第六章 贝叶斯计算（软件运用）

Wang Shujia

Contents

本章学习目标

1. 能够建立贝叶斯模型及编写模型代码
2. 能够操作运用 WinBUGS/OpenBUGS/JAGS/Stan 等贝叶斯统计分析软件
3. 能够对 MCMC 抽样的收敛性进行诊断

1 贝叶斯软件简介

WinBUGS/OpenBUGS

- BUGS: **B**ayesian inference **U**sing **G**ibbs **S**ampling
- WinBUGS：最早（1997），最稳定
  - 作者:Spiegelhalter,Thomas,Best,Lunn, MRC Biostatistics Unit & Imperial College School of Medicine, London
  - 网站：http://www.mrc-bsu.cam.ac.uk/bugs/winbugs/
  - 特点：还用老掉牙的 Pascal，Windows 平台，不开源（虽然 0 费用）
  - 2007 年后没有维护
  - 可在 R 中运行：R2WinBUGS
- OpenBUGS
  - 作者：Andrew Thomas, University of Helsinki
  - 网站：http://www.openbugs.info/w/
  - 特点：开源，也是 Pascal，与 WinBUGS 差别不大
  - 有维护，可扩展，可以自己写函数、分布和抽样方法

JAGS

- JAGS: **J**ust **A**nother **G**ibbs **S**ampler，为了在 Unix 上运行 WinBUGS 而产生
  - 作者：Martyn Plummer (2003), International Agency for Research on Cancer
  - 网站：http://calvin.iarc.fr/~martyn/software/jags/
  - 特点：开源，C++，**跨平台**，语法与 BUGS 类似
  - 可扩展
  - 可在 R 中运行：R2jags

Stan

- Stan 名字为纪念蒙特卡罗先驱 Stanislaw Ulam (1909-1984)
  - 作者：Stan Development Team（2016）. Stan Modeling Language Users Guide and Reference Manual, Version 2.14.0.
  - 网站：www.mc-stan.org
  - 算法：基于 Hamiltonian Monte Carlo (HMC) 抽样 (非 MCMC)，叫 No U-Turn Sampler (NUTS)
  - 通过计算后验密度函数对数的梯度，加快收敛
  - 可以计算后验密度函数对数：$\log f(\theta|x)$
  - 特点：开源，C++，**跨平台**，对复杂模型或大数据集更有效率
  - 可在 R 中运行：RStan
  - 集成了大量应用软件包，可以在 R、Python、Matlab 等软件运行

2 WinBUGS 运用

例 1：汽车油耗问题

目的：研究汽车自重 x（吨）对油耗 y（加仑/英里）的影响。

	1	2	3	4	5	6	7	8	9	10
y	3.40	3.80	9.10	2.20	2.60	2.90	2.00	2.70	1.90	3.40
x	5.50	5.90	6.50	3.30	3.60	4.60	2.90	3.60	3.10	4.90

线性模型：

$$y_i = \alpha + \beta x_i + \varepsilon_i, (i = 1, 2, \ldots, n)$$

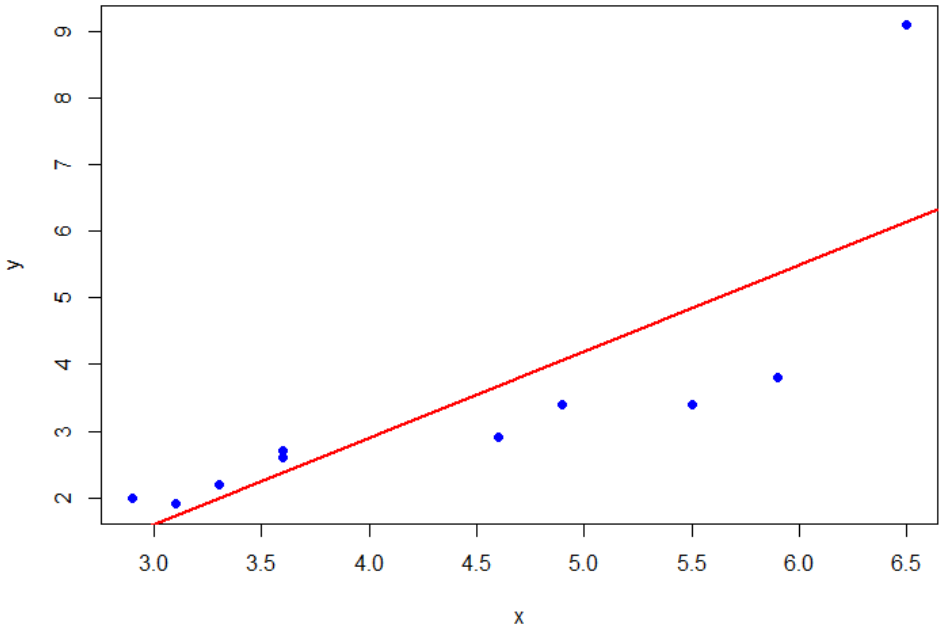
模型假设：

1. 线性性：y 与 x 是线性关系
2. 正态性：$\varepsilon_i \sim N(0, \sigma^2)$
3. 独立性：$\varepsilon_1, \varepsilon_2, \ldots, \varepsilon_n$ 相互独立
4. 方差齐性：各个 ε_i 的方差相同

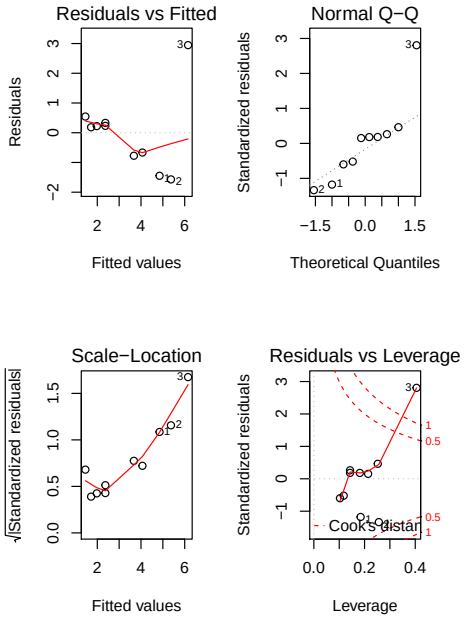
[fragile] 线性模型（传统）

```
> y=c(3.4,3.8,9.1,2.2,2.6,2.9,2.0,2.7,1.9,3.4)
> x=c(5.5,5.9,6.5,3.3,3.6,4.6,2.9,3.6,3.1,4.9)
> summary(lm(y~x))

Call:
lm(formula = y ~ x)

Residuals:
    Min      1Q  Median      3Q     Max
-1.5707 -0.7470  0.2031  0.3060  2.9463

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  -2.3294     1.6231  -1.435  0.18916
x             1.3051     0.3565   3.661  0.00639 **
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 1.362 on 8 degrees of freedom
Multiple R-squared:  0.6262,    Adjusted R-squared:  0.5795
F-statistic:  13.4 on 1 and 8 DF,  p-value: 0.006391
```

散点图：结果不太好



模型诊断图：存在奇异点



2.1 建立 BUGS 模型

贝叶斯数学模型及相应 BUGS 模型代码

贝叶斯模型 (无信息先验)：

$$
\begin{aligned}
y_i &\sim N(\alpha + \beta x_i, \sigma^2) \\
\alpha &\sim N(0, 1000) \\
\beta &\sim N(0, 1000) \\
\sigma^2 &\sim IG(1000, 1000)
\end{aligned}
$$

BUGS 代码：

```
model{
for(i in 1:n ){
  y[i]~dnorm(mu[i],tau)
  #tau 是精度，方差的倒数!
  mu[i]<-alpha+beta*x[i]
  }
alpha~dnorm(0,0.001)#must be proper
beta~dnorm(0,0.001)
tau~dgamma(0.001,0.001)#Is Gamma!
sigma<-1/sqrt(tau)
}
```

完整的 BUGS 代码

```
model{
  for(i in 1:n ){
    y[i]~dnorm(mu[i],tau)
    mu[i]<-alpha+beta*x[i]
    }
  alpha~dflat()
  beta~dflat()
  tau~dgamma(0.001,0.001)
  sigma<-1/sqrt(tau)
  }
#data
list(n=10,y=c(3.4,3.8,9.1,2.2,2.6,2.9,2.0,2.7,1.9,3.4),
x=c(5.5,5.9,6.5,3.3,3.6,4.6,2.9,3.6,3.1,4.9))
#initial values,
list(tau=5,alpha=4,beta=1.9) #GLS-2sd
list(tau=0.74,alpha=-2.3294,beta=1.3051)#GLS
list(tau=3,alpha=0.91,beta=3.32) #GLS+2sd
```

2.2 编译和运行 WinBUGS

WinBUGS 操作步骤

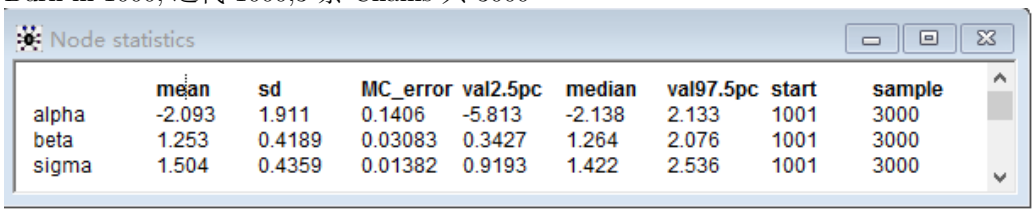
安装 WinBUGS 软件后，按照说明进行 decode。

1. 检查模型（Model ▷ Secification ▷ check model）
2. 载入数据（load data）

3. 编译模型（compile model）
4. 载入初始值（load initial values）
5. 产生 burn-in 样本（Model▷Update)
6. 指定待估参数（Inference▷Samples）
7. 抽取后验分布样本（Model▷Update)
8. 给出计算结果（Inference▷Summary）（node:*，表示所有参数）
9. 检查 MCMC 的有效性和收敛性

运行结果

Burn-in 1000, 迭代 1000,3 条 Chains 共 3000



	mean	sd	MC_error	val2.5pc	median	val97.5pc	start	sample
alpha	-2.093	1.911	0.1406	-5.813	-2.138	2.133	1001	3000
beta	1.253	0.4189	0.03083	0.3427	1.264	2.076	1001	3000
sigma	1.504	0.4359	0.01382	0.9193	1.422	2.536	1001	3000

结果是否可靠？MCMC 是否收敛？

2.3 检查 MCMC 的有效性和收敛性

MCMC 的有效性 (Efficiency)

- MCMC 产生的序列 $\{\theta^{(i)}\}_{i=1}^{N}$ 是相关的，不是独立的
- 如果序列是 iid 的，则后验均值的方差为

$$Var(\bar{\theta}) = Var\big[\frac{1}{N}\sum_{i=1}^{N}\theta^{(i)}\big] = \frac{\sigma^2}{N}\ (\sigma^2 = Var(\theta)).$$

- 如果序列不是 iid 的，则后验均值的方差为

$$Var(\bar{\theta}) = Var\big[\frac{1}{N}\sum_{i=1}^{N}\theta^{(i)}\big] = \frac{\sigma^2}{N}\times IF,\ IF = \left(1+2\sum_{k=1}^{\infty}\rho_k\right),$$

其中 $\rho_k = Corr(\theta^{(i)}, \theta^{(i+k)})$ 是滞后 k 期的自相关系数.

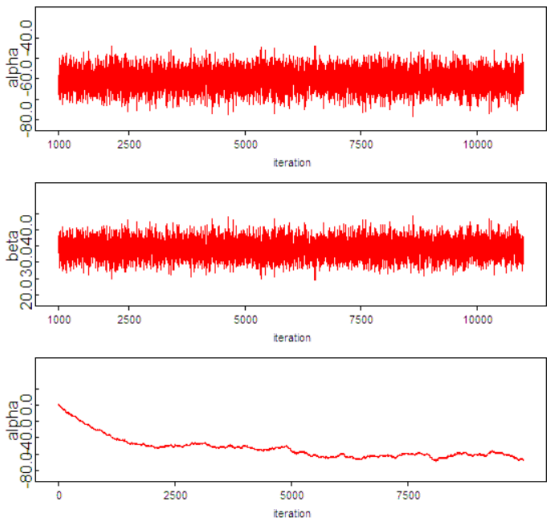
- IF 称为**非效率因子** (Inefficiency Factor)
- **有效样本容量** (Effective Sample Size)：$ESS = N/IF$（表示 MCMC 样本存在自相关时，N 个 MCMC 样本相当于 ESS 个 iid 样本量，如果存在负自相关，IF 可以小于 1）

MCMC 的收敛性 (Convergence)

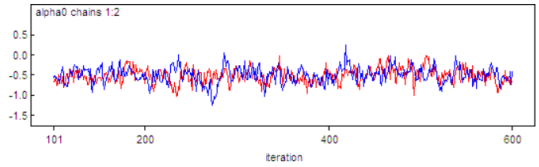
MCMC 收敛性判断：一般只能判断哪种情况不收敛，不能证明其收敛。

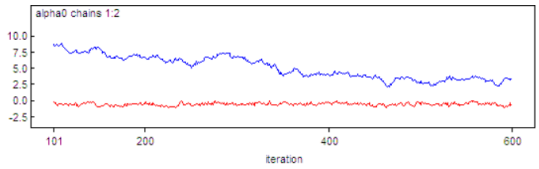
1. Trace 或 History：有无趋势；多条链（多个初始值）是否重合
   初始值推荐：参数的估计值；估计值的 $\pm 2\sigma$
2. 后验分布的密度函数：是否光滑
3. 自相关图：是否存在自相关
4. BGR 图：Brooks-Gelman-Rubin 判断准则：
   BGR=（Pooled CI)/(Average CI)，收敛则 BGR=1
5. R-hat：缩减因子，等于 1 表示收敛
6. n.eff：有效样本容量 (effective sample size)，一般 ESS=1000。

判断 Trace/History



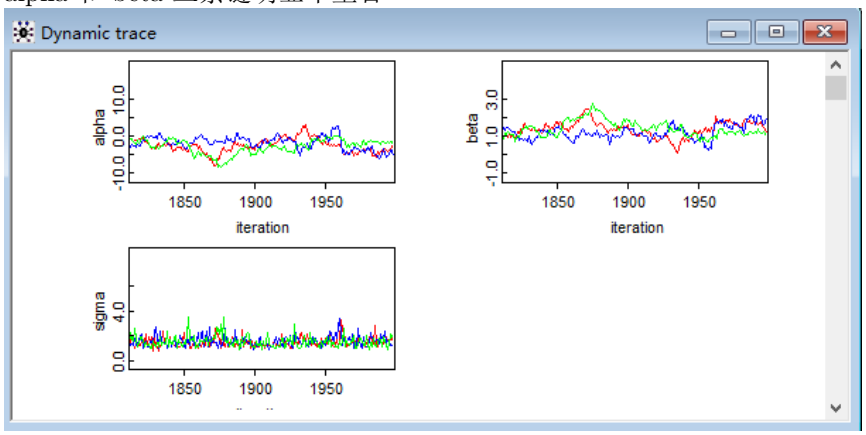
判断 Trace/History：Two Chains



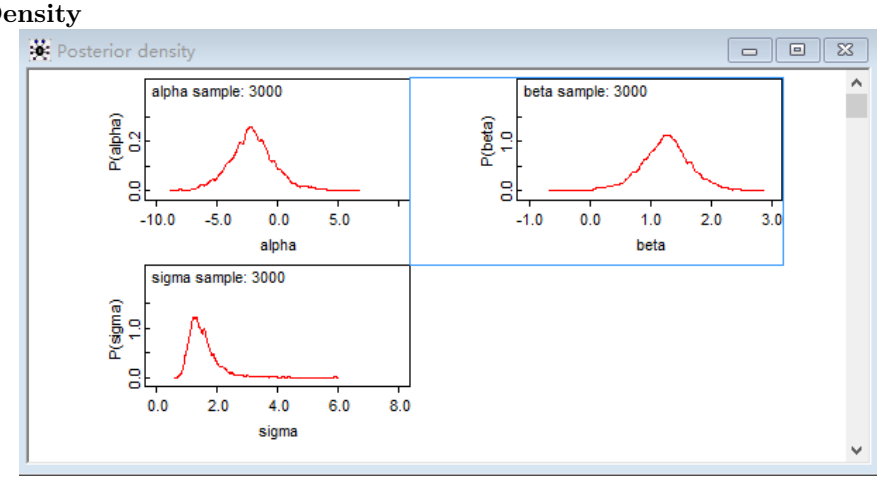


Trace

alpha 和 beta 三条链明显不重合

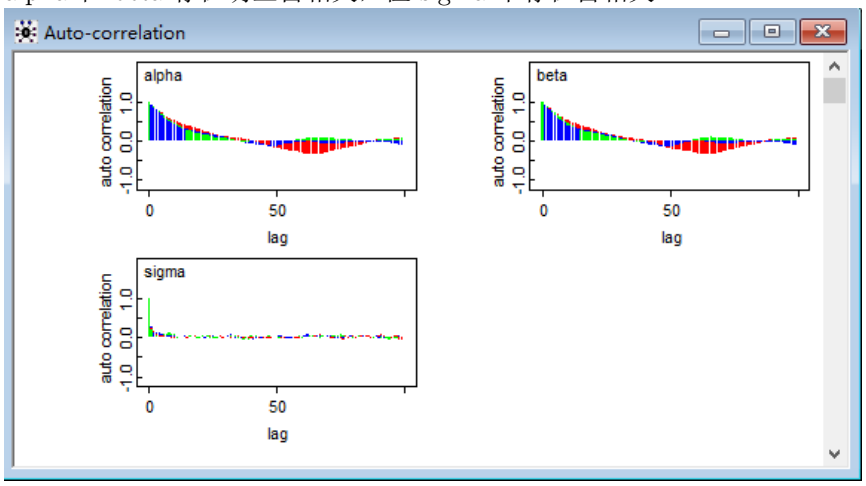


Density

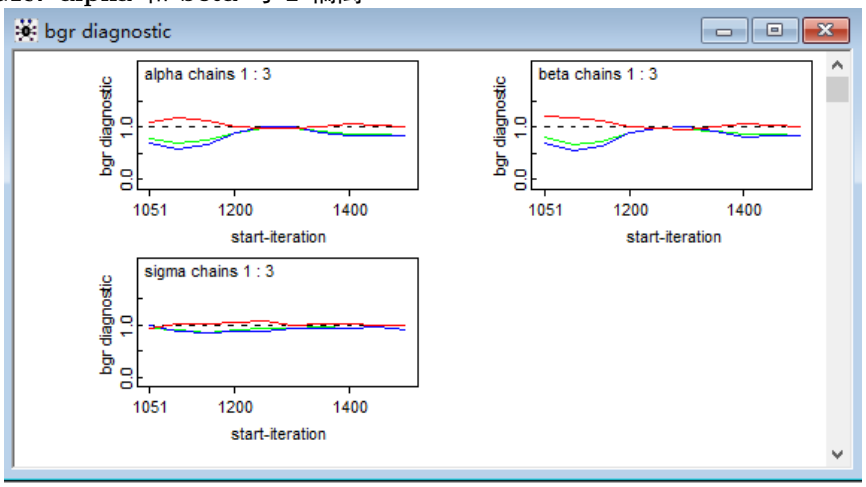


自相关

alpha 和 beta 存在明显自相关，但 sigma 不存在自相关。



BGR：alpha 和 beta 与 1 偏离



存在自相关：改进方法

1. 改变 thin 的值：隔 k 个取一个
2. 自变量中心化

thin=10:

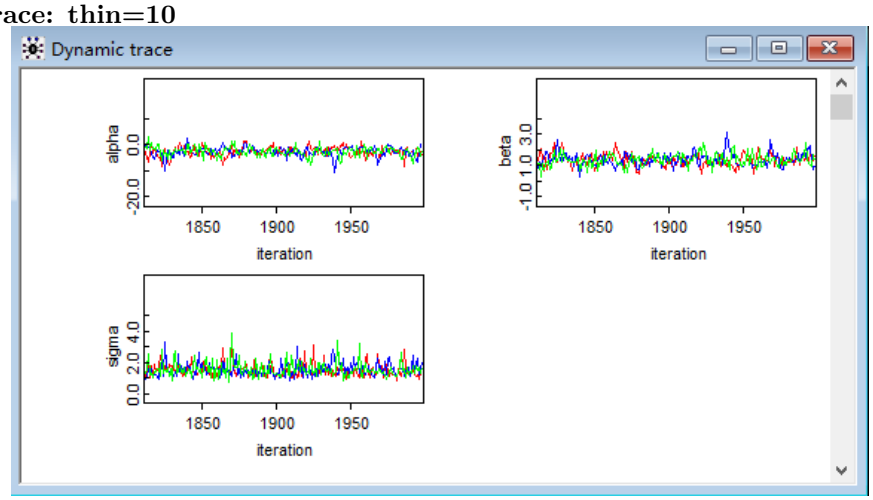
Node statistics

	mean	sd	MC_error	val2.5pc	median	val97.5pc	start	sample
alpha	-2.361	1.897	0.05555	-6.184	-2.35	1.332	1001	3000
beta	1.31	0.4138	0.0122	0.5014	1.306	2.15	1001	3000
sigma	1.509	0.4435	0.008473	0.9107	1.423	2.659	1001	3000

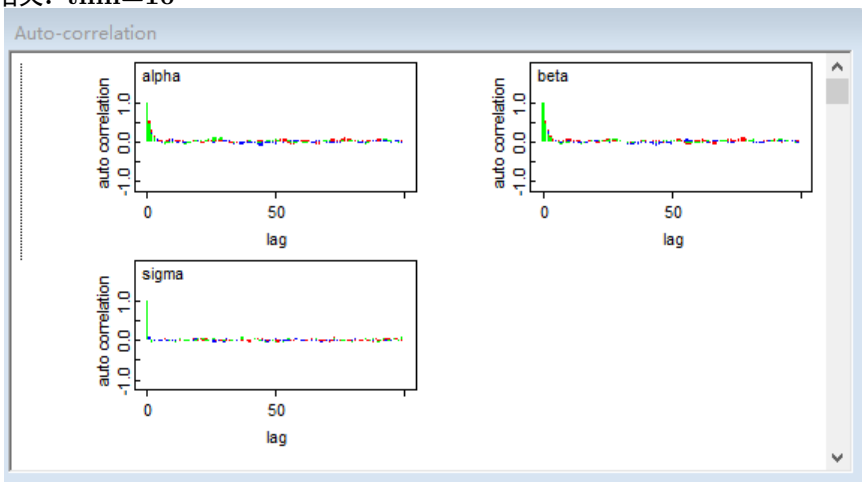
Node statistics

	mean	sd	MC_error	val2.5pc	median	val97.5pc	start	sample
alpha	-2.093	1.911	0.1406	-5.813	-2.138	2.133	1001	3000
beta	1.253	0.4189	0.03083	0.3427	1.264	2.076	1001	3000
sigma	1.504	0.4359	0.01382	0.9193	1.422	2.536	1001	3000

Trace：thin=10



自相关：thin=10

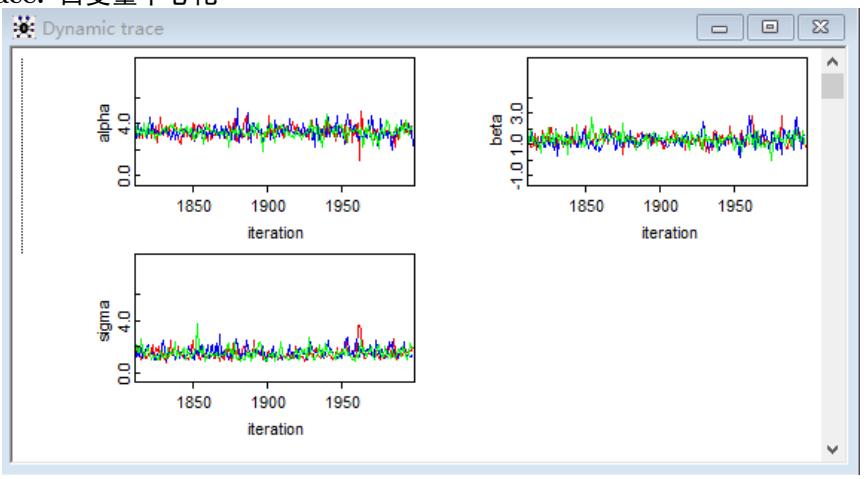


自变量中心化

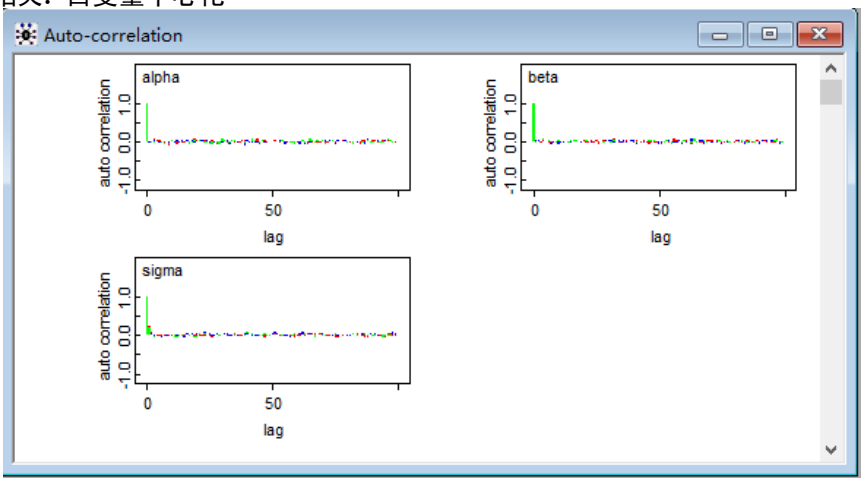
```
model{
  for(i in 1:n ){
   y[i]~dnorm(mu[i],tau)#tau是精度， 不是方差！
   mu[i]<-alpha+beta*(x[i]-mean(x[]))
   }
```

```
  alpha~dnorm( 0.0,0.01)
  beta~dnorm( 0.0,0.01)
  tau~dgamma(0.1,0.1)
  sigma<-1/sqrt(tau)
  }
#data
list(n=10,y=c(3.4,3.8,9.1,2.2,2.6,2.9,2.0,2.7,1.9,3.4),
x=c(5.5,5.9,6.5,3.3,3.6,4.6,2.9,3.6,3.1,4.9))
# initial values (chain 1)
list(tau=1,alpha=0,beta=1)
#chain 2
list(tau=1,alpha=1,beta=1.5)
```

Trace: 自变量中心化



自相关: 自变量中心化



2.4 在 R 中调用 WinBUGS

在 R 中调用 WinBUGS 的方法

调用软件包：R2WinBUGS

```
library(R2WinBUGS)
```

1. 在 R 中准备数据和初值：list 格式
2. 调用 R2WinBUGS：output<-bugs(数据，初值，参数，链条数，样本数等)
3. 输出结果：print(output)
4. 结果图形显示：plot(output)
5. MCMC 有效性：n.eff, Rhat
6. MCMC 收敛性：
   - output.mcmc<-as.mcmc.list(output) （装入结果数据）
   - autocorr.plot(output.mcmc) （自相关检验）
   - traceplot(output.mcmc) （多条马氏链是否重合）
7. 调用其它软件包做图：
   (a) library(lattice): xyplot(output.mcmc)
   (b) library(mcmcplots): traplot(output.mcmc); denplot(output.mcmc)

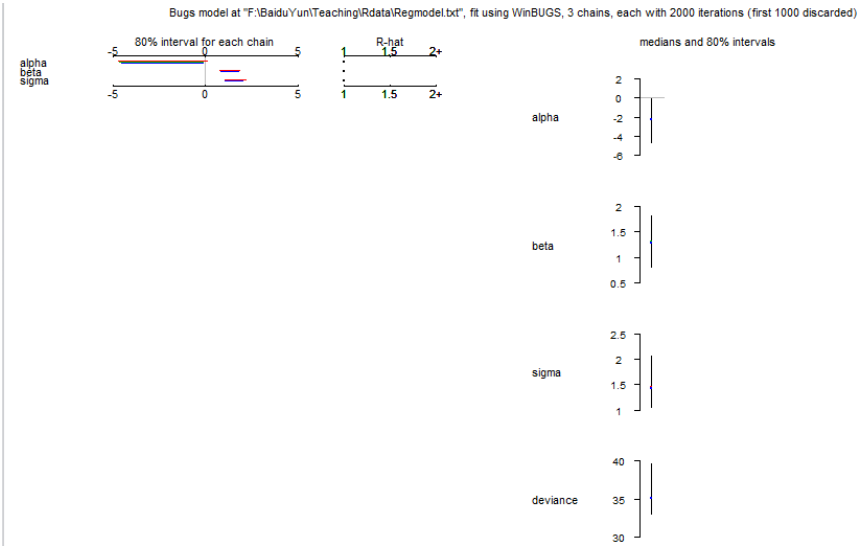
在 R 中调用 R2WinBUGS 的代码

```
#Load the data
n<-10
y<-c(3.4,3.8,9.1,2.2,2.6,2.9,2.0,2.7,1.9,3.4)
x<-c(5.5,5.9,6.5,3.3,3.6,4.6,2.9,3.6,3.1,4.9)
#Call WinBUGS:
library(R2WinBUGS)
data<-list("n","x","y")
parameters<-list("alpha","beta","sigma")
# inits
inits1<-list(tau=5,alpha=4,beta=1.9)
inits2<-list(tau=0.74,alpha=-2.33,beta=1.31)
inits3<-list(tau=3,alpha=0.91,beta=3.32)
inits<-list(inits1,inits2,inits3)
# Call model
output<-bugs(data,inits,parameters,n.chains=3,n.iter=2000,
   n.burnin=1000,n.thin=1,
   model.file="F:\\BaiduYun\\Teaching\\Rdata\\Regmodel.txt",
   bugs.directory="D:\\WinBUGS\\")
#修改为自己电脑中WinBUGS所在的文件夹！！！
```

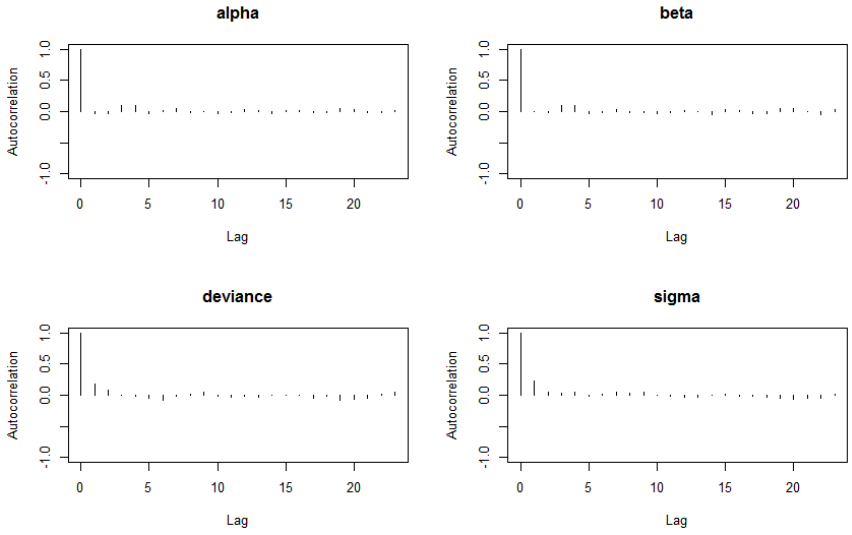
模型结果

```
> print(output)
Inference for Bugs model at "F:\BaiduYun\Teaching\Rdata\Regmodel.txt", fit using WinBUGS,
 3 chains, each with 2000 iterations (first 1000 discarded)
 n.sims = 3000 iterations saved
          mean  sd 2.5%  25%  50%  75% 97.5% Rhat n.eff
alpha    -2.3 1.9 -6.3 -3.4 -2.3 -1.2   1.3    1  3000
beta      1.3 0.4  0.5  1.0  1.3  1.6   2.1    1  3000
sigma     1.5 0.4  0.9  1.2  1.4  1.7   2.5    1  1700
deviance 35.8 2.8 32.6 33.8 35.1 37.2  43.2    1   960

For each parameter, n.eff is a crude measure of effective sample size,
and Rhat is the potential scale reduction factor (at convergence, Rhat=1).

DIC info (using the rule, pD = Dbar-Dhat)
pD = 3.3 and DIC = 39.1
DIC is an estimate of expected predictive error (lower deviance is better).
> 
```

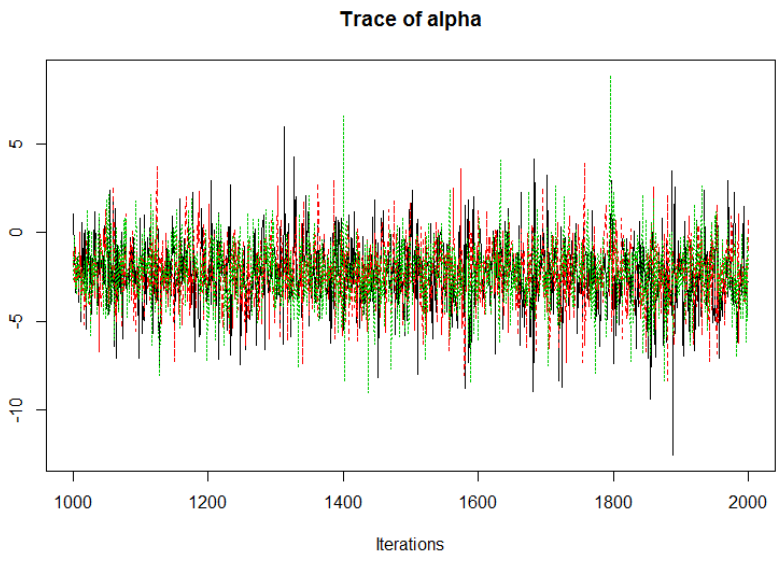
模型结果图示：**plot(output)**



自相关判断：**autocorr.plot(output.mcmc)**

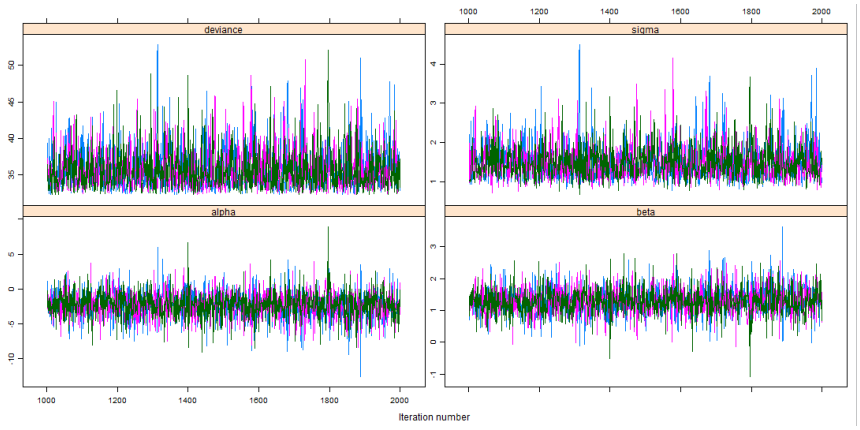


收敛性：traceplot(output.mcmc)

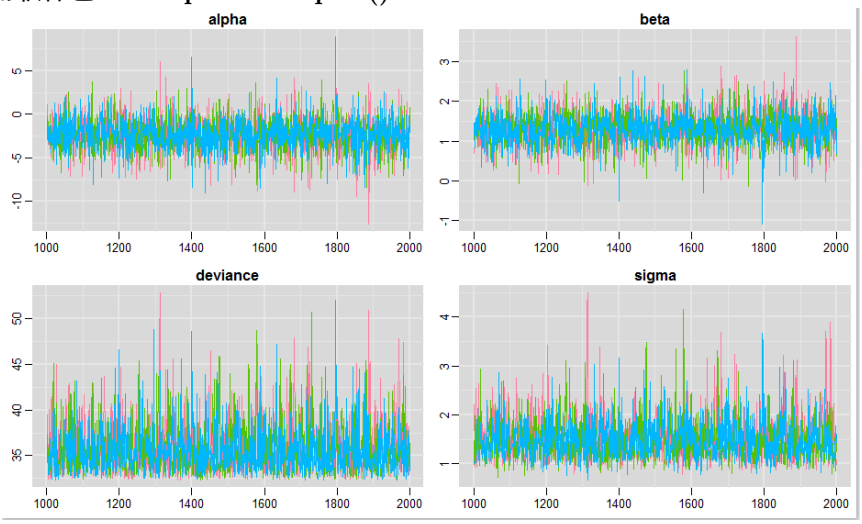


运用软件包 lattice：xyplot()

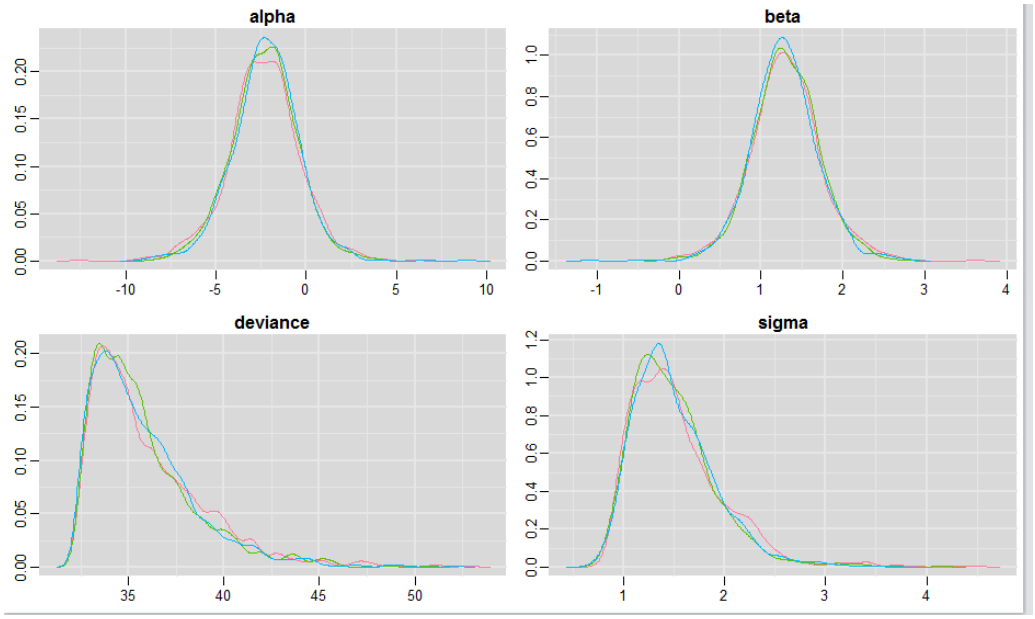
```
library(lattice)
xyplot(output.mcmc, layout=c(2,2), aspect="fill")
```



运用软件包 **mcmcplots**：**traplot()**



运用软件包 **mcmcplots**：**denplot()**



2.5 模型的稳健性改进：正态分布推广到厚尾 t-分布

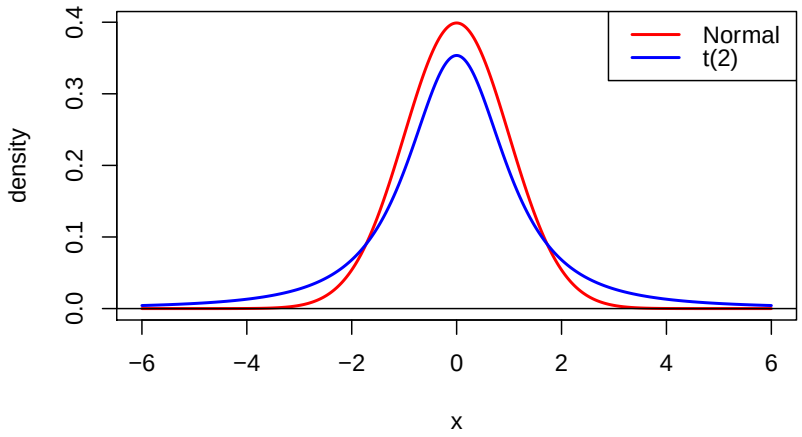
[fragile] 如何处理模型的异常点？

- 在贝叶斯模型中，如果采用无信息先验，模型系数和误差估计与传统线性模型结果一致
- 在这个线性模型例子中，第 3 个数据点是异常点。该异常点如何处理？
  - 传统方法：删除，或数据变换
  - 贝叶斯方法：极为简单，允许误差项的分布有更长的尾部，把正态误差改为长尾分布。

模型：正态误差改为 t(2) 分布

```
model{
  for(i in 1:n ){
    y[i]~dt(mu[i],tau,2)
    mu[i]<-alpha+beta*x[i]
    }
  alpha~dflat()
  beta~dflat()
  tau~dgamma(0.001,0.001)
  sigma<-1/sqrt(tau)
  }
```

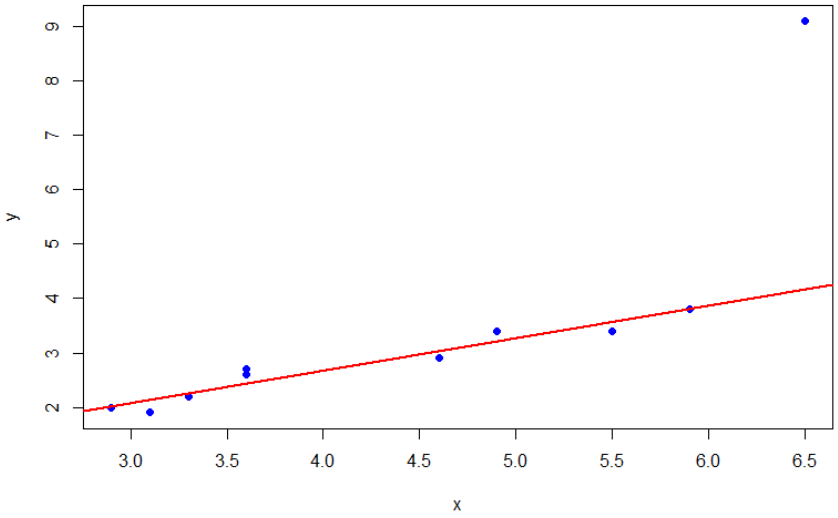
标准正态与 t(2) 分布



结果：异常点对系数估计几乎没有影响

```
> print(output,digits=3)
Inference for Bugs model at "F:\BaiduYun\Teaching\Rdata\Regmodel.txt", fit using WinBUGS,
 3 chains, each with 20000 iterations (first 2000 discarded), n.thin = 5
 n.sims = 10800 iterations saved
           mean    sd   2.5%    25%    50%    75%  97.5%  Rhat n.eff
alpha     0.284 0.404 -0.526  0.063  0.282  0.506  1.090 1.001  7700
beta      0.598 0.095  0.417  0.545  0.595  0.647  0.797 1.001  9200
sigma     0.235 0.113  0.101  0.161  0.210  0.277  0.507 1.001  8200
deviance 16.449 3.133 12.900 14.150 15.620 17.880 24.640 1.001 11000

For each parameter, n.eff is a crude measure of effective sample size,
and Rhat is the potential scale reduction factor (at convergence, Rhat=1).

DIC info (using the rule, pD = Dbar-Dhat)
pD = 3.7 and DIC = 20.2
DIC is an estimate of expected predictive error (lower deviance is better).
```

散点图：贝叶斯 t-分布模型



3 RStan 运用

3.1 介绍

Why Stan?

- **能力强**：Stan 能解决复杂模型，e.g.
  - Multilevel generalized linear models
  - Interacted predictors at multiple levels
  - Nonconjugate coeffcient priors
  - Latent effects etc
- **速度快**：Stan stands for No-U-Turn sampler, a variant of Hamiltonian Monte Carlo（HMC），可以运用并行运算，基于 C 或 C++，计算速度快（只是代码的首次编译时间稍长）
- **功能多**：Stan 除了具备其它软件的所有功能外，还能计算后验密度函数的对数及其梯度（一阶导数），在更深入的研究中有用
- **范围广**：Stan 除了用于贝叶斯模型，还可以用于计算优化问题
- 网络资源：
  - https://mc-stan.org/users/documentation/
  - https://github.com/stan-dev/stan/wiki/Prior-Choice-Recommendations

RStan 的安装和准备

- RStan 的安装
  - https://github.com/stan-dev/rstan/wiki/RStan-Getting-Started
- 调用
  - library("rstan") # observe startup messages
- 如果 Windows 是多核 CPU 且内存足够，可以用并行运算，并行运算运行多个马尔可夫链
  - options(mc.cores = parallel::detectCores())
- 自动保存编译进程，除非有修改，否则无需重新编译
  - rstan_options(auto_write = TRUE)

运用 RStan 的一般流程

1. 为准备一个 Stan 模型代码文件；
2. 准备一个输入数据的列表；
3. 从后验分布中抽取样本；
4. 检验马尔可夫链的收敛性；
5. 基于后验分布的样本进行贝叶斯推断。

3.2 建立贝叶斯模型和 Stan 代码

八校例子：实验数据

为研究一项训练方法对 SAT-V 成绩的影响 (effects)，对八个中学安排了对照组随机化试验。每个学校训练效果的点估计 y 和标准误 σ 如下表，试据此建立一个多层**元分析** (meta analysis) 模型，对该训练方法进行综合评价。

学校 (j)	效应估计值 (y_j)	标准误 (σ_j)
A	28	15
B	8	10
C	-3	16
D	7	11
E	-1	9
F	1	11
G	18	10
H	12	18

问题讨论

1. 八个学校之间训练方法的效果是否存在显著差异？
    - A,G 效果较好（大于 18），C、E 反效果，其它效果一般。
    - 正确吗？
    - 检查 95% 置信区间
    - 假设 $\theta_A \sim N(28, 15)$，则 $P(\theta > 28|y) = 0.5$，结论可疑
2. 如何估计该训练方法在八间学校的总体效果？
    - 因为八间学校采用同一训练方法，所以可以看作训练效果 y 来自同一个正态总体 $N(\mu, \sigma^2)$。
    - 合理吗？

$$E(\theta|y) = \sum_{i=1}^{8} \frac{y_i/\sigma_i^2}{\sum_{j=1}^{8} 1/\sigma_j^2} = 7.7$$

    - $P(\theta_A - \theta_C < 0|y) = 0.5$

八校模型：贝叶斯两层元分析

贝叶斯模型：

$$\begin{aligned} y_j &\sim \mathsf{Normal}(\theta_j, \sigma_j), \quad j = 1, \ldots, 8 \\ \theta_j &\sim \mathsf{Normal}(\mu, \tau), \quad j = 1, \ldots, 8 \\ p(\mu, \tau) &\propto 1, \end{aligned}$$

其中 σ_j（试验效果估计值 y_j 的标准差）为已知。
这是一个两层模型：

- 第一层是 y_j 对 θ_j（学校效应），为正态线性模型：

$$y_j = \theta_j + \varepsilon_j, \varepsilon_j \sim N(0, \sigma_j)$$

- 第二层是 θ_j 对 μ（总平均效应），也是线性模型：

$$\theta_j = \mu + \tau\eta_j, \eta_j \sim N(0, 1)$$

- μ 为 θ_j 的均值，τ 为 θ_j 的标准差，它们是参数 θ_j 的参数，因此 (μ, τ) 称为超参数 (Hyper parameter)。

第一步：准备一个 Stan 模型代码文件

一个 Stan 模型代码文件通常包含如下模块（顺序不能变）：

1. *functions*, where we define functions to be used in the blocks below.
2. **data**, declares the data to be used for the model
3. *transformed data*, makes transformations of the data passed in above
4. **parameters**, defines the unknowns to be estimated, including any restrictions on their values.
5. *transformed parameters*, often it is preferable to work with transformations of the parameters and data declared above; in this case we define them here.
6. **model**, the log probability function is defined.
7. *generated quantities*, generates a range of outputs from the model (posterior predictions, forecasts, values of loss functions, etc.).

八校模型的 Stan 代码构成

1. data：把贝叶斯后验分布中作为条件的数据类型 (interger,real,vector,array)、范围、名称一一列出。
   - J：学校数，正整数
   - $(y_1, \ldots, y_J)$，各学校的效果估计值，向量
   - $(\sigma_1, \ldots, \sigma_J)$，各学校的效果估计值的标准误，向量，正实数
2. parameters：指定待估参数，这里 θ_j 没有指定为参数，因为第二层模型中，它还依赖于两个超参数，因此属于变换的参数。
   - 总效应 μ；标准差 τ
   - 标准化的学校效应：$(\eta_1, \ldots, \eta_J)$。
3. transformed parameters：指明变换的参数，以及变换关系。把需要变换的参数 θ_j 放在这里，可以提升迭代法计算的效率。
   - $(\theta_1, \ldots, \theta_J)$；$\theta_j = \mu + \tau\eta_j$
4. model：指定模型的分布或似然函数的对数。注意：在 Stan 的分布 $N(\mu, \sigma)$ 中，给出的是标准差（一般情形给出的是方差，而 WinBUGS 给出的精度）!
   - 为了提高效率，模型部分一般以向量形式给出，而不用循环语句
   - $\eta_j \sim \text{Normal}(\eta|0, 1)$；$y_j \sim \text{Normal}(y|\theta_j, \sigma_j)$

八校模型的 **Stan** 模型代码

```
//Save as shools.stan
data {
   int<lower=0> J;  // number of schools
   real y[J];   // estimated treatment effects
   real<lower=0> sigma[J];  // s.e. of effect estimates
 }
parameters {
   real mu;
   real<lower=0> tau;
   vector[J] eta;
 }
transformed parameters {
   vector[J] theta;
   theta = mu + tau * eta;
 }
model {
   eta ~ normal(0, 1);   # diff with 'dnorm'
   y ~ normal(theta, sigma);
}
```

注意：

- 模型代码一般保存为一个.rstan 文件
- 最后一行必须是空行，不能包含任何字符，包括空格
- Stan 中有很多函数（概率分布、矩阵运算等），与 R 和 BUGS 存在部分差异
- 如果需要计算后验密度函数，model 模块可以写成如下： ”target” 表示目标函数，”lpdf” stands for ”log probability density function”:

```
model {
   target += normal_lpdf(eta | 0, 1);
   target += normal_lpdf(y | theta, sigma);
}
```

3.3 准备 Stan 的输入数据

第二步：准备一个数据列表

Stan 的数据格式是一个列表（list），可以包含字符名称、向量等。建议以文件形式保存在硬盘。

```
schools_data <- list(J = 8,
      y = c(28,  8, -3,  7, -1,  1, 18, 12),
      sigma = c(15, 10, 16, 11,  9, 11, 10, 18))
```

3.4 应用 Stan 从后验分布中抽取样本

第三步：从后验分布中抽取样本

所有结果存储在 rstan 的 stanfit 对象中。

```
library(rstan)
fit <- stan(
  file = "schools.stan",#Stan program
  data = schools_data,  #named list of data
  chains = 4,           #number of Markov chains
  warmup = 1000, #number of warmup iterations per chain
  iter = 2000,   #total number of iterations per chain
  cores = 2,     #number of cores
  refresh = 0    #no progress shown
  )
```

3.5 模型结果及抽样收敛性

第四步：检验 MCMC 收敛性及给出结果

给出及图示的常用方法

- 用 print() 给出收敛性判断及参数估计结果
- 用 plot() 给出结果的图形
  - plot(fit)
    - plot(fit, show_density = TRUE, ci_level = 0.5, fill_color = "purple")
    - plot(fit, plotfun = "hist", pars = "theta", include = FALSE)
    - plot(fit, plotfun = "trace", pars = c("mu", "tau"), inc_warmup = TRUE)
    - plot(fit, plotfun = "rhat") + ggtitle("Example of adding title to plot")
- 用 traceplot() 判断多条链的收敛性（是否重合）
- 用 pairs() 给出参数分布直方图和相互之间的散点图组合

Rstan 中用 ggplot2 作图

stan_plot Posterior intervals and point estimates

stan_trace Traceplots

stan_hist Histograms

stan_dens Kernel density estimates

stan_scat Scatterplots

stan_diag Diagnostics for Hamiltonian Monte Carlo and the No-U-Turn Sampler

stan_rhat Rhat

stan_ess Ratio of effective sample size to total posterior sample size

stan_mcse Ratio of Monte Carlo standard error to posterior standard deviation

stan_ac Autocorrelation

其它软件包

1. shinystan：全面显示结果和 MCMC 信息，但要人机交互
2. bayesplot 给出基于 ggplot 的 MCMC 诊断、后验分布结果及检查图形
3. rstanarm 和 brms 给出回归模型（包括广义回归模型）的 stan 集成
4. loo 用于模型比较
5. tidybayes: Tidy Data and 'Geoms' for Bayesian Models

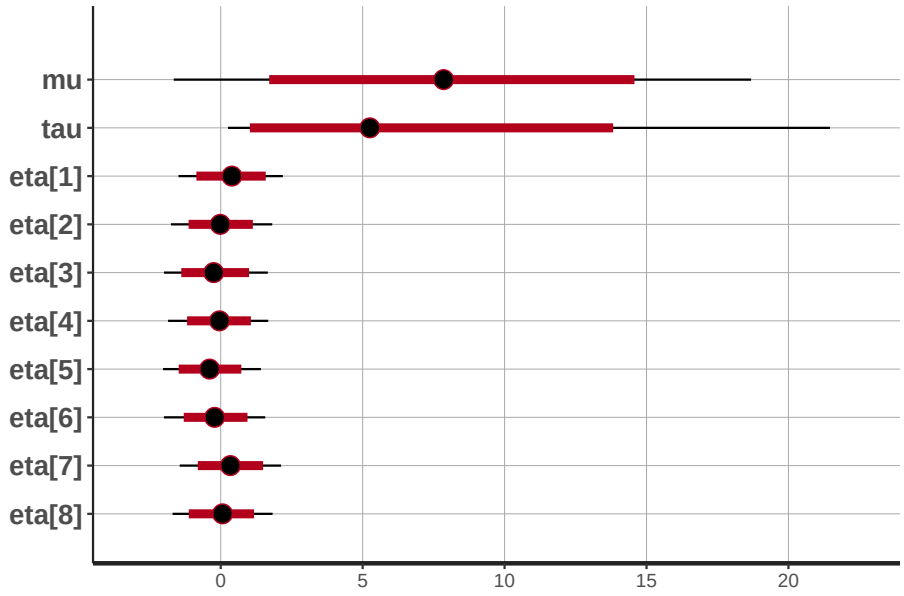
Stan 给出的基本结果：print(fit)

```
print(fit, pars=c("theta", "mu", "tau", "lp__"),
      probs=c(.1,.5,.9))
```

```
Inference for Stan model: schools.
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    10%    50%    90% n_eff Rhat
theta[1]  11.52    0.15 8.53   2.31  10.17  22.66  3071    1
theta[2]   7.86    0.09 6.34   0.19   7.85  15.69  4816    1
theta[3]   5.90    0.14 7.97  -3.67   6.36  14.87  3257    1
theta[4]   7.68    0.10 6.69  -0.42   7.57  15.78  4637    1
theta[5]   4.94    0.10 6.25  -3.28   5.33  12.54  4243    1
theta[6]   6.20    0.11 6.64  -2.03   6.58  14.30  3747    1
theta[7]  10.68    0.11 6.74   2.68  10.00  19.65  3621    1
theta[8]   8.45    0.14 7.75  -0.23   8.01  17.54  2917    1
mu         8.02    0.13 5.22   1.71   7.85  14.57  1731    1
tau        6.66    0.17 5.84   1.03   5.25  13.82  1213    1
lp__     -39.55    0.07 2.67 -43.17 -39.26 -36.39  1297    1

Samples were drawn using NUTS(diag_e) at Mon Mar 11 10:32:02 2019.
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).
```

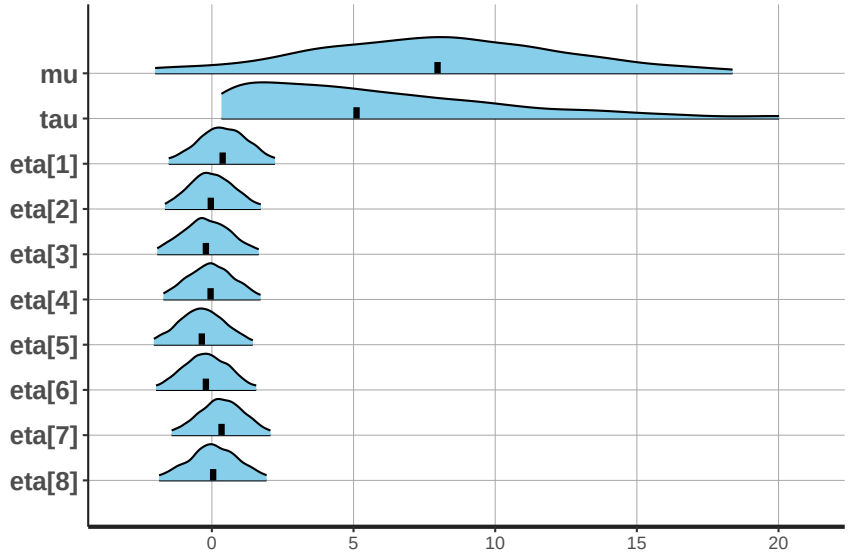
其中“lp___”是后验分布密度的对数（非标准化）。

Stan 结果的区间：plot(fit)



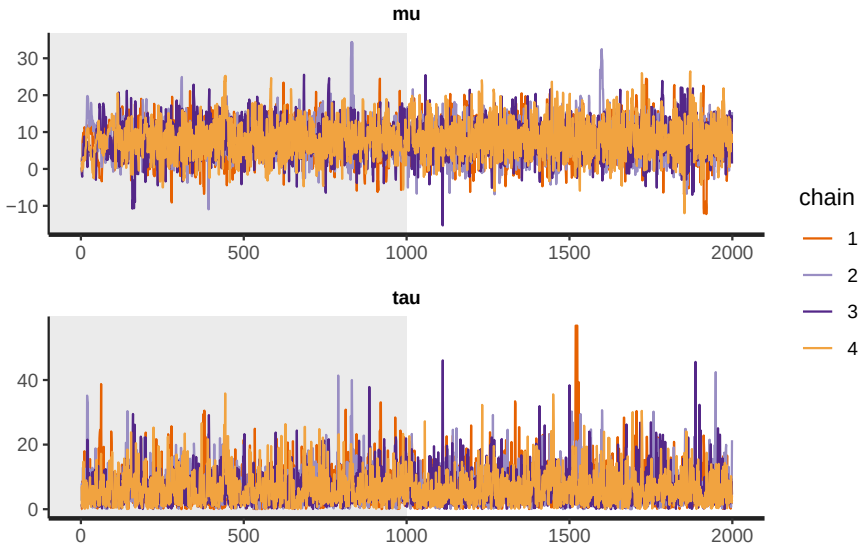
参数的后验密度函数及 95% 可信区间

```
plot(fit, show_density = TRUE, ci_level = 0.95,
     fill_color = "skyblue")
```



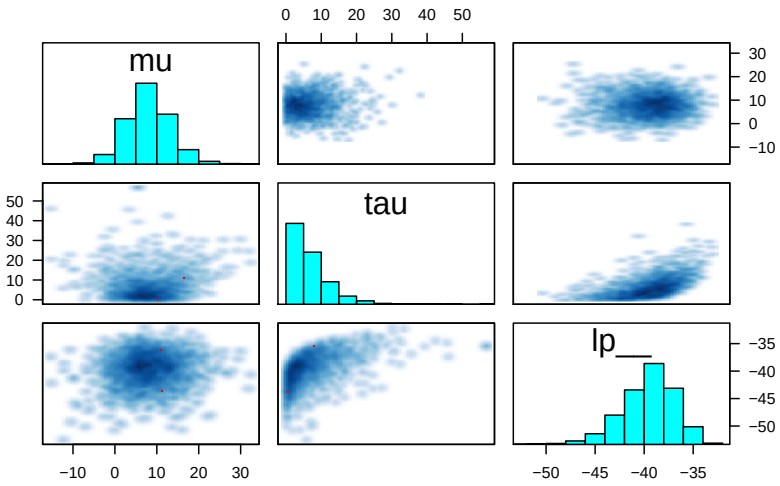
MCMC 收敛性：traceplot(fit)

```
traceplot(fit, pars = c("mu", "tau"),
          inc_warmup = TRUE, nrow = 2)
```



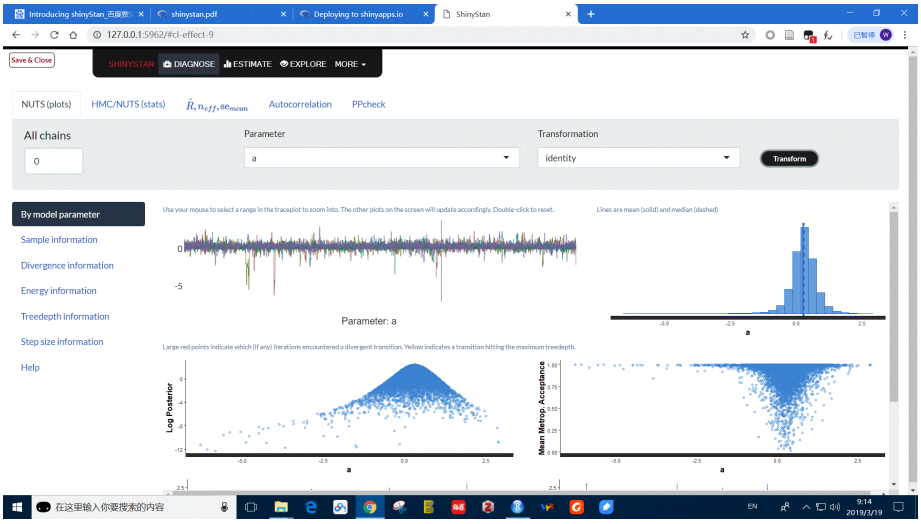
pairs(fit)

```
pairs(fit1, pars=c("mu", "tau", "lp__"))
```



Shinystan

```
library("shinystan")
launch_shinystan(fit)
```



4 转换点 (Change Point) 模型的贝叶斯估计

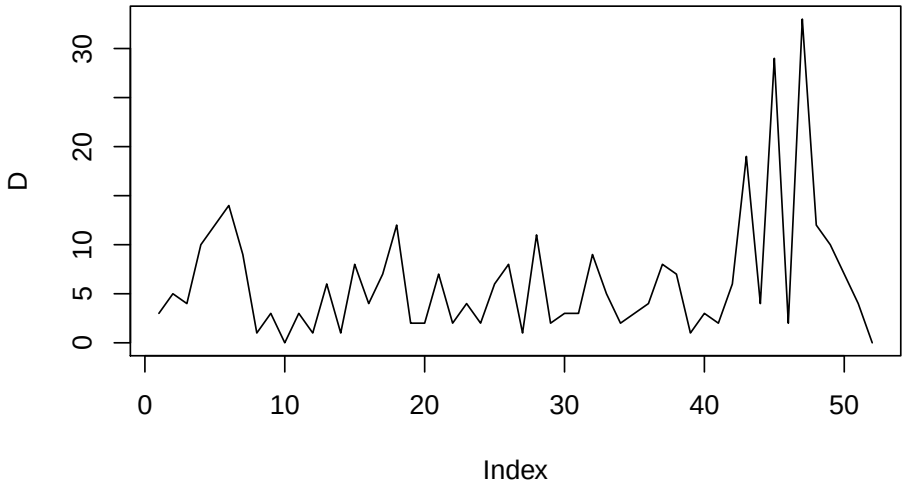
4.1 转换点 (Change Point) 模型

例 2：美军在阿富汗死亡人数转换点的估计

美军（美军）October 2001 至 January 2007 在阿富汗各月死亡人数（BaM：data(afghan.deaths)）显示，塔利班最近明显改变了攻击策略：

3	5	4	10	12	14	9	1	3	0
3	1	6	1	8	4	7	12	2	2
7	2	4	2	6	8	1	11	2	3
3	9	5	2	3	4	8	7	1	3
2	6	19	4	29	2	33	12	10	7
4	0								

美军在阿富汗死亡人数



转换点 (Change Point) 问题

问题是：塔利班什么时候改变攻击策略的?

设 D_t 表示第 t 个月的死亡人数 $(t = 1, \ldots, n)$，则

- 早期数据：$D_1, D_2, \ldots, D_k | \lambda \sim \text{Pois}(\lambda)$
- 后期数据：$D_{k+1}, D_{k+2}, \ldots, D_n | \phi \sim \text{Pois}(\phi)$
- 需要估计泊松分布的均值 λ, ϕ 和转换点 (change point) k

4.2 转换点模型的 WinBUGS 应用

贝叶斯模型一

$D(t) \sim \text{Poisson}(\mu(t)), t = 1, \ldots, n$

$\log \mu(t) = b_1 + \delta(t-k) b_2$

其中

$$\delta(t-k) = \begin{cases} 0 & \text{if } (t-k) < 0 \\ 1 & \text{if } (t-k) \geq 0 \end{cases}$$

先验分布:

$$\begin{aligned} b_1 &\sim N(0, 1E-6) \\ b_2 &\sim N(0, 1E-6) \\ k &\sim \text{uniform}(1, n) \end{aligned}$$

参数变换：$\lambda = \exp(b_1),\ \phi = \exp(b_1 + b_2)$

第一步：建立一个 **BUGS** 模型代码文件

```
// Save as ch6_cp_model.bug
model{
  for(year in 1 : N){
    D[year] ~ dpois(mu[year])
    log(mu[year])<-b[1]+step(year-changeyear)*b[2]
    }
  for (j in 1:2) {b[j] ~ dnorm(0.0,1.0E-6)}
  changeyear ~ dunif(1,N) #is continuous Unif
  lambda <- exp(b[1])
  phi <- exp(b[1]+b[2])
```

第二步：准备数据列表

```
# 准备数据
library("BaM")
data(afghan.deaths)
D <- afghan.deaths
N <- length(D)
# 准备数据列表（WinBUGS 输入数据）
data = list("N","D")
# 准备初始值
inits = function() {list(b = c(0,0),changeyear = 40)}
# 指明模型参数
parameters <- c("changeyear","b")
```

第三步：运行 **WinBUGS** 进行 **MCMC** 抽样

```
output <- bugs(
     data,
     inits,
     parameters,
     model.file = "ch6_cp_model.bug",
     n.chains = 4,
     n.iter = 10000,
     n.burnin = 2000,
     n.thin = 5,
     bugs.directory = "C:\\WinBUGS14\\"
)
```

第四步：检验 **MCMC** 收敛性及给出结果

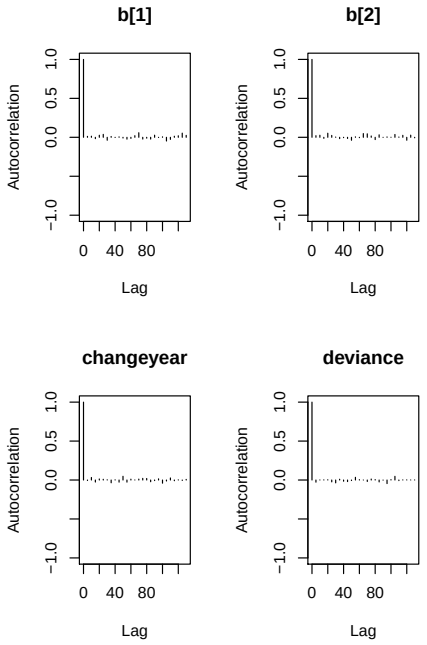
```
print(output, digits = 3)
```

```
Inference for Bugs model at "ch6_cp_model.bug", fit using WinBUGS,
 4 chains, each with 10000 iterations (first 2000 discarded), n.thin = 5
 n.sims = 6400 iterations saved
              mean    sd    2.5%     25%     50%     75%   97.5%  Rhat
changeyear  42.331 0.486  41.140  42.090  42.390  42.700  42.970 1.001
b[1]         1.587 0.070   1.446   1.541   1.589   1.633   1.721 1.001
b[2]         0.886 0.117   0.654   0.807   0.886   0.964   1.116 1.001
deviance   376.053 2.484 373.500 374.200 375.300 377.300 382.500 1.001
           n.eff
changeyear  4900
b[1]        5200
b[2]        6400
deviance    6400
For each parameter, n.eff is a crude measure of effective sample size,
and Rhat is the potential scale reduction factor (at convergence, Rhat=1).
DIC info (using the rule, pD = Dbar-Dhat)
pD = 2.6 and DIC = 378.6
DIC is an estimate of expected predictive error (lower deviance is better).
```

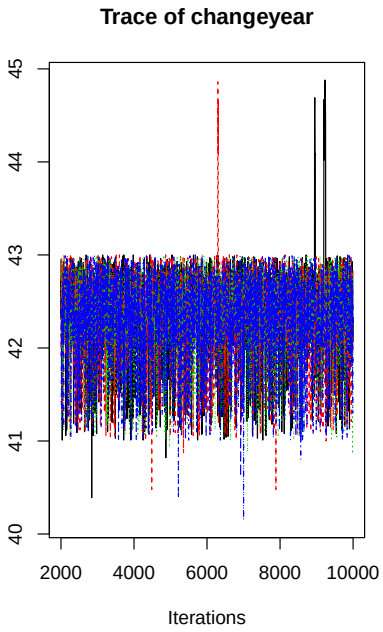
参数变换后的结果

```
B <- output$mean$b
lambda <- exp(B[1])
lambda
phi <- exp(B[1] + B[2])
phi
output$mean$changeyear
[1] 4.887069
[1] 11.85329
[1] 42.33081
```

MCMC 收敛性：自相关



MCMC 收敛性：traceplot



4.3 转换点模型的 Stan 应用

贝叶斯模型二

- 早期数据：$D_1, D_2, \ldots, D_k | \lambda \sim \text{Pois}(\lambda)$
- 后期数据：$D_{k+1}, D_{k+2}, \ldots, D_T | \phi \sim \text{Pois}(\phi)$

先验分布：

$$\begin{aligned}
&\lambda \sim \text{gamma}(\alpha, \beta) \\
&\phi \sim \text{gamma}(\gamma, \delta) \\
&k \sim \text{discrete uniform on } \{1, 2, ..., T\}
\end{aligned}$$

1. 先验参数的选择
   - 前面 40 个数据的均值为 4.95，后面 12 个数据的均值为 10.66，据此可给出先验分布的超参数：
   - $\alpha = 1, \beta = 0.2; \gamma = 4, \delta = 0.4$
2. 如何写出模型的 Stan 代码?
   - Stan 不支持离散型参数的抽样（因为 Hamiltonian Monte Carlo 是梯度法）
   - 思路：把联合似然函数中的 k 通过边缘化去掉

似然函数

利用 (k, D) 的联合分布的边缘化，去掉 k：

$$\begin{aligned}
p(D|\lambda, \phi) &= \sum_{k=1}^{T} p(k, D|\lambda, \phi) \\
&= \sum_{k=1}^{T} \pi(k) p(D|k, \lambda, \phi) \\
&= \sum_{k=1}^{T} \left[\text{Uniform}(k|1, T) \prod_{t=1}^{T} \text{Poisson}(D_t | t < k ? \lambda : \phi) \right]
\end{aligned}$$

其中用到如下 Stan 函数：

$$c ? x_1 : x_2 = \begin{cases} x_1 & \text{If } c \text{ is TRUE} \\ x_2 & \text{If } c \text{ is FALSE} \end{cases}$$

似然函数的对数

$$\log p(D|\lambda,\phi) = \log\_\text{sum}\_\exp_{k=1}^{T}\Big[\text{logUniform}(k|1,T) + \sum_{t=1}^{T}\text{logPoisson}(D_t|t<k?\lambda:\phi)\Big],$$

其中利用了 Stan 的如下函数：

$$\log\_\text{sum}\_\exp_{i=1}^{n} A_i = \log\sum_{i=1}^{n}\exp(A_i).$$

Stan 代码

```
data {
  real<lower=0> alpha_e;
  real<lower=0> beta_e;
  real<lower=0> alpha_l;
  real<lower=0> beta_l;
  int<lower=1> T;
  int<lower=0> D[T];
}
transformed data {
  real log_unif;
  log_unif = -log(T);
}
parameters {
  real<lower=0> lambda;
  real<lower=0> phi;
}

transformed parameters {
 vector[T] lp;
 lp = rep_vector(log_unif, T);
  for (k in 1:T)
  for (t in 1:T)
   lp[k] = lp[s] + poisson_lpmf(
    D[t] | t < k ? lambda:phi);
}
model {
  lambda ~ gamma(alpha_e,beta_e);
  phi ~ gamma(alpha_l,beta_l);
  target += log_sum_exp(lp);
}
```

代码优化

计算 lp 时用了双重循环语句，降低了效率。

```
transformed parameters {
  vector[T] lp;
```

```
  {
    vector[T + 1] lp_e;
    vector[T + 1] lp_l;
    lp_e[1] = 0;
    lp_l[1] = 0;
    for (t in 1:T) {
      lp_e[t + 1] = lp_e[t] + poisson_lpmf(D[t] | e);
      lp_l[t + 1] = lp_l[t] + poisson_lpmf(D[t] | l);
    }
    lp = rep_vector(log_unif + lp_l[T + 1], T)
                  + head(lp_e, T) - head(lp_l, T);
  }
}
```

如何计算分布转换点 k?

在每次 MCMC 抽样中，样本来自后验分布 $p(\lambda, \phi|D)$，并计算了 lp 值。

记

$$p(k|D) \propto q(k|D) = \frac{1}{M}\sum_{i=1}^{M}\exp[\mathrm{lp}(i,k)],$$

其中 $\mathrm{lp}(i,k)$ 表示转换点为 k 时，对后验分布抽取的第 i 个 MCMC 样本的 lp 值。由于后验分布为 $p(\lambda, \phi|D)$，因此对所有 MCMC 样本平均相当于把 (λ, ϕ) 边缘化消去了。通过对上式正则化，可以得出转换点 k 的后验分布：

$$p(k|D) = \frac{q(k|D)}{\sum_{j=1}^{T} q(j|D)}$$

Softmax 函数

为了计算 $p(k|D)$，Stan 定义了一个新的函数 Softmax。

假设 x 为 M 维 simplex 向量（所有分量之和等于 1），定义：

$$\mathrm{softmax}(x) = \frac{\exp(x)}{\sum_{i=1}^{M}\exp(x_i)}$$

显然，

$$\begin{aligned}\log \mathrm{softmax}(x) &= y - \log\sum_{i=1}^{M}\exp(y_i)\\ &= y - \log\_\mathrm{sum}\_\exp(y)\end{aligned}$$

计算分布转换点 k 的 Stan 代码

只需在 Stan 模型代码的最后增加一个 generated quantities 模块，即可以得到后验分布 $p(k|D)$ 的 MCMC 样本。

```
generated quantities {
    int<lower=1,upper=T> switchpoint;
    switchpoint = categorical_rng(softmax(lp));
}
```

准备一个数据列表

```
data(afghan.deaths)
D <- afghan.deaths
T <- length(D)
al_e <- 1
be_e <- 0.2
al_l <- 4
be_l <- 0.4
data <- list(T=T, D=D, al_e=al_e, be_e=be_e,
             al_l=al_l, be_l=be_l)
```

从后验分布中抽取样本

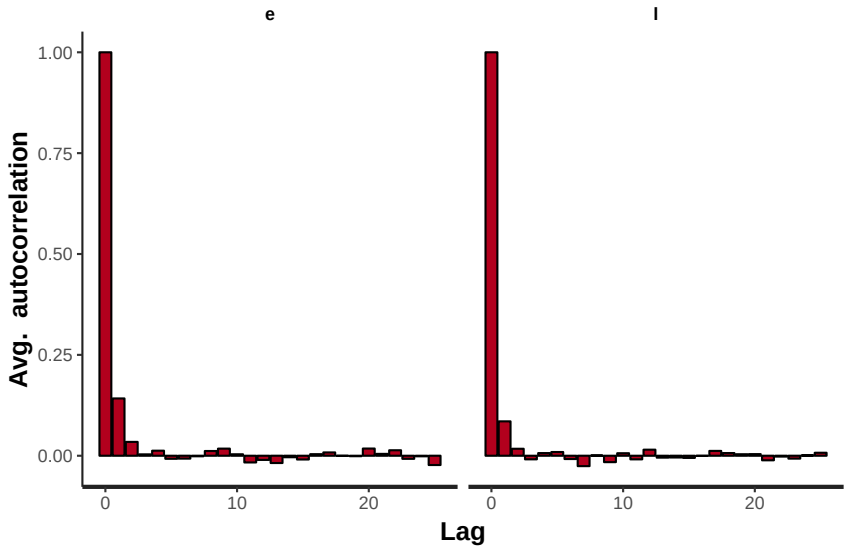
```
fit <- stan(
  file = "ch6_cp_model.stan", #Stan program
  data = data,  #named list of data
  chains = 4,   #number of Markov chains
  warmup = 2000, #number of warmup iterations per chain
  iter = 5000,   #total number of iterations per chain
  cores = 4,   #number of cores (could use one per chain)
  refresh = 0   #no progress shown
  )
```

检验 MCMC 收敛性及给出结果

```
print(fit)
Inference for Stan model: changepoint_model.
4 chains, each with iter=5000; warmup=2000; thin=1;
post-warmup draws per chain=3000, total post-warmup draws=12000.
               mean se_mean    sd    2.5%     25%     50%     75%   97.5% n_eff Rhat
e              4.90    0.00  0.34    4.26    4.66    4.90    5.13    5.61 10884    1
l             11.83    0.01  1.09    9.78   11.08   11.80   12.53   14.09 11164    1
lp[1]       -301.54    0.25 26.70 -360.74 -317.91 -299.40 -282.48 -255.64 11115    1
lp[2]       -297.25    0.25 25.89 -354.75 -313.09 -295.17 -278.72 -252.78 11111    1
lp[3]       -294.72    0.24 25.26 -350.84 -310.16 -292.70 -276.63 -251.37 11110    1
...
lp[51]      -238.25    0.05  5.73 -250.82 -241.84 -237.73 -234.15 -228.65 10920    1
lp[52]      -234.84    0.05  5.49 -247.02 -238.19 -234.24 -230.89 -225.93 10815    1
switchpoint   42.83    0.00  0.41   42.00   43.00   43.00   43.00   43.00 11831    1
lp__        -185.77    0.01  1.01 -188.44 -186.18 -185.44 -185.04 -184.78  5455    1
Samples were drawn using NUTS(diag_e) at Sun Mar 17 22:29:15 2019.
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).
```

检查自相关

stan_ac(fit, pars=c(”e”, ”l”))



转换点的估计

```r
print(fit, pars=c("switchpoint"))
```

```
Inference for Stan model: ch6_cp_model.
4 chains, each with iter=5000; warmup=2000; thin=1;
post-warmup draws per chain=3000, total post-warmup draws=12000.

             mean se_mean  sd 2.5% 25% 50% 75% 97.5% n_eff Rhat
switchpoint 42.83       0 0.4   42  43  43  43    43 11863    1

Samples were drawn using NUTS(diag_e) at Wed Mar 20 23:27:04 2019.
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).
```

4.4 转换点模型的 Gibbs 抽样器算法应用

美军在阿富汗死亡人数的转换点模型

- 早期数据：$D_1, D_2, \ldots, D_k | \lambda \sim \text{Pois}(\lambda)$
- 后期数据：$D_{k+1}, D_{k+2}, \ldots, D_n | \phi \sim \text{Pois}(\phi)$

先验分布假设：

$$\begin{aligned}
&\lambda \sim \text{gamma}(\alpha, \beta) \\
&\phi \sim \text{gamma}(\gamma, \delta) \\
&k \sim \text{discrete uniform on } \{1, 2, ..., n\}
\end{aligned}$$

- 前 40 个均值为 4.95，后 12 个均值为 10.66
- $\alpha = 1, \beta = 0.2; \gamma = 4, \delta = 0.4$
- Gibbs 抽样需要推导出条件后验分布

转换点模型参数的条件后验分布

后验分布：

$$\begin{aligned}
p(\lambda, \phi, k | x) &\propto L(\lambda, \phi, k | x) \pi(\lambda) \pi(\phi) \pi(k) \\
&= \left[\prod_{i=1}^{k} \frac{\lambda^i e^{-\lambda}}{x_i!} \right] \left[\prod_{i=k+1}^{n} \frac{\phi^i e^{-\phi}}{x_i!} \right] \left[\frac{\beta^\alpha \lambda^{\alpha-1} e^{\beta\lambda}}{\Gamma(\alpha)} \right] \left[\frac{\delta^\gamma \phi^{\gamma-1} e^{\delta\phi}}{\Gamma(\gamma)} \right] \left[\frac{1}{n} \right] \\
&\propto \lambda^{\alpha + \sum_{i=1}^{k} x_i - 1} e^{-(\beta+k)\lambda} \phi^{\gamma + \sum_{i=k+1}^{n} x_i - 1} e^{-(\delta+n-k)\phi}
\end{aligned}$$

全部条件后验分布（full conditionals）：

$$\lambda|\phi,k\sim\text{gamma}(\alpha+\sum_{i=1}^{k}x_i,\beta+k)$$
$$\phi|\lambda,k\sim\text{gamma}(\gamma+\sum_{i=k+1}^{n}x_i,\delta+n-k)$$

转换点的条件后验分布

k 的条件后验分布：

$$\begin{aligned}f(\boldsymbol{x}|\lambda,\phi,k)&=\left[\prod_{i=1}^{k}\frac{\lambda^i e^{-\lambda}}{x_i!}\right]\left[\prod_{i=k+1}^{n}\frac{\phi^i e^{-\phi}}{x_i!}\right]\\&=\left[\prod_{i=1}^{n}\frac{\phi^{x_i} e^{-\phi}}{x_i!}\right]\left[e^{k(\phi-\lambda)}\left(\frac{\lambda}{\phi}\right)^{\sum_{i=1}^{k}x_i}\right]\\&=h(\boldsymbol{x},\phi)L(\boldsymbol{x}|k)\end{aligned}$$

第一个函数与 k 无关，以第二个函数为 k 的似然函数，设先验为 $p(k)=1/n$，对任意一个 k^*，根据贝叶斯定理，

$$\begin{aligned}p(k^*|\boldsymbol{x},\lambda,\phi)&=\frac{h(\boldsymbol{x},\phi)L(\boldsymbol{x}|k^*)p(k^*)}{\sum_{l=1}^{n}h(\boldsymbol{x},\phi)L(\boldsymbol{x}|l)p(l)}\\&\propto\frac{L(\boldsymbol{x}|k^*)}{\sum_{l=1}^{n}L(\boldsymbol{x}|l)}\end{aligned}$$

上式给出了每次迭代中，k 在 $\{1,2,\ldots,n\}$ 中取值的概率。

Gibbs Sampler Algorithm

1. 给定初值：$\boldsymbol{\theta}^{(0)}=(\lambda^{(0)},\phi^{(0)},k^{(0)})$
2. 依次抽取 $\boldsymbol{\theta}^{(i)}=(\lambda^{(i)},\phi^{(i)},k^{(i)})$：

$$\begin{aligned}\lambda^{(i)}&\sim\text{gamma}(\alpha+\sum_{l=1}^{k^{(i-1)}}x_l,\beta+k^{(i-1)})\\\phi^{(i)}&\sim\text{gamma}(\gamma+\sum_{l=k^{(i-1)}+1}^{n}x_l,\delta+n-k^{(i-1)})\\\text{for}&\quad(j\text{ in }1:n)\\&\quad p_j=e^{j(\phi^{(i)}-\lambda^{(i)})}\left(\frac{\lambda^{(i)}}{\phi^{(i)}}\right)^{\sum_{l=1}^{j}x_l}\\&\quad p_j^{(i)}=\frac{p_j}{\sum_{l=1}^{n}p_l}\\k^{(i)}&=\text{sample}(1:n,size=1,prob=(p_1^{(i)},\ldots,p_n^{(i)}))\end{aligned}$$

转换点模型 Gibbs 迭代抽样的 R 函数

```
gibss.cp <- function(theta.matrix,y,a,b,g,d) {
  n <- length(y)
  k.prob <- rep(0,times=n)
  for (i in 2:nrow(theta.matrix)) {
    lambda <- rgamma(1,a+sum(y[1:theta.matrix[(i-1),3]]),
                     b+theta.matrix[(i-1),3])
    phi <- rgamma(1,g+sum(y[theta.matrix[(i-1),3]:n]),
                    d+n-theta.matrix[(i-1),3])
    for (j in 1:n) {
       k.prob[j] <- exp(j*(phi-lambda))*(lambda/phi)^sum(y[1:j])
       }
    k.prob <- k.prob/sum(k.prob)
    k <- sample(1:n,size=1,prob=k.prob)
    theta.matrix[i,]<-c(lambda,phi,k) #storing the set of parameters
  }
 return(theta.matrix)
}
```

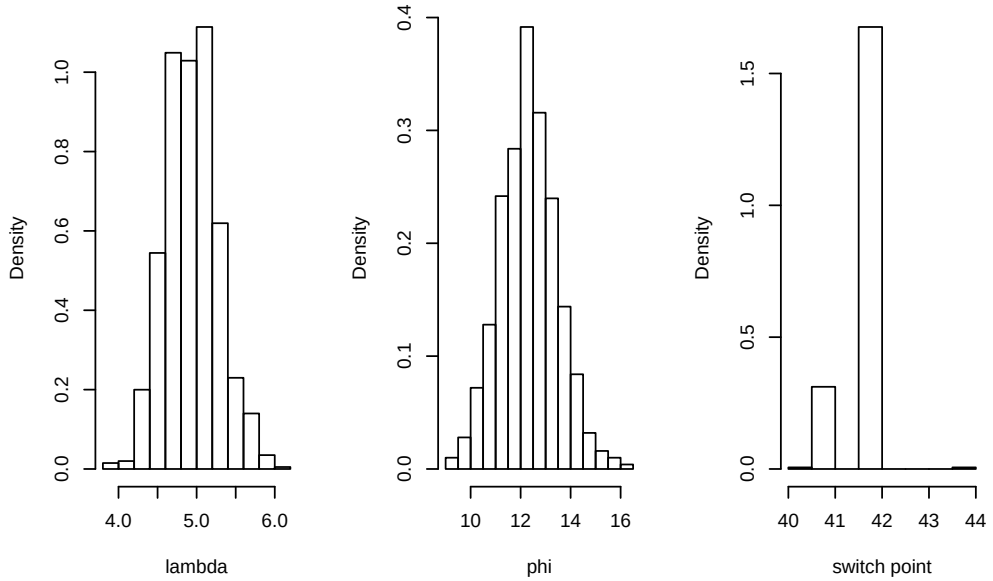
转换点模型 Gibbs 抽样的 R 代码（续）

```
library(BaM)
data(afghan.deaths)
y <- afghan.deaths
my.alpha <- 1; my.beta <- 0.2
my.gamma <- 4; my.delta <- 0.4
tot.draws <- 2000 # Total number of samples
# initial values for (lambda, phi, k)
init.param.values <- c(5.95, 10.66, 40)
init.theta.matrix <- matrix(0, nrow=tot.draws, ncol=3)
init.theta.matrix <- rbind(init.param.values,init.theta.matrix)
my.Gibbs.samples <- gibss.cp(init.theta.matrix, y=y,a=my.alpha,
                             b=my.beta, g=my.gamma, d=my.delta)
# remove first 1000 sampled values as "burn-in":
my.post <- my.Gibbs.samples[-(1:1000),]
```

转换点模型 Gibbs 抽样的 R 代码：运行结果

```
> summary(my.post)
       V1              V2              V3
 Min.   :3.573   Min.   : 8.073   Min.   :39.00
 1st Qu.:4.650   1st Qu.:11.464   1st Qu.:42.00
 Median :4.873   Median :12.201   Median :42.00
 Mean   :4.882   Mean   :12.224   Mean   :41.85
 3rd Qu.:5.106   3rd Qu.:12.964   3rd Qu.:42.00
 Max.   :6.321   Max.   :16.983   Max.   :44.00

# Posterior medians for lambda, phi and k
> apply(my.post,2,median)
[1]  4.872718 12.201327 42.000000
```

转换点模型 Gibbs 抽样的 R 代码：直方图



结果比较

```
WinBUGS Model:
lambda; phi; changeyear
4.89; 11.85; 42.33
Stan Model:
4.90; 11.83; 42.83
Gibbs Sampler:
4.90; 12.37; 41.86
```

5 课堂讨论

例 1 汽车油耗问题

针对例 1 的汽车油耗问题，

1. 建立贝叶斯模型；
2. 写出对应 Stan 代码，运用 Rstan 进行求解，并验证收敛性；
3. 把误差项改为 t 分布，重新运行 RStan 并比较运行结果。