(Spooky) Covariance

ISTA 410 / INFO 510: Bayesian Modeling and Inference

U. of Arizona School of Information

🎃 October 27, 2021🎃

Outline

Last time:

- Multilevel linear model
- Varying slopes, varying intercepts
- Observed correlation between parameters

Today:

- Modeling correlation between parameters
- Gaussian processes

Example: ozone in Beijing

Data set: ~ 5 years of air quality monitoring data from Beijing

- Weather properties: temperature, pressure, dewpoint, rain, wind speed
- Pollutants: ozone, sulfur dioxide, nitrous oxide, carbon monoxide, particulates
- Measurements collected hourly at 12 monitoring stations

Simple modeling task: model ozone (in $\mu g/m^3$, about twice ppb) as a function of temperature

Preprocessing: group measurements by date/station

Basic model

Exploratory plotting suggests:

- the log maximum daily ozone reading is linearly associated with high temperature
- use max instead of average to prevent daily temporal associations from contributing

Preliminary model:

$$\log O_3 \sim \text{Normal}(\theta, \sigma)$$
$$\theta = \alpha + \beta_T T_{\max}$$
$$\alpha \sim \text{Normal}(0, 3)$$
$$\beta \sim \text{Normal}(0, 1)$$

Prior predictive reasonableness

Hang on, let's check those priors:

$$\alpha \sim \text{Normal}(0, 3)$$

$$\beta \sim \text{Normal}(0, 1)$$

Prior predictive reasonableness

Hang on, let's check those priors:

$$\alpha \sim \text{Normal}(0, 3)$$
$$\beta \sim \text{Normal}(0, 1)$$

If $\alpha = 3, \beta = 1$, then on a 30 degree day we get $\log O_3 \approx 33$; meaning about 100 trillion ppb.

A really bad ozone day might be a couple hundred ppb (about $\log O_3 \approx 6$ or 7). Let's rein in these priors.

Prior predictive reasonableness

With new priors:

$$\begin{aligned}
\log O_3 &\sim \text{Normal}(\theta, \sigma) \\
\theta &= \alpha + \beta_T T_{\max} \\
\alpha &\sim \text{Normal}(0, 2) \\
\beta &\sim \text{Normal}(0, 0.1)
\end{aligned}$$

Now a high estimate combined with a hot day gives us something more like $\log O_3 \approx 6$.

Let's proceed!

Model specification

In Python:

```
with pm.Model() as linear_model:
    alpha = pm.Normal('alpha', 0, 2)
    beta = pm.Normal('beta', 0, 0.1)

    # Model equation
    theta = alpha[dailies.dropna()['station_id']]
        + beta * dailies.dropna()['TEMP']

    # Likelihood
    y_ = pm.Normal('y', mu=theta, sigma = sigma,
         observed = dailies.dropna()['log_ozone'])
```

Model results

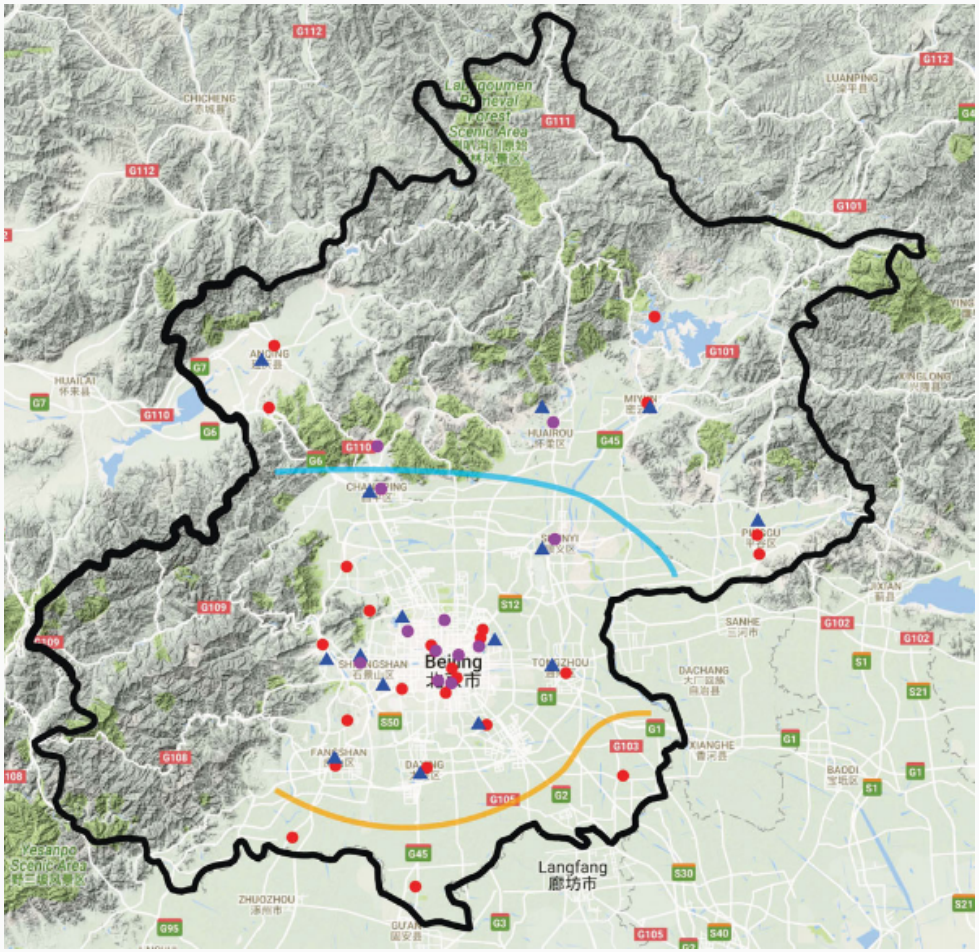
Here is a summary table for the preliminary model:

	mean	sd	hdi_3%	hdi_97%
alpha	3.454	0.009	3.436	3.471
beta	0.054	0.000	0.053	0.055
sigma	0.612	0.003	0.606	0.619

As expected:

- warmer days have more ozone
- specifically, change of 1 degree in high temperature associated with about a 5% increase in peak ozone concentration

Why multilevel model?



A map of Beijing. Purple dots are the 12 monitoring stations.

Why multilevel model?

We should expect some variation among geographic sites:

- Ozone source density may vary
- Topography influences local airflow patterns
- Sensor calibration differences

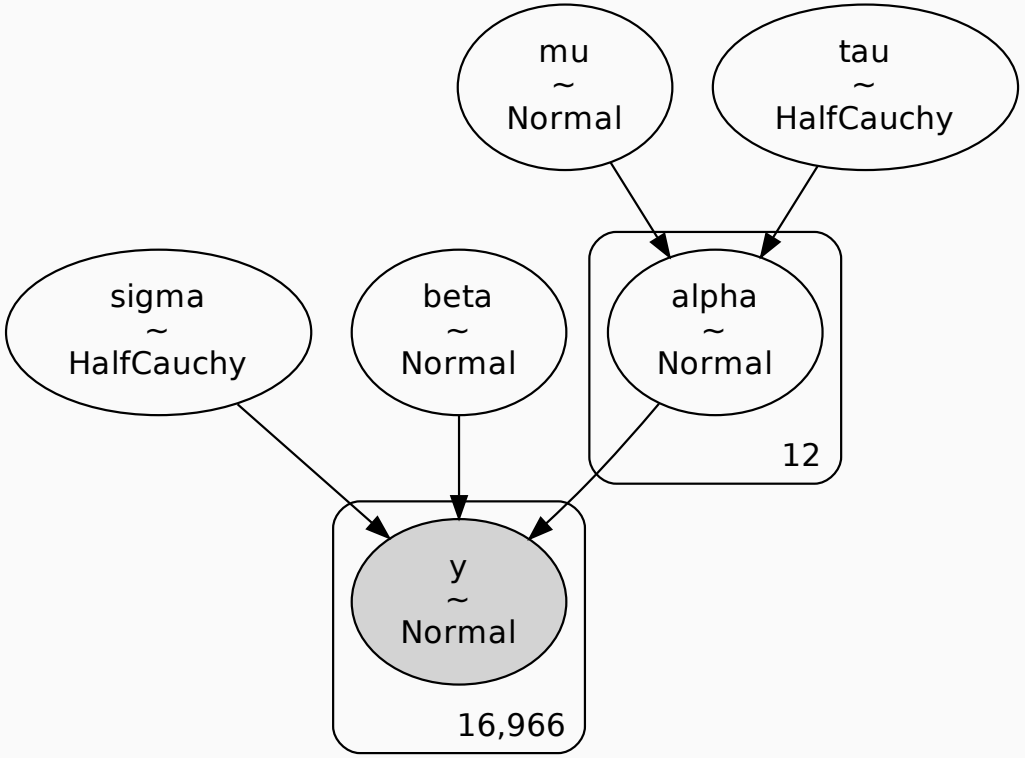
Simple extension of our model: allow varying intercepts across monitoring stations, to allow for each station to have a different "baseline" ozone level.

Varying-intercepts model

So let's extend the model:

$$
\begin{aligned}
\log O_3 &\sim \text{Normal}(\theta, \sigma) \\
\theta &= \alpha_j + \beta_T T_{\max} \\
\beta &\sim \text{Normal}(0, 1) \\
\alpha_j &\sim \text{Normal}(\mu, \tau) \\
\mu &\sim \text{Normal}(0, 3) \\
\tau &\sim \text{HalfCauchy}(1)
\end{aligned}
$$

Varying-intercepts model



Model specification

```
with pm.Model() as multilevel_model:
    # Hyperparameters
    mu = pm.Normal('mu', 0, 2)
    tau = pm.HalfCauchy('tau', 1)
    # Parameters
    sigma = pm.HalfCauchy('sigma', 1)
    alpha = pm.Normal('alpha', mu, tau, shape = 12)
    beta = pm.Normal('beta', 0, 0.1)
    # Model equation
    theta = alpha[dailies.dropna()['station_id']]
    + beta * dailies.dropna()['TEMP']
    # Likelihood
    y_ = pm.Normal('y', mu=theta, sigma = sigma,
        observed = dailies.dropna()['log_ozone'])
```

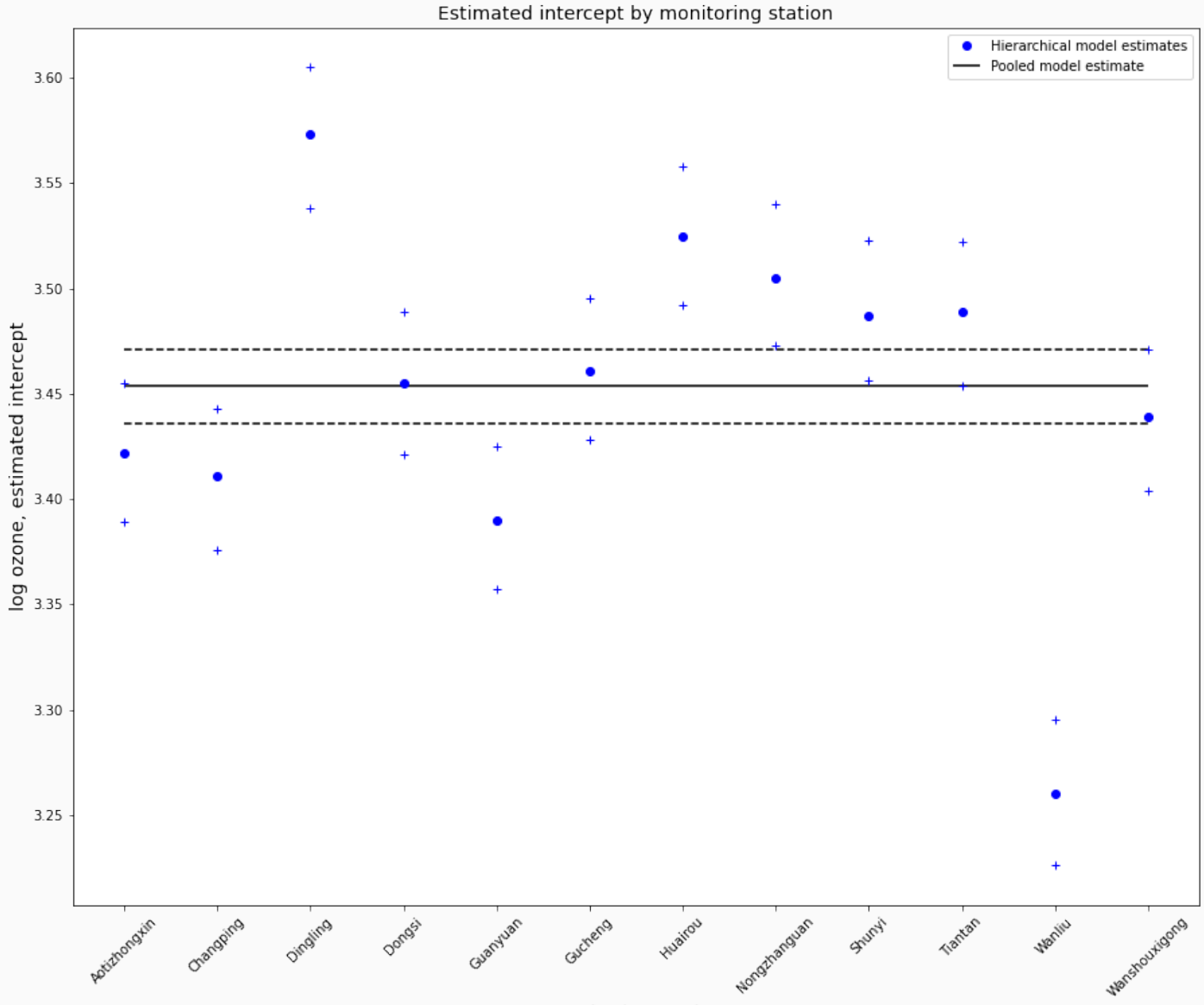
Model fitting

Results:

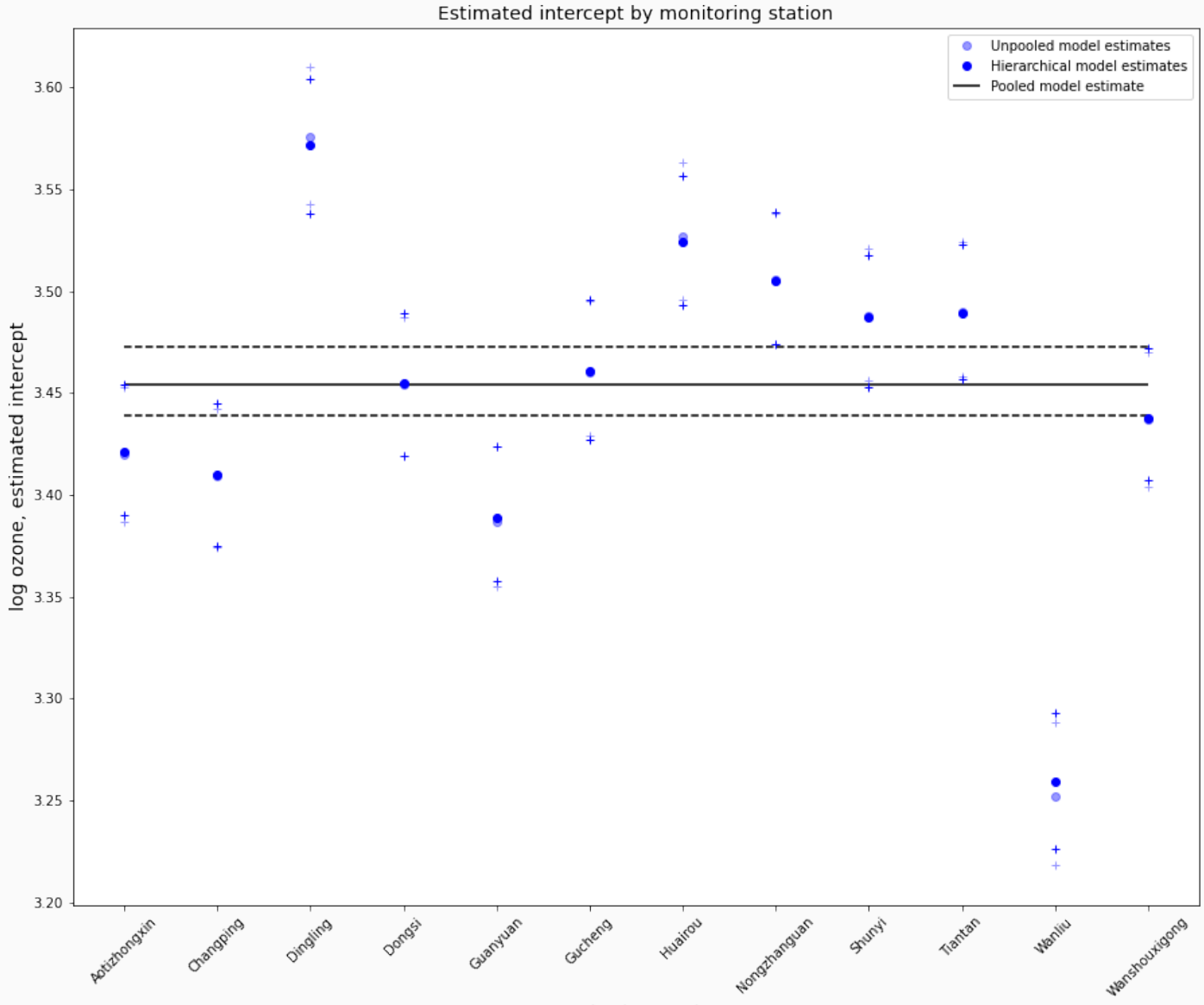
	mean	sd	hdi_3%	hdi_97%
mu	3.451	0.029	3.396	3.504
alpha[0]	3.422	0.018	3.389	3.455
alpha[1]	3.411	0.018	3.376	3.443
alpha[2]	3.573	0.018	3.538	3.605
alpha[3]	3.455	0.018	3.421	3.489
alpha[4]	3.390	0.018	3.357	3.425

- Hierarchical mean parameter similar to pooled intercept, but station intercepts vary

Model fitting



Model fitting



Model comparison

Comparing the models using PSIS-LOO:

	rank	loo	p_loo	d_loo	weight	se	dse	warning	loo_scale
multilevel	0	-15629.400655	17.039457	0.000000	0.969016	189.518973	0.00000	False	log
simple	1	-15757.707232	6.678447	128.306577	0.030984	190.474747	16.99114	False	log

- Multilevel model: better predictive score
- Also allows us to estimate which locations have elevated (Dingling) or depressed (Wanliu) baseline ozone

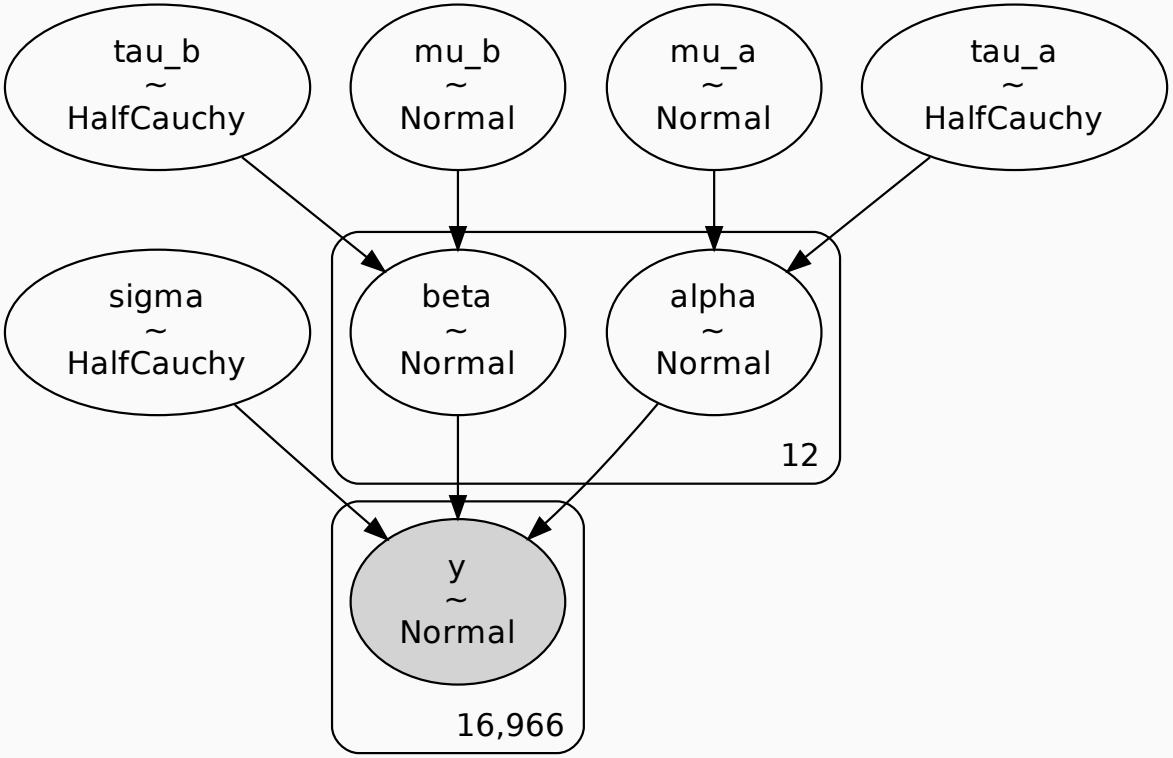
Varying slopes

Of course, once we have *varying intercepts*, it seems natural to also want *varying slopes*.

Extend the model again:

$$
\begin{aligned}
\log O_3 &\sim \text{Normal}(\mu, \sigma) \\
\mu &= \alpha_j + \beta_{T,j} T_{\max} \\
\beta_{T,j} &\sim \text{Normal}(\mu_\beta, \tau_\beta) \\
\alpha_j &\sim \text{Normal}(\mu_\alpha, \tau_\beta) \\
\mu_\alpha &\sim \text{Normal}(0, 2) \\
\tau_\alpha &\sim \text{HalfCauchy}(1) \\
\mu_\beta &\sim \text{Normal}(0, 0.1) \\
\tau_\beta &\sim \text{HalfCauchy}(1)
\end{aligned}
$$

Varying slopes



Model comparison

Finally, we add the varying-slopes model to the model comparison:

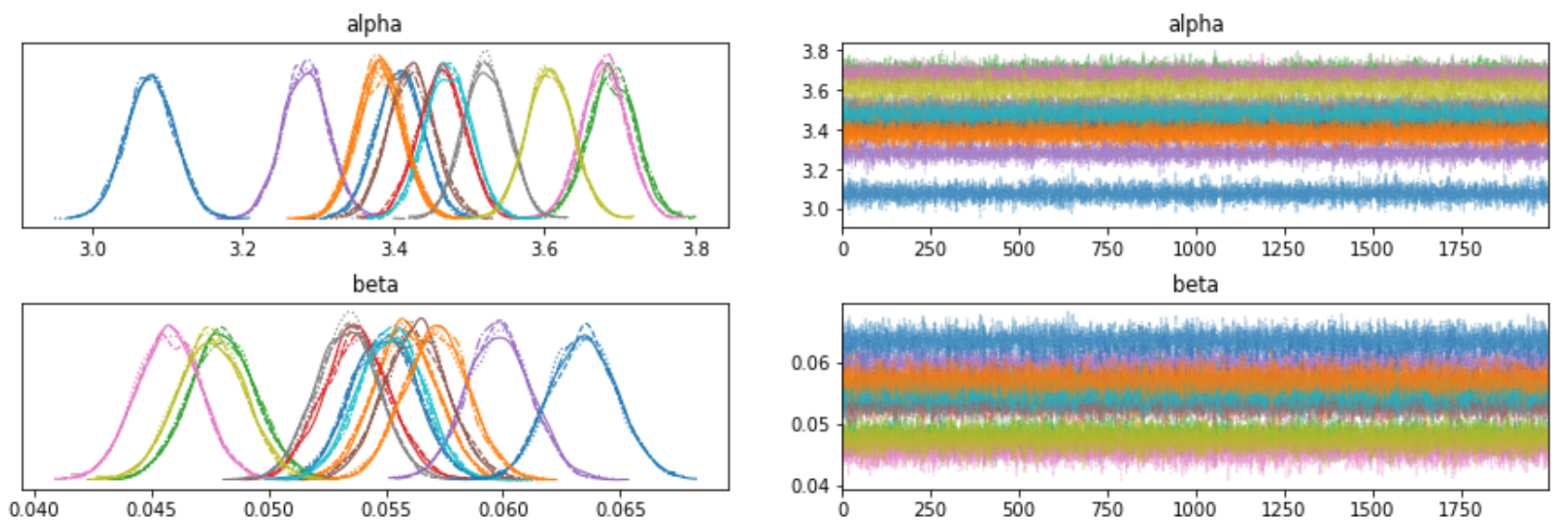
	rank	loo	p_loo	d_loo	weight	se	dse
varying slopes	0	-15557.545038	29.396342	0.000000	8.172186e-01	190.105625	0.000000
varying intercepts	1	-15629.400655	17.039457	71.855617	5.087248e-12	189.518973	13.373548
simple	2	-15757.707232	6.678447	200.162195	1.827814e-01	190.474747	26.581303

See a further improvement in model fit from varying slopes

Covarying parameters

Slopes and intercepts

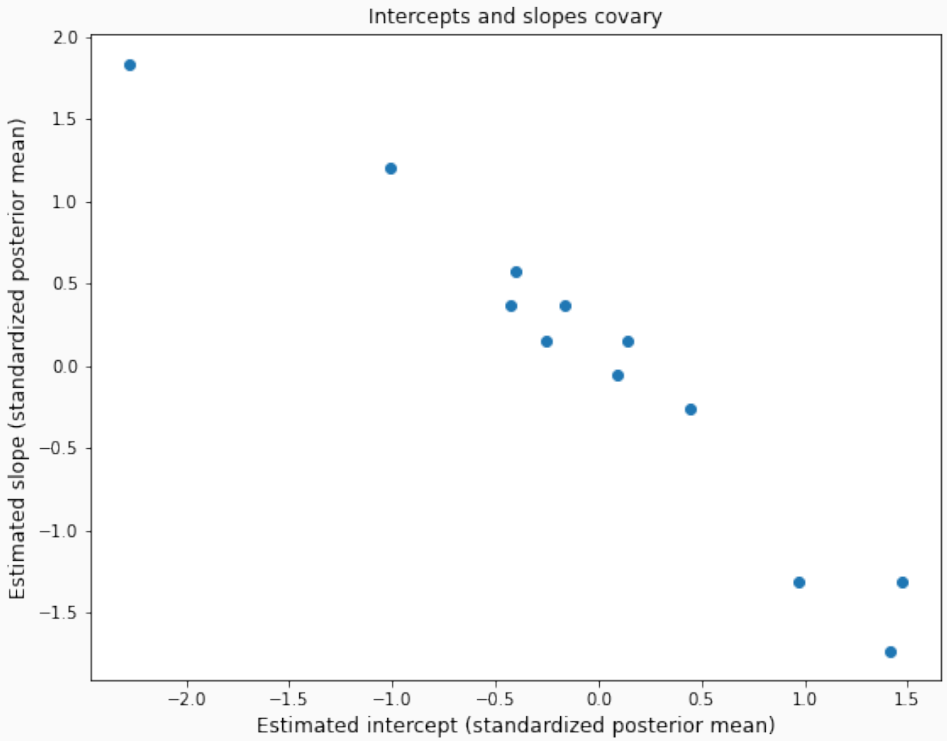
Here is part of the traceplot from the varying-slopes model:



Colors are the same. What do we see?

Covariance between parameters

- Intercepts vary across stations
- Slopes vary across stations
- Intercepts and slopes are correlated



A station with a high baseline ozone sees a smaller relative effect with temperature

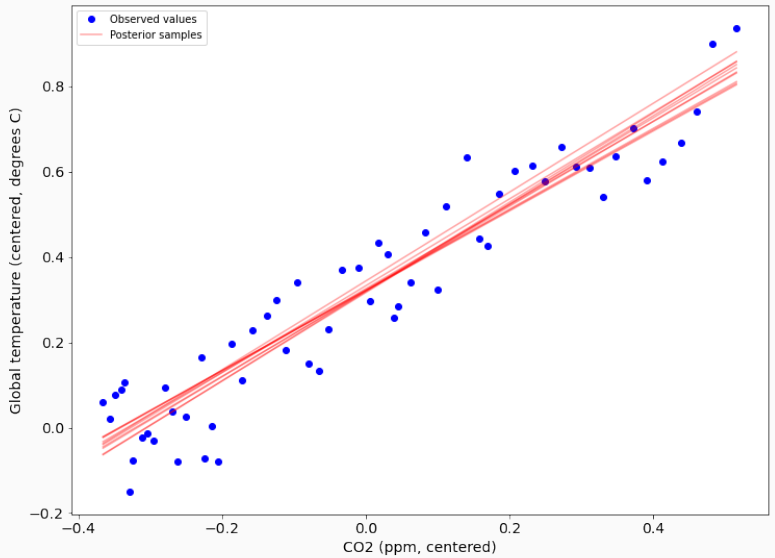
Covariance between parameters

Right now:

- slopes and intercepts are *a priori* independent
- we can see the negative association in the posterior, but the model cannot learn that pattern
- if we add a new monitoring station, it restarts with the same independence assumption:
  - the model knows something about typical intercepts
  - the model knows something about typical slopes
  - the model knows nothing about the relationship between them

Varying slopes and covariance

Covariance across parameter types is common for these models:



Each pair α, β represents a line

- They all have to stay close to the data
- If you change the slope you change the intercept

Covariance between parameters

In order to capture this effect, we need to explicitly include correlations in the model:

- Instead of thinking of 12 different intercepts and 12 different slopes, think of 12 intercept-slope pairs
- Right now, α_i, β_i are *a priori* normally distributed
- Replace the normal priors on α, β with multivariate normal

Multivariate normal distribution

Multivariate normal distribution:

- Generalizes normal distribution to produce vectors instead of scalars
- Specified by a vector of means and a covariance matrix
  - mean vector: contains mean of each component
  - covariance matrix: contains variance of each component, and covariance of each pair of components

Multivariate normal distribution

2×2 covariance matrix:

$$\Sigma = \begin{pmatrix} \sigma_a^2 & \sigma_a \sigma_b \rho_{ab} \\ \sigma_a \sigma_b \rho_{ab} & \sigma_b^2 \end{pmatrix}$$

- σ_a^2 – variance of a
- σ_b^2 – variance of b
- ρ_{ab} – correlation of a and b

The model

$$
\begin{aligned}
\log O_3 &\sim \text{Normal}(\mu, \sigma) \\
\mu &= \alpha_j + \beta_j T_{\max} \\
\begin{pmatrix} \alpha_j \\ \beta_j \end{pmatrix} &\sim \text{MVNormal}\left(\begin{pmatrix} \mu_\alpha \\ \mu_\beta \end{pmatrix}, \Sigma\right) \\
\Sigma &= \begin{pmatrix} \sigma_\alpha & 0 \\ 0 & \sigma_\beta \end{pmatrix} \mathsf{R} \begin{pmatrix} \sigma_\alpha & 0 \\ 0 & \sigma_\beta \end{pmatrix} \\
\mu_\alpha &\sim \text{Normal}(0, 2) \\
\mu_\beta &\sim \text{Normal}(0, 0.1) \\
\sigma_\alpha &\sim \text{HalfCauchy}(1) \\
\sigma_\beta &\sim \text{HalfCauchy}(1) \\
\sigma &\sim \text{HalfCauchy}(1) \\
\mathsf{R} &\sim \mathit{LKJ}(2)
\end{aligned}
$$

Multivariate normals and covariance matrices

Multivariate normal distribution

Multivariate normal:

- Generalization of the ordinary normal distribution to produce vectors
- Ordinary normal: parameterized by mean and variance
- MV normal: parameterized by mean vector and covariance matrix

Covariance matrix

m-by-m covariance matrix: for m components, m variables all varying together

- m standard deviations / variances
- $(m^2 - m)/2$ correlations / covariances
- total: $m(m+1)/2$ parameters

Covariance matrix

Covariance: measures how two random variables vary "together"

Covariance of two random variables:

$$\mathrm{Cov}\,(X, Y) = E[(X - E[X])(Y - E[Y])]$$

What's the covariance of X with itself?

$$\mathrm{Cov}\,(X, X) = E[(X - E[X])^2] = \sigma_X^2$$

Covariance matrix

Covariance matrix of $X_1, X_2, \ldots$:

$$\begin{pmatrix} \operatorname{Cov}(X_1, X_1) & \operatorname{Cov}(X_1, X_2) & \operatorname{Cov}(X_1, X_3) & \ldots \\ \operatorname{Cov}(X_2, X_1) & \operatorname{Cov}(X_2, X_2) & \operatorname{Cov}(X_2, X_3) & \ldots \\ \ldots & & & \end{pmatrix}$$

Covariance matrix

We can write this in terms of variances and correlations:

$$\begin{pmatrix} \sigma_1^2 & \sigma_1\sigma_2\rho_{12} & \sigma_1\sigma_3\rho_{13} & \dots \\ \sigma_1\sigma_2\rho_{12} & \sigma_2^2 & \sigma_2\sigma_3\rho_{23}\dots & \\ \dots & & & \end{pmatrix}$$

Example

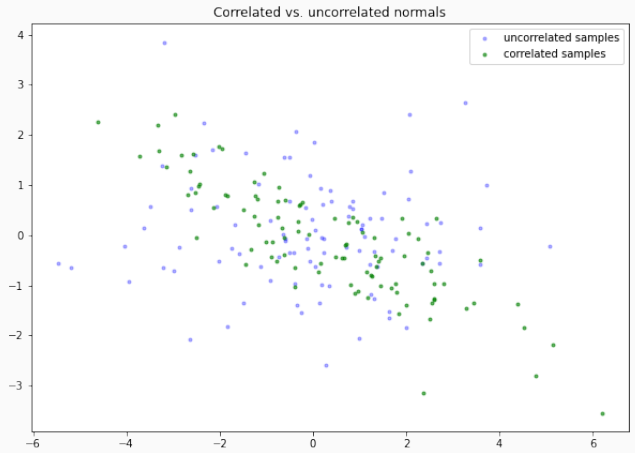
Say $\sigma_1^2 = 4, \sigma_2^2 = 1, \rho_{12} = -0.8$; then the covariance matrix is

$$\Sigma = \begin{pmatrix} 4 & -1.6 \\ -1.6 & 1 \end{pmatrix}$$

What does a sample from $\mathrm{MVNormal}(0, \Sigma)$ look like?

Example

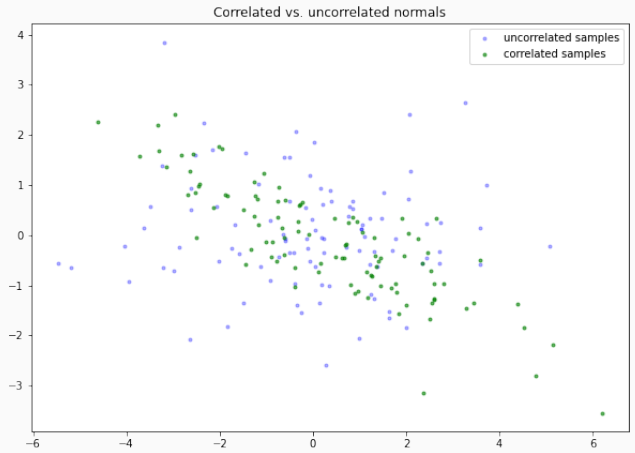
Compare to two independent normals with the same mean, variance:



What would the covariance matrix be for the independent samples?

Example

Compare to two independent normals with the same mean, variance:

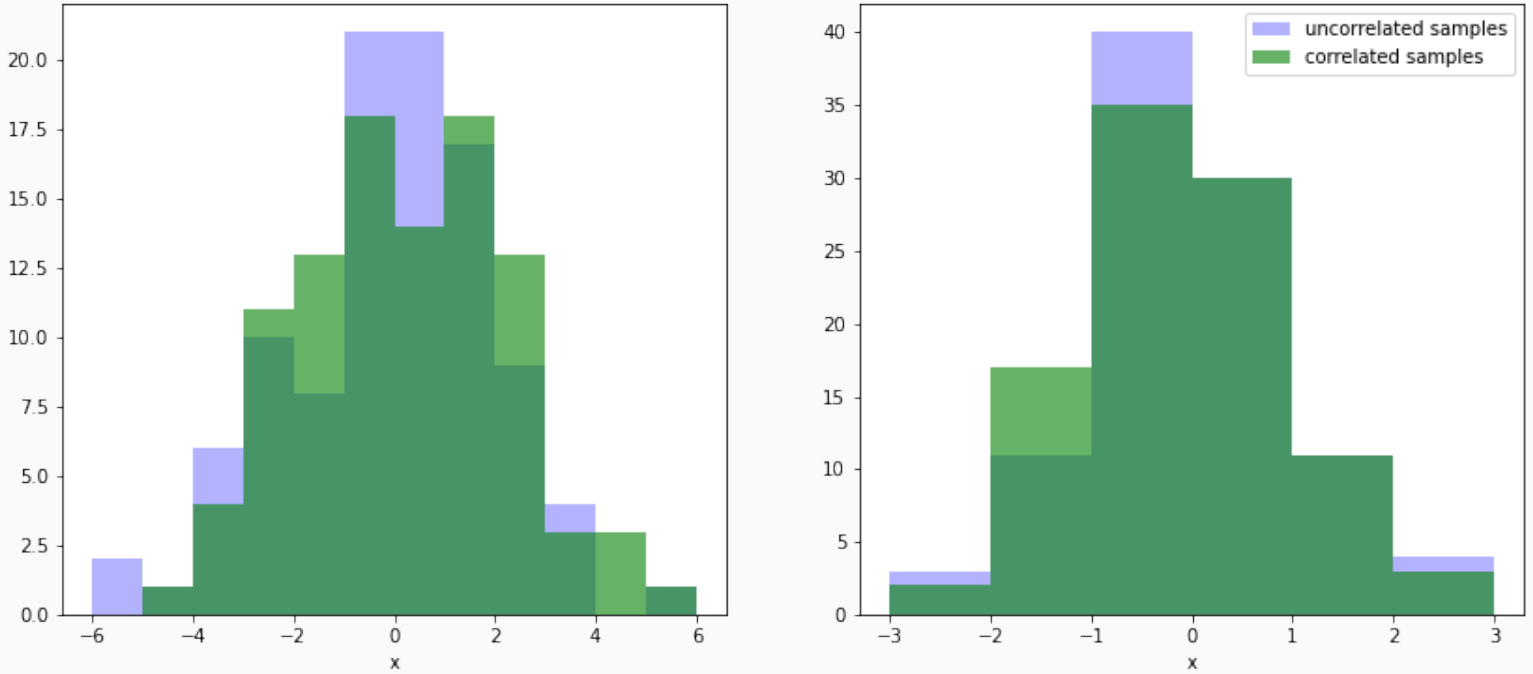


What would the covariance matrix be for the independent samples?

$$\Sigma = \begin{pmatrix} 4 & 0 \\ 0 & 1 \end{pmatrix}$$

Example

Compare the marginal distributions:



Can't tell apart! Marginals are $N(0, 2)$ and $N(0, 1)$ in both cases.

Priors for covariance matrices

Wishart distribution

Previously common: Wishart/inverse-Wishart distribution

Why not to use it:

- Often doesn't reflect our prior beliefs about correlations
- Can't decouple correlation and variance of individual variances
- Computationally hard for MCMC samplers
- Conjugacy not that big a deal anyway

Factoring the covariance matrix

Matrix multiplication interlude:

$$\begin{pmatrix} a_{11} & a_{12} & \dots \\ a_{21} & a_{22} & \dots \\ \vdots & \vdots & \end{pmatrix} \begin{pmatrix} b_{11} & b_{12} & \dots \\ b_{21} & b_{22} & \dots \\ \vdots & \vdots & \end{pmatrix} = ?$$

Matrix multiplication

Grab 1st row and 1st column:

$$\begin{pmatrix} a_{11} & a_{12} & \dots \\ a_{21} & a_{22} & \dots \\ \vdots & \vdots & \end{pmatrix} \begin{pmatrix} b_{11} & b_{12} & \dots \\ b_{21} & b_{22} & \dots \\ \vdots & \vdots & \end{pmatrix} = ?$$

Matrix multiplication

Grab 1st row and 1st column:

$$\begin{pmatrix} a_{11} & a_{12} & \dots \\ a_{21} & a_{22} & \dots \\ \vdots & \vdots & \end{pmatrix} \begin{pmatrix} b_{11} & b_{12} & \dots \\ b_{21} & b_{22} & \dots \\ \vdots & \vdots & \end{pmatrix} = ?$$

Multiply componentwise and add:

$$\text{product entry} = a_{11}b_{11} + a_{12}b_{21} + a_{13}b_{31} + \dots$$

Matrix multiplication

2x2 example:

$$\begin{pmatrix} A & B \\ C & D \end{pmatrix} \quad \begin{pmatrix} a & b \\ c & d \end{pmatrix} \quad \begin{pmatrix} & \\ & \end{pmatrix}$$

Matrix multiplication

2x2 example:

$$\begin{pmatrix} A & B \\ C & D \end{pmatrix} \quad \begin{pmatrix} a & b \\ c & d \end{pmatrix}$$

$$\begin{pmatrix} Aa + Bc & \\ & \end{pmatrix}$$

Matrix multiplication

2x2 example:

$$\begin{pmatrix} A & B \\ C & D \end{pmatrix} \quad \begin{pmatrix} a & b \\ c & d \end{pmatrix}$$

$$\begin{pmatrix} Aa + Bc & Ab + Bd \\ Ca + Dc & Cb + Dd \end{pmatrix}$$

2×2 **covariance matrix**

We can factor the covariance matrix:

$$\begin{pmatrix} \sigma_1 & 0 \\ 0 & \sigma_2 \end{pmatrix} \begin{pmatrix} 1 & \rho_{12} \\ \rho_{12} & 1 \end{pmatrix} \begin{pmatrix} \sigma_1 & 0 \\ 0 & \sigma_2 \end{pmatrix}$$

2×2 **covariance matrix**

We can factor the covariance matrix:

$$\begin{pmatrix} \sigma_1 & 0 \\ 0 & \sigma_2 \end{pmatrix} \begin{pmatrix} 1 & \rho_{12} \\ \rho_{12} & 1 \end{pmatrix} \begin{pmatrix} \sigma_1 & 0 \\ 0 & \sigma_2 \end{pmatrix}$$

So we have two parts:

- diagonal matrix of standard deviations
- correlation matrix

Priors:

- Standard deviations are easy (Exp, HC, etc.)
- Correlations are a little trickier

Why do we need a fancy prior?

Need a prior for the correlation matrix R, and we can't just pick anything:

- Say we have 3 variables, x, y, z
- 3 pairwise correlations:
  - $\rho_{xy} = 0.9$
  - $\rho_{xz} = 0.8$
  - $\rho_{yz} = -0.9$
- What's wrong?

Why do we need a fancy prior?

Need a prior for the correlation matrix R, and we can't just pick anything:

- Say we have 3 variables, x, y, z
- 3 pairwise correlations:
  - $\rho_{xy} = 0.9$
  - $\rho_{xz} = 0.8$
  - $\rho_{yz} = -0.9$
- What's wrong?
- Not every matrix can be a correlation / covariance matrix!

LKJ disribution

LKJ distribution: a prior for correlation matrices.

- Named for Lewandowski, Kurowicka, and Joe (2009)
- Depends on a shape parameter η, which specifies difference from zero correlations
  - $\eta = 1$: uniform correlation matrices
  - $\eta > 1$: suppress big correlations
  - $\eta < 1$: prefer big correlations
- PDF:

$$p(\mathrm{R}|\eta) \propto (\det \mathrm{R})^{\eta - 1}$$

- In our simple case:

$$p(\rho|\eta = 2) \propto 1 - \rho^2$$

LKJ distribution in PyMC3

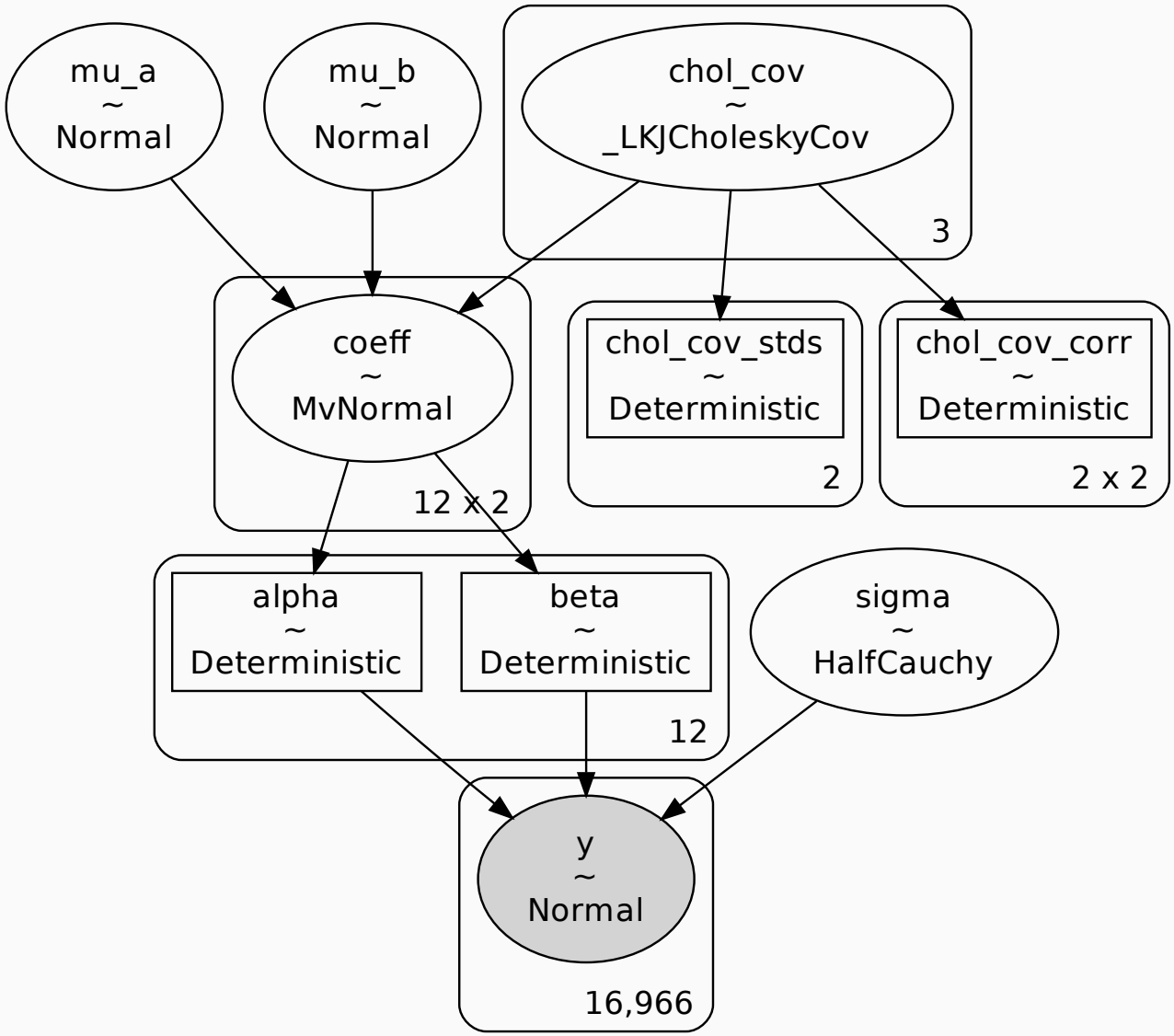
```
import theano.tensor as tt
with pm.Model() as cov_model:
    mu_a = pm.Normal('mu_a', 0, 2)
    mu_b = pm.Normal('mu_b', 0, 0.1)
    sigma = pm.HalfCauchy('sigma', 1)

    sd_dist = pm.HalfCauchy.dist(1)
    chol, corr, sd = pm.LKJCholeskyCov('chol_cov', eta = 2,
            n = 2, sd_dist = sd_dist, compute_corr = True)

    # tt.stack makes a vector; chol holds the covariance matrix
    coeff = pm.MvNormal('coeff', mu = tt.stack([mu_a, mu_b]),
                        chol=chol, shape=(12, 2))
    alpha = pm.Deterministic('alpha', coeff[:, 0])
    beta = pm.Deterministic('beta', coeff[:, 1])

    theta = alpha[data['station_id']] +
            beta[data['station_id']] * data['TEMP']
    y_ = pm.Normal('y', mu=theta, sigma = sigma, observed = data['log_ozone'])
```

Plate diagram



Results

When we run this model:

```
Sampling 4 chains for 500 tune and 500 draw iterations (2_000 + 2_000 draws total) took 172 seconds.
The acceptance probability does not match the target. It is 0.9894609716484106, but should be close t
The acceptance probability does not match the target. It is 0.9819508058560595, but should be close t
There were 457 divergences after tuning. Increase `target_accept` or reparameterize.
The acceptance probability does not match the target. It is 0.085127247008543, but should be close to
The acceptance probability does not match the target. It is 0.9893641325448185, but should be close t
The rhat statistic is larger than 1.4 for some parameters. The sampler did not converge.
The estimated number of effective samples is smaller than 200 for some parameters.
```

- Ugly stuff!
- But, we've run into this problem with hierarchical models before, and we know some tricks

Non-centered parameterization

Remember when we did the 8 schools model:

Bad:

$$\theta \sim \text{Normal}(\mu, \sigma)$$

Good:

$$\theta = \mu + \sigma\eta$$

$$\eta \sim \text{Normal}(0, 1)$$

- Essentially, modeling the standardized θs instead of modeling the θs directly:

$$x_i = \mu_i + \sigma z_i$$

- Mathematically equivalent, but much easier for the sampler

Non-centered parameterization

So if we standardized the usual hierarchical normal model using

$$\theta = \mu + \sigma\eta$$

what do we do for the multivariate version?

- μ is easy – just use the mean vector
- η is easy – just use a bivariate standard normal
- σ is the square root of the variance, so we should replace it with the square root of the covariance

Cholesky decomposition

When a matrix is symmetric and positive definite (e.g. a covariance matrix) it can be factored:

$$\Sigma = \mathsf{L}(\mathsf{L})^T$$

where L is a lower triangular matrix (hence L).

If z is a vector of independent standard normals, then Lz has the distribution of $\mathrm{MVNormal}(0, \Sigma)$ – exactly what we want!

$$\begin{pmatrix} \alpha_j \\ \beta_j \end{pmatrix} = \begin{pmatrix} \alpha_j \\ \beta_j \end{pmatrix} + \mathsf{L} \begin{pmatrix} z_{\alpha,i} \\ z_{\beta,i} \end{pmatrix}$$

Good news: PyMC3 already gives us the Cholesky factor (because it's useful for numerical stability anyway)

Non-centered parameterization

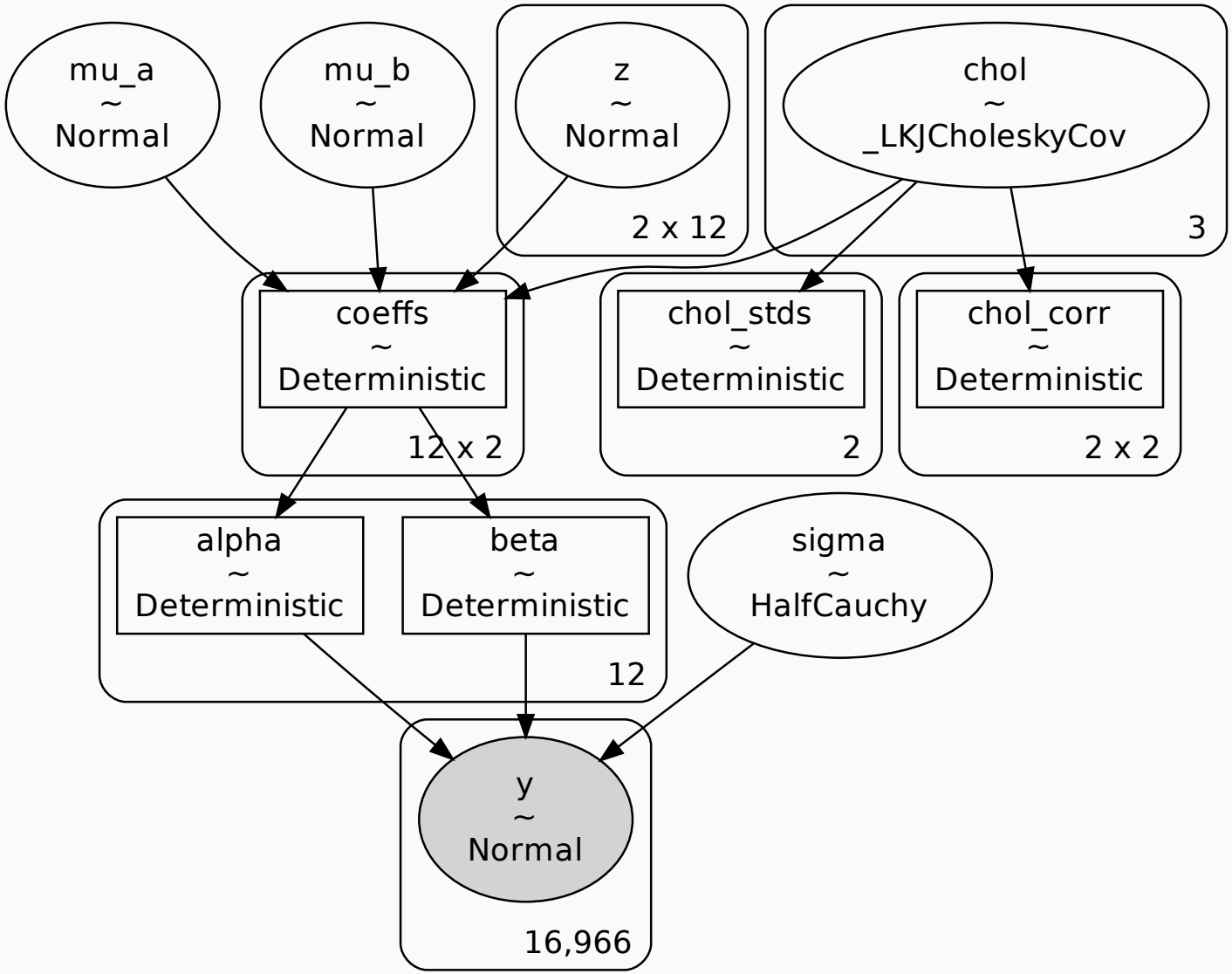
Finishing the non-centered parameterization:

```
import theano.tensor as tt
with pm.Model() as cov_model:
    # clipped out same stuff

    chol, corr, sd = pm.LKJCholeskyCov('chol', eta = 2, n = 2,
     sd_dist = sd_dist, compute_corr = True)
    # make standard normals
    z = pm.Normal('z', 0, 1, shape = (2, 12))
    # tt.dot is matrix multiplication
    coeffs = pm.Deterministic('coeffs',
             tt.stack([mu_a, mu_b]) + tt.dot(chol, z).T)

    # clipped out same stuff
```

Plate diagram

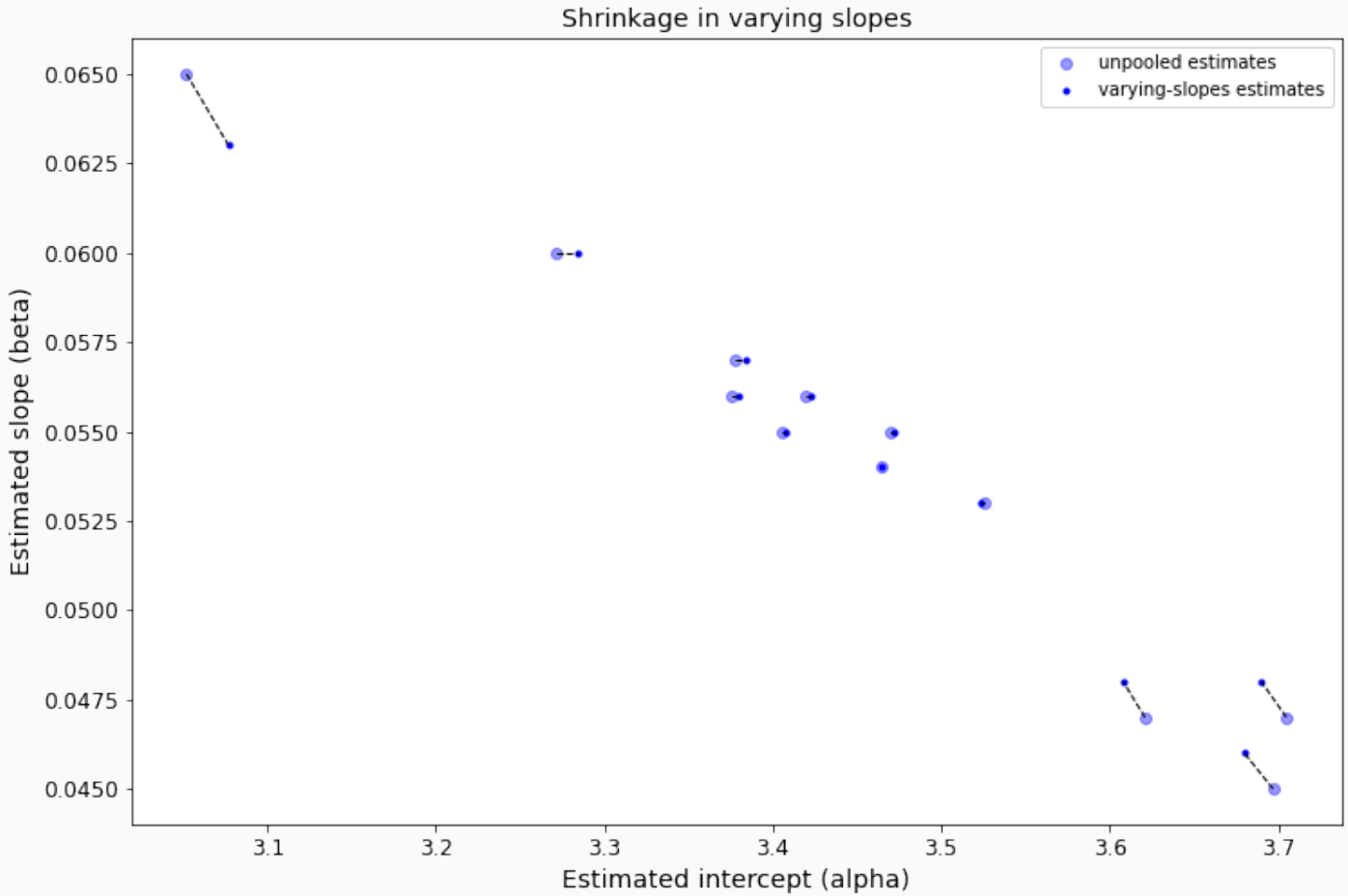


Results

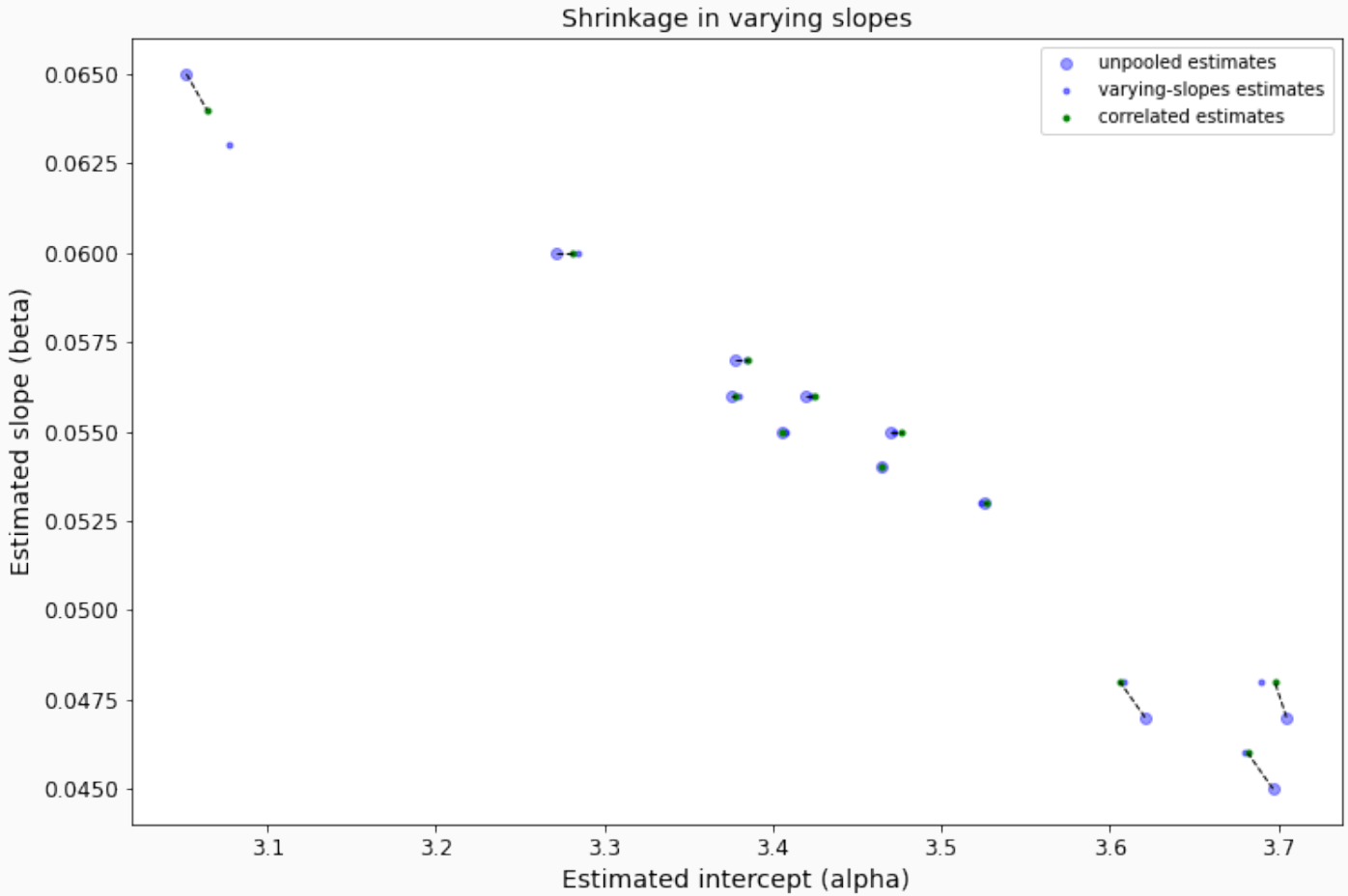
Result: no divergences, good Rhats, sampling takes about 2/3 the time

- α, β estimates similar to basic varying-slopes model
- Estimated correlation $\rho \approx -0.9$
- Two-dimensional shrinkage

Shrinkage with no correlations



Shrinkage with no correlations



What other correlations?

- These correlations appear as artifacts of the mathematical model
- Are there places we might expect to see correlations due to the physical nature of the data?
  - Data come from air quality monitoring stations
  - Monitoring stations are located in physical space
  - Some are far apart, some are close together
  - Might reasonably expect that stations located close together are correlated
- Can we model this type of behavior specifically?

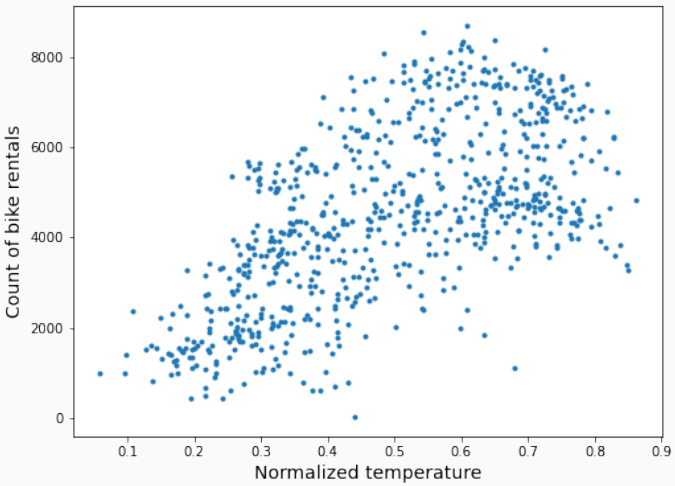
Example: bike share data

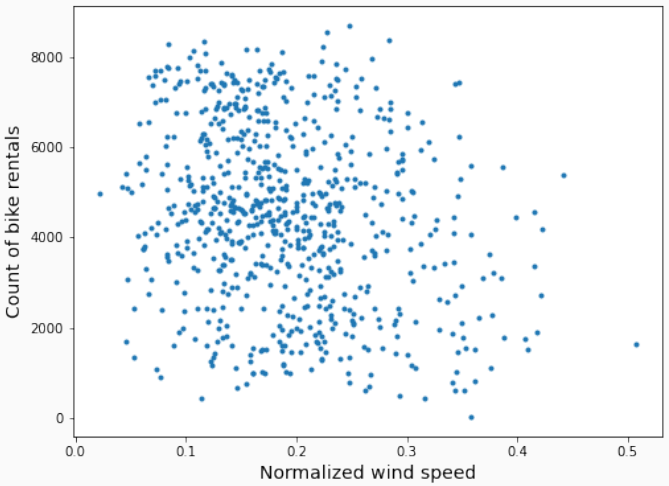
Bike share programs

- Bike share programs: short-term rentals for bicycles
- Have data on count of renters, along with daily weather data

Goal: estimate influence of temperature, wind speed

Make a plot to check reasonableness





Simple Poisson regression model

We have count data, so use Poisson regression:

$$
\begin{aligned}
y_j &\sim \text{Poisson}(\lambda_j) \\
\log \lambda_j &= \alpha + \beta_T T_j + \beta_w w_j \\
\alpha &\sim \text{Normal}(0, 5) \\
\beta_T &\sim \text{Normal}(0, 1) \\
\beta_w &\sim \text{Normal}(0, 1)
\end{aligned}
$$

Results and predictive check

- Summary:

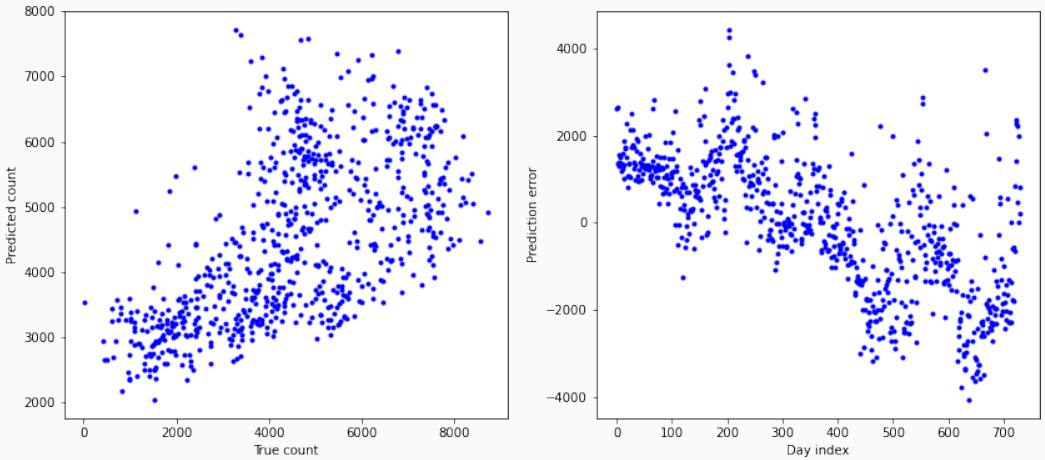
	mean	sd	hdi_3%	hdi_97%
alpha	7.812	0.002	7.808	7.817
beta_temp	1.450	0.003	1.444	1.456
beta_wind	-0.823	0.008	-0.837	-0.809

Results and predictive check

- Summary:

	mean	sd	hdi_3%	hdi_97%
alpha	7.812	0.002	7.808	7.817
beta_temp	1.450	0.003	1.444	1.456
beta_wind	-0.823	0.008	-0.837	-0.809

- Posterior predictive error vs. date:



Adding in varying intercepts

The posterior predictions show mediocre fit to data – in particular, prediction error clearly follows a trend over time.

Add in varying intercepts by month:

$$
\begin{aligned}
y_j &\sim \text{Poisson}(\lambda_j) \\
\log \lambda_j &= \alpha_{\text{month}(j)} + \beta_T T_j + \beta_w w_j \\
\beta_T &\sim \text{Normal}(0, 1) \\
\beta_w &\sim \text{Normal}(0, 1) \\
\alpha_{\text{month}(j)} &\sim ?
\end{aligned}
$$

Varying intercepts

We could simply use our usual strategy and do something like:

$$\alpha \sim \text{Normal}(\mu, \tau)$$

with some hyperpriors on μ, τ

- Usual multilevel strategy oriented around the idea of exchangeable groups
- Share information among groups
- Exchangeability: the model doesn't change if we permute the index of the groups
- Time points not really exchangeable

Varying intercepts

Alternative:

- Sample varying intercepts from a multivariate normal with correlations
- Here, we can impose some structure on the correlations:
  - Months closer in time are more similar
  - Months closer in time should have higher correlations
- How do we impose this? Put it into the covariance matrix

New model

Make α a multivariate normal:

$$\boldsymbol{\alpha} \sim \text{MVNormal}\left(\begin{pmatrix} 0 \\ 0 \\ \vdots \\ 0 \end{pmatrix}, \mathsf{K}\right)$$

- Covariance between α_i, α_j should depend on how close months i and j are in time
- So, K_{ij} should be a function of i, j

Covariance function

Set:

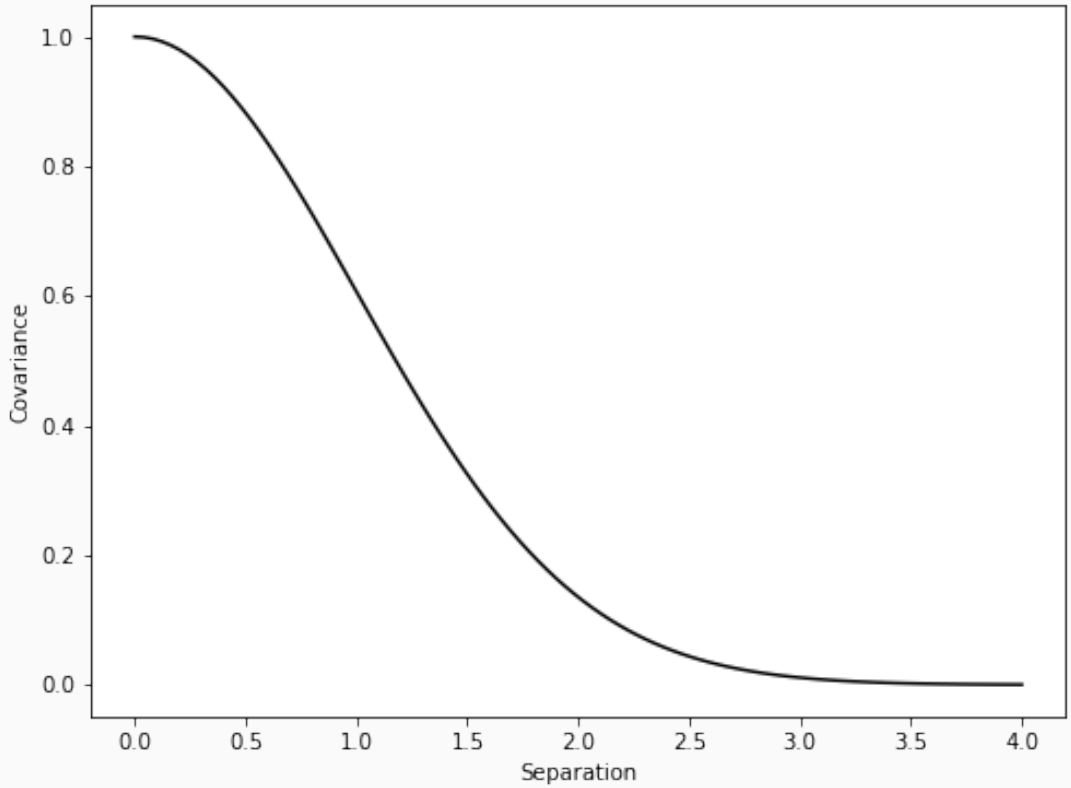
$$\mathsf{K}_{ij} = \eta^2 \exp\left(-\frac{(i-j)^2}{2\ell^2}\right) + \sigma^2 \delta_{ij}$$

Parameters:

- η^2 – magnitude of correlations
- ℓ^2 – length scale
- σ^2 – self variance
    - Even if your model doesn't need this, a small amount useful for numerical stability

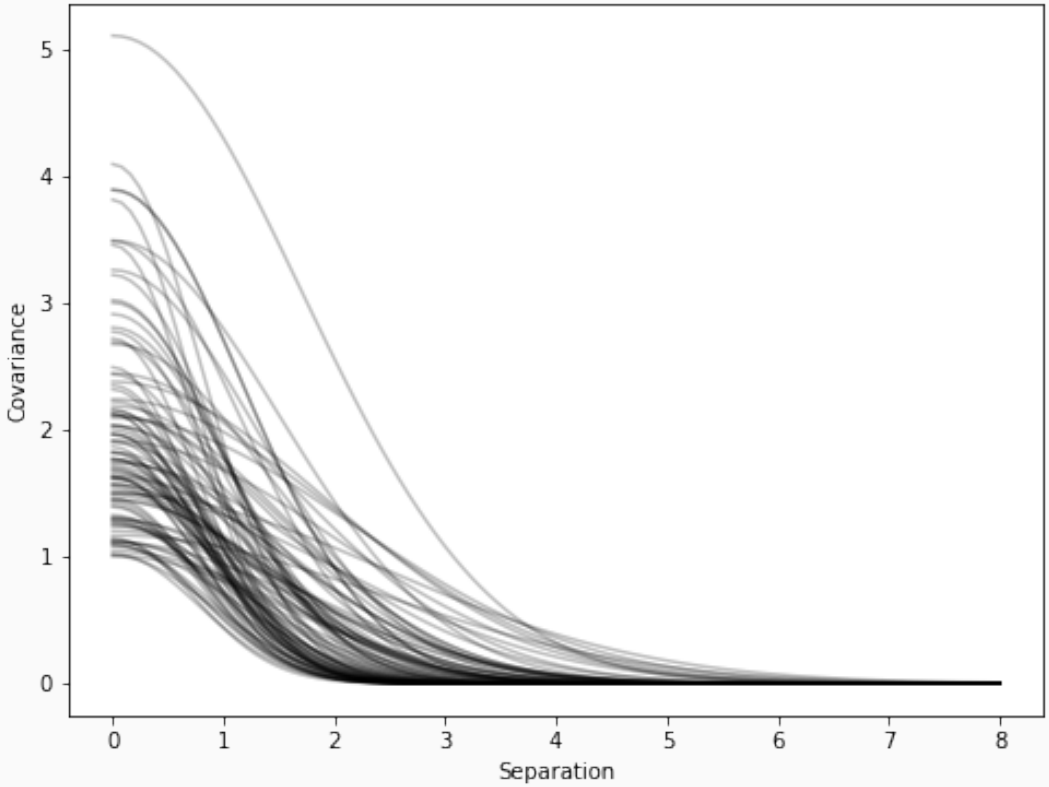
What does this look like?

What this means is the covariance between two αs is a function of their separation:



What does this look like?

Parameterized by varying η^2, ℓ^2:

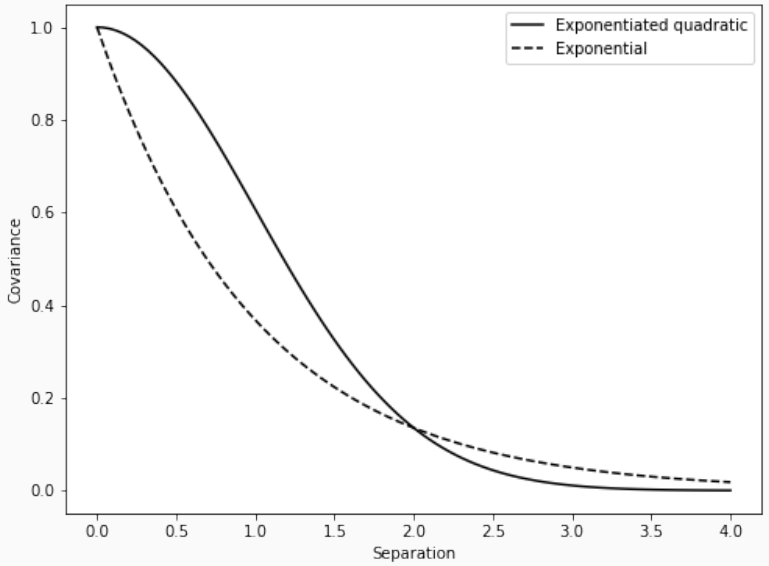


What about a different functional relationship?

The formula from before:

$$\mathsf{K}_{ij} = \eta^2 \exp\left(-\frac{(i-j)^2}{2\ell^2}\right) + \sigma^2\delta_{ij}$$

is an exponentiated quadratic; what about another form?



What to add to the model?

```
with pm.Model() as bike_model:
    beta_temp = pm.Normal('beta_temp', 0, 2)
    beta_wind = pm.Normal('beta_wind', 0, 2)
    eta = pm.Exponential('eta', 1)
    ls = pm.Exponential('ls', 4)

    Kij = (eta ** 2) * pm.math.exp(-(separation ** 2) / (ls ** 2)) + 0.01 * np.
    k = pm.MvNormal('k', mu=tt.zeros(24), cov=Kij, shape = 24)

    theta = pm.math.exp(k[bikes['month_index']] + beta_temp * bikes['temp']
            + beta_wind * bikes['windspeed'])

    y_ = pm.Poisson('y', theta, observed = bikes['cnt'])
```

`separation` is a 24×24 matrix with `i, j` entry equal to $|i - j|$

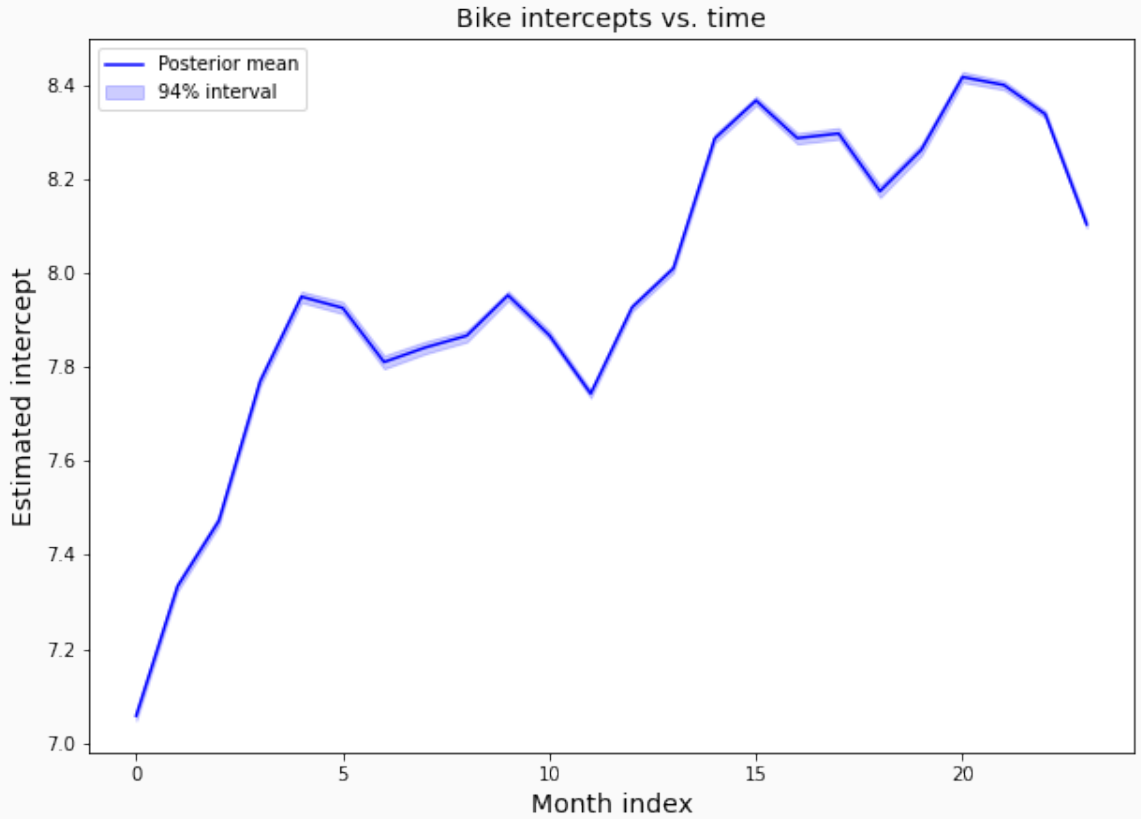
Results

Results from a summary table:

	mean	sd	hdi_3%	hdi_97%
beta_temp	0.965	0.008	0.951	0.982
beta_wind	-0.694	0.008	-0.708	-0.680
alpha[0]	7.058	0.005	7.048	7.069
alpha[1]	7.334	0.005	7.324	7.344
alpha[2]	7.472	0.005	7.463	7.482
alpha[3]	7.769	0.005	7.759	7.779
alpha[4]	7.949	0.006	7.938	7.960

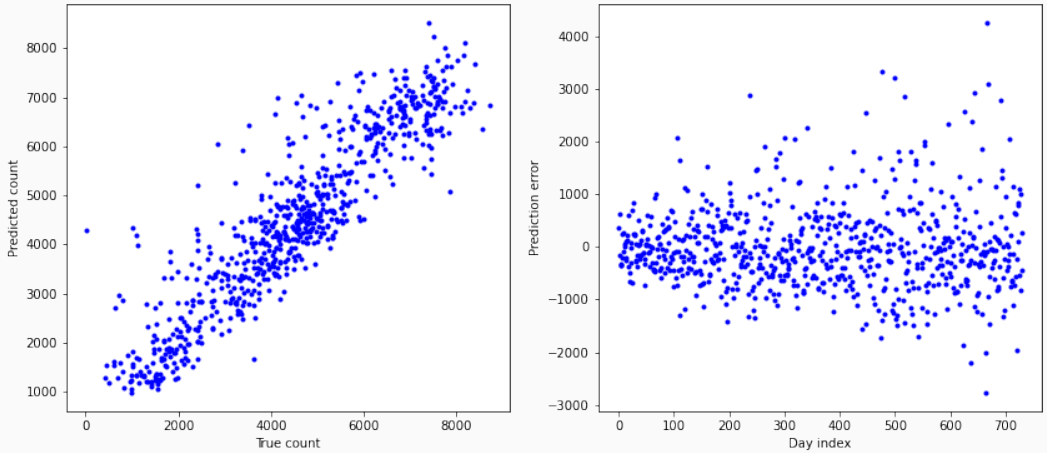
Intercepts over time

We can look at the intercepts estimated as a function of the month:



Posterior predictive check

Same predictive check as before:



Gaussian processes as random functions

Time grouping in the bike example

In the bike share example:

- We used k_{month} as our varying intercept
- Why monthly?

Time grouping in the bike example

In the bike share example:

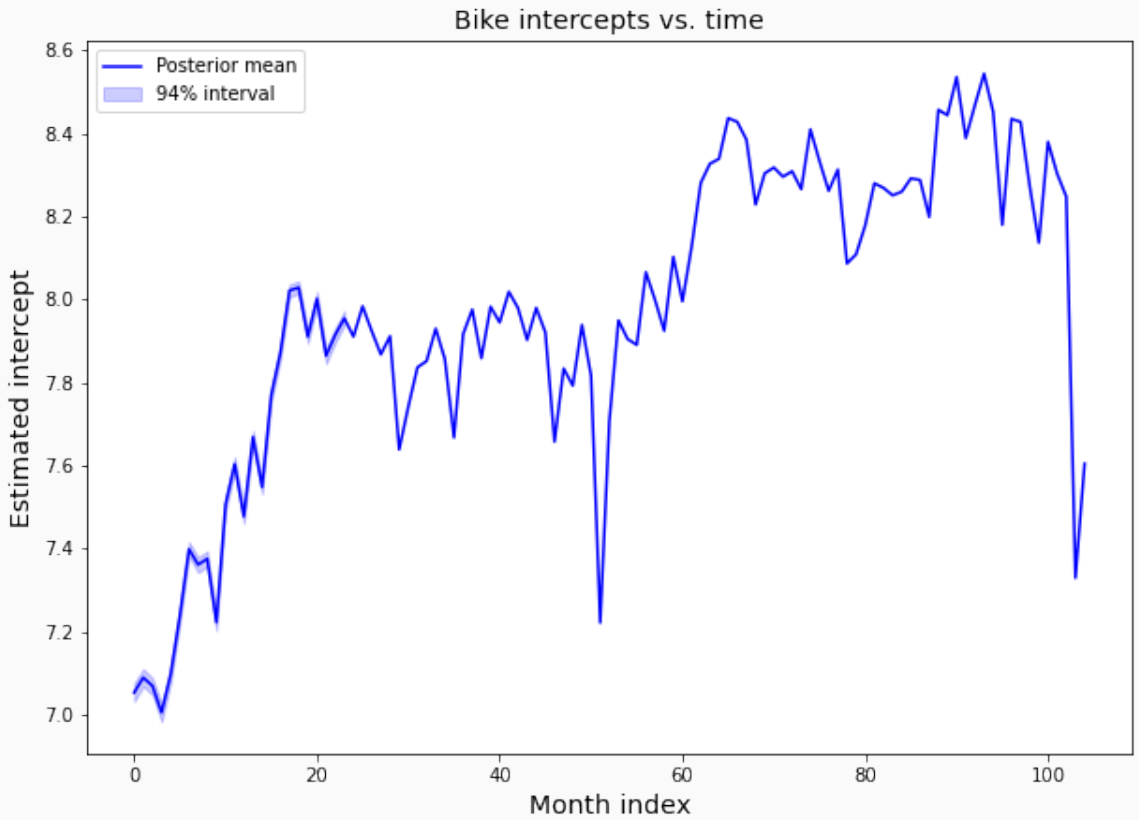
- We used k_{month} as our varying intercept
- Why monthly?

Try weekly instead:

- 105 varying intercepts
- Same approach: 105-dimensional multivariate normal; covariance matrix built in the same way

Intercepts over time

Now we get more resolution on the intercepts:



Gaussian process regression

- Weekly and monthly versions identical in spirit, just with different data resolution for the intercepts
- Unified way to think of this:

$$\log \lambda_j = \alpha(t_j) + \beta_T T_j + \beta_w w_j$$

  where α is a continuous function of time
- We're not trying to estimate a vector from observations of each component
- We're trying to estimate a function from several observations of function values

GP: the definition

A Gaussian process is a random *function* – i.e., we're really talking about a probability distribution on a space of functions.

The feature that makes a GP a GP: if you pick any n values of x, then the vector of function values $(\mu(x_1), \mu_(x_2), \ldots, \mu(x_n))$ has a multivariate normal distribution:

$$(\mu(x_1), \ldots \mu_(x_n)) \sim \mathrm{Normal}((m(x_1), \ldots, m(x_n)), K(x_1, \ldots, x_n))$$

The GP is determined by its mean function m and covariance K.

GP: the definition

Typically, the covariance matrix is determined by a function called the *kernel* $k(x, x')$.

- $k(x, x')$ determines how much the value of $\mu(x)$ depends on $\mu(x')$.
- Common (not universal) property: $k(x, x')$ depends on the distance between x, x'
- Idea: we're looking for continuous functions, so the values of $\mu(x), \mu(x')$ should be close if x, x' are close; but if they're far apart

Squared exponential covariance

Very common choice: squared exponential covariance function:

$$k(x, x') = \eta^2 \exp\left(-\frac{(x - x')^2}{2\ell^2}\right)$$

Covariance is high when $x - x'$ is small, falls off at longer ranges.

Hyperparameters:

- η: the maximum covariance
- ℓ: the *length scale*, controls how quickly covariance decays

In practice

How this is realized in practice:

- We have a set of observations $f(x_i)$
- GP property says

$$f(x_i) \sim \mathrm{MvNormal}((m(x_1), \ldots, m(x_n)), K(x_1, \ldots, x_n))$$

- So we evaluate the covariance function $k(x, x')$ at each pair of observed x values and use that to build a covariance matrix
- The Gaussian process distribution

$$\mathcal{GP}(\mu(x), k(x, x'))$$

  is really a prior distribution on the space of continuous functions

Summary

Summary:

- Using multivariate normal distributions, we can model parameters that are explicitly correlated
- Pool information across different types of parameters as well as different groups
- LKJ distribution gives a regularizing prior on correlation matrices
- Many data sets naturally include observations that should be correlated based on, e.g. time or distance
- Including these correlations amounts to estimating an underlying function
- $\mathcal{GP}$ is a prior distribution on a space of functions, parameterized by a mean function and covariance

Next time:

Summary

Today:

- Hierarchical linear regression models
  - Varying intercepts
  - Varying slopes

Next week:

- Multivariate normal mechanics
- Priors for covariance matrices
- Gaussian processes