Missing data and measurement error

ISTA 410 / INFO 510: Bayesian Modeling and Inference

U. of Arizona School of Information

November 22, 2021

Outline

Last week:

- Zero-inflated models
- Parametric functional models

Today:

- Dealing with missing and erroneous data

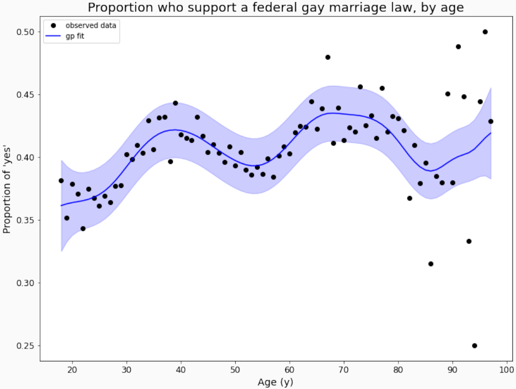
Measurement error

Why we have to think about this

- All measurements have error – some probability of being incorrect, or some difference from the "true" value
- Particularly when data is pooled, aggregated, or averaged, treating observations as exact can lead to too-narrow uncertainties
- If measurement error is nonrandom, can bias estimates

Underestimating error

Previously, we fit a Gaussian process to this data:



But there is a problem.

How our model accounted for error

Our GP model:

$$
\begin{aligned}
y &\sim \mathrm{Normal}(\mu_i, \sigma) \\
\mu_i &= f(x_i) \\
\sigma &\sim \mathrm{Exponential}(1) \\
f &\sim \mathcal{GP}(0, k) \\
k &= \eta^2 \mathrm{ExpQuad}(\ell^2) \\
\eta^2, \ell^2 &\sim \mathrm{Exponential}(1)
\end{aligned}
$$

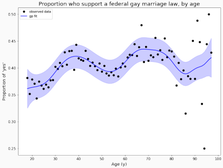
Every other linear regression:

$$
\begin{aligned}
y &\sim \mathrm{Normal}(\mu_i, \sigma) \\
\mu_i &= \boldsymbol{\beta X} \\
\sigma &\sim \mathrm{Exponential}(1) \\
\beta_j &\sim \mathrm{Normal}(0, 1)
\end{aligned}
$$

Heteroskedasticity

In a world where all errors are normally distributed and have the same variance, this is fine!

- We do not live in such a world
- Even the most mundane of random errors can easily fail this:



- The problem here: *heteroskedasticity* (errors do not have constant variance)
- In this case the reason is easily understood: variable sample size

A return to DAGs

Reimplementing with Latent GP

To explicitly separate out the underlying random variable p_{True} from the observed p, we'll re-implement with the Latent class:

```
with pm.Model() as naes_model:
    eta = pm.Exponential('eta', 1)
    ls = pm.Exponential('ls', 1/5)
    sigma = pm.HalfCauchy('sigma', 1)

    cov_func = (eta ** 2) * pm.gp.cov.ExpQuad(1, ls=ls)
    gp_fit = pm.gp.Marginal(cov_func = cov_func)

    y_ = gp_fit.marginal_likelihood('y', X = naes.age.values[:, None],
                                     y=naes.prop_yes.values, noise = sigma)
```

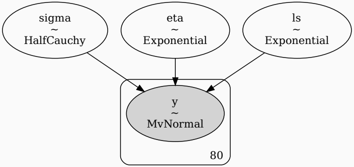
Reimplementing with `Latent` GP

To explicitly separate out the underlying random variable p_{True} from the observed p, we'll re-implement with the `Latent` class:

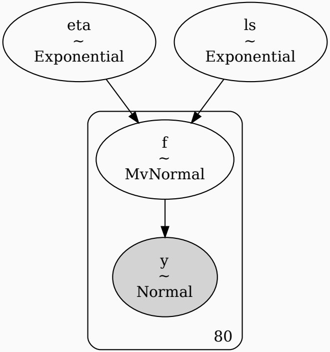
```
pSE = np.sqrt(naes.prop_yes * (1 - naes.prop_yes) / naes.n)

with pm.Model() as naes_model:
    eta = pm.Exponential('eta', 1)
    ls = pm.Exponential('ls', 1/5)

    cov_func = (eta ** 2) * pm.gp.cov.ExpQuad(1, ls=ls)
    gp_fit = pm.gp.Latent(cov_func = cov_func)
    f = gp_fit.prior('f', X=naes.age.values[:, None], reparameterize=False)

    y_ = pm.Normal('y', mu=f, sigma=pSE, observed = naes.prop_yes)
```

Model diagrams

Original model:

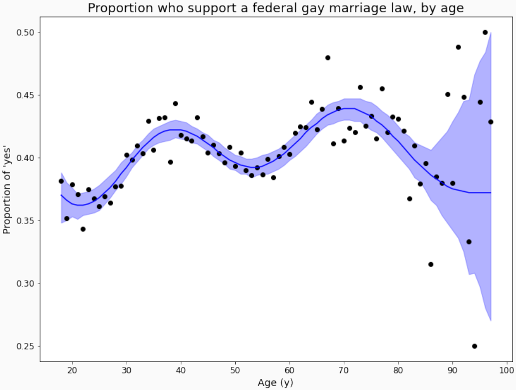


Including measurement error:



Result

The result has much more uncertainty about the underlying function at the high age ranges:



Error on predictors

What if the errors are on the predictors?

- Sometimes predictors are measured with very high accuracy, but sometimes not
- We can probably trust most observations of age in the NAES data set
- Another dataset from Rethinking, `elephants.csv`, also has age as a predictor
    - Age of bull elephants predicting count of mating events
    - Problem: age only estimated, not always accurately

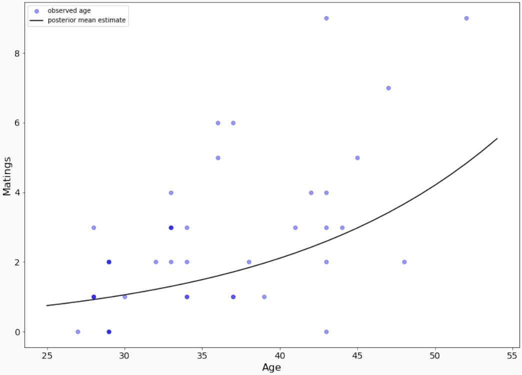
Error on predictors

Start with a simple Poisson model:

```
with pm.Model() as simple_model:
    alpha = pm.Normal('alpha', 0, 10)
    beta = pm.Normal('beta', 0, 1)

    rate = pm.math.exp((alpha + beta * (elephants['AGE']
        - elephants['AGE'].median())))

    y_ = pm.Poisson('y', rate, observed = elephants['MATINGS'])
```

Error on predictors

Result of the model fit:



Error on predictors

- Now, it may be reasonable to assume that the observed mating counts are accurate, assuming consistent observation
- However, we cannot just ask elephants their age
- Estimates are uncertain – suppose they have a standard deviation of 5 years
  - Add parameters to the model for the true ages
  - Treat the observed ages as a normal random variable centered at the true ages

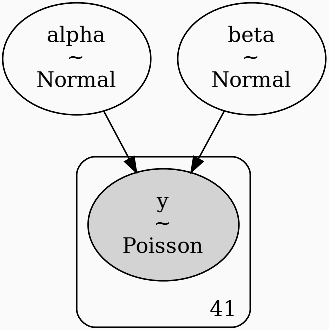
Error on predictors

```
with pm.Model() as error_model:
    alpha = pm.Normal('alpha', 0, 10)
    beta = pm.Normal('beta', 0, 1)

    age_true = pm.Normal('age_true', 40, 8, shape = len(elephants))
    rate = pm.math.exp(alpha + beta * (age_true - 40))

    age_obs = pm.Normal('age_obs', mu=age_true,
        sigma = 5, observed = elephants['AGE'])

    y_ = pm.Poisson('y', rate, observed = elephants['MATINGS'])

    error_trace = pm.sample(2000, target_accept = 0.9)
```

- Something new here: two observed variables
- No reason this isn't allowed – PyMC3 just links up the computational graph and computes the log posterior

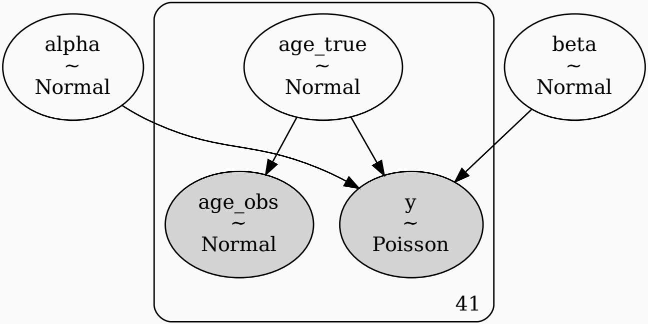
Diagramming the model

Original model:



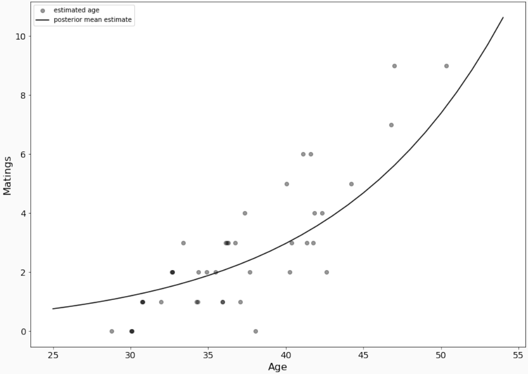
Diagramming the model

New model:



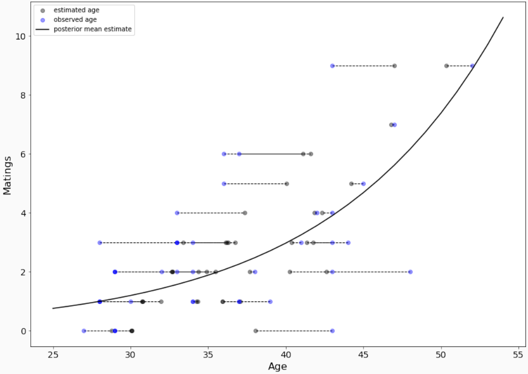
Error on predictors

Result of the model fit:



Error on predictors

Compare to original data:



Shrinkage with measurement error

This is another instance of shrinkage:

- Shrinkage in multilevel models:
  - Extreme observations “shrunk” toward overall average
  - Model is skeptical of clusters that don’t fit the general pattern
- Now:
  - Extreme observations “shrunk” toward estimated trend
  - Model is skeptical of age estimates that don’t agree with predicted number of matings
- Can do this on both predictor and outcome – see ch. 15

Missing data

Missing values

Common in real-world data: some rows in the data set are missing values for some of the variables

- Option 1: drop rows with NAs (complete case analysis)
  - At best, this is inefficient because we are getting rid of data
  - At worst, if missingness is associated with some of our variables, this can introduce biases
- Option 2: Replace NAs with a fixed value (e.g. mean or mode of the nonmissing values, or 0)
  - This is wrong and bad
  - Don't

Missing values are measurement errors

The third option: *impute* the missing values

- Missingness is a measurement error – just a specific type of measurement error
- So, we can go back to the DAG, explicitly include missingness in the model
- A missing value is now just an unknown parameter
- Bayesian imputation: we fill in the missing gap, but not with a fixed value (like mean(x)); instead, with a probability distribution

Back to the DAG

The DAG for missing value imputation is especially important:

- Missingness generally has a cause
- If this cause is related to our predictors or outcome
- Put the missingness mechanism into the DAG, see what we can learn

Three forms of missingness

As we'll see, there are three major categories of missingness

- can be distinguished by the structural causal relationship between missingness and other variables
- some types allow us to impute and proceed with estimation
- some types will hopelessly confound estimates

Three unwieldy terms

- Missing completely at random (MCAR): missingness not related to predictors or outcomes
- Missing at random (MAR): missingness related to predictors, but not outcomes
- Missing not at random (MNAR): missingness related to outcomes

Easier to understand if we draw a DAG

The dog and the homework

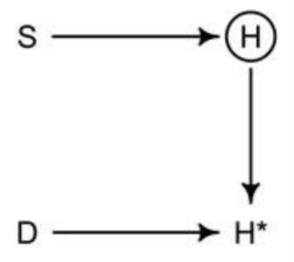
The book has a nice metaphor:

- Students, all of whom have dogs, study a varying amount (S) and produce homework of varying quality (H)
- Predictably, S and H are positively associated – students who study more produce better work
- After the homework is complete, some of the students' dogs eat their homework (D)

Our ability to estimate the causal effect of S on H will depend on the nature of the relationship between D and other variables

Missing completely at random

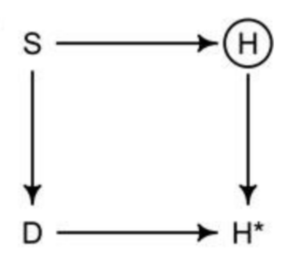
The first category: missing completely at random



- D is independent of the other variables
- Some dogs are good dogs, some are... less good dogs
- Good news: this still lets us identify causal effects, because it does not in principle affect the joint distribution of S, H – just reduces effective sample size

Missing at random

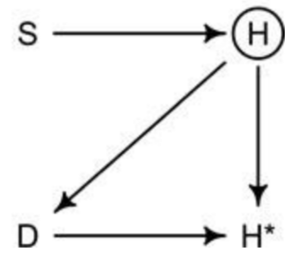
Second category: missing at random



- D is related to the predictors
- Students who study more spend less time with their dogs. In frustration, the dogs retaliate
- Good news: although this opens a backdoor path, it can be blocked by conditioning on S (which we were going to do anyway)

Missing not at random

Third category: missing not at random



- *D* is related to the *outcomes*
- Dogs eat specifically bad homework (or good). (Maybe the students, recognizing the homework is bad, feed it to the dog so they don't have to turn it in)
- Now we have a backdoor path that cannot be blocked.

Summary

Today:

- Measurement errors are ubiquitous
- Measurement errors can be explicitly included in a model
- Missing data can be treated as a specific type of error, and may or may not affect causal inference

Next time:

- Imputing missing values