# Single-node model - Bài toán đồng xu lệch (bias coin)

# Objective

* Nắm bắt khái niệm cơ bản Bayesian-MCMC
* So sánh cách tiếp cận bài toán trên quan điểm frequentis và bayesian
* Xây dựng mô hình stan trên R cho bài toán bằng package bayvl

# Problem

Bài toán tung đồng xu là bài toán kinh điển trong xác suất thống kê. Nếu chúng ta có 1 đồng xu "xịn", hai mặt đồng đều, ta có thể giả định xác suất khi tung đồng xu ra mặt head và mặt tail là đồng khả năng.

Khi đó việc tính xác suất mặt head khá đơn giản là chia đôi = 50%

Bây giờ nếu chúng ta giả định đồng xu của chúng ta bị méo. Xác suất khi tung ra 2 mặt không đồng đều. Vậy có cách nào để đo độ méo của đồng xu không?

# Solve

**Hướng tiếp cận frequentis:**

Chúng ta có thể thực nghiệm tung đồng xu, giả sử khi tung 10 lần cho kết quả các lần tung liên tiếp như dưới đây (1 là mặt head, 0 là tail)

66.6%

75%

40%

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | 0 | 1 | 1 | 0 | 1 | 0 | 0 | 0 | 0 |

Như vậy, ta có thể tính được xác suất đồng xu ra mặt head, tuy nhiên giá trị này thay đổi sau mỗi lần tung.

Kết luận xác suất ra mặt head hay độ lệch của đồng xu thế nào là rất khó khăn. Chúng ta có thể cho rằng khi số lượt thử nghiệm càng nhiều, xác suất có được khi thực nghiệm sẽ ngày càng tiệm cận về độ lệch thực, tuy nhiêu bao nhiêu được gọi là "nhiều"? Ba lần, 10 lần hay 100 lần, 1000 lần hay nhiều hơn nữa...?

**Hướng tiếp cận baysian-MCMC:**

Với các phép đo khác, ví dụ khi đi mua thịt, để biết thịt nặng bao nhiêu, chúng ta có thể sử dụng phương pháp cân. Bằng việc sử dụng các quả cân có trọng lượng chuẩn, chọn quả cân đầu tiên bất kỳ, khi thực nghiệm so sánh dần các quả cân với miếng thịt và điều chỉnh theo ta có thể xác định tương đối chính xác trọng lượng miếng thịt và quan trọng nhất là kết quả này "có thể tin được". Nếu số lần cân càng nhiều với các quả cân chia nhỏ hơn, kết quả sẽ càng chính xác hơn.

Vậy giả sử chúng ta có "quả cân" cho đồng xu của chúng ta, liệu chúng ta cách nào để "cân" được độ lệch của đồng xu của chúng ta không?

Thuật toán bayesian-MCMC có thể được mô tả như một cách để "cân" đồng xu theo các bước như sau:

- Đầu tiên chúng ta có bộ dữ liệu thực nghiệm tung đồng xu 10 lần với kết quả như trên.

- Bước tiếp theo chúng ta thử với một bộ các "quả cân" là các đồng xu có độ lệch khác nhau biết trước, 50%, 51%, 49%....

- Chọn "quả cân" đầu tiên để bắt đầu thử. Quả cân chọn đầu tiên thường là quả cân ta tin rằng gần với vật cần cân nhất. Ví dụ, ta tin rằng đồng xu của chúng ta đều 50%, ta có thể chọn quả cân 50%

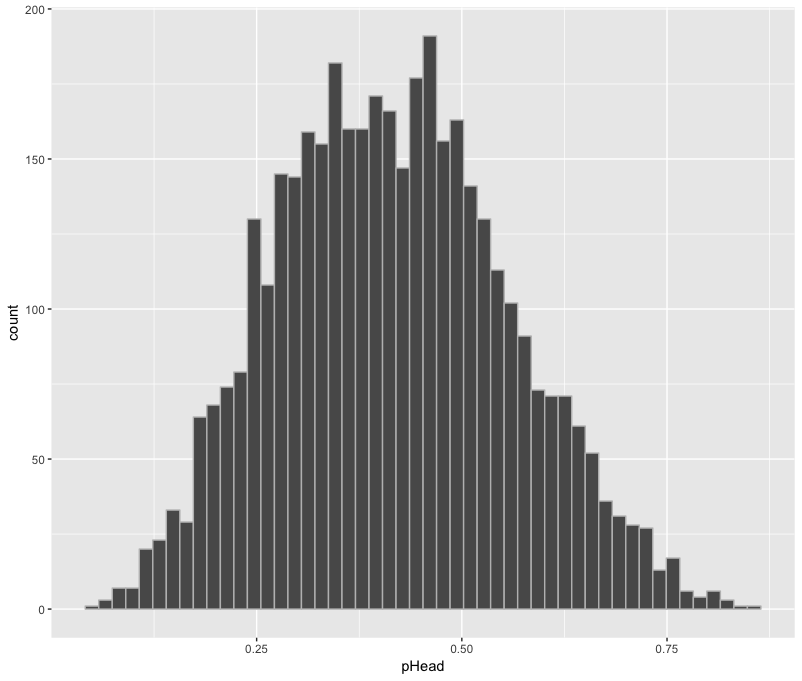
- Với mỗi "quả cân", thao tác "cân" được định nghĩa là thao tác thực nghiệm tung 10 lần "quả cân", kết quả sẽ được so sánh với kết quả tung đồng xu méo của chúng ta xem có khớp không. Ví dụ, giả xử chúng ta làm n lần "cân" với mỗi quả cân và xem trong n lần có bao nhiêu cho kết quả khớp với bộ dữ liệu đồng xu méo ở bước 1.

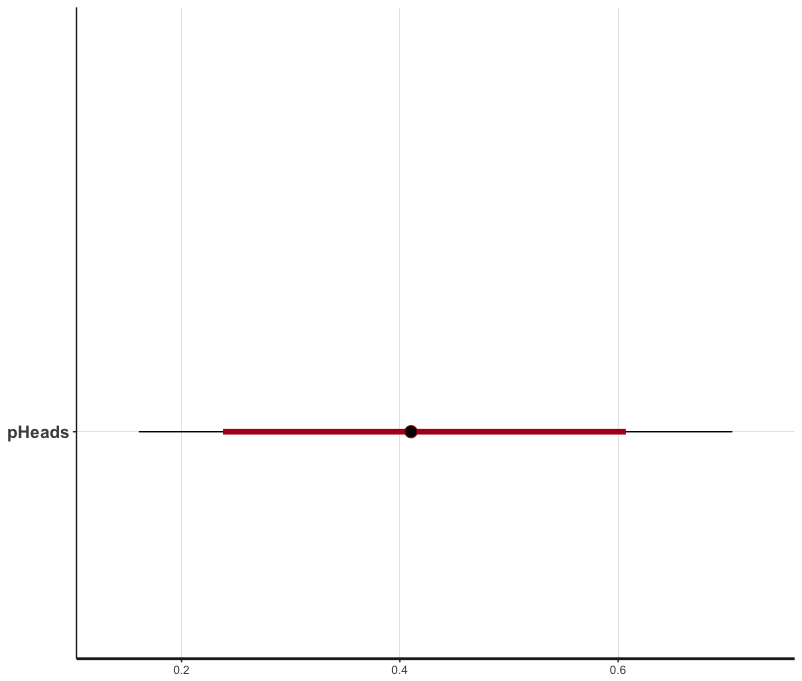
- Ghi nhận kết quả, điều chỉnh theo kết quả "cân" và chuyển sang "quả cân" khác, lại làm n lần "cân".

- Toàn bộ quá trình "cân" như trên kết quả sẽ dần hội tụ về "quả cân" có kết quả sát với đồng xu méo của chúng ta nhiều nhất.

*Giải thích "quả cân" chưa rõ ràng, quả cân là 1 phân phối???*

Nếu vẽ đồ thị kết quả "cân" chúng ta sẽ có phân phối dạng:





*Stan là gì?*

Stan là một ngôn ngữ lập trình xác suất được sử dụng cho các suy luận thống kê (Stan is a probabilistic programming language for statistical inference)

*Bayesvl in R:*

Sử dụng package bayesvl bài toán sẽ được mô hình dưới dạng lưới 1 biến (single-node model) như sau:

|  |
| --- |
| dag <- bayesvl()  dag <- bvl\_addNode(dag, "y", "bern","uniform(0, 1)")  dag <- bvl\_model2Stan(dag)  cat(dag@stancode)  N = 10 # Specify the total number of flips, denoted N.  data\_list <- c(1,0,1,1,0,1,0,0,0,0) # the trials of bias coin  data <- list(Nobs=N, y=data\_list)  fit <- bvl\_modelFit(dag, data) |

Trong đó y là biến với phân phối binomial sử dụng hàm likelyhood bernoulli, hàm prior uniform(0,1).

Như trên code ta có thể thấy ta chọn "quả cân" đầu tiên là "quả cân" phân phối đều dựa trên phân phối **uniform(0,1)**

Ta có thể chọn các "quả cân" để thử đầu tiên khác nhau nhưng kết quả sẽ cho ta khác biệt không nhiều.

Mô hình thể hiện 1 node lưới:

~ bernoulli(theta\_y)

Code bằng stan có thể được tạo tự động bằng package bayesvl, nếu sử dụng code stan trực tiếp trên R để thực hiện "cân" đồng xu ta có code sau:

|  |
| --- |
| library(rstan)  # The Stan model as a string.  model\_string <- "  // Here we define the data we are going to pass into the model  data {  int<lower=0> Nobs; // Number of trials  int<lower=0,upper=1> y[Nobs]; // Sample of N flips (heads=1, tails=0)  }  // Here we define what 'unknowns' aka parameters we have.  parameters {  real<lower=0, upper=1> pHeads; // Probability of heads  }  // The generative model  model {  pHeads ~ uniform(0, 1);  for (n in 1:Nobs)  y[n] ~ bernoulli(pHeads);  }  "  N = 10 # Specify the total number of flips, denoted N.  data\_list <- c(1,0,1,1,0,1,0,0,0,0) # the trials of bias coin  data <- list(Nobs=N, y=data\_list)  # Compiling and producing posterior samples from the model.  stan\_samples <- stan(model\_code = model\_string, data = data)  # Plotting and summarizing the posterior distribution  stan\_samples  traceplot(stan\_samples)  plot(stan\_samples)  # Export the samples to a data.frame for easier handling.  posterior <- as.data.frame(stan\_samples) |