Three Examples of Markov Chains and Gibbs Sampling

Gui Larangeira, California State University East Bay, glarangeira@horizon.csueastbay.edu

Keywords: Stochastic Processes, Simulations, Bayesian Inference, Gibbs Sampling, Financial Applications, Markov Chain Monte Carlo methods, R Statistical Software, WinBUGS

Abstract: We illustrate the use of Markov Chains and Gibbs Sampling in modelling stochastic processes with three simple applications: (a) Obtaining the distribution of returns of a portfolio of two correlated stocks (modeled as normal distributions) (b) Using Bayesian Inference (a Beta Binomial conjugate pair model) to estimate the prevalence of spam email and (c) Fitting a t-distribution to the monthly returns of a stock (using WinBUGS).

1. Introduction

A Stochastic Process can be defined as a sequence of random variable "steps", $X_1, X_2, ., X_n$, where the indexes represent the discrete steps. A Markov Chain is a model for particular Stochastic Processes that assumes single-step dependency. A Gibbs Sampler is a simulated Markov Chain with a limiting distribution that targets a desired target distribution. Especially in Bayesian Inference, Gibbs Sampling is a computational tool used where an analytical solution isn't readily available.In our first example we illustrate a Gibbs Sampler without Bayesian Inference. In the second example, we illustrate the Gibbs Sampler with a Bayesian Model and finally in our third example we use WinBUGS to fit a t-distribution, a problem that is challenging to do analytically.

2. A Bivariate Correlated Normal Model for Simple Two-Stock Portfolio

We assume two stocks' monthly returns X and Y are standard normals with known correlation ρ. Financial assets are often correlated (with the correlation often called β in financial literature). Then the overall return of the portfolio can be modeled as the *bivariate* $W = aX + bY$, where a and b are the weights of each stock in the portfolio. The conditional distributions are given by:

$$X \sim Norm(\rho Y, \sigma)$$
$$Y \sim Norm(\rho X, \sigma)$$

where

$$\sigma = \sqrt{1 - \rho^2}$$

Also, to model $X_1 \sim Norm(\alpha_1, \beta_1^2)$ we can consider a linear transformation $X_1 = \alpha_1 X + \beta_1$.

We can then use these relationships to build a Markov Model and Gibbs sampling method as show next. In this initial example, no Bayesian Inference is required, just sampling from the target distributions through a Gibbs Sampler.

The R program that implements this model is shown below:

```
set.seed(1212)
m = 100000; x = y = numeric(m); x[1] = y[1] = 0  # initial conditions
rho = 0.8; sgm = sqrt(1 - rho^2)                 # correlation and standard deviation

for (i in 2:m) {                                 # Gibbs Sampler Loop
  x[i] = rnorm(1, rho*y[i-1], sgm)
  y[i] = rnorm(1, rho*x[i], sgm)
}
```

```r
alpha.1 = beta.1 = .01
alpha.2 = .015; beta.2 = .012
x1 = alpha.1 +beta.1*x          # x1 ~ N(alpha.1, beta.1)
y1 = alpha.2 +beta.2*y          # x2 ~ N(alpha.2, beta.2)

c(mean(x), mean(y)); c(mean(x1),mean(y1))
```

```
## [1] -0.006323863 -0.004893309
```

```
## [1] 0.009936761 0.014941280
```

```r
c(sd(x),sd(y)); c(sd(x1),sd(y1))
```

```
## [1] 0.9977258 0.9993164
```

```
## [1] 0.009977258 0.011991797
```

```r
c(cor(x,y),cor(x1,y1))          # check that mus, sigma and rhos are as planned
```

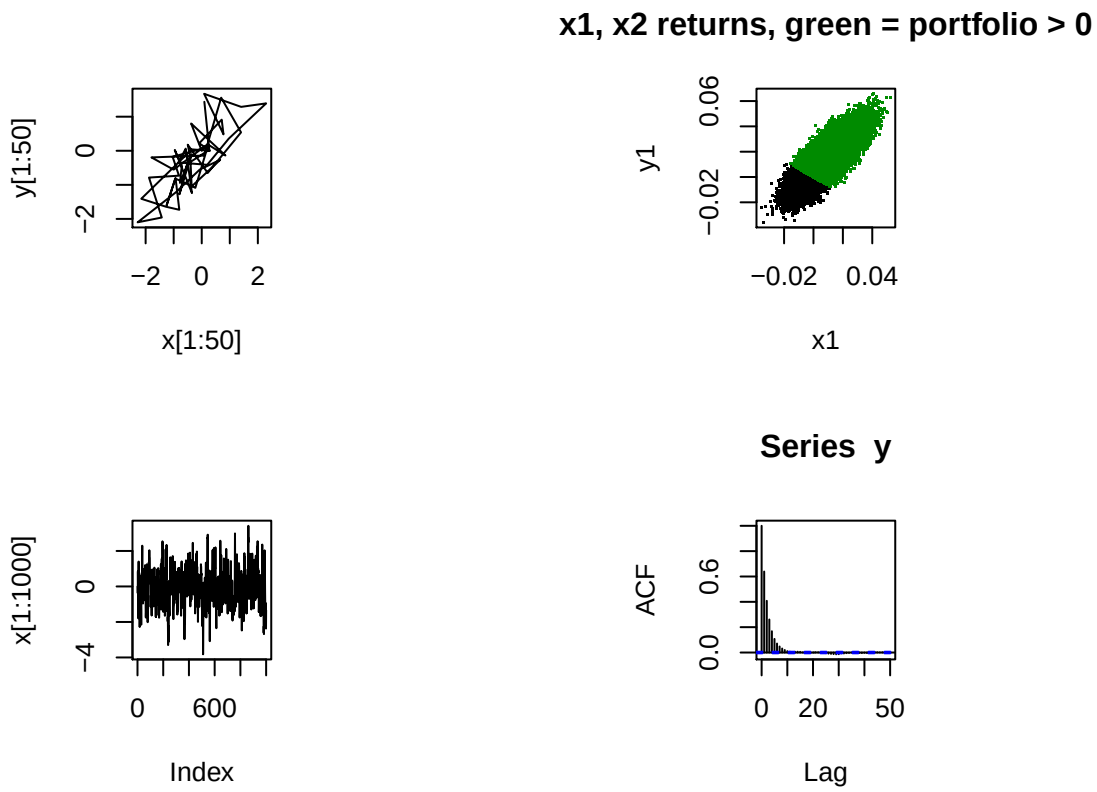
```
## [1] 0.7982824 0.7982824
```

```r
a= .4; b= .6                    # Portfolio weights
w = a*x1 + b*y1; mean(w); sd(w) # Overall portfolio returns
```

```
## [1] 0.01293947
```

```
## [1] 0.01065559
```

```r
# plots include the burn-in period

par(mfrow=c(2,2), pty="s") # 2 x 2 array of square plots
plot(x[1:50], y[1:50], type="l")
plot(x1, y1, pch=".", main = "x1, x2 returns, green = portfolio > 0");points(x1[w>0], y1[w>0], pch=".",
plot(x[1:1000], type="l"); acf(y)
```




```
par(mfrow=c(1,1), pty="m") # back to default, single rectangular plot
```

We calculate the average return and standard deviation (volatility) of the portfolio as $\mu_w = .0129$ and $\sigma_w = .0106$. For validation, the X and Y are very close to standard normals, and ρ very close to 0.8 as intended. Moreover, graphical diagnostics seem to be good indicators as the chain is "mixing well". That is, the simulated values X_i and Y_i seem to move freely, taking reasnobale values, as shown on left side of the 2X2 plot.

3. Bayesian Inference and Gibbs Sampler for Spam Filter

We consider a spam filter and use a Bayesian Inference model to describe the process. Our goal is to estimate the prevalence π of spam without having to directly count the instances of spam. Our model is thus represented:

Events:

$\{S = 1\}$: Email is in fact Spam. $\{S = 0\}$: Email is not Spam

$\{R = 1\}$: Email is *marked* as Spam. $\{R = 0\}$ Email is *marked* as NOT Spam

Parameters and definitions:

Sensitivity: $\eta = P\{R = 1|S = 1\} = 0.90$:

Specificity: $\theta = P\{R = 0|S = 0\} = 0.95$

Prevalence: $\psi = P\{S = 1\}$. Then by the fundamental laws of probability, we can write:

$\tau = P\{R = 1\} = P\{R = 1, S = 1\} + P\{R = 1, S = 0\} = \psi\eta + (1 - \psi)(1 - \theta)$

3.1 Frequentist estimates for ψ and τ

It is easy to count the number of rejected emails to estimate the probability τ. However, it would be a lot more labour-intensive to count the number of spam e-mails directly. So we can rewrite the equation above to solve for prevalence ψ, which is usually the parameter of interest:

$$\psi = \frac{\tau + \theta - 1}{\eta + \theta - 1}$$

Let's assume for sake of example that we mark 233 emails as spam out of total of 1000 emails. In R code, this would translated as:

```
# Assumptions and Parameters
# {S=1} email is spam; {S=0} not spam
# {R=1} email blocked; {R=0} passed

eta = 0.90;     # Sensitivity
theta = 0.95;   # Specificity
r = 233;        # Rejected Emails
n = 1000;       # Total number of emails
tau = r/n       # Estimate of rejection rate P{R=1}
print(tau)
```

```
## [1] 0.233
```

```
psi = (tau + theta - 1)/(eta + theta -1)
psi
```

```
## [1] 0.2152941
```

We can then build traditional frequentist CI as $\tau \pm 1.96\sqrt{\tau(1-\tau)/n}$. Again, in R, this is

```
tau.l = tau - 1.96 * sqrt(tau*(1-tau)/n)
tau.u = tau + 1.96 * sqrt(tau*(1-tau)/n)
round(c(tau.l, tau.u),3)
```

```
## [1] 0.207 0.259
```

```
# Taking the tau endpoints and "transforming" them into the the psi interval through the psi equation a
psi.l = (tau.l + theta - 1)/(eta + theta -1)
psi.u = (tau.u + theta - 1)/(eta + theta -1)
round(c(psi.l, psi.u),3)
```

```
## [1] 0.184 0.246
```

In many situations, this procedure for getting point and interval estimates of works well. However, especially when n is relatively small, there are surprisingly many cases in which the displayed formula yields nonsensical point estimates of outside the interval (0, 1) for ψ. For example, consider the example:

```
eta = 0.99; theta = 0.97;
r = 5;      n = 250;
tau = r/n ; tau
```

```
## [1] 0.02
```

```
psi = (tau + theta - 1)/(eta + theta -1)
psi
```

```
## [1] -0.01041667
```

And we see that the probability psi arrived at is non-valid number.

3.2 Applying the Bayesian Method to estimating τ

To introduce the Bayesian Inference Method, we begin by a straighforward example where Gibbs Sampling is not required. We want to estimate *tau* and we know from Bayes Equation that

$$Posterior \propto Prior \times Likelihood$$

Now we begin with a non-informative prior, $U(0,1)$ or $Beta(\alpha_0 = 1, \beta_0 = 1)$. So the Posterior in this context can be written as

$$p(\tau|r) \propto p(\tau) \times p(r|\tau)$$

We begin with a non-informative prior $Beta(\alpha_0 = 1, \beta_0 = 1)$ for τ. So the posterior for τ can be written as

$$\tau \sim Beta(\alpha_0 = 1, \beta_0 = 1)$$
$$p(\tau|r) \propto p(\tau) \times p(r|\tau)$$

$$\propto \tau^{\alpha_0 - 1}(1-\tau)^{\beta_0 - 1} \times \tau^r(1-\tau)^{n-r}$$

$$= \tau^{\alpha_0 + r - 1}(1-\tau)^{beta_0 + n - r - 1}$$

where the right-hand side reveals itself readily as the density function of $Beta(\alpha_n, \beta_n)$. This functional "compatibility" between the *Beta* (prior and posterior) and *Binomial* (likelihood) families in Bayesian Inference is what leads them to be considered a *conjugate pair*. Then If $n = 1000$ and $r = 233$ we define

$$\alpha_n = \alpha_0 + r = 1 + 233 = 234$$
$$\beta_n = \beta_0 + n - r = 1 + 1000 - 233 = 768$$

We can thus use the resulting Beta density posterior for $p(\tau|r)$ to derive the mean and CI for τ. In R:

```
alpha.0 = 1; beta.0 = 1       # Prior Information
n = 1000; r = 233             # Data

alpha.n = alpha.0 +r            # Posterior parameters
beta.n = beta.0 + n - r

alpha.n/(alpha.n+beta.n)      # Posterior density mean
```

```
## [1] 0.2335329
```

```
qbeta(c(.025,.975),alpha.n,beta.n) # CI for the posterior density mean
```

```
## [1] 0.2078629 0.2602105
```

3.3 Applying the Bayesian Method to estimating prevalence ψ

Next we use a Gibbs sampler to obtain point and interval estimates for ψ. Unlike the estimate for τ (where we were able to count r the spam-marked emails), we need a Gibbs sampler because it is not feasible to directly count the spam emails.

$$p = P\{S = 1|R = 1\} = P\{S = 1, R = 1\}/P\{R = 1\}$$

$$p = \psi\eta/\tau$$

and

$$q = P\{S = 1|R = 0\} =$$

$$\psi(1-\eta)/(1-\tau)$$

Let X be the number among the r rejected emails that are indeed spam and Y be the number of spam emails among the n-r emails passed through. We can simulate these *latent counts* X and Y using

$$X|r, \psi \sim Binom(r, p)$$

$$Y|r, \psi \sim Binom(n-r, q)$$

$$\psi|X, Y \sim Beta(\alpha_n, \beta_n)$$

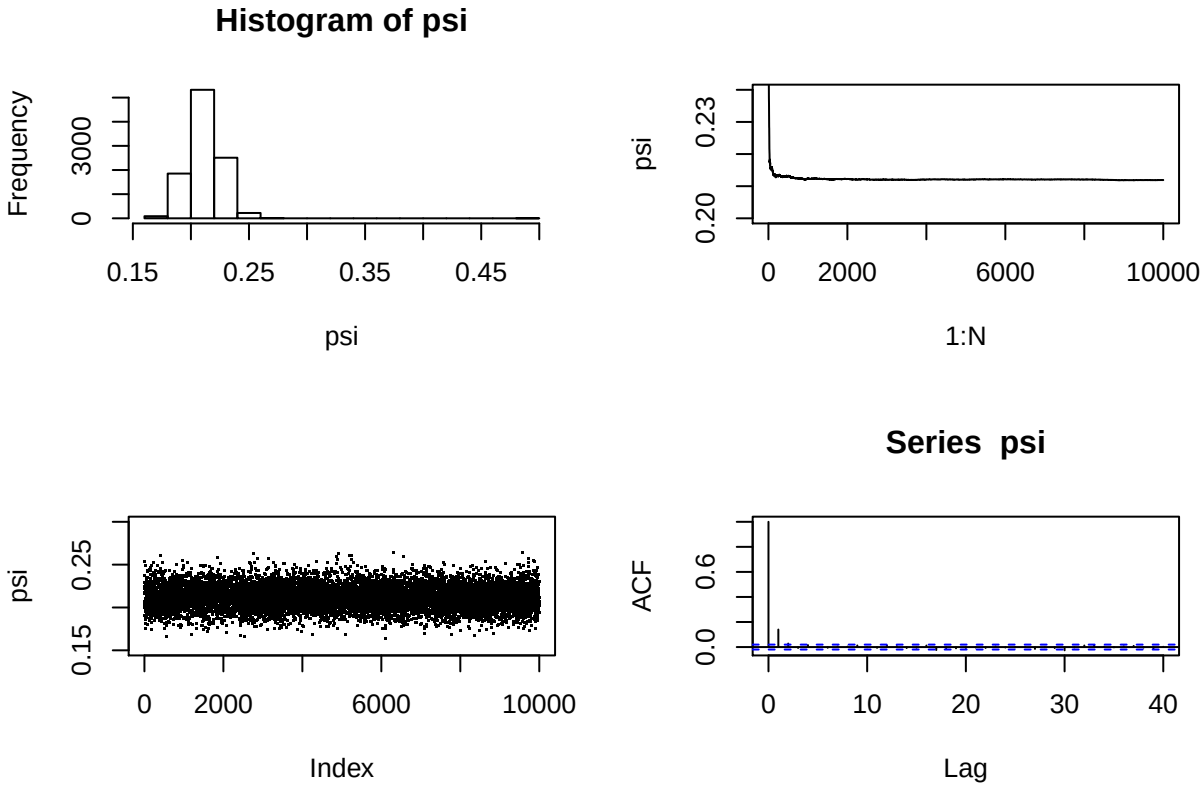
where we consider $V = X + Y$, and $\alpha_n = alpha_0 + V$ and $beta_n = beta_0 + n - V$.

We use one value of ψ to find the next in this iterative manner. This is in effect a Markov Chain for which the limiting distribution is the posterior of ψ. At each iteration, we use the prior distribution and the data. The continued and recursive use of this information will cause the convergence of the simulated values toward the appropriate posterior distribution. We illustrate this Gibbs Sampler method below:

```
N = 10000             # Number of iterations
Nb = 2000; N1 = N+1   # Burn-in

psi = numeric(N)
psi[1] = .5           # Initial value

                      # Gibbs Sampler Loop
for(i in 2:N) {
tau=psi[i-1]*eta+(1-psi[i-1])* (1-theta)
X = rbinom(1, r, psi[i-1]* eta/tau)
Y = rbinom(1, n-r, psi[i-1]*(1-eta)/(1-tau))
psi[i] = rbeta(1, alpha.0+X+Y, beta.0+n-X-Y)
}

mean(psi[Nb:N])
```

```
## [1] 0.2118245
```

```
par(mfrow=c(2,2))
hist(psi)
plot(1:N,cumsum(psi)/(1:N),type="l",ylab= "psi", ylim=c(0.20,0.24))
plot(psi,type='p',pch='.',ylim=c(0.15,0.30))
acf(psi)
```



Again visual diagnostics (the bottom psi[i] and ACF plots) show proper mixing and Markov model assumptions are respected. Also the psi[i] seem to converge to a reasonable value. Further, we can obtain an empirical Bayesian credible interval by looking at the post burn-in psi[i]:

3.4 Estimating a Bayesian Credible Interval for ψ

```
psi.sort <- sort(psi)
L = as.integer(.05/2*(N-Nb))
U = as.integer((1-.05/2)*(N-Nb))

psi.L = psi.sort[L]
psi.U = psi.sort[U]
round(c(psi.L,psi.U),3)
```

```
## [1] 0.184 0.222
```

And we can see that this interval contains the point value estimated above through the Gibbs Sampler.

4. Fitting stock return data to t-distribution with WinBUGS

In Section 1 we modeled monthly stock returns as Normal. Here we use the more realistic t-distribution, which more accurately describes the outliers frequently found in stock data ("fat tails", kurtosis). We use the popular WinBUGS (Bayesian analysis Using Gibbs Sampling) software:

Below we call the WinBUGS/OpenBUGS libraries from within R:

```
library(R2WinBUGS); library(R2OpenBUGS)
```

```
## Loading required package: coda
```

```
## Loading required package: boot
```

```
##
## Attaching package: 'boot'
```

```
## The following object is masked _by_ '.GlobalEnv':
##
##     tau
```

```
##
## Attaching package: 'R2OpenBUGS'
```

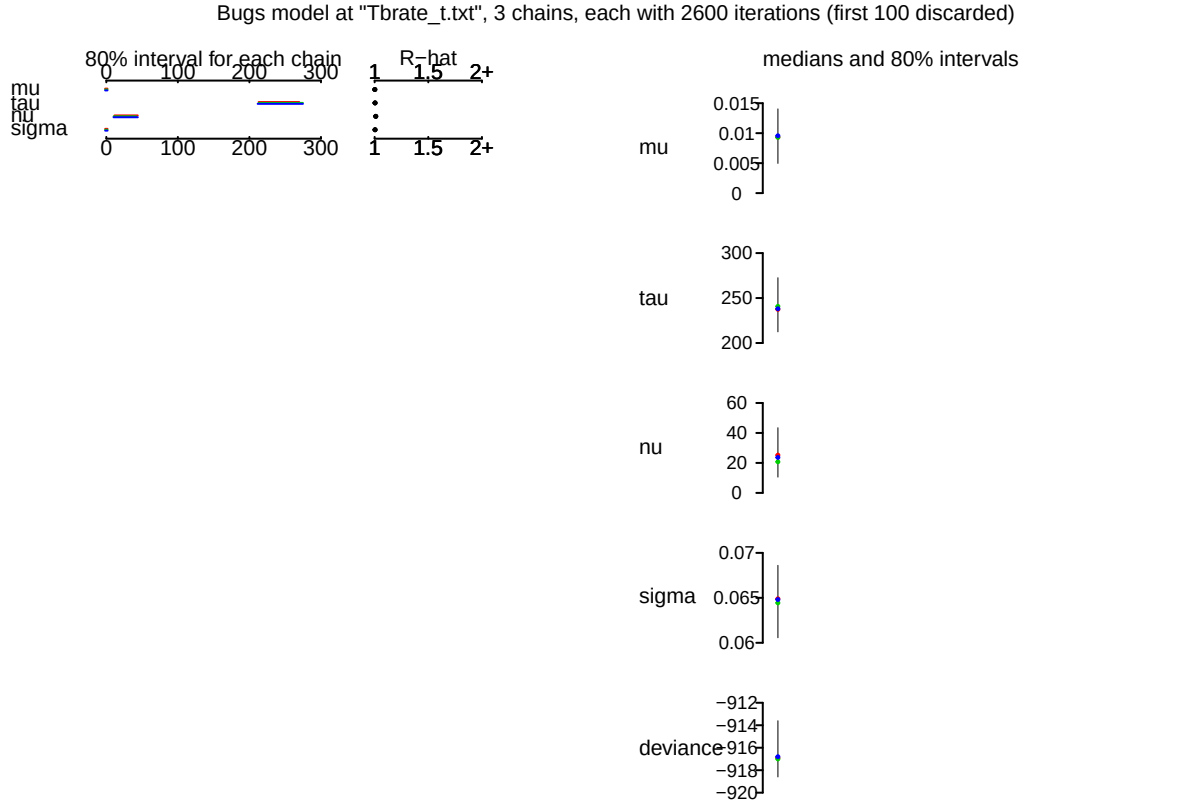
```
## The following objects are masked from 'package:R2WinBUGS':
##
##     as.bugs.array, attach.all, attach.bugs, bugs, bugs.data,
##     bugs.log, detach.all, detach.bugs, monitor, read.bugs,
##     write.model
```

```
data(CRSPmon, package="Ecdat")
ibm = CRSPmon[,2]               # loads ibm and y as ibm returns
y= as.numeric(ibm)
N=length(y)
ibm_data=list("y","N")

inits=function(){ list(mu=rnorm(1,0,.3),tau=runif(1,1,10),nu=runif(1,1,30))}
univt.mcmc = bugs(ibm_data,inits,model.file="Tbrate_t.txt",
                  parameters=c("mu","tau","nu","sigma"),
                  n.chains=3,n.iter=2600,n.burnin=100,
                  n.thin=1,
                  bugs.seed=10)
print(univt.mcmc,digits=4)
```

```
## Inference for Bugs model at "Tbrate_t.txt",
## Current: 3 chains, each with 2600 iterations (first 100 discarded)
## Cumulative: n.sims = 7500 iterations saved
##               mean      sd      2.5%       25%       50%       75%
## mu          0.0094  0.0035    0.0027    0.0071    0.0094    0.0118
## tau       240.9409 24.3706  199.4000  224.0000  238.7000  255.0000
## nu         25.0819 12.0561    7.3279   14.7775   23.1500   34.7500
## sigma       0.0647  0.0032    0.0581    0.0626    0.0647    0.0668
## deviance -916.4080  2.1635 -919.1000 -917.9000 -916.9000 -915.5000
##              97.5%   Rhat n.eff
## mu          0.0165 1.0020  1600
## tau       295.9050 1.0042   770
## nu         48.2152 1.0107   200
## sigma       0.0708 1.0042   770
## deviance -910.7000 1.0042   780
##
## 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 = 2.0350 and DIC = -914.4000
```

```
## DIC is an estimate of expected predictive error (lower deviance is better).
```

```
plot(univt.mcmc)
```

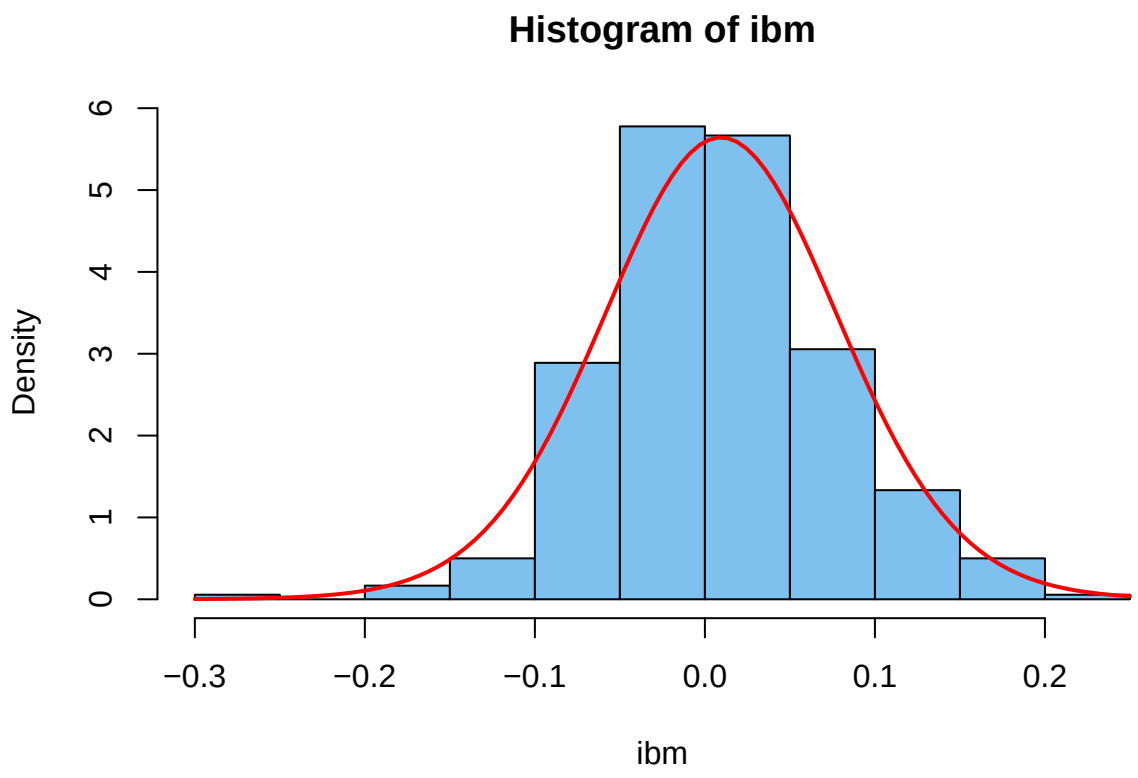



```
# WinBUGS Model Tbrate_t:

# model{
# for(i in 1:N){
# y[i]~dt(mu,tau,nu)      Defines y as a t distribution with parameter mu and...
# }
# mu ~ dnorm(0.0,1.0E-6)  ... Prior mu (mean)
# tau ~ dgamma(0.1,.01)   ... Prior tau (1/scale^2)
# sigma<-1/sqrt(tau)      ... Scale
# nu~dunif(2,50)          ... Prior of degrees of freedom
# }


x=seq(from = -0.3, to =0.3, by = .01)
hist(ibm, freq = FALSE,  col="skyblue2")
library(metRology)
```

```
##
## Attaching package: 'metRology'
```

```
## The following objects are masked from 'package:base':
##
##     cbind, rbind
```

```r
x=seq(from = -0.3, to =0.3, by = .01)
hist(ibm, freq = FALSE,  col="skyblue2")
# curve(dt(x, .0094, 25)*240, lwd=2, col="red", add=T)

curve(dt.scaled(x, df = 25, mean = 0.0094, sd = .07,ncp=0, log = FALSE), lwd=2, col="red", add=T)
```




```r
# Need to correctly plot the t with mu = .0094 and df = 25 over the histogram.
```

5. Bibliography:

[1] Trumbo, B. & Suess, E., PSGS, Springer Verlag, 2010

[2] Trumbo & Suess, Entry in Encyclopedia of Social Network Analysis and Mining

[3] Ruppert, D., Statistics and Data Analysis for Financial Engineering, Springer Verlag 2011

[4] WinBUGS Manual