

Tenuous Relations and Timeseries Analysis

Nathaniel Forde
Data Science @ Personio

Agenda

1. Motivation: Background Problem
2. Structural Autoregressive Models
3. Structural Autoregressive Models in PYMC
4. Vector Autoregressive Models
5. Bayesian Vector Autoregressive Models in PYMC
6. Hierarchical BVARs in PYMC

Motivation: Background Problem

Disclaimers and Acknowledgements



Note

No Personio data used in this presentation. The problem discussed motivated the research, but the models presented below will use fake or public data.

We will be discussing the problem of App performance measurements and customer feedback metrics.

How do they influence one another and how to best distil the relationship between the two?



Note

Acknowledgements: The work on the BVAR discussed here took inspiration and borrowed heavily from the PYMC labs blogpost [here](#). The example PRs benefited from the feedback of the PYMC devs.



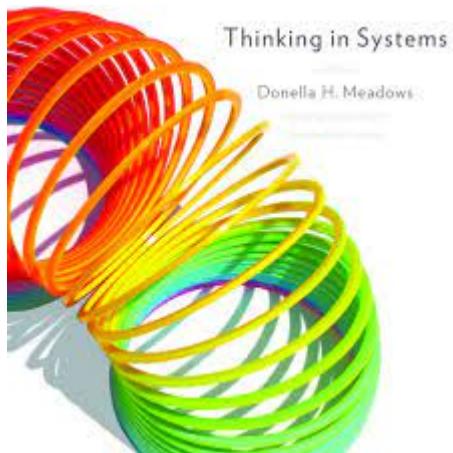
Warning

Not an Economist!!!

Systems and Feedback: App Performance

“Balancing feedback loops are equilibrating or goal seeking structures in systems and are both sources of stability and resistance to change”

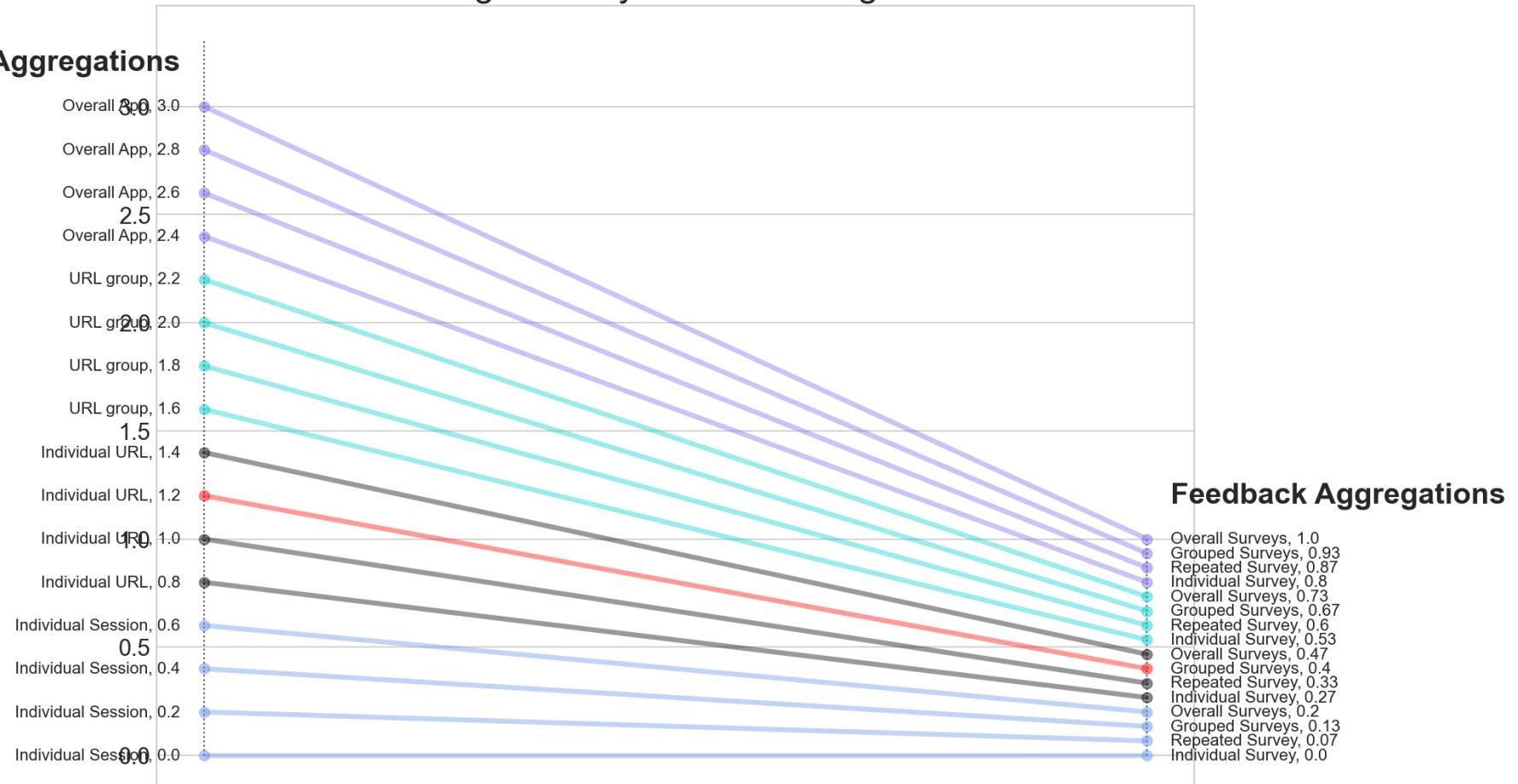
“Allowing performance standards to be influenced by past performance... sets up a reinforcing feedback of eroding goals that sets a system drifting toward low performance.”



Disentangling Individual effects

Assessing the Ways feedback is generated

App Performance Aggregations

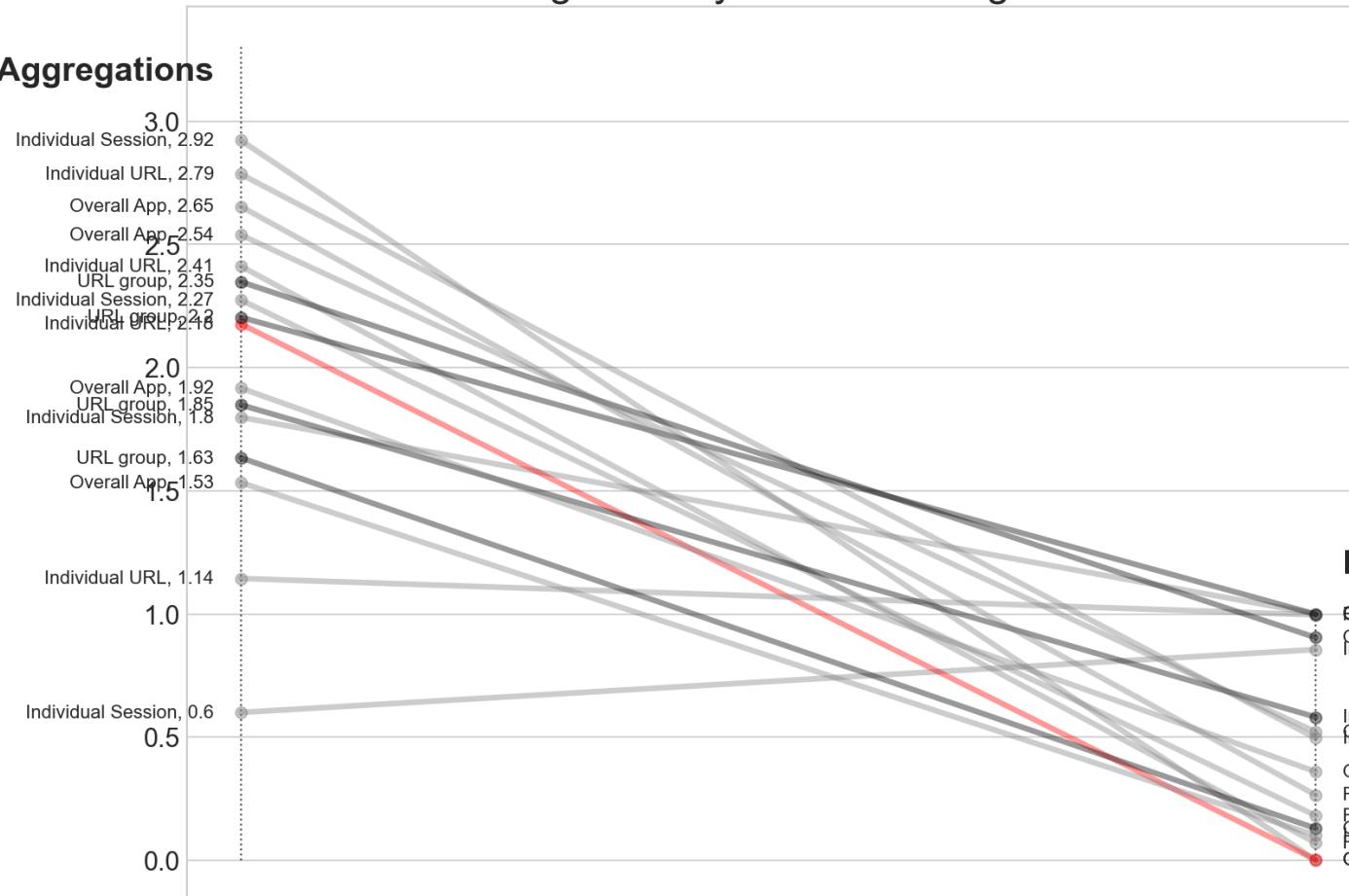


Identifying the Channel of Transmission

8
12

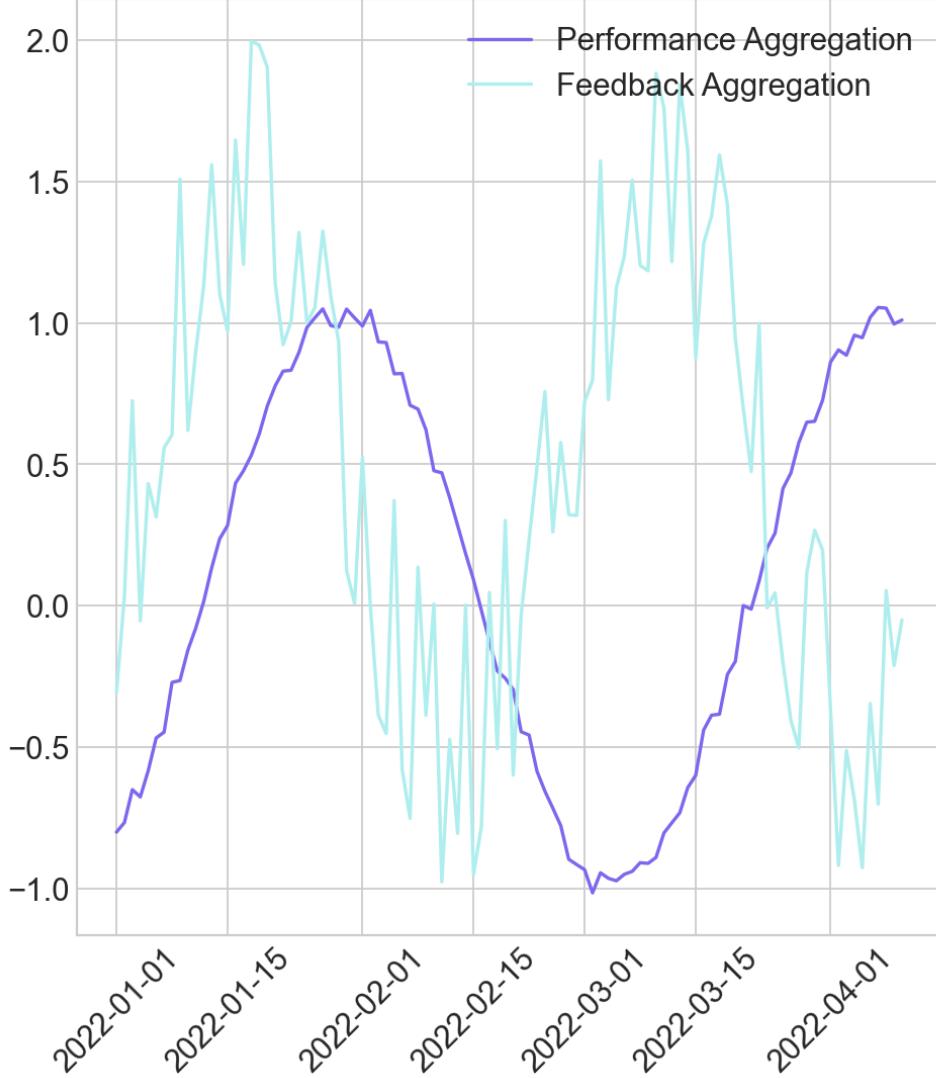
Assessing the Ways feedback is generated

App Performance Aggregations

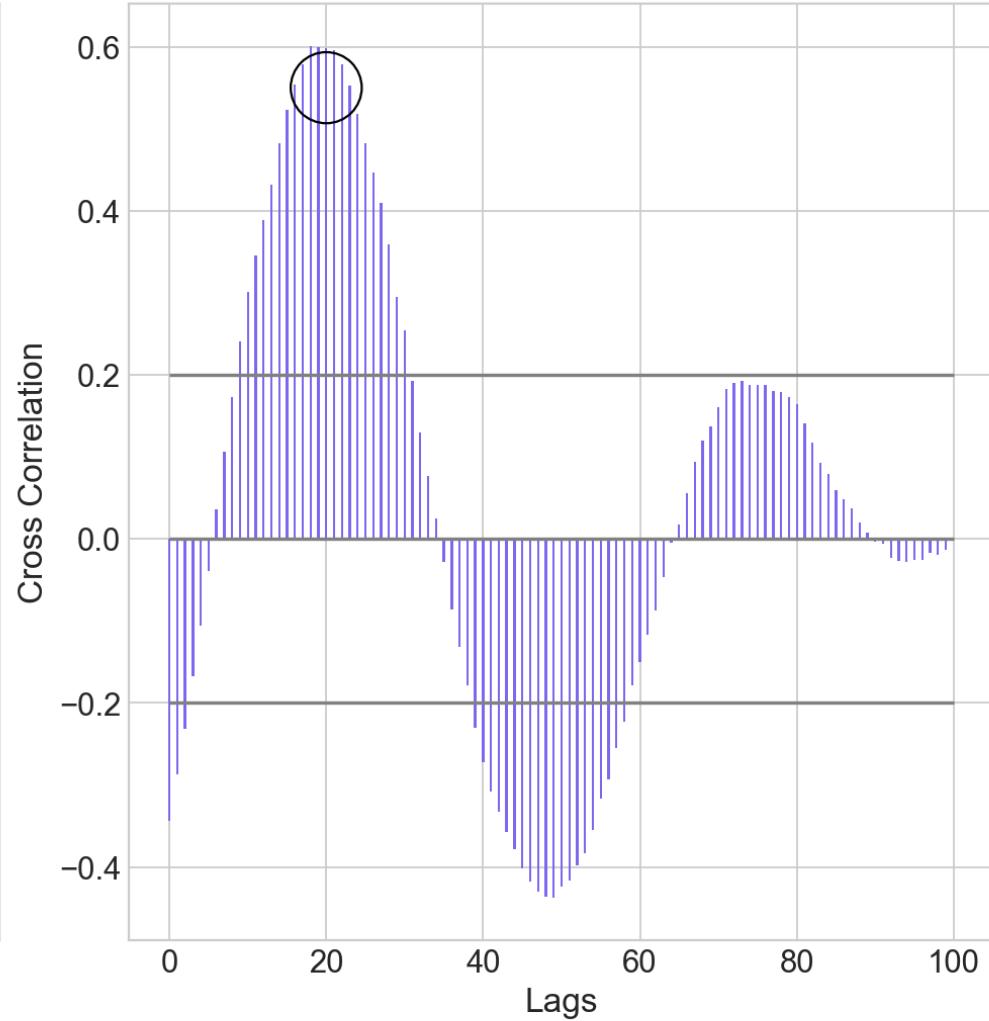


Lagging Effects in Time

9
1
2



Cross Correlation of Lagged terms between Performance and Feedback



Structural Autoregressive Models

Autogressive Models

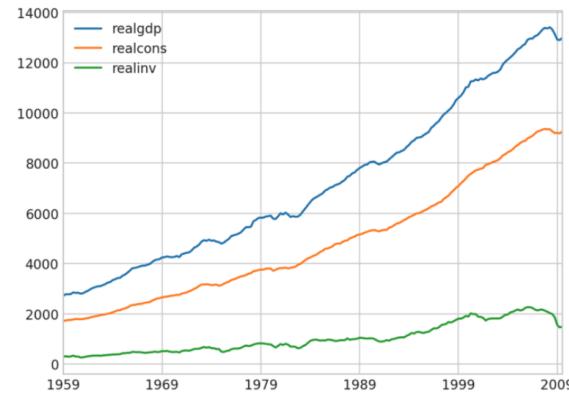
Moving Average Representation and Impulse Response

- The coefficients in: $Y_t = \mu + \sum_{j=0}^{\infty} b_j e_{t-j}$ are known as the impulse response function **IRF** representing the change in Y_{t+j} due to a shock at time t .

- $\frac{\partial}{\partial e_t} Y_{t+j} = b_j$

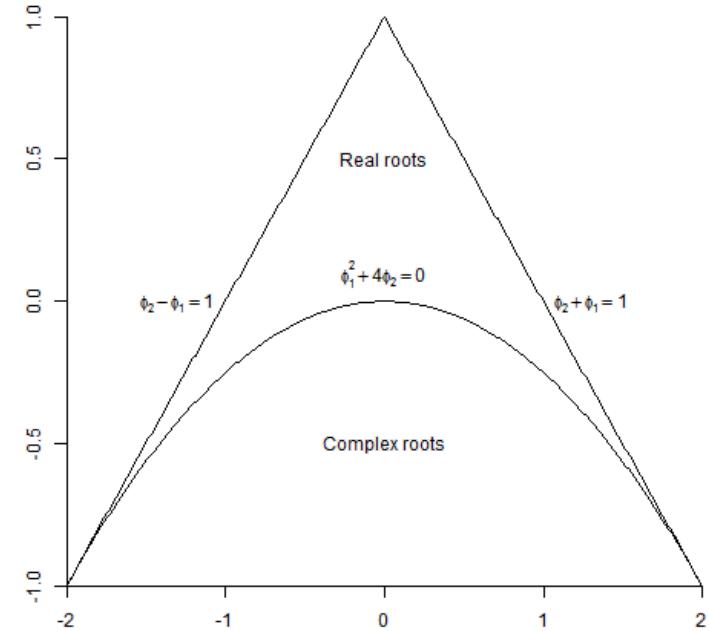
- They can be recovered from the AR representation by recursive reconstruction.
- IRFs are often reported when scaled by the standard deviation of shocks.

Stationarity Requirements



Tricks to achieve stationarity

Stationary conditions for AR(2) model



Conditions required for Stationarity

Structural Autoregressive Models

- Bayesian Structural Timeseries (BSTS) are often seen as an alternative to Auto-regressive models¹
- But BSTS models can also incorporate latent auto-regressive components.
- Benefits of going Bayesian:
 - Transparent understanding of the sources of uncertainty within your model and a capacity to introduce outside information by way of priors or other covariates.
 - Plausible counterfactual inference with posterior predictive distributions

¹ Nice post by Kim Larsen [here](#)

Characterising Structure

$$Y \sim TruncNorm(\mu, \sigma, 0, \infty)$$

$$\mu = AR_\mu + f(seasonality) + f(trend) + \dots$$

$$\sigma \sim InverseGamma(3, .5)$$

$$trend = \alpha + \beta \cdot T_i$$

$$\alpha \sim Normal(7, 1)$$

$$\beta \sim Normal(0.009, .1)$$

$$seasonality = FourierFeatures' \beta_{fourier}$$

$$\beta_{fourier} \sim Normal(0, 0.5)$$

$$AR_\mu \sim \mu_{ar} + a_1 \cdot Y_{t-1} + a_2 \cdot Y_{t-2} \dots + \gamma$$

Structural Autoregressive Models in PYMC

Bayesian Structural Timeseries

Priors

Additive Structural Components

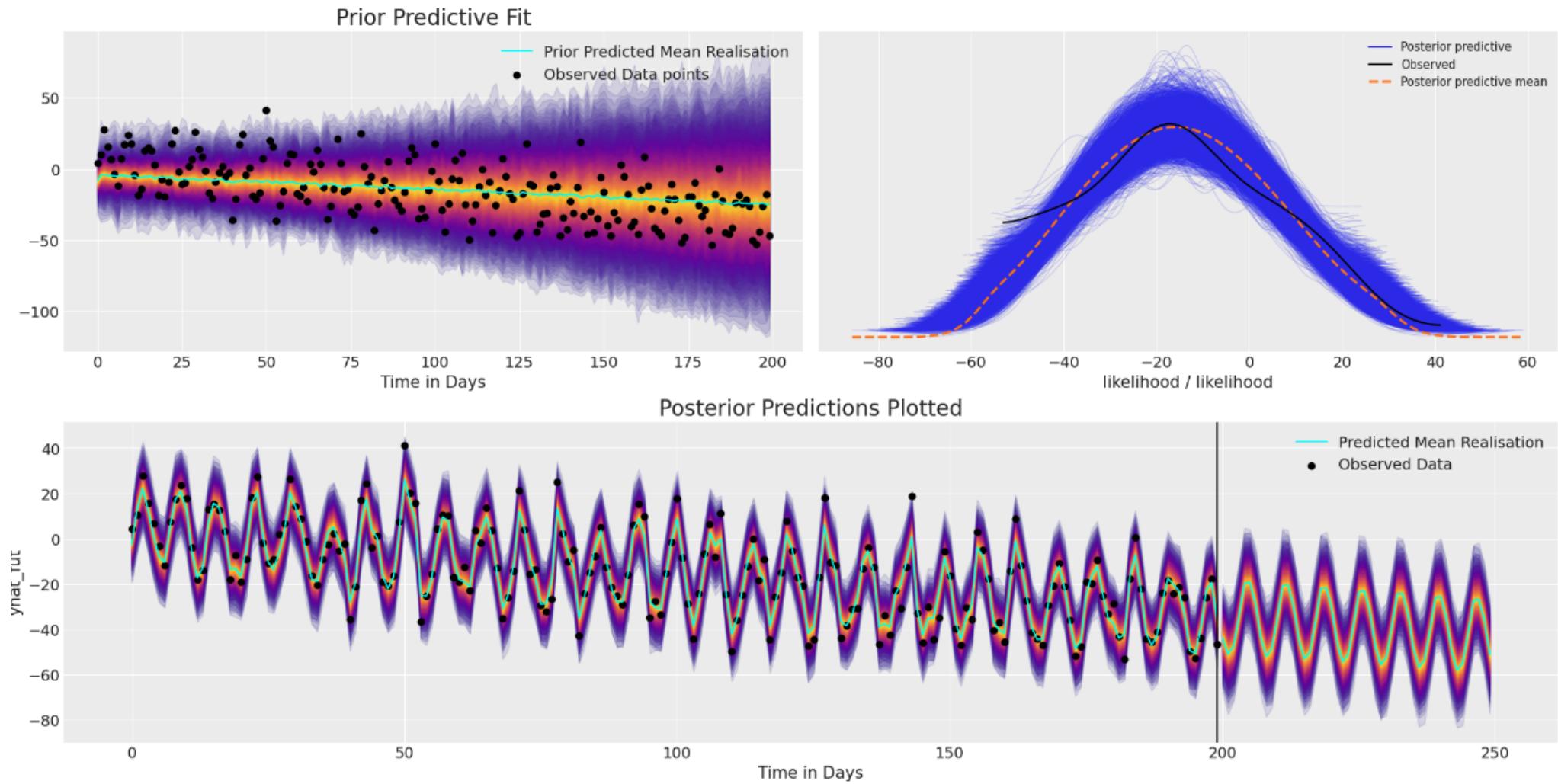
Likelihood

```

1
2 with AR:
3     ## Data containers to enable prediction
4     t = pm.MutableData("t", t_data, dims="obs_id")
5     y = pm.MutableData("y", ar_data, dims="obs_id")
6     # AR priors: The first coefficient will be the intercept term
7     coefs = pm.Normal("coefs", priors["coefs"]["mu"], priors["coefs"]["sigma"])
8     sigma = pm.HalfNormal("sigma", priors["sigma"])
9     ## Priors for the linear trend component
10    alpha = pm.Normal("alpha", priors["alpha"]["mu"], priors["alpha"]["sigma"])
11    beta = pm.Normal("beta", priors["beta"]["mu"], priors["beta"]["sigma"])
12    ## Priors for seasonality
13    beta_fourier = pm.Normal(
14        "beta_fourier",
15        mu=priors["beta_fourier"]["mu"],
16        sigma=priors["beta_fourier"]["sigma"],
17        dims="fourier_features",
18    )
19
20    # AR process: We need one init variable for each lag, hence size is variable too
21    init = pm.Normal.dist(
22        priors["init"]["mu"], priors["init"]["sigma"], size=priors["init"]["size"]
23    )
24    # Steps of the AR model minus the lags required given specification
25    ar1 = pm.AR(
26        "ar",
27        coefs,
28        sigma=sigma,

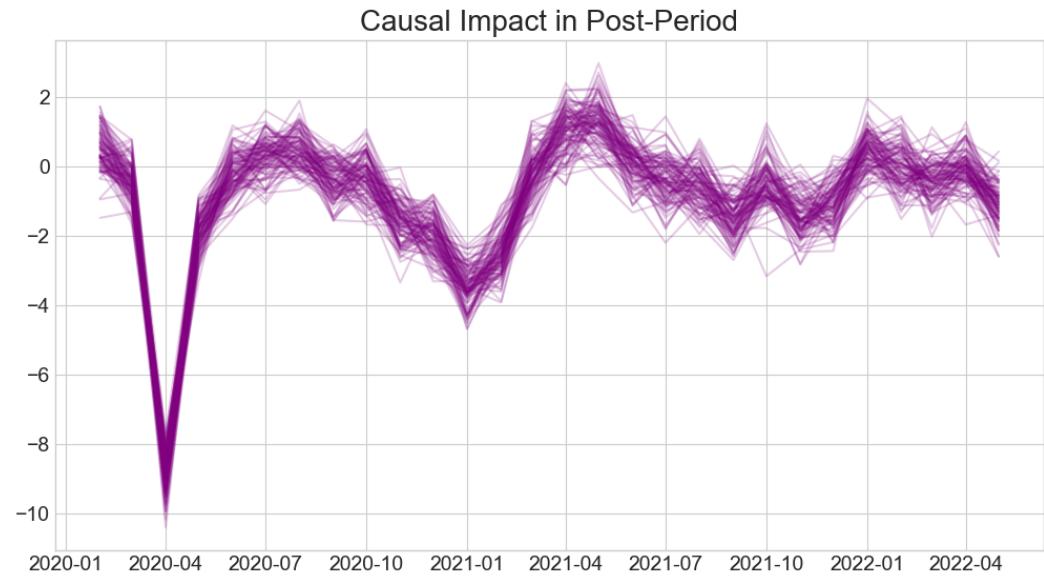
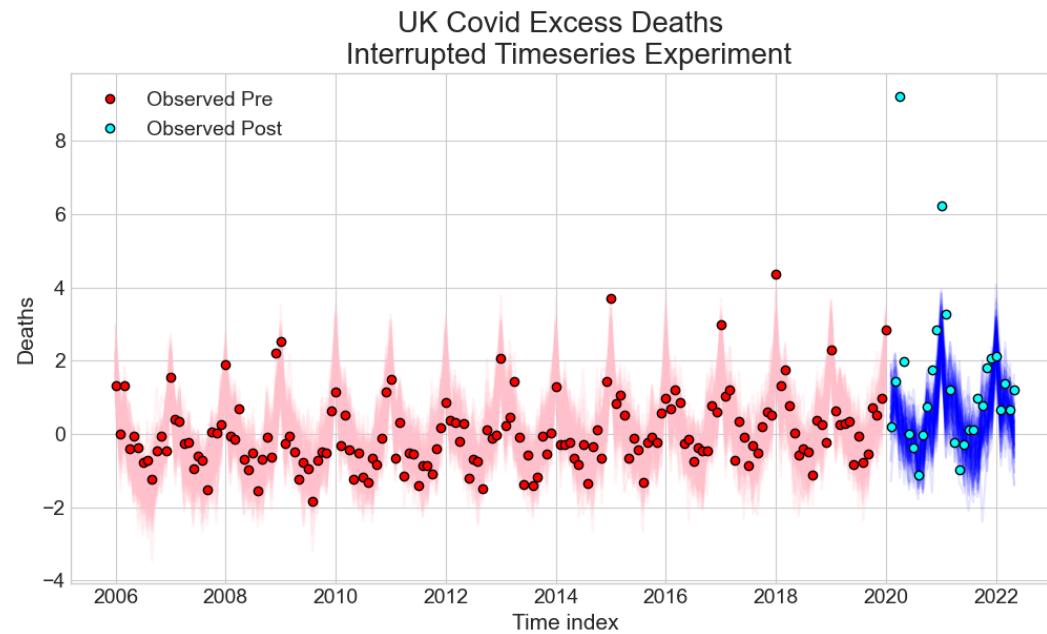
```

Prior Predictive and Posterior Predictive Distributions



Autoregressive BSTS prior and posterior fits. For details see PYMC docs [here](#)

Causal Inference with Interrupted Timeseries Designs



Causal Impact analysis for more detail see [CausalPy](#)

Vector Autoregressive Models

Multivariate Timeseries

- Multivariate Linear Projection Solution applies giving

■

$$\mathbf{y}_T = \nu + A_1 \mathbf{y}_{T-1} + A_2 \mathbf{y}_{T-2} \dots A_p \mathbf{y}_{T-p} + \mathbf{e}_t$$

- Wold Decomposition:

$$\mathbf{y}_T = \mu + \Omega_1 \mathbf{e}_{T-1} + \Omega_2 \mathbf{e}_{T-2} \dots \Omega_p \mathbf{e}_{T-p}$$

- Multivariate Auto-regressive representation with MV white noise error terms

$$\begin{bmatrix} gdp \\ inv \\ con \end{bmatrix}_T = \nu + A_1 \begin{bmatrix} gdp \\ inv \\ con \end{bmatrix}_{T-1} + A_2 \begin{bmatrix} gdp \\ inv \\ con \end{bmatrix}_{T-2} \dots A_p \begin{bmatrix} gdp \\ inv \\ con \end{bmatrix}_{T-p} + \mathbf{e}_t$$

VAR(2)

A VAR can have n lags and m equations such that the lagged terms of each series appear in all equations.

$$gdp_t = \beta_{gdp1} \cdot gdp_{t-1} + \beta_{gdp2} \cdot gdp_{t-2} + \beta_{cons1} \cdot cons_{t-1} + \beta_{cons2} \cdot cons_{t-2} + \epsilon_{gdp}$$

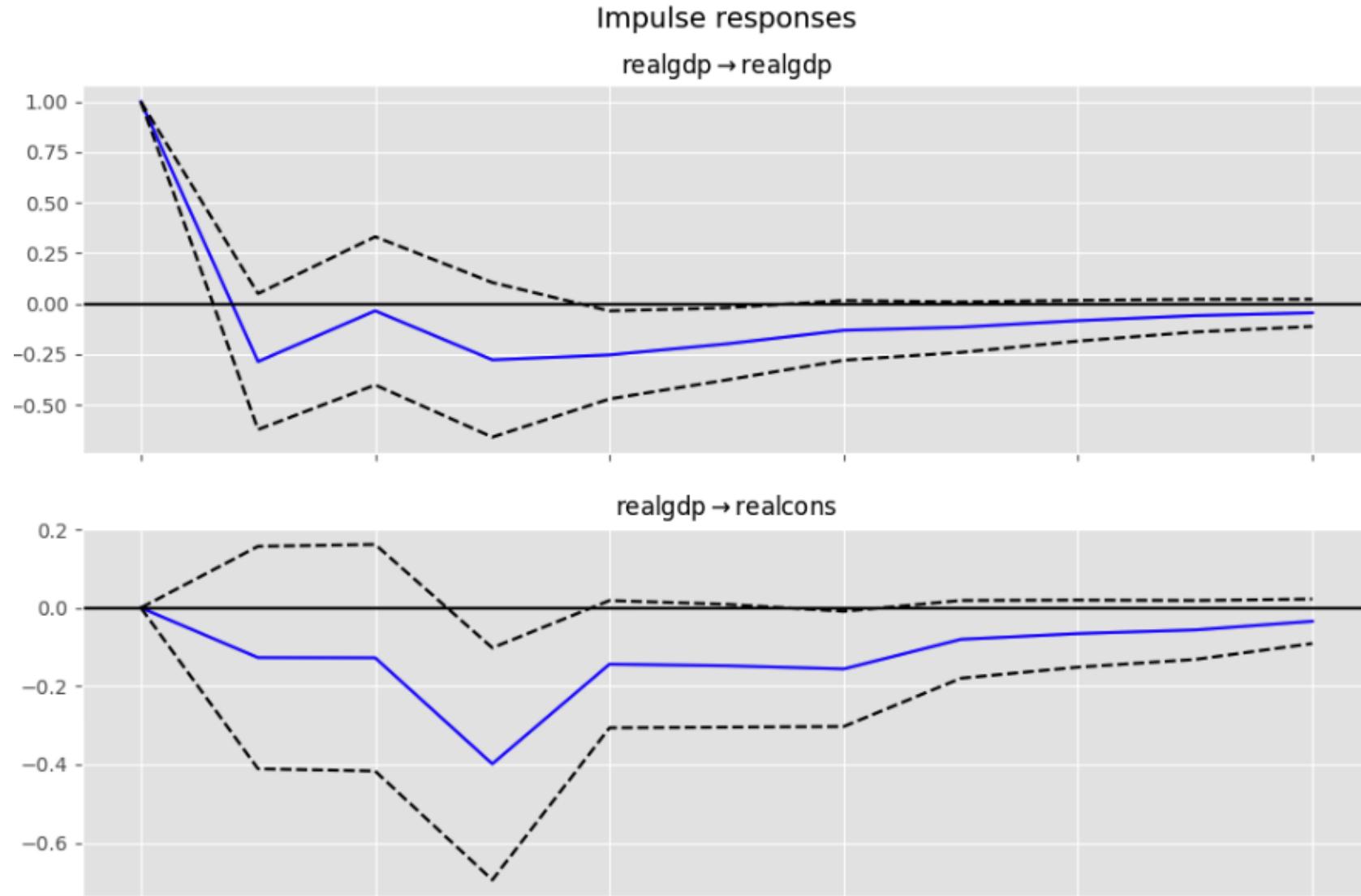
$$cons_t = \beta_{cons1} \cdot cons_{t-1} + \beta_{cons2} \cdot cons_{t-2} + \beta_{gdp1} \cdot gdp_{t-1} + \beta_{gdp2} \cdot gdp_{t-2} + \epsilon_{cons}$$

As such the **coefficients** on the cross-equation variables are of particular interest when investigating the influence of one series on another.

The MV IRF function can be created in an analogous way to the univariate timeseries function giving an interpretation of e.g the change in gdp_t due to a shock in $cons_{t-2}$.

This opens up the possibility of testing how influence between variables percolates over time.

Impulse Response



Statsmodels implementation of IRF

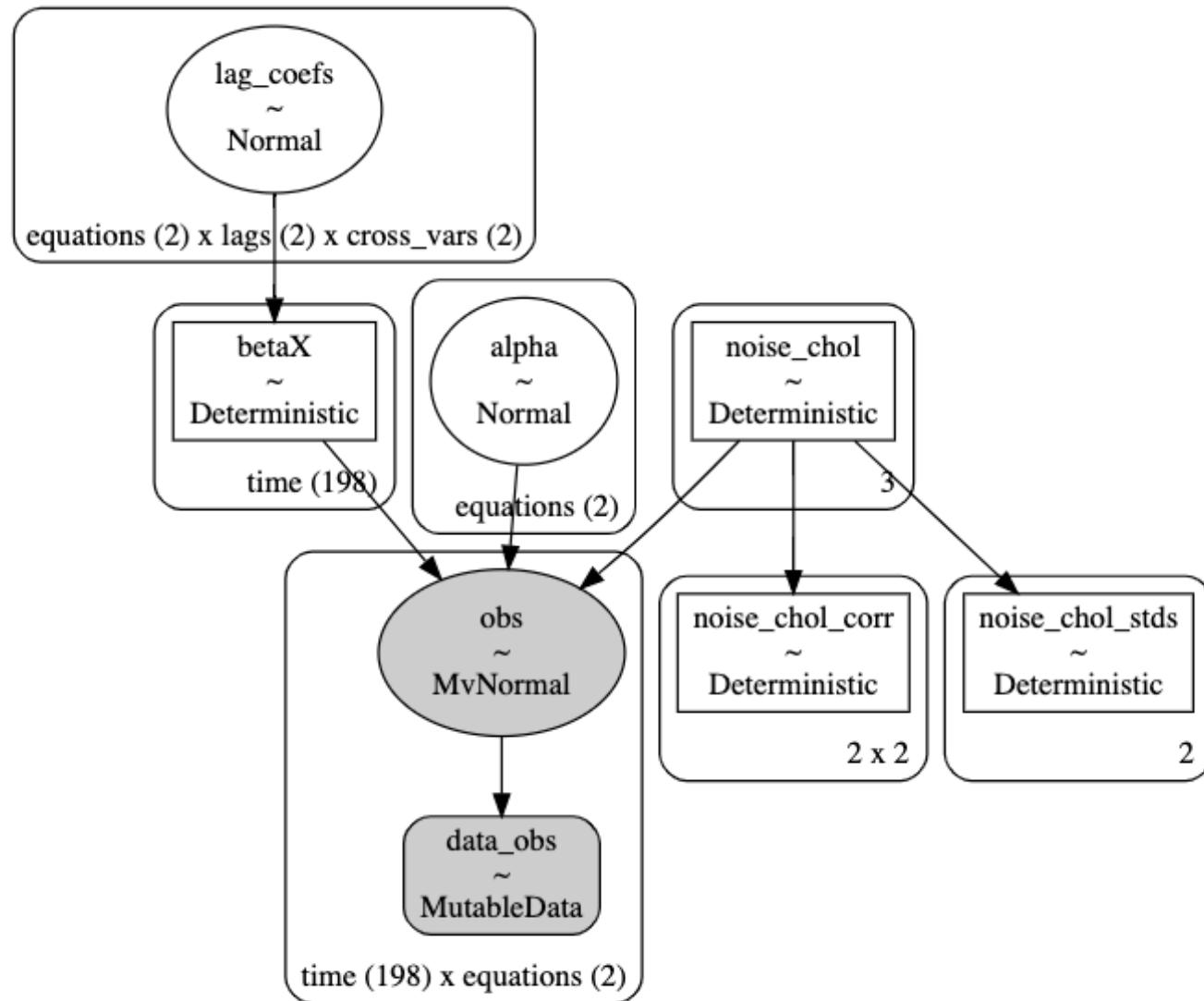
Vector Autoregressive Models in PYMC

Arranging the Coefficient Matrices

- There are allot of indices and variables in a VAR model.
- They can be concisely expressed in matrix notation, but it's not intuitive (to me) programatically.

```
1  ### Define a helper function that will construct our autoregressive step for the marginal contribution of each lagged
2  ### term in each of the respective time series equations
3 def calc_ar_step(lag_coefs, n_eqs, n_lags, df):
4     ars = []
5     for j in range(n_eqs):
6         ar = pm.math.sum(
7             [
8                 pm.math.sum(lag_coefs[j, i] * df.values[n_lags - (i + 1) : -(i + 1)], axis=-1)
9                 for i in range(n_lags)
10            ],
11            axis=0,
12        )
13     ars.append(ar)
14     betaY = pm.math.stack(ars, axis=-1)
15
16 return betaY
```

The Broad Structure



VAR(2)

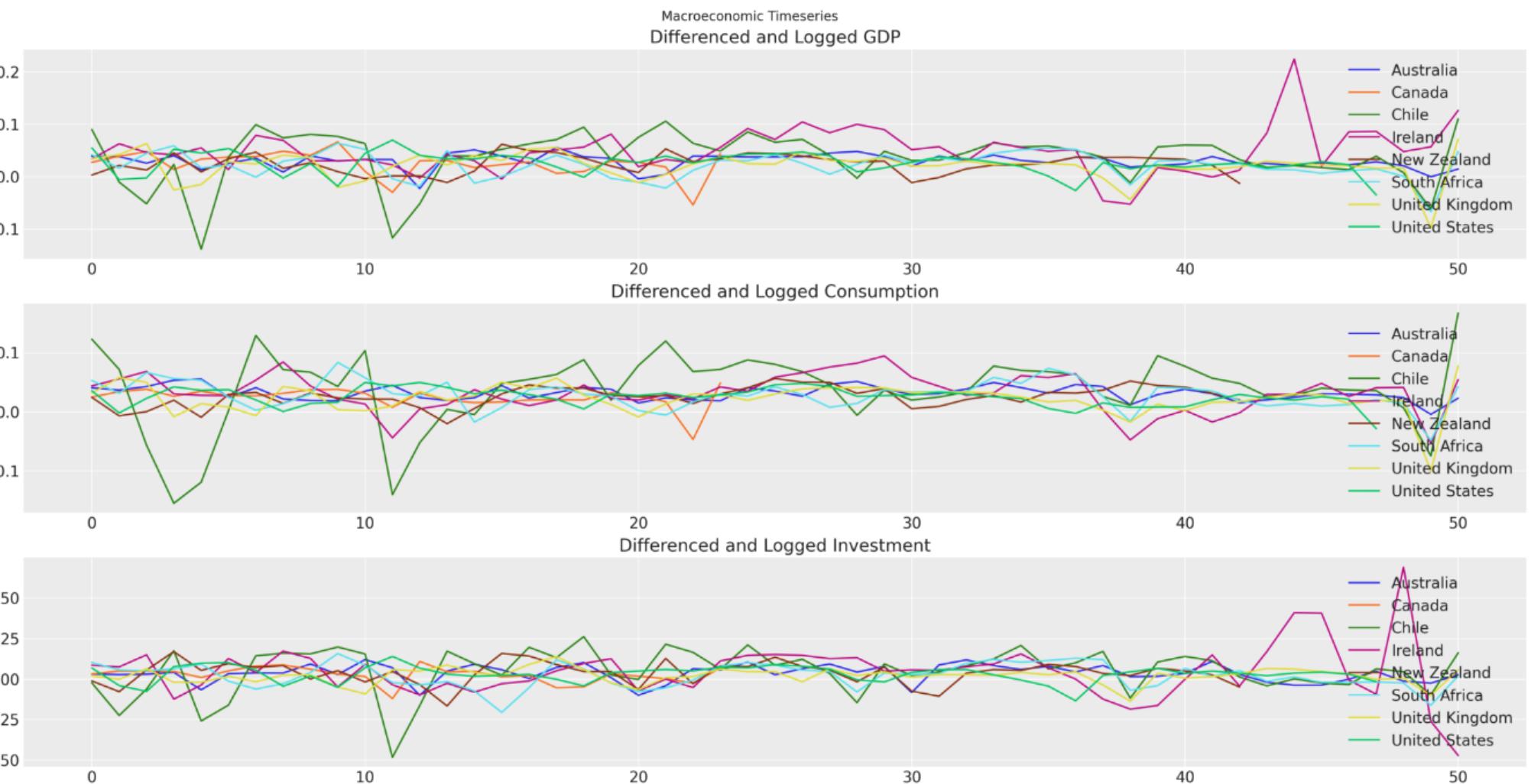
Implementation in Code

```

1 with pm.Model(coords=coords) as model:
2     lag_coefs = pm.Normal(
3         "lag_coefs",
4         mu=priors["lag_coefs"]["mu"],
5         sigma=priors["lag_coefs"]["sigma"],
6         dims=["equations", "lags", "cross_vars"],
7     )
8     alpha = pm.Normal(
9         "alpha", mu=priors["alpha"]["mu"], sigma=priors["alpha"]["sigma"], dims=("equations",)
10    )
11    data_obs = pm.Data("data_obs", df.values[n_lags:], dims=["time", "equations"], mutable=True)
12
13    betaX = calc_ar_step(lag_coefs, n_eqs, n_lags, df)
14    betaX = pm.Deterministic("betaX", betaX, dims=["time",])
15    mean = alpha + betaX
16
17    if mv_norm:
18        n = df.shape[1]
19        ## Under the hood the LKJ prior will retain the correlation matrix too.
20        noise_chol, _, _ = pm.LKJCholeskyCov(
21            "noise_chol",
22            eta=priors["noise_chol"]["eta"],
23            n=n,
24            sd_dist=pm.HalfNormal.dist(sigma=priors["noise_chol"]["sigma"]),
25        )
26        obs = pm.MvNormal(
27            "obs", mu=mean, chol=noise_chol, observed=data_obs, dims=["time", "equations"]
28        )

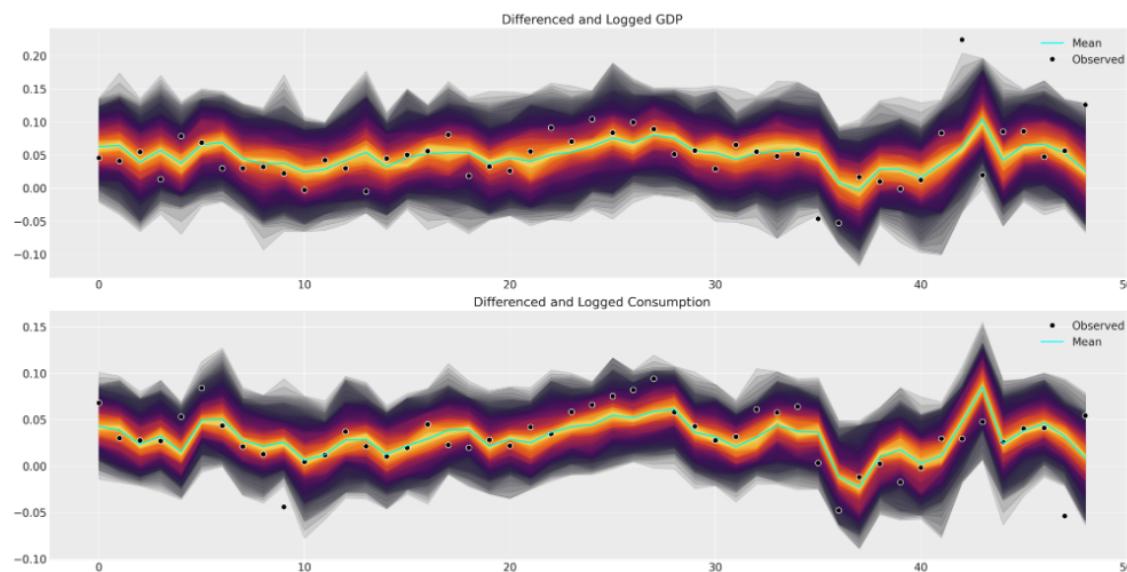
```

Applying the Theory



Macroeconomic Data

The Special Case of Ireland



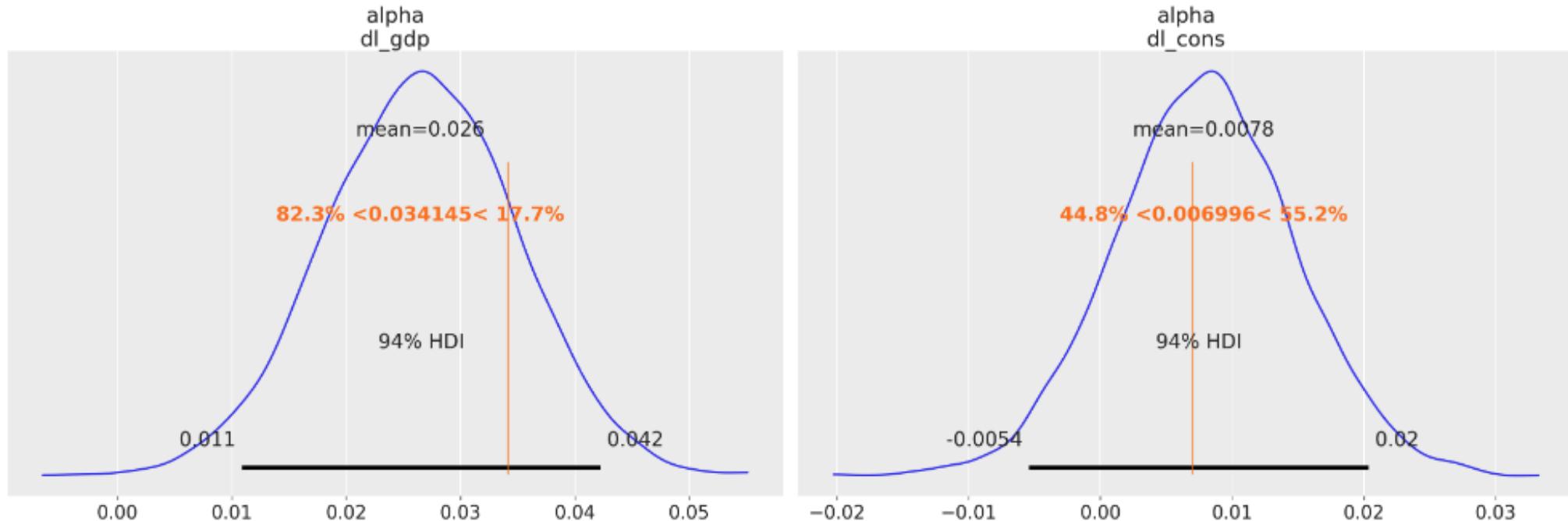
Ireland's Posterior Predictive Checks

Table 1: Posterior Mean Correlation

	GDP	CONS
GDP	1	0.43
CONS	0.43	1

Comparison with Statsmodels MLE fit

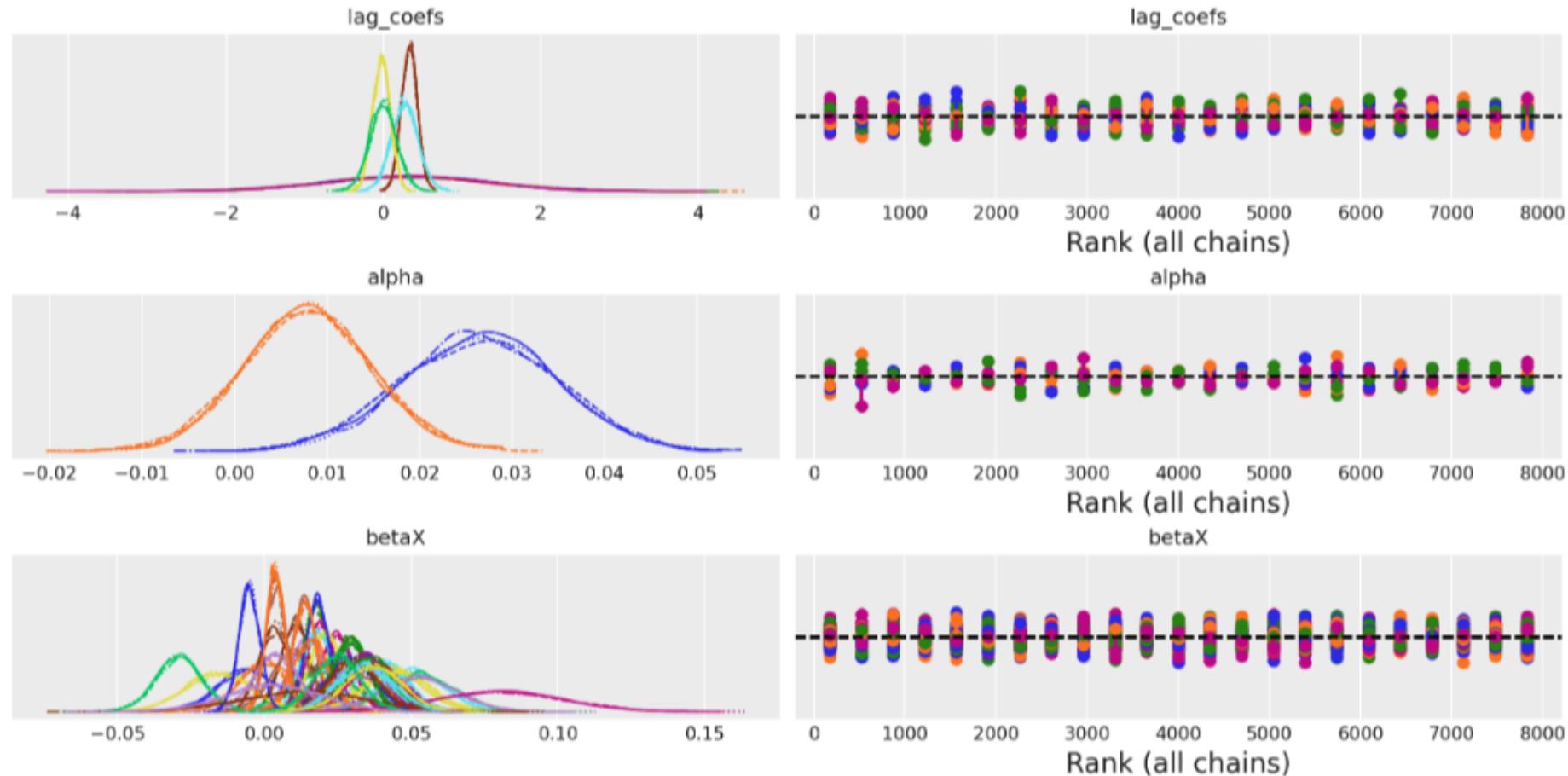
```
az.plot_posterior(idata_ireland, var_names=["alpha"], ref_val=[0.034145, 0.006996])
```



Compare

Convergence Checks

```
az.plot_trace(idata_ireland, var_names=["lag_coefs", "alpha", "betaX"], kind="rank_
```

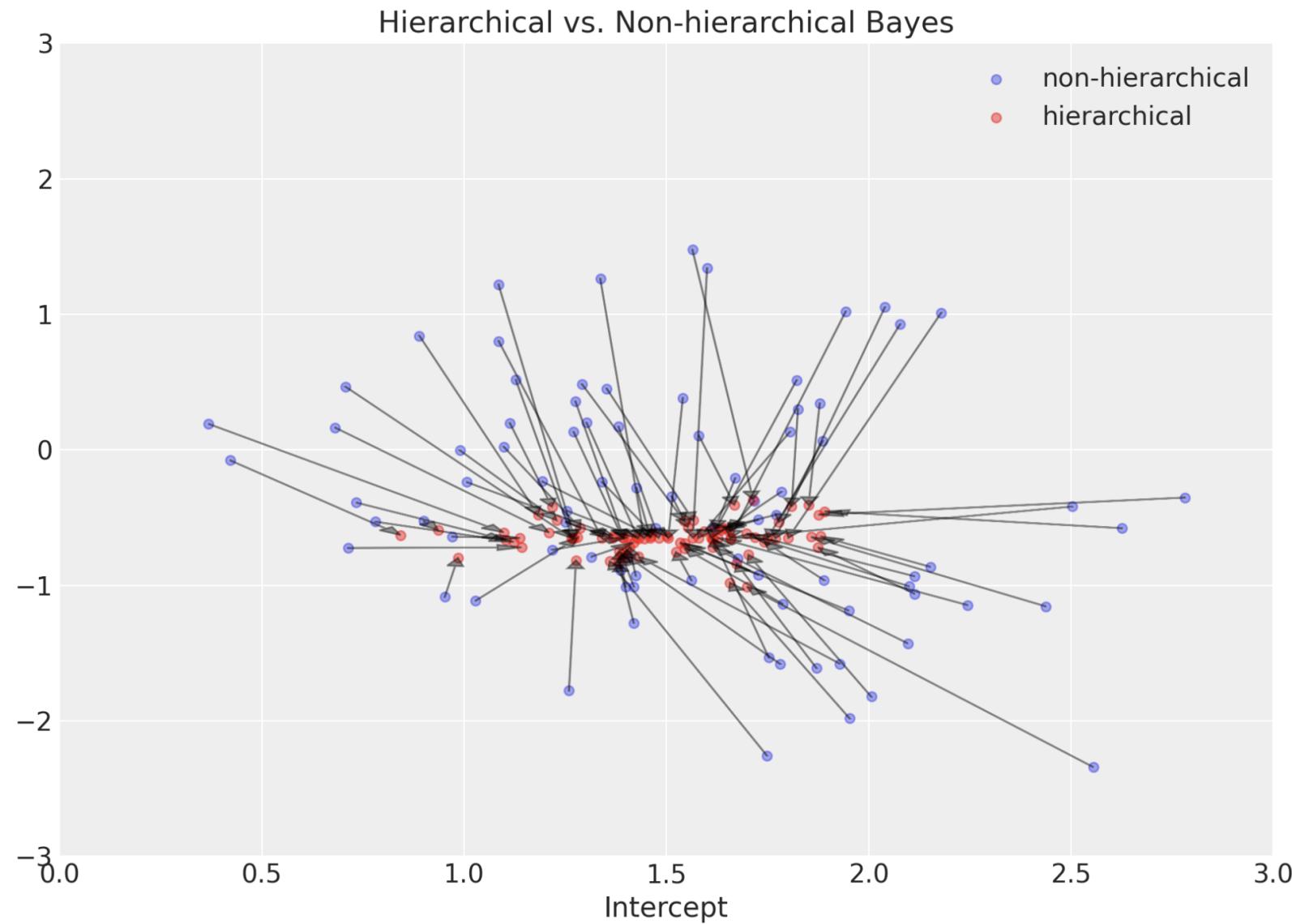


Sampling Chains

Hierarchical Bayesian VARs

“Those who only know one country, know no country” – Seymour Martin Lipset

Pooling, Shrinkage and Hierarchy



Shrinkage to Centre Effect

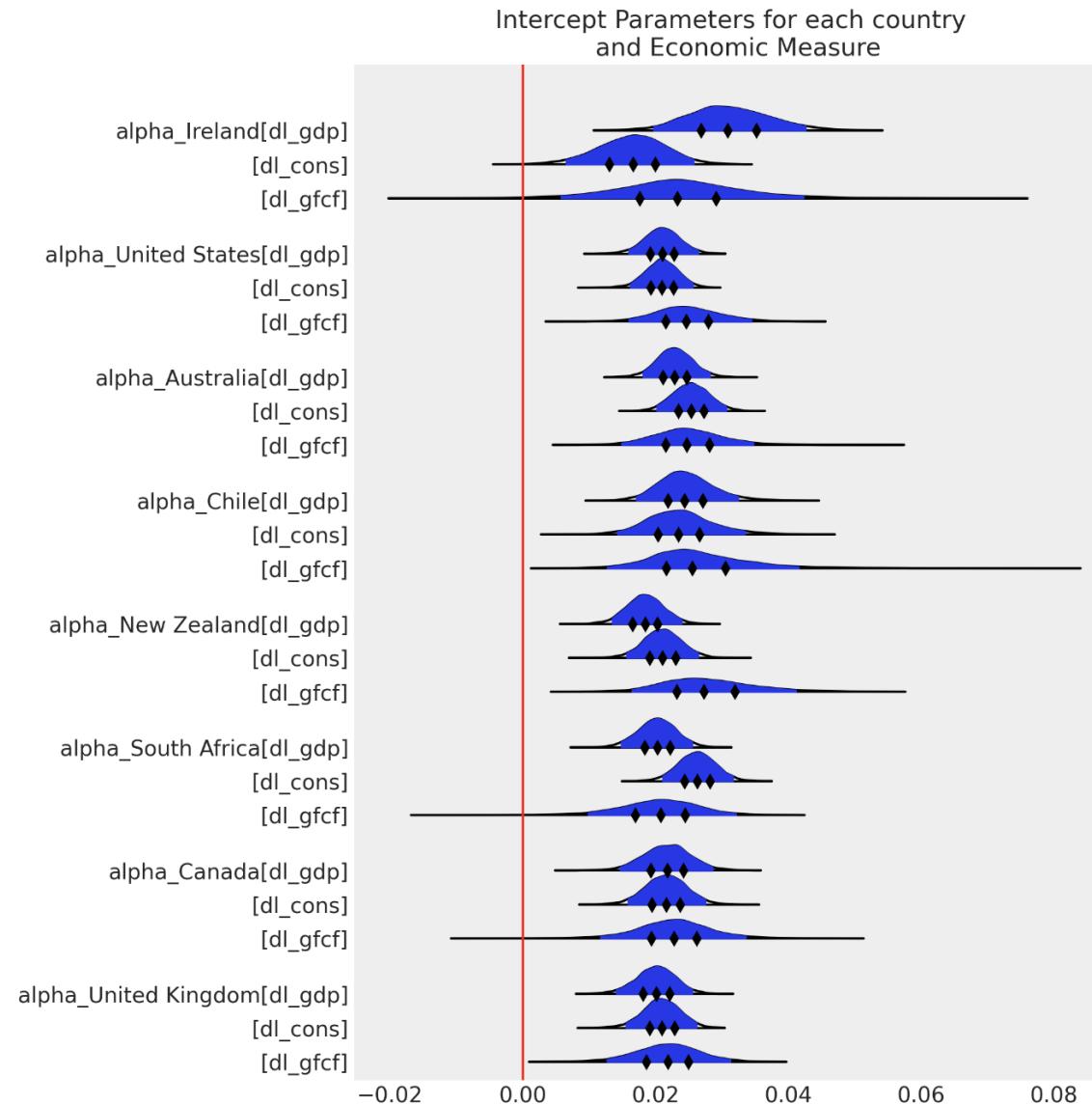
How to model Hierarchical Structure

```

1 groups = df[group_field].unique()
2
3     with pm.Model(coords=coords) as model:
4         ## Hierarchical Priors
5         rho = pm.Beta("rho", alpha=2, beta=2)
6         alpha_hat_location = pm.Normal("alpha_hat_location", 0, 0.1)
7         alpha_hat_scale = pm.InverseGamma("alpha_hat_scale", 3, 0.5)
8         beta_hat_location = pm.Normal("beta_hat_location", 0, 0.1)
9         beta_hat_scale = pm.InverseGamma("beta_hat_scale", 3, 0.5)
10        omega_global, _, _ = pm.LKJCholeskyCov(
11            "omega_global", n=n_eqs, eta=1.0, sd_dist=pm.Exponential.dist(1)
12        )
13        ## Group Specific Parameter
14        for grp in groups:
15            df_grp = df[df[group_field] == grp][cols]
16            z_scale_beta = pm.InverseGamma(f"z_scale_beta_{grp}", 3, 0.5)
17            z_scale_alpha = pm.InverseGamma(f"z_scale_alpha_{grp}", 3, 0.5)
18            lag_coefs = pm.Normal(
19                f"lag_coefs_{grp}",
20                mu=beta_hat_location,
21                # Hierarchical offset
22                sigma=beta_hat_scale * z_scale_beta,
23                dims=["equations", "lags", "cross_vars"],
24            )
25            alpha = pm.Normal(
26                f"alpha_{grp}",
27                mu=alpha_hat_location,
28                # Hierarchical Offset

```

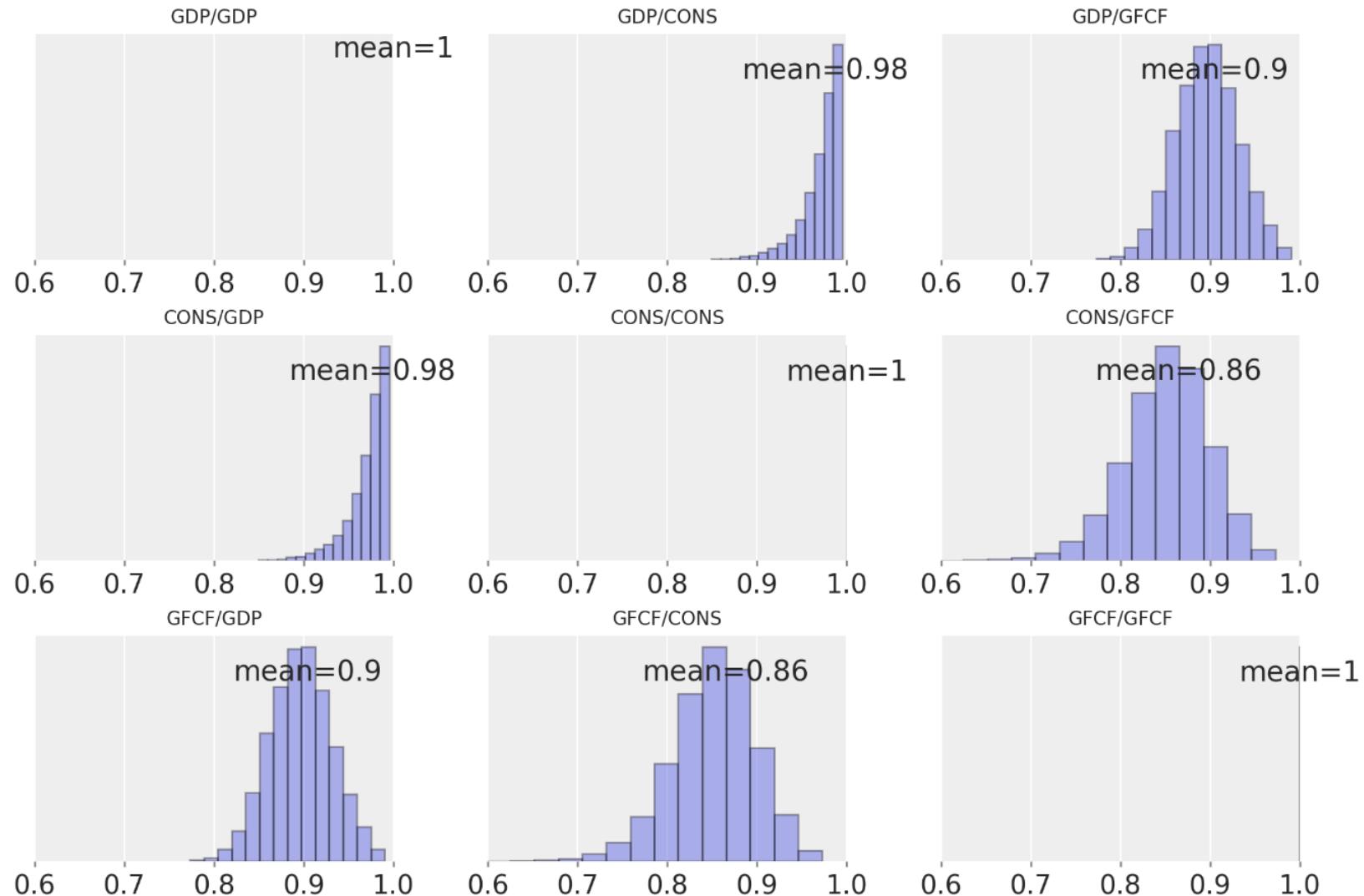
Ireland is an Outlier



Country Specific Estimates

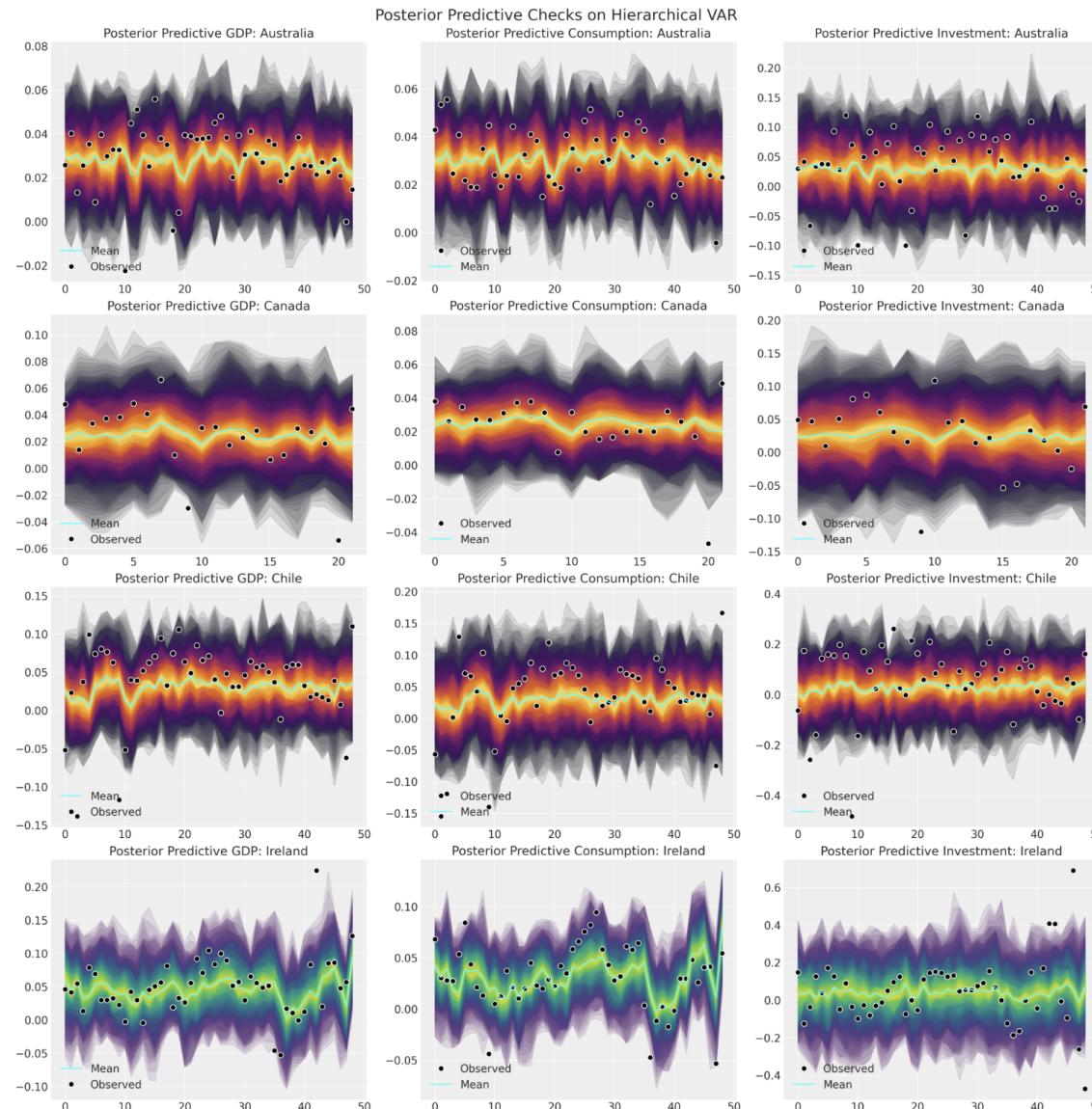
The Global Correlation Structure

The Posterior Correlation Estimates



Global Correlations

The Posterior Predictive Fits



Posterior Predictive Fits by country. See PYMC docs [here](#)

Wrap Up

- Bayesian Timeseries models are flexible and transparent tools for forecasting and causal inference.
- VAR models can help interrogate questions of the relationships between timeseries data.
- Bayesian VAR models allow the specification of regularising priors and encode structural assumptions about direction of influence.
- Hierarchical Bayesian VAR models offer the chance to borrow information across groups and interrogate the questions about the relationships between timeseries within and across the groups.
- In the context with sparse or short timeseries the hierarchical model can augment our understanding of the lagged relationships of influence.