# Macroeconometrics and Machine Learning

Jean-Galaad BARRIERE, Flavien GERVOIS

19/01/2023

## Introduction

In financial econometrics, numerous approaches have been developed to explain the returns of assets. It is often assumed that the excess returns (i.e. the difference between the actual returns and the risk-free rate) 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.

Macroeconomic variables can also be used in asset pricing models. As already shown in the literature[1], some macroeconomic variables could generate risk premia. To illustrate, an asset which is more exposed to industrial production growth, inflation or housing construction index might command a higher return. Indeed, assuming that an investor wants to hold a well-diversified portfolio, he might be less interested in holding assets whose returns are highly correlated with the global state of the economy.

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 a number of macroeconomic variables. Selecting a few relevant explanatory variables among a large set of variables is typically a problem which can be solved using machine learning techniques. In this paper, we present and replicate a recent article by David Rapach and Guofu Zhou (Rapach and Zhou 2018), which uses some of those techniques. Their paper (to which we will refer as "the original article" in this document) 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.

The article compares the performance of conventional PCA and sparse PCA in asset return factor models. It begins by applying conventional and sparse PCA to a large set of macroeconomic variables. Once the principal components are extracted, they are used 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 (Giglio and Xiu 2021). 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.

The empirical results of the article demonstrate that sparse PCA is a valuable machine learning technique. In comparison to conventional PCA, sparse PCA leads to a a significant increase in the interpretability of the factors, without losing much in terms of explaining the total variations of the macroeconomic variables.

[1]See for example (Chen, Roll, and Ross 1986)

Furthermore, some of the macro factors generate significant risk premia, while this is not the case of the conventional macro factors.

In the following sections, we present and replicate the different steps of the article of Rapach and Zhou, and then extend some of their results.

## PCA and Sparse PCA

The central machine learning technique presented in this paper is the sparse PCA. It is an extension of the Principal Component Analysis in which the weight vectors are sparse. As a consequence, each principal component is a linear combination of a small number of variables. The degree of sparsity can be adjusted by a shrinkage parameter. The main advantage of this technique lies in interpretability. It is actually much easier to give an economic interpretation to a linear combination or a handful of economic variables than to to a linear combination of more than a hundred variables. For instance, if a PC is built out of a few price indices, we can interpret it as an index of inflation.

We perform both conventional and sparse PCA on a set of 120 macroeconomic variables from the FRED-MD database[2]. 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.

### Initial treatment of the variables

Before performing the sparse PCA, following Rapach and Zhou, we need to apply some treatments to the variables. They transform the variables to render them stationary and to take into account lags in the reporting. Their appendix precisely presents the indicates the treatments to achieve stationarity. They give less details on the treatment related to the lags (and do not list the variables which are reported with a two-month lag). We are nonetheless confident that our treatments are really close to the ones they perform.

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, what transformation should be performed on them (log, log growth, difference) and whether they have to be lagged. 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 (1960:02 to 2019:12).

```r
library(dplyr)

#importation of FRED data
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)

# Importation of csv file with variables metadata
df <- read.csv("data/variables.csv",sep=";")
```

[2]Downloaded from https://research.stlouisfed.org/econ/mccracken/fred-databases/

```r
# Selection of relevant variables
df <- filter(df,Inclusion==1)
var <- df$FRED_ticker
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]
  no_lag <- df$No_lag[df$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]))
    }
    if(transfo=="Log growth"){
      tmp <- data[,i]
      tmp <- tmp/lag(tmp)
      tmp<-log(tmp)
      data[,i]<-c(tmp)
    }
    if(no_lag==0){
      data[,i]<-c(0,data[-nrow(data),i])
    }
  }
}
```

```
## 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 a conventional 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

# replacement of the NAs
tmp<- data0[,67]
tmp[is.na(tmp)]<-mean(tmp,na.rm = T)
```

```r
data0[,67]<-tmp
#sum(is.na(data0)) #control absence of NA

## PCA :
pca <- PCA(data0, ncp=9, graph=F)
pca_v2 <- PCA(data0, ncp=15, graph=F)
table1 <- pca$eig
table1 <- round(table1,2)
```

```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 | 21.09 | 17.57 | 17.57 |
| comp 2 | 16.93 | 14.11 | 31.68 |
| comp 3 | 7.72 | 6.43 | 38.11 |
| comp 4 | 6.28 | 5.24 | 43.35 |
| comp 5 | 5.01 | 4.17 | 47.52 |
| comp 6 | 3.74 | 3.12 | 50.64 |
| comp 7 | 3.08 | 2.57 | 53.21 |
| comp 8 | 2.78 | 2.32 | 55.53 |
| comp 9 | 2.62 | 2.18 | 57.71 |

The first nine conventional PCs collectively explain 57.71% 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. Our plots are very similar, except for the sign of PC7.

```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="")
}
```

```r
sd_pc <- sapply(as.data.frame(pca$ind$coord),sd)
```

## 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. We set the parameter `sumabsv` to 3 to get a similar outcome. The `SPC()` function is used to calculate the weights of the sparse PCs. We use those weights to compute the sparse

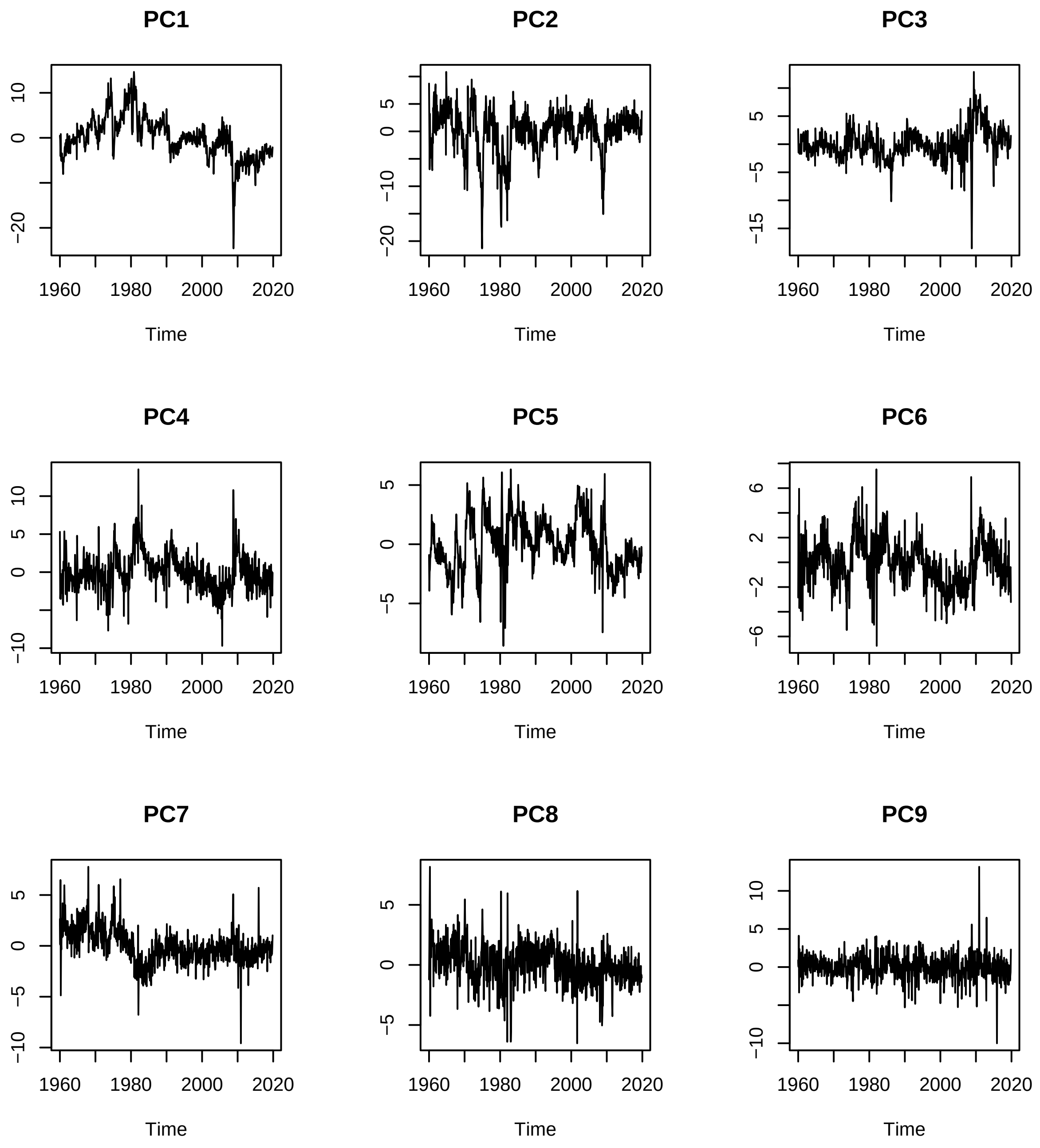

Figure 1: Conventional principal components

macro variables by multiplying the initial scaled macro variables by the matrix of the weights. The result is stored in the variable `u`.

```
library(PMA)

data_unscaled <- as.matrix(data0)
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] 106
```

```
# Percentage of variance
components <- paste0("comp ", 1:9)
table2 <- data.frame(Component = components,
                     Cumulative_percentage_of_variance = spca$prop.var.explained)
table2<-mutate(table2,
               Cumulative_percentage_of_variance = round(100*Cumulative_percentage_of_variance,2))
kable(table2, caption = "First 9 components of the SPCA")
```

Table 2: First 9 components of the SPCA

| Component | Cumulative_percentage_of_variance |
|---|---:|
| comp 1 | 7.09 |
| comp 2 | 13.27 |
| comp 3 | 19.72 |
| comp 4 | 25.85 |
| comp 5 | 31.49 |
| comp 6 | 36.82 |
| comp 7 | 40.45 |
| comp 8 | 43.30 |
| comp 9 | 46.35 |

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.

### Interpretation of the sparse factors

Looking at the active weights, we can try to give an economic interpretation to the sparse PCs. In the table below, we indicate the macro variables whose weights are active in the computation of each sparse PC. The weights that we computed do not exactly match those presented in ***Table 3*** of the original article. Nevertheless, based on the table below, we can give our sparse PC the same interpretation as Rapach and Zhou, except for the ninth component. For the latter, the weights of the ninth component diverge too much from those of the original article (only 3 common active variables). In our results, it is difficult to interpret this component as an index of credit. We therefore keep the name "SPC 9".

```
#### Identification of active weights
component_names <- c("Yields","Production", "Inflation", "Housing", "Spreads",
                     "Employment", "Costs", "Money", "SPC9")
active_weights<-rep("", 9)
for(i in 1:9){
```

```r
  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, caption = "Active weights of the sparse PCs")
```

Table 3: Active weights of the sparse PCs

| 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 | BUSINVx ; COMPAPFFx ; TB3SMFFM ; TB6SMFFM ; T1YFFM ; T5YFFM ; T10YFFM ; AAAFFM ; BAAFFM ; CPIMEDSL ; CUSR0000SAS ; DSERRG3M086SBEA |
| 6 | Employment | UNRATE ; 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 ; PPICMM ; MZMSL |
| 9 | SPC9 | DPCERA3M086SBEA ; CMRMTSPLx ; RETAILx ; IPNCONGD ; IPNMAT ; CE16OV ; CLAIMSx ; USCONS ; CES0600000007 ; AWOTMAN ; AWHMAN ; AMDMNOx ; ISRATIOx |

The interpretation of the sparse PCs is confirmed by looking at their plots. We note that for some of our PCs, the sign is opposite to the one found in the original article, notably for yields and housing. Apart from those sign differences, our plots are very close to the ones in ***Figure 2*** of the original article (except, once again, for the ninth component). On the plots below, we clearly see the (opposite) of the bust of the housing bubble and the sharp fall in employment during the 2008 economic crisis.

```r
v<-spca$v
u<-data0%*%v

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="")
}
```

## Innovations to the PCs

The set of macro factors used in the rest of the article 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

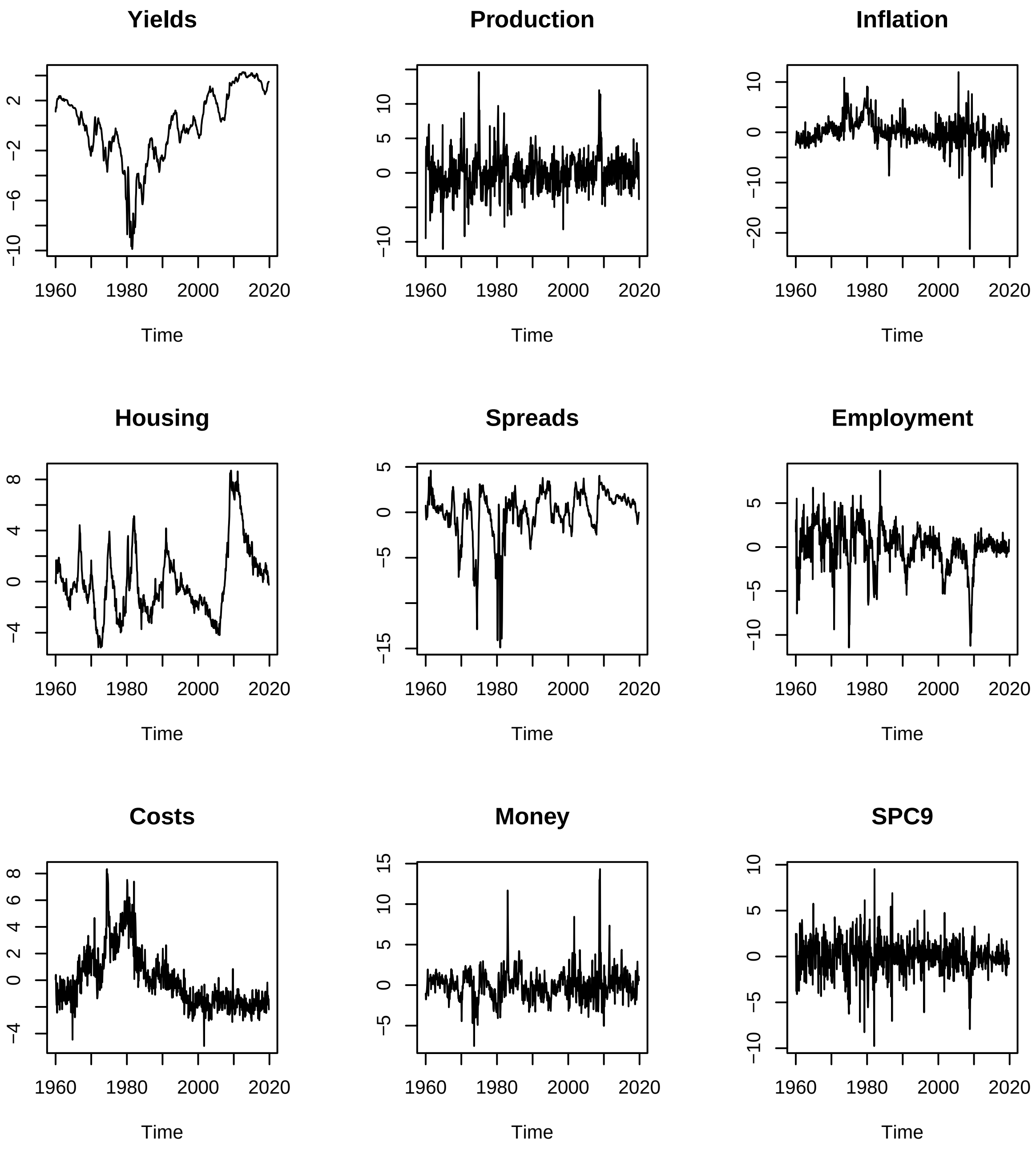

Figure 2: Sparse principal components

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. As noted by the authors, even though the PCs are by construction orthogonal to each other, this is not necessarily the case for their innovations.

### Conventional PCA

For the conventional PCA, the coordinates of each of the 120 macroeconomic variables in the space of the 9 PCs are stored in `pca$ind$coord` . 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
data_pca_v2 <- pca_v2$ind$coord
row.names(data_pca) <- data$date
ar_pca <- VAR(data_pca, p=1)
ar_pca_v2 <- VAR(data_pca_v2, 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 | 0.45 | 0.79 | 0.04 | 0.16 | 0.02 | -0.07 | -0.03 | -0.04 |
| Dim.2 | 0.45 | 1.00 | 0.17 | 0.67 | -0.08 | -0.13 | -0.23 | 0.08 | -0.11 |
| Dim.3 | 0.79 | 0.17 | 1.00 | -0.13 | 0.44 | -0.15 | -0.01 | -0.05 | 0.03 |
| Dim.4 | 0.04 | 0.67 | -0.13 | 1.00 | -0.17 | -0.47 | 0.18 | -0.06 | -0.08 |
| Dim.5 | 0.16 | -0.08 | 0.44 | -0.17 | 1.00 | -0.08 | 0.01 | 0.09 | -0.07 |
| Dim.6 | 0.02 | -0.13 | -0.15 | -0.47 | -0.08 | 1.00 | -0.10 | -0.16 | -0.16 |
| Dim.7 | -0.07 | -0.23 | -0.01 | 0.18 | 0.01 | -0.10 | 1.00 | -0.18 | -0.03 |
| Dim.8 | -0.03 | 0.08 | -0.05 | -0.06 | 0.09 | -0.16 | -0.18 | 1.00 | -0.02 |
| Dim.9 | -0.04 | -0.11 | 0.03 | -0.08 | -0.07 | -0.16 | -0.03 | -0.02 | 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. The coordinates of each of the 120 macroeconomic variables in the space of the 9 sparse PCs are stored in the variable `u`.

```
data_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 | 0.13 | -0.15 | 0.14 | 0.11 | -0.20 | -0.17 | 0.05 | -0.20 |

|  | Yields | Production | Inflation | Housing | Spreads | Employment | Costs | Money | SPC9 |
|---|---|---|---|---|---|---|---|---|---|
| Production | 0.13 | 1.00 | -0.02 | 0.18 | 0.07 | -0.50 | -0.09 | 0.03 | -0.61 |
| Inflation | -0.15 | -0.02 | 1.00 | -0.06 | -0.03 | 0.08 | 0.24 | -0.55 | 0.13 |
| Housing | 0.14 | 0.18 | -0.06 | 1.00 | -0.17 | -0.20 | 0.00 | -0.04 | -0.38 |
| Spreads | 0.11 | 0.07 | -0.03 | -0.17 | 1.00 | -0.06 | -0.11 | 0.09 | 0.05 |
| Employment | -0.20 | -0.50 | 0.08 | -0.20 | -0.06 | 1.00 | 0.07 | -0.08 | 0.43 |
| Costs | -0.17 | -0.09 | 0.24 | 0.00 | -0.11 | 0.07 | 1.00 | -0.09 | -0.01 |
| Money | 0.05 | 0.03 | -0.55 | -0.04 | 0.09 | -0.08 | -0.09 | 1.00 | -0.10 |
| SPC9 | -0.20 | -0.61 | 0.13 | -0.38 | 0.05 | 0.43 | -0.01 | -0.10 | 1.00 |

Once again, our results look very close to those of the original article, except for the ninth sparse PC. As noted above, due to differences in signs, some elements of our correlation matrix have the opposite signs of those presented in the original article.

## Risk premia estimates

In this section, we estimate the risk premia of the conventional and sparse macro factors derived in the preceding section. The objective is to determine whether some of the macro factors generate some significant risk premia. The computation uses the three-pass-methodology developed by Giglio and Xiu.

### Portfolio data

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

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).

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

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

First, we import the asset return

```
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)
```

### Three-pass methodology

We assume that the underlying model for the asset excess returns is the following :

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

Where $\boldsymbol{\xi_t}$ is the K-vector of unobserved fundamental factor innovations, $\boldsymbol{\beta}$ the fundamental factor exposures, $\boldsymbol{\gamma}$ the risk premia for the fundamental factors, and $\boldsymbol{\varepsilon_t}$ idiosyncratic errors.

The motivation for the three-pass methodology is to avoid omitted factor biases. The intuitive way to estimate the risk premia generated by our set of macro factors would be to use the following model :

$$\boldsymbol{r_t} = \boldsymbol{\alpha} + \beta'\gamma_{\boldsymbol{g}} + \beta'\boldsymbol{g_t} + \varepsilon_{\boldsymbol{t}}$$

Where $\boldsymbol{g_t}$ are the macro factors (i.e. the innovations to the PCs) and $\gamma_{\boldsymbol{g}}$ the risk premia for the macro factors. The risk premia would then be estimated by a simple two-pass methodology. However, such estimates would be biased due to potential omitted variables. By using the property of rotation invariance, the three-pass methodology makes it possible to derive unbiased estimates of the risk premia.

The methodology consists in three steps which are precisely described below.

### Step 1 : PCA on excess returns

The first step consists in applying conventional PCA to demeaned excess returns for the N test assets. We therefore estimate the rotated fundamental factors $\tilde{\boldsymbol{\xi}}_{\boldsymbol{t}}$.

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

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

### Step 2 : Time series and cross-sectional regressions

In the second step, we run a time-series regressions of r on $\tilde{\boldsymbol{\xi}}_{\boldsymbol{t}}$ to estimate $\tilde{\boldsymbol{\beta}}$. We can then run a cross-sectional regression of $\bar{r}$ on the columns of $\tilde{\boldsymbol{\beta}}$ to estimate $\tilde{\boldsymbol{\gamma}}$.

```r
library(forecast) #used for TS regression
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 (equation 3.3)

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

### Step 3 : Regressions of macro factors on asset return PCs

In this final third step we run a time-series regression of $\boldsymbol{g_t}$ on $\tilde{\boldsymbol{\xi}}_{\boldsymbol{t}}$ to estimate $\tilde{\boldsymbol{\Theta}}$.

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

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

The three-pass estimator of $\boldsymbol{\gamma}_g$ is the product of the estimates of $\tilde{\boldsymbol{\Theta}}$ and $\tilde{\boldsymbol{\gamma}}$.

```
gamma_g_conv <- theta_conv %*% gamma
```

### Calculation of the Covariance matrix

In order to calculate the $t_{stat}$, we need the Covariance matrix of $\hat{\gamma}_g$. This cannot be obtained easily, since $\hat{\gamma}_g$ is the product of two estimators ($\hat{\Theta}$ and $\hat{\gamma}$). The original paper does not explain how the t-stat are computed. Therefore, we had to use the cited paper of Giglio and Xiu to implement the computation of the $t_{stat}$. They found an asymptotic estimator of the Covariance matrix, which can be estimated with the following:

$$\hat{Cov(\ \hat{\gamma}_g)} = Mat.\hat{\Pi}_{11}.Mat^T + Mat.\hat{\Pi}_{12}.\hat{\Theta}^T + \hat{\Theta}.\hat{\Pi}_{21}.Mat^T + \hat{\Theta}.\hat{\Pi}_{22}.\hat{\Theta}^T$$

Where, if we define by $\tilde{\xi}$ the matrix storing every $\tilde{\boldsymbol{\xi}}_{\boldsymbol{t}}$ for $0 < t < T+1$, we get $\hat{\Sigma}^v = \tilde{\boldsymbol{\xi}}\tilde{\boldsymbol{\xi}}^T$ and $Mat^T = (\hat{\gamma}^T.(\hat{\Sigma}^v)^{-1} \otimes I_d)$

Here, $\hat{\Pi}_{11}$, $\hat{\Pi}_{12}$, $\hat{\Pi}_{21}$ and $\hat{\Pi}_{22}$ are the HAC estimators of Newey and West (1987). They can account for heterogeneity and autocorrelation standard error.

The common libraries used to calculate it do not work here, so we had to calculate them by ourselves, using the definitions given in the paper.

```
library(matrixcalc)
V <- t(ksi)
sigma_v <- V%*%t(V)
Z <- t(g_t_conv) - (theta_conv) %*% V

#First, we calculate the Newey West estimators
Pi_11 <- matrix(0,135,135)
Ti <- ncol(Z)
for (t in 1:Ti){
  Pi_11 <- Pi_11 + vec(Z[,t]%*%t(V[,t]))%*%t(vec(Z[,t]%*%t(V[,t])))
}
  #usually, the lag NW = 4, and then q+1=5
for (m in 1:4){
  for (t in (m+1):Ti){
    Pi_11 <- Pi_11 +
      (1-m/5)*(vec(Z[,t-m]%*%t(V[,t-m])) %*% t(vec(Z[,t]%*%t(V[,t]))) +
                 vec(Z[,t]%*%t(V[,t])) %*% t(vec(Z[,t-m]%*%t(V[,t-m]))))
  }
}
Pi_11 <- Pi_11/Ti

Pi_12 <- matrix(0,135,15)
for (t in 1:Ti){
  Pi_12 <- Pi_12 + vec(Z[,t]%*%t(V[,t]))%*%V[,t]
}
for (m in 1:4){
  for (t in (m+1):Ti){
    Pi_12 <- Pi_12 +
      (1-m/5)*(vec(Z[,t-m]%*%t(V[,t-m])) %*% V[,t]
               + vec(Z[,t]%*%t(V[,t])) %*% V[,t-m])
  }
}
Pi_12 <- Pi_12/Ti
```

```
Pi_21 <- matrix(0,15,135)
for (t in 1:Ti){Pi_21 <- Pi_21 + V[,t] %*% t(vec(Z[,t]%*%t(V[,t]))) }
for (m in 1:4){
  for (t in (m+1):Ti){
    Pi_21 <- Pi_21 +
      (1-m/5)*(V[,t-m] %*% t(vec(Z[,t]%*%t(V[,t])))
               + V[,t] %*% t(vec(Z[,t-m]%*%t(V[,t-m]))))
  }
}
Pi_21 <- Pi_21/Ti

Pi_22 <- matrix(0,15,15)
for (t in 1:Ti){Pi_22 <- Pi_22 + V[,t]%*%t(V[,t])}
for (m in 1:4){
  for (t in (m+1):Ti){
    Pi_22 <- Pi_22 +
      (1-m/5)*(V[,t-m] %*% t(V[,t]) + V[,t] %*% t(V[,t-m]))
  }
}
Pi_22 <- Pi_22/Ti

#Now, we calculate the asymptotic covariance matrix
Mat <- t(kronecker(t(gamma) %*% solve(sigma_v), diag(9), FUN = "*"))

Cov_gamma_g_conv <- t(Mat) %*% Pi_11 %*% Mat +
  t(Mat) %*% Pi_12 %*% t(theta_conv) +
  theta_conv %*% Pi_21 %*% Mat +
  theta_conv %*% Pi_22 %*% t(theta_conv)
```

Now, we can calculate the $R_g^2$ and the $t_{stat}$ as the paper did. The $t_{stat}$ are calculated with the formula

$$t_{\hat{\gamma_{gi}}} = \frac{\hat{\gamma_{gi}} - \bar{\gamma_g}}{se(\hat{\gamma_{gi}})}$$

with $\bar{\gamma_g} = 0$, and with $se(\hat{\gamma_{gi}})$ calculated with the Covariance matrix we now have.

```
# Computation of the Rg² and the t_stat
r_squared_g_conv <- vector()
t_stat_g_conv <- vector()
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)
  t_stat_g_conv <- c(t_stat_g_conv, gamma_g_conv[i,]/(Cov_gamma_g_conv[i,i]))
}
r_squared_g_conv <- round(100*r_squared_g_conv,2)
```

### Estimation results

**Conventional PCA** We can now print our estimations in the conventional PCA case. We are quite close to the results of the original article, regarding the $R_g^2$. We find that none of the conventional PCs generates a significant risk premium at the 90%-level, while the original article only finds a significant risk premium for the PC8. Our estimates of $\hat{\gamma_g}$ are however quite far from theirs. But since we know we do not have the same PCA as we've seen, although there have the same magnitude, except for the component.

```r
df <- data.frame(Factor = paste0("PC",1:9),
                 gamma_g = round(gamma_g_conv,3),
                 R_g_squared = paste0(r_squared_g_conv,"%"), t_stat = round(t_stat_g_conv,3))

kable(df, caption = "Estimators of the risk premia for the conventional PCA",
      row.names = F)
```

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

| Factor | gamma_g | R_g_squared | t_stat |
|---|---:|---|---:|
| PC1 | 0.146 | 3.14% | 0.703 |
| PC2 | -0.004 | 3.88% | -0.014 |
| PC3 | 0.052 | 2.11% | 0.443 |
| PC4 | 0.003 | 3.44% | 0.022 |
| PC5 | -0.004 | 1.16% | -0.193 |
| PC6 | 0.059 | 3.2% | 0.754 |
| PC7 | 0.011 | 2.76% | 0.223 |
| PC8 | -0.052 | 4.46% | -0.438 |
| PC9 | -0.076 | 2.76% | -0.892 |

**Sparse PCA** We now do the same with the vectors obtained with the sparse PCA. Here, our results appear to be much closer to the ones of the original article. As before, the $R_g^2$ are very close. As Zhou and Rapach, we find that the yields generate a significant risk premium at the 1% level. Surprisingly, our estimate of this risk premium is negative, while it should be negative (insofar as the sparse PC for yiels is the opposite of the one computed in the original article). Our sparse PC for housing is significant at the 20% level, while it is significant at the 1% level in the original article. More genrally, our risk premia estimates and t-stats are quite close to the ones of the original article, but some issues remain regarding the signs of the estimates.

```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]

Z <- t(g_t_sparse) - theta_sparse %*% V

#First, we calculate the Newey West estimators
Pi_11 <- matrix(0,135,135)
Ti <- ncol(Z)
for (t in 1:Ti){Pi_11 <- Pi_11 +
  vec(Z[,t]%*%t(V[,t]))%*%t(vec(Z[,t]%*%t(V[,t])))}
#usually, the lag NW = 4, thus q+1=5
for (m in 1:4){
  for (t in (m+1):Ti){
    Pi_11 <- Pi_11 +
      (1-m/5)*(vec(Z[,t-m]%*%t(V[,t-m])) %*% t(vec(Z[,t]%*%t(V[,t])))
               + vec(Z[,t]%*%t(V[,t])) %*% t(vec(Z[,t-m]%*%t(V[,t-m]))))
  }
}
Pi_11 <- Pi_11/Ti

Pi_12 <- matrix(0,135,15)
for (t in 1:Ti){Pi_12 <- Pi_12 + vec(Z[,t]%*%t(V[,t]))%*%V[,t]}
for (m in 1:4){
  for (t in (m+1):Ti){
    Pi_12 <- Pi_12 +
      (1-m/5)*(vec(Z[,t-m]%*%t(V[,t-m])) %*% V[,t]
               + vec(Z[,t]%*%t(V[,t])) %*% V[,t-m])
  }
}
Pi_12 <- Pi_12/Ti

Pi_21 <- matrix(0,15,135)
for (t in 1:Ti){Pi_21 <- Pi_21 + V[,t] %*% t(vec(Z[,t]%*%t(V[,t]))) }
for (m in 1:4){
  for (t in (m+1):Ti){
    Pi_21 <- Pi_21 +
      (1-m/5)*(V[,t-m] %*% t(vec(Z[,t]%*%t(V[,t])))
               + V[,t] %*% t(vec(Z[,t-m]%*%t(V[,t-m]))))
  }
}
Pi_21 <- Pi_21/Ti

Pi_22 <- matrix(0,15,15)
for (t in 1:Ti){Pi_22 <- Pi_22 + V[,t]%*%t(V[,t])}
for (m in 1:4){
  for (t in (m+1):Ti){
    Pi_22 <- Pi_22 +
      (1-m/5)*(V[,t-m] %*% t(V[,t])
               + V[,t] %*% t(V[,t-m]))
  }
}
Pi_22 <- Pi_22/Ti

#Now, we calculate the asymptotic covariance matrix
Mat <- t(kronecker(t(gamma) %*% solve(sigma_v), diag(9), FUN = "*"))

Cov_gamma_g_sparse <- t(Mat) %*% Pi_11 %*% Mat +
  t(Mat) %*% Pi_12 %*% t(theta_sparse) +
  theta_sparse %*% Pi_21 %*% Mat +
  theta_sparse %*% Pi_22 %*% t(theta_sparse)

gamma_g_sparse <- theta_sparse%*% gamma
r_squared_g_sparse <- vector()
t_stat_g_sparse <- vector()

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)
  t_stat_g_sparse <- c(t_stat_g_sparse,
                       gamma_g_sparse[i,]/(Cov_gamma_g_sparse[i,i]))
}
r_squared_g_sparse <- round(100*r_squared_g_sparse,2)

df <- data.frame(Factor = component_names,
                 gamma_g = round(gamma_g_sparse,4),
                 R_g_squared = paste0(r_squared_g_sparse,"%"),
                  t_stat = round(t_stat_g_sparse, 3))
kable(df, caption = "Estimators of the risk premia for the sparse PCA",
      row.names = F)
```

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

| Factor | gamma_g | R_g_squared | t_stat |
|---|---:|---|---:|
| Yields | -0.0666 | 13.67% | -2.441 |
| Production | 0.0171 | 3.41% | 0.072 |
| Inflation | 0.0929 | 2.06% | 0.665 |
| Housing | -0.0136 | 4.23% | -1.128 |
| Spreads | -0.0241 | 2.19% | -0.877 |
| Employment | 0.0569 | 2.35% | 0.748 |
| Costs | 0.0525 | 1.72% | 1.916 |
| Money | -0.0342 | 4.65% | -0.240 |
| SPC9 | -0.0154 | 5.09% | -0.085 |

## Extensions

### 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 assets' 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.

```
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
```

```

## 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)

kable(summary(lm_pca_2)$coefficients)
```

**Conventional PCA**

| | Estimate | Std. Error | t value | Pr(>\|t\|) |
|---|---|---|---|---|
| (Intercept) | -3.9151195 | 0.1073252 | -36.4790222 | 0.0000000 |
| betag_t_convDim.1 | -0.3670601 | 0.2174917 | -1.6876969 | 0.0924002 |
| betag_t_convDim.2 | -0.2741985 | 0.3017416 | -0.9087195 | 0.3641515 |
| betag_t_convDim.3 | 0.2307868 | 0.2397331 | 0.9626823 | 0.3364013 |
| betag_t_convDim.4 | -0.5677929 | 0.2159251 | -2.6295830 | 0.0089433 |
| betag_t_convDim.5 | 0.6723108 | 0.1719860 | 3.9091019 | 0.0001121 |
| betag_t_convDim.6 | 0.5697628 | 0.1772629 | 3.2142245 | 0.0014352 |
| betag_t_convDim.7 | 0.1832690 | 0.1705663 | 1.0744738 | 0.2833837 |
| betag_t_convDim.8 | 0.2010384 | 0.1593724 | 1.2614381 | 0.2080287 |
| betag_t_convDim.9 | -0.4838797 | 0.1432874 | -3.3769869 | 0.0008191 |

```

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)
kable(summary(lm_spca_2)$coefficients)
```

**Sparse PCA**

| | Estimate | Std. Error | t value | Pr(>\|t\|) |
|---|---|---|---|---|
| (Intercept) | -4.3375227 | 0.1256350 | -34.5247828 | 0.0000000 |
| beta_sg_t_sparseYields | 0.1078119 | 0.0236522 | 4.5582190 | 0.0000072 |
| beta_sg_t_sparseProduction | 0.7668328 | 0.2611474 | 2.9363985 | 0.0035502 |
| beta_sg_t_sparseInflation | 0.4954408 | 0.2465886 | 2.0091797 | 0.0453199 |
| beta_sg_t_sparseHousing | -0.2620175 | 0.0499521 | -5.2453742 | 0.0000003 |
| beta_sg_t_sparseSpreads | 0.2048431 | 0.1210783 | 1.6918228 | 0.0916097 |
| beta_sg_t_sparseEmployment | -0.0864510 | 0.2111826 | -0.4093660 | 0.6825327 |
| beta_sg_t_sparseCosts | 0.1884424 | 0.1319554 | 1.4280765 | 0.1542014 |
| beta_sg_t_sparseMoney | -0.2123499 | 0.1441289 | -1.4733331 | 0.1416002 |

| | Estimate | Std. Error | t value | Pr(>\|t\|) |
|---|---|---|---|---|
| beta_sg_t_sparseSPC9 | -0.1257930 | 0.1817974 | -0.6919406 | 0.4894539 |

Our estimates are significantly differetn from the ones computed with the three-pass methodology. This suggests that the concerns of the authors about omitted factor bias is well-founded. 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.

## Very sparse PCA

We further extend the results of the article by implementing a “very sparse PCA”. We change the shrinkage parameter so as to obtain sparser principal components. In the most extreme case, we obtain sparse PC which are composed of only one variable. We present below the results of our estimations with this “very sparse PCA”. Building sparse PC with only one variable is equivalent to successively choose the variable which explains the largest part of the residual variations of the dataset. Of course, those PCs are even simpler to interpret, as they only correspond to one variable. To make the interpretation easier, we replaced the negative weights -1 by +1 (because the sparse components 1, 2, 4 and 9 are built with weight -1). We see that our very sparse PC can have the same interpretation as in the original article, even for the ninth component (which now really corresponds to credit). However, those very sparse PCs only explain 7.50% of the total variance of the dataset. The plots of the very sparse PCs confirm that we can give the same the same interpretation as in the original article.

| Component | Cumulative_percentage_of_variance |
|---|---|
| comp 1 | 0.83 |
| comp 2 | 1.67 |
| comp 3 | 2.50 |
| comp 4 | 3.33 |
| comp 5 | 4.17 |
| comp 6 | 5.00 |
| comp 7 | 5.83 |
| comp 8 | 6.67 |
| comp 9 | 7.50 |

| Sparse_Component | Component_name | Active_weights |
|---|---|---|
| 1 | Yields | FEDFUNDS |
| 2 | Production | IPMANSICS |
| 3 | Inflation | CPITRNSL |
| 4 | Housing | PERMIT |
| 5 | Spreads | T5YFFM |
| 6 | Employment | USTPU |
| 7 | Costs | CES0600000008 |
| 8 | Money | M2SL |
| 9 | Credit | CONSPI |

We then compute the estimation of the risk premia associated with our very sparse macro factors. The estimates of $\gamma_g$ are quite close to the ones we computed in the preceding section, especially for the first 7 SPCs. As in the original article, yields and housing generate significant risk premia. But we note that our measures of inflation, spreads and costs also generate significant risk premia.

To conclude on this extension, sparse PCA can be used to extract the macro variables which explain the largest part of the total variation of the dataset. Using the three-pass methodology on 9 factors, we find that 4 out of 9 generate a significant risk premia (at the 95% level).

## Overcoming the linearity in the modelisation

### *What happens if we get rid of the linearity assumption?*

The model adopted in the paper, leading to the three-steps methodology, is based on two equations: $\boldsymbol{r_t} = \beta'.\gamma + \beta'.\boldsymbol{\xi_t} + \boldsymbol{\epsilon_t}$, (1) giving $\boldsymbol{r_t} = \tilde{\beta}'\tilde{\gamma} + \tilde{\beta}'\tilde{\boldsymbol{\xi}}_{\boldsymbol{t}} + \boldsymbol{\epsilon_t}$, $(1bis)$ and $\bar{\boldsymbol{g}}_{\boldsymbol{t}} = \Theta'.\boldsymbol{\xi_t} + \boldsymbol{\zeta_t}$, (2) giving $\bar{\boldsymbol{g}}_{\boldsymbol{t}} = \tilde{\Theta}'\tilde{\boldsymbol{\xi}}_{\boldsymbol{t}} + \zeta_t$, $(2bis)$,

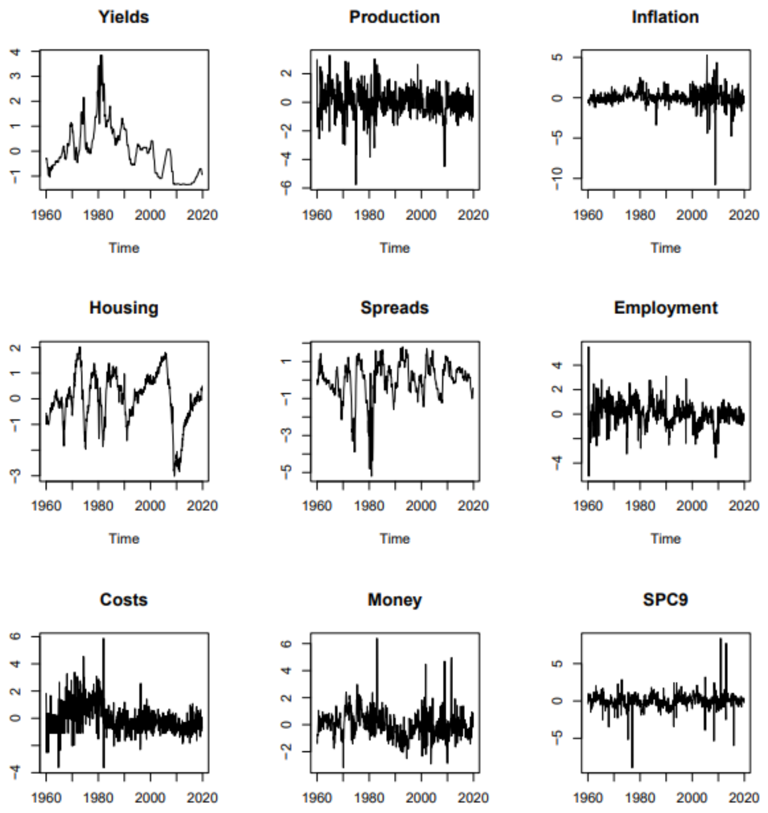

Figure 3: Plot of the very sparse PCs

| Factor | gamma_g | R_g_squared | t_stat |
|---|---:|---|---:|
| Yields | 0.0232 | 7.72% | 6.675 |
| Production | -0.0026 | 3.35% | -0.086 |
| Inflation | 0.0430 | 2.03% | 1.914 |
| Housing | 0.0091 | 5.45% | 3.723 |
| Spreads | -0.0150 | 3.2% | -2.508 |
| Employment | 0.0222 | 3.27% | 1.048 |
| Costs | 0.0583 | 2.19% | 2.115 |
| Money | 0.0067 | 5.56% | 0.157 |
| Credit | -0.0022 | 2.38% | -0.086 |

Figure 4: Risk premia estimates of the very sparse macro factors

where $\bar{\boldsymbol{g}}_{\boldsymbol{t}} = \boldsymbol{g}_{\boldsymbol{t}} - \boldsymbol{\delta}$. The meaning of the first equation is simply that we extracted a new base $(\tilde{\boldsymbol{\xi}}_{\boldsymbol{t}})_{t<T+1}$ with a PCA on the $(r_t)_i$, and now we express the $(r_t)_i$ in the firsts components of this regression.

But the idea of the second regression is way more questionable: we aim to express the observable factors $(g_t)_i$ in the base $(\tilde{\boldsymbol{\xi}}_{\boldsymbol{t}})_{t<T+1}$. $\Theta$ is then the base change matrix. But this has nothing evident: it only makes sense if we consider, either that the $(g_t)_i$ are very closed to each other, either that the "true" function linking the $(g_t)_i$ with the excess returns $(r_t)_t$ is linear (insofar as the base $(\tilde{\boldsymbol{\xi}}_{\boldsymbol{t}})_{t<T+1}$ is built on linear combinations of $(r_t)_i$).

Therefore, we now develop a non linear approach to link $(g_t)_i$ and $(r_t)_i$. In the three-steps methodology, for identification issues, we considered the rotated variables $\tilde{\boldsymbol{\xi}}_{\boldsymbol{t}}$ which are a linear transformation of the true unobserved fundamental factors.

Since we do not have a lot of data at our disposal, we should keep using the few vectors of the base $\tilde{\boldsymbol{\xi}}_{\boldsymbol{t}}$ instead of the many vectors of $(r_t)_i$ to help our model to converge quickly. We can write the new equation $(2')$ this way: $\bar{\boldsymbol{g}}_{\boldsymbol{t}} = \mathcal{F}(\boldsymbol{\xi}_{\boldsymbol{t}}) + \boldsymbol{z}^*_{\boldsymbol{t}}$ with $\boldsymbol{z}^*_{\boldsymbol{t}}$ the new residual.

This leads to the equation we will use:

$$\bar{\boldsymbol{g}}_{\boldsymbol{t}} = \tilde{\mathcal{F}}(\tilde{\boldsymbol{\xi}}_{\boldsymbol{t}}) + \boldsymbol{z}^*_{\boldsymbol{t}}(2bis')$$

In this new model, the desired $\gamma_g$ is now equal to $\gamma_g = \tilde{\mathcal{F}}(\tilde{\gamma})$.

In this precise case, we obviously have no preferred model relating our observable $(g_t)_i$ to $(r_t)_i$. Therefore we cannot really choose a specific parametric modelization.

Among all the non-parametric modelizations, we need one able to be able to return efficiently multidimensional output, since $(g_t)_i$ is multidimensional. That is why we choose the Vector Generalized Additive Model (VGAM).

We can then add a fourth step to the three-steps methodology, namely: "Train a VAGM function on $(\bar{\boldsymbol{g}}_{\boldsymbol{t}})_t$ and $(\tilde{\boldsymbol{\xi}}_{\boldsymbol{t}})_t$ to obtain $\hat{\tilde{\mathcal{F}}} = \mathcal{F}_{VGAM}$".

Since our data can take negative values, we had to choose a family of probability functions working on $\mathbb{R}$. After several tests, it appears that the gaussian function perform the best. Therefore we kept it.

We ran two differents regressions, a "smooth" one and a "non-smooth" one. With the smooth regression, we authorize interpolation of degree 4 (the default value if we authorize it) on each $(\tilde{\xi}_{ti})_{t<T+1}$, whereas we do not authorize it in the non-smooth one.

```
library(VGAM)
res <- data.frame(gt=g_t_conv, ksi=ksi)
gamma_df <- data.frame(t(gamma))
names(gamma_df) <- names(res)[10:24]

gam_smooth<-vgam(cbind(gt.Dim.1, gt.Dim.2, gt.Dim.3, gt.Dim.4, gt.Dim.5, gt.Dim.6,
                       gt.Dim.7, gt.Dim.8, gt.Dim.9)~ s(ksi.Dim.1)
                 +s(ksi.Dim.2)+s(ksi.Dim.3)+ s(ksi.Dim.4)+s(ksi.Dim.5)
                 +s(ksi.Dim.6)+ s(ksi.Dim.7)+s(ksi.Dim.8)+s(ksi.Dim.9)
                 +s(ksi.Dim.10)+s(ksi.Dim.11)+s(ksi.Dim.12)+s(ksi.Dim.13)
                 +s(ksi.Dim.14)+s(ksi.Dim.15), uninormal(), data=res)
gamma_g_conv_VGAM_smooth <- as.matrix(predict(gam_smooth,
                                              gamma_df,
                                              type="response"))

gam<-vgam(cbind(gt.Dim.1, gt.Dim.2, gt.Dim.3, gt.Dim.4, gt.Dim.5, gt.Dim.6,
                gt.Dim.7, gt.Dim.8, gt.Dim.9)
          ~ (ksi.Dim.1)+(ksi.Dim.2)+(ksi.Dim.3)+ (ksi.Dim.4)+(ksi.Dim.5)+
            (ksi.Dim.6)+ (ksi.Dim.7)+(ksi.Dim.8)+(ksi.Dim.9)+(ksi.Dim.10)+
            (ksi.Dim.11)+(ksi.Dim.12)+(ksi.Dim.13)+(ksi.Dim.14)+(ksi.Dim.15),
          uninormal(), data=res)
gamma_g_conv_VGAM <- as.matrix(predict(gam, gamma_df, type="response"))
```

This non-linear method results in estimates of $\gamma_g$ which are close to the ones we obtained with the previous three-steps methodology, specially in the non-smooth regression. It enforces us in the idea that this method can work, although we don't have a lot of data to train our VGAM (T=678).

Evaluating the efficiency of our new algorithm is more difficult. We cannot express the $t_{stat}$ as previously, because the formula we used for the covariances worked only under linearity assumptions. We could lineralise our $\hat{\mathcal{F}}_{VGAM}$ to obtain an estimate of $\hat{\hat{\Theta}}$ and re-use the formula, but it would not be very precise. And since $\gamma_g$ is a product of estimates, it seems that we cannot obtain an easy estimate of the $t_{stat}$ this time.

However, we can assess the efficiency of this new method in the third step by looking at the mean squared residuals on each dimension. The results make it clear that the MSR of the non-smooth VGAM are the same as in the original three-step methodology, while the ones of the smooth VGAM regression are smaller. The smooth VGAM regression therefore appears to be a great improvement.

However, we have to note that VGAM, especially smooth VGAM, are subject to overfitting, due to the high number of parameters they use and the little amount of data we have. Therefore, the good MSR we obtain with the smooth VGAM can either express a truth: the model is non linear and we need to smooth the VGAM. Or it can just be overfitting. To solve this, we could try some cross-validation, but we already have very few data at our disposal.

```
MeanR2_VGAM_smooth <- colMeans(resid(gam_smooth)[,2*c(1:9)-1]**2)
MeanR2_VGAM_ns <- colMeans(resid(gam)[,2*c(1:9)-1]**2)
MeanR2_old <- colMeans(lm_factors_conv$residuals**2)

df <- data.frame(gamma_1 = round(gamma_g_conv,2), MR2_1 = round(MeanR2_old,2),
                 gamma_2 = round(gamma_g_conv_VGAM,2),
                 MR2_2 = round(MeanR2_VGAM_ns,2),
                 gamma_3 = round(gamma_g_conv_VGAM_smooth,2),
                 MR2_3 = round(MeanR2_VGAM_smooth,2))

kable(df, caption = "Comparaison of the differents methods for the third-step,
      with the Conventional PCA. 1: 3-step methodology, 2: Not-smooth VGAM,
      3: smooth VGAM, gamma: gamma_g, MR2: Mean Square Error", row.names = F)
```

Table 10: Comparaison of the differents methods for the third-step, with the Conventional PCA. 1: 3-step methodology, 2: Not-smooth VGAM, 3: smooth VGAM, gamma: gamma_g, MR2: Mean Square Error

| gamma_1 | MR2_1 | gamma_2 | MR2_2 | gamma_3 | MR2_3 |
|---:|---:|---:|---:|---:|---:|
| 0.15 | 3.39 | 0.16 | 3.39 | 0.15 | 2.96 |
| 0.00 | 6.40 | 0.01 | 6.40 | 0.16 | 5.80 |
| 0.05 | 4.17 | 0.07 | 4.17 | -0.10 | 3.72 |
| 0.00 | 2.96 | -0.01 | 2.96 | -0.12 | 2.73 |
| 0.00 | 1.64 | 0.00 | 1.64 | 0.01 | 1.51 |
| 0.06 | 1.98 | 0.08 | 1.98 | 0.20 | 1.82 |
| 0.01 | 1.89 | -0.02 | 1.89 | -0.07 | 1.76 |
| -0.05 | 2.26 | -0.09 | 2.26 | 0.18 | 2.01 |
| -0.08 | 2.43 | -0.09 | 2.43 | -0.12 | 2.23 |

We also apply this extension to the sparse PCA. Once again, we see that our smooth VGAM method performs better in term of MSR, when the not-smooth VGAM method gives almost the same result as the original three-step methodology.

```r
res <- data.frame(gt=g_t_sparse, ksi=ksi)
gamma_df <- data.frame(t(gamma))
names(gamma_df) <- names(res)[10:24]

gam_smooth<-vgam(cbind(gt.Yields, gt.Production, gt.Inflation, gt.Housing,
                       gt.Spreads, gt.Employment, gt.Costs, gt.Money, gt.SPC9)~
                   s(ksi.Dim.1)+s(ksi.Dim.2)+s(ksi.Dim.3)+
                   s(ksi.Dim.4)+s(ksi.Dim.5)+s(ksi.Dim.6)+ s(ksi.Dim.7)
                 +s(ksi.Dim.8)+s(ksi.Dim.9)+s(ksi.Dim.10)+s(ksi.Dim.11)
                 +s(ksi.Dim.12)+s(ksi.Dim.13)+s(ksi.Dim.14)+s(ksi.Dim.15), uninormal(), data=res)
gamma_g_sparse_VGAM_smooth <- as.matrix(predict(gam_smooth,
                                                gamma_df,
                                                type="response"))

gam<-vgam(cbind(gt.Yields, gt.Production, gt.Inflation, gt.Housing,
                gt.Spreads, gt.Employment, gt.Costs, gt.Money, gt.SPC9)
          ~ (ksi.Dim.1)+(ksi.Dim.2)+(ksi.Dim.3)+ (ksi.Dim.4)+
            (ksi.Dim.5)+(ksi.Dim.6)+ (ksi.Dim.7)+(ksi.Dim.8)+
            (ksi.Dim.9)+(ksi.Dim.10)+(ksi.Dim.11)+(ksi.Dim.12)+
            (ksi.Dim.13)+(ksi.Dim.14)+(ksi.Dim.15), uninormal(), data=res)
gamma_g_sparse_VGAM <- as.matrix(predict(gam, gamma_df, type="response"))

MeanR2_VGAM_smooth <- colMeans(resid(gam_smooth)[,2*c(1:9)-1]**2)
MeanR2_VGAM_ns <- colMeans(resid(gam)[,2*c(1:9)-1]**2)
MeanR2_old <- colMeans(lm_factors_sparse$residuals**2)

df <- data.frame(gamma_1 = round(gamma_g_sparse,2),
                 MR2_1 = round(MeanR2_old,2),
                 gamma_2 = round(gamma_g_sparse_VGAM,2),
                 MR2_2 = round(MeanR2_VGAM_ns,2),
                 gamma_3 = round(gamma_g_sparse_VGAM_smooth,2),
                 MR2_3 = round(MeanR2_VGAM_smooth,2))

kable(df, caption = "Comparaison of the differents methods for the third-step,
      with the Sparse PCA. 1: 3-step methodology, 2: Not-smooth VGAM,
      3: smooth VGAM, gamma: gamma_g, MR2: Mean Square Error", row.names = F)
```

Table 11: Comparaison of the differents methods for the third-step, with the Sparse PCA. 1: 3-step methodology, 2: Not-smooth VGAM, 3: smooth VGAM, gamma: gamma_g, MR2: Mean Square Error

| gamma_1 | MR2_1 | gamma_2 | MR2_2 | gamma_3 | MR2_3 |
|---:|---:|---:|---:|---:|---:|
| -0.07 | 0.07 | -0.07 | 0.07 | -0.05 | 0.07 |
| 0.02 | 5.45 | 0.02 | 5.45 | 0.13 | 4.96 |
| 0.09 | 4.60 | 0.12 | 4.60 | 0.03 | 4.14 |
| -0.01 | 0.25 | -0.01 | 0.25 | -0.11 | 0.23 |
| -0.02 | 0.91 | -0.02 | 0.91 | 0.04 | 0.80 |
| 0.06 | 2.77 | 0.07 | 2.77 | 0.32 | 2.50 |
| 0.05 | 1.02 | 0.05 | 1.02 | 0.05 | 0.92 |

| gamma_1 | MR2_1 | gamma_2 | MR2_2 | gamma_3 | MR2_3 |
|---:|---:|---:|---:|---:|---:|
| -0.03 | 2.35 | -0.04 | 2.35 | 0.07 | 2.01 |
| -0.02 | 3.04 | 0.00 | 3.04 | 0.04 | 2.79 |

## Conclusion

To conclude, sparse PCA appears to be a relevant machine learning technique for asset pricing factor models, and can be seen as a balance between two extreme cases in the utilization of macroeconomic variables for asset pricing. On one hand, using a few arbitrary macroeconomic variables (as was done in the pioneering work of (Chen, Roll, and Ross 1986)) does not take into account the wide variety of macroeconomic variables available. Furthermore, an economic concept such as employment can be expressed with a wide number of variables (total employment, employment by sector, working hours etc.).

On the other hand, a conventional PCA can help reducing the dimension of the macroeconomic dataset while taking into account a wider share of the available macroeconomic data. But this comes at the cost of economic interpretability: the factors constructed by the PCA cannot be easily linked to economic concepts such as employment, inflation or housing.

Our article finds a sort a third way by using the sparse PCA. The sparse macro factors incorporate most of the available information (in terms of total variance) while still being interpretable in terms of economic concepts.

We applied both conventional and sparse PCA on a set of 120 macroeconomic variables from FRED-MD to construct macro factors. The risk premia of those macro factors were then estimated using a large number of American asset portfolio over the period 1963-2019.

We found that sparse macro factors measuring yields and housing generate significant risk premia. The estimation of the risk premia was based on the three-pass methodology, due to potential omitted factor bias when estimating a model only based on macro factors. Our extension provides some evidence that this concern is well-founded.

Finally, we extended the results by introducing very sparse PCA and non-linear relations between macro factors and unobserved fundamental factors.

## References

Chen, Nai-Fu, Richard Roll, and Stephen A. Ross. 1986. "Economic Forces and the Stock Market." *The Journal of Business* 59 (3): 383–403. https://www.jstor.org/stable/2352710.

Giglio, Stefano, and Dacheng Xiu. 2021. "Asset Pricing with Omitted Factors." *Journal of Political Economy* 129 (7): 1947–90. https://doi.org/10.1086/714090.

Rapach, David, and Guofu Zhou. 2018. "Sparse Macro Factors." *SSRN Electronic Journal*. https://doi.org/10.2139/ssrn.3259447.