ベイズ統計モデリング入門

はじめに

統計モデリングとベイズ推論を行う上で、理解したい三つの概念

- モデル
- パラメータ

- データ

モデルを立て、与えられたデータを用いて、パラメータを推定し、未知のデータを予測したり現象の起こる原理を説明する。

データ

データはある種のランダムさを持って得られたものであると考える。

データの発生の仕方をうまく説明したい。 また推定されたパラメータに基づいて、将来的に得られるであろうデータを予測したい。

モデル

データの発生の仕方を説明するための大枠。 または分析する前提としての枠組み。

基本的にはモデルをどう設定するか、よいモデルを作れるかが分析者のやること。

パラメータ

モデルの細かな設定を決めるための数値などをいう。

確率

乱数やサンプリングという考え方について、Rやstanを用いながら親しみを持ってもらう。

例題

まずは上の統計モデリングの三つの概念について、特にパラメータの推定について理解するため、 次のような簡単な問題を考えよう。

手元にコインがある。これを投げて表が出るか、裏が出るかを予測したい。 すでに10回投げて7回表が出るということを観測している。

- モデルとして、コインはある一定の確率pで表が出て、$1-p$で裏が出るとする。
- このpがパラメータである。
- また10回投げて7回表が出た、というのが現在のデータである。

このデータを用いてpを推定する、というのがパラメータの推定。

推定されたパラメータを用いて、次に表が出るか裏が出るかを予測することができる。

どのようにパラメータpを推定するか？

- 10回中7回なので$p = 0.7$と推定
- コインだから表裏半々ぐらい出るはずなので、ちょっと表が出やすくて$p = 0.6$と推定

など、色々考え方がある。

一つ目の推測は、最尤法という方法に基づいた結論と同じものになる。 最尤法の考え方を理解するために、次のような実験を行うことにしよう。

```
x <- rbinom(n = 1, size = 10, prob = 0.7)
x
```

```
## [1] 8
```

```
y <- rbinom(n = 1, size = 10, prob = 0.6)
y
```

```
## [1] 5
```

```
z <- rbinom(n = 1, size = 10, prob = 0.5)
z
```

```
## [1] 7
```

```
x1 <- rbinom(n = 100, size = 10, prob = 0.8)
x1
```

```
##   [1]  6  9  6 10  8  8  7  7  9  8  9  7  8  8 10  8  9  9  9  6  9 10  6
##  [24]  9  6  9  9  9  8  7  8 10  8 10  7  8  8 10  8 10  8  7  8  7  8  9
##  [47]  9  9  9  9  8  9  7  6  7  5  8  7  9  9  7  6  8  8 10  7  6  8  8
##  [70]  9  9  9  6  9  7  5  7  7  9  9  8  5  9  6  6  6  8  8  7  9  7  8
##  [93]  8 10  8  9  7  6  5  7
```

```
x2 <- rbinom(n = 100, size = 10, prob = 0.7)
x2
```

```
##   [1]  9  7  7  7  8 10  8  8  5  8  6  7  6  7  7  6  8  6  8  6  6  9  6
##  [24]  7  8  8  5  9  8  7  6  7  6  8  6  8  8  8  9  7  8  6  7  9  7  8
##  [47]  5  8  8  6  5  8  6  7  8  8  7  8  6  4  6  4  6  8  5  7 10  7  7
##  [70]  8  6  8  8  8  7  7  9  6  9  5  7  9  8  7  9  5  6  3  7  8  7  6
##  [93] 10  7  8  7  6  4  8  6
```

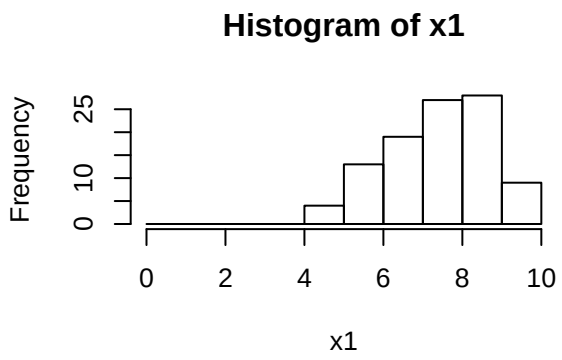
```
x3 <- rbinom(n = 100, size = 10, prob = 0.6)
x3
```

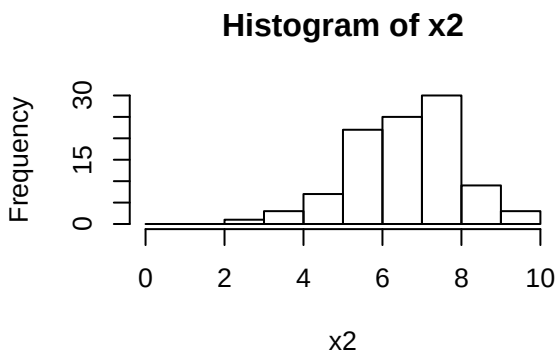
```
##   [1]  5  5  7  9  7  7  7  8  6  8  7  5  5  6  6  6  5  8  8  8  7  6  5
##  [24]  7  5  5  7  6  3  8  5  5  8  3  7  4  7  4  8  6  6  7  5  8  4  2
##  [47]  7  5  7  5  5  7  7  7  7  8  5  7  7  9  3  4  3  5  5  3  6  8  8
##  [70]  7  5  4  8  5  8  7  8  8  6  2  7  6  4  6  7  3  7  7  6 10  7  6
##  [93]  7  3  3  6  4  7  7  8
```

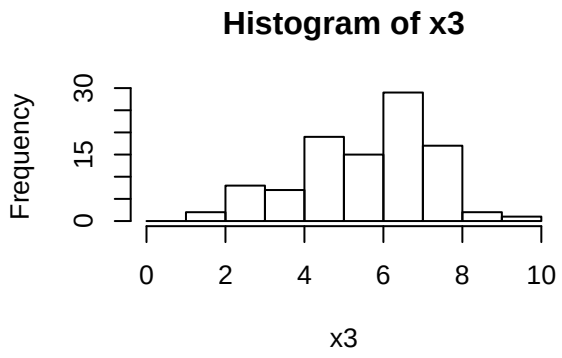
```
x4 <- rbinom(n = 100, size = 10, prob = 0.5)
x4
```

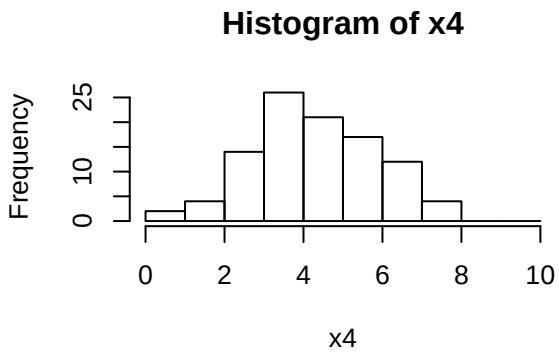
```
##   [1] 5 3 4 3 4 3 6 6 2 5 6 6 6 7 4 3 6 3 2 4 7 5 4 4 7 6 5 6 5 4 3 6 8 4 6
##  [36] 5 3 4 3 5 5 5 5 3 7 5 4 0 7 8 4 7 3 5 4 4 4 1 5 4 7 2 8 3 7 7 4 6 4 5
##  [71] 5 4 5 5 4 4 4 5 4 8 3 7 4 4 4 4 2 6 6 7 6 5 6 3 6 3 6 7 5 5
```

```
par(mfrow=c(2,2))
hist(x1, breaks = 0:10)
hist(x2, breaks = 0:10)
hist(x3, breaks = 0:10)
hist(x4, breaks = 0:10)
```









このようにパラメータpの値によって、表の出る回数にばらつきがある。 この表のでる回数の割合をパラメータpに対する尤度という。

今回のデータでは、表の出た回数が7回なので、 パラメータpごとに表が7回出る確率を計算し、それを$L(p)$と表す。 これを尤度関数という。

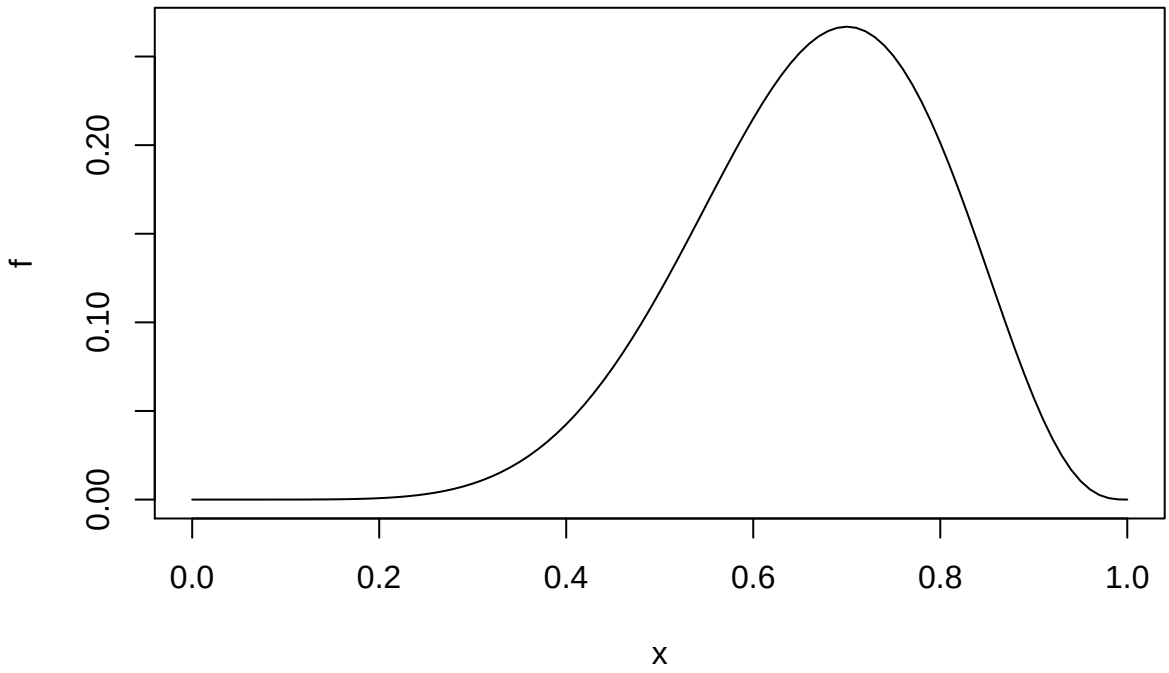
今回のデータに基づいて尤度関数を計算すると

$$L(p) = 120\, p^7 (1-p)^3$$

となる。

最尤法では尤度関数が最大となるpをパラメータの推定値とする。 これのグラフをかくと、

```
f <- function(p){120*p^7*(1-p)^3}
plot(f, 0, 1)
```



であり、$p = 0.7$の時に最大となることがわかる。

今回はパラメータpを持つ二項分布モデルに基づいて、現象を説明している。

ではベイズ推論の枠組みでは、上の問題に対してどのような回答をするか？

パラメータpをある決まった値として推定するのではなく、 ある程度幅を持ったなぜ分布として推定する。

分布にすると何が違うかを説明する。 例えば与えられたデータが

- 2回投げて1回表
- 100回投げて50回表

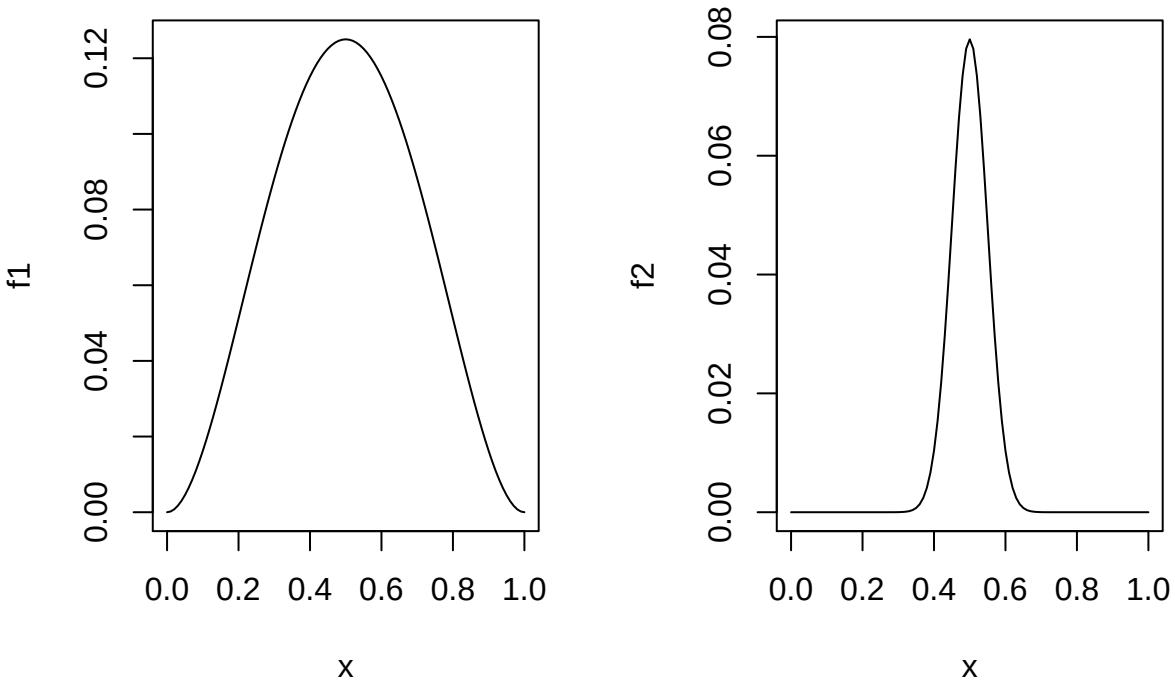
というふた通りの可能性を考えよう。 この時、最尤法でパラメータを推定するとどちらも$p = 0.5$となる。 しかし、後者の方が信頼できそうだろう。

このような信頼度の違いを分布の形状として表現することができる。

例えばpの分布が以下の二つでどのように違うか検討せよ。

```
f1 <- function(p){choose(2,1) * p^2 * (1-p)^2}
f2 <- function(p){choose(100,50) * p^50 * (1-p)^50}

par(mfrow=c(1,2))
plot(f1, 0, 1)
plot(f2, 0, 1)
```



これがパラメータに分布を持たせることの一つの利点である。

ベイズ推論の考え方

パラメータはある決まった真の値があるのではなく、 何らかの確率分布にしたがっているとして推測を行う。

- 尤度関数
- 事前分布
- 事後分布

の三つの概念について理解する。

尤度関数

上ですでに出てきたように、パラメータを入力すると現在観測しているデータが発生する確率を計算する関数。

モデルとデータが与えられると関数が定まる。

事前分布と事後分布

いずれもパラメータの確率分布。 データを観測する前の分布のことを事前分布、データを観測した後の分布を事後分布という。

観測を通して事前分布を事後分布に取り替えることを、パラメータ更新、ベイズ更新などという。

パラメータの分布

ベイズ推論ではデータの発生を確率現象と捉えるのみでなく、パラメータも確率的な現象と考える。

確率分布とベイズの定理

ここからは、ベイズ推論の枠組みを理解するために以下の設定を通して、 確率分布、特にパラメータ分布の更新について理解しよう。

設定

外からは見分けのつかない二つの箱A, Bの中に赤いボールと白いボールがいくつか入っている。

以下の手順に従ってボールを取り出し赤いボールなら賞金1万円、白いボールなら罰金1万円もらえるというゲームを考えよう。

- まずはじめにどちらかの箱を選ぶ。
- 次に観察期間として選んだ箱から規定の回数、ボールを取り出し色を確認して戻すという操作を行う。
- そののち、実際にゲームに挑戦するか決める。
- 挑戦する場合は先ほど選んだ箱の中からボールを一つとり、色に応じて賞金を得る。

さて、どのような戦略を立てるべきか。

例題1

- 箱Aには赤いボールが5個、白いボールが0個
- 箱Bには赤いボールが0個、白いボールが5個

入っているということがあらかじめわかっているとしよう。

この場合には、観察期間として1回ボールを確認すれば、選んだ箱がAなのかBなのか完全に決定することができる。

従ってそれに応じて最終的に挑戦するかどうか決定すればよい。

例題2

- 箱Aには赤いボールが3個、白いボールが1個
- 箱Bには赤いボールが1個、白いボールが3個

入っていることがわかっているとしよう。

また観察期間として、一度だけボールの色を確認できるとする。

Rの乱数を利用して実験してみよう。

```
rbinom(n = 10, size = 1, prob = 0.75)
```

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

```
rbinom(n = 10, size = 1, prob = 0.25)
```

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

赤いボールが出た場合には箱Aと推定し、白いボールが出た場合には箱Bと推定するのがもっともらしいであろう。

さらにそれを踏まえた上で、最終的に挑戦するかを決定できる。

このような判断を数学的に考えていこう。 まずは

- 確率分布
- 条件付き確率
- 同時確率
- 周辺確率
- 尤度関数
- ベイズの定理

という概念を順番に理解していこう。

確率分布

Xという出来事が起こる確率を$P(X)$とかく。 例えば

$$P(\text{コインを投げて表が出る}) = \frac{1}{2}$$

のように書く。 これを省略して

$$P(\text{表}) = \frac{1}{2}$$

などとも書く。

今回の問題では箱がAなのかBなのか、ボールが赤なのか白なのか、といった出来事について確率を考える。

$$P(A) = \frac{1}{2}$$

$$P(B) = \frac{1}{2}$$

と設定しよう。 二つの箱からランダムに選ぶのでこのように設定するのが妥当だろう。

確率変数と確率分布

上のようにある状況で起こりうる出来事全体が$X_0, X_1, \ldots, X_n$である時、 $P(X_1)$から$P(X_n)$にたいして0以上1以下の数を与える。

このようなものを（離散的な）確率分布という。 この時、合計が1になるように決めなければならない。

例1

「コインを1回投げる」という操作に注目すると、起こりうる結果は表が出るか裏が出るかどちらか。 この結果を表す確率変数Xを導入する。

$P(\text{表})$と$P(\text{裏})$を決めればこの現象を記述できる、と考える。 この二つの値が0以上1以下で、合計が1になるように配分してやる。
例えば

$$P(X = \text{表}) = \frac{1}{2},\ P(X = \text{裏}) = \frac{1}{2}$$

としてもよいし、

$$P(X = \text{表}) = \frac{1}{3},\ P(X = \text{裏}) = \frac{2}{3}$$

としてもよい。 極端な場合には

$$P(X = \text{表}) = 1,\ P(X = \text{裏}) = 0$$

としてもよい。

例2

「サイコロを1回投げる」という操作に注目すると、起こりうる結果は1,2,3,4,5,6のどれかが出るということ。 サイコロの目を表す確率変数をXとする。

この場合は

$$P(X = 1),\ P(X = 2),\ P(X = 3),\ P(X = 4),\ P(X = 5),\ P(X = 6)$$

を決めてやる。 例えば

$$P(X = 1) = \frac{1}{6},\ P(X = 2) = \frac{1}{6},\ P(X = 3) = \frac{1}{6},\ P(X = 4) = \frac{1}{6},\ P(X = 5) = \frac{1}{6},\ P(X = 6) = \frac{1}{6}$$

としてもよいし、

$$P(X = 1) = \frac{1}{10},\ P(X = 2) = \frac{2}{10},\ P(X = 3) = \frac{3}{10},\ P(X = 4) = \frac{1}{10},\ P(X = 5) = \frac{2}{10},\ P(X = 6) = \frac{1}{10}$$

としてもよい。

条件付き確率と同時分布

次にボールを一つ取り出した時、赤であるか白であるかの確率を計算したいが、このままでは難しい。

なぜならどちらの箱を選んだかによって赤いボールの出やすさが異なるからである。

このように箱の種類とボールの色といういくつかの条件を組み合わせる場合には、 条件付き確率や同時分布を用いて記述する。

例えば選んだ箱がAだという条件のもと、赤いボールを取り出す確率を表現したい。

Yという条件のもとでのXの確率を

$$P(X \mid Y)$$

と表し、これを条件付き確率という。

箱の種類を表す確率変数をθとし、取り出したボールの色を表す変数をXとする。

Aの箱という条件のもとで赤いボールを取り出す確率は

$$P(X = \text{赤} \mid \theta = A) = \frac{3}{4}$$

であり、Aの箱という条件のもとで白いボールを取り出す確率は

$$P(X = \text{白} \mid \theta = A) = \frac{1}{4}$$

となる。 この二つの合計は1になっている。 よってこれは確率分布を与えていることに注意しよう。

同様に箱Bを選んだという条件のもとでのボールの色に関する条件付き確率を表現すると、

$$P(X = \text{赤} \mid \theta = B) = \frac{1}{4},\ P(X = \text{白} \mid \theta = B) = \frac{3}{4}$$

と書くことができる。 これも確率分布を与えている。

同時分布と周辺分布

箱を選びボールを取り出す、という二つの条件をまとめて考えると、起こりうる出来事としては

- 箱Aから赤いボール
- 箱Aから白いボール
- 箱Bから赤いボール
- 箱Bから白いボール

の4パターンがある。 このように二つ以上の条件をまとめて考えた確率を同時確率という。

これらの確率を計算すると、

$$P(X = \text{赤} \cap \theta = A) = \frac{3}{4} \times \frac{1}{2} = \frac{3}{8}$$
$$P(X = \text{白} \cap \theta = A) = \frac{1}{4} \times \frac{1}{2} = \frac{1}{8}$$
$$P(X = \text{赤} \cap \theta = B) = \frac{1}{4} \times \frac{1}{2} = \frac{1}{8}$$
$$P(X = \text{白} \cap \theta = B) = \frac{3}{4} \times \frac{1}{2} = \frac{3}{8}$$

と計算できる。 4つを合計すると1になるので、これも確率分布である。

また$P(\theta = A), P(\theta = B)$や$P(X = \text{赤}), P(X = \text{白})$と行ったように、どちらかの条件のみを考えた確率分布を周辺分布という。
この場合には

$$P(X = \text{赤}) = P(X = \text{赤} \cap \theta = A) + P(X = \text{赤} \cap \theta = B) = \frac{1}{2}$$
$$P(X = \text{白}) = P(X = \text{白} \cap \theta = A) + P(X = \text{白} \cap \theta = B) = \frac{1}{2}$$

として計算できる。 クロス集計表を思い浮かべるとよい。

ベイズの定理

クロス集計表をじっと見ていると、次のような関係が成り立つことがわかる。

$$P(X \mid Y)P(Y) = P(X \cap Y) = P(Y \mid X)P(X)$$

これを式変形して得られる次の公式をベイズの定理という。

$$P(X \mid Y) = \frac{P(Y \mid X)P(X)}{P(Y)}$$

これを用いることで、$P(A \mid \text{赤})$と$P(B \mid \text{赤})$のいずれが大きいかを計算することができる。

言い換えると、赤いボールを一個とったあとで、目の前の箱がAであるのかBであるのか、どちらの確率が高いかを計算できる。

例題1の場合で言えば、 Aの箱を選んだという条件のもとで赤いボールが出る確率は

$$P(X = \text{赤} \mid \theta = A) = 1$$

であり、Aの箱を選んだという条件のもとで白いボールが出る確率は

$$P(X = \text{白} \mid \theta = A) = 0$$

となる。 またBの箱から取ったという条件のもとでの赤白のそれぞれの確率は

$$P(X = \text{赤} \mid \theta = B) = 0,\ P(X = \text{白} \mid \theta = B) = 1$$

となる。

同時分布は

$$P(\theta = A \cap X = \text{白}) = 0,\ P(\theta = A \cap X = \text{赤}) = \frac{1}{2},\ P(\theta = B \cap X = \text{白}) = \frac{1}{2},\ P(\theta = B \cap X = \text{赤}) = 0$$

となり、さらに

$$P(X = \text{赤}) = P(\theta = A \cap X = \text{赤}) + P(\theta = B \cap X = \text{赤}) = \frac{1}{2}$$

$$P(X = \text{白}) = P(\theta = A \cap X = \text{白}) + P(\theta = B \cap X = \text{白}) = \frac{1}{2}$$

であることがわかる。

さて、赤いボールを取り出したとき、選んだ箱がどちらの箱の確率が高いだろうか？

$P(\theta = A \mid X = \text{赤})$と$P(\theta = B \mid X = \text{赤})$の大小を比較しよう。

ベイズの定理を使えば、

$$P(\theta = A \mid \text{赤}) = \frac{P(\text{赤} \mid \theta = A)P(\theta = A)}{P(\text{赤})} = 1 \times \frac{1}{2} \div \frac{1}{2} = 1$$

$$P(\theta = B \mid \text{赤}) = \frac{P(\text{赤} \mid \theta = B)P(\theta = B)}{P(\text{赤})} = 0 \times \frac{1}{2} \div \frac{1}{2} = 0$$

と計算できる。

例題2の場合を考えよう。

1回ボールを取り出すと赤いボールがでた、という事実を観測データXとしよう。

すると、赤いボールが出る確率をパラメータの関数として表現すると以下のようになる。

$$L(A) = P(X = \text{赤} \mid \theta = A) = \frac{3}{4}$$

$$L(B) = P(X = \text{赤} \mid \theta = B) = \frac{1}{4}$$

このLが尤度関数である。

ベイズの定理を使えば、

$$P(\theta = A \mid X = \text{赤}) = \frac{P(X = \text{赤} \mid \theta = A)P(\theta = A)}{P(X = \text{赤})} = \frac{L(A)P(\theta = A)}{P(X = \text{赤})} = \frac{3}{4} \times \frac{1}{2} \div \frac{1}{2} = \frac{3}{4}$$

$$P(\theta = B \mid X = \text{赤}) = \frac{P(X = \text{赤} \mid \theta = B)P(\theta = B)}{P(X = \text{赤})} = \frac{L(B)P(\theta = B)}{P(X = \text{赤})} = \frac{1}{4} \times \frac{1}{2} \div \frac{1}{2} = \frac{1}{4}$$

と計算できる。

これが事後分布。

この事後分布を用いて、次のボールYが赤である確率を計算すると、

$$P(Y = \text{赤}) = P(Y = \text{赤} \mid \theta = A)P(\theta = A \mid X = \text{赤}) + P(Y = \text{赤} \mid \theta = B)P(\theta = B \mid X = \text{赤}) = \frac{3}{4} \times \frac{3}{4} + \frac{1}{4} \times \frac{1}{4} = \frac{10}{16}$$

となる。

さてここまでの言葉遣いを用いて、ベイズ推測の枠組みを改めて説明すると、以下のようになる。

$P(\theta = A), P(\theta = B)$という確率分布を事前分布、$P(\theta = A \mid \text{白}), P(\theta = B \mid \text{白})$という確率分布を事後分布という。

- モデルは確率分布$P(\text{赤} \mid \theta = A), P(\text{白} \mid \theta = A)$という確率分布から決まるもの
- パラメータは箱の種類AまたはB
- データは観察期間にでたボールの記録

ということになる。

例題3

次に箱の設定は前と同じで観察できる回数が5回になったとしよう。

前回とはデータとして想定する出来事の起こり方が変わってくる。

- 例題2ではボールを取り出して赤が出るか白が出るか
- 例題3ではボールを5回取り出して何回赤が出るか

という点が異なる。

このような出来事の確率分布を記述するため

- 二項分布

について理解する。

```
rbinom(n = 1, size = 5, prob = 0.75)
```

```
## [1] 4
```

```
rbinom(n = 1, size = 5, prob = 0.25)
```

```
## [1] 1
```

今回の問題設定は、このような数を見たときにprobの値がどのようになっているか推測する、ということができる。

二項分布

表と裏がそれぞれ$\frac{1}{2}$の確率で出るコイン、つまり$P(\text{表}) = P(\text{裏}) = \frac{1}{2}$であるようなコインを$N$回投げる、 という操作を考えよう。

$N = 1$のとき、表が出るか裏が出るかどちらかで、表が1回出る確率、0回出る確率はどちらも$\frac{1}{2}$である。

$N = 2$のとき、表表、表裏、裏表、裏裏の4パターンあり、

$$P(\text{表表}) = P(\text{表裏}) = P(\text{裏表}) = P(\text{裏裏}) = \frac{1}{2} \times \frac{1}{2} = \frac{1}{4}$$

と全て等しい。これは1回目の確率と2回目の確率をかけて計算できる。

出方のパターンではなく、表の出る回数Xの分布を考えよう。 $N = 2$の時は表の出る回数は0,1,2のいずれかなので$P(X = 0),\ P(X = 1),\ P(X = 2)$という確率分布を与える。 上で計算したことから

$$P(X = 0) = \frac{1}{4},\ P(X = 1) = \frac{1}{4} + \frac{1}{4} = \frac{1}{2},\ P(X = 2) = \frac{1}{4}$$

となることがわかる。 $X = 1$となるのは2通りあって、それらの確率の和を考えればよい。

$N = 3$ならば、表裏のでる出方は表表表、表表裏、表裏表、表裏裏、裏表表、裏表裏、裏裏表、裏裏裏、と八通りあり、いずれも確率$\frac{1}{2}$である。
従って

$$P(X = 0) = \frac{1}{8},\ P(X = 1) = \frac{1}{8} + \frac{1}{8} + \frac{1}{8} = \frac{3}{8},\ P(X = 2) = \frac{1}{8} + \frac{1}{8} + \frac{1}{8} = \frac{3}{8},\ P(X = 3) = \frac{1}{8}$$

と計算できる。

$N = 4$の場合、表裏の出るパターンを全て列挙し、表の出る回数の確率分布を計算しよう。

一般にN回コインを投げた時、表がk回出るパターンは

$$_NC_k = \frac{N!}{k!(N-k)!}$$

通りある。

例えば

$$_2C_1 = \frac{2}{1} = 2,\ _4C_2 = \frac{4!}{2! \times 2!} = 6$$

などとなる。

したがってN回コインを投げた時の、表が出る回数の確率分布は

$$P(X = k) = \ _NC_k \left(\frac{1}{2}\right)^k \left(\frac{1}{2}\right)^{N-k}$$

と計算できる。

より一般に1回コインを投げた時に表の出る確率pとする。 つまり

$$P(\text{表}) = p,\ P(\text{裏}) = 1 - p$$

とする。 この時N回コインを投げた時の、表が出る回数の確率分布は

$$P(X = k) = \ _NC_k\ p^k (1-p)^{N-k}$$

となる。 これが二項分布である。

例題4

箱の個数がA, Bで異なるとしよう。 例えばAの箱が3個、Bの箱が2個あるとする。

この時は、パラメータの事前分布を

$$P(\theta = A) = \frac{3}{5},\ P(\theta = B) = \frac{2}{5}$$

と設定する。

5回ボールを取り出した結果、赤いボールが4個、白いボールが1個、というデータが与えられたとしよう。

確率変数Xを赤いボールの個数とする。

このとき尤度関数は、

$$L(A) = P(X = 4 \mid \theta = A) = {}_5C_4 \left(\frac{3}{4}\right)^4 \left(\frac{1}{4}\right)^1 = \frac{5 \times 3^4}{4^5}$$

$$L(B) = P(X = 4 \mid \theta = B) = {}_5C_4 \left(\frac{3}{4}\right)^1 \left(\frac{1}{4}\right)^4 = \frac{5 \times 3}{4^5}$$

であり、

$$P(X = 4) = P(X = 4 \mid \theta = A)P(\theta = A) + P(X = 4 \mid \theta = B)P(\theta = B) = \frac{5 \times 3^4}{4^5} \times \frac{3}{5} + \frac{5 \times 3}{4^5} \times \frac{2}{5} = \frac{3^5 + 6}{4^5}$$

となる。

したがってベイズの定理を使えば、事後分布は

$$P(\theta = A \mid X = 4) = \frac{P(X = 4 \mid \theta = A)P(\theta = A)}{P(X = 4)} = \frac{L(A)P(\theta = A)}{P(X = 4)} = \frac{5 \times 3^4}{4^5} \times \frac{3}{5} \div \frac{3^5 + 6}{4^5} = \frac{3^5}{3^5 + 6}$$

$$P(\theta = B \mid X = 4) = \frac{P(X = 4 \mid \theta = B)P(\theta = B)}{P(X = 4)} = \frac{L(B)P(\theta = B)}{P(X = 4)} = \frac{5 \times 3}{4^5} \times \frac{2}{5} \div \frac{3^5 + 6}{4^5} = \frac{6}{3^5 + 6}$$

と計算できる。

この事後分布を用いて、次のボールYが赤である確率を計算すると、

$$P(Y = \text{赤}) = P(Y = \text{赤} \mid \theta = A)P(\theta = A \mid X = 4) + P(Y = \text{赤} \mid \theta = B)P(\theta = B \mid X = 4) = \frac{3}{4} \times \frac{3^5}{3^5 + 6} + \frac{1}{4} \times \frac{6}{3^5 + 6}$$

となる。

例題5

次に箱の種類がA, B, C, Dと4種類あり、それぞれ赤白のボールの内訳が、

- 箱Aは赤いボールが1個、白いボールが4個
- 箱Bは赤いボールが2個、白いボールが3個
- 箱Cは赤いボールが3個、白いボールが2個
- 箱Dは赤いボールが4個、白いボールが1個

となっているとしよう。 またこの箱が全て一つずつあるとする。

パラメータの事前分布としては

$$P(\theta = A) = P(\theta = B) = P(\theta = C) = P(\theta = D) = \frac{1}{4}$$

とすればよい。

5個ボールを取り出して赤いボールを3個白いボールを2個取り出した、というデータが与えられたとしよう。

確率変数Xを赤いボールの個数とする。

このとき尤度関数は、

$$L(A) = P(X = 3 \mid \theta = A) = {}_5C_3 \left(\frac{1}{5}\right)^3 \left(\frac{4}{5}\right)^2$$

$$L(B) = P(X = 3 \mid \theta = B) = {}_5C_3 \left(\frac{2}{5}\right)^3 \left(\frac{3}{5}\right)^2$$

$$L(C) = P(X = 3 \mid \theta = C) = {}_5C_3 \left(\frac{3}{5}\right)^3 \left(\frac{2}{5}\right)^2$$

$$L(D) = P(X = 3 \mid \theta = D) = {}_5C_3 \left(\frac{4}{5}\right)^3 \left(\frac{1}{5}\right)^2$$

であり、

$$P(X = 3) = P(X = 3 \mid \theta = A)P(\theta = A)+P(X = 3 \mid \theta = B)P(\theta = B)+P(X = 3 \mid \theta = C)P(\theta = C)+P(X = 3 \mid \theta = D)P(\theta =$$

$$= \frac{1}{4} \times 10 \times \frac{1}{5^5} \times (4^2 + 2^3 \times 3^2 + 3^3 \times 2^2 + 4^3) = \frac{26}{5^3}$$

となる。

したがってベイズの定理を使えば事後分布を、

$$P(\theta = A \mid X = 3) = \frac{P(X = 3 \mid \theta = A)P(\theta = A)}{P(X = 3)} = \frac{L(A)P(\theta = A)}{P(X = 3)} = 10 \times \frac{4^2}{5^5} \times \frac{1}{4} \div \frac{26}{5^3} = \frac{4}{65}$$

$$P(\theta = B \mid X = 3) = \frac{P(X = 3 \mid \theta = B)P(\theta = B)}{P(X = 3)} = \frac{L(B)P(\theta = B)}{P(X = 3)} = 10 \times \frac{2^3 \times 3^2}{5^5} \times \frac{1}{4} \div \frac{26}{5^3} = \frac{18}{65}$$

$$P(\theta = \theta = C \mid X = 3) = \frac{P(X = 3 \mid \theta = C)P(\theta = C)}{P(X = 3)} = \frac{L(C)P(\theta = C)}{P(X = 3)} = 10 \times \frac{3^3 \times 2^2}{5^5} \times \frac{1}{4} \div \frac{26}{5^3} = \frac{27}{65}$$

$$P(\theta = D \mid X = 3) = \frac{P(X = 3 \mid \theta = D)P(\theta = D)}{P(X = 3)} = \frac{L(D)P(\theta = D)}{P(X = 3)} = 10 \times \frac{4^3}{5^5} \times \frac{1}{4} \div \frac{26}{5^3} = \frac{16}{65}$$

と計算できる。

この事後分布を用いて、次のボールYが赤である確率は

$$P(Y = \text{赤}) = P(Y = \text{赤} \mid \theta = A)P(\theta = A \mid X = 3)+P(Y = \text{赤} \mid \theta = B)P(\theta = B \mid X = 3)+P(Y = \text{赤} \mid \theta = C)P(\theta = C \mid X = 3)$$

により計算できる。

例題6

箱AからDの中身は前と同じだが、 今回は箱の個数がどのようになっているかわからないとする。

特に何も情報がない場合、全ての箱の割合が均等である一様分布を設定するべきであろう。

また、事前に情報がある場合や、例えば賞金や罰金の額が均等でないといった場合、何らかの情報を持った事前分布を設定することもありうる。

このようにパラメータの事前分布と呼ばれる確率分布を自分で設定する必要がある。 多くの場合、事前分布は一様分布を設定する。
これを無情報事前分布といったりする。

例題7

さらに箱の種類が101種類、型番0番から100番まであり、赤いボールの割合pが型番と一致するとする。

さらに、各番号の箱の個数が一定であるかはわからない状況であるとする。

この場合、モデルが二項分布であることは同じであるが、事前分布の設定を変えることになる。

まず尤度関数を計算しよう。

型番pの箱を選んでいるとしたとき、n回ボールを取り出したうちk回赤いボールになる確率は

$$L(p) = P(X = k \mid \ p) = {}_nC_k \ p^k (1-p)^{n-k}$$

と計算できる。

例えば、5回中4回赤いボールというデータが与えられたとすると、

$$L(p) = P(X = 4 \mid \ p) = {}_5C_4 \ p^4 (1-p)^1$$

さて、事前分布は各pごとにp番の箱がいくつあるか？を事前に設定しておき、 各pごとに値を定めておくものだった。

今回の設定では箱の種類が多いので、連続的な分布を用いて近似的に考えることにしよう。

連続的な分布の場合は、pの取りうる値が無数にあるので、 各pの箱の割合を与えるのではなく、p付近の密度を与える関数を考える。
それを確率密度関数という。

例えば事前になんの情報もない場合には$p \quad = \quad 0$から$p \quad = \quad 1$までが均等にあることを表す一様分布を事前分布として用いる。
一様分布の確率密度関数は

$$f(p) = 1 \ (0 \leq p \leq 1)$$

である。

上と同じようにベイズの定理を用いて事後分布を計算すると、

$$f(p \mid X = 4) = \frac{L(p) f(p)}{P(X = 4)} = \frac{5}{P(X = 4)} p^4 (1-p)^1$$

となる。

ここで$P(X = 4)$を計算するためには積分の計算が必要になる。

例題8

箱に関する設定は上の例題と同じとしよう。

なんらかの情報がある場合には事前分布を自由に設定できる。

ここではベータ分布と呼ばれる確率分布を使うことにしよう。 ベータ分布の確率密度関数は

$$f(p) = \frac{1}{C} \ p^{\alpha - 1} (1-p)^{\beta - 1}$$

である。 ここでα, βはベータ分布のパラメータである。

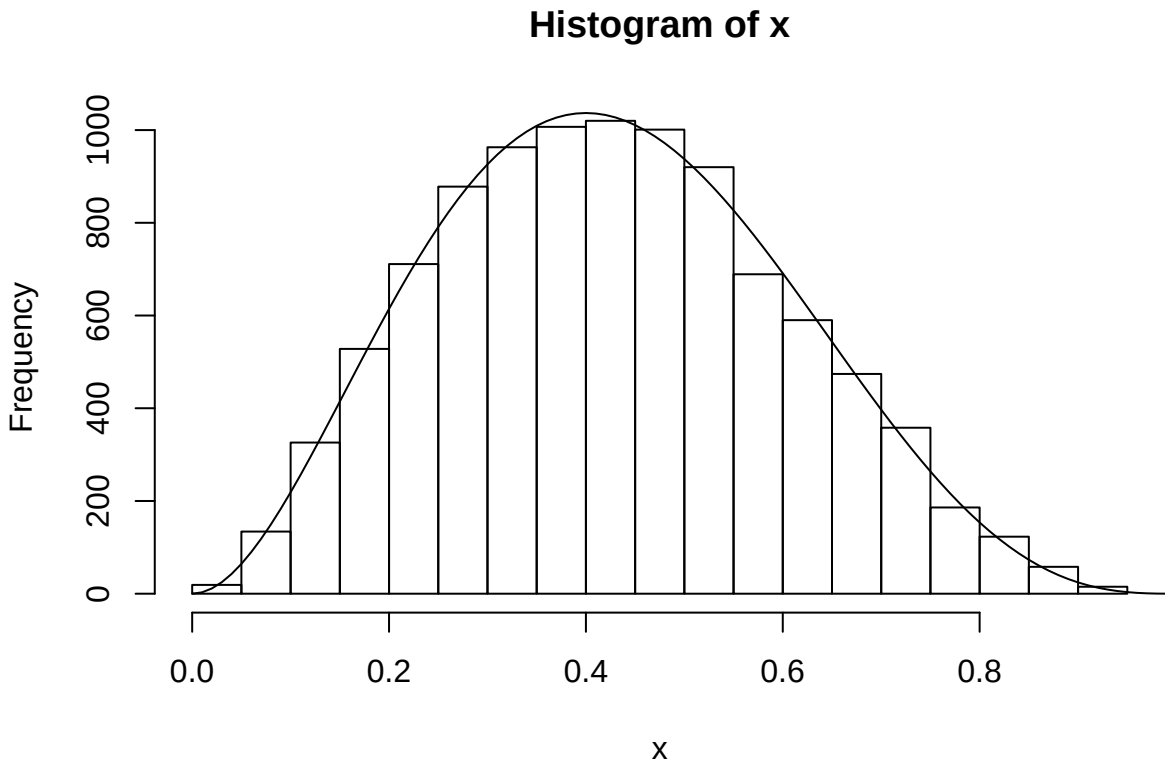
また分母のCは面積を1にするため（確率分布の条件）の定数なので、今回はあまり気にしないでよい。

このxの関数を密度関数に持つ確率分布をベータ分布という。 例えば$Be(1, 1)$は一様分布である。

ベータ分布からサンプリングしてみよう。 また`alpha, beta`を変えると分布の様子がどのように変わるか観察してみよう。

```
n <- 10000
alpha <- 3
beta <- 4
x<-rbeta(n = n, shape1 = alpha, shape2 = beta)
```

```
f <- function(x){x^(alpha-1)*(1-x)^(beta-1)*n*0.05/beta(alpha, beta)}
hist(x)
plot(f,0,1, add=TRUE)
```



事前分布をベータ分布とした時、 事後分布は

$$f(p \mid X = k) = L(p) \times f(p) = p^{\alpha-1}(1-p)^{\beta-1} \times p^k(1-p)^{N-k} = p^{\alpha+k-1}(1-p)^{\beta+N-k-1}$$

となるので、またベータ分布になる。

連続型確率分布

改めて連続型確率分布の考え方について整理しよう。

データやパラメータなどの値が有限でない、あるいはそのように近似できる場合を考える。 例えば、色々なものの大きさや量を測る場合を考えよう。
このようなものを連続型の確率変数という。

それに対して数え上げられるだけの可能性しかとらないものを離散型の確率変数という。

上の例題で言えば箱の種類であったり、ある出来事が何回起こるかといった整数の値で表されるものなどである。

離散型の確率変数の場合、確率分布とは起こりうる結果が$X = x_1, x_2, \ldots, x_n$に対して、

$$P(X = x_1),\ P(X = x_2), \ldots, P(X = x_n)$$

に0以上の数を割り当て、それらの和が1となるように分布させたものだった。

一方で連続型の確率変数の場合、飛び飛びの値ではなくぎっしり詰まった（ある範囲の）数すべてを取ることになる。 したがって、ある決まった値をとる$k)$は0でないとおかしなことになってしまう。

連続分布の場合、$P(a \leq X \leq b)$のように範囲を指定して確率を計算する。

一般に、確率変数Xが連続分布に従うとき、密度関数と呼ばれる関数$f(x)$があって$P(a \leq X \leq b)$は$f(x)$とx軸で囲まれた部分の面積として与えられ
この面積を表す記号として

$$\int_a^b f(x)dx$$

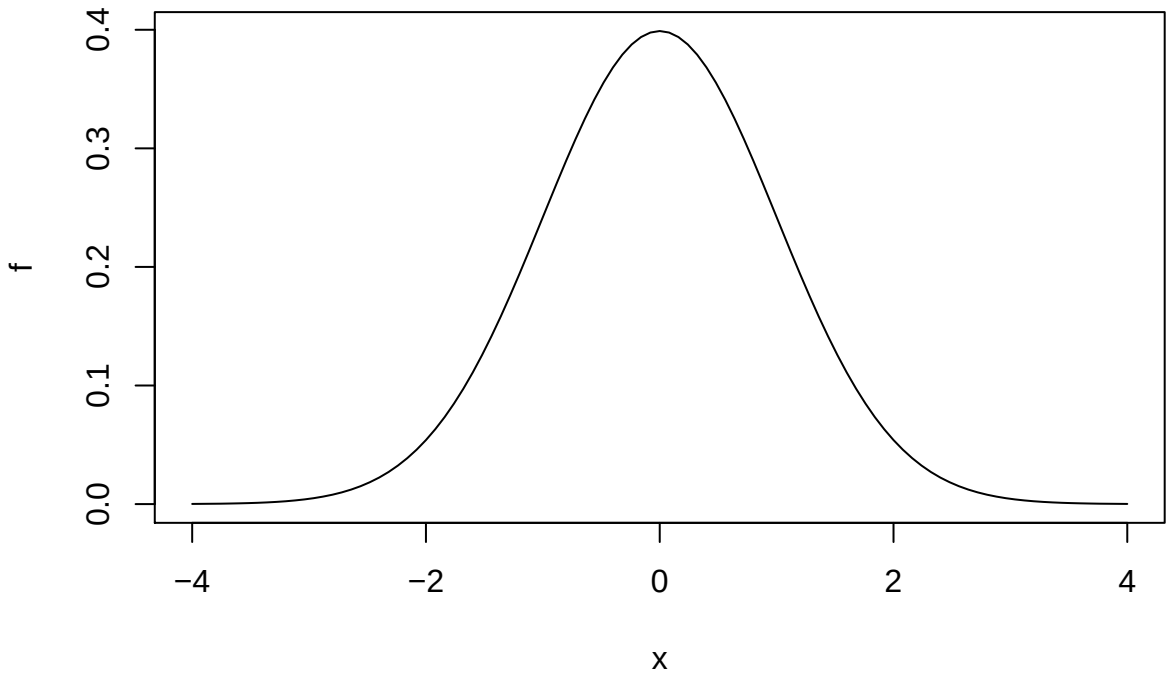
を定める。

例えばあとで見る標準正規分布の場合、 その密度関数は

$$f(x) = \frac{1}{\sqrt{2\pi}} \exp(-\frac{x^2}{2})$$

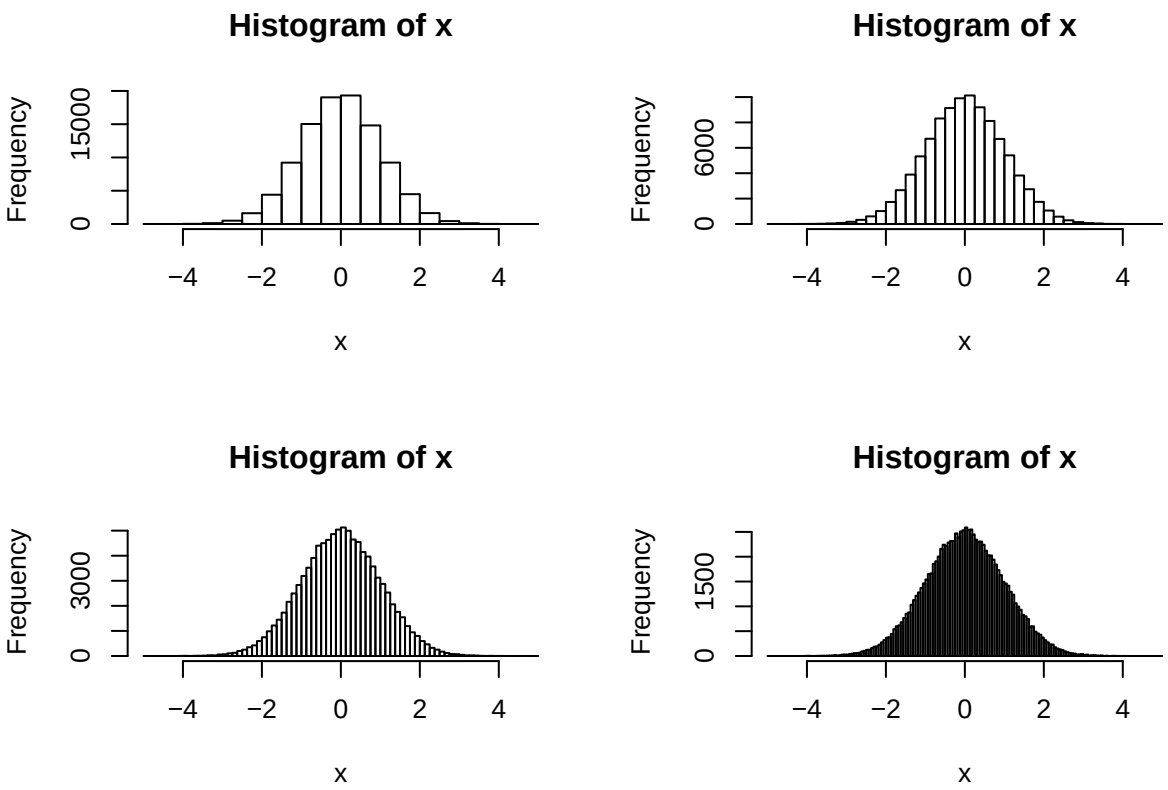
で与えられ、そのグラフは

```
mu <- 0
sigma <- 1
f <- function(x){(1/sqrt(2*pi)*sigma)*exp(-(x-mu)^2/(2*sigma^2))}
plot(f,-4,4)
```



ここから乱数をサンプリングし、ヒストグラムをかくと、

```
N <- 100000
x <- rnorm(N)
par(mfrow=c(2,2))
hist(x,breaks = (0:20)/2-5)
hist(x,breaks = (0:40)/4-5)
hist(x,breaks = (0:80)/8-5)
hist(x,breaks = (0:160)/16-5)
```



となる。 幅を細くしていけば、密度関数の形に近づいていくことが見える。

ヒストグラムでは高さがxがある範囲にあるサンプルの個数で、全体の個数に対する割合が確率であると考えられる。

このように見れば、確率密度関数とその面積を用いて確率を表現できることが納得できる。

連続型確率分布に対するベイズの定理

ベイズ推論に置いて扱う事象は、

- パラメータがどのような条件をみたすか
- 得られたデータがどのようなものであるか

の大きく二つがある。

例えば先ほどまでの例題で言えば

- 箱の名前、種類を表す変数X
- 赤いボールを何個取り出したかを表す変数θ

の二つがあった。

このように二つの連続型確率変数X, Yを同時に考えたい場合、 二変数の密度関数$f(x, y)$を与えることで確率分布を記述する。このとき

$$P(a \leq X \leq b, c \leq Y \leq d) = \int_a^b \int_c^d f(x, y) dx dy$$

として確率分布が与えられる。

$$f(x) = \int_{-\infty}^{\infty} f(x, y) dy$$

とすればXの密度関数が得られる。 これは離散分布の場合に

$$P(X = x) = \sum_i P(X = x, Y = y_i)$$

としたのと同じこと。

またこの場合に条件付き分布は密度関数

$$f(y \mid x) = \frac{f(x, y)}{f(x)}$$

により与えられる。

連続分布に対してもベイズの定理が成り立つ。

$$f(y \mid x) = \frac{f(x \mid y)f(y)}{f(x)}$$

連続型確率分布の例

以下ではいくつかの連続型確率分布の例とそれを用いたベイズ推定の例題を紹介する。

- 一様分布
- 正規分布
- ガンマ分布

一様分布

ある範囲の実数が一様に現れるような確率分布を一様分布という。

例えば0以上1以下の実数が一様に現れるような確率分布であれば、その確率密度関数は

$$f(x) = \begin{cases} 1 & 0 \leq x \leq 1 \\ 0 & x < 0, 1 < x \end{cases}$$

となる。 このとき、例えば$P(0.5 \leq x \leq 0.6) = 0.1$と計算できる。

正規分布

正規分布は次の密度関数$f(x)$で定まる確率分布。

$$f(x) = \frac{1}{\sqrt{2\pi}\sigma} \exp(-\frac{(x-\mu)^2}{2\sigma^2})$$

ここでμとσは正規分布のパラメータ。

確率密度関数のグラフは$x = \mu$がピークで左右対称。 この幅の広がり具合を決めるのがσであり、

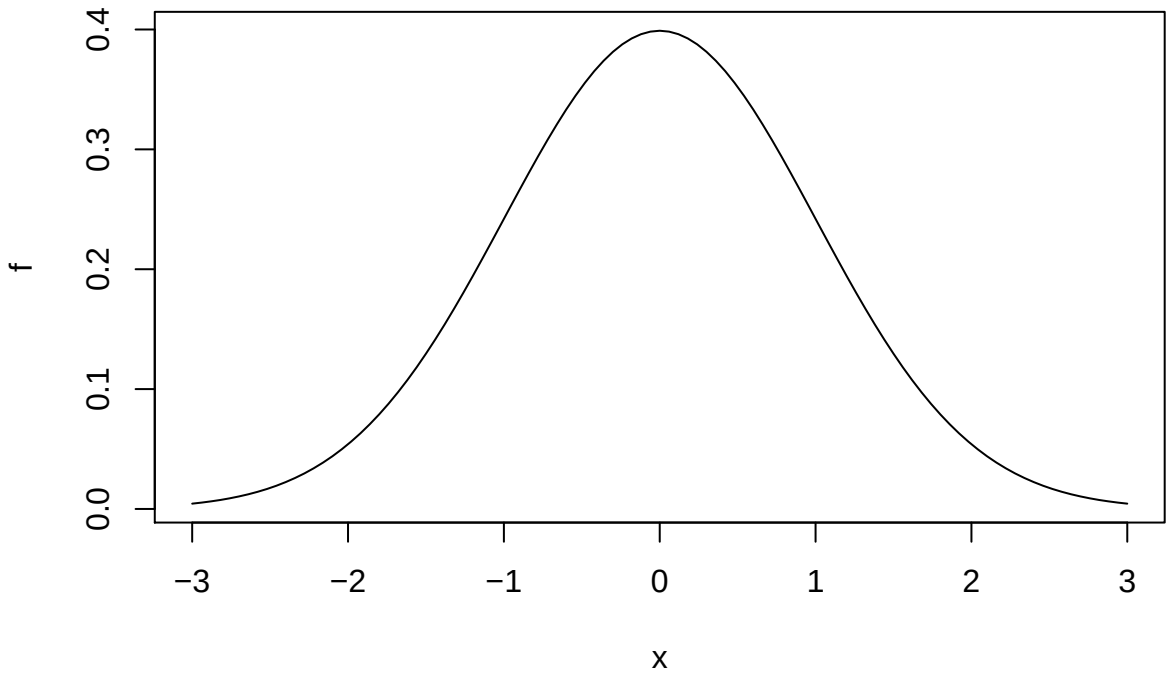
パラメータμ, σの正規分布に従う確率変数は

- 平均μ
- 分散σ^2

である。

正規分布は誤差の分布として使われる。 色々と良い性質を持ち、もっとも基本的な連続分布。

```r
mu <- 0
sigma <- 1
f <- function(x){(1/sqrt(2*pi)*sigma)*exp(-(x-mu)^2/(2*sigma^2))}
plot(f,-3,3)
```



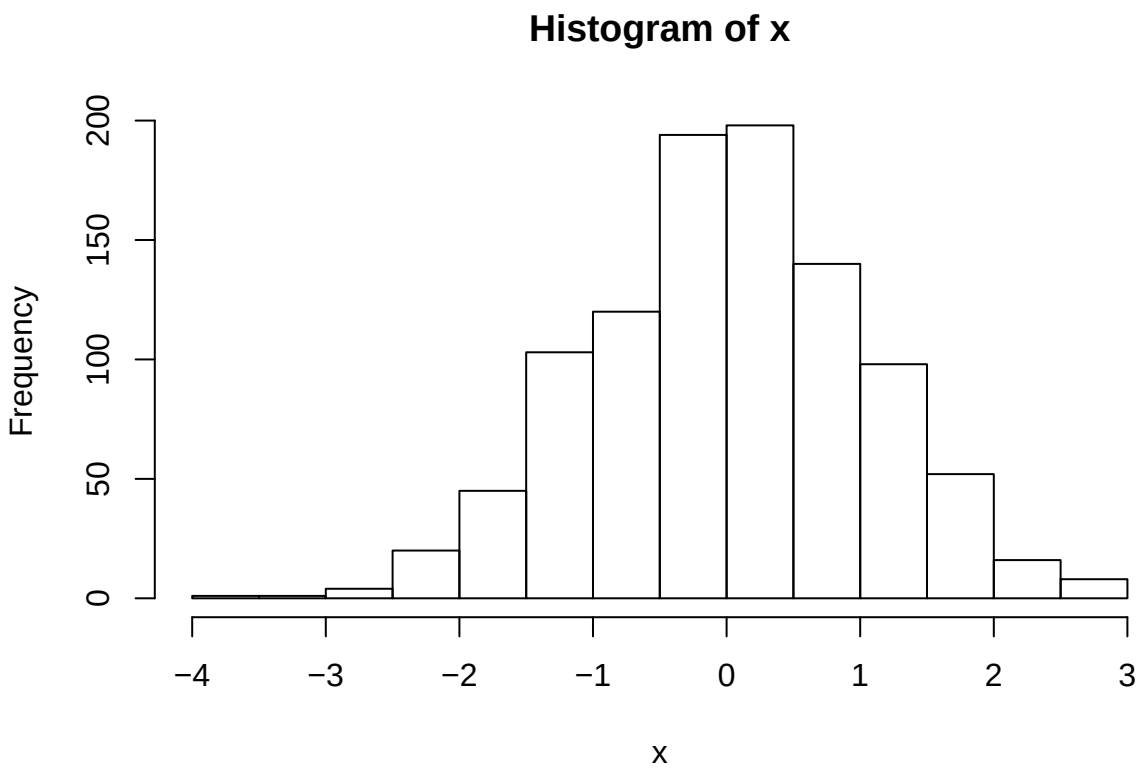
確率はRで計算できる

```r
pnorm(1)
```

```
## [1] 0.8413447
```

正規分布からのサンプリングは

```r
mu <- 0
sigma <- 1
N <- 1000
x <- rnorm(n = N, mean = mu, sd = sigma)
hist(x)
```

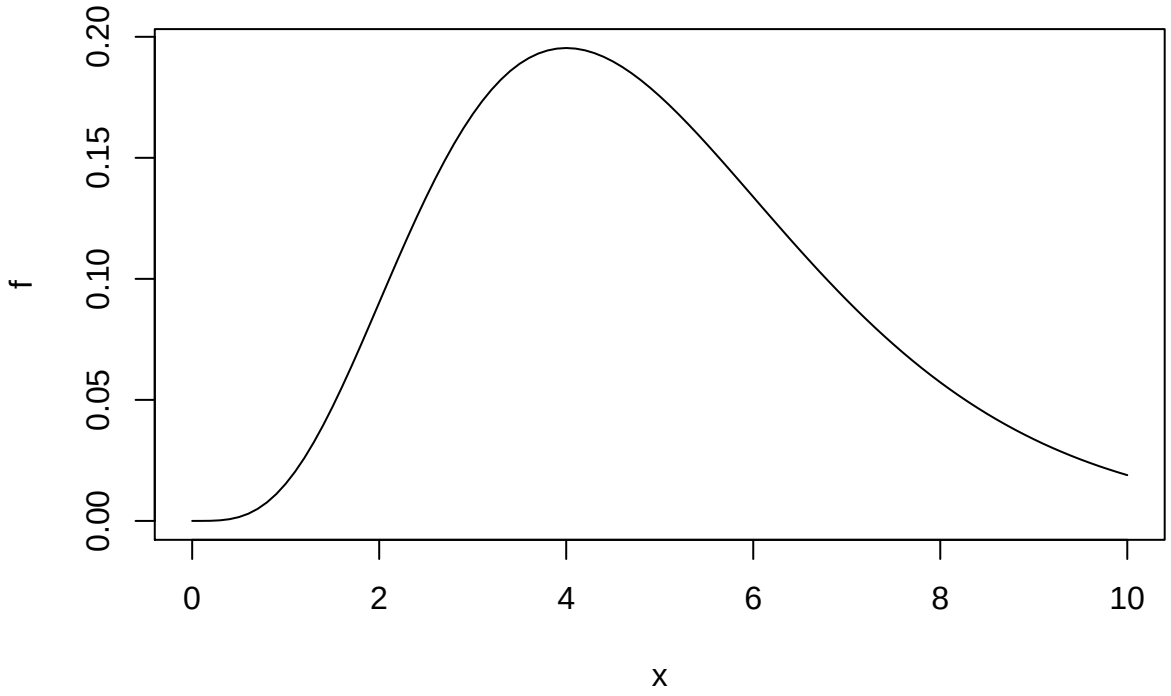


ガンマ分布

ガンマ分布は次のような密度関数を持つ。

$$f(x) = \frac{\theta^k}{\Gamma(k)} e^{-x\theta} x^{k-1}$$

```
theta <- 1
k <- 5
f <- function(x){exp(-x*theta)*x^{k-1}*theta^k/gamma(k)}
plot(f,0,10)
```



ガンマ分布のパラメータはkとθである。 ガンマ分布に従う確率変数は

- 平均はk/θ
- 分散はk/θ^2

となる。

連続型確率分布を用いたベイズ推定

それでは改めて連続型確率分布を用いたベイズ推定の問題を考えてみよう。

例題：ポアソン分布のパラメータ推定

ある地域での一日の事故件数Xを予測したい。

これをポアソン分布モデルを用いて予測してみよう。 データとして、これまで10日間の事故件数データが、

```
x<-c(8,4,5,5,7,9,7,5,5,4)
```

と与えられているとしよう。

ポアソン分布はパラメータλをもつ離散型確率分布である。 パラメータλのポアソン分布に従う確率変数Xは

$$P(X=k)=e^{-\lambda}\frac{\lambda^k}{k!}$$

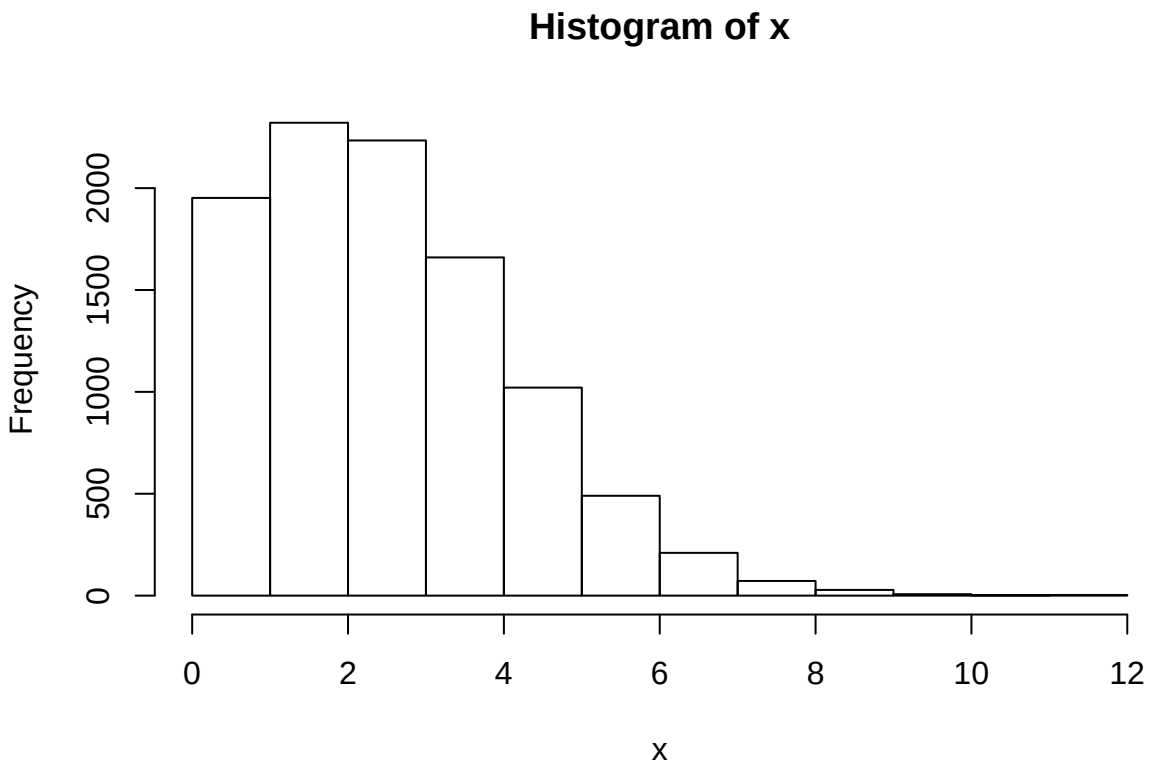
と確率を計算でき、

- 平均λ
- 分散λ

である。

ポアソン分布からサンプリングした変数のヒストグラムを、λの値を変えて観察してみよう。

```r
N <- 10000
lambda <- 3
x <- rpois(n = N, lambda = lambda)
hist(x,breaks=0:max(x))
```



今回のデータから尤度関数を計算すると、

$$L(\lambda) = e^{-\lambda}\frac{\lambda^{x_1}}{x_1!} \times \cdots \times e^{-\lambda}\frac{\lambda^{x_n}}{x_n!} = \prod_{i=1}^{N} e^{-\lambda}\frac{\lambda^{x_i}}{x_i!} = e^{-10\lambda}\frac{\lambda^{59}}{x_1! \cdots x_n!}$$

Rで尤度関数を計算すると、

```r
x<-c(8,4,5,5,7,9,7,5,5,4)
L <- function(lambda){
  lik <- 1
  for (i in 1:length(x)){
    lik <- lik * exp(-lambda)*lambda^x[i]/factorial(x[i])
  }
  return(lik)
}
```

パラメータλの事前分布として、ガンマ分布を用いることにする。

事前分布をパラメータ$k = 5, \theta = 1$のガンマ分布としよう。

$$f(\lambda) = \frac{\theta^5}{\Gamma(5)}e^{-\lambda}\lambda^4$$

すると、事後分布は

$$e^{-11\lambda}\lambda^{63}$$

により決まるガンマ分布になる。 つまりパラメータ$k = 64, \theta = 11$のガンマ分布である。

$$p(\lambda \mid D) = \frac{9^{64}}{64!} e^{-9\lambda} \lambda^{63}$$

この事後分布を用いて、一日の事故件数を予測することができる。

$$P(X = x) = \int_{-\infty}^{\infty} e^{-\lambda} \frac{\lambda^x}{x!} p(\lambda \mid D) d\lambda$$

例題

上の例題ではパラメータの分布に連続型確率分布を用いたが、データの分布にも連続型分布を用いることができる。

そのような例題として、正規分布に従うデータのベイズ推定の例題を用意した。

正規分布の例題

工業製品の規格を調査する。 ある工場で作られた製品の大きさが、以下のようであった。

```
x <- c(102.3,100.4,99.3,100.2,100.3,99.4,101.5,99.3,98.9,100.1)
```

製品が規格通り製造されているか調査したい。

正規分布モデルを用いる。 パラメータは平均μと標準偏差σで、これに対して上のデータから尤度関数を計算すると

$$L(\mu, \sigma) = \frac{1}{\sqrt{2\pi}\sigma} \exp(\frac{-(x_1 - \mu)^2}{2\sigma^2}) \times \cdots \times \frac{1}{\sqrt{2\pi}\sigma} \exp(\frac{-(x_N - \mu)^2}{2\sigma^2}) = \prod_{i=1}^{N} \frac{1}{\sqrt{2\pi}\sigma} \exp(\frac{-(x_i - \mu)^2}{2\sigma^2})$$

である。

```
x <- c(102.3,100.4,99.3,100.2,100.3,99.4,101.5,99.3,98.9,100.1)
L <- function(mu, sigma){
  L <- 1
  for(i in 1:10){
    L <- L*dnorm(x[i],mu,sigma)
  }
}
```

さてここでは

- 平均の事前分布として、平均100, 分散1の正規分布
- 分散の事前分布として、平均1, 分散1の逆ガンマ分布

を用いることにしよう。

$$p_1(\mu) = \frac{1}{\sqrt{2\pi}} \exp(-\frac{(\mu - 100)^2}{2})$$

$$p_2(\sigma^2) = 4(\sigma^2)^{-4} \exp(-\frac{2}{\sigma^2})$$

すると事後分布は

$$p(\mu, \sigma^2 \mid D) = L(\mu, \sigma^2) p_1(\mu) p_2(\sigma^2)$$

$$= \prod_{i=1}^{N} \frac{1}{\sqrt{2\pi}\sigma} \exp(\frac{-(x_i - \mu)^2}{2\sigma^2}) \times \frac{1}{\sqrt{2\pi}} \exp(-\frac{(\mu - 100)^2}{2}) \times 4(\sigma^2)^{-4} \exp(-\frac{2}{\sigma^2})$$

となる。

これはσ, μについて、二変数の確率分布となっているが、 σに適当な値を入れれば正規分布、μに適当な値を入れれば逆ガンマ分布になる。

この事後分布を用いての予測をすると、

$$P(X \leq a) = \int_{-\infty}^{a} dx \int \frac{1}{\sqrt{2\pi\sigma^2}} \exp(-\frac{(x-\mu)^2}{2\sigma^2}) p(\mu, \sigma^2 \mid D) d\mu d\sigma$$

と計算できる。

自然な共役分布

一般的にはパラメータの事後分布の密度関数を正確に計算したり、 それを用いてパラメータ推定値の平均や分散、あるいは予測分布を計算するのは困難で

これを克服するための一つの方法は、事前分布として自然共役分布と呼ばれる特定の分布を用いることである。 ベイズ推論の枠組みで、尤度関数はモデル

例えば

- 二項分布のパラメータ推定にベータ分布
- 正規分布のパラメータ推定に正規分布、逆ガンマ分布
- ポアソン分布のパタメータ推定にガンマ分布

などがある。 これらを使えば、正確に事後分布を決定することができる。

階層モデル

ある学校で問題数10問のテストを20人の生徒に行ったところ、以下のような結果をえた。 生徒が問題に正解する能力を正解率pとして、
これをパラメータとする二項分布モデルで考えてみよう。

すると、二項分布で想定される分布の様子、あるいは分散の大きさと異なってしまう。

このような場合、正解率に個人差があると考える。

ただし個々の正解率をパラメータ$p_1, p_2, \ldots, p_{20}$とするのではなく、 正解率が別の分布に従って決まっていると考える。

このようなモデルを階層モデルという。

マルコフ連鎖モンテカルロ法(MCMC)

例えば階層モデルなどより複雑なモデルなどの場合にも、 コンピュータの助けを借りることでパラメータの推定やデータの予測が可能になる。
これを実現するための一つの方法がMCMCを用いたランダムサンプリングである。

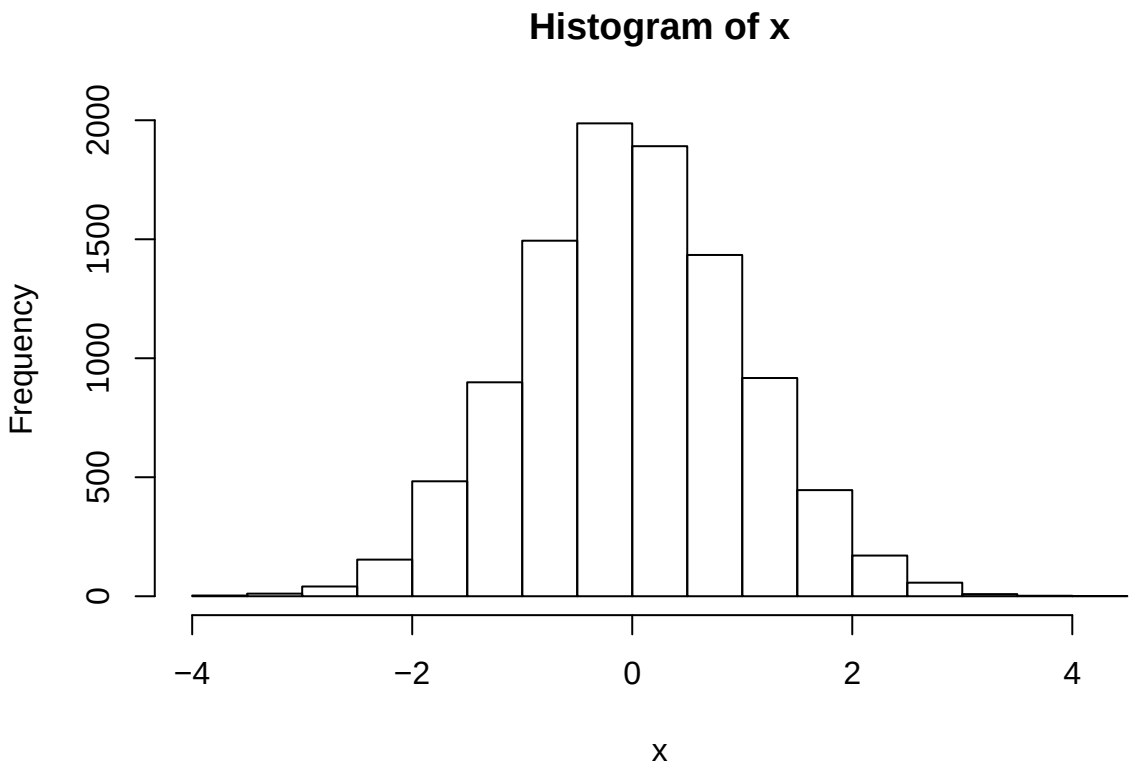
ただし、 この場合には正確な分布を決定するわけではなく、 事後分布に従ったランダムサンプリングが近似的にできるのみであることに注意しよう。

問題設定によっては収束しにくい分布などもあるので、上の自然な共役分布を使った場合と違ってより慎重な運用が必要になる。

MCMCの概要

確率密度関数が与えられたとき、その密度関数に従ってサンプリングを行いたい。 よく知られている確率密度関数であれば、以下のようにRでも簡単にサ

```
N<-10000
x<-rnorm(N)
hist(x)
```



しかし、より複雑なモデリングを考える場合には、あらかじめ用意されている関数では乱数をサンプリングできない。

モンテカルロ法の例

0または1の値をとり、1をとる確率が円の面積の1/4になるような乱数を生成しよう。一様分布に従う乱数はあらかじめ用意されているとする。

```
sample <- function(n){
  x <- c()
  for(i in 1:n){
    r <- runif(2)
    x[i] <- ifelse(r[1]^2+r[2]^2<1, 1, 0)
  }
  return(x)
}

N <- 1000
4*sum(sample(N))/N
```

```
## [1] 3.032
```

メトロポリス法

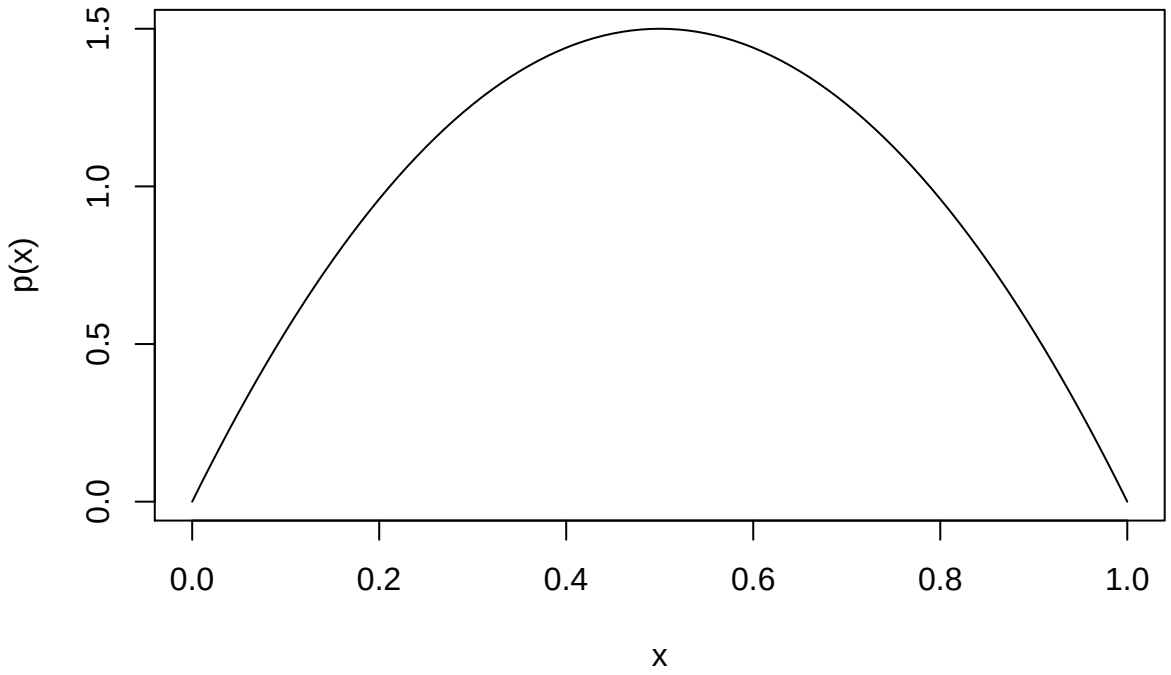
一様乱数があらかじめ与えられている時、確率密度関数$f(x)$に従う乱数$\{x_1, x_2, \ldots\}$を以下の手順でサンプリングする。

x_iからx_{i+1}を作る方法。

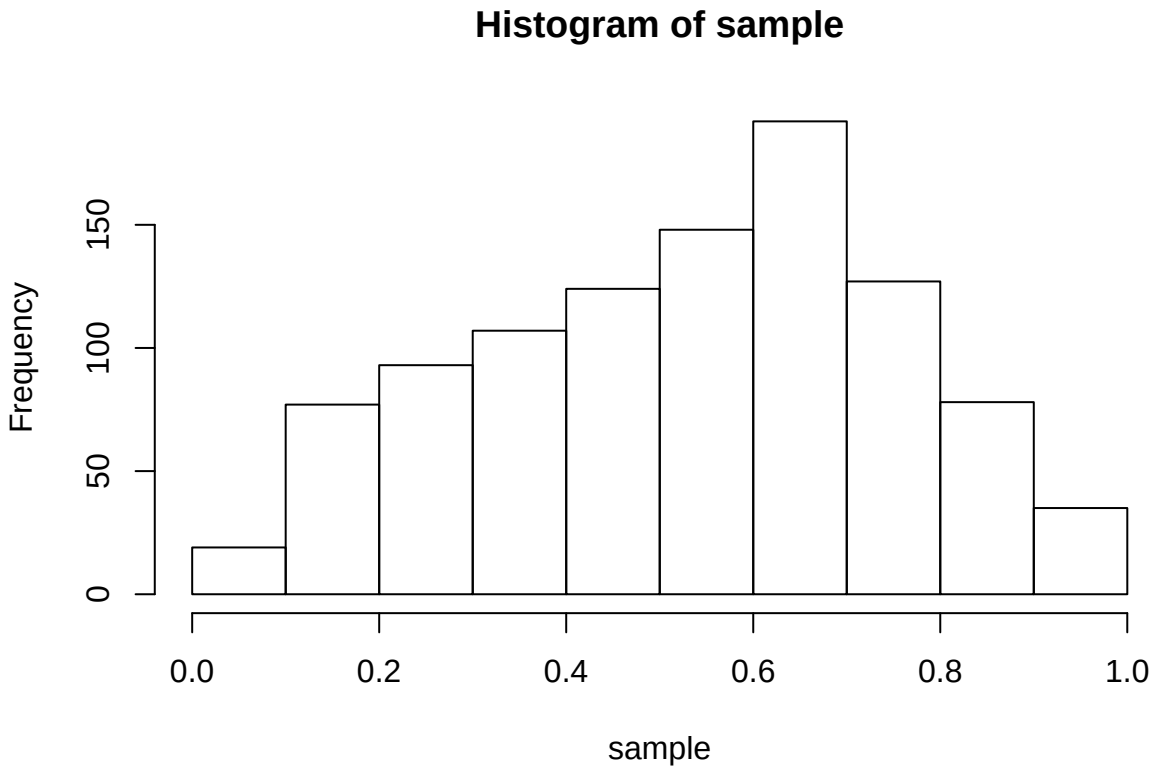
- まず適当な一様乱数からϵを生成し、次ステップの候補を$x' = x_i + \epsilon$とする。
- 次のような条件で実際に$x_{i+1} = x'$とするか、あるいは$x_{i+1} = x_i$のままにするかを決める。
- $r = f(x_{i+1})/f(x_i)$とする。
- rが1より大きい時、$x_{i+1} = x'$とする。
- rが1より小さい場合、確率rで$x_{i+1} = x'$にする。

・実際には、0から1の間の一様乱数qを発生させ、$r > q$なら$x_{i+1} = x'$とし$r < q$なら$x_{i+1} = x_i$とする。

```
p <- function(x){
  return(-6*x*(x-1))
}
x<-0:100/100
plot(x,p(x),'l')
```




```
step <- function(x){
  x_next <- x + runif(1, min=-1, max=1)#
  r <- p(x_next)/p(x)
  if(r > 1){
    return(x_next)
  }
  else{
    q <- runif(1)
    if(r>q) return(x_next)
    else return(x)
  }
}
sample <- c()
sample[1] <- runif(1)
N <- 1000
for(i in 1:(N-1)){
  sample[i+1] <- step(sample[i])
}
hist(sample)
```



Stanを用いたMCMCサンプリング

Stanの使用法

.stanファイルに分析するdata, parameter, modelを記述する。

実際に分析するデータや、分析結果の処理はRで行う。

実際にstanのコードとそれを用いた推定について、簡単なモデルから順番に見ていこう。

二項分布のパラメータ推定

コインを50回投げて28回表が出たとする。 このデータを元に、コインの表の出る確率pを推定する。

まずはstanファイルを用意して、以下のように記述する。

```
data{
  int D;  //
  int N;  //
}

parameters{
  real<lower=0,upper=1> p;  //
}

model{
  D ~ binomial(N,p);  //
}
```

次にRのスクリプトに以下のように記述する。

```
library(rstan)
```

```
## Loading required package: ggplot2
```

```
## Loading required package: StanHeaders
```

```
## rstan (Version 2.17.3, GitRev: 2e1f913d3ca3)
```

```
## For execution on a local, multicore CPU with excess RAM we recommend calling
## options(mc.cores = parallel::detectCores()).
## To avoid recompilation of unchanged Stan programs, we recommend calling
## rstan_options(auto_write = TRUE)
```

```
d <- list(D=28, N=50)
fit1 <- stan('binom1.stan', data=d)
```

```
##
## SAMPLING FOR MODEL 'd06ec3f2e0cf24bb2a3cdc911e8284d0' NOW (CHAIN 1).
##
## Gradient evaluation took 1.4e-05 seconds
## 1000 transitions using 10 leapfrog steps per transition would take 0.14 seconds.
## Adjust your expectations accordingly!
##
##
## Iteration:    1 / 2000 [  0%]  (Warmup)
## Iteration:  200 / 2000 [ 10%]  (Warmup)
## Iteration:  400 / 2000 [ 20%]  (Warmup)
## Iteration:  600 / 2000 [ 30%]  (Warmup)
## Iteration:  800 / 2000 [ 40%]  (Warmup)
## Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Iteration: 2000 / 2000 [100%]  (Sampling)
##
##  Elapsed Time: 0.013629 seconds (Warm-up)
##                0.012441 seconds (Sampling)
##                0.02607 seconds (Total)
##
##
## SAMPLING FOR MODEL 'd06ec3f2e0cf24bb2a3cdc911e8284d0' NOW (CHAIN 2).
##
## Gradient evaluation took 9e-06 seconds
## 1000 transitions using 10 leapfrog steps per transition would take 0.09 seconds.
## Adjust your expectations accordingly!
##
##
## Iteration:    1 / 2000 [  0%]  (Warmup)
## Iteration:  200 / 2000 [ 10%]  (Warmup)
## Iteration:  400 / 2000 [ 20%]  (Warmup)
## Iteration:  600 / 2000 [ 30%]  (Warmup)
## Iteration:  800 / 2000 [ 40%]  (Warmup)
## Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Iteration: 1001 / 2000 [ 50%]  (Sampling)
```

```
## Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Iteration: 2000 / 2000 [100%]  (Sampling)
##
##  Elapsed Time: 0.017234 seconds (Warm-up)
##                0.013015 seconds (Sampling)
##                0.030249 seconds (Total)
##
##
## SAMPLING FOR MODEL 'd06ec3f2e0cf24bb2a3cdc911e8284d0' NOW (CHAIN 3).
##
## Gradient evaluation took 5e-06 seconds
## 1000 transitions using 10 leapfrog steps per transition would take 0.05 seconds.
## Adjust your expectations accordingly!
##
##
## Iteration:    1 / 2000 [  0%]  (Warmup)
## Iteration:  200 / 2000 [ 10%]  (Warmup)
## Iteration:  400 / 2000 [ 20%]  (Warmup)
## Iteration:  600 / 2000 [ 30%]  (Warmup)
## Iteration:  800 / 2000 [ 40%]  (Warmup)
## Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Iteration: 2000 / 2000 [100%]  (Sampling)
##
##  Elapsed Time: 0.01506 seconds (Warm-up)
##                0.015339 seconds (Sampling)
##                0.030399 seconds (Total)
##
##
## SAMPLING FOR MODEL 'd06ec3f2e0cf24bb2a3cdc911e8284d0' NOW (CHAIN 4).
##
## Gradient evaluation took 4e-06 seconds
## 1000 transitions using 10 leapfrog steps per transition would take 0.04 seconds.
## Adjust your expectations accordingly!
##
##
## Iteration:    1 / 2000 [  0%]  (Warmup)
## Iteration:  200 / 2000 [ 10%]  (Warmup)
## Iteration:  400 / 2000 [ 20%]  (Warmup)
## Iteration:  600 / 2000 [ 30%]  (Warmup)
## Iteration:  800 / 2000 [ 40%]  (Warmup)
## Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Iteration: 1800 / 2000 [ 90%]  (Sampling)
```

```
## Iteration: 2000 / 2000 [100%]  (Sampling)
##
##  Elapsed Time: 0.012515 seconds (Warm-up)
##                0.011591 seconds (Sampling)
##                0.024106 seconds (Total)
```

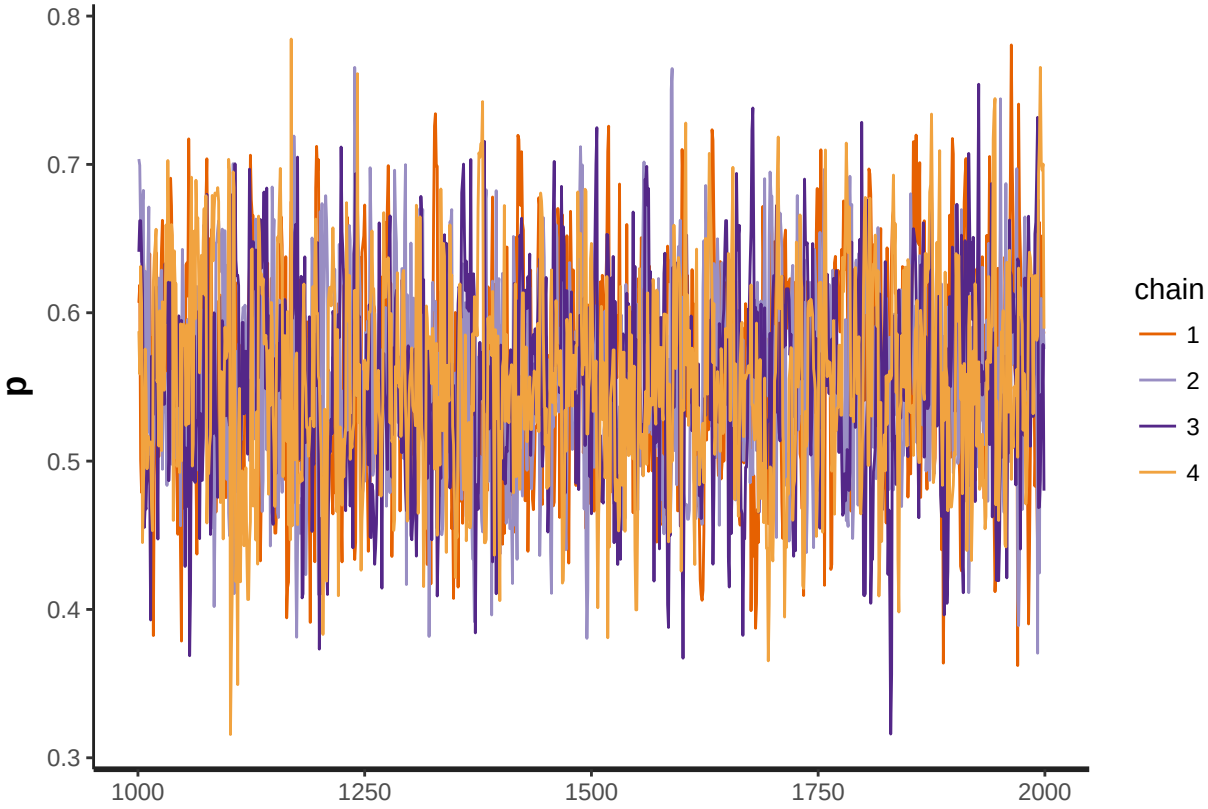
```
fit1
```

```
## Inference for Stan model: d06ec3f2e0cf24bb2a3cdc911e8284d0.
## 4 chains, each with iter=2000; warmup=1000; thin=1;
## post-warmup draws per chain=1000, total post-warmup draws=4000.
##
##        mean se_mean   sd   2.5%    25%    50%    75%  97.5% n_eff Rhat
## p      0.56    0.00 0.07   0.43   0.51   0.56   0.61   0.69  1407    1
## lp__ -36.21    0.02 0.71 -38.18 -36.39 -35.95 -35.75 -35.70  1645    1
##
## Samples were drawn using NUTS(diag_e) at Mon Jul  2 10:49:05 2018.
## 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).
```

推定結果は以下のようにみることができる。

パラメータのサンプリング、横軸がステップ数

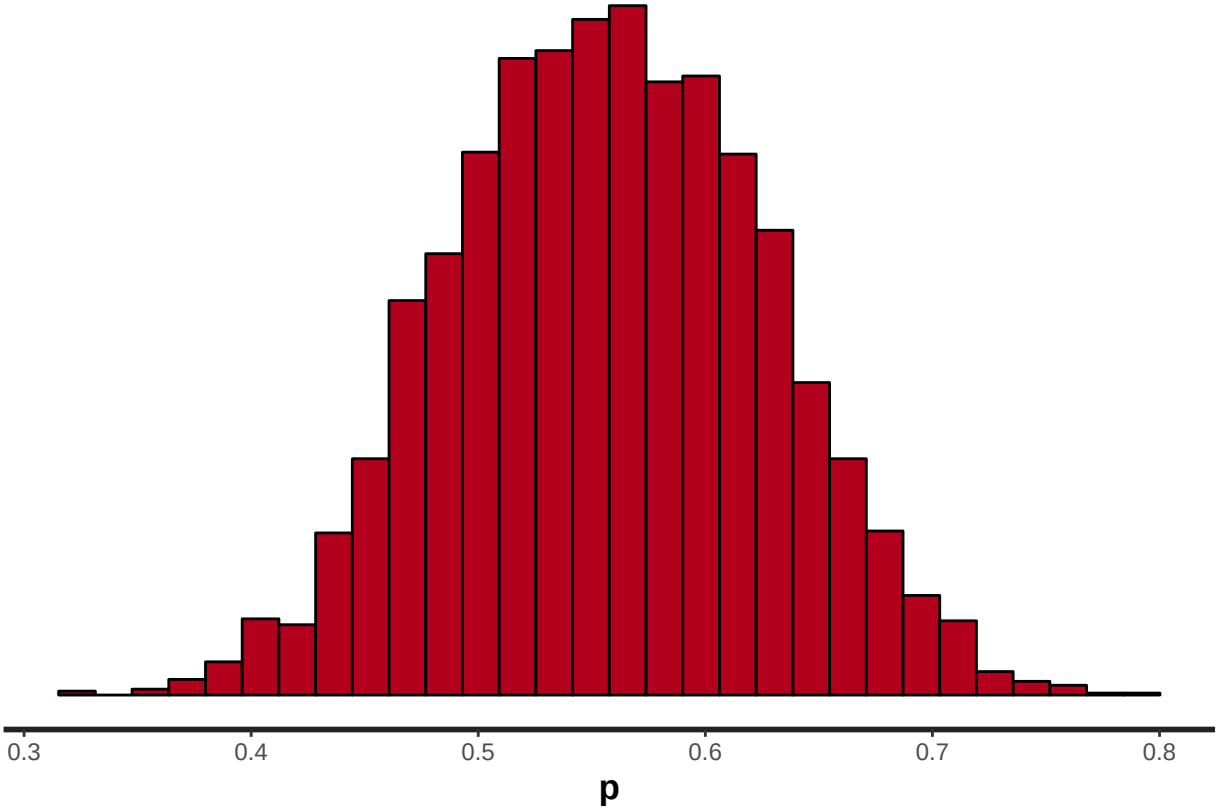
```
stan_trace(fit1,pars="p")
```



パラメータのサンプルのヒストグラム

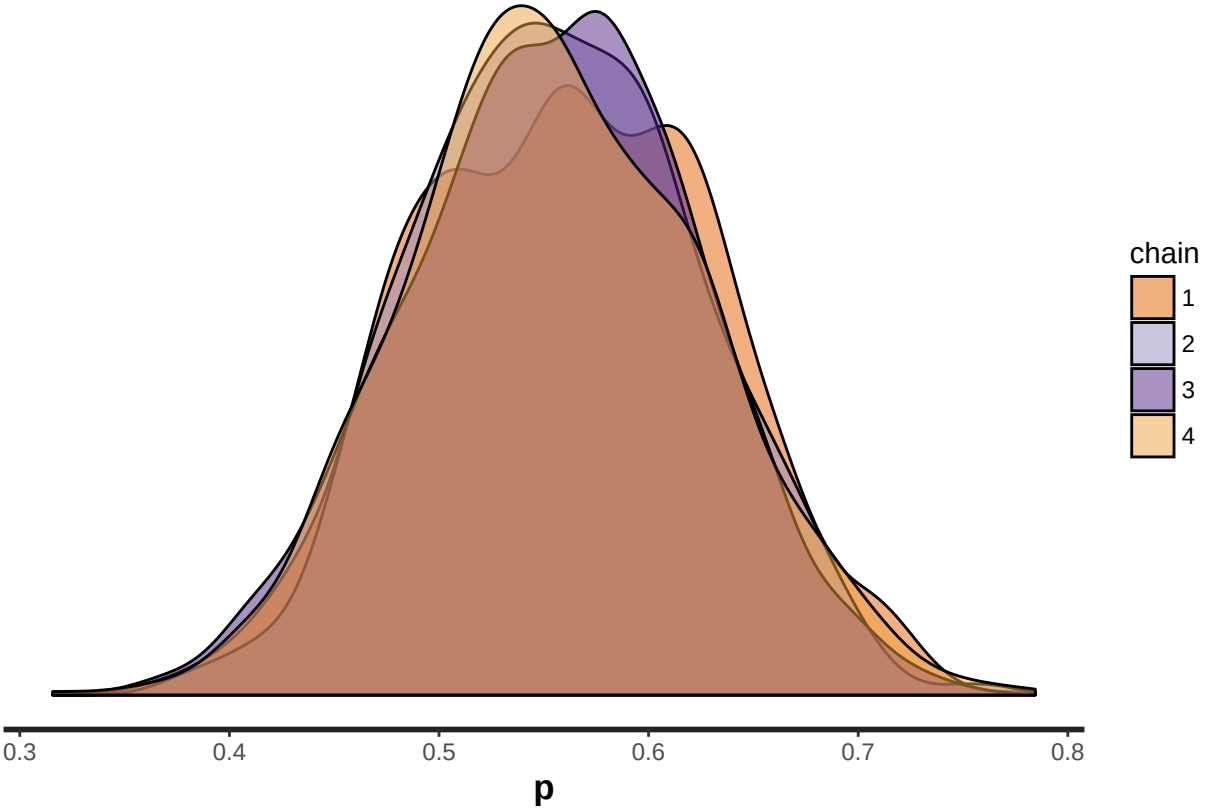
```
stan_hist(fit1,pars="p")
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



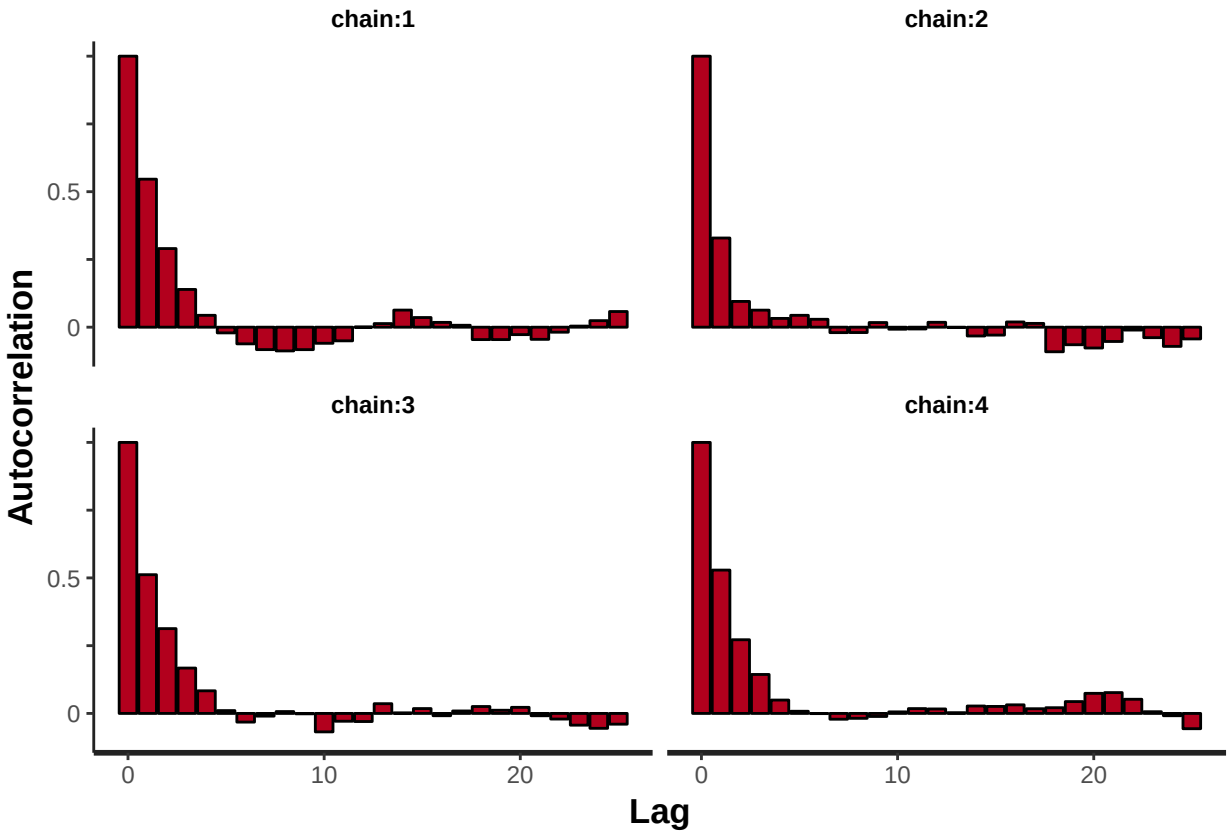
パラメータ事後分布の密度推定

```
stan_dens(fit1,pars="p",separate_chains = TRUE)
```



パラメータの自己相関

```
stan_ac(fit1,pars = "p",separate_chains = TRUE)
```



収束の判断。

- Rhatが1に近い（1.05以下が一つの目安）
- チェインごとの確率密度が重なっている
- 自己相関が低い。定常分布に収束したら相関はないはず

サンプリングした上で平均など色々な関数の値を計算することができる。 またそれを用いて未知のデータについて予測できる。

例えば次にコインを投げて表の出る確率は、以下のように予測できる。

```
p <- rstan::extract(fit1)$p
mean(p)
```

```
## [1] 0.5570509
```

ポアソン分布のパラメータ推定

次にポアソン分布のパラメータ推定をしてみよう。

前に扱った事故件数のデータを用いる。

まず.stanファイルにモデルについて記述する。

```
data{
  int N;  //
  int x[N];  //
}

parameters{
  real<lower=0> lambda;  //
}
```

```
model{
  for (i in 1:N){
    x[i] ~ poisson(lambda);  //
    }
}
```

次にデータを用意し、Rで分析する。

```
x <- c(8,4,5,5,7,9,7,5,5,4)
d <- list(x=x, N=length(x))
fit2 <- stan('poisson1.stan', data=d)
```

```
##
## SAMPLING FOR MODEL 'd9d5669abb02a1e798ca9ce33cd34eb4' NOW (CHAIN 1).
##
## Gradient evaluation took 1.5e-05 seconds
## 1000 transitions using 10 leapfrog steps per transition would take 0.15 seconds.
## Adjust your expectations accordingly!
##
##
## Iteration:    1 / 2000 [  0%]  (Warmup)
## Iteration:  200 / 2000 [ 10%]  (Warmup)
## Iteration:  400 / 2000 [ 20%]  (Warmup)
## Iteration:  600 / 2000 [ 30%]  (Warmup)
## Iteration:  800 / 2000 [ 40%]  (Warmup)
## Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Iteration: 2000 / 2000 [100%]  (Sampling)
##
##  Elapsed Time: 0.017984 seconds (Warm-up)
##                0.021672 seconds (Sampling)
##                0.039656 seconds (Total)
##
##
## SAMPLING FOR MODEL 'd9d5669abb02a1e798ca9ce33cd34eb4' NOW (CHAIN 2).
##
## Gradient evaluation took 1e-05 seconds
## 1000 transitions using 10 leapfrog steps per transition would take 0.1 seconds.
## Adjust your expectations accordingly!
##
##
## Iteration:    1 / 2000 [  0%]  (Warmup)
## Iteration:  200 / 2000 [ 10%]  (Warmup)
## Iteration:  400 / 2000 [ 20%]  (Warmup)
## Iteration:  600 / 2000 [ 30%]  (Warmup)
## Iteration:  800 / 2000 [ 40%]  (Warmup)
## Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Iteration: 1400 / 2000 [ 70%]  (Sampling)
```

```
## Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Iteration: 2000 / 2000 [100%]  (Sampling)
##
##  Elapsed Time: 0.018555 seconds (Warm-up)
##                0.023517 seconds (Sampling)
##                0.042072 seconds (Total)
##
##
## SAMPLING FOR MODEL 'd9d5669abb02a1e798ca9ce33cd34eb4' NOW (CHAIN 3).
##
## Gradient evaluation took 6e-06 seconds
## 1000 transitions using 10 leapfrog steps per transition would take 0.06 seconds.
## Adjust your expectations accordingly!
##
##
## Iteration:    1 / 2000 [  0%]  (Warmup)
## Iteration:  200 / 2000 [ 10%]  (Warmup)
## Iteration:  400 / 2000 [ 20%]  (Warmup)
## Iteration:  600 / 2000 [ 30%]  (Warmup)
## Iteration:  800 / 2000 [ 40%]  (Warmup)
## Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Iteration: 2000 / 2000 [100%]  (Sampling)
##
##  Elapsed Time: 0.020717 seconds (Warm-up)
##                0.02489 seconds (Sampling)
##                0.045607 seconds (Total)
##
##
## SAMPLING FOR MODEL 'd9d5669abb02a1e798ca9ce33cd34eb4' NOW (CHAIN 4).
##
## Gradient evaluation took 8e-06 seconds
## 1000 transitions using 10 leapfrog steps per transition would take 0.08 seconds.
## Adjust your expectations accordingly!
##
##
## Iteration:    1 / 2000 [  0%]  (Warmup)
## Iteration:  200 / 2000 [ 10%]  (Warmup)
## Iteration:  400 / 2000 [ 20%]  (Warmup)
## Iteration:  600 / 2000 [ 30%]  (Warmup)
## Iteration:  800 / 2000 [ 40%]  (Warmup)
## Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Iteration: 2000 / 2000 [100%]  (Sampling)
##
```

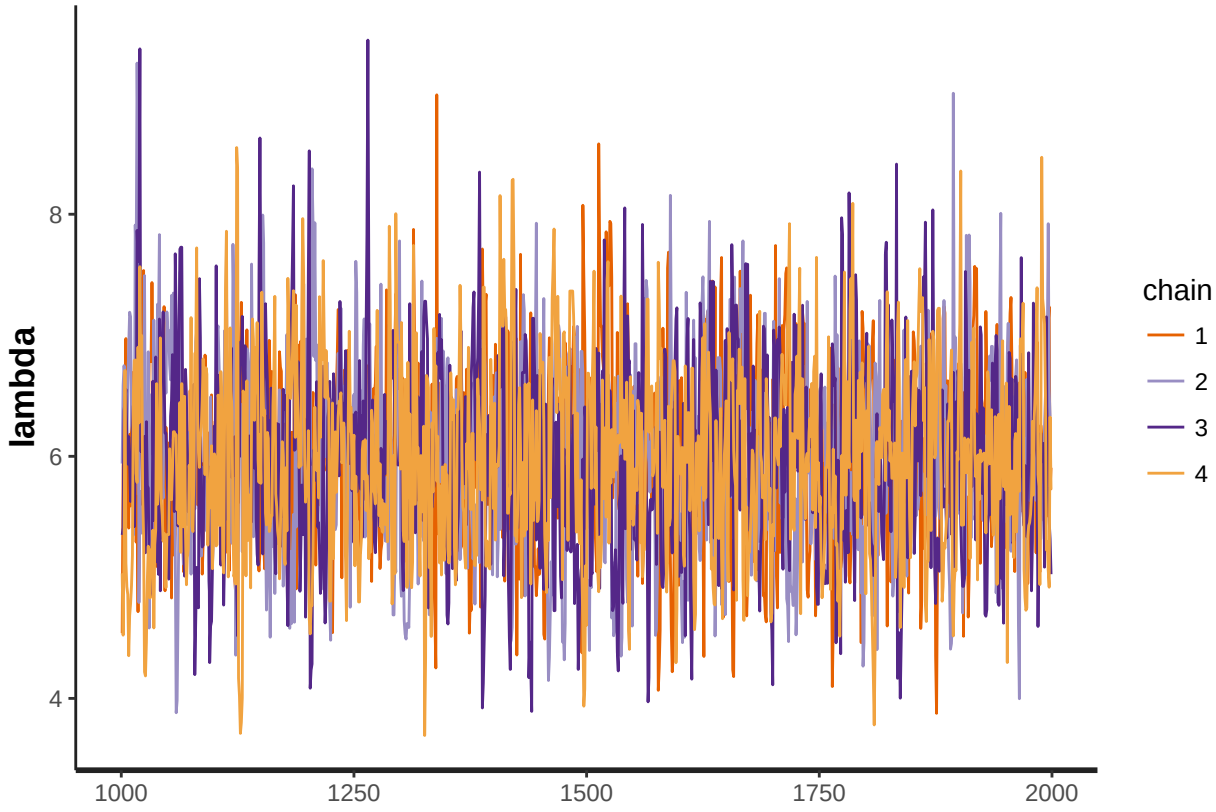
```
##  Elapsed Time: 0.018755 seconds (Warm-up)
##                0.018029 seconds (Sampling)
##                0.036784 seconds (Total)
```

```
fit2
```

```
## Inference for Stan model: d9d5669abb02a1e798ca9ce33cd34eb4.
## 4 chains, each with iter=2000; warmup=1000; thin=1;
## post-warmup draws per chain=1000, total post-warmup draws=4000.
##
##         mean se_mean   sd  2.5%   25%   50%   75% 97.5% n_eff Rhat
## lambda  5.99    0.02 0.78  4.54  5.44  5.97  6.52  7.54  1753    1
## lp__   47.00    0.02 0.73 44.95 46.83 47.27 47.45 47.51  1628    1
##
## Samples were drawn using NUTS(diag_e) at Mon Jul  2 10:50:18 2018.
## 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).
```

パラメータのサンプリング、横軸がステップ数

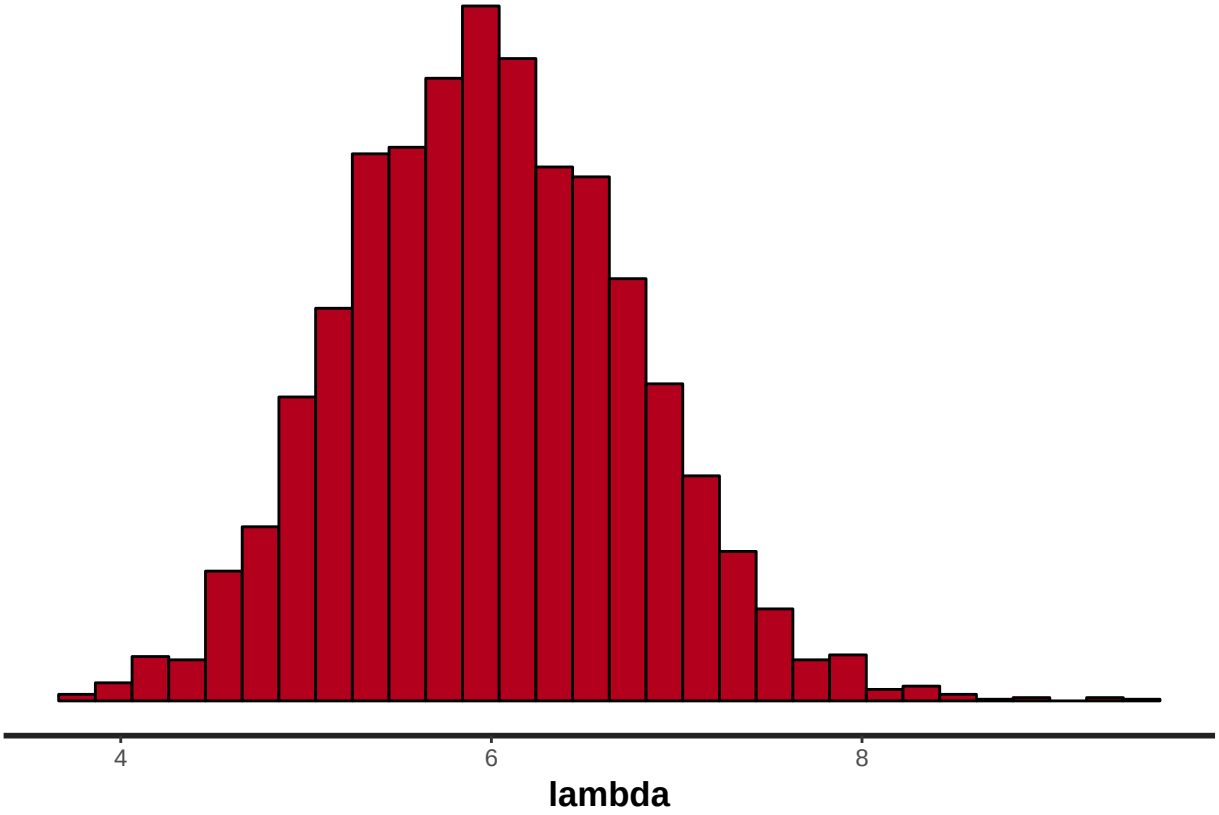
```
stan_trace(fit2,pars="lambda")
```



パラメータのサンプルのヒストグラム

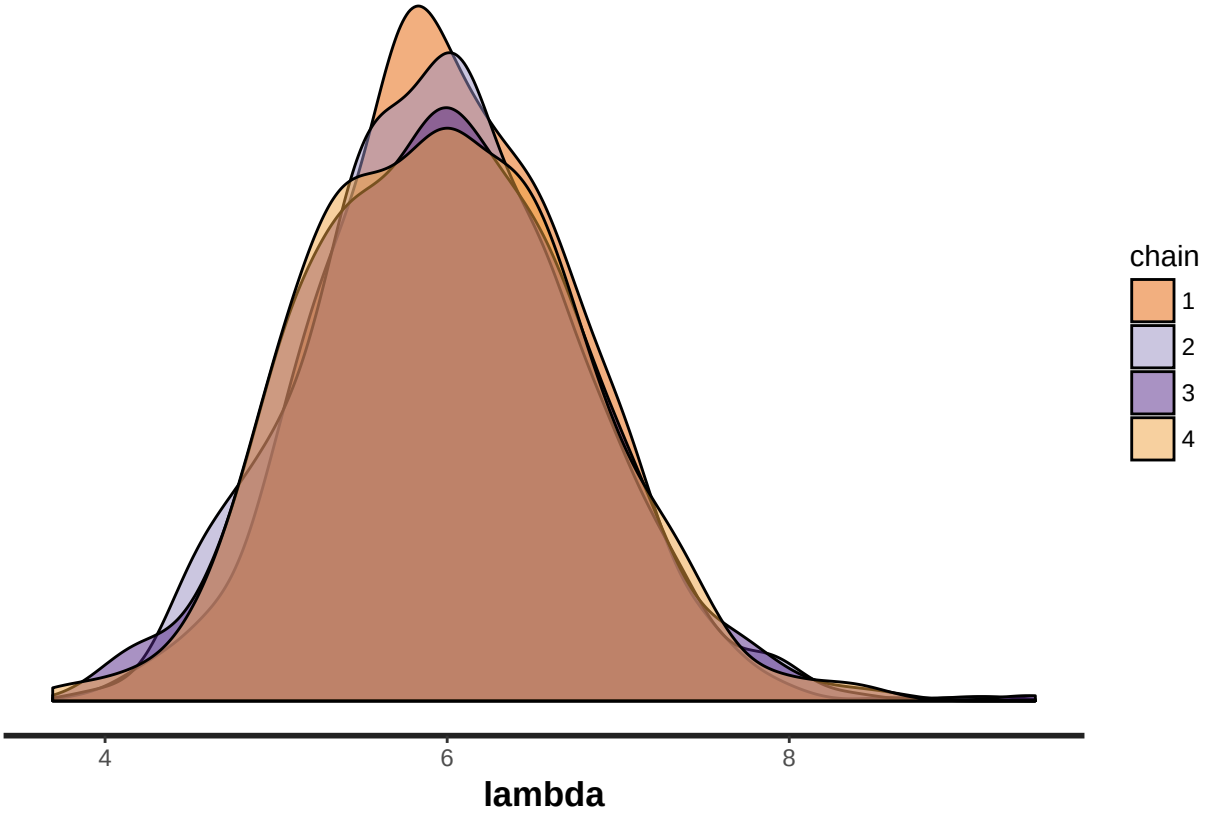
```
stan_hist(fit2,pars="lambda")
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



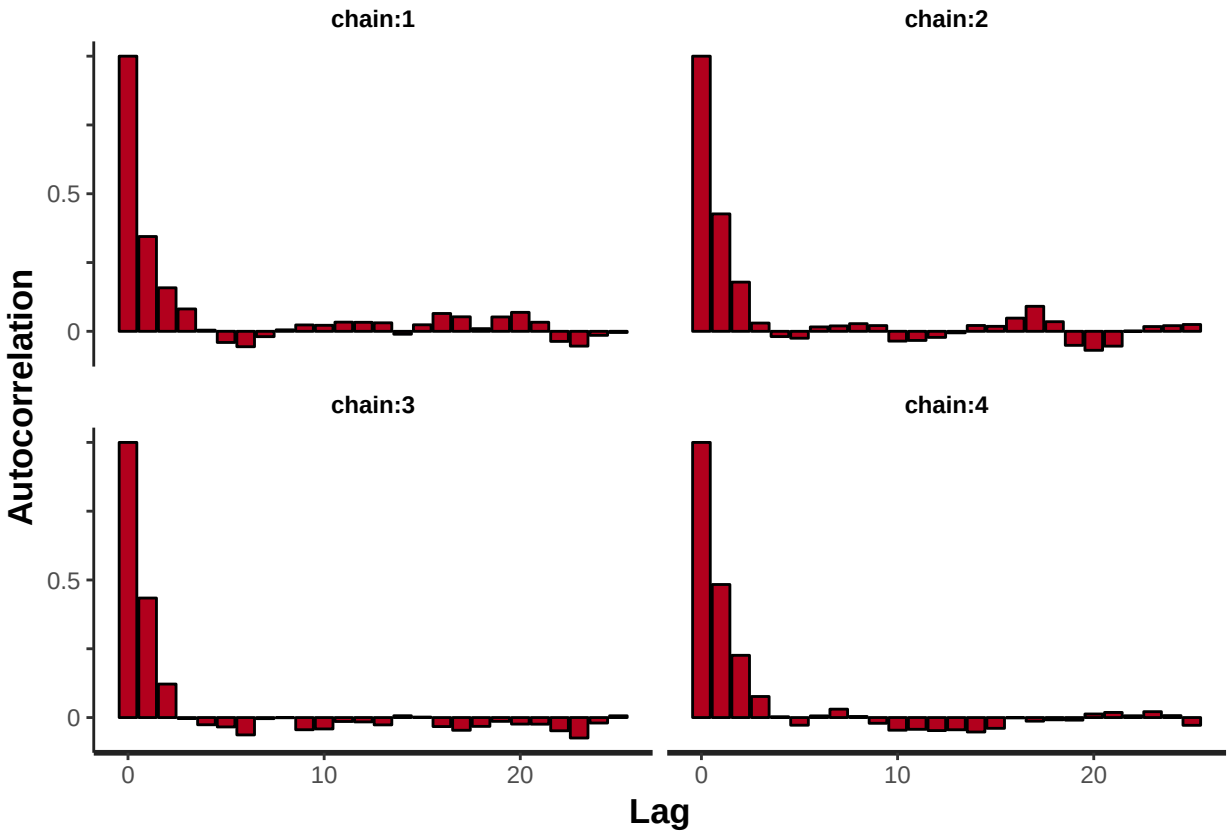
パラメータ事後分布の密度推定

```
stan_dens(fit2,pars="lambda",separate_chains = TRUE)
```



パラメータの自己相関

```
stan_ac(fit2,pars = "lambda",separate_chains = TRUE)
```



事故件数の平均の予測値は以下のようになる。

```
lambda <- rstan::extract(fit2)$lambda
mean(lambda)
```

```
## [1] 5.993912
```

正規分布パラメータの推定

次に正規分布のパラメータを推定してみよう。

前と同じ、製品の規格検査のデータを使おう。

```
data{
  int N;      //
  real y[N];  //
}

parameters{
  real mu;    //
  real<lower=0> sigma; //
}

model{
  mu ~ normal(0,100);   //
  sigma ~ cauchy(0,5);  //
  for(i in 1:N){
    y[i] ~ normal(mu,sigma);
  }
```

```
}
```

```
y <- c(102.3,100.4,99.3,100.2,100.3,99.4,101.5,99.3,98.9,100.1)
d <- list(y=y,N=length(y))
fit3 <- stan(file = "normal1.stan", data = d)
```

```
##
## SAMPLING FOR MODEL 'c81f4001bf9438379267dee6a2a4f8a2' NOW (CHAIN 1).
##
## Gradient evaluation took 1.6e-05 seconds
## 1000 transitions using 10 leapfrog steps per transition would take 0.16 seconds.
## Adjust your expectations accordingly!
##
##
## Iteration:    1 / 2000 [  0%]  (Warmup)
## Iteration:  200 / 2000 [ 10%]  (Warmup)
## Iteration:  400 / 2000 [ 20%]  (Warmup)
## Iteration:  600 / 2000 [ 30%]  (Warmup)
## Iteration:  800 / 2000 [ 40%]  (Warmup)
## Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Iteration: 2000 / 2000 [100%]  (Sampling)
##
##  Elapsed Time: 0.032047 seconds (Warm-up)
##                0.024745 seconds (Sampling)
##                0.056792 seconds (Total)
##
##
## SAMPLING FOR MODEL 'c81f4001bf9438379267dee6a2a4f8a2' NOW (CHAIN 2).
##
## Gradient evaluation took 6e-06 seconds
## 1000 transitions using 10 leapfrog steps per transition would take 0.06 seconds.
## Adjust your expectations accordingly!
##
##
## Iteration:    1 / 2000 [  0%]  (Warmup)
## Iteration:  200 / 2000 [ 10%]  (Warmup)
## Iteration:  400 / 2000 [ 20%]  (Warmup)
## Iteration:  600 / 2000 [ 30%]  (Warmup)
## Iteration:  800 / 2000 [ 40%]  (Warmup)
## Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Iteration: 2000 / 2000 [100%]  (Sampling)
##
##  Elapsed Time: 0.02746 seconds (Warm-up)
##                0.029602 seconds (Sampling)
##                0.057062 seconds (Total)
```

```
##
##
## SAMPLING FOR MODEL 'c81f4001bf9438379267dee6a2a4f8a2' NOW (CHAIN 3).
##
## Gradient evaluation took 5e-06 seconds
## 1000 transitions using 10 leapfrog steps per transition would take 0.05 seconds.
## Adjust your expectations accordingly!
##
##
## Iteration:    1 / 2000 [  0%]  (Warmup)
## Iteration:  200 / 2000 [ 10%]  (Warmup)
## Iteration:  400 / 2000 [ 20%]  (Warmup)
## Iteration:  600 / 2000 [ 30%]  (Warmup)
## Iteration:  800 / 2000 [ 40%]  (Warmup)
## Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Iteration: 2000 / 2000 [100%]  (Sampling)
##
##  Elapsed Time: 0.025804 seconds (Warm-up)
##                0.026558 seconds (Sampling)
##                0.052362 seconds (Total)
##
##
## SAMPLING FOR MODEL 'c81f4001bf9438379267dee6a2a4f8a2' NOW (CHAIN 4).
##
## Gradient evaluation took 6e-06 seconds
## 1000 transitions using 10 leapfrog steps per transition would take 0.06 seconds.
## Adjust your expectations accordingly!
##
##
## Iteration:    1 / 2000 [  0%]  (Warmup)
## Iteration:  200 / 2000 [ 10%]  (Warmup)
## Iteration:  400 / 2000 [ 20%]  (Warmup)
## Iteration:  600 / 2000 [ 30%]  (Warmup)
## Iteration:  800 / 2000 [ 40%]  (Warmup)
## Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Iteration: 2000 / 2000 [100%]  (Sampling)
##
##  Elapsed Time: 0.029466 seconds (Warm-up)
##                0.031164 seconds (Sampling)
##                0.06063 seconds (Total)
```

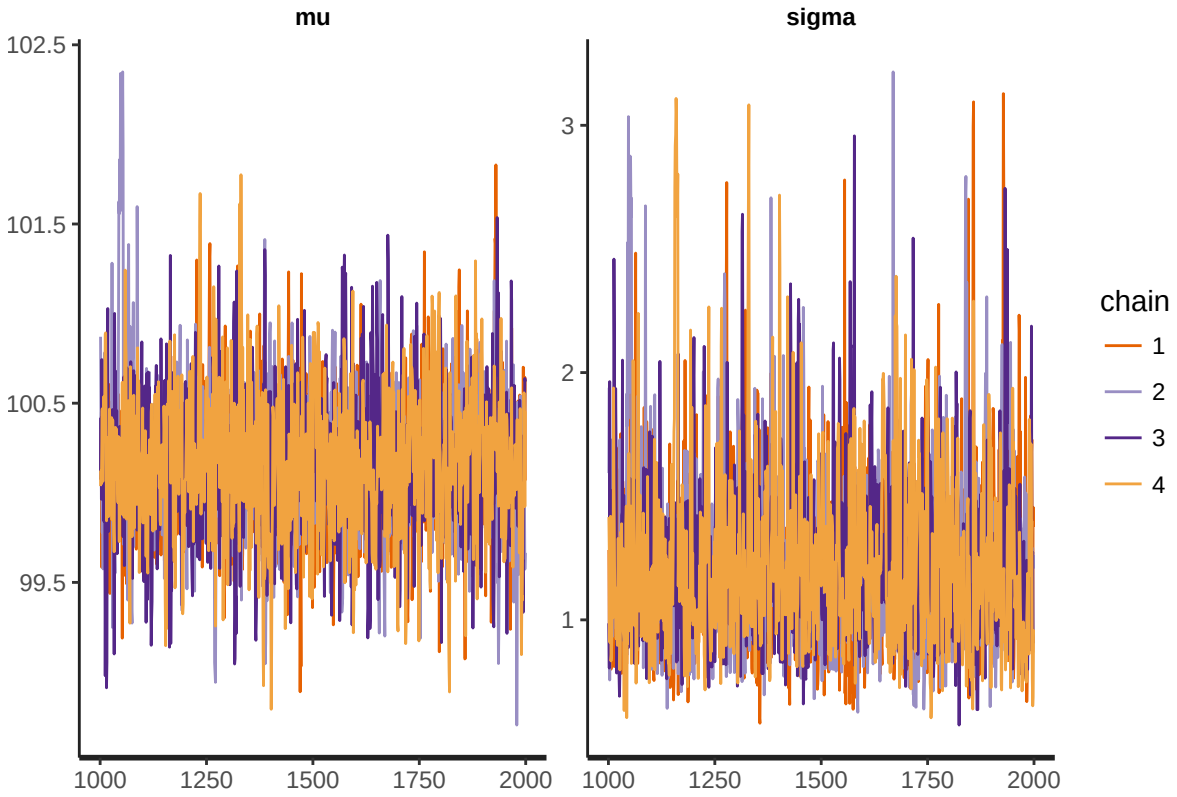
```
fit3
```

```
## Inference for Stan model: c81f4001bf9438379267dee6a2a4f8a2.
## 4 chains, each with iter=2000; warmup=1000; thin=1;
```

```
## post-warmup draws per chain=1000, total post-warmup draws=4000.
##
##          mean se_mean   sd  2.5%   25%    50%    75%  97.5% n_eff Rhat
## mu     100.17    0.01 0.40 99.40 99.92 100.16 100.42 100.97  1770    1
## sigma    1.21    0.01 0.34  0.75  0.99   1.15   1.37   2.05  1446    1
## lp__    -6.65    0.03 1.09 -9.58 -7.05  -6.33  -5.88  -5.59  1166    1
##
## Samples were drawn using NUTS(diag_e) at Mon Jul  2 10:51:24 2018.
## 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).
```

パラメータのサンプリング、横軸がステップ数

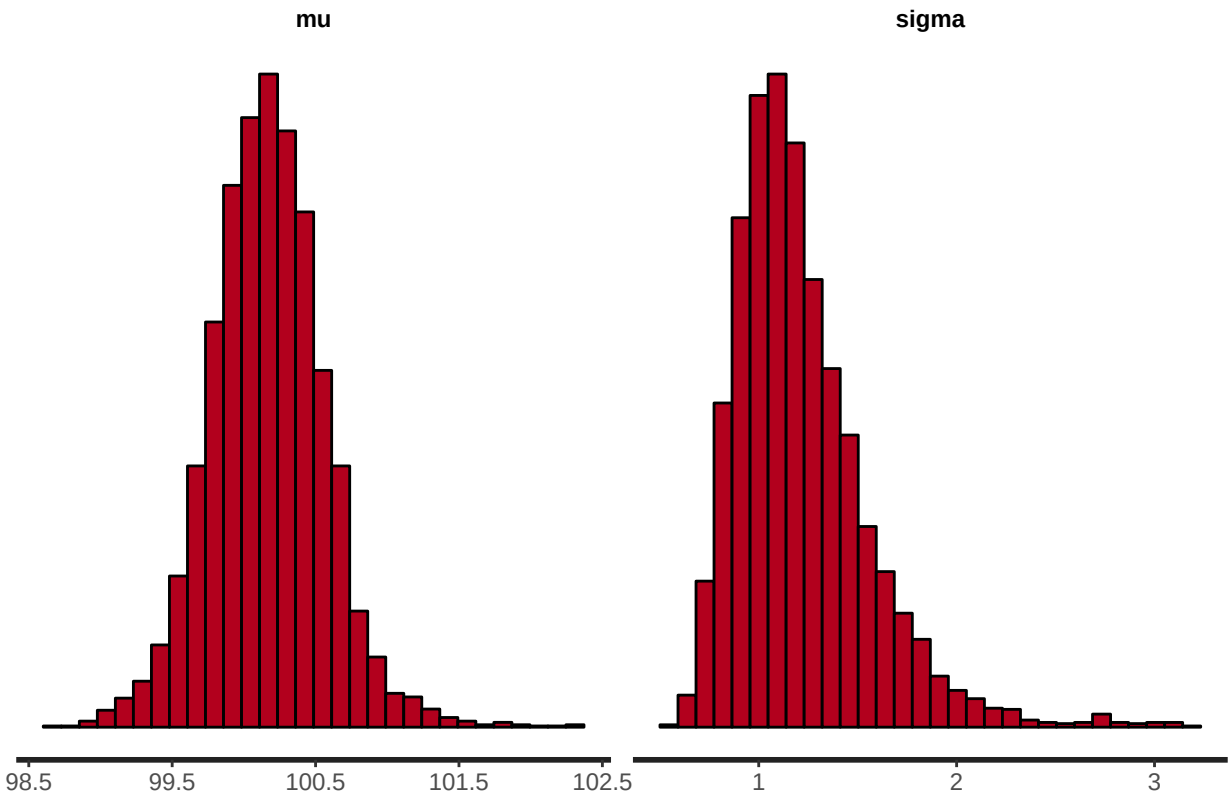
```
stan_trace(fit3, pars=c("mu","sigma"))
```



パラメータのサンプルのヒストグラム

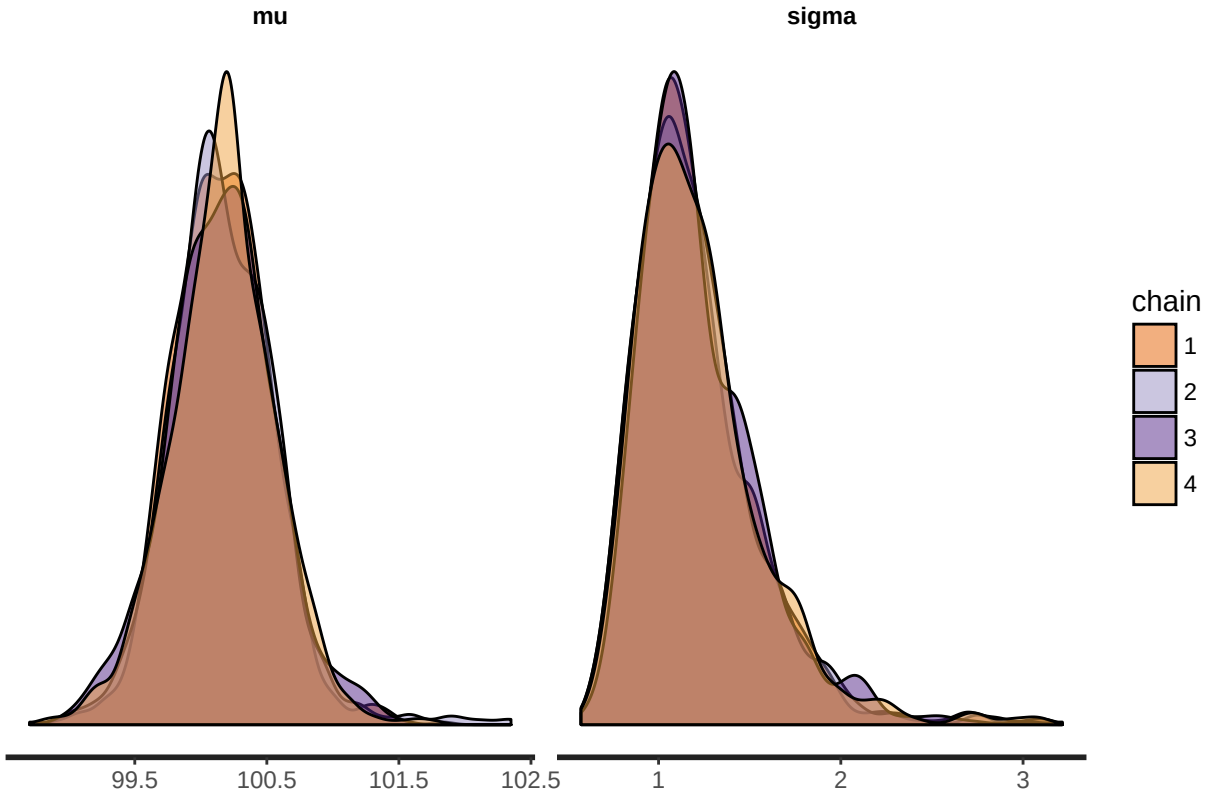
```
stan_hist(fit3,pars=c("mu","sigma"))
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



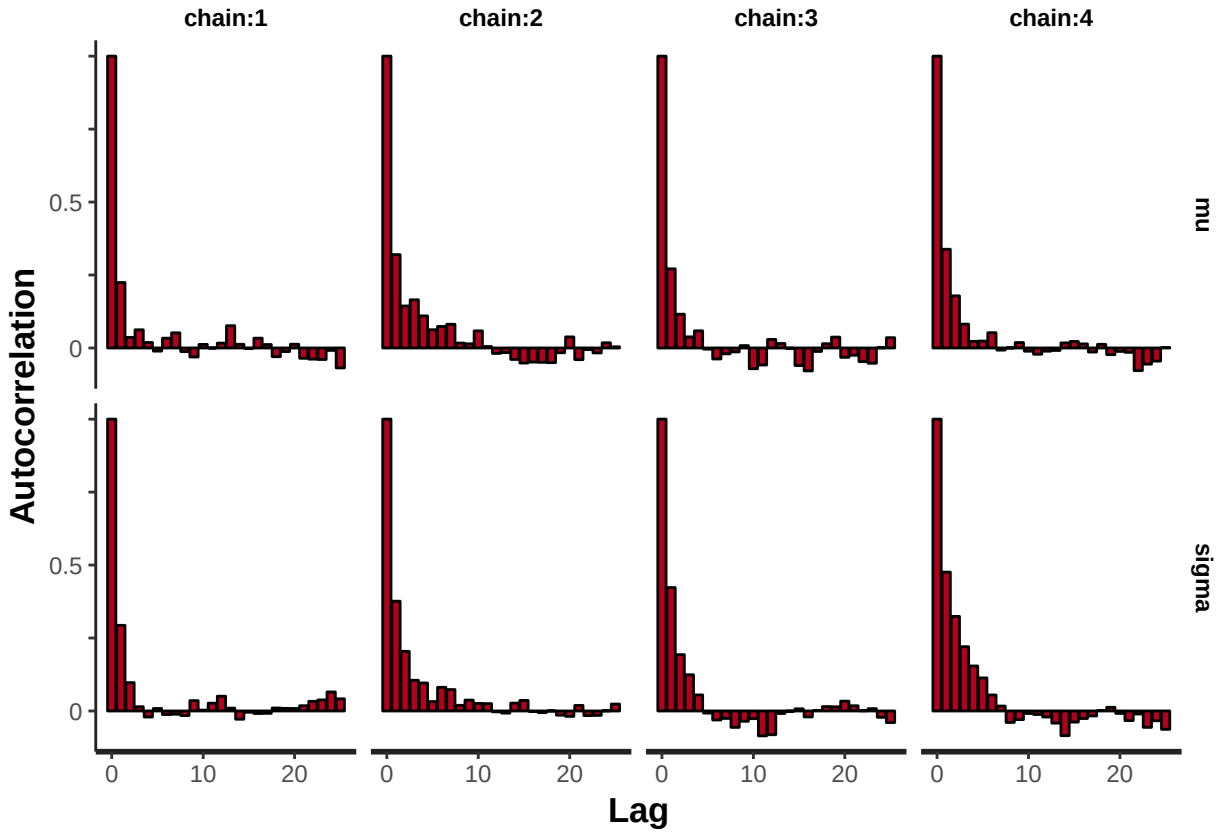
パラメータ事後分布の密度推定

```
stan_dens(fit3, pars=c("mu","sigma"),separate_chains = TRUE)
```



パラメータの自己相関

```
stan_ac(fit3,pars = c("mu","sigma"),separate_chains = TRUE)
```



t検定

上の正規分布のパラメータの推定の応用で、 二つのグループの母平均を比較してみよう。

次のように、二つのクラスでテストをした結果が得られたとする。 このニクラスの平均点に差があると言えるかどうか、ベイズ推定しよう。

各クラスの点数は正規分布に従うと仮定し、 そのパラメータをそれぞれmu_x, sigma_xとmu_y, sigma_yとしてモデルを作る。

```
data{
  int N;  //
  real y[N];  // 1
  real x[N];  // 2
}

parameters{
  real mu_y;  //y
  real<lower=0> sigma_y;  //y
  real mu_x;  //x
  real<lower=0> sigma_x;  //x
}

model{
  mu_y ~ normal(0,100); //
  sigma_y ~ cauchy(0,5);//
  mu_x ~ normal(0,100); //
```

```
  sigma_x ~ cauchy(0,5);//

  y ~ normal(mu_y,sigma_y);
  x ~ normal(mu_x,sigma_x);
}

generated quantities{
  real diff;  //2
  diff = mu_x-mu_y;
}
```

```
library(rstan)
N<-10
y<-c(60,39,75,64,36,45,38,28,3,9)
x<-c(73,67,19,100,100,0,58,39,84,62)

d<-list(N=N, x=x, y=y)
fit <- stan(file = "ttest.stan", data = d)
```

```
##
## SAMPLING FOR MODEL 'b6f16321d07933d8582b0bc79e69a9e1' NOW (CHAIN 1).
##
## Gradient evaluation took 1.2e-05 seconds
## 1000 transitions using 10 leapfrog steps per transition would take 0.12 seconds.
## Adjust your expectations accordingly!
##
##
## Iteration:    1 / 2000 [  0%]  (Warmup)
## Iteration:  200 / 2000 [ 10%]  (Warmup)
## Iteration:  400 / 2000 [ 20%]  (Warmup)
## Iteration:  600 / 2000 [ 30%]  (Warmup)
## Iteration:  800 / 2000 [ 40%]  (Warmup)
## Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Iteration: 2000 / 2000 [100%]  (Sampling)
##
##  Elapsed Time: 0.07221 seconds (Warm-up)
##                0.033201 seconds (Sampling)
##                0.105411 seconds (Total)
##
##
## SAMPLING FOR MODEL 'b6f16321d07933d8582b0bc79e69a9e1' NOW (CHAIN 2).
##
## Gradient evaluation took 7e-06 seconds
## 1000 transitions using 10 leapfrog steps per transition would take 0.07 seconds.
## Adjust your expectations accordingly!
##
##
## Iteration:    1 / 2000 [  0%]  (Warmup)
## Iteration:  200 / 2000 [ 10%]  (Warmup)
## Iteration:  400 / 2000 [ 20%]  (Warmup)
```

```
## Iteration:  600 / 2000 [ 30%]  (Warmup)
## Iteration:  800 / 2000 [ 40%]  (Warmup)
## Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Iteration: 2000 / 2000 [100%]  (Sampling)
##
##  Elapsed Time: 0.064086 seconds (Warm-up)
##                0.029029 seconds (Sampling)
##                0.093115 seconds (Total)
##
##
## SAMPLING FOR MODEL 'b6f16321d07933d8582b0bc79e69a9e1' NOW (CHAIN 3).
##
## Gradient evaluation took 1.1e-05 seconds
## 1000 transitions using 10 leapfrog steps per transition would take 0.11 seconds.
## Adjust your expectations accordingly!
##
##
## Iteration:    1 / 2000 [  0%]  (Warmup)
## Iteration:  200 / 2000 [ 10%]  (Warmup)
## Iteration:  400 / 2000 [ 20%]  (Warmup)
## Iteration:  600 / 2000 [ 30%]  (Warmup)
## Iteration:  800 / 2000 [ 40%]  (Warmup)
## Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Iteration: 2000 / 2000 [100%]  (Sampling)
##
##  Elapsed Time: 0.090028 seconds (Warm-up)
##                0.032222 seconds (Sampling)
##                0.12225 seconds (Total)
##
##
## SAMPLING FOR MODEL 'b6f16321d07933d8582b0bc79e69a9e1' NOW (CHAIN 4).
##
## Gradient evaluation took 7e-06 seconds
## 1000 transitions using 10 leapfrog steps per transition would take 0.07 seconds.
## Adjust your expectations accordingly!
##
##
## Iteration:    1 / 2000 [  0%]  (Warmup)
## Iteration:  200 / 2000 [ 10%]  (Warmup)
## Iteration:  400 / 2000 [ 20%]  (Warmup)
## Iteration:  600 / 2000 [ 30%]  (Warmup)
## Iteration:  800 / 2000 [ 40%]  (Warmup)
## Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Iteration: 1001 / 2000 [ 50%]  (Sampling)
```

```
## Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Iteration: 2000 / 2000 [100%]  (Sampling)
##
##  Elapsed Time: 0.075548 seconds (Warm-up)
##                0.031717 seconds (Sampling)
##                0.107265 seconds (Total)
```

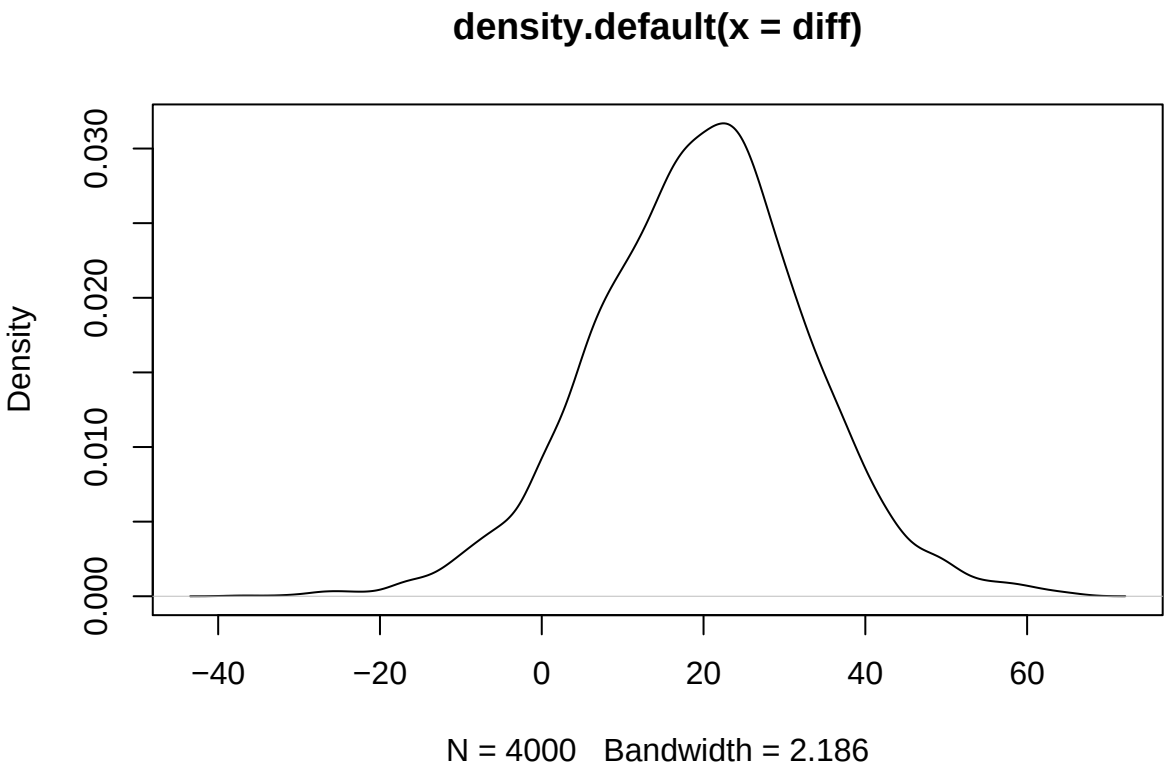
```
fit
```

```
## Inference for Stan model: b6f16321d07933d8582b0bc79e69a9e1.
## 4 chains, each with iter=2000; warmup=1000; thin=1;
## post-warmup draws per chain=1000, total post-warmup draws=4000.
##
##           mean se_mean    sd   2.5%    25%    50%    75%  97.5% n_eff Rhat
## mu_y     39.51    0.14  7.79  23.85  34.62  39.50  44.31  54.75  3088    1
## sigma_y  23.72    0.11  5.89  15.33  19.60  22.66  26.58  38.35  2885    1
## mu_x     59.23    0.19 10.94  37.27  52.47  59.48  65.97  80.82  3480    1
## sigma_x  34.01    0.17  8.80  21.95  27.99  32.43  38.11  56.03  2834    1
## diff     19.72    0.23 13.34  -7.53  11.12  20.10  28.22  46.27  3288    1
## lp__    -77.76    0.04  1.57 -81.63 -78.55 -77.41 -76.57 -75.81  1539    1
##
## Samples were drawn using NUTS(diag_e) at Mon Jul  2 10:52:34 2018.
## 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).
```

```
diff <- rstan::extract(fit)$diff
p <- sum(ifelse(diff>0,1,0))/length(diff)
p
```

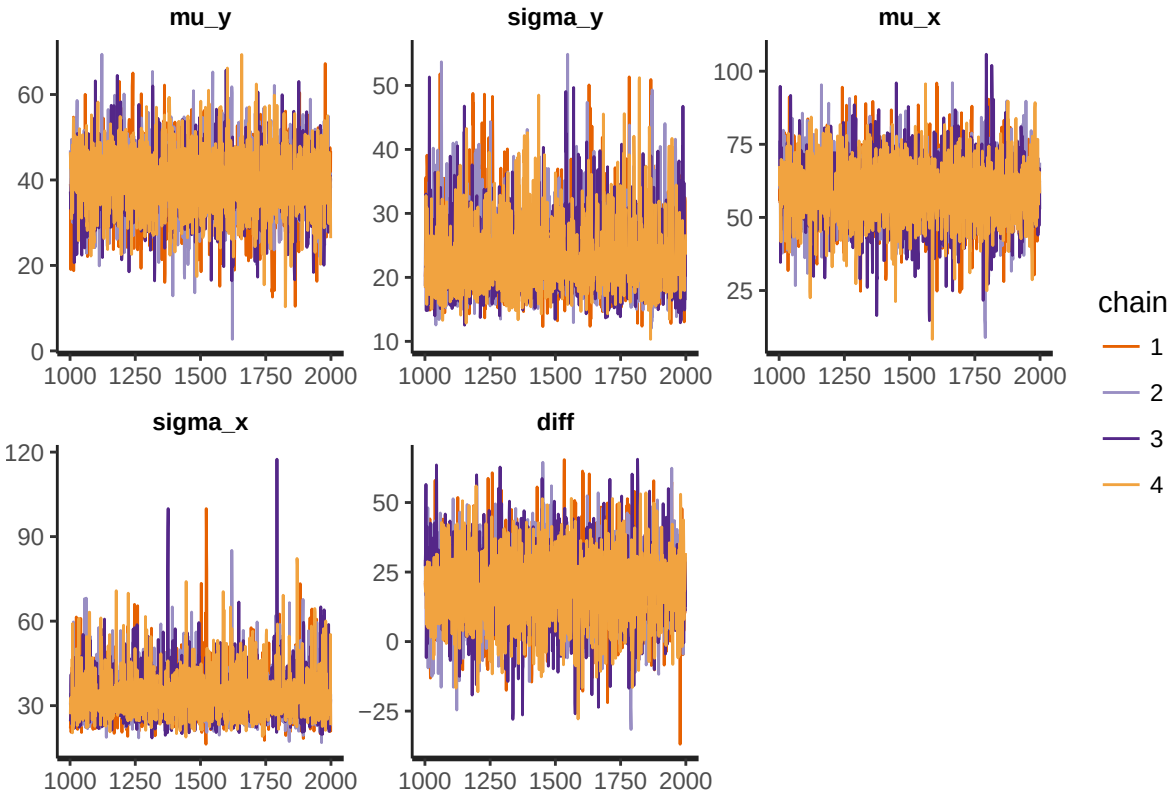
```
## [1] 0.933
```

```
plot(density(diff))
```



パラメータのサンプリング、横軸がステップ数

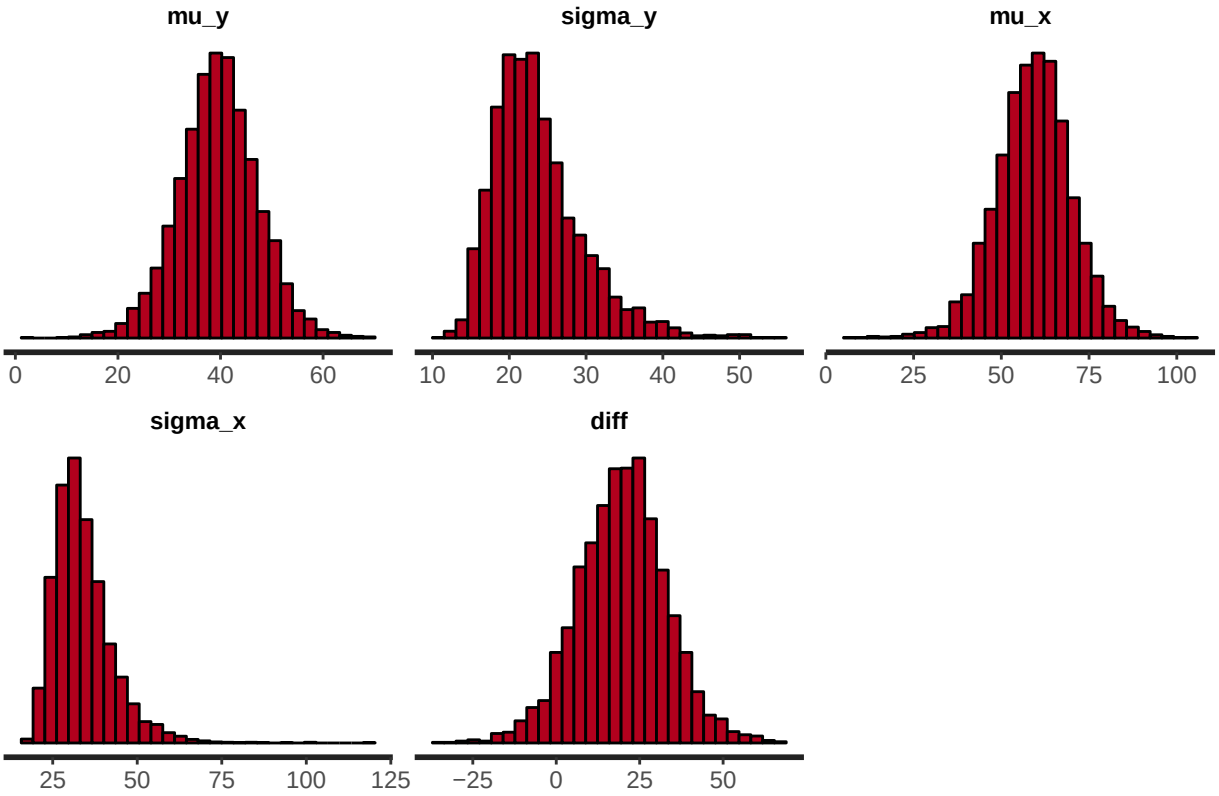
```
stan_trace(fit)
```



パラメータのサンプルのヒストグラム

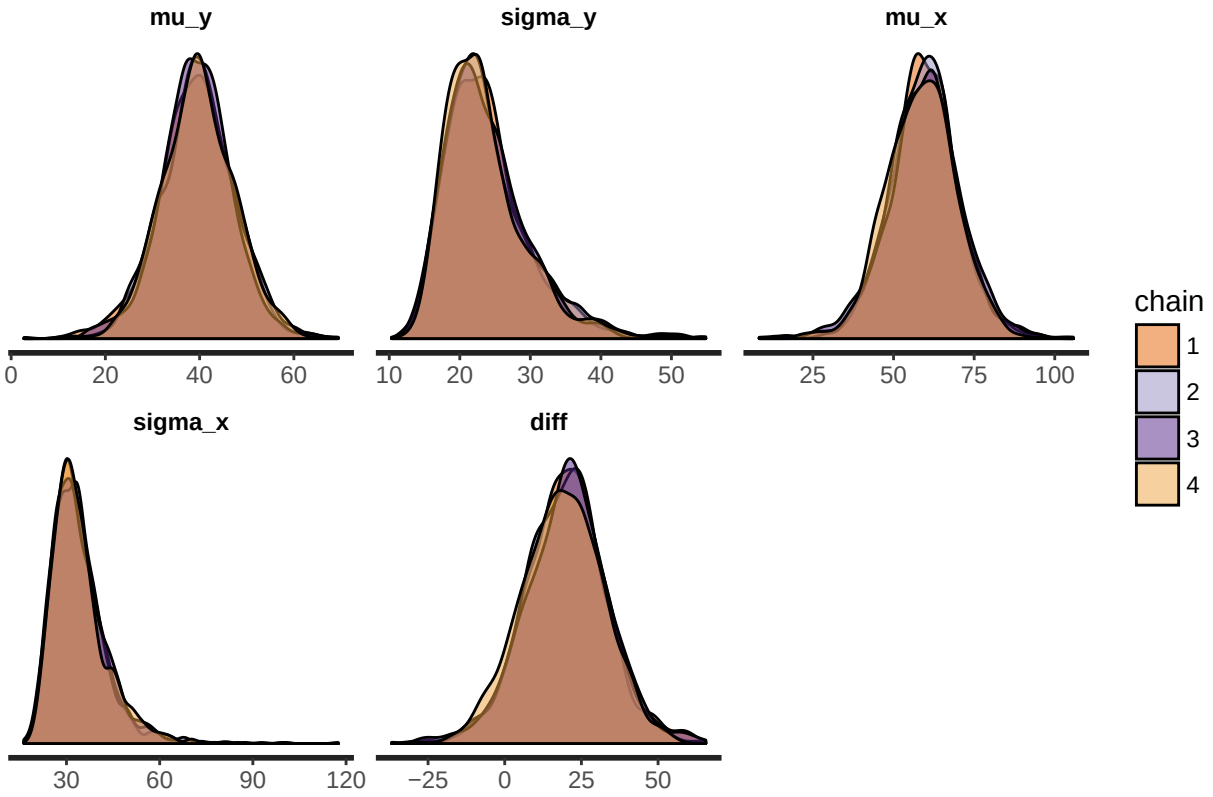
```
stan_hist(fit)
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



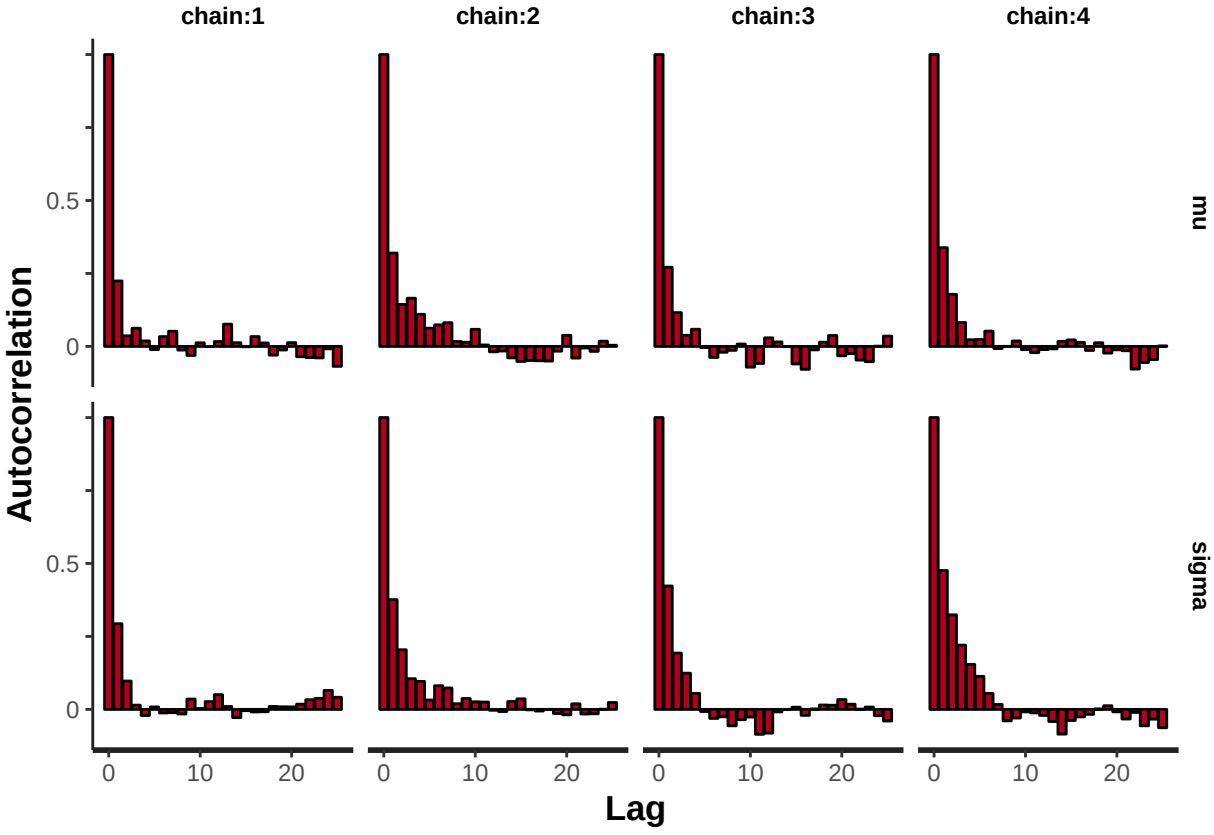
パラメータ事後分布の密度推定

```
stan_dens(fit,separate_chains = TRUE)
```



パラメータの自己相関

```
stan_ac(fit3,separate_chains = TRUE)
```



線形回帰

回帰分析もベイズ推定の枠組みで扱うことができる。 ここでは以下のデータを用いて単回帰分析を行う。

```
data{
  int N;  //
  real x[N];  //
  real y[N];  //
}

parameters{
  real a; //
  real b; //
  real<lower=0> sigma;  //
}

model{
  for (i in 1:N){
    y[i] ~ normal(b+a*x[i],sigma);
  }
}
```

```
library(rstan)
N<-30
x<-runif(N)
noise<-rnorm(N,mean=0,sd=0.1)
a<-1
b<-2
y<-b+a*x+noise
d<-list(x=x, y=y, N=N)
fit<-stan(file='linreg.stan',data=d)
```

```
##
## SAMPLING FOR MODEL '588538e1cebbf0d9aad8232558d90dcc' NOW (CHAIN 1).
##
## Gradient evaluation took 2.5e-05 seconds
## 1000 transitions using 10 leapfrog steps per transition would take 0.25 seconds.
## Adjust your expectations accordingly!
##
##
## Iteration:    1 / 2000 [  0%]  (Warmup)
## Iteration:  200 / 2000 [ 10%]  (Warmup)
## Iteration:  400 / 2000 [ 20%]  (Warmup)
## Iteration:  600 / 2000 [ 30%]  (Warmup)
## Iteration:  800 / 2000 [ 40%]  (Warmup)
## Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Iteration: 2000 / 2000 [100%]  (Sampling)
##
##  Elapsed Time: 0.069678 seconds (Warm-up)
##                0.076709 seconds (Sampling)
```

```
##                0.146387 seconds (Total)
##
##
## SAMPLING FOR MODEL '588538e1cebbf0d9aad8232558d90dcc' NOW (CHAIN 2).
##
## Gradient evaluation took 7e-06 seconds
## 1000 transitions using 10 leapfrog steps per transition would take 0.07 seconds.
## Adjust your expectations accordingly!
##
##
## Iteration:    1 / 2000 [  0%]  (Warmup)
## Iteration:  200 / 2000 [ 10%]  (Warmup)
## Iteration:  400 / 2000 [ 20%]  (Warmup)
## Iteration:  600 / 2000 [ 30%]  (Warmup)
## Iteration:  800 / 2000 [ 40%]  (Warmup)
## Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Iteration: 2000 / 2000 [100%]  (Sampling)
##
##  Elapsed Time: 0.091899 seconds (Warm-up)
##                0.077677 seconds (Sampling)
##                0.169576 seconds (Total)
##
##
## SAMPLING FOR MODEL '588538e1cebbf0d9aad8232558d90dcc' NOW (CHAIN 3).
##
## Gradient evaluation took 8e-06 seconds
## 1000 transitions using 10 leapfrog steps per transition would take 0.08 seconds.
## Adjust your expectations accordingly!
##
##
## Iteration:    1 / 2000 [  0%]  (Warmup)
## Iteration:  200 / 2000 [ 10%]  (Warmup)
## Iteration:  400 / 2000 [ 20%]  (Warmup)
## Iteration:  600 / 2000 [ 30%]  (Warmup)
## Iteration:  800 / 2000 [ 40%]  (Warmup)
## Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Iteration: 2000 / 2000 [100%]  (Sampling)
##
##  Elapsed Time: 0.078774 seconds (Warm-up)
##                0.070208 seconds (Sampling)
##                0.148982 seconds (Total)
##
##
## SAMPLING FOR MODEL '588538e1cebbf0d9aad8232558d90dcc' NOW (CHAIN 4).
```

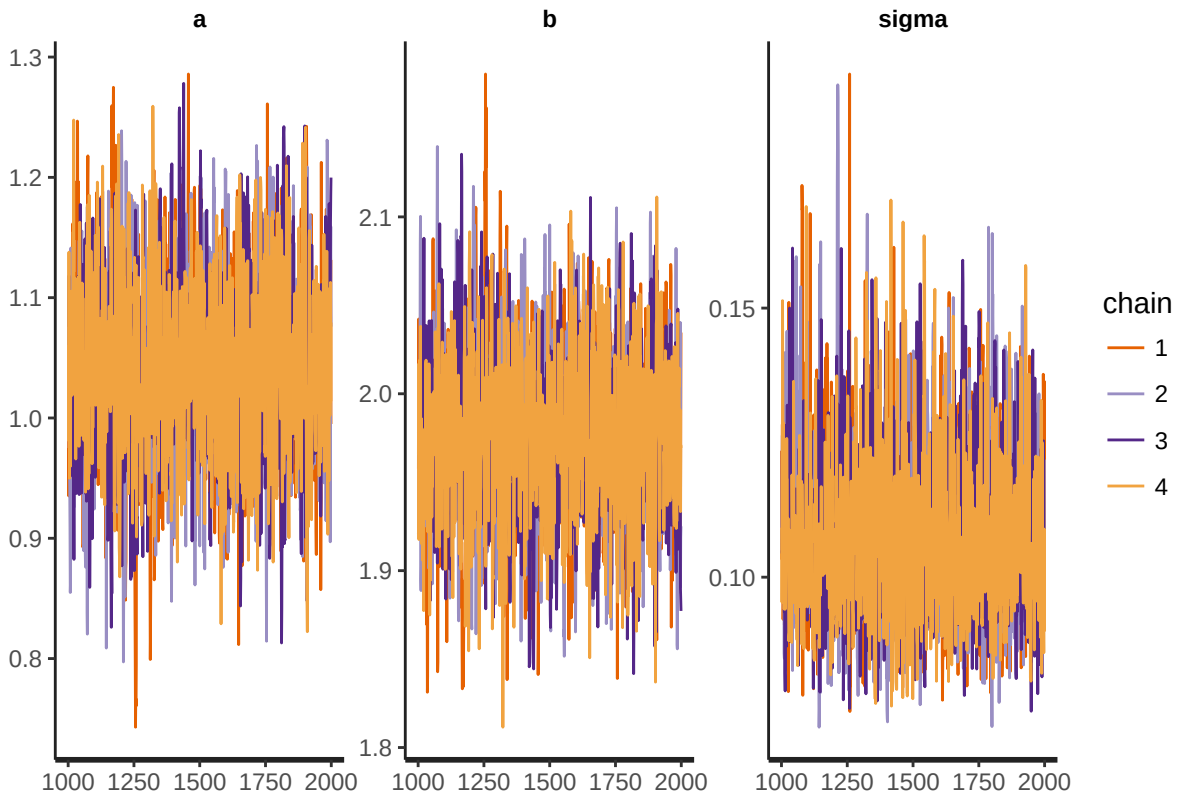
```
##
## Gradient evaluation took 1.2e-05 seconds
## 1000 transitions using 10 leapfrog steps per transition would take 0.12 seconds.
## Adjust your expectations accordingly!
##
##
## Iteration:    1 / 2000 [  0%]  (Warmup)
## Iteration:  200 / 2000 [ 10%]  (Warmup)
## Iteration:  400 / 2000 [ 20%]  (Warmup)
## Iteration:  600 / 2000 [ 30%]  (Warmup)
## Iteration:  800 / 2000 [ 40%]  (Warmup)
## Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Iteration: 2000 / 2000 [100%]  (Sampling)
##
##  Elapsed Time: 0.090038 seconds (Warm-up)
##                0.073104 seconds (Sampling)
##                0.163142 seconds (Total)
```

```
fit
```

```
## Inference for Stan model: 588538e1cebbf0d9aad8232558d90dcc.
## 4 chains, each with iter=2000; warmup=1000; thin=1;
## post-warmup draws per chain=1000, total post-warmup draws=4000.
##
##        mean se_mean   sd  2.5%   25%   50%   75% 97.5% n_eff Rhat
## a      1.04    0.00 0.07  0.90  0.99  1.04  1.09  1.19  1491    1
## b      1.97    0.00 0.05  1.88  1.94  1.97  2.01  2.07  1494    1
## sigma  0.11    0.00 0.02  0.08  0.10  0.11  0.12  0.14  1866    1
## lp__  50.37    0.04 1.31 47.13 49.76 50.73 51.32 51.86  1198    1
##
## Samples were drawn using NUTS(diag_e) at Mon Jul  2 10:53:44 2018.
## 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).
```

パラメータのサンプリング、横軸がステップ数

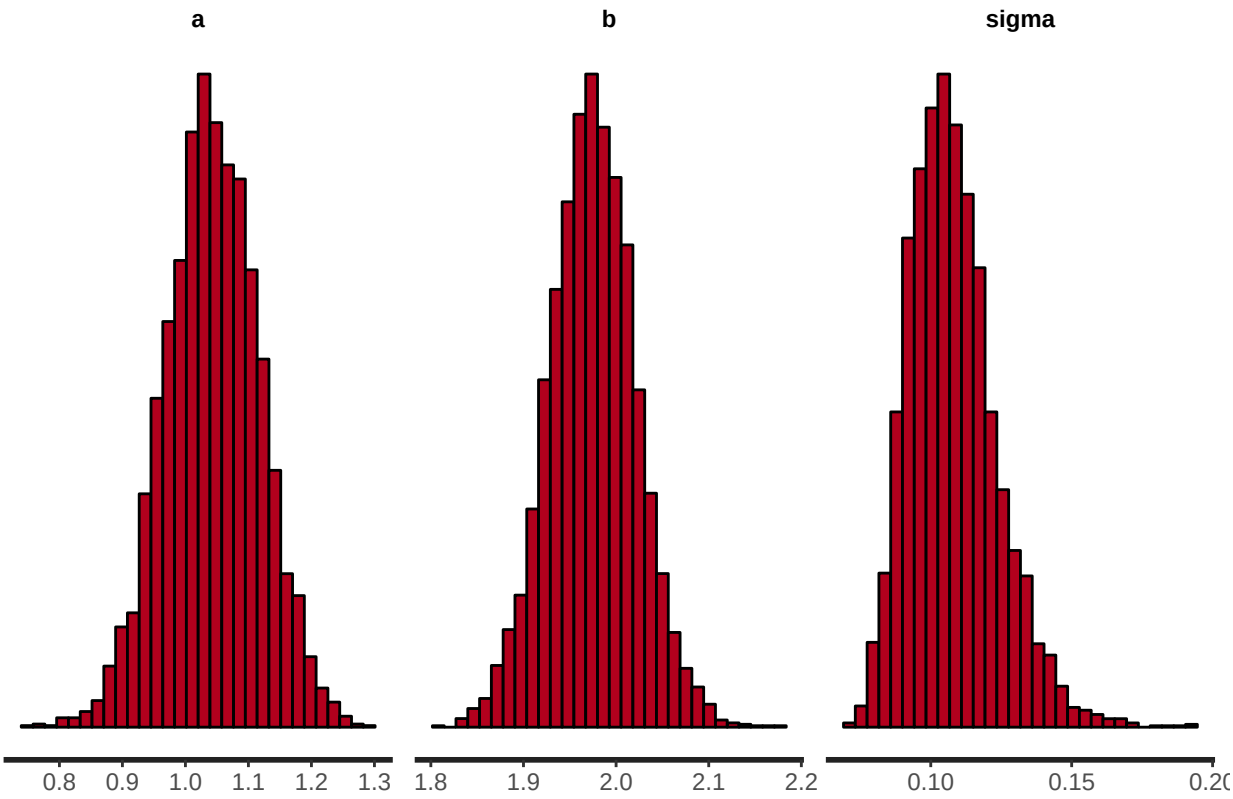
```
stan_trace(fit)
```



パラメータのサンプルのヒストグラム

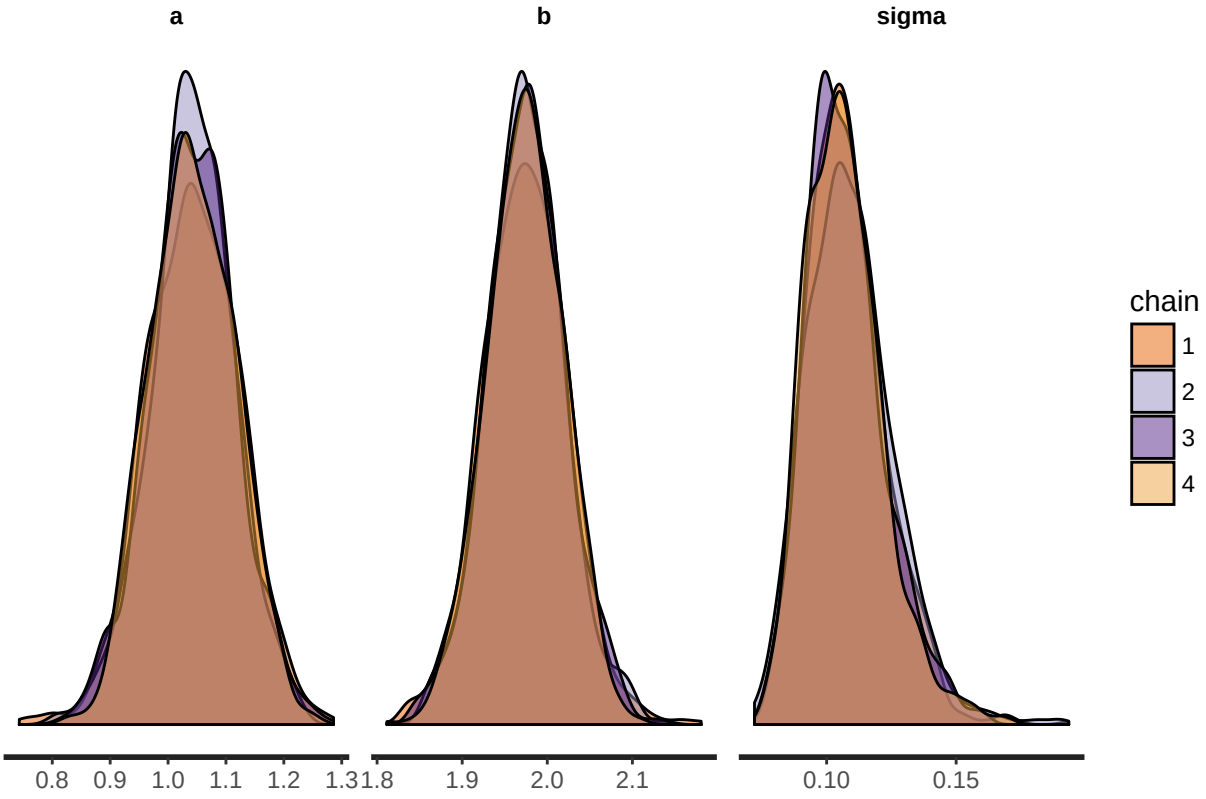
```
stan_hist(fit)
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



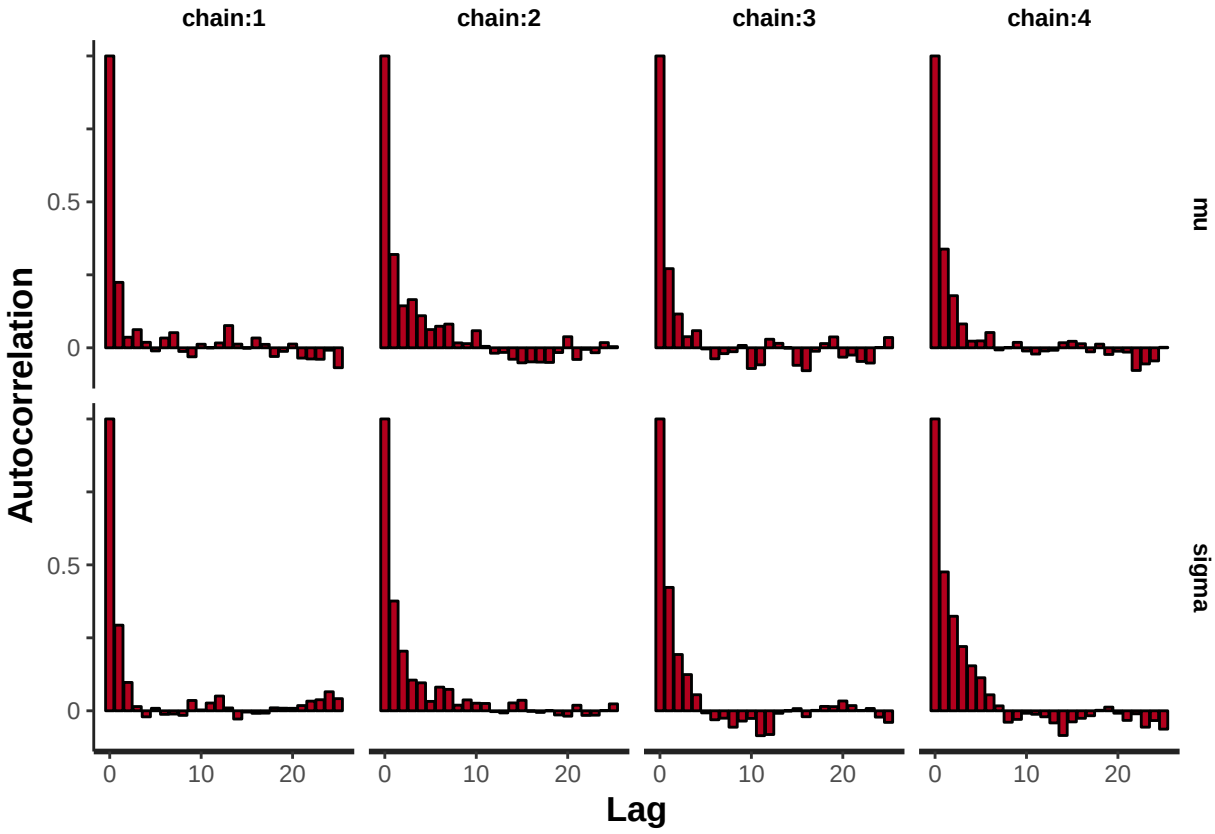
パラメータ事後分布の密度推定

```
stan_dens(fit,separate_chains = TRUE)
```



パラメータの自己相関

```
stan_ac(fit3,separate_chains = TRUE)
```



階層ベイズモデル

統計的モデリングの概要

データを発生させるメカニズムがある確率分布で記述できるとし、 それをよく説明する確率分布を作る。

階層ベイズとは

データが単純に一つの確率分布から発生するのではなく、　パラメータが発生するメカニズムとしてさらに確率分布を用いるなど、いくつかの確率分布を積み上げてモデルを作る。

例えば個体差、場所差などを入れることができる。

具体的に、以下の事例を見ながら、階層モデリングの様子を把握しよう。

ロジスティック回帰

ある20人の生徒に小テストを行った。 テストは10問からなり、結果は以下のようになった。

```
x <- c(10,9,6,7,8,3,4,9,3,7,10,2,0,5,6,3,1,9,0,0)
```

まずは二項分布モデルに基づいて、生徒の学力を正解率pとして推定しよう。 尤度関数を計算すると、以下のようになる。

$$L(p) = C\, p^{102}(1-p)^{98}$$

ここでCは適当な定数。

事前分布を一様分布として、事後分布は上で求めた結果からベータ分布$B(103, 99)$となる。 このときpの平均はおよそ0.51で、これに対する二項分布の

一方でデータから計算できる分散は12.1となる。

このように二項分布モデルで推定した分散より、 実際のデータの分散が大きくなる現象を過分散という。

これを解消するために、階層モデルを用いて予測する。

例えばここでは、生徒の個人差のパラメータr_iを用いることにしよう。 またこの個人差のパラメータは標準偏差sigmaの正規分布の従うとする。

```
data{
  int N;  //
  int M;  //
  int x[N]; //
}

parameters{
  real r[N];  //
  real beta;  //
}

transformed parameters{
  real<lower=0,upper=1> p[N]; //
  for (i in 1:N){
    p[i] = inv_logit(beta+r[i]);
  }
}

model{
  for (i in 1:N){
    x[i] ~ binomial(M,p[i]);
  }
}
```

上のコードを実行する。

```
library(rstan)
N<-20
x <- c(10,9,6,7,8,3,4,9,3,7,10,2,0,5,6,3,1,9,0,0)
d<-list(x=x, N=N, M=10)
fit<-stan(file='logreg.stan',data=d)
```

```
## In file included from file29a7e479809.cpp:8:
## In file included from /usr/local/lib/R/3.4/site-library/StanHeaders/include/src/stan/model/model_heade
## In file included from /usr/local/lib/R/3.4/site-library/StanHeaders/include/stan/math.hpp:4:
## In file included from /usr/local/lib/R/3.4/site-library/StanHeaders/include/stan/math/rev/mat.hpp:4:
## In file included from /usr/local/lib/R/3.4/site-library/StanHeaders/include/stan/math/rev/core.hpp:12:
## In file included from /usr/local/lib/R/3.4/site-library/StanHeaders/include/stan/math/rev/core/gevv_vv
## In file included from /usr/local/lib/R/3.4/site-library/StanHeaders/include/stan/math/rev/core/var.hpp
## In file included from /usr/local/lib/R/3.4/site-library/BH/include/boost/math/tools/config.hpp:13:
## In file included from /usr/local/lib/R/3.4/site-library/BH/include/boost/config.hpp:39:
## /usr/local/lib/R/3.4/site-library/BH/include/boost/config/compiler/clang.hpp:200:11: warning: 'BOOST_
## #  define BOOST_NO_CXX11_RVALUE_REFERENCES
##           ^
## <command line>:6:9: note: previous definition is here
## #define BOOST_NO_CXX11_RVALUE_REFERENCES 1
##         ^
```

```
## In file included from file29a7e479809.cpp:8:
## In file included from /usr/local/lib/R/3.4/site-library/StanHeaders/include/src/stan/model/model_heade
## In file included from /usr/local/lib/R/3.4/site-library/StanHeaders/include/stan/math.hpp:4:
## In file included from /usr/local/lib/R/3.4/site-library/StanHeaders/include/stan/math/rev/mat.hpp:4:
## In file included from /usr/local/lib/R/3.4/site-library/StanHeaders/include/stan/math/rev/core.hpp:14:
## In file included from /usr/local/lib/R/3.4/site-library/StanHeaders/include/stan/math/rev/core/matrix_
## In file included from /usr/local/lib/R/3.4/site-library/StanHeaders/include/stan/math/rev/mat/fun/Eige
## In file included from /usr/local/lib/R/3.4/site-library/StanHeaders/include/stan/math/prim/mat/fun/Eig
## In file included from /usr/local/lib/R/3.4/site-library/RcppEigen/include/Eigen/Dense:1:
## In file included from /usr/local/lib/R/3.4/site-library/RcppEigen/include/Eigen/Core:531:
## /usr/local/lib/R/3.4/site-library/RcppEigen/include/Eigen/src/Core/util/ReenableStupidWarnings.h:10:3
##     #pragma clang diagnostic pop
##                              ^
## In file included from file29a7e479809.cpp:8:
## In file included from /usr/local/lib/R/3.4/site-library/StanHeaders/include/src/stan/model/model_heade
## In file included from /usr/local/lib/R/3.4/site-library/StanHeaders/include/stan/math.hpp:4:
## In file included from /usr/local/lib/R/3.4/site-library/StanHeaders/include/stan/math/rev/mat.hpp:4:
## In file included from /usr/local/lib/R/3.4/site-library/StanHeaders/include/stan/math/rev/core.hpp:14:
## In file included from /usr/local/lib/R/3.4/site-library/StanHeaders/include/stan/math/rev/core/matrix_
## In file included from /usr/local/lib/R/3.4/site-library/StanHeaders/include/stan/math/rev/mat/fun/Eige
## In file included from /usr/local/lib/R/3.4/site-library/StanHeaders/include/stan/math/prim/mat/fun/Eig
## In file included from /usr/local/lib/R/3.4/site-library/RcppEigen/include/Eigen/Dense:2:
## In file included from /usr/local/lib/R/3.4/site-library/RcppEigen/include/Eigen/LU:47:
## /usr/local/lib/R/3.4/site-library/RcppEigen/include/Eigen/src/Core/util/ReenableStupidWarnings.h:10:3
##     #pragma clang diagnostic pop
##                              ^
## In file included from file29a7e479809.cpp:8:
## In file included from /usr/local/lib/R/3.4/site-library/StanHeaders/include/src/stan/model/model_heade
## In file included from /usr/local/lib/R/3.4/site-library/StanHeaders/include/stan/math.hpp:4:
## In file included from /usr/local/lib/R/3.4/site-library/StanHeaders/include/stan/math/rev/mat.hpp:4:
## In file included from /usr/local/lib/R/3.4/site-library/StanHeaders/include/stan/math/rev/core.hpp:14:
## In file included from /usr/local/lib/R/3.4/site-library/StanHeaders/include/stan/math/rev/core/matrix_
## In file included from /usr/local/lib/R/3.4/site-library/StanHeaders/include/stan/math/rev/mat/fun/Eige
## In file included from /usr/local/lib/R/3.4/site-library/StanHeaders/include/stan/math/prim/mat/fun/Eig
## In file included from /usr/local/lib/R/3.4/site-library/RcppEigen/include/Eigen/Dense:3:
## In file included from /usr/local/lib/R/3.4/site-library/RcppEigen/include/Eigen/Cholesky:12:
## In file included from /usr/local/lib/R/3.4/site-library/RcppEigen/include/Eigen/Jacobi:29:
## /usr/local/lib/R/3.4/site-library/RcppEigen/include/Eigen/src/Core/util/ReenableStupidWarnings.h:10:3
##     #pragma clang diagnostic pop
##                              ^
## In file included from file29a7e479809.cpp:8:
## In file included from /usr/local/lib/R/3.4/site-library/StanHeaders/include/src/stan/model/model_heade
## In file included from /usr/local/lib/R/3.4/site-library/StanHeaders/include/stan/math.hpp:4:
## In file included from /usr/local/lib/R/3.4/site-library/StanHeaders/include/stan/math/rev/mat.hpp:4:
## In file included from /usr/local/lib/R/3.4/site-library/StanHeaders/include/stan/math/rev/core.hpp:14:
## In file included from /usr/local/lib/R/3.4/site-library/StanHeaders/include/stan/math/rev/core/matrix_
## In file included from /usr/local/lib/R/3.4/site-library/StanHeaders/include/stan/math/rev/mat/fun/Eige
## In file included from /usr/local/lib/R/3.4/site-library/StanHeaders/include/stan/math/prim/mat/fun/Eig
## In file included from /usr/local/lib/R/3.4/site-library/RcppEigen/include/Eigen/Dense:3:
## In file included from /usr/local/lib/R/3.4/site-library/RcppEigen/include/Eigen/Cholesky:43:
## /usr/local/lib/R/3.4/site-library/RcppEigen/include/Eigen/src/Core/util/ReenableStupidWarnings.h:10:3
##     #pragma clang diagnostic pop
##                              ^
## In file included from file29a7e479809.cpp:8:
```

```
## In file included from /usr/local/lib/R/3.4/site-library/StanHeaders/include/src/stan/model/model_heade
## In file included from /usr/local/lib/R/3.4/site-library/StanHeaders/include/stan/math.hpp:4:
## In file included from /usr/local/lib/R/3.4/site-library/StanHeaders/include/stan/math/rev/mat.hpp:4:
## In file included from /usr/local/lib/R/3.4/site-library/StanHeaders/include/stan/math/rev/core.hpp:14:
## In file included from /usr/local/lib/R/3.4/site-library/StanHeaders/include/stan/math/rev/core/matrix_
## In file included from /usr/local/lib/R/3.4/site-library/StanHeaders/include/stan/math/rev/mat/fun/Eige
## In file included from /usr/local/lib/R/3.4/site-library/StanHeaders/include/stan/math/prim/mat/fun/Eig
## In file included from /usr/local/lib/R/3.4/site-library/RcppEigen/include/Eigen/Dense:4:
## In file included from /usr/local/lib/R/3.4/site-library/RcppEigen/include/Eigen/QR:17:
## In file included from /usr/local/lib/R/3.4/site-library/RcppEigen/include/Eigen/Householder:27:
## /usr/local/lib/R/3.4/site-library/RcppEigen/include/Eigen/src/Core/util/ReenableStupidWarnings.h:10:3
##     #pragma clang diagnostic pop
##                              ^
## In file included from file29a7e479809.cpp:8:
## In file included from /usr/local/lib/R/3.4/site-library/StanHeaders/include/src/stan/model/model_heade
## In file included from /usr/local/lib/R/3.4/site-library/StanHeaders/include/stan/math.hpp:4:
## In file included from /usr/local/lib/R/3.4/site-library/StanHeaders/include/stan/math/rev/mat.hpp:4:
## In file included from /usr/local/lib/R/3.4/site-library/StanHeaders/include/stan/math/rev/core.hpp:14:
## In file included from /usr/local/lib/R/3.4/site-library/StanHeaders/include/stan/math/rev/core/matrix_
## In file included from /usr/local/lib/R/3.4/site-library/StanHeaders/include/stan/math/rev/mat/fun/Eige
## In file included from /usr/local/lib/R/3.4/site-library/StanHeaders/include/stan/math/prim/mat/fun/Eig
## In file included from /usr/local/lib/R/3.4/site-library/RcppEigen/include/Eigen/Dense:5:
## In file included from /usr/local/lib/R/3.4/site-library/RcppEigen/include/Eigen/SVD:48:
## /usr/local/lib/R/3.4/site-library/RcppEigen/include/Eigen/src/Core/util/ReenableStupidWarnings.h:10:3
##     #pragma clang diagnostic pop
##                              ^
## In file included from file29a7e479809.cpp:8:
## In file included from /usr/local/lib/R/3.4/site-library/StanHeaders/include/src/stan/model/model_heade
## In file included from /usr/local/lib/R/3.4/site-library/StanHeaders/include/stan/math.hpp:4:
## In file included from /usr/local/lib/R/3.4/site-library/StanHeaders/include/stan/math/rev/mat.hpp:4:
## In file included from /usr/local/lib/R/3.4/site-library/StanHeaders/include/stan/math/rev/core.hpp:14:
## In file included from /usr/local/lib/R/3.4/site-library/StanHeaders/include/stan/math/rev/core/matrix_
## In file included from /usr/local/lib/R/3.4/site-library/StanHeaders/include/stan/math/rev/mat/fun/Eige
## In file included from /usr/local/lib/R/3.4/site-library/StanHeaders/include/stan/math/prim/mat/fun/Eig
## In file included from /usr/local/lib/R/3.4/site-library/RcppEigen/include/Eigen/Dense:6:
## In file included from /usr/local/lib/R/3.4/site-library/RcppEigen/include/Eigen/Geometry:58:
## /usr/local/lib/R/3.4/site-library/RcppEigen/include/Eigen/src/Core/util/ReenableStupidWarnings.h:10:3
##     #pragma clang diagnostic pop
##                              ^
## In file included from file29a7e479809.cpp:8:
## In file included from /usr/local/lib/R/3.4/site-library/StanHeaders/include/src/stan/model/model_heade
## In file included from /usr/local/lib/R/3.4/site-library/StanHeaders/include/stan/math.hpp:4:
## In file included from /usr/local/lib/R/3.4/site-library/StanHeaders/include/stan/math/rev/mat.hpp:4:
## In file included from /usr/local/lib/R/3.4/site-library/StanHeaders/include/stan/math/rev/core.hpp:14:
## In file included from /usr/local/lib/R/3.4/site-library/StanHeaders/include/stan/math/rev/core/matrix_
## In file included from /usr/local/lib/R/3.4/site-library/StanHeaders/include/stan/math/rev/mat/fun/Eige
## In file included from /usr/local/lib/R/3.4/site-library/StanHeaders/include/stan/math/prim/mat/fun/Eig
## In file included from /usr/local/lib/R/3.4/site-library/RcppEigen/include/Eigen/Dense:7:
## In file included from /usr/local/lib/R/3.4/site-library/RcppEigen/include/Eigen/Eigenvalues:58:
## /usr/local/lib/R/3.4/site-library/RcppEigen/include/Eigen/src/Core/util/ReenableStupidWarnings.h:10:3
##     #pragma clang diagnostic pop
##                              ^
## In file included from file29a7e479809.cpp:8:
## In file included from /usr/local/lib/R/3.4/site-library/StanHeaders/include/src/stan/model/model_heade
```

```
## In file included from /usr/local/lib/R/3.4/site-library/StanHeaders/include/stan/math.hpp:4:
## In file included from /usr/local/lib/R/3.4/site-library/StanHeaders/include/stan/math/rev/mat.hpp:12:
## In file included from /usr/local/lib/R/3.4/site-library/StanHeaders/include/stan/math/prim/mat.hpp:83:
## In file included from /usr/local/lib/R/3.4/site-library/StanHeaders/include/stan/math/prim/mat/fun/csr
## In file included from /usr/local/lib/R/3.4/site-library/RcppEigen/include/Eigen/Sparse:26:
## In file included from /usr/local/lib/R/3.4/site-library/RcppEigen/include/Eigen/SparseCore:66:
## /usr/local/lib/R/3.4/site-library/RcppEigen/include/Eigen/src/Core/util/ReenableStupidWarnings.h:10:3
##     #pragma clang diagnostic pop
##                              ^
## In file included from file29a7e479809.cpp:8:
## In file included from /usr/local/lib/R/3.4/site-library/StanHeaders/include/src/stan/model/model_heade
## In file included from /usr/local/lib/R/3.4/site-library/StanHeaders/include/stan/math.hpp:4:
## In file included from /usr/local/lib/R/3.4/site-library/StanHeaders/include/stan/math/rev/mat.hpp:12:
## In file included from /usr/local/lib/R/3.4/site-library/StanHeaders/include/stan/math/prim/mat.hpp:83:
## In file included from /usr/local/lib/R/3.4/site-library/StanHeaders/include/stan/math/prim/mat/fun/csr
## In file included from /usr/local/lib/R/3.4/site-library/RcppEigen/include/Eigen/Sparse:27:
## In file included from /usr/local/lib/R/3.4/site-library/RcppEigen/include/Eigen/OrderingMethods:71:
## /usr/local/lib/R/3.4/site-library/RcppEigen/include/Eigen/src/Core/util/ReenableStupidWarnings.h:10:3
##     #pragma clang diagnostic pop
##                              ^
## In file included from file29a7e479809.cpp:8:
## In file included from /usr/local/lib/R/3.4/site-library/StanHeaders/include/src/stan/model/model_heade
## In file included from /usr/local/lib/R/3.4/site-library/StanHeaders/include/stan/math.hpp:4:
## In file included from /usr/local/lib/R/3.4/site-library/StanHeaders/include/stan/math/rev/mat.hpp:12:
## In file included from /usr/local/lib/R/3.4/site-library/StanHeaders/include/stan/math/prim/mat.hpp:83:
## In file included from /usr/local/lib/R/3.4/site-library/StanHeaders/include/stan/math/prim/mat/fun/csr
## In file included from /usr/local/lib/R/3.4/site-library/RcppEigen/include/Eigen/Sparse:29:
## In file included from /usr/local/lib/R/3.4/site-library/RcppEigen/include/Eigen/SparseCholesky:43:
## /usr/local/lib/R/3.4/site-library/RcppEigen/include/Eigen/src/Core/util/ReenableStupidWarnings.h:10:3
##     #pragma clang diagnostic pop
##                              ^
## In file included from file29a7e479809.cpp:8:
## In file included from /usr/local/lib/R/3.4/site-library/StanHeaders/include/src/stan/model/model_heade
## In file included from /usr/local/lib/R/3.4/site-library/StanHeaders/include/stan/math.hpp:4:
## In file included from /usr/local/lib/R/3.4/site-library/StanHeaders/include/stan/math/rev/mat.hpp:12:
## In file included from /usr/local/lib/R/3.4/site-library/StanHeaders/include/stan/math/prim/mat.hpp:83:
## In file included from /usr/local/lib/R/3.4/site-library/StanHeaders/include/stan/math/prim/mat/fun/csr
## In file included from /usr/local/lib/R/3.4/site-library/RcppEigen/include/Eigen/Sparse:32:
## In file included from /usr/local/lib/R/3.4/site-library/RcppEigen/include/Eigen/SparseQR:35:
## /usr/local/lib/R/3.4/site-library/RcppEigen/include/Eigen/src/Core/util/ReenableStupidWarnings.h:10:3
##     #pragma clang diagnostic pop
##                              ^
## In file included from file29a7e479809.cpp:8:
## In file included from /usr/local/lib/R/3.4/site-library/StanHeaders/include/src/stan/model/model_heade
## In file included from /usr/local/lib/R/3.4/site-library/StanHeaders/include/stan/math.hpp:4:
## In file included from /usr/local/lib/R/3.4/site-library/StanHeaders/include/stan/math/rev/mat.hpp:12:
## In file included from /usr/local/lib/R/3.4/site-library/StanHeaders/include/stan/math/prim/mat.hpp:83:
## In file included from /usr/local/lib/R/3.4/site-library/StanHeaders/include/stan/math/prim/mat/fun/csr
## In file included from /usr/local/lib/R/3.4/site-library/RcppEigen/include/Eigen/Sparse:33:
## In file included from /usr/local/lib/R/3.4/site-library/RcppEigen/include/Eigen/IterativeLinearSolvers
## /usr/local/lib/R/3.4/site-library/RcppEigen/include/Eigen/src/Core/util/ReenableStupidWarnings.h:10:3
##     #pragma clang diagnostic pop
##                              ^
## 14 warnings generated.
```

```
##
## SAMPLING FOR MODEL 'logreg' NOW (CHAIN 1).
##
## Gradient evaluation took 2.5e-05 seconds
## 1000 transitions using 10 leapfrog steps per transition would take 0.25 seconds.
## Adjust your expectations accordingly!
##
##
## Iteration:    1 / 2000 [  0%]  (Warmup)
## Iteration:  200 / 2000 [ 10%]  (Warmup)
## Iteration:  400 / 2000 [ 20%]  (Warmup)
## Iteration:  600 / 2000 [ 30%]  (Warmup)
## Iteration:  800 / 2000 [ 40%]  (Warmup)
## Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Iteration: 2000 / 2000 [100%]  (Sampling)
##
##  Elapsed Time: 0.168391 seconds (Warm-up)
##                0.165651 seconds (Sampling)
##                0.334042 seconds (Total)
##
##
## SAMPLING FOR MODEL 'logreg' NOW (CHAIN 2).
##
## Gradient evaluation took 1.4e-05 seconds
## 1000 transitions using 10 leapfrog steps per transition would take 0.14 seconds.
## Adjust your expectations accordingly!
##
##
## Iteration:    1 / 2000 [  0%]  (Warmup)
## Iteration:  200 / 2000 [ 10%]  (Warmup)
## Iteration:  400 / 2000 [ 20%]  (Warmup)
## Iteration:  600 / 2000 [ 30%]  (Warmup)
## Iteration:  800 / 2000 [ 40%]  (Warmup)
## Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Iteration: 2000 / 2000 [100%]  (Sampling)
##
##  Elapsed Time: 0.168928 seconds (Warm-up)
##                0.156544 seconds (Sampling)
##                0.325472 seconds (Total)
##
##
## SAMPLING FOR MODEL 'logreg' NOW (CHAIN 3).
##
## Gradient evaluation took 1.3e-05 seconds
```

```
## 1000 transitions using 10 leapfrog steps per transition would take 0.13 seconds.
## Adjust your expectations accordingly!
##
##
## Iteration:    1 / 2000 [  0%]  (Warmup)
## Iteration:  200 / 2000 [ 10%]  (Warmup)
## Iteration:  400 / 2000 [ 20%]  (Warmup)
## Iteration:  600 / 2000 [ 30%]  (Warmup)
## Iteration:  800 / 2000 [ 40%]  (Warmup)
## Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Iteration: 2000 / 2000 [100%]  (Sampling)
##
##  Elapsed Time: 0.170236 seconds (Warm-up)
##                0.149817 seconds (Sampling)
##                0.320053 seconds (Total)
##
##
## SAMPLING FOR MODEL 'logreg' NOW (CHAIN 4).
##
## Gradient evaluation took 2.2e-05 seconds
## 1000 transitions using 10 leapfrog steps per transition would take 0.22 seconds.
## Adjust your expectations accordingly!
##
##
## Iteration:    1 / 2000 [  0%]  (Warmup)
## Iteration:  200 / 2000 [ 10%]  (Warmup)
## Iteration:  400 / 2000 [ 20%]  (Warmup)
## Iteration:  600 / 2000 [ 30%]  (Warmup)
## Iteration:  800 / 2000 [ 40%]  (Warmup)
## Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Iteration: 2000 / 2000 [100%]  (Sampling)
##
##  Elapsed Time: 0.16312 seconds (Warm-up)
##                0.151383 seconds (Sampling)
##                0.314503 seconds (Total)
```

```
fit
```

```
## Inference for Stan model: logreg.
## 4 chains, each with iter=2000; warmup=1000; thin=1;
## post-warmup draws per chain=1000, total post-warmup draws=4000.
##
##          mean se_mean   sd    2.5%     25%     50%     75%   97.5% n_eff
## sigma    2.46    0.02 0.67    1.48    1.99    2.36    2.82    3.96  1451
## r[1]     3.36    0.04 1.63    0.86    2.26    3.15    4.22    7.14  2157
```

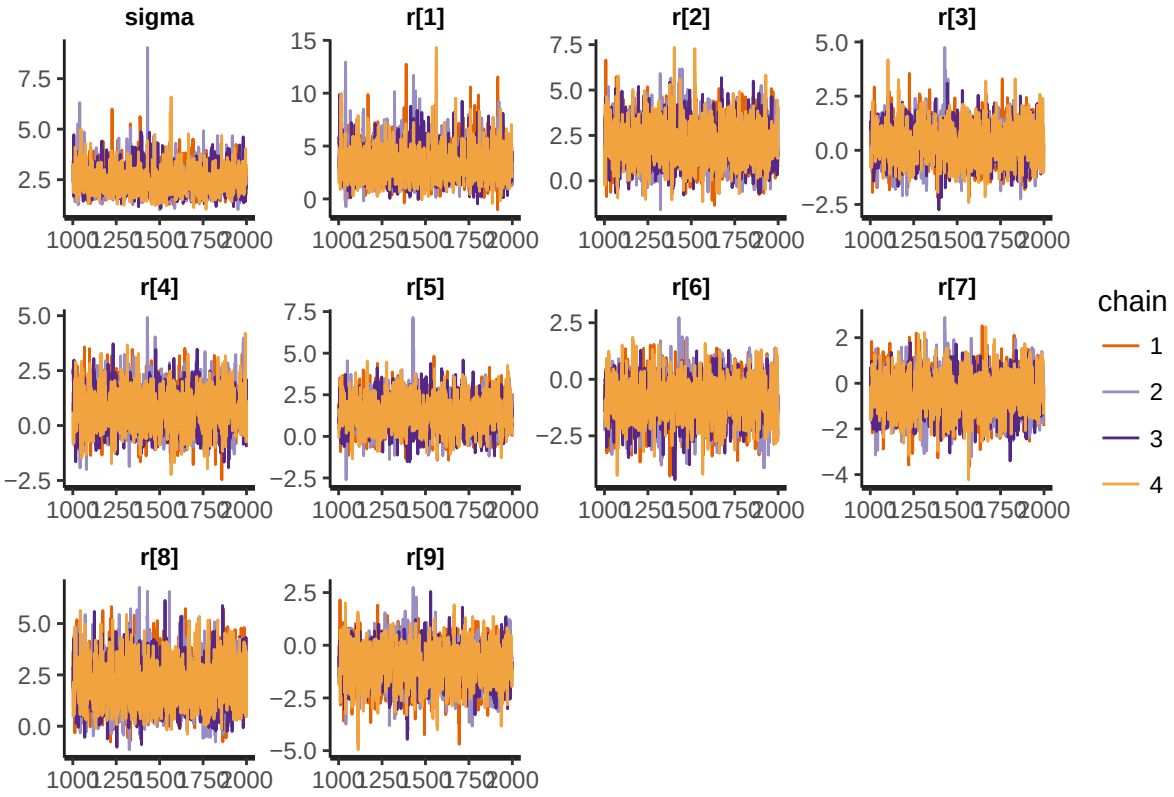
```
## r[2]     2.06    0.02 1.11    0.06    1.30    1.96    2.76    4.43  4000
## r[3]     0.34    0.02 0.84   -1.26   -0.20    0.32    0.88    1.99  1583
## r[4]     0.79    0.02 0.90   -0.92    0.19    0.78    1.37    2.60  1840
## r[5]     1.31    0.02 0.95   -0.45    0.67    1.26    1.90    3.30  1805
## r[6]    -0.92    0.02 0.90   -2.76   -1.50   -0.90   -0.31    0.80  1724
## r[7]    -0.48    0.02 0.86   -2.21   -1.04   -0.47    0.09    1.16  1675
## r[8]     2.05    0.02 1.11    0.13    1.29    1.96    2.74    4.45  2132
## r[9]    -0.92    0.02 0.90   -2.76   -1.49   -0.88   -0.32    0.78  1753
## r[10]    0.79    0.02 0.88   -0.80    0.20    0.75    1.36    2.63  1760
## r[11]    3.37    0.03 1.63    0.80    2.28    3.13    4.25    7.12  2233
## r[12]   -1.45    0.02 0.95   -3.40   -2.06   -1.41   -0.82    0.32  1879
## r[13]   -3.49    0.04 1.62   -7.12   -4.39   -3.26   -2.37   -0.98  1984
## r[14]   -0.08    0.02 0.85   -1.74   -0.63   -0.10    0.46    1.60  1578
## r[15]    0.34    0.02 0.87   -1.32   -0.23    0.33    0.90    2.10  1585
## r[16]   -0.92    0.02 0.90   -2.69   -1.51   -0.91   -0.30    0.80  1762
## r[17]   -2.16    0.02 1.11   -4.58   -2.82   -2.09   -1.40   -0.21  2124
## r[18]    2.03    0.02 1.09    0.12    1.27    1.95    2.69    4.47  2223
## r[19]   -3.47    0.04 1.60   -7.30   -4.33   -3.28   -2.36   -0.95  1958
## r[20]   -3.46    0.03 1.55   -7.24   -4.27   -3.23   -2.39   -1.11  2041
## beta     0.07    0.02 0.60   -1.11   -0.30    0.07    0.44    1.22   935
## p[1]     0.94    0.00 0.07    0.75    0.91    0.96    0.99    1.00  4000
## p[2]     0.86    0.00 0.10    0.61    0.81    0.88    0.94    0.99  4000
## p[3]     0.59    0.00 0.14    0.32    0.49    0.60    0.70    0.84  4000
## p[4]     0.68    0.00 0.14    0.38    0.60    0.70    0.79    0.91  4000
## p[5]     0.77    0.00 0.12    0.50    0.70    0.79    0.87    0.96  4000
## p[6]     0.32    0.00 0.14    0.09    0.22    0.31    0.41    0.61  4000
## p[7]     0.41    0.00 0.14    0.15    0.30    0.40    0.51    0.69  4000
## p[8]     0.86    0.00 0.10    0.63    0.81    0.88    0.94    0.99  4000
## p[9]     0.32    0.00 0.14    0.09    0.22    0.30    0.41    0.61  4000
## p[10]    0.68    0.00 0.14    0.39    0.60    0.70    0.79    0.91  4000
## p[11]    0.94    0.00 0.07    0.75    0.91    0.96    0.99    1.00  4000
## p[12]    0.23    0.00 0.12    0.04    0.13    0.21    0.30    0.49  4000
## p[13]    0.06    0.00 0.07    0.00    0.01    0.04    0.09    0.24  4000
## p[14]    0.50    0.00 0.15    0.22    0.40    0.50    0.60    0.78  4000
## p[15]    0.59    0.00 0.14    0.30    0.49    0.60    0.70    0.85  4000
## p[16]    0.32    0.00 0.14    0.09    0.21    0.31    0.41    0.62  4000
## p[17]    0.14    0.00 0.10    0.01    0.07    0.12    0.20    0.38  4000
## p[18]    0.86    0.00 0.10    0.61    0.81    0.88    0.94    0.99  4000
## p[19]    0.06    0.00 0.07    0.00    0.01    0.04    0.09    0.25  4000
## p[20]    0.06    0.00 0.06    0.00    0.02    0.04    0.09    0.23  4000
## lp__  -117.13    0.12 4.05 -126.04 -119.57 -116.77 -114.22 -110.29  1055
##       Rhat
## sigma    1
## r[1]     1
## r[2]     1
## r[3]     1
## r[4]     1
## r[5]     1
## r[6]     1
## r[7]     1
## r[8]     1
## r[9]     1
## r[10]    1
## r[11]    1
```

```
## r[12]    1
## r[13]    1
## r[14]    1
## r[15]    1
## r[16]    1
## r[17]    1
## r[18]    1
## r[19]    1
## r[20]    1
## beta     1
## p[1]     1
## p[2]     1
## p[3]     1
## p[4]     1
## p[5]     1
## p[6]     1
## p[7]     1
## p[8]     1
## p[9]     1
## p[10]    1
## p[11]    1
## p[12]    1
## p[13]    1
## p[14]    1
## p[15]    1
## p[16]    1
## p[17]    1
## p[18]    1
## p[19]    1
## p[20]    1
## lp__     1
##
## Samples were drawn using NUTS(diag_e) at Mon Jul  2 10:55:40 2018.
## 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).
```

パラメータのサンプリング、横軸がステップ数

```
stan_trace(fit)
```

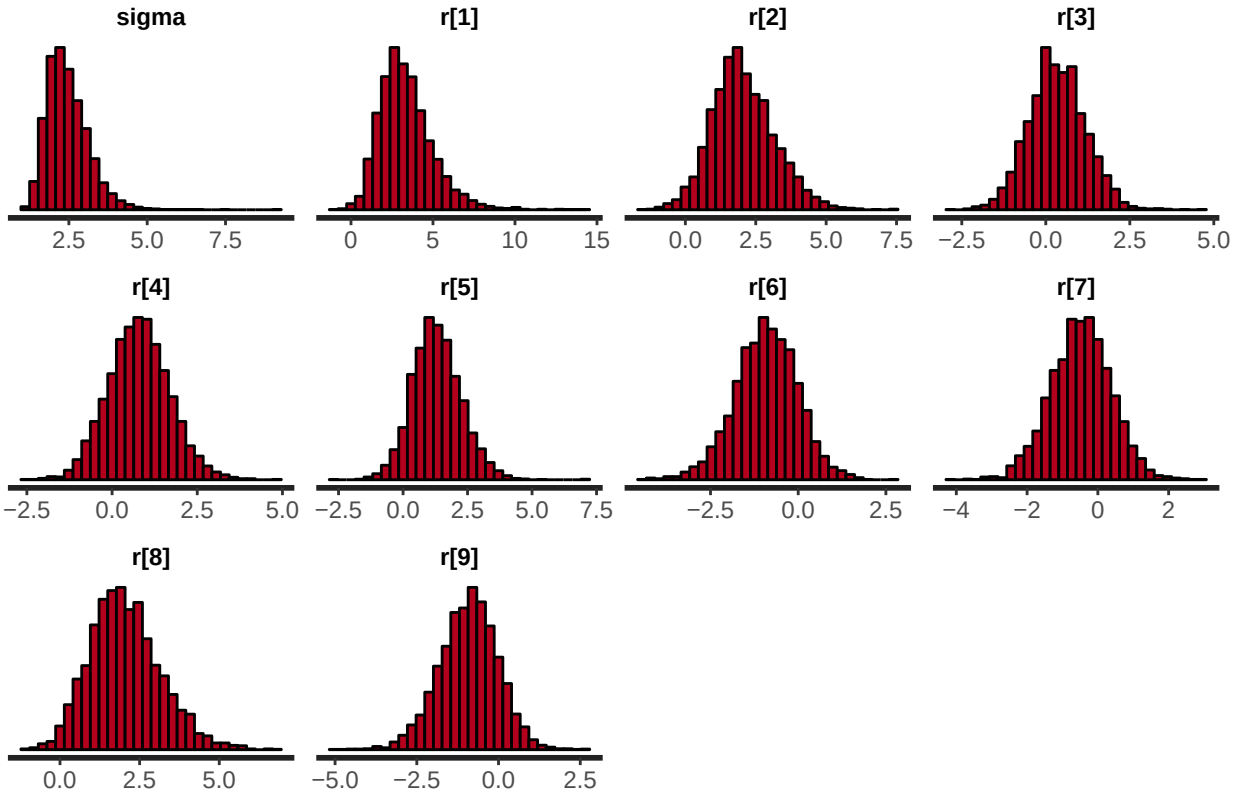
```
## 'pars' not specified. Showing first 10 parameters by default.
```



パラメータのサンプルのヒストグラム

```
stan_hist(fit)
```

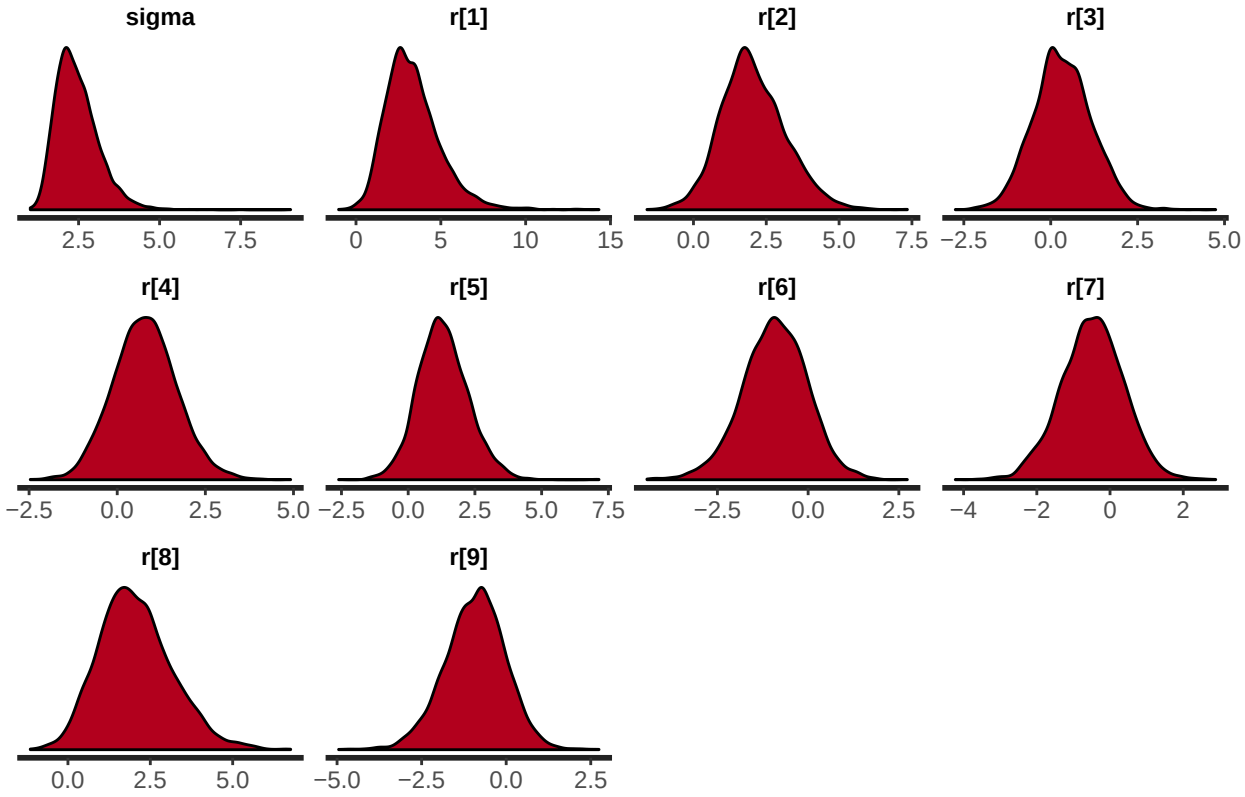
```
## 'pars' not specified. Showing first 10 parameters by default.
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



パラメータ事後分布の密度推定

```
stan_dens(fit)
```

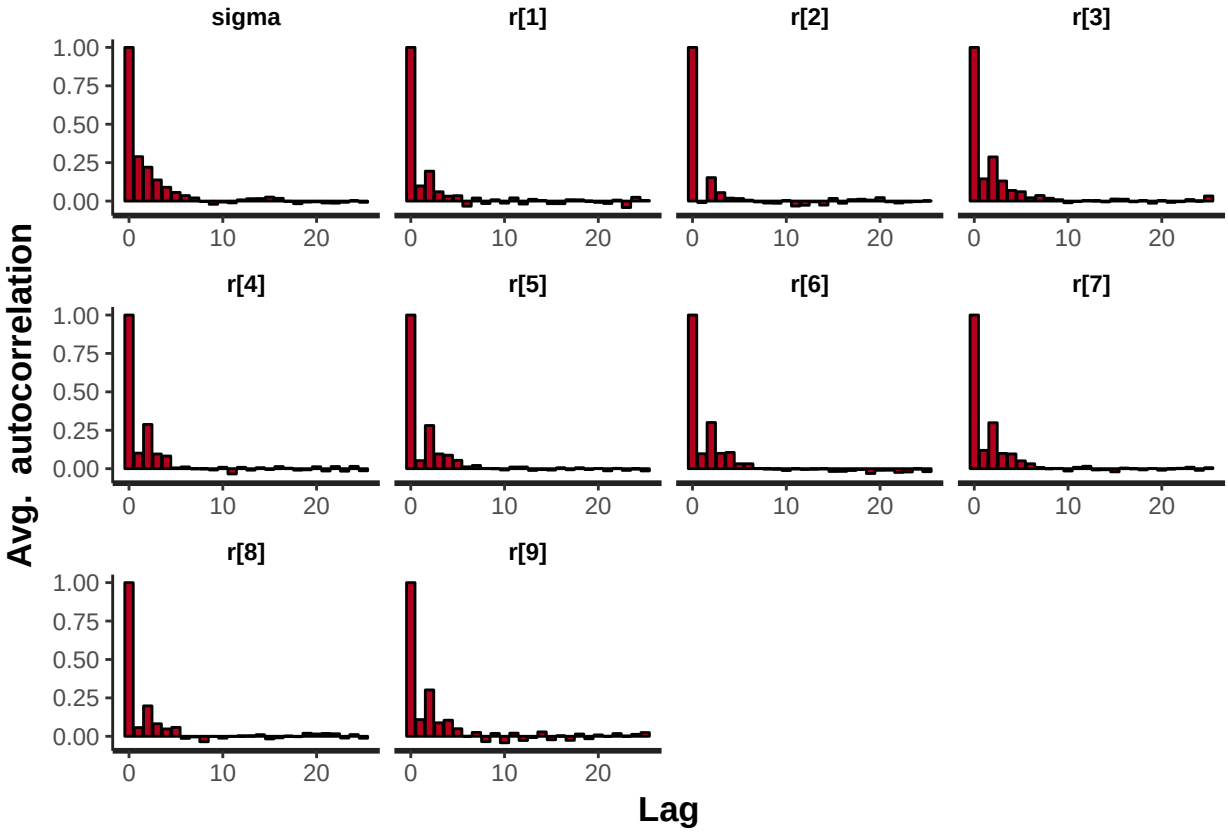
```
## 'pars' not specified. Showing first 10 parameters by default.
```



パラメータの自己相関

```
stan_ac(fit)
```

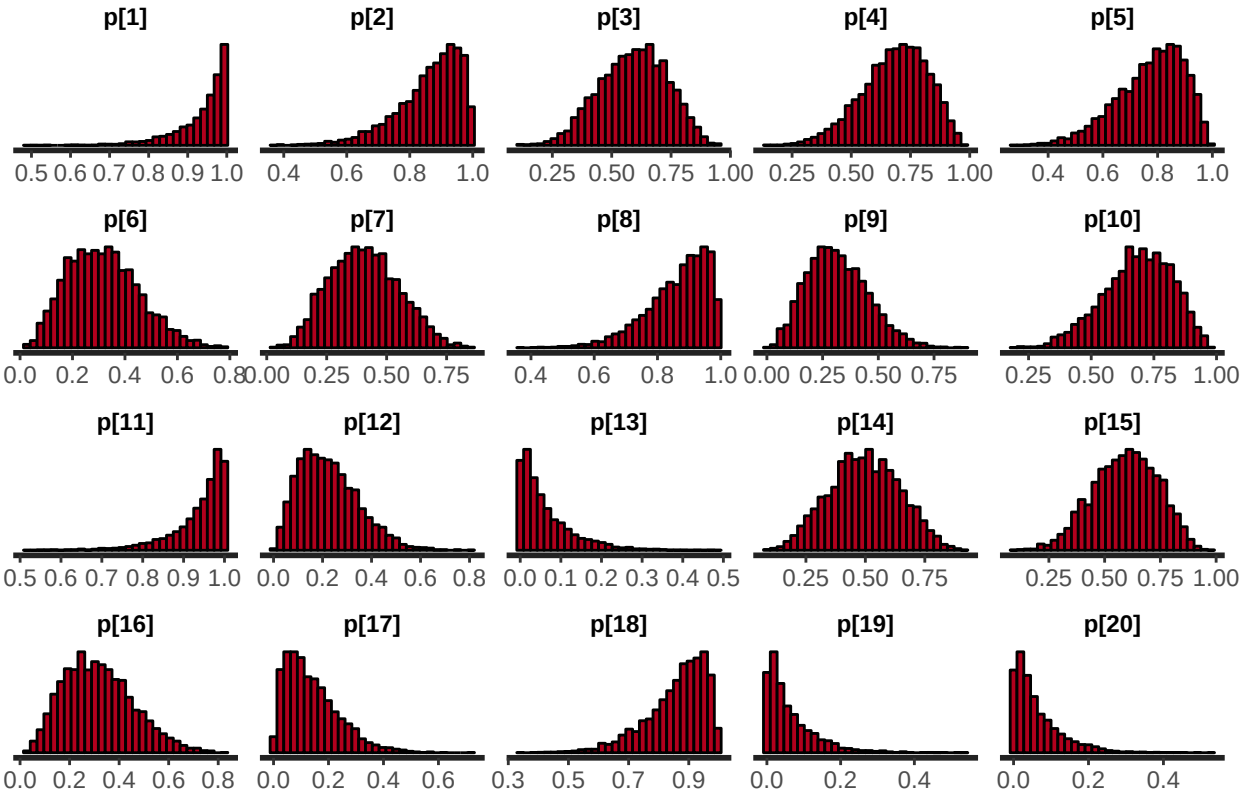
```
## 'pars' not specified. Showing first 10 parameters by default.
```



各生徒の正解率の推定は以下のようになる。

```
stan_hist(fit, pars = "p")
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



状態空間モデル

状態方程式と観測方程式

観測値y_nを直接説明するのではなく、真の状態x_nの変化とその観測に分けて考える。

真の状態の様子を説明する式をシステムモデルや状態方程式と呼ぶ。

$$x_n = F_n(x_{n-1}) + G_n(v_n)$$

ここでv_nはシステムノイズといわれ、ある確率分布に従う乱数。

また真の状態から観測値が発生する様子を説明する式を観測モデルや観測方程式という。

$$y_n = H_n(x_n) + w_n$$

ここでw_nは観測ノイズといわれ、ある確率分布に従う乱数。

一番基本的な例は線形ガウス型モデルと言われるもので、 上の式において、線形関数$F_n(x) = F_n x, G_n(v) = G_n v, H_n(x) = H_n x$とし、$v_n \sim N(0, Q_n), w_n \sim N(0, R_n)$は正規分布に従う乱数。

また時系列モデルの基本的な例であるARモデル

$$y_n = a_1 y_{n-1} + a_2 y_{n-2} + v_n,\ v_n \sim N(0, \sigma^2)$$

は、状態空間モデルの言葉を用いると、

$$x_n = \begin{pmatrix} y_n \\ y_{n-1} \end{pmatrix}, F = \begin{pmatrix} a_1 & a_2 \\ 1 & 0 \end{pmatrix}, G = \begin{pmatrix} 1 \\ 0 \end{pmatrix}, H = \begin{pmatrix} 1 & 0 \end{pmatrix}, Q = \sigma^2, R = 0$$

として表すことができる。

ローカルレベルモデル

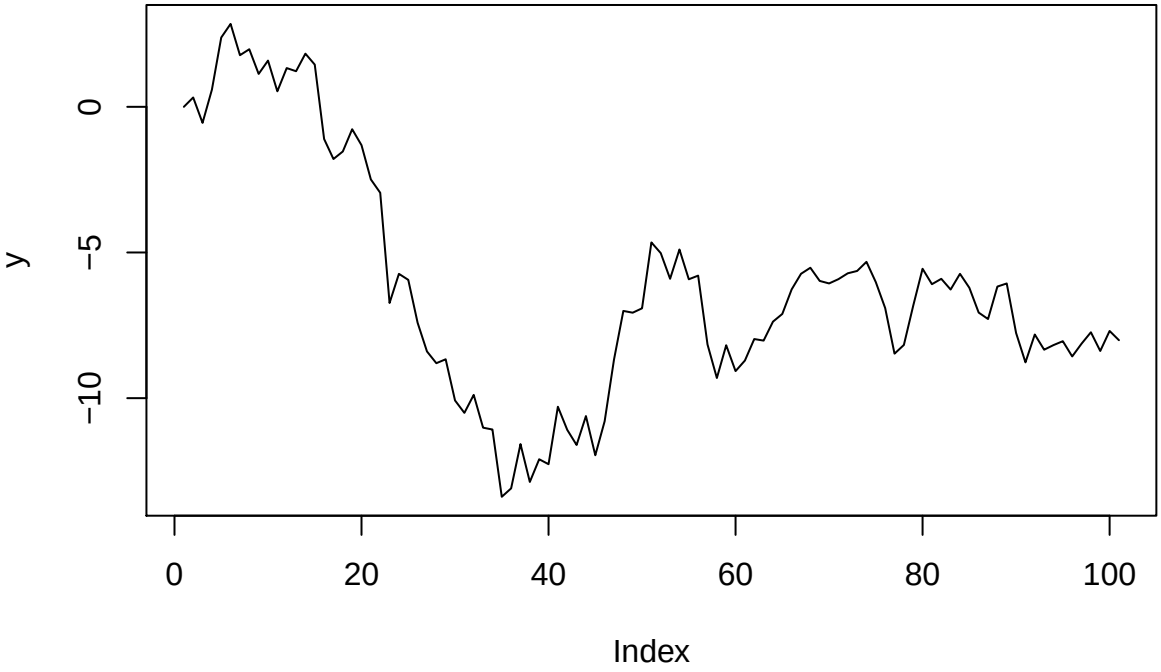
ローカルレベルモデルは次のような状態方程式と観測方程式をもつ。

tを時刻、x_tを状態、y_tを観測値として、

$$y_t = x_t + \epsilon$$
$$x_t = x_{t-1} + \delta$$
$$\epsilon \sim N(0, \sigma_1)$$
$$\delta \sim N(0, \sigma_2)$$

状態方程式はいわゆるランダムウォークになっている。

```
z<-rnorm(100)
y<-0
for (i in 1:100){
  y[i+1]<-y[i]+z[i]
}
plot(y,type='l')
```



推定するパラメータは状態x_t及び誤差の分散 σ_1, σ_2

```
data {
  int T; //
  real y[T]; //
}

parameters {
  real x[T]; //
  real<lower=0> sigma_s; //
  real<lower=0> sigma_o; //
}

model {
```

```
  //
  for (i in 1:T){
    y[i] ~ normal(x[i], sigma_o);
  }
  //
  for (i in 2:T){
    x[i] ~ normal(x[i-1], sigma_s);
  }
}
```

```
#
Nile
```

```
## Time Series:
## Start = 1871
## End = 1970
## Frequency = 1
##   [1] 1120 1160  963 1210 1160 1160  813 1230 1370 1140  995  935 1110  994
##  [15] 1020  960 1180  799  958 1140 1100 1210 1150 1250 1260 1220 1030 1100
##  [29]  774  840  874  694  940  833  701  916  692 1020 1050  969  831  726
##  [43]  456  824  702 1120 1100  832  764  821  768  845  864  862  698  845
##  [57]  744  796 1040  759  781  865  845  944  984  897  822 1010  771  676
##  [71]  649  846  812  742  801 1040  860  874  848  890  744  749  838 1050
##  [85]  918  986  797  923  975  815 1020  906  901 1170  912  746  919  718
##  [99]  714  740
```

```
y<-list(y=as.numeric(Nile),T=length(Nile))
y
```

```
## $y
##   [1] 1120 1160  963 1210 1160 1160  813 1230 1370 1140  995  935 1110  994
##  [15] 1020  960 1180  799  958 1140 1100 1210 1150 1250 1260 1220 1030 1100
##  [29]  774  840  874  694  940  833  701  916  692 1020 1050  969  831  726
##  [43]  456  824  702 1120 1100  832  764  821  768  845  864  862  698  845
##  [57]  744  796 1040  759  781  865  845  944  984  897  822 1010  771  676
##  [71]  649  846  812  742  801 1040  860  874  848  890  744  749  838 1050
##  [85]  918  986  797  923  975  815 1020  906  901 1170  912  746  919  718
##  [99]  714  740
##
## $T
## [1] 100
```

```
library(rstan)
fit <- stan(file = 'locallevel.stan', data = y,
                iter = 4000, chains = 4)
```

```
## In file included from file29a3c49a119.cpp:8:
## In file included from /usr/local/lib/R/3.4/site-library/StanHeaders/include/src/stan/model/model_heade
## In file included from /usr/local/lib/R/3.4/site-library/StanHeaders/include/stan/math.hpp:4:
## In file included from /usr/local/lib/R/3.4/site-library/StanHeaders/include/stan/math/rev/mat.hpp:4:
## In file included from /usr/local/lib/R/3.4/site-library/StanHeaders/include/stan/math/rev/core.hpp:12:
## In file included from /usr/local/lib/R/3.4/site-library/StanHeaders/include/stan/math/rev/core/gevv_vv
## In file included from /usr/local/lib/R/3.4/site-library/StanHeaders/include/stan/math/rev/core/var.hpp
## In file included from /usr/local/lib/R/3.4/site-library/BH/include/boost/math/tools/config.hpp:13:
## In file included from /usr/local/lib/R/3.4/site-library/BH/include/boost/config.hpp:39:
## /usr/local/lib/R/3.4/site-library/BH/include/boost/config/compiler/clang.hpp:200:11: warning: 'BOOST_
## #  define BOOST_NO_CXX11_RVALUE_REFERENCES
```

```
##           ^
## <command line>:6:9: note: previous definition is here
## #define BOOST_NO_CXX11_RVALUE_REFERENCES 1
##         ^
## In file included from file29a3c49a119.cpp:8:
## In file included from /usr/local/lib/R/3.4/site-library/StanHeaders/include/src/stan/model/model_heade
## In file included from /usr/local/lib/R/3.4/site-library/StanHeaders/include/stan/math.hpp:4:
## In file included from /usr/local/lib/R/3.4/site-library/StanHeaders/include/stan/math/rev/mat.hpp:4:
## In file included from /usr/local/lib/R/3.4/site-library/StanHeaders/include/stan/math/rev/core.hpp:14:
## In file included from /usr/local/lib/R/3.4/site-library/StanHeaders/include/stan/math/rev/core/matrix_
## In file included from /usr/local/lib/R/3.4/site-library/StanHeaders/include/stan/math/rev/mat/fun/Eige
## In file included from /usr/local/lib/R/3.4/site-library/StanHeaders/include/stan/math/prim/mat/fun/Eig
## In file included from /usr/local/lib/R/3.4/site-library/RcppEigen/include/Eigen/Dense:1:
## In file included from /usr/local/lib/R/3.4/site-library/RcppEigen/include/Eigen/Core:531:
## /usr/local/lib/R/3.4/site-library/RcppEigen/include/Eigen/src/Core/util/ReenableStupidWarnings.h:10:3
##     #pragma clang diagnostic pop
##                              ^
## In file included from file29a3c49a119.cpp:8:
## In file included from /usr/local/lib/R/3.4/site-library/StanHeaders/include/src/stan/model/model_heade
## In file included from /usr/local/lib/R/3.4/site-library/StanHeaders/include/stan/math.hpp:4:
## In file included from /usr/local/lib/R/3.4/site-library/StanHeaders/include/stan/math/rev/mat.hpp:4:
## In file included from /usr/local/lib/R/3.4/site-library/StanHeaders/include/stan/math/rev/core.hpp:14:
## In file included from /usr/local/lib/R/3.4/site-library/StanHeaders/include/stan/math/rev/core/matrix_
## In file included from /usr/local/lib/R/3.4/site-library/StanHeaders/include/stan/math/rev/mat/fun/Eige
## In file included from /usr/local/lib/R/3.4/site-library/StanHeaders/include/stan/math/prim/mat/fun/Eig
## In file included from /usr/local/lib/R/3.4/site-library/RcppEigen/include/Eigen/Dense:2:
## In file included from /usr/local/lib/R/3.4/site-library/RcppEigen/include/Eigen/LU:47:
## /usr/local/lib/R/3.4/site-library/RcppEigen/include/Eigen/src/Core/util/ReenableStupidWarnings.h:10:3
##     #pragma clang diagnostic pop
##                              ^
## In file included from file29a3c49a119.cpp:8:
## In file included from /usr/local/lib/R/3.4/site-library/StanHeaders/include/src/stan/model/model_heade
## In file included from /usr/local/lib/R/3.4/site-library/StanHeaders/include/stan/math.hpp:4:
## In file included from /usr/local/lib/R/3.4/site-library/StanHeaders/include/stan/math/rev/mat.hpp:4:
## In file included from /usr/local/lib/R/3.4/site-library/StanHeaders/include/stan/math/rev/core.hpp:14:
## In file included from /usr/local/lib/R/3.4/site-library/StanHeaders/include/stan/math/rev/core/matrix_
## In file included from /usr/local/lib/R/3.4/site-library/StanHeaders/include/stan/math/rev/mat/fun/Eige
## In file included from /usr/local/lib/R/3.4/site-library/StanHeaders/include/stan/math/prim/mat/fun/Eig
## In file included from /usr/local/lib/R/3.4/site-library/RcppEigen/include/Eigen/Dense:3:
## In file included from /usr/local/lib/R/3.4/site-library/RcppEigen/include/Eigen/Cholesky:12:
## In file included from /usr/local/lib/R/3.4/site-library/RcppEigen/include/Eigen/Jacobi:29:
## /usr/local/lib/R/3.4/site-library/RcppEigen/include/Eigen/src/Core/util/ReenableStupidWarnings.h:10:3
##     #pragma clang diagnostic pop
##                              ^
## In file included from file29a3c49a119.cpp:8:
## In file included from /usr/local/lib/R/3.4/site-library/StanHeaders/include/src/stan/model/model_heade
## In file included from /usr/local/lib/R/3.4/site-library/StanHeaders/include/stan/math.hpp:4:
## In file included from /usr/local/lib/R/3.4/site-library/StanHeaders/include/stan/math/rev/mat.hpp:4:
## In file included from /usr/local/lib/R/3.4/site-library/StanHeaders/include/stan/math/rev/core.hpp:14:
## In file included from /usr/local/lib/R/3.4/site-library/StanHeaders/include/stan/math/rev/core/matrix_
## In file included from /usr/local/lib/R/3.4/site-library/StanHeaders/include/stan/math/rev/mat/fun/Eige
## In file included from /usr/local/lib/R/3.4/site-library/StanHeaders/include/stan/math/prim/mat/fun/Eig
## In file included from /usr/local/lib/R/3.4/site-library/RcppEigen/include/Eigen/Dense:3:
## In file included from /usr/local/lib/R/3.4/site-library/RcppEigen/include/Eigen/Cholesky:43:
```

```
## /usr/local/lib/R/3.4/site-library/RcppEigen/include/Eigen/src/Core/util/ReenableStupidWarnings.h:10:3
##     #pragma clang diagnostic pop
##                              ^
## In file included from file29a3c49a119.cpp:8:
## In file included from /usr/local/lib/R/3.4/site-library/StanHeaders/include/src/stan/model/model_heade
## In file included from /usr/local/lib/R/3.4/site-library/StanHeaders/include/stan/math.hpp:4:
## In file included from /usr/local/lib/R/3.4/site-library/StanHeaders/include/stan/math/rev/mat.hpp:4:
## In file included from /usr/local/lib/R/3.4/site-library/StanHeaders/include/stan/math/rev/core.hpp:14:
## In file included from /usr/local/lib/R/3.4/site-library/StanHeaders/include/stan/math/rev/core/matrix_
## In file included from /usr/local/lib/R/3.4/site-library/StanHeaders/include/stan/math/rev/mat/fun/Eige
## In file included from /usr/local/lib/R/3.4/site-library/StanHeaders/include/stan/math/prim/mat/fun/Eig
## In file included from /usr/local/lib/R/3.4/site-library/RcppEigen/include/Eigen/Dense:4:
## In file included from /usr/local/lib/R/3.4/site-library/RcppEigen/include/Eigen/QR:17:
## In file included from /usr/local/lib/R/3.4/site-library/RcppEigen/include/Eigen/Householder:27:
## /usr/local/lib/R/3.4/site-library/RcppEigen/include/Eigen/src/Core/util/ReenableStupidWarnings.h:10:3
##     #pragma clang diagnostic pop
##                              ^
## In file included from file29a3c49a119.cpp:8:
## In file included from /usr/local/lib/R/3.4/site-library/StanHeaders/include/src/stan/model/model_heade
## In file included from /usr/local/lib/R/3.4/site-library/StanHeaders/include/stan/math.hpp:4:
## In file included from /usr/local/lib/R/3.4/site-library/StanHeaders/include/stan/math/rev/mat.hpp:4:
## In file included from /usr/local/lib/R/3.4/site-library/StanHeaders/include/stan/math/rev/core.hpp:14:
## In file included from /usr/local/lib/R/3.4/site-library/StanHeaders/include/stan/math/rev/core/matrix_
## In file included from /usr/local/lib/R/3.4/site-library/StanHeaders/include/stan/math/rev/mat/fun/Eige
## In file included from /usr/local/lib/R/3.4/site-library/StanHeaders/include/stan/math/prim/mat/fun/Eig
## In file included from /usr/local/lib/R/3.4/site-library/RcppEigen/include/Eigen/Dense:5:
## In file included from /usr/local/lib/R/3.4/site-library/RcppEigen/include/Eigen/SVD:48:
## /usr/local/lib/R/3.4/site-library/RcppEigen/include/Eigen/src/Core/util/ReenableStupidWarnings.h:10:3
##     #pragma clang diagnostic pop
##                              ^
## In file included from file29a3c49a119.cpp:8:
## In file included from /usr/local/lib/R/3.4/site-library/StanHeaders/include/src/stan/model/model_heade
## In file included from /usr/local/lib/R/3.4/site-library/StanHeaders/include/stan/math.hpp:4:
## In file included from /usr/local/lib/R/3.4/site-library/StanHeaders/include/stan/math/rev/mat.hpp:4:
## In file included from /usr/local/lib/R/3.4/site-library/StanHeaders/include/stan/math/rev/core.hpp:14:
## In file included from /usr/local/lib/R/3.4/site-library/StanHeaders/include/stan/math/rev/core/matrix_
## In file included from /usr/local/lib/R/3.4/site-library/StanHeaders/include/stan/math/rev/mat/fun/Eige
## In file included from /usr/local/lib/R/3.4/site-library/StanHeaders/include/stan/math/prim/mat/fun/Eig
## In file included from /usr/local/lib/R/3.4/site-library/RcppEigen/include/Eigen/Dense:6:
## In file included from /usr/local/lib/R/3.4/site-library/RcppEigen/include/Eigen/Geometry:58:
## /usr/local/lib/R/3.4/site-library/RcppEigen/include/Eigen/src/Core/util/ReenableStupidWarnings.h:10:3
##     #pragma clang diagnostic pop
##                              ^
## In file included from file29a3c49a119.cpp:8:
## In file included from /usr/local/lib/R/3.4/site-library/StanHeaders/include/src/stan/model/model_heade
## In file included from /usr/local/lib/R/3.4/site-library/StanHeaders/include/stan/math.hpp:4:
## In file included from /usr/local/lib/R/3.4/site-library/StanHeaders/include/stan/math/rev/mat.hpp:4:
## In file included from /usr/local/lib/R/3.4/site-library/StanHeaders/include/stan/math/rev/core.hpp:14:
## In file included from /usr/local/lib/R/3.4/site-library/StanHeaders/include/stan/math/rev/core/matrix_
## In file included from /usr/local/lib/R/3.4/site-library/StanHeaders/include/stan/math/rev/mat/fun/Eige
## In file included from /usr/local/lib/R/3.4/site-library/StanHeaders/include/stan/math/prim/mat/fun/Eig
## In file included from /usr/local/lib/R/3.4/site-library/RcppEigen/include/Eigen/Dense:7:
## In file included from /usr/local/lib/R/3.4/site-library/RcppEigen/include/Eigen/Eigenvalues:58:
## /usr/local/lib/R/3.4/site-library/RcppEigen/include/Eigen/src/Core/util/ReenableStupidWarnings.h:10:3
```

```
##     #pragma clang diagnostic pop
##                              ^
## In file included from file29a3c49a119.cpp:8:
## In file included from /usr/local/lib/R/3.4/site-library/StanHeaders/include/src/stan/model/model_heade
## In file included from /usr/local/lib/R/3.4/site-library/StanHeaders/include/stan/math.hpp:4:
## In file included from /usr/local/lib/R/3.4/site-library/StanHeaders/include/stan/math/rev/mat.hpp:12:
## In file included from /usr/local/lib/R/3.4/site-library/StanHeaders/include/stan/math/prim/mat.hpp:83:
## In file included from /usr/local/lib/R/3.4/site-library/StanHeaders/include/stan/math/prim/mat/fun/csr
## In file included from /usr/local/lib/R/3.4/site-library/RcppEigen/include/Eigen/Sparse:26:
## In file included from /usr/local/lib/R/3.4/site-library/RcppEigen/include/Eigen/SparseCore:66:
## /usr/local/lib/R/3.4/site-library/RcppEigen/include/Eigen/src/Core/util/ReenableStupidWarnings.h:10:3
##     #pragma clang diagnostic pop
##                              ^
## In file included from file29a3c49a119.cpp:8:
## In file included from /usr/local/lib/R/3.4/site-library/StanHeaders/include/src/stan/model/model_heade
## In file included from /usr/local/lib/R/3.4/site-library/StanHeaders/include/stan/math.hpp:4:
## In file included from /usr/local/lib/R/3.4/site-library/StanHeaders/include/stan/math/rev/mat.hpp:12:
## In file included from /usr/local/lib/R/3.4/site-library/StanHeaders/include/stan/math/prim/mat.hpp:83:
## In file included from /usr/local/lib/R/3.4/site-library/StanHeaders/include/stan/math/prim/mat/fun/csr
## In file included from /usr/local/lib/R/3.4/site-library/RcppEigen/include/Eigen/Sparse:27:
## In file included from /usr/local/lib/R/3.4/site-library/RcppEigen/include/Eigen/OrderingMethods:71:
## /usr/local/lib/R/3.4/site-library/RcppEigen/include/Eigen/src/Core/util/ReenableStupidWarnings.h:10:3
##     #pragma clang diagnostic pop
##                              ^
## In file included from file29a3c49a119.cpp:8:
## In file included from /usr/local/lib/R/3.4/site-library/StanHeaders/include/src/stan/model/model_heade
## In file included from /usr/local/lib/R/3.4/site-library/StanHeaders/include/stan/math.hpp:4:
## In file included from /usr/local/lib/R/3.4/site-library/StanHeaders/include/stan/math/rev/mat.hpp:12:
## In file included from /usr/local/lib/R/3.4/site-library/StanHeaders/include/stan/math/prim/mat.hpp:83:
## In file included from /usr/local/lib/R/3.4/site-library/StanHeaders/include/stan/math/prim/mat/fun/csr
## In file included from /usr/local/lib/R/3.4/site-library/RcppEigen/include/Eigen/Sparse:29:
## In file included from /usr/local/lib/R/3.4/site-library/RcppEigen/include/Eigen/SparseCholesky:43:
## /usr/local/lib/R/3.4/site-library/RcppEigen/include/Eigen/src/Core/util/ReenableStupidWarnings.h:10:3
##     #pragma clang diagnostic pop
##                              ^
## In file included from file29a3c49a119.cpp:8:
## In file included from /usr/local/lib/R/3.4/site-library/StanHeaders/include/src/stan/model/model_heade
## In file included from /usr/local/lib/R/3.4/site-library/StanHeaders/include/stan/math.hpp:4:
## In file included from /usr/local/lib/R/3.4/site-library/StanHeaders/include/stan/math/rev/mat.hpp:12:
## In file included from /usr/local/lib/R/3.4/site-library/StanHeaders/include/stan/math/prim/mat.hpp:83:
## In file included from /usr/local/lib/R/3.4/site-library/StanHeaders/include/stan/math/prim/mat/fun/csr
## In file included from /usr/local/lib/R/3.4/site-library/RcppEigen/include/Eigen/Sparse:32:
## In file included from /usr/local/lib/R/3.4/site-library/RcppEigen/include/Eigen/SparseQR:35:
## /usr/local/lib/R/3.4/site-library/RcppEigen/include/Eigen/src/Core/util/ReenableStupidWarnings.h:10:3
##     #pragma clang diagnostic pop
##                              ^
## In file included from file29a3c49a119.cpp:8:
## In file included from /usr/local/lib/R/3.4/site-library/StanHeaders/include/src/stan/model/model_heade
## In file included from /usr/local/lib/R/3.4/site-library/StanHeaders/include/stan/math.hpp:4:
## In file included from /usr/local/lib/R/3.4/site-library/StanHeaders/include/stan/math/rev/mat.hpp:12:
## In file included from /usr/local/lib/R/3.4/site-library/StanHeaders/include/stan/math/prim/mat.hpp:83:
## In file included from /usr/local/lib/R/3.4/site-library/StanHeaders/include/stan/math/prim/mat/fun/csr
## In file included from /usr/local/lib/R/3.4/site-library/RcppEigen/include/Eigen/Sparse:33:
## In file included from /usr/local/lib/R/3.4/site-library/RcppEigen/include/Eigen/IterativeLinearSolvers
```

```
## /usr/local/lib/R/3.4/site-library/RcppEigen/include/Eigen/src/Core/util/ReenableStupidWarnings.h:10:3
##     #pragma clang diagnostic pop
##                              ^
## 14 warnings generated.
##
## SAMPLING FOR MODEL 'locallevel' NOW (CHAIN 1).
##
## Gradient evaluation took 0.00013 seconds
## 1000 transitions using 10 leapfrog steps per transition would take 1.3 seconds.
## Adjust your expectations accordingly!
##
##
## Iteration:    1 / 4000 [  0%]  (Warmup)
## Iteration:  400 / 4000 [ 10%]  (Warmup)
## Iteration:  800 / 4000 [ 20%]  (Warmup)
## Iteration: 1200 / 4000 [ 30%]  (Warmup)
## Iteration: 1600 / 4000 [ 40%]  (Warmup)
## Iteration: 2000 / 4000 [ 50%]  (Warmup)
## Iteration: 2001 / 4000 [ 50%]  (Sampling)
## Iteration: 2400 / 4000 [ 60%]  (Sampling)
## Iteration: 2800 / 4000 [ 70%]  (Sampling)
## Iteration: 3200 / 4000 [ 80%]  (Sampling)
## Iteration: 3600 / 4000 [ 90%]  (Sampling)
## Iteration: 4000 / 4000 [100%]  (Sampling)
##
##  Elapsed Time: 3.54657 seconds (Warm-up)
##                2.4615 seconds (Sampling)
##                6.00807 seconds (Total)
##
##
## SAMPLING FOR MODEL 'locallevel' NOW (CHAIN 2).
##
## Gradient evaluation took 3.2e-05 seconds
## 1000 transitions using 10 leapfrog steps per transition would take 0.32 seconds.
## Adjust your expectations accordingly!
##
##
## Iteration:    1 / 4000 [  0%]  (Warmup)
## Iteration:  400 / 4000 [ 10%]  (Warmup)
## Iteration:  800 / 4000 [ 20%]  (Warmup)
## Iteration: 1200 / 4000 [ 30%]  (Warmup)
## Iteration: 1600 / 4000 [ 40%]  (Warmup)
## Iteration: 2000 / 4000 [ 50%]  (Warmup)
## Iteration: 2001 / 4000 [ 50%]  (Sampling)
## Iteration: 2400 / 4000 [ 60%]  (Sampling)
## Iteration: 2800 / 4000 [ 70%]  (Sampling)
## Iteration: 3200 / 4000 [ 80%]  (Sampling)
## Iteration: 3600 / 4000 [ 90%]  (Sampling)
## Iteration: 4000 / 4000 [100%]  (Sampling)
##
##  Elapsed Time: 3.70241 seconds (Warm-up)
##                2.47072 seconds (Sampling)
##                6.17314 seconds (Total)
##
```

```
##
## SAMPLING FOR MODEL 'locallevel' NOW (CHAIN 3).
##
## Gradient evaluation took 2.7e-05 seconds
## 1000 transitions using 10 leapfrog steps per transition would take 0.27 seconds.
## Adjust your expectations accordingly!
##
##
## Iteration:    1 / 4000 [  0%]  (Warmup)
## Iteration:  400 / 4000 [ 10%]  (Warmup)
## Iteration:  800 / 4000 [ 20%]  (Warmup)
## Iteration: 1200 / 4000 [ 30%]  (Warmup)
## Iteration: 1600 / 4000 [ 40%]  (Warmup)
## Iteration: 2000 / 4000 [ 50%]  (Warmup)
## Iteration: 2001 / 4000 [ 50%]  (Sampling)
## Iteration: 2400 / 4000 [ 60%]  (Sampling)
## Iteration: 2800 / 4000 [ 70%]  (Sampling)
## Iteration: 3200 / 4000 [ 80%]  (Sampling)
## Iteration: 3600 / 4000 [ 90%]  (Sampling)
## Iteration: 4000 / 4000 [100%]  (Sampling)
##
##  Elapsed Time: 3.98527 seconds (Warm-up)
##                2.57264 seconds (Sampling)
##                6.55791 seconds (Total)
##
##
## SAMPLING FOR MODEL 'locallevel' NOW (CHAIN 4).
##
## Gradient evaluation took 4.7e-05 seconds
## 1000 transitions using 10 leapfrog steps per transition would take 0.47 seconds.
## Adjust your expectations accordingly!
##
##
## Iteration:    1 / 4000 [  0%]  (Warmup)
## Iteration:  400 / 4000 [ 10%]  (Warmup)
## Iteration:  800 / 4000 [ 20%]  (Warmup)
## Iteration: 1200 / 4000 [ 30%]  (Warmup)
## Iteration: 1600 / 4000 [ 40%]  (Warmup)
## Iteration: 2000 / 4000 [ 50%]  (Warmup)
## Iteration: 2001 / 4000 [ 50%]  (Sampling)
## Iteration: 2400 / 4000 [ 60%]  (Sampling)
## Iteration: 2800 / 4000 [ 70%]  (Sampling)
## Iteration: 3200 / 4000 [ 80%]  (Sampling)
## Iteration: 3600 / 4000 [ 90%]  (Sampling)
## Iteration: 4000 / 4000 [100%]  (Sampling)
##
##  Elapsed Time: 4.06486 seconds (Warm-up)
##                2.14048 seconds (Sampling)
##                6.20533 seconds (Total)
## Warning: There were 4 chains where the estimated Bayesian Fraction of Missing Information was low. See
## http://mc-stan.org/misc/warnings.html#bfmi-low
## Warning: Examine the pairs() plot to diagnose sampling problems
```

```r
#
x<-rstan::extract(fit)$x
#
head(x)
```

```
##
## iterations     [,1]     [,2]     [,3]     [,4]     [,5]     [,6]     [,7]
##       [1,] 1265.328 1267.142 1223.893 1153.454 1164.526 1112.658 1110.236
##       [2,] 1141.017 1106.692 1110.546 1117.360 1102.643 1105.184 1103.807
##       [3,] 1065.611 1084.028 1094.086 1126.755 1101.137 1070.741 1068.906
##       [4,] 1139.964 1163.316 1091.057 1162.989 1149.804 1104.176 1066.598
##       [5,] 1217.745 1176.010 1112.151 1179.402 1093.452 1213.066 1121.999
##       [6,] 1201.200 1179.798 1176.760 1230.589 1202.392 1164.031 1149.640
##
## iterations     [,8]     [,9]    [,10]    [,11]     [,12]    [,13]    [,14]
##       [1,] 1145.590 1107.997 1115.071 1109.120 1094.4635 1080.801 1055.072
##       [2,] 1124.995 1144.721 1118.354 1097.247 1090.7412 1067.894 1129.136
##       [3,] 1079.317 1070.651 1051.433 1057.936 1061.5220 1033.660 1044.383
##       [4,] 1048.310 1122.530 1177.150 1110.292 1087.0930 1082.915 1042.008
##       [5,] 1158.727 1151.866 1142.224 1101.127  961.5538 1007.631 1025.288
##       [6,] 1151.376 1133.712 1139.472 1113.909 1151.6361 1132.299 1095.761
##
## iterations     [,15]     [,16]     [,17]     [,18]     [,19]     [,20]
##       [1,] 1091.4043 1060.5668 1073.2017 1078.6491 1071.9905 1100.0328
##       [2,] 1092.9798 1121.7157 1094.9829 1087.1165 1037.2745 1100.1211
##       [3,]  997.7795  988.8451  974.1815 1002.9541  922.1050  986.5842
##       [4,] 1032.0900 1055.6537 1066.4008 1023.2785  971.7413 1019.6242
##       [5,]  967.3521  964.2265 1040.0241  966.0199  982.0146 1100.0631
##       [6,] 1092.0312 1093.9897 1054.5498 1074.8022 1072.5627 1087.4056
##
## iterations    [,21]    [,22]    [,23]    [,24]    [,25]    [,26]     [,27]
##       [1,] 1116.771 1091.835 1064.292 1084.718 1080.698 1052.213  999.5401
##       [2,] 1145.279 1097.828 1180.026 1154.733 1144.921 1116.414 1055.8782
##       [3,] 1087.481 1113.733 1123.493 1183.317 1110.834 1138.921 1127.5048
##       [4,] 1064.973 1116.902 1078.642 1106.303 1276.063 1213.988 1104.3394
##       [5,] 1031.808 1166.163 1197.186 1132.392 1196.200 1103.780 1018.6142
##       [6,] 1096.259 1127.582 1105.907 1118.433 1107.306 1098.303 1066.0389
##
## iterations     [,28]     [,29]    [,30]    [,31]    [,32]    [,33]
##       [1,]  956.3460  916.4992 883.6367 845.8709 818.4256 839.3030
##       [2,] 1019.2203 1001.5097 937.0166 902.0835 863.7876 900.1358
##       [3,] 1062.3283 1011.1932 955.0645 956.2741 882.1596 868.3047
##       [4,]  982.7758  965.1653 903.6504 891.6829 882.0223 828.0452
##       [5,] 1016.5508  909.9248 856.8577 781.4422 802.3278 819.1904
##       [6,] 1055.9297 1018.8305 959.8969 920.2607 905.0223 883.9429
##
## iterations    [,34]    [,35]    [,36]    [,37]    [,38]    [,39]    [,40]
##       [1,] 837.0882 863.6213 899.8530 903.5524 886.3301 842.6479 826.3629
##       [2,] 881.7438 867.3401 845.8323 852.7994 861.6207 843.5253 865.4063
##       [3,] 849.6707 858.2173 852.5246 864.7252 952.1667 943.8864 909.2843
##       [4,] 818.2263 809.3930 817.0123 845.7491 939.2353 920.3837 928.8175
##       [5,] 865.0920 852.7553 807.4858 800.3931 979.6895 924.9318 865.8403
##       [6,] 893.8284 917.8047 885.1938 887.7603 876.6841 889.6961 858.0790
##
```

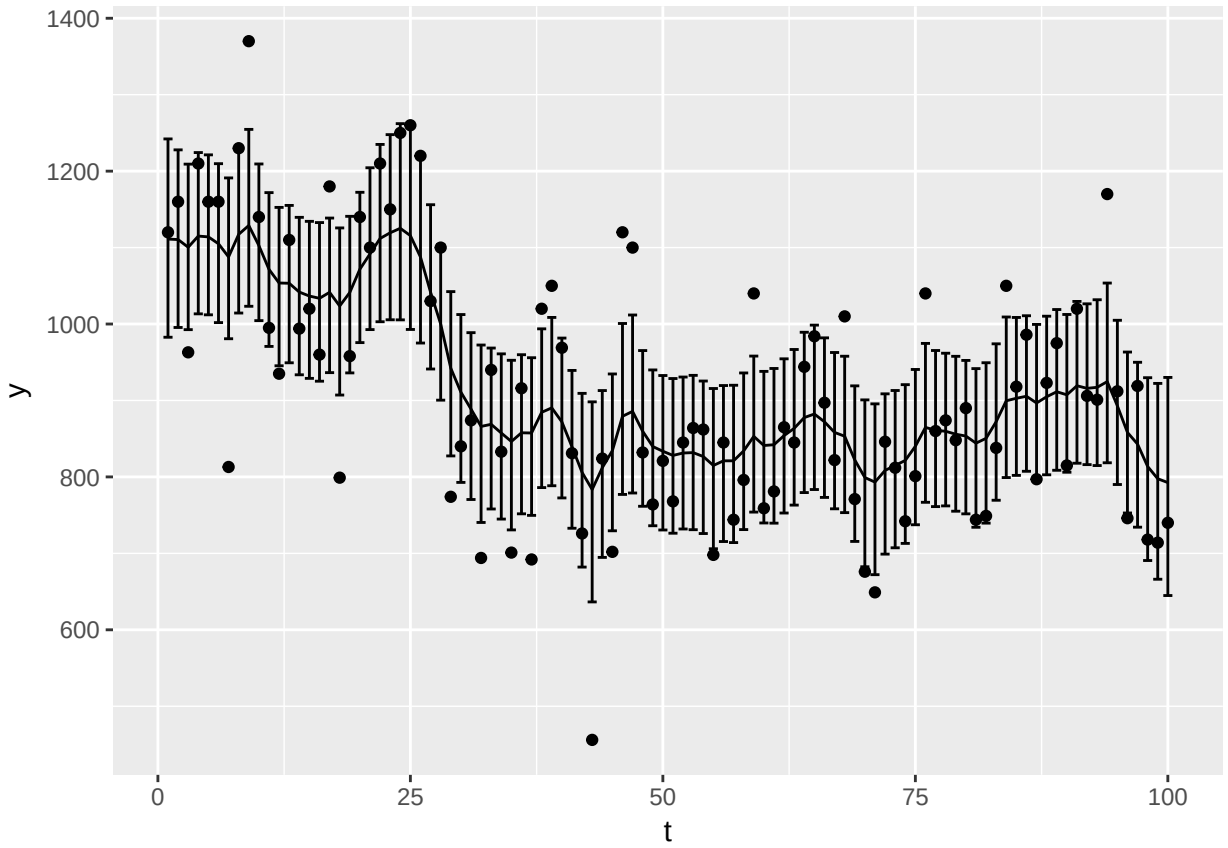
```
## iterations    [,41]    [,42]    [,43]    [,44]    [,45]    [,46]    [,47]
##       [1,] 780.2688 773.9639 733.5575 732.6733 770.8877 805.2349 827.4255
##       [2,] 867.0219 847.3333 813.9727 836.2897 876.3254 854.5042 844.1683
##       [3,] 860.6043 822.7486 742.0471 829.0128 781.0360 846.7639 869.5207
##       [4,] 881.7945 819.1369 764.8084 789.2491 765.7091 847.5225 931.8123
##       [5,] 886.6857 830.8510 729.3193 763.6230 843.1439 921.9990 889.1847
##       [6,] 871.5138 861.1631 860.2498 833.6441 865.5550 870.8242 864.9500
##
## iterations    [,48]    [,49]    [,50]    [,51]    [,52]    [,53]    [,54]
##       [1,] 820.3059 835.7282 802.7880 811.8301 812.7103 894.6444 895.6890
##       [2,] 826.3155 901.8976 903.2227 912.4045 897.2354 872.6986 875.9515
##       [3,] 819.8731 782.9252 802.3506 726.9155 731.7984 834.4573 849.5791
##       [4,] 907.8631 846.5696 862.7203 838.1970 829.5417 835.8464 865.1859
##       [5,] 895.2603 839.5904 871.7531 828.8251 809.2656 859.0242 865.4351
##       [6,] 848.7475 813.6656 816.8793 804.1639 800.9935 804.7276 792.7383
##
## iterations    [,55]    [,56]    [,57]    [,58]    [,59]    [,60]    [,61]
##       [1,] 913.7594 900.3593 834.7732 777.0410 823.4012 806.8534 765.1363
##       [2,] 872.5953 923.3350 903.8707 923.7574 901.2872 880.7993 902.5146
##       [3,] 875.2509 833.7024 800.3698 756.4527 849.6427 907.5513 859.0093
##       [4,] 800.2377 827.5798 834.9799 823.8701 873.6154 915.6471 864.8090
##       [5,] 823.9672 766.1271 759.9173 793.3695 788.6779 820.9503 786.1275
##       [6,] 806.2978 797.4454 797.3591 806.1452 821.7270 826.3692 867.9408
##
## iterations    [,62]    [,63]    [,64]    [,65]    [,66]    [,67]    [,68]
##       [1,] 813.5879 799.9209 802.3753 836.3621 813.0315 817.6796 821.1365
##       [2,] 885.1964 881.7007 920.7801 909.3448 899.8326 885.0369 818.8063
##       [3,] 873.9205 881.7930 881.6002 965.4797 933.7719 884.5719 861.8359
##       [4,] 842.6648 840.2274 877.5201 884.2002 868.2981 905.7498 951.7780
##       [5,] 801.1437 769.6911 803.8786 793.7137 861.3503 868.1659 867.8012
##       [6,] 847.4579 854.2621 899.9570 910.6563 909.2126 877.9099 860.7687
##
## iterations    [,69]    [,70]    [,71]    [,72]    [,73]    [,74]    [,75]
##       [1,] 851.5417 862.0603 905.6509 900.8220 891.0017 861.1943 826.8637
##       [2,] 800.3112 768.1633 776.3171 789.3177 820.3406 845.1631 869.5826
##       [3,] 855.1790 880.1294 825.9414 821.1790 810.0013 794.5811 820.6980
##       [4,] 857.2019 826.7490 782.0973 768.6428 753.7297 818.5259 848.1593
##       [5,] 814.4283 747.2786 770.1837 804.2405 871.1112 925.6319 886.6550
##       [6,] 826.2093 799.7592 814.7031 825.8050 836.6225 822.9081 824.8929
##
## iterations    [,76]    [,77]    [,78]    [,79]    [,80]    [,81]    [,82]
##       [1,] 842.2546 801.2493 817.4373 801.5009 783.4245 773.4471 835.7449
##       [2,] 828.4330 812.5098 849.7926 830.5783 784.7898 804.9183 818.6881
##       [3,] 834.0012 870.4583 797.4502 836.3139 822.9880 814.7187 851.4743
##       [4,] 863.2502 982.8369 897.2158 842.7272 798.3852 821.2727 835.9794
##       [5,] 907.4972 862.4240 760.7995 825.1875 826.8077 775.1191 852.3011
##       [6,] 828.0621 862.5074 865.2167 874.2638 871.4013 848.5806 810.6822
##
## iterations    [,83]    [,84]    [,85]    [,86]     [,87]    [,88]
##       [1,] 844.8669 857.9237 852.6431 865.4939  887.4060 909.9705
##       [2,] 844.2442 909.0511 914.8031 941.4777  934.6643 901.5794
##       [3,] 860.8572 948.1799 973.1492 996.1817 1012.0534 956.9778
##       [4,] 804.3115 891.4808 865.4240 911.0174  900.5878 956.0578
##       [5,] 869.2094 889.0590 838.9656 891.1646  870.0722 953.4993
```

```
##       [6,] 823.0541 814.2413 823.7380 830.5673  824.5801 848.4116
##
## iterations     [,89]    [,90]     [,91]    [,92]    [,93]    [,94]
##       [1,]  985.2748 938.8023  936.7414 910.2064 944.2776 928.2265
##       [2,]  951.5146 965.6878  949.4773 906.7138 891.2026 869.1070
##       [3,] 1008.1641 976.4950  964.1283 882.1334 921.1213 995.5548
##       [4,]  982.4295 961.3554 1014.2617 935.9672 963.8301 960.7672
##       [5,] 1020.5065 907.0942  945.1499 938.6485 933.0158 931.2451
##       [6,]  824.8511 829.4118  845.9514 827.0925 854.8608 858.1677
##
## iterations    [,95]    [,96]    [,97]    [,98]    [,99]   [,100]
##       [1,] 898.6970 863.2208 852.9672 795.7996 773.1803 763.7361
##       [2,] 913.5776 807.8359 816.4976 778.7583 754.1777 764.4659
##       [3,] 907.6929 863.2395 840.3238 784.7946 745.5105 691.3077
##       [4,] 921.7407 848.9438 855.6681 880.7204 920.2547 822.0427
##       [5,] 838.7549 828.2207 774.5602 829.8755 742.8715 776.5138
##       [6,] 847.4493 817.1834 818.9047 827.0394 818.6611 788.5570
```

```r
d2<-data.frame(x)
up<-c()
lo<-c()
M<-c()

for(i in 1:length(d2)){
  up[i]<-quantile(d2[,i],0.975)
  lo[i]<-quantile(d2[,i],0.025)
  M[i]<-mean(d2[,i])
}

df<-data.frame(y=y$y, t=1:y$T, x=M, upper=up, lower=lo)
g<-ggplot(data = df, aes(x = t, y = y))
g<-g+geom_point()
g<-g+geom_errorbar(data = df, aes(ymin=lower, ymax=upper, x=t))
g<-g+geom_line(aes(x=t,y=x),data=df)
g
```



トレンドモデル 二回差分のトレンド項を持つ、次のような状態空間モデルを考える。

tを時刻、x_tを状態、y_tを観測値として、

$$y_t = x_t + \epsilon$$

$$x_t - x_{t-1} = x_{t-1} - x_{t-2} + \delta$$

$$\epsilon \sim N(0, \sigma_1)$$

$$\delta \sim N(0, \sigma_2)$$

推定するパラメータは状態x_t及び誤差の分散 σ_1, σ_2

```
data {
  int T; //
  real y[T]; //
}

parameters {
  real x[T]; //
  real<lower=0> sigma_s; //
  real<lower=0> sigma_o; //
}

model {
  //
  for (i in 1:T){
    y[i] ~ normal(x[i], sigma_o);
  }
  //
```

```
  for (i in 3:T){
    x[i] ~ normal(2*x[i-1]-x[i-2], sigma_s);
  }
}
```

```
#
Nile
```

```
## Time Series:
## Start = 1871
## End = 1970
## Frequency = 1
##   [1] 1120 1160  963 1210 1160 1160  813 1230 1370 1140  995  935 1110  994
##  [15] 1020  960 1180  799  958 1140 1100 1210 1150 1250 1260 1220 1030 1100
##  [29]  774  840  874  694  940  833  701  916  692 1020 1050  969  831  726
##  [43]  456  824  702 1120 1100  832  764  821  768  845  864  862  698  845
##  [57]  744  796 1040  759  781  865  845  944  984  897  822 1010  771  676
##  [71]  649  846  812  742  801 1040  860  874  848  890  744  749  838 1050
##  [85]  918  986  797  923  975  815 1020  906  901 1170  912  746  919  718
##  [99]  714  740
```

```
y<-list(y=as.numeric(Nile),T=length(Nile))
y
```

```
## $y
##   [1] 1120 1160  963 1210 1160 1160  813 1230 1370 1140  995  935 1110  994
##  [15] 1020  960 1180  799  958 1140 1100 1210 1150 1250 1260 1220 1030 1100
##  [29]  774  840  874  694  940  833  701  916  692 1020 1050  969  831  726
##  [43]  456  824  702 1120 1100  832  764  821  768  845  864  862  698  845
##  [57]  744  796 1040  759  781  865  845  944  984  897  822 1010  771  676
##  [71]  649  846  812  742  801 1040  860  874  848  890  744  749  838 1050
##  [85]  918  986  797  923  975  815 1020  906  901 1170  912  746  919  718
##  [99]  714  740
##
## $T
## [1] 100
```

```
library(rstan)
fit <- stan(file = 'trendmodel.stan', data = y,
                iter = 4000, chains = 4)
```

```
##
## SAMPLING FOR MODEL '8f4ae7eb030d6a862908f49755c46a85' NOW (CHAIN 1).
##
## Gradient evaluation took 5.6e-05 seconds
## 1000 transitions using 10 leapfrog steps per transition would take 0.56 seconds.
## Adjust your expectations accordingly!
##
##
## Iteration:    1 / 4000 [  0%]  (Warmup)
## Iteration:  400 / 4000 [ 10%]  (Warmup)
## Iteration:  800 / 4000 [ 20%]  (Warmup)
## Iteration: 1200 / 4000 [ 30%]  (Warmup)
## Iteration: 1600 / 4000 [ 40%]  (Warmup)
## Iteration: 2000 / 4000 [ 50%]  (Warmup)
## Iteration: 2001 / 4000 [ 50%]  (Sampling)
## Iteration: 2400 / 4000 [ 60%]  (Sampling)
```

```
## Iteration: 2800 / 4000 [ 70%]  (Sampling)
## Iteration: 3200 / 4000 [ 80%]  (Sampling)
## Iteration: 3600 / 4000 [ 90%]  (Sampling)
## Iteration: 4000 / 4000 [100%]  (Sampling)
##
##  Elapsed Time: 33.3676 seconds (Warm-up)
##                46.9979 seconds (Sampling)
##                80.3655 seconds (Total)
##
##
## SAMPLING FOR MODEL '8f4ae7eb030d6a862908f49755c46a85' NOW (CHAIN 2).
##
## Gradient evaluation took 2.9e-05 seconds
## 1000 transitions using 10 leapfrog steps per transition would take 0.29 seconds.
## Adjust your expectations accordingly!
##
##
## Iteration:    1 / 4000 [  0%]  (Warmup)
## Iteration:  400 / 4000 [ 10%]  (Warmup)
## Iteration:  800 / 4000 [ 20%]  (Warmup)
## Iteration: 1200 / 4000 [ 30%]  (Warmup)
## Iteration: 1600 / 4000 [ 40%]  (Warmup)
## Iteration: 2000 / 4000 [ 50%]  (Warmup)
## Iteration: 2001 / 4000 [ 50%]  (Sampling)
## Iteration: 2400 / 4000 [ 60%]  (Sampling)
## Iteration: 2800 / 4000 [ 70%]  (Sampling)
## Iteration: 3200 / 4000 [ 80%]  (Sampling)
## Iteration: 3600 / 4000 [ 90%]  (Sampling)
## Iteration: 4000 / 4000 [100%]  (Sampling)
##
##  Elapsed Time: 31.9412 seconds (Warm-up)
##                42.8821 seconds (Sampling)
##                74.8233 seconds (Total)
##
##
## SAMPLING FOR MODEL '8f4ae7eb030d6a862908f49755c46a85' NOW (CHAIN 3).
##
## Gradient evaluation took 5.8e-05 seconds
## 1000 transitions using 10 leapfrog steps per transition would take 0.58 seconds.
## Adjust your expectations accordingly!
##
##
## Iteration:    1 / 4000 [  0%]  (Warmup)
## Iteration:  400 / 4000 [ 10%]  (Warmup)
## Iteration:  800 / 4000 [ 20%]  (Warmup)
## Iteration: 1200 / 4000 [ 30%]  (Warmup)
## Iteration: 1600 / 4000 [ 40%]  (Warmup)
## Iteration: 2000 / 4000 [ 50%]  (Warmup)
## Iteration: 2001 / 4000 [ 50%]  (Sampling)
## Iteration: 2400 / 4000 [ 60%]  (Sampling)
## Iteration: 2800 / 4000 [ 70%]  (Sampling)
## Iteration: 3200 / 4000 [ 80%]  (Sampling)
## Iteration: 3600 / 4000 [ 90%]  (Sampling)
## Iteration: 4000 / 4000 [100%]  (Sampling)
```

```
##
##  Elapsed Time: 34.6953 seconds (Warm-up)
##                35.286 seconds (Sampling)
##                69.9813 seconds (Total)
##
##
## SAMPLING FOR MODEL '8f4ae7eb030d6a862908f49755c46a85' NOW (CHAIN 4).
##
## Gradient evaluation took 3.2e-05 seconds
## 1000 transitions using 10 leapfrog steps per transition would take 0.32 seconds.
## Adjust your expectations accordingly!
##
##
## Iteration:    1 / 4000 [  0%]  (Warmup)
## Iteration:  400 / 4000 [ 10%]  (Warmup)
## Iteration:  800 / 4000 [ 20%]  (Warmup)
## Iteration: 1200 / 4000 [ 30%]  (Warmup)
## Iteration: 1600 / 4000 [ 40%]  (Warmup)
## Iteration: 2000 / 4000 [ 50%]  (Warmup)
## Iteration: 2001 / 4000 [ 50%]  (Sampling)
## Iteration: 2400 / 4000 [ 60%]  (Sampling)
## Iteration: 2800 / 4000 [ 70%]  (Sampling)
## Iteration: 3200 / 4000 [ 80%]  (Sampling)
## Iteration: 3600 / 4000 [ 90%]  (Sampling)
## Iteration: 4000 / 4000 [100%]  (Sampling)
##
##  Elapsed Time: 31.3 seconds (Warm-up)
##                33.2997 seconds (Sampling)
##                64.5997 seconds (Total)
```

```
## Warning: There were 65 divergent transitions after warmup. Increasing adapt_delta above 0.8 may help. See
## http://mc-stan.org/misc/warnings.html#divergent-transitions-after-warmup
```

```
## Warning: There were 4540 transitions after warmup that exceeded the maximum treedepth. Increase max_treed
## http://mc-stan.org/misc/warnings.html#maximum-treedepth-exceeded
```

```
## Warning: There were 4 chains where the estimated Bayesian Fraction of Missing Information was low. See
## http://mc-stan.org/misc/warnings.html#bfmi-low
```

```
## Warning: Examine the pairs() plot to diagnose sampling problems
```

```
#
x<-rstan::extract(fit)$x
#
head(x)
```

```
##
## iterations     [,1]     [,2]     [,3]     [,4]     [,5]     [,6]     [,7]
##       [1,] 1152.009 1147.519 1141.295 1134.436 1127.712 1119.276 1110.390
##       [2,] 1144.182 1146.062 1144.424 1150.134 1144.547 1132.248 1115.227
##       [3,] 1146.444 1147.335 1145.784 1140.344 1137.233 1135.030 1139.157
##       [4,] 1171.736 1169.789 1163.912 1157.728 1150.355 1141.595 1130.764
##       [5,] 1139.827 1132.859 1126.365 1120.021 1112.683 1105.551 1099.321
##       [6,] 1136.661 1133.680 1128.328 1124.693 1123.046 1119.361 1109.387
##
## iterations     [,8]     [,9]    [,10]    [,11]    [,12]    [,13]    [,14]
##       [1,] 1099.048 1088.617 1077.811 1067.682 1057.640 1049.787 1042.205
```

```
##       [2,] 1097.104 1081.053 1073.842 1076.956 1076.626 1069.563 1061.939
##       [3,] 1138.870 1133.855 1126.848 1122.968 1112.753 1106.146 1096.185
##       [4,] 1120.071 1110.231 1101.050 1092.545 1082.200 1070.510 1058.271
##       [5,] 1091.892 1085.271 1075.945 1068.824 1062.499 1056.213 1048.833
##       [6,] 1098.761 1084.755 1071.194 1054.728 1045.512 1038.231 1033.571
##
## iterations    [,15]    [,16]    [,17]    [,18]    [,19]     [,20]
##       [1,] 1036.102 1030.660 1024.293 1016.264 1007.000  996.4985
##       [2,] 1052.496 1037.924 1048.000 1054.516 1070.397 1088.9735
##       [3,] 1094.847 1088.152 1078.340 1070.101 1063.662 1057.4297
##       [4,] 1046.203 1035.388 1025.701 1016.595 1007.149  998.2084
##       [5,] 1042.684 1036.548 1031.320 1027.363 1022.995 1019.3721
##       [6,] 1026.827 1023.448 1016.911 1011.695 1004.809  999.4892
##
## iterations     [,21]     [,22]     [,23]     [,24]     [,25]    [,26]
##       [1,]  986.1751  978.2772  969.4803  965.1713  960.4119 955.2103
##       [2,] 1087.7803 1077.1476 1055.0389 1031.6648 1012.0330 987.5638
##       [3,] 1045.6862 1036.4959 1030.6262 1024.8725 1008.4594 986.7460
##       [4,]  989.3474  982.0113  975.2709  968.8196  963.8546 960.1732
##       [5,] 1014.2227 1008.2098 1001.0230  992.6374  983.4743 974.4308
##       [6,]  993.2791  986.6183  984.8475  981.3819  974.9670 969.1929
##
## iterations    [,27]    [,28]    [,29]    [,30]    [,31]    [,32]    [,33]
##       [1,] 949.0372 942.4673 933.8116 923.4519 911.6344 898.3649 888.4076
##       [2,] 968.7905 948.9774 926.1196 916.8994 913.6362 905.0261 894.4025
##       [3,] 966.7769 948.3170 931.7694 916.6210 900.5703 883.0208 868.8676
##       [4,] 956.8751 952.3704 946.3768 940.0977 934.5540 931.2438 925.1512
##       [5,] 965.5282 956.0074 947.1809 938.8858 931.5836 924.4326 917.5662
##       [6,] 965.2128 956.8363 942.2517 926.0624 908.5385 889.8302 870.4735
##
## iterations    [,34]    [,35]    [,36]    [,37]    [,38]    [,39]    [,40]
##       [1,] 879.0463 870.4161 862.4842 853.1578 845.6834 838.4465 832.7737
##       [2,] 873.9562 864.5981 849.5046 844.2167 841.0724 836.6578 827.0430
##       [3,] 857.5954 850.1556 849.1857 848.6906 847.4411 847.3128 852.7216
##       [4,] 917.2499 910.7083 904.4054 897.7773 891.5477 884.8567 876.2959
##       [5,] 910.0639 902.4112 895.6762 890.0859 883.1205 877.5415 873.5672
##       [6,] 854.1385 837.8055 822.0020 810.3847 797.2620 783.3567 776.2964
##
## iterations    [,41]    [,42]    [,43]    [,44]    [,45]    [,46]    [,47]
##       [1,] 828.3525 823.2900 821.5197 821.3587 820.6195 820.7342 821.6432
##       [2,] 821.8091 808.8340 797.2086 783.8514 776.4750 766.7333 766.6984
##       [3,] 850.8463 848.3558 841.4994 844.1941 846.5194 850.4386 858.6041
##       [4,] 867.1368 859.2317 853.5909 850.3487 846.8160 842.2408 838.4074
##       [5,] 870.5024 869.4589 868.3150 868.2296 870.0525 870.6311 870.5974
##       [6,] 774.4006 772.2269 773.1644 775.7850 775.6220 774.6279 770.6006
##
## iterations    [,48]    [,49]    [,50]    [,51]    [,52]    [,53]    [,54]
##       [1,] 820.1875 819.5626 822.1061 828.3401 830.8642 832.0331 832.4222
##       [2,] 779.9228 791.2089 801.9189 811.1234 816.4588 821.0585 817.3508
##       [3,] 860.5204 860.3142 858.0436 857.2864 855.0057 852.4662 853.5896
##       [4,] 835.3034 832.1952 828.5927 825.9964 825.4897 824.0539 824.4330
##       [5,] 868.2739 865.1971 861.8035 856.6034 852.9855 848.8304 844.5209
##       [6,] 766.6076 758.3770 752.5075 750.8417 748.5316 753.6642 763.2790
##
```

```
## iterations    [,55]    [,56]    [,57]    [,58]    [,59]    [,60]    [,61]
##       [1,] 831.0175 830.7084 830.7866 831.8328 833.3667 832.2466 833.3417
##       [2,] 809.8553 805.0883 801.3042 786.1905 782.0477 783.6606 783.7063
##       [3,] 853.1007 854.0615 854.8006 861.0101 871.0286 875.8635 876.3697
##       [4,] 825.0533 826.9737 826.3588 825.0286 823.9821 822.5125 822.1640
##       [5,] 838.2399 831.6243 825.9335 822.3594 819.3513 816.6037 814.5176
##       [6,] 779.9486 795.4782 804.9169 817.5063 827.0897 836.7109 844.4322
##
## iterations    [,62]    [,63]    [,64]    [,65]    [,66]    [,67]    [,68]
##       [1,] 834.0520 833.9583 833.6039 832.8260 830.8161 832.6766 835.6843
##       [2,] 792.2579 806.1124 818.9087 834.6020 839.8902 829.7634 820.9609
##       [3,] 881.6261 883.5650 882.6444 877.5868 874.4436 868.9162 859.1339
##       [4,] 820.3366 819.2020 820.1812 822.7939 824.9520 825.7363 827.3478
##       [5,] 813.8902 812.0783 810.9055 810.8965 811.5382 813.3606 814.9146
##       [6,] 849.9394 855.0411 859.4423 857.8932 852.5200 848.7433 848.1349
##
## iterations    [,69]    [,70]    [,71]    [,72]    [,73]    [,74]    [,75]
##       [1,] 838.3219 843.5423 847.8292 851.2903 854.4196 858.1689 861.5885
##       [2,] 820.9521 822.9578 821.9170 825.3417 824.5981 823.1676 821.3395
##       [3,] 855.6774 850.0951 848.7772 842.4661 839.2479 834.3134 826.5286
##       [4,] 828.3397 827.0542 825.5481 824.4026 823.3909 823.3804 824.0503
##       [5,] 817.6506 820.4157 822.1726 823.9772 824.7880 824.9426 826.3343
##       [6,] 846.1857 847.2081 852.9201 861.9749 865.1364 870.7603 877.4166
##
## iterations    [,76]    [,77]    [,78]    [,79]    [,80]    [,81]    [,82]
##       [1,] 868.2048 874.4065 878.7210 882.0279 884.8340 885.9484 885.3099
##       [2,] 828.3893 831.7404 834.8331 832.6847 829.4954 834.2500 843.0062
##       [3,] 822.7128 815.5030 817.1340 818.2593 823.0504 831.6954 842.3468
##       [4,] 823.5839 824.7475 824.4153 826.7093 828.7931 832.3198 835.6159
##       [5,] 828.9573 830.9997 832.4720 835.5358 839.6234 841.8097 844.4169
##       [6,] 884.0977 891.2992 899.2075 907.8697 914.0098 918.6667 922.5119
##
## iterations    [,83]    [,84]    [,85]    [,86]    [,87]    [,88]    [,89]
##       [1,] 885.6867 885.8545 889.5031 891.0405 890.2326 890.3096 890.1949
##       [2,] 850.0938 859.9359 871.2204 879.8011 885.0822 891.3550 888.2657
##       [3,] 848.9230 852.7456 854.3815 862.4846 869.9231 878.0207 879.5654
##       [4,] 838.2974 840.5511 841.6844 841.2028 839.8494 838.4895 838.5425
##       [5,] 846.9950 848.9564 852.8648 857.5982 862.5593 867.0841 871.8841
##       [6,] 922.0033 924.8654 928.4683 933.4418 936.4889 937.2167 931.7041
##
## iterations    [,90]    [,91]    [,92]    [,93]    [,94]    [,95]    [,96]
##       [1,] 888.6970 887.5943 885.5468 886.2886 885.5473 882.0371 880.2857
##       [2,] 879.9202 868.9002 856.8224 842.2075 833.1097 831.1394 824.3551
##       [3,] 880.9796 881.8778 879.5565 880.0946 875.6882 869.3004 861.2420
##       [4,] 839.1036 840.1967 842.0077 842.0584 844.3354 847.6487 850.2647
##       [5,] 876.9697 881.4895 887.2026 891.5076 894.5877 896.8570 898.2728
##       [6,] 923.2181 915.7181 910.1716 904.0021 895.7694 883.9829 869.5275
##
## iterations    [,97]    [,98]    [,99]   [,100]
##       [1,] 874.9919 870.7804 867.1796 861.9925
##       [2,] 816.4765 813.5355 809.4566 807.7666
##       [3,] 857.4048 855.7485 853.1973 849.4280
##       [4,] 852.8889 855.0477 858.9802 860.8798
##       [5,] 900.2117 902.3349 904.6376 907.3741
```

```
##       [6,] 855.8268 842.0671 828.8500 818.0378
```

```
d2<-data.frame(x)
up<-c()
lo<-c()
M<-c()

for(i in 1:length(d2)){
  up[i]<-quantile(d2[,i],0.975)
  lo[i]<-quantile(d2[,i],0.025)
  M[i]<-mean(d2[,i])
}

df<-data.frame(y=y$y, t=1:y$T, x=M, upper=up, lower=lo)
g<-ggplot(data = df, aes(x = t, y = y))
g<-g+geom_point()
g<-g+geom_errorbar(data = df, aes(ymin=lower, ymax=upper, x=t))
g<-g+geom_line(aes(x=t,y=x),data=df)
g
```