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)
```

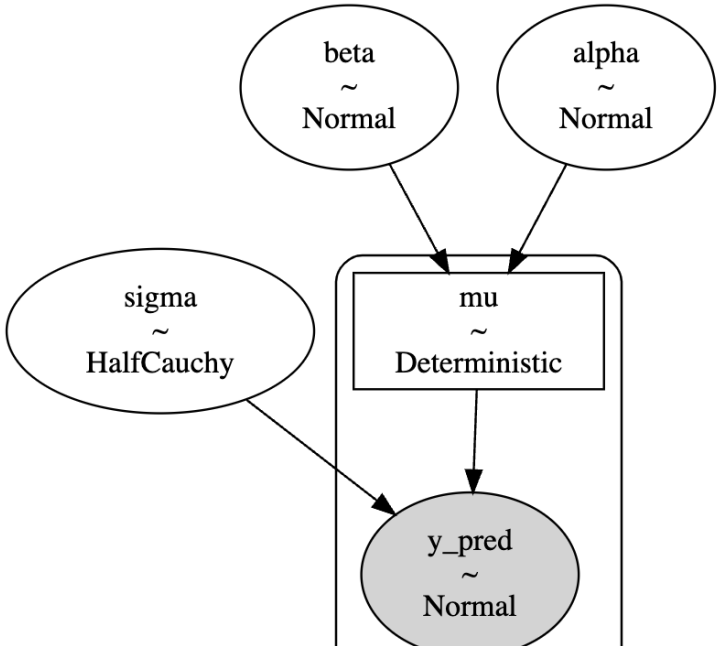
```
Auto-assigning NUTS sampler...
Initializing NUTS using jitter+adapt_diag...
Multiprocess sampling (4 chains in 4 jobs)
NUTS: [alpha, beta, sigma]
```

```
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)
```

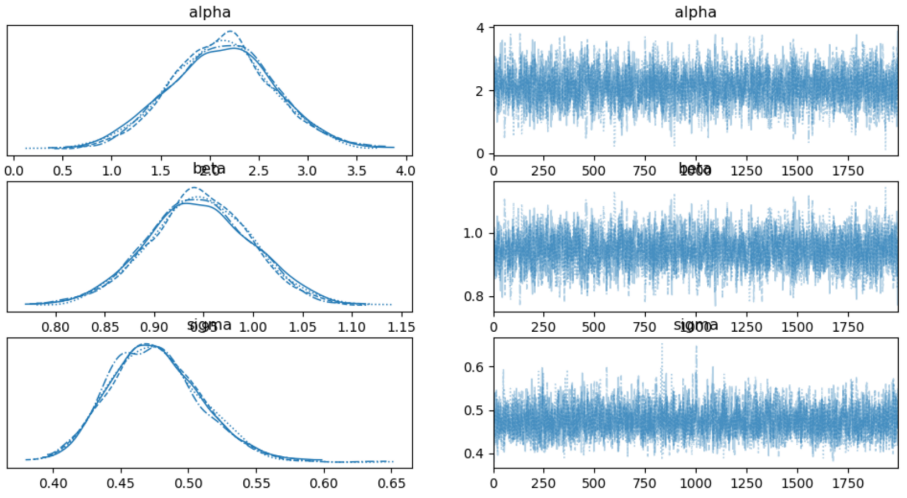
[127]:



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"]);
```



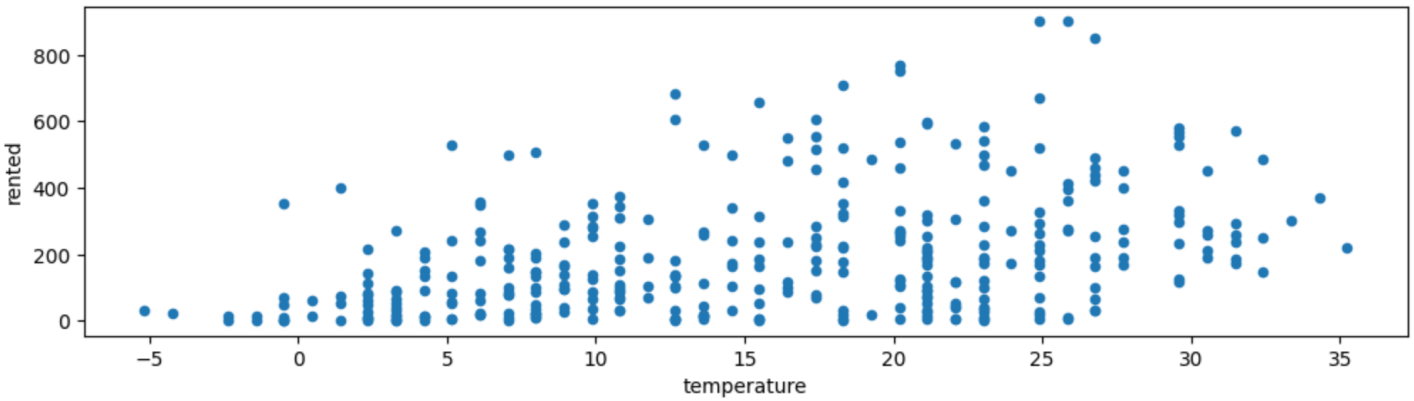
[128]:
```
az.summary(idata_g, var_names="alpha beta sigma".split(), kind="stats")
```

[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

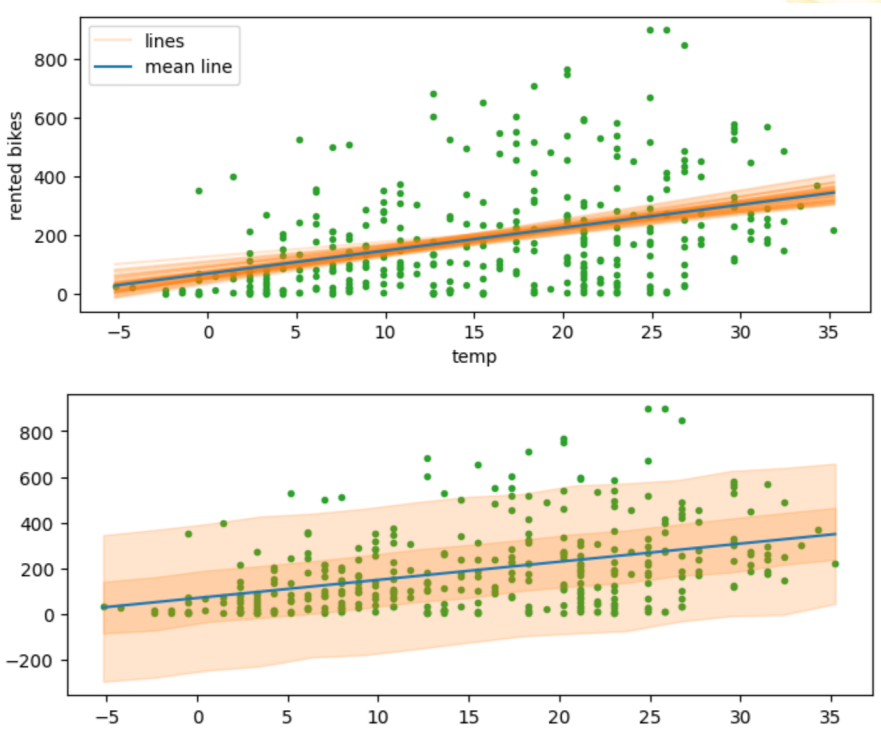


- Use intermediate variable $\mu = \alpha + \beta X$, mean number of bikes rented for temperature X

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

Linear Model: Bike Rental Example

- **Fit model with PyMC**: $\mu = 69 + 7.9X$
- **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**
  1. Model outputs negative bike numbers
  2. Model predicts real numbers, but count is discrete
- **Root cause**: Not surprising since using Gaussian likelihood:
  - Gaussian extends to negative numbers
  - Gaussian is continuous
- **Solutions**:
  - Hack: clip predictions below 0 and discretize output
  - Elegant: use model defined only for discrete positive numbers

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

Generalized Liner Model (GLM)

- **Generalization of linear model** using different distributions for likelihood:

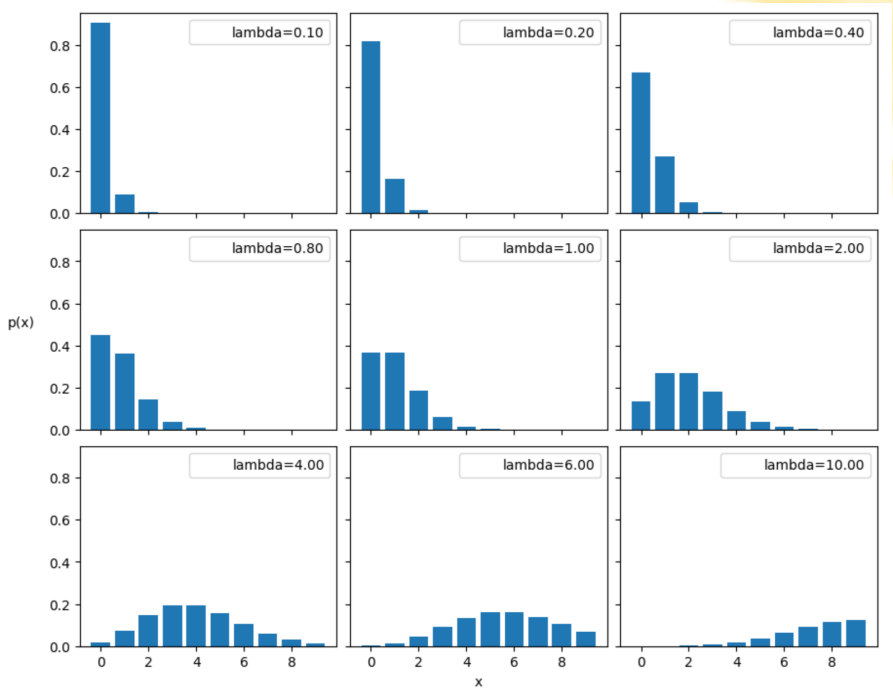
$$\begin{cases} \alpha \sim \textit{prior} \\ \beta \sim \textit{prior} \\ \mu = \alpha + \beta X \\ \theta \sim \textit{prior} \\ Y \sim \phi(f(\mu), \theta) \end{cases}$$

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

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")

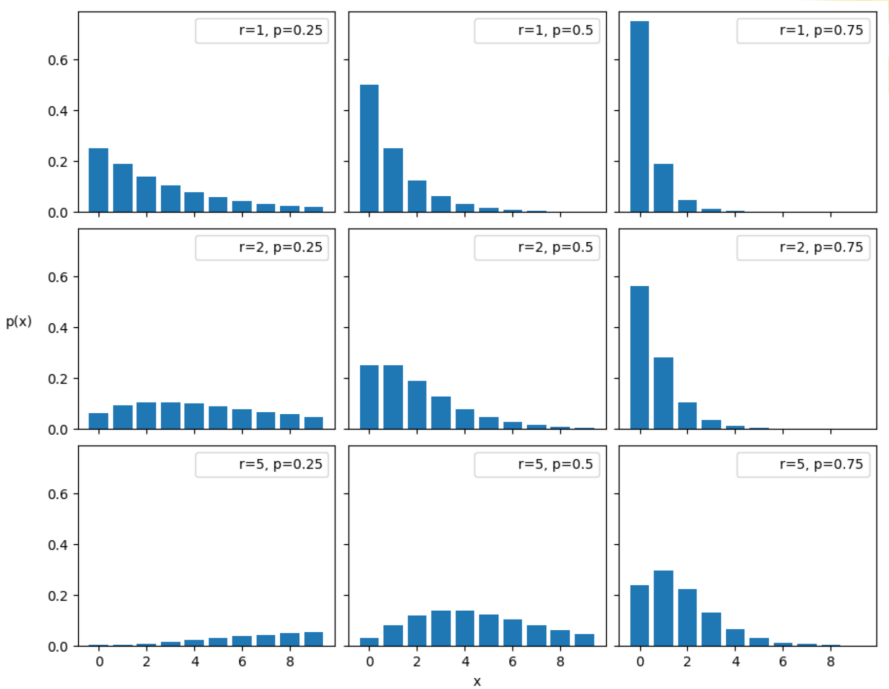


Overdispersion

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

Negative Binomial Distribution

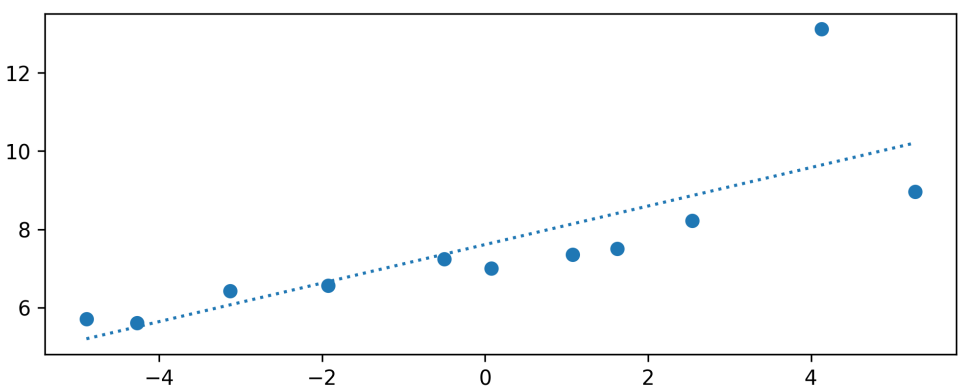
- **Negative binomial distribution**
  - Models failures/trials to achieve fixed number of successes in IID Bernoulli trials
  - Generalizes geometric distribution, modeling trials to first success
  - Models overdispersion (mean $\ll$ variance)



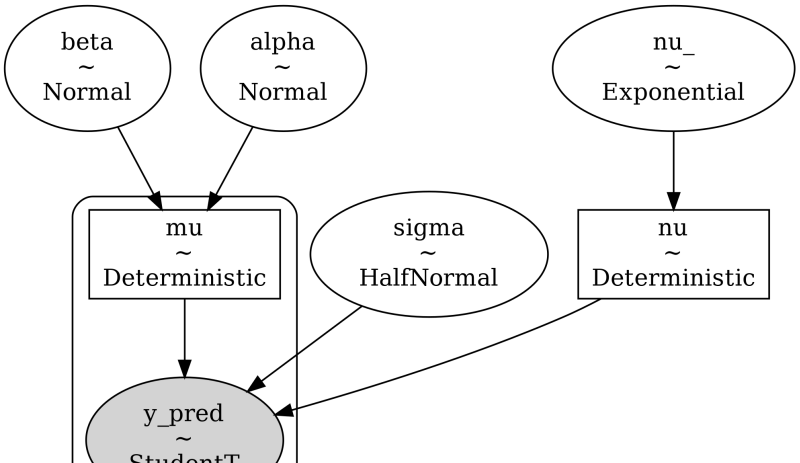
- **E.g.,**
  - Customer service: unsuccessful interactions before success
  - Sports: games lost before winning a fixed number of games

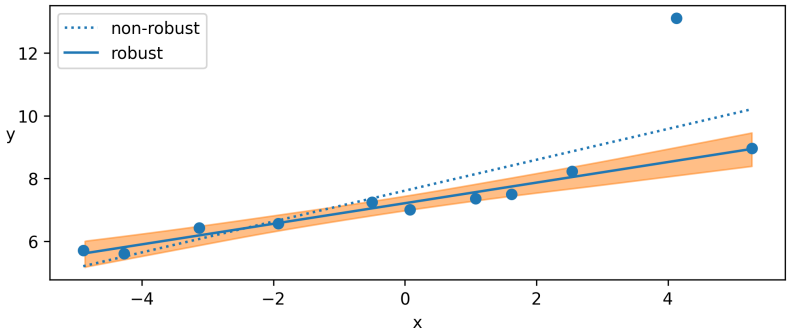
Robust Regression

- Outliers pull regression line away from bulk of data
- Robust regression is just another generalized linear model
  - Instead of some feature transform / winsorization like in the hacker ML



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





- ***Logistic Regression***
- Multiple Linear Regression
- Comparing Models

Logistic Regression

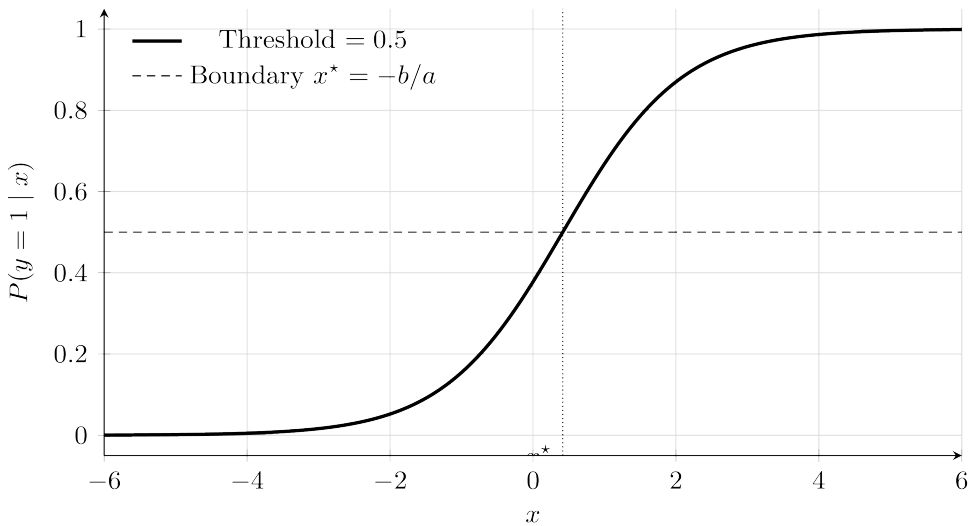
- **Logistic regression model**
  - Is a generalized linear model
  - Models the response variable as binary
    - E.g., ham/spam, safe/unsafe, cloudy/sunny, hotdog/not hotdog
- The **logistic model** is:

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

- 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 of 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.,

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

- Same treatment as frequentist approach, since it has a Bayesian flavor
  - But once it's Bayesian, you can generalize even more!
  - No gradient descent, but solved with Bayesian inference
- **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 the ratio between favorable vs unfavorable events

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

- Transformation 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:

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

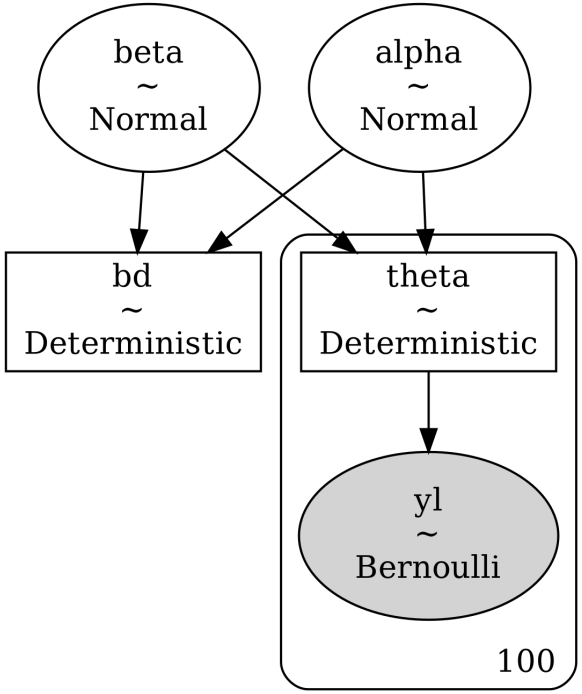
  - Logistic regression models log-odds as linear regression
  - β is the increase of log-odds for a unit change of x

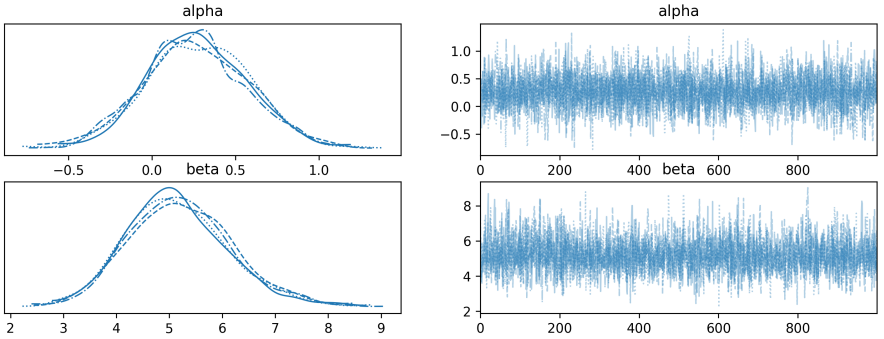
Classification with Logistic Regression: Bayesian

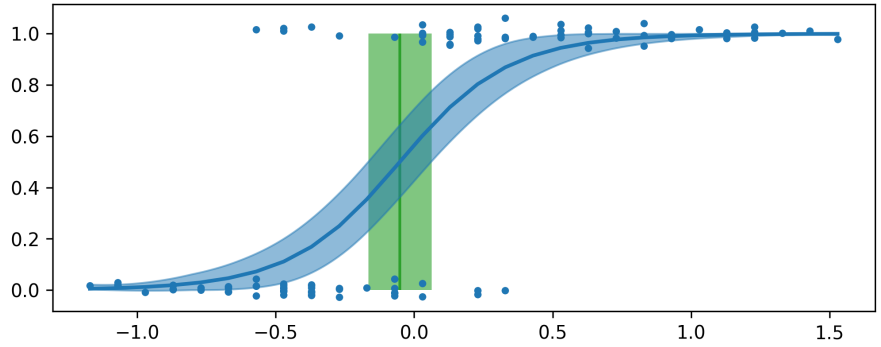
```
with pm.Model() as model_lrs:
    # Linear part.
    alpha = pm.Normal("alpha", mu=0, sigma=1)
    beta = pm.Normal("beta", mu=0, sigma=5)
    mu = alpha + x_c * beta
    # Sigmoid.
    theta = pm.Deterministic("theta", pm.math.sigmoid(mu))
    # Model.
    yl = pm.Bernoulli("yl", p=theta, observed=y_0)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

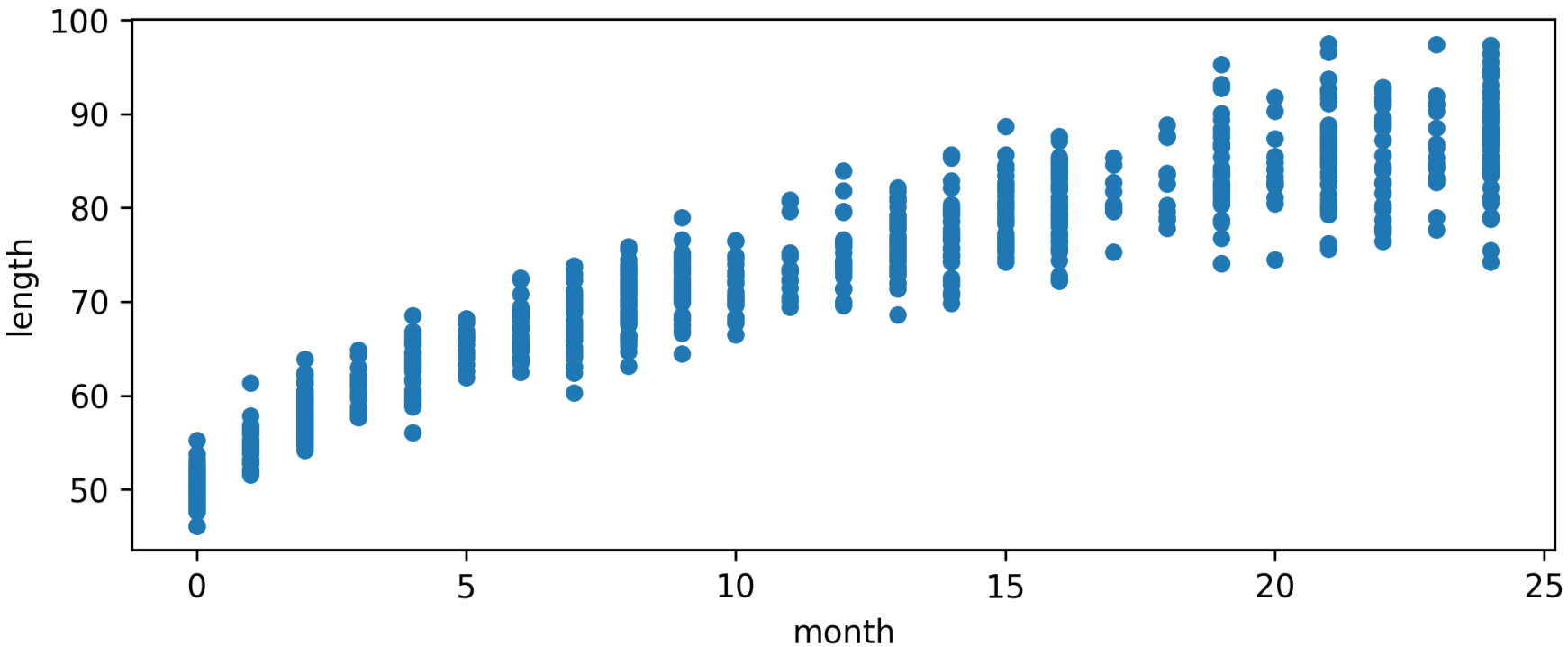






Heteroskedasticity

- **Heteroskedasticity** = when variance of errors is not constant across observations
  - E.g., variance can be a linear function of the dependent variable
- **Example**:
  - 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

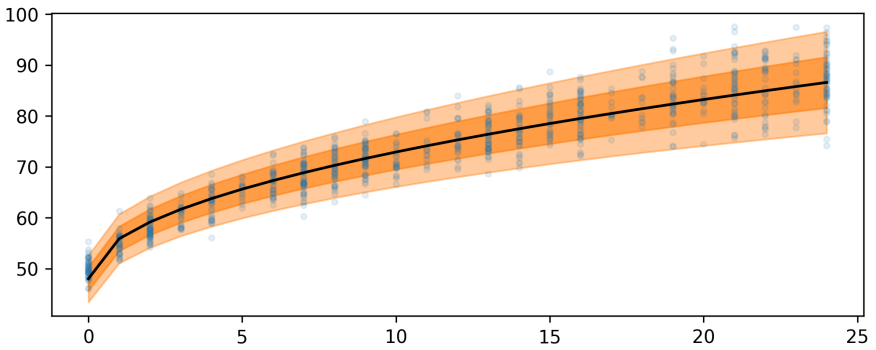


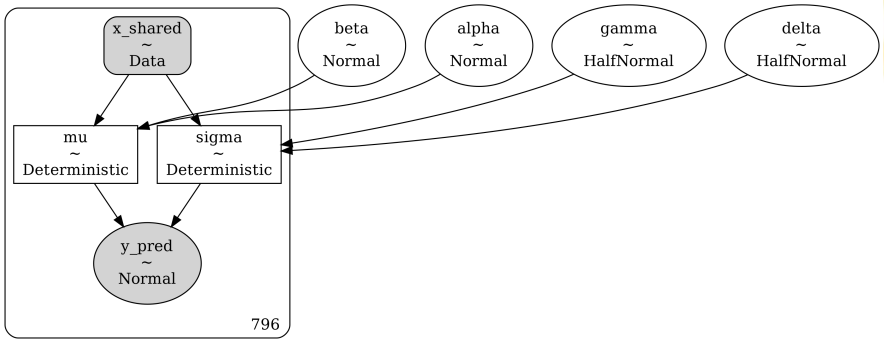
Heteroskedasticity: Bayesian Model

```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%
```





SCIENCE ACADEMY

- Logistic Regression
- ***Multiple Linear Regression***
- Comparing Models

Multiple Linear Regression

- In prediction problems, it is common to have several independent variables
  - 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

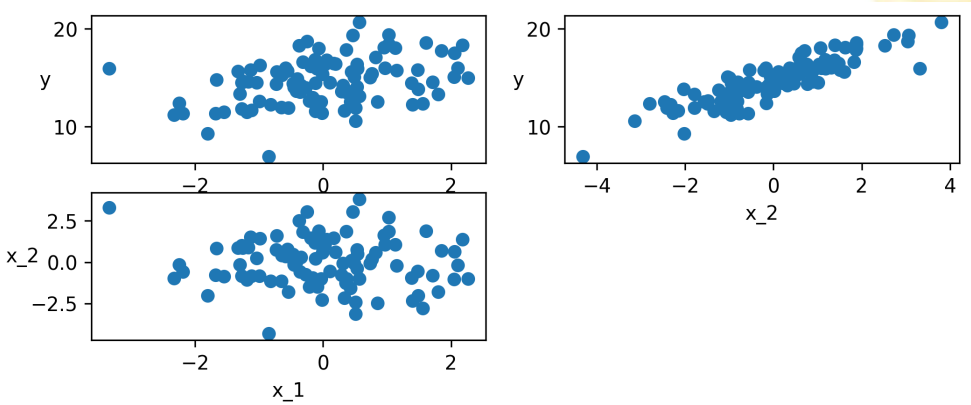
$$\mu = \alpha + \beta_1 x_1 + \beta_2 x_2 + ... + \beta_k x_k$$

  - 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
```

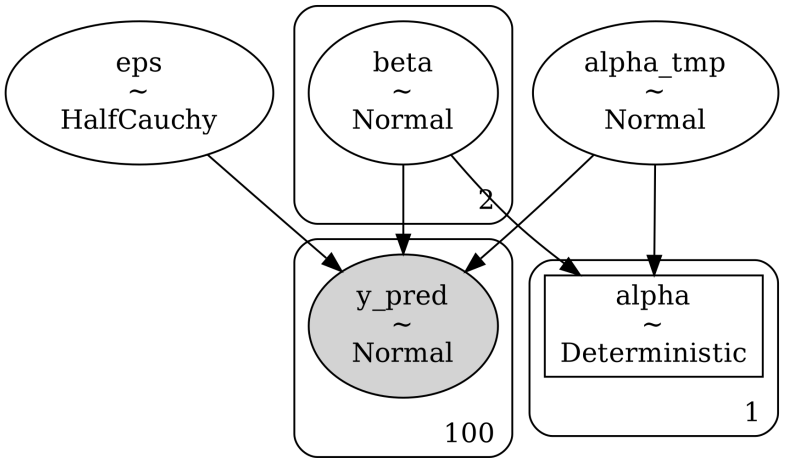


- Create PyMC model

```
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)
```

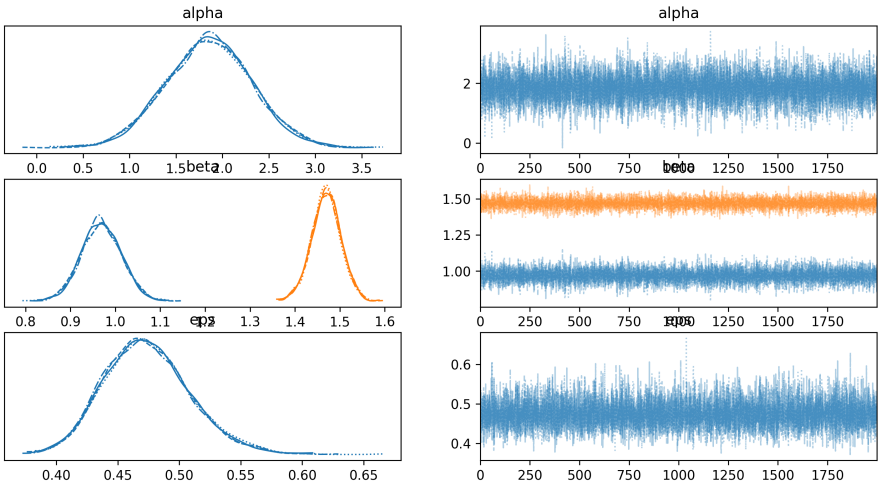
```
Auto-assigning NUTS sampler...
Initializing NUTS using jitter+adapt_diag...
Multiprocess sampling (4 chains in 4 jobs)
NUTS: [alpha_tmp, beta, eps]
```

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



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

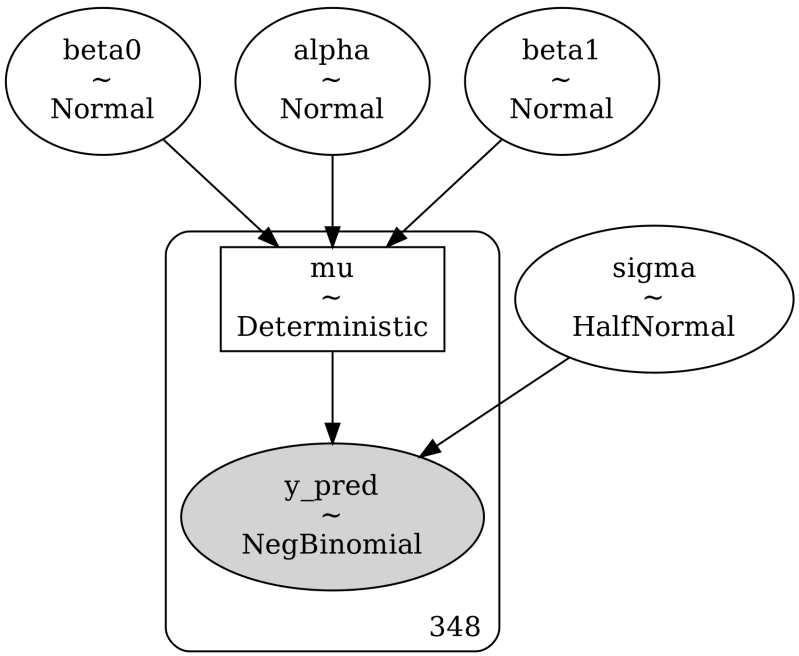
Multiple Regression: Rented Bike Example 1/2

- **Assumption**: number of bike rented is function of temperature and hour of the day
- Create PyMC 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()
```

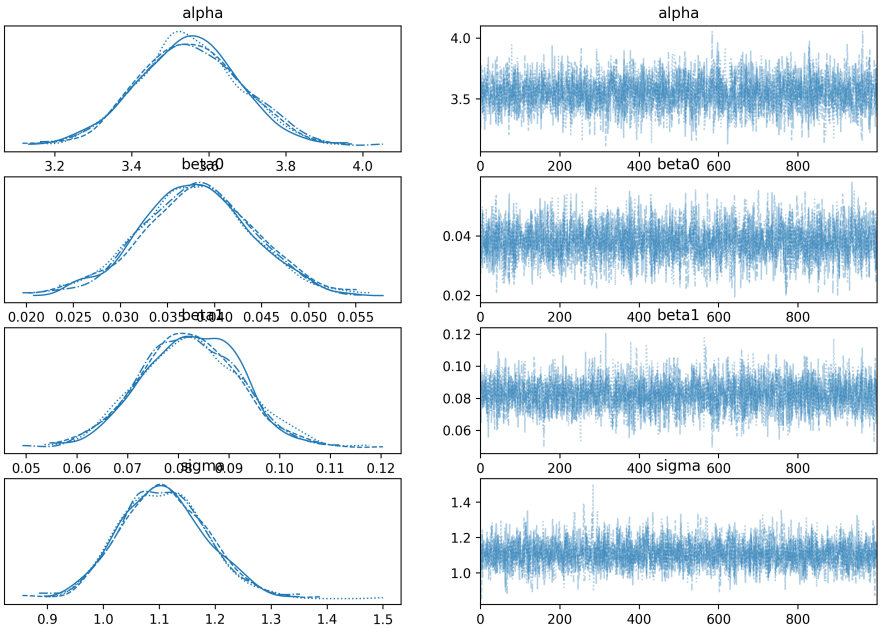
```
Auto-assigning NUTS sampler...
Initializing NUTS using jitter+adapt_diag...
Multiprocess sampling (4 chains in 4 jobs)
NUTS: [alpha, beta0, beta1, sigma]
```

```
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

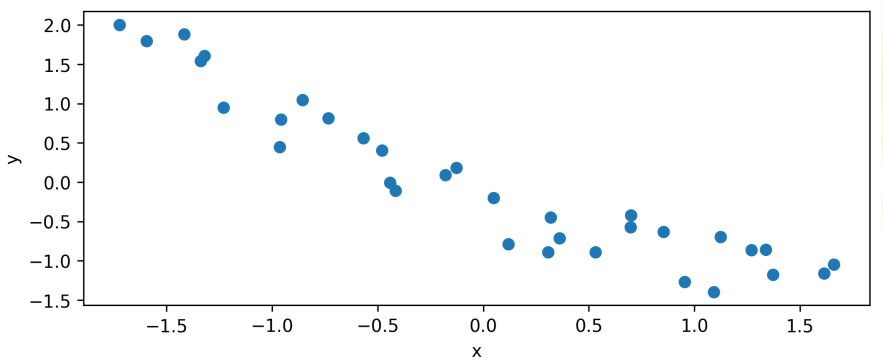
- Logistic Regression
- Multiple Linear Regression
- ***Comparing Models***
  - The Balance Between Simplicity and Accuracy
  - Measures of Predictive Accuracy

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

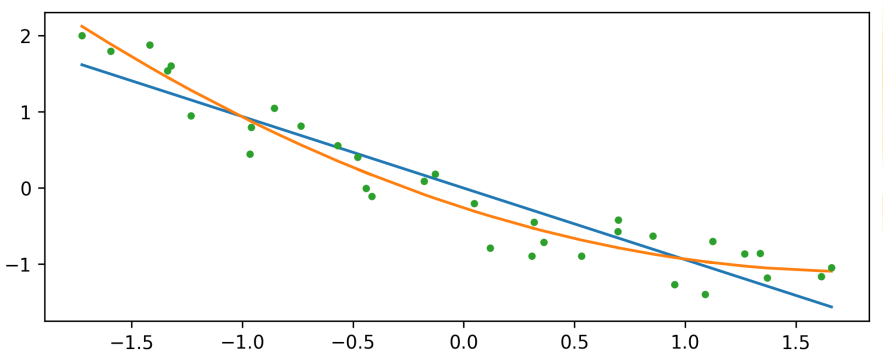
- Goal of PPC:
  - Evaluate model's data explanation
  - Understand model limitations
  - Improve model
- Given data from parabola + noise:
  - Fit with linear model
  - Fit with quadratic model
  - Compare predicted (posterior) vs observed data

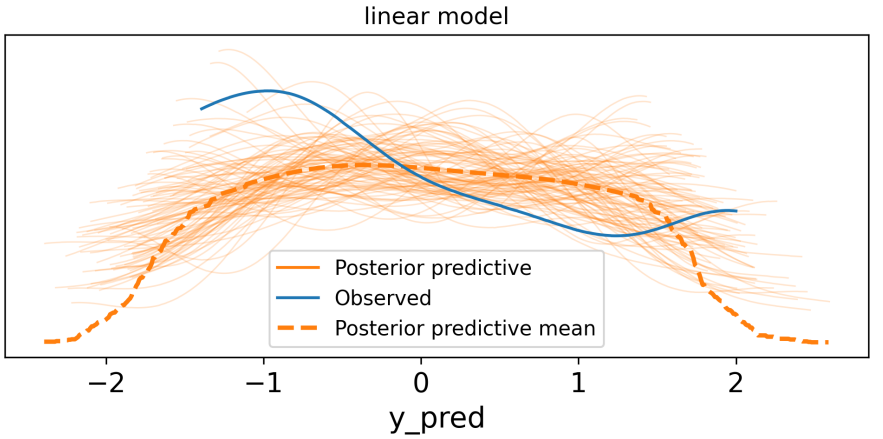


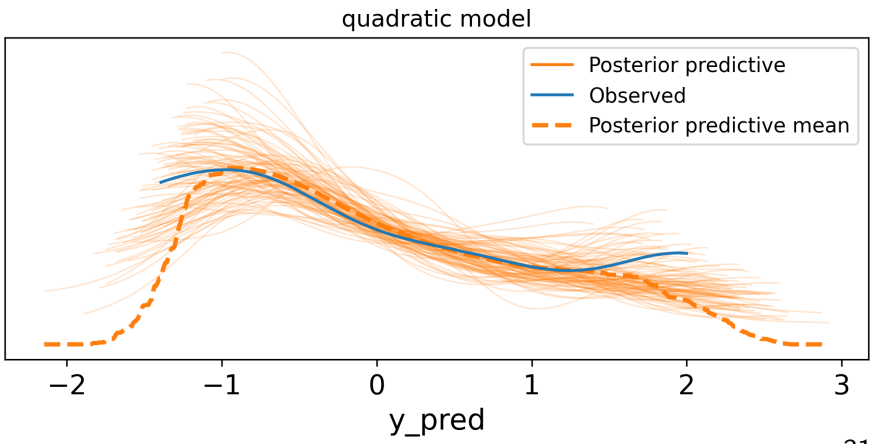

```python
# Linear model.
with pm.Model() as model_l:
    # mu = alpha + beta * x
    alpha = pm.Normal('alpha', mu=0, sigma=1)
    beta = pm.Normal('beta', mu=0, sigma=10)
    mu = alpha + beta * x_c[0]
    #
    sigma = pm.HalfNormal('sigma', 5)
    #
    y_pred = pm.Normal('y_pred', mu=mu, sigma=sigma, 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







SCIENCE ACADEMY

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

- Logistic Regression
- Multiple Linear Regression
- Comparing Models
  - ***The Balance Between Simplicity and Accuracy***
  - Measures of Predictive Accuracy

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

- Logistic Regression
- Multiple Linear Regression
- Comparing Models
  - The Balance Between Simplicity and Accuracy
  - ***Measures of Predictive Accuracy***

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