Macro ML

Jean-Galaad BARRIERE

05/11/2022

Introduction

In financial econometrics, numerous approaches have been developed to explain the returns of assets. It is often assumed that the excess returns are related to a given set of factors. The exposition of an asset to a factor must be compensated by a "risk premium' '. Therefore, the excess return of an asset depends on those risk premia multiplied by the exposition of the asset to each of the factors.

A key issue of financial factor models resides in the choice of the factors. Various models have been developed, using different sets of factors. For instance, the Fama-French three-factor model is based on market excess return, outperformance of small versus big companies and outperformance of high book-to-market versus low book-to-market companies.

Our article investigates how macro factors can be used in asset pricing models. As already shown in the literature, some macroeconomic variables (such as GDP growth, inflation, unemployment or housing prices) could generate risk premia. Nonetheless, the difficulty lies in the identification of the relevant macroeconomic variables among a very large set of macroeconomic indicators. Some previous papers have arbitrarily chosen one or two macroeconomic variables. Our article innovates by using machine learning techniques so as to construct a few factors out of a large set of macroeconomic variables. The central ML technique used here is **sparse Principal Component Analysis** (PCA). As we will see below, the main advantage of sparse PCA over PCA lies in the interpretability of the factors.

[reste de l'intro]

PCA and Sparse PCA

Our article performs a sparse PCA on a set of 120 macroeconomic variables from the FRED-MD database. Those variables cover various categories: output and income, labor market, housing, consumption, money and credit, interest and exchanges rates, and prices. Here are some examples of macroeconomic variables: real personal income, industrial production indices, civilian unemployment, number of employees by sector, number of housing permits, M1 money stock, commercial and industrial loans, fed fund rates, consumer price indices.

Before performing the sparse PCA, we need some treatment on the FRED-MD data. We use a csv file on which we reported metadata on the FRED-MD macroeconomic variables, in particular : whether they should be included in the analysis and what transformation should be performed on them (log, log growth, difference). These indications come from ***Table 1*** of the article. After selecting the relevant variables and performing the transformations, we restrict the dataset to the time period considered (1960:01 to 2019:12)

```
library(dplyr)

file <- "data/2020-11.csv"
data0 <- read.csv(file = file)

x <- data0$sasdate
```

```r
# we drop the rows which have no date
data1 <- data0[(x!="Transform:" & nchar(x)>2),]
y<-data1[,1]

# extraction of variable names
varnames <- data.frame("FRED_ticker"=colnames(data1)[-1])
write.csv(varnames, "varnames.csv", row.names = F)

##### Keeping only relevant time series
# Importation of csv file with variables metadata
df <- read.csv("data/variables.csv",sep=";")
df <- filter(df,Inclusion==1)
var <- df$FRED_ticker

#on garde la date
var <- c("sasdate", var)

data <- data1[var]

### Transformation of the time series
var_names <- colnames(data)
for(i in 2:length(var_names)){ # exclusion of 1st column (date)
  variable <- var_names[[i]]
  transfo <- df$Transformation[df$FRED_ticker==variable]
  if(!is.null(transfo)){
    if(transfo=="Log"){
      data[,i]<-log(data[,i])
    }
    if(transfo=="Difference"){
      data[,i]<-c(NA, diff(data[,i])) # length is decreased by 1 when we take the difference
    }
    if(transfo=="Log growth"){
      tmp <- data[,i]
      tmp <- tmp/lag(tmp)
      tmp<-log(tmp)
      data[,i]<-c(tmp) # length is decreased by 1 when we take the difference
    }
  }
}
```

```
## Warning in log(tmp): Production de NaN
```

```r
## Time interval
data$sasdate<-as.Date(data$sasdate, format = "%m/%d/%Y") # conversion to date
data <- filter(data, sasdate>="1960-02-01" & sasdate<"2020-01-01")

### Saving to RDS
saveRDS(data, "data/FRED_data.rds")
```

PCA

We first perform of traditional PCA on the 120 variables, and select 9 components. We use the same package as the authors

```r
library(FactoMineR)
library(knitr)

data <- readRDS("data/FRED_data.rds")
data0 <- dplyr::select(data, -1) # we drop the date column
sum(is.na(data0))
```

```
## [1] 2
```

```r
pca <- PCA(data0, ncp=9, graph=F)
```

```
## Warning in PCA(data0, ncp = 9, graph = F): Missing values are imputed by the
## mean of the variable: you should use the imputePCA function of the missMDA
## package
```

```r
table1 <- pca$eig
```

```r
kable(table1[1:9,], caption = "First 9 components of the PCA")
```

Table 1: First 9 components of the PCA

	eigenvalue	percentage of variance	cumulative percentage of variance
comp 1	20.741160	17.284300	17.28430
comp 2	17.368041	14.473368	31.75767
comp 3	7.985066	6.654221	38.41189
comp 4	6.449405	5.374504	43.78639
comp 5	4.896562	4.080468	47.86686
comp 6	3.668845	3.057370	50.92423
comp 7	3.122703	2.602253	53.52648
comp 8	2.785973	2.321644	55.84813
comp 9	2.704916	2.254097	58.10222

The first nine conventional PCs collectively explain 58.1022246% of the total variation in the macroeconomic variables.

The outcome of our PCA is somewhat different from the results presented in the article. Indeed, the weights of the components are different. This can be explained by modifications of the FRED-MD data between the redaction of the paper on our replication. We noticed that some variables do not have exactly the same name in our version of the FRED data and in the original article. Despite these differences, we are reassured by the fact that in the original article, the first nine PCs collectively explain 57% of the total variation.

We plot the principal components that we extracted from the 120 FRED-MD macroeconomic variables, as the authors do in **Figure 1** of their article.

```r
pca_ts <- ts(data=pca$ind$coord, start = c(1960,1), frequency=12)
par(mfrow = c(3, 3), mar = c(5.1, 4.1, 4.1, 2.1))
for(i in 1:9){
  plot(pca_ts[,i],
       main = paste0("PC",i),
       ylab="")
}
```

Sparse PCA

We now perform a sparse PCA, using the same R package as the authors. Before running the `SPC` function, we scale the variables (so that they have a unit variance). In the article, the authors set the shrinkage parameter

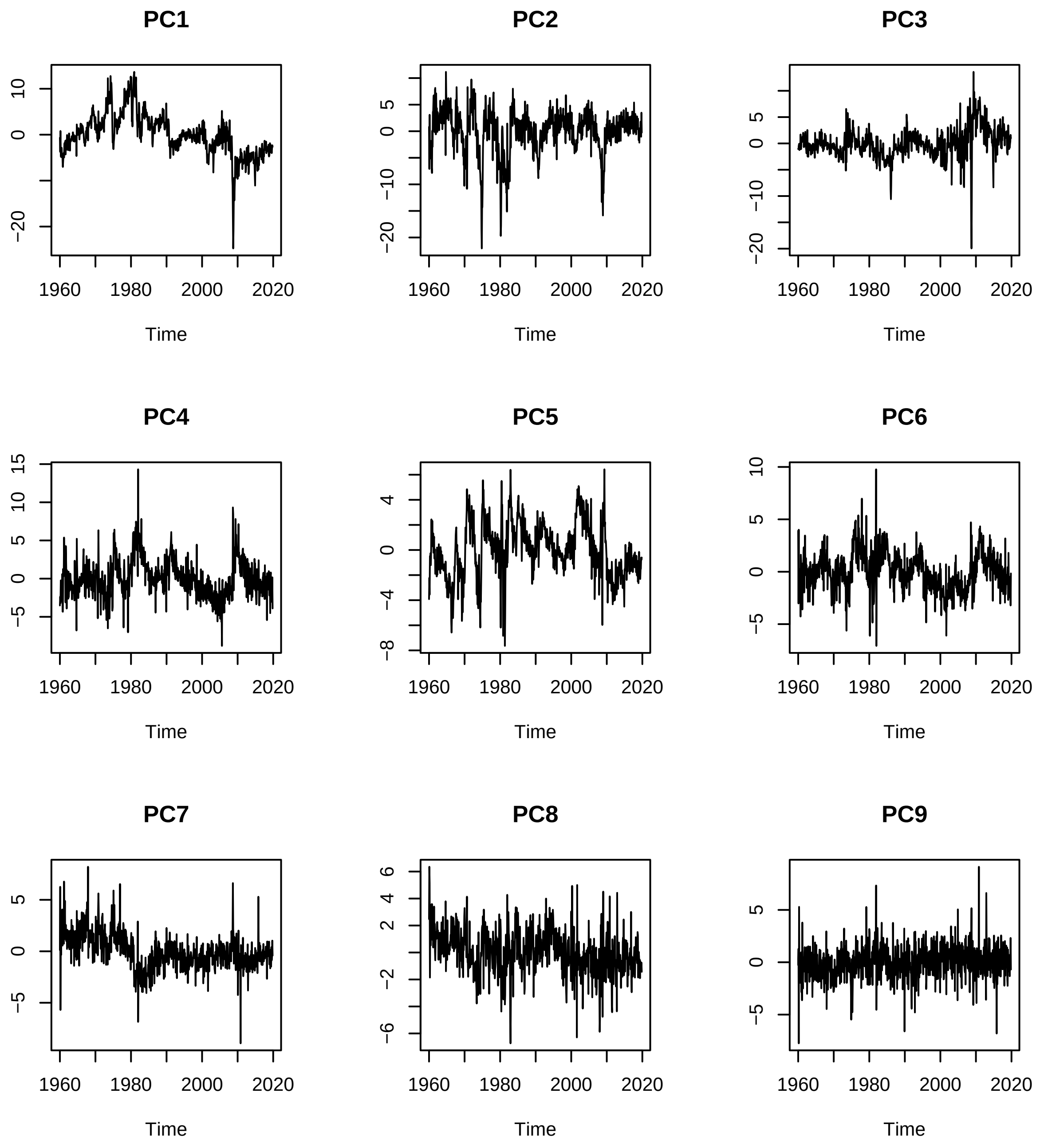


Figure 1: Conventional principal components

so that only 108 weights are active. The set the parameter `sumabsv` to 3 to get a similar outcome.

```
library(PMA)
data0<-as.matrix(data0)
data0<-scale(data0) # we scale variables
spca <- SPC(data0,sumabsv = 3, K=9)
```

```
## 12345678910111213
## 1234567891011121314151617181920
## 12345678910111213141516
## 1234567891011121314151617181920
## 1234567891011121314151617
## 1234567891011121314151617181920
## 1234567891011121314151617181920
## 1234567891011121314151617181920
## 1234567891011121314151617181920
```

```
weights <- spca$v
row.names(weights)<- colnames(data0)
sum(weights!=0)
```

```
## [1] 107
```

```
# Percentage of variance
components <- paste0("comp ", 1:9)
table2 <- data.frame(Component = components,
                     Cumulative_percentage_of_variance = spca$prop.var.explained)
kable(table2, caption = "First 9 components of the SPCA")
```

Table 2: First 9 components of the SPCA

Component	Cumulative_percentage_of_variance
comp 1	0.0708785
comp 2	0.1325496
comp 3	0.1970411
comp 4	0.2583984
comp 5	0.3148364
comp 6	0.3680362
comp 7	0.4042698
comp 8	0.4335150
comp 9	0.4640008

```
#### Identification of active weights
component_names <- c("Yields","Production", "Inflation", "Housing", "Spreads", "Employment", "Costs", "
active_weights<-rep("", 9)
for(i in 1:9){
  active_weights[i] <- paste0(row.names(weights)[weights[,i]!=0], collapse = " ; ")
}
active_weights_df <- data.frame(Sparse_Component = 1:9,
                                Component_name = component_names,
                                Active_weights = active_weights)
kable(active_weights_df)
```

Sparse_Component	Component_name	Active_weights
1	Yields	S.P.div.yield ; FEDFUNDS ; CP3Mx ; TB3MS ; TB6MS ; GS1 ; GS5 ; GS10 ; AAA ; BAA
2	Production	INDPRO ; IPFPNSS ; IPFINAL ; IPCONGD ; IPDCONGD ; IPBUSEQ ; IPMAT ; IPDMAT ; IPMANSICS ; CUMFNS ; MANEMP ; DMANEMP
3	Inflation	WPSFD49207 ; WPSFD49502 ; WPSID61 ; CPIAUCSL ; CPITRNSL ; CUSR0000SAC ; CPIULFSL ; CUSR0000SA0L2 ; CUSR0000SA0L5 ; PCEPI ; DNDGRG3M086SBEA
4	Housing	HOUST ; HOUSTNE ; HOUSTMW ; HOUSTS ; HOUSTW ; PERMIT ; PERMITNE ; PERMITMW ; PERMITS ; PERMITW ; REALLN
5	Spreads	COMPAPFFx ; TB3SMFFM ; TB6SMFFM ; T1YFFM ; T5YFFM ; T10YFFM ; AAAFFM ; BAAFFM ; CPIMEDSL ; CUSR0000SAS ; DSERRG3M086SBEA
6	Employment	PAYEMS ; USGOOD ; USCONS ; MANEMP ; DMANEMP ; NDMANEMP ; SRVPRD ; USTPU ; USWTRADE ; USTRADE ; USFIRE
7	Costs	USFIRE ; BUSINVx ; M2REAL ; S.P.div.yield ; CPIAPPSL ; CPIMEDSL ; CUSR0000SAD ; CUSR0000SAS ; DDURRG3M086SBEA ; DSERRG3M086SBEA ; CES0600000008 ; CES2000000008 ; CES3000000008
8	Money	BUSINVx ; M1SL ; M2SL ; M2REAL ; BOGMBASE ; TOTRESNS ; WPSFD49207 ; WPSFD49502 ; WPSID61 ; WPSID62 ; OILPRICEx ; PPICMM ; MZMSL
9	SPC9	DPCERA3M086SBEA ; CMRMTSPLx ; RETAILx ; IPNCONGD ; IPNMAT ; HWIURATIO ; CE16OV ; UNRATE ; CLAIMSx ; USCONS ; CES0600000007 ; AWOTMAN ; AWHMAN ; AMDMNOx ; ISRATIOx

The result of our sparse PCA is quite satisfactory, insofar as they are very similar to those represented in the article. As in the article, the nine components of the PCA explain 46% of the total variation in the 120 macroeconomic variables. By looking at the active weights of each component, we see that they do not exactly match those presented in ***Table 3*** of the article. We can nevertheless give them the same interpretation as in the article, except for the ninth component. The active weights of the ninth component diverge too much from those of the original article. In our results, it is difficult to interpret this component as an index for credit ; we therefore keep the name "SPC 9".

```r
spca_ts <- ts(data=spca$u, start = c(1960,1), frequency=12)
par(mfrow = c(3, 3), mar = c(5.1, 4.1, 4.1, 2.1))
for(i in 1:9){
  plot(spca_ts[,i],
       main = component_names[i],
       ylab="")
}
```

Innovation correlations

Autorégression sur la PCA

`pca$ind$coord` contient les coordonnées pour chaque observation dans l'espace des 9 PC.

```r
library(vars)
```

```
## Warning: le package 'vars' a été compilé avec la version R 4.2.2

## Le chargement a nécessité le package : MASS

##
## Attachement du package : 'MASS'

## L'objet suivant est masqué depuis 'package:dplyr':
##
##     select
```

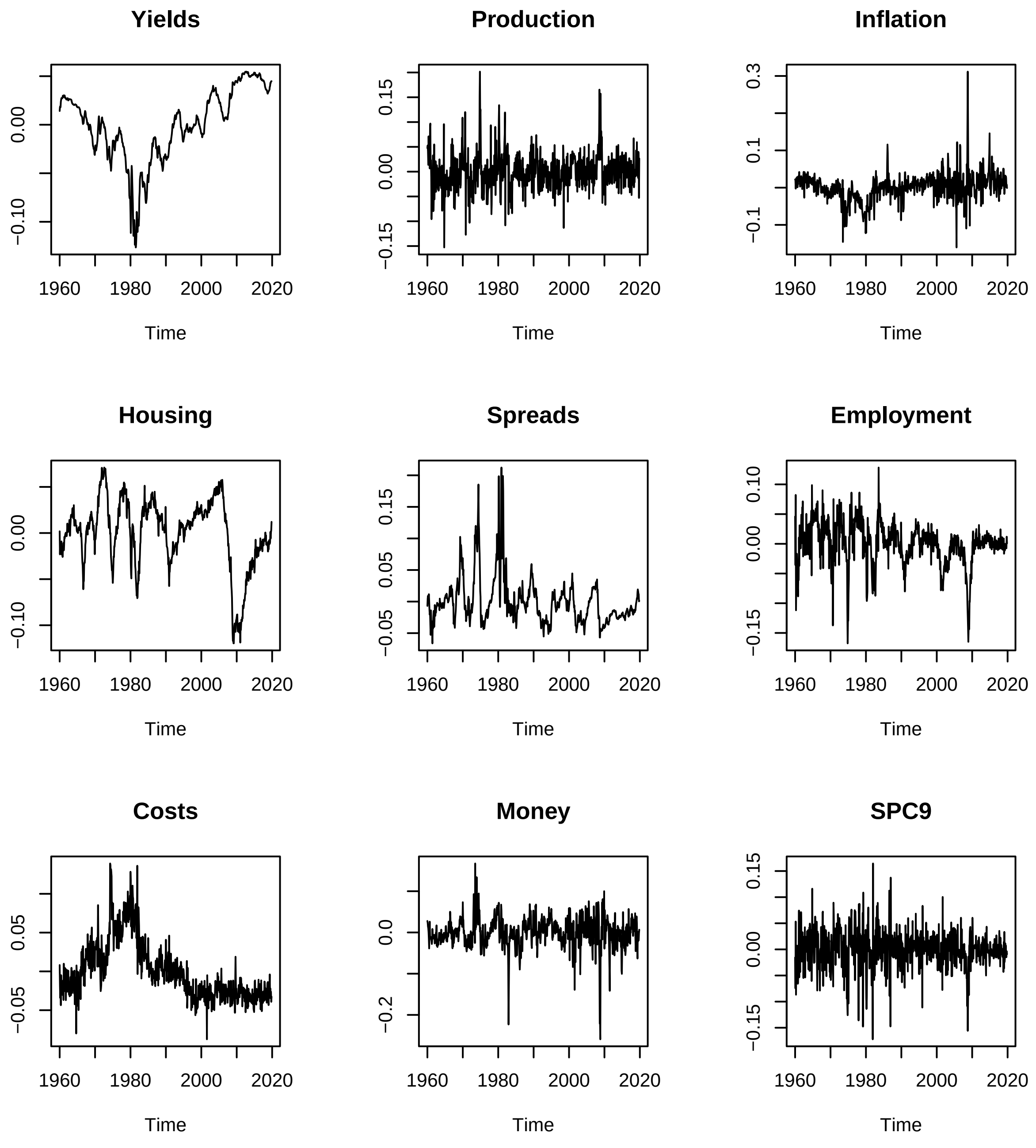


Figure 2: Sparse principal components

```
## Le chargement a nécessité le package : strucchange
```

```
## Warning: le package 'strucchange' a été compilé avec la version R 4.2.2
```

```
## Le chargement a nécessité le package : zoo
```

```
##
## Attachement du package : 'zoo'
```

```
## Les objets suivants sont masqués depuis 'package:base':
##
##     as.Date, as.Date.numeric
```

```
## Le chargement a nécessité le package : sandwich
```

```
## Le chargement a nécessité le package : urca
```

```
## Le chargement a nécessité le package : lmtest
```

```
View(pca$ind$coord)
data_pca <- pca$ind$coord
row.names(data_pca) <- data$date
ar_pca <- VAR(data_pca, p=1)
#summary(ar_pca)
```

La matrice de corrélation des résidus ressemble beaucoup à celle de l'article (table 4).

Autorégression sur la Sparse PCA

spca$u contient les coordonnées pour chaque observation dans l'espace des 9 SPC.

```
View(spca$u)
data_spca <- spca$u
row.names(data_spca) <- data$date
ar_spca <- VAR(data_spca, p=1)
```

```
## Warning in VAR(data_spca, p = 1): No column names supplied in y, using: y1, y2, y3, y4, y5, y6, y7,
```

```
summary(ar_spca)
```

```
##
## VAR Estimation Results:
## =========================
## Endogenous variables: y1, y2, y3, y4, y5, y6, y7, y8, y9
## Deterministic variables: const
## Sample size: 718
## Log Likelihood: 17482.878
## Roots of the characteristic polynomial:
## 0.9811 0.9811 0.8827 0.6967 0.6967 0.5048 0.2926 0.2108 0.01959
## Call:
## VAR(y = data_spca, p = 1)
##
##
## Estimation results for equation y1:
## ===================================
## y1 = y1.l1 + y2.l1 + y3.l1 + y4.l1 + y5.l1 + y6.l1 + y7.l1 + y8.l1 + y9.l1 + const
##
##         Estimate Std. Error t value Pr(>|t|)
## y1.l1  9.756e-01  5.735e-03 170.101  < 2e-16 ***
## y2.l1 -4.595e-04  5.199e-03  -0.088 0.929597
## y3.l1  1.520e-02  5.763e-03   2.637 0.008549 **
## y4.l1 -1.082e-02  4.126e-03  -2.623 0.008898 **
## y5.l1  6.057e-04  4.705e-03   0.129 0.897593
## y6.l1 -1.627e-02  4.854e-03  -3.352 0.000845 ***
## y7.l1 -1.689e-02  5.749e-03  -2.938 0.003409 **
## y8.l1  8.303e-03  4.867e-03   1.706 0.088426 .
## y9.l1 -1.075e-02  4.578e-03  -2.348 0.019154 *
## const  4.199e-05  1.344e-04   0.312 0.754871
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##
## Residual standard error: 0.003602 on 708 degrees of freedom
## Multiple R-Squared: 0.9908,  Adjusted R-squared: 0.9907
## F-statistic:  8483 on 9 and 708 DF,  p-value: < 2.2e-16
##
##
## Estimation results for equation y2:
## ===================================
## y2 = y1.l1 + y2.l1 + y3.l1 + y4.l1 + y5.l1 + y6.l1 + y7.l1 + y8.l1 + y9.l1 + const
##
##         Estimate Std. Error t value Pr(>|t|)
## y1.l1  0.0512819  0.0529635   0.968  0.33325
## y2.l1  0.0274812  0.0480072   0.572  0.56721
## y3.l1  0.0772579  0.0532159   1.452  0.14701
## y4.l1 -0.1034457  0.0381008  -2.715  0.00679 **
## y5.l1  0.2776104  0.0434482   6.389 3.02e-10 ***
## y6.l1 -0.3167042  0.0448301  -7.065 3.86e-12 ***
## y7.l1  0.0335897  0.0530925   0.633  0.52716
## y8.l1 -0.0297738  0.0449437  -0.662  0.50789
## y9.l1  0.0024004  0.0422806   0.057  0.95474
## const -0.0000654  0.0012414  -0.053  0.95800
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##
## Residual standard error: 0.03326 on 708 degrees of freedom
## Multiple R-Squared: 0.2148,  Adjusted R-squared: 0.2048
## F-statistic: 21.52 on 9 and 708 DF,  p-value: < 2.2e-16
##
##
## Estimation results for equation y3:
## ===================================
## y3 = y1.l1 + y2.l1 + y3.l1 + y4.l1 + y5.l1 + y6.l1 + y7.l1 + y8.l1 + y9.l1 + const
##
##         Estimate Std. Error t value Pr(>|t|)
## y1.l1 -7.141e-03  4.534e-02  -0.158 0.874896
## y2.l1 -6.086e-02  4.110e-02  -1.481 0.139124
## y3.l1  2.874e-01  4.556e-02   6.309 4.94e-10 ***
## y4.l1 -3.205e-02  3.262e-02  -0.982 0.326225
## y5.l1 -1.651e-01  3.720e-02  -4.439 1.05e-05 ***
## y6.l1 -9.155e-03  3.838e-02  -0.239 0.811541
## y7.l1 -2.385e-01  4.545e-02  -5.247 2.04e-07 ***
## y8.l1 -1.437e-01  3.848e-02  -3.735 0.000203 ***
## y9.l1 -8.993e-02  3.620e-02  -2.485 0.013201 *
## const -2.397e-05  1.063e-03  -0.023 0.982015
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##
## Residual standard error: 0.02848 on 708 degrees of freedom
## Multiple R-Squared: 0.4255,  Adjusted R-squared: 0.4182
## F-statistic: 58.27 on 9 and 708 DF,  p-value: < 2.2e-16
##
##
## Estimation results for equation y4:
## ===================================
## y4 = y1.l1 + y2.l1 + y3.l1 + y4.l1 + y5.l1 + y6.l1 + y7.l1 + y8.l1 + y9.l1 + const
##
##         Estimate Std. Error t value Pr(>|t|)
## y1.l1 -6.124e-04  1.130e-02  -0.054    0.957
## y2.l1  6.347e-03  1.024e-02   0.620    0.535
## y3.l1 -1.683e-02  1.135e-02  -1.483    0.139
## y4.l1  9.876e-01  8.126e-03 121.541  < 2e-16 ***
## y5.l1 -3.708e-02  9.266e-03  -4.001 6.97e-05 ***
## y6.l1 -1.149e-02  9.561e-03  -1.202    0.230
## y7.l1  4.922e-03  1.132e-02   0.435    0.664
## y8.l1 -1.104e-02  9.585e-03  -1.151    0.250
## y9.l1  9.199e-03  9.017e-03   1.020    0.308
## const  1.377e-05  2.648e-04   0.052    0.959
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##
## Residual standard error: 0.007094 on 708 degrees of freedom
## Multiple R-Squared: 0.9644,  Adjusted R-squared: 0.9639
## F-statistic:  2129 on 9 and 708 DF,  p-value: < 2.2e-16
##
##
## Estimation results for equation y5:
## ===================================
## y5 = y1.l1 + y2.l1 + y3.l1 + y4.l1 + y5.l1 + y6.l1 + y7.l1 + y8.l1 + y9.l1 + const
##
##         Estimate Std. Error t value Pr(>|t|)
## y1.l1 -5.246e-02  2.173e-02  -2.414 0.016020 *
## y2.l1 -5.367e-02  1.969e-02  -2.725 0.006585 **
## y3.l1  1.606e-02  2.183e-02   0.736 0.462144
## y4.l1  1.155e-02  1.563e-02   0.739 0.460152
## y5.l1  8.880e-01  1.782e-02  49.818  < 2e-16 ***
## y6.l1  5.074e-02  1.839e-02   2.759 0.005947 **
## y7.l1  7.936e-03  2.178e-02   0.364 0.715692
## y8.l1  3.371e-02  1.844e-02   1.828 0.067959 .
## y9.l1 -6.253e-02  1.735e-02  -3.605 0.000334 ***
## const  6.397e-06  5.093e-04   0.013 0.989982
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##
```

```
## Residual standard error: 0.01365 on 708 degrees of freedom
## Multiple R-Squared: 0.8681,  Adjusted R-squared: 0.8665
## F-statistic: 517.9 on 9 and 708 DF,  p-value: < 2.2e-16
##
##
## Estimation results for equation y6:
## ===================================
## y6 = y1.l1 + y2.l1 + y3.l1 + y4.l1 + y5.l1 + y6.l1 + y7.l1 + y8.l1 + y9.l1 + const
##
##         Estimate Std. Error t value Pr(>|t|)
## y1.l1  5.317e-02  4.166e-02   1.276  0.20226
## y2.l1 -1.131e-01  3.776e-02  -2.994  0.00285 **
## y3.l1  2.670e-02  4.186e-02   0.638  0.52372
## y4.l1  1.506e-01  2.997e-02   5.025  6.4e-07 ***
## y5.l1 -9.649e-02  3.418e-02  -2.823  0.00489 **
## y6.l1  5.767e-01  3.526e-02  16.353  < 2e-16 ***
## y7.l1  1.057e-01  4.176e-02   2.532  0.01156 *
## y8.l1  5.503e-02  3.535e-02   1.557  0.12003
## y9.l1 -6.119e-02  3.326e-02  -1.840  0.06620 .
## const -6.546e-05  9.765e-04  -0.067  0.94657
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##
## Residual standard error: 0.02617 on 708 degrees of freedom
## Multiple R-Squared: 0.5142,  Adjusted R-squared: 0.508
## F-statistic: 83.27 on 9 and 708 DF,  p-value: < 2.2e-16
##
##
## Estimation results for equation y7:
## ===================================
## y7 = y1.l1 + y2.l1 + y3.l1 + y4.l1 + y5.l1 + y6.l1 + y7.l1 + y8.l1 + y9.l1 + const
##
##         Estimate Std. Error t value Pr(>|t|)
## y1.l1 -1.676e-01  2.941e-02  -5.700 1.76e-08 ***
## y2.l1  8.488e-02  2.666e-02   3.184  0.00151 **
## y3.l1 -9.094e-02  2.955e-02  -3.078  0.00217 **
## y4.l1 -2.328e-02  2.115e-02  -1.101  0.27141
## y5.l1  3.289e-02  2.412e-02   1.363  0.17321
## y6.l1  5.195e-02  2.489e-02   2.087  0.03724 *
## y7.l1  6.763e-01  2.948e-02  22.943  < 2e-16 ***
## y8.l1 -2.863e-02  2.495e-02  -1.147  0.25161
## y9.l1  6.813e-02  2.348e-02   2.902  0.00382 **
## const -5.084e-05  6.893e-04  -0.074  0.94122
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##
## Residual standard error: 0.01847 on 708 degrees of freedom
## Multiple R-Squared: 0.7585,  Adjusted R-squared: 0.7554
## F-statistic:   247 on 9 and 708 DF,  p-value: < 2.2e-16
##
##
## Estimation results for equation y8:
```

```
## ===================================
## y8 = y1.l1 + y2.l1 + y3.l1 + y4.l1 + y5.l1 + y6.l1 + y7.l1 + y8.l1 + y9.l1 + const
##
##         Estimate Std. Error t value Pr(>|t|)
## y1.l1  5.460e-02  4.918e-02   1.110  0.26732
## y2.l1 -8.621e-02  4.458e-02  -1.934  0.05354 .
## y3.l1 -6.484e-02  4.942e-02  -1.312  0.18992
## y4.l1  2.151e-02  3.538e-02   0.608  0.54348
## y5.l1  1.067e-01  4.035e-02   2.645  0.00834 **
## y6.l1  2.750e-02  4.163e-02   0.661  0.50908
## y7.l1  4.385e-04  4.930e-02   0.009  0.99291
## y8.l1  4.842e-01  4.174e-02  11.601  < 2e-16 ***
## y9.l1 -6.013e-02  3.926e-02  -1.531  0.12611
## const -3.272e-05  1.153e-03  -0.028  0.97737
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##
## Residual standard error: 0.03089 on 708 degrees of freedom
## Multiple R-Squared: 0.3239,  Adjusted R-squared: 0.3153
## F-statistic: 37.69 on 9 and 708 DF,  p-value: < 2.2e-16
##
##
## Estimation results for equation y9:
## ===================================
## y9 = y1.l1 + y2.l1 + y3.l1 + y4.l1 + y5.l1 + y6.l1 + y7.l1 + y8.l1 + y9.l1 + const
##
##         Estimate Std. Error t value Pr(>|t|)
## y1.l1  3.558e-02  5.598e-02   0.636 0.525290
## y2.l1 -2.023e-01  5.075e-02  -3.986 7.42e-05 ***
## y3.l1 -1.094e-01  5.625e-02  -1.946 0.052103 .
## y4.l1  1.338e-01  4.027e-02   3.321 0.000943 ***
## y5.l1 -2.443e-01  4.593e-02  -5.319 1.40e-07 ***
## y6.l1  8.079e-02  4.739e-02   1.705 0.088637 .
## y7.l1  5.385e-02  5.612e-02   0.960 0.337578
## y8.l1 -5.610e-02  4.751e-02  -1.181 0.238009
## y9.l1 -2.623e-01  4.469e-02  -5.869 6.74e-09 ***
## const  1.886e-05  1.312e-03   0.014 0.988537
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##
## Residual standard error: 0.03516 on 708 degrees of freedom
## Multiple R-Squared: 0.1244,  Adjusted R-squared: 0.1133
## F-statistic: 11.18 on 9 and 708 DF,  p-value: < 2.2e-16
##
##
##
## Covariance matrix of residuals:
##            y1         y2         y3         y4         y5         y6         y7
## y1  1.298e-05  1.733e-05  1.278e-05  1.993e-06 -5.481e-06 -9.337e-06 -5.371e-06
## y2  1.733e-05  1.107e-03  2.377e-05 -3.996e-05 -1.579e-05 -4.303e-04 -6.186e-05
## y3  1.278e-05  2.377e-05  8.110e-04 -1.385e-05 -1.613e-05 -6.348e-05 -1.259e-04
## y4  1.993e-06 -3.996e-05 -1.385e-05  5.033e-05 -4.255e-06  3.595e-05  5.562e-07
## y5 -5.481e-06 -1.579e-05 -1.613e-05 -4.255e-06  1.862e-04  1.305e-05 -3.132e-06
## y6 -9.337e-06 -4.303e-04 -6.348e-05  3.595e-05  1.305e-05  6.847e-04  3.699e-05
## y7 -5.371e-06 -6.186e-05 -1.259e-04  5.562e-07 -3.132e-06  3.699e-05  3.411e-04
## y8 -2.185e-05 -2.200e-05 -5.079e-04 -1.032e-05  4.682e-05  5.387e-05  4.014e-05
## y9 -1.326e-05 -6.835e-04 -1.351e-04  9.399e-05  5.816e-06  3.850e-04 -8.130e-07
##            y8         y9
## y1 -2.185e-05 -1.326e-05
## y2 -2.200e-05 -6.835e-04
## y3 -5.079e-04 -1.351e-04
## y4 -1.032e-05  9.399e-05
## y5  4.682e-05  5.816e-06
## y6  5.387e-05  3.850e-04
## y7  4.014e-05 -8.130e-07
## y8  9.542e-04  8.732e-05
## y9  8.732e-05  1.236e-03
##
## Correlation matrix of residuals:
##          y1       y2       y3        y4       y5       y6        y7       y8
## y1  1.00000  0.14461  0.12459  0.077988 -0.11150 -0.09906 -0.080731 -0.19634
## y2  0.14461  1.00000  0.02510 -0.169342 -0.03479 -0.49437 -0.100689 -0.02141
## y3  0.12459  0.02510  1.00000 -0.068540 -0.04150 -0.08518 -0.239412 -0.57733
## y4  0.07799 -0.16934 -0.06854  1.000000 -0.04395  0.19364  0.004245 -0.04707
## y5 -0.11150 -0.03479 -0.04150 -0.043948  1.00000  0.03656 -0.012426  0.11108
## y6 -0.09906 -0.49437 -0.08518  0.193640  0.03656  1.00000  0.076541  0.06664
## y7 -0.08073 -0.10069 -0.23941  0.004245 -0.01243  0.07654  1.000000  0.07035
## y8 -0.19634 -0.02141 -0.57733 -0.047068  0.11108  0.06664  0.070354  1.00000
## y9 -0.10466 -0.58440 -0.13493  0.376772  0.01212  0.41849 -0.001252  0.08039
##           y9
## y1 -0.104655
## y2 -0.584396
## y3 -0.134931
## y4  0.376772
## y5  0.012120
## y6  0.418486
## y7 -0.001252
## y8  0.080390
## y9  1.000000
```

Encore une fois, on n'est pas trop loin des résultats de l'article! cf table 4

Risk premia estimates

We now turn to the estimation of the risk premia of the sparse macro factors. The objective is to determine whether some of the macro factors generate some significant risk premia.

We import the data on portfolio returns and keep the same time period as the authors (1963:07 to 2019:12).

```r
R <- readRDS("data/portfolios.rds")
R <- filter(R, date<='2019-12-01')
dates <- R$date
R<-dplyr::select(R,-1)
```

We need to compute the excess returns of each portfolios. This requires data on the risk-free rate at every period in time. The authors use the CRSP risk-free return. However, as these data are not freely available, we replace the risk-free rate by TB3MS variable from FREDMD (3-Month Treasury Bill Secondary Market Rate, Discount Basis).

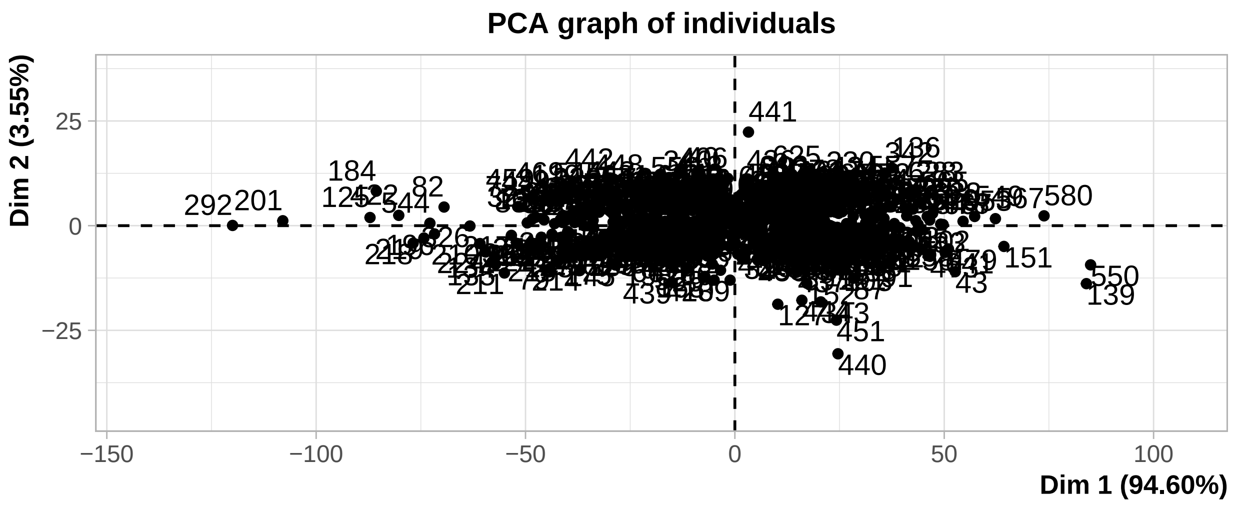
```r
data_rf <- read.csv(file = "data/TB3MS.csv")
data_rf <- dplyr::select(data_rf, -1) # we remove the date
for (i in 1:ncol(R)){
  R[,i] <- as.numeric(R[,i]) - data_rf[,1]
}
```

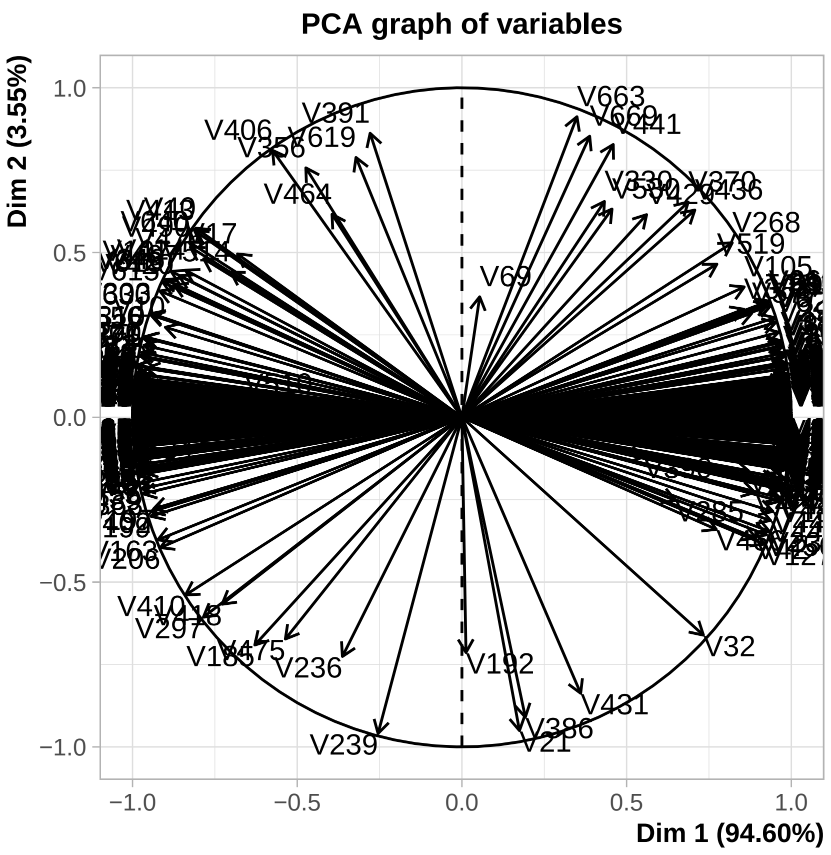
We demean the excess returns of each portfolio

```r
R_d <- R-t(as.matrix(colMeans(R))) # the result is != 0 due to approx errors
```

We run a PCA of the excess returns of our portfolios, to estimated the rotated fundamental factors (denoted `ksi`)

```r
t <- nrow(R_d)
n <- ncol(R_d)
R_d <- t(as.matrix(R_d))
mat <- (t(R_d) %*% R_d)/(t*n)
r_pca <- PCA(mat, ncp=15)
```





```
ksi <- t(r_pca$var$coord) #eigenvectors
V <- sqrt(t)*t(r_pca$var$coord)

# estimator of beta (exposure to factors)
beta <- (1/t)*R_d%*%t(V)

r_mean <- colMeans(R) #average return
gamma <- solve(t(beta)%*%beta) %*% t(beta) %*% as.matrix(r_mean) #OLS


# alternative : with OLS
lm1 <- lm(r_mean~-1+beta)
summary(lm1)
```

```
##
## Call:
## lm(formula = r_mean ~ -1 + beta)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.62699 -0.07578  0.00926  0.08367  0.36106
##
## Coefficients:
##               Estimate Std. Error  t value Pr(>|t|)
## betaDim.1   -0.0285316  0.0001884 -151.410  < 2e-16 ***
## betaDim.2   -0.1425057  0.0029320  -48.604  < 2e-16 ***
## betaDim.3  -0.2647945  0.0115596  -22.907  < 2e-16 ***
## betaDim.4   0.0522575  0.0230164    2.270  0.02382 *
## betaDim.5   0.4417834  0.0281623   15.687  < 2e-16 ***
## betaDim.6  -0.2261978  0.0369536   -6.121 2.64e-09 ***
## betaDim.7   0.5512277  0.0657879    8.379 1.56e-15 ***
## betaDim.8  -1.0062995  0.0770880  -13.054  < 2e-16 ***
## betaDim.9   0.0949227  0.0760880    1.248  0.21308
## betaDim.10  0.9460650  0.1055167    8.966  < 2e-16 ***
## betaDim.11  1.7915206  0.1258126   14.240  < 2e-16 ***
## betaDim.12  0.1629277  0.1537956    1.059  0.29020
## betaDim.13  0.9970662  0.2213628    4.504 9.26e-06 ***
## betaDim.14  0.6403207  0.2170883    2.950  0.00341 **
## betaDim.15 -0.3667756  0.2448742   -1.498  0.13514
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.1397 on 330 degrees of freedom
## Multiple R-squared:  0.9985, Adjusted R-squared:  0.9985
## F-statistic: 1.501e+04 on 15 and 330 DF,  p-value: < 2.2e-16
```

```
# R² proche de l'article avec intercept, mais erronné sans intercept (calcul du R² dans un modèle sans
```

The last step is to run a time-series regression of the observed factors on the rotated fundamental factors.

```
# we restrict the observed factors to the good time period
dates_pca <- data$sasdate
indices_dates <- dates_pca>="1963-07-01" & dates_pca<= "2019-12-01"

# residuals of the VAR(1)
res <- residuals(ar_pca)
G <- res[indices_dates[-1],] # we drop the first element of res (ar(1) has one obs less)
G <- t(G)

eta <- G %*% t(V) %*% solve(V %*% t(V))


gamma_g <- eta %*% gamma

# with tslm
library(forecast)
```

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

```
G_ts <- ts(t(G))
ksi_ts <- ts(t(ksi))
lm3 <- tslm(G_ts~0+ksi_ts)
coefficients(lm3)
```

```
##                     Dim.1       Dim.2       Dim.3        Dim.4       Dim.5
## ksi_tsDim.1   -0.02622195  0.03898848  0.05522340 -0.113762508  0.03924571
## ksi_tsDim.2   -0.53307603 -0.49959300 -0.84739424 -0.004976292 -0.43866476
## ksi_tsDim.3   -0.85562007  1.42615325 -1.09671012  1.439989042 -0.08502611
## ksi_tsDim.4    1.24912210 -1.58537089  2.89827844 -1.469363224  1.68190104
## ksi_tsDim.5   -4.19807263  0.94992689 -5.44014187  0.776596086 -2.42713369
## ksi_tsDim.6   -0.42025362  -0.30655328  -0.03643104   1.714589634 -1.10024993
## ksi_tsDim.7   -1.32291583   0.42995463  -2.05502811  -0.316390831 -1.82346996
## ksi_tsDim.8    2.72747331   6.49418380   3.34384800   3.307999165  0.74001086
## ksi_tsDim.9    4.84457259   6.62482922   3.56960726   3.226987730 -0.07154799
## ksi_tsDim.10   0.12223593  -2.82679386   5.63092764  -4.801099718 -0.30994590
## ksi_tsDim.11   0.54538087 -10.40258396   0.27627369  -4.811645600  1.69842833
## ksi_tsDim.12   7.97929835  13.47782740   3.96604345   5.706298544 -3.94058504
## ksi_tsDim.13 -13.57450644 -21.92335720 -12.22998552 -14.867184604  0.41915430
## ksi_tsDim.14  16.42161527   6.69578881  28.96255564   4.441849514 12.31936103
## ksi_tsDim.15   0.79994326   3.28954501   3.61654467   3.223294396  1.85701114
##                    Dim.6      Dim.7       Dim.8       Dim.9
## ksi_tsDim.1    0.07160832 -0.0558630 -0.04361391 -0.14503325
## ksi_tsDim.2   -0.31246330  0.2272108  0.17981635  0.02296036
## ksi_tsDim.3   -0.29593249  0.2412482  0.08377032 -0.16055951
## ksi_tsDim.4   -0.74926655  0.4278156  0.39008016  0.19267738
## ksi_tsDim.5    0.28958409 -0.7812686  1.23757484 -2.70416917
## ksi_tsDim.6    0.07048915  1.1263968 -1.74297851  1.27679624
## ksi_tsDim.7    1.87482142  0.9348885 -2.00438262 -2.75745452
## ksi_tsDim.8   -1.42417845  1.5501358 -6.03193103  2.93947778
## ksi_tsDim.9    2.53588904  1.3814895 -2.25812747  2.81471747
## ksi_tsDim.10  -3.22787061 -0.1673414  3.64040442 -0.29934065
## ksi_tsDim.11  -0.43632292  0.6355837  3.17373568  3.47656588
## ksi_tsDim.12   1.81104574  1.4275629 -4.80721412  4.24709709
## ksi_tsDim.13   5.67935223 -4.3478390 -2.20728810  0.48302689
## ksi_tsDim.14 -12.24445569 -2.4108450  1.78435846  8.96871358
## ksi_tsDim.15  -0.96553635 -3.1019740  5.73949160  6.06656735
```

```r
####### same for sparse PCA :

# residuals of the VAR(1)
res_spca <- residuals(ar_spca)
G_spca <- res_spca[indices_dates[-1],] # we drop the first element of res (ar(1) has one obs less)
G_spca <- t(G_spca)

eta_spca <- G_spca %*% t(V) %*% solve(V %*% t(V))


gamma_g_spca <- eta_spca %*% gamma
```