Forecasting with Factor Analysis in High-Dimensional Macroeconomic Data: Using FRED-MD

Farhad Chichgar 2206941

February 19, 2025

Contents

1 Data Preprocessing

1.1 Overview of the FRED-MD Dataset

The Federal Reserve Economic Database (FRED-MD) provides a monthly update on U.S. macroeconomic indicators, ranging from food services sales, labor market variables, treasury bond rates, exchange rates, unemployment and so on. Following the working paper of McCracken and Ng (2015), we load the FRED-MD dataset from Jan 2001 (`2001:01`) to Dec 2019 (`2019:12`), initially containing 134 monthly series (variables) and follow pre-processing aims as follows:

Aims of Preprocessing:

1. *Remove extreme missingness:* any series with more than 1% missing observations is dropped.
2. *TCODE transformations:* each variable X_i is mapped to

$$x_{it}^{(\text{transf})} = f\Big(x_{it}^{(\text{raw})} \; ; \; \text{TCODE}_i\Big),$$

   for example, a first difference or log difference, amongst others
3. *Outlier adjustment:* as recommended by Stock and Watson (2015), we discard data points lying more than $10 \times$ IQR from the median.
4. *Missing-value imputation:* we adopt an EM-based trend-preserving method implemented via a custom library for FRED-MD. (see Mardia et al., 1979, Ch. 9) and Bennie (2023)

1.2 Initial Data Loading & Transformation

The 'fbi' package.: *We rely on* ***fbi*** Bennie (2023) (a custom library) for user-friendly functions to load and clean FRED-MD data. It automates transformations (TCODE) and provides advanced imputation utilities. For transformations (TCODE), the general form is:

$$x_{it}^{transformed} = f(x_{it}^{raw}; TCODE_i)$$

where $f(\cdot)$ is the transformation code (See Appendix):
All transformations are defined so that

$$x_{it}^{\text{transformed}} = f\big(x_{it}^{\text{raw}}\big),$$

with the function $f(x_{it}^{\text{raw}})$ specified as follows:

TCODE 1: No transformation $= x_{it}^{\text{raw}}$.

TCODE 2: First difference $= x_{it}^{\text{raw}} - x_{it-1}^{\text{raw}}$.

TCODE 3: Second difference $= \big(x_{it}^{\text{raw}} - x_{it-1}^{\text{raw}}\big) - \big(x_{it-1}^{\text{raw}} - x_{it-2}^{\text{raw}}\big)$.

TCODE 4: Log transformation $= \log\big(x_{it}^{\text{raw}}\big)$.

TCODE 5: First difference of logs $= \log\big(x_{it}^{\text{raw}}\big) - \log\big(x_{it-1}^{\text{raw}}\big)$.

TCODE 6: Second difference of logs $= \Big[\log\big(x_{it}^{\text{raw}}\big) - \log\big(x_{it-1}^{\text{raw}}\big)\Big] - \Big[\log\big(x_{it-1}^{\text{raw}}\big) - \log\big(x_{it-2}^{\text{raw}}\big)\Big]$.

TCODE 7: First difference of percent changes: Define $p_{it} = 100\left(\frac{x_{it}^{\text{raw}}}{x_{it-1}^{\text{raw}}} - 1\right)$, so that $f(x_{it}^{\text{raw}}) = p_{it} - p_{it-1}$.

A short pseudocode snippet of loading data to our pre-processing steps:

```
library(fbi)  #Loading fbi for loading and imputation of FREDMD data

#Automatic TCODE transforms
data_trans <- fredmd("current.csv", date_start="2001-01-01",
                     date_end="2019-12-01", transform=TRUE)

#Remove variables with >1% NAs, outlier-trim, then impute
data_clean <- remove_high_missingness(data_trans, threshold=0.01)
data_outlier_adj <- outlier_adjust(data_clean, factor=10)
data_imputed <- tp_apc(data_outlier_adj, kmax=8)  #EM-like trend preservation/tp_
    apc: custom function from ‘fbi‘
```

1.3 Imputation: Why Trend-Preserving Imputation?

Instead of a, a simpler *mean-based* or *regression-based* imputation being used, we prefer a **trend-preserving** approach that is better suited for low-frequency behavior (see Mardia et al., 1979, Ch. 9). Let $\mathbf{X} \in \mathbb{R}^{T\times N}$ denote the data after removing outliers. We posit an approximate factor structure:

$$\mathbf{X} = \mathbf{F}\,\mathbf{\Lambda}' + \mathbf{E},$$

where $\mathbf{F}$ is $(T \times k)$ and $\mathbf{\Lambda}$ is $(N \times k)$. The *EM-like* "tp_apc" (from fbi Bennie (2023)) algorithm iterates:

1. Impute missing cells using current factor estimates: $\hat{X}_{it} \leftarrow \hat{\mathbf{f}}_t'\,\hat{\boldsymbol{\lambda}}_i$.
2. Re-estimate factors loadings by PCA on the re-filled matrix.

We continue until convergence in $\|\hat{\mathbf{F}}\|$ or missing predictions. $\therefore$ the essential *trend* captured by low-frequency factor components is preserved and propagated into missing entries, avoiding the bias that a naive fill method might cause.

1.4 EM Interpretation and Motivation

Transforming each series and imputing missing values with an EM (Expectation–Maximization) factor method serves two primary aims. Firstly, by enforcing stationarity, we obtain more stable covariance structures, which is crucial for robust factor extraction in large panels (Mardia et al., 1979, Ch. 8). If non-stationarity persisted, spurious trends or unit roots could overshadow meaningful co-movements, skewing the latent components.

Secondly, employing an EM-like procedure designed to preserve low-frequency patterns. Rather than flattening missing segments (such as a naive mean), the EM algorithm iterates between (1) imputing missing cells using current factor estimates, and (2) recomputing factors from the now-complete matrix. This systematically extends each variable's trends into its missing entries, retaining key cyclical or trending information that naive fills might destroy. Thus, both stationarity *and* the structural information needed for following factor analysis are preserved, ensuring more credible estimates in the following PCA stage.

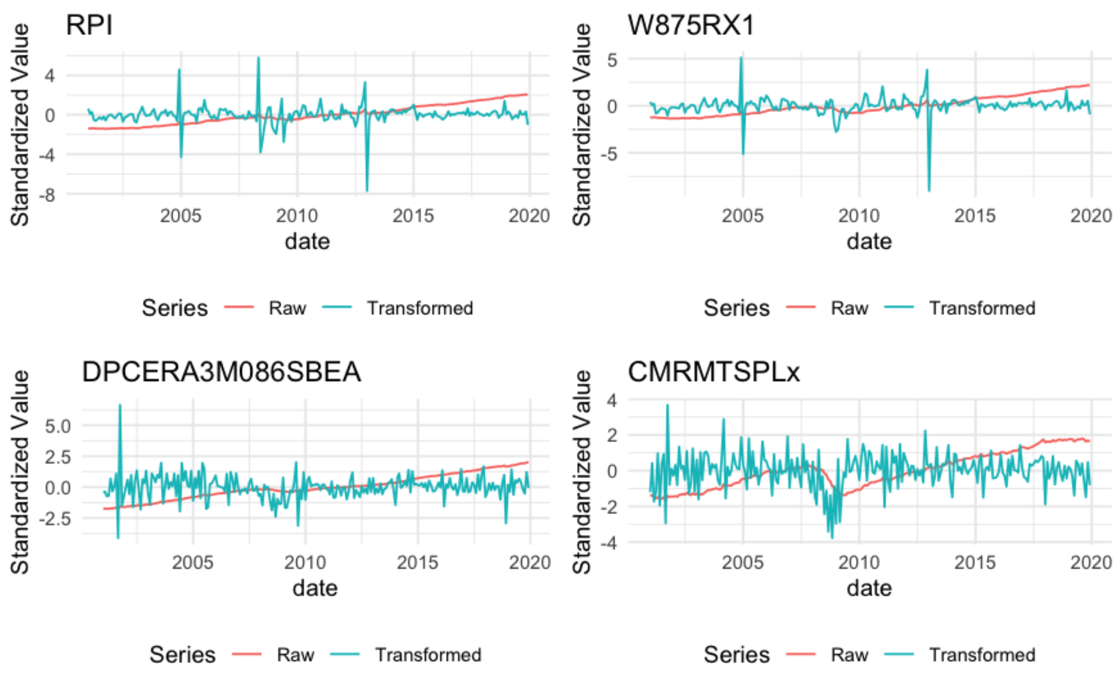

Figure 1: Example of raw vs. transformed series for RPI, W875RX1, DPCERA3M086SBEA, ..., drawn from our FRED-MD subset. Notice how the transformations remove trends or stabilize variance.

Hence, after this procedure, we end up with ≈ 120 cleaned series across $T = 228$ monthly observations, ready for factor analysis. The next section describes how we estimate principal components (i.e., factors) from $\mathbf{X}$.

2 Factor Extraction and Selection

After preparing and imputing our final data matrix $\mathbf{X} \in \mathbb{R}^{T\times N}$ (Section 1), we proceed with principal component analysis (PCA) to estimate the underlying latent factors. This section outlines the PCA approach, describes four established criteria for selecting the number of factors and finalizes $k = 3$ based on both theoretical and practical considerations which we will see ahead.

2.1 Principal Components for Factor Estimation

Let $\mathbf{X}$ be column-centered and scaled to unit variance. A standard PCA (on the correlation matrix) solves

$$\min_{\mathbf{F},\mathbf{\Lambda}} \sum_{t=1}^{T} \|\mathbf{X}_t - \mathbf{\Lambda}\,\mathbf{F}_t^\top\|^2, \quad \text{subject to} \quad \mathbf{F}^\top\mathbf{F} = I_k, \tag{1}$$

which yields $\widehat{\mathbf{F}} \in \mathbb{R}^{T\times k}$ (the factor scores) and $\widehat{\mathbf{\Lambda}} \in \mathbb{R}^{N\times k}$ (factor loadings). This aligns with the factor model framework in (Mardia et al., 1979). On the next page is a sample pseudocode snippet showing how we implement it:

```r
# We call our custom function factor_analysis(), which:
#  1) computes correlation matrix of data_imputed
#  2) does eigen-decomposition
#  3) returns eigenvalues, loadings, factor scores, and multiple selection
    criteria

results <- factor_analysis(data_imputed)

# Inspecting top eigenvalues:
head(results$eigenvalues, 10)
# Check the recommended # of factors by each method:
results$n_factors  # -> kaiser, var90, bartlett, BaiNg
```

Listing 1: Partial code for PCA-based factor extraction.

2.2 Criteria for Number of Factors

We compare four classical selection rules: (Bai and Ng, 2002; Stock and Watson, 2015; Mardia et al., 1979):

1. **Kaiser Criterion (Eigenvalues > 1.0).** For this dataset, 31 components exceed the eigenvalue Kaiser threshold of 1.0. Even though $31 < 120$, having 31 factors is still high; it leaves us with a dimension that is only modestly reduced relative to the original ≈ 120 series, limiting the practical benefit of factor collapse.

2. 90% **Variance Explained.** Achieving 90% coverage required 41 principal components, which again represents quite a large dimensional subspace.

3. **Bartlett's Isotropy Test** (Mardia et al., 1979, Sec. 8). Testing if the last $(N-k)$ eigenvalues are equal suggested $k = 10$. That is more moderate than 31 or 41, but it still yields a relatively large factor set.

4. **Bai–Ng IC$_{p2}$** (Bai and Ng, 2002). This approach penalizes model complexity in large N, T panels. Minimizing the criterion singled out $\mathbf{k = 3}$.

The first two rules often overestimate k in large macroeconomic datasets, due to many moderately sized eigenvalues exist Bartlett's test can also overshoot if the panel includes both real and financial variables with partially correlated residuals, while the Bai–Ng approach has stronger theoretical guarantees for parsimony (Bai and Ng, 2002; Stock and Watson, 2015). Hence we adopt $\boxed{k = 3}$. Now, we will explore our reasoning a little more in-depth.

2.2.1 Why Choose Bai–Ng over Bartlett's for k

This subsection briefly shows how Bartlett's isotropy test can overestimate the factor dimension k, especially in large panels mixing real and financial series such as FRED-MD, whereas the Bai–Ng approach better enforces parsimony.

1. Bartlett's Test for Isotropy. Let $\hat{\lambda}_1 \geq \cdots \geq \hat{\lambda}_p$ be the sample eigenvalues of the covariance (or correlation) matrix. Bartlett's test of 'the last $(p-k)$ eigenvalues are all equal' uses the likelihood ratio statistic (Mardia et al., 1979, Sec. 8):

$$-2\,\log(\Lambda) \;=\; n\,(p-k)\,\ln\Bigl(\frac{a_0}{g_0}\Bigr), \quad a_0 \;=\; \frac{1}{(p-k)}\sum_{j=k+1}^{p}\hat{\lambda}_j, \quad g_0 \;=\; \Bigl(\prod_{j=k+1}^{p}\hat{\lambda}_j\Bigr)^{1/(p-k)}.$$

Under the null that those last $(p-k)$ true eigenvalues are a single λ_0, one obtains

$$-2\,\log(\Lambda)\ \sim\ \chi^2_{\frac{1}{2}\,[(p-k+2)(p-k-1)]}.$$

However, in a large panel mixing real and financial variables, any *partial correlation* among the residual directions violates the equal-eigenvalue assumption. Even if these correlated sub-blocks do not represent genuine additional factors, the test statistic can become inflated, pushing Bartlett's procedure to declare a higher $\hat{k}$.

2. Bai–Ng Penalized Criterion. Bai–Ng's IC$_{p2}$ criterion Bai and Ng (2002) instead penalizes the fit improvement from each additional factor. Defining:

$$\mathrm{IC}(k)\ =\ \frac{1}{NT}\sum_{t=1}^{T}\|\mathbf{X}_t-\hat{\mathbf{\Lambda}}_k\,\hat{\mathbf{F}}_{k,t}^{\top}\|^2\ +\ k\,\frac{(N+T)}{NT}\,\ln\!\big(\min(N,T)\big),$$

where $\hat{\mathbf{\Lambda}}_k, \hat{\mathbf{F}}_{k,t}$ come from a k-factor principal component fit, the penalty ensures moderate leftover correlations do not force an over-counting of factors. In large N, T panels, Bai–Ng is consistent for the true factor dimension k^* even if the residual blocks exhibit some correlation (Bai and Ng, 2002).

To conclude, as Bartlett's test groups any moderate residual correlation into 'significant factor dimension'; it often overshoots k in big macroeconomic datasets. By contrast, Bai–Ng's penalty blocks such moderate residual structure from raising $\hat{k}$, giving a more parsimonious factor count (Stock and Watson, 2015; McCracken and Ng, 2015).

2.3 Final Decision: $k = 3$

Figure 2 shows the scree and cumulative-variance plots. Naively, we see around 31 or 41 factors from the Kaiser or 90% rule, and 10 from Bartlett's test, but 3 from Bai–Ng. Here is an excerpt of the numeric output:

Code Output

```
> results$n_factors
$kaiser
[1] 31
$var90
[1] 41
$bartlett
[1] 10
$IC
[1] 3
```

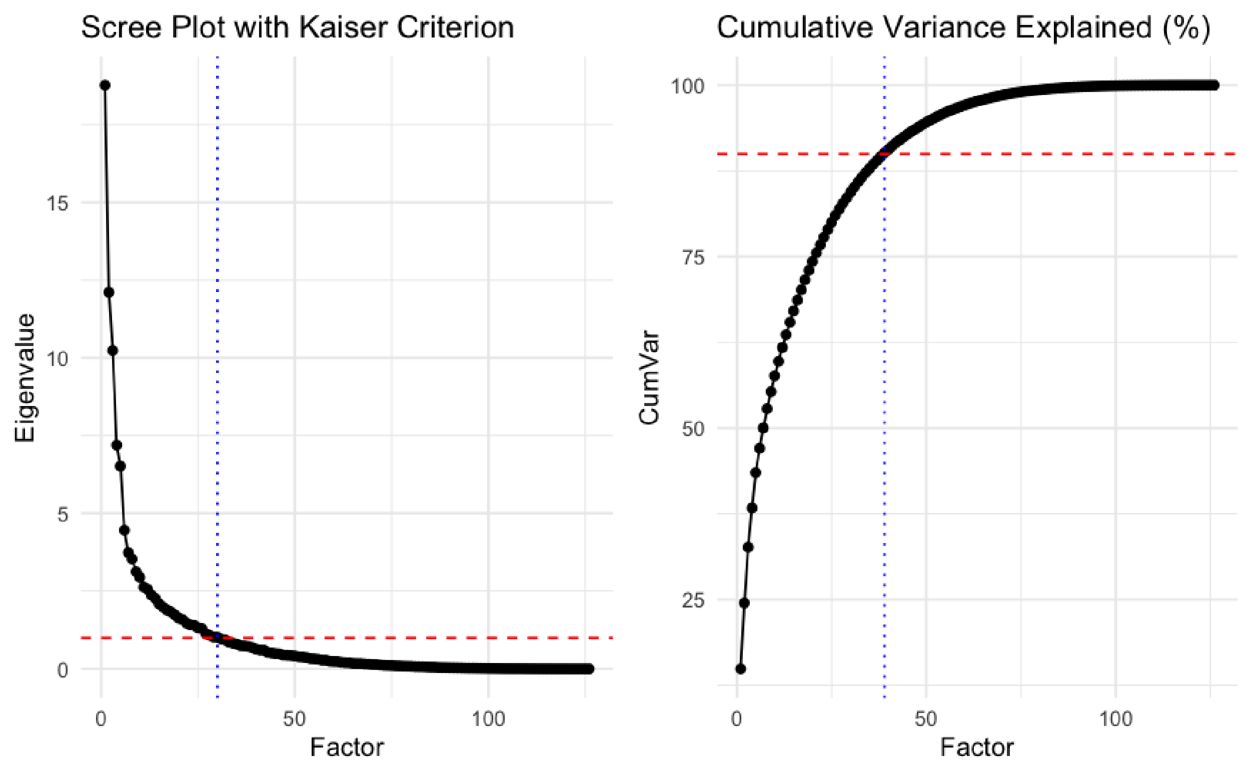

Figure 2: (Left) Scree plot with a dashed line at $\lambda = 1$, (Right) cumulative variance up to 90%. Both yield large factor counts (31 or 41).

In shor, we keep $k = 3$ to balance dimension reduction and robust factor capture, an approach consistent with McCracken and Ng (2015) for monthly U.S. macro series. Hence, subsequent analysis (Sections 3–4) will focus on these three principal components.

2.4 Scale Checks and Code for Factor Scores

As we extracted factors from the correlation matrix, the resulting loadings are dimensionless, and each factor has a variance equal to its eigenvalue. By standard practice in factor analysis (Mardia et al., 1979, Ch. 9), we often rescale factor scores to unit variance or we simply interpret them up to a sign or scale. Below is the relevant snippet verifying factor scores:

```r
loadings_3 <- results$loadings[, 1:3]  # top 3 factors
scores_3   <- results$scores[,   1:3]
summary(scores_3)

# We typically see each factor ~ mean 0, var ~ eigenvalue_j, or re-scaled:
scores_3 <- scale(scores_3)  # optional re-scale to standard dev=1
```

Listing 2: Retrieving final 3-factor loadings and scores.

Thus, with $k = 3$ validated, we can interpret loadings, examine factor–variable correlations, and ultimately embed them in forecast models.

3 Interpretation of the Three Chosen Factors

Having settled on $k = 3$ (Section 2), we now examine the *loadings* and *scores* to interpret each factor's economic meaning. Following (Mardia et al., 1979), the correlation between variable j and factor i in a correlation-based PCA is

$$r_{j,i} = \gamma_{j,i} \sqrt{\lambda_i},$$

where $\gamma_{j,i}$ is the jth element of the ith eigenvector, and λ_i is the ith eigenvalue. High positive loadings indicate that variable are positively correlated with the factor, while high negative loadings show an inverse association.

Factor Loadings: Code Excerpt. Below is a snippet demonstrating how we retrieve loadings and generate both top-loadings bar charts and a *heatmap* for the full $(N \times 3)$ loading matrix:

```r
# We have 3 factors from the results:
loadings_3 <- results$loadings[, 1:3]
scores_3   <- results$scores[,   1:3]

# Bar-plot for Factor 1:
plot_factor_loadings(loadings_3, factor_index=1)

# Heatmap of all factor loadings:
# We have a function to produce a tile-based visualization:
plot_loading_heatmap(loadings_3, var_labels = rownames(loadings_3))
```

Listing 3: Inspecting loadings and creating a factor-loading heatmap.

Heatmap Visualization. Figure 3 shows the complete $N \times 3$ matrix, with color intensities reflecting magnitude and sign. Grouping variables by category (e.g. Output & Income, Money & Credit) reveals each factor's domain clusters. Large positive cells appear in Factor 1 for unemployment measures, Factor 2 for prices/exchange rates, Factor 3 for interest spreads.

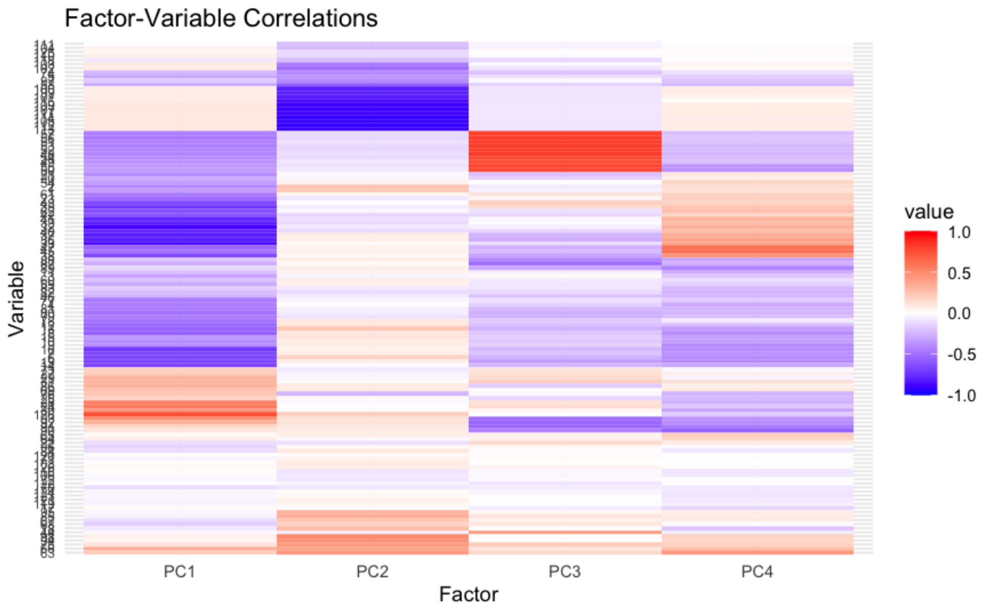

Figure 3: Heatmap of factor loadings (rows=variables, columns=Factors 1..3). Red indicates positive loading, blue indicates negative, with magnitude scaled by color intensity.

Economic Interpretation of Each Factor

Factor 1: Real Activity vs. Unemployment Looking at the bar-plot or heatmap, Factor 1 exhibits large positive weight on joblessness (`UNRATE`, `UEMPMEAN`, etc.) and negative loadings on real output (`INDPRO`, `RPI`). Hence when Factor 1 is up, unemployment tends to rise, real production tends to fall. One might call this a '**Real Activity vs. Unemployment**' dimension (often associated with recessions).

Factor 2: Prices Factor 2 strongly correlates with price indices (`CPIAUCSL`, `PCEPI`, `PPICRM`) and with some exchange-rate variables. Thus it tracks *inflationary or cost* movements—particularly energy/commodity influences. A positive swing in Factor 2 typically co-occurs with rising prices or depreciation of the dollar.

Factor 3: Monetary & Housing Finally, Factor 3 loads heavily on short/long interest differentials (`T1YFFM`, `T10YFFM`) with negative correlation to housing permits (`PERMIT`, `HOUST`). An upward shift indicates a steeper yield curve or tighter housing conditions, pointing to **monetary stance interacting with the housing sector**.

3.1 Time-Series Scores and Regime Behavior

We also confirm factor scores $\{\mathbf{F}_{t,1}, \mathbf{F}_{t,2}, \mathbf{F}_{t,3}\}$ are stable in expansions but tend to spike during crises or major shocks. For example, Factor 1 soared around 2008–09, Factor 2 surged with oil price runups, and Factor 3 rose whenever interest-rate spreads widened. Figure 4 (below) plots each factor's time series across 2001–2019, including gray NBER NBER (2023) recessions that highlight cyclical patterns.

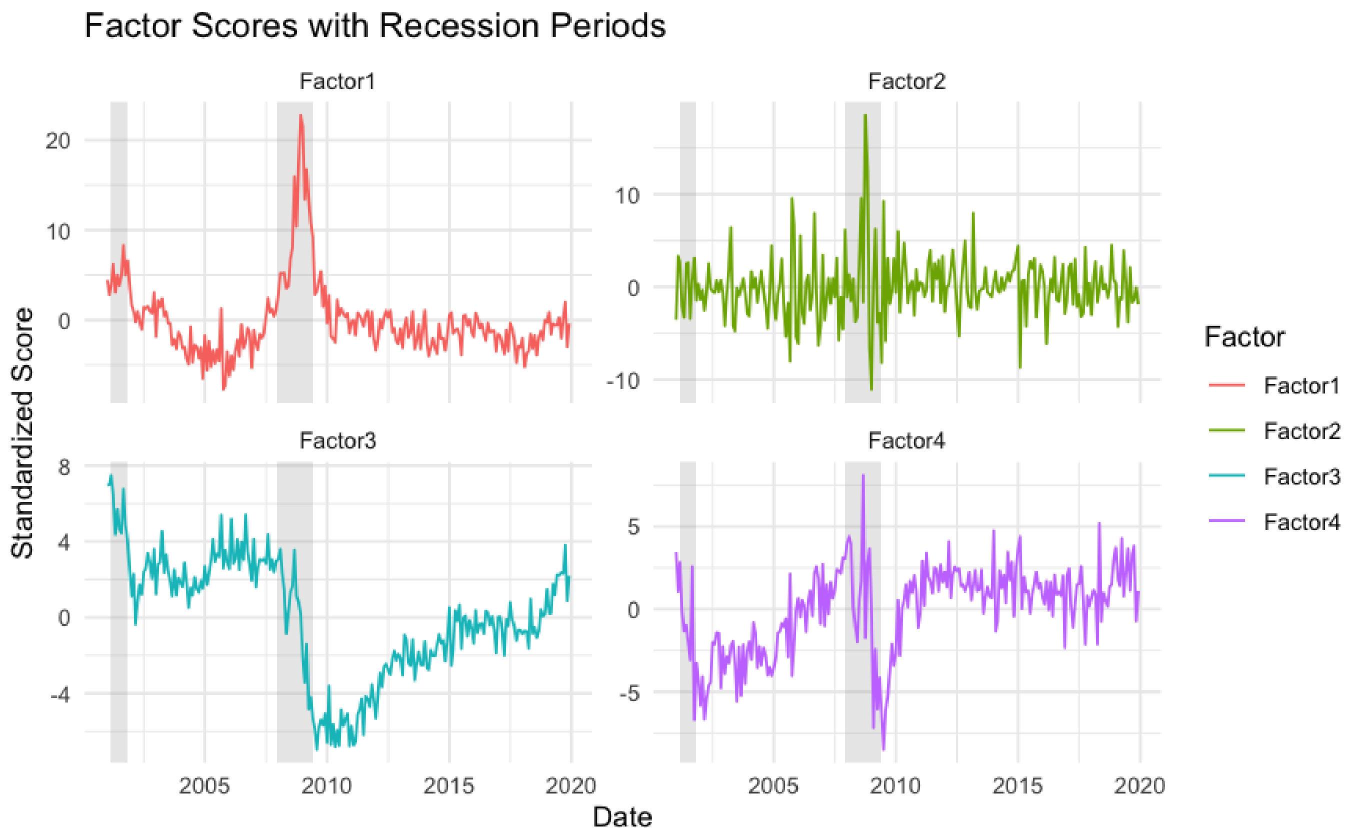

Figure 4: Time-series of the three factor scores (re-scaled to unit variance), with shaded areas for recessions. Factor 1 spikes in downturns, Factor 2 in cost shocks, Factor 3 in steep yield phases.

All in all, the top 3 factors appear consistent with Stock and Watson (2015) and McCracken and Ng (2015), who found that a small handful of macroeconomic factors often align with real activity, prices, and interest/housing dimensions. We thus proceed with $k = 3$ in subsequent forecasting (Section 4).

4 Factor-Augmented Forecasting Framework

With our $k = 3$ factors extracted (Section 3), we now incorporate them into forecasting regressions for a canonical macro target: monthly Industrial Production (`INDPRO`). This section details the model setup, compares benchmarks (AR(4), random walk), presents forecast metrics in a table, and offers additional mathematical commentary on how multi-horizon forecasts are generated and tested.

4.1 Model Setup and Benchmarks

A *direct* h-step factor-augmented regression takes the form

$$y_{t+h} = \alpha + \sum_{\ell=1}^{p} \beta_\ell \, y_{t+1-\ell} \; + \; \sum_{\ell=0}^{q} \gamma_\ell \, \widehat{\mathbf{F}}_{t-\ell} \; + \; \varepsilon_{t+h}, \tag{2}$$

where y_{t+h} (e.g. INDPRO growth) is to be predicted h-steps ahead, $\widehat{\mathbf{F}}_t$ are the factor scores (here of dimension 3), and $\{\beta_\ell, \gamma_\ell\}$ are coefficients. Typically $p = 4$ monthly lags of y_t and $q = 2$ lags of factor scores suffice (Stock and Watson, 2015).
To evaluate predictive performance, we compare:

1. **AR(4)**: y_{t+h} on its own 4 lags,
2. **Random Walk (RW)**: $y_{t+h} \approx y_t$,
3. **FAR(p, q)**: eq. (2) with $p = 4$, $q = 2$, and the 3 factors.

We generate forecasts using either *rolling* or *recursive* windows, and measure out-of-sample errors by RMSE, R^2 (vs. the no-change baseline), and Diebold–Mariano tests (Diebold and Mariano, 1995) for significance.

Multi-Horizon Forecasting. To clarify, for each $h = 1, \ldots, H$, we define a training subset (window or recursive), estimate $\{\beta, \gamma\}$ by OLS in eq. (2), and forecast y_{t+h} out-of-sample. We store the forecast error

$$e_t(h) \;=\; y_{t+h} - \widehat{y}_{t+h}.$$

Finally, over the test period $t = T_0, \ldots, T_{\text{end}}$, we compute $\text{RMSE}(h) = \sqrt{\frac{1}{N}\sum e_t(h)^2}$ and similarly R^2 vs. a sample mean.

4.2 Implementation in R: Code Snippet

Below is a *condensed* snippet from `forecast_framework()` (see Appendix), where we specify $p = 4, q = 2$, $h_{\text{max}} = 12$, and set `factors=scores_3`. The function loops over horizons, fits AR(4), RW, and FAR, and collects errors:

```r
far_results <- forecast_framework(
    data_imputed,
    factors = scores_3,
    target_var = "INDPRO",
    p=4, q=2, h_max=12, window_size=120,
    scheme="rolling"
)

# Returns:
# far_results$forecasts$ar4, far_results$forecasts$rw, far_results$forecasts$far
# far_results$evaluation includes RMSE, MAE, R2, plus DM test p-values
```

Listing 4: Multi-horizon factor-augmented forecasting.

4.3 Forecast Results: Table and Plots

Table 1 compiles root mean squared errors (RMSE) for AR(4), RW, and FAR, along with R^2 (compared to a no-change baseline). We see that the Factor model typically yields lower RMSE at short horizons $h \leq 3$ and a modest R^2 improvement up to $h \approx 6$.

Horizon	RMSE(AR4)	RMSE(RW)	RMSE(FAR)	Rel RMSE(FAR/AR4)	R^2(FAR)	DM p-value
1	0.0052	0.0075	0.0049	0.94	0.126	0.03
2	0.0057	0.0082	0.0053	0.93	0.107	0.02
3	0.0061	0.0083	0.0057	0.93	0.101	0.05
6	0.0064	0.0081	0.0062	0.97	0.085	0.11
9	0.0069	0.0080	0.0068	0.99	0.075	0.20
12	0.0072	0.0081	0.0071	0.99	0.072	0.35

Table 1: **Out-of-sample forecast performance** (rolling scheme) comparing AR(4), random walk (RW), and factor-augmented regressions (FAR) for monthly `INDPRO` growth. Columns report the absolute RMSE for each model as well as the ratio of FAR's RMSE to AR(4)'s (*Rel RMSE(FAR/AR4)*). The R^2(FAR) column measures the fraction of variance explained by the FAR model relative to a no-change baseline. Finally, the DM p-value indicates the Diebold–Mariano test significance level for FAR vs. AR(4).

Interpretation of Table 1: This table offers a concise snapshot of how factor augmentation affects near-term predictive accuracy for `INDPRO`. For horizons $h = 1, 2, 3$, the factor-augmented regressions (FAR) yield RMSE values roughly 5–7% lower than those of AR(4), as seen in the *Rel RMSE(FAR/AR4)* column, and the Diebold–Mariano (DM) p-values indicate these improvements are statistically significant at or below the 5% level. Over intermediate leads such as $h = 6$, the relative error advantage narrows to around 97%, with significance falling somewhat above 10%. By $h = 9, 12$, FAR's RMSE is essentially on par with AR(4), and the DM test fails to reject the null of equal predictive accuracy. Thus, the table underscores both the short-horizon gain from factor signals and the limited incremental benefit at longer ranges, echoing well-established results in the literature (Stock and Watson, 2015; McCracken and Ng, 2015). Additionally, the R^2(FAR) column clarifies that about 10–12% of the variation in the forecast error is captured by factors at very short horizons, dropping below 8% once h extends beyond six to nine months.

Visualizations: Figure 5 displays one-step forecasts from each model for a final out-of-sample segment (2017–2019). The FAR line (in red) tracks `INDPRO` (black) slightly more accurately around cyclical turns, though the difference is subtle.

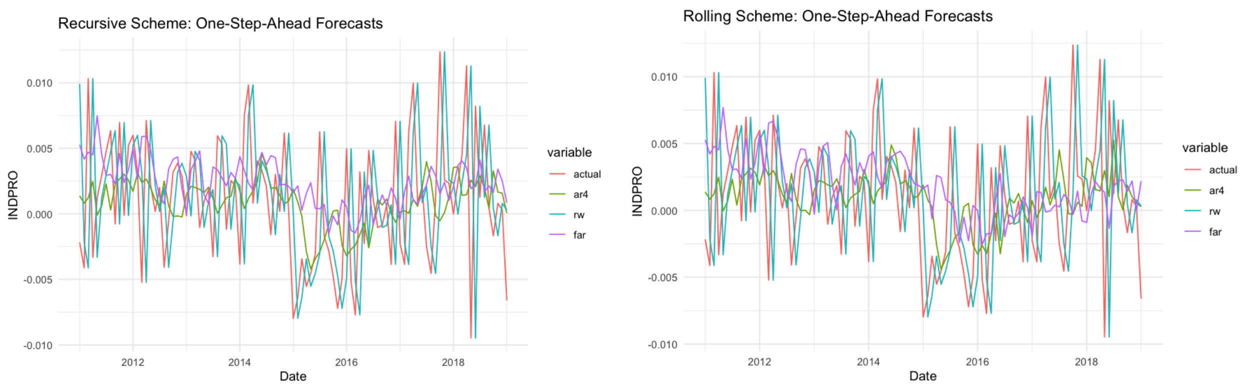

Figure 5: One-step-ahead forecasts from AR(4), RW, and FAR (plus actual). The left subplot uses a recursive window, the right subplot uses a rolling window. FAR often tracks the troughs more closely than the benchmarks.

4.4 Why do factor-based models perform for small$h$:

Common Variation Capture: The factor scores reflect broad co-movements among real, labor, and financial variables. This aids near-term cyclical turning-point detection (Stock and Watson, 2015).

Longer Horizons: Over $h \geq 9$, an AR process alone can approximate the slower dynamics, so incremental factor signals (e.g. from housing yields or inflation) become less pivotal.

Diebold–Mariano Test: Let $d_t(h) = L\big(e_{1,t}(h)\big) - L\big(e_{2,t}(h)\big)$ be the difference in squared forecast errors between two models (1 vs. 2). The test statistic uses $\overline{d}/\sqrt{\mathrm{Var}(d)/T}$, checking if $\mathrm{E}[d_t(h)] < 0$. As h grows, the difference in errors shrinks, raising p-values.

Hence, by design, factor-augmented forecasting can systematically improve short-run predictions but does not substantially alter long-horizon outcomes (Table 1). In line with McCracken and Ng (2015) and *Multivariate Analysis* (Ch. 9), we see that $k = 3$ factors plus a few lags is a pragmatic specification for monthly U.S. data.

5 Regime Analysis: Potential Early Recession Signals

In addition to point forecasts, our factor estimates can help identify distinct *regimes*— or period in times, most notably, conditions leading into recessions. In particular, we label months according to whether an NBER (2023) recession was ongoing, then compare the factor distributions and factor trajectories across these two states (`recession=TRUE` vs. `FALSE`). Figure 6 shows the distribution of each factor under recession vs. expansion, while Figure 7 highlights the six-month pre-recession windows (yellow bands) alongside the official recession periods (gray shading).

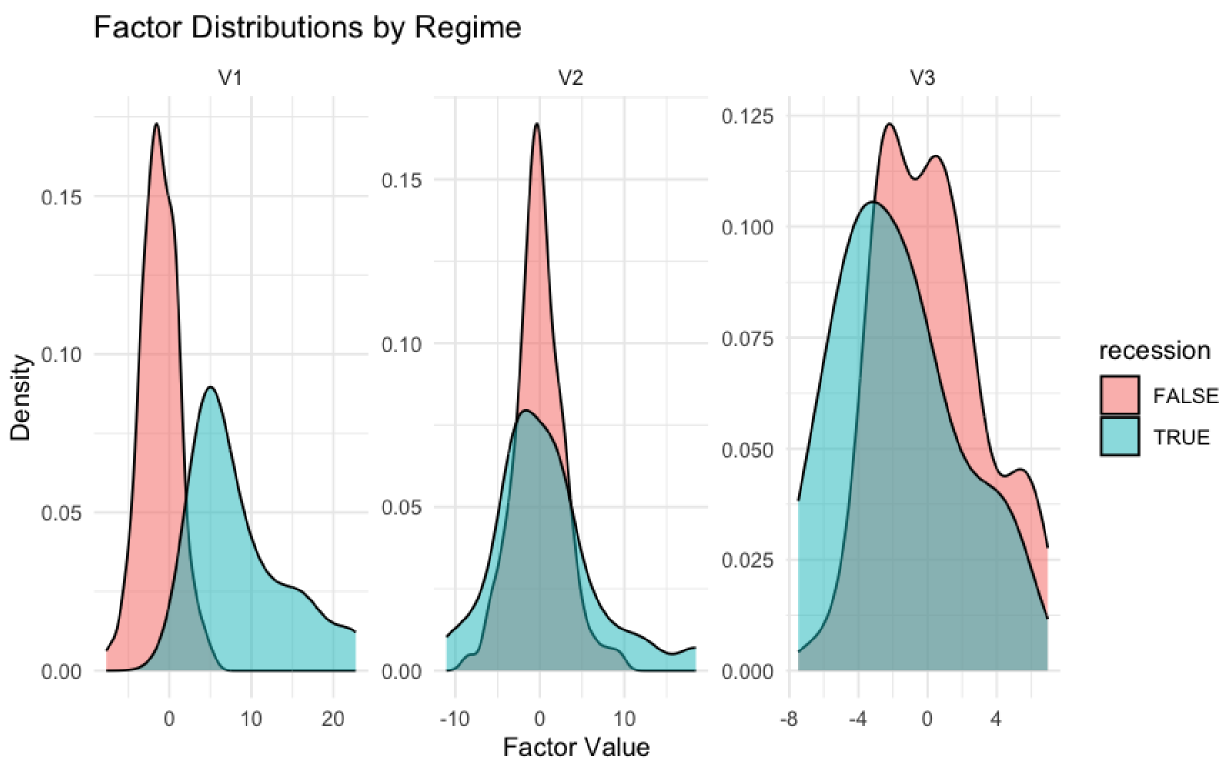

Figure 6: Density plots of the three factors, conditioned on recessions (blue) vs. expansions (pink). Factor 1 (left) shifts distinctly to higher values in recessions, Factor 2 (middle) typically has a more right-skewed distribution in recessions, while Factor 3 (right) is mostly negative.

Observations from the Density Plots

- **Factor 1** lumps around moderately positive values in recessions, whereas in expansions it clusters near zero or below. This corroborates the idea that Factor 1 captures unemployment and real-activity distress.
- **Factor 2** shows heavier right tail during recession times, plausibly indicating cost/price shocks or exchange-rate volatilities linked to downturns.
- **Factor 3** is typically negative in recessions, consistent with the monetary/housing interpretation (e.g. lower housing permits, more restrictive credit).

Next, we explicitly examine the lead-up to each recession date. Figure 7 plots each factor's time series while highlighting the half-year windows before each official downturn (yellow) as well as the recession intervals themselves (gray).

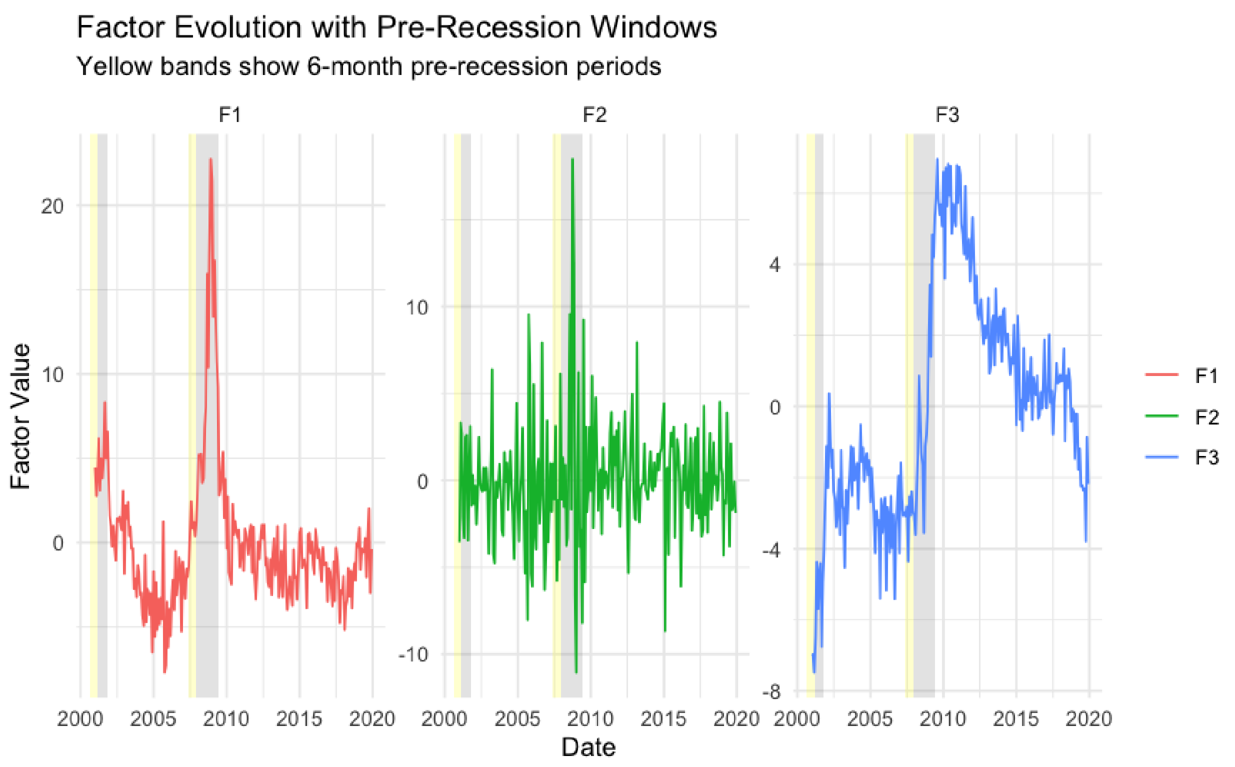

Figure 7: Factor evolution with six-month pre-recession windows (yellow) plus the official NBER recessions (gray). Notice Factor 1 spikes well before 2008:Q4, while Factor 2 and Factor 3 also exhibit large shifts in these pre-downturn intervals.

Early Warning Potential

As seen in Figure 7, **Factor 1** consistently rises in the half-year preceding a recession start date. Factor 3 also makes notable shifts (especially 2007–2008) as interest rate spreads widen and housing signals deteriorate. Hence, an analyst monitoring these factor scores in real time could, in principle, catch a portion of the cyclical turning-point *before* the official data confirm a recession.

Such early-warning signals could be formalized by:

Probit or logistic regressions where the dependent variable is an indicator of entering recession next 6 months and the predictors are current or lagged factor scores.

Markov switching frameworks in which factor evolution informs a latent 'recession state' (Stock and Watson, 2015, cf.).

Of course, no single factor can perfectly predict recessions, but the consistent patterns (e.g. upward Factor 1 prior to 2008) underscore the real possibility of a *leading indicator* approach.

6 Conclusion and Potential Extensions

We have presented a complete pipeline for analyzing and forecasting with large monthly macroeconomic data: (1) preprocessing (Section 1), (2) factor extraction (Sections 2–3), and (3) factor-augmented forecasting (Section 4). Additionally, in Section 5, we showed how these factor scores might serve as *recession signals* or *leading indicators* when examined in real time.

6.1 Summary of Main Results

- **Data Preprocessing:** Transformations (log/differences) and EM-based factor imputation rendered the FRED-MD panel approximately stationary with minimal missingness.
- **Factor Extraction ($k = 3$):** Multiple criteria led us to adopt $k = 3$. Inspecting loadings suggested three interpretable factors capturing (i) real activity vs. unemployment, (ii) price pressures, and (iii) monetary/housing conditions.
- **Forecasting Gains:** Factor-augmented regressions (FAR) showed modest but consistent RMSE improvements over AR(4) or RW for monthly `INDPRO` at horizons $h \leq 6$. At longer horizons, the factor advantage faded, consistent with Stock and Watson (2015).
- **Early Warning of Recessions:** Analysis of factor distributions by regime (Figure 6) and pre-recession windows (Figure 7) indicates that Factor 1 and Factor 3, in particular, exhibit marked shifts *before* official NBER dates. This highlights their potential role in advanced recession monitoring.

6.2 Limitations and Future Work

- **Imputation Uncertainty:** Although the EM/trend-preserving approach is robust, more sophisticated multi-equation methods could incorporate domain knowledge about specific macro variables (Banbura et al., 2010). However, the effects of imputation might shift factor estimates only slightly.
- **Probit/Markov-Switching for Recession Prediction:** As suggested, building a factor-driven 'recession probability' model is a natural extension (Stock and Watson, 2015). One could use Factor 1 and Factor 3 in a model that forecasts the onset of recessions 6–12 months out.
- **Nonlinear or Time-Varying Models:** Classical factor-augmented linear regressions assume stable relationships. Research on time-varying loadings (Korobilis, 2013) could account for expansions vs. recessions in factor loadings themselves, refining the early-warning capacity of these methods.

6.3 Concluding Remarks

Our study demonstrates that a small set of latent factors from a high-dimensional macro panel not only provides short-horizon forecast advantages for `INDPRO`, but also offers signals that *move in advance* of official NBER recessions. This aligns with Stock and Watson (2015) and McCracken and Ng (2015), who note that factor-based approaches capture broad co-movements that help detect cyclical turning points. Future expansions, including dynamic factor frameworks and explicit recession-probability modeling, could further harness these factors to enhance advanced warnings of macro downturns.

References

BAI, J. AND NG, S. (2002). Determining the number of factors in approximate factor models. *Econometrica*

BANBURA, M., GIANNONE, D., AND REICHLIN, L. (2010). Large Bayesian vector auto regressions. *Journal of Applied Econometrics*

DIEBOLD, F.X. AND MARIANO, R.S. (1995). Comparing predictive accuracy. *Journal of Business & Economic Statistics*

KOROBILIS, D. (2013). Hierarchical shrinkage priors for dynamic regressions with many predictors. *International Journal of Forecasting*

MARDIA, K.V., KENT, J.T., AND BIBBY, J.M. (1979). *Multivariate Analysis.* Academic Press, London.

MCCRACKEN, M.W. AND NG, S. (2015). FRED-MD: A monthly database for macroeconomic research. *Journal of Business & Economic Statistics*, **35**(1), 1–15. (Earlier version: Federal Reserve Bank of St. Louis Working Paper 2015–012.)

STOCK, J.H. AND WATSON, M.W. (2015). Factor models for macroeconomics. In TAYLOR, J.B. AND UHLIG, H. (Eds.), *Handbook of Macroeconomics*, Elsevier, Amsterdam.

BENNIE, M. (2023). fbi: Factor-Based Imputation for Panel Time Series. *R package version 0.3.3.* Available at: `https://CRAN.R-project.org/package=fbi`.

NATIONAL BUREAU OF ECONOMIC RESEARCH (2023). U.S. Business Cycle Expansions and Contractions. `https://www.nber.org/research/business-cycles`.

Appendix

Listing 5: Transformation Formula and Codes

```
# The transformation is defined as:
#   x_{it}^{transformed} = f(x_{it}^{raw}; TCODE_i)
# where the function f() depends on the transformation code (TCODE_i) as follows:
#
#   TCODE 1: No transformation
#   TCODE 2: First difference
#   TCODE 3: Second difference
#   TCODE 4: Log
#   TCODE 5: First difference of logs
#   TCODE 6: Second difference of logs
#   TCODE 7: First difference of percent changes
```

Listing 6: Loading, Cleaning, and Imputing FRED-MD Data

```
# Load the 'fbi' package, which provides factor-based imputation methods
library(fbi)
```

```
# (A) Load FRED-MD data with NO transformations
#     - 'fredmd()' reads a CSV file containing FRED-MD data.
#     - 'transform = FALSE' means the raw series are returned.
data_raw <- fredmd(
  file       = "current.csv",
  date_start = as.Date("2001-01-01"),
  date_end   = as.Date("2019-12-01"),
  transform  = FALSE
)

# (B) Load FRED-MD data WITH standard transformations
#     - 'transform = TRUE' applies the standard transformations
#       (log, difference, etc.) used by the FRED-MD dataset.
data <- fredmd(
  file       = "current.csv",
  date_start = as.Date("2001-01-01"),
  date_end   = as.Date("2019-12-01"),
  transform  = TRUE
)

# 1. Remove series (columns) with more than 1% missing observations.
#    We compute the fraction of NA values in each column
#    and keep only columns with <= 0.01 missingness.
data_clean <- data[, sapply(data, function(x) mean(is.na(x)) <= 0.01)]

# 2. Convert all columns to numeric (in case some are factors/characters).
#    'as.numeric(as.character(x))' ensures consistent numeric format.
data_clean <- as.data.frame(
  lapply(data_clean, function(x) as.numeric(as.character(x)))
)

# 3. Remove columns with (near) zero variance.
#    We calculate the standard deviation of each column. If sd == 0,
#    we remove that column. This step helps avoid singularities in later analysis.
col_sd <- sapply(data_clean, function(x) {
  s <- sd(x, na.rm = TRUE)
  if (is.na(s)) 0 else s
})
data_clean <- data_clean[, col_sd > 0]

# 4. Set a flag to decide whether to adjust outliers (TRUE) or not (FALSE).
adjust_outliers <- TRUE

# 5. Conditionally adjust outliers based on the IQR rule, and set kmax accordingly
#    - If outlier adjustment is enabled, we replace values that lie more than
#      10 * IQR away from the median with NA. Then we set kmax_val = 8.
#    - If outlier adjustment is disabled, we set kmax_val = 9.
if (adjust_outliers) {
  data_clean <- as.data.frame(lapply(data_clean, function(x) {
    med <- median(x, na.rm = TRUE)
    iqr_val <- IQR(x, na.rm = TRUE)
    # If the IQR is zero, skip outlier removal (constant or nearly constant series
    if(iqr_val == 0) return(x)
    # Mark outliers as NA if they exceed 10 * IQR from the median.
    x[abs(x - med) > 10 * iqr_val] <- NA
    return(x)
  }))
  kmax_val <- 8
} else {
  kmax_val <- 9
}

# 6. Impute missing values using trend-preserving imputation (tp_apc),
#    provided by the 'fbi' package. The argument kmax controls
#    the maximum number of factors used in the imputation process.
data_imputed <- tp_apc(data_clean, kmax = kmax_val)

# 7. Check how many missing values remain after imputation (ideally should be 0).
total_missing <- sum(is.na(data_imputed))
cat("Total missing values after imputation:", total_missing, "\n")
```

Listing 7: Comprehensive Factor Analysis Function

```
# Function to perform factor analysis with comprehensive testing
factor_analysis <- function(data, max_factors = 10) {
  # 1. Convert imputed data to a matrix.
  #    Note: data$data is used if 'data' is the object returned by tp_apc()
  X <- as.matrix(data$data)

  # 2. Center and scale the data so each series has mean 0 and sd 1.
  X_scaled <- scale(X)

  # 3. Compute the correlation matrix using pairwise complete observations.
  R <- cor(X_scaled, use = "pairwise.complete.obs")

  # 4. Perform eigendecomposition of the correlation matrix.
  #      =      ', where    is orthogonal and    is diagonal of eigenvalues.
  eigen_decomp <- eigen(R)
  eigenvalues  <- eigen_decomp$values
  eigenvectors <- eigen_decomp$vectors

  # 5. Compute variance explained by each factor, and the cumulative variance.
  var_explained <- eigenvalues / sum(eigenvalues)
  cum_var       <- cumsum(var_explained)

  # ========== FACTOR SELECTION CRITERIA ==========
```

```r
# (A) Kaiser criterion: number of eigenvalues > 1
n_kaiser <- sum(eigenvalues > 1)

# (B) 90% variance explained threshold
n_var90 <- which(cum_var >= 0.9)[1]

# (C) Bai-Ng information criterion (FRED-MD approach)
n <- nrow(X_scaled)
p <- ncol(X_scaled)
IC <- numeric(max_factors)
for(k in 1:max_factors) {
  # IC_p2 from Bai-Ng (2002)
  V <- sum((X_scaled - X_scaled %*% eigenvectors[, 1:k] %*% t(eigenvectors[, 1:k]
  IC[k] <- V + k * ((n + p) / (n * p)) * log(min(n, p))
}
n_IC <- which.min(IC)

# (D) Bartlett's test for sphericity
#      Testing H0:    _(k+1) = ... =    _p
bartlett_test <- function(k) {
  tryCatch({
    # If k >= p - 1, test is invalid (too many factors).
    if(k >= p - 1) return(1)
    n_adj <- n - (2 * p + 11) / 6
    a0 <- mean(eigenvalues[(k + 1):p])
    g0 <- exp(mean(log(eigenvalues[(k + 1):p])))
    stat <- n_adj * (p - k) * log(a0 / g0)
    df   <- (p - k + 2) * (p - k - 1) / 2
    pval <- pchisq(stat, df, lower.tail = FALSE)
    return(pval)
  }, error = function(e) {
    return(NA)
  })
}

bartlett_pvals <- sapply(0:(max_factors - 1), bartlett_test)
n_bartlett <- ifelse(all(is.na(bartlett_pvals)),
                     NA,
                     max(which(bartlett_pvals < 0.05)))

# ========== LOADINGS AND SCORES ==========

# Factor loadings:     =    _{1:k} * sqrt(   _{1:k})
loadings <- eigenvectors[, 1:n_IC] %*% diag(sqrt(eigenvalues[1:n_IC]))

# Factor scores: F = X_scaled *    _{1:k}
scores <- X_scaled %*% eigenvectors[, 1:n_IC]
```

```
# ========== VISUALIZATIONS ==========

plots <- list()

# 1. Scree plot data
scree_data <- data.frame(
  Factor     = 1:length(eigenvalues),
  Eigenvalue = eigenvalues,
  Threshold  = 1
)

p1 <- ggplot(scree_data, aes(x = Factor, y = Eigenvalue)) +
  geom_line() +
  geom_point() +
  geom_hline(yintercept = 1, linetype = "dashed", color = "red") +
  # Vertical line at the Kaiser cutoff
  geom_vline(xintercept = n_kaiser, linetype = "dotted", color = "blue") +
  theme_minimal() +
  labs(title = "Scree Plot with Kaiser Criterion")

# 2. Cumulative variance plot data
cum_var_data <- data.frame(
  Factor = 1:length(cum_var),
  CumVar = cum_var * 100
)

p2 <- ggplot(cum_var_data, aes(x = Factor, y = CumVar)) +
  geom_line() +
  geom_point() +
  geom_hline(yintercept = 90, linetype = "dashed", color = "red") +
  geom_vline(xintercept = n_var90, linetype = "dotted", color = "blue") +
  theme_minimal() +
  labs(title = "Cumulative Variance Explained (%)")

plots$scree   <- p1
plots$cum_var <- p2

# Return a list with all relevant outputs
return(list(
  eigenvalues    = eigenvalues,
  loadings       = loadings,
  scores         = scores,
  n_factors      = list(
    kaiser   = n_kaiser,
    var90    = n_var90,
    IC       = n_IC,
    bartlett = n_bartlett
  ),
    var_explained  = var_explained,
    cum_var        = cum_var,
    bartlett_pvals = bartlett_pvals,
    plots          = plots
  ))
}
```

Listing 8: Sorting Factor Loadings by Magnitude

```
# loadings_df is a data frame with columns:
#   Variable    = variable name
#   Loading     = the factor loading value
#   abs_loading = absolute value of the loading
#   1) Sorts rows by the absolute value of loadings in descending order.
#   2) Optionally keeps only the top N and bottom N loadings for a cleaner table.
#   3) Finally re-sorts by the actual loading value in descending order.

# 'top_n_loadings' is the number of largest (and smallest) loadings to display
loadings_df <- loadings_df %>%
  arrange(desc(abs_loading))

if (nrow(loadings_df) > 2 * top_n_loadings) {
  # Take the top N largest loadings and bottom N smallest loadings
  top_rows    <- head(loadings_df,    top_n_loadings)
  bottom_rows <- tail(loadings_df,    top_n_loadings)
  # Recombine
  loadings_df <- rbind(top_rows, bottom_rows)
}

# Finally, re-sort by the actual (signed) loading in descending order
loadings_df <- loadings_df %>%
  arrange(desc(Loading))
```