Macroeconometrics and Machine Learning : Lasso method

Kilian ANDRU, Gaëlle DORMION

Import data

```
library(MTS)
```

```
## Warning: le package 'MTS' a été compilé avec la version R 4.3.2
```

```
library(fbi)
library(glmnet)
```

```
## Warning: le package 'glmnet' a été compilé avec la version R 4.3.2

## Le chargement a nécessité le package : Matrix

## Loaded glmnet 4.1-8
```

```
# Setting start and end date of the analysis
start_date = "1960-01-01"
end_date = "2019-12-01"         # We might want to leave out the covid period


# Load the most recent fred-md dataset

# Load transformed data
data = fredmd("https://files.stlouisfed.org/files/htdocs/fred-md/monthly/current.csv",
              date_start = as.Date(start_date), date_end = as.Date(end_date))


var_desc = fredmd_description

# Period
dates <- data$date
```

Data formatting

We remove useless columns and columns with NA values

```
#Remove the first column "date"
raw_data <- data[,seq(2,length(data))]
```

```r
series <- names(data)

#Remove columns with NA values
names(raw_data[, colSums(is.na(raw_data)) > 0])
```

```
## [1] "ACOGNO"         "ANDENOx"       "TWEXAFEGSMTHx" "UMCSENTx"
## [5] "VIXCLSx"
```

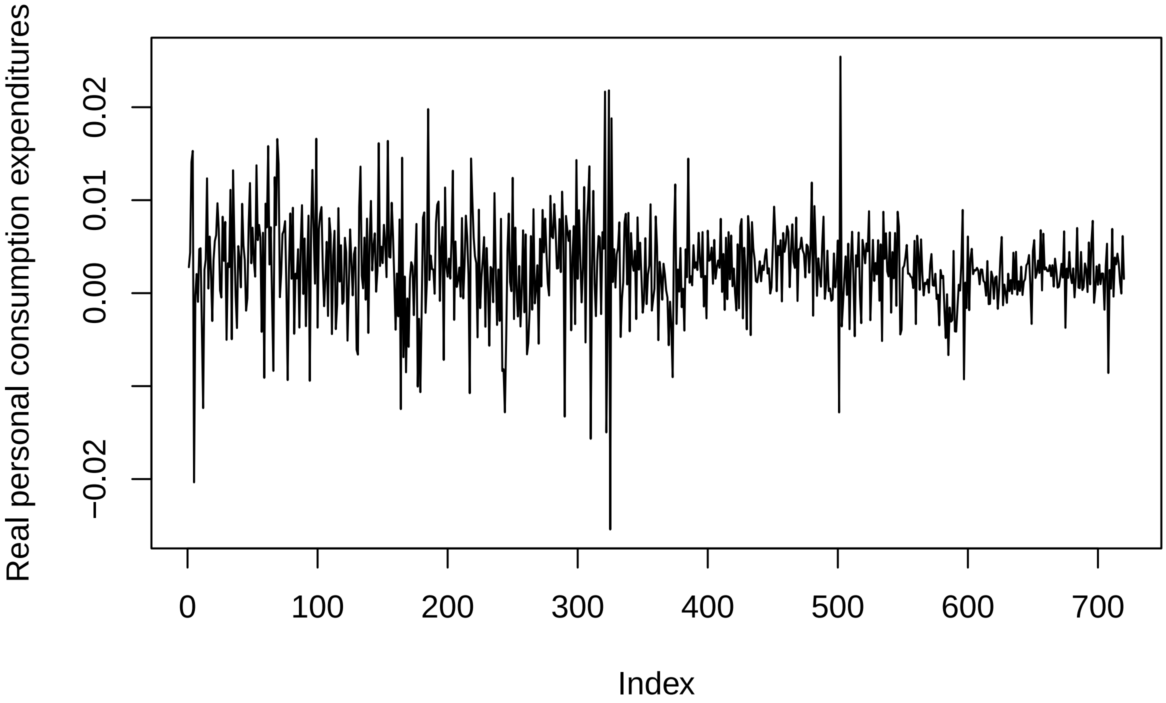
```r
x <- raw_data[, colSums(is.na(raw_data)) == 0]
```

```r
#We check if there are still missing values
table(sapply(x, function(x) sum(length(which(is.na(x))))))
```

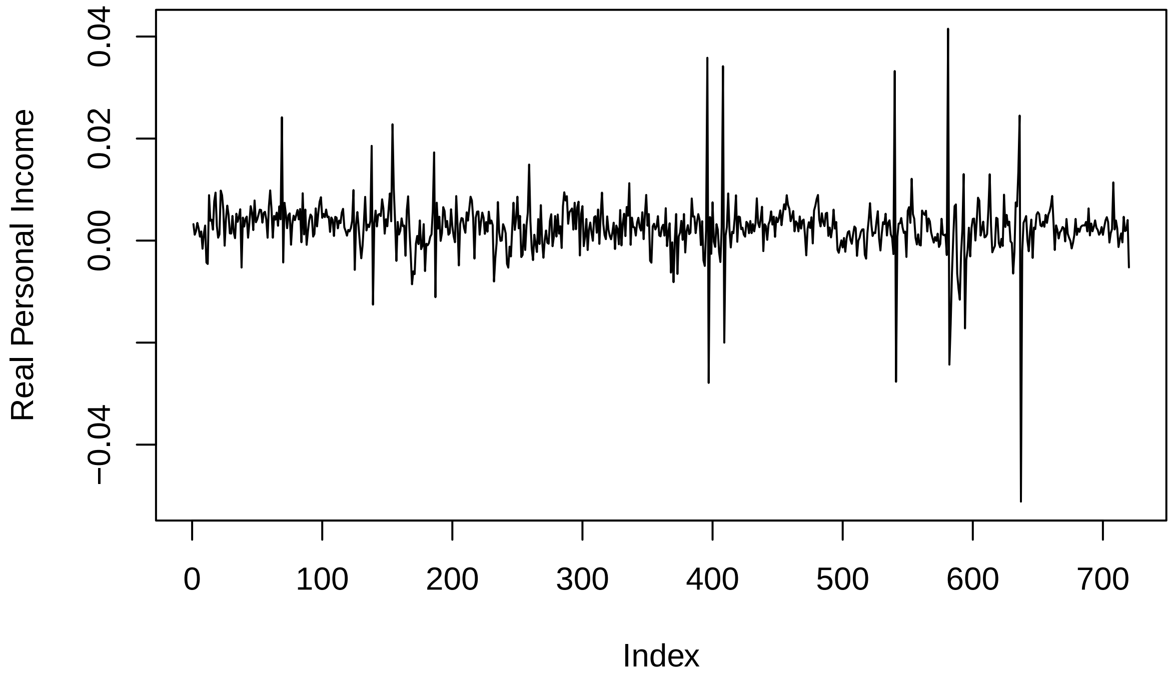
```
##
##   0
## 122
```

We plot the two time series of interest

```r
plot(x$DPCERA3M086SBEA,type = "l",ylab = var_desc$description[var_desc$fred == "DPCERA3M086SBEA"])
```

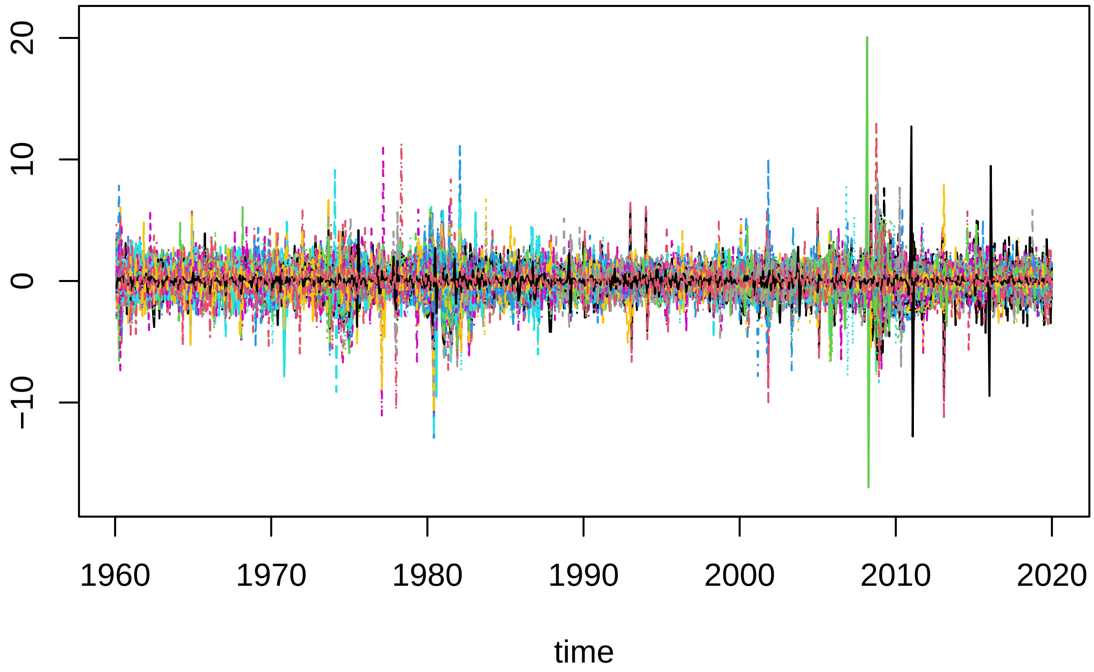


```
plot(x$RPI, type = "l",ylab = var_desc$description[var_desc$fred == "RPI"])
```



Let standardize the data

```
x_standard <- x

for (col in names(x)){
  mean <- mean(x[,col])
  st_d <- sd(x[,col])
  x_standard[,col] <- (x[,col] - mean)/ st_d
}

#Time variable
time <- c(1:nrow(x_standard))/12 + 1960

#Number of period
nb_T <- nrow(x_standard)

#Number of variable
nb_N <- ncol(x_standard)

#Plot the series
MTSplot(x_standard,time)
```



The standardized data looks similar as what we got in class

Lasso regression on real personnal consumption

function for lasso regression

```
Lasso_function <- function(p, r, alpha, variable, plot,residuals){
  if (p <= (r)){
    print("Must have p > r")
  }

  #outcome
  Y <- x_standard[(p+1):nb_T,which(names(x_standard)==variable)]
  #initialize endogenous variable
  X_st <- x_standard[p:(nb_T-1),which(names(x_standard)==variable)]
  if (p >= 2){
    for (i in seq(1,p-1)){
      Y_lag <- x_standard[seq(p-i,nb_T-1-i),which(names(x_standard)==variable)]
      X_st <- cbind(X_st,Y_lag)
    }
  }
  #exogenous varaibles and lag
  for (i in seq(0,r)){
    temp <- x_standard[(p-i):(nb_T-i-1),-which(names(x_standard)==variable)]
    X_st <- cbind(X_st,temp)
  }
  X_st <- as.matrix(X_st)


  #We use a cross validation to get the optimal lambda
  cv_fit <- cv.glmnet(X_st,Y,alpha=alpha)
  lambda <- cv_fit$lambda.min

  #To get the residuals
  if (residuals){
    resid <- Y - predict(cv_fit, newx = X_st)
    return(resid)
  }

  #We get the estimated lasso model
  lasso_fit <- glmnet(X_st,Y,alpha=alpha,lambda = lambda)
  #Plot if needed
  if (plot){
    plot(glmnet(X_st,Y,alpha=alpha))
    plot(cv_fit)
  }

  return(lasso_fit)
}
```

We can now extract the coefficients to try to forecast the series, we start by getting the coefficients and the corresponding lag

```
list_coeff <- function(Lasso_fit){
  coefficients <- Lasso_fit$beta

  #For Y ariable
  coef_Y <- coefficients[1:13]
  lag <- (1:13)
  coefficients_Y <- data.frame(coef_Y,lag)
  #Remove varaibles with null coeff
  coefficients_Y <- coefficients_Y[coefficients_Y$coef_Y !=0,]

  #For exogenous varaibles
  coef_X <- coefficients[seq(14,length(coefficients))]
  FRED <- row.names(Lasso_fit$beta)[seq(14,length(coefficients))]

  #We initialize the dataframe, there is 131 variables for each lag
  coefficients_X <- data.frame(coeff_X = coef_X[seq(1,131)])
  coefficients_X$lag <- 1
  coefficients_X$FRED <- FRED[seq(1,131)]
  #Remove varaibles with null coeff
  coefficients_X <- coefficients_X[coefficients_X$coeff_X !=0,]

  #For other lag
  for (i in (1:11)){
    coeff_X <- coef_X[seq(i*131+1,(i+1)*131)]
    temp <- data.frame(coeff_X)
    temp$lag <- i+1
    temp$FRED <- FRED[seq(i*131+1,(i+1)*131)]

    #Remove varaibles with null coeff
    temp <- temp[temp$coeff_X != 0,]

    #Add to dataframe
    coefficients_X <- rbind(coefficients_X, temp)
  }

  return (list(COEF_Y = coefficients_Y, COEF_X = coefficients_X))
}
```

Forecasting

Here we compute the standardized forecasts

```
forecasting <- function(Lasso_fit, variable, h, standard){

    #outcome
    Y <- x_standard[,which(names(x_standard)==variable)]
    #other variables
    X_st <- x_standard[,-which(names(x_standard)==variable)]

    list_coef <- list_coeff(Lasso_fit)

    #We initialize the list of forecast values
    forecasts <- c()
    #for each period
    for (i in seq(1,h)){
      value <- 0

      #for the endogenous variables, each lag
      for (k in seq(1,nrow(list_coef$COEF_Y))){
        lag <- as.numeric(list_coef$COEF_Y$lag[k])
        #coeff
        coeff <- as.numeric(list_coef$COEF_Y$coef_Y[k])

        #if we reuse a forecasted value
        if (i > lag){
          value <- value + forecasts[length(forecasts)-lag+1]*coeff

        }else{ #When we use value from the data set
          value <- value + Y[nb_T+ i -lag]*coeff

        }

      }
      #For endogenous variables
      for (k in seq(1,nrow(list_coef$COEF_X))){
        lag <-list_coef$COEF_X$lag[k]
        #coeff
        coeff <- as.numeric(list_coef$COEF_X$coeff_X[k])
        index <- which(names(X_st)== list_coef$COEF_X$FRED[k])

        #There is no forecasted value for exogenous, we only use the value it is in the dataset
        if (i <= lag){
          value <- value + X_st[nb_T+ i -lag, index]*coeff
        }
      }
      forecasts <- c(forecasts,value)
    }
  #If we want standardized data
  if (standard) {
    return(forecasts)
  }else{
    my <- mean(x[,which(names(x)==variable)])
    sy <- sd(x[,which(names(x)==variable)])
    return(forecasts*sy + my)
  }

}
```

Forecasting error using naive forecasting

```
forecasting_error <- function(Lasso_fit, variable, h,sigma){
  Forecasts <- forecasting(Lasso_fit, variable, h, standard = FALSE)

  min <- c()
  max <- c()
  for (i in seq(1,h)){
    error <- sd(x[,which(names(x)==variable)])*sigma*(i)^(1/2)
    min <- c(min,Forecasts[1] - error)
    max <- c(max,Forecasts[1] + error)
  }

  return(data.frame(Forecasts,min,max))
}
```

Plot the forecasted values

```
library(ggplot2)
plot_forecast <- function(forecasts_data, variable, start){

  Forecasts <- x[,which(names(x)==variable)]

  #forecast horizon
  h <- nrow(forecasts_data)

  #time variable
  time <- c(1:(length(Forecasts)+h))/12 + 1960

  #DAta to plot
  data_plot <- data.frame(Forecasts)
  data_plot$min <- NA
  data_plot$max <- NA
  #Adding forecast
  data_plot <- rbind(data_plot,forecasts_data)

  data_plot$time <- time

  data_plot <- data_plot[data_plot$time >= start,]

  p <- ggplot(data=data_plot, aes(time, Forecasts)) +
    geom_ribbon(aes(ymin = min, ymax = max), fill = "steelblue2") +
    geom_line(color = "firebrick", size = 0.1)

  return(p)

}
```

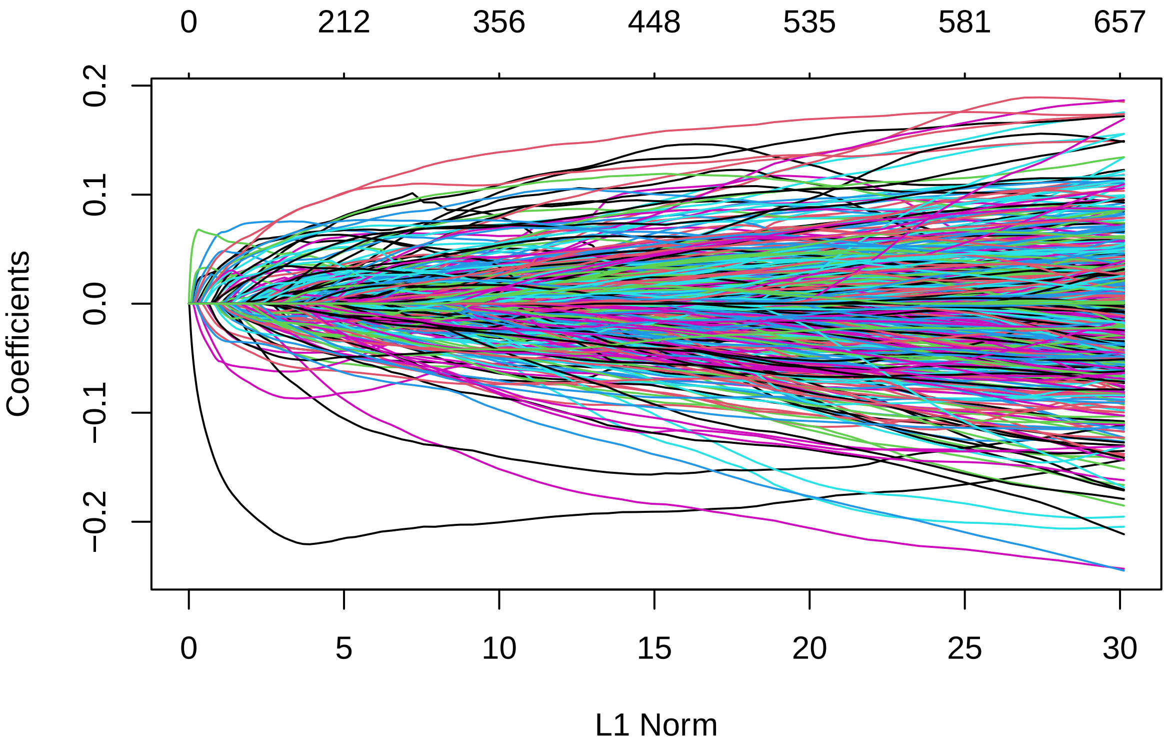
Application to variables

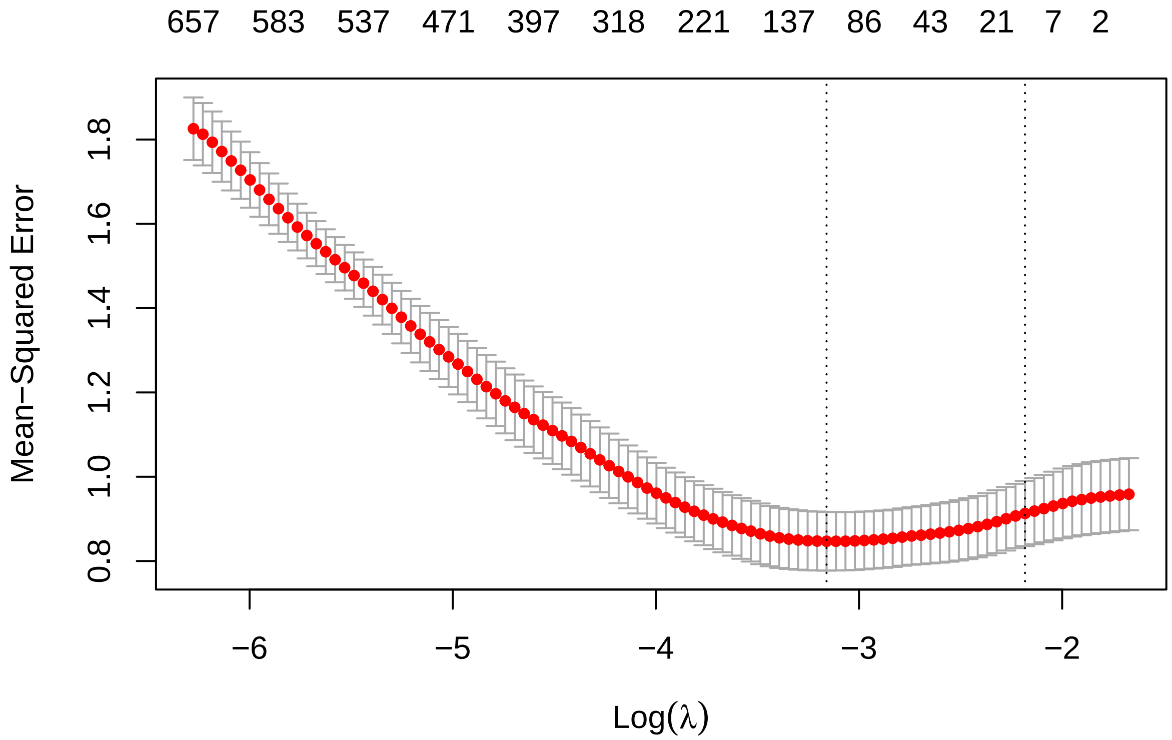
Real personal consumption

Now we try to find the best Lasso-VAR model to fit real personal consumption.

We take p and r large, useless variables will be removed with the lasso.

```
p = 13
r =12
alpha = 1
variable = "DPCERA3M086SBEA"


Lasso_fit <- Lasso_function(p = p, r = r, alpha = alpha, variable = variable, plot = TRUE, residuals = 
```


0
212
356
448
535
581
657
0.2
0.1
0.0
−0.1
−0.2
Coefficients
0
5
10
15
20
25
30
L1 Norm



```
print(Lasso_fit$lambda)
```

```
## [1] 0.04243689
```

```
print((log(nb_N-1)/nb_T)^(1/2))
```

```
## [1] 0.08161385
```

```
print(Lasso_fit$dev.ratio)
```

```
## [1] 0.3972482
```

We extract the residuals to compute the forecasting error

```
residual <- Lasso_function(p = p, r = r, alpha = alpha, variable = variable, plot = FALSE, residuals =
sigma <- sd(residual)
```

Number of variables kept, for the outcome

```
coefficients_Y <- Lasso_fit$beta[1:13]
length(coefficients_Y[coefficients_Y !=0])
```

```
## [1] 3
```

Number of variables kept, for the exogenous variables

```
coefficients_X <- Lasso_fit$beta[seq(14,length(Lasso_fit$beta))]
length(coefficients_X[coefficients_X !=0])
```

```
## [1] 98
```

We get the varaibles with higher coefficient in absolute value (X_st correspond to Y_lag of order 1)

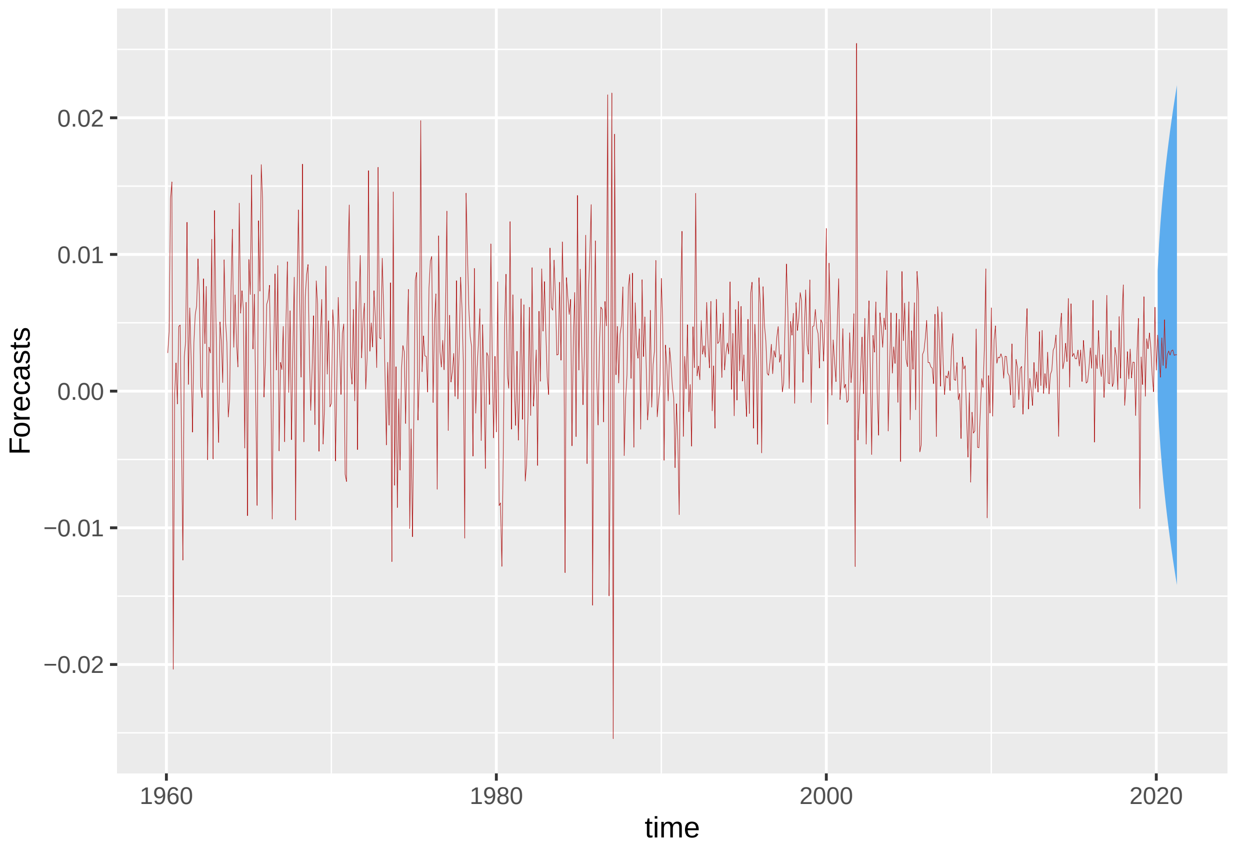
```
VAR_names <- row.names(Lasso_fit$beta)
coeffs <- Lasso_fit$beta[seq(1,length(Lasso_fit$beta))]
table_coefficients <- data.frame(VAR_names, coeffs)

lags <- c(1,2,3,4,5,6,7,8,9,10,11,12,13)
for (i in seq(1,13)){
  for (k in seq(1,121)){
    lags <- c(lags,i)
  }
}

table_coefficients$lags <- lags
table_coefficients$abs_coff <- abs(table_coefficients$coeffs)

table_coefficients <-table_coefficients[order(table_coefficients$abs_coff,decreasing = TRUE),]

head(table_coefficients)
```

```
##      VAR_names      coeffs lags   abs_coff
## 1         X_st -0.20597404    1 0.20597404
## 572   FEDFUNDS -0.08218649    5 0.08218649
## 76      M2REAL  0.07517290    1 0.07517290
## 869   IPDCONGD  0.07195246    8 0.07195246
## 1122 HWIURATIO  0.06957909   10 0.06957909
## 217        BAA -0.06143196    2 0.06143196
```

We plot the forecasted values

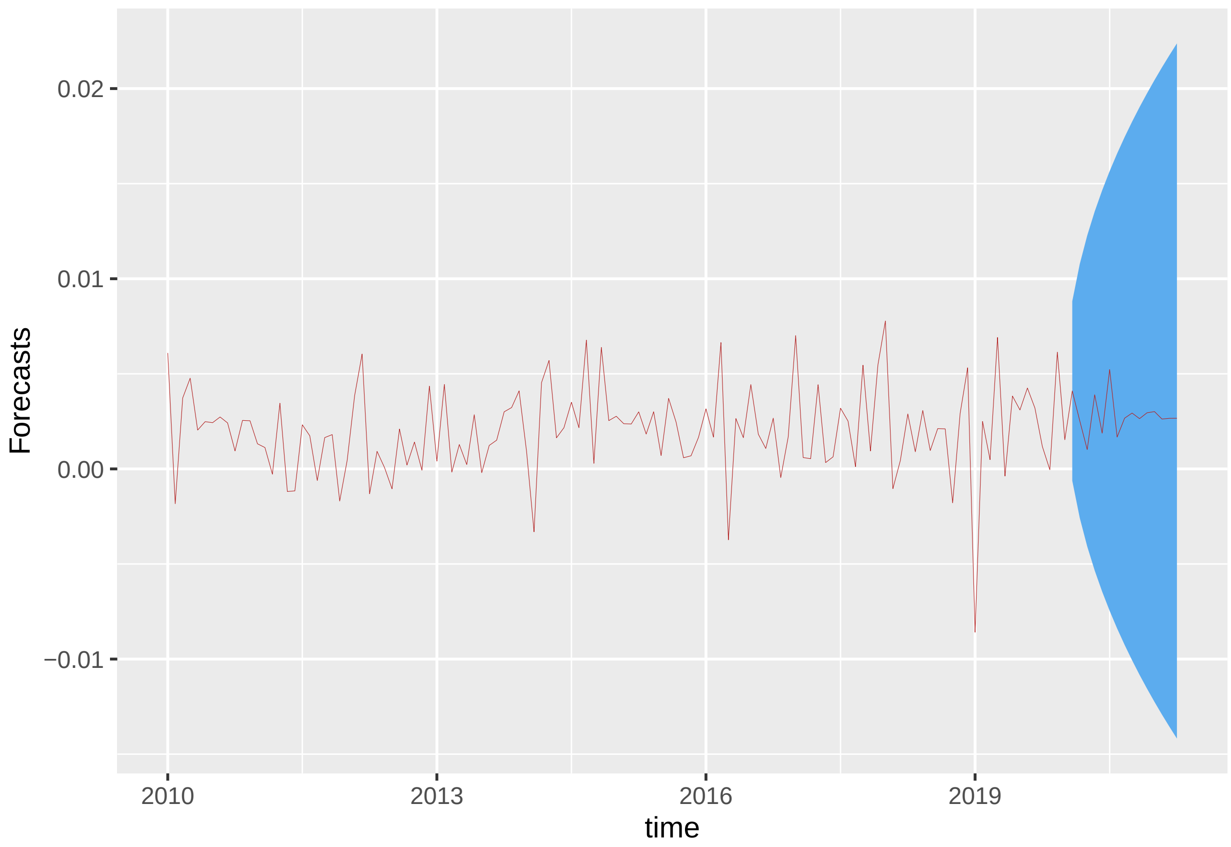
```
forecast_data <- forecasting_error(Lasso_fit, variable, h= 15,sigma)
plot_forecast(forecast_data, variable, start=1960)
```

```
## Warning: Using 'size' aesthetic for lines was deprecated in ggplot2 3.4.0.
## i Please use 'linewidth' instead.
## This warning is displayed once every 8 hours.
## Call 'lifecycle::last_lifecycle_warnings()' to see where this warning was
## generated.
```



We focus on a smaller period

```r
plot_forecast(forecast_data, variable, start=2010)
```

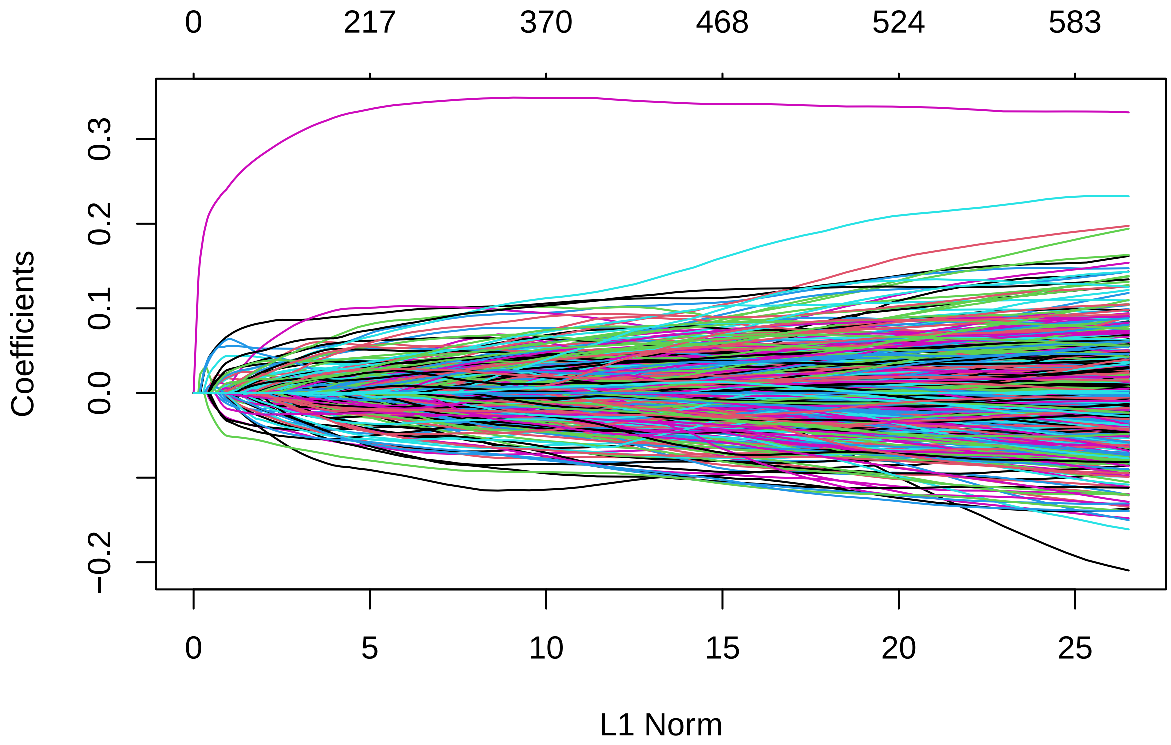


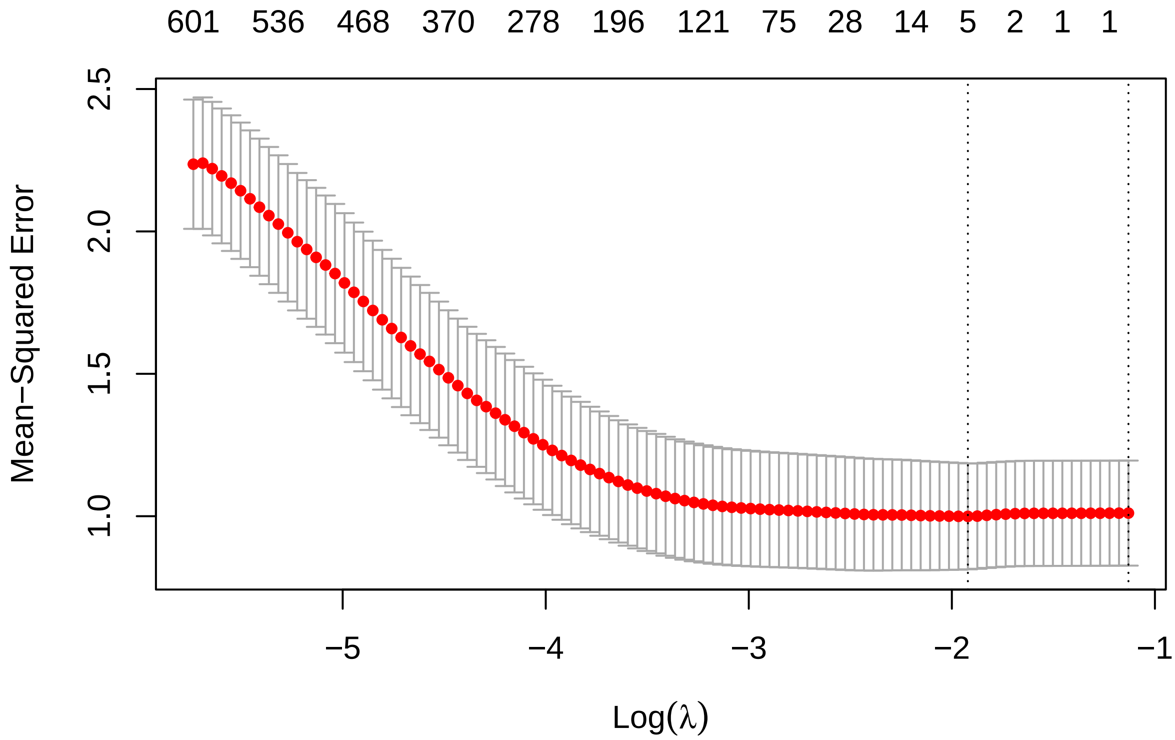
Real personal income

Now we try on real personal income

We take p and r large, useless variables will be removed with the lasso

```
p = 13
r =12
alpha = 1
variable = "RPI"


Lasso_fit <- Lasso_function(p = p, r = r, alpha = alpha, variable = variable, plot = TRUE, residuals =
```


0
217
370
468
524
583
Coefficients
0.3
0.2
0.1
0.0
−0.2
0
5
10
15
20
25
L1 Norm



```
print(Lasso_fit$lambda)
```

```
## [1] 0.1464438
```

```
print((log(nb_N-1)/nb_T)^(1/2))
```

```
## [1] 0.08161385
```

```
print(Lasso_fit$dev.ratio)
```

```
## [1] 0.1038434
```

We extract the residuals to compute the forecasting error

```
residual <- Lasso_function(p = p, r = r, alpha = alpha, variable = variable, plot = FALSE, residuals = 
sigma <- sd(residual)
```

Number of variables kept, for the outcome

```
coefficients_Y <- Lasso_fit$beta[1:13]
length(coefficients_Y[coefficients_Y !=0])
```

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

Number of variables kept, for the exogenous variables

```
coefficients_X <- Lasso_fit$beta[seq(14,length(Lasso_fit$beta))]
length(coefficients_X[coefficients_X !=0])
```

```
## [1] 4
```

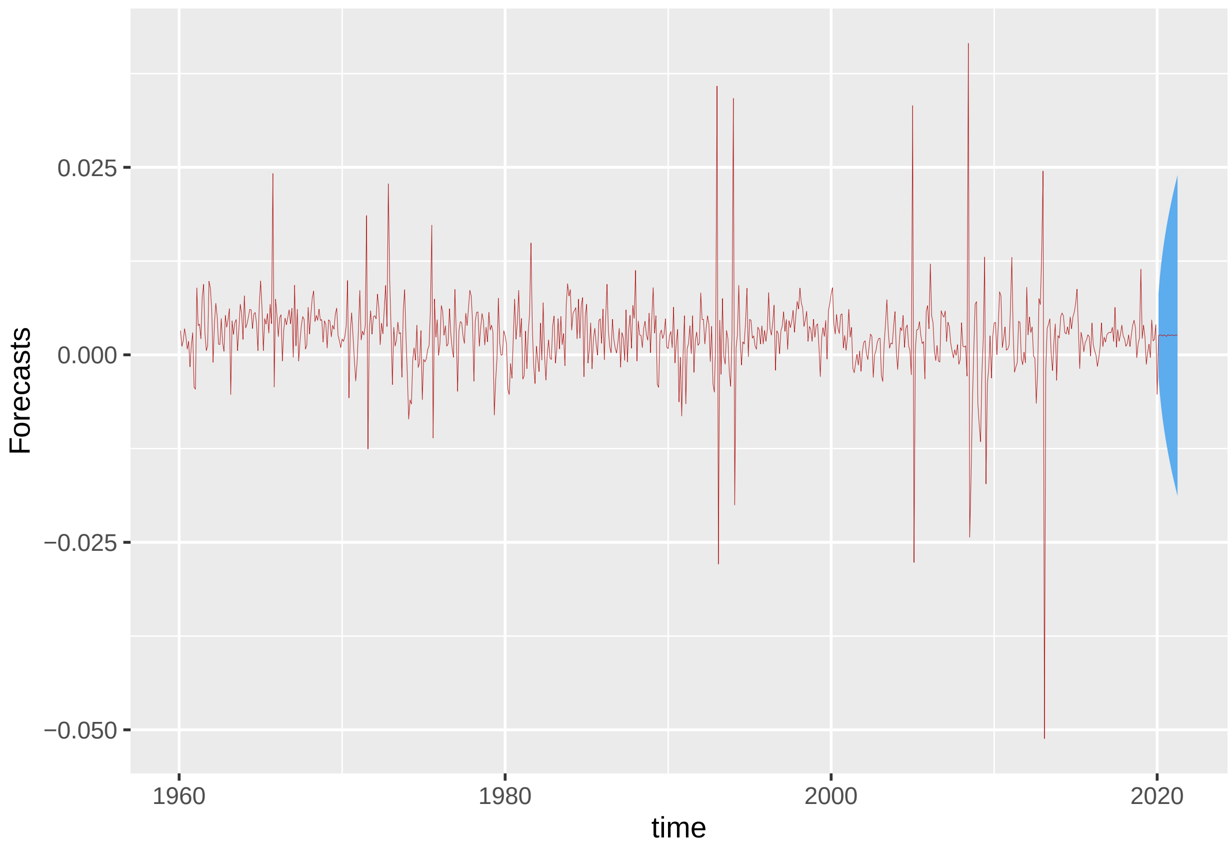
We get the varaibles with higher coefficient in absolute value (X_st correspond to Y_lag of order 1)

```
VAR_names <- row.names(Lasso_fit$beta)
coeffs <- Lasso_fit$beta[seq(1,length(Lasso_fit$beta))]
table_coefficients <- data.frame(VAR_names, coeffs)

lags <- c(1,2,3,4,5,6,7,8,9,10,11,12,13)
for (i in seq(1,13)){
  for (k in seq(1,121)){
    lags <- c(lags,i)
  }
}

table_coefficients$lags <- lags
table_coefficients$abs_coff <- abs(table_coefficients$coeffs)

table_coefficients <-table_coefficients[order(table_coefficients$abs_coff,decreasing = TRUE),]

head(table_coefficients)
```

```
##      VAR_names     coeffs lags   abs_coff
## 321  NONBORRES 0.17551450    3 0.17551450
## 48      MANEMP 0.02610497    1 0.02610497
## 1107    INDPRO 0.01546044   10 0.01546044
## 50    NDMANEMP 0.01420956    1 0.01420956
## 12       Y_lag 0.01151505   12 0.01151505
## 1         X_st 0.00000000    1 0.00000000
```

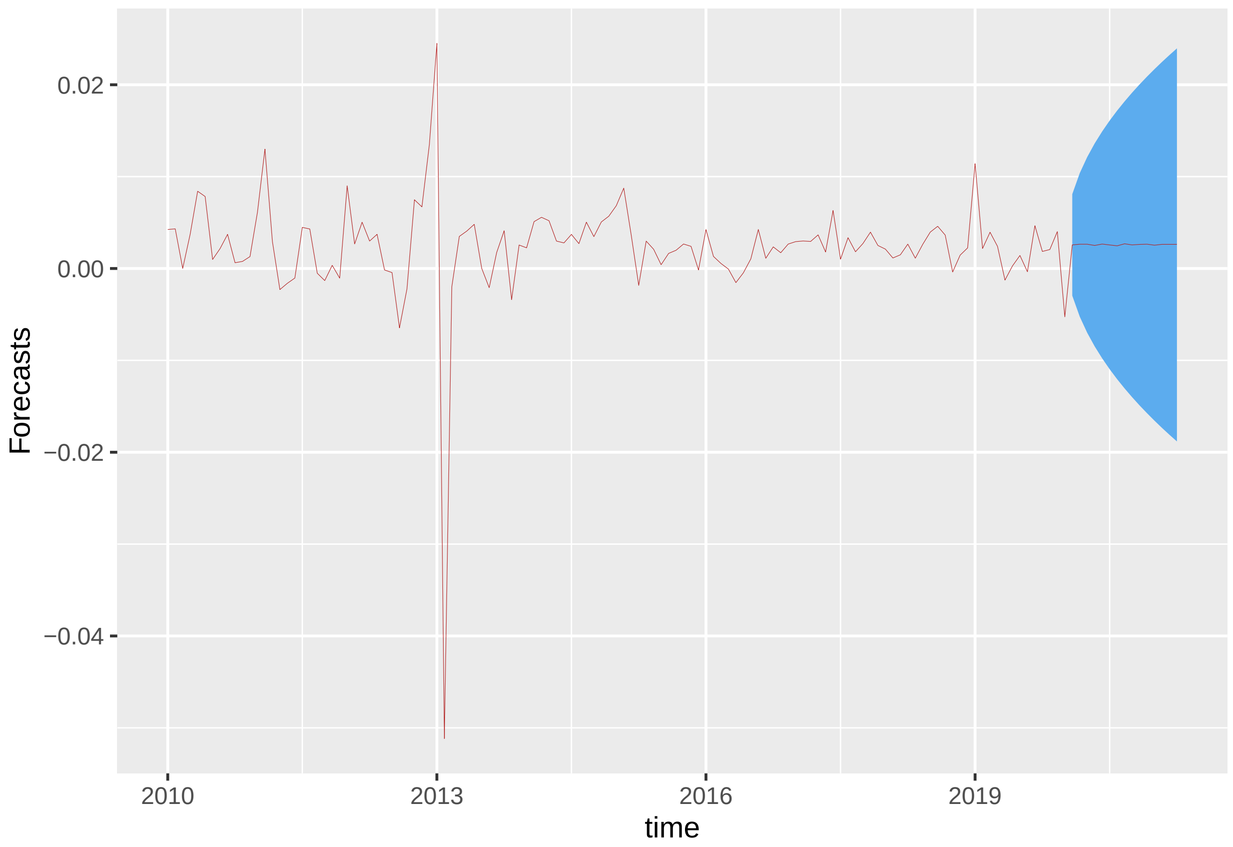
We plot the forecasted values

```
forecast_data <- forecasting_error(Lasso_fit, variable, h= 15,sigma)
plot_forecast(forecast_data, variable, start=1960)
```



We focus on a smaller period

```
plot_forecast(forecast_data, variable, start=2010)
```


0.02
0.00
−0.02
−0.04
Forecasts
2010
2013
2016
2019
time