ERASMUS SCHOOL OF ECONOMICS

BACHELOR THESIS BSC2 ECONOMETRICS / ECONOMICS

FEB63008-18

The Congo Case: Forecasting GDP Growth with Factor Models and Machine Learning Methods

Author:
Dominik Damjakob

Student Number:
423328

Supervisor:
Prof. dr. P.H.B.F. Franses

Second Assessor:
dr. A.M. Schnucker

Date final version: July 5, 2019

Abstract

This study uses static factor methods to forecast the annual GDP growth of the Democratic Republic of Congo. In a three step approach, factors are computed from a data set of growth variables for other African countries and international variables via sparse and non-sparse Principal Component Analysis. Thereafter, machine learning techniques in the form of Elastic Net and Bayesian Model Averaging are employed to forecast residuals of an autoregressive model. It is found that the forecasts do not outperform estimations of a simple autoregressive model. Further, it is detected that the Congolese data suffer from structural breaks and that models taking such breaks into account gain improvements in predictive performance over comparable autoregressive models.

Contents

1. Introduction

Modern macroeconomic data are often characterized by two properties: first, they are limited in time, as credible data only exist for recent years for many variables. Second, there is an abundance of variables for more recent years. Therefore, it has become a common challenge that the number of variables for a forecasting problem is similar or larger than the number of observations. Classical Econometric models, such as Ordinary Least Squares Regressions, are not constructed for such data sets. This caused the creation of parsimonious estimation methods, which aim to reduce the number of explanatory variables used. With such methods, estimation and forecasting problems can be solved for a large number of variables in a macroeconomic setting.

Among other studies, Kim and Swanson (2018) provide an extensive comparison of different parsimonious methods, all following a three step approach: First factor methods are utilized to extract common components from a data set. Second, an autoregressive (AR) model is fitted and the factors estimate its residuals in a third step with machine learning methods. From the methods presented by Kim and Swanson (2013), this paper chooses Principal Component Analysis (PCA) and Sparse Principal Component Analysis (SPCA) for the factor models, while the Elastic Net and Bayesian Model Averaging (BMA) are picked for the machine learning tools.

With its significant impact on most industrial and financial sectors as well as the political and socio-economic well-being of a population, one variable commonly forecasted by parsimonious frameworks is the Gross Domestic Product (GDP) growth of a country. Yet, current literature focuses on estimates for industrialized countries with a variety of explanatory variables available. An economic region for which, according to my knowledge, no GDP forecasts were computed with parsimonious methods for most African countries. Due to insufficient statistical infrastructure, GDP growth forecasts for African countries are often less accurate than for Western regions. This especially applies to the Democratic Republic of Congo (DRC), one of the largest countries in Sub-Saharan Africa. As the DRC is a diverse and populous state, uniquely covering the wide variety of both problems and opportunities of Sub-Saharan Africa, GDP forecasts for the DRC are of special interest. Further, with its positioning in the geographic heart of Africa it is susceptible to shocks from different African economic regions, which have potential predictive power for the DRC's GDP growth.

Macroeconomic data such as GDP growth do not provide a high frequency and are thus

generated over a long time period. As they are additionally subject to a variety of effects and potential shifts of the political and socio-economic background, such data are often subject to structural parameter breaks, changing parameter estimates and diluting forecasting accuracy. Hence, identification of potential structural breaks is important when calculating reasonable forecasting estimates in a parsimonious framework.

The above considerations highlight the possibility of potential forecasting improvements and the relevance of structural breaks. Hence, this paper investigates the following two research questions:

1. *Can GDP growth variables from African countries improve forecasting performance for the Democratic Republic of Congo's GDP growth as compared to current IMF forecasts or autoregressive models, if a combination of Factor Analysis methods (sparse and non-sparse Principal Component Analysis) mixed with Machine Learning techniques (Elastic Net and Bayesian Model Averaging) is applied?*

2. *Are such GDP forecasts affected by structural parameter breaks and do forecasts that take potential breaks into account outperform the previously described models?*

The first question is academically relevant, as it applies contemporary machine learning methods to a new problem, namely the DRC's GDP growth. This also allows for testing state of the art machine learning models such as the Elastic Net and BMA in combination with factor models in the form of PCA and SPCA and their applicability to other forecasting problems. From an Economic perspective, the estimation can pinpoint specific drivers of DRC's GDP growth, extending Economic knowledge in an African framework. The second question is important as potential structural breaks are influential not only for this study, but might be shared by other variables concerning the DRC. Thus, findings can be used for data selection by future studies. Further, identification of structural breaks can clarify macroeconomic, societal and political issues that caused such a change.

From an industry perspective, GDP forecasts are used in various settings. This analysis can show whether specific machine learning methods that utilize GDP growth data from other countries improve the predictive accuracy of current GDP forecasts. As the methods are applied to the DRC, which suffers from more inaccurate forecasts, this effect can be even more notable.

The findings of the paper show that overall, the forecasting methods do not provide an improvement in forecasting accuracy as compared to an AR model, though they outperform the IMF forecasts. Further, structural breaks are identified for different forecasting horizons in 1975 and around 2000, and factor and machine learning models that consider such parameter breaks outperform AR forecasts over the same data set.

The rest of the paper proceeds as follows: Section 2 provides an overview of the DRC, Section 3 discusses previous academic GDP growth and forecasting literature and Section 4 gives insights into the data. Afterwards, Section 5 discusses the relevant methodology and Section 6 explains the results in detail and answers the research questions, including a simulation to evaluate the forecasting performance of the chosen models and a description of the factor model results. Lastly, Section 7 concludes.

2. Country Review

The DRC, between 1971 and 1997 also referred to as Zaire, is a state in Central Africa. After its status as a Belgian colony, first as a property of King Leopold II and from 1908 onwards in the hands of the Belgian government, it gained independence in 1960 (Nzongola-Ntalaja, 2002). In 1965, Mobutu Sese Seko came into power and established a cleptocratic dictatorship, deteriorating the economy (Haskin, 2005). This dictatorship came to an end with worsening economic conditions and the start of the two Congo Wars (Turner, 2007). In 1997, Mobutu was overturned by Laurent-Désiré Kabila in the first Congo War that lasted from 1996 until 1997. After a short peace period, the former winner coalition broke, resulting in the second Congo War (1998 - 2003). This war was characterized by the involvement of other African countries, such as Rwanda, Burundi, Angola, Namibia and Simbabwe, as well as a depletion of the DRC's natural resources and the necessary infrastructure by all affiliated parties (Montague, 2002; Nest, Grignon, & Kisangani, 2006). The second Congo war ended with a peace process and the signing of a treaty in 2003, followed by democratic elections in 2006. Yet, the DRC remains in political turmoil, signified by the Kivu conflict in Eastern Congo which started in 2004.

Today, the DRC is still stuck in a politically troubled position. The national government lacks power over the eastern provinces, which are disputed territories between different rebel groups and warlords. In fact, due to the weakness of the central government it is often claimed that the DRC is a failed state (Trefon, 2011). Further, the DRC is still far from establishing

a functional democratic system. On the Economist's 2018 Democracy Index it ranks 165th, only placing above North Korea and Syria (The Economist Intelligence Unit, 2019), while the Reporters without Borders' World Press Freedom Index (Reporters Without Borders, 2019) holds it on position 154 out of 180 countries. With population estimates ranging between 81 and 98 million inhabitants (for data sources, the reader is referred to Table 8 in the Appendix), it is the fourth largest African country. Its population growth rate of 3.3% in 2017 compares to an overall Sub-Saharan African growth rate of 2.7%, indicating that the DRC's population share will increase in the future.

On the United Nations' 2018 Human Development Index (United Nations Development Programme, 2018) the DRC is listed at position 176 out of 189 countries with a score of 0.457. Drivers for this positioning are a low life expectancy at birth (60 years) and few expected years of schooling (9.8). Additionally, the DRC had a purchasing price power adjusted GDP per capita of only $889 in 2017, which is less than a fourth of the figure for all of Sub-Saharan Africa with $3,838. The DRC's economic situation is also highlighted by other key figures: In 2012, 76.6% of its population lived below the absolute poverty threshold ($1.90 per day in 2011 purchasing price power) compared to 43.7% for sub-Saharan Africa in total while these proportions were 91% vs. 68.8% for a threshold of $3.20 per day.

Despite its large population, the DRC contributed less than 2.5% to the Sub-Saharan African GDP in 2017 with a GDP of $37.6 billion (and $1,671 billion for all of Sub-Saharan Africa). However, these numbers do not provide an accountable estimate of the DRC's true economic power, as it has a large informal economic sector (Adoho & Doumbia, 2018). Its economy is mainly based on agriculture with 80 million hectares of arable land and the production of natural resources. Especially the Eastern provinces provide an abundance of minerals such as diamonds, gold, copper and coltan. The country is further rich in energy resources with oil (mainly located in western Congo) and natural gas. A map of the DRC's natural resources is given in the Appendix in Figure 8. In the agricultural sector, especially the production of natural rubber and coffee is important (World Atlas).

The DRC's economy and political structure are in parts shaped by its geography; the centre of the large country (the DRC is Africa's second biggest state) is occupied by the Congo Basin, home to the world's second biggest rain forest and surrounded by groups of mountains (World Atlas). This basin was created by the Congo river, which is Africa's second longest river and

of highest importance for the DRC's infrastructure, as it allows for shipment of goods. Access to the sea is only granted by a small area in the West with the DRC's largest international harbour, Matadi. As notable parts of the DRC's infrastructure, especially its streets and train grids, dilapidated during the dictatorship of Mobutu and the Congo Wars, this complicates trade across the country, strengthening the separation between eastern and western areas of the DRC. A graphical proof for these sparse links can be seen in Figure 7 in the Appendix, which shows the DRC's national transport system. As a consequence, eastern areas of the DRC mainly trade with neighbouring countries and not the western sections, shipping their goods via harbours at the Eastern African coast (Foster & Benitez, 2011). Further, due to the bad state of infrastructure, the DRC often allocates mining rights with the obligation to construct the necessary infrastructure around the mines to countries such as China (Kabemba, 2016). Lastly, the DRC's most important trade partner is China with a share of 45% of Congo's exports and 21% of its imports in 2017 (MIT). An overview of the DRCs' trade statistics can be found in the Appendix, with Figure 9 showing its main export goods and Figure 10 listing its main trade partners.

3. Literature Review

Economic growth has long been a focus of theoretical academic research. For example, Solow (1956) and Swan (1956) initiated a long-run growth model in which capital per capita converges to a steady-state level and GDP growth is driven by exogenous parameters, such as the population growth rate and technical progress. Lucas Jr (1988) and Romer (1986) internalize the technological progress factor in the Endogenous Growth model, focusing on the improvements in human capital that generate economic growth with increasing returns-to-scale. A consequence is the postulation of conditional convergence, according to which countries with lower economic starting levels, such as the DRC, can catch up with more developed economies by investing in non-human and human capital.

Barro (1996) analyzes large cross-country data sets, finding strong support for the conditional convergence hypothesis subject to variables such as schooling, life expectancy and the rule of law. Yet, for most African economies, this catch-up phase did not occur yet. Sachs and Warner (1997) argue that this is due to trade policy fallacies and geographical factors. Further, Sachs and Warner (1995) show that economies with large ratios of natural resource exports, such as the DRC, experience weaker GDP growth even after controlling for other factors, though these findings are not robust to other measures of natural resource abundance like resource reserves

(Stijns, 2005).

Recent academic literature made use of parsimonious methods in a variety of ways. Aang, Piazzesi, and Wei (2006) use a large set of variables to forecast the Yield curve. Eickmeier, Lemke, and Marcellino (2011) predict monetary variables and Gregory (1999) macroeconomic fluctuations between G7 countries. Besides pure financial and monetary variables studies, Kim and Swanson (2018) analyze the forecasting power of parsimonious estimators on different macroeconomic variables, including the GDP growth of the USA. Kim (2018) follows a similar approach for South Korea, while Artis and Banerjee (2005) focus on the United Kingdom. All three studies detect that sophisticated econometric parsimonious frameworks can increase predictive accuracy when forecasting GDP growth.

With regards to the chosen machine learning and factor methods, which are all entailed in Kim and Swanson (2013), PCA and SPCA are applied in a variety of forecasting settings. For example, Skittides and Früh (2014) employ PCA to forecast wind speed with past events and Ritchie, Holzinger, Li, Pendergrass, and Kim (2015) predict phenotype traits with genome data using SPCA. For the machine learning techniques, Schumacher (2010) uses the Elastic Net on a factor model to predict German GDP growth with international variables, while Bai and Ng (2008) use a similar framework with targeted predictors to forecast US inflation. BMA is used by Koop and Potter (2004) in a predictive factor model setting to forecast US GDP growth and inflation. The forecasting studies generally conclude that the machine learning techniques can boost forecasting performance for an appropriate choice of explanatory variables.

4. Data

Real annual GDP growth rates for 52 African countries from 1963 until 2017 are available at the World Bank data repository. As for 28 countries, data are not available throughout the entire time period, Franses and Vasilev (2019) provide an adjusted data set where for all countries except for South Sudan missing data were imputed with PC regressions based on demographic, production and financial variables. This study extends the data set with the 2017 World Bank annual real GDP growth data and observations for Somalia and Eritrea, for which no World Bank GDP growth data exists after 2011 (Eritrea) respectively 1990 (Somalia) are removed from the data set. Moreover, real annual GDP growth variables for the USA and the world in total are added to the data set to provide information on global GDP movement. Further, World Bank price estimates for the commodities Oil (Crude Oil, avg.), Gold, Copper, Rubber (SGP/MYS) and

Table 1: Descriptive statistics for the DRC's annual real percentage GDP growth

mean	median	max	min	std. dev.	25th percentile	75th percentile	Jarque-Bera
1.356	2.200	9.500	-13.500	5.216	-1.450	5.800	3.873

Coffee (Robusta) are reduced by US inflation to proxy a real price change and complete the set of explanatory variables. To compare forecasts with estimates by real institutions, the DRC's last available growth forecasts between 2013 and 2017 for the five consecutive year are downloaded from the IMF data repository.

The original data contain large deviations, such as observations more than four standard deviations off the mean, which can be caused by external effects, measurement errors, or, in the case of the African GDP variables, inaccurate estimates. Therefore, all variables were consecutively adjusted such that data more than three standard deviation off the mean are set at three standard deviations off the mean[1]. This gives a total of 57 variables for the time period from 1963 until 2017, given as annual percentage change. Especially for small changes, this is highly similar to the transformation in Kim and Swanson (2018), where changes of log values are used.

Descriptive Statistics for the DRC are given in Table 1, while those of the other 56 variables are listed in Table 9 in the Appendix. Table 1 indicates that, on average, the DRC experienced weak positive growth over the past 55 years with a mean GDP growth of 1.26%. Additionally, with a Jarque-Bera test statistic of 3.87, normality is not rejected at a 5% critical value.

Figure 1 plots the DRC's annual percentage real GDP growth over the sample period. Clearly, the figure can be roughly split into three periods: first, the DRC experiences substantial, yet highly fluctuating growth up to the the mid 1980s with a short period of economic decline towards the end of the 19070s. Subsequently, a strong recessive period follows during the 1990s, which are the last years of Mobutu's dictatorship and the beginning of the Congo Wars. This is followed by a third period after the end of the Congo Wars, which is shaped by a resurging

[1]The forecasting analysis was also performed for the original data set and the original data set with large outliers set to an average between the previous and the subsequent data point. Further, both of these specifications were first performed with all countries included in the data set and in a second analysis with Liberia and Equatorial Guinea removed from the data set, as the two countries exhibit excessive fluctuations. Lastly, as Franses and Janssens (2019) show that pre-whitening the explanatory variables can be important to get non-spurious principal components, the residuals of an AR(1) regression were used as explanatory variables for all previous specifications. However, none of the different data settings provide a substantial improvement in predictive power over all forecasting periods as compared to the method used in this paper.

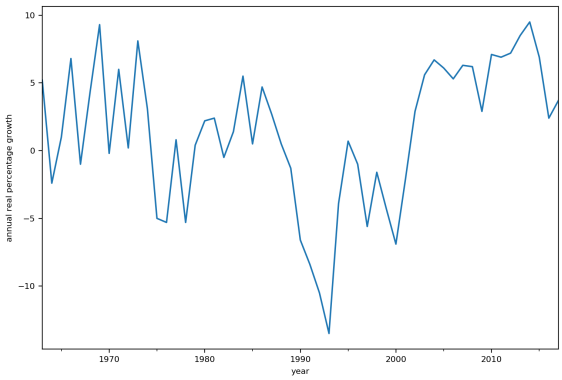


Figure 1: Real Percentage annual GDP growth of the DRC

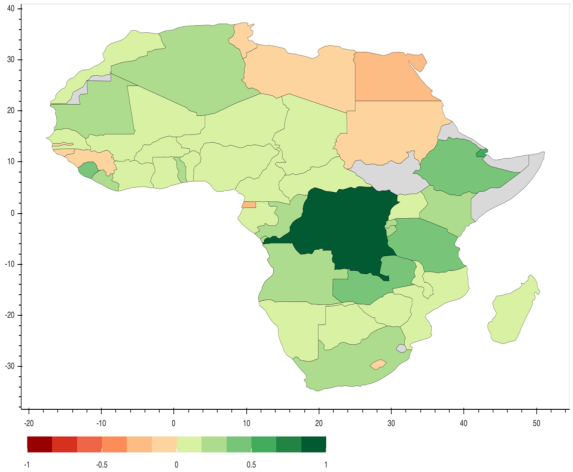


Figure 2: Correlation of annual real GDP growth between the African countries lagged by one year and the DRC. For grey countries / disputed regions, data are not included in the data set

economy and strong economic growth, though a sharp decline of GDP growth can be observed at the end of the sample.

A map indicating the correlation of all African in-sample countries lagged by 1 year with the DRC is shown in Figure 2, while Figures 11 and 12 in the Appendix show specifications without a lag and with a lag of five years. The figures indicate that the DRC's economic performance exhibits strongest correlation with Central and South African countries, though for a five year lag Botswana, Zimbabwe and Lesotho are negatively correlated. Especially neighbouring countries, such as Angola and Tanzania are moving in accordance with the DRC. Thus, it can be expected that factors for these countries will have the strongest predictive power on the DRC's economic growth. For an overview of the correlation between all 57 sample variables, the reader is referred to Figure 13 in the Appendix. No indication of sample covariances is given, as the subsequent statistical analysis uses standardized variables.

5. Methodology

5.1. Factor Models

The general framework of factor models, as defined in Kim and Swanson (2018) and Stock and Watson (2011, 2002) assumes that the variation of a dependent variable is caused by structural components called factors. To forecast the dependent variable it is thus necessary to employ a

two step process: first, factors F which are collected in the data set X in the form of common components are identified and in a second step they are used to compute forecasts of the dependent variable y. For a data set consisting of observations over T time periods and N explanatory variables, Stock and Watson (2002) define the factor model as

$$\boldsymbol{X} = \boldsymbol{F}\boldsymbol{\Lambda}' + \boldsymbol{e} \tag{1}$$

where $\boldsymbol{X}$ is the T x N-dimensional matrix of explanatory variable observations, $\boldsymbol{F}$ is a T x r-dimensional matrix of factors with $r < N$, $\boldsymbol{\Lambda}$ an N x r-dimensional matrix of factor loadings and $\boldsymbol{e}$ a T x N-dimensional vector of idiosyncratic components with a mean of 0. In this and all following sections, the N columns of $\boldsymbol{X}$ are standardized to have mean 0 and a variance of 1. As it is assumed that the number of underlying factors, r, is smaller than the number of explanatory variables, N, factor models allow for a dimensionality reduction of the explanatory variables, as performed in Swanson and Xiong (2018) and Kim and Swanson (2018). For a given time period $t = 1,...T$ and variable $j = 1,...,N$, the formulation becomes

$$X_{j,t} = F_t\Lambda_j + e_{j,t} \tag{2}$$

where $F_t = (f_{1,t},...,f_{r,t})$ is a vector consisting of the r factor terms at time period t and $\Lambda_j = (\lambda_{j,1},...,\lambda_{j,r})'$ holds the factor loading for observation j. With the factor formulations as in Equations 1 and 2, the h-step forecasting model of the dependent variable can be defined as

$$y_{t+h} = W_t\beta_w + F_t\beta_F + \epsilon_{t+h} \tag{3}$$

where Y_{t+h} is the dependent variable at time $t+h$, W_t is a $1xs$ dimensional vector of additional explanatory variables, such as lags of Y, β_w and β_F are the parameters of the forecasting equation and ϵ_{t+h} is a zero mean forecast error.

To estimate Equation 1, PCA and SPCA are employed in this paper. Their methodology is explained in Section 5.2. Afterwards, the machine learning methods described in 5.3 in the form of the Elastic Net and BMA are used to find sparse solutions for Equation 3. This is equivalent to specification Sp1 in Kim and Swanson (2018), and therefore this paper uses static factor methods as described by Chamberlain and Rothschild (1982), which assume that the factors

included in F_t do not include lags from periods before t. As this is equivalent to a dynamic factor model with lag loadings set to 0, the forecastig literature on dynamic forecasting models (Stock & Watson, 2006), which was adapted by Kim and Swanson (2018) can be applied nonetheless. Static factor models are chosen as they generate greater predictive power in Kim and Swanson (2013).

It is notable that such estimates for β_w and β_F do not cause the generated regressor problem, as their least squares estimates are $\sqrt{T}$ consistent and asymptotically normal if $\frac{\sqrt{T}}{N} \to 0$ (Bai & Ng, 2008), such that the estimates are consistent.

5.2. Factor Estimation Methods

Factor estimation methods are used to solve Equation 1 and provide an estimate for the number of underlying factors, r. In this paper, PCA and SPCA are applied. While PCA is the academic standard approach to factor estimation and also generates the Mean Squared Predictive Error (MSPE) best forecasts in Kim and Swanson (2013), SPCA is picked because it allows for a clearer identification of the factors and their drivers by penalizing non-zero parameters.

5.2.1. Principal Component Analysis

PCA is a well-studied transformation technique that extracts the orthogonal components responsible for the largest variability from a data set (Hastie, Tibshirani, & Friedman, 2009). For this purpose, recall that Equation 2 can be rewritten as

$$X_t = F_t \mathbf{\Lambda}' + e_t \tag{4}$$

with $X_t = (X_{1,t},...,X_{N,t})$ being the tth row of $\boldsymbol{X}$, $\mathbf{\Lambda}' = (\Lambda_1,...,\Lambda_N)$ being the factor matrix and $t = 1,...,T$. Though Kim and Swanson (2018) include the explanatory variable y in the set of explanatory variables, $\boldsymbol{X}$, the two variables are treated separately in this paper, because such a separation is suggested if N is large (Stock & Watson, 2006). Then, a variance estimator $\hat{V} = \frac{1}{T}\sum_{t=1}^{T} X_t X_t'$ is constructed and uncorrelated linear components Λ_j for $j = 1,..,T$ are found such that they maximize $\Lambda_j' \hat{V} \Lambda_j$. The orthogonality property of $\mathbf{\Lambda}$, which requires that $\mathbf{\Lambda}'\mathbf{\Lambda} = I_{rxr}$, where I_{rxr} is a r x r identity matrix, and consequently $\lambda_i'\lambda_i = 1$ and $\lambda_i'\lambda_j = 0$ for $i,j = 1,...,T$, $i \neq j$, serves as a normalization of the problem in order to identify the factor loadings.

Essentially, PCA models the problem of finding the r normalized eigenvectors with the largest eigenvalues. Yet, it can also be related to Singular Value Decomposition (SVD) (Wall, Rechtsteiner, & Rocha, 2003). In SVD, an mxn-dimensional matrix is factorized into $\boldsymbol{USV'}$, where $\boldsymbol{U}$ (mxm) and $\boldsymbol{V}$ (nxn) are orthonormal matrices and $\boldsymbol{S}$ is an mxn-dimensional rectangular diagonal matrix with the positive elements σ_i on the ith diagonal position. Wall et al. (2003) indicate that for the SVD $\boldsymbol{X} = \boldsymbol{USV'}$,

$$\boldsymbol{X'X} = \boldsymbol{VS'SV'} \tag{5}$$

such that V gives the eigenvectors and the elements σ_i the corresponding eigenvalues of the covariance matrix $\boldsymbol{X'X}$. This also implies for the principal components that

$$\boldsymbol{F} \quad = \quad \boldsymbol{XV} \quad = \quad \boldsymbol{USV'V} \quad = \quad \boldsymbol{US}. \tag{6}$$

Lastly, as the number of factors, r, is not known a priori, PCA is used to find all N eigenvectors. Thereafter, selection criteria as defined in Bai and Ng (2002) can be used to identify the number of factors, r. In this paper, this step will be performed by the information criterion P2, such that

$$IC(r) = log(\frac{tr(\boldsymbol{e'e})}{TN}) + k\frac{N+T}{NT}log(min(N,T)) \tag{7}$$

with $\boldsymbol{e} = \boldsymbol{X} - \boldsymbol{F\Lambda'}$ is minimized over $k = 1,...,15$.

5.2.2. Sparse Principal Component Analysis

In PCA, interpreting factors is not straightforward as the factor loadings are typically non-zero. Thus, Zou, Hastie, and Tibshirani (2006) define SPCA, which penalizes non-zero parameters. For this, they transform PCA to a regressive framework and define the Nxr-matrices $\boldsymbol{\Delta}$ and $\boldsymbol{\Lambda}$, an approximation of the factor loadings, for the optimization problem

$$(\hat{\boldsymbol{\Delta}}, \hat{\boldsymbol{\Lambda}}) = arg\min_{\boldsymbol{\Delta},\boldsymbol{\Lambda}} \left(\sum_{t=1}^{T} \|\boldsymbol{X_t} - \boldsymbol{\Delta\Lambda' X_t}\|^2 + \sum_{j=1}^{r} \eta_{1,j}\|\lambda_j\|_1 + \eta \sum_{j=1}^{r} \|\lambda_j\|^2 \right) \quad s.t. \quad \boldsymbol{\Delta'\Delta} = I_r \tag{8}$$

where $X_t = (X_{1,t}, X_{2,t}, ..., X_{N,t})$ is specified to be a column vector, λ_j is the jth column of $\boldsymbol{\Lambda}$, $|\lambda_j|_1 = \sum_{i=1}^{r} |\lambda_{j,i}|$ and $|\lambda_j|^2 = \sum_{i=1}^{r} \lambda_{j,i}^2$. This problem is computationally difficult to optimize, as it

is not jointly convex in $\boldsymbol{\Delta}$ and $\boldsymbol{\Lambda}$. However, for fixed values of $\hat{\boldsymbol{\Delta}}$ respectively $\hat{\boldsymbol{\Lambda}}$, the optimization with respect to the other parameter matrix follows a more standard procedure.

Therefore, Zou et al. (2006) suggest a two step procedure to optimize Equation 8. First, for fixed values of $\hat{\boldsymbol{\Delta}}$, an orthonormal matrix $\hat{\boldsymbol{\Delta}}^{+}$ can be defined such that $[\hat{\boldsymbol{\Delta}}, \hat{\boldsymbol{\Delta}}^{+}]$ is an orthonormal matrix and

$$\sum_{t=1}^{T} \|X_t - \hat{\boldsymbol{\Delta}}\hat{\boldsymbol{\Lambda}}' X_t\|^2 \quad = \quad \|\boldsymbol{X} - \boldsymbol{X}\hat{\boldsymbol{\Lambda}}\hat{\boldsymbol{\Delta}}'\|^2 \quad = \quad \|\boldsymbol{X}\hat{\boldsymbol{\Delta}}^{+}\|^2 + \sum_{j=1}^{r} \|\boldsymbol{X}\hat{\delta}_j - \boldsymbol{X}\lambda_j\|^2. \tag{9}$$

Now, for each loading i with $i = 1,...,r$, substitution of Equation 9 into Equation 8 with a deletion of the constant part of Equation 9 and a separation of the r independent optimizations changes the program to

$$\hat{\lambda_j} = arg\min_{j} \left(\|\boldsymbol{X}\hat{\delta}_j - \boldsymbol{X}\lambda_j\|^2 + \eta_{1,j}\|\lambda_j\|_1 + \eta\|\lambda_j\|^2 \right) \tag{10}$$

for $i = 1,...,r$, which can each be solved with the LARS-EN algorithm explained in Section 5.3.1.

For the second step, $\hat{\boldsymbol{\Lambda}}$ is fixed. Therefore, the two penalties in Equation 8, which only depend on values of Λ, can be deleted and the resulting problem is

$$\hat{\boldsymbol{\Delta}} \quad = \quad arg\min_{\boldsymbol{\Delta}} \sum_{t=1}^{T} \|X_t - \boldsymbol{\Delta}\hat{\boldsymbol{\Lambda}}' X_t\|2 \quad = \quad arg\min_{\boldsymbol{\Delta}} \|\boldsymbol{X} - \boldsymbol{X}\hat{\boldsymbol{\Lambda}}\boldsymbol{\Delta}'\|^2 \quad s.t. \quad \hat{\boldsymbol{\Delta}}'\hat{\boldsymbol{\Delta}} = \boldsymbol{I}_{rxr}. \tag{11}$$

As this problem no longer contains any penalty terms, it is easier to solve. Using the reduced rank Procustres rotation (Mardia & Bibby, 1979), which states that for an nxp-matrix M and an nxk-matrix N, the constrained minimization problem

$$\hat{\boldsymbol{A}} = arg\min_{\boldsymbol{A}} \|\boldsymbol{M} - \boldsymbol{N}\boldsymbol{A}'\|^2 \quad s.t. \quad \boldsymbol{A}'\boldsymbol{A} = \boldsymbol{I}_{kxk} \tag{12}$$

and the SVD $\boldsymbol{M}'\boldsymbol{N} = \boldsymbol{U}\boldsymbol{D}\boldsymbol{V}'$, $\hat{\boldsymbol{A}} = \boldsymbol{U}\boldsymbol{V}'$. Consequently, for the SVD $\boldsymbol{X}'\boldsymbol{X}\hat{\boldsymbol{\Lambda}} = \boldsymbol{U}\boldsymbol{D}\boldsymbol{V}'$, I get that $\hat{\boldsymbol{\Delta}} = \boldsymbol{U}\boldsymbol{V}'$.

The algorithm presented by Zou et al. (2006) starts by approximating $\hat{\boldsymbol{\Delta}}$ with the principal components $\boldsymbol{F}$ and then repeatedly computes estimates for $\hat{\boldsymbol{\Lambda}}$ and $\hat{\boldsymbol{\Delta}}$. The program terminates when the estimates are either sufficiently converging or a maximum number of iterations (50) has been reached. For the parameter r, the approximations yielded in 5.2.1 are used and only

the r eigenvalue-largest principal components are passed to the algorithm.

5.3. Robust Estimation Methods

Machine Learning Methods can be utilized to provide robust estimates of the β parameters in Equation 3. More precisely, as there are two pairs of explanatory variables, W_t and F_t, the methodology first estimates the relationship

$$y_{t+h} = W_t\beta_w + \epsilon^*_{t+h} \tag{13}$$

generating consistent estimates $\hat{\beta}_w$, and thereafter the methods of Elastic Net and BMA are used to compute estimates $\hat{\beta}_f$ from

$$\hat{\epsilon}^*_{t+h} = F_t\beta_F + \epsilon_{t+h} \tag{14}$$

where $\hat{\epsilon}^*_{t+h} = y_{t+h} - W_t\beta_w$. This approach follows the structure from Kim and Swanson (2018). In this section, two contemporary approaches are applied. The first is the Elastic Net, as it combines two often-employed shrinkage methods, the Lasso and the Ridge regression, and the scond is BMA, which is generally viewed to be one of the best model selection techniques (Fernandez, Ley, & Steel, 2001b; Ravazzolo, Paap, Van Dijk, & Franses, 2008).

5.3.1. Elastic Net

In an Elastic Net, a regularized regression framework is used to estimate parameters, in the case of this paper, to estimate β_f. To do so, it combines the linear L1 penalty of a Lasso (Tibshirani, 1996) and the squared L2 penalty of a Ridge regression (Hoerl & Kennard, 1970) to the regularized Naive Elastic Net expression

$$\hat{\beta}^{NEN} = arg\min_{\beta}\left(|Y - X\beta|^2 + \eta_1|\beta|_1 + \eta_2|\beta|^2\right) \tag{15}$$

where $|\beta|_1 = \sum_{j=1}^{r}|\beta_j|$, $|\beta|^2 = \sum_{j=1}^{r}\beta_j^2$ and X and Y are the standard exogenous respectively dependent regression variables. (Zou & Hastie, 2005). For rewritten matrices with $\boldsymbol{X}^+ = (1+\eta_2)^{-\frac{1}{2}}\begin{pmatrix}\boldsymbol{X}\\ \sqrt{\eta_2}\boldsymbol{I_n}\end{pmatrix}$ and $\boldsymbol{Y}^+ = \begin{pmatrix}\boldsymbol{Y}\\ \boldsymbol{0_N}\end{pmatrix}$, the problem becomes the Lasso-like expression

$$\hat{\beta}^+ = arg\min_{\beta^+}\left(|Y^+ - X^+\beta|^2 + \gamma|\beta|_1\right) \tag{16}$$

with $\gamma = \frac{\eta_1}{1+\eta_2}$ and $\beta^+ = \sqrt{1+\eta_2}\beta$, such that $\beta^{NEN} = \frac{1}{\sqrt{1+\eta_2}}\beta$. This problem contains only one L1 penalty and can thus be solved more efficiently. Zou and Hastie (2005) notice that the Naive Elastic Net underestimates β due to the two separate penalization terms. Therefore, they define the Elastic Net estimator to be

$$\hat{\beta}^{EN} = (1+\eta_2)\hat{\beta}^{NEN} \quad \text{such that} \quad \hat{\beta}^{EN} = \sqrt{(1+\eta_2)}\hat{\beta}^{+}$$

To solve Equation 16, Zou and Hastie (2005) extend the Least Angle Regression (LARS) algorithm (Efron, Hastie, Johnstone, Tibshirani, et al., 2004) into a LARS-EN version. The LARS algorithm allows for an efficient solution of regression problems with high-dimensional data, that is, a regression problem with a large number of explanatory variables, by increasing the parameter estimate into the direction of the largest correlation between explanatory variables and the current regression residual up to the point where a different combination of explanatory variables shows a larger correlation. The LARS-EN algorithm calls the LARS algorithm for fixed values of η_2 and optimizes Equation 16. Then, cross-validation (CV) (Hastie et al., 2009) is utilized to find the best values for η_1 and η_2. Zou and Hastie (2005) use 10-fold CV, which is explained in Section 5.4 and choose the values for η_1 and η_2 which result in the smallest CV error.

5.3.2. Bayesian Model Averaging

In Bayesian Model Averaging, it is assumed that the exact formulation of a forecasting Equation follows one out of Q different models. For possible models M_q with $q = 1,...,Q$, and random dependent variables y_{t+h}, this means that forecasts can be calculated as

$$E[y_{t+h}|Data] = \sum_{q=1}^{Q}\Big(E[y_{t+h}|Data, M_q]p(M_q|Data]\Big) \tag{17}$$

due to the law of conditional probability (Chipman et al., 2001; Hoeting, Madigan, Raftery, & Volinsky, 1999). Kim and Swanson (2018) implement BMA in a factor model, assuming that models M_q are all linear regression problems with conjugate priors - that is, their posterior distribution belongs to the same distribution family as their prior probability - and differ based on their inclusion of explanatory variables only. As each potential explanatory variable can either

be included or excluded, this gives, for r potential explanatory variables, a total of $Q = 2^r$ model candidates. Due to the large number of candidate models, this problem is complex by nature, but Koop and Potter (2004) provide an algorithm to solve it. Instead of computing a forecast for each of the Q potential forecasting models, their algorithm draws models from $p(M_q|Data)$, simplifying the complexity of the problem. To include the common variables W_F, such as an intercept or lags of y, these are integrated out using a noninformative prior with

$$y^*_{t+h} = \beta^* F^*_t + \epsilon^*_t \tag{18}$$

where $y^*_{t+h} = [I_T - W_t(W_t'W_t)^{-1}W_t']Y_{t+h}$, $F^*_t = [I_T - W_t(W_t'W_t)^{-1}W_t']\hat{F}_t$ and ϵ_{t+h} is normally distributed with mean 0 and constant variance σ^2. The prior distributions are then defined by $\alpha|\sigma^{-2} \sim N(\underline{\beta}, \sigma^2\underline{B})$ and $\sigma^{-2} \sim G(\underline{s}^{-2}, \underline{\nu})$, where $G(a,b)$ is the Gamma distribution with mean a and b degrees of freedom (Poirier, 1995, p. 100) and $\underline{\beta}$, $\underline{B}$, $\underline{s}^{-2}$ and $\underline{\nu}$ are hyperparameters. If γ is an indicator vector stating whether a variable is included (1) or excluded (0) from a model, sequential estimates for $p(\gamma|y,\sigma^2)$ and $p(\sigma^2|y,\gamma)$ can then be computed with Gibbs Samplers if σ^2 is assumed to be unknown. This requires orthogonal explanatory variables F_t, as fulfilled by PCA. For S models drawn and with $E(y_{t_1}|Data, M^{(s)})$ giving the predictive mean for models $M^{(S)}$,

$$\frac{1}{S}\sum_{s=1}^{S} E(y_{t+h}|Data, M^{(S)}) \rightarrow E(y_{t+h}|Data) \tag{19}$$

for $S \rightarrow \infty$, and averaged models estimates will converge to their true mean values.

To run the algorithm, it is necessary to specify the probability density functions for $p(\gamma|y,\sigma^2)$ and $p(y|\gamma,\sigma^2)$. While $p(y,\gamma,\sigma^2)$ is simply the marginal likelihood of the regression and $p(\sigma^2|y,\gamma)$ has the inverted-Gamma form of a linear regression model with the inclusion of explanatory variables as given by γ,

$$p(\gamma) = \prod_{j=1}^{r} \theta_j^{\gamma_j}(1-\theta_j)^{1-\gamma_j} \tag{20}$$

where θ_j, the prior probability that a potential explanatory variable gets included in the model, is set to $\frac{1}{2}$.

For the other four hyperparameters, values are selected in accordance with Fernandez, Ley, and Steel (2001a) and Kim and Swanson (2018). The Gamma function is assigned improper, noninformative parameters with $\underline{\nu} = 0$, such that s^{-2} is not part of the marginal likelihood or the

posterior function. For $\underline{\alpha}$, a vector of zeros with length r is used and $\underline{A} = (gZ'Z)^{-1}$, where two different values for g, namely $\frac{1}{T}$ (specification BMA0) and $\frac{1}{r^2}$ (BMA1) are chosen.

5.4. Forecast Errors

The employed techniques, especially the two robust estimation methods, are statistically non-trivial and therefore statistical inference of standard errors is not possible. Instead, CV (Hastie et al., 2009) can help in deriving an estimate for the forecast errors. In CV, observations are sequentially removed from the training sample to serve as a testing sample. The most common form of CV is k-fold CV, in which k observations form the testing sample and the remaining $T-k$ observations construct the training sample. For $k = 1$, this is also referred to as Leave-one-out CV. If $\hat{y}_t$ gives 1-step ahead estimate when only observation y_t forms the training sample, then the CV 1-step ahead forecast error can be computed as

$$\hat{e}_1^2 = \frac{1}{T}\sum_{t=1}^{T}(\hat{y}_t - y_t) \tag{21}$$

For a time series, such as the prevailing problem, CV does not yield consistent estimates as the dependent variables are correlated. Therefore, the CV training sample for y_t only includes observations up to $t-1$ and removes all observations later than t from the model (Hyndman & Athanasopoulos, 2018; Tashman, 2000). The h-step ahead forecast error can then be calculated as

$$\hat{e}_h^2 = \frac{1}{T-m+1}\sum_{t=1}^{T}(\hat{y}_t - y_t) \tag{22}$$

where m is the minimum number of observations required by the estimation and forecast estimates $\hat{y}_t$ are based on observations $1,..,t-h$. As out-of-sample h-step forecasts with $h > 1$ are computed in this research paper, which can miss up to the last $h-1$ observations, forecast errors for these estimates are also constructed based on observations $1,...,t-o$, where $o = 1,...,h$.

5.5. Structural Breaks

Though it it an assumption of factor models that the loadings are constant over time, Stock and Watson (2002) show that for small intertemporal instabilities the standard PCA estimator is consistent. However, for larger instabilities this changes and the number of included factors, r,

increases. Breitung and Eickmeier (2011) derive that for a single change of the factor loadings

$$y_{it} = F_t\mathbf{\Lambda}_1^{(1)} + \epsilon_{i,t} \quad \text{for } t = 1,...,T^* \quad \text{and} \quad y_{it} = F_t\mathbf{\Lambda}_1^{(2)} + \epsilon_{i,t} \quad \text{for } t = T^*+1,...,T \tag{23}$$

for $i = 1,...,N$, with $F_t = (f_{t,1}, f_{t,2}, ..., f_{t,r})$ and the i.i.d. idiosyncratic error variables $\epsilon_t = (\epsilon_{1,t}, ..., \epsilon_{N,t})$, the number of observed factors increases from r, its actual value, to $2r$. More generally, for k structural breaks in the sample, the expected number of observed factors increases to $(k+1)*r$. Breitung and Eickmeier (2011) provide a variety of break tests, but conclude that for samples with a small number of observations the Lagrange Multiplier (LM) is best. Assuming that the break happens simultaneously for all parameters in Equation 3 for a known break date, the test is defined by $s_i = TR_i^2$, where R_i^2 is the R^2 of

$$\hat{\epsilon}_{i,t} = \theta_i'\hat{F}_t + \kappa_i'W_t + \phi'\hat{F}_t^* + \psi'W_t^* + \tilde{\epsilon}_{i,t} \tag{24}$$

with $\hat{\epsilon}_{i,t} = y_{i,t} - \hat{\beta}_i'W_t - \hat{\lambda}_i'\hat{F}_t$, $\hat{f}_t$ is the factor estimate for the entire sample, W_t the additional parameters used for the AR model and

$$\hat{F}_t^* = \begin{cases} 0 & \text{for t = 1, ..., T*} \\ \hat{F}_t & \text{for t = T* + 1, ..., T} \end{cases} \qquad W_t^* = \begin{cases} 0 & \text{for t = 1, ..., T*} \\ W_t & \text{for t = T* + 1, ..., T} \end{cases}. \tag{25}$$

Under a set of standard assumptions, such as as Assumptions A-G in Bai and Perron (2003), independence of the factors and the $\epsilon_{i,t}$ error terms and if $T \to \infty$ and $N \to \infty$, s_i is $\chi^2(r)$ distributed. For the DRC, possible break dates are marked by political events, such as the end of the Mobutu dictatorship in 1997 and the official end of the DRC's civil war in 2003.

For an unknown break date, which is only assumed to lie within $[T\tau_0, T(1-\tau_0)]$ with $\tau_0 \in [0,1]$, the sup-LM statistic can be computed as

$$\kappa_{i,T}(\tau_0) = \sup_{\tau \in [\tau_0, 1-\tau 0]} (s_i^\tau) \tag{26}$$

where s_i^τ is the LM statistic for break point τ. If, additional to the previous assumptions, $\frac{T}{n} \to 0$ and

$$\frac{1}{T}\sum_{t=1}^{\tau T} F_t F_t' \xrightarrow{p} \tau\Sigma_f$$

$$\frac{1}{T}\sum_{t=1}^{\tau T} F_t \epsilon_{i,t} \implies \sigma_i \Sigma_f^{0.5} W_i(\tau)$$

where Σ_f is a positive definite matrix and $W_i(a)$ an r-dimensional standard Brownian motion, then

$$\kappa(\tau_0) = \sup_{\tau \in [\tau_0, 1-\tau 0]} \left(\frac{[(1) - W(\tau)]'[(1) - W(\tau)]}{\tau(1-\tau)} \right). \tag{27}$$

Critical values for this distribution are interfered from Andrews (1993).

5.6. Autoregressive Models

As mentioned in Section 5.1, a set of additional explanatory variables W_t is used. Following the methodology from Kim and Swanson (2018), this paper will define W_t such that it models an AR process of order p, with $W_t = (1, y_{t-1}, ..., y_{t-p})$, where p is chosen such that is maximizes the Bayesian Information Criterion under the assumption of normally distributed error terms (Heij, De Boer, Franses, Kloek, & Dijk, 2004)

$$BIC(p) = log(s_p^2) + p\frac{log(N)}{N} \tag{28}$$

where s_p^2 is the maximum likelihood estimator of the AR model's error variance. The AR model of order p is further used as a benchmark against which the combined factor and machine learning models are compared.

5.7. Simulations

To check for the potential improvements in predictive power of the factor methods as compared to forecasts generated with an AR model, a simulation study is applied. Similar to the Monte Carlo analysis' of Stock and Watson (2002) and Breitung and Eickmeier (2011), the following underlying Data Generating Process (DGP) is assumed:

$$f_{t,i} = \sum_{j=1}^{p} \alpha_{j,i} f_{t-j,i} + \epsilon_{factors,t,i} \tag{29}$$

$$y_t = F_t \beta + \epsilon_{1,t} \tag{30}$$

$$X_t = F_t \mathbf{\Gamma} + \epsilon_{2:N,t} \tag{31}$$

$$(1 - aL)\epsilon_{i,t} = (1 + b^2)v_{i,t} + bv_{i+1,t} + bv_{i-1,t} \tag{32}$$

$$v_{i,t} = \sigma_{i,t}\eta_{i,t} \tag{33}$$

$$\sigma^2_{i,t} = \delta_0 + \delta_1\sigma^2_{i,t-1} + \delta_1 v^2_{i,t-1} \tag{34}$$

where $\alpha_{j,i}$ gives the AR parameter for the jth lag of the ith factor, $F_t = (f_{t,1}, f_{t,2}, ..., f_{t,r})$ if there are r different factors in total and they are based on an AR(p) model each, β is a parameter vector generating the composition of the dependent variable, $\Gamma = (\gamma_1, \gamma_2, ..., \gamma_N)$ is a matrix holding the parameter vectors that generate the N explanatory variables and $\epsilon_{2:N,t} = (\epsilon_{2,t}, ..., \epsilon_{N,t})$. Additionally, values for the error term parameters a, b, δ_0 and δ_1, which are defined as in Stock and Watson (2002) are given in the Appendix. Also, the idiosyncratic components $\epsilon_{factors}$ and $\eta_{i,t}$ are all normally i.i.d. with mean zero and variance one.

While Equation 32 introduces correlated error terms into the simulation, Equations 33 and 34 simulate heteroscedasticity. Both specifications are important for a realistic simulation, as macroeconomic variables tend to exhibit cross-correlation and heteroscedasticity. The main difference between the above specification and Stock and Watson (2002) is that the factors are constructed based on an AR(p) process, while the explanatory and dependent variable are formed by factors from one factor only. This models the analyzed process, which first utilizes an AR regression model and then constructs static factor estimation methods on the residual.

The simulations are tested for $r = 1$ and $r = 5$. In both cases, all elements of $\mathbf{\Gamma}$ are standard normally distributed. For the AR process, three different specifications are analyzed and described in the Appendix.

Besides checking for the influence of the number of factors, r, and the number of lags in the factor DGP, p, simulations can also determine how the used framework operates if the number of observations, T, is small compared to the number of explanatory variables, N. With only 52 observations and 50 explanatory variables, such a problem also applies here and hence the outcome of the simulation study is important for the reliability of the results.

6. Results

6.1. Simulation Results

Simulations are modelled to mirror potential versions of the DRC forecasting experiment. For each different specification, 100 repetitions of five forecasts for a dependent variable are cre-

ated. The specifications differ by their number of observations, T and the forecasting period, which is h from Equation 3. Both of these are observable variables and for the DRC forecasting experiment, $T = 52$ and $h = 1, 2, 3, 4, 5$. Further, the specifications differ by observable characteristics of the underlying data generating process, namely the number of factors, the number of AR lags, the specifications of the AR processes, the correlation of error terms and potential heteroscedasticity of error terms. The simulations are performed for different input specifications, as outlined in the Appendix, and forecasts are created for an AR(p) model, the methods of an Elastic Net and the two BMA versions using PCs as an explanatory variable. No simulations for SPCA forecasts are created as their computation is complex due to the repetitive optimization with Elastic Nets and therefore not computationally feasible.

The simulations can not only clarify whether the employed methods provide accurate forecasts in comparison to an AR model, but also whether the methodology performs sufficiently well at filtering out the correct AR lag and the number of factors of the data generating process. Results for key statistics are summarized in Table 2 for the more realistic case of correlated, heteroscedastic error terms while Table 14 in the Appendix provides information about the specifications with uncorrelated, homoscedastic error terms. As AR configurations 1 and 2 model an AR(1) process while configuration 3 is an AR(3) process, Table 2 shows that the methodology performs well at identifying the correct number of lags, p. Further, Table 2 also indicates that the PCA selection criterion does reasonably well at selecting the correct number of PC factors for AR configurations 1 and 2, though the number of PCA factors is sometimes sharply overestimated. Yet, as the factors are subsequently used for robust machine learning techniques, this is not of too much concern. However, for AR configuration 3 the estimates are more off, overestimating the number of factors for specifications with $h = 1$ and a small number of observations, while underestimating the number of factors in other cases. With regards to the β estimates of the three machine learning models, Table 2 indicates that the Elastic Net is most restrictive at selecting parameters, but often underestimatss the true number of factors in the model. The two BMA models, on the contrary, typically use most factors for their estimation results, thus overestimating the number of parameters in many cases. Still, with concern to forecasting accuracy, the BMA1 optimization performs best for all specifications with $h = 1$, and many 5-period forecast specifications. In general, it appears that most gains in predictive power occur for shorter forecasting horizons, whereas the predictive power of the mixed models fac-

Table 2: Simulation Outcomes for different specifications of a factor model with correlated and heteroscedastic errors and 100 repetitions per specification

lags	T	# factors	AR Config	mean lags	mean PCA facs	mean non-0 β_is EN	mean non-0 β_is BMA0	mean non-0 β_is BMA1	Best model
1	50	3	Config. 1	1.404	3.486	3.288	3.486	3.486	BMA1
	50	3	Config. 2	1.528	3	1.388	3	3	BMA1
	50	3	Config. 3	3.304	14.798	8.608	14.79	14.548	BMA1
	50	5	Config. 1	1.534	7.318	5.956	7.318	7.314	BMA1
	50	5	Config. 2	1.4	5.054	2.97	5.054	5.054	BMA1
	50	5	Config. 3	2.904	19.546	8.54	19.538	18.802	BMA1
	100	3	Config. 1	1.362	4.738	4.268	4.738	4.738	BMA1
	100	3	Config. 2	1.166	3	1.772	3	3	BMA1
	100	3	Config. 3	3.402	15.156	10.718	15.134	14.994	BMA1
	100	5	Config. 1	1.21	8.764	7.238	8.764	8.762	BMA1
	100	5	Config. 2	1.218	5.124	4.142	5.124	5.124	BMA1
	100	5	Config. 3	3.024	19.62	10.136	19.552	18.782	BMA1
	400	3	Config. 1	2.11	10.582	7.58	10.55	10.582	BMA1
	400	3	Config. 2	1.048	3	2.442	3	3	BMA1
	400	3	Config. 3	3.866	1.442	0.39	1.146	1.136	BMA1
	400	5	Config. 1	1.294	11.456	8.754	11.372	11.452	BMA1
	400	5	Config. 2	1.032	5.742	4.87	5.736	5.742	BMA1
	400	5	Config. 3	3.234	1	0.062	1	1	BMA1
5	50	3	Config. 1	1.368	3.042	0.872	3.042	3.042	EN
	50	3	Config. 2	1.524	3	0.574	3	3	AR
	50	3	Config. 3	3.238	17.106	7.448	17.086	16.62	AR
	50	5	Config. 1	1.56	5.9	1.792	5.9	5.89	AR
	50	5	Config. 2	1.452	5.004	1.612	5.004	5.004	EN
	50	5	Config. 3	2.962	19.754	8.42	19.748	18.942	AR
	100	3	Config. 1	1.292	3.068	1.158	3.068	3.068	BMA1
	100	3	Config. 2	1.164	3	0.806	3	3	EN
	100	3	Config. 3	3.432	17.76	8.312	17.72	17.23	BMA1
	100	5	Config. 1	1.212	7.028	1.304	7.028	7.024	AR
	100	5	Config. 2	1.2	5	0.716	5	5	AR
	100	5	Config. 3	3.072	19.906	8.94	19.828	19	BMA1
	400	3	Config. 1	1.966	5.478	3.912	5.468	5.478	BMA1
	400	3	Config. 2	1.04	3	0.338	2.998	3	AR
	400	3	Config. 3	3.924	1.47	0.362	1.412	1.408	BMA0
	400	5	Config. 1	1.316	13.256	3.952	13.012	13.076	BMA1
	400	5	Config. 2	1.04	5	0.836	4.998	5	AR
	400	5	Config. 3	3.432	1	0.01	1	1	BMA1

tors partly dissipates for longer forecasting horizons. This suggests that PCA does not perform sufficiently well at identifying the correct factors for such factor models.

Overall, the results from Table 2 are also supported by Table 14 in the Appendix. Further, Table 15 underlines the strength of BMA1 forecasts, especially for 1-period forecast horizons, even though the BMA0 forecasts often reach comparable results.

6.2. (Sparse) Principal Components

From the correlation matrix of the 56 explanatory variables, sparse and non-sparse PCs are constructed to identify structural patterns within the data set. Integral information of the PC factor loadings is contained in the normed eigenvectors with an Euclidean length of one. Figure 3 shows a heatmap for each variables' individual contribution to the 27 PCs, which are the components that each account for at least 1% of the explained sample correlation. The factor loadings are based on data over the entire time period, to allow for a more accurate estimation. Overall, it can be seen that no variable, whether it is a country's GDP growth or a raw material price change, accounts for an abnormally large contribution to the size of a principal compo-

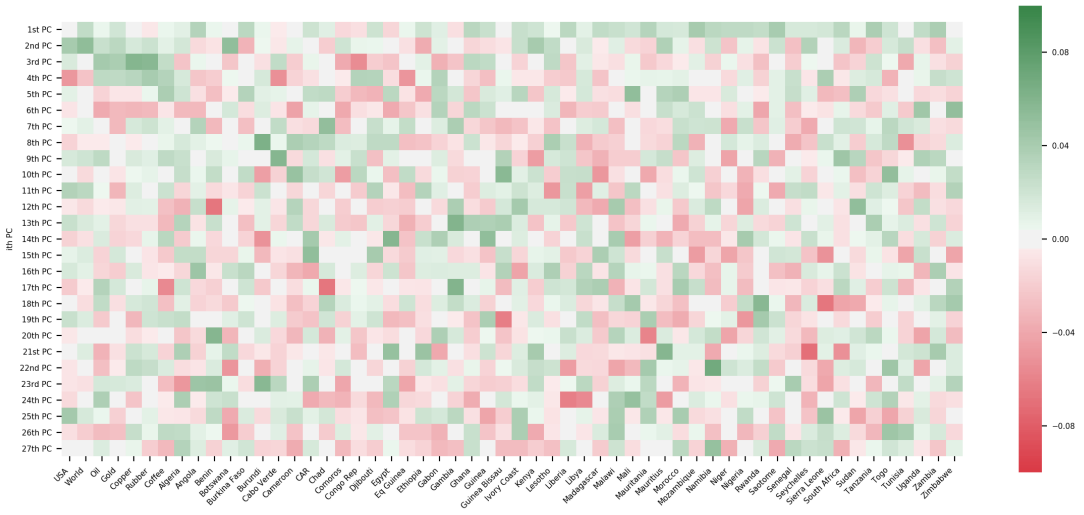


Figure 3: heatmap of the proportional contribution to the first 27 PCs

nent. Additionally, no obvious cluster of factor loadings can be interfered from the heatmap, which would indicate grouped explanatory variables. For a clearer picture of the factor loadings, Table 10 in the Appendix gives the factor loadings of the first three normed eigenvectors, while Figure 4 depicts the fraction of sample correlation explained by each PC. The figure shows that no PC individually accounts for a majority of the sample correlation, as the first PC only accounts for a share of 10.3% of the correlation. Further, while there is a drop between the second (8.7%) an the third PC (6.3%), the decrease in explained correlation is more gradual afterwards and the share of explained correlation only drops below 1% after the 27th PC.

To allow for a clearer identification of individual factors, the SPCs, which can set factor loadings to 0 due to their Elastic Net framework, can help. Figure 5 depicts each variables' share of the first three SPCs, while Table 11 gives the corresponding parameter estimates. Figure 5 signals that the first SPC has 10 non-zero factor loadings, and that all these factor loadings belong to Sub-Saharan African countries. As these countries form a varied mix of different economies, such as South Africa, Ethiopia and Angola, with different economic development and diverse main industries, this is a general factor for Sub-Saharan GDP growth. The second SPC attributes its largest factor to worldwide GDP growth, followed by African countries like Botswana, the Seychelles and Sao Tome. All of these countries have in common that a large

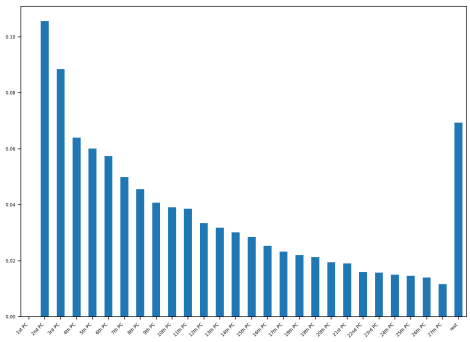


Figure 4: Fraction of explained correlation by all PCs which explain at least 1% of sample correlation

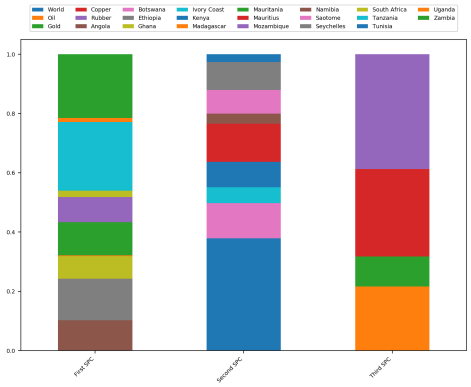


Figure 5: Variables with non-zero factor loadings for the first three SPCs

share of their GDP is built on tourism and that they are popular for international holidays. Thus, the second SPC can be seen as a proxy for the global state of the economy and its influence on African GDP growth, specifically through tourism. The third SPC is solely formed by raw materials, namely rubber, copper, oil and gold. Therefore, this factor models the impact of raw material prices on African GDP growth, which is especially important for raw-material exporting countries like the DRC.

For further information, Figure 14 in the Appendix plots the first components for both PCA and SPCA, as well as the DRC's real annual GDP growth, while Table 12 in the Appendix gives the correlations for the three variables. The plot shows that the two components capture some variation of the DRC's GDP growth, but that the contained information is limited. This is further supported by the correlation matrix with correlations between 0.3 and 0.4. Lastly, Table 13 provides the descriptive statistics for the two components.

6.3. Residual Analysis

With the factor estimates provided by PCA and SPCA as explanatory variables, forecasts can be computed according to Equation 3 and estimates $\hat{\epsilon_{t+h}}$ can be computed. By construction, their mean is 0, which is enforced by the constant term in the AR model. Other assumptions from the methodology are that the idiosyncratic components are i.i.d., as specified by the structural break framework, while the BMA methodology requires that the error terms are normally distributed.

Table 3 provides descriptive statistics of the error terms for all estimation methods and

Table 3: Descriptive statistics of the residuals for the factor models and a forecasting horizon (h) of one respectively five years

h	factor method	optimization	std	Skewness	Kurtosis	Jarque-Bera	1st autocorr.	2nd autocorr.
1	PCA	EN	5.181	-0.629	2.879	3.458	0.677	0.522
	PCA	BMA0	3.920	-0.246	1.963	2.854	-0.086	0.025
	PCA	BMA1	3.915	-0.246	1.968	2.835	-0.088	0.022
	SPCA	EN	5.089	-0.650	2.921	3.671	0.647	0.457
	SPCA	BMA0	3.859	-0.173	1.985	2.489	-0.098	-0.018
	SPCA	BMA1	3.877	-0.195	1.975	2.607	-0.095	-0.006
5	PCA	EN	5.369	-0.646	2.877	3.294	0.734	0.561
	PCA	BMA0	5.263	-0.809	3.166	5.184	0.695	0.471
	PCA	BMA1	5.258	-0.815	3.181	5.268	0.694	0.467
	SPCA	EN	5.232	-0.773	3.196	4.759	0.734	0.561
	SPCA	BMA0	5.213	-0.866	3.318	6.077*	0.686	0.445
	SPCA	BMA1	5.215	-0.864	3.313	6.044*	0.687	0.448

* indicates rejection of normality at 5% significance, ** at 1, *** at 0.1%. The test assumes that the Jarque-Bera statistic is $\chi^2(2)$ distributed

forecasting horizons of one and five years. The table shows that then Elastic Net residuals exhibit a larger standard deviation, a stronger skewness and a larger kurtosis than the BMA residuals for a forecasting horizon of one year, while this is not the case for a forecasting horizon of five years. This is in line with the simulation results, where the estimations and thus also the residuals for the Elastic Net and BMA become more similar for larger forecasting horizons. Further, the Jarque-Bera test indicates that normality is only rejected at a 5% significance level for the two BMA specifications and factors computed with SPCA, while normality is not rejected for any residual for 90% confidence intervals. Hence, it can be concluded that the estimated residuals do not strongly violate normality and fulfill the assumptions of BMA.

With regards to the autocorrelations, Table 3 depicts that for a forecast horizon of one year, only the two Elastic Net models show strong first and second autocorrelations. For a forecast horizon of five years, the BMA values again become more similar to the Elastic Net values, and the autocorrelation increases. Figure 15 in the Appendix plots the first 20 autocorrelations for residuals computed with PCA and BMA1, with 95% confidence intervals based on variances computed with Bartlett's formula. As all autocorrelations in Figure 15a are inside the 95% confidence interval, no significant autocorrelations are detected for this specification of the residual. For a forecasting horizon of 5 years, this changes and now the first two autocorrelations are significant at a 5% significance level. Such a finding is expected, as consecutive forecasts contain overlapping intervals. Further, as this finding only concerns the first two autocorrelations for one forecasting horizon, the assumptions of the structural break tests are not severely violated.

Table 4: Forecasts for the DRC's annual real GDP growth and forecasting horizons from 1 to 5 years for different models

h	forecast for	act. value	IMF estimate	AR	PCA EN	PCA BMA0	PCA BMA1	SPCA EN	SPCA BMA0	SPCA BMA1
1	2014	9.5	10.4880	5.885	6.044	5.938	5.965	6.372	6.254	6.170
	2015	6.9	10.2500	6.783	6.964	6.854	6.879	7.015	6.879	6.905
	2016	2.4	9.3080	5.038	5.065	5.050	5.051	5.349	5.254	5.227
	2017	3.7	6.6310	1.940	1.985	1.961	1.964	2.055	2.012	2.009
	2018	-	5.8920	2.841	2.969	2.888	2.909	3.104	3.043	3.012
2	2015	6.9	8.4890	5.342	5.342	5.346	5.350	5.701	5.499	5.514
	2016	2.4	7.8960	6.000	6.000	6.008	6.016	6.000	6.061	6.086
	2017	3.7	7.2980	4.346	4.346	4.347	4.348	4.586	4.462	4.462
	2018	-	6.4030	1.896	1.896	1.898	1.899	1.982	1.929	1.938
	2019	-	5.7070	2.596	2.596	2.604	2.609	2.596	2.627	2.639
3	2016	2.4	7.3000	4.772	4.772	4.780	4.787	5.175	4.955	4.963
	2017	3.7	6.8000	5.083	5.083	5.097	5.113	5.083	5.133	5.154
	2018	-	6.2000	3.816	3.816	3.819	3.821	3.816	3.933	3.933
	2019	-	6.5000	1.789	1.789	1.794	1.796	1.991	1.890	1.902
	2020	-	5.2000	2.374	2.374	2.388	2.398	2.453	2.423	2.435
4	2017	3.7	4.2470	3.289	3.289	3.293	3.296	3.707	3.470	3.496
	2018	-	4.9690	3.614	3.614	3.621	3.629	3.614	3.659	3.680
	2019	-	4.8650	2.835	2.835	2.836	2.836	2.835	2.868	2.878
	2020	-	5.0110	1.488	1.488	1.490	1.491	1.701	1.574	1.594
	2021	-	5.5190	1.877	1.877	1.882	1.886	1.877	1.937	1.957
5	2018	-	2.9870	2.579	2.579	2.622	2.648	3.232	3.094	2.958
	2019	-	3.2730	2.773	2.773	2.821	2.854	2.773	2.924	2.963
	2020	-	3.7250	2.270	2.270	2.272	2.273	2.571	2.475	2.447
	2021	-	4.3080	1.399	1.399	1.419	1.424	1.399	1.441	1.454
	2022	-	4.7020	1.650	1.650	1.691	1.706	1.978	1.841	1.831

6.4. Forecasts

To test the accuracy of the forecasts and provide an answer to the question of improved forecasting accuracy, forecasting estimates of the DRC's real annual percentage GDP growth are given in Table 4 together with the actual in-sample values and the IMF estimates from October of the prior year. As for each forecasting horizon exactly five forecasts are created based on the same explanatory variables, forecasts for horizons of more than one year include out-of-sample forecasts. Table 4 shows that overall, the factor model forecasts do not largely deviate from the AR forecasts, which are always based on one lag as $p = 1$ is chosen as the best criterion according to the Bayesian Information Criterion. In fact, for forecasting horizons of more than one year the Elastic Net estimate based on PC factors is equal to the AR forecast in most cases. In contrast, the IMF estimates deviate from both the AR and the factor model estimates.

Forecast errors for all in-sample forecasts are given in Table 5. As indicated by the table's last column, except for the 1-step ahead 2014 forecast, the above described models always outperform the IMF forecasts. Yet, this comparison may not be too realistic, as the forecasting models are based on GDP growth estimates that were not available when the models predict the forecast. For example, the World Bank 2015 GDP growth data were not available in 2015, such that 1-step ahead forecasts for the 2016 GDP growth could only be made once the forecasts are published, which did not happen before 2017. Also, it is a notable finding that the IMF forecasts

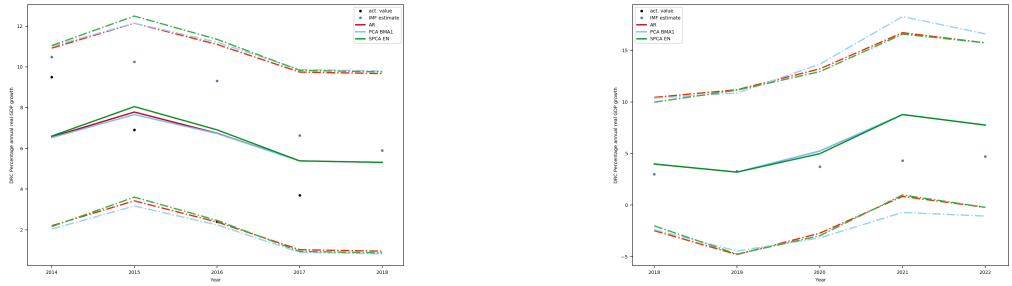
Table 5: Forecast Errors for the DRC's annual real GDP growth and forecasting horizons from 1 to 5 years for different models

h	forecast for	IMF estimate	AR	PCA EN	PCA BMA0	PCA BMA1	SPCA EN	SPCA BMA0	SPCA BMA1	best model
1	2014	-0.988	3.615	3.456	3.562	3.535	3.128	3.246	3.330	IMF
	2015	-3.35	0.117	-0.064	0.046	0.021	-0.115	0.021	-0.005	SPCA BMA1
	2016	-6.908	-2.638	-2.665	-2.650	-2.651	-2.949	-2.854	-2.827	AR
	2017	-2.931	1.760	1.715	1.739	1.736	1.645	1.688	1.691	SPCA EN
2	2015	-1.589	1.558	1.558	1.554	1.550	1.199	1.401	1.386	SPCA EN
	2016	-5.496	-3.600	-3.600	-3.608	-3.616	-3.600	-3.661	-3.686	AR
	2017	-3.598	-0.646	-0.646	-0.647	-0.648	-0.886	-0.762	-0.762	AR
3	2016	-4.9	-2.372	-2.372	-2.380	-2.387	-2.775	-2.555	-2.563	AR
	2017	-3.1	-1.383	-1.383	-1.397	-1.413	-1.383	-1.433	-1.454	AR
4	2017	-0.547	0.411	0.411	0.407	0.404	-0.007	0.230	0.204	SPCA EN

tend to overestimate the in-sample GDP growth, whereas the computed forecasts often provide estimates that are too low. This is likely due the sharpe decrease of the DRC's GDP growth at the end of the sample period, as the growth rate falls from a high-growth environment with values above 9% to growth rates between 2% and 4%. Therefore, this forecasting environment is favorable for stationary AR based models which have a mean reverting property.

Table 5 further shows that the simple AR model provides the best forecast in five out of 10 cases, with the Elastic Net model based on SPC factors winning for three forecasts and the BMA1 model with PC factors winning for one forecast. Therefore, it can not be concluded that the factor models with machine learning optimization methods are able to improve forecasting accuracy.

With regards to Cross-validation estimates of the forecasting errors' standard deviation, which are given in Tables 16 and 17 in the Appendix, the standard deviation estimates increase for longer forecasting horizons, while the difference for specific forecasting methods is only marginal. Also, there is a slight tendency for the forecasting errors to increase for longer gaps between the training data and the forecasted variable. This small increase may be due to the occurrence of structural breaks, which are tested in the next section. One potential reason for the similarity of the AR forecasts with the factor model estimates is that for all forecasts, the information criterion in Equation 7 only selects the first factor. This result was consistent for other factor selection methods described in Bai and Ng (2002), as well as for for an application of selection criteria to SPCA directly. An explanation for this finding is that the first few principal components do not explain a majority of the sample correlation, as indicated by Figure 4. Further, inclusion of a larger number of factors, such as all factors with individually explained sample correlation above a threshold of 1% or 2% did change the forecasting estimates, but this change was only marginal compared to the difference between, for example, the factor model



(a) Forecasts for $h = 1$

(b) Forecasts for $h = 5$

Figure 6: 1 and 5-step ahead forecasts of the DRC's real annual percentage GDP growth for optimization with the AR model, BMA1 with PCA factors and EN with SPCA factors against the IMF estimates and the actual values. Dashed-pointed lines give 2-standard deviation (approximately 70%) Confidence Intervals

forecasts and the IMF estimates, and did not improve in-sample forecasting accuracy.

To provide a graphical representation of the different forecast, estimates for the best models by Table 5 are plotted in Figure 6 for forecasting horizons of one and five years and Figure 16 in the Appendix for two, three and four years. For in-sample forecasts, the actual growth rates are also given, as are estimates for the CV-based confidence intervals. The figures clearly indicate the similarity between the AR and the factor model forecasts. Moreover, due to the large confidence intervals, the forecasts are predicted to not be too accurate and are not significantly different from the IMF forecasts. However, due to the limited sample size for the CV, forecasting errors may be overestimated such that this finding is not too robust. Still, the similarity between the forecasting errors and the in-sample residuals in Table 3 is a favorable sign for the importance of the large forecasting errors.

6.5. Structural Break Tests

The above results depend on the assumption that neither the AR process nor the factor models are affected by strong, or only impacted by weak structural parameter breaks. First, this assumption is tested for known break dates. As was outlined in the Section 2, the DRC experienced impactful political events during the past 30 years that could have potentially changed the country's growth path, for example the Congo Wars. As potential break dates, the start of the First Congo War (1996), the end of the Mobutu dictatorship (1997) and the end of the second

Table 6: Break tests for known break dates in 1996, 1997, 2003, 2004 and 2008 with assumed breaks in the AR process and the factor loadings

	$h = 1$		$h = 2$		$h = 3$		$h = 4$		$h = 5$	
year	LM stat.	p-value	LM stat.	p-value	LM stat.	p-value	LM stat.	p-value	LM stat.	p-value
1996	5.691	0.128	4.201	0.241	6.191	0.103	9.132	0.028	10.703	0.013
1997	6.062	0.109	5.951	0.114	7.363	0.061	11.647	0.009	14.882	0.002
2003	6.132	0.105	7.380	0.061	6.866	0.076	9.961	0.019	11.105	0.011
2008	3.007	0.391	3.436	0.329	5.062	0.167	5.862	0.119	6.063	0.109

Congo War (2003) are chosen. Further, the start of the Great Recession (2008) is investigated.

Table 6 summarizes the LM tests for all four break tests. The LM statistics are $\chi^2(3)$ distributed, as all models only contain a constant, one factor and one lag of the dependent variable by choice of the selection criteria. While for 1-step ahead forecasts, none of the break dates is significant even at a 10% significance level, both two and three year forecasting horizons indicate structural breaks in 1997 and 2003 at a 10% significance level. Additionally, 4 and 5-step ahead forecasts show significant breaks in 1996, 1997 and 2003 at a 5% significance level. These findings indicate the the political events distorted the data structure, whereas no significant effect of the Great Recession was found.

The different significance levels of the potential break dates for different forecasting horizon show that the chosen break dates do not provide full information about the potential occurrence of parameter changes. Therefore, a test for an unknown break date is more applicable. For this test, the parameter value $\tau = 0.185$ is chosen to perform statistical analysis over 63% of the sample. Table 18 in the Appendix lists all LM statistics for the tested 33 years, while Table 7 lists the maximum LM statistic for each forecasting horizon. Using the critical values from Andrews (1993) for $\tau = 0.2$ and $p = 3$, which are approximately equal to critical values for $\tau = 0.185$, the statistics reject the assumption of no structural breaks for all forecasting horizons except for the 2-step ahead forecasts at a 10% significance level. Moreover, except for the 3 year forecast horizon, the same conclusion also occurs at a 5% significance level. It thus appears plausible to assume that the sample is indeed impacted by structural breaks. Still, the exact placement of the structural break differs per forecast. Whereas forecasting horizons of more than 1 year show their largest LM statistic for years around 2000 and therefore for time periods during the second Congo War, the 1-step ahead forecast exhibits its most significant LM statistic for 1975, which takes place during the Mobutu dictatorship. To highlight the difference between these two break sets, Table 18 tells that forecast horizons of more than one year do not indicate the

Table 7: Maximum LM statistics for an unknown break date with $\tau = 0.185$ for simulataneous breaks of the AR process and the factor loadings. The critical values from Andrews (1993) for the comparable bound at $\tau = 0.2$ are 11.80 at 10% significance, 13.69 at 5% significance and 17.28 at 1% significance

	$h = 1$	$h = 2$	$h = 3$	$h = 4$	$h = 5$
year of maximum	1975	2001	2000	1999	1998
max LM statistic	13.844**	10.749	12.970*	15.485**	16.225**

* indicates rejection of no breaks at 5% significance, ** at 1, *** at 0.1%.

possibility of structural breaks at any acceptable significance level in 1975, while 1-step ahead forecasts are insignificant around 2000.

As structural breaks can worsen the functionality of factor models, forecasts that take the occurrence of parameter breaks into account can provide better forecasts. With two very distinct date estimates for parameter changes, this gives two different forecasting models with break dates in 1975 respectively 2000. As only forecasts for periods beyond 2013 are created, data before the respective break date are only used for the PCA and SPCA calculations, while factors and lagged dependent variables from these time periods are deleted from the sample. Tables 19 and 21 in the Appendix show the forecasts for the new specifications. Overall, the forecasts do not improve, but their relative performance compared to the AR model, especially for the BMA specifications, over the same data sample improves as indicated by Tables 20 and 22. Thus, the non-improvement of the forecasts may simply be due to an insufficient number of observations, as for example for the break date in 2000 only 14 observations are used for the AR process and the machine learning models, whereas the factor model and machine learning methodology is now better able to capture the underlying stochastic process. Hence, the general conclusion for this section is that there is strong support for the occurrence of structural breaks and models that take such breaks into account appear to be better able to replicate the data-generating process.

7. Conclusion

This paper analyzed the predictive power of GDP growth from African countries bundled with international variables onto the DRC's annual GDP growth in a factor model framework. The paper's findings are two-fold: First, the factor models did not improve the models predictive power as compared to an AR model. This is potentially due to a limited number of PC and SPC factors selected by criteria suggested in Bai and Ng (2002). Yet, when compared to the official IMF forecasts, the models perform favorable, plausibly due to the mean-reverting property of

the AR model.

Second, it was found that the framework is impacted by structural parameter breaks, which can differ depending on the forecasting horizon. For 1-step ahead forecasts, the LM test for an unknown break date exhibits its most extreme statistics in 1975, while for 2-to-5-step ahead forecasts this occurs around 2000. While the first break happens in the midst of the Mobutu dictatorship, the second break date marks the period of the Congo Wars, which changed the DRC from a dictatorship to a formally democratic system. Inclusion of the break dates can improve predictive performance as compared to an AR model.

Especially for forecasts with structural break assumptions, the BMA models outperform Elastic Net specifications. This suits to the simulation outcomes, where BMA models are often the best-performing models in terms of forecast accuracy. A key difference between the Elastic Net and BMA models is that the Elastic Net enforces stricter restrictions on its parameter values, such that they are more often equal to zero. For BMA specifications, this effct is less drastic as models are selected in an iterative process based on prior and posterior probabilities. The finding of less non-zero parameter values for BMA specifications is consistent with both the simulation and the findings for the DRC's GDP growth.

As limitations, this research only included growth data from other countries and price changes of natural resources as explanatory variables. Though different variable transformations were tested, they did not have a significant influence on the results. Inclusion of other explanatory variables can increase the predictive power of the model. Additionally, higher frequency data such as quarterly growth data can benefit the study, as they allow for a larger testing sample, enabling the computation of significant MSPE comparison statistics, and for more reliable estimates for later break dates. A second limitation is that this study checks a specific subset of machine learning techniques and factor methods. Other optimization methods, such as the Random Forest model, Support Vector Machines or an addition of Neural Networks, could improve forecast accuracy. Also, usage of Independent Component Analysis can alter the factor selection process. As a third limitation, the study only considers forecasts for the DRC. As the used variables are applicable to other African countries as well, it can be interesting to check how forecast estimates perform for distinct African countries, and whether differences in forecasting accuracy are attributable to specific characteristics, such as a country's geographic position, the openness to trade or the structure of its exports and imports.

References

Aang, A., Piazzesi, M., & Wei, M. (2006). What does the yield curve tell us about gdp growth? *Journal of Econometrics*, *131*(1-2), 359–403.

Adoho, F. M., & Doumbia, D. (2018). Informal sector heterogeneity and income inequality : evidence from the democratic republic of congo. *Policy Research working paper; no. WPS 8328*.

Andrews, D. W. (1993). Tests for parameter instability and structural change with unknown change point. *Econometrica: Journal of the Econometric Society*, 821–856.

Artis, M. J., & Banerjee, A. (2005). Factor forecasts for the uk. *Journal of Forecasting*, *24*(4), 279–298.

Bai, J., & Ng, S. (2002). Determining the number of factors in approximate factor models. *Econometrica*, *70*(1), 191–221.

Bai, J., & Ng, S. (2008). Forecasting economic time series using targeted predictors. *Journal of Econometrics*, *146*(2), 304–317.

Bai, J., & Perron, P. (2003). Computation and analysis of multiple structural change models. *Journal of applied econometrics*, *18*(1), 1–22.

Barro, R. J. (1996). *Determinants of economic growth: a cross-country empirical study* (Tech. Rep.). National Bureau of Economic Research.

Breitung, J., & Eickmeier, S. (2011). Testing for structural breaks in dynamic factor models. *Journal of Econometrics*, *163*(11), 71–84.

Chamberlain, G., & Rothschild, M. (1982). *Arbitrage, factor structure, and mean-variance analysis on large asset markets.* National Bureau of Economic Research Cambridge, Mass., USA.

Chipman, H., George, E. I., McCulloch, R. E., Clyde, M., Foster, D. P., & Stine, R. A. (2001). The practical implementation of bayesian model selection. *Lecture Notes-Monograph Series*, 65–134.

Efron, B., Hastie, T., Johnstone, I., Tibshirani, R., et al. (2004). Least angle regression. *The Annals of statistics*, *32*(2), 407–499.

Eickmeier, S., Lemke, W., & Marcellino, M. G. (2011). Classical time-varying favar models – estimation, forecasting and structural analysis. *Bundesbank Series 1 Discussion Paper*, *2011*(4).

Fernandez, C., Ley, E., & Steel, M. F. (2001a). Benchmark priors for bayesian model averaging. *Journal of Econometrics*, *100*(2), 381–427.

Fernandez, C., Ley, E., & Steel, M. F. (2001b). Model uncertainty in cross-country growth regressions. *Journal of applied Econometrics*, *16*(5), 563–576.

Foster, V., & Benitez, D. A. (2011). *The democratic republic of congo's infrastructure: a continental perspective*. The World Bank.

Franses, P. H., & Janssens, E. (2019). Spurious principal components. *Applied Economics Letters*, *26*(1), 37–39.

Franses, P. H., & Vasilev, S. (2019). *Real gdp growth in africa, 1963-2016*. (Working paper)

Gregory, H. A. C., Allan W. (1999). Common and country-specific fluctuations in productivity, investment, and the current account. *Journal of Monetary Economics*, *44*(3), 423–451.

Haskin, J. M. (2005). *The tragic state of the congo: From decolonization to dictatorship*. Algora Publishing.

Hastie, T., Tibshirani, R., & Friedman, J. (2009). *The elements of statistical learning: data mining, inference, and prediction, springer series in statistics*. Springer New York.

Heij, C., De Boer, P., Franses, P., Kloek, T., & Dijk, H. (2004). *Econometric methods with applications in business and economics*. Oxford University Press.

Hoerl, A. E., & Kennard, R. W. (1970). Ridge regression: Biased estimation for nonorthogonal problems. *Technometrics*, *12*(1), 55–67.

Hoeting, J. A., Madigan, D., Raftery, A. E., & Volinsky, C. T. (1999). Bayesian model averaging: a tutorial. *Statistical science*, 382–401.

Hyndman, R. J., & Athanasopoulos, G. (2018). *Forecasting: principles and practice*. OTexts.

Kabemba, C. (2016). China-democratic republic of congo relations: From a beneficial to a developmental cooperation. *African Studies Quarterly*, *16*.

Kim, H. H. (2018). Looking into the black box of the korean economy: the sparse factor model approach. *Journal of Asia Pacific Economy*, *23*(1), 1–16.

Kim, H. H., & Swanson, N. R. (2013). Mining big data using parsimonious factor and shrinkage methods. econstor working papers, no. 2013-16.

Kim, H. H., & Swanson, N. R. (2018). Mining big data using parsimonious factor, machine learning, variable selection and shrinkage methods. *International Journal of Forecasting*, *34*(2), 339–354.

Koop, G., & Potter, S. (2004). Forecasting in dynamic factor models using bayesian model averaging. *The Econometrics Journal*, *7*(2), 550–565.

Lucas Jr, R. E. (1988). On the mechanics of economic development. *Journal of monetary economics*, *22*(1), 3–42.

Mardia, K., & Bibby, J. (1979). *Multivariate analysis: Probability and mathematical statistics*. Academic press London.

MIT. (n.d.). *Orc: Democratic republic of the congo.* `https://atlas.media.mit.edu/en/profile/country/cod/`. (Accessed: June 10, 2019)

Montague, D. (2002). Stolen goods: Coltan and conflict in the democratic republic of congo. *Sais Review*, *22*(1), 103–118.

Nest, M. W., Grignon, F., & Kisangani, E. F. (2006). *The democratic republic of congo: Economic dimensions of war and peace* (Vol. 53). Lynne Rienner Boulder, CO.

Nzongola-Ntalaja, G. (2002). *The congo: From leopold to kabila: A people's history*. Zed Books.

Poirier, D. J. (1995). *Intermediate statistics and econometrics: a comparative approach*. Mit Press.

Ravazzolo, F., Paap, R., Van Dijk, D., & Franses, P. H. (2008). Chapter 15 bayesian model averaging in the presence of structural breaks. In *Forecasting in the presence of structural breaks and model uncertainty* (pp. 561–594). Emerald Group Publishing Limited.

Reporters Without Borders. (2019). *World press freedom index - 2019*. `https://rsf.org/en/ranking`.

Ritchie, M. D., Holzinger, E. R., Li, R., Pendergrass, S. A., & Kim, D. (2015). Methods of integrating data to uncover genotype–phenotype interactions. *Nature Reviews Genetics*, *16*(2), 85.

Romer, P. M. (1986). Increasing returns and long-run growth. *Journal of political economy*, *94*(5), 1002–1037.

Sachs, J. D., & Warner, A. M. (1995). *Natural resource abundance and economic growth* (Tech. Rep.). National Bureau of Economic Research.

Sachs, J. D., & Warner, A. M. (1997). Sources of slow growth in african economies. *Journal of African economies*, *6*(3), 335–376.

Schumacher, C. (2010). Factor forecasting using international targeted predictors: The case of german gdp. *Economics Letters*, *107*(2), 95–98.

Skittides, C., & Früh, W.-G. (2014). Wind forecasting using principal component analysis. *Renewable Energy*, *69*, 365–374.

Solow, R. M. (1956). A contribution to the theory of economic growth. *The quarterly journal of economics*, *70*(1), 65–94.

Stijns, J.-P. C. (2005). Natural resource abundance and economic growth revisited. *Resources policy*, *30*(2), 107–130.

Stock, J. H., & Watson, M. (2011). Dynamic factor models. *Oxford handbook on economic forecasting*.

Stock, J. H., & Watson, M. W. (2002). Forecasting using principal components from a large number of predictors. *Journal of the American Statistical Association*, *97*(460), 1167–1179.

Stock, J. H., & Watson, M. W. (2006). Forecasting with many predictors. *Handbook of economic forecasting*, *1*, 515–554.

Swan, T. W. (1956). Economic growth and capital accumulation. *Economic record*, *32*(2), 334–361.

Swanson, N. R., & Xiong, W. (2018). Big data analytics in economics: What have we learned so far, and where should we go from here? *Canadian Journal of Economics*, *51*(3), 695–746.

Tashman, L. J. (2000). Out-of-sample tests of forecasting accuracy: an analysis and review. *International journal of forecasting*, *16*(4), 437–450.

The Economist Intelligence Unit. (2019). *Democracy index 2018: Me too? political participation, protest and democracy* (Tech. Rep.).

Tibshirani, R. (1996). Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society: Series B (Methodological)*, *58*(1), 267–288.

Trefon, T. (2011, November 21). *Failed state: Can dr congo recover?* Retrieved from `https://www.bbc.com/news/world-africa-15775445` (Accessed: June 10, 2019)

Turner, T. (2007). *The congo wars: conflict, myth and reality*. Zed Books.

United Nations Development Programme. (2018). *Human development indices and indicators 2018 statistical update* (Tech. Rep.).

Wall, M. E., Rechtsteiner, A., & Rocha, L. M. (2003). Singular value decomposition and principal component analysis. In *A practical approach to microarray data analysis* (pp. 91–109). Springer.

World Atlas. (n.d.-a). *Democratic republic of the congo geography.* `https://www.worldatlas.com/webimage/countrys/africa/drc/cdland.htm`. (Accessed: June 10, 2019)

World Atlas. (n.d.-b). *What are the major natural resources of the democratic republic of the congo?* `https://www.worldatlas.com/articles/what-are-the-major-natural-resources-of-the-democratic-republic-of-the-congo.html`. (Accessed: June 10, 2019)

Zou, H., & Hastie, T. (2005). Regularization and variable selection via the elastic net. *Journal of the royal statistical society: series B (statistical methodology)*, *67*(2), 301–320.

Zou, H., Hastie, T., & Tibshirani, R. (2006). Sparse principal component analysis. *Journal of computational and graphical statistics*, *15*(2), 265–286.

Appendix

A. Appendix to the Country Review

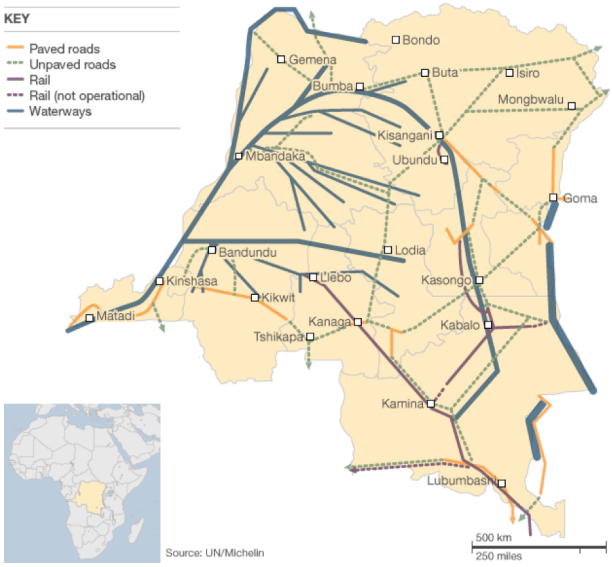


Figure 7: Digrammatic map of the DRC's national transport systems (Trefon, 2011)

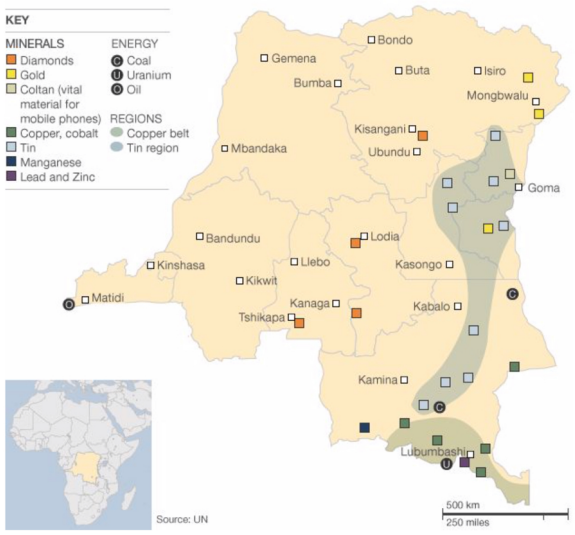
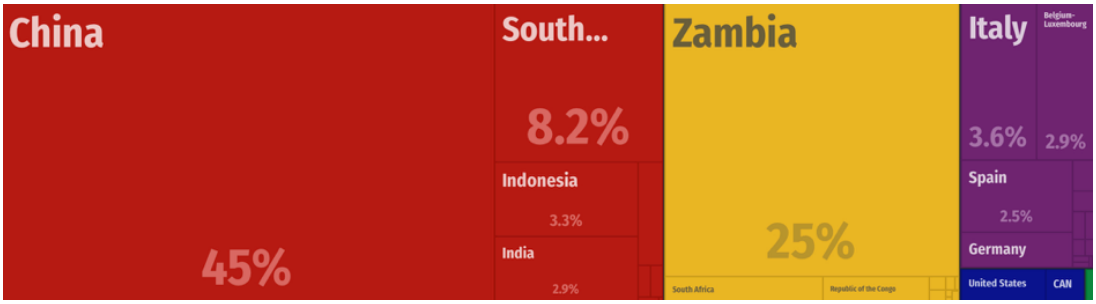


Figure 8: Map of the natural resources in the DRC (Trefon, 2011)

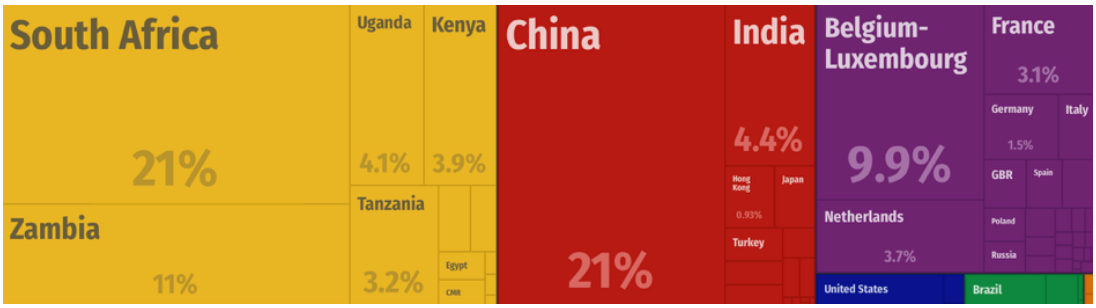
Table 8: Sources for the Data in Section 2. All web pages retrieved on June 10, 2019

variable	country / region	institution	url link
population	DRC	World Bank	World Bank DRC overview
population	DRC	IMF	IMF overview page
% population growth	DRC, Sub-Saharan Africa	World Bank	World Bank population growth
PPP adjusted GDP per capita	DRC, Sub-Saharan Africa	World Bank	World Bank PPP adjusted GDP data
population share below $1.90 per day	DRC, Sub-Saharan Africa	World Bank	World Bank poverty data
population share below $3.20 per day	DRC, Sub-Saharan Africa	World Bank	World Bank poverty data
GDP data	DRC, Sub-Saharan Africa	World Bank	World Bank GDP data
agricalture overview	DRC	World Bank	World Bank agricultural overview

Figure 9: Exported goods of the DRC in percentage of total exports. 2017 data (MIT)



(a) destinations of the DRC's exports



(b) origins of the DRC's imports

Figure 10: Destinations of the DRC's exports and origins of its imports in percentage of total exports respectivly imports. 2017 data (MIT)

B. Appendix to the Methodology

B.1. Parameter specifications of the simulation

In AR configuration 1, there exists 1 lag with $\alpha = (0.8, 0.65, -0.7)$ for 3 factors and $\alpha = (0.8, 0.65, -0.7, 0.7, -0.8)$ for 5 factors. In specification 2, weaker parameters are chosen such that $\alpha = (0.3, 0.2, -0.25)$ and $\alpha = (0.3, 0.2, -0.25, 0.25, -0.3)$. For configuration 3, 3 lags are chosen ($p = 3$) such that for lag 1, $\alpha_1 = (0.6, 0.5, -0.5)$, for lag 2 $\alpha_2 = (0.2, 0.3, -0.2)$ and for lag 3 $\alpha_3 = (0.1, -0.2, 0.3)$. For 5 factors, this gives $alpha_1 = (0.6, 0.5, -0.5, 0.3, -0.4), \alpha_2 = (0.2, 0.3, -0.2, 0.55, -0.4)$ and $\alpha_3 = (0.1, -0.2, 0.3, 0.1, -0.1$.

The homogenous specification with uncorrelated errors is given by $\delta_0 = 1, \delta_1 = 0, \delta_2 = 0, a = 0$ and $b = 0$. For the correlated heterogenous errors, $\delta_0 = 0.7, \delta_1 = 0.25, \delta_2 = 0.05, a = 0.1$ and $b - 0.2$.

C. Appendix to the Data Section

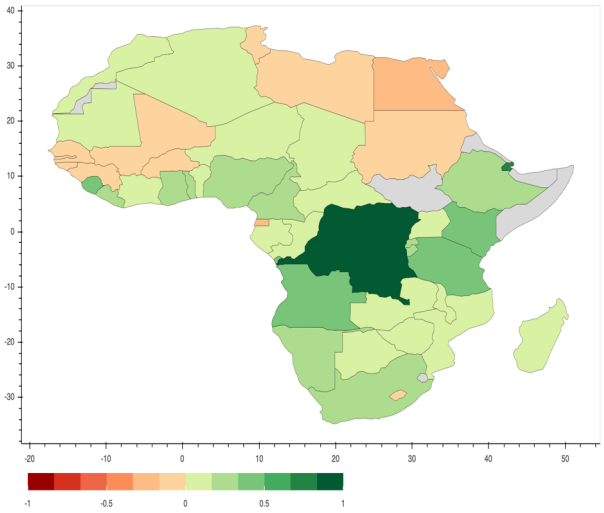


Figure 11: Correlation of annual real GDP growth between the African countries and the DRC. For grey countries / disputed regions, data are not included in the data set

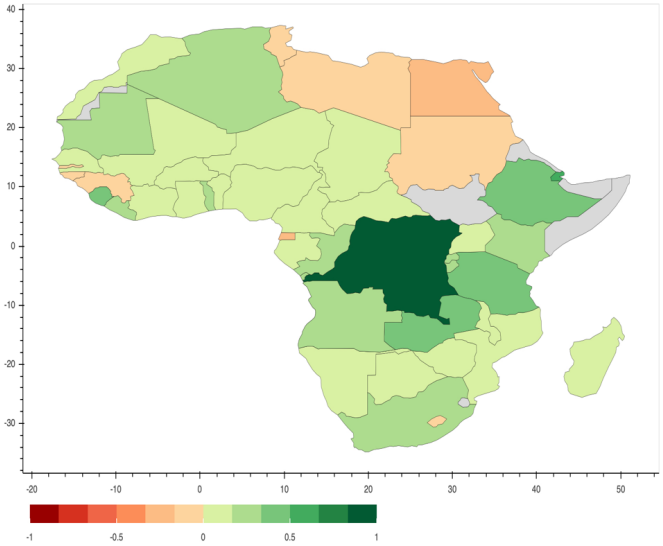


Figure 12: Correlation of annual real GDP growth between the African countries lagged by 5 years and the DRC. For grey countries / disputed regions, data are not included in the data set

Table 9: Descriptive statistics of annual real percentage GDP growth for the USA, the world and the other African countries, as well as percentage price changes of commodities reduced by inflation

Country	mean	max	min	std. dev.	Country	mean	max	min	std. dev.
USA	3.01	7.26	-2.78	2.07	Ivory Coast	4.05	17.60	-11.00	5.39
World	3.46	6.67	-1.24	1.57	Kenya	4.75	15.99	-4.70	3.75
Oil	6.87	118.36	-54.60	37.16	Lesotho	4.56	19.85	-10.73	5.10
Gold	3.84	67.28	-37.54	21.15	Liberia	1.04	44.98	-42.91	14.65
Copper	5.26	105.01	-56.07	33.25	Libya	0.03	39.06	-39.00	13.01
Rubber	4.54	123.10	-54.37	39.52	Madagascar	2.06	9.90	-9.33	3.80
Coffee	1.80	95.44	-53.43	31.21	Malawi	4.31	16.70	-10.20	4.95
Algeria	4.08	16.97	-8.81	4.30	Mali	4.10	18.92	-7.40	4.94
Angola	3.78	22.60	-15.42	6.40	Mauritania	3.69	19.54	-5.10	5.28
Benin	3.81	10.00	-4.90	2.94	Mauritius	5.83	14.20	-4.97	3.60
Botswana	8.30	26.40	-7.70	6.07	Morocco	4.99	13.30	-5.40	3.83
Burkina Faso	4.39	11.00	-1.80	3.16	Mozambique	5.53	21.54	-10.49	5.34
Burundi	2.67	17.59	-8.00	4.98	Namibia	4.10	12.15	-1.80	2.68
Cabo Verde	5.89	19.20	-2.30	4.73	Niger	2.62	13.50	-13.54	5.39
Cameroon	3.73	20.11	-10.90	5.46	Nigeria	3.95	27.75	-15.70	7.93
CAR	1.70	9.50	-10.33	4.01	Rwanda	5.19	26.13	-15.75	6.98
Chad	3.36	28.02	-21.29	8.22	Sao Tome	4.38	22.59	-10.30	6.07
Comoros	3.11	12.86	-5.40	3.25	Senegal	3.06	8.90	-6.60	3.60
Congo Rep	3.99	20.29	-9.00	5.43	Seychelles	4.71	21.20	-8.20	6.06
Djibouti	2.44	7.10	-6.60	3.24	Sierra Leone	2.75	23.39	-17.90	6.88
Egypt	4.96	13.09	0.60	2.71	South Africa	2.95	7.90	-2.10	2.41
Eq Guinea	10.88	68.47	-36.80	19.20	Sudan	3.98	16.70	-6.30	5.27
Ethiopia	3.67	13.90	-14.00	7.06	Tanzania	4.66	8.50	0.50	1.99
Gabon	3.75	28.01	-20.50	8.09	Togo	3.77	15.50	-12.75	5.51
Gambia	3.92	12.40	-4.30	3.29	Tunisia	4.58	14.15	-1.90	3.19
Ghana	3.68	14.00	-8.96	4.21	Uganda	5.47	15.80	-6.20	3.89
Guinea	3.50	8.66	-1.50	1.72	Zambia	3.46	16.60	-8.60	4.63
Guinea Bissau	2.43	18.20	-13.78	5.40	Zimbabwe	2.95	22.60	-17.70	7.49

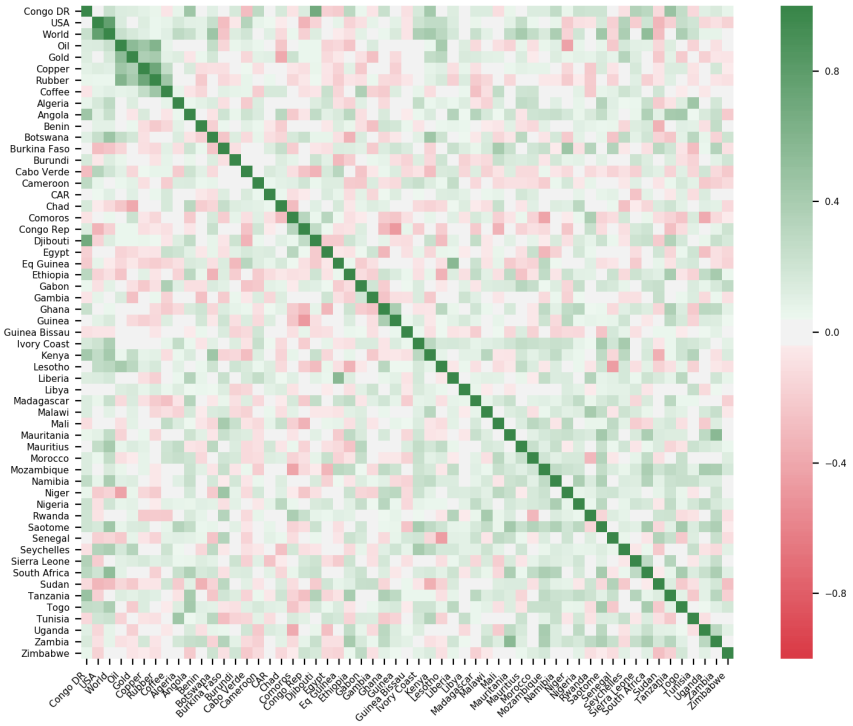


Figure 13: Correlation heatmap for all 57 variables.

D. Appendix to the Result Section

Table 10: Factor loadings for the first three PCs as defined by the three eigenvectors corresponding to the three largest eigenvalues

	First PC	Second PC	Third PC		First PC	Second PC	Third PC
USA	-0.004	0.252	0.087	Ivory Coast	0.156	0.148	-0.091
World	0.167	0.326	0.004	Kenya	0.127	0.256	0.008
Oil	0.039	0.148	0.244	Lesotho	0.025	0.170	0.165
Gold	0.119	0.185	0.227	Liberia	0.194	0.005	-0.135
Copper	0.007	0.103	0.338	Libya	0.058	0.094	0.061
Rubber	-0.035	0.138	0.333	Madagascar	0.172	-0.026	-0.036
Coffee	0.038	0.055	0.150	Malawi	0.143	0.071	-0.063
Algeria	0.065	0.110	-0.140	Mali	0.120	-0.073	0.037
Angola	0.220	-0.079	0.027	Mauritania	0.209	-0.099	-0.128
Benin	0.030	-0.048	-0.004	Mauritius	0.108	0.224	-0.078
Botswana	0.018	0.317	-0.132	Morocco	0.098	0.051	-0.023
Burkina Faso	0.189	-0.202	0.000	Mozambique	0.262	-0.066	0.060
Burundi	0.036	0.026	-0.052	Namibia	0.191	0.130	-0.100
Cabo Verde	-0.025	-0.035	0.155	Niger	0.178	-0.152	0.030
Cameroon	-0.012	0.052	-0.110	Nigeria	0.181	0.051	-0.055
CAR	0.135	0.017	0.037	Rwanda	0.162	-0.048	-0.152
Chad	0.071	-0.122	0.000	Saotome	0.252	0.138	0.014
Comoros	-0.088	0.121	-0.220	Senegal	0.106	-0.160	-0.115
Congo Rep	-0.064	0.039	-0.313	Seychelles	0.104	0.218	0.026
Djibouti	0.134	0.029	-0.112	Sierra Leone	0.117	-0.012	0.006
Egypt	-0.095	0.001	-0.144	South Africa	0.198	0.086	-0.143
Eq Guinea	0.035	-0.060	0.002	Sudan	0.043	-0.197	-0.092
Ethiopia	0.137	-0.228	0.054	Tanzania	0.245	-0.115	0.123
Gabon	0.072	0.056	-0.207	Togo	0.124	0.097	0.023
Gambia	-0.068	0.018	-0.141	Tunisia	0.009	0.156	-0.228
Ghana	0.198	-0.094	0.175	Uganda	0.169	-0.064	0.045
Guinea	0.146	-0.068	0.162	Zambia	0.239	-0.165	-0.056
Guinea Bissau	-0.014	-0.047	0.007	Zimbabwe	0.023	0.058	-0.100

Table 11: Factor loadings for the first three SPCs as defined by the columns of Lambda belonging which correspond to the three eigenvalue approximations for the three largest eigenvalues.

	First PC	Second PC	Third PC		First PC	Second PC	Third PC
USA	0	0	0	Ivory Coast	0	0.121	0
World	0	0.844	0	Kenya	0	0.191	0
Oil	0	0	0.399	Lesotho	0	0	0
Gold	0	0	0.187	Liberia	0	0	0
Copper	0	0	0.544	Libya	0	0	0
Rubber	0	0	0.714	Madagascar	0.008	0	0
Coffee	0	0	0	Malawi	0	0	0
Algeria	0	0	0	Mali	0	0	0
Angola	0.259	0	0	Mauritania	0.283	0	0
Benin	0	0	0	Mauritius	0	0.286	0
Botswana	0	0.264	0	Morocco	0	0	0
Burkina Faso	0	0	0	Mozambique	0.215	0	0
Burundi	0	0	0	Namibia	0	0.075	0
Cabo Verde	0	0	0	Niger	0	0	0
Cameroon	0	0	0	Nigeria	0	0	0
CAR	0	0	0	Rwanda	0	0	0
Chad	0	0	0	Saotome	0	0.178	0
Comoros	0	0	0	Senegal	0	0	0
Congo Rep	0	0	0	Seychelles	0	0.211	0
Djibouti	0	0	0	Sierra Leone	0	0	0
Egypt	0	0	0	South Africa	0.054	0	0
Eq Guinea	0	0	0	Sudan	0	0	0
Ethiopia	0.354	0	0	Tanzania	0.587	0	0
Gabon	0	0	0	Togo	0	0	0
Gambia	0	0	0	Tunisia	0	0.057	0
Ghana	0.192	0	0	Uganda	0.035	0	0
Guinea	0	0	0	Zambia	0.544	0	0
Guinea Bissau	0	0	0	Zimbabwe	0	0	0

Table 12: Correlation matrix for the DRC's annual real percentage GDP growth, the first normed PC and the first normed SPC

	Congo DR	First PC	First SPC
Congo DR	1	0.313	0.396
First PC		1	0.650
First SPC			1

Table 13: Descriptive statistics for the first normed PC and SPC

mean	median	max	min	std	25th percentile	75th percentile	Jarque-Bera
3.916	3.644	8.020	-0.938	1.963	2.8190	5.298	0.135
10.188	10.282	26.381	-5.264	7.850	4.776	15.229	0.974

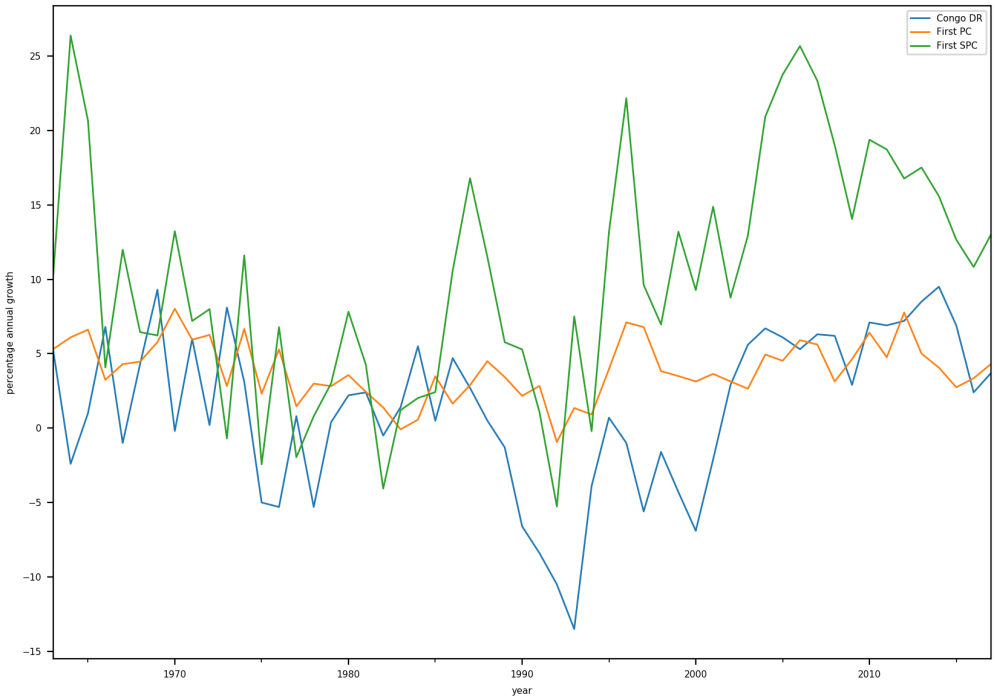


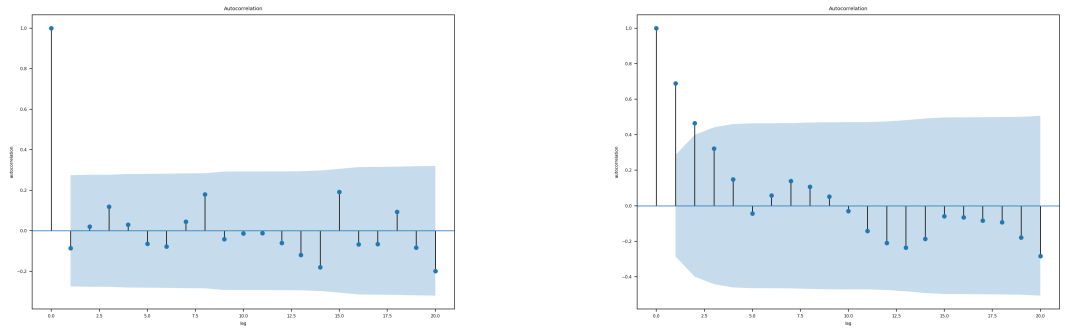
Figure 14: Plot of the DRC's annual percentage real GDP growth, as well as the first PC and SPC with normed factor loadings for the 56 explanatory variables

Table 14: Simulation Outcomes for different specifications of a factor model with uncorrelated and homoscedastic errors and 100 repetitions per specification

lags	T	# factors	AR Config	mean lags	mean PCA facs	mean non-0 β_is EN	mean non-0 β_is BMA0	mean non-0 β_is BMA1	Best model
hlag	T	# factors	AR Config	mean lags	mean PCA facs	mean non-0 betas EN	mean non-0 betas BMA0	mean non-0 betas BMA1	Best model
1	50	3	Config. 1	1.404	3.486	3.288	3.486	3.486	BMA1
	50	3	Config. 2	1.528	3.000	1.388	3.000	3.000	BMA1
	50	3	Config. 3	3.304	14.798	8.608	14.790	14.548	BMA1
	50	5	Config. 1	1.534	7.318	5.956	7.318	7.314	BMA1
	50	5	Config. 2	1.400	5.054	2.970	5.054	5.054	BMA1
	50	5	Config. 3	2.904	19.546	8.540	19.538	18.802	BMA1
	100	3	Config. 1	1.362	4.738	4.268	4.738	4.738	BMA1
	100	3	Config. 2	1.166	3.000	1.772	3.000	3.000	BMA1
	100	3	Config. 3	3.402	15.156	10.718	15.134	14.994	BMA1
	100	5	Config. 1	1.210	8.764	7.238	8.764	8.762	BMA1
	100	5	Config. 2	1.218	5.124	4.142	5.124	5.124	BMA1
	100	5	Config. 3	3.024	19.620	10.136	19.552	18.782	BMA1
	400	3	Config. 1	2.110	10.582	7.580	10.550	10.582	BMA1
	400	3	Config. 2	1.048	3.000	2.442	3.000	3.000	BMA1
	400	3	Config. 3	3.866	1.442	0.390	1.146	1.136	BMA1
	400	5	Config. 1	1.294	11.456	8.754	11.372	11.452	BMA1
	400	5	Config. 2	1.032	5.742	4.870	5.736	5.742	BMA1
	400	5	Config. 3	3.234	1.000	0.062	1.000	1.000	BMA1
5	50	3	Config. 1	1.404	3.454	0.918	3.454	3.454	AR
	50	3	Config. 2	1.528	3.000	0.602	3.000	3.000	AR
	50	3	Config. 3	3.304	14.658	6.308	14.650	14.356	AR
	50	5	Config. 1	1.534	7.322	2.022	7.322	7.322	AR
	50	5	Config. 2	1.400	5.056	1.620	5.056	5.056	EN
	50	5	Config. 3	2.904	19.518	8.488	19.512	18.738	AR
	100	3	Config. 1	1.362	4.762	1.620	4.762	4.762	BMA1
	100	3	Config. 2	1.166	3.000	0.594	3.000	3.000	EN
	100	3	Config. 3	3.402	15.248	8.316	15.226	15.056	BMA1
	100	5	Config. 1	1.210	8.726	1.578	8.724	8.720	EN
	100	5	Config. 2	1.218	5.156	0.618	5.156	5.156	AR
	100	5	Config. 3	3.024	19.706	9.280	19.630	18.852	BMA1
	400	3	Config. 1	2.110	10.608	6.094	10.554	10.608	BMA1
	400	3	Config. 2	1.048	3.000	0.178	2.998	3.000	EN
	400	3	Config. 3	3.866	1.438	0.386	1.254	1.202	BMA0
	400	5	Config. 1	1.294	11.416	3.912	11.288	11.402	BMA0
	400	5	Config. 2	1.032	5.750	0.684	5.740	5.750	AR
	400	5	Config. 3	3.234	1.000	0.010	1.000	1.000	BMA1

Table 15: Mean Squared Predictive Errors for the different simulations specifications. 100 repetitions per specification. MSPE refers to the MSPE ratio vs. an AR(p) model

				Uncorr., Homosc.			Corr., Heterosc.		
lags	T	# factors	AR Config	MSPE EN	MSPE BMA0	MSPE BMA1	MSPE EN	MSPE BMA0	MSPE BMA1
1	50	3	Config. 1	0.697	0.611	0.604	0.697	0.611	0.604
	50	3	Config. 2	0.977	0.988	0.979	0.977	0.988	0.979
	50	3	Config. 3	0.915	0.893	0.912	0.915	0.893	0.912
	50	5	Config. 1	0.683	0.568	0.571	0.683	0.568	0.571
	50	5	Config. 2	0.950	0.940	0.938	0.950	0.940	0.938
	50	5	Config. 3	0.922	0.899	0.872	0.922	0.899	0.872
	100	3	Config. 1	0.721	0.634	0.631	0.721	0.634	0.631
	100	3	Config. 2	0.976	0.986	0.980	0.976	0.986	0.980
	100	3	Config. 3	0.883	0.861	0.872	0.883	0.861	0.872
	100	5	Config. 1	0.650	0.521	0.519	0.650	0.521	0.519
	100	5	Config. 2	0.920	0.929	0.920	0.920	0.929	0.920
	100	5	Config. 3	0.861	0.785	0.784	0.861	0.785	0.784
	400	3	Config. 1	0.689	0.578	0.577	0.689	0.578	0.577
	400	3	Config. 2	0.957	0.942	0.942	0.957	0.942	0.942
	400	3	Config. 3	0.984	1.000	1.000	0.984	1.000	1.000
	400	5	Config. 1	0.626	0.491	0.492	0.626	0.491	0.492
	400	5	Config. 2	0.916	0.885	0.885	0.916	0.885	0.885
	400	5	Config. 3	0.991	0.995	0.983	0.991	0.995	0.983
5	50	3	Config. 1	0.997	1.020	1.026	1.003	1.017	1.019
	50	3	Config. 2	1.021	1.033	1.045	1.015	1.025	1.034
	50	3	Config. 3	1.010	1.111	1.072	1.012	1.048	1.024
	50	5	Config. 1	1.005	1.039	1.034	1.012	1.043	1.043
	50	5	Config. 2	0.993	1.016	1.018	0.998	1.020	1.018
	50	5	Config. 3	1.002	1.038	1.021	1.001	1.064	1.015
	100	3	Config. 1	0.991	0.992	0.993	0.996	0.998	0.999
	100	3	Config. 2	1.000	1.001	1.003	0.997	1.000	1.001
	100	3	Config. 3	0.959	0.953	0.960	0.942	0.912	0.918
	100	5	Config. 1	1.001	1.011	1.009	0.998	1.016	1.016
	100	5	Config. 2	1.005	1.010	1.014	1.003	1.009	1.011
	100	5	Config. 3	0.968	0.947	0.956	0.964	0.948	0.954
	400	3	Config. 1	0.986	0.998	0.992	0.988	1.000	0.997
	400	3	Config. 2	1.000	1.002	1.005	1.000	1.001	1.005
	400	3	Config. 3	0.993	1.000	1.000	0.992	1.000	1.000
	400	5	Config. 1	1.001	0.995	0.996	1.003	1.000	1.001
	400	5	Config. 2	1.004	1.007	1.011	1.005	1.007	1.010
	400	5	Config. 3	1.003	1.000	0.997	1.000	0.999	0.995



(a) Autocorrelations for $h = 1$

(b) Autocorrelations for $h = 5$

Figure 15: Autocorrelations for optimization with PCA and BMA1 for a forcasting horizon of one and five years. 95% Confidence Intervals are based on variances computed with Bartlett's formula

Table 16: Cross-validation statistics for forecast horizons between one and and four years and data gaps between zero and three years between the observed data and the forecasted year

h	factor method	ML method	gap	Mean	Median	Std. Dev.
1	PCA	EN	0	-0.139	-1.846	4.397
	PCA	BMA0	0	-0.280	-2.050	4.688
	PCA	BMA1	0	-0.032	-1.806	4.487
	SPCA	EN	0	-0.197	-2.028	4.445
	SPCA	BMA0	0	-0.116	-1.898	4.406
	SPCA	BMA1	0	-0.116	-1.899	4.389
2	PCA	EN	0	-0.615	-1.203	5.040
	PCA	BMA0	0	-1.001	-2.365	5.359
	PCA	BMA1	0	-0.841	-2.129	5.137
	SPCA	EN	0	-0.412	-1.203	5.181
	SPCA	BMA0	0	-0.432	-1.232	5.119
	SPCA	BMA1	0	-0.413	-1.224	5.159
	PCA	EN	1	-0.101	-1.511	5.256
	PCA	BMA0	1	-0.731	-2.966	5.890
	PCA	BMA1	1	-0.182	-1.510	5.296
	SPCA	EN	1	-0.250	-2.241	5.119
	SPCA	BMA0	1	-0.205	-1.459	5.008
	SPCA	BMA1	1	-0.154	-1.434	5.042
3	PCA	EN	0	-0.459	-2.012	5.784
	PCA	BMA0	0	-0.384	-2.190	5.951
	PCA	BMA1	0	-0.424	-1.988	5.804
	SPCA	EN	0	-0.577	-2.012	5.519
	SPCA	BMA0	0	-0.501	-1.940	5.581
	SPCA	BMA1	0	-0.499	-1.931	5.572
	PCA	EN	1	-0.286	-1.358	5.261
	PCA	BMA0	1	-0.550	-1.925	5.160
	PCA	BMA1	1	0.372	-1.436	7.248
	SPCA	EN	1	-0.345	-1.358	5.164
	SPCA	BMA0	1	-0.280	-1.333	5.137
	SPCA	BMA1	1	-0.251	-1.335	5.171
	PCA	EN	2	-0.245	-1.127	5.272
	PCA	BMA0	2	0.127	-1.182	7.290
	PCA	BMA1	2	-1.091	-1.639	4.908
	SPCA	EN	2	-0.121	-1.006	5.366
	SPCA	BMA0	2	-0.085	-1.078	5.357
	SPCA	BMA1	2	-0.068	-1.097	5.394
4	PCA	EN	0	-0.889	-1.307	5.876
	PCA	BMA0	0	-1.345	-2.118	5.632
	PCA	BMA1	0	-1.208	-1.149	5.411
	SPCA	EN	0	-0.855	-1.217	5.914
	SPCA	BMA0	0	-0.952	-1.237	5.635
	SPCA	BMA1	0	-0.844	-1.242	5.795
	PCA	EN	1	-0.718	-1.158	5.935
	PCA	BMA0	1	-1.088	-2.461	6.215
	PCA	BMA1	1	-1.358	-2.677	5.458
	SPCA	EN	1	-0.714	-1.158	5.565
	SPCA	BMA0	1	-0.831	-1.145	5.443
	SPCA	BMA1	1	-0.734	-1.138	5.603
	PCA	EN	2	-0.916	-2.019	6.021
	PCA	BMA0	2	-1.184	-2.872	6.216
	PCA	BMA1	2	-0.500	-3.127	7.281
	SPCA	EN	2	-0.823	-1.434	5.944
	SPCA	BMA0	2	-0.789	-1.224	5.819
	SPCA	BMA1	2	-0.738	-1.200	5.931
	PCA	EN	3	-0.904	-2.975	6.327
	PCA	BMA0	3	-1.324	-3.786	6.385
	PCA	BMA1	3	-1.057	-3.017	6.107
	SPCA	EN	3	-0.747	-2.975	6.212
	SPCA	BMA0	3	-0.877	-2.865	6.238
	SPCA	BMA1	3	-0.766	-2.862	6.216

Table 17: Cross-validation statistics for a forecast horizon of five years and data gaps between zero and four years between the observed data and the forecasted year

factor method	ML method	gap	Mean	Median	Std. Dev.
PCA	EN	0	-0.574	-1.377	6.412
PCA	BMA0	0	-0.376	-2.463	7.607
PCA	BMA1	0	-0.625	-1.315	6.378
SPCA	EN	0	-0.889	-1.377	5.999
SPCA	BMA0	0	-1.028	-1.299	5.823
SPCA	BMA1	0	-0.781	-1.284	5.997
PCA	EN	1	0.112	-1.422	7.703
PCA	BMA0	1	0.054	-2.378	7.605
PCA	BMA1	1	0.055	-1.680	7.667
SPCA	EN	1	0.342	-0.631	7.956
SPCA	BMA0	1	0.016	-0.965	7.427
SPCA	BMA1	1	0.126	-1.022	7.654
PCA	EN	2	0.105	-1.655	7.880
PCA	BMA0	2	0.234	-3.752	10.662
PCA	BMA1	2	0.126	-2.896	8.426
SPCA	EN	2	0.177	-0.897	7.958
SPCA	BMA0	2	-0.146	-1.173	7.670
SPCA	BMA1	2	-0.060	-1.294	7.731
PCA	EN	3	0.154	-1.625	7.941
PCA	BMA0	3	-0.040	-2.077	8.186
PCA	BMA1	3	0.048	-2.318	9.500
SPCA	EN	3	0.115	-1.490	7.802
SPCA	BMA0	3	-0.152	-1.622	7.555
SPCA	BMA1	3	0.036	-1.618	7.747
PCA	EN	4	0.265	-1.523	7.973
PCA	BMA0	4	-0.295	-3.477	8.504
PCA	BMA1	4	0.857	-1.507	8.840
SPCA	EN	4	0.285	-1.523	7.979
SPCA	BMA0	4	0.055	-1.493	7.640
SPCA	BMA1	4	0.205	-1.481	7.795

Table 18: Break tests for unknown break dates and $\tau = 0.185$ with assumed breaks in the AR process and the factor loadings. P-values refer to the probability of no break for this specific break point

	$h = 1$		$h = 2$		$h = 3$		$h = 4$		$h = 5$	
year	LM stat.	p-value	LM stat.	p-value	LM stat.	p-value	LM stat.	p-value	LM stat.	p-value
1974	13.294	0.004	3.752	0.289	0.749	0.862	1.838	0.607	0.224	0.974
1975	13.844	0.003	5.611	0.132	1.157	0.763	2.132	0.545	1.267	0.737
1976	13.554	0.004	5.114	0.164	1.142	0.767	2.596	0.458	1.457	0.692
1977	9.731	0.021	5.564	0.135	1.684	0.641	2.214	0.529	1.132	0.769
1978	10.851	0.013	3.429	0.330	1.886	0.596	2.557	0.465	1.216	0.749
1979	12.317	0.006	3.457	0.326	1.590	0.662	2.413	0.491	1.283	0.733
1980	13.410	0.004	4.546	0.208	2.051	0.562	2.311	0.510	1.331	0.722
1981	12.848	0.005	4.534	0.209	2.126	0.547	2.266	0.519	1.264	0.738
1982	11.330	0.010	4.908	0.179	2.034	0.565	1.754	0.625	1.264	0.738
1983	11.539	0.009	4.882	0.181	1.126	0.771	1.788	0.618	1.758	0.624
1984	11.113	0.011	4.840	0.184	1.139	0.768	2.179	0.536	2.494	0.476
1985	10.144	0.017	4.907	0.179	1.035	0.793	2.534	0.469	3.021	0.388
1986	10.943	0.012	4.476	0.214	1.063	0.786	2.531	0.470	2.824	0.420
1987	11.356	0.010	4.638	0.200	1.019	0.797	2.425	0.489	2.729	0.435
1988	11.327	0.010	5.101	0.165	1.410	0.703	3.482	0.323	3.206	0.361
1989	11.530	0.009	5.479	0.140	2.525	0.471	5.369	0.147	5.198	0.158
1990	11.179	0.011	4.868	0.182	2.585	0.460	6.280	0.099	7.935	0.047
1991	9.716	0.021	4.412	0.220	3.805	0.283	8.300	0.040	8.778	0.032
1992	6.810	0.078	3.474	0.324	6.446	0.092	8.549	0.036	8.780	0.032
1993	4.412	0.220	4.837	0.184	6.757	0.080	8.527	0.036	8.708	0.033
1994	5.817	0.121	4.935	0.177	5.114	0.164	8.546	0.036	8.730	0.033
1995	5.969	0.113	4.180	0.243	4.795	0.187	8.356	0.039	8.788	0.032
1996	5.692	0.128	4.201	0.241	6.191	0.103	9.132	0.028	10.703	0.013
1997	6.062	0.109	5.951	0.114	7.363	0.061	11.647	0.009	14.882	0.002
1998	5.480	0.140	6.071	0.108	8.168	0.043	13.844	0.003	16.225	0.001
1999	5.649	0.130	6.438	0.092	10.811	0.013	15.485	0.001	16.044	0.001
2000	6.032	0.110	9.182	0.027	12.970	0.005	15.290	0.002	14.724	0.002
2001	7.155	0.067	10.749	0.013	12.220	0.007	13.640	0.003	13.049	0.005
2002	7.301	0.063	9.382	0.025	9.328	0.025	11.444	0.010	11.729	0.008
2003	6.132	0.105	7.380	0.061	6.866	0.076	9.961	0.019	11.105	0.011
2004	4.727	0.193	5.683	0.128	5.680	0.128	9.075	0.028	9.720	0.021
2005	3.691	0.297	4.876	0.181	5.213	0.157	7.704	0.053	8.240	0.041
2006	3.429	0.330	4.597	0.204	4.530	0.210	6.634	0.085	8.329	0.040

Table 19: Forecasts for the DRC's annual real GDP growth and forecasting horizons from one to five years for different models assuming a structural break in 1975

h	forecast for	act. value	IMF estimate	AR	PCA EN	PCA BMA0	PCA BMA1	SPCA EN	SPCA BMA0	SPCA BMA1
1	2014	9.5	10.488	7.142	7.142	7.154	7.161	7.142	7.205	7.229
	2015	6.9	10.250	8.167	8.167	8.179	8.188	8.167	8.195	8.207
	2016	2.4	9.308	5.933	5.933	5.936	5.936	5.933	5.984	5.990
	2017	3.7	6.631	2.153	2.153	2.158	2.159	2.153	2.170	2.176
	2018	-	5.892	3.233	3.233	3.247	3.252	3.233	3.269	3.281
2	2015	6.9	8.489	6.288	6.288	6.286	6.285	6.288	6.336	6.353
	2016	2.4	7.896	6.995	6.995	6.994	6.993	6.995	7.014	7.023
	2017	3.7	7.298	4.961	4.961	4.961	4.960	4.961	4.992	4.998
	2018	-	6.403	2.126	2.126	2.126	2.126	2.126	2.142	2.148
	2019	-	5.707	2.925	2.925	2.926	2.926	2.925	2.934	2.937
3	2016	2.4	7.300	5.616	5.616	5.609	5.605	5.616	5.679	5.701
	2017	3.7	6.800	5.864	5.864	5.860	5.857	5.864	5.874	5.878
	2018	-	6.200	4.351	4.351	4.350	4.350	4.351	4.382	4.389
	2019	-	6.500	2.074	2.074	2.073	2.073	2.074	2.112	2.124
	2020	-	5.200	2.732	2.732	2.730	2.729	2.732	2.751	2.759
4	2017	3.7	4.247	4.374	4.374	4.386	4.393	4.374	4.456	4.485
	2018	-	4.969	4.704	4.704	4.718	4.730	4.704	4.719	4.727
	2019	-	4.865	3.697	3.697	3.698	3.698	3.697	3.709	3.713
	2020	-	5.011	1.953	1.953	1.959	1.960	1.953	1.989	2.002
	2021	-	5.519	2.457	2.457	2.469	2.473	2.457	2.483	2.494
5	2018	-	2.987	3.395	3.687	3.475	3.507	3.964	3.764	3.728
	2019	-	3.273	3.662	3.968	3.770	3.782	4.065	3.854	3.883
	2020	-	3.725	2.967	3.039	2.971	2.972	3.190	3.107	3.104
	2021	-	4.308	1.764	1.921	1.804	1.808	1.856	1.819	1.823
	2022	-	4.702	2.111	2.350	2.174	2.200	2.393	2.288	2.280

Table 20: Forecast Errors for the DRC's annual real GDP growth and forecasting horizons from one to five years for different models assuming a structural break in 1975

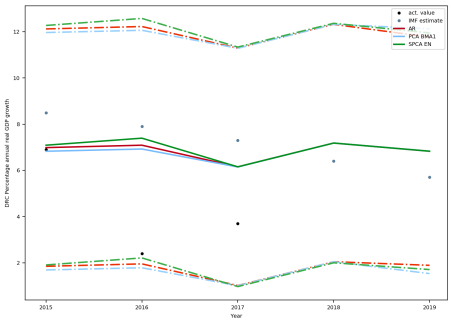
h	forecast for	IMF estimate	AR	PCA EN	PCA BMA0	PCA BMA1	SPCA EN	SPCA BMA0	SPCA BMA1	best model
1	2014	-0.988	2.358	2.358	2.346	2.339	2.358	2.295	2.271	IMF
	2015	-3.350	-1.267	-1.267	-1.279	-1.288	-1.267	-1.295	-1.307	AR
	2016	-6.908	-3.533	-3.533	-3.536	-3.536	-3.533	-3.584	-3.590	AR
	2017	-2.931	1.547	1.547	1.542	1.541	1.547	1.530	1.524	SPCA BMA1
2	2015	-1.589	0.612	0.612	0.614	0.615	0.612	0.564	0.547	SPCA BMA1
	2016	-5.496	-4.595	-4.595	-4.594	-4.593	-4.595	-4.614	-4.623	PCA BMA1
	2017	-3.598	-1.261	-1.261	-1.261	-1.260	-1.261	-1.292	-1.298	PCA BMA1
3	2016	-4.900	-3.216	-3.216	-3.209	-3.205	-3.216	-3.279	-3.301	PCA BMA1
	2017	-3.100	-2.164	-2.164	-2.160	-2.157	-2.164	-2.174	-2.178	PCA BMA1
4	2017	-0.547	-0.674	-0.674	-0.686	-0.693	-0.674	-0.756	-0.785	IMF

Table 21: Forecasts for the DRC's annual real GDP growth and forecasting horizons from one to five years for different models assuming a structural break in 2000

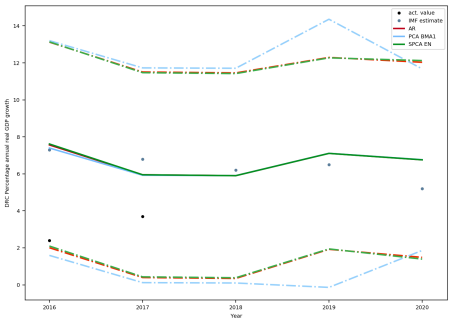
h	forecast for	act. value	IMF estimate	AR	PCA EN	PCA BMA0	PCA BMA1	SPCA EN	SPCA BMA0	SPCA BMA1
1	2014	9.5	10.488	6.558	6.558	6.527	6.523	6.600	6.362	6.411
	2015	6.9	10.250	7.781	7.844	7.595	7.656	8.048	7.369	7.507
	2016	2.4	9.308	6.742	6.947	6.720	6.725	6.907	6.568	6.601
	2017	3.7	6.631	5.382	5.382	5.381	5.381	5.382	5.401	5.403
	2018	-	5.892	5.309	5.309	5.306	5.305	5.309	5.339	5.338
2	2015	6.9	8.489	6.984	7.109	6.745	6.827	7.089	5.994	6.489
	2016	2.4	7.896	7.089	7.167	6.846	6.923	7.393	6.552	6.771
	2017	3.7	7.298	6.153	6.153	6.145	6.146	6.153	5.996	6.020
	2018	-	6.403	7.182	7.182	7.182	7.182	7.182	7.209	7.211
	2019	-	5.707	6.828	6.828	6.830	6.830	6.828	6.811	6.812
3	2016	2.4	7.300	7.563	7.690	7.303	7.396	7.616	7.313	7.378
	2017	3.7	6.800	5.951	5.951	5.923	5.924	5.951	6.023	6.032
	2018	-	6.200	5.902	5.902	5.905	5.904	5.902	5.927	5.925
	2019	-	6.500	7.107	7.107	7.111	7.110	7.107	7.145	7.149
	2020	-	5.200	6.759	6.759	6.766	6.767	6.759	6.807	6.804
4	2017	3.7	4.247	5.701	5.701	5.673	5.671	5.701	5.708	5.708
	2018	-	4.969	4.731	4.731	4.739	4.739	4.731	4.797	4.805
	2019	-	4.865	5.722	5.722	5.723	5.723	5.722	5.774	5.771
	2020	-	5.011	7.438	7.438	7.441	7.441	7.438	7.467	7.469
	2021	-	5.519	6.943	6.943	6.947	6.948	6.943	6.986	6.983
5	2018	-	2.987	3.978	3.978	3.989	3.990	3.978	4.189	4.152
	2019	-	3.273	3.191	3.191	3.202	3.201	3.191	3.284	3.292
	2020	-	3.725	5.237	5.237	5.238	5.238	4.980	5.356	5.336
	2021	-	4.308	8.778	8.778	8.783	8.783	8.778	8.807	8.808
	2022	-	4.702	7.755	7.755	7.765	7.766	7.755	7.864	7.849

Table 22: Forecast Errors for the DRC's annual real GDP growth and forecasting horizons from one to five years for different models assuming a structural break in 2000

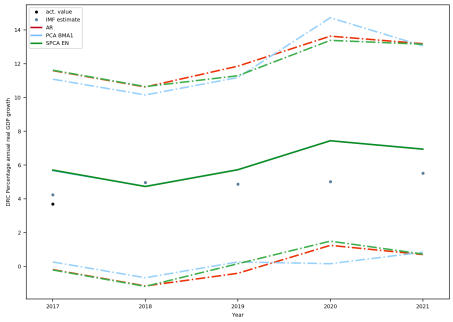
h	forecast for	IMF estimate	AR	PCA EN	PCA BMA0	PCA BMA1	SPCA EN	SPCA BMA0	SPCA BMA1	best model
1	2014	-0.988	2.942	2.942	2.973	2.977	2.900	3.138	3.089	IMF
	2015	-3.350	-0.881	-0.944	-0.695	-0.756	-1.148	-0.469	-0.607	SPCA BMA0
	2016	-6.908	-4.342	-4.547	-4.320	-4.325	-4.507	-4.168	-4.201	SPCA BMA0
	2017	-2.931	-1.682	-1.682	-1.681	-1.681	-1.682	-1.701	-1.703	PCA BMA0
2	2015	-1.589	-0.084	-0.209	0.155	0.073	-0.189	0.906	0.411	PCA BMA1
	2016	-5.496	-4.689	-4.767	-4.446	-4.523	-4.993	-4.152	-4.371	SPCA BMA0
	2017	-3.598	-2.453	-2.453	-2.445	-2.446	-2.453	-2.296	-2.320	SPCA BMA0
3	2016	-4.900	-5.163	-5.290	-4.903	-4.996	-5.216	-4.913	-4.978	IMF
	2017	-3.100	-2.251	-2.251	-2.223	-2.224	-2.251	-2.323	-2.332	PCA BMA0
4	2017	-0.547	-2.001	-2.001	-1.973	-1.971	-2.001	-2.008	-2.008	IMF



(a) Forecasts for $h = 2$



(b) Forecasts for $h = 3$



(c) Forecasts for $h = 4$

Figure 16: 2, 3 and 4-step ahead forecasts of the DRC's real annual percentage GDP growth for optimization with the AR model, BMA1 with PCA factors and EN with SPCA factors against the IMF estimates and the actual values. Dashed-pointed lines give 2-standard deviation (approximately 70%) Confidence Intervals

E. Code Appendix

```
import pickle

import numpy as np

import pandas as pd


def do_pca(X, nrfactors = -1):
    # estimates factors of X via PCA
    import numpy as np
    x_data = np.copy(X)
    rows, cols = x_data.shape
```

```
    if nrfactors == -1:
        rows, nrfactors = x_data.shape
    x_data = standardize(x_data)

    cov = np.cov(x_data, rowvar = False, bias = True)
    eigval, eigvec = np.linalg.eigh(cov)
    eigvec = eigvec * (cols ** 0.5)
    fac = np.matmul(x_data, eigvec) / cols

    return eigval, eigvec[:,-nrfactors : ], (fac[:,-nrfactors : ])


def do_pca_traintest(X, nrfactors = -1):
    # separate estimation of PCA for training and testing samples, both in X
    import numpy as np
    x_data = np.copy(X)
    rows, cols = x_data.shape
    if nrfactors == -1:
        rows, nrfactors = x_data.shape
    x_train = x_data[:-1, :]
    x_train = standardize(x_train)
    x_test = stand_test(x_data[-1, :].reshape((1, cols)), x_train)

    cov = np.cov(x_train, rowvar=False, bias = True)
    eigval, eigvec = np.linalg.eigh(cov)
    eigvec = eigvec * (cols ** 0.5)
    fac_train = x_train.dot(eigvec) / cols
    fac_test = x_test.dot(eigvec) / cols

    return eigval, eigvec[:,-nrfactors : ], (fac_train[:,-nrfactors : ]), (fac_test
     [:,-nrfactors : ])
```

```
def nr_included_factors_pca(X):
    # implementation of Panel IC from Bai and NG(2002) with the Kim and Swanson (2013)
      penalty
    import numpy as np
    x_data = np.copy(X)
    x_data = standardize(x_data)
    T, n = x_data.shape
    max_factors = 20
    IC = np.inf
    min_i = 0
    eigval, eigvec, fac = do_pca(x_data)
    for i in range(1, max_factors + 1):
        fac_use = fac[:, -i: ]
        x_approx = fac_use.dot(np.transpose(eigvec[:, -i: ]))
        error = x_data - x_approx
        errorstat = np.trace(np.transpose(error).dot(error)) / (T*n)
        penalty = (i) * float(n + T) / (n*T) * np.log(min(n, T))
        ICnew = np.log(errorstat) + penalty
        if ICnew <= IC:
            IC = ICnew
            min_i = i

    return min_i


def do_spca(X, r = 0):
    # implementation of SPCA
    import numpy as np
    from sklearn.linear_model import ElasticNetCV
```

```
x_data = np.copy(X)
rows, cols = x_data.shape
if r <= 0:
    r = cols
x_data = standardize(x_data)

eigval, eigvec, fac = do_pca(x_data)
Deltas = eigvec
Lambdas = np.zeros((cols, cols))

convergence_cond = False
looprep = 0
while not convergence_cond:
    # in each loop, optimize Lambda and Delta
    looprep += 1
    #print(looprep)
    Delas_old = Deltas.copy()
    Lambdas_old = Lambdas.copy()
    for j in range(cols):
        X_use = x_data.dot(Deltas[:, j])
        EN = ElasticNetCV(l1_ratio= [1, 0.5], cv = 10, fit_intercept=False,
         alphas=np.arange(1, 100, 1))
        #y_hat_use, beta_use = cv_en(X_use, x_data, etas2 = [0])
        EN.fit(x_data, X_use)
        beta_use = EN.coef_
        Lambdas[:, j] = beta_use

    for_svd = np.dot(np.dot(np.transpose(x_data), x_data), Lambdas)
    u, s, vh = np.linalg.svd(for_svd)
    Deltas = np.dot(u, vh)
```

```
        # some convergence criterion to stop
        Deltas_diff = Deltas - Delas_old
        Lambdas_diff = Lambdas - Lambdas_old
        deltas_ratio = np.sum(np.abs(Deltas_diff)) / np.sum(np.abs(Deltas))
        lambdas_ratio = np.sum(np.abs(Lambdas_diff[:, -r:])) / np.sum(np.abs
         (Lambdas[:, -r:]))

        threshold = 0.001
        if ((looprep > 5) & (deltas_ratio < threshold) & (lambdas_ratio < threshold)):
            indexer = np.sum(np.abs(Lambdas), axis=0)
            Lambdas = Lambdas[:, indexer > 0]
            for coli in range(Lambdas.shape[1]):
                Lambdas[:, coli] /= (Lambdas[:, coli] @ Lambdas[:, coli]) ** 0.5
            fac = np.matmul(x_data, Lambdas)

            if r < Lambdas.shape[1]:
                return Lambdas[:, -r:], fac[:, -r:]
            else:
                return Lambdas, fac

        if looprep == 50:
            indexer = np.sum(np.abs(Lambdas), axis = 0)
            Lambdas = Lambdas[:, indexer > 0]
            for coli in range(Lambdas.shape[1]):
                Lambdas[:, coli] /= (Lambdas[:, coli] @ Lambdas[:, coli]) ** 0.5
            fac = np.matmul(x_data, Lambdas)

            if r < Lambdas.shape[1]:
                return Lambdas[:, -r:], fac[:, -r:]
            else:
                return Lambdas, fac
```

```
def ar_sel(Y, max_lag):
    # select number of lags of AR model
    import numpy as np
    y = np.copy(Y)
    from statsmodels.tsa.ar_model import AR
    ar = AR(y)
    model_fit = ar.fit(maxlag = max_lag, ic = 'bic')
    lags = model_fit.k_ar

    return lags



def ar_reg(Y_train, arlag, hlag, maxarlag, maxhlag):
    # select correct y values, fit AR model
    from sklearn.linear_model import LinearRegression
    import numpy as np
    y_train = np.copy(Y_train)
    n = len(y_train)
    y_train_dep = y_train[maxarlag + maxhlag - 1:]
    y_train_expl = np.ones((len(y_train_dep), arlag))
    if arlag > 0:
        for i in range(arlag):
            y_train_expl[:, i] = y_train[maxarlag + maxhlag - i - hlag - 1:-i - hlag]
    linreg = LinearRegression(fit_intercept=True)
    linreg.fit(y_train_expl, y_train_dep)
    y_hat_train = linreg.predict(y_train_expl)

    y_test_expl = np.ones((1, arlag))
```

```
    if arlag > 0:
        y_test_expl[0, 0] = y_train_dep[-hlag]
        if arlag > 1:
            y_test_expl[0, 1:] = y_train_expl[-1, :-1]
    y_hat_test = linreg.predict(y_test_expl)
    if arlag > 0:
        betas_train = np.append(linreg.intercept_, linreg.coef_)
    else:
        betas_train = [linreg.intercept_]

    return y_train_dep, betas_train, y_hat_train, y_hat_test



def do_en(Dep, Expl, eta1, eta2):
    # implementation of the Elastic Net
    import numpy as np
    from sklearn.linear_model import LassoLars
    dep = np.copy(Dep)
    expl = np.copy(Expl)
    rows, cols = expl.shape
    idStack = np.identity(cols) * (eta2**0.5)
    explPlTr = (1/((1+eta2)**0.5)) * np.concatenate((expl, idStack), axis=0)
    depPlTr = np.concatenate((dep, np.zeros(cols)), axis=0)

    LL = LassoLars(alpha = ((eta1 / (1 + eta2)**0.5)  ), fit_intercept = False)
    LL.fit(explPlTr, depPlTr)
    #LL.fit(expl, dep)
    betas = LL.coef_
    betas *= (1 + eta2)**0.5
    # Lreg = LinearRegression()
    y_hat = np.dot(expl, betas)
```

```
    return y_hat, betas


def do_en_testtrain(dep, expl, expltest, eta1, eta2):
    # estimate EN for training and testing data
    import numpy as np
    from sklearn.linear_model import LassoLars
    rows, cols = expl.shape
    idStack = np.identity(cols) * (eta2**0.5)
    explPlTr = (1/((1+eta2)**0.5)) * np.concatenate((expl, idStack), axis=0)
    depPlTr = np.concatenate((dep, np.zeros(cols)), axis=0)

    LL = LassoLars(alpha = ((eta1 / (1 + eta2)**0.5)  ), fit_intercept = False)
    LL.fit(explPlTr, depPlTr)
    betas = LL.coef_
    betas *= (1 + eta2)**0.5
    y_hat_train = np.dot(expl, betas)
    y_hat_test = np.dot(expltest, betas)

    return y_hat_train, y_hat_test, betas


def do_bma(nburn, nkeep, dep, expl, prior_opt):
    # implementation of BMA
    import numpy as np
    import statsmodels.api as sm
    y = dep.copy()
    X = expl.copy()

    T, k = X.shape
```

```
X -= np.mean(X, axis=0)

X_reg = sm.add_constant(X)
res = sm.OLS(y, X_reg).fit()
t_stats = res.tvalues

molddraw = np.zeros(k + 1)
# MAYBE, MAYBE NOT
molddraw[0] = 1
for i in range(1, k + 1):
    if np.abs(t_stats[i]) > 0.5:
        molddraw[i] = 1

Xold = X_reg[:, molddraw == 1]
kold = np.sum(molddraw)
molddraw = molddraw[1:]

if prior_opt == 1:
    g0 = 1 / (k ** 2)
else:
    g0 = 1 / T
ytynm = (y - np.mean(y)) @ (y - np.mean(y))
xtxinv = np.linalg.inv((Xold.T) @ Xold)
ymy = y @ y - y @ Xold @ xtxinv @ (Xold.T) @ y
g1 = g0 / (g0 + 1)
g2 = 1 / (g0 + 1)
lprobold = 0.5 * kold * np.log(g1) - 0.5 * (T - 1) * np.log(g2 * ymy + g1 * ytynm)

inccount = np.zeros(k)
msize = 0
```

```
b1mo = np.zeros(k)

numrep = nburn + nkeep
for currrep in range(numrep):
    Xnew = np.copy(Xold)
    mnewdraw = np.copy(molddraw)

    rancheck = np.round(np.random.rand() * k).astype(int)
    if rancheck > 0:
        incl_to_here = np.sum(molddraw[:rancheck - 1]).astype('int')
        if molddraw[rancheck - 1] == 1:
            Xnew = np.append(Xold[:, :incl_to_here + 1], Xold[:, incl_to_here + 2:],
            axis=1)
            mnewdraw[rancheck - 1] = 0
        else:
            # Check INdices!!!
            Xnew = np.append(np.append(Xold[:, : incl_to_here + 1], X[:,
             [rancheck - 1]], axis=1),Xold[:, incl_to_here + 1:], axis=1)
            mnewdraw[rancheck - 1] = 1
    knew = np.sum(mnewdraw) + 1
    xtxinv = np.linalg.pinv((Xnew.T) @ Xnew)
    ymy = y @ y - y @ Xnew @ xtxinv @ (Xnew.T) @ y
    lprobnew = 0.5 * knew * np.log(g1) - 0.5 * (T - 1) *
        np.log(g2 * ymy + g1 * ytynm)


    if np.log(np.random.rand())  < (lprobnew - lprobold):
        Xold = Xnew
        lprobold = lprobnew
        molddraw = mnewdraw
        kold = knew
```

```
        if currrep >= nburn:

            inccount += molddraw
            msize += kold

            Q1inv = (1 + g0) * ((Xold.T) @ Xold)
            Q1 = np.linalg.pinv(Q1inv)
            b1 = Q1 @ (Xold.T) @ y
            curr_belem = 0
            # CHECK!!!!
            for i in range(k):
                if molddraw[i] == 1:
                    curr_belem += 1
                    bmean = b1[curr_belem]

                    b1mo[i] += bmean

    b1mo /= nkeep

    return b1mo


def cv_en(Y_use, Xx_use):
    # optimization of EN hyperparameters by CV
    import numpy as np
    y_use = np.copy(Y_use)
    X_use = np.copy(Xx_use)
    cvs = 10
    cv_splitparams = np.floor(np.arange(0, X_use.shape[0] + 0.5,
     X_use.shape[0] / cvs)).astype('int')
```

```
cv_splitparams[-1] -= 1
etas1 = np.append(np.arange(0, 1, 0.1), np.arange(1, 100, 1))
etas2 = [0, 0.01, 0.1, 1, 10, 100, 1000]
IcColl = np.zeros((len(etas1), len(etas2)))



for eta1i in range(len(etas1)):
    eta1 = etas1[eta1i]
    for eta2i in range(len(etas2)):
        eta2 = etas2[eta2i]
        y_hat_test_help = np.zeros(X_use.shape[0])
        y_test_help = np.zeros(X_use.shape[0])
        for i in range(len(cv_splitparams) - 1):
            X_train = np.delete(X_use, np.arange(cv_splitparams[i],
             cv_splitparams[i + 1] + 1), axis=0)
            y_train = np.delete(y_use, np.arange(cv_splitparams[i],
             cv_splitparams[i + 1] + 1))
            X_test = X_use[cv_splitparams[i]: cv_splitparams[i + 1] + 1, :]
            y_test = y_use[cv_splitparams[i]: cv_splitparams[i + 1] + 1]

            y_hat_train, y_hat_test, beta_en = do_en_testtrain(y_train,
             X_train, X_test, eta1, eta2)
            y_hat_test_help[cv_splitparams[i]: cv_splitparams[i + 1] + 1]
                = y_hat_test
            y_test_help[cv_splitparams[i]: cv_splitparams[i + 1] + 1] = y_test

        resid = y_test_help - y_hat_test_help
        sigma2 = np.dot(resid, resid)
        IcColl[eta1i, eta2i] = np.log(sigma2) + (np.sum(beta_en > 0) +
         np.sum(beta_en < 0)) * np.log(X_test.shape[0]) / X_test.shape[0]
        IcColl[eta1i, eta2i] = np.log(sigma2) + (X_test.shape[1]) *
```

```
            np.log(X_test.shape[0]) / X_test.shape[0]

    IC = np.inf
    eta1best = 0
    eta2best = 0
    for eta1i in range(len(etas1)):
        eta1 = etas1[eta1i]
        for eta2i in range(len(etas2)):
            eta2 = etas2[eta2i]
            if IcColl[eta1i, eta2i] <= IC:
                eta1best = eta1
                eta2best = eta2
                IC = IcColl[eta1i, eta2i]

    y_hat_use, beta_use = do_en(y_use, X_use, eta1best, eta2best)
    return y_hat_use, beta_use


def standardize(X_dat):
    # standardize variables
    import numpy as np
    from scipy.stats import zscore
    X = np.copy(X_dat)
    X = zscore(X, axis = 0)
    return X


def stand_test(x_test, x_train):
    # standardize testing and training data
    import numpy as np
    X_test = np.copy(x_test)
```

```
    X_train = np.copy(x_train)
    X_test = X_test.astype(float)
    rows, cols = X_test.shape
    for i in range(cols):
        X_test[:, i] = np.divide(X_test[:, i] - np.mean(X_train[:, i]),
         np.std(X_train[:, i]))
    return X_test


def sort_factors(Facs, fmax):
    # sorting the factors if lags are included
    import numpy as np
    facs = np.copy(Facs)
    if fmax == 1:
        return facs
    rows, cols = facs.shape
    facs_new = np.ones((rows - fmax + 1, fmax * cols))
    for i in range(fmax):
        facs_new[:, i * cols : (i+1) * cols] = facs[fmax - 1 - i:rows-i, :]

    return facs_new


def smooth_data(X_use):
    # smoothing the data
    import numpy as np
    X = np.copy(X_use)
    for j in range(X.shape[1]):
        helper = np.copy(X[:, j])
        cond = False
        while cond == False:
```

```
            hm = np.mean(helper)
            hs = np.std(helper)
            violations = 0
            for i in range(X_use.shape[0]):
                if helper[i] > hm + 3*hs:
                    helper[i] = hm + 3*hs
                    violations += 1
                elif helper[i] < hm - 3*hs:
                    helper[i] = hm - 3 * hs
                    violations += 1
            if violations == 0:
                cond = True
        X[:, j] = helper
    return X


def cal_forecasts(hlag = 1):
    # calculation of the DRC forecasts
    import numpy as np
    import pandas as pd

    np.random.seed(2)

    maxarlag = 4

    data_df = pd.read_excel('research_data.xlsx', index_col='year', parse_dates=[0])
    data = smooth_data(data_df.to_numpy())
    y = data[:, 0]
    XX = data[:, 1:]

    T = data_df.shape[0]
```

```
xlen = 48
break_start = 36

#y = np.arange(1963, 2018)
#XX = np.arange(1963, 2018).reshape(55, 1)

y_hats_ar = np.zeros(T - xlen - maxarlag + 2)
y_hats_pcaen = np.zeros(T - xlen - maxarlag + 2)
y_hats_pcabma0 = np.zeros(T - xlen - maxarlag + 2)
y_hats_pcabma1 = np.zeros(T - xlen - maxarlag + 2)
y_hats_spcaen = np.zeros(T - xlen - maxarlag + 2)
y_hats_spcabma0 = np.zeros(T - xlen - maxarlag + 2)
y_hats_spcabma1 = np.zeros(T - xlen - maxarlag + 2)

for i in range(T - xlen - maxarlag + 2 + - hlag + hlag):
    print('At Forecast: ', i)
    # recursive estimation
    if i < T - xlen - maxarlag + 2 + - hlag + 1:
        y_use = np.copy(y[break_start:i + xlen + maxarlag + hlag - 2])
        ar_lag = ar_sel(y_use[maxarlag + hlag - 2: ], maxarlag)
        print(ar_lag)
        y_train, betas_train, y_hat_train_ar, y_hat_test_ar = ar_reg(y_use,
         ar_lag, hlag, maxarlag, hlag)

        X_use = np.copy(XX[maxarlag + hlag - 1 - hlag: maxarlag + i + xlen - 1,:])
    else:
        y_use = np.copy(y[break_start:])
        ar_lag = ar_sel(y_use[maxarlag + hlag - 2:], maxarlag)
        y_train, betas_train, y_hat_train_ar, y_hat_test_ar = ar_reg(y_use,
         ar_lag, hlag, maxarlag, hlag)
        if ar_lag == 0:
```

```
            y_hat_test_ar = betas_train
        elif ar_lag == 1:
            y_hat_test_ar = betas_train[0] + betas_train[1] * y[maxarlag +
             i + xlen - 2]
        else:
            y_hat_test_ar = betas_train[0]
            for k in range(1, ar_lag + 1):
                y_hat_test_ar += betas_train[k] * y[maxarlag + i + xlen - 1 - k]

        X_use = np.append(np.copy(XX[maxarlag + hlag - 1 - hlag: T - hlag, :]),
                          np.copy(XX[maxarlag + i + xlen - 2, :]).reshape((1,
                           XX.shape[1])), axis=0)

    #print(y_train, X_use)
    # do PCA
    nr_factors_pca = nr_included_factors_pca(X_use)
    eigval, eigvec, facs= do_pca(X_use,nr_factors_pca)
    facs = facs[break_start :, :]

    # do SPCA
    Lambdas, facs_spca = do_spca(X_use, nr_factors_pca)
    facs_spca = facs_spca[break_start :, :]

    #test and train variables, non-adjusted
    resid_train = y_train - y_hat_train_ar
    fac_train = facs[:-1, :]
    fac_test = facs[-1, :]
    fac_train_spca = facs_spca[:-1, :]
    fac_test_spca = facs_spca[-1, :]

    # standardize and center, test and train variables, adjusted
```

```
fac_Adj = standardize(facs)
fac_trainAdj = fac_Adj[:-1, :]
fac_testAdj = fac_Adj[-1, :]
fac_Adj_spca = standardize(facs_spca)
fac_trainAdj_spca = fac_Adj_spca[:-1, :]
fac_testAdj_spca = fac_Adj_spca[-1, :]
resid_trainAdj = resid_train - np.mean(resid_train)

# do EN on residual
y_hat_train_pcaen, minBeta_pcaen = cv_en(resid_trainAdj, fac_trainAdj)
# forecast
y_hat_test_pcaen = fac_testAdj.dot(minBeta_pcaen) + y_hat_test_ar

#BMA
beta_pcbma0 = do_bma(300, 1000, resid_train, fac_train, 0)
beta_pcbma1 = do_bma(300, 1000, resid_train, fac_train, 1)

y_hat_test_pcabma0 = beta_pcbma0 @ fac_test + y_hat_test_ar
y_hat_test_pcabma1 = beta_pcbma1 @ fac_test + y_hat_test_ar



# EN SPCA
y_hat_train_pcaen_spca, minBeta_spcaen = cv_en(resid_trainAdj,
    fac_trainAdj_spca)
# forecast
y_hat_test_pcaen_spca = fac_testAdj_spca.dot(minBeta_spcaen) + y_hat_test_ar

# BMA SPCA
beta_pcbma0_spca = do_bma(300, 1000, resid_train, fac_train_spca, 0)
beta_pcbma1_spca = do_bma(300, 1000, resid_train, fac_train_spca, 1)
```

```
        y_hat_test_pcabma0_spca = beta_pcbma0_spca @ fac_test_spca + y_hat_test_ar
        y_hat_test_pcabma1_spca = beta_pcbma1_spca @ fac_test_spca + y_hat_test_ar

        # save forecasts and check parameter
        y_hats_ar[i] = y_hat_test_ar
        y_hats_pcaen[i] = y_hat_test_pcaen
        y_hats_pcabma0[i] = y_hat_test_pcabma0
        y_hats_pcabma1[i] = y_hat_test_pcabma1
        y_hats_spcaen[i] = y_hat_test_pcaen_spca
        y_hats_spcabma0[i] = y_hat_test_pcabma0_spca
        y_hats_spcabma1[i] = y_hat_test_pcabma1_spca

        # print(np.mean(np.square(resid_train)))
        # print(np.mean(np.square(y_train - (fac_trainAdj.dot(minBeta_pcaen) +
         y_hat_train_ar))))
        # print(np.mean(np.square(y_train - (fac_train @ beta_pcbma0 +
         y_hat_train_ar))))
        # print(np.mean(np.square(y_train - (fac_train @ beta_pcbma1 +
         y_hat_train_ar))))
        # print(np.mean(np.square(y_train - (fac_trainAdj_spca.dot(minBeta_pcaen_spca)
         + y_hat_train_ar))))
        # print(np.mean(np.square(y_train - (fac_train_spca @ beta_pcbma0_spca
         + y_hat_train_ar))))
        # print(np.mean(np.square(y_train - (fac_train_spca @ beta_pcbma1_spca
         + y_hat_train_ar))))


    return [y_hats_ar, y_hats_pcaen, y_hats_pcabma0, y_hats_pcabma1,
     y_hats_spcaen, y_hats_spcabma0, y_hats_spcabma1]
```

```
def factor_ana():
    # analysis of the PCA and SPCA factors
    import numpy as np
    import pandas as pd

    np.random.seed(2)

    maxarlag = 4

    data_df = pd.read_excel('research_data.xlsx', index_col = 'year',
        parse_dates=[0])
    data = smooth_data(data_df.to_numpy())
    y = data[:, 0]
    XX = data[:, 1:]
    #y = data_df.iloc[:, 0].to_numpy()
    #XX = data_df.iloc[:, 1:].to_numpy()

    T = data_df.shape[0]
    xlen = 48

    hlag = 1
    i = 4

    y_use = np.copy(y[:i + xlen + maxarlag + hlag - 2])
    X_use = np.copy(XX[maxarlag + hlag - 1 - hlag: maxarlag + i + xlen - 1, :])
    ar_lag = ar_sel(y_use[maxarlag + hlag - 2:], maxarlag)

    # do PCA
    nr_factors_pca = nr_included_factors_pca(X_use)
    print(nr_factors_pca)
    eigval, eigvec, facs= do_pca(X_use)
```

```
    for coli in range(eigvec.shape[1]):
        eigvec[:, coli] /= (eigvec[:, coli] @ eigvec[:, coli]) ** 0.5
    facs = np.matmul(XX, eigvec)

    # do SPCA
    Lambdas, facs_spca = do_spca(X_use)

    Lambdas_df = pd.DataFrame(Lambdas)
    Eigvec_df = pd.DataFrame(eigvec)
    Eigval_df = pd.DataFrame(eigval)

    Lambdas_df.to_csv('Lambdas.csv')
    Eigvec_df.to_csv('Eigvec.csv')
    Eigval_df.to_csv('Eigval.csv')

    print('First SPCA')
    print(data_df.iloc[:, 1:].columns[Lambdas[:, -1] > 0])
    print('Second SPCA')
    print(data_df.iloc[:, 1:].columns[Lambdas[:, -2] > 0])
    print('Third SPCA')
    print(data_df.iloc[:, 1:].columns[Lambdas[:, -3] > 0])
    #import pickle
    #with open('static_res_forecasts.pickle', 'wb') as output:
    #    pickle.dump([eigval, eigvec, Lambdas], output)


def cal_resids(hlag = 1):
    # calculate the residuals for the results section
    import numpy as np
    import pandas as pd
```

```
np.random.seed(2)

maxarlag = 4

data_df = pd.read_excel('research_data.xlsx', index_col='year',
    parse_dates=[0])
data = smooth_data(data_df.to_numpy())
y = data[:, 0]
XX = data[:, 1:]

T = data_df.shape[0]
xlen = 48

#y = np.arange(1963, 2018)
#XX = np.arange(1963, 2018).reshape(55, 1)

y_hats_ar = np.zeros(T - xlen - maxarlag + 2)
y_hats_pcaen = np.zeros(T - xlen - maxarlag + 2)
y_hats_pcabma0 = np.zeros(T - xlen - maxarlag + 2)
y_hats_pcabma1 = np.zeros(T - xlen - maxarlag + 2)
y_hats_spcaen = np.zeros(T - xlen - maxarlag + 2)
y_hats_spcabma0 = np.zeros(T - xlen - maxarlag + 2)
y_hats_spcabma1 = np.zeros(T - xlen - maxarlag + 2)

i = 4
print('At Forecast: ', i)
# recursive estimation
if i < T - xlen - maxarlag + 2 + - hlag + 1:
    y_use = np.copy(y[:i + xlen + maxarlag + hlag - 2])
    ar_lag = ar_sel(y_use[maxarlag + hlag - 2: ], maxarlag)
    print(ar_lag)
```

```
        y_train, betas_train, y_hat_train_ar, y_hat_test_ar = ar_reg(y_use,
        ar_lag, hlag, maxarlag, hlag)

        X_use = np.copy(XX[maxarlag + hlag - 1 - hlag: maxarlag + i + xlen - 1,:])
    else:
        y_use = np.copy(y)
        ar_lag = ar_sel(y_use[maxarlag + hlag - 2:], maxarlag)
        y_train, betas_train, y_hat_train_ar, y_hat_test_ar = ar_reg(y_use,
        ar_lag, hlag, maxarlag, hlag)
        if ar_lag == 0:
            y_hat_test_ar = betas_train
        elif ar_lag == 1:
            y_hat_test_ar = betas_train[0] + betas_train[1]
                * y[maxarlag + i + xlen - 2]
        else:
            y_hat_test_ar = betas_train[0]
            for k in range(1, ar_lag + 1):
                y_hat_test_ar += betas_train[k] * y[maxarlag + i + xlen - 1 - k]

        X_use = np.append(np.copy(XX[maxarlag + hlag - 1 - hlag: T - hlag, :]),
                          np.copy(XX[maxarlag + i + xlen - 2, :])
                           .reshape((1, XX.shape[1])), axis=0)

    # do PCA
    nr_factors_pca = nr_included_factors_pca(X_use)
    eigval, eigvec, facs= do_pca(X_use,nr_factors_pca)

    # do SPCA
    Lambdas, facs_spca = do_spca(X_use, nr_factors_pca)

    Lambdas = do_spca(X_use, nr_factors_pca)
```

```
#test and train variables, non-adjusted
resid_train = y_train - y_hat_train_ar
fac_train = facs[:-1, :]
fac_test = facs[-1, :]
fac_train_spca = facs_spca[:-1, :]
fac_test_spca = facs_spca[-1, :]

# standardize and center, test and train variables, adjusted
fac_Adj = standardize(facs)
fac_trainAdj = fac_Adj[:-1, :]
fac_testAdj = fac_Adj[-1, :]
fac_Adj_spca = standardize(facs_spca)
fac_trainAdj_spca = fac_Adj_spca[:-1, :]
fac_testAdj_spca = fac_Adj_spca[-1, :]
resid_trainAdj = resid_train - np.mean(resid_train)

# do EN on residual
y_hat_train_pcaen, minBeta_pcaen = cv_en(resid_trainAdj, fac_trainAdj)
# forecast
y_hat_test_pcaen = fac_testAdj.dot(minBeta_pcaen) + y_hat_test_ar

#BMA
beta_pcbma0 = do_bma(300, 1000, resid_train, fac_train, 0)
beta_pcbma1 = do_bma(300, 1000, resid_train, fac_train, 1)

y_hat_test_pcabma0 = beta_pcbma0 @ fac_test + y_hat_test_ar
y_hat_test_pcabma1 = beta_pcbma1 @ fac_test + y_hat_test_ar



# EN SPCA
```

```
y_hat_train_pcaen_spca, minBeta_spcaen = cv_en(resid_trainAdj, fac_trainAdj_spca)
# forecast
y_hat_test_pcaen_spca = fac_testAdj_spca.dot(minBeta_spcaen) + y_hat_test_ar

# BMA SPCA
beta_pcbma0_spca = do_bma(300, 1000, resid_train, fac_train_spca, 0)
beta_pcbma1_spca = do_bma(300, 1000, resid_train, fac_train_spca, 1)

y_hat_test_pcabma0_spca = beta_pcbma0_spca @ fac_test_spca + y_hat_test_ar
y_hat_test_pcabma1_spca = beta_pcbma1_spca @ fac_test_spca + y_hat_test_ar

y_hat_train_pcabma0_spca = fac_train_spca @ beta_pcbma0_spca + y_hat_train_ar
y_hat_train_pcabma1_spca = fac_train_spca @ beta_pcbma1_spca + y_hat_train_ar

# save forecasts and check parameter
e_train_pcaen = y_train - y_hat_train_pcaen
e_train_pcabma0 = y_train - y_hat_train_pcabma0
e_train_pcabma1 = y_train - y_hat_train_pcabma1
e_train_spcaen = y_train - y_hat_train_pcaen_spca
e_train_spcabma0 = y_train - y_hat_train_pcabma0_spca
e_train_spcabma1 = y_train - y_hat_train_pcabma1_spca


e_df = pd.DataFrame({'PCA EN': e_train_pcaen, 'PCA BMA0': e_train_pcabma0,
 'PCA BMA1': e_train_pcabma1,
                     'SPCA EN': e_train_spcaen, 'SPCA BMA0': e_train_spcabma0,
                      'SPCA BMA1': e_train_spcabma1})
filename = 'ferrors' + str(hlag) + '.csv'
e_df.to_csv(filename)
for i in range(e_df.shape[1]):
    print(e_df.columns[i])
```

```
        print(e_df.iloc[:, i].autocorr(lag = 1))
        print(e_df.iloc[:, i].autocorr(lag = 2))


def break_test(hlag = 1, break_at = 35):
    # calculate the break test statistics
    import numpy as np
    import pandas as pd

    np.random.seed(2)

    maxarlag = 4

    data_df = pd.read_excel('research_data.xlsx', index_col='year', parse_dates=[0])
    data = smooth_data(data_df.to_numpy())
    y = data[:, 0]
    XX = data[:, 1:]

    T = data_df.shape[0]
    xlen = 48



    y_dep = y[hlag:]
    y_expl = y[:-hlag]
    #X_expl = XX[:-hlag]

    # do PCA
    #nr_factors_pca = nr_included_factors_pca(X_use)
    eigval, eigvec, facs= do_pca(XX,1)

    # do SPCA
```

```
facs_spca = facs
#Lambdas, facs_spca = do_spca(XX, 1)

facs_expl = facs[:-hlag, :]
facs_spca_expl = facs_spca[:-hlag, :]

from sklearn.linear_model import LinearRegression
reg = LinearRegression(fit_intercept=False)
expl_help = np.append(np.ones((len(y_expl), 1)), y_expl.reshape((len(y_expl),
 1)), axis = 1)
expl_pca = np.append(expl_help, facs_expl, axis = 1)
expl_spca = np.append(expl_help, facs_spca_expl, axis = 1)
reg.fit(expl_pca, y_dep)
error_pca = y_dep - reg.predict(expl_pca)
reg.fit(expl_spca, y_dep)
error_spca = y_dep - reg.predict(expl_spca)

expl_pca2 = np.copy(expl_pca)
expl_pca2[:break_at, :] = 0
expl_pca_testreg = np.append(expl_pca, expl_pca2, axis = 1)
expl_spca2 = np.copy(expl_spca)
expl_spca2[:break_at, :] = 0
expl_spca_testreg = np.append(expl_spca, expl_spca2, axis=1)

reg = LinearRegression(fit_intercept=False)
reg.fit(np.append(expl_pca, expl_pca2, axis = 1), error_pca)
R2_pca = reg.score(np.append(expl_pca, expl_pca2, axis = 1), error_pca)
reg.fit(np.append(expl_spca, expl_spca2, axis = 1), error_spca)
R2_spca = reg.score(np.append(expl_spca, expl_spca2, axis = 1), error_spca)

s_pca = expl_pca.shape[0] * R2_pca
```

```
    s_spca = expl_pca.shape[0] * R2_spca

    from scipy.stats import chi2
    prob_pca = chi2.cdf(s_pca, 3)
    prob_spca = chi2.cdf(s_spca, 3)

    #print(R2_pca, s_pca, prob_pca)
    #print(R2_spca, s_spca, prob_spca)

    return s_pca, prob_pca


def all_break_test():
    #execution of break test for known and unknown break dates
    t0 = 10
    size = 54 - 2*10
    t_fordistr = 10/54
    stats_help = np.zeros(size)
    second_stats = np.zeros((size + 1, 15))
    for i in range(size):
        for j in range(5):
            s_pca, prob_pca = break_test(hlag = j+1, break_at = (t0 + i))
            second_stats[i + 1, j*3 : (j+1)*3] = [1964 + t0 + i, s_pca, prob_pca]
    for i in range(5):
        second_stats[-1, i*3 : (i+1)*3] = second_stats[np.argmax(second_stats
         [:-1, i* 3 + 1]), i*3 : (i+1)*3]

    stats1_df = pd.DataFrame(first_stats)
    stats2_df = pd.DataFrame(second_stats)

    stats1_df.to_csv('break_test1.csv')
```

```
    stats2_df.to_csv('break_test2/.csv')



def cal_cv_forecasterrors(hlag=1):
    # CV forecast error estimates
    import numpy as np
    import pandas as pd

    np.random.seed(2)

    maxarlag = 4

    data_df = pd.read_excel('research_data.xlsx', index_col='year', parse_dates=[0])
    data = smooth_data(data_df.values)
    y = data[:, 0]
    XX = data[:, 1:]

    T = data_df.shape[0]
    xlen = 48
    startsize = 25

    for_fill = np.zeros((T - startsize, hlag))
    e_coll = [np.copy(for_fill), np.copy(for_fill), np.copy(for_fill),
     np.copy(for_fill), np.copy(for_fill),
              np.copy(for_fill), np.copy(for_fill)]

    for lag_cons in range(hlag):
        for i in range(T - startsize):
            if i % 10 == 0:
                print('at lag: ', i)
            # recursive estimation
```

```
if lag_cons == 0:
    y_use = np.copy(y[:i + startsize])
    ar_lag = ar_sel(y_use[maxarlag + hlag - 2:], maxarlag)
    y_train, betas_train, y_hat_train_ar, y_hat_test_ar = ar_reg(y_use,
     ar_lag, hlag, maxarlag, hlag)
    X_use = np.copy(XX[maxarlag - 1: i + startsize - hlag + 1, :])
    y_test = np.copy(y[i + startsize])
else:
    y_use = np.copy(y[:i + startsize - lag_cons])
    ar_lag = ar_sel(y_use[maxarlag + hlag - 2:i + startsize - lag_cons],
     maxarlag)
    y_train, betas_train, y_hat_train_ar, y_hat_test_ar = ar_reg(y_use,
     ar_lag, hlag, maxarlag, hlag)
    if ar_lag == 0:
        y_hat_test_ar = betas_train
    elif ar_lag == 1:
        y_hat_test_ar = betas_train[0] + betas_train[1] * y[i +
            startsize - 1]
    else:
        y_hat_test_ar = betas_train[0]
        # print(y.shape)
        # print(ar_lag)
        # print(i + startsize)
        # print(betas_train[ar_lag])
        for k in range(1, ar_lag + 1):
            y_hat_test_ar += betas_train[k] * y[i + startsize - k]
    y_test = np.copy(y[i + startsize])
    X_use = np.append(np.copy(XX[maxarlag - 1: i + startsize - hlag -
     lag_cons, :]),np.copy(XX[i + startsize - hlag, :]).reshape((1,
                        XX.shape[1])), axis=0)
```

```
# print(y_train, X_use)
# do PCA
nr_factors_pca = nr_included_factors_pca(X_use)
eigval, eigvec, facs = do_pca(X_use, nr_factors_pca)

# do SPCA
Lambdas, facs_spca = do_spca(X_use, nr_factors_pca)

# test and train variables, non-adjusted
resid_train = y_train - y_hat_train_ar
fac_train = facs[:-1, :]
fac_test = facs[-1, :]
fac_train_spca = facs_spca[:-1, :]
fac_test_spca = facs_spca[-1, :]

# standardize and center, test and train variables, adjusted
fac_Adj = standardize(facs)
fac_trainAdj = fac_Adj[:-1, :]
fac_testAdj = fac_Adj[-1, :]
fac_Adj_spca = standardize(facs_spca)
fac_trainAdj_spca = fac_Adj_spca[:-1, :]
fac_testAdj_spca = fac_Adj_spca[-1, :]
resid_trainAdj = resid_train - np.mean(resid_train)

# do EN on residual
y_hat_train_pcaen, minBeta_pcaen = cv_en(resid_trainAdj, fac_trainAdj)
# forecast
y_hat_test_pcaen = fac_testAdj.dot(minBeta_pcaen) + y_hat_test_ar

# BMA
beta_pcbma0 = do_bma(300, 1000, resid_train, fac_train, 0)
```

```
            beta_pcbma1 = do_bma(300, 1000, resid_train, fac_train, 1)

            y_hat_test_pcabma0 = beta_pcbma0 @ fac_test + y_hat_test_ar
            y_hat_test_pcabma1 = beta_pcbma1 @ fac_test + y_hat_test_ar

            # EN SPCA
            y_hat_train_pcaen_spca, minBeta_pcaen_spca = cv_en(resid_trainAdj,
             fac_trainAdj_spca)
            y_hat_test_pcaen_spca = fac_testAdj_spca.dot(minBeta_pcaen_spca) +
             y_hat_test_ar

            # BMA SPCA
            beta_pcbma0_spca = do_bma(300, 1000, resid_train, fac_train_spca, 0)
            beta_pcbma1_spca = do_bma(300, 1000, resid_train, fac_train_spca, 1)

            y_hat_test_pcabma0_spca = beta_pcbma0_spca @ fac_test_spca + y_hat_test_ar
            y_hat_test_pcabma1_spca = beta_pcbma1_spca @ fac_test_spca + y_hat_test_ar

            # save forecasts and check parameter
            e_coll[0][i, lag_cons] = y_hat_test_ar - y_test
            e_coll[1][i, lag_cons] = y_hat_test_pcaen - y_test
            e_coll[2][i, lag_cons] = y_hat_test_pcabma0 - y_test
            e_coll[3][i, lag_cons] = y_hat_test_pcabma1 - y_test
            e_coll[4][i, lag_cons] = y_hat_test_pcaen_spca - y_test
            e_coll[5][i, lag_cons] = y_hat_test_pcabma0_spca - y_test
            e_coll[6][i, lag_cons] = y_hat_test_pcabma1_spca - y_test

    for_fill = np.zeros(15)
    e_stats = [np.copy(for_fill), np.copy(for_fill), np.copy(for_fill),
     np.copy(for_fill), np.copy(for_fill),
               np.copy(for_fill), np.copy(for_fill)]
```

```
    for set in range(7):
        # stats: 5 means, 5 medians, 5 stds, for all: all 5 for ar, then for en, ...
        e_stats[set][0: 6 - hlag] = np.mean(e_coll[set][:, 0], axis=0)
        e_stats[set][5: 11 - hlag] = np.median(e_coll[set][:, 0], axis=0)
        e_stats[set][10: 6 - hlag] = np.std(e_coll[set][:, 0], axis=0)
        if hlag > 1:
            for iter in range(1, hlag):
                e_stats[set][5 - hlag + iter] = np.mean(e_coll[set][:, iter], axis=0)
                e_stats[set][10 - hlag + iter] = np.median(e_coll[set][:, iter], axis=0)
                e_stats[set][15 - hlag + iter] = np.std(e_coll[set][:, iter], axis=0)

    return e_stats


def simulation(reps=25, T=200, nvars=51, nfactors=2, xlen=100, ar_spec=2,
maxfaclag=1, hlag=1, delta1=0, delta2=0,
               a_errors=0, b_errors=0):
    import numpy as np
    import matplotlib.pyplot as plt

    np.random.seed(2)

    maxarlag = 4
    maxhlag = 5
    xlen = T - 12
    # import pandas as pd
    # data_df = pd.read_excel('testdata.xlsx')
    # # data_df = data_df.drop(['Eq Guinea', 'Liberia'], axis = 1)
    # T = data_df.shape[0]
    # xlen = 170
    errors_ar = np.zeros((reps, T - xlen - maxarlag - maxhlag + 2))
```

```
errors_pcaen = np.zeros((reps, T - xlen - maxarlag - maxhlag + 2))
errors_pcabma0 = np.zeros((reps, T - xlen - maxarlag - maxhlag + 2))
errors_pcabma1 = np.zeros((reps, T - xlen - maxarlag - maxhlag + 2))
errors_reg = np.zeros((reps, T - xlen - maxarlag - maxhlag + 2))
arlag_coll = np.zeros((reps, T - xlen - maxarlag - maxhlag + 2))
inclfactors_coll = np.zeros((reps, T - xlen - maxarlag - maxhlag + 2))
nonzerobetaPcaEn_coll = np.zeros((reps, T - xlen - maxarlag - maxhlag + 2))
nonzerobetaPcaBma0_coll = np.zeros((reps, T - xlen - maxarlag - maxhlag + 2))
nonzerobetaPcaBma1_coll = np.zeros((reps, T - xlen - maxarlag - maxhlag + 2))

if ar_spec == 0:
    if nfactors == 3:
        factor_coefs = np.array([0.8, 0.65, -0.7])
    else:
        factor_coefs = np.array([0.8, 0.65, -0.7, 0.7, -0.8])
elif ar_spec == 1:
    if nfactors == 3:
        factor_coefs = np.array([0.3, 0.2, -0.25])
    else:
        factor_coefs = np.array([0.3, 0.2, -0.25, 0.25, -0.3])
else:
    if nfactors == 3:
        factor_coefs = np.array([[0.6, 0.5, -0.5], [0.2, 0.3, -0.2],
         [0.1, -0.2, 0.3]])
    else:
        factor_coefs = np.array(
            [[0.6, 0.5, -0.5, 0.3, -0.4], [0.2, 0.3, -0.2, 0.55, -0.4],
             [0.1, -0.2, 0.3, 0.1, -0.1]])

if nfactors == 3:
    factodep = np.array([0.8, -0.5, 0.7])
```

```
    else:
        factodep = np.array([0.8, -0.5, 0.7, 0.9, -0.55])
    factovar = np.random.normal(size=(nfactors, nvars))

    for repitition in range(reps):
        print(repitition)

        factorerrors = np.random.normal(0, 1, size=(T + 9, nfactors))
        factors = np.zeros((T + 11, nfactors))
        factors[: 3, :] = 5 + np.random.normal(3, 1, (3, nfactors))
        for i in range(3, T + 11):
            if ar_spec < 2:
                factors[i, :] = factors[i - 1, :] @ factor_coefs + factorerrors[i - 2, :]
            else:
                factors[i, :] = factors[i - 1, :] @ factor_coefs[0] +
                 factors[i - 2, :] @ factor_coefs[1] \
                                + factors[i - 3, :] @ factor_coefs[2] +
                                 factorerrors[i - 2, :]
        factors = factors[11:, :]
        etas = np.random.normal(0, 1, (T, nvars + 1))
        delta0 = 1 / (1 - delta1 - delta2)

        vs = np.zeros((T, nvars + 1))
        sigmassq = np.zeros((T, nvars + 1))
        sigmassq[0, :] = 1
        vs[0, :] = np.multiply(sigmassq[0, :], etas[0, :])
        for vsi in range(1, T):
            sigmassq[vsi, :] = delta0 + delta1 * sigmassq[vsi - 1, :] + delta2 *
             np.square(vs[vsi - 1, :])
            vs[vsi, :] = np.multiply(np.sqrt([sigmassq[vsi, :]]), etas[vsi, :])
```

```
variserrors = np.zeros((T, nvars + 1))
for varsj in range(nvars):
    variserrors[0, varsj] = (1 + b_errors ** 2) * vs[0, varsj]
     + b_errors * vs[0, varsj + 1] +b_errors * vs[ 0, varsj - 1]
variserrors[0, -1] = (1 + b_errors ** 2) * vs[0, -1] + b_errors * vs[0, 0] +
 b_errors * vs[0, -2]
for varsi in range(1, T):
    for varsj in range(nvars):
        variserrors[varsi, varsj] = a_errors * variserrors[varsi - 1, varsj]
         + (1 + b_errors ** 2) * vs[varsi, varsj] + b_errors
         * vs[varsi, varsj + 1] + b_errors * vs[varsi, varsj - 1]
    variserrors[varsi, -1] = a_errors * variserrors[varsi - 1, -1]
    + (1 + b_errors ** 2) * vs[
        varsi, -1] + b_errors * vs[varsi, 0]
        + b_errors * vs[varsi, -2]

errorsy = variserrors[:, 0]  # .reshape((T, 1))
errorsXX = variserrors[:, 1:]
if nfactors == 1:
    XX = np.outer(factors, factovar) + errorsXX
    y = factors @ factodep + errorsy
else:
    XX = factors @ factovar + errorsXX
    y = factors @ factodep + errorsy

y_tested = np.zeros(T - xlen - maxarlag - maxhlag + 2)
y_hats_ar = np.zeros(T - xlen - maxarlag - maxhlag + 2)
y_hats_pcaen = np.zeros(T - xlen - maxarlag - maxhlag + 2)
y_hats_pcabma0 = np.zeros(T - xlen - maxarlag - maxhlag + 2)
y_hats_pcabma1 = np.zeros(T - xlen - maxarlag - maxhlag + 2)
y_hat_reg = np.zeros(T - xlen - maxarlag - maxhlag + 2)
```

```
for i in range(T - xlen - maxarlag - maxhlag + 2):
    # print(i)
    # recursive estimation
    y_use = np.copy(y[:i + xlen + maxarlag + maxhlag - 2])
    ar_lag = ar_sel(y_use[maxarlag + maxhlag - 2:], maxarlag)
    y_train, betas_train, y_hat_train_ar, y_hat_test_ar =
    ar_reg(y_use, ar_lag, hlag, maxarlag, maxhlag)
    y_test = np.copy(y[i + xlen + maxarlag + maxhlag - 2])

    X_use = np.copy(XX[maxarlag + maxhlag - maxfaclag - hlag:
    maxhlag + maxarlag + i + xlen - 1 - hlag, :])

    # do PCA
    nr_factors_pca = nr_included_factors_pca(X_use)
    eigval, eigvec, facs = do_pca(X_use, nr_factors_pca)
    # facs = sort_factors(facs, maxfaclag)

    # test and train variables, non-adjusted
    resid_train = y_train - y_hat_train_ar
    fac_train = facs[:-1, :]
    fac_test = facs[-1, :]

    # standardize and center, test and train variables, adjusted
    fac_Adj = standardize(facs)
    fac_trainAdj = fac_Adj[:-1, :]
    fac_testAdj = fac_Adj[-1, :]
    resid_trainAdj = resid_train - np.mean(resid_train)

    # do EN on residual
    y_hat_train_pcaen, minBeta_pcaen = cv_en(resid_trainAdj, fac_trainAdj)
```

```
            # forecast
            y_hat_test_pcaen = fac_testAdj.dot(minBeta_pcaen) + y_hat_test_ar

            # BMA
            beta_pcbma0 = do_bma(300, 1000, resid_train, fac_train, 0)
            beta_pcbma1 = do_bma(300, 1000, resid_train, fac_train, 1)

            y_hat_test_pcabma0 = beta_pcbma0 @ fac_test + y_hat_test_ar
            y_hat_test_pcabma1 = beta_pcbma1 @ fac_test + y_hat_test_ar

            # save forecasts and check parameters
            y_tested[i] = y_test
            y_hats_ar[i] = y_hat_test_ar
            y_hats_pcaen[i] = y_hat_test_pcaen
            y_hats_pcabma0[i] = y_hat_test_pcabma0
            y_hats_pcabma1[i] = y_hat_test_pcabma1
            y_hat_reg[i] = y_hat_test_reg

            arlag_coll[repitition, i] = ar_lag
            inclfactors_coll[repitition, i] = nr_factors_pca
            nonzerobetaPcaEn_coll[repitition, i] = np.sum(minBeta_pcaen != 0)
            nonzerobetaPcaBma0_coll[repitition, i] = np.sum(beta_pcbma0 != 0)
            nonzerobetaPcaBma1_coll[repitition, i] = np.sum(beta_pcbma1 != 0)

        errors_ar[repitition, :] = y_tested - y_hats_ar
        errors_pcaen[repitition, :] = y_tested - y_hats_pcaen
        errors_pcabma0[repitition, :] = y_tested - y_hats_pcabma0
        errors_pcabma1[repitition, :] = y_tested - y_hats_pcabma1

    return [arlag_coll, inclfactors_coll, nonzerobetaPcaEn_coll,
    nonzerobetaPcaBma0_coll, nonzerobetaPcaBma1_coll,
```

```
            errors_ar, errors_pcaen, errors_pcabma0, errors_pcabma1]


def all_simulations():
    # run all simulations in this paper
    import pickle
    import numpy as np

    Ts = [50, 100, 400]
    nvars = 50
    nfactors = [3, 5]
    results1 = list()
    results5 = list()

    for ti in Ts:
        print(ti)
        for nfactorsi in nfactors:
            print(nfactorsi)
            for ar_lagsi in range(3):
                print(ar_lagsi)
                results1.append(
                    simulation(reps=100, T=ti, nvars=50, nfactors=nfactorsi,
                    xlen=20, ar_spec=ar_lagsi, maxfaclag=1,
                               hlag=1, delta1=0, delta2=0, a_errors=0, b_errors=0))

    with open('simulation_lag1static.pickle', 'wb') as output:
        pickle.dump(results1, output)

    for ti in Ts:
        print(ti)
        for nfactorsi in nfactors:
```

```
            print(nfactorsi)
            for ar_lagsi in range(3):
                print(ar_lagsi)
                results5.append(
                    simulation(reps=100, T=ti, nvars=50, nfactors=nfactorsi, xlen=20,
                     ar_spec=ar_lagsi, maxfaclag=1,hlag=5, delta1=0, delta2=0,
                      a_errors=0, b_errors=0))

    with open('simulation_lag5static.pickle', 'wb') as output:
        pickle.dump(results5, output)

    for ti in Ts:
        print(ti)
        for nfactorsi in nfactors:
            print(nfactorsi)
            for ar_lagsi in range(3):
                print(ar_lagsi)
                results1.append(
                    simulation(reps=100, T=ti, nvars=50, nfactors=nfactorsi,
                    xlen=20, ar_spec=ar_lagsi, maxfaclag=1,hlag=1, delta1=0.25,
                    delta2=0.05, a_errors=0.1, b_errors=0.2))

    with open('simulation_lag1staticCorrHet.pickle', 'wb') as output:
        pickle.dump(results1, output)

    for ti in Ts:
        print(ti)
        for nfactorsi in nfactors:
            print(nfactorsi)
            for ar_lagsi in range(3):
                print(ar_lagsi)
```

```
                results5.append(
                    simulation(reps=100, T=ti, nvars=50, nfactors=nfactorsi, xlen=20,
                    ar_spec=ar_lagsi, maxfaclag=1,hlag=5, delta1=0.25, delta2=0.05,
                     a_errors=0.1, b_errors=0.2))

    with open('simulation_lag5staticCorrHet.pickle', 'wb') as output:
        pickle.dump(results5, output)



def smooth_data(X_use):
    X = np.copy(X_use)
    for j in range(X.shape[1]):
        helper = np.copy(X[:, j])
        cond = False
        while cond == False:
            hm = np.mean(helper)
            hs = np.std(helper)
            violations = 0
            for i in range(X_use.shape[0]):
                if helper[i] > hm + 3 * hs:
                    helper[i] = hm + 3 * hs
                    violations += 1
                elif helper[i] < hm - 3 * hs:
                    helper[i] = hm - 3 * hs
                    violations += 1
            if violations == 0:
                cond = True
        X[:, j] = helper
    return X
```

```
def sim_to_csv_wen():
    # create a table from the simulation results
    hlags = [1, 3, 5]
    Ts = [50, 100, 400]
    nfactors = [1, 5]
    ar_configs = ['Config. 1', 'Config. 2', 'Config. 3']
    numbers = np.zeros((54, 12))
    models = ['AR', 'EN', 'BMA0', 'BMA1', 'Reg']
    best_model = ['a']*54
    ar_config_used = ['a'] * 54
    counter = 0
    for hlagi in range(3):
        filename = 'simulation_lag' + str(hlags[hlagi]) + 'static.pickle'
        with open(filename, 'rb') as data:
            dataset = pickle.load(data)



        for ti in Ts:
            for nfaci in nfactors:
                for ar_lagsi in range(3):
                    dat = dataset[counter - hlagi * len(dataset)]
                    numbers[counter, 0] = hlags[hlagi]
                    numbers[counter, 1] = ti
                    numbers[counter, 2] = nfaci
                    ar_config_used[counter] = ar_configs[ar_lagsi]
                    for jl in range(6):
                        numbers[counter, jl + 3] = np.mean(np.mean(dat[jl]))

                    min_jl = 0
                    min_ratio = 1
                    comp = np.mean(np.mean(np.square(dat[5])))
```

```
                    for jl in range(6, len(dat)):
                        numbers[counter, jl + 2] = np.mean(np.mean(
                            np.square(dat[jl]))) / comp
                        if np.mean(np.mean(np.square(dat[jl]))) / comp <= min_ratio:
                            min_jl = jl - 5
                    best_model[counter] = models[min_jl]
                    counter += 1

    simu_results = pd.DataFrame(numbers, columns = ['hlag', 'T', '# factors',
     'mean lags', 'mean PCA facs','mean non-0 betas EN', 'mean non-0 betas BMA0',
              'mean non-0 betas BMA1', 'MSPE ratio EN', 'MSPE ratio BMA0',
                 'MSPE ratio BMA1', 'MSPE ratio EN'])
    simu_results['AR Config'] = ar_config_used
    simu_results['Best model'] = best_model

    simu_results = simu_results.iloc[:, [0, 1, 2, 12, 3, 4, 5, 6, 7, 13, 11, 8, 9, 10]]
    simu_results.to_csv('Simulation results.csv')


def sim_to_csv_woen():
    # table without elastic net
    hlags = [1, 5]
    Ts = [50, 100, 400]
    nfactors = [3, 5]
    ar_configs = ['Config. 1', 'Config. 2', 'Config. 3']
    numbers = np.zeros((36, 11))
    models = ['AR', 'EN', 'BMA0', 'BMA1']
    best_model = ['a']*36
    ar_config_used = ['a'] * 36
    counter = 0
    for hlagi in range(2):
```

```
    filename = 'simulation_lag' + str(hlags[hlagi]) + 'staticCorrHet.pickle'
    with open(filename, 'rb') as data:
        dataset = pickle.load(data)



    for ti in Ts:
        for nfaci in nfactors:
            for ar_lagsi in range(3):
                dat = dataset[counter - hlagi * len(dataset)]
                numbers[counter, 0] = hlags[hlagi]
                numbers[counter, 1] = ti
                numbers[counter, 2] = nfaci
                ar_config_used[counter] = ar_configs[ar_lagsi]
                for jl in range(6):
                    numbers[counter, jl + 3] = np.mean(np.mean(dat[jl]))


                min_jl = 0
                min_ratio = 1
                comp = np.mean(np.mean(np.square(dat[5])))
                for jl in range(6, len(dat)):
                    numbers[counter, jl + 2] = np.mean(np.mean
                     (np.square(dat[jl]))) / comp
                    if np.mean(np.mean(np.square(dat[jl]))) / comp <= min_ratio:
                        min_jl = jl - 5
                best_model[counter] = models[min_jl]
                counter += 1

simu_results = pd.DataFrame(numbers, columns = ['hlag', 'T', '# factors',
 'mean lags', 'mean PCA facs','mean non-0 betas EN', 'mean non-0 betas BMA0',
     'mean non-0 betas BMA1','MSPE ratio EN', 'MSPE ratio BMA0', 'MSPE ratio BMA1'])
simu_results['AR Config'] = ar_config_used
```

```
    simu_results['Best model'] = best_model

    simu_results = simu_results.iloc[:, [0, 1, 2, 11, 3, 4, 5, 6, 7, 12, 8, 9, 10]]
    simu_results.to_csv('Simulation results.csv')



def transformed_data():
    # create a data series with the adjusted data set
    data_df1 = pd.read_excel('research_data.xlsx', index_col='year', parse_dates=[0])
    data = data_df1.to_numpy()
    data = smooth_data(data)
    data_df = pd.DataFrame(data, index=data_df1.index, columns=data_df1.columns)
    data_df.to_csv('adjustd_data.csv')



def cv_tables():
    # create the csv tables
    import pickle
    import numpy as np
    with open('cv_pca_staticforec.pickle', 'rb') as data:
        stats_coll = pickle.load(data)

    print(stats_coll[1])

    nums = np.zeros((90, 5))
    nums_short = np.zeros((36, 5))
    nums_mid = np.zeros((54, 5))
    facmeths = ['a']*90
    facmeths_short = ['a']*36
    facmeths_mid = ['a'] * 54
```

```
mlmeths = ['a'] * 90
mlmeths_short = ['a'] * 36
mlmeths_mid = ['a'] * 54
facmeths_list = ['PCA', 'SPCA']
mlmeths_list = ['EN', 'BMA0', 'BMA1']

counter = 0
counter_short = 0
counter_mid = 0
for i in range(5):
    for j in range(i + 1):

        for k in range(6):
            nums[counter, 0] = i+1
            nums[counter, 1] = j
            nums[counter, 2] = stats_coll[k+1][i, 4 - i + j]
            nums[counter, 3] = stats_coll[k+1][i, 9 - i + j]
            nums[counter, 4] = stats_coll[k+1][i, 14 - i + j]

            facmeths[counter] = facmeths_list[np.floor(k / 3).astype(int)]
            mlmeths[counter] = mlmeths_list[int(k % 3)]
            counter += 1

            if ((i == 0) | (i == 4)):
                nums_short[counter_short, 0] = i + 1
                nums_short[counter_short, 1] = j
                nums_short[counter_short, 2] = stats_coll[k+1][i, 4 - i + j]
                nums_short[counter_short, 3] = stats_coll[k+1][i, 9 - i + j]
                nums_short[counter_short, 4] = stats_coll[k+1][i, 14 - i + j]

                facmeths_short[counter_short] = facmeths_list[np.floor(k / 3).
```

```
                    astype(int)]
                mlmeths_short[counter_short] = mlmeths_list[int(k % 3)]
                counter_short += 1

            else:
                nums_mid[counter_mid, 0] = i + 1
                nums_mid[counter_mid, 1] = j
                nums_mid[counter_mid, 2] = stats_coll[k + 1][i, 4 - i + j]
                nums_mid[counter_mid, 3] = stats_coll[k + 1][i, 9 - i + j]
                nums_mid[counter_mid, 4] = stats_coll[k + 1][i, 14 - i + j]

                facmeths_mid[counter_mid] = facmeths_list[np.floor(k / 3).astype(int)]
                mlmeths_mid[counter_mid] = mlmeths_list[int(k % 3)]
                counter_mid += 1

cv_res_short = pd.DataFrame(nums_short, columns = ['$h$', 'gap', 'Mean',
 'Median', 'Std. Dev.'])
cv_res_short['factor method'] = facmeths_short
cv_res_short['ML method'] = mlmeths_short
cv_res_short = cv_res_short.iloc[:, [0, 5, 6, 1, 2, 3, 4]]
cv_res_short.to_csv('CVResShort.csv')

cv_res = pd.DataFrame(nums, columns=['$h$', 'gap', 'Mean', 'Median', 'Std. Dev.'])
cv_res['factor method'] = facmeths
cv_res['ML method'] = mlmeths
cv_res = cv_res.iloc[:, [0, 5, 6, 1, 2, 3, 4]]
cv_res.to_csv('CVRes.csv')

cv_res_mid = pd.DataFrame(nums_mid, columns=['$h$', 'gap', 'Mean',
 'Median', 'Std. Dev.'])
cv_res_mid['factor method'] = facmeths_mid
```

```
    cv_res_mid['ML method'] = mlmeths_mid
    cv_res_mid = cv_res_mid.iloc[:, [0, 5, 6, 1, 2, 3, 4]]
    cv_res_mid.to_csv('CVResMid.csv')



def forecast_comp():
    # create the forecast tables
    import pickle
    import pandas as pd
    import numpy as np
    import numpy as np
    with open('static_res_forecasts_break38.pickle', 'rb') as data:
        forecasts = pickle.load(data)

    imf_df = pd.read_excel('imf estimates.xlsx', index_col = [0], parse_dates = [0])
    print(imf_df)
    data_df = pd.read_excel('research_data.xlsx', index_col='year', parse_dates=[0])

    all_forecasts = np.zeros((25, 11))
    all_ferrors = np.zeros((10, 10))
    best_model = ['a']*10
    fmodels = ['IMF', 'AR', 'PCA EN', 'PCA BMA0', 'PCA BMA1', 'SPCA EN',
     'SPCA BMA0', 'SPCA BMA1']
    counter = 0
    e_counter = 0
    for i in range(5):
        print(i)
        for j in range(5):
            all_forecasts[counter, 0] = i + 1
            all_forecasts[counter, 1] = 2014 + i + j
            if i + j < 4:
```

```
            all_forecasts[counter, 2] = data_df.iloc[-4 + (i+j), 0]
        all_forecasts[counter, 3] = imf_df.iloc[i+j, i]
        for k in range(7):
            all_forecasts[counter, 4 + k] = forecasts[k][i, j]
        counter += 1

        if i+j < 4:
            all_ferrors[e_counter, 0] = i + 1
            all_ferrors[e_counter, 1] = 2014 + i + j
            all_ferrors[e_counter, 2] = data_df.iloc[-4 + (i+j), 0] -
             imf_df.iloc[i + j, i]
            for k in range(7):
                all_ferrors[e_counter, 3 + k] = data_df.iloc[-4 + (i+j), 0] -
                 forecasts[k][i, j]
            best_model[e_counter] = fmodels[np.argmin(np.abs(
             all_ferrors[e_counter, 2:]))]
            e_counter += 1

all_forecasts_df = pd.DataFrame(all_forecasts, columns =
 ['$h$', 'forecast for', 'act. value','IMF estimate', 'AR', 'PCA EN', 'PCA BMA0',
         'PCA BMA1', 'SPCA EN', 'SPCA BMA0', 'SPCA BMA1'])
all_forecasts_df.replace(0, np.nan, inplace=True)

all_forecasts_df.to_csv('forecast_comp.csv')

all_ferrors_df = pd.DataFrame(all_ferrors,
        columns=['$h$', 'forecast for', 'IMF estimate', 'AR', 'PCA EN',
              'PCA BMA0', 'PCA BMA1', 'SPCA EN', 'SPCA BMA0', 'SPCA BMA1'])
all_ferrors_df['best model'] = best_model
all_ferrors_df.to_csv('ferrors_comp.csv')
```

```
def corr_heatmap():
    # create teh correlation hatmap
    import pandas as pd
    import seaborn as sns
    import matplotlib.pyplot as plt
    import matplotlib
    matplotlib.rc('font', size=5.5)
    matplotlib.rc('lines', linewidth=1)
    matplotlib.rcParams['figure.figsize'] = (10, 7)

    data_df1 = pd.read_excel('research_data.xlsx', index_col = 'year', parse_dates=[0])
    data = data_df1.to_numpy()
    data = smooth_data(data)
    data_df = pd.DataFrame(data, index = data_df1.index, columns = data_df1.columns)
    corr = data_df.corr()
    print(corr['USA'])
    ax = sns.heatmap(corr, vmin = -1, vmax = 1, center = 0, cmap =
    sns.diverging_palette(10, 133,n = 200), square = True)

    ax.set_xticklabels(ax.get_xticklabels(), rotation = 45, horizontalalignment =
     'right')
    plt.show()


def corr_geomap():
    # create the African correlation maps
    import geopandas as gpd
    import pandas as pd
    shapefile = '/Users/dominik/Downloads/ne_110m_admin_0_countries/
     ne_110m_admin_0_countries.shp'
```

```
gdf = gpd.read_file(shapefile)[['ADMIN', 'ADM0_A3', 'geometry']]
gdf.columns = ['country', 'country_code', 'geometry']
#print(gdf.sort_values('country').country_code)

data_df1 = pd.read_excel('research_data.xlsx', index_col='year', parse_dates=[0])
data = data_df1.to_numpy()
data = smooth_data(data)
data_df = pd.DataFrame(data, index=data_df1.index, columns=data_df1.columns)
data_df = data_df.drop(['World', 'USA', 'Oil', 'Gold', 'Copper', 'Rubber', 'Coffee'],
  axis=1)
corr = data_df.corr()
corr = corr.iloc[:, [0]]
names = pd.read_excel('country_labels.xlsx', index_col=0)
corr['code'] = names.Code
corr.columns = ['Correlation', 'Code']
import numpy as np
to_add = pd.DataFrame({'Correlation': ['No data', 'No data', 'No data', 'No data',
 'No data', 'No data'],'Code': ['ERI', 'SOM', 'SWZ', 'SOL', 'SAH', 'SDS']},
                      index=['Eritrera','Somalia','Swaziland', 'Somalialand',
                       'Western Sahara', 'South Sudan'])
corr = corr.append(to_add)

merged = gdf.merge(corr, left_on='country_code', right_on='Code', how = 'inner')



import json
merged_json = json.loads(merged.to_json())
json_data = json.dumps(merged_json)

from bokeh.io import output_notebook, show, output_file
from bokeh.plotting import figure
```

```
    from bokeh.models import GeoJSONDataSource, LinearColorMapper, ColorBar
    #from bokeh.palettes import brewer
    #from bokeh import palettes

    geosource = GeoJSONDataSource(geojson=json_data)

    palette = ['#005a32', '#238443', '#41ab5d', '#78c679', '#addd8e', '#d9f0a3',
     '#fdd49e', '#fdbb84','#fc8d59','#ef6548','#d7301f', '#990000']
    palette = palette[::-1]
    color_mapper = LinearColorMapper(palette=palette, low=-1, high=1,
      nan_color = '#d9d9d9')
    color_bar = ColorBar(color_mapper=color_mapper, label_standoff=8,
     width=500, height=20,border_line_color=None, location=(0, 0),
     orientation='horizontal')
    p = figure(plot_height=600, plot_width=950, toolbar_location=None)
    p.xgrid.grid_line_color = None
    p.ygrid.grid_line_color = None
    p.patches('xs', 'ys', source=geosource, fill_color={'field': 'Correlation',
     'transform': color_mapper},line_color='black', line_width=0.25, fill_alpha=1)
    p.add_layout(color_bar, 'below')
    show(p)


def plot_y():
    # plot the GDP growth
    import pandas as pd
    import matplotlib.pyplot as plt
    import matplotlib
    matplotlib.rc('font', size=8)
    matplotlib.rc('lines', linewidth=1.5)
    matplotlib.rcParams['figure.figsize'] = (9, 6)
```

```
    data_df1 = pd.read_excel('research_data.xlsx', index_col='year', parse_dates=[0])
    data = data_df1.to_numpy()
    data = smooth_data(data)
    data_df = pd.DataFrame(data, index=data_df1.index, columns=data_df1.columns)
    data_df.iloc[:, 0].plot()
    plt.ylabel('annual real percentage growth')
    plt.show()


def spca_graph():
    # create the SPCA graphs
    import pandas as pd
    import matplotlib.pyplot as plt
    import pandas as pd
    import matplotlib.pyplot as plt
    import matplotlib
    import numpy as np
    matplotlib.rc('font', size=7)
    matplotlib.rc('lines', linewidth=10)
    matplotlib.rcParams['figure.figsize'] = (10, 7)

    data = pd.read_excel('research_data.xlsx', index_col='year', parse_dates=[0])
    lambdas = pd.read_csv('Lambdas.csv', index_col = [0])
    lambdas.index = data.columns[1:]
    lambdas = lambdas.iloc[:, [-1, -2, -3]]
    lambdas.columns = ['First SPC', 'Second SPC', 'Third SPC']

    for i in range(3):
        lambdas.iloc[:, i] /= np.sum(lambdas.iloc[:, i])

    ax = lambdas.loc[np.sum(lambdas, axis = 1) > 0, :].transpose().
```

```
     plot(kind = 'bar', stacked = True)
    #plt.legend(loc='above')#, bbox_to_anchor=(1.07, 0.5))
    ax.set_xticklabels(ax.get_xticklabels(), rotation=45, horizontalalignment='right')
    ax.legend(loc='upper center', bbox_to_anchor=(0.5, 1.12), ncol=8)

    plt.show()



def pca_heatmap():
    # create the PCA heatmap
    import pandas as pd
    import seaborn as sns
    import matplotlib.pyplot as plt
    import matplotlib
    import numpy as np
    matplotlib.rc('font', size=5.5)
    matplotlib.rc('lines', linewidth=1)
    matplotlib.rcParams['figure.figsize'] = (15, 6)

    data = pd.read_excel('research_data.xlsx', index_col='year', parse_dates=[0])
    eigvec = pd.read_csv('Eigvec.csv', index_col=[0])
    eigvec.index = data.columns[1:]
    eigvec = eigvec.iloc[:, -np.arange(1, 28)]
    colnames = []
    for i in range(27):
        if (i > 9) & (i < 14):
            pcname = str(i + 1) + 'th PC'
        elif i % 10 == 0:
            pcname = str(i + 1) + 'st PC'
        elif i % 10 == 1:
            pcname = str(i + 1) + 'nd PC'
```

```
        elif i % 10 == 2:
            pcname = str(i + 1) + 'rd PC'
        else:
            pcname = str(i + 1) + 'th PC'
        colnames.append(pcname)
    eigvec.columns = [colnames]

    for i in range(27):
        eigvec.iloc[:, i] /= np.sum(np.abs(eigvec.iloc[:, i]))
    ax = sns.heatmap(eigvec.transpose(), vmin = -0.1, vmax = 0.1, center = 0,
                     cmap = sns.diverging_palette(10, 133, n = 200), square = True)

    ax.set_xticklabels(ax.get_xticklabels(), rotation = 45,
     horizontalalignment = 'right')
    plt.ylabel('ith PC')
    plt.show()


def pca_eigval():
    # show the PCA eigenvalues
    import pandas as pd
    import seaborn as sns
    import matplotlib.pyplot as plt
    import matplotlib
    import numpy as np
    matplotlib.rc('font', size=5.5)
    matplotlib.rc('lines', linewidth=1)
    matplotlib.rcParams['figure.figsize'] = (10, 7)

    data = pd.read_excel('research_data.xlsx', index_col='year', parse_dates=[0])
    eigval_help = pd.read_csv('Eigval.csv', index_col=[0])
```

```
eigval = np.zeros(28)
for i in range(27):
    eigval[i] = eigval_help.iloc[-i].copy()
#eigval[:-1] = np.copy(eigval_help.iloc[-np.arange(1, 28)])
eigval[-1] = np.sum(eigval_help[:-28])
sum_keep = np.sum(eigval)
for i in range(28):
    eigval[i] /= sum_keep
colnames = []
for i in range(27):
    if (i > 9) & (i < 14):
        pcname = str(i + 1) + 'th PC'
    elif i % 10 == 0:
        pcname = str(i + 1) + 'st PC'
    elif i % 10 == 1:
        pcname = str(i + 1) + 'nd PC'
    elif i % 10 == 2:
        pcname = str(i + 1) + 'rd PC'
    else:
        pcname = str(i + 1) + 'th PC'
    colnames.append(pcname)
colnames.append('rest')
eigval = pd.DataFrame({'fraction of explained correlation': eigval},
 index = colnames)

ax = eigval.plot(kind = 'bar', legend=False)
ax.set_xticklabels(ax.get_xticklabels(), rotation=45, horizontalalignment='right')
plt.show()
```

```
def component_comparison():
    # tables for PCA and SPCA comparison
    import pandas as pd
    import matplotlib.pyplot as plt
    import matplotlib
    import numpy as np
    matplotlib.rc('font', size=5.5)
    matplotlib.rc('lines', linewidth=1)
    matplotlib.rcParams['figure.figsize'] = (10, 7)
    data = pd.read_excel('research_data.xlsx', index_col='year', parse_dates=[0])
    eigvec = pd.read_csv('Eigval.csv', index_col=[0])
    lambdas = pd.read_csv('Lambdas.csv', index_col=[0])
    for i in range(eigvec.shape[1]):
        eigvec.iloc[:, i] /= np.sum(eigvec.iloc[:, i])
    for i in range(lambdas.shape[1]):
        lambdas.iloc[i, :] /= np.sum(lambdas.iloc[:, i])
    pc_fac = data.iloc[:, 1:].to_numpy() @ eigvec.to_numpy()[:, -1]
    spc_fac = data.iloc[:, 1:].to_numpy() @ lambdas.to_numpy()[:, -1]

    component_comp = data.iloc[:, :1]
    component_comp.loc[:, 'First PC'] = pc_fac
    component_comp.loc[:, 'First SPC'] = spc_fac
    component_comp.to_csv('princomps.csv')

    corr = component_comp.corr()
    corr.to_csv('pca_spca_corr')

    component_comp.plot()
    plt.ylabel('percentage annual growth')
    plt.show()
```

```
def plot_autocorr():
    # create the autocorrelation plots
    import pandas as pd
    import matplotlib.pyplot as plt
    import matplotlib
    import numpy as np
    from statsmodels.graphics.tsaplots import plot_acf
    matplotlib.rc('font', size=5.5)
    matplotlib.rc('lines', linewidth=1)
    matplotlib.rcParams['figure.figsize'] = (10, 7)

    res_df = pd.read_csv('ferrors1.csv', index_col=[0])

    #matplotlib.pyplot.acorr(res_df.loc[:, 'PCA BMA1'], normed = False)
    #plt.plot(10000 / (res_df.shape[0]**0.5), 'r')
    #autocorrelation_plot(res_df.loc[:, 'PCA BMA1'])
    plot_acf(res_df.loc[:, 'PCA BMA1'].to_numpy(), lags = 20)
    plt.xlabel('lag')
    plt.ylabel('autocorrelation')
    plt.show()



def plot_forecasts():
    # plot the forecasts
    import pandas as pd
    import matplotlib
    import numpy as np
    import matplotlib.pyplot as plt
    matplotlib.rc('font', size=7)
```

```
matplotlib.rc('lines', linewidth=2)
matplotlib.rcParams['figure.figsize'] = (10, 7)

forecasts_df = pd.read_csv('forecast_comp.csv', index_col = 2)
forecasts_df.index = pd.to_datetime(forecasts_df.index.astype(int), format='%Y')

import pickle
with open('cv_pca_staticforec.pickle', 'rb') as data:
    stats_coll = pickle.load(data)
errors_ar = np.zeros(25)
errors_pcabma1 = np.zeros(25)
errors_spcaen = np.zeros(25)
for i in range(5):
    errors_ar[5*i: 5*(i+1)] = stats_coll[0][i, -5:]
    errors_pcabma1[5*i: 5*(i+1)] = stats_coll[3][i, -5:]
    errors_spcaen[5*i: 5*(i+1)] = stats_coll[4][i, -5:]
forecasts_df['low_ar'] = forecasts_df['AR'] - errors_ar
forecasts_df['high_ar'] = forecasts_df['AR'] + errors_ar
forecasts_df['low_bma'] = forecasts_df['PCA BMA1'] - errors_pcabma1
forecasts_df['high_bma'] = forecasts_df['PCA BMA1'] + errors_pcabma1
forecasts_df['low_en'] = forecasts_df['SPCA EN'] - errors_spcaen
forecasts_df['high_en'] = forecasts_df['SPCA EN'] + errors_spcaen
print(forecasts_df.columns)

for i in range(5):
    ax = forecasts_df.iloc[5*i :5*(i+1), [2, 3, 4, 7, 8, 11, 12, 13, 14, 15, 16]].
     plot(color =['xkcd:black', 'xkcd:steel blue', 'xkcd:scarlet',
     'xkcd:sky blue', 'xkcd:emerald green','xkcd:tomato red', 'xkcd:tomato red',
     'xkcd:light blue', 'xkcd:light blue', 'xkcd:medium green','xkcd:medium green'],
     style = ['.', '.', '-', '-', '-', '-.', '-.', '-.', '-.', '-.', '-.'])
    fco = forecasts_df.columns
```

```
ax.legend([fco[2], fco[3], fco[4], fco[7], fco[8], '_nolegend_',
 '_nolegend_', '_nolegend_', '_nolegend_','_nolegend_', '_nolegend_'])
import datetime
datemin = forecasts_df.iloc[5*i :5*(i+1), :].index[0] - pd.Timedelta(days = 365)
datemax= forecasts_df.iloc[5*i :5*(i+1), :].index[-1] + pd.Timedelta(days = 366)
ax.set_xlim(datemin, datemax)
#ax.set_xticks = ['2014', '2015', '2016', '2017', '2018']
plt.xlabel('Year')
plt.ylabel('DRC Percentage annual real GDP growth')
plt.show()
```