# GLMM, Concepts, & R

Bill Last Updated:

03 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 erros. Please feel free to let me know in case you find one. Thank you!

# Chapter 1

# Basics

## 1.1 Logit

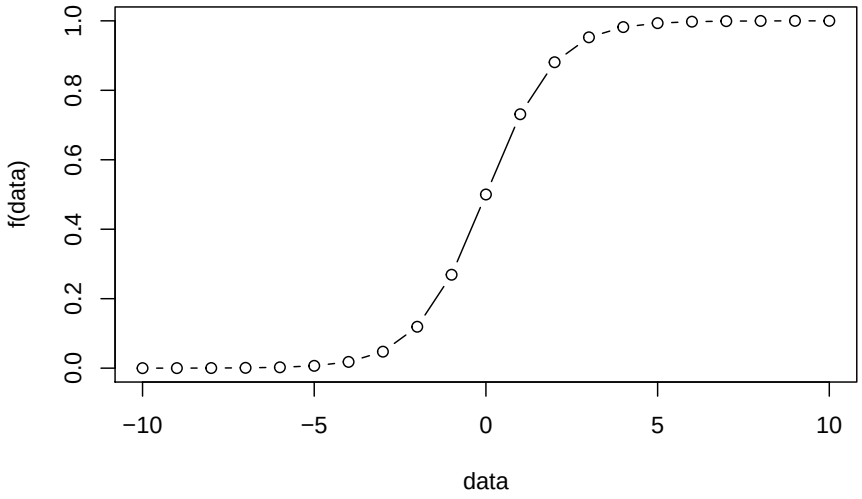
$$f(x) = log(\frac{p(y=1)}{1-p(y=1)})$$

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 $\beta_i$ actually change the "log-odds" of the event. Of course, note that the magnitude of $\beta_i$ 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)
```

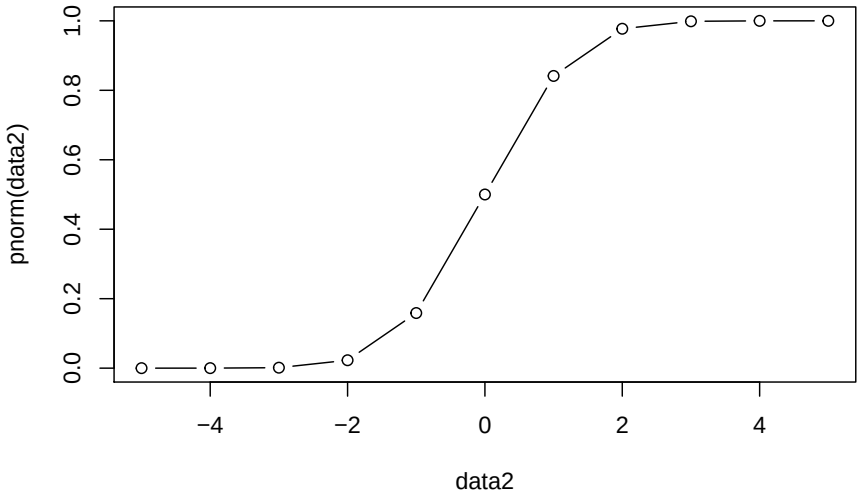
```
##
## Call:
```

```
## 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 $\beta$ and $\sigma$ 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

# MLE

## 2.1 Basic idea of MLE

Suppose that we flip a coin, $y_i = 0$ for tails and $y_i = 1$ for heads. If we get $p$ heads from $n$ trials, we can get the proportion of heads is $p/n$, which is the sample mean. If we do not do any further calculation, this is our best guess.

Suppose that the true proablity is $\rho$, then we can get:

$$\mathbf{L}(y_i) = \begin{cases} \rho & y_i = 1 \\ 1 - \rho & y_i = 0 \end{cases}$$

Thus, we can also write it as follows.

$$\mathbf{L}(y_i) = \rho^{y_i}(1-\rho)^{1-y_i}$$

Thus, we can get:

$$\prod \mathbf{L}(y_i|\rho) = \rho^{\sum y_i}(1-\rho)^{\sum(1-y_i)}$$

Further, we can get a log-transformed format.

$$log(\prod \mathbf{L}(y_i|\rho)) = \sum y_i log\rho + \sum(1-y_i)log(1-\rho)$$

To maximize the log-function above, we can calculate the derivative with respect to $\rho$.

$$\frac{\partial log(\prod \mathbf{L}(y_i|\rho))}{\partial \rho} = \sum y_i \frac{1}{\rho} - \sum(1-y_i)\frac{1}{1-\rho}$$

Set the derivative to zero and solve for $\rho$, we can get

$$\sum y_i \frac{1}{\rho} - \sum (1 - y_i) \frac{1}{1 - \rho} = 0$$

$$\Rightarrow (1 - \rho) \sum y_i - \rho \sum (1 - y_i) = 0$$

$$\Rightarrow \sum y_i - \rho \sum y_i - n\rho + \rho \sum y_i = 0$$

$$\Rightarrow \sum y_i - n\rho = 0$$

$$\Rightarrow \rho = \frac{\sum y_i}{n} = \frac{p}{n}$$

Thus, we can see that the $\rho$ maximizing the likelihood function is equal to the sample mean.

## 2.2 Coin flip example, probit, and logit

In the example above, we are not really trying to estimate a lot of regression coefficients. What we are doing actually is to calculate the sample mean, or intercept in the regresion sense. What does it mean? Let's use some data to explain it.

Suppose that we flip a coin 20 times and observe 8 heads. We can use the R's glm function to esimate the $\rho$. If the result is consistent with what we did above, we should observe that the *cdf* of the esimate of $\beta_0$ (i.e., intercept) should be equal to $8/20 = 0.4$.

```
coins<-c(rep(1,times=8),rep(0,times=12))
table(coins)
```

```
## coins
##  0  1
## 12  8
```

```
coins<-as.data.frame(coins)
```

### 2.2.1 Probit

```
probitresults <- glm(coins ~ 1, family = binomial(link = "probit"), data = coins)
probitresults
```

```
##
## Call:  glm(formula = coins ~ 1, family = binomial(link = "probit"),
##     data = coins)
##
## Coefficients:
## (Intercept)
##     -0.2533
##
## Degrees of Freedom: 19 Total (i.e. Null);  19 Residual
## Null Deviance:       26.92
## Residual Deviance: 26.92     AIC: 28.92
```

```
pnorm(probitresults$coefficients)
```

```
## (Intercept)
##         0.4
```

As we can see the intercept is $-0.2533$, and thus $\Phi(-0.2533471) = 0.4$

### 2.2.2 Logit

We can also use logit link to calculate the intercept as well. Recall that

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

Thus,

$$p(y=1) = \frac{e^{\beta_0}}{1+e^{\beta_0}}$$

```
logitresults <- glm(coins ~ 1, family = binomial(link = "logit"), data = coins)
logitresults$coefficients
```

```
## (Intercept)
##  -0.4054651
```

```
exp(logitresults$coefficients)/(1+exp(logitresults$coefficients))
```

```
## (Intercept)
##         0.4
```

Note that, the defaul link for the binomial in the glm function in logit.

## 2.3 Further on logit

The probablity of $y = 1$ is as follows:

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

Thus, the likelihood function is as follows:

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

$$= \prod (1 + e^{-(\beta_0+\beta_1 x_1+...+\beta_n x_n)})^{-y_i} (1 + e^{\beta_0+\beta_1 x_1+...+\beta_n x_n})^{-(1-y_i)}$$

Thus, the log-likelihood is as follows:

$$logL = \sum (-y_i \cdot log(1+e^{-(\beta_0+\beta_1 x_1+...+\beta_n x_n)}) - (1-y_i) \cdot log(1+e^{\beta_0+\beta_1 x_1+...+\beta_n x_n}))$$

Typically, optimisers minimize a function, so we use negative log-likelihood as minimising that is equivalent to maximising the log-likelihood or the likelihood itself.

```
#Source of R code: https://www.r-bloggers.com/logistic-regression/

mle.logreg = function(fmla, data)
{
  # Define the negative log likelihood function
  logl <- function(theta,x,y){
    y <- y
    x <- as.matrix(x)
    beta <- theta[1:ncol(x)]

    # Use the log-likelihood of the Bernouilli distribution, where p is
    # defined as the logistic transformation of a linear combination
    # of predictors, according to logit(p)=(x%*%beta)
    loglik <- sum(-y*log(1 + exp(-(x%*%beta))) - (1-y)*log(1 + exp(x%*%beta)))
    return(-loglik)
  }

  # Prepare the data
  outcome = rownames(attr(terms(fmla),"factors"))[1]
  dfrTmp = model.frame(data)
  x = as.matrix(model.matrix(fmla, data=dfrTmp))
```

```
  y = as.numeric(as.matrix(data[,match(outcome,colnames(data))]))

  # Define initial values for the parameters
  theta.start = rep(0,(dim(x)[2]))
  names(theta.start) = colnames(x)

  # Calculate the maximum likelihood
  mle = optim(theta.start,logl,x=x,y=y, method = 'BFGS', hessian=T)
  out = list(beta=mle$par,vcov=solve(mle$hessian),ll=2*mle$value)
}
```

```
mydata = read.csv(url('https://stats.idre.ucla.edu/stat/data/binary.csv'))
mylogit1 = glm(admit~gre+gpa+as.factor(rank), family=binomial, data=mydata)

mydata$rank = factor(mydata$rank) #Treat rank as a categorical variable
fmla = as.formula("admit~gre+gpa+rank") #Create model formula
mylogit2 = mle.logreg(fmla, mydata) #Estimate coefficients


 print(cbind(coef(mylogit1), mylogit2$beta))
```

```
##                           [,1]         [,2]
## (Intercept)      -3.989979073 -3.772676422
## gre               0.002264426  0.001375522
## gpa               0.804037549  0.898201239
## as.factor(rank)2 -0.675442928 -0.675543009
## as.factor(rank)3 -1.340203916 -1.356554831
## as.factor(rank)4 -1.551463677 -1.563396035
```

## 2.4 References

http://www.columbia.edu/~so33/SusDev/Lecture_9.pdf

# Chapter 3

# Linear Mixed Models

## 3.1 LM and GLM

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

### 3.1.1 LM

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

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

where,

$$\mu(X) : randomcomponent$$

$$X^T\beta : covariates$$

### GLM

$$Y \sim exponential family$$

Link function

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

Possion regression:

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

Link function is log link, and it becomes:

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

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

Reciprocal link:

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

## 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 Stat Concepts

## 4.1 Score

The score is the gradient (the vector of partial derivatives) of $logL(\theta)$, with respect to an m-dimensional parameter vector $\theta$.

$$S(\theta) = \frac{\partial \ell}{\partial \theta}$$

Typically, they use $\nabla$ to denote the partical derivative.

$$\nabla \ell$$

Such differentiation will generate a $m \times 1$ row vector, which indicates the sensitivity of the likelihood.

Quote from Steffen Lauritzen's slides: "Generally the solution to this equation must be calculated by iterative methods. One of the most common methods is the Newton–Raphson method and this is based on successive approximations to the solution, using Taylor's theorem to approximate the equation."

For instance, using logit link, we can get the first derivative of log likelihood logistic regression as follows. We can not really find $\beta$ easily to make the equation to be 0.

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

## 4.2 Gradient and Jacobian

**Remarks**: This part discusses gradient in a more general sense.

When $f(x)$ is only in a single dimension space:

$\mathbb{R}^n \to \mathbb{R}$

$$\nabla f(x) = [\frac{\partial f}{\partial x_1}, \frac{\partial f}{\partial x_2}, ..., \frac{\partial f}{\partial x_n}]$$

When $f(x)$ is only in a m-dimension space (i.e., Jacobian): $\mathbb{R}^n \to \mathbb{R}^{\geqslant}$

$$Jac(f) = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} & \frac{\partial f_1}{\partial x_3} & \cdots & \frac{\partial f_1}{\partial x_n} \\ \frac{\partial f_2}{\partial x_1} & \frac{\partial f_2}{\partial x_2} & \frac{\partial f_2}{\partial x_3} & \cdots & \frac{\partial f_2}{\partial x_n} \\ \cdots & & & & \\ \frac{\partial f_m}{\partial x_1} & \frac{\partial f_m}{\partial x_2} & \frac{\partial f_n}{\partial x_3} & \cdots & \frac{\partial f_m}{\partial x_n} \end{bmatrix}$$

For instance,

$\mathbb{R}^n \to \mathbb{R}$:

$$f(x, y) = x^2 + 2y$$
$$\nabla f(x, y) = [\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}] = [2x, 2]$$

$\mathbb{R}^n \to \mathbb{R}^{\geqslant}$

$$f(x, y) = (x^2 + 2y, x^3)$$
$$Jac(f) = \begin{bmatrix} 2x & 2 \\ 2x^2 & 0 \end{bmatrix}$$

## 4.3 Hessian and Fisher Information

Hessian matrix or Hessian is a square matrix of second-order partial derivatives of a scalar-valued function, or scalar field.

$\mathbb{R}^n \to \mathbb{R}$

$$Hessian = \nabla^2(f) = \begin{bmatrix} \frac{\partial^2 f}{\partial x_1^2} & \frac{\partial^2 f}{\partial x_1 \partial x_2} & \frac{\partial^2 f}{\partial x_1 \partial x_3} & \cdots & \frac{\partial^2 f}{\partial x_1 \partial x_n} \\ \frac{\partial^2 f}{\partial x_2 \partial x_1} & \frac{\partial^2 f}{\partial x_2^2} & \frac{\partial^2 f}{\partial x_2 \partial x_3} & \cdots & \frac{\partial^2 f}{\partial x_2 \partial x_n} \\ \frac{\partial^2 f}{\partial x_3 \partial x_1} & \frac{\partial^2 f}{\partial x_3 \partial x_2} & \frac{\partial^2 f}{\partial x_3^2} & \cdots & \frac{\partial^2 f}{\partial x_3 \partial x_n} \\ \cdots & & & & \\ \frac{\partial^2 f}{\partial x_n \partial x_1} & \frac{\partial^2 f}{\partial x_n \partial x_2} & \frac{\partial^2 f}{\partial x_n \partial x_3} & \cdots & \frac{\partial^2 f}{\partial x_n^2} \end{bmatrix}$$

As a special case, in the context of logit:

Suppose that the log likelihood function is $\ell(\theta)$. $\theta$ is a $m$ demension vector.

$$\theta = \begin{bmatrix} \theta_1 \\ \theta_2 \\ \theta_3 \\ \theta_4 \\ ... \\ \theta_m \end{bmatrix}$$

$$Hessian = \nabla^2(\ell) = \begin{bmatrix} \frac{\partial^2 \ell}{\partial \theta_1^2} & \frac{\partial^2 \ell}{\partial \theta_1 \partial \theta_2} & \frac{\partial^2 \ell}{\partial \theta_1 \partial \theta_3} & \cdots & \frac{\partial^2 \ell}{\partial \theta_1 \partial \theta_m} \\ \frac{\partial^2 \ell}{\partial \theta_2 \partial \theta_1} & \frac{\partial^2 \ell}{\partial \theta_2^2} & \frac{\partial^2 \ell}{\partial \theta_1 \partial \theta_3} & \cdots & \frac{\partial^2 \ell}{\partial \theta_1 \partial \theta_m} \\ \frac{\partial^2 \ell}{\partial \theta_3 \partial \theta_1} & \frac{\partial^2 \ell}{\partial \theta_3 \theta_2} & \frac{\partial^2 \ell}{\partial \theta_3^2} & \cdots & \frac{\partial^2 \ell}{\partial \theta_3 \partial \theta_m} \\ ... & & & & \\ \frac{\partial^2 \ell}{\partial \theta_m \partial \theta_1} & \frac{\partial^2 \ell}{\partial \theta_m \theta_2} & \frac{\partial^2 \ell}{\partial \theta_m \partial \theta_3} & \cdots & \frac{\partial^2 \ell}{\partial \theta_m \partial \theta_m} \end{bmatrix}$$

"In statistics, the observed information, or observed Fisher information, is the negative of the second derivative (the Hessian matrix) of the"log-likelihood" (the logarithm of the likelihood function). It is a sample-based version of the Fisher information." (Direct quote from Wikipedia.)

Thus, the observed information matrix:

$$-Hessian = -\nabla^2(\ell)$$

Expected (Fisher) information matrix:

$$E[-\nabla^2(\ell)]$$

## 4.4 Canonical link function

Inspired by a Stack Exchange post, I created the following figure:

$$\frac{Paramter}{\theta} \longrightarrow \gamma^{'}(\theta) = \mu \longrightarrow \frac{Mean}{\mu} \longrightarrow g(\mu) = \eta \longrightarrow \frac{Linearpredictor}{\eta}$$

For the case of $n$ time Bernoulli (i.e., Binomial), its canonical link function is logit. Specifically,

$$\frac{Paramter}{\theta = \beta^T x_i} \longrightarrow \gamma^{'}(\theta) = \frac{e^{\beta^T x_i}}{1 + e^{\beta^T x_i}} \longrightarrow \frac{Mean}{\mu = \frac{e^{\beta^T x_i}}{1+e^{\beta^T x_i}}} \longrightarrow g(\mu) = log\frac{\frac{e^{\beta^T x_i}}{1+e^{\beta^T x_i}}}{1 - \frac{e^{\beta^T x_i}}{1+e^{\beta^T x_i}}} \longrightarrow \frac{Linearpredictor}{\eta = \beta^T x_i}$$

Thus, we can see that,

$$\theta \equiv \eta$$

The link function $g(\mu)$ relates the linear predictor $\eta = \beta^T x_i$ to the mean $\mu$.

**Remarks**:

(1) Parameter is $\theta = \beta^T x_i$ (Not $\mu$!).

(2) $\mu = p(y = 1) = \frac{e^{\beta^T x_i}}{1+e^{\beta^T x_i}}$ (Not logit!).

(3) Link function (i.e., $g(\mu)$) = logit = logarithm of odds = log $\frac{Event-Happened}{Event-Not-Happened}$.

(4) $g(\mu) = log\frac{\mu}{1-\mu} = \beta^T x_i$. Thus, link function = linear predictor = log odds!

(5) Quote from the Stack Exchange post "Newton Method and Fisher scoring for finding the ML estimator coincide, these links simplify the derivation of the MLE."

(Recall, we know that $\mu$ or $p(y = 1)$ is the mean function. Recall that, $n$ trails of coin flips, and get $p$ heads. Thus $\mu = \frac{p}{n}$.)

## 4.5 Ordinary Least Squares (OLS)

Suppose we have $n$ observation, and $m$ variables.

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

Thus, we can write it as the following $n$ equations.

$$\begin{aligned} y_1 &= \beta_0 + \beta_1 x_{11} + \beta_2 x_{12} + ... + \beta_m x_{1m} \\ y_2 &= \beta_0 + \beta_1 x_{21} + \beta_2 x_{22} + ... + \beta_m x_{2m} \\ y_3 &= \beta_0 + \beta_1 x_{31} + \beta_2 x_{32} + ... + \beta_m x_{3m} \end{aligned}$$

...

$$y_n = \beta_0 + \beta_1 x_{n1} + \beta_2 x_{n2} + ... + \beta_m x_{nm}$$

We can combine all the $n$ equations as the following one:

$$y_i = \beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + ... + \beta_m x_{im} (i \in [1, n])$$

We can further rewrite it as a matrix format as follows.

$$y = X\beta$$

Where,

$$y = \begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \\ ... \\ y_n \end{bmatrix}$$

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

$$\beta = \begin{bmatrix} \beta_0 \\ \beta_1 \\ \beta_2 \\ \beta_3 \\ ... \\ \beta_m \end{bmatrix}$$

Since later we need the inverse of $X$, we need to make it into a square matrix.

$$X^T y = X^T X \hat{\beta} \Rightarrow \hat{\beta} = (X^T X)^{-1} X^T y$$

We can use R to implement this calculation. As we can see, there is no need to do any iterations at all, but rather just pure matrix calculation.

```r
X<-matrix(rnorm(1000),ncol=2) # we define a 2 column matrix, with 500 rows
X<-cbind(1,X) # add a 1 constant
beta_true<-c(2,1,2) # True regression coefficients
beta_true<-as.matrix(beta_true)
y=X%*%beta_true+rnorm(500)

transposed_X<-t(X)
beta_hat<-solve(transposed_X%*%X)%*%transposed_X%*%y
beta_hat
```

```
##          [,1]
## [1,] 2.017690
## [2,] 1.054682
## [3,] 2.037671
```

**Side Notes** The function of as.matrix will automatically make c(2,1,2) become the dimension of $3 \times 1$, you do not need to transpose the $\beta$.

## 4.6 Taylor series

$$\begin{aligned} f(x)|_a &= f(a) + \frac{f^{'}(a)}{1!}(x-a) + \frac{f^{'}(a)}{2!}(x-a)^2 + \frac{f^{''}(a)}{3!}(x-a)^3 + ... \\ &= \sum_{n=0}^{\infty} \frac{f^n(a)}{n!}(x-a)^n \end{aligned}$$

For example:

$$\begin{aligned} e^x|_{a=0} &= e^a + \frac{e^a}{1!}(x-a) + \frac{e^a}{2!}(x-a)^2 + ... + \frac{e^a}{n!}(x-a)^n \\ &= 1 + \frac{1}{1!}x + \frac{1}{2!}x^2 + ... + \frac{1}{n!}x^n \end{aligned}$$

if $x = 2$

$e^2 = 7.389056$

$e^2 \approx 1 + \frac{1}{1!}x = 1 + \frac{1}{1!}2 = 3$

$e^2 \approx 1 + \frac{1}{1!}x + \frac{1}{2!}x^2 = 1 + \frac{1}{1!}2 + \frac{1}{2!}2 = 5$ ...

$e^2 \approx 1 + \frac{1}{1!}x + \frac{1}{2!}x^2 + \frac{1}{3!}x^2 + \frac{1}{4!}x^2 + \frac{1}{5!}x^2 = 7.2666...$

## 4.7 Fisher scoring

[I will come back to this later.]

https://www2.stat.duke.edu/courses/Fall00/sta216/handouts/diagnostics.pdf

https://stats.stackexchange.com/questions/176351/implement-fisher-scoring-for-linear-regression

## 4.8 References

1. Steffen Lauritzen's slides:

http://www.stats.ox.ac.uk/~steffen/teaching/bs2HT9/scoring.pdf

2. The Stack Exchange post:

https://stats.stackexchange.com/questions/40876/what-is-the-difference-between-a-link-function-and-a-canonical-link-function

3. Wilipedia for OLS

https://en.wikipedia.org/wiki/Ordinary_least_squares

4. Gradient and Jacobian

https://math.stackexchange.com/questions/1519367/difference-between-gradient-and-jacobian

https://www.youtube.com/watch?v=3xVMVT-2_t4

https://math.stackexchange.com/questions/661195/what-is-the-difference-between-the-gradient-and-the-directional-derivative

5. Hessian

https://en.wikipedia.org/wiki/Hessian_matrix

6. Observed information

https://en.wikipedia.org/wiki/Observed_information

7. Fisher information

https://people.missouristate.edu/songfengzheng/Teaching/MTH541/Lecture%20notes/Fisher_info.pdf

8. Link function

https://en.wikipedia.org/wiki/Generalized_linear_model#Link_function

https://stats.stackexchange.com/questions/40876/what-is-the-difference-between-a-link-function-and-a-canonical-link-function

# Chapter 5

# Basic R

This section is about R coding.

## 5.1 apply, lapply, sapply

### 5.1.1 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
```

### 5.1.2 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
```

### 5.1.3 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
```

## 5.2 C

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

```
##           [,1]      [,2]
## [1,] 0.7767640 0.3839558
## [2,] 0.8404593 0.9506320
## [3,] 0.8705815 0.7041046
## [4,] 0.9530419 0.4219814
```

```
str(mydata1)
```

```
##  num [1:4, 1:2] 0.777 0.84 0.871 0.953 0.384 ...
```

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

```
## [1] 0.7767640 0.8404593 0.8705815 0.9530419 0.3839558 0.9506320 0.7041046
## [8] 0.4219814
```

```
str(mydata2)
```

```
##  num [1:8] 0.777 0.84 0.871 0.953 0.384 ...
```

# Chapter 6

# 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.

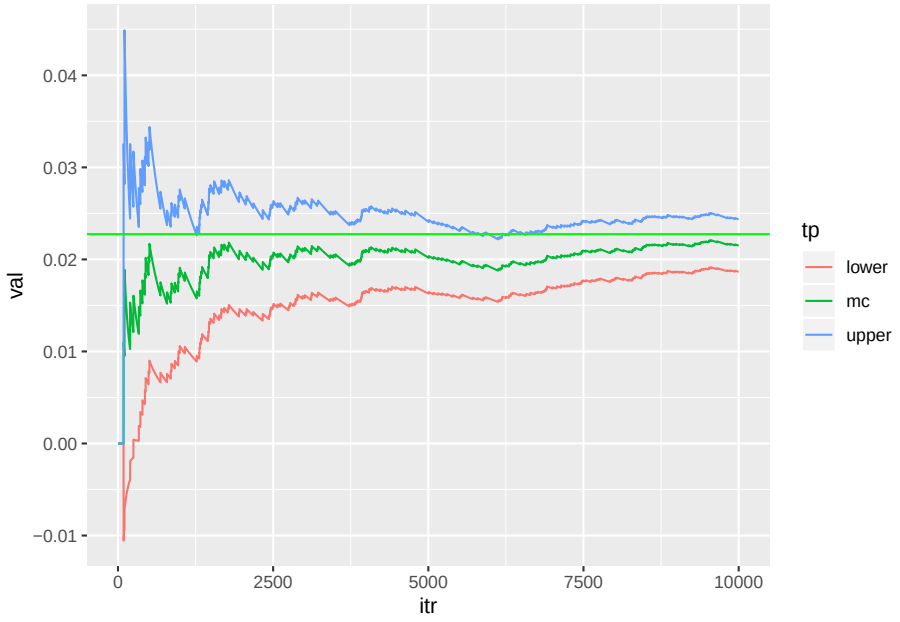
## 6.1 Monte carlo approximation

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).
```




## 6.2 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 $h(x_i)\frac{f(x_i)}{g(x_i)}$.

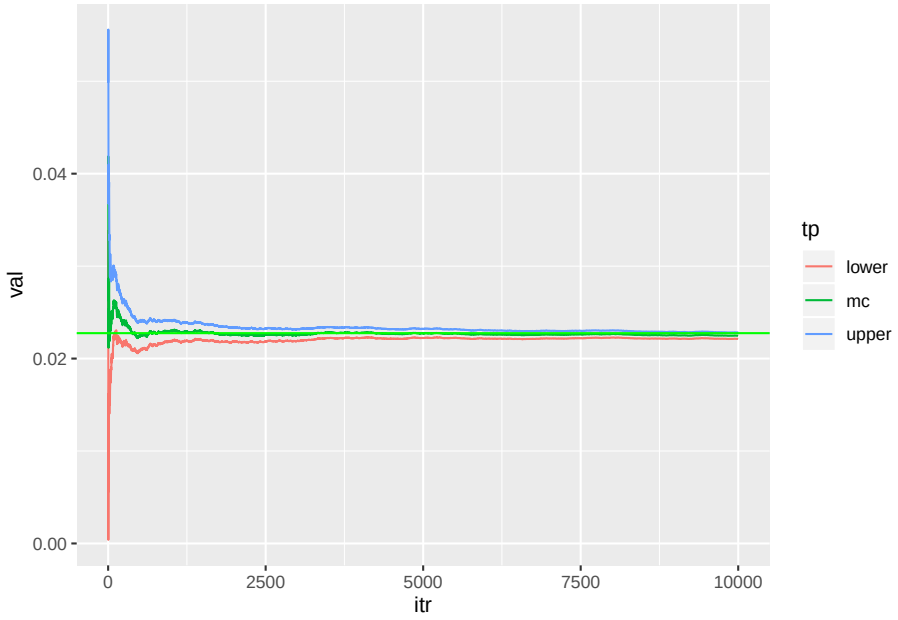
Using the same explane above, we can use a shifted exponential distribution to help calculate the intergral for normal distribution. Specifically,

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

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).
```




## 6.3 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:

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

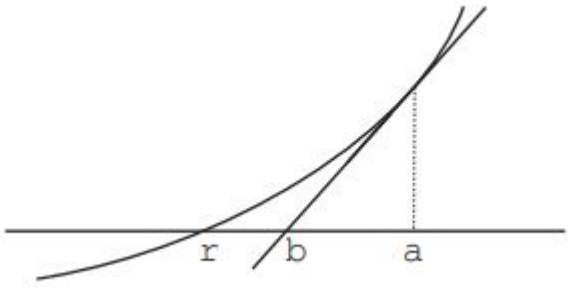



Figure 6.1: Credit of this figure: https://www.math.ubc.ca/~anstee/math104/newtonmethod.pdf

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.

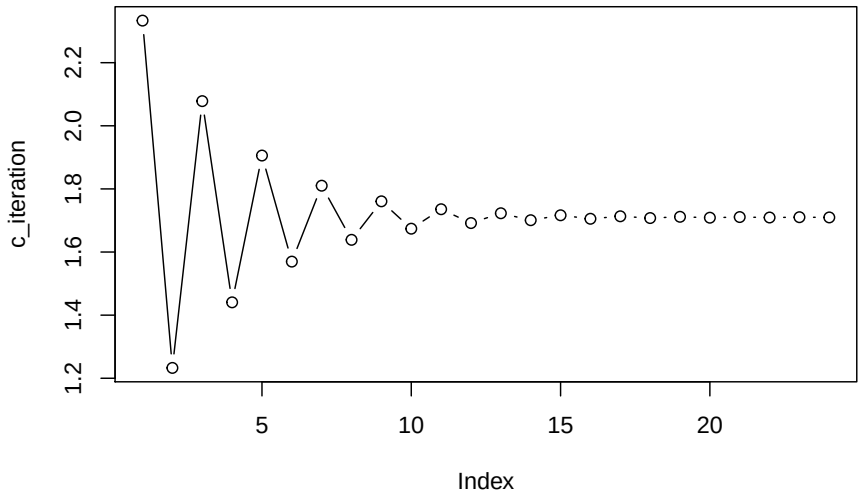
$$\beta_{t+1} = \beta_t - \frac{\ell^{'}(\beta_t)}{\ell^{''}(\beta_t)}$$

### 6.3.1 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.

```
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")
```




### 6.3.2 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")
```

## 6.4 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:

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

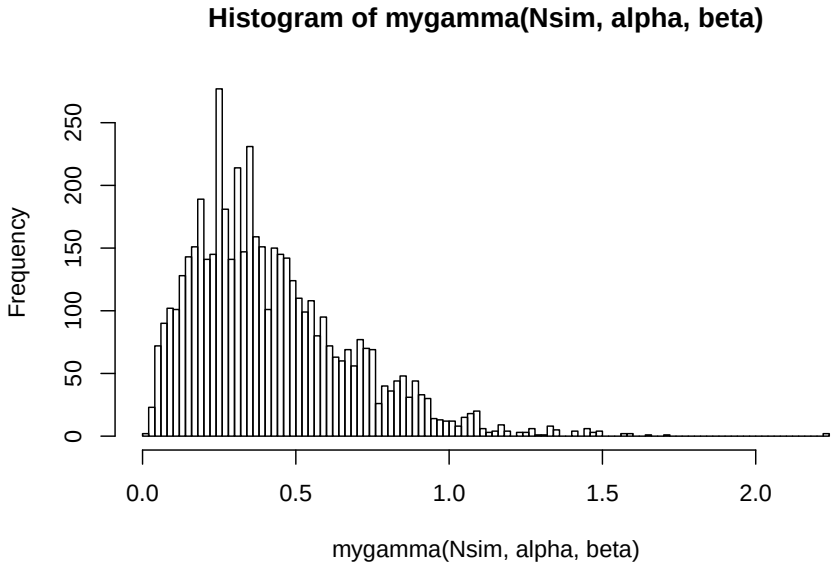
(4) $\rho = min(p, 1)$

(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)
```




## 6.5 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 posterior $p(Z|X,\theta)$. 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 $\theta_0$, 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 $\theta_i$ and $\theta_{i+1}$ 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.)

## 6.6 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 7

# Generalized Linear Mixed Models

## 7.1 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 $\epsilon$ 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"

## 7.2 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 8

# 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.

## 8.1 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 $\theta$.

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

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)
}
```

## 8.2 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
```

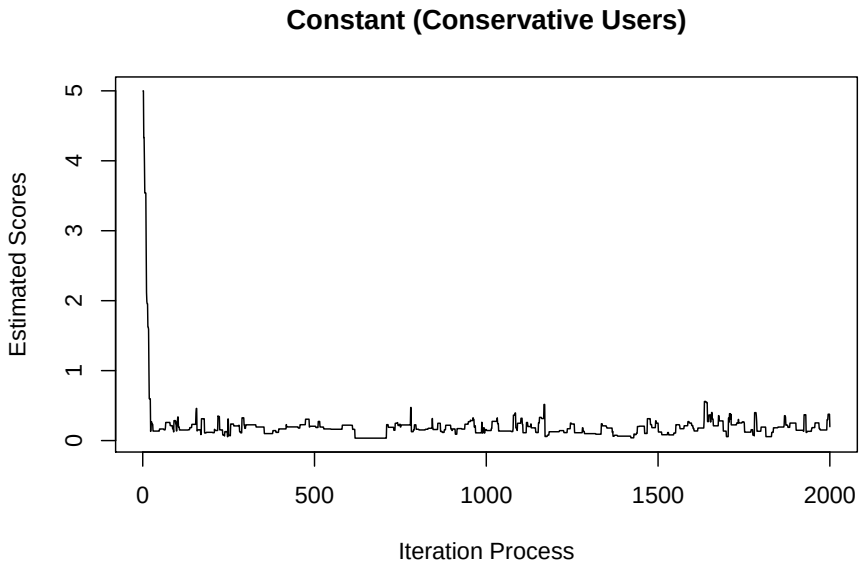
## 8.3 Simulating Data of Conservative Users on Twitter and Model Testing

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.

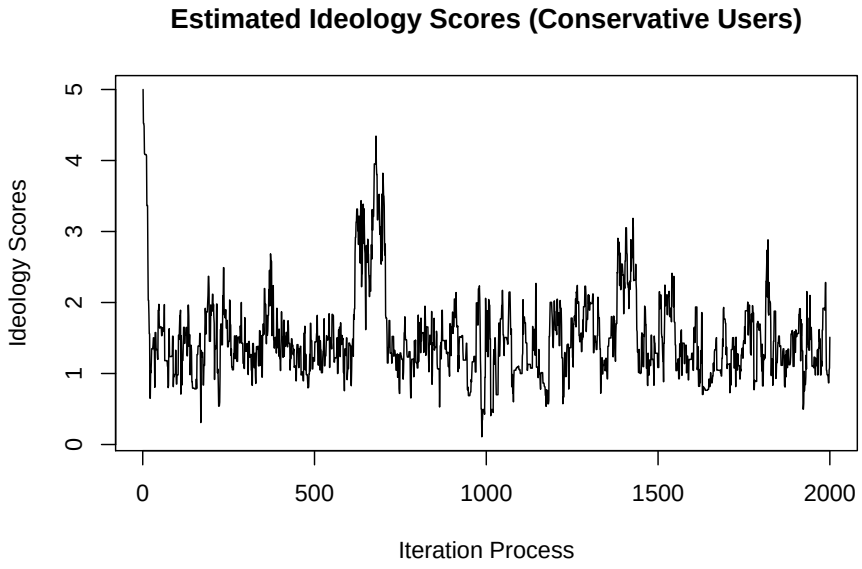
```
#This user approximately follows 45 Republican Senators and 10 Democrat Senators.
Data_user<-as.data.frame(matrix(c(ifelse(runif(50)<.1,0,1),ifelse(runif(50)<.8,0,1))),
colnames(Data_user)<-c("R_User")
Data_combined<-cbind(Data_Elites,Data_user)

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
```

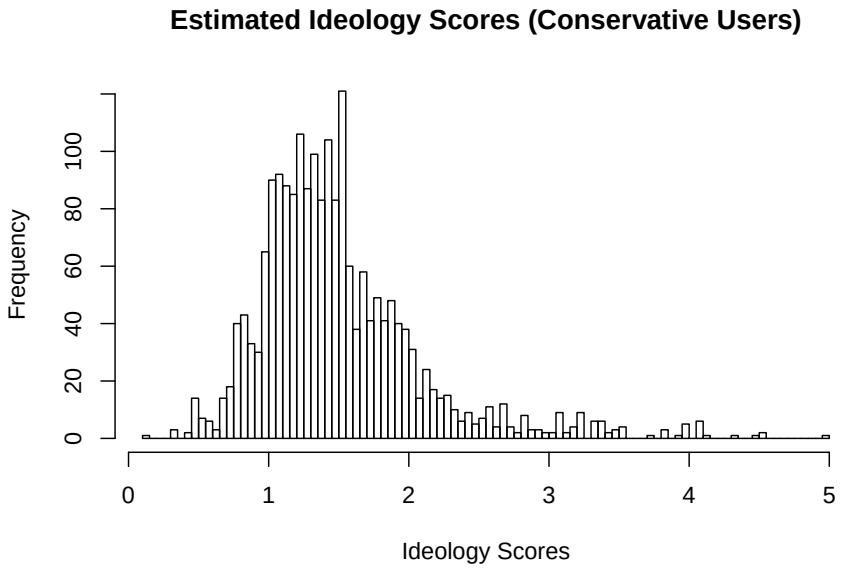
```
plot(fit_C$theta[,1],main="Constant (Conservative Users)",
     xlab="Iteration Process",ylab="Estimated Scores",type="l")
```




```
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)
```




## 8.4 Simulating Data of Liberal Users on Twitter and Model Testing

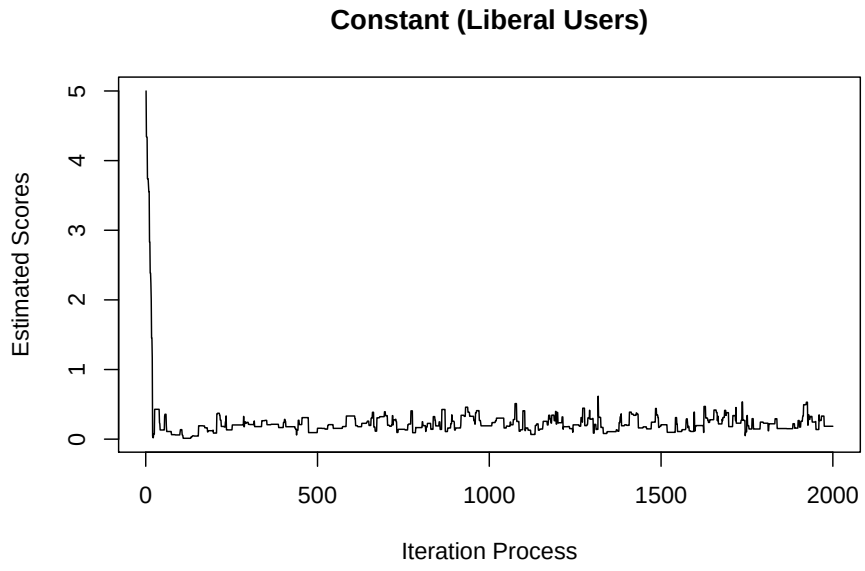
To further verify the Metropolis Hastings algorithm, I plan to test the opposite estimate. Specifically, assume that we observe another user, who is more liberal. To simulate Twitter following data for this user, I assign this user to follow more Democrat senators. In this case, we would expect to see 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
```

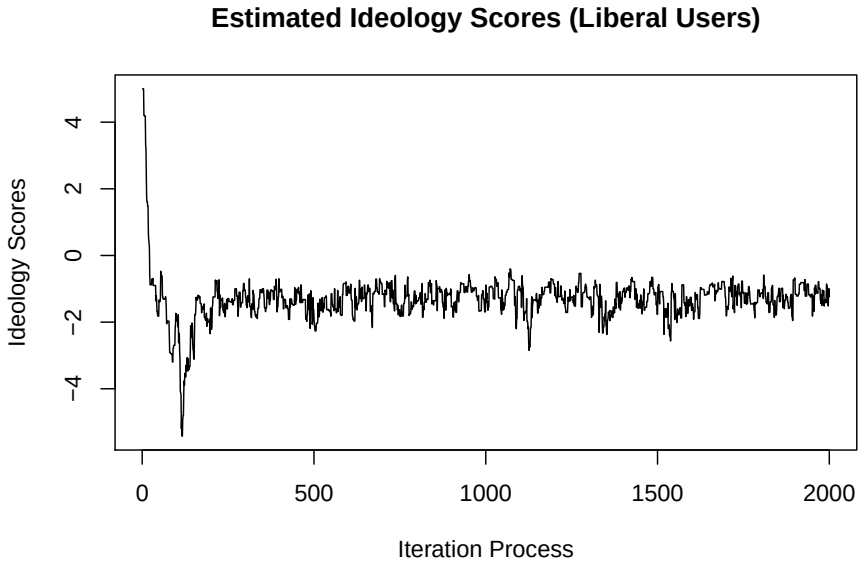
```
fit_L<-Bayes_logit(Y_data,X_data)
fit_L$acceptance_rate
```

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

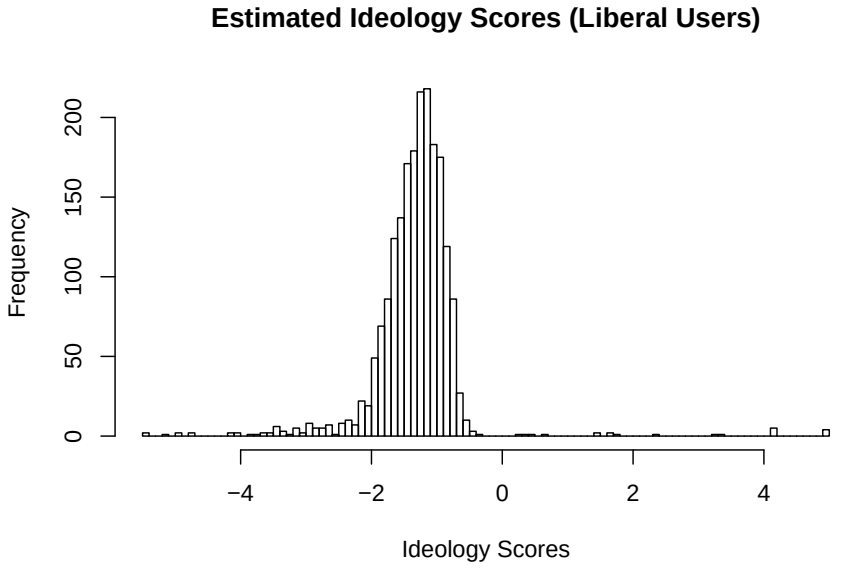
```
plot(fit_L$theta[,1],main="Constant (Liberal Users)",
     xlab="Iteration Process",ylab="Estimated Scores",type="l")
```




```
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)
```



# Chapter 9

# Practice: Learning on the Battle Field

## 9.1 R code

```
#https://fivethirtyeight.com/contributors/josh-hermsmeyer/
# https://github.com/ryurko/nflscrapR-data/blob/master/legacy_data/README.md

#mydata1 = read.csv('plays.txt')
#unique(mydata1$gameId)

#unique(mydata1$PassLength)
#table(mydata1$PassLength)
#table(mydata1$PassResult)
#table(mydata1$numberOfPassRushers)


##mydata3 = read.csv(url('https://raw.githubusercontent.com/ryurko/nflscrapR-data/master/legacy_d
##write.csv(mydata3,'2017playbyplay.csv')

mydata3<-read.csv('2017playbyplay.csv')
nrow(mydata3)
table(mydata3$Passer)
table(mydata3$PlayType)

#mydata5<-mydata3[!duplicated(mydata3[,c('GameID','Passer')]),]
#unique(mydata3$GameID)
mydata6<-subset(mydata3,down==1)
```

```
mydata7<-subset(mydata6,PlayType=='Pass'|PlayType=='Run')
#table(mydata7$PlayType)
#table(droplevels(mydata7$PlayType))

mydata7$PlayType<-droplevels(mydata7$PlayType)
table(mydata7$PlayType)

#http://rstudio-pubs-static.s3.amazonaws.com/6975_c4943349b6174f448104a5513fed59a9.htm
source("http://pcwww.liv.ac.uk/~william/R/crosstab.r")
mydata8<-mydata7[,c('Passer','PlayType','GameID','posteam','DefensiveTeam','Yards.Gaine
#results<-crosstab(mydata8, row.vars = "GameID", col.vars = "PlayType", type = "r")
#p1<-results$crosstab
#hist(p1[,1],20)



library(plyr)
count_vector<-count(mydata8, "GameID")

l_new<-length(count_vector$freq)
time<-c()
for(i in 1:l_new)
{time<-append(time,rep(1:count_vector$freq[i]))}
nrow(time)
mydata8$time<-time
mydata8$play_new<-ifelse(mydata8$PlayType=='Pass',1,0)

n_counting<-0  # help counting the number of pairs

## The following code collects all the rows of each pair. However, it is difficult to a
# in such a format.

#empty_df = mydata8[FALSE,]
#for (i in 1:l_new) # level of different game
#{
#   for(j in 1:((count_vector$freq[i])-1)) # within the same game
#   {
#      if(i==1)
#      {row_id<-j}
#      else {row_id<-sum(count_vector$freq[1:(i-1)])+j}
#
#      #print(row_id)
#      if(as.character(mydata8[row_id,]$posteam)!=as.character(mydata8[row_id+1,]$post
#      {
```

```
#         print("not same team")
#         if (nrow(empty_df)==0)
#            {empty_df<-mydata8[row_id:(row_id+1),]}
#         else
#            {
#              if(row.names(mydata8[row_id,])!=row.names(tail(empty_df,1)))
#                {empty_df<-rbind(empty_df,mydata8[row_id,])}
#              empty_df<-rbind(empty_df,mydata8[row_id+1,])
#            }
#        n_counting<-n_counting+1
#       }
#   }
#}


# The following code only collects the second row of the pair, but adds data of
### PT_L: type of play in the last first down from the other team
### TG_L: Yards.Gained in the last play
### FirstDown: did they get first down or not. Note that, if yes, it means it was a fumble.

PT_L="Pass"
TG_L=0
FD_L=0

pari_data= mydata8[1,]
pari_data<-cbind(pari_data,PT_L,TG_L,FD_L)
pari_data<-pari_data[FALSE,]

for (i in 1:l_new) # level of different game
{
  for(j in 1:((count_vector$freq[i])-1)) # within the same game
  {

    if(i==1)
    {row_id<-j}
    else {row_id<-sum(count_vector$freq[1:(i-1)])+j}

    print(row_id)
    if(as.character(mydata8[row_id,]$posteam)!=as.character(mydata8[row_id+1,]$posteam))
    {
      print("not same team")
      PT_L<-as.character(mydata8[row_id,]$PlayType)
      TG_L<-mydata8[row_id,]$Yards.Gained
      FD_L<-mydata8[row_id,]$FirstDown
```

```
      new_row<-cbind(mydata8[(row_id+1),],PT_L,TG_L,FD_L)
      pari_data<-rbind(pari_data,new_row)
     }

      n_counting<-n_counting+1
  }
}

pari_data$same<-ifelse(pari_data$PlayType==pari_data$PT_L,1,0)

#write.csv(pari_data,'pari_data.csv')

write.table(pari_data, file = "pari_data.csv",row.names=FALSE,na = "", sep=",")
```

### Remarks

1. mylogit1: in general, a team has a different play in their first down, compared to the other team in the last first down.
2. mylogit2: If the defence team passed in the last first down, the offence team is less likely to use pass. If the defence team gained more yards, the offence team is more likely to pass in the next first down. If the defence team fumbled, it will reduce the chance the offence team to do the pass.

```
pari_data2<-read.csv('pari_data.csv')

mylogit1 = glm(same~1, family=binomial, data=pari_data2)
summary(mylogit1)
```

```
##
## Call:
## glm(formula = same ~ 1, family = binomial, data = pari_data2)
##
## Deviance Residuals:
##    Min      1Q  Median      3Q     Max
## -1.117  -1.117  -1.117   1.239   1.239
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept) -0.14395    0.02809  -5.124    3e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
```

```
##
##     Null deviance: 7035.5  on 5093  degrees of freedom
## Residual deviance: 7035.5  on 5093  degrees of freedom
## AIC: 7037.5
##
## Number of Fisher Scoring iterations: 3
```

```
mylogit2 = glm(play_new~same+TG_L+FD_L, family=binomial, data=pari_data2)
summary(mylogit2)
```

```
##
## Call:
## glm(formula = play_new ~ same + TG_L + FD_L, family = binomial,
##     data = pari_data2)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.6114  -0.9783  -0.9382   1.0995   1.5672
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept)  0.175629   0.040712   4.314  1.6e-05 ***
## same        -0.757822   0.057618 -13.152  < 2e-16 ***
## TG_L         0.010439   0.003873   2.695  0.00704 **
## FD_L        -0.268115   0.148835  -1.801  0.07164 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 7034.3  on 5093  degrees of freedom
## Residual deviance: 6850.1  on 5090  degrees of freedom
## AIC: 6858.1
##
## Number of Fisher Scoring iterations: 4
```

```
library(lme4)
mylogit3 = glmer(same~play_new+TG_L+FD_L+(1|GameID), family= binomial("logit"), data=pari_data2)
```

```
## boundary (singular) fit: see ?isSingular
```

```
summary(mylogit3)
```

```
## Generalized linear mixed model fit by maximum likelihood (Laplace
```

```
##   Approximation) [glmerMod]
##  Family: binomial  ( logit )
## Formula: same ~ play_new + TG_L + FD_L + (1 | GameID)
##    Data: pari_data2
##
##      AIC      BIC   logLik deviance df.resid
##   6862.4   6895.1  -3426.2   6852.4     5089
##
## Scaled residuals:
##     Min      1Q  Median      3Q     Max
## -1.3918 -0.7763 -0.7532  0.9061  1.6255
##
## Random effects:
##  Groups Name        Variance  Std.Dev.
##  GameID (Intercept) 1.562e-15 3.953e-08
## Number of obs: 5094, groups:  GameID, 256
##
## Fixed effects:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept)  0.197140   0.040513   4.866 1.14e-06 ***
## play_new    -0.757838   0.057619 -13.153  < 2e-16 ***
## TG_L         0.006027   0.003824   1.576  0.11502
## FD_L        -0.392792   0.150715  -2.606  0.00916 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Correlation of Fixed Effects:
##          (Intr) ply_nw TG_L
## play_new -0.627
## TG_L     -0.270 -0.043
## FD_L     -0.147  0.031 -0.041
## convergence code: 0
## boundary (singular) fit: see ?isSingular
```

```
#Bill_1<- bild(play_new ~ TG_L+FD_L, data = mydata8, id="GameID",start = NULL, depende
#summary(Bill_1)

#locust2 <- bild(as.factor(PlayType) ~ time + I(time^2), data = mydata8,id="GameID",st
```

## 9.2 References

https://arxiv.org/pdf/1403.7993.pdf

http://www.dartmouth.edu/~chance/teaching_aids/books_articles/probability_book/Chapter11.pdf

https://rpubs.com/JanpuHou/326048

# Chapter 10

# Project Draft

```
mydata3<-read.csv('Schnibbe 1502 Binary Data.csv')
head(mydata3)
```

```
##   X0
## 1  0
## 2  1
## 3  0
## 4  0
## 5  1
## 6  0
```

```
NO_new<-rep(1:222)
mydata4<-cbind(mydata3,NO_new)
head(mydata4)
```

```
##   X0 NO_new
## 1  0      1
## 2  1      2
## 3  0      3
## 4  0      4
## 5  1      5
## 6  0      6
```

```
a1 = glmer(X0 ~ 1 + (1|NO_new), data = mydata4,family=binomial)
summary(a1)
```

```
## Generalized linear mixed model fit by maximum likelihood (Laplace
##   Approximation) [glmerMod]
##  Family: binomial  ( logit )
## Formula: X0 ~ 1 + (1 | NO_new)
##    Data: mydata4
##
##      AIC      BIC   logLik deviance df.resid
##    243.3    250.1   -119.6    239.3      220
##
## Scaled residuals:
##     Min      1Q  Median      3Q     Max
## -0.5461 -0.5461 -0.5461 -0.5461  1.8311
##
## Random effects:
##  Groups Name        Variance  Std.Dev.
##  NO_new (Intercept) 1.246e-07 0.000353
## Number of obs: 222, groups:  NO_new, 222
##
## Fixed effects:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)  -1.2098     0.1603  -7.549 4.38e-14 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
a2 = glm(X0 ~ 1, data = mydata4,family=binomial)
summary(a2)
```

```
##
## Call:
## glm(formula = X0 ~ 1, family = binomial, data = mydata4)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -0.7225  -0.7225  -0.7225  -0.7225   1.7151
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)  -1.2098     0.1595  -7.583 3.38e-14 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 239.29  on 221  degrees of freedom
## Residual deviance: 239.29  on 221  degrees of freedom
```

```
## AIC: 241.29
##
## Number of Fisher Scoring iterations: 4
```

## 10.1 Background

The following code is from this website: http://www.biostat.umn.edu/~baolin/teaching/probmods/GLMM_mcmc.html. I will remove it on this page after I complete my practice and learning.

In this example, it simulates a longitudinal data with 4 variables for each of 1000 separate individuals. Specifically, there are three continuous covariates (varying over time) and one ordinal covariate (constant over time). We will consider a random intercept model (mean zero and variance 100), and fit the data with glmer() from lme4 R package.

```
n = 1000; p = 3; K = 4; sig = 10
set.seed(123)

## time varying covariates
Xl = vector('list', K)
# 4 list, each 1000 individuals
for(i in 1:K) Xl[[i]] = matrix(rnorm(n*p), n,p)

## constant covariate
Z = rbinom(n, 2,0.2)

## random effects
#just 1000 random numubers?
U = rnorm(n)*sig

## fixed effects
# It ends a 1000*4 matrix
etaX = sapply(Xl, rowSums)

## random errors
eps = matrix(rnorm(n*K), n,K)

## logit model
eta = etaX + U + eps
# calculate probability
prb = 1/(1+exp(-eta))
D = 1*(matrix(runif(n*K),n,K)<prb) # comparing it to prb, and change to 1 and 0; 1000*4
# Select the first list from "Xl", and then add other 3 lists--> 4000 * 3
Xs = Xl[[1]]
```

```
for(k in 2:K) Xs = rbind(Xs, Xl[[k]])

## GLMM model
library(lme4)
sid = rep(1:n, K) # a vector of 1-1000, 4 repetitions
## model fit with GLMMM (default to Laplace approximation)
# subjects as the random effect
a1 = glmer(c(D) ~ Xs + Z[sid] + (1|sid), family=binomial)

a1
```

```
## Generalized linear mixed model fit by maximum likelihood (Laplace
##   Approximation) [glmerMod]
##  Family: binomial  ( logit )
## Formula: c(D) ~ Xs + Z[sid] + (1 | sid)
##       AIC       BIC    logLik  deviance  df.resid
##  3213.666  3251.430 -1600.833  3201.666      3994
## Random effects:
##  Groups Name        Std.Dev.
##  sid    (Intercept) 5.816
## Number of obs: 4000, groups:  sid, 1000
## Fixed Effects:
## (Intercept)          Xs1          Xs2          Xs3       Z[sid]
##      0.1537       0.6650       0.6429       0.6074       0.0199
```

```
## MH sampling of random effects | data
## logit\Pr(D_i|eta_i,U) = eta_i+U; U \sim N(0,Vu)
## proposal dist: N(Uc,Vc)

U.mh <- function(Di,eta, Vu, Uc,Vc, B=100){
  ub = rep(0, B)
  ub[1] = rnorm(1)*sqrt(Vc)+Uc # random starting value
  prb = 1/(1+exp(-eta-ub[1]))
  llk0 = dnorm(ub[1],sd=sqrt(Vu), log=TRUE) + sum(log(Di*prb+(1-Di)*(1-prb))) - dnorm(u
  for(k in 2:B){
    ub[k] = ub[k-1]
    uk = rnorm(1)*sqrt(Vc)+Uc
    prb = 1/(1+exp(-eta-uk))
    llk1 = dnorm(uk,sd=sqrt(Vu), log=TRUE) + sum(log(Di*prb+(1-Di)*(1-prb))) - dnorm(uk
    alpha = exp( llk1 - llk0  )
    if(alpha>=1){
      ub[k] = uk
      llk0 = llk1
    } else{
      aa = runif(1)
```

```
      if(aa<alpha){
        ub[k] = uk
        llk0 = llk1
      }
    }
  }
  return(ub)
}

library(numDeriv)
UV.est <- function(Di,eta,Vu,Uc){
  llk0 = function(xpar){
    Uc = xpar
    prb = 1/(1+exp(-eta-Uc))
    res = dnorm(Uc,sd=sqrt(Vu), log=TRUE) + sum(log(Di*prb+(1-Di)*(1-prb)))
    -res
  }
  tmp = try(optim(Uc, llk0, method='Brent', lower=Uc-10,upper=Uc+10) )
  if(class(tmp)=='try-error') tmp = optim(Uc, llk0)
  Uc = tmp$par
  Vc = 1/hessian(llk0, Uc)
  c(Uc,Vc)
}
UV.mh <- function(Vu,beta,Uc, D,X,subj){
  ## Cov matrix
  sid = unique(subj);  n = length(sid)
  Uc = Vc = rep(0,n)
  for(i in 1:n){
    ij = which(subj==sid[i]);  ni = length(ij)
    Xi = X[ij,,drop=FALSE]
    eta = Xi%*%beta
    zi = UV.est(D[ij],eta,Vu,Uc[i])
    Uc[i] = zi[1]; Vc[i] = zi[2]
  }
  return(list(Uc=Uc,Vc=Vc) )
}

#Newton Raphson update
# Compute first/second derives of complete data log likelihood
## score and fisher information
SF.mh <- function(Vu,beta,Uc,Vc, D,X,subj){
  ## S/hessian matrix
  sid = unique(subj);  n = length(sid)
  p = dim(X)[2]
  S = rep(0, p)
```

```
  FI = matrix(0, p,p)
  sig2 = 0
  for(i in 1:n)
    {
    ij = which(subj==sid[i]);  ni = length(ij)
    Xi = X[ij,,drop=FALSE]
    eta = Xi%*%beta
    zi = U.mh(D[ij],eta,Vu,Uc[i],Vc[i], B=5e3)[-(1:1e3)]
    theta = sapply(eta, function(b0)  mean(1/(1+exp(-b0-zi))) )
    theta2 = sapply(eta, function(b0) mean(exp(b0+zi)/(1+exp(b0+zi))^2) )
    FI = FI + t(Xi)%*%(theta2*Xi)
    S = S+colSums((D[ij]-theta)*Xi)
    sig2 = sig2 + mean(zi^2)
    }
  return(list(S=S, FI=FI, sig2=sig2/n) )
}

library(lme4)
sid = rep(1:n, K)
a1 = glmer(c(D) ~ Xs + Z[sid] + (1|sid), family=binomial)
## extract variance and fixed effects parameters; + mode/variance of (random effects|da
Vu = (getME(a1,'theta'))^2; beta = fixef(a1); Um = ranef(a1,condVar=TRUE)
D = c(D); X = cbind(1,Xs,Z[sid]); subj = sid
Uc = unlist(Um[[1]]); Vc = c( attr(Um[[1]], 'postVar') )
for(b in 1:100){
  ## NR updates with MH sampling
  obj = SF.mh(Vu,beta,Uc,Vc, D,X,subj)
  Vu = obj$sig2
  tmp = solve(obj$FI,obj$S)
  beta = beta + tmp
  ## Proposal dist update
  tmp1 = UV.mh(Vu,beta,Uc, D,X,subj)
  Uc = tmp1$Uc; Vc = tmp1$Vc
  cat(b, ':', tmp, ';', obj$S/n, '\n\t', sqrt(Vu), beta, '\n')
}
```

## 10.2 Important Examples with R code

1. Fitting mixed models with (temporal) correlations in R

https://bbolker.github.io/mixedmodels-misc/notes/corr_braindump.html

2. Mixed effects logistic regression

https://stats.idre.ucla.edu/r/dae/mixed-effects-logistic-regression/

## 10.3 References

1. Data

http://www.michelecoscia.com/?page_id=379