"YOU DON'T ALWAYS NEED A PLAN. SOMETIMES YOU JUST NEED TO BREATHE, TRUST, LET GO AND SEE WHAT HAPPENS."
― **MANDY HALE**

BAYESIAN STATISTICS

CLASS 1

GOALS FOR TODAY’S LECTURE

- Review terminology
  - Prior distribution
  - Sampling distribution
  - Posterior distribution
  - Marginal distribution of the data
- Understand simple coding in Stan
- Run a simple analysis in Stan

TERMINOLOGY

Sampling distribution: P(Y| parameters)
(for example, $P(Y|\boldsymbol{\mu},\boldsymbol{\sigma}^2)$ or $P(Y|\mathbf{p})$ or $P(Y|\boldsymbol{\lambda})$)

Prior distribution: P(parameter)
(for example, $P(\boldsymbol{\mu})$, $P(\boldsymbol{\sigma}^2)$, $P(\mathbf{p})$ or $P(\boldsymbol{\lambda})$)

Posterior distribution: P(parameters|Y)
(for example, $P(\boldsymbol{\mu},\boldsymbol{\sigma}^2|Y)$ or $P(\mathbf{p}|Y)$ or $P(\boldsymbol{\lambda}|Y)$)

HOW IT ALL FITS TOGETHER (BAYES RULE)

Posterior distribution

Sampling distribution

Prior distribution

$$P(p|Y) = \frac{P(Y|p)P(p)}{P(Y)}$$

Marginal distribution of Y

GOAL: POSTERIOR DISTRIBUTION

DATA

+

PRIOR

=

POSTERIOR

ONLY FOCUS ON DISTRIBUTIONS (DENOMINATOR IS THE "NORMALIZING CONSTANT")

$$P(p|Y) \propto P(Y|p)P(p)$$

DISTRIBUTIONS (DO NOT NEED MATH!!)

- Focus on characteristics of data to decide distributions
    - What is the support? (in other words, what values can this data take on?)
    - Is it discrete or continuous

COMMON DISTRIBUTIONS

- Counting number of successes (this means that you want to estimate a proportion!) – Binomial
- Count data (number of bikes rented within a given hour, number of diseased trees in an acre, number of customers in a day, etc) – Poisson, Negative Binomial
- ONLY positive data (continuous) – Gamma or Inv-Gamma
- Continuous - Normal

COMMON PRIORS

- In Binomial distribution, only have p (a proportion) – Beta distribution (noninformative: Beta(1,1))
- In Poisson distribution, only have λ (a mean….this mean can ONLY be positive) – Gamma (noninformative: Gamma(0.001,0.001))
- Gamma distribution, α and β (both need to be positive) – use Gamma for both (noninformative: Gamma(0.001, 0.001))
- Normal distribution, μ and σ (μ is all real values and σ is only positive) – Normal for μ and Inverse-Gamma for σ (noninformative: Normal(0, 10000) and Gamma(0.001,0.001))

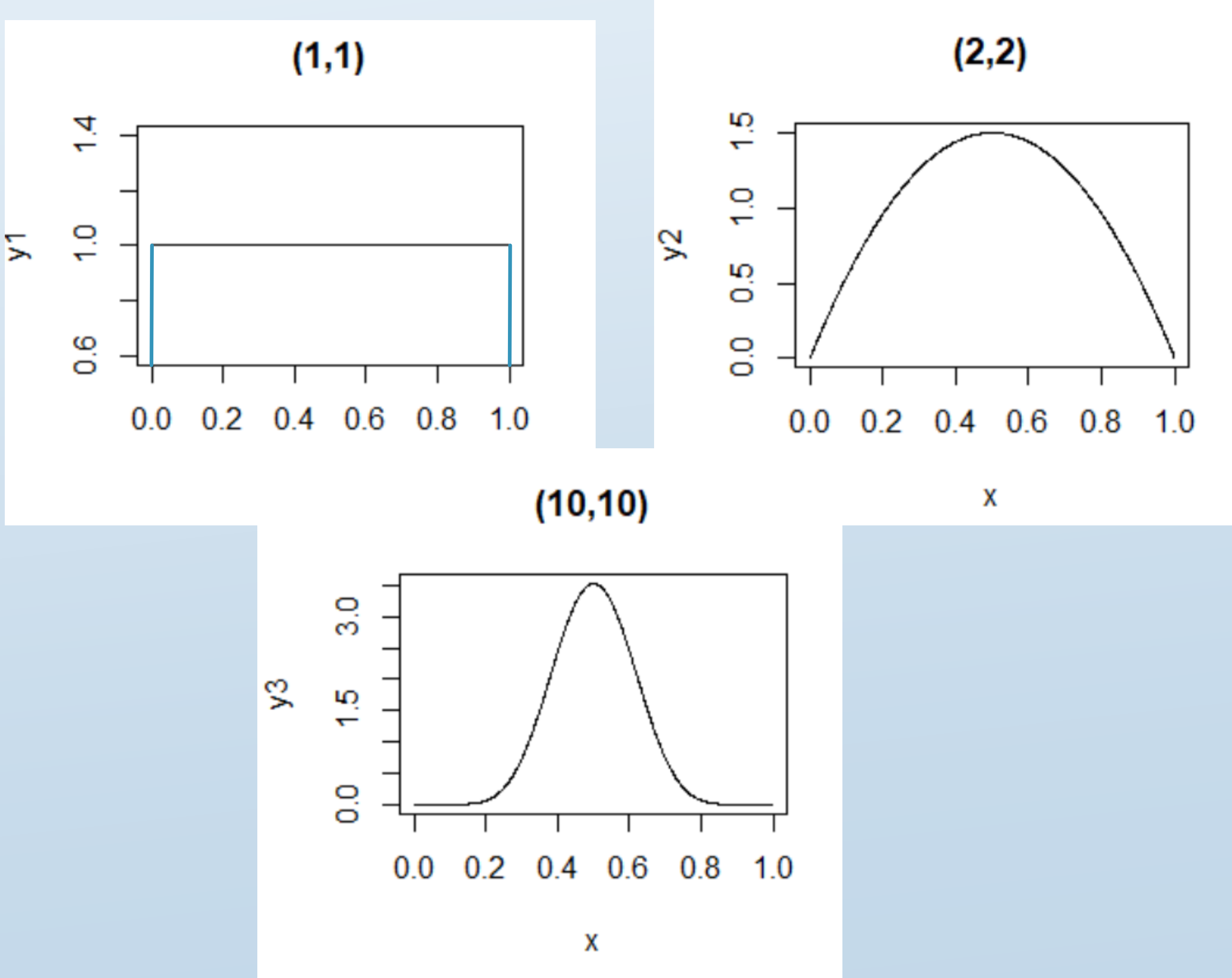
- NOTE: Sometimes a χ^2 or even Inverse-χ^2 is used instead of Gamma (this is a special form of the Gamma distribution)

SIMPLE EXAMPLE

- Want to estimate the proportion of students at NCSU who voted in 2020 Democratic primary
  - What information will we gather?
  - Sampling distribution:
  - Parameters:
  - Prior distribution:

EXPLORE THE BETA DISTRIBUTION

```
x<-seq(0.001,0.999,length=1000)
y1<-dbeta(x,1,1)
plot(x,y1,type='l‘, main=‘(1,1)’)
y2<-dbeta(x,2,2)
plot(x,y2,type='l‘,main=‘(2,2)’)
y3<-dbeta(x,10,10)
plot(x,y3,type='l‘,main=‘(10,10)’)
```



USING PYMC

STEPS IN DOING A BAYESIAN STATISTICS

- Decide what type of data is being collected – (this will decide sampling distribution)
- Figure out parameters in the sampling distribution (set prior distributions)
- Put information into PyMC
- Make sure you have convergence of chains for posterior distribution
- Use posterior distribution to answer questions

PYMC: DEFINING MODEL

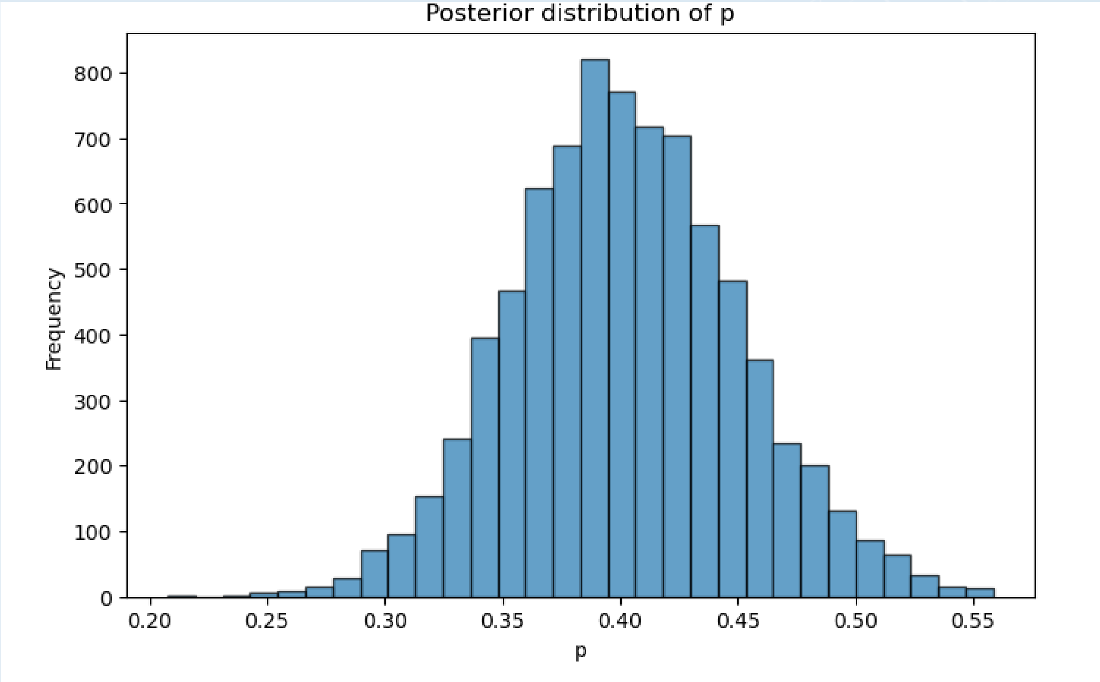
```
##Get packages:
import pymc as pm
import arviz as az
# Provide data
n = 100
y = 40
# Model
with pm.Model() as binom_model:
    p = pm.Beta("p", alpha=1, beta=1)    ### Prior distribution
    y_obs = pm.Binomial("y_obs", n=n, p=p, observed=y)    ### Sampling distribution
    trace = pm.sample(2000, return_inferencedata=True, random_seed=18569) ### Run the code
az.summary(trace) ### Summary
```

PACKAGES IN PYTHON:

- You will need to install pymc and pytensor

EXTRACT POSTERIOR SAMPLES

```
import matplotlib.pyplot as plt
# Extract posterior samples
posterior_samples = trace.posterior["p"].values.flatten()
# Plot histogram
plt.figure(figsize=(8, 5))
plt.hist(posterior_samples, bins=30, edgecolor='black', alpha=0.7)
plt.title("Posterior distribution of p")
plt.xlabel("p")
plt.ylabel("Frequency")
plt.show()
```



GET INFORMATION FROM POSTERIOR SAMPLES

import numpy as np

Compute probability of p <= 0.3 prob_p_leq_0_3 = np.mean(posterior_samples <= 0.3) print(f"Probability that p <= 0.3: {prob_p_leq_0_3:.4f}") # Compute 95% credible interval cred_interval = np.quantile(posterior_samples, [0.025, 0.975]) print(f"95% Credible Interval: {cred_interval}")

Probability that p <= 0.3: 0.0160

95% Credible Interval: [0.31026572 0.50097313]

IN CLASS EXAMPLE

- A political science student wants to estimate the proportion of students at NCSU who voted in the 2020 election. Identify the sampling distribution, the number of parameters and potential prior(s).
- The student gathered a sample of 150 students of which 100 indicated that they did vote.
- Get the posterior distribution(s) of the parameter(s) and find 95% probability interval(s). Assume a uniform prior for p.
- See class example on Moodle for more information and questions.

USING RSTAN

STEPS IN DOING A BAYESIAN STATISTICS

- Decide what type of data is being collected – (this will decide sampling distribution)
- Figure out parameters in the sampling distribution (set prior distributions)
- Put information into STAN
- Make sure you have convergence of chains for posterior distribution
- Use posterior distribution to answer questions

MODEL INFO IN STAN (ALWAYS NEED THESE 3 SECTIONS)

Data

Parameters

Model

MODEL INFO IN STAN

Data

This is where you define your data (integer, real, are there any bounds on information here?)

Parameters

Model

MODEL INFO IN STAN (SEPARATE FILE)

Data

This is where you define your data (integer, real, are there any bounds on information here?)

Parameters

This is where you will define all of your parameters in the analysis (if not defined here, it will get confused)

Model

MODEL INFO IN STAN (SEPARATE FILE)

Data

This is where you define your data (integer, real, are there any bounds on information here?)

Parameters

This is where you will define all of your parameters in the analysis (if not defined here, it will get confused)

Model

This is where you will define your model (all priors and sampling distributions)

DATA

```
Data {
Int <lower=0, upper=1> y;
Real <lower=0, upper=1> y;

}
```

You can also indicate lower values and upper values for data (will give an error if someone tries inputting values that go beyond the limit).

DATA

```
Data {
Int <lower=0> n;
Real y[n];
Vector [n] y;

}
```

When your data is a vector (more than one observation), there are two ways to specify this.

DATA

```
Data {
Int <lower=0> n;
Int <lower=0> m;
Real y[n,m];
matrix [n,m] y;

}
```

When your data is a matrix (for example a dataframe), there are two ways to specify this. This data frame has n rows and m columns.

PARAMETERS

```
parameters{
  real alpha;
  vector[5] beta;
  real<lower=0> sigma;
}
```

This is where you define ALL your parameters!! You can define them as just one number, a vector of numbers or a dataframe (same notation that was used in the “Data” section)

MODEL

```
model {
    p ~ beta(1,1);
    y ~ binomial(n, p);
  }
```

This is where you define all of your prior distributions and sampling distributions.

STAN

- These three sections MUST appear for your STAN code!! Many different ways of creating a STAN program (can put it in an external file….must have extensions .stan and have a blank line at the end)
- You can also code directly in R (which is how I will be showing it). You MUST have quotations at beginning and end of STAN code!!!
- Besides the STAN code, you need to organize your data into a list
    - For example: binom.data=list(n=100, y=40)

DATA

STAN file

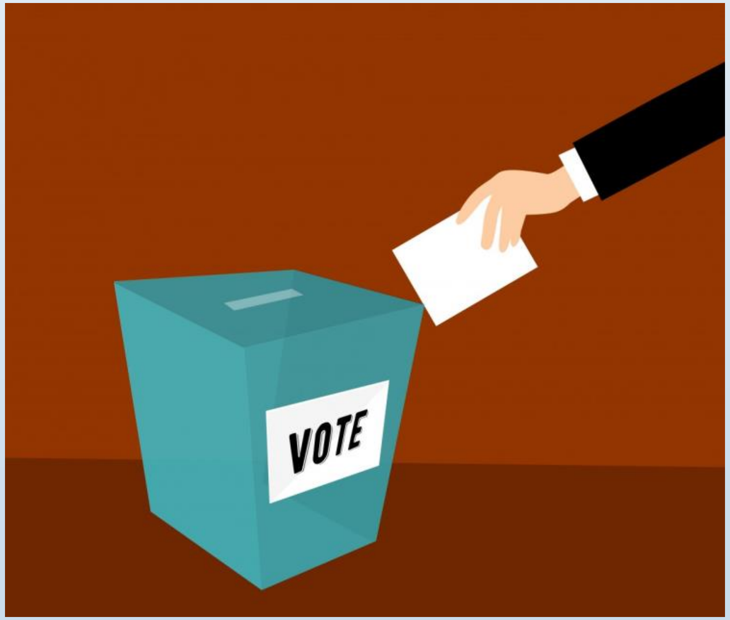
```
data{
 int <lower=0> n;
 vector[n] y;
 matrix[n,5] x;
}
```

R code

```
regress.dat=list(n=nrow(x),x=x,y=ameshousing$Sale_Price)
```

These two have to match up. Notice that both contain:
n, y and x (all with matching dimensions!)

CODE EXAMPLE IN STAN



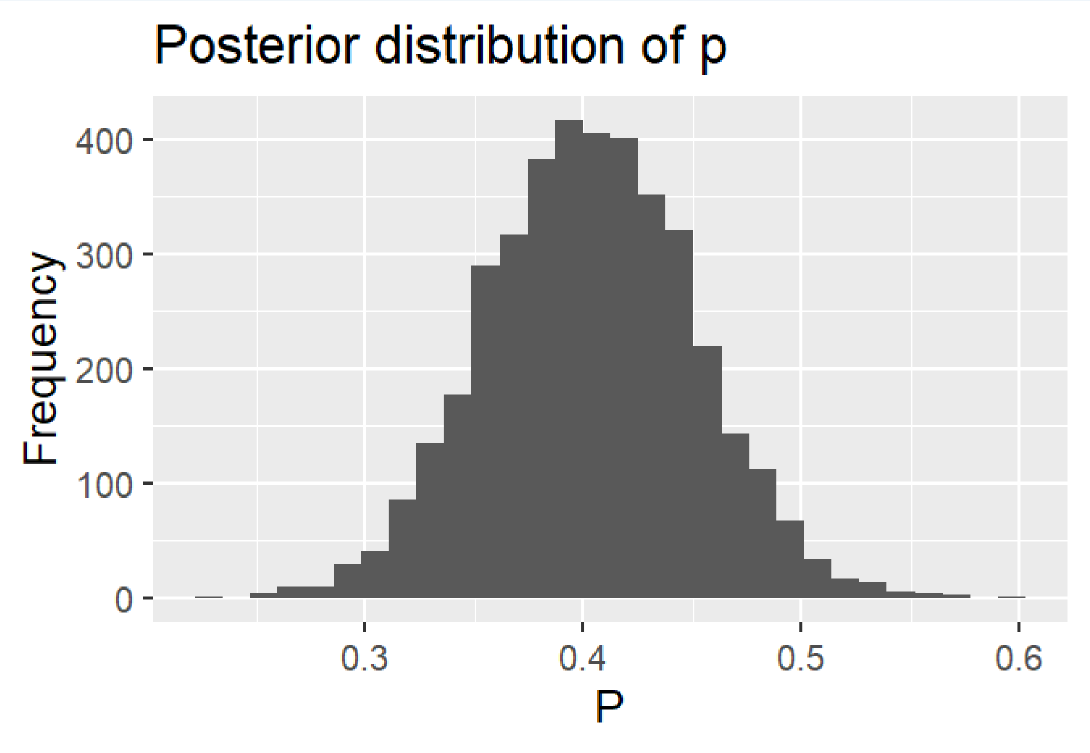
Going back to example

- Want to estimate the proportion of students who voted in 2020 democratic primary
- Sampling distribution: binomial (p)
- Prior distribution: Beta(1,1)
- Data: value for y (number who voted) and n (total sample)

CODE FOR EXAMPLE

```
ex1 <- "
data {
        int <lower=0> y;
        int <lower=0> n;
        }
parameters {
        real <lower=0, upper=1> p;
                    }
model {
        p ~ beta(1,1);
        y ~ binomial(n, p);
          }
“
binom.data=list(n=100, y=40)
binom.stan=stan(model_code = ex1,data=binom.data,seed=18569)
```

GET POSTERIOR SAMPLES




```
post.samp.binom=extract(binom.stan)

new.p=post.samp.binom$p

p.post=data.frame(new.p)

ggplot(p.post,aes(new.p))+geom_histogram()+labs(
title="Posterior distribution of
p",y="Frequency",x="P")
```

GET INFORMATION ABOUT P

```
###Probability p is lower than 0.30
> sum(new.p<=0.3)/length(new.p)
[1] 0.015
###95% Probability Interval
> quantile(new.p,p=c(0.025,0.975))
     2.5%     97.5%
0.3118352 0.4960807
```

IN CLASS EXAMPLE

- A political science student wants to estimate the proportion of students at NCSU who voted in the 2020 election. Identify the sampling distribution, the number of parameters and potential prior(s).
- The student gathered a sample of 150 students of which 100 indicated that they did vote.
- Get the posterior distribution(s) of the parameter(s) and find 95% probability interval(s). Assume a uniform prior for p.
- See class example on Moodle for more information and questions.