Intro to Bayesian Statistics with PyMC3

By Taylor Smith and Jonathan Hayne

Overview

- Introduction to Bayesian Statistics
- Coding Example with PyMC3
- Goals
    - To give a very brief overview of foundational concepts of Bayesian statistics in order for one to be prepared to better handle more advanced concepts later
    - Present a classic example as motivation for the Bayesian framework
    - Demonstrate a key platform used for Bayesian statistics

Philosophical Introduction

- Frequentist
  - Defines probability as the limit of the relative frequency in a large number of trials.
  - Based on the idea of infinitely repeatable data generating process
  - View parameters as fixed and the data as random
- Bayesian
  - Probability is defined as a subjective degree of belief concerning whether the event will occur.
  - An expression of uncertainty surrounding an event
  - Data is viewed as fixed with the parameters randomly distributed

Philosophical Introduction

- $P(\theta|X) \propto L(\theta|X)P(\theta)$
  - Prior - represents our prior beliefs about θ before seeing the data
  - Likelihood - represents what the data has to say about the possible values of θ.
  - Posterior - reflects our beliefs about θ after having viewed the data
- The posterior - always a compromises between the prior and likelihood
  - The more data you have, the more influence the likelihood will have on the posterior
  - The less uncertainty expressed in the prior, the more data that will be needed to strongly influence the posterior.

Basic Steps of a Bayesian Model

- Every Bayesian model will involve:
  - The specification of a probability model incorporating whatever prior knowledge is available.
  - Update by conditioning on the data.
  - Evaluating the fit of the model and whether the assumptions are properly met.
- Very flexible framework
- Assumptions are very clear and explicit

Inference

- The posterior distribution is often intractable (cannot be analytically solved) for all but very simple models.
- This held Bayesian statistics back for a long time.
- With advance of computational power, Bayesian statistics has massively grown in popularity.
- Two common methods for analyzing the posterior:
    - Markov Chain Monte Carlo (MCMC)
    - Variational Inference

Markov Chain Monte Carlo

- Method for sampling from a distribution using Markov Chains that have the desired distribution as its equilibrium distribution.
- Effectively, sample from distributions you know how to sample in such a way that it comes to represent the distribution you are trying to sample.
- Guaranteed to converge eventually
- Before convergence, known as burn-in. Not good estimate of posterior.
- But will take a long time.
- Many different kinds.
    - Gibbs Sampling
    - Metropolis-Hastings
    - Hamiltonian Monte Carlo

Variational Inference

- “Variational Inference is that thing you implement while waiting for your Gibbs sampler to converge” - David Blei, allegedly
- Approximates an intractable distribution with a tractable one
- Generally speaking, reduces Kullback-Leibler Divergence between the approximate distribution and the true posterior.
- Can be much faster than MCMC methods.
- Will always be biased.

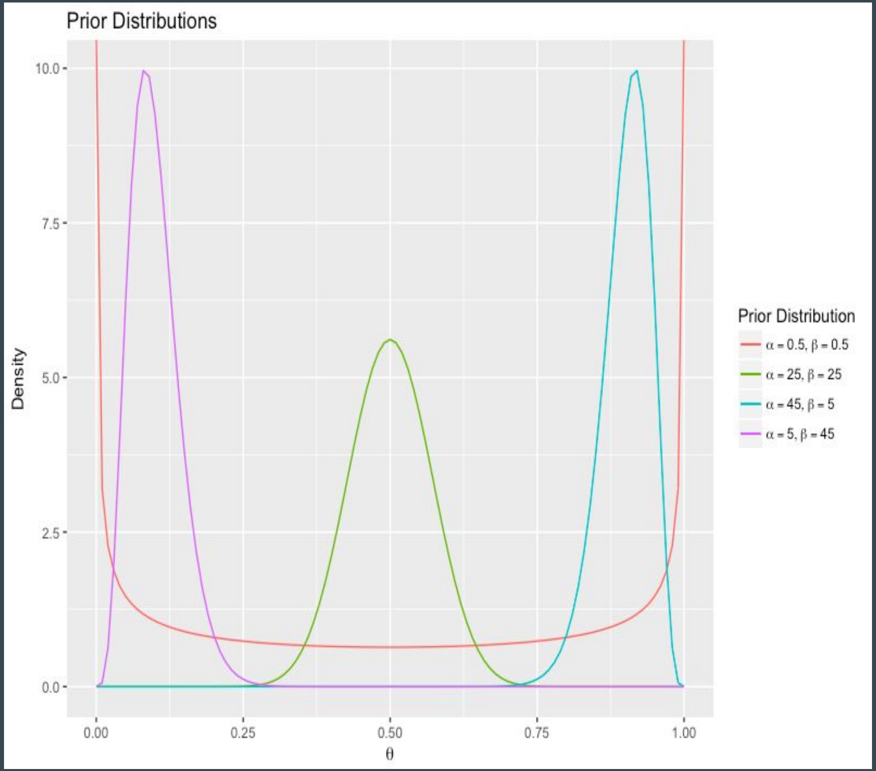
Priors

- Brilliant way to inject prior information into analyses or evil that destroys objectivity?
- It’s definitely the first one...
- If you have meaningful prior information, why would you not want to incorporate it?
- If you don’t have meaningful prior information, then you use a noninformative or weakly-informative prior.
- Most important thing, priors should be a reflection of your degree of certainty. Don’t want to overly restrict values but also don’t want to assign prior probability to ridiculous values.

How to pick your priors

- Conjugate priors are often a natural choice
  - Conjugacy - if the posterior distribution and the prior distribution are in the same family of distributions.
  - Normal prior is conjugate to a normal likelihood
  - Beta distribution is conjugate to a binomial likelihood (probability)
  - Gamma distribution is conjugate to a poisson likelihood (counts)
  - For example, suppose you put a prior of Beta(5, 20) on some probability and then you observe in your data that in 9 of 15 trial this event happened, the posterior would be Beta(14, 26)
- Prior elicitation - ways of converting expert knowledge into quantitative distributions
- Sometimes you just have expert knowledge
  - Normal(93, 2.5) would be good prior for fastball velocity in MLB
- Cross-validation for hyperparameters

Illustration of choice of priors



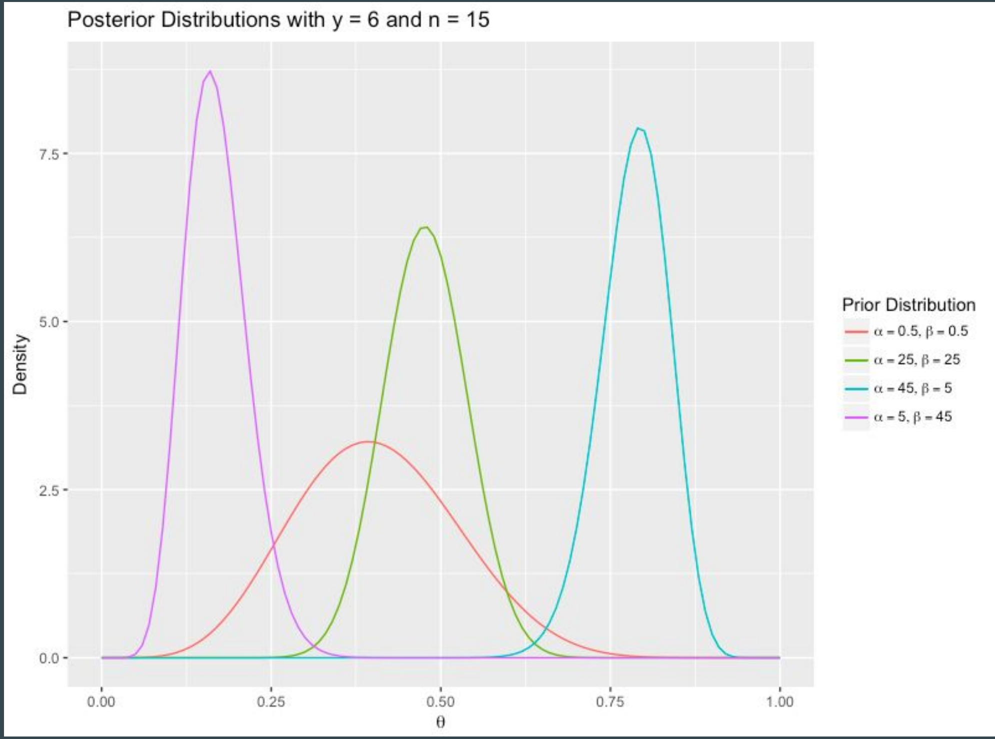
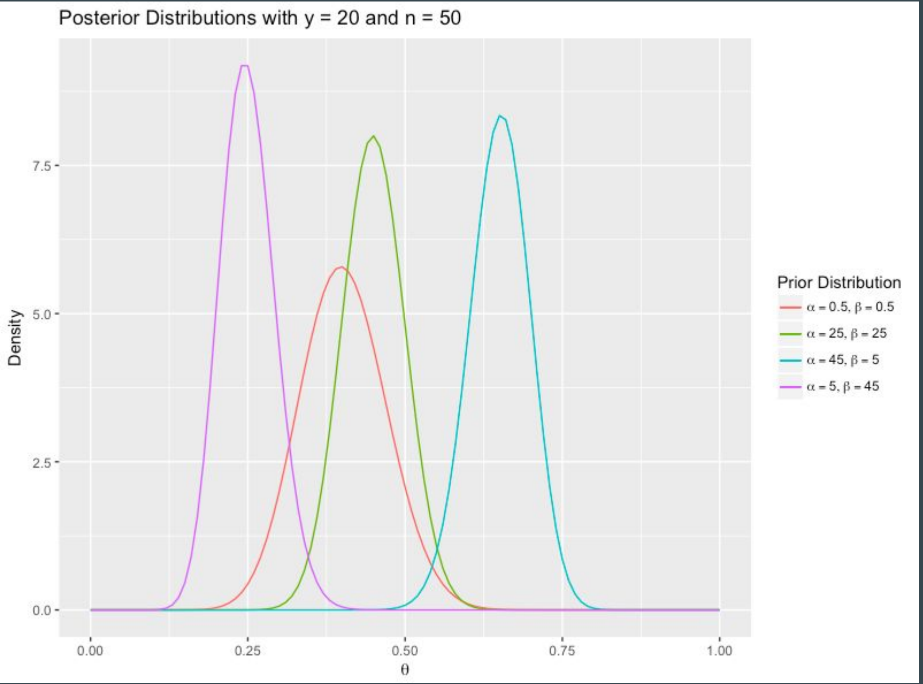
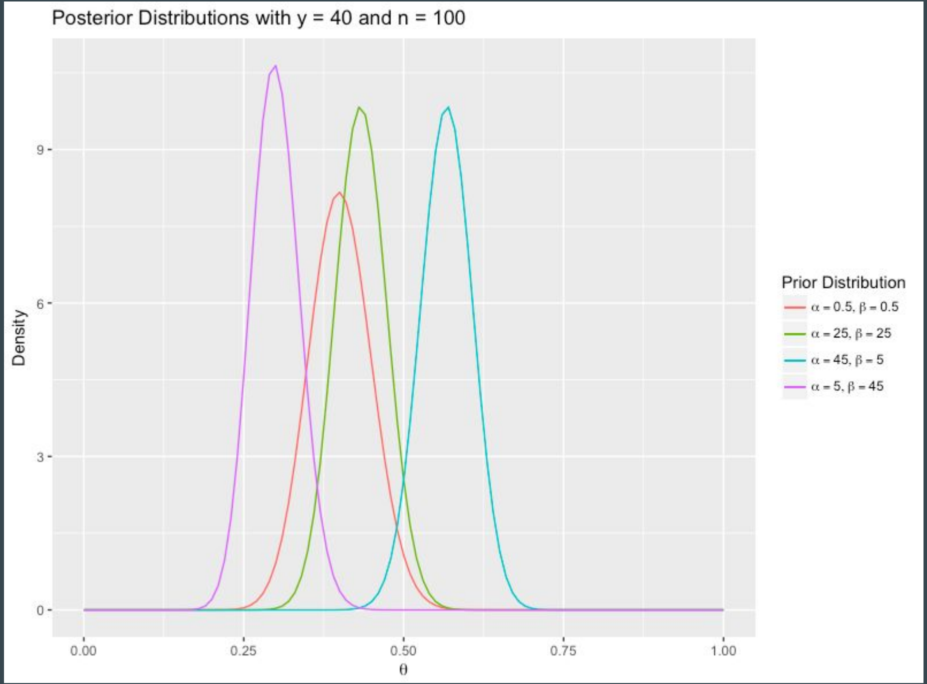


Illustration of choice of priors





Hierarchical/Multilevel Models

- Probably the greatest strength of the Bayesian framework
- Useful when you have multiple levels within your data: counties in a state, documents in a corpus, players on a team, students in a class
- When there are differences between the individuals levels but they are all apart of the same population and thus share a structure.
- Hierarchical models allow us to exploit this structure by sharing statistical strength.
- For example, the faster a pitch is the harder it is for a batter to hit. This will effect some batters more than others but in general how it affects the whole tells us about how it affects the individual.

Hierarchical/Multilevel Models

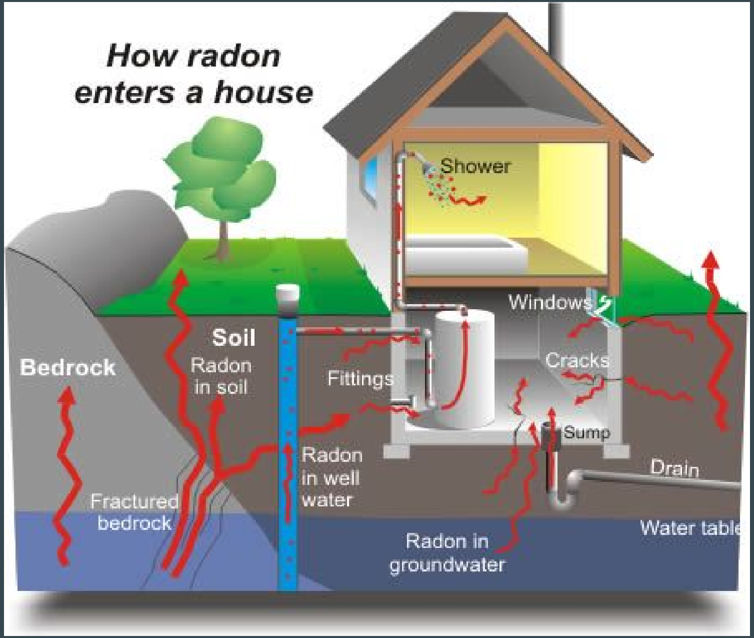
- Can be extended to a very wide variety of models (Hierarchical Latent Dirichlet, Hierarchical Neural Networks, Hierarchical Dirichlet Process Hidden Semi-markov Model)
- We will keep it simple here and demonstrate a hierarchical linear model in PyMC3

PyMC3

- “A Python package for Bayesian statistical modeling and Probabilistic Machine Learning which focuses on advanced Mrkov Chain Monte Carlo and variational fitting techniques.
- Very intuitive model specification and easy to use

Coding Example

- Andrew Gelman's classic hierarchical dataset: radon levels within houses in Minnesota
- Target variable: the log of the radon level
- Explanatory variables. Whether the measurement was taken in the basement or the first floor.



Models

- We fit three separate models.
  - One where we pool the counties and fit a single regression.
  - One where we fit a separate regression for each county
  - A hierarchical model where we have county-specific slopes and intercepts that are realizations of a shared distribution over possible county slopes and intercepts
- Notationally, we can represent the models as such:
  - $\text{radon}_{i,c}=\beta_0+\text{floor}_{i,c}*\beta_1+\epsilon$
  - $\text{radon}_{i,c}=\beta_{0,c}+\text{floor}_{i,c}*\beta_{1,c}+\epsilon$
  - $\beta_{0,c}\sim N(\mu_0,\sigma_0^{\,2}),\ \beta_{1,c}\sim N(\mu_0,\sigma_0^{\,2}),$

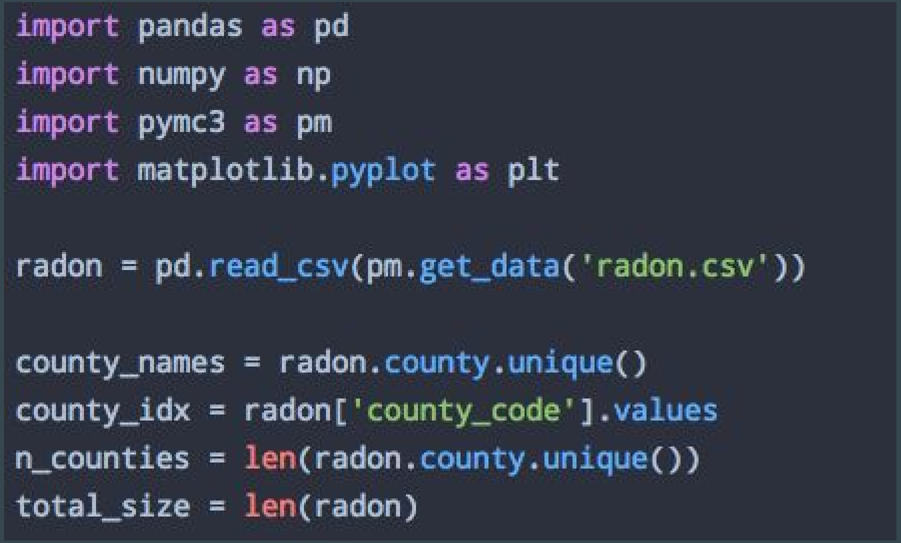
$$\text{radon}_{i,c}=\beta_{0,c}+\text{floor}_{i,c}*\beta_{1,c}+\epsilon$$

Models

- Spoilers: The last model is the best
  - We believe there are differences between counties which the first model ignores
  - However, we believe that overall counties are similar and share a structure so ignoring all other points outside of a county is a poor practice.
  - The last is a compromise between the two, acknowledges differences between counties but also recognizes that overall they should behave similarly
- These models represent complete pooling, no pooling, and partial pooling respectively.

Setup

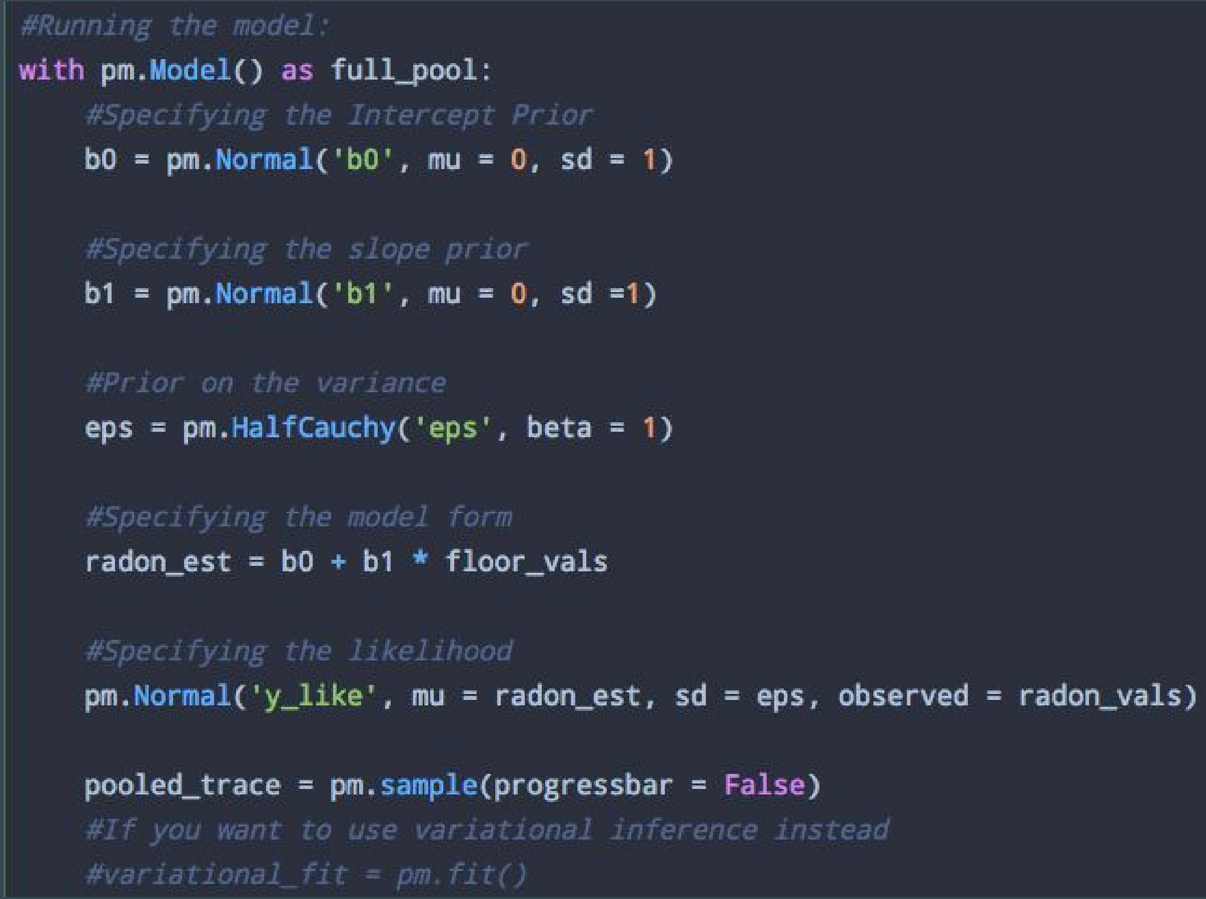
- Nothing to see here.
- Importing libraries and reading data.



```
import pandas as pd
import numpy as np
import pymc3 as pm
import matplotlib.pyplot as plt

radon = pd.read_csv(pm.get_data('radon.csv'))

county_names = radon.county.unique()
county_idx = radon['county_code'].values
n_counties = len(radon.county.unique())
total_size = len(radon)
```

Pooled Model

- Specify priors for b0, b1, sigma
- Specify the structure for the estimate of the log of radon
- Specify the likelihood
- Fit the model



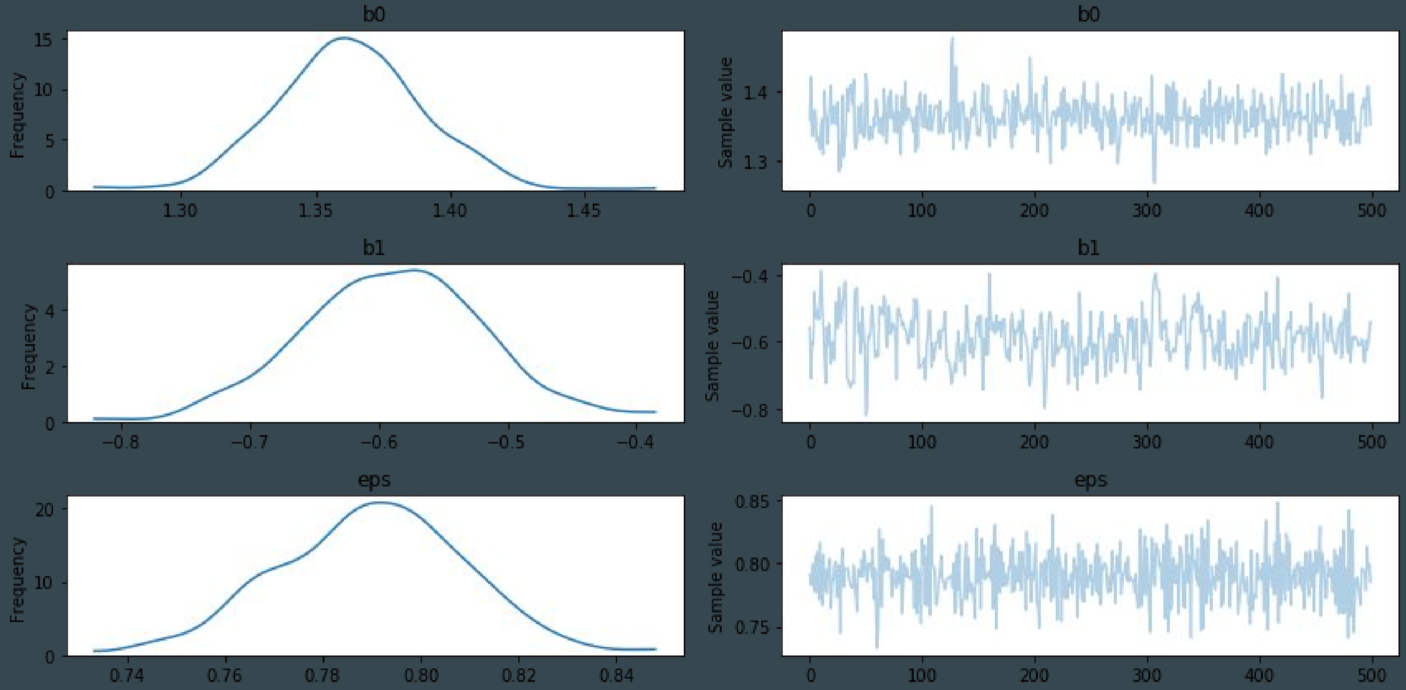
```
#Running the model:
with pm.Model() as full_pool:
    #Specifying the Intercept Prior
    b0 = pm.Normal('b0', mu = 0, sd = 1)

    #Specifying the slope prior
    b1 = pm.Normal('b1', mu = 0, sd =1)

    #Prior on the variance
    eps = pm.HalfCauchy('eps', beta = 1)

    #Specifying the model form
    radon_est = b0 + b1 * floor_vals

    #Specifying the likelihood
    pm.Normal('y_like', mu = radon_est, sd = eps, observed = radon_vals)

    pooled_trace = pm.sample(progressbar = False)
    #If you want to use variational inference instead
    #variational_fit = pm.fit()
```

Traceplots

- Need to make sure that our Markov Chains have converged (ran enough iterations that they are sampling from the true posterior)
- Not an exact science but trace plots are a good place to start.
- They follow the train of values picked from the MCMC
- Want your trace plots to look like white noise.
- If they don’t you may need to run more iterations or change the structure of the model.

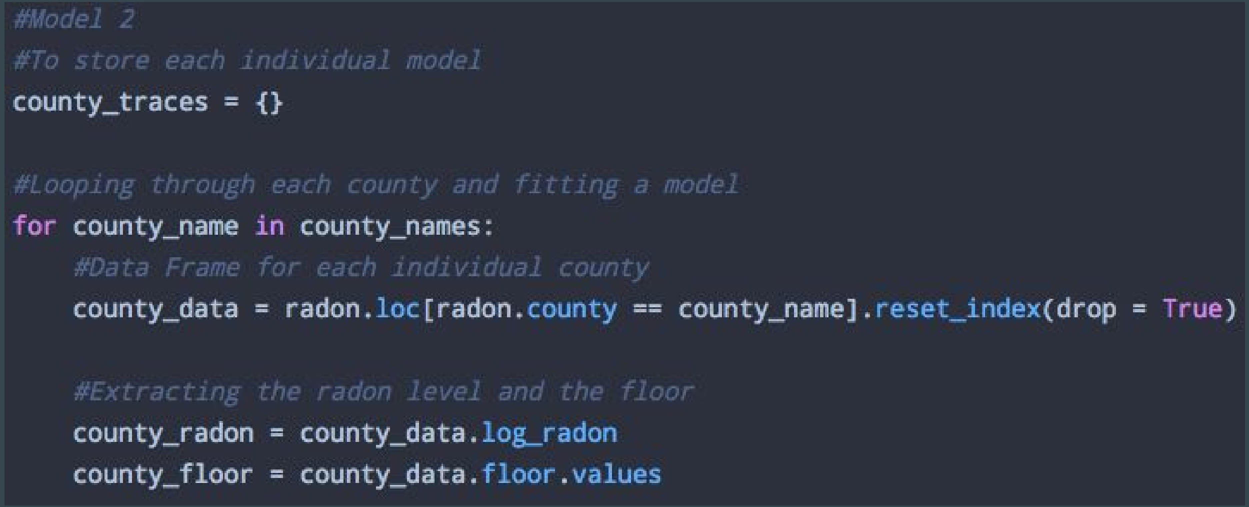
Traceplots and Posterior Distributions for Parameters

- Traceplots look okay.
- We can see our sampled posterior distribution for each of our parameters.



No Pooling

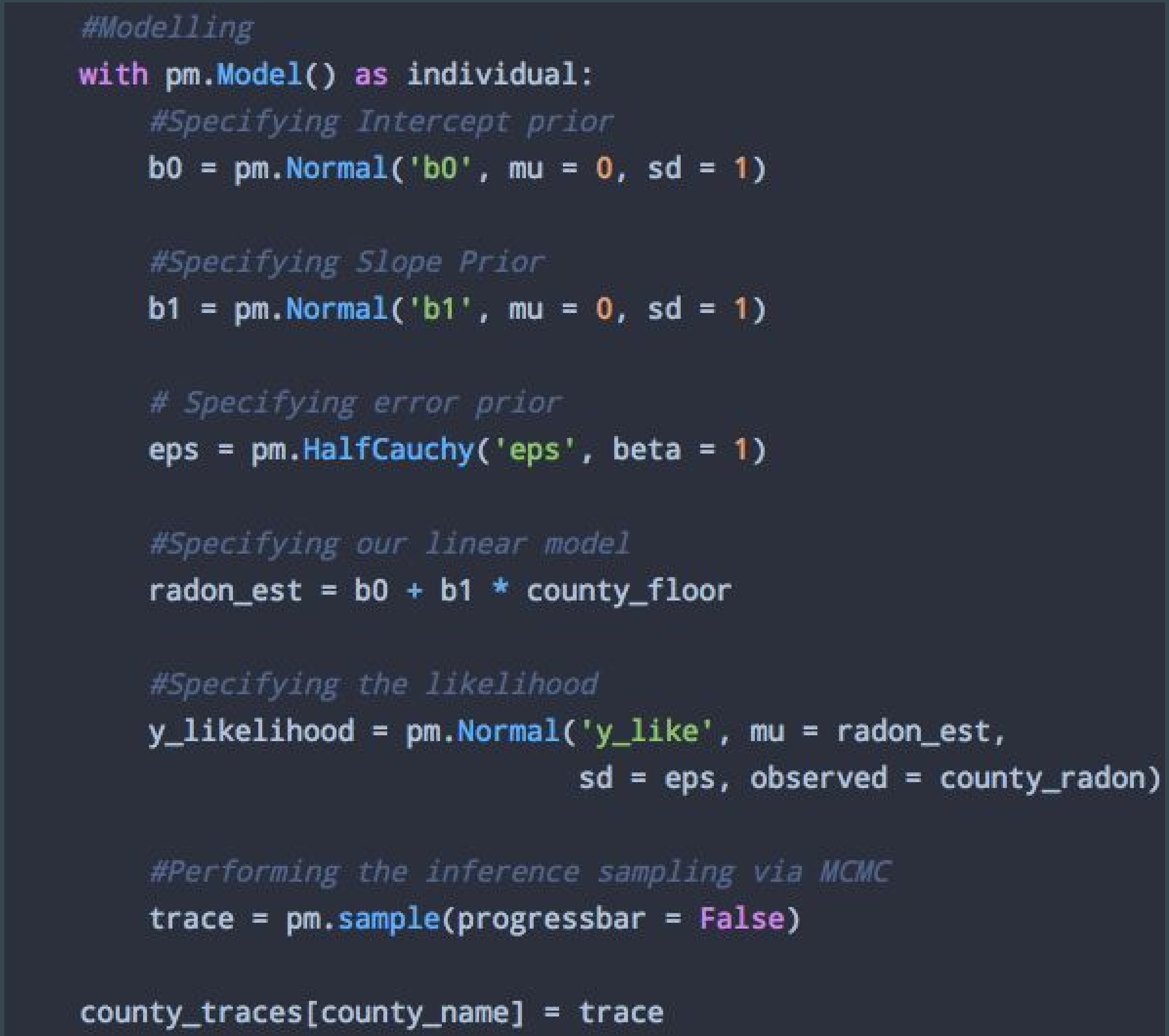
- Setting up a loop to fit a separate regression for each county ignoring all other points.



```
#Model 2
#To store each individual model
county_traces = {}

#Looping through each county and fitting a model
for county_name in county_names:
    #Data Frame for each individual county
    county_data = radon.loc[radon.county == county_name].reset_index(drop = True)

    #Extracting the radon level and the floor
    county_radon = county_data.log_radon
    county_floor = county_data.floor.values
```

No Pooling

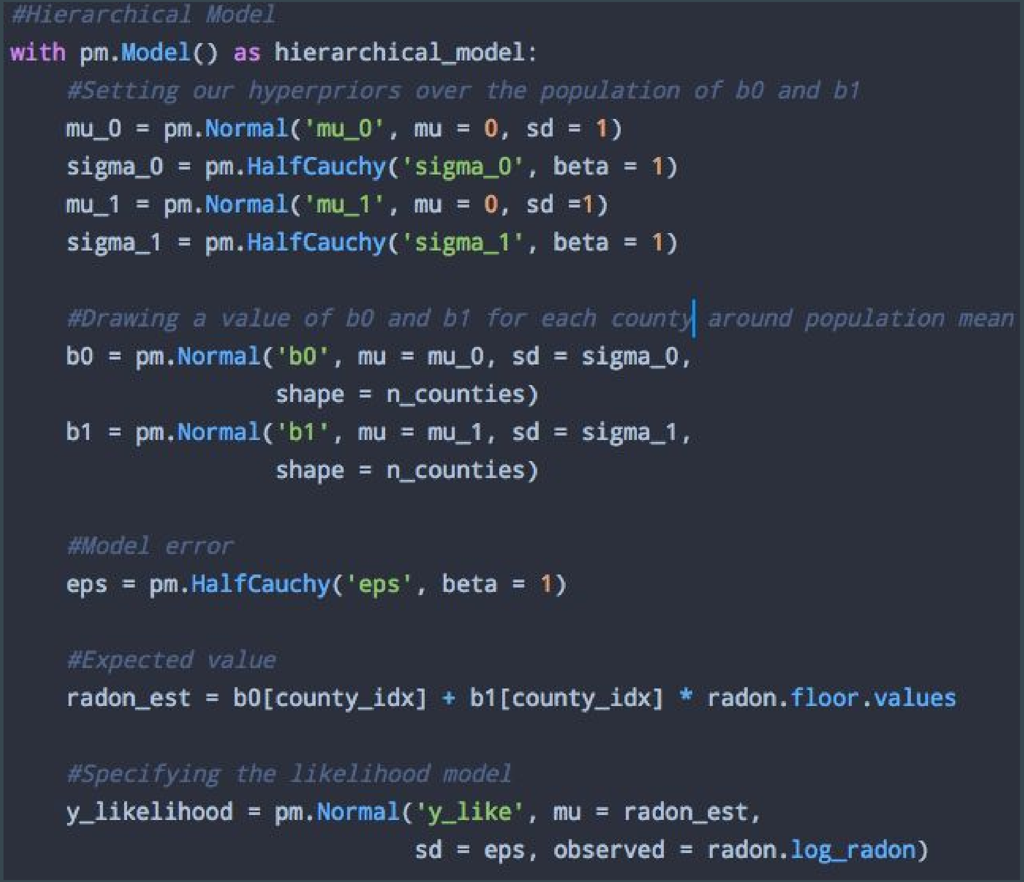
- Structure within the loop is the same.
- Set priors
- Set up predictive form
- Likelihood
- Fit



```
#Modelling
with pm.Model() as individual:
    #Specifying Intercept prior
    b0 = pm.Normal('b0', mu = 0, sd = 1)

    #Specifying Slope Prior
    b1 = pm.Normal('b1', mu = 0, sd = 1)

    # Specifying error prior
    eps = pm.HalfCauchy('eps', beta = 1)

    #Specifying our linear model
    radon_est = b0 + b1 * county_floor

    #Specifying the likelihood
    y_likelihood = pm.Normal('y_like', mu = radon_est,
                            sd = eps, observed = county_radon)

    #Performing the inference sampling via MCMC
    trace = pm.sample(progressbar = False)

county_traces[county_name] = trace
```

Hierarchical Model

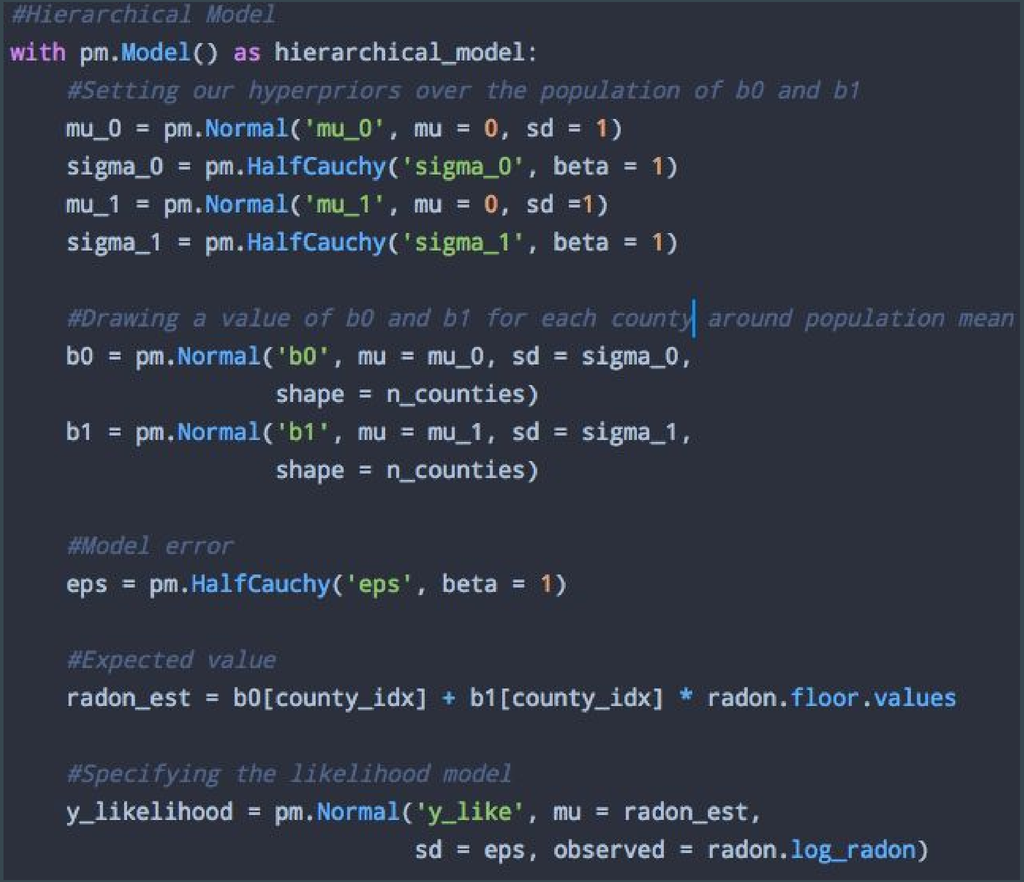
- Specify priors for our hyperparameters (the population means and such)
- Specify the n-county coefficents as draws from our normal distributions with the hyperprior parameters



```
#Hierarchical Model
with pm.Model() as hierarchical_model:
    #Setting our hyperpriors over the population of b0 and b1
    mu_0 = pm.Normal('mu_0', mu = 0, sd = 1)
    sigma_0 = pm.HalfCauchy('sigma_0', beta = 1)
    mu_1 = pm.Normal('mu_1', mu = 0, sd =1)
    sigma_1 = pm.HalfCauchy('sigma_1', beta = 1)

    #Drawing a value of b0 and b1 for each county around population mean
    b0 = pm.Normal('b0', mu = mu_0, sd = sigma_0,
                   shape = n_counties)
    b1 = pm.Normal('b1', mu = mu_1, sd = sigma_1,
                   shape = n_counties)

    #Model error
    eps = pm.HalfCauchy('eps', beta = 1)

    #Expected value
    radon_est = b0[county_idx] + b1[county_idx] * radon.floor.values

    #Specifying the likelihood model
    y_likelihood = pm.Normal('y_like', mu = radon_est,
                             sd = eps, observed = radon.log_radon)
```

Hierarchical Model

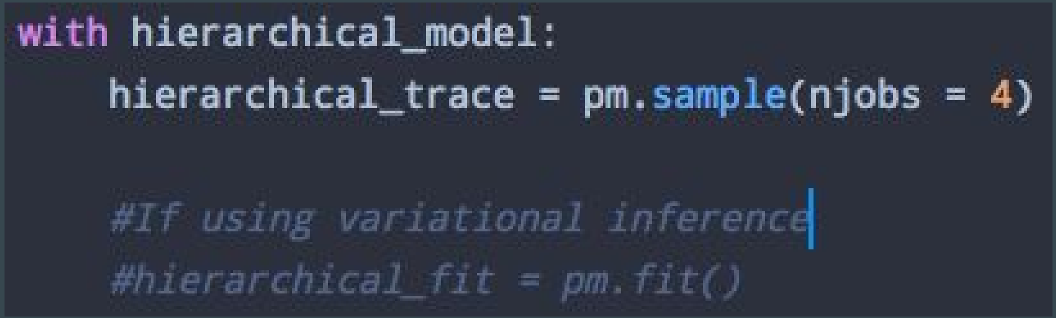
- Set up the predictive form
- Specify likelihood



```
#Hierarchical Model
with pm.Model() as hierarchical_model:
    #Setting our hyperpriors over the population of b0 and b1
    mu_0 = pm.Normal('mu_0', mu = 0, sd = 1)
    sigma_0 = pm.HalfCauchy('sigma_0', beta = 1)
    mu_1 = pm.Normal('mu_1', mu = 0, sd =1)
    sigma_1 = pm.HalfCauchy('sigma_1', beta = 1)

    #Drawing a value of b0 and b1 for each county around population mean
    b0 = pm.Normal('b0', mu = mu_0, sd = sigma_0,
                   shape = n_counties)
    b1 = pm.Normal('b1', mu = mu_1, sd = sigma_1,
                   shape = n_counties)

    #Model error
    eps = pm.HalfCauchy('eps', beta = 1)

    #Expected value
    radon_est = b0[county_idx] + b1[county_idx] * radon.floor.values

    #Specifying the likelihood model
    y_likelihood = pm.Normal('y_like', mu = radon_est,
                             sd = eps, observed = radon.log_radon)
```

Hierarchical Model

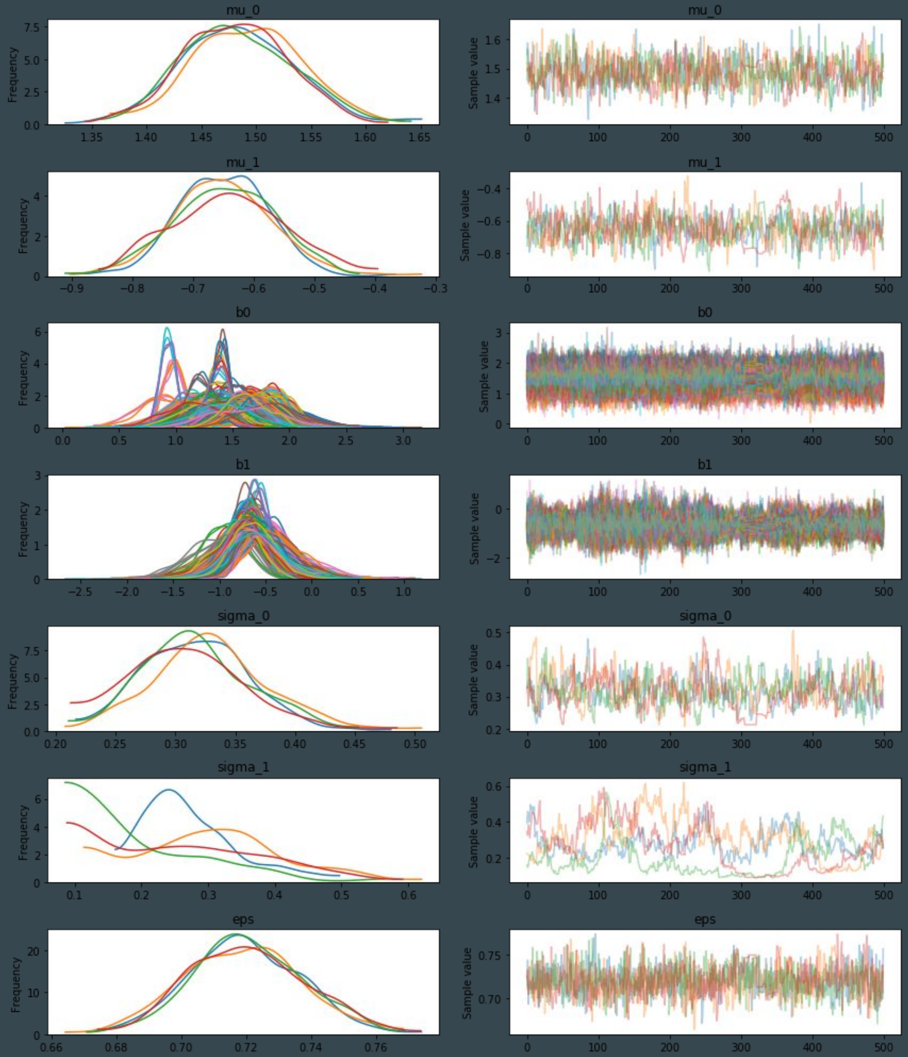
- Fit the model
- The argument njobs = 4 specifies that we will run 4 different MCMC chains
- Typically running multiple and comparing them gives you a feel of if it has converged



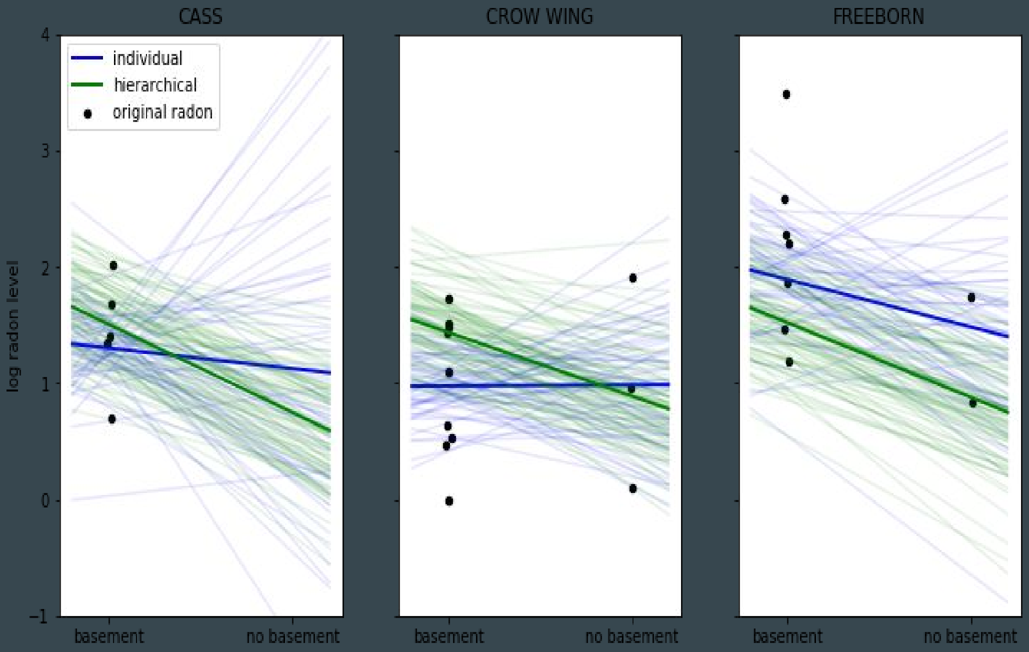
```
with hierarchical_model:
    hierarchical_trace = pm.sample(njobs = 4)

    #If using variational inference
    #hierarchical_fit = pm.fit()
```

Traceplots & Posteriors

- First four and last one all look fine.
  - Good mixing, looks like white noise
- The ones for the hyperparameters on the variances look terrible.
  - May need to run more iterations
  - May need to use different hyperpriors
- Gives us good feel for how each of the counties parameters differ from one another

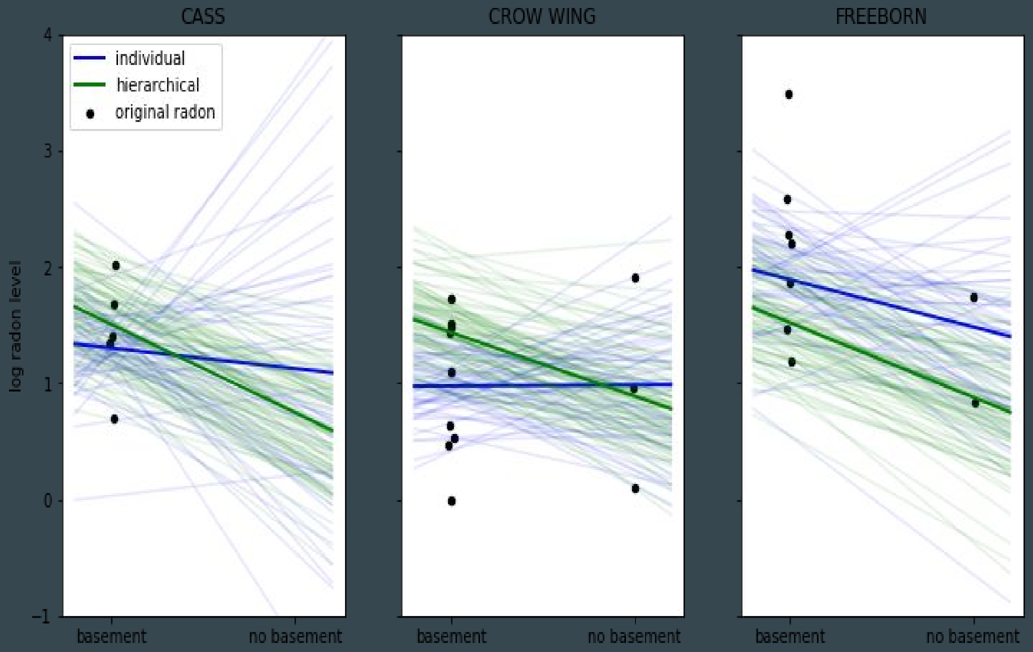


Comparison of Models



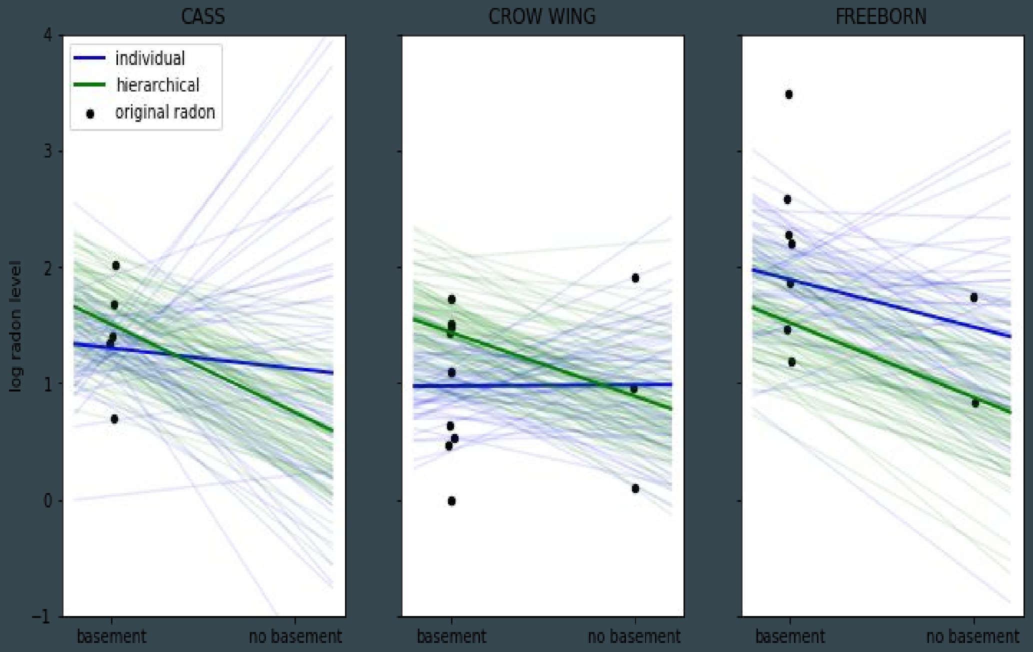
- Shows the differences between partial pooled and no pooling estimates.
- Bold lines are the means of respective posteriors
- Light colored lines represent other draws from posterior showing uncertainty

Comparison of Models



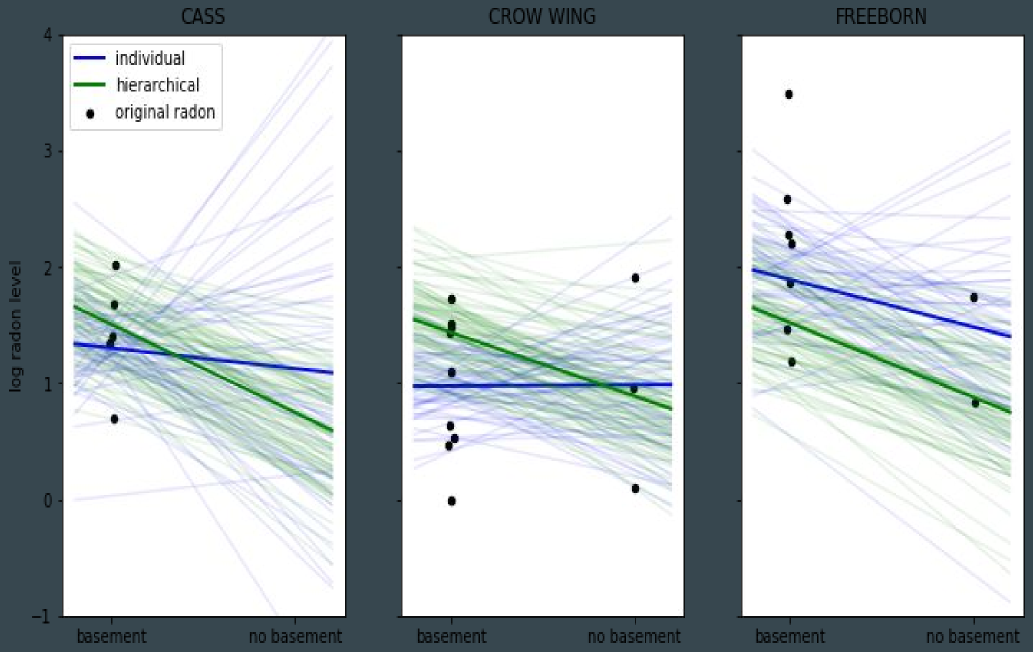
- Hierarchical model has less uncertainty in predictions (tighter grouping).
- In crow wing, in individual fit had increase in radon level in non-basement. We know this doesn't make sense. Hierarchical model knows this too by sharing information with other counties.

Comparison of Models



- We only have basement readings in Cass county, individual model doesn't handle that well
- Hierarchical does because it knows what other counties do.

Comparison of Models



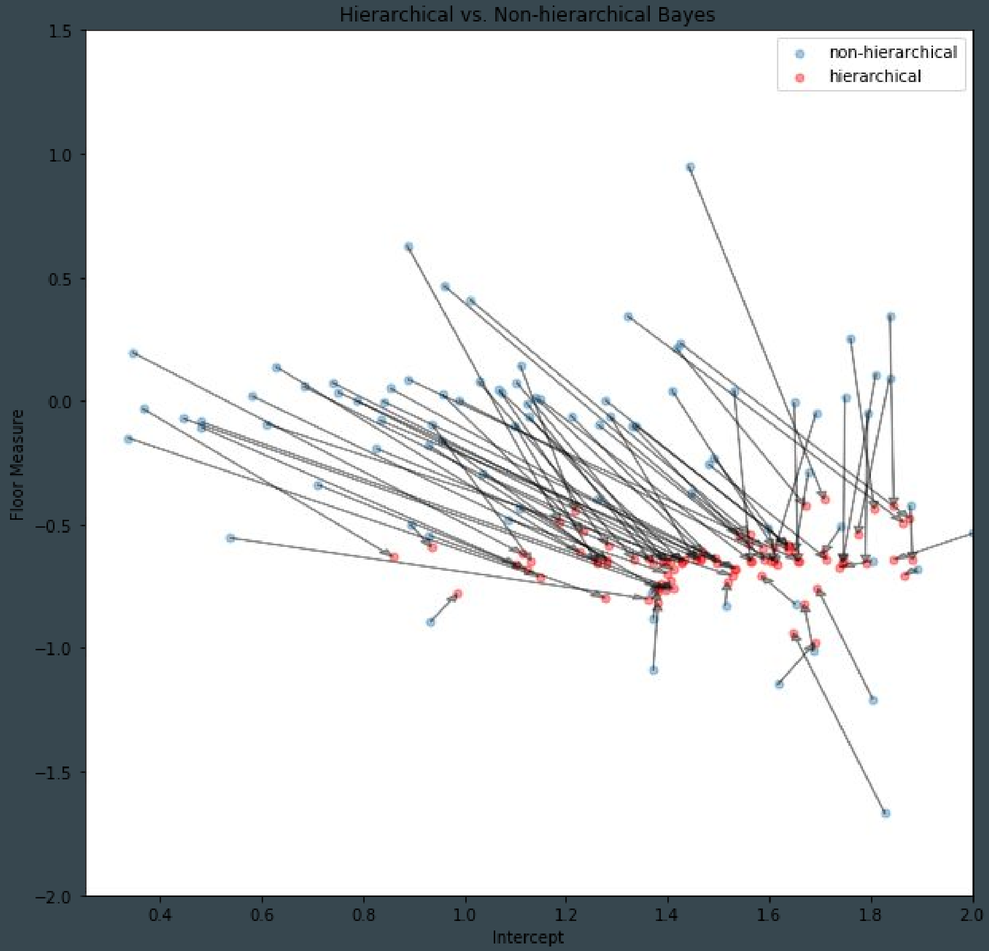
- Freeborn had very high readings. Much higher than other counties.
- Freeborn probably not a radon hotspot probably just random chance that one county in 85 would look like this.
- Hierarchical model knows that and accounts for it.

A Little More on Hierarchical In General

- Shrinkage - the idea of pulling the individual values towards the population mean
- We expect individuals to behave similarly to each other due to a shared structure.
- The more observations you have for a single individual the more it will get to speak for itself and stand alone from a population.
- If you only have a few observations, it’s better to view them as very close to the population overall.
- Fewer individual observations means more shrinkage, more observations less shrinkage.
- A county with 5 high radon readings is likely due to chance and should be closer to the population, but if you have 500 high radon readings then you know something is up.

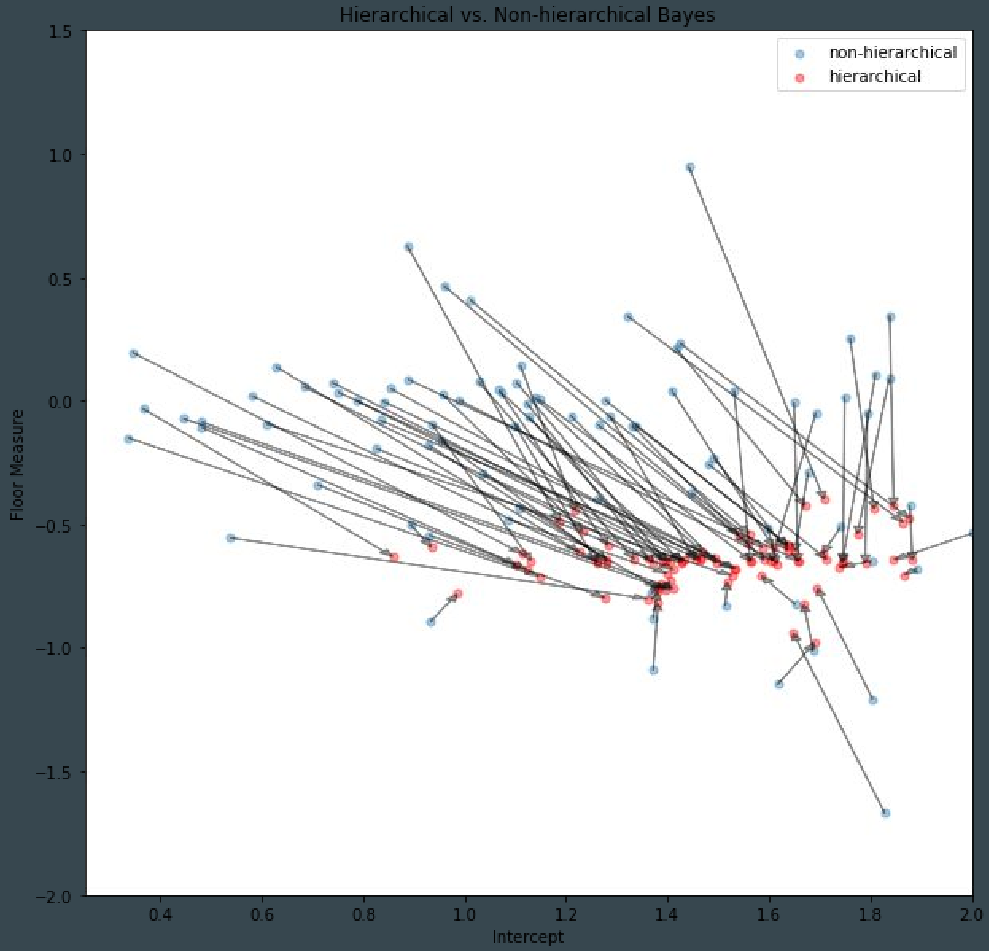
Shrinkage For Our Problem

- Individual estimates connected to their hierarchical estimate.
- Look how much more spread out all the values you are for the individual estimates.
- Since counties are largely arbitrary geographically that doesn't really make sense.
- Expect differences but not big differences.



Shrinkage For Our Problem

- More variance in the intercept between county than in the effect of moving from the basement to the first floor.
- This makes intuitive sense, wouldn't expect that relationship to change much.
- However the amount of radon in the ground is more likely to vary.



For more information

- Bayesian Statistics:
  - Andrew Gelman and company's book "Bayesian Data Analyis" or "Doing Bayesian Data Analysis" by John Kruschke
  - Online Coursera Courses with Duke
  - Ryan Bakker in the political science department teaches a fantastic graduate level course in Bayesian Analysis
- PyMC3
  - Loads of Tutorials online
  - Documentation is pretty clear

Other parting thoughts

- This is very simple cursory example to show basic of Bayes and some of its philosophical and practical strengths.
- It’s an incredibly rich and power field.
- For a lot of things, using R in conjunction with Stan can be easier and more fluid than PyMC3 due to other supporting packages.

Special Thanks

- Thomas Wiecki’s tutorial for how to make the dope plots
- Ryan Bakker