

# **covbayesvar**: A Python Package for Large Bayesian VAR Model with Covid Volatility

**Sudiksha Joshi**  
University of Washington

---

## Abstract

This paper introduces **covbayesvar**, a Python package for estimating large Bayesian Vector Autoregression (BVAR) models, that account for COVID-induced volatility. The package enables us to estimate the model with hierarchical prior selection when the parameters proliferate, accounting for structural shifts in macroeconomic and financial data during the pandemic. Incorporating various priors, it is versatile enough to answer wide-ranging policy-related questions. With detailed programming examples, I explain how to apply the functions on macro and financial data to construct unconditional and conditional forecasts, scenario analysis, assess joint predictive densities of variables, and how we can employ entropic tilting to modify the forecast distribution and construct forecasts conditional on long-term targets set by the Federal Reserve Bank. Accessible via PyPI, the package includes extensive documentation and code examples. **covbayesvar** advances the state-of-the-art in open-source econometric software, offering researchers a robust tool for analyzing large-scale economic systems under unprecedented uncertainty.

*Keywords:* Bayesian statistics, vector autoregressions, time series, macroeconomics, hierarchical model, forecasts, Minnesota prior, Inverse-Wishart prior, scenario analysis, entropic tilting, FRED-MD, conditional probability, joint predictive density

---

## 1. Introduction

Macroeconometric forecasting often entails smaller time series of historical data, particularly, if we forecast using frequentist vector autoregressions (VARs) at the monthly frequency. Central banks and financial institutions employ vector autoregressions to forecast future economic conditions that assess the impact of monetary policy transmission mechanisms to understand how policy instruments such as interest rates affect the dynamic relationship between inflation, output, and unemployment. Among these are examining how fiscal stimulus or tax changes affect debt burden, can the US economy lands softly, and how high should the mortgage rates be to slow down the real estate market. This is critical in making decisions that have far-reaching consequences for the economy. Regulators and banks use these models to simulate adverse economic scenarios and analyze how banks will perform under stress such as severe recessions, rising defaults on loans, and diminished cash reserves.

Albeit these flexible time series models capture convoluted dynamic relationships among macro and financial variables, they become densely parametrized, yielding imprecise out-of-sample forecasts. Creating a problem of the

“curse of dimensionality” where the number of observations exceeds the number of parameters to estimate, statisticians and economists have introduced Bayesian methods to address the problems of unstable and burgeoning parameters. Amongst those are BVARs with informative priors that shrink the overly parameterized models towards parsimonious models with sparse coefficients, developed by Giannone Lenza and Primiceri (2015). However, this methodology isn’t necessarily most suitable to estimate a model on the pandemic-driven extreme observations, when key macro variables such as unemployment rate, oil prices, industrial production, and retail sales vary substantially. For instance, WTI oil prices nosedived to -53 percent in March 2020 compared to one year ago, then skyrocketed to 272 percent in April 2021 when measured in year-over-year growth rates. Such extreme outliers warrant changing the estimation strategy to account for wider uncertainty when acutely large shocks propagate through the economy and skew the distribution of the time series data.

Owing to the contaminated data from the COVID outbreak, I forecast and estimate the responses to shocks when extreme observations such as those seen during the pandemic, starting from March 20, 2020, are present. Recognizing that the COVID-19 pandemic altered the statistical properties of a plethora of economic and financial variables, such as the moments, and autocorrelation, I employ a Bayesian inferential approach to combine standard prior beliefs with the information present in the data. Currently, there is no equivalent to the MATLAB estimation of BVAR models in Python that incorporates covid volatility in big data. Therefore, in Python, I forecast and estimate the responses to shocks when extreme observations such as those seen during the pandemic, starting from March 20, 2020, are present. Incorporating the Bayesian VAR developed by Primeceri and Lenza (2021), I first fit a Bayesian VAR model with covid volatility on medium-sized macro and financial variables data with the model specifications from Primiceri and Lenza (2022). Unlike their model with seven variables, I added numerous financial variables in a model typically used to analyze macro variables, extending the model to 28 variables. Before estimating the model, I translated the MATLAB codes, where macroeconomists write codes for forecasting exercises, to Python, and built a Python package called [covbayesvar](#). Users can install this package as it is available in the Python Package Index (PyPI). It implements the modeling approach to prior selection and conditional forecasting as done in Giannone et. al (2015), Lenza and Primiceri (2022), and Crump et. al (2019). This package encompasses all the functions necessary to transform the data, estimate the model for inferences, calculate reduced-form impulse responses, construct unconditional (baseline) forecasts based on the historical correlations between variables, forecasts conditional on shocks on certain variables for given time horizons, plot scenario analysis of those shocks, plot joint distribution of forecasts, and employ entropic tilting to make projections conditional on long run targets set by the Federal Reserve Bank. To illustrate the usage of these functions, I’ve created and shared [Google Colab notebooks](#) that showcase how to construct the model using macro and financial data and make forecasts. Furthermore, a separate documentation of all the functions with simple examples is present in html format, and Github.

Next, I briefly introduce the BVAR model mathematically in Section 2 and elucidate the main Python function that implements the model in Section 3. After providing the overview of the important estimation and forecasting functions, using a simple Monte Carlo simulation of the constant volatility BVAR model, in section 5, I evaluate how the simulated posterior estimates of the parameters fare relative to the true hypothesized values. Subsequently, I analyze using real macroeconomic and financial data, presenting the variables in the model, followed by forecasting results, a scenario analysis exercise, and a routine to condition the forecasts on Fed’s long-run targets using entropic tilting in section 6.

## 2. Model

Lenza and Primiceri's (2021) method explicitly models the changing volatilities in shocks and incorporates significant financial and macroeconomic innovations during the pandemic. Unlike other models that explain volatilities such as TVP-VAR, we know when the shock hit the economy, i.e., the variance of the innovations is known. So, I estimate the model and measure changes in the volatility, keeping in mind that the volatility can persist for several periods in the future. As the first extreme observation was in March 2020, I rescale the standard deviation of the shocks in March, April, and May to unknown parameters  $\eta_1, \eta_2, \eta_3$ , and estimate these parameters using Bayesian methods explained in Giannone et al. (2015).

Then, I estimate the variance of the residuals after March 2020, assuming that the volatility decays at a constant rate every month henceforth. The main model is a VAR(12):

$$y_t = \alpha + A_1 y_{t-1} + \cdots + A_{13} y_{t-13} + \eta_t \epsilon_t, \quad \epsilon_t \sim N(0, \Sigma)$$

where  $\eta_t = \eta_1$  before the pandemic began, and scales up the residual covariance matrix during the pandemic at time  $t^*$ .

$$\eta_t = \begin{cases} \eta_1 & t = t^* \\ \eta_2 & t = t^* + 1 \\ \eta_3 & t = t^* + 2 \\ \eta_{t+j} = 1 + (\eta_3 - 1)n_4^{-j/2} & \text{otherwise.} \end{cases}$$

The scaling factors take distinct values in the initial three periods after the pandemic began, before decaying at the rate  $n_4$  henceforth. Since more lags proliferate the number of parameters in the model, we cannot feasibly estimate the model by OLS, especially when the “curse of dimensionality” precludes estimation in a richly parameterized model. Thus, I shrink the autoregressive parameter coefficients using the Minnesota prior. This prior presumes that the variables follow a random walk with drift, and shrink the coefficients of faraway lags. I also impose the natural conjugate prior, wherein I draw the elements of the variance-covariance matrix  $\Sigma$  from the inverse-Wishart distribution. Then, I obtain the posterior estimates of the parameters, fit generalized impulse responses, and forecast. More specifically, to estimate equation (1), if we assume that the scaling factor  $\eta_t$  is known, then we can rewrite equation (1) where I stack coefficients as:

$$y_t = X_t \Phi + \eta_t \epsilon_t,$$

where  $X_t = I_k \otimes [1, y'_{t-1}, y'_{t-2}, \dots, y'_{t-13}]$  and  $\Phi = \text{vec}([\alpha, A_1, A_2, \dots, A_{13}])$ .

Normalizing the coefficients to account for the scaled idiosyncratic error, I divide equation (2) by  $f_t$  as:

$$\frac{y_t}{\eta_t} = \frac{X_t}{\eta_t} \Phi + \epsilon_t \quad \Rightarrow \quad \tilde{y}_t = \tilde{X}_t \Phi + \epsilon_t.$$

Using the transformed data to estimate the parameters  $\tilde{y}_t$  and  $\tilde{X}_t$ , firstly, the prior distribution for the VAR error variance-covariance matrix  $\Sigma$  is the normal inverse-Wishart with the scale  $S$  and degrees of freedom  $n+2$ . In other words,  $\Sigma \sim IW(S, n+2)$ .

Secondly, the conditional distribution of the coefficient matrix  $\Phi$  given the estimates of the variance-covariance matrix  $\Sigma$  is:

$$\Phi|\Sigma \sim N(\psi, \Sigma \otimes \Omega).$$

Thirdly, I characterize extreme values via the Pareto distribution, where the prior belief on all the scaling factors is the Pareto distribution with the unit values as the scale and shape parameters. To corroborate, the shape parameter connotes the speed at which the tail falls off, and the scale parameter regulates the threshold above which the distribution becomes conspicuously relevant. The unit values indicate that the Pareto distribution has a very long and flat tail, attributing that most of the probability mass is concentrated in lower values and the higher values occasionally occur. This is compatible with the belief that the variance of the errors is large during the pandemic:

$$\eta_t \sim \text{Pareto}(1, 1), \quad t = 0, 1, 2.$$

Fourthly, I impose a single unit root prior to model the Bernoulli probability  $p$  that a series is non-stationary. Additionally, I set the prior beliefs to establish the uncertainty around the Bernoulli probability  $p$  using the Beta distribution. The model and standard deviation are the hyperparameters of the Beta distribution that affect the shape of the distribution. Relatedly, the prior distribution on the rate at which the scaling factor decays  $\beta$  is a Beta distribution with a large mode of 0.8, and a standard deviation of 0.2. This implies the probability density function peaks at 0.8, with a dispersion of 0.2.

Finally, I employ another shrinkage prior, known as the sum of coefficients prior, to shrink the coefficients towards zero to preclude an overfit model. Conjecturing that a no-change forecast is a good forecast when the sample begins, it tapers the importance of deterministic components in a VAR estimated condition on initial observations. The deterministic element is:

$$E[y_t | Y_1, Y_2, \dots, Y_{12}, \Phi].$$

Formulated using the Theil Mixed Estimation, it forms conjugate priors by constructing artificial data and appending them when the actual sample ends. Centering the prior belief that all the coefficients on their own lags sum to 1 in each equation:

$$(A_1 + A_2 + \dots + A_{12}) |\Sigma \sim N(I_k, V(\omega, \tilde{y}_0, \Sigma)),$$

where  $\omega$  is the hyperparameter that regulates how dispersed the prior beliefs are. As  $\omega \rightarrow \infty$ , the prior is flat, and as  $\omega \rightarrow 0$ , the model has a unit root in each equation, ruling out cointegrated variables. It creates a sparse model as it restrains higher probability mass on smaller values of the coefficients, curtailing the number of parameters in the model. Whereas the Minnesota prior allows us to identify the important variables in the model, the sum of the coefficients prior identifies few important coefficients. Therefore, utilizing both the sum of coefficients and Minnesota priors in the same model additionally regularizes the model and allows the different types of priors to capture different features of the data. To estimate the posterior density of the parameters, I adopt a hierarchical technique of sampling using the Metropolis-Hastings algorithm defined by Primiceri and Lenza (2022). Briefly, first, I initialize the hyperparameters at their posterior mode; and draw the candidate values of these hyperparameters from the proposal distribution. Pre-specifying the acceptance rate at roughly 25 percent, I accept and reject

the proposed draws of hyperparameters; then draw the coefficients  $\Phi$  and variance-covariance matrix  $\Sigma$  from the normal-inverse Wishart density function.

### 3. Python Implementation: Main Functions

This section sheds light on four main functions in the `covbayesvar` package that are instrumental in defining the prior distributions, estimating the model to update the posterior distributions, and estimating and plotting the unconditional and conditional forecasts.

#### Priors and Hyperparameters

We assume external beliefs via additional inputs in the `set_priors_covid` function. Here, the priors enable us to regularize or shrink the parameter space that prevents overfitting when data is insufficient to estimate the parameters reliably, and the economy experiences sudden structural shifts.

The function returns a tuple with several dictionaries and lists, where each argument represents:

##### 1. **r (Priors):**

A dictionary containing the priors for the BVAR model. This includes settings like `hyperpriors`, `Vc`, `pos`, and the hyperparameters for the Minnesota prior `MAlpha`, the time horizon for forecasts `hz`, and MCMC-related parameters such as the total number of draws in the MCMC simulation `Ndraws` and the initial number of draws burnt-in `Ndrawsdiscard`.

##### 2. **mode (Hyperpriors' Mode Values):**

A dictionary containing the mode values for hyperpriors, such as:

- `lambda`: The tightness of the prior for coefficients on own lags.
- `miu`: A prior for the sum of coefficients.
- `theta`: Controls the overall shrinkage.
- `eta`: Hyperparameters related to volatility changes due to the onset of COVID-19.

##### 3. **sd (Standard Deviations for Hyperpriors):**

This dictionary contains the standard deviations for the hyperpriors `lambda`, `miu`, `theta`, `eta`, reflecting the uncertainty around their respective mode values.

##### 4. **priorcoef (Priors' Coefficients):**

A dictionary of coefficients for the hyperpriors, calculated using functions like `gamma_coef()` for gamma priors and `beta_coef()` for beta priors. These coefficients are necessary for estimating the posterior distribution in the BVAR model.

##### 5. **MIN and MAX (Bounds for Optimization):**

These dictionaries set lower and upper bounds for variables in the maximization process. For example, `MIN['lambda'] = 0.0001` ensures that the prior's tightness cannot be smaller than 0.0001, and `MAX['lambda'] = 5` prevents it from exceeding 5.

##### 6. **albet and mosd (Beta Distribution Parameters):**

These are parameters related to the Beta distribution that affect volatility `eta`. The `alpha` and `beta` parameters describe the distribution of hyperparameters that govern the dynamics during COVID-19.

## Estimating the BVAR

The function `bvarGLP_covid` estimates the Bayesian VAR model after considering changes in volatility due to the COVID-19 pandemic. Originally founded upon the model derived by Giannone, Lenza and Primiceri (2015) that utilizes Markov chain Monte Carlo (MCMC) methods, this function additionally captures volatile movements starting from March 2020 to estimate the posterior distribution of the model parameters.

### Input Parameters

1. `y` : The matrix of economic time series data, where rows represent time periods and columns represent different variables.
2. `lags` : The number of lags to include in the VAR model.
3. `kwargs`: Additional arguments to specify the BVAR model settings, including:
  - `mcmc`: Indicator for running MCMC.
  - `MCMCconst`: MCMC constant.
  - `MNpsi` : Minnesota prior hyperparameter.
  - `sur`: Indicator for stochastic volatility (SV) in the model.
  - `noc` : Indicator for no constant in the model.
  - `Ndraws` : Number of MCMC draws.
  - `hyperpriors` : Indicator for using hyperpriors.
  - `Tcovid` : Time index that determines the onset of volatility due to COVID-19.

The function sets the priors using the `set_priors_covid` function which prepares various hyperparameters needed to estimate the model. Next, it prepares the input data matrix `y` by constructing lagged values as regressors. This involves creating a matrix `x` that includes a constant term and the lagged values of `y`. With the prepared inputs, it constructs the Minnesota prior mean vector `b` and defines the initial values for the key hyperparameters `lambda`, `theta`, `mu`, `alpha` that control the prior distributions. Later, it adjusts for volatility, particularly emphasizing changes post-COVID-19. To establish scaling factors, it compares the volatility before and after COVID-19. Fitting an AR(1) model to each variable, it estimates its residual variance and numerically optimizes over hyperparameters using the `csminwel` function to maximize the log marginal likelihood of the VAR model.

## Unconditional and Conditional Forecast

Crump et. al (2019) define unconditional predictive density as:

$$f(y_{T+1}, \dots, y_{T+h} | y_{1:T}) = \int f(y_{T+1}, \dots, y_{T+h} | y_{1:T}, \theta) f(\theta | y_{1:T}) d\theta$$

where,  $\theta$  has all the coefficients of the model. Computing the predictive density requires iterating in two steps. First, we draw the coefficients from their posterior distribution. For instance, at the  $m$ -th draw, the posterior estimate is  $\theta^{(m)}$ . Then, given the posterior estimate, we sample from:

$$f(y_{T+1}, \dots, y_{T+h} | y_{1:T}, \theta^{(m)})$$

by drawing from the forecast errors  $\epsilon_{T+1}, \epsilon_{T+2}, \dots, \epsilon_{T+h}$  and iterating the model forward using the simulation smoother.

Using the VAR model estimates, the function `VARcf_DKcks` calculates the conditional forecasts by applying the Kalman filter and smoother. After reformulating the model in state-space form, we can either generate point forecasts using the Kalman filter or generate forecasts at various quantiles using the Durban-Koopman simulation smoother. Reflecting an uncertain future, the forecasts at various quantiles are forecast intervals that widen as the forecast horizon increases.

## Input Parameters

1. **x**: A matrix of observable variables containing historical data of dimension  $(T \times N)$ , wherein  $T$  is the number of time series observations, and  $N$  is the number of variables in the model.
2. **p**: The number of lags in the VAR model.
3. **beta**: The VAR model's coefficient matrix of dimension  $(Np + 1) \times N$ , where the first  $Np$  rows are lagged coefficients and the last row represents the constant.
4. **Su**: The model's variance-covariance matrix of dimension  $(N \times N)$ .
5. **nDraws**: The number of draws to generate using the Durban-Koopman smoother. If `nDraws = 0`, the function performs a simple Kalman smoothing; otherwise, it performs simulation smoothing with `nDraws` samples.

## Quantile Plots

Designing line charts with quantile bands to visualize forecast intervals and the historical data, the `quantile_plot` function plots the forecasts, sandwiched between the forecast bands. These bands represent different percentiles, where the inner band is the 25th to 75th percentile, and the outer band is the 5th to 95th band.

## Input Parameters:

1. **time**: It denotes a sequence of dates on the x-axis.
2. **quantiles**: This is a matrix of quantile values to be plotted. It can either have 5 or 7 columns:
  - For a 5-column matrix, the columns represent the outer lower quantile, inner lower quantile, center (median or mean), inner upper quantile, and outer upper quantile.
  - For a 7-column matrix, the additional two columns represent middle-lower and middle-upper quantiles.
3. **base\_color**: An RGB tuple that defines the base color for the quantile bands. If not provided, a default blue color is used.

4. **run\_scenario\_analysis**: If `True`, the plot will use a red color scheme to indicate a scenario analysis, which typically highlights a special case or alternative scenario. Otherwise, it uses a blue color scheme.
5. **show\_plot**: If `True`, the function will display the plot immediately after creating it.

## 4. Monte Carlo Simulations

Through a simple simulation exercise in the Google Colab *MCMC Simulations.ipynb* file, I evaluate and test the performance of the BVAR with a constant volatility model to assess how accurately the `bvarGLP` function estimates the parameters. So, we compare the true parameters of the simulated model with the estimated parameters drawn from the distribution and examine how far apart they are. For this, I randomly draw 500-time series of observations from a VAR(1) process:

$$Y_t = c + AY_{t-1} + \epsilon_t, \quad \epsilon_t \sim N(0, \Sigma)$$

Simulating randomly generated data from the aforementioned model, I run the BVAR model and plot the posterior distribution of each of the estimated parameters in Figure 1:  $\beta_{ij}$ ,  $\forall i = 1, 2, 3; j = 1, 2$ , which are the coefficients of the matrix  $A$  and the constant vector of the model;  $\sigma_{ij}$ ,  $\forall i = 1, 2$ , which are the variances and covariance terms of the errors in  $\Sigma$ ; and  $\lambda$ , governing how the coefficients shrink in estimation.

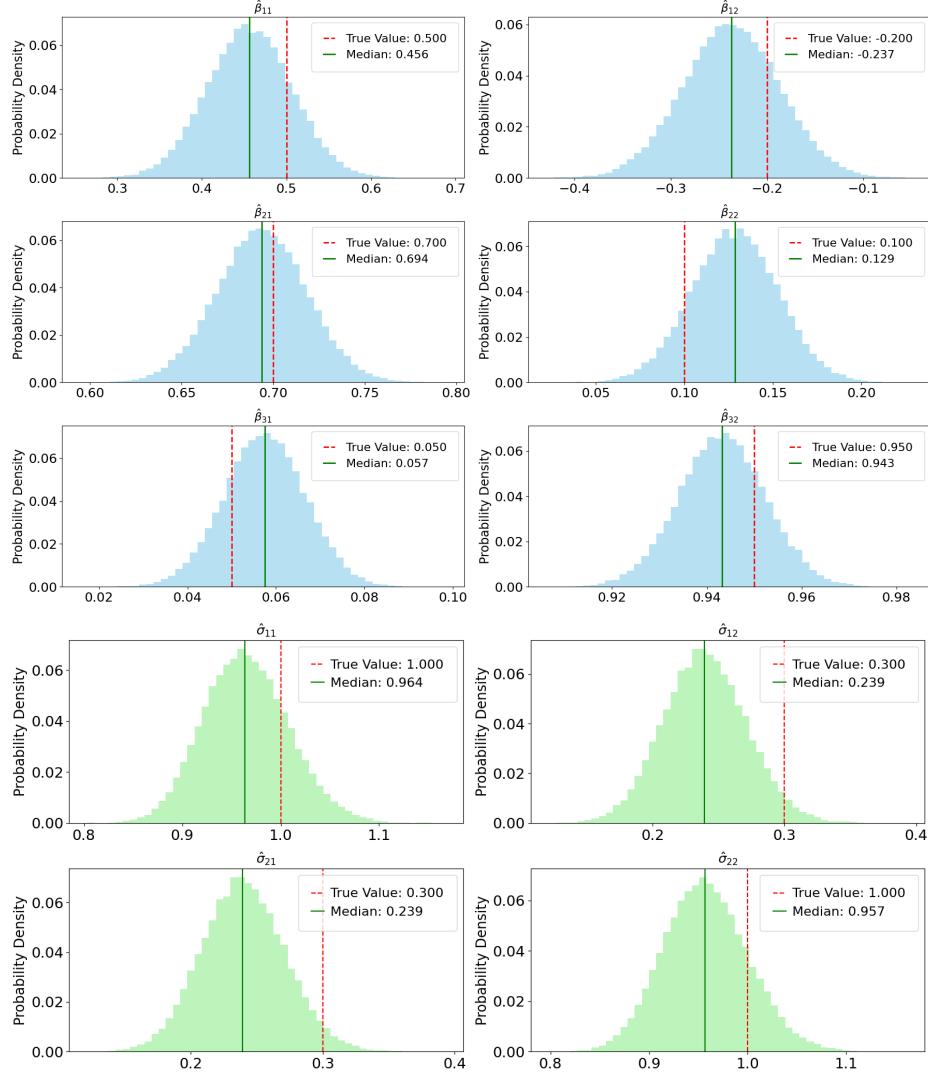


Figure 1: Posterior distribution of the coefficients of matrix  $A$ , and the variance and covariance terms of  $\Sigma$  from MCMC simulation with 100,000 draws where half of the draws were burnt-in, along with the true parameters.

Furthermore, Table 1 displays the posterior mean or the average of the 50,000 saved posterior samples, the 95 percent credible interval of the parameter, and the mean square error (MSE) for each parameter, which measures how close the estimates are to the true values. The true values for all the parameters lie within the 95 percent credible interval, suggesting that the posterior distribution is capturing the true value well, albeit the posterior mean may be slightly off. Likewise, in nearly all estimates, the MSE is very low, indicating that the posterior samples are generally very close to the true values.

Parameter	True Value	Posterior Mean	95% Credible Interval	MSE
$\beta_{11}$	0.5	0.4564	(0.3582, 0.5553)	0.0044
$\beta_{12}$	-0.2	-0.2372	(-0.336, -0.1387)	0.0039
$\beta_{21}$	0.7	0.6938	(0.6455, 0.7419)	0.0006
$\beta_{22}$	0.1	0.1285	(0.0802, 0.1762)	0.0014
$\beta_{31}$	0.05	0.0574	(0.0394, 0.0753)	0.0001
$\beta_{32}$	0.95	0.9433	(0.9252, 0.9613)	0.0001
$\sigma_{11}$	1	0.9652	(0.8849, 1.0538)	0.0031
$\sigma_{12}$	0.3	0.2399	(0.1799, 0.303)	0.0046
$\sigma_{21}$	0.3	0.2399	(0.1799, 0.303)	0.0046
$\sigma_{22}$	1	0.958	(0.8774, 1.0458)	0.0036
$\lambda$	0.2	0.1971	(0.1971, 0.1971)	0

Table 1: Summary of Posterior distributions of all parameters, 95 percent credible intervals, and mean square error as a function of each MCMC sample of the parameter and the true value of the parameter.

Finally, I create trace plots to visualize if the MCMC chains have converged to the true posterior distribution. The trace plots of all the parameters depict stationary patterns as the values fluctuate around the mean without trending over time, implying that the chain has converged. In the next section, I explain how we can apply this model using a time series data of macro and financial variables, referencing from the Google Colab notebooks.

## 5. Data

First, I downloaded the monthly data of 28 variables with four macro and financial variables each from January 1962 to October 2024 from FRED-MD. The macro variables are the measures of the labor market such as the unemployment rate, output such as industrial production, housing starts, and prices measured via CPI. Financial variables such as the 10-year Treasury yields, yield, spreads, and the S&P 500 index. Keeping the rate, such as the unemployment rate and borrowing rates, in levels or percentage points, I transformed the remaining variables, such as CPI, and VIX into logged values, specifically,  $100 \times \log$  of the levels, and we interpret the transformed variables as year-over-year growth rates.

The Google Colab notebook named as *Descriptives.ipynb* procures this data, and transforms every column identified as “log” to  $100 \times \log$  using the `transform_data` function to interpret them as percentage growth. Taking logs also helps in stabilizing the mean and variance of these non-stationary time series by compressing the spread of the data. This is particularly crucial when variables such as asset prices, volatility indexes, exhibit exponential growth rates. Without transforming them first will yield spurious correlations in VAR, and explode the forecasts. Lastly, I plot the time series of the original variable, and their transformation to detect the evolution of the variables and discern patterns or breaks (if any) in the data. The complete list of variables is in section A1 of the Appendix.

## 6. Large BVAR Model with Covid-Volatility: An Applied Example

This section presents an example from the Google Colab *main.ipynb* file, illustrating snippets of codes from *main.ipynb*. Open that file to review and run the code, and supplement it with guidelines from this section. To

run the model from the *covbayesvar* package, install the package using the `pip` command and import the functions from the package's module known as `large_bvar` as:

```
1 !pip install --index-url https://test.pypi.org/simple/ covbayesvar==0.1.6
2 import covbayesvar.large_bvar as bvar
```

If we estimate the model and produce the forecasts using updated data for the first time, we set all the boolean flags to `True` so that the script runs the model and unconditional and conditional forecasts, and saves the estimates from the simulated draws in pickle files. Once we have already run the model first, if we would like to produce the conditional forecasts based on a different set of shocks, we can set the boolean flags of estimating the BVAR and unconditional forecasts to `False`.

```
1 # Configuration settings
2 vis = True # Set to False to hide figures
3 estimateBVAR = True
4 runUNC = True
5 plot_uncond_forecasts = True
6 plot_joint_uncond_forecasts = True
7 plot_scenario_analyses = True
8 runCF = True
9 lags = 12 # Number of lags in VAR, if monthly data
10 Ndraws = 40000 # Number of draws in MCMC simulation
11 discard = 20000 # Number of initial draws to discard (burn-in period)
12 vint = datetime(2024, 11, 1) # The vintage date (forecasting start date)
```

Incorporating the COVID-related specifications, we determine the index when COVID began in March 2020, and the data from that date as a structural break, to ensure that the model accounts for the unprecedented volatility.

```
1 # COVID specific settings
2 if covid:
3     # Index of start date of estimation
4     T0 = 0
5     # Index of end date of estimation (February 2020)
6     TFeb2020 = np.where((dates.dt.year == 2020) & (dates.dt.month == 2))[0][0]
7     # First time period of COVID (March 2020)
8     Tcovid = TFeb2020 - T0 + 1
```

I establish the prior values in `prior_params` dictionary, which are the hyperparameters or the starting points of the parameters based on prior beliefs. These are the most likely value of each hyperparameter, measures of uncertainty or degree of spread around each parameter's mode, and the lower and upper bounds within which the estimated parameters must lie during Gibbs sampling.

- **lambda\_mode (default: 0.2)**: Mode of the “tightness” parameter of the Minnesota prior, which controls the scale of all the variances and covariances. Lower values tighten the prior, reducing the influence of the data on the model parameters to prevent overfitting.
- **miu\_mode (default: 1)**: Mode for the persistence (mean reversion) hyperparameter. It reflects the prior beliefs on how long the shocks persist or how slowly the shocks dissipate over time.
- **theta\_mode (default: 1)**: Mode for the cross-variable shrinkage parameter. This parameter controls how strongly we expect the variables to interact in the model, i.e., if a shock to one variable significantly impacts others in the model. Collectively, `lambda_mode`, `miu_mode`, and `theta_mode` influence the spread of the inverse-Wishart prior  $\Sigma$ . Specifically, these hyperparameters configure the variance structure of  $\Sigma$ , determining how much weight is placed on the data (likelihood), and prior beliefs of the parameters to drive the mean of the posterior distribution.
- **eta\_mode (default: [0, 0, 0, 0.8])**: Mode for the COVID-19 scaling factor’s decay parameter,  $\eta_4$ , specifically applied to the first three months after the disease outbreak. Since the scaling factors  $\eta_{t+j}, i = 0, 1, 2$  take fixed values in the first three periods and decay at a constant rate from the fourth month, the vector `[0, 0, 0, 0.8]` controls the scaling factor for the residual covariance matrix for shocks in April, May, June 2020, and afterward.  $\eta_4 \sim Beta(0.2, 0.8)$ .
- **eta\_sd (default: [0, 0, 0, 0.2])**: Standard deviation for the COVID-19 scaling factor. A low standard deviation (0.2) around the scaling factor indicates that we expect relatively small deviations from the central value of the scaling factor, reflecting a strong belief in the priors for the initial COVID-19 shock scaling.
- **eta\_min (default: [1, 1, 1, 0.005])**: Minimum bounds for each element of the scaling factor during COVID. It ensures a minimum level of impact, particularly for the fourth element (0.005), which controls the decay of the impact of shocks after the initial COVID months.
- **eta\_max (default: [500, 500, 500, 0.995])**: Maximum bound for each element of the COVID-19 scaling factor. High bounds on the first three elements (500) allow substantial impact from shocks in the first three COVID months. The fourth element (0.995) restricts the long-term effect slightly below 1, allowing for decay in the impact of shocks over time.

```

1 priors_params = {
2     'lambda_mode': 0.2, # "tightness" of the Minnesota prior: controls the scale of
3     # variances and covariances
4     'miu_mode': 1, # mean reversion hyperparameter
5     'theta_mode': 1, # mode of cross-variable shrinkage
6     'lambda_sd': 0.4, # standard deviation of the Minnesota tightness prior
7     'miu_sd': 1, # standard deviation of the persistence prior
8     'theta_sd': 1,
9     'eta_mode': [0, 0, 0, 0.8], # mode of COVID-19 scaling factor, applied to first 3 months
10    # of COVID-19 period
11    'eta_sd': [0, 0, 0, 0.2], # standard deviation of the covid-19 scaling factor
12    'lambda_min': 0.0001,
13    'alpha_min': 0.1,
14    'theta_min': 0.0001,
15    'miu_min': 0.0001,
16    'eta_min': [1, 1, 1, 0.005],
17    'lambda_max': 5,
```

```

16     'alpha_max': 5,
17     'theta_max': 50,
18     'miu_max': 50,
19     'eta_max': [500, 500, 500, 0.995]
20
21 }
```

The code estimates the BVAR on the completed transformed dataset without any missing values with 12 lags (as the data is monthly) and initial parameters.

```

1 # Estimate on complete panel
2 # Find the last time period that does not have NaN values
3 Testim = np.nanmax(np.where(~np.isnan(data_transformed.sum(axis=1)))[0]) + 1
4 bvar_results = bvar.bvarGLP_covid(data_transformed[:Testim, :], lags=lags,
5     priors_params=priors_params, mcmc=1, MCMCconst=1, MNpsi=0, sur=0, noc=0, Ndraws=Ndraws,
6     Ndrawsdiscard=discard, hyperpriors=1, Tcovid=Tcovid)
```

The arguments of the function `bvarGLP_covid` are:

- **lags**: The number of lags (12, in this case, since the data is monthly).
- **mcmc=1**: This specifies that the BVAR estimation will use the Markov Chain Monte Carlo (MCMC) method.
- **MCMCconst=1**: This parameter controls a constant factor used in the MCMC algorithm.
- **MNpsi=0**: A hyperparameter for the Minnesota prior, often set to 0 to strongly shrink the coefficients towards 0.
- **sur=0**: This parameter specifies whether to use seemingly unrelated regressions (SUR). If set to 1, the model uses SUR, allowing for cross-equation error correlation.
- **noc=0**: A flag for including no constant in the model. If set to 0, the model includes a constant term.
- **Ndraws** and **Ndrawsdiscard**: These specify the number of total MCMC draws and the number of discarded draws (burn-in period), respectively.
- **hyperpriors=1**: Enables hyperpriors for regularization.

After estimating the model, I plot the distribution of the parameters across all 20,000 saved draws. Figure 2 illustrates the distribution of the standard deviation of the March shocks or the factors that scale the shocks in the BVAR model during and after COVID-19 and the volatility decay parameter  $\eta_4$ . The posterior distribution of the standard deviation of the Minnesota prior. Since the standard deviation of the Minnesota prior is tightly distributed, the coefficients in  $\Phi$  are shrunk towards 0, yielding more precise estimates. However, if the distribution appears wider, the uncertainty about the true parameters in  $\Phi$  rises.

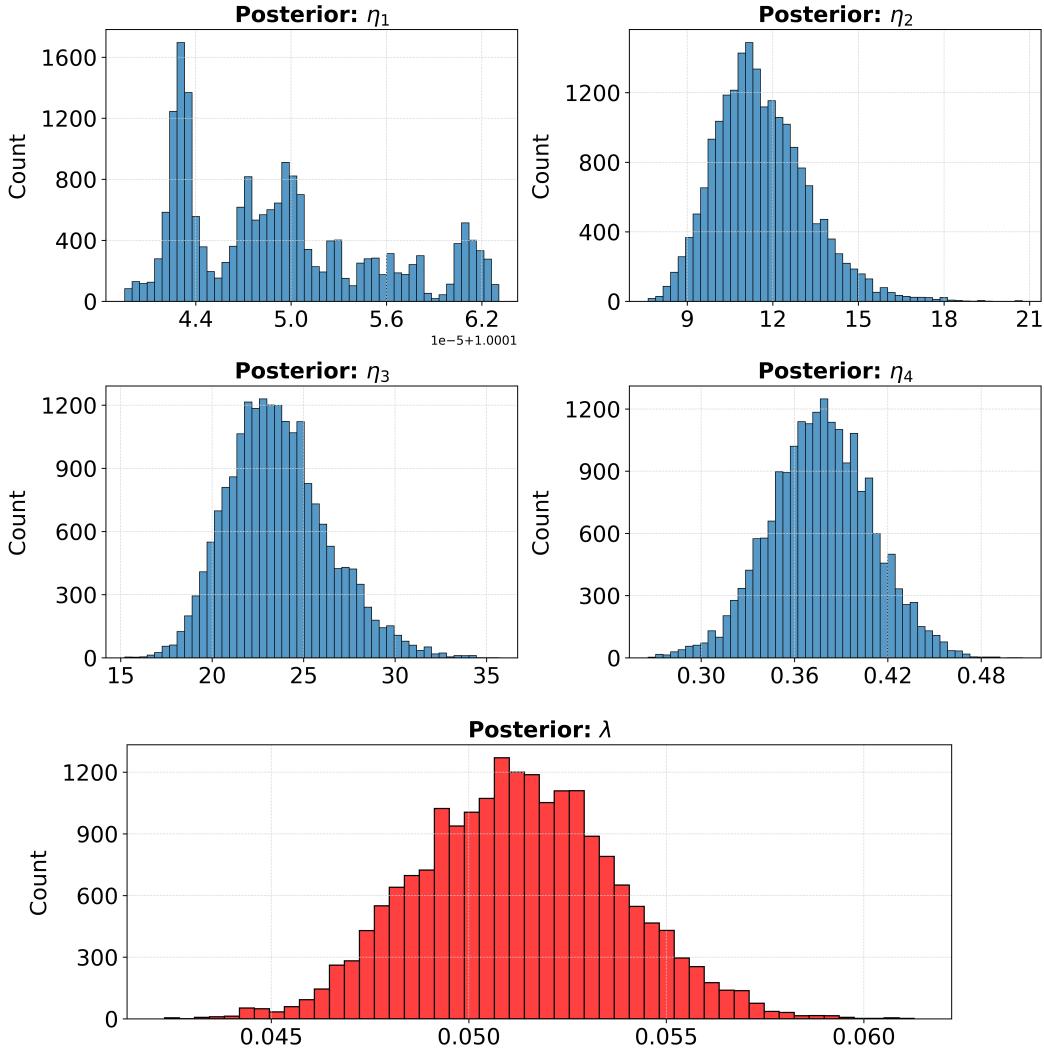


Figure 2. Posterior distribution of the Minnesota prior's standard deviation and the volatility scaling factors and the volatility decay parameter.

After estimating the parameters, I construct the unconditional forecasts for a forecast horizon of 36 months of all twenty-eight variables starting from October 2024. To accomplish that, I extract the saved parameter estimates from each of the  $j = 1, \dots, 20000$  draws from their posterior distribution. Then, I call the `VARcf_DKcks` function that generates the unconditional forecasts for  $j$ -th draw, which accumulates the forecast results for each draw across variables and months ahead in a 3D matrix called `PredY_unc`. In other words, the 3D array stores the forecasts at the  $h$  periods for  $n$  variables across all  $J$  draws.

```

1 for j in range(ndraws): # Loop through the number of draws
2     if (j % 10) == 0 or j == ndraws - 1:
3         print(f"Generating unconditional forecasts: {j} of {ndraws} draws...")
4         sys.stdout.flush()
5
6     # Extract the j-th draw for beta and sigma
7     beta_j = bvar_results['mcmc']['beta'][:, :, j]
8     Gamma_j = np.vstack((beta_j[1:, :], beta_j[0, :]))

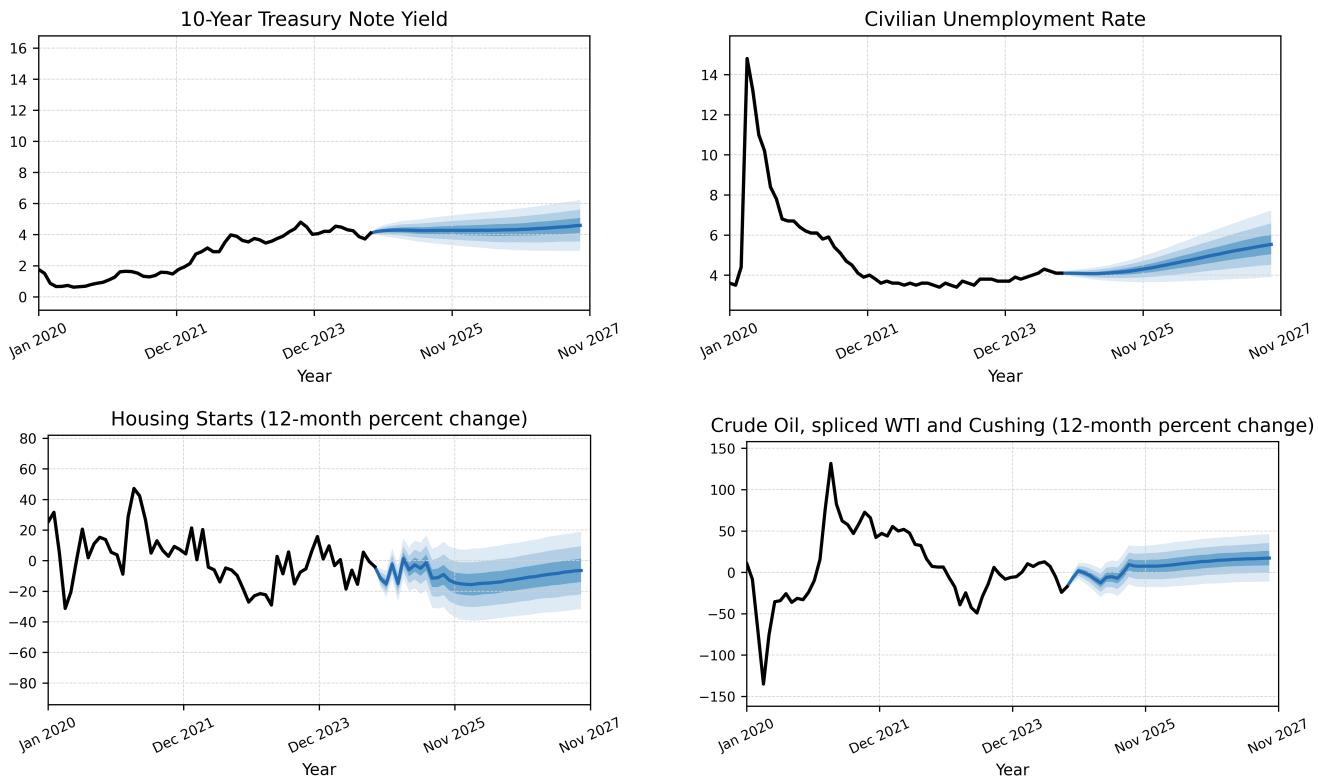
```

```

9   Su_j = bvar_results['mcmc']['sigma'][:, :, j]
10
11 PredY_unc[:, :, j] = bvar.VARcf_DKcks(YFore, bvar_results['lags']['lags'], Gamma_j,
    Su_j, 1)

```

I plot the unconditional forecasts in various quantiles: [0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8] along with the historical data. For instance, figure 3 portrays the historical data from January 2020 to October 2024, followed by the model-based “baseline” probabilistic forecasts in varying shades of blue. The dark blue line denotes the most likely “median” forecast or the forecast in the middle of the distribution. Half of the forecasted values lie above the forecast denoted by dark blue, and the other half lies below this point. Likewise, the forecasts in the quantiles 0.2 and 0.8 connote that 20 percent of the forecasts fall below this point, and 80 percent are above it, implying that these forecasts are closer to the tails of the distribution. Whilst figure 3 shows the forecasts only for four variables, the code generates similar graphs for all the variables and saves them individually as PNG files.



*Figure 3. Actual values of the industrial production growth, unemployment rate, housing starts and oil prices (in black) till October 2024. Median forecasts for three years (in dark blue) and shaded regions represent forecast bands at 80 and 60 percent coverage intervals.*

## Scenario Analysis

Further exploiting the versatile capabilities of the large BVAR model, we can employ the model to answer more policy-based questions, such as the impact of supply and demand shocks on predicted paths of key economic variables. I model a “forward-looking” scenario where tighter immigration rules such as mass deportations of undocumented

workers and border controls shrink the labor force, constricting the supply of workers in the economy. Fewer workers can reduce the production of goods in the short run, especially those employed in construction, hospitality, and food services, weakening demand. This supply shock puts an upward pressure on wages as employees may compete more aggressively for a smaller pool of workers in the short run. To model this simple scenario, I assume that the total employees on non-farm payroll reduce by 5 percent 5-10 months into the future and the average hourly earnings of production and non-supervisory workers rise by 2 percent after labor supply shock hits the economy 7-10 months ahead.

Coupled with the supply shocks, the PCE inflation declines by 1 percent point in the short run: 3-5 months ahead. Leaving other variables unconstrained, figure 3 depicts the scenario analyses for different segments of the economy, with posterior median response surrounded by the 68 and 90 percent coverage intervals. Here, the conditional forecasts assume that the fixed paths of all employees in the nonfarm payroll and average hourly earnings may possess information about how other variables could react. These shocks in non-farm payroll employees and average hourly earnings define a scenario:

$$S_t = \{\alpha_{L_s} \epsilon_{l,T+s} + \alpha_{m_s} \epsilon_{m,T+s}, s = 1, 2, \dots\},$$

where  $\alpha_{L_s}$  and  $\alpha_{m_s}$  are the coefficients associated with non-farm payroll employees, and average hourly earnings, respectively.

Similar to the unconditional predictive density, the conditional predictive densities additionally depend on a conditioning set  $C_t$  that contains scenarios or realized future paths of a set of variables:

$$f(y_{T+1}, \dots, y_{T+h}|y_{1:T}, C_t) = \int f(y_{T+1}, \dots, y_{T+h}|y_{1:T}, \theta, C_T) f(\theta|Y_{1:T}) d\theta$$

Comparable to simulating from the unconditional predictive density, here we also iterate in two steps. First, we draw the coefficients  $\theta^{(m)}$  at the  $m$ -th draw from their posterior distribution conditional on the scenario. Then, given the posterior estimate, we sample from  $f(y_{T+1}, \dots, y_{T+h}|y_{1:T}, \theta^{(m)}, C_T)$  using Kalman filter. Crump et. al (2021) apply the Kalman filter and simulation smoother derived in Banbura, Giannone and Lenza (2015) to make the model computationally feasible with a large dimension system.

In terms of code, the python function `VARcf_DKcks` generates and stores the conditional forecasts in a 3D matrix, just as the same function geneated the unconditional forecasts earlier. First, we define the indices of the variables on which we impose the shocks, then initialize a matrix of NaN values which will store the conditional forecasts. Next, we set the shocks for specified periods on the variables. Looping through the MCMC draws from the posterior distribution, we add the magnitude of the shock values to the unconditional forecasts and use the posterior estimate of the parameter from each draw to generate the conditional forecasts at each draw.

```

1 # Find indices of specific variables: All employees, total non farm
2 idxCV1 = Spec.index[Spec['SeriesID'] == 'PAYEMS'].tolist()[0]
3 # Average Hourly Earnings of Employees
4 idxCV2 = Spec.index[Spec['SeriesID'] == 'CES0600000008'].tolist()[0]
5 # Create a matrix of NaNs to store shocks
6 # n is the number of variables and T is the length of the initial data
7 Ycond = np.nan * np.ones((len(DateAll), n))
8
```

```

9 # Fill the initial part of Ycond with transformed data
10 Ycond[:T, :] = data_transformed
11 # Create a matrix for shocks
12 Shock = np.nan * np.ones((h_fore.sum(), n)) # h_fore is a boolean array indicating
13   forecasts
13 Shock[5:10, idxCV1] = -5 # Apply shocks to % change in All Employees
14 Shock[7:10, idxCV2] = 2 # Apply shocks to % change in Average Hourly Earnings
15
16 # Initialize PredY_con for conditional density forecasts
17 PredY_con = np.nan * np.ones((len(DateAll), data_transformed.shape[1], ndraws))
18 for j in range(ndraws): # Loop through the number of draws
19
20   if (j % 10) == 0:
21     print(f"Generating conditional forecasts: {j} of {ndraws} draws...")
22     sys.stdout.flush()
23   # Extract the j-th draw for beta and sigma
24   Ycond[h_fore_1d, :] = PredY_unc[h_fore_1d, :, j] + Shock
25   beta_j = bvar_results['mcmc']['beta'][..., :, j]
26   Gamma = np.vstack((beta_j[1:, :], beta_j[0, :]))
27   Su = bvar_results['mcmc']['sigma'][..., :, j]
28   PredY_con[:, :, j] = bvar.VARcf_DKcks(Ycond, bvar_results['lags']['lags'], Gamma, Su, 0)
29
30
31 # Compute the difference between conditional and unconditional forecasts
32 dY = PredY_con - PredY_unc

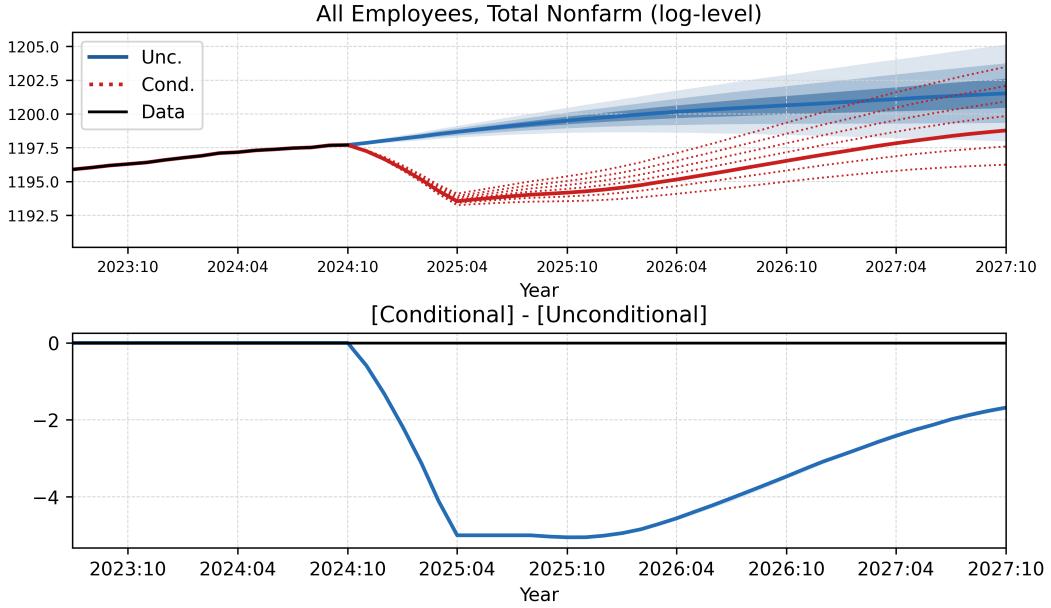
```

The top graphs in Figure 4 a) and b) show the trajectory of all employees in the non-farm payroll and industrial production index after the former slumps by 5 percent points, respectively, from its benchmark forecasted paths. In contrast, the bottom graphs showcase the scenario, which is the difference between the forecasts conditional on the shocks on both variables, and the unconditional or baseline forecasts, which are devoid of any shocks. Mathematically,

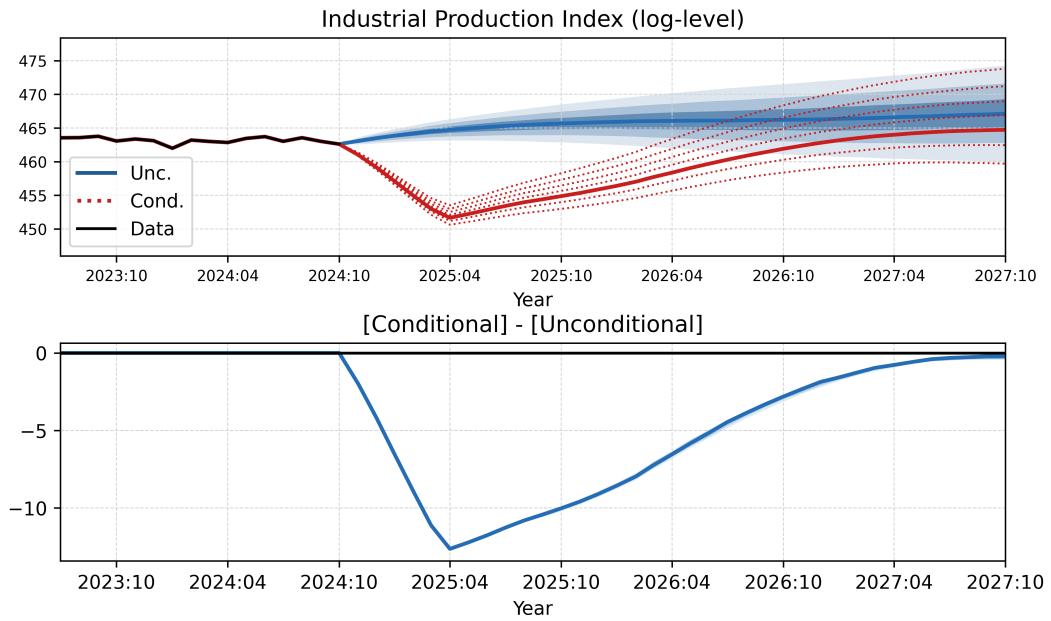
$$D[y_{T+s}|y_{1:T}, C_T, \theta] = E[y_{T+s}|y_{1:T}, C_T, \theta] - E[y_{T+s}|Y_{1:T}, \theta]$$

$$\Rightarrow \int y_{T+s} f(y_{T+s}|y_{1:T}, \theta, C_T) dy_{T+s} - \int y_{T+s} f(y_{T+s}|y_{1:T}, \theta) dy_{T+s}$$

These forecasts are devoid of any structural identification schemes for the errors and are entirely based on past historical correlations among the variables in the system. Because the scenarios are derived from the reduced-form models, without identifying the structural shocks, I do not interpret them. In the Google Colab notebook, I use the `quantile_plot` function to generate these  $(2 \times 1)$  subplots for all 28 variables and save each of them as PNG files.



*Fig 4a.* Conditional forecasts (in red) and unconditional forecasts (in blue) of log-transformed data on all employees in non-farm payroll where the shaded areas are the 60 and 80 percent forecast intervals. Lighter-shaded regions are forecasted paths that are less likely to occur. The forecast bands or credible intervals denoted by red dots result from the conditions that the number of employees in non-farm payroll falls by 5 percent, and wages rise by 2 percent.



*Fig 4b.* Conditional forecasts (in red) and unconditional forecasts (in blue) of the log-transformed Industrial Production Index.

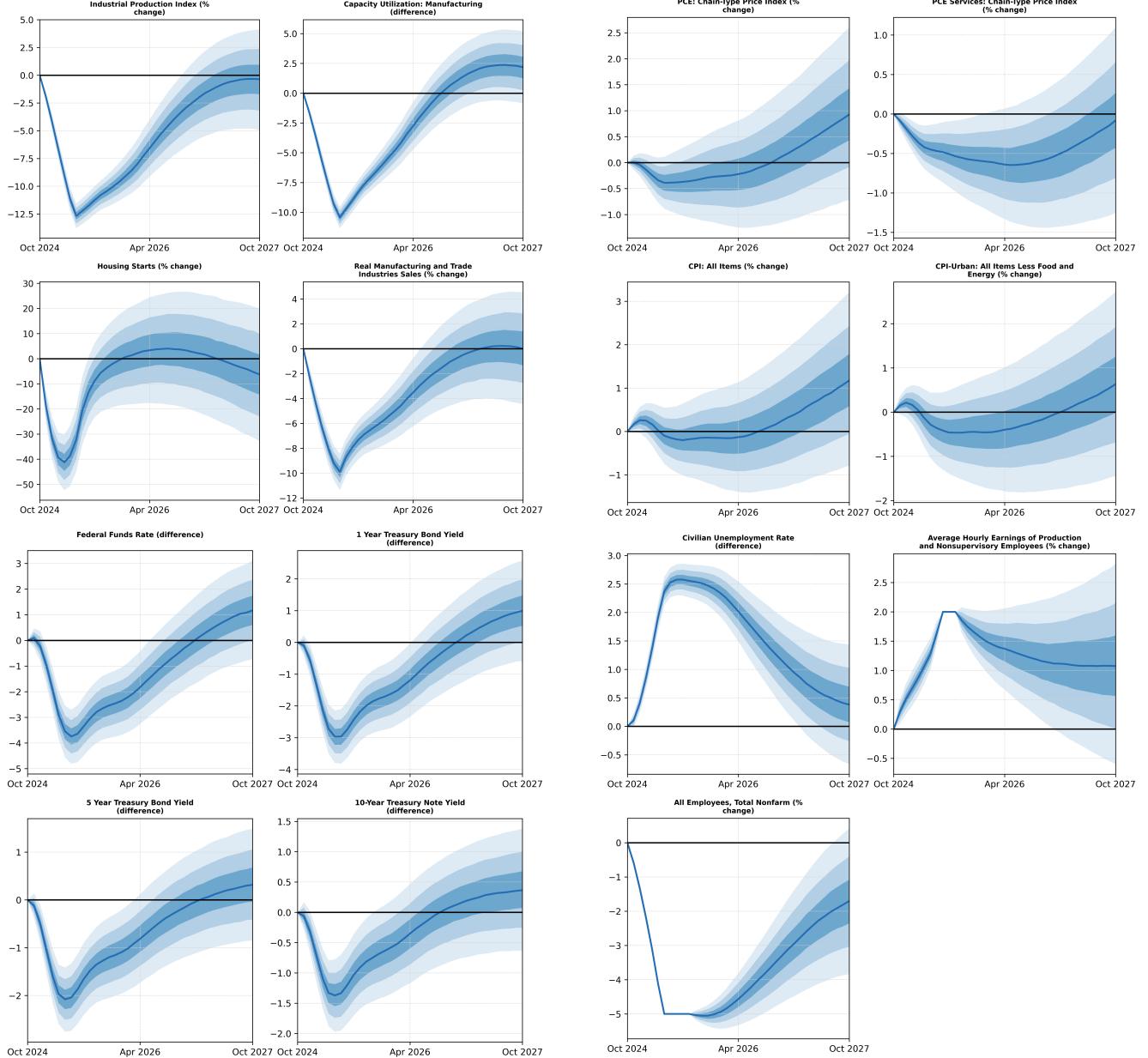
In the same spirit, the difference between conditional and unconditional forecasts is equivalent to the generalized impulse response functions to shocks in non-farm payroll employees and average hourly earnings. Formally, the impulse response function is the difference between the counterfactual and the baseline as:

$$\frac{\partial y_{i,T+h}}{\partial v_{j,T}} = \mathbb{E} [y_{i,T+h}|y_{1:T-1}, v_{j,T} = 1, \gamma] - \mathbb{E} [y_{i,T+h}|y_{1:T-1}, \gamma]$$

where,  $v_{j,T}$  is the structural shock which is a linear combination of forecast errors,  $v_{j,T} = \alpha' e_T$ . In the above example, it is the linear combination of the forecast errors on non-farm payroll employees and average hourly earnings. Figure 5 displays the elasticities or responsiveness of all the variables to the persistent negative shocks on employees in the non-farm payroll and increasing hourly earnings.

Notably, we see that the measures of economic activity - capacity utilization, real manufacturing, and trade industrial sales decline by 10 percent from their baseline paths. Because of the relatively sudden shocks, firms cannot instantly replace or automate away the lost labor, slowing down output in the housing, construction, and manufacturing sectors, and spilling over into overall industrial production. The contracting economy dwindle the supply of goods and services, which also reduces the jobs available as the demand falls.

This spikes the unemployment rate as witnessed in the graph in the short run, immediately after the shock. A rise in the unemployment rate by 2.5 percent owing to lower output dips inflation if the negative output effect is larger than the cost-push effect from higher wages. But in the long run, if employers increasingly automate, offshore, or reorganize production, wage and price pressures dampen. In other words, they revert to their long-run means as evident in the scenario analysis in Figure 5.



*Figure 5. Response of all the variables when non-farm payroll employees persistently contract by 5 percent and average hourly earnings rise by 2 percent. The blue line connotes the median response, and the shaded regions indicate the 80 and 60 percent coverage intervals.*

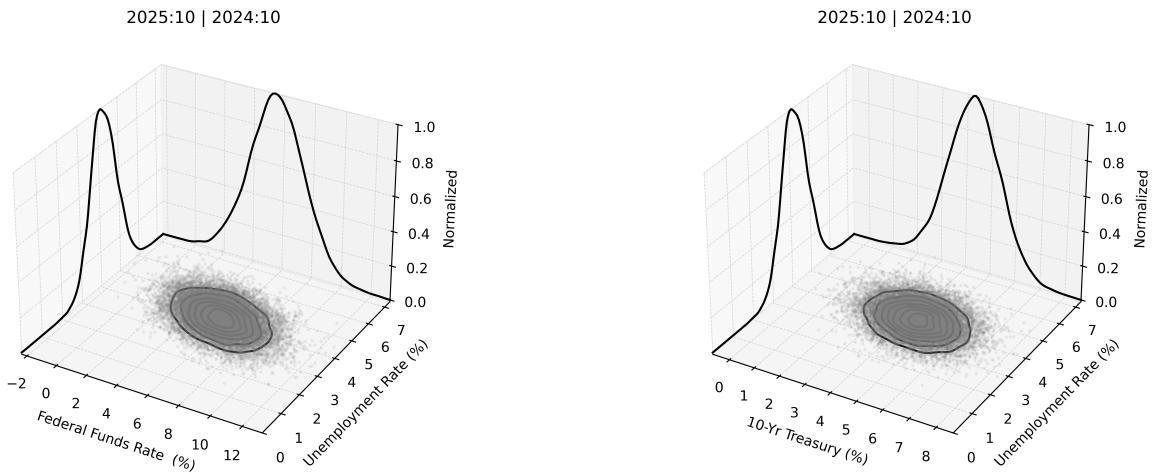
## Joint Forecast Density: Unveiling Macro Risks, and Relationships

Until now, I had emphasized estimating the baseline and conditional forecasts of individual variables and computing the univariate densities of the forecasts. While this is useful to understand the expected path of the macro variables, we can also depict the entire plausible set of outcomes and demonstrate how two variables might co-move. For instance, with what probability does a high unemployment rate trigger a lower federal funds rate, or can both be high simultaneously? How does the predicted unemployment vary with monetary regime? How will the joint forecasts alter if the Fed targets lower inflation or unemployment? Are there potential trade-offs or correlations in

the forecast period, and how uncertain are the forecasts?

To answer these kinds of questions, we can construct joint forecast density plots of any forecast horizon between two variables as exemplified in figure 6. The vertical dimension tells us the estimated joint probability of observing two macro variables one year ahead in October 2025, using the historical data till October 2024. Every point in the 2D plain is a possible combination, and the height denotes the likelihood of the combination from the predictive density. The grey dots, representing the cloud of points, are the posterior draws from the BVAR's predictive distribution. The side panes denote the marginal density plots, indicating the marginal distribution of forecasts. Showing a likelihood of various combinations, it helps to form internally consistent narratives under different scenarios. The tallest part of the surface is around the center of the dot cloud, implying the highest joint density or the “most typical” scenario occurs where the Fed Funds rate is around 4–5 percent and unemployment around 3.5–4 percent. A single peak in the graph connotes that the joint forecast distribution is unimodal. Predominantly, this occurs when the federal funds rate is approximately 5 percent. The joint density contours are concentrated in specific areas, suggesting a slight negative correlation between unemployment rate and the federal funds rate. Namely, lower unemployment (3–5%) is likely to coincide with either moderate (~5%) or significantly elevated (~12%) federal funds rate.

As dots heavily cluster when the unemployment rate is around 3–5 percent, the model expects a resilient labor market even in scenarios with aggressive monetary transmission (of ~12%). The latter can occur when inflation is out of control, raising the federal funds rate in response, but the economy is strong and lands softly. However, the right tail indicates that there are risks of elevated unemployment (~6–8%) in adverse economic conditions, albeit these are less likely. Another less likely scenario is stagflation, which marks periods of high unemployment and inflation mired with stagnant growth in output, triggering the Fed to reign in on spiralling prices by hiking the policy rates. This is because high unemployment clusters with lower federal funds rate in the left graph. The spread of the density contours signal uncertainty in forecasts where broader spread along the federal funds rate axis connotes higher uncertainty about the future path of the interest rates relative to those of unemployment rate.



*Figure 6. Joint forecasts of the variables for October 2025 using the historical data till October 2024. The grey cloud of the points in the center and the lightly visible contour line are the posterior predictive densities. The marginal densities of the forecasts are in the side panels.*

# Entropic Tilting: Softly Anchoring Forecasts to Long-Run Targets

Previously, I created baseline forecasts of all the variables without considering any targets, or long-run steady-state rates of variables set by the Federal Reserve. The BVAR model generates forecasts using MCMC simulations, assigning equal weights to each draw in the forecast distribution. Sometimes, central banks explicitly target variables such as inflation and unemployment rates, assuming that the inflation rate will converge to 2 percent in the long run. For instance, table 2 presents the central tendency of long-run projections of the FOMC released in the Summary of Economic Projections (SEP) in December 2024, capturing the middle-ground belief of policymakers regarding the long-run path of these variables. How can we adjust the forecasts to reflect these beliefs? The goal is to tilt the baseline distribution to reflect these beliefs while remaining as close to the original distribution. In this exercise, I employ the method of entropic tilting elucidated in Tallman and Zaman (2020) to modify the baseline distribution and refer to the *Entropic\_Tilting.ipynb* file in Google Colab and walk through the implementation procedure mathematically and programmatically. Unlike the scenario analyses above where I fixed the future paths of certain variables, also known as hard conditioning; I now relax that by allowing the conditioned future values of certain variables to lie within a certain range, also known as soft conditioning. Because we generally don't know the realized paths of endogenous variables in the long horizon, constraining the conditioned path of the variable within a certain range or an interval instead of an exact path is simpler. Furthermore, soft constraints recognize the uncertainty around the paths observed in the future.

Variable	Projection for 2026 (%)	Projection for 2027 (%)	Long run projection (%)
Unemployment rate	4.1 – 4.4	4 – 4.4	3.9 – 4.3
PCE inflation rate	2 – 2.2	2	2
Federal funds rate	3.1 – 3.6	2.9 – 3.6	2.8 – 3.6

Table 2: Summary of Economic Projections released on December 18, 2024. These projections are central tendencies for change in the unemployment rate and PCE inflation rate, and the projected path of the federal funds rate. The central tendencies exclude the outlier projections (the three highest and lowest forecasts) and reflect FOMC's consensus view.

Let  $y \in \mathbb{R}^n$  represent a vector of forecasts for  $n$  variables. If we run  $J$  draws from the MCMC simulations, then  $y_j \forall j = 1, 2, \dots, J$  is a forecast at the  $j^{\text{th}}$  draw. Initially, we assign equal weight to each draw,  $w_j = \frac{1}{J}$ . Now, we re-weight the baseline distribution  $f(y)$  to a tilted distribution  $\hat{f}(y)$  such that two conditions are met. Firstly, the reweighted distribution must satisfy specific moment conditions. Secondly, the tilted distribution must be as close to the original distribution.

The moment conditions are

$$E_{\hat{f}}[g(y)] = \bar{g},$$

where  $g(y)$  is a set of  $m$  moment conditions, and  $\bar{g} \in \mathbb{R}^m$  contains the target moments, which are the central tendency forecasts.

To determine the tilted distribution, we minimize the Kullback Leibler (KL) divergence between  $f(y)$  and  $\hat{f}(y)$ :

$$\min_{w_j^*} KL(\hat{f}, f) = \sum_{j=1}^J w_j^* \log(Jw_j^*) \text{ s.t. } \sum_{j=1}^J w_j^* = 1, \frac{1}{J} \sum_{j=1}^J w_j^* g(y_j) = \bar{g}.$$

The solution to find the optimal weights is

$$w_j^* = \frac{\exp(\gamma^\top g(y_j))}{\sum_{j=1}^J \exp(\gamma^\top g(y_j))}.$$

Here,  $\gamma$  is the Lagrange multiplier associated with the moment condition, and  $g(y_j)$  is the moment function evaluated at  $j^{\text{th}}$  draw. We normalize so that the weights across all draws sum to 1. Before we can find the optimal weights, we optimize the Lagrange multiplier as

$$\gamma = \arg \min_{\gamma} \sum_{j=1}^J \exp(\gamma^\top g(y_j) - \gamma^\top \bar{g}).$$

Finally, the re-weighted (tilted) moments should match the target values  $\bar{g}$  within a small tolerance:

$$E_{\hat{f}}[g(y)] = \frac{1}{J} \sum_{j=1}^J w_j^* g(y_j)$$

The code in *Entropic\_Tilting.py* file implements “soft conditioning” by adjusting the baseline forecasts of key variables to reflect long-run projections from FOMC’s SEP.

Below is the average of the range of the central tendency values.

```

1 # Conditioning assumptions: SEP released on Dec 2024 SEP for 2027
2 # Center of the central tendency: midpoint of the range
3 # find the index of the PCE inflation rate, Federal Funds Rate, and Unemployment Rate in
4 # the Spec DataFrame
5 idxCVPCE = Spec[Spec['SeriesName'] == 'PCE: Chain-Type Price Index'].index[0]
6 valCVPCE = (2 + 2) / 2
7
8 # Federal Funds Rate
9 idxCVFFR = Spec[Spec['SeriesName'] == 'Federal Funds Rate'].index[0]
10 valCVFFR = (2.9 + 3.6) / 2
11
12 # Unemployment Rate
13 idxCVLR = Spec[Spec['SeriesName'] == 'Civilian Unemployment Rate'].index[0]
14 valCVLR = (4 + 4.4) / 2

```

Next, we procure the BVAR estimates `bvar_results` and distribution of unconditional forecasts `PredY_unc` computed and saved in pickle files in the main.py file. Then, we tilt the original forecast distribution to anchor the mean of the distribution to the midpoint of the central tendencies of SEP projections.

```

1 ##### Entropic Tilting: Shifting the mean
2 #####
3 n = PredY_unc.shape[1] # Number of series (second dimension)
4 # 12-month change

```

```

5 dPredY_unc = np.vstack((
6     np.full((12, n, ndraws), np.nan), # Add NaNs at the beginning
7     PredY_unc[12:, :, :] - PredY_unc[:-12, :, :]
8 ))
9
10 # MCMC Draws for tilting variables (PCE inflation, unemployment rate, and federal funds
11 # rate) at forecast horizon
12 YYh = np.vstack((
13     dPredY_unc[-1, idxCVPCE, :].T, # Extract indices of federal funds rate and PCE
14     inflation rate
15     PredY_unc[-1, [idxCVLR, idxCVFFR], :] # Extract unemployment rate
16 )).T
17 # Target values: FOMC's central tendency projections for the targeted variables
18 target = np.array([valCVPCE, valCVLR, valCVFFR])
19
20 # Objective function to find the optimal lambda (Lagrange multiplier) values
21 def fun(gamma, YY, g0):
22     return np.sum(np.exp((YY - g0) @ gamma))
23
24 # Optimization setup
25 opts = {'tol': 1e-20, 'options': {'maxiter': 1000}}
26 objective = lambda x: fun(x, YYh, target) # Objective function for optimization
27 gamma_init = np.ones(len(target)) # Initial guess for gamma
28 # Uses a quasi-Newton method for constrained optimization:
29 # minimizes the divergence between the original and tilted distribution while ensuring the
30 # mean of the tilted
31 # distribution satisfies the target constraints
32 res = minimize(objective, gamma_init, method='L-BFGS-B', **opts)
33 # optimal gamma values that adjust the weights of the forecast draws to align the
34 # distribution with the targets
35 gammaStarMean = res.x
36
37 # minimized objective function value (Kullback-Leibler divergence between the original and
38 # tilted distributions)
39 fStar = res.fun
40 # re-calculate weights for the forecast draws using the optimal gamma values
41 # normalize the weights to sum to 1
42 wStarMean = np.exp(YYh @ gammaStarMean) / np.sum(np.exp(YYh @ gammaStarMean))
43
44 # Verify moment condition
45 # check the mean of the re-weighted draws to ensure it aligns with the target values
46 print("Moment:")
47 # weighted average of the forecasted draws that must equalize the target values
48 print(np.mean(wStarMean[:, None] * YYh * 10000, axis=0)) # Scale by 10000
49 print("Target:")
50 print(target)

```

A disadvantage of tilting towards the mean is that the tilted forecasted distribution will be sensitive to outliers

or extreme values, distorting results if outliers are unreliable. Skewing the forecasting distribution, the mean may not accurately represent the central tendency. On the other hand, extreme values don't affect the median, and central banks often focus on median forecasts while generating counterfactual forecasts. Therefore, I present an example where I tilt the forecast distribution so that the median of the forecast distribution matches the target values, ensuring that the tilted distribution stays as close as possible to the original distribution. I achieve this by minimizing the KL divergence:

$$\min_{w_j^*} KL(\hat{f}, f) = \sum_{j=1}^J w_j^* \log(Jw_j^*) \text{ s.t. } \sum_{j=1}^J w_j^* = 1, \frac{1}{J} \sum_{j=1}^J w_j^* g(y_j) = 0.5$$

Now, the moment condition changes to:

$$E_{\hat{f}}[g(y)] = 0.5, \text{ where } g(y_j) = 1\{y_j \leq \text{Target}\}$$

Therefore, 50 percent of the tilted distribution's mass should lie below the target  $\bar{g}$ .

```

1 ##### Entropic Tilting: Shifting the median
2 #####
3 # Create the indicator matrix to check if every value in the forecast matrix is less than
4 # or equal to the target value
5 YYhTemp = YYh <= target
6 YYhTemp = YYhTemp.astype(int) # Convert the binary (True/False) to integer (1 and 0)
7
8 # Define the optimization objective
9 # find the optimal gamma that minimizes the Kullback-Leibler divergence between the
10 # original and tilted distributions
11 objective = lambda gamma: np.sum(np.exp((YYhTemp - 0.5) @ gamma))
12
13 # Perform optimization
14 gamma_init = np.ones(len(target)) # Initial guess
15 res = minimize(objective, gamma_init, method='L-BFGS-B', options=opts)
16 gammaStarMedian = res.x
17 fStar = res.fun
18
19 # Calculate weights for each forecast draw using the optimal gamma values
20 # Reweight the forecast draws (rows of YYh) using the optimized gamma values.
21 # Normalize the weights so they sum to 1.
22 wStarMedian = np.exp(YYhTemp @ gammaStarMedian) / np.sum(np.exp(YYhTemp @ gammaStarMedian))
23
24 # Check that conditioning assumptions are satisfied
25 # Sort the forecast values (YYh) for each variable to compute the weighted median.
26 temp_s = np.sort(YYh, axis=0) # Sorted values
27 idx = np.argsort(YYh, axis=0) # Indices for sorting
28 cumsum_w = np.cumsum(wStarMedian[idx[:, 0]]) # Cumulative weights for the first column
29 j = np.argmin(np.abs(cumsum_w - 0.5)) # Find the index closest to 0.5
30 print("Value at median:", temp_s[j]) # Value corresponding to median

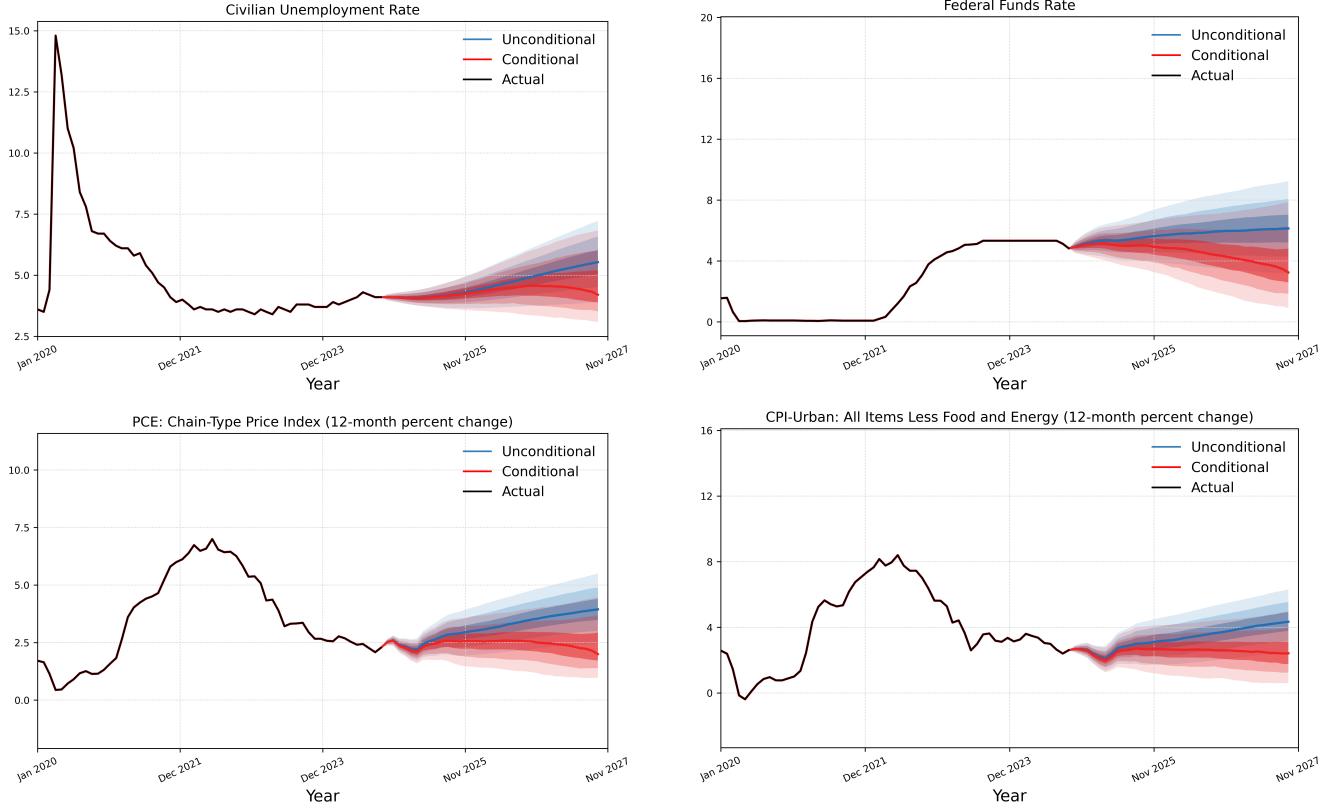
```

```

28 print("Target:")
29 print(target)
30
31 print("Median:")
32 median_values = np.array([
33     bvar.wquantile(YYh[:, 0].reshape(1,-1), 0.5, wStarMedian), # Weighted median for
34         column 1
35     bvar.wquantile(YYh[:, 1].reshape(1,-1), 0.5, wStarMedian),
36     bvar.wquantile(YYh[:, 2].reshape(1,-1), 0.5, wStarMedian)
37 ])
38 print(median_values)

```

Figure 7 illustrates the unconditional (baseline) forecast distribution from the BVAR model without imposing any external constraints or target (in shades of blue) and the conditional forecast distribution that considers long-run targets for PCE inflation rate, unemployment rate, and projected paths of the federal funds rate (in shades of red). Thereby, I introduce judgment by anchoring the median of the BVAR forecasts to the midpoint of the central tendency forecasts for the 2027 reference period. Upon tilting to the median, the red bands shift the forecast distribution such that 50 percent of the probability mass lies below their respective targets. Additionally, the median forecasts of PCE inflation, unemployment, and federal funds rate (denoted by dark red in the center of the conditional probabilistic distribution) converge to the stated targets of 2, 4.2, and 2.25 percent, respectively. This makes the forecasts more consistent with the consensus views of the FOMC, particularly in the medium to long run, where uncertainty might be higher, widening the forecast intervals.

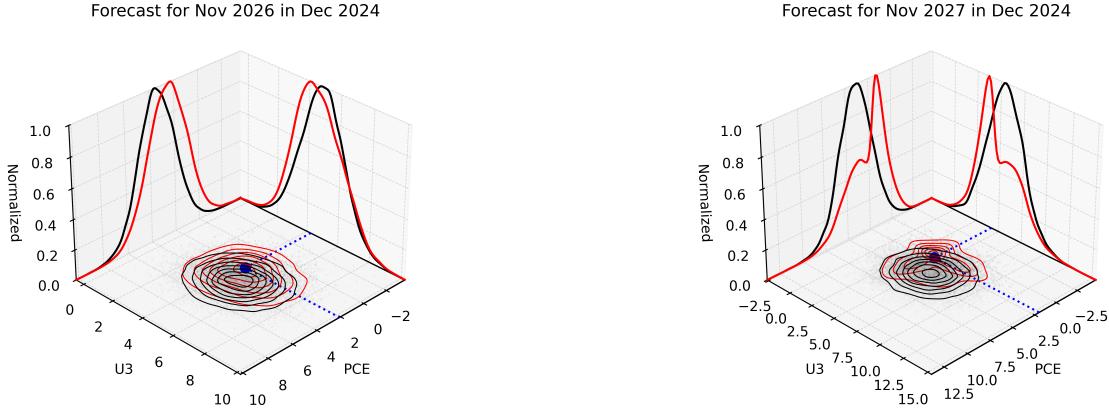


*Figure 7. Unconditional forecasts and forecasts conditional on SEP projections of PCE inflation rate to 2 percent, unemployment rate to 4.2 percent, and federal funds rate to 3.25 percent 3 years into the future. These conditional forecasts are produced after tilting the forecast distribution such that the median of the forecast distribution anchors to the target SEP projections.*

Figure 8 exhibits joint predictive densities of the unemployment and PCE inflation rates for 2 and 3 years into the future. The black lines are the unconditional joint predictive densities analogous to those in Figure 6. Alternatively, the red counterparts are the tilted distributions, matching the central tendencies of SEP projections for 2026 and 2027, respectively. The blue dots in the center of the red contours are the midpoint values of the central tendencies of the forecasts of a given year. These are the averages of the lower and upper range of the values in Table 2, yielding the midpoint forecast of 4.25 and 4.2 percent for the unemployment rate in 2026, and 2027, respectively. The unconditional and conditional marginal forecasts for densities for two years ahead (left) are comparable and unimodal, while those for three years ahead (right) are multi-modal.

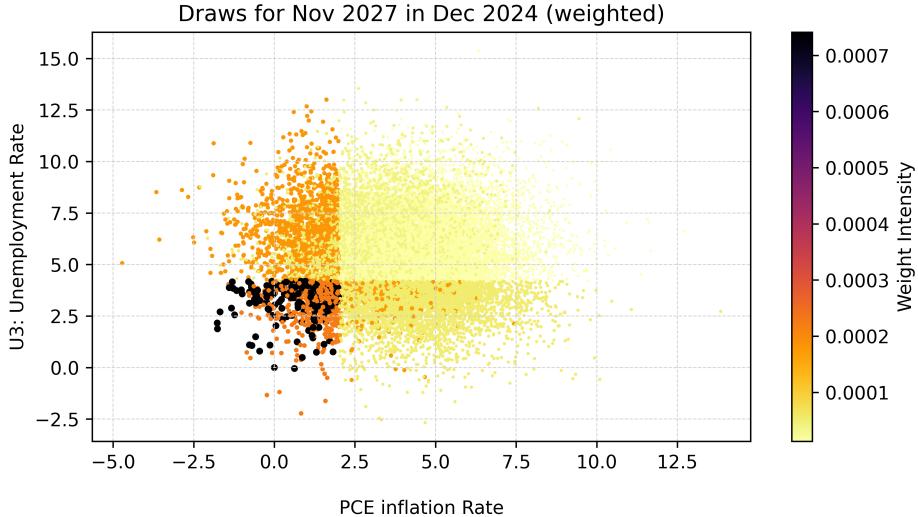
Multiple peaks in the distribution can occur due to a few reasons. Firstly, as the forecast horizon extends further into the future, forecasts become more uncertain, amplifying the spread of the forecast distribution. Secondly, the Federal Reserve's stated long-run targets (for example, 2 percent inflation) anchor for expectations. However, the path to reaching these targets may vary significantly depending on how the Fed reacts to economic developments. To elaborate, the joint predictive forecast density for November 2027 has three peaks. One occurs when the PCE inflation rate is approximately 2 percent, and the unemployment rate is approximately 4.2 percent. This peak corresponds to the Fed's long-run target where inflation stabilizes at 2 percent and the unemployment rate at 4.2 percent. The red-tilted density heavily weights this outcome, reflecting alignment with the SEP projections. In a stable and ideal economic scenario, the variables smoothly converge to the long-run equilibrium. Another peak occurs when the PCE inflation rate is approximately 3 percent, and the unemployment rate is approximately 5.5 percent. Signifying a stagflation-like scenario where inflation remains above target and unemployment is higher, a

potential reason might be the wage-price spiral. Finally, a smaller peak occurs when inflation and unemployment rates are at approximately 1 and 3.5 percent, respectively. Whilst less likely, this optimistic outcome connotes a scenario where aggressive tightening of monetary policy can risk deflation and drive unemployment below long-run expectations.



*Figure 8. Joint forecast density functions of the forecast of the unemployment rate and PCE inflation rate for November 2026 (left), and November 2027 (right).*

Thirdly, since entropic tilting modifies the posterior distribution by reweighting the MCMC draws to satisfy moment conditions, reweighting magnifies certain regions of the joint density (combination of unemployment and inflation rate) that aligns with the targets, and down-weights other regions. To elucidate, if the original posterior distribution already had higher density near the targets, tilting amplifies these regions, making the modes more pronounced. Moreover, if more draws from the less-dense regions are favored to satisfy the moments, tilting creates new modes. Relatedly, the scatterplot in Figure 9 visualizes the joint draws of inflation and unemployment rates for November 2027. Here, each point in the graph represents one MCMC draw from the posterior predictive distribution color-coded by the magnitude of the weights. For instance, MCMC draws (or points with darker colors) are weighted more than those with lighter colors. So, the black-colored draws contribute most to the tilted distribution as they strongly align with the moment conditions ( $U3 = 4.2\%$ ,  $PCE = 2\%$ ). On the flip side, the yellow-colored draws are down-weighted as they are far from the moment conditions, contributing minimally to the tilted distribution.



*Figure 9: Joint MCMC draws of PCE inflation and unemployment rate for November 2027 in December 2024 categorized by weights. The size of each point proportionally changes with the weights optimized by entropic tilting so that the median of the tilted distribution anchors with the midpoint of the SEP's central tendencies.*

## 6. Summary and Discussion

In this paper, I detailed the procedure to implement a large BVAR model with covid volatility using the novel python’s `covbayesvar` package, which is an amalgamation of the works of Giannone Lenza and Primiceri (2015), Crump et. al (2021), and Lenza and Primiceri (2021). Whilst the authors use quarterly frequency data in these papers, I applied the model to the monthly frequency with medium-sized data, explained the functionalities of the most crucial Python functions, and conducted a Markov chain simulation exercise. Then, I presented use cases, namely unconditional and conditional forecasts where the model is subject to a linear combination of supply shocks in two variables - non-farm payroll employees, and hourly wage. To the best of my knowledge, this is the first open-source package which can estimate the model with COVID volatility and is versatile as it answers a breadth of policy questions via a reduced-form approach.

Nevertheless, we can extend it to solve a diverse array of questions. A simple extension is to use a similar BVAR with COVID volatility to construct an exchange rate index for the US, which, to my knowledge, no one has constructed before using a BVAR model. That entails gathering a larger set of international-oriented data on bilateral exchange rates, import and export price indices, trade balance of goods on a balance of payment basis, and trade data on US imports and exports of goods. Although I run the model for the US only, we can apply this model for more than one country, and enhance the efficiency by parallelizing the code using the `joblib` package, exploiting all CPU cores. This is feasible and will drastically reduce the time to run the model, and produce and store the results. An alternative is to deploy a cluster of computers to radically speed up the estimation process, such as SageMaker instances in Amazon Web Services.

## References

1. Banbura, Marta, Domenico Giannone, and Michele Lenza. “Conditional Forecasts and Scenario Analysis with Vector Autoregressions for Large Cross-Sections.” *International Journal of Forecasting* 31, no. 3 (August 2015): 739–56. <https://doi.org/10.1016/j.ijforecast.2014.08.013>.
2. Crump, Richard K., Stefano Eusepi, Domenico Giannone, Eric Qian, and Argia M. Sbordone. “A Large Bayesian VAR of the United States Economy.” *NY Fed Staff Report*. [https://www.newyorkfed.org/research/staff\\_reports/sr976](https://www.newyorkfed.org/research/staff_reports/sr976).
3. Lenza, Michele, and Giorgio Primiceri. “How to Estimate a VAR after March 2020.” *Journal of Applied Econometrics* 37, no. 4 (July 2022): 688–99. <https://doi.org/10.1002/jae.2895>.
4. Giannone, Domenico, Michele Lenza, and Giorgio E. Primiceri. “Prior Selection for Vector Autoregressions.” *The Review of Economics and Statistics* 97, no. 2 (May 2015): 436–51. [https://doi.org/10.1162/REST\\_a\\_00483](https://doi.org/10.1162/REST_a_00483).

## Appendix

### A1. Description of monthly macro and financial variables

Series Name	Units	Transformation	isFinancial	Prior
Industrial Production Index	Index 2017=100	log	0	RW
Capacity Utilization: Manufacturing	Percent of Capacity	lin	0	RW
Housing Starts	Thousands of Units	log	0	RW
Real Personal Income ex. Transfer Receipts	Billions of Chained 2012 Dollars	log	0	RW
Real Personal Consumption Expenditure	Index 2017=100	log	0	RW
Real Manufacturing and Trade Industries Sales	Millions of Chained 2017 Dollars	log	0	RW
All Employees, Total Nonfarm	Thousands of Persons	log	0	RW
Civilian Unemployment Rate	Percent	lin	0	RW
Average Hourly Earnings of Production and Non-supervisory Employees	Dollars per Hour, Monthly	log	0	RW
Initial Claims		log	0	RW
CPI: All Items	Index 1982–1984=100	log	0	RW
CPI-Urban: All Items Less Food and Energy	Index	log	0	RW
PCE: Chain-Type Price Index	Index 2012=100	log	0	RW
PCE Services: Chain-Type Price Index	Index	log	0	RW
PPI by Commodity: Metals and Metal Products	Index	log	1	RW
Crude Oil, spliced WTI and Cushing	Dollars per Barrel	log	1	RW
10-Year Treasury Note Yield	Percent	lin	1	RW
1 Year Treasury Bond Yield	Percent	lin	1	RW
5 Year Treasury Bond Yield	Percent	lin	1	RW
Federal Funds Rate	Percent	lin	0	RW
Moody Seasoned Aaa Corporate Bond Yield	Percent	lin	1	RW
Moody Seasoned Baa Corporate Bond Yield	Percent	lin	1	RW
M2 Money Stock	Billions of Dollars	log	1	RW
S&P 500 Index	Index	log	1	RW
CBOE Volatility Index: VIX	Index	log	1	WN
Japan / U.S. Foreign Exchange Rate		log	1	RW
U.S. / U.K. Foreign Exchange Rate		log	1	RW
Canada / U.S. Foreign Exchange Rate		log	1	RW