stan

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

```
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。

```r
library(rstan)
```

```
## Loading required package: ggplot2

## Loading required package: StanHeaders

## rstan (Version 2.16.2, packaged: 2017-07-03 09:24:58 UTC, GitRev: 2e1f913d3ca3)

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

```r
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: 0.011284 seconds (Warm-up)
##                0.009944 seconds (Sampling)
##                0.021228 seconds (Total)
##
##
## SAMPLING FOR MODEL 'd06ec3f2e0cf24bb2a3cdc911e8284d0' NOW (CHAIN 2).
##
## 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: 0.011176 seconds (Warm-up)
##                0.008809 seconds (Sampling)
##                0.019985 seconds (Total)
##
##
## SAMPLING FOR MODEL 'd06ec3f2e0cf24bb2a3cdc911e8284d0' NOW (CHAIN 3).
##
## 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: 0.009485 seconds (Warm-up)
##                0.011934 seconds (Sampling)
##                0.021419 seconds (Total)
##
##
## SAMPLING FOR MODEL 'd06ec3f2e0cf24bb2a3cdc911e8284d0' 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: 0.009423 seconds (Warm-up)
##                0.00941 seconds (Sampling)
##                0.018833 seconds (Total)
```

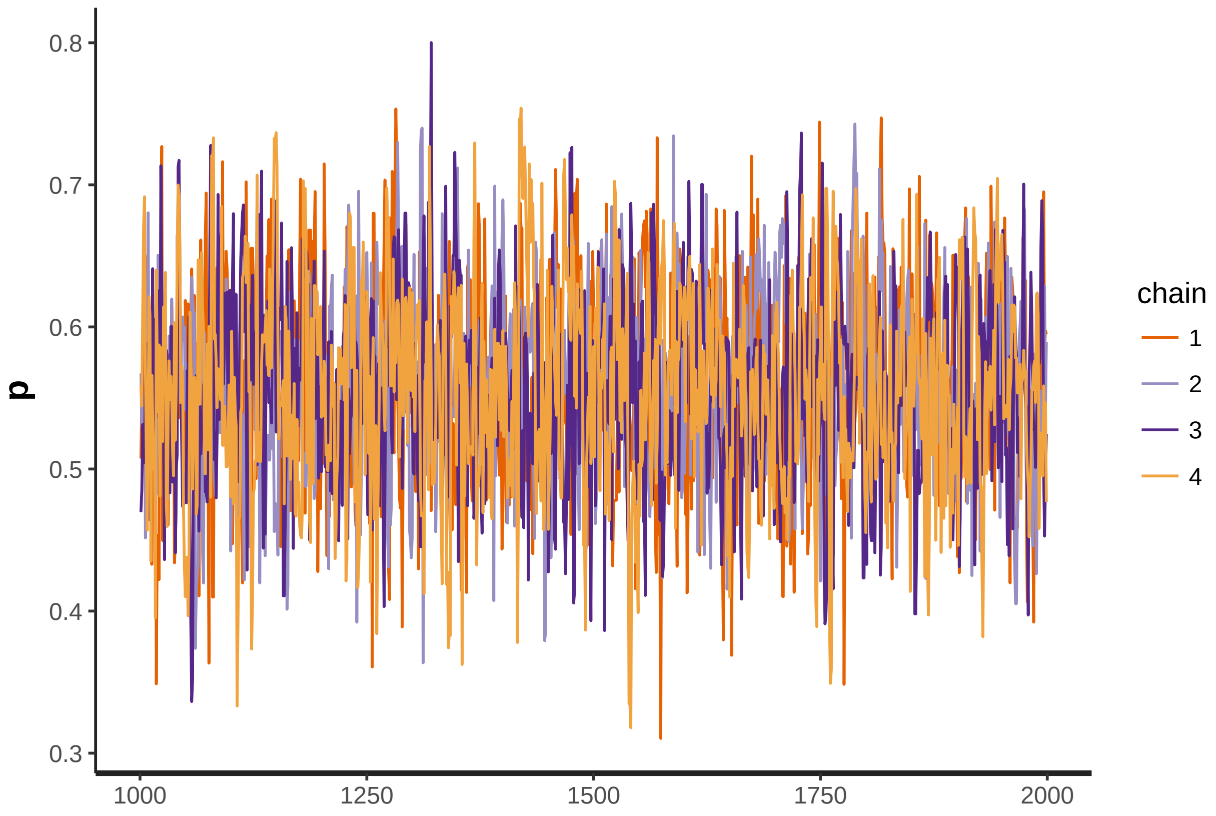
```
fit1
```

```
## 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   0.42   0.51   0.56   0.60   0.69  1566    1
## lp__ -36.19    0.02 0.72 -38.25 -36.34 -35.91 -35.74 -35.70  1663    1
##
## Samples were drawn using NUTS(diag_e) at Fri Nov 10 16:23:48 2017.
## 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

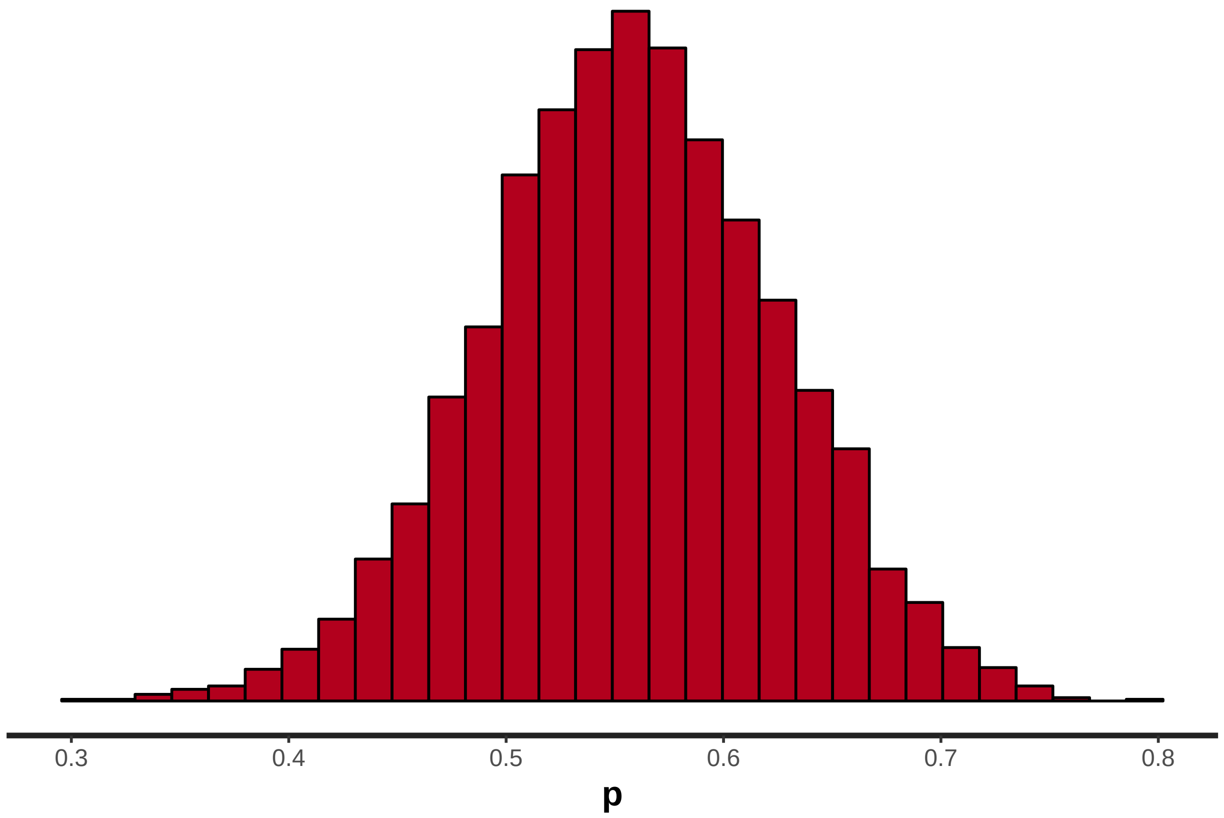
```
stan_trace(fit1,pars="p")
```



パラメータのサンプルのヒストグラム

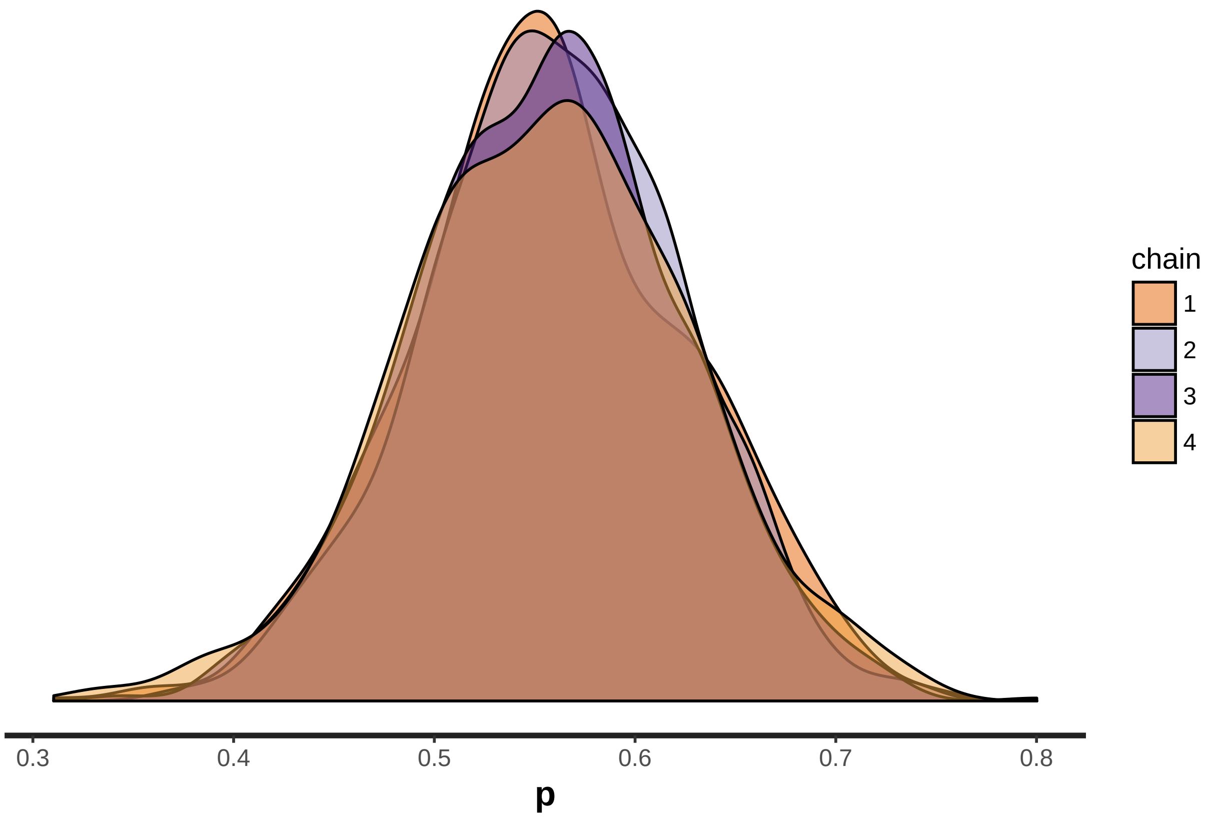
```
stan_hist(fit1,pars="p")
```

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



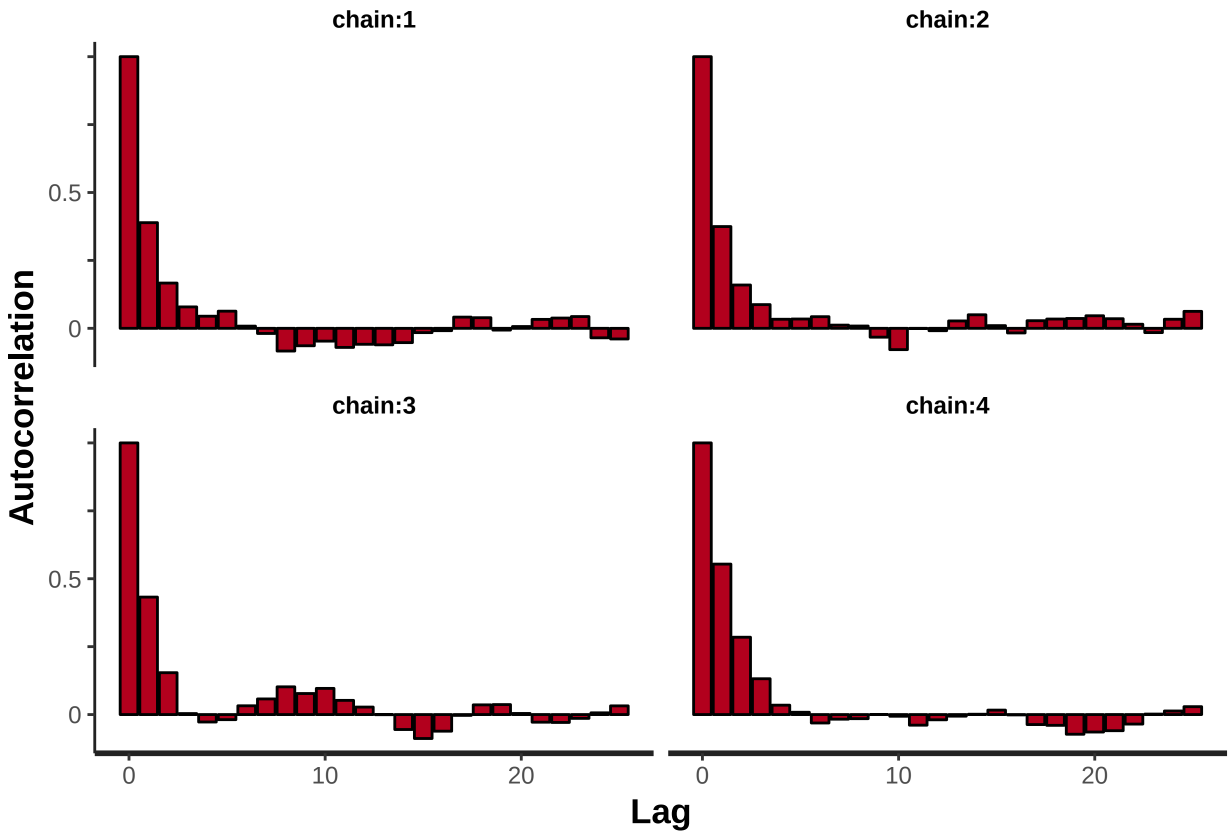
パラメータ事後分布の密度推定

```
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）
- チェインごとのカーネル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] 0.5576991
```

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。

```r
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 1.6e-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: 0.010753 seconds (Warm-up)
##                0.010652 seconds (Sampling)
##                0.021405 seconds (Total)
##
##
## SAMPLING FOR MODEL 'd9d5669abb02a1e798ca9ce33cd34eb4' NOW (CHAIN 2).
##
## 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: 0.011154 seconds (Warm-up)
##                0.011802 seconds (Sampling)
##                0.022956 seconds (Total)
##
##
## SAMPLING FOR MODEL 'd9d5669abb02a1e798ca9ce33cd34eb4' 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: 0.011765 seconds (Warm-up)
##                0.010722 seconds (Sampling)
##                0.022487 seconds (Total)
##
##
## SAMPLING FOR MODEL 'd9d5669abb02a1e798ca9ce33cd34eb4' NOW (CHAIN 4).
##
## Gradient evaluation took 3e-06 seconds
## 1000 transitions using 10 leapfrog steps per transition would take 0.03 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)
```

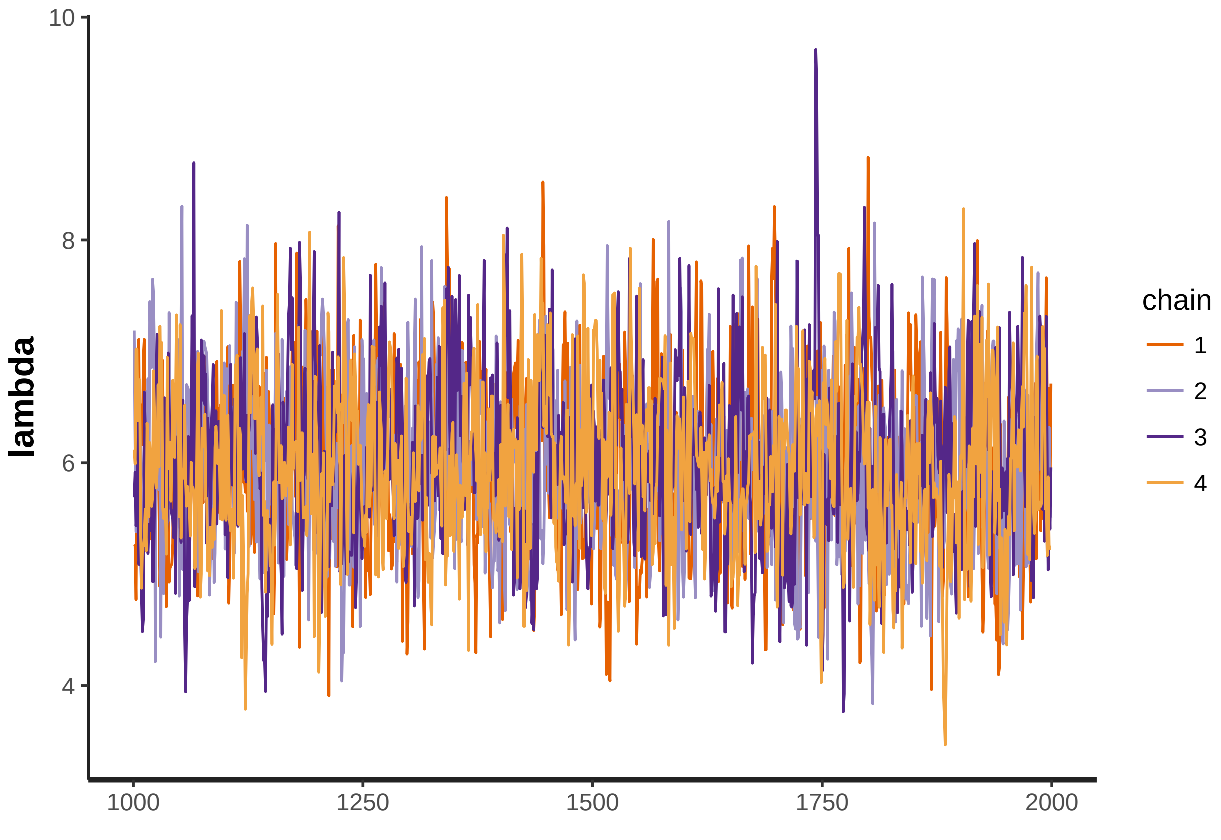
```
##
##  Elapsed Time: 0.011011 seconds (Warm-up)
##                0.010259 seconds (Sampling)
##                0.02127 seconds (Total)
```

```
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  5.98    0.02 0.75  4.57  5.48  5.94  6.48  7.52  1453    1
## lp__   47.03    0.02 0.69 45.10 46.88 47.29 47.46 47.51  1816    1
##
## Samples were drawn using NUTS(diag_e) at Fri Nov 10 16:24:25 2017.
## 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

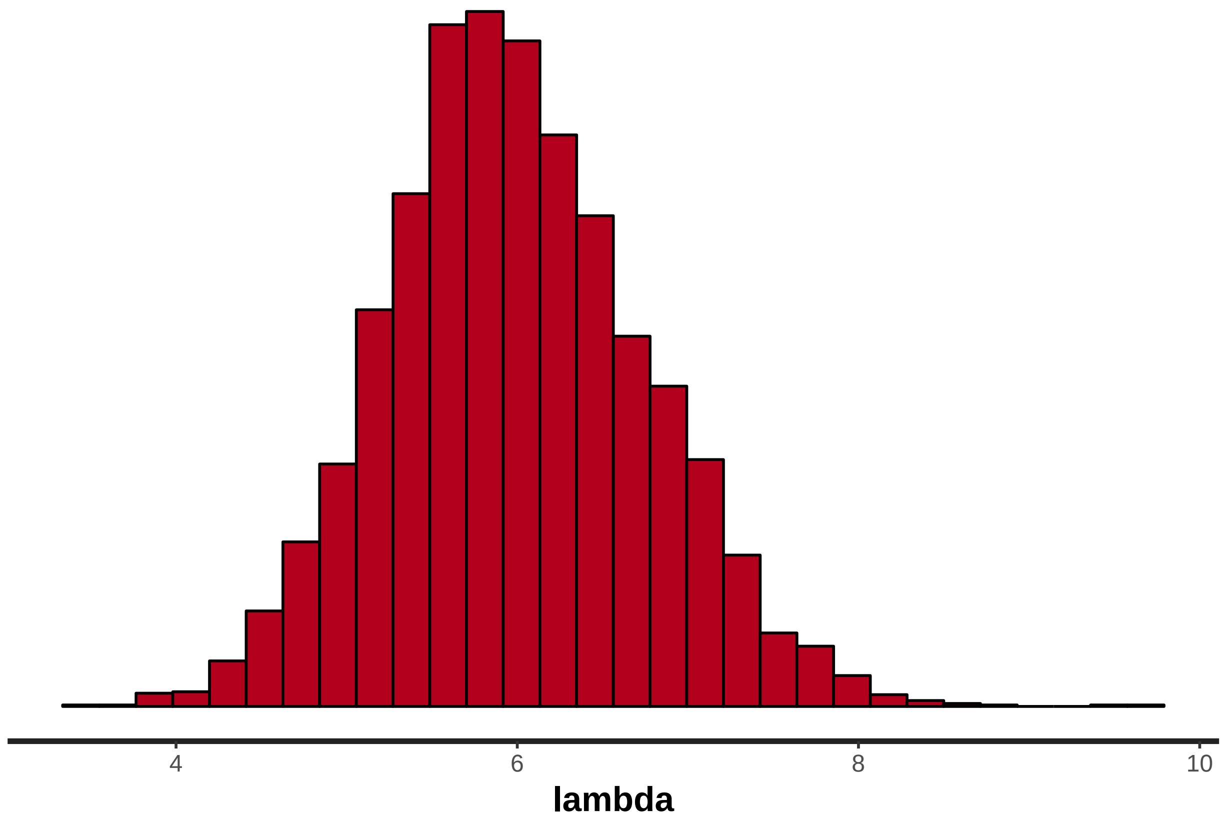
```
stan_trace(fit2,pars="lambda")
```



パラメータのサンプルのヒストグラム

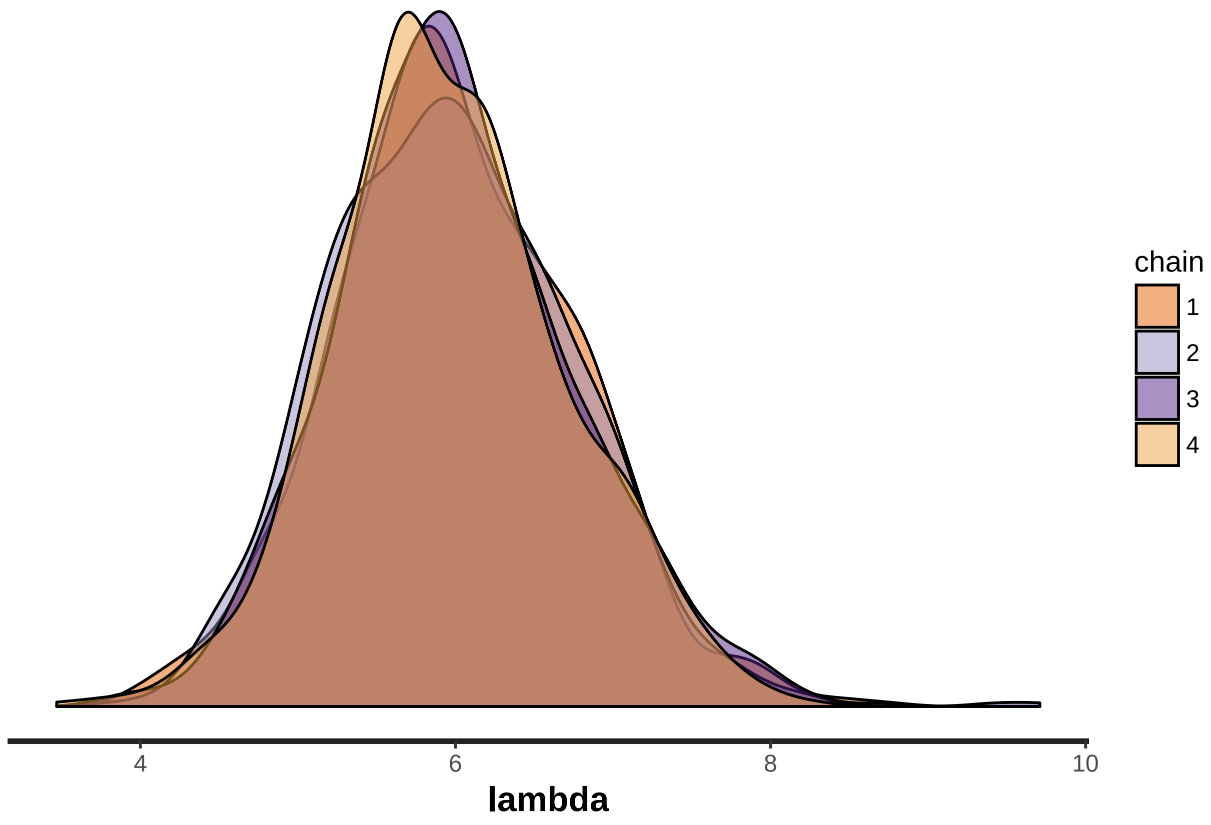
```
stan_hist(fit2,pars="lambda")
```

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



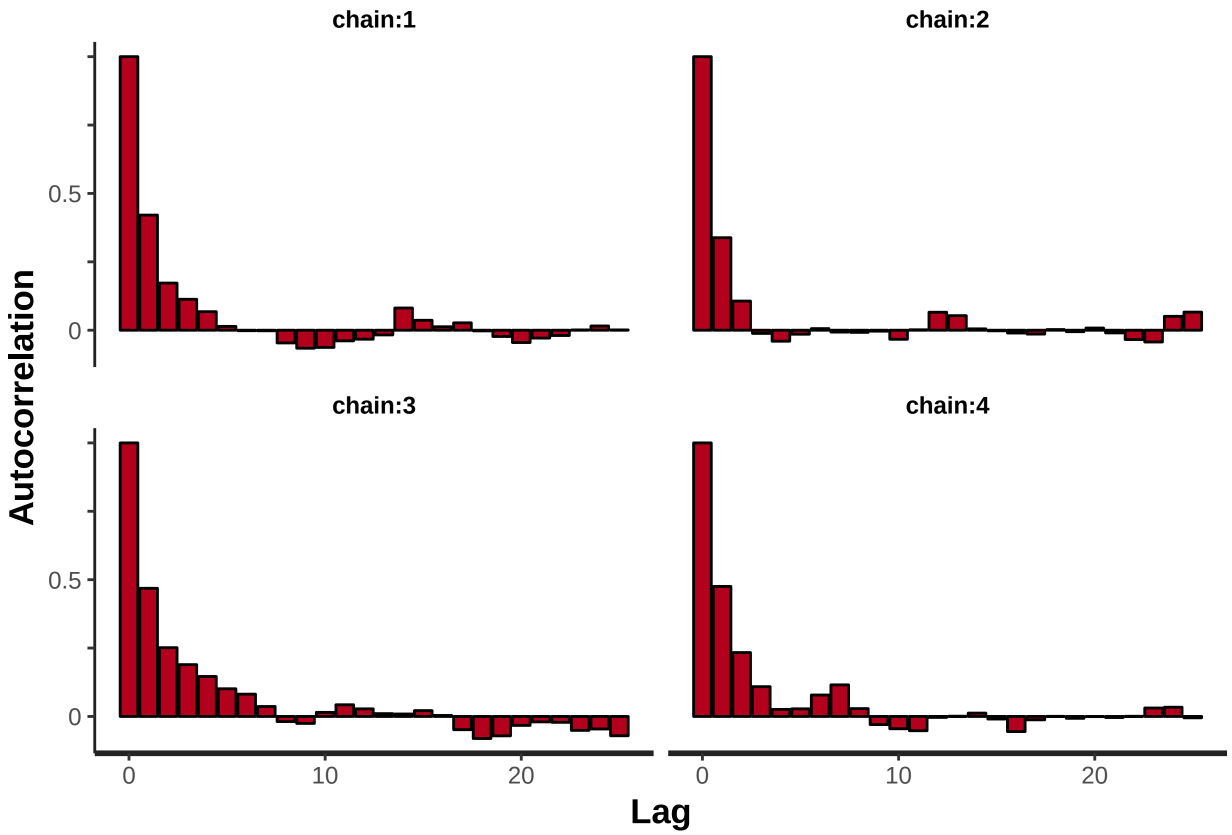
パラメータ事後分布の密度推定

```r
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] 5.984548
```

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)
```

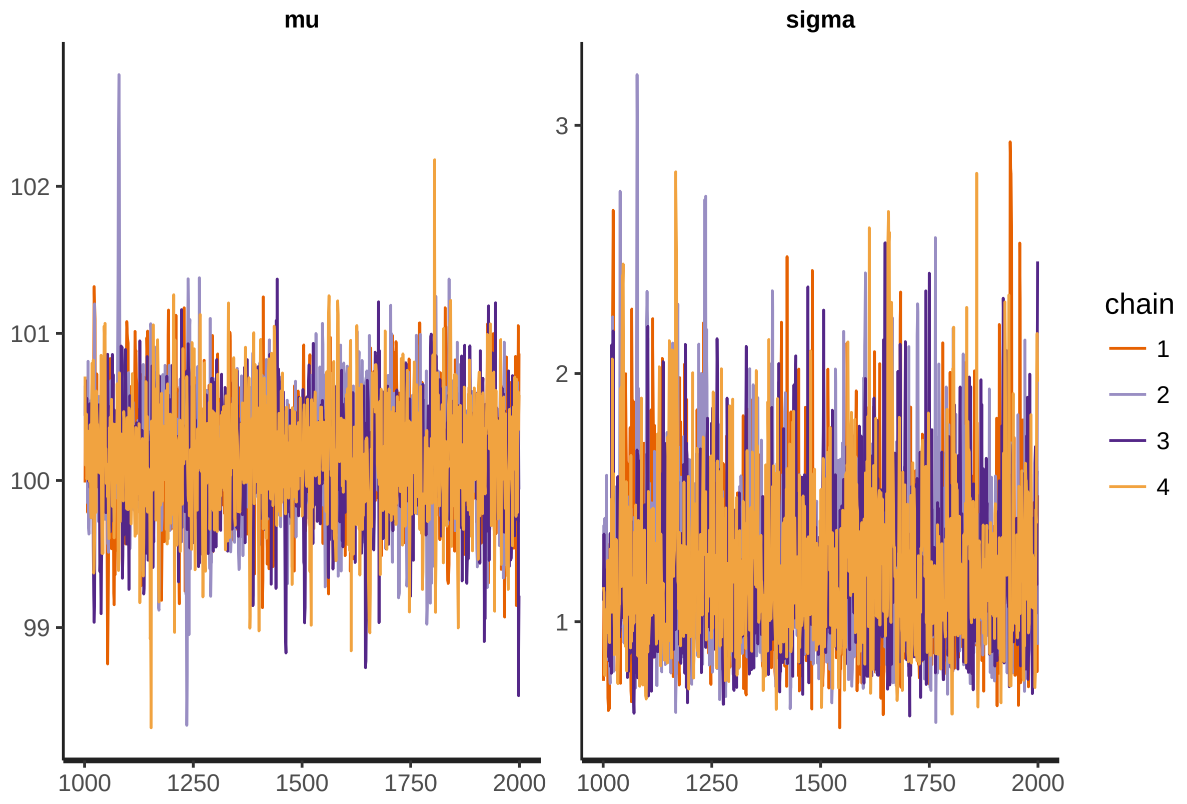
```
##
## SAMPLING FOR MODEL 'c81f4001bf9438379267dee6a2a4f8a2' NOW (CHAIN 1).
##
## Gradient evaluation took 1.1e-05 seconds
## 1000 transitions using 10 leapfrog steps per transition would take 0.11 seconds.
## Adjust your expectations accordingly!
##
##
## Iteration:    1 / 2000 [  0%]  (Warmup)
## 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: 0.021183 seconds (Warm-up)
##                0.023247 seconds (Sampling)
##                0.04443 seconds (Total)
##
##
## SAMPLING FOR MODEL 'c81f4001bf9438379267dee6a2a4f8a2' NOW (CHAIN 2).
##
## 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: 0.020673 seconds (Warm-up)
##                0.016795 seconds (Sampling)
##                0.037468 seconds (Total)
##
##
## SAMPLING FOR MODEL 'c81f4001bf9438379267dee6a2a4f8a2' NOW (CHAIN 3).
##
## 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: 0.020498 seconds (Warm-up)
##                0.032523 seconds (Sampling)
##                0.053021 seconds (Total)
##
##
## SAMPLING FOR MODEL 'c81f4001bf9438379267dee6a2a4f8a2' 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: 0.017063 seconds (Warm-up)
##                0.018565 seconds (Sampling)
##                0.035628 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    100.16    0.01 0.40 99.34 99.91 100.17 100.41 100.94  2183    1
## sigma   1.22    0.01 0.33  0.75  0.98   1.15   1.38   2.04  1790    1
## lp__   -6.65    0.03 1.07 -9.45 -7.07  -6.33  -5.89  -5.59  1399    1
##
## Samples were drawn using NUTS(diag_e) at Fri Nov 10 16:24:59 2017.
## 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

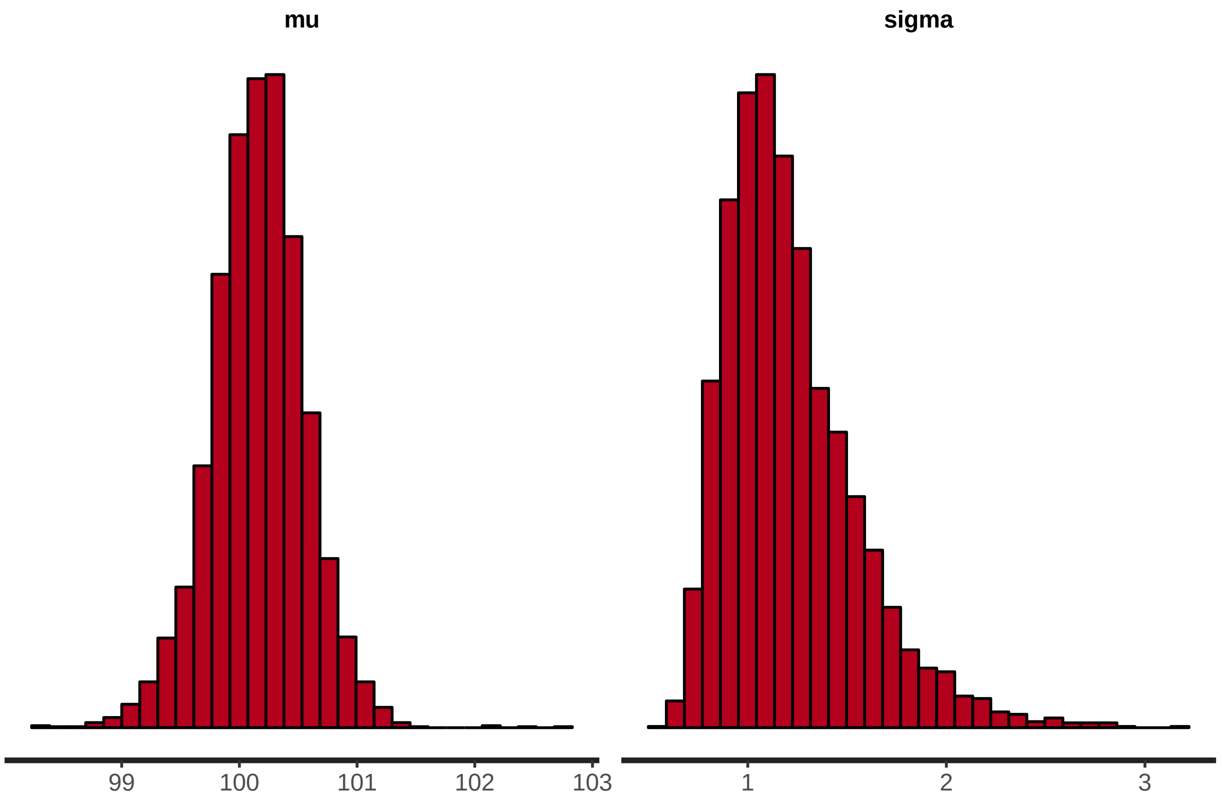
```r
stan_trace(fit3, pars=c("mu","sigma"))
```



パラメータのサンプルのヒストグラム

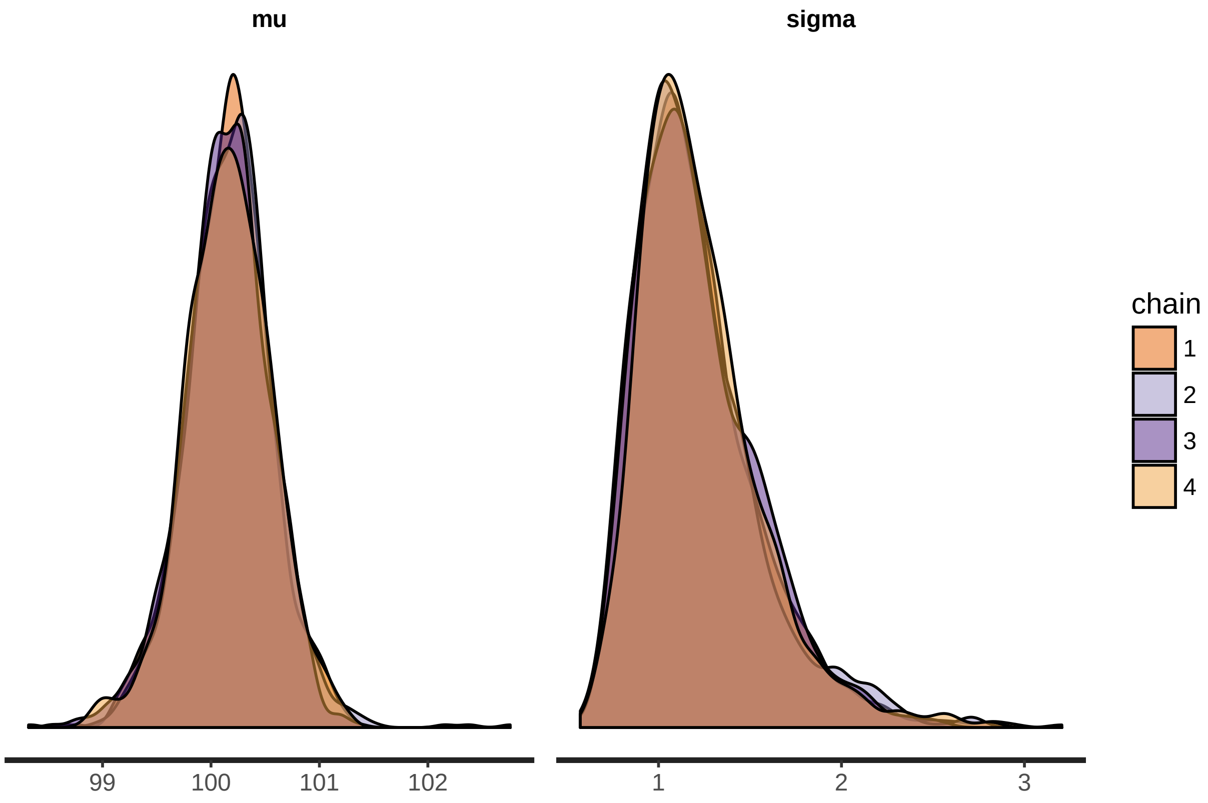
```r
stan_hist(fit3,pars=c("mu","sigma"))
```

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



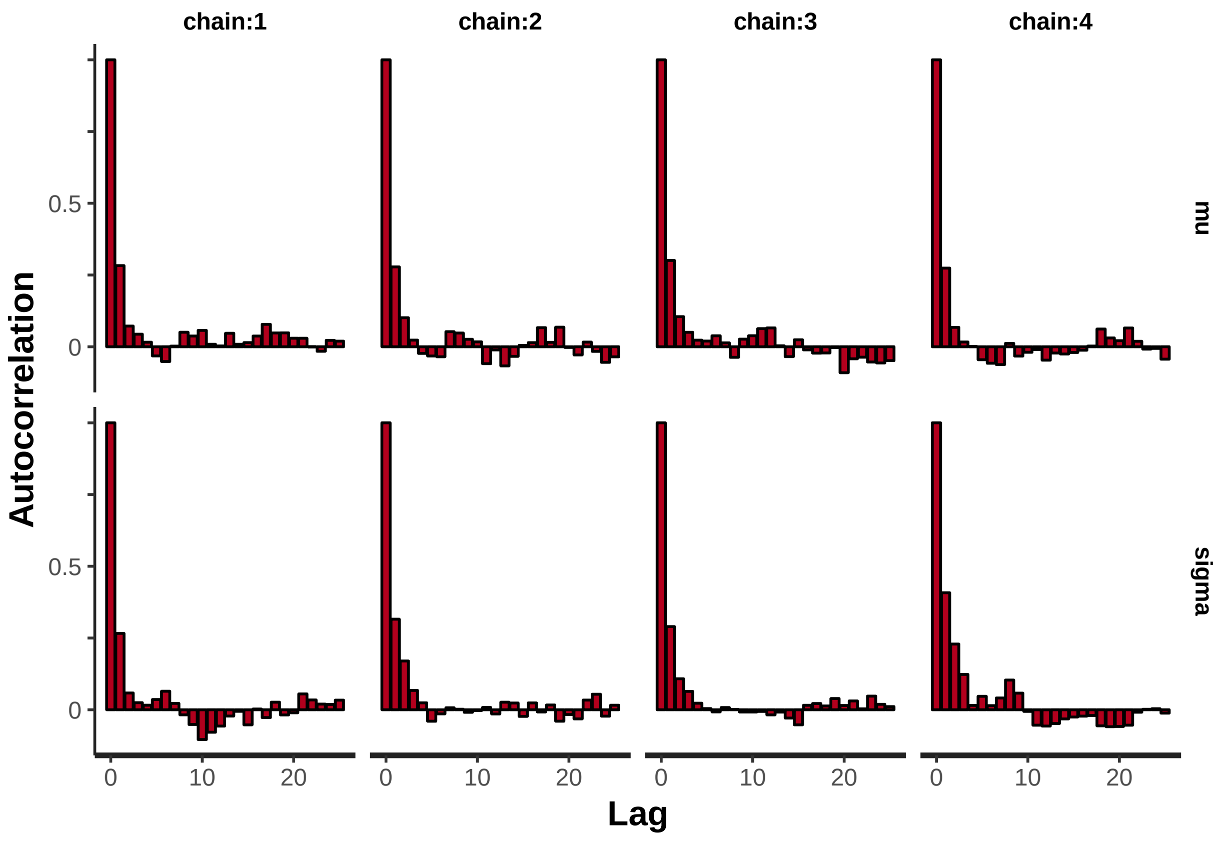
パラメータ事後分布の密度推定

```
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 1.1e-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: 0.036231 seconds (Warm-up)
##                0.019663 seconds (Sampling)
##                0.055894 seconds (Total)
##
##
## SAMPLING FOR MODEL 'b6f16321d07933d8582b0bc79e69a9e1' NOW (CHAIN 2).
##
## 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: 0.038376 seconds (Warm-up)
##                0.022136 seconds (Sampling)
##                0.060512 seconds (Total)
##
##
## SAMPLING FOR MODEL 'b6f16321d07933d8582b0bc79e69a9e1' NOW (CHAIN 3).
##
## Gradient evaluation took 3e-06 seconds
## 1000 transitions using 10 leapfrog steps per transition would take 0.03 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: 0.033273 seconds (Warm-up)
##                0.017433 seconds (Sampling)
##                0.050706 seconds (Total)
##
##
## SAMPLING FOR MODEL 'b6f16321d07933d8582b0bc79e69a9e1' NOW (CHAIN 4).
##
## Gradient evaluation took 3e-06 seconds
## 1000 transitions using 10 leapfrog steps per transition would take 0.03 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: 0.05074 seconds (Warm-up)
##                0.027377 seconds (Sampling)
##                0.078117 seconds (Total)
```

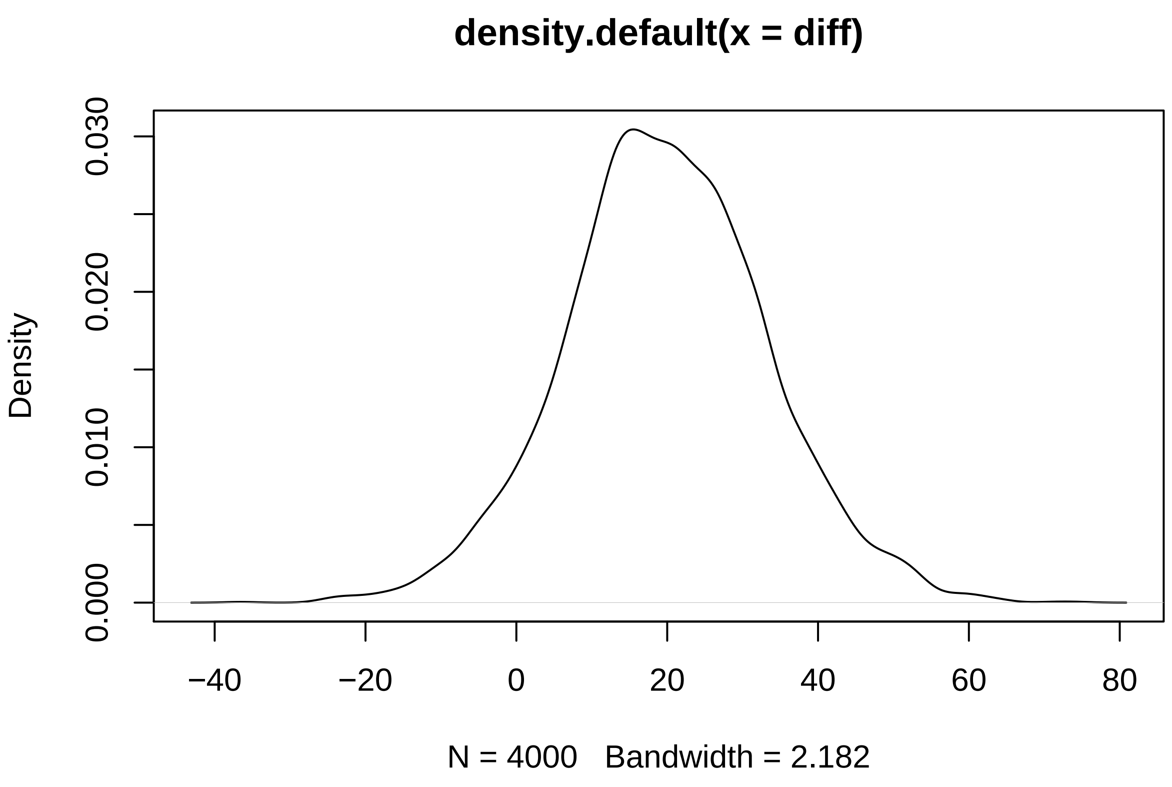
```
fit
```

```
## 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     39.37    0.13  7.68  24.56  34.57  39.27  44.31  54.47  3352    1
## sigma_y  23.65    0.11  5.92  15.20  19.57  22.59  26.62  37.92  2817    1
## mu_x     59.26    0.20 11.03  37.50  52.28  59.28  66.17  81.51  3128    1
## sigma_x  33.93    0.17  8.83  21.87  28.04  32.33  38.05  55.08  2629    1
## diff     19.89    0.23 13.43  -6.34  11.44  19.68  28.51  47.65  3324    1
## lp__    -77.75    0.04  1.59 -81.73 -78.52 -77.38 -76.61 -75.80  1549    1
## 
## Samples were drawn using NUTS(diag_e) at Fri Nov 10 16:25:35 2017.
## 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] 0.93325
```

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



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
```

```r
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 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: 0.050608 seconds (Warm-up)
##                0.042601 seconds (Sampling)
##                0.093209 seconds (Total)
##
##
## SAMPLING FOR MODEL '588538e1cebbf0d9aad8232558d90dcc' NOW (CHAIN 2).
##
## 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: 0.04409 seconds (Warm-up)
##                0.045203 seconds (Sampling)
##                0.089293 seconds (Total)
##
##
```

```
## SAMPLING FOR MODEL '588538e1cebbf0d9aad8232558d90dcc' NOW (CHAIN 3).
##
## 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: 0.057161 seconds (Warm-up)
##                0.047978 seconds (Sampling)
##                0.105139 seconds (Total)
##
##
## SAMPLING FOR MODEL '588538e1cebbf0d9aad8232558d90dcc' NOW (CHAIN 4).
##
## 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: 0.048096 seconds (Warm-up)
##                0.042942 seconds (Sampling)
##                0.091038 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       1.07    0.00 0.07  0.92  1.02  1.07  1.11  1.20  1417    1
## b       1.96    0.00 0.04  1.87  1.93  1.96  1.99  2.05  1417    1
## sigma   0.11    0.00 0.02  0.09  0.10  0.11  0.12  0.15  1995    1
## lp__   48.66    0.04 1.36 45.13 48.07 49.03 49.65 50.18  1185    1
##
## Samples were drawn using NUTS(diag_e) at Fri Nov 10 16:26:08 2017.
## 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).
```

回帰直線を引くと、以下のようになる。

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、以下で二つの事例を見ながら、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。

```r
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 file22915893f649.cpp:8:
## In file included from /Library/Frameworks/R.framework/Versions/3.4/Resources/library/StanHeaders/incl
## In file included from /Library/Frameworks/R.framework/Versions/3.4/Resources/library/StanHeaders/incl
## In file included from /Library/Frameworks/R.framework/Versions/3.4/Resources/library/StanHeaders/incl
## In file included from /Library/Frameworks/R.framework/Versions/3.4/Resources/library/StanHeaders/incl
## In file included from /Library/Frameworks/R.framework/Versions/3.4/Resources/library/StanHeaders/incl
## In file included from /Library/Frameworks/R.framework/Versions/3.4/Resources/library/StanHeaders/incl
## In file included from /Library/Frameworks/R.framework/Versions/3.4/Resources/library/BH/include/boos
## In file included from /Library/Frameworks/R.framework/Versions/3.4/Resources/library/BH/include/boos
## /Library/Frameworks/R.framework/Versions/3.4/Resources/library/BH/include/boost/config/compiler/clan
## #  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 file22915893f649.cpp:8:
## In file included from /Library/Frameworks/R.framework/Versions/3.4/Resources/library/StanHeaders/incl
## In file included from /Library/Frameworks/R.framework/Versions/3.4/Resources/library/StanHeaders/incl
## In file included from /Library/Frameworks/R.framework/Versions/3.4/Resources/library/StanHeaders/incl
## In file included from /Library/Frameworks/R.framework/Versions/3.4/Resources/library/StanHeaders/incl
## /Library/Frameworks/R.framework/Versions/3.4/Resources/library/StanHeaders/include/stan/math/rev/core
##     static void set_zero_all_adjoints() {
##                 ^
## In file included from file22915893f649.cpp:8:
## In file included from /Library/Frameworks/R.framework/Versions/3.4/Resources/library/StanHeaders/incl
## In file included from /Library/Frameworks/R.framework/Versions/3.4/Resources/library/StanHeaders/incl
## In file included from /Library/Frameworks/R.framework/Versions/3.4/Resources/library/StanHeaders/incl
## In file included from /Library/Frameworks/R.framework/Versions/3.4/Resources/library/StanHeaders/incl
## /Library/Frameworks/R.framework/Versions/3.4/Resources/library/StanHeaders/include/stan/math/rev/core
##     static void set_zero_all_adjoints_nested() {
##                 ^
## In file included from file22915893f649.cpp:8:
## In file included from /Library/Frameworks/R.framework/Versions/3.4/Resources/library/StanHeaders/incl
## In file included from /Library/Frameworks/R.framework/Versions/3.4/Resources/library/StanHeaders/incl
## In file included from /Library/Frameworks/R.framework/Versions/3.4/Resources/library/StanHeaders/incl
## In file included from /Library/Frameworks/R.framework/Versions/3.4/Resources/library/StanHeaders/incl
## /Library/Frameworks/R.framework/Versions/3.4/Resources/library/StanHeaders/include/stan/math/prim/mat
##       size_t fft_next_good_size(size_t N) {
##              ^
## In file included from file22915893f649.cpp:8:
## In file included from /Library/Frameworks/R.framework/Versions/3.4/Resources/library/StanHeaders/incl
## In file included from /Library/Frameworks/R.framework/Versions/3.4/Resources/library/StanHeaders/incl
## In file included from /Library/Frameworks/R.framework/Versions/3.4/Resources/library/StanHeaders/incl
## In file included from /Library/Frameworks/R.framework/Versions/3.4/Resources/library/StanHeaders/incl
## In file included from /Library/Frameworks/R.framework/Versions/3.4/Resources/library/StanHeaders/incl
## In file included from /Library/Frameworks/R.framework/Versions/3.4/Resources/library/StanHeaders/incl
## In file included from /Library/Frameworks/R.framework/Versions/3.4/Resources/library/BH/include/boost
## In file included from /Library/Frameworks/R.framework/Versions/3.4/Resources/library/BH/include/boost
## In file included from /Library/Frameworks/R.framework/Versions/3.4/Resources/library/BH/include/boost
## In file included from /Library/Frameworks/R.framework/Versions/3.4/Resources/library/BH/include/boost
## /Library/Frameworks/R.framework/Versions/3.4/Resources/library/BH/include/boost/multi_array/concept_c
##       typedef typename Array::index_range index_range;
##                                           ^
## /Library/Frameworks/R.framework/Versions/3.4/Resources/library/BH/include/boost/multi_array/concept_c
##       typedef typename Array::index index;
##                                     ^
## /Library/Frameworks/R.framework/Versions/3.4/Resources/library/BH/include/boost/multi_array/concept_c
##       typedef typename Array::index_range index_range;
##                                           ^
## /Library/Frameworks/R.framework/Versions/3.4/Resources/library/BH/include/boost/multi_array/concept_c
##       typedef typename Array::index index;
##                                     ^
## 8 warnings generated.
##
## SAMPLING FOR MODEL 'logreg' NOW (CHAIN 1).
##
## Gradient evaluation took 1.9e-05 seconds
## 1000 transitions using 10 leapfrog steps per transition would take 0.19 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: 0.111631 seconds (Warm-up)
##                0.103818 seconds (Sampling)
##                0.215449 seconds (Total)
##
##
## SAMPLING FOR MODEL 'logreg' 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: 0.104668 seconds (Warm-up)
##                0.111924 seconds (Sampling)
##                0.216592 seconds (Total)
##
##
## SAMPLING FOR MODEL 'logreg' NOW (CHAIN 3).
##
## 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: 0.114115 seconds (Warm-up)
##                0.113783 seconds (Sampling)
##                0.227898 seconds (Total)
```

```
##
##
## SAMPLING FOR MODEL 'logreg' NOW (CHAIN 4).
##
## 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: 0.12606 seconds (Warm-up)
##               0.124761 seconds (Sampling)
##               0.250821 seconds (Total)
```

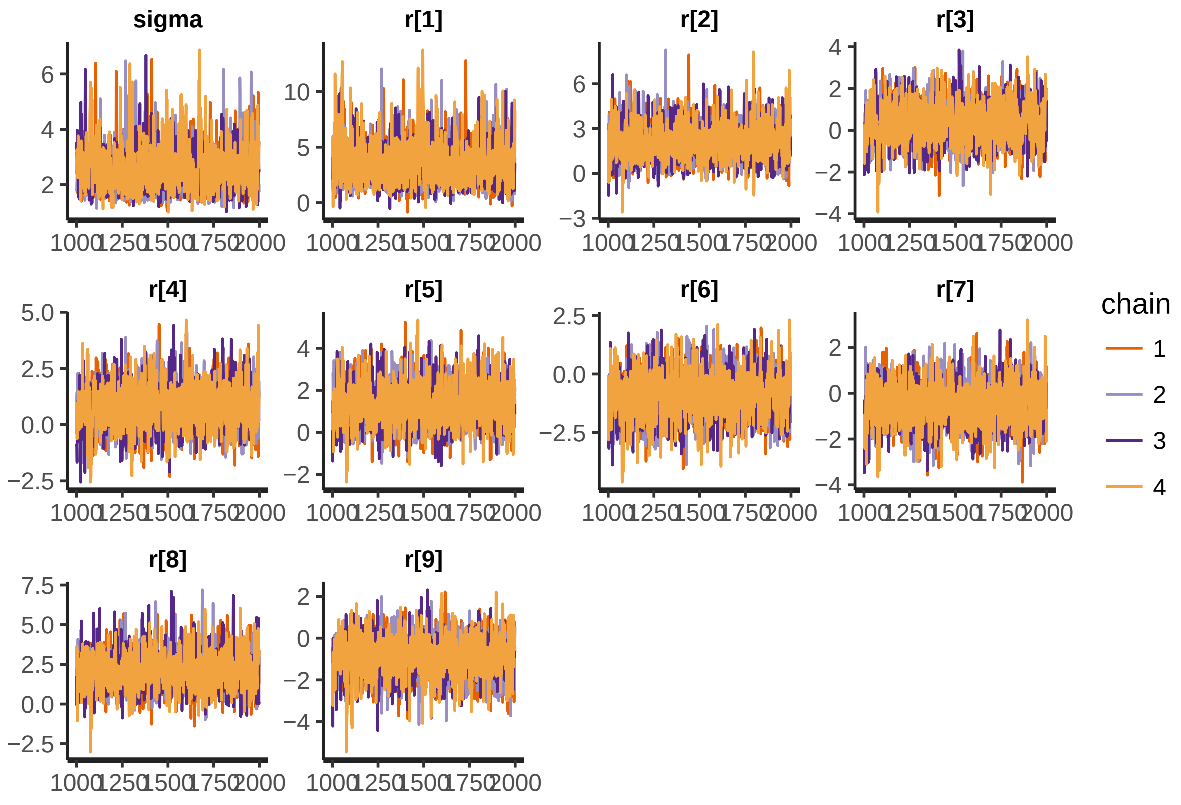
```
fit
```

```
## 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    2.52    0.02 0.74    1.45    2.01    2.39    2.87    4.38   988
## r[1]     3.50    0.04 1.75    0.88    2.30    3.22    4.40    7.70  1669
## r[2]     2.10    0.02 1.14    0.08    1.34    2.01    2.76    4.59  2092
## r[3]     0.38    0.02 0.89   -1.38   -0.19    0.36    0.95    2.17  1389
## r[4]     0.84    0.02 0.92   -0.96    0.24    0.82    1.43    2.70  1556
## r[5]     1.37    0.02 0.97   -0.42    0.71    1.32    1.98    3.41  1666
## r[6]    -0.88    0.02 0.92   -2.73   -1.47   -0.85   -0.29    0.91  1472
## r[7]    -0.43    0.02 0.90   -2.22   -1.02   -0.41    0.19    1.29  1441
## r[8]     2.09    0.02 1.11    0.13    1.36    1.99    2.76    4.49  1999
## r[9]    -0.87    0.02 0.92   -2.82   -1.44   -0.87   -0.26    0.88  1380
## r[10]    0.84    0.02 0.91   -0.90    0.25    0.82    1.43    2.71  1539
## r[11]    3.46    0.04 1.64    0.94    2.30    3.25    4.39    7.29  1823
## r[12]   -1.41    0.02 0.99   -3.47   -2.02   -1.36   -0.75    0.47  1593
## r[13]   -3.54    0.04 1.72   -7.67   -4.36   -3.28   -2.37   -0.96  1480
## r[14]   -0.02    0.02 0.87   -1.69   -0.57   -0.02    0.54    1.65  1421
## r[15]    0.39    0.02 0.91   -1.39   -0.20    0.38    0.97    2.27  1513
## r[16]   -0.87    0.02 0.92   -2.71   -1.46   -0.85   -0.25    0.87  1576
## r[17]   -2.14    0.03 1.15   -4.69   -2.80   -2.04   -1.36   -0.15  1565
## r[18]    2.09    0.02 1.13    0.06    1.34    2.01    2.76    4.59  2099
## r[19]   -3.53    0.05 1.80   -7.74   -4.37   -3.24   -2.33   -0.96  1313
## r[20]   -3.51    0.04 1.68   -7.51   -4.39   -3.28   -2.38   -0.95  1615
## beta     0.02    0.02 0.66   -1.27   -0.38    0.02    0.44    1.34   867
## 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    0.62    0.81    0.88    0.94    0.99  4000
## p[3]      0.59    0.00 0.14    0.30    0.49    0.60    0.70    0.84  4000
## p[4]      0.68    0.00 0.14    0.40    0.59    0.70    0.78    0.91  4000
## p[5]      0.78    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    0.21    0.30    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    0.93    0.99  4000
## p[9]      0.32    0.00 0.13    0.09    0.22    0.31    0.40    0.61  4000
## p[10]     0.69    0.00 0.13    0.40    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    0.05    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 0.14    0.23    0.40    0.50    0.60    0.77  4000
## p[15]     0.59    0.00 0.15    0.29    0.49    0.60    0.70    0.86  4000
## p[16]     0.32    0.00 0.14    0.09    0.22    0.31    0.41    0.60  4000
## p[17]     0.14    0.00 0.10    0.01    0.06    0.12    0.19    0.38  4000
## p[18]     0.86    0.00 0.10    0.61    0.81    0.89    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 0.07    0.00    0.01    0.04    0.09    0.25  4000
## lp__  -117.21    0.13 4.22 -126.59 -119.76 -116.76 -114.20 -110.34   990
##       Rhat
## sigma 1.01
## r[1]  1.00
## r[2]  1.00
## r[3]  1.00
## r[4]  1.00
## r[5]  1.00
## r[6]  1.00
## r[7]  1.00
## r[8]  1.00
## r[9]  1.00
## r[10] 1.00
## r[11] 1.00
## r[12] 1.00
## r[13] 1.00
## r[14] 1.00
## r[15] 1.00
## r[16] 1.00
## r[17] 1.00
## r[18] 1.00
## r[19] 1.00
## r[20] 1.00
## beta  1.00
## p[1]  1.00
## p[2]  1.00
## p[3]  1.00
## p[4]  1.00
## p[5]  1.00
## p[6]  1.00
## p[7]  1.00
## p[8]  1.00
## p[9]  1.00
## p[10] 1.00
## p[11] 1.00
## p[12] 1.00
## p[13] 1.00
## p[14] 1.00
## p[15] 1.00
## p[16] 1.00
## p[17] 1.00
## p[18] 1.00
## p[19] 1.00
## p[20] 1.00
## lp__  1.01
##
## Samples were drawn using NUTS(diag_e) at Fri Nov 10 16:27:14 2017.
## 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)
```

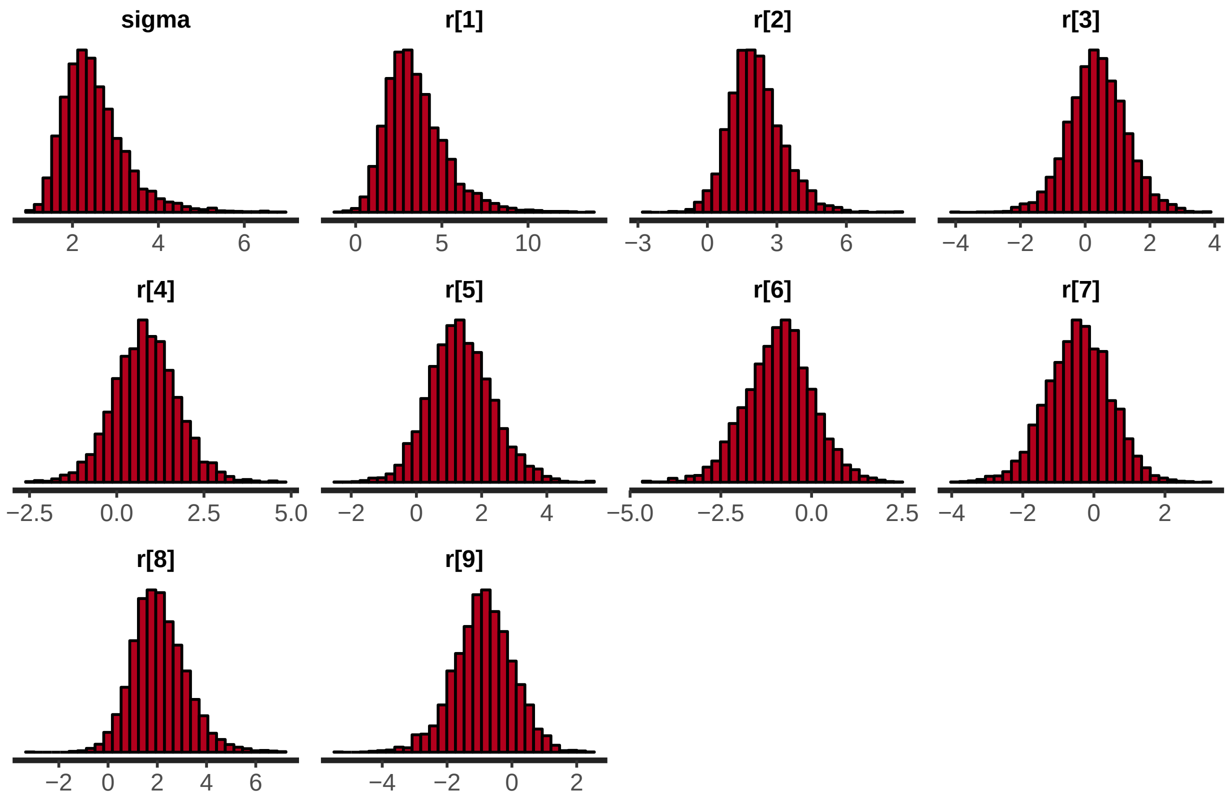
```
## 'pars' not specified. Showing first 10 parameters by default.
```



パラメータのサンプルのヒストグラム

```
stan_hist(fit)
```

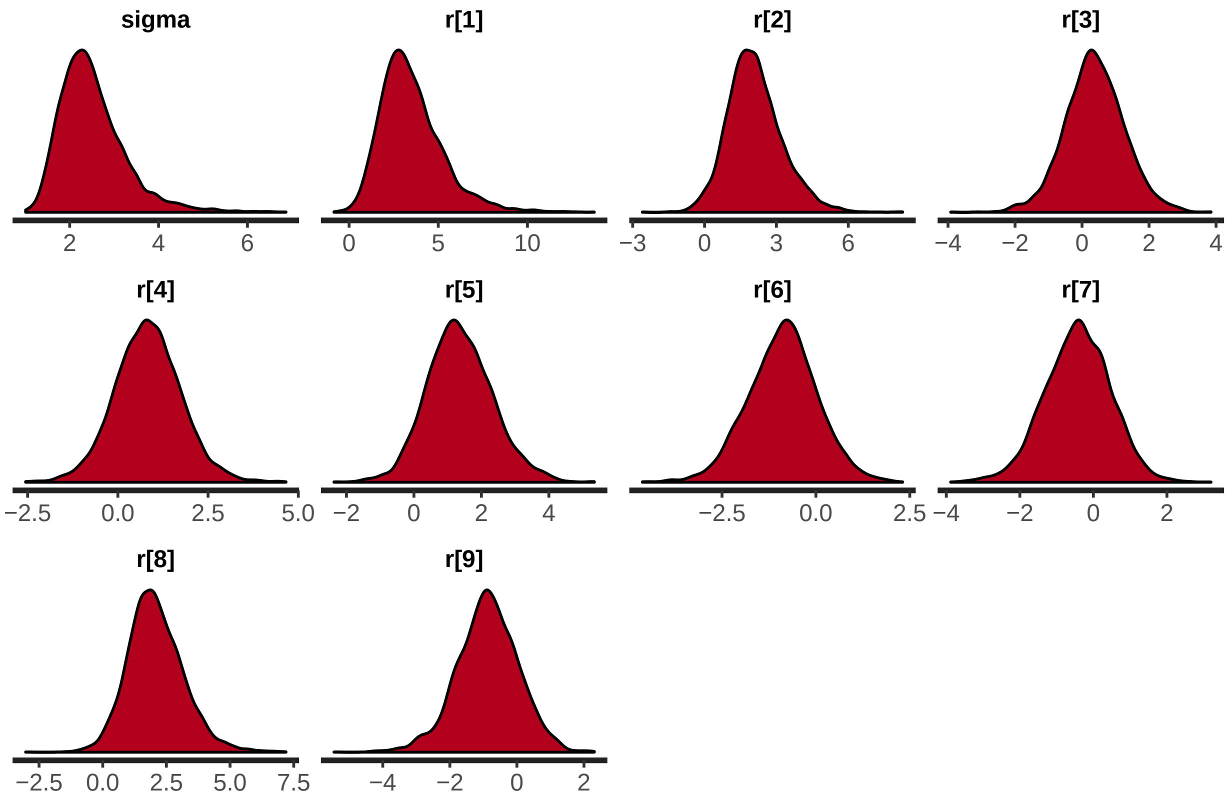
```
## 'pars' not specified. Showing first 10 parameters by default.
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



パラメータ事後分布の密度推定

```r
stan_dens(fit)
```

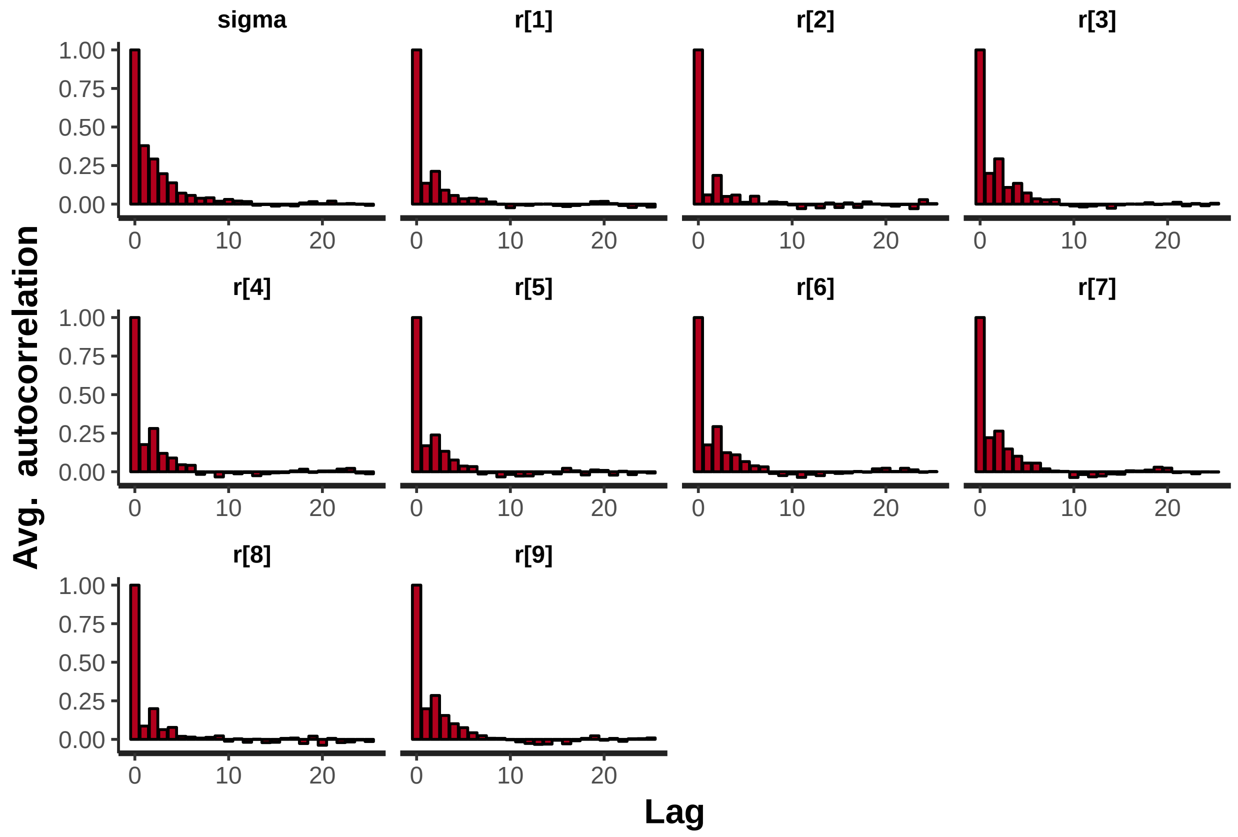
```
## 'pars' not specified. Showing first 10 parameters by default.
```



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。

ポアソン回帰

釣りの結果を推定しよう。

場所による差、天気による差、

対数リンク

状態空間モデル

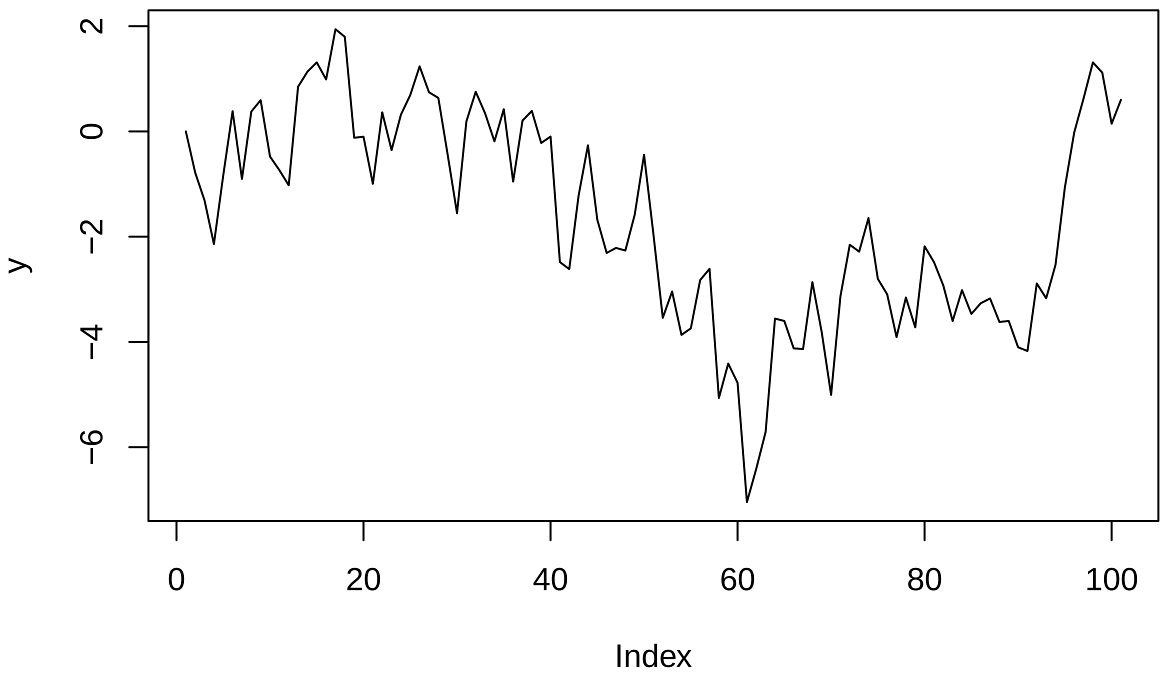
状態空間モデルと一般的な時系列モデル(AR等)の違い

線形正規モデルから解説する。 特別な場合がARなど。

状態方程式と観測方程式

観測値と状態に分けて考える。 ランダムウォーク

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



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、

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

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 = 1000, chains = 4)
```

```
##
## SAMPLING FOR MODEL 'c82f834f463f59b8c4dc621aec7e86ce' NOW (CHAIN 1).
##
## Gradient evaluation took 2e-05 seconds
## 1000 transitions using 10 leapfrog steps per transition would take 0.2 seconds.
## Adjust your expectations accordingly!
##
##
## Iteration:   1 / 1000 [  0%]  (Warmup)
## Iteration: 100 / 1000 [ 10%]  (Warmup)
## Iteration: 200 / 1000 [ 20%]  (Warmup)
## Iteration: 300 / 1000 [ 30%]  (Warmup)
## Iteration: 400 / 1000 [ 40%]  (Warmup)
## Iteration: 500 / 1000 [ 50%]  (Warmup)
## Iteration: 501 / 1000 [ 50%]  (Sampling)
## Iteration: 600 / 1000 [ 60%]  (Sampling)
## Iteration: 700 / 1000 [ 70%]  (Sampling)
## Iteration: 800 / 1000 [ 80%]  (Sampling)
## Iteration: 900 / 1000 [ 90%]  (Sampling)
## Iteration: 1000 / 1000 [100%]  (Sampling)
##
##  Elapsed Time: 1.31494 seconds (Warm-up)
##                0.346421 seconds (Sampling)
##                1.66136 seconds (Total)
##
##
## SAMPLING FOR MODEL 'c82f834f463f59b8c4dc621aec7e86ce' NOW (CHAIN 2).
##
## 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 / 1000 [  0%]  (Warmup)
## Iteration: 100 / 1000 [ 10%]  (Warmup)
## Iteration: 200 / 1000 [ 20%]  (Warmup)
## Iteration: 300 / 1000 [ 30%]  (Warmup)
## Iteration: 400 / 1000 [ 40%]  (Warmup)
## Iteration: 500 / 1000 [ 50%]  (Warmup)
## Iteration: 501 / 1000 [ 50%]  (Sampling)
## Iteration: 600 / 1000 [ 60%]  (Sampling)
## Iteration: 700 / 1000 [ 70%]  (Sampling)
## Iteration: 800 / 1000 [ 80%]  (Sampling)
## Iteration: 900 / 1000 [ 90%]  (Sampling)
## Iteration: 1000 / 1000 [100%]  (Sampling)
##
##  Elapsed Time: 1.12053 seconds (Warm-up)
##                0.260649 seconds (Sampling)
##                1.38118 seconds (Total)
##
##
## SAMPLING FOR MODEL 'c82f834f463f59b8c4dc621aec7e86ce' 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 / 1000 [  0%]  (Warmup)
## Iteration: 100 / 1000 [ 10%]  (Warmup)
## Iteration: 200 / 1000 [ 20%]  (Warmup)
## Iteration: 300 / 1000 [ 30%]  (Warmup)
## Iteration: 400 / 1000 [ 40%]  (Warmup)
## Iteration: 500 / 1000 [ 50%]  (Warmup)
## Iteration: 501 / 1000 [ 50%]  (Sampling)
## Iteration: 600 / 1000 [ 60%]  (Sampling)
## Iteration: 700 / 1000 [ 70%]  (Sampling)
## Iteration: 800 / 1000 [ 80%]  (Sampling)
## Iteration: 900 / 1000 [ 90%]  (Sampling)
## Iteration: 1000 / 1000 [100%]  (Sampling)
##
##  Elapsed Time: 2.17013 seconds (Warm-up)
##                0.941001 seconds (Sampling)
##                3.11113 seconds (Total)
##
##
## SAMPLING FOR MODEL 'c82f834f463f59b8c4dc621aec7e86ce' NOW (CHAIN 4).
##
## 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 / 1000 [  0%]  (Warmup)
## Iteration: 100 / 1000 [ 10%]  (Warmup)
## Iteration: 200 / 1000 [ 20%]  (Warmup)
## Iteration: 300 / 1000 [ 30%]  (Warmup)
## Iteration: 400 / 1000 [ 40%]  (Warmup)
## Iteration: 500 / 1000 [ 50%]  (Warmup)
## Iteration: 501 / 1000 [ 50%]  (Sampling)
## Iteration: 600 / 1000 [ 60%]  (Sampling)
## Iteration: 700 / 1000 [ 70%]  (Sampling)
## Iteration: 800 / 1000 [ 80%]  (Sampling)
## Iteration: 900 / 1000 [ 90%]  (Sampling)
## Iteration: 1000 / 1000 [100%]  (Sampling)
##
##  Elapsed Time: 1.42502 seconds (Warm-up)
##                0.445435 seconds (Sampling)
##                1.87045 seconds (Total)
```

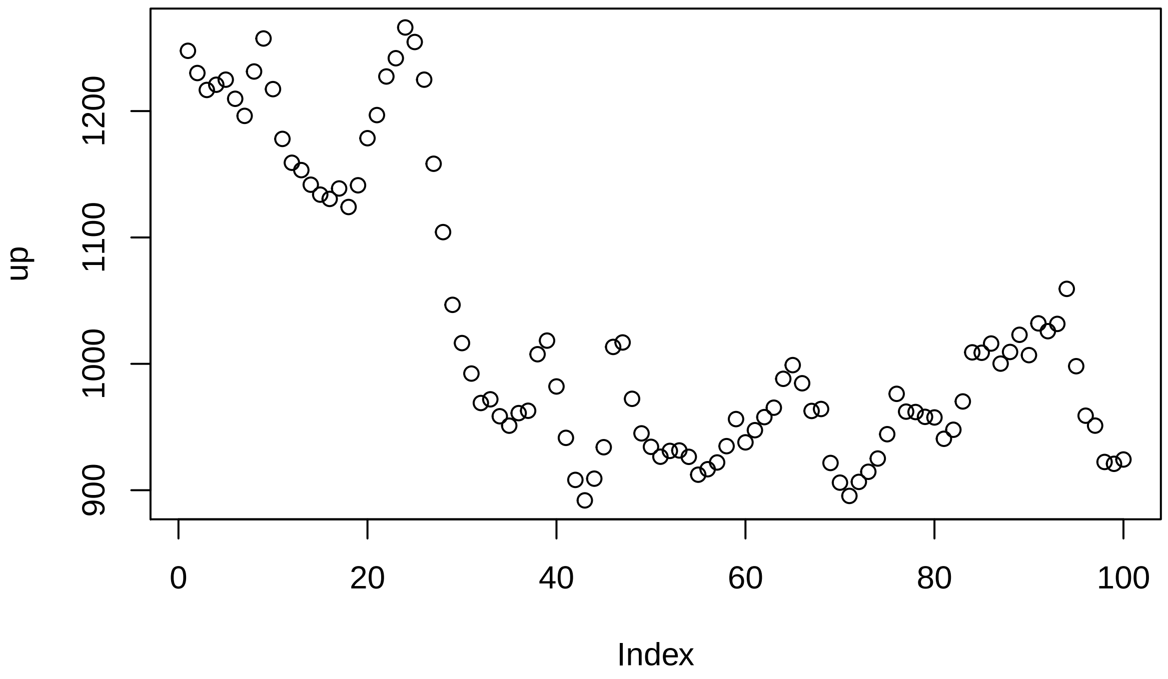
```
## Warning: There were 4 chains where the estimated Bayesian Fraction of Missing Information was low. S
## http://mc-stan.org/misc/warnings.html#bfmi-low
```

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

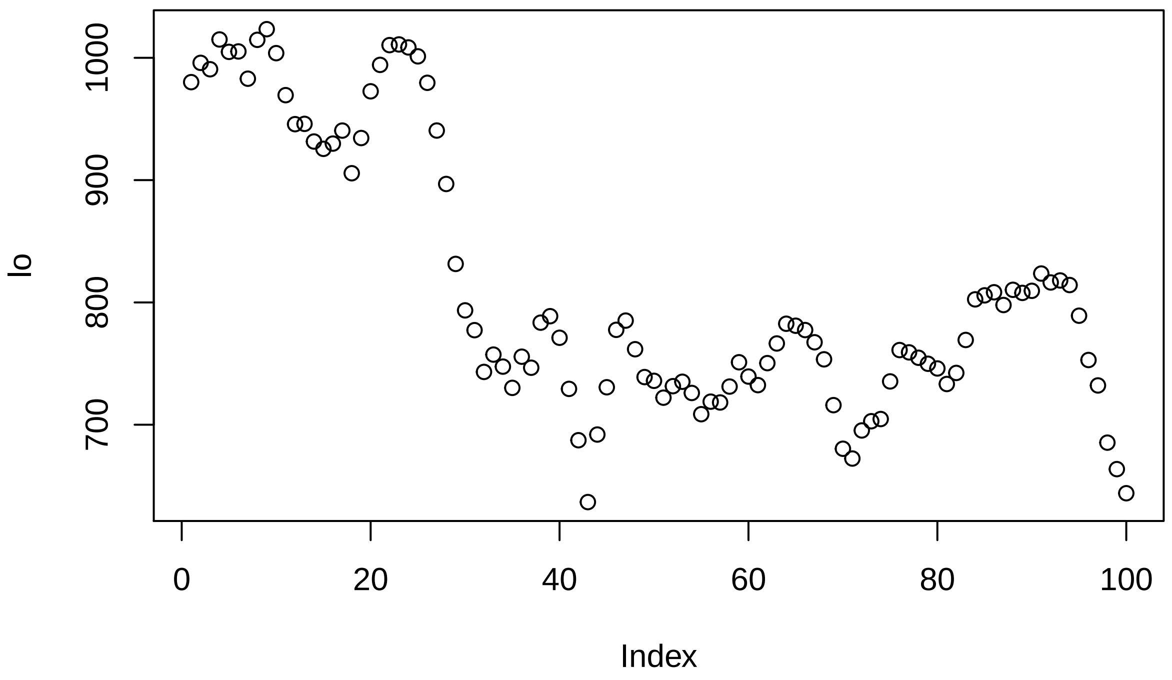
```
#
x<-rstan::extract(fit)$x
#
head(x)
```

```
##
## iterations     [,1]     [,2]     [,3]     [,4]      [,5]     [,6]     [,7]
##       [1,] 1182.379 1159.529 1121.307 1105.147 1083.3589 1080.375 1106.213
##       [2,] 1111.046 1090.037 1058.899 1019.730  980.8239 1053.889 1095.575
##       [3,] 1077.371 1053.650 1035.277 1093.222 1124.3479 1100.743 1005.469
##       [4,] 1114.905 1117.805 1107.962 1108.065 1146.7533 1151.156 1117.758
##       [5,] 1067.056 1068.012 1051.595 1083.842 1093.5616 1060.369 1068.065
##       [6,] 1142.786 1122.747 1104.869 1110.352 1061.3885 1042.649  985.748
##
## iterations     [,8]     [,9]    [,10]    [,11]    [,12]    [,13]    [,14]
##       [1,] 1134.053 1181.401 1160.211 1191.760 1159.102 1057.025 1022.802
##       [2,] 1078.814 1091.867 1075.408 1083.458 1057.227 1015.930 1020.108
##       [3,] 1132.621 1113.754 1120.722 1066.752 1029.123 1064.112 1026.603
##       [4,] 1131.024 1129.251 1132.274 1113.211 1094.091 1073.534 1059.265
##        [5,] 1089.411 1090.842 1063.898 1048.651 1048.383 1044.509 1065.195
##        [6,] 1091.048 1108.606 1075.909 1093.453 1043.479 1111.276 1111.907
##
## iterations      [,15]    [,16]    [,17]    [,18]    [,19]    [,20]    [,21]
##        [1,]  980.6653 1039.716 1019.392 1022.352 1131.732 1181.283 1174.795
##        [2,] 1038.3769 1051.857 1077.553 1084.296 1094.480 1106.005 1105.089
##        [3,]  985.8908  988.450 1049.867  905.643 1008.064 1014.834 1059.300
##        [4,] 1066.1969 1081.618 1080.735 1051.028 1062.445 1060.951 1104.271
##        [5,] 1080.3048 1022.702 1039.362 1034.051 1028.562 1073.226 1099.311
##        [6,] 1087.1512 1099.293 1100.148 1100.103 1139.829 1177.223 1137.200
##
## iterations     [,22]    [,23]    [,24]    [,25]    [,26]    [,27]     [,28]
##        [1,] 1180.332 1137.266 1111.376 1137.721 1185.679 1084.332 1079.8767
##        [2,] 1087.206 1057.306 1093.908 1095.428 1032.595 1002.170  995.6431
##        [3,] 1154.883 1230.207 1224.306 1213.293 1167.191 1066.833  933.1819
##        [4,] 1126.356 1118.477 1056.277 1050.595 1021.384 1007.445  984.8980
##        [5,] 1143.768 1168.292 1157.575 1134.874 1107.355 1085.840 1031.3348
##        [6,] 1182.695 1076.702 1022.657 1044.729 1034.462  954.440  954.1092
##
## iterations      [,29]    [,30]    [,31]    [,32]    [,33]    [,34]    [,35]
##        [1,]  988.3337 928.0389 873.3482 762.5567 735.0175 761.1732 811.0299
##        [2,]  959.0313 986.1616 952.4151 955.4640 906.7778 890.5698 878.8338
##        [3,]  848.1642 863.2489 828.5433 743.1970 817.1509 844.4844 797.4035
##        [4,]  912.1016 890.8886 873.0965 875.0898 876.9228 849.3860 819.1949
##        [5,] 1013.3301 971.4844 927.8545 916.3805 924.3561 904.8278 899.6841
##        [6,]  953.9968 938.6246 895.7355 898.5996 876.8573 874.6986 821.9218
##
## iterations     [,36]    [,37]     [,38]     [,39]    [,40]    [,41]
##        [1,] 857.2443 859.1795  886.8062  860.3774 802.4757 807.6594
##        [2,] 891.5422 916.6969  916.9012  899.1313 864.2810 823.2722
##        [3,] 863.1145 951.9868 1015.4124 1008.0857 920.2105 873.7634
##        [4,] 834.5604 870.3413  888.1861  892.6921 898.7361 894.6481
##        [5,] 855.1445 829.6234  822.2582  806.8752 823.3879 798.3953
##        [6,] 888.0513 834.1214  880.0120  835.6389 883.4613 846.6044
##
## iterations     [,42]    [,43]    [,44]    [,45]    [,46]    [,47]    [,48]
##        [1,] 783.8362 809.0406 805.0778 816.5262 845.0562 864.0398 878.4741
##        [2,] 818.0753 776.9364 788.2265 770.2543 798.6433 824.2163 791.4644
##        [3,] 837.1191 750.0655 833.0696 855.1930 956.9312 943.8958 989.4226
##        [4,] 889.5093 849.0696 833.6635 805.8250 828.8116 887.5773 901.0681
##        [5,] 791.5946 782.0151 783.5979 778.6634 820.2227 834.7479 837.7726
##        [6,] 797.8016 746.7037 788.1229 840.8966 848.6339 822.6798 784.9328
##
## iterations     [,49]    [,50]    [,51]    [,52]    [,53]    [,54]    [,55]
##        [1,] 871.6814 905.0708 869.6492 872.7558 887.7927 844.4604 836.5169
##        [2,] 773.4154 785.3081 787.6036 790.3518 780.6482 758.3185 787.3641
##        [3,] 920.3131 871.6446 832.0964 883.5871 833.2858 822.9692 760.8175
##        [4,] 886.2385 915.8603 862.0414 847.5559 853.9261 873.8443 857.4048
##        [5,] 852.5101 844.5053 897.1499 897.5745 851.1938 837.6679 823.3287
##        [6,] 806.5706 883.5772 934.1025 901.9723 870.9671 821.4830 878.9994
##
## iterations     [,56]    [,57]    [,58]    [,59]    [,60]    [,61]    [,62]
##        [1,] 872.9182 885.5668 900.0969 862.9793 924.4026 902.6099 852.9680
##        [2,] 815.1898 823.5419 816.3661 815.6267 798.2275 815.2345 807.1304
##        [3,] 785.0438 739.8679 872.9194 922.5718 905.7447 871.0730 912.9682
##        [4,] 820.4126 815.5009 824.2284 833.2900 845.9225 809.9090 817.3150
##        [5,] 807.2794 825.7776 822.5440 805.5746 785.5752 762.5684 803.9216
##        [6,] 847.0606 808.9296 857.3207 800.8746 794.4609 757.7220 821.8190
##
## iterations    [,63]    [,64]    [,65]    [,66]    [,67]    [,68]    [,69]
##        [1,] 806.0107 805.1650 800.7469 823.5053 823.4123 776.9818 784.8097
##        [2,] 852.3082 841.1712 843.5323 858.9239 859.9233 871.1937 852.5266
##        [3,] 934.8111 938.4858 952.2916 939.2395 859.6173 925.5920 811.8085
##        [4,] 892.1655 876.1749 839.3026 903.0661 880.6276 869.7879 817.9667
##        [5,] 779.1983 784.3468 816.9368 833.7659 833.6438 841.0912 846.3949
##        [6,] 872.1405 882.1857 810.7249 825.7713 853.0527 785.8826 757.5261
##
## iterations    [,70]    [,71]    [,72]    [,73]    [,74]    [,75]    [,76]
##        [1,] 709.4203 776.7325 797.9510 829.6728 891.7736 857.2331 827.4570
##        [2,] 853.3694 850.6497 844.9821 856.7402 815.7361 826.5428 821.6042
##        [3,] 740.9105 775.5828 782.9296 806.4619 844.0782 886.3443 980.7409
##        [4,] 827.9748 834.0329 823.0862 786.9274 832.5759 867.1806 898.4915
##        [5,] 822.3031 811.5701 770.9057 720.9611 759.5974 810.5663 851.0005
##        [6,] 818.0170 779.0484 754.4611 775.6448 722.2602 788.6617 771.6757
##
## iterations    [,77]    [,78]    [,79]    [,80]    [,81]    [,82]    [,83]
##        [1,] 795.5351 826.0044 775.1075 748.7280 818.2558 818.4314 819.4453
##        [2,] 819.8062 842.0730 871.9649 847.7674 832.8301 876.0308 886.2850
##        [3,] 868.3600 813.9707 824.5634 816.5602 727.2815 799.9837 829.3998
##        [4,] 937.1881 891.3999 897.8928 895.4622 882.9078 912.2391 895.1595
##        [5,] 875.1189 860.3846 845.7020 839.0986 860.5623 860.3518 900.4985
##        [6,] 783.4933 767.9779 773.9209 817.4152 784.5756 828.8439 810.9568
##
## iterations    [,84]    [,85]    [,86]    [,87]    [,88]    [,89]    [,90]
##        [1,] 894.9707 880.7396 862.5786 927.6089 892.7759 929.5812 921.3107
##        [2,] 901.6628 858.8860 876.5049 892.5929 867.5649 844.6858 850.9432
##        [3,] 895.8917 904.6019 895.7052 836.8092 855.5165 875.4013 897.3431
##        [4,] 891.2791 866.4256 870.4134 876.7121 883.0795 891.4360 925.5120
##        [5,] 897.8641 925.0196 850.8724 864.1993 849.3533 847.4689 829.8650
##        [6,] 847.8191 862.8380 837.9551 848.2283 911.2638 949.8015 973.5994
##
## iterations    [,91]    [,92]    [,93]    [,94]    [,95]    [,96]    [,97]
##        [1,] 915.5433 893.1365 872.4405 846.1003 853.3336 773.6446 791.7647
##        [2,] 846.1954 858.3870 852.0355 807.1534 807.5646 828.4310 769.2749
##        [3,] 980.1123 968.0039 994.0115 907.8684 879.5668 784.2841 804.0079
##        [4,] 925.9428 891.5597 905.1246 943.2817 895.8369 840.7698 848.0847
##        [5,] 825.5526 859.2396 859.1052 894.9438 890.9310 893.5245 873.3310
##        [6,] 950.4500 958.5304 883.3149 890.6757 911.6900 936.3396 855.8868
##
## iterations    [,98]    [,99]   [,100]
##        [1,] 688.2670 717.7142 692.3746
##        [2,] 784.8499 783.6975 774.8089
##        [3,] 712.3169 688.0411 697.3341
##        [4,] 836.5395 875.6981 885.4936
##        [5,] 816.7320 803.0704 794.2998
##        [6,] 811.1893 831.7246 793.2175
```

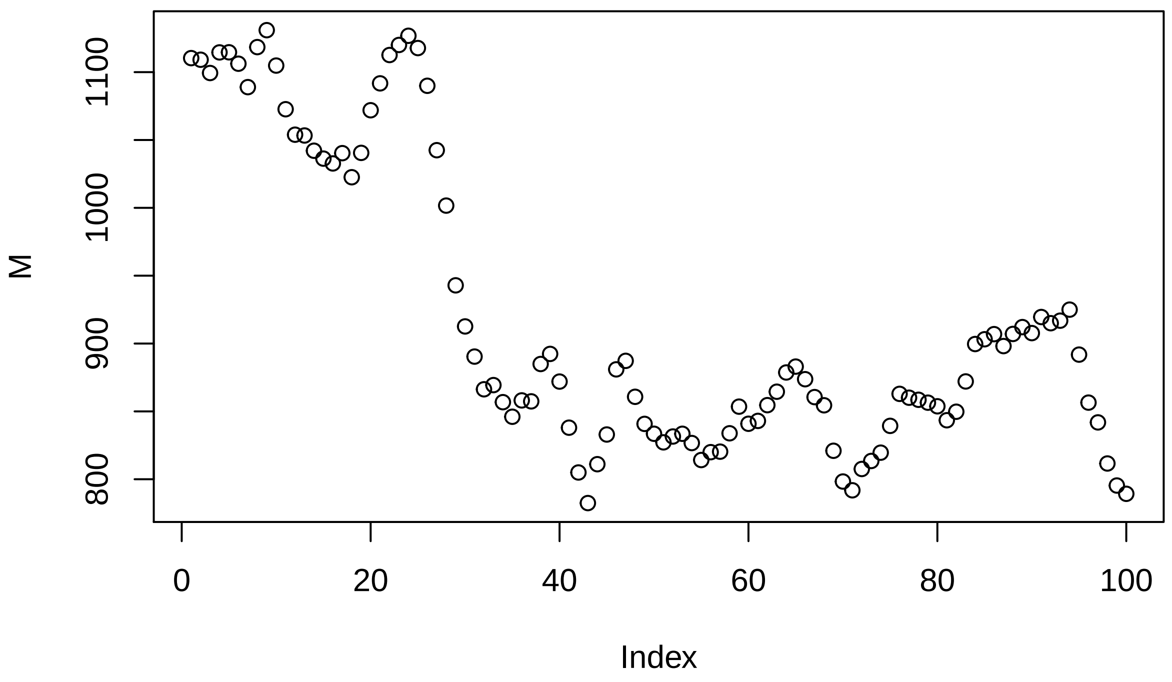
```r
d2<-data.frame(x)
up<-c()
lo<-c()
M<-c()
#for   aggregate
for(i in 1:length(d2)){
  up[i]<-quantile(d2[,i],0.975)
  lo[i]<-quantile(d2[,i],0.025)
  M[i]<-mean(d2[,i])
}
plot(up)
```



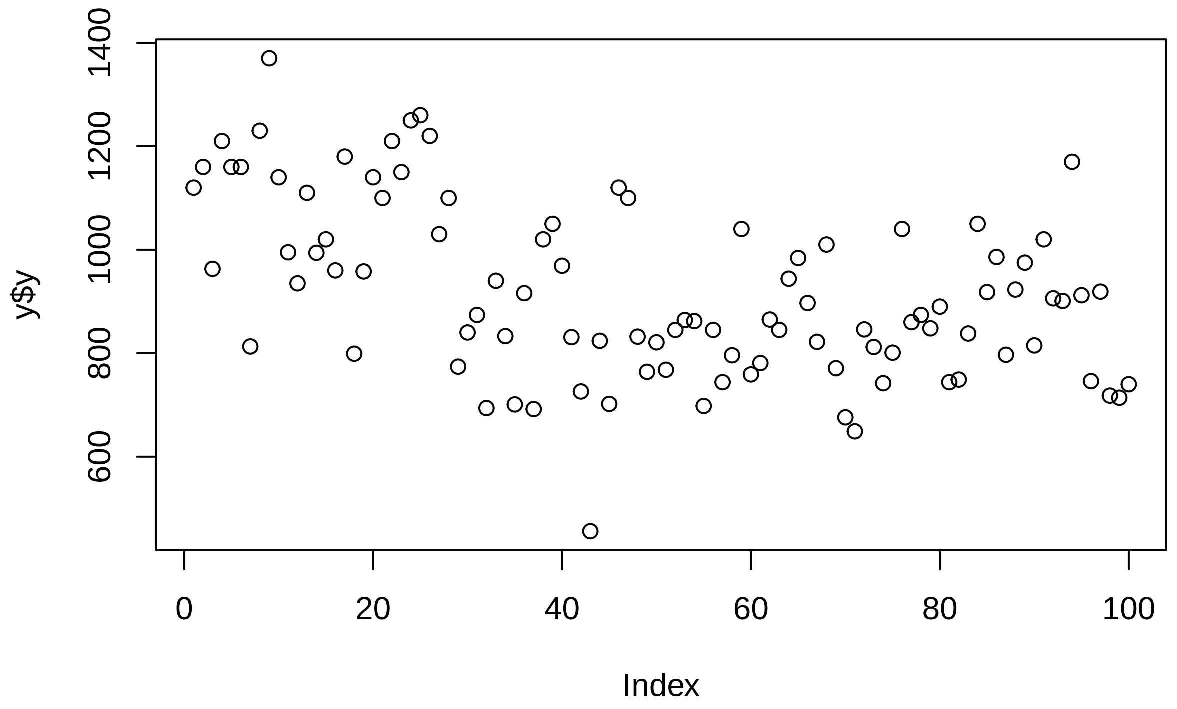
```r
plot(lo)
```



```r
plot(M)
```



```r
plot(y$y)
```



```
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)
```

トレンドモデル

トレンドモデル

季節変動