Non-Linear Dimensionality Reduction for Macroeconomic Forecasting

Alessandro Ciancetta, Nicole Poynarova, Alessandro Tenderini

June 4, 2023

Abstract

We investigate whether non-linear techniques for dimensionality reduction can improve the performances of the linear state-of-the-art models for macroeconomic forecasting. Besides testing existing techniques, we present a new non-linear model, the Time-Varying Principal Component Regression (TV-PCR). Using the FRED-MD dataset, which includes a variety of macroeconomic time series from the U.S. economy, we compare the performance of linear and non-linear dimensionality reduction techniques in forecasting. Our results suggest that non-linear models can improve the prediction of certain variables, highlighting the potential of non-linear models to capture complex patterns. Notably, non-linear models outperform linear ones at nowcasting and forecasting the U.S. unemployment rate. The study concludes that the best model, whether linear or nonlinear, largely depends on the specific variable and forecasting horizon.

Keywords — Non-Linear Dimensionality Reduction, Principal Component Regression, Factor Models, Forecasting, Time Series Analysis.

1. Introduction

Macroeconomic forecasting encounters a significant concern when it comes to the inclusion of vast sets of information within econometric models. The issues that one has to face when using large-dimensional data are various. A problem known since at least the beginning of the literature on VAR models (Sims 1980), is the so-called "curse of dimensionality" problem. This essentially involves the lack of statistical identification of the parameters in the model due to the limited size of the available samples and also due to the fast growth of the number of parameters as more variables are included in the model. The most natural approach to solve the curse of dimensionality has resorted to dimensionality reduction techniques to obtain a parsimonious representation of the data to include in the model. Among these techniques, we can distinguish between two-step and one-step procedures. Two-step procedures consists of first extracting a latent representation of the data and then use them in a different predictive model. Examples of these methods are the classical Principal Component Regression (PCR henceforth) first proposed in Hotelling (1957) or the factor-augmented VAR of Bernanke, Boivin, and Eliasz (2005), and are still in fashion as they are difficult benchmarks to beat in forecasting. One-step procedures are developed in fully-endogenous models where it is possible to adapt the estimation of the latent variables according to their predictive ability within the model. The prominent example of this type of model is the Dynamic Factor

Model (DFM henceforth; Barigozzi and Luciani 2019; Forni et al. 2000; Sargent and Sims 1977). The Kalman-filter based estimation of the latent components in the DFM makes it a very suitable model to deal with several issues arising in the practice of macroeconomic forecasting, like missing values and real-time inclusion of new observations.

However, a major issue of these models is the ubiquitous assumption of a linear relationship between the observable and latent variables. Allowing non-linear mappings between the observable and latent spaces provides the model with more flexibility and can potentially lead to a better representation of the signal in the data and improve forecasting performances. The first attempts to estimate non-linear representations date back to the work on principal curves by Hastie and Stuetzle (1989), that started a dense research program on the theory of principal manifolds (see Gorban et al. 2008 for an extensive review). Estimation of non-linear latent variables can be achieved both with kernel methods or by directly estimating a non-linear function of the latent variables using non-linear approximators. The latter approach is based on the well-known class of neural networks known as autoencoders (Kramer 1991), developed in fields other than macroeconometrics. Indeed, the focus of the macroeconometric literature has mainly been on linear models, with exceptions mainly related to dealing with heteroskedasticity and regime switches (Kock and Teräsvirta 2011). Techniques for non-linear forecasting are nevertheless growing in popularity, and many recent works show the potential of this methods (see for instance Hauzenberger, Huber, and Klieber 2023).

In this work, we will compare the forecasting performances of models based on linear and non-linear dimensionality reduction techniques on a panel of macroeconomic time series of the U.S. economy. The main aim of the work is to study whether allowing non-linear mapping between the observed and latent variables in the models can improve the performances of state-of-the-art methods. Our contribution is twofold. First, we propose a new non-linear method (the Time-Varying Principal Component Regression, TV-PCR) that we believe is more suitable for forecasting, as it is takes into account the sequential nature of time-series data. Second, we conduct an extensive and thorough assessment of several models that are not usually adopted in the macroeconometric literature, comparing their performances at both nowcasting and forecasting. We find that the non-linear models that we consider improve the results over linear models for specific variables (notably, the unemployment rate). However, we find that the forecasts obtained by PCR and DFMs are a hard benchmark to beat and overall the increased flexibility of non-linear models is not always helpful in improving the forecasting ability of the model.

The work is organized as follows. Section 2 reviews in detail the linear and non-linear dimensionality reduction methods employed in this work and introduces the TV-PCR. Section 3 describes the data, the preprocessing steps and the methodology adopted for forecast evaluation. Section 4 presents the findings and the analysis of the results. We then conclude with some final remarks about the results and future research.

2. Non-linear dimensionality reduction and forecasting

In this section we review several non-linear dimensionality techniques that we will use in the forecast evaluation exercise, and subsequently introduce the TV-PCR model. The notation that we will use is as follows. Let $X \in \mathbb{R}^{T \times K}$ be a panel of time series with K variables and T observa-

tions. We denote an observation as x_t, so that $X = (x_1, x_2, ..., x_T)'$. The dimensionality reduction occurring via a function $f : \mathbb{R}^K \to \mathbb{R}^q$, can be either linear or non-linear. When $f(X)$ is applied, we denote by $Z \in \mathbb{R}^{T \times q}$ the low-dimensional representation $Z = f(X) = (z_1, z_2, ..., z_T)'$.

2.1. Principal Component Based Methods

Principal Component Analysis (PCA) is a natural starting point for generalizations to non-linear dimensionality reduction. Non-linearities can be introduced in the method by manipulating the original data in two ways. The first is to consider a function g mapping the regressors onto a covariates matrix $W = g(X)$. The second method is to apply a kernel function onto the sample covariance matrix, $h : \kappa = h(W'W)$. The principal components are obtained by performing Singular Value Decomposition (SVD) on κ. Then the principal components Z and the original variables X are:

$$Z = f(X) = g(X)\lambda(\kappa) = W\lambda(\kappa) \tag{1}$$

Following this, h and g could be a variety of different linear or non linear functions, so we will explore different methods and models that can be built from this foundation.

Linear PCR Standard PCA is a particular case of the formulation above obtained by simply considering identity mappings: $W = X$ and $\kappa = X'X$. After having extracted the components in Z, the regression model is

$$y_t = \beta_0 + \beta_1 y_{t-1} + Z_{t-1}\gamma_1 + \cdots + Z_{t-p}\gamma_p + \epsilon_t, \tag{2}$$

where p denotes the number of lags of the principal components included in the model. Even though linear PC is quite easy to implement, the assumption of linearity can be restrictive and prevents the method from capturing non-linear relationships between the variables (Malthouse 1998).

.

Quadratic PCR One extension into non-linearity, is the introduction of Quadratic Principal Component Analysis, introduced by Bai and Ng 2008. The idea of this method is that a quadratic connection between the covariate matrix and the covariance matrix is applied. We obtain the transformed matrices $W = (X, X^2)$ and $\kappa = W'W$, where X^2 denotes the Hadamard Product of the matrix ($X^2 = X \odot X$), representing the entry wise multiplication of the matrices (Million 2007). Bai and Ng (2008) showed that by using this extension of PCA, the principal components may have higher predictive power.

Kernel PCR We now move onto the Kernel PCR. This again acts in accordance of the previous two PCR methods. First introduced by Schölkopf, Smola, and Müller (1998), it essentially involves calculating eigenvalues and eigenvectors of the the kernel matrix rather than on the covariance matrix. Kernels provide information on the neighbours of observations and quantifies their similarities based on non-linear relationships. In this case the Euclidean distance is used to find the distance between points. In our application, we consider two kernels, namely the Gaussian kernel, where the

$(i,j)^{th}$ entry of the $K \times K$ matrix is defined as:

$$\kappa_{ij} = \exp\left(\frac{-\|x_{\bullet i} - x_{\bullet j}\|^2}{2c_1^2}\right) \tag{3}$$

or a polynomial kernel:

$$\kappa_{ij} = \exp\left(\frac{x_{\bullet i} x'_{\bullet i}}{2c_0^2} + 1\right) \tag{4}$$

where c_0 and c_1 are scaling factors.

Time-varying PCR The dimensionality reduction methods that we have so far examined, have many different uses, but are not specifically designed for time series data. These methods would in theory provide somewhat inaccurate forecasting predictions, as they may miss the time-dependent features of the data. In order to overcome these limitations, we introduce a different Principal Component based method, called "Time-Varying PCR".

The main idea of this technique is to account for time varying dependence structures. Ideally, this is done by allowing the partial effect of each predictor on the target to change over time. However, it is not possible to estimate a different effect for each and every point in time as it leads to complete over-fitting. Our idea is to retrieve this time variation by creating an augmented data matrix and extracting its principal components. The method proceeds as follows. We first build the transformed matrix W (with dimensions $T \times (TK)$) which is produced by concatenating T matrices, each denoted by X_t, where the first $t-1$ rows are replaced by the zero vectors:

$$W = (X_1, X_2, \ldots, X_{T-1}, X_T) = \begin{bmatrix} x_1 & 0 & 0 & 0 & \cdots \\ x_2 & x_2 & 0 & 0 & \cdots \\ x_3 & x_3 & x_3 & 0 & \cdots \\ \vdots & \vdots & \vdots & \vdots & \vdots \end{bmatrix}$$

Once the matrices are built, then the principal components are extracted from the matrix W, we obtain the matrix Z and we run a standard PCR.

Penalized TVPCR As the transformed matrix W presents a large amount of columns, we introduce the penalized version of the TVPCR. First, we establish the LASSO linear regression which uses L1 regularization. Such a penalty term is the sum of the absolute values of the coefficients, multiplied by a regularization parameter. The regularization parameter λ controls the strength of the penalty term and we tuned it via time series cross validation. The objective function for lasso regression can be written as:

$$\min_{\beta} \sum_{i=1}^{n} (y_i - \mathbf{x}_i^T \beta)^2 + \lambda \sum_{j=1}^{p} |\beta_j| \tag{5}$$

Then, we estimate the Ridge regression which uses L2 regularization with the following objective function:

$$\min_{\beta} \sum_{i=1}^{n} (y_i - \mathbf{x}_i^T \beta)^2 + \lambda \sum_{j=1}^{p} \beta_j^2 \tag{6}$$

Last, we also consider the Adaptive LASSO which is a variant of lasso regression that uses a modified version of the L1 penalty. Instead of penalizing the absolute values of the coefficients, the weighted sum of the absolute values of the coefficients is used:

$$\min_{\beta} \sum_{i=1}^{n} (y_i - \mathbf{x}_i^T \beta)^2 + \lambda \sum_{j=1}^{p} \frac{|\beta_j|}{\hat{w_j}} \tag{7}$$

The weights are actually the inverse of estimates from another penalized model. In this work we use the LASSO estimates as initial estimates. During the forecast evaluation of both penalized models, we re-tune the parameter λ at each step. In fact, every time we expand the window, we create a new testing and training set and perform cross validation on a rolling basis over a sequence of λ values. Then, we select the λ associated with the lower RMSE and use it to train the model and predict the next observation. In the case of the adaptive LASSO, we need to tune two penalization parameters at each step through two separate rolling-window cross-validations.

2.2. Other manifold-learning techniques

Diffusion Map Diffusion Maps is a dimensionality reduction technique developed by Coifman and Lafon (2006), that utilises the principles of diffusion and Markov chains while preserving the data's fundamental geometry and nonlinear relationships.
The first step is to produce a distance matrix between data points $x_{\bullet i}$ and $x_{\bullet j}$ in high dimensional Euclidean space. This distance matrix is then converted into a N x N *affinity matrix*. If the data is viewed in high dimensions, there may be causes where a "walk-over" is done to get to get from one point to another. A probabilistic way to capture this distance is a random-walk, with its formula being a Gaussian kernel:

$$w(x_{\bullet i}, x_{\bullet j}) = \exp\left(-\frac{||x_{\bullet i} - x_{\bullet j}||^2}{c_2}\right) \tag{8}$$

where c_2 is a tuning parameter, chosen by knowing the structure and density of the data, mentioned by Porte, Herbst, and Hereman Delaporte et al. 2008. The next step is to normalize this affinity matrix to ensure that each row sums up to 1. This normalization step is crucial for the subsequent diffusion process. The formula to calculate this is:

$$p_{i\to j} = \text{prob}(x_{\bullet i} \to x_{\bullet j}) = \frac{w(x_{\bullet i}, x_{\bullet j})}{\sum_j w(x_{\bullet i}, x_{\bullet j})} = \frac{\exp(-\frac{||x_{\bullet i} - x_{\bullet j}||^2}{c_2})}{\sum_j \exp(-\frac{||x_{\bullet i} - x_{\bullet j}||^2}{c_2})} \tag{9}$$

From these probabilities, we are able to construct a transition matrix of a Markov chain. Specifically $p_{i\to j}$ is the transition probability of moving from the $i^{th}$to the j^{th} data point, classified by P. P^n is the probability that point $x_{\bullet i}$ moves to point $x_{\bullet j}$ in n = 1,2,.... steps. The next step is to find the eigenvalue decomposition of P^n:

$$p^n_{i\to j} = \sum_{s\geq 0} \lambda_s^n \psi_s(x_{\bullet i}) \phi_s(x_{\bullet j}) \tag{10}$$

where λ_s is the eigenvalues of $p_{i\to j}$, ψ_s is the bi-orthogonal right eigenvectors, and ϕ_s is the bi-orthogonal left eigenvectors (in which these are obtained by using spectral decomposition Coifman

et al. 2005). Then, the *diffusion distance* between $x_{\bullet i}$ and $x_{\bullet j}$ is calculated, to meaningfully assess the similarity or dissimilarity between points, which is critical for capturing the underlying structure and relationships in high-dimension. It is given by this formula shown in the (Coifman and Lafon 2006) paper using the eigenvectors:

$$\xi_n^2(x_{\bullet i}, x_{\bullet j}) = \sum_{s=1}^{\infty} \lambda_s^{2n} (\psi_s(x_{\bullet i}) - \psi_s(x_{\bullet j}))^2 \tag{11}$$

Then the family of diffusion maps is presented, where the eigenvectors are used as the "new coordinates" for the data.

$$\Xi_n(x_{\bullet i}) = \begin{pmatrix} \lambda_1^n \psi_1(x_{\bullet i}) \\ \lambda_2^n \psi_2(x_{\bullet i}) \\ \vdots \\ \lambda_q^n \psi_q(x_{\bullet i}) \end{pmatrix} \tag{12}$$

Due to the spectrum decay (where the magnitudes of the eigenvalues gradually decrease as we move from the first eigenvalue to the subsequent ones), only the first q eigenvectors and eigenvalues can be used. The Euclidean distance in the diffusion co-ordinates approximates the diffusion distance, as demonstrated by proposition 1 in the Coifman and Lafon paper (ibid.):

$$||\Xi_n(x_{\bullet i}) - \Xi_n(x_{\bullet j})|| = \xi_n(x_{\bullet i}, x_{\bullet j}) \tag{13}$$

To apply this non linear dimensionality reduction model to our data, the R packages "diffusionMap" and "destiny" will be used (Angerer et al. 2015; Richards and Cannoodt 2019).

Locally-Linear Embeddings LLEs presented by Saul and Roweis (2001) described an unsupervised learning algorithm that finds the low-dimensional, neighborhood-preserving embeddings of high-dimensional data. The assumption is that neighbouring data points in the high-dimensional space, should also be neighbours in the low dimensional embedding, allowing the data manifold's intrinsic structure to be captured by the LLE.

Finding the set of nearest neighbours of $x_{\bullet i}$ is the first step of the LLE algorithm. The KNN algorithm is implemented to find it's k nearest neighbours and models the points onto graph using the pairwise euclidean distances between the points $x_{\bullet i}$ and $x_{\bullet j}$. Then the reconstruction weights are computed to best approximate each data point using its neighbors. From this, a weight matrix Ω is defined, where w_{ij} is the $(i,j)^{th}$ element of Ω. Each of the weights are found by reconstructing every point by its neighbors, minimizing the reconstruction error, where C is the cost function. It is given by this equation:

$$C(\Omega) = \sum_i \left(x_{\bullet i} - \sum_j \omega_{ij} x_{\bullet j} \right)^2 \tag{14}$$

This is the squared difference between a data point $x_{\bullet i}$ and its linear reconstruction from its neighbors. For C to be minimised, it has two specific constraints (stated by Ghojogh et al. 2020). The first being that each data point is reconstructed only from its neighbours, resulting in $\omega_{ij} = 0$ if $x_{\bullet i}$ and $x_{\bullet j}$ are not neighbours. The second constraint is that when you sum each row in Ω it sums up

to 1 (ie: $\sum_{j=1}^{k} \omega_{ij}$=1).
The final step of the algorithm is to map each high dimensional data point, $x_{\bullet i}$, into its corresponding low dimensional vector $z_{\bullet i}$. To do this, d-dimensional coordinates of $z_{\bullet i}$ are chosen in a way that minimizes the cost function defined above, but with the difference that the weights Ω are fixed while optimizing the co-ordinates $z_{\bullet i}$. The new minimization cost equation now establishes the vectors to be in quadratic form :

$$\Phi(Z) = \sum_{i} \left| z_{\bullet i} - \sum_{j} (\Omega_{ij} z_{\bullet j}) \right|^2 \tag{15}$$

The "RDRToolbox" (Bartenhagen 2023) package provides us with the necessary tools in R to apply the LLE model to our time series data.

Isometric Feature Mapping Tenenbaum, Silva, and Langford (2000) first presented ISOMAP, which is a technique where the underlying global geometry is learnt from a dataset and "easy local metric information" is used to discover information in curved dimensions. The main characteristic of this model, is that the *geodesic distance* is preserved, developed from a weighted graph. Geodesic distances are more useful in non linear instances, as it is the shortest path distance along a curve or surface, minimising the distance between points on a manifold or curved surface at the local level. This is the extension of the multidimensional scaling (MDS), where the Euclidean distance is calculated instead.

There are several steps of the ISOMAP algorithm. The first step is using the KNN technique to find "k" nearest neighbours of data points on a manifold. Then once the number of neighbours is specified, a neighbourhood graph G is created from the data, linking neighbours with edges. The second step is finding the geodesic distance by using Dijkstra's algorithm. In short, this algorithm finds the shortest path of a weighted graph (i.e $d_G(i, j)$). Finally, the lower dimensions embeddings are found using Multidimensional scaling (MDS). Because the distances between points are known, MDS arranges each object in N-dimensional space to maintain the between-point distances.

For our data set, we have used again the "RDRToolbox" package used in the LLE model (Bartenhagen 2023).

t-Stochastic Neighbour Embeddings T-SNE is a non linear dimensionality reduction technique built upon the SNE by Hinton and Roweis (2002), whereby the neighbourhood identity is preserved optimally in low dimensional space. The generic SNE has some limitations including an issue that crowding of points occurs in the low-dimensional space meaning points which are not similar at all appear close together, hence Maaten and Hinton (2008) extended SNE and developed "t-SNE".
To fully explore t-SNE, firstly the stochastic neighbourhood embedding technique (SNE) must be introduced. Instead of finding Euclidean distances between the data points $x_{\bullet i}$ and $x_{\bullet j}$ in high dimensional data, conditional probabilities are calculated, which represent the similarities between these data points. A Gaussian distribution is modelled for each data point, with conditional probability $p_{i|j}$ that a data point $x_{\bullet j}$ belongs in the same class as $x_{\bullet i}$:

$$p_{i|j} = \frac{\exp(-|x_{\bullet i} - x_{\bullet j}|^2 / 2\sigma_j^2)}{\sum_{k \neq j} \exp(-|x_{\bullet k} - x_{\bullet j}|^2 / 2\sigma_j^2)} \tag{16}$$

where σ_j is the variance centred at point $x_{\bullet i}$. For the corresponding points in the low dimensional data, $z_{\bullet i}$, a similar conditional probability can be calculated $q_{j|i}$, with the difference that the variance is set to $\frac{1}{\sqrt{2}}$ giving:

$$q_{j|j} = \frac{\exp(-|z_{\bullet i} - z_{\bullet j}|^2)}{\sum_{k \neq j} \exp(-|z_{\bullet k} - z_{\bullet j}|^2)} \tag{17}$$

In order to make the optimisation problem easier, the probabilities are symmetrised, giving corresponding distributions P (the high dimensional representation) and Q (the low dimensional representation) of the data. Obtaining:

$$P = (p_{ij})_{i,j=1}^n, p_{ij} = \frac{p_{i|j} + p_{j|i}}{2n} \tag{18}$$

$$Q = (q_{ij})_{i,j=1}^n, q_{ij} = \frac{(1 + |z_{\bullet i} - z_{\bullet j}|^2)^{-1}}{\sum_{k \neq j}(1 + |z_{\bullet k} - z_{\bullet j}|^2)^{-1}} \tag{19}$$

Using a Student's t-distribution, the relation between each neighbour is modelled to handle the crowding problem and preserve the global structure. The heavy-tailed nature of the student's t-distribution is exploited by t-SNE, reducing the crowding effect and preventing the points from being overly compressed by increasing the likelihood that distant points in the high-dimensional space will be chosen as neighbours.
The main aim is to minimize the Kullback–Leibler divergence between P and Q. It is defined as:

$$KL(P|Q) = \sum_{i=1}^{n} \sum_{j=1}^{n} p_{ij} log \frac{p_{ij}}{q_{ij}} \tag{20}$$

To solve this minimisation problem, gradient descent is performed with respect to $z_{\bullet i}$:

$$\frac{\partial KL(P|Q)}{\partial z_{\bullet i}} = 4 \sum_{j=1}^{n} (p_{ij} - q_{ij})(z_{\bullet i} - z_{\bullet j})(1 + |z_{\bullet i} - z_{\bullet i}|^2)^{-1} \tag{21}$$

We use the R package "RTSNE" (Krijthe 2015) to estimate the t-SNE model.

2.3. Dynamic Factor Models

The Dynamic Factor Models (DFM), are a widely-adopted class of linear models for macroeconomic forecasting. The way that this model works, is by firstly mapping out the features into the latent space, then in order to get the predictions of the next state, a dynamic model is fitted. The original state space is given by:

$$\begin{aligned} x_t &= \Lambda f_t + \xi_t \\ f_t &= A_1 f_{t-1} + ... + A_p f_{t-p} + \eta_t \end{aligned} \tag{22}$$

where x_t is the observed variables in the dataset at a specific time t, in a $n \times 1$ vector, f_t is the latent factors in matrix form with size $q \times 1$, Λ is the $K \times q$ matrix of factor loadings and A_i, for $i = 1, ..., p$ are the $q \times q$ autoregressive matrices which describe the latent factor loadings. The latent factors and the parameters of the DFM are estimated using the EM algorithm (Barigozzi

and Luciani 2019). The factors are initialized with to the first q principal components, and the parameters $\Lambda, A_1, \ldots, A_p$ are estimated via least-squares on the initialized factors. Then, given the estimated parameters, the Kalman filter is applied to get an estimate of the latent factors. Given the new estimated factors, the parameters are re-estimated via OLS. The procedure is iterated until convergence of the estimated parameters.

Deep Dynamic Factor Model Andreini, Izzo, and Ricco (2020) consider a generalization of the DFM to relax the assumption of linearity. The Deep Dynamic Factor Models (D2FM), allows for the possibility of non-linearities in the encodings between the observed and the latent variables and also between subsequent realizations of the latent variables. The general formulation of the model is:

$$x_t = D(f_t) + \xi_t \tag{23}$$

$$f_t = R(f_{t-1}, \ldots, f_{t-p}) + \eta_t, \tag{24}$$

where both the static mapping $D : \mathbb{R}^q \to \mathbb{R}^K$ and the dynamic mapping $R : \mathbb{R}^q \to \mathbb{R}^q$ are in general non-linear. The model can be estimated using neural networks as universal approximators of functions (Cybenko 1989; Hornik, Stinchcombe, and White 1989). In particular, we can estimate Eq. 23 using autoencoders, by parametrizing the expression through an encoding function $E : \mathbb{R}^K \to \mathbb{R}^q$ and a decoding function $E : \mathbb{R}^q \to \mathbb{R}^K$ as $x_t = D(E(x_t)) + \xi_t$. Also, Eq. 24 can be estimated using recurrent architectures. For our application, we design the network using Gated Recurrent Units (Cho et al. 2014; Chung et al. 2014).

3. Data and methodology

We will compare the forecasting performances of the models using the FRED-MD data (McCracken and Ng 2016).[1] FRED-MD is a constantly-updated database of 127 monthly U.S. macroeconomic indicators. We use the FRED-MD vintage spanning the period from January 1960 to January 2023. The preprocessing procedure that we use follows three steps. First, we transform each variable to ensure stationarity. Second, we remove outliers from each series by considering them as missing values. Finally, we impute the missing values. In each step, we follow the indications of McCracken and Ng (ibid.). We will focus in particular on the ten policy-relevant variables identified by Kim and Swanson (2011) and described in Table 1 with the details about the type of transformation adopted in the preprocessing. The imputation of missing values is a necessary step in our exercise as not all the models that we will consider have a natural and comparable way of dealing with missing values. However, it is worth noticing that methods like the Kalman-filter-based estimation of state-space models can be particularly suitable precisely because they allow to update the estimated factors even when some variables are missing. This feature is of particular relevance in real-time applications, where the Kalman filter allows to easily deal with mixed-frequency data and new releases of some predictors with no delay. In order to make sure that we compare the non-linear models against the strongest linear benchmark, we run the forecast evaluation for the

[1] The dataset is freely available at `http://research.stlouisfed.org/econ/mccracken/fred-databases/`.

DFM by fitting the model both on the inputed data and on the data with missing values.

Since no outlier adjustments have been made to the raw data, we check for outliers in the stationary series and remove them prior to constructing the factors, thus treating them as missing data. We define an outlier as an observation that deviates from the sample median by more than ten interquartile ranges. This step is of particular relevance to deal with the Covid-19 period. In particular, for many variables, the first months of 2020 registered values completely out of the ordinary. Any comparison between models would be misleading if these observations were included, as the results would be dominated by a big-magnitude error that all the models would make corresponding to those observations. Differences in the errors would essentially be random and would not reflect any actual ability of predicting those few observations coming from a completely different distribution and with no information existing about them in past data. For this reason, we treat these values as any other outlier and impute it as a missing value. We then impute missing values using the EM algorithm presented in Stock and Watson 2002. Data is demeaned and standardized, and all missing values are initialized to zero, that is to their unconditional mean. We extract the first $r = 8$ principal components of the matrix X, obtaining a matrix of factors $\widehat{F} \in \mathbb{R}^{T \times r}$ and a matrix of loadings $\widehat{\theta} \in \mathbb{R}^{K \times r}$. The missing values in series i at time t are then updated to the value predicted by the estimated factors, $\widehat{X}_{it} = \widehat{\theta}_i' \widehat{F}_t$. At this step, each variable is re-multiplied by the standard deviation, and the mean is re-added, so that the values that were missing can now be treated as observations for series i at time t. This procedure is iterated until the estimates of the factors converge to a fixed value. The imputed dataset we use in the analysis is therefore a balanced monthly panel with no outliers.

In the forecasting-evaluation exercise, we will use a rolling-window approach consisting of fixing the dimension of a training set of consecutive observations, estimating a model over that set of observations, getting an out-of-sample forecast and then repeat the procedure by discarding the first observation in the window and appending the next available observation to it. In particular, we focus on the forecasting horizons 1,3,4,6. This choice depends on the fact that the variables in the FRED-MD dataset (as any var are available only after

To make sure that we do not introduce any future information in the set of training data points, at each step of the evaluation we first pick the train set from the raw dataset, and only after we apply the preprocessing procedure (transformation, outlier detection, imputation) on the training set. Appendix A. contains an example of the preprocessing procedure on the whole sample period for the ten policy-relevant variables.

Computational considerations We implement the models and the forecast evaluation procedure in R, with the only exception of the D2FM, which is implemented in Python using the PyTorch library.[2] Some considerations regarding the running time of the simulations are worth noting. At each step, we estimate each model on a window of 672 observations and we predict a value h-steps ahead. This means that each model is re-estimated $99 - h$ times, for $h \in \{1, 3, 4, 6\}$. Overall we consider 17 different models. We consider two kernels for Kernel-PCR and we run the DFM twice (with and without missing values), so that we end up with a total of 19 methods to run. Additionally, to explore the potential of the TV-PCR, we run cross-validation to select the optimal hyperparameters for LASSO TV-PCR and Adaptive LASSO TV-PCR specification. Since we

[2]The replication files can be found at https://github.com/aciancetta/replication-CPT23.

Variable	Description	Transformation
UNRATE	Unemployment rate	$x_t - x_{t-1}$
W875RX1	Real personal income ex transfer receipts	$\log x_t - \log x_{t-1}$
GS10	10-Year Treasury Rate	$x_t - x_{t-1}$
CPIAUCSL	Consumer price index (all items)	$\Delta \log x_t - \Delta \log x_{t-1}$
WPSFD49207	Producer Price Index (Finished goods)	$\Delta \log x_t - \Delta \log x_{t-1}$
PAYEMS	All Employees (total monfarm)	$\log x_t - \log x_{t-1}$
HOUST	Housing Starts (Total New Privately Owned)	$\log x_t$
INDPRO	Industrial production	$\log x_t - \log x_{t-1}$
M2SL	M2 Money Stock	$\Delta \log x_t - \Delta \log x_{t-1}$
S.P500	S&P500 Common Stock Price Index: Composite	$\log x_t - \log x_{t-1}$

Table 1: *Description of the policy-relevant variable in Kim and Swanson (2011) and transformation used in the preprocessing. The forecast evaluation focuses on the performances of the models in predicting these variables.*

want to make sure that the results are not driven by the optimally-tuned penalization but rather by the non-linear dimension reduction technique itself, we also cross-validate the baseline LASSO regression and LASSO-PCR. We also run cross-validation to select the optimal number of principal components in PCR. Cross-validation is carried out by further subsetting the window at each step, considering the last 10% of the window (67 observations) as a validation set. We use a grid of 13 values, so that for each step in the evaluation the model is re-estimated $67 \times 13 = 871$ times and the hyperparameter associated to the lowest RMSE on the validation set is selected to fit the model on the whole window and produce the forecast h-step ahead of the last observation in the window. As it can now be easily imagined, this procedure is very time consuming and required much computational time, but it is very consistent to the model selection procedures that would be adopted in real time. As an additional challenge, the TV-PCR requires to build and extract the principal components of a $672 \times 85,344$ matrix. Our implementation strategy consisted of extracting and storing the list of matrices containing the principal components for each window and each of the ten target variables separately from the actual estimation of the TV-PCR. This allowed us to extract the "time-varying" components only once, making the evaluation computationally feasible. We used the R package RSpectra to extract the first 500 components of the modified matrix using an efficient algorithm for singular value decomposition. The overall computational time required for evaluating each model, at each horizon and for each target variable are reported in Table B.4.

4. Results

In this section, we report the results of the rolling-window forecast evaluation. We evaluate the models of Section 2 according to the root mean squared errors (RMSE) of their forecasts for four different horizons. We report the results separately for the full sample and the pre-Covid period, as the economic crisis caused by the pandemic marked a strong temporary regime change in the time series that would risk dominating the results if the sub-sample results were not also compared. Beside comparing the RMSE, we also analyse the evolution of the forecast errors over time and their correlation with the predictors to study possible sources of information that the model does not capture.

RMSE of the forecasts Table 2 and Table 3 report the RMSE respectively over the full sample and in the pre-Covid period for the 10 target variables. The values in the tables represent the ratio of the RMSE of each model over the RMSE of our benchmark model, PCR. Hence, values smaller than one indicate that our models perform better than our benchmark. The tables are organized in a way that makes it easy to distinguish different groups of models. Indeed, the first 5 models represent our linear models, followed by the non-linear manifold-learning techniques. Then, we have the non-linear PC-based models where we separate the five versions of the TVPCR from the Kernel and Quadratic PCR. As the last non linear-model we have the D2FM. In the tables we also include the p-value for the one-sided Diebold and Mariano (1995) test of the superior forecasting ability of the models over the PCR benchmark.

Looking at the results, we can see that there is no clear best model or clear winner between linear and nonlinear techniques across all horizons and variables. However, there are some interesting results. For the UNRATE, the best model for all horizons, both over the full sample and pre-covid, is always a non-linear model. Still, there is not a clear pattern in the winning models, as t-SNE is the best model for nowcasting in full sample and in the 1-month ahead pre-Covid forecast, while the Gaussian Kernel PCR wins in the pre-Covid nowcasting and the 1-quarter ahead forecasting. It is interesting to see that D2FM is the best model for both the 1-month and 1-quarter forecast in the full sample, but its errors are not significantly different from those of PCR in terms of the Diebold-Mariano test. We can see another interesting result for GS10, where our nonlinear models win over the linear models in all tables except for pre-Covid nowcasting, where the best model is the PCR benchmark. In this case, we can also notice that the best models in the full sample always belong to the group of the manifold-learning techniques, with LLE prevailing in backcasting and nowcasting and t-SNE being better for the 4 and 6-step forecasts. We can also see something interesting for the S.P500, where it seems like nonlinear models always help to predict the present and future of the main financial market benchmark. In fact, with the exception of backcasting, nonlinear models win in all tables, both full-sample and pre-Covid. Finally, HOUST represents the only result in net favor of linear models, as LASSO is always the best model for this variable, except for the 1-quarter full-sample forecast, where it is still the linear benchmark that wins. Overall, it is clear from the results that there is no universally superior type of model, but that the optimal model may vary depending on the variable in question and the specific time horizon, and sometimes there may simply be no superior model. However, we still find some interesting results as the superiority of nonlinear models for UNRATE, GS10 and the future performance of S.P500.

(a) *Backcasting (1-step prediction)*

Specification	UNRATE	W875RX1	GS10	CPIAUCSL	WPSFD49207	PAYEMS	HOUST	INDPRO	M2SL	S.P.500
AR(1)	1.0765	8.3529	1.1125	102.3602	62.8883	192.2805	89.5411	46.6125	70.5774	6.6059
LASSO	0.9824	0.9170 *	0.9639	**0.8921 .**	0.9799	0.7891	**0.8372 ****	1.1400	1.0262	1.0293
LASSO PCR	0.9573	0.9124 *	0.9124 .	0.9473	0.9850	0.8602	1.0273	1.0299	1.0290	**0.9963**
DFM	0.8523	0.8451 .	0.9423	1.0178	1.0500	0.7997	29.1936	1.0147	**0.9262**	1.0112
DFM (no pre-imputation)	2.9375	1.2017	0.9865	0.9879	0.9763	3.5989	23.6515	0.9785	1.0988	0.9968
Diffusion Map	**0.8147**	0.9821	1.5057	1.5676	1.4290	**0.6924**	1.0890	0.9783	1.0987	1.2749
t-SNE	0.8206	0.8392 .	0.8987 .	1.0017	0.9998	0.8247	1.0646	1.0195	0.9923	1.0271
LLE	0.8213	**0.8384 .**	**0.8887 .**	1.0015	1.0015	0.8323	1.0775	1.0392	0.9925	1.0249
ISOMAP	0.8219	0.8502 *	0.8982 .	1.0038	1.0066	0.8214	1.0990	**0.9779**	0.9790	1.0243
Quadratic PCR	2.1255	1.8270	1.3642	1.4795	1.5694	1.3971	1.1383	1.7698	2.2359	1.7776
Kernel PCR (quadratic)	3.2980	1.1882	3.4411	2.7865	3.0178	1.1430	1.1511	1.3211	1.4172	3.9906
Kernel PCR (Gaussian)	0.8820	0.9205	0.9463	1.0259	1.0217	0.9324	1.0719	1.0925	1.0393	1.0125
TV-PCR (50 components)	1.2243	1.6250	1.2579	1.0599	**0.9298**	1.2647	1.2583	1.5481	1.5175	1.0561
TV-PCR (100 components)	1.2679	1.3623	1.4590	1.4453	1.1246	1.3592	1.3935	1.8131	1.9205	1.3552
Ridge TV-PCR	0.8840	0.8936 *	0.9429	0.9886	1.0297	0.8638	1.5100	1.0378	0.9713	1.0334
LASSO TV-PCR	0.9933	1.0660	0.9164	1.0050	1.0066	0.7797	1.1199	1.0849	1.0289	1.0202
Adaptive LASSO TV-PCR	1.0123	1.1637	1.0698	1.1495	1.0813	0.7715	1.2923	1.2205	1.1238	1.0793
D2FM	0.8853	0.8390 .	1.0033	1.0306	1.1085	0.7659	2.4389	1.0048	0.9447	1.0722

(b) *Nowcasting (3-step prediction)*

Specification	UNRATE	W875RX1	GS10	CPIAUCSL	WPSFD49207	PAYEMS	HOUST	INDPRO	M2SL	S.P.500
AR(1)	0.9814	0.9076 .	0.9896	8.8986	11.3613	72.4222	70.6199	8.6699	7.2384	1.1128
LASSO	0.9948	0.8613 .	1.0055	1.0206	0.9825	0.8304	**0.8968 ***	0.9703	0.9695 *	0.9885 **
LASSO PCR	1.0112	0.8604 .	1.0012	1.0234	**0.9810**	0.8068	0.9163	0.9592	0.9638 .	0.9940
DFM	0.9899	**0.8601 .**	0.9820	1.0198	0.9827	**0.8063**	23.1292	0.9057	0.9545 .	0.9894 **
DFM (no pre-imputation)	3.2043	1.1388	0.9868	1.0196	0.9827	3.3296	23.1578	**0.9029**	1.0537	0.9888 **
Diffusion Map	1.1499	1.3715	2.2538	2.0611	2.3426	1.5029	1.9339	1.3533	0.9886	1.5935
t-SNE	**0.9535**	0.8759 .	0.9916	1.0267	0.9832	0.8833	1.1416	0.9617	0.9820 *	**0.9869 ***
LLE	0.9552	0.8804 .	**0.9795**	1.0263	0.9832	0.8774	1.1376	0.9648	0.9813 **	0.9998
ISOMAP	0.9702	0.9293	0.9915	1.0314	0.9827	0.8988	1.1658	0.9603	0.9817 *	0.9993
Quadratic PCR	1.3603	1.4344	2.0253	1.4911	1.1847	1.2166	1.3268	1.3775	1.3789	1.2479
Kernel PCR (quadratic)	2.0537	2.8329	1.6042	3.8741	3.2588	1.5059	2.5675	2.2406	2.6509	2.4721
Kernel PCR (Gaussian)	1.0599	0.9625	1.0220	1.0354	0.9973	0.8416	1.2587	0.9755	0.9930 *	0.9935
TV-PCR (50 components)	1.3280	1.0749	1.4456	1.3426	1.2142	1.1951	1.1647	1.1545	1.1373	1.2547
TV-PCR (100 components)	1.1741	1.2301	1.7952	1.4257	1.5334	1.0653	1.7241	1.3319	1.3826	1.6037
Ridge TV-PCR	0.9730	0.8886 .	1.0752	1.0722	1.0121	0.8677	1.6372	0.9479	0.9818	0.9936
LASSO TV-PCR	1.0351	0.8607 .	0.9936	1.0145	0.9816	0.8115	1.1145	0.9466	**0.9417 .**	0.9881 *
Adaptive LASSO TV-PCR	1.0661	0.8948	1.0554	1.0071	1.0005	0.9148	1.1911	0.9698	0.9451	0.9990
D2FM	0.9686	0.8933 .	1.0144	1.0359	1.0035	0.8103	1.5883	0.9057	0.9579	1.0490

(c) *1-month ahead forecasting (4-step prediction)*

Specification	UNRATE	W875RX1	GS10	CPIAUCSL	WPSFD49207	PAYEMS	HOUST	INDPRO	M2SL	S.P.500
AR(1)	0.8524	0.9466	0.9843	2.6205	5.0096	45.7130	67.3890	4.2870	2.9774	1.0263
LASSO	0.8782	0.9032 .	0.9903	1.0125	**0.9916 .**	0.7988	**0.9783**	0.9707	0.9978	1.0037
LASSO PCR	0.8572	0.9082 .	0.9888	1.0173	0.9920 .	0.8026	1.0263	0.9555	0.9947	1.0082
DFM	0.8548	0.8999 .	0.9874	1.0131	0.9917 .	0.8027	22.0900	0.9472	1.0101	1.0084
DFM (no pre-imputation)	2.7662	1.1944	0.9906	1.0131	**0.9916 .**	3.3160	22.0277	**0.9445**	1.1174	1.0081
Diffusion Map	1.6668	1.3407	1.9502	1.0140	1.8001	1.4367	2.2043	1.0340	1.0581	1.7106
t-SNE	0.8749	0.9124 .	**0.9774 .**	1.0161	0.9946	0.9104	1.1636	0.9762 .	**0.9931 ***	1.0052
LLE	0.8648	0.9125	0.9821 .	1.0172	0.9933 .	0.8899	1.1179	0.9602 .	0.9933 *	1.0008
ISOMAP	0.8981	0.9407	0.9928	1.0150	0.9926 *	0.9160	1.1479	0.9643	0.9932 *	1.0006
Quadratic PCR	1.3615	1.4732	2.9255	2.1004	1.5340	1.1302	1.2609	1.0428	1.1277	1.1633
Kernel PCR (quadratic)	2.6193	2.6748	4.9515	2.5585	1.9832	1.8217	3.8235	1.6925	1.6666	2.3190
Kernel PCR (Gaussian)	0.8862	0.9874	1.0588	1.0338	0.9962	0.8412	1.2090	0.9643	0.9959 *	**0.9990**
TV-PCR (50 components)	0.8688	1.0799	1.6163	1.2290	1.2933	1.1179	1.1759	1.1502	1.1295	1.4652
TV-PCR (100 components)	0.8965	1.2926	1.6868	1.4907	1.3766	0.9595	1.3058	1.3729	1.2210	1.8029
Ridge TV-PCR	0.8967	0.9084	1.0259	1.0555	1.0139	0.8839	1.5819	0.9746	1.0137	1.0418
LASSO TV-PCR	0.8726	**0.8992 .**	1.0050	1.0114	1.0060	0.7965	1.0862	0.9806	1.0032	1.0075
Adaptive LASSO TV-PCR	0.9112	0.9033 .	1.0981	1.0044	1.0408	**0.7869**	1.1930	0.9920	1.0110	1.0175
D2FM	**0.8306**	0.9340 ***	1.0098	1.0374	0.9948	0.8280	1.5566	0.9532	1.0253	1.1336

(d) *1-quarter ahead forecasting (6-step prediction)*

Specification	UNRATE	W875RX1	GS10	CPIAUCSL	WPSFD49207	PAYEMS	HOUST	INDPRO	M2SL	S.P.500
AR(1)	0.9003	0.9476	0.9757 *	1.0076	1.4945	20.3753	71.9371	1.4199	**0.9867**	1.0070
LASSO	0.9259	**0.9003**	0.9782 *	0.9942	1.0002	0.8668	1.0014	0.9795	0.9977	1.0217
LASSO PCR	0.9104	0.9304	0.9843 .	0.9904	1.0080	0.8666	1.1492	0.9792	1.0087	1.0034
DFM	0.8977	0.9006	0.9824 .	0.9941	1.0119	**0.8657**	23.6518	0.9758	0.9985	0.9916
DFM (no pre-imputation)	2.9210	1.1957	0.9813	0.9942	1.0118	3.5797	23.6239	**0.9731**	1.1049	0.9914
Diffusion Map	0.9911	0.9298	2.1618	1.2572	1.2400	2.0341	2.7944	1.0345	1.0488	2.3696
t-SNE	0.9057	0.9226	**0.9650 *** **	**0.9888**	1.0036	0.9240	1.2905	0.9894	0.9945	0.9960
LLE	0.9050	0.9209	0.9715 .	0.9945	**0.9999**	0.9174	1.2014	0.9798	0.9922 *	0.9910
ISOMAP	0.9178	0.9512 .	0.9845	0.9963	1.0010	0.9396	1.2525	0.9786	0.9931 .	0.9989
Quadratic PCR	1.0333	0.9790	2.8954	1.3714	1.4419	1.0821	1.1398	1.1004	1.0770	1.5811
Kernel PCR (quadratic)	2.2993	1.9448	2.2617	3.1574	2.1767	2.2420	3.0762	2.1991	1.4028	2.6514
Kernel PCR (Gaussian)	0.8865	0.9251	0.9697	0.9962	1.0079	0.8665	1.3289	0.9805	1.0001	**0.9799**
TV-PCR (50 components)	1.3322	1.2173	1.3211	1.3409	1.2449	1.6522	1.5731	1.7831	1.4061	1.1461
TV-PCR (100 components)	1.5356	1.7721	1.5546	2.5103	2.2729	1.8380	1.7792	2.6134	1.8227	1.4260
Ridge TV-PCR	0.9180	0.9641	1.0367	1.0052	1.0081	0.9299	1.7267	1.0099	1.0225	1.0022
LASSO TV-PCR	0.9203	0.9016	0.9945	0.9900	1.0253	0.8666	1.1649	0.9778	1.0036	1.0032
Adaptive LASSO TV-PCR	0.9630	0.9292	1.1056	1.0268	1.0522	0.8672	1.3015	0.9895	1.0395	1.0630
D2FM	**0.8789**	0.9170	0.9916	1.0182	1.0128	0.8684	1.6266	0.9781	1.0088	1.1245

Note: '***' $p < 0.001$, '**' $p < 0.01$, '*' $p < 0.05$, '.' $p < 0.1$

Table 2: *Relative RMSFE over Principal Component Regression (benchmark model) in a pseudo-real-time rolling window evaluation over the whole sample. Size of the rolling window: 672 months. Test period: January 2015 – January 2023. Significance levels refers to the one-sided Diebold-Mariano test of the superior forecasting ability over the benchmark.*

(a) *Backcasting (1-step prediction)*

Specification	UNRATE	W875RX1	GS10	CPIAUCSL	WPSFD49207	PAYEMS	HOUST	INDPRO	M2SL	S.P.500
AR(1)	1.4412	7.3246	1.3168	110.5347	72.7760	208.2704	103.3908	46.9342	117.0522	7.0412
LASSO	1.0012	0.9923	0.9930	**0.8466 ***	**0.9573**	0.7708 **	**0.8746 ****	1.0458	1.0097	1.0474
LASSO PCR	0.9941	1.0206	0.9830	0.9306 .	1.0131	0.7736 **	0.9187 .	1.0296	1.0390	1.0213
DFM	1.0261	1.0032	1.0087	1.0337	1.0861	**0.7565 ****	33.8470	1.0463	1.1421	1.0332
DFM (no pre-imputation)	1.0471	1.0141	0.9868	1.0054	0.9948	0.9135 ***	29.1983	0.9882	0.9870	1.0046
Diffusion Map	0.9781	**0.9853**	1.0476	1.0264	1.0218	1.0219	1.0582	1.0003	**0.9829**	1.0291
t-SNE	1.0630	1.0093	0.9997	1.0008	1.0224	0.9128	1.1139	1.0063	1.0039	1.0671
LLE	0.9677	1.0087	1.0059	1.0049	1.0253	0.8912 .	1.1184	0.9939	1.0163	1.0432
ISOMAP	**0.9290 .**	1.0282	1.0073	1.0052	1.0235	0.9565	1.1312	1.0613	1.0146	1.0416
Quadratic PCR	1.0233	1.0362	**0.9783**	1.0432	1.0656	0.9880	1.1364	1.0378	1.0276	1.0654
Kernel PCR (quadratic)	0.9810	1.0214	0.9941	0.9874	1.0270	0.9473	1.1216	1.0729	0.9849	1.0303
Kernel PCR (Gaussian)	0.9758	0.9936	1.0166	1.0158	1.0116	1.0971	1.0600	1.0631	1.0154	1.0260
TV-PCR (50 components)	0.9726	1.2512	1.2931	0.9631	0.9951	1.1287	1.1297	1.1532	1.1801	1.0466
TV-PCR (100 components)	1.0071	1.6238	1.5063	1.0646	1.0718	1.4394	1.0816	1.2330	1.2095	1.1795
Ridge TV-PCR	1.0704	1.1306	1.0304	0.9830	1.0717	0.8463 *	1.6019	1.0758	1.1143	1.0615
LASSO TV-PCR	1.0265	1.0460	1.0010	0.9527	1.0047	0.7594 **	1.1540	0.9984	1.0354	1.0407
Adaptive LASSO TV-PCR	1.2435	1.2801	1.1166	1.1439	1.0518	0.9169	1.3142	0.9847	1.1633	1.1870
D2FM	1.1457	1.0141	1.0759	1.0453	1.2214	0.9937	2.5397	**0.9422**	1.1667	1.2029

(b) *Nowcasting (3-step prediction)*

Specification	UNRATE	W875RX1	GS10	CPIAUCSL	WPSFD49207	PAYEMS	HOUST	INDPRO	M2SL	S.P.500
AR(1)	1.0323	1.2756	1.0083	11.0975	14.1451	82.0812	101.4026	5.6153	11.5644	1.1690
LASSO	1.0203	1.0093	1.0446	0.9854	0.9845 *	**0.8158 ****	**0.8689 ****	1.0385	**0.9501 ***	0.9714 *
LASSO PCR	1.0093	0.9959	1.0462	0.9894	0.9814 **	0.8645 *	0.9205	1.0348	0.9567 .	0.9886
DFM	1.0264	0.9928	1.0145	0.9826	0.9848 *	0.8481 *	33.4648	1.0358	0.9729	0.9720 **
DFM (no pre-imputation)	1.0173	0.9931	1.0036	0.9824	0.9848 *	0.8480 *	33.2363	1.0366	0.9729	0.9710 **
Diffusion Map	0.9774	1.0851	1.0342	0.9966	0.9903	1.1052	1.1056	0.9513 *	0.9938	0.9776
t-SNE	1.1033	**0.9306 ***	1.0234	0.9932	0.9889	0.9586	1.0381	**0.9398 .**	1.0075	0.9801 *
LLE	1.0423	1.0304	1.0191	0.9920	0.9848 *	0.8605 *	1.0590	1.0010	1.0010	0.9902
ISOMAP	1.0176	1.0999	1.0295	0.9975	0.9865 *	0.9940	1.0683	1.0935	1.0018	0.9896 .
Quadratic PCR	1.0494	1.0187	1.0175	0.9806 .	0.9885	0.9216 .	1.0843	1.0592	1.0076	0.9818
Kernel PCR (quadratic)	1.0236	1.0325	1.0139	1.0006	**0.9728 ***	0.8903 *	1.1278	1.0161	0.9923	0.9870
Kernel PCR (Gaussian)	**0.9752**	1.1145	1.0353	1.0036	0.9786 **	1.0826	1.0332	1.0175	0.9967	1.0007
TV-PCR (50 components)	1.0865	1.3042	1.2424	1.2225	1.1627	1.1569	1.1084	0.9515	1.0391	1.0562
TV-PCR (100 components)	1.2094	1.8069	1.3845	1.3592	1.2148	1.2323	1.1834	1.2659	1.1783	1.2680
Ridge TV-PCR	1.0100	1.1834	1.1557	1.0424	1.0346	0.8463 *	1.4565	1.0416	1.0559	**0.9631**
LASSO TV-PCR	1.0189	1.0168	1.0180	0.9762	0.9838 .	0.8500 *	1.0999	1.0255	0.9872	0.9754 *
Adaptive LASSO TV-PCR	1.2269	1.3289	1.1137	**0.9759**	1.0246	0.8874 .	1.2526	1.1315	1.0466	1.0035
D2FM	1.0040	1.2476	1.0080	1.0216	1.0418	0.9307	2.4333	1.0291	0.9760	1.0477

(c) *1-month ahead forecasting (4-step prediction)*

Specification	UNRATE	W875RX1	GS10	CPIAUCSL	WPSFD49207	PAYEMS	HOUST	INDPRO	M2SL	S.P.500
AR(1)	0.9729	1.3531	0.9846	3.3698	6.4371	52.0580	91.9495	2.0638	4.4778	1.0522
LASSO	0.9773	1.0084	0.9879	**0.9995**	0.9934	0.8824 *	**0.8653 ***	0.9847	1.0402	1.0042
LASSO PCR	0.9871	1.0061	0.9843	1.0158	0.9943	0.8929 .	0.9609	0.9936	**0.9614**	1.0135
DFM	0.9718	1.0104	0.9905	1.0017	0.9934	**0.8799 ***	30.4276	0.9837	0.9816	1.0036
DFM (no pre-imputation)	0.9624 .	1.0110	0.9889	1.0020	0.9935	0.8800 *	30.1005	0.9844	0.9817	1.0036
Diffusion Map	1.0512	1.0103	1.0129	1.0156	0.9999	1.1994	1.1169	0.9430 *	1.0074	1.0308
t-SNE	**0.9324 .**	1.0343	0.9923	1.0049	0.9916	1.0306	1.0731	0.9324 .	0.9931	1.0071
LLE	0.9938	1.0503	0.9909	1.0050	0.9899 .	0.9764	0.9868	0.9437 *	0.9895 *	0.9994
ISOMAP	1.0160	1.0554	0.9991	1.0065	0.9893 .	1.0726	1.0028	0.9895	0.9906 *	**0.9985**
Quadratic PCR	0.9724	1.0424	0.9839	1.0136	0.9980	1.1663	1.0629	0.9952	0.9875 *	1.0172
Kernel PCR (quadratic)	0.9586	1.0329	**0.9816**	1.0075	**0.9884 .**	1.0687	1.0999	0.9777	0.9870 .	1.0045
Kernel PCR (Gaussian)	1.0233	1.0685	1.0044	1.0282	0.9972	1.0971	0.9664	1.0066	1.0000	1.0211
TV-PCR (50 components)	1.0232	1.2683	1.1600	1.1307	1.2392	1.4089	1.0559	1.0055	1.1675	1.2729
TV-PCR (100 components)	1.1825	1.7543	1.3698	1.4381	1.2891	1.7947	1.1468	1.1496	1.3963	1.4948
Ridge TV-PCR	1.0476	1.1324	1.0815	1.0771	1.0423	0.9382	1.4974	1.0036	1.1053	1.0856
LASSO TV-PCR	0.9899	1.0049	0.9897	1.0022	1.0386	0.8907 *	0.9716	0.9452 *	0.9979	1.0027
Adaptive LASSO TV-PCR	1.1952	1.0536	1.1521	1.0046	1.1010	0.9278	1.0727	**0.9308 .**	1.1491	1.0109
D2FM	0.9520 *	1.1840	1.0025	1.0459	0.9996	1.0546	2.0464	0.9721	1.0233	1.1948

(d) *1-quarter ahead forecasting (6-step prediction)*

Specification	UNRATE	W875RX1	GS10	CPIAUCSL	WPSFD49207	PAYEMS	HOUST	INDPRO	M2SL	S.P.500
AR(1)	1.0478	1.3440	0.9826 *	1.0474	1.5763	20.2835	87.6376	**0.8811 .**	1.0710	1.0419
LASSO	0.9991	1.0126	0.9829 *	1.0051	0.9990	0.9285 ***	**0.9052 .**	0.9877	0.9840	1.0062
LASSO PCR	1.0269	1.0079	0.9827 *	1.0060	**0.9778 .**	0.9172 ***	0.9490	0.9951	1.0076	0.9923
DFM	1.0471	1.0141	0.9868	1.0054	0.9948	**0.9135 *****	29.1983	0.9882	0.9870	1.0046
DFM (no pre-imputation)	1.0318	1.0144	0.9854	1.0051	0.9946	**0.9135 *****	28.9009	0.9887	0.9869	1.0042
Diffusion Map	1.0003	1.0427	1.0074	1.0115	1.0008	1.2593	1.1272	0.9699 *	0.9938	0.9989
t-SNE	1.0235	1.0631	**0.9667 .**	0.9981	1.0022	1.1531	1.1371	1.0035	1.0057	0.9962
LLE	1.0188	1.0201	0.9947	1.0061	0.9920	1.0004	1.0210	0.9476 **	0.9916	**0.9899 .**
ISOMAP	1.0176	1.0210	1.0053	1.0077	0.9972	1.0655	1.0413	1.0233	0.9937	0.9917
Quadratic PCR	1.0255	1.0448	0.9875	1.0096	1.0034	1.0418	1.0409	0.9828	1.0005	1.0057
Kernel PCR (quadratic)	1.0162	1.0319	0.9994	1.0106	1.0000	1.0343	1.1054	0.9430 *	0.9978	1.0038
Kernel PCR (Gaussian)	**0.9956**	1.0681	1.0003	1.0047	1.0010	1.1853	1.0231	0.9977	0.9999	0.9947
TV-PCR (50 components)	1.2233	1.5211	1.2188	1.2765	1.0778	1.3927	1.1673	1.0118	1.2239	1.1332
TV-PCR (100 components)	1.4692	1.8664	1.2060	1.5677	1.3312	1.7396	1.2648	0.9945	1.5889	1.2498
Ridge TV-PCR	1.1095	1.1595	1.0815	1.0278	0.9893	0.9182 *	1.3656	1.0685	**0.8878 ***	1.0527
LASSO TV-PCR	1.0399	1.0276	1.0049	**0.9870**	1.0198	0.9177 ***	0.9769	0.9553 **	1.0482	1.0268
Adaptive LASSO TV-PCR	1.2830	1.2381	1.2325	1.0417	1.0925	0.9573	1.2018	1.1163	1.2867	1.1499
D2FM	1.0158	1.1700	0.9796	1.0586	0.9976	1.0034	1.8995	1.0056	1.0310	1.2182

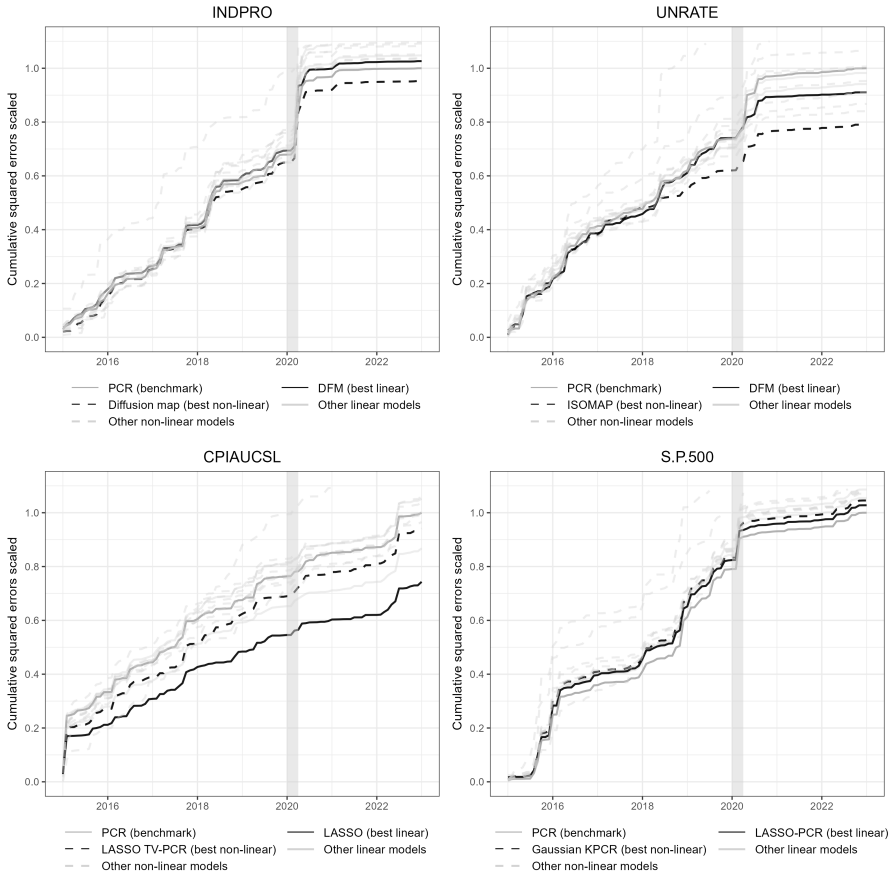
Note: '***' $p < 0.001$, '**' $p < 0.01$, '*' $p < 0.05$, '.' $p < 0.1$

Table 3: *Relative RMSFE over Principal Component Regression (benchmark model) in a pseudo-real-time rolling window evaluation over the pre-Covid sample. Size of the rolling window: 672 months. Test period: January 2015 – December 2019. Significance levels refers to the one-sided Diebold-Mariano test of the superior forecasting ability over the benchmark.*

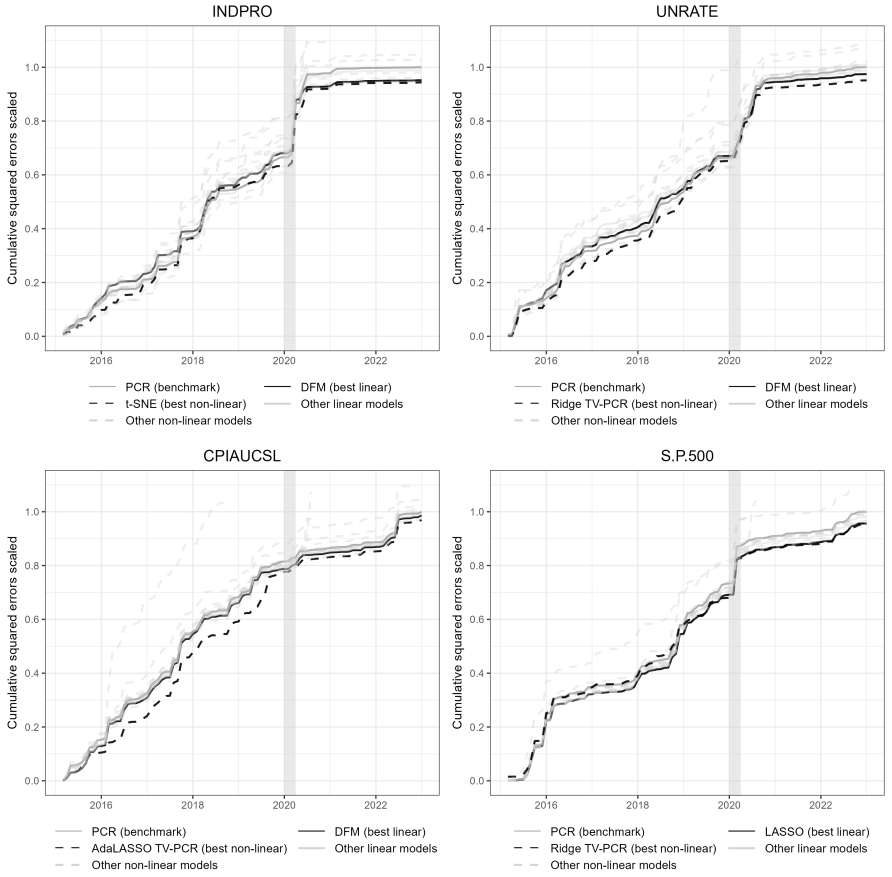
Analysis of the residuals As part of our analysis, we select four variables, each chosen for its relevance and importance in capturing key economic dynamics. We select UNRATE which represents the unemployment rate and provides valuable insights into labor market conditions. CPIAUCSL, the consumer price index, is chosen to capture inflationary trends. In order to analyze the performance of the real economy, we choose INDPRO, which represents industrial production and offers insights into economic activity. Lastly, we choose S.P500, which tracks the performance of the largest listed companies in the U.S., as a benchmark for the financial economy.

Figure 1 shows the plot of the cumulative squared errors over different horizons for our four variables, allowing a comprehensive comparison of the performance of the linear and nonlinear models against the benchmark (PCR). The shaded area in the plots corresponds to the Covid recession from January to April 2020. To draw these plots, we divided the forecast errors into pre and post-covid periods. We then calculate two means and two standard deviations and scaled the two sets of errors separately. This scaling was done in order to improve clarity and allow for a better representation of the relatively smaller pre-covid errors compared to post-covid ones. Finally, we calculate the cumulative squared errors for each model and scale them all by the total sum of errors of the benchmark model. In these figures, we highlight the curves of both the best linear and non-linear models in terms of full sample RMSE for each specific horizon and target variable. In this way, we are able to assess whether linear or nonlinear models are better in each case. It is important to note that the best model in the whole sample shown on the graph is not necessarily the best model in the whole sample in the tables, because of this separate scaling of the error before and after covid. From the backcasting plot, it can be seen how non-linear models perform better than linear models for only two out of the four variables, INDPRO and UNRATE. In particular, for the latter, the best linear model appears to be pretty distanced from the best non-linear one, ISOMAP, and there seem to be other nonlinear models that also fall below it. This is a nice result that highlights the superiority of nonlinear models for UNRATE. In contrast, our benchmark PCR seems to be undefeated for S.P500, for which we have seen in the tables that nonlinear models are best, but only for future predictions. Moving to nowcasting, we can notice that the variants of the TV-PCR perform better than the rest of the models for three variables. The AdaLASSO TV-PCR is the best model for CPIAUCSL and it clearly stands out from other models in the pre-covid period. For UNRATE and S.P500, Ridge TV-PCR is the best model and this is again consistent with the superiority of nonlinear models for UNRATE. Dealing with the 4-step predictions, we can see again that t-SNE, is the best non-linear model, with the lowest line over the whole period for UNRATE, and is particularly distanced in pre-Covid. Last, for the 6-step prediction, the LASSO TV-PCR is the best non-linear model and performs better than the best linear models, for both INDPRO and CPIAUCSL. However, the most interesting result is again a non-linear model being the best model for UNRATE. Overall, these graphs are consistent with the results in table 2 and table 3 in showing the superiority of nonlinear techniques for UNRATE prediction.

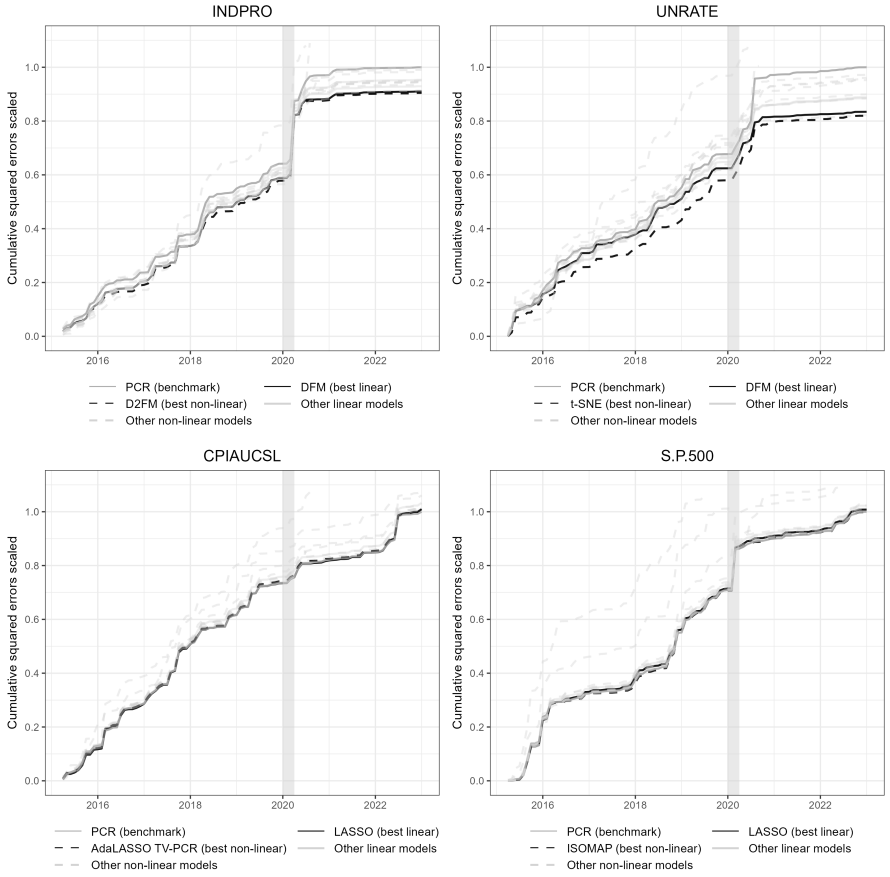
(a) *Backcasting (1-step prediction)*



(b) *Nowcasting (3-step prediction)*



(c) *1-month ahead forecasting (4-step prediction)*



(d) *1-quarter ahead forecasting (6-step prediction)*

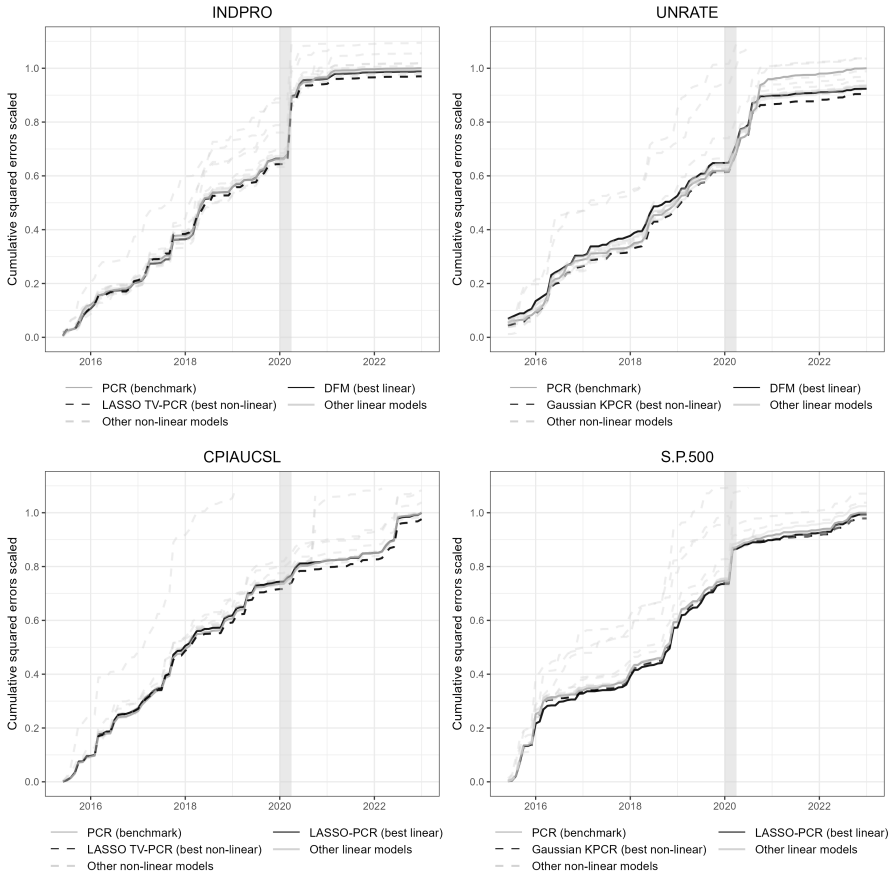


Figure 1: *Cumulative squared forecast errors for four policy-relevant variables. The cumulative squared errors are reported as the ratio to the total cumulative squared error of Principal Component Regression (benchmark model). The shaded area corresponds to the Covid-19 period. For each variable, the best linear and non-linear models are reported in black.*

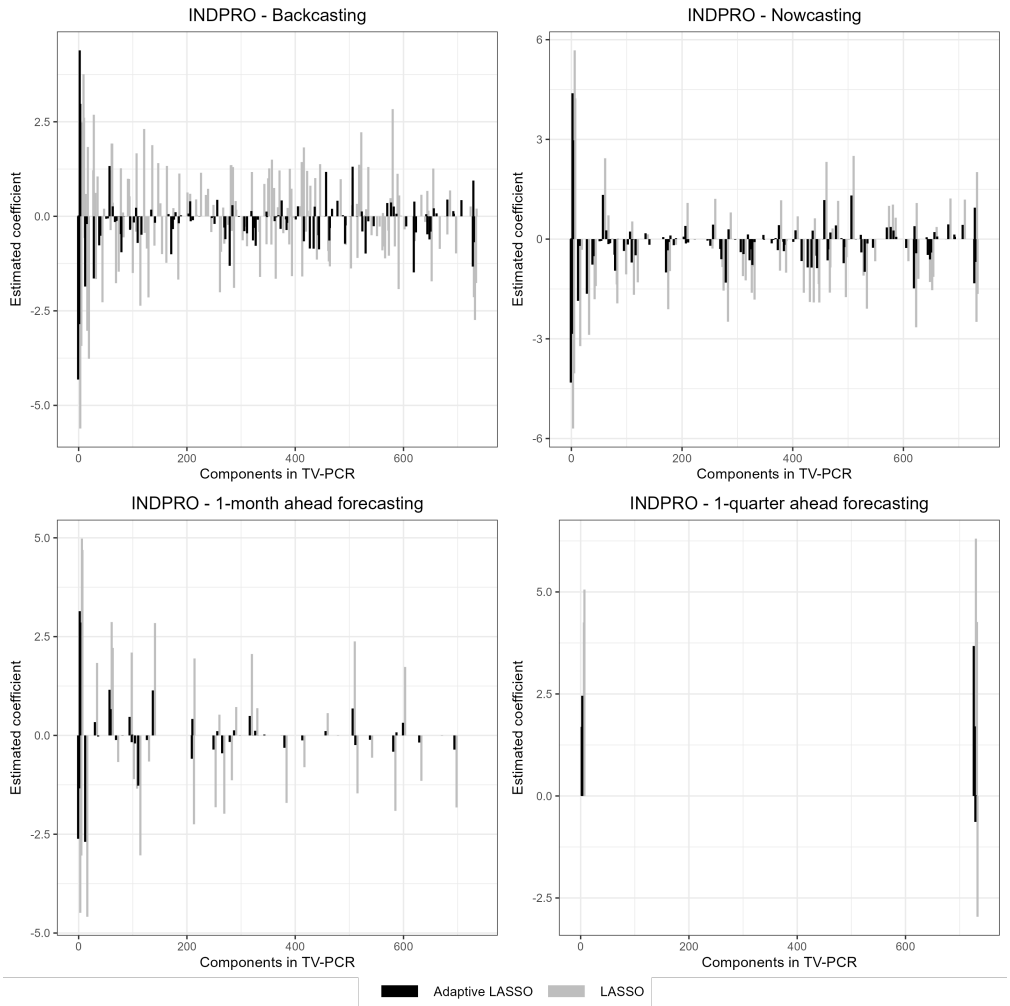


Figure 2: *Coefficients of LASSO TVPCR and AdaLASSO TVPCR for the index of industrial production.*

Selected components in Time-Varying PCR In the previous paragraph we have seen that our TVPCR is often the best model in forecasting the index of industrial production and the consumer price index. Let us dive deeper into the penalized version of the TVPCR, examining the estimated values of the coefficients for our LASSO and Adaptive LASSO models across our four different time horizons, as shown in figure 2. This comparison helps us understand the differences between the two methods and how their respective coefficients evolve. We can see that the Adaptive LASSO's coefficients are generally more shrunk towards zero compared to the LASSO's for all h-step forecasts. In particular, for backcasting, we can notice how the LASSO selects way more predictors than the Adaptive LASSO. This could be because Adaptive LASSO penalties, which are adjusted using weights determined by an initial LASSO model, are particularly effective in penalizing less influential predictors. Interestingly, as we attempt to predict further into the future, we can see that the time series cross-validation tends to select larger penalties that select fewer coefficients, with most of them being set to zero for 1-quarter ahead forecasting. This phenomenon is likely due to the increasing uncertainty as we move further into the future, which reduces the number of components useful for prediction.

4.1. Residuals correlations and limitations

We now turn to analyze the unexplained variance of the models by looking at how the residuals correlate with the covariates. Figure 3 shows the absolute value of the correlation between the residuals of each model for each target and all the variables that we have in our data set. In almost all of the models, with the exception of the AR(1), a clear pattern emerges, showing significant correlations with the same groups of variables. This correlation is particularly high, for the residuals of each target variable, with the covariates having similarities with the corresponding target variables. This suggests that our models' predictions show little variability compared to the variation in the ground truth. Consequently, the residuals exhibit a clear correlation with both the ground truth and other covariates that are associated with the target variables. The absence of (fix) this correlation, or a smaller value, is indicative of less-flat predictions produced by the models. However, this information alone does not provide any conclusive evidence that these non-flat predictions are better in terms of accuracy. In fact, forecasts with higher variance may be better or worse than 'flat' forecasts, and the RMSE table is needed to draw conclusions. For instance, the AR(1) model stands out from other models by not showing correlations with any group of covariates, but its predictions have much worse accuracy than the other models. The heatmap highlights a significant limitation in all models: a high level of unexplained variance in the residuals. Nevertheless, it is interesting to observe that four versions of TVPCR (TVPCR 50, TVPCR 100, LASSO TVPCR, and AdaLASSO TVPCR) exhibit less correlation between the residuals and the covariates when compared to other models while still achieving good performances in terms of RMSE.

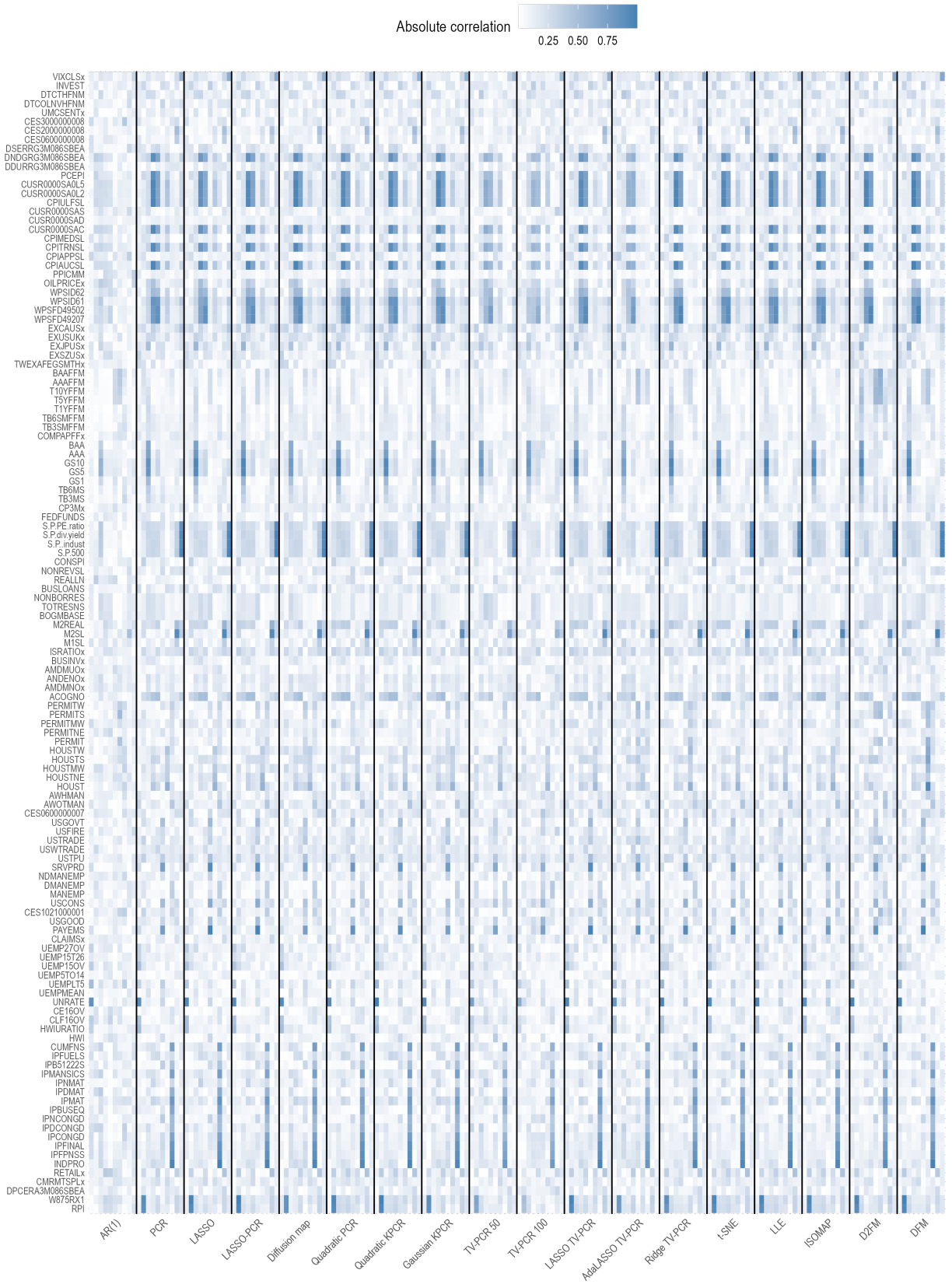


Figure 3: *Correlation of the residuals of each model with the predictors for the 1-step predictions in the pre-Covid sample.*

5. Conclusions and further research

In this work, we compared the forecasting performances of models based on linear and non-linear dimensionality reduction techniques using the FRED-MD dataset. The main aim of the work was to study whether allowing for non-linear mappings can lead to better forecasting performances than linear methods like Principal Component Regression or Dynamic Factor Models. Overall, the analysis carried out in this study reveals no universally superior model, but still provides a wide range of insights. The optimal prediction model seems to depend largely on the variable to be predicted and the specific prediction horizon, and sometimes there is no model among the ones considered performing better that PCR. However, our analysis shows a significant outperformance of nonlinear models in predicting an important variable, UNRATE. All our results indicate that nonlinear techniques improve the predictions of the U.S. Unemployment Rate for backcasting, nowcasting, 1-month ahead and 1-quarter ahead forecasting, both when we consider the full test sample and when we just consider the period before the Covid-19 pandemic. Such results are interesting because the unemployment rate is a relevant variable for policy-making. This outcome suggests that nonlinear models may be well-suited to capture patterns overlooked by their linear counterparts. The other interesting results are the better forecasts of the future performance of S&P 500 and the 10-Year Treasury Rate (GS10), with nonlinear models. However, we must also recognize cases such as the housing starts (HOUST) in which linear models, particularly LASSO, have shown consistently superior performance. Another important consideration is that we still find a significant amount of unexplained variance in the residuals across most of the models and target variables, indicating that the models are unable to capture much of the variability in the realized values. Apart from including additional specifications to the comparison, it might be interesting to explore the potential of ensemble models incorporating both linear and non-linear specifications. The key principle would be to exploit the strengths of both types of models, potentially achieving higher predictive accuracy.

References

Andreini, Paolo, Cosimo Izzo, and Giovanni Ricco (2020). *Deep Dynamic Factor Models*. Papers 2007.11887. arXiv.org.

Angerer, Philipp, Laleh Haghverdi, Maren Büttner, Fabian Theis, Carsten Marr, and Florian Buettner (2015). "destiny: diffusion maps for large-scale single-cell data in R". In: *Bioinformatics* 32.8, pp. 1241–1243.

Bai, Jushan and Serena Ng (2008). "Forecasting economic time series using targeted predictors". In: *Journal of Econometrics* 146.2, pp. 304–317.

Barigozzi, Matteo and Matteo Luciani (2019). *Quasi Maximum Likelihood Estimation and Inference of Large Approximate Dynamic Factor Models via the EM algorithm.*

Bartenhagen, Christoph (2023). *RDRToolbox: A package for nonlinear dimension reduction with Isomap and LLE.* R package version 1.50.0.

Bernanke, Ben S., Jean Boivin, and Piotr Eliasz (Feb. 2005). "Measuring the Effects of Monetary Policy: A Factor-Augmented Vector Autoregressive (FAVAR) Approach". In: *The Quarterly Journal of Economics* 120.1, pp. 387–422.

Cho, Kyunghyun, Bart van Merrienboer, Dzmitry Bahdanau, and Yoshua Bengio (2014). *On the Properties of Neural Machine Translation: Encoder-Decoder Approaches*. arXiv: 1409.1259.

Chung, Junyoung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio (2014). *Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling*. arXiv: 1412.3555.

Coifman, R. R., S. Lafon, A. B. Lee, M. Maggioni, B. Nadler, F. Warner, and S. W. Zucker (2005). "Geometric diffusions as a tool for harmonic analysis and structure definition of data: Diffusion maps". In: *Proceedings of the National Academy of Sciences* 102.21, pp. 7426–7431.

Coifman, Ronald R. and Stéphane Lafon (2006). "Diffusion maps". In: *Applied and Computational Harmonic Analysis* 21.1. Special Issue: Diffusion Maps and Wavelets, pp. 5–30. ISSN: 1063-5203.

Cybenko, George V. (1989). "Approximation by superpositions of a sigmoidal function". In: *Mathematics of Control, Signals and Systems* 2, pp. 303–314.

Delaporte, Jacqueline, Ben M. Herbst, Willy A. Hereman, and Van der Walt Stéfan (2008). *An introduction to diffusion maps*. Tech. rep.

Diebold, Francis and Roberto Mariano (1995). "Comparing Predictive Accuracy". In: *Journal of Business & Economic Statistics* 20, pp. 134–44.

Forni, Mario, Marc Hallin, Marco Lippi, and Lucrezia Reichlin (2000). "The Generalized Dynamic-Factor Model: Identification and Estimation". In: *The Review of Economics and Statistics* 82.4, pp. 540–554.

Ghojogh, Benyamin, Ali Ghodsi, Fakhri Karray, and Mark Crowley (2020). *Locally Linear Embedding and its Variants: Tutorial and Survey*. arXiv: 2011.10925 [stat.ML].

Gorban, Alexander, Balázs Kégl, Donald Wunsch, and Andrei Zinovyev (2008). *Principal Manifolds for Data Visualisation and Dimension Reduction, LNCSE 58*.

Hastie, Trevor and Werner Stuetzle (1989). "Principal Curves". In: *Journal of the American Statistical Association* 84.406, pp. 502–516.

Hauzenberger, Niko, Florian Huber, and Karin Klieber (2023). "Real-time inflation forecasting using non-linear dimension reduction techniques". In: *International Journal of Forecasting* 39.2, pp. 901–921.

Hinton, Geoffrey E and Sam Roweis (2002). "Stochastic Neighbor Embedding". In: *Advances in Neural Information Processing Systems*. Ed. by S. Becker, S. Thrun, and K. Obermayer. Vol. 15. MIT Press.

Hornik, Kurt, Maxwell Stinchcombe, and Halbert White (1989). "Multilayer feedforward networks are universal approximators". In: *Neural Networks* 2.5, pp. 359–366.

Hotelling, Harold (1957). "The Relations of the Newer Multivariate Statistical Methods to Factor Analysis". In: *British Journal of Statistical Psychology* 10.2, pp. 69–79.

Kim, Hyun Hak and Norman R. Swanson (2011). *Forecasting financial and macroeconomic variables using data reduction methods: New empirical evidence*. Working Paper 2011-19.

Kock, Anders Bredahl and Timo Teräsvirta (2011). "61 Forecasting With Nonlinear Time Series Models". In: *The Oxford Handbook of Economic Forecasting*. Oxford University Press.

Kramer, Mark A. (1991). "Nonlinear principal component analysis using autoassociative neural networks". In: *AIChE Journal* 37.2, pp. 233–243.

Krijthe, Jesse H. (2015). *Rtsne: T-Distributed Stochastic Neighbor Embedding using Barnes-Hut Implementation*. R package version 0.16.

Maaten, Laurens van der and Geoffrey Hinton (Nov. 2008). "Viualizing data using t-SNE". In: *Journal of Machine Learning Research* 9, pp. 2579–2605.

Malthouse, E.C. (1998). "Limitations of nonlinear PCA as performed with generic neural networks". In: *IEEE Transactions on Neural Networks* 9.1, pp. 165–173.

McCracken, Michael W. and Serena Ng (2016). "FRED-MD: A monthly database for macroeconomic research". In: *Journal of Business & Economic Statistics* 34.4, pp. 574–589.

Million, Elizabeth (2007). "The Hadamard Product Elizabeth Million April 12 , 2007 1 Introduction and Basic Results". In.

Richards, Joseph and Robrecht Cannoodt (2019). *diffusionMap: Diffusion Map*. R package version 1.2.0.

Sargent, Thomas and Christopher Sims (1977). *Business cycle modeling without pretending to have too much a priori economic theory*. Working Papers 55. Federal Reserve Bank of Minneapolis.

Saul, Lawrence K. and Sam T. Roweis (2001). "An Introduction to Locally Linear Embedding". In.

Schölkopf, Bernhard, Alex Smola, and Klaus-Robert Müller (1998). "Nonlinear Component Analysis as a Kernel Eigenvalue Problem". In: *Neural Computation* 10, pp. 1299–1319.

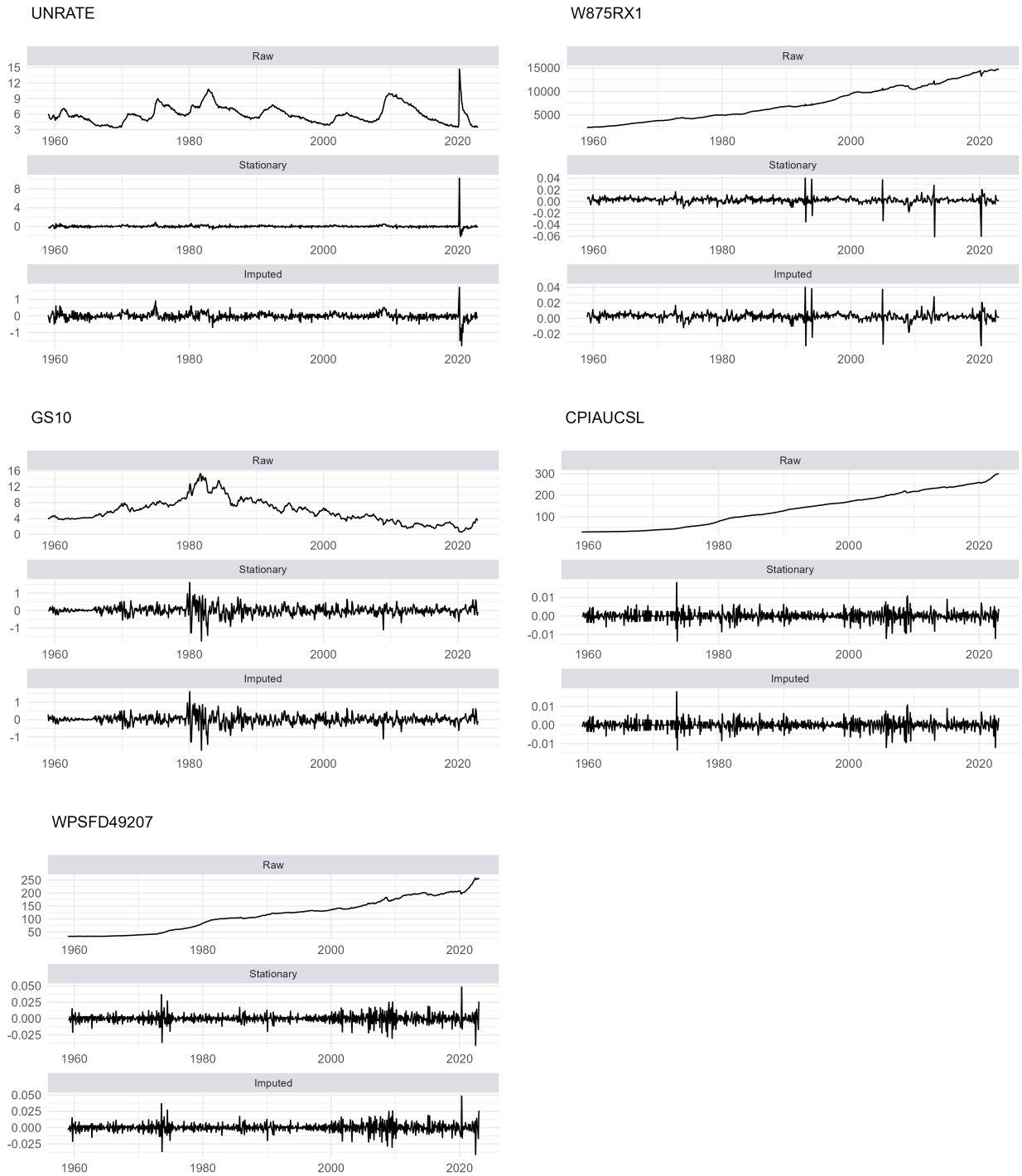
Sims, Christopher A. (1980). "Macroeconomics and Reality". In: *Econometrica* 48.1, pp. 1–48.

Stock, James H. and Mark W. Watson (2002). "Forecasting Using Principal Components from a Large Number of Predictors". In: *Journal of the American Statistical Association* 97.460, pp. 1167–1179.

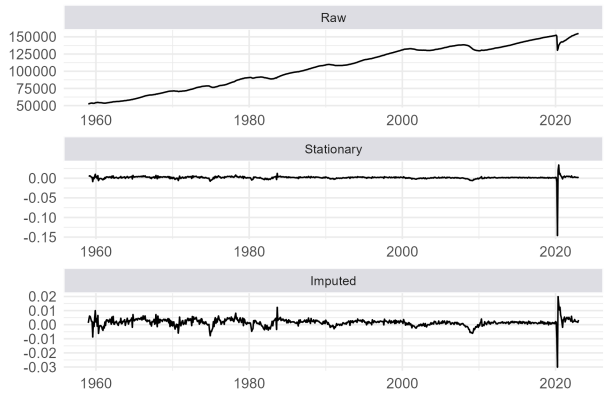
Tenenbaum, Joshua B., Vin de Silva, and John C. Langford (2000). "A global geometric framework for nonlinear dimensionality reduction". In: *Science* 290.5500, pp. 2319–2323.

A.. Policy-relevant variables and preprocessing example

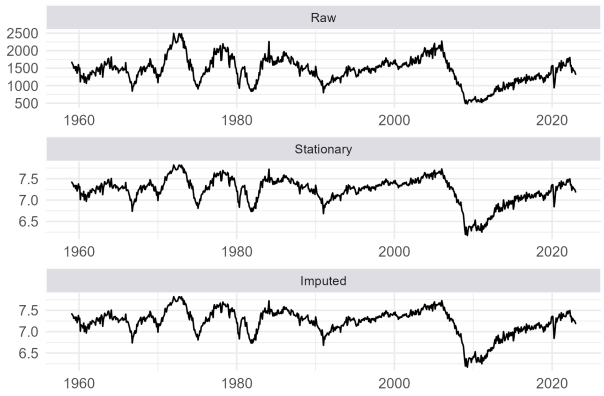
The Figures below reports the policy-relevant series before and after the preprocessing using the whole sample in our dataset.



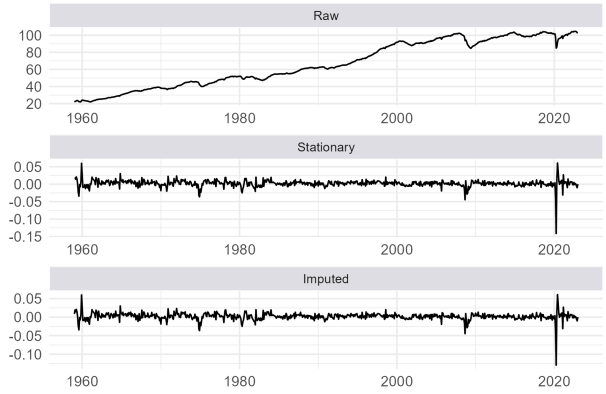
PAYEMS



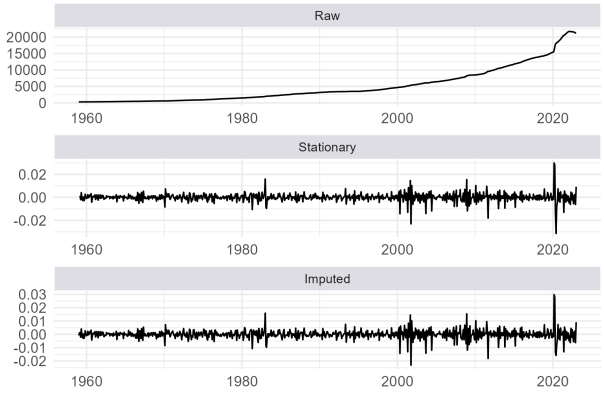
HOUST



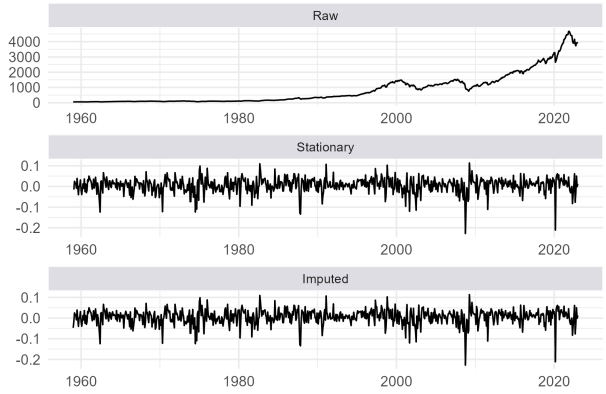
INDPRO



M2SL



S.P.500



B.. Overall computational time

Model	**Overall time**
AR(1)	<1h
PCR	6h
LASSO	40h
LASSO PCR	40h
DFM	6h
Diffusion Map	8h
t-SNE	<1h
LLE	<1h
ISOMAP	2h
Quadratic PCR	<1h
Kernel PCR	<1h
TV-PCR (50 components)	<1h*
TV-PCR (100 components)	<1h*
Ridge TV-PCR	20h*
LASSO TV-PCR	40h*
Adaptive LASSO	50h*
D2FM	12h
Total	**∼300h**

Table B.4: *Approximate overall computational time for the rolling-window evaluation on four horizons and ten target variables. Values marked with an asterik are referred to the execution time if the principal components of the modified matrix for TV-PCR are already available on disk. This operation took another 10 hours of time. Execution times are estimated using 7 cores in parallel execution on a machine equipped with CPU Intel-i7 11370h and GPU NVIDIA GeForce RTX 3050.*