

The bayesvl R package.

Hướng dẫn sử dụng v0.8

Vương Quân Hoàng^(1,2)
Lã Việt Phương^(1,2)

⁽¹⁾ AISDL, Vuong & Associates

⁽²⁾ SDAG, Centre for Interdisciplinary Social Research, Phenikaa University

Email: qvuong.ulb@gmail.com; lvphuong@gmail.com



Ngày 16 tháng 5 năm 2019 (10:00 PM); Bản #6
Hà Nội, Việt Nam

Giới thiệu về dự án BayesVL

“BayesVL” là dự án dài hạn nhằm phát triển chương trình máy tính chạy trên nền tảng ngôn ngữ lập trình phục vụ tính toán thống kê R. Chương trình thống kê này tập trung vào xây dựng thuật toán ứng dụng mô phỏng MCMC, được đóng gói trong “R package” **bayesvl**^[1]. Dự án và chương trình đang phát triển, kèm theo các tài liệu hướng dẫn bao gồm cả tài liệu tham khảo, có thể truy cập mở hoàn toàn trên Github <<https://github.com/sshpa/bayesvl>>^[2].

Việc phát triển chương trình **bayesvl** sử dụng ngôn ngữ và môi trường tính toán thống kê R bắt đầu từ cuối 2017, dựa trên xu hướng mới của thế giới^[3,4]. Nhóm phát triển AISDL cũng hướng đến yêu cầu cải thiện chất lượng nghiên cứu, giải quyết các vấn đề của thống kê truyền thống như độ tin cậy của kết quả, khả năng tái lập, và

những tranh cãi về cách hiểu về “ p -value”^[5,6]. Bên cạnh đó, chúng tôi cũng nhận thấy sự kết hợp hiệu quả giữa các mẫu dữ liệu mô phỏng MCMC dù trên Stan hay JAGS với năng lực đồ họa phong phú trên R với rất nhiều các thư viện mở trên CRAN. Đồ họa và khả năng biểu đạt hình ảnh hiệu quả là phương tiện truyền tải kết quả nghiên cứu rất quan trọng^[7].

Cơ sở toán học

Định lý Bayes cho phân bố xác suất có điều kiện:

$$f(\theta|data) = \frac{f(data|\theta) \times f(\theta)}{f(data)}$$

Trong đó, $f(\theta|data)$ là phân phối posterior cho một tham số θ , $f(data|\theta)$ là mật độ lấy mẫu (*sampling density*) cho dữ liệu, $f(\theta)$ là phân phối prior cho tham số θ , $f(data)$ là xác suất ngoại biên (marginal probability) của dữ liệu. Vì mật độ lấy mẫu tỉ lệ thuận với likelihood function nên ta có thể viết lại định lý Bayes như sau:

$$\underbrace{p(\theta|data)}_{\text{posterior}} \propto \underbrace{p(data|\theta)}_{\text{likelihood}} \times \underbrace{p(\theta)}_{\text{prior}}$$

Mục tiêu của thống kê Bayesian là thể hiện mức độ bất định (uncertain) của các tham số của mô hình thông qua một phân phối xác suất prior cho, sau đó cập nhật xác suất này với dữ liệu để đưa ra phân phối posterior mà mức độ bất định đã giảm xuống.

Trong quan điểm của thống kê Bayesian, ta bắt đầu bằng một xác suất prior của một sự kiện, sau đó cập nhật niềm tin về sự kiện đó để có được xác suất posterior. Mỗi khi có thêm dữ liệu mới, xác suất posterior này trở thành prior để các phép tính tiếp theo được thực hiện. Trên thực tế, quá trình tính toán xác suất này rất tương đồng với nghiên cứu khoa học. Trong mọi nghiên cứu, dữ liệu được thu thập để đánh giá mức độ tin cậy một giả thuyết khoa học. Khi đó, ta không bắt đầu nghiên cứu bằng ở trạng thái không có niềm tin, mà những nghiên cứu trước đã cung cấp những thông tin tiền đề để quá trình update niềm tin được thực hiện.

Mức độ phát triển bayesvl v0.8

Tại thời điểm **bayesvl** được đánh số version 0.8, giữa tháng 5-2019, chương trình chứa khoảng 3000 dòng mã. Tại một số mốc thời gian trước v0.8, các phần mã đã được sử dụng để tiến hành các nghiên cứu của nhóm đã xuất bản hoặc đang trong quá trình bình duyệt^[8-11].

bayesvl v0.8 đã bao gồm hướng dẫn sử dụng tiếng Việt và tiếng Anh, và bản thân chương trình có thể sẵn sàng sử dụng trong nhiều loại bài toán thống kê.

Một số tài liệu về thống kê Bayesian

Một số tài liệu chính được nhóm phát triển chương trình nghiên cứu và sử dụng được liệt kê ở mục **References** ^[12-17], cũng như tham khảo một vài tài liệu khác có giá trị thông tin gián tiếp ^[18-23].

Trình bày Hướng dẫn **bayesvl** R Package bằng ứng dụng

Nguyên lý chính của việc trình bày nhằm hướng tới mục tiêu chính sau đây:

- Tập trung vào cách sử dụng **bayesvl**, và không lặp lại các nội dung toán học hay kỹ thuật MCMC đã trở thành tiêu chuẩn cơ bản của các sách về thống kê Bayesian.
- Sử dụng các bài toán thực, dữ liệu thực và kết quả thực để làm rõ logic từ xây dựng đầu bài, xây dựng thuật toán mô phỏng, tiến hành mô phỏng, và kiểm tra, khai thác kết quả.
- Các đoạn mã ứng dụng được tách riêng tại phần xử lý tương ứng để làm nổi bật tính năng, và kết nối cơ bản với lý thuyết.

Bài toán số 1

Bài toán sử dụng cơ sở dữ liệu “20180224_Legends_345.csv” ^[22]. Đây là cơ sở dữ liệu mã hóa các truyện cổ tích Việt Nam thành các biến số phục vụ cho việc xử lý thống kê. Bộ dữ liệu này đã được sử dụng để tiến hành phân tích Bayesian trong nghiên cứu ^[8].

Bài toán số 1 ở đây sẽ đánh giá ảnh hưởng tam giáo với nói dối trong truyện cổ tích, phân tích hành vi nhất vật chính nói dối.

Mô hình đơn giản như sau:

$$\text{Out} \sim \text{VB} + \text{VC} + \text{VT} + \text{Lie} + \text{Viol} + (\text{Int1} + \text{Int2})$$

Cài đặt **bayesvl** R Package

Chương trình **bayesvl** có thể được cài đặt trực tiếp trên R từ Github tại địa chỉ <<https://github.com/sshpa/bayesvl>> theo các lệnh cơ bản như sau:

```
> install.packages("devtools")  
> devtools::install_github("sshpa/bayesvl")
```

Nạp package BayesVL

```
> library("bayesvl")
```

Dữ liệu và đánh giá mô hình

Việc đầu tiên là chúng ta cần đưa dữ liệu vào chương trình ứng dụng **bayesvl**. Dữ liệu có hai tác dụng cơ bản:

- a) Xây dựng đầu bài;
- b) Tiến hành mô phỏng để tìm kiếm kết quả.

Dữ liệu và xây dựng mô hình

Trước hết để gọi cơ sở dữ liệu cho mô hình, chúng ta sử dụng câu lệnh trong R như sau để gọi bộ dữ liệu mẫu "Legends345" được cấp sẵn trong package **bayesvl**:

```
> Data(Legends345)  
> data1 <- Legends345  
> head(data1)
```

Các biến sử dụng bao gồm:

- Lie: nhân vật chính có nói dối không
- Viol: nhân vật chính có hành vi bạo lực không
- VB: nhân vật chính có làm theo các giá trị Phật giáo không
- VC: nhân vật chính có làm theo các giá trị Khổng giáo không
- VT: nhân vật chính có làm theo các giá trị Lão giáo không
- Int1: có tác động của các yếu tố siêu nhiên
- Int2: có tác động của các yếu tố con người
- Out: kết thúc truyện có hậu với nhân vật chính

Đây là các biến quan sát (Observation data), được thu thập trực tiếp thông qua việc đọc truyện và mã hóa các yếu tố vào bảng dữ liệu.

Mô tả toàn bộ bộ dữ liệu mẫu Legends345 có thể xem bằng lệnh:

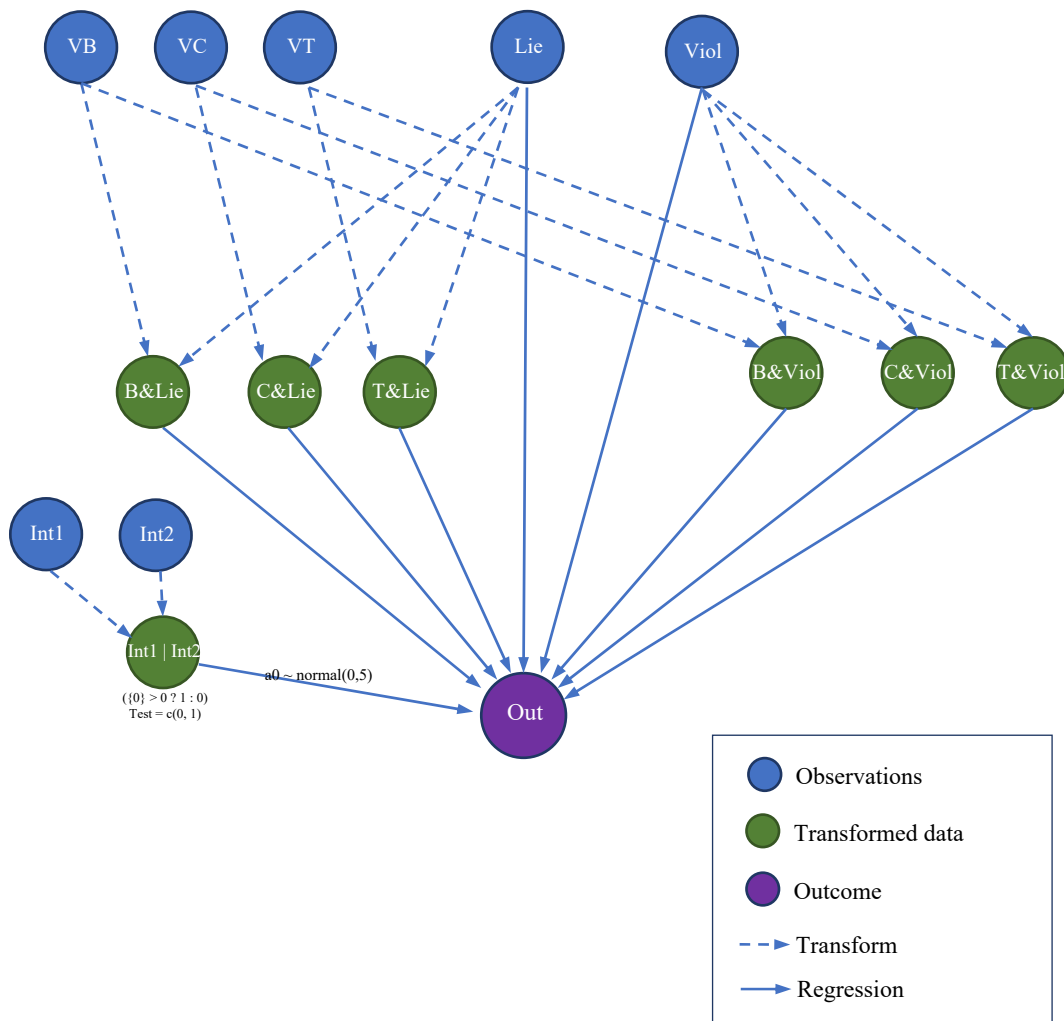
```
> help(Legends345)
```

Mô hình

Mục đích đánh giá ảnh hưởng Tam giáo với nói dối, các quan hệ được thiết lập là các quan hệ ảnh hưởng của Tam giáo “VB”, “VC”, “VT” tới hành vi nói dối “Lie” của nhân vật chính.

Đánh giá dựa trên kết thúc của tác phẩm có tốt cho nhân vật chính hay không. Trong tác phẩm, nhân vật chính nói dối có được khuyến khích bởi đạo nào không (Ví dụ nói dối nhưng vẫn thành công).

Dựa trên các yếu tố đó, mô hình sơ bộ được thiết kế như Hình 1.



Hình 1

Hình 1 thực chất là một sơ đồ logic quan hệ nhân-quả (tác động) giữa các (lớp) biến, dẫn đến kết cục “Out”.

Diễn giải mô hình 1 quan hệ:

Đây là mô hình đa lớp (multilevel) varying intercept với công thức toán cơ bản cho phương trình hồi quy tuyến tính đa lớp có dạng như sau:

$$y_i = \alpha_{j[i]} + \beta x_i + \epsilon_i$$

Khởi tạo mô hình và các biến dữ liệu quan sát (observation data) trong mô hình bằng bayesvl trên R như sau:

```
# Design the model
model <- bayesvl()
model <- bvl_addNode(model, "O", "binom")
model <- bvl_addNode(model, "Lie", "binom")
model <- bvl_addNode(model, "Viol", "binom")
model <- bvl_addNode(model, "VB", "binom")
model <- bvl_addNode(model, "VC", "binom")
model <- bvl_addNode(model, "VT", "binom")
model <- bvl_addNode(model, "Int1", "binom")
model <- bvl_addNode(model, "Int2", "binom")
```

Hàm `bvl_addNode(model, "O", "binom")` và các hàm tương tự khác là hàm mới được viết riêng cho phần mềm bayesvl. Hàm `bvl_addNode` được sử dụng để bổ sung thêm node vào đồ thị. Trong đó, hàm bao gồm các yếu tố:

- **dag:** tức `model` trong hàm ví dụ ở trên. Đây là một đối tượng để chúng ta bổ sung node. `dag` được gọi ra từ dòng code đầu tiên khi chúng ta khởi tạo mô hình: `model <- bayesvl()`.
- **name:** tức "O" trong hàm ví dụ ở trên, yếu tố `name` định danh cho một node trong mô hình của chúng ta theo các biến.
- **dist:** tức trong hàm ví dụ ở trên, **dist** định nghĩa phân phối cho các **node** gồm 2 dạng là: **normal** (chuẩn) và **binomial** (nhị thức). Khi code, hai dạng phân phối sẽ được mã hóa thành "norm" hoặc "binom". Trong trường hợp ta kết hợp hai biến đã có, dạng phân phối sẽ là "trans", tức **transform**.

Tổng thể, với hàm `bvl_addNode(model, "O", "binom")` được ví dụ ở trên, chúng ta có thể hiểu là hàm sẽ bổ sung thêm node vào mô hình `model`, node đó sẽ được định danh là "O" với phân phối ở dạng nhị thức (binomial).


Cơ sở toán học cho việc tính toán xác suất của biến x có phân phối nhị thức:


$$pr(x|n, p) = \binom{n}{x} p^x (1 - p)^{n-x}$$

Trong đó, n là số lần thử, p là xác suất x xuất hiện trong mỗi lần thử.

Hướng dẫn về các hàm dựng mô hình lưới trên bayesvl có thể tra cứu trên R bằng lệnh **help** thông dụng của R:

```
> help("bayesvl graphs")
```

Trong mô hình được khởi tạo, các biến ("O", "Lie", "Viol",...) đều được định nghĩa ở phân phối binom, các biến này thể hiện trên mô hình quan hệ Hình 1 bằng các nodes có màu **blue** .

Để đánh giá ảnh hưởng của Tam giáo ("VB", "VC", "VT") và nói dối ("Lie") lên kết thúc câu truyện, ta ghép các cặp biến Tam giáo với biến nói dối để tạo các biến tranformed data mới (trên mô hình quan hệ Hình 1 thể hiện bằng các nodes có màu **green** ):

- **B_and_Lie**: nhân vật chính làm theo các giá trị cốt lõi Phật giáo nhưng cho thấy các chi tiết/nội dung nói dối
- **C_and_Lie**: nhân vật chính làm theo các giá trị cốt lõi Khổng giáo nhưng cho thấy các chi tiết/nội dung nói dối
- **T_and_Lie**: nhân vật chính làm theo các giá trị cốt lõi Lão giáo nhưng cho thấy các chi tiết/nội dung nói dối


Để đánh giá ảnh hưởng của Tam giáo ("VB", "VC", "VT") và bạo lực ("Viol") lên kết thúc câu truyện, ta ghép các cặp biến Tam giáo với biến bạo lực để tạo các biến tranformed data mới:

- **B_and_Viol**: nhân vật chính làm theo các giá trị cốt lõi Phật giáo nhưng vẫn có hành vi bạo lực
- **C_and_Viol**: nhân vật chính làm theo các giá trị cốt lõi Khổng giáo nhưng vẫn có hành vi bạo lực
- **T_and_Viol**: nhân vật chính làm theo các giá trị cốt lõi Lão giáo nhưng vẫn có hành vi bạo lực

Về mặt toán học có thể thể hiện thông qua toán tử (*) giữa hai biến:

```
B and Lie = B * Lie
C and Lie = C * Lie
T and Lie = T * Lie

B and Viol = B * Viol
C and Viol = C * Viol
T and Viol = T * Viol
```

Quan hệ transform data thể hiện trên hình Hình 1 bằng mũi tên nét đứt (dash-line arrow ).

Tạo các nodes transformed data bằng bayesvl trên R:

```
model <- bvl addNode(model, "B and Viol", "trans")
model <- bvl addNode(model, "C and Viol", "trans")
model <- bvl addNode(model, "T and Viol", "trans")
```

Định nghĩa quan hệ transforming data như "B_and_Viol" từ các dữ liệu quan sát ghi nhận được (observation) như "VB" và "Viol" được thể hiện qua toán tử (*):

```
model <- bvl addArc(model, "VB", "B and Viol", "**")
model <- bvl addArc(model, "Viol", "B and Viol", "**")

model <- bvl addArc(model, "VC", "C and Viol", "**")
model <- bvl addArc(model, "Viol", "C and Viol", "**")

model <- bvl addArc(model, "VT", "T and Viol", "**")
model <- bvl addArc(model, "Viol", "T and Viol", "**")
```

Cũng tương tự như hàm `bvl_addNode`, hàm được viết riêng cho `bayesvl` nhằm bổ sung đường nối giữa các nodes trong một mô hình. Ví dụ, với hàm `bvl_addArc(model, "VB", "B_and_Viol", "**")` ta có các yếu tố:

- **dag**: tức `model` trong hàm ví dụ ở trên. Đây là một đối tượng để chúng ta bổ sung node. `dag` được gọi ra từ dòng code đầu tiên khi chúng ta khởi tạo mô hình: `model <- bayesvl()`.
- **from** và **to**: tức "VB" và "B_and_Viol" trong mô hình. Thể hiện chúng ta muốn tạo đường nối từ node nào đến node nào.
- Toán tử "*" định nghĩa cho quan hệ transforming data giữa hai node. Trong các trường hợp khác như quan hệ hồi quy, quan hệ giữa các nodes trong mô hình sẽ có dạng "varint" hoặc "slope".

Để đánh giá nội dung truyện thay đổi khi có tác động của các yếu tố bên ngoài (thần thánh hoặc con người) ta có 2 biến quan sát "Int1" và "Int2" tương ứng. Kết hợp 2 biến này ta có 1 biến transformed data mới:

- `Int1_or_Int2`: có tác động của yếu tố siêu nhiên hoặc yếu tố con người

Sự can thiệp của thần thánh ("Int1") bao gồm các nhân vật như ông Bụt, Bồ-tát, hay Thần Tiên. Còn sự can thiệp của con người ("Int2") tập trung chủ yếu vào các nhân vật có quyền lực thế tục như vua, quan, hay địa chủ.

Thể hiện dạng thuật toán có điều kiện, ta trình bày:

```
Int1 or Int2 = (Int1 + Int2 > 0 ? 1 : 0)
```

Như vậy biến mới bằng tổng của 2 biến quan sát "Int1" và "Int2", nếu kết quả tổng của 2 biến quan sát ban đầu > 0 thì có nghĩa là có ít nhất 1 trong 2 biến quan sát = 1 (có sự can thiệp), khi đó giá trị biến *transformed data* mới "Int1_or_Int2" sẽ = 1, nếu không có biến nào trong biến quan sát = 1 thì giá trị biến *transformed data* mới = 0.

Như vậy ta sẽ có 1 biến thể hiện có tác động ít nhất của yếu tố siêu nhiên (thần tiên, bụt,...) hoặc quyền lực con người (vua, quan,...)

Code dựng biến mới bằng bayesvl:

```
model <- bvl addNode(model, "Int1 or Int2", "trans",  
fun = "({0} > 0 ? 1 : 0)", out type = "int", lower = 0,  
test = c(0, 1))
```

Vì biến "Int1_or_Int2" chúng ta định tạo có điều kiện phức tạp hơn so với các biến khác nên hàm `bvl_addNode` trong dòng code này phức tạp hơn vì sẽ có các yếu tố cần được bổ sung thêm. Trong đó, `model`, "Int1_or_Int2" và "trans" vẫn có tính chất tương tự như trên. Bên cạnh đó:

- `fun`: là thủ tục biến đổi của node. Điều kiện của `fun` được dựa trên cơ sở toán học của biến `Int1_or_Int2` mà chúng ta đã có quy ước ở trên.
- `out_type`: thể hiện định dạng của biến số như `int` (integer - số nguyên) hay `real` (số thực).

Định nghĩa quan hệ của biến transforming data mới ("Int1_or_Int2") từ các dữ liệu quan sát ("Int1" và "Int2") được thể hiện qua toán tử (+):

```
model <- bvl addArc(model, "Int1", "Int1 or Int2", "+")  
model <- bvl addArc(model, "Int2", "Int1 or Int2", "+")
```

Để đánh giá tương quan với kết quả của truyện có hậu với nhân vật chính hay không ("O"), ta thực hiện hồi quy của các biến đã tạo với biến quan sát "O". Vì chúng ta thực hiện hồi quy nên quan hệ của các biến với biến quan "O" sẽ là quan hệ hồi quy, vì thế trong hàm `bvl_addArc` như trình bày sau đây, quan hệ giữa các nodes sẽ là "varint" hoặc "slope".

Ta dựng các quan hệ hồi quy giữa các biến *transformed data* có yếu tố nói dối với biến quan sát về kết quả ("O"):

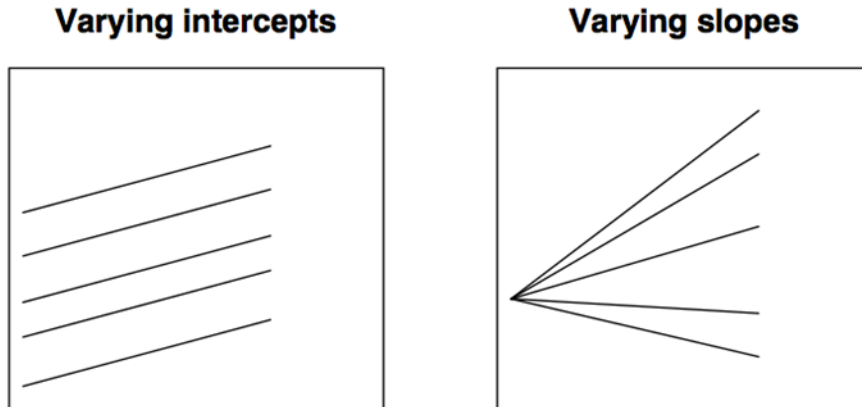
```
model <- bvl_addArc(model, "B_and_Lie", "O", "slope")  
model <- bvl_addArc(model, "C_and_Lie", "O", "slope")  
model <- bvl_addArc(model, "T_and_Lie", "O", "slope")  
  
model <- bvl_addArc(model, "Lie", "O", "slope")
```

Quan hệ hồi quy giữa các biến transformed data có yếu tố bạo lực với biến quan sát về kết quả ("O"):

```
model <- bvl_addArc(model, "B_and_Viol", "O", "slope")  
model <- bvl_addArc(model, "C_and_Viol", "O", "slope")  
model <- bvl_addArc(model, "T_and_Viol", "O", "slope")
```

```
model <- bvl_addArc(model, "Viol", "O", "slope")
```

Đây là các quan hệ hồi quy ở dạng **slope**. Trong đó, các hệ số chặn (*intercepts*) xác định vị trí đường hồi quy theo trực y của biến **outcome**, hệ số góc **slope** xác định góc của đường hồi quy. Như vậy khi hệ số *intercepts* thay đổi ta có dạng đường hồi quy:



Ta tạo quan hệ hồi quy phân lớp (varying intercepts) cho biến *Int1_or_Int2*, như vậy ta có thể đánh giá kết quả truyền khi có và không có yếu tố tác động bên ngoài:

```
model <- bvl_addArc(model, "Int1 or Int2", "O", "varint",
priors = c("a0 ~ normal(0,5)", "sigma ~ normal(0,5)"))
```

Tổng kết lại, để dựng toàn bộ mô hình toán học cho mô hình lưới quan hệ như hình Hình 1 sử dụng “**bayesvl**” trên R như sau:

```
# Design the model
model <- bayesvl()
model <- bvl_addNode(model, "O", "binom")
model <- bvl_addNode(model, "Lie", "binom")
model <- bvl_addNode(model, "Viol", "binom")
model <- bvl_addNode(model, "VB", "binom")
model <- bvl_addNode(model, "VC", "binom")
model <- bvl_addNode(model, "VT", "binom")
model <- bvl_addNode(model, "Int1", "binom")
model <- bvl_addNode(model, "Int2", "binom")

model <- bvl_addNode(model, "B and Viol", "trans")
model <- bvl_addNode(model, "C and Viol", "trans")
model <- bvl_addNode(model, "T and Viol", "trans")
model <- bvl_addArc(model, "VB", "B and Viol", "**")
model <- bvl_addArc(model, "Viol", "B and Viol", "**")
model <- bvl_addArc(model, "VC", "C and Viol", "**")
model <- bvl_addArc(model, "Viol", "C and Viol", "**")
model <- bvl_addArc(model, "VT", "T and Viol", "**")
model <- bvl_addArc(model, "Viol", "T and Viol", "**")
model <- bvl_addArc(model, "B and Viol", "O", "slope")
model <- bvl_addArc(model, "C and Viol", "O", "slope")
model <- bvl_addArc(model, "T and Viol", "O", "slope")

model <- bvl_addArc(model, "Viol", "O", "slope")
```

```

model <- bvl_addNode(model, "B_and_Lie", "trans")
model <- bvl_addNode(model, "C_and_Lie", "trans")
model <- bvl_addNode(model, "T_and_Lie", "trans")
model <- bvl_addArc(model, "VB", "B_and_Lie", "**")
model <- bvl_addArc(model, "Lie", "B_and_Lie", "**")
model <- bvl_addArc(model, "VC", "C_and_Lie", "**")
model <- bvl_addArc(model, "Lie", "C_and_Lie", "**")
model <- bvl_addArc(model, "VT", "T_and_Lie", "**")
model <- bvl_addArc(model, "Lie", "T_and_Lie", "**")
model <- bvl_addArc(model, "B_and_Lie", "O", "slope")
model <- bvl_addArc(model, "C_and_Lie", "O", "slope")
model <- bvl_addArc(model, "T_and_Lie", "O", "slope")

model <- bvl_addArc(model, "Lie", "O", "slope")

model <- bvl_addNode(model, "Int1_or_Int2", "trans",
fun = "{0} > 0 ? 1 : 0", out_type = "int", lower = 0)

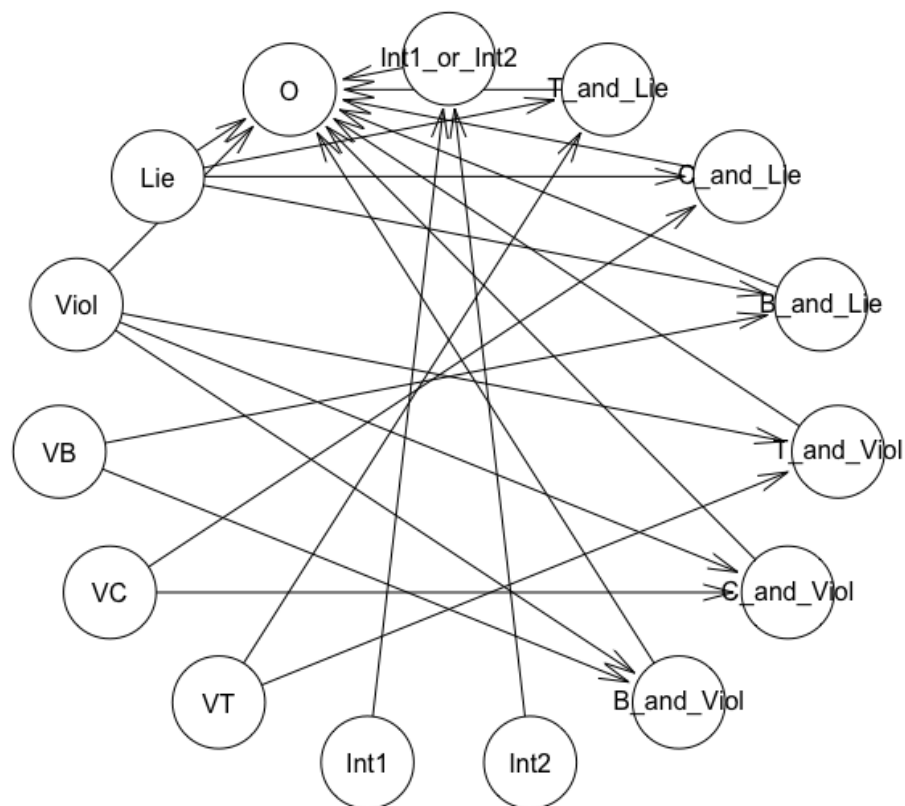
model <- bvl_addArc(model, "Int1", "Int1_or_Int2", "+")
model <- bvl_addArc(model, "Int2", "Int1_or_Int2", "+")

model <- bvl_addArc(model, "Int1_or_Int2", "O", "varint",
priors = c("a0 ~ normal(0,5)", "sigma ~ normal(0,5)"))

```

Nếu muốn vẽ lại sơ đồ lưới này trên R để kiểm tra các kết nối logic, ta sử dụng lệnh sau đây của “the **bayesvl** R package”:

```
> bvl_bnPlot(model)
```



Hình 2

Tiếp cận từ lưới Bayesian của chương trình **bayesvl** có xuất phát điểm từ việc sử dụng sơ đồ logic trong nghiên cứu về hiện tượng “Cộng tính văn hóa” đã xuất bản trên Palgrave Communications (www.nature.com/palcomms)^[8], và sau đó là nghiên cứu về phố cổ Hà Nội, với hàng loạt lưới *dag* ngẫu nhiên^[9].

Như vậy ta đã dựng lại toàn bộ mô hình quan hệ từ hình Hình 1 thành mô hình hồi quy **bayesvl** (Hình 2). Toàn bộ các công thức toán và mô hình hồi quy Stan sẽ được **bayesvl** tự động tạo theo mô hình trên.

Ta có thể kiểm tra mô hình ở từng *node* của lưới.

Ví dụ, công thức toán học *transform* dữ liệu cho *node* **B_and_Lie**, có thể được kiểm tra bằng lệnh của **bayesvl** như sau:

```
> bvl_formula(model, "B_and_Lie")
B and Lie ~ VB*Lie
```

Hoặc **Int1_or_Int2**:

```
> bvl_formula(model, "Int1 or Int2")
Int1 or Int2 ~ (Int1+Int2 > 0 ? 1 : 0)
```

Đồng thời, người sử dụng có thể kiểm tra lại toàn bộ mô hình, gồm cả các *node*, *transformed node* và logic liên kết các biến, bằng lệnh **summary** như sau:

```
> summary(model)

Model Info:
 nodes:      15
 arcs:       23
 scores:     NA
 formula:    O ~ b_B_and_Viol_O * VB*Viol + b_C_and_Viol_O * VC*Viol
+ b_T_and_Viol_O * VT*Viol + b_Viol_O * Viol + b_B_and_Lie_O * VB*Lie
+ b_C_and_Lie_O * VC*Lie + b_T_and_Lie_O * VT*Lie + b_Lie_O * Lie +
a_Int1_or_Int2[(Int1+Int2 > 0 ? 1 : 0)]

Estimates:
 model is not estimated!
```

Ta có thể thấy ở phần **formula**, công thức tổng quát của mô hình là:

$$O \sim b_{B_and_Viol_O} * VB * Viol + b_{C_and_Viol_O} * VC * Viol + b_{T_and_Viol_O} * VT * Viol + b_{Viol_O} * Viol + b_{B_and_Lie_O} * VB * Lie + b_{C_and_Lie_O} * VC * Lie + b_{T_and_Lie_O} * VT * Lie + b_{Lie_O} * Lie + a_{Int1_or_Int2}[(Int1+Int2 > 0 ? 1 : 0)]$$

Cơ sở toán học:

$$O_i \sim \alpha[x_{varint}] + \beta_j * x_{ji}$$

Trong đó O_i “outcome”, là biến phụ thuộc (thường được biết đến là response hay dependent variable); x_{varint} là biến varying intercept, x_j là biến độc lập thứ j

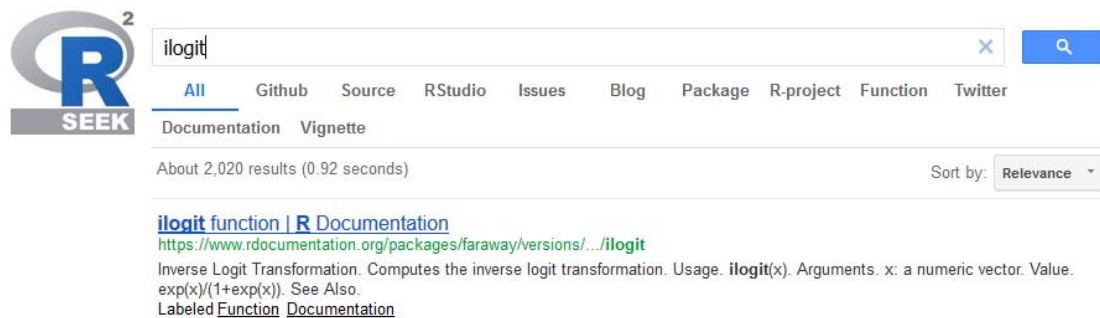
Biến **O** là biến **binomial**, do đó nếu viết dưới dạng thống kê ta có phân phối O có dạng:

$$O \sim \text{binomial}(\text{ilogit}(\theta))$$

Trong đó, hàm nghịch đảo **inverse logit**:

$$\text{ilogit}(\theta) = \text{logit}^{-1}(\alpha_{j[i]} + \beta_i x_i)$$

Người sử dụng có thể tra cứu hàm trong R có thể tìm kiếm nhanh trong <https://rseek.org>, để đi đến trang tài liệu liên quan. Ví dụ, trong trường hợp của hàm `ilogit` vừa sử dụng:



The Stan Code

Stan là ngôn ngữ thống kê xác suất thông dụng được sử dụng để xây dựng mô hình thống kê. Ngôn ngữ Stan được chuyên biệt hóa cho mô hình thống kê Bayesian. Việc sử dụng ngôn ngữ Stan trên R thông thường khá phức tạp, vì vậy, bayesVL hỗ trợ người dùng tạo code Stan tự động cho mô hình như trên. Ta chỉ cần gọi lệnh:

```
model string <- bvl model2Stan(model)
cat(model string)
```

Toàn bộ code stan được tạo ra như sau:

```
functions{
  int numLevels(int[] m) {
    int sorted[num elements(m)];
    int count = 1;
    sorted = sort asc(m);
    for (i in 2:num elements(sorted)) {
      if (sorted[i] != sorted[i-1])
        count = count + 1;
    }
    return(count);
  }
}
data{
  // Define variables in data
  int<lower=1> Nobs; // Number of observations (an integer)
  int<lower=0,upper=1> O[Nobs]; // outcome variable
  int<lower=0,upper=1> Lie[Nobs];
  int<lower=0,upper=1> Viol[Nobs];
```

```

int<lower=0,upper=1> VB[Nobs];
int<lower=0,upper=1> VC[Nobs];
int<lower=0,upper=1> VT[Nobs];
int<lower=0,upper=1> Int1[Nobs];
int<lower=0,upper=1> Int2[Nobs];
}
transformed data{
  // Define transformed data
  vector[Nobs] B and Viol;
  vector[Nobs] C and Viol;
  vector[Nobs] T and Viol;
  vector[Nobs] B and Lie;
  vector[Nobs] C and Lie;
  vector[Nobs] T and Lie;
  int Int1 or Int2[Nobs];
  int NInt1 or Int2;
  for (i in 1:Nobs) {
    Int1 or Int2[i] = (Int1[i]+Int2[i] > 0 ? 1 : 0);
  }
  NInt1 or Int2 = numLevels(Int1 or Int2);

  for (i in 1:Nobs) {
    T and Lie[i] = VT[i]*Lie[i];
  }

  for (i in 1:Nobs) {
    C and Lie[i] = VC[i]*Lie[i];
  }

  for (i in 1:Nobs) {
    B and Lie[i] = VB[i]*Lie[i];
  }

  for (i in 1:Nobs) {
    T and Viol[i] = VT[i]*Viol[i];
  }

  for (i in 1:Nobs) {
    C and Viol[i] = VC[i]*Viol[i];
  }

  for (i in 1:Nobs) {
    B and Viol[i] = VB[i]*Viol[i];
  }
}
parameters{
  // Define parameters to estimate
  real b B and Viol O;
  real b C and Viol O;
  real b T and Viol O;
  real b Viol O;
  real b B and Lie O;
  real b C and Lie O;
  real b T and Lie O;
  real b Lie O;
  real a0 Int1 or Int2;
  real<lower=0> sigma Int1 or Int2;
  vector[NInt1 or Int2] u Int1 or Int2;
}
transformed parameters{
  // Transform parameters
  real theta O[Nobs];
  vector[NInt1 or Int2] a Int1 or Int2;
  // Varying intercepts definition
  for(k in 1:NInt1 or Int2) {
    a Int1 or Int2[k] = a0 Int1 or Int2 + u Int1 or Int2[k];
  }

  for (i in 1:Nobs) {
    theta O[i] = b B and Viol O * B and Viol[i] + b C and Viol O *
C and Viol[i] + b T and Viol O * T and Viol[i] + b Viol O * Viol[i] +
b B and Lie O * B and Lie[i] + b C and Lie O * C and Lie[i] + b T and Lie O
* T and Lie[i] + b Lie O * Lie[i] + a Int1 or Int2[Int1 or Int2[i]+1];
  }
}
model{

```

```

// Priors
b B and Viol O ~ normal( 0, 10 );
b C and Viol O ~ normal( 0, 10 );
b T and Viol O ~ normal( 0, 10 );
b Viol O ~ normal( 0, 10 );
b B and Lie O ~ normal( 0, 10 );
b C and Lie O ~ normal( 0, 10 );
b T and Lie O ~ normal( 0, 10 );
b Lie O ~ normal( 0, 10 );
a0 Int1 or Int2 ~ normal(0,5);
sigma Int1 or Int2 ~ normal(0,5);
u Int1 or Int2 ~ normal(0, sigma Int1 or Int2);

// Likelihoods
O ~ binomial logit(1, theta O);
}
generated quantities {
  // simulate data from the posterior
  int<lower=0,upper=1> yrep O[Nobs];
  // log-likelihood posterior
  vector[Nobs] log lik O;
  int<lower=0,upper=1> yrep Int1 or Int2 1[Nobs];
  int<lower=0,upper=1> yrep Int1 or Int2 2[Nobs];
  for (i in 1:num elements(yrep O)) {
    yrep O[i] = binomial rng(O[i], inv logit(theta O[i]));
  }
  for (i in 1:Nobs) {
    log lik O[i] = binomial logit lpmf(O[i] | 1, theta O[i]);
  }
  for (i in 1:Nobs) {
    yrep Int1 or Int2 1[i] = binomial rng(O[i], inv logit(b B and Viol O
* B and Viol[i] + b C and Viol O * C and Viol[i] + b T and Viol O *
T and Viol[i] + b Viol O * Viol[i] + b B and Lie O * B and Lie[i] +
b C and Lie O * C and Lie[i] + b T and Lie O * T and Lie[i] + b Lie O *
Lie[i] + a Int1 or Int2[1]));
  }
  for (i in 1:Nobs) {
    yrep Int1 or Int2 2[i] = binomial rng(O[i], inv logit(b B and Viol O
* B and Viol[i] + b C and Viol O * C and Viol[i] + b T and Viol O *
T and Viol[i] + b Viol O * Viol[i] + b B and Lie O * B and Lie[i] +
b C and Lie O * C and Lie[i] + b T and Lie O * T and Lie[i] + b Lie O *
Lie[i] + a Int1 or Int2[2]));
  }
}
}

```

Kiểm tra *posteriors* theo điều kiện:

Chương trình **bayesvl** cho phép ta thực hiện dự báo (predict) giá trị outcome của mô hình sau hồi quy. Để thực hiện chèn code này ta cần thêm tham số test khi tạo các node trong mô hình. Ta có thể thấy khi tạo node **Int1_or_Int2**, code bayesvl có dạng:

```

model <- bvl addNode(model, "Int1 or Int2", "trans", fun = "({0} > 0 ? 1
: 0)", out type = "int", lower = 0, test = c(0, 1))

```

Tham số **test = c(0, 1)** cho phép **bayesvl** chèn thêm code tính “fixed predicted outcome” ở **Int1_or_Int2 = 0** và **Int1_or_Int2=1**.

Lệnh này sẽ yêu cầu khi chạy mô phỏng mô hình, phần mềm sẽ tính lại cho ta các tập giá trị outcome **yrep_Int1_or_Int2_1** và **yrep_Int1_or_Int2_2** sau mỗi vòng hồi quy. Như vậy ta sẽ có n tập giá trị outcome mới.

Thao tác này tương đương với việc chèn code stan đánh giá mô hình vào khối lệnh “quantities” của Stan khi chạy như sau:

```
stan code = "
int<lower=0,upper=1> yrep Int1 or Int2 1[Nobs];
int<lower=0,upper=1> yrep Int1 or Int2 2[Nobs];

for (i in 1:Nobs) {
  yrep Int1 or Int2 1[i] = binomial rng(O[i],
    inv_logit(b_B_and_Viol_O * B_and_Viol[i] + b_C_and_Viol_O *
    C_and_Viol[i] + b_T_and_Viol_O * T_and_Viol[i] + b_Viol_O * Viol[i] +
    b_B_and_Lie_O * B_and_Lie[i] + b_C_and_Lie_O * C_and_Lie[i] +
    b_T_and_Lie_O * T_and_Lie[i] + b_Lie_O * Lie[i] + a Int1 or Int2[1]));
}
for (i in 1:Nobs) {
  yrep Int1 or Int2 2[i] = binomial rng(O[i],
    inv_logit(b_B_and_Viol_O * B_and_Viol[i] + b_C_and_Viol_O *
    C_and_Viol[i] + b_T_and_Viol_O * T_and_Viol[i] + b_Viol_O * Viol[i] +
    b_B_and_Lie_O * B_and_Lie[i] + b_C_and_Lie_O * C_and_Lie[i] +
    b_T_and_Lie_O * T_and_Lie[i] + b_Lie_O * Lie[i] + a Int1 or Int2[2]));
}
"

model <- bvl modelFit(model, data1, warmup = 2000, iter = 5000, chains
= 4, cores = 4, ppc = stan code)
```

Ta có thể phân tích các giá trị này ở phần cuối.

Các priors mô hình:

Xem lại các priors trong mô hình:

```
> bvl stanPriors(model)
b B and Viol O ~ normal( 0, 10 );
b C and Viol O ~ normal( 0, 10 );
b T and Viol O ~ normal( 0, 10 );
b Viol O ~ normal( 0, 10 );
b B and Lie O ~ normal( 0, 10 );
b C and Lie O ~ normal( 0, 10 );
b T and Lie O ~ normal( 0, 10 );
b Lie O ~ normal( 0, 10 );
a0 Int1 or Int2 ~ normal(0,5);
sigma Int1 or Int2 ~ normal(0,5);
u Int1 or Int2 ~ normal(0, sigma Int1 or Int2);
```

Lưu ý phần lớn các giá trị *priors* sử dụng trong mô hình (và trong thực tế) là giá trị mặc định “default”. Để thay đổi *prior*, ta có thể đặt thêm tham số *priors* khi viết hàm tạo arc quan hệ **bayesvl**, ví dụ:

```
> model <- bvl addArc(model, "Int1 or Int2", "O", "varint", priors =
c("a0 ~ normal(0,5)", "sigma ~ normal(0,5)"))
```

Bên cạnh việc tạo mô hình thống kê R/Stan dựa trên sơ đồ logic hay kiểm tra các *priors* trong mô hình, **bayesvl** còn hỗ trợ xem lại các tham số đã dùng trong mô hình

thông qua hàm `bvl_stanParams(model)` hay tổng hợp và mô phỏng các mẫu từ mô hình với hàm `bvl_modelFit(model)`. Thông tin thêm về các hàm này có thể tra cứu trên R bằng lệnh:

```
> help("bayesvl stan")
```

Đánh giá mô hình bằng **bnlearn**:

Chương trình R **bnlearn** thường được sử dụng để đo lường quan hệ giữa các biến trong lưới **model** thông qua xác suất tương tác của từng **arc**. Khi chạy đánh giá mô hình thành công, giá trị **strength** sẽ được hiển thị. Giá trị này chính là giá trị p -value trong các áp dụng *frequentist* truyền thống.

```
> bvl_bnScore(model, data1)
[1] -3158.136

> bvl_bnStrength(model, data1)
```

	from	to	strength
1	Lie	B_and_Lie	1.282892e-19
2	Lie	C_and_Lie	6.639677e-36
3	Lie	T_and_Lie	1.713908e-15
4	Lie	O	1.000000e+00
5	Viol	B_and_Viol	1.282892e-19
6	Viol	C_and_Viol	6.639677e-36
7	Viol	T_and_Viol	1.713908e-15
8	Viol	O	1.000000e+00
9	VB	B_and_Viol	7.681205e-15
10	VB	B_and_Lie	8.533048e-17
11	VC	C_and_Viol	7.681205e-15
12	VC	C_and_Lie	8.533048e-17
13	VT	T_and_Viol	7.681205e-15
14	VT	T_and_Lie	8.533048e-17
15	Int1	Int1_or_Int2	2.315429e-54
16	Int2	Int1_or_Int2	1.520310e-46
17	B_and_Viol	O	1.000000e+00
18	C_and_Viol	O	1.000000e+00
19	T_and_Viol	O	1.000000e+00
20	B_and_Lie	O	1.000000e+00
21	C_and_Lie	O	1.000000e+00
22	T_and_Lie	O	1.000000e+00
23	Int1_or_Int2	O	1.000000e+00

Có thể thấy, mô hình hiện tại có 23 **arc** nhưng các giá trị p -values nổi với kết cục được thể hiện đều không có ý nghĩa thống kê (do giả thiết xuất phát của **bnlearn** nhằm phục vụ “independence test”).

*Vì vậy, không thể sử dụng **bnlearn** để tìm kiếm kết quả cho mô hình này.*

Thực hiện mô phỏng MCMC

Markov Chain Monte Carlo (thường được viết tắt là MCMC) ngày nay được sử dụng phổ biến để mô phỏng phân phối xác suất của các *posteriors*. Gọi lệnh thực hiện mô phỏng MCMC trên R từ trong chương trình **baeysvl**:

```
> model <- bvl modelFit(model, data1, warmup = 2000, iter = 5000,
chains = 4, cores = 4)
```

Lệnh này cho biết sẽ có 4 xích Markov được mô phỏng từ dữ liệu thực nghiệm. Mỗi xích có độ dài 5000 bước, trong đó có 2000 bước dò dẫm, nghĩa là không được tính vào **n_eff** mà chỉ để tạo ra tính ổn định của xích.

Kết quả thu được từ mô phỏng MCMC

```
> summary(model)
Model Info:
  nodes:      15
  arcs:       23
  scores:     NA
  formula:    O ~ b_B_and_Viol_O * VB*Viol + b_C_and_Viol_O * VC*Viol +
b_T_and_Viol_O * VT*Viol + b_Viol_O * Viol + b_B_and_Lie_O * VB*Lie +
b_C_and_Lie_O * VC*Lie + b_T_and_Lie_O * VT*Lie + b_Lie_O * Lie +
a_Int1_or_Int2[(Int1+Int2 > 0 ? 1 : 0)]

Estimates:
Inference for Stan model: d4bbc50738c6da1b2c8e7cfedb604d80.
4 chains, each with iter=5000; warmup=2000; thin=1;
post-warmup draws per chain=3000, total post-warmup draws=12000.
```

	mean	se_mean	sd	2.5%	25%	50%	75%	97.5%	n_eff	Rhat
b_B_and_Viol_O	2.55	0.05	1.46	0.13	1.50	2.41	3.42	5.73	915	1.01
b_C_and_Viol_O	-0.28	0.01	0.61	-1.46	-0.68	-0.31	0.13	0.93	6689	1.00
b_T_and_Viol_O	-0.96	0.01	1.09	-3.21	-1.65	-0.91	-0.26	1.14	6820	1.00
b_Viol_O	-0.62	0.01	0.42	-1.43	-0.90	-0.62	-0.35	0.23	5892	1.00
b_B_and_Lie_O	0.70	0.02	1.44	-1.78	-0.28	0.56	1.52	4.03	6546	1.00
b_C_and_Lie_O	1.47	0.02	0.68	0.21	0.97	1.45	1.94	2.86	1676	1.01
b_T_and_Lie_O	2.23	0.02	1.59	-0.41	1.10	2.06	3.16	5.85	4523	1.00
b_Lie_O	-1.05	0.01	0.37	-1.77	-1.30	-1.05	-0.81	-0.32	3984	1.00
a_Int1_or_Int2[1]	1.20	0.00	0.21	0.78	1.05	1.20	1.33	1.62	7767	1.00
a_Int1_or_Int2[2]	1.35	0.00	0.19	0.99	1.23	1.35	1.48	1.73	3512	1.00
a0_Int1_or_Int2	1.18	0.04	1.34	-1.91	0.87	1.25	1.57	3.83	1353	1.00
sigma_Int1_or_Int2	1.49	0.04	1.82	0.04	0.28	0.78	1.98	6.67	1759	1.00

Kết quả trên ta có thể thấy mô hình hội tụ tốt, điều này được thể hiện qua hai cột tiêu chuẩn đánh giá của MCMC là **n_eff** (*effective sample size*) và **Rhat**. Các giá trị của **n_eff** cho biết mô phỏng thô số lượng các mẫu (*samples*) độc lập mà người dùng có thể có^[12]. Trong khi đó, các giá trị của **Rhat** thể hiện mô phỏng phức tạp hơn của hội tụ của các xích Markov tới phân phối mục tiêu.

Thông thường, **Rhat** xấp xỉ 1 cho thấy các chuỗi đều có phân phối giống nhau, trong khi lớn hơn 1.1 là mô hình có chưa bắt đầu hội tụ, vì vậy không các mẫu không đáng tin cậy. Trong khi đó, **n_eff** có các giá trị trên 1000 mẫu là dấu hiệu tốt cho suy luận thống kê. Trong mô hình hiện tại, kết quả khá tốt khi hầu hết các biến đều cho thấy **Rhat** có giá trị ở khoảng 1 và **n_eff** đều trên 2000.

Sản xuất hình ảnh và kiểm tra kết quả

Bayesvl hỗ trợ sản xuất đồ họa để thực hiện kiểm tra mô phỏng MCMC và kết quả thông qua hàm có dạng `bvl_plotX`. Trong đó, `X` là các thông tin kết quả chúng ta cần sản xuất đồ họa như “Gelman shrink factors” ([Gelman](#)), các tham số sau ([Params](#)), hay các cặp tham số ([Pairs](#)). Có thể xem thêm các hàm trên `bvl` bằng lệnh:

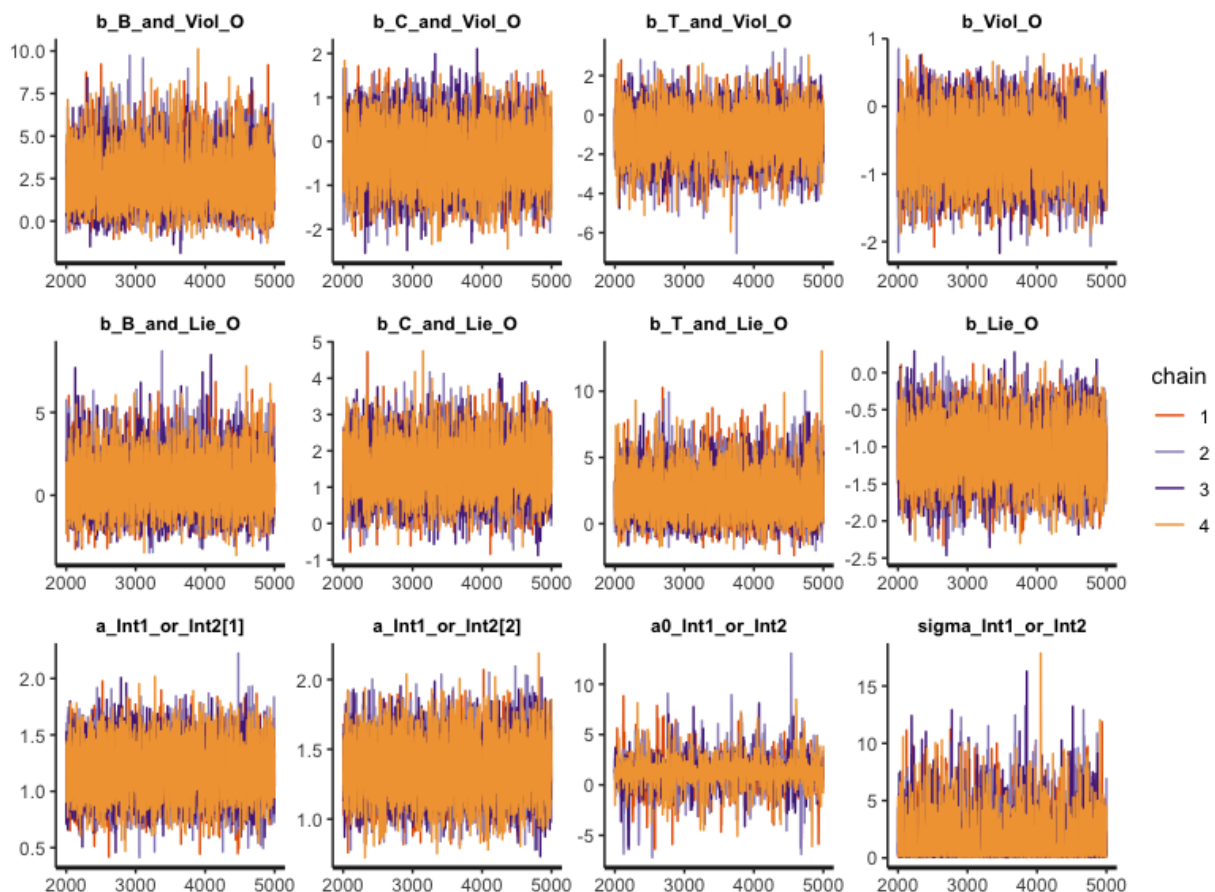
```
> help("bayesvl plots")
```

Kiểm tra đồ họa xích Markov:

Người sử dụng có thể sử dụng `bvl_trace(model)` để tạo đồ họa các xích MCMC:

```
> bvl_trace(model)
```

Các xích thu được trình bày trong hình 3:



Hình 3

Mỗi xích trong hình 3 đều được chia thành 4 xích thành phần, với 5000 vòng lặp cho mỗi xích (*iterations*). Về tổng thể, các xích không có chuỗi nào bất thường (*divergent chains*), tức là bị phân ly và cho thấy dấu hiệu mạnh của hiện tượng tự tương quan

(phản tính chất Markov của phân phối). Nếu phân tách một xích thành từng phần và so sánh thì ta sẽ có các phần khá tương đồng nhau về mặt hình ảnh.

Kiểm tra Gelman shrink factors:

Hệ số co Gelman (“Gelman shrink factor”) hay còn được gọi là hệ số suy giảm quy mô tiềm năng (*potential scale reduction factor*) thường được sử dụng trong chẩn đoán hội tụ. Chẩn đoán hội tụ này rất cần thiết để đưa ra các kết luận dựa trên phân phối sau, mô tả chính xác mô phỏng tham số và các yếu tố không chắc chắn.

Chẩn đoán hội tụ này chính là các giá trị **Rhat** được thể hiện ở phần kết quả trên. Chuẩn đoán này do Gelman & Rubin ^[25] và sau đó là Brooks & Gelman ^[26] phát triển.

Cơ sở toán học:

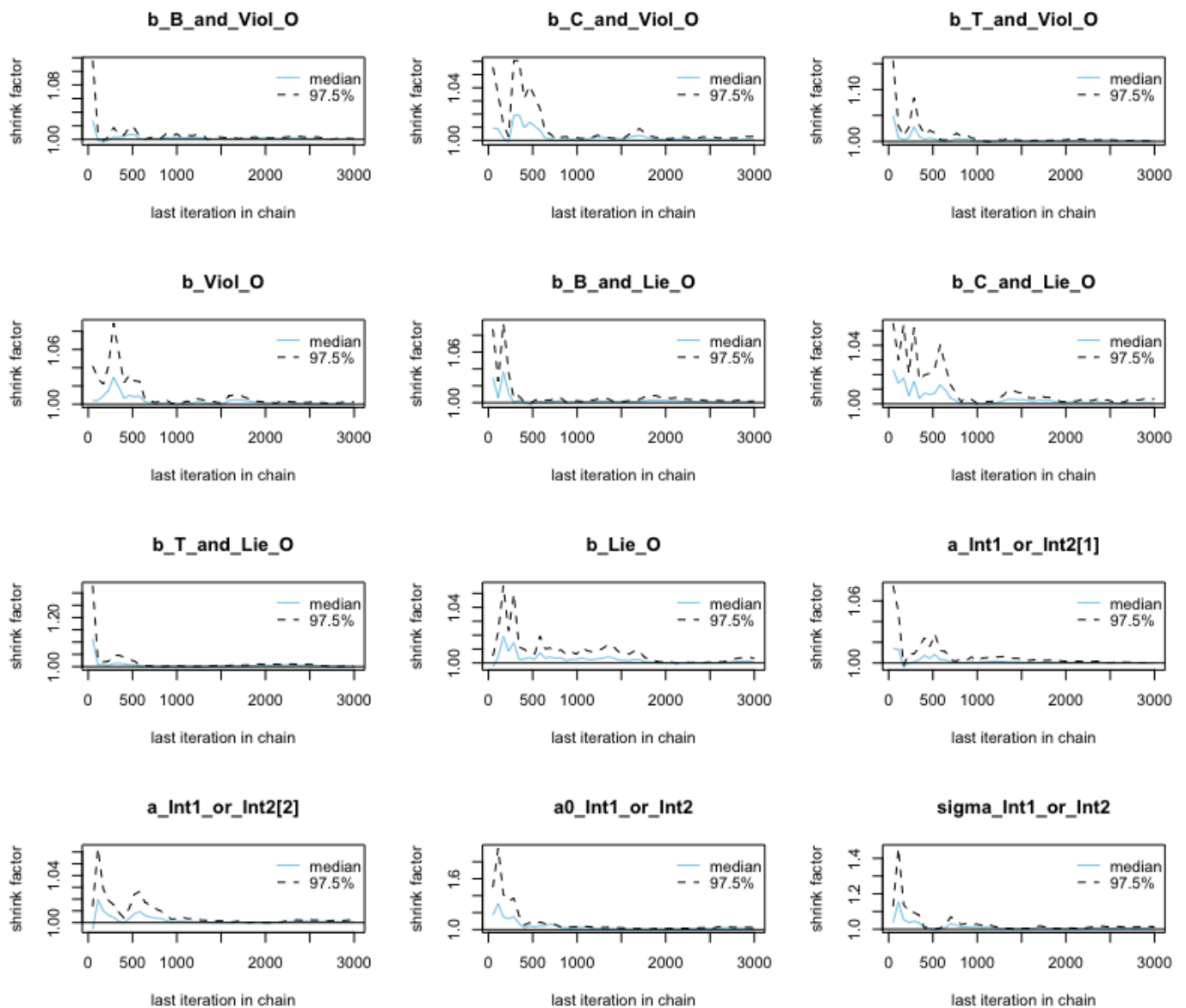
$$\hat{R} = \sqrt{\frac{\widehat{\text{Var}}(\theta)}{W}}$$

Trong đó \hat{R} hay Rhat là hệ số suy giảm quy mô tiềm năng, $\widehat{\text{Var}}(\theta)$ là phương sai được ước tính (Estimated variance) và W là phương sai trong chuỗi (Within chain variance) ^[27].

Chương trình **bayesvl** sẽ kiểm tra “Gelman shrink factor” trong R bằng lệnh như trong ví dụ sau:

```
> bvl plotGelmans(model, NULL, 4, 3)
```

Hình ảnh:



Hình 3a

Kết quả hiện nay cho thấy giá trị *mean* của hệ số suy giảm quy mô tiềm năng với 97.5%. Bên cạnh đó, chúng ta cũng có hệ số suy giảm quy mô tiềm năng đa biến được Gelman và Brooks gợi ý. Hình ảnh cho thấy hệ số suy giảm hội tụ về 1.0 khá nhanh, đáp ứng rất tốt tiêu chuẩn kỹ thuật của mô phỏng MCMC.

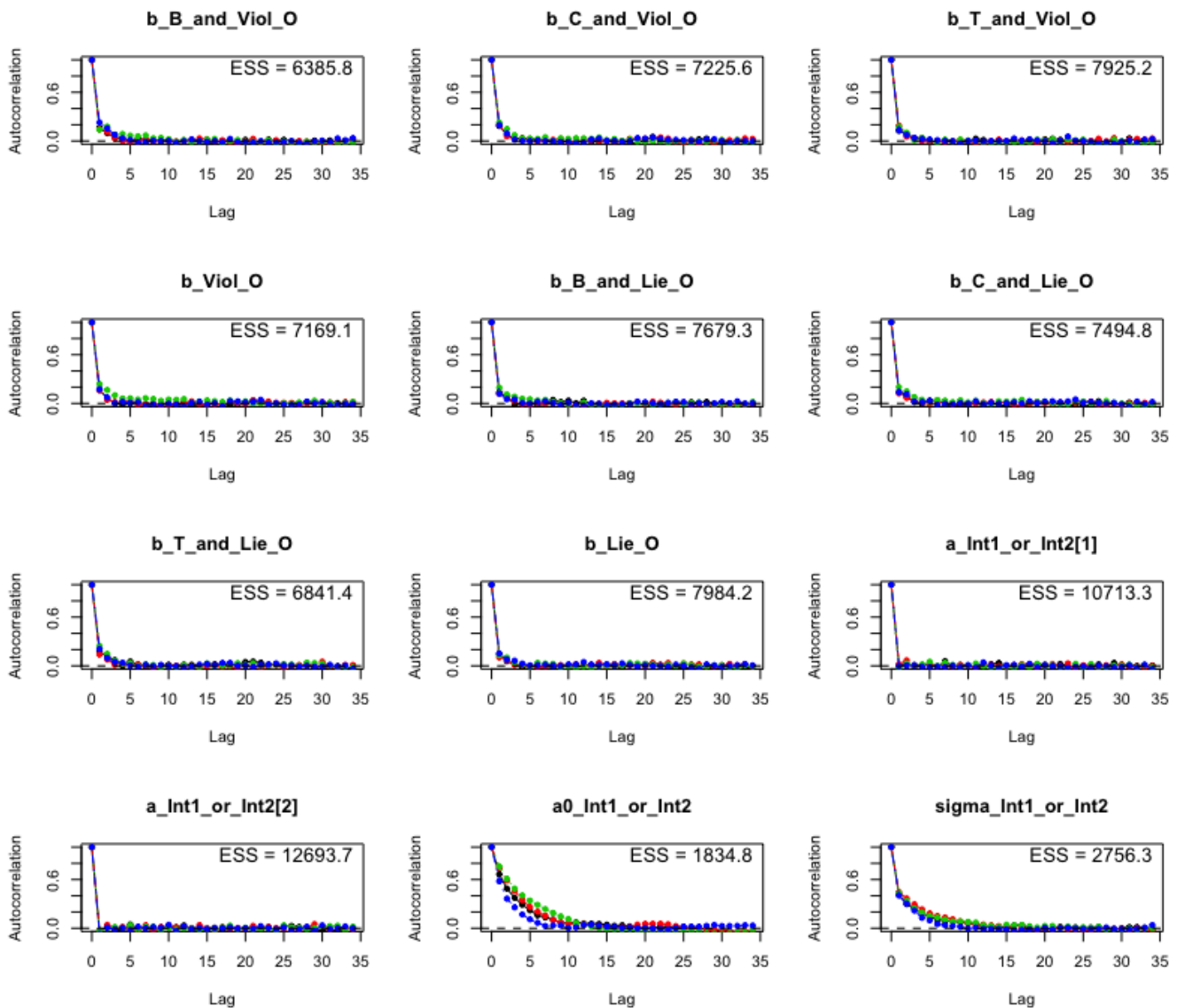
Autocorrelation của từng hệ số:

Thuật toán MCMC sản xuất ra các mẫu tự tương quan (autocorrelation) với nhau chứ không độc lập. Vì vậy, việc bị mixing chậm do tỉ lệ chấp thuận quá cao hoặc có thấp có thể dẫn đến các quá trình không đảm bảo tính chất Markov. Việc kiểm tra nhằm đảm bảo sau một số bước hữu hạn, hiện tượng autocorrelation sẽ bị triệt tiêu (về 0).

Lệnh của **bayesvl** cho phép vẽ hàm hệ số tự tương quan (ACF) để kiểm tra như sau:

```
> bvl_plotAcfs(model, NULL, 4, 3)
```

Hình 3b cung cấp các ACF:



Hình 3b

Hình 3b cho thấy số lượng *effective sample size* (ESS) đều trên 1000, hầu hết nhanh chóng hội tụ trước lag 3, đảm bảo hiệu quả tính toán nhờ đáp ứng tính chất Markov của các xích.

Tham số tự tương quan (*autocorrelation parameter*) cho mức trễ L ($\text{lag} = L$) được tính theo công thức sau đây:

$$ACF_L = \left(\frac{T}{T-L} \right) \frac{\sum_{t=1}^{T-L} (x_t - \bar{x})(x_{t+L} - \bar{x})}{\sum_{t=1}^T (x_t - \bar{x})^2}$$

Trong đó, x_t là giá trị lấy mẫu của của x tại vòng lặp t , T là tổng số giá trị được lấy mẫu, \bar{x} là giá trị trung bình của toàn bộ giá trị được lấy mẫu, và L là độ trễ tính bằng số bước.

Đánh giá tổng quan các hệ số hồi quy:

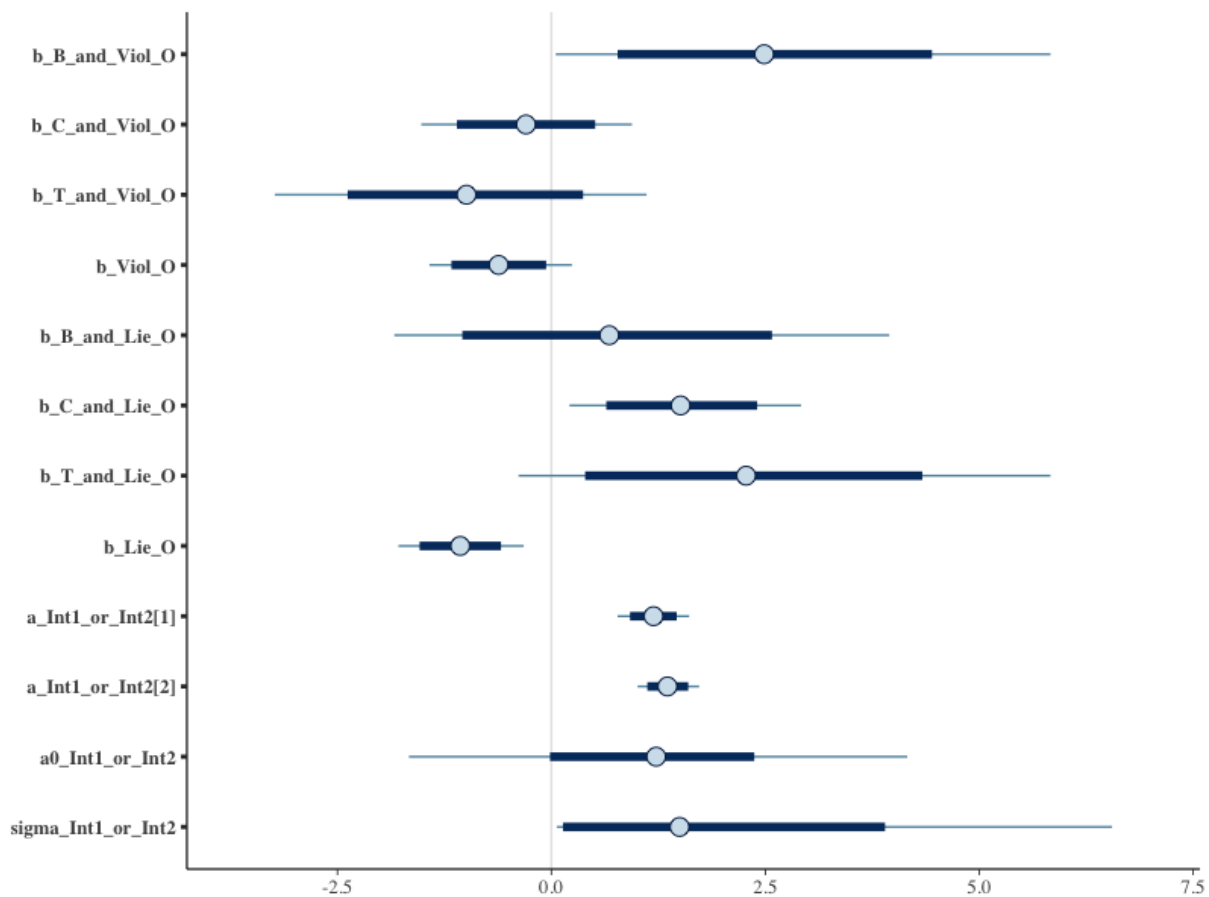
Ta có thể đánh giá mức độ phù hợp của mô hình đối với dữ liệu thông qua predictive posterior distributions. Mật độ posterior của tham số được tính bằng phân bố prior của tham số nhân với likelihood function:

$$p(\theta|\text{Data}) \propto p(\text{Data}|\theta)p(\theta)$$

Trong R ta sử dụng câu lệnh sau của **bayesvl**:

```
> bvl plotIntervals(model)
```

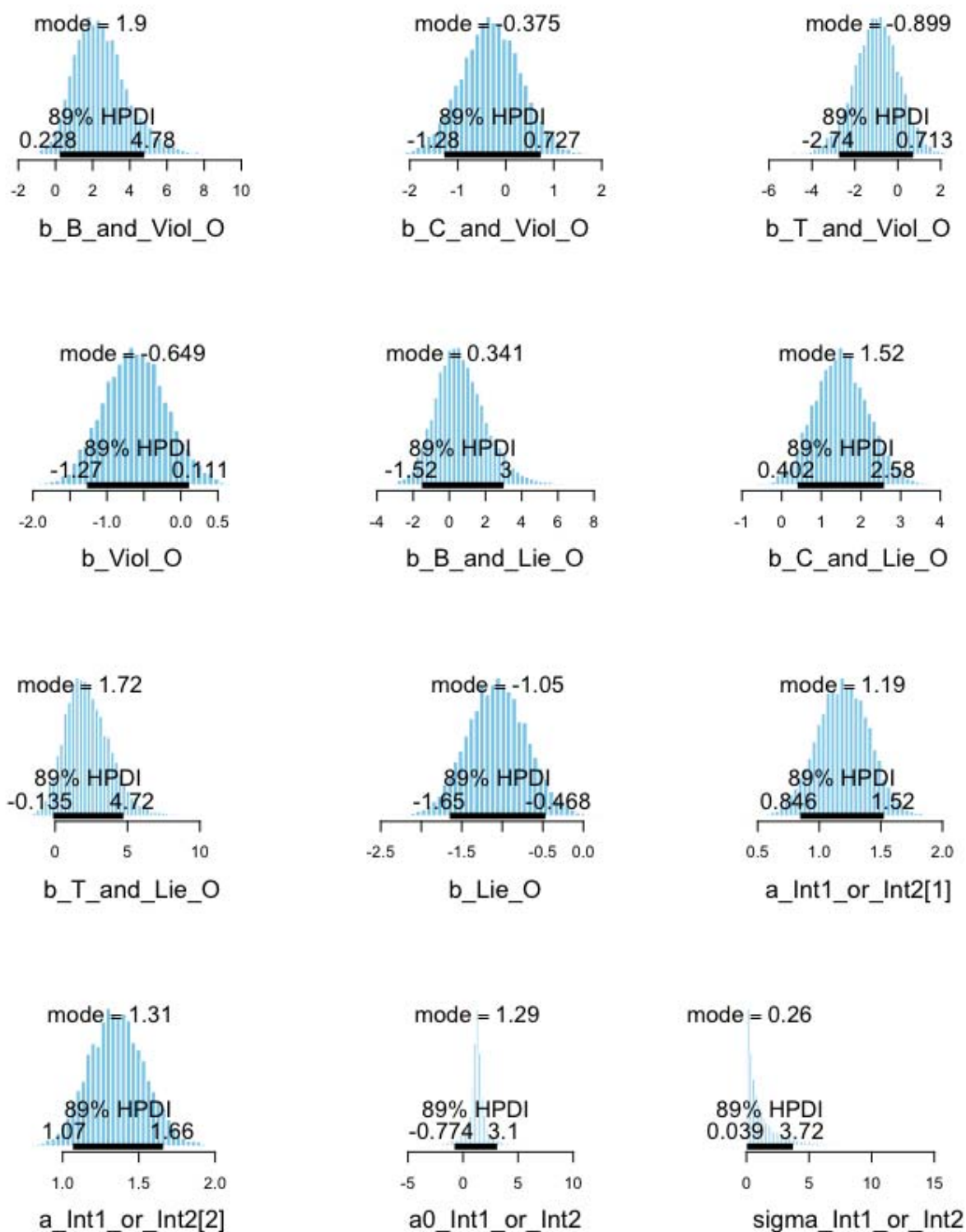
Các hệ số hồi quy so sánh qua đồ họa:



Hình 4

Phân phối của các hệ số:

```
> bvl plotParams(model, 3, 3)
```



Hình 5

Phân phối của các hệ số đều thỏa mãn các yếu tố kỹ thuật với HPDI (*Highest Posterior Distribution Intervals*) ở mức 89%.

Để thay đổi khoảng credibility của phân phối, có thể sử dụng lệnh đầy đủ của **bayesvl** nhưng thay đổi tham số của **credMass** ví dụ 95% như sau:

```
bvl_plotParams <- function(dag, row = 2, col = 2,
  credMass = 0.95, params = NULL)
```

NULL có nghĩa là tiến hành việc in cho tất cả các tham số. Hay có thể dùng lệnh viết tắt như trên, nhưng với trật tự xác định (không được đổi vị trí các đối số của hàm):

```
> bvl_plotParams(model, 3, 3, .95)
```

Còn với trường hợp người sử dụng cần đích danh một hay vài tham số, thì sẽ sử dụng lệnh viết gọn như sau:

```
>bvl_plotParams(model, 4, 3, 0.95,
  c("b_B", "a_O"))
```

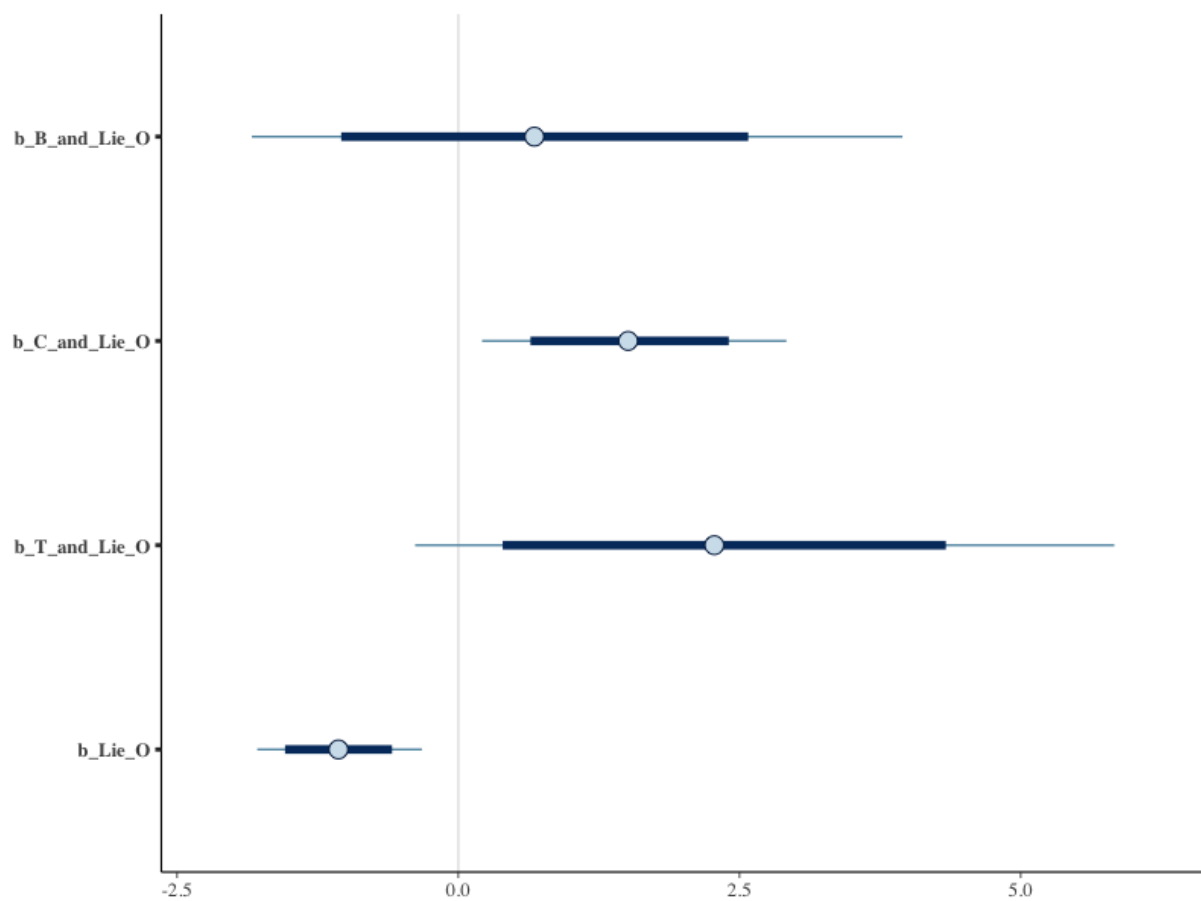
Chúng ta sẽ so sánh và nhận định kĩ hơn từng nhóm hệ số theo các yếu tố nói dối, bạo lực, và tác động người hoặc thần tiên ở dưới.

Đánh giá riêng các hệ số nhóm yếu tố nói dối:

Code R có dạng:

```
> bvl_plotIntervals(model, c("b_B_and_Lie_O", "b_C_and_Lie_O",
  "b_T_and_Lie_O", "b_Lie_O"))
```

Hình ảnh các hệ số thu được với các yếu tố liên quan tới “Lie”:

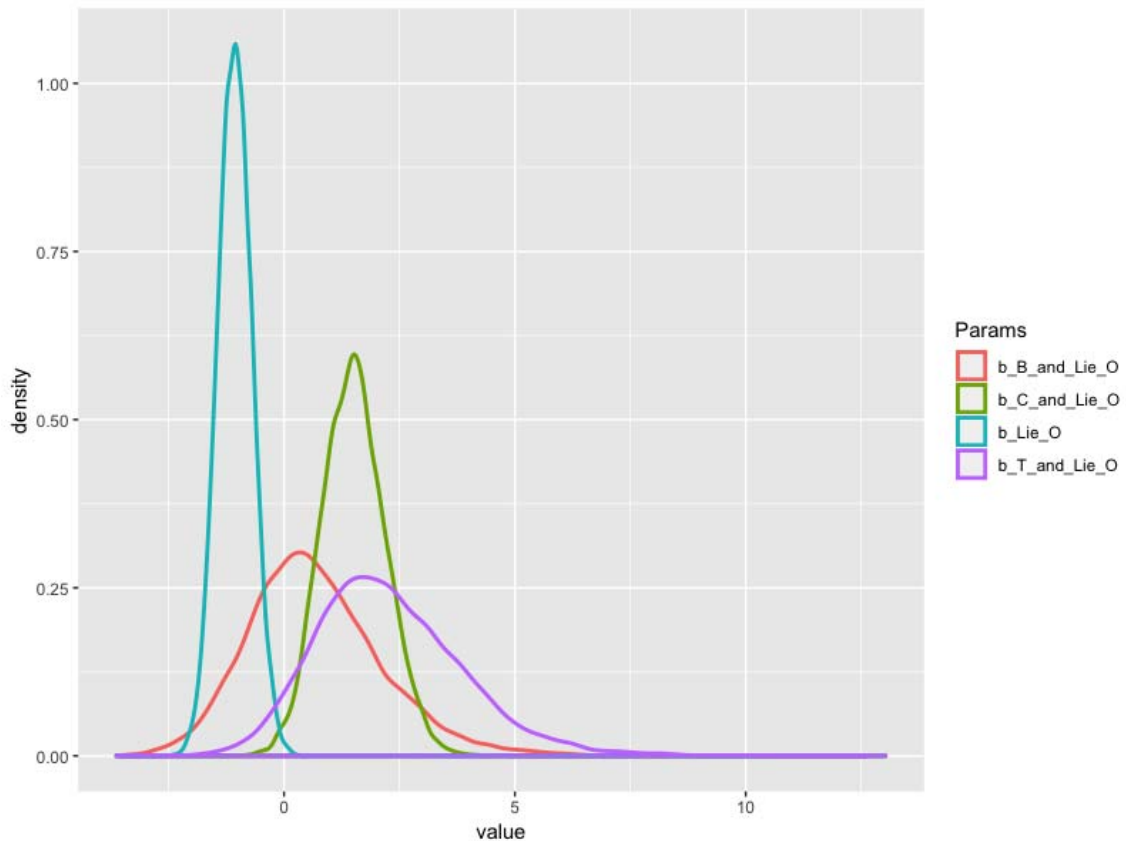


Hình 6

Tương quan so sánh các mật độ phân phối từ tập posteriors sau mô phỏng:

```
> bvl plotDensity(model, c("b B and Lie O", "b C and Lie O",  
"b T and Lie O", "b Lie O"))
```

Hình ảnh Hình 7



Hình 7 - Density

Kết quả sơ bộ ở Hình 6 và 7, ta có thể thấy nói chung, nói dối không mang lại kết cục tốt đẹp cho nhân vật chính. Hệ số **b_Lie_O** âm thể hiện nói dối đi ngược với kết cục có hậu của nhân vật chính. Phân phối này hẹp, có khoảng tin cậy tốt.

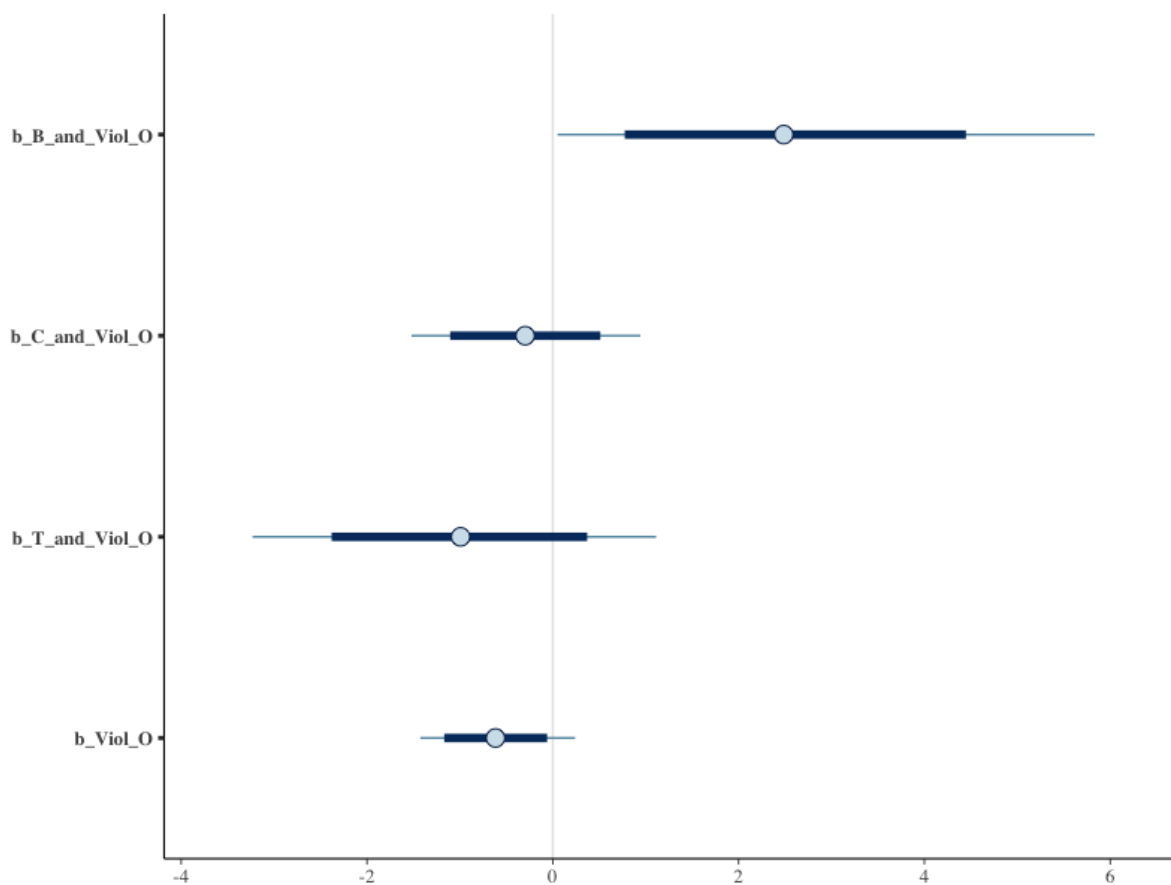
Tuy nhiên các nhóm hệ số kết hợp khi nhân vật chính nói dối nhưng thuộc một trong tam giáo đều dương. Khi có ảnh hưởng của đạo giáo, nhân vật chính dường như kết thúc có hậu hơn, cho dù nói dối.

Trong 3 đạo, đạo phật có ảnh hưởng khuyến khích nói dối ít nhất, hệ số **b_B_and_Lie_O** nhỏ hơn nhiều so với hai đạo còn lại.

Đánh giá riêng các hệ số yếu tố bạo lực:

```
> bvl_plotIntervals(model, c("b_B_and_Viol_O",
  "b_C_and_Viol_O", "b_T_and_Viol_O", "b_Viol_O"))
```

Hình ảnh thu được trong hai Hình 8.

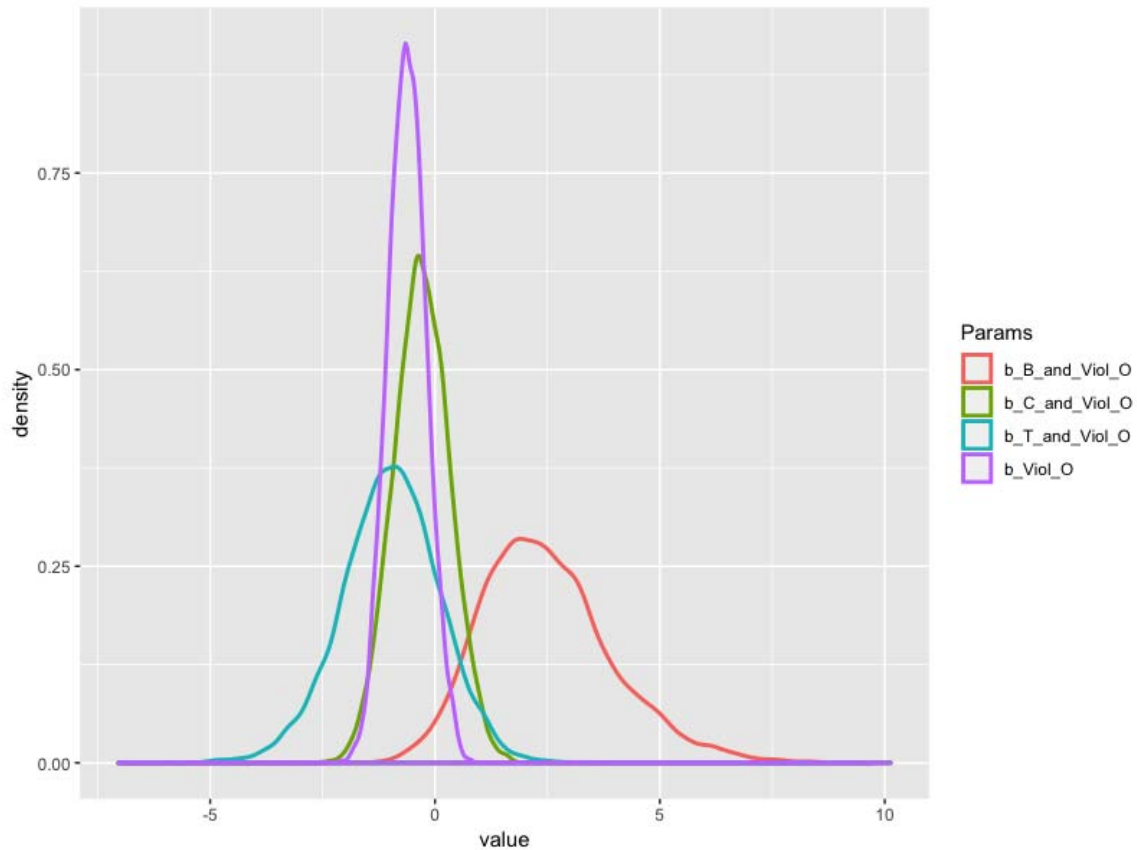


Hình 8

Code R để sản xuất hình ảnh về mật độ `bvl_plotDensity`:

```
> bvl_plotDensity(model, c("b B and Viol O", "b C and Viol O",  
"b T and Viol O", "b Viol O"))
```

Đồ họa tương ứng thu được:



Hình 9 - Density (Viol)

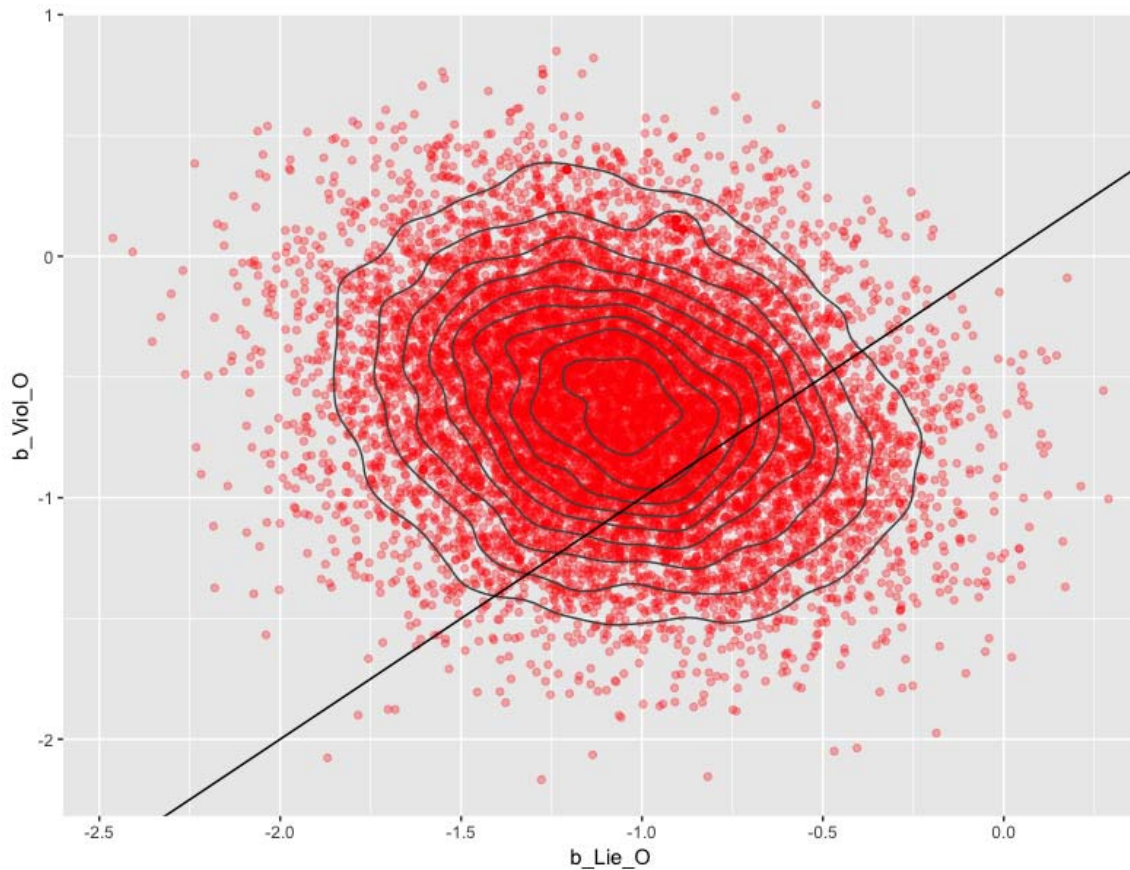
Kết quả sơ bộ ta có thể thấy nói chung, bạo lực không được khuyến khích trong các câu truyện, bạo lực thường không mang lại kết cục tốt đẹp cho nhân vật chính. Hệ số `b_Viol_O` âm thể hiện bạo lực đi ngược với kết cục có hậu của nhân vật chính. Phân phối này hẹp, có khoảng tin cậy tốt.

Khi kết hợp với các yếu tố tam giáo cho thấy các hệ số của đạo khổng và lão vũn âm và không cách xa quá so với hệ số bạo lực chung, tuy nhiên khi kết hợp với yếu tố phật giáo, cho thấy dù nhân vật có hành vi bạo lực nhưng nếu có yếu tố phật giáo, kết của vẫn tốt lên (hệ số dương).

Nếu so sánh tương quan giữa bạo lực và nói dối:

```
> bvl plotDensity2d(model, "b Lie O", "b Viol O")
```

Nhằm thu được đồ họa các cặp tương quan mật độ, trong Hình 10:



Hình 10

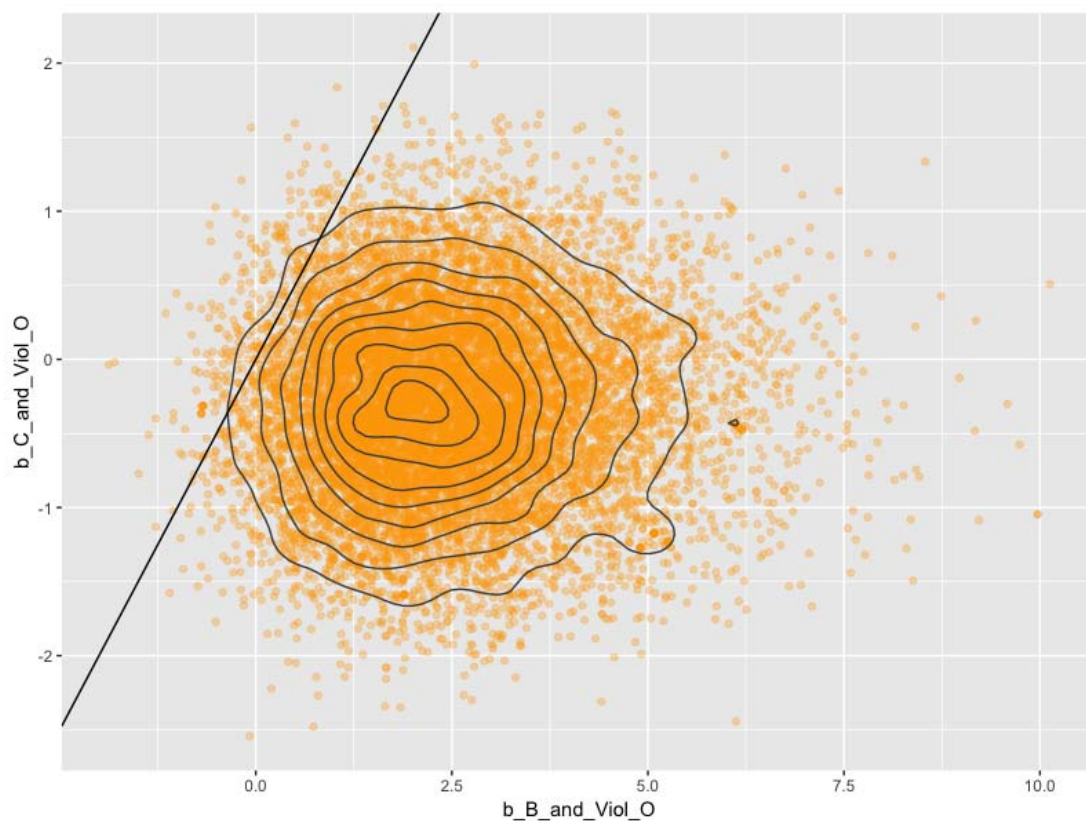
Ta thấy cả 2 hệ số đều âm, có xu hướng ngược với kết quả có hậu của nhân vật chính. Hay có thể nói khi có hành vi nói dối hoặc bạo lực, nhân vật chính có xu hướng không có kết cục tốt đẹp.

Nếu so tương quan cặp hệ số này, ở khoảng tập trung nhiều nhất, hệ số bạo lực yếu hơn nhiều so với nói dối, ảnh hưởng của bạo lực đến kết cục câu truyện không nhiều, trong khi đó yếu tố nói dối có mức độ mạnh hơn nhiều.

So sánh các yếu tố Tam giáo với nhau khi kết hợp với yếu tố bạo lực:

```
> bvl_plotDensity2d(model, "b_B_and_Viol_O", "b_C_and_Viol_O",
  color_scheme = "orange")
```

Hình ảnh trong Hình 11:

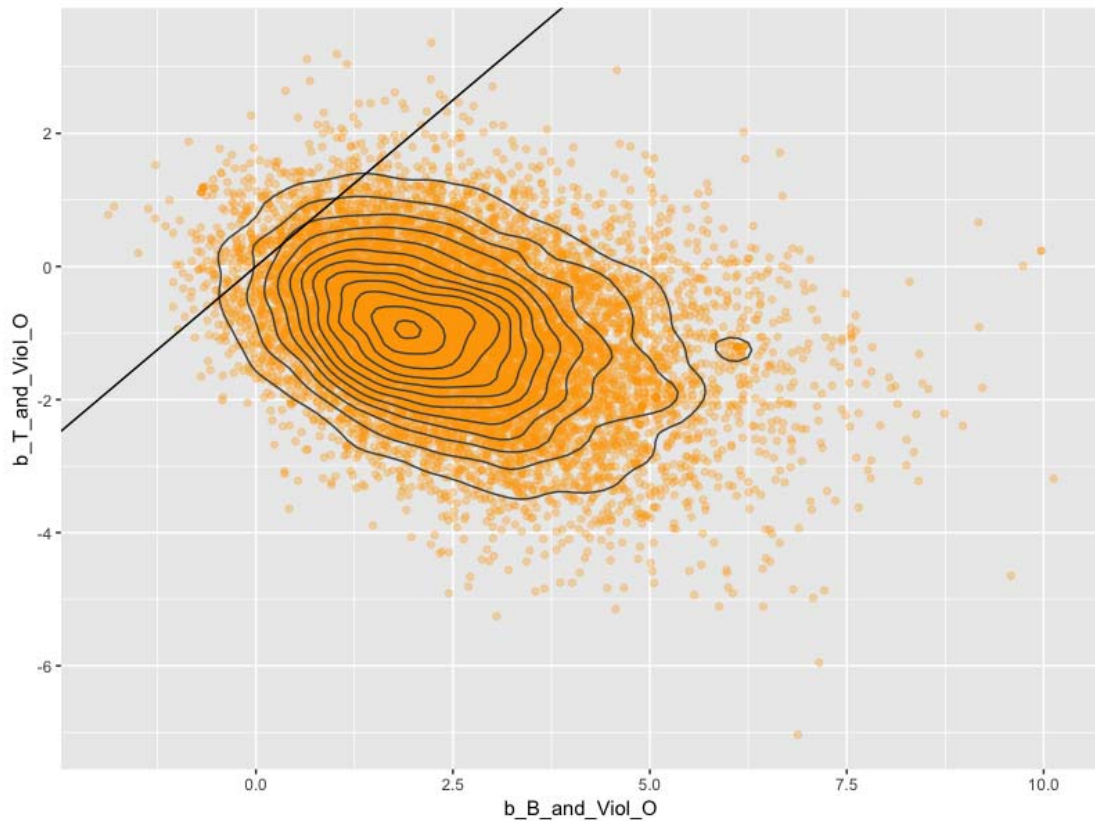


Hình 11

Yếu tố phật giáo và khổng giáo có dấu trái ngược, trong khi đạo phật cho thấy xu hướng chấp nhận yếu tố bạo lực (hệ số dương) thì đạo khổng có hệ số âm. Tuy nhiên ảnh hưởng phật giáo mạnh hơn nhiều.

```
> bvl plotDensity2d(model, "b B and Viol O", "b T and Viol O",
  color scheme = "orange")
```

Hình trong Hình 12:



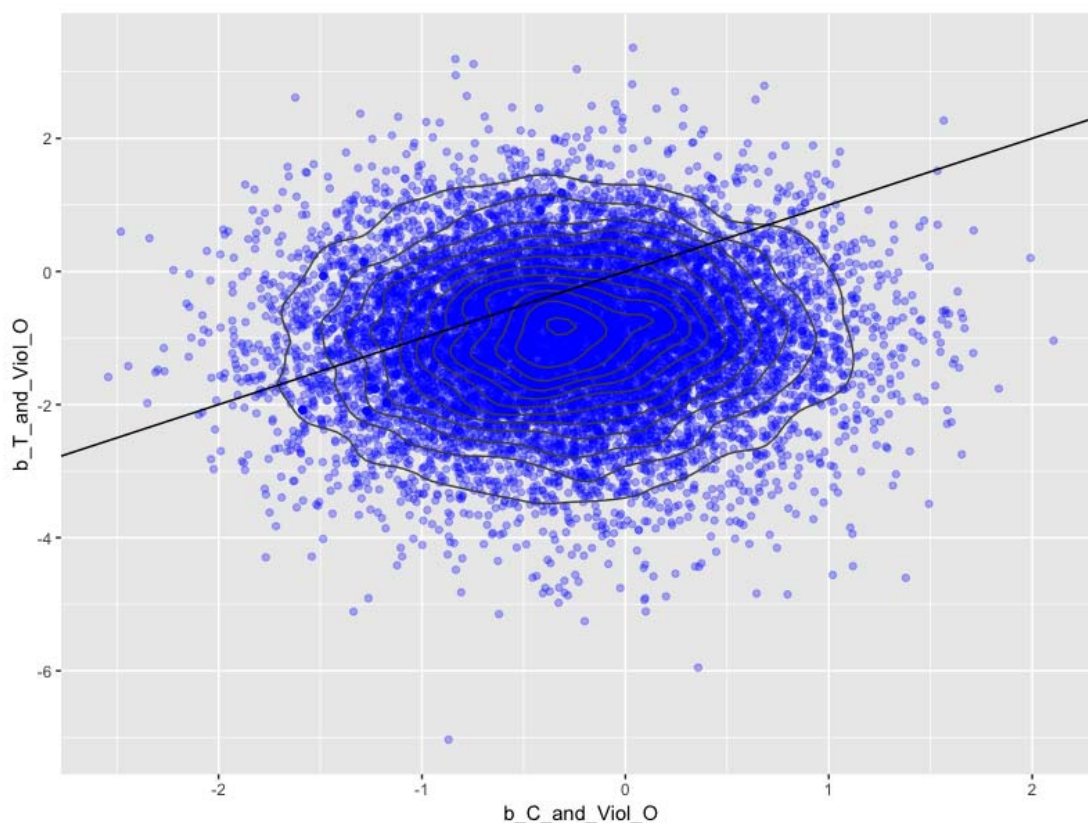
Hình 12

Cũng tương tự ở trên, yếu tố phật giáo và lão giáo có dấu trái ngược, trong khi phật giáo có xu hướng ủng hộ yếu tố bạo lực (hệ số dương) thì lão giáo có hệ số âm. Tuy nhiên ảnh hưởng phật giáo mạnh hơn nhiều.

Tiếp theo là cặp hệ số khổng và lão:

```
> bvl_plotDensity2d(model, "b_C_and_Viol_O", "b_T_and_Viol_O",
  color scheme = "blue")
```

Hình 13 so sánh tương quan khổng-lão:



Hình 13

Yếu tố khổng giáo và lão giáo khá tương đồng, cả 2 đều âm nhẹ và tập trung ở cùng khoảng giá trị. Có thể nói lão giáo và khổng giáo không hỗ trợ bạo lực mạnh.

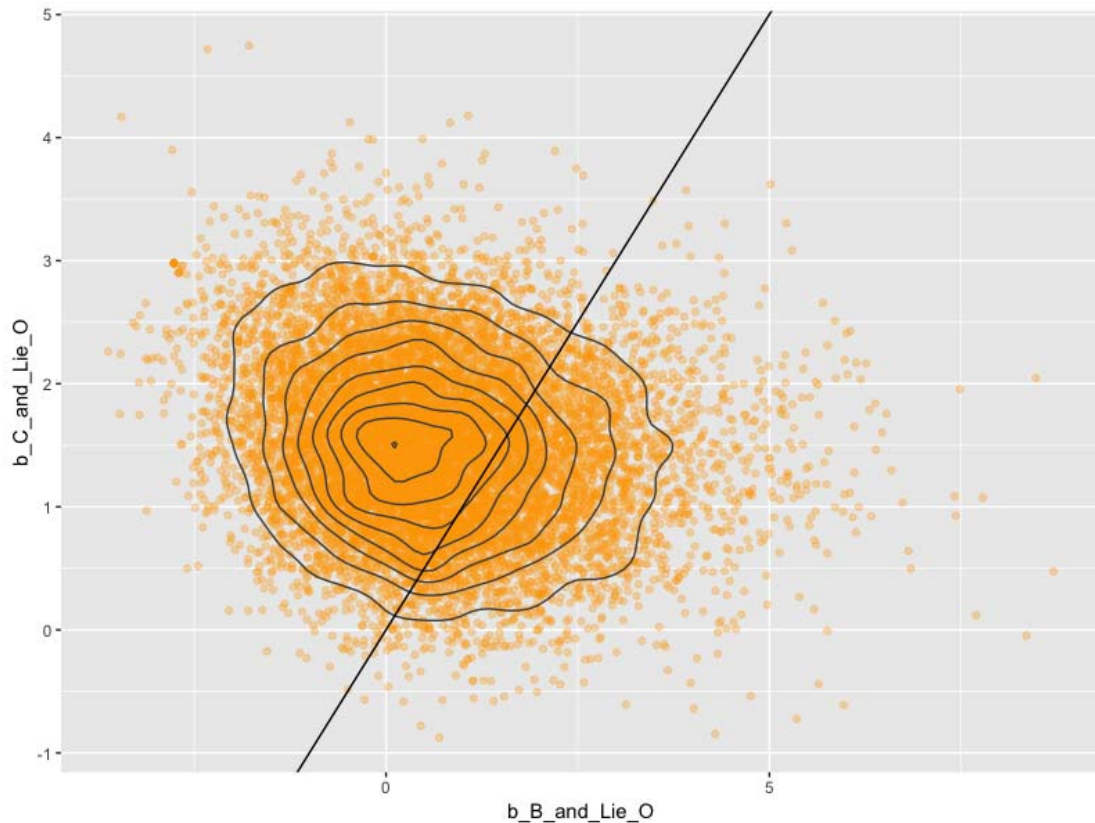
Kết quả ở Hình 11 và 12 cho thấy phật giáo có xu hướng chấp nhận bạo lực rất lý thú khi một trong các giá trị nền tảng của đạo phật là sự đề cao nhân sinh, cấm sát sinh. Đối với kết quả lão giáo và khổng giáo không có xu hướng chấp nhận bạo lực mạnh như trong Hình 13, ta có thể tạm lý giải các nhân vật mang giá trị đạo khổng thường là những người tiến thân theo con đường học hành. Trong khi đó, Lão giáo coi trọng vô vi, không màng thế sự, nên tránh xa xô bồ của bạo lực là điều có thể giải thích được.

So sánh các yếu tố tam giáo kết hợp với nói dối:

Đầu tiên là cặp phật giáo và khổng giáo:

```
> bvl plotDensity2d(model, "b B and Lie O", "b C and Lie O",
color scheme = "orange")
```

Ta thu được Hình 14:



Hình 14

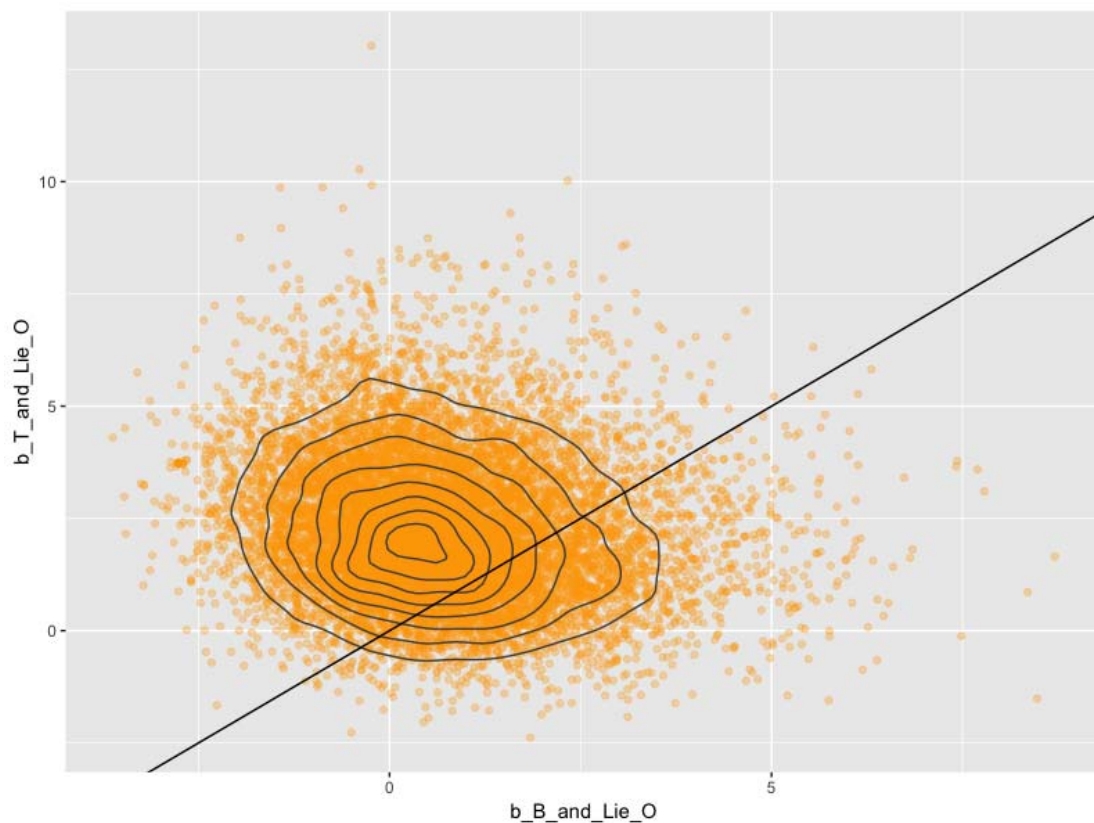
Yếu tố phật giáo và khổng giáo có cùng dấu ở khoảng tập trung mean, tuy nhiên hệ số của phật giáo rất nhỏ, khoảng phân phối gần như nằm giữa âm và dương. Có thể thấy phật giáo không ủng hộ yếu tố nói dối rõ rệt.

Ngược lại yếu tố khổng kết hợp với nói dối có hệ số dương mạnh, toàn bộ khoảng tin cậy 95% đều nằm ở hệ số dương.

Tiếp đó, với cặp phật giáo và lão giáo:

```
> bvl_plotDensity2d(model, "b_B_and_Lie_O", "b_T_and_Lie_O",
  color_scheme = "orange")
```

Ta có thể kiểm tra quan hệ hình học trong Hình 15 dưới đây:



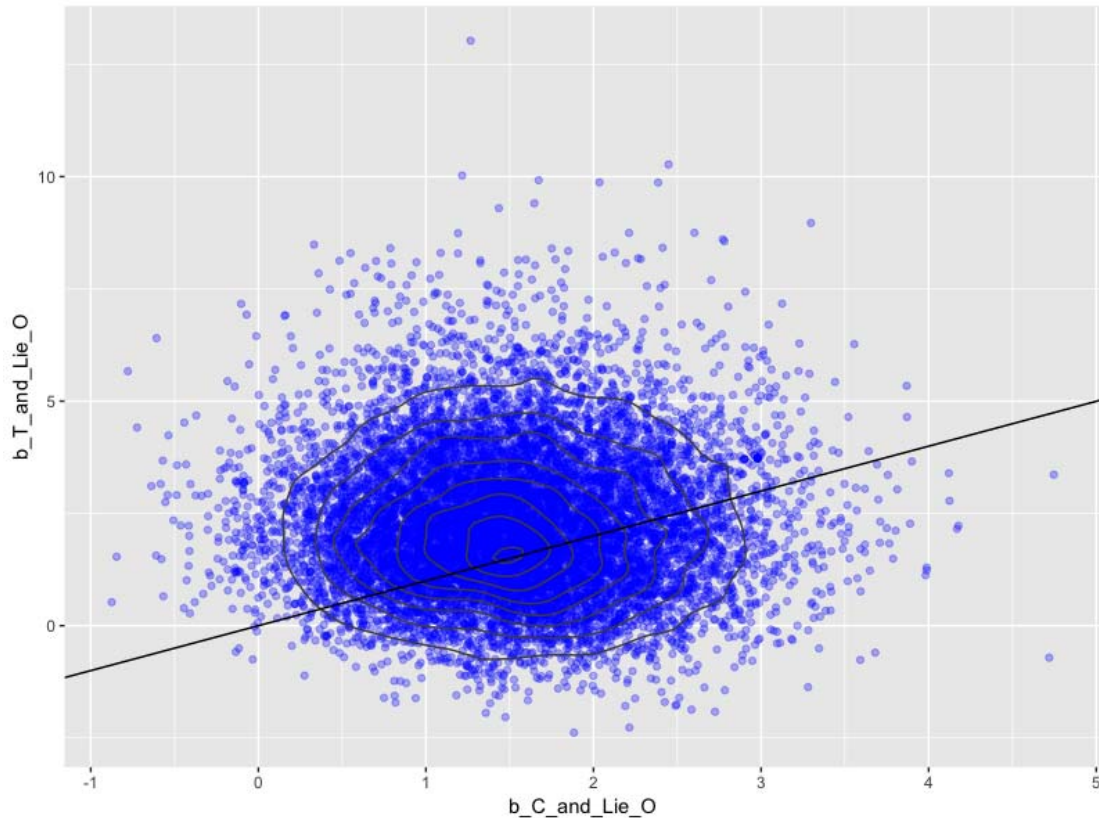
Hình 15

Tương tự với Khổng giáo, Lão giáo cũng hỗ trợ yếu tố nói dối mạnh hơn nhiều so với Phật giáo.

Cuối cùng, là cặp khổng giáo và lão giáo:

```
> bvl plotDensity2d(model, "b C and Lie O", "b T and Lie O",
color scheme = "blue")
```

Kiểm tra về mặt hình học trong Hình 16:



Hình 16

Nếu so sánh 2 yếu tố Lão và Khổng khi kết hợp với nói dối, phân phối các hệ số này khá tương đồng và đều tập trung ở khoảng dương.

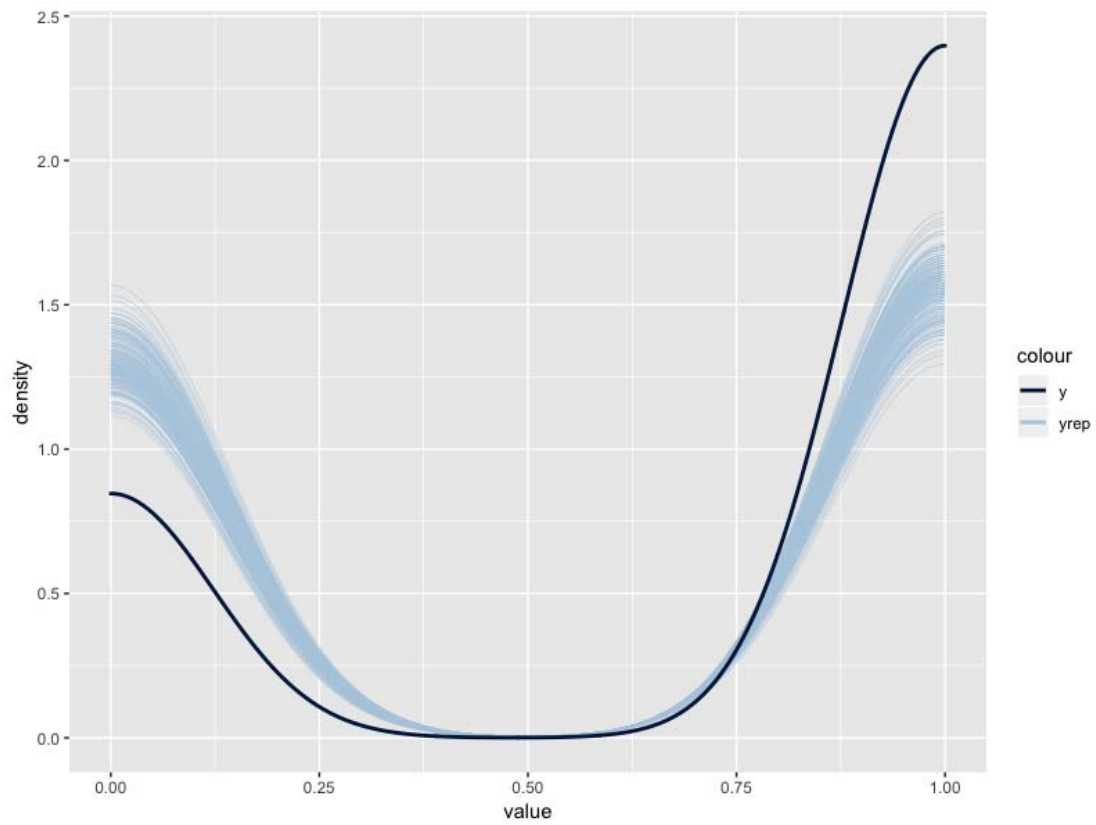
Tính chất quan trọng của khổng giáo tại Việt Nam là đạo tiến thân, với mong muốn làm quan, phụng sự đất nước, nên vì thế có thể hành vi nói dối, dù sai trái nhưng hỗ trợ cho quá trình tiến thân của nhân vật.

So sánh có can thiệp và không có can thiệp:

Code kiểm tra biến outcome ("O") khi có và không có yếu tố bên ngoài tác động (xem phần *predict*) cho ta 2 đồ thị phân phối **y_rep** dưới đây.

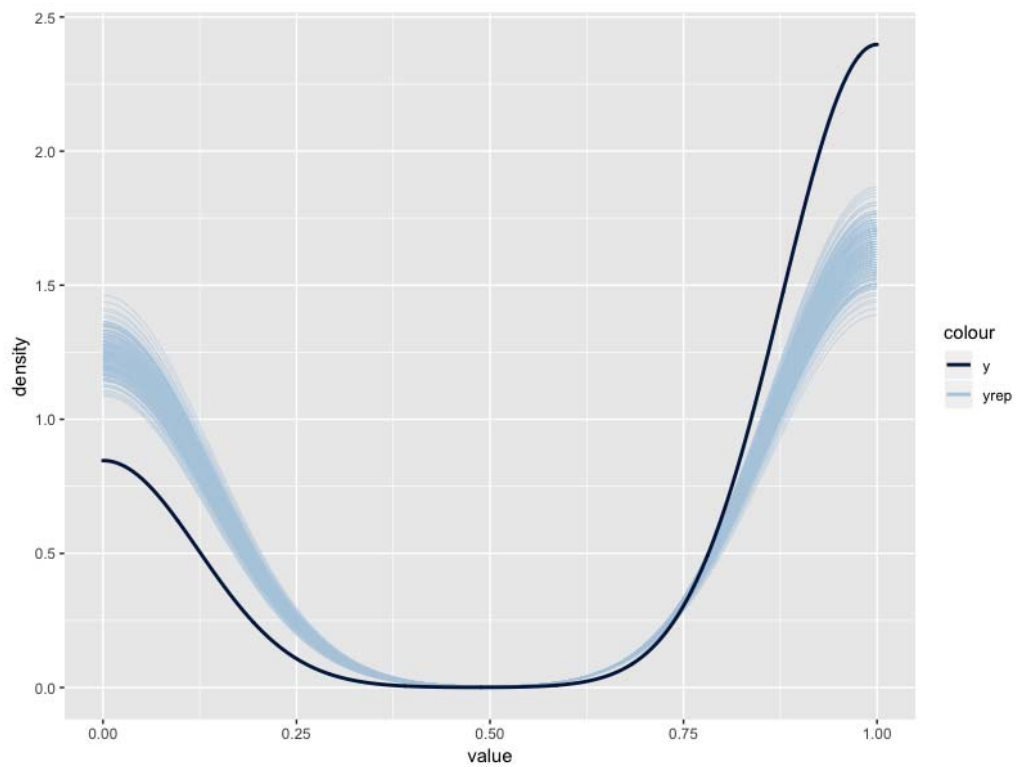
```
> bvl_plotTest(model, "O", "Int1 or Int2 1")
> bvl_plotTest(model, "O", "Int1_or_Int2_2")
```

Cặp hình ảnh dưới đây tương ứng với cặp giá trị điều khiển của **Int1_or_Int2** trong cặp lệnh phía trên:



Hình 17a

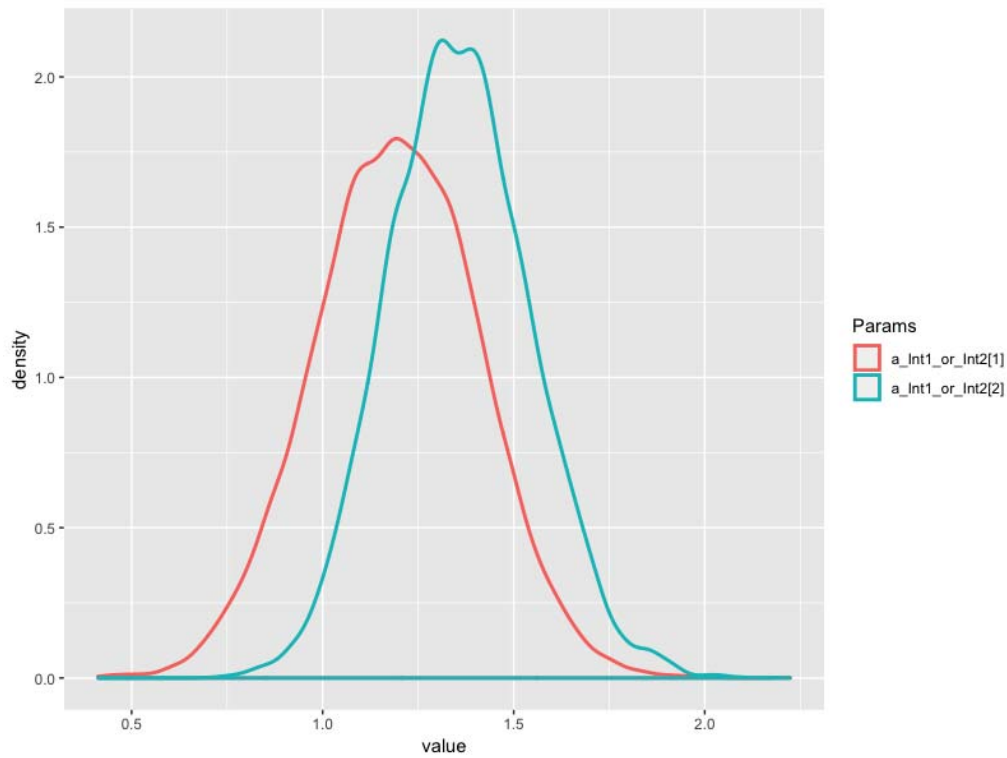
Hình 17a là hình dạng khi `Int1_or_Int2=1` và Hình 17b cho biết hình dạng mô phỏng khi `Int1_or_Int2=2`:



Hình 17b

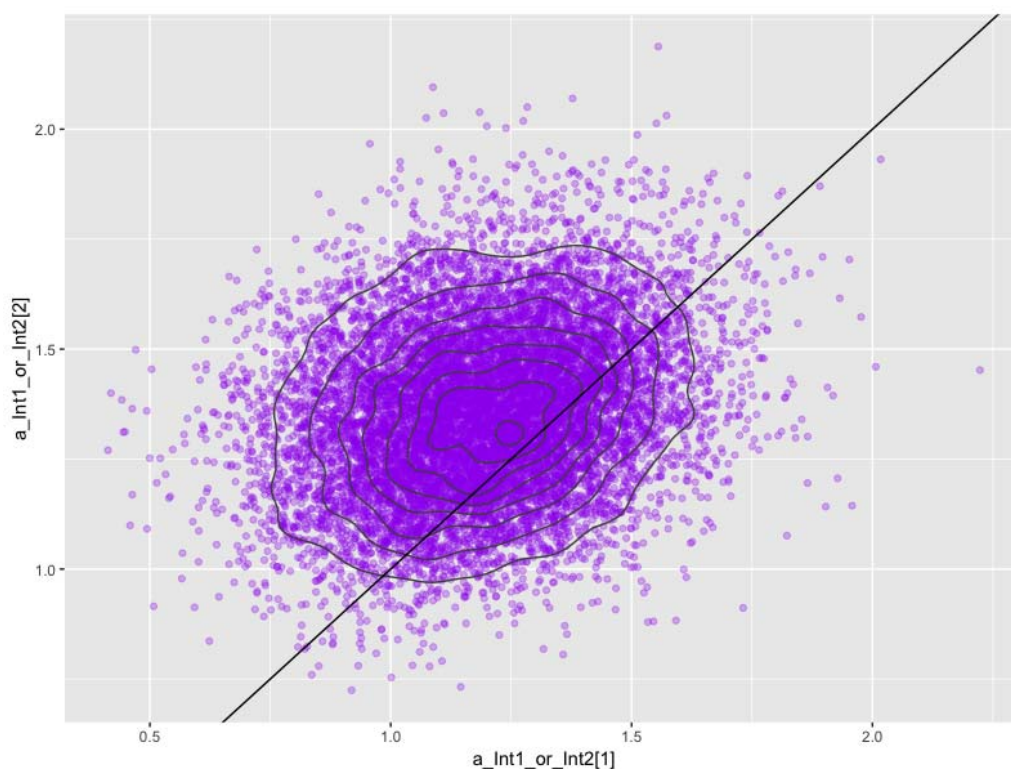
Có thể thấy 2 đồ thị này khá giống nhau, kết cục câu truyện không khác nhau nhiều khi có hoặc không có tác động bên ngoài.

Nếu phân tích hệ số `a_Int1_or_Int2[1]` và `a_Int1_or_Int2[2]` ta cũng có thể thấy 2 hệ số này phân phối cùng dáng điệu. Khi có tác động của các thể lực bên ngoài, nhân vật chính có tốt lên nhưng không nhiều.



Hình 18

Kiểm tra tương quan thay đổi của hai mức giá trị điều khiển về mặt hình học qua hình 19 dưới đây:



Hình 19

Hình 19 cũng cho thấy hai giá trị hội tụ đều có khoảng phân phối khá tương đồng, đều dương và tập trung chủ yếu ở cùng một khoảng giá trị.

Cập nhật hướng dẫn

Hướng dẫn sử dụng sẽ được tiếp tục cập nhật dựa trên việc bổ sung các bài toán AISDL đánh giá có tính tiêu biểu với thuật toán, và theo quy tắc Rules for Versioning của **bayesvl**.

Đội ngũ

Nhóm nghiên cứu AISDL và SDAG đã đóng góp trong các công đoạn khác nhau (dữ liệu, thử chương trình, kiểm tra kết quả, viết bản thảo nghiên cứu) của dự án phát triển chương trình **bayesvl** tới thời điểm v0.8 bao gồm:

- Hồ Mạnh Tùng,
- Nguyễn Tô Hồng Kông,
- Hồ Mạnh Toàn,
- Phạm Hùng Hiệp,
- Nguyễn Minh Hoàng, và
- Vương Thu Trang.

Cảm ơn

Nhóm nghiên cứu AISDL và SDAG cảm ơn hỗ trợ nguồn lực nghiên cứu về tài chính và dữ liệu của Vương & Associates, đặc biệt là Đàm Thu Hà và Nghiêm Phú Kiên Cường. Logo biểu trưng: họa sỹ Bùi Quang Khiêm.

References

- [1] R Development Core Team. (2010). R: A Language and Environment for Statistical Computing. Vienna, Austria: R Foundation for Statistical Computing. Retrieved from <http://www.R-project.org/>
- [2] Vương, Q. H., & La, V. P. (2019). BayesVL package for Bayesian statistical analyses in R. Github: <<https://github.com/sshpa/bayesvl>>. Version 0.8.
- [3] Ho, M. T., Vương, Q. H. (2019). The values and challenges of ‘openness’ in addressing the reproducibility crisis and regaining public trust in social sciences and humanities. *European Science Editing*, 45(1), 14-17, DOI: 10.20316/ESE.2019.45.17021.
- [4] Vương, Q. H., Ho, M. T., & La, V.-P. (2019). ‘Stargazing’ and p-hacking behaviours in social sciences: some insights from a developing country. *European Science Editing*, 45(2), 54-55.
- [5] Vương, Q. H. (2018). “How did researchers get it so wrong?” The acute problem of plagiarism in Vietnamese social sciences and humanities. *European Science Editing*, 44(3), 56-58. doi:10.20316/ese.2018.44.18003
- [6] Vương, Q. H. (2017). Open data, open review and open dialogue in making social sciences plausible. Retrieved from <http://blogs.nature.com/scientificdata/2017/12/12/authors-corner-open-data-open-review-and-open-dialogue-in-making-social-sciences-plausible/>
- [7] Vương, Q. H., & Napier, N. K. (2017). Academic research: The difficulty of being simple and beautiful. *European Science Editing*, 43(2), 32-33; DOI: 10.20316/ESE.2017.43.002
- [8] Vương, Q.-H., Bui, Q.-K., La, V.-P., Vương, T.-T., Nguyen, V.-H. T., Ho, M.-T., . . . Ho, M.-T. (2018). Cultural additivity: behavioural insights from the interaction of Confucianism, Buddhism and Taoism in folktales. *Palgrave Communications*, 4(1), 143. DOI: 10.1057/s41599-018-0189-2
- [9] Vương, Q.-H., Bui, Q.-K., La, V.-P., Vương, T.-T., Ho, M.-T., Nguyen, H.-K. T., . . . Ho, M.-T. (2019, 2019, January 26). *Cultural evolution in Vietnam’s early 20th century: a Bayesian networks analysis of Franco-Chinese house designs*. (Working Paper No. PKA-1901). arXiv Preprints, arXiv:1903.00817v1 [Stat.AP].

- [10] Ho, M.-T., La, V.-P., Nguyen, M.-H., Vuong, T.-T., Nghiem, K.-C. P., Tran, T., . . . Vuong, Q.-H. (2019). Health care, medical insurance, and economic destitution: A dataset of 1,042 stories. *Data*, 4(2), 57, DOI: 10.3390/data4020057.
- [11] Le, A.-V., Do, D.-L., Pham, D.-Q., Hoang, P.-H., Duong, T.-H., Nguyen, H.-N., ... Vuong, Q.-H. (2019). Exploration of youth's digital competencies: a dataset in the educational context of Vietnam. *Data*, 4(2), 69, DOI: 10.3390/data4020069.
- [12] McElreath, R. (2018). *Statistical Rethinking: A Bayesian Course with Examples in R and Stan*. London, UK: Chapman and Hall/CRC.
- [13] Scutari, M. (2010). Learning Bayesian Networks with the bnlearn R Package. *Journal of Statistical Software*, 35(3), 1-22. doi:10.18637/jss.v035.i03
- [14] Muth, C., Oravecz, Z., & Gabry, J. (2018). User-friendly Bayesian regression modeling: A tutorial with rstanarm and shinystan. *Quantitative Methods for Psychology*, 14(2), 99-119. DOI: 10.20982/tqmp.14.2.p099
- [15] Gabry, J., & Goodrich, B. (2016). rstanarm: Bayesian applied regression modeling via Stan, R package version 2.10.0. Retrieved from <https://cran.rproject.org/web/packages/rstanarm/index.html>
- [16] Huu, N. V., Vuong, Q.-H., & Ngoc, T. M. (2005). Central limit theorem for functional of jump Markov processes. *Vietnam Journal of Mathematics*, 33(4), 443-461.
- [17] Thao, H. T. P., & Vuong, Q.-H. (2015). A Merton model of credit risk with jumps. *Journal of Statistics Applications and Probability Letters*, 2(2), 97-103. DOI: 10.12785/jsapl/020201
- [18] Vuong, Q. H. (2001). *Black-Scholes PDE: A finance application*. Paper presented at the International Conference on Differential Equation Approximation and Applications, November 2001, Vietnam National University and Institute of Mathematics.
- [19] Vuong, Q. H., & Napier, N. K. (2014). Making creativity: the value of multiple filters in the innovation process. *International Journal of Transitions and Innovation Systems*, 3(4), 294-327. DOI: 10.1504/IJTIS.2014.068306
- [20] Vuong, Q. H., Napier, N. K., & Tran, T. D. (2013). A categorical data analysis on relationships between culture, creativity and business stage: the case of Vietnam. *International Journal of Transitions and Innovation Systems*, 3(1), 4-24.

- [21] Vuong, Q. H., Ho, M. T., Nguyen, H. K., & Vuong, T. T. (2018). Healthcare consumers' sensitivity to costs: a reflection on behavioural economics from an emerging market. *Palgrave Communications*, 4, 70, DOI: 10.1057/s41599-018-0127-3.
- [23] Vuong, Q. H. (2015). Be rich or don't be sick: estimating Vietnamese patients' risk of falling into destitution. *SpringerPlus*, 4(1), 529, DOI: 10.1186/s40064-015-1279-x.
- [24] Vuong, Q. H. (2019). *The three teachings, lies, and violence in Vietnamese folktales: a dataset and simulations*. Open Science Framework. Retrieved from: <https://osf.io/kqhf9/>
- [25] Gelman, A., & Rubin, D. B. (1992). Inference from iterative simulation using multiple sequences. *Statistical Science*, 7(4), 457-472, DOI: 10.1214/ss/1177011136
- [26] Brooks, S. P., & Gelman, A. (1998). General Methods for Monitoring Convergence of Iterative Simulations. *Journal of Computational and Graphical Statistics*, 7(4), 434-455, DOI: 10.1080/10618600.1998.10474787
- [27] Lynch, S. M. (2007). *Introduction to Applied Bayesian Statistics and Estimation for Social Scientists*. New York, NY: Springer.