Lecture 10: Distributions as Models

STAT GU4206/GR5206 *Statistical Computing & Introduction to Data Science*

Gabriel Young
Columbia University

November 15, 2019

Course Notes

Last Time

- ✓ **Permutation test**. Testing if two distributions are the same.
- ✓ In-class example: Non-parametric version of the two-sample t-test.
- **Empirical size**. Simulation based validation of the classic 2-sample t-sample.
- Lab: Sampling distribution, distribution of p-value, empirical size and power.

This Time

- **Distributions as Models:**
- Method of Moments
- Maximum Likelihood Estimation
- Bayesian Estimation
- Markov Chain Monte Carlo (MCMC)

Method of Moments

Cats

The cats dataset includes the heart and body weights of samples of male and female cats. All the cats are adults and over 2 kg in body weight.

```
> # install.packages("MASS")
> library(MASS)
> head(cats)
```

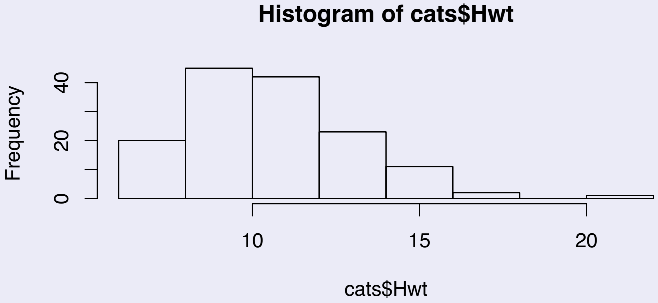
```
  Sex Bwt Hwt
1   F 2.0 7.0
2   F 2.0 7.4
3   F 2.0 9.5
4   F 2.1 7.2
5   F 2.1 7.3
6   F 2.1 7.6
```

The Distribution of the Data

We've studied how to visually inspect the distribution of a continuous random variable.

```
> hist(cats$Hwt)
```

Gamma



The Distribution of the Data

R Functions to study the Data's Distribution

`quantile(x, probs)` calculates the quantiles at `probs` from `x`.

```
> quantile(cats$Hwt, c(0.25, 0.5, 0.75))
   25%    50%    75%
 8.950 10.100 12.125
```

The Distribution of the Data

R Functions to study the Data's Distribution

`ecdf()`: **e**mprical **c**umulative **d**istribution **f**unction. No assumptions but also no guess about the distribution beyond the observations.

- In math ECDF is written as $\hat{F}$ or $\hat{F}_n$
- Conceptually, `quantile()` and `ecdf()` are inverses to each other.

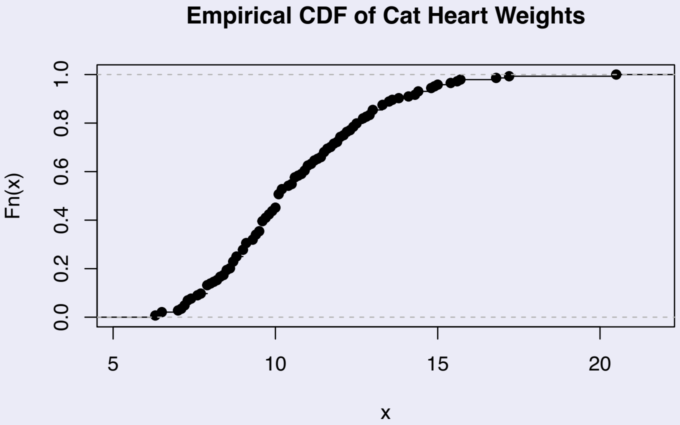
qnorm ( ) ⟷ pnorm

$P(Z < ?) =$ $F(z)$

quantile ecdf

The Distribution of the Data

```
> plot(ecdf(cats$Hwt),
+       main = "Empirical CDF of Cat Heart Weights")
```



The Distribution of the Data

R Functions to study the Data's Distribution

`density(x)`: estimates the density of `x` by counting how many observations fall in a little window around each point, then smoothing.

- "Bandwidth" = width of window around each point
- AKA calculates a 'kernel density estimate'
- `density()` returns a collection of x, y values suitable for plotting

The Distribution of the Data

R Functions to study the Data's Distribution

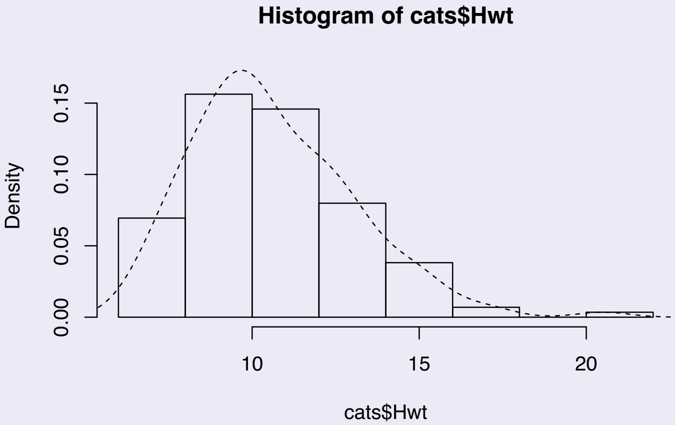
`density(x)`: estimates the density of `x` by counting how many observations fall in a little window around each point, then smoothing.

- "Bandwidth" = width of window around each point
- AKA calculates a 'kernel density estimate'
- `density()` returns a collection of x, y values suitable for plotting

Note, `density()` is an *estimate* of the pdf, not the truth.

The Distribution of the Data

```
> hist(cats$Hwt, probability = TRUE, ylim = c(0, 0.17))
> lines(density(cats$Hwt), lty = "dashed")
```



Why Do We Care About the Distribution of the Data?

- The data itself is too much information and *overly detailed*. Don't need to keep around every single data point.
- Plus, the exact data would never repeat itself if we re-sampled anyways.

Why Do We Care About the Distribution of the Data?

- The data itself is too much information and *overly detailed*. Don't need to keep around every single data point.
- Plus, the exact data would never repeat itself if we re-sampled anyways.
- **Goal**: Store information that *summarizes* what will *generalize* to other situations.
    - Can do this by using a model and only keeping the model's parameters.

How Do We Fit Distributional Models to Data?

Recall that most models are defined by *parameters* (like (μ, σ^2) for the normal). So *fitting* a model to data means finding those parameters such that the model best fits the data.

How Do We Fit Distributional Models to Data?

Recall that most models are defined by *parameters* (like (μ, σ^2) for the normal). So *fitting* a model to data means finding those parameters such that the model best fits the data.

- Match moments (mean, variances, etc.).
- Match other summary statistics.
- Maximize the likelihood.

Method of Moments Estimation

Recall the Gamma Distribution

$$f(x)=\frac{1}{\beta^{\alpha}\Gamma(\alpha)}x^{\alpha-1}e^{-\frac{x}{\beta}}$$

- The **gamma** distributions are a family of probability distributions defined by the density functions,

$$f(x)=\frac{x^{a-1}e^{-x/s}}{s^a\Gamma(a)},$$

  where the gamma function $\Gamma(a)=\int_0^\infty u^{a-1}e^{-u}du$ is chosen so that the total probability of all non-negative x is 1.
- Parameter a is the **shape**, and s is the **scale**. rate $=\lambda=\frac{1}{\beta}$
- The expected value is as, and the variance as^2.

$\alpha\beta$ $\alpha\beta^2$

Method of Moments

- Pick enough moments that they *identify* the parameters. At least one moment per parameter.
- Write equations for the moments in terms of the parameters.
  - E.g. for gamma,

$$\overset{\text{sample}}{\mu} = \underset{\text{parameter}}{a}s, \quad \sigma^2 = as^2.$$

- Solve the moment equations for the parameters (usually done by hand).
  - E.g. for gamma,

$$\hat{a} = \frac{\mu^2}{\underset{\text{sample}}{\sigma^2}}, \quad \hat{s} = \frac{\sigma^2}{\underset{\text{sample}}{\mu}}.$$

$\bar{X} = \alpha\beta$

$S^2 = \alpha\beta^2$

Check Yourself

Tasks

- Write a function `gamma.MMest` that takes as input a data vector and returns estimates of the scale parameters a and s using the moment equations from the previous slide.
- Plug cat heart weights into your function to get estimates of a and s.

Check Yourself

Tasks

- Write a function gamma.MMest that takes as input a data vector and returns estimates of the scale parameters *a* and *s* using the moment equations from the previous slide.
- Plug cat heart weights into your function to get estimates of *a* and *s*.

Tasks

```
> gamma.MMest <- function(data) {
+   m <- mean(data)
+   v <- var(data)
+   return(c(a = m^2/v, s = v/m))
+ }
```

Check Yourself

Tasks

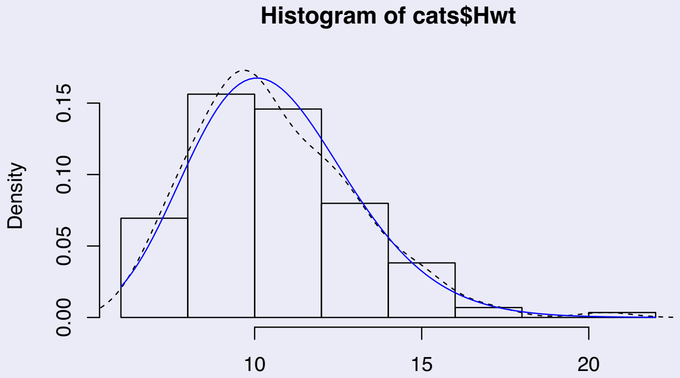
- Write a function `gamma.MMest` that takes as input a data vector and returns estimates of the scale parameters a and s.
- Plug cat heart weights into your function to get estimates of a and s.

Tasks

```
> gamma.MMest(cats$Hwt)

        a           s
19.0653121  0.5575862
```

Method of Moments

```
> hist(cats$Hwt, probability = TRUE, ylim = c(0, 0.17))
> lines(density(cats$Hwt), lty = "dashed")
> cat.MM <- gamma.MMest(cats$Hwt)
> curve(dgamma(x, shape = cat.MM["a"], scale = cat.MM["s"]),
+       add = TRUE, col = "blue")
```



Method of Moments

- Sometimes we can't solve the moment equations for the parameters by hand. In that case, do it numerically.
- Set up a difference function between the data and the model and then minimize this function.

Method of Moments

- Sometimes we can't solve the moment equations for the parameters by hand. In that case, do it numerically.
- Set up a difference function between the data and the model and then minimize this function.

Cats Example

```
> gamma.mean <- function(a, s) {return(a*s)}
> gamma.var  <- function(a, s) {return(a*s^2)}
> gamma.diff <- function(params, data) {
+   a <- params[1]
+   s <- params[2]
+   return((mean(data) - gamma.mean(a,s))^2
+          + (var(data) - gamma.var(a,s))^2)
+ }
```

objective function

Method of Moments

- Sometimes we can't solve the moment equations for the parameters by hand. In that case, do it numerically.
- Set up a difference function between the data and the model and then minimize this function.

Cats Example

```
> nlm(gamma.diff, c(19, 1), data = cats$Hwt)[1:3]
```

nonlinear minimization

```
$minimum
[1] 1.899648e-13

$estimate
[1] 19.0653140  0.5575862

$gradient
[1] -5.338805e-09 -2.119918e-07
```

More generally...

Cats Example

- Nothing special about moments. Could match other data summaries too.
  - Examples: the median, quantiles...
- Try to solve for parameters exactly by hand. If you can't set up a discrepancy function and minimize it.

More generally...

Cats Example

- Nothing special about moments. Could match other data summaries too.
  - Examples: the median, quantiles...
- Try to solve for parameters exactly by hand. If you can't set up a discrepancy function and minimize it.
- Just make sure your summaries converge to the population values.
  - How? Simulate then estimate and estimates should converge as the sample grows.

Check Yourself: Checking Your Estimator

Task

- Simulate 100 random variables from a gamma distribution with shape parameter equal to 19 and scale parameter equal to 45. Run the `gamma.MMest` with these values as the input.
- Do the same thing but simulate $10,000$ random variables. Next, $1,000,000$ random variables.
- Does it seem like our estimates are converging to the truth?

Check Yourself: Checking Your Estimator

Solutions

```
> gamma.MMest(rgamma(100, shape = 19, scale = 45))

       a        s
15.76858 53.27196

> gamma.MMest(rgamma(10000, shape = 19, scale = 45))

       a        s
18.62646 46.03568

> gamma.MMest(rgamma(1000000, shape = 19, scale = 45))

       a        s
18.94818 45.12988
```

Section II

Maximum Likelihood Estimation

Maximum Likelihood

- Usually we think of parameters, θ, as fixed and consider the probability of different outcomes $f(x, \theta)$ with θ constant and x changing.

Maximum Likelihood

- Usually we think of parameters, θ, as fixed and consider the probability of different outcomes $f(x, \theta)$ with θ constant and x changing.
- **Likelihood** of a parameter value is given by $L(\theta | x_1, \ldots, x_n)$: what probability does θ give the data?
  - For continuous variables, use the probability density.
  - Calculate $f(x, \theta)$ letting θ change with data constant.
  - *Not* the probability of θ.

Maximum Likelihood

- Usually we think of parameters, θ, as fixed and consider the probability of different outcomes $f(x, \theta)$ with θ constant and x changing.
- **Likelihood** of a parameter value is given by $L(\theta | x_1, \ldots, x_n)$: what probability does θ give the data?
  - For continuous variables, use the probability density.
  - Calculate $f(x, \theta)$ letting θ change with data constant.
  - *Not* the probability of θ.
- **Maximum likelihood** is the guess that the parameter is whatever makes the data most likely.
- Most likely parameter value is the **maximum likelihood estimate** or the **MLE**.

Coding the Likelihood Function

fn of x $f(x_1,\dots,x_n;\theta)=\prod_{i=1}^{n} f(x_i;\theta)$

- With independent data points $x_1, x_2, \dots, x_n$ the likelihood is

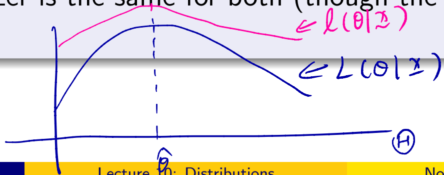
fn of θ

$$L(\theta|x_1,\dots,x_n)=\prod_{i=1}^{n} f(x_i,\theta).$$

- Multiplying lots of small numbering is bad, so we usually take the log:

$$\ell(\theta|x_1,\dots,x_n)=\sum_{i=1}^{n}\log f(x_i,\theta).$$

- Note the maximizer is the same for both (though the maximum value will be different).



Check Yourself

Tasks

- Write a function `gamma.ll` which takes as input a parameter vector (with shape and scale) and a data vector and from that returns the log likelihood assuming the data are independent draws from a gamma distribution with scale and shape indicated by the input parameter vec. HINT: Use `dgamma()`.
- Test your function on the cats heart weight data and parameter values scale equals 19 and shape equals 0.5.

Check Yourself

Solution

```
> gamma.ll <- function(params, data) {
+   a <- params[1]
+   s <- params[2]
+   return(sum(dgamma(data, shape = a,
+                          scale = s, log = TRUE)))
+ }
> gamma.ll(c(19, 0.05), cats$Hwt)
```

```
[1] -21598.19
```

How do we maximize the likelihood?

How do we maximize it?

- Sometimes, like for the normal distribution, we can do this by hand with calculus.
- Other times we need to use numerical methods... *minimize* the negative log likelihood.

How do we maximize the likelihood?

How do we maximize it?

- Sometimes, like for the normal distribution, we can do this by hand with calculus.
- Other times we need to use numerical methods... *minimize* the negative log likelihood.

```
> nlm(gamma.ll, c(19, 1), data = cats$Hwt)[1:3]
$minimum
[1] -1334280770

$estimate
[1] 409228.0 469356.4

$gradient
[1] -3404.4834  -125.5524
```

mini

Not work

How do we maximize the likelihood?

```
> neg.gamma.ll <- function(params, data) {
+   a <- params[1]
+   s <- params[2]
+   return(-sum(dgamma(data, shape = a,
+                            scale = s, log = TRUE)))
+ }
> nlm(neg.gamma.ll, c(19, 1), data = cats$Hwt)$minimum

[1] 325.5476

> nlm(neg.gamma.ll, c(19, 1), data = cats$Hwt)$estimate

[1] 20.299930  0.523674
```

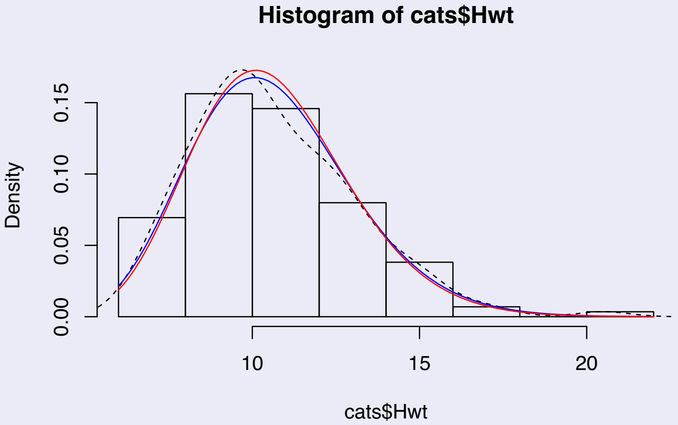
How do we maximize the likelihood?

```
> neg.gamma.ll <- function(params, data) {
+   a <- params[1]
+   s <- params[2]
+   return(-sum(dgamma(data, shape = a,
+                            scale = s, log = TRUE)))
+ }
> nlm(neg.gamma.ll, c(19, 1), data = cats$Hwt)$minimum

[1] 325.5476

> nlm(neg.gamma.ll, c(19, 1), data = cats$Hwt)$estimate

[1] 20.299930  0.523674

> cat.MM <- gamma.MMest(cats$Hwt)
> neg.gamma.ll(cat.MM, cats$Hwt)

[1] 325.6886
```

Maximum Likelihood

```
> hist(cats$Hwt, probability = TRUE, ylim = c(0, 0.17))
> lines(density(cats$Hwt), lty = "dashed")
> cat.MLE <- nlm(neg.gamma.ll, c(19, 1), data = cats$Hwt)$estimate
> curve(dgamma(x, shape = cat.MM["a"], scale = cat.MM["s"]),
+       add = TRUE, col = "blue")
> curve(dgamma(x, shape = cat.MLE[1], scale = cat.MLE[2]),
+       add = TRUE, col = "red")
```

Maximum Likelihood



Why the MLE?

- Usually *consistent*: converges to the truth as we get more data.
- Usually *efficient*: converges to the truth as least as fast as anything else.

Checking Fit

- Plot the data with your estimates (like in the last slide).
- Calculate summary statistics not used in fitting and compare with those of the fitted model.
    - Some plotting tools to help with this.
- Use statistical tests. KS

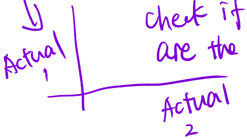
Checking Fit: Summary Statistics

```
> # Model quantiles
> qgamma(c(0.01, 0.05, 0.95, 0.99), shape = cat.MM["a"],
+        scale = cat.MM["s"])
[1]  5.795333  6.966974 14.926292 17.097730 → true quantile
> # Data quantiles:
> quantile(cats$Hwt, c(0.01, 0.05, 0.95, 0.99))
    1%      5%     95%     99%
 6.500   7.300 14.885 17.028
```

Checking Fit: Summary Statistics

Quantile-Quantile (Q-Q) Plots

- Plots theoretical vs. actual quantiles. → no matter which x which y
- Ideally, a straight line when the distributions are the same.
- `qqnorm()` and `qqline()` are specialized for checking normality.
- Could also plot quantiles of two samples against each other.
- `qqplot(x,y)` gives a Q-Q plot of one vector against another.



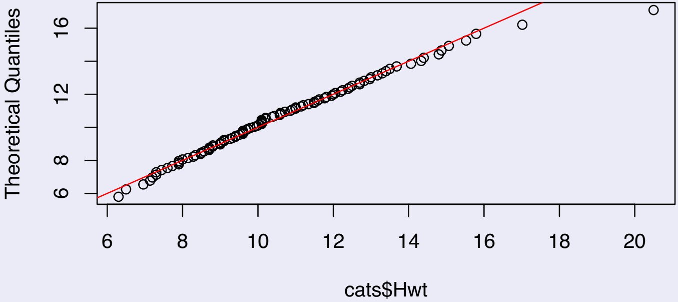
check if 2 dist are the same

Checking Fit: Summary Statistics

Quantile-quantile Plot

plug in the estimates

```
> a <- cat.MM["a"]; s <- cat.MM["s"]
> qqplot(cats$Hwt, qgamma((1:99)/100, shape = a, scale = s),
+        ylab = "Theoretical Quantiles")
> abline(0, 1, col = "red")
```



Checking Fit: Summary Statistics

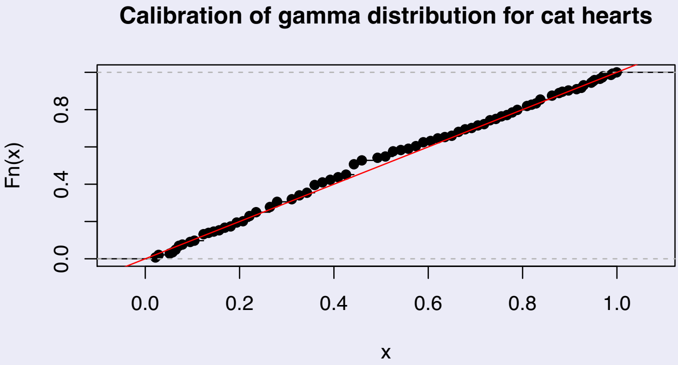
Calibration Plots

- If the distribution is right, 50% of data should be below the median, 90% should be below the 90th percentile, etc.
- **Calibration** probabilities: events with probability p% should happen about p% of the time, not more or less.
- Can look at calibration by calculating the (empirical) CDF and the (theoretical) CDF and plotting.
  - Ideal calibration is a straight line up the diagonal.
  - Systematic deviations should be a warning sign.

Checking Fit: Summary Statistics

Calibration Plots

```
> plot(ecdf(pgamma(cats$Hwt, shape = a, scale = s)),
+        main = "Calibration of gamma distribution for cat hearts")
> abline(0, 1, col = "red")
```



Checking Fit: Kolmogorov-Smirnoff Test

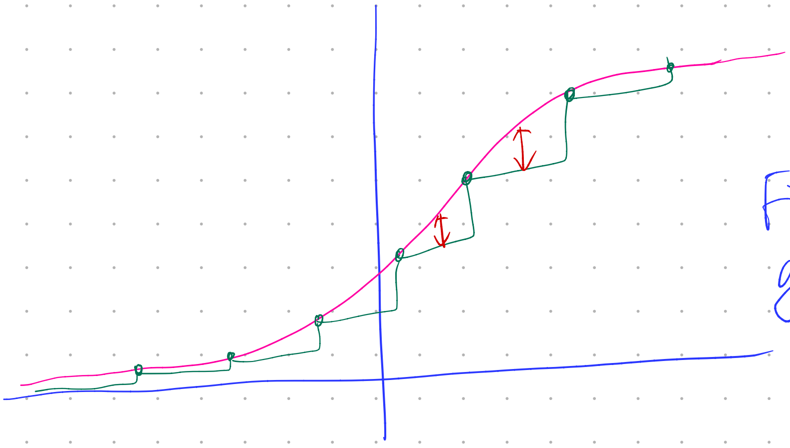
- How much should Q-Q or calibration plot wiggle around the diagonal?

Checking Fit: Kolmogorov-Smirnoff Test

- How much should Q-Q or calibration plot wiggle around the diagonal?
- Answer a different question: define the biggest gap between theoretical and empirical CDF

$$D_{KS} = \max_x \left| F(x) - \hat{F}(x) \right|$$

- D_{KS} always has the same distribution *if* the theoretical CDF is fixed and correct.
- Also works for comparing empirical CDF of two samples to see if they come from the same distribution. $\left| \hat{F}_1(x) - \hat{F}_2(x) \right|$



Find the max gap = $\max_{x} |F(x) - \hat{F}(x)|$

Checking Fit: Kolmogorov-Smirnoff Test

H_0 = $X \sim$ Normal $X \sim$ Gamma

H_1: X not Normal X not Gamma

```
> ks.test(cats$Hwt, pgamma, shape = a, scale = s)

        One-sample Kolmogorov-Smirnov test

data:  cats$Hwt
D = 0.068637, p-value = 0.5062
alternative hypothesis: two-sided
```

Fail to regect H_0

Checking Fit: Kolmogorov-Smirnoff Test

Warning

- More complicated and not properly handled by built-in `R` if parameters are estimated by the data.
  - Fit looks better than it really is.
- Hack: estimate the model using 90% of the data and check the fit using the K-S test using the other 10% (feels a little bit like *cross*-validation).

Checking Fit: Kolmogorov-Smirnoff Test

```
> n      <- length(cats$Hwt)
> train  <- sample(1:n, size = round(.9*n))
> cat.MM <- gamma.MMest(cats$Hwt[train])
> a <- cat.MM["a"]
> s <- cat.MM["s"]
> a

       a
18.34851

> s

        s
0.5773683
```

90% of n

Checking Fit: Kolmogorov-Smirnoff Test

everything except train

```
> ks.test(cats$Hwt[-train], pgamma, shape = a, scale = s)

        One-sample Kolmogorov-Smirnov test

data:  cats$Hwt[-train]
D = 0.2031, p-value = 0.6105
alternative hypothesis: two-sided
```

Checking Fit: Kolmogorov-Smirnoff Test

Can also test whether two samples come from the same distribution.

```
> ks.test(cats$Hwt[cats$Sex == "F"],
+         cats$Hwt[cats$Sex == "M"])

        Two-sample Kolmogorov-Smirnov test

data:  cats$Hwt[cats$Sex == "F"] and cats$Hwt[cats$Sex == "M"]
D = 0.49419, p-value = 3.847e-07
alternative hypothesis: two-sided
```

→ Reject H_0

H_0: Male heart rate = female HR

Section III

Bayesian Models and Estimation

Bayesian Estimation

Frequentist Statistics

- In frequentist statistics, we seek to estimate an unknown parameter θ.
- θ is a fixed (non-random) unknown number.
- θ is not a random variable.
- An estimator $\hat{\theta}$ is random variable.

Bayesian Statistics

- In contrast to the frequentist paradigm, Bayesian statistics assumes that θ is a random variable.
- The primary goal of Bayesian methods is to estimate the posterior distribution $\pi(\theta|x_1, \ldots, x_n)$.
- The posterior distribution $\pi(\theta|x_1, \ldots, x_n)$ is the probability distribution of θ conditional on the observed data $x_1, \ldots, x_n$.

Bayesian Estimation

Some notation

- $\pi(\theta)$ is the **prior** distribution of θ.
- $f(x_1, \ldots, x_n|\theta)$ is the distribution of the sample. This is the *frequentist* distribution of sample $x_1 \ldots, x_n$. (Also the likelihood)
- $\pi(\theta|x_1, \ldots, x_n)$ is the **posterior** distribution of θ given the observed data $x_1, \ldots, x_n$.

Choose a prior

- You need to choose a prior $\pi(\theta)$ before the data are collected.
- This does add subjectivity to our model.

Use Bayes' rule to compute the posterior!

$$\pi(\theta|x_1, \ldots, x_n) = \frac{f(\theta, x_1, \ldots, x_n)}{m(x_1, \ldots, x_n)} = \frac{f(x_1, \ldots, x_n|\theta)\pi(\theta)}{m(x_1, \ldots, x_n)}$$

Bayesian Estimation

Bayes' estimator

The Bayes' estimator $\hat{\theta}_B$ of θ is the conditional expectation of $\theta|x_1, \ldots, x_n$.

$$\hat{\theta}_B = E[\theta|x_1, \ldots, x_n]$$

Analytic expression

$$\hat{\theta}_B = \int \theta \times \pi(\theta|x_1, \ldots, x_n) d\theta$$

Kernel trick

- When computing $\pi(\theta|x_1, \ldots, x_n)$, we can often only look at the kernel of $f(x_1, \ldots, x_n|\theta)\pi(\theta)/m(x_1, \ldots, x_n)$. This allows us to not calculate the marginal distribution $m(x_1, \ldots, x_n)$.
- The kernel of a pdf (or pmf) $f(x)$ is its functional form without the normalizing constant. E.g., the kernel of a standard normal is $e^{-x^2/2}$.

Bayesian Estimation: Bernoulli with Beta Prior

Classic Example: Problem Statement

Let $X_1, \ldots, X_n \overset{iid}{\sim} Bernoulli(p)$. Note that $Y = \sum X_i \sim Binomial(n, p)$ and n is known. The goal is to identify the posterior distribution $\pi(p|x_1, \ldots, x_n)$ and Bayes' estimator $\hat{p}_B$.

Chose prior

- Choose $p \sim Beta(\alpha, \beta)$
- Note that $0 < p < 1$ so a beta distribution is a natural choice.

Expressions

- Binomial: $f(y|p) = \binom{n}{y} p^y (1-p)^{n-y}$
- Beta prior: $\pi(p) = \frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)\Gamma(\beta)} p^{\alpha-1}(1-p)^{\beta-1}$

Bayesian Estimation: Bernoulli with Beta Prior

Classic Example: Problem Statement

Let $X_1, \ldots, X_n \overset{iid}{\sim} Bernoulli(p)$, or $Y = \sum X_i \sim Binomial(n, p)$.

Start solution

$$\begin{aligned}\pi(p|x_1, \ldots, x_n) &= \frac{f(x_1, \ldots, x_n|p)\pi(p)}{m(x_1, \ldots, x_n)} = \frac{f(x_1, \ldots, x_n|\theta)\pi(p)}{m} \\ &= \binom{n}{y} p^y (1-p)^{n-y} \frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)\Gamma(\beta)} p^{\alpha-1}(1-p)^{\beta-1} \frac{1}{m}\end{aligned}$$

Kernel trick

Thus

$$\pi(p|x_1, \ldots, x_n) = a \times p^{y+\alpha-1}(1-p)^{n-y+\beta-1}$$

Where a is the normalizing constant.

Bayesian Estimation: Bernoulli with Beta Prior

Classic Example: Problem Statement

Let $X_1, \ldots, X_n \overset{iid}{\sim} Bernoulli(p)$, or $Y = \sum X_i \sim Binomial(n, p)$.

Kernel trick

- The *kernel trick* is to identify that $\pi(p|x_1, \ldots, x_n)$ is proportional to the kernel $p^{y+\alpha-1}(1-p)^{n-y+\beta-1}$.
- The pdf with kernel $p^{y+\alpha-1}(1-p)^{n-y+\beta-1}$ is a Beta distribution.

Posterior and Bayes estimator

The posterior distribution of p given the observed data $y = \sum_{i=1}^{n} x_i$ is

$$p|y \sim Beta(y+\alpha, n-y+\beta).$$

Bayesian Estimation: Bernoulli with Beta Prior

Classic Example: Problem Statement

Let $X_1, \ldots, X_n \overset{iid}{\sim} \mathit{Bernoulli}(p)$, or $Y = \sum X_i \sim \mathit{Binomial}(n, p)$.

Bayes estimator

The Bayes estimator of p is

$$\begin{aligned}\hat{p}_B &= \frac{Y + \alpha}{(Y + \alpha) + (n - Y + \beta)} \\ &= \frac{Y + \alpha}{\alpha + \beta + n} \\ &= \frac{\sum X_i + \alpha}{\alpha + \beta + n}\end{aligned}$$

Bayesian Estimation: Bernoulli with Beta Prior

Classic Example: Problem Statement

Let $X_1, \ldots, X_n \overset{iid}{\sim} Bernoulli(p)$, or $Y = \sum X_i \sim Binomial(n, p)$.

Conjugate prior

- The prior is a Beta and the posterior is also a Beta.
- Here we say: *The Beta family is conjugate for the Binomial family.*

Weighted average

- The Bayes' estimator of p can be expressed as a weighted average of $\bar{X}$ and $E[p]$

$$\hat{p}_B = \left(\frac{n}{\alpha+\beta+n}\right)\bar{X} + \left(\frac{\alpha+\beta}{\alpha+\beta+n}\right)\left(\frac{\alpha}{\alpha+\beta}\right)$$

Markov Chain Monte Carlo (MCMC)

Goal: Estimate posterior

Markov Chain Monte Carlo

The Markov Chain

- A discrete-time Markov chain is a sequence of random variables $X_1, X_2, X_3, \ldots$ with the Markov property, namely that the probability of moving to the next state depends only on the present state and not on the previous states (Wiki)

$$P(X_{t+1} = x | X_1 = x_1, X_2 = x_2, \ldots, X_t = x_t) = P(X_{t+1} = x | X_t = x_t)$$

- The Markov chain X_t (or $X^{(t)}$) is a discrete-time stochastic process.

Convergence of the chain

The distribution of X_t (or $X^{(t)}$) converges to limiting stationary distribution of the chain when the chain is irreducible and aperiodic (statement taken from Givens & Hoeting).

Markov Chain Monte Carlo

- **Irreducible:** possible to get to any state from any state.
- **Aperiodic:** distribution of $X^{(t)}$ after T steps is close to the distribution of $X^{(t)}$ after $T+1$ steps. Dist. doesn't change

Study on your own!

A more complete and formal introduction is required to understand the intricate properties of MCMC. Study Markov chains on your own if you are interested.

MCMC Motivation

- The MCMC setup is similar to the accept-reject algorithm.
- The goal is to simulate from target distribution $f(x)$.
- Suppose we can easily simulate from proposal distribution $g(\cdot|x^{(t)})$.
- MCMC algorithms construct an irreducible and aperiodic Markov chain $X^{(t)}$ which converges to limiting stationary distribution $f(x)$.

Metropolis-Hastings algorithm

The Metropolis-Hastings algorithm is a general method of simulating a Markov chain.

Metropolis-Hastings algorithm We have the Initial State

i. Sample a candidate X^* from proposal $g(\cdot|x^{(t)})$. $X^* \sim g$

ii. Compute Metropolis-Hastings ratio $R(x^{(t)}, X^*)$, where

$$R(u, v) = \frac{f(v)g(u|v)}{f(u)g(v|u)} = \text{a number}$$

When $R(u,v)$ is big, $X^{(t+1)} = X^*$

Small, $X^{(t+1)} = X^t$

iii. Sample a value for $X^{(t+1)}$ according to

$$X^{(t+1)} = \begin{cases} X^* & \text{with probability } \min\{R(x^{(t)}, X^*), 1\}, \\ x^{(t)} & \text{otherwise} \end{cases}$$

iv. Increment t and return to step i.

Independence chains

Independence chains

- Suppose the proposal distribution from the Metropolis-Hastings algorithm has been chosen such that

$$g(x^*|x^{(t)}) = g(x^*),$$

for some density g.

- This yields an **independence chain**, where each candidate is drawn independently of the past.
- The Metropolis-Hastings ratio is

$$R(x^{(t)}, X^*) = \frac{f(X^*)g(x^{(t)})}{f(x^{(t)})g(X^*)}$$

Bayesian Application

MCMC in Bayesian

- In a Bayesian setting, we seek to identify or compute the posterior distribution $\pi(\theta|x_1, \ldots, x_n)$.
- In our famous Beta prior Bernoulli example, we were able to identify the closed form posterior. Very cool!
- This closed form solution often does not exist.
- The posterior distribution $\pi(\theta|x_1, \ldots, x_n)$ does exist even if a nice closed form solution does not exist.
- Thus we must use numerical methods to estimate $\pi(\theta|x_1, \ldots, x_n)$.
- MCMC applications in Bayesian statistics are some of the most widely used methods in statistics, data science, machine learning, and many more fields.

Bayesian Application

MCMC in Bayesian setup

- In this setting, the Markov chain is denoted by $\theta^{(t)}$. Our target distribution is the posterior $\pi(\theta|x_1,\ldots,x_n)$.
- Recall that the data generating process is given by $f(x_1,\ldots,x_n|\theta)$. This is also the likelihood function $L(\theta|x_1,\ldots,x_n)$.
- Assume the proposal distribution g is chosen to be the prior distribution $\pi(\theta)$.
- The above choice of proposal distribution g produces an independence chain.
- Recall the posterior is calculated by

$$\pi(\theta|x_1,\ldots,x_n)=\frac{f(x_1,\ldots,x_n|\theta)\pi(\theta)}{m(x_1,\ldots,x_n)}=\frac{L(\theta|x_1,\ldots,x_n)\pi(\theta)}{m}$$

Bayesian Application

MCMC in Bayesian setup continued

- Thus the Metropolis-Hastings ratio $R(\theta^{(t)}, \theta^*)$ is

$$\frac{\pi(\theta^*|x_1,\ldots,x_n)\pi(\theta^{(t)})}{\pi(\theta^{(t)}|x_1,\ldots,x_n)\pi(\theta^*)} = \frac{\frac{L(\theta^*|x_1,\ldots,x_n)\cancel{\pi(\theta^*)}}{\cancel{m}}\cancel{\pi(\theta^{(t)})}}{\frac{L(\theta^{(t)}|x_1,\ldots,x_n)\cancel{\pi(\theta^{(t)})}}{\cancel{m}}\cancel{\pi(\theta^*)}}$$

- After simplifying, the Metropolis-Hastings ratio is

$$R(\theta^{(t)}, \theta^*) = \frac{L(\theta^*|x_1,\ldots,x_n)}{L(\theta^{(t)}|x_1,\ldots,x_n)}$$

Don't do log likelihood

use the raw likelihood !

- In this Bayesian application of Markov Chain Monte Carlo, the Metropolis-Hastings ratio is computed using a likelihood ratio.

Bayesian Application

Independence Chain Bayesian Metropolis-Hastings algorithm

Take the initial draw $\theta^{(0)}$ from $\pi(\theta)$.

i. Sample a candidate θ^* from proposal $\pi(\theta)$.

ii. Compute Metropolis-Hastings ratio $R(\theta^{(t)}, \theta^*)$, where

$$R(\theta^{(t)}, \theta^*) = \frac{L(\theta^*|x_1, \ldots, x_n)}{L(\theta^{(t)}|x_1, \ldots, x_n)}$$

iii. Sample a value for $\theta^{(t+1)}$ according to

$$\theta^{(t+1)} = \begin{cases} \theta^* & \text{with probability } \min\{R(\theta^{(t)}, \theta^*), 1\}, \\ \theta^{(t)} & \text{otherwise} \end{cases}$$

iv. Increment t and return to step i.

Bayesian Estimation: Bernoulli with Beta Prior

Note

- A closed form solution exists for this exercise. Numerical techniques are not necessary. This is solved via MCMC for illustration.

Classic example

- Let $X_1, \ldots, X_n \overset{iid}{\sim} Bernoulli(p)$, or $Y = \sum X_i \sim Binomial(n, p)$
- The likelihood function in terms of $x_1, \ldots, x_n$:

$$L(p|x_1, \ldots, x_n) = \prod_{i=1}^{n} p^{x_i}(1-p)^{n-x_i}$$

- The likelihood function in terms of $y = \sum x_i$:

$$L(p|y) = \binom{n}{y} p^y (1-p)^{n-y}$$

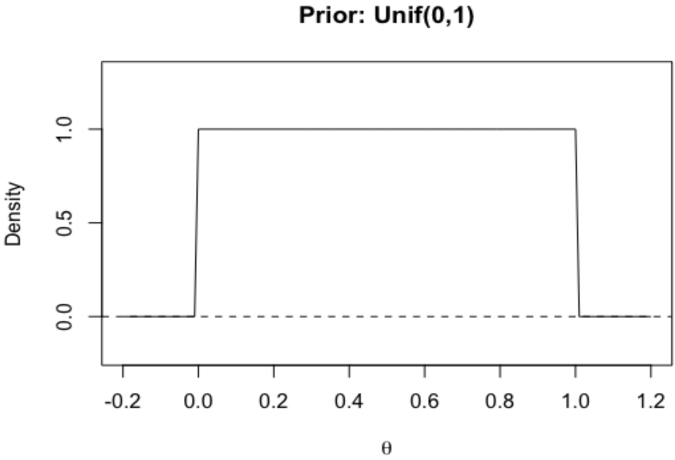
Bayesian Estimation: Bernoulli with Beta Prior

Simulate a dataset

True success probability is $p = .25$. The known sample size is $n = 50$.

```
> set.seed(1983)
> p <- .25
> n <- 50
> X <- rbinom(n, size = 1, prob = p)
> X
 [1] 0 0 0 0 1 0 0 0 0 0 1 1 0 1 0 0 0 0 0 0 0 0 0 0 1 0 0 0
[29] 0 0 0 1 0 0 0 0 0 0 0 0 0 1 0 1 0 0 0 0 0 1
> mean(X)
[1] 0.18
```

Prior 1: Unif(0,1), Note: $\theta = p$



Prior 1: Code Lab

Beta(1,1) = Unif(0,1)

MCMC

$\theta^{(0)}$

```
> theta_1 <- rbeta(1,1,1) # Draw theta_(0)
> n.samps <- 10000 # Number of iterations
> theta_vec <- rep(NA,(n.samps+1))
> theta_vec[1] <- theta_1
> # MCMC loop
> for (t in 1:n.samps) {
+   theta_star <- rbeta(1,1,1) # Draw theta* from from proposa
+   theta_t <- theta_vec[t] # theta_(t)
+   # Compute MH ratio
+   MH_ratio <- prod(dbinom(X,size=1,prob=theta_star))/prod(dbi
+   # Select new case
+   prob_vec <- c(min(MH_ratio,1),1-min(MH_ratio,1))
+   theta_vec[t+1] <- sample(c(theta_star,theta_t),1,prob = pro
+ }
```

θ^*

prob=theta_t

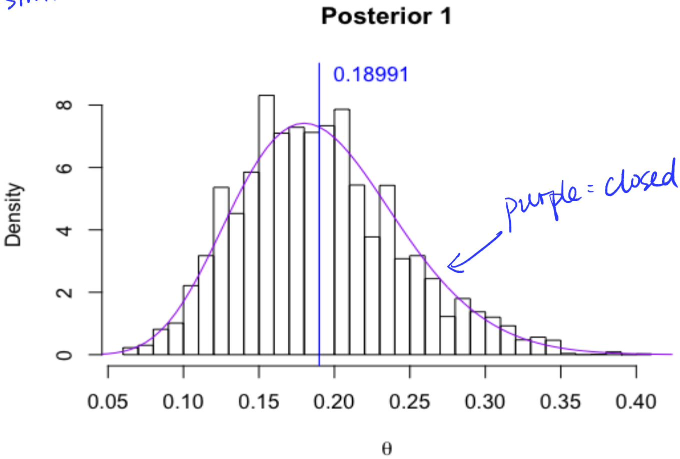
$\prod_{i=1}^{n} P(x_i|p)$

prob_vec

Posterior 1, Note: $\theta = p$

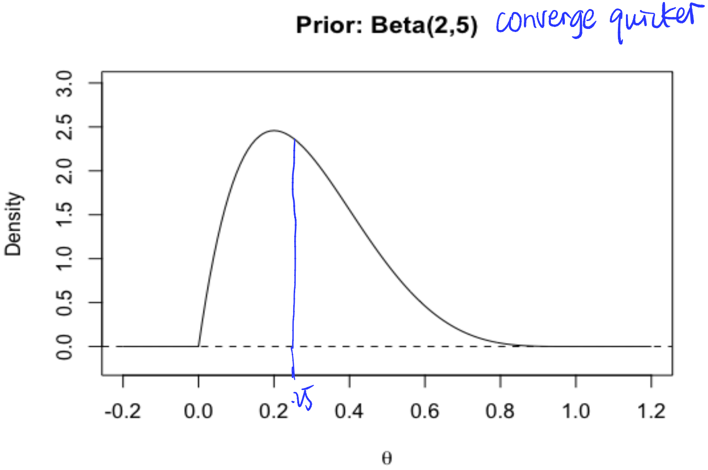
The purple curve is closed form posterior Beta$(y+1, n-y+1)$

hist = simulated dist.



purple = closed form

Prior 2: Beta(2,5), Note: $\theta = p$



converge quicker

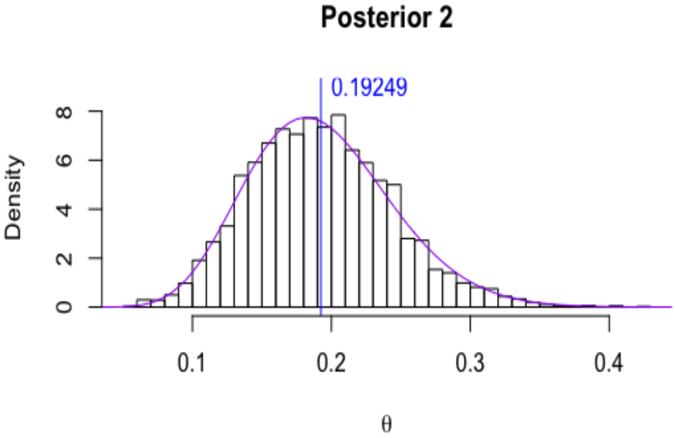
Prior 2: Code

MCMC

```
> theta_1 <- rbeta(1,2,5) # Draw theta_(0)
> n.samps <- 10000 # Number of iterations
> theta_vec <- rep(NA,(n.samps+1))
> theta_vec[1] <- theta_1
> # MCMC loop
> for (t in 1:n.samps) {
+  theta_star <- rbeta(1,2,5) # Draw theta* from from proposal
+  theta_t <- theta_vec[t] # theta_(t)
+  # Compute MH ratio
+  MH_ratio <- prod(dbinom(X,size=1,prob=theta_star))/prod(dbi
+  # Select new case
+  prob_vec <- c(min(MH_ratio,1),1-min(MH_ratio,1))
+  theta_vec[t+1] <- sample(c(theta_star,theta_t),1,prob = pro
+  }
```

Posterior 2, Note: $\theta = p$

The purple curve is closed form posterior Beta$(y + 2, n - y + 5)$



Bayesian Estimation MCMC: Mixing

Good mixing and bad mixing

- A simulated Markov chain exhibits **good mixing** if $\theta^{(t)}$ moves quickly away from its starting value and is able to sample values from all portions of the parameter space governed by the posterior.
- A simulated Markov chain exhibits **poor mixing** when $\theta^{(t)}$ fails to sample all portions of the parameter space governed by the posterior.

Burn-in or thinning your chain

- Markov chains are dependent on the initial draw $\theta^{(0)}$. This is drawn from the proposal (or prior).
- The chain can take several iterations to move away from its starting value.
- It is common to discard the initial portion of the simulated chain. This is referred to as the **burn-in** period.

Bayesian Estimation MCMC: Diagnostics

Trace plots

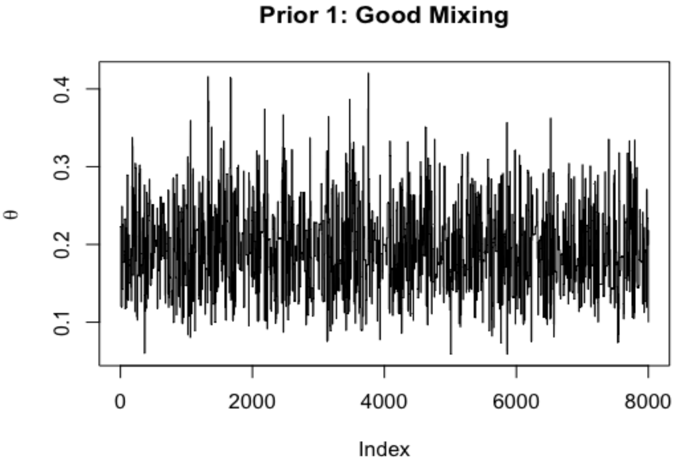
- A **Trace plot** is a lineplot of the simulated chain $\theta^{(t)}$ as a function of its iterations.
- Mixing properties of the chain can be easily noticed with a trace plot.

Auto correlation

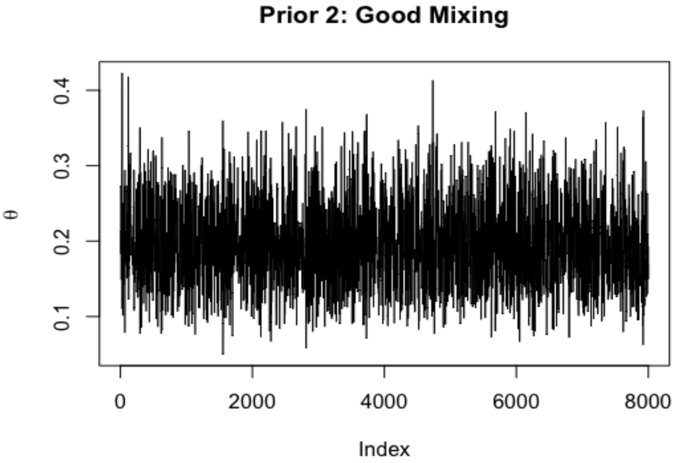
ACF plot

- Plotting the empirical autocorrelation function is another common diagnostic tool.
- This is the **acf()** function in R.
- A **quick decay** of the chain's autocorrelations indicate good mixing properties.
- A **slow decay** of the chain's autocorrelations indicate poor mixing properties.

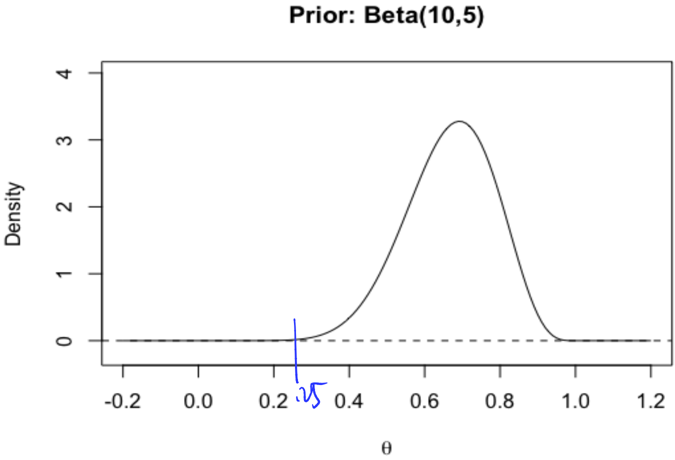
Prior 1: Unif(0,1) Good Mixing



Prior 2: Beta(2,5) Good Mixing

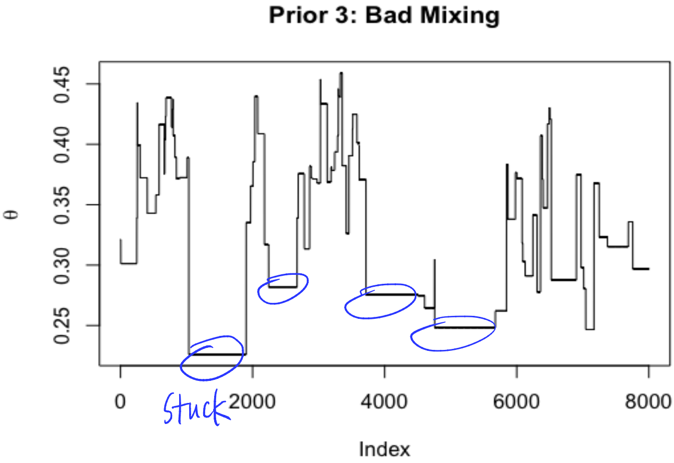


Prior 3: Beta(10,5), Note: $\theta = p$



Prior 3: Beta(10,5) Poor Mixing

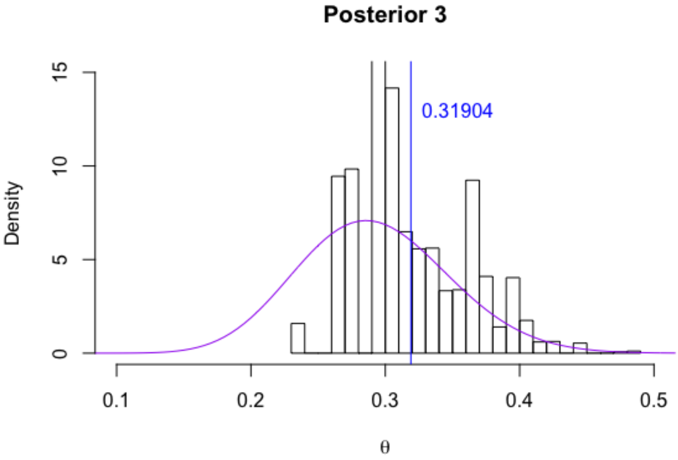
Still get to the convergence, but need more iteration



Stuck

Posterior 3, Note: $\theta = p$, 10,000 Iterations

The purple curve is closed form posterior Beta$(y + 10, n - y + 5)$



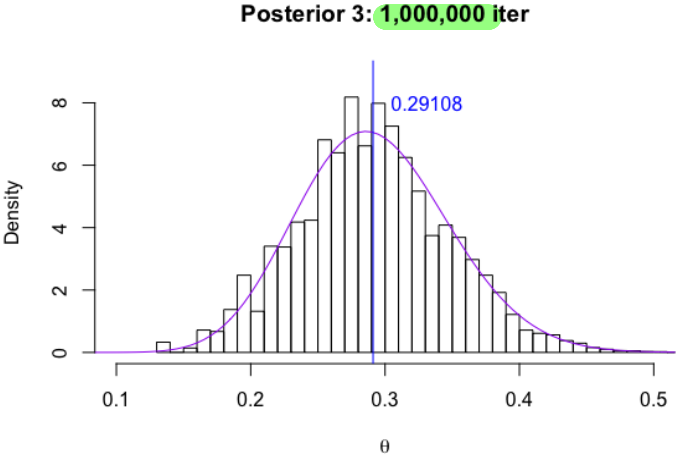
Bayesian Estimation MCMC

The chain almost always converges!

- Even if a chain exhibits poor mixing properties, **the simulated Markov chain will often converge to the target posterior.**
- Thus!! You can always take more and more iterations to estimate the posterior distribution.
- This is one of many reasons why MCMC is such a powerful method!
- However, it is possible to choose a proposal distribution that does not produce an irreducible and aperiodic chain. Hence does not converge.
- The topic of *MCMC convergence* deserves more attention.

Posterior 3: With 1,000,000 Iterations

The purple curve is closed form posterior Beta$(y + 10, n - y + 5)$



Bayesian Estimation MCMC

Some final thoughts

- MCMC is an important and widely used tool in both academia and industry.
- Take courses in Bayesian statistics, computational statistics and machine learning to learn more.

Gibbs sampler

- The **Gibbs sampler** is adapted for multidimensional target distributions.
- This MCMC algorithm is also widely used.

Optional Reading

- Chapter 7 (Markov Chain Monte Carlo) in Computational Statistics.