# 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%

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...? Độ lệch 40% (0.4) có thể coi là “tin cậy” chưa?

**Hướng tiếp cận baysian-MCMC:**

Bây giờ, giả sử ta có trong tay 1 tập các đồng xu lệch ở các mức độ khác nhau, 0.1, 0.2…0.99. Nếu ta lấy 1 đồng xu có độ lệch x bất kỳ ra gieo thử, sẽ có xác xuất px là khi gieo 10 lần đạt kết quả như trên. Hay nói cách khác ta có xác xuất px là đồng xu của chúng ta chính là đồng xu có độ lệch x.

Nếu mỗi đồng xu ta gieo 10,000 và thống kê lại, ta sẽ có 1 phân phối ghi nhận số lần thành công của các đồng xu ở các độ lệch khác nhau.

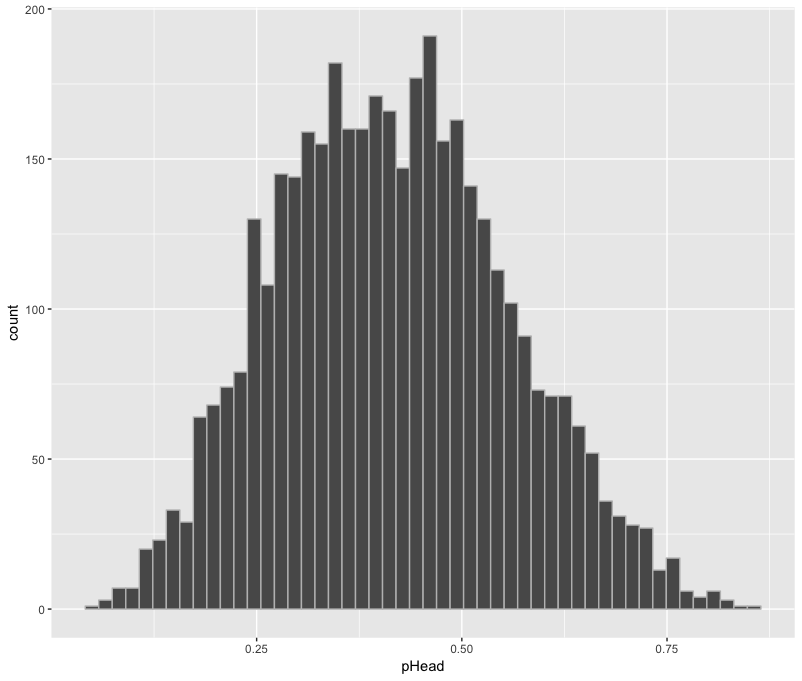


Fig 1

Ta thấy rằng mean của phân phối này ở loanh quanh khoảng đâu đó 4.1, hay nói cách khác, đồng xu của chúng ta có nhiều khả năng là đồng xu lệch 4.1.

Với các phép đo khác, ví dụ khi muốn xác định trọng lượng của 1 vật, chúng ta có thể sử dụng phương pháp cân. Bằng cách sử dụng bộ các quả cân có trọng lượng chuẩn, chọn quả cân đầu tiên bất kỳ, khi thực nghiệm so sánh quả cân với vật cần đo và điều chỉnh theo độ lệch, bằng nhiều lần cân ta có thể xác định quả cân có trọng lượng gần với vật cần cân nhất, 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.

Tương tự như 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 là như là một phương pháp sử dụng mô phỏng 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.

- Ta chọn quả cân đầu tiên là 1 phân phối của các đồng xu lệch khác nhau như trên Fig 1. 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 là đồng xu đều 50%, ta có thể chọn quả cân là phân phối chuẩn, trong đó mean của phân phối là ở 0.5. Giá trị sigma quyết định phân phối rộng hay hẹp, nó cũng thể hiện ta tin tưởng vào đánh giá của chúng ta nhiều hay ít.

Ví dụ:

Ta thấy phân phối bên trái Fig 2 thể hiện “quả cân” 0.3, với độ tin khá cao (khoảng 95% hẹp)

Phân phối bên phải Fig 3 là quả cân 0.6, không chắc chắn lắm

Nếu ta hoàn toàn không chắc, có thể chọn phân phối uniform

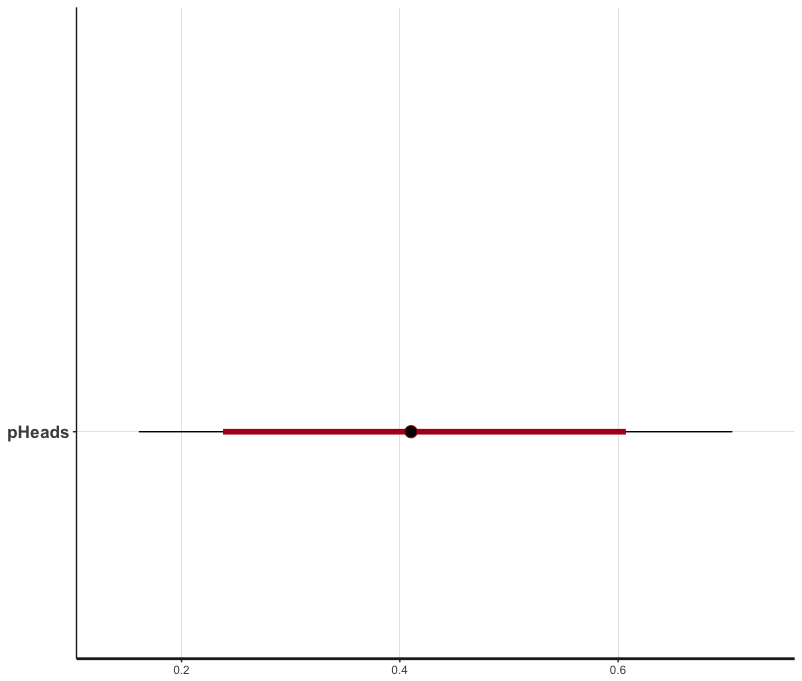
Nhìn trên phân phối này, dễ dàng thấy ta chẳng thiên về độ lệch nào cả, tất cả như nhau. Đây là quả cân dễ chọn nhất.

- Với mỗi "quả cân", thao tác "cân" được định nghĩa là thao tác thực nghiệm mô phỏng tung đồng xu thử n lần, kết quả sẽ được so sánh với kết quả thực nghiệm thực tế của chúng ta xem có khớp không.

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

- 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, tức là cho ta 1 phân phối chuẩn để đánh giá.

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)

Với bài toán đồng xu, chúng ta có thể lập trình trên stan để thực hiện quá trình mô phỏng “cân” đồng xu. Tuy nhiên thao tác này đòi hỏi có kỹ năng lập trình nhất định đồng thời mỗi lần thử mô hình, chúng ta cần điều chỉnh lại code.

*Bayesvl in R:*

Package bayesvl là package cho phép thiết kế mô hình dưới dạng lưới. Ví dụ bài toán đồng xu, 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)") |

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)

Sau khi tạo mô hình, trên bayesvl, package sẽ tự động “viết” code stan cho chúng ta. Xem code được tạo ra bằng lệnh

|  |
| --- |
| dag <- bvl\_model2Stan(dag)  cat(dag@stancode) |

Để thực hiện quá trình mô phỏng với dữ liệu thực nghiệm:

|  |
| --- |
| 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) |

Nếu không sử dụng bayesvl, ta sẽ phải code stan trực tiếp trên R để thực hiện "cân" đồng xu như 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) |

Phân tích kết quả: