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.

Once the principal components are extracted, we use them as factors in asset pricing models. The goal is to determine whether those factors are relevant and whether they generate significant risk premia. The estimation of the of the risk premia uses the **three-pass methodology** developed by Giglio and Xiu. Their methodology is designed to compute unbiased risk premia estimates under omission of relevant risk factors and measurement error. The concern about factor omission is indeed well founded. If we assume that the asset excess returns are only determined by the macro factors derived from the PCA, we might omit other relevant factors. The three-pass methodology solves this problem.

[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)

```r
library(dplyr)

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

x <- data0$sasdate
# 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

```
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
```

```
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
```

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

```
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.

```
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 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, trace=F)
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)
```

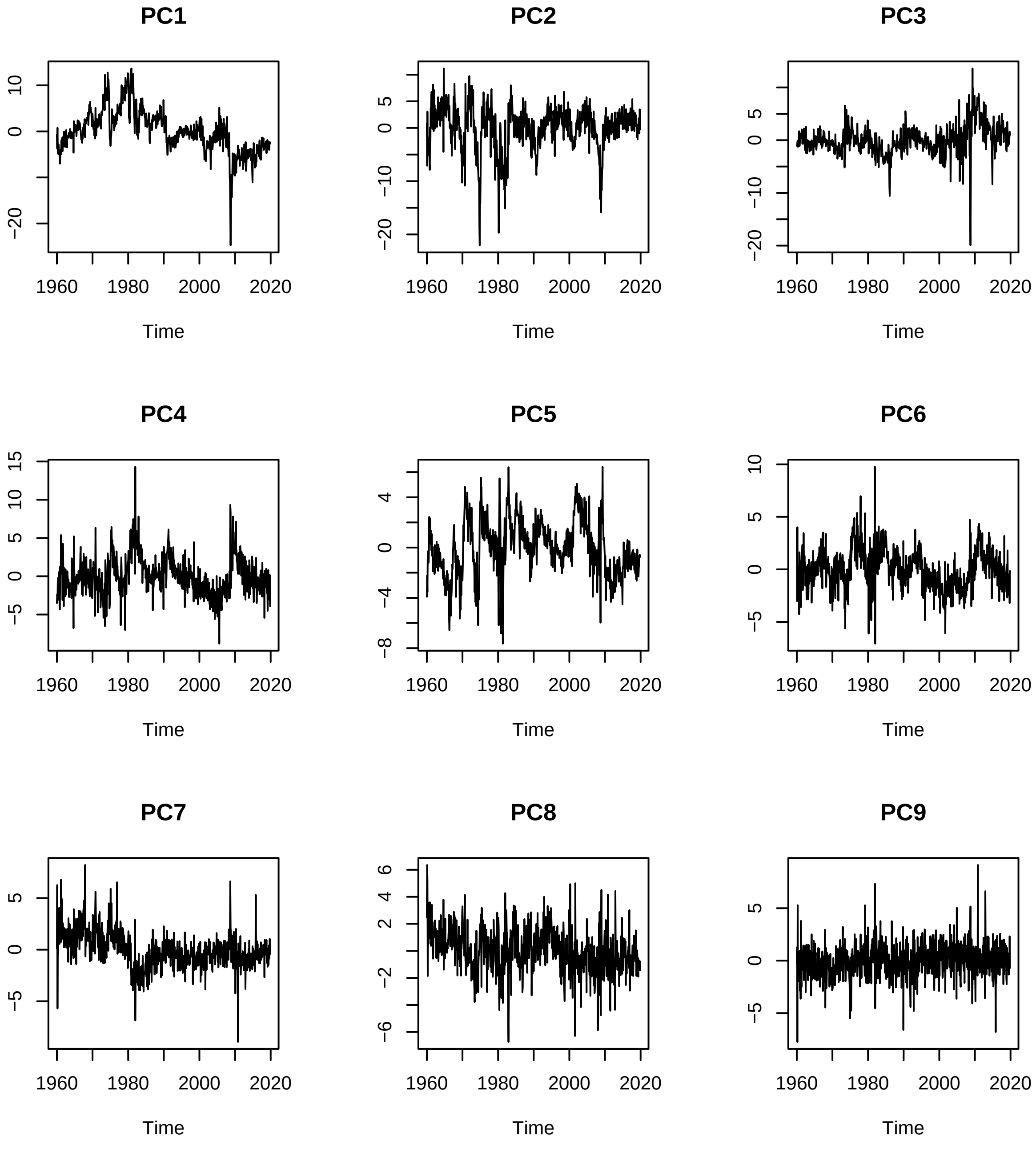

Figure 1: Conventional principal components

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
u <- scale(spca$u)

spca_ts <- ts(data=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="")
}
```

Even though our sparse components have similar interpretations as those derived by the authors, our plots are very different from those presented in **Figure 2** of the article

Innovations to the PCs

The set of macro factors is composed of the innovations to the principal components which have been extracted by the PCA. The innovations are computed by running a first-order vector autoregression (VAR(1)) on the principal components. For both the conventional and sparse PCAs, we run a VAR(1) on the PCs, we compute the residuals (which correspond to the innovations) and we then compute the correlations between those residuals.

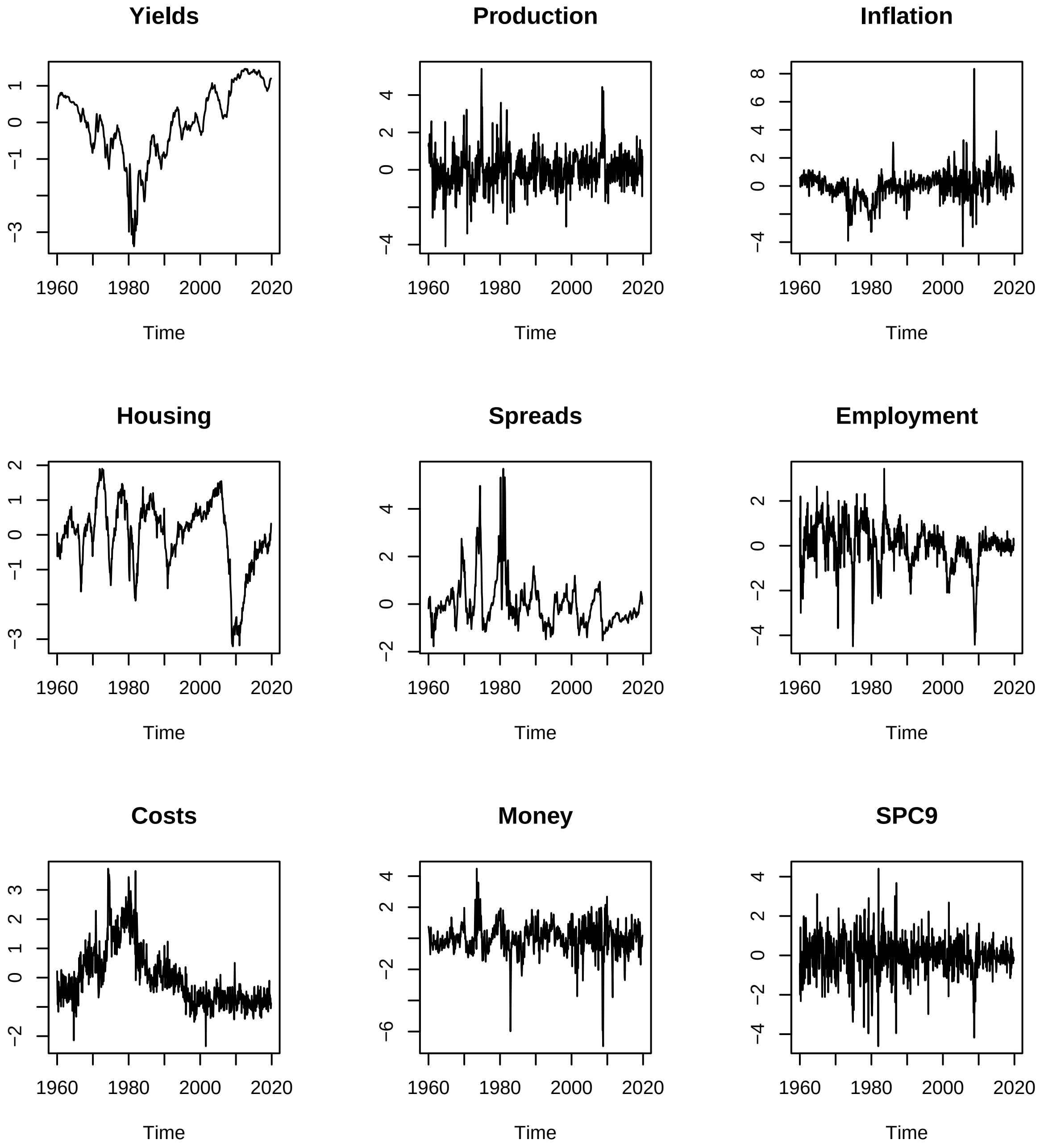

Figure 2: Sparse principal components

Conventional PCA

We begin with the conventional PCA. `pca$ind$coord` contains the coordinates of each of the 120 macroeconomic variables in the space of the 9 PCs. We use the package `vars` to run the VAR(1).

```
library(vars)
```

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

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

```
data_pca <- pca$ind$coord
row.names(data_pca) <- data$date
ar_pca <- VAR(data_pca, p=1)
correlations_pca <- round(cor(residuals(ar_pca)),2)
kable(correlations_pca, caption = "Innovation correlations to conventional PCs")
```

Table 4: Innovation correlations to conventional PCs

	Dim.1	Dim.2	Dim.3	Dim.4	Dim.5	Dim.6	Dim.7	Dim.8	Dim.9
Dim.1	1.00	0.39	0.79	0.07	0.14	-0.03	-0.07	-0.02	-0.05
Dim.2	0.39	1.00	0.09	0.70	-0.11	-0.18	-0.23	-0.12	-0.12
Dim.3	0.79	0.09	1.00	-0.15	0.45	-0.06	-0.03	0.01	0.04
Dim.4	0.07	0.70	-0.15	1.00	-0.09	-0.52	0.16	-0.11	0.03
Dim.5	0.14	-0.11	0.45	-0.09	1.00	-0.13	0.07	-0.05	-0.09
Dim.6	-0.03	-0.18	-0.06	-0.52	-0.13	1.00	-0.04	-0.21	-0.02
Dim.7	-0.07	-0.23	-0.03	0.16	0.07	-0.04	1.00	-0.19	0.09
Dim.8	-0.02	-0.12	0.01	-0.11	-0.05	-0.21	-0.19	1.00	0.05
Dim.9	-0.05	-0.12	0.04	0.03	-0.09	-0.02	0.09	0.05	1.00

The results of this correlation matrix are very close to the one displayed in **Table 4** of the original article.

Sparse PCA

We follow the same method with the sparse PCA. Here, the coordinates of each of the 120 macroeconomic variables in the space of the 9 sparse PCs are stored in `spca$u`.

```
data_spca <- scale(spca$u)
row.names(data_spca) <- data$date
colnames(data_spca) <- component_names
ar_spca <- VAR(data_spca, p=1)
correlations_spca <- round(cor(residuals(ar_spca)),2)
kable(correlations_spca, caption = "Innovation correlations to sparse PCs")
```

Table 5: Innovation correlations to sparse PCs

	Yields	Production	Inflation	Housing	Spreads	Employment	Costs	Money	SPC9
Yields	1.00	0.14	0.12	0.08	-0.11	-0.10	-0.08	-0.20	-0.10
Production	0.14	1.00	0.03	-0.17	-0.03	-0.49	-0.10	-0.02	-0.58
Inflation	0.12	0.03	1.00	-0.07	-0.04	-0.09	-0.24	-0.58	-0.13
Housing	0.08	-0.17	-0.07	1.00	-0.04	0.19	0.00	-0.05	0.38
Spreads	-0.11	-0.03	-0.04	-0.04	1.00	0.04	-0.01	0.11	0.01
Employment	-0.10	-0.49	-0.09	0.19	0.04	1.00	0.08	0.07	0.42
Costs	-0.08	-0.10	-0.24	0.00	-0.01	0.08	1.00	0.07	0.00
Money	-0.20	-0.02	-0.58	-0.05	0.11	0.07	0.07	1.00	0.08

	Yields	Production	Inflation	Housing	Spreads	Employment	Costs	Money	SPC9
SPC9	-0.10	-0.58	-0.13	0.38	0.01	0.42	0.00	0.08	1.00

Once again, our results look quite similar to those of the original article, except for the ninth sparse PC. However, for some correlations, the reported sign is the opposite of the one indicated in the original article.

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 <- 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).

```
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_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`)

```
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,graph = F)

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
df_pca <- data.frame(Factor = paste0("PC ", 1:9),
                     gamma_g=gamma_g)
kable(df_pca, caption = "Estimators of the risk premia for the conventional PCA")
```

Table 6: Estimators of the risk premia for the conventional PCA

	Factor	gamma_g
Dim.1	PC 1	-0.2045160
Dim.2	PC 2	-1.6692103
Dim.3	PC 3	0.2112325
Dim.4	PC 4	-1.1171945
Dim.5	PC 5	0.2812944
Dim.6	PC 6	-0.0945888
Dim.7	PC 7	-0.1963948
Dim.8	PC 8	0.4166464
Dim.9	PC 9	0.1918584

```r
# with tslm
library(forecast)
```

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

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

####### 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
df_spca <- data.frame(Factor = component_names,
                     gamma_g=gamma_g_spca)
kable(df_spca, caption = "Estimators of the risk premia for the sparse PCA")
```

Table 7: Estimators of the risk premia for the sparse PCA

	Factor	gamma_g
Yields	Yields	0.0139576
Production	Production	0.5255091
Inflation	Inflation	-0.1311787
Housing	Housing	-0.0054916
Spreads	Spreads	0.0016294
Employment	Employment	-0.4756850
Costs	Costs	-0.0894712
Money	Money	0.2579683
SPC9	SPC9	-0.2508959

Three-pass methodology, with notations of Zhou-Rapach

J'ai essayé de répliquer le résultat en utilisant les notations de notre article (Zhou-Rapach).

Importation of asset returns

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

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]
}
R <- t(R)
```

First, we stationarize and normalise our data

```
#the stationarize version
R_sta <- matrix(0,nrow=nrow(R),ncol = (ncol(R)-1))
for (i in 1:nrow(R_sta)){
  R_sta[i,] <- diff(R[i,])
}

#the demeanded version
R_d <- R_sta
for (i in 1:nrow(R_d)){
  R_d[i,] <- R_sta[i,] - mean(R_sta[i,])
}
```

1- Apply conventional PCA to demeaned excess returns for the N test assets to estimate the rotated fundamental factors STEP 1

```
t <- ncol(R)
n <- nrow(R)
mat <- (t(R_d) %*% R_d)/(t*n)
r_pca <- PCA(mat, ncp=15, graph=F, scale.unit = TRUE)

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

#normalize Vt
for (i in 1:nrow(V)){
  V[i,] <- (1/sqrt(t))*(V[i,])/sqrt(var(V[i,]))
}
#V %*% t(V) #it's now eq to the identity of dim 15 = p_hat

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

2- Run time-series regressions of r on ksi to estimate beta Run a cross-sectional regression of r__bar on the columns of beta to estimate gamma Pb : time-regressions de r ou de r demeaned? Je pense que c'est r demeaned (mais ça équivaut normalement à régresser r avec constante)

```
r_mean <- matrix(rowMeans(R_sta)) #average return
gamma <- solve(t(beta)%*%beta) %*% t(beta) %*% r_mean #OLS
```

```r
#Now, we calculate Rf2 associated with this cross-sectional reg
r_mean_mean <- mean(r_mean) #almost 0 by construction
Rf2 <- sum((beta %*% gamma - r_mean_mean)**2) / sum((r_mean-r_mean_mean)**2)
Rf2
```

```
## [1] 0.4945828
```

STEP 2 - variant avec OLS

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

```
## 
## Call:
## lm(formula = r_mean ~ beta)
## 
## Residuals:
##       Min        1Q    Median        3Q       Max 
## -0.016690 -0.001599  0.000168  0.001830  0.009673 
## 
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)    
## (Intercept)  0.005176   0.002672   1.937  0.05360 .  
## betaDim.1   -0.015336   0.012588  -1.218  0.22400    
## betaDim.2   -0.028100   0.006543  -4.295 2.31e-05 ***
## betaDim.3    0.040245   0.015867   2.536  0.01166 *  
## betaDim.4    0.028457   0.012974   2.193  0.02898 *  
## betaDim.5    0.023143   0.010850   2.133  0.03366 *  
## betaDim.6   -0.068836   0.016343  -4.212 3.27e-05 ***
## betaDim.7    0.028611   0.015865   1.803  0.07223 .  
## betaDim.8   -0.043887   0.015614  -2.811  0.00524 ** 
## betaDim.9   -0.005115   0.016989  -0.301  0.76354    
## betaDim.10   0.074361   0.016447   4.521 8.59e-06 ***
## betaDim.11  -0.041440   0.015991  -2.591  0.00998 ** 
## betaDim.12   0.026547   0.019276   1.377  0.16939    
## betaDim.13  -0.051394   0.022219  -2.313  0.02134 *  
## betaDim.14  -0.041297   0.018465  -2.237  0.02599 *  
## betaDim.15   0.003620   0.022158   0.163  0.87032    
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## 
## Residual standard error: 0.002994 on 329 degrees of freedom
## Multiple R-squared:  0.4803, Adjusted R-squared:  0.4566 
## F-statistic: 20.27 on 15 and 329 DF,  p-value: < 2.2e-16
```

```r
#on trouve le même R² à la main avec l'étape d'avant
```

3. Run time-series regressions of g on ksi to estimate theta

```r
# we restrict the observed factors to the good time period
dates_pca <- data$sasdate
# we drop the first element of res (ar(1) has one obs less)
indices_dates <- dates_pca>="1963-08-01" & dates_pca<= "2019-12-01"
#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],]
G <- t(G)
#shall we normalize? I don't think

#to use the OLS, we delet the constant using the time average
G_d <- G - rowMeans(G)
eta <- G_d %*% t(V) %*% solve(V %*% t(V))
gamma_g <- eta %*% gamma
rg <- 1:9
for (i in 1:9){
  rg[i] <- (sum(((eta %*% V)[i,] - rowMeans(G_d)[i])**2))/(sum((G_d[i,] - rowMeans(G_d)[i])**2))
}
df_pca <- data.frame(Factor = paste0("PC ", 1:9),
                     gamma_g=gamma_g, Rg = rg)
kable(df_pca, caption = "Estimators of the risk premia for the conventional PCA")
```

Table 8: Estimators of the risk premia for the conventional PCA

	Factor	gamma_g	Rg
Dim.1	PC 1	0.2163436	0.0207982
Dim.2	PC 2	0.1703469	0.0151173
Dim.3	PC 3	0.2309174	0.0154250
Dim.4	PC 4	-0.0874224	0.0094812
Dim.5	PC 5	-0.0943458	0.0112453
Dim.6	PC 6	0.0087065	0.0158669
Dim.7	PC 7	-0.4860649	0.0324882
Dim.8	PC 8	-0.2009284	0.0217213
Dim.9	PC 9	-0.1300165	0.0280519

```
# with tslm
#library(forecast)
#G_ts <- ts(t(G))
#ksi_ts <- ts(t(ksi))
#lm3 <- tslm(G_ts~0+ksi_ts)

####### 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)
G_spca_d <- G_spca - rowMeans(G_spca)
eta_spca <- G_spca_d %*% t(V) %*% solve(V %*% t(V))
gamma_g_spca <- eta_spca %*% gamma
rgs <- 1:9
for (i in 1:9){
  rgs[i] <- (sum(((eta_spca %*% V)[i,] - rowMeans(G_spca_d)[i])**2))/(sum((G_spca_d[i,] - rowMeans(G_spc
}
df_spca <- data.frame(Factor = component_names,
                     gamma_g=gamma_g_spca, Rg = rgs)
```

```r
kable(df_spca, caption = "Estimators of the risk premia for the sparse PCA")
```

Table 9: Estimators of the risk premia for the sparse PCA

	Factor	gamma_g	Rg
Yields	Yields	0.0275326	0.0485138
Production	Production	0.1049744	0.0118606
Inflation	Inflation	0.0276291	0.0160758
Housing	Housing	-0.0137221	0.0248810
Spreads	Spreads	0.0921908	0.0219630
Employment	Employment	0.0425052	0.0073500
Costs	Costs	0.1080195	0.0226510
Money	Money	0.0682238	0.0176477
SPC9	SPC9	0.2024095	0.0156179

Modifs

STEP1- Apply conventional PCA to demeaned excess returns for the N test assets to estimate the rotated fundamental factors STEP 1

```r
r_t <- t(R) # excess returns, one row per date

r_pca <- PCA(r_t, ncp=15, graph=F, scale.unit = TRUE)
ksi <- r_pca$ind$coord #rotated factors
```

STEP 2- Run time-series regressions of r on ksi to estimate beta. Run a cross-sectional regression of r_bar on the columns of beta to estimate gamma Pb : time-regressions de r ou de r demeaned? Je pense que c'est r demeaned (mais ça équivaut normalement à régresser r avec constante)

```r
lm_step2 <- tslm(ts(r_t)~ts(ksi)) #without constant

beta <- t(lm_step2$coefficients)
beta <- beta[,-1] # we drop the constant

r_bar <- colMeans(t(R)) #average return
lm_step2_CS <- lm(r_bar~beta+0) # no intercept in the model (euation 3.3)
summary(lm_step2_CS)
```

```
##
## Call:
## lm(formula = r_bar ~ beta + 0)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.43017 -0.06579  0.00426  0.06979  0.38689
##
## Coefficients:
##                    Estimate Std. Error  t value Pr(>|t|)
## betats(ksi)Dim.1  -10.39620    0.01910 -544.271  < 2e-16 ***
## betats(ksi)Dim.2   -1.48255    0.01914  -77.474  < 2e-16 ***
## betats(ksi)Dim.3    0.24949    0.01860   13.411  < 2e-16 ***
## betats(ksi)Dim.4   -0.57035    0.01842  -30.967  < 2e-16 ***
## betats(ksi)Dim.5   -0.12294    0.01733   -7.096 7.89e-12 ***
## betats(ksi)Dim.6   -0.26862    0.01817  -14.780  < 2e-16 ***
## betats(ksi)Dim.7    0.22274    0.01823   12.221  < 2e-16 ***
## betats(ksi)Dim.8   -0.06708    0.01732   -3.874 0.000129 ***
## betats(ksi)Dim.9    0.03106    0.01880    1.652 0.099492 .
## betats(ksi)Dim.10  -0.14594    0.01827   -7.990 2.28e-14 ***
## betats(ksi)Dim.11   0.17223    0.01766    9.751  < 2e-16 ***
## betats(ksi)Dim.12   0.20244    0.01816   11.148  < 2e-16 ***
## betats(ksi)Dim.13  -0.08104    0.01609   -5.036 7.82e-07 ***
## betats(ksi)Dim.14   0.05553    0.01815    3.060 0.002399 **
## betats(ksi)Dim.15   0.04418    0.01889    2.339 0.019917 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.1205 on 330 degrees of freedom
## Multiple R-squared:  0.9989, Adjusted R-squared:  0.9989
## F-statistic: 2.019e+04 on 15 and 330 DF,  p-value: < 2.2e-16
```

```r
gamma <- matrix(coefficients(lm_step2_CS)) # without the constant

# R² très proche de l'article!
```

STEP 3. Run time-series regressions of g on ksi to estimate theta

```r
# we restrict the observed factors to the good time period
dates_pca <- data$sasdate
# we drop the first element of res (ar(1) has one obs less)
indices_dates <- dates_pca>="1963-07-01" & dates_pca<= "2019-12-01"

# residuals of the VAR(1)
g_t_conv <- ts(residuals(ar_pca)[indices_dates[-1],])

lm_factors_conv <- tslm(g_t_conv~ts(ksi)) # without constant (equation 3.2)
theta_conv <- t(coefficients(lm_factors_conv))
theta_conv <- theta_conv[,-1]

r_squared_g_conv <- vector()
# Computation of the R²
for(i in 1:9){
  lm_tmp <- lm(g_t_conv[,i]~ksi)
  r_squared_g_conv<-c(r_squared_g_conv, summary(lm_tmp)$r.squared)
}
r_squared_g_conv <- round(100*r_squared_g_conv,2)
```

Conclusion :

```r
gamma_g_conv <- theta_conv%*% gamma
df <- data.frame(Factor = paste0("PC",1:9),
                 gamma_g = round(gamma_g_conv,3),
                 R_g_squared = paste0(r_squared_g_conv,"%"))
kable(df, caption = "Estimators of the risk premia for the conventional PCA", row.names = F)
```

Table 10: Estimators of the risk premia for the conventional PCA

Factor	gamma_g	R_g_squared
PC1	0.087	7.61%
PC2	-0.044	2.67%
PC3	-0.024	7.91%

Factor	gamma_g	R_g_squared
PC4	0.075	3.37%
PC5	-0.042	5.82%
PC6	-0.038	2.61%
PC7	0.027	2.4%
PC8	0.137	4.73%
PC9	0.023	4.09%

On n'est pas trop loin des résultats de l'article!

Same method for SPCA

```r
g_t_sparse <- ts(residuals(ar_spca)[indices_dates[-1],])



lm_factors_sparse <- tslm(g_t_sparse~ts(ksi)) # without constant (equation 3.2)
theta_sparse <- t(coefficients(lm_factors_sparse))
theta_sparse <- theta_sparse[,-1]
View(theta_sparse)

r_squared_g_sparse <- vector()
# Computation of the R²
for(i in 1:9){
  lm_tmp <- lm(g_t_sparse[,i]~ksi)
  r_squared_g_sparse<-c(r_squared_g_sparse, summary(lm_tmp)$r.squared)
}
r_squared_g_sparse <- round(100*r_squared_g_sparse,2)


gamma_g_sparse <- theta_sparse%*% gamma
df <- data.frame(Factor = paste0("PC",1:9),
                 gamma_g = round(gamma_g_sparse,4),
                 R_g_squared = paste0(r_squared_g_sparse,"%"))
kable(df, caption = "Estimators of the risk premia for the sparse PCA", row.names = F)
```

Table 11: Estimators of the risk premia for the sparse PCA

Factor	gamma_g	R_g_squared
PC1	-0.0216	13.46%
PC2	0.0016	2.99%
PC3	-0.0071	7.41%
PC4	-0.0037	3.53%
PC5	0.0095	1.78%
PC6	0.0017	1.55%
PC7	0.0239	4.71%
PC8	0.0117	3.12%
PC9	0.0104	5.88%

Biases without the three-pass methodology

What happens if we do not use the 3-pass methodology?

The motivation for using the three-pass methodology is to avoid potential omitted factors bias. We now study whether this concern is relevant, i.e. whether there is evidence of such biases. To achieve this, we estimate the risk premia with a simple two-pass methodology, and then compare our results to the outcome of the three-pass methodology.

Let us therefore assume that the true model for asset returns only depends on our macro factors :

$\boldsymbol{r_t} = \boldsymbol{\alpha} + \boldsymbol{\beta}'\boldsymbol{\gamma} + \boldsymbol{\beta}'\boldsymbol{g_t} + \boldsymbol{\varepsilon_t}$

Where $\boldsymbol{g_t}$ are the macro factors (i.e. the innovations to the PCs) and $\boldsymbol{\gamma}$ the risk premia.

If this assumption is true, then we can derive unbiased estimates of the risk premia with a two-pass methodology. This methodology consists in two steps :

1. Time series regression of the demeaned asset excess returns on the innovations to the macro factors, to estimate the risk exposures of each asset ($\boldsymbol{\beta}$)
2. Cross-sectional regression of the average returns of each asset on the asset' risk exposures to estimate the risk premia ($\boldsymbol{\gamma}$)

We run this estimation on the macro factors obtained with the conventional PCA, and then on the sparse macro factors.

Conventional PCA Importation of returns

`g_t_conv`correspond to the conventional macro factors, i.e. the innovatiopns to the conventional PCs.

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

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]
}

r_t <- R

r_t <- ts(R) # excess returns

#R_d <- R-t(as.matrix(colMeans(R))) # excess returns
```

```
## TS regression
g_t_conv <- ts(residuals(ar_pca)[indices_dates[-1],])

lm_pca <- tslm(r_t~g_t_conv)
beta <- t(lm_pca$coefficients)

beta <- beta[,-1]#we drop the constants

# CS regression
r_bar <- colMeans(R)

lm_pca_2 <- lm(r_bar~beta)
summary(lm_pca_2)
```

```
##
```

```
## Call:
## lm(formula = r_bar ~ beta)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.79576 -0.09108  0.01090  0.10234  0.56196
##
## Coefficients:
##                     Estimate Std. Error t value Pr(>|t|)
## (Intercept)          -3.6884     0.1201 -30.715  < 2e-16 ***
## betag_t_convDim.1  -0.6246     0.1708  -3.657 0.000297 ***
## betag_t_convDim.2  -0.8329     0.3026  -2.752 0.006243 **
## betag_t_convDim.3   0.3252     0.1887   1.723 0.085731 .
## betag_t_convDim.4  -0.6278     0.2054  -3.057 0.002417 **
## betag_t_convDim.5   0.6811     0.1167   5.836 1.26e-08 ***
## betag_t_convDim.6   0.6500     0.1921   3.384 0.000799 ***
## betag_t_convDim.7  -0.5455     0.1699  -3.211 0.001450 **
## betag_t_convDim.8   1.1264     0.1636   6.885 2.85e-11 ***
## betag_t_convDim.9   0.4085     0.1359   3.007 0.002840 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.1783 on 335 degrees of freedom
## Multiple R-squared:  0.3059, Adjusted R-squared:  0.2873
## F-statistic: 16.41 on 9 and 335 DF,  p-value: < 2.2e-16
```

```
kable(coefficients(lm_pca_2))
```

	x
(Intercept)	-3.6883959
betag_t_convDim.1	-0.6245589
betag_t_convDim.2	-0.8329041
betag_t_convDim.3	0.3251857
betag_t_convDim.4	-0.6277976
betag_t_convDim.5	0.6811120
betag_t_convDim.6	0.6500034
betag_t_convDim.7	-0.5454852
betag_t_convDim.8	1.1263959
betag_t_convDim.9	0.4084767

```
g_t_sparse <- ts(residuals(ar_spca)[indices_dates[-1],])

lm_spca <- tslm(r_t~g_t_sparse)
beta_s <- t(lm_spca$coefficients)
beta_s <- beta_s[,-1]#we drop the constants


lm_spca_2 <- lm(r_bar~beta_s)
summary(lm_spca_2)
```

Sparse PCA

```
##
```

```
## Call:
## lm(formula = r_bar ~ beta_s)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.74678 -0.07905  0.00636  0.09389  0.48956
##
## Coefficients:
##                            Estimate Std. Error t value Pr(>|t|)
## (Intercept)               -4.459824   0.101639 -43.879  < 2e-16 ***
## beta_sg_t_sparseYields      0.054163   0.008281   6.540 2.30e-10 ***
## beta_sg_t_sparseProduction  0.487093   0.099155   4.912 1.41e-06 ***
## beta_sg_t_sparseInflation  -0.056045   0.061979  -0.904 0.366513
## beta_sg_t_sparseHousing    -0.010292   0.021462  -0.480 0.631856
## beta_sg_t_sparseSpreads    -0.200486   0.053265  -3.764 0.000197 ***
## beta_sg_t_sparseEmployment -0.017481   0.095673  -0.183 0.855130
## beta_sg_t_sparseCosts      -0.237301   0.055723  -4.259 2.68e-05 ***
## beta_sg_t_sparseMoney      -0.365995   0.070224  -5.212 3.28e-07 ***
## beta_sg_t_sparseSPC9       -0.092743   0.085350  -1.087 0.277987
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.1715 on 335 degrees of freedom
## Multiple R-squared:  0.3575, Adjusted R-squared:  0.3403
## F-statistic: 20.71 on 9 and 335 DF,  p-value: < 2.2e-16
```

Even though those estimates are biased, we find that the sparse components 1 and 4 (yield and housing) generate significant risk premia. This result is consistent with the result of the original article.