Covariance and Gaussian processes

ISTA 410 / INFO 510: Bayesian Modeling and Inference

U. of Arizona School of Information

November 1, 2021

Outline

Last time:

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

Today:

- Modeling correlation between parameters
- Gaussian processes

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

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:

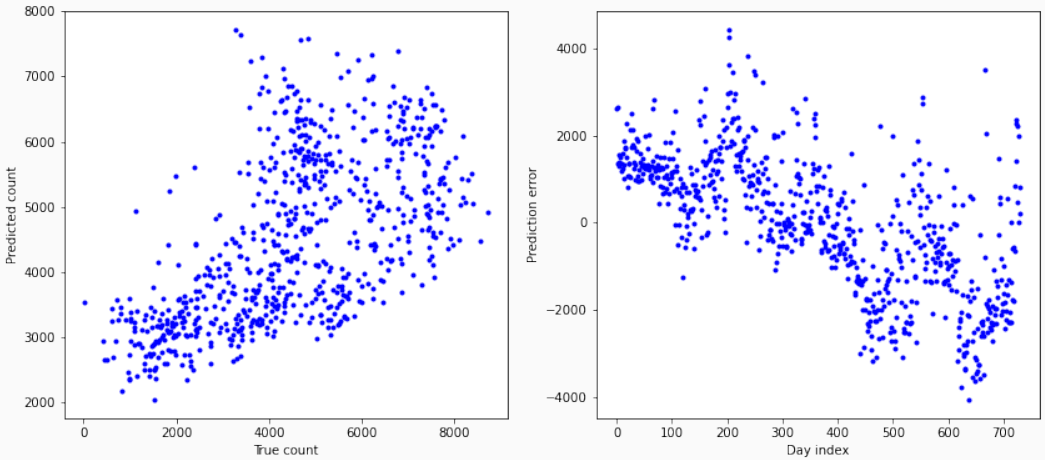
	mean	sd	hdi_3%	hdi_97%
alpha	7.812	0.002	7.808	7.817
beta_temp	1.450	0.003	1.444	1.456
beta_wind	-0.823	0.008	-0.837	-0.809

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:

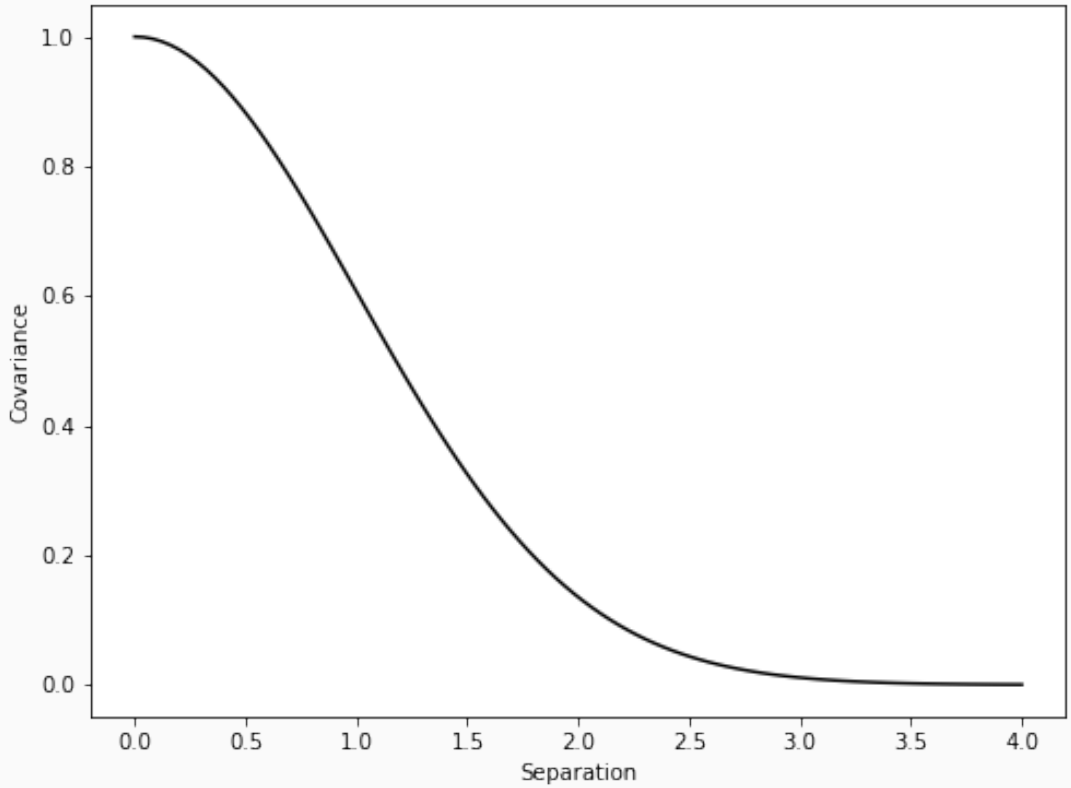
$$\mathsf{K}_{ij} = \eta^2 \exp\left(-\frac{(i-j)^2}{2\ell^2}\right) + \sigma^2 \delta_{ij}$$

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

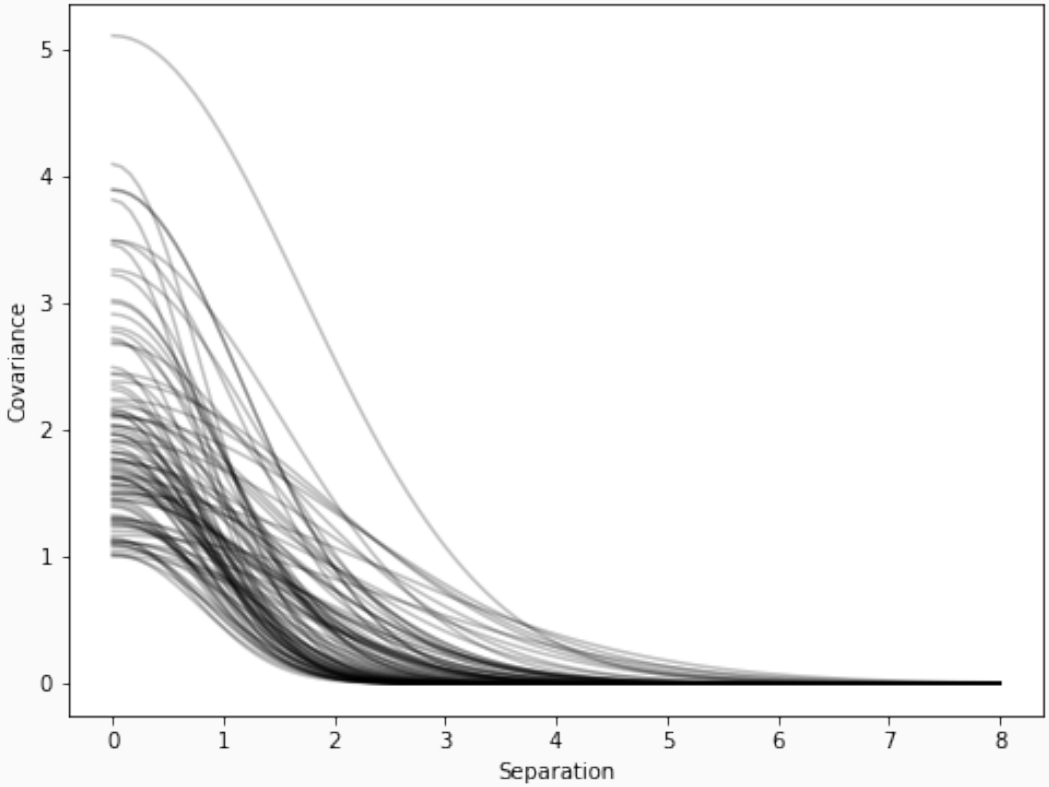
What does this look like?

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:

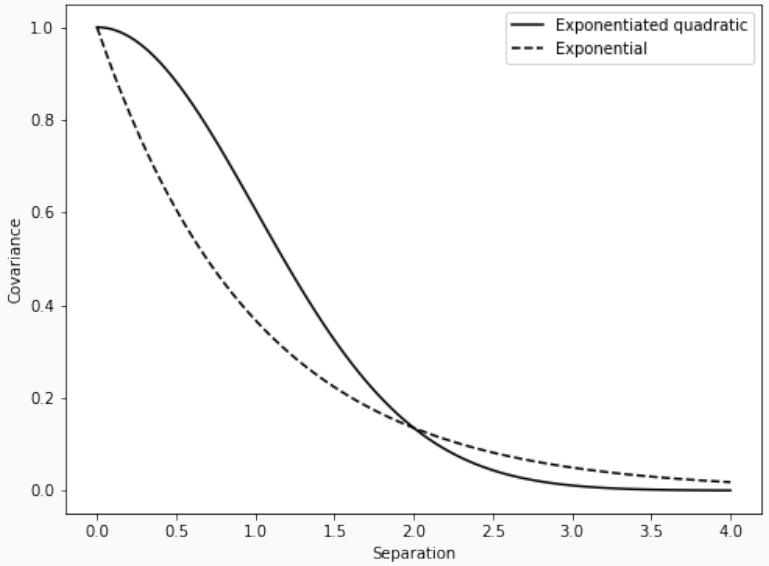


What about a different functional relationship?

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.eye(24))
    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 $|i - j|$

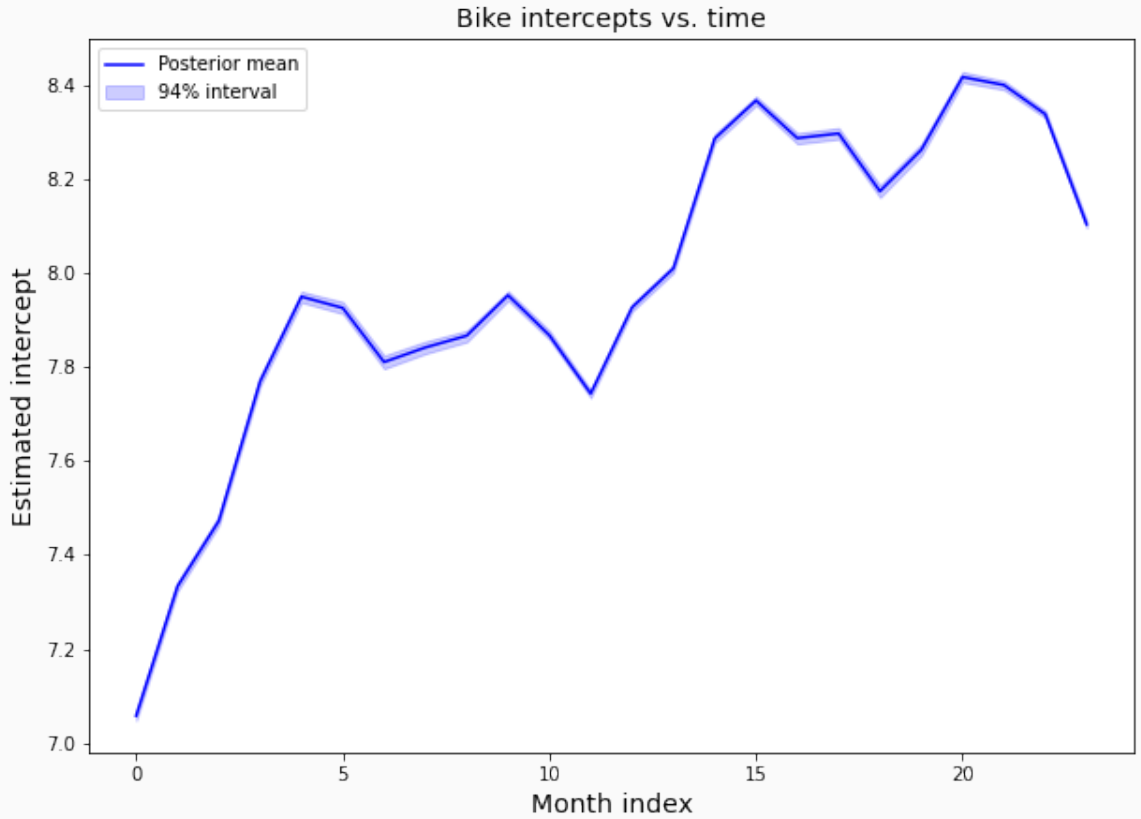
Results

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

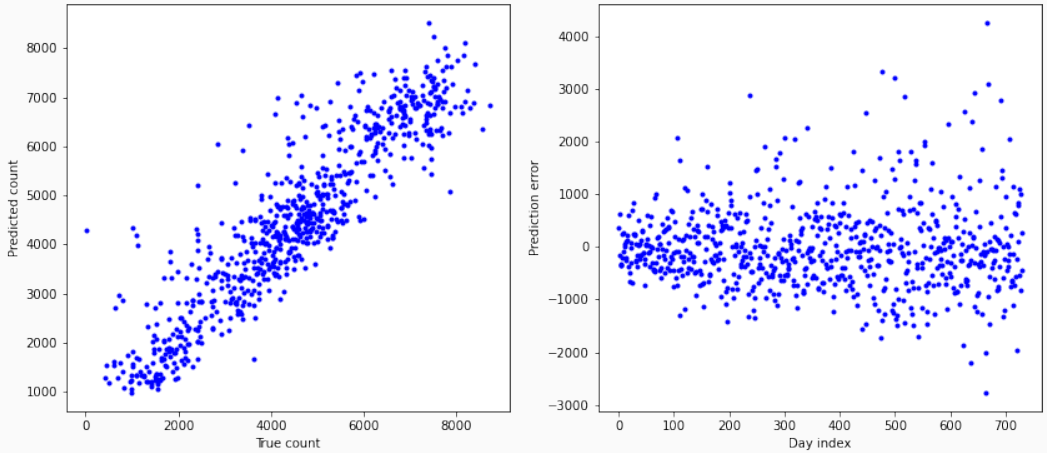
Intercepts over time

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:

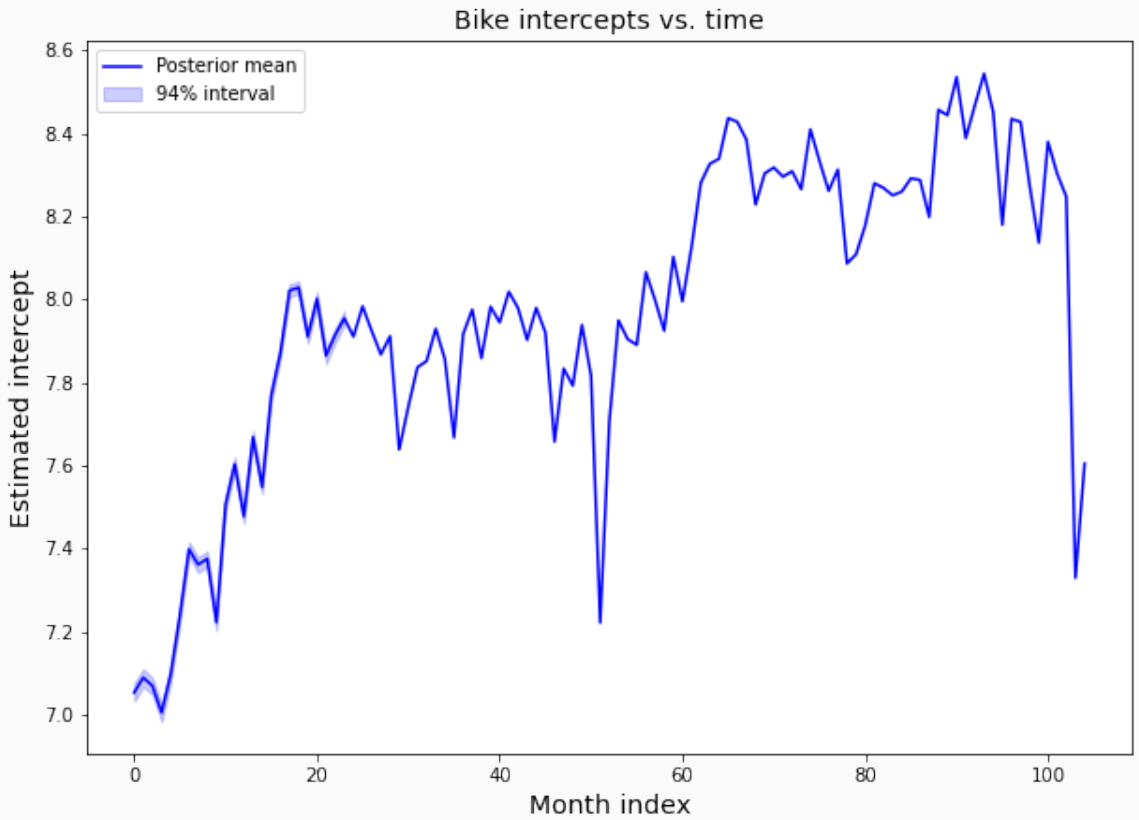
- We used k_{month} as our varying intercept
- Why monthly?

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 they don't have to be

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

Specifying a GP

- A MVNormal is specified by a mean vector and a covariance matrix
- A GP is specified by a mean function and a covariance function

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

  - What this means: any finite collection of function values $f(x_1), f(x_2), \ldots, f(x_n)$ has a MVNormal distribution:

$$f(\vec{x}) \sim \mathrm{MVNormal}(\mu(\vec{x}), \mathsf{K}(\vec{x}))$$

    where

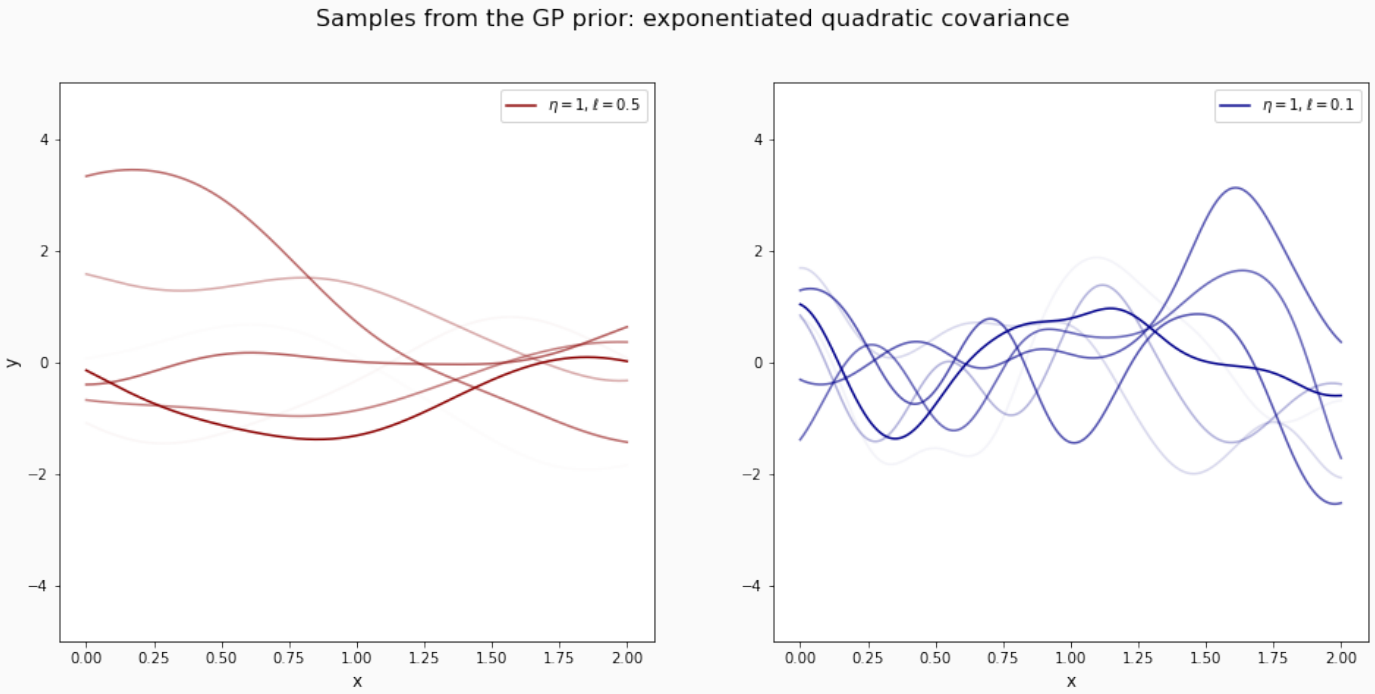
$$\mathsf{K}(\vec{x})_{ij} = k(x_i, x_j)$$

Choosing a covariance function

All of the interesting features of a GP come down to the covariance function:

- Common to leave the mean function as 0
- Covariance function influences various properties of the resulting function
    - Smoothness: related to how rapidly correlations decay over short distance
    - Length scale: related to how rapidly correlations decay over longer distances
    - Periodicity: put a periodic component in the correlation function
    - Linear or other trends: covariance that varies with distance from a fixed point

Draws from a prior

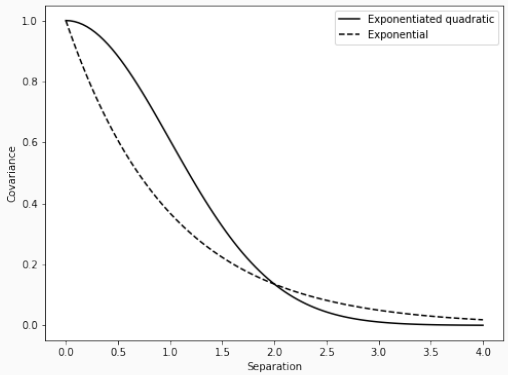


Smoothness

Last time: exponentiated quadratic vs exponential

- Exp. Quad:

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

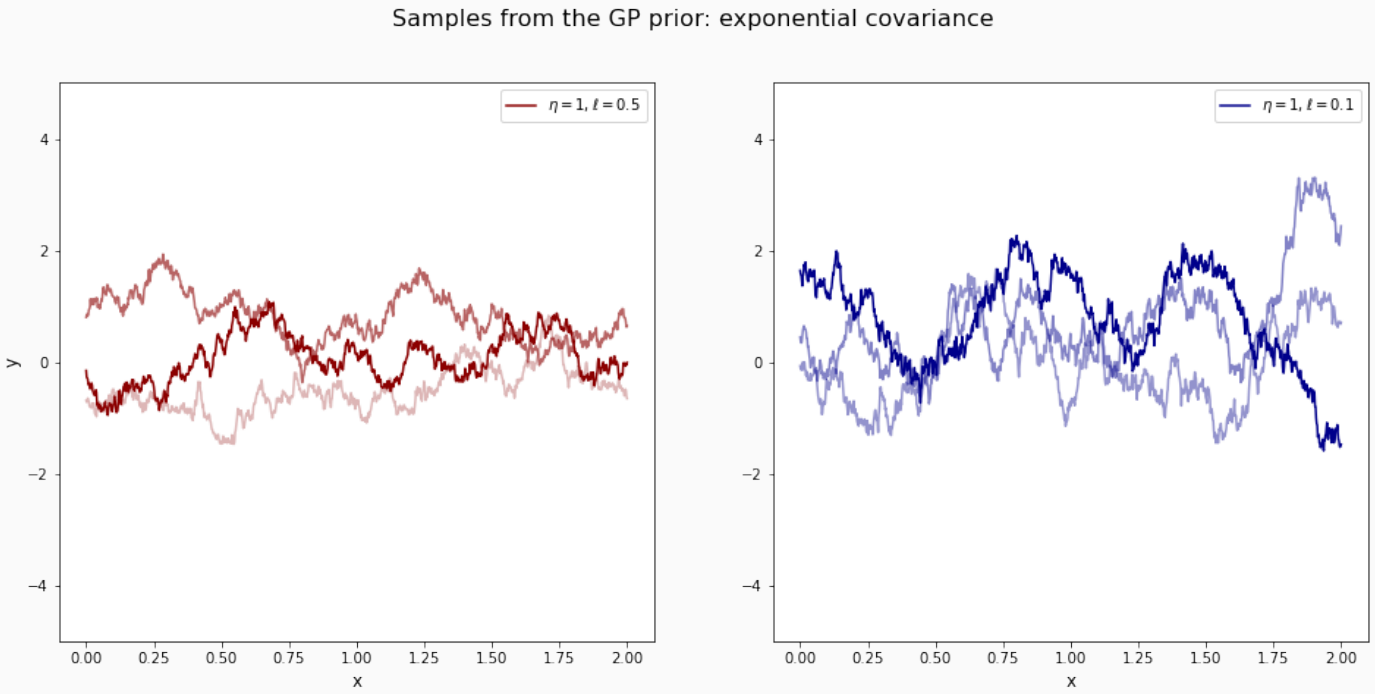


- Exponential:

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

  - Key difference: correlation for infinitesimally separated x, x'

Smoothness


Samples from the GP prior: exponential covariance
$\eta = 1, \ell = 0.5$
$\eta = 1, \ell = 0.1$
y
x
4
2
0
−2
−4
0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75
2.00

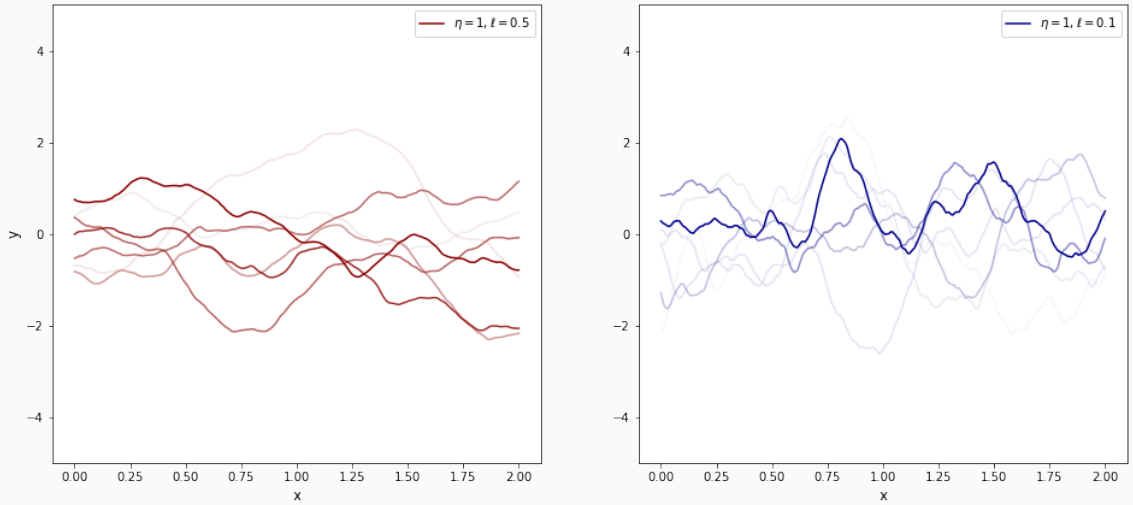
Matern kernels

Smoothness interpolation:

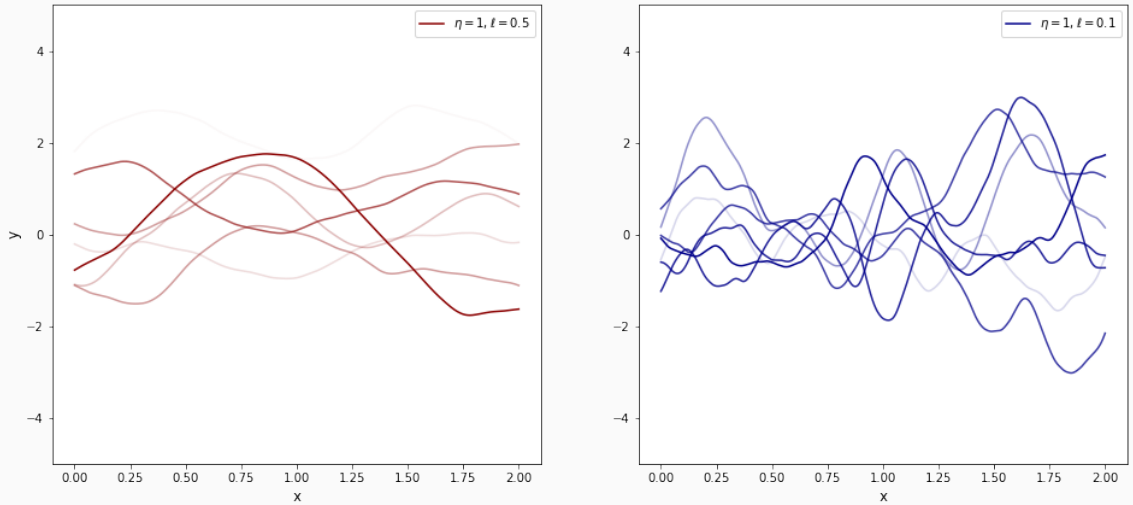
- Matern_ν kernel – $\nu = 1/2, 3/2, 5/2, \ldots$
  - Formula built with gamma and Bessel functions
- $\nu = 1/2$ equivalent to exponential kernel
- $\nu \to \infty$ recovers exponentiated quadratic
- Draws from the prior have $\mathrm{floor}(\nu)$ derivatives

Matern kernels

Samples from the GP prior: Matern(3/2) covariance



Samples from the GP prior: Matern(5/2) covariance



Long term correlations

All of the previous kernels only define "short term" correlations – all go to 0 with distance

- Periodic kernel:

$$k(x, x') = \eta^2 \exp\left(-\frac{\sin^2(\pi|x - x'|/T)}{2\ell^2}\right)$$

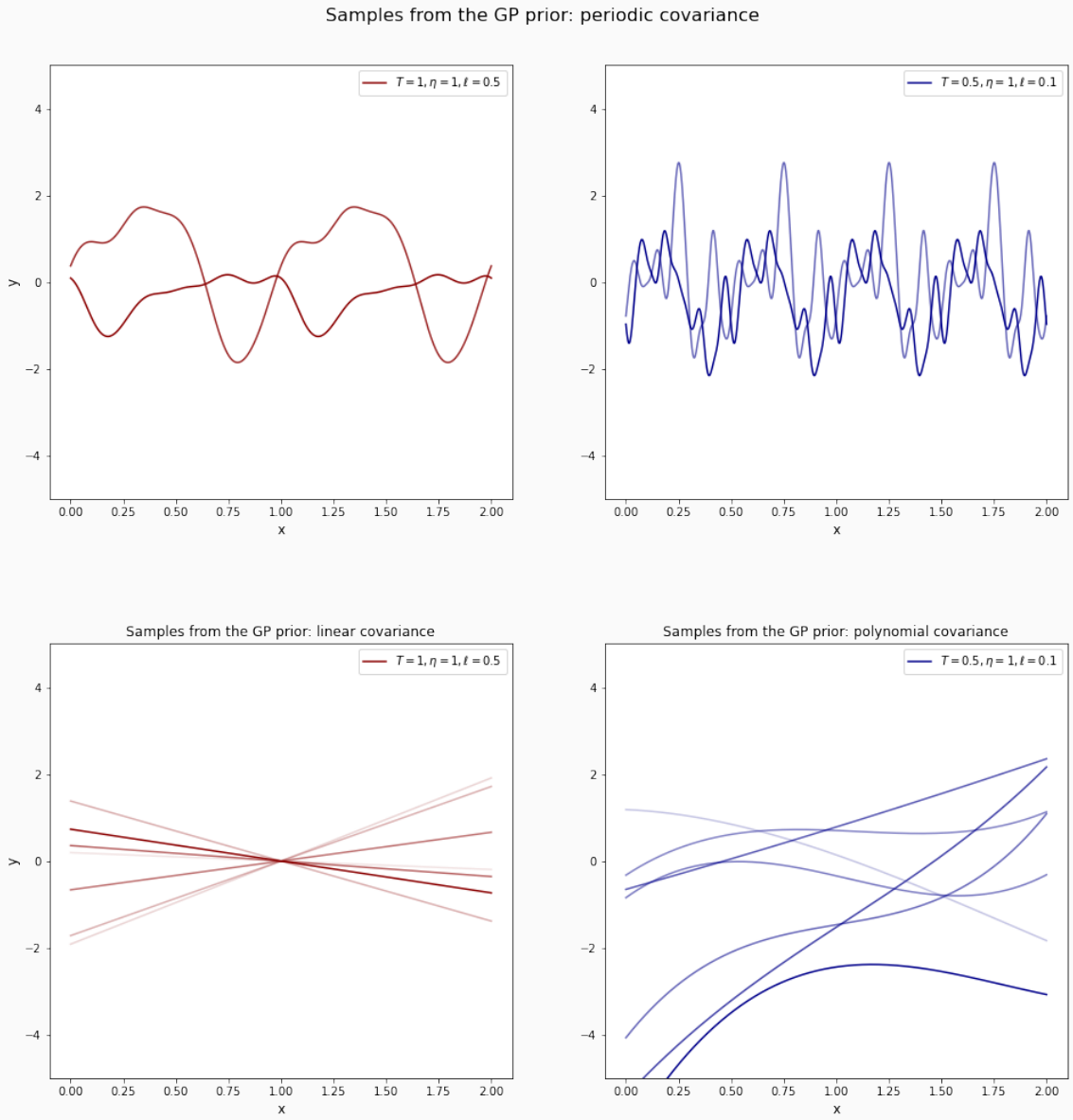
- Linear kernel:

$$k(x, x') = (x - c)(x' - c)$$

- Polynomial kernel:

$$k(x, x') = [(x - c)(x' - c) + offset]^d$$

Periodic


Samples from the GP prior: periodic covariance
$T = 1, \eta = 1, \ell = 0.5$
$T = 0.5, \eta = 1, \ell = 0.1$
Samples from the GP prior: linear covariance
Samples from the GP prior: polynomial covariance
x
y

PyMC3's gp submodule

PyMC3 Gaussian processes

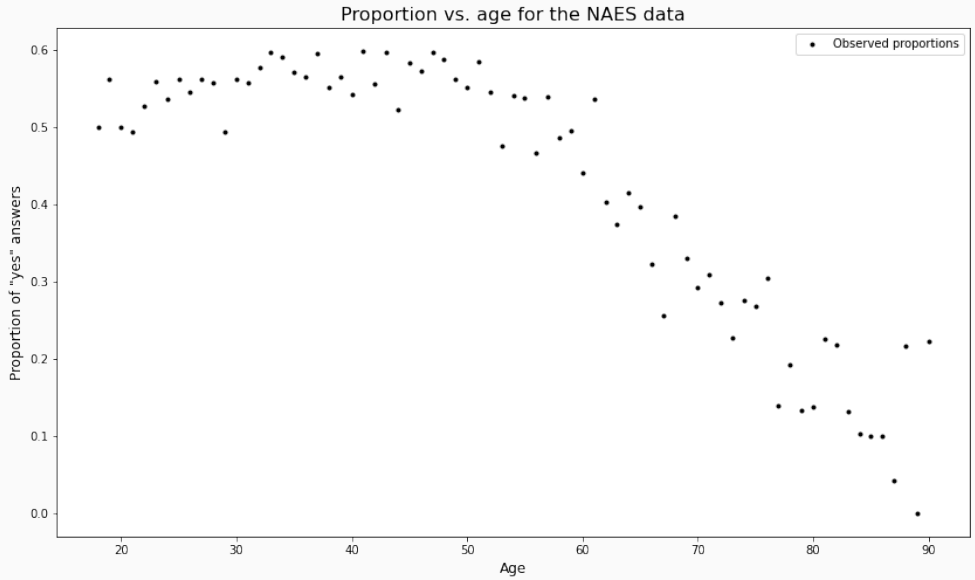
In PyMC3:

- Two implementations of Gaussian processes:
  - `Marginal`: a little simpler to use, but requires that the observed data is GP plus Gaussian noise
  - `Latent`: a little more flexible
- Many standard covariance functions:
  - `Exponential`, `ExpQuad`, `Matern32`, `Matern52`
  - `Periodic`, `Linear`

Example

Exercise 21. from BDA3: data from the 2004 National Annenberg Election Survey

- % of respondents who believe they know someone gay vs. age



Mathematical model

We can write the following model:

$$
\begin{aligned}
y(x) &\sim \text{Normal}(\mu(x), \sigma) \\
\mu(x) &\sim \mathcal{GP}(0, k) \\
k(x, x') &= \text{Matern}_{5/2}(x, x') \\
\eta &\sim \text{Exponential}(1) \\
\ell &\sim \text{Exponential}(0.2) \\
\sigma &\sim \text{HalfCauchy}(1)
\end{aligned}
$$

Notice similarity to a linear regression model!

PyMC3 model

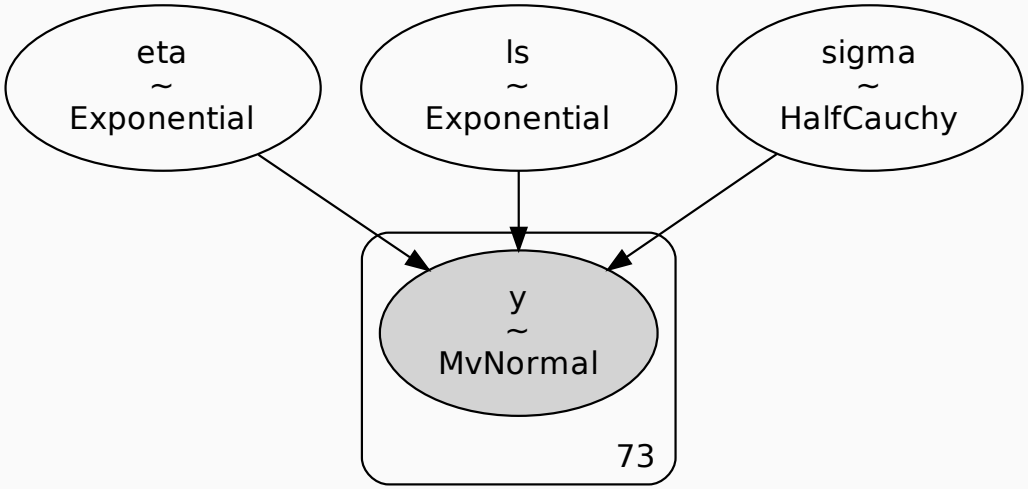
```
eta = pm.Exponential('eta', 1)
ls = pm.Exponential('ls', 0.2)
sigma = pm.HalfCauchy('sigma', 1)

cov = pm.gp.cov.Matern52(1, ls=ls)
gp = pm.gp.Marginal(cov_func=cov) # No variable name!

y_ = gp.marginal_likelihood('y',
        X=age, y=prop, noise=sigma)
# noise can be a covariance function too!
```

Example

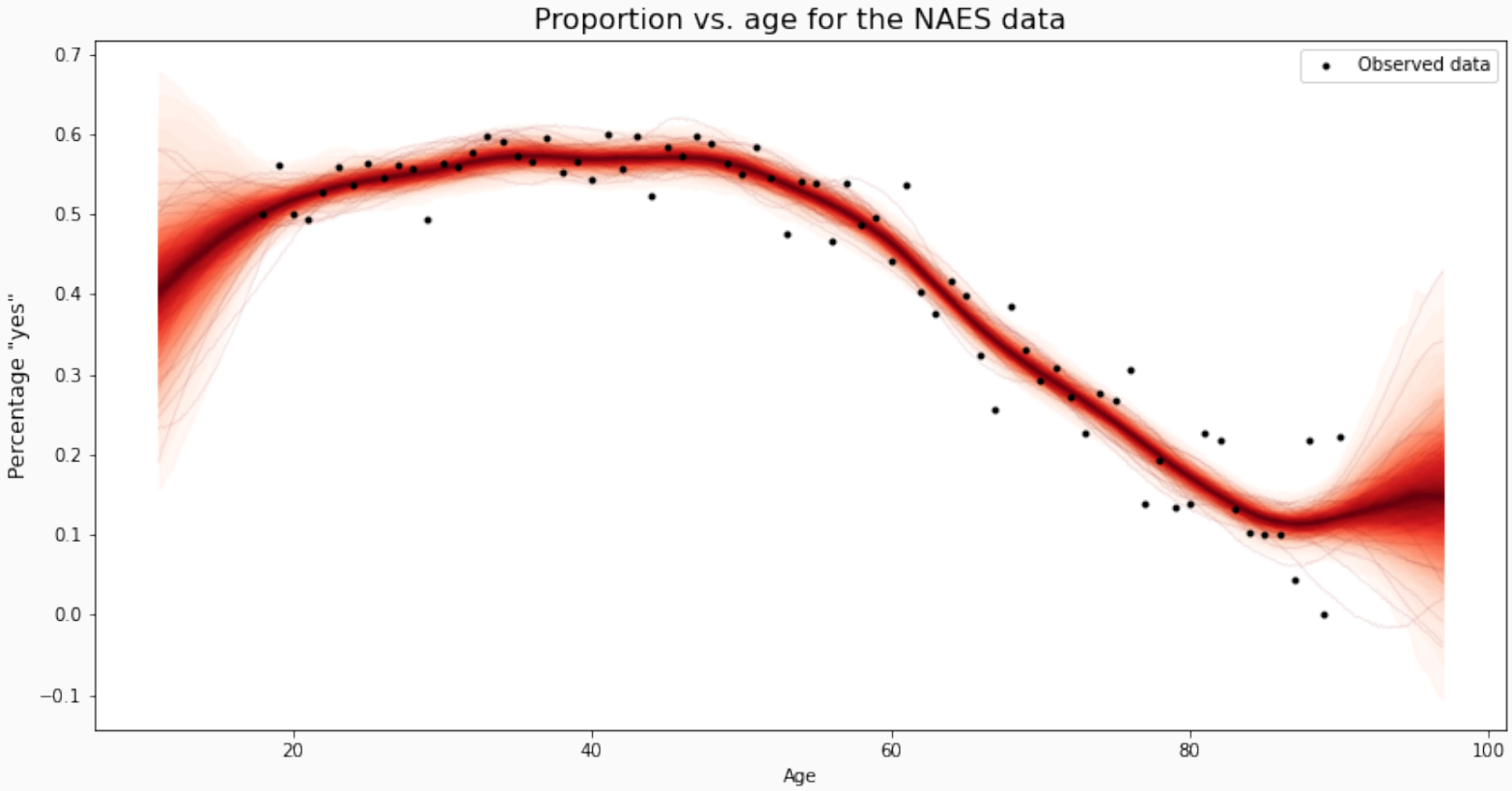
What does the plate diagram look like?



- There's no special Gaussian process variable here – it's just a multivariate normal
- `Marginal` and `marginal_likelihood` are bookkeeping tools

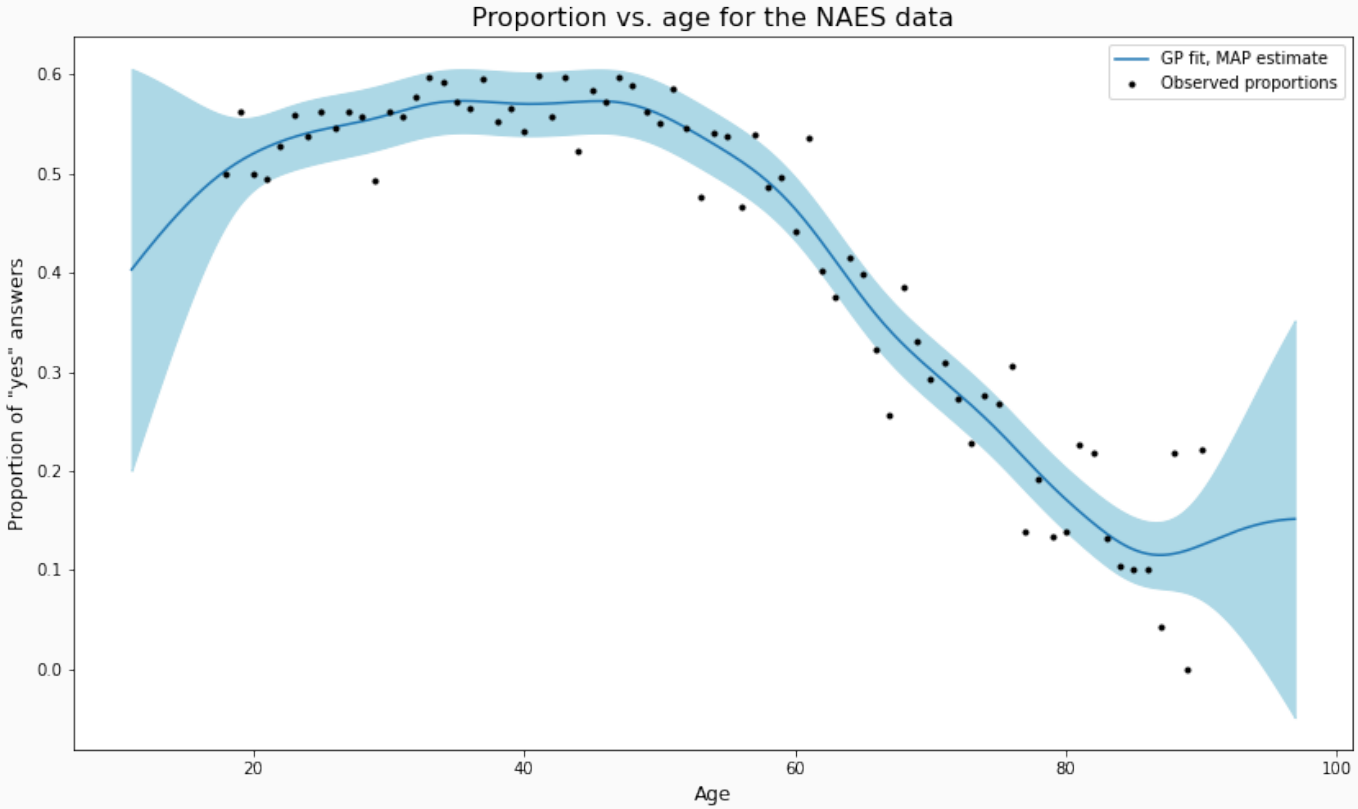
Example

Plotting many samples from the posterior:



Example

Alternative: estimate MAP and use `gp.predict`



Rewriting the bicycle example

- No Gaussian likelihood – Poisson GLM
- Have to use the `Latent` class here
  - Define the GP with a given mean and covariance function
  - Use the `prior` method to create a multivariate normal for a set of x values

Rewriting the bicycle example

Original version:

```
# Priors for parameters
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', 1)
# Gaussian process for time-varying intercepts.
cov_matrix = (eta ** 2) * pm.math.exp(-(weekly_distance ** 2)
                          / (2 * ls ** 2))
                        + 0.01 * np.eye(105)
alpha = pm.MvNormal('alpha', mu=tt.zeros(105),
                    cov=cov_matrix, shape = 105)
# Model equation, likelihood, sampling
theta = pm.math.exp(alpha[bikes['week']] + beta_temp * bikes['temp']
                    + beta_wind * bikes['windspeed'])
y_ = pm.Poisson('y', theta, observed = bikes['cnt'])
trace = pm.sample(init = 'advi')
```

Rewriting the bicycle example

With PyMC3 GP module:

```
# Priors for parameters
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', 1)
# Gaussian process for time-varying intercepts.
cov_func = (eta ** 2) * pm.gp.cov.ExpQuad(1, ls=ls)
         + 0.01 * pm.gp.cov.WhiteNoise(1)
alpha_gp = pm.gp.Latent(cov_func=cov_func)
f = alpha_gp.prior('f', X=bikes['week'][::7, None]
                   , reparameterize=False)
# Model equation, likelihood, sampling
theta = pm.math.exp(f[bikes['week']] + beta_temp * bikes['temp']
                    + beta_wind * bikes['windspeed'])
y_ = pm.Poisson('y', theta, observed = bikes['cnt'])
trace = pm.sample(init = 'advi')
```

Computational issues

Computational difficulties:

- In principle inference is "just" linear algebra if the likelihood is Gaussian
- The linear algebra involves inverting the covariance matrix for the data: $\mathcal{O}(n^3)$
  - Without approximate methods, becomes intractable past a few thousand data points
  - "Cut off" covariance kernels, try to get sparsity in the matrix
  - Compute the covariance matrix only at $m < n$ "inducing points", reduce to $\mathcal{O}(nm^2)$
- Replace MCMC with MAP estimation or approximate inference methods

Additive kernel example

Example: cherry blossom data

Cherry blossoms: big in Japan

- Recorded date of peak flowering
- Records go back to early 10th century CE
- Target: estimate mean peak flowering date over time, separate into slow and fast trends

What might we expect to detect?

Try to detect:

- A long term smooth trend, capturing multi-century trends
- Short-term periodic effects, allowed to decay away from exact periodicity

- Slow trend: exponentiated quadratic kernel
- Faster trend: periodic kernel times Matern(5/2)

Model

$$
\begin{aligned}
y_i &\sim \text{Normal}(\mu(t), \sigma) \\
\mu(t) &= f_1(t) + f_2(t) \\
f_1(t) &\sim \mathcal{GP}(0, k_{slow}) \\
f_2(t) &\sim \mathcal{GP}(0, k_{fast}) \\
k_{slow}(t, t') &= \text{ExpQuad}(t, t') \\
k_{fast}(t, t') &= \text{Matern}_{5/2}(t, t') \times \text{Periodic}(t, t')
\end{aligned}
$$

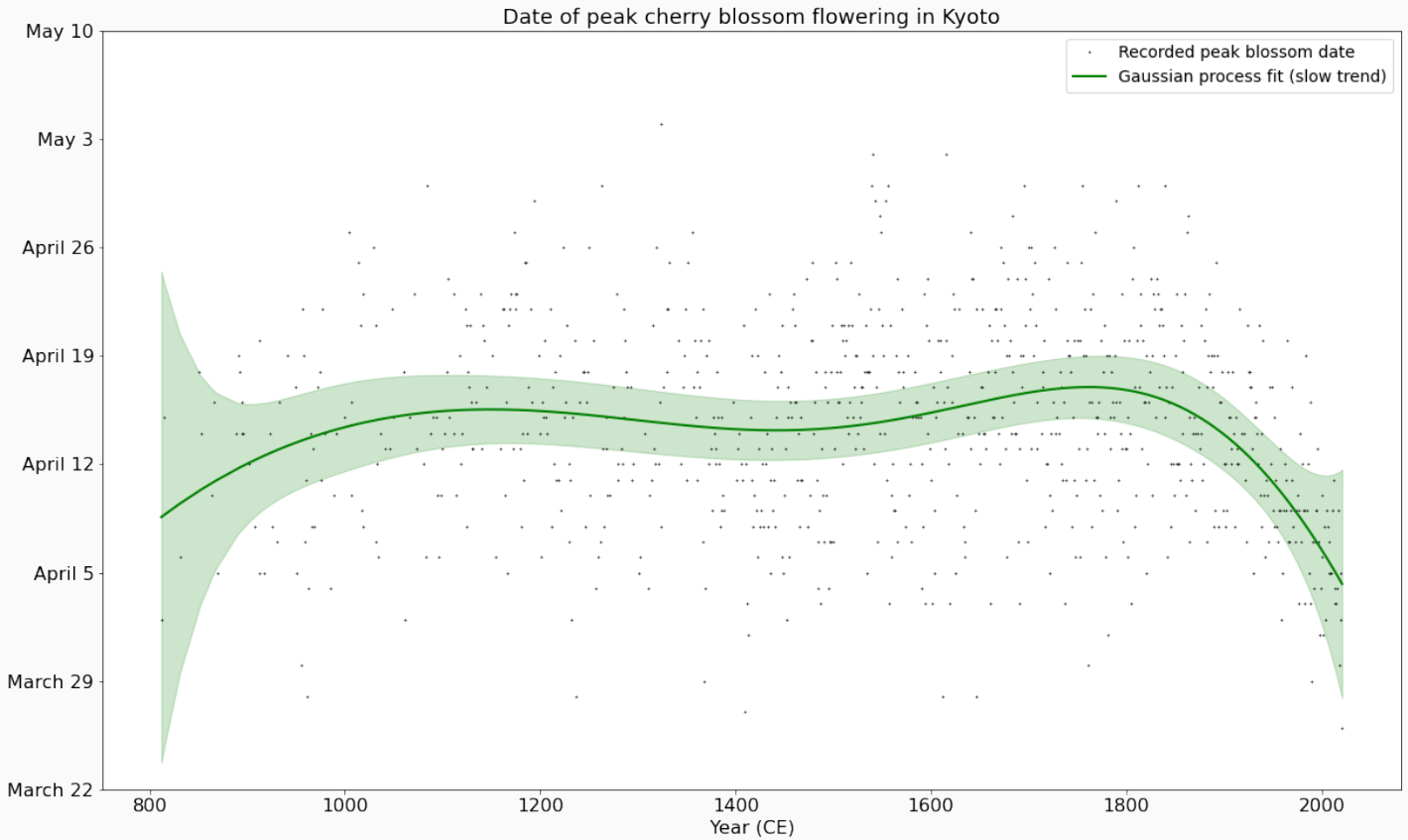
(priors for covariance parameters clipped)

In PyMC3

```
# Periodic component
eta_periodic = pm.Exponential('eta_periodic', 0.5)
l_periodic = pm.Gamma('l_periodic', 5, 1/25)
l_periodic_decay = pm.Gamma('l_periodic_decay', 5, 1/25)
period = pm.Exponential('period', 0.001)
periodic_cov = (eta_periodic ** 2)
             * pm.gp.cov.Periodic(1, ls=l_periodic, period = period)
             * pm.gp.cov.ExpQuad(1, ls=l_periodic_decay)
gp_periodic = pm.gp.Marginal(cov_func=periodic_cov)
# Slow trend
eta_slow = pm.HalfCauchy('eta_slow', 2)
l_slow = pm.Gamma('l_slow', 5, 1/25)
slow_cov = (eta_slow ** 2) * pm.gp.cov.ExpQuad(1, ls=l_slow)
gp_slow = pm.gp.Marginal(cov_func=slow_cov)
#Build model
gp_full = gp_slow + gp_periodic
sigma = pm.HalfCauchy('eta_noise', 5)
y_ = gp_full.marginal_likelihood('y', X=t, y=flower_date, noise=sigma)
```

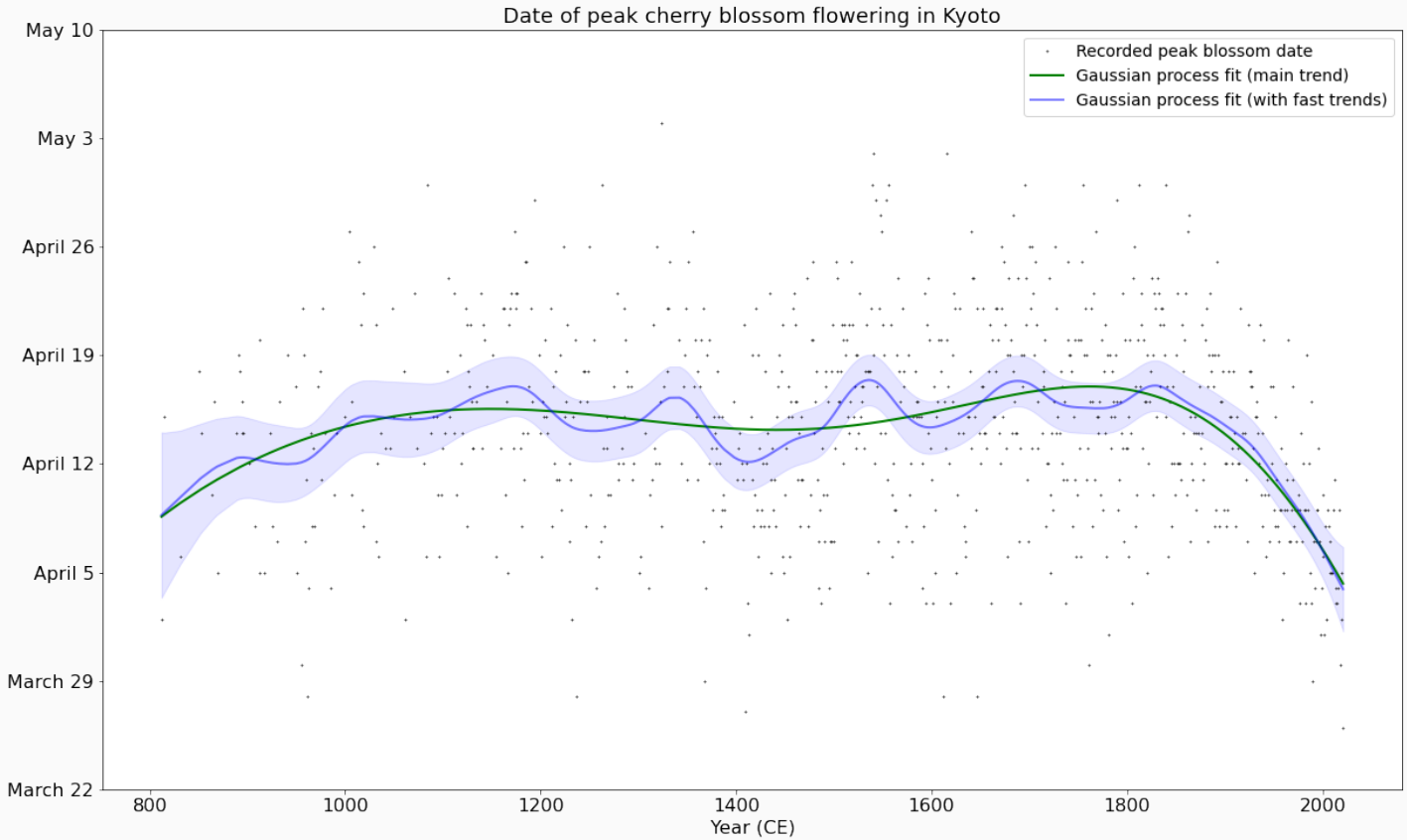
Cherry blossom regression

Plot only the slow trend:



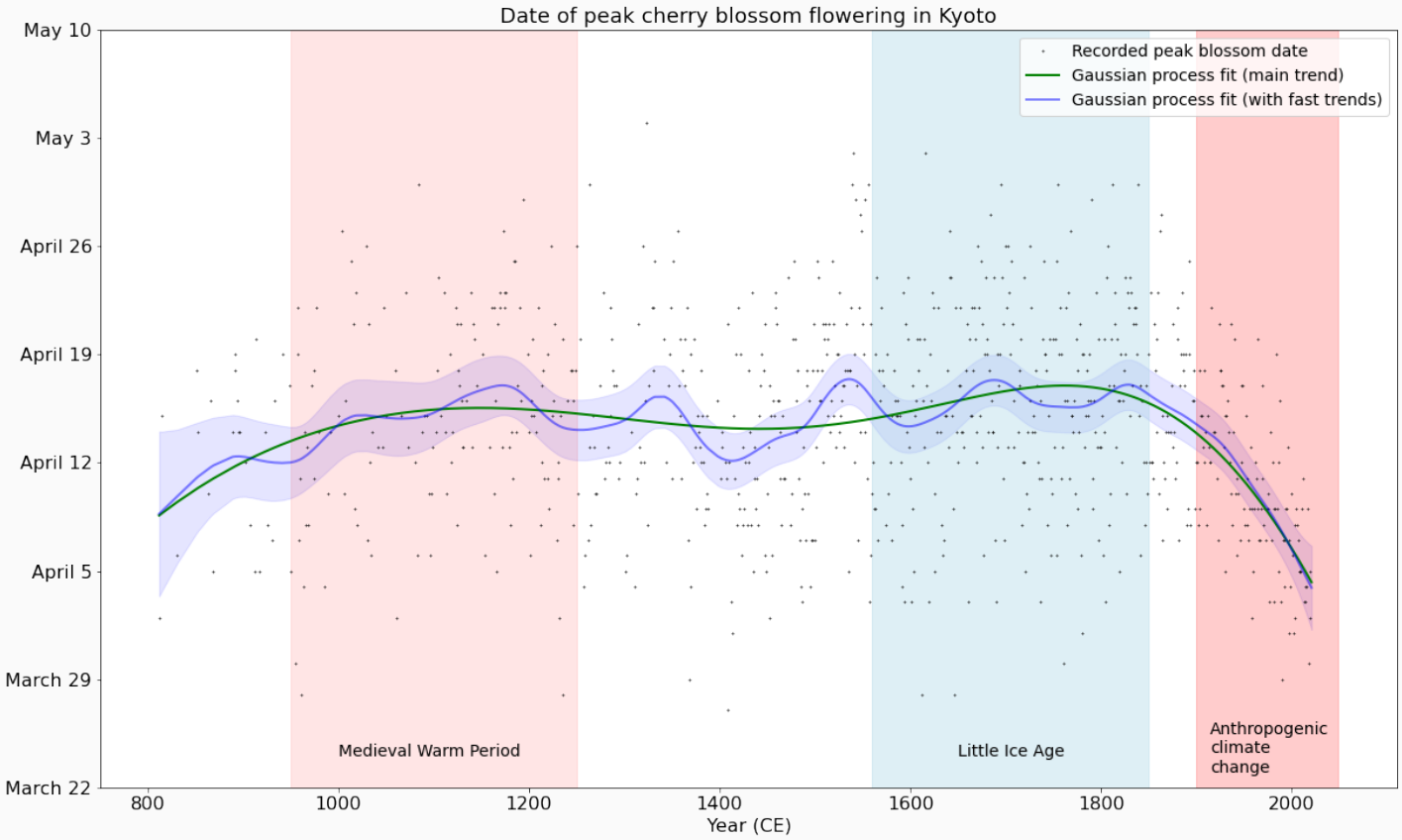
Cherry blossom regression

With the full fit:



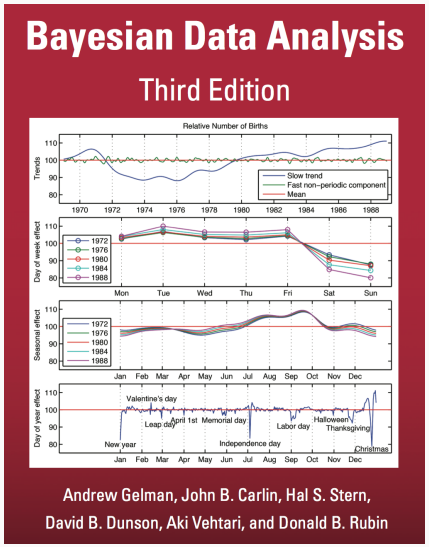
Cherry blossom regression

With the full fit:



Bigger example

For a big, complex example:



- Additive GP with 7 covariance kernels (and 2 non-GP components)
- Includes effects for long-term trends, rapid fluctuations, periodic effects at multiple scales

Summary

Summary:

- GPs allow flexible regression models without explicit function parameterization
- Covariance kernels determine the properties (smoothness, periodicity, etc.) of the functions in the prior
- Additive GPs allow decomposition of functions, modeling different scales of variation separately

Next week:

- Analyzing sequences with hidden state models

Summary

Today:

- Hierarchical linear regression models
  - Varying intercepts
  - Varying slopes

Next week:

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