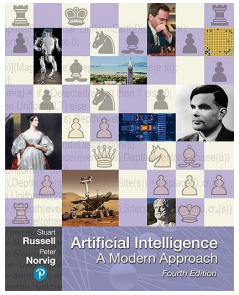
UNIVERSITY OF MARYLAND

MSML610: Advanced Machine Learning

Probabilistic Programming



Instructor: Dr. GP Saggese - gsaggese@umd.edu

References:

- AIMA (Artificial Intelligence: a Modern Approach)
  - Chap 15: Probabilistic programming
- Martin, Bayesian Analysis with Python, 2018 (2e)

- ***Concepts***
- Coin Example
- Posterior-Based Decisions
- Groups Comparison
- Hierarchical Models
- Simple Linear Model
- Multiple linear regression
- Comparing Models
- Inference Engines

EDA vs Inference

- **Exploratory data analysis**
  - Summarize, interpret, check data
  - Visually inspect the data
  - Compute descriptive statistics
  - Communicate results
- **Inferential statistics / inference**
  - Draw insights from a limited set of data
  - Make predictions for future unobserved data points
  - Understand a phenomenon
  - Choose among competing explanations for the same observations

Good vs Bad Way to Do Statistics

- **Bad** 😈
  - Learn a collection of "statistical recipes"
    - Make assumption / approximate to make math workable
  - Given data and problem
    - Pick one recipe
    - Try until you get a "low" p-value
  - For machine learning
    - Iterate until you get a "good" fit on out-of-sample data
- **Good** 😊
  - General approach to statistical inference (Bayesian statistics)
    - Remove limitations from closed analytical form
  - Probabilistic approach unifies (seemingly) disparate methods
    - E.g., statistical methods and machine learning
    - E.g., `statsmodels` linear regression vs `sklearn` decision tree
    - Deep unity of different recipes
  - Modern tools (e.g., PyMC3) solve previously unsolvable models

Data

- Data **comes from**:
  - Experiments
  - Simulations
  - Surveys
  - Field observations
- Data is stochastic due to **uncertainty**
  - Ontological: system is intrinsically stochastic
  - Technical: measurement precision is limited or noisy
  - Epistemic: conceptual limitations in understanding
- Collecting data is **costly**
  - Consider questions before collecting data
  - Experiment design is a branch of statistics for data collection
- Data is **rarely clean and tidy**
- Data needs to be **interpreted** through mental and formal models

Models

- **Models** are simplified descriptions of a given system/process
  - A more complex model is not always a better one
  - VC dimension made it mathematical precise
    - *"You need at least 10 data points per effective degree of freedom of the hypothesis set"*
- **Goals**
  - Capture the most relevant aspects of the system
  - Ignore minor details

Bayes' Theorem: Recap

- **Bayes' theorem** posits that for model parameters θ and data X

$$\Pr(\theta|X) = \frac{\Pr(X|\theta) \cdot \Pr(\theta)}{\Pr(X)}$$

where:

- $\Pr(\theta|X)$
  - **Posterior**: probability for parameters θ after seeing data X
- $\Pr(X|\theta)$
  - **Likelihood** (aka "statistical model"): plausibility of data X given parameters θ
- $\Pr(\theta)$
  - **Prior**: knowledge about parameter θ before any data
- $\Pr(X)$
  - **Evidence** ("marginal likelihood"): probability of observing data X
  - "Marginal" as it averages over all possible parameter values

- In other words:

$$\text{Posterior} = \frac{\text{Likelihood} \cdot \text{Prior}}{\text{Evidence}}$$

Bayesian Models

- **Probability** measures uncertainty about parameters
- **Bayes' theorem** updates probabilities with new data, reducing uncertainty (hopefully)

$$\Pr(hyp|data) = \frac{\Pr(data|hyp)\Pr(hyp)}{\Pr(data)}$$

- **Bayesian modeling workflow**
  1. Design a model using probabilities based on data and assumptions
     - Assumptions on data generation
     - Model can be a crude approximation
  2. Apply Bayes' theorem to "condition" the model on data
  3. Validate model against:
     - Data
     - Subject expertise
     - Related models

- Steps may involve backtracking:
  - Correct coding errors
  - Improve model
  - Gather more or different data

- Concepts
- ***Coin Example***
  - Analytical Approach
  - Frequentist vs Bayesian
  - Probabilistic Programming
- Posterior-Based Decisions
- Groups Comparison
- Hierarchical Models
- Simple Linear Model
- Multiple linear regression
- Comparing Models
- Inference Engines

- Concepts
- Coin Example
  - ***Analytical Approach***
  - Frequentist vs Bayesian
  - Probabilistic Programming
- Posterior-Based Decisions
- Groups Comparison
- Hierarchical Models
- Simple Linear Model
- Multiple linear regression
- Comparing Models
- Inference Engines

Coin Example: Problem

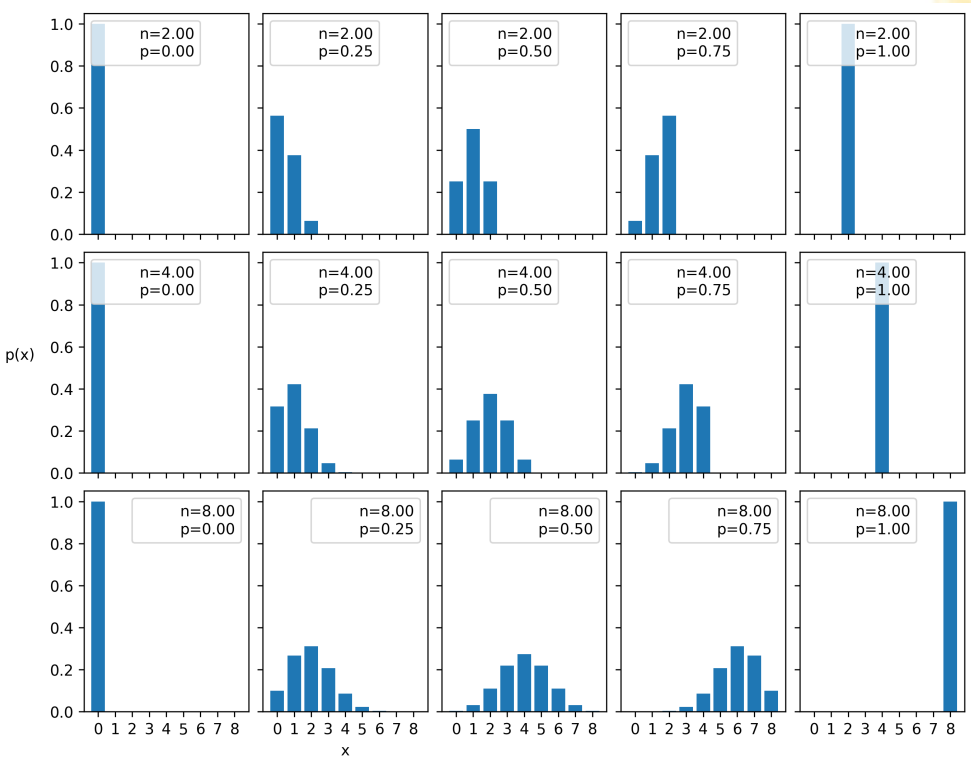
- **Problem**:
  - Toss a coin N times
  - Record the number of heads Y and tails $N - Y$
  - Question: *"How biased is the coin?"*
- There is **true uncertainty**
  - An underlying parameter exists, but it is unknown
  - θ represents the coin bias
    - 0: always tails
    - 1: always heads
    - 0.5: half tails, half heads
- **Model assumptions**:
  - Independent Identically Distributed (IID)
    - Independence: coin tosses don't affect each other
    - Identically distributed: coin's bias is constant
  - Likelihood $Y|\theta$ as a binomial distribution
    - Probability of Y heads out of N tosses, given θ
  - Prior θ as a beta distribution
    - Adopts several shapes
    - Beta is the conjugate prior of the binomial distribution

Binomial Distribution

Probability of k heads out of n tosses given bias p

$$X \sim Binomial(n, p)$$

$$\Pr(k) = \frac{n!}{k!(n-k)!} p^k (1-p)^{n-k}$$

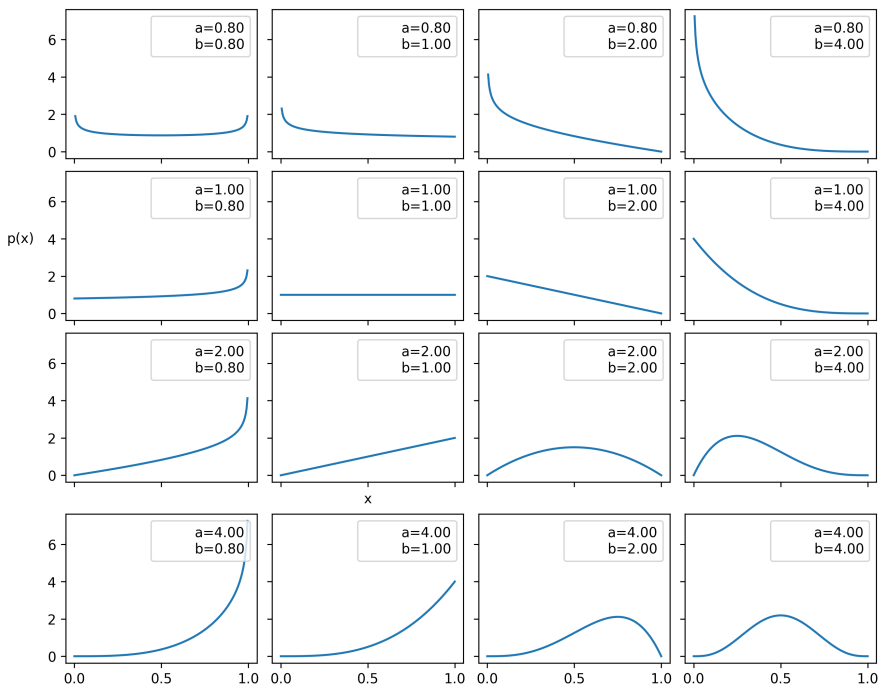


Beta Distribution

- Continuous PDF in [0, 1]
- Adopts several shapes
  - Uniform, increasing, decreasing, Gaussian-like, U-like
  - α: "success" parameter
  - β: "failure" parameter
  - $\alpha > \beta$: Skews toward 1, higher probability of success
  - $\alpha = \beta$: Symmetric, centered around 0.5
- Models probability or proportion
  - E.g., probability of success in a Bernoulli trial θ
- Beta is the conjugate prior of the binomial distribution

$$X \sim Beta(\alpha, \beta)$$

$$\Pr(\theta) = \frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)\Gamma(\beta)}\theta^{\alpha-1}(1-\theta)^{\beta-1}$$



Conjugate Prior of a Likelihood

- **Conjugate prior** is a prior that, when combined with a likelihood, returns a posterior with the same functional form as the prior
  - E.g.,

Prior	Likelihood	Posterior
Beta	Binomial	Beta
Normal	Normal	Normal

- **Properties**
  - Prior and posterior have the same distribution
  - Posterior has a closed analytical form
    - Update parameters from the prior using data in multiple iterations
  - Ensures tractability of the posterior

Coin Example: Analytical Solution

- The **posterior** is proportional to **likelihood** $\times$ **prior**

$$\Pr(\theta|y) \propto \Pr(y|\theta)\Pr(\theta)$$

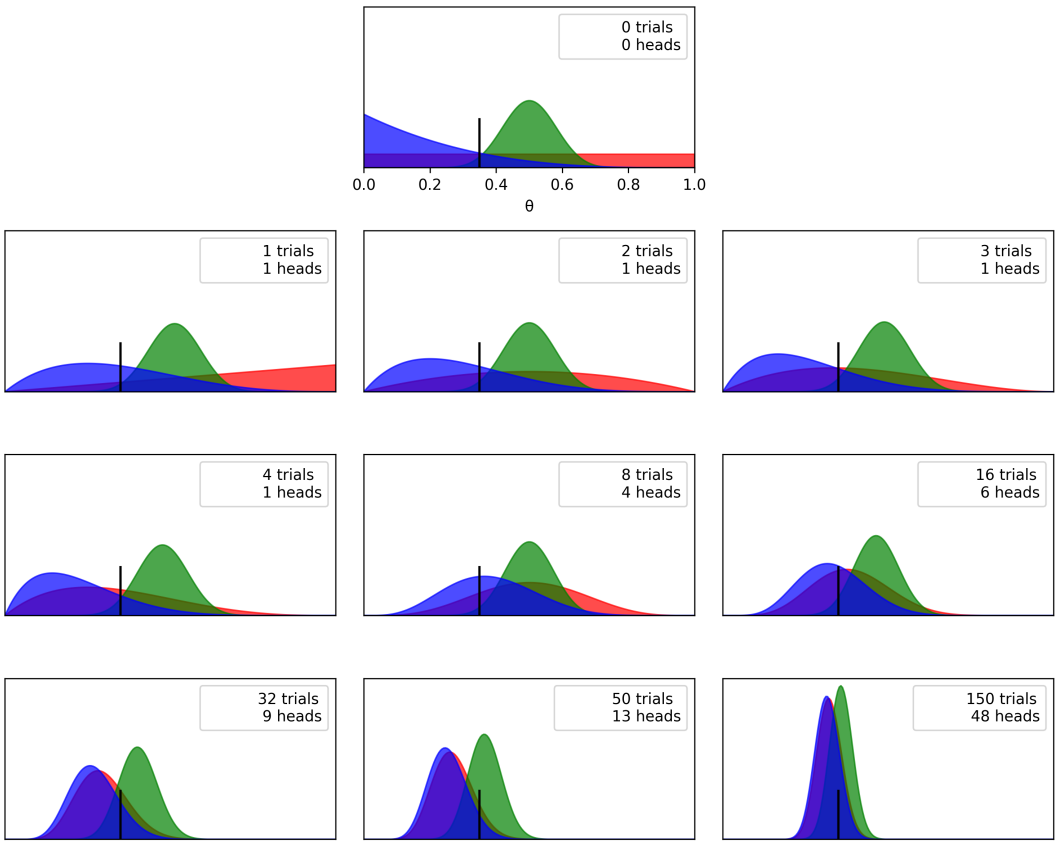
- Substituting **likelihood** with a Binomial and **prior** with a Beta

$$\begin{aligned}
&\Pr(\theta \mid Y) \\
&= \underbrace{\frac{N!}{y!(N-y)!}\theta^{y}(1-\theta)^{N-y}}_{\text{likelihood}} \underbrace{\frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)\Gamma(\beta)}\theta^{\alpha-1}(1-\theta)^{\beta-1}}_{\text{prior}} \\
&\propto \underbrace{\theta^{y}(1-\theta)^{N-y}}_{\text{likelihood}} \underbrace{\theta^{\alpha-1}(1-\theta)^{\beta-1}}_{\text{prior}} \\
&= \theta^{y+\alpha-1}(1-\theta)^{N-y+\beta-1} \\
&= \text{Beta}\left(\alpha_{\text{prior}}+y, \beta_{\text{prior}}+N-y\right)
\end{aligned}$$

- This is how **the posterior is updated** given the data

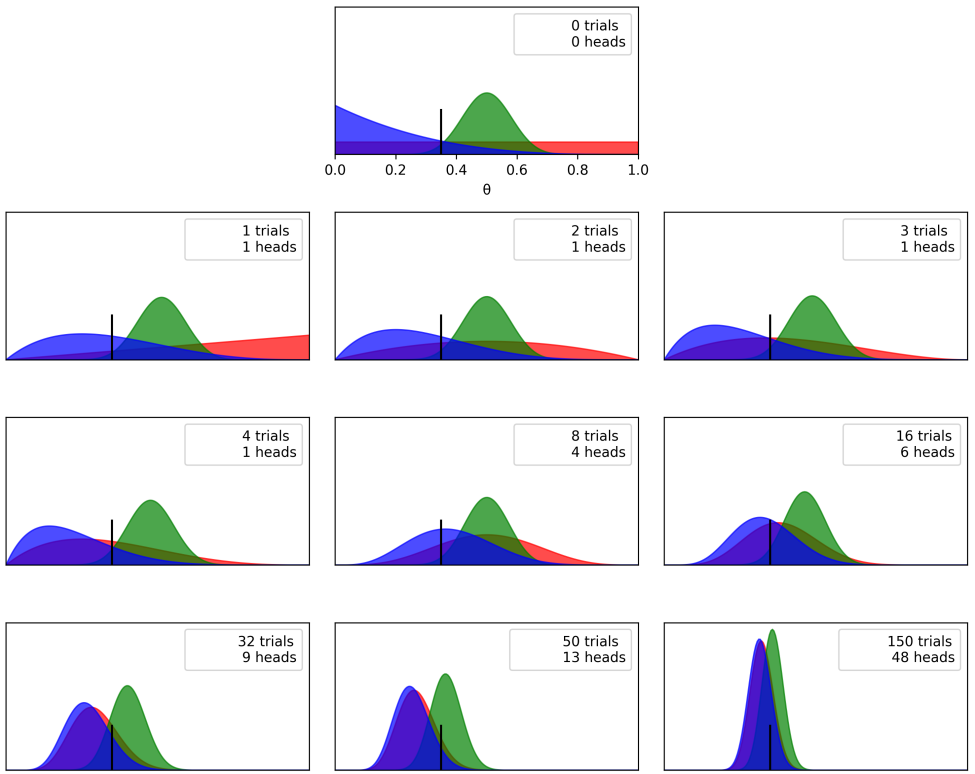
Coin Example: Effect of Priors (1/2)

- The true (unknown) value of the coin bias is 0.35
- Start with 3 different priors and update the model
  - Red: uniform prior
    - All bias values equally probable
  - Green: Gaussian-like prior around 0.5
    - Coin mostly unbiased
  - Blue: skewed towards tail
    - Coin biased
- Apply data to update the posterior distribution
- Update model



Coin Example: Effect of Priors (2/2)

- **Outcome of Bayesian analysis**
  - Posterior distribution, not a single value
- **Spread of posterior**
  - Proportional to uncertainty
  - Decreases with more data
  - Decreases faster if aligned with prior
  - With enough data, models with different priors converge to same result
- Applying posterior sequentially or at once yields same result



- Concepts
- Coin Example
  - Analytical Approach
  - ***Frequentist vs Bayesian***
  - Probabilistic Programming
- Posterior-Based Decisions
- Groups Comparison
- Hierarchical Models
- Simple Linear Model
- Multiple linear regression
- Comparing Models
- Inference Engines

Frequentist Approach vs Priors

- **Detractors of Bayesian approach** complain that:
  - *"One should let the data speak"*
  - The prior doesn't let the data speak for itself
- ⚖ **Counterpoints**
  - *"Data doesn't speak, but murmurs"*
    - Data doesn't have meaning per-se
    - Make sense of data only in context of models (e.g., mental models, mathematical models)
    - A prior is a mathematical model
  - Every statistical model has a prior, even if not explicit
    - Frequentist statistics still makes assumptions (i.e., has a prior), but are hidden
    - E.g., maximum likelihood estimate (MLE) in frequentist approach corresponds to a uniform prior and mode of the posterior
    - E.g, MLE is a point-estimate, not a distribution of plausible values

Advantages of Using Prior

- **Assumptions are clear and explicit**
  - Instead of hidden by frequentist or hacker ML approach
- **Prior**
  - Encourages deeper analysis of problem and data
  - Forces understanding before seeing data
- Posterior averaged over priors is **less prone to overfitting**
- Spread of distribution measures **uncertainty**
- Well-chosen prior simplifies and **speeds up inference**
  - *"When you encounter computational problems, there's often an issue with your model"* (Gelman, 2008)

How to Choose Priors

- **Weakly-informative priors** (aka "flat", "vague", "diffuse priors")
  - Provide minimal information
    - Coefficient of linear regression centered around 0: $\beta \sim \mathit{Normal}(0, 10)$
- **Regularizing priors**
  - Known information about the parameter
    - Parameter is positive: $\sigma \sim \mathit{HalfCauchy}(0, 5)$
    - Parameter close to zero, above/below a number, or in a range
    - $\beta \sim \mathit{Laplace}(0, 1)$ (lasso prior) encourages sparsity
    - $\beta \sim \mathit{Normal}(0, 1)$ discourages extreme values
- **Informative priors**
  - Strong priors from previous knowledge (expert opinion, studies)
    - From experimental data: $\beta_1 \sim \mathit{Normal}(2.5, 0.5^2)$
    - From previous data, about 5% of cases positive: $p \sim \mathit{Beta}(2, 38)$
- **Prior elicitation**
  - Compute least informative distribution given constraints
    - Estimate distribution using maximum entropy to satisfy constraints
    - E.g., beta distribution with 90% of mass between 0.1 and 0.7

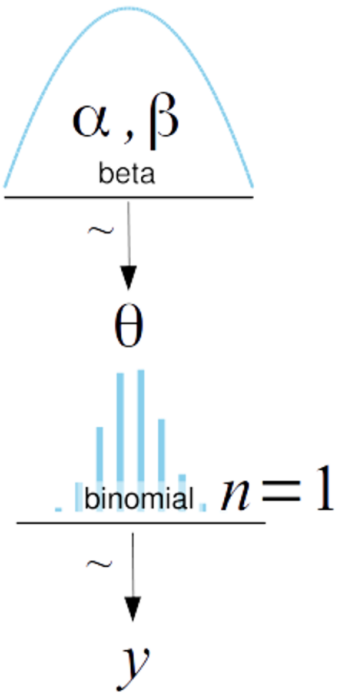
Communicating the Model of a Bayesian Analysis

1. **Communicate assumptions / hypothesis**
   - Describe priors and probabilistic models
   - E.g., coin-flip distributions:

$$\begin{cases} \theta \sim \text{Beta}(\alpha, \beta) \\ y \sim \textit{Binomial}(n = 1, p = \theta) \end{cases}$$

2. **Communicate Bayesian analysis result**
   - Describe posterior distribution
   - Summarize location and dispersion
   - Mean (or mode, median)
   - Std dev
     - Misleading for skewed distributions
   - Highest-posterior density (HPD)
     - Shortest interval containing a portion of probability density (e.g., 95% or 50%)
     - Amount is arbitrary (e.g., `ArviZ` defaults to 94%)

Kruschke diagram



Confidence Intervals vs Credible Intervals

- People confuse:
  - **Frequentist confidence intervals**
  - **Bayesian credible intervals**
- In the frequentist framework, there is a true (unknown) parameter value
  - A **confidence interval** may or may not contain the true parameter value
  - Interpretation of a 95% confidence interval
    - ❌ No: *"There is a 95% probability that the true value is in this interval"*
    - ✅ Yes: *"If repeated many times, 95% of intervals would contain the true value"*
- In the Bayesian framework, parameters are random variables
  - Interpretation of a 95% **Bayesian credible interval**
    - *"There is a 95% probability that the true parameter lies within this interval, given the observed data"*
    - Bayesian **credible interval** is intuitive

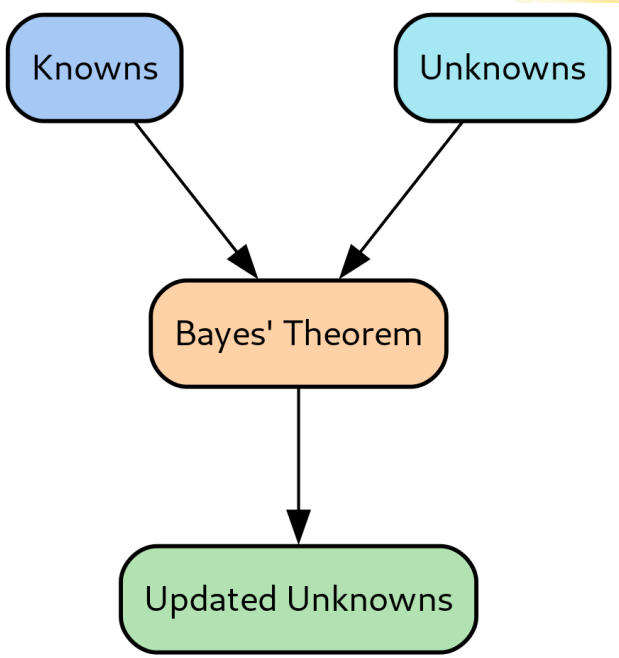
Confidence Intervals vs Credible Intervals (ELI5)

- **Confidence Interval (Frequentist)**
  - Imagine fishing in a lake without seeing the fish
  - You throw your net
  - 95% confidence interval: *"If I threw this net 100 times, about 95 nets would catch the fish."*
  - Important: Once the net is thrown, it either caught the fish or not. The 95% makes sense across many attempts
- **Credible Interval (Bayesian)**
  - Imagine a magical map showing where fish *probably* are, based on past observations
  - 95% credible interval: *"Given my map, there's a 95% chance the fish is inside this part of the lake."*
  - The fish's location is uncertain, and probability describes your belief

- Concepts
- Coin Example
  - Analytical Approach
  - Frequentist vs Bayesian
  - ***Probabilistic Programming***
- Posterior-Based Decisions
- Groups Comparison
- Hierarchical Models
- Simple Linear Model
- Multiple linear regression
- Comparing Models
- Inference Engines

Bayesian Statistics

- Given:
  - The **"knows"**
    - Model structure (modeled as a graph of probability distributions)
    - Data, observations (modeled as constants)
  - The **"unknowns"**
    - Model parameters (modeled as probability distributions)
- Use Bayes' theorem to:
  - Condition unknowns to knowns
  - Reduce the uncertainty about the unknowns



- **Problem**
  - Most probabilistic models are analytically intractable
- **Solution**
  - Probabilistic programming
    - Specify a probabilistic model using code
    - Solve models using numerical techniques

Probabilistic Programming Languages

- **Steps**:
  1. Specify models using code
  2. Numerical models solve inference problems without need of user to understand how
     - Universal inference engines
     - `PyMC3`: flexible Python library for probabilistic programming
     - `Theano`: library to define, optimize, evaluate mathematical expressions using tensors
     - `ArviZ`: library to interpret probabilistic model results
- **Pros**:
  - Compute results without analytical closed form
  - Treat model solving as a black box
  - Focus on model design, evaluation, interpretation
- **Probabilistic programming languages**
  - Similar impact as Fortran on scientific computing
  - Build algorithms but ignore computational details

Coin Example: Numerical Solution (1/3)

- It's a synthetic example!
  - Assume you know the true value of θ (not true in general)
- **Workflow**
  - Model the prior θ and the likelihood $Y|\theta$

$$\begin{cases} \theta \sim \text{Beta}(\alpha = 1, \beta = 1) \\ Y \sim \text{Binomial}(n = 1, p = \theta) \end{cases}$$

  - Observe samples of the variable Y
  - Run inference
  - Generate samples of the posterior
  - Summarize posterior
    - E.g., Highest-Posterior Density (HPD)
  - ...

Coin Example: Numerical Solution (2/3)

- Generate data from ground truth model
- Build PyMC model matching mathematical model
- PyMC uses NUTS sampler, computes 4 chains
- No trace diverges
- Kernel density estimation (KDE) for posterior
- Should be Beta
- Traces appear "noisy" and non-diverging (good)
- Numerical summary of posterior: mean, std dev, HDI
- $\mathbb{E}[\hat{\theta}] \approx 0.324$
- $\Pr(\hat{\theta} \in [0.031, 0.653]) = 0.94$

[18]:
```
np.random.seed(123)
n = 4
# Unknown value.
theta_real = 0.35

# Generate some observational data.
data = stats.bernoulli.rvs(p=theta_real, size=n)
data
```

[18]:
```
array([1, 0, 0, 0])
```

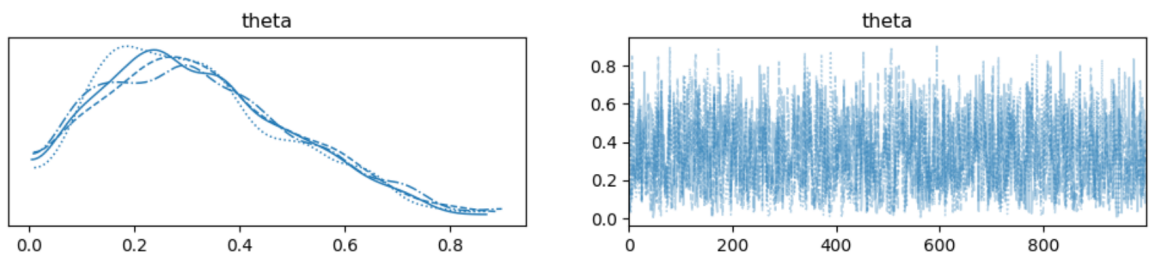
[19]:
```
with pm.Model() as our_first_model:
    # Prior.
    theta = pm.Beta('theta', alpha=1., beta=1.)
    # Likelihood.
    y = pm.Bernoulli('y', p=theta, observed=data)
    # (Numerical) Inference to estimate the posterior distribution through samples.
    idata = pm.sample(1000, random_seed=123)
```

```
Auto-assigning NUTS sampler...
Initializing NUTS using jitter+adapt_diag...
Multiprocess sampling (4 chains in 4 jobs)
NUTS: [theta]
```

```
Sampling 4 chains, 0 divergences ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 100% 0:00:00 / 0:00:00
```

```
Sampling 4 chains for 1_000 tune and 1_000 draw iterations (4_000 + 4_000 draws total) took 1 seconds.
```

[20]:
```
az.plot_trace(idata);
```



[21]:
```
az.summary(idata)
```

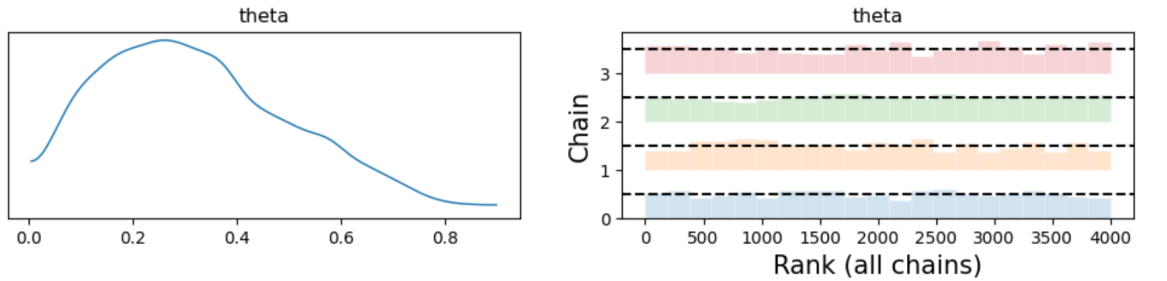
[21]:

	mean	sd	hdi_3%	hdi_97%	mcse_mean	mcse_sd	ess_bulk	ess_tail	r_hat
theta	0.324	0.179	0.031	0.653	0.005	0.003	1500.0	1737.0	1.0

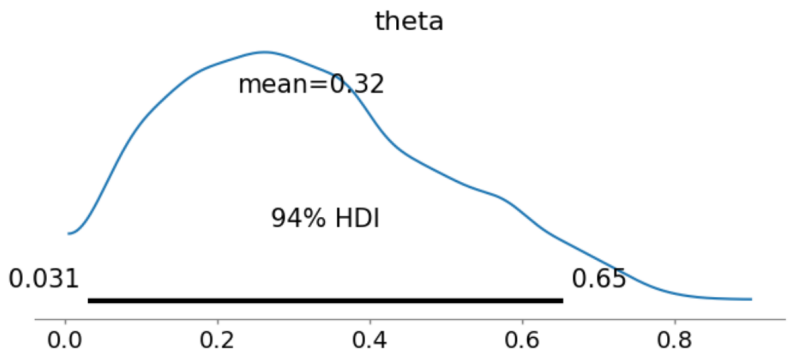
Coin Example: Numerical Solution (3/3)

- Compute single KDE for all chains
- Rank plot to check results
- Histograms should look uniform, exploring different (and all) posterior regions
- Plot single KDE with all statistics

```
[22]: az.plot_trace(idata, kind="rank_bars", combined=True);
```




```
[23]: az.plot_posterior(idata);
```



- Concepts
- Coin Example
- ***Posterior-Based Decisions***
  - Chemical Shift: Example
  - Posterior Predictive Checks
- Groups Comparison
- Hierarchical Models
- Simple Linear Model
- Multiple linear regression
- Comparing Models
- Inference Engines

Posterior-Based Decisions

- Sometimes describing the posterior is not enough
  - You need to make decisions based on our inference
- E.g., is the coin fair ($\theta = 0.5$) or biased?
  - Since $\mathbb{E}[\hat{\theta}] = 0.324$ it seems that the coin is biased
  - You can't rule out that the coin in unbiased since
    - $HPI = [0.03, 0.65]$
    - $0.5 \in HPI$
- If you want a sharper decision, you need to:
  - Collect more data to reduce the spread of the posterior
  - Define a more informative prior

Savage-Dickey Density Ratio

- The **Savage-Dickey ratio** tests a *point null-hypotheses* in Bayesian inference
- **Idea**: compare prior and posterior densities at a single point θ_0

$$BF_{01} = \frac{p(\theta_0|H_1)}{p(\theta_0|\mathcal{D}, H_1)}$$

where:

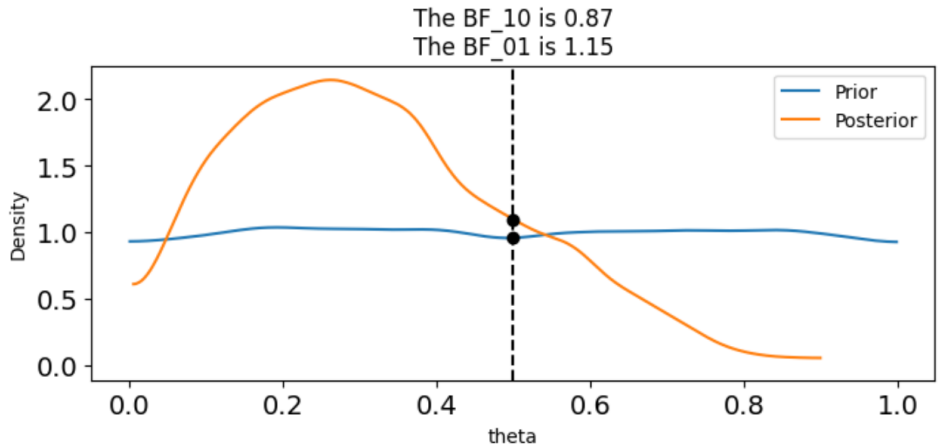
  - $p(\theta_0|H_1)$ is the *prior* density θ under the alternative hypothesis H_1, evaluated at θ_0
  - $p(\theta_0|\mathcal{D}, H_1)$ is the *posterior* density θ under H_1 evaluated at θ_0

Bayes Factor (BF)	Interpretation
1 - 3	Not enough evidence
3 - 10	Substantial evidence
10 - 100	Strong evidence
> 100	Decisive evidence

- **Intuition**: show how data changes belief about θ_0
  - If posterior density at θ_0 is much smaller than prior density, strong evidence against null hypothesis H_0
- **Cons**:

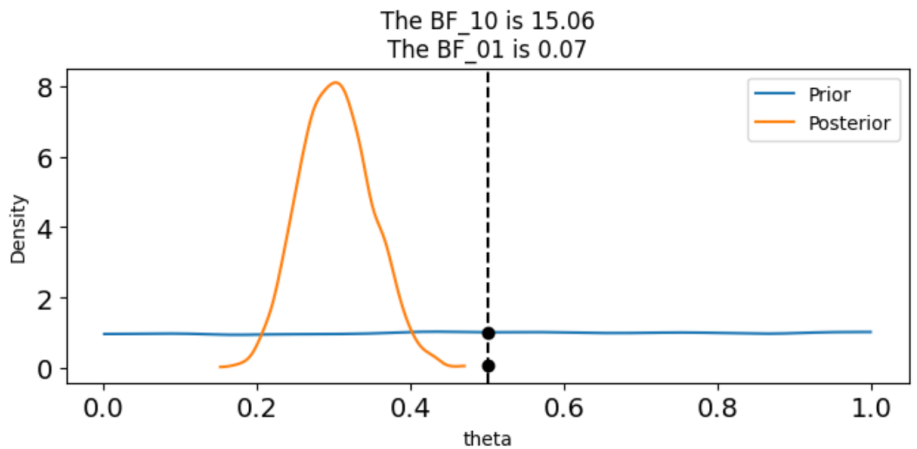
Savage-Dickey Density Ratio: Example

```
[29]: az.plot_bf(idata1, var_name="theta", prior=np.random.uniform(0, 1, 10000), ref_val=0.5);
```




```
[30]: az.plot_bf(idata2, var_name="theta", prior=np.random.uniform(0, 1, 10000), ref_val=0.5);
```

```
The reference value is outside of the posterior. This translate into infinite support for H1
```



- H_0: "coin is fair"
  - The prior for H_0 is 0.87
  - The posterior for H_0 is 1.15
  - $BF_{10} = 0.76$ $\rightarrow$ no evidence to reject the null hypothesis
- In the other case, very strong evidence
  - $BF_{10} = 15.06$ $\rightarrow$ strong evidence to reject the null hypothesis

ROPE: Region of Practical Equivalence

- **ROPE** = an interval for a parameter where all values inside are considered "equivalent"
- **Example**
  - H_0: *"coin is fair"* iff $\theta = 0.5$ is impractical
  - ROPE: $\theta \in [0.45, 0.55]$ is equivalent to 0.5
- **Hypothesis testing with ROPE and HPI**
  - Compare ROPE (Region Of Practical Equivalence) with HPI (Highest-Posterior Interval)
    - If HPI is within ROPE $\rightarrow$ no effect, H_1 is rejected
    - If HPI is outside ROPE *to* effect present, H_0 is rejected
    - If HPI overlaps with ROPE $\rightarrow$ result is inconclusive
- Decide ROPE before analysis based on domain knowledge
  - Picking it after analysis is like picking the p-value threshold after seeing the p-value

Loss Function: Motivation

- You need to **make decisions** based on inference
- For many problems, **decision cost is asymmetric**
  - E.g., cost of a bad decision $>$ or $<$ benefit of a good decision
  - E.g., vaccines may cause overreaction, but benefits outweigh risks
- To make the **best decision**, measure:
  - Benefits of a correct decision
  - Cost of a mistake
  - Decide trade-off between benefits and costs using a loss function
  - Use loss function for decisions
- Loss quantifies *"how bad is an estimation mistake?"*
  - Larger loss indicates worse estimation

Loss Function

- Aka "cost function"
  - The inverse is known as "objective", "fitness", "utility function"
- Use a function to measure the difference between:
  - The true value θ; and
  - The estimated value $\hat{\theta}$

Loss	Expression	Point estimate
Quadratic loss	$(\theta - \hat{\theta})^2$	Mean of posterior
Absolute loss	$\lvert\theta - \hat{\theta}\rvert$	Median of posterior
1-0 loss	$I(\theta \neq \hat{\theta})$	Mode of posterior

- Algorithm to make decisions in Bayesian statistics using loss function
  - Goal: pick a single value $\hat{\theta}$
  - You don't know the true value θ
  - Estimate θ in terms of the posterior distribution
  - Find the value $\hat{\theta}$ that minimizes the expected loss function

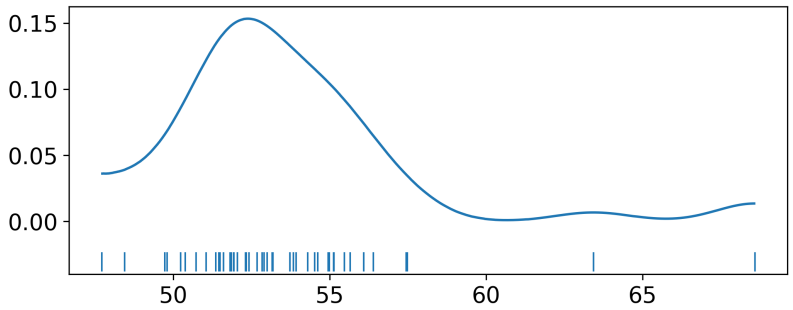
Tutorial: Bayesian Coin

- Bayesian Coin

- Concepts
- Coin Example
- Posterior-Based Decisions
  - ***Chemical Shift: Example***
  - Posterior Predictive Checks
- Groups Comparison
- Hierarchical Models
- Simple Linear Model
- Multiple linear regression
- Comparing Models
- Inference Engines

Chemical Shift

- Nuclear magnetic resonance (NMR)
  - Used to study molecules of living things
  - Measures observable quantities related to unobservable molecular properties (e.g., chemical shift)
- Data looks Gaussian with a couple of outliers

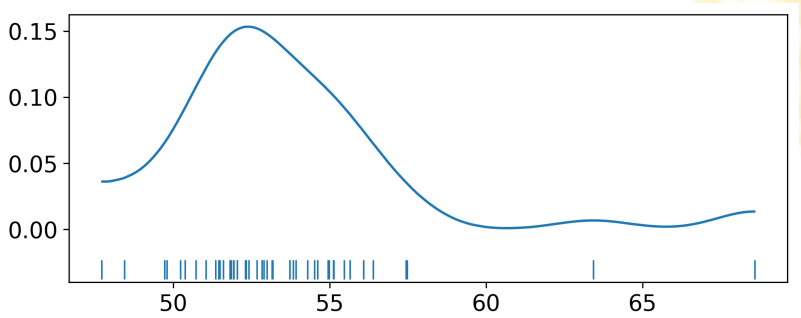


Use of Gaussians in Statistics

- Aka "normal" distribution
- **Gaussians are easy to work with and abundant in nature**
- **Pros**:
  - Average of large sample size tends to be Gaussian (by Central Limit Theorem)
  - Many phenomena approximated using Gaussians (since they are average of effects)
  - Conjugate prior of Gaussian is Gaussian
- **Cons**:
  - Not robust to outliers
  - Important to relax assumption of Gaussianity

Chemical Shift: Example

- Assume Gaussian is a decent approximation of the data
- The **likelihood** $y|\mu, \sigma$ comes from a normal distribution:



$$Y \sim N(\mu, \sigma)$$

- **Priors** for mean and sigma of Y
  - **Mean** from uniform distribution $U(l, h)$:

$$\mu \sim U(l, h)$$

    - Set μ larger than data range, e.g., $[40, 70]$
  - Std dev from half-normal distribution (i.e., a regular normal, but restricted to non-negative values):

$$\sigma \sim HalfNormal(0, \sigma_\sigma)$$

    - If unknown, set $\sigma_\sigma = 10$

Chemical Shift: PyMC

[87]:
```
with pm.Model() as model_g:
    # The mean is Uniform in [40, 70] (which is larger than the data).
    mu = pm.Uniform("mu", lower=40, upper=70)
    # The std dev is half normal with a large value (which is a large value based on the data).
    sigma = pm.HalfNormal("sigma", sigma=10)
    # The model is N(mu, sigma).
    y = pm.Normal("y", mu=mu, sigma=sigma, observed=data)
    # Sample.
    idata_g = pm.sample(1000)
```

```
Auto-assigning NUTS sampler...
Initializing NUTS using jitter+adapt_diag...
Multiprocess sampling (4 chains in 4 jobs)
NUTS: [mu, sigma]
```

```
Sampling 4 chains, 0 divergences ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 100% 0:00:00 / 0:00:00
```

```
Sampling 4 chains for 1_000 tune and 1_000 draw iterations (4_000 + 4_000 draws total) took 1 seconds.
```

- The PyMC code is **one-to-one with the model**:

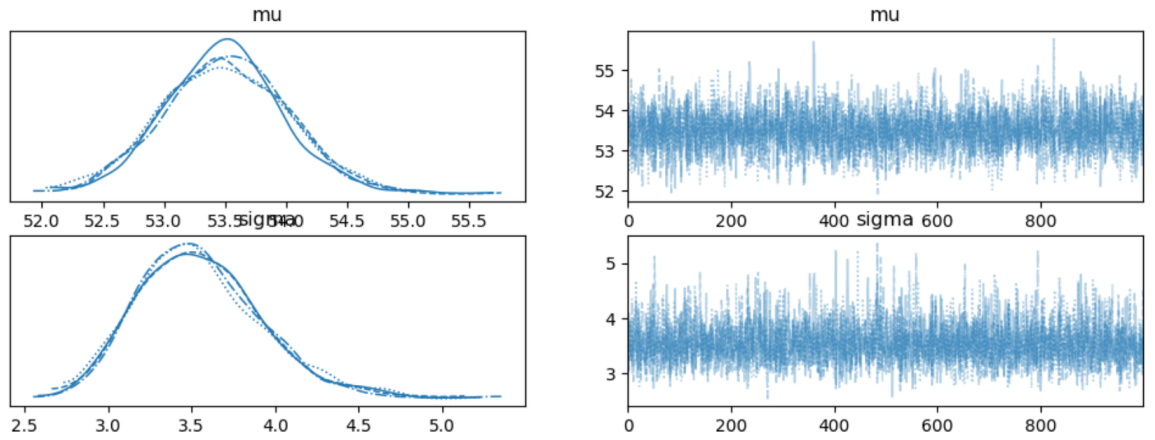
$$\begin{cases} \mu \sim U(l = 40, h = 70) \\ \sigma \sim HalfNormal(0, \sigma_\sigma = 10) \\ Y \sim N(\mu, \sigma) \end{cases}$$

  - Get 1000 samples from the posterior

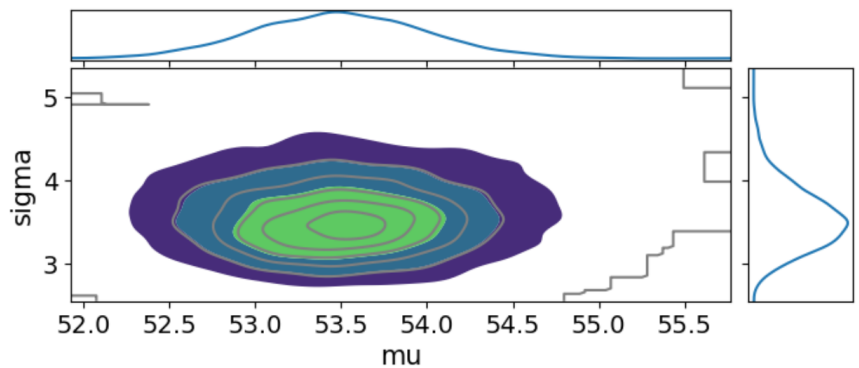
Chemical Shift: PyMC

- Compute 4 traces for 2 variables μ, σ
- Results are well-formed
- $\mu \in [52.55, 54.45]$
- $\sigma \in [2.86, 4.23]$
- Is the model good?

[92]:
```
# There are 4 traces for 2 variables.
az.plot_trace(idata_g);
```



[93]:
```
# The posterior distribution of the params is bi-dimensional, since it has mu and sigma.
az.plot_pair(idata_g, kind='kde', marginals=True);
```



[94]:
```
# Report a summary of the inference.
az.summary(idata_g, kind="stats").round(2)
```

[94]:

	mean	sd	hdi_3%	hdi_97%
mu	53.50	0.51	52.55	54.46
sigma	3.55	0.38	2.86	4.23

- Concepts
- Coin Example
- Posterior-Based Decisions
  - Chemical Shift: Example
  - ***Posterior Predictive Checks***
- Groups Comparison
- Hierarchical Models
- Simple Linear Model
- Multiple linear regression
- Comparing Models
- Inference Engines

Samples from Posterior Distribution

- Given a posterior distribution $\Pr(\theta|y)$, you can generate predictions $\tilde{y}$ based on the data y and the estimated parameters $\hat{\theta}$:

$$\Pr(\tilde{y}|y) = \int_{\theta} \Pr(\tilde{y}|\theta) \Pr(\theta|y) d\theta = \int \text{model} \times \text{posterior}$$

- This is called the "**posterior predictive distribution**" as it predicts future data using the posterior distribution
- Conceptually:
  - Sample a value of θ from the posterior $\Pr(\theta|y)$
  - Feed the value of θ to the likelihood $\Pr(y|\theta)$
  - Obtain $\tilde{y}$
- This process has two sources of uncertainty:
  - Parameter uncertainty
    - Captured by the posterior $\Pr(\theta|y)$
  - Sampling uncertainty
    - Captured by the likelihood $\Pr(y|\theta)$

Posterior Predictive Check (PPC)

- **Intuition**: can the model reproduce observed data?
- **PPC approach**:
  - Generate predictions $\tilde{y}$ with observed data y
  - Check consistency between predicted values and observed data
- Models should always be checked
- Differences can arise by:
  - Mistakes
  - Limitations of the model
    - E.g., the model works well for average behavior but fails to predict rare values
  - Limitations of the data

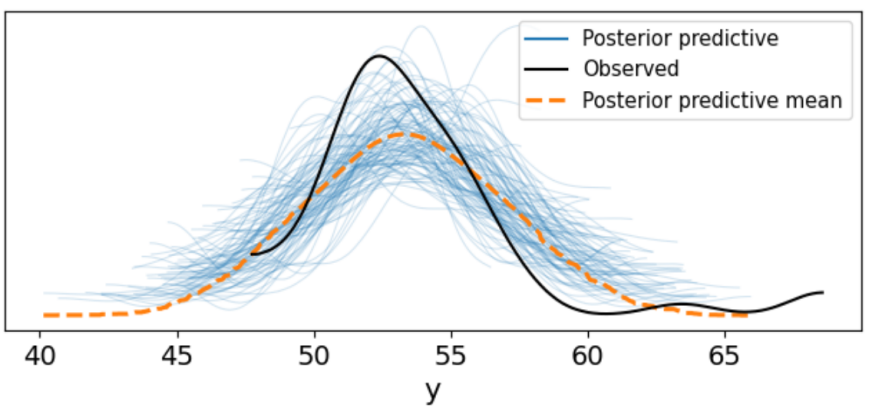
Bayesian Workflow Using PPC

1. Given a process
   - True distribution of the process is unknown / unknowable
2. **Sample the process**
   - Get finite sample y through sampling
   - E.g., experiment, survey, simulation
3. **Inference**
   - Build probabilistic model using prior $\Pr(\theta)$ and likelihood $\Pr(y|\theta)$ to get posterior distribution $\Pr(\theta|y)$
   - Posterior distribution: distribution of model parameters θ given data
4. **Predictive distribution**
   - Compute predictions from posterior distribution (i.e., posterior predictive distribution)
   - Posterior predictive distribution is the distribution of predicted samples averaged over posterior distribution
5. **Validation**
   - Validate model by comparing original samples vs predicted samples

Chemical Shift Example: PPC

```
[95]: # Compute 100 posterior predictive samples.
      y_pred_g = pm.sample_posterior_predictive(idata_g, model=model_g)

      Sampling: [y]

         Sampling ... ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 100% 0:00:00 / 0:00:00
```

```
[96]: # Black: KDE of the data (observed)
      # Blue: KDEs of the posterior predictive samples
      # Orange: KDE of the posterior predictive mean
      az.plot_ppc(y_pred_g, mean=True, num_pp_samples=100);
```

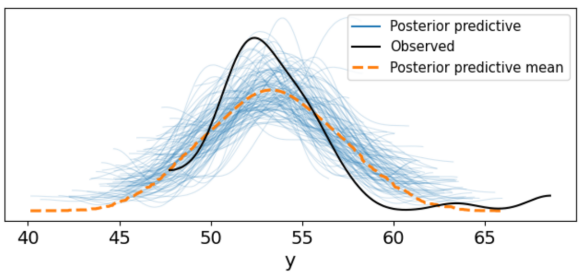


- Sample the posterior
- Apply the model
- Get predictive posterior distribution (dashed orange distribution)
- Compare to data (black distribution)
- **Is the PPC model good?**
- **No**
  - Posterior mean is more to the right than data
  - Posterior std dev is larger

Chemical Shift: Model Critique

```
[95]: # Compute 100 posterior predictive samples.
      y_pred_g = pm.sample_posterior_predictive(idata_g, model=model_g)

      Sampling: [y]

         Sampling ... ——————————————————————————— 100% 0:00:00 / 0:00:00
```

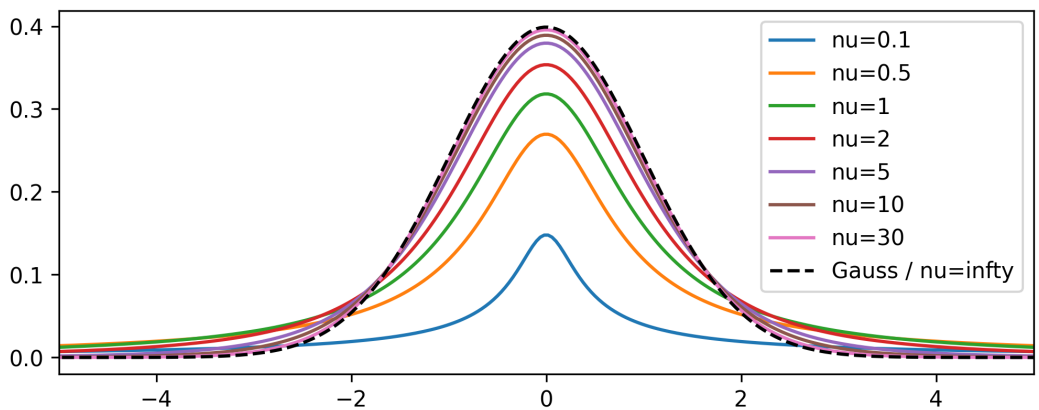
```
[96]: # Black: KDE of the data (observed)
      # Blue: KDEs of the posterior predictive samples
      # Orange: KDE of the posterior predictive mean
      az.plot_ppc(y_pred_g, mean=True, num_pp_samples=100);
```



- **Problem**
  - Two data points on the tails of the distribution
  - The normal distribution is
    - "surprised" by these points
    - "reacts" by adjusting the mean towards them and increasing the standard deviation
- **Solution**
  - Declare points as outliers and discard
    - E.g., equipment malfunction (you need evidence!)
  - Change the model
- **Bayesian philosophy**
  - Encode assumptions into the model (e.g., priors, likelihoods)
  - Instead of ad-hoc heuristics (e.g., outlier removal rules)

Student's t-distribution: Recap

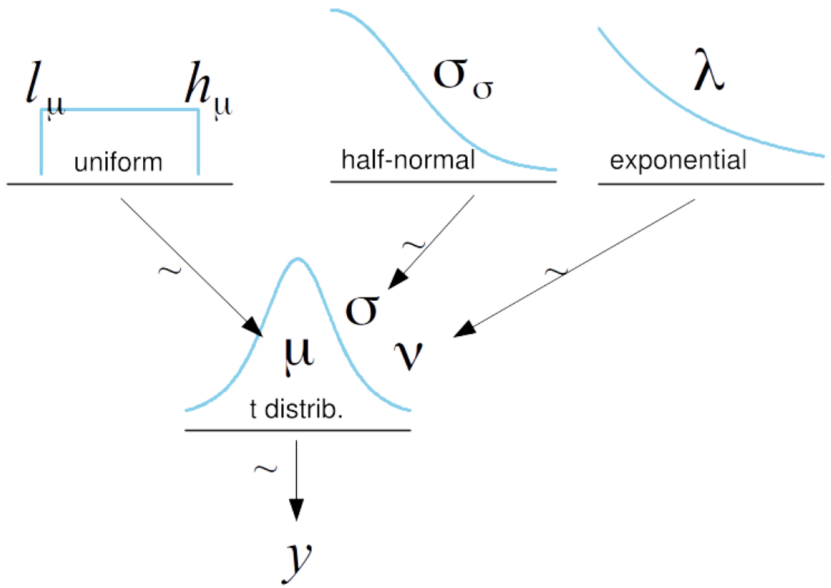
- **Student's t-distribution** has 3 params:
  1. Mean μ
     - It doesn't always exist
  2. Scale σ
     - Similar to std dev, but it doesn't always exist
  3. Degrees of freedom $\nu \in [0, \infty]$
     - Aka "normality parameter", since it controls how "normal" is the distribution
     - With $\nu = 1$: heavy tails and no mean (Cauchy)
     - With $\nu \to \infty$ we recover the Gaussian
- Student's t has **heavy tails**
  - Heavy tails (high kurtosis) means *"values are more likely to be far from the mean compared to a Normal"*



Chemical Shift: Use Student's t-dist (1/3)

- Use Student's t-distribution model instead of Normal

$$\begin{cases} \mu \sim U(l, h) \\ \sigma \sim HalfNornal(0, \sigma) \\ \nu \sim Exp(\lambda) \\ y \sim StudentT(\mu, \sigma, \nu) \end{cases}$$

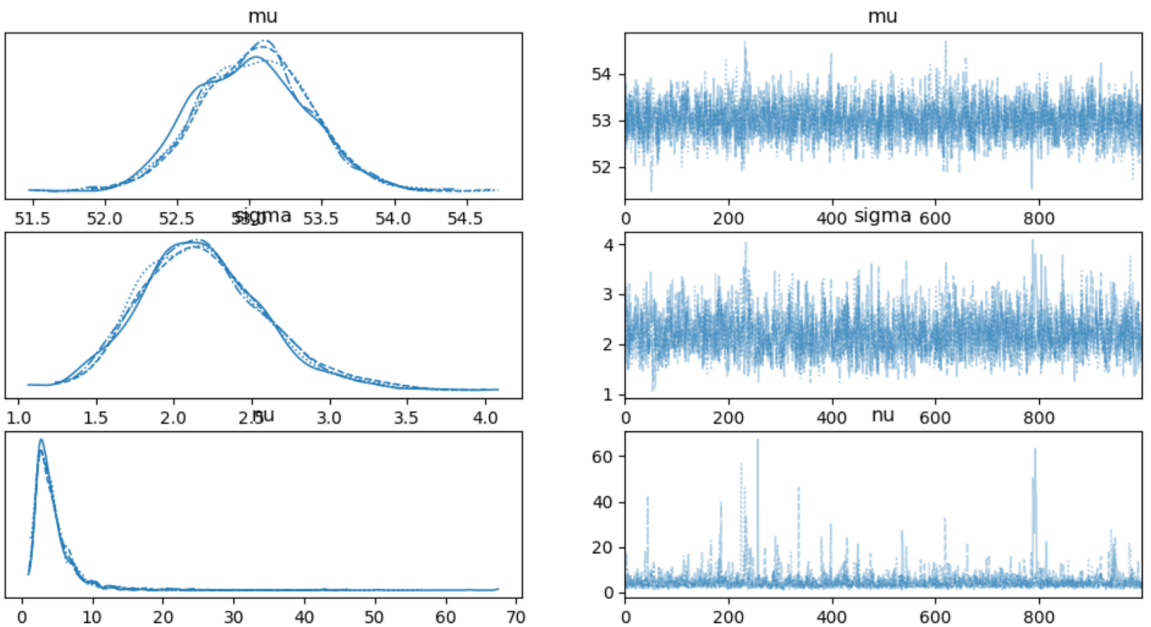


Chemical Shift: Use Student's t-dist (2/3)

[102]:
```
# Use a Student-T model.
with pm.Model() as model_t:
    mu = pm.Uniform("mu", 40, 75)
    sigma = pm.HalfNormal("sigma", sigma=10)
    # A student with nu = 30 is close to a Gaussian.
    nu = pm.Exponential("nu", 1/30)
    #
    y = pm.StudentT("y", mu=mu, sigma=sigma, nu=nu, observed=data)
    idata_t = pm.sample(1_000)
```

```
Auto-assigning NUTS sampler... •••
```

```
az.plot_trace(idata_t); •••
```



[104]:
```
az.summary(idata_t, kind="stats").round(2)
```

[104]:

	mean	sd	hdi_3%	hdi_97%
mu	53.03	0.38	52.35	53.76
sigma	2.19	0.40	1.45	2.95
nu	4.65	3.92	1.20	9.20

- Outliers decrease ν (less Gaussian) instead of increasing mean and standard deviation
  - μ similar to Gaussian estimate
  - σ smaller
  - $\nu \approx 5$ (not very Gaussian)
- Estimation more robust
  - Outliers have less effect

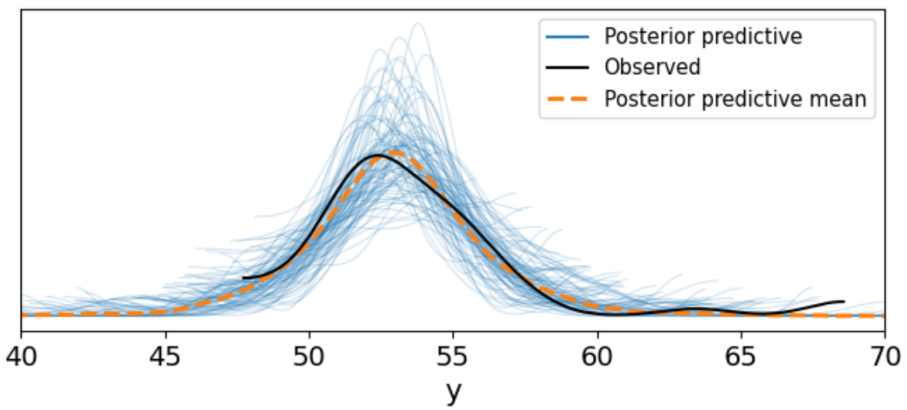
Chemical Shift: Use Student's t-dist (3/3)

```
[105]: # Compute 100 posterior predictive samples.
       y_ppc_t = pm.sample_posterior_predictive(idata_t, model_t);
```

```
Sampling: [y]
```

```
Sampling ... ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 100% 0:00:00 / 0:00:00
```

```
[106]: ax = az.plot_ppc(y_ppc_t, num_pp_samples=100, mean=True)
       ax.set_xlim(40, 70);
```



- PPC (posterior predictive check) fits better than Normal model
- Plot is "hairy" because KDE is estimated only in data interval and 0 outside

Tutorial: Robust Modeling

- Bayesian Coin

- Concepts
- Coin Example
- Posterior-Based Decisions
- ***Groups Comparison***
- Hierarchical Models
- Simple Linear Model
- Multiple linear regression
- Comparing Models
- Inference Engines

Group Comparison

- **Group comparison** tests for statistically significant results between *"treatment"* and *"control group"*
- E.g.,
  - *How well do patients respond to a new drug vs a placebo?*
  - *Is there a reduction in car accidents after new traffic regulation?*
  - *Does college student performance improve without cellphones at school?*
- **Effect size** quantifies the difference between two groups
  - From *"does it work?"* (hypothesis testing) to *"how well does it work?"* (estimate effect size)

Bogus Control Groups

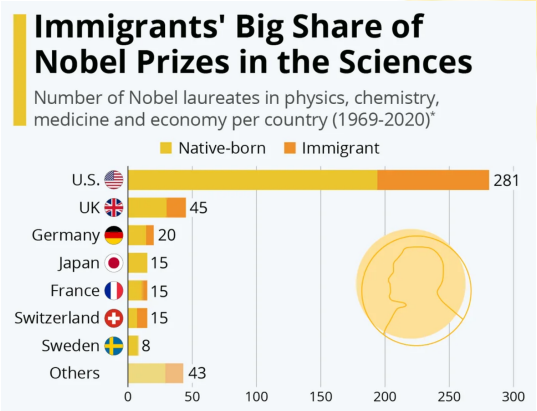
- When something is claimed to be harder/better/faster/stronger, **ask for the baseline used for comparison**
  - E.g.,
    - Sell sugary yogurts to boost the immune system by comparing it to using milk
    - A better control group would be a less sugary yogurt
- **Placebo** is a psychological phenomenon where a patient experiences improvements after receiving an inactive treatment
  - Using a placebo is better than "no treatment"
  - Shows difficulty in accounting for all factors in an experiment

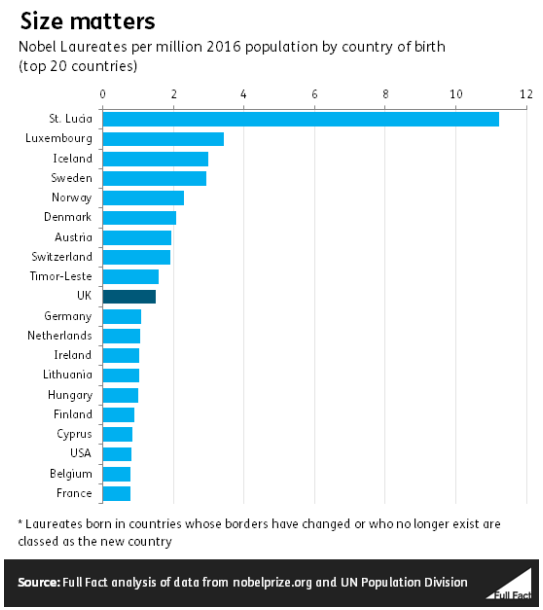
Group Comparison Bayesian-Style

- **Frequentist approach**
  - Compare p-value of difference of means in each group
- **Bayesian approach**
  - Compare posterior distribution of means between groups using:
    - Plot of posterior
    - Cohen's d
    - Probability of superiority

Sample Size Effect

- **Sample size effect** is the impact of the number of observations on statistical results (e.g., p-values, confidence intervals)
- **Small sample effect**
  - Large mean difference might not be statistically significant (low p-value)
  - Estimates (e.g., means, proportions, correlations) fluctuate widely due to high sampling variability
  - Outliers have a disproportionate influence
  - Inference is unstable: results may not replicate
- **Large sample**:
  - Tiny mean difference can be highly significant (small p-value) but meaningless
- E.g., Cohen's d, probability of superiority



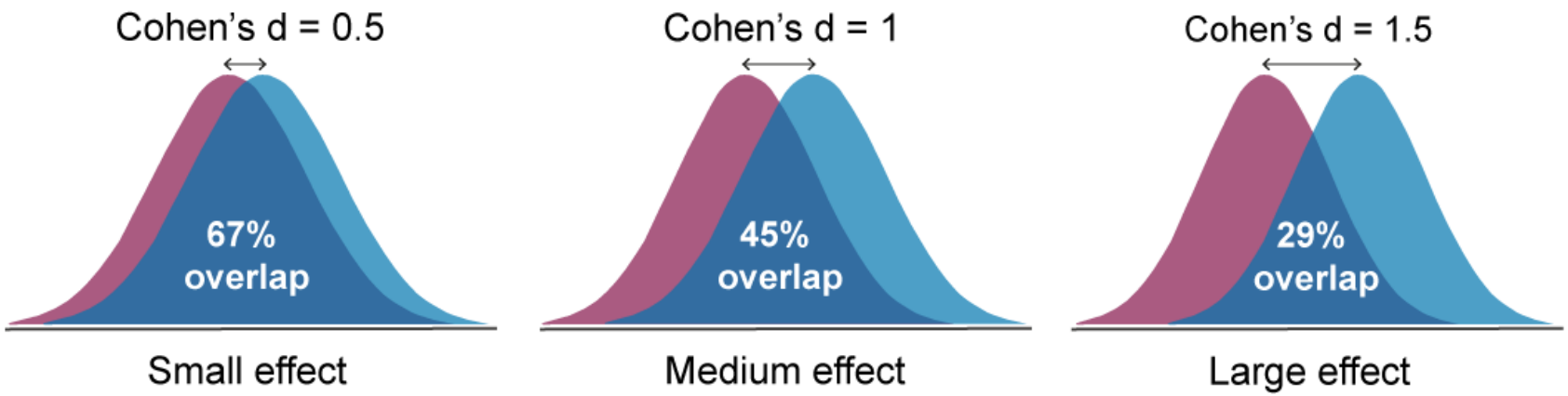


Cohen's d

- **Cohen's d** is the difference of means relative to pooled standard deviation

$$\frac{\mu_2 - \mu_1}{\sqrt{(\sigma_1^2 + \sigma_2^2)/2}}$$

  - Normalizes effect by variability for pooled std dev
  - Variability of each group normalizes mean difference
  - Similar to a Z-score, number of std dev values differ



- Bayesian approach
  - Compute posterior distribution of means and std → formula
  - Compute distribution of Cohen's d → summary statistics

- Concepts
- Coin Example
- Posterior-Based Decisions
- Groups Comparison
- ***Hierarchical Models***
- Simple Linear Model
- Multiple linear regression
- Comparing Models
- Inference Engines

Hierarchical Models

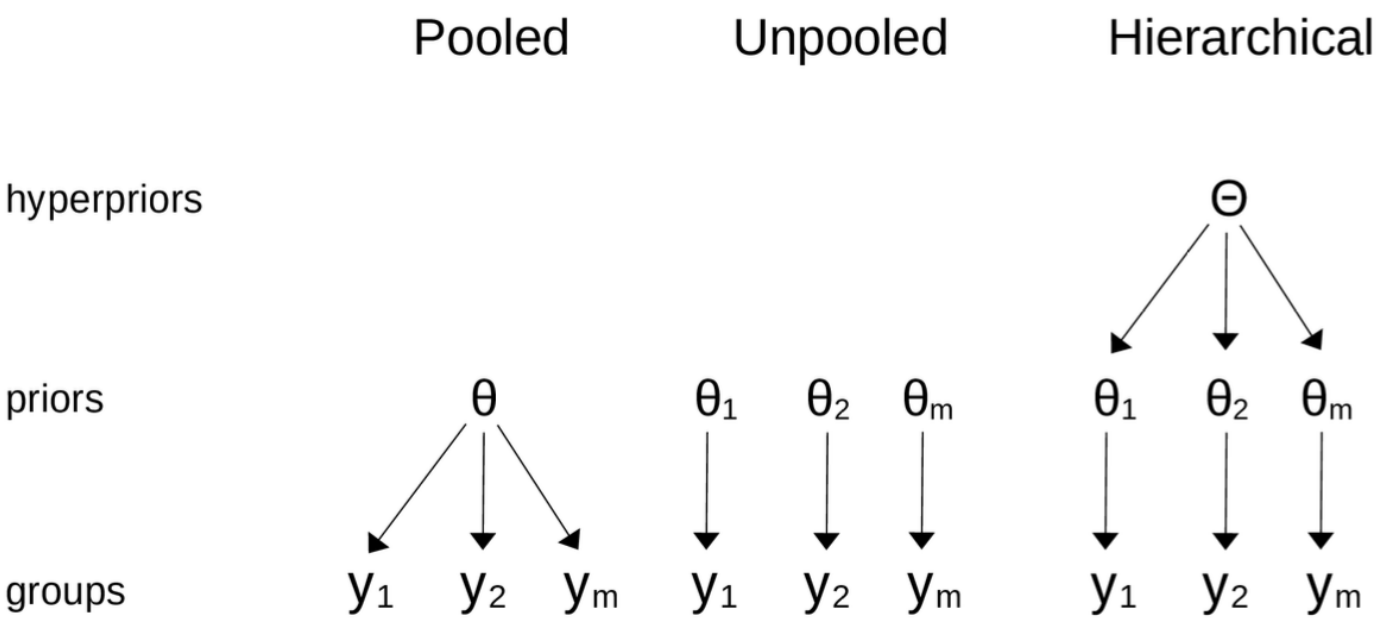
- Aka "multilevel", "nested", "mixed-effects" models
- **Key observation**: often data points share common structure, but also have variations
  - **Group data**
    - E.g., sales in cities: each city is a different market, but there are common trends
  - **Hierarchical structure**
    - E.g., students in a school: each student is different, with common educational factors
  - **Repeated measurements on same objects**
- **Approach**: 💡
  - Model shares information between groups, but it allows differences
  - Parameters of prior distributions have a prior distribution
    - Aka "hyper-priors" (!)
- You can't do this with frequentist approach, only with Bayesian approach

Hierarchical Models: Examples

- Many data problems lend themselves to **hierarchical descriptions**
- E.g.,
  - Medical research:
    - Estimate drug effectiveness in treating a disease
    - Categorize patients by demographics, disease severity
    - Estimate cure probability for each subgroup
  - Market research
    - Understand consumer purchasing behavior
    - Categorize consumers by age, gender, income, education

Unpooled, Pooled, Hierarchical Models

- **Pooled**
  - Groups have the same priors
- **Unpooled**
  - Groups have different priors
- **Hierarchical**
  - Groups have different priors which come from a common prior



Hierarchical Models: Chemical Shift

- Proteins are made of 20 amino acids
  - Study proteins with nuclear magnetic resonance
  - Measure "chemical shift"
- The data looks like:

	ID	aa	theo	exp	diff
0	1BM8	ILE	61.18	58.27	2.91
1	1BM8	TYR	56.95	56.18	0.77
2	1BM8	SER	56.35	56.84	-0.49
3	1BM8	ALA	51.96	51.01	0.95
4	1BM8	ARG	56.54	54.64	1.90
...	...	...	...	...	...
1771	1KS9	LYS	55.79	57.51	-1.72
1772	1KS9	ARG	58.91	59.02	-0.11
1773	1KS9	LYS	59.49	58.92	0.57
1774	1KS9	GLU	59.48	58.36	1.12
1775	1KS9	SER	58.07	60.55	-2.48

1776 rows × 5 columns

- `ID`: Code of the protein
- `aa`: Name of the amino acid
- `theo`: Theoretical values of chemical shift
- `exp`: Experimental value
- `diff`: Difference between theoretical and experimental value

Hierarchical Models: Chemical Shift

- Experimental measures of chemical shifts vs theoretical values
- Evaluate model using different modeling styles
  1. **Pooled**
     - Compute difference between estimates and measures, fit Gaussian
     - More accurate estimates, lose amino acid information
  2. **Unpooled**
     - Fit 20 Gaussians for 20 amino acids
     - Detailed analysis, less accuracy
  3. **Hierarchical**
     - Model groups assuming common population

Chemical Shift: Unpooled Model

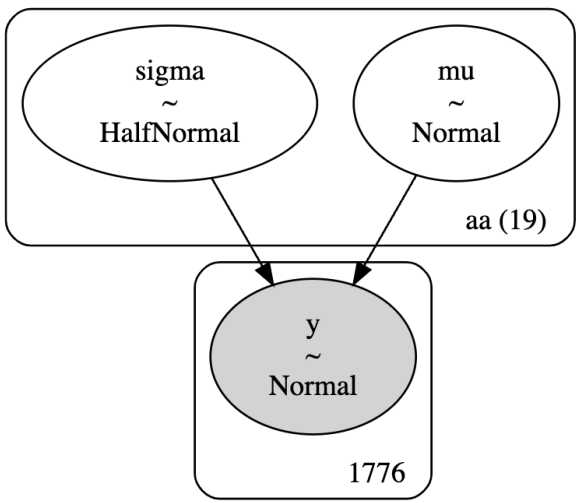
In [9]:
```
# Non-hierarchical model.
with pm.Model(coords=coords) as cs_nh:
    # One separate prior for each group.
    mu = pm.Normal('mu', mu=0, sigma=10, dims="aa")
    sigma = pm.HalfNormal("sigma", sigma=10, dims="aa")
    # Likelihood.
    y = pm.Normal("y", mu=mu[idx], sigma=sigma[idx], observed=diff)
    idata_cs_nh = pm.sample()
```

```
Auto-assigning NUTS sampler...
Initializing NUTS using jitter+adapt_diag...
Multiprocess sampling (4 chains in 4 jobs)
NUTS: [mu, sigma]
Output()
```

```
Sampling 4 chains for 1_000 tune and 1_000 draw iterations (4_000 + 4_
```

In [10]:
```
pm.model_to_graphviz(cs_nh)
```

Out[10]:



- **Model each group independently**
  - Use same model structure to compare groups

Chemical Shift: Hierarchical Model

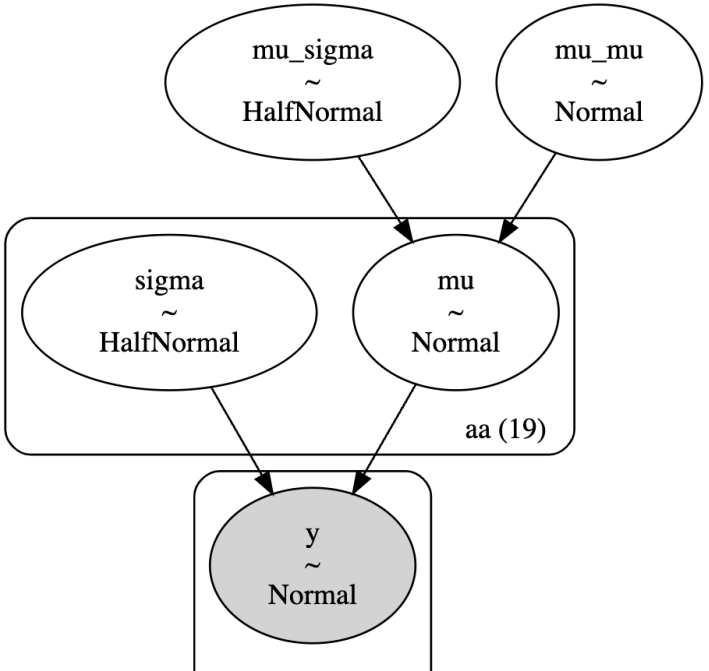
In [12]:
```
with pm.Model(coords=coords) as cs_h:
    # Hyper-priors.
    mu_mu = pm.Normal("mu_mu", mu=0, sigma=10)
    mu_sigma = pm.HalfNormal("mu_sigma", sigma=10)

    # Priors.
    mu = pm.Normal("mu", mu=mu_mu, sigma=mu_sigma, dims="aa")
    sigma = pm.HalfNormal("sigma", sigma=10, dims="aa")

    # Likelihood (same as before).
    y = pm.Normal("y", mu=mu[idx], sigma=sigma[idx], observed=diff)
    idata_cs_h = pm.sample()
```

```
Auto-assigning NUTS sampler...
Initializing NUTS using jitter+adapt_diag...
Multiprocess sampling (4 chains in 4 jobs)
NUTS: [mu_mu, mu_sigma, mu, sigma]
Output()
```

```
Sampling 4 chains for 1_000 tune and 1_000 draw iterations (4_000 + 4_
```

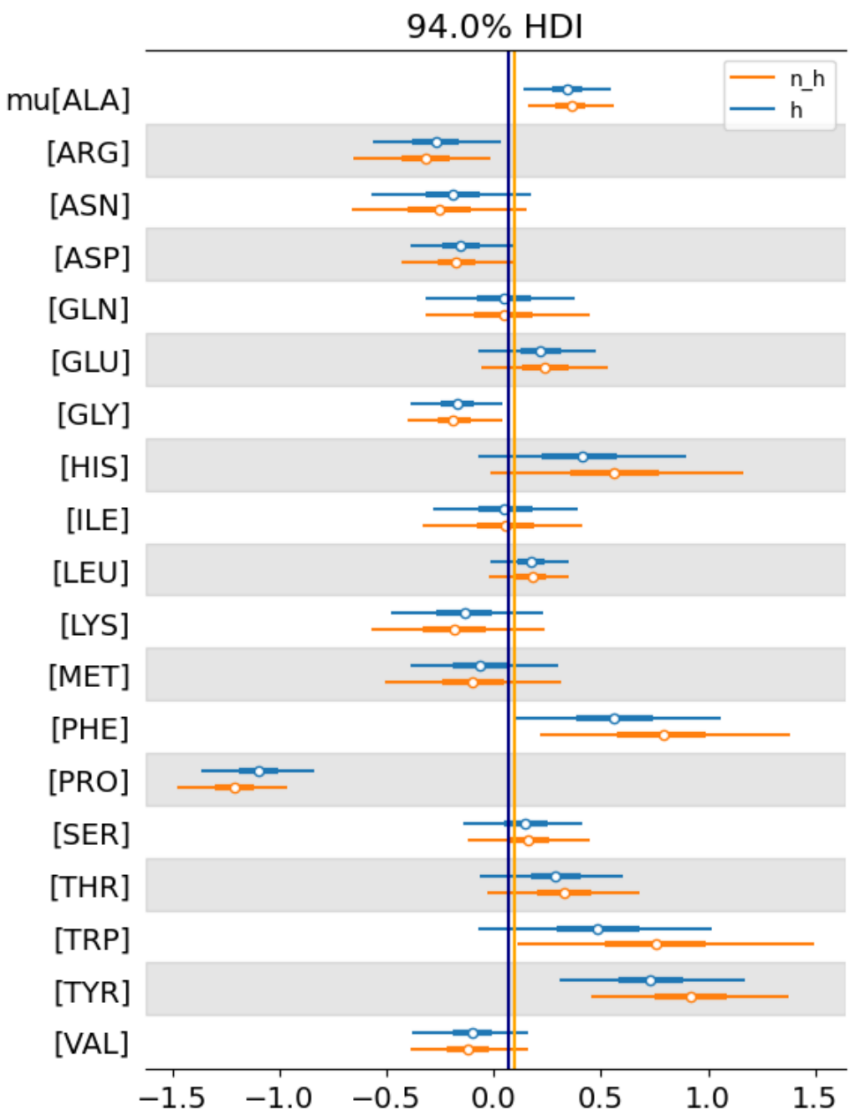
In [13]:
```
pm.model_to_graphviz(cs_h)
```

Out[13]:



- **Add two hyperpriors on μ**
  - One for mean of μ
  - One for standard deviation of μ
- **Assume same variance σ for all groups**
  - Modeling choice
  - Option to add hyperpriors for σ
- Intermediate situation between single group and 20 separate groups

Chemical Shift: Results



- Compare estimates of two models
  - 20 groups
  - Each model has 4 estimated variable
- Plot 94% credible intervals
  - Blue is hierarchical
  - Orange is non-hierarchical
  - Black vertical line: global mean (hierarchical model)
  - Blue / hierarchical means pulled towards mean compared to orange (non-hierarchical)
  - **There is shrinkage**

Shrinkage

- **Hierarchical models shrink parameters towards a common mean**
  - As groups share information through the hyper-prior
  - Model groups as neither independent nor a single group, but in between
  - Less responsive to extreme values in individual groups
  - Improves estimation for small groups using data from others
- **Amount of shrinkage** depends on data:
  - Groups with more data influence estimates more
  - Similar groups reinforce common estimation
  - It's a global optimization!
- **Result**: inference is more stable

You Need to Know When to Stop

- You can create hierarchical models with as many levels as you want
- More levels:
  - Don't improve inference quality
  - Complicate interpretation
- Goal:
  - Leverage data structure
  - *"Add as many degrees as freedom as needed, but not more than what is warranted"* (Occam's razor)

- Concepts
- Coin Example
- Posterior-Based Decisions
- Groups Comparison
- Hierarchical Models
- ***Simple Linear Model***
  - Logistic Regression
- Multiple linear regression
- Comparing Models
- Inference Engines

Linear Model

- Many problems can be stated as:
  - X and Y: uni-dimensional continuous RVs
  - Dataset of paired observations $\{(x_1, y_1), ..., (x_n, y_n)\}$
  - X are independent variables
  - Model/predict dependent variable Y
  - Assume a linear relationship between Y and X
- E.g.,
  - Average temperature vs Nobel laureates in a country
  - Sugar intake vs Blood glucose
  - Years of education vs Annual income
  - Advertising spend vs Sales
  - Study hours vs Exam score
  - Coffee consumption vs Productivity

Linear Model: Frequentist Approach

- A **linear model** is described by:

$$y = \alpha + \beta x$$

- **Frequentist approach**:
  - Find parameters α, β using least square fitting
    - Minimize average quadratic error between observed y and predicted $\hat{y}$
  - Point-estimate from least squares corresponds to maximum a-posteriori with flat priors

Linear Model: Bayesian Approach

- A **linear model** is described by:

$$y = \alpha + \beta x$$

- **Bayesian approach assumes**:
  - Data is Gaussian with mean $\alpha + \beta x$ and standard deviation σ: $y \sim N(\mu = \alpha + \beta x, \sigma)$
  - α, β, σ are independent
  - Priors:

$$\begin{cases} \alpha \sim Normal(\mu_\alpha, \sigma_\alpha) \\ \beta \sim Normal(\mu_\beta, \sigma_\beta) \\ \sigma \sim HalfNormal(0, \sigma_\epsilon) \end{cases}$$

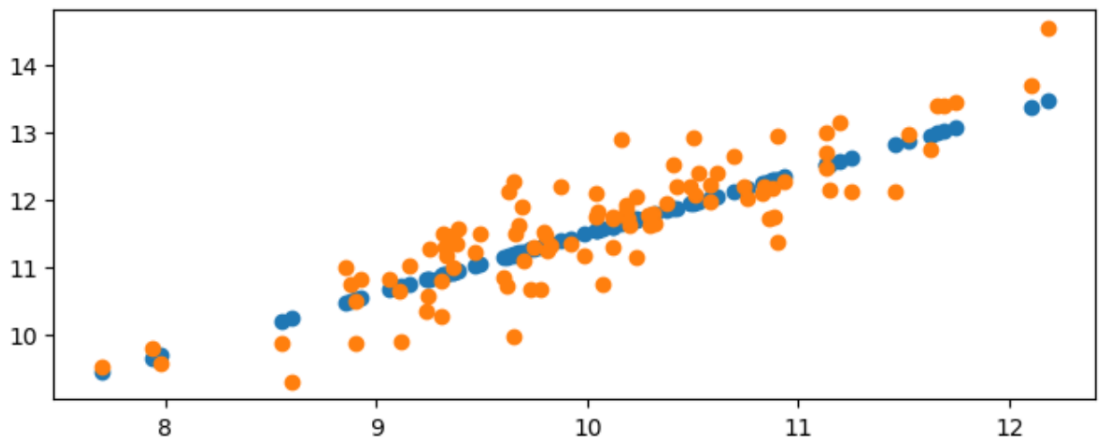
- Use **vague priors**:
  - Prior of intercept α often centered around 0
  - Info on sign of β sometimes available
  - Half-Cauchy / exponential distribution for σ
  - Uniform prior if parameter has hard boundaries

Linear Model: Synthetic Example

- Generate random data from ground truth
  - $\alpha = 2.5$
  - $\beta = 0.9$
  - Add noise

[122]:
```
np.random.seed(1)

# Number of samples.
N = 100

# Parameters.
alpha_real = 2.5
beta_real = 0.9
sigma_eps_real = 0.5

# Generate data.
x = np.random.normal(10, 1, N)
y_real = alpha_real + beta_real * x

# Add noise.
eps_real = np.random.normal(0, sigma_eps_real, size=N)
y = y_real + eps_real
```

[123]:
```
plt.scatter(x, y_real)
plt.scatter(x, y);
```



Linear Model: Synthetic Example

- Create linear model in PyMC
- Use vague priors
  - Prior of intercept α and β centered around 0
  - Half-Cauchy distribution for σ

[125]:
```
with pm.Model() as model_g:
    alpha = pm.Normal("alpha", mu=0, sigma=10)
    beta = pm.Normal("beta", mu=0, sigma=1)
    sigma = pm.HalfCauchy("sigma", 5)
    #
    mu = pm.Deterministic("mu", alpha + beta * x)
    y_pred = pm.Normal("y_pred", mu=mu, sigma=sigma, observed=y)
    idata_g = pm.sample(2000, tune=1000)
```

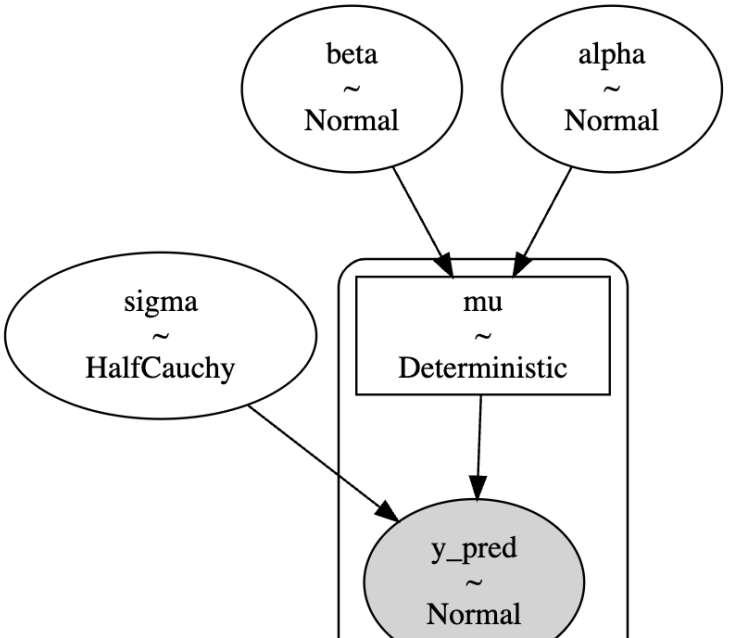
```
Auto-assigning NUTS sampler...
Initializing NUTS using jitter+adapt_diag...
Multiprocess sampling (4 chains in 4 jobs)
NUTS: [alpha, beta, sigma]
```

```
Sampling 4 chains, 0 divergences
```

```
Sampling 4 chains for 1_000 tune and 2_000 draw iterations (4_000 
```

[127]:
```
pm.model_to_graphviz(model_g)
```

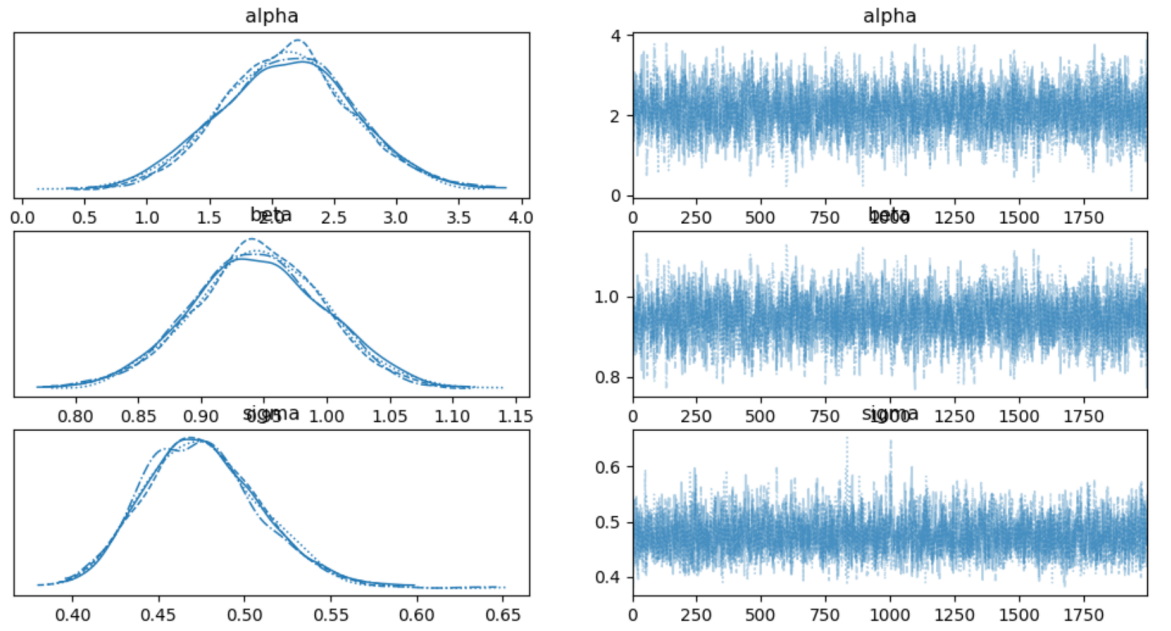
[127]:



Linear Model: Synthetic Example

- Run the sampler
- Ground truth
  - $\alpha = 2.5$
  - $\beta = 0.9$
- Recovered values
  - $\alpha = 2.12$
  - $\beta = 0.95$

[129]:
```
az.plot_trace(idata_g, var_names=["alpha", "beta", "sigma"]);
```



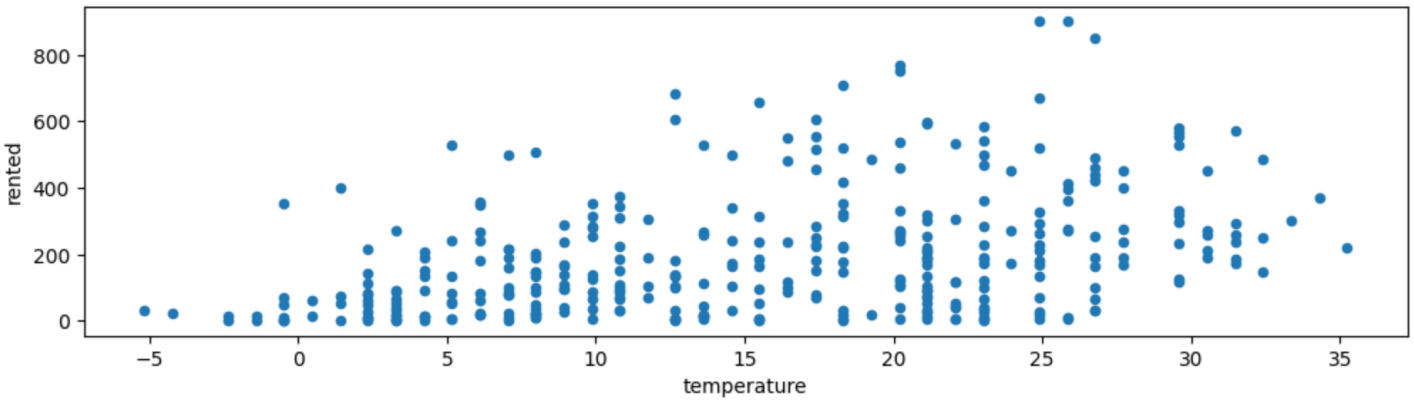
[128]:
```
az.summary(idata_g, var_names="alpha beta sigma".split(), kind="stats")
```

[128]:

	mean	sd	hdi_3%	hdi_97%
alpha	2.118	0.538	1.099	3.136
beta	0.946	0.053	0.845	1.046
sigma	0.476	0.034	0.412	0.538

Linear Model: Bike Rental Example

- Model relationship between temperature X and number of bikes rented Y

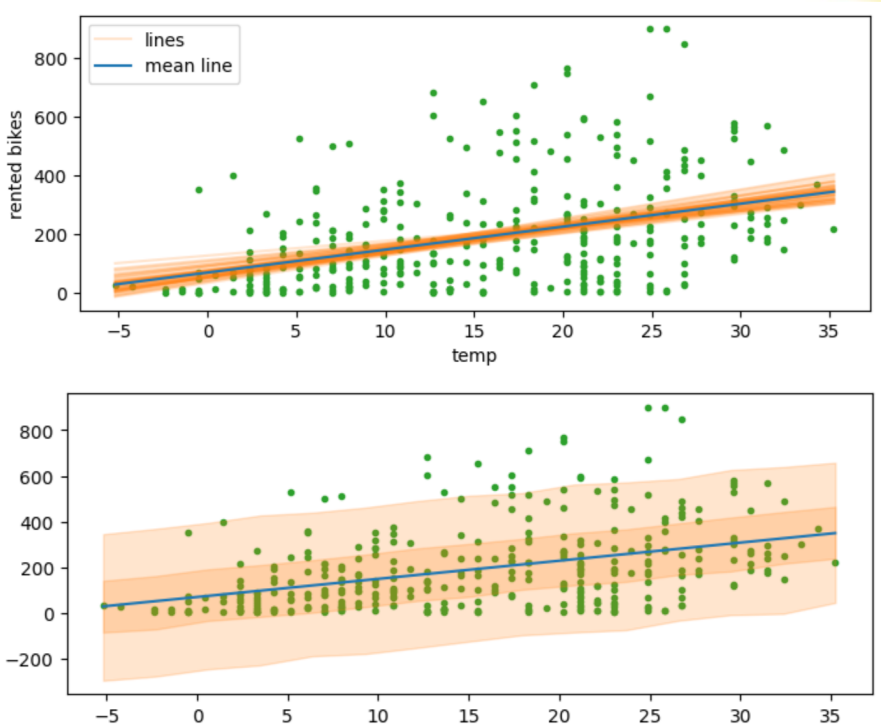


- Use intermediate variable $\mu = \alpha + \beta X$, mean number of bikes rented for temperature X

$$Y \sim N(\mu, \sigma)$$

Linear Model: Bike Rental Example

- Fit model: $\mu = 69 + 7.9X$
- Interpreting the model
  - For temperature 0 expected rented bikes $= 69$
  - Each degree increase: expected value increases by 7.9
- Parameters have uncertainty
  - Posterior accounts for combined uncertainty
  - Bands represent quantiles [0.25, 0.75] and [0.03, 0.97] of prediction



- **Do you see any problem in the model?** 🧩

Linear Model: Bike Rental Example (Criticism)

- **Problems**
  - Model outputs negative bike numbers
  - Model predicts real numbers, but count is discrete
- **Root cause**: Not surprising since using Normal likelihood:
  - Extends to negative numbers
  - Is continuous
- **Solutions**:
  - Clip predictions below 0 and discretize output (hack)
  - Use model defined for positive numbers only

Generalized Liner Model (GLM)

- Generalization of linear model using different distributions for likelihood:

$$\begin{cases} \alpha \sim \textit{prior} \\ \beta \sim \textit{prior} \\ \mu = \alpha + \beta X \\ \theta \sim \textit{prior} \\ Y \sim \phi(f(\mu), \theta) \end{cases}$$

where:

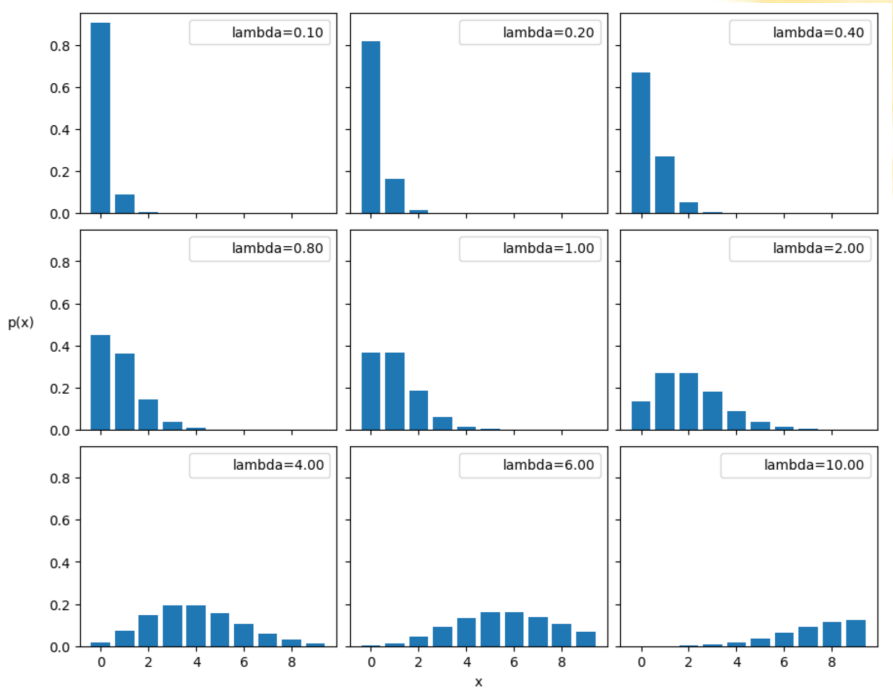
- ϕ: arbitrary distribution
  - E.g., Normal, Student's t, negative binomial
- θ: auxiliary parameter for ϕ
  - E.g., σ for Normal
- $f()$: "inverse link function" transforming μ from the real line to domain of ϕ

Count Data

- **Count data** = when the RV is a discrete number and bounded at 0
  - E.g., the number of rented bikes
- If the number is large, we can use a continuous distribution
  - E.g., Gaussian
- Other times, it's modeled as discrete
  - E.g., Poisson, negative binomial

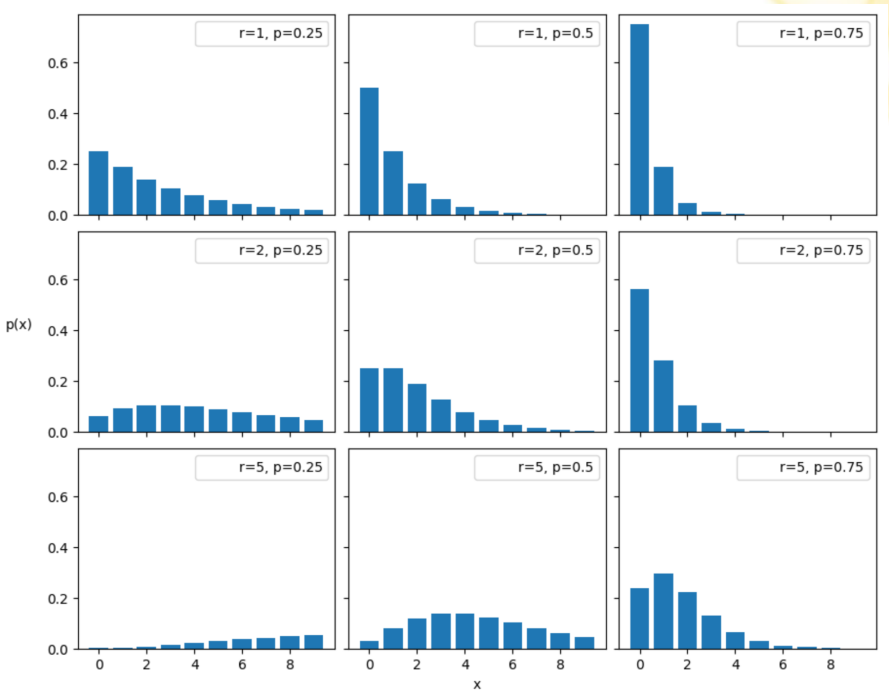
Poisson Distribution

- **Poisson distribution**
  - Models number of events in a fixed interval (e.g., time, space)
  - Assumes events happen independently at a constant average rate
- **E.g.,**
  - Call centers: number of customers per hour
  - Natural events: number of earthquakes in a region over time
  - Traffic flow: number of cars through a toll booth per day
  - Biology: count mutations in a DNA segment over time
- **Cons**:
  - Assumes mean equals variance (can't model "overdispersion")



Negative Binomial Distribution

- **Negative binomial distribution**
  - Models failures/trials to achieve fixed number of successes in IID Bernoulli trials
  - Generalizes geometric distribution, which models number of trials to achieve first success
  - Models overdispersion (i.e., mean « variance)
- **E.g.,**
  - Customer service: unsuccessful interactions before success
  - Sports: games lost before winning a fixed number of games

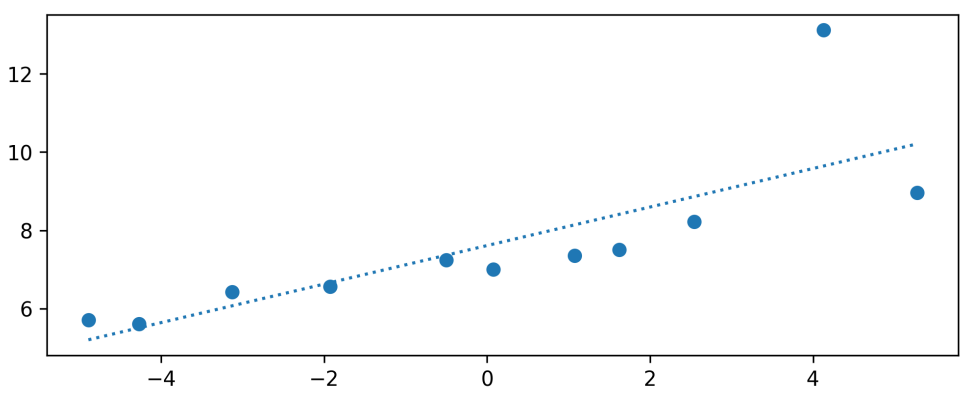


Overdispersion

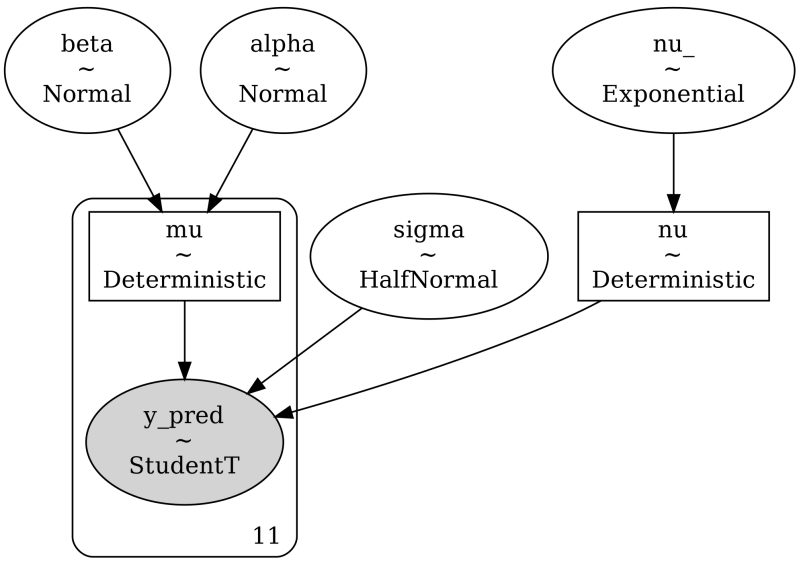
- **Overdispersion** occurs when variance of count data is much larger (e.g., 15-20x) than mean
- **E.g.,**
  - Epidemiology: when modeling disease outbreak, number of new infections on a given day might fluctuate widely
  - Customer service: large variation of number of daily complaints
- **How to model it?**
  - Poisson distribution:
    - Mean and variance equal ❌
  - Negative binomial:
    - Second parameter controls variance ✅
  - **Model using Normal**:
    - Mismatch when predicting negative rented bikes
    - Poor fit even on positive side
  - **Model using Negative Binomial**:
    - Better fit, though not perfect
    - Right tail predictions differ, but high demand probability is low
    - Overall better than Normal model

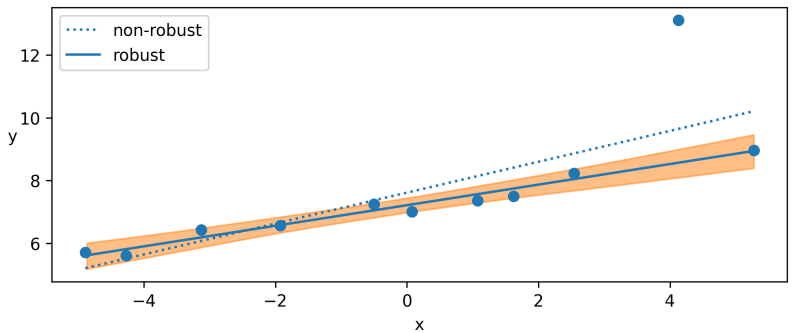
Robust Regression

- Outliers pull regression line away from bulk of data



- Student's t-distribution
  - Has heavier tails than Normal
  - Gives less importance to outliers





- Concepts
- Coin Example
- Posterior-Based Decisions
- Groups Comparison
- Hierarchical Models
- Simple Linear Model
  - ***Logistic Regression***
- Multiple linear regression
- Comparing Models
- Inference Engines

Logistic Regression

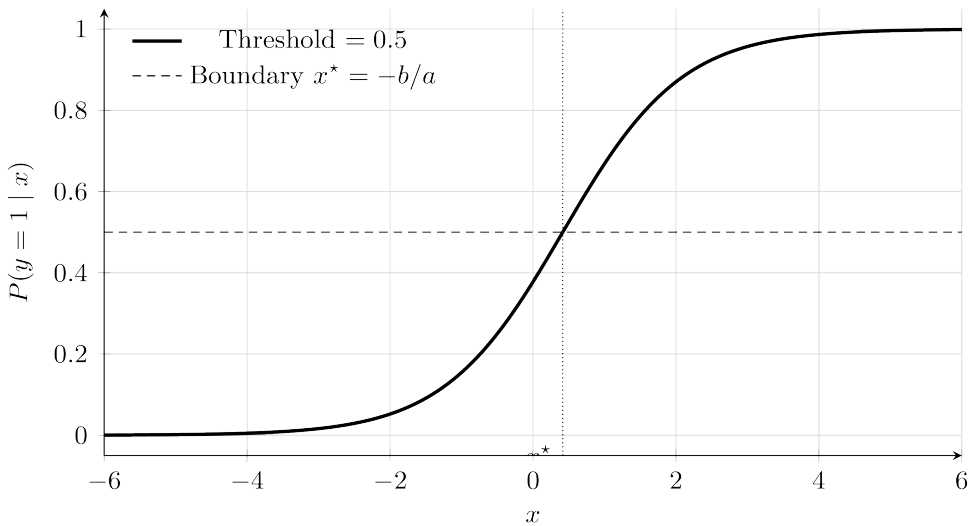
- Logistic regression model
  - Is a generalized linear model
  - Models the response variable as binary
    - E.g., ham/spam, safe/unsafe, cloudy/sunny, hotdog/not hotdog
- The logistic model is:

$$\begin{cases} \theta = \mathit{logit}(\alpha + \beta x) \\ y \sim \mathit{Bernoulli}(\theta) \end{cases}$$

- Logistic function (aka sigmoid)

$$\mathit{logit}(z) = \frac{1}{1 + \exp^{-z}}$$

  - Converts real numbers from θ to [0, 1] for the Bernoulli distribution



Iris Dataset

- Classical dataset of measurements from flowers from 3 closely related species of Iris setosa, virginica, and versicolor
  - E.g., we want to predict the probability of a flower being `setosa` from the `sepal_length`

	sepal_length	sepal_width	petal_length	petal_width	species
0	5.1	3.5	1.4	0.2	setosa
1	4.9	3.0	1.4	0.2	setosa
2	4.7	3.2	1.3	0.2	setosa
3	4.6	3.1	1.5	0.2	setosa
4	5.0	3.6	1.4	0.2	setosa

Classification with Logistic Regression

- **Logistic regression**:
  - Computes the probability that the output is a certain value
  - Models $\theta = \Pr(Y = 1|X)$
  - Classifies the output using a decision rule, e.g.,

$$\Pr(Y = \texttt{versicolor}|\texttt{sepal\_length}) > 0.5$$

- "Classification with logistic regression" might seem a **misnomer**
  - **Regression** predicts a continuous variable given input variables
  - **Classification** predicts a discrete variable given input variables
  - Is a "regression" since it computes the probability that the output is a certain value

Boundary Decision for a Classifier

- **Boundary Decision** δ for a classifier = values of independent variables making probability equal to 0.5
- For logistic regression δ:

$$0.5 = \mathit{logit}(\alpha + \beta\delta)$$
$$0 = \alpha + \beta\delta$$
$$\delta = -\frac{\alpha}{\beta}$$

  - Decision boundary has uncertainty due to uncertainty in α and β
- The probability threshold 0.5 chosen for equal misclassification risk
  - Misclassification cost may not be symmetrical
  - E.g., minimize false negatives ("patient has disease, not predicted") or false positives ("patient doesn't have disease, predicted")

Odds

- The **odds** of event $y = 1$ is:

$$\text{odds} \triangleq \frac{\Pr(y = 1)}{1 - \Pr(y = 1)}$$

  - Ratio between favorable vs unfavorable events
- Transformation exists between probability, odds, and log-odds
- More intuitive than probability in "gambling"
  - E.g., odds of getting a 2 rolling a fair die are $\frac{1/6}{5/6} = \frac{1}{5} = 0.2$
  - One favorable event for 5 unfavorable events
- Interpreting **logistic regression in terms of odds**:
  - Since $\theta = \mathit{logit}(\alpha + \beta x)$ and $\theta = \Pr(y = 1)$, we get:

$$\alpha + \beta x = \log(\frac{\Pr(y = 1)}{1 - \Pr(y = 1)})$$

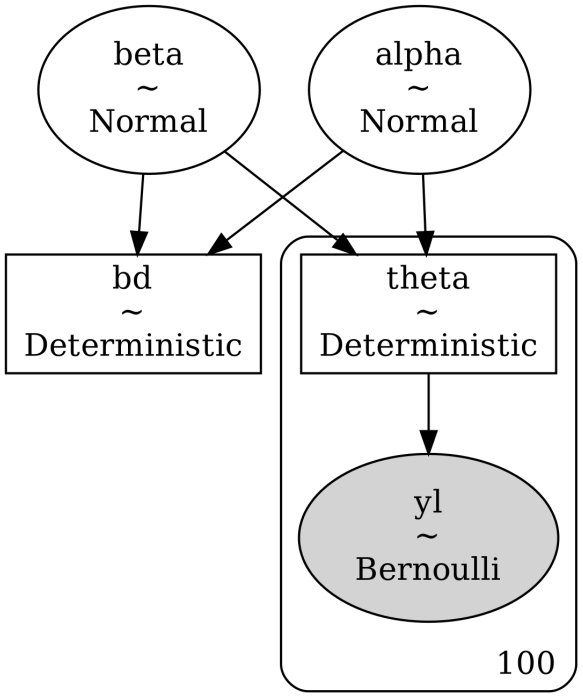
  - Logistic regression models log-odds as linear regression
  - β is the increase of log-odds for a unit change of x

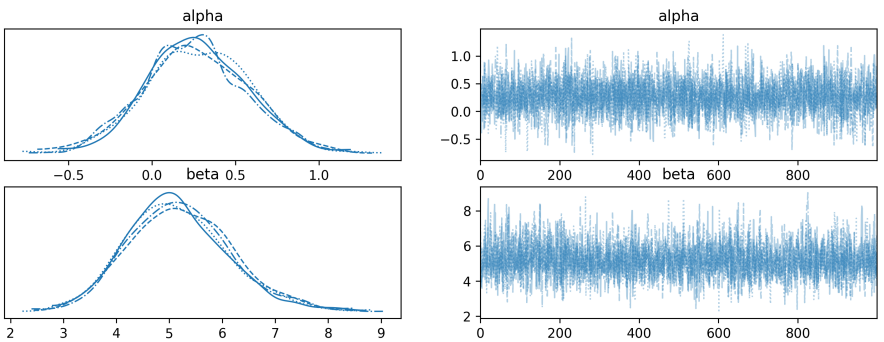
Classification with Logistic Regression: Bayesian

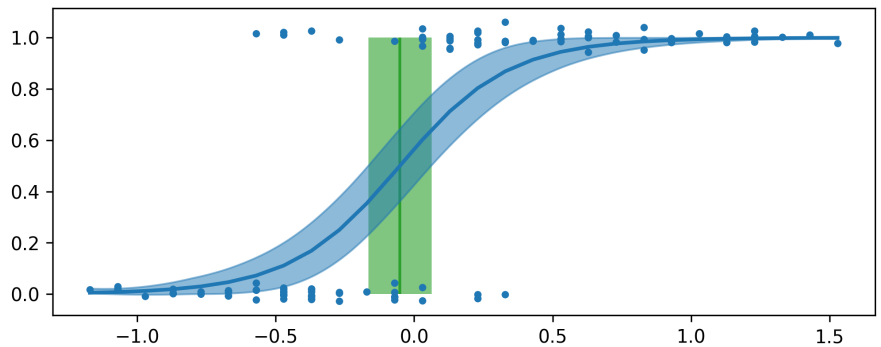
```
: with pm.Model() as model_lrs:
    # Linear part.
    alpha = pm.Normal("alpha", mu=0, sigma=1)
    beta = pm.Normal("beta", mu=0, sigma=5)
    mu = alpha + x_c * beta
    # Sigmoid.
    theta = pm.Deterministic("theta", pm.math.sigmoid(mu))
    # Model.
    yl = pm.Bernoulli("yl", p=theta, observed=y_0)
    # Intercept?
    bd = pm.Deterministic("bd", - alpha / beta)
    #
    idata_lrs = pm.sample(random_seed=123)
```

```
Auto-assigning NUTS sampler...
Initializing NUTS using jitter+adapt_diag...
Multiprocess sampling (4 chains in 4 jobs)
NUTS: [alpha, beta]
```

Sampling 4 chains, 0 divergences

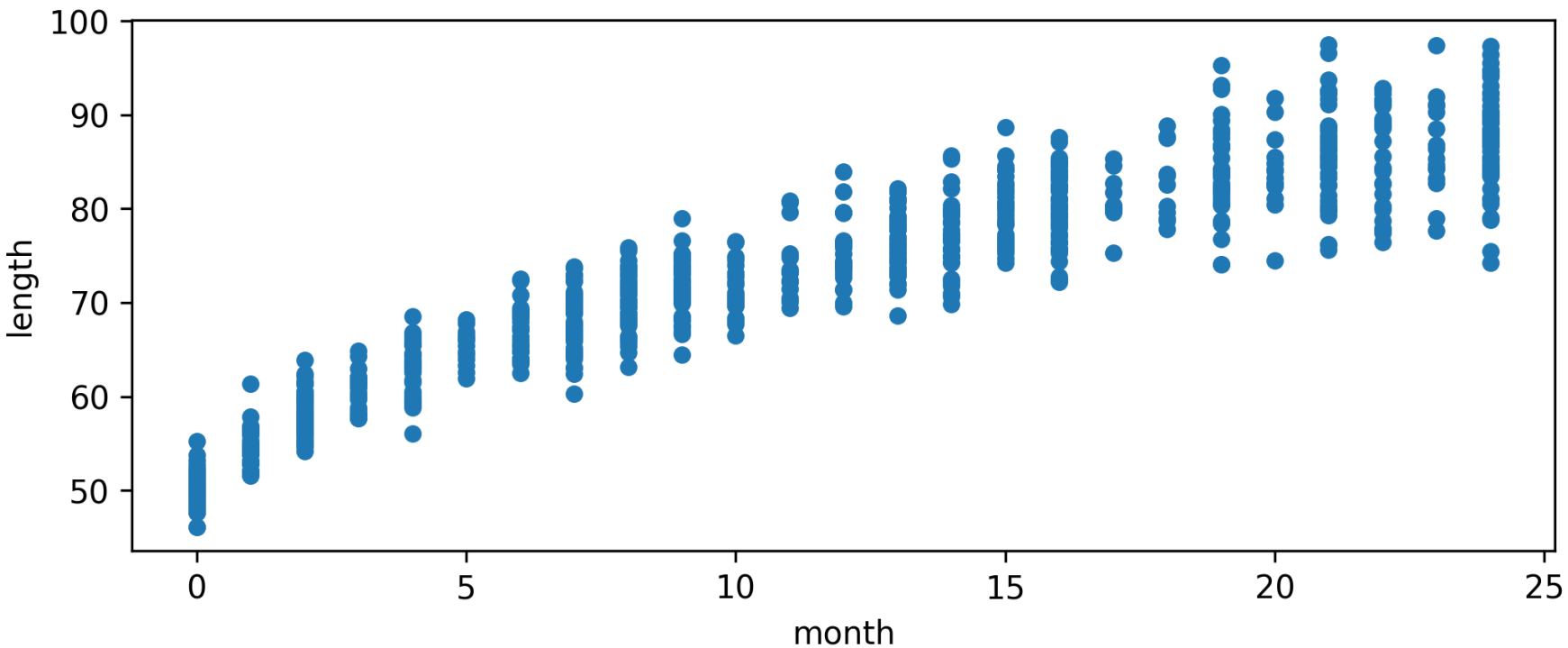






Heteroskedasticity

- **Heteroskedasticity** = when variance of errors is not constant across observations
  - E.g., variance can be a linear function of the dependent variable
- **Example**:
  - Variance of baby height as a function of age is heteroskedastic
  - σ is a linear function of the predictor variable
  - Mean μ is square root of a linear model

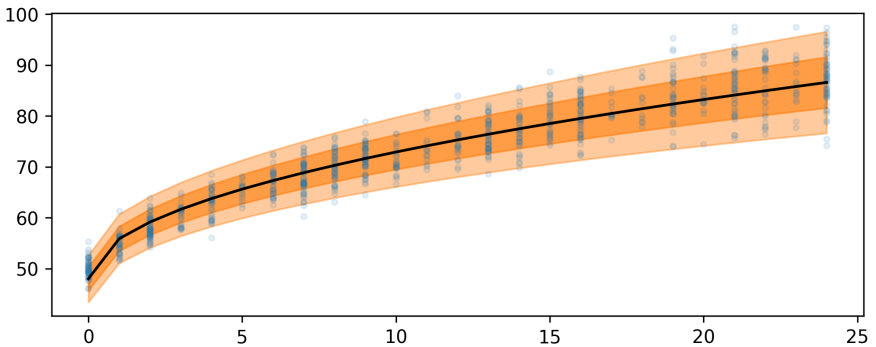


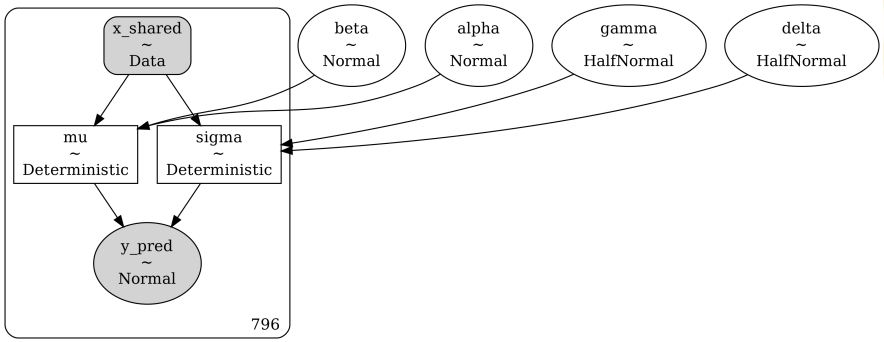
Heteroskedasticity: Bayesian Model

```python
with pm.Model() as model_vv:
    # Create a shared variable so that the data can change after the model is created.
    x_shared = pm.Data("x_shared", data.month.values.astype(float))
    # Linear model for the mean is a function of sqrt(x).
    alpha = pm.Normal("alpha", sigma=10)
    beta = pm.Normal("beta", sigma=10)
    mu = pm.Deterministic("mu", alpha + beta * x_shared ** 0.5)
    # Linear model for the std dev.
    gamma = pm.HalfNormal("gamma", sigma=10)
    delta = pm.HalfNormal("delta", sigma=10)
    sigma = pm.Deterministic("sigma", gamma + delta * x_shared)
    # Fit.
    y_pred = pm.Normal("y_pred", mu=mu, sigma=sigma, observed=data.length)
    #
    idata_vv = pm.sample(random_seed=123)
```

```
Auto-assigning NUTS sampler...
Initializing NUTS using jitter+adapt_diag...
Multiprocess sampling (4 chains in 4 jobs)
NUTS: [alpha, beta, gamma, delta]
```

```
Sampling 4 chains, 0 divergences ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 100%
```





- Concepts
- Coin Example
- Posterior-Based Decisions
- Groups Comparison
- Hierarchical Models
- Simple Linear Model
- ***Multiple linear regression***
- Comparing Models
- Inference Engines

Multiple Linear Regression

- In prediction problems, several independent variables are common
  - E.g., student's grades = f(family income, mother's education, ...)
- **Problem formulation**
  - k independent variables
  - N observations
  - Find a hyperplane of dimension k to explain data

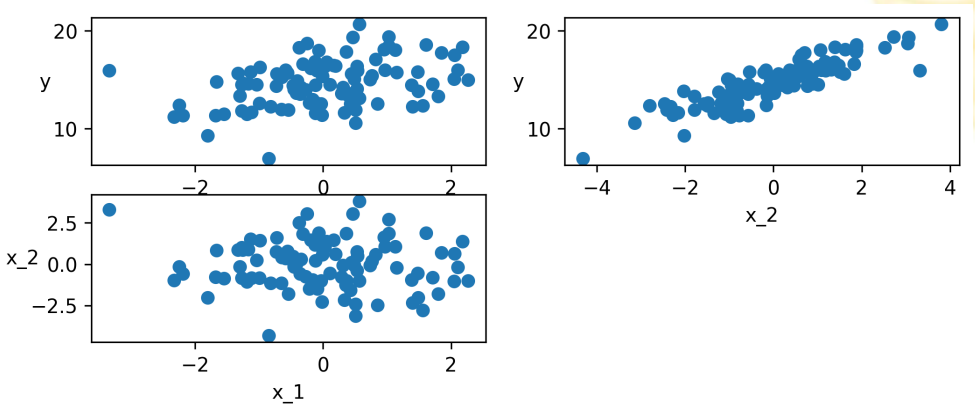
$$\mu = \alpha + \beta_1 x_1 + \beta_2 x_2 + ... + \beta_k x_k$$

  - Same as polynomial regressions but with independent variables

Multiple Regression: Synthetic Example 1/2

- Generate random data

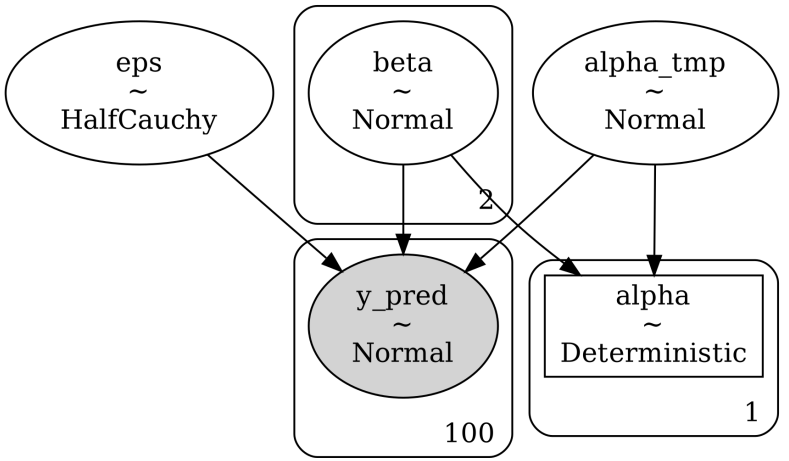
```
np.random.seed(314)

N = 100
#N = 1000
alpha_real = 2.5
beta_real = [0.9, 1.5]
eps_stddev_real = 0.5
eps_real = np.random.normal(0, eps_stddev_real, size=N)

# Independent variables.
X = np.array([np.random.normal(i, j, N) for i, j in zip(
    # mean of gaussian.
    [10, 2],
    # std dev.
    [1, 1.5])]).T
X_mean = X.mean(axis=0, keepdims=True)
X_centered = X - X_mean

# Create samples.
y = alpha_real + np.dot(X, beta_real) + eps_real
```



- Create PyMC model

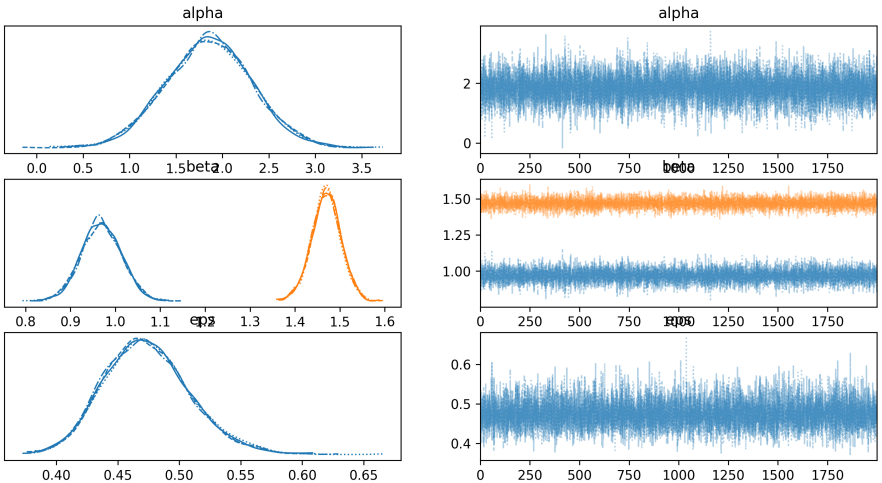
```
with pm.Model() as model_mlr:
    alpha_tmp = pm.Normal('alpha_tmp', mu=0, sigma=10)
    # Beta is a vector.
    beta = pm.Normal('beta', mu=0, sigma=1, shape=2)
    eps = pm.HalfCauchy('eps', 5)
    # mu.
    mu = alpha_tmp + pm.math.dot(X_centered, beta)
    # Extract alpha.
    alpha = pm.Deterministic('alpha', alpha_tmp - pm.math.dot(X_mean, beta))

    # Model.
    y_pred = pm.Normal('y_pred', mu=mu, sigma=eps, observed=y)

    idata_mlr = pm.sample(2000)
```

```
Auto-assigning NUTS sampler...
Initializing NUTS using jitter+adapt_diag...
Multiprocess sampling (4 chains in 4 jobs)
NUTS: [alpha_tmp, beta, eps]
```



Multiple Regression: Synthetic Example 2/2

- Solve model



	mean	sd	hdi_3%	hdi_97%
alpha[0]	1.85	0.46	1.01	2.75
beta[0]	0.97	0.04	0.88	1.05
beta[1]	1.47	0.03	1.41	1.53
eps	0.47	0.03	0.41	0.54

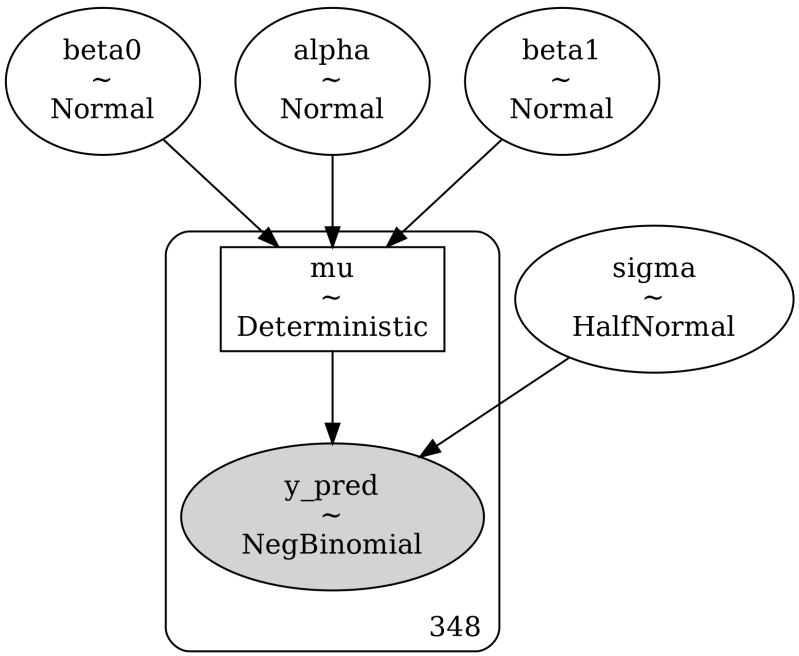
Multiple Regression: Rented Bike Example 1/2

- Assumption: number of bike rented is function of temperature and hour of the day
- Create model

```
with pm.Model() as model_mlb:
    alpha = pm.Normal("alpha", mu=0, sigma=1)
    beta0 = pm.Normal("beta0", mu=0, sigma=10)
    beta1 = pm.Normal("beta1", mu=0, sigma=10)
    sigma = pm.HalfNormal("sigma", 10)
    mu = pm.Deterministic("mu", pm.math.exp(alpha + beta0 * bikes.temperature +
    beta1 * bikes.hour))
    _ = pm.NegativeBinomial("y_pred", mu=mu, alpha=sigma, observed=bikes.rented)
    #
    idata_mlb = pm.sample()
```

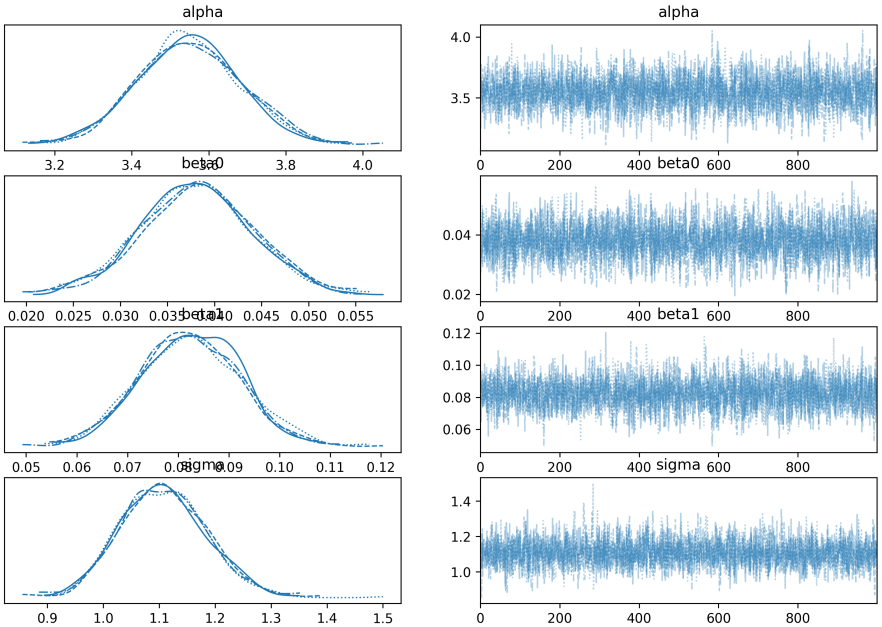
```
Auto-assigning NUTS sampler...
Initializing NUTS using jitter+adapt_diag...
Multiprocess sampling (4 chains in 4 jobs)
NUTS: [alpha, beta0, beta1, sigma]
```

```
Sampling 4 chains, 0 divergences
```



Multiple Regression: Rented Bike Example 2/2

- Solve model



	mean	sd	hdi_3%	hdi_97%
alpha	3.55	0.13	3.32	3.82
beta0	0.04	0.01	0.03	0.05
beta1	0.08	0.01	0.06	0.10
sigma	1.11	0.08	0.97	1.25

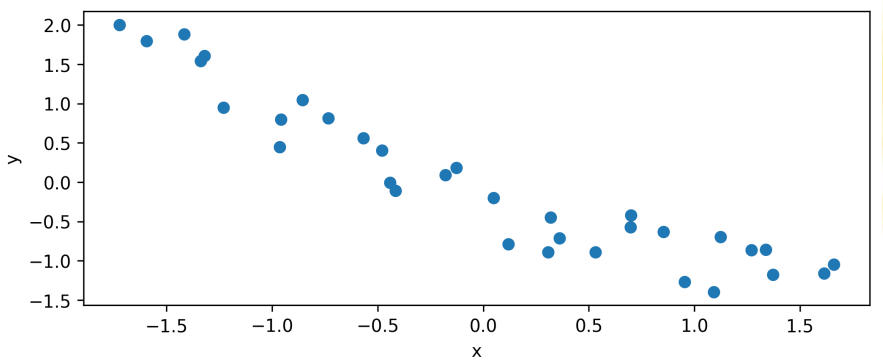
- Concepts
- Coin Example
- Posterior-Based Decisions
- Groups Comparison
- Hierarchical Models
- Simple Linear Model
- Multiple linear regression
- ***Comparing Models***
  - The Balance Between Simplicity and Accuracy
  - Measures of Predictive Accuracy
  - Regularizing priors
  - Regularizing Priors
  - Mixture Models
  - 7.5, Zero-Inflated and Hurdle Models
  - 7.6, Mixture Models and Clustering
  - 7.7, Non-Finite Mixture Models
  - 7.8, Continuous Mixtures
- Inference Engines

Models as Maps of the Real World

- Often you need to compare models to understand which one is **"better"**
- Models are a **map, not a copy** of the real world
  - *"All models are wrong, but some are useful"* (Box, 1976)
    - All models are wrong since they aren't the actual territory
    - Some models describe a problem better than others
- Models have a **purpose**
  - Are approximations to understand a problem
  - A model can't reproduce all aspects equally well
  - Different models capture different data aspects

Posterior Predictive Checks

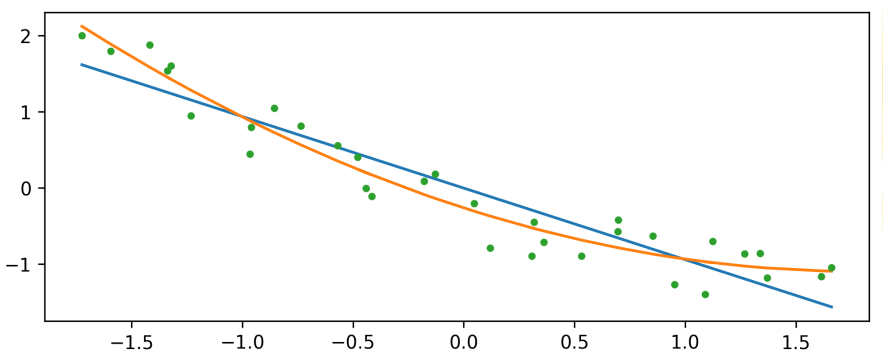
- Goal of PPC:
  - Evaluate model's data explanation
  - Understand model limitations
  - Improve model
- Given data from parabola + noise:
  - Fit with linear model
  - Fit with quadratic model
  - Compare predicted (posterior) vs observed data

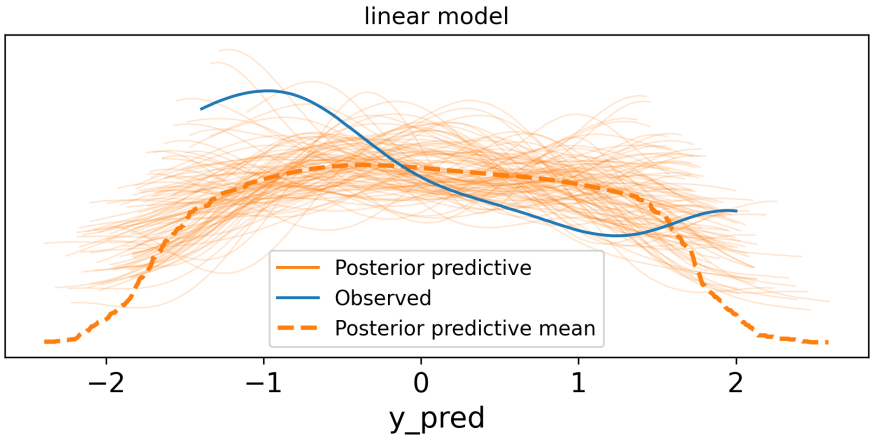


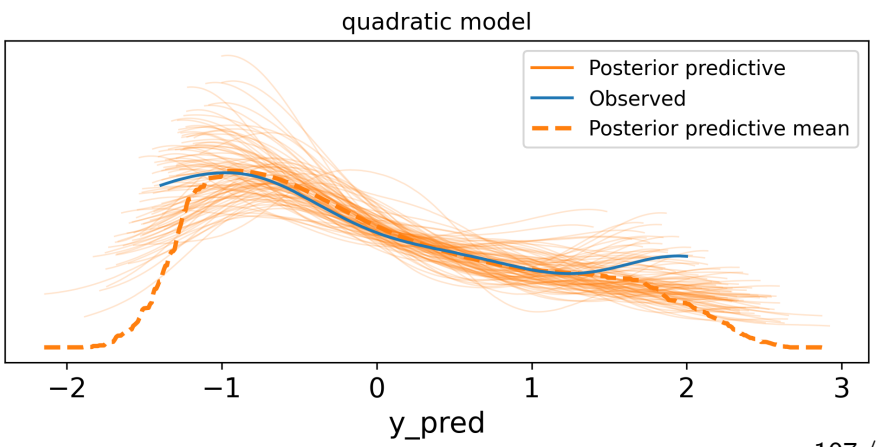

```python
# Linear model.
with pm.Model() as model_l:
    # mu = alpha + beta * x
    alpha = pm.Normal('alpha', mu=0, sigma=1)
    beta = pm.Normal('beta', mu=0, sigma=10)
    mu = alpha + beta * x_c[0]
    #
    sigma = pm.HalfNormal('sigma', 5)
    #
    y_pred = pm.Normal('y_pred', mu=mu, sigma=sigma, observed=y_c)
    #
    idata_l = pm.sample(2000, idata_kwargs={"log_likelihood": True})
    idata_l.extend(pm.sample_posterior_predictive(idata_l))


# Quadratic model.
with pm.Model() as model_p:
    # mu = alpha + beta_1 * x + beta_2 * x^2
    alpha = pm.Normal('alpha', mu=0, sigma=1)
    # Beta is a 2-dim vector.
    beta = pm.Normal('beta', mu=0, sigma=10, shape=order)
    mu = alpha + pm.math.dot(beta, x_c)
    #
    sigma = pm.HalfNormal('sigma', 5)
    #
    y_pred = pm.Normal('y_pred', mu=mu, sigma=sigma, observed=y_c)
    #
    idata_q = pm.sample(2000, idata_kwargs={"log_likelihood": True})
    idata_q.extend(pm.sample_posterior_predictive(idata_q))
```

Posterior Predictive Checks

- Compare KDE of observed and predicted data:
  - Linear model KDE doesn't match
  - Quadratic model KDE matches better
  - High uncertainty in both models, especially at tails
- Compare mean / interquartile range for data vs model
  - Plot dispersion of mean and IQR for models vs data
  - Data set provides a single point
  - Posterior provides a distribution
- Statistics "orthogonal" to model's focus are more informative







Bayesian P-Value for a Statistic

- A Bayesian p-value summarizes the comparison between simulated and observed data
- **Procedure**
  - Given the posterior predictive $\tilde{Y}$
  - Choose a summary statistic T (E.g., mean, median, standard deviation)
  - Compute T for:
    - The observed data T_{obs}
    - The simulated data T_{sim} from the posterior predictive
  - Compute the Bayesian p-value as the portion of simulated datasets where the test statistic is smaller than the observed data:

$$\text{Bayesian p-value} \triangleq \Pr(T_{sim} \geq T_{obs} | \tilde{Y})$$

  - If observed values agree with predicted ones, the value should be around 0.5

Bayesian P-Value for Entire Distribution

- Instead of using a summary statistic, one can compute "the probability of predicting a lower or equal value for each observed value"
- If the model is well calibrated, it captures all observations equally well, the probability should be the same for all observed values
  - The output should be a uniform distribution

Bayesian P-Value: Example

- Study the height of people in a population
- **Fit the Bayesian model**
  - Assume a normal distribution with unknown mean and variance
  - Collect observed data of heights (e.g., 100 people)
  - Specify a prior distribution for mean and variance
  - Combine observed data with prior to obtain a posterior distribution of mean and variance of population height
- **Compute Bayesian p-value**
  - From posterior distribution:
    - Generate new simulated datasets
    - For each dataset, compute mean height
  - Use test statistic T, as the difference between the mean of the replicated dataset and the observed mean
  - Compute Bayesian p-value: the proportion of replicated datasets where the test statistic is $>=$ test statistic for observed data
    - A value close to 0.5 means the observed data is covered by the model
    - A value close to 0 or 1 indicates a poor fit

Bayesian vs Frequentist P-Value

- **Frequentist p-value** is the probability of getting observed data as or more extreme, assuming the null hypothesis is true
- **Bayesian p-value** is the probability that simulated data from the model (i.e., posterior predictive check) is as or more extreme than the observed data
- P-value measures inconsistency between observed data and:
  - A null hypothesis (frequentist approach)
  - Model (Bayesian approach)
- Does p-value incorporate uncertainty?
  - (frequentist) No: use single point estimates
  - (Bayesian) Yes, incorporate uncertainty of parameter estimates

- Concepts
- Coin Example
- Posterior-Based Decisions
- Groups Comparison
- Hierarchical Models
- Simple Linear Model
- Multiple linear regression
- Comparing Models
  - ***The Balance Between Simplicity and Accuracy***
  - Measures of Predictive Accuracy
  - Regularizing priors
  - Regularizing Priors
  - Mixture Models
  - 7.5, Zero-Inflated and Hurdle Models
  - 7.6, Mixture Models and Clustering
  - 7.7, Non-Finite Mixture Models
  - 7.8, Continuous Mixtures
- Inference Engines

Occam's Razor

- *"If you have **equivalent** explanations for the same phenomenon, you should choose the **simpler** one"*
  - Quality of explanation $\approx$ accuracy
  - Simpler $\approx$ number of model parameters
- Ideally we should balance complexity and accuracy in a quantitative way
- Complexity vs accuracy
  - Increasing model complexity (e.g., number of model parameters) is accompanied by increasing in-sample accuracy, but not necessarily out-of-sample accuracy
  - The complex model:
    - Did not "learn" from the data but just "memorize" it
    - Do a bad job generalizing to predict potentially observable data

Overfitting and Underfitting

- A model is **overfit** when it has many parameters, fitting the training data well but unseen data poorly
  - Overfitting in terms of signal/noise:
    - Each dataset has "signal" and "noise"
    - We want the model to learn the signal
    - A model overfits when it learns the noise, obscuring the signal
- A model is **underfit** when it has few parameters, fitting the dataset poorly
  - An underfit model doesn't learn the signal well
  - E.g., a constant fits a dataset, only learning the mean

Bias-Variance Trade-Off

- A model has **high bias** when it has low ability to accommodate the data (i.e., underfitting)
  - E.g., a polynomial of degree 0
- A model has **high variance** when it has high capacity and it is sensitive to details in the data, capturing noise (i.e., overfitting)
  - E.g., a polynomial of degree 5
- Trade-off between bias and variance
  - Goal: balance simplicity and goodness of fit
  - Aim for a model that "fits the data right," avoiding overfitting or underfitting

- Concepts
- Coin Example
- Posterior-Based Decisions
- Groups Comparison
- Hierarchical Models
- Simple Linear Model
- Multiple linear regression
- Comparing Models
  - The Balance Between Simplicity and Accuracy
  - ***Measures of Predictive Accuracy***
  - Regularizing priors
  - Regularizing Priors
  - Mixture Models
  - 7.5, Zero-Inflated and Hurdle Models
  - 7.6, Mixture Models and Clustering
  - 7.7, Non-Finite Mixture Models
  - 7.8, Continuous Mixtures
- Inference Engines

Accuracy Measures

- **In-sample accuracy** is measured on the data used to fit a model
- **Out-of-sample accuracy** is measured on data not used to fit a model
  - Aka "predictive accuracy"
- In-sample accuracy $>$ out-of-sample accuracy
- There is a trade-off between how much data is used for training and for evaluating true accuracy

Information Criteria: Intuition

- Information criteria are tools to compare models in terms of fitting the data taking into account their complexity through a penalization term
  - Out-of-sample accuracy $\approx$ in-sample accuracy $+$ a term penalizing model complexity

Model Parameters for Bayesian vs Non-Bayesian Set-Up

- Model parameters θ are:
  - A point estimate (frequentist)
  - A distribution estimated from the posterior (Bayesian)

Squared Error Metric

- Squared error between the data and the predictions from the model

$$\frac{1}{N}\sum_i (y_i - \mathbb{E}[y_i|\theta])^2$$

  where $\mathbb{E}[y_i|\theta]$ is the predicted value given the parameters
- Squaring errors:
  - Ensures that errors do not cancel out
  - Emphasizes large errors (vs absolute values)

Log-Likelihood

- In a probabilistic set-up, log-likelihood is more natural
- **Log-likelihood** is defined as:

$$\sum_i \log \Pr(y_i|\theta)$$

- Deviance is defined as:

$$-2\sum_i \log \Pr(y_i|\theta)$$

Maximum Likelihood Estimation (MLE)

- MLE finds the parameter values that make the observed data most probable
  - Denoted by $\hat{\theta}_{MLE}$
  - It's a point not a distribution
- Procedure:
  - Given the data $x_1, x_2, ..., x_n$
  - Assume it comes from a distribution with an unknown parameter θ
  - Pick the value of θ that makes the data most likely given a likelihood function

$$\begin{cases} L(\theta) = \log \Pr(x_1, x_2, ..., x_n|\theta) \\ \hat{\theta}_{MLE} = \text{argmax}_\theta L(\theta) \end{cases}$$

- In Bayesian terms, MLE is equivalent to the mode of θ using flat priors
  - Aka MAP (maximum a posteriori)

Akaike Information Criterion (AIC)

- AIC is defined as

$$AIC = -2 \sum \log \Pr(y_i | \hat{\theta}_{MLE}) + 2\text{num}_{params}$$

  where:

  - $\hat{\theta}_{MLE}$ is the maximum likelihood estimation of θ
  - num_{params} is the number of parameters

- **Interpretation**:

  - The first term measures how well the model fits the data
  - The second term penalizes complex models

- **Cons**:

  - Discard information about uncertainty of posterior estimation
  - MLE assumes flat priors (vs informative and weakly informative priors)
  - Number of parameters is not always a good measure of complexity
    - E.g., in hierarchical models the effective number of params is smaller

Bayesian Information Criteria

- Widely Applicable Information Criteria (WAIC)
  - Bayesian version of AIC
  - It has two terms:
    - One that measures how good the fit is
    - One that penalizes complex models
  - WAIC uses the posterior distribution to estimate both terms ‘
- Bayesian Information Criteria (BIC)
  - It is not fully Bayesian
    - Like AIC, it assumes flat priors and uses MLE

Cross-Validation

- Cross-validation (CV)
  - Simple and effective solution to use all data to compare models
  - **Procedure**
    - Partition data into K portions of equal size and similar statistics
    - Use $K-1$ partitions to train the model and the remaining partition to test it
    - Repeat K times
    - Average the results
- Leave-one-out cross-validation (LOO-CV)
  - **Procedure**:
    - The model is fit for all data, excluding one observation
    - The model's predictive accuracy is tested on the left out observation
    - Repeat the process for all observations and average
  - **Cons**
    - It is very computationally expensive since one needs to refit the model
- How to adapt cross-validation to a Bayesian approach?
  - CV and LOO require multiple model fits and fitting a Bayesian model is very expensive
  - Yes! There is a way to approximate using a single fit to the data

ELPD with LOO-CV

- We want to compute $ELPD_{LOO-CV}$ where:
  - "Expected Log-Pointwise predictive Density" (ELPD)
    - It should be ELPPD and not ELPD!
  - We use "Leave-One-Out Cross-Validation" (LOO-CV) to compute it
- The definition of ELPD with LOO-CV is:

$$ELPD_{LOO-CV} = \sum_{i=1}^{n} \log \int p(y_i|\theta)(\theta|y_{-i})d\theta$$

  - Where:
    - Fit a model using all the data without y_{-i}
    - Predict the unseen y_i with the model
    - Integrate on all the posterior values
    - Repeat for all the points
- How to compute it efficiently?
  - Use "Pareto smooth importance sampling leave-one-out cross-validation"

Pointwise Predictive Density (PPD)

- The **pointwise predictive density** (or probability) for a given data point y_i is defined as the posterior predictive probability, given the rest of the data

$$PPD \triangleq \Pr(y_i|data - \{i\}) = \int p(y_i|\theta)p(\theta|y_{-i})d\theta$$

where:

  - y_i is the observed data point
  - $p(y_i|\theta)$ is the likelihood of the data point y_i given the model parameters θ
  - $p(\theta|y_{-i})$ is the posterior distribution of the model parameters given the rest of the observed data
  - The integral averages over the posterior distribution of θ, capturing the uncertainty about the model parameters

- **Interpretation**
  - PPD measures how well a model, training on the data excluding y_i, predicts y_i
  - It's like cross-validation but using the Bayesian concept of averaging over the distribution of the parameters

Expected Log Pointwise Predictive Density

- The ELPD is the average over unseen points of the log PPD

$$ELPD = \sum_{i=1}^{n} \log \int p(y_i|\theta_{-i}) p(\theta_{-i}|y_{-i}) d\theta$$

- **Interpretation**
  - It can be used to determine which model generalizes better to new data
  - ELPD measures the predictive accuracy of a Bayesian model on unseen data
  - Train on y_{-i}, i.e., all data excluding y_i
  - Test on y_i

Approximating PPD

- Calculating analytically the pointwise posterior density integral

$$PPD = \int p(y_i|\theta)p(\theta|y_{-i})d\theta$$

  is difficult

  - The posterior $p(\theta|y_{-i})$ rarely has a closed form
  - The integral on θ is on a high-dimensional space

- It can be approximated numerically given posterior samples s of the model parameters $\theta^{(s)}$

$$PPD \approx \frac{1}{S}\sum_{s} p(y_i|\theta_{-i}^{(s)})$$

  - Suppose we already have posterior samples $\theta^{(s)} \sim p(\theta|y)$ from the full dataset

PSIS-LOO-CV

- Compute the Expected Log Pointwise Predictive Density (ELPD) using Leave-One-Out Cross-Validation (LOO-CV):

$$ELPD_{LOO-CV} \triangleq \sum_i \log \int p(y_i|\theta)p(\theta|y_{-i})d\theta$$

- **Problem**: Train once per point
- **Solution**:
  - Pareto-Smoothed Importance Sampling (PSIS) Leave-One-Out Cross-Validation (LOO-CV) estimates the formula without refitting the model for every point
  - **Importance sampling**:
    - Use the full dataset to approximate the posterior distribution when a single observation is left out
    - Re-weight posterior samples based on importance
  - **Pareto-smoothing**:
    - Stabilize importance weights, reducing the impact of extreme weights
    - E.g., if an observation left out has a large influence on the posterior distribution
    - Provide diagnostics to assess the reliability of importance weights

Predictive Accuracy with Arviz

- If the inference data has the log-likelihood group

  `pm.sample(idata_kwargs="log_likelihood": True)`

  metrics such as WAIC and LOO (with / without ELPD) can be automatically computed
- In the first section
  - The first row is ELPD
  - The second row is the effective number of parameters
- In the second section, there is the Pareto k diagnostic
  - Since all the values are between 0 and 0.7, the approximation can be trusted

Comparing Predictive Accuracy with Arviz

- In general the predictive accuracy metrics should be interpreted in relation to other models

Model Averaging

- You have multiple models explaining the data: what do you do?
  1. Select a single model
     - Simple solution used in frequentist approach
     - "Model selection"
  2. Report all the models with their informations (e.g., standard errors, posterior predictive checks)
     - Express advantages and shortcomings of the models
  3. Average all the models
     - Build a meta-model using a weighted average of each model
     - Weight prediction by the difference between information criteria (e.g., WAIC, LOO) of the models
     - A hierarchical model is a continuous versions of multiple discrete models

Evidence of Data Given a Model

- The Bayesian way to compare k models is to calculate the evidence of each model $\Pr(Y|M_k)$, i.e., the probability of observed data Y given each model M_k
  - Typically we ignore the evidence when we do parameter inference
- Consider the Bayes theorem for the parameters θ and the data Y, given a model M_k

$$\Pr(\theta|Y, M_k) = \frac{\Pr(Y|\theta, M_k)\Pr(\theta|M_k)}{\Pr(Y|M_k)}$$

- We find the parameters θ that maximizes the ratio, independently of the probability of the evidence

$$\text{argmax}_\theta \Pr(\theta|y, M_k) = \text{argmax}_\theta \Pr(y|\theta, M_k)\Pr(\theta|M_k)$$

- Even if we need to choose the best model among $M_1, ..., M_k$ we can pick the one that maximizes

$$\text{argmax}_k \Pr(M_k|y) \propto \Pr(y|M_k)\Pr(M_k)$$

Bayes Factors

- The Bayes factors are defined as the ratio of the two marginal likelihoods under competing hypotheses

$$BF = \frac{\Pr(y|M_0)}{\Pr(y|M_1)}$$

where $BF > 1$ means that the model 0 explains the data better than model 1 | **Bayes factor** | **Support** | |———-|———-| | 1-3 | Anecdotal | | 3-10 | Moderate | | 10-30 | Strong | | 30-100 | Very strong | | >100 | Extreme |

- Intuition
  - Bayes factors are a quantitative tool that helps compare how likely two competing explanations (i.e., models) are, given the evidence you find
  - Bayes factors are like a scale that weigh how much evidence supports one theory over another

Assumption of Bayes Factors

- The assumption of Bayes factor is that the models have the same prior probability
- Otherwise we need to compute the "posterior odds" as "Bayes factors" x "prior odds"

$$\frac{\Pr(M_0|y)}{\Pr(M_1|y)} = \frac{\Pr(y|M_0)}{\Pr(y|M_1)} \frac{\Pr(M_0)}{\Pr(M_1)} = \text{Bayes factors} \times \text{prior odds}$$

Bayes Factors: Pros and Cons

- Looking at the definition of marginal likelihood (aka evidence):

$$p(y) = \int_{\theta} p(y|\theta)p(\theta)d\theta$$

- Making the dependency of the model M_k explicit

$$p(y|M_k) = \int_{\theta_k} p(y|\theta_k, M_k)p(\theta_k, M_k)d\theta_k$$

- Pros
  - Models with more parameters have a larger prior, so the Bayes factor has a built-in Occam's Razor
- Cons
  - The marginal likelihood needs to be computed numerically over a large dimensional space
  - The marginal likelihood depends on the value of the prior
    - Changing the prior might not affect the inference of θ but have a direct effect on the marginal likelihood

Hierarchical Models: Candies in a Jar Examples

- Each classroom has a jar filled with candies, each different but coming from the same candy shop
- Kids in each classroom need to guess the number of candies in each jar
- Individual guesses
  - Think of each jar as its own little puzzle
  - E.g., guess based on how big the jar is, how filled it is
  - Each jar has certain "parameters"
- Group learning
  - Consider what you learn from other jars since they come from the same candy shop
  - E.g., the shop prefers to use a certain type of candies, or fills the jar up to a certain level
  - The jars have certain "hyper-parameters"
- Sharing info
  - As you make more guesses, you start sharing what you have learned with your friends about each jar
  - The hierarchical model lets the info flow across models for individual jar

Computing Bayes Factors as Hierarchical Models

- The computation of Bayes factors can be framed as a hierarchical model
  - The high-level parameter is an index assigned to each model and sampled from a categorical distribution
- We perform inference of the competing models at the same time, using a discrete variable jumping between models
  - The proportion we use to sample each model is proportional to $\Pr(M_k|y)$
- Then we compute the Bayes factors
- The models can be different in the prior, in the likelihood, or both

Common Problems When Computing Bayes Factors

1. If one model is better than the other, then we will spend more time sampling from it
   - Cons: under-sample one of the models
2. Values of the parameters are updated, even when the parameters are not used to fit that model
   - E.g., when model 0 is chosen, the parameters in model 1 are updated, but they are only restricted by the prior
   - If the prior is too vague, the parameter values might be too far from previous accepted values and the step is rejected
   - TODO: ?

- Solutions to improve sampling
  - Force both models to be visited equally
  - Use "pseudo priors"

Using Sequential Monte Carlo to Compute Bayes Factors

- TODO

Bayes Factors and Information Criteria

-
- If we take the log of Bayes factors, we turn ratio of marginal likelihood into a difference, which is similar to comparing differences in information criteria
- We can interpret each marginal likelihood as having:
  - a fitting term (i.e., how well the model fits the data)
  - penalizing term (i.e., averaging over the prior)
    - more parameters $\rightarrow$ more diffused the prior $\rightarrow$ greater penalty
-
- TODO

- Concepts
- Coin Example
- Posterior-Based Decisions
- Groups Comparison
- Hierarchical Models
- Simple Linear Model
- Multiple linear regression
- Comparing Models
  - The Balance Between Simplicity and Accuracy
  - Measures of Predictive Accuracy
  - ***Regularizing priors***
  - Regularizing Priors
  - Mixture Models
  - 7.5, Zero-Inflated and Hurdle Models
  - 7.6, Mixture Models and Clustering
  - 7.7, Non-Finite Mixture Models
  - 7.8, Continuous Mixtures
- Inference Engines

- Concepts
- Coin Example
- Posterior-Based Decisions
- Groups Comparison
- Hierarchical Models
- Simple Linear Model
- Multiple linear regression
- Comparing Models
  - The Balance Between Simplicity and Accuracy
  - Measures of Predictive Accuracy
  - Regularizing priors
  - ***Regularizing Priors***
  - Mixture Models
  - 7.5, Zero-Inflated and Hurdle Models
  - 7.6, Mixture Models and Clustering
  - 7.7, Non-Finite Mixture Models
  - 7.8, Continuous Mixtures
- Inference Engines

Priors and Regularization

- Using weakly/informative priors is a way of pushing a model to prevent overfitting and generalize well
- This is similar to the idea of "regularization"
- Regularization
  - Reduce information that a model can represent and reduce chances to capture noise instead of signal
  - E.g., penalize large values for the parameters in a model
  - E.g., ridge and Lasso regression applies regularization to least square method

Popular Regularization Methods in Bayesian Framework

- Ridge regression
  - Normal distribution for coefficients of linear model, pushing them toward zero
- Lasso regression
  - MAP of posterior using Laplace priors for coefficients
  - Because Laplace distribution looks like Gaussian with a sharp peak at zero, it provides "variable selection" since it induces sparsity of model

- Concepts
- Coin Example
- Posterior-Based Decisions
- Groups Comparison
- Hierarchical Models
- Simple Linear Model
- Multiple linear regression
- Comparing Models
  - The Balance Between Simplicity and Accuracy
  - Measures of Predictive Accuracy
  - Regularizing priors
  - Regularizing Priors
  - ***Mixture Models***
  - 7.5, Zero-Inflated and Hurdle Models
  - 7.6, Mixture Models and Clustering
  - 7.7, Non-Finite Mixture Models
  - 7.8, Continuous Mixtures
- Inference Engines

Mixture Models

- Mixture models are models that assume that data comes from a mixture of distributions or where the solution can be approximated as a mix of simpler distributions
- Mixtures can model data from sub-populations
  - E.g., distribution of heights in adult human population, made of male and female
  - E.g., clustering of handwritten digits (e.g., 10 sub-populations)
- Mixture models as statistical trick to handle complex distributions
  - E.g., mix of Gaussians to describe an arbitrary distribution
    - E.g., multimodal, skewed distributions
    - The number of distributions depends on the accuracy of the approximation and the details of the data
  - E.g., Kernel density estimation (KDE) is a (non-Bayesian) non-parametric estimation technique
    - Place a Gaussian with a fixed variance on top of each data points
    - Use the sums of all the individual Gaussians to approximate the empirical distribution of the data

Finite Mixture Models

- $=$ the PDF of the observed data is a weighted sum of the PDFs of K subgroups

$$p(y) = \sum_{i=1}^{K} w_i p(y|theta_i)$$

where:

- K is finite
- $w_i \in [0, 1]$ (it's like the probability of component i)
- $\sum_i w_i = 1$
- $p(y|\theta_i)$ are simpler distributions (e.g., Gaussian or Poisson)

Conceptual Solution of a Mixture Model

- The assumption of a mixture model is that the data is generated by K underlying distributions / components
  - In other words, each data point is assumed to come from one of the components, but we don't know which
- The goal is to determine which component of the model k the data point x most likely belongs to
  - For each point x, assign probabilities of belonging to one of the K components $\Pr(k|x)$
  - This is modeled as a random variable, which is called "latent variable" since it can't be observed
  - E.g., for two components ($K = 2$) we use a Bernoulli, using a Beta distribution as prior (since it is conjugate prior for Bernoulli)
  - E.g., for K outcomes we can use as prior
    - A Categorial distribution to generalize the Bernoulli
    - A Dirichlet distribution to generalize the Beta distribution
- One can estimate the likelihood of the observed data based on:
  - Parameters of the individual components
  - Probabilities that a given data point belongs to each component

Categorical Distribution

- Discrete distribution representing "rolling a K-sided unfair die" or "choosing a random card from a deck of cards"
  - It generalizes a Bernoulli distribution $K = 2$
- It is parametrized with probabilities for each possible outcome $(p_1, p_2, ..., p_K)$, where p_i is the probability of outcome i

Dirichlet Distribution

- Defined in a simplex which is a generalization of a triangle in K-dimensions, where K vectors elements are in $[0, 1]$ and sum to 1
  - A 1d simplex is an interval
  - A 2d simplex is a triangle
  - A 3d simplex is a tetrahedron
  - ...
- Dirichlet distribution generalizes the Beta distribution
  - Beta models the probability of a single proportion (e.g., the probability of success in a Bernoulli trial)
  - Beta models 2 outcomes, one with probability p and the other with $1 - p$
  - It uses two parameters to describe its shape
- The Dirichlet distribution models the probability of K outcomes
  - It represents the uncertainty over the proportions of different outcomes in a multinomial experiment

Dirichlet Distribution: Examples

- Bucket of colored balls
  - You have a bucket of colored balls, each ball comes in 3 colors
  - You want to figure out the probability of picking a ball of each color
  - You are uncertain about the probability and you model that uncertainty
- You want to divide a pie into 4 different slices
  - You want to model the size of the slices (making sure they split the entire pie), modeling the uncertainty

Re-Parametrization Using Marginalization

- Consider a mixture model that include latent variables representing the component from which each observation is drawn
- Performing posterior sampling on these models is inefficient for several reasons:
  - Discrete variables introduce discontinuities
  - Latent variable introduces dependencies
- A solution is "marginalization"
  - Removing the latent variable by integrating out from the model
  - The observations are directly modeled from a mixture distribution
  - This makes sampling more efficient
  - The problem is that the model becomes more complex mathematically
  - `pymc` has distributions (e.g., `NormalMixture`) where the latent variable is already marginalized

Non-Identifiability of Parameters

- Parameters in a model are "not identifiable" when one or more parameters cannot be uniquely determined
- In practice multiple model fits give different parameters corresponding to the same model
  - E.g., given a bimodal distribution sum of two Gaussians, solving the same model can switch the value of the means (aka "label switching problem")
  - E.g., variables with high-correlation in linear models
- Solutions
  - Arrange the means of the components to be in increasing order
    - E.g., in `pymc` add a "potential" to the likelihood (which doesn't depend on the data) so that a constraint is not violated
  - Use informative priors to guide a model towards a canonical representation

How to Choose K

- How to decide the number of components K in a finite mixture model?
- Solution
  - Start with a small number of components K
  - Increase K to improve the model-fit evaluation
  - Use model selection techniques to find best trade-off
    - E.g., using posterior-predictive checks, WAIC, LOO, or domain expertise

- Concepts
- Coin Example
- Posterior-Based Decisions
- Groups Comparison
- Hierarchical Models
- Simple Linear Model
- Multiple linear regression
- Comparing Models
  - The Balance Between Simplicity and Accuracy
  - Measures of Predictive Accuracy
  - Regularizing priors
  - Regularizing Priors
  - Mixture Models
  - ***7.5, Zero-Inflated and Hurdle Models***
  - 7.6, Mixture Models and Clustering
  - 7.7, Non-Finite Mixture Models
  - 7.8, Continuous Mixtures
- Inference Engines

Hurdle Models

- A Bernoulli model determines if the count variable is zero or a different distribution truncated at 0, i.e., which doesn't assume the value 0
- E.g., hurdle Poisson, NegativeBinomial, Gamma, LogNormal

Zero-Inflated vs Hurdle Models

- Zero-inflated models are a mixture of zeros and a distribution of zeros and non-zeros
  - The probability $\Pr(x = 0)$ can only be larger than the base distribution, i.e., the probability of zero is "inflated"
- Hurdle models are a mixture of zeros and non-zeros
  - The probability $\Pr(x = 0)$ is independent of the base distribution

Hanging root-ograms

- Useful for treating issues such as over-dispersion and/or excess zeros in count data models

1. Plot the square root of observed and predicted values
   - It makes it simper to adjust for scale differences
2. Bars of observed data are hanging from the expected values
   - It makes it simple to see if it's a good fit or not by comparing the bottom of the values to a base

- Concepts
- Coin Example
- Posterior-Based Decisions
- Groups Comparison
- Hierarchical Models
- Simple Linear Model
- Multiple linear regression
- Comparing Models
  - The Balance Between Simplicity and Accuracy
  - Measures of Predictive Accuracy
  - Regularizing priors
  - Regularizing Priors
  - Mixture Models
  - 7.5, Zero-Inflated and Hurdle Models
  - ***7.6, Mixture Models and Clustering***
  - 7.7, Non-Finite Mixture Models
  - 7.8, Continuous Mixtures
- Inference Engines

Clustering

- Goal: group objects so that the objects in a given group (aka cluster) are "closer" to each other than to those in the other groups
  - Degree of closeness is computed using a distance (e.g., Euclidean distance)
- Clustering is an unsupervised learning task
  - Similar to classification but without knowing the correct labels

Soft- vs Hard- Clustering

- Soft-clustering computes the probability of each data point belonging to each cluster
- Hard-clustering assigns each point to a single cluster
- One can turn soft-clustering into hard- by assigning each data point to the cluster with the highest probability
  - E.g., assigning points to the cluster with highest probability
  - E.g., using a threshold for probability of 0.5 (as in classification for logistic regression)

Probabilistic Clustering

- Aka "model-based clustering"
- = clustering using probabilistic model, i.e., compute the probability of each point belonging to one of the customers
- E.g., we assume that data is generated by a mixture models

- Concepts
- Coin Example
- Posterior-Based Decisions
- Groups Comparison
- Hierarchical Models
- Simple Linear Model
- Multiple linear regression
- Comparing Models
  - The Balance Between Simplicity and Accuracy
  - Measures of Predictive Accuracy
  - Regularizing priors
  - Regularizing Priors
  - Mixture Models
  - 7.5, Zero-Inflated and Hurdle Models
  - 7.6, Mixture Models and Clustering
  - ***7.7, Non-Finite Mixture Models***
  - 7.8, Continuous Mixtures
- Inference Engines

How Many Mixture Models to Use?

- For some problems, it is easy to choose the number of mixtures to use, since we know how many groups there are in the data
  - E.g., when clustering handwritten digits we know there are 10 groups
- For other problems, we can't choose the number of groups a priori and we want it to estimate from the data
  - E.g., Dirichlet process

Parametric vs Non-Parametric Models

- Parametric models have a fixed number of parameters
- Non-parametric models have (theoretically) an infinite number of parameters
  - Very flexible
  - The data decides how many parameters are needed
  - E.g., Gaussian process (GP), Bayesian Additive Regression Trees (BART), Dirichlet process (DP)

Dirichlet Process

- Dirichlet distribution is the *n*-dimensional generalization of beta distribution
- Dirichlet process (DP) is the infinite-dimensional generalization of the Dirichlet distribution
  - A draw from a DP is actually a (discrete) Dirichlet distribution
- A DP has:
  - A base distribution $\mathcal{H}$ (e.g., Gaussian, Poisson, Laplace)
  - α a concentration parameter
    - Increasing α means less and less concentrated realization
    - For $\alpha \to \infty$ the realization of a DP are equal to the base distribution

Categorical Distribution

- The categorical distribution (the most general discrete distribution) is parameterized specifying the probabilities of each possible outcome
- It is specified by:
  - Indicating the positions on the x axis and the height on the y axis
  - x positions are integers
  - y heights must sum to 1
- A generalization of this is to have x sampled from $\mathcal{H}$
  - $\mathcal{H}$ to be a Gaussian, so x are on the real line
  - $\mathcal{H}$ to be a Beta, so x are in [0, 1]
  - $\mathcal{H}$ to be a Poisson, so x are non-negative integers 0, 1, 2, ...

Stick-Breaking Process

- $=$ process to generate values on the y axis so that the sum is 1
- You start with a stick of length 1
- Break it into two parts (not necessarily equal)
  - Use the "concentration parameter" α to control the size of the break
- Save the first part
- Pick the second part and break it again
- Repeat until you get K values

Dirichlet Process Using Stick-Breaking

- The locations are sampled from the base distribution
- The weights are controlled by α
- As $\alpha \to \infty$ the DP distribution approximates the base distribution

Infinite Mixture Model

- We can place a Gaussian on each data point and weight from a Dirichlet process
- The same approach can be use for Laplace distribution

- Concepts
- Coin Example
- Posterior-Based Decisions
- Groups Comparison
- Hierarchical Models
- Simple Linear Model
- Multiple linear regression
- Comparing Models
  - The Balance Between Simplicity and Accuracy
  - Measures of Predictive Accuracy
  - Regularizing priors
  - Regularizing Priors
  - Mixture Models
  - 7.5, Zero-Inflated and Hurdle Models
  - 7.6, Mixture Models and Clustering
  - 7.7, Non-Finite Mixture Models
  - ***7.8, Continuous Mixtures***
- Inference Engines

- Concepts
- Coin Example
- Posterior-Based Decisions
- Groups Comparison
- Hierarchical Models
- Simple Linear Model
- Multiple linear regression
- Comparing Models
- ***Inference Engines***
    - 10.1, Inference Engines

- Concepts
- Coin Example
- Posterior-Based Decisions
- Groups Comparison
- Hierarchical Models
- Simple Linear Model
- Multiple linear regression
- Comparing Models
- Inference Engines
  - ***10.1, Inference Engines***

Inference Engines as Black

- Bayesian methods are numerically challenging
  - Intractable / computationally intractable integral to solve
- Probabilistic programming separates:
  1. Model building
     - Users should not care how sampling is carried out
  2. Inference process (can be a black box, leave PyMC to handle computation)
- Understanding how the posterior is sampled can help understand how different methods can fail
  - E.g., diagnose samples