GLMM, Concepts, & R

Bill Last Updated:

19 January, 2020

Contents

Preface: Motivation

All the notes I have done here are the preparation for my stat master project, which will be about Generalized Linear Mixed Models. While I have tried my best, probably there are still some typos and errors. Please feel free to let me know in case you find one. Thank you!

Chapter 1

Basics

1.1 Logit

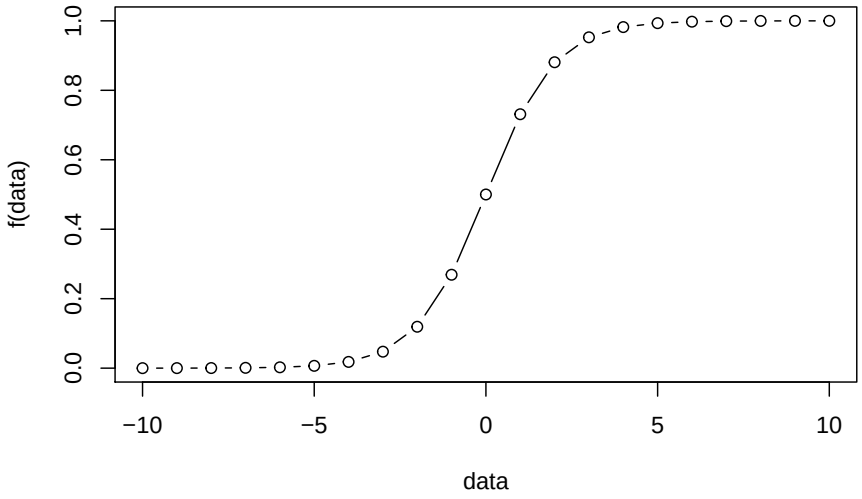
$$f(x) = log(\frac{p(y=1)}{1-p(y=1)})$$

The basic idea of logistic regression:

$$p(y=1) = \frac{1}{1+e^{-(\beta_0+\beta_1 x_1+...+\beta_n x_n)}} = \frac{e^{\beta_0+\beta_1 x_1+...+\beta_n x_n}}{1+e^{\beta_0+\beta_1 x_1+...+\beta_n x_n}}$$

Thus, $e^{\beta_0+\beta_1 x_1+...+\beta_n x_n}$ can be from $-\infty$ to $+\infty$, and $p(y=1)$ will be always within the range of $(0,1)$.

```
f<-function(x){exp(x)/(1+exp(x))}
data<-seq(-10,10,1)
plot(data,f(data),type = "b")
```



We can also write the function into another format as follows:

$$log\frac{p(y=1)}{1-p(y=1)} = \beta_0 + \beta_1 x_1 + ... + \beta_n x_n$$

Thus, we know that the regression coeficients of β_i actually change the "log-odds" of the event. Of course, note that the magnitude of β_i is dependent upon the units of x_i.

The following is an example testing whether that home teams are more likely to win in NFL games. The results show that the odd of winning is the same for both home and away teams.

```
mydata = read.csv(url('https://raw.githubusercontent.com/nfl-football-ops/Big-Data-Bowl
mydata$result_new<-ifelse(mydata$HomeScore>mydata$VisitorScore,1,0)
summary(mydata$result_new)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.0000  0.0000  0.0000  0.4945  1.0000  1.0000
```

```
mylogit1 = glm(result_new~1, family=binomial, data=mydata)
summary(mylogit1)
```

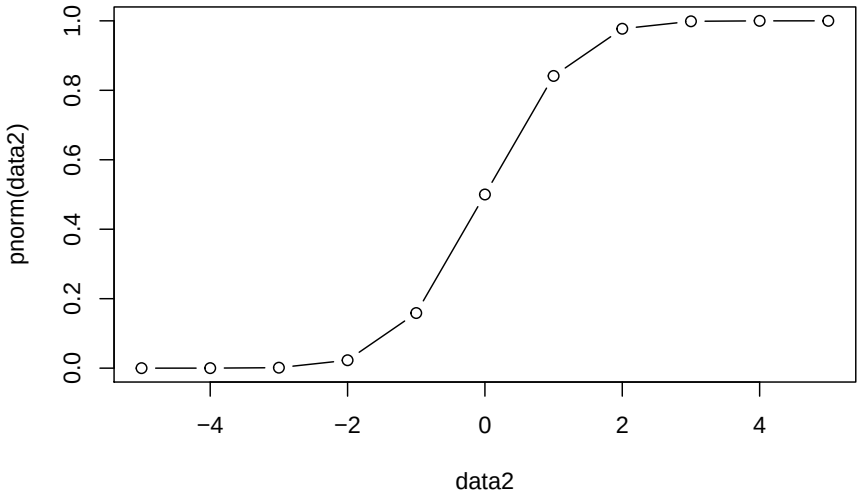
```
##
## Call:
```

```
## glm(formula = result_new ~ 1, family = binomial, data = mydata)
##
## Deviance Residuals:
##    Min      1Q  Median      3Q     Max
## -1.168  -1.168  -1.168   1.187   1.187
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept) -0.02198    0.20967  -0.105    0.917
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 126.14  on 90  degrees of freedom
## Residual deviance: 126.14  on 90  degrees of freedom
## AIC: 128.14
##
## Number of Fisher Scoring iterations: 3
```

1.2 Probit

As noted above, logit $f(x) = log(\frac{p(y=1)}{1-p(y=1)})$ provides the resulting range of $(0, 1)$. Another way to provide the same rage is through the cdf of normal distribution.The following R code is used to illusrate this process.

```
data2<-seq(-5,5,1)
plot(data2,pnorm(data2),type = "b")
```



Thus, the cdf of normal distribution can be used to indicate the probability of $p(y = 1)$.

$$\Phi(\beta_0 + \beta_1 x_1 + ... + \beta_n x_n) = p(y = 1)$$

Similar to logit model, we can also write the inverse function of the cdf to get the function that can be from $-\infty$ to $+\infty$.

$$\beta_0 + \beta_1 x_1 + ... + \beta_n x_n = \Phi^{-1}(p(y = 1))$$

Thus, for example, if $X\beta$ = -2, based on $\Phi(\beta_0 + \beta_1 x_1 + ... + \beta_n x_n) = p(y = 1)$ we can get that the $p(y = 1) = 0.023$.

In contrast, if $X\beta = 3$, the $p(y = 1) = 0.999$.

```
pnorm(-2)
```

```
## [1] 0.02275013
```

```
pnorm(3)
```

```
## [1] 0.9986501
```

Let's assume that there is a latent variable called Y^* such that

$$Y^* = X\beta + \epsilon, \epsilon \sim N(0, \sigma^2)$$

You could think of Y^* as a kind of "proxy" between $X\beta + \epsilon$ and the observed $Y(1 or 0)$. Thus, we can get the following. Note that, it does not have to be zero, and can be any constant.

$$Y^* = \begin{cases} 0 & if\ y_i^* \leq 0 \\ 1 & if\ y_i^* > 0 \end{cases}$$

Thus,

$$y_i^* > 0 \Rightarrow \beta^{'} X_i + \epsilon_i > 0 \Rightarrow \epsilon_i > -\beta^{'} X_i$$

Thus, we can write it as follows. Note that $\frac{\epsilon_i}{\sigma} \sim N(0, 1)$

$$p(y = 1|x_i) = p(y_i^* > 0|x_i) = p(\epsilon_i > -\beta^{'} X_i) = p(\frac{\epsilon_i}{\sigma} > \frac{-\beta^{'} X_i}{\sigma}) = \Phi(\frac{\beta^{'} X_i}{\sigma})$$

We thus can get:

$$p(y = 0|x_i) = 1 - \Phi(\frac{\beta^{'} X_i}{\sigma})$$

For $p(y = 1|x_i) = \Phi(\frac{\beta^{'} X_i}{\sigma})$, we can not really estimate both β and σ as they are in a ratio. We can assume $\sigma = 1$, then $\epsilon \sim N(0, 1)$. We know y_i and x_i since we observe them. Thus, we can write it as follows.

$$p(y = 1|x_i) = \Phi(\beta^{'} X_i)$$

Chapter 2

LM and GLM

Before moving to LMM, I would like to review LM and GLM first.

2.1 LM

$$Y|X \sim N(\mu(X), \sigma^2 I)$$

$$E(Y|X) = \mu(X) = X^T\beta$$

where,

$\mu(X)$: $randomcomponent$

$X^T\beta$: $covariates$

2.2 GLM-Definition

Ref: https://ocw.mit.edu/courses/mathematics/18-650-statistics-for-applications-fall-2016/lecture-slides/MIT18_650F16_GLM.pdf

$$Y \sim exponentialfamily$$

Link function

$$g(\mu(X)) = X^T\beta$$

2.3 GLM-log link example

$$\mu_i = \gamma e^{\delta t_i}$$

Link function is log link, and it becomes:

$$log(\mu_i) = log(\gamma) + log(\delta t_i) = \beta_0 + \beta_1 t_i$$

(This is somehow similar to Poisson distribution.)

2.4 GLM-Reciprocal link:

$$\mu_i = \frac{\alpha x_i}{h + x_i}$$

Reciprocal link:

$$g(\mu_i) = \frac{1}{\mu_i} = \frac{1}{\alpha} + \frac{h}{\alpha}\frac{1}{x_i} = \beta_0 + \beta_1 \frac{1}{x_i}$$

2.5 GLM-exponential family:

In a more general sense, for exponential family:

$$\begin{aligned} P_\theta(X) = P(X,\theta) &= e^{\sum \eta_i(\theta)T_i(X)} C(\theta)h(x) \\ &= e^{\sum \eta_i(\theta)T_i(X)} e^{-log(\frac{1}{c(\theta)})} h(x) \\ &= e^{\sum \eta_i(\theta)T_i(X) - log(\frac{1}{c(\theta)})} h(x) \\ &= e^{\sum \eta_i(\theta)T_i(X) - B(\theta)} h(x) \end{aligned}$$

Normal distribution

For normal distributions, it belongs to exponential family.

$$\begin{aligned} P_\theta(X) &= \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\frac{(x-\mu)^2}{\sigma^2}} \\ &= e^{-\frac{1}{2}\frac{(x-\mu)^2}{\sigma^2}} e^{log(\frac{1}{\sigma\sqrt{2\pi}})} \\ &= e^{-\frac{1}{2}\frac{(x-\mu)^2}{\sigma^2} - log(\sigma\sqrt{2\pi})} \\ &= e^{-\frac{1}{2\sigma^2}x^2 - \frac{1}{2\sigma^2}\mu^2 + \frac{x\mu}{\sigma^2} - log(\sqrt{2\pi}\sigma)} \\ &= e^{-\frac{1}{2\sigma^2}x^2 + \frac{x\mu}{\sigma^2} - (\frac{1}{2\sigma^2}\mu^2 + log(\sqrt{2\pi}\sigma))} \end{aligned}$$

Where,

$\eta_1 = -\frac{1}{2\sigma^2}$ and $T_1(x) = x^2$

$\eta_2 = -\frac{\mu}{\sigma^2}$ and $T_2(x) = x$

$B(\theta) = \frac{1}{2\sigma^2}\mu^2 + log(\sqrt{2\pi}\sigma)$

$h(x) = 1$

In the case above, $\theta = (\mu, \sigma^2)$. If σ^2 is known, $\theta = \mu$. In this case, we can rewrite the normal pdf as follows.

$$\begin{aligned} P_\theta(X) &= e^{-\frac{1}{2\sigma^2}x^2 - \frac{1}{2\sigma^2}\mu^2 + \frac{x\mu}{\sigma^2} - log(\sqrt{2\pi}\sigma)} \\ &= e^{\frac{x\mu}{\sigma^2} - \frac{1}{2\sigma^2}\mu^2} e^{-\frac{1}{2\sigma^2}x^2 - log(\sqrt{2\pi}\sigma)} \end{aligned}$$

Where,

$\eta_1 = -\frac{\mu}{\sigma^2}$ and $T_1(x) = x$

$B(\theta) = \frac{1}{2\sigma^2}\mu^2$

$$\begin{aligned} h(x) &= e^{-\frac{1}{2\sigma^2}x^2 - log(\sqrt{2\pi}\sigma)} \\ &= \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\frac{x^2}{\sigma^2}} \end{aligned}$$

Thus, we can see that $h(x)$ is a normal pdf $\sim N(0, \sigma^2)$.

Bernoulli

Another example, x is descrete. For example, Bernoulli:

$$\begin{aligned} &= p^x(1-p)^{1-x} \\ &= e^{log(p^x(1-p)^{1-x})} \\ &= e^{xlog(p) + (1-x)log(1-p)} \\ &= e^{xlog(p) - xlog(1-p) + log(1-p)} \\ &= e^{xlog(\frac{p}{1-p}) + log(1-p)} \end{aligned}$$

Where,

$\eta_1 = log(\frac{p}{1-p})$ and $T_1(x) = x$

$B(\theta) = log(\frac{1}{1-p})$

$h(x) = 1$

2.6 Canonical exponential family

Canonical exponential family:

$$f_\theta(x) = e^{\frac{x\theta - b(\theta)}{\phi} + c(x,\phi)}$$

where, $b(.)$ and $c(.,.)$ are known.

Normal distribution

Again, use the normal pdf:

$$\begin{aligned}
P_\theta(X) &= \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\frac{(x-\mu)^2}{\sigma^2}} \\
&= e^{-\frac{1}{2}\frac{(x-\mu)^2}{\sigma^2}} e^{log(\frac{1}{\sigma\sqrt{2\pi}})} \\
&= e^{-\frac{1}{2}\frac{(x-\mu)^2}{\sigma^2} - log(\sigma\sqrt{2\pi})} \\
&= e^{-\frac{1}{2\sigma^2}x^2 - \frac{1}{2\sigma^2}\mu^2 + \frac{x\mu}{\sigma^2} - log(\sqrt{2\pi}\sigma)} \\
&= e^{\frac{x\mu}{\sigma^2} - \frac{\mu^2}{2\sigma^2} + (-\frac{1}{2\sigma^2}x^2 - log(\sqrt{2\pi}\sigma))} \\
&= e^{\frac{x\mu - \frac{1}{2}\mu^2}{\sigma^2} + (-\frac{1}{2\sigma^2}x^2 - log(\sqrt{2\pi}\sigma))}
\end{aligned}$$

Where (we assume σ^2 is known.),

$\theta = \mu$

$\phi = \sigma^2$

$b(\theta) = \frac{1}{2}\theta^2$

$$\begin{aligned}
c(x,\phi) &= -\frac{1}{2\sigma^2}x^2 - log(\sqrt{2\pi}\sigma) \\
&= -\frac{1}{2\sigma^2}x^2 - \frac{1}{2}log(2\pi\sigma^2) \\
&= -\frac{1}{2}(\frac{x^2}{\sigma^2} + log(2\pi\sigma^2)) \\
&= -\frac{1}{2}(\frac{x^2}{\phi} + log(2\pi\phi))
\end{aligned}$$

2.7 Canonical exponential family - Expected value and variance

First derivative

Canonical exponential family:

$$f_\theta(x) = e^{\frac{x\theta - b(\theta)}{\phi} + c(x,\phi)}$$

log likelihood (only one observation)

$$log f_\theta(x)$$

$$\begin{aligned} E[\frac{\partial(log f_\theta(X))}{\partial\theta}] &= E[\frac{\frac{\partial f_\theta(X)}{\partial\theta}}{f_\theta(X)}] \\ &= \int \frac{\frac{\partial f_\theta(X)}{\partial\theta}}{f_\theta(X)} f_\theta(X)dx \\ &= \int \frac{\partial f_\theta(X)}{\partial\theta} dx \\ &= \frac{\partial}{\partial\theta}\int f_\theta(X)dx \\ &= \frac{\partial 1}{\partial\theta} \\ &= 0 \end{aligned}$$

Second derivative

Second derivative

$$
\begin{aligned}
E[\frac{\partial^2(log f_\theta(X))}{\partial\theta^2}] &= E[\frac{\partial}{\partial\theta}(\frac{\frac{\partial f_\theta(X)}{\partial\theta}}{f_\theta(X)})] \\
&= E[\frac{\frac{\partial^2 f_\theta(X)}{\partial\theta^2} f_\theta(X) - (\frac{\partial f_\theta(X)}{\partial\theta})^2}{f_\theta^2(X)}] \\
&= \int \frac{\frac{\partial^2 f_\theta(X)}{\partial\theta^2} f_\theta(X) - (\frac{\partial f_\theta(X)}{\partial\theta})^2}{f_\theta(X)} dx \\
&= \int (\frac{\partial^2 f_\theta(X)}{\partial\theta^2} - \frac{(\frac{\partial f_\theta(X)}{\partial\theta})^2}{f_\theta(X)}) dx \\
&= \int \frac{\partial^2 f_\theta(X)}{\partial\theta^2} dx - \int \frac{(\frac{\partial f_\theta(X)}{\partial\theta})^2}{f_\theta(X)} dx \\
&= \frac{\partial^2}{\partial\theta^2} \int f_\theta(X) dx - \int \frac{(\frac{\partial f_\theta(X)}{\partial\theta})^2}{f_\theta(X)} dx \\
&= 0 - \int \frac{(\frac{\partial f_\theta(X)}{\partial\theta})^2}{f_\theta(X)} dx \\
&= 0 - \int \frac{(\frac{\partial f_\theta(X)}{\partial\theta})^2}{(f_\theta(X))^2} f_\theta(x) dx \\
&= -E[(\frac{\frac{\partial f_\theta(X)}{\partial\theta}}{f_\theta(X)})^2] \\
&= -E[(\frac{\partial(log f_\theta(X))}{\partial\theta})^2]
\end{aligned}
$$

Based on the first derivative, we can get:

$$log(f_\theta(X)) = \frac{X\theta - b(\theta)}{\phi} + c(X, \phi)$$

$$E[\frac{\partial(log(f_\theta(X)))}{\partial\theta}] = E[\frac{X - b^{'}(\theta)}{\phi}] = \frac{E(X) - b^{'}(\theta)}{\phi} = 0$$

Thus, we can get,

$$E(X) = b^{'}(\theta)$$

For second derivative, from the calculation above, we know that,

$$\begin{aligned} E[\frac{\partial^2(logf_\theta(X))}{\partial\theta^2}] &= -E[(\frac{\partial(logf_\theta(X))}{\partial\theta})^2] \\ &= -E[(\frac{X-b^{'}(\theta)}{\phi})^2] \\ &= -E[(\frac{X-E(X)}{\phi})^2] \\ &= -\frac{Var(X)}{\phi^2} \end{aligned}$$

At the same time,

$$\begin{aligned} E[\frac{\partial^2(logf_\theta(X))}{\partial\theta^2}] &= E[\frac{\partial(\frac{X-b^{'}(\theta)}{\phi})}{\partial\theta}] \\ &= E[-\frac{b^{''}(\theta)}{\phi}] \\ &= -\frac{b^{''}(\theta)}{\phi} \end{aligned}$$

Thus,

$$Var(X) = b^{''}(\theta)\phi$$

2.8 Expected value and variance - Possion Example

Example of possion distribution

$$P(\lambda) = \frac{\lambda^k e^{-\lambda}}{k!}$$

If we put k as y, and λ as μ, we can get:

$$P(\mu) = \frac{\mu^y e^{-\mu}}{y!}$$

Compare to,

$$f_\theta(y) = e^{\frac{y\theta-b(\theta)}{\phi}+c(y,\phi)}$$

We can write it as the exponential format:

$$\begin{aligned} P(\mu) &= \frac{\mu^y e^{-\mu}}{y!} \\ &= e^{log(\mu^y)+log(e^{-\mu})-log(y!)} \\ &= e^{ylog(\mu)-\mu-log(y!)} \end{aligned}$$

We thus know that $\theta = log(\mu)$. We can contintue to write the equation above as follows.

$$= e^{y\theta-e^{\theta}-log(y!)}$$

Thus, we can get:

$$E(X) = \frac{\partial(e^{\theta})}{\partial\theta} = e^{\theta} = \mu$$

$$Var(X) = \frac{\partial^{''}(e^{\theta})}{\partial\theta^2}\phi = \frac{\partial^{''}(e^{\theta})}{\partial\theta^2} = \mu$$

2.9 Canonical link

A link functin can link $X^T\beta$ to the mean μ.

That is,

$$g(\mu) = X^T\beta \rightarrow \mu = g^{-1}(X^T\beta)$$

We know that

$$\mu = b^{'}(\theta)$$

Thus,

$$b^{'}(\theta) = g^{-1}(X^T\beta)$$

Thus,

$$g = b^{'-1}(\theta)$$

2.10 Canonical link - Bernoulli

PMF of Bernoulli:

$$
\begin{aligned}
&= p^y(1-p)^{1-y} \\
&= e^{log(p^y(1-p)^{1-y})} \\
&= e^{ylog(p)+(1-y)log(1-p)} \\
&= e^{ylog(p)-ylog(1-p)+log(1-p)} \\
&= e^{ylog(\frac{p}{1-p})+log(1-p)}
\end{aligned}
$$

Copared to the following:

$$f_\theta(y) = e^{\frac{y\theta - b(\theta)}{\phi} + c(y,\phi)}$$

We need to change the format of Bernoulli:

$$\theta = log\frac{p}{1-p}$$

Thus,

$$e^\theta = \frac{p}{1-p} \rightarrow p = \frac{e^\theta}{1+e^\theta}$$

After that, we can contintue the Bernoulli:

$$
\begin{aligned}
&= e^{y\theta+log(1-\frac{e^\theta}{1+e^\theta})} \\
&= e^{y\theta+log(\frac{1}{1+e^\theta})} \\
&= e^{y\theta-log(1+e^\theta)}
\end{aligned}
$$

Where,

$$b(\theta) = log(1+e^\theta)$$

We can then try to calculate the derivative:

$$b^{'}(\theta) = \frac{\partial(log(1+e^\theta))}{\partial\theta} = \frac{e^\theta}{1+e^\theta}$$

We know that

$$b^{'}(\theta) = \mu$$

Thus, we can get

$$\mu = \frac{e^\theta}{1+e^\theta}$$

We can then calculate the inverse function:

$$\theta = log(\frac{\mu}{1-\mu})$$

Thus,

$$g(\mu) = log(\frac{\mu}{1-\mu}) = X^T\beta$$

2.11 NR - Bernoulli

We know that the PMF for Bernoulli:

$$\begin{aligned} &= p^y(1-p)^{1-y} \\ &= e^{y\theta - log(1+e^\theta)} \\ &= e^{yx^T\beta - log(1+e^{x^T\beta})} \end{aligned}$$

Thus,

$$\ell(\beta|Y,X) = \sum_{i=1}^{n}(Y_iX_i^T\beta - log(1+e^{X_i^T\beta}))$$

Thus, teh gradient is:

$$\nabla_\ell(\beta) = \sum_{i=1}^{n}(Y_iX_i - \frac{e^{X_i^T\beta}}{1+e^{X_i^T\beta}})$$

The Hessian is:

$$H_\ell(\beta) = -\sum_{i=1}^{n}\frac{e^{X_i^T\beta}}{(1+e^{X_i^T\beta})^2}X_iX_i^T$$

Thus,

$$\beta^{k+1} = \beta^k - (H_\ell(\beta^k))^{-1}\nabla_\ell(\beta^k)$$

2.12 Iteratively Re-weighted Least Squares

Chapter 3

Linear Mixed Models

3.1 LMM

The following is a shortened version of Jonathan Rosenblatt's LMM tutorial. http://www.john-ros.com/Rcourse/lme.html.

In addition, another reference is from Douglas Bates's R package document. https://cran.r-project.org/web/packages/lme4/vignettes/lmer.pdf?fbclid=IwAR1nmmRP9A0BrhKdgBibNjM5acR_spTpXV8QlQGdmTWyQz3ZtV3LYn6kCbQ

Assume that y is a function of x and u, where x is the fixed effect and u is the random effect. Thus, we can get,

$$y|x, u = x'\beta + z'u + \epsilon$$

For random effect, one example can be that you want to test the treatment effect, and sample 8 observations from 4 groups. You measure before and after the treatment. In this case, x represents the treatment effect, whereas z represents the group effect (i.e., random effect). Note that, in this case, it reminds the paired t-test. Remember in SPSS, why do we do paired t-test? Typically, it is the case when we measure a subject (or, participant) twice. In this case, we can consider each participant as an unit of random effect (rather than as group in the last example.)

3.2 Calculate mean

The following code generates 4 numbers ($N(0, 10)$) for 4 groups. Then, replicate it within each group.That is, in the end, there are 8 observations.

Note that, in the following code, there are no "independent variables". Both the linear model and mixed model are actually just trying to calculate the mean. Note that lmer(y~1+1|groups) and lmer(y~1|groups) will generate the same results.

```
set.seed(123)
n.groups <- 4 # number of groups
n.repeats <- 2 # samples per group
#Generating index for observations belong to the same group
groups <- as.factor(rep(1:n.groups, each=n.repeats))
n <- length(groups)
#Generating 4 random numbers, assuming normal distribution
z0 <- rnorm(n.groups, 0, 10)
z <- z0[as.numeric(groups)] # generate and inspect random group effects
z
```

```
## [1] -5.6047565 -5.6047565 -2.3017749 -2.3017749 15.5870831 15.5870831  0.7050839
## [8]  0.7050839
```

```
epsilon <- rnorm(n,0,1) # generate measurement error
beta0 <- 2 # this is the actual parameter of interest! The global mean.
y <- beta0 + z + epsilon # sample from an LMM

# fit a linear model assuming independence
# i.e., assume that there is no "group things".
lm.5 <- lm(y~1)

# fit a mixed-model that deals with the group dependence
#install.packages("lme4")
library(lme4)
lme.5.a <- lmer(y~1+1|groups)
lme.5.b <- lmer(y~1|groups)
lm.5
```

```
##
## Call:
## lm(formula = y ~ 1)
##
## Coefficients:
## (Intercept)
##       4.283
```

```
lme.5.a
```

```
## Linear mixed model fit by REML ['lmerMod']
```

```
## Formula: y ~ 1 + 1 | groups
## REML criterion at convergence: 36.1666
## Random effects:
##  Groups   Name        Std.Dev.
##  groups   (Intercept) 8.8521
##  Residual             0.8873
## Number of obs: 8, groups:  groups, 4
## Fixed Effects:
## (Intercept)
##       4.283
```

```
lme.5.b
```

```
## Linear mixed model fit by REML ['lmerMod']
## Formula: y ~ 1 | groups
## REML criterion at convergence: 36.1666
## Random effects:
##  Groups   Name        Std.Dev.
##  groups   (Intercept) 8.8521
##  Residual             0.8873
## Number of obs: 8, groups:  groups, 4
## Fixed Effects:
## (Intercept)
##       4.283
```

3.3 Test the treatment effect

As we can see that, LLM and paired t-test generate the same t-value.

```
times<-rep(c(1,2),4) # first time and second time
times
```

```
## [1] 1 2 1 2 1 2 1 2
```

```
data_combined<-cbind(y,groups,times)
data_combined
```

```
##               y groups times
## [1,] -3.4754687      1     1
## [2,] -1.8896915      1     2
## [3,]  0.1591413      2     1
## [4,] -1.5668361      2     2
```

```
## [5,] 16.9002303      3     1
## [6,] 17.1414212      3     2
## [7,]  3.9291657      4     1
## [8,]  3.0648977      4     2
```

```
lme_diff_times<- lmer(y~times+(1|groups))


t_results<-t.test(y~times, paired=TRUE)

lme_diff_times
```

```
## Linear mixed model fit by REML ['lmerMod']
## Formula: y ~ times + (1 | groups)
## REML criterion at convergence: 35.0539
## Random effects:
##  Groups   Name        Std.Dev.
##  groups   (Intercept) 8.845
##  Residual             1.013
## Number of obs: 8, groups:  groups, 4
## Fixed Effects:
## (Intercept)        times
##      4.5691      -0.1908
```

```
print("The following results are from paired t-test")
```

```
## [1] "The following results are from paired t-test"
```

```
t_results$statistic
```

```
##         t
## 0.2664793
```

3.4 Another example

```
data(Dyestuff, package='lme4')
attach(Dyestuff)
```

```
## The following objects are masked from Dyestuff (pos = 6):
##
##     Batch, Yield
```

```
Dyestuff
```

```
##    Batch Yield
## 1      A  1545
## 2      A  1440
## 3      A  1440
## 4      A  1520
## 5      A  1580
## 6      B  1540
## 7      B  1555
## 8      B  1490
## 9      B  1560
## 10     B  1495
## 11     C  1595
## 12     C  1550
## 13     C  1605
## 14     C  1510
## 15     C  1560
## 16     D  1445
## 17     D  1440
## 18     D  1595
## 19     D  1465
## 20     D  1545
## 21     E  1595
## 22     E  1630
## 23     E  1515
## 24     E  1635
## 25     E  1625
## 26     F  1520
## 27     F  1455
## 28     F  1450
## 29     F  1480
## 30     F  1445
```

```
lme_batch<- lmer( Yield ~ 1 + (1|Batch)  , Dyestuff )
summary(lme_batch)
```

```
## Linear mixed model fit by REML ['lmerMod']
## Formula: Yield ~ 1 + (1 | Batch)
##    Data: Dyestuff
##
## REML criterion at convergence: 319.7
##
## Scaled residuals:
```

```
##     Min      1Q  Median      3Q     Max
## -1.4117 -0.7634  0.1418  0.7792  1.8296
##
## Random effects:
##  Groups   Name        Variance Std.Dev.
##  Batch    (Intercept) 1764     42.00
##  Residual             2451     49.51
## Number of obs: 30, groups:  Batch, 6
##
## Fixed effects:
##             Estimate Std. Error t value
## (Intercept)  1527.50      19.38    78.8
```

3.5 Full LMM model

In the following, I used the data from the package of lme4. For Days + (1 | Subject), it only has random intercept; in contrast, Days + (Days| Subject) has both random intercept and random slope for Days. Note that, random effects do not generate specific slopes for each level of Days, but rather just a variance of all the slopes.

Therefore, we can see that "Days + (Days| Subject)" and "Days + (1+Days| Subject)" generate the same results. For more discussion, you can refer to the following link: https://www.jaredknowles.com/journal/2013/11/25/getting-started-with-mixed-effect-models-in-r

```
data(sleepstudy, package='lme4')
attach(sleepstudy)
```

```
## The following objects are masked from sleepstudy (pos = 6):
##
##     Days, Reaction, Subject
```

```
fm1 <- lmer(Reaction ~ Days + (1 | Subject), sleepstudy)
summary(fm1)
```

```
## Linear mixed model fit by REML ['lmerMod']
## Formula: Reaction ~ Days + (1 | Subject)
##    Data: sleepstudy
##
## REML criterion at convergence: 1786.5
##
## Scaled residuals:
```

```
##     Min      1Q  Median      3Q     Max
## -3.2257 -0.5529  0.0109  0.5188  4.2506
##
## Random effects:
##  Groups   Name        Variance Std.Dev.
##  Subject  (Intercept) 1378.2   37.12
##  Residual              960.5   30.99
## Number of obs: 180, groups:  Subject, 18
##
## Fixed effects:
##             Estimate Std. Error t value
## (Intercept) 251.4051     9.7467   25.79
## Days         10.4673     0.8042   13.02
##
## Correlation of Fixed Effects:
##      (Intr)
## Days -0.371
```

```
fm2<-lmer ( Reaction ~ Days + ( Days| Subject ) , data= sleepstudy )
summary(fm2)
```

```
## Linear mixed model fit by REML ['lmerMod']
## Formula: Reaction ~ Days + (Days | Subject)
##    Data: sleepstudy
##
## REML criterion at convergence: 1743.6
##
## Scaled residuals:
##     Min      1Q  Median      3Q     Max
## -3.9536 -0.4634  0.0231  0.4633  5.1793
##
## Random effects:
##  Groups   Name        Variance Std.Dev. Corr
##  Subject  (Intercept) 611.90   24.737
##           Days         35.08    5.923   0.07
##  Residual             654.94   25.592
## Number of obs: 180, groups:  Subject, 18
##
## Fixed effects:
##             Estimate Std. Error t value
## (Intercept)  251.405      6.824  36.843
## Days          10.467      1.546   6.771
##
## Correlation of Fixed Effects:
##      (Intr)
```

```
## Days -0.138
```

```
fm3<-lmer ( Reaction ~ Days + (1+Days| Subject ) , data= sleepstudy )
summary(fm3)
```

```
## Linear mixed model fit by REML ['lmerMod']
## Formula: Reaction ~ Days + (1 + Days | Subject)
##    Data: sleepstudy
##
## REML criterion at convergence: 1743.6
##
## Scaled residuals:
##     Min      1Q  Median      3Q     Max
## -3.9536 -0.4634  0.0231  0.4633  5.1793
##
## Random effects:
##  Groups   Name        Variance Std.Dev. Corr
##  Subject  (Intercept) 611.90   24.737
##           Days         35.08    5.923   0.07
##  Residual             654.94   25.592
## Number of obs: 180, groups:  Subject, 18
##
## Fixed effects:
##             Estimate Std. Error t value
## (Intercept)  251.405      6.824  36.843
## Days          10.467      1.546   6.771
##
## Correlation of Fixed Effects:
##      (Intr)
## Days -0.138
```

3.6 Serial correlations in time and space

The hierarchical model of $y|x, u = x'\beta + z'u + \epsilon$ can work well for correlations within blocks, but not for correlations in time as the correlations decay in time. The following uses nlme package to calculate time serial data.

```
library(nlme)
head(nlme::Ovary,n=50)
```

```
## Grouped Data: follicles ~ Time | Mare
##    Mare        Time follicles
## 1     1 -0.13636360        20
```

```
## 2     1 -0.09090910        15
## 3     1 -0.04545455        19
## 4     1  0.00000000        16
## 5     1  0.04545455        13
## 6     1  0.09090910        10
## 7     1  0.13636360        12
## 8     1  0.18181820        14
## 9     1  0.22727270        13
## 10    1  0.27272730        20
## 11    1  0.31818180        22
## 12    1  0.36363640        15
## 13    1  0.40909090        18
## 14    1  0.45454550        17
## 15    1  0.50000000        14
## 16    1  0.54545450        18
## 17    1  0.59090910        14
## 18    1  0.63636360        16
## 19    1  0.68181820        17
## 20    1  0.72727270        18
## 21    1  0.77272730        18
## 22    1  0.81818180        17
## 23    1  0.86363640        14
## 24    1  0.90909090        12
## 25    1  0.95454550        12
## 26    1  1.00000000        14
## 27    1  1.04545500        10
## 28    1  1.09090900        11
## 29    1  1.13636400        16
## 30    2 -0.15000000         6
## 31    2 -0.10000000         6
## 32    2 -0.05000000         8
## 33    2  0.00000000         7
## 34    2  0.05000000        16
## 35    2  0.10000000        10
## 36    2  0.15000000        13
## 37    2  0.20000000         9
## 38    2  0.25000000         7
## 39    2  0.30000000         6
## 40    2  0.35000000         8
## 41    2  0.40000000         8
## 42    2  0.45000000         6
## 43    2  0.50000000         8
## 44    2  0.55000000         7
## 45    2  0.60000000         9
## 46    2  0.65000000         6
## 47    2  0.70000000         4
```

```
## 48    2  0.75000000         5
## 49    2  0.80000000         8
## 50    2  0.85000000        11
```

```
fm1Ovar.lme <- nlme::lme(fixed=follicles ~ sin(2*pi*Time) + cos(2*pi*Time),
                  data = Ovary,
                  random = pdDiag(~sin(2*pi*Time)),
                  correlation=corAR1() )
summary(fm1Ovar.lme)
```

```
## Linear mixed-effects model fit by REML
##  Data: Ovary
##        AIC     BIC   logLik
##   1563.448 1589.49 -774.724
##
## Random effects:
##  Formula: ~sin(2 * pi * Time) | Mare
##  Structure: Diagonal
##         (Intercept) sin(2 * pi * Time) Residual
## StdDev:    2.858385           1.257977 3.507053
##
## Correlation Structure: AR(1)
##  Formula: ~1 | Mare
##  Parameter estimate(s):
##       Phi
## 0.5721866
## Fixed effects: follicles ~ sin(2 * pi * Time) + cos(2 * pi * Time)
##                        Value Std.Error  DF   t-value p-value
## (Intercept)        12.188089 0.9436602 295 12.915760  0.0000
## sin(2 * pi * Time) -2.985297 0.6055968 295 -4.929513  0.0000
## cos(2 * pi * Time) -0.877762 0.4777821 295 -1.837159  0.0672
##  Correlation:
##                    (Intr) s(*p*T
## sin(2 * pi * Time)  0.000
## cos(2 * pi * Time) -0.123  0.000
##
## Standardized Within-Group Residuals:
##         Min          Q1         Med          Q3         Max
## -2.34910093 -0.58969626 -0.04577893  0.52931186  3.37167486
##
## Number of Observations: 308
## Number of Groups: 11
```

Chapter 4

Basic R

This section is about R coding.

4.1 apply, lapply, sapply

4.1.1 apply

```
m_trying <- matrix(C<-(1:10),nrow=2, ncol=5)
m_trying
```

```
##      [,1] [,2] [,3] [,4] [,5]
## [1,]    1    3    5    7    9
## [2,]    2    4    6    8   10
```

```
## Operating on the columns
apply(m_trying, 2, sum)
```

```
## [1]  3  7 11 15 19
```

```
## Operating on the rows
apply(m_trying, 1, sum)
```

```
## [1] 25 30
```

4.1.2 lapply

"lapply returns a list of the same length as X, each element of which is the result of applying FUN to the corresponding element of X."

lapply operates on lists. Thus, as we can see below, even if m_trying is not a list, each cell becomes a list.

```
results1<-lapply(m_trying,sum)
str(results1)
```

```
## List of 10
##  $ : int 1
##  $ : int 2
##  $ : int 3
##  $ : int 4
##  $ : int 5
##  $ : int 6
##  $ : int 7
##  $ : int 8
##  $ : int 9
##  $ : int 10
```

```
is.list(results1)
```

```
## [1] TRUE
```

4.1.3 sapply

"sapply() function takes list, vector or data frame as input and gives output in vector or matrix."

```
results2<-sapply(m_trying, sum)
str(results2)
```

```
##  int [1:10] 1 2 3 4 5 6 7 8 9 10
```

```
is.list(results2)
```

```
## [1] FALSE
```

```
is.matrix(results2)
```

```
## [1] FALSE
```

```
is.data.frame(results2)
```

```
## [1] FALSE
```

```
is.vector(results2)
```

```
## [1] TRUE
```

4.2 C

```
mydata1<-matrix(runif(4*2),4,2)
mydata1
```

```
##           [,1]      [,2]
## [1,] 0.6557058 0.2891597
## [2,] 0.7085305 0.1471136
## [3,] 0.5440660 0.9630242
## [4,] 0.5941420 0.9022990
```

```
str(mydata1)
```

```
##  num [1:4, 1:2] 0.656 0.709 0.544 0.594 0.289 ...
```

```
mydata2<-c(mydata1)
mydata2
```

```
## [1] 0.6557058 0.7085305 0.5440660 0.5941420 0.2891597 0.1471136 0.9630242
## [8] 0.9022990
```

```
str(mydata2)
```

```
##  num [1:8] 0.656 0.709 0.544 0.594 0.289 ...
```

Chapter 5

Computing Techniques

Since GLMM can use EM algorithm in its maximum likelihood calculation (see McCulloch, 1994), it is practically useful to rehearse EM and other computing techniques.

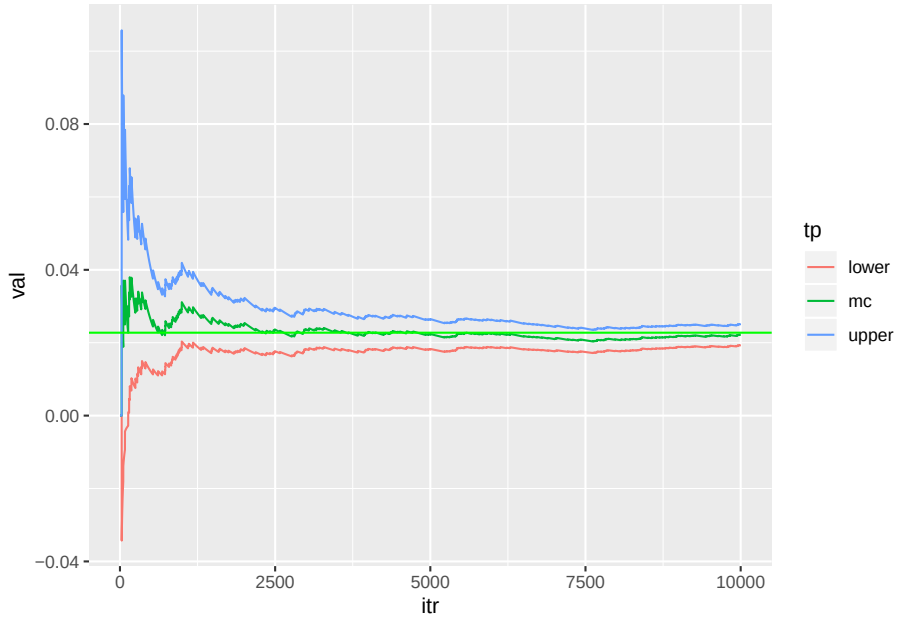
5.1 Monte carlo approximation

Example: calculate the integral of $p(z > 2)$ when $z \sim N(0, 1)$. To use Monte Carlo approximation, we can have an indicator function, which will determine whether the sample from $N(0, 1)$ will be included into the calculation of the integral.

```
Nsim=10^4

indicator=function(x){
y=ifelse((x>2),1,0)
return(y)}

newdata<-rnorm(Nsim, 0,1 )

mc=c(); v=c(); upper=c(); lower=c()

for (j in 1:Nsim)
{
mc[j]=mean(indicator(newdata[1:j]))
v[j]=(j^{-1})*var(indicator(newdata[1:j]))
upper[j]=mc[j]+1.96*sqrt(v[j])
lower[j]=mc[j]-1.96*sqrt(v[j])
}
```

```
library(ggplot2)
values=c(mc,upper,lower)
type=c(rep("mc",Nsim),rep("upper",Nsim),rep("lower",Nsim))
iter=rep(seq(1:Nsim),3)
data=data.frame(val=values, tp=type, itr=iter)
Rcode<-ggplot(data,aes(itr,val,col=tp))+geom_line(size=0.5)
Rcode+geom_hline(yintercept=1-pnorm(2),color="green",size=0.5)
```

```
## Warning: Removed 2 rows containing missing values (geom_path).
```



5.2 Importance sampling

Importance sampling has samples generated from a different distribution than the distribution of interest. Specifically, assume that we want to calculate the expected value of $h(x)$, and $x \sim f(x)$.

$$E(h(x)) = \int h(x)f(x)dx = \int h(x)\frac{f(x)}{g(x)}g(x)dx$$

We can sample x_i from $g(x)$ and then calculate the mean of $h(x_i)\frac{f(x_i)}{g(x_i)}$.

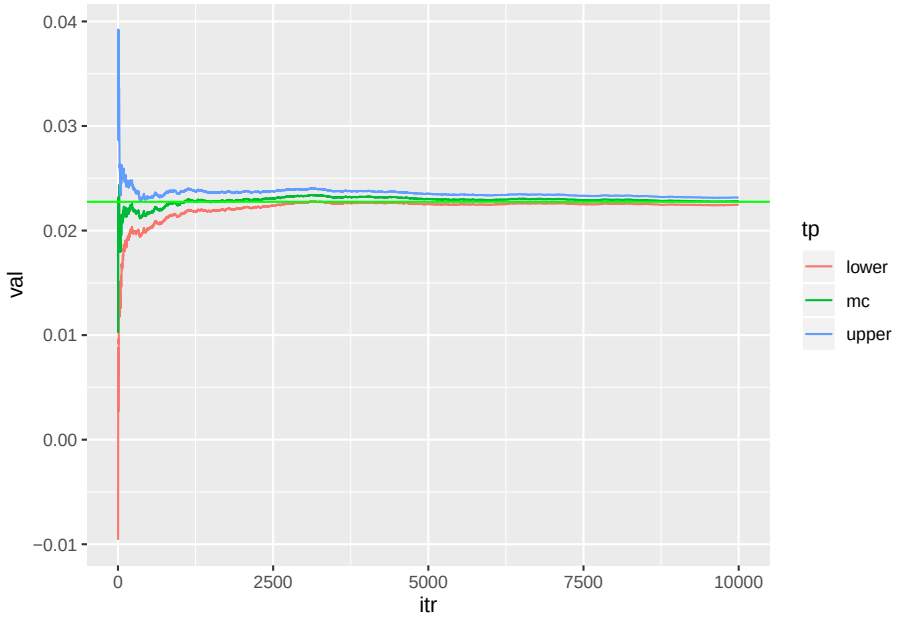
Using the same explane above, we can use a shifted exponential distribution to help calculate the intergral for normal distribution. Specifically,

$$\int_2^\infty \frac{1}{2\pi} e^{-\frac{1}{2}x^2} dx = \int_2^\infty \frac{\frac{1}{2\pi} e^{-\frac{1}{2}x^2}}{e^{-(x-2)}} e^{-(x-2)} dx$$

The idea is that, we can generate x_i from exponential distribution of $e^{-(x-2)}$, and then insert them into the targeted "expected (value) function" of $\frac{\frac{1}{2\pi} e^{-\frac{1}{2}x^2}}{e^{-(x-2)}}$. Thus, as you can see, importance sampling is based on the law of large numbers (i.e., If the same experiment or study is repeated independently a large number of times, the average of the results of the trials must be close to the expected value). We can use it to calculate integral based on link of the definition of expected value.

```
Nsim=10^4
normal_density=function(x)
{y=(1/sqrt(2*pi))*exp(-0.5*(x^2))
return(y)}
x=2-log(runif(Nsim))
ImpS=c(); v=c(); upper=c(); lower=c()
for (j in 1:Nsim)
{
ImpS[j]=mean(normal_density(x[1:j])/exp(-(x[1:j]-2)))
v[j]=(j^{-1})*var(normal_density(x[1:j])/exp(-(x[1:j]-2)))
upper[j]=ImpS[j]+1.96*sqrt(v[j])
lower[j]=ImpS[j]-1.96*sqrt(v[j])
}

library(ggplot2)
values=c(ImpS,upper,lower)
type=c(rep("mc",Nsim),rep("upper",Nsim),rep("lower",Nsim))
iter=rep(seq(1:Nsim),3)
data=data.frame(val=values, tp=type, itr=iter)
ggplot(data,aes(itr,val,col=tp))+geom_line(size=0.5)+
geom_hline(yintercept=1-pnorm(2),color="green",size=0.5)
```

```
## Warning: Removed 2 rows containing missing values (geom_path).
```



5.3 Newton Raphson algorithm

The main purpose of Newton Raphson algorithm is to calculate the root of a function (e.g., $x^2 - 3 = 0$). We know that in order to maximize the MLE, we need to calculate the first derivatice of the function and then set it to zero $\ell^{'}(x) = 0$. Thus, we can use the same Newton Raphson method to help calculate the MLE maximization as well.

There are different ways to understand Newton Raphson method, but I found the method fo geometric the most easy way to explain.

Specifically, suppose that you want to calculate the root of a function $f(x) = 0$. We assume the root is r. However, we do not that, and we randomly guess a point of a. Thus, we can get a tangent line with slope of $f^{'}(a)$ and a point of $(a, f(a))$. Since we know the slope and one of its points, we can write the function for this tangent line.

$$y - f(a) = f^{'}(a)(x - a)$$

To calculate the $x - intercept$, namely b in the figure, we can set $y = 0$, and get the following:

$$-f(a) = f^{'}(a)(x - a) \Rightarrow x(or, b) = a - \frac{f(a)}{f^{'}(a)}$$

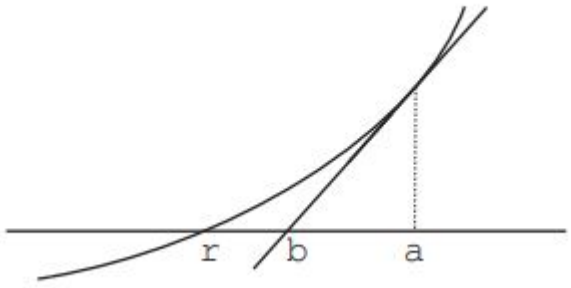


Figure 5.1: Credit of this figure: https://www.math.ubc.ca/~anstee/math104/newtonmethod.pdf

If there is significant difference of $|a - b|$, we know that our orginal guess of a is not good. We better use b as the next guess, and calculate its tangent line again. To generalize, we can write it as follows.

$$x_{t+1} = x_t - \frac{f(x_t)}{f'(x_t)}$$

Okay, this method above is to calculate the root. For MLE, we can also use this method to calculate the root for the $\ell^{'} = 0$. We can write it as follows.

$$x_{t+1} = x_t - \frac{\ell^{'}(x_t)}{\ell^{''}(x_t)}$$

Often, x is not just a single unknow parameter, but a vector. For this case, we can write it as follows.

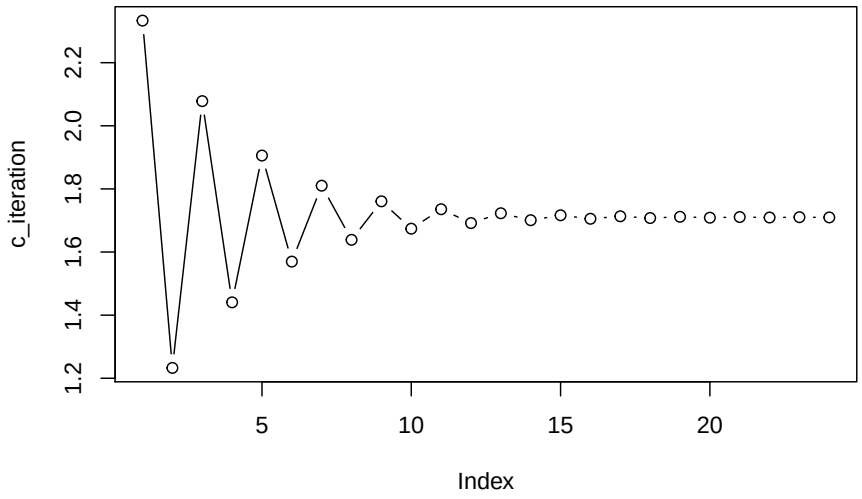
$$\beta_{t+1} = \beta_t - \frac{\ell^{'}(\beta_t)}{\ell^{''}(\beta_t)}$$

5.3.1 Calculate the root

$x^3 - 5 = 0$

Note that, this is obviously not a maximization problem. In contrast, it involves a function with zero. As we can see, we can think it as the first order of Taylor approximation. That is, $f^{'}(x) = x^3 - 5 = 0$. As we can see the following plot, it converts very quickly.

```r
f_firstorder=function(x){x^3-5}
f_secondorder=function(x){3*x}
x_old=1;tolerance=1e-3
max_its=2000;iteration=1;difference=2
c_iteration<-c() ## to collect numbers generated in the iteration process
while(difference>tolerance & iteration<max_its){
  x_updated=x_old-(f_firstorder(x_old)/f_secondorder(x_old))
  difference=abs(x_updated-x_old);
  iteration=iteration+1;
  x_old=x_updated
  c_iteration<-c(c_iteration,x_updated)}

plot(c_iteration,type="b")
```



5.3.2 Logistic regression

Suppose we have n observation, and m variables.

$$\begin{bmatrix} x_{11} & x_{12} & x_{13} & \dots & x_{1m} \\ x_{21} & x_{22} & x_{23} & \dots & x_{2m} \\ \dots & & & & \\ x_{n1} & x_{n2} & x_{n3} & \dots & x_{nm} \end{bmatrix}$$

Typically, we add a vector of 1 being used to estimate the constant.

$$\begin{bmatrix} 1 & x_{11} & x_{12} & x_{13} & ... & x_{1m} \\ 1 & x_{21} & x_{22} & x_{23} & ... & x_{2m} \\ ... & & & & & \\ 1 & x_{n1} & x_{n2} & x_{n3} & ... & x_{nm} \end{bmatrix}$$

And, we have observe a vector of n y_i as well, which is a binary variable:

$$Y = \begin{bmatrix} 1 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ ... \\ 1 \end{bmatrix}$$

Using the content from the MLE chapter, we can get:

$$\mathbf{L} = \prod_{i=1}^{n} p_i^{y_i}(1-p_i)^{(1-y_i)}$$

Further, we can get a log-transformed format.

$$log(\mathbf{L}) = \sum_{i=1}^{n}[y_i log(p_i) + (1-y_i)log(1-p_i)]$$

Given that $p_i = \frac{e^{\beta_0+\beta_1 x_1+...+\beta_n x_n}}{1+e^{\beta_0+\beta_1 x_1+...+\beta_n x_n}} = \frac{e^{\beta^T x}}{1+e^{\beta^T x}}$, we can rewrite it as follows:

$$log(\mathbf{L}) = \ell = \sum_{i=1}^{n}[y_i log(\frac{e^{\beta^T x_i}}{1+e^{\beta^T x_i}}) + (1-y_i)log(1-\frac{e^{\beta^T x_i}}{1+e^{\beta^T x_i}})]$$

Before doing the derivative, we set.

$$\frac{e^{\beta^T x_i}}{1+e^{\beta^T x_i}} = p(\beta^T x_i)$$

$$log(\mathbf{L}) = \ell = \sum_{i=1}^{n}[y_i log(p(\beta^T x_i)) + (1-y_i)log(1-p(\beta^T x_i))]$$

Note that, $\frac{\partial p(\beta^T x_i)}{\partial(\beta^T x_i)} = p(\beta^T x_i)(1-p(\beta^T x_i))$. We will use it later.

$$
\begin{aligned}
\nabla \ell &= \sum_{i=1}^{n} [y_i \frac{1}{p(\beta^T x_i)} \frac{\partial p(\beta^T x_i)}{\partial (\beta^T x_i)} \frac{\partial (\beta^T x_i)}{\partial \beta} + (1-y_i) \frac{1}{1-p(\beta^T x_i)} (-1) \frac{\partial p(\beta^T x_i)}{\partial (\beta^T x_i)} \frac{\partial (\beta^T x_i)}{\partial \beta}] \\
&= \sum_{i=1}^{n} x_i^T [y_i \frac{1}{p(\beta^T x_i)} p(\beta^T x_i)(1-p(\beta^T x_i)) + (1-y_i) \frac{1}{1-p(\beta^T x_i)} (-1) p(\beta^T x_i)(1-p(\beta^T x_i))] \\
&= \sum_{i=1}^{n} x_i^T [y_i \frac{1}{p(\beta^T x_i)} p(\beta^T x_i)(1-p(\beta^T x_i)) - (1-y_i) \frac{1}{1-p(\beta^T x_i)} p(\beta^T x_i)(1-p(\beta^T x_i))] \\
&= \sum_{i=1}^{n} x_i^T [y_i (1-p(\beta^T x_i)) - (1-y_i) p(\beta^T x_i)] \\
&= \sum_{i=1}^{n} x_i^T [y_i - y_i p(\beta^T x_i) - p(\beta^T x_i) + y_i p(\beta^T x_i)] \\
&= \sum_{i=1}^{n} x_i^T [y_i - p(\beta^T x_i)] \\
&= \sum_{i=1}^{n} x_i^T [y_i - \frac{e^{\beta^T x_i}}{1+e^{\beta^T x_i}}]
\end{aligned}
$$

As noted, the Newton Raphson algorithm needs the second order.

$$
\begin{aligned}
\nabla^2 \ell &= \frac{\partial \sum_{i=1}^{n} x_i^T [y_i - p(\beta^T x_i)]}{\partial \beta} \\
&= -\sum_{i=1}^{n} x_i^T \frac{\partial p(\beta^T x_i)}{\partial \beta} \\
&= -\sum_{i=1}^{n} x_i^T \frac{\partial p(\beta^T x_i)}{\partial (\beta^T x_i)} \frac{\partial (\beta^T x_i)}{\partial \beta} \\
&= -\sum_{i=1}^{n} x_i^T p(\beta^T x_i)(1-p(\beta^T x_i)) x_i
\end{aligned}
$$

The following are the data simulation (3 IVs and 1 DV) and Newton Raphson analysis.

```
# Data generation
set.seed(123)
n=500
x1_norm<-rnorm(n)
x2_norm<-rnorm(n,3,4)
x3_norm<-rnorm(n,4,6)
x_combined<-cbind(1,x1_norm,x2_norm,x3_norm) # dimension: n*4
```

```
coefficients_new<-c(1,2,3,4)  #true regression coefficient
inv_logit<-function(x,b){exp(x%*%b)/(1+exp(x%*%b))}
prob_generated<-inv_logit(x_combined,coefficients_new)
y<-c()
for (i in 1:n) {y[i]<-rbinom(1,1,prob_generated[i])}

# Newton Raphson

#We need to set random starting values.
beta_old<-c(1,1,1,1)
tolerance=1e-3
max_its=2000;iteration=1;difference=2
W<-matrix(0,n,n)

while(difference>tolerance & iteration<max_its)
  {
  # The first order
  f_firstorder<-t(x_combined)%*%(y-inv_logit(x_combined,beta_old))
  # The second order
  diag(W) = inv_logit(x_combined,beta_old)*(1-inv_logit(x_combined,beta_old))
  f_secondorder<--t(x_combined)%*%W%*%x_combined
  # Calculate the beta_updated
  beta_updated=beta_old-(solve(f_secondorder)%*%f_firstorder)
  difference=max(abs(beta_updated-beta_old));
  iteration=iteration+1;
  beta_old=beta_updated}

beta_old
```

```
##                [,1]
##          0.9590207
## x1_norm 1.7974165
## x2_norm 3.0072303
## x3_norm 3.9578107
```

$$\frac{\partial \ell}{\partial \beta} = \sum_{i=1}^{n} [y_i \frac{1}{p(\beta^T x_i)} \frac{\partial p(\beta^T x_i)}{\partial (\beta^T x_i)} \frac{\partial (\beta^T x_i)}{\partial \beta} + (1-y_i) \frac{1}{1-p(\beta^T x_i)} (-1) \frac{\partial p(\beta^T x_i)}{\partial (\beta^T x_i)} \frac{\partial (\beta^T x_i)}{\partial \beta}]$$

$$= \sum_{i=1}^{n} [y_i \frac{1}{p(\beta^T x_i)} \phi(\beta^T x_i) - (1-y_i) \frac{1}{1-p(\beta^T x_i)} \phi(\beta^T x_i)] x_i$$

$$\Phi(\beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_3) = p(y=1)$$

```
# Data generation
n=500
x1_norm<-rnorm(n)
x2_norm<-rnorm(n)
x3_norm<-rnorm(n)
x_combined<-cbind(1,x1_norm,x2_norm,x3_norm)
coefficients_new<-c(2,2,3,3)  #true regression coefficient
inv_norm<-function(x,b){pnorm(x%*%b)}
prob_generated<-inv_norm(x_combined,coefficients_new)
y<-c()
for (i in 1:n) {y[i]<-rbinom(1,1,prob_generated[i])}

# Newton Raphson

#We need to set random starting values.
x_old<-c(1,1,1,1)
tolerance=1e-3
max_its=2000;iteration=1;difference=2

while(difference>tolerance & iteration<max_its){
  x_updated=x_old-(f_firstorder(x_old)/f_secondorder(x_old))
  difference=abs(x_updated-x_old);
  iteration=iteration+1;
  x_old=x_updated
  c_iteration<-c(c_iteration,x_updated)}

plot(c_iteration,type="b")
```

5.4 Metropolis Hastings

Metropolis–Hastings is a MCMC method for obtaining a sequence of random samples from a probability distribution from which direct sampling is difficult. By using the samples, we can plot the distribution (through histgram), or we can calculate the integral (e.g., you need to calculate the expected value).

(Side note: does this remind you the importance sampling? Very similiar!)

Basic logic (my own summary):

(1) Set up a random starting value of x_0.

(2) Sample a y_0 from the instrumental function of $q(x)$.

(3) Calculate the following:

$p = \frac{f(y_0)}{f(x_0)} \frac{q(x_0)}{q(y_0)}$

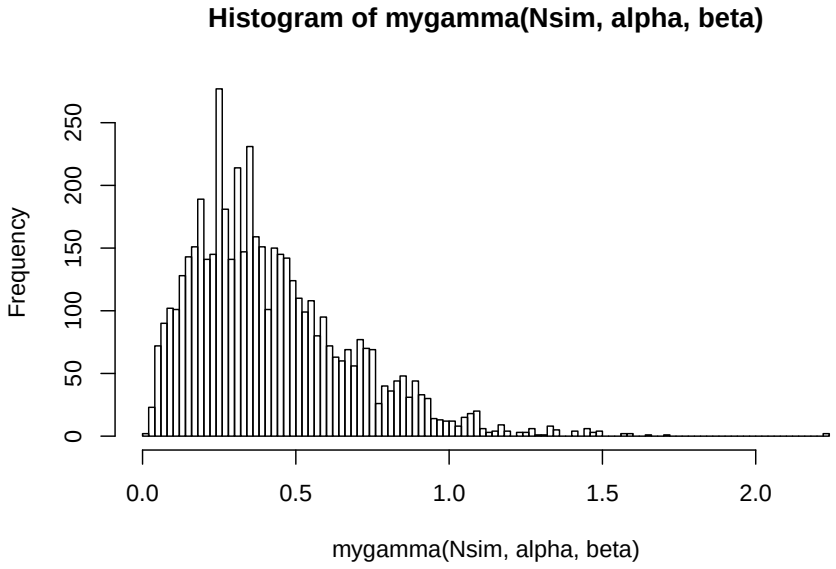
(4) $\rho = min(p, 1)$

(5) $x_1 = \begin{cases} y_0 & p \\ x_0 & 1-p \end{cases}$

(6) Repeat n times (n is set subjectively.)

Use normal pdf to sample gamma distribution

```
alpha=2.7; beta=6.3 # I randomly chose alpha and beta values for the target gamma function

Nsim=5000  ## define the number of iteration

X=c(rgamma(1,1)) # initialize the chain from random starting numbers
mygamma<-function(Nsim,alpha,beta){
for (i in 2:Nsim){
  Y=rnorm(1)
  rho=dgamma(Y,alpha,beta)*dnorm(X[i-1])/(dgamma(X[i-1],alpha,beta)*dnorm(Y))
  X[i]=X[i-1] + (Y-X[i-1])*(runif(1)<rho)
}
X
}

hist(mygamma(Nsim,alpha,beta), breaks = 100)
```



5.5 EM

EM algorithm is an iterative method to find ML or maximum a posteriori (MAP) estimates of parameters.

Direct Ref: http://www.di.fc.ul.pt/~jpn/r/EM/EM.html

Suppose that we only observe X, and do not know Z. We thus need to construct the complete data of (X, Z). Given $p(Z|X, \theta)$, we can compute the likelihood of the complete dataset:

$$p(X, Z|\theta) = p(Z|X, \theta)p(X|\theta)$$

The EM algorithm:

(0) We got X and $p(Z|X, \theta)$

(1) Random assign a θ_0, since we do not know any of them.

(2) E-step: $Q_{\theta_i} = E_{Z|X,\theta_i}[logp(X, Z|\theta)]$

(3) M-step: compute $\theta_{i+1} \leftarrow argmaxQ_{\theta_i}$

(4) If θ_i and θ_{i+1} are not close enough, $\theta_i \leftarrow \theta_{i+1}$. Goto step 2.

For examples, you can refer to the following link: http://www.di.fc.ul.pt/~jpn/r/EM/EM.html

(It is em_R.r in R_codes folder. Personally, I can also refer to Quiz 2 in 536.)

5.6 References

1. The UBC PDF about Newton

https://www.math.ubc.ca/~anstee/math104/newtonmethod.pdf

2. Some other pages about Newton and logistic regression

http://www.win-vector.com/blog/2011/09/the-simpler-derivation-of-logistic-regression/

https://stats.stackexchange.com/questions/344309/why-using-newtons-method-for-logistic-regression-optimization-is-called-iterati

https://tomroth.com.au/logistic/

https://www.stat.cmu.edu/~cshalizi/350/lectures/26/lecture-26.pdf

https://www.stat.cmu.edu/~cshalizi/402/lectures/14-logistic-regression/lecture-14.pdf

http://hua-zhou.github.io/teaching/biostatm280-2017spring/slides/18-newton/newton.html

3. MH

https://www.youtube.com/watch?v=VGRVRjr0vyw

Chapter 6

Generalized Linear Mixed Models

6.1 Basics of GLMM

Recall the formula in the probit model:

$$Y^* = X\beta + \epsilon, \epsilon \sim N(0, \sigma^2) = N(0, I)$$

Similar to LMM, binary model with random effect can be written as follows.

$$Y^* = X\beta + Zu + \epsilon$$

where,

$$\epsilon \sim N(0, I)$$

$$u \sim N(0, D)$$

We also assume ϵ and u are independent.Thus, we know that D represents the virances of the random effects. If we make $u = 1$, the model becomes the usual probit model. McCulloch (1994) states that there are a few advantages to use probit, rather than logit models. (Note that, however, probit is not canonical link function, but logit is!)

The following is the note from Charle E. McCulloch's "Maximum likelihood algorithems for Generalized Linear Mixed Models"

6.2 Some References

http://www.biostat.umn.edu/~baolin/teaching/linmods/glmm.html

http://www.biostat.umn.edu/~baolin/teaching/probmods/GLMM_mcmc.html

https://bbolker.github.io/mixedmodels-misc/glmmFAQ.html

Chapter 7

Twitter Example

The following is part of my course project for Stat 536. It aims to replicate part of the findings from Barbera (2015) Birds of the Same Feather Tweet Together: Bayesian Ideal Point Estimation Using Twitter Data. Political Analysis 23 (1). Note that, the following model is much simpler than that in the original paper.

7.1 Model

Suppose that a Twitter user is presented with a choice between following or not following another target $j \in \{1, ..., m\}$. Let $y_j = 1$ if the user decides to follow j, and $y_j = 0$ otherwise.

$$y_j = \begin{cases} 1 & Following \\ 0 & NotFollowing \end{cases}$$

$$p(y_j = 1|\theta) = \frac{exp(-\theta_0|\theta_1 - x_j|^2)}{1 + exp(-\theta_0|\theta_1 - x_j|^2)}$$

We additionally know the priors of θ.

$$\theta_i \sim N(0, 10^2)(i = 0, 1)$$

The likelihood function is as follows.

$$L(Y|\theta) = \prod_{j=1}^{m} (\frac{exp(-\theta_0|\theta_1 - x_j|^2)}{1 + exp(-\theta_0|\theta_1 - x_j|^2)})^{y_j} (1 - \frac{exp(-\theta_0|\theta_1 - x_j|^2)}{1 + exp(-\theta_0|\theta_1 - x_j|^2)})^{(1-y_j)}$$

Thus, the posterior is as follows.

$$L(Y|\theta) \cdot N(\theta_0|0,10) \cdot N(\theta_1|0,10)$$

$$\propto \prod_{j=1}^{m} \left(\frac{exp(-\theta_0|\theta_1 - x_j|^2)}{1+exp(-\theta_0|\theta_1 - x_j|^2)}\right)^{y_j} \left(1-\frac{exp(-\theta_0|\theta_1 - x_j|^2)}{1+exp(-\theta_0|\theta_1 - x_j|^2)}\right)^{(1-y_j)} \cdot exp\left(-\frac{1}{2}\left(\frac{\theta_0}{10}\right)^2\right) \cdot exp\left(-\frac{1}{2}\left(\frac{\theta_1}{10}\right)^2\right)$$

```
#Establish the function for logistic regression
Expit<-function(x){exp(x)/(1+exp(x))}

#Construct the posterior - in a log-format
#To make sure that the estimate of theta_1 is stable,
#the following code wants to make sure that theta_0 is always greater than zero.

log_post<-function(Y, X, theta)
  {
  if(theta[1]<=0){post=-Inf}
  if(theta[1]>0){
  prob1<-Expit(-theta[1]*((theta[2]-X)^2))
  likelihood<-sum(dbinom(Y,1,prob1,log = TRUE))
  priors<-sum(dnorm(theta,0,10,log=TRUE))
  post=likelihood+priors}
  return(post)
   }

Bayes_logit<-function (Y,X,n_samples=2000)
{
#Initial values
  theta<-c(5,5)
#store data
  keep.theta<-matrix(0,n_samples,2)
  keep.theta[1,]<-theta

#acceptance and rejection
  acc<-att<-rep(0,2)
#current log posterior
  current_lp<-log_post(Y,X,theta)

  for (i in 2:n_samples)
  {

    for(j in 1:2)
    {
      #attempt + 1
      att[j]<-att[j]+1
```

```
      can_theta<-theta
      can_theta[j]<-rnorm(1,theta[j],0.5)
      #candidate of log posterior
      candidate_lp<-log_post(Y,X,can_theta)
      Rho<-min(exp(candidate_lp-current_lp),1)
      Random_probability<-runif(1)
      if (Random_probability<Rho)
      {
        theta<-can_theta
        current_lp<-candidate_lp
        #acceptance + 1, as long as Random_probability<Rho
        acc[j]<-acc[j]+1
      }
    }
    #save theta
    keep.theta[i,]<-theta
  }
#Return: including theta and acceptance rate
  list(theta=keep.theta,acceptance_rate=acc/att)
}
```

7.2 Simulating Data of Senators on Twitter

Assume that we have 100 senators, 50 Democrats and 50 Republicans, who we know their ideology. Assume that Democrats have negative ideology scores to indicate that they are more liberal, whereas Republicans have positive scores to indicate that they are more conservative. The following is data simulation for senators.

```
# Republicans are more conservative, and they have positive numbers.
Republicans<-c()
Republicans<-rnorm(50,1,0.5)
No_Republicans<-rep(1:50,1)
Part_1<-cbind(No_Republicans,Republicans)

# Democrats are more liberal, and they have negative numbers.
Democrats<-c()
Democrats<-rnorm(50,-1,0.5)
No_Democrats<-rep(51:100,1)
Part_2<-cbind(No_Democrats,Democrats)
Data_Elites<-rbind(Part_1,Part_2)
Data_Elites<-as.data.frame(Data_Elites)
colnames(Data_Elites) <- c("Elite_No","Elite_ideology")
```

```
head(Data_Elites)
```

```
##   Elite_No Elite_ideology
## 1        1      1.0541992
## 2        2      0.3805544
## 3        3      1.3568577
## 4        4      0.9922547
## 5        5      1.0089966
## 6        6      0.8878271
```

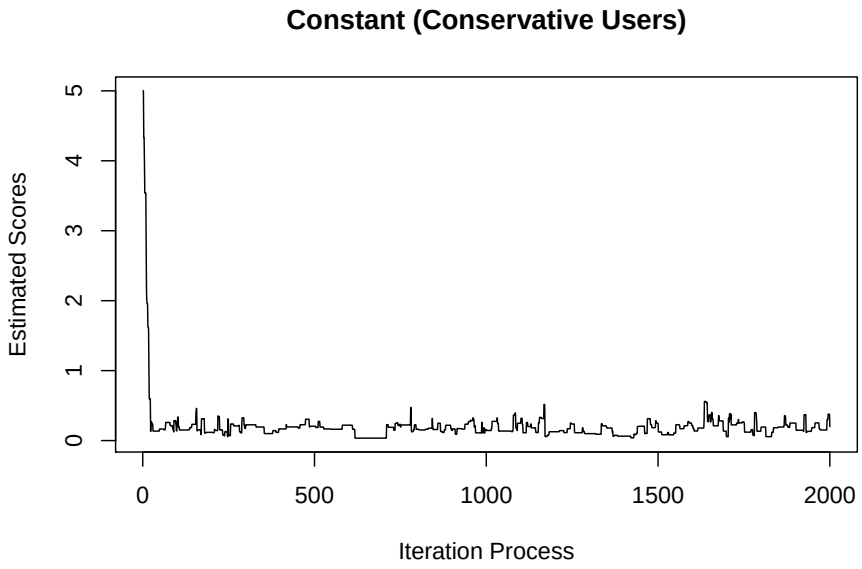
7.3 Simulating Data of Conservative Users on Twitter and Model Testing

Assume that we observe one Twitter user, who is more conservative. To simulate Twitter following data for this user, I assign this user to follow more Republican senators. Thus, if the Metropolis Hastings algorithm works as intended, we would expect to see a positive estimated value for their ideology. Importantly, as we can see in the histogram below, the estimated value indeed is positive, providing preliminary evidence for the statistical model and the algorithm. In addition, for the acceptance rate, we can see that the constant has a lower number than ideology, since we only accept a constant when it is positive.

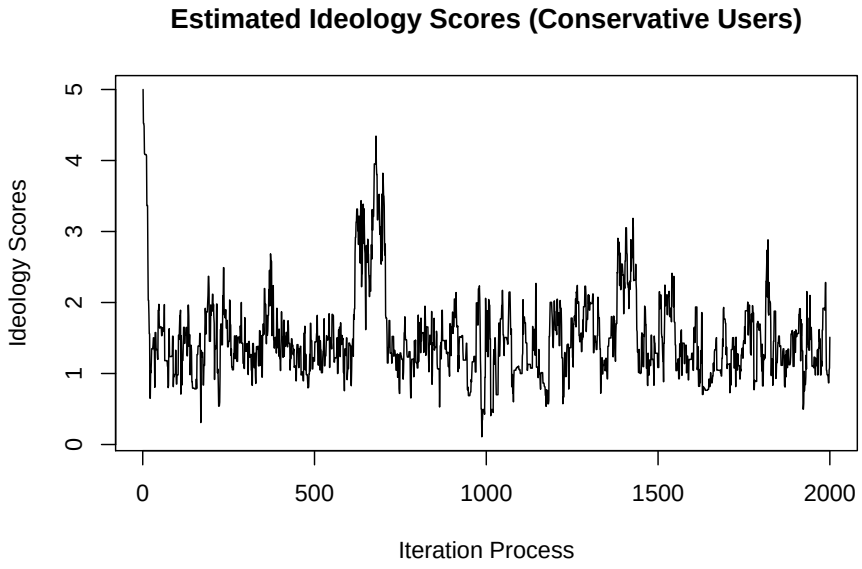
```
#This user approximately follows 45 Republican Senators and 10 Democrat Senators.
Data_user<-as.data.frame(matrix(c(ifelse(runif(50)<.1,0,1),ifelse(runif(50)<.8,0,1))),
colnames(Data_user)<-c("R_User")
Data_combined<-cbind(Data_Elites,Data_user)

X_data<-Data_combined$Elite_ideology
Y_data<-Data_combined$R_User

fit_C<-Bayes_logit(Y_data,X_data)
fit_C$acceptance_rate
```

```
## [1] 0.1320660 0.5557779
```

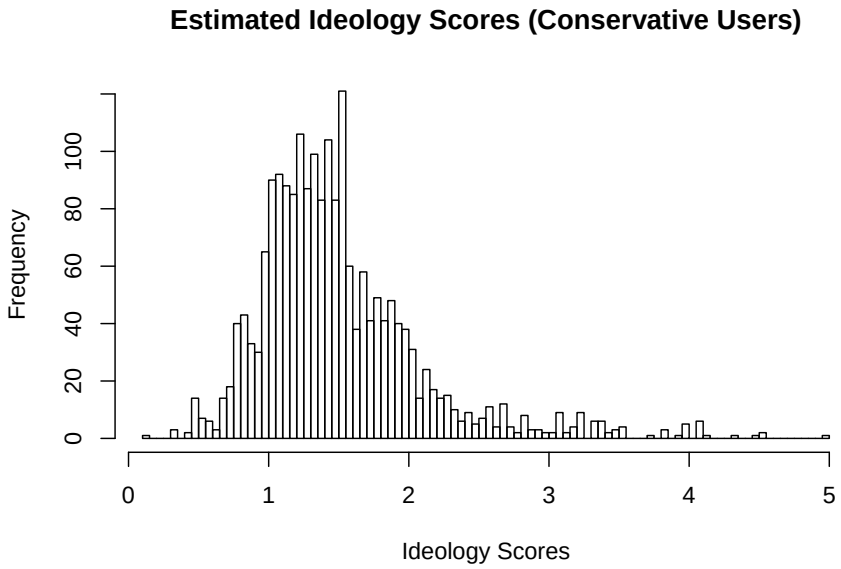
```
plot(fit_C$theta[,1],main="Constant (Conservative Users)",
     xlab="Iteration Process",ylab="Estimated Scores",type="l")
```




```
plot(fit_C$theta[,2],main="Estimated Ideology Scores (Conservative Users)",
     xlab="Iteration Process",ylab="Ideology Scores",type="l")
```



```
hist(fit_C$theta[,2],main="Estimated Ideology Scores (Conservative Users)",
     xlab="Ideology Scores",breaks = 100)
```



7.4 Simulating Data of Liberal Users on Twitter and Model Testing

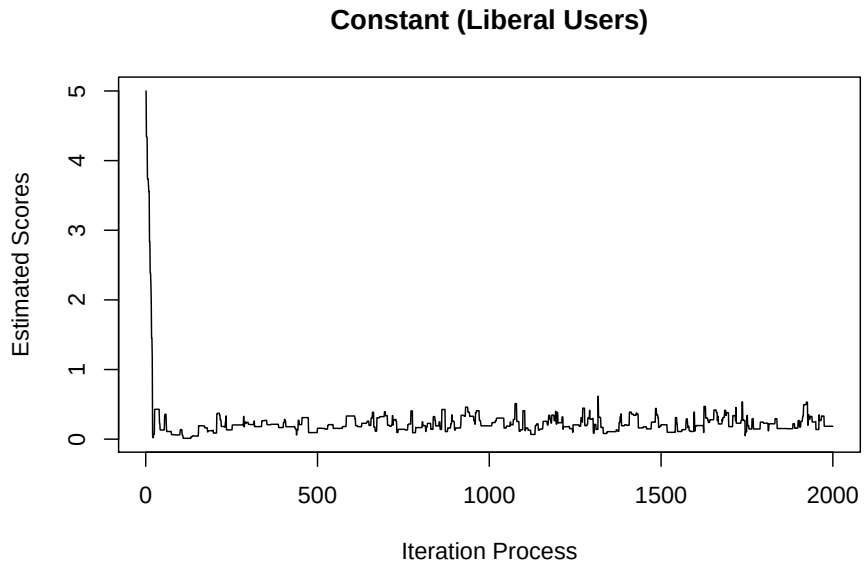
To further verify the Metropolis Hastings algorithm, I plan to test the opposite estimate. Specifically, assume that we observe another user, who is more liberal. To simulate Twitter following data for this user, I assign this user to follow more Democrat senators. In this case, we would expect to see a negative value for their estimated ideology. As we can see in the histogram shown below, as expected, the estimated value is negative, providing convergent evidence for the model and the algorithm.

```
#This user approximately follows 10 Republican Senators and 45 Democrat Senators.
Data_user<-as.data.frame(matrix(c(ifelse(runif(50)<.8,0,1),ifelse(runif(50)<.1,0,1))),
colnames(Data_user)<-c("L_User")
Data_combined<-cbind(Data_Elites,Data_user)

X_data<-Data_combined$Elite_ideology
Y_data<-Data_combined$L_User
```

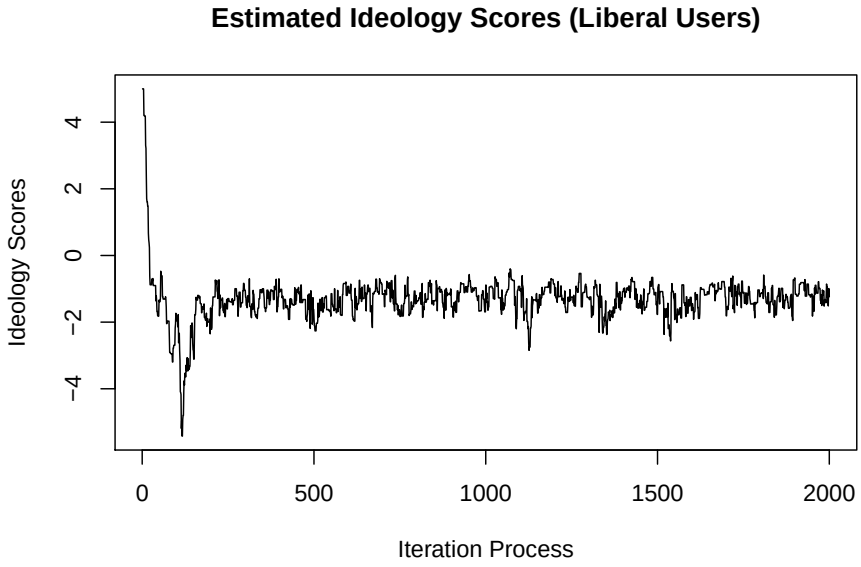
```
fit_L<-Bayes_logit(Y_data,X_data)
fit_L$acceptance_rate
```

```
## [1] 0.1585793 0.5092546
```

```
plot(fit_L$theta[,1],main="Constant (Liberal Users)",
     xlab="Iteration Process",ylab="Estimated Scores",type="l")
```




```
plot(fit_L$theta[,2],main="Estimated Ideology Scores (Liberal Users)",
     xlab="Iteration Process",ylab="Ideology Scores",type="l")
```




```
hist(fit_L$theta[,2],main="Estimated Ideology Scores (Liberal Users)",
     xlab="Ideology Scores",breaks = 100)
```