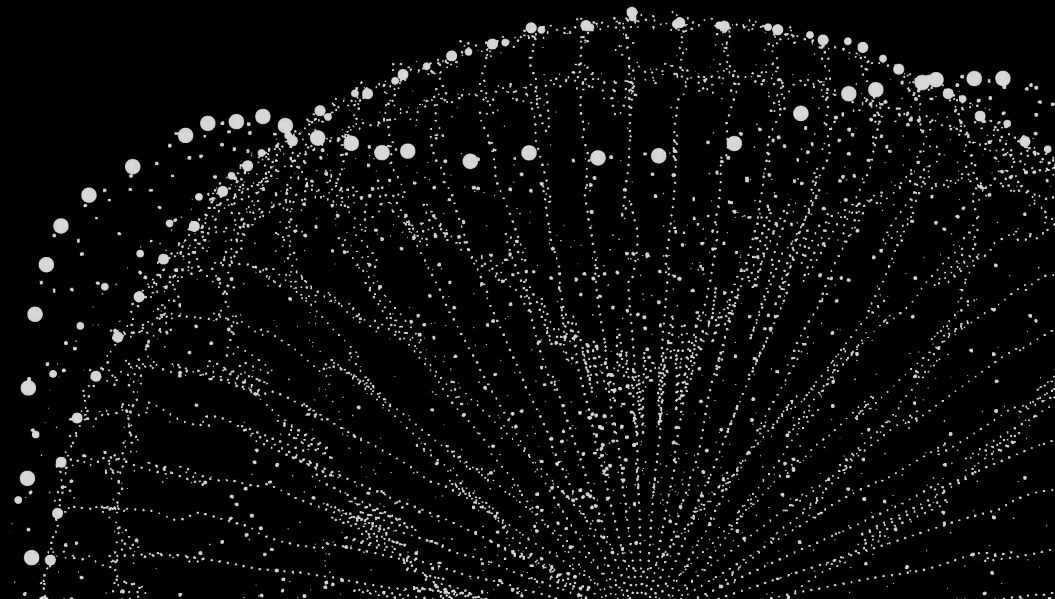
Hierarchical *modelling*

Omar Sosa Rodriguez

June 2021



faculty

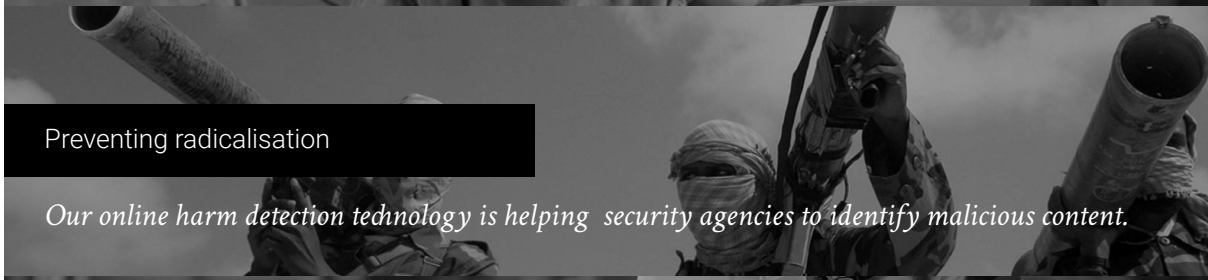

Faculty exists to
make AI real

Faculty is a world leader in *applied AI*

We configure, deploy and maintain high performance AI systems to solve our customers' most important problems.

Predicting COVID hospitalisations

Our forecasting technology is helping the NHS to manage critical resources through COVID-19.



Preventing radicalisation

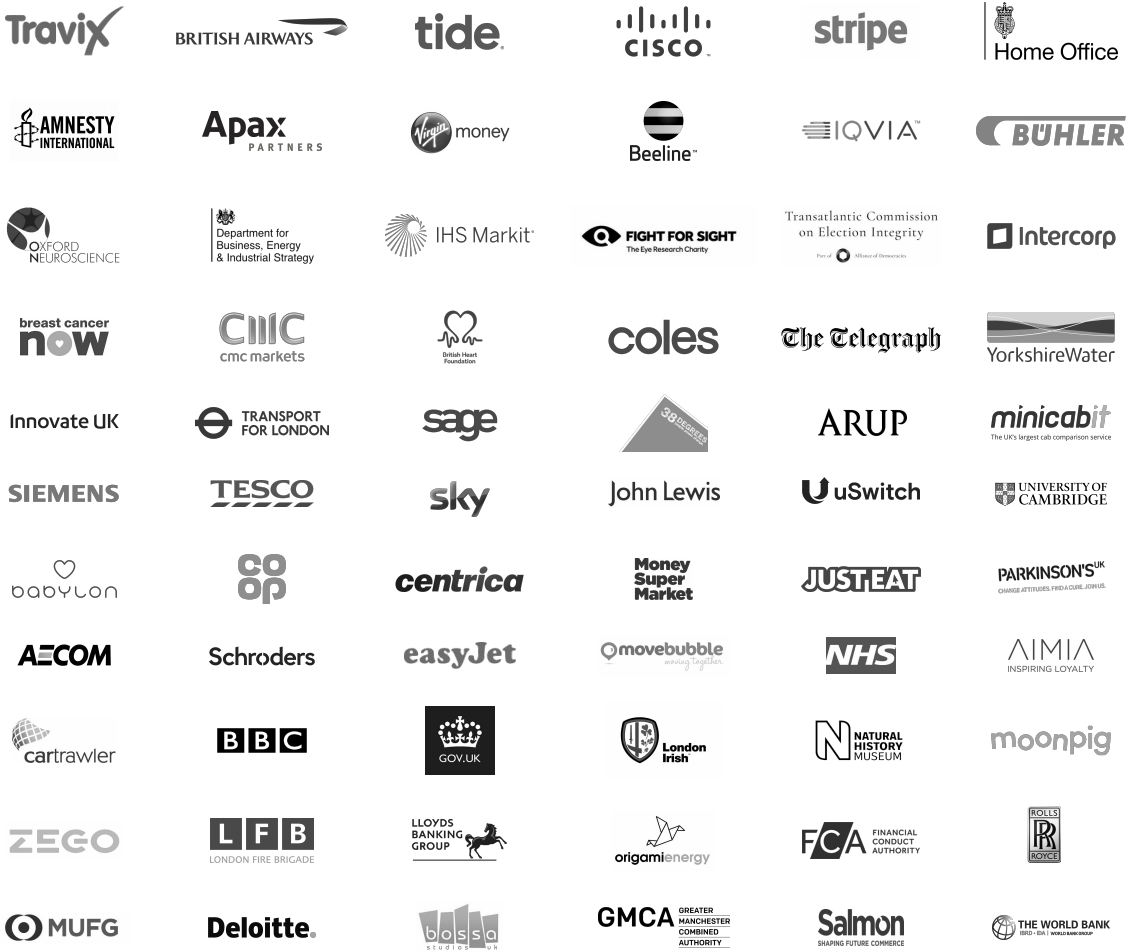
Our online harm detection technology is helping security agencies to identify malicious content.

Improving asset performance

Our computer vision technology automates the inspection of physical infrastructure.

Optimising marketing spend

Our uplift modelling technology is helping a fashion retailer to optimise a $200m marketing spend.



We have delivered

230

data science customers

We work in

13

countries

We cover

23

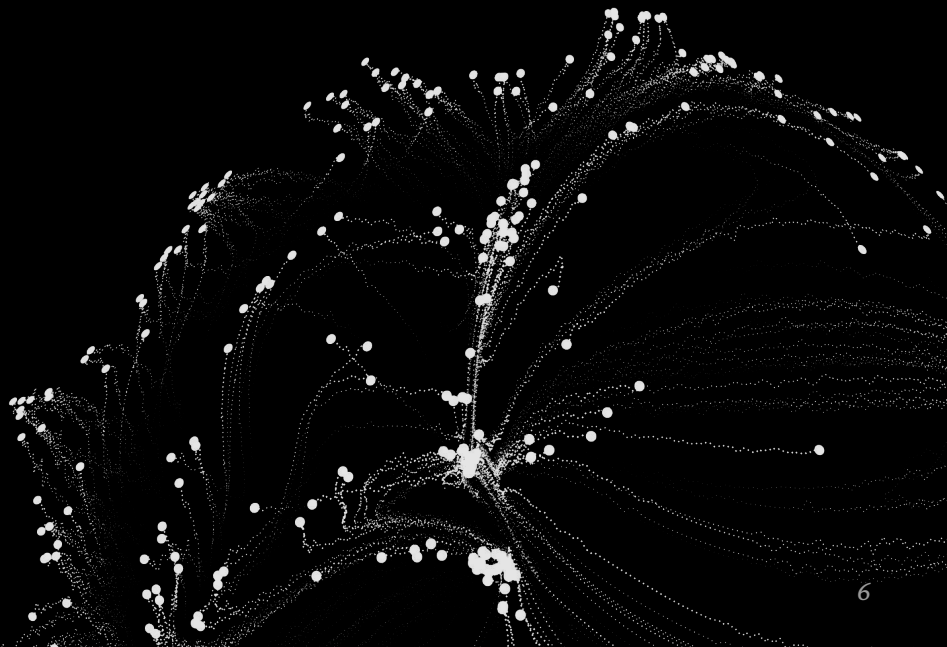
sectors

Contents

01

Multilevel structures

01.1 Example: Radon in Minnesota



Very often the data that we are trying to model will have some categorical features.

NetflixOriginals.csv (37.77 KB)

Detail | Compact | Column

Title	Genre	Premiere	Runtime	IMDB Score	Language
Enter the Anime	Documentary	August 5, 2019	58	2.5	English/Japanese
Dark Forces	Thriller	August 21, 2020	81	2.6	Spanish
The App	Science fiction/Drama	December 26, 2019	79	2.6	Italian
The Open House	Horror thriller	January 19, 2018	94	3.2	English
Kaali Khuhi	Mystery	October 30, 2020	90	3.4	Hindi
Drive	Action	November 1, 2019	147	3.5	Hindi
Leyla Everlasting	Comedy	December 4, 2020	112	3.7	Turkish
The Last Days of American Crime	Heist film/Thriller	June 5, 2020	149	3.7	English
Paradox	Musical/Western /Fantasy	March 23, 2018	73	3.9	English

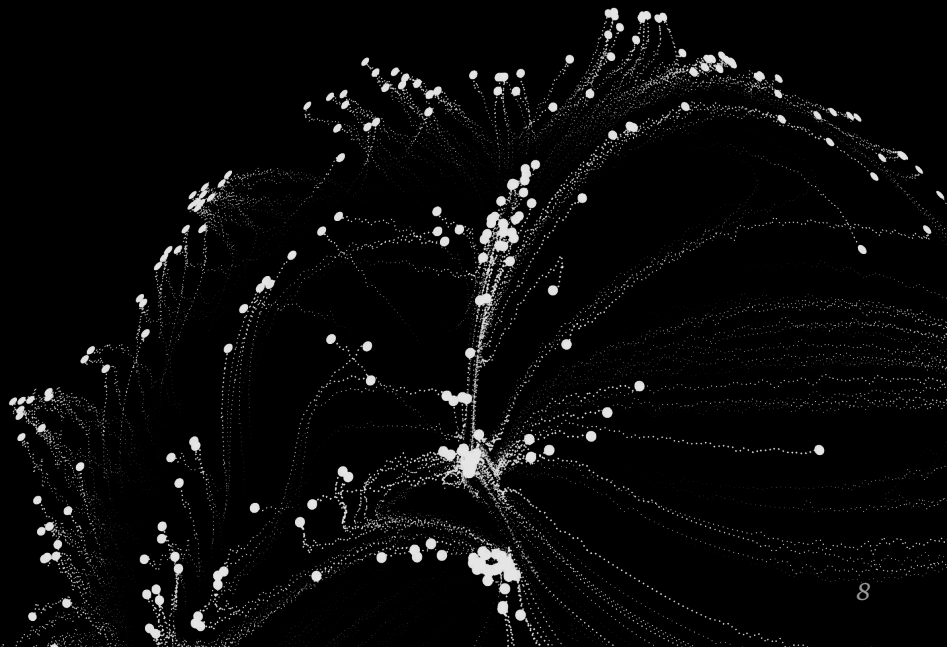
Other Examples

01. Geography
02. Demographics
03. Culture
04. Dates*
05. Names / IDs
06. Species

etc.

01 *Multilevel* structures

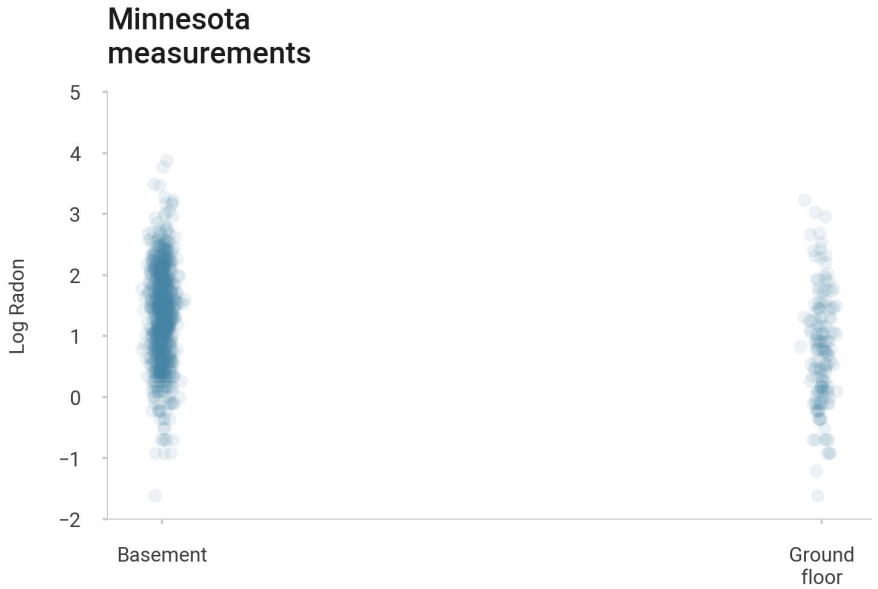
01.1 Example: Radon in Minnesota



Radon concentration was measured in houses across many **counties** in Minnesota.
Sometimes in the basement, sometimes in the ground floor.

	county	floor	log_radon
0	Aitkin	0	0.788457
1	Aitkin	-1	0.788457
2	Aitkin	-1	1.064711
3	Aitkin	-1	0.000000
4	Anoka	-1	1.131402
...	...	...	...
914	Wright	-1	1.856298
915	Wright	-1	1.504077
916	Wright	-1	1.609438
917	Yellow Medicine	-1	1.308333
918	Yellow Medicine	-1	1.064711

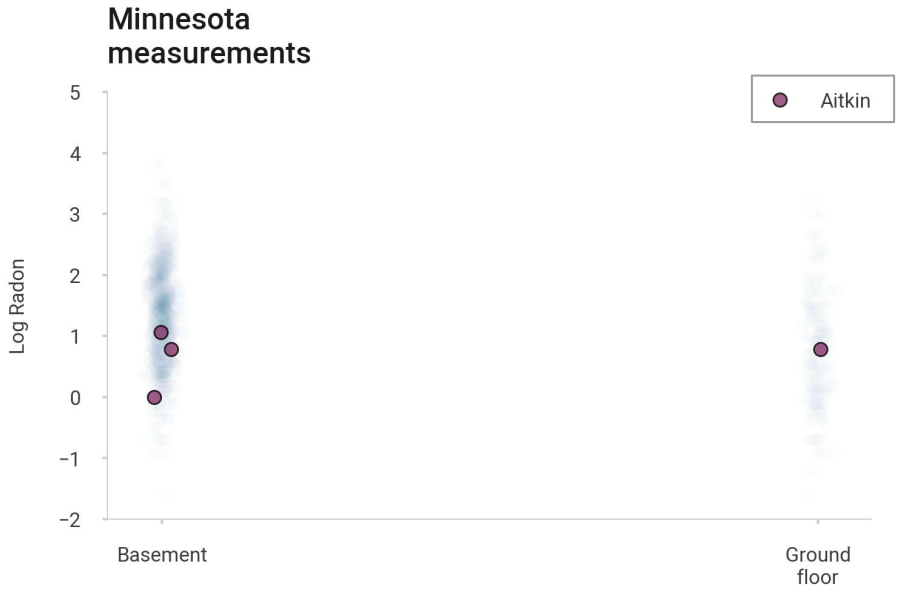
919 rows × 3 columns



Some counties have a very **limited number of observations**

	county	floor	log_radon
0	Aitkin	0	0.788457
1	Aitkin	-1	0.788457
2	Aitkin	-1	1.064711
3	Aitkin	-1	0.000000
4	Anoka	-1	1.131402
...	...	...	...
914	Wright	-1	1.856298
915	Wright	-1	1.504077
916	Wright	-1	1.609438
917	Yellow Medicine	-1	1.308333
918	Yellow Medicine	-1	1.064711

919 rows × 3 columns



Some counties have a very **limited number of observations**

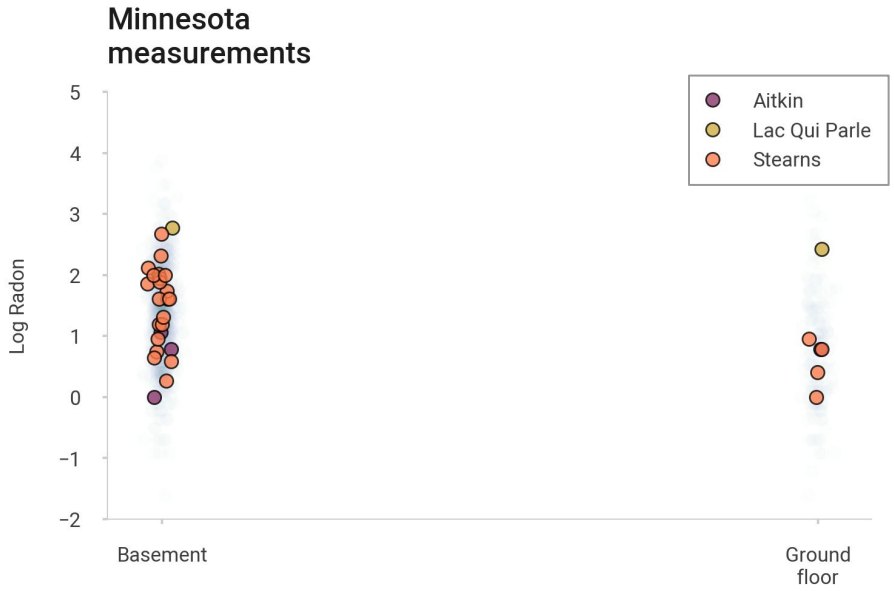
	county	floor	log_radon
0	Aitkin	0	0.788457
1	Aitkin	-1	0.788457
2	Aitkin	-1	1.064711
3	Aitkin	-1	0.000000
4	Anoka	-1	1.131402
...	...	...	...
914	Wright	-1	1.856298
915	Wright	-1	1.504077
916	Wright	-1	1.609438
917	Yellow Medicine	-1	1.308333
918	Yellow Medicine	-1	1.064711

919 rows × 3 columns

Only a few counties have enough data

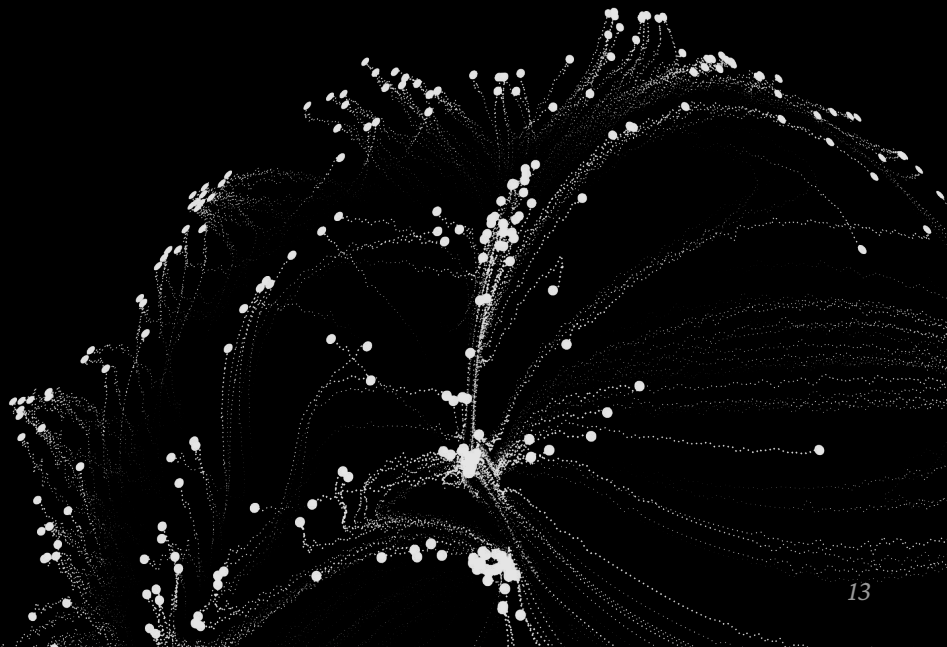
	county	floor	log_radon
0	Aitkin	0	0.788457
1	Aitkin	-1	0.788457
2	Aitkin	-1	1.064711
3	Aitkin	-1	0.000000
4	Anoka	-1	1.131402
...	...	...	...
914	Wright	-1	1.856298
915	Wright	-1	1.504077
916	Wright	-1	1.609438
917	Yellow Medicine	-1	1.308333
918	Yellow Medicine	-1	1.064711

919 rows × 3 columns



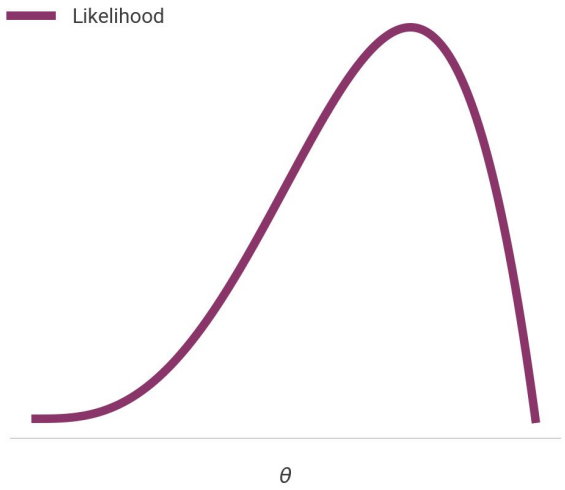
02

Bayesian inference



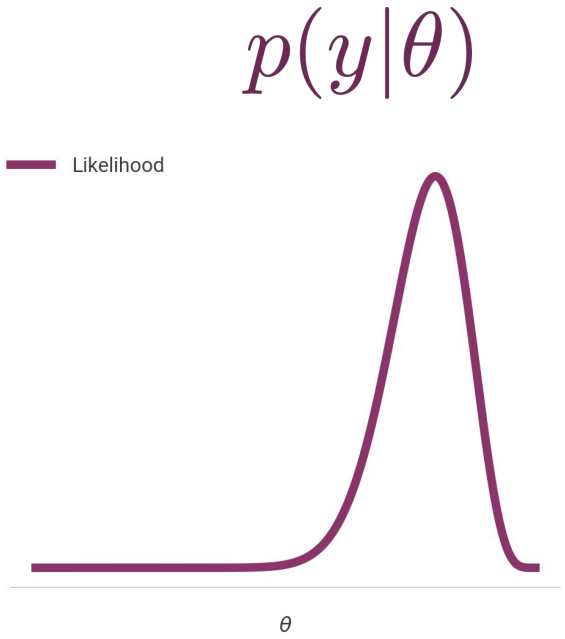
faculty

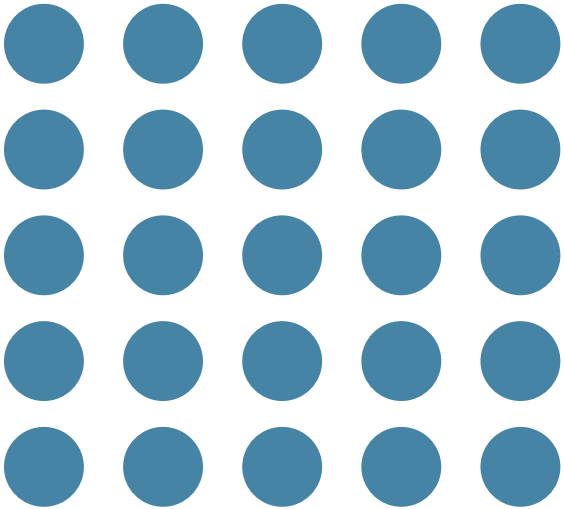
We start with a **likelihood** function which encodes our assumption about the data generating process. It tells us which values of the parameters are more “consistent” with the data we have.



$$y \xleftarrow{p(y|\theta)} \theta$$

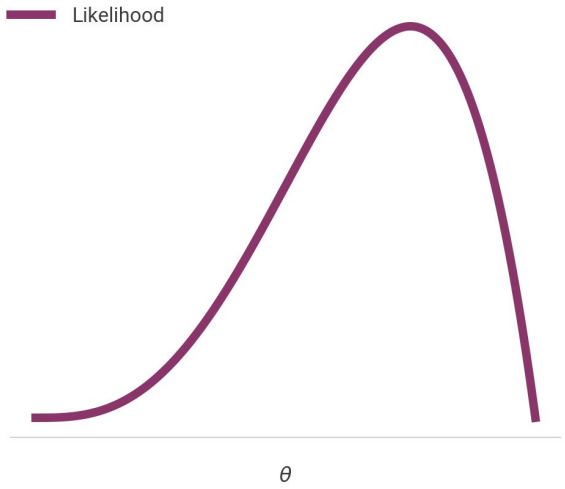
Informative data results in a *tight* likelihood function.



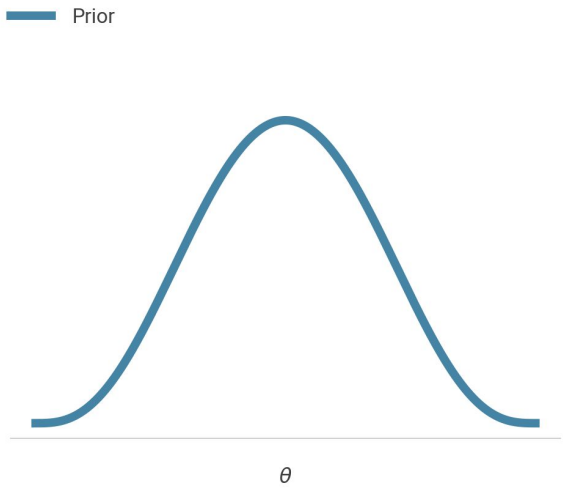


Uninformative data yields a *diffuse* likelihood.

$$p(y|\theta)$$



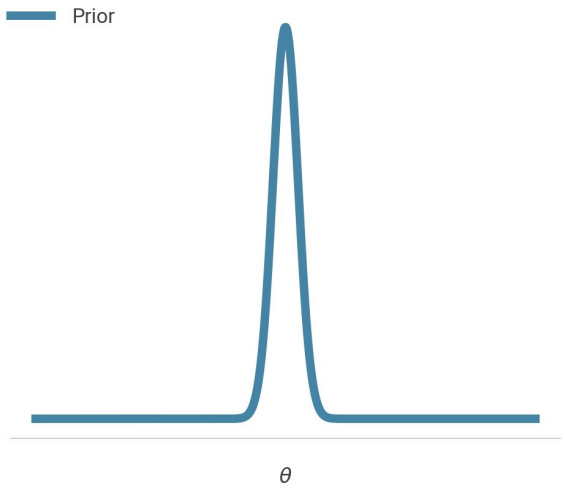
Bayesian inference also quantifies domain expertise through a **prior** probability distribution.



$$p(\theta)$$

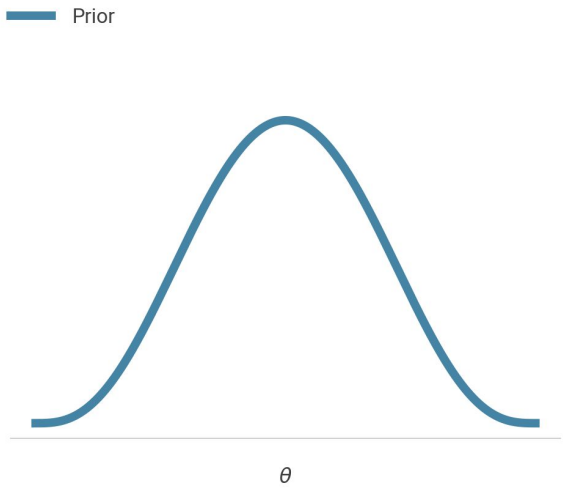
A *tight* prior distribution represents high confidence in our domain expertise.

$$p(\theta)$$

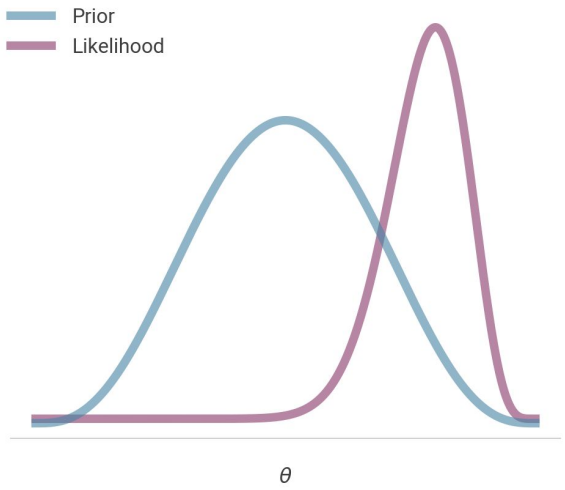


A *diffuse* prior represents high uncertainty or poor domain expertise.

$$p(\theta)$$

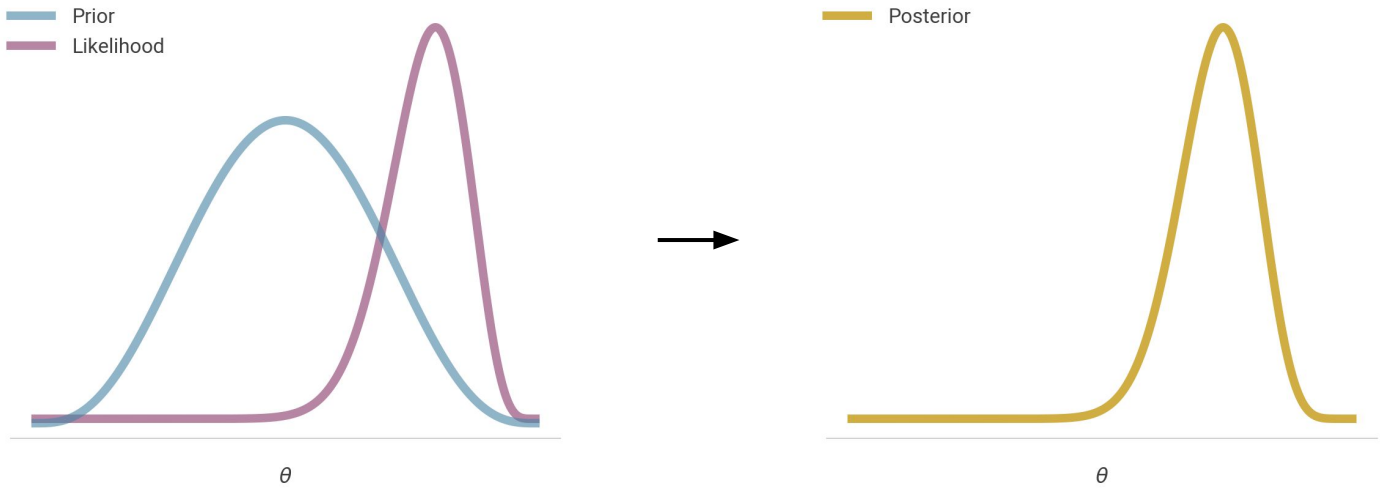


Bayesian inference is the result of a rigorous combination between prior knowledge and observations.



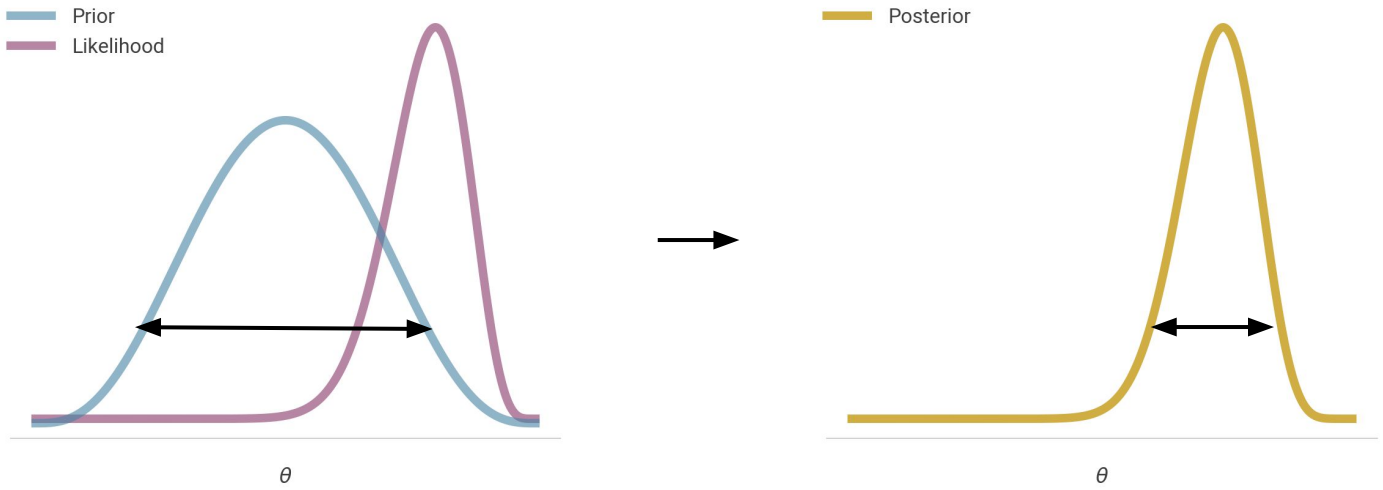
Our **posterior** knowledge is obtained via *Bayes'* rule.

$$p(\theta)\,p(y|\theta) \propto p(\theta|y)$$



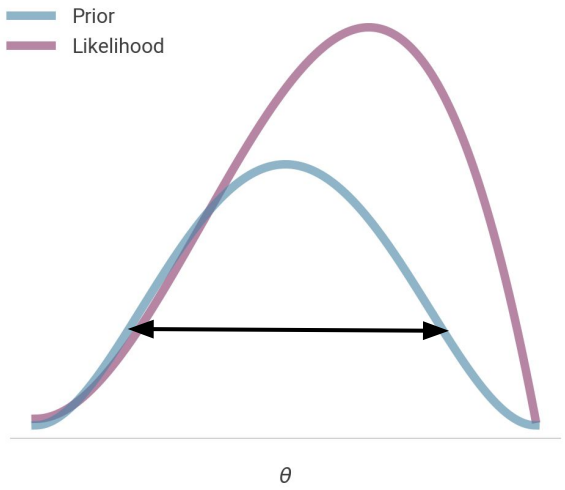
Our **posterior** knowledge is obtained via *Bayes'* rule.

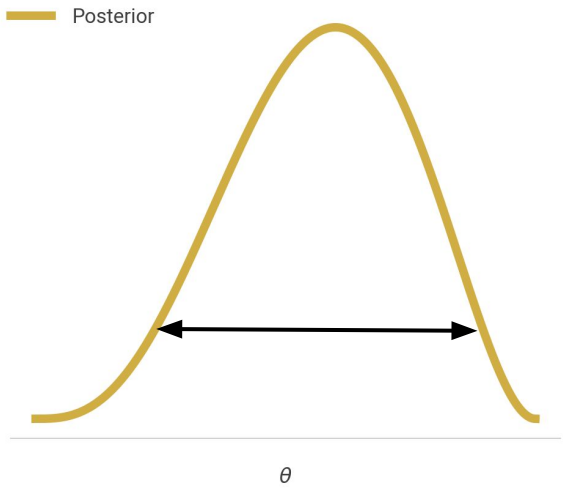
$$p(\theta)\,p(y|\theta) \propto p(\theta|y)$$



Poor domain expertise and non-informative data is the correct recipe for cooking poor inferences.

$$p(\theta)\,p(y|\theta) \propto p(\theta|y)$$



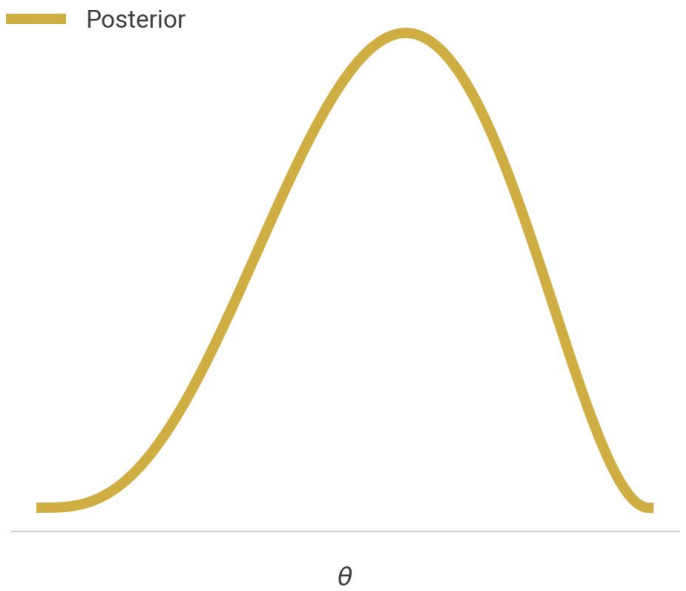


The functional form form of the posterior *is not* the final goal. We want to use it compute **expectation** values, which requires solving complicated integrals

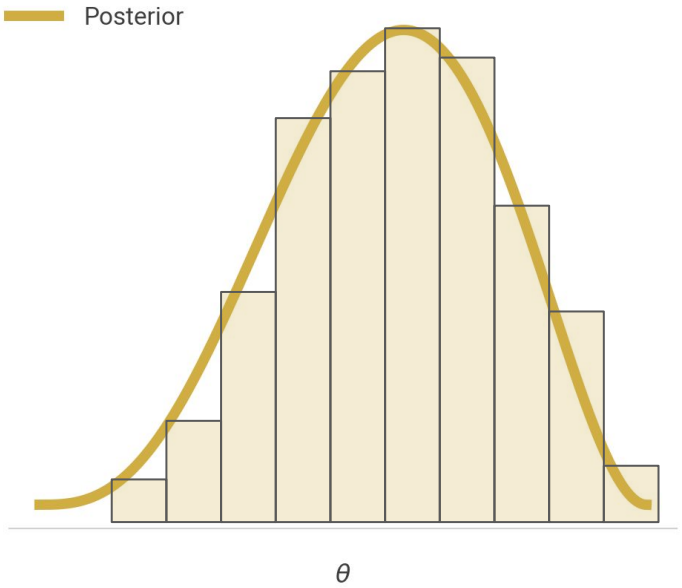
$$\mathbb{E}\left[f(\theta)\right] = \int f(\theta)p(\theta|y)d\theta$$

Questions are functions.

Answers are expectations

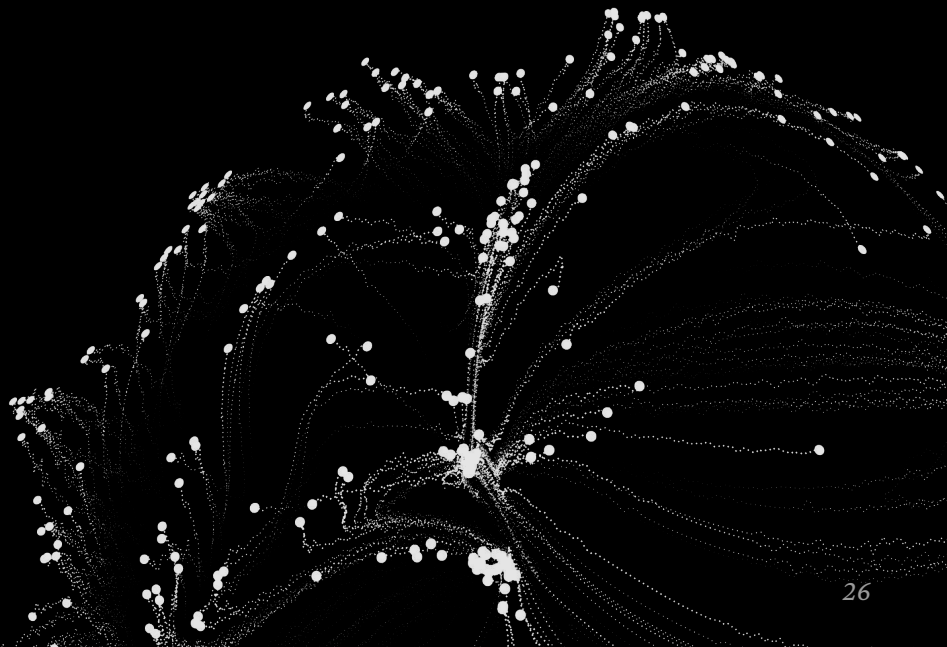


Any expectation can be approximated with **samples** and MCMC is the gold standard for sampling from an *arbitrary distribution*. Efficient MCMC is implemented in many python libraries.



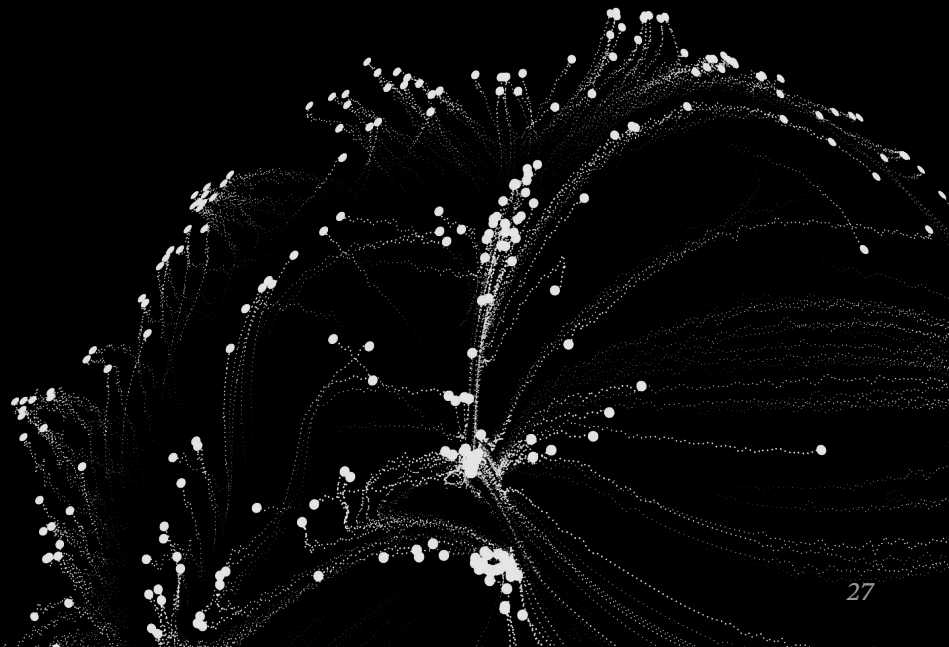
03

Traditional approaches



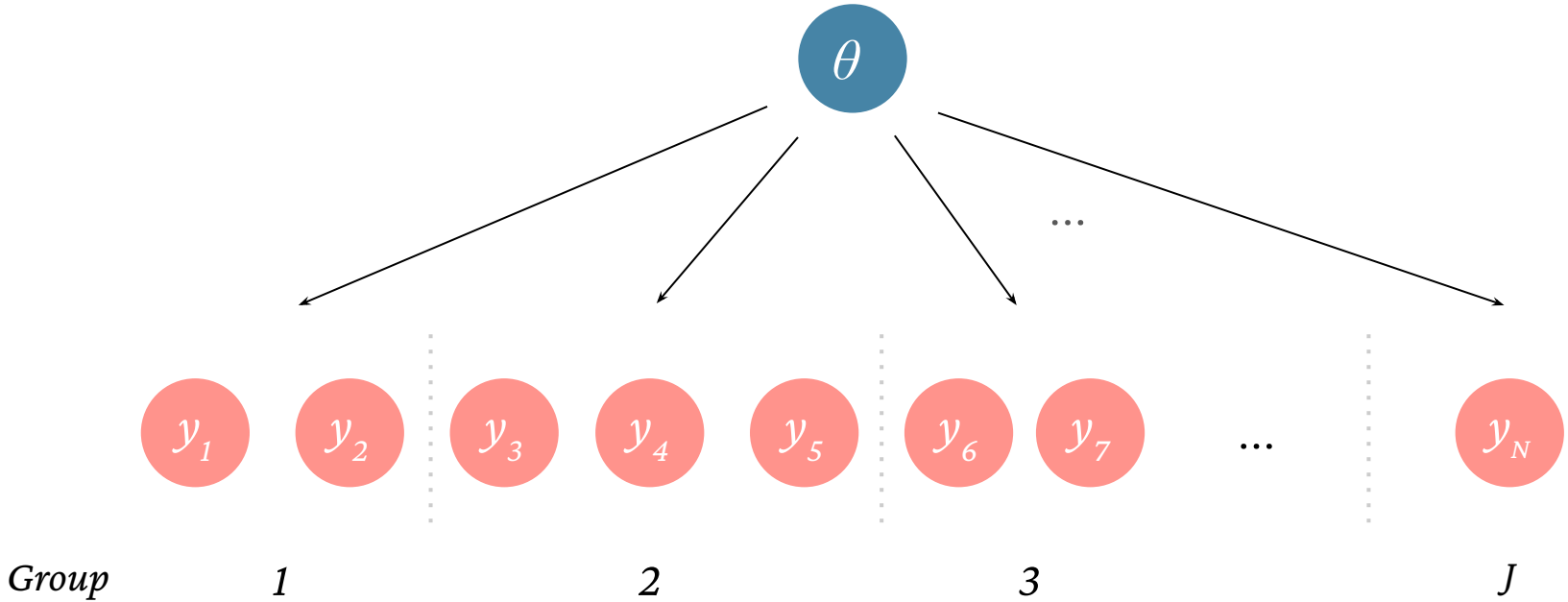
faculty

03
Traditional approaches

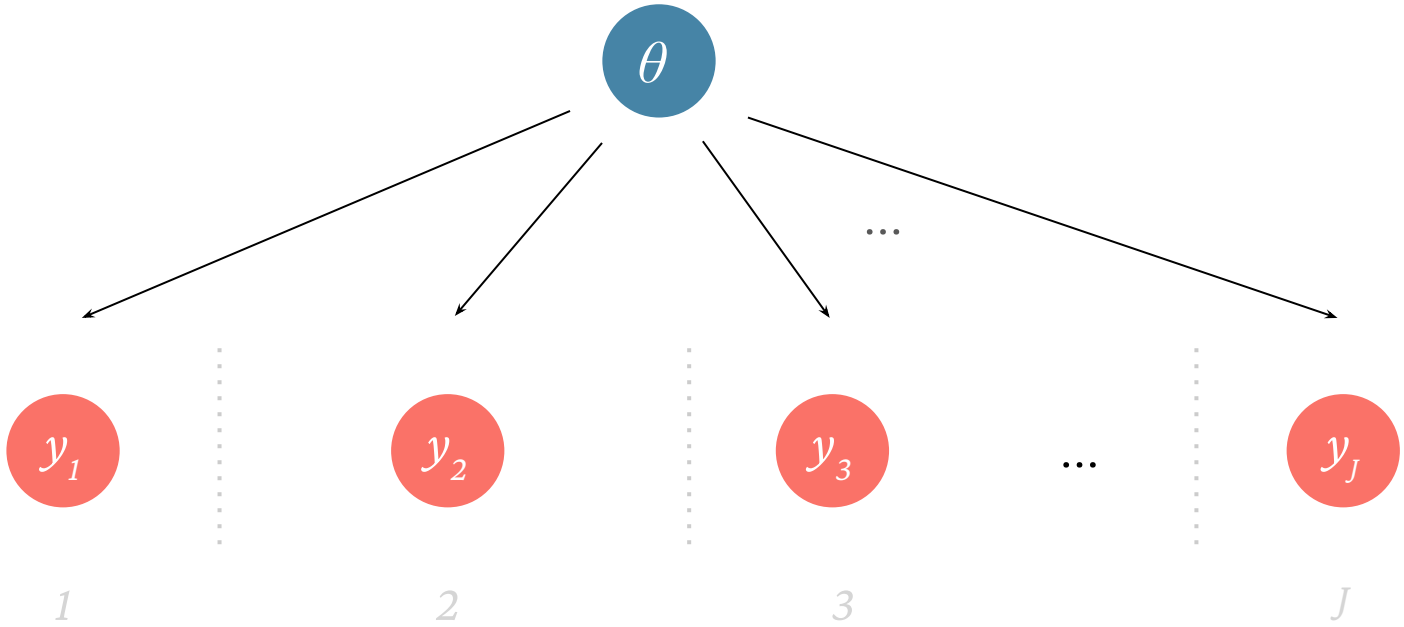


faculty

Complete pooling ignores the group-level information and considers all data as belonging to the same category. All groups are described with the same model.



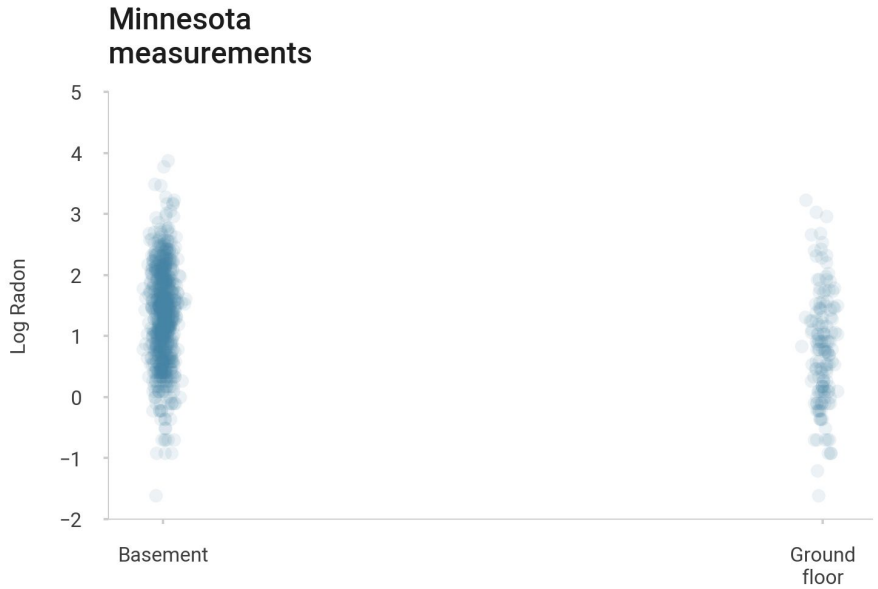
Note: we will reserve the index on the data variables to label the group-membership as opposed to the usual practice of labelling the individual observations.



Data for *all* counties in Minnesota is modelled with the same linear model.

$$y \sim \text{normal}(\alpha + \beta \cdot \text{floor}, \sigma)$$

Likelihood: $p(y|\theta)$

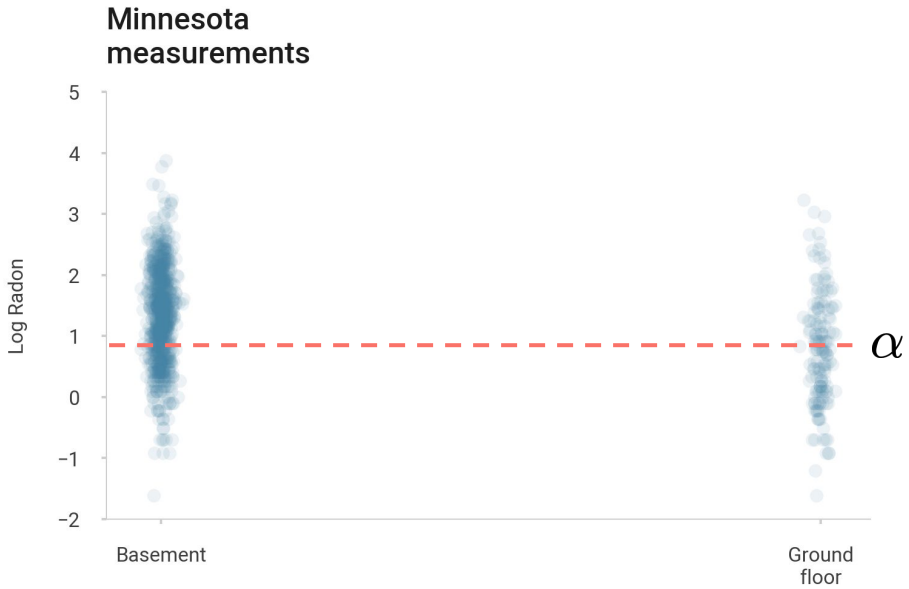


Data for *all* counties in Minnesota is modelled with the same linear model.

$$y \sim \text{normal}(\alpha + \beta \cdot \text{floor}, \sigma)$$

Likelihood: $p(y|\theta)$

α captures the *average* level at the ground floor.



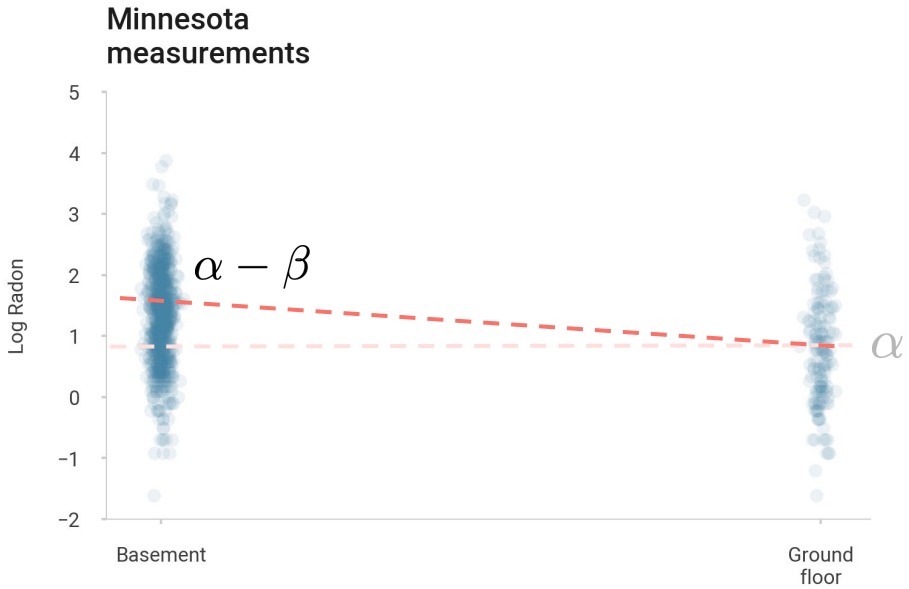
Data for *all* counties in Minnesota is modelled with the same linear model.

$$y \sim \text{normal}(\alpha + \beta \cdot \text{floor}, \sigma)$$

Likelihood: $p(y|\theta)$

α captures the *average* level at the ground floor.

β is the *average* increase from basement to ground floor



Data for *all* counties in Minnesota is modelled with the same linear model.

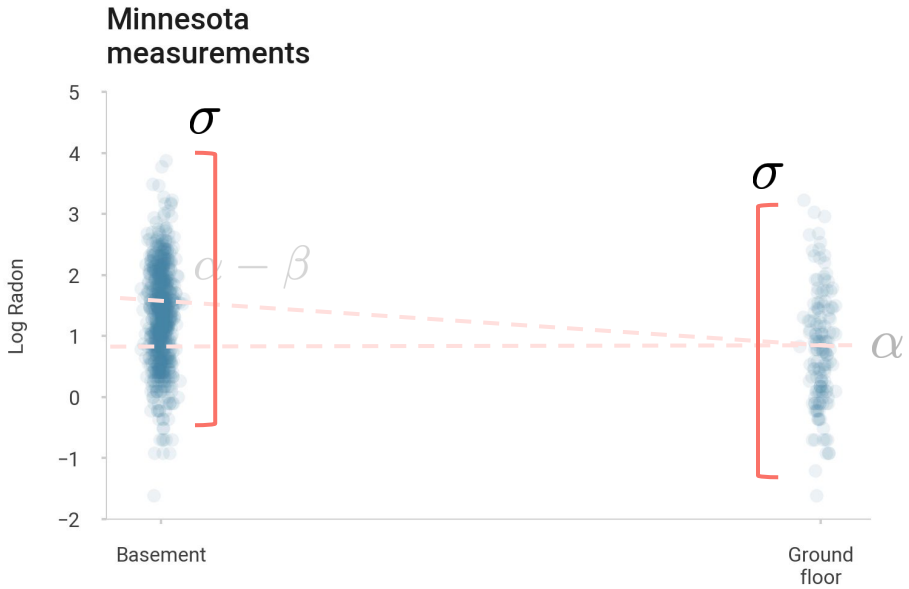
$$y \sim \text{normal}(\alpha + \beta \cdot \text{floor}, \sigma)$$

Likelihood: $p(y|\theta)$

α captures the *average* level at the ground floor.

β is the *average* increase from basement to ground floor

σ measures simultaneously the *within*-county variation and *across* counties



Data for *all* counties in Minnesota is modelled with the same linear model.

$$y \sim \text{normal}(\alpha + \beta \cdot \text{floor}, \sigma)$$

$$\alpha \sim \text{normal}(0, 5)$$

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

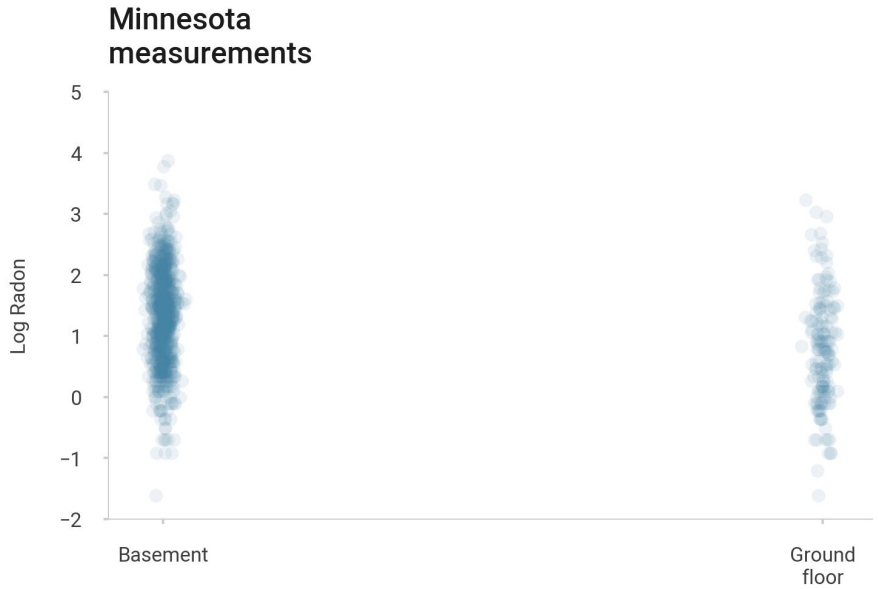
priors: $p(\theta)$

$$\frac{1}{\sigma} \sim \text{gamma}(1, 0.5)$$

α captures the *average* level at the ground floor.

β is the *average* increase from basement to ground floor

σ measures simultaneously the *within*-county variation and *across* counties



```
...
from numpyro import sample, deterministic
from numpyro.distributions import Normal, InverseGamma

def complete_pooling(floor, log_radon=None):
    α = sample("α", Normal(0, 5))
    β = sample("β", Normal(0, 1))

    μ = deterministic("μ", α + β * floor)
    σ = sample("σ", InverseGamma(1, 0.5))
    sample("log_radon", Normal(μ, σ), obs=log_radon)
```

Numpyro provides some helper functions to build models


```
...
from numpyro import sample, deterministic
from numpyro.distributions import Normal, InverseGamma

def complete_pooling(floor, log_radon=None):
    α = sample("α", Normal(0, 5))
    β = sample("β", Normal(0, 1))

    μ = deterministic("μ", α + β * floor)
    σ = sample("σ", InverseGamma(1, 0.5))
    sample("log_radon", Normal(μ, σ), obs=log_radon)
```

All of our model is defined in a function.

```
...
from numpyro import sample, deterministic
from numpyro.distributions import Normal, InverseGamma

def complete_pooling(floor, log_radon=None):
    α = sample("α", Normal(0, 5))
    β = sample("β", Normal(0, 1))

    μ = deterministic("μ", α + β * floor)
    σ = sample("σ", InverseGamma(1, 0.5))
    sample("log_radon", Normal(μ, σ), obs=log_radon)
```

Sample statements contribute to the total posterior density


```
...
from numpyro import sample, deterministic
from numpyro.distributions import Normal, InverseGamma

def complete_pooling(floor, log_radon=None):
    α = sample("α", Normal(0, 5))
    β = sample("β", Normal(0, 1))

    μ = deterministic("μ", α + β * floor)
    σ = sample("σ", InverseGamma(1, 0.5))
    sample("log_radon", Normal(μ, σ), obs=log_radon)
```

Some statements correspond to the priors


```
...
from numpyro import sample, deterministic
from numpyro.distributions import Normal, InverseGamma

def complete_pooling(floor, log_radon=None):
    α = sample("α", Normal(0, 5))
    β = sample("β", Normal(0, 1))

    μ = deterministic("μ", α + β * floor)
    σ = sample("σ", InverseGamma(1, 0.5))
    sample("log_radon", Normal(μ, σ), obs=log_radon)
```

```
...
from numpyro import sample, deterministic
from numpyro.distributions import Normal, InverseGamma

def complete_pooling(floor, log_radon=None):
    α = sample("α", Normal(0, 5))
    β = sample("β", Normal(0, 1))

    μ = deterministic("μ", α + β * floor)
    σ = sample("σ", InverseGamma(1, 0.5))
    sample("log_radon", Normal(μ, σ), obs=log_radon)
```

Some to the likelihood. Notice the **obs** keyword argument.

Can also sample transformations of parameters

```
...
from numpyro import sample, deterministic
from numpyro.distributions import Normal, InverseGamma

def complete_pooling(floor, log_radon=None):
    α = sample("α", Normal(0, 5))
    β = sample("β", Normal(0, 1))

    μ = deterministic("μ", α + β * floor)
    σ = sample("σ", InverseGamma(1, 0.5))
    sample("log_radon", Normal(μ, σ), obs=log_radon)
```

Obtaining the posterior samples it's really easy with Numpyro

```
from numpyro.infer import MCMC, NUTS

settings = dict(num_samples=2000, num_warmup=2000,
num_chains=4)
mcmc = MCMC(NUTS(complete_pooling), **settings)
data = dict(
    floor=[-1, -1, 0, -1, 0, -1, ...],
    log_radon=[0.78, 1.06, -0.35, 0.40, ...],
)
mcmc.run(seed, **data)
samples = mcmc.get_samples()
```

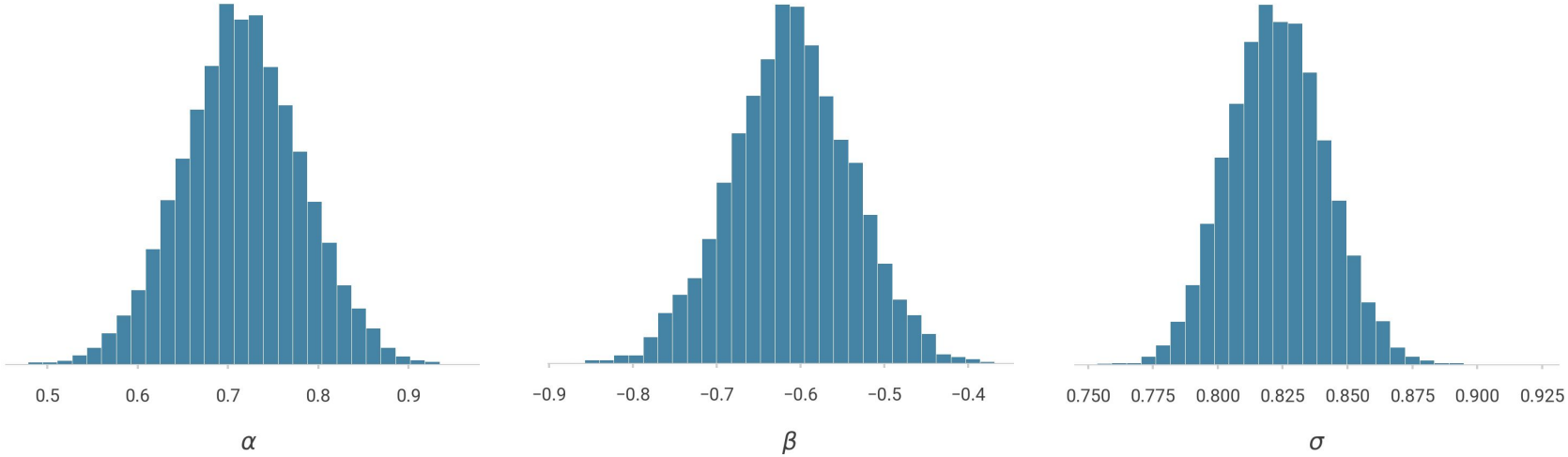
Construct your MCMC instance

```
from numpyro.infer import MCMC, NUTS

settings = dict(num_samples=2000, num_warmup=2000,
num_chains=4)
mcmc = MCMC(NUTS(complete_pooling), **settings)
data = dict(
    floor=[-1, -1, 0, -1, 0, -1, ...],
    log_radon=[0.78, 1.06, -0.35, 0.40, ...],
)
mcmc.run(seed, **data)
samples = mcmc.get_samples()
```

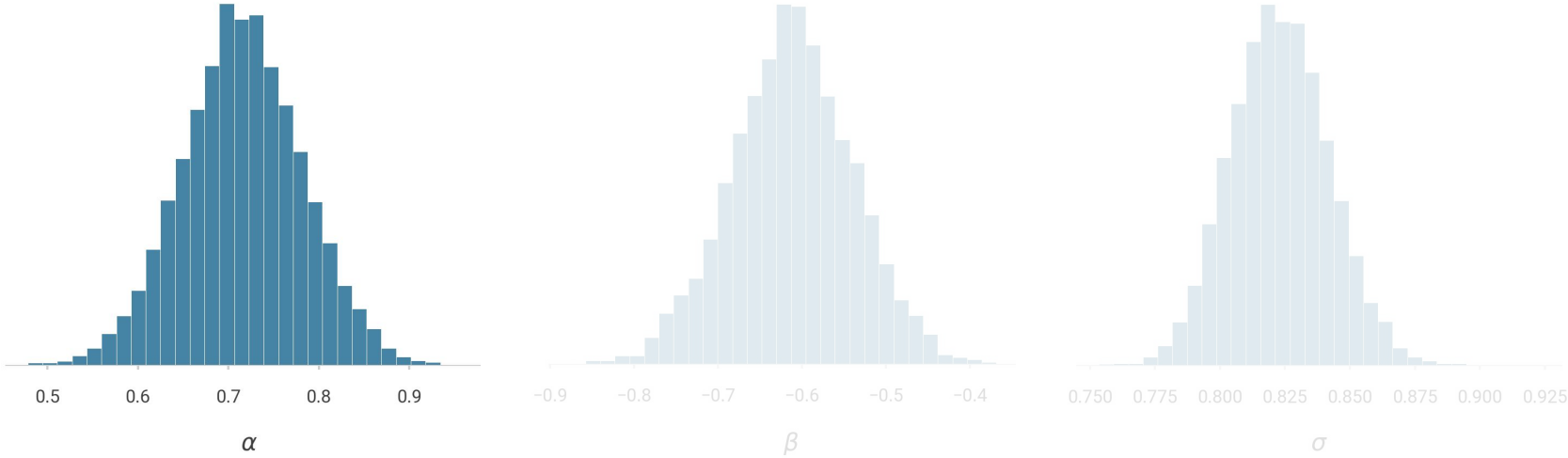
```
from numpyro.infer import MCMC, NUTS

settings = dict(num_samples=2000, num_warmup=2000,
num_chains=4)
mcmc = MCMC(NUTS(complete_pooling), **settings)
data = dict(
    floor=[-1, -1, 0, -1, 0, -1, ...],
    log_radon=[0.78, 1.06, -0.35, 0.40, ...],
)
mcmc.run(seed, **data)
samples = mcmc.get_samples()
```

```
from numpyro.infer import MCMC, NUTS

settings = dict(num_samples=2000, num_warmup=2000,
num_chains=4)
mcmc = MCMC(NUTS(complete_pooling), **settings)
data = dict(
    floor=[-1, -1, 0, -1, 0, -1, ...],
    log_radon=[0.78, 1.06, -0.35, 0.40, ...],
)
mcmc.run(seed, **data)
samples = mcmc.get_samples()
```

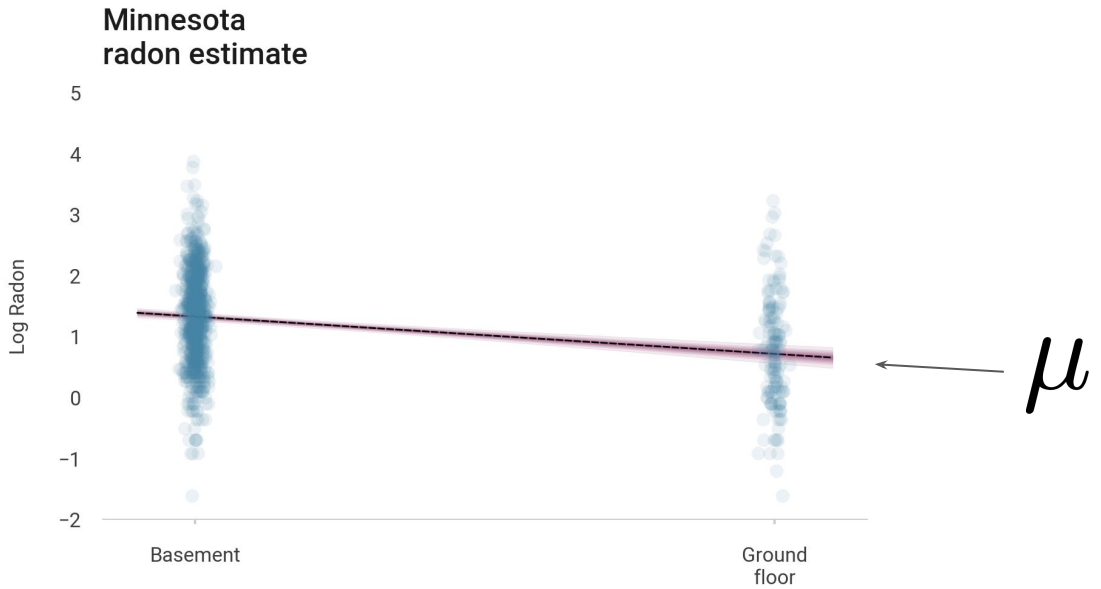
Running MCMC provides the posterior samples for each parameter specified in the model.
The samples can the be used to provide probabilistic predictions.



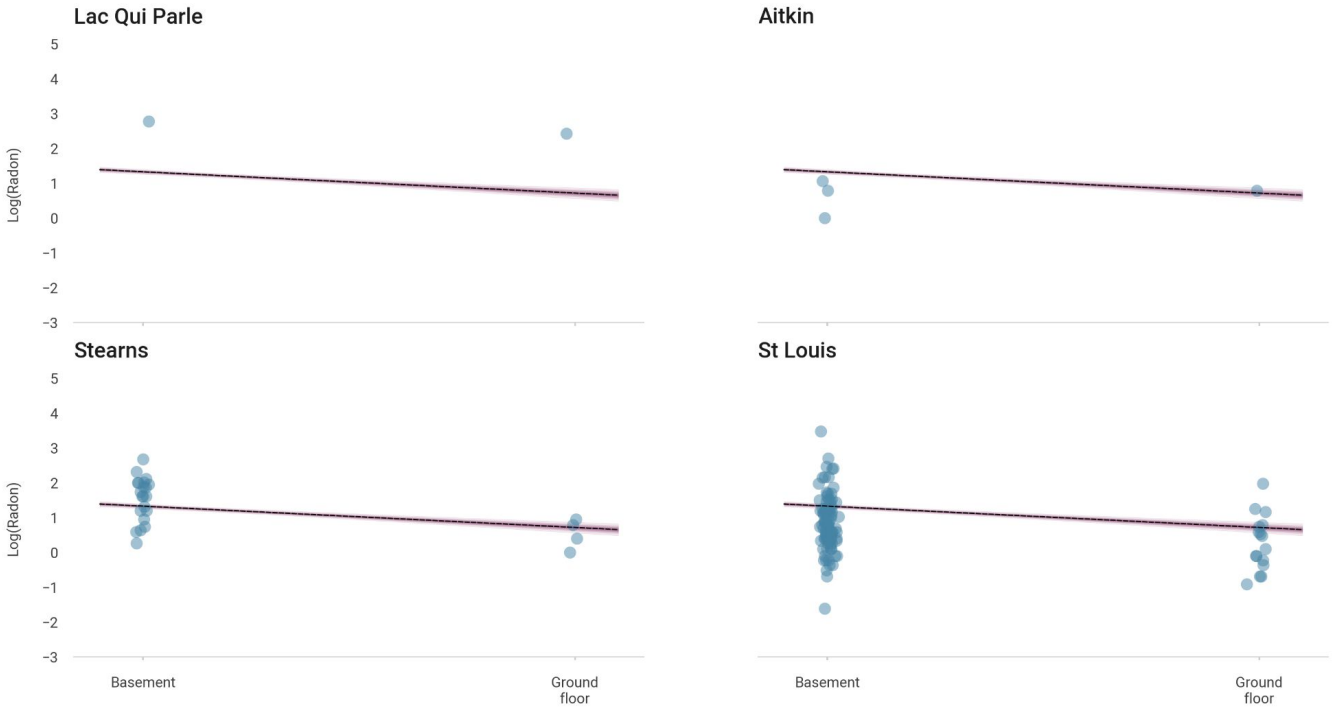
Running MCMC provides the posterior samples for each parameter specified in the model.
The samples can the be used to provide probabilistic predictions.



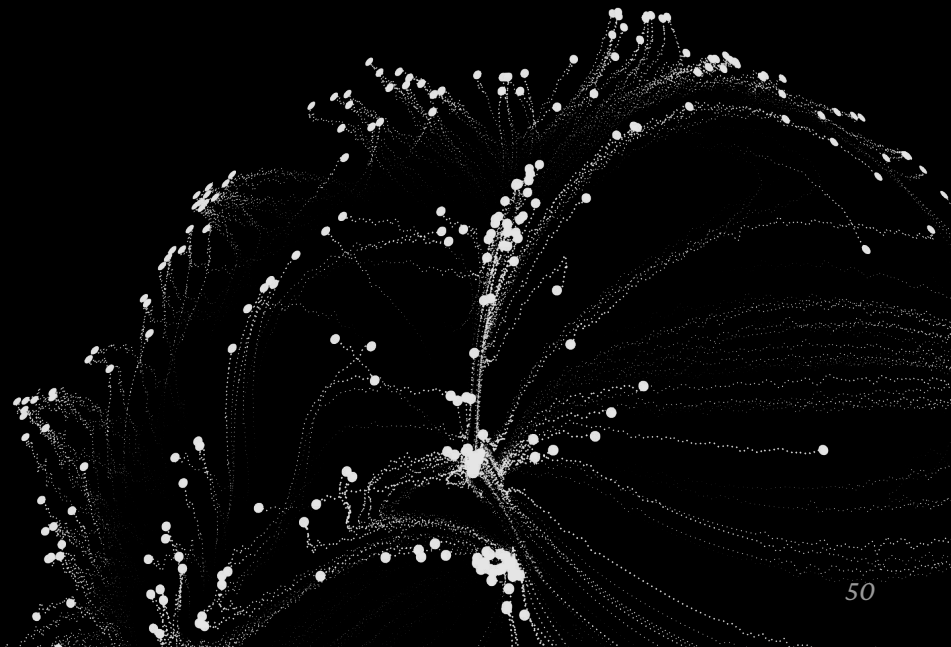
The complete pooling model does a good job at capturing the overall radon levels in Minnesota.



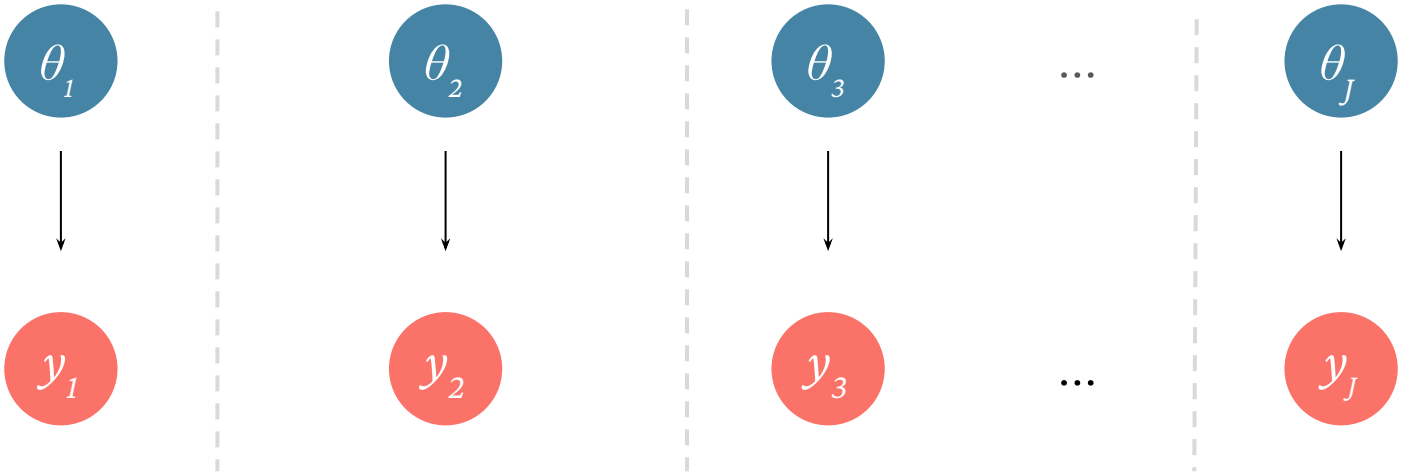
The complete pooling model does a good job at capturing the overall radon levels in Minnesota.
But “county level” estimates can be virtually useless.



03
Traditional approaches

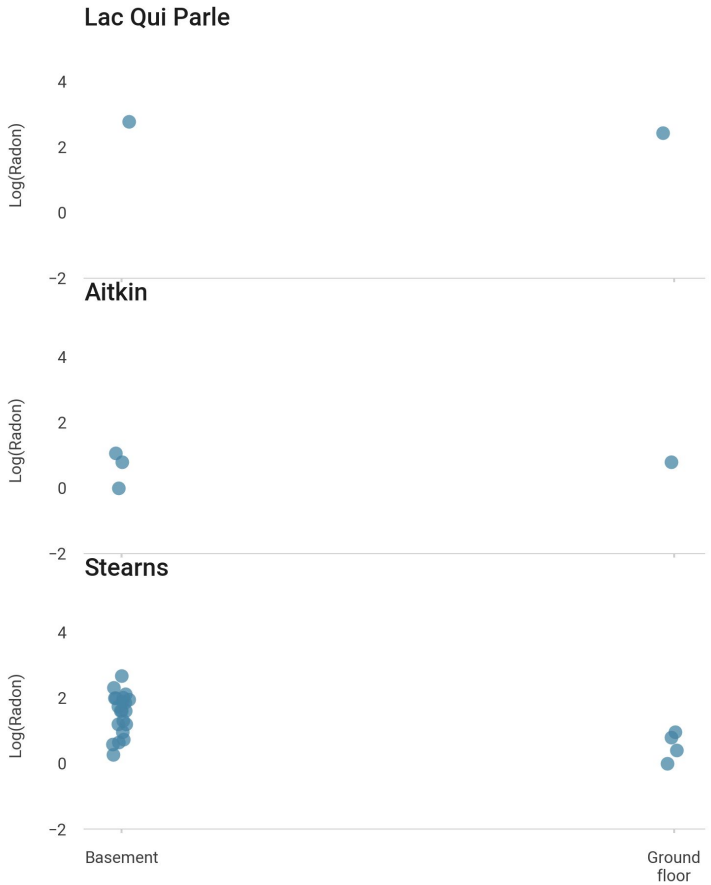


No pooling treats each group independently. Fitting a different model for each group.



For $j = 1, 2, \ldots, J$:

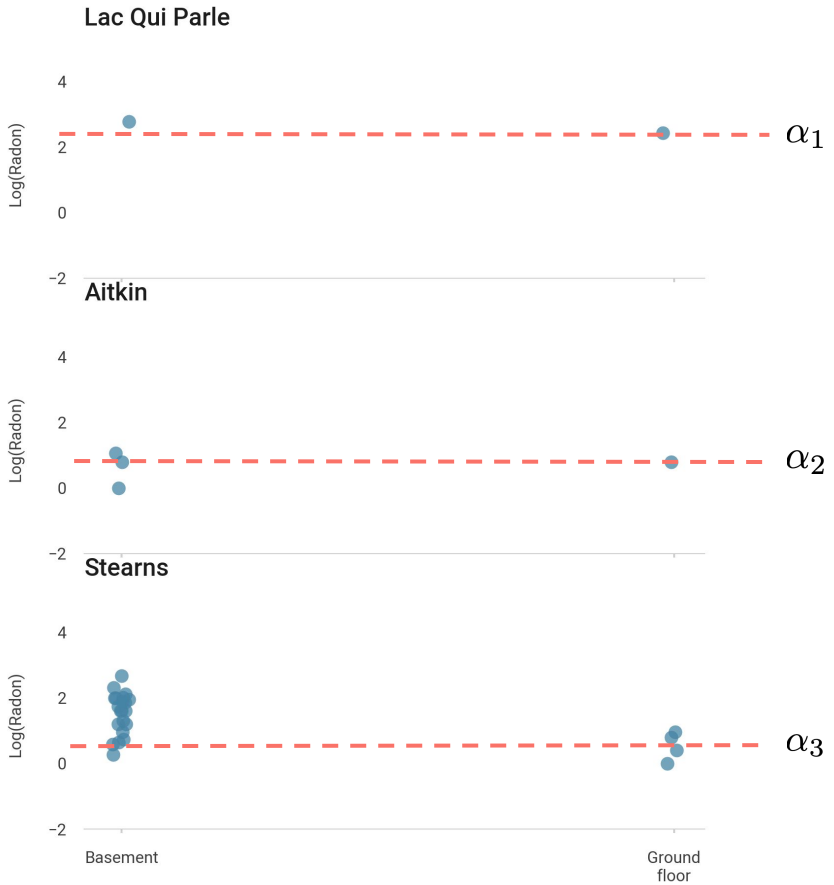
$$y_j \sim \text{normal}(\alpha_j + \beta_j \cdot \text{floor}, \tau)$$



For $j = 1, 2, \ldots, J$:

$$y_j \sim \text{normal}(\alpha_j + \beta_j \cdot \text{floor}, \tau)$$

α_j to captures *each county's* level at the ground floor.

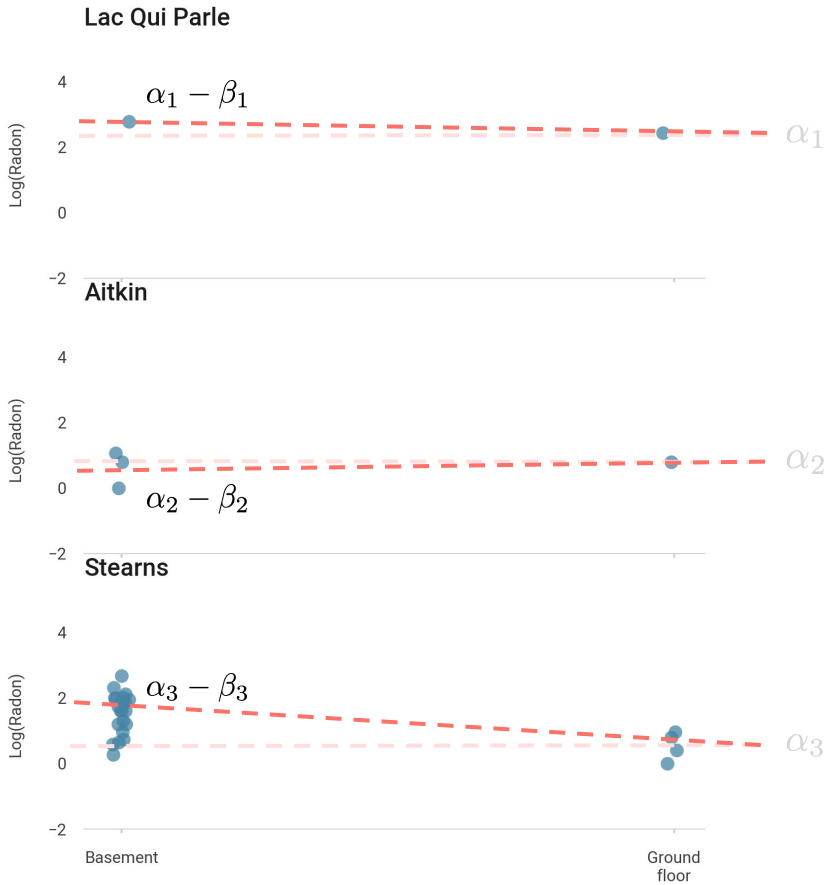


For $j = 1, 2, \ldots, J$:

$$y_j \sim \text{normal}(\alpha_j + \beta_j \cdot \text{floor}, \tau)$$

α_j to captures *each county's* level at the ground floor.

β_j is *each county's* increase from basement to ground floor.



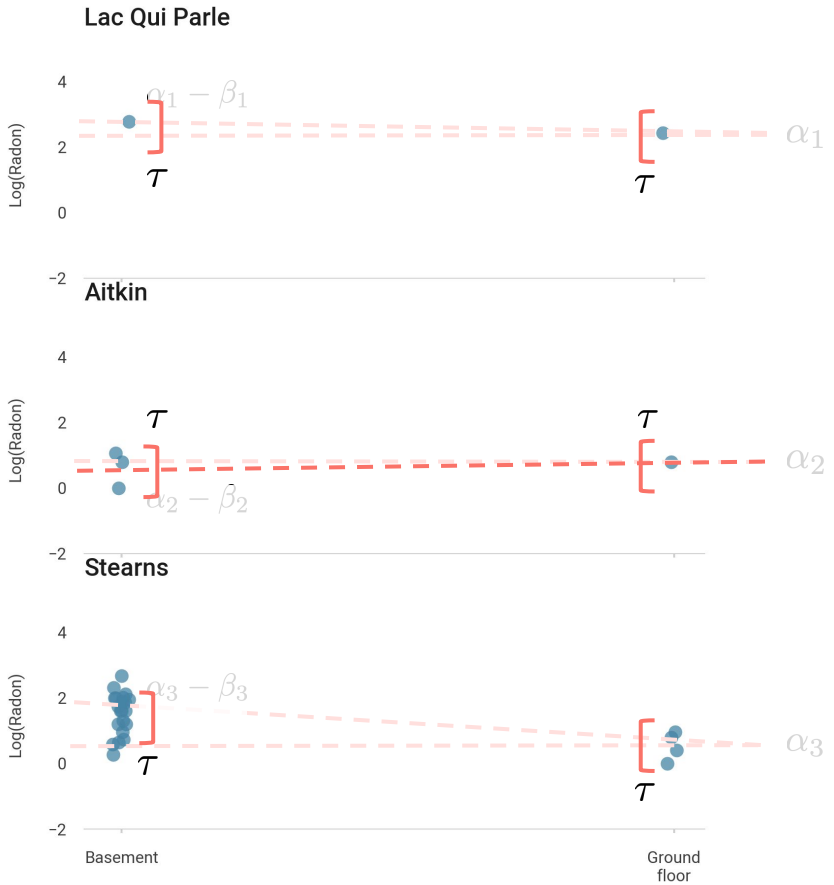
For $j = 1, 2, \ldots, J$:

$$y_j \sim \text{normal}(\alpha_j + \beta_j \cdot \text{floor}, \tau)$$

α_j to captures *each county's* level at the ground floor.

β_j is *each county's* increase from basement to ground floor.

τ measures the average *within-county* variation only.



For $j = 1, 2, \ldots, J$:

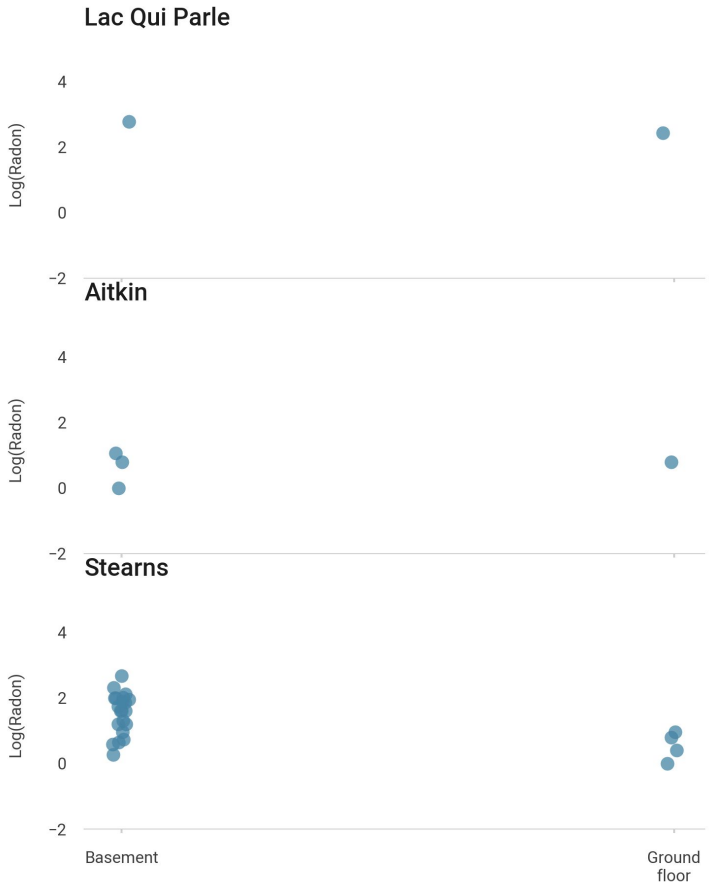
$$y_j \sim \text{normal}(\alpha_j + \beta_j \cdot \text{floor}, \tau)$$
$$\alpha_j \sim \text{normal}(0, 5)$$
$$\beta_j \sim \text{normal}(0, 1)$$

$$\frac{1}{\tau} \sim \text{gamma}(1, 0.5)$$

α_j to captures *each county's* level at the ground floor.

β_j is *each county's* increase from basement to ground floor.

τ measures the average *within-county* variation only.



```
...
from numpyro import plate

def no_pooling(county, floor, log_radon=None):
    # N_COUNTIES was defined in global scope
    with plate("counties", N_COUNTIES):
        α = sample("α", Normal(0, 5))
        β = sample("β", Normal(0, 1))

    μ = deterministic("μ", α[county] + β[county] * floor)
    τ = sample("τ", InverseGamma(1, 0.5))
    sample("log_radon", Normal(μ, τ), obs=log_radon)
```

We must now also provide the **county** as input

```
...
from numpyro import plate

def no_pooling(county, floor, log_radon=None):
    # N_COUNTIES was defined in global scope
    with plate("counties", N_COUNTIES):
        α = sample("α", Normal(0, 5))
        β = sample("β", Normal(0, 1))

    μ = deterministic("μ", α[county] + β[county] * floor)
    τ = sample("τ", InverseGamma(1, 0.5))
    sample("log_radon", Normal(μ, τ), obs=log_radon)
```

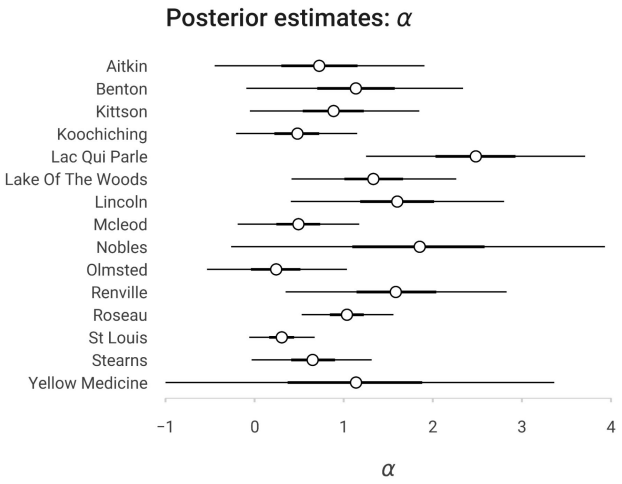
Use a *plate* for automatic **broadcasting**

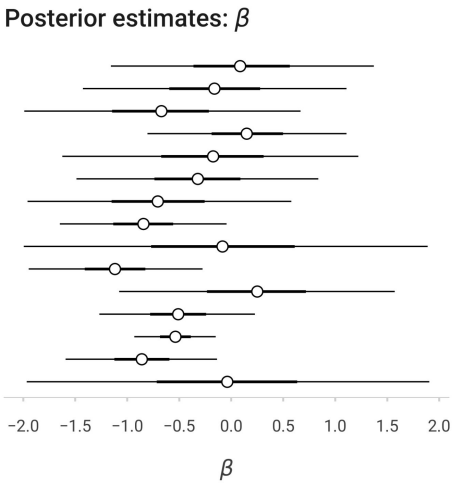
```
...
from numpyro import plate

def no_pooling(county, floor, log_radon=None):
    # N_COUNTIES was defined in global scope
    with plate("counties", N_COUNTIES):
        α = sample("α", Normal(0, 5))
        β = sample("β", Normal(0, 1))

    μ = deterministic("μ", α[county] + β[county] * floor)
    τ = sample("τ", InverseGamma(1, 0.5))
    sample("log_radon", Normal(μ, τ), obs=log_radon)
```

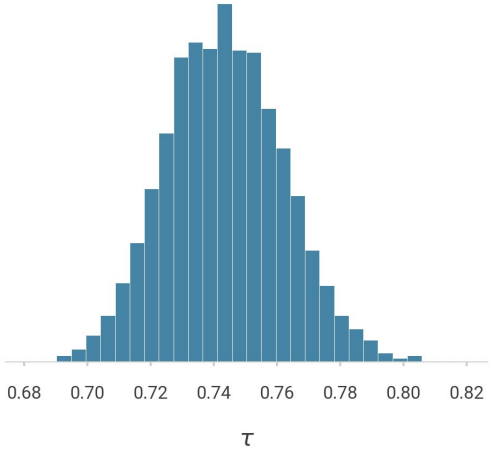
```
...
from numpyro import plate

def no_pooling(county, floor, log_radon=None):
    # N_COUNTIES was defined in global scope
    with plate("counties", N_COUNTIES):
        α = sample("α", Normal(0, 5))
        β = sample("β", Normal(0, 1))

    μ = deterministic("μ", α[county] + β[county] * floor)
    τ = sample("τ", InverseGamma(1, 0.5))
    sample("log_radon", Normal(μ, τ), obs=log_radon)
```

Each observation is modelled with its corresponding parameters

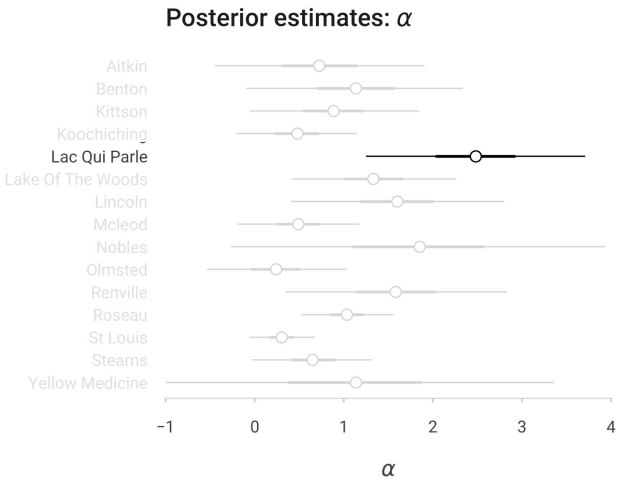
We obtain separate estimates for each county, although in many cases the posterior is very wide. Plots below are only showing 15 selected counties among the 80+ possible ones.

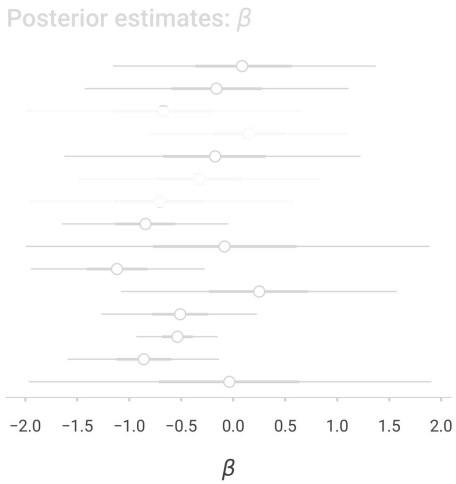


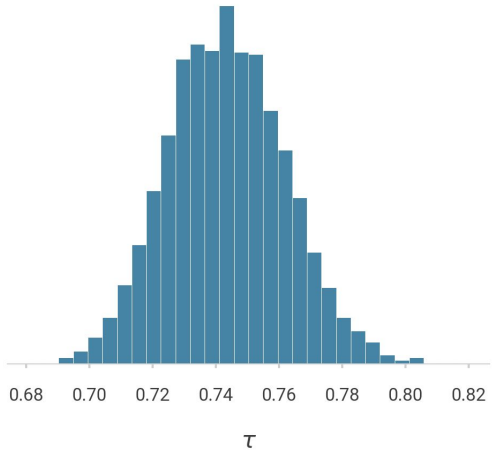




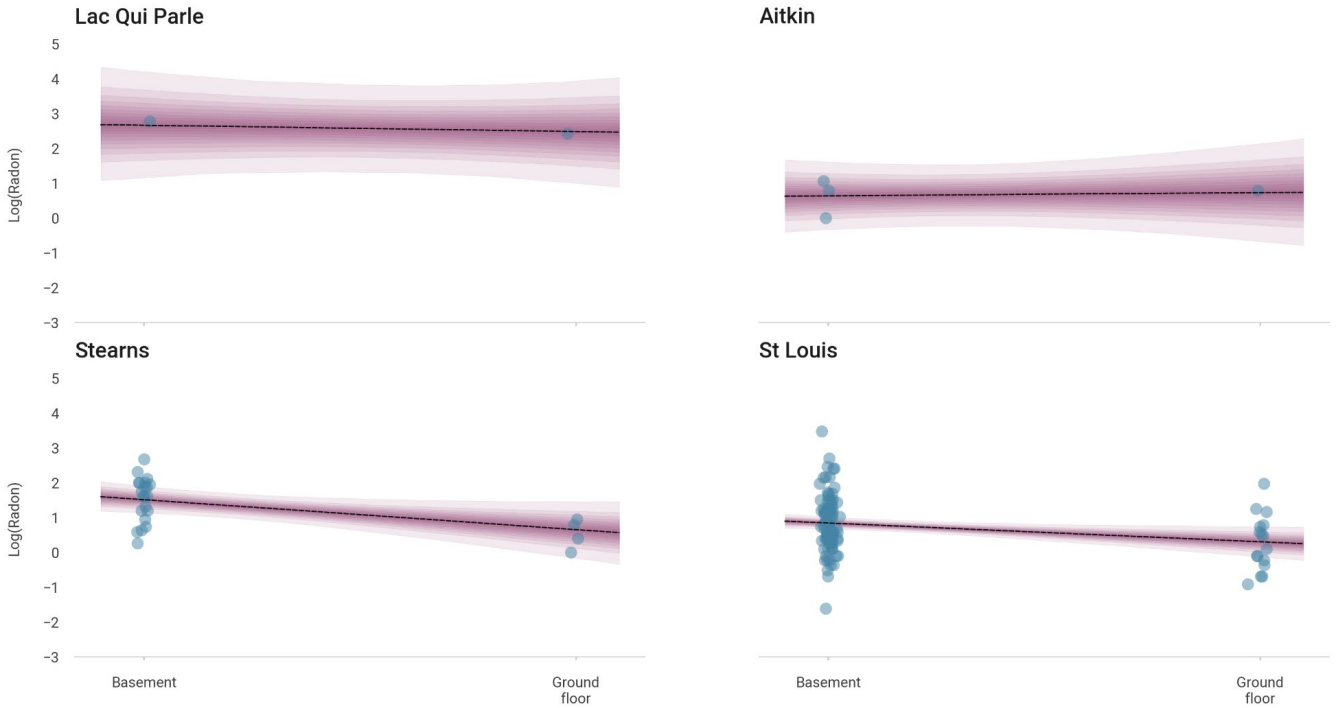
We obtain separate estimates for each county, although in many cases the posterior is very wide. Plots below are only showing 15 selected counties among the 80+ possible ones.





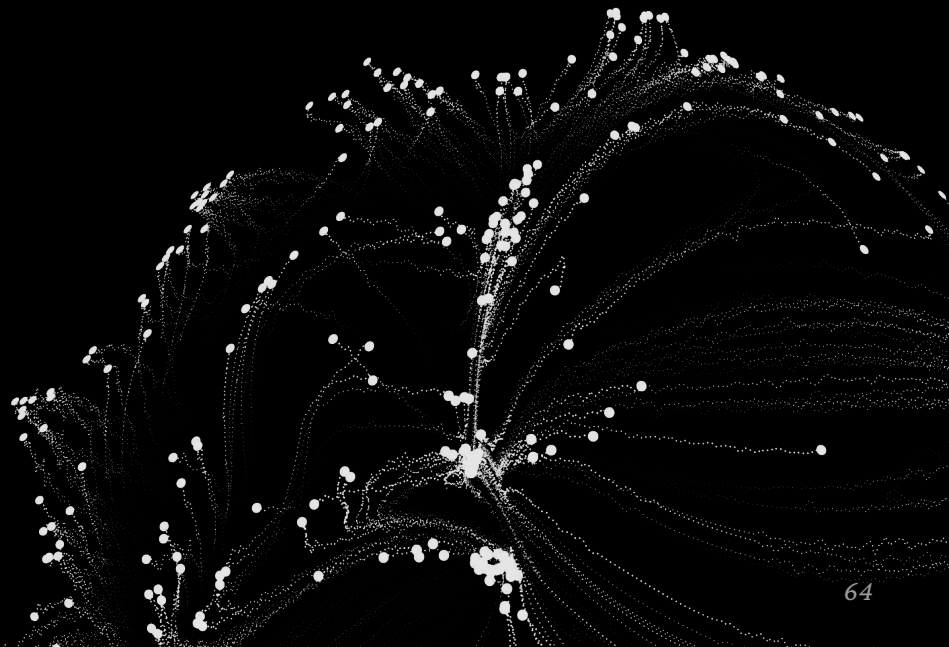


We are able to make bespoke predictions, but counties with uninformative data end up with very uncertain and/or extreme estimates.



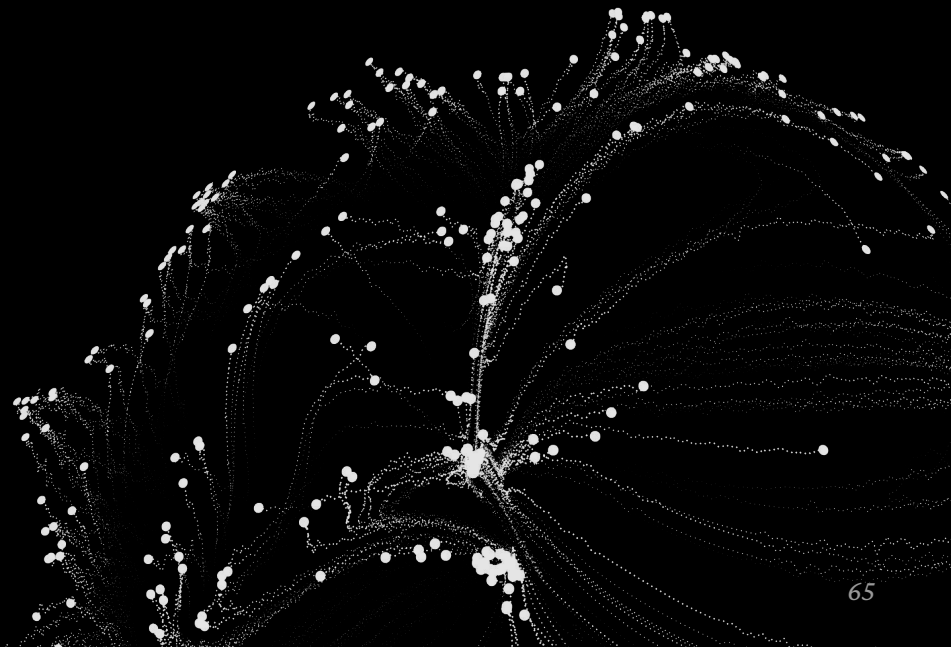
04

Hierarchical modelling



faculty

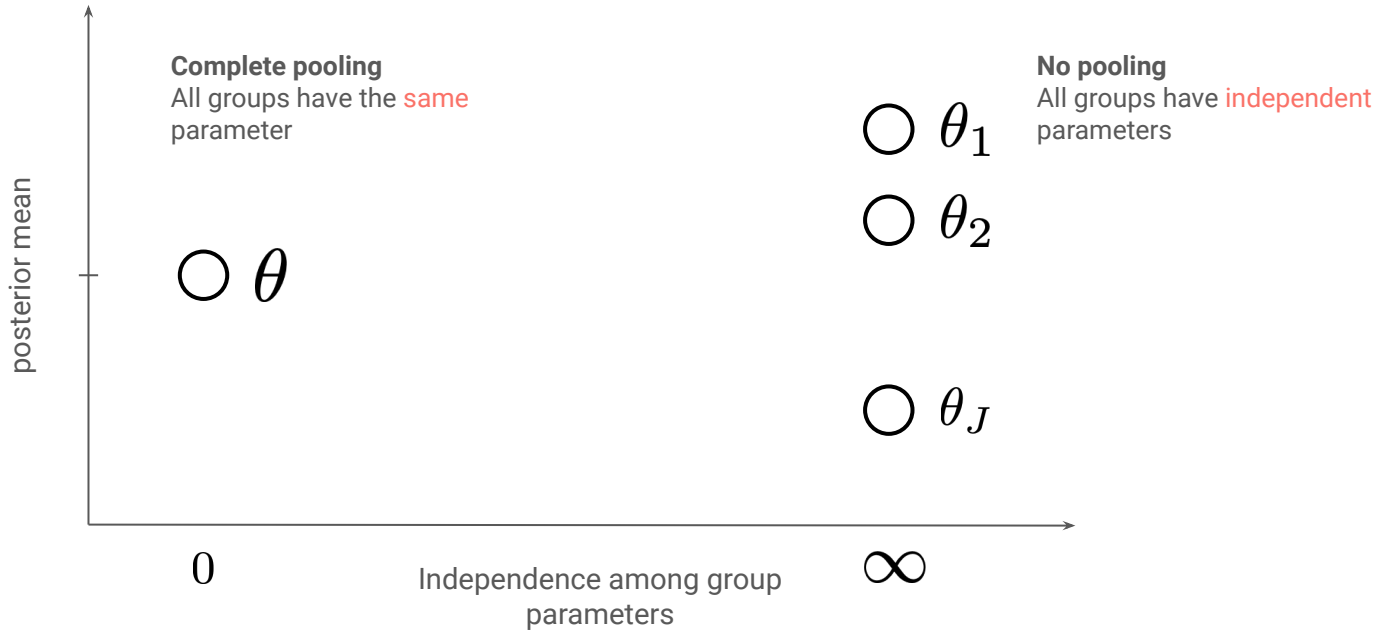
04
Hierarchical modelling



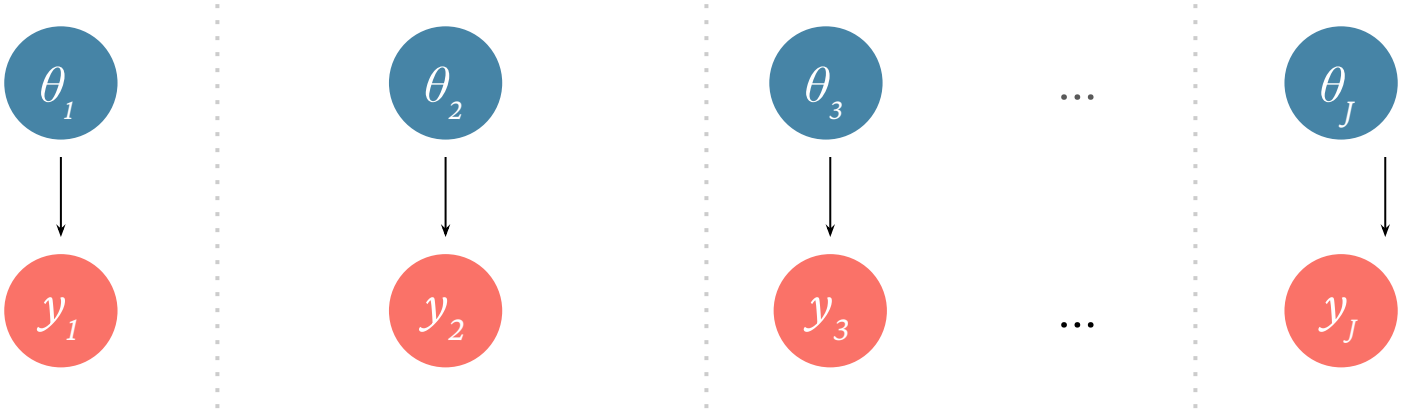
faculty

We can understand No pooling and Complete Pooling approaches as the 2 extremes of a continuous spectrum of modelling approaches:

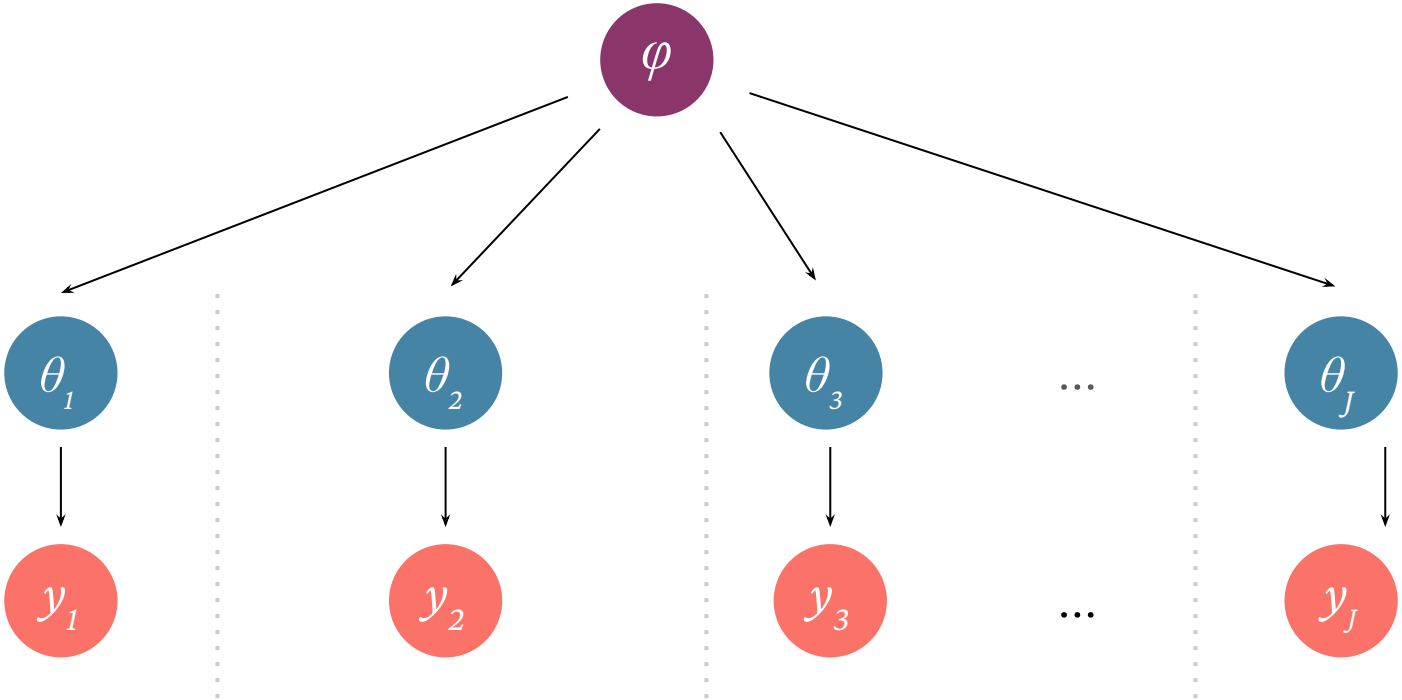
We can understand No pooling and Complete Pooling approaches as the 2 extremes of a continuous spectrum of modelling approaches:



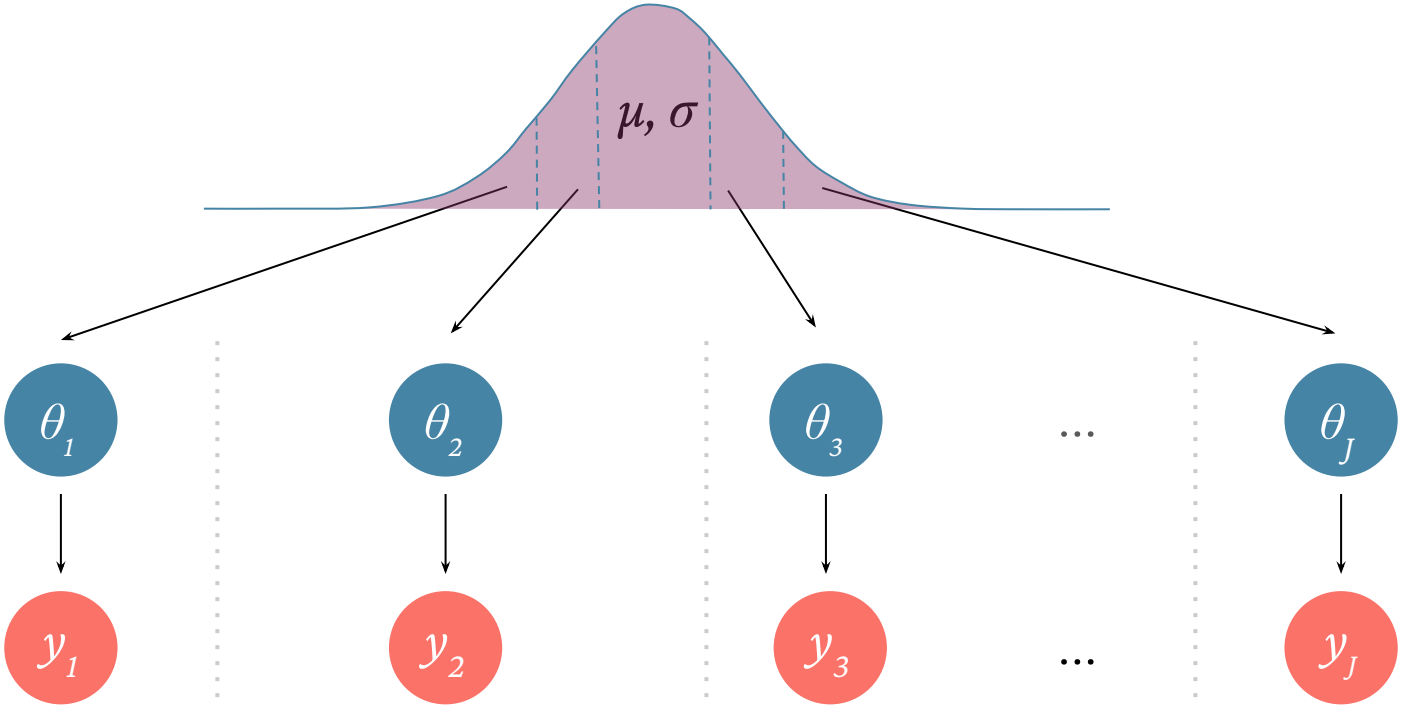
Mathematically, we obtain partial pooling by introducing a latent model that describes the distribution of the group parameters.



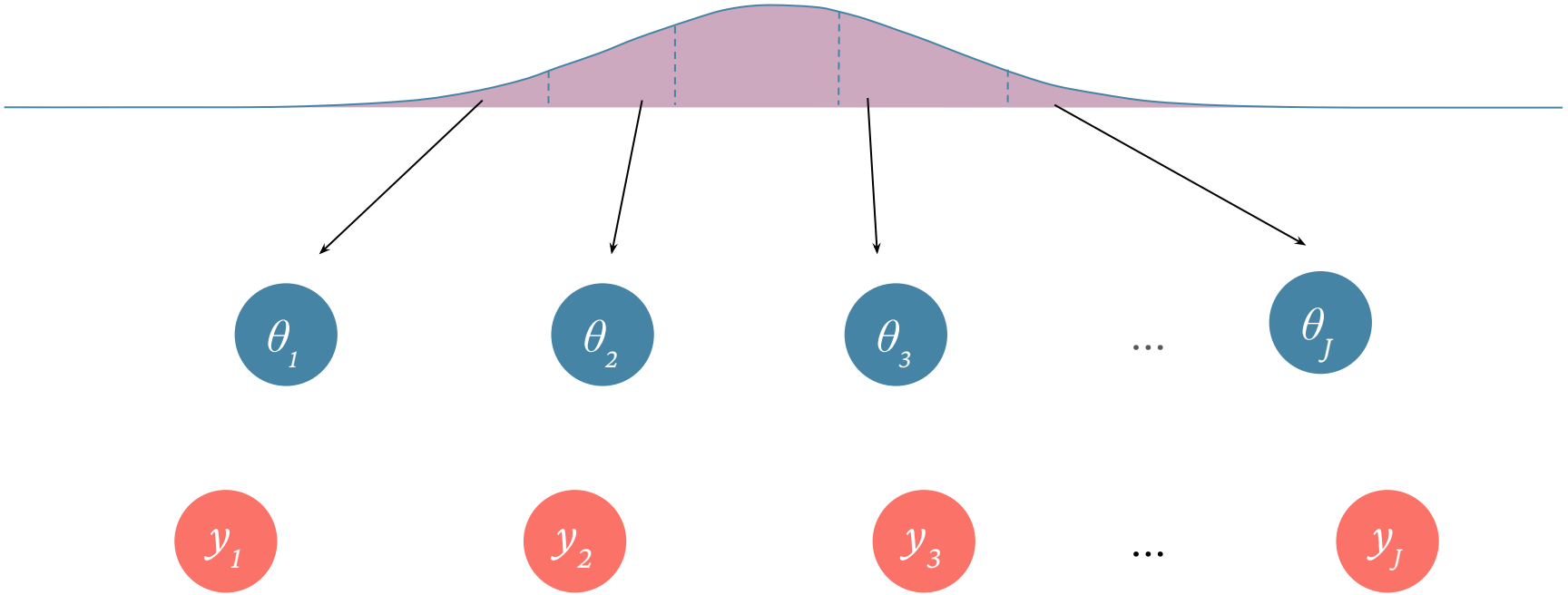
Mathematically, we obtain partial pooling by introducing a latent model that describes the distribution of the group parameters.



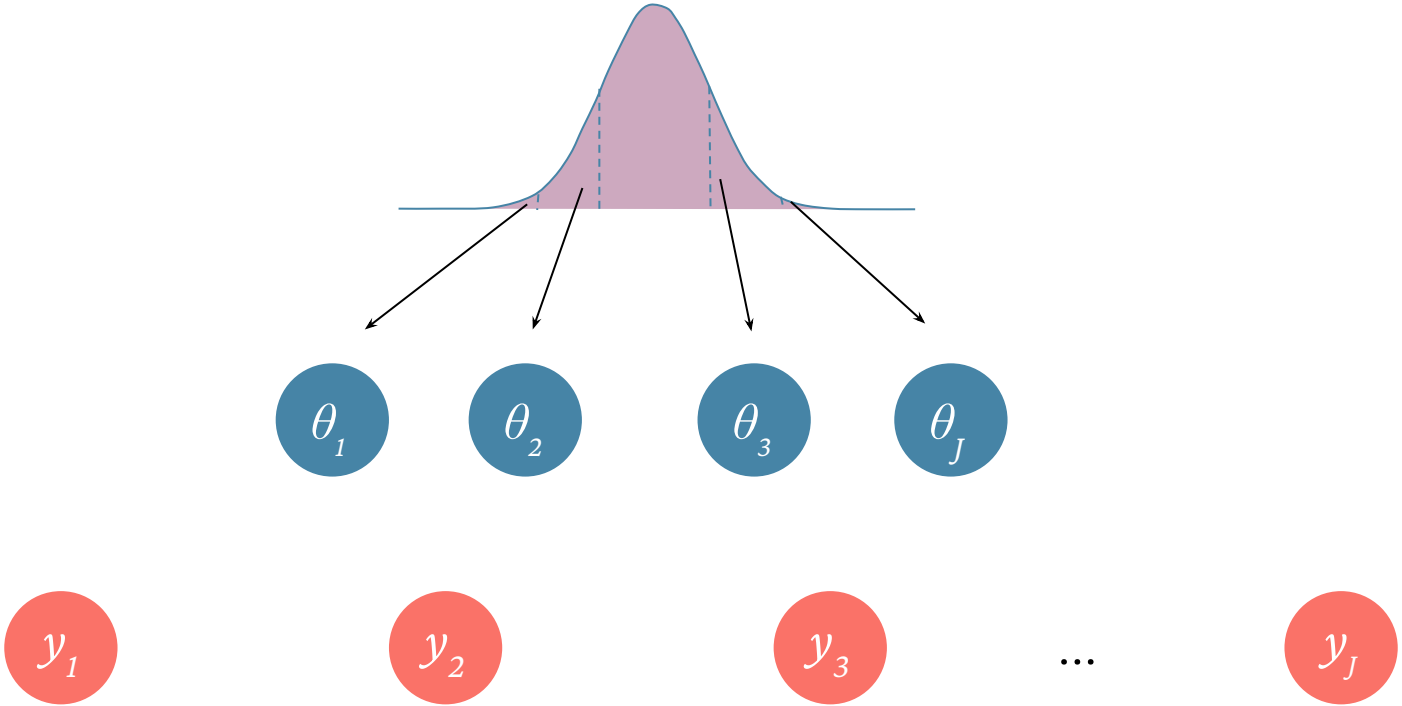
A popular choice is to use a latent Gaussian model, parametrized in terms of its mean and variance.



A high latent variance, imposes weak correlations and allows the parameter to vary independently.



A small variance, imposes strong correlations and forces the parameters to be similar



α_j captures *each* level at the ground floor.

β_j is *each* increase from basement to ground floor.

τ measures the *within*-county variation only.

μ_α captures the *average* level at the ground floor in **Minnesota**

σ_α *controls* how similar the group countly-level α's are

For j = 1, 2, ... , J:

$$y_j \sim \text{normal}(\alpha_j + \beta_j \cdot \text{floor}, \tau)$$
$$\alpha_j \sim \text{normal}(\mu_\alpha, \sigma_\alpha)$$
$$\beta_j \sim \text{normal}(0, 1)$$

$$\frac{1}{\tau} \sim \text{gamma}(1, 0.5)$$
$$\mu_\alpha \sim \text{normal}(0, 5)$$
$$\sigma_\alpha \in (0, \infty)$$

α_j captures *each* level at the ground floor.

β_j is *each* increase from basement to ground floor.

τ measures the *within*-county variation only.

μ_α captures the *average* level at the ground floor in **Minnesota**

σ_α *controls* how similar the group county-level α's are

For $j = 1, 2, \ldots, J$:

$$y_j \sim \text{normal}(\alpha_j + \beta_j \cdot \text{floor}, \tau)$$

$$\alpha_j \sim \text{normal}(\mu_\alpha, \sigma_\alpha)$$

$$\beta_j \sim \text{normal}(0, 1)$$

$$\frac{1}{\tau} \sim \text{gamma}(1, 0.5)$$

$$\mu_\alpha \sim \text{normal}(0, 5)$$

$$\sigma_\alpha \in (0, \infty)$$

α_j captures *each* level at the ground floor.

β_j is *each* increase from basement to ground floor.

τ measures the *within*-county variation only.

μ_α captures the *average* level at the ground floor in **Minnesota**

σ_α *controls* how similar the group countly-level α's are

For j = 1, 2, ... , J:

$$y_j \sim \text{normal}(\alpha_j + \beta_j \cdot \text{floor}, \tau)$$
$$\alpha_j \sim \text{normal}(\mu_\alpha, \sigma_\alpha)$$
$$\beta_j \sim \text{normal}(0, 1)$$

$$\frac{1}{\tau} \sim \text{gamma}(1, 0.5)$$
$$\mu_\alpha \sim \text{normal}(0, 5)$$
$$\sigma_\alpha \in (0, \infty)$$

α_j captures *each* level at the ground floor.

β_j is *each* increase from basement to ground floor.

τ measures the *within*-county variation only.

μ_α captures the *average* level at the ground floor in **Minnesota**

σ_α *controls* how similar the group countly-level α's are

For j = 1, 2, ... , J:

$$y_j \sim \text{normal}(\alpha_j + \beta_j \cdot \text{floor}, \tau)$$
$$\alpha_j \sim \text{normal}(\mu_\alpha, \sigma_\alpha)$$
$$\beta_j \sim \text{normal}(0, 1)$$

$$\frac{1}{\tau} \sim \text{gamma}(1, 0.5)$$
$$\mu_\alpha \sim \text{normal}(0, 5)$$
$$\sigma_\alpha \in (0, \infty)$$

```
...

def partial_pooling(σ_α, county, floor, log_radon=None):

    μ_α = sample("μ_α", Normal(0, 5))

    with plate("counties", N_COUNTIES):
        α = sample("α", Normal(μ_α, σ_α))
        β = sample("β", Normal(0, 1))

    μ = deterministic("μ", α[county] + β[county] * floor)
    τ = sample("τ", InverseGamma(1, 0.5))
    sample("log_radon", Normal(μ, τ), obs=log_radon)

...
```

The amount of **pooling** required, σ_{α}, will be provided alongside the data

```
...

def partial_pooling(σ_α, county, floor, log_radon=None):

    μ_α = sample("μ_α", Normal(0, 5))

    with plate("counties", N_COUNTIES):
        α = sample("α", Normal(μ_α, σ_α))
        β = sample("β", Normal(0, 1))

    μ = deterministic("μ", α[county] + β[county] * floor)
    τ = sample("τ", InverseGamma(1, 0.5))
    sample("log_radon", Normal(μ, τ), obs=log_radon)


...
```

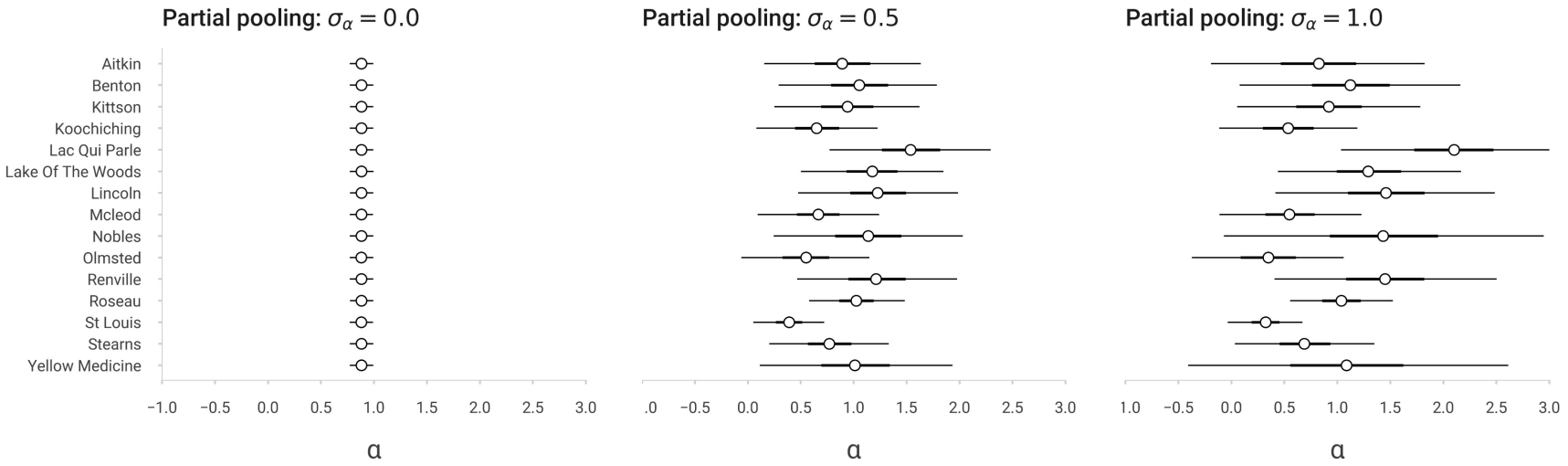
The intercepts are given a model instead of just a prior

```
...

def partial_pooling(σ_α, county, floor, log_radon=None):

    μ_α = sample("μ_α", Normal(0, 5))

    with plate("counties", N_COUNTIES):
        α = sample("α", Normal(μ_α, σ_α))
        β = sample("β", Normal(0, 1))

    μ = deterministic("μ", α[county] + β[county] * floor)
    τ = sample("τ", InverseGamma(1, 0.5))
    sample("log_radon", Normal(μ, τ), obs=log_radon)

...
```

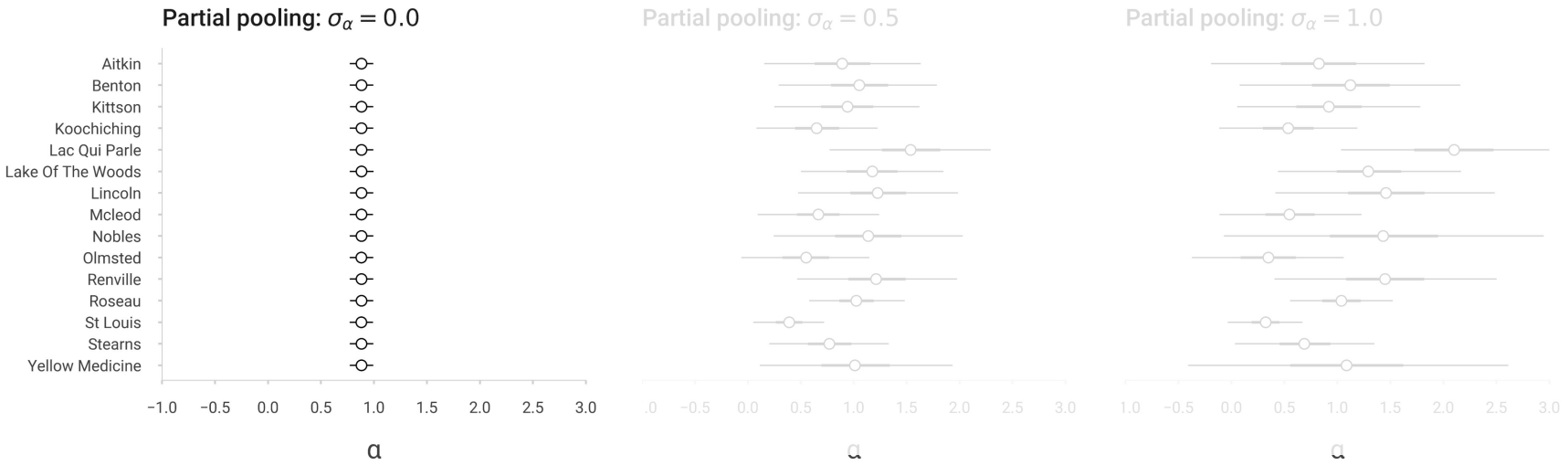
We provide **hyper-priors** for the parameters of the latent model

```
...

def partial_pooling(σ_α, county, floor, log_radon=None):

    μ_α = sample("μ_α", Normal(0, 5))

    with plate("counties", N_COUNTIES):
        α = sample("α", Normal(μ_α, σ_α))
        β = sample("β", Normal(0, 1))

    μ = deterministic("μ", α[county] + β[county] * floor)
    τ = sample("τ", InverseGamma(1, 0.5))
    sample("log_radon", Normal(μ, τ), obs=log_radon)

...
```

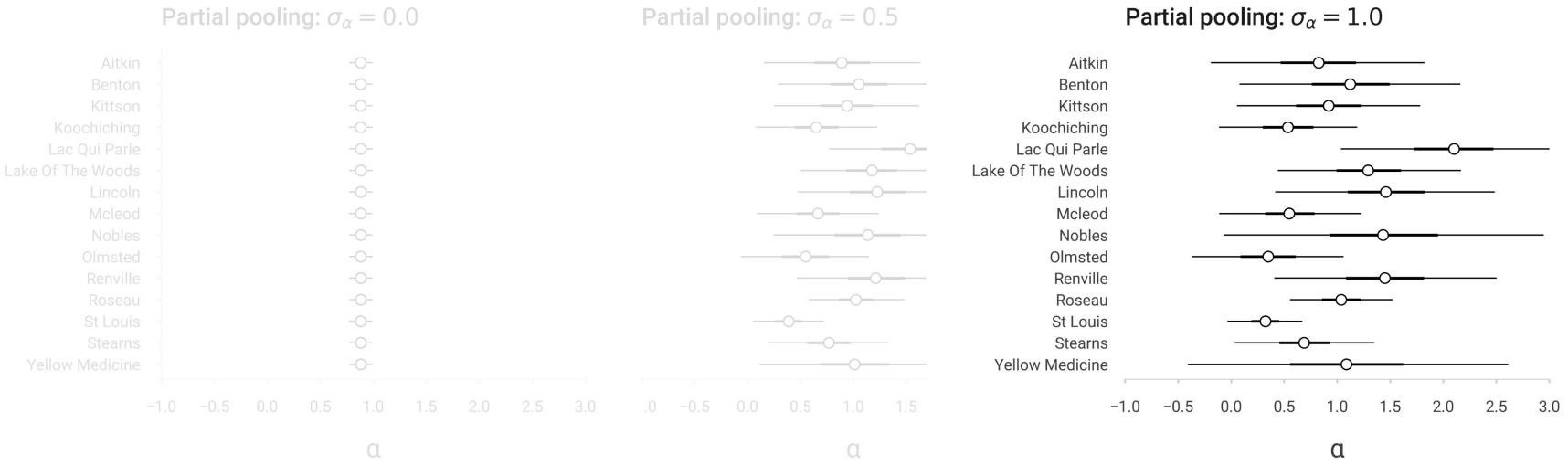
We can verify that the partial pooling can recover the complete pooling and no pooling extremes.



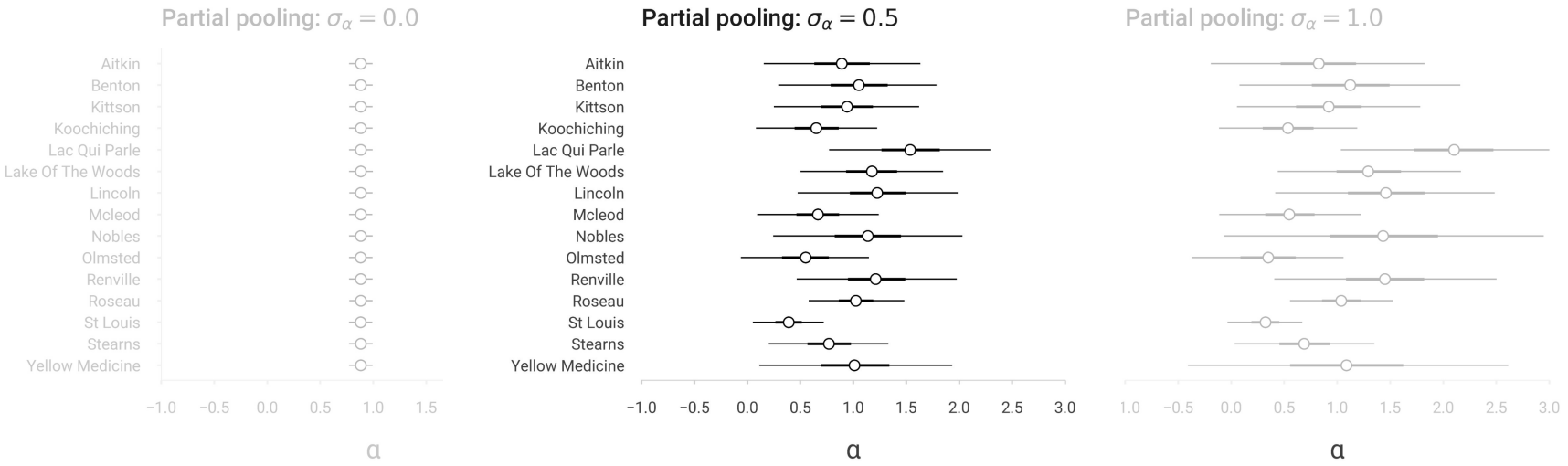
At zero allowed variation, all group estimates have to be the same.



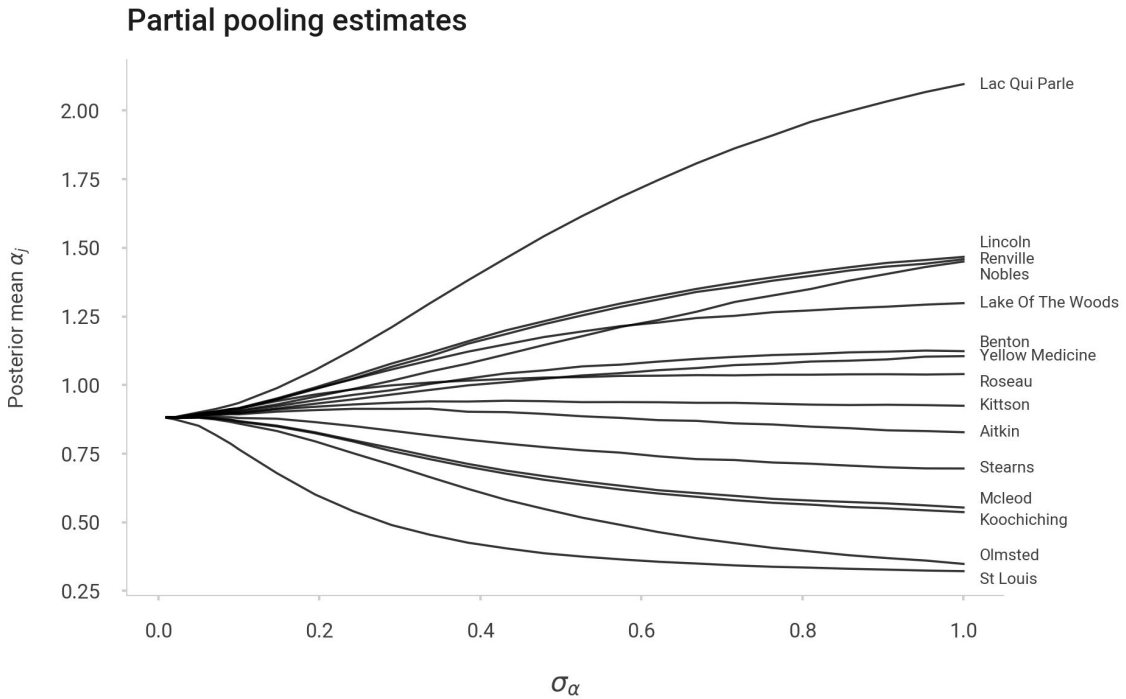
At “infinite” allowed variation, all group estimates can be completely uncorrelated.



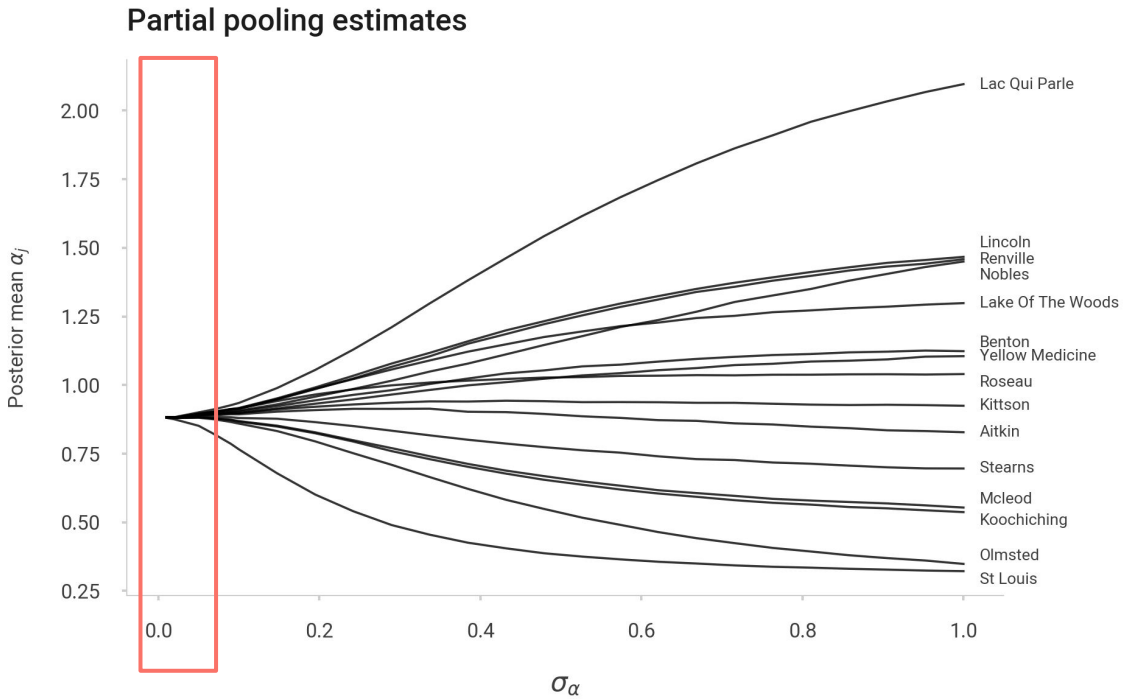
Other values of σ_α result in a middle ground between the two extremes.



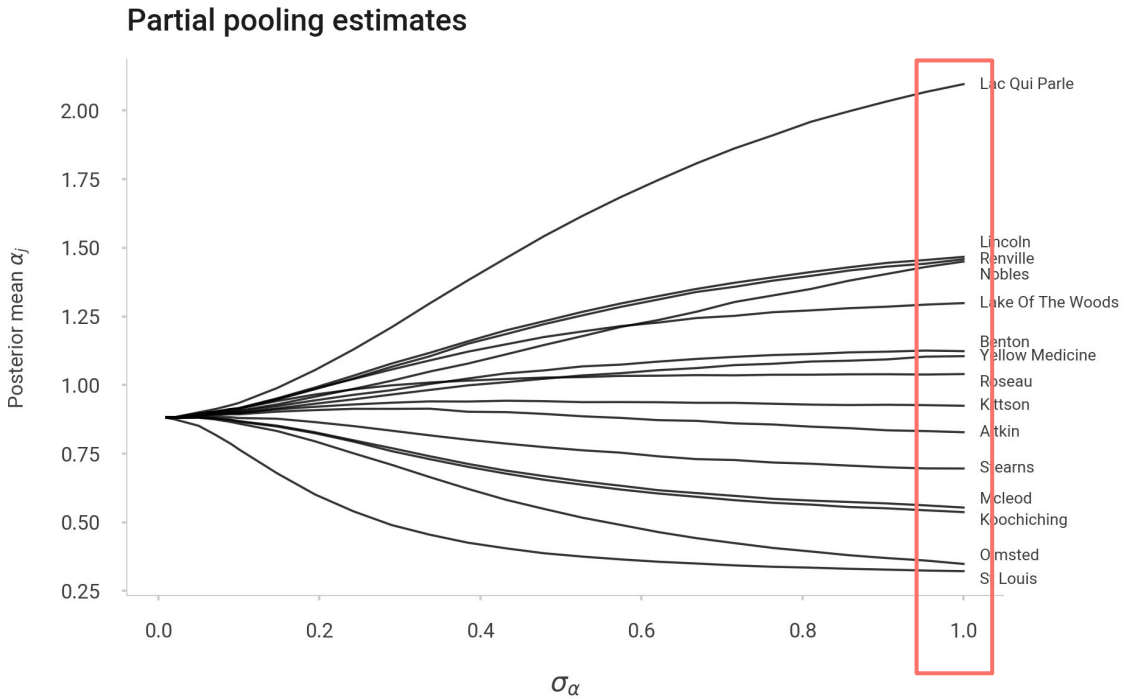
We can visualize the full spectrum of models.



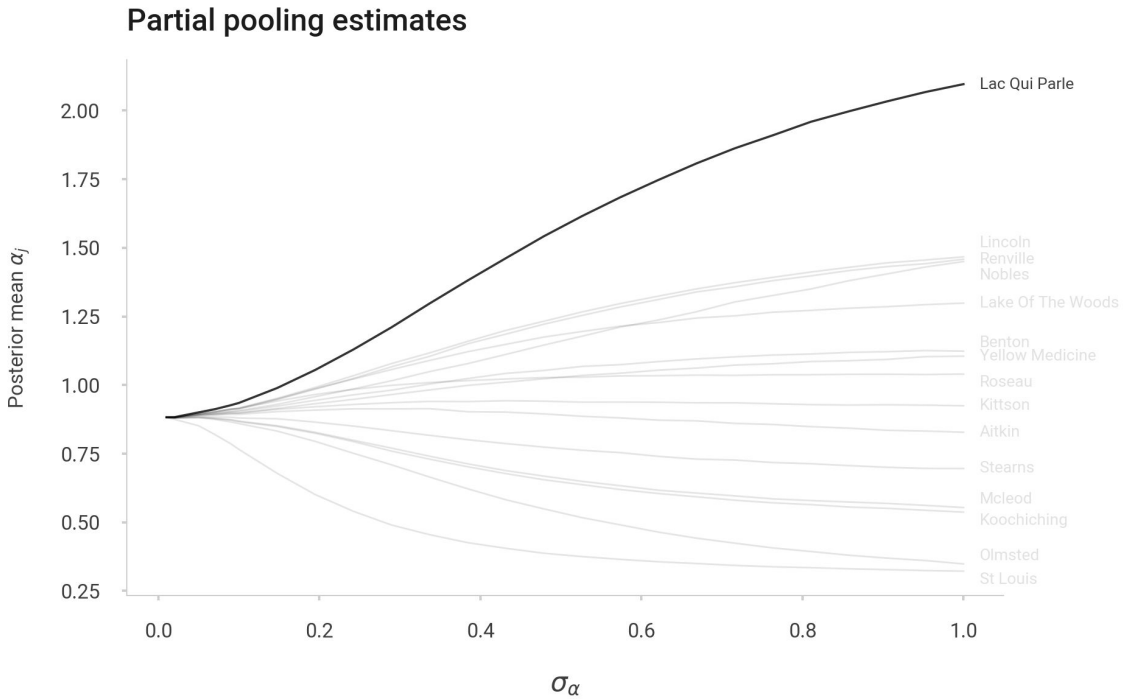
From complete pooling



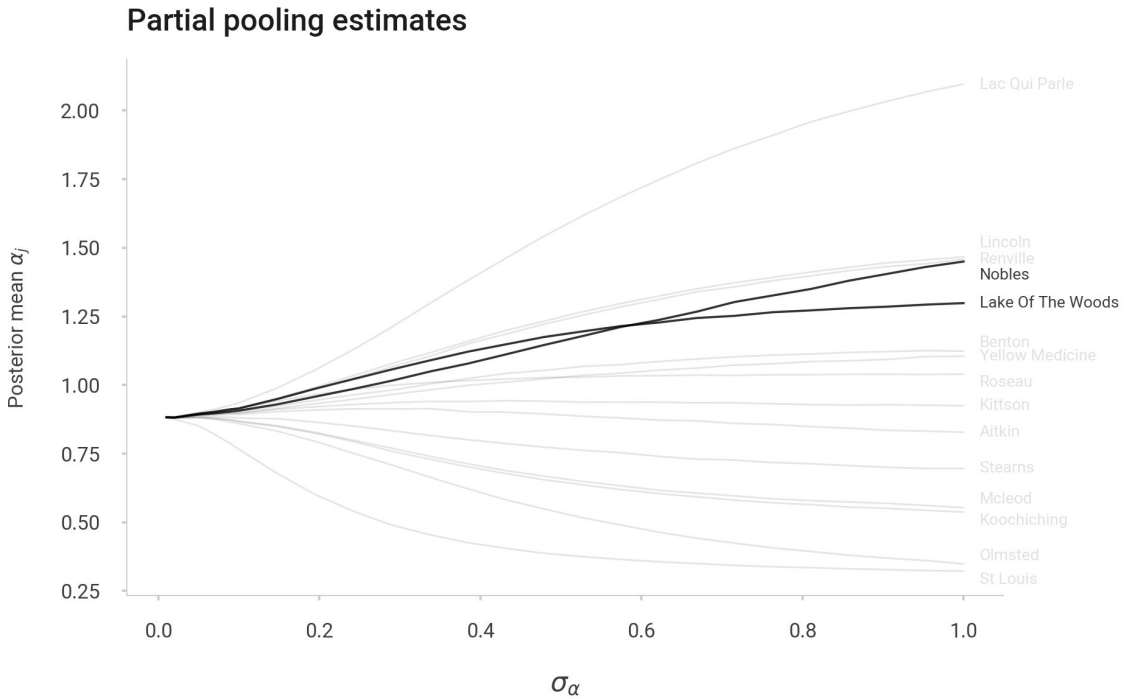
To no pooling.



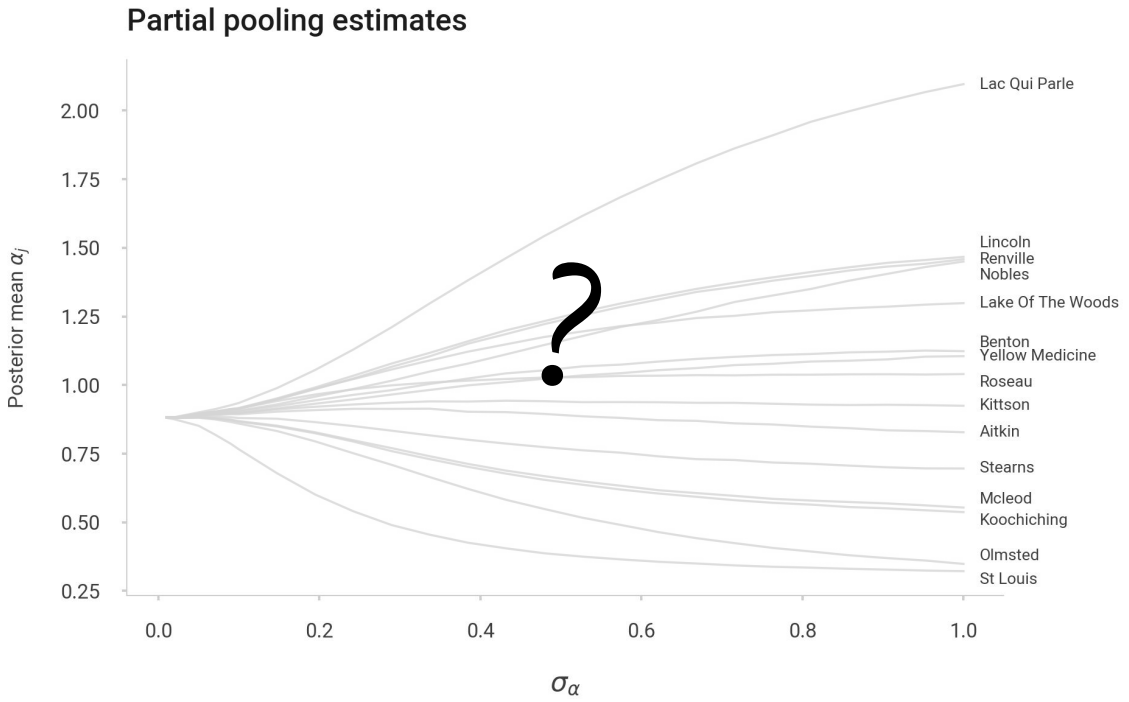
We can visualize the full spectrum of models.



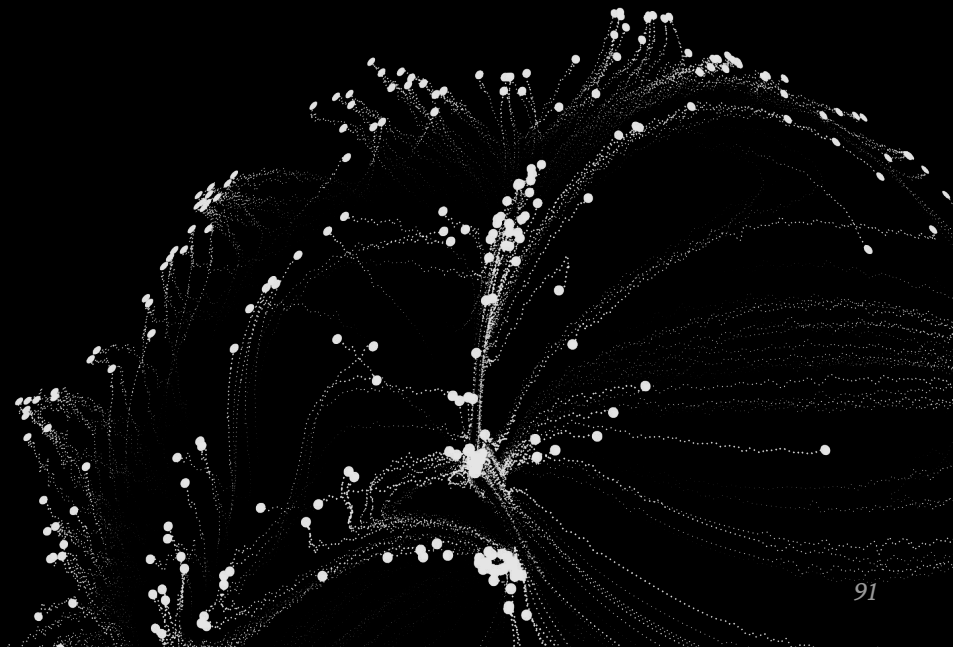
We can visualize the full spectrum of models.



How much pooling should we use?



04
Hierarchical modelling



faculty

In a hierarchical model, the right amount of pooling is also estimated from the data.

α_j captures *each* level at the ground floor.

β_j is *each* increase from basement to ground floor.

τ measures the *within*-county variation only.

μ_α captures the *average* level at the ground floor in **Minnesota**

σ_α *controls* how similar the group countly-level α's are

For j = 1, 2, ... , J:

$$
\begin{aligned}
y_j &\sim \text{normal}(\alpha_j + \beta_j \cdot \text{floor}, \tau) \\
\alpha_j &\sim \text{normal}(\mu_\alpha, \sigma_\alpha) \\
\beta_j &\sim \text{normal}(0, 1) \\
\frac{1}{\tau} &\sim \text{gamma}(1, 0.5) \\
\mu_\alpha &\sim \text{normal}(0, 5) \\
\sigma_\alpha &\sim \text{exponential}(1)
\end{aligned}
$$

In a hierarchical model, the right amount of pooling is also estimated from the data.

α_j captures *each* level at the ground floor.

β_j is *each* increase from basement to ground floor.

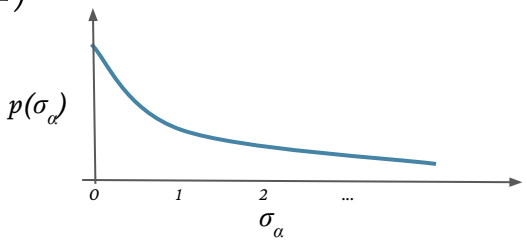
τ measures the *within*-county variation only.

μ_α captures the *average* level at the ground floor in **Minnesota**

σ_α *controls* how similar the group countly-level α's are

For j = 1, 2, ... , J:

$$y_j \sim \text{normal}(\alpha_j + \beta_j \cdot \text{floor}, \tau)$$
$$\alpha_j \sim \text{normal}(\mu_\alpha, \sigma_\alpha)$$
$$\beta_j \sim \text{normal}(0, 1)$$
$$\frac{1}{\tau} \sim \text{gamma}(1, 0.5)$$
$$\mu_\alpha \sim \text{normal}(0, 5)$$
$$\sigma_\alpha \sim \text{exponential}(1)$$



```
...

def hierarchical(county, floor, log_radon=None):
    μ_α = sample("μ_α", Normal(0, 5))
    σ_α = sample("σ_α", Exponential(1))

    with plate("counties", N_COUNTIES):
        α = sample("α", Normal(μ_α, σ_α))
        β = sample("β", Normal(0, 1))

    μ = deterministic("μ", α[county] + β[county] * floor)
    τ = sample("τ", InverseGamma(1, 0.5))
    sample("log_radon", Normal(μ, τ), obs=log_radon)

...
```

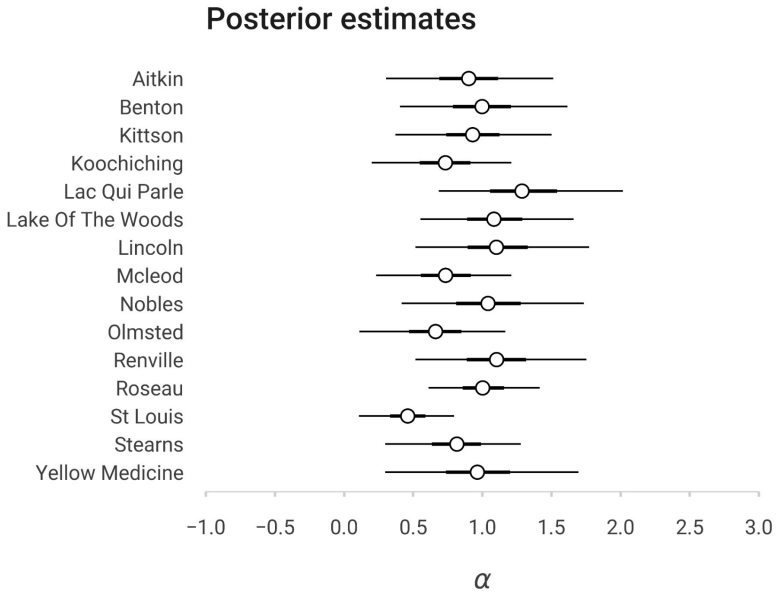
We no longer provide a hard-coded pooling

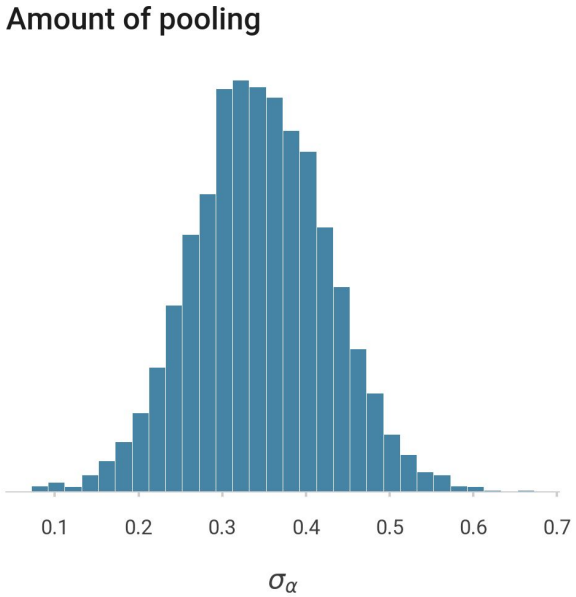
```
...

def hierarchical(county, floor, log_radon=None):
    μ_α = sample("μ_α", Normal(0, 5))
    σ_α = sample("σ_α", Exponential(1))

    with plate("counties", N_COUNTIES):
        α = sample("α", Normal(μ_α, σ_α))
        β = sample("β", Normal(0, 1))

    μ = deterministic("μ", α[county] + β[county] * floor)
    τ = sample("τ", InverseGamma(1, 0.5))
    sample("log_radon", Normal(μ, τ), obs=log_radon)

...
```

We simply specify a hyper-prior

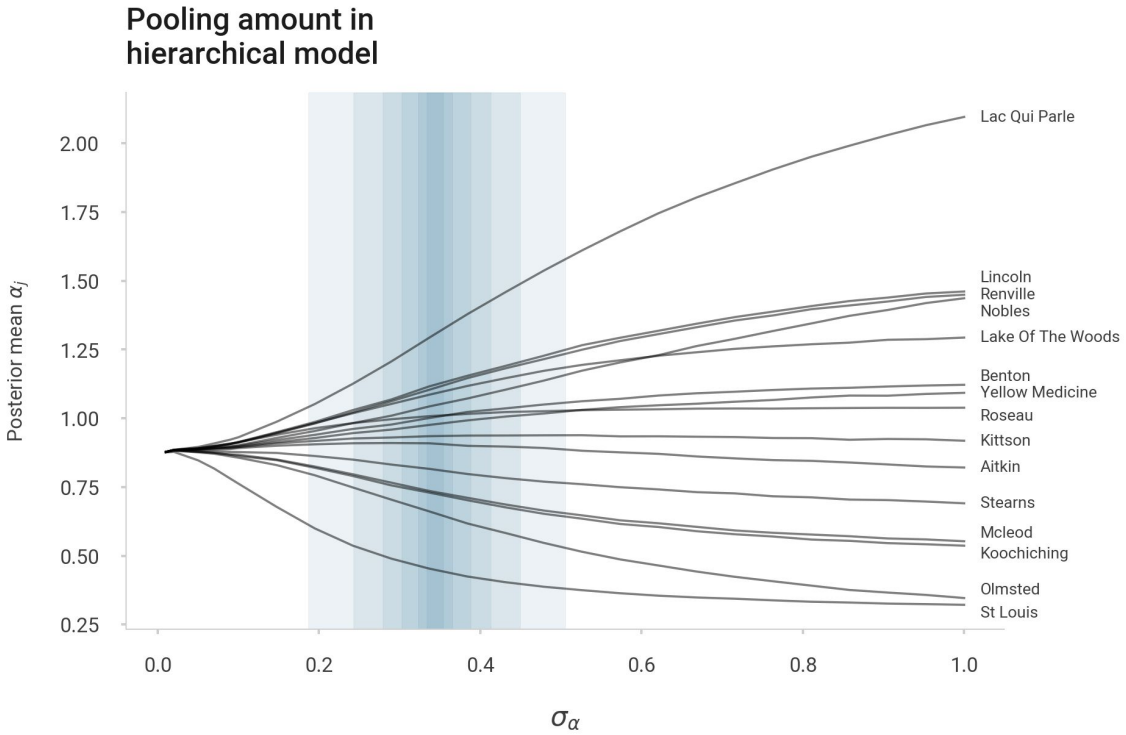
```
...

def hierarchical(county, floor, log_radon=None):
    μ_α = sample("μ_α", Normal(0, 5))
    σ_α = sample("σ_α", Exponential(1))

    with plate("counties", N_COUNTIES):
        α = sample("α", Normal(μ_α, σ_α))
        β = sample("β", Normal(0, 1))

    μ = deterministic("μ", α[county] + β[county] * floor)
    τ = sample("τ", InverseGamma(1, 0.5))
    sample("log_radon", Normal(μ, τ), obs=log_radon)

...
```

The posterior estimates exhibit a natural amount of pooling





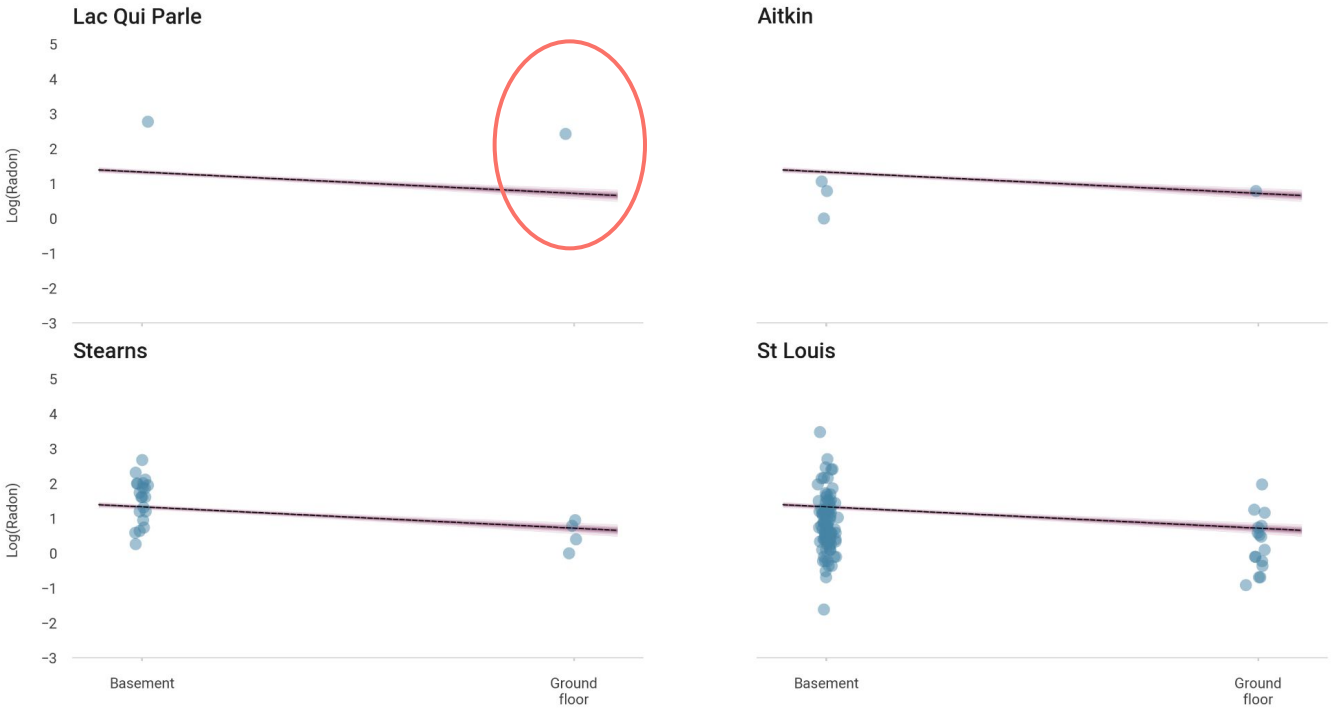
The posterior estimates exhibit a natural amount of pooling



Complete pooling

No pooling

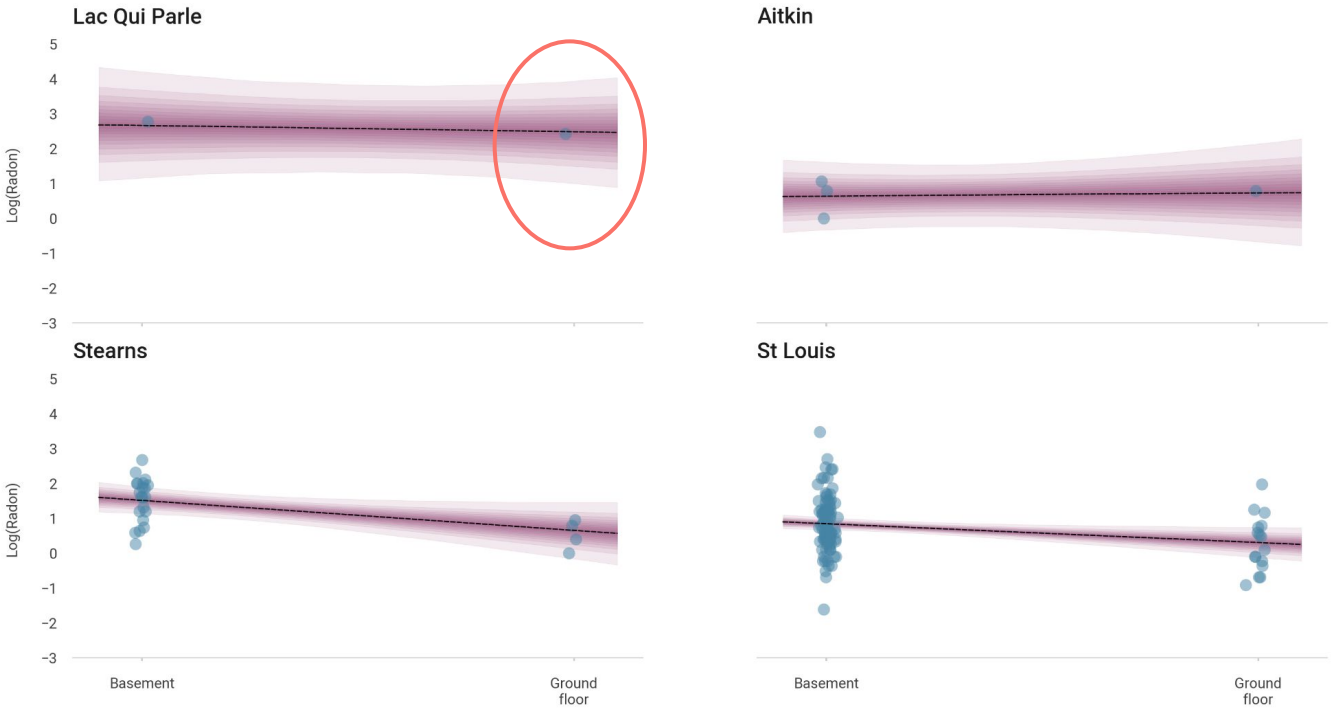
Hierarchical



Complete pooling

No pooling

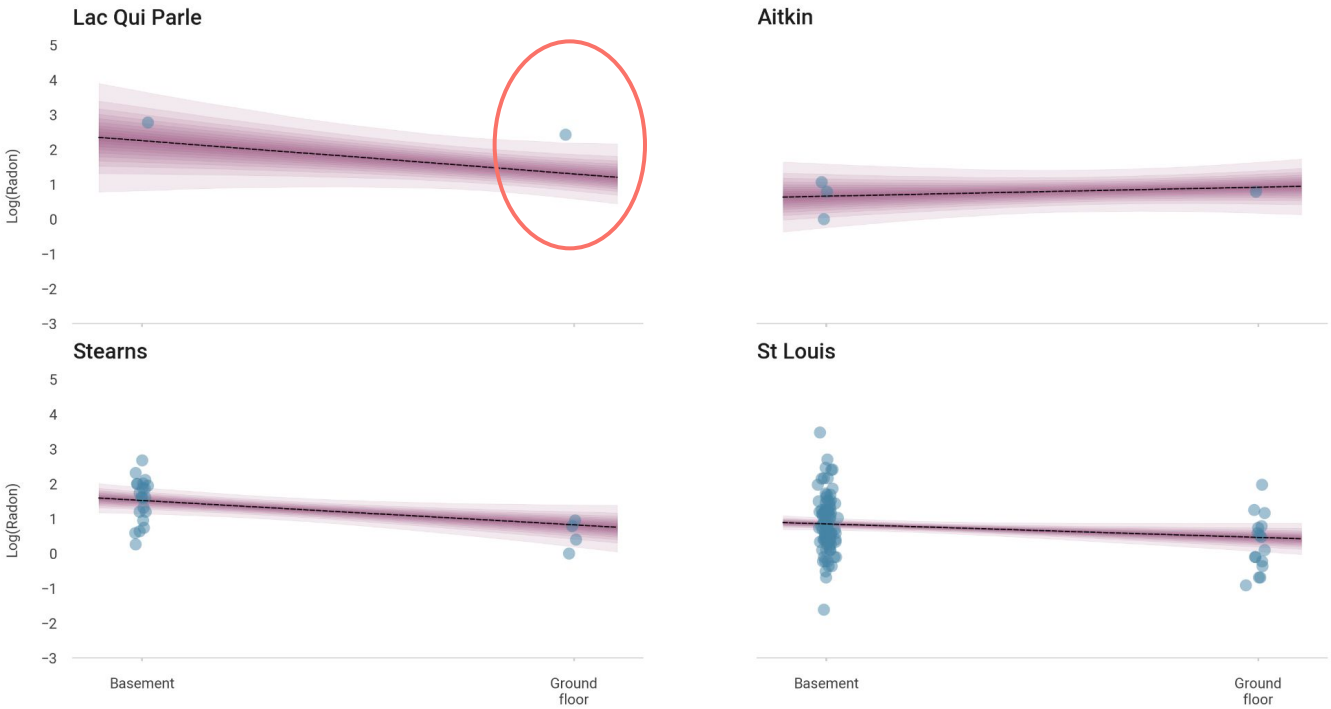
Hierarchical



Complete pooling

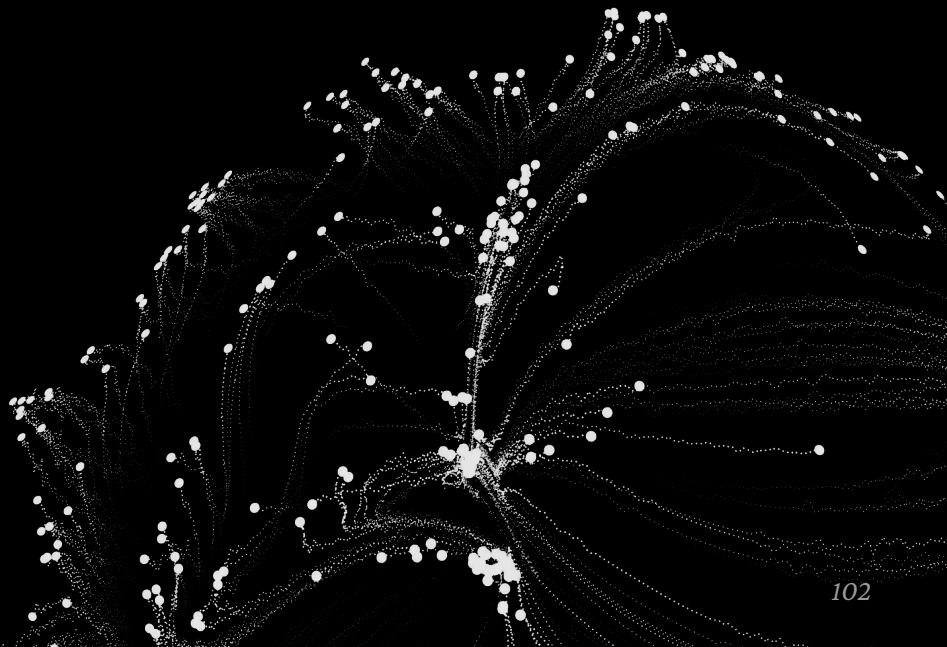
No pooling

Hierarchical



05

Group-level predictors



Sometimes, we might have some further information about the different groups.
This information does not directly relate to the data, but rather to the groups.

	county	floor	log_radon
0	Aitkin	0	0.788457
1	Aitkin	-1	0.788457
2	Aitkin	-1	1.064711
3	Aitkin	-1	0.000000
4	Anoka	-1	1.131402
...	...	...	...
914	Wright	-1	1.856298
915	Wright	-1	1.504077
916	Wright	-1	1.609438
917	Yellow Medicine	-1	1.308333
918	Yellow Medicine	-1	1.064711

919 rows × 3 columns

	county	uranium
0	Aitkin	0.502054
1	Anoka	0.428565
2	Becker	0.892741
3	Beltrami	0.552472
4	Benton	0.866849
...	...	...
82	Watonwan	1.201100
83	Wilkin	1.266220
84	Winona	1.589170
85	Wright	0.913909
86	Yellow Medicine	1.426590

87 rows × 2 columns

We incorporate the group level info by turning into our second model into a regression:

α_j captures *each* level at the ground floor.

β_j is *each* increase from basement to ground floor.

τ measures the *within*-county variation only.

μ_α captures the *average* level at the ground floor in **Minnesota**

γ_α captures the *relationship* between Uranium and Radon.

σ_α *controls* how similar the group countly-level α's are

For j = 1, 2, ... , J:

$$y_j \sim \text{normal}(\alpha_j + \beta_j \cdot \text{floor}, \tau)$$
$$\alpha_j \sim \text{normal}(\mu_\alpha + \gamma_\alpha \cdot \text{u}_j, \sigma_\alpha)$$
$$\beta_j \sim \text{normal}(0, 1)$$

$$\frac{1}{\tau} \sim \text{gamma}(1, 0.5)$$
$$\mu_\alpha \sim \text{normal}(0, 5)$$
$$\gamma_\alpha \sim \text{normal}(0, 5)$$
$$\sigma_\alpha \sim \text{exponential}(1)$$

We incorporate the group level info by turning into our second model into a regression:

α_j captures *each* level at the ground floor.

β_j is *each* increase from basement to ground floor.

τ measures the *within*-county variation only.

μ_α captures the *average* level at the ground floor in **Minnesota**

γ_α captures the *relationship* between Uranium and Radon.

σ_α *controls* how similar the group countly-level α's are

For j = 1, 2, ... , J:

$$y_j \sim \text{normal}(\alpha_j + \beta_j \cdot \text{floor}, \tau)$$

$$\alpha_j \sim \text{normal}(\mu_\alpha + \gamma_\alpha \cdot \text{u}_j, \sigma_\alpha)$$

$$\beta_j \sim \text{normal}(0, 1)$$

$$\frac{1}{\tau} \sim \text{gamma}(1, 0.5)$$

$$\mu_\alpha \sim \text{normal}(0, 5)$$

$$\gamma_\alpha \sim \text{normal}(0, 5)$$

$$\sigma_\alpha \sim \text{exponential}(1)$$

We incorporate the group level info by turning into our second model into a regression:

α_j captures *each* level at the ground floor.

β_j is *each* increase from basement to ground floor.

τ measures the *within*-county variation only.

μ_α captures the *average* level at the ground floor in **Minnesota**

γ_α captures the *relationship* between Uranium and Radon.

σ_α *controls* how similar the group countly-level α's are

For j = 1, 2, ... , J:

$$y_j \sim \text{normal}(\alpha_j + \beta_j \cdot \text{floor}, \tau)$$

$$\alpha_j \sim \text{normal}(\mu_\alpha + \gamma_\alpha \cdot \text{u}_j, \sigma_\alpha)$$

$$\beta_j \sim \text{normal}(0, 1)$$

$$\frac{1}{\tau} \sim \text{gamma}(1, 0.5)$$

$$\mu_\alpha \sim \text{normal}(0, 5)$$

$$\gamma_\alpha \sim \text{normal}(0, 5)$$

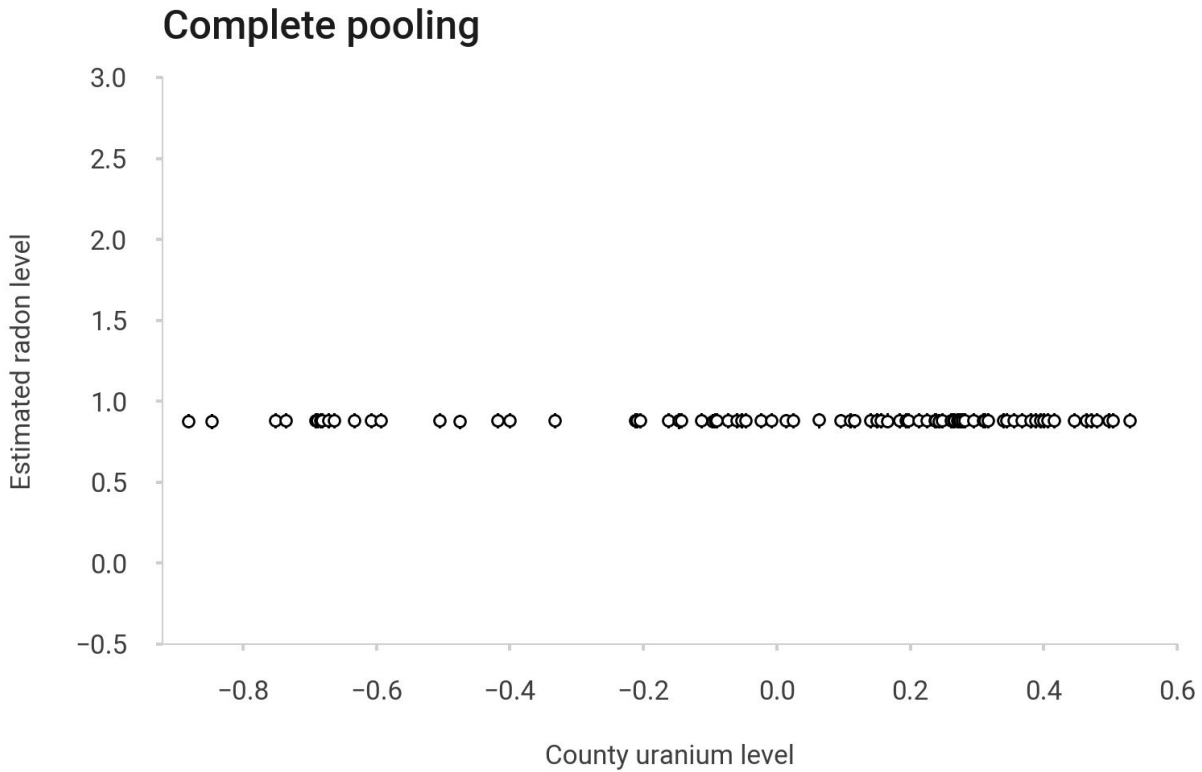
$$\sigma_\alpha \sim \text{exponential}(1)$$

No group variation

Uncontrolled variation

Controlled variation

Controlled & Explained

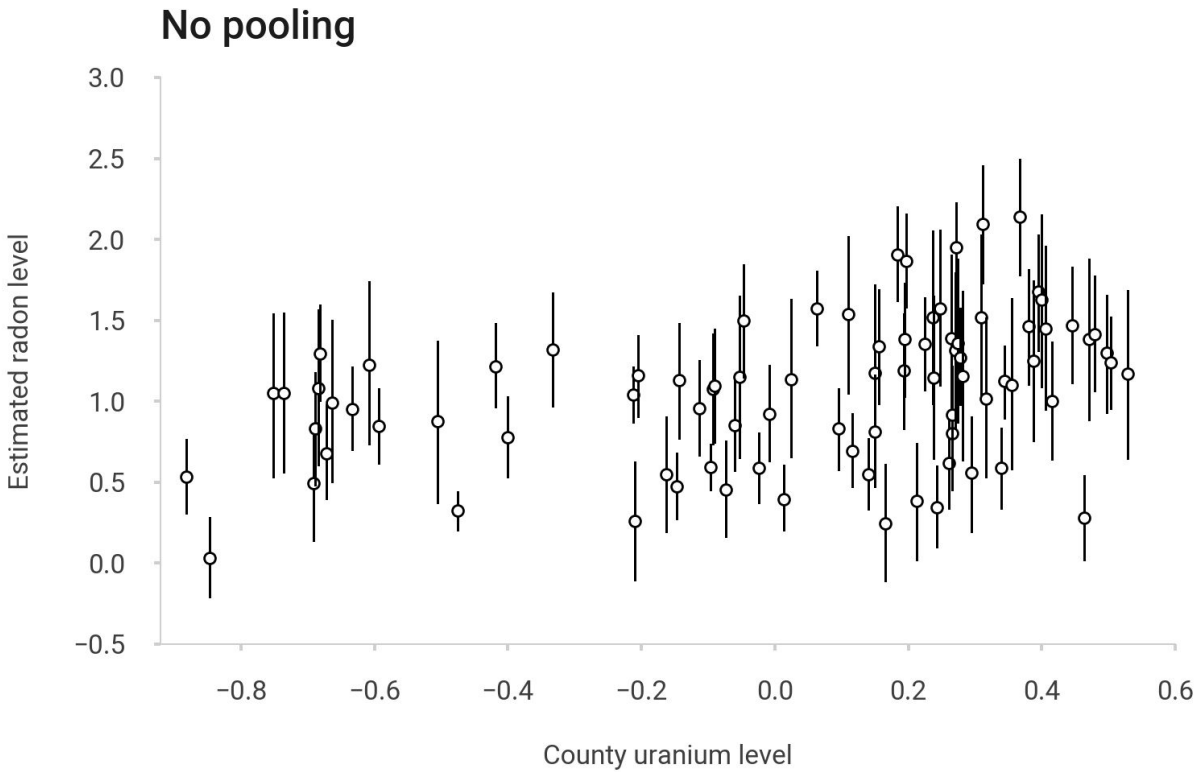


No group variation

Uncontrolled variation

Controlled variation

Controlled & Explained

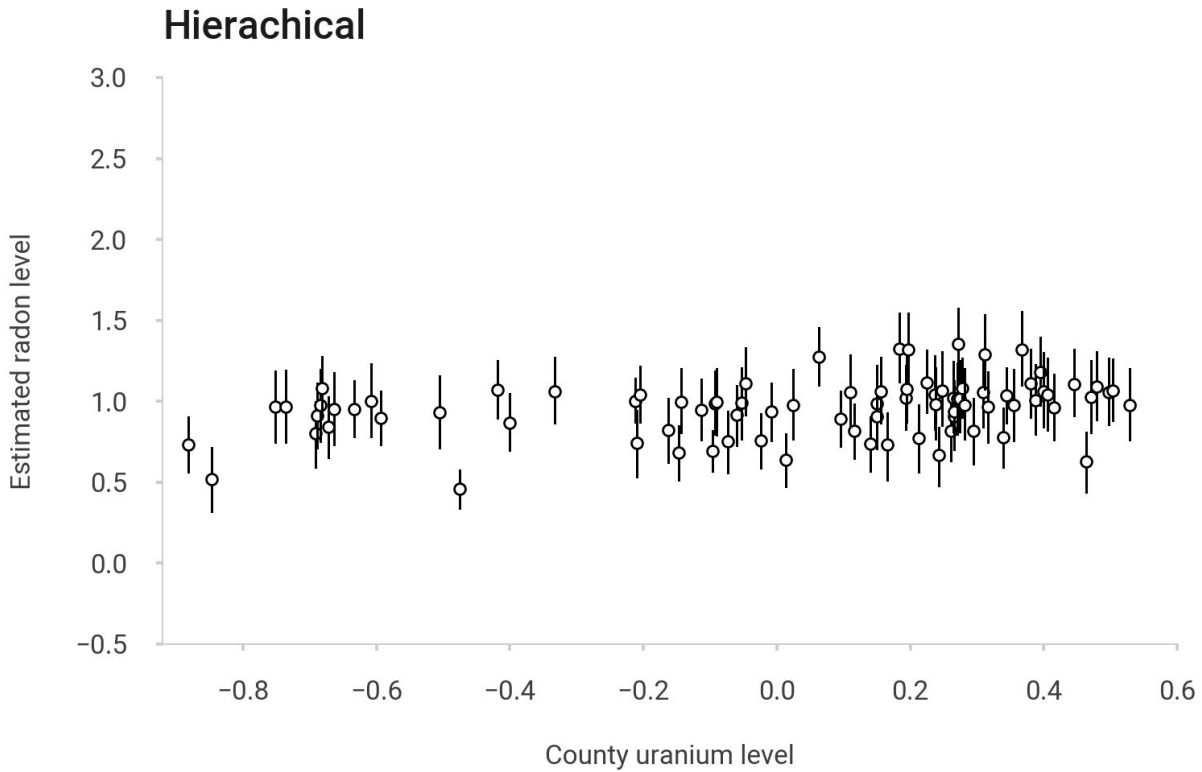


No group variation

Uncontrolled variation

Controlled variation

Controlled & Explained

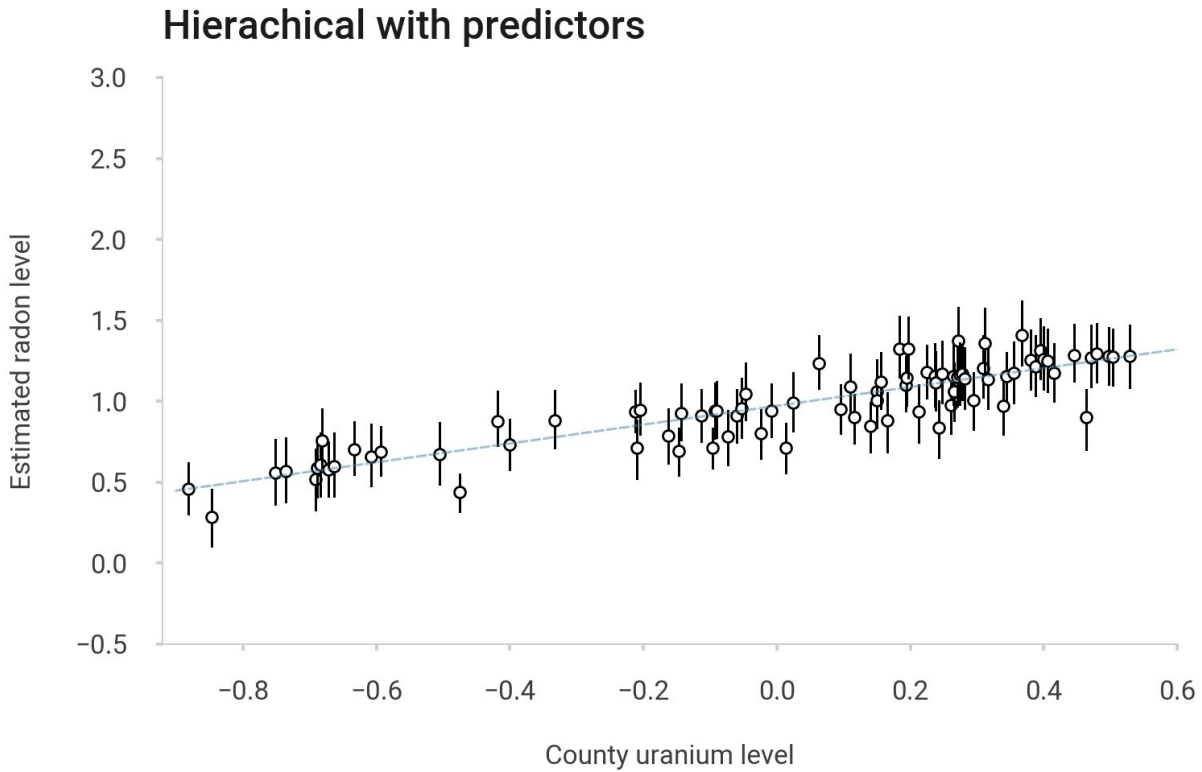


No group variation

Uncontrolled variation

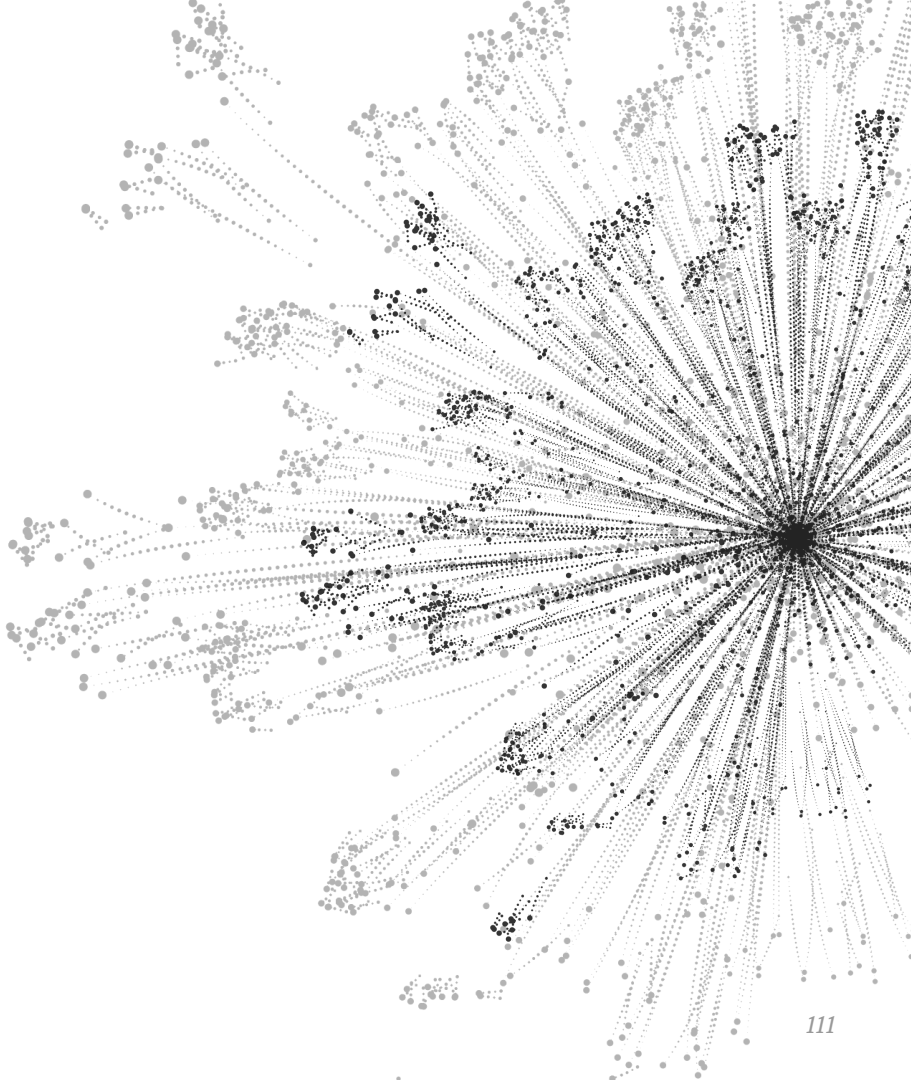
Controlled variation

Controlled & Explained



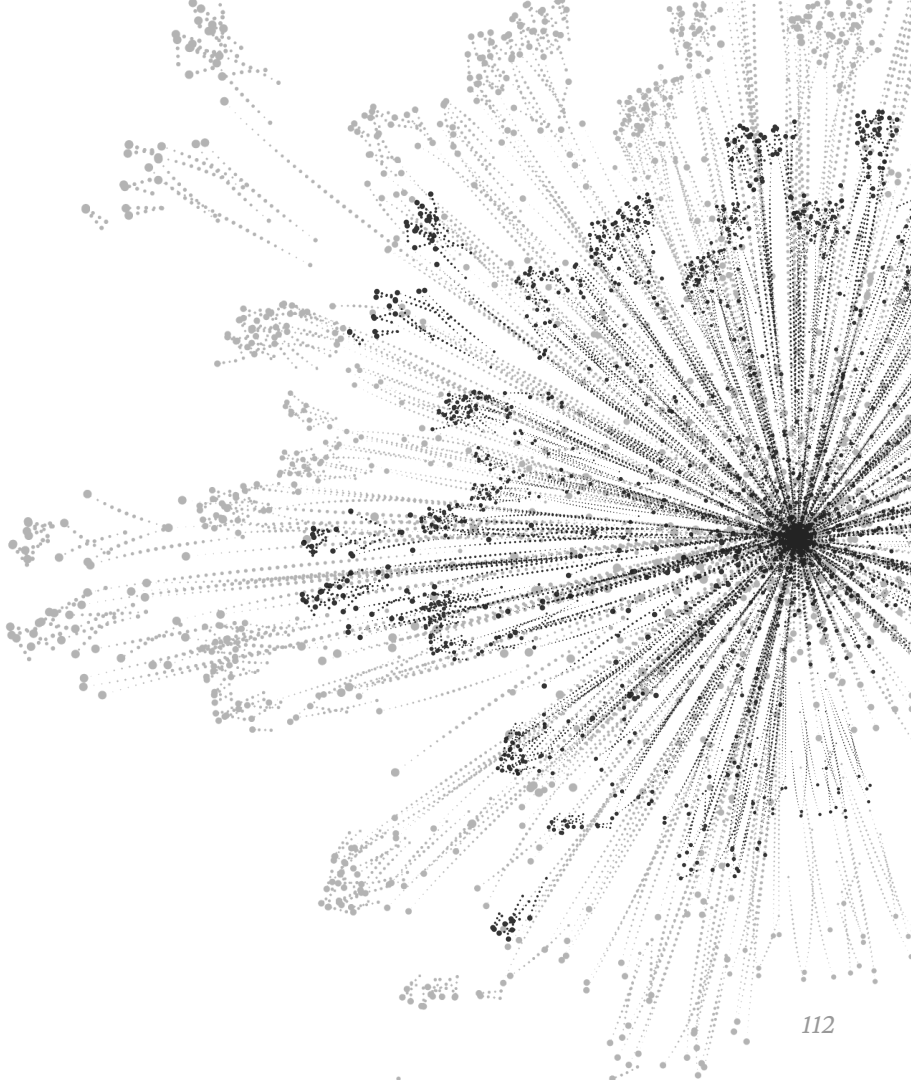
Key points

- Categorical data is a **recurrent challenge** in data science
- Traditional approaches usually account for the either the **similarity across groups, or the differences between them**. Never both.
- Hierarchical modelling provides a **rigorous framework** for combining both approaches.
- Hierarchical models implement the **different data sources** at the right level
- Hierarchical modelling works by **splitting the sources** of uncertainty.
- Straightforward approach (and elegant) to make **predictions for unseen groups**.



Warnings!

- Careful with the **non-bayesian approaches to hierarchical models** -- prepackaged software will make assumptions on your behalf.
- With the need for a Bayesian comes a **need for patience**.
- **The complexity of hierarchical models** can quickly get out of hands even with seemingly innocent data.
- It will often require **parametrize your model** in very specific ways.
- You will have to **tune your MCMC** very deliberately.



06

Q & A

faculty