   0.056367  -0.971 0.332305
## rea.l16    0.186723   0.364205   0.513 0.608521
## rpo.l16    0.260621   0.314313   0.829 0.407618
## dprod.l17  0.024832   0.055180   0.450 0.653000
## rea.l17   -0.353296   0.364329  -0.970 0.332915
## rpo.l17   -0.432424   0.313636  -1.379 0.168931
## dprod.l18  0.003462   0.054583   0.063 0.949467
## rea.l18    0.825666   0.364767   2.264 0.024267 *
## rpo.l18    0.676088   0.312269   2.165 0.031116 *
## dprod.l19  0.006364   0.054312   0.117 0.906797
## rea.l19   -1.365027   0.366750  -3.722 0.000233 ***
## rpo.l19   -0.535190   0.313672  -1.706 0.088933 .
## dprod.l20 -0.002440   0.053617  -0.046 0.963732
## rea.l20    0.844938   0.372104   2.271 0.023825 *
## rpo.l20    0.177568   0.314862   0.564 0.573177
## dprod.l21 -0.074854   0.053251  -1.406 0.160781
## rea.l21    0.543955   0.377990   1.439 0.151101
## rpo.l21   -0.142374   0.316199  -0.450 0.652820
## dprod.l22 -0.039390   0.052084  -0.756 0.450040
## rea.l22   -0.981520   0.377050  -2.603 0.009665 **
## rpo.l22    0.229827   0.317209   0.725 0.469267
## dprod.l23  0.003772   0.052016   0.073 0.942234
## rea.l23    0.703182   0.377398   1.863 0.063339 .
## rpo.l23   -0.093460   0.302180  -0.309 0.757303
## dprod.l24 -0.013552   0.051445  -0.263 0.792390
## rea.l24   -0.117849   0.243644  -0.484 0.628933
## rpo.l24   -0.075547   0.172429  -0.438 0.661585
## const      1.285499   1.052178   1.222 0.222695
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##
## Residual standard error: 18.74 on 322 degrees of freedom
## Multiple R-Squared: 0.3075,  Adjusted R-squared: 0.1527
## F-statistic: 1.986 on 72 and 322 DF,  p-value: 2.953e-05
##
##
## Estimation results for equation rea:
## ====================================
## rea = dprod.l1 + rea.l1 + rpo.l1 + dprod.l2 + rea.l2 + rpo.l2 + dprod.l3 + rea.l3 + rpo.l3 + dprod.l4
##
```

```
##              Estimate Std. Error t value Pr(>|t|)
## dprod.l1   0.0027433  0.0120024   0.229  0.81935
## rea.l1     1.2002428  0.0549723  21.834  < 2e-16 ***
## rpo.l1     0.1150021  0.0378845   3.036  0.00260 **
## dprod.l2   0.0156521  0.0119362   1.311  0.19069
## rea.l2    -0.2071506  0.0854941  -2.423  0.01594 *
## rpo.l2    -0.1648768  0.0654154  -2.520  0.01220 *
## dprod.l3   0.0313444  0.0120940   2.592  0.00998 **
## rea.l3    -0.0344062  0.0841122  -0.409  0.68277
## rpo.l3     0.1010787  0.0693897   1.457  0.14618
## dprod.l4  -0.0094031  0.0125233  -0.751  0.45329
## rea.l4     0.0471566  0.0836112   0.564  0.57315
## rpo.l4    -0.0123990  0.0692063  -0.179  0.85792
## dprod.l5  -0.0192391  0.0122145  -1.575  0.11622
## rea.l5    -0.0652641  0.0831752  -0.785  0.43323
## rpo.l5    -0.0446418  0.0683295  -0.653  0.51401
## dprod.l6  -0.0042791  0.0123802  -0.346  0.72984
## rea.l6     0.0584687  0.0824975   0.709  0.47900
## rpo.l6     0.0212631  0.0686538   0.310  0.75698
## dprod.l7   0.0049937  0.0122107   0.409  0.68284
## rea.l7     0.0128498  0.0823282   0.156  0.87607
## rpo.l7     0.0452353  0.0687691   0.658  0.51115
## dprod.l8   0.0223272  0.0121766   1.834  0.06763 .
## rea.l8    -0.0746515  0.0826217  -0.904  0.36692
## rpo.l8    -0.0189397  0.0686238  -0.276  0.78273
## dprod.l9   0.0041061  0.0121440   0.338  0.73550
## rea.l9    -0.0087456  0.0817952  -0.107  0.91492
## rpo.l9    -0.0356857  0.0688202  -0.519  0.60444
## dprod.l10 -0.0084309  0.0119181  -0.707  0.47982
## rea.l10    0.0666433  0.0800923   0.832  0.40598
## rpo.l10   -0.0562057  0.0688509  -0.816  0.41491
## dprod.l11  0.0132557  0.0118998   1.114  0.26613
## rea.l11    0.0666683  0.0792910   0.841  0.40108
## rpo.l11    0.0575540  0.0694617   0.829  0.40796
## dprod.l12  0.0107516  0.0119472   0.900  0.36883
## rea.l12    0.0382539  0.0786598   0.486  0.62707
## rpo.l12    0.0286399  0.0704822   0.406  0.68476
## dprod.l13 -0.0220708  0.0120370  -1.834  0.06764 .
## rea.l13   -0.1353641  0.0781822  -1.731  0.08434 .
## rpo.l13   -0.0045797  0.0701617  -0.065  0.94800
## dprod.l14 -0.0017813  0.0120594  -0.148  0.88267
## rea.l14    0.0025411  0.0785492   0.032  0.97421
## rpo.l14   -0.1223102  0.0680324  -1.798  0.07314 .
## dprod.l15  0.0149710  0.0120277   1.245  0.21414
## rea.l15   -0.1566280  0.0788148  -1.987  0.04774 *
## rpo.l15    0.0617027  0.0679826   0.908  0.36476
## dprod.l16 -0.0175335  0.0122101  -1.436  0.15198
## rea.l16    0.1379796  0.0788933   1.749  0.08125 .
## rpo.l16    0.0774075  0.0680859   1.137  0.25642
## dprod.l17  0.0093050  0.0119530   0.778  0.43687
## rea.l17    0.0402195  0.0789203   0.510  0.61067
## rpo.l17   -0.0347633  0.0679394  -0.512  0.60922
## dprod.l18  0.0016700  0.0118236   0.141  0.88776
## rea.l18   -0.0221396  0.0790152  -0.280  0.77951
```

```
## rpo.l18   -0.0924757  0.0676432  -1.367  0.17254
## dprod.l19 -0.0058374  0.0117651  -0.496  0.62012
## rea.l19    0.0520030  0.0794447   0.655  0.51320
## rpo.l19    0.1454712  0.0679470   2.141  0.03303 *
## dprod.l20  0.0001256  0.0116145   0.011  0.99138
## rea.l20   -0.0683630  0.0806045  -0.848  0.39700
## rpo.l20   -0.1176956  0.0682048  -1.726  0.08538 .
## dprod.l21  0.0052453  0.0115351   0.455  0.64961
## rea.l21   -0.0904455  0.0818796  -1.105  0.27015
## rpo.l21    0.0399443  0.0684944   0.583  0.56018
## dprod.l22 -0.0129843  0.0112824  -1.151  0.25065
## rea.l22    0.1569282  0.0816759   1.921  0.05557 .
## rpo.l22    0.0825970  0.0687133   1.202  0.23023
## dprod.l23  0.0026010  0.0112677   0.231  0.81759
## rea.l23    0.0697901  0.0817514   0.854  0.39391
## rpo.l23   -0.1443374  0.0654578  -2.205  0.02816 *
## dprod.l24 -0.0094767  0.0111440  -0.850  0.39574
## rea.l24   -0.1147715  0.0527778  -2.175  0.03039 *
## rpo.l24    0.0761649  0.0373513   2.039  0.04225 *
## const     -0.0013919  0.2279209  -0.006  0.99513
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##
## Residual standard error: 4.059 on 322 degrees of freedom
## Multiple R-Squared: 0.9674,  Adjusted R-squared: 0.9601
## F-statistic: 132.7 on 72 and 322 DF,  p-value: < 2.2e-16
##
##
## Estimation results for equation rpo:
## ====================================
## rpo = dprod.l1 + rea.l1 + rpo.l1 + dprod.l2 + rea.l2 + rpo.l2 + dprod.l3 + rea.l3 + rpo.l3 + dprod.l4
##
##             Estimate Std. Error t value Pr(>|t|)
## dprod.l1   0.0120772  0.0176663   0.684   0.4947
## rea.l1     0.0488382  0.0809140   0.604   0.5465
## rpo.l1     1.4207544  0.0557623  25.479  < 2e-16 ***
## dprod.l2  -0.0325787  0.0175689  -1.854   0.0646 .
## rea.l2     0.0098230  0.1258392   0.078   0.9378
## rpo.l2    -0.5808481  0.0962852  -6.033 4.43e-09 ***
## dprod.l3   0.0091117  0.0178012   0.512   0.6091
## rea.l3    -0.1362833  0.1238051  -1.101   0.2718
## rpo.l3     0.1630096  0.1021351   1.596   0.1115
## dprod.l4  -0.0162194  0.0184331  -0.880   0.3796
## rea.l4     0.1950945  0.1230677   1.585   0.1139
## rpo.l4    -0.0932126  0.1018650  -0.915   0.3608
## dprod.l5  -0.0027545  0.0179786  -0.153   0.8783
## rea.l5    -0.1473567  0.1224260  -1.204   0.2296
## rpo.l5     0.1140119  0.1005745   1.134   0.2578
## dprod.l6   0.0053810  0.0182225   0.295   0.7680
## rea.l6     0.0883678  0.1214285   0.728   0.4673
## rpo.l6    -0.1111048  0.1010518  -1.099   0.2724
## dprod.l7   0.0056362  0.0179729   0.314   0.7540
## rea.l7     0.0521672  0.1211793   0.430   0.6671
```

```
## rpo.l7     0.1019101  0.1012216   1.007   0.3148
## dprod.l8   0.0084877  0.0179228   0.474   0.6361
## rea.l8     0.0137240  0.1216113   0.113   0.9102
## rpo.l8    -0.0606556  0.1010077  -0.601   0.5486
## dprod.l9   0.0030528  0.0178749   0.171   0.8645
## rea.l9    -0.0061720  0.1203947  -0.051   0.9591
## rpo.l9    -0.0423865  0.1012968  -0.418   0.6759
## dprod.l10  0.0068309  0.0175422   0.389   0.6972
## rea.l10   -0.1313809  0.1178883  -1.114   0.2659
## rpo.l10    0.1491986  0.1013419   1.472   0.1419
## dprod.l11  0.0053928  0.0175153   0.308   0.7584
## rea.l11    0.0703464  0.1167088   0.603   0.5471
## rpo.l11   -0.0484895  0.1022411  -0.474   0.6356
## dprod.l12  0.0229968  0.0175852   1.308   0.1919
## rea.l12    0.0757939  0.1157798   0.655   0.5132
## rpo.l12   -0.0635448  0.1037431  -0.613   0.5406
## dprod.l13  0.0005881  0.0177173   0.033   0.9735
## rea.l13   -0.1524936  0.1150767  -1.325   0.1861
## rpo.l13   -0.1049181  0.1032714  -1.016   0.3104
## dprod.l14  0.0097894  0.0177503   0.552   0.5817
## rea.l14   -0.0014917  0.1156170  -0.013   0.9897
## rpo.l14    0.1586517  0.1001372   1.584   0.1141
## dprod.l15 -0.0104785  0.0177036  -0.592   0.5543
## rea.l15    0.0665518  0.1160079   0.574   0.5666
## rpo.l15   -0.1134519  0.1000639  -1.134   0.2577
## dprod.l16 -0.0034378  0.0179721  -0.191   0.8484
## rea.l16    0.0008143  0.1161235   0.007   0.9944
## rpo.l16    0.0185167  0.1002159   0.185   0.8535
## dprod.l17  0.0193144  0.0175937   1.098   0.2731
## rea.l17    0.1435874  0.1161632   1.236   0.2173
## rpo.l17    0.0886784  0.1000003   0.887   0.3759
## dprod.l18  0.0210960  0.0174032   1.212   0.2263
## rea.l18   -0.2183100  0.1163028  -1.877   0.0614 .
## rpo.l18   -0.0242869  0.0995644  -0.244   0.8074
## dprod.l19 -0.0155282  0.0173171  -0.897   0.3705
## rea.l19    0.0222532  0.1169350   0.190   0.8492
## rpo.l19   -0.0245805  0.1000115  -0.246   0.8060
## dprod.l20 -0.0101667  0.0170954  -0.595   0.5525
## rea.l20    0.2401303  0.1186421   2.024   0.0438 *
## rpo.l20    0.1271540  0.1003909   1.267   0.2062
## dprod.l21  0.0187847  0.0169786   1.106   0.2694
## rea.l21   -0.2134482  0.1205189  -1.771   0.0775 .
## rpo.l21   -0.1910191  0.1008173  -1.895   0.0590 .
## dprod.l22  0.0156614  0.0166066   0.943   0.3463
## rea.l22    0.1139135  0.1202192   0.948   0.3441
## rpo.l22    0.1957348  0.1011395   1.935   0.0538 .
## dprod.l23  0.0098688  0.0165850   0.595   0.5522
## rea.l23   -0.0477602  0.1203303  -0.397   0.6917
## rpo.l23   -0.2328022  0.0963476  -2.416   0.0162 *
## dprod.l24 -0.0077454  0.0164029  -0.472   0.6371
## rea.l24   -0.0054986  0.0776839  -0.071   0.9436
## rpo.l24    0.1362206  0.0549775   2.478   0.0137 *
## const      0.3457830  0.3354778   1.031   0.3034
## ---
```

```
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##
## Residual standard error: 5.975 on 322 degrees of freedom
## Multiple R-Squared: 0.9863,  Adjusted R-squared: 0.9832
## F-statistic: 321.3 on 72 and 322 DF,  p-value: < 2.2e-16
##
##
##
## Covariance matrix of residuals:
##         dprod    rea    rpo
## dprod 351.186  1.378 -8.259
## rea     1.378 16.479  1.881
## rpo    -8.259  1.881 35.702
##
## Correlation matrix of residuals:
##          dprod     rea      rpo
## dprod  1.00000 0.01811 -0.07376
## rea    0.01811 1.00000  0.07756
## rpo   -0.07376 0.07756  1.00000
```

Granger causality test

Granger causality tests whether lagged X improves prediction of Y beyond Y's own lags. Bivariate tests use pairs; multivariate tests condition on the full system.

```
# Bivariate Granger Causality
grangertest(data.oil$rea,data.oil$dprod,24)  # Does dprod → rea?
```

```
## Granger causality test
##
## Model 1: data.oil$dprod ~ Lags(data.oil$dprod, 1:24) + Lags(data.oil$rea, 1:24)
## Model 2: data.oil$dprod ~ Lags(data.oil$dprod, 1:24)
##   Res.Df  Df      F   Pr(>F)
## 1    346
## 2    370 -24 1.9925 0.004199 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
grangertest(data.oil$rpo,data.oil$dprod,24)  # Does dprod → rpo?
```

```
## Granger causality test
##
## Model 1: data.oil$dprod ~ Lags(data.oil$dprod, 1:24) + Lags(data.oil$rpo, 1:24)
## Model 2: data.oil$dprod ~ Lags(data.oil$dprod, 1:24)
##   Res.Df  Df      F Pr(>F)
## 1    346
## 2    370 -24 0.8022 0.7344
```

```
grangertest(data.oil$dprod,data.oil$rea,24)  # Does rea → dprod?
```

```
## Granger causality test
##
## Model 1: data.oil$rea ~ Lags(data.oil$rea, 1:24) + Lags(data.oil$dprod, 1:24)
## Model 2: data.oil$rea ~ Lags(data.oil$rea, 1:24)
##   Res.Df  Df      F  Pr(>F)
```

```
## 1    346
## 2    370 -24 1.5388 0.05262 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
grangertest(data.oil$rpo,data.oil$rea,24)    # Does rea → rpo?
```

```
## Granger causality test
##
## Model 1: data.oil$rea ~ Lags(data.oil$rea, 1:24) + Lags(data.oil$rpo, 1:24)
## Model 2: data.oil$rea ~ Lags(data.oil$rea, 1:24)
##   Res.Df  Df      F  Pr(>F)
## 1    346
## 2    370 -24 1.4967 0.06482 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
grangertest(data.oil$rea,data.oil$rpo,24)    # Does rpo → rea?
```

```
## Granger causality test
##
## Model 1: data.oil$rpo ~ Lags(data.oil$rpo, 1:24) + Lags(data.oil$rea, 1:24)
## Model 2: data.oil$rpo ~ Lags(data.oil$rpo, 1:24)
##   Res.Df  Df      F Pr(>F)
## 1    346
## 2    370 -24 1.1431 0.2938
```

```
grangertest(data.oil$dprod,data.oil$rpo,24)  # Does rpo → dprod?
```

```
## Granger causality test
##
## Model 1: data.oil$rpo ~ Lags(data.oil$rpo, 1:24) + Lags(data.oil$dprod, 1:24)
## Model 2: data.oil$rpo ~ Lags(data.oil$rpo, 1:24)
##   Res.Df  Df      F Pr(>F)
## 1    346
## 2    370 -24 0.6026 0.9315
```

```
# Multivariate Granger Causality (conditions on the whole VAR)
causality(var.oil, cause = "dprod", vcov. = sandwich)  # H0: lags of dprod don't help explain others
```

```
## $Granger
##
##  Granger causality H0: dprod do not Granger-cause rea rpo
##
## data:  VAR object var.oil
## F-Test = 1.5861, df1 = 48, df2 = 966, p-value = 0.007595
##
##
## $Instant
##
##  H0: No instantaneous causality between: dprod and rea rpo
##
## data:  VAR object var.oil
## Chi-squared = 2.3604, df = 2, p-value = 0.3072
```

```
causality(var.oil, cause = "rea",   vcov. = sandwich)
```

```
## $Granger
```

```
##
##  Granger causality H0: rea do not Granger-cause dprod rpo
##
## data:  VAR object var.oil
## F-Test = 2.1454, df1 = 48, df2 = 966, p-value = 1.535e-05
##
##
## $Instant
##
##  H0: No instantaneous causality between: rea and dprod rpo
##
## data:  VAR object var.oil
## Chi-squared = 2.5847, df = 2, p-value = 0.2746
```

```
causality(var.oil, cause = "rpo",   vcov. = sandwich)
```

```
## $Granger
##
##  Granger causality H0: rpo do not Granger-cause dprod rea
##
## data:  VAR object var.oil
## F-Test = 2.0971, df1 = 48, df2 = 966, p-value = 2.776e-05
##
##
## $Instant
##
##  H0: No instantaneous causality between: rpo and dprod rea
##
## data:  VAR object var.oil
## Chi-squared = 4.5552, df = 2, p-value = 0.1025
```

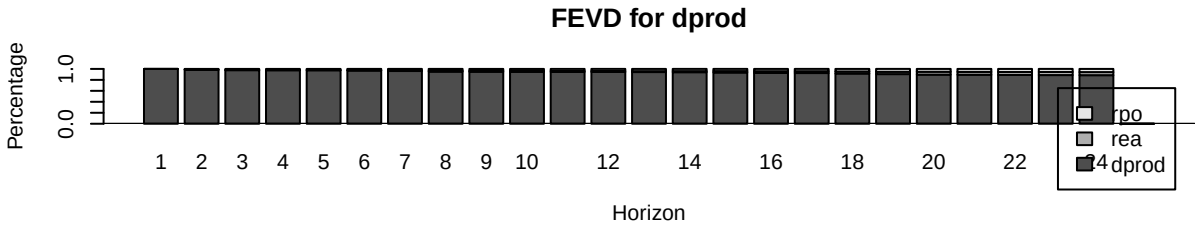
Global demand (rea) helps predict future oil production (dprod) at conventional levels (p 0.004), while other bivariate links are weak or insignificant. However, once we condition on the full VAR (multivariate Granger tests), all three variables—production, demand, and the real oil price—contain predictive information for the others (p<1%), indicating meaningful feedback among them. Instantaneous-causality p-values are not significant, so there's no strong evidence of same-period effects after controlling for lags, which is consistent with using a recursive (Cholesky) identification.

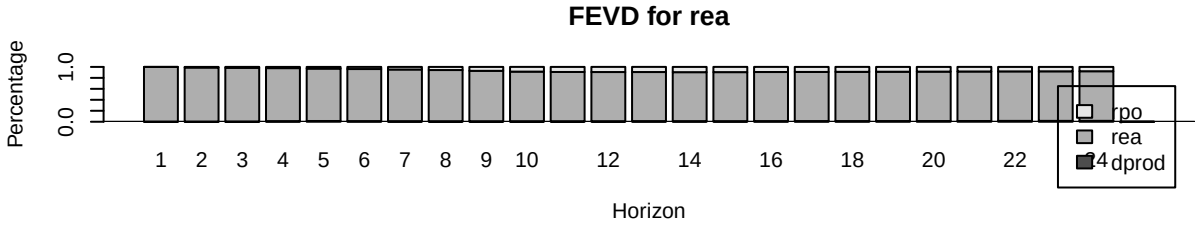
In plain language: demand, supply, and prices evolve jointly, each helping to forecast the others over time, with demand's predictive role most evident in the pairwise tests.

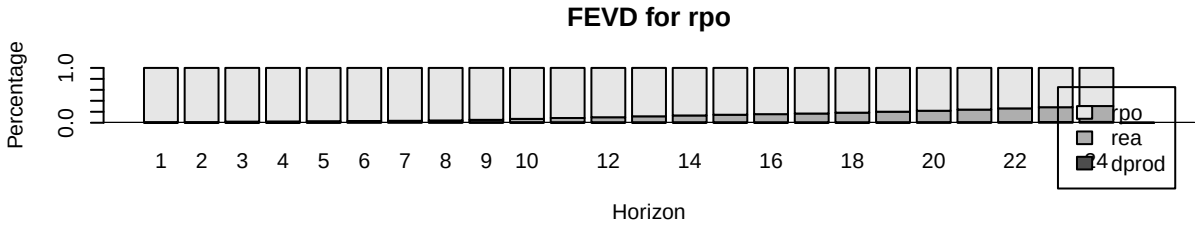
Forcase Error Variance decomposition (FEVD)

Forecast Error Variance Decomposition attributes the h-step forecast error variance of each variable to shocks from all variables under the identification scheme.

```
var.oil.fevd <- fevd(var.oil, n.ahead = 24)   # Share of forecast error variance by shock at horizons
plot(var.oil.fevd)
```








```
# remark: dprod shocks explain only small fractions of "rea" and "rpo"
```

Impulse responses function (IRF)

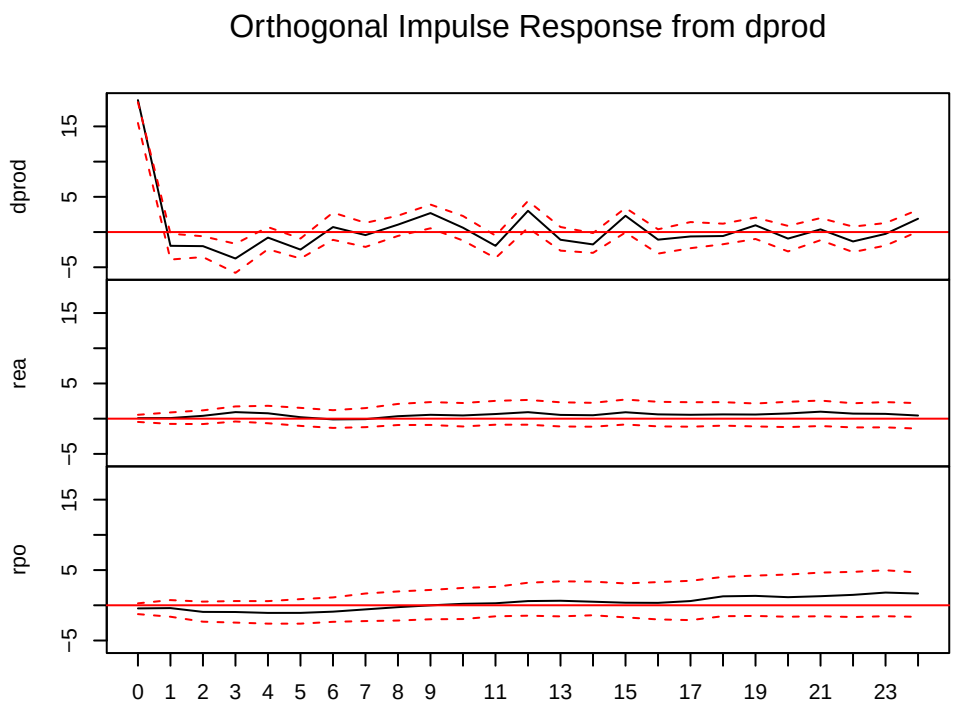
IRFs trace the effect of a one-unit structural shock over time. With recursive (Cholesky) identification, the variable order implies contemporaneous causal ordering.

```
# Impulse responses: recursive order = default (p,u,r)
irf.dprod <- irf(var.oil, n.ahead = 24,
                 ci = 0.95,
                 impulse = "dprod",
                 response = c("dprod", "rea", "rpo"))  # Responses to a supply shock
head(irf.dprod$irf)
```

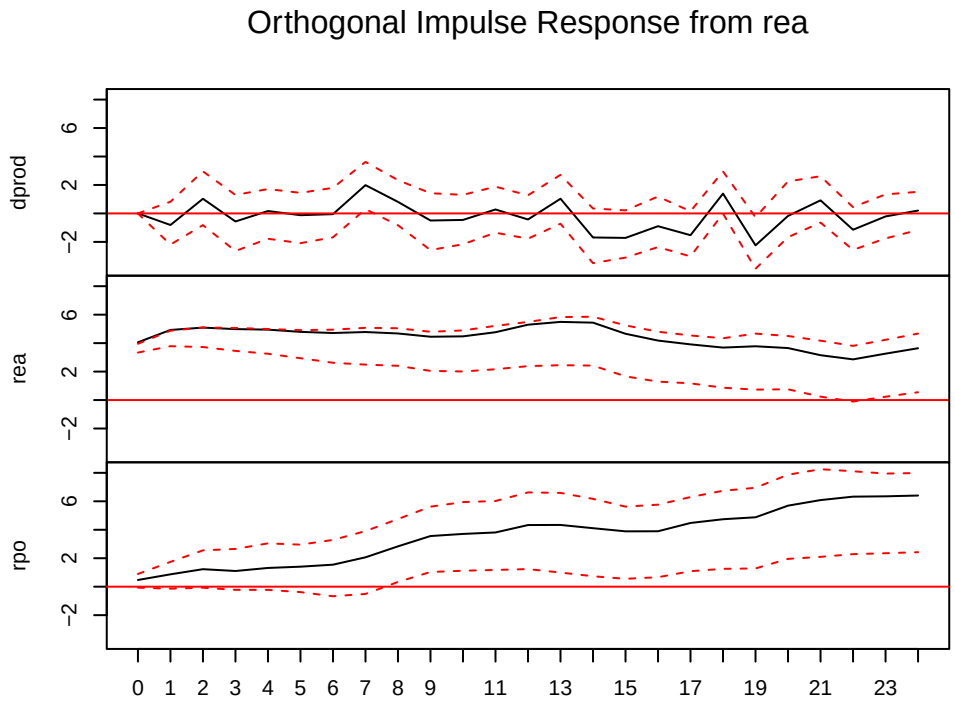
```
## $dprod
##            dprod         rea          rpo
##  [1,] 18.7399568  0.07352518 -0.440703241
##  [2,] -1.9548097  0.08897573 -0.396214851
##  [3,] -2.0000525  0.40661505 -0.936006309
##  [4,] -3.7575987  0.93152203 -0.950538115
##  [5,] -0.7886691  0.76555563 -1.080588663
##  [6,] -2.4942708  0.20789617 -1.076620532
##  [7,]  0.7117749 -0.10599017 -0.902430935
##  [8,] -0.4328097 -0.06941784 -0.581652641
##  [9,]  1.0507161  0.35433632 -0.264823585
## [10,]  2.6941924  0.54849671 -0.007272787
## [11,]  0.6150223  0.45869496  0.210417242
## [12,] -1.9538706  0.65745381  0.290185223
```

```
## [13,]  3.0161105  0.92540439  0.600487117
## [14,] -1.0937876  0.53530530  0.647083558
## [15,] -1.7583682  0.48715486  0.510901723
## [16,]  2.3069889  0.91194729  0.366270212
## [17,] -1.0803190  0.60947013  0.346710885
## [18,] -0.6254037  0.55238031  0.604759364
## [19,] -0.5652865  0.60506682  1.272995220
## [20,]  0.9597093  0.58648718  1.339330948
## [21,] -0.9290587  0.74366857  1.154081395
## [22,]  0.3831415  0.99364600  1.293565168
## [23,] -1.3337586  0.71849472  1.481884229
## [24,] -0.2572264  0.68113739  1.806696162
## [25,]  1.8912140  0.44678615  1.673072433
```

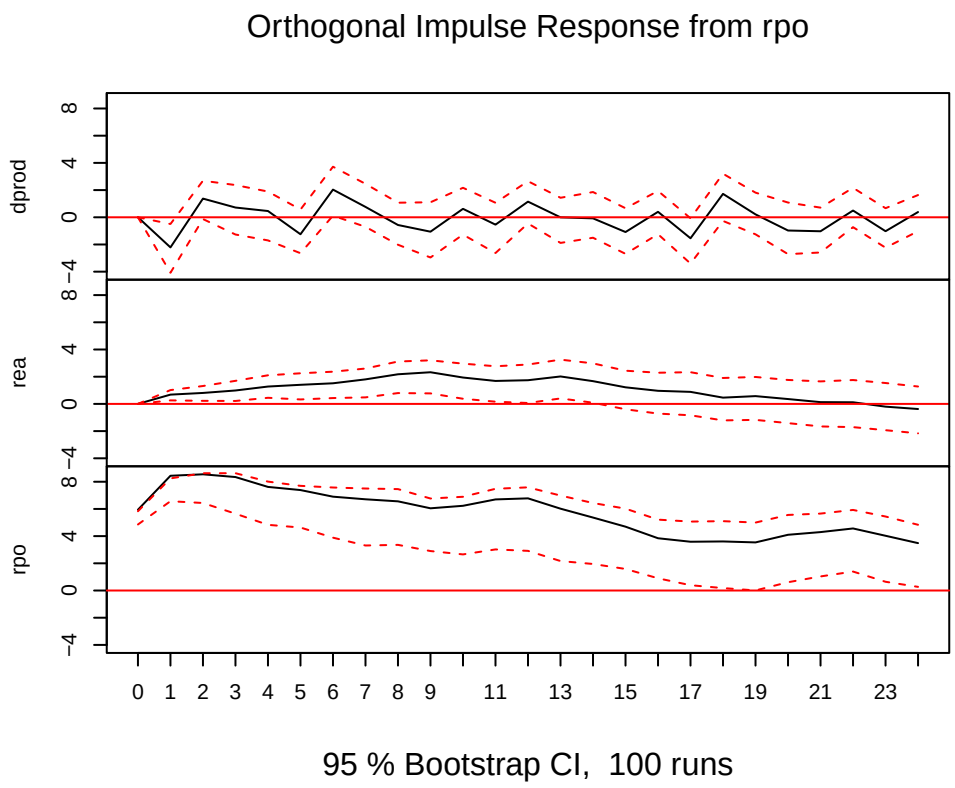
```
plot(irf.dprod)  # Bands reflect 95% CI
```




```
irf.rea  <- irf(var.oil, n.ahead = 24,
                ci = 0.95,
                impulse = "rea",
                response = c("dprod", "rea", "rpo"))
plot(irf.rea)
```



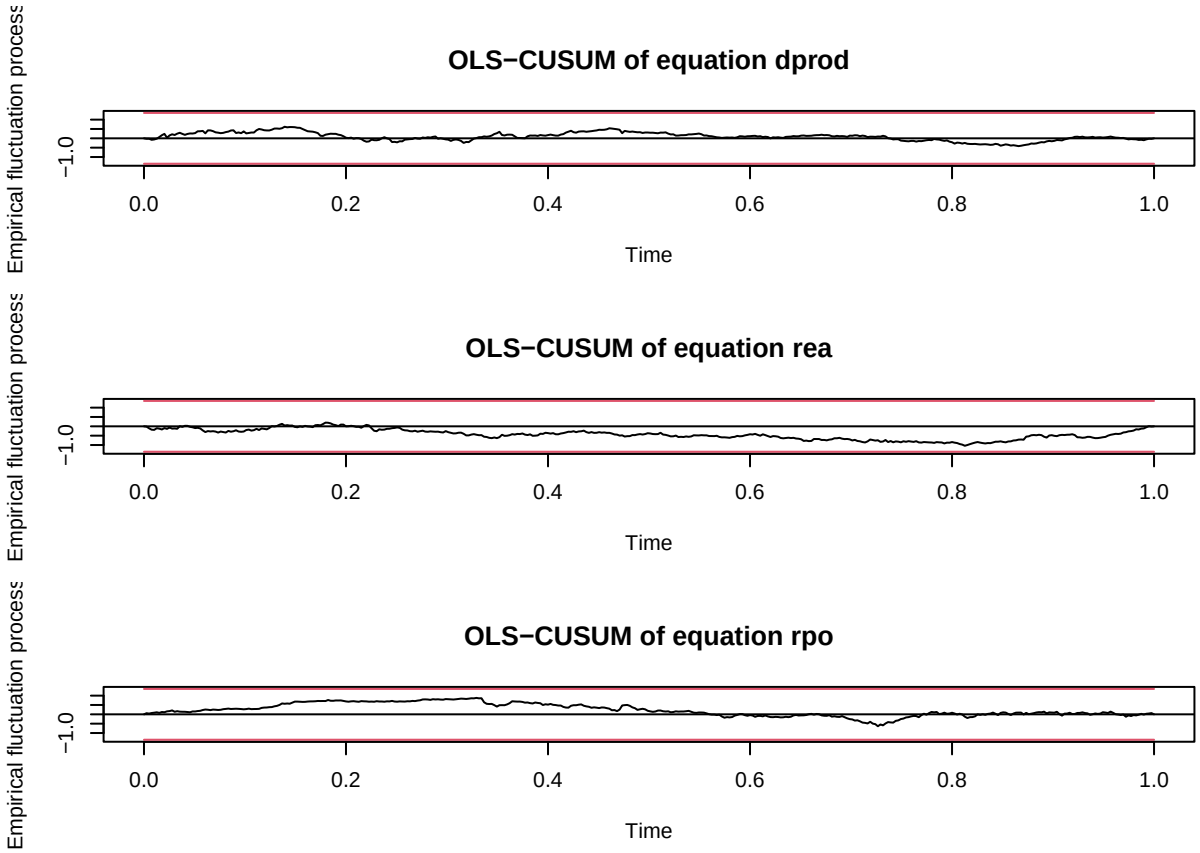

```
irf.rpo  <- irf(var.oil, n.ahead = 24,
                ci = 0.95,
                impulse = "rpo",
                response = c("dprod", "rea", "rpo"))
plot(irf.rpo)
```



Diagnostic tests

We check stability (CUSUM), serial correlation (Portmanteau), conditional heteroskedasticity (ARCH), and normality to validate VAR assumptions.

```
# Stability test
var.oil.break <- stability(var.oil, type = "OLS-CUSUM")  # Detects parameter instability
plot(var.oil.break)
```




```
# Residual test
resid1 <- serial.test(var.oil, lags.pt = 24)  # Portmanteau test for autocorrelation
resid1
```

```
##
##  Portmanteau Test (asymptotic)
##
## data:  Residuals of VAR object var.oil
## Chi-squared = 59.797, df = 0, p-value < 2.2e-16
```

```
# arch test
arch1 <- arch.test(var.oil, lags.single = 24, lags.multi = 24, multivariate.only = F)
arch1
```

```
## $dprod
##
##  ARCH test (univariate)
##
## data:  Residual of dprod equation
## Chi-squared = 37.832, df = 24, p-value = 0.03607
##
##
## $rea
##
##  ARCH test (univariate)
##
## data:  Residual of rea equation
## Chi-squared = 25.08, df = 24, p-value = 0.4014
```

```
##
##
## $rpo
##
##  ARCH test (univariate)
##
## data:  Residual of rpo equation
## Chi-squared = 27.481, df = 24, p-value = 0.2825
##
##
##
##  ARCH (multivariate)
##
## data:  Residuals of VAR object var.oil
## Chi-squared = 848.49, df = 864, p-value = 0.6403
```

```
arch2 <- arch.test(var.oil, lags.single = 24, lags.multi = 24, multivariate.only = T)
arch2
```

```
##
##  ARCH (multivariate)
##
## data:  Residuals of VAR object var.oil
## Chi-squared = 848.49, df = 864, p-value = 0.6403
```

```
# Normality test
norm1 <- normality.test(var.oil, multivariate.only = F)  # Jarque-Bera type
norm1
```

```
## $dprod
##
##  JB-Test (univariate)
##
## data:  Residual of dprod equation
## Chi-squared = 144.96, df = 2, p-value < 2.2e-16
##
##
## $rea
##
##  JB-Test (univariate)
##
## data:  Residual of rea equation
## Chi-squared = 4.5652, df = 2, p-value = 0.102
##
##
## $rpo
##
##  JB-Test (univariate)
##
## data:  Residual of rpo equation
## Chi-squared = 84.191, df = 2, p-value < 2.2e-16
##
##
## $JB
##
##  JB-Test (multivariate)
```

```
##
## data:  Residuals of VAR object var.oil
## Chi-squared = 203.64, df = 6, p-value < 2.2e-16
##
##
## $Skewness
##
##  Skewness only (multivariate)
##
## data:  Residuals of VAR object var.oil
## Chi-squared = 22.362, df = 3, p-value = 5.485e-05
##
##
## $Kurtosis
##
##  Kurtosis only (multivariate)
##
## data:  Residuals of VAR object var.oil
## Chi-squared = 181.28, df = 3, p-value < 2.2e-16
```

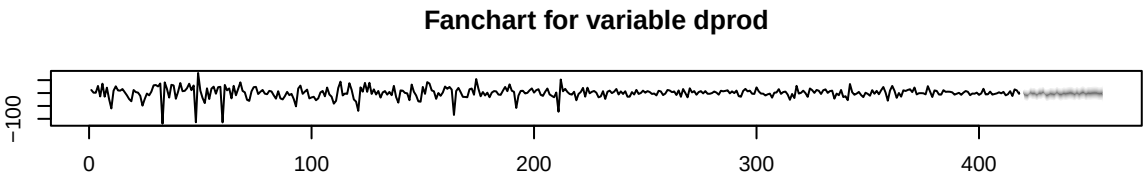
```
norm2 <- normality.test(var.oil, multivariate.only = T)
norm2
```

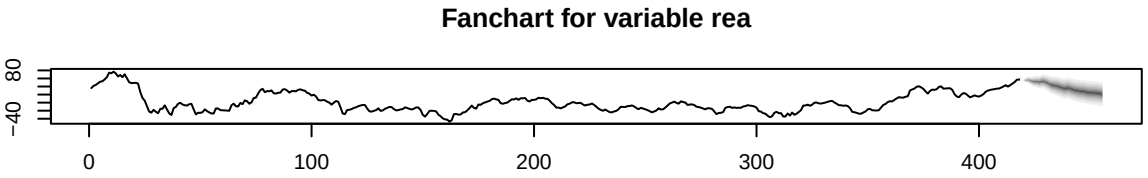
```
## $JB
##
##  JB-Test (multivariate)
##
## data:  Residuals of VAR object var.oil
## Chi-squared = 203.64, df = 6, p-value < 2.2e-16
##
##
## $Skewness
##
##  Skewness only (multivariate)
##
## data:  Residuals of VAR object var.oil
## Chi-squared = 22.362, df = 3, p-value = 5.485e-05
##
##
## $Kurtosis
##
##  Kurtosis only (multivariate)
##
## data:  Residuals of VAR object var.oil
## Chi-squared = 181.28, df = 3, p-value < 2.2e-16
```

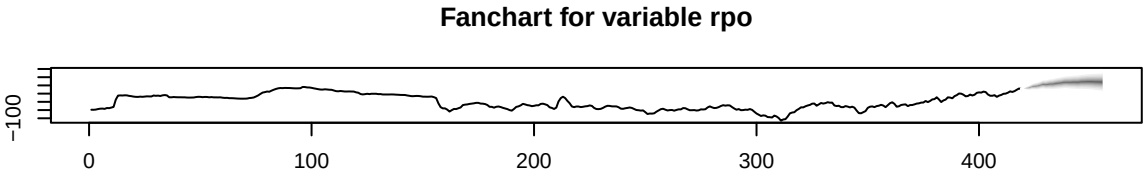
forecast

We produce multi-step VAR forecasts and visualize uncertainty via fan charts (approximate ±k intervals).

```
h <- 36   # forecast horizon

var.oil.pred <- predict(var.oil, n.ahead = h, ci = 0.95) # Multi-step VAR forecast
#x11(); par(mai=rep(0.5, 4)); fanchart(var.oil.pred)  #open new graph window; plot margin=0.5in all sid
fanchart(var.oil.pred)
```








```
# example to compare: https://fred.stlouisfed.org/series/POILBREUSDM
```

Obtain the structural shocks

Structural shocks u are recovered from reduced-form residuals via = B0 u, where B0 is lower-triangular under recursive identification. We recover B0 from contemporaneous IRFs.

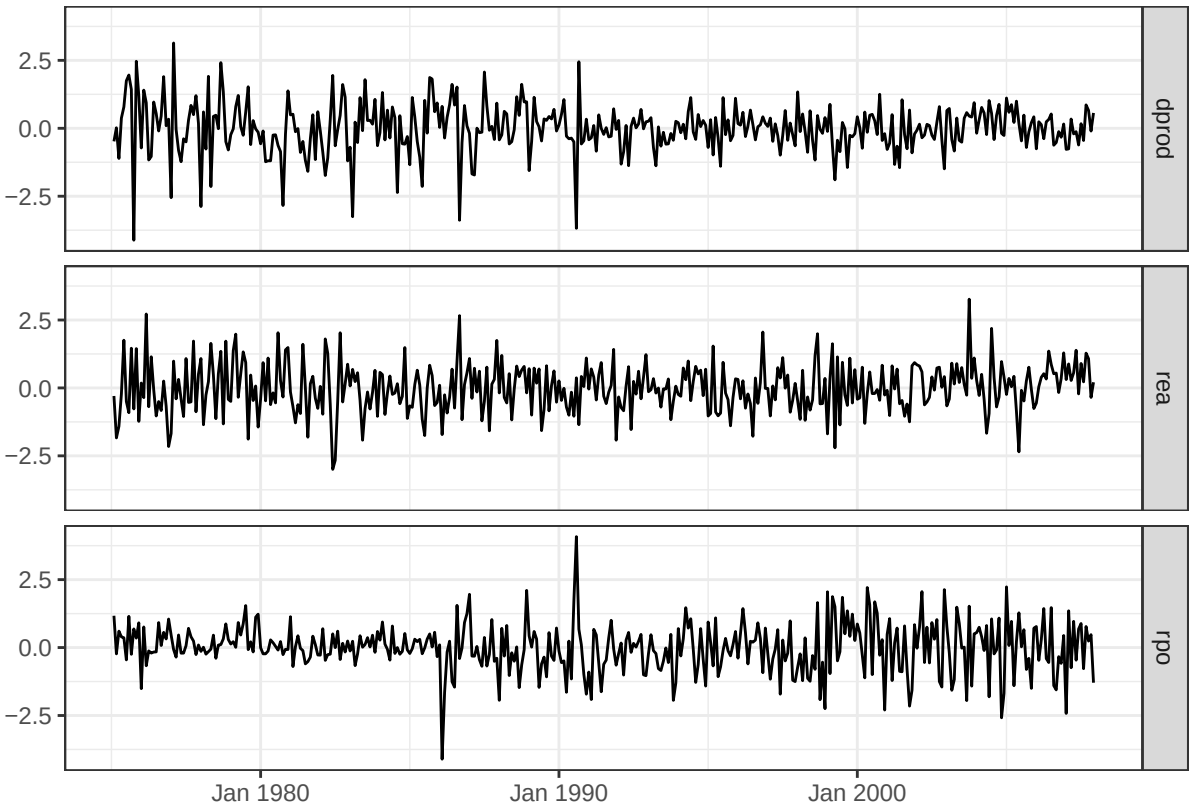
```
# Save the reduced-form residuals
n <- length(data.oil[,1])
# nlag is lag order

# Extract the residuals
res <- residuals(var.oil)   # sample size - lag order
head(res)
```

```
##         dprod        rea       rpo
## 1  -9.0246211 -1.2372804  7.029941
## 2   0.3460242 -7.4628322 -2.246519
## 3 -20.7820606 -5.8129124  3.411275
## 4   7.0424487 -0.8742261  2.069175
## 5  14.7677591  7.1839361  2.625407
## 6  32.7883931 -1.8985810 -3.696447
```

```
tail(res)
```

```
##         dprod        rea       rpo
## 390  3.162872  3.6672079  5.606938
## 391 -8.414511  0.8747278 -4.304708
## 392 16.138898  5.2595987  4.829112
## 393 12.337843  4.4162935  1.628370
## 394 -1.894184 -1.4229373  2.700905
## 395 10.587012  0.8855633 -7.833780
```

```r
res <- xts(res,order.by = index(data.oil)[-seq(1,nlag)])

# Define the structural shocks from the recursive form's irf function
# Mapping between reduced-form shocks and standardized structural shocks:

# e1 = sig1*u1
# e2 = a*u1 + sig2*u2
# e3 = b*u1 + c*u2 + sig3*u3

# The structural VAR: (Lecture note p.33)
# y1 = lags + e1 = lags + sig1*u1
# y2 = lags + e2 = lags + a*u1 + sig2*u2
# y3 = lags + e3 = lags + b*u1 + c*u2 + sig3*u3

# So the contemporaneous IRF is given by {sig1,sig2,sig3,a,b,c}:
#  sig1 is the irf of u1 to y1; sig2 is irf of u2 to y2; sig3 is irf of u3 to y3
#  a is the irf of u1 to y2;  b is the irf of u1 to y3;  c is the irf of u2 to y3

sig1 <- irf.dprod$irf$dprod[1,1]  # contemporaneous IRF of y1 to u1 ( 1)
a    <- irf.dprod$irf$dprod[1,2]  # contemporaneous effect of u1 on y2 (a)
b    <- irf.dprod$irf$dprod[1,3]  # contemporaneous effect of u1 on y3 (b)

sig2 <- irf.rea$irf$rea[1,2]      # 2
c    <- irf.rea$irf$rea[1,3]      # contemporaneous effect of u2 on y3 (c)

sig3 <- irf.rpo$irf$rpo[1,3]      # 3

# Structural shocks: standardized (solve   = B0 u for u)
u1 <- res[,1]/sig1                          # u1 =  1/ 1
u2 <- (res[,2] - a*u1)/sig2                 # u2 = ( 2 - a u1)/ 2
u3 <- (res[,3] - b*u1 - c*u2)/sig3          # u3 = ( 3 - b u1 - c u2)/ 3

u <- cbind(u1,u2,u3)
autoplot(u) + xlab("") + theme_bw()
```



- Alternative way to recover structural shocks using Cholesky decomposition Direct Cholesky: $\Sigma = PP$ with P lower-triangular. Then u = P^{-1} . We adjust degrees of freedom when estimating Σ.

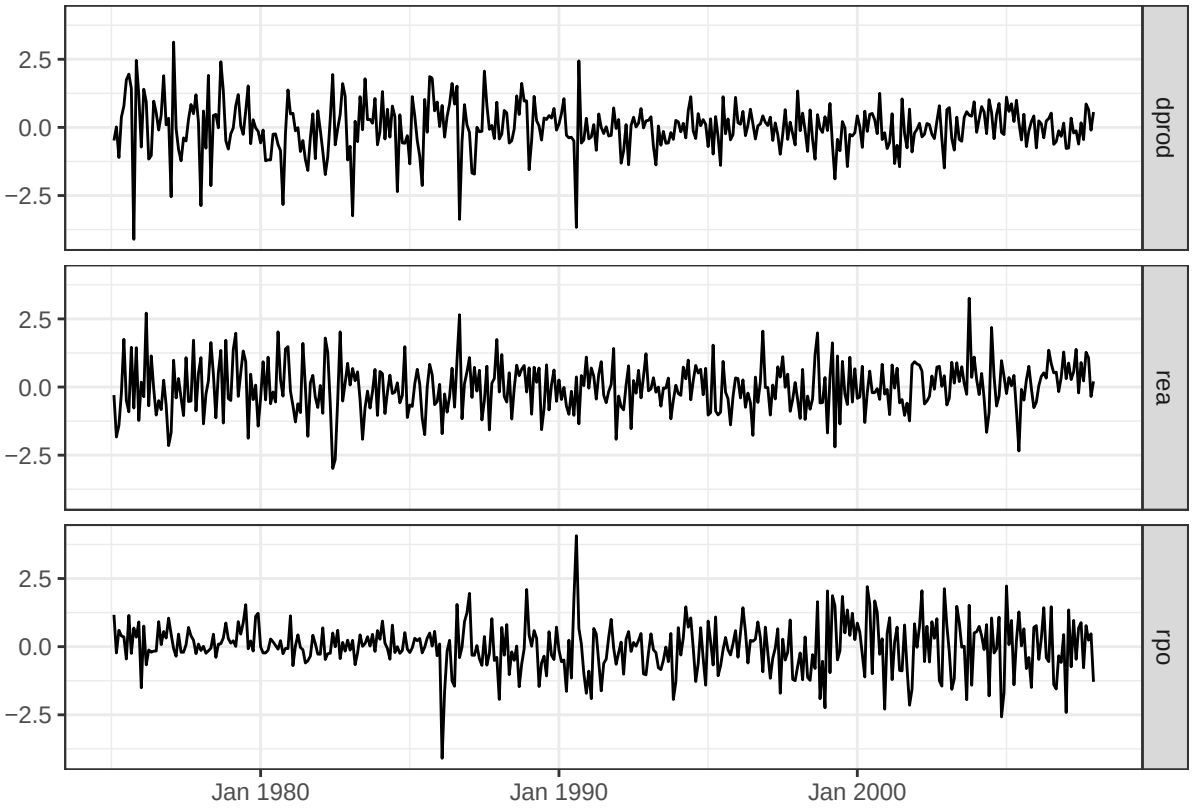
```
N <- dim(res)[1]
# n is the total sample size n = N+24

res.var_cov_mat <-  var(res)*(N-1)/(N-25*3)  # DoF-adjusted Σ for reduced-form shocks
#variance-covariance matrix of reduced-form shocks (omega)
#Note: The "var()" function computes sample variance which by defaults assumes the degree of freedom is

res.chol_mat <- t(chol(res.var_cov_mat)) #lower-triangular matrix of Cholesky decomposition
                                         #t() means transpose of a matrix

# Comparison
res.chol_mat%*%t(res.chol_mat);res.var_cov_mat # Verify Σ  P P¸
```

```
##            dprod       rea       rpo
## dprod 353.380894  1.386470 -8.310377
## rea     1.386470 16.581856  1.892983
## rpo    -8.310377  1.892983 35.924646
```

```
##            dprod       rea       rpo
## dprod 353.380894  1.386470 -8.310377
## rea     1.386470 16.581856  1.892983
## rpo    -8.310377  1.892983 35.924646
```

```
u_chol <- solve(res.chol_mat) %*% t(res) #use "solve" to compute the inverse of lower-triangular matrix

u_chol <- t(u_chol)# Conform dimensions (time × shocks)
```

```
#Note: The structural shocks obtained by Cholesky decomposition (u_chol) are close to those structural
data <- xts(u_chol[,1:3], order.by = index(res), unique = FALSE, tzone = "")
autoplot(data) + xlab("") + theme_bw()
```



Implications of structural shocks for macro aggregates

We aggregate monthly shocks to quarter averages and run distributed-lag OLS to estimate cumulative effects on real GDP and inflation.

```
qlag <- 12   # number of lagged shocks in the OLS regression

# define quarterly average of structural shocks
ep.q <- endpoints(u,'quarters') # Quarter endpoints in xts index
u.q  <- period.apply(u, INDEX=ep.q, FUN = mean, na.rm=TRUE) # quarter average at quarter-end
# period.apply(): Apply Function Over Specified Interval

head(u.q)
```

```
##                dprod         rea        rpo
## Mar 1975 -0.23155328 -1.06757758  0.4701941
## Jun 1975  0.01828797  0.04046925  0.4530098
## Sep 1975  1.71401026  0.01694409  0.1497185
## Dec 1975 -0.15665182 -0.18572207  0.6480297
## Mar 1976  0.53992624  0.84757556 -0.4737486
## Jun 1976 -0.41187668  0.12853826 -0.1630177
```

```
tail(u.q)
```

```
##                dprod       rea        rpo
```

```
## Sep 2006 -0.20656041 0.6440534 -0.4953204
## Dec 2006 -0.09800946 0.4893144 -0.1575383
## Mar 2007 -0.39736770 0.4884208 -0.6042075
## Jun 2007 -0.31454077 0.5747174  0.3944514
## Sep 2007  0.19365500 0.8014633  0.2948173
## Dec 2007  0.37407896 0.3117878 -0.1936515
```

Subset the real GDP series: 1975 - 2007

We fetch GDPC1 from FRED, convert to quarterly, align dates, and keep 1975–2007 for comparability with oil data.

```
# Load Real GDP Data from FRED
# Fetch US Real GDP data (GDPC1) from FRED
getSymbols("GDPC1", src = "FRED", from = "1970-01-01", to = "2010-12-31")
```

```
## [1] "GDPC1"
```

```
y <- GDPC1
# Convert to quarterly data if needed and subset to 1975-2007
y <- to.quarterly(y, name = "rgdp") # OHLC format; 4th column is Close

rgdp <- y["1975/2007"][,4] # Keep 'Close' column as level for quarter-end
names(rgdp) <- "rgdp"
head(rgdp)
```

```
##                rgdp
## 1975-01-01 5957.035
## 1975-04-01 5999.610
## 1975-07-01 6102.326
## 1975-10-01 6184.530
## 1976-01-01 6323.649
## 1976-04-01 6370.025
```

```
tail(rgdp)
```

```
##                rgdp
## 2006-07-01 16420.74
## 2006-10-01 16561.87
## 2007-01-01 16611.69
## 2007-04-01 16713.31
## 2007-07-01 16809.59
## 2007-10-01 16915.19
```

```
index(u.q) <- index(rgdp)  # use quarter-start to denote timing
data.q <- cbind(rgdp, u.q)
data.q <- as.ts(data.q)    # package dynlm works only with "ts" objects
head(data.q)
```

```
## Time Series:
## Start = 1
## End = 6
## Frequency = 1
##       rgdp       dprod         rea        rpo
## 1 5957.035 -0.23155328 -1.06757758  0.4701941
## 2 5999.610  0.01828797  0.04046925  0.4530098
```

```
## 3 6102.326  1.71401026  0.01694409  0.1497185
## 4 6184.530 -0.15665182 -0.18572207  0.6480297
## 5 6323.649  0.53992624  0.84757556 -0.4737486
## 6 6370.025 -0.41187668  0.12853826 -0.1630177
```

```
tail(data.q)
```

```
## Time Series:
## Start = 127
## End = 132
## Frequency = 1
##         rgdp       dprod       rea        rpo
## 127 16420.74 -0.20656041 0.6440534 -0.4953204
## 128 16561.87 -0.09800946 0.4893144 -0.1575383
## 129 16611.69 -0.39736770 0.4884208 -0.6042075
## 130 16713.31 -0.31454077 0.5747174  0.3944514
## 131 16809.59  0.19365500 0.8014633  0.2948173
## 132 16915.19  0.37407896 0.3117878 -0.1936515
```

OLS distributed-lags regression

The regression rgdp_t = + Σ_{k=0}^{12} _k shock_{t−k} + e_t. Cumulative IR = Σ_{j=0}^{h} _j.

```
# Distributed-lags regression: Predict current values of a dependent variable based on both the current

eq1.y <- dynlm(rgdp ~ L(dprod,0:qlag), data = data.q)  # rgdp on supply shock lags
eq2.y <- dynlm(rgdp ~ L(rea,0:qlag),   data = data.q)  # rgdp on global demand shock lags
eq3.y <- dynlm(rgdp ~ L(rpo,0:qlag),   data = data.q)  # rgdp on oil-specific demand shock lags

summary(eq1.y)       # first coefficient is intercept
```

```
##
## Time series regression with "ts" data:
## Start = 13, End = 132
##
## Call:
## dynlm(formula = rgdp ~ L(dprod, 0:qlag), data = data.q)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -4346.0 -2756.9  -207.7  3231.4  5939.5
##
## Coefficients:
##                     Estimate Std. Error t value Pr(>|t|)
## (Intercept)        11197.92     293.52  38.150   <2e-16 ***
## L(dprod, 0:qlag)0    369.05     613.16   0.602    0.549
## L(dprod, 0:qlag)1    132.30     616.69   0.215    0.831
## L(dprod, 0:qlag)2     80.52     617.59   0.130    0.897
## L(dprod, 0:qlag)3    290.89     607.83   0.479    0.633
## L(dprod, 0:qlag)4    349.76     612.42   0.571    0.569
## L(dprod, 0:qlag)5    228.02     610.71   0.373    0.710
## L(dprod, 0:qlag)6    219.83     608.58   0.361    0.719
## L(dprod, 0:qlag)7    293.68     605.82   0.485    0.629
## L(dprod, 0:qlag)8    173.02     598.14   0.289    0.773
## L(dprod, 0:qlag)9    222.34     595.63   0.373    0.710
```

```
## L(dprod, 0:qlag)10   -21.51     566.33  -0.038    0.970
## L(dprod, 0:qlag)11   -80.25     567.30  -0.141    0.888
## L(dprod, 0:qlag)12  -271.20     571.92  -0.474    0.636
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 3196 on 106 degrees of freedom
## Multiple R-squared:  0.01947,    Adjusted R-squared:  -0.1008
## F-statistic: 0.1619 on 13 and 106 DF,  p-value: 0.9996
```

Define cumulative impulse responses and confidence intervals

We compute cumulative sums of ˆ and their standard errors using the cumulative variance formula var(Σ ˆ) = 1 V 1 with cumulative summation.

```
# Cumulative impulse responses: Integrating the impulse response over time

b1  <- cumsum(coef(eq1.y)[2:(qlag+2)])   # The coefficients from t to t-12 (\nabla_h = Σ_{j=0}^{h} ^_j (
v1  <- vcov(eq1.y)[2:(qlag+2),2:(qlag+2)] # V = Var(^_{0..12}; vcov(): Variance-Covariance Matrix
```

```
# Remark: var(a+b+c) = var(a)+var(b)+var(c)+2cov(a,b)+2cov(b,c)+2cov(a,c)
# This is the cumsum of the rows and columns or the variance-covariance matrix

# Example of using apply() with cumsum
# m1 <- matrix(C<-(1:10),nrow=5, ncol=6)
# apply(m1, 2, cumsum)

vb1 <- apply(apply(v1, 2, cumsum), 1, cumsum) # Cumulative variance of cumulative sums
sb1 <- sqrt(diag(vb1))   # SE of the cumsum of slope coefficients

# To verify, we may compare
all.equal(sum(sum(v1[1:3,1:3])), vb1[3,3]) # Simple check for h=3 case
```

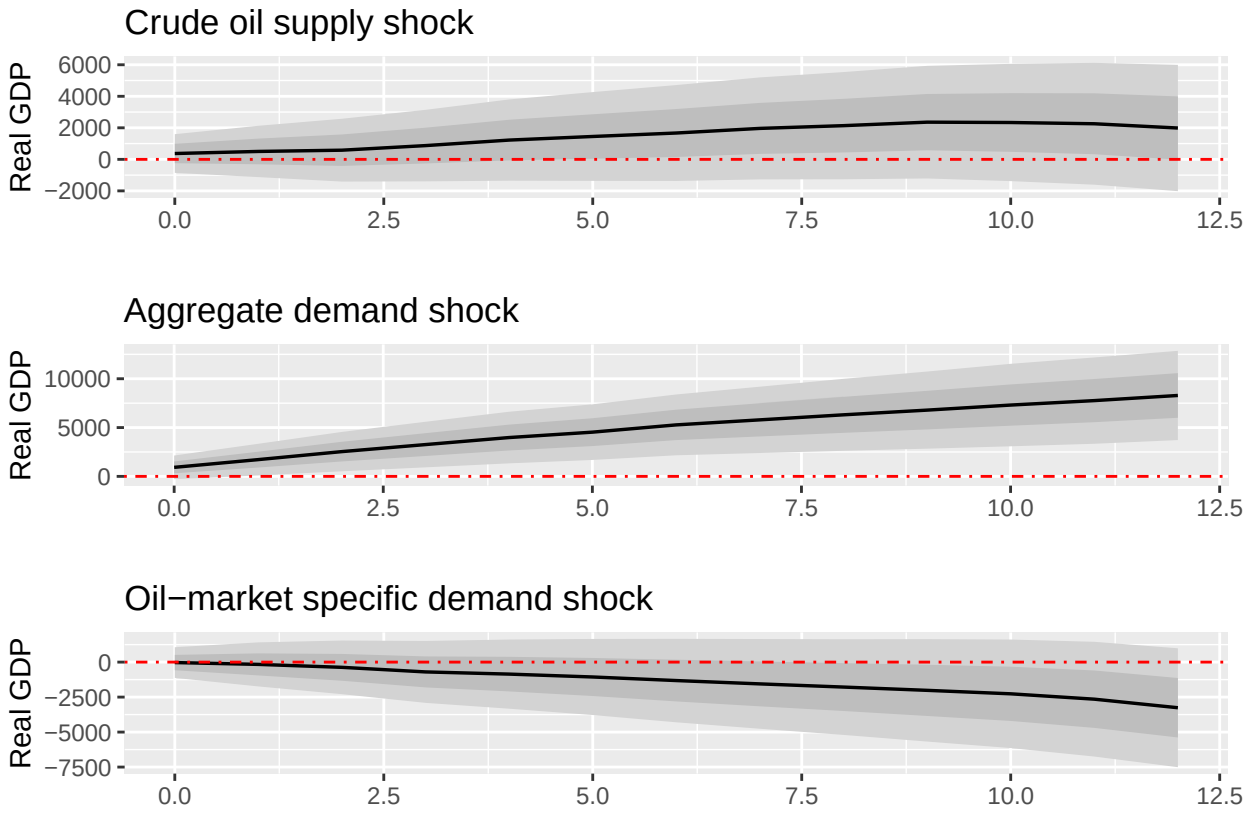
```
## [1] TRUE
```

```
# Plot cumulative impulse responses with +/- SE or 2SE bands
plot1 <- ggplot(as.data.frame(b1), aes(x = 0:qlag, y = b1)) +
  geom_ribbon(aes(ymin = b1 - 2*sb1, ymax = b1 + 2*sb1),
              fill = "lightgrey") +
  geom_ribbon(aes(ymin = b1 - sb1, ymax = b1 + sb1),
              fill = "grey") +
  geom_path(size = 0.6) +    # set the size of the time series line plot
  geom_hline(yintercept = 0, linetype = 4, colour = "red") +
  xlab("") +
  ylab("Real GDP") +
  ggtitle("Crude oil supply shock")
```

```
# similarly, we define rgdp's responses to "rea" and "rpo"
b2  <- cumsum(coef(eq2.y)[2:(qlag+2)])
v2  <- vcov(eq2.y)[2:(qlag+2),2:(qlag+2)]
vb2 <- apply(apply(v2, 2, cumsum), 1, cumsum)
sb2 <- sqrt(diag(vb2))   # SE of the cumsum of slope coefficients

b3  <- cumsum(coef(eq3.y)[2:(qlag+2)])
v3  <- vcov(eq3.y)[2:(qlag+2),2:(qlag+2)]
```

```r
vb3 <- apply(apply(v3, 2, cumsum), 1, cumsum)
sb3 <- sqrt(diag(vb3))   # SE of the cumsum of slope coefficients
```

```r
# plot cumulative impulse responses with +/- SE or 2SE bands
plot2 <- ggplot(as.data.frame(b2), aes(x = 0:qlag, y = b2)) +
  geom_ribbon(aes(ymin = b2 - 2*sb2, ymax = b2 + 2*sb2),
              fill = "lightgrey") +
  geom_ribbon(aes(ymin = b2 - sb2, ymax = b2 + sb2),
              fill = "grey") +
  geom_path(size = 0.6) +    # set the size of the time series line plot
  geom_hline(yintercept = 0, linetype = 4, colour = "red") +
  xlab("") +
  ylab("Real GDP") +
  ggtitle("Aggregate demand shock")

plot3 <- ggplot(as.data.frame(b3), aes(x = 0:qlag, y = b3)) +
  geom_ribbon(aes(ymin = b3 - 2*sb3, ymax = b3 + 2*sb3),
              fill = "lightgrey") +
  geom_ribbon(aes(ymin = b3 - sb3, ymax = b3 + sb3),
              fill = "grey") +
  geom_path(size = 0.6) +    # set the size of the time series line plot
  geom_hline(yintercept = 0, linetype = 4, colour = "red") +
  xlab("") +
  ylab("Real GDP") +
  ggtitle("Oil-market specific demand shock")

ggarrange(plot1,plot2,plot3, ncol = 1)
```



Implications to quarterly inflation

We construct monthly PCE inflation as $100 \cdot \Delta\log(\text{PCEPI})$, aggregate to quarters, align with shocks, and estimate distributed-lag effects on PCE inflation.

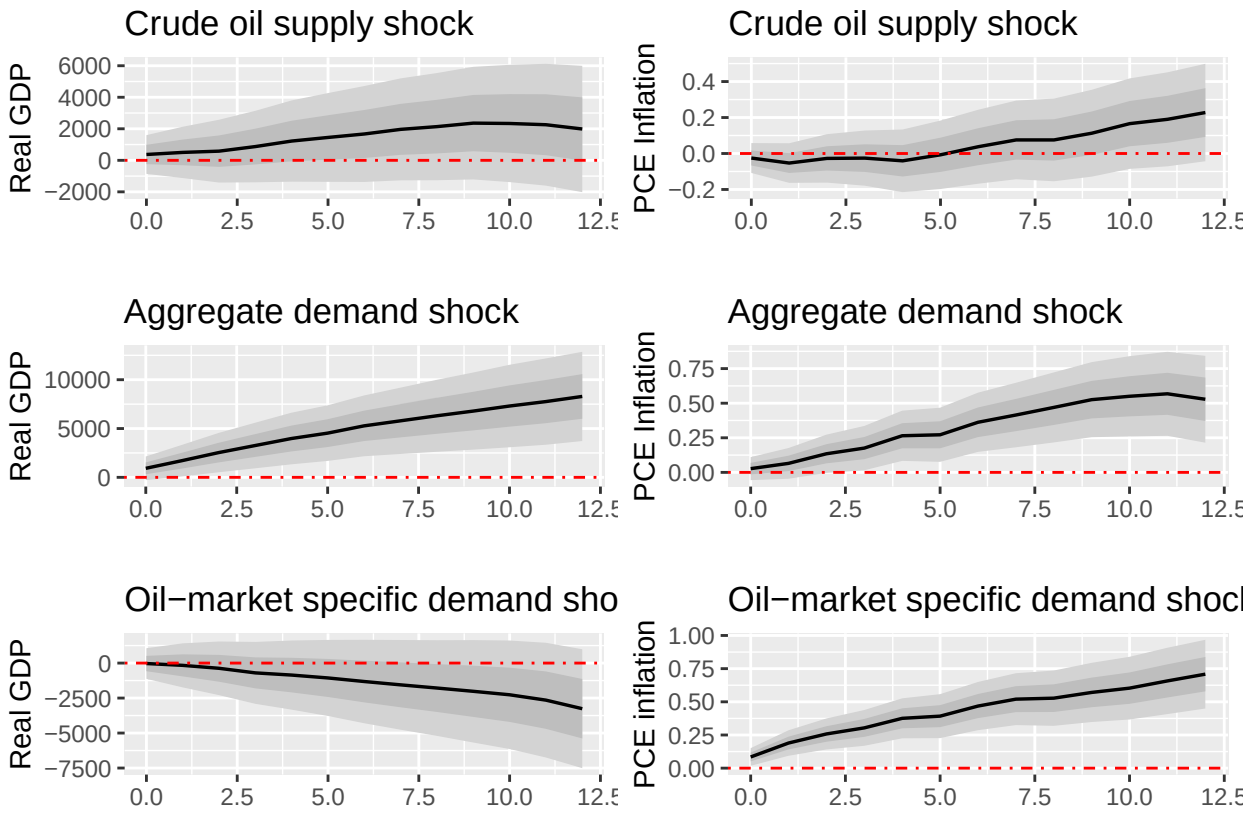
```
# Load PCE Inflation Data from FRED

# Fetch US PCE Price Index data (PCEPI) from FRED
getSymbols("PCEPI", src = "FRED", from = "1970-01-01", to = "2010-12-31")
```

```
## [1] "PCEPI"
```

```
p <- PCEPI
# Convert to monthly percent change to represent inflation rate
p <- diff(log(p)) * 100 #  _t = 100·[log(P_t) - log(P_{t-1})]
# Convert to xts object if needed for endpoints function
p <- as.xts(p)

ep.q <- endpoints(p,'quarters')
p.q  <- period.apply(p, INDEX=ep.q, FUN = mean, na.rm=TRUE) # quarter average at quarter-end

# subset the inflation series: 1975 - 2007
pce <- p.q["1975/2007"]
names(pce) <- "pce"

# use quarter-start to denote timing
index(pce) <- index(rgdp)
index(u.q) <- index(rgdp)

data.q <- cbind(pce, u.q)
data.q <- as.ts(data.q)    # package dynlm works only with "ts" objects

# OLS distributed-lags regression
eq1.y <- dynlm(pce ~ L(dprod,0:qlag), data = data.q)  # Inflation on supply shock lags
eq2.y <- dynlm(pce ~ L(rea,0:qlag),   data = data.q)  # Inflation on aggregate demand shock lags
eq3.y <- dynlm(pce ~ L(rpo,0:qlag),   data = data.q)  # Inflation on oil-specific demand shock lags


b4  <- cumsum(coef(eq1.y)[2:(qlag+2)])
v4  <- vcov(eq1.y)[2:(qlag+2),2:(qlag+2)]
vb4 <- apply(apply(v4, 2, cumsum), 1, cumsum)
sb4 <- sqrt(diag(vb4))   # SE of the cumsum of slope coefficients

b5  <- cumsum(coef(eq2.y)[2:(qlag+2)])
v5  <- vcov(eq2.y)[2:(qlag+2),2:(qlag+2)]
vb5 <- apply(apply(v5, 2, cumsum), 1, cumsum)
sb5 <- sqrt(diag(vb5))   # SE of the cumsum of slope coefficients

b6  <- cumsum(coef(eq3.y)[2:(qlag+2)])
v6  <- vcov(eq3.y)[2:(qlag+2),2:(qlag+2)]
vb6 <- apply(apply(v6, 2, cumsum), 1, cumsum)
sb6 <- sqrt(diag(vb6))   # SE of the cumsum of slope coefficients
```

- Plot cumulative impulse responses with +/- SE or 2SE bands

```
plot4 <- ggplot(as.data.frame(b4), aes(x = 0:qlag, y = b4)) +
  geom_ribbon(aes(ymin = b4 - 2*sb4, ymax = b4 + 2*sb4),
              fill = "lightgrey") +
  geom_ribbon(aes(ymin = b4 - sb4, ymax = b4 + sb4),
              fill = "grey") +
  geom_path(size = 0.6) +    # set the size of the time series line plot
  geom_hline(yintercept = 0, linetype = 4, colour = "red") +
  xlab("") +
  ylab("PCE Inflation") +
  ggtitle("Crude oil supply shock")

plot5 <- ggplot(as.data.frame(b5), aes(x = 0:qlag, y = b5)) +
  geom_ribbon(aes(ymin = b5 - 2*sb5, ymax = b5 + 2*sb5),
              fill = "lightgrey") +
  geom_ribbon(aes(ymin = b5 - sb5, ymax = b5 + sb5),
              fill = "grey") +
  geom_path(size = 0.6) +    # set the size of the time series line plot
  geom_hline(yintercept = 0, linetype = 4, colour = "red") +
  xlab("") +
  ylab("PCE Inflation") +
  ggtitle("Aggregate demand shock")

plot6 <- ggplot(as.data.frame(b6), aes(x = 0:qlag, y = b6)) +
  geom_ribbon(aes(ymin = b6 - 2*sb6, ymax = b6 + 2*sb6),
              fill = "lightgrey") +
  geom_ribbon(aes(ymin = b6 - sb6, ymax = b6 + sb6),
              fill = "grey") +
  geom_path(size = 0.6) +    # set the size of the time series line plot
  geom_hline(yintercept = 0, linetype = 4, colour = "red") +
  xlab("") +
  ylab("PCE inflation") +
  ggtitle("Oil-market specific demand shock")

ggarrange(plot1, plot4, plot2, plot5, plot3, plot6, nrow = 3, ncol = 2)
```



Conclusion

The results indicate that aggregate demand shocks are the dominant driver of macro outcomes: they lift both real GDP and PCE inflation strongly and persistently. Oil-market–specific demand shocks, by contrast, are stagflationary—reducing real GDP while pushing inflation higher. Pure oil supply shocks (increased production) have comparatively modest and less persistent effects on both variables, with responses often near zero initially and remaining small relative to demand-driven movements. Further work should extend the sample through the present, test alternative identification schemes (different Cholesky orderings, sign restrictions, or external instruments for oil supply), evaluate robustness to lag length and stability/breaks (e.g., 1986, 2008, 2014), consider richer models (SVAR, BVAR, or time-varying parameter VAR), incorporate additional controls (monetary policy, financial conditions, global activity indices), and perform out-of-sample forecast evaluation against benchmarks; examining core PCE inflation and alternative activity measures would also strengthen the conclusions.