Chapter 8

The Monte Carlo origin (MCMC)

Markov Chain Monte Carlo (MCMC) has two components:

- The Monte Carlo,
- The Markov Chain.

The Monte Carlo origin (MCMC)

"The first thoughts and attempts I made to practice [the Monte Carlo method] *were suggested by a question which occurred to me in 1946 as I was convalescing from an illness and playing solitaires. The question was what are the chances that a Canfield solitaire laid out with 52 cards will come out successfully? After spending a lot of time trying to estimate them by pure combinatorial calculations, I wondered whether a more practical method than 'abstract thinking' might not be to lay it out say one hundred times and simply observe and count the number of successful plays. This was already possible to envisage with the beginning of the new era of fast computers, and I immediately thought of problems of neutron diffusion and other questions of mathematical physics, and more generally how to change processes described by certain differential equations into an equivalent form interpretable as a succession of random operations. Later* [in 1946], *I described the idea to John von Neumann, and we began to plan actual calculations."*

–Stanisław Ulam

CLT

If we sample $X_1, X_2, \ldots, X_n$ random variables that are independent and identically distribution the

$$\frac{X_1 + X_2 + \ldots + X_n}{n} \to \mathbf{N}(\mu, \frac{\sigma}{\sqrt{n}}),$$

where $\mu = \mathbb{E}[X]$. How does this help analyzing output from a Monte Carlo algorithm?

A Markov chain, X_t, is a time series with the following property:

Memoryless

Given $X_0, X_1, \ldots, X_t$ the distribution of X_{t+1} satisfies

$$f(X_{t+1}|X_t, X_{t-1}, \ldots, X_0) = f(X_{t+1}|X_t).$$

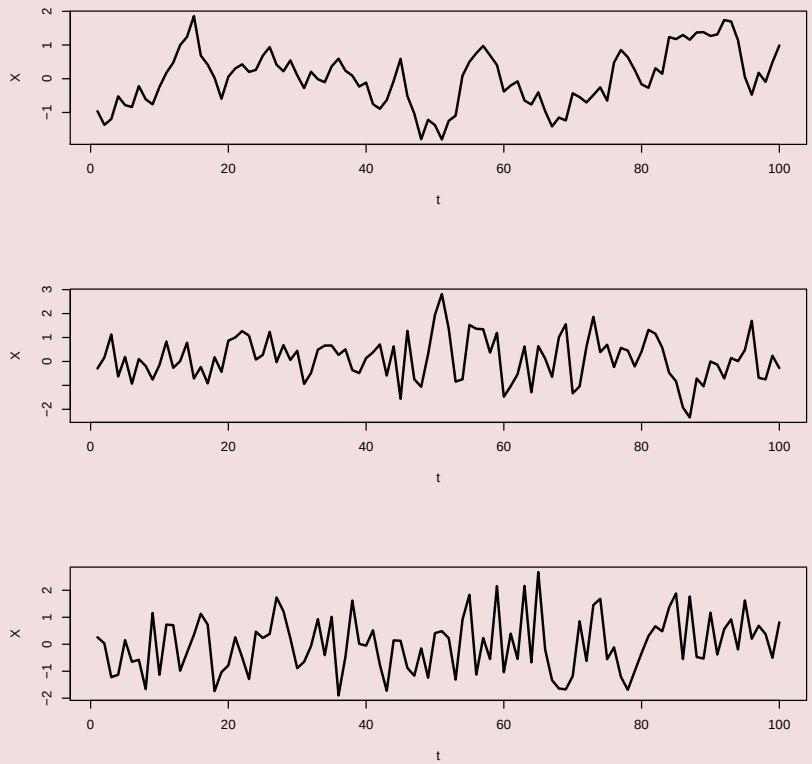
Figure : Andrey Markov

Example AR(1)

Three examples of AR(1) processes:

$$X_t = aX_{t-1} + \epsilon_t$$
$$\epsilon_t \sim N(0, \sigma)$$

1. $a = 0.9, \sigma = \sqrt{1 - 0.9^2}$
2. $a = 0.1, \sigma = \sqrt{1 - 0.1^2}$
3. $a = 0, \sigma = 1$



Stationary distribution

- For all three processes if we can thin the series:

$$X_T, X_{2T}, X_{3T}, \ldots$$

  where T is large.
- It turns out that all three series have the same **stationary distribution**, p.

Stationary distribution, p

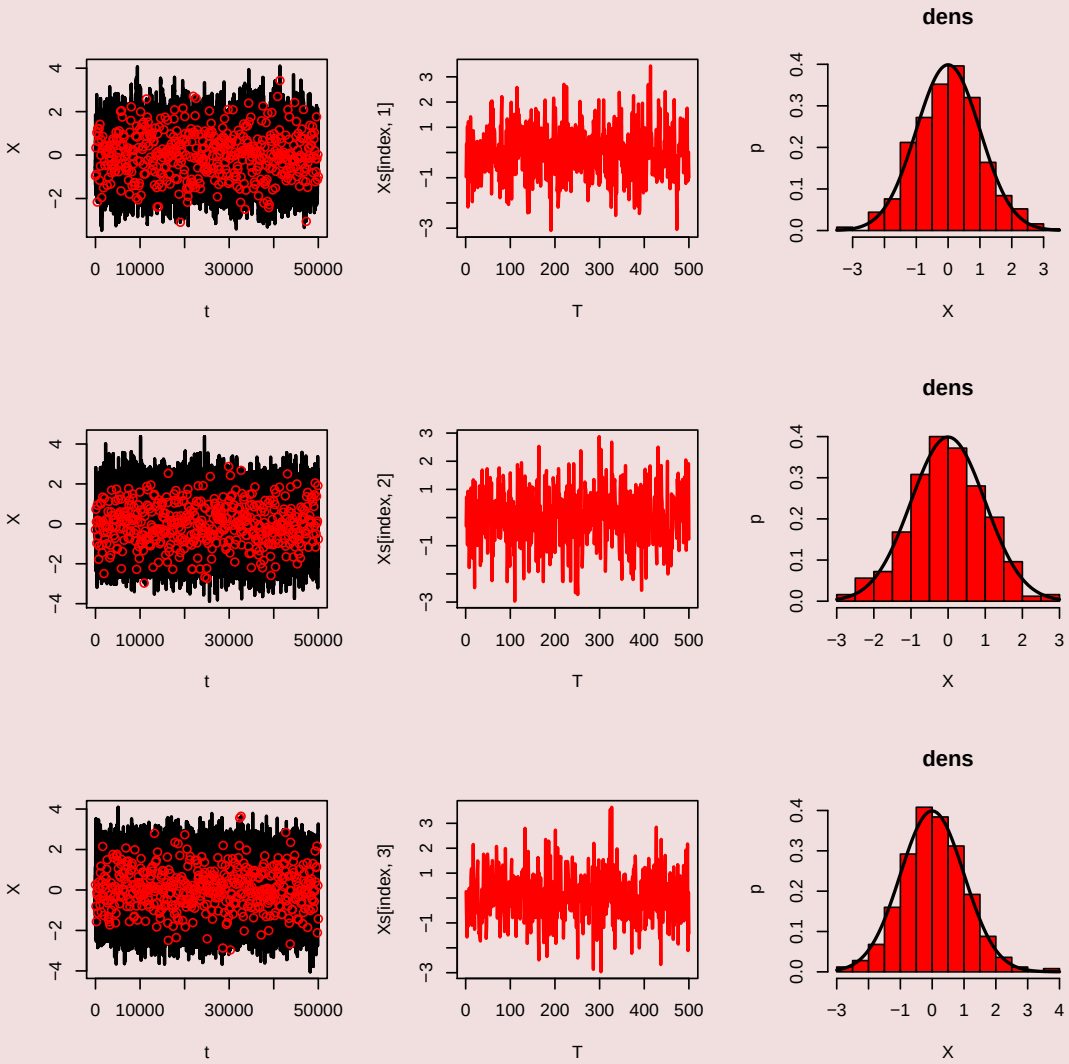
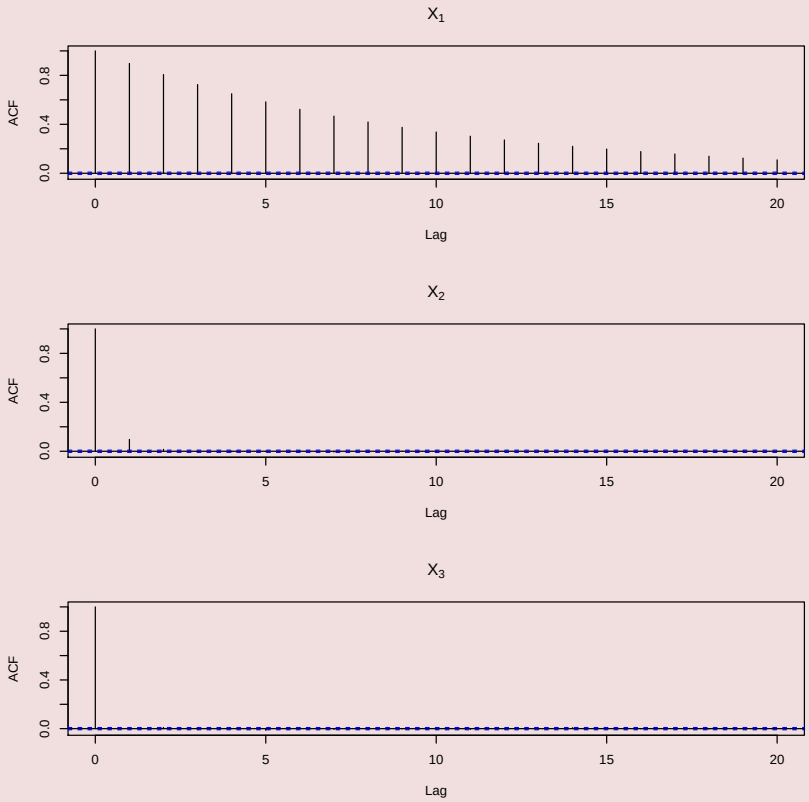


Figure : $T = 500$

The dependence of the chains

For time series one often look at the autocorrelation (ACF) function:



Generate a Markov Chain with stationary distribution p That is we choose a density f such that the stationary distribution is p.

THE JOURNAL OF CHEMICAL PHYSICS VOLUME 21, NUMBER 6 JUNE, 1953

Equation of State Calculations by Fast Computing Machines

NICHOLAS METROPOLIS, ARIANNA W. ROSENBLUTH, MARSHALL N. ROSENBLUTH, AND AUGUSTA H. TELLER,
Los Alamos Scientific Laboratory, Los Alamos, New Mexico

AND

EDWARD TELLER,* *Department of Physics, University of Chicago, Chicago, Illinois*

(Received March 6, 1953)

A general method, suitable for fast computing machines, for investigating such properties as equations of state for substances consisting of interacting individual molecules is described. The method consists of a modified Monte Carlo integration over configuration space. Results for the two-dimensional rigid-sphere system have been obtained on the Los Alamos MANIAC and are presented here. These results are compared to the free volume equation of state and to a four-term virial coefficient expansion.

Biometrika (1970), **57**, 1, *p.* 97
Printed in Great Britain

Monte Carlo sampling methods using Markov chains and their applications

BY W. K. HASTINGS
University of Toronto

SUMMARY

A generalization of the sampling method introduced by Metropolis *et al.* (1953) is presented along with an exposition of the relevant theory, techniques of application and methods and difficulties of assessing the error in Monte Carlo estimates. Examples of the methods, including the generation of random orthogonal matrices and potential applications of the methods to numerical problems arising in statistics, are discussed.

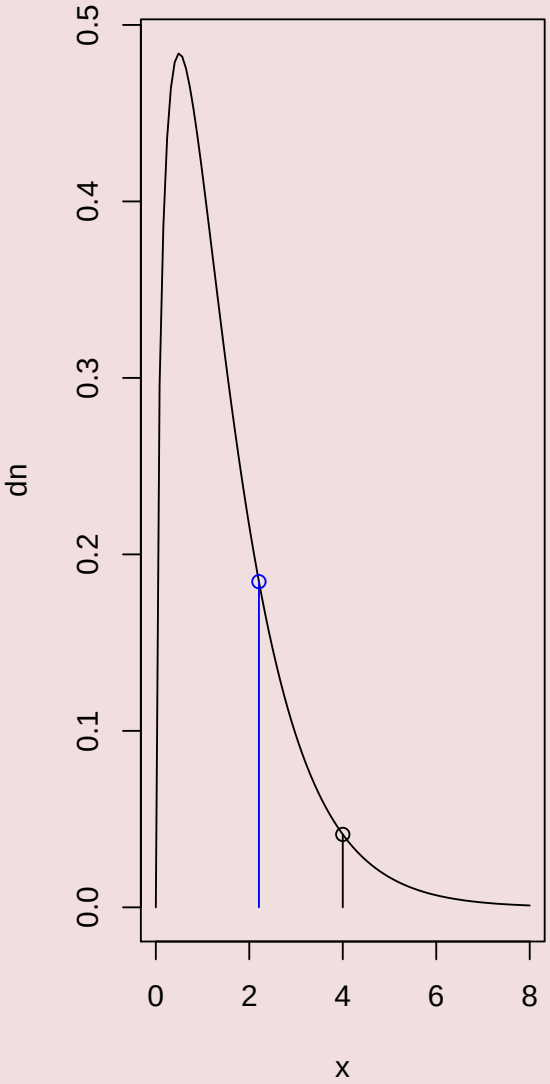
1. INTRODUCTION

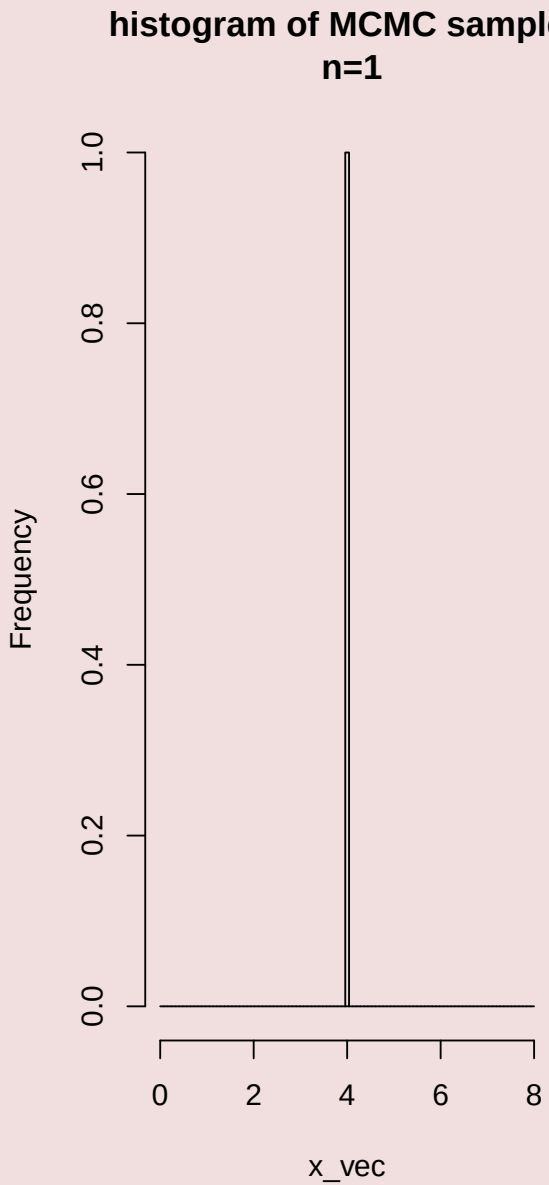
Metropolis-Hastings algorithm

One iteration of a symmetric random walk

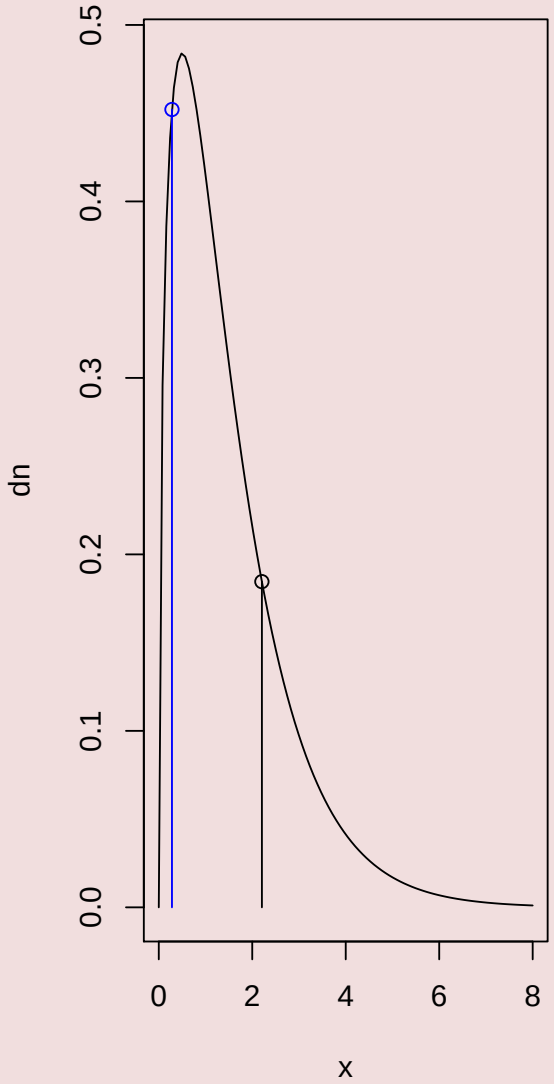
- Generate a symmetric variable centered around the previous value, most common Normal $X^* \sim N(X^{old}, \sigma)$.
- Generate $U \sim U[0,1]$.
- The new value is

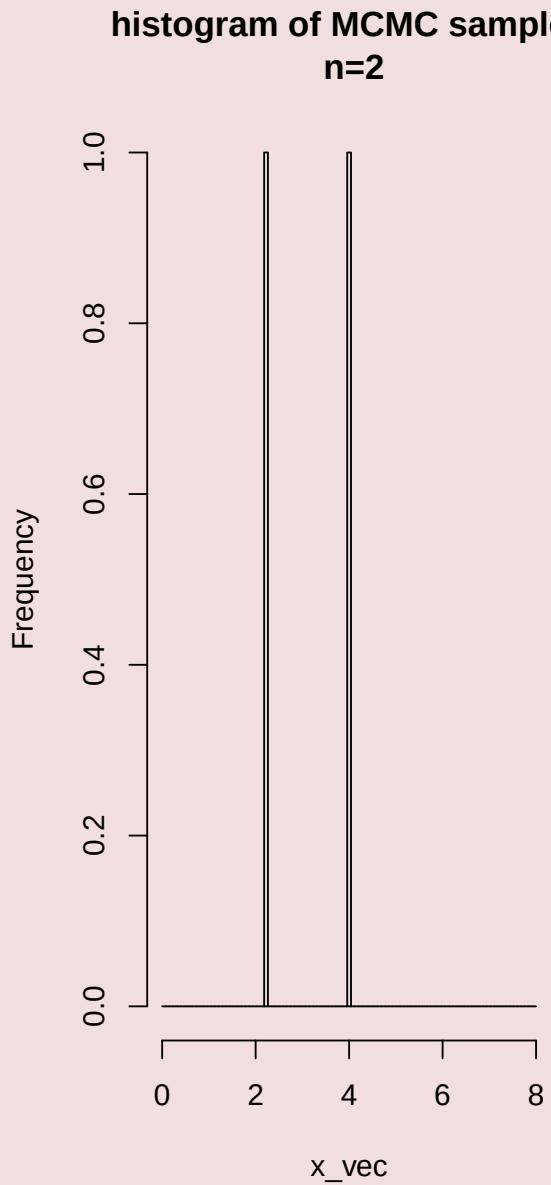
$$X^{new} = \begin{cases} X^* & \text{if } U \leq \frac{p(X^*)}{p(X^{old})}, \\ X^{old} & \text{otherwise.} \end{cases}$$


dn
0.0
0.1
0.2
0.3
0.4
0.5
0
2
4
6
8
x

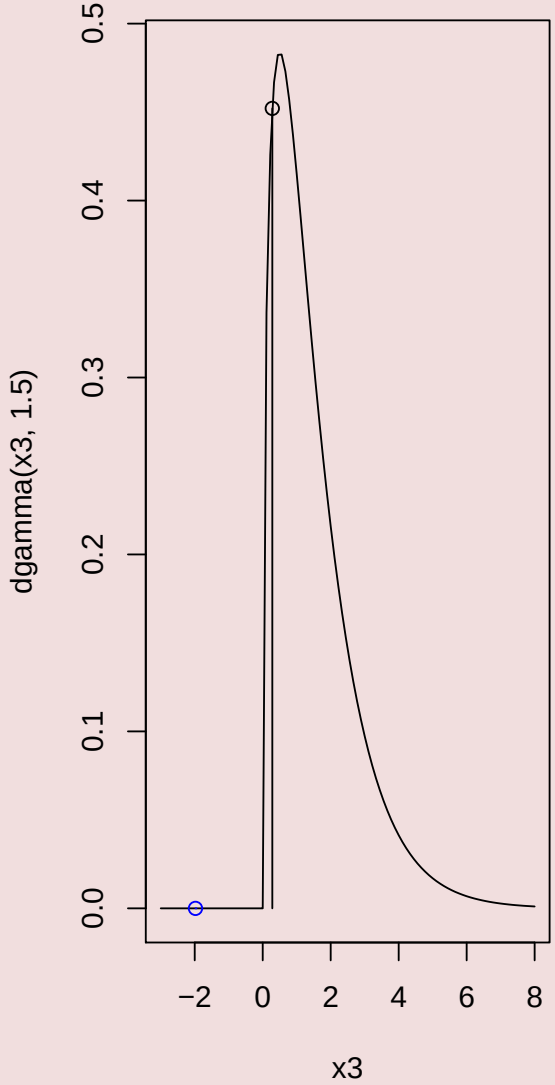

histogram of MCMC samples
n=1
Frequency
0.0
0.2
0.4
0.6
0.8
1.0
0
2
4
6
8
x_vec

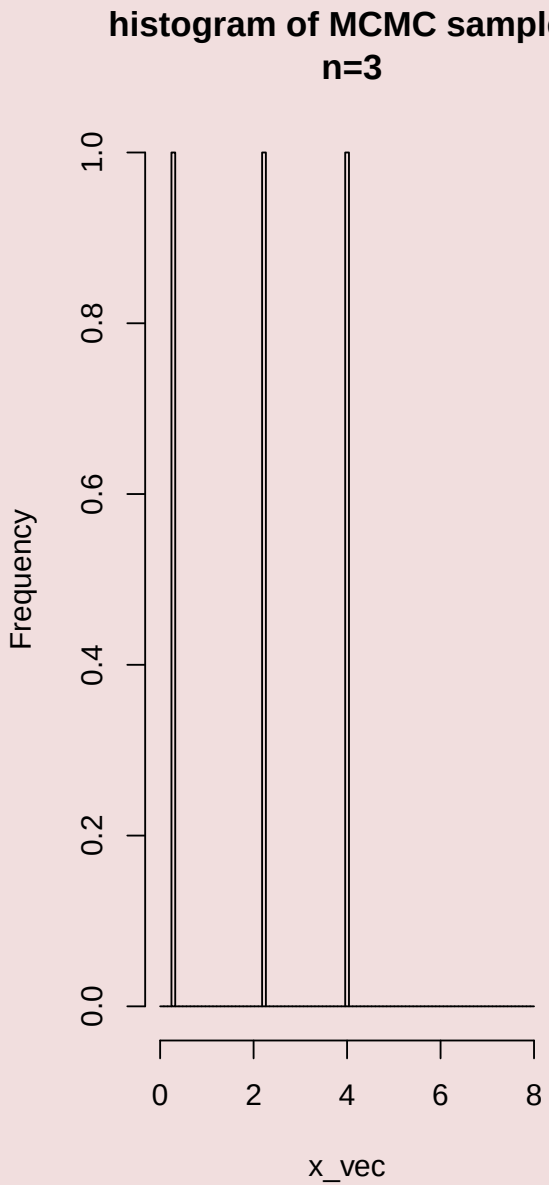
In Action


dn
0.0
0.1
0.2
0.3
0.4
0.5
0
2
4
6
8
x

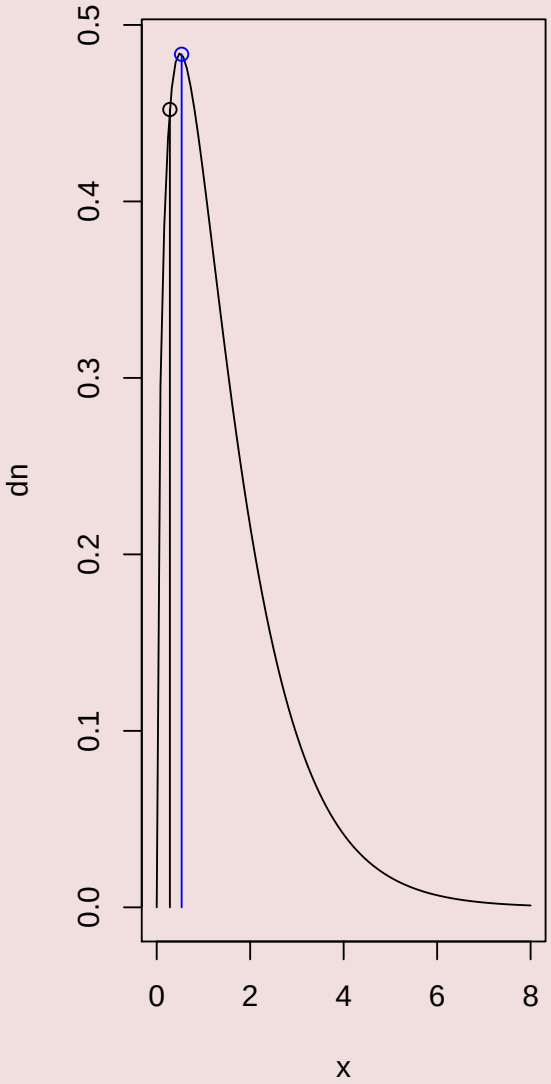

histogram of MCMC samples
n=2
Frequency
0.0
0.2
0.4
0.6
0.8
1.0
0
2
4
6
8
x_vec

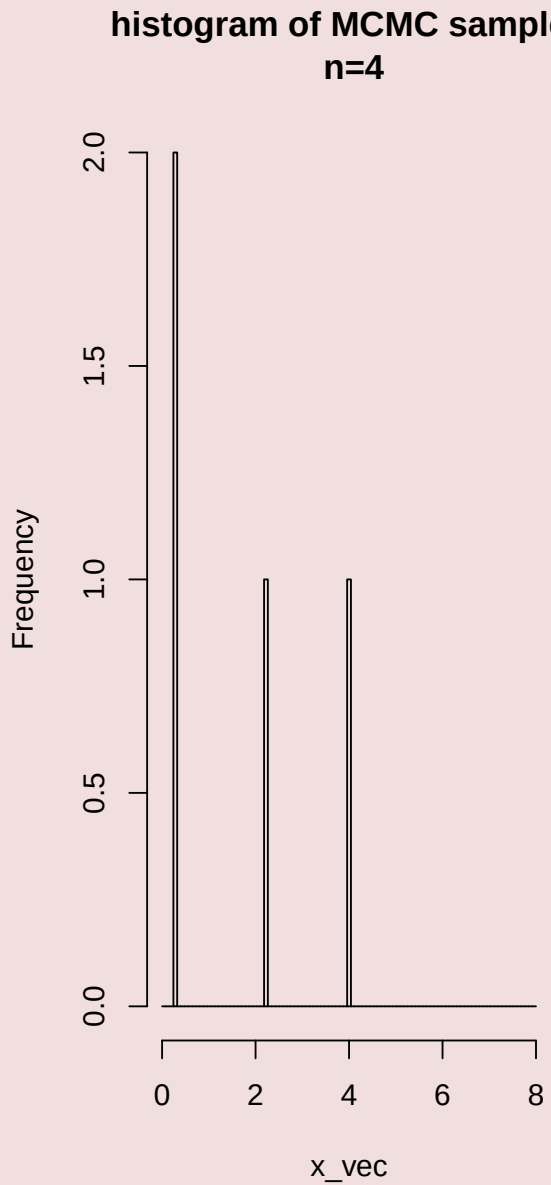
In Action


dgamma(x3, 1.5)
0.0
0.1
0.2
0.3
0.4
0.5
−2
0
2
4
6
8
x3

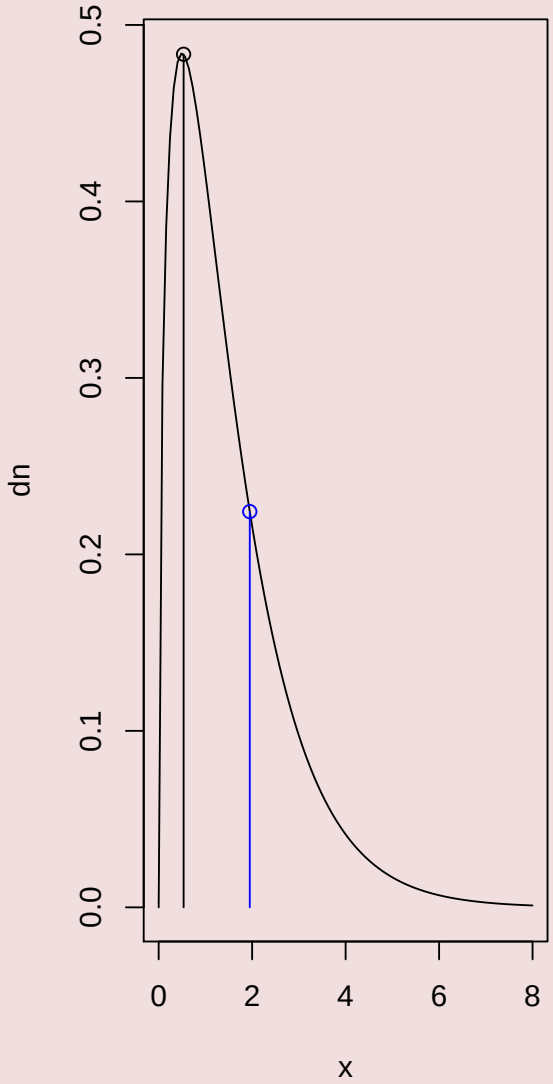

histogram of MCMC samples
n=3
Frequency
0.0
0.2
0.4
0.6
0.8
1.0
0
2
4
6
8
x_vec

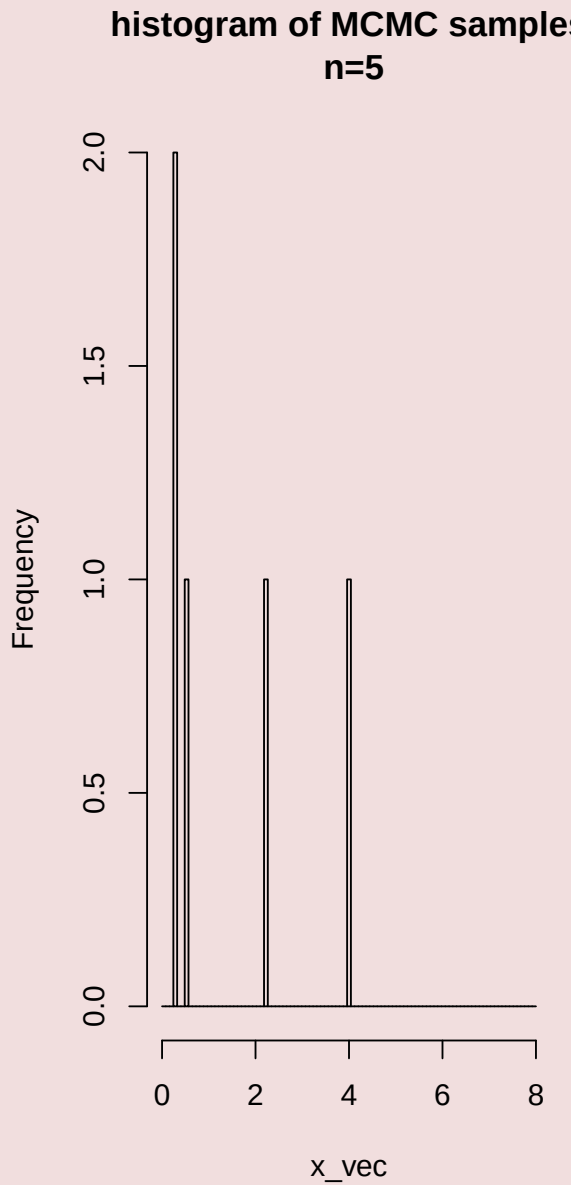
In Action

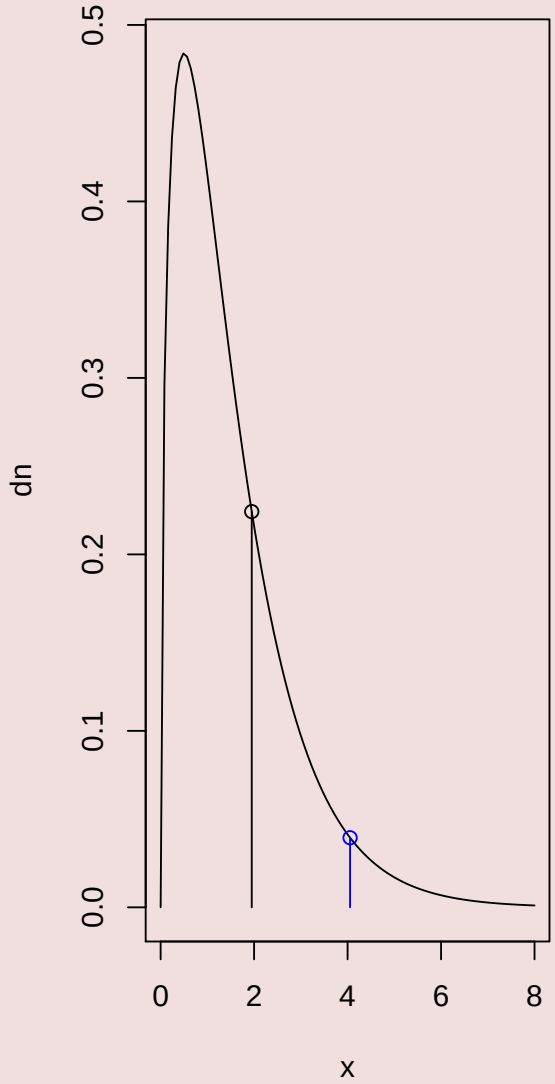

dn
0.0
0.1
0.2
0.3
0.4
0.5
0
2
4
6
8
x

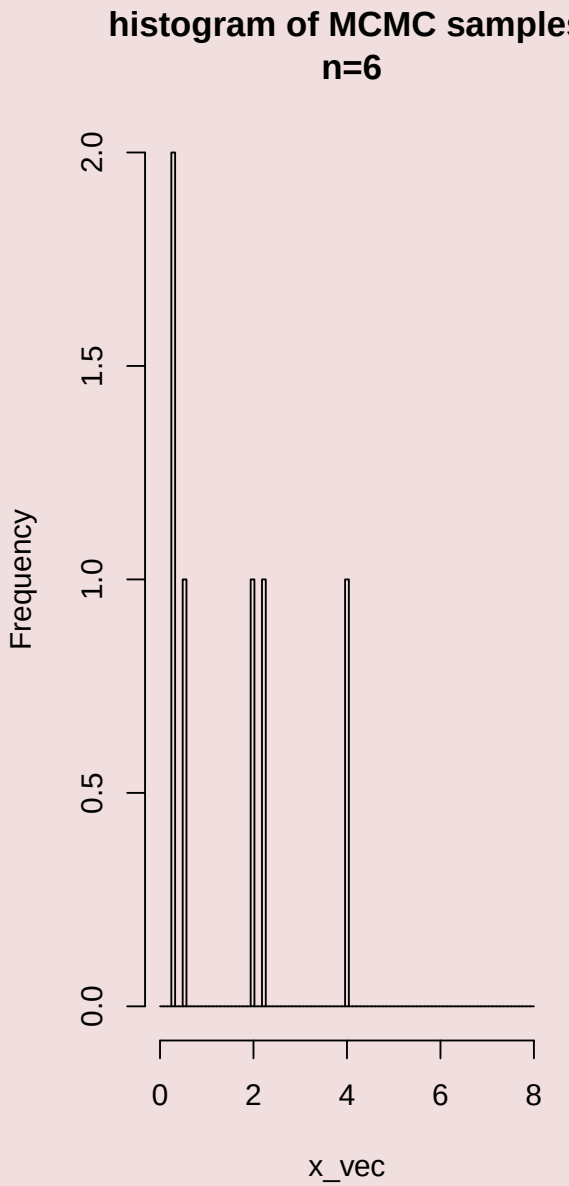

histogram of MCMC samples
n=4
Frequency
0.0
0.5
1.0
1.5
2.0
0
2
4
6
8
x_vec

In Action

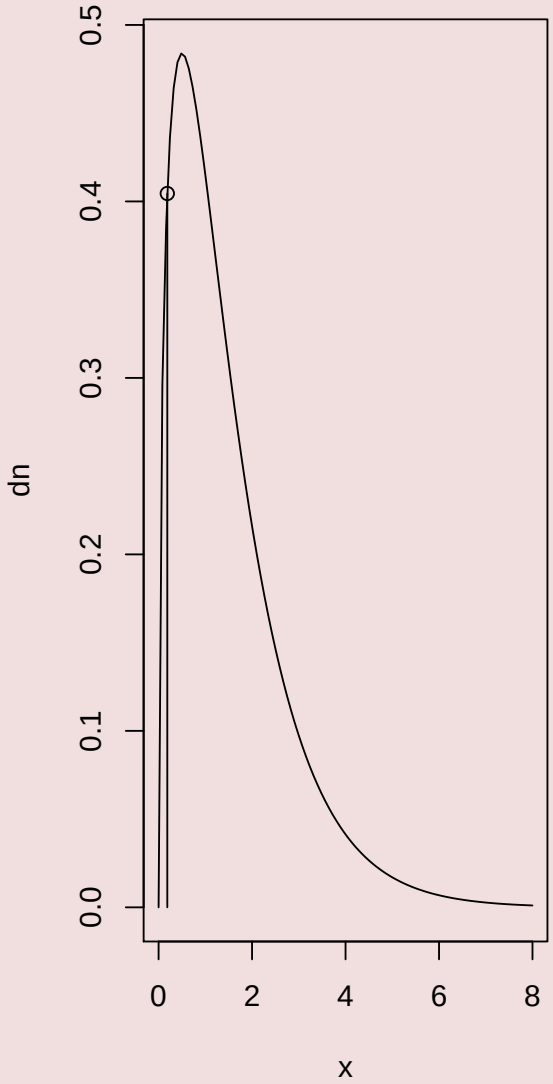

dn
0.0
0.1
0.2
0.3
0.4
0.5
0
2
4
6
8
x

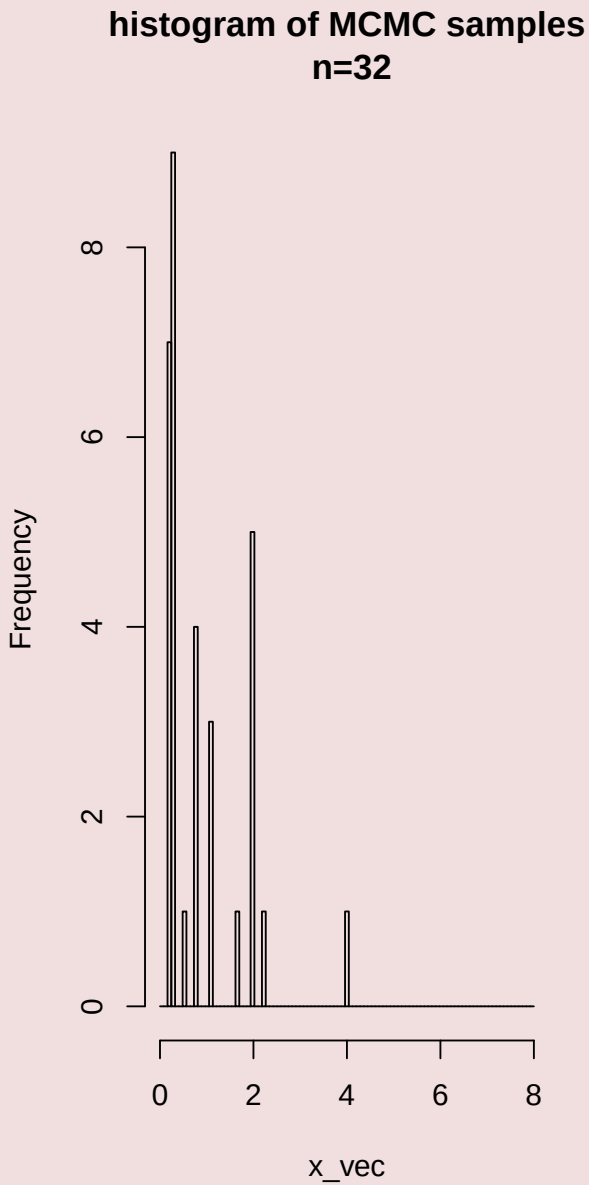

histogram of MCMC samples
n=5
Frequency
0.0
0.5
1.0
1.5
2.0
0
2
4
6
8
x_vec

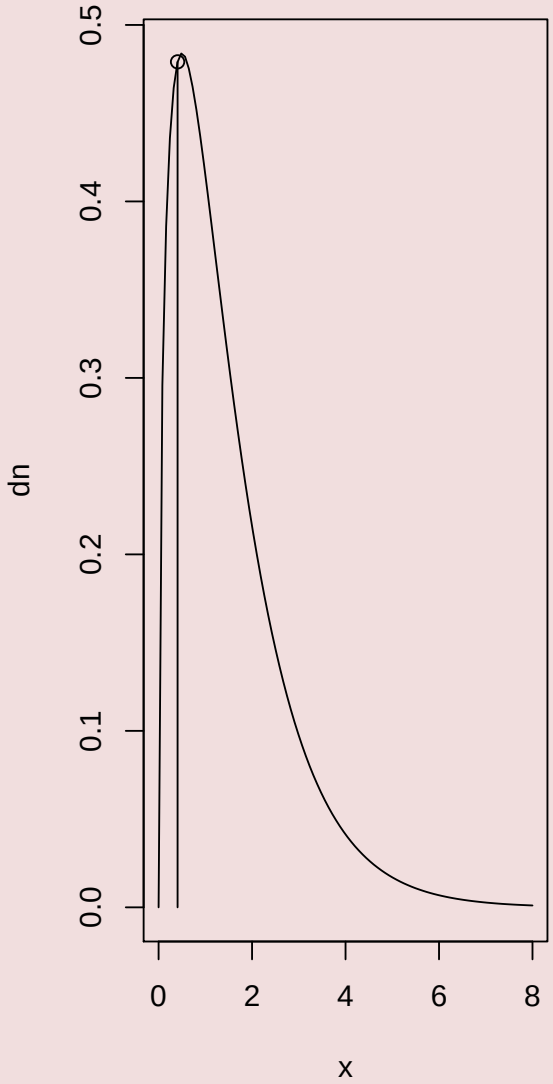

dn
0.0
0.1
0.2
0.3
0.4
0.5
0
2
4
6
8
x

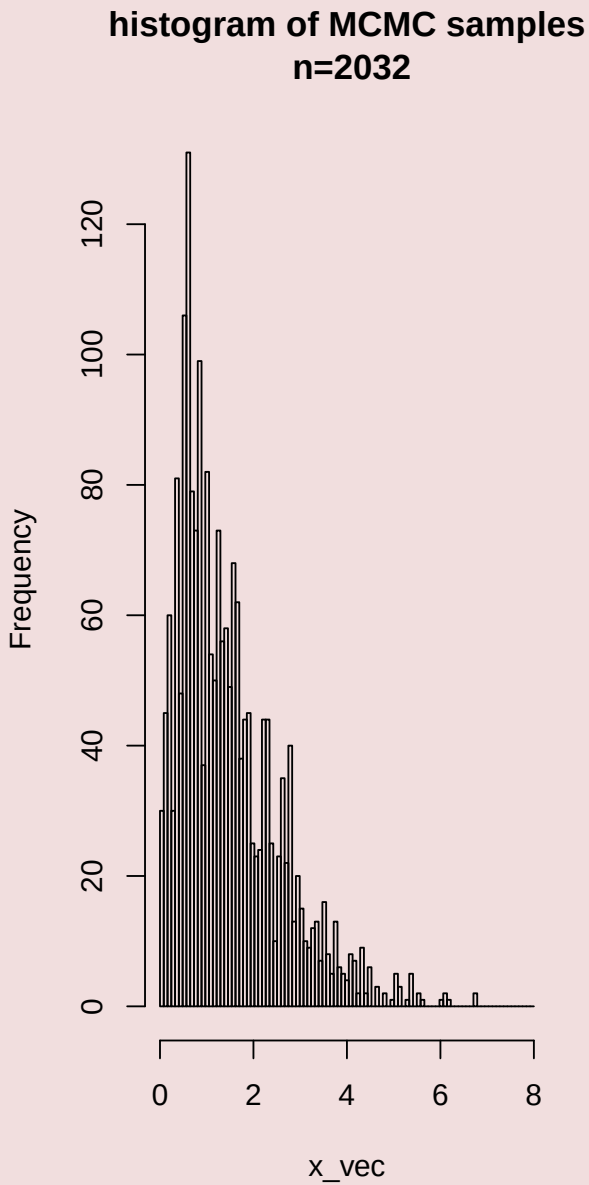

histogram of MCMC samples
n=6
Frequency
0.0
0.5
1.0
1.5
2.0
0
2
4
6
8
x_vec

In Action

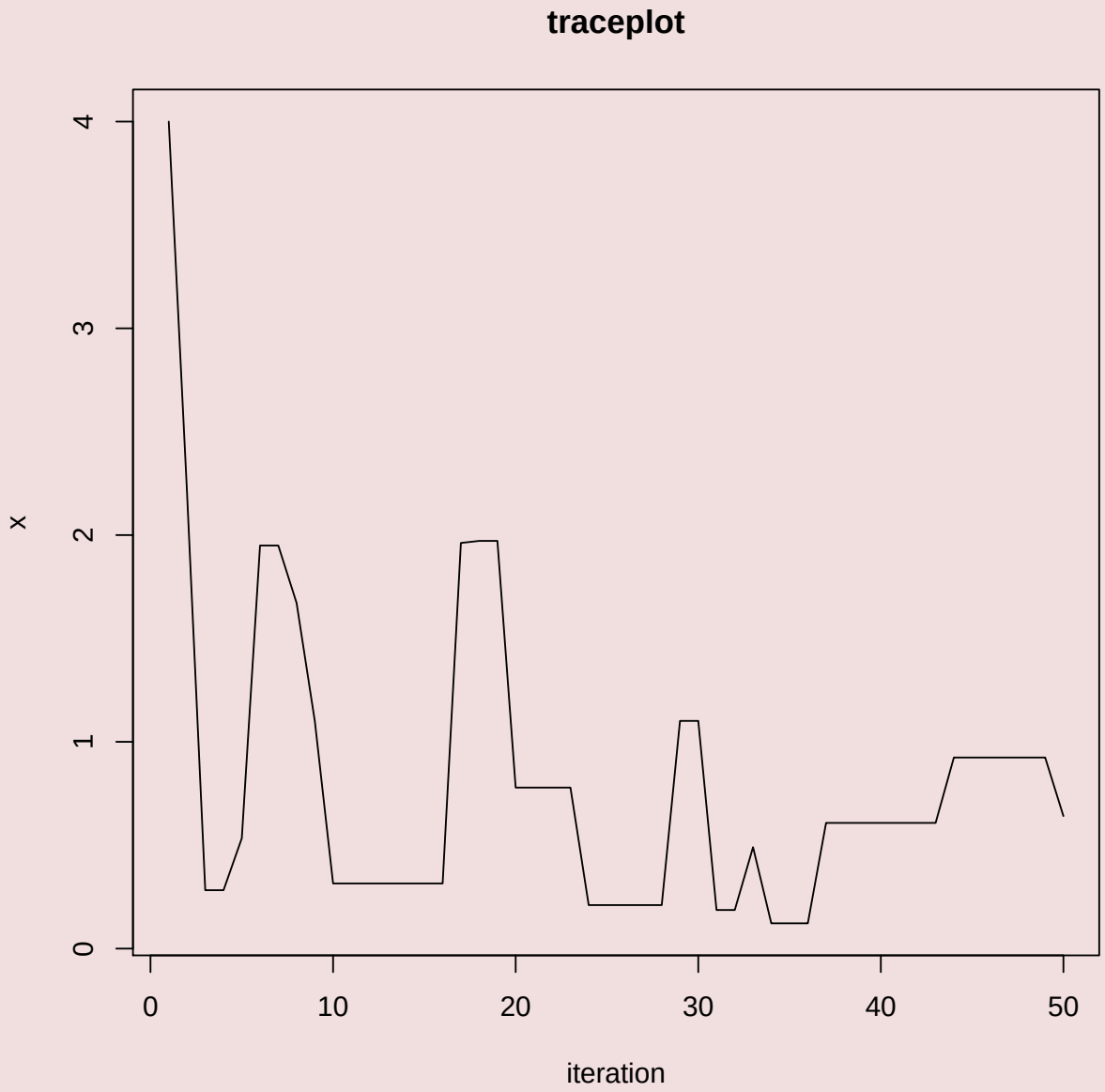

dn
0.0
0.1
0.2
0.3
0.4
0.5
0
2
4
6
8
x


histogram of MCMC samples
n=32
Frequency
0
2
4
6
8
0
2
4
6
8
x_vec

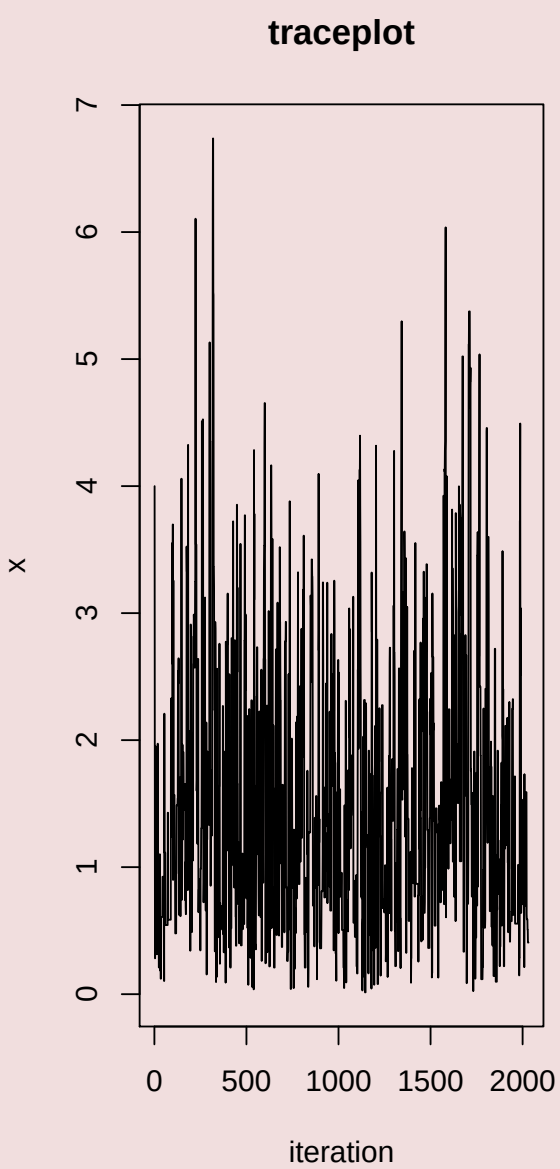

dn
0.0
0.1
0.2
0.3
0.4
0.5
0
2
4
6
8
x

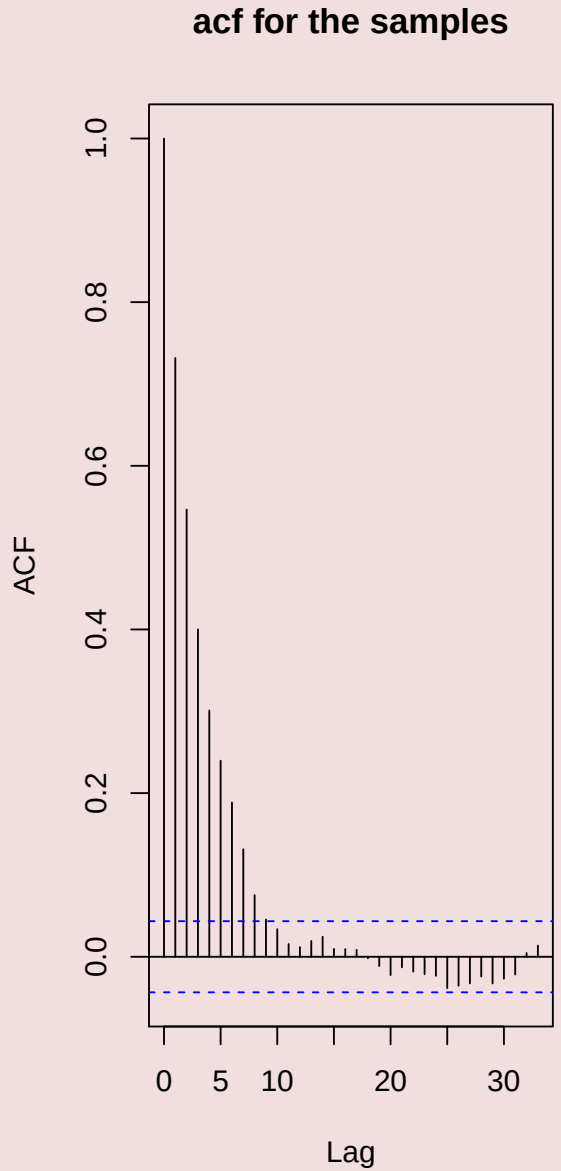

histogram of MCMC samples
n=2032
Frequency
0
20
40
60
80
100
120
0
2
4
6
8
x_vec

Traceplot



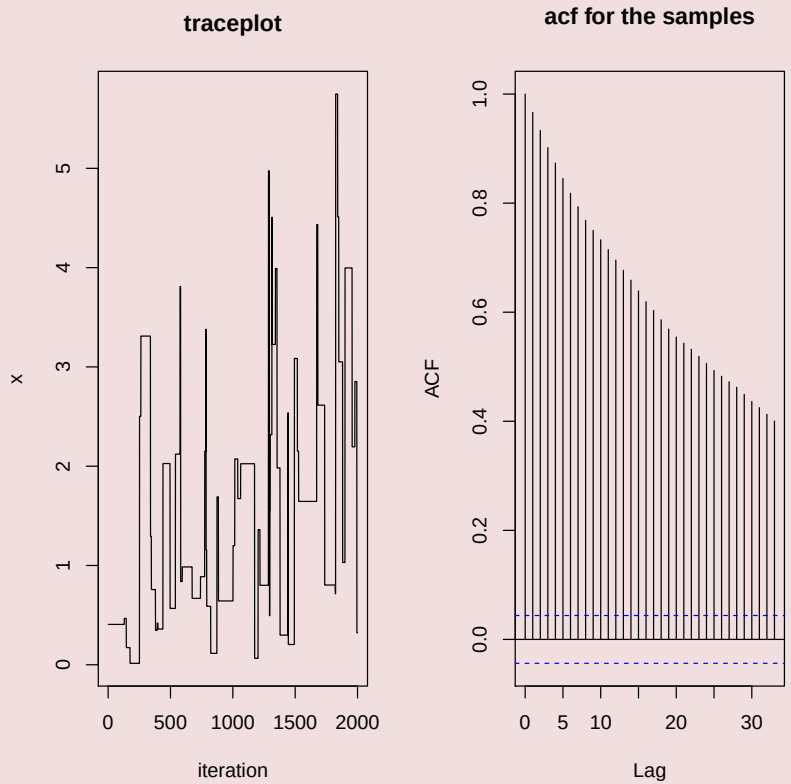
Traceplot


traceplot
x
0
1
2
3
4
5
6
7
0
500
1000
1500
2000
iteration


acf for the samples
ACF
0.0
0.2
0.4
0.6
0.8
1.0
0
5
10
20
30
Lag

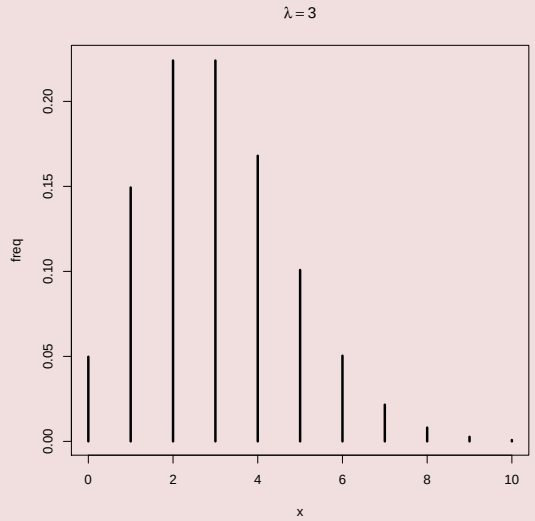
The choice of σ

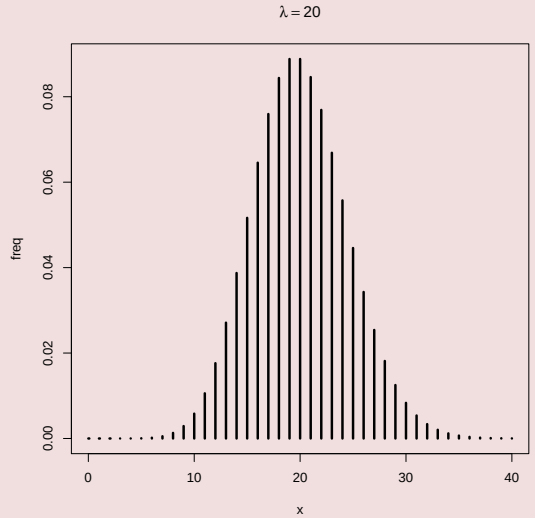
- The choice of σ extremely important for the mixing (the ACF) of the algorithm.
- In stan there are several parameters that are set in the MCMC this is done during the **warmup**.



Poisson regression

$$y \sim Po(\lambda),$$
$$\mathbb{E}[y] = \lambda,$$
$$\mathbb{V}[y] = \lambda.$$





- Counts (non negative integers) without upper limit
- One parameter, λ.
- Variance equal to mean.
- For large λ close to normal.

data sets

	culture	population	contact	total_tools
1	Malekula	1100	low	13
2	Tikopia	1500	low	22
3	Santa Cruz	3600	low	24
4	Yap	4791	high	43
5	Lau Fiji	7400	high	33
6	Trobriand	8000	high	19
7	Chuuk	9200	high	40
8	Manus	13000	low	28
9	Tonga	17500	high	55
10	Hawaii	275000	low	71

Poisson model

$$
\begin{aligned}
tools_i &\sim Po(\lambda_i) \\
g(\lambda_i) &= \alpha + \log(population_i)\beta_p + contact_i\beta_c \\
\alpha &\sim N(0, 10) \\
\beta_p &\sim N(0, 10) \\
\beta_c &\sim N(0, 10)
\end{aligned}
$$

- Density:

$$p(\alpha, \beta_c, \beta_p) \propto N(\alpha; 0, 10) N(\beta_c; 0, 10) N(\beta_p; 0, 10) \cdot \prod_{i=1}^{n} Po(t_i; g^{-1}(\alpha + \log(p_i)\beta_p + c_i\beta_c)).$$

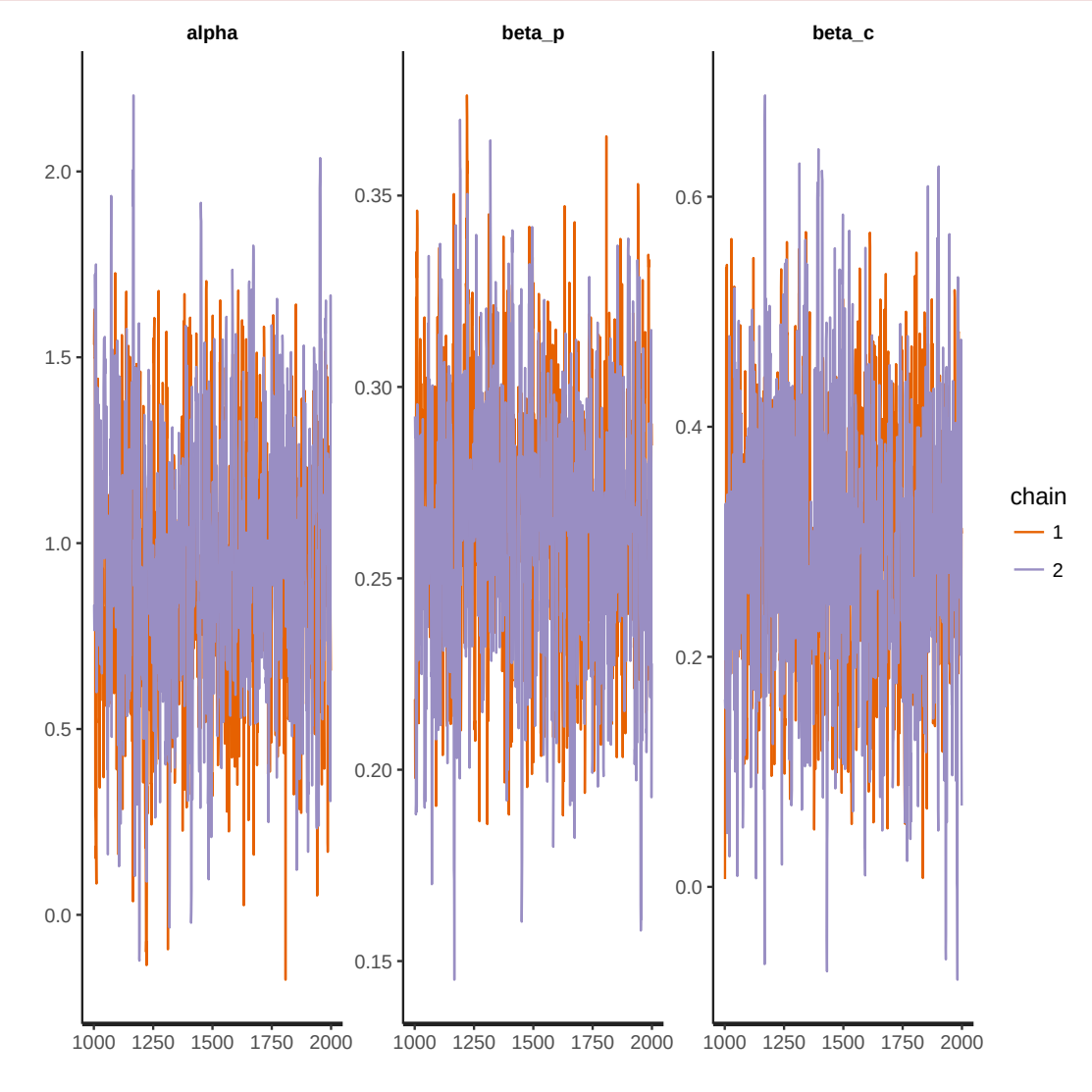
Hence,

$$\begin{aligned} p(\alpha, \beta_c, \beta_p) = & N(\alpha; 0, 10) N(\beta_c; 0, 10) N(\beta_p; 0, 10) \cdot \\ & \prod_{i=1}^{n} Po(t_i; g^{-1}(\alpha + \log(p_i)\beta_p + c_i\beta_c)) \cdot \\ & \left(\int N(\tilde{\alpha}; 0, 10) N(\tilde{\beta}_c; 0, 10) N(\tilde{\beta}_p; 0, 10) \cdot \right. \\ & \left. \prod_{i=1}^{n} Po(t_i; g^{-1}(\tilde{\alpha} + \log(p_i)\tilde{\beta}_p + c_i\tilde{\beta}_c)) d\tilde{\alpha} d\tilde{\beta}_p d\tilde{\beta}_c \right)^{-1} \end{aligned}$$

Markov Chain Monte Carlo vs Monte Carlo

- Direct sampling (Monte Carlo) will requires evaluation of $p(\alpha, \beta_c, \beta_p)$.
- What does MCMC

Checking the chains



Checking the chains

```
print(simple_fit, probs=c(0.1,0.9), digits=2,pars=c("alpha","beta_p","beta_c"))
```

```
Inference for Stan model: poisson_stan.
2 chains, each with iter=2000; warmup=1000; thin=1;
post-warmup draws per chain=1000, total post-warmup
draws=2000.

         mean se_mean   sd  10%  90% n_eff Rhat
alpha    0.94    0.01 0.35 0.49 1.40   627    1
beta_p   0.26    0.00 0.03 0.22 0.31   655    1
beta_c   0.30    0.00 0.12 0.15 0.45   797    1
```

Let X_{ij} be samples $i = 1, 2, \ldots, n$ and chains $j = 1, 2, \ldots, m$.

$$n_{eff} = \frac{n}{1 + \sum_{i=1}^{\infty} acf(i)}$$

Within Chain variance

Let X_{ij} be samples $i = 1, 2, \ldots, n$ and chains $j = 1, 2, \ldots, m$.

$$W = \frac{1}{m} \sum_{i=1}^{m} S_i^2,$$

where

$$s_j^2 = \frac{1}{n-1} \sum_{i=1}^{n} (X_{ij} - \bar{X}_j)^2.$$

Between Chain variance

Let X_{ij} be samples $i = 1, 2, \ldots, n$ and chains $j = 1, 2, \ldots, m$.

$$B = \frac{n}{m-1} \sum_{i=1}^{m} (\bar{\bar{X}} - \bar{X}_j)^2$$

where

$$\bar{\bar{X}} = \frac{1}{m} \sum_{j=1}^{m} \bar{X}_j.$$

Between Chain variance

Let X_{ij} be samples $i = 1, 2, \ldots, n$ and chains $j = 1, 2, \ldots, m$.

$$\hat{\mathbb{V}}[X] = (1 - \frac{1}{n})W + \frac{1}{n}B.$$

where

$$\hat{R} = \sqrt{\frac{\hat{\mathbb{V}}[X]}{W}}$$

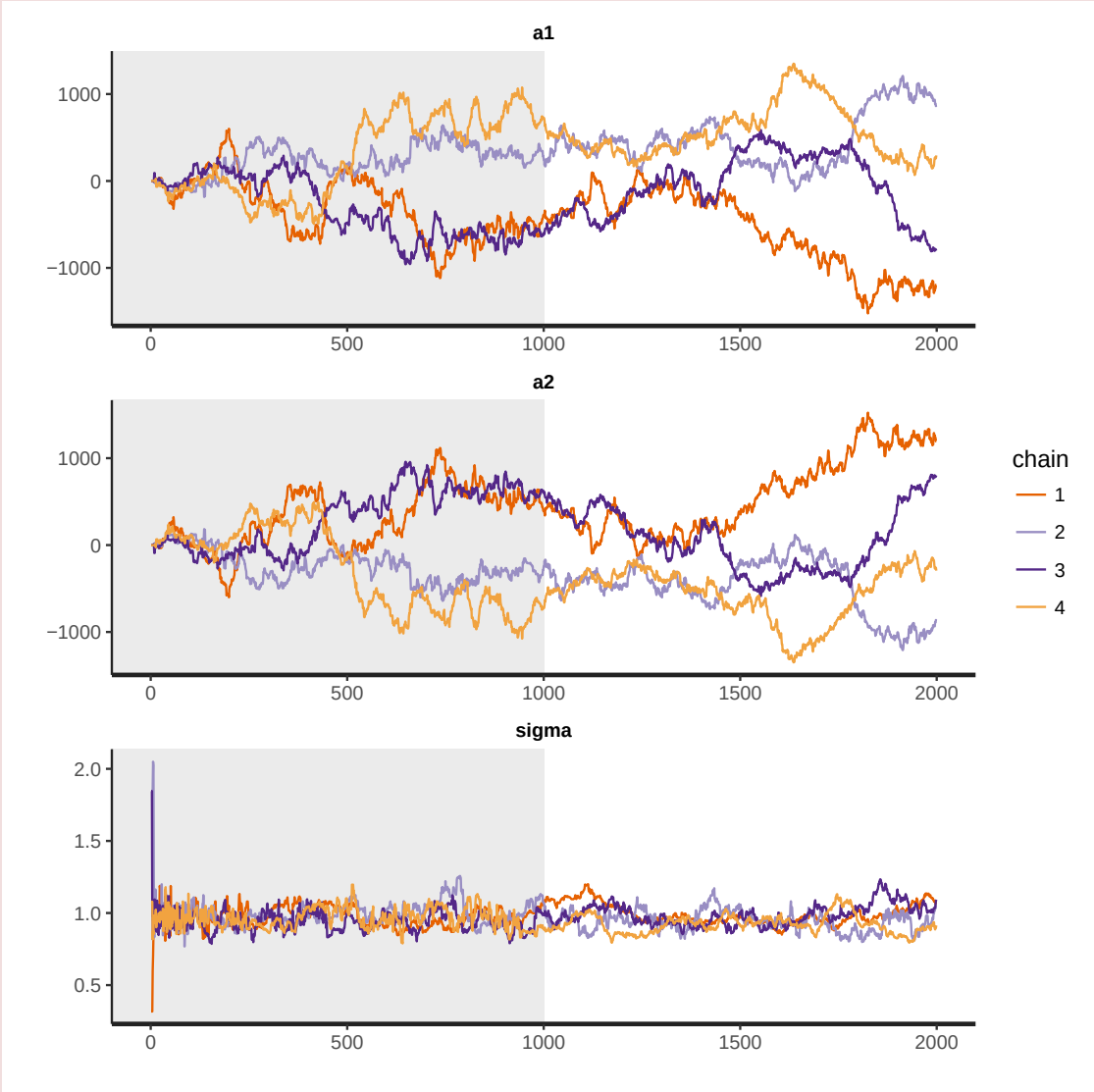
when $\hat{R}$ is high above 1.1, then it indicates that the chain does not have the same distribution.

Detour, dumb model

$$y_i \sim N(\mu, \sigma)$$
$$\mu = \alpha_1 + \alpha_2$$

Non proper prior $p(\alpha_1, \alpha_2, \sigma) \propto 1$.

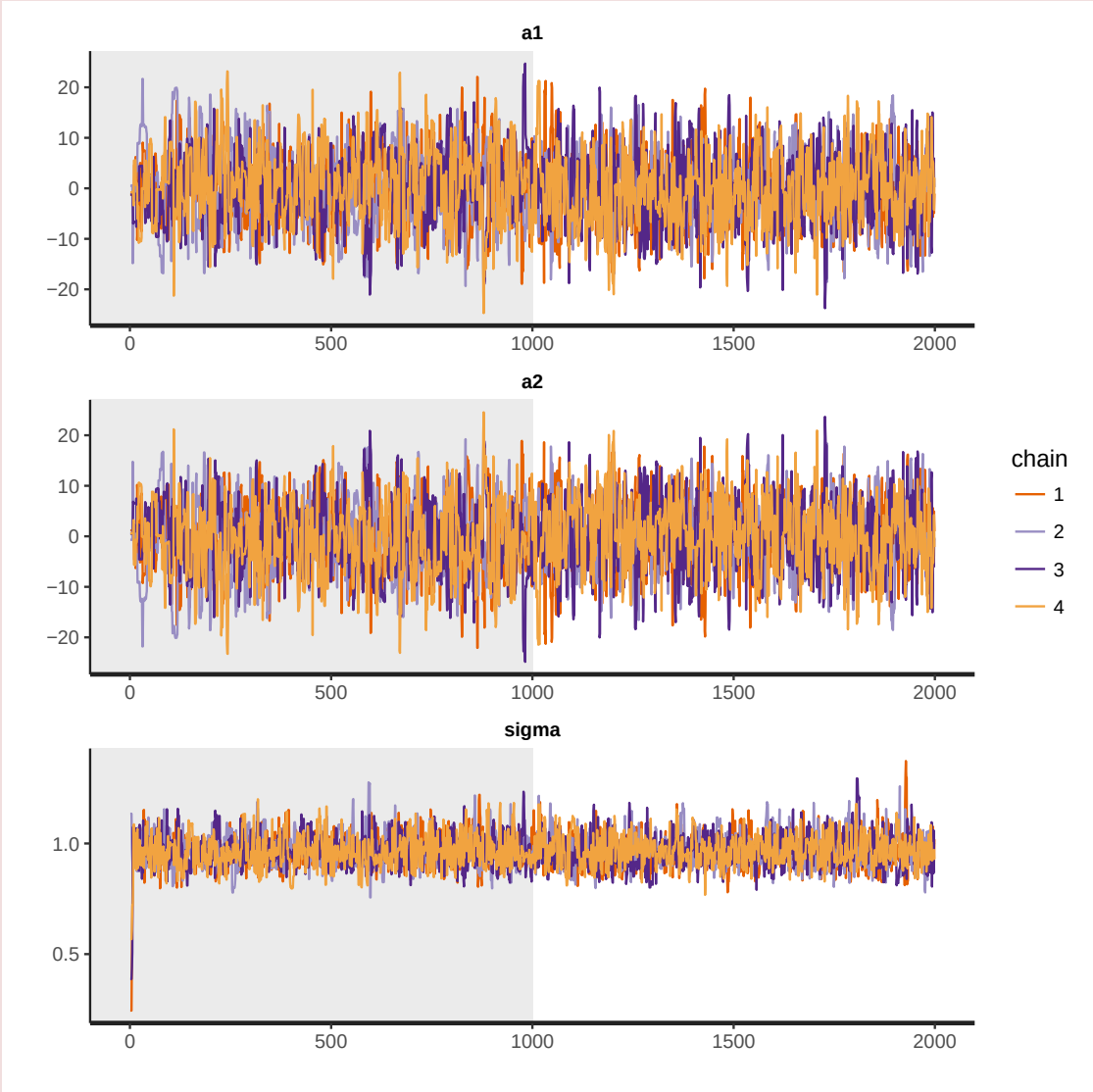
Detour, how does failure look

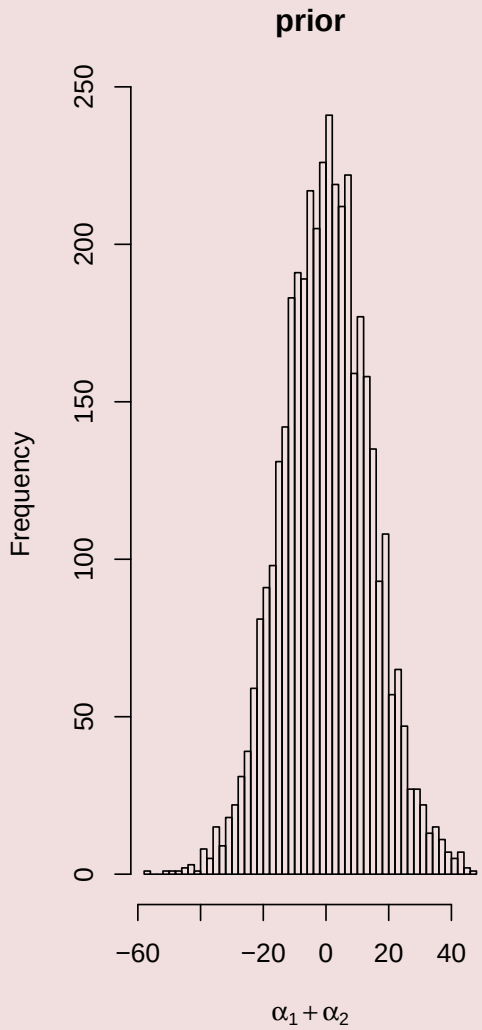


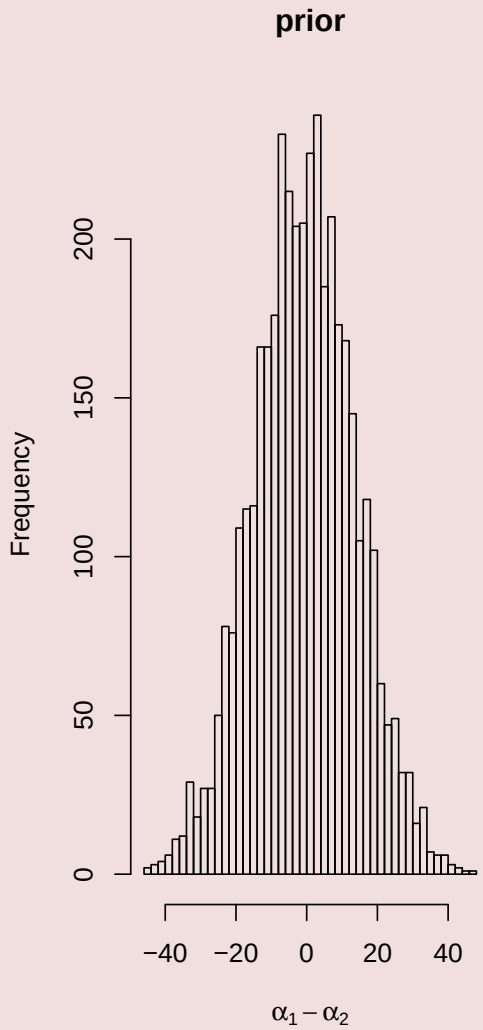
Detour, dumb model, dumb fix

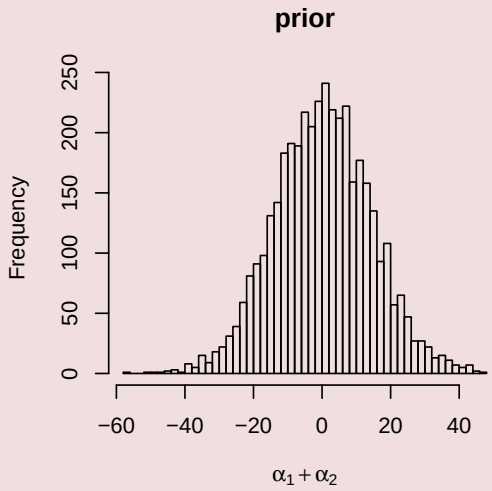
$$
\begin{aligned}
y_i &\sim N(\mu, \sigma) \\
\mu &= \alpha_1 + \alpha_2 \\
\alpha_1 &\sim N(0, 10) \\
\alpha_2 &\sim N(0, 10)
\end{aligned}
$$

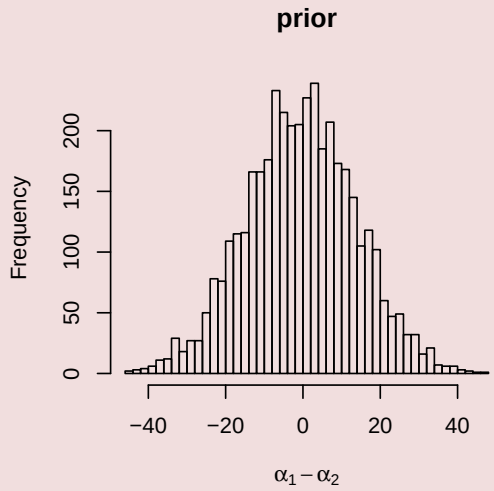
trace post fix

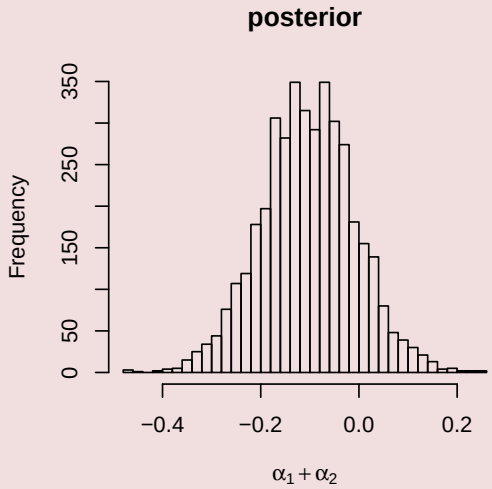


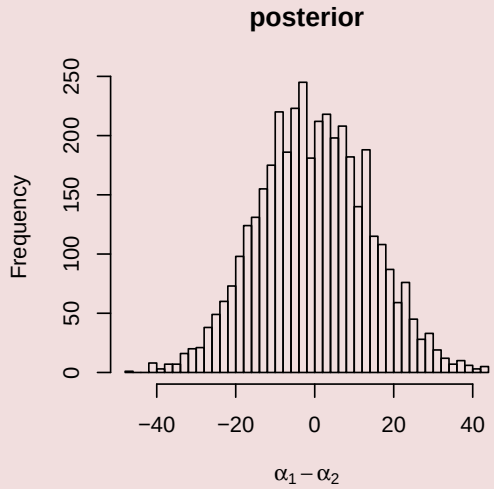

prior
Frequency
0
50
100
150
200
250
−60
−20
0
20
40
$\alpha_1 + \alpha_2$


prior
Frequency
0
50
100
150
200
−40
−20
0
20
40
$\alpha_1 - \alpha_2$

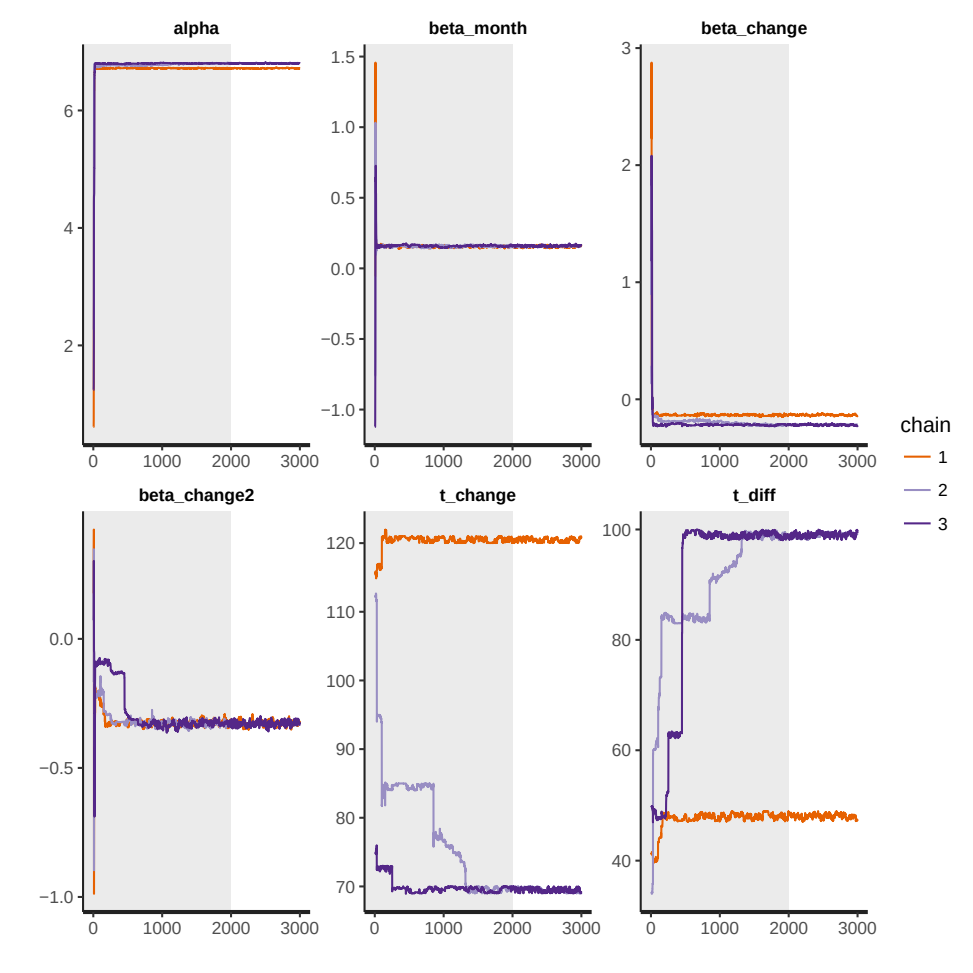

prior
Frequency
0
50
100
150
200
250
−60
−40
−20
0
20
40
$\alpha_1 + \alpha_2$


prior
Frequency
0
50
100
150
200
−40
−20
0
20
40
$\alpha_1 - \alpha_2$

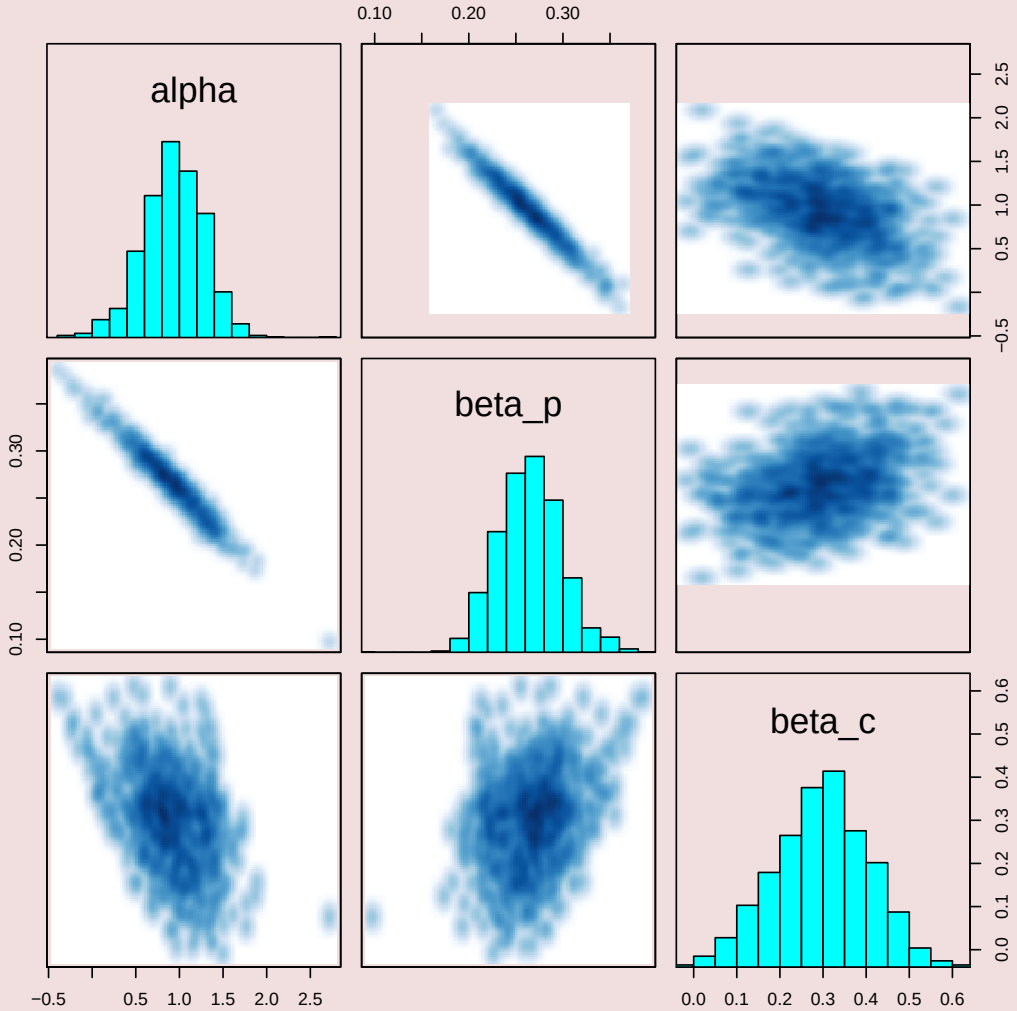

posterior
Frequency
0
50
150
250
350
−0.4
−0.2
0.0
0.2
$\alpha_1 + \alpha_2$


posterior
Frequency
0
50
100
150
200
250
−40
−20
0
20
40
$\alpha_1 - \alpha_2$

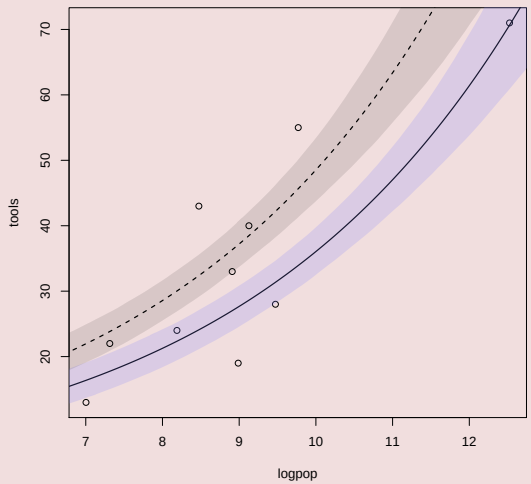
Checking the chains, failure number 2



Back to: Poisson model, posterior fit



Posterior fit



Fitted simple_fit through stan then:

```
log_pop.seq <- seq(from = 6, to = 13, length.out=100)
samples <- extract(simple_fit)
lambda_mat <- sapply(1:length(samples$alpha), function(i){

return(exp(samples$alpha[i] + log_pop.seq * samples$beta_p[i]))
})
lambda_mat_c <- sapply(1:length(samples$alpha), function(i){

return(exp(samples$alpha[i] + log_pop.seq * samples$beta_p[i] +
samples$beta_c[i]))
})
```

Complete code in Rmarkwdown on homepage later this week.

Use the right scale