Question 02 / A

Initially, we will load the dataframe with the data, perform the transformations indicated for each series, and create the inflation variable from the CPIAUCSL series.

```
# Loading the data

library(readr)
data <- read_csv("G:/Meu Drive/PUC - Rio/3º Semestre/Econometria IV/Assignment/2021-12.csv")
head(data)
```

```
## # A tibble: 6 x 128
##   sasdate   RPI W875RX1 DPCERA3M086SBEA CMRMTSPLx RETAILx INDPRO IPFPNSS IPFINAL
##   <chr>   <dbl>   <dbl>           <dbl>     <dbl>   <dbl>  <dbl>   <dbl>   <dbl>
## 1 Transf~     5       5               5         5       5      5       5       5
## 2 1/1/19~ 2442.   2293.            17.3   292266.  18236.   22.1    23.4    22.3
## 3 2/1/19~ 2452.   2302.            17.5   294425.  18370.   22.5    23.7    22.5
## 4 3/1/19~ 2468.   2318.            17.6   293419.  18523.   22.8    23.9    22.6
## 5 4/1/19~ 2484.   2335.            17.6   299323.  18534.   23.3    24.2    22.9
## 6 5/1/19~ 2498.   2350.            17.8   301364.  18680.   23.7    24.4    23.1
## # i 119 more variables: IPCONGD <dbl>, IPDCONGD <dbl>, IPNCONGD <dbl>,
## #   IPBUSEQ <dbl>, IPMAT <dbl>, IPDMAT <dbl>, IPNMAT <dbl>, IPMANSICS <dbl>,
## #   IPB51222S <dbl>, IPFUELS <dbl>, CUMFNS <dbl>, HWI <dbl>, HWIURATIO <dbl>,
## #   CLF16OV <dbl>, CE16OV <dbl>, UNRATE <dbl>, UEMPMEAN <dbl>, UEMPLT5 <dbl>,
## #   UEMP5TO14 <dbl>, UEMP15OV <dbl>, UEMP15T26 <dbl>, UEMP27OV <dbl>,
## #   CLAIMSx <dbl>, PAYEMS <dbl>, USGOOD <dbl>, CES1021000001 <dbl>,
## #   USCONS <dbl>, MANEMP <dbl>, DMANEMP <dbl>, NDMANEMP <dbl>, ...
```

```
# Considering only variables with all observations in the sample

library(dplyr)

new_data <- data %>%
  select_if(~ !any(is.na(.)))

# Removing the dependent variable (CPIAUCSL) from the dataframe and reserving it
# in a separate separate dataframe:

df_cpiaucsl <- data.frame(CPIAUCSL = new_data$CPIAUCSL)

# We remove the dependent variable from the dataframe because it receives a
# distinct transformation from the one indicated by the first line

# Loading the package to transform the variable panel

library(fbi)

# Making the transformations of the other series in the panel

data_fred <- fredmd(file = 'G:/Meu Drive/PUC - Rio/3º Semestre/Econometria IV/Assignment/2021-12.csv' ,
                    date_start = NULL, date_end = NULL, transform = TRUE)

# Now, let's construct the inflation series from CPIAUCSL
```

```r
library(dplyr)
library(tidyverse)

df_cpiaucsl <- slice(df_cpiaucsl, -1)  # Excluding the first row

df_cpiaucsl <- df_cpiaucsl %>%
  mutate(inflation = (CPIAUCSL - lag(CPIAUCSL))/(CPIAUCSL))

# Bringing the series together in one dataframe

data_fred <- cbind(data_fred, df_cpiaucsl)

## We need drop out all the variables with NA values
# Excluding the first two rows of data_fred (some variables has NA values due to
# transformation):

data_fred <- data_fred[-c(1:2), ]

# Identifying variables with NA values

vars_with_na <- colnames(data_fred)[colSums(is.na(data_fred)) > 0]

# Excluding these variables from dataframe data_fred

data_fred <- data_fred[, setdiff(colnames(data_fred), vars_with_na)]
```

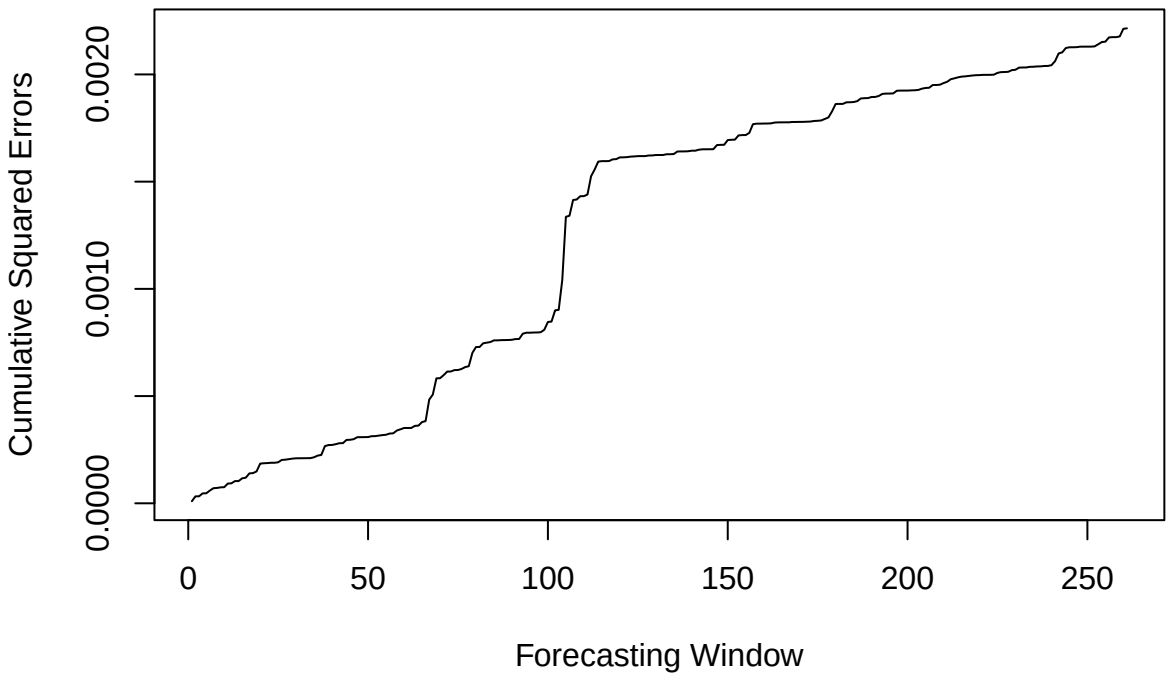
The dataframe data_fred is ready. Now, we move on to the rolling window framework estimations for each of the models.

1) Autoregressive Model

The order of the autoregressive model is defined by the BIC criterion in each rolling window.

```r
# Load required packages
library(stats)

# Set the length of the moving window
window_length <- 492

# Create a vector to store the squared one-step-ahead forecast errors and forecasts
forecast_errors <- numeric(nrow(data_fred) - window_length)
forecasts <- numeric(nrow(data_fred) - window_length)

# Set the initial position of the moving window
x <- 1

# Iterate over the observations
while ((x + window_length) <= nrow(data_fred)) {
  # Subset the data within the moving window
  window_data <- data_fred[x:(x + window_length - 1), ]

  # Determine the order p of the AR model using BIC criterion
```

```r
  max_p <- min(10, ncol(window_data) - 1)  # Set a maximum order of 10 or less

  bic_values <- vector("numeric", max_p)

  for (p in 1:max_p) {
    model <- arima(window_data$inflation, order = c(p, 0, 0), method = "CSS-ML")
    bic_values[p] <- BIC(model)
  }

  best_order <- which.min(bic_values)

  # Fit the AR model with the best order using OLS
  ar_model <- arima(window_data$inflation, order = c(best_order, 0, 0), method = "CSS-ML")

  # Compute the forecast for the next observation
  next_observation <- data_fred[x + window_length, "inflation"]
  forecast <- predict(ar_model, n.ahead = 1)$pred[1]

  # Store the forecast
  forecasts[x] <- forecast

  # Compute the squared one-step-ahead forecast error
  forecast_error <- (forecast - next_observation)^2
  forecast_errors[x] <- forecast_error

  # Move the window forward
  x <- x + 1
}

# The forecasts, as well as the squared forecast errors, start in March 2000;
# and run until November 2021

# Compute the cumulative squared errors over the forecasting window
cumulative_errors <- cumsum(forecast_errors)

# Plot the cumulative squared errors
plot(cumulative_errors, type = "l", xlab = "Forecasting Window",
     ylab = "Cumulative Squared Errors")
```



2) AR + Principal Component Regression (PCR)

In this model, we combine an AR(2) part with a PCR part. The factors are estimated from Principal Component Analysis of all variables except the two inflation lags. In each rolling window, the number of factors selected is the minimum number of principal components that explain up to 90% of the variance. Then, the estimated factors are pooled with the two inflation lags and the model is estimated.

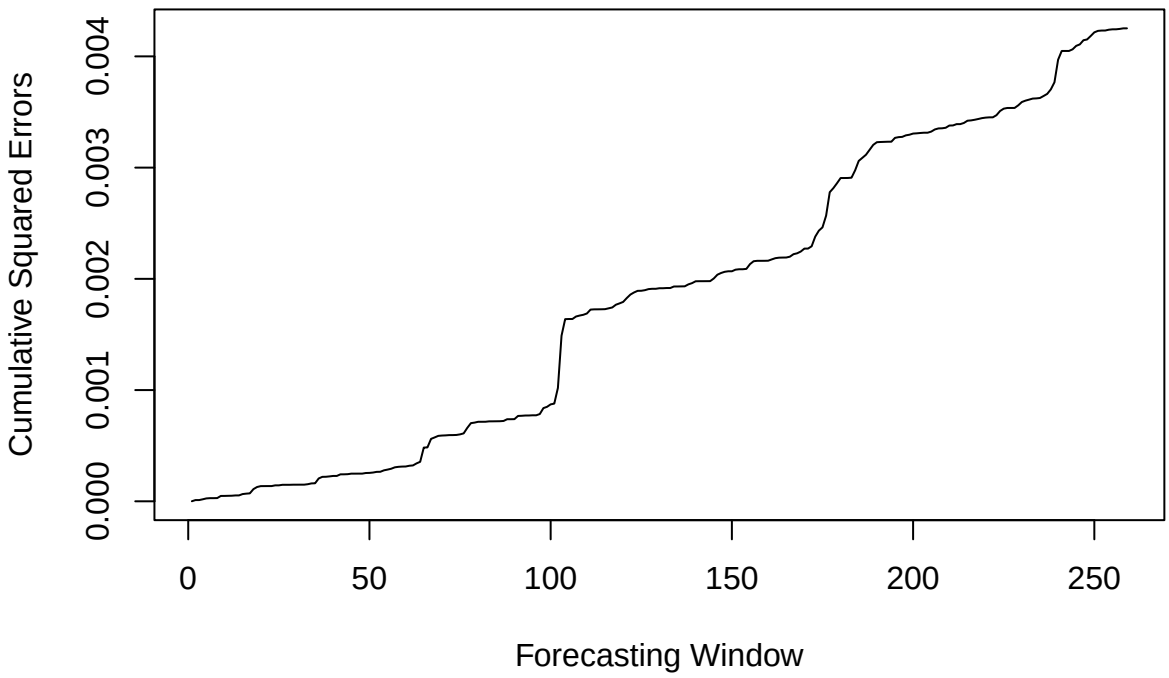
We then construct an additional dataframe, consisting of the data_fred plus two inflation lags.

```
# We consider two lags of inflation to include in the AR part of the model
# Thus, let's construct a new dataframe from data_fred:

data_fred_laginf <- data_fred %>%
  mutate(inflation_lag1 = lag(inflation, 1),
         inflation_lag2 = lag(inflation, 2))

# We have to exclude the two first rows (NA observations):

data_fred_laginf <- data_fred_laginf %>%
  slice(-(1:2))

# Now, we can estimate the model in a rolling window framework
```

We can now proceed with the estimation and forecasting of the AR + PCR model.

```r
# Load required libraries
library(pls)

# Set the length of the moving window
window_length <- 492

# Initialize vectors to store forecasts and squared errors
forecasts_pcr <- numeric(nrow(data_fred_laginf) - window_length)
squared_errors_pcr <- numeric(nrow(data_fred_laginf) - window_length)

# Set the initial position of the moving window
a <- 1

# Loop over the observations
while ((a + window_length) <= nrow(data_fred_laginf)) {
  # Define the current window
  window <- data_fred_laginf[a:(a + window_length - 1), ]

  # Split the window into training matrix X and response vector y
  y <- window$inflation
  X <- window[ ,!colnames(window) %in% c("inflation", "inflation_lag1",
                                         "inflation_lag2", "date")]

  # Compute the Principal Components and select the number of factors
  pca <- prcomp(X)
  cumulative_variance <- cumsum(pca$sdev^2) / sum(pca$sdev^2)
  num_factors <- which.max(cumulative_variance >= 0.9) # Number of factors
  # explaining 90% of variance

  # Compute the principal components
  principal_components <- pca$x[ , 1:num_factors]

  # Prepare the lagged inflation variable
  lagged_inflation <- window[ , c("inflation_lag1", "inflation_lag2")]

  # Combine lagged inflation and principal components as predictors
  predictors <- cbind(lagged_inflation, principal_components)

  # Perform Principal Component Regression
  pcr_model <- pcr(y ~ ., data = data.frame(predictors), scale = TRUE, validation = "CV")

  # To perform prediction, we need the next predictors

  next_window <- data_fred_laginf[(a + 1):(a + window_length), ]
  t <- next_window$inflation
  Z <- next_window[ ,!colnames(window) %in% c("inflation", "inflation_lag1",
                                              "inflation_lag2", "date")]

  next_pca <- prcomp(Z)
  next_cumulative_variance <- cumsum(next_pca$sdev^2) / sum(next_pca$sdev^2)
  next_num_factors <- which.max(next_cumulative_variance >= 0.9) # Number of
  # factors explaining 90% of variance
```

```r
  next_principal_components <- next_pca$x[, 1:next_num_factors]
  next_lagged_inflation <- next_window[ , c("inflation_lag1", "inflation_lag2")]
  next_predictors <- cbind(next_lagged_inflation, next_principal_components)

  # With the next predictors, we can compute the forecasts

  forecast_pcr <- predict(pcr_model, newdata = data.frame(next_predictors))[[1]]

  # Store the forecast
  forecasts_pcr[a] <- forecast_pcr

  # Compute the squared one-step-ahead forecasting error
  squared_error_pcr <- (forecast_pcr - data_fred_laginf$inflation[a + window_length])^2

  # Store the squared error
  squared_errors_pcr[a] <- squared_error_pcr

  # Moving window forward
  a <- a + 1

}

# Due to the inclusion of two lags of the "inflation" variable, the forecasts
# (and the squared forecast errors) start in May 2000 and run until November 2021.

# Compute the cumulative squared errors over the forecasting window
cumulative_errors_pcr <- cumsum(squared_errors_pcr)

# Plot the cumulative squared errors
plot(cumulative_errors_pcr, type = "l", xlab = "Forecasting Window",
     ylab = "Cumulative Squared Errors")
```



In the complementary part of the AR model in the next two models, we will use two lags of each variable from the original dataframe (except for the CPIAUCSL variable). Therefore, we will create a dataframe (called data_fred_lagged) from the original dataframe to perform the estimations.

```
# For the two models below (Ridge and LASSO), we use a dataframe with two
# lags of each variable

library(tidyselect)

# Select the relevant variables to lag
lag_columns <- select(data_fred, -1, -CPIAUCSL) %>% names()

# Create a new dataframe with the lags of relevant variables
data_fred_lagged <- data_fred %>%
  mutate(across(all_of(lag_columns), list(lag1 = ~lag(.), lag2 = ~lag(., 2)), .names = "{col}_{.fn}"))

# Excluding the first 2 rows (NA observations)
data_fred_lagged <- slice(data_fred_lagged, -(1:2))

# Now, the dataframe is ready.
```

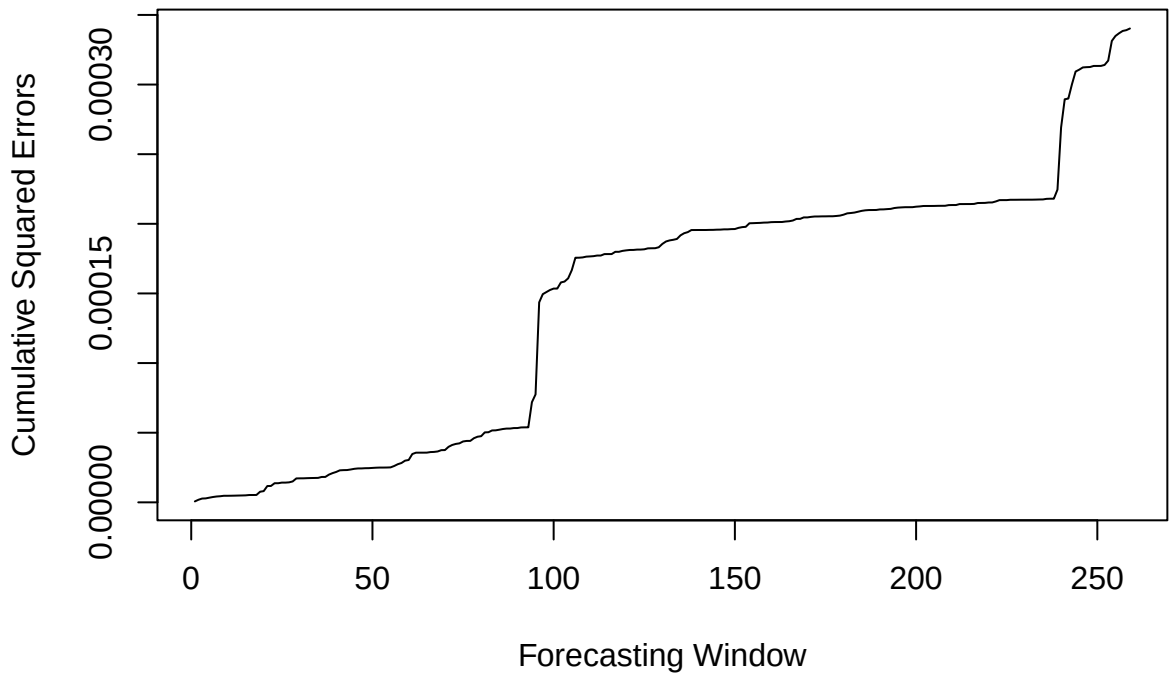
Let's define the ic.glmnet function (by Gabriel Vasconcelos, available at https://github.com/gabrielrvsc/HDeconometrics)

3) Ridge Regression

We now proceed to forecast inflation using a Ridge Regression model. The model features two lags of inflation and two lags of the other variables in the dataset, except CPIAUCSL. The model parameters are estimated with the penalty term selected by the BIC criterion.

```
# Load required libraries
library(glmnet)

# Set the window size
window_size <- 492

# Initialize vectors to store forecasts and squared errors
forecasts_ridge <- numeric(nrow(data_fred_lagged) - window_size)
squared_errors_ridge <- numeric(nrow(data_fred_lagged) - window_size)

# Set the initial position of the moving window
a <- 1

# Loop over the observations
for (a in 1:(nrow(data_fred_lagged) - window_size)) {
  # Subset the data for the current window
  window_data <- data_fred_lagged[a:(a + window_size - 1), ]

  # Extract the dependent variable
  y <- window_data$inflation

  # Extract the independent variables (X)
  X <- as.matrix(window_data[, !(names(window_data) %in% c("inflation",
                                                           "CPIAUCSL", "date"))])

  # Estimate Ridge Regression with BIC penalty term selection
  ridge_fit <- ic.glmnet(X, y, crit = 'bic', alpha = 0)

  # Defining the model
  ridge_model <- glmnet(X, y, alpha = 0)

  # Predict the inflation with Ridge model prediction
  forecast_ridge <- predict(ridge_model,
                            newx = as.matrix(data_fred_lagged[(a + window_size),
!(names(data_fred_lagged) %in% c("inflation", "CPIAUCSL","date"))]),
                            s = ridge_fit$lambda)

  # Store the forecast
  forecasts_ridge[a] <- forecast_ridge

  # Compute the squared one-step-ahead forecasting error
  squared_error_ridge <- (forecasts_ridge[a] - data_fred_lagged$inflation[a + window_size])^2

  # Store the squared error
  squared_errors_ridge[a] <- squared_error_ridge

}
```

```r
# Due to the inclusion of two lags of the "inflation" variable and the other variables
# (except "CPIAUCSL"), the forecasts (and the squared forecast errors) start in
# May 2000 and run until November 2021.

# Compute cumulative squared errors
cumulative_errors_ridge <- cumsum(squared_errors_ridge)

# Plot cumulative squared errors
plot(cumulative_errors_ridge, type = "l", xlab = "Forecasting Window",
     ylab = "Cumulative Squared Errors")
```

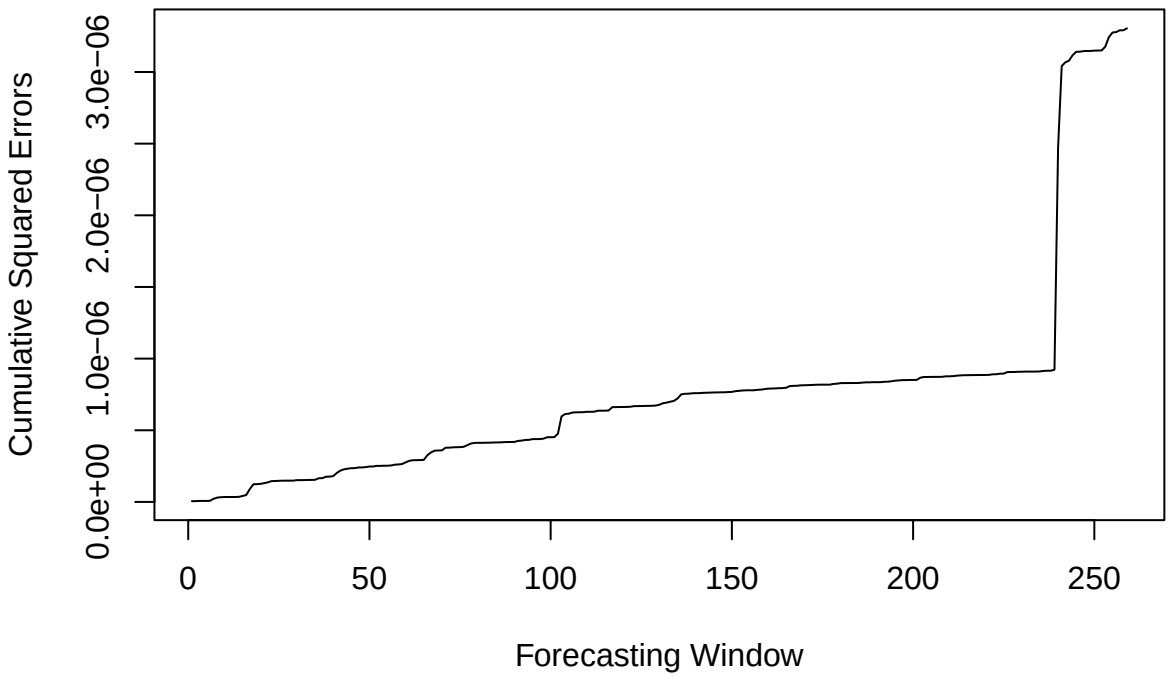


4) LASSO Regression

We proceed with inflation forecasting using a LASSO model. The model features two lags of inflation and two lags of the other dataset variables, except CPIAUCSL. The model parameters are estimated with a penalty term selected by the BIC criterion.

```r
# Our dataset is ready

# Load required libraries
library(glmnet)

# Set the window size
window_size <- 492
```

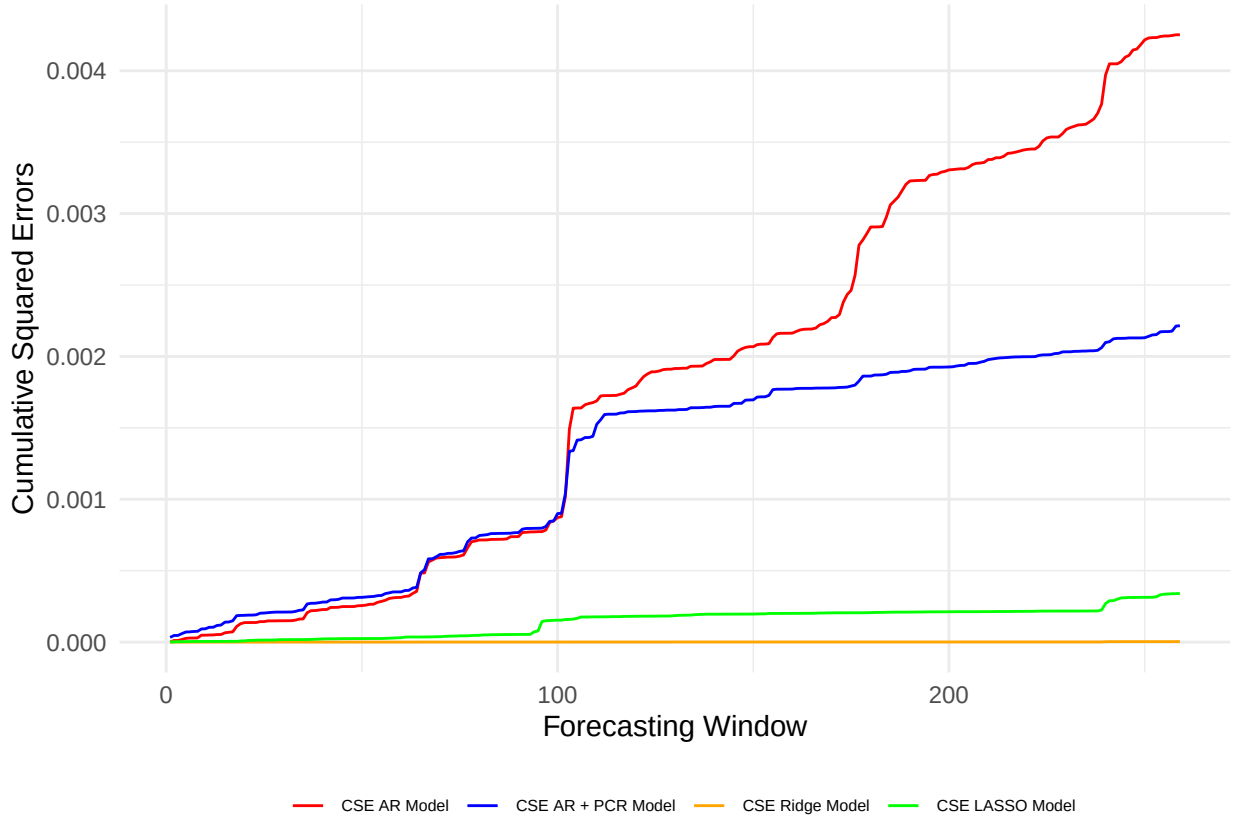
```r
# Initialize vectors to store forecasts and squared errors
forecasts_lasso <- numeric(nrow(data_fred_lagged) - window_size)
squared_errors_lasso <- numeric(nrow(data_fred_lagged) - window_size)

# Set the initial position of the moving window
a <- 1

# Loop over the observations
for (a in 1:(nrow(data_fred_lagged) - window_size)) {
  # Subset the data for the current window
  window_data <- data_fred_lagged[a:(a + window_size - 1), ]

  # Extract the dependent variable
  y <- window_data$inflation

  # Extract the independent variables (X)
  X <- as.matrix(window_data[, !(names(window_data) %in% c("inflation","CPIAUCSL",
                                                           "date"))])

  # Estimate LASSO Regression with BIC penalty term selection
  lasso_fit <- ic.glmnet(X, y, crit = 'bic', alpha = 1)

  # Defining the model
  lasso_model <- glmnet(X, y, alpha = 1)

  # Predict the inflation with LASSO model prediction
  forecast_lasso <- predict(lasso_model,
                            newx = as.matrix(data_fred_lagged[(a + window_size), !(names(data_fred_lagg
                            s = lasso_fit$lambda)

  # Store the forecast
  forecasts_lasso[a] <- forecast_lasso

  # Compute the squared one-step-ahead forecasting error
  squared_error_lasso <- (forecasts_lasso[a] - data_fred_lagged$inflation[a + window_size])^2

  # Store the squared error
  squared_errors_lasso[a] <- squared_error_lasso

}

# Due to the inclusion of two lags of the "inflation" variable and the other variables
# (except "CPIAUCSL"), the forecasts (and the squared forecast errors) start in
# May 2000 and run until November 2021.

# Compute cumulative squared errors
cumulative_errors_lasso <- cumsum(squared_errors_lasso)

# Plot cumulative squared errors
plot(cumulative_errors_lasso, type = "l", xlab = "Forecasting Window",
     ylab = "Cumulative Squared Errors")
```



Let's plot a graph to better inspect the cumulative square errors of each of the four forecasting models.

```
# Plotting all cumulative squared errors

library(ggplot2)

# We will exclude the first two cumulative squared error entries for the AR model,
# which correspond to March and April 1959; thus, the four cumulative squared
# error vectors will have the same number of entries, all starting in May 1959.

# Creating vectors
cse_ar <- c(cumulative_errors[3:length(cumulative_errors)])
cse_ar_pcr <- c(cumulative_errors_pcr)
cse_ridge <- c(cumulative_errors_ridge)
cse_lasso <- c(cumulative_errors_lasso)

# Creating a dataframe with cumulative square errors

df_cse <- data.frame(
  x = 1:length(cse_ar),
  y = c(cse_ar, cse_ar_pcr, cse_ridge, cse_lasso),
  vetor = rep(c("CSE AR Model", "CSE AR + PCR Model", "CSE Ridge Model",
                "CSE LASSO Model"), times = c(length(cse_ar), length(cse_ar_pcr),
                                              length(cse_ridge), length(cse_lasso)))
)
```

```r
# Plot with ggplot2
ggplot(df_cse, aes(x = x, y = y, color = vetor)) +
  geom_line(na.rm = TRUE) +
  labs(x = "Forecasting Window", y = "Cumulative Squared Errors", color = NULL) +
  scale_color_manual(values = c("CSE AR Model" = "blue", "CSE AR + PCR Model" = "red",
                                "CSE Ridge Model" = "green", "CSE LASSO Model" = "orange"),
                     labels = c("CSE AR Model", "CSE AR + PCR Model", "CSE Ridge Model",
                                "CSE LASSO Model")) +
  theme_minimal() +
  theme(legend.position = "bottom",
        legend.key.size = unit(0.5, "cm"),
        legend.key.width = unit(0.5, "cm"),
        legend.text = element_text(size = 6))
```



We found that the Ridge Model has the lowest cumulative square errors over the windows and is therefore the winning model in this exercise. The LASSO model performs close to the Ridge model. The AR and AR + PCR models are less effective, with the AR model being the worst in terms of performance.

It is interesting to note that the AR model and the AR + PCR model performed quite similarly until approximately window number 100, which coincides with the period of the Global Financial Crisis of 2008. From then on, the AR model performs persistently worse.

The AR model shows an increase in cumulative square errors from the occurrence of the COVID 19 pandemic. The AR + PCR and LASSO models also show worsening performance during the pandemic period, but of lesser magnitude. The Ridge model has practically no worsening of performance over the windows.

QUESTION 02 / B

This question consists of measuring the importance of each variable in three of the above models (AR + PCR, Ridge and LASSO). Initially, we will work with the Ridge and LASSO models. We set up a dataframe to store the importance measures of all the variables of these models in each of the rolling windows.

Let's start with the Ridge model.

```
# Let's create a dataframe to store the importance measure of each variable
# for each window in the Ridge and LASSO models

# First, Ridge model

number_windows <- nrow(data_fred_lagged) - window_size + 1

variables_importance_ridge <- data.frame(matrix(ncol = ncol(data_fred_lagged) - 1,
                                        nrow = number_windows))

colnames(variables_importance_ridge) <- paste(colnames(data_fred_lagged[, -1]),
                                       "_importance", sep = "")

variables_importance_ridge <- subset(variables_importance_ridge,
                              select = -c(inflation_importance, CPIAUCSL_importance))

# Now, we have a dataframe to store the results for each window
```

We now estimate the Ridge model, storing in the created dataframe the importance measures of each variable in each rolling window.

```
# Ridge Regression:

# Load required libraries
library(glmnet)

# Set the window size
window_size <- 492

# Loop over the observations
for (a in 1:(nrow(data_fred_lagged) - window_size + 1)) {
  # Subset the data for the current window
  window_data <- data_fred_lagged[a:(a + window_size - 1), ]

  # Extract the dependent variable
  y <- window_data$inflation

  # Extract the independent variables (X)
  X <- as.matrix(window_data[, !(names(window_data) %in% c("inflation","CPIAUCSL", "date"))])

  # Estimate Ridge Regression with BIC penalty term selection
  ridge_fit <- ic.glmnet(X, y, crit = 'bic', alpha = 0)

  # Defining the model
  ridge_model <- glmnet(X, y, alpha = 0)
```

```
  # Computing the variable importance
  coef <- coef(ridge_model, s = ridge_fit$lambda)[-1] # Excluding the intercept
  var_sd <- apply(X, 2, sd) # Computing the standard deviation for each variable
  variables_importance_ridge[a, ] <- coef * var_sd

}
```

We now proceed in the same way for the LASSO model. We create a dataframe to store the importance measures of each variable. We then estimate the model by filling the dataframe with the importance measure considered in the statement.

```
# Now, LASSO model

# Creating a dataframe to store the measures of importance for each variable

number_windows <- nrow(data_fred_lagged) - window_length + 1

variables_importance_lasso <- data.frame(matrix(ncol = ncol(data_fred_lagged) - 1,
                                                nrow = number_windows))

colnames(variables_importance_lasso) <- paste(colnames(data_fred_lagged[, -1]),
                                              "_importance", sep = "")

variables_importance_lasso <- subset(variables_importance_lasso,
                                     select = -c(inflation_importance, CPIAUCSL_importance))

# Now, we have a dataframe to store the results for each window
```

```
# LASSO Regression:

# Load required libraries
library(glmnet)

# Set the window size
window_size <- 492

# Loop over the observations
for (a in 1:(nrow(data_fred_lagged) - window_size + 1)) {
  # Subset the data for the current window
  window_data <- data_fred_lagged[a:(a + window_size - 1), ]

  # Extract the dependent variable
  y <- window_data$inflation

  # Extract the independent variables (X)
  X <- as.matrix(window_data[, !(names(window_data) %in% c("inflation","CPIAUCSL", "date"))])

  # Estimate Ridge Regression with BIC penalty term selection
  lasso_fit <- ic.glmnet(X, y, crit = 'bic', alpha = 1)

  # Defining the model
  lasso_model <- glmnet(X, y, alpha = 1)
```

```
  # Computing the variable importance
  coef <- coef(lasso_model, s = lasso_fit$lambda)[-1] # Excluding the intercept
  var_sd <- apply(X, 2, sd) # Computing the standard deviation for each variable
  variables_importance_lasso[a, ] <- coef * var_sd

}

# We have computed the importance of each variable for each forecasting window
```

Now, the problem asks us to aggregate the measures of importance of the variables in each rolling window into groups. The correspondence between each variable and its respective group is obtained from the 'fbi' package. Three observations are important at this point of the problem.

First, we need to add a new group, named 'lag', to group the two lags of inflation. Second, we find that the variable 'IPB51222S' is written differently in the table that performs the matching (in it, the s is lowercase), so we need to change the name of this variable in that table. Finally, we note that the variable 'BOGMBASE' is present in the templates and is not included in the description in the 'fbi' package. We then need to consult the FRED St. Louis homepage to conclude that this variable should be included in the 'Money and Credit' group.

```
# Now, we will aggregate the importance of each variable into its respective
# group

# Let's associate each variable with its group. This information is available
# in the package 'fbi':

library(fbi)

variables_description <- fredmd_description

View(variables_description)


# Creating a dataframe to associate each variable with its group

variables_group_assoc <- data.frame(matrix(ncol = 1, nrow = 3*nrow(variables_description)))

variables <- variables_description$fred
variables <- paste0(variables, "_importance")
variables_lag1 <- paste0(variables_description$fred, "_lag1_importance")
variables_lag2 <- paste0(variables_description$fred, "_lag2_importance")
variables_group_assoc$variables <- rep(c(variables, variables_lag1, variables_lag2))

variables_group_assoc$group <- rep(variables_description$group, times = 3)
variables_group_assoc <- variables_group_assoc[ , -1]

# Adding the 'inflation' variable and its lags

inflationlag1 <- data.frame(variables = "inflation_lag1_importance", group = "lag")
inflationlag2 <- data.frame(variables = "inflation_lag2_importance", group = "lag")

variables_group_assoc <- rbind(variables_group_assoc, inflationlag1, inflationlag2)

# We need to rename one variable:
```

```
variables_group_assoc$variables <- sub("IPB51222s_importance", "IPB51222S_importance",
                                       variables_group_assoc$variables)
variables_group_assoc$variables <- sub("IPB51222s_lag1_importance", "IPB51222S_lag1_importance",
                                       variables_group_assoc$variables)
variables_group_assoc$variables <- sub("IPB51222s_lag2_importance", "IPB51222S_lag2_importance",
                                       variables_group_assoc$variables)

match_columns <- variables_importance_ridge[, names(variables_importance_ridge) %in%
                                              variables_group_assoc$variables]

# We see that the variable "BOGMBASE" is NOT included in the variables_description,
# but is present in our models. Let's include it in our variables description. In
# the St. Louis FRED, we verify that the variable belongs to group "Money and Credit"

add_var1 <- data.frame(variables = "BOGMBASE_importance", group = "Money and Credit")
add_var2 <- data.frame(variables = "BOGMBASE_lag1_importance", group = "Money and Credit")
add_var3 <- data.frame(variables = "BOGMBASE_lag2_importance", group = "Money and Credit")

variables_group_assoc <- rbind(variables_group_assoc, add_var1, add_var2, add_var3)

# Filtering the variables in variables description (some variables in the description
# are not included in the models)

variables_group_assoc <- subset(variables_group_assoc,
                                variables %in% colnames(variables_importance_ridge))
```

Let's set up a dataframe where the columns correspond to the groups and each row corresponds to each rolling window. Using the dataframe with importance measures for variables, let's fill in the importance dataframe of groups using the association between variable and its respective group available in the variables_group_assoc dataframe.

```
# New dataframe with groups importance:

groups <- unique(variables_group_assoc$group)

# Compute the sum of the variables of a group in a row using lapply and rowSums

groups_sums <- lapply(groups, function(Group) {
  group_variables <- variables_group_assoc$variables[variables_group_assoc$group == Group]
  group_columns <- c(group_variables)
  rowSums(variables_importance_ridge[group_columns])
})

# Create a new dataframe with columns named by groups
groups_importance_ridge <- data.frame(groups_sums)

# Attribute the names of the columns to new dataframe
colnames(groups_importance_ridge) <- paste0(groups)
```

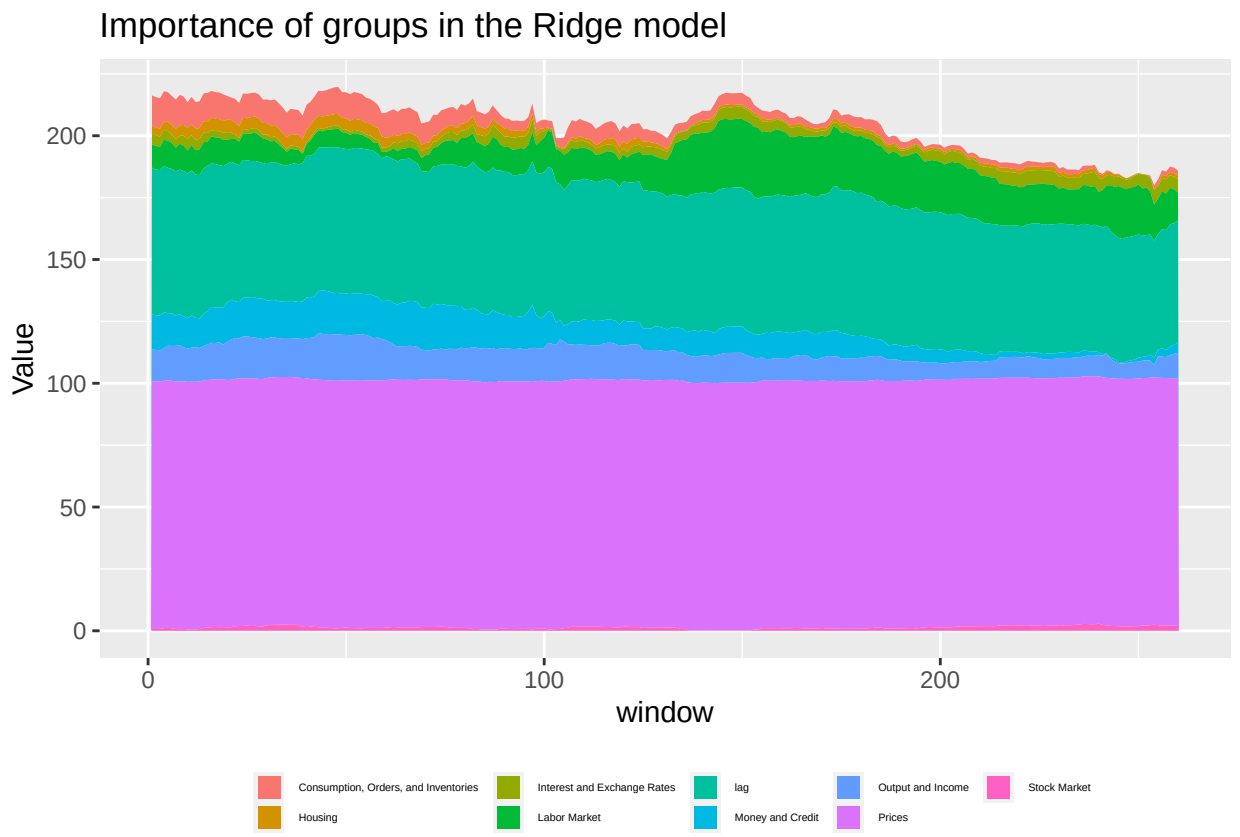
Next, we normalized the importance measures of the groups in each window by setting the largest value in module equal to 100.

```r
# Normalizing the groups importance

groups_importance_ridge <- apply(groups_importance_ridge, 1, function(row) {
  max_val <- max(abs(row))  # Find the maximum value in the row
  modified_row <- abs(100 * (row / max_val))  # Divide each value by the maximum value
  modified_row
})

# Transpose the result to match the original dataframe structure
groups_importance_ridge <- as.data.frame(t(groups_importance_ridge))
```

Let's create an area chart, where the x-axis consists of the sequential windows and the y-axis consists of the normalized importance of each group, where the thickness of the area is proportional to the importance measure. Each group is identified by a distinct color.

Let us apply this graph for the three models of this problem.

```r
# Creating a graph

groups_importance_ridge2 <- groups_importance_ridge %>%
  mutate(window = 1:nrow(groups_importance_ridge))

df_long_ridge <- groups_importance_ridge2 %>%
  pivot_longer(-window, names_to = "Variables", values_to = "Value")

# stacked area chart
ggplot(df_long_ridge, aes(x = window, y = Value, fill = Variables)) +
  guides(fill = guide_legend(title = "")) +
  labs(title = "Importance of groups in the Ridge model", fill = "") +
  geom_area(size = 2) +
  theme(legend.position = "bottom",
        legend.text = element_text(size = 3.5),
        legend.key.size = unit(0.4, "cm"),
        legend.spacing = unit(0.05, "cm"))
```



We now proceed in the same way for the LASSO model. First, we form the dataframe with the importance measures of groups.

```
# Compute the sum of the variables of a group in a row using lapply and rowSums

groups_sums <- lapply(groups, function(Group) {
  group_variables <- variables_group_assoc$variables[variables_group_assoc$group == Group]
  group_columns <- c(group_variables)
  rowSums(variables_importance_lasso[group_columns])
})

# Create a new dataframe with columns named by groups
groups_importance_lasso <- data.frame(groups_sums)

# Attribute the names of the columns to new dataframe
colnames(groups_importance_lasso) <- paste0(groups)
```

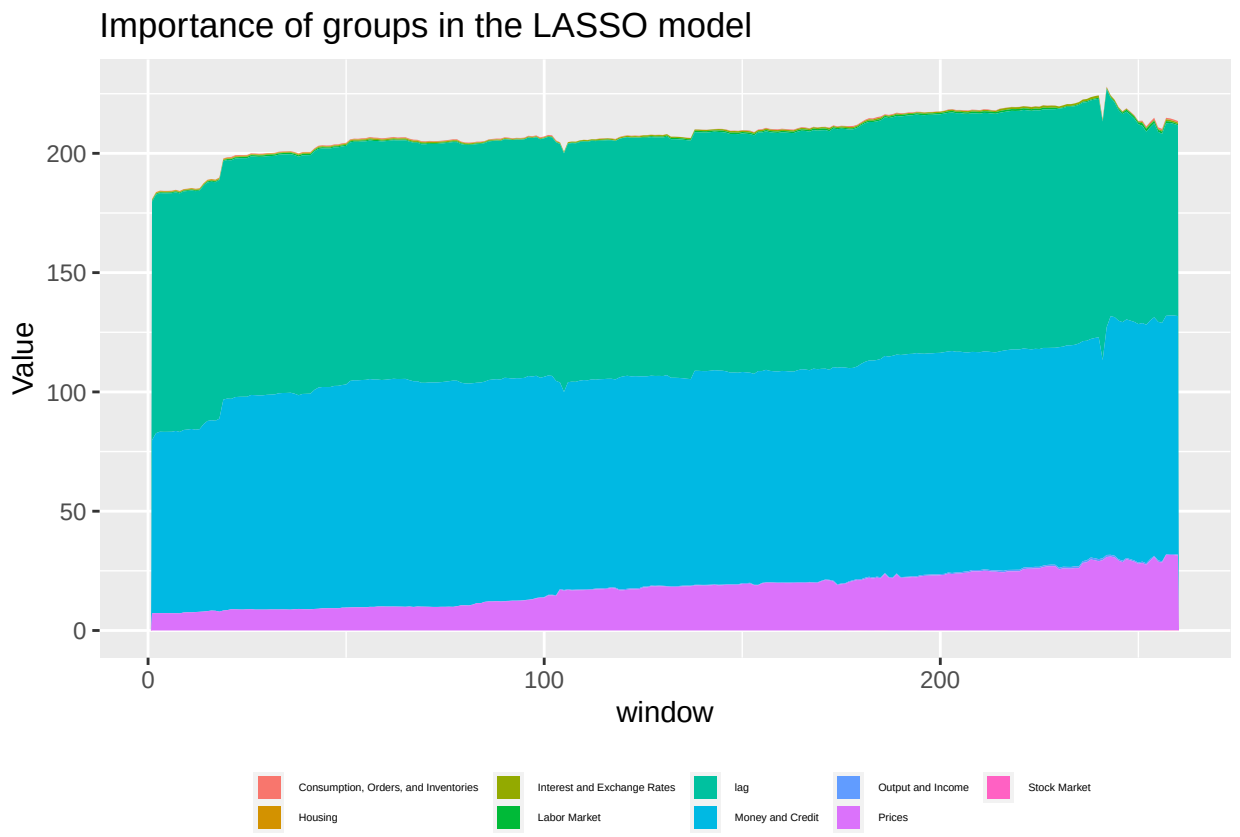
Next, we normalized the importance measures in the same way as in the Ridge model.

```
# Normalizing the groups importance

groups_importance_lasso <- apply(groups_importance_lasso, 1, function(row) {
  max_val <- max(abs(row))  # Find the maximum value in the row
  modified_row <- abs(100 * (row / max_val))  # Divide each value by the maximum value
  modified_row
})
```

```
# Transpose the result to match the original dataframe structure
groups_importance_lasso <- as.data.frame(t(groups_importance_lasso))
```

We then set up an area graph for the LASSO model.

```
# Creating a graph

groups_importance_lasso2 <- groups_importance_lasso %>%
  mutate(window = 1:nrow(groups_importance_lasso))

df_long_lasso <- groups_importance_lasso2 %>%
  pivot_longer(-window, names_to = "Variables", values_to = "Value")

# stacked area chart
ggplot(df_long_lasso, aes(x = window, y = Value, fill = Variables)) +
  guides(fill = guide_legend(title = "")) +
  labs(title = "Importance of groups in the LASSO model", fill = "") +
  geom_area(size = 2) +
  theme(legend.position = "bottom",
        legend.text = element_text(size = 3.5),
        legend.key.size = unit(0.4, "cm"),
        legend.spacing = unit(0.05, "cm"))
```



We will now compute the importance measures for the AR + PCR model. First, we set up the dataframe to store the importance measures for all variables in this model.

```r
# Finally, let's compute the measure of importance for PCR model

# Create a dataframe to store the importance of each variable

number_windows <- nrow(data_fred_laginf) - window_length + 1

variables_importance_pcr <- data.frame(matrix(ncol = ncol(data_fred_laginf) - 1,
                                              nrow = number_windows))

colnames(variables_importance_pcr) <- paste(colnames(data_fred_laginf[, -1]),
                                            "_importance", sep = "")

variables_importance_pcr <- subset(variables_importance_pcr,
                                   select = -c(inflation_importance))
```

Now, let's proceed with the estimation of the model to fill the newly created dataframe. For the variables in the PCR part of the model, the measure of importance is defined by the statement (product of the loading and the estimated coefficient).

For the AR part (with the two inflation lags), the importance measure considered is the same as for the LASSO and Ridge models: product of the standard deviation of each lag with its respective estimated coefficient.

```r
# Load required libraries
library(pls)

# Set the length of the moving window
window_length <- 492

# Set the initial position of the moving window
a <- 1

# Loop over the observations
while ((a + window_length - 1) <= nrow(data_fred_laginf)) {
  # Define the current window
  window <- data_fred_laginf[a:(a + window_length - 1), ]

  # Split the window into training matrix X and response vector y
  y <- window$inflation
  X <- window[ ,!colnames(window) %in% c("inflation", "inflation_lag1", "inflation_lag2", "date")]
  Z <- window[ , colnames(window) %in% c("inflation_lag1", "inflation_lag2")]

  # Compute the Principal Components and select the number of factors
  pca <- prcomp(X)
  cumulative_variance <- cumsum(pca$sdev^2) / sum(pca$sdev^2)
  num_factors <- which.max(cumulative_variance >= 0.9) # Number of factors explaining 90% of variance

  # Compute the principal components
  principal_components <- pca$x[ , 1:num_factors]

  # Prepare the lagged inflation variable
  lagged_inflation <- window[ , c("inflation_lag1", "inflation_lag2")]

  # Combine lagged inflation and principal components as predictors
```

```
  predictors <- cbind(lagged_inflation, principal_components)

  # Perform Principal Component Regression
  pcr_model <- pcr(y ~ ., data = data.frame(predictors), scale = TRUE, validation = "CV")

  # Compute the importance of each variable
  imp_pcr <- pca$rotation[ ,1:num_factors] %*% pcr_model$coefficients[-(1:2),  , num_factors]
  imp_inf1 <- apply(Z, 2, sd)[1] * pcr_model$coefficients[1,  , num_factors]
  imp_inf2 <- apply(Z, 2, sd)[2] * pcr_model$coefficients[2,  , num_factors]
  imp_pcr <- rbind(imp_pcr, imp_inf1, imp_inf2)

  # Store the variable importance
  variables_importance_pcr[a, ] <- t(imp_pcr)

  # Moving window forward
  a <- a + 1
}
```

With the importance measures for each variable obtained, we will now compute the importance measures for each group. We proceed in the same way as in the previous cases. We only need to filter the correspondence table, in order to keep only the inflation lags.

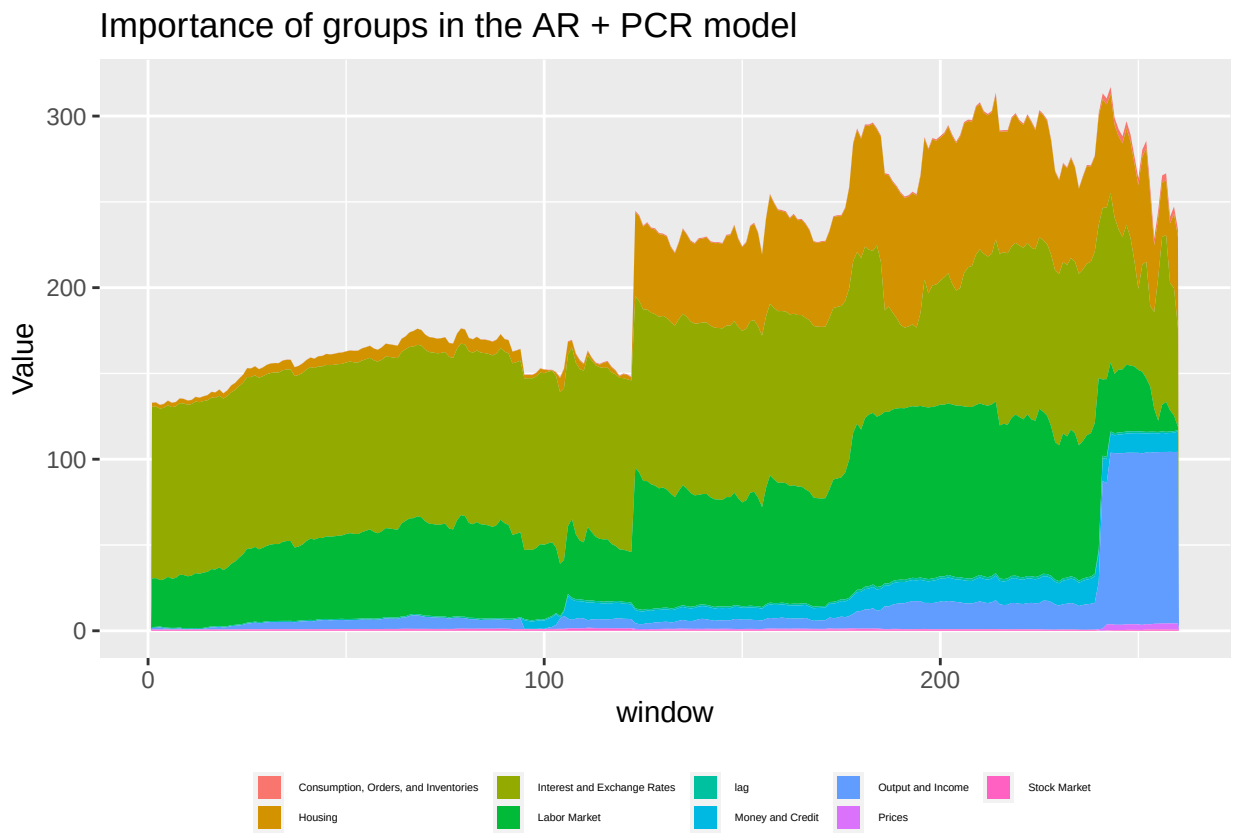
```
# We obtained a dataframe with importance of each variable
# We need to filter the variables names in the variables_groups_assoc (remove the
# lags of variables, except for inflation)

variables_group_assoc_inf <- variables_group_assoc[!(grepl("lag1", variables_group_assoc$variables) |
                                                       grepl("lag2", variables_group_assoc$variables))
                                                     grepl("inflation_lag1_importance", variables_group
                                                     grepl("inflation_lag2_importance", variables_group


# Now, we proceed exactly in the same way as Ridge and LASSO models

# Compute the sum of the variables of a group in a row using lapply and rowSums

groups_inf <- unique(variables_group_assoc_inf$group)

groups_sums <- lapply(groups_inf, function(Group) {
  group_variables <- variables_group_assoc_inf$variables[variables_group_assoc_inf$group == Group]
  group_columns <- c(group_variables)
  rowSums(variables_importance_pcr[group_columns])
})

# Create a new dataframe with columns named by groups
groups_importance_pcr <- data.frame(groups_sums)

# Attribute the names of the columns to new dataframe
colnames(groups_importance_pcr) <- paste0(groups_inf)
```

We will normalize the group importance measures using the same criteria as in the previous models.

```r
# Normalizing the groups importance

groups_importance_pcr <- apply(groups_importance_pcr, 1, function(row) {
  max_val <- max(abs(row))  # Find the maximum value in the row
  modified_row <- abs(100 * (row / max_val))  # Divide each value by the maximum value
  modified_row
})

# Transpose the result to match the original dataframe structure
groups_importance_pcr <- as.data.frame(t(groups_importance_pcr))
```

Finally, we will create an area plot for the AR + PCR model.

```r
# Creating a graph

groups_importance_pcr2 <- groups_importance_pcr %>%
  mutate(window = 1:nrow(groups_importance_pcr))

df_long_pcr <- groups_importance_pcr2 %>%
  pivot_longer(-window, names_to = "Variables", values_to = "Value")

# stacked area chart
ggplot(df_long_pcr, aes(x = window, y = Value, fill = Variables)) +
  guides(fill = guide_legend(title = "")) +
  labs(title = "Importance of groups in the AR + PCR model", fill = "") +
  geom_area(size = 2) +
  theme(legend.position = "bottom",
        legend.text = element_text(size = 3.5),
        legend.key.size = unit(0.4, "cm"),
        legend.spacing = unit(0.05, "cm"))
```



We find that the "Prices" group is the most important in the Ridge model, across all windows. The "lag" group is the second most important, with "Labor Market" and "Output and Income" showing secondary importance.

In the LASSO model, the "lag" group appears to be quite relevant, together with "Money and Credit". The "Prices" group is of secondary importance, and the other variables seem to be negligible.

Finally, in the AR + PCR model, "Interest and Exchange Rate" appears as the most important group, followed by "Labor Market". "Housing" emerges as an important group after the Global Financial Crisis of 2008. In the COVID 19 pandemic, "Output and Income" emerges as a relevant group. It is interesting to note that, in this model, there are greater changes in importance of groups of variables in the Global Financial Crisis and in the COVID 19 Pandemic.

Therefore, the three models differ considerably with respect to the degree of importance assigned to each of the nine groups. Specifically, "lag" and "Prices", important groups in the Ridge and LASSO models, present rather small relevance in the AR + PCR model.