元器件生存时间模拟分析

Miao Cai

2019-04-09

目录

1 模拟设置 **2**

2 贝叶斯推断 **2**

3 马尔科夫链蒙特卡洛模拟 **2**

4 后验分布收敛性检验 **2**

5 统计软件 **2**

6 先验分布 **3**

7 后验分布 **4**

7.1 诊断 5

附录: R及Stan代码 **7**

参考文献 **14**

1 模拟设置

根据你写的内容，介绍模拟的背景，风险1和2，温度设置，检测时间以及样本量的设置等

2 贝叶斯推断

$$P(\theta|D) = \frac{P(\theta)P(D|\theta)}{\int P(\theta)P(D|\theta)}$$

自己根据先前的文献写500字左右的贝叶斯推断的介绍，1面左右。

3 马尔科夫链蒙特卡洛模拟

自己根据先前的文献写500字左右的马尔科夫链蒙特卡洛模拟的介绍，1面左右。

4 后验分布收敛性检验

由于本研究中贝叶斯推断采用马尔科夫链蒙特卡洛模拟抽样来逼近参数的后验分布，因此贝叶斯推断的有效性取决于模拟抽样是否达到了参数的稳定的后验分布。只有模拟抽样的后验分布达到稳定时，我们得到的参数估计才是有效的。在此研究中，我们采用三种办法来检验后验分布是否达到稳定状态：各个参数的有效抽样次数，Gelman-Rubin统计量和轨迹图。各种办法的检验标准如下：

- 有效抽样次数：当各个参数的有效抽样次数均大于100时，提示马尔科夫链蒙特卡洛模拟得到的参数后验分布达到了稳定状态，
- Gelman-Rubin统计量$\hat{R}$：当各个参数的Gelman-Rubin统计量均小于1.1时，提示马尔科夫链蒙特卡洛模拟得到的参数后验分布达到了稳定状态，
- 轨迹图：当各个参数的轨迹图没有明显的序列相关，在真实后验分布上下不规则震荡，并且整个图呈毛毛虫状时，提示马尔科夫链蒙特卡洛模拟得到的参数后验分布达到了稳定状态。

5 统计软件

本次模拟主要运用统计计算环境`R`（3.5.3版本）软件来实现[1]，其中贝叶斯建模与推断运用R软件包`rstan`（2.18.2版本）来实现[2]。与其他的贝叶斯推断软件`WinBUGS`, `OpenBUGS`以及`JAGS`相比，Stan具有更灵活的因变量分布，更方便的自定义函数，更加活跃的用户群体论坛，以及依托于汉密尔顿马尔科夫链蒙特卡洛模拟的在大数据和多维空间情况下的更加高效和快速的抽样算法[3]，因此适用于本模拟研究的统计推断。为了检验后验分布是否收敛，我们将抽样次数设置为2000，退火（热身）次数设置为1000，因此理论上最大的实际抽样次数为3000次，并且设置三条并行计算抽样的马尔科夫链，用以检验各个参数的后验分布是否收敛。

6 先验分布

对于本研究中的参数$\alpha_{10}, \alpha_{11}, \alpha_{20}, \alpha_{21}$，我们对其设置同样的弱信息量的伽马先验分布Gamma$(1, 10)$。选择此弱先验分布的原因是由我们的研究背景可以知道：

- 参数$\alpha_{10}, \alpha_{11}, \alpha_{20}, \alpha_{21}$均必须为正数，
- 参数$\alpha_{10}, \alpha_{11}, \alpha_{20}, \alpha_{21}$值均较小（一般小于1）。

在`rstan`中，伽马分布的参数化形式（即概率密度函数）如下所示：

$$\text{Gamma}(y|\alpha, \beta) = \frac{\beta^{\alpha}}{\Gamma(\alpha)} y^{\alpha-1} \exp(-\beta y)$$

伽马分布包括两个参数α与β，其中α为形状参数，β为率参数。该参数化形势下的伽马分布的均值为α/β，方差为α/β^2。该先验分布的概率密度图如下所示：

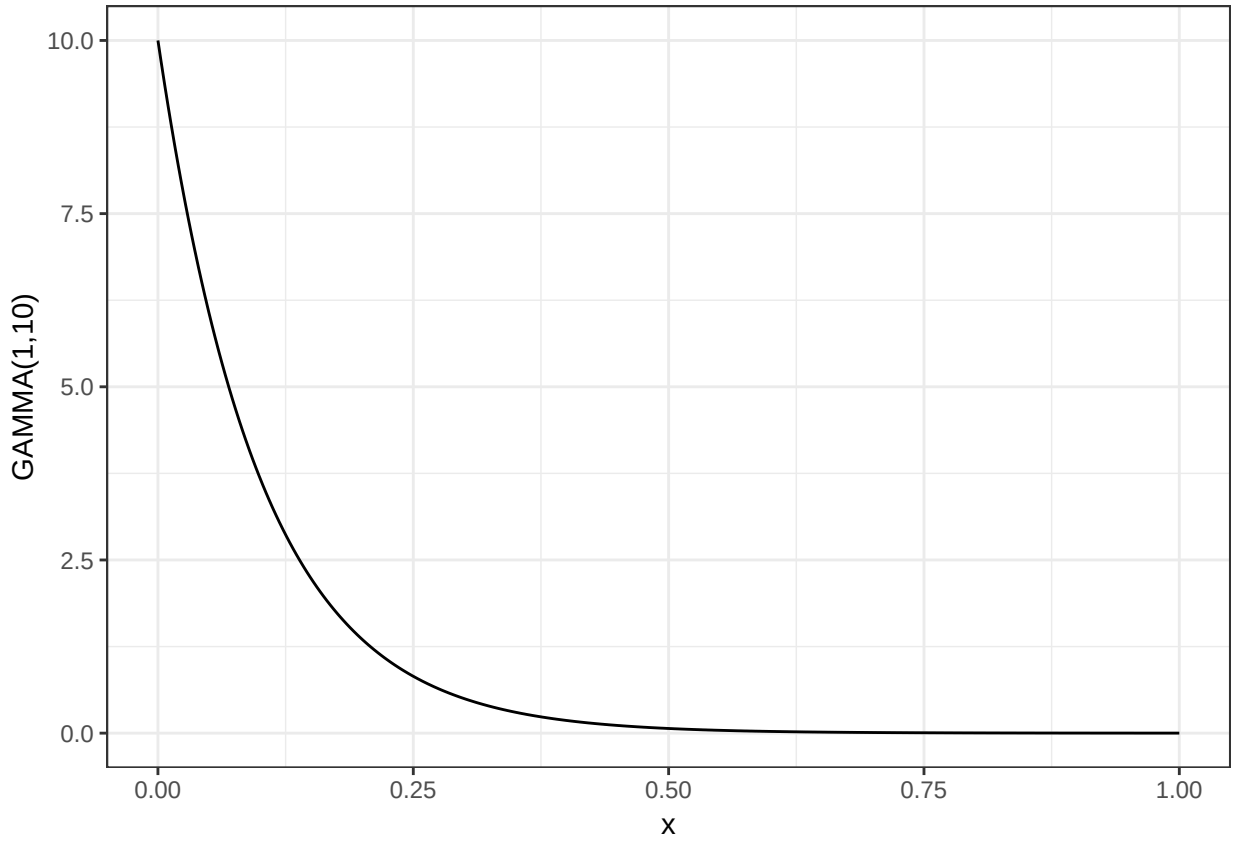

图 1: Gamma(1, 10)的先验分布

由该图我们可以看出，四个不同的α参数的先验分布的值域实际可处于$[0, +\infty]$之间，但是大部分概率密度均处于0-0.5之间，符合我们对此元器件故障参数的背景知识，并且此伽马分布也能够覆盖到所有参数α的可能的取值，因此是合理的弱先验分布。

7 后验分布

表 1: n=10

X1	mean	se_mean	sd	2.5%	25%	50%	75%	97.5%	n_eff	Rhat
a10	0.0062	0.0002	0.0040	0.0002	0.0030	0.0058	0.0088	0.0154	590.4234	1.0081
a11	0.0211	0.0019	0.0229	0.0006	0.0057	0.0140	0.0280	0.0903	140.2150	1.0252
a20	0.0034	0.0001	0.0024	0.0002	0.0015	0.0030	0.0048	0.0091	888.5289	1.0020
a21	0.0166	0.0007	0.0165	0.0003	0.0048	0.0115	0.0235	0.0615	491.1611	1.0050
p0	0.8333	0.0007	0.0364	0.7556	0.8088	0.8353	0.8596	0.8965	2907.5946	1.0020
p1	0.1077	0.0006	0.0303	0.0561	0.0855	0.1051	0.1274	0.1726	2758.8568	1.0021
p2	0.0590	0.0004	0.0223	0.0230	0.0430	0.0563	0.0723	0.1092	3084.9396	1.0000

表 2: n=50

X1	mean	se_mean	sd	2.5%	25%	50%	75%	97.5%	n_eff	Rhat
a10	0.0050	0.0001	0.0027	0.0003	0.0028	0.0050	0.0074	0.0096	546.8515	1.0040
a11	0.0198	0.0013	0.0195	0.0007	0.0058	0.0143	0.0271	0.0765	233.2281	1.0091
a20	0.0020	0.0000	0.0011	0.0001	0.0012	0.0020	0.0029	0.0039	579.0775	1.0026
a21	0.0148	0.0006	0.0145	0.0004	0.0042	0.0107	0.0205	0.0582	553.8225	1.0023
p0	0.8750	0.0001	0.0067	0.8615	0.8706	0.8750	0.8795	0.8884	3017.2135	0.9992
p1	0.0902	0.0001	0.0058	0.0790	0.0863	0.0900	0.0941	0.1020	3130.6166	0.9991
p2	0.0348	0.0001	0.0038	0.0278	0.0322	0.0347	0.0372	0.0426	3060.4882	0.9993

表 3: n=100

X1	mean	se_mean	sd	2.5%	25%	50%	75%	97.5%	n_eff	Rhat
a10	0.0056	0.0001	0.0026	0.0009	0.0035	0.0057	0.0080	0.0096	322.7226	1.0314
a11	0.0159	0.0007	0.0143	0.0004	0.0042	0.0116	0.0227	0.0520	479.2416	1.0244
a20	0.0020	0.0001	0.0011	0.0001	0.0011	0.0021	0.0029	0.0037	156.1490	1.0224
a21	0.0174	0.0027	0.0195	0.0005	0.0048	0.0105	0.0222	0.0766	53.4758	1.0519
p0	0.8740	0.0001	0.0034	0.8676	0.8717	0.8740	0.8764	0.8804	2304.3549	1.0013
p1	0.0910	0.0001	0.0029	0.0854	0.0889	0.0910	0.0930	0.0965	2211.9462	1.0009
p2	0.0350	0.0000	0.0018	0.0315	0.0337	0.0349	0.0362	0.0386	2952.0432	1.0004

表 4: n=500

X1	mean	se_mean	sd	2.5%	25%	50%	75%	97.5%	n_eff	Rhat
a10	0.0053	1e-04	0.0025	0.0007	0.0032	0.0056	0.0073	0.0091	458.1226	1.0009
a11	0.0168	8e-04	0.0155	0.0007	0.0057	0.0113	0.0241	0.0592	357.6127	1.0017
a20	0.0019	0e+00	0.0010	0.0002	0.0011	0.0020	0.0028	0.0035	534.8533	1.0034
a21	0.0154	8e-04	0.0152	0.0003	0.0047	0.0103	0.0211	0.0552	396.0187	1.0083
p0	0.8784	0e+00	0.0007	0.8771	0.8780	0.8784	0.8788	0.8797	2955.2101	0.9995
p1	0.0879	0e+00	0.0006	0.0868	0.0876	0.0879	0.0883	0.0891	3035.0492	0.9992
p2	0.0337	0e+00	0.0004	0.0330	0.0334	0.0337	0.0339	0.0344	2312.8737	0.9995

表 5: n=1000

X1	mean	se_mean	sd	2.5%	25%	50%	75%	97.5%	n_eff	Rhat
a10	0.0051	1e-04	0.0026	0.0006	0.0028	0.0052	0.0073	0.0093	476.6922	1.0173
a11	0.0184	7e-04	0.0166	0.0004	0.0057	0.0133	0.0268	0.0623	520.7142	1.0184
a20	0.0020	0e+00	0.0010	0.0001	0.0011	0.0021	0.0029	0.0035	498.8544	1.0102
a21	0.0159	8e-04	0.0164	0.0005	0.0042	0.0103	0.0217	0.0622	431.6192	1.0103
p0	0.8771	0e+00	0.0003	0.8765	0.8769	0.8771	0.8773	0.8778	2888.0558	1.0002
p1	0.0888	0e+00	0.0003	0.0882	0.0886	0.0888	0.0890	0.0894	2701.1201	1.0000
p2	0.0341	0e+00	0.0002	0.0337	0.0340	0.0341	0.0342	0.0345	2639.1983	0.9997

对以上每个表各个参数α与p的后验分布与真实值之间的关系进行分析，差距是多少，可能是是由什么造成的（主要是样本量以及先验分布的选择）。

7.1 诊断

我们通过三种不同的方法来诊断此研究中贝叶斯估计的参数值收敛情况：有效抽样次数，Gelman-Rubin统计量$\hat{R}$，以及参数抽样的轨迹图。

7.1.1 有效抽样次数

由参数后验分布的倒数第二列中的有效抽样次数（n_eff）我们可以发现：大部分参数的后验分布抽样次数都大于100，提示模型的马尔科夫链蒙特卡洛模拟达到了稳定的参数后验分布。我们也发现对于部分模拟中，参数有效抽样次数仅为50个上下（例如样本量n=100时，参数a10的有效抽样次数仅为42.4）。这是因为后验分布的模拟抽取时，抽样之间存在很强的序列相关性，这种情况在参数之间高度相关时存在。我们可以通过其他的统计量来验证稳定的后验分布的假设。

7.1.2 Gelman-Rubin统计量

由参数后验分布的最后一列中的Gelman-Rubin统计量（R_hat）我们可以发现，所有样本量下各个参数的Gelman-Rubin统计量（R_hat）均小于1.1，因此强烈提示马尔科夫链蒙特卡洛模拟得到的参数后验分布达到了稳定状态，`rstan`返回的结果可以用于分析。

7.1.3 轨迹图

从各个样本量下各个参数的轨迹图我们可以发现：尽管每个参数的三条马尔科夫链的初始值不同，但是他们最终都交汇到后验分布的均值处进行上下震荡，三条链充分混合，并且后验分布呈毛毛虫状，因此提示马尔科夫链蒙特卡洛模拟得到的参数后验分布达到了稳定状态。

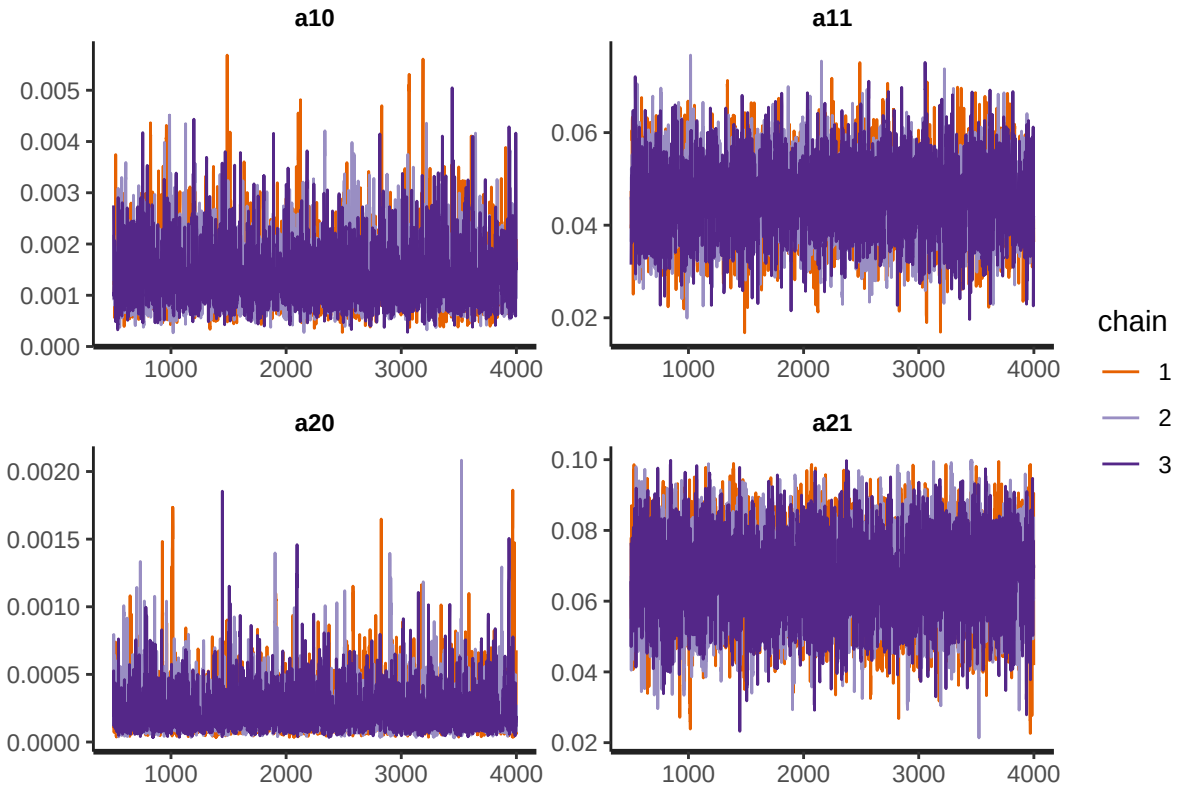

图 2: MCMC轨迹图示例

附录: R及Stan代码

1. 定义函数

```
set.seed(123)

alpha = matrix(c(0.001, 0.05, 0.0001, 0.08), ncol = 2, byrow = TRUE)
w = c(45, 55)
t = matrix(seq(10, 40, 10), ncol = 2, byrow = TRUE)
K = c(10, 50, 100)



lambda = function(r = 1:2, j = 1:2) {
  alpha_r0 = alpha[r, 1]
  alpha_r1 = alpha[r, 2]
  return(alpha_r0*exp(alpha_r1*w[j]))
}

p = function(i = 1:2, j = 1:2){
  lam1j = lambda(1, j) # failure due to factor 1
  lam2j = lambda(2, j) # failure due to factor 2

  p0 = exp(-(lam1j + lam2j)*t[i, j])
  p1 = (lam1j/(lam1j + lam2j))*(1 - p0)
  p2 = (lam2j/(lam1j + lam2j))*(1 - p0)

  return(c(p0, p1, p2))
}

SimMatrix = function(N = 10, N_device = 10){
  T1 = matrix(rep(-1, 3*N), ncol = 3, byrow = TRUE)
  T2 = T1; T3 = T1; T4 = T1;
  T1 = t(rmultinom(N, N_device, prob = p(1, 1)))
  for (i in 1:N) {
    T2[i,] = t(rmultinom(1, T1[i,1], prob = p(1, 2)))
  }
  for (i in 1:N) {
    T3[i,] = t(rmultinom(1, T2[i,1], prob = p(2, 1)))
  }
  for (i in 1:N) {
```

```
    T4[i,] = t(rmultinom(1, T3[i,1], prob = p(2, 2)))
  }
  return(list(T1, T2, T3, T4))
}

DAT0 = SimMatrix(100, 100)



create_standat = function(N = 10, N_device = 10){
  dat_list = SimMatrix(N, N_device)
  stan_dat = list(
    n = nrow(dat_list[[1]]),
    DAT1 = dat_list[[1]],
    DAT2 = dat_list[[2]],
    DAT3 = dat_list[[3]],
    DAT4 = dat_list[[4]],
    W = w,
    Tim = t
  )
  return(stan_dat)
}

post_result = function(N_sim = 10, N = 10, N_device = 10){
  tab_result = matrix(rep(NA, 12*N_sim), byrow = TRUE, ncol = 4)

  for (i in 1:N_sim) {
    stan_dat = create_standat(N, N_device)
    fit00 <- stan(
      model_code = reliabMAC, data = stan_dat,
      warmup = 500, iter = 1000, chains = 1, cores = 1, seed = i)
    tab_result[i,1:4] = summary(fit00)$summary[1:4,1]
    tab_result[i,1:4] = summary(fit00)$summary[1:4,1]
    tab_result[i,1:4] = summary(fit00)$summary[1:4,1]
  }

  return(tab_result)
}
```

2. **Stan代码**

```
library(rstan)
rstan_options(auto_write = TRUE)
options(scipen = 99)

reliabMAC = "
data {
  int n;
  int DAT1[n, 3];
  int DAT2[n, 3];
  int DAT3[n, 3];
  int DAT4[n, 3];
  vector[2] W;
  matrix[2, 2] Tim;
}
parameters{
  real<lower=0, upper=0.1> a10;
  real<lower=0, upper=0.1> a11;
  real<lower=0, upper=0.1> a20;
  real<lower=0, upper=0.1> a21;
}
transformed parameters{
  simplex[3] p11;
  simplex[3] p12;
  simplex[3] p21;
  simplex[3] p22;

  p11[1] = exp(-(a10*exp(a11*W[1]) + a20*exp(a21*W[1]))*Tim[1, 1]);
  p11[2]  = (a10*exp(a11*W[1]))/(a10*exp(a11*W[1]) + a20*exp(a21*W[1]))*(1 - p11[1]);
  p11[3]  = 1-p11[1]-p11[2];

  p12[1] = exp(-(a10*exp(a11*W[2]) + a20*exp(a21*W[2]))*Tim[1, 2]);
  p12[2]  = (a10*exp(a11*W[2]))/(a10*exp(a11*W[2]) + a20*exp(a21*W[2]))*(1 - p12[1]);
  p12[3]  = 1-p12[1]-p12[2];

  p21[1] = exp(-(a10*exp(a11*W[1]) + a20*exp(a21*W[1]))*Tim[2, 1]);
  p21[2]  = (a10*exp(a11*W[1]))/(a10*exp(a11*W[1]) + a20*exp(a21*W[1]))*(1 - p21[1]);
  p21[3]  = 1-p21[1]-p21[2];

  p22[1] = exp(-(a10*exp(a11*W[2]) + a20*exp(a21*W[2]))*Tim[2, 2]);
```

```
  p22[2]  = (a10*exp(a11*W[2]))/(a10*exp(a11*W[2]) + a20*exp(a21*W[2]))*(1 - p22[1]);
  p22[3]  = 1-p22[1]-p22[2];
  //p12 = [p0, p1, p2]';
  //p21 = [p0, p1, p2]';
  //p22 = [p0, p1, p2]';
}
model{
  for (i in 1:n){
    //target += multinomial_lpmf(DAT[i,] | p);
    DAT1[i,] ~ multinomial(p11);
    //DAT[i,] ~ multi_log(p0, p1, p2);
  }
  for (i in 1:n){
    DAT2[i,] ~ multinomial(p12);
  }
  for (i in 1:n){
    DAT3[i,] ~ multinomial(p21);
  }
  for (i in 1:n){
    DAT4[i,] ~ multinomial(p22);
  }
  a10 ~ uniform(0, 0.1);
  a11 ~ uniform(0, 0.1);
  a20 ~ uniform(0, 0.1);
  a21 ~ uniform(0, 0.1);
}
"
```

3. 正式模拟

- 模拟 *A1*

```
set.seed(123)
library(data.table)
alpha = matrix(c(0.001, 0.05, 0.0001, 0.08), ncol = 2, byrow = TRUE)

A1_10 = post_result(300, 1, 10)
A1_50 = post_result(300, 1, 50)
A1_100 = post_result(300, 1, 100)

data.table::fwrite(as.data.frame(A1_10), 'data/N = 10/A1_10.csv')
```

```
data.table::fwrite(as.data.frame(A1_50), 'data/N = 10/A1_50.csv')
data.table::fwrite(as.data.frame(A1_100), 'data/N = 10/A1_100.csv')
```

- 模拟 $A2$

```
set.seed(123)
alpha = matrix(c(0.005, 0.05, 0.0005, 0.08), ncol = 2, byrow = TRUE)

A2_10 = post_result(100, 10, 10)
A2_50 = post_result(100, 10, 50)
A2_100 = post_result(100, 10, 100)

data.table::fwrite(as.data.frame(A2_10), 'data/N = 10/A2_10.csv')
data.table::fwrite(as.data.frame(A2_50), 'data/N = 10/A2_50.csv')
data.table::fwrite(as.data.frame(A2_100), 'data/N = 10/A2_100.csv')
```

- 模拟 $A3$

```
set.seed(123)
alpha = matrix(c(0.008, 0.05, 0.0008, 0.08), ncol = 2, byrow = TRUE)

A3_10 = post_result(100, 10, 10)
A3_50 = post_result(100, 10, 50)
A3_100 = post_result(100, 10, 100)

data.table::fwrite(as.data.frame(A3_10), 'data/N = 10/A3_10.csv')
data.table::fwrite(as.data.frame(A3_50), 'data/N = 10/A3_50.csv')
data.table::fwrite(as.data.frame(A3_100), 'data/N = 10/A3_100.csv')
```

4. 生成结果报告

```
pacman::p_load(data.table, tidyverse)



report_tab = function(dat, alpha){
  dat_result = rbindlist(list(dat[,c(1, 5, 9)],
                 dat[,c(2, 6, 10)],
                 dat[,c(3, 7, 11)],
                 dat[,c(4, 8, 12)]))
  dat_result[, alpha := c(replicate(100, 'a01'),
                      replicate(100, 'a11'),
                      replicate(100, 'a20'),
```

```
                      replicate(100, 'a21'))]
  dat_result[, true_est := rep(alpha_3, each = 100)]
  names(dat_result) = c('estimate', 'n_eff', 'Rhat', 'alpha', 'true_est')

  final_dat = dat_result[,.(Bias = 1/100*sum(abs(estimate - true_est)),
            MSE = 1/100*sum((estimate - true_est)^2),
            n_eff = mean(n_eff),
            Rhat = mean(Rhat)), by = alpha]

  return(final_dat)
}

# A1
A1_10 = fread("data/20190409serve200/A1_10.csv")
A1_50 = fread("data/20190409serve200/A1_50.csv")
A1_100 = fread("data/20190409serve200/A1_100.csv")

report_tab(A1_10, c(0.001, 0.05, 0.0001, 0.08)) %>%
  fwrite('data/20190409serve200/reportA1_10.csv')
report_tab(A1_50, c(0.001, 0.05, 0.0001, 0.08)) %>%
  fwrite('data/20190409serve200/reportA1_50.csv')
report_tab(A1_100, c(0.001, 0.05, 0.0001, 0.08)) %>%
  fwrite('data/20190409serve200/reportA1_100.csv')

# A2
A2_10 = fread("data/20190409serve200/A2_10.csv")
A2_50 = fread("data/20190409serve200/A2_50.csv")
A2_100 = fread("data/20190409serve200/A2_100.csv")

report_tab(A2_10, c(0.005, 0.05, 0.0005, 0.08)) %>%
  fwrite('data/20190409serve200/reportA2_10.csv')
report_tab(A2_50, c(0.005, 0.05, 0.0005, 0.08)) %>%
  fwrite('data/20190409serve200/reportA2_50.csv')
report_tab(A2_100, c(0.005, 0.05, 0.0005, 0.08)) %>%
  fwrite('data/20190409serve200/reportA2_100.csv')

# A3
A3_10 = fread("data/20190409serve200/A3_10.csv")
A3_50 = fread("data/20190409serve200/A3_50.csv")
```

```
A3_100 = fread("data/20190409serve200/A3_100.csv")

report_tab(A3_10, c(0.008, 0.05, 0.0008, 0.08)) %>%
  fwrite('data/20190409serve200/reportA3_10.csv')
report_tab(A3_50, c(0.008, 0.05, 0.0008, 0.08)) %>%
  fwrite('data/20190409serve200/reportA3_50.csv')
report_tab(A3_100, c(0.008, 0.05, 0.0008, 0.08)) %>%
  fwrite('data/20190409serve200/reportA3_100.csv')
```

5. 轨迹图示例

```
require(rstan)

set.seed(123)
stan_dat = create_standat(10, 100)

fit00 <- stan(
  model_code = reliabMAC, data = stan_dat,
  warmup = 500, iter = 4000, chains = 3, cores = 3, seed = 1)
saveRDS(fit00, 'data/20190409serve200/fit00.Rds')

fit00 = readRDS('data/20190409serve200/fit00.Rds')
plot(fit00, plotfun = 'trace', pars = c('a10', 'a11', 'a20', 'a21'))
ggplot2::ggsave('data/20190409serve200/sample trace.png', dpi = 300, width = 10, height = 6.18)
```

参考文献

[1] R CORE TEAM. R: A language and environment for statistical computing[M]. Vienna, Austria: R Foundation for Statistical Computing, 2019.

[2] STAN DEVELOPMENT TEAM. RStan: The R interface to Stan[M]., 2018.

[3] GELMAN Andrew, LEE Daniel, GUO Jiqiang. Stan: A probabilistic programming language for bayesian inference and optimization[J]. Journal of Educational and Behavioral Statistics, 2015, 40(5): 530–543.