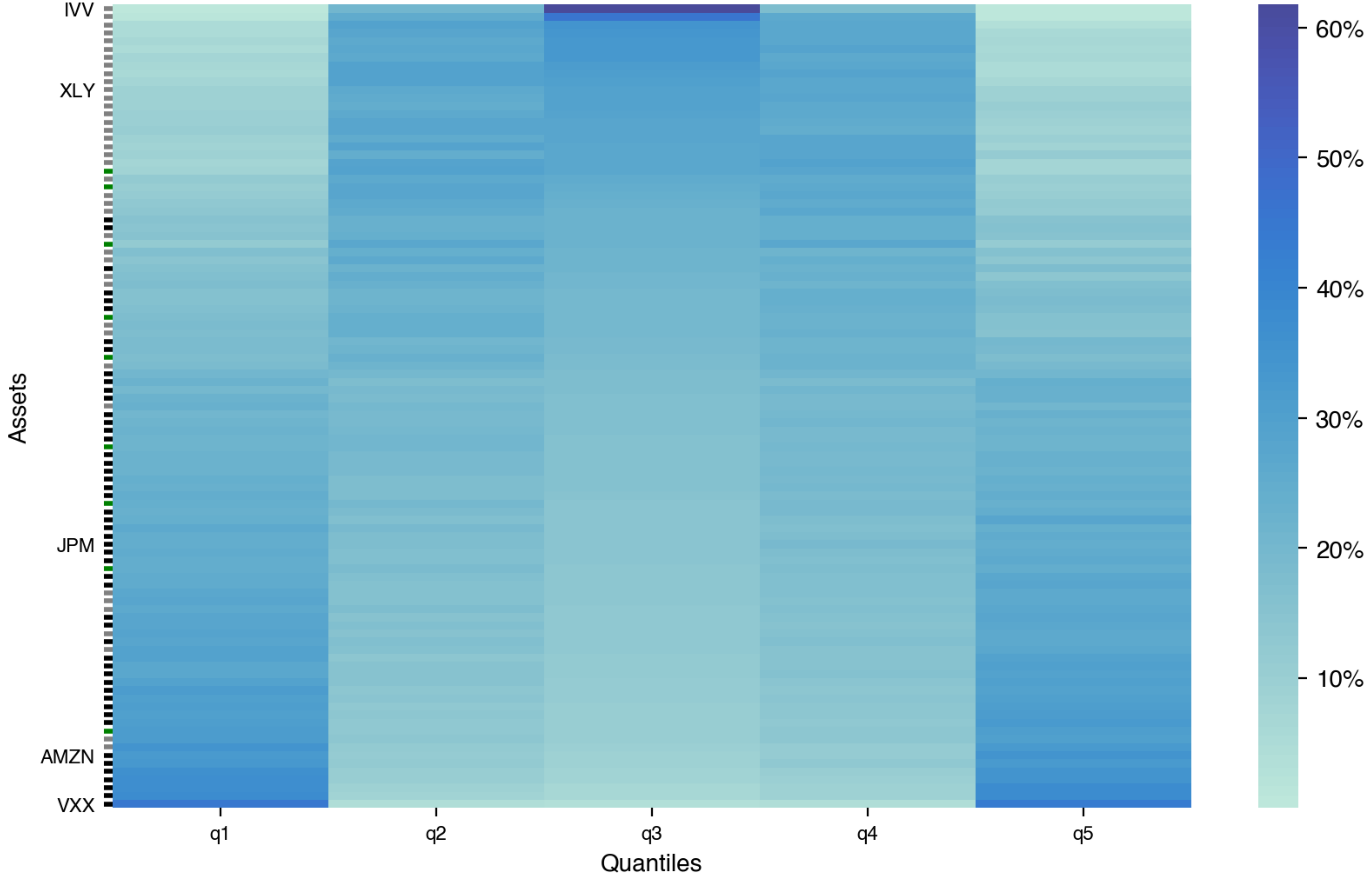
My M6 Model: A Bayesian Dynamic Factor Model with Heteroskedasticity

Dan Weitzenfeld, 11/07/2023

M6 Forecasting component



Agenda

- Thought process
- My model: Bayesian dynamic factor model with heteroskedasticity
- Pros and cons of my approach
- Future work

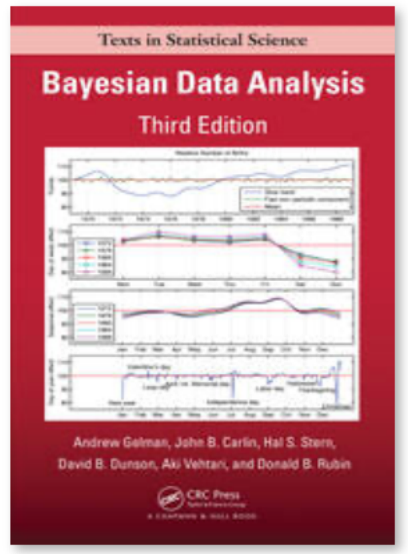
Agenda

- **Thought process**
- My model: Bayesian dynamic factor model with heteroskedasticity
- Pros and cons of my approach
- Future work

Thought process

- Forecasting component would come down to correctly modeling covariance and volatility, not picking winners and losers
- Probabilistic forecasting calls for Bayesian methods

"The essential characteristic of Bayesian methods is their **explicit use of probability** for quantifying uncertainty in inferences based on statistical data analysis."



Gelman, A., Carlin, J.B., Stern, H.S., Dunson, D.B., Vehtari, A., & Rubin, D.B. (2013). Bayesian Data Analysis (3rd ed.). Chapman and Hall/CRC. https://doi.org/10.1201/b16018

Probabilistic Forecasting: My Decision Tree

Can I build a model of the data generating process?

Yes → Do I have time to run MCMC?

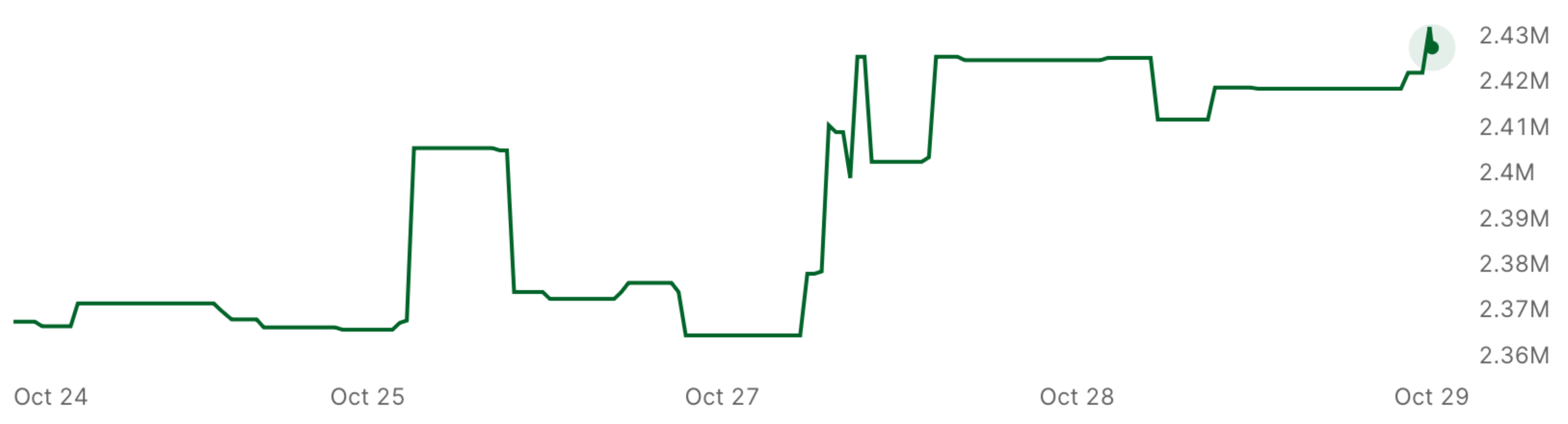
- Yes → **Bayesian Models**
  Implemented in pymc
- No → **Mixture Density Networks**
  Implemented in Keras

No → **Mixture Density Networks**

TSA check-ins

2427725.72 forecast ↑59,999.4

Forecast



Above 2.2 million		Yes --¢	No 5¢
Above 2.3 million	99¢	Yes --¢	No 2¢
Above 2.4 million	80¢ ↑16	Yes 90¢	No 30¢
Above 2.5 million	1¢	Yes 2¢	No 99¢

Kalshi

S&P500 daily bracket[1]

4115.00 ↓22.23 (0.54%)

4,075 to 4,099.99	25¢	Yes 35¢	No 75¢
4,100 to 4,124.99	25¢	Yes 45¢	No 75¢
4,125 to 4,149.99	25¢	Yes 38¢	No 75¢
4,150 to 4,174.99	10¢	Yes 99¢	No 99¢

Schematic of the Data Generating Process

Monthly return = Expected value + Noise

Asset Intercept

Dynamic Factor Model

+ Offset for recent returns

Expected value

Bayesian dynamic factor model…

Asset Intercept

Offset for recent volatility

+ Offset for earnings month

Noise Magnitude

… with heteroskedasticity

What is probabilistic programming?

“A **probabilistic programming language** is a high-level language that makes it easy for a developer to **define probability models** and then **“solve” these models automatically.**”

Cronin, Beau. “What is probabilistic programming?” O’Reilly, 2013, https://www.oreilly.com/content/probabilistic-programming/

Agenda

- Thought process
- **My model: Bayesian dynamic factor model with heteroskedasticity**
- Pros and cons of my approach
- Future work

Schematic of the Data Generating Process

Monthly return = Expected value + Noise

Bayesian dynamic factor model…	… with heteroskedasticity
Asset Intercept	Asset Intercept
Dynamic Factor Model	Offset for recent volatility
+ Offset for recent returns	+ Offset for earnings month
Expected value	Noise Magnitude

Bayesian dynamic factor model…

… with heteroskedasticity

Math

i	assets
t	time (months)
k	factors
c	asset class

$$r_{i,t} = \mu_{i,t} + \epsilon_{i,t}$$

Monthly return = Expected value + Noise

$$\mu_{i,t} = \alpha_i + \sum_{k=1}^{K} w_{k,i} * \lambda_{k,t} + \beta_{recent\_returns_i} * X_{recent\_returns_{i,t}}$$

Asset Intercept + Dynamic Factor Model + Offset for recent returns

$$\epsilon_{i,t} \sim t(0, \sigma_{i,t}, \nu_i)$$

$$log(\sigma_{i,t}) = \theta_i + \beta_{recent\_vol_i} * X_{recent\_vol_t} + \beta_{earnings_i} * I_{earnings_t}$$

Asset Intercept + Offset for recent volatility + Offset for earnings month

Factor loadings $w_{k,i} \sim t(\mu_w = 0, \sigma_w = 1, \nu_w = 10)$

Factor dynamics $\lambda_{k,t} = \rho * \lambda_{k,t-1} + \epsilon_\lambda$

Hierarchical Distributions

$$\alpha_i \sim t(\mu_\alpha = 0, \sigma_\alpha, \nu_\alpha = 10)$$

$$\beta_{recent\_returns_i} \sim t(\mu_{recent\_returns,c}, \sigma_{recent\_returns,c}, \nu = 10)$$

Hierarchical Distributions

$$\theta_i \sim t(\mu_{\theta,c}, \sigma_{\theta,c}, \nu_\theta)$$

$$\beta_{recent\_vol_i} \sim \mathcal{N}(\mu_{recent\_vol,c}, \sigma_{recent\_vol,c})$$

$$\beta_{earnings_i} \sim \mathcal{N}(\mu_{earnings}, \sigma_{earnings})$$

$$\nu_i \sim Gamma(\mu_\nu = 7, \sigma_\nu)$$

i assets
t time (months)
k factors
c asset class

Asset Intercept

$$\mu_{i,t} = \alpha_i + \sum_{k=1}^{K} w_{k,i} * \lambda_{k,t} + \beta_{recent\_returns_i} * X_{recent\_returns_{i,t}}$$

Asset Intercept + Dynamic Factor Model + Offset for recent returns

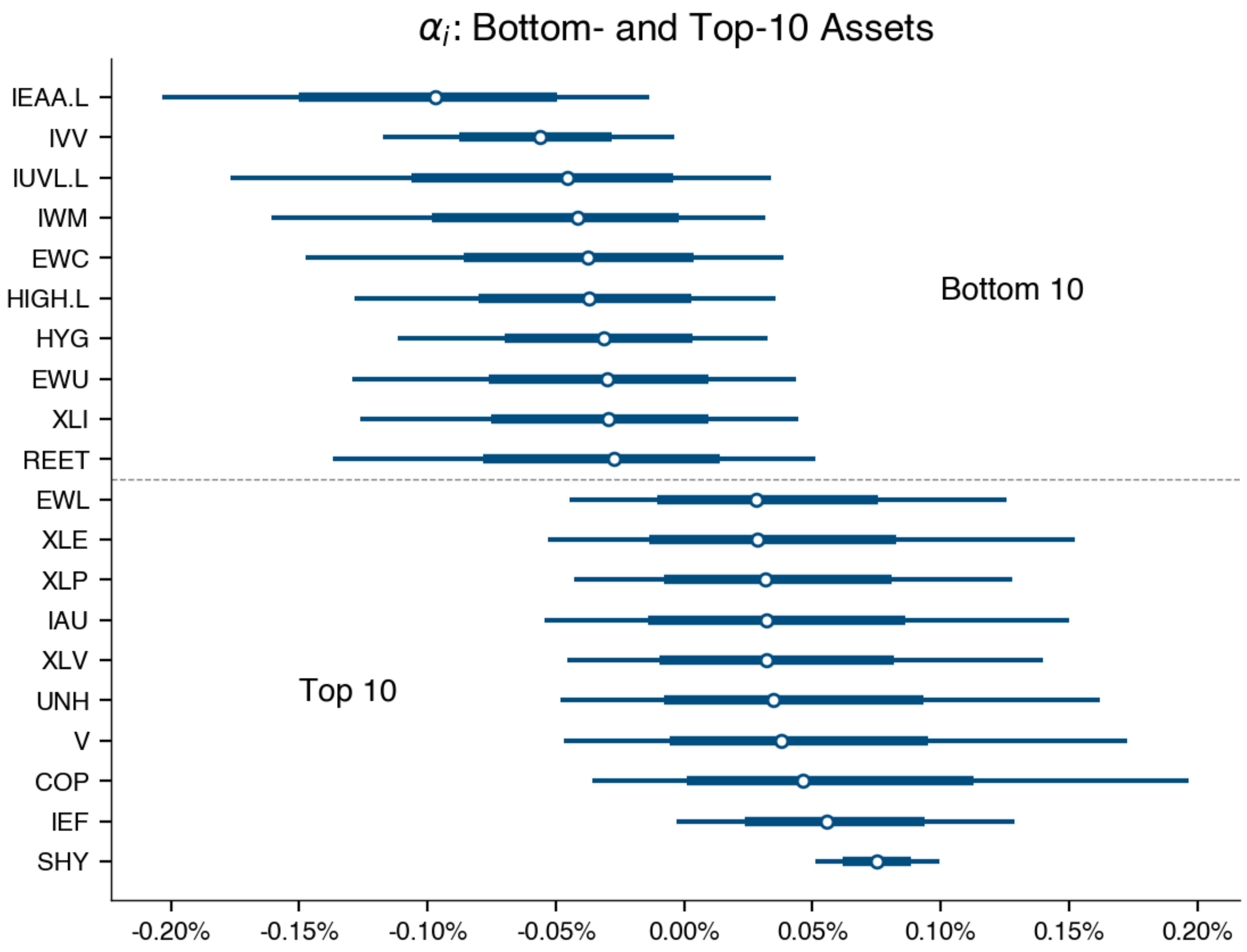
Factor loadings $w_{k,i} \sim t(\mu_w = 0, \sigma_w = 1, \nu_w = 10)$

Factor dynamics $\lambda_{k,t} = \rho * \lambda_{k,t-1} + \epsilon_\lambda$

Hierarchical Distributions

$$\alpha_i \sim t(\mu_\alpha = 0, \sigma_\alpha, \nu_\alpha = 10)$$

$$\beta_{recent\_returns_i} \sim t(\mu_{recent\_returns,c}, \sigma_{recent\_returns,c}, \nu = 10)$$



i	assets
t	time (months)
k	factors
c	asset class

Latent factors, not Fama-French

$$\mu_{i,t} = \alpha_i + \sum_{k=1}^{K} w_{k,i} * \lambda_{k,t} + \beta_{recent\_returns_i} * X_{recent\_returns_{i,t}}$$

Asset Intercept + Dynamic Factor Model + Offset for recent returns

Factor loadings $w_{k,i} \sim t(\mu_w = 0, \sigma_w = 1, \nu_w = 10)$

Factor dynamics $\lambda_{k,t} = \rho * \lambda_{k,t-1} + \epsilon_\lambda$

Hierarchical Distributions

$$\alpha_i \sim t(\mu_\alpha = 0, \sigma_\alpha, \nu_\alpha = 10)$$

$$\beta_{recent\_returns_i} \sim t(\mu_{recent\_returns,c}, \sigma_{recent\_returns,c}, \nu = 10)$$

- Latent factors, unobserved
- Fama-French factors:
  - Sort companies on a measurable dimension (size, book/market, profitability) and take the difference in returns
    - e.g. for size, Small Minus Big

i	assets
t	time (months)
k	factors
c	asset class

Factor dynamics: AR(1)

$$\mu_{i,t} = \alpha_i + \sum_{k=1}^{K} w_{k,i} * \lambda_{k,t} + \beta_{recent\_returns_i} * X_{recent\_returns_{i,t}}$$

Asset Intercept + Dynamic Factor Model + Offset for recent returns

Factor loadings $w_{k,i} \sim t(\mu_w = 0, \sigma_w = 1, \nu_w = 10)$

Factor dynamics $\lambda_{k,t} = \rho * \lambda_{k,t-1} + \epsilon_\lambda$

Hierarchical Distributions

$$\alpha_i \sim t(\mu_\alpha = 0, \sigma_\alpha, \nu_\alpha = 10)$$

$$\beta_{recent\_returns_i} \sim t(\mu_{recent\_returns,c}, \sigma_{recent\_returns,c}, \nu = 10)$$

$$\lambda_{k,t} = \underbrace{\rho * \lambda_{k,t-1}}_{\text{Short-term Reversal}} + \epsilon_\lambda$$

$$\lambda_{k,t} = \underbrace{\rho * \lambda_{k,t-1}}_{\text{Short-term Reversal}} + \underbrace{\beta * \sum_{j=2}^{12} \lambda_{k,t-j}}_{\text{Long-term Momentum}} + \epsilon_\lambda$$

7 factors in 2 dimensions

i assets
t time (months)
k factors
c asset class

$$\mu_{i,t} = \alpha_i + \sum_{k=1}^{K} w_{k,i} * \lambda_{k,t} + \beta_{recent\_returns_i} * X_{recent\_returns_{i,t}}$$

Asset Intercept + **Dynamic Factor Model** + Offset for recent returns

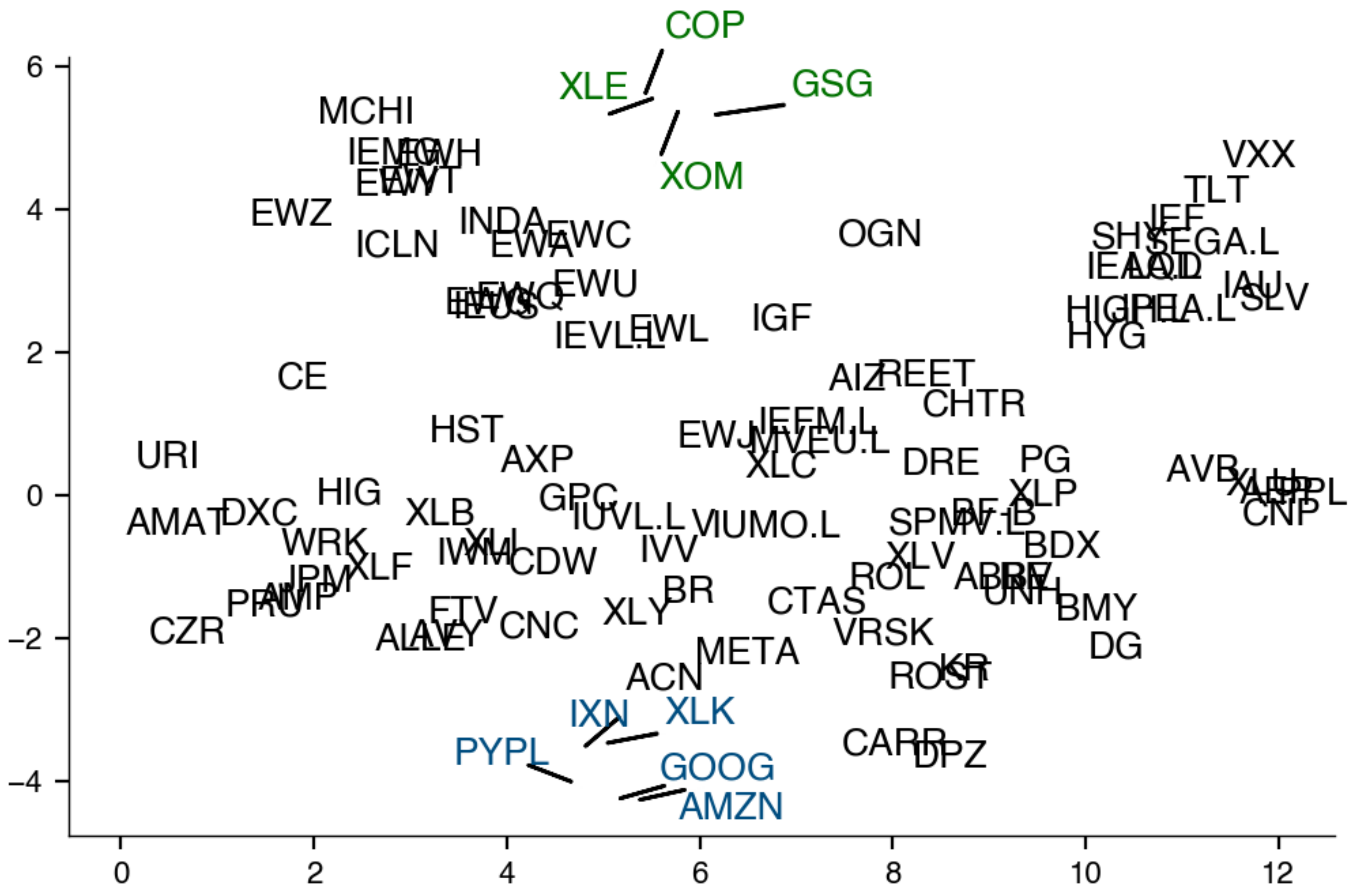
Factor loadings $w_{k,i} \sim t(\mu_w = 0, \sigma_w = 1, \nu_w = 10)$

Factor dynamics $\lambda_{k,t} = \rho * \lambda_{k,t-1} + \epsilon_\lambda$

Hierarchical Distributions

$$\alpha_i \sim t(\mu_\alpha = 0, \sigma_\alpha, \nu_\alpha = 10)$$

$$\beta_{recent\_returns_i} \sim t(\mu_{recent\_returns,c}, \sigma_{recent\_returns,c}, \nu = 10)$$



Recent Returns

i	assets
t	time (months)
k	factors
c	asset class

$$\mu_{i,t} = \alpha_i + \sum_{k=1}^{K} w_{k,i} * \lambda_{k,t} + \beta_{recent\_returns_i} * X_{recent\_returns_{i,t}}$$

Asset Intercept + Dynamic Factor Model + Offset for recent returns

S	M	T	W	T	F	S
29	30	31	1	2	3	4
5	6	7	8	9	10	11
12	13	14	15	16	17	18
19	20	21	22	23	24	25
26	27	28	29	30	1	2
3	4	5	6	7	8	9

Returns in week prior to t normalized by asset

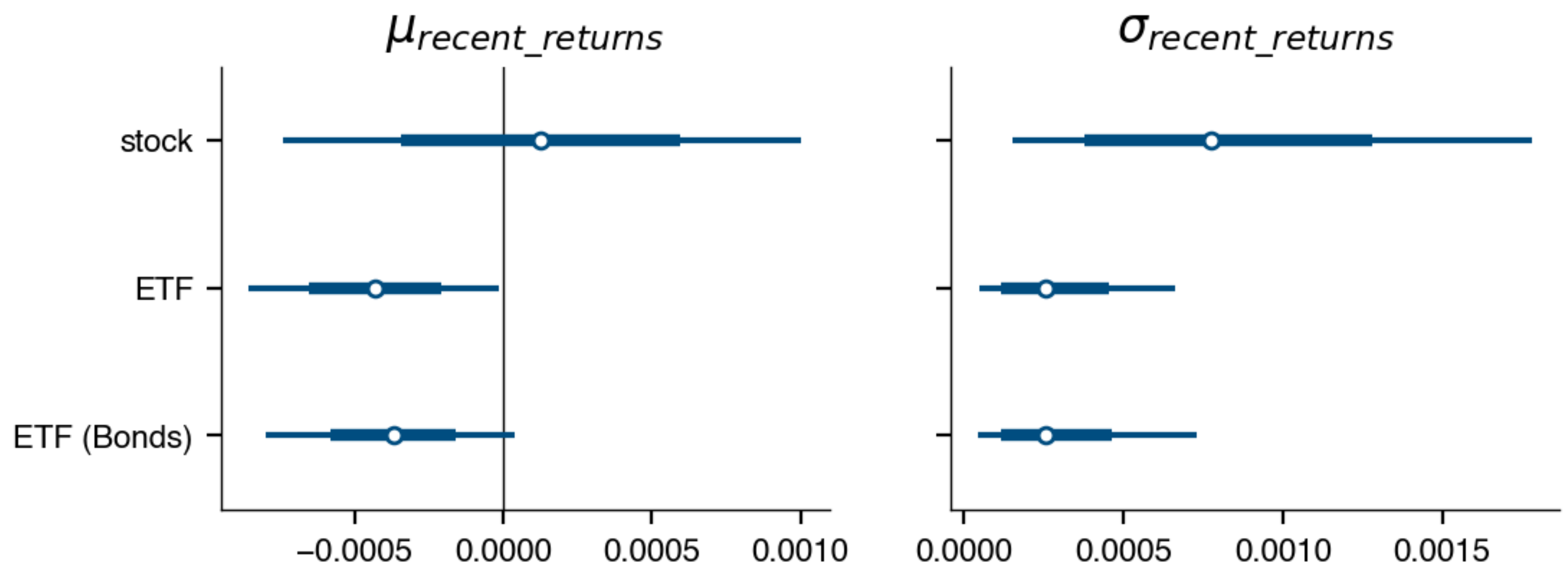
Factor loadings $w_{k,i} \sim t(\mu_w = 0, \sigma_w = 1, \nu_w = 10)$

Factor dynamics $\lambda_{k,t} = \rho * \lambda_{k,t-1} + \epsilon_\lambda$

Hierarchical Distributions

$$\alpha_i \sim t(\mu_{alpha} = 0, \sigma_{alpha}, \nu_{alpha} = 10)$$

$$\beta_{recent\_returns_i} \sim t(\mu_{recent\_returns,c}, \sigma_{recent\_returns,c}, \nu = 10)$$



Winners and Losers

i	assets
t	time (months)
k	factors
c	asset class

- Noise is symmetric, so these components are solely responsible for predicting winners and losers

$$\mu_{i,t} = \alpha_i + \sum_{k=1}^{K} w_{k,i} * \lambda_{k,t} + \beta_{recent\_returns_i} * X_{recent\_returns_{i,t}}$$

Asset Intercept + Dynamic Factor Model + Offset for recent returns

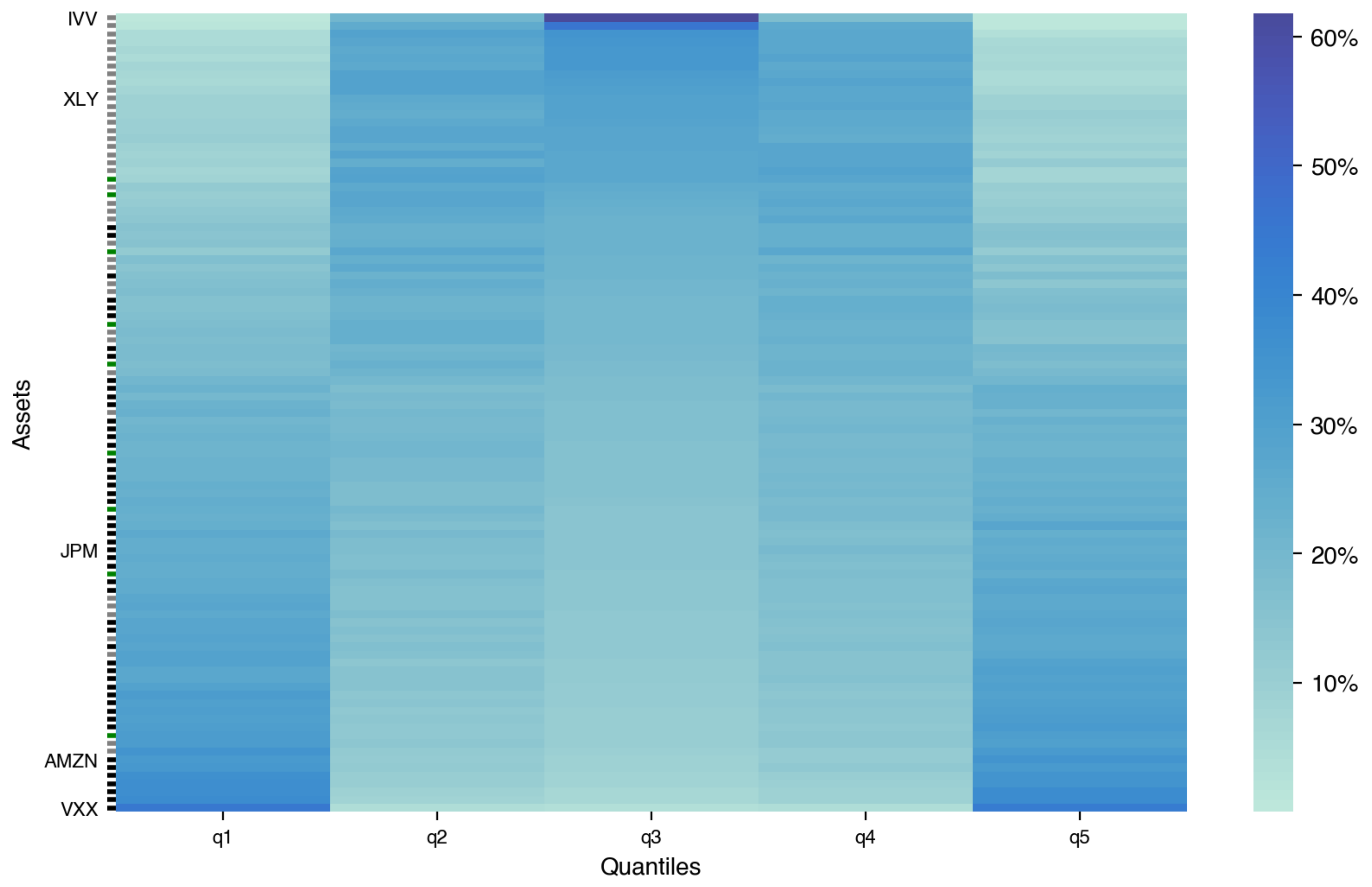
Factor loadings $w_{k,i} \sim t(\mu_w = 0, \sigma_w = 1, \nu_w = 10)$

Factor dynamics $\lambda_{k,t} = \rho * \lambda_{k,t-1} + \epsilon_\lambda$

Hierarchical Distributions

$$\alpha_i \sim t(\mu_{alpha} = 0, \sigma_{alpha}, \nu_{alpha} = 10)$$

$$\beta_{recent\_returns_i} \sim t(\mu_{recent\_returns,c}, \sigma_{recent\_returns,c}, \nu = 10)$$



Noise

i	assets
t	time (months)
k	factors
c	asset class

$$r_{i,t} = \mu_{i,t} + \epsilon_{i,t}$$

Monthly return = Expected value + Noise

$$\epsilon_{i,t} \sim t(0, \sigma_{i,t}, \nu_i)$$

$$log(\sigma_{i,t}) = \theta_i + \beta_{recent\_vol_i} * X_{recent\_vol_t} + \beta_{earnings_i} * I_{earnings_t}$$

Asset Intercept + Offset for recent volatility + Offset for earnings month

Hierarchical Distributions

$$\theta_i \sim t(\mu_{\theta,c}, \sigma_{\theta,c}, \nu_\theta)$$

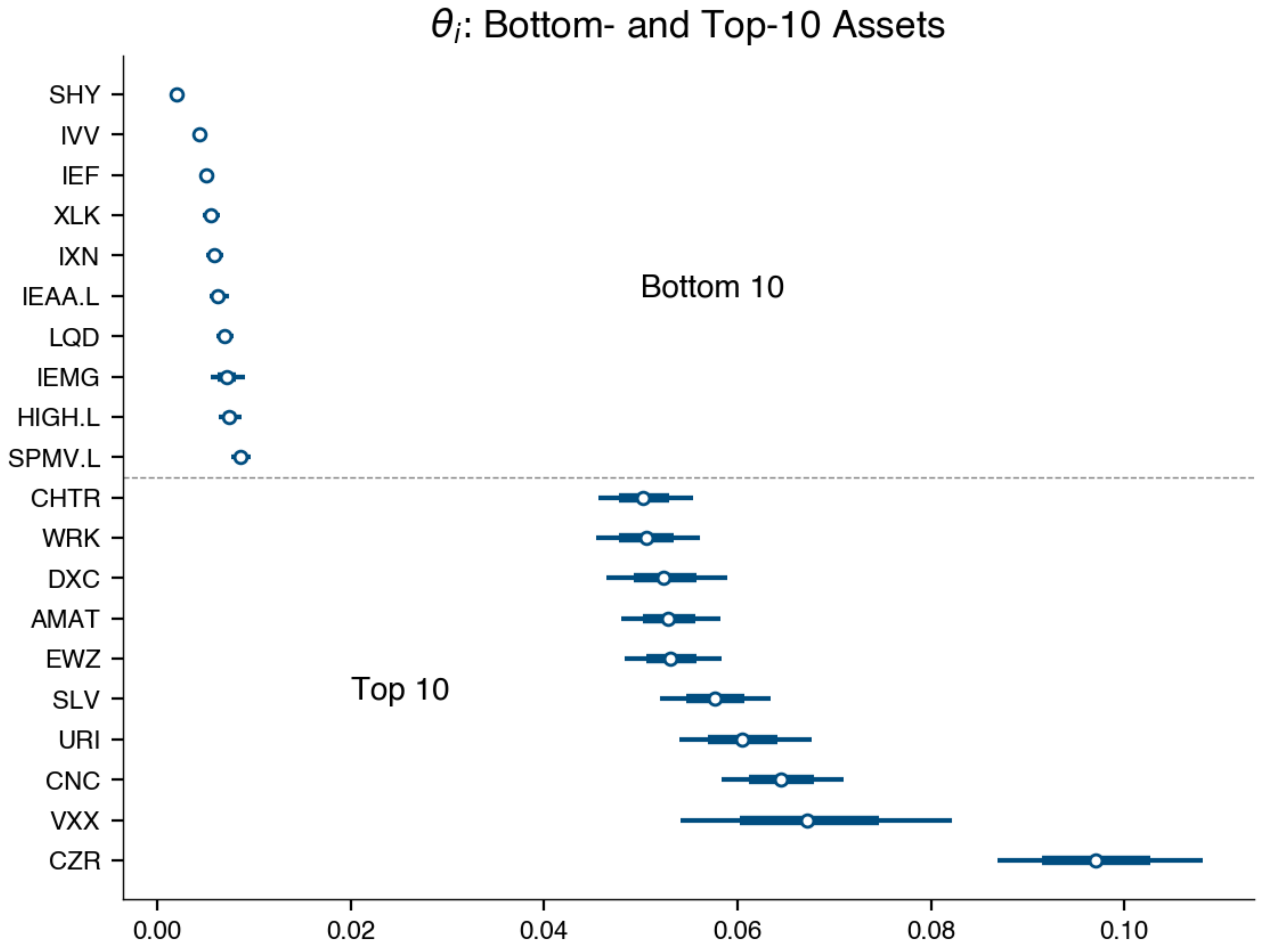
$$\beta_{recent\_vol_i} \sim \mathcal{N}(\mu_{recent\_vol,c}, \sigma_{recent\_vol,c})$$

$$\beta_{earnings_i} \sim \mathcal{N}(\mu_{earnings}, \sigma_{earnings})$$

$$\nu_i \sim Gamma(\mu_\nu = 7, \sigma_\nu)$$

Asset Intercept

i	assets
t	time (months)
k	factors
c	asset class



$$\epsilon_{i,t} \sim t(0, \sigma_{i,t}, \nu_i)$$

$$log(\sigma_{i,t}) = \theta_i + \beta_{recent\_vol_i} * X_{recent\_vol_t} + \beta_{earnings_i} * I_{earnings_t}$$

Asset Intercept + Offset for recent volatility + Offset for earnings month

Hierarchical Distributions

$$\theta_i \sim t(\mu_{\theta,c}, \sigma_{\theta,c}, \nu_\theta)$$

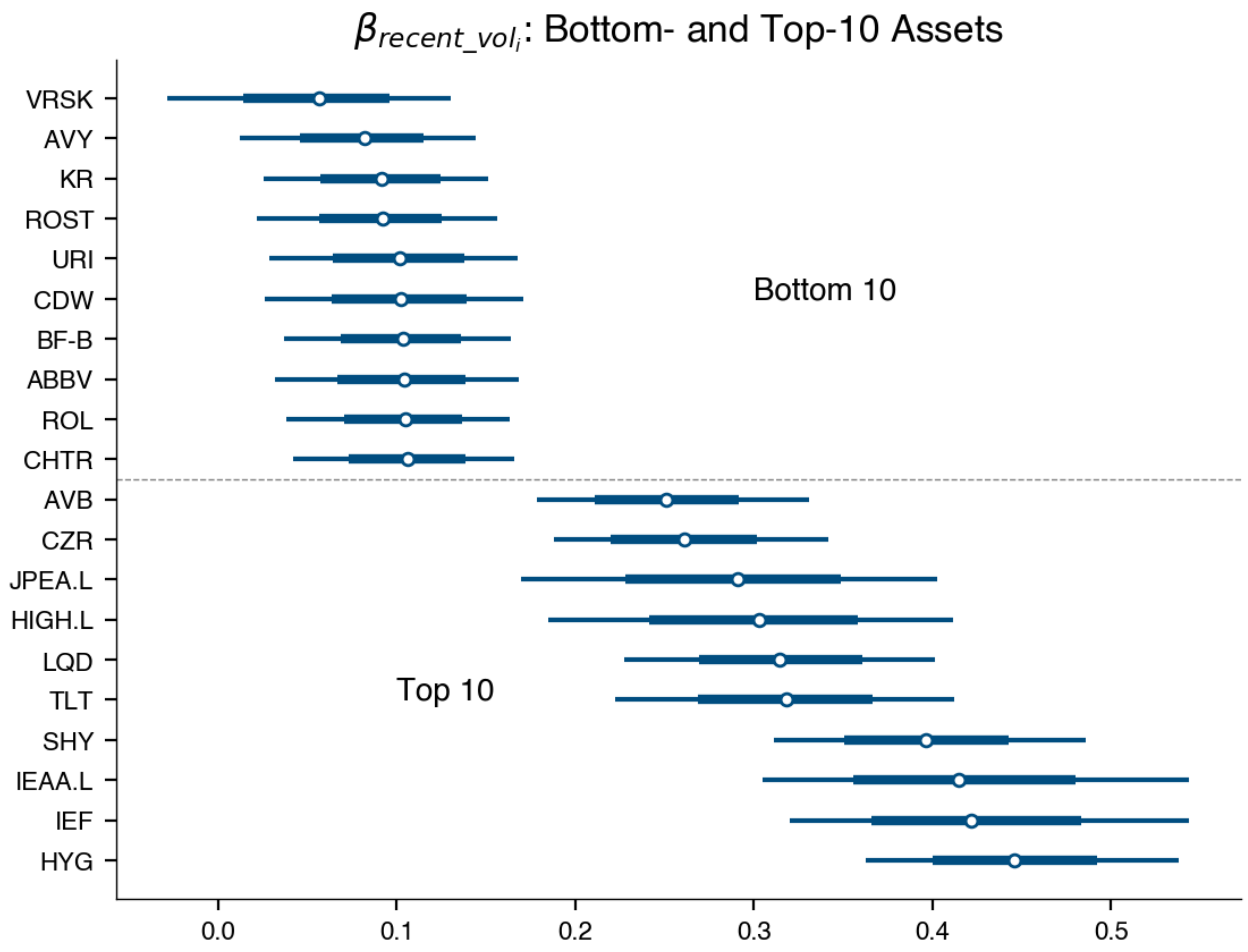
$$\beta_{recent\_vol_i} \sim \mathcal{N}(\mu_{recent\_vol,c}, \sigma_{recent\_vol,c})$$

$$\beta_{earnings_i} \sim \mathcal{N}(\mu_{earnings}, \sigma_{earnings})$$

$$\nu_i \sim Gamma(\mu_\nu = 7, \sigma_\nu)$$

i	assets
t	time (months)
k	factors
c	asset class

Recent Volatility



$$\epsilon_{i,t} \sim t(0, \sigma_{i,t}, \nu_i)$$

$$log(\sigma_{i,t}) = \theta_i + \beta_{recent\_vol_i} * X_{recent\_vol_t} + \beta_{earnings_i} * I_{earnings_t}$$

Asset Intercept + **Offset for recent volatility** + Offset for earnings month

Hierarchical Distributions

$$\theta_i \sim t(\mu_{\theta,c}, \sigma_{\theta,c}, \nu_\theta)$$

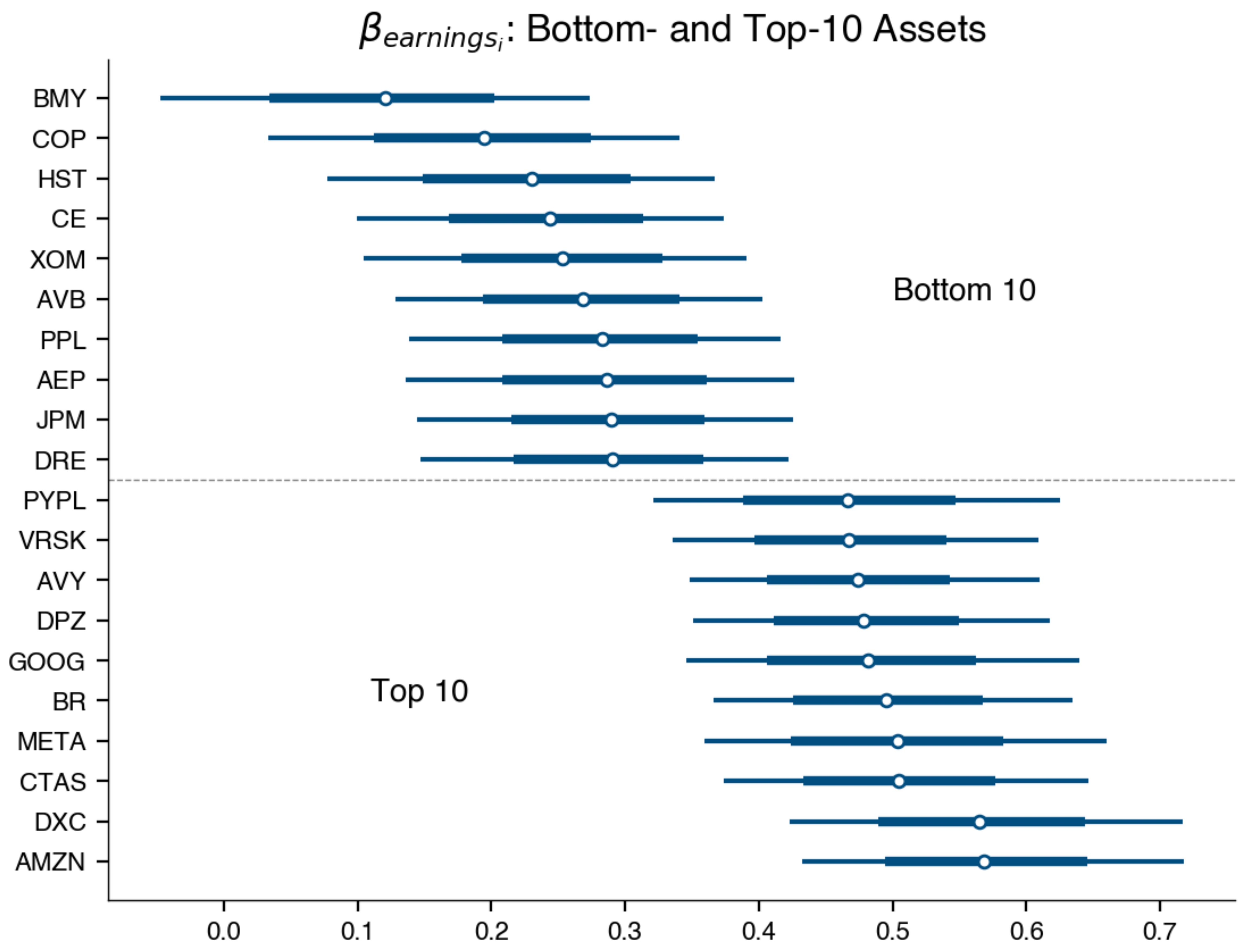
$$\beta_{recent\_vol_i} \sim \mathcal{N}(\mu_{recent\_vol,c}, \sigma_{recent\_vol,c})$$

$$\beta_{earnings_i} \sim \mathcal{N}(\mu_{earnings}, \sigma_{earnings})$$

$$\nu_i \sim Gamma(\mu_\nu = 7, \sigma_\nu)$$

i assets
t time (months)
k factors
c asset class

Earnings Month



$$\epsilon_{i,t} \sim t(0, \sigma_{i,t}, \nu_i)$$

$$log(\sigma_{i,t}) = \theta_i + \beta_{recent\_vol_i} * X_{recent\_vol_t} + \beta_{earnings_i} * I_{earnings_t}$$

Asset Intercept + Offset for recent volatility + Offset for earnings month

Hierarchical Distributions

$$\theta_i \sim t(\mu_{\theta,c}, \sigma_{\theta,c}, \nu_\theta)$$

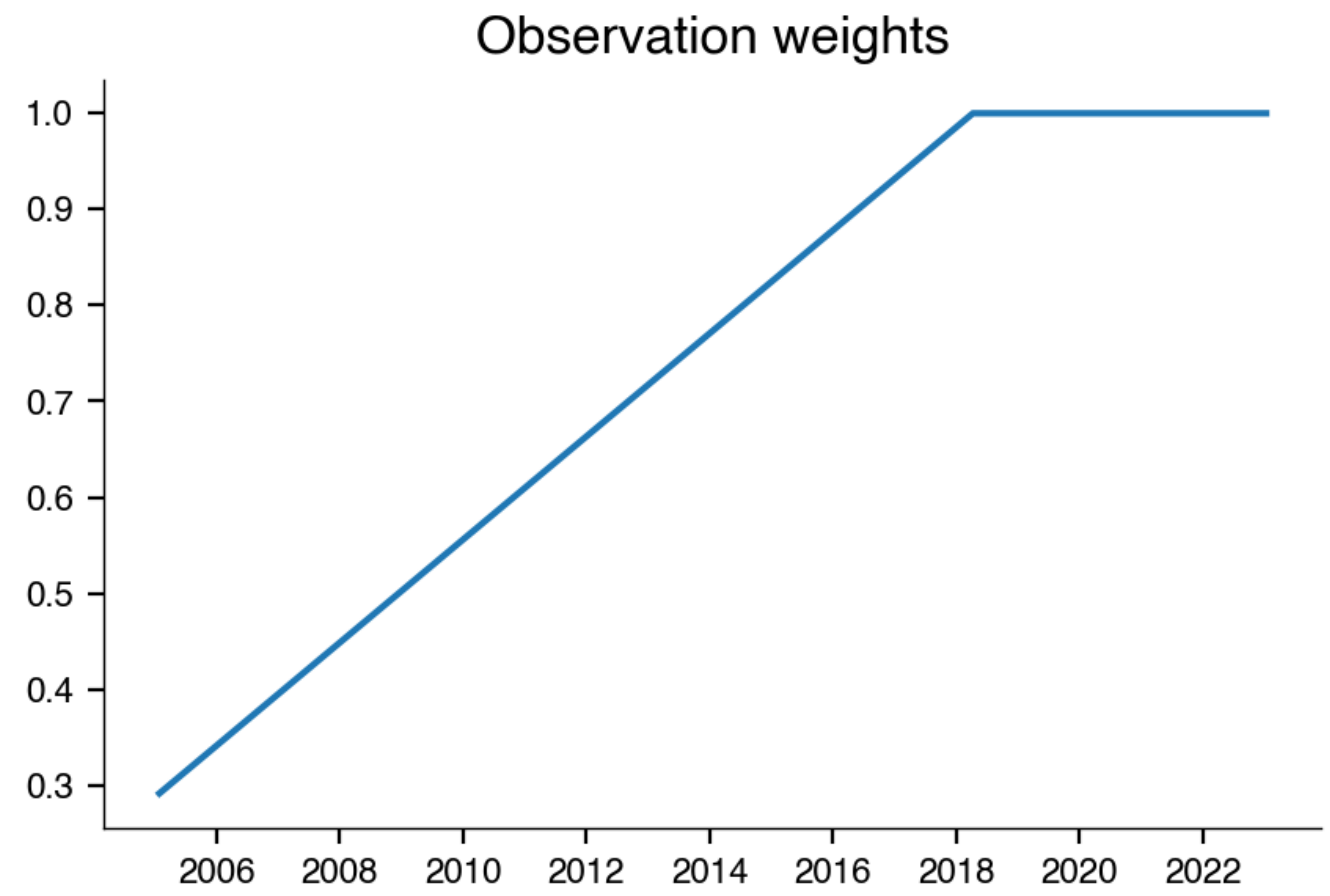
$$\beta_{recent\_vol_i} \sim \mathcal{N}(\mu_{recent\_vol,c}, \sigma_{recent\_vol,c})$$

$$\beta_{earnings_i} \sim \mathcal{N}(\mu_{earnings}, \sigma_{earnings})$$

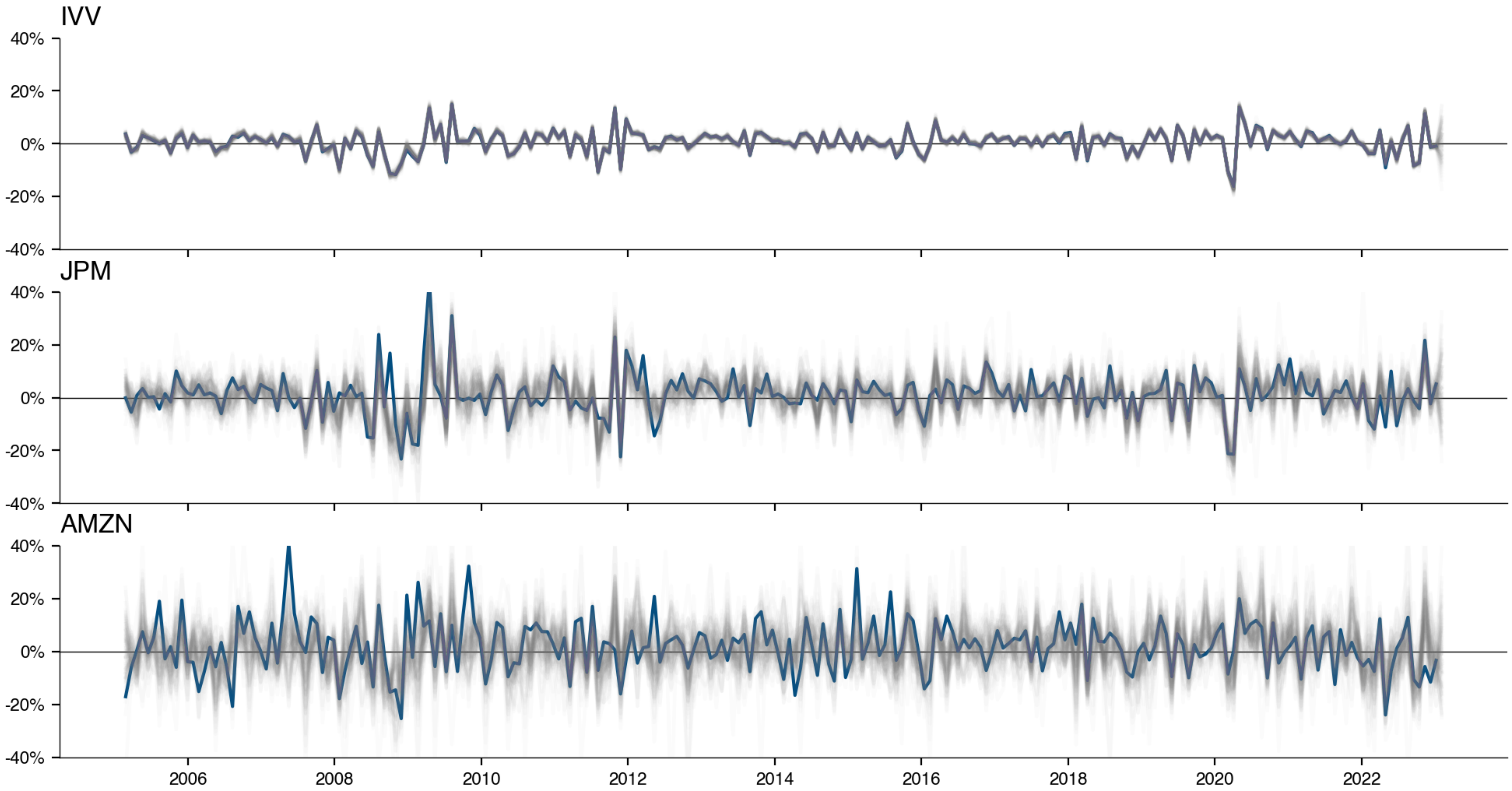
$$\nu_i \sim Gamma(\mu_\nu = 7, \sigma_\nu)$$

Downweighting the distant past

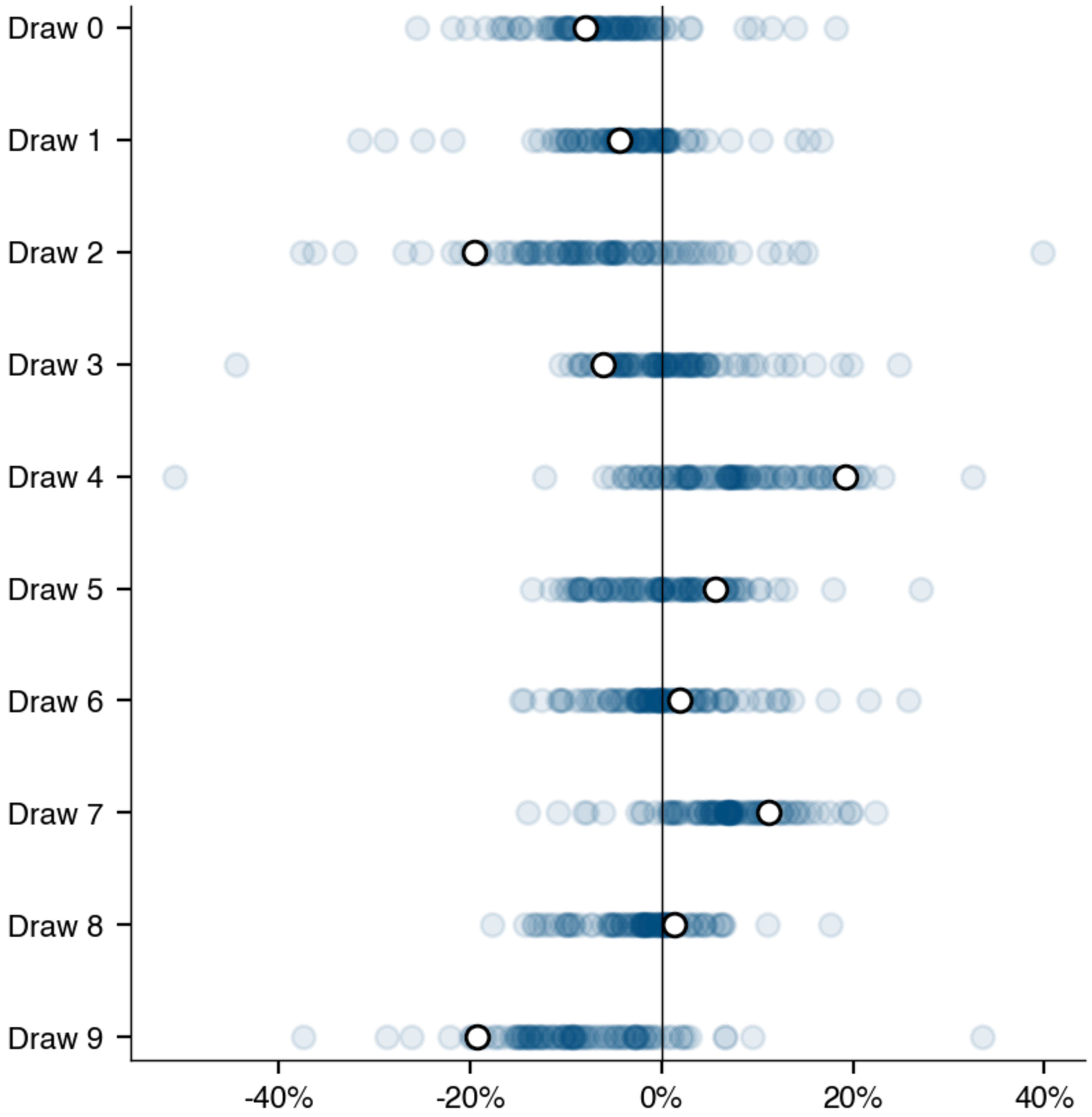
- Factor loadings are static over time…
- … but the recent past is more relevant



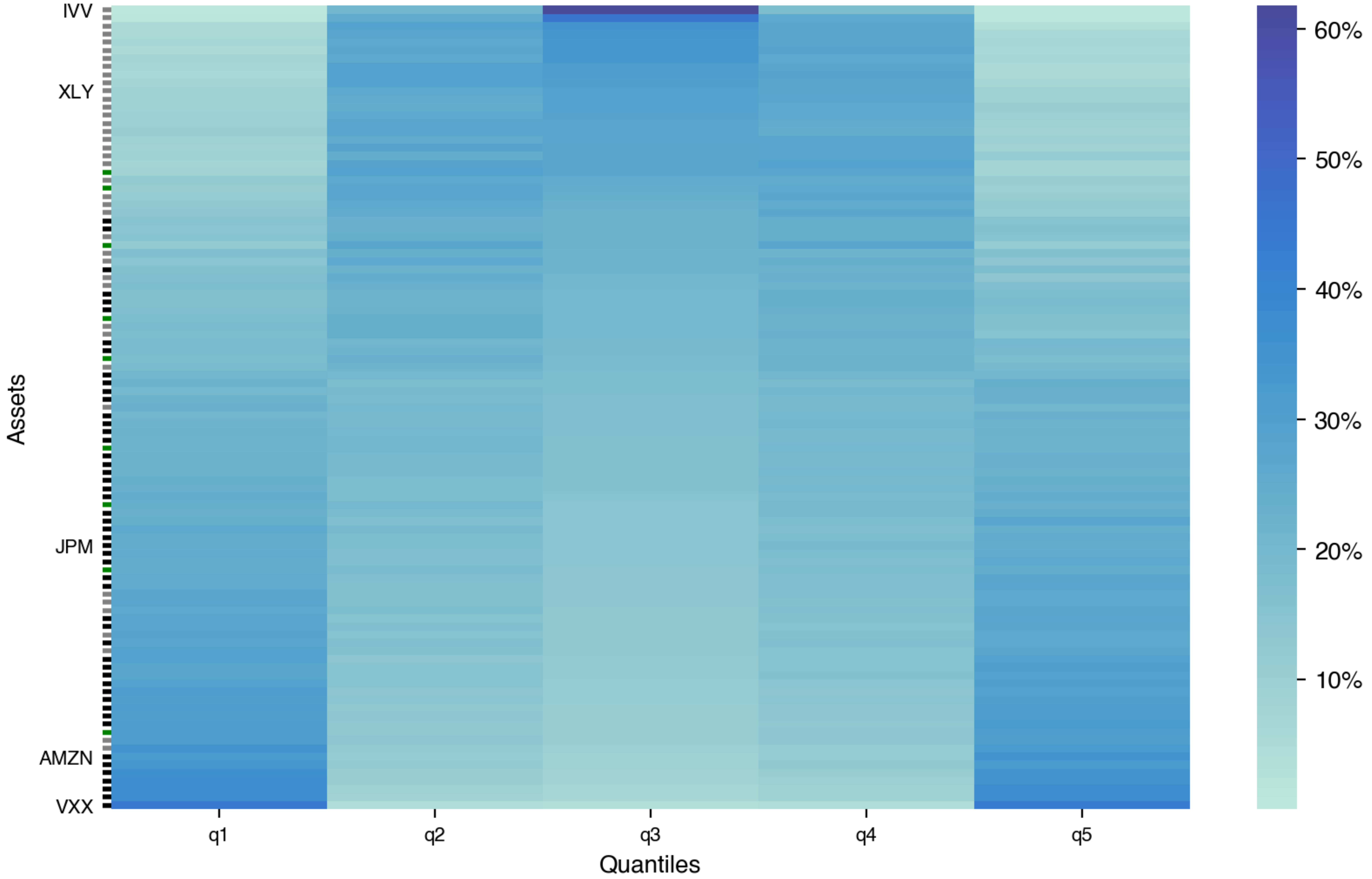
Posterior Predictive Check



Generating a forecast: samples from the future



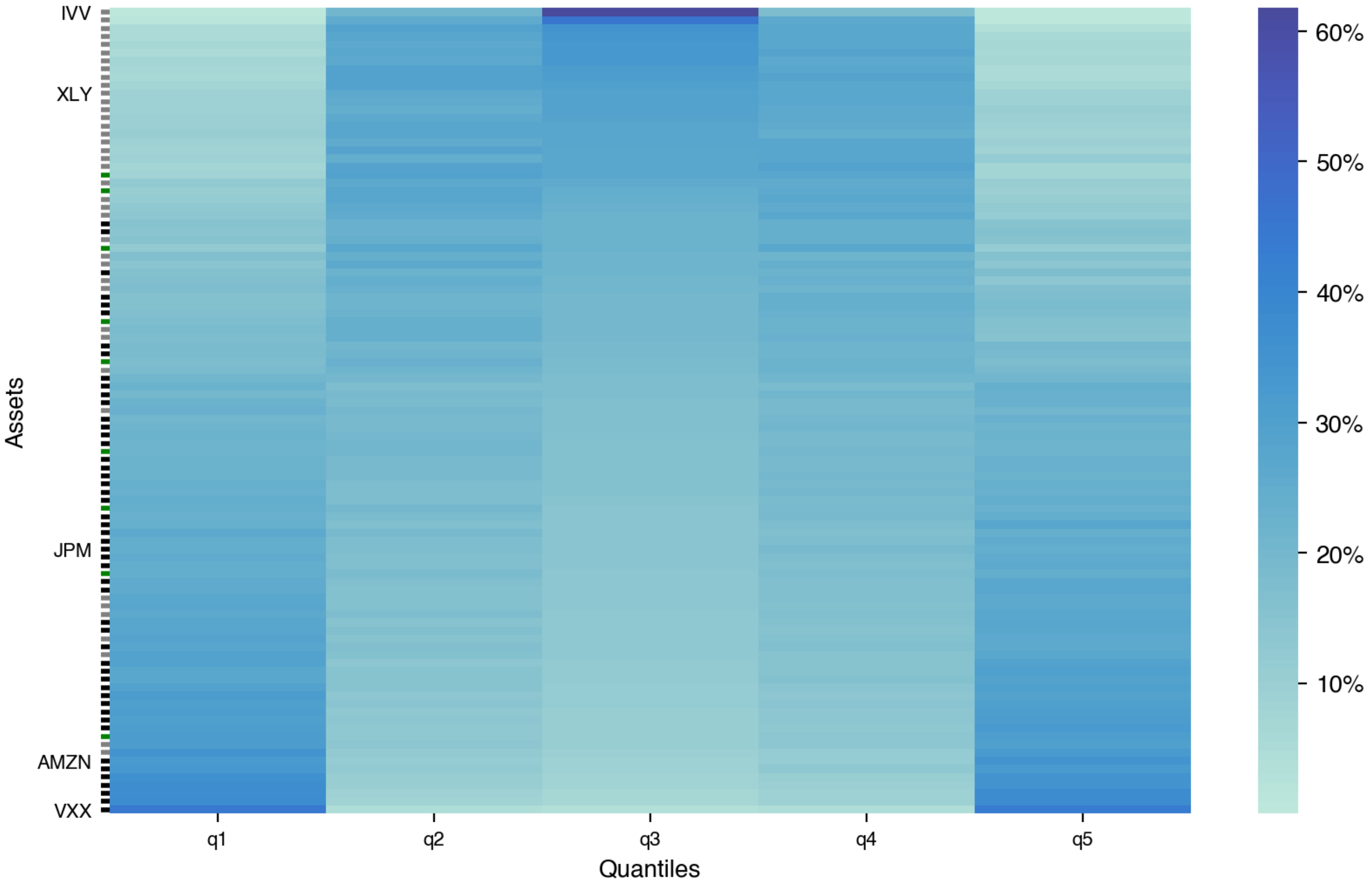
Generating a forecast: samples from the future



Agenda

- Thought process
- My model: Bayesian dynamic factor model with heteroskedasticity
- **Pros and cons of my approach**
- Future work

PRO: well-calibrated probabilities



- But*: probabilities are conditional on the model being the correct model*

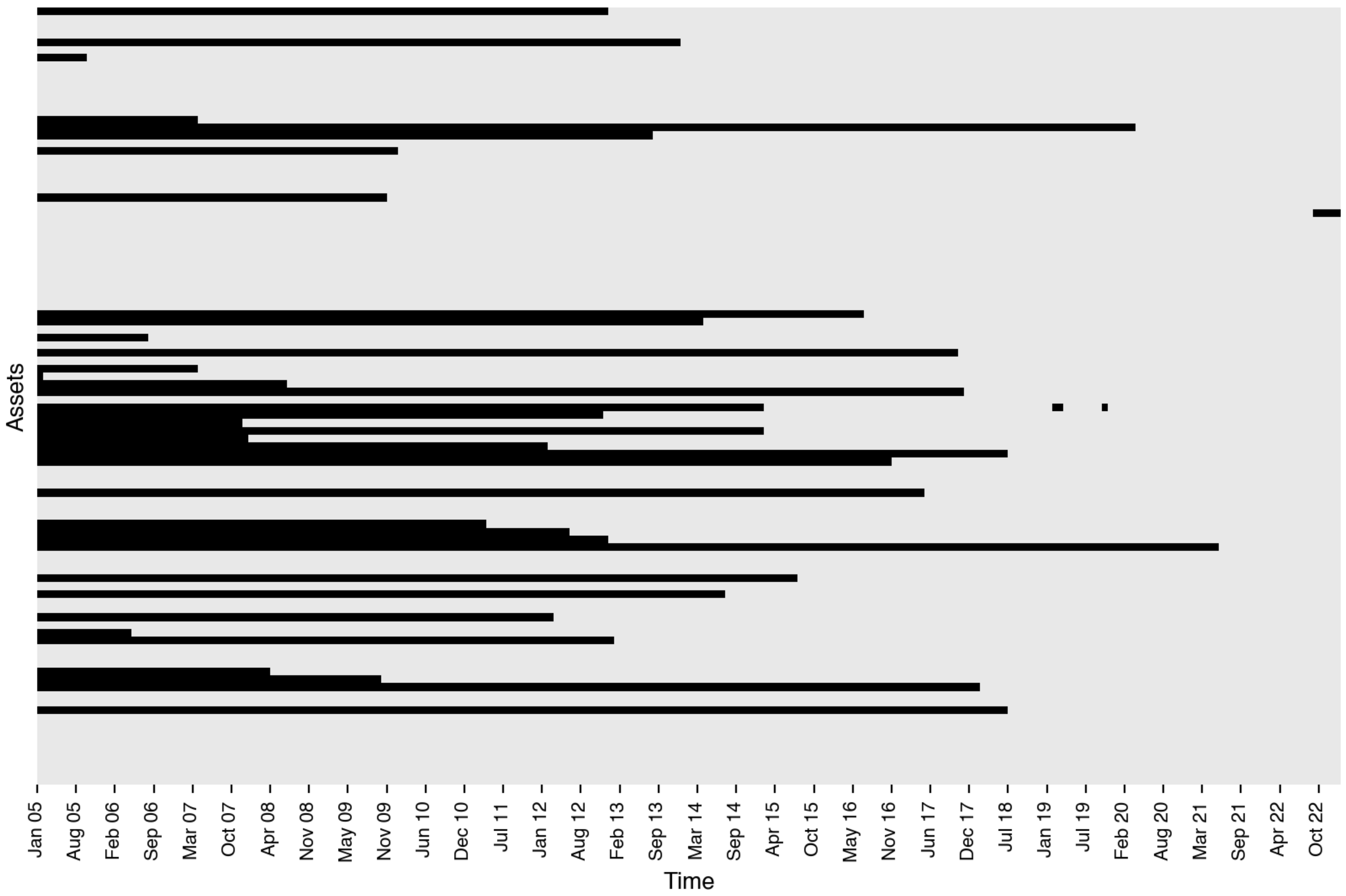
PRO: modular & interpretable

- Can model expected value and noise separately

Monthly return = Expected value + Noise

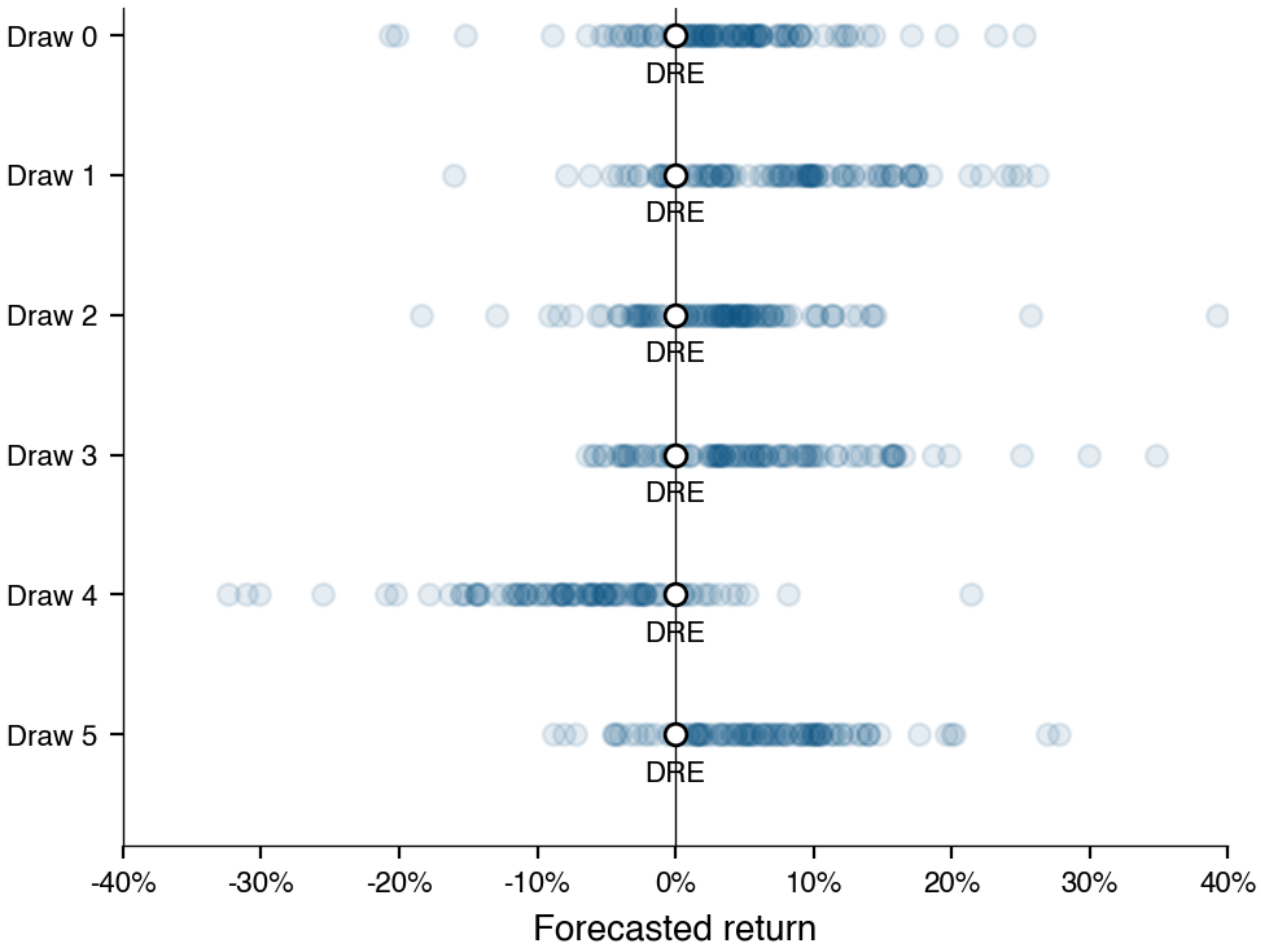
Expected value	Noise Magnitude
Asset Intercept	Asset Intercept
Dynamic Factor Model	Offset for recent volatility
+ Offset for recent returns	+ Offset for earnings month
Expected value	Noise Magnitude

PRO: Missing data are not a problem



PRO: Samples from the future are easy to edit

- DRE was acquired midway through the competition
- Organizers: we will treat DRE as if it has zero return from here on
    - Does not mean 100% probability of the middle quantile!
- Because we get samples or simulations of possible futures, I just set DRE to zero in each sample



PRO: Hierarchical Distributions

$$\mu_{i,t} = \alpha_i + \sum_{k=1}^{K} w_{k,i} * \lambda_{k,t} + \beta_{recent\_returns_i} * X_{recent\_returns_{i,t}}$$

Asset Intercept + Dynamic Factor Model + Offset for recent returns

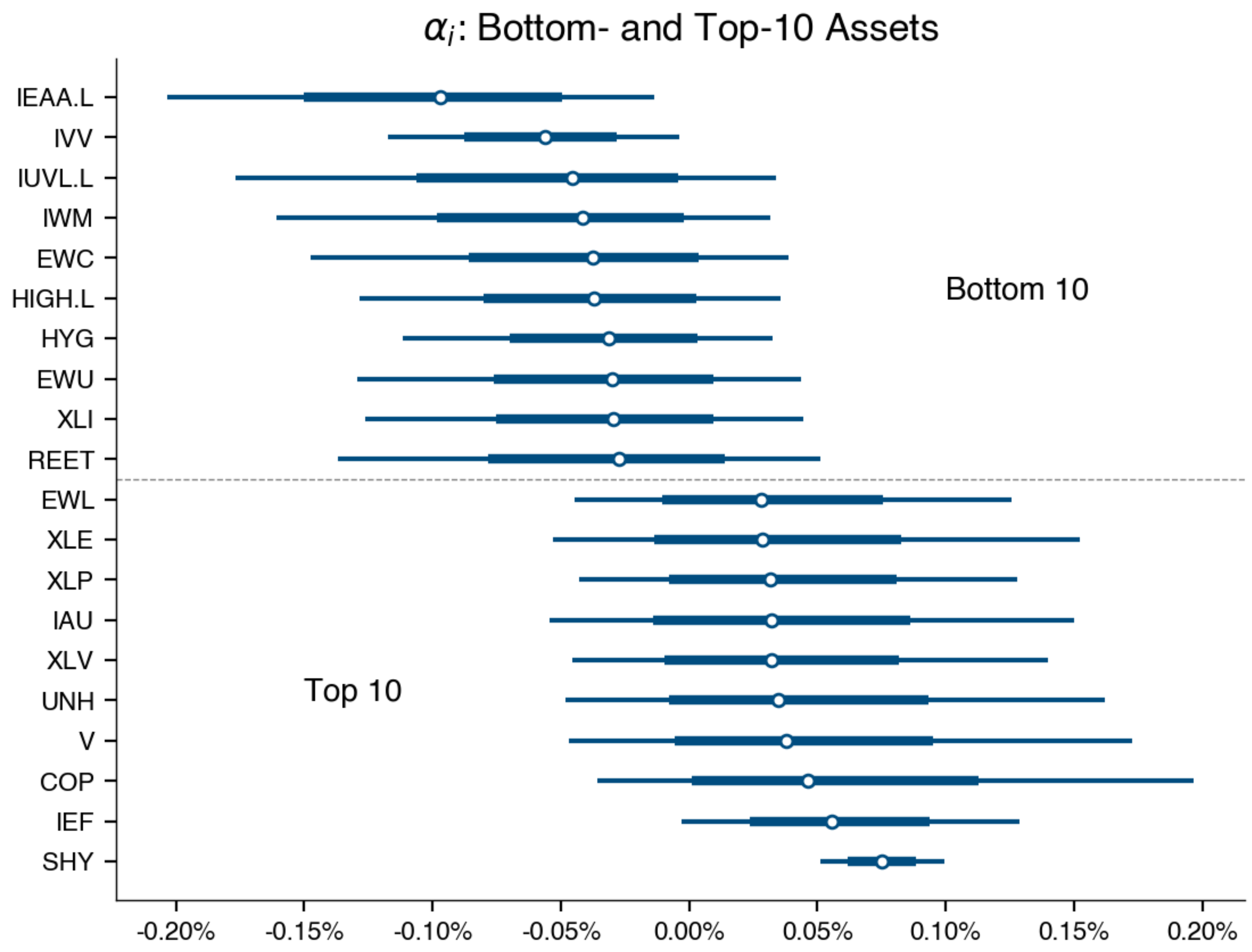
Factor loadings $w_{k,i} \sim t(\mu_w = 0, \sigma_w = 1, \nu_w = 10)$

Factor dynamics $\lambda_{k,t} = \rho * \lambda_{k,t-1} + \epsilon_\lambda$

Hierarchical Distributions

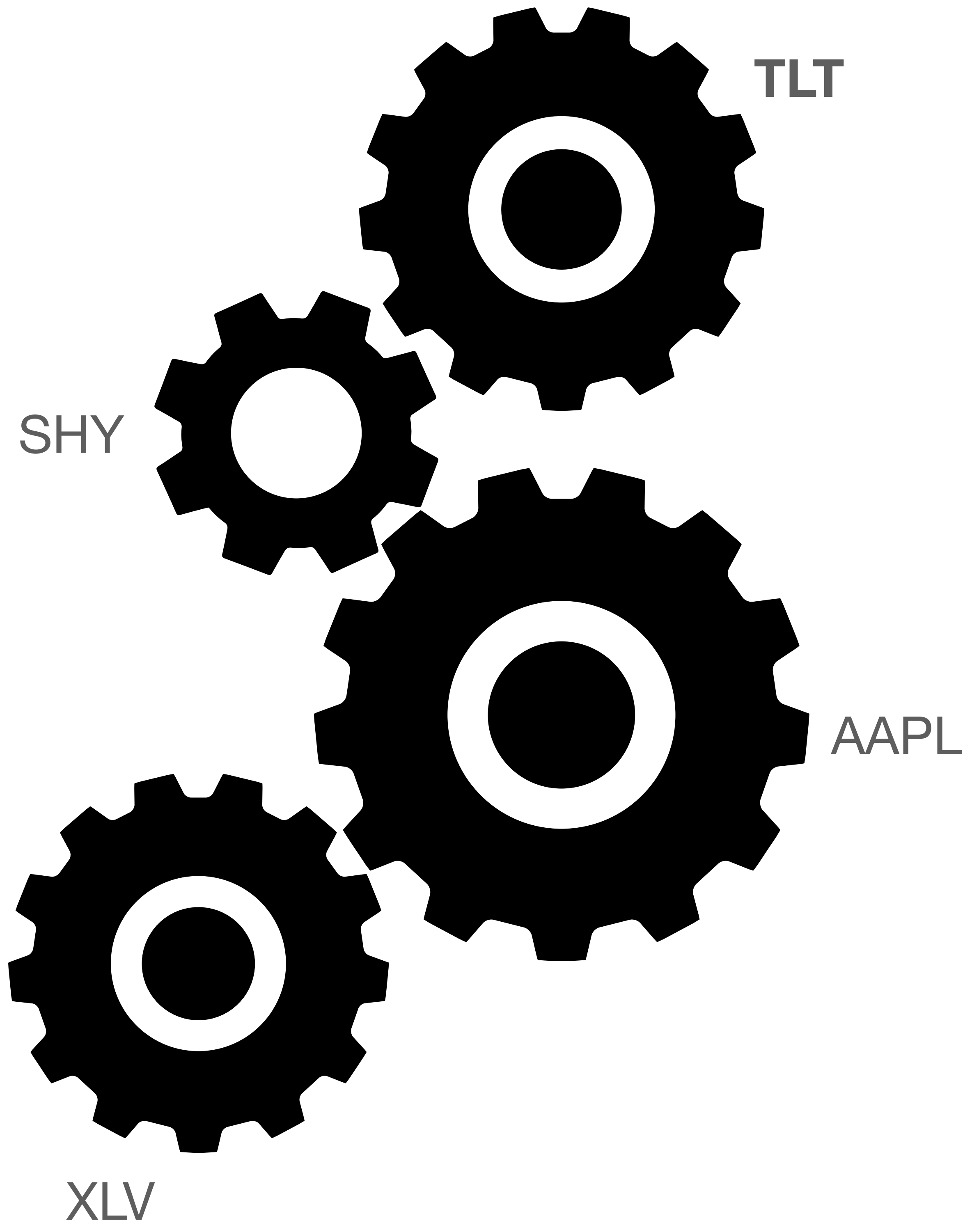
$$\alpha_i \sim t(\mu_\alpha = 0, \sigma_\alpha, \nu_\alpha = 10)$$

$$\beta_{recent\_returns_i} \sim t(\mu_{recent\_returns,c}, \sigma_{recent\_returns,c}, \nu = 10)$$



PRO: Probabilistic subjective forecasts

```
this_mod = m6_models.ModelR(
    df=df,
    df_earnings=df_earnings,
    df_cal=df_cal,
    n_factors=7,
    forecasts={'TLT': {
        {'mean': .02, 'sd': .1}
    }},
    weight_cutoff=datetime(2018, 5, 1)
)
```



- Assumes you can make good subjective forecasts

CON: Inference is costly

- Inference is costly (but decreasingly so)
  - ~90min on Intel Macbook Pro…
  - …~25min on an M2 Mac mini
  - Traditional cross-validation or backtesting requires either the cloud or a lot of patience

Agenda

- Thought process
- My model: Bayesian dynamic factor model with heteroskedasticity
- Pros and cons of my approach
- **Future work**

Future work

- Factor dynamics
  - VAR
- Dynamic factor loadings
  - Instrumented PCA

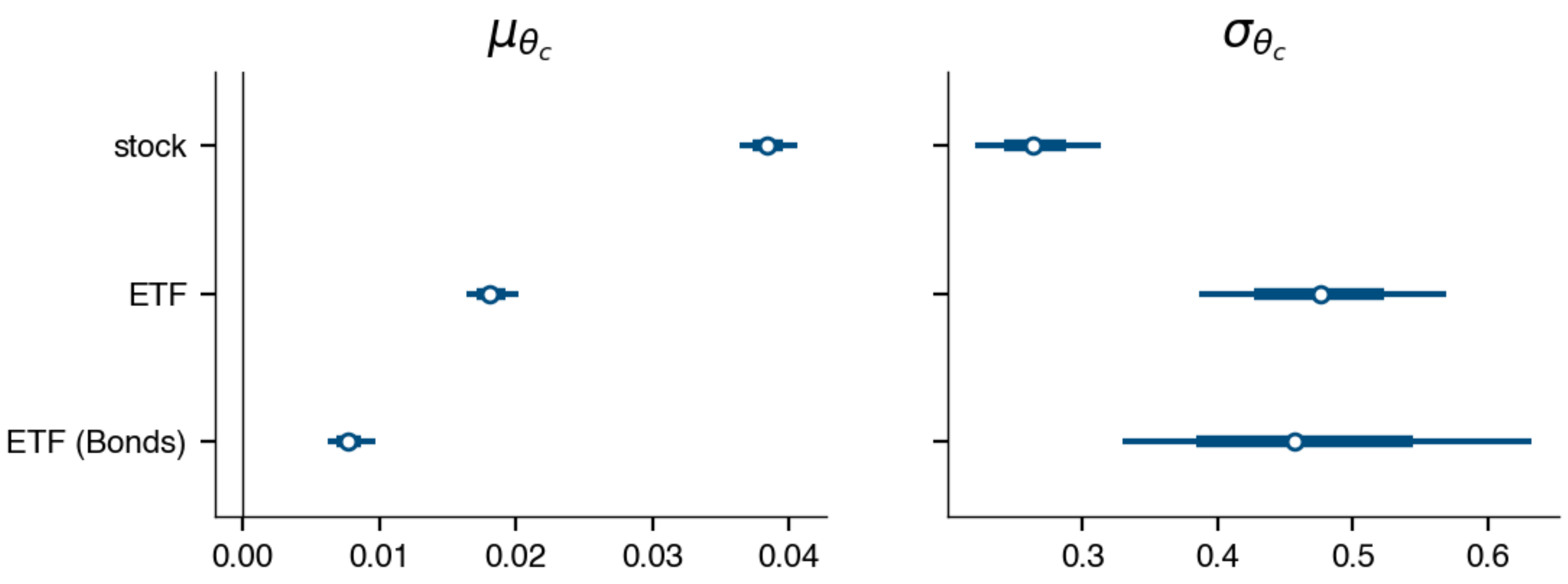
Summary

- Probabilistic forecasting -> probabilistic programming
- My model: Bayesian dynamic factor model with heteroskedasticity
- Strengths of my approach included:
  - well-calibrated probabilities
  - modular and interpretable
  - easy handling of missing data

Appendix

i	assets
t	time (months)
k	factors
c	asset class

Asset Intercept



$$\epsilon_{i,t} \sim t(0, \sigma_{i,t}, \nu_i)$$

$$log(\sigma_{i,t}) = \theta_i + \beta_{recent\_vol_i} * X_{recent\_vol_t} + \beta_{earnings_i} * I_{earnings_t}$$

Asset Intercept + Offset for recent volatility + Offset for earnings month

Hierarchical Distributions

$$\theta_i \sim t(\mu_{\theta,c}, \sigma_{\theta,c}, \nu_\theta)$$

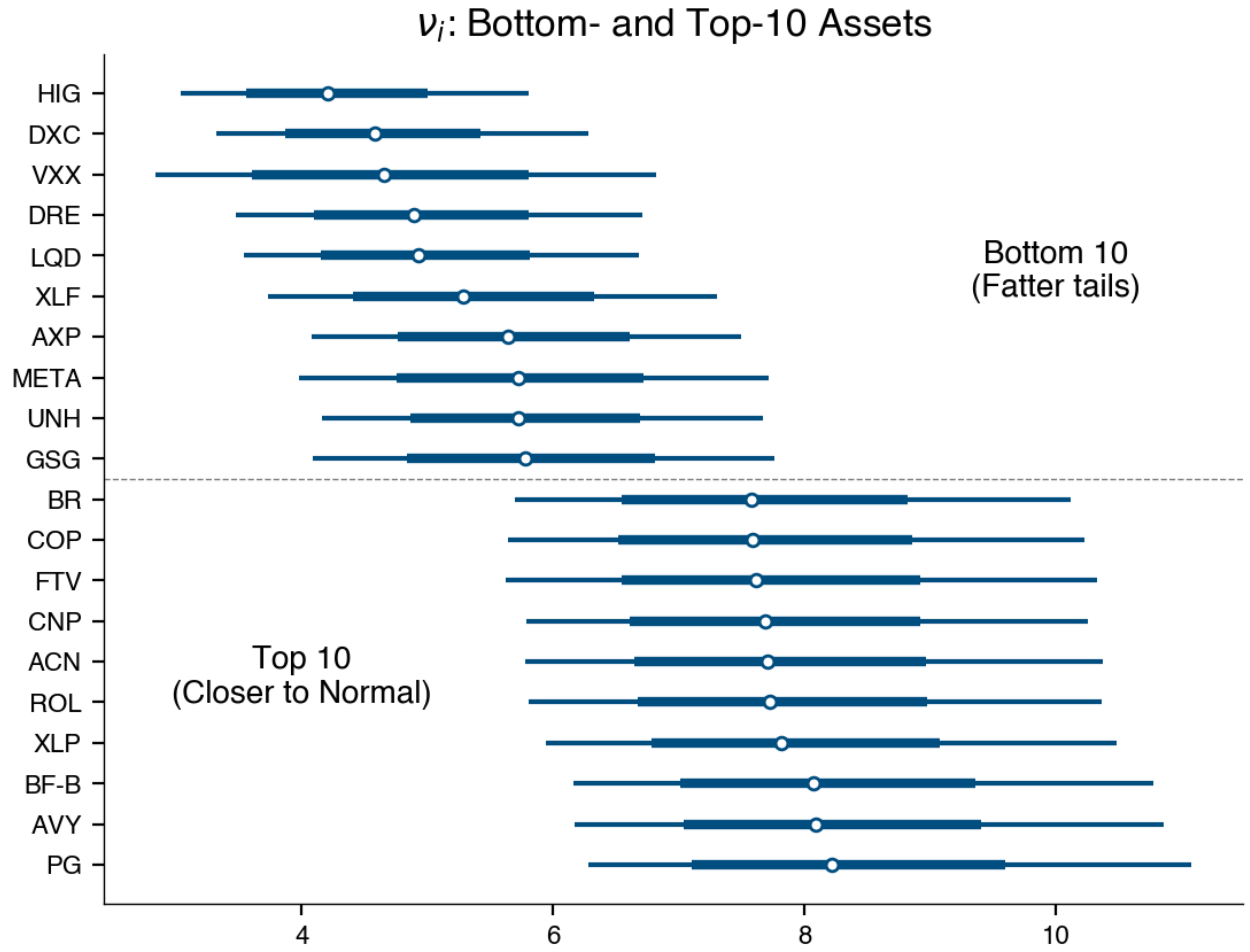
$$\beta_{recent\_vol_i} \sim \mathcal{N}(\mu_{recent\_vol,c}, \sigma_{recent\_vol,c})$$

$$\beta_{earnings_i} \sim \mathcal{N}(\mu_{earnings}, \sigma_{earnings})$$

$$\nu_i \sim Gamma(\mu_\nu = 7, \sigma_\nu)$$

i	assets
t	time (months)
k	factors
c	asset class

Tail Fatness



$$\epsilon_{i,t} \sim t(0, \sigma_{i,t}, \nu_i)$$

$$log(\sigma_{i,t}) = \theta_i + \beta_{recent\_vol_i} * X_{recent\_vol_t} + \beta_{earnings_i} * I_{earnings_t}$$

Asset Intercept + Offset for recent volatility + Offset for earnings month

Hierarchical Distributions

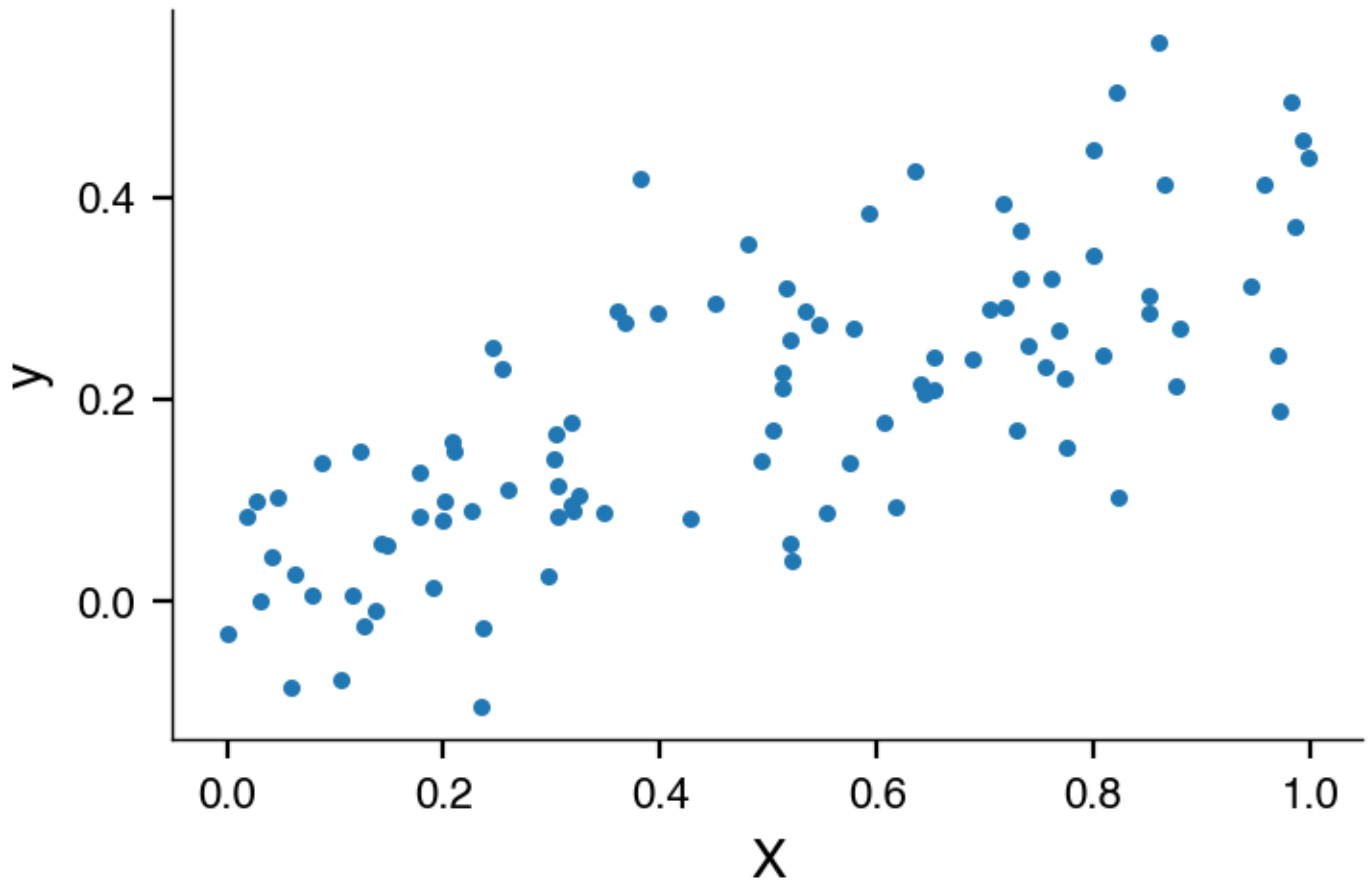
$$\theta_i \sim t(\mu_{\theta,c}, \sigma_{\theta,c}, \nu_\theta)$$

$$\beta_{recent\_vol_i} \sim \mathcal{N}(\mu_{recent\_vol,c}, \sigma_{recent\_vol,c})$$

$$\beta_{earnings_i} \sim \mathcal{N}(\mu_{earnings}, \sigma_{earnings})$$

$$\nu_i \sim Gamma(\mu_\nu = 7, \sigma_\nu)$$

Probabilistic programming example



$$y = \beta * x + \epsilon$$

```
with pm.Model() as m:
    beta = pm.Normal('beta', mu=0, sigma=3)
    sigma = pm.HalfNormal('sigma', 2)
    y = pm.Normal(
        'y',
        mu=beta*x,
        sigma=sigma,
        observed=y)
```

"A **probabilistic programming language** is a high-level language that makes it easy for a developer to **define probability models** and then **"solve" these models automatically.**"

Probabilistic programming example

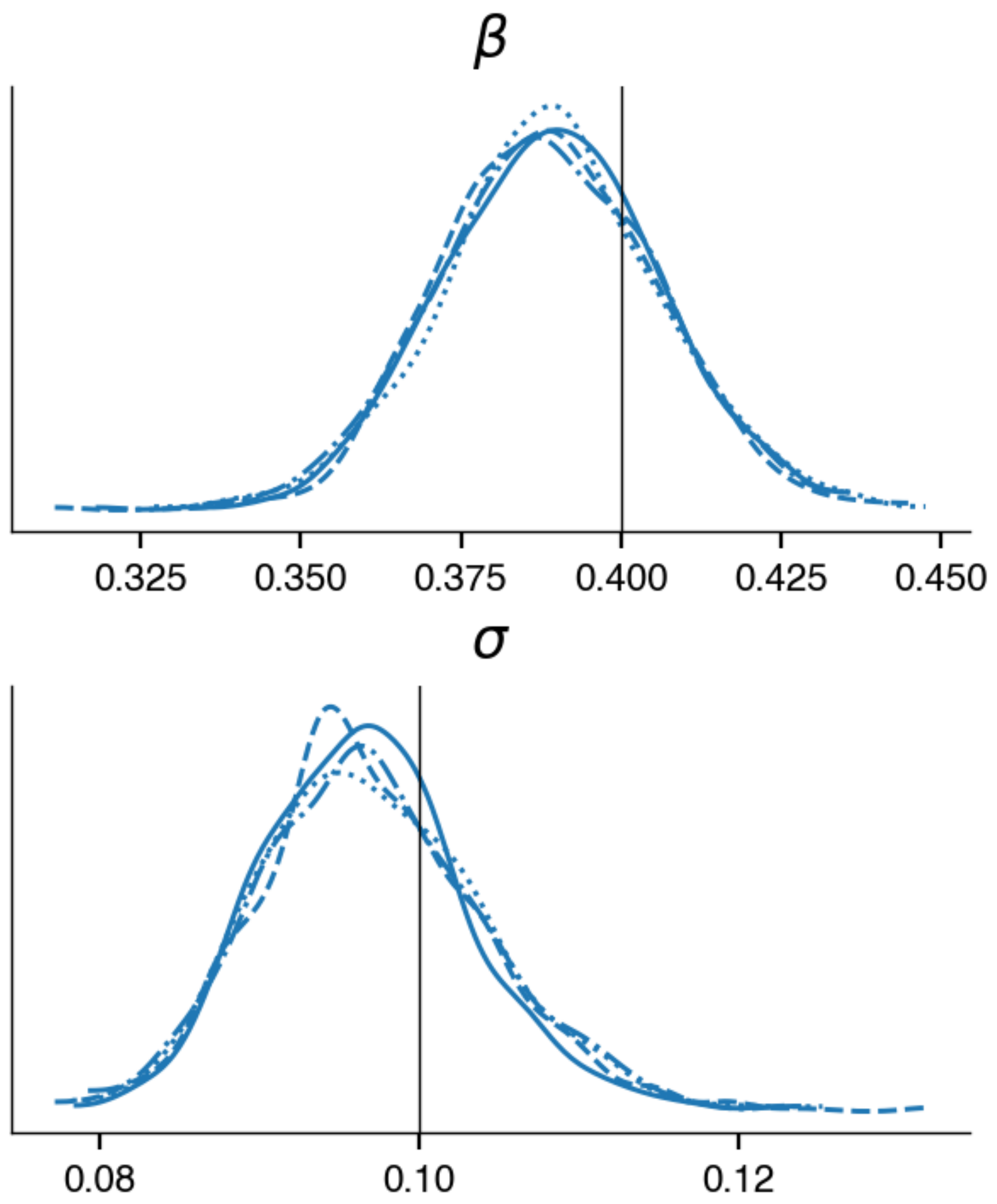
$$y = \beta * x + \epsilon$$

```python
with m:
    trace = pm.sample()
```

```
Auto-assigning NUTS sampler...
Initializing NUTS using jitter+adapt_diag...
Multiprocess sampling (4 chains in 4 jobs)
NUTS: [beta, sigma]
```

100.00% [8000/8000 00:00<00:00 Sampling 4 chains, 0 divergences]

```
Sampling 4 chains for 1_000 tune and 1_000 draw iterations (4_000 + 4_000 draws total) took 1 seconds.
```



“A **probabilistic programming language** is a high-level language that makes it easy for a developer to **define probability models** and then **“solve” these models automatically.**”

Probabilistic programming example