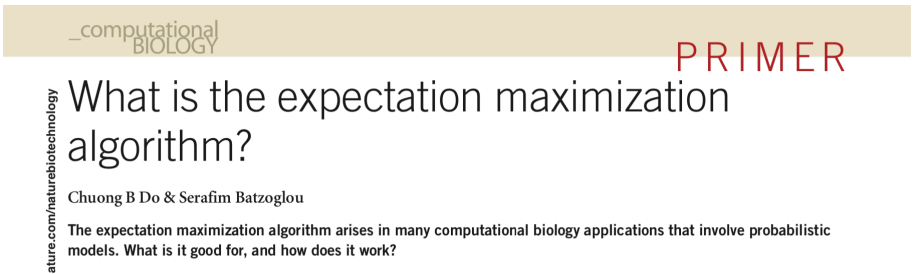
What is the Expectation Maximization (EM) Algorithm?

Kazuki Yoshida
Division of Rheumatology, Immunology and Allergy
Brigham and Women's Hospital & Harvard Medical School
@kaz_yos kaz-yos kazukiyoshida@mail.harvard.edu

2019-05-20
Mini-Statistics Camp Series
BWH Bioinformatics Club

Article Covered

- Do and Batzoglou. "What is the expectation maximization algorith?" *Nat. Biotechnol.* 2008;26:897. [Do and Batzoglou, 2008] pdf1 pdf2



To be presented as a part of the Mini Statistics Camp 2019 hosted by the BWH/HMS Bioinformatics Club. In addition to the EM Algorithm, the slides contain the corresponding Bayesian inference using the Data Augmentation method and Stan.

Motivation

- Probabilistic models, such as hidden Markov models and Bayesian networks, are commonly used to model biological data.
- Often, the only data available for training probabilistic models are incomplete, requiring special handling.
- The Expectation-Maximization (EM) algorithm enables parameter estimation in probabilistic models with incomplete data.

Incomplete data

- Ideal dataset: Matrix with all cells filled
- Incomplete data encompass:
  - Typical missing data
  - Latent class (entirely unobserved class assignment)

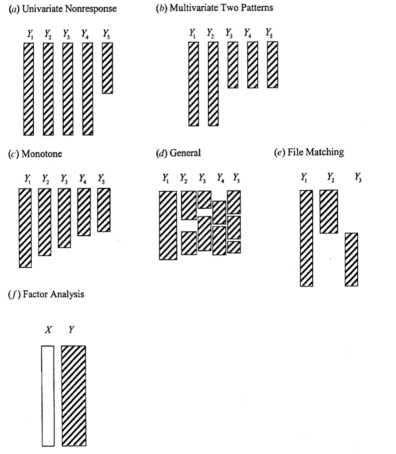


Figure 1.1. Examples of missing-data patterns. Rows correspond to observations, columns to variables.

[Little and Rubin, 2002]

EM Algorithm Applications

- Many probabilistic models in computational biology include latent variables. [Do and Batzoglou, 2008]
  - Gene expression clustering [D'haeseleer, 2005] (Gaussian mixture: "soft"-variant of k-means clustering [Bishop, 2006])
  - Motif finding [Lawrence and Reilly, 1990]
  - Haplotype inference [Excoffier and Slatkin, 1995]
  - Protein domain profiling [Krogh et al., 1994]
  - RNA family profiling [Eddy and Durbin, 1994]
  - Discovery of transcriptional modules [Segal et al., 2003]
  - Tests of linkage disequilibrium [Slatkin and Excoffier, 1996]
  - Protein identification [Nesvizhskii et al., 2003]
  - Medical imaging [De Pierro, 1995]
- Here we examine the EM algorithm using a very simple latent class model.

A Coin-Flipping Experiment

- Two coins A (0) and B (1) with unknown head probabilities (θ_0, θ_1)
- Repeat 5 times
  1. Randomly pick either coin with equal probability and record
  2. Toss 10 times and record the number of heads

A Coin-Flipping Experiment (More Formal)

- Two coins A (0) and B (1) with unknown head probabilities (θ_0, θ_1)
- For $i = 1, \ldots, 5$
  1. Draw $Z_i \sim \text{Bernoulli}(p = 0.5), Z_i \in \{0, 1\}$
  2. Draw $X_i | Z_i \sim \text{Binomial}(n = 10, p = \theta_{Z_i}), X_i \in \{0, \ldots, 10\}$

Index i	Coin Z_i	Heads X_i
1	1	5
2	0	9
3	0	8
4	1	4
5	0	7

- Note that you only need the number of heads (sufficient statistic), not the entire sequence.

Complete-Data Maximum Likelihood

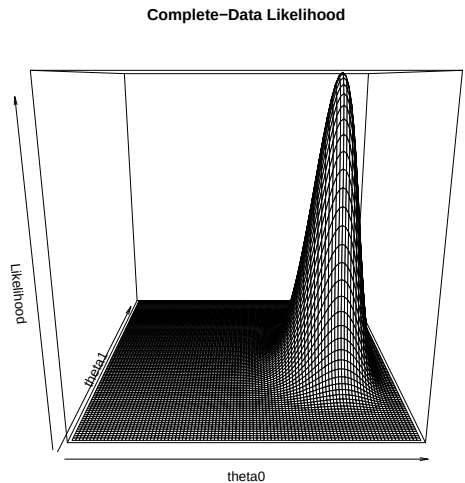
- If we observe both the coin identity Z_i and heads X_i, the MLE is the total heads / total tosses for each coin.
- Here we introduce a very redundant expanded table for later reuse.

Index i	Coin Z_i	Prob. Coin A $E[(1-Z_i)\|Z_i, X_i]$	Prob. Coin B $E[Z_i\|Z_i, X_i]$	Heads X_i	Heads Coin A $E[(1-Z_i)X_i\|Z_i, X_i]$	Heads Coin B $E[Z_iX_i\|Z_i, X_i]$
1	1 (B)	0	1	5	0×5	1×5
2	0 (A)	1	0	9	1×9	0×9
3	0 (A)	1	0	8	1×8	0×8
4	1 (B)	0	1	4	0×4	1×4
5	0 (A)	1	0	7	1×7	0×7
Sum		3	2	33	24	9

- MLE: $\widehat{\theta}_0 = 24/(3 \times 10) = 0.80$; $\widehat{\theta}_1 = 9/(2 \times 10) = 0.45$

Complete-Data Likelihood Visualization

- The maximum likelihood estimate (MLE) is the parameter values at the peak of the figure.
- Here the complete-data likelihood function has a unique global maximum.
- There is an analytical solution: Heads / Tosses for each coin.
- See the likelihood expression.



A Contrived Coin-Flipping Experiment

- Two identical-looking coins with unknown head probabilities
- Repeat 5 times
  1. You are randomly given either coin, but you do not know which.
  2. You toss 10 times, record the number of heads, and return the coin.

Index i	Coin Z_i	Heads X_i
1	?	5
2	?	9
3	?	8
4	?	4
5	?	7

- Can we still estimate the two unknown head probabilities given this incomplete data?

A Contrived Coin-Flipping Experiment (More Formal)

- Two identical-looking coins with unknown head probabilities (θ_0, θ_1) (index arbitrary)
- For $i = 1, \ldots, 5$
  1. Draw *latent* $Z_i \sim \text{Bernoulli}(p = 0.5), Z_i \in \{0, 1\}$
  2. Draw $X_i | Z_i \sim \text{Binomial}(n = 10, p = \theta_{Z_i}), X_i \in \{0, \ldots, 10\}$

Index i	Coin Z_i	Heads X_i
1	?	5
2	?	9
3	?	8
4	?	4
5	?	7

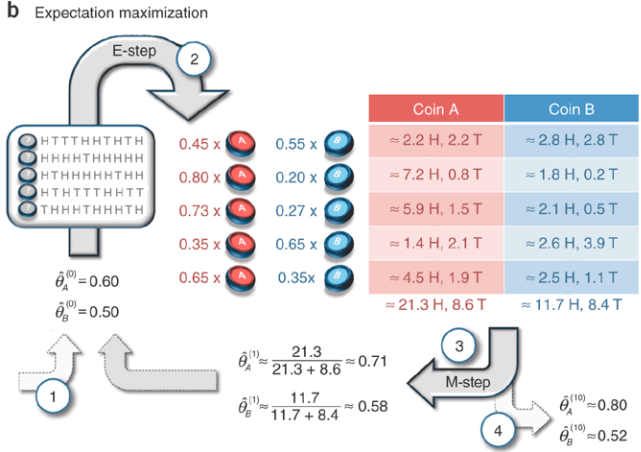
How Do We Approach Incomplete Data

- Now we cannot compute the proportion of heads among tosses for each coin.
- However, one possible iterative scheme is:
  - Assign some initial guess for parameters
  - Guess coin identities given data and assuming these parameter values
  - Perform MLE given data and assuming coin identities
- The Expectation-Maximization (EM) Algorithm [Dempster et al., 1977] is a refinement of this idea for MLE.
- The Data Augmentation Method [Tanner and Wong, 1987] is another type of refinement for Bayesian estimation.

Expectation-Maximization Algorithm

EM Algorithm to the Rescue

- The Expectation-Maximization (EM) Algorithm [Dempster et al., 1977]
- After random initialization of parameters, two steps alternates until convergence to an MLE.
- Steps repeated
  1. E-Step (compute Expected sufficient statistics):
     - Estimate probabilities of latent states given current parameters (coin identity probabilities)
     - Obtain expected sufficient statistics (weighted head counts distributed across coins)
  2. M-Step (Maximize expected log-likelihood):
     - Obtain MLE of parameters given expected sufficient statistics and update parameters
- By using weighted training data, the EM algorithm accounts for the confidence in the guessed latent state.



(**a**) Maximum likelihood estimation. For each set of ten tosses, the maximum likelihood procedure accumulates the counts of heads and tails for coins A and B separately. These counts are then used to estimate the coin biases. (**b**) Expectation maximization. 1. EM starts with an initial guess of the parameters. 2. In the E-step, a probability distribution over possible completions is computed using the current parameters. The counts shown in the table are the expected numbers of heads and tails according to this distribution. 3. In the M-step, new parameters are determined using the current completions. 4. After several repetitions of the E-step and M-step, the algorithm converges.

Parameter Initialization

- Randomly initialize the parameters
  - $\widehat{\theta}_0^{(0)} := 0.6$
  - $\widehat{\theta}_1^{(0)} := 0.5$

E-Step (0) I

- Current parameters: $\widehat{\theta}_0^{(0)} = 0.6, \widehat{\theta}_1^{(0)} = 0.5$

Index i	Coin Z_i	Prob. Coin A $E[(1-Z_i)\|X_i]$	Prob. Coin B $E[Z_i\|X_i]$	Heads X_i	Heads Coin A $E[(1-Z_i)X_i\|X_i]$	Heads Coin B $E[Z_iX_i\|X_i]$
1	?	?	?	5	? $\times$ 5	? $\times$ 5
2	?	?	?	9	? $\times$ 9	? $\times$ 9
3	?	?	?	8	? $\times$ 8	? $\times$ 8
4	?	?	?	4	? $\times$ 4	? $\times$ 4
5	?	?	?	7	? $\times$ 7	? $\times$ 7
Sum		?	?	33	?	?

- First, we need coin probabilities for each i given the current parameter values $\widehat{\boldsymbol{\theta}}^{(0)}$.

E-Step (0) II

- We will focus on $E_{\widehat{\boldsymbol{\theta}}^{(0)}}[Z_i|X_i = x_i]$, the probability of Coin B given the number of heads observed and current parameters ($\widehat{\theta}_0^{(0)} = 0.6, \widehat{\theta}_1^{(0)} = 0.5$).
- This quantity can be calculated as follows for the first row (5 heads).

```
A <- dbinom(x = 5, size = 10, prob = 0.60) # Prob. of 5 heads given Coin A
B <- dbinom(x = 5, size = 10, prob = 0.50) # Prob. of 5 heads given Coin B
B / (A + B)                                # Prob. of Coin B given 5 heads
```

```
[1] 0.5508511
```

- See also Coin Probability Derivation.

E-Step (0) III

- Now we have the probabilities of coin identities.
- This is sometimes called "soft assignment" [Hastie et al., 2016] (8.5) as opposed to "hard assignment" as in the K-means algorithm [Bishop, 2006] (9.3.2).

Index i	Coin Z_i	Prob. Coin A $E[(1-Z_i)\|X_i]$	Prob. Coin B $E[Z_i\|X_i]$	Heads X_i	Heads Coin A $E[(1-Z_i)X_i\|X_i]$	Heads Coin B $E[Z_iX_i\|X_i]$
1	?	0.45	0.55	5	$? \times 5$	$? \times 5$
2	?	0.80	0.20	9	$? \times 9$	$? \times 9$
3	?	0.73	0.27	8	$? \times 8$	$? \times 8$
4	?	0.35	0.65	4	$? \times 4$	$? \times 4$
5	?	0.65	0.35	7	$? \times 7$	$? \times 7$
Sum		2.99	2.01		?	?

- We then have to work with the sufficient statistic (head counts).

E-Step (0) IV

- Since we only know the probabilistic coin identities, we distribute the observed head counts across coins.

Index i	Coin Z_i	Prob. Coin A $E[(1-Z_i)\|X_i]$	Prob. Coin B $E[Z_i\|X_i]$	Heads X_i	Heads Coin A $E[(1-Z_i)X_i\|X_i]$	Heads Coin B $E[Z_iX_i\|X_i]$
1	?	0.45	0.55	5	0.45×5	0.55×5
2	?	0.80	0.20	9	0.80×9	0.20×9
3	?	0.73	0.27	8	0.73×8	0.27×8
4	?	0.35	0.65	4	0.35×4	0.65×4
5	?	0.65	0.35	7	0.65×7	0.35×7
Sum		2.99	2.01		?	?

- We then add up the fractional head counts for each coin.
- This pattern of "the distribution of the sufficient statistics" is characteristics to the EM in the exponential family.

E-Step (0) V

- Calculate the expected heads and consider expected tosses.

Index i	Coin Z_i	Prob. Coin A $E[(1-Z_i)\|X_i]$	Prob. Coin B $E[Z_i\|X_i]$	Heads X_i	Heads Coin A $E[(1-Z_i)X_i\|X_i]$	Heads Coin B $E[Z_iX_i\|X_i]$
1	?	0.45	0.55	5	0.45×5	0.55×5
2	?	0.80	0.20	9	0.80×9	0.20×9
3	?	0.73	0.27	8	0.73×8	0.27×8
4	?	0.35	0.65	4	0.35×4	0.65×4
5	?	0.65	0.35	7	0.65×7	0.35×7
Sum		2.99	2.01		21.3	11.7

- In expectation, Coin A was chosen 2.99 times, resulting in 29.9 expected tosses, whereas Coin B was chosen 2.01 times, resulting in 20.1 expected tosses.
- The observed heads are distributed across coins. The sums indicate 21.3 expected heads for Coin A and 11.7 expected heads for Coin B.

M-Step (0) I

- Now using the current expected heads and tosses for each coin, recalculate the MLE.

Index i	Coin Z_i	Prob. Coin A $E[(1-Z_i)\|X_i]$	Prob. Coin B $E[Z_i\|X_i]$	Heads X_i	Heads Coin A $E[(1-Z_i)X_i\|X_i]$	Heads Coin B $E[Z_iX_i\|X_i]$
1	?	0.45	0.55	5	0.45×5	0.55×5
2	?	0.80	0.20	9	0.80×9	0.20×9
3	?	0.73	0.27	8	0.73×8	0.27×8
4	?	0.35	0.65	4	0.35×4	0.65×4
5	?	0.65	0.35	7	0.65×7	0.35×7
Sum		2.99	2.01		21.3	11.7

- MLE: $\widehat{\theta}_0^{(1)} = 21.3/(2.99 \times 10) = 0.71$; $\widehat{\theta}_1^{(1)} = 11.7/(2.01 \times 10) = 0.58$
- These maximize the expected log likelihood.

E-Step (1) I

- Current parameters: $\widehat{\theta}_0^{(1)} = 0.71, \widehat{\theta}_1^{(1)} = 0.58$
- Calculate the probabilities again and update the expected tosses and heads.

Index i	Coin Z_i	Prob. Coin A $E[(1-Z_i)\|X_i]$	Prob. Coin B $E[Z_i\|X_i]$	Heads X_i	Heads Coin A $E[(1-Z_i)X_i\|X_i]$	Heads Coin B $E[Z_iX_i\|X_i]$
1	?	0.30	0.70	5	0.30×5	0.70×5
2	?	0.81	0.19	9	0.81×9	0.19×9
3	?	0.71	0.29	8	0.71×8	0.29×8
4	?	0.19	0.81	4	0.19×4	0.81×4
5	?	0.57	0.43	7	0.57×7	0.43×7
Sum		2.58	2.42	33	19.21	13.79

- In expectation, Coin A was chosen 2.58 times, resulting in 25.8 expected tosses, whereas Coin B was chosen 2.42 times, resulting in 24.2 expected tosses.
- The sums indicate 19.21 expected heads for Coin A and 13.79 expected heads for Coin B.

M-Step (1) I

- Now using the current expected heads and tosses for each coin, recalculate the MLE.

Index i	Coin Z_i	Prob. Coin A $E[(1-Z_i)\|X_i]$	Prob. Coin B $E[Z_i\|X_i]$	Heads X_i	Heads Coin A $E[(1-Z_i)X_i\|X_i]$	Heads Coin B $E[Z_iX_i\|X_i]$
1	?	0.30	0.70	5	0.30×5	0.70×5
2	?	0.81	0.19	9	0.81×9	0.19×9
3	?	0.71	0.29	8	0.71×8	0.29×8
4	?	0.19	0.81	4	0.19×4	0.81×4
5	?	0.57	0.43	7	0.57×7	0.43×7
Sum		2.58	2.42	33	19.21	13.79

- MLE: $\widehat{\theta}_0^{(2)} = 19.21/(2.58 \times 10) = 0.75$; $\widehat{\theta}_1^{(2)} = 13.79/(2.42 \times 10) = 0.57$

Automated Version I

- The `em_step` function perform one cycle of the E-step and M-step.

```
suppressMessages(library(tidyverse)); options(crayon.enabled = FALSE)
rel_dbinom <- function(X, theta) {
  p_X_Z0 <- dbinom(x = X, size = 10, prob = theta[1])
  p_X_Z1 <- dbinom(x = X, size = 10, prob = theta[2])
  tibble("Prob. Coin A" = p_X_Z0 / (p_X_Z0 + p_X_Z1),
         "Prob. Coin B" = p_X_Z1 / (p_X_Z0 + p_X_Z1))
}
em_step <- function(theta) {
  X <- c(5,9,8,4,7)
  exp_choice <- bind_rows(rel_dbinom(X[1], theta),
                          rel_dbinom(X[2], theta),
                          rel_dbinom(X[3], theta),
                          rel_dbinom(X[4], theta),
                          rel_dbinom(X[5], theta))
  exp_head <- sweep(exp_choice, MARGIN = 1, STATS = X, FUN = "*")
  colnames(exp_head) <- c("Heads Coin A","Heads Coin B")
  E <- bind_cols(tibble(Index = c(as.character(1:5), "Sum")),
                 bind_rows(exp_choice, colSums(exp_choice)),
                 tibble(X = c(X, sum(X))),
                 bind_rows(exp_head, colSums(exp_head)))
  M <- as.numeric(colSums(exp_head) / (colSums(exp_choice) * 10))
  list(E = E, M = M)
}
```

EM Step (2)

```
em_step(theta = c(0.6, 0.5)) %>% magrittr::extract2("M") %>%
  em_step() %>% magrittr::extract2("M") %>%
  em_step()
```

```
$E
# A tibble: 6 x 6
  Index `Prob. Coin A` `Prob. Coin B`     X `Heads Coin A` `Heads Coin B`
  <chr>          <dbl>          <dbl> <dbl>          <dbl>          <dbl>
1 1              0.218          0.782     5           1.09           3.91
2 2              0.870          0.130     9           7.83           1.17
3 3              0.751          0.249     8           6.01           1.99
4 4              0.112          0.888     4          0.446           3.55
5 5              0.577          0.423     7           4.04           2.96
6 Sum            2.53           2.47     33          19.4           13.6

$M
[1] 0.7680988 0.5495359
```

Iterative Version I

- The `em_iter` function fully automate the iterations until convergence at the specified tolerance.

```
em_iter <- function(theta, tolerance = 10^(-3)) {
  thetas <- tibble(theta0 = theta[1], theta1 = theta[2])
  theta_prev <- theta
  theta_curr <- em_step(theta)$M
  while (sqrt(sum((theta_curr - theta_prev)^2)) > tolerance) {
    theta_prev <- theta_curr
    thetas <- bind_rows(thetas, tibble(theta0 = theta_prev[1], theta1 = theta_prev[2]))
    theta_curr <- em_step(theta_prev)$M
  }
  thetas <- bind_rows(thetas, tibble(theta0 = theta_curr[1], theta1 = theta_curr[2]))
  return(thetas)
}
```

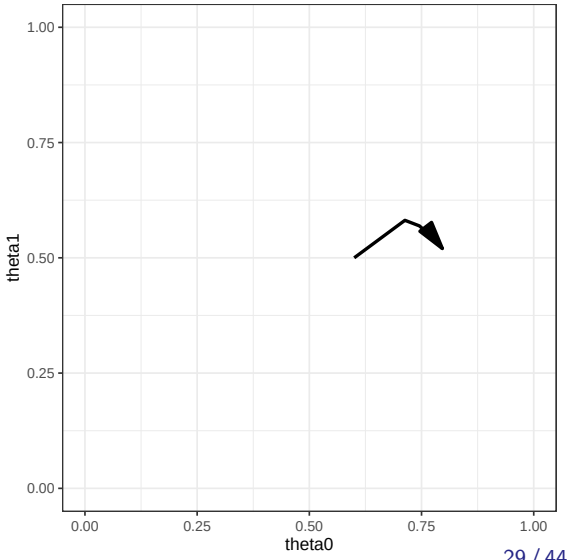
Iterative Version II

```
(em_iter_out <- em_iter(theta = c(0.6, 0.5), tolerance = 10^(-3)))
```

```
# A tibble: 9 x 2
  theta0 theta1
   <dbl>  <dbl>
1  0.6    0.5
2  0.713  0.581
3  0.745  0.569
4  0.768  0.550
5  0.783  0.535
6  0.791  0.526
7  0.795  0.522
8  0.796  0.521
9  0.796  0.520
```

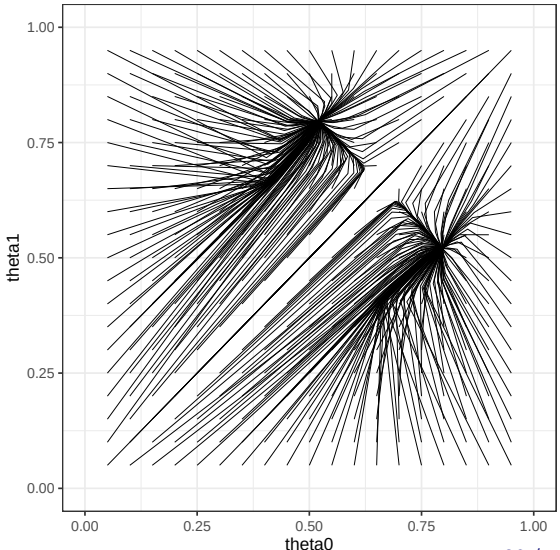
Visual Representation of Iteration

- The algorithm deterministically converge to the local maximum by monotonically improving the parameter estimate.
- As with most optimization methods for non-concave function (i.e., multiple local maxima), the EM algorithm comes with guarantees only of convergence to a local maximum.



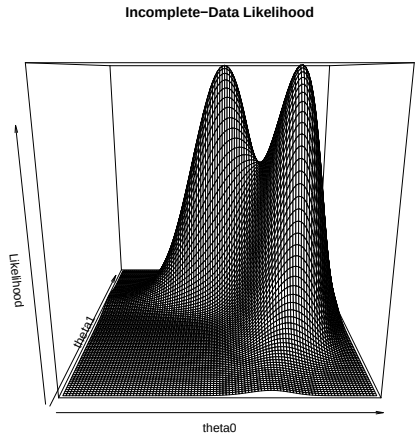
Multiple Initialization and Label Indeterminancy

- Multiple initial starting parameters are often helpful.
- In this instance, at least three $\widehat{\boldsymbol{\theta}}$ seem to exist: (0.80, 0.52), (0.52, 0.80), (0.66, 0.66).
- Note $\boldsymbol{\theta} = (0.80, 0.52)$ and $\boldsymbol{\theta} = (0.52, 0.80)$ give the same models because the labeling θ_0 and θ_1 (which coin we call 0 or 1) is arbitrary.
- $\boldsymbol{\theta} = (0.66, 0.66)$ corresponds to a model where we really only have one type of coins.



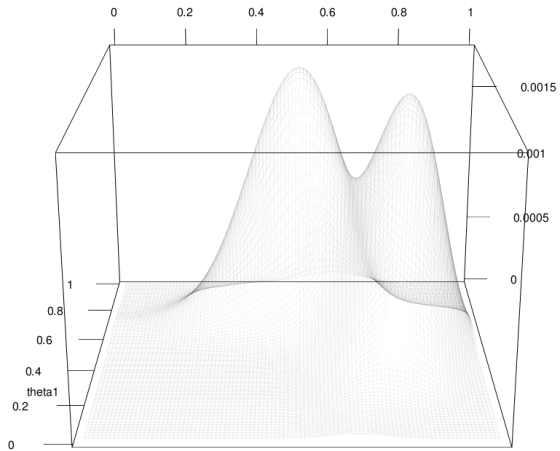
Incomplete-Data Likelihood

- In this specific instance, the incomplete-data likelihood can be graphed with grid search as the parameter space is small and low dimensional ($[0,1]^2$).
- The incomplete-data likelihood is bimodal and has a saddle point between the modes.
- This shape explains the three solutions.



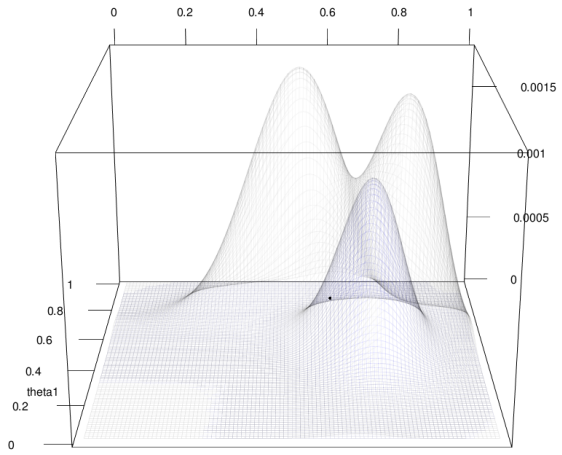
EM: Incomplete-Data Likelihood

- Incomplete-data likelihood function.
- Animated gif on Github



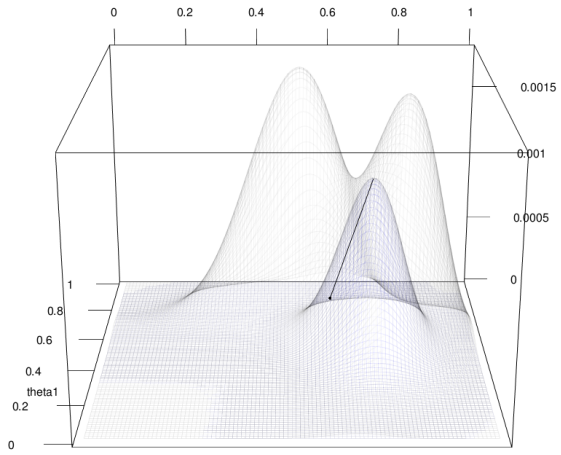
EM: E Step (0)

- $g_0(\theta)$ added



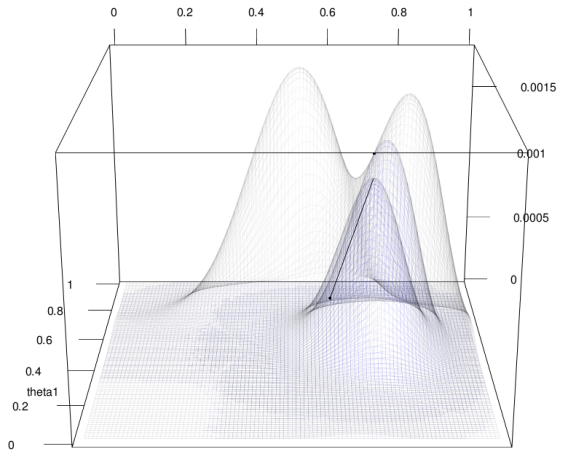
EM: M Step (0)

- $g_0(\theta)$ maximized



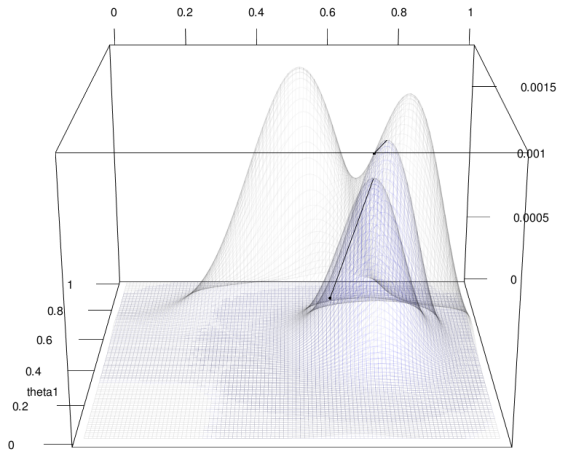
EM: E Step (1)

- $g_1(\theta)$ added



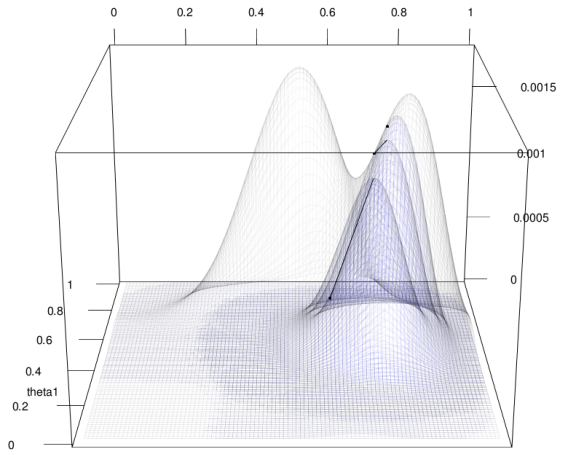
EM: M Step (1)

- $g_1(\theta)$ maximized



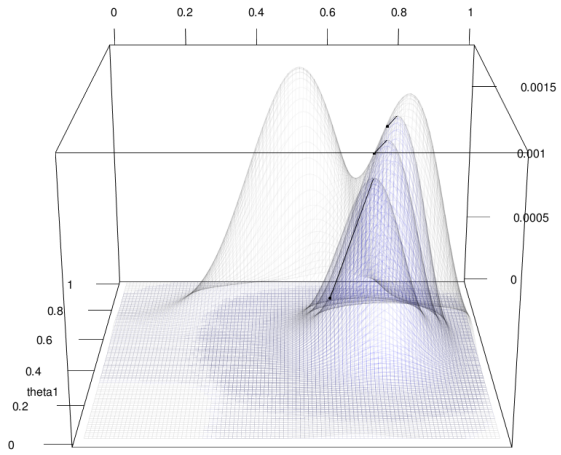
EM: E Step (2)

- $g_2(\theta)$ added



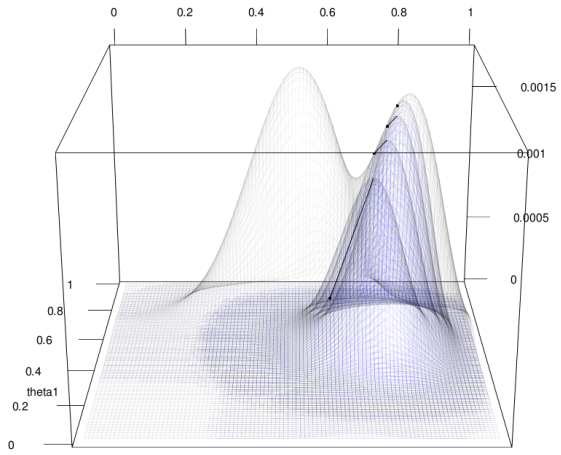
EM: M Step (2)

- $g_2(\theta)$ maximized



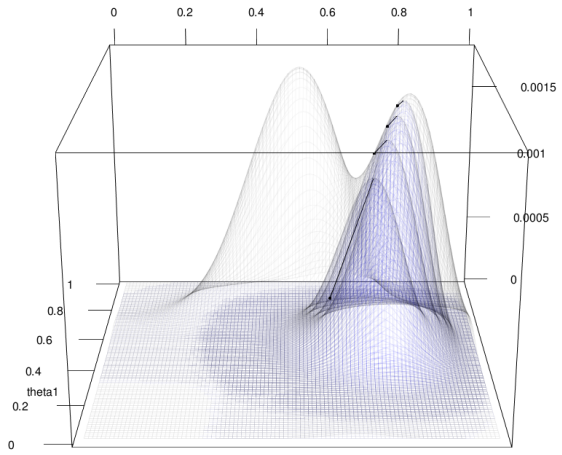
EM: E Step (3)

- $g_3(\theta)$ added



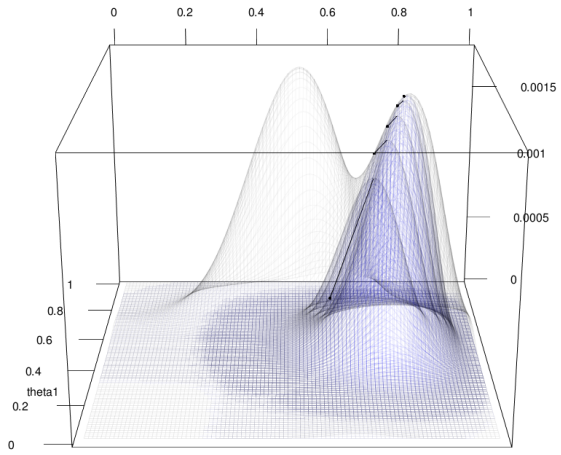
EM: M Step (3)

- $g_3(\theta)$ maximized



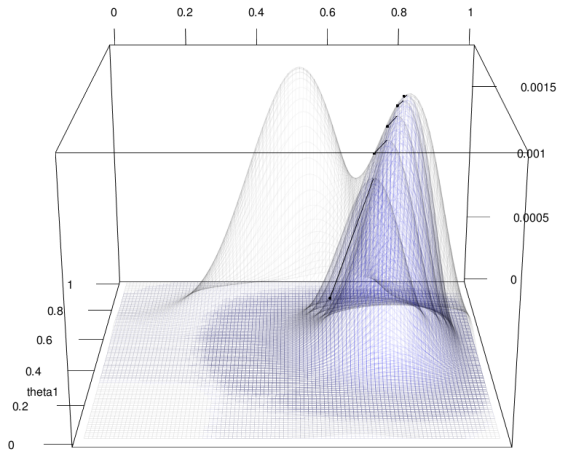
EM: E Step (4)

- $g_4(\theta)$ added



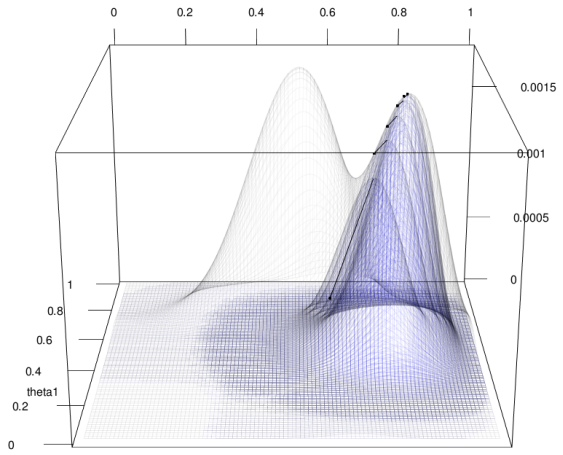
EM: M Step (4)

- $g_4(\theta)$ maximized



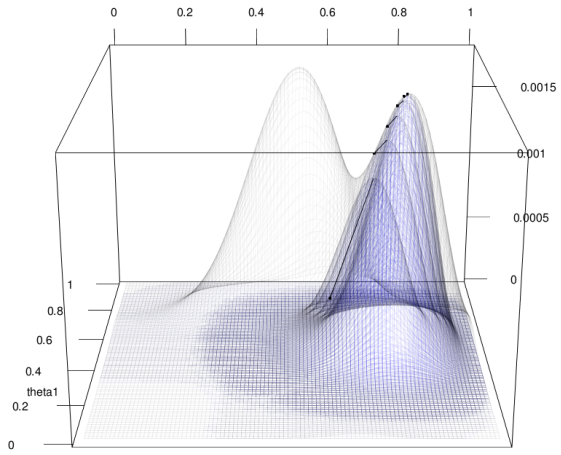
EM: E Step (5)

- $g_5(\theta)$ added



EM: M Step (5)

- $g_5(\theta)$ maximized



EM Extra Slides

Complete-Data Likelihood and Log Likelihood I

$$
\begin{aligned}
L(\boldsymbol{\theta}|\mathbf{z},\mathbf{x}) &= \prod_{i=1}^{5} p(z_i, x_i|\boldsymbol{\theta}) \\
&= \prod_{i=1}^{5} p(x_i|z_i,\boldsymbol{\theta})p(z_i|\boldsymbol{\theta}) \\
&= \prod_{i=1}^{5} p(x_i|z_i,\boldsymbol{\theta})p(z_i) \\
&= \prod_{i=1}^{5} p(x_i|z_i,\boldsymbol{\theta})(0.5)
\end{aligned}
$$

Complete-Data Likelihood and Log Likelihood II

$$\propto \prod_{i=1}^{5} \left[\theta_0^{x_i}(1-\theta_0)^{10-x_i}\right]^{1-z_i} \left[\theta_1^{x_i}(1-\theta_1)^{10-x_i}\right]^{z_i}$$

$$\log L(\boldsymbol{\theta}|\mathbf{z},\mathbf{x}) \propto \sum_{i=1}^{5} \left\{(1-z_i)\log\left[\theta_0^{x_i}(1-\theta_0)^{10-x_i}\right] + z_i \log\left[\theta_1^{x_i}(1-\theta_1)^{10-x_i}\right]\right\}$$

$$= \sum_{i=1}^{5} \{(1-z_i)\left[x_i \log\theta_0 + (10-x_i)\log(1-\theta_0)\right]$$

$$+z_i\left[x_i\log\theta_1 + (10-x_i)\log(1-\theta_1)\right]\}$$

- We can take partial derivatives with respect to θ_0 and θ_1 and set them to zero to solve for MLE, which gives us heads/tosses for each coin.

Coin Probability Derivation

- Probability of Coin B given the number of heads observed and current parameters:

$$
\begin{aligned}
E_{\widehat{\theta}^{(0)}}[Z_i|X_i = x_i] &= P_{\widehat{\theta}^{(0)}}[Z_i = 1|X_i = x_i] \\
&\quad \text{Bayes rule} \\
&= \frac{P_{\widehat{\theta}^{(0)}}[X_i = x_i|Z_i = 1]P_{\widehat{\theta}^{(0)}}[Z_i = 1]}{\sum_{z=0}^{1} P_{\widehat{\theta}^{(0)}}[X_i = x_i|Z_i = z]P_{\widehat{\theta}^{(0)}}[Z_i = z]} \\
&\quad \text{Coin choice probability} = 0.5 \\
&= \frac{P_{\widehat{\theta}^{(0)}}[X_i = x_i|Z_i = 1](0.5)}{\sum_{z=0}^{1} P_{\widehat{\theta}^{(0)}}[X_i = x_i|Z_i = z](0.5)} \\
&= \frac{P_{\widehat{\theta}^{(0)}}[X_i = x_i|Z_i = 1]}{P_{\widehat{\theta}^{(0)}}[X_i = x_i|Z_i = 0] + P_{\widehat{\theta}^{(0)}}[X_i = x_i|Z_i = 1]}
\end{aligned}
$$

Expected Log Likelihood I

$$
\begin{aligned}
& E_{\widehat{\boldsymbol{\theta}}^{(0)}}[\log L(\boldsymbol{\theta}|\mathbf{z},\mathbf{x})|\mathbf{x}] \\
\propto\ & E_{\widehat{\boldsymbol{\theta}}^{(0)}}\left[\sum_{i=1}^{5}\left\{(1-z_i)\log\left[\theta_0^{x_i}(1-\theta_0)^{10-x_i}\right]+z_i\log\left[\theta_1^{x_i}(1-\theta_1)^{10-x_i}\right]\right\}\,\middle|\,\mathbf{x}\right] \\
=\ & \sum_{i=1}^{5}\Big\{(1-E_{\widehat{\boldsymbol{\theta}}^{(0)}}[z_i|\mathbf{x}])\,[x_i\log\theta_0+(10-x_i)\log(1-\theta_0)] \\
& \qquad +E_{\widehat{\boldsymbol{\theta}}^{(0)}}[z_i|\mathbf{x}]\,[x_i\log\theta_1+(10-x_i)\log(1-\theta_1)]\Big\} \\
=\ & \sum_{i=1}^{5}\Big\{(1-E_{\widehat{\boldsymbol{\theta}}^{(0)}}[z_i|x_i])\,[x_i\log\theta_0+(10-x_i)\log(1-\theta_0)]
\end{aligned}
$$

Expected Log Likelihood II

$$+E_{\widehat{\theta}^{(0)}}[z_i|x_i]\,[x_i \log \theta_1 + (10 - x_i)\log(1 - \theta_1)]\Big\}$$

- The only change after the conditional expectation is that the coin indicators are now probabilities (*i.e.*, expectation of indicators).
- This confirms the validity of using expected heads and tails.

Incomplete-Data Likelihood Derivation I

By iid

$$L(\boldsymbol{\theta}|\mathbf{x}) = \prod_{i=1}^{5} p(x_i|\boldsymbol{\theta})$$

Introduce latent sate

$$= \prod_{i=1}^{5} \sum_{z_i=0}^{1} p(x_i, z_i|\boldsymbol{\theta})$$

$$= \prod_{i=1}^{5} \sum_{z_i=0}^{1} p(x_i|z_i, \boldsymbol{\theta}) p(z_i|\boldsymbol{\theta})$$

Incomplete-Data Likelihood Derivation II

z_i does not depend on $\boldsymbol{\theta}$

$$= \prod_{i=1}^{5} \sum_{z_i=0}^{1} p(x_i|z_i, \boldsymbol{\theta}) p(z_i)$$

$p(z_i)$ constant

$$= \prod_{i=1}^{5} \sum_{z_i=0}^{1} p(x_i|z_i, \boldsymbol{\theta})(0.5)$$

$$= \prod_{i=1}^{5} \sum_{z_i=0}^{1} (0.5) \binom{10}{x_i} \left[\theta_0^{x_i}(1-\theta_0)^{10-x_i}\right]^{1-z_i} \left[\theta_1^{x_i}(1-\theta_1)^{10-x_i}\right]^{z_i}$$

Monotone Improvement in EM Algorithm I

- This part proves that the EM Algorithm is guaranteed to improve the parameter estimate toward the local optimum every step.
  [Do and Batzoglou, 2008, Murphy, 2012]

$$\log\left(p(\mathbf{x}|\boldsymbol{\theta})\right) = \log\left(\sum_{\mathbf{z}} p(\mathbf{x},\mathbf{z}|\boldsymbol{\theta})\right)$$

Introduce arbitrary distribution Q

$$= \log\left(\sum_{\mathbf{z}} Q(\mathbf{z})\frac{p(\mathbf{x},\mathbf{z}|\boldsymbol{\theta})}{Q(\mathbf{z})}\right)$$

Rewrite as expectation

$$= \log\left(E_Q\left[\frac{p(\mathbf{x},\mathbf{z}|\boldsymbol{\theta})}{Q(\mathbf{z})}\right]\right)$$

Monotone Improvement in EM Algorithm II

Jensen's inequality on concave log

$$
\begin{aligned}
&\geq E_Q\left[\log\left(\frac{p(\mathbf{x},\mathbf{z}|\boldsymbol{\theta})}{Q(\mathbf{z})}\right)\right] \\
&= \sum_{\mathbf{z}} Q(\mathbf{z}) \log\left(\frac{p(\mathbf{x},\mathbf{z}|\boldsymbol{\theta})}{Q(\mathbf{z})}\right) \\
&= \sum_{\mathbf{z}} Q(\mathbf{z}) \log\left(\frac{p(\mathbf{z}|\mathbf{x},\boldsymbol{\theta})p(\mathbf{x}|\boldsymbol{\theta})}{Q(\mathbf{z})}\right) \\
&= \sum_{\mathbf{z}} Q(\mathbf{z}) \log\left(\frac{p(\mathbf{z}|\mathbf{x},\boldsymbol{\theta})}{Q(\mathbf{z})}\right) + \sum_{\mathbf{z}} Q(\mathbf{z}) \log\left(p(\mathbf{x}|\boldsymbol{\theta})\right) \\
&= \sum_{\mathbf{z}} Q(\mathbf{z}) \log\left(\frac{p(\mathbf{z}|\mathbf{x},\boldsymbol{\theta})}{Q(\mathbf{z})}\right) + \log\left(p(\mathbf{x}|\boldsymbol{\theta})\right) \sum_{\mathbf{z}} Q(\mathbf{z}) \\
&= \log\left(p(\mathbf{x}|\boldsymbol{\theta})\right) + \sum_{\mathbf{z}} Q(\mathbf{z}) \log\left(\frac{p(\mathbf{z}|\mathbf{x},\boldsymbol{\theta})}{Q(\mathbf{z})}\right)
\end{aligned}
$$

Monotone Improvement in EM Algorithm III

$$= \log\left(p(\mathbf{x}|\boldsymbol{\theta})\right) - \mathbb{KL}\left(Q(\mathbf{z})||p(\mathbf{z}|\mathbf{x},\boldsymbol{\theta})\right)$$

- This inequality gives the lower bound for $\log\left(p(\mathbf{x}|\boldsymbol{\theta})\right)$ for all $\boldsymbol{\theta}$.
- This lower bound is improved (maximized) by reducing the KL divergence [Murphy, 2012] by setting $Q(\mathbf{z}) = p(\mathbf{z}|\mathbf{x},\boldsymbol{\theta})$, which also gives equality.
- As $\boldsymbol{\theta}$ is the unknown quantity that we want to estimate, we can use $Q(\mathbf{z}) = p(\mathbf{z}|\mathbf{x},\widehat{\boldsymbol{\theta}}^{(t)})$ as our best available option. In this case, equality holds at $\log\left(p(\mathbf{x}|\widehat{\boldsymbol{\theta}}^{(t)})\right)$.
- Consider the following function $g_t(\boldsymbol{\theta})$, which uses $Q(\mathbf{z}) = p(\mathbf{z}|\mathbf{x},\widehat{\boldsymbol{\theta}}^{(t)})$. Note that only the numerator term within the log has a free parameter $\boldsymbol{\theta}$. [Do and Batzoglou, 2008]

Monotone Improvement in EM Algorithm IV

$$g_t(\boldsymbol{\theta}) = \sum_{\mathbf{z}} p\left(\mathbf{z}|\mathbf{x}, \widehat{\boldsymbol{\theta}}^{(t)}\right) \log\left(\frac{p(\mathbf{x}, \mathbf{z}|\boldsymbol{\theta})}{p\left(\mathbf{z}|\mathbf{x}, \widehat{\boldsymbol{\theta}}^{(t)}\right)}\right)$$

- Note that $\log\left(p(\mathbf{x}|\boldsymbol{\theta})\right) \geq g_t(\boldsymbol{\theta})$ for all $\boldsymbol{\theta}$ by the inequality.
- At $\widehat{\boldsymbol{\theta}}^{(t)}$, $g_t(\widehat{\boldsymbol{\theta}}^{(t)})$ meets the equality condition, thus, $g_t(\widehat{\boldsymbol{\theta}}^{(t)}) = \log p(\mathbf{x}|\widehat{\boldsymbol{\theta}}^{(t)})$. That is, g_t "touches" the incomplete-data likelihood function at the current parameter estimates. [Murphy, 2012]
- Consider an update rule to find $\boldsymbol{\theta}^{(t+1)}$ that maximizes this g_t function: $\widehat{\boldsymbol{\theta}}^{(t+1)} = \arg\max_{\boldsymbol{\theta}} g_t(\boldsymbol{\theta})$. Then the following inequality holds.

Monotone Improvement in EM Algorithm V

$$
\begin{aligned}
\log p\left(\mathbf{x}|\widehat{\boldsymbol{\theta}}^{(t+1)}\right) &\geq g_t\left(\widehat{\boldsymbol{\theta}}^{(t+1)}\right) && \text{By above inequality} \\
&\geq g_t\left(\widehat{\boldsymbol{\theta}}^{(t)}\right) && \text{As } \widehat{\boldsymbol{\theta}}^{(t+1)} \text{ maximizes } g_t \\
&= \log p\left(\mathbf{x}|\widehat{\boldsymbol{\theta}}^{(t)}\right) && \text{Equality holds at current value}
\end{aligned}
$$

- Therefore, $\log p\left(\mathbf{x}|\widehat{\boldsymbol{\theta}}^{(t+1)}\right) \geq \log p\left(\mathbf{x}|\widehat{\boldsymbol{\theta}}^{(t)}\right)$. That is, this update rule is guaranteed to improve the parameter estimate for the incomplete-data likelihood at each step.

Monotone Improvement in EM Algorithm VI

- Now compare this update rule to the EM algorithm.

$$
\begin{aligned}
\widehat{\boldsymbol{\theta}}^{(t+1)} &= \arg\max_{\boldsymbol{\theta}} g_t(\boldsymbol{\theta}) \\
&= \arg\max_{\boldsymbol{\theta}} \sum_{\mathbf{z}} p\left(\mathbf{z}|\mathbf{x}, \widehat{\boldsymbol{\theta}}^{(t)}\right) \log \left(\frac{p(\mathbf{x}, \mathbf{z}|\boldsymbol{\theta})}{p\left(\mathbf{z}|\mathbf{x}, \widehat{\boldsymbol{\theta}}^{(t)}\right)} \right) \\
&= \arg\max_{\boldsymbol{\theta}} \sum_{\mathbf{z}} p\left(\mathbf{z}|\mathbf{x}, \widehat{\boldsymbol{\theta}}^{(t)}\right) \left[\log p(\mathbf{x}, \mathbf{z}|\boldsymbol{\theta}) - \log p\left(\mathbf{z}|\mathbf{x}, \widehat{\boldsymbol{\theta}}^{(t)}\right) \right] \\
&\quad \text{Drop constant second term free of } \boldsymbol{\theta} \\
&= \arg\max_{\boldsymbol{\theta}} \sum_{\mathbf{z}} p\left(\mathbf{z}|\mathbf{x}, \widehat{\boldsymbol{\theta}}^{(t)}\right) \log p(\mathbf{x}, \mathbf{z}|\boldsymbol{\theta})
\end{aligned}
$$

Monotone Improvement in EM Algorithm VII

- This is maximization of the expected complete-data log likelihood. The expectation is over the distribution $\mathbf{z}$ given the observed data $\mathbf{x}$ and assuming the current parameter value $\widehat{\boldsymbol{\theta}}^{(t)}$.
- Therefore, the EM algorithm is equivalent to the update rule with the guaranteed improvement at each step.

Data Augmentation Method

From EM to DA

- A related Bayesian computation method is the *Data Augmentation* method. [Tanner and Wong, 1987, Tanner and Wong, 2010]
- Here we treat the simplest case that reduces to the two-step Gibbs sampling. [Geman and Geman, 1984, van Dyk and Meng, 2001]
- After random initialization of parameters, two steps alternates until convergence to a posterior distribution.
  1. Imputation (I) Step:
     - Estimate probabilities of latent states given current parameters
     - Draw a latent state
  2. Posterior (P) Step:
     - Draw new parameters given data and latent state
- The resulting parameter draws from the later sequence approximate draws from the target posterior.

Model Configuration

- To set up a Bayesian computation, we need probability models for the data (likelihood) as well as the parameters (prior).
- Likelihood

$$Z_i \sim \text{Bernoulli}(p = 0.5), Z_i \in \{0, 1\}$$
$$X_i | Z_i, \boldsymbol{\theta} \sim \text{Binomial}(n = 10, p = \theta_{Z_i}), X_i \in \{0, \ldots, 10\}$$

- Prior

$$\theta_0 \sim \text{Beta}(a_0, b_0)$$
$$\theta_1 \sim \text{Beta}(a_1, b_1)$$

- Here we will consider independent uniform priors ($a_j = b_j = 1, j = 0, 1$).

Parameter Initialization

- Randomly initialize the parameters
  - $\theta_0^{(0)} := 0.6$
  - $\theta_1^{(0)} := 0.5$

I-Step (1) I

- Current parameters: $\theta_0^{(0)} = 0.6, \theta_1^{(0)} = 0.5$

Index i	Coin Z_i	Prob. Coin A $E[(1-Z_i)\|X_i]$	Prob. Coin B $E[Z_i\|X_i]$	Heads X_i	Heads Coin A $E[(1-Z_i)X_i\|X_i]$	Heads Coin B $E[Z_iX_i\|X_i]$
1	?	?	?	5	? $\times$ 5	? $\times$ 5
2	?	?	?	9	? $\times$ 9	? $\times$ 9
3	?	?	?	8	? $\times$ 8	? $\times$ 8
4	?	?	?	4	? $\times$ 4	? $\times$ 4
5	?	?	?	7	? $\times$ 7	? $\times$ 7
Sum				33	?	?

- First, we need coin probabilities for each i given the current parameter values $\boldsymbol{\theta}^{(0)}$.
- This calculation is the same as the EM algorithm.

I-Step (1) II

- Now we have the probabilities of coin identities.

Index i	Coin Z_i	Prob. Coin A $E[(1-Z_i)\|X_i]$	Prob. Coin B $E[Z_i\|X_i]$	Heads X_i	Heads Coin A $E[(1-Z_i)X_i\|X_i]$	Heads Coin B $E[Z_iX_i\|X_i]$
1	?	0.45	0.55	5	? $\times$ 5	? $\times$ 5
2	?	0.80	0.20	9	? $\times$ 9	? $\times$ 9
3	?	0.73	0.27	8	? $\times$ 8	? $\times$ 8
4	?	0.35	0.65	4	? $\times$ 4	? $\times$ 4
5	?	0.65	0.35	7	? $\times$ 7	? $\times$ 7
Sum				33	?	?

- We will now draw $Z_i^{(1)}$.

```
set.seed(737265171)
rbinom(n = 5, size = 1, prob = c(0.55, 0.20, 0.27, 0.65, 0.35))
```

```
[1] 0 0 0 1 0
```

I-Step (1) III

- We have imputed the latent coin identities.

Index i	Coin Z_i	Prob. Coin A $E[(1-Z_i)\|X_i]$	Prob. Coin B $E[Z_i\|X_i]$	Heads X_i	Heads Coin A $E[(1-Z_i)X_i\|X_i]$	Heads Coin B $E[Z_iX_i\|X_i]$
1	0	0.45	0.55	5	$? \times 5$	$? \times 5$
2	0	0.80	0.20	9	$? \times 9$	$? \times 9$
3	0	0.73	0.27	8	$? \times 8$	$? \times 8$
4	1	0.35	0.65	4	$? \times 4$	$? \times 4$
5	0	0.65	0.35	7	$? \times 7$	$? \times 7$
Sum				33	?	?

- We will proceed assuming these imputed latent coin identities.

I-Step (1) IV

- We have imputed the latent coin identities.

Index i	Coin Z_i	Prob. Coin A $E[(1-Z_i)\|X_i]$	Prob. Coin B $E[Z_i\|X_i]$	Heads X_i	Heads Coin A $E[(1-Z_i)X_i\|X_i]$	Heads Coin B $E[Z_iX_i\|X_i]$
1	0	0.45	0.55	5	1×5	0×5
2	0	0.80	0.20	9	1×9	0×9
3	0	0.73	0.27	8	1×8	0×8
4	1	0.35	0.65	4	0×4	1×4
5	0	0.65	0.35	7	1×7	0×7
Sum				33	29	4

- We will proceed assuming these imputed latent coin identities.

P-Step (1) I

- Now using the complete data on $(\mathbf{Z}, \mathbf{X})$, construct a posterior $p(\boldsymbol{\theta}|\mathbf{Z}, \mathbf{X})$.

Index i	Coin Z_i	Prob. Coin A $E[(1-Z_i)\|X_i]$	Prob. Coin B $E[Z_i\|X_i]$	Heads X_i	Heads Coin A $E[(1-Z_i)X_i\|X_i]$	Heads Coin B $E[Z_iX_i\|X_i]$
1	0	0.45	0.55	5	1×5	0×5
2	0	0.80	0.20	9	1×9	0×9
3	0	0.73	0.27	8	1×8	0×8
4	1	0.35	0.65	4	0×4	1×4
5	0	0.65	0.35	7	1×7	0×7
Sum					29	4

- Using imputed coin identities, we have 29 head and 11 tails (40 tosses) for Coin A and 4 heads and 6 tails (10 tosses) for Coin B.

P-Step (1) II

- By conjugacy, we can updated the beta distributions as follows.

$$\theta_0^{(1)} \sim \text{Beta}(1 + 29, 1 + 11)$$
$$\theta_1^{(1)} \sim \text{Beta}(1 + 4, 1 + 6)$$

- Draw updated values.

```
c(rbeta(n = 1, shape1 = 1 + 29, shape2 = 1 + 11),
  rbeta(n = 1, shape1 = 1 +  4, shape2 = 1 +  6)) %>% round(3)
```

```
[1] 0.760 0.471
```

- We now have updated parameter draws: $\widehat{\theta}_0^{(1)} := 0.760$; $\widehat{\theta}_1^{(1)} := 0.471$

I-Step (2) I

- Current parameters: $\theta_0^{(1)} = 0.760, \theta_1^{(1)} = 0.471$
- Calculate the probabilities again and impute the latent states.

Index i	Coin Z_i	Prob. Coin A $E[(1-Z_i)\|X_i]$	Prob. Coin B $E[Z_i\|X_i]$	Heads X_i	Heads Coin A $E[(1-Z_i)X_i\|X_i]$	Heads Coin B $E[Z_iX_i\|X_i]$
1	1	0.17	0.83	5	0×5	1×5
2	0	0.97	0.03	9	1×9	0×9
3	0	0.90	0.10	8	1×8	0×8
4	1	0.06	0.94	4	0×4	1×4
5	0	0.73	0.27	7	1×7	0×7
Sum				33	24	9

```
rbinom(n = 5, size = 1, prob = c(0.83, 0.03, 0.10, 0.94, 0.27))
```

```
[1] 1 0 0 1 0
```

I-Step (2) II

- Using imputed coin identities, we have 24 head and 6 tails (30 tosses) for Coin A and 9 heads and 11 tail (20 tosses) for Coin B.

Index i	Coin Z_i	Prob. Coin A $E[(1-Z_i)\|X_i]$	Prob. Coin B $E[Z_i\|X_i]$	Heads X_i	Heads Coin A $E[(1-Z_i)X_i\|X_i]$	Heads Coin B $E[Z_iX_i\|X_i]$
1	1	0.17	0.83	5	0×5	1×5
2	0	0.97	0.03	9	1×9	0×9
3	0	0.90	0.10	8	1×8	0×8
4	1	0.06	0.94	4	0×4	1×4
5	0	0.73	0.27	7	1×7	0×7
Sum				33	24	9

P-Step (2) I

- We have 24 head and 6 tails (30 tosses) for Coin A and 9 heads and 11 tail (20 tosses) for Coin B.
- The posterior distributions are:

$$\theta_0^{(2)} \sim \text{Beta}(1 + 24, 1 + 6)$$
$$\theta_1^{(2)} \sim \text{Beta}(1 + 9, 1 + 11)$$

- Draw updated values.

```
c(rbeta(n = 1, shape1 = 1 + 24, shape2 = 1 +  6),
  rbeta(n = 1, shape1 = 1 +  9, shape2 = 1 + 11)) %>% round(3)
```

P-Step (2) II

```
[1] 0.677 0.483
```

- We now have updated parameter draws.
  - $\theta_0^{(2)} := 0.677$
  - $\theta_1^{(2)} := 0.483$
- In the limit, the draws for the missing data (I-Step) and the parameters (P-Step) are from the joint posterior *distribution* of the missing data and the parameters. [Little and Rubin, 2002]
- Note that this algorithm does not converge to a point unlike the EM algorithm.

Automated Version I

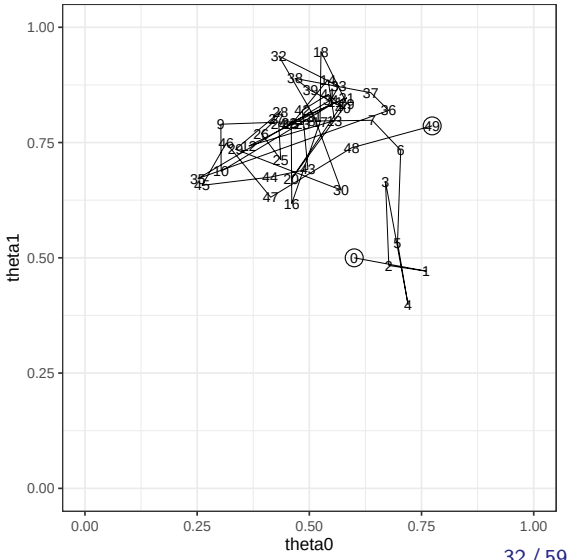
```
ip_step <- function(theta, a, b) {
  X <- c(5,9,8,4,7)
  imp_coin <- bind_rows(rel_dbinom(X[1], theta),
                        rel_dbinom(X[2], theta),
                        rel_dbinom(X[3], theta),
                        rel_dbinom(X[4], theta),
                        rel_dbinom(X[5], theta)) %>%
    mutate(Coin = rbinom(n = 5, size = 1,
                         prob = `Prob. Coin B`),
           X = X,
           `Heads Coin A` = X * (1 - Coin),
           `Heads Coin B` = X * Coin) %>%
    select(Coin, `Prob. Coin A`, `Prob. Coin B`,
           X, `Heads Coin A`, `Heads Coin B`)
  imp_coin <- bind_cols(tibble(Index = c(as.character(1:5), "Sum")),
                        bind_rows(imp_coin, colSums(imp_coin)))

  Heads_A <- imp_coin$`Heads Coin A`[6]
  Tails_A <- (5 - imp_coin$Coin[6]) * 10 - Heads_A
```

Automated Version II

```
  Heads_B <- imp_coin$`Heads Coin B`[6]
  Tails_B <- imp_coin$Coin[6] * 10 - Heads_B
  theta_post_draws <- c(rbeta(n = 1, shape1 = a[1] + Heads_A, shape2 = b[1] + Tails_A),
                        rbeta(n = 1, shape1 = a[1] + Heads_B, shape2 = b[2] + Tails_B))

  list(I = imp_coin, P = theta_post_draws)
}

ip_iter <- function(theta, a = c(1,1), b = c(1,1), iter = 10) {
  thetas <- data.frame(theta0 = c(theta[1], rep(as.numeric(NA), iter)),
                       theta1 = c(theta[2], rep(as.numeric(NA), iter)))
  for (i in seq_len(iter)) {
    thetas[i+1,] <- ip_step(as.numeric(thetas[i,]), a, b)$P
  }
  return(as.tibble(thetas))
}
```

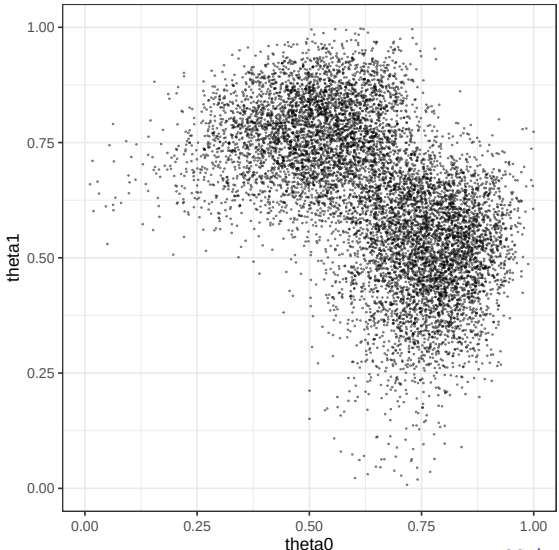
Visual Representation of Initial Iterations

- The algorithm does not converge to a point.
- Sampling is performed proportional to the posterior density.
- More samples are obtained from parameter values that are more likely.



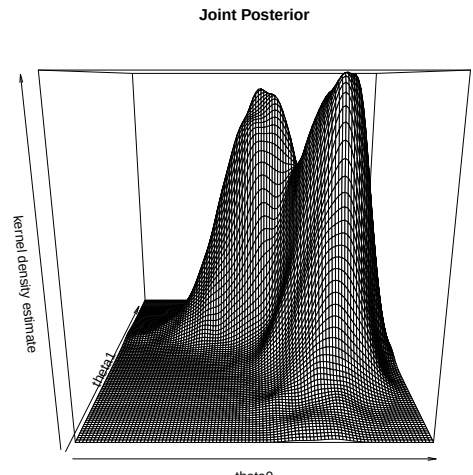
Visual Representation of Posterior Samples

- 10^4 posterior samples were obtained.
- The first 10% of posterior samples were discarded to reduce the influence of the initialization values.



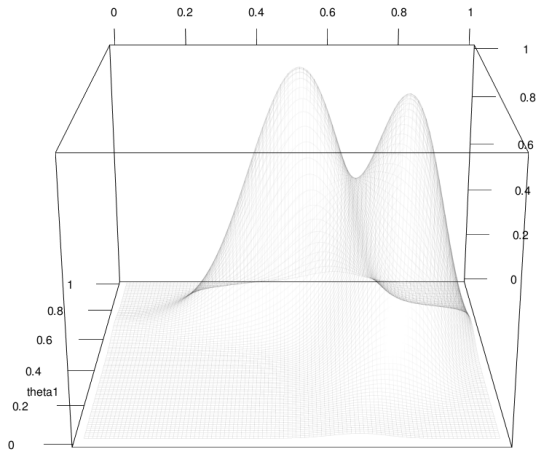
Visual Representation of Posterior Density

- The first 10% of posterior samples were discarded to reduce the influence of the initialization values.
- Similarly to the EM results with multiple initialization values, the posterior exhibits bimodality.



DA: Animation

- Incomplete-data likelihood function.
- Animated gif on Github
- Note that the marginalized posterior and tentative posteriors are not drawn to the scale.



Sampling Using Stan

Stan: Hamiltonian Monte Carlo

- Traditional Bayesian posterior sampling software, such as WinBUGS [Lunn et al., 2000] and JAGS [Plummer, 2003], are Gibbs samplers.
- Gibbs sampling in this incomplete-data setting implements the data augmentation method.
- Stan [Carpenter et al., 2017] is a modern Bayesian posterior sampler, which uses more efficient joint posterior sampling scheme based on Hamiltonian Monte Carlo (HMC) [Betancourt, 2017].
- However, HMC cannot handle discrete parameters, so the latent state have to be integrated (summed) out of the posterior (marginal posterior). [Team, 2019] (Chapter 7) [Lambert, 2018] (Chapters 16 and 19.4)
- This is very similar to the EM algorithm, which uses the marginal likelihood.

Marginalized Posterior Derivation I

- We are interested in the posterior distribution of the parameters given the observed data only.

Introduce latent state

$$p(\boldsymbol{\theta}|\mathbf{X}) = \sum_{\mathbf{z}} p(\boldsymbol{\theta}, \mathbf{z}|\mathbf{X})$$

$$= \sum_{z_1=0}^{1} \cdots \sum_{z_5=0}^{1} p(\boldsymbol{\theta}, z_1, \ldots, z_5|\mathbf{X}_1, \ldots, \mathbf{X}_5)$$

Bayes rule

$$\propto \sum_{z_1=0}^{1} \cdots \sum_{z_5=0}^{1} p(\boldsymbol{\theta}, z_1, \ldots, z_5, \mathbf{X}_1, \ldots, \mathbf{X}_5)$$

Marginalized Posterior Derivation II

iid given parameter

$$= \sum_{z_1=0}^{1} \cdots \sum_{z_5=0}^{1} \prod_{i=1}^{5} p(X_i|z_i, \boldsymbol{\theta}) p(z_i) p(\boldsymbol{\theta})$$

$$= \prod_{i=1}^{5} \sum_{z_i=0}^{1} p(X_i|z_i, \boldsymbol{\theta}) p(z_i) p(\boldsymbol{\theta})$$

$$= p(\boldsymbol{\theta}) \prod_{i=1}^{5} \sum_{z_i=0}^{1} p(X_i|z_i, \boldsymbol{\theta}) p(z_i)$$

$$= p(\boldsymbol{\theta}) \prod_{i=1}^{5} \sum_{z_i=0}^{1} p(X_i|z_i, \boldsymbol{\theta})(0.5)$$

Marginalized Posterior Derivation III

$$\propto p(\boldsymbol{\theta}) \prod_{i=1}^{5} \sum_{z_i=0}^{1} p(X_i|z_i, \boldsymbol{\theta})$$

- See How to interchange a sum and a product? for swapping the sums and the product.

Stan Implementation I

- The marginalized posterior expression can be implemented as follows in the Stan language.

```
stan_code <- readr::read_file("./coin.stan")
cat(stan_code)
```

```
/* Stan Code */
data {
    real<lower=0> a[2];
    real<lower=0> b[2];
    int<lower=0> N;
    int<lower=0> X[N];
}

parameters {
    real<lower=0,upper=1> theta[2];
}
```

Stan Implementation II

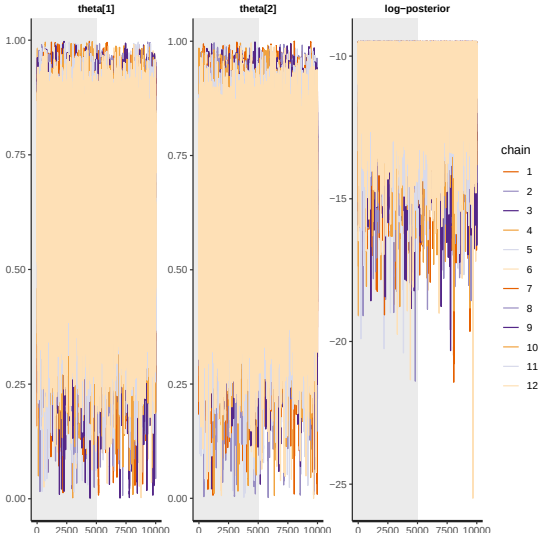
```
model {
    /* Prior's contribution to posterior log probability. */
    for (i in 1:2) {
        target += beta_lpdf(theta[i] | a[i], b[i]);
    }
    /* Data (likelihood)'s contribution to posterior log probability. */
    for (i in 1:N) {
        /* This part sums out the latent coin identity. */
        target += log_sum_exp(binomial_lpmf(X[i] | 10, theta[1]),
                              binomial_lpmf(X[i] | 10, theta[2]));
    }
}
```

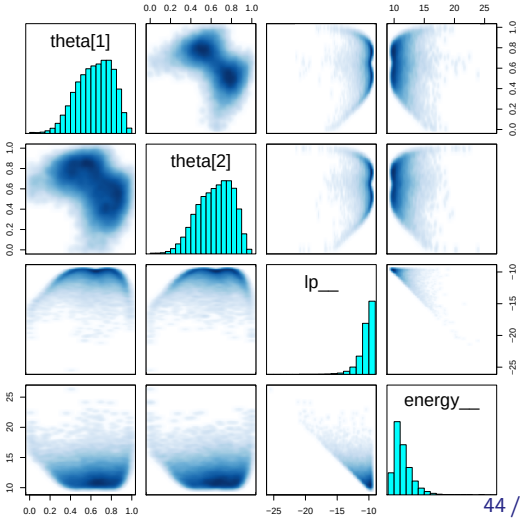
Stan Implementation III

- Printing the model object gives a summary.

```
Inference for Stan model: 4d47aa8560f5c81f88fd6a3e63af7988.
12 chains, each with iter=10000; warmup=5000; thin=1;
post-warmup draws per chain=5000, total post-warmup draws=60000.

          mean se_mean   sd   2.5%    25%    50%   75% 97.5% n_eff Rhat
theta[1]   0.64    0.00 0.17   0.29   0.51   0.65  0.77  0.92 12103    1
theta[2]   0.64    0.00 0.17   0.29   0.52   0.66  0.77  0.91 12097    1
lp__     -10.43    0.01 1.08 -13.43 -10.76 -10.05 -9.73 -9.51 14398    1

Samples were drawn using NUTS(diag_e) at Sun Apr 28 05:16:22 2019.
For each parameter, n_eff is a crude measure of effective sample size,
and Rhat is the potential scale reduction factor on split chains (at
convergence, Rhat=1).
```

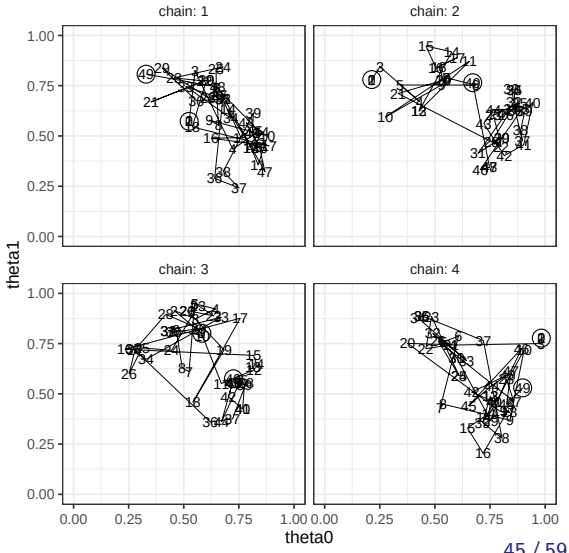
Stan Diagnostic Plots





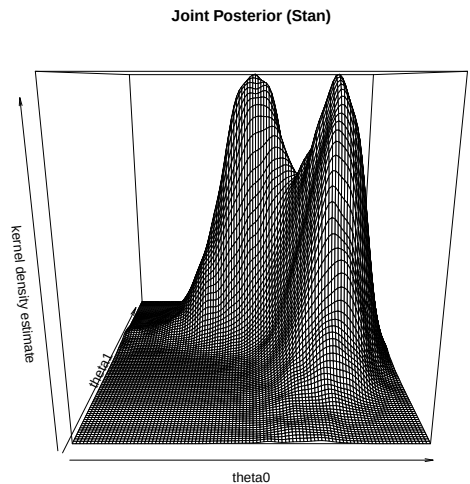
Visual Representation of Stan Initial Iterations

- Duplicated points are rejected HMC proposal



Stan Posterior Samples

- The posterior distribution is essentially the same as the data augmentation version.
- The same bimodality issue persists.
- However, quantities that do not refer to a specific component like the posterior predictive for a new observation have no issue. [Team, 2019] (Chapter 5.5)



Stan Implementation (Ordered Prior) I

- We can avoid the indeterminancy (bimodality) by restricting the prior such that $\theta_0 \leq \theta_1$ with probability 1, which in turn restricts the posterior.
- This approach required some tricks as explained in the Stan code.

```
stan_code_ordered <- readr::read_file("./coin_ordered.stan")
cat(stan_code_ordered)
```

Stan Implementation (Ordered Prior) II

```
/* Stan Code */
data {
    real<lower=0> a[2];
    real<lower=0> b[2];
    int<lower=0> N;
    int<lower=0> X[N];
}

parameters {
    /* ordered cannot be directly used with <lower=0,upper=1> */
    /* https://groups.google.com/forum/#!topic/stan-users/7r02EU7mL3o */
    /* https://mc-stan.org/docs/2_19/stan-users-guide/reparameterizations.html */
    /* This means that HMC works with the density function for log_odds_theta? */
    ordered[2] log_odds_theta;
}

transformed parameters {
    real<lower=0,upper=1> theta[2];
    for (j in 1:2) {
```

Stan Implementation (Ordered Prior) III

```
        theta[j] = inv_logit(log_odds_theta[j]);
    }
}

model {
    /* Prior's contribution to posterior log probability. */
    for (j in 1:2) {
        /* Need for Jacobian adjustment */
        /* http://rpubs.com/kaz_yos/stan_jacobian */
        /*  */
        /* We have to make the distribution of log_odss_theta (eta here) contribute. */
        /* Let eta = log(theta / (1 - theta)) */
        /* theta = e^eta / (1 + e^eta) = expit(eta) */
        /* Check: https://www.wolframalpha.com/input/?i=e%5Ex+%2F+(1+%2Be%5Ex) */
        /* d theta / d eta = d expit(eta) / d eta = e^eta / (1 + e^eta)^2 */
        /* f_eta(eta) = f_theta(expit(eta)) * |d expit(eta) / d eta| */
        /*            = f_theta(theta) * (e^eta / (1 + e^eta)^2) */
        /*  */
        /* log density contributions */
        /* log f_theta(theta) can be evaluated using beta_lpdf */
```

Stan Implementation (Ordered Prior) IV

```
        /* log (e^eta / (1 + e^eta)^2) = eta - 2 * log(1 + e^eta) */
        /*  */
        /* Beta part */
        target += beta_lpdf(theta[j] | a[j], b[j]);
        /* Jacobian part */
        target += log_odds_theta[j] - 2 * log(1 + exp(log_odds_theta[j]));
    }
    /*  */
    /* Data (likelihood)'s contribution to posterior log probability. */
    for (i in 1:N) {
        /* This part sums out the latent coin identity. */
        target += log_sum_exp(binomial_lpmf(X[i] | 10, theta[1]),
                              binomial_lpmf(X[i] | 10, theta[2]));
    }
}
```

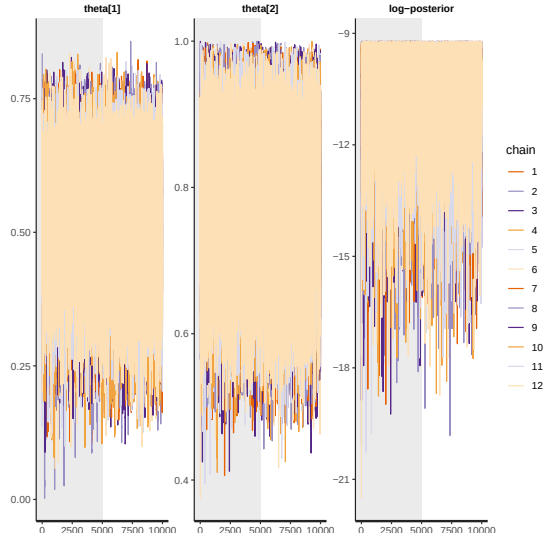
Stan Implementation (Ordered Prior) V

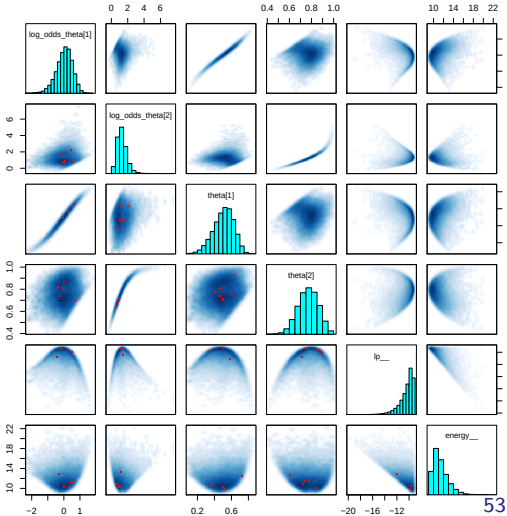
```
Inference for Stan model: 3bf7e32cefa48a5bf3972056c564adbb.
12 chains, each with iter=10000; warmup=5000; thin=1;
post-warmup draws per chain=5000, total post-warmup draws=60000.

                    mean se_mean   sd   2.5%    25%    50%   75% 97.5% n_eff
log_odds_theta[1]   0.03    0.00 0.50  -1.07  -0.28   0.07  0.38  0.90 11743
log_odds_theta[2]   1.30    0.00 0.60   0.37   0.89   1.23  1.63  2.69 35238
theta[1]            0.51    0.00 0.12   0.25   0.43   0.52  0.59  0.71 12335
theta[2]            0.77    0.00 0.09   0.59   0.71   0.77  0.84  0.94 32297
lp__              -10.41    0.01 1.15 -13.47 -10.91 -10.06 -9.56 -9.24 12808
                  Rhat
log_odds_theta[1]    1
log_odds_theta[2]    1
theta[1]             1
theta[2]             1
lp__                 1

Samples were drawn using NUTS(diag_e) at Sun Apr 28 05:17:07 2019.
For each parameter, n_eff is a crude measure of effective sample size,
and Rhat is the potential scale reduction factor on split chains (at
```

Stan Implementation (Ordered Prior) VI

```
convergence, Rhat=1).
```

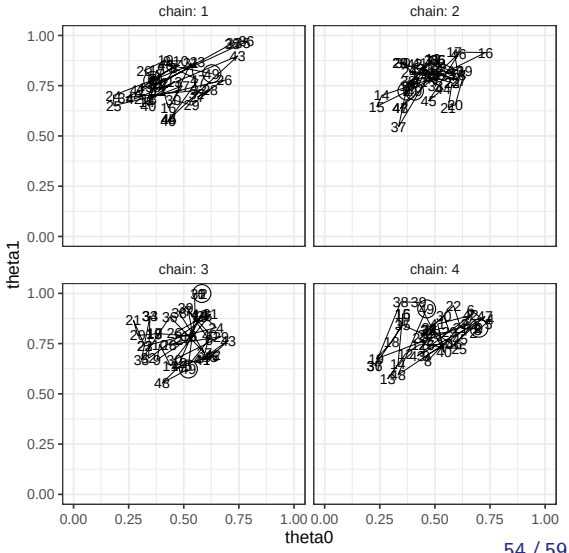
Stan Diagnostic Plots (Ordered Prior)





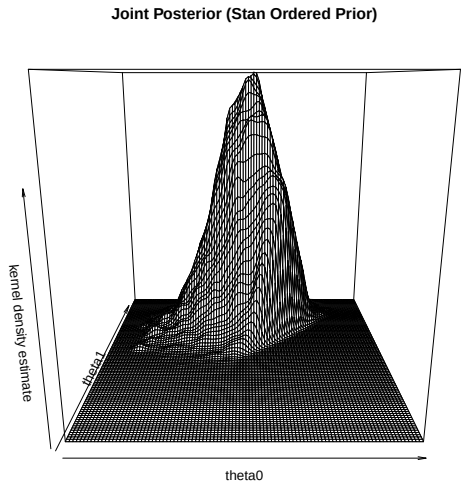
Visual Representation of Stan Initial Iterations (Ordered Prior)

- The lower diagonal part is ruled out by the prior, so the chains only wander around the upper diagonal space.



Stan Posterior Samples (Ordered Prior)

- We have a unimodal posterior as we effectively banned the lower diagonal mode previously seen.



Referenced Cited I

[Betancourt, 2017] Betancourt, M. (2017).
A Conceptual Introduction to Hamiltonian Monte Carlo.
arXiv:1701.02434 [stat].

[Bishop, 2006] Bishop, C. M. (2006).
Pattern Recognition and Machine Learning.
Information Science and Statistics. Springer, New York.

[Carpenter et al., 2017] Carpenter, B., Gelman, A., Hoffman, M. D., Lee, D., Goodrich, B., Betancourt, M., Brubaker, M., Guo, J., Li, P., and Riddell, A. (2017).
Stan: A Probabilistic Programming Language.
J Stat Softw, 76(1):1–32.

[De Pierro, 1995] De Pierro, A. R. (1995).
A modified expectation maximization algorithm for penalized likelihood estimation in emission tomography.
IEEE Trans Med Imaging, 14(1):132–137.

[Dempster et al., 1977] Dempster, A. P., Laird, N. M., and Rubin, D. B. (1977).
Maximum Likelihood from Incomplete Data via the EM Algorithm.
J. Roy. Statist. Soc. Ser. B, 39(1):1–38.

[D'haeseleer, 2005] D'haeseleer, P. (2005).
How does gene expression clustering work?
Nat. Biotechnol., 23(12):1499–1501.

[Do and Batzoglou, 2008] Do, C. B. and Batzoglou, S. (2008).
What is the expectation maximization algorithm?
Nat. Biotechnol., 26(8):897–899.

Referenced Cited II

[Eddy and Durbin, 1994] Eddy, S. R. and Durbin, R. (1994).
RNA sequence analysis using covariance models.
Nucleic Acids Res., 22(11):2079–2088.

[Excoffier and Slatkin, 1995] Excoffier, L. and Slatkin, M. (1995).
Maximum-likelihood estimation of molecular haplotype frequencies in a diploid population.
Mol. Biol. Evol., 12(5):921–927.

[Geman and Geman, 1984] Geman, S. and Geman, D. (1984).
Stochastic Relaxation, Gibbs Distributions, and the Bayesian Restoration of Images.
IEEE Transactions on Pattern Analysis and Machine Intelligence, PAMI-6(6):721–741.

[Hastie et al., 2016] Hastie, T., Tibshirani, R., and Friedman, J. (2016).
The Elements of Statistical Learning: Data Mining, Inference, and Prediction, Second Edition.
Springer, New York, NY, 2nd edition edition.

[Krogh et al., 1994] Krogh, A., Brown, M., Mian, I. S., Sjölander, K., and Haussler, D. (1994).
Hidden Markov models in computational biology. Applications to protein modeling.
J. Mol. Biol., 235(5):1501–1531.

[Lambert, 2018] Lambert, B. (2018).
A Student's Guide to Bayesian Statistics.
SAGE, Los Angeles.
OCLC: on1020621409.

[Lawrence and Reilly, 1990] Lawrence, C. E. and Reilly, A. A. (1990).
An expectation maximization (EM) algorithm for the identification and characterization of common sites in unaligned biopolymer sequences.
Proteins, 7(1):41–51.

Referenced Cited III

[Little and Rubin, 2002] Little, R. J. A. and Rubin, D. B. (2002).
Statistical Analysis with Missing Data.
Wiley-Interscience, Hoboken, N.J, 2 edition edition.

[Lunn et al., 2000] Lunn, D. J., Thomas, A., Best, N., and Spiegelhalter, D. (2000).
WinBUGS – A Bayesian Modelling Framework: Concepts, Structure, and Extensibility.
Statistics and Computing, 10(4):325–337.

[Murphy, 2012] Murphy, K. P. (2012).
Machine Learning: A Probabilistic Perspective.
Adaptive Computation and Machine Learning Series. MIT Press, Cambridge, MA.

[Nesvizhskii et al., 2003] Nesvizhskii, A. I., Keller, A., Kolker, E., and Aebersold, R. (2003).
A statistical model for identifying proteins by tandem mass spectrometry.
Anal. Chem., 75(17):4646–4658.

[Plummer, 2003] Plummer, M. (2003).
JAGS: A Program for Analysis of Bayesian Graphical Models Using Gibbs Sampling.

[Segal et al., 2003] Segal, E., Yelensky, R., and Koller, D. (2003).
Genome-wide discovery of transcriptional modules from DNA sequence and gene expression.
Bioinformatics, 19 Suppl 1:i273–282.

[Slatkin and Excoffier, 1996] Slatkin, M. and Excoffier, L. (1996).
Testing for linkage disequilibrium in genotypic data using the Expectation-Maximization algorithm.
Heredity (Edinb), 76 (Pt 4):377–383.

Referenced Cited IV

[Tanner and Wong, 1987] Tanner, M. A. and Wong, W. H. (1987).
The Calculation of Posterior Distributions by Data Augmentation.
Journal of the American Statistical Association, 82(398):528–540.

[Tanner and Wong, 2010] Tanner, M. A. and Wong, W. H. (2010).
From EM to Data Augmentation: The Emergence of MCMC Bayesian Computation in the 1980s.
Statist. Sci., 25(4):506–516.

[Team, 2019] Team, S. D. (2019).
Stan User's Guide.
2.19 edition.

[van Dyk and Meng, 2001] van Dyk, D. A. and Meng, X.-L. (2001).
The Art of Data Augmentation.
Journal of Computational and Graphical Statistics, 10(1):1–50.