In [567…

```
# Factor analysis based on R code from:
# [https://github.com/martinbaumgaertner/hfdshocks]
# ADAPTATIONS:
# --> code structure, data preparation, graphics and notation
# --> Eigenvectors & rotation analysis
# --> Analysis of factors through time (barplot)
# --> Additional sovereign spread factor
# --> Linear regression analysis of found factors on OIS curve,
#     with coefficients and percentage explained variance (R-squared)
# --> variance share of the (rotated) principal componenents
#
# Linear Regression of OIS2Y on the conference factors for classification in
# [http://www.bilkent.edu.tr/~refet/ABGMR_replication_files.zip]
# ADAPTATIONS:
# --> code structure, data preparation, graphics and notation
# --> new estimation period for Odyssean vs Delphic analysis
#
# The instrumental VAR analysis is based on Julia code from
# [http://www.bilkent.edu.tr/~refet/ABGMR_replication_files.zip]
# ADAPTATIONS:
# --> Translation to R
# --> code structure, data preparation, graphics and notation
# --> new estimation periods
# --> additional instrument: sovereign spread factor
```

In [568…

```
library(dplyr)
library(readr)
library(NlcOptim)
library(tidyr)
library(data.table)
library(purrr)
library(sandwich)
library(Matrix)
library(ggplot2)
library(zoo)
library(vars)
library(expm)
library(lubridate)
library(reshape2)
library(yahoofinancer)
```

# Factor analysis

In [569…

```
plot_eigenvalues <- function(values){
    plot(values, type = "b", pch = 16, col = "steelblue", lty = 1, lwd = 2,
         main = "Eigenvalues Plot", xlab = "Principal Component", ylab = "Ei
}

plot_eigenvectors <- function(title = "Eigenvectors", vectors = NULL, vector
  i = length(vector_names)
  n = dim(vectors)[1]

 matplot(1:n, vectors[,1:i], type = "b", pch = 16, col = 1:i, lty = 1, lwd =
          xlim = c(1, n), ylim = c(min(vectors), max(vectors) + 0.1 * max(ve
          main = title, xlab = "", ylab = "Eigenvector Value / Loading", xax

  legend("topright", legend = vector_names[1:i], col = 1:i, pch = 16, lty =
         title = "Principal Component", x.intersp = 1.5, y.intersp = 0.8)  #
```

```
  axis(1, at = 1:n, labels = colnames, las = 2)
  if (!is.null(colnames)) {
    column_names <- colnames[1:n]
  }
  options(repr.plot.width = 7)
}
```

In [570…

```
load_hfd<-function(path, sheet, exclude_date=c("2001-09-17","2008-10-08","20

  url="https://www.ecb.europa.eu/pub/pdf/annex/Dataset_EA-MPD.xlsx"

  # download data if needed
  file_path <- paste0(path, "EAMPD.xlsx")
  if (!file.exists(file_path)) {
    curl::curl_download(url, file_path)
  }

  data<-readxl::read_excel(paste0(path,"EAMPD.xlsx"),sheet = sheet)


  data<-data %>%
    setNames(tolower(names(.)))%>%
    dplyr::mutate(ois_2y = coalesce(ois_2y, de2y)) %>%
    dplyr::mutate(ois_5y = coalesce(ois_5y, de5y)) %>%
    dplyr::mutate(ois_10y = coalesce(ois_10y, de10y))

  if(!is.null(columns)){
    data <- data %>%dplyr::select(c("date", all_of(columns)))
  }

  data <- data %>%filter_at(vars(-date), any_vars(!is.na(.)))

  if(!is.null(range)){
    data<-data %>%
      dplyr::filter(date >= as.POSIXct(range[1],tz="UTC") & date<= as.POSIXc
  }
  if(!is.null(exclude_date)){
    data<-data %>%
      dplyr::filter(!(date %in% as.POSIXct(exclude_date,tz="UTC")))
  }

  return(data)
}
```

In [113…

```
factor_model<-function(data,center=T,scaleZ=T){

  Z<-data %>% select_if(is.numeric)%>%as.matrix()

  Tn<-nrow(Z)
  nn<-ncol(Z)

  # Standardization
  meanZ<-colMeans(Z)
  sdZ<-apply(Z, 2, sd)

  X = sweep(Z,2,meanZ, "-")
  #X = sweep(X, 2, sdZ, "/")


  # Eigenvalue decomposition
  ev<-eigen(t(X)%*%X,only.values=F)
  neg<-which(ev$values<0)
```

```
  if(!length(neg)==0){
    break
  }

  if(TRUE){
    plot_eigenvalues(ev$values)
  }

  # normalized vectors are on a scale of 1/sqrt(nn)
  # redistribute this between factors and vectors
  # for convenience
  vectors <- ev$vectors*sqrt(nn)
  factors<-X%*%ev$vectors/sqrt(nn)
  return(list(factors=factors,vectors=vectors))
}
```

In [113…

```
loadings<-function(ois_data,factors){
  loadings_release<-full_join(ois_data,factors,by="date") %>%
    dplyr::select(-date)%>%
    pivot_longer(cols=colnames(factors)[-1],names_to = "factor",values_to =
    pivot_longer(cols=colnames(ois_data)[-1],names_to = "ois",values_to = "c
    split(list(.$factor,.$ois))%>%
    map(~ lm(ois_value ~ shock, data = .x))%>%
    map( function(u) tibble(coef=coef(u)[2],ser=sqrt(vcovHC(u)[2,2]),r2=summ
    rbindlist(idcol = TRUE ) %>%
    tibble() %>%
    separate(.id,c("shock","ois"),"\\.")# %>%
  #mutate(ois=str_remove(ois,"_release|_conference"))
  return(loadings_release)
}
```

In [116…

```
# whether or not to add the spread
SPREAD = TRUE
```

In [116…

```
rotate<-function(data, factors, vectors, n, crisis_date="2008-09-04",window=

  ois_matrix<-data %>% select_if(is.numeric)%>%as.matrix()
  Tn<-nrow(ois_matrix)
  nn<-ncol(ois_matrix)

  #QE is calculated by minimizing variance before financial crisis
  date_vector<-data %>%pull(date)
  idx_pre<-1:(which(date_vector==as.POSIXlt(crisis_date,tz="UTC"))-1)
  idx_post<-(which(date_vector==as.POSIXlt(crisis_date,tz="UTC"))-1):Tn

  scale<-apply(factors, 2, sd)
  factors<-sweep(factors, 2, scale, "/")[,1:n]
  factors_pre<-factors[idx_pre,]

  # scale vectors by the standard deviation of the factor
  # will determine weight in optimization
  loading <- vectors[,1:n]*scale[1:n]

    # NOTE: x := t(A) where A is your typical rotation matrix y=Ax
    # this is because we are working with column vectors, and we can then
    # perform the rotation as C*x = t(AR) = t(R)*t(A)

  con=function(x){
    f=NULL
    #ORTHONORMAL restrictions
    #normal restrictions
    f=rbind(f,x[1]^2 + x[4]^2 + x[7]^2-1)
    f=rbind(f,x[2]^2 + x[5]^2 + x[8]^2-1)
```

```
    f=rbind(f,x[3]^2 + x[6]^2 + x[9]^2-1)
    #orthogonal restrictions
    f=rbind(f,x[1]*x[2] + x[4]*x[5] + x[7]*x[8]-0)
    f=rbind(f,x[1]*x[3] + x[4]*x[6] + x[7]*x[9]-0)
    f=rbind(f,x[2]*x[3] + x[5]*x[6] + x[8]*x[9]-0)
    #rotated base vectors are V
    f=rbind(f,x[4]*loading[1,1] + x[5]*loading[1,2] + x[6]*loading[1,3]-0)
    f=rbind(f,x[7]*loading[1,1] + x[8]*loading[1,2] + x[9]*loading[1,3]-0)
    return(list(ceq=f,c=NULL))
  }

 con2=function(x){
    f=NULL
    #ORTHONORMAL restrictions t(x) * x  = Ix
    #only upper triangle to remove redundant constraints
    n=sqrt(length(x))
    orthonormal = t(x) %*% x - diag(sqrt(length(x)))
    upper_triangle_indices <- upper.tri(orthonormal) | row(orthonormal) == c
    f = rbind(f, matrix(orthonormal[upper_triangle_indices], ncol=1) )

    # compute rotated loadings
    rotated_loadings = loading[,1:n] %*% x
    # vector 2,3 don't load on ois_1m
    f = rbind(f, rotated_loadings[1,2]-0)
    f = rbind(f, rotated_loadings[1,3]-0)
    if(SPREAD){
        # vector 4 doesn't load on ois_1m
        f = rbind(f, rotated_loadings[1,4]-0)
        # vector 2,3 do not load on is_de_spread
        f = rbind(f, rotated_loadings[8,3]-0)
        f = rbind(f, rotated_loadings[8,2]-0)
        #f = rbind(f, rotated_loadings[8,1]-0)
    }


    return(list(ceq=f,c=NULL))
  }

  obj<-function(U){
    # minimize the variance of the third factor in pre-crisis period
    n <- sqrt(length(U))
    xx<-factors_pre[,1:n]%*%U[,3]
    as.numeric(t(xx)%*%xx)
  }


  if(SPREAD){
      n=4
  }else{
      n=3
  }

  sol<-solnl(diag(n),objfun=obj,confun=con2)$par

  rotate_factors<- factors[,1:n] %*% sol %>% as_tibble
  rotate_vectors<-vectors[,1:n] %*% sol %>%as_tibble
  return(list(factors=rotate_factors, vectors=rotate_vectors))
}
```

In [116…

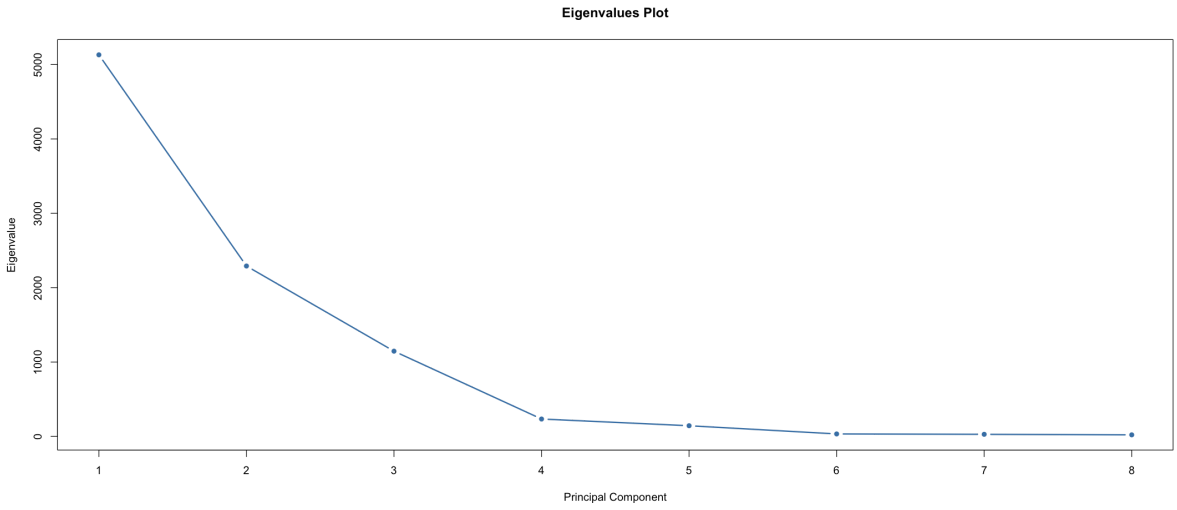
```
scale <- function(data, factors, factor_names, factor_scalers){
  factors<-factors %>%
    dplyr::select(1:length(factor_names)) %>%
    dplyr::rename_with(~factor_names, .cols = everything())
  ois_matrix<-data %>%
```

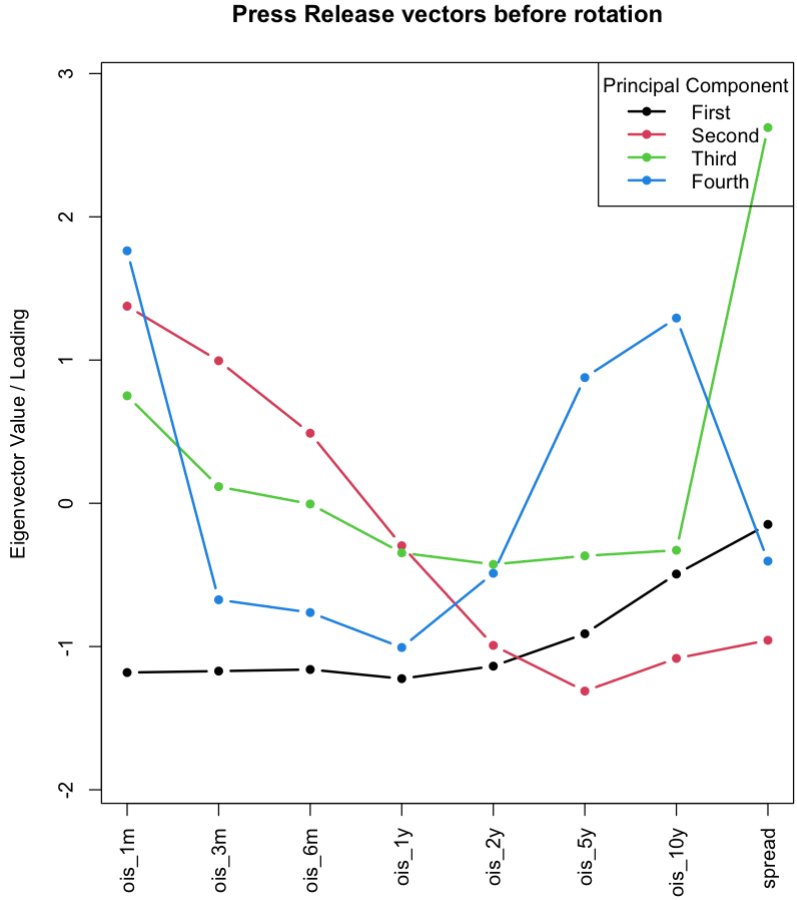
```
    dplyr::select(starts_with(factor_scalers))%>%
    as.matrix()

  full<-bind_cols(data %>%
                    dplyr::select(date),factors,ois_matrix %>% as_tibble(.))

  for (i in seq_along(factor_names)) {
    scale <- coef(lm(as.formula(paste(factor_scalers[i], "~", factor_names[i
    factor_name = factor_names[i]
    factors[factor_name] = factors[factor_name] * scale
    }

  return(bind_cols(data %>%dplyr::select(date),factors))

}
```

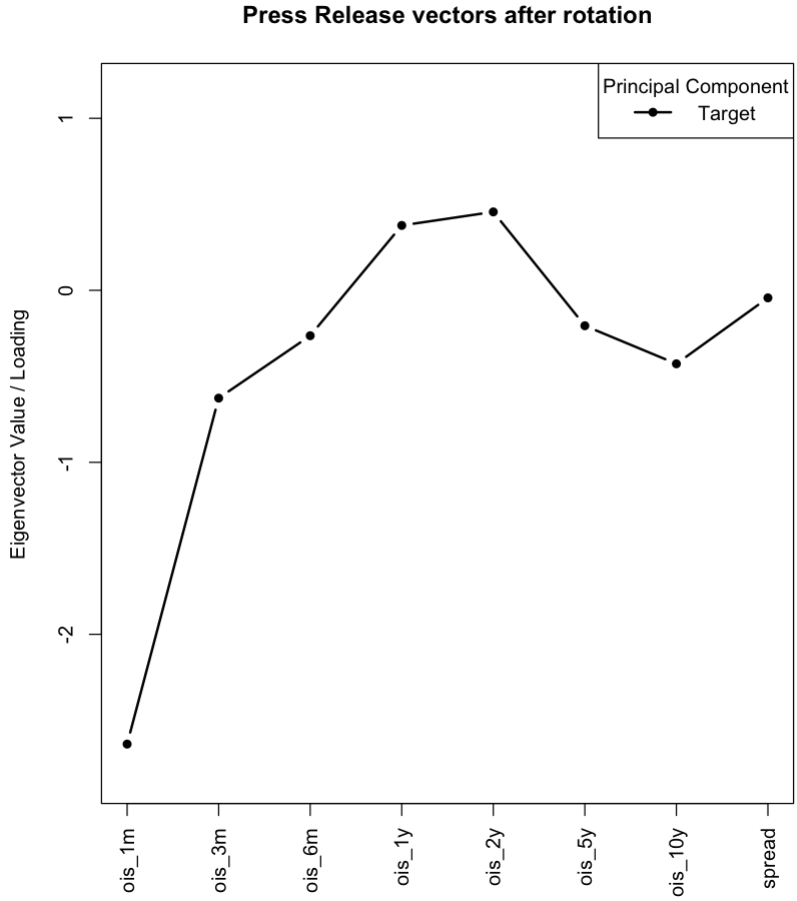
In [116…

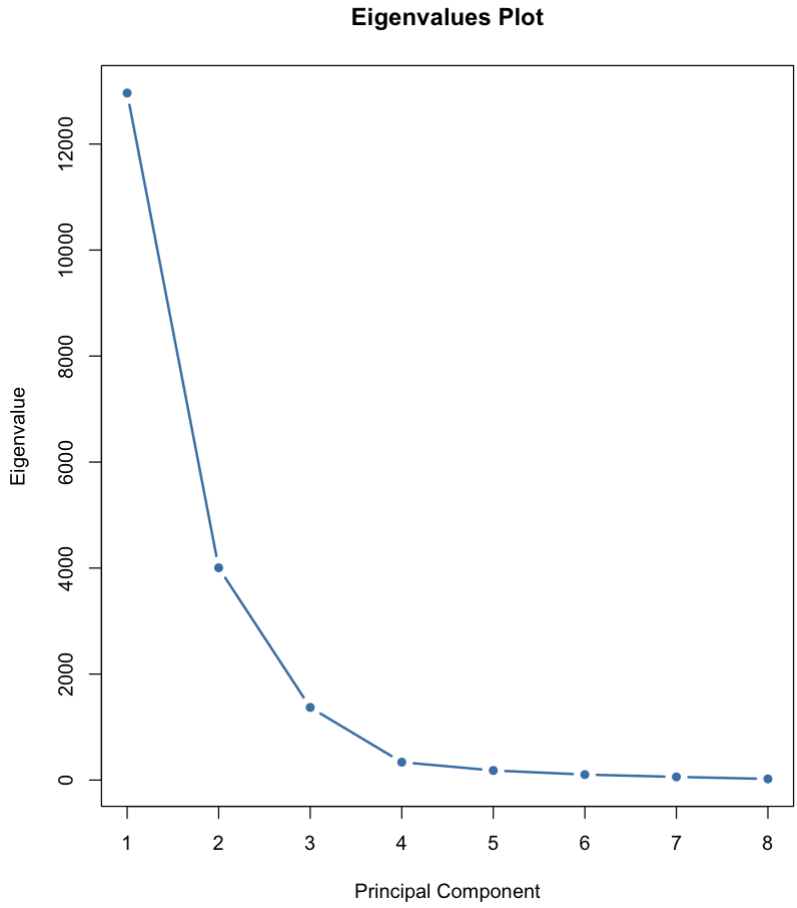
```
path=""
crisis_date="2008-09-04"
range=c("2001-12-31","2023-06-15")
exclude_date=c("2001-09-17","2008-10-08","2008-11-06")


if(SPREAD){

    eampd_variables <- c("ois_1m", "ois_3m", "ois_6m", "ois_1y", "ois_2y", "

    release<-load_hfd(path, sheet="Press Release Window", exclude_date=exclu

    release <- release %>% mutate(spread = +(it2y - de2y)) %>%  dplyr::selec

    vector_names <- c("First", "Second", "Third", "Fourth")

    release_factor_names <- c("Target")
    release_factor_scalers <- c("ois_1m")

    conference<-load_hfd(path, sheet="Press Conference Window", exclude_date
    conference <- conference %>% mutate(spread = +(it2y - de2y)) %>% dplyr::

    conference_factor_names <- c("Timing", "FG", "QE", "SEUR")
    conference_factor_scalers <- c("ois_6m", "ois_2y", "ois_10y", "spread")

    n=4

}else{
    eampd_variables <- c("ois_1m", "ois_3m", "ois_6m", "ois_1y", "ois_2y", "

    release<-load_hfd(path, sheet="Press Release Window", exclude_date=exclu

    vector_names <- c("First", "Second", "Third")
    release_factor_names <- c("Target")
    release_factor_scalers <- c("ois_1m")

    conference<-load_hfd(path, sheet="Press Conference Window", exclude_date
    conference_factor_names <- c("Timing", "FG", "QE")
    conference_factor_scalers <- c("ois_6m", "ois_2y", "ois_10y")

    n=3
}




fm<-factor_model(release)
release_vectors<-fm$vectors
```

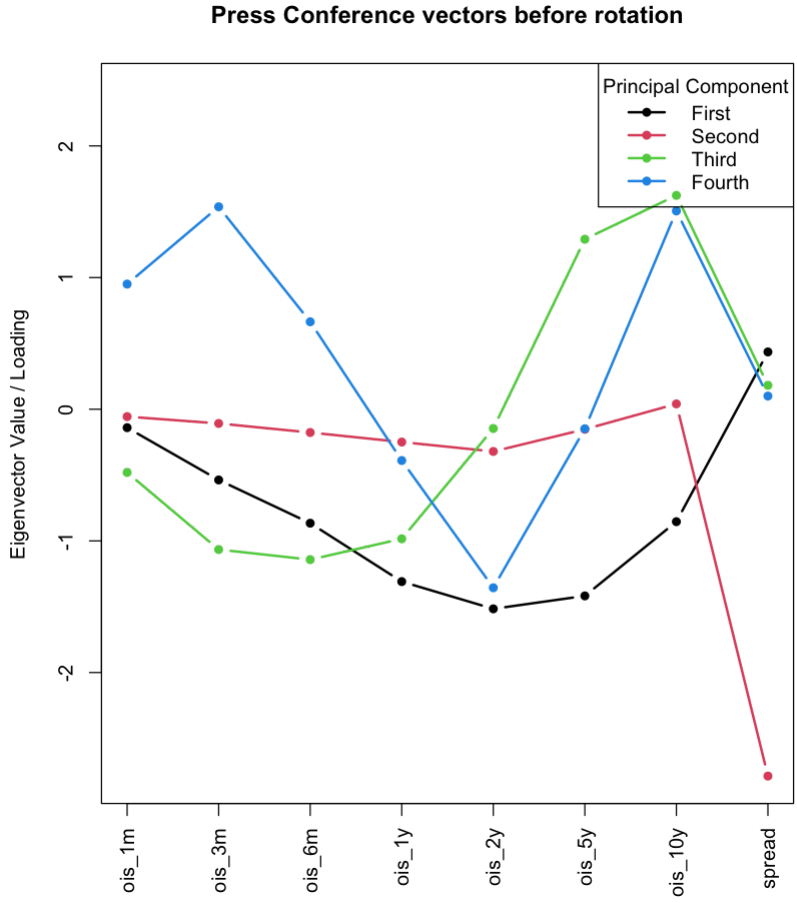
```
release_factors<-fm$factors
r<-rotate(release, release_factors, release_vectors, n, crisis_date=crisis_d
release_factors_rotated = r$factors
release_vectors_rotated = r$vectors
release_factors_scaled<-scale(release, release_factors_rotated, release_fact
loadings_release<-loadings(release,release_factors_scaled)

if(TRUE){
    plot_eigenvectors(title = "Press Release vectors before rotation",
                  vectors=release_vectors,
                  vector_names=vector_names,
                  colnames=names(release)[-1])
    plot_eigenvectors(title = "Press Release vectors after rotation",
                  vectors=release_vectors_rotated,
                  vector_names=release_factor_names,
                  colnames=names(release)[-1])
}


fm<-factor_model(conference)
conference_vectors<-fm$vectors
conference_factors<-fm$factors
r<-rotate(conference, conference_factors, conference_vectors, n, crisis_date
conference_factors_rotated<-r$factors
conference_vectors_rotated<-r$vectors

conference_factors_scaled<-scale(conference, conference_factors_rotated, con
loadings_conference<-loadings(conference,conference_factors_scaled)

options(repr.plot.width = 20)

if(TRUE){
    plot_eigenvectors(title = "Press Conference vectors before rotation",
                  vectors=conference_vectors,
                  vector_names=vector_names,
                  colnames=names(conference)[-1])
    plot_eigenvectors(title = "Press Conference vectors after rotation",
                  vectors=conference_vectors_rotated,
                  vector_names=conference_factor_names,
                  colnames=names(conference)[-1])
}
```

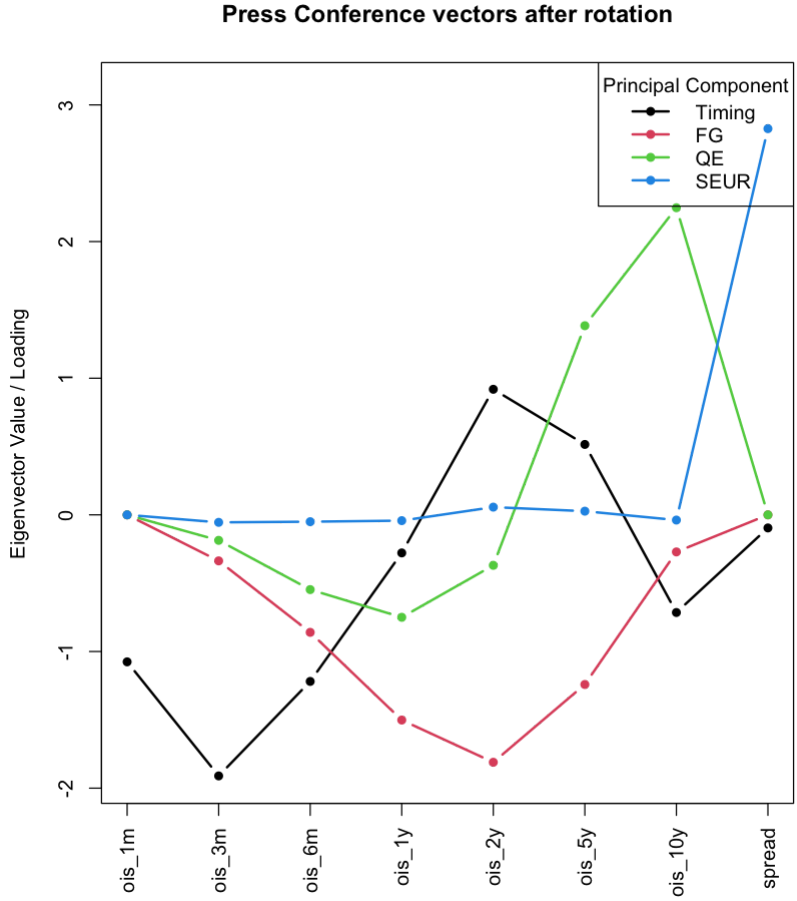



Press Release vectors before rotation
Principal Component
First
Second
Third
Fourth
Eigenvector Value / Loading
3
2
1
0
-1
-2
ois_1m
ois_3m
ois_6m
ois_1y
ois_2y
ois_5y
ois_10y
spread


Press Release vectors after rotation
Principal Component
Target
Eigenvector Value / Loading
1
0
-1
-2
ois_1m
ois_3m
ois_6m
ois_1y
ois_2y
ois_5y
ois_10y
spread


Eigenvalues Plot
Eigenvalue
0
2000
4000
6000
8000
10000
12000
1
2
3
4
5
6
7
8
Principal Component


Press Conference vectors before rotation
Principal Component
First
Second
Third
Fourth
Eigenvector Value / Loading
-2
-1
0
1
2
ois_1m
ois_3m
ois_6m
ois_1y
ois_2y
ois_5y
ois_10y
spread

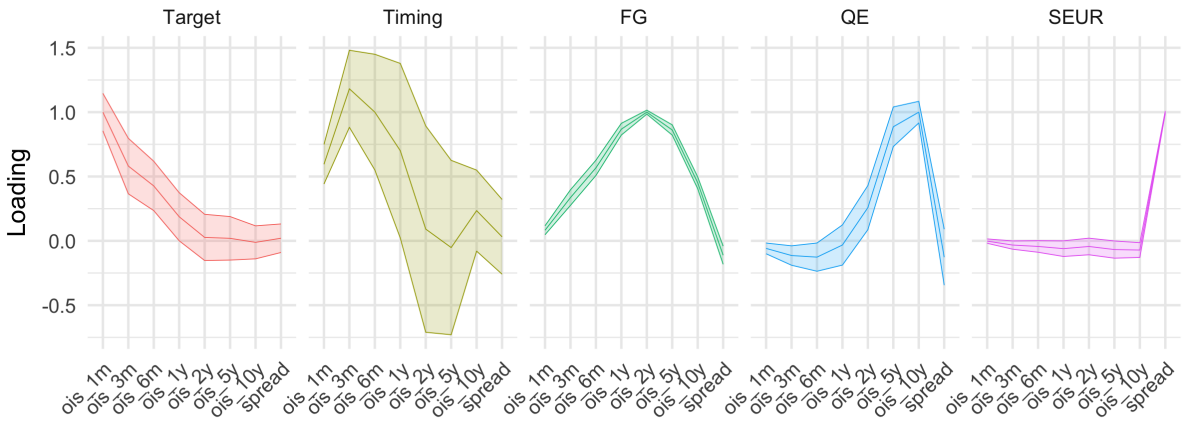



In [116…

```
# WALD test for number of factors
# take analysis from Altavilla2019
```

In [116…

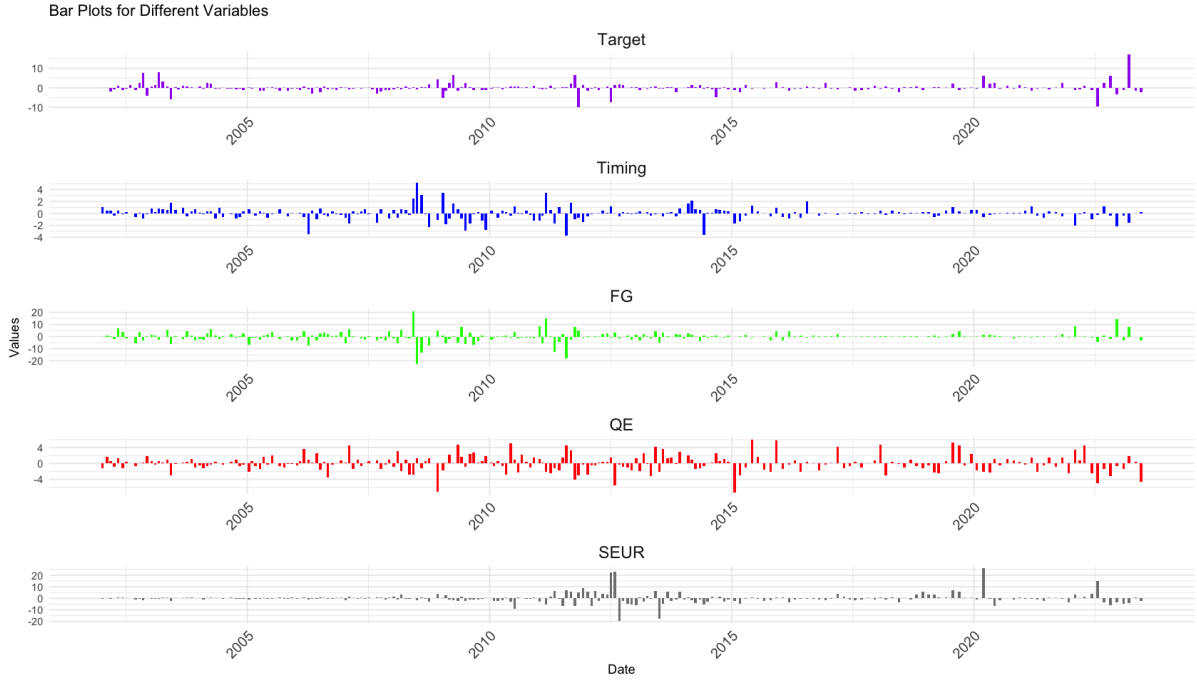
```
load <- rbind(loadings_conference, loadings_release)

load$shock <- factor(load$shock, levels = c("Target", "Timing", "FG", "QE",

load$lower <- load$coef - load$ser

load$lower <- load$coef - load$ser
load$upper <- load$coef + load$ser


options(repr.plot.width = 20)
ggplot(load, aes(x = factor(ois, levels = colnames(release)[-1]), y = coef,
  geom_line() +
  geom_ribbon(aes(ymin = lower, ymax = upper, fill = shock), alpha = 0.2) +
  labs(x = "", y = "Loading") +
  theme_minimal(base_size = 30) +
  facet_wrap(~ shock, ncol = 5) +
  theme(axis.text.x = element_text(angle = 45, hjust = 1),
        legend.position = "none")
```




In [119…

```
factors <- release_factors_scaled %>%
  left_join(conference_factors_scaled, by = "date")

data_melted <- melt(factors, id.vars = "date", variable.name = "Variable")

# Create separate bar plots with facets
ggplot(data_melted, aes(x = date, y = value, fill = Variable)) +
  geom_bar(stat = "identity") +
  facet_wrap(vars(Variable), scales = "free", ncol = 1) +
  labs(title = "Bar Plots for Different Variables",
       x = "Date", y = "Values") +
  scale_fill_manual(values = c("Timing" = "blue", "FG" = "green", "QE" = "re
  theme_minimal() +
  theme(legend.position = "none",
        axis.text.x = element_text(angle = 45, hjust = 1, size = 12),
        strip.text = element_text(size = 14)
       )

options(repr.plot.width = 16)
options(repr.plot.height = 8)
```

```
Warning message:
“Removed 20 rows containing missing values (`position_stack()`).”
```



# Regression Analysis

In [114…

```
regression_tables <- function(factors_scaled, eampd_data, significance_level

    factors <- names(factors_scaled)[-1]
    eampd_variables <- names(eampd_data)[-1]
    factor_data <- factors_scaled[, factors]
    result_table <- data.frame()


    print(c(factors, "R_squared"))

    for (eampd in eampd_variables) {

        formula_obj <- as.formula(paste(eampd, "~", paste(factors, collapse
        model <- lm(formula_obj, data = data.frame(factor_data, eampd_data))

        coeffs <- coef(summary(model))
        coefficient <- round(coeffs[, 1], 2)
        std_error <- round(coeffs[, 2], 2)
        p_value <- coeffs[, 4]
        R_squared <- round(summary(model)$r.squared, 2)

        cat(eampd)
        for(i in seq_along(factors)){
            sig <- ifelse(p_value[i] < 0.001, "***",
                          ifelse(p_value < 0.05, "**",
                          ifelse(p_value < 0.1, "*", "")))
            cat(" ",coefficient[i])
            cat(sig)
            cat(paste0("(",std_error[i], ")"))
        }
        cat(" ", R_squared)
        cat(" \n")
    }

}



extra_vars <- c("stoxx50", "sx7e", "eurusd", "eurgbp", "eurjpy")

extra_release <- load_hfd(path, sheet="Press Release Window", exclude_date=e
release_data <- release %>%left_join(extra_release, by = "date")
release_factors_standard <- release_factors_scaled #%>% mutate(across(colnam


extra_conference <- load_hfd(path, sheet="Press Conference Window", exclude_
conference_data <- conference %>% left_join(extra_conference, by = "date")
conference_factors_standard <- conference_factors_scaled #%>%  mutate(across


res_release <- regression_tables(release_factors_standard, release_data)

res_conf <- regression_tables(conference_factors_standard, conference_data)
```

```
[1] "Target"    "R_squared"
ois_1m  1***(0.04)  0.72
ois_3m  0.58***(0.06)  0.32
ois_6m  0.43***(0.05)  0.22
ois_1y  0.19**(0.06)  0.04
ois_2y  0.03(0.07)  0
ois_5y  0.02(0.06)  0
ois_10y  -0.01(0.05)  0
spread  0.02(0.07)  0
stoxx50  -0.04***(0.01)  0.07
sx7e  -0.03*(0.02)  0.02
eurusd  0.01(0.01)  0.01
eurgbp  0.01**(0.01)  0.03
eurjpy  0.01(0.01)  0.01
[1] "Timing"    "FG"        "QE"        "SEUR"      "R_squared"
ois_1m  0.59***(0.04)  0.08***(0.01)  -0.06**(0.02)  0**(0.01)  0.53
ois_3m  1.18***(0.03)  0.34***(0.01)  -0.11***(0.01)  -0.03***(0.01)  0.95
ois_6m  1***(0.02)  0.57***(0.01)  -0.13***(0.01)  -0.04***(0.01)  0.98
ois_1y  0.7***(0.03)  0.87***(0.01)  -0.03**(0.01)  -0.06***(0.01)  0.99
ois_2y  0.09**(0.03)  1***(0.01)  0.26***(0.01)  -0.04***(0.01)  0.99
ois_5y  -0.05(0.04)  0.86***(0.01)  0.89***(0.02)  -0.07***(0.01)  0.98
ois_10y  0.23***(0.03)  0.45***(0.01)  1***(0.02)  -0.07***(0.01)  0.97
spread  0.03**(0.01)  -0.11***(0)  -0.13***(0.01)  1***(0)  1
stoxx50  0.02(0.04)  -0.02(0.01)  -0.01(0.02)  -0.06***(0.01)  0.22
sx7e  0.04(0.06)  -0.02(0.01)  0.09(0.03)  -0.15***(0.01)  0.37
eurusd  0.06**(0.02)  0.05***(0.01)  0.06***(0.01)  -0.01**(0.01)  0.33
eurgbp  0.06***(0.02)  0.04***(0)  0.04***(0.01)  0**(0)  0.35
eurjpy  0.08***(0.02)  0.05***(0.01)  0.08***(0.01)  -0.01**(0)  0.45
```

In [118…

```
release_factors_standard_2018 <- release_factors_standard %>% filter(date >=
release_data_2018 <- release_data %>% filter(date >= as.Date("2018-09-01"))
conference_factors_standard_2018 <- conference_factors_standard %>% filter(d
conference_data_2018 <- conference_data %>% filter(date >= as.Date("2018-09-

res_release <- regression_tables(release_factors_standard_2018, release_data
res_conf <- regression_tables(conference_factors_standard_2018, conference_d
```

```
[1] "Target"    "R_squared"
ois_1m  0.91***(0.12)  0.6
ois_3m  0.4**(0.18)  0.12
ois_6m  0.2(0.17)  0.03
ois_1y  -0.19(0.19)  0.03
ois_2y  -0.45**(0.19)  0.13
ois_5y  -0.45**(0.17)  0.16
ois_10y  -0.34**(0.12)  0.18
spread  -0.13(0.2)  0.01
stoxx50  -0.02(0.02)  0.02
sx7e  -0.02(0.04)  0
eurusd  -0.02*(0.01)  0.07
eurgbp  -0.01(0.01)  0.02
eurjpy  -0.03**(0.01)  0.13
[1] "Timing"    "FG"        "QE"        "SEUR"      "R_squared"
ois_1m  0.25*(0.15)  -0.03*(0.03)  0.05*(0.04)  0*(0.01)  0.29
ois_3m  0.94***(0.11)  0.21***(0.02)  -0.07**(0.03)  -0.02**(0.01)  0.76
ois_6m  1.2***(0.14)  0.64***(0.03)  -0.18***(0.04)  -0.05***(0.01)  0.94
ois_1y  0.99***(0.12)  0.99***(0.03)  -0.09**(0.03)  -0.09***(0.01)  0.98
ois_2y  -0.22**(0.1)  0.9***(0.02)  0.37***(0.03)  -0.03***(0.01)  0.99
ois_5y  -0.17(0.12)  0.75***(0.03)  0.87***(0.03)  -0.07***(0.01)  0.99
ois_10y  0.37**(0.1)  0.55***(0.02)  1.01***(0.03)  -0.08***(0.01)  0.99
spread  0.09(0.07)  -0.08***(0.02)  -0.15***(0.02)  1.01***(0.01)  1
stoxx50  -0.23(0.16)  -0.13***(0.04)  0.08(0.04)  -0.06***(0.01)  0.51
sx7e  -0.58(0.38)  -0.15(0.08)  0.34(0.1)  -0.14***(0.03)  0.49
eurusd  -0.03(0.11)  0.04(0.03)  0.05(0.03)  -0.02(0.01)  0.32
eurgbp  0.09(0.07)  0.05(0.02)  0.03(0.02)  0(0.01)  0.45
eurjpy  -0.02(0.09)  0.05(0.02)  0.09***(0.02)  0(0.01)  0.6
```

In [114…

```
variance_shares <- function(data, vectors, factors, n){
    data <- as.matrix(data)
    total_variance <- sum(diag(data %*% t(data)))
    shares <- numeric(max(ncol(vectors), ncol(factors)))
    for (i in 1:max(ncol(vectors), ncol(factors))) {
      if(is.null(factors)){
            factors <- data %*% vectors
      }
      shares[i] <- sum(factors[, i]^2) / total_variance
    }
    return(shares[1:n])
}

release_shares_v <- variance_shares(release[-1], release_vectors, NULL, n)
release_shares_f <- variance_shares(release[-1], NULL, release_factors_scale
conference_shares_v <- variance_shares(conference[-1], conference_vectors, N
conference_shares_f <- variance_shares(conference[-1], NULL, conference_fact


release_shares_v
release_shares_f
conference_shares_v
conference_shares_f
```

4.5262897251739 · 2.04934797260617 · 1.02023353449928 · 0.204655081792015

0.129927342745991 · <NA> · <NA> · <NA>

5.44603120368689 · 1.68235917776113 · 0.574978146522125 · 0.141552752177034

0.0121579970972591 · 0.198466003560542 · 0.0485468691732608 · 0.216921306714198

In [114…

```
get_conference_factors <- function() {
    #tData <- read_csv("vardailydataset.csv",  show_col_types = FALSE)
```

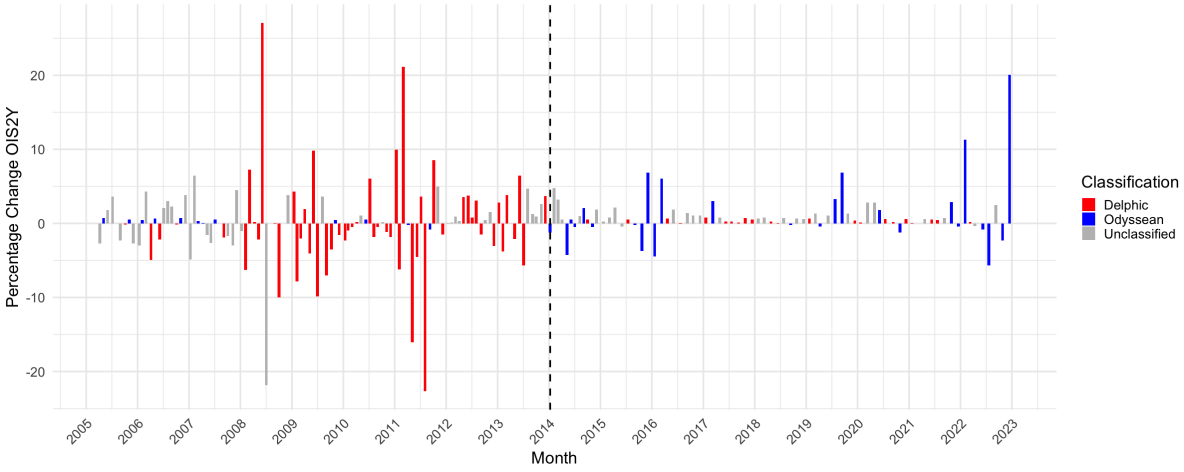
```
    #return(tData[,"ILS2Y"])
    return(conference_factors_scaled)
}

get_release_factors <- function() {
    #tData <- read_csv("vardailydataset.csv",  show_col_types = FALSE)
    #return(tData[,"ILS2Y"])
    return(release_factors_scaled)
}

get_ils <- function() {
    #tData <- read_csv("vardailydataset.csv",  show_col_types = FALSE)
    #colnames(tData)[colnames(tData) == "Date"] <- "date"
    #return(na.omit(tData[,c("date", "ILS2Y")]))
    return(read_csv("ILS.csv",  show_col_types = FALSE))
}

get_ois <- function() {
    #tData <- read_csv("vardailydataset.csv",  show_col_types = FALSE)
    #colnames(tData)[colnames(tData) == "Date"] <- "date"
    r#return(tData[,c("date", "OIS2Y")])
    return(read_csv("OIS.csv",  show_col_types = FALSE))
    }

get_eurusd <- function() {
    #tData <- read_csv("vardailydataset.csv",  show_col_types = FALSE)
    #colnames(tData)[colnames(tData) == "Date"] <- "date"
    #return(na.omit(tData[,c("date", "EURUSD")]))
    return(read_csv("EURUSD.csv",  show_col_types = FALSE))
    }

get_stoxx <- function() {
    #tData <- read_csv("vardailydataset.csv",  show_col_types = FALSE)
    #colnames(tData)[colnames(tData) == "Date"] <- "date"
    #return(na.omit(tData[,c("date", "STOXX")]))
    return(read_csv("STOXX.csv",  show_col_types = FALSE))
    }

conference_factors <- get_conference_factors()
release_factors <- get_release_factors()
ils_data <- get_ils()
ois_data <- get_ois()
stoxx_data <- get_stoxx()
eurusd_data <- get_eurusd()

base_dataset <- ils_data %>%
  left_join(release_factors, by = "date") %>%
  left_join(conference_factors, by = "date") %>%
  left_join(ois_data, by = "date") %>%
  left_join(stoxx_data, by = "date") %>%
  left_join(eurusd_data, by = "date") %>%
  mutate(
    DOIS2Y =  round(OIS2Y - lag(OIS2Y), 2),
    DILS2Y =  round(ILS2Y - lag(ILS2Y), 2),
    LOGSTOXX = 100*log(STOXX),
    DLOGSTOXX = round(LOGSTOXX - lag(LOGSTOXX), 2),
    LOGEURUSD = 100*log(EURUSD),
    date = as.Date(ils_data$date, format = "%Y-%m-%d")
    )

cols_a = c("date", "DOIS2Y", "DILS2Y", "DLOGSTOXX",  colnames(conference_fac
announcements = na.omit(base_dataset)[, cols_a]
```

```
cols_v = c(c("date", "OIS2Y", "ILS2Y", "LOGSTOXX", "LOGEURUSD"), colnames(re
var_daily = base_dataset %>%
    mutate(
    across(colnames(conference_factors)[-1], ~ coalesce(., 0)),
    across(colnames(release_factors)[-1], ~ coalesce(., 0)))
var_daily <- var_daily[, cols_v]
```

In [117…

```
#announcements <- na.omit(read_csv("ilsdailydata.csv"))
#colnames(announcements) <- c("date", "DOIS2Y", "DILS2Y", "DLOGSTOXX", "Timi


mp_pre <- c( fitted(lm(DOIS2Y~Timing+FG, data = announcements %>% filter(dat
mp_btw <- fitted(lm(DOIS2Y~Timing+FG+QE, data = announcements %>% filter(dat
mp_pst <- fitted(lm(DOIS2Y~Timing+FG+QE, data = announcements %>% filter(dat
mp <- c(mp_pre, mp_btw, mp_pst)

ils_pre <- c( fitted(lm(DILS2Y~Timing+FG, data = announcements %>% filter(da
ils_btw <- fitted(lm(DILS2Y~Timing+FG+QE, data = announcements %>% filter(da
ils_pst <- fitted(lm(DILS2Y~Timing+FG+QE, data = announcements %>% filter(da
ils <- c(ils_pre, ils_btw, ils_pst)

sto_pre <- c(fitted(lm(DLOGSTOXX~Timing+FG, data = announcements %>% filter(
sto_btw <- fitted(lm(DLOGSTOXX~Timing+FG+QE, data = announcements %>% filter
sto_pst <- fitted(lm(DLOGSTOXX~Timing+FG+QE, data = announcements %>% filter
sto <- c(sto_pre, sto_btw, sto_pst)


df <- announcements %>% mutate(DOIS2Y = mp*100, DILS2Y = ils*100, DLOGSTOXX

df <- df[, c("date", "DOIS2Y", "DILS2Y", "DLOGSTOXX")]
df <-  na.omit(df)


classify_as_odyssean_delphic <- function(x) {
  a=x["DOIS2Y"]
  b=x["DILS2Y"]
  c=x["DLOGSTOXX"]
  if ((a>=0&b>=0&c>=0) | (a<=0&b<=0&c<=0)) {
    return("Delphic")
  }else if ((a>=0&b<=0&c<=0) | (a<=0&b>=0&c>=0)) {
    return("Odyssean")
  } else {
    return("Unclassified")
  }
}


df$Classification <- apply(df[,  c("DOIS2Y", "DLOGSTOXX", "DILS2Y")], 1, cla

bar_chart <- ggplot(df, aes(x = date, y = DOIS2Y, fill = Classification)) +
  geom_bar(stat = "identity", width = 20) +
  labs(x = "Month", y = "Percentage Change OIS2Y",
       fill = "Classification") +
  scale_fill_manual(values = c("Delphic" = "red", "Odyssean" = "blue", "Uncl
  theme_minimal(base_size = 20) +
  theme(axis.text.x = element_text(angle = 45, hjust = 1)) +
  geom_vline(xintercept = as.Date("2014-01-09"), color = "black", linetype =
  #geom_vline(xintercept = as.Date("2008-01-01"), color = "black", linetype
  scale_x_date(date_breaks = "1 year", date_labels = "%Y")


options(repr.plot.width = 20)
```

```
print(bar_chart)
```




# Instrumental Var Analysis

In [115…

```
# Creation of the VAR dataset from replication files from
# Sovereign spreads and unconventional monetary policy in the Euro area: A t
# and OIS data from ECB databank
# This allows to have data up to end 2020, including covid period.
#tData <- read_delim("endog_vars_daily.csv", delim = ",", show_col_types = F
#tData <- tData %>% mutate(date = as.Date(paste(year, month, day, sep = "-")
#tData
```

In [117…

```
irf_ext <- function(y, Z, p, H) {
  T <- nrow(y)
  K <- ncol(y)
  T_z <- nrow(Z)
  K_z <- ncol(Z)

  Y <- t(y[(p+1):T, ])
  X <- matrix(1, nrow = 1, ncol = T-p)

  for (j in 1:p) {
    W <- t(y[(p+1-j):(T-j), ])
    X <- rbind(X, W)
  }

  ZZ <- Z[(p+1):T_z, ]
  B <- (Y %*% t(X)) %*% solve(X %*% t(X))
  U <- Y - B %*% X
  Sigma <- U %*% t(U) / (T_z - K * p - 1)

  # Z might have fewer observations
  # Thus we have to keep this into considerations
  Sigma_eta_Z <- U[, (T - T_z + 1): dim(U)[2]] %*% ZZ
  Sigma_eta_1_Z <- Sigma_eta_Z[1, ] # rescaling factor
  H1 <- Sigma_eta_Z %*% t(Sigma_eta_1_Z) / (Sigma_eta_1_Z %*% t(Sigma_eta_1_
  scale <- Sigma_eta_Z[1, ]

  A0inv <- cbind(H1, matrix(0, nrow = K, ncol = K-1))
  A <- rbind(B[, -1], cbind(diag(K*(p-1)), matrix(0, nrow = K*(p-1), ncol =
  J <- cbind(diag(K), matrix(0, nrow = K, ncol = K*(p-1)))

  #print(all(abs(eigen(A)$values) < 1)) # STABILITY ?
  IRF <- matrix(A0inv[, 1], nrow = K)
  for (h in 1:H) {
```

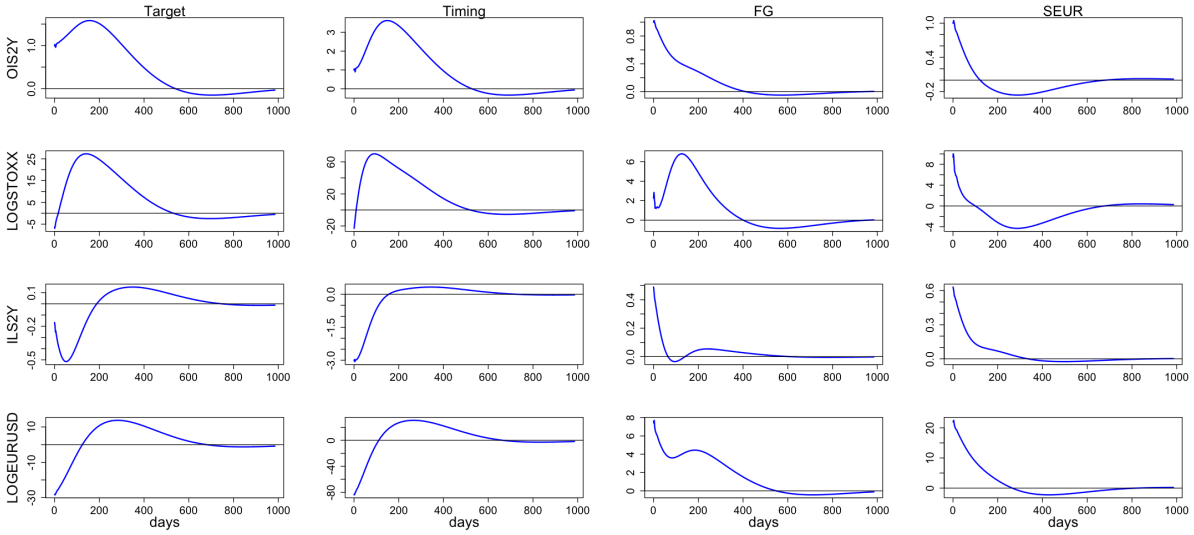
```
    C <- J %*% (A%^%h) %*% t(J)
    if(h%%500==0){
            #print(A^h)
        }
    IRF <- cbind(IRF, (C %*% A0inv[, 1]))
  }

  return(list(IRF =t(IRF), A0inv = A0inv, B = B, U = U, scale = scale))
}


library(readr)
library(data.table)


irf <- function(shock= "Target", variables, tData, start="2014-01-03", end=N

    start_qe <- which(tData$date == start)[1]
    end_qe <- which(tData$date == end)[1]

    start_qe <- which.min(abs(as.Date(tData$date) - as.Date(start)))
    end_qe <- which.min(abs(as.Date(tData$date) - as.Date(end)))

    if(shock=="Target"){
        print(c("start", as.character(tData$date[start_qe]),
                "end", as.character(tData$date[end_qe])))
    }


    X <- as.matrix(tData[start_qe:end_qe, variables])
    Z <- as.matrix(tData[start_qe:end_qe,][,shock])

    IRF_result <- irf_ext(X, Z, p, H)
    IRF <- IRF_result$IRF
    A0inv <- IRF_result$A0inv
    B <- IRF_result$B
    U <- IRF_result$U
    return(IRF)
}


plot_irf <- function(tData, shocks, variables, estimation_start, estimation_
  ncol <- length(shocks)
  nrow <- length(variables)
  par(mfcol = c(nrow, ncol), mar = c(4, 6, 2, 1))
  par(cex.axis = 1.5, cex.main = 5)

  # Loop through each shock
  for (j in seq_along(shocks)) {

    shock <- shocks[j]
    IRF <- irf(shock = shock, variables, tData, start = estimation_start, en
    irf_df <- as.data.frame(cbind(seq(0, nrow(IRF) - 1), IRF))

    colnames(irf_df) <- c(c("date"), variables)


    # Normalize the IRF so that the effect of the shock on the OIS2Y
    # matches the found coefficient in the regression
    if (shock == "Target") {
        irf_df[-1] <- irf_df[-1] / irf_df[1, "OIS2Y"]
    } else if (shock == "Timing") {
        irf_df[-1] <- irf_df[-1] / irf_df[1, "OIS2Y"]
    } else if (shock == "FG") {
```

```
        irf_df[-1] <- irf_df[-1] / irf_df[1, "OIS2Y"]
    } else if (shock == "QE") {
        irf_df[-1] <- irf_df[-1] / irf_df[1, "OIS2Y"]
    } else if (shock == "SEUR") {
        irf_df[-1] <- irf_df[-1] / irf_df[1, "OIS2Y"]
    }


    for (i in seq_along(variables)) {
        var <- variables[i]
        ylim = c(min(irf_df[,var]), max(irf_df[,var]))
        plot(irf_df[,"date"], irf_df[,var], type = "l", col = color, lwd=2,
        abline(h = 0, col = "black", lwd = 1)

        if(j == 1){
            mtext(var, side = 2, line = 3, cex=1.3)
        }
        if(i == 1){
           mtext(shock, side = 3, line = 0, cex=1.3)
        }
        if(i == nrow ){
            mtext("days", side = 1, line = 2.5, cex=1.3)
        }
    }

  }
}
```

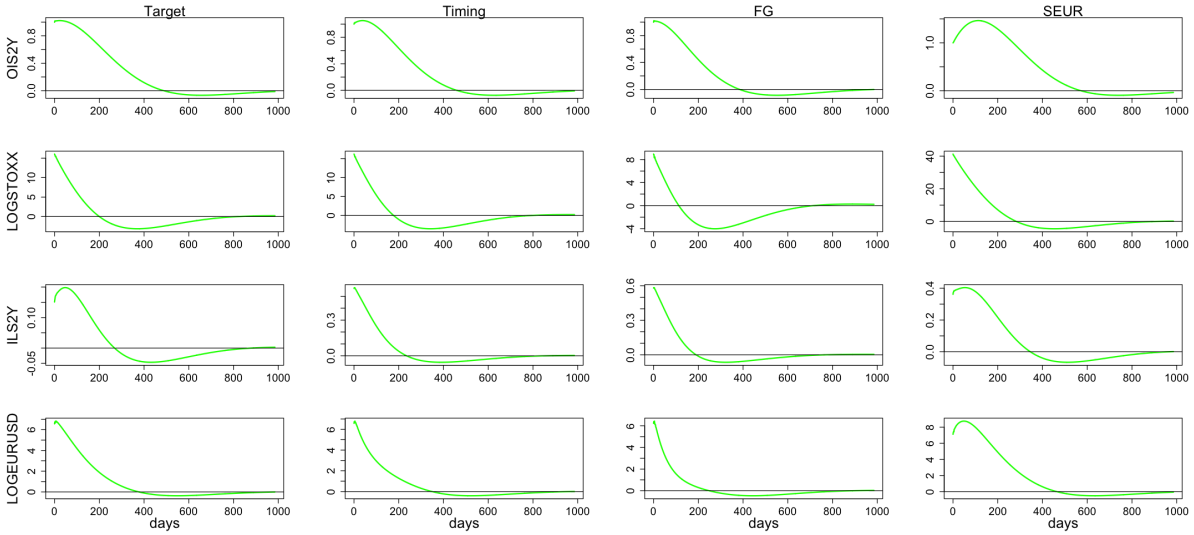
In [117…

```
# ORIGINAL DATA FROM ALTAVILLA
tData <- read_csv("vardailydataset.csv",  show_col_types = FALSE)
tData[,"date"] = tData[,"Date"]
data <- tData

# NEW DATA
data <- na.omit(var_daily)

options(repr.plot.width = 18)

shocks <- c("Target", "Timing", "FG", "SEUR")
shocks_qe <- c("Target", "Timing", "FG", "QE", "SEUR")
variables <- c("OIS2Y", "LOGSTOXX", "ILS2Y", "LOGEURUSD")

plot_irf(data, shocks, variables,  "2001-04-03", "2008-01-03", "blue")
plot_irf(data, shocks, variables, "2008-01-03", "2014-01-01", "green")
plot_irf(data, shocks_qe, variables, "2014-01-03", "2018-09-13", "red")
plot_irf(data, shocks_qe, variables, "2018-09-13", "2023-01-01", "orange")
```

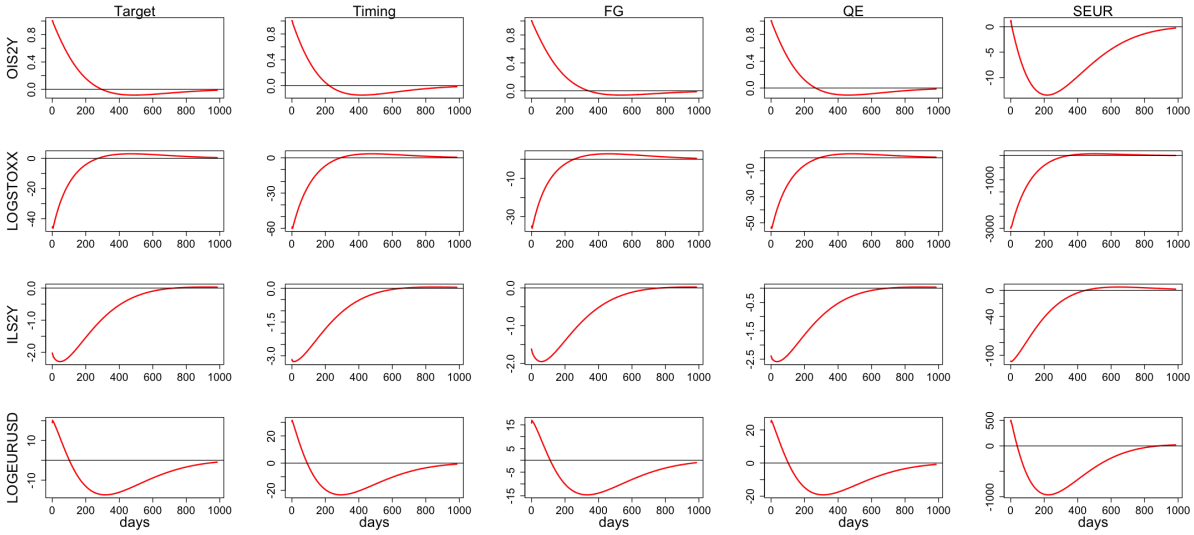
```
[1] "start"      "2005-03-31" "end"        "2008-01-03"
[1] "start"      "2008-01-03" "end"        "2014-01-01"
```

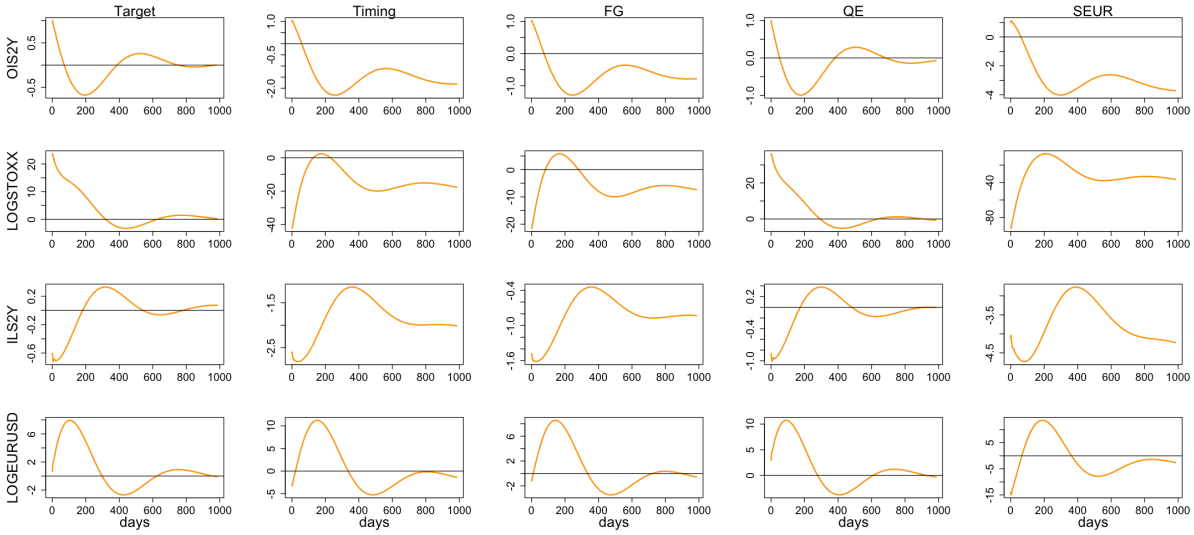



```
[1] "start"      "2014-01-03" "end"        "2018-09-13"
```




```
[1] "start"      "2018-09-13" "end"        "2022-12-30"
```




Target
Timing
FG
QE
SEUR
OIS2Y
LOGSTOXX
ILS2Y
LOGEURUSD
days

In [ ]: