Introduction to Bayesian Statistics with R

Day 3

Oliver Lindemann

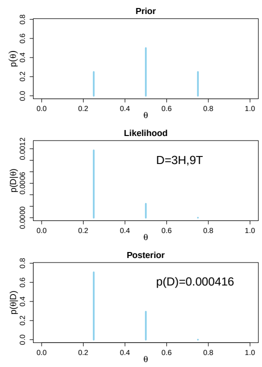
Erasmus University Rotterdam, NL

Bayes rule

$$\underbrace{p(\theta|D)}_{\text{Posterior}} = \underbrace{p(D|\theta)}_{\text{Likelihood}}\,\underbrace{p(\theta)}_{\text{Prior}} \;/\; \underbrace{p(D)}_{\text{Evidence}}$$

Discrete Likelihood $p(D|\theta)$

Example from previous session: Discrete values



How to determine $p(D|\theta)$?

How to determine the posterior distribution of continuous parameters?

1. Analytically
2. Approximation (Grid approximation)
3. Simulation (Markov Chain Monte Carlo)

Likelihood $p(D|\theta)$

Coin flip example: $D = \{y_1, \dots, y_N\}$

- N: number of flips
- z: Number of 'heads', $z = \sum_i^N y_i$

Bernoulli Distribution

$$p(y_i|\theta) = \theta^y\,(1-\theta)^{(1-y)}$$

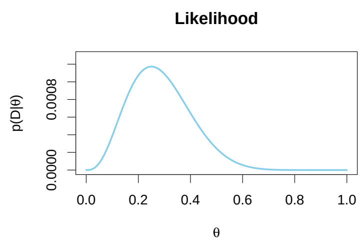
Multiple coin flips

$$p(D|\theta) = p(\{y_1, \dots, y_N\}|\theta) = \prod_i \theta^{y_i}\,(1-\theta)^{(1-y_i)}$$

$$p(D|\theta) = p(z, N|\theta) = \theta^z\,(1-\theta)^{N-z}$$

Continuous likelihood function: $p(D|\theta)$

$$\begin{aligned} p(D|\theta) = p(z = 3, N = 12|\theta) &= \theta^z\,(1-\theta)^{N-z} \\ &= \theta^3\,(1-\theta)^9 \end{aligned}$$



Continuous likelihood function: $p(D|\theta)$


```r
Theta = seq(0,1, by=0.01)
z = 3
N = 12
likelihood = Theta^z * (1-Theta)^(N-z) # Bernoulli likelihood

plot( Theta, likelihood, type="l", lwd=3,
      main="Likelihood",
      xlim=c(0,1),
      xlab=bquote(theta),
      ylim=c(0,1.1*max(likelihood)),
      ylab=bquote(paste("p(D|",theta,")")),
      col="skyblue" )
```


Continuous description of beliefs: Prior $p(\theta)$

Problem

We need a mathematical formula that describes the prior beliefs.

Formula of $p(\theta)$ should be have certain characteristics

1. Bayes' nominator:
   $p(D|\theta) \cdot p(\theta)$ should have the same form as $p(\theta)$
2. Bayes' denominator:
   $p(D) = \int d\theta\, p(D|\theta)p(\theta)$ should be solvable analytically

Continuous description of beliefs: Prior $p(\theta)$

Think about it:

- Form of likelihood, $p(D|\theta)$ — $\theta^z(1-\theta)^{(N-z)}$
- If the prior, $p(\theta)$, has the form — $\theta^a(1-\theta)^b$
- Likelihood $\times$ prior looks then like this — $\theta^{(z+a)}(1-\theta)^{(N-z+b)}$

Continuous prior: Beta Distribution

Probability function of the form is called

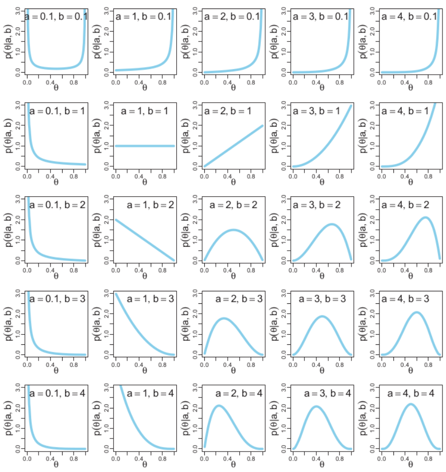
Beta Distribution

$$p(\theta|a,b) = \text{beta}(\theta|a,b)$$
$$= \theta^{(a-1)}(1-\theta)^{(b-1)}/B(a,b) \quad (3.1)$$

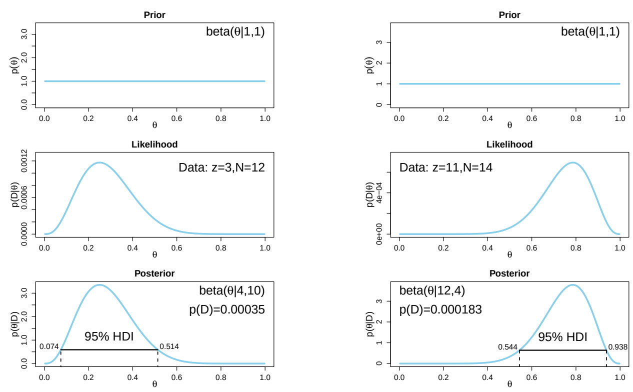
$B(a,b)$ is a normalizing constants $B(a,b) = \int d\theta\, \theta^{(a-1)}(1-\theta)^{(b-1)}$

In R

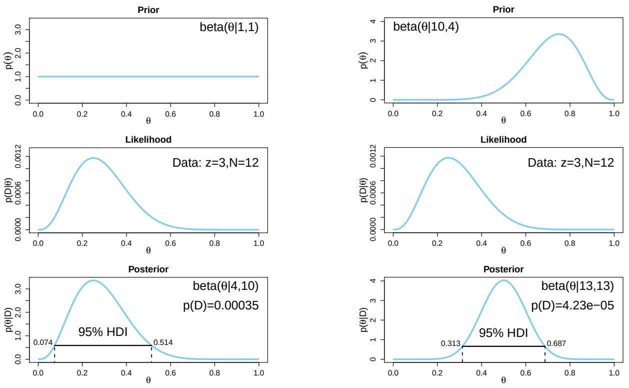
```r
# beta distribution
dbeta(theta, a, b)
# normalizing constant
beta(a,b)
```

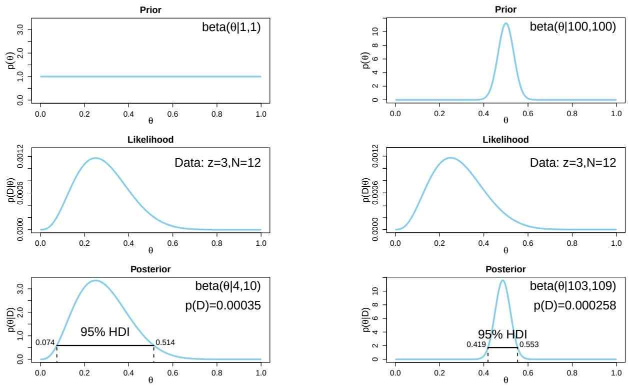
Updating Continuous Prior & Highest Density Interval (HDI)



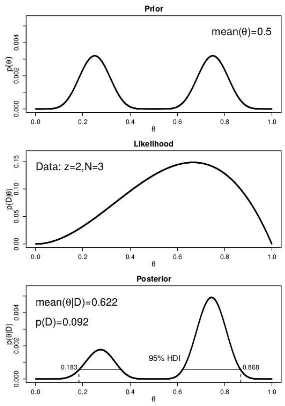
Impact of Priors



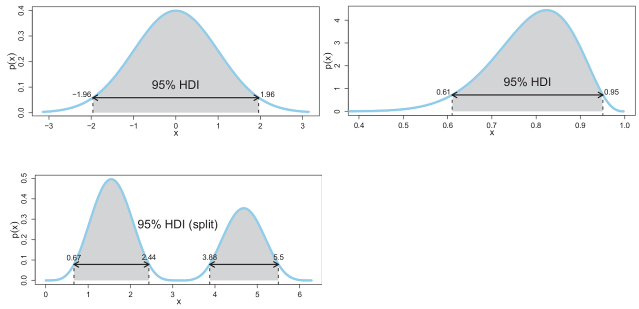
Impact of Priors



Impact of Priors: Arbitrarily shaped Distributions



Examples of 95% highest density intervals (HDIs)



Calculating Posterior

Data	D	$= z, N$
Likelihood	$p(\theta\|z,N)$	$= \theta^z\,(1-\theta)^{(N-z)}$
Prior	$p(\theta)$	$= \text{Beta}(\theta\|a,b)$ $= \theta^{(a-1)}\,(1-\theta)^{(b-1)}/B(a,b)$
Evidence	$p(D)$	$p(z,N)$

Posterior (Bayes rule)

$$p(\theta|z,N) = \frac{p(z,N|\theta)\,p(\theta)}{p(z,N)}$$

Calculating Posterior

$$p(\theta|z,N) = \frac{p(z,N|\theta)\,p(\theta)}{p(z,N)}$$

$$\begin{aligned} p(\theta|z,N) &\propto p(z,N|\theta)\,p(\theta) \\ &\propto \theta^z\,(1-\theta)^{(N-z)}\;\theta^{(a-1)}\,(1-\theta)^{(b-1)} \\ &\propto \theta^{(z+a-1)}\,(1-\theta)^{(N-z+b-1)} \end{aligned}$$

$$p(\theta|z,N) = \frac{\theta^{(z+a-1)}\,(1-\theta)^{(N-z+b-1)}}{B(z+a,N-z+b)}$$

$$p(\theta|a,b) = \theta^{(a-1)}(1-\theta)^{(b-1)}/B(a,b)$$

Finding appropriate parameter (a,b) for my Beta prior distribution?

Approximating Posterior Distributions

Bayes theorem for discrete & continuous values

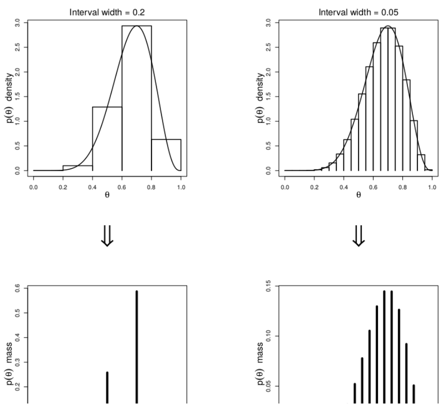
Bayes Rule (continuous values)

$$p(\theta|D) = \frac{p(D|\theta) \cdot p(\theta)}{\int d\theta \; p(D|\theta) \; p(\theta)} \tag{2.6}$$

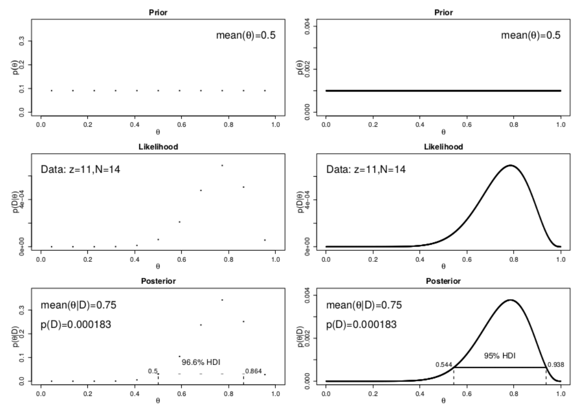
Bayes Rule (discrete values)

$$p(\theta|D) = \frac{p(D|\theta) \cdot p(\theta)}{\sum_\theta p(D|\theta) \; p(\theta)} \tag{2.5}$$

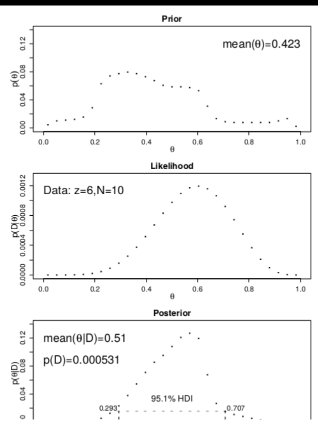
Grip Approximation



Discretizing



Advantage: Modelling Arbitrarily shaped Distributions



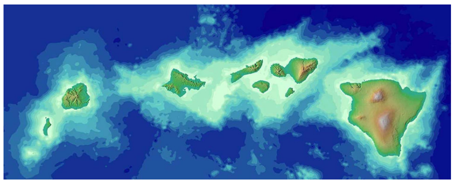
Problem with grip approximation

- not very handy method
- not appropriate for models with large parameter space
  - e.g. 6 parameter
  - each modeled 1000 values (grid density)
  - $1,000^6 = 1,000,000,000,000,000,000$

Markov Chain Monte Carlo (MCMC) simulation

Markov Chain Monte Carlo simulation

Metaphor



- Politician lives on a long chain of islands
- Goal: Visit each island in proportion to its population
- Problem:
  - Unknown number of islands?
  - Unknown total population?
- But: Major of the current and the next island can be asked

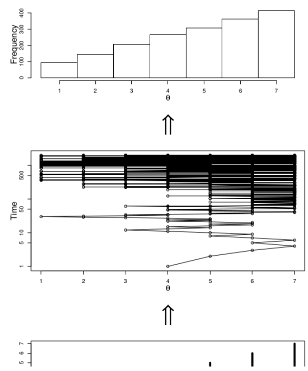
MCMC: Metropolis Algorithm

Random walk:

1. choose randomly step to the left or right (*proposal distribution*)
2. if $p(\text{proposed}) > p(\text{current})$
   - move to proposed
3. else
   - calculate ratio $p_{\text{move}} = \frac{p(\text{proposed})}{p(\text{current})}$
   - move in with likelihood p_{move} to next place

MCMC: Metropolis Algorithm

(Demonstration)



MCMC: Metropolis Algorithm

Random walk:

1. choose randomly step to the left or right (*proposal distribution*)
2. if $p(\text{proposed}) > p(\text{current})$
   - move to proposed
3. else
   - calculate ratio $p_{\text{move}} = \frac{p(\text{proposed})}{p(\text{current})}$
   - move in with likelihood p_{move} to next place

Target Distribution

(step 2 & 3)

$$p_{\text{move}} = \min\left(\frac{p(\text{proposed})}{p(\text{current})}, 1\right)$$

MCMC: Metropolis Algorithm

Markov Chain Monte Carlo (MCMC)

We must be able to

1. create a random value from the **proposal distribution**
2. compute ratio to evaluate the **target distribution** at any position: $p_{\text{move}} = p(\text{proposed}) / p(\text{current})$
3. generate random value from uniform distribution (to move or not move according p_{move})

Note: Target distribution does not have to be normalized, because merely need to know the ratio between two positions (e.g. slope).

MCMC: Continuous data

Metropolis algorithm is applied to

- continuous data
- any number of dimensions
- more general proposal distributions

But, the essence of the procedure is always the same as in our discrete one-dimensional example of the islands.

We just need to be able to compute values from not normalized target distribution $P(\theta)$.

MCMC: Applied to Bayes Theorem

$$p(\theta|D) = \frac{p(D|\theta) \cdot p(\theta)}{\int d\theta \; p(D|\theta) \; p(\theta)}$$

Sample from not normalized *target distribution*

$$p(\theta|D) \propto p(D|\theta) \cdot p(\theta)$$

Random walk though continuous parameter space

Algorithm

1. start at arbitrary point
2. (randomly) propose a movement
   - proposal distribution can have any form
   - should be center round zero
   - typically normal distribution
3. Decide accept proposal or not
   - calculate $p(\theta_{\text{move}})$
   - draw random number between $[0, 1]$

Good news!

- This can be automatised
- The software packages like BUGS or JAGS will do the sampling for us.

JAGS

JAGS: "Just Another Gibbs Sampler"

JAGS

- using the model description language BUGS
- `http://mcmc-jags.sourceforge.net/`

BUGS language (WinBUGS)

- probabilistic modelling language
- Markov chain Monte Carlo (MCMC) methods.
- Declarative Programming language (!)
  - In contrast to imperative language, a declarative language declares *what* needs to be done rather than *how* to do it.

Similar software: Stan, NIMBLE

MCMC sampling in R and JAGS

Data

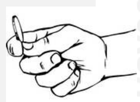
- $y_i = \{y_1 \ldots .y_N\}$: coin flips
- N: number of flips

Model

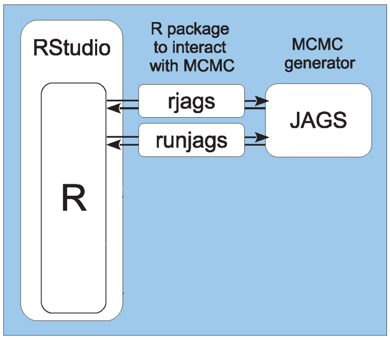
- likelihood: flips are Bernoulli distributed
  $p(y_i|\theta) = \theta^y \, (1-\theta)^{(1-y)}$
- prior $p(\theta)$: uniform
  $\text{unif}(0, 1) = \text{beta}(1, 1)$

```
# The JAGS model
model
{
  # Likelihood
  for (i in 1:N)
  {
     y[i] ~  dbern( theta )
  }

  # Prior of theta
  theta ~ dunif(0, 1)
}
```



JAGS in R



Distributions in R and JAGS

Distribution	Density	Distribution	Quantile
Bernoulli	dbern	pbern	qbern
Beta	dbeta	pbeta	qbeta
Binomial	dbin	pbin	qbin
Chi-square	dchisqr	pchisqr	qchisqr
Double exponential	ddexp	pdexp	qdexp
Exponential	dexp	pexp	qexp
F	df	pf	qf
Gamma	dgamma	pgamma	qgamma
Generalized gamma	dgen.gamma	pgen.gamma	qgen.gamma
Noncentral hypergeometric	dhyper	phyper	qhyper
Logistic	dlogis	plogis	qlogis
Log-normal	dlnorm	plnorm	qlnorm
Negative binomial	dnegbin	pnegbin	qnegbin
Noncentral Chi-square	dnchisqr	pnchisqr	qnchisqr
Normal	dnorm	pnorm	qnorm
Pareto	dpar	ppar	qpar
Poisson	dpois	ppois	qpois
Student t	dt	pt	qt
Weibull	dweib	pweib	qweib

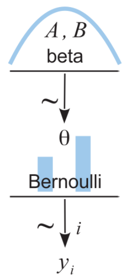
Table 5.3: Functions to calculate the probability density, probability function, and quantiles of some of the distributions provided by the **bugs** module.

MCMC Sample Diagnostic

Bayes models

```
# The JAGS model
model
{
  # Likelihood
  for (i in 1:N)
  {
     y[i] ~  dbern( theta )
  }

  # Prior of theta
  theta ~ dbeta(a, b)
  a = 1
  b = 1
}
```



Main goals of generating MCMC samples

Samples should be

1. **representative** of the posterior distribution
2. sufficient size so that estimates are **accurate** and **stable**
   - burn-in
   - thinning
3. sample process should be **efficient**

Trace plots (first 500 samples)

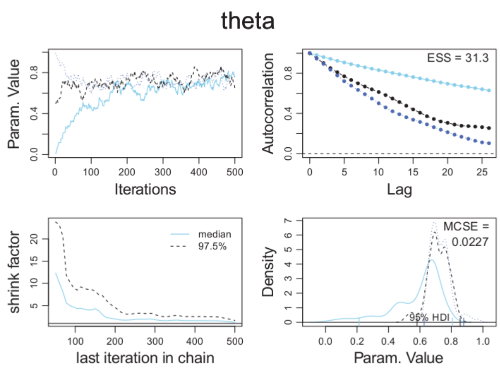


Figure 7.10: Illustration of MCMC diagnostics. Three chains were generated by starting a Metropolis algorithm at different initial values, with proposal SD=0.02

Trace plots (later samples after burn in)

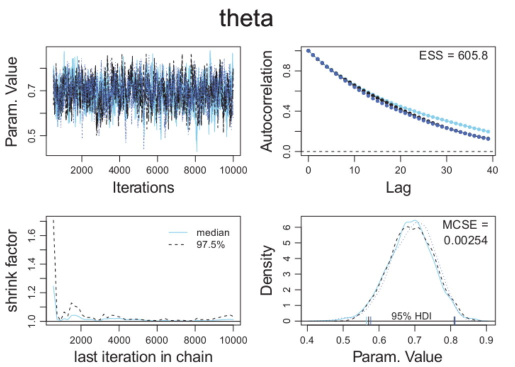


Figure 7.11: Illustration of MCMC diagnostics. Three chains were generated by

Two Coinflips

```
# two independent coin flips
model
{
  # Likelihood
  for (i in 1:N)
  {
     ya[i] ~ dbern( theta_a )
     yb[i] ~ dbern( theta_b )
  }

  theta_a ~ dbeta(a, b)
  theta_b ~ dbeta(a, b)
  a <- 1
  b <- 1
}
```