Econ436 Project 3, Commodity Response to Exchange Rate

Eric Weber

May 1, 2019

```
#Notes on the data frame I made:
#doesn't include "PALLFNF" which is an aggregate index of all commodities
#changed "PFSHMEAL" to "PFISH" to match the description
#there were two "POILAPSP"s in original data, I took the one with more data

#indice 1: Long description
#indice 2: short name
#indice 3: category name
#indices 4 to 474: data from Jan-1980 to March-2019

#start=================================================================================
library(openxlsx)
```

```
## Warning: package 'openxlsx' was built under R version 3.5.3
```

```
library(vars)
```

```
## Warning: package 'vars' was built under R version 3.5.3
## Loading required package: MASS
## Loading required package: strucchange
## Warning: package 'strucchange' was built under R version 3.5.3
## 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: package 'sandwich' was built under R version 3.5.3
## Loading required package: urca
## Warning: package 'urca' was built under R version 3.5.3
## Loading required package: lmtest
## Warning: package 'lmtest' was built under R version 3.5.3
```

```
library(tseries)  #for ADF
```

```
## Warning: package 'tseries' was built under R version 3.5.3
```

```
library(urca)     #for unit root test

setwd("~/R")

#COMMODITY DATA
```

```r
comData <- read.xlsx("commodities_all.xlsx")
comData[2,51] <- "Crude Oil, WTI" #shorter name for West Texas Intermediate
#get the code-names of all commodities, minus the description column and xchg rate column
codeList <- names(comData)[-c(1,2)]


#IRF DATA (pre-made, because the vars package takes so long to compute)
dfirf <- read.csv("Commod_IRF_Vectors.csv",check.names=FALSE)
dfirf <- dfirf[,-1]


#function for getting the log-difference of a data column, given the column's name
get_log_diff <- function(code) {
  vect <- as.double(comData[(4):474,code])
  n <- length(vect)
  vect <- (log(vect[2:n])-log(vect[1:(n-1)]))*100
  return(vect)
}

#convert a short-code-name of a commodity to its more readable long name
getLongNames <- function(names) {
  n <- length(names)
  for (i in 1:n) names[i] <- comData[2,names[i]]
  return(names)
}

#convert short-code-names of commodities to their category
getCategory <- function(names) {
  n <- length(names)
  for (i in 1:n) names[i] <- comData[3,names[i]]
  return(names)
}

#DataFrame IRFs==========================================================================
#Generating a data Frame of all relevant IRF vectors (and 95% bounds of those vectors).

#creating the table
#for each commodity store:
# TWEX on TWEX: irf.11$irf$TWEX, irf.11$Lower$TWEX, irf.11$Upper$TWEX
# COMMOD on TWEX: irf.21$irf$TWEX, irf.21$Lower$TWEX, irf.21$Upper$TWEX

#Codes: Ti, TL, TU... Ci, CL, CU

#WE DON'T NEED THIS CODE BECAUSE WE LOADED THE IRF VECTORS FROM CSV FILE
#THIS IS THE CODE TO GENERATE THAT CSV FILE

# tableCols <- c("start")
# for (code in codeList) {
#   tableCols <- c(tableCols, paste(code,"Ti"),paste(code,"TL"),paste(code,"TU"))
#   tableCols <- c(tableCols, paste(code,"Ci"),paste(code,"CL"),paste(code,"CU"))
# }
# tableCols <- tableCols[-1] #remove dummy
#
```

```
# dfirf <- data.frame(matrix(ncol=length(tableCols), nrow=13))
# colnames(dfirf) <- tableCols
#
# #for looping
# for (code in codeList) {
#   TWEX <- get_log_diff("XCHG")
#   COMMOD <- get_log_diff(code)
#   z = as.matrix(cbind(TWEX,COMMOD))  #create two columns
#   p.est <- as.numeric(VARselect(z)$selection[1])  #gets AIC estimated p-lag-value
#   var.1c <- VAR(z, p=p.est, type = "none")
#
#   irf.11 = irf(var.1c, impulse = "TWEX", n.ahead=12, response="TWEX", boot=TRUE, cumulative=TRU
#   irf.21 = irf(var.1c, impulse = "TWEX", n.ahead=12, response="COMMOD", boot=TRUE, cumulative=T
#
#   # TWEX on TWEX:
#   dfirf[,paste(code,"Ti")] <- irf.11$irf$TWEX
#   dfirf[,paste(code,"TL")] <- irf.11$Lower$TWEX
#   dfirf[,paste(code,"TU")] <- irf.11$Upper$TWEX
#   # COMMOD on TWEX:
#   dfirf[,paste(code,"Ci")] <- irf.21$irf$TWEX
#   dfirf[,paste(code,"CL")] <- irf.21$Lower$TWEX
#   dfirf[,paste(code,"CU")] <- irf.21$Upper$TWEX
#
#   #print(code)
#   # if (readline("Press enter to continue, type 'end' to exit loop...") == "end") break
# }
#
# #write.csv(dfirf,"Commod_IRF_Vectors.csv")


#DataFrame Dynamic Elasticities============================================================
#Generating a data Frame for the dynamic elasticity vectors of all commodities

#creating the table
#for each commodity store:
# Dynamic Elasticity: DYNELAST, DYNELAST_LOWER, DYNELAST_UPPER
#Codes: D, DL, DU

tableCols <- c("start")
for (code in codeList) {
  tableCols <- c(tableCols, paste(code,"D"),paste(code,"DL"),paste(code,"DU"))
}
tableCols <- tableCols[-1] #remove dummy
dfde <- data.frame(matrix(ncol=length(tableCols), nrow=13))
colnames(dfde) <- tableCols
#View(dfde)  #test

#for looping
for (code in codeList) {

  irfcom <- dfirf[,paste(code,"Ci")]
  irftwex <- dfirf[,paste(code,"Ti")]
  dynelas <- ((-1)*irfcom)/irftwex
```

```r
  DyneElastLower <- ((-1)*(dfirf[,paste(code,"CL")]))/(dfirf[,paste(code,"TL")])
  DyneElastUpper <- ((-1)*(dfirf[,paste(code,"CU")]))/(dfirf[,paste(code,"TU")])

  #SINCE COMMOD RESPONSE IS LOWER THAN TWEX RESPONSE ALWAYS, "UPPER" AND "LOWER" SWITCH
  dfde[,paste(code,"D")] <- dynelas
  dfde[,paste(code,"DL")] <- DyneElastUpper
  dfde[,paste(code,"DU")] <- DyneElastLower

  #print(code)
  #if (readline("Press enter to continue, type 'end' to exit loop...") == "end") break
}

#write.csv(dfde,"Commod_DE_Vectors.csv")

#Data Frame For Rankings ================================================================
#and classification of the dynamic elasticities

tableCols <- c("Avg DE","Avg Lower","Avg Upper","Classify","Test Result")
dfrnk <- data.frame(matrix(ncol=length(tableCols), nrow=length(codeList)), row.names=codeList)
colnames(dfrnk) <- tableCols
#View(dfrnk) #test

for (code in codeList) {

  dfrnk[code,"Avg DE"] <- signif(mean(dfde[,paste(code,"D")]),4)
  dfrnk[code,"Avg Lower"] <- signif(mean(dfde[,paste(code,"DL")]),4)
  dfrnk[code,"Avg Upper"] <- signif(mean(dfde[,paste(code,"DU")]),4)

  if (dfrnk[code,"Avg DE"] < (1-0.2)) {
    dfrnk[code,"Classify"] <- "UNDER"
  } else if (dfrnk[code,"Avg DE"] > (1+0.2)) {
    dfrnk[code,"Classify"] <- "OVER"
  } else {
    dfrnk[code,"Classify"] <- "SAME"
  }

  if (dfrnk[code,"Avg Upper"] < 1) {
    dfrnk[code,"Test Result"] <- "UNDER"
  } else if (dfrnk[code,"Avg Lower"] > 1) {
    dfrnk[code,"Test Result"] <- "OVER"
  } else {
    dfrnk[code,"Test Result"] <- "---"
  }
  #print(code)
}

#give it the long names so we can read it
row.names(dfrnk) <- getLongNames(codeList)
#sort descending (hence the minus)
#rank the lower bounds of the dynamic elasticities as ascending...
dfrnk <- dfrnk[order(-dfrnk[,"Avg DE"]),]
dfrnk
```

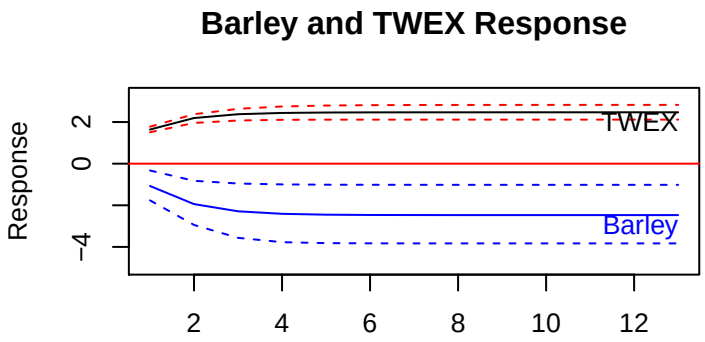
	Avg DE	Avg Lower	Avg Upper	Classify
Oil, Dubai	1.42800	0.519400	2.4940	OVER
Nickel	1.41500	0.606000	2.3120	OVER
Crude Oil, Price index	1.40300	0.596400	2.2850	OVER
Crude Oil, Dated Brent	1.39900	0.472500	2.3240	OVER
Copper	1.31800	0.730100	2.1430	OVER
Rubber	1.26900	0.571200	2.0050	OVER
Lead	1.25600	0.595700	2.1820	OVER
Coal	1.21100	0.261000	2.5620	OVER
Crude Oil, WTI	1.20100	0.351300	2.1370	OVER
Aluminum	1.17600	0.620400	2.0280	SAME
Olive Oil	1.17300	0.720100	1.7910	SAME
Lamb	1.14300	0.743500	1.7850	SAME
Sugar, Free Market	1.05300	0.363100	2.0830	SAME
Salmon	0.97050	0.357400	1.8610	SAME
Barley	0.96180	0.346400	1.7280	SAME
Tin	0.91280	0.280000	1.7640	SAME
Rapeseed oil	0.86900	0.350600	1.4710	SAME
Cocoa beans	0.83140	0.327200	1.3890	SAME
Fishmeal	0.78390	0.392500	1.4200	UNDER
Zinc	0.77580	0.156900	1.4630	UNDER
Palm oil	0.74180	-0.124200	1.9050	UNDER
Hard Logs	0.73600	0.258900	1.4740	UNDER
Wool, coarse	0.72240	0.274400	1.2860	UNDER
Sunflower oil	0.69780	0.045370	1.6580	UNDER
Groundnuts	0.66380	-0.004894	1.3980	UNDER
Soybean Meal	0.64460	0.022890	1.1990	UNDER
Soybean Oil	0.62990	0.129500	1.3290	UNDER
Iron Ore	0.60540	0.180900	1.1540	UNDER
Cotton	0.60340	0.010860	1.4370	UNDER
Soybeans	0.60120	0.036520	1.1930	UNDER
Rice	0.57170	0.066550	1.4350	UNDER
Hides	0.55650	0.025050	1.2140	UNDER
Hard Sawnwood	0.52400	0.073020	1.1990	UNDER
Uranium	0.50970	-0.140700	1.4440	UNDER
Wheat	0.47920	0.070560	0.9999	UNDER
Coffee, Robusta	0.46880	0.028900	1.0570	UNDER
Wool, fine	0.46290	0.043770	0.9809	UNDER
Tea	0.32550	-0.337000	1.0650	UNDER
Sugar, U.S. import price	0.30050	-0.145900	0.8498	UNDER
Maize	0.28850	-0.179600	0.7984	UNDER
Coffee, Other Mild Arabicas	0.24960	-0.271700	1.0960	UNDER
Pork	0.23880	-0.521800	1.1630	UNDER
Shrimp	0.20130	-0.108900	0.5030	UNDER
Beef	0.13810	-0.088260	0.5231	UNDER
Bananas	0.08774	-0.577500	1.1580	UNDER
Oranges	0.08168	-0.333600	0.6862	UNDER
Poultry	0.02430	-0.208100	0.4126	UNDER
Soft Sawnwood	-0.02240	-0.267000	0.3015	UNDER
Soft Logs	-0.03624	-0.295400	0.1812	UNDER

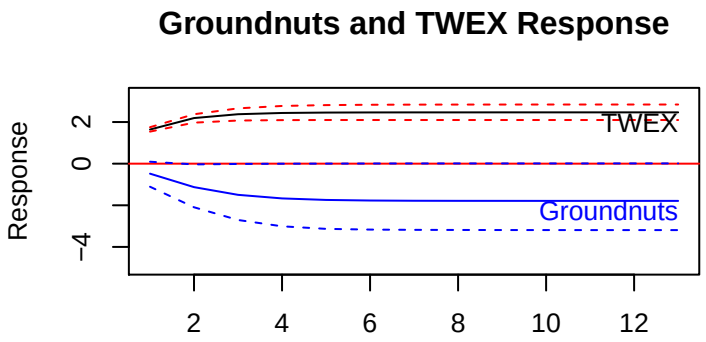
	Test Result
Oil, Dubai	---
Nickel	---
Crude Oil, Price index	---

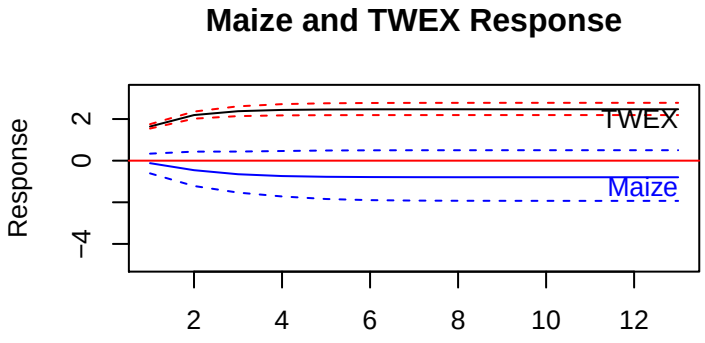
```
## Crude Oil, Dated Brent              ---
## Copper                              ---
## Rubber                              ---
## Lead                                ---
## Coal                                ---
## Crude Oil, WTI                      ---
## Aluminum                            ---
## Olive Oil                           ---
## Lamb                                ---
## Sugar, Free Market                  ---
## Salmon                              ---
## Barley                              ---
## Tin                                 ---
## Rapeseed oil                        ---
## Cocoa beans                         ---
## Fishmeal                            ---
## Zinc                                ---
## Palm oil                            ---
## Hard Logs                           ---
## Wool, coarse                        ---
## Sunflower oil                       ---
## Groundnuts                          ---
## Soybean Meal                        ---
## Soybean Oil                         ---
## Iron Ore                            ---
## Cotton                              ---
## Soybeans                            ---
## Rice                                ---
## Hides                               ---
## Hard Sawnwood                       ---
## Uranium                             ---
## Wheat                             UNDER
## Coffee, Robusta                     ---
## Wool, fine                        UNDER
## Tea                                 ---
## Sugar, U.S. import price          UNDER
## Maize                             UNDER
## Coffee, Other Mild Arabicas         ---
## Pork                                ---
## Shrimp                            UNDER
## Beef                              UNDER
## Bananas                             ---
## Oranges                           UNDER
## Poultry                           UNDER
## Soft Sawnwood                     UNDER
## Soft Logs                         UNDER
```

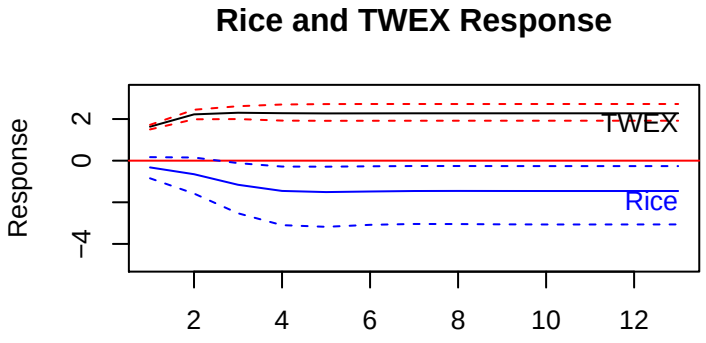
```
#write.csv(dfrnk,"Commod_Rnk_Table.csv")

#IRF PLOTS==============================================================================
#PLOTS FOR: Commod and TWEX Response to Twex (IRFs, 1% change in XCHG Rate)

par(mfrow=c(2,2))
for (code in codeList) {
```

```
  #irf.11: twex response to twex
  irf.ti <- dfirf[,paste(code,"Ti")]
  irf.tl <- dfirf[,paste(code,"TL")]
  irf.tu <- dfirf[,paste(code,"TU")]

  #irf.21: commod resp to twex
  irf.ci <- dfirf[,paste(code,"Ci")]
  irf.cl <- dfirf[,paste(code,"CL")]
  irf.cu <- dfirf[,paste(code,"CU")]

  #plot(irf.11$irf$TWEX, main=paste("TWEX to TWEX:",comData[2,code]), type="l", ylab="TWEX", xla
  plot(irf.ti, main=paste(comData[2,code],"and TWEX Response"), type="l", ylab="Response", xlab=""
  lines(irf.tl, col="red", lty=2)
  lines(irf.tu, col="red", lty=2)
  text(13,irf.ti[13],labels="TWEX",adj=c(1,1))
  abline(h=0,col="red")

  lines(irf.ci, col="blue")
  lines(irf.cl, col="blue", lty=2)
  lines(irf.cu, col="blue", lty=2)
  text(13,irf.ci[13],labels=comData[2,code],adj=c(1,1),col="blue")

  #if (readline("Press enter to continue, type 'end' to exit loop...") == "end") break
}
```

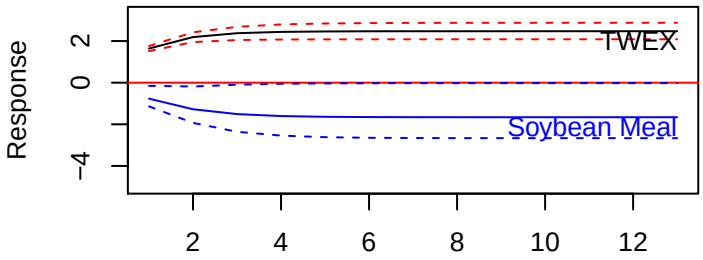




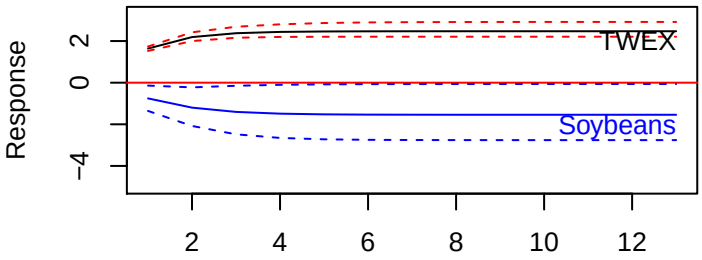




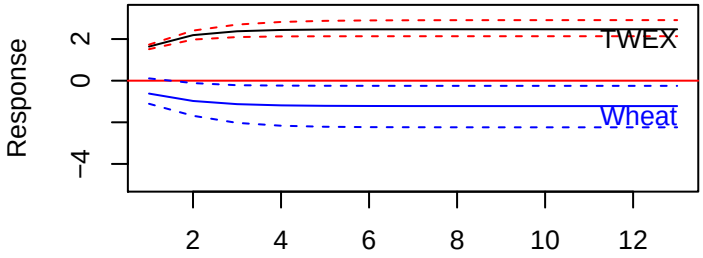
Soybean Meal and TWEX Response



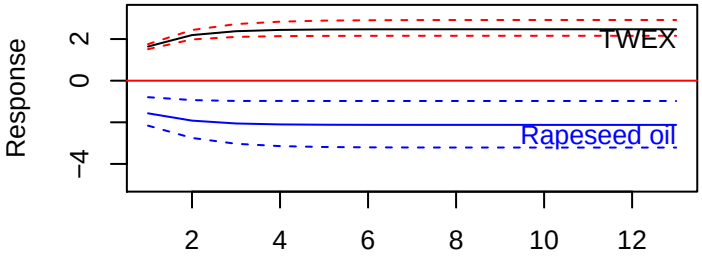
Soybeans and TWEX Response



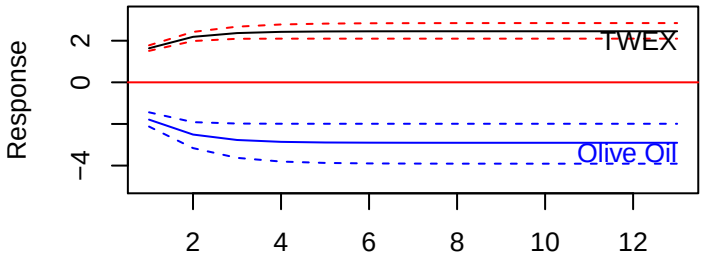
Wheat and TWEX Response



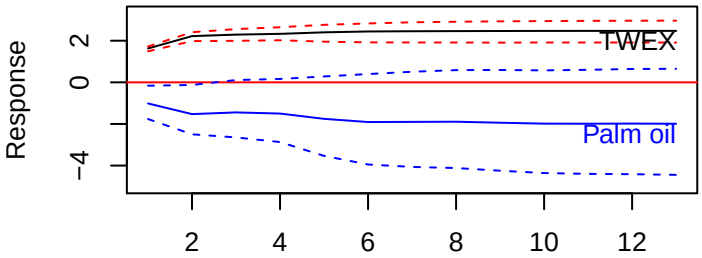
Rapeseed oil and TWEX Response



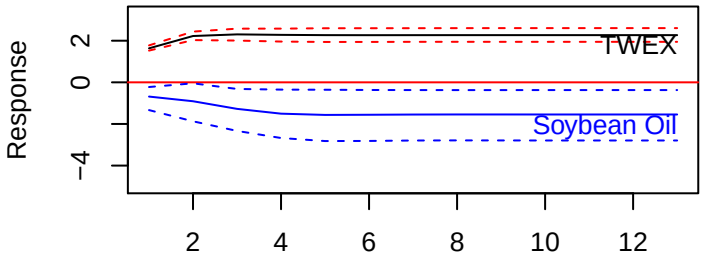
Olive Oil and TWEX Response



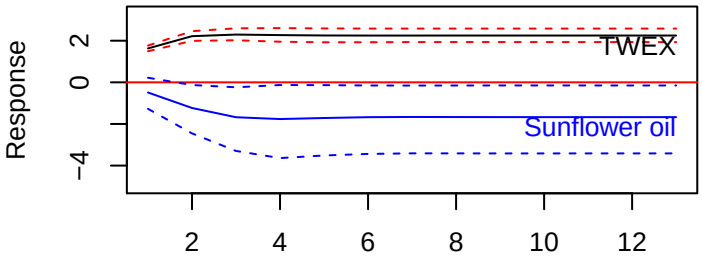
Palm oil and TWEX Response

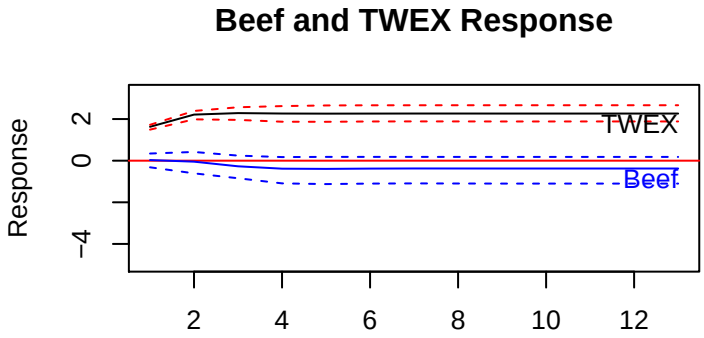


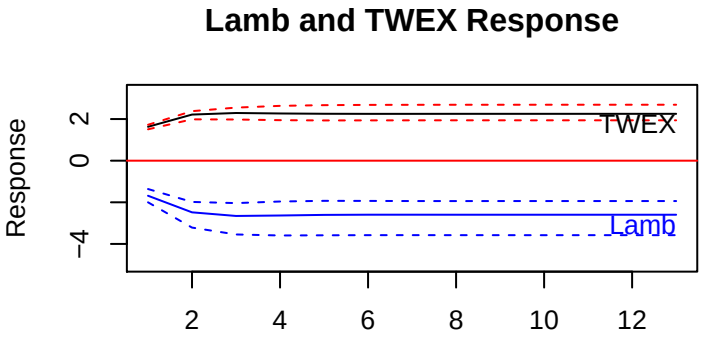
Soybean Oil and TWEX Response

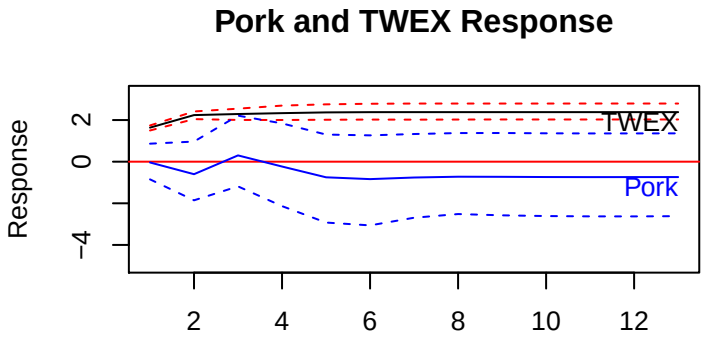


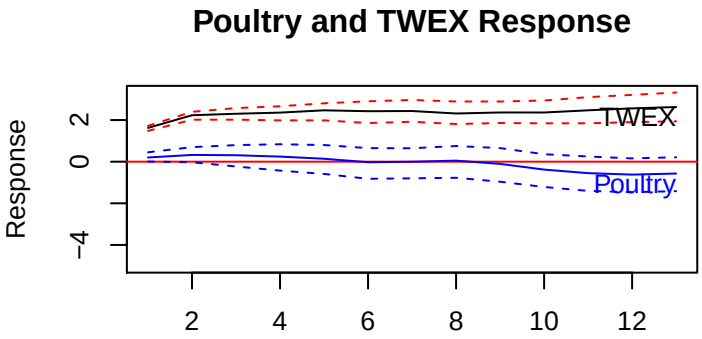
Sunflower oil and TWEX Response

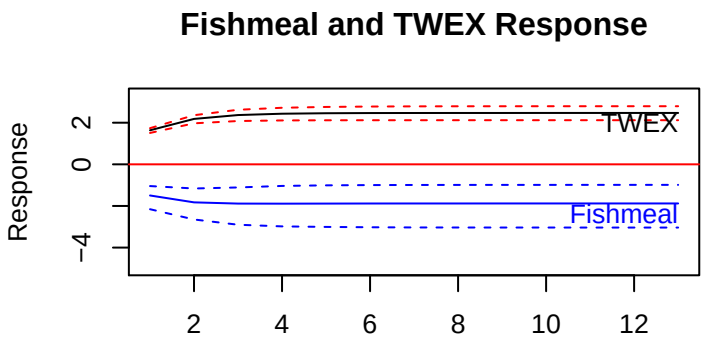


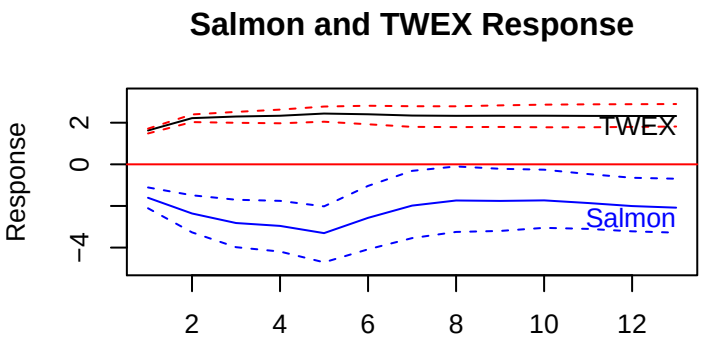

Beef and TWEX Response
Response
TWEX
Beef

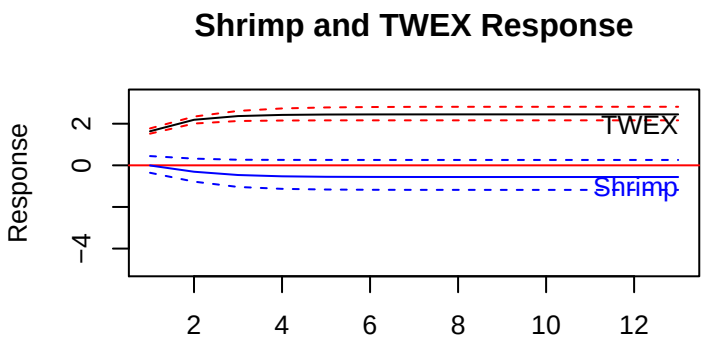

Lamb and TWEX Response
Response
TWEX
Lamb

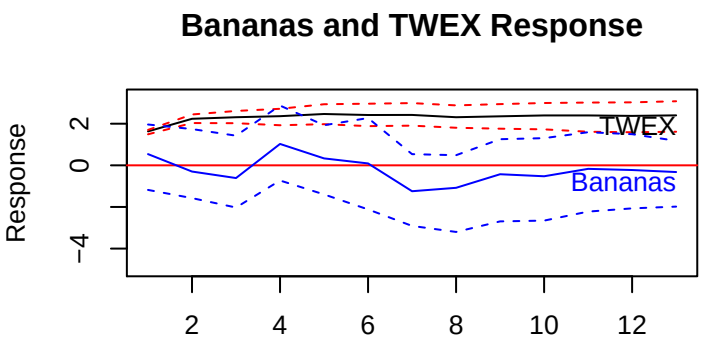

Pork and TWEX Response
Response
TWEX
Pork


Poultry and TWEX Response
Response
TWEX
Poultry

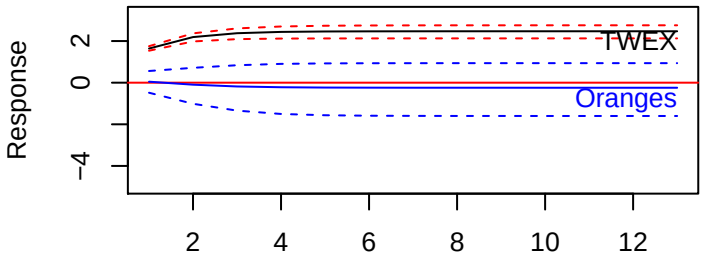

Fishmeal and TWEX Response
Response
TWEX
Fishmeal


Salmon and TWEX Response
Response
TWEX
Salmon

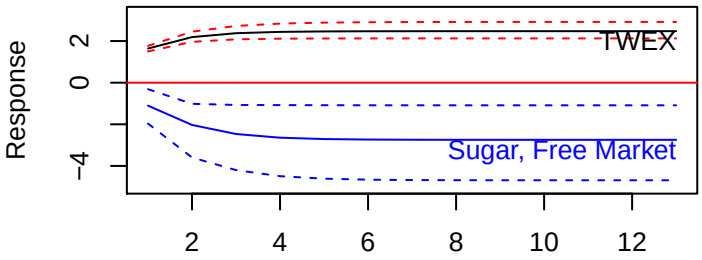

Shrimp and TWEX Response
Response
TWEX
Shrimp


Bananas and TWEX Response
Response
TWEX
Bananas

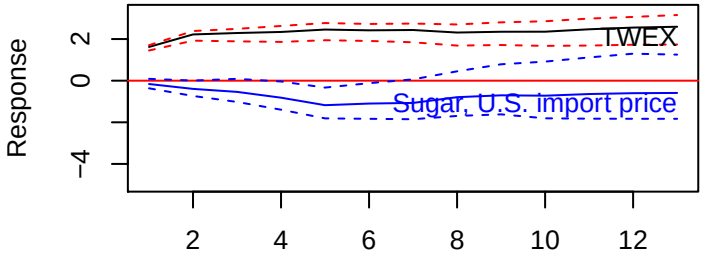
Oranges and TWEX Response



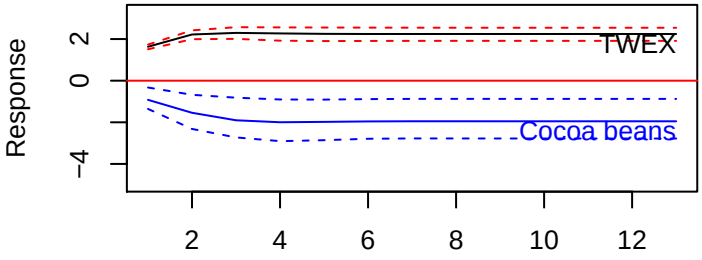
Sugar, Free Market and TWEX Response



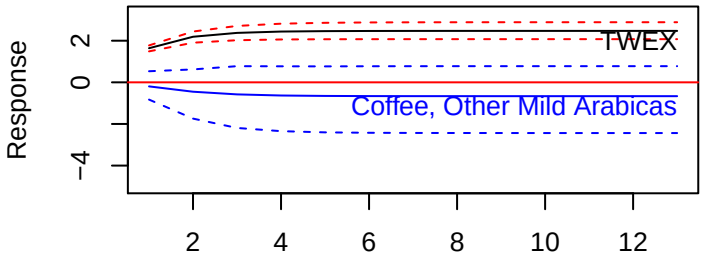
Sugar, U.S. import price and TWEX Response



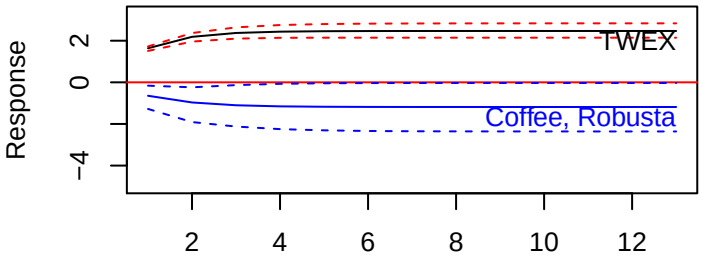
Cocoa beans and TWEX Response



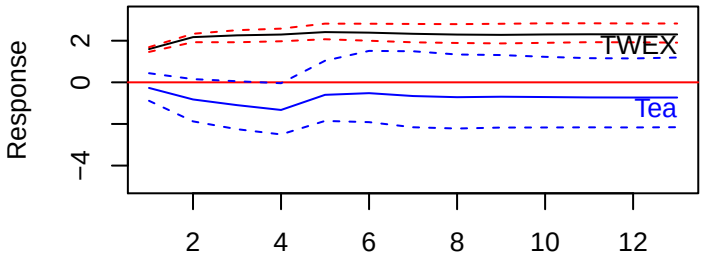
Coffee, Other Mild Arabicas and TWEX Respons



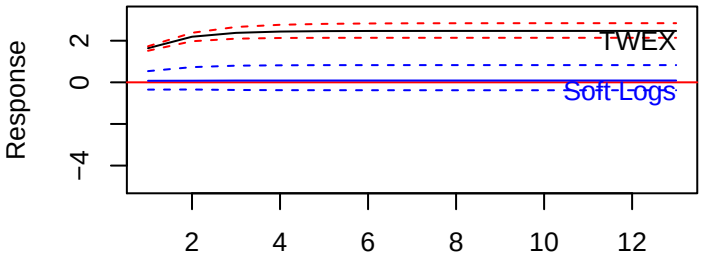
Coffee, Robusta and TWEX Response



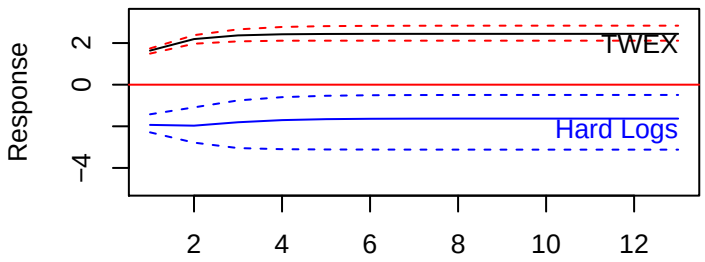
Tea and TWEX Response



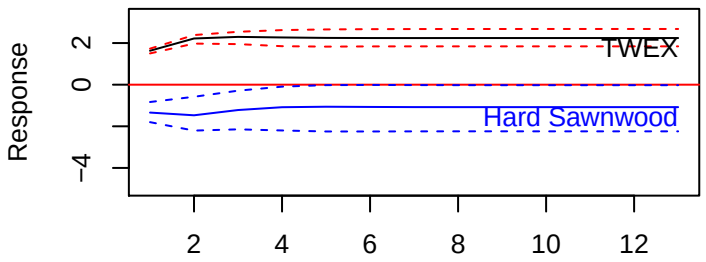
Soft Logs and TWEX Response



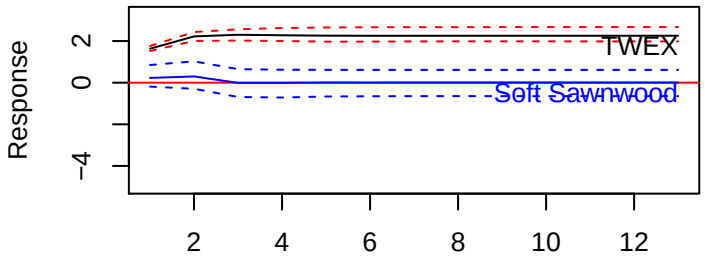
Hard Logs and TWEX Response



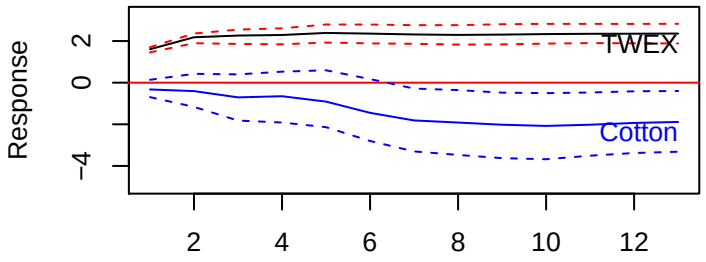
Hard Sawnwood and TWEX Response



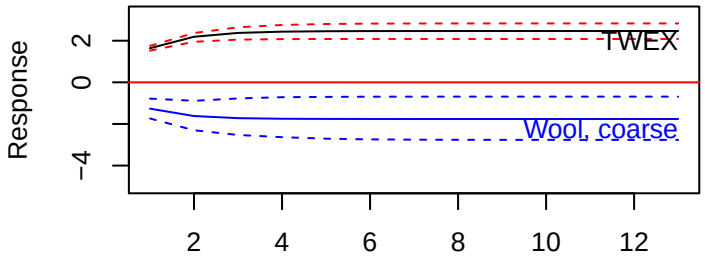
Soft Sawnwood and TWEX Response



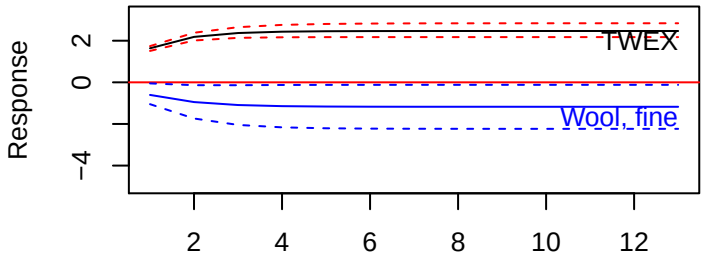
Cotton and TWEX Response



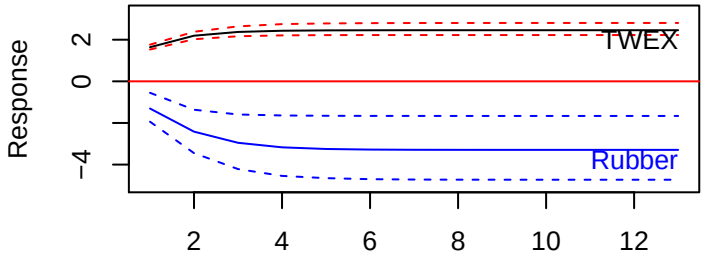
Wool, coarse and TWEX Response



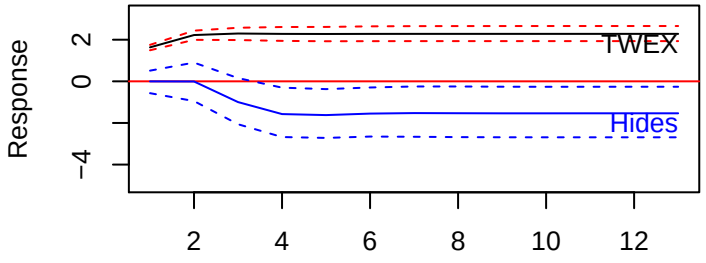
Wool, fine and TWEX Response



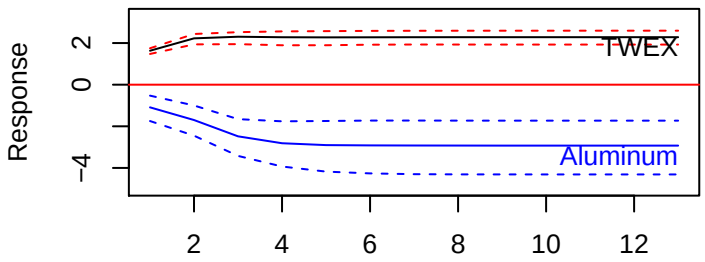
Rubber and TWEX Response



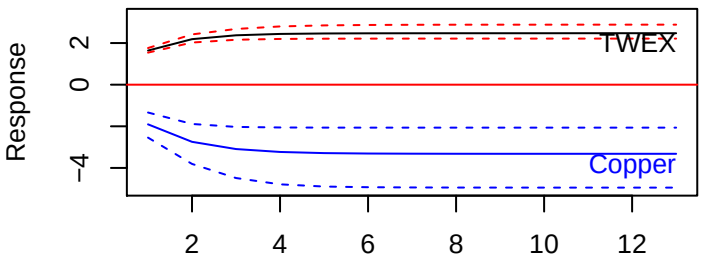
Hides and TWEX Response



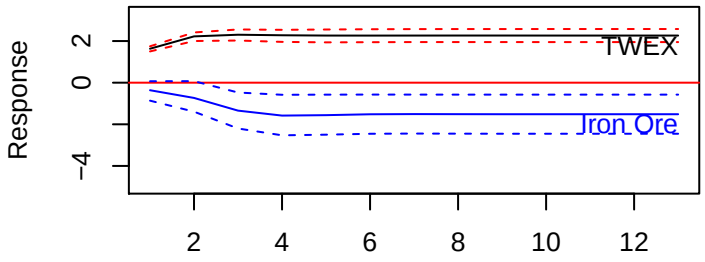
Aluminum and TWEX Response



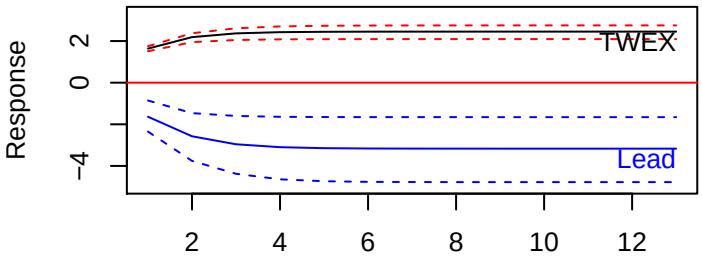
Copper and TWEX Response



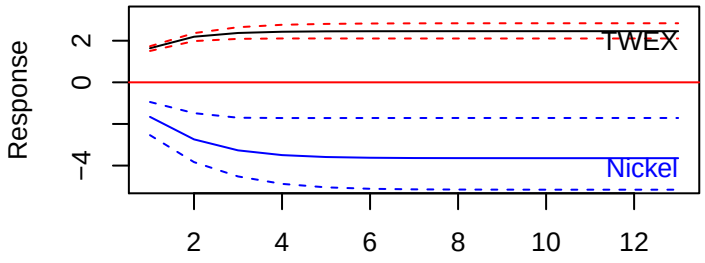
Iron Ore and TWEX Response



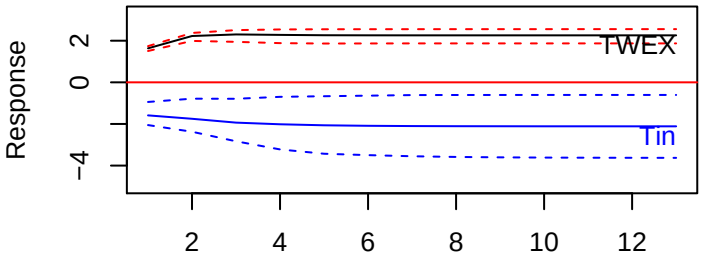
Lead and TWEX Response



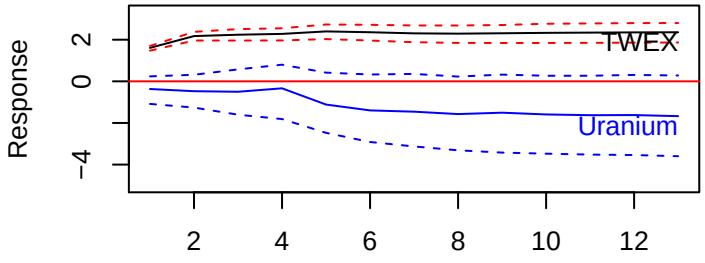
Nickel and TWEX Response



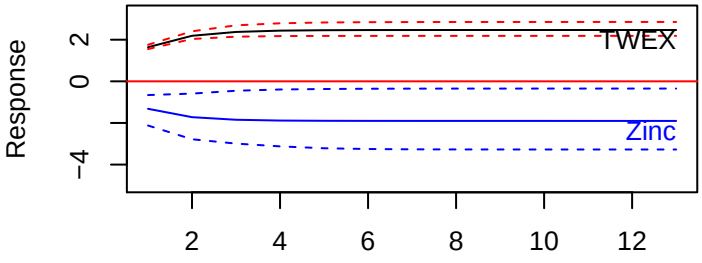
Tin and TWEX Response

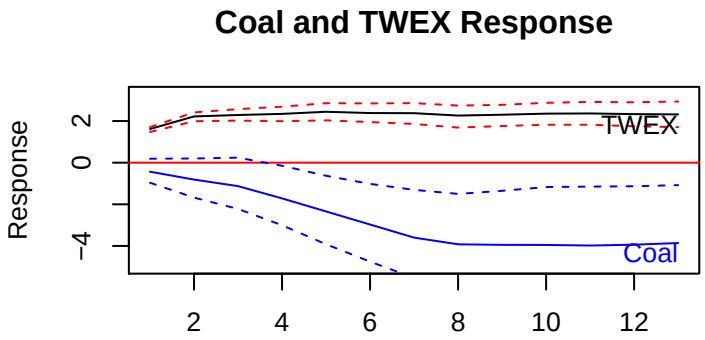


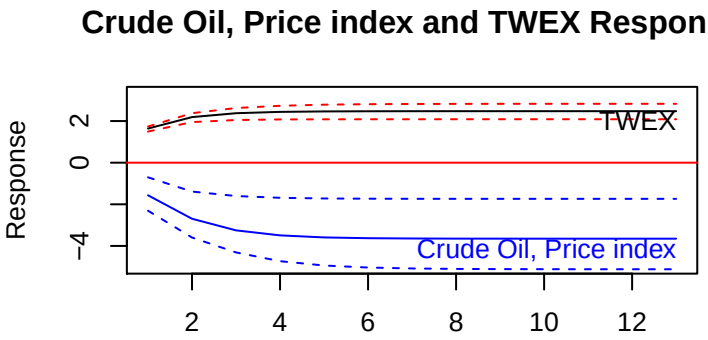
Uranium and TWEX Response

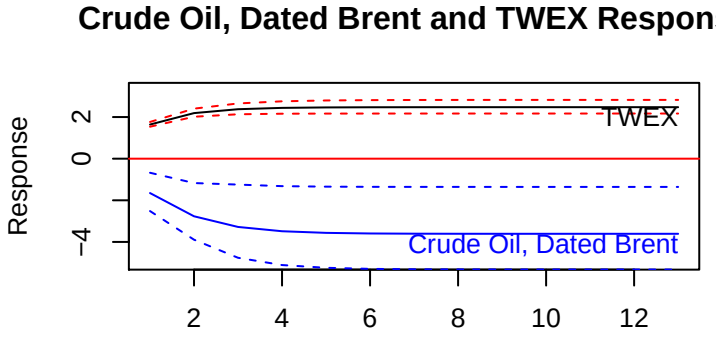


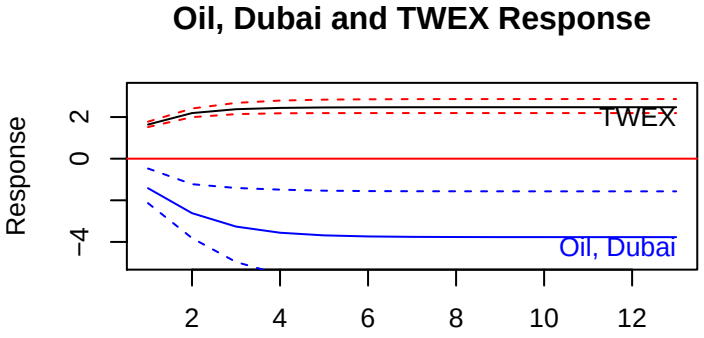
Zinc and TWEX Response



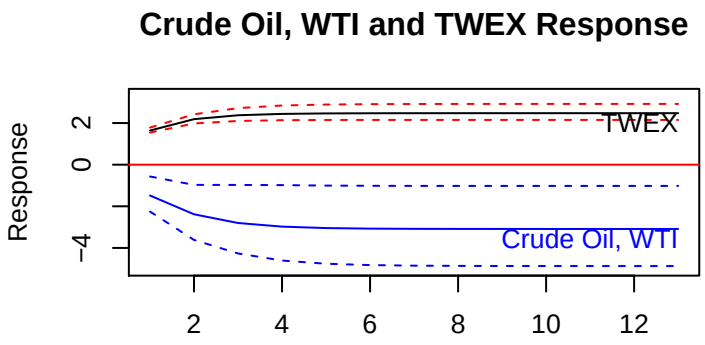






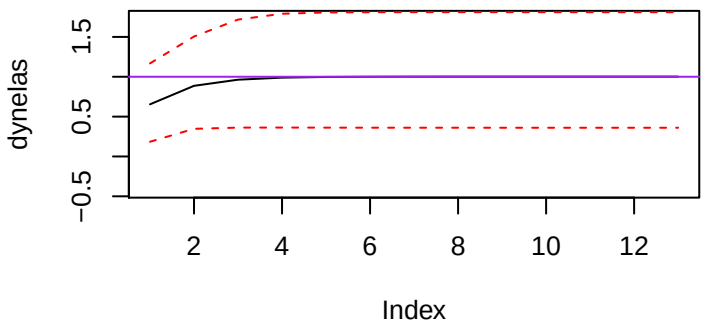



```
#Dynamic Elasticity PLOTS=============================================================
#PLOTS FOR: Dynamic Elasticity

par(mfrow=c(2,2))
```

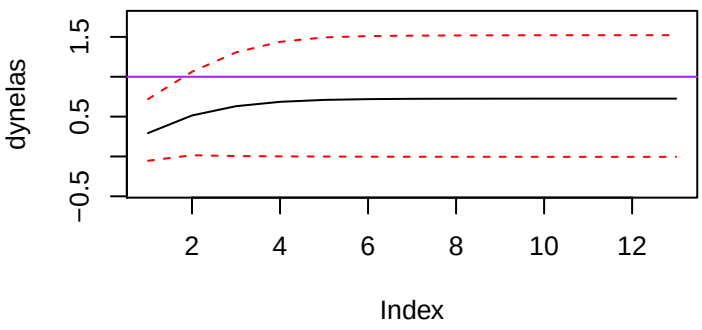



```
for (code in codeList) {

  dynelas <- dfde[,paste(code,"D")]
  DyneElastLower <- dfde[,paste(code,"DL")]
  DyneElastUpper <- dfde[,paste(code,"DU")]

  plot(dynelas, ylim=c(-0.43, 1.74), type="l", main=paste("Dynamic Elast:",comData[2,code]))
  lines(DyneElastLower, col="red", lty=2)  #lower 95% bound
  lines(DyneElastUpper, col="red", lty=2)  #upper 95% bound
  abline(h=1, col="purple") #horizontal at 1.0... designation of over or under reaction
  #abline(h=mean(dynelas), col="green")  #mean of dynamic elasticity
  #abline(h=mean(DyneElastLower), col="green", lty=2) #mean of lower bound
  #abline(h=mean(DyneElastUpper), col="green", lty=2) #mean of upper bound

  #if (readline("Press enter to continue, type 'end' to exit loop...") == "end") break
}
```

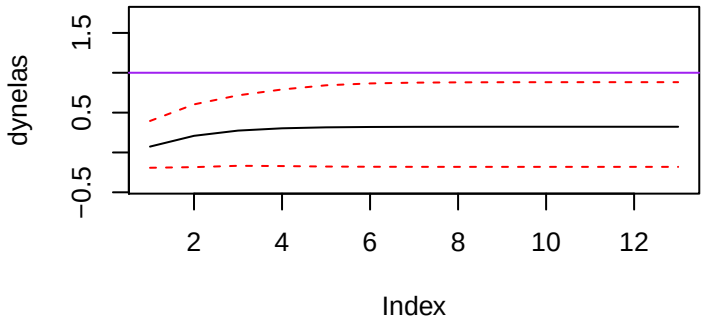
Dynamic Elast: Barley



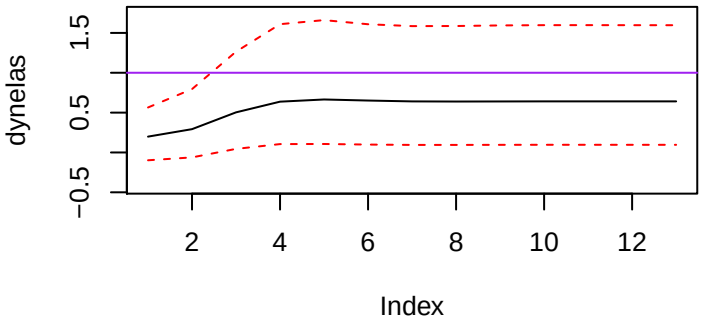
Dynamic Elast: Groundnuts



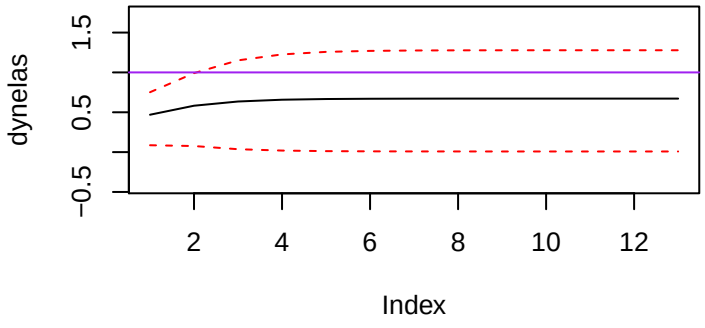
Dynamic Elast: Maize



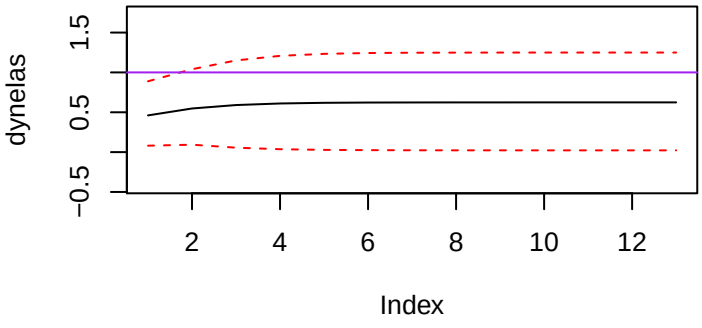
Dynamic Elast: Rice



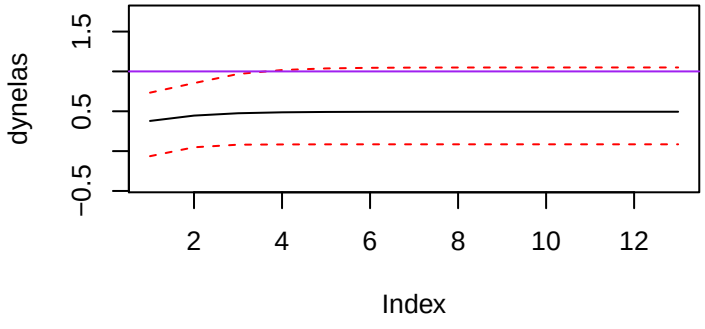
Dynamic Elast: Soybean Meal



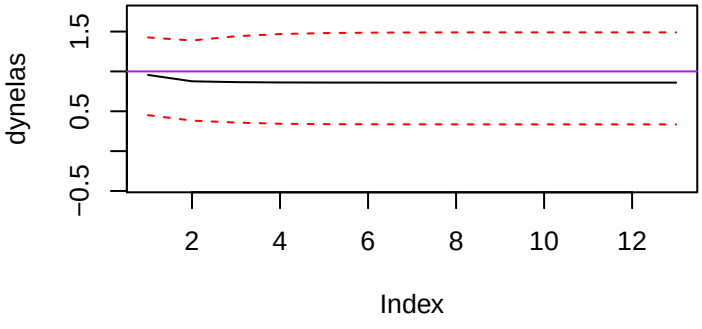
Dynamic Elast: Soybeans



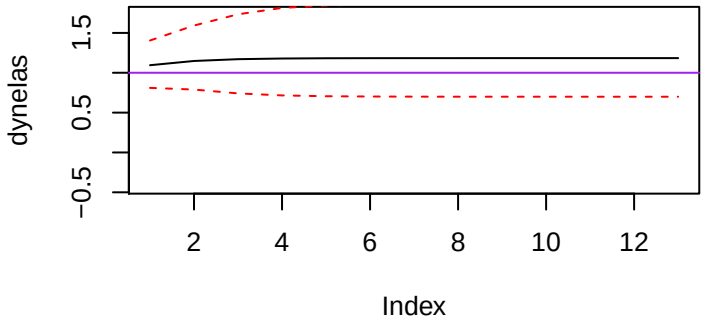
Dynamic Elast: Wheat



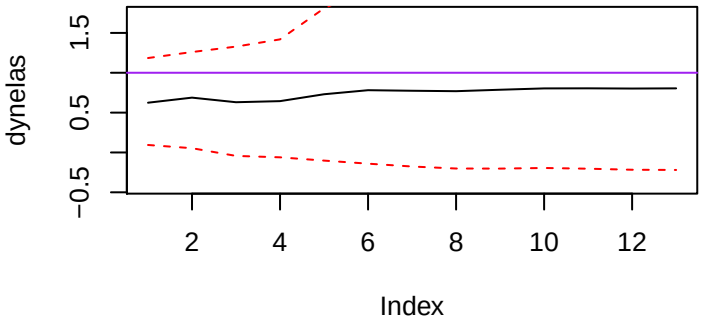
Dynamic Elast: Rapeseed oil



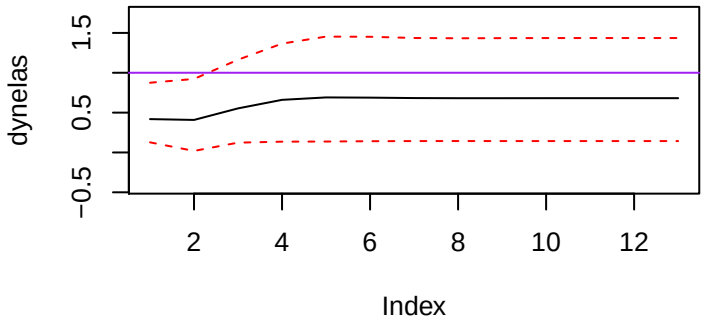
Dynamic Elast: Olive Oil



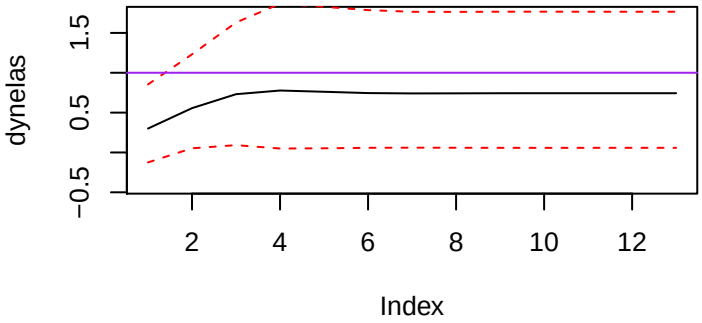
Dynamic Elast: Palm oil



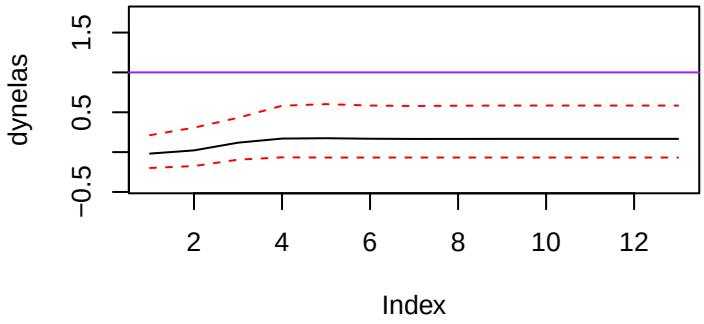
Dynamic Elast: Soybean Oil



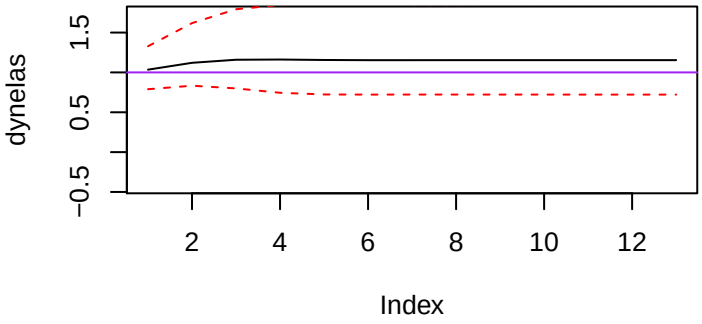
Dynamic Elast: Sunflower oil



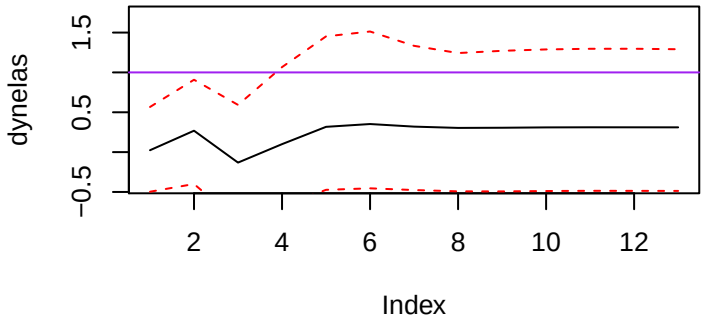
Dynamic Elast: Beef



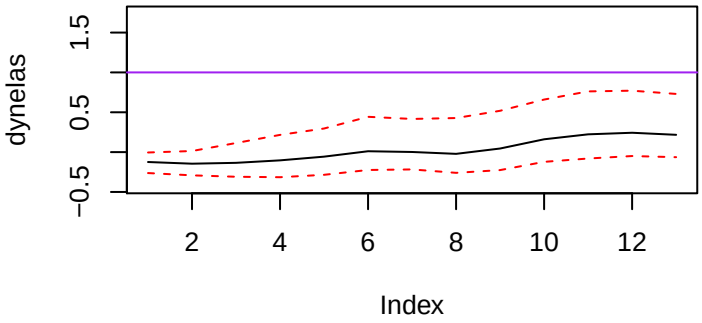
Dynamic Elast: Lamb



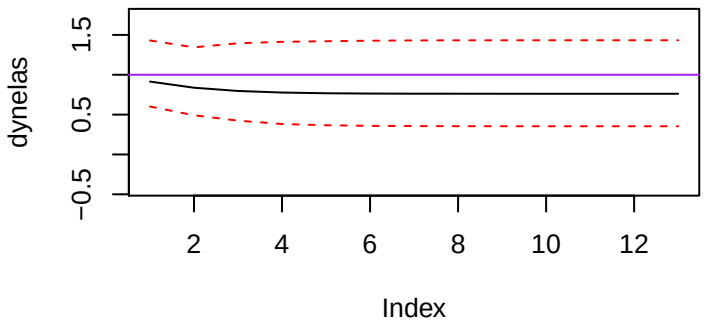
Dynamic Elast: Pork



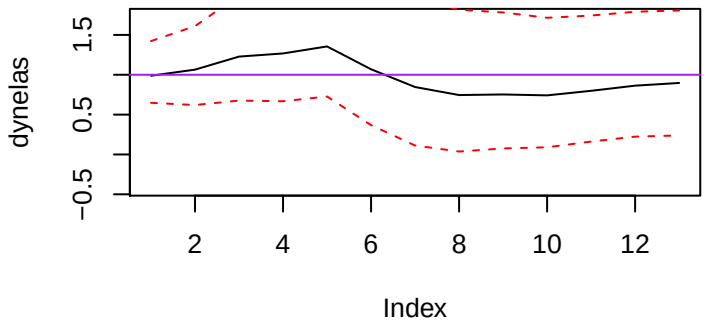
Dynamic Elast: Poultry



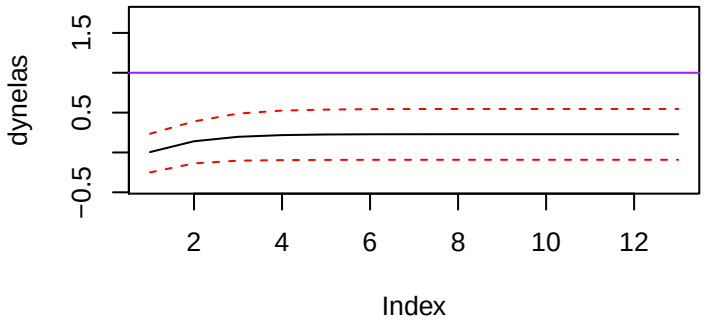
Dynamic Elast: Fishmeal



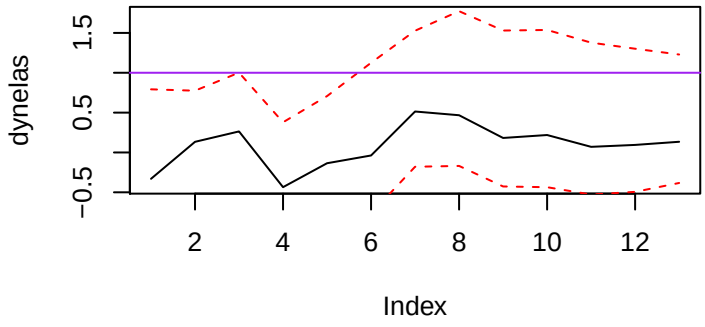
Dynamic Elast: Salmon



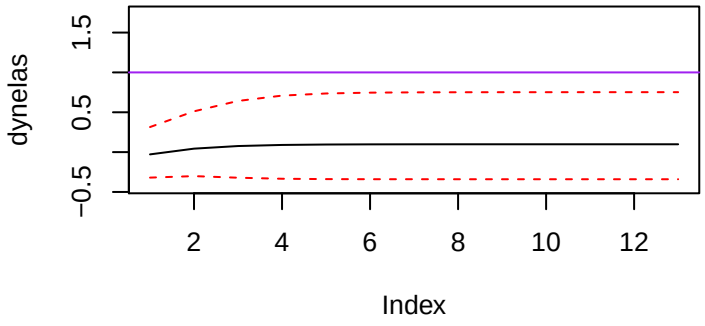
Dynamic Elast: Shrimp



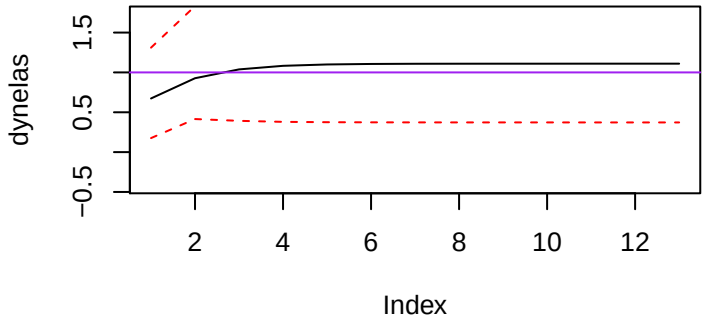
Dynamic Elast: Bananas



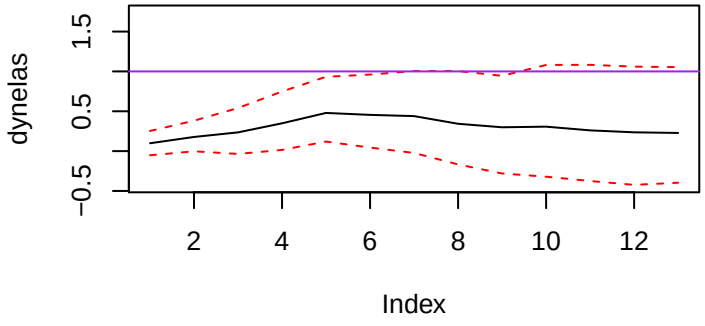
Dynamic Elast: Oranges



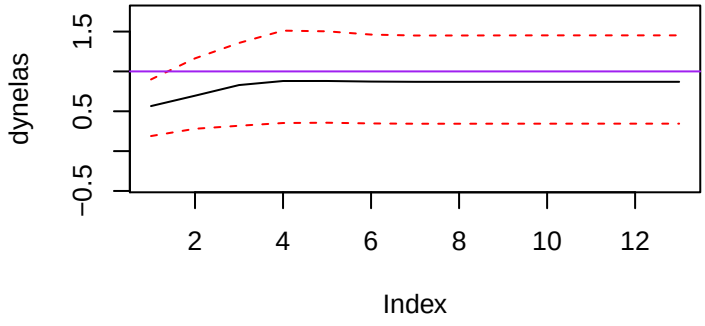
Dynamic Elast: Sugar, Free Market



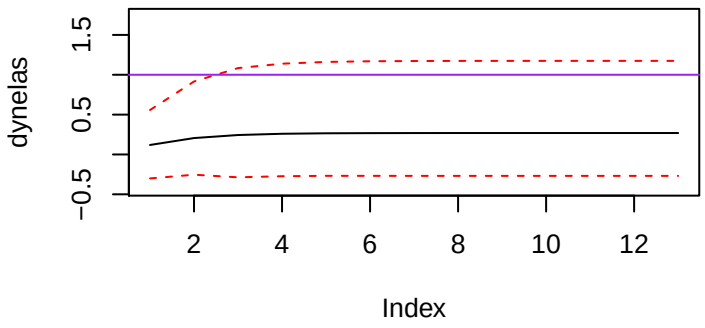
Dynamic Elast: Sugar, U.S. import price



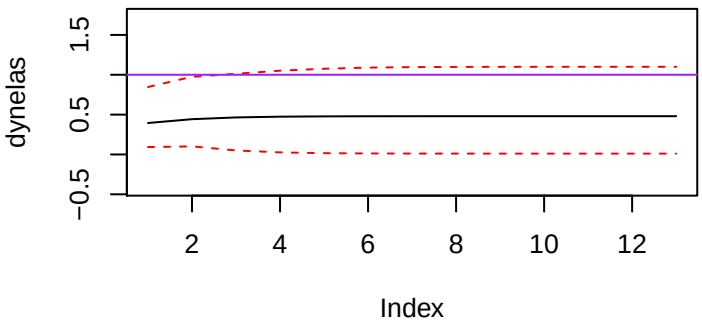
Dynamic Elast: Cocoa beans



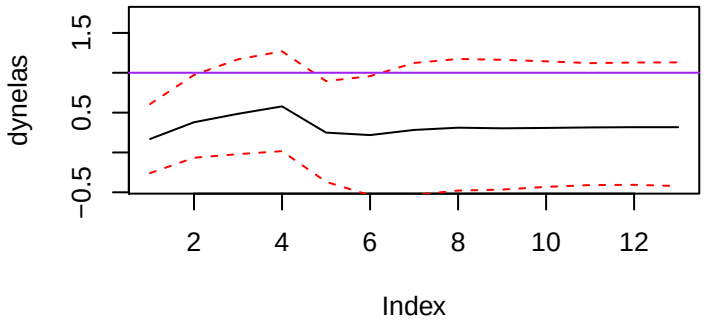
Dynamic Elast: Coffee, Other Mild Arabicas



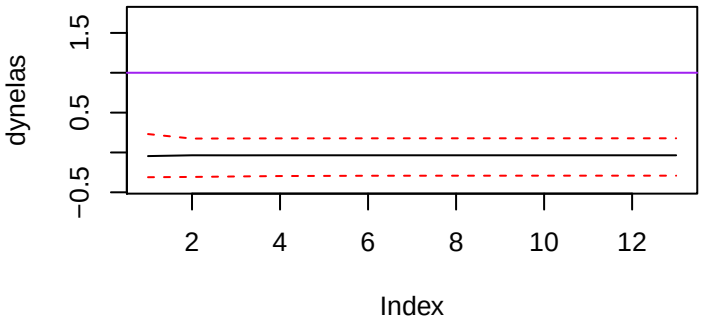
Dynamic Elast: Coffee, Robusta



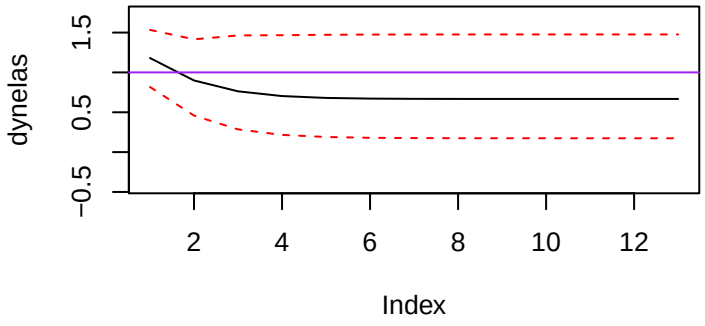
Dynamic Elast: Tea



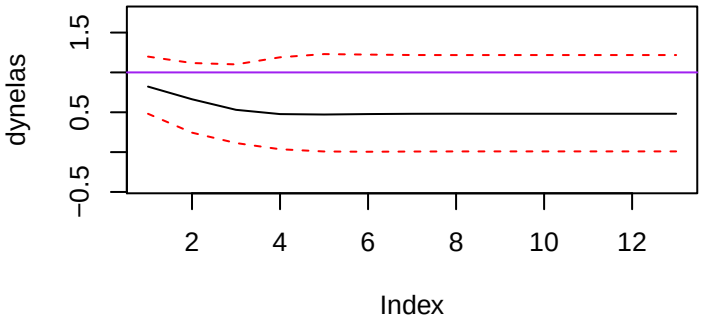
Dynamic Elast: Soft Logs



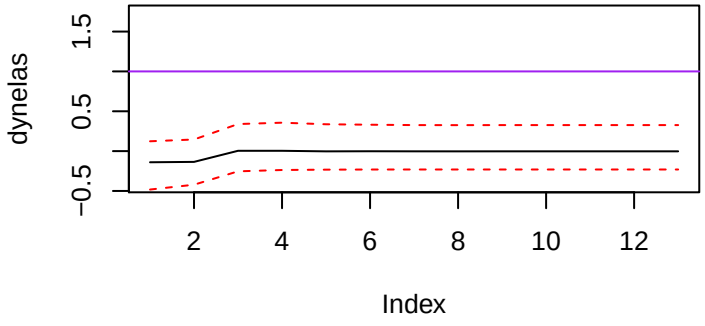
Dynamic Elast: Hard Logs



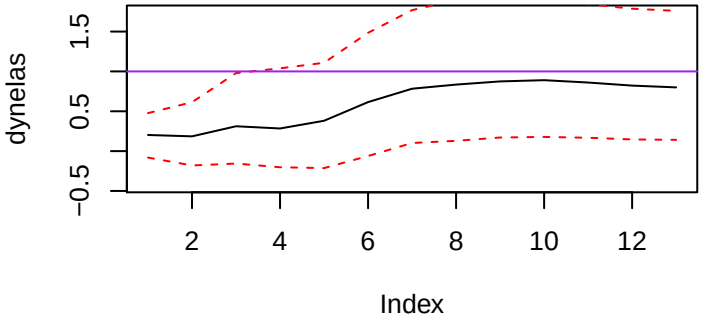
Dynamic Elast: Hard Sawnwood

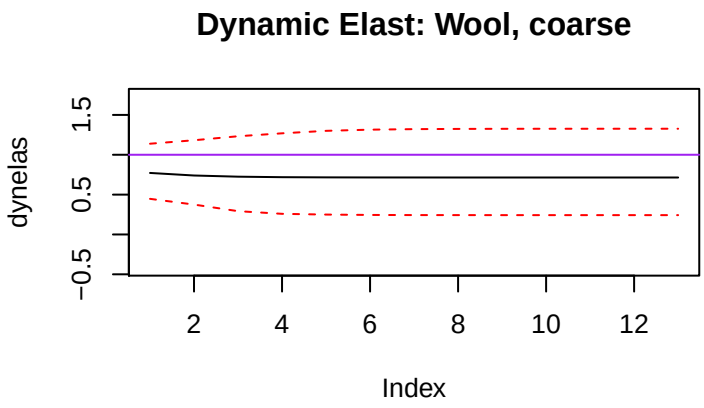


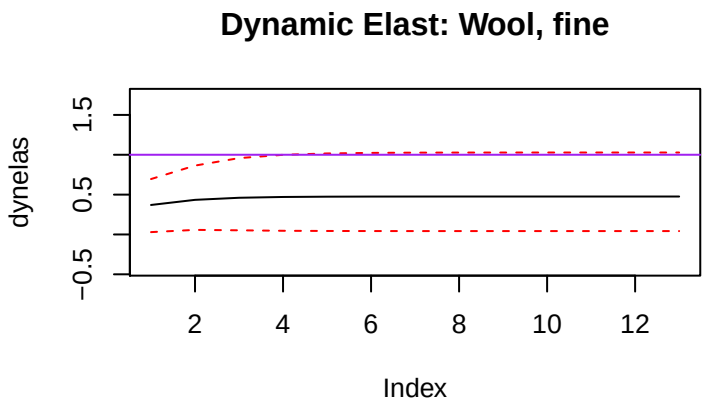
Dynamic Elast: Soft Sawnwood

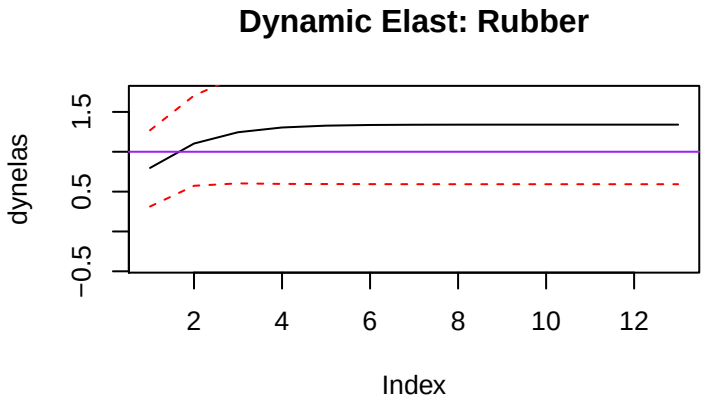


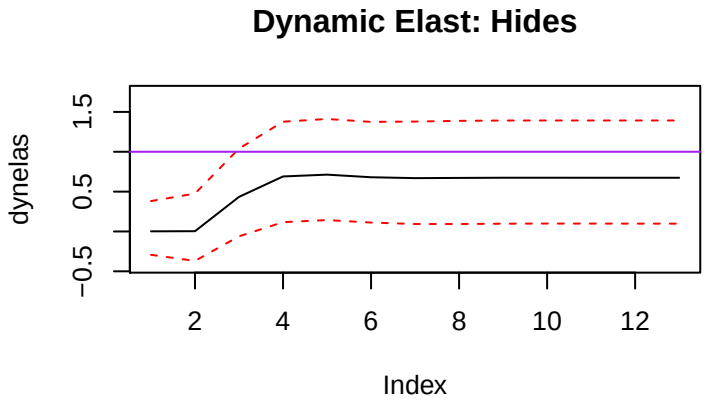
Dynamic Elast: Cotton

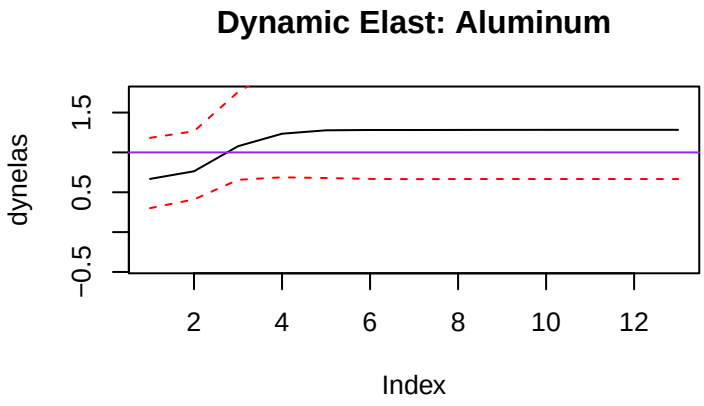


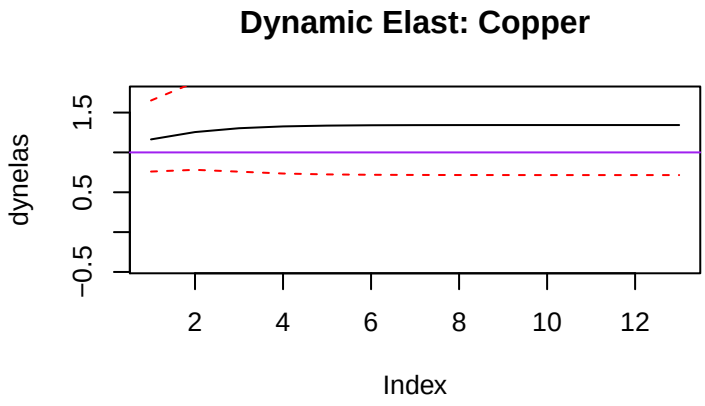

Dynamic Elast: Wool, coarse
dynelas
−0.5
0.5
1.5
2
4
6
8
10
12
Index

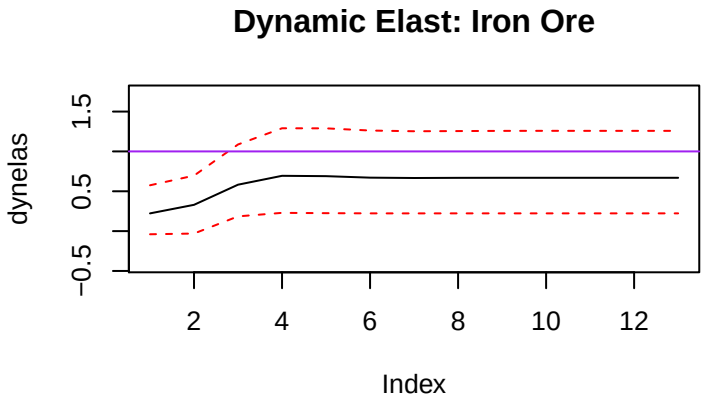

Dynamic Elast: Wool, fine
dynelas
−0.5
0.5
1.5
2
4
6
8
10
12
Index

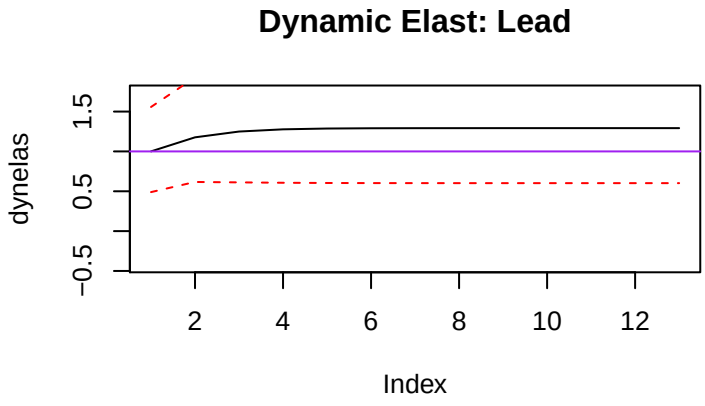

Dynamic Elast: Rubber
dynelas
−0.5
0.5
1.5
2
4
6
8
10
12
Index

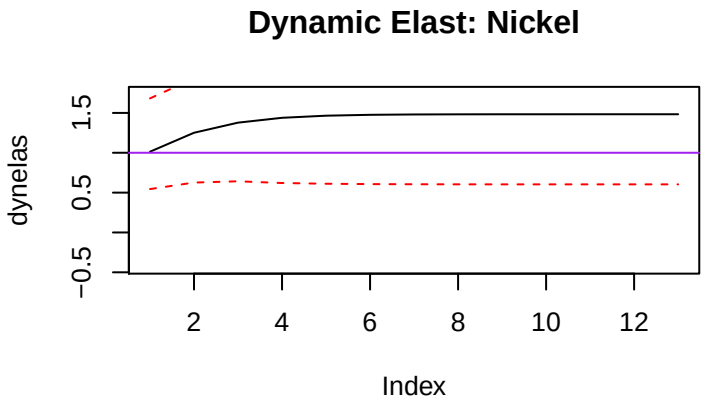

Dynamic Elast: Hides
dynelas
−0.5
0.5
1.5
2
4
6
8
10
12
Index

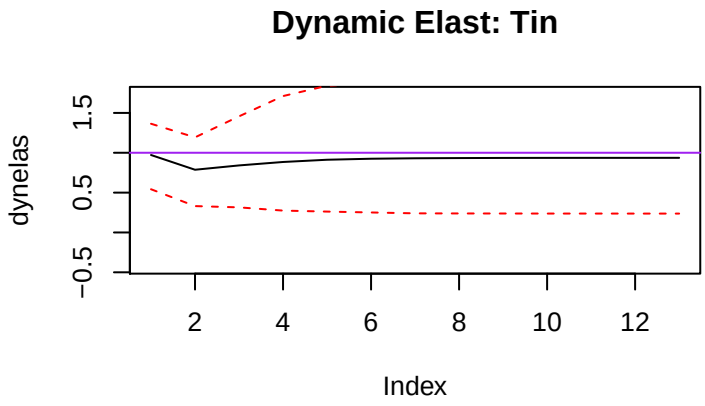

Dynamic Elast: Aluminum
dynelas
−0.5
0.5
1.5
2
4
6
8
10
12
Index

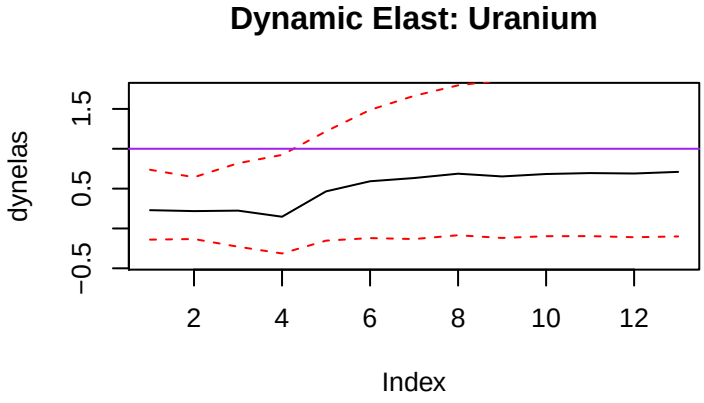

Dynamic Elast: Copper
dynelas
−0.5
0.5
1.5
2
4
6
8
10
12
Index

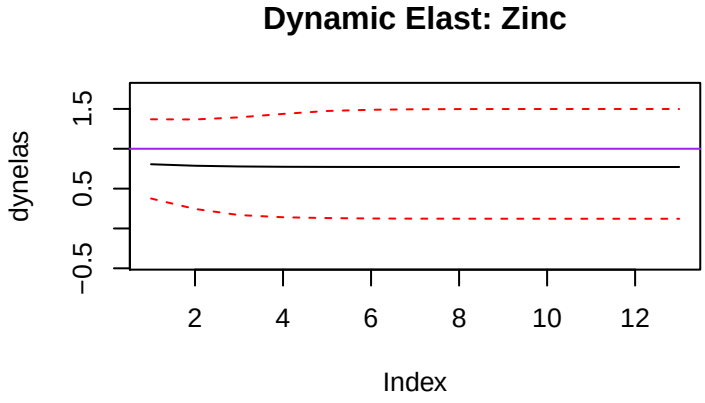

Dynamic Elast: Iron Ore
dynelas
−0.5
0.5
1.5
2
4
6
8
10
12
Index

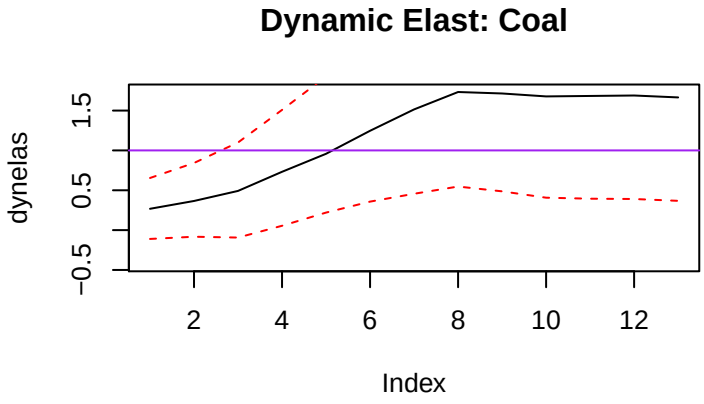

Dynamic Elast: Lead
dynelas
−0.5
0.5
1.5
2
4
6
8
10
12
Index

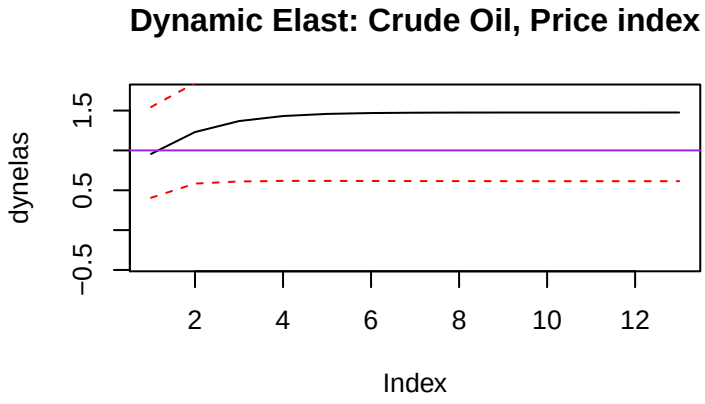

Dynamic Elast: Nickel
dynelas
−0.5
0.5
1.5
2
4
6
8
10
12
Index

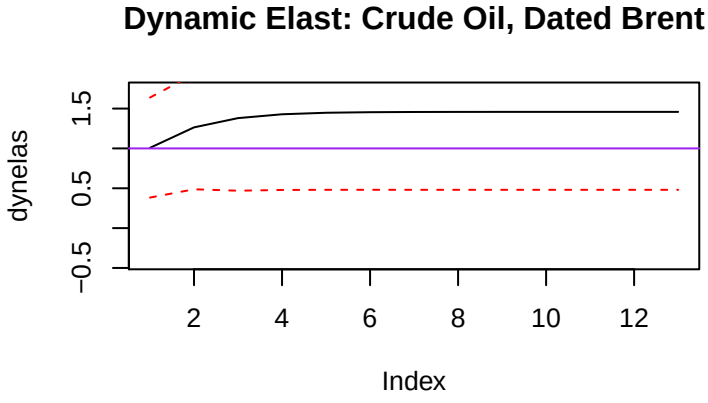

Dynamic Elast: Tin
dynelas
−0.5
0.5
1.5
2
4
6
8
10
12
Index

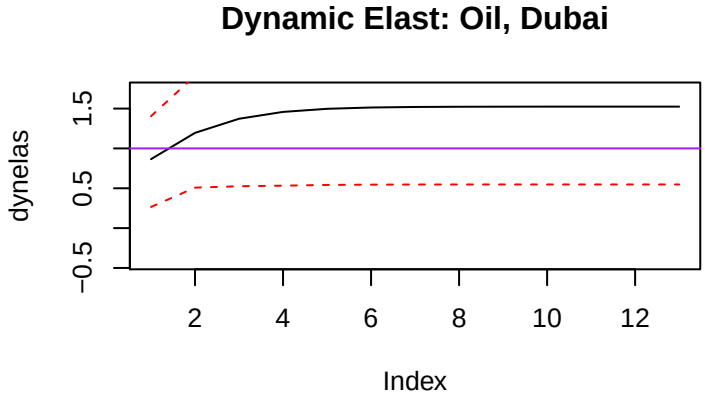

Dynamic Elast: Uranium
dynelas
−0.5
0.5
1.5
2
4
6
8
10
12
Index


Dynamic Elast: Zinc
dynelas
−0.5
0.5
1.5
2
4
6
8
10
12
Index

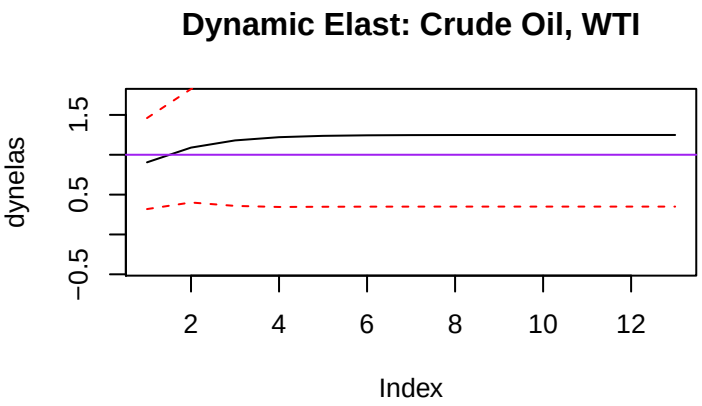

Dynamic Elast: Coal
dynelas
−0.5
0.5
1.5
2
4
6
8
10
12
Index


Dynamic Elast: Crude Oil, Price index
dynelas
−0.5
0.5
1.5
2
4
6
8
10
12
Index


Dynamic Elast: Crude Oil, Dated Brent
dynelas
−0.5
0.5
1.5
2
4
6
8
10
12
Index


Dynamic Elast: Oil, Dubai
dynelas
−0.5
0.5
1.5
2
4
6
8
10
12
Index

```
#end============================================================
```


Dynamic Elast: Crude Oil, WTI
dynelas
1.5
0.5
−0.5
2
4
6
8
10
12
Index