Macroeconometrics and Machine Learning Project

Louis Blanco & Jeanne Astier

18 janvier 2023

This project is based on the article "Forecasting Inflation in a Data-Rich Environment : The Benefits of Machine Learning Methods" by Marcelo C. Medeiros, Gabriel F. R. Vasconcelos, Álvaro Veiga and Eduardo Zilberman (2019). It is organized as follows : part 1 explains the problem under study and its interests. Part 2 describes our replication of the article results in forecasting US inflation. Part 3 presents an extension of these results, applying the same methodology to forecast French inflation. Part 4 provides economic interpretation of the results and methodology used. Part 5 provides the codes used to get the results presented (also available here for better readability).

1 Introduction

Price Gains Slow More Than Expected : this is the title of the November Inflation Report published by the New York Times on December 13th 2022. It highlights the complexity of inflation forecasting : after a year during which the Fed repeatedly underestimated inflation's hold on the U.S. [1], it seems now to be overestimating it. These forecast errors, common to many Central Banks (the ECB addresses in its Economic Bulletin 2022 its recent errors in the inflation projections), are particularly problematic as inflation forecast is of paramount importance for economic decisions. Central Banks rely on inflation forecasts to inform their monetary policy. These forecasts are also used by households, private sector companies, policymakers to build their expectations and make their decisions. In some macroeconomic models, not only realized inflation but also expectations of inflation play a central role. Hence there is a discrepancy between the prime importance of inflation

1. *2022 shattered economic forecasts. Can the Fed get 2023 right ?*, The Washington Post, December 12th 2022.

forecasts and their lack of accuracy. These forecasts errors also undermine Central Banks credibility, degrading again the expectations. Poor inflation forecasts might thus have important welfare costs.

In this context, there is a challenge to improve performance of forecasting models. According to literature, the traditional simple univariate forecasting models turn out to be difficult to improve. Even the attempts to use Machine Learning (ML) models on a small set of variables, or "Big data" summarized in a small set of factors, did not seem conclusive. However, combining the two, as it has already been done outside of the economic field, might lead to significant improvement. The objective of the article studied is not only to beat univariate benchmark models, but also to prove that the combination of ML and "Big data" is the more effective that using just ML with few variables of Big Data with factor models.

2 Replication of article results

We replicate the article results by estimating benchmark models and several ML ones on a large set of variables to predict US inflation. We then compare the performances of the forecasts made by these different models, to determine which ones are the most accurate.

We face several obstacles in this replication, which lead us to make choices in how to run it. The main one is the computing power required to run all the models compared. Not only the ML models are power-consuming to be trained, but also all the processes are replicated many times (forecasts are computed at 12 different time horizons at each one of the 312 out-of-sample windows, with a rolling-window framework). This factor is highly limiting since our laptops do not have very high computing power, so the predictions can take very long to be computed [2]. This explains our choice to stick to the replication of general results of the paper, and not to replicate exhaustively all the robustness checks made by the authors that would have asked several days of code running with our tools. For this reason, we also gave up on replicating forecasts for 4 models among the 18 tested.

An other issue is that the data available to us do not seem to correspond exactly to what the authors describe. Even sticking to their description of the source of the data (January 2016 edition of FRED-MD data) and to the treatment they apply (among others, removing the variables with missing dates in the time period considered), we end up with slightly less variables than they do,

2. The total computing time for the replication results presented is about 35 hours.

without being able to identify which ones and the reasons for this. This might explain why our results statistics do not always exactly correspond to the article original ones.

As in the original article, we train several models on FRED-MD data from 1960 ; we compare performance of these different models in an out-of-sample window from January 1990 to December 2015. At each date t of the out-of-sample window, the compute forecasts based on a rolling-window framework of fixed length, at time horizons $t+1$, $t+1$, ..., $t+12$. We also compute forecasts for the accumulated inflation over the following 3, 6, and 12 months. The models are compared according to different statistics : root mean squared error (RMSE) and mean absolute error (MAE). We also compute, for square and absolute losses, respectively, the average p-values (accross horizons) for the model confidence sets (MCSs) based on the Tmax statistic as described in Hansen, Lunde, and Nason (2011). The estimated models are the following ones :

— benchmark models : Random Walk (RW), Auto-Regressive (AR)
— shrinkage models : Ridge regression, Least Absolute Shrinkage and Selection Operator (LASSO), adaptive LASSO (adaLASSO), Elastic Net (ElNet), adaptive Elastic Net (adaElNet)
— factor models : classic factor model (Factor), Target factor model (T. Factor)
— ensemble methods : Bagging, Complete Subset Regressions (CSR)
— Breiman Random Forest (RF)
— hybrid linear-Random Forest model : Random-Forest Ordinary Least Squares (RF/OLS).

The results of this replication are summarized in Tables 1 and 2.

3 Extension : application to forecasting of French inflation

In the conclusion of their article, Marcelo C. Medeiros, Gabriel F. R. Vasconcelos, Álvaro Veiga and Eduardo Zilberman write : "*Although our article focuses on inflation forecasting in the US, one can easily apply ML methods to forecast other macroeconomic series in a variety of countries.*" Hence as an extension, we try to apply the benchmark and ML models comparison implemented in their article to French data, aiming at forecasting French inflation. There is a double issue at stake with this extension. First, is the domination of Random Forest over all other models (see part 4) specific to US data, or does it hold on a different dataset ? Second, are the ML models still performing better

Table 1 – Replication of forecasting results : summary statistics for the out-of-sample period from 1990 to 2015

	Forecasting precision								Model confidence set	
	(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)	(9)	(10)
Models	ave. RMSE	ave. MAE	max RMSE	max MAE	min RMSE	min MAE	# min RMSE	# min MAE	ave. p-v. Tmax sq	ave. p-v. Tmax abs
RW	1.00	1.00	1.00	1.00	1.00	1.00	0	0	0.00	0.00
AR	0.61	0.81	0.88	0.87	0.56	0.70	2	0	1.00	0.32
LASSO	0.75	0.94	0.84	1.01	0.62	0.67	3	0	0.28	0.32
adaLASSO	0.85	0.96	0.88	1.04	0.64	0.66	0	0	0.07	0.13
ElNet	0.76	0.87	0.81	1.05	0.62	0.68	1	1	0.26	0.32
adaElNet	0.85	0.98	0.88	1.06	0.65	0.69	1	1	0.08	0.13
Ridge	0.79	0.89	0.86	0.92	0.73	0.71	0	1	0.05	0.13
BVAR	0.62	0.82	0.90	0.89	0.57	0.70	0	0	0.32	0.32
Bagging	1.38	1.25	1.50	1.35	0.82	0.83	0	0	0.05	0.05
CSR	0.74	0.90	0.82	0.95	0.65	0.69	0	0	0.07	0.13
Factor	0.73	0.89	0.83	1.01	0.58	0.66	1	2	0.29	0.32
T. Factor	0.65	0.80	0.84	0.85	0.58	0.66	1	0	0.32	0.32
RF	0.69	0.80	0.83	0.83	0.66	0.71	5	9	0.29	1.00
RF/OLS	0.77	0.87	0.81	0.91	0.64	0.71	1	1	0.20	0.32

NOTE : The table reports for each model different summary statistics across all the forecasting horizons (1 to 12 months and accumulated 3, 6 and 12-month). Columns (1) and (2) report the average root mean square error (RMSE) and the average mean absolute error (MAE). Columns (3) ans (4) report, respectively, the maximum RMSE and MAE over the horizons considered. Columns (5) and (6) report, respectively, the minimum RMSE and MAE over the horizons considered. All these statistics (columns 1 to 6) are normalized for the benchmark RW model to one. Columns (7) and (8) report the number of times (across horizons) each model achieved the lowest RMSE and MAE, respectively. Columns (9) and (10) present for square and absolute losses, the average p-values for the model confidence sets (MCSs) based on the Tmax statistic as described in Hansen, Lunde, and Nason (2011).

than benchmarks with a smaller number of covariates ? Indeed, we cannot not find as many monthly series available on a large time period for France as we have for the US with FRED-MD dataset. We download monthly series about France from FRED database, but way less series are available than for the US, and additionally most of them start later than 1960. We will thus see how much "Big Data" is needed for ML models to outperform univariate ones. Our dataset is made of FRED monthly time series about France. The variables cover the same fields as the US ones used in the original article : output and income ; labor market ; housing ; consumption, orders, and inventories ; money and credit ; interest and exchange rates ; prices ; stock market. The dataset we use has 64 variables from January 1985 to December 2020. Additionnally to FRED time series, we use French

Table 2 – Replication of forecasting results : RMSE and MAE ratios (1990–2015)

Panel (a) : RMSE ratio

	Forecasting horizon														
Model	1	2	3	4	5	6	7	8	9	10	11	12	3m	6m	12m
AR	0,88	0,80	0,78	0,79	0,77	0,77	0,76	0,74	0,75	0,81	0,81	0,74	**0,64**	**0,66**	**0,56**
RF	**0,84**	**0,75**	**0,73**	**0,76**	**0,73**	**0,74**	**0,75**	**0,74**	**0,74**	**0,78**	**0,80**	**0,72**	0,66	0,73	0,67

Panel (b) : MAE ratio

	Forecasting horizon														
Model	1	2	3	4	5	6	7	8	9	10	11	12	3m	6m	12m
AR	0,87	0,78	0,77	0,78	0,79	0,78	0,75	0,74	0,78	0,83	0,85	0,75	**0,70**	**0,80**	0,82
RF	**0,81**	**0,72**	**0,71**	**0,74**	**0,74**	**0,75**	**0,73**	**0,72**	**0,75**	**0,77**	**0,81**	**0,73**	0,71	0,84	**0,81**

NOTE : The table reports, for each forecasting horizon (1 to 12 months and accumulated 3, 6 and 12 months), the root mean squared error (RMSE) and mean absolute error (MAE) ratios with respect to the random walk model for the full out-of-sample period (1990–2015). The statistics for the best-performing model are highlighted in bold.

inflation data series from OECD databse, as a larger time span is available.

We train and test on these French data the same models we have trained and tested on the US data ; but taking into account the different time span available, their performances are compared on a reduced out-of-sample window, from 2005 to 2020. Hence, compared to the replication part, the lengths of both train and test datasets are reduced. We compute, at each date t of the out-of-sample window, the same forecasts based on a rolling-window framework of fixed length, at time horizons $t+1$, $t+1$, ..., $t+12$, as well as accumulated inflation over the following 3, 6, and 12 months. RMSE and MAE statistics and average p-values for MCSs based on the Tmax are used to compare the different models.

The results of this replication are available in Tables 3 and 4.

4 Interpretation of the results

Replication of forecasting US inflation leads to results summarized in Tables 1 and 2. Table 1 reports summary statistics across all forecasting horizons, on the period 1990 to 2015. It shows that ML models (except Bagging) and the use of a large set of predictors systematically improve the quality of inflation forecasts over RW benchmark. The AR benchmark however seems to perform better than several ML models. The RF model clearly outperforms all the other ML alternatives ; it

TABLE 3 – Extension : results for forecasting of French inflation - summary statistics for the out-of-sample period from 2000 to 2015

	Forecasting precision								Model confidence set	
	(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)	(9)	(10)
Models	ave. RMSE	ave. MAE	max RMSE	max MAE	min RMSE	min MAE	# min RMSE	# min MAE	ave. p-v. Tmax sq	ave. p-v. Tmax abs
RW	1.00	1.00	1.00	1.00	1.00	1.00	0	0	0.00	0.18
AR	0.59	0.61	1.05	1.08	0.32	0.36	0	0	0.01	0.15
LASSO	0.78	0.64	1.73	1.17	0.11	0.12	1	1	0.00	0.18
adaLASSO	0.90	0.62	2.41	1.22	0.33	0.25	0	2	0.00	0.06
ElNet	0.70	0.68	1.48	1.21	0.12	0.12	0	0	0.00	0.07
adaElNet	0.75	0.63	1.54	1.22	0.39	0.27	1	1	0.00	0.07
Ridge	0.87	0.94	2.84	2.99	0.64	0.70	2	3	0.17	0.00
BVAR	0.57	0.59	1.01	1.06	0.30	0.34	0	0	0.00	0.00
Bagging	5.16	1.40	20.32	4.01	0.89	0.44	0	0	0.00	0.00
CSR	0.72	0.61	1.41	1.19	0.44	0.39	0	0	0.00	0.01
Factor	0.49	0.52	0.96	0.99	0.23	0.28	11	8	1.00	1.00
RF	0.61	0.62	1.27	1.19	0.36	0.36	0	0	0.01	0.01
RF/OLS	1.13	0.71	3.12	1.49	0.30	0.30	0	0	0.00	0.00

NOTE : The table reports for each model different summary statistics across all the forecasting horizons (1 to 12 months and accumulated 3, 6 and 12-month). Columns (1) and (2) report the average root mean square error (RMSE) and the average mean absolute error (MAE). Columns (3) ans (4) report, respectively, the maximum RMSE and MAE over the horizons considered. Columns (5) and (6) report, respectively, the minimum RMSE and MAE over the horizons considered. All these statistics (columns 1 to 6) are normalized for the benchmark RW model to one. Columns (7) and (8) report the number of times (across horizons) each model achieved the lowest RMSE and MAE, respectively. Columns (9) and (10) present for square and absolute losses, the average p-values for the model confidence sets (MCSs) based on the Tmax statistic as described in Hansen, Lunde, and Nason (2011).

also seems to beat the AR benchmark. RF has the lowest MAEs across the horizons and its RMSEs are of same orders of magnitude as AR ones. The RF model also has the highest p-value in the MCS with absolute errors (but AR model has the highest p-value with squared errors). Finally, RF model presents the highest number of lowest RMSEs and MAEs accross horizons.

Our replication thus leads to the same conclusion as the original article : superiority of RF. However, our results show a slightly more qualified supremacy of RF than the original results. In particular, in the replication the AR benchmark seems to perform better than in the original article, and RF model slightly worse, leading to close forecast qualities for those two models. This could be explained by the issue mentioned in Part 2, that the data we use to contains less variables than what is described in the article. The "Big Data" would then be less "Big", leading to lower accuracy in

Table 4 – Extension : results for forecasting of French inflation - RMSE and MAE ratios (1990–2015)

Panel (a) : RMSE ratio

	Forecasting horizon														
Model	1	2	3	4	5	6	7	8	9	10	11	12	3m	6m	12m
AR	**1,01**	**1,04**	**1,05**	**1,04**	1,03	1,01	0,99	0,97	0,96	0,95	0,94	0,94	**0,44**	**0,32**	0,39
RF	1,27	1,21	1,16	1,10	**1,02**	**0,96**	**0,93**	**0,90**	**0,90**	**0,90**	**0,90**	**0,90**	0,74	0,48	**0,36**

Panel (b) : MAE ratio

	Forecasting horizon														
Model	1	2	3	4	5	6	7	8	9	10	11	12	3m	6m	12m
AR	**1,02**	**1,03**	**1,05**	**1,06**	1,08	1,07	1,06	1,03	1,02	1,01	0,99	0,98	**0,45**	**0,36**	0,40
RF	1,19	1,17	1,15	1,11	**1,03**	**0,98**	**0,94**	**0,95**	**0,96**	**0,99**	**0,98**	**0,98**	0,73	0,51	**0,36**

NOTE : The table reports, for each forecasting horizon (1 to 12 months and accumulated 3, 6 and 12 months), the root mean squared error (RMSE) and mean absolute error (MAE) ratios with respect to the random walk model for the full out-of-sample period (2000–2015). The statistics for the best-performing model are highlighted in bold.

forecasting of RF model compared to benchmark.

The replication results also show, as the original article does, that among shrinkage models, sparsity-inducing methods are slightly worse than nonsparsity-inducing ones. Since RF does not impose sparsity, this might suggest that sparsity is not a desired feature to improve forecasting accuracy. On the contrary, quite different from article results, factor models do not seem to be strongly outperformed by other methods. The adoption of target factors improves the quality of the forecasts and make the results quite close to the RF ones, even if less accurate.

Table 2 shows more in detail the results of the comparison between RF and the AR benchmark, according to RMSE and MAE ratios with respect to the RW alternative for all forecasting horizons. We see that RF is beating AR and RW benchmarks on all time horizons from $t + 1$ to $t + 12$, according to both RMSE and MAE ; but the AR benchmark is performing better on accumulated inflation forecasts. This better performance of AR on accumulated inflation explains why on average, AR accuracy is close to (but lower than) RF one.

Extension to forecasting French inflation leads to results summarized in Tables 3 and 4. Table 3 reports summary statistics across all forecasting horizons, on the period 2005 to 2020. It shows that, if ML models allow to improve the quality of inflation forecasts over RW benchmark on average, their performance is not systematically better as it was the case with US data. Particularly, the accuracy improvement does not seem to be constant over horizons : all ML models perform worse than RW benchmark at least once (they have a maximum RMSE and maximum MAE greater than one). We can also notice that the AR benchmark seems to perform better than several ML models. Contrary to what we found with US data, the supremacy of RF model over other ML alternatives is not very clear : it has the lower average and maximum RMSE and MAE, but is outperformed by LASSO and ElNet concerning minimum RMSE and MAE, and has a very low p-value in MCSs. Table 4 shows more in detail the results of the comparison between RF and the AR benchmark : as opposed to extension results, we see that AR is more accurate than RF not only for the accumulated inflation forecasts, but also for the horizons $t + 1$ to $t + 4$. It is thus clear that AR benchmark is not outperformed by RF model as much as it is with US data. Most importantly, the factor model is performing way better than it was with US data, and is outperforming all benchmarks and ML alternatives : it beats the RW benchamrk in terms on RMSE and MAE at all horizons, it has the lowest average and max RMSE and MAE, it is at most time horizons the model with lower RMSE and MAE, and it as the highest p-value in MCSs.

Our extension interestingly develops the results of original article and replication. It shows that, even though ML models can beat RW benchmark, factor model performs the best when the number of variables available is reduced. It thus confirms that improving usual forecasts cannot be done just by replacing traditional models with ML ones : an other key feature is so-called "Big Data". Applying ML models to a reduced number of variables, as we did with French data, does not allow them to forecast inflation more accurately than AR benchmark. In this context of no "Big Data", the factor model seems to perform the best. This conclusion is consistent with the original article stance.

This conclusion is however a little vague, and we do not precisely give a definition for "Big Data". In the original article, 122 variables are used to forecast US inflation, leading to supremacy of RF model. In our extension, with use 62 variables and factor model remains better than ML models. Studying more precisely what number of variables is needed for ML models to bring an improvement would be interesting. It would of course also depend on what variables are used.

The extension results, as the original article and replication results, also show that, among shrinkage models, sparsity-inducing methods (LASSO, adaLASSO, ElNet, adaElNEt) are slightly worse than non-sparsity-inducing ones (RR). This confirms that the higher the number of variables, the better the results of ML models.

The high performance of factor model in forecasting of French inflation is consistent with its high popularity for macroeconomic forecasting in a context of no "Big Data" : when there are not a lot a regressors available (as it is the case in our extension and more generally in traditionnal macroeconomic forecasting), the high level of aggregation of factor models seems to be adequate, but is not anymore when the number of regressors increases (as it is the case in the original article and, to a lesser extent, in our replication). The RF model becomes more performant only when the number of regressors increases. This is also consistent with a conclusion of the original article, that the superiority of RF is due both to the variable selection mechanism induced by the model and to the presence of nonlinearities in the relation between inflation and its predictors. When more variables are available, the variable selection mechanism can only be more relevant. Moreover, the higher the number of variables, the higher the probability to have non-linear relationships between regressors and inflation.

These results lead us to re-evaluate the authors' stance that "*one can easily apply ML methods to forecast other macroeconomic series in a variety of countries.*" For ML methods to be as relevant for forecasting other countries macroeconomic series as they are for the US, these countries need to have a high data quantity available over a large time span, which is not necessarily the case, as we have seen with France.

5 Appendix : code

The R code used to get the results presented above is included below. It sticks as close as possible to the original article. For better readability, it can also be found on this repository.

5.1 Replication

The code for the replication part is divided into 4 parts : data preparation, RW and out-of-sample inflation, computation of forecasts with the different models compared, model performances comparison. These 4 scripts also call functions from two other scripts : functions.R and rolling_window.R.

5.1.1 Data preparation

```
# library dedicated to handle FRED data
library(devtools)
install_github("cykbennie/fbi", force = TRUE)

# other libraries
library(fbi)
library(tidyverse)
library(TTR)



# set working directory
setwd(dirname(rstudioapi::getSourceEditorContext()$path))

start_date = "1960-01-01"

# 1. loading data from FRED-MD
# the 'fredmd' function by defaults transform the data to stationnarize it
# (first difference, second difference, log, first or second difference of log,
# first difference of percent point).
data = fredmd("https://files.stlouisfed.org/files/htdocs/fred-md/
monthly/2016-02.csv")

# download of data without transformation
data_raw = fredmd("https://files.stlouisfed.org/files/htdocs/fred-md/monthly/2016-0

# meta-data about the FRED variables
varlist = fredmd_description
vars = intersect(colnames(data),varlist$fred)

# 2. data cleaning
```

```r
# keep only variables for which we have meta-data
data = data %>% as_tibble()%>%
  select(all_of(c("date",vars)))
varlist = varlist%>%filter(fred%in%vars)

# additional transformation for price variables : simple differenciation of logs
# (= inflation)
prices_varlist = varlist%>%filter(group=="Prices",tcode==6)
data = data%>% as_tibble()%>%
  select( -all_of(prices_varlist$fred) )
prices = data_raw %>% as_tibble() %>%
  select(all_of(prices_varlist$fred))%>%
  mutate_all(.funs = function(x)100*c(NA,x%>%log()%>%diff()))
data = cbind(data%>%as.data.frame(),prices%>%as.data.frame())

# we keep only variables with all observations in the sample period
data = data %>%
  filter(date>=start_date)%>%
  select_if(~ !any(is.na(.)))

save(data,file = "data/data.rda")
```

5.1.2 RW and out-of-sample inflation

```r
# libraries
library(roll) # library for rolling window framework

# set working directory
setwd(dirname(rstudioapi::getSourceEditorContext()$path))

# load data
load("data/data.rda")
```

```
dates = data$date
data = data%>%select(-date)%>%as.matrix()
rownames(data) = as.character(dates)

# out-of-sample window to compare performances of models

# subsample period : Jan1990 to Dec2015 (312 obs)
nwindows = 312

# 1. out of sample true inflation (y)
# y = inflation computed from baseline price index (CPI)
y = data[,"CPIAUCSL"]

# for forecasts for the accumulated inflation over the following 3, 6,
# and 12 months
y = cbind(y,roll_prod(1+y,3)-1,roll_prod(1+y,6)-1,roll_prod(1+y,12)-1)
# rolling products

yout = tail(y,nwindows) # true value of inflation, to compare with forecasts

# 2. benchmark model: random walk forecasts

# empty matrix
rw = matrix(NA,nwindows,12)

# forecast at date t for inflation(t+h) is inflation(t)
for(i in 1:12){
  aux = data[(nrow(data)-nwindows-i+1):(nrow(data)-i),"CPIAUCSL"]
  rw[,i]=aux;
}

# forecast at date t for accumulated h-month inflation(t+1:t+h) is
# inflation(t-(h-1):t)
rw3 = tail(embed(y[,2],4)[,4],nwindows)
rw6 = tail(embed(y[,3],7)[,7],nwindows)
rw12 = tail(embed(y[,4],13)[,13],nwindows)
rw = cbind(rw,rw3,rw6,rw12)
colnames(rw) = c(paste("t+",1:12,sep = ""),"acc3","acc6","acc12")


save(yout,file = "forecasts/yout.rda")
save(rw,file = "forecasts/rw.rda")
```

5.1.3 Building of forecasts with models

This code has to be run for every model tested on the data, changing model and function names and adding the appropriate options when running the rolling window (adaptive, etc.).

```
### packages for specific models ###
library(devtools)
# install_github("gabrielrvsc/HDeconometrics")
library(HDeconometrics)
library(glmnet)
library(randomForest)

# set working directory
setwd(dirname(rstudioapi::getSourceEditorContext()$path))

source("functions/rolling_window.R")
source("functions/functions.R")

##### CHANGES TO BE MADE HERE TO RUN DIFFERENT MODELS ####
# The file with the forecasts will be saved with model_name
model_name = "LASSO"
# The function called to run models is model_function,
# which is a function from functions.R
model_function = runlasso
#####

load("data/data.rda")
dates = data$date
data = data%>%select(-date)%>%as.matrix()
rownames(data) = as.character(dates)

####### run rolling window ##########
nwindows = 312
model_list = list()

for(i in 1:12){
  old <- Sys.time() # get start time
  model = rolling_window(model_function,data,nwindows+i-1,i,"CPIAUCSL")
  model_list[[i]] = model
  cat(i,"\n")
  new <- Sys.time() - old # calculate difference
  print(new) # print in nice format
}

forecasts = Reduce(cbind,lapply(model_list,
function(x)head(x$forecast,nwindows)))

forecasts = accumulate_model(forecasts)

save(forecasts,file = paste("forecasts/",model_name,".rda",sep = ""))

plot(tail(data[,"CPIAUCSL"],312),type = "l")
lines(forecasts[,1],col = 3)
```

5.1.4 Comparison of models performances

```
library(tidyverse)
library("writexl")
library(MCS) # model confidence sets procedure
library(mlRFinance) # must be installed from github.com/PedroBSB/mlRFinance
library(modelconf) # must be installed from github.com/nielsaka/modelconf

# set working directory
setwd(dirname(rstudioapi::getSourceEditorContext()$path))

load("forecasts/yout.rda")
load("forecasts/rw.rda")

# load all forecasts
model_files = setdiff(list.files("forecasts/"),c("rw.rda","yout.rda"))

models_list = list()

for(i in 1:length(model_files)){
  load(paste("forecasts/",model_files[i],sep = ""))

    # correct accumulated predictions
  forecasts = forecasts[,1:(ncol(forecasts)-3)] # remove accumulated forecasts
  acc3 = sapply(1:(nrow(forecasts)), function(x){
    prod(1+(forecasts[x,1:3]))-1
  })
  acc6 = sapply(1:(nrow(forecasts)), function(x){
    prod(1+(forecasts[x,1:6]))-1
  })
  acc12 = sapply(1:(nrow(forecasts)), function(x){
    prod(1+(forecasts[x,1:12]))-1
  })

  forecasts = cbind(forecasts,acc3,acc6,acc12)
  colnames(forecasts) = c(paste("t+",1:12,sep = ""),"acc3","acc6","acc12")

  models_list[[i]] = forecasts
}

names(models_list) = model_files

### ROOT MEAN SQUARED ERRORS ###

# RMSE of random walk predictions
rwe = sqrt(colMeans((rw[,1:12]-yout[,1])^2))

# MSE of other model predictions
errors = lapply(models_list, function(x){
  sqrt(colMeans((x[,1:12]-yout[,1])^2))
})%>% Reduce(f=cbind)
colnames(errors) = model_files

# MSE of accumulated predictions
rweacc = sqrt(colMeans((rw[,13:15]-yout[,2:4])^2))
errorsacc = lapply(models_list, function(x){
  sqrt(colMeans((x[,13:15]-yout[,2:4])^2,na.rm=TRUE))
})%>% Reduce(f=cbind)
colnames(errorsacc) = model_files

# final table with MSE of all models
res = rbind(errors, errorsacc)
resrw = rbind(cbind(rwe), cbind(rweacc))
res = cbind(res, resrw)
```

```
res_df = data.frame(res)

# average MSE
mean <- summarize_all(res_df, mean)
res_df <- rbind(res_df, mean)

# normalize w.r.t MSE random wals
res_mse_df <- res_df[,1:(ncol(res_df))]/res_df[, ncol(res_df)]

# max MSE
max <- summarize_all(res_mse_df, max)
res_mse_df <- rbind(res_mse_df, max)

# min MSE
min <- summarize_all(res_mse_df, min)
res_mse_df <- rbind(res_mse_df, min)

rownames(res_mse_df) <- c(paste("t+",1:12,sep = ""),
"acc3","acc6","acc12", "average", "max", "min")

# model with lower MSE for each horizon
res_mse_df$is_min <- names(res_mse_df)[apply(res_mse_df, MARGIN = 1,
FUN = which.min)]

# export
write_xlsx(res_mse_df,"MSE.xlsx")



### MEAN ABSOLUTE ERROR ###

# MEA of random walk predictions
rwe = (colMeans(abs(rw[,1:12]-yout[,1])))
```

```
# MEA of other model predictions
errors = lapply(models_list, function(x){
  (colMeans(abs(x[,1:12]-yout[,1])))
})%>% Reduce(f=cbind)
colnames(errors) = model_files

# MAE of accumulated predictions
rweacc = (colMeans(abs(rw[,13:15]-yout[,2:4])))
errorsacc = lapply(models_list, function(x){
  (colMeans(abs(x[,13:15]-yout[,2:4]),na.rm=TRUE))
})%>% Reduce(f=cbind)
colnames(errorsacc) = model_files

# final table with MAE of all models
res = rbind(errors, errorsacc)
resrw = rbind(cbind(rwe), cbind(rweacc))
res = cbind(res, resrw)
res_df = data.frame(res)

# average MAE
mean <- summarize_all(res_df, mean)
res_df <- rbind(res_df, mean)

# normalize w.r.t MSE random wals
res_mae_df <- res_df[,1:(ncol(res_df))]/res_df[, ncol(res_df)]

# max MAE
max <- summarize_all(res_mae_df, max)
res_mae_df <- rbind(res_mae_df, max)

# min MAE
```

```r
min <- summarize_all(res_mae_df, min)
res_mae_df <- rbind(res_mae_df, min)

rownames(res_mae_df) <- c(paste("t+",1:12,sep = ""),
"acc3","acc6","acc12", "average", "max", "min")

# model with lower MAE for each horizon
res_mae_df$is_min <- names(res_mae_df)[apply(res_mae_df, MARGIN = 1,
FUN = which.min)]

# export
write_xlsx(res_mae_df,"MAE.xlsx")



### MCS TEST WITH RMSE ###

# RMSE of random walk predictions
rwe = sqrt(colMeans((rw[,1:12]-yout[,1])^2))

# MSE of other model predictions
errors = lapply(models_list, function(x){
  sqrt(colMeans((x[,1:12]-yout[,1])^2))
})%>% Reduce(f=cbind)
colnames(errors) = model_files

# MSE of accumulated predictions
rweacc = sqrt(colMeans((rw[,13:15]-yout[,2:4])^2))
errorsacc = lapply(models_list, function(x){
  sqrt(colMeans((x[,13:15]-yout[,2:4])^2,na.rm=TRUE))
})%>% Reduce(f=cbind)
colnames(errorsacc) = model_files
```

```r
# final table with MSE of all models
res = rbind(errors, errorsacc)
resrw = rbind(cbind(rwe), cbind(rweacc))
res = cbind(res, resrw)
res_df = data.frame(res)

# MCS test
estMCS(res_df[1:15,], test = "t.max", B = 25000, l = 12)


### MCS TEST WITH MAE ###

# MEA of random walk predictions
rwe = (colMeans(abs(rw[,1:12]-yout[,1])))

# MEA of other model predictions
errors = lapply(models_list, function(x){
  (colMeans(abs(x[,1:12]-yout[,1])))
})%>% Reduce(f=cbind)
colnames(errors) = model_files

# MAE of accumulated predictions
rweacc = (colMeans(abs(rw[,13:15]-yout[,2:4])))
errorsacc = lapply(models_list, function(x){
  (colMeans(abs(x[,13:15]-yout[,2:4]),na.rm=TRUE))
})%>% Reduce(f=cbind)
colnames(errorsacc) = model_files

# final table with MAE of all models
res = rbind(errors, errorsacc)
resrw = rbind(cbind(rwe), cbind(rweacc))
res = cbind(res, resrw)
```

```
res_df = data.frame(res)

# MCS test
estMCS(res_df[1:15,], test = "t.max", B = 25000, l = 12)
```

5.1.5 functions.R

```
## different forecast models to compare ##



# dataprep function: preparation of the data taken as input by the model
dataprep = function(ind,df,variable,horizon,add_dummy = TRUE, univar = FALSE, facto
{
  df=df[ind,] # predictors
  y=df[,variable] # variable to predict

  if(nofact==TRUE){
    if(univar==FALSE){
      x=df # prediction of y by the other variables
    }else{
      x = as.matrix(df[,variable]) # y predicted only by its past values
    }
  }else{
    if(univar==FALSE){
      factors=princomp(scale(df))$scores[,1:4] # computation of 4 principal compone
      if(factonly == TRUE){
        x = cbind(df[,variable],factors) # principal components only
      }else{
        x = cbind(df,factors) # add principal components to other predictors
      }
    }else{
      x = as.matrix(df[,variable]) # y predicted only by its past values
    }
  }

  X=embed(as.matrix(x),4)

  Xin=X[-c((nrow(X)-horizon+1):nrow(X)),]  # train sample for predictors
  Xout=X[nrow(X),]
  Xout=t(as.vector(Xout)) # test sample for predictors
  yin=tail(y,nrow(Xin)) # train sample for predicted y

  # deal with inflation outlier in 2008
  if("2008-11-01" %in% names(yin)){

    dummy=rep(0,length(yin))
    intervention=which(names(yin)=="2008-11-01")
    dummy[intervention]=1
    if(add_dummy == TRUE){
      Xin=cbind(Xin,dummy)
      Xout=cbind(Xout,0)
    }

  }else{
    dummy = rep(0,length(yin))
    if(add_dummy == TRUE){
      Xin=cbind(Xin,dummy)
      Xout=cbind(Xout,0)
    }
  }

  return(list(dummy = dummy, Xin = Xin, Xout = Xout, yin = yin))
}
```

```
# LASSO model (non-adaptative and adaptative)
# Ridge model (when alpha = 0)
# ElasticNet model (when alpha = 0.5) (non-adaptive and adaptive)
runlasso=function(ind,df,variable,horizon, alpha = 1, alpha2 = 1, adaptive = FALSE){
  prep_data = dataprep(ind,df,variable,horizon)
  Xin = prep_data$Xin
  yin = prep_data$yin
  Xout = prep_data$Xout

  modelest = ic.glmnet(Xin,yin,alpha = alpha)
  if(adaptive==TRUE){
    classo = coef(modelest)
    penalty = (abs(classo[-1])+1/sqrt(length(yin)))^(-1)
    modelest = ic.glmnet(Xin,yin, penalty.factor = penalty, alpha = alpha2)
  }

  forecast=predict(modelest,Xout)

  ### outputs ###
  coeflvl=coef(modelest)[-1]
  coefpar=coeflvl*apply(Xin,2,sd)
  lambda=modelest$lambda
  outputs=list(coeflvl=coeflvl,coefpar=coefpar,lambda=lambda)

  return(list(forecast=forecast, outputs=outputs))
}

# AR model
runar=function(ind,df,variable,horizon, type = "fixed"){
  prep_data = dataprep(ind,df,variable,horizon, univar = TRUE, add_dummy = FALSE)
  Xin = prep_data$Xin
  yin = prep_data$yin
```

```r
  Xout = prep_data$Xout
  dummy = prep_data$dummy

  if(type=="fixed"){
    modelest=lm(yin~Xin+dummy)
    best = ncol(Xin)
  }

  if(type=="bic"){
    bb=Inf
    best = 1
    for(i in seq(1,ncol(Xin),1)){
      m=lm(yin~Xin[,1:i]+dummy)
      crit=BIC(m)
      if(crit<bb){
        bb=crit
        modelest=m
        best = i
      }
    }
  }
  coef=coef(modelest)
  coef[is.na(coef)] = 0
  forecast=c(1,Xout[,1:best],0)%*%coef

  return(list(forecast=forecast))
}

# Random Forest model
runrf=function(ind,df,variable,horizon){
  prep_data = dataprep(ind,df,variable,horizon)
  Xin = prep_data$Xin
  yin = prep_data$yin
  Xout = prep_data$Xout

  modelest=randomForest::randomForest(Xin,yin, importance = TRUE)
  forecast=predict(modelest,Xout)

  ## outputs
  importance = randomForest::importance(modelest)
  outputs = list(importance = importance)

  return(list(forecast=forecast, outputs = outputs))
}

# RF/OLS model
runrfols=function(ind,df,variable,horizon){
  prep_data = dataprep(ind,df,variable,horizon, add_dummy = FALSE)
  Xin = prep_data$Xin
  yin = prep_data$yin
  Xout = prep_data$Xout
  dummy = prep_data$dummy

  modelest=randomForest::randomForest(Xin,yin,keep.inbag = TRUE,maxnodes = 25,ntree
  samples=modelest$inbag

  predaux=rep(0,ncol(samples))
  for(k in 1:ncol(samples)){
    saux=samples[,k]
    sboot=c()
    for(i in 1:length(saux)){
      sboot=c(sboot,rep(i,saux[i]))
    }
    xaux=Xin[sboot,]
    yaux=yin[sboot]
    tr=randomForest::getTree(modelest,k)
    selected=unique(tr[,3])
    selected=sort(selected[selected>0])
    modelols=lm(yaux~xaux[,selected]+dummy[sboot])
    cols=coef(modelols)
    cols[is.na(cols)]=0
    predaux[k]=c(1,Xout[selected],0)%*%cols
  }

  forecast=mean(predaux)

  return(list(forecast=forecast))
}

# adaLASSO/RF model
runadalassorf=function(ind,df,variable,horizon){
  prep_data = dataprep(ind,df,variable,horizon)
  Xin = prep_data$Xin
  yin = prep_data$yin
  Xout = prep_data$Xout

  lasso = HDeconometrics::ic.glmnet(Xin,yin)
  classo = coef(lasso)
  penalty = (abs(classo[-1])+1/sqrt(length(yin)))^(-1)
  adalasso = HDeconometrics::ic.glmnet(Xin,yin, penalty.factor = penalty)

  selected=which(adalasso$coef[-1]!=0)
  modelest=randomForest::randomForest(Xin[,selected],yin)
  forecast=predict(modelest,Xout[selected])

  return(list(forecast=forecast))
```

```
}

# Bagging model
runbagging=function(ind,df,variable,horizon){
  prep_data = dataprep(ind,df,variable,horizon)
  Xin = prep_data$Xin
  yin = prep_data$yin
  Xout = prep_data$Xout



  modelest=bagging(Xin,yin,R=100,l=5,pre.testing = "group-joint")
  forecast = predict(modelest,Xout)

  ## outputs
  nselect=modelest$coefficients
  nselect[nselect!=0]=1
  nselect[is.na(nselect)]=0
  nselect=colSums(nselect)

  outputs = list(nselect = nselect)

  return(list(forecast=forecast, outputs = outputs))
}

# Complete Subset Regression (CSR) model
runcsr=function(ind,df,variable,horizon){
  prep_data = dataprep(ind,df,variable,horizon)
  Xin = prep_data$Xin
  yin = prep_data$yin
  Xout = prep_data$Xout

  indice = which(colnames(df)==variable)
  f.seq=seq(indice,ncol(Xin)-1,ncol(df)+4)
  modelest=csr(Xin,yin,fixed.controls =c(f.seq,ncol(Xin)))
  forecast = predict(modelest,Xout)

  ## outputs
  nselect=modelest$coefficients
  nselect[nselect!=0]=1
  nselect[is.na(nselect)]=0
  nselect=colSums(nselect)

  outputs = list(nselect = nselect)

  return(list(forecast=forecast, outputs = outputs))
}

# factor model
runfact=function(ind,df,variable,horizon){
  prep_data = dataprep(ind,df,variable,horizon, factonly = TRUE, add_dummy = FALSE)
  Xin = prep_data$Xin
  yin = prep_data$yin
  Xout = prep_data$Xout
  dummy = prep_data$dummy

  bb=Inf
  for(i in seq(5,20,5)){
    m=lm(yin~Xin[,1:i]+dummy)
    crit=BIC(m)
    if(crit<bb){
      bb=crit
      modelest=m
      f.coef=coef(modelest)
      coefdum=f.coef[length(f.coef)]
      f.coef=f.coef[-length(f.coef)]
    }
  }
  coef=rep(0,ncol(Xin)+1)
  coef[1:length(f.coef)]=f.coef
  coef=c(coef,coefdum)
  coef[is.na(coef)]=0

  forecast=(cbind(1,Xout,0)%*%coef)[1]

  ## outputs
  outputs = list(coef = coef)

  return(list(forecast=forecast, outputs = outputs))
}

# Target factors model
runtfact=function(ind,df,variable,horizon){

  dfaux = df[ind,]
  if("2008-11-01" %in% rownames(dfaux)){
    if(variable %in% c("CPI","PCE")){
      dummy=rep(0,nrow(dfaux))
      intervention=which(rownames(dfaux)=="2008-11-01")
      dummy[intervention]=1
    }else{dummy = rep(0,nrow(dfaux))
    }
  }else{
    dummy = rep(0,nrow(dfaux))
  }

  index = which(colnames(dfaux)==variable)
```

```
mat = cbind(embed(dfaux[,variable],5),tail(dummy,nrow(dfaux)-4),tail(dfaux[,-inde
pretest=tfaux(mat,pre.testing="individual",fixed.controls = 1:4)[-c(1:6)]

pretest[pretest!=0]=1
aux = rep(0,ncol(dfaux))
aux[index] = 1
aux[-index] = pretest
selected=which(pretest==1)
dfreduced = df[,selected]

prep_data = dataprep(ind,dfreduced,variable,horizon, add_dummy = FALSE, factonly
Xin = prep_data$Xin
yin = prep_data$yin
Xout = prep_data$Xout
dummy = prep_data$dummy

bb=Inf
for(i in seq(5,20,5)){
  m=lm(yin~Xin[,1:i]+dummy)
  crit=BIC(m)
  if(crit<bb){
    bb=crit
    modelest=m
    f.coef=coef(modelest)
    coefdum=f.coef[length(f.coef)]
    f.coef=f.coef[-length(f.coef)]
  }
}
coef=rep(0,ncol(Xin)+1)
coef[1:length(f.coef)]=f.coef
coef=c(coef,coefdum)
coef[is.na(coef)]=0
```

```
  forecast=(cbind(1,Xout,0)%*%coef)[1]

  ## outputs
  outputs = list(coef = coef)

  return(list(forecast=forecast, outputs = outputs))
}

# build accumulated inflation
accumulate_model = function(forecasts){

  acc3 = c(rep(NA,2),sapply(1:(nrow(forecasts)-2), function(x){
    prod(1+diag(forecasts[x:(x+2),1:3]))-1
  }))
  acc6 = c(rep(NA,5),sapply(1:(nrow(forecasts)-5), function(x){
    prod(1+diag(forecasts[x:(x+5),1:6]))-1
  }))
  acc12 = c(rep(NA,11),sapply(1:(nrow(forecasts)-11), function(x){
    prod(1+diag(forecasts[x:(x+11),1:12]))-1
  }))

  forecasts = cbind(forecasts,acc3,acc6,acc12)
  colnames(forecasts) = c(paste("t+",1:12,sep = ""),"acc3","acc6","acc12")

  return(forecasts)

}

# function for LASSO, ElNet, Ridge
ic.glmnet = function (x, y, crit = c("bic", "aic", "aicc",
                                     "hqc"), alpha = 1, ...)
```

```
{
  if (is.matrix(x) == FALSE) {
    x = as.matrix(x)
  }
  if (is.vector(y) == FALSE) {
    y = as.vector(y)
  }
  crit = match.arg(crit)
  n = length(y)
  model = glmnet(x = x, y = y, alpha = alpha,...)
  coef = coef(model)
  lambda = model$lambda
  df = model$df
  yhat = cbind(1, x) %*% coef
  residuals = (y - yhat)
  mse = colMeans(residuals^2)
  sse = colSums(residuals^2)
  nvar = df + 1
  bic = n * log(mse) + nvar * log(n)
  aic = n * log(mse) + 2 * nvar
  aicc = aic + (2 * nvar * (nvar + 1))/(n - nvar - 1)
  hqc = n * log(mse) + 2 * nvar * log(log(n))
  sst = (n - 1) * var(y)
  r2 = 1 - (sse/sst)
  adjr2 = (1 - (1 - r2) * (n - 1)/(nrow(x) - nvar - 1))
  crit = switch(crit, bic = bic, aic = aic, aicc = aicc, hqc = hqc)
  selected = best.model = which(crit == min(crit))
  ic = c(bic = bic[selected], aic = aic[selected], aicc = aicc[selected],
         hqc = hqc[selected])
  result = list(coefficients = coef[, selected], ic = ic, lambda = lambda[selected]
                nvar = nvar[selected], glmnet = model, residuals = residuals[,
                                                                           select
                df = df, call = match.call())
  class(result) = "ic.glmnet"
  return(result)
}

tfaux=function (mat, pre.testing = c("group-joint","joint","individual"), fixed.con
                t.stat = 1.96,ngroups=10)
{
  pre.testing=match.arg(pre.testing)
  y = mat[, 1]
  X = mat[, -1]
  if (pre.testing == "joint") {
    if (nrow(X) < ncol(X)) {
      stop("Error:␣Type␣=␣joint␣is␣only␣for␣data␣with␣more␣observations␣than␣variab
    }
    m1 = lm(y ~ X)
    t1 = summary(m1)$coefficients[-1, 3]
    s1 = which(abs(t1) > t.stat)
    if (length(s1) == 0) {
      stop("Error:␣The␣pre-testing␣excluded␣all␣variables",
           "/n")
    }
  }
  if (pre.testing == "group-joint") {

    N=ncol(X)
    n=ceiling(N/ngroups)
    varind=1:N
    t1=rep(NA,N)
    for(i in 1:ngroups){
      selected=sample(order(varind),min(n,length(varind)),replace = FALSE)
      X0=X[,varind[selected]]
      m1=lm(y~X0)

      t0=rep(0,length(selected))
      aux=which(is.na(coef(m1)[-1]))
      t = summary(m1)$coefficients[-1, 3]
      if(length(aux)==0){
        t0=t
      }else{
        t0[-aux]=t
      }

      t1[varind[selected]]=t0
      varind=varind[-selected]
    }

    s1 = which(abs(t1) > t.stat)
    if (length(s1) == 0) {
      stop("Error:␣The␣pre-testing␣excluded␣all␣variables",
           "/n")
    }
  }
  if (pre.testing == "individual") {
    if (length(fixed.controls) > 0) {
      w = X[, fixed.controls]
      nonw = setdiff(1:ncol(X), fixed.controls)
    }
    else {
      w = rep(0, nrow(X))
      nonw = 1:ncol(X)
    }
    store.t = rep(NA, ncol(X))
    store.t[fixed.controls] = Inf
    for (i in nonw) {
      m1 = lm(y ~ X[, i] + w)
      t1 = summary(m1)$coefficients[2, 3]
      store.t[i] = t1
    }
    s1 = which(abs(store.t) > t.stat)
  }
  if (length(s1) > nrow(X)) {
    stop("Error:␣The␣pre-testing␣was␣not␣able␣to␣reduce␣the␣dimension␣to␣N<T")
  }
  m2 = lm(y ~ X[, s1])
  final.coef = rep(0, ncol(X))
  final.coef[s1] = coef(m2)[-1]
  names(final.coef) = colnames(X)
  final.coef = c(coef(m2)[1], final.coef)
  return(final.coef)
}
```

5.1.6 rolling_window.R

```
## rolling window function ##

# inputs
#   - fn: function applied for the forecast - from functions.R script
#   - df: dataset
#   - nwindow: number of windows (1 by default)
#   - horizon: time horizon for the forecast
#   - variable: variable to predict (y)
#
# outputs
#   - forecast: predicted inflation value made by the model
#   - outputs: parameters of the model#
```

```
rolling_window=function(fn,df,nwindow=1,horizon,variable,...){

  ind = 1:nrow(df)
  window_size = nrow(df)-nwindow
  indmat = matrix(NA,window_size,nwindow)
  indmat[1,] = 1:ncol(indmat)
  for(i in 2:nrow(indmat)){
    indmat[i,]=indmat[i-1,]+1
  }

  rw = apply(indmat,2,fn,df=df,horizon=horizon,variable=variable,...)
  # estimation of the model on each column of indmat
  forecast = unlist(lapply(rw,function(x)x$forecast))
  outputs = lapply(rw,function(x)x$outputs)
  return(list(forecast=forecast, outputs=outputs))

}
```

5.2 Extension

The code for the extension is very similar to the replication one, except for the first part of data preparation and small details in the other three parts (number of windows set to 180, name of inflation variable changed to "CPI", among others). Hence we reproduce only the first part of the code here, and refer to the Replication codes for the following 3 parts. The scripts functions.R and rolling_window.R are the same as above.

5.2.1 Data preparation

```
library(fbi)
library(tidyverse)
library(TTR)
library("readxl")
library(lubridate)

# set working directory
setwd(dirname(rstudioapi::getSourceEditorContext()$path))

start_date = "1985-01-01"
end_date = "2020-12-01"

# 1. loading data from excel (downloaded from FRED monthly data for France)

data <- read_excel("data/data_France.xls", sheet = 2)
data$DATE <- ymd(data$DATE)
data <- data.frame(data)



# we keep only variables with all observations in the sample period
data = data[data$DATE >= start_date, ]
data = data[data$DATE <= end_date, ]
data = data[ , apply(data, 2, function(x) !any(is.na(x)))]

# total : 86 variables
# 1985 - 2020 : 63 variables

# 2. data cleaning

# stationnarization of the data: simple differences (except for inflation : already
data_transform = data[, names(data) != "CPI"]

for (i1 in 2:length(data_transform))
{
  diffy <- diff(as.numeric(data_transform[,i1]))
  data_transform[,i1] <- c(diffy, NA)
}

# add inflation data
inflation = data[, names(data) %in% c("DATE", "CPI")]
inflation$CPI <- as.numeric(inflation$CPI)
inflation$CPI <- inflation$CPI/100

data = merge(x = data_transform, y = inflation, by = "DATE", all = TRUE)

names(data)[names(data) == "DATE"] <- "date"
names(data)

# set dates as index
# data_new <- data[,-1]
# rownames(data_new) <- data[,1]
# data = data_new

save(data,file = "data/data.rda")
```

5.2.2 RW and out-of-sample inflation

See above.

5.2.3 Building of forecasts with models

See above.

5.2.4 Comparison of models performances

See above.