Macro ML

Jean-Galaad BARRIERE, Flavien GERVOIS

15/01/2022

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 ("Asset Pricing with Omitted Factors | Journal of Political Economy: Vol 129, No 7," n.d.a). 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 techniques. 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)

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

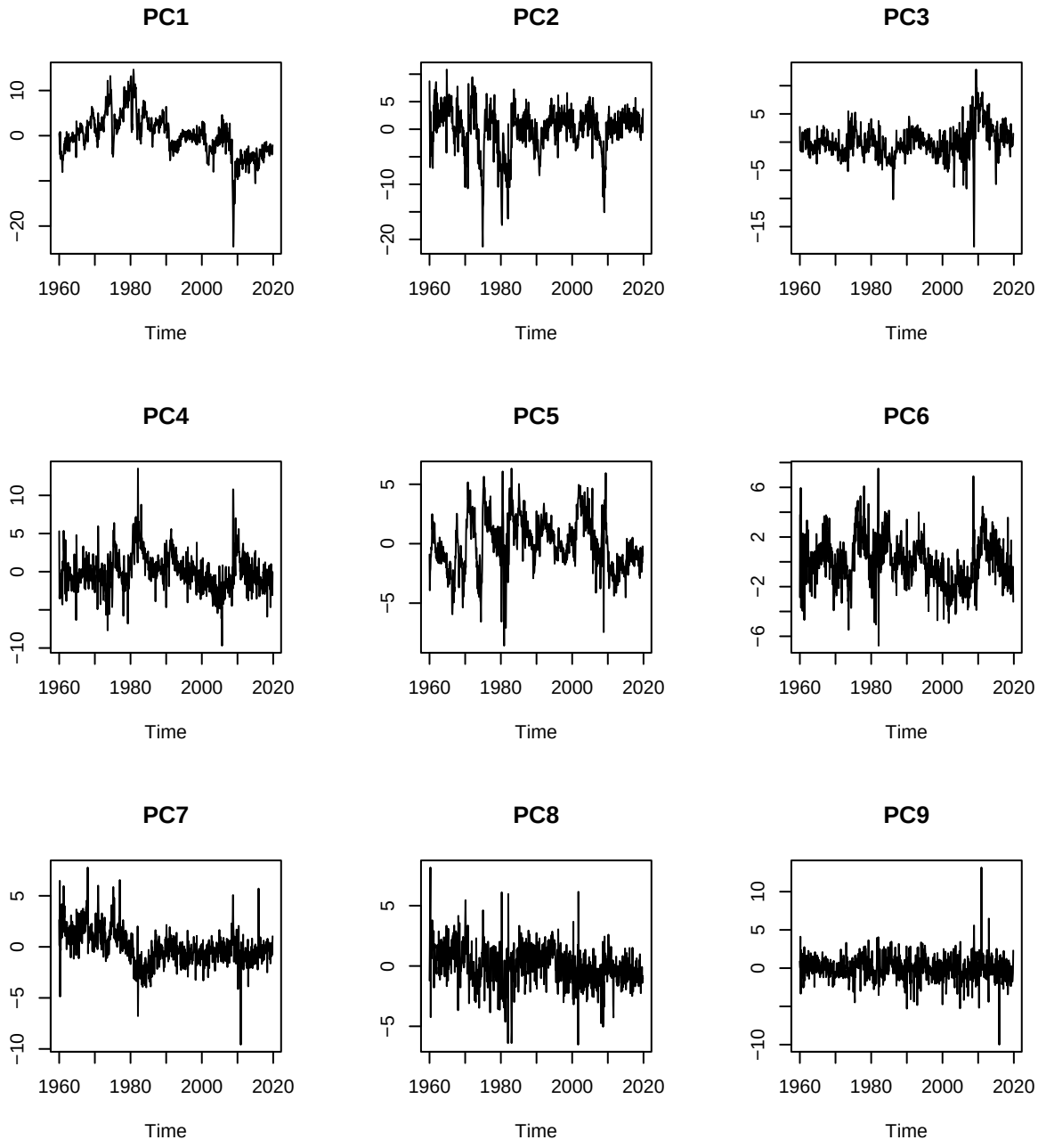

Figure 1: Conventional principal components

```r
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
```

```r
# 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. Nevertheles, 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".

```r
#### Identification of active weights
component_names <- c("Yields","Production", "Inflation", "Housing", "Spreads", "Employment", "Costs", "I
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, caption = "Active weights of the sparse PCs")
```

Table 3: Active weights of the sparse PCs

Sparse_Component_number	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.

```
#u <- scale(spca$u)*sd_pc
v<-spca$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 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

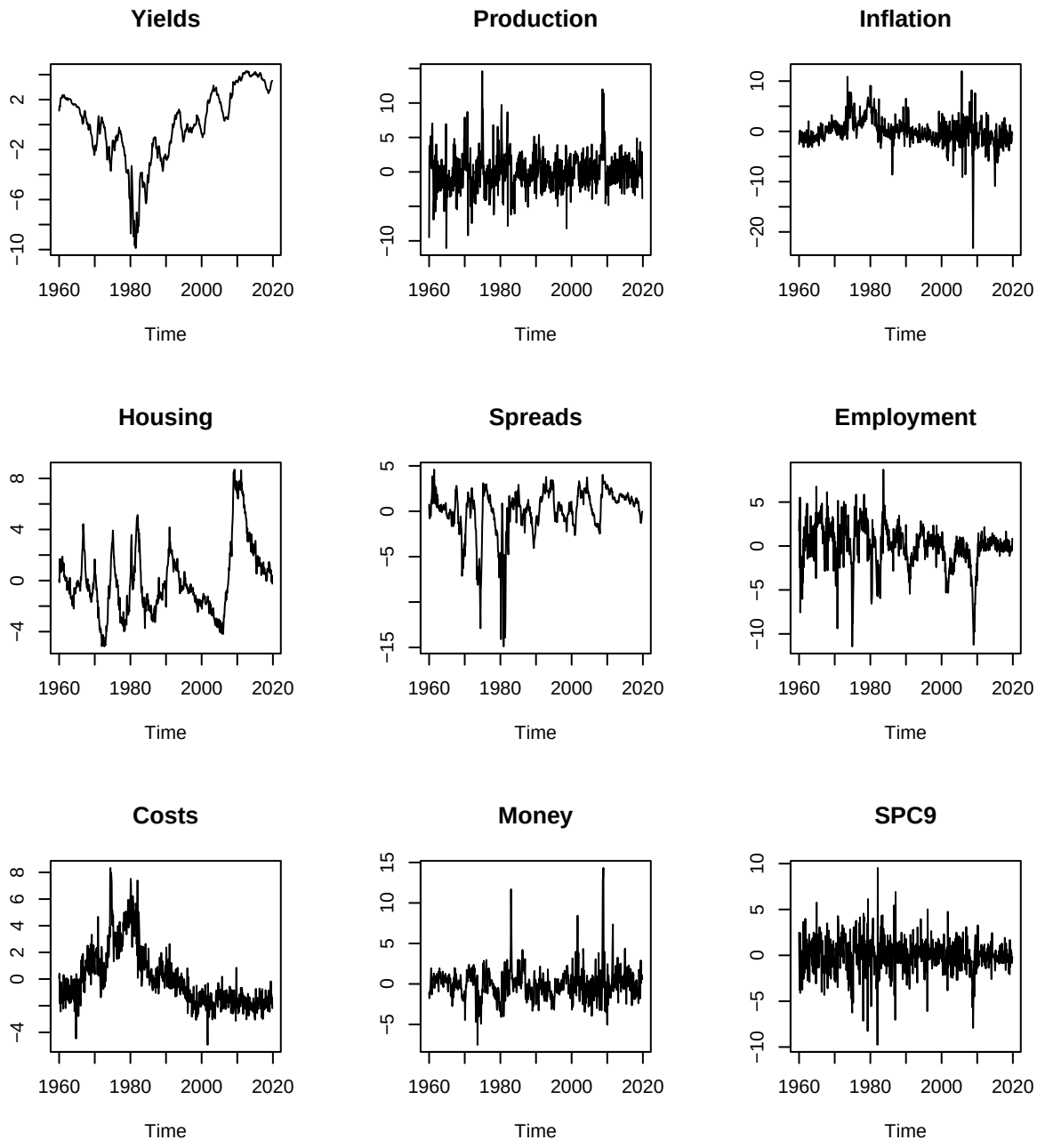

Figure 2: Sparse principal components

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
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	0.45	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
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

	Yields	Production	Inflation	Housing	Spreads	Employment	Costs	Money	SPC9
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 ("Asset Pricing with Omitted Factors | Journal of Political Economy: Vol 129, No 7," n.d.b).

Portfolio data

[A rédiger]

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.

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)
```

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, **[etc., à rédiger]**

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 :

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

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

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)

```
library(forecast) #used for TS regression
```

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

```
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)
#summary(lm_step2_CS)

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

#We store the covariance matrix for the latter calcul of t-stat
Cov_gamma <- NeweyWest(lm_step2_CS, lag = 4, prewhite = FALSE)
```

Step 3 : Regressions of macro factors on asset return PCs

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

Calculation of the Covariance matrix

We are using an asymptotic estimation of the covariance matrix of $\hat{\gamma}_g$. This is described in **je vais mettre le détail, ainsi que les formules liées et pourquoi j'ai pas pu utiliser de bibliothèques (sandwich ou plm) qui ne s'appliquent pas ici car on traite d'un produit**

```r
#V <- sqrt(nrow(ksi))*t(ksi)
#sigma_v <- (1/nrow(ksi))*V%*%t(V)
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(0,9,9)
Ti <- ncol(Z)
for (t in 1:Ti){Pi_11 <- Pi_11 + (Z[,t]%*%t(V[,t]))%*%t(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)*(Z[,t-m]%*%t(V[,t-m]) %*% t(Z[,t]%*%t(V[,t])) + Z[,t]%*%t(V[,t]) %*% t(Z[,
  }
}
Pi_11 <- Pi_11/Ti

Pi_12 <- matrix(0,9,1)
for (t in 1:Ti){Pi_12 <- Pi_12 + (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)*(Z[,t-m]%*%t(V[,t-m]) %*% V[,t] + Z[,t]%*%t(V[,t]) %*% V[,t-m])
  }
}
Pi_12 <- Pi_12/Ti

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

#Now, we calculate the asymptotic covariance matrix
```

```r
#this appears in the formula
Mat <- t(kronecker(t(gamma) %*% solve(sigma_v), rep(1,9), FUN = "*"))
#we use theta_conv = t(eta) and Pi_22 is a scalar
Cov_gamma_g_conv <- Mat %*% Pi_11 %*% t(Mat) + (Mat * as.vector(Pi_12)) %*% theta_conv + t((Mat * as.ve
```

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

r_squared_g_conv <- vector()
t_stat_g_conv <- vector()

# Computation of the R² and the t_stat
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,]/sqrt(Cov_gamma_g_conv[i,i]))
}
r_squared_g_conv <- round(100*r_squared_g_conv,2)
```

Estimation results

Conventional PCA [à rédiger]

For the calcul of each t-stats associated to each $\hat{\gamma_{gi}}$, we use the following definition, with $\bar{\gamma_g} = 0$:

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

The paper explain using an heteroskedastic-compatible and autocorellation-robust estimator, the Newey and West (1987) one. In R, we can get it (the variance, instead to the standard error) with the vcovNW function of the package plm.

[Flavien, penses-tu que je devrais détailler un peu comment ça fonctionne?] [Je ne pense pas]

```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%	0.438
PC2	-0.004	3.88%	-0.003
PC3	0.052	2.11%	0.041
PC4	0.003	3.44%	0.001
PC5	-0.004	1.16%	-0.001
PC6	0.059	3.2%	0.013
PC7	0.011	2.76%	0.001
PC8	-0.052	4.46%	-0.006
PC9	-0.076	2.76%	-0.015

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

Sparse PCA Same method for SPCA

```
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

Pi_11 <- matrix(0,9,9)
Ti <- ncol(Z)
for (t in 1:Ti){Pi_11 <- Pi_11 + (Z[,t]%*%t(V[,t]))%*%t(Z[,t]%*%t(V[,t]))}
for (m in 1:4){
  for (t in (m+1):Ti){
    Pi_11 <- Pi_11 + (1-m/5)*(Z[,t-m]%*%t(V[,t-m]) %*% t(Z[,t]%*%t(V[,t])) + Z[,t]%*%t(V[,t]) %*% t(Z[,
  }
}
Pi_11 <- Pi_11/Ti

Pi_12 <- matrix(0,9,1)
for (t in 1:Ti){Pi_12 <- Pi_12 + (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)*(Z[,t-m]%*%t(V[,t-m]) %*% V[,t] + Z[,t]%*%t(V[,t]) %*% V[,t-m])
  }
}
Pi_12 <- Pi_12/Ti

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

Mat <- t(kronecker(t(gamma) %*% solve(sigma_v), rep(1,9), FUN = "*"))
Cov_gamma_g_sparse <- Mat %*% Pi_11 %*% t(Mat) + (Mat * as.vector(Pi_12)) %*% theta_sparse + t((Mat * as

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

df <- data.frame(Factor = paste0("PC",1:9),
                 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
PC1	-0.0666	13.67%	-0.297
PC2	0.0171	3.41%	0.026
PC3	0.0929	2.06%	0.062
PC4	-0.0136	4.23%	-0.006
PC5	-0.0241	2.19%	-0.009
PC6	0.0569	2.35%	0.012
PC7	0.0525	1.72%	0.005
PC8	-0.0342	4.65%	-0.005
PC9	-0.0154	5.09%	-0.003

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 :

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

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
## -0.88551 -0.08092  0.02000  0.11239  0.45891
##
## Coefficients:
##                   Estimate Std. Error t value Pr(>|t|)
## (Intercept)        -3.9151     0.1073 -36.479  < 2e-16 ***
## betag_t_convDim.1  -0.3671     0.2175  -1.688 0.092400 .
## betag_t_convDim.2  -0.2742     0.3017  -0.909 0.364151
## betag_t_convDim.3   0.2308     0.2397   0.963 0.336401
## betag_t_convDim.4  -0.5678     0.2159  -2.630 0.008943 **
## betag_t_convDim.5   0.6723     0.1720   3.909 0.000112 ***
## betag_t_convDim.6   0.5698     0.1773   3.214 0.001435 **
## betag_t_convDim.7   0.1833     0.1706   1.074 0.283384
## betag_t_convDim.8   0.2010     0.1594   1.261 0.208029
## betag_t_convDim.9  -0.4839     0.1433  -3.377 0.000819 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.1886 on 335 degrees of freedom
## Multiple R-squared:  0.2229, Adjusted R-squared:  0.2021
## F-statistic: 10.68 on 9 and 335 DF,  p-value: 1.383e-14
```

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

	x
(Intercept)	-3.9151195
betag_t_convDim.1	-0.3670601
betag_t_convDim.2	-0.2741985
betag_t_convDim.3	0.2307868
betag_t_convDim.4	-0.5677929
betag_t_convDim.5	0.6723108
betag_t_convDim.6	0.5697628
betag_t_convDim.7	0.1832690
betag_t_convDim.8	0.2010384
betag_t_convDim.9	-0.4838797

```
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
## -0.94511 -0.07988  0.01465  0.10877  0.47096
##
## Coefficients:
##                            Estimate Std. Error t value Pr(>|t|)
## (Intercept)                -4.33752    0.12564 -34.525  < 2e-16 ***
## beta_sg_t_sparseYields      0.10781    0.02365   4.558 7.24e-06 ***
## beta_sg_t_sparseProduction  0.76683    0.26115   2.936  0.00355 **
## beta_sg_t_sparseInflation   0.49544    0.24659   2.009  0.04532 *
## beta_sg_t_sparseHousing    -0.26202    0.04995  -5.245 2.77e-07 ***
## beta_sg_t_sparseSpreads     0.20484    0.12108   1.692  0.09161 .
## beta_sg_t_sparseEmployment -0.08645    0.21118  -0.409  0.68253
## beta_sg_t_sparseCosts       0.18844    0.13196   1.428  0.15420
## beta_sg_t_sparseMoney      -0.21235    0.14413  -1.473  0.14160
## beta_sg_t_sparseSPC9       -0.12579    0.18180  -0.692  0.48945
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.1794 on 335 degrees of freedom
## Multiple R-squared:  0.2968, Adjusted R-squared:  0.2779
## F-statistic: 15.71 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.

Very sparse PCA

Replication with a different shrinkage parameter so as to select only one or two variables by PC.

References

"Asset Pricing with Omitted Factors | Journal of Political Economy: Vol 129, No 7." n.d.a. https://www.journals.uchicago.edu/doi/abs/10.1086/714090.

———. n.d.b. https://www.journals.uchicago.edu/doi/abs/10.1086/714090.

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.

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