GLMM, Concepts, & R

Bill Last Updated:

17 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

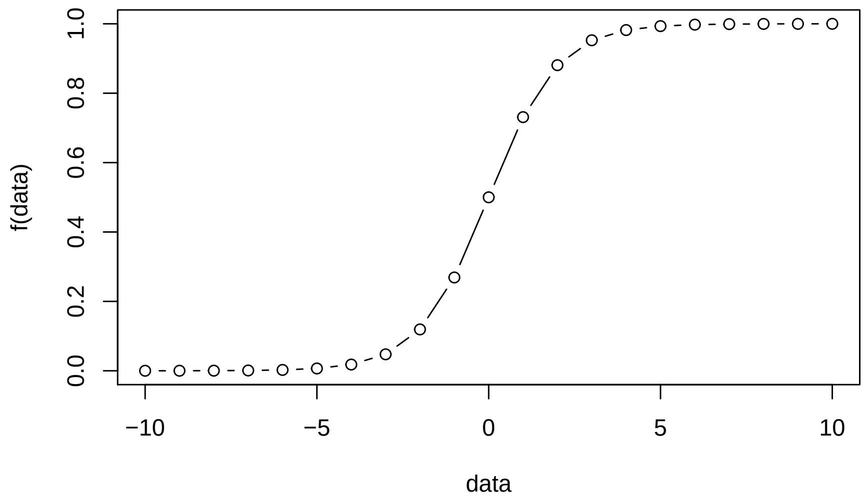
$$f(x) = log(\frac{p(y=1)}{1-p(y=1)})$$

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-Bow
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)
```

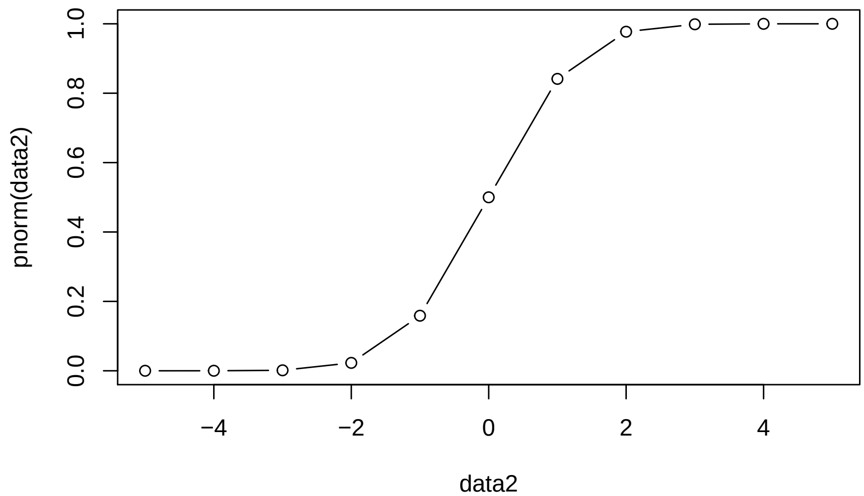
```
##
## Call:
```

```
## 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

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 ρ, then we can get:

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

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 ρ.

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

Set the derivative to zero and solve for ρ, 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 ρ 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 ρ. If the result is consistent with what we did above, we should observe that the *cdf* of the esimate of β_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

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

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

where,

$\mu(X) : randomcomponent$

$X^T\beta : covariates$

3.1.2 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

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

3.1.3 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.)

3.1.4 GLM-Reciprocal link:

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

Reciprocal link:

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

3.1.5 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 $T_1(x) = x$

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

$h(x) = 1$

3.1.6 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.),

$\theta = \mu$

$\phi = \sigma^2$

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

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

3.1.7 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,

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

3.1.8 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:

$$\begin{aligned} E(X) &= \frac{\partial(e^\theta)}{\partial\theta} = e^\theta = \mu \\ Var(X) &= \frac{\partial^{''}(e^\theta)}{\partial\theta^2}\phi = \frac{\partial^{''}(e^\theta)}{\partial\theta^2} = \mu \end{aligned}$$

3.1.9 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,

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

3.1.10 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,

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

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,

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

3.1.11 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,

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

3.1.12 Iteratively Re-weighted Least Squares

3.2 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.)

3.3 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
```

3.4 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
```

3.5 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
```

3.6 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
```

```r
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
```

```r
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
```

3.7 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
```

```r
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 θ.

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

Typically, they use ∇ 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 β 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): $\mathbb{R}^n \to \mathbb{R}^{\gtrdot}$

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

For instance,

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

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

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

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

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.

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

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

As a special case, in the context of logit:

Suppose that the log likelihood function is $\ell(\theta)$. θ 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 μ.

Remarks:

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

(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 μ 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.

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

...

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

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.

```
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 3×1, you do not need to transpose the β.

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.

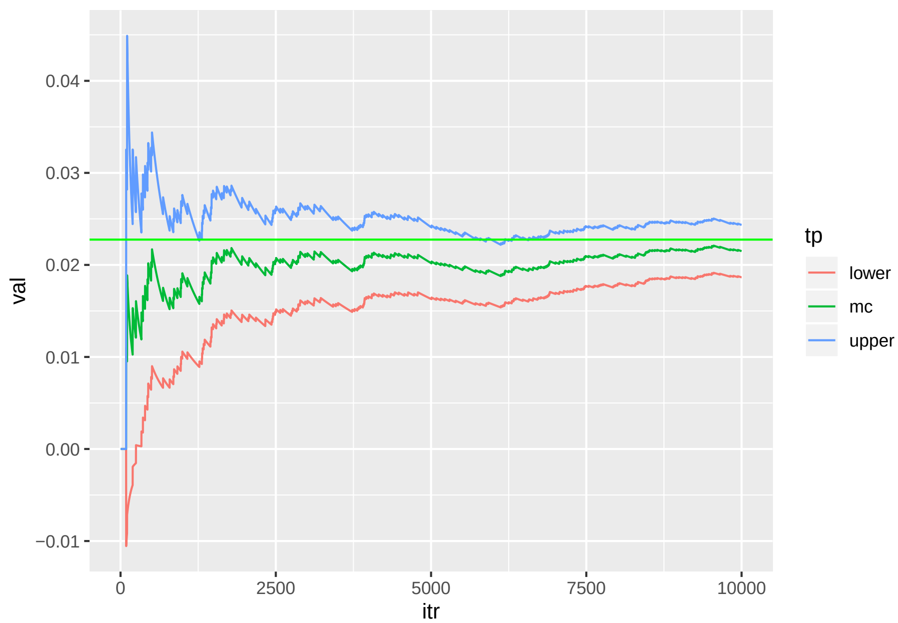
6.1 Monte carlo approximation

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

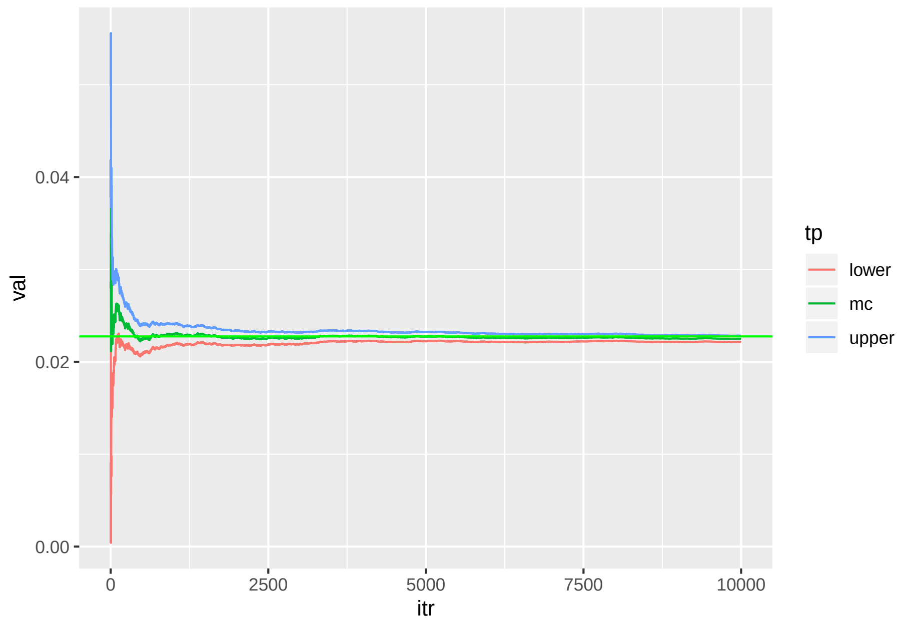
Using the same explane above, we can use a shifted exponential distribution to help calculate the intergral for normal distribution. Specifically,

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

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 derivative 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:

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

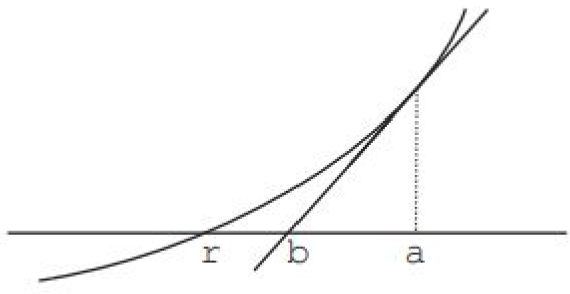


Figure 6.1: Credit of this figure: https://www.math.ubc.ca/~anstee/math104/newtonmethod.pdf

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.

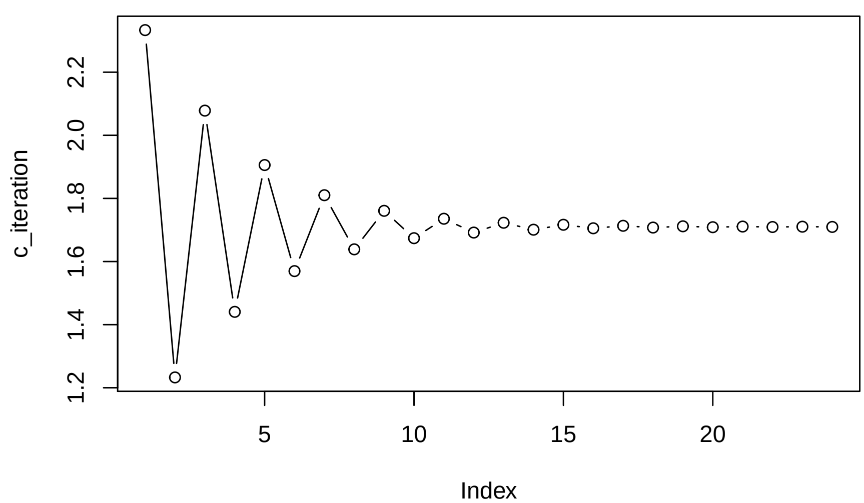
$$\beta_{t+1} = \beta_t - \frac{\ell'(\beta_t)}{\ell''(\beta_t)}$$

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
```

```r
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
```

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

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

```
# 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:

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

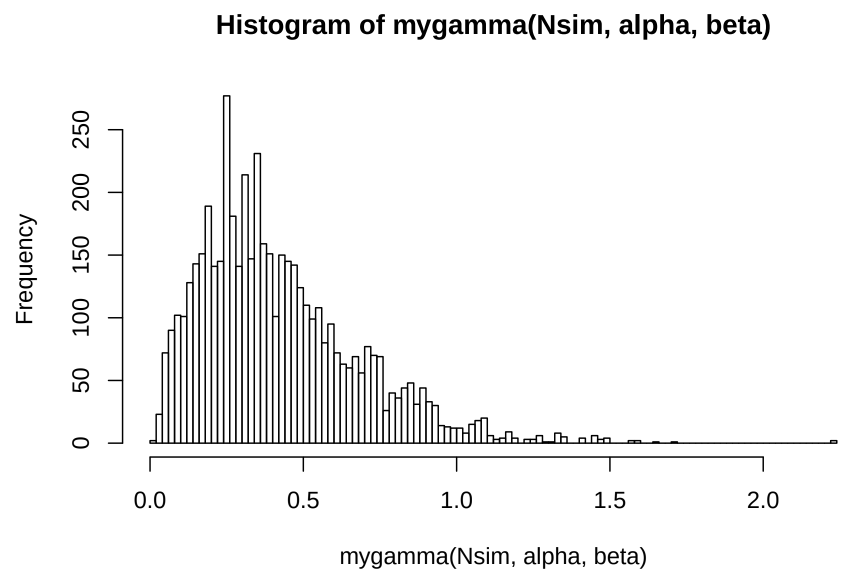
(4) $\rho = min(p, 1)$

(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 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.)

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 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"

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 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.

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

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

```
#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
```

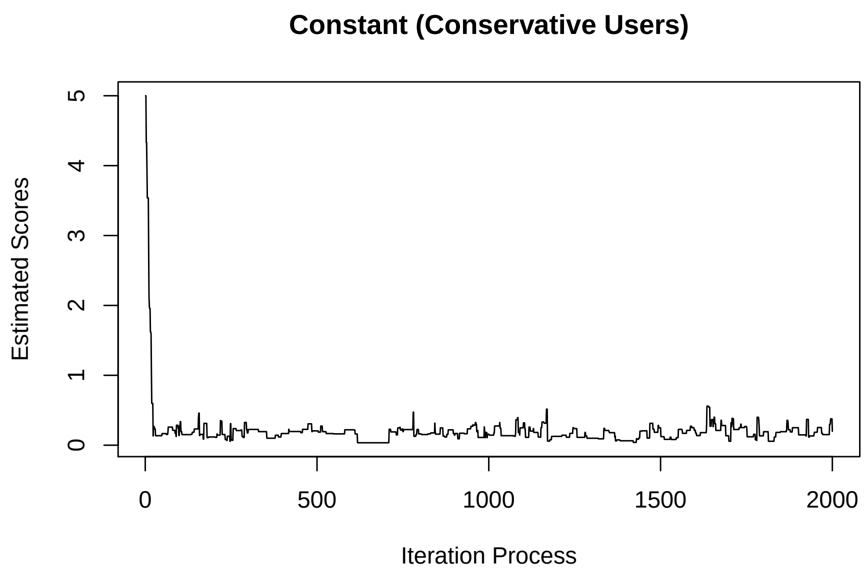
8.3 Simulating Data of Conservative Users on Twitter and Model Testing

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.

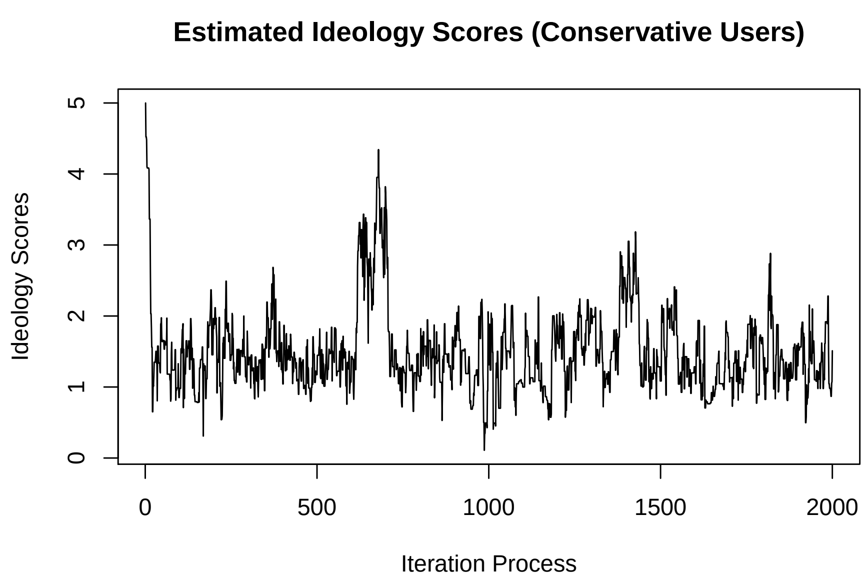
```
#This user approximately follows 45 Republican Senators and 10 Democrat Senators.
Data_user<-as.data.frame(matrix(c(ifelse(runif(50)<.1,0,1),ifelse(runif(50)<.8,0,1))),
colnames(Data_user)<-c("R_User")
Data_combined<-cbind(Data_Elites,Data_user)

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
```

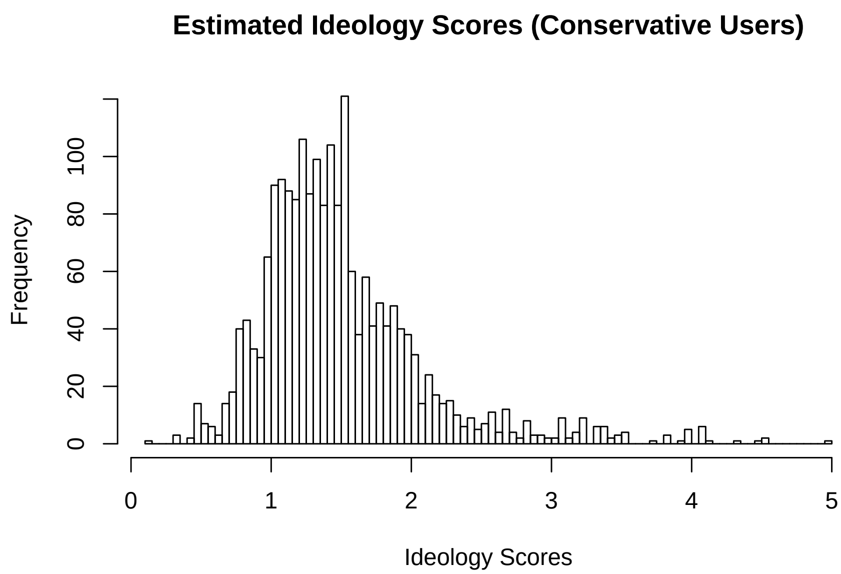
```
plot(fit_C$theta[,1],main="Constant (Conservative Users)",
     xlab="Iteration Process",ylab="Estimated Scores",type="l")
```




```
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

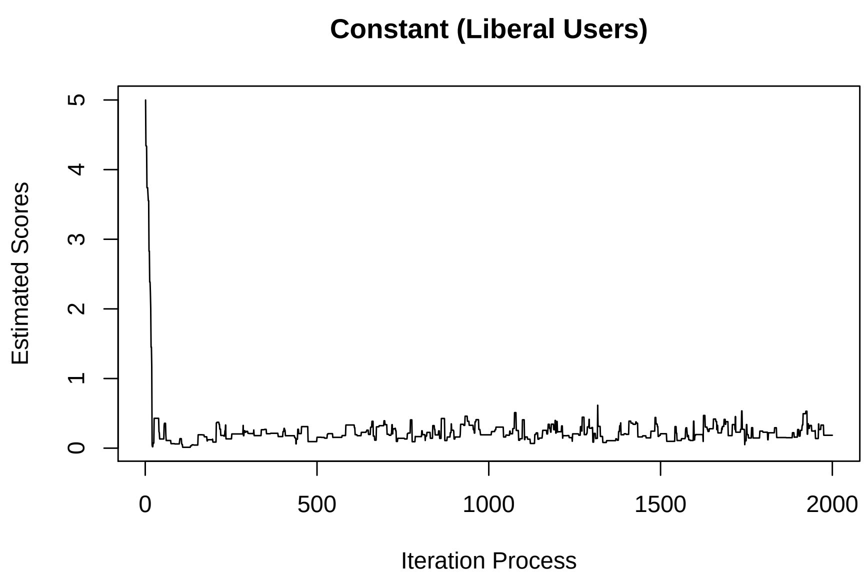
To further verify the Metropolis Hastings algorithm, I plan to test the opposite estimate. Specifically, assume that we observe another user, who is more liberal. To simulate Twitter following data for this user, I assign this user to follow more Democrat senators. In this case, we would expect to see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

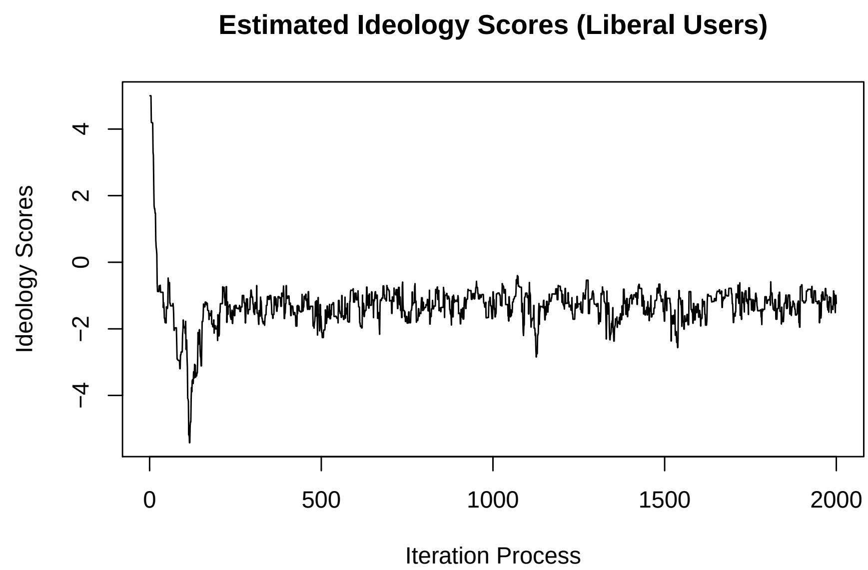
```
fit_L<-Bayes_logit(Y_data,X_data)
fit_L$acceptance_rate
```

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

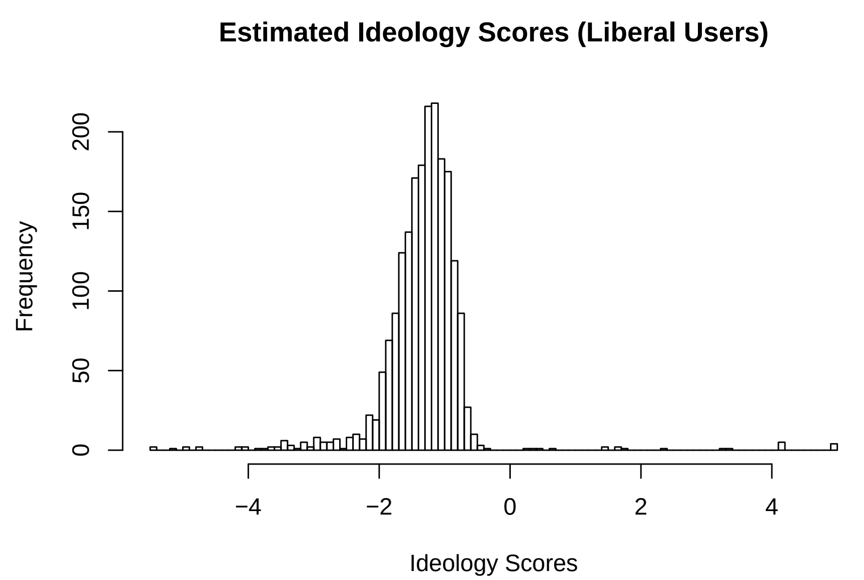
```
plot(fit_L$theta[,1],main="Constant (Liberal Users)",
     xlab="Iteration Process",ylab="Estimated Scores",type="l")
```




```
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.Gain
#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,]$posteam)!=as.character(mydata8[row_id+1,]$poste
#      {
```

```
#        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
```

```r
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.

The R code:

```
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
```

```r
# 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
```

```r
## 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

Chapter 11

Bayesian - 1

The following is the part of the class note that I took from the online course of "Bayesian Statistics: From Concept to Data Analysis." (https://www.coursera.org/learn/bayesian-statistics/home/welcome)

Important note: All the notes here are just for my own study purpose. I do not clain any copyright. You can use it for study purpose as well, but not for any business purposes.

11.1 Frequentist perspective

$$\theta = \{fair, loaded\}$$

$$x \sim Bin(5, \theta)$$

$$\begin{aligned} f(x|\theta) &= \begin{cases} \binom{5}{x}(\frac{1}{2})^5 & if\ \theta = fair \\ \binom{5}{x}(0.7)^x(0.3)^{5-x} & if\ \theta = loaded \end{cases} \\ &= \binom{5}{x}(\frac{1}{2})^5 I_{\{\theta=fair\}} + \binom{5}{x}(0.7)^x(0.3)^{5-x} I_{\{\theta=loaded\}} \end{aligned}$$

When $x = 2$

$$f(\theta|x=2) = \begin{cases} \binom{5}{x}(\frac{1}{2})^5 = 0.3125 & if\ \theta = fair \\ \binom{5}{x}(0.7)^x(0.3)^{5-x} = 0.1323 & if\ \theta = loaded \end{cases}$$

Thus, based on MLE, it suggests that it should be "fair", since it has a greater probablity if we observe 2 head out of 5 trials.

However, we can not know the following probability: given that we observe $x = 2$, what is the probability that θ is fair?

$$P(\theta = fair|X = 2)$$

From the frequentist's perspective, the coin is the fixed coin. And thus, the probablity of $P(\theta = fair|x = 2)$ is equal to $P(\theta = fair)$.

$$P(\theta = fair|x = 2) = P(\theta = fair)$$

As,

$$P(\theta = fair) \in C(0, 1)(i.e., either\ 0\ or\ 1)$$

11.2 Bayesian perspective

Prior $P(loaded) = 0.6$

$$\begin{aligned} f(\theta|X) &= \frac{f(x|\theta)f(\theta)}{\sum_\theta f(x|\theta)f(\theta)} \\ &= \frac{\binom{5}{x}[(\frac{1}{2})^5 \times 0.4 \times I_{\{\theta=fair\}} + (0.7)^x(0.3)^{5-x} \times 0.6 \times I_{\{\theta=loaded\}}]}{\binom{5}{x}[(\frac{1}{2})^5 \times 0.4 + (0.7)^x(0.3)^{5-x} \times 0.6]} \end{aligned}$$

$$\begin{aligned} f(\theta|X = 2) &= \frac{0.0125 I_{\{\theta=fair\}} + 0.0079 I_{\{\theta=loaded\}}}{0.0125 + 0.0079} \\ &= 0.612 I_{\{\theta=fair\}} + 0.388 I_{\{\theta=loaded\}} \end{aligned}$$

Thus, we can say that:

$$P(\theta = loaded|X = 2) = 0.388$$

We can change the prior, and get different posterior probabilities:

$$P(\theta = loaded) = \frac{1}{2} \rightarrow P(\theta = loaded|X = 2) = 0.297$$

$$P(\theta = loaded) = \frac{9}{10} \rightarrow P(\theta = loaded|X = 2) = 0.792$$

11.3 Continous parameters

In the examples above, θ is discrete. In contrast, the examples below use continous θ.

$$f(\theta|y) = \frac{f(y|\theta)f(\theta)}{f(y)} = \frac{f(y|\theta)f(\theta)}{\int f(y|\theta)f(\theta)d\theta} = \frac{likelihood \times prior}{normalizing - constant} \propto likelihood \times prior$$

Note that, the posterior is a PDF of θ, which is not in the function of $f(y)$. Thus, removing the denominator (i.e., the normalizing constant) does not change the form of the posterior.

11.3.1 Uniform

Suppose that θ is the probablity of a coin getting head. We could assign a uniform distribution.

$$\theta \sim U[0, 1]$$

$$f(\theta) = I_{\{0 \leqq \theta \leqslant 1\}}$$

(It is interesting to see how to write the pdf for uniform distribution.)

$$f(\theta|Y = 1) = \frac{\theta^1(1-\theta)^0 I_{\{0 \leqq \theta \leqslant 1\}}}{\int_{-\infty}^{+\infty} \theta^1(1-\theta)^0 I_{\{0 \leqq \theta \leqslant 1\}} d\theta} = \frac{\theta I_{\{0 \leqq \theta \leqslant 1\}}}{\int_0^1 \theta d\theta} = 2\theta I_{\{0 \leqq \theta \leqslant 1\}}$$

If we ignore the normalizing constant, we will get

$$f(\theta|Y = 1) \propto \theta^1(1-\theta)^0 I_{\{0 \leqq \theta \leqslant 1\}} = \theta I_{\{0 \leqq \theta \leqslant 1\}}$$

Thus, we can see that with vs. without the noramlizing constant is the "2".

11.3.2 Uniform: prior versus posterior

When θ follows uniform distribution:

Prior

$$P(0.025 < \theta < 0.975) = 0.95$$

$$P(0.05 < \theta) = 0.95$$

Posterior

$$P(0.025 < \theta < 0.975) = \int_{0.025}^{0.975} 2\theta d\theta = 0.95$$

$$P(0.05 < \theta) = 1 - P(\theta < 0.05) = \int_{0}^{0.05} 2\theta d\theta = 1 - 0.05^2 = 0.9975$$

Thus, we can see that, while $P(0.025 < \theta < 0.975)$ is the same for prior and posterior, $P(0.05 < \theta)$ is not the same.

11.3.3 Uniform: equal tailed versus HPD

Equal tailed

$$P(\theta < q|Y = 1) = \int_{0}^{q} 2\theta d\theta = q^2$$

$$P(\sqrt{0.025} < \theta < \sqrt{0.975}|Y = 1) = P(0.158 < \theta < 0.987) = 0.95$$

We can say that: there's a 95% probability that θ is in between 0.158 and 0.987.

Highest Posterior Density

$$P(\theta > \sqrt{0.05}|Y = 1) = P(\theta > 0.224|Y = 1) = 0.95$$

11.4 Bernoulli/binomial likelihood with uniform prior

$$\begin{aligned}
f(\theta|Y = 1) &= \frac{\theta^{\sum y_i}(1-\theta)^{\sum n - y_i} I_{\{0 \leqq \theta \leqslant 1\}}}{\int_{-\infty}^{+\infty} \theta^{\sum y_i}(1-\theta)^{n - \sum y_i} I_{\{0 \leqq \theta \leqslant 1\}} d\theta} \\
&= \frac{\theta^{\sum y_i}(1-\theta)^{\sum n - y_i} I_{\{0 \leqq \theta \leqslant 1\}}}{\frac{\Gamma(\sum y_i + 1)\Gamma(n - \sum y_i + 1)}{\Gamma(n+2)} \int_{-\infty}^{+\infty} \frac{\Gamma(n+2)}{\Gamma(\sum y_i + 1)\Gamma(n - \sum y_i + 1)} \theta^{\sum y_i}(1-\theta)^{n - \sum y_i} I_{\{0 \leqq \theta \leqslant 1\}} d\theta} \\
&= \frac{\Gamma(n+2)}{\Gamma(\sum y_i + 1)\Gamma(n - \sum y_i + 1)} \theta^{\sum y_i}(1-\theta)^{\sum n - y_i} I_{\{0 \leqq \theta \leqslant 1\}}
\end{aligned}$$

Thus,

$$\theta|y \sim Beta(\sum y_i + 1, n - \sum y_i + 1)$$

Side note: $Beta(1,1) = Uniform(0,1)$:

$Beta(\alpha,\beta) = \frac{x^{\alpha-1}(1-x)^{\beta-1}}{B(\alpha,\beta)} I_{\{0 \leqq x \leqslant 1\}}$

Thus, we can get the following since the support for beta distribution is $[0,1]$:

$Beta(1,1) = \frac{x^0(1-x)^0}{B(\alpha,\beta)} = 1 \times I_{\{0 \leqq x \leqslant 1\}}$

11.5 Conjugate priors

As noted above, beta prior (or, Uniform) leads to beta posterior. In a more general sense, Beta prior always leads to beta posterior.

For instance,

$$\begin{aligned} f(\theta|y) \propto f(y|\theta)f(\theta) &= \theta^{\sum y_i}(1-\theta)^{\sum n-y_i}\frac{\theta^{\alpha-1}(1-\theta)^{\beta-1}}{B(\alpha,\beta)} I_{\{0 \leqq \theta \leqslant 1\}} \\ &= \frac{1}{B(\alpha,\beta)}\theta^{\sum y_i+\alpha-1}(1-\theta)^{\sum n-y_i+\beta-1} I_{\{0 \leqq \theta \leqslant 1\}} \\ &\propto \theta^{\sum y_i+\alpha-1}(1-\theta)^{\sum n-y_i+\beta-1} I_{\{0 \leqq \theta \leqslant 1\}} \end{aligned}$$

Thus,

$$f(\theta|y) \sim Beta(\alpha + \sum y_i, \beta + \sum n - y_i)$$

Conjugate prior: prior and posterior share the same distribution. As we can see, both the data and the prior contribute to the posterior.

For the prior of $Beta(\alpha,\beta)$, the mean is

$$Mean_{prior} = \frac{\alpha}{\alpha+\beta}$$

Posterior mean is,

$$\begin{aligned} Mean_{posterior} &= \frac{\alpha + \sum y_i}{\alpha + \sum y_i + \beta + n - \sum y_i} \\ &= \frac{\alpha + \sum y_i}{\alpha+\beta+n} \\ &= \frac{\alpha+\beta}{\alpha+\beta+n}\frac{\alpha}{\alpha+\beta} + \frac{n}{\alpha+\beta+n}\frac{\sum y_i}{n} \\ &= Weight_{prior} \times Mean_{prior} + Weight_{data} \times Mean_{data} \end{aligned}$$

Side Note

(1) Binomial proportion confidence interval:

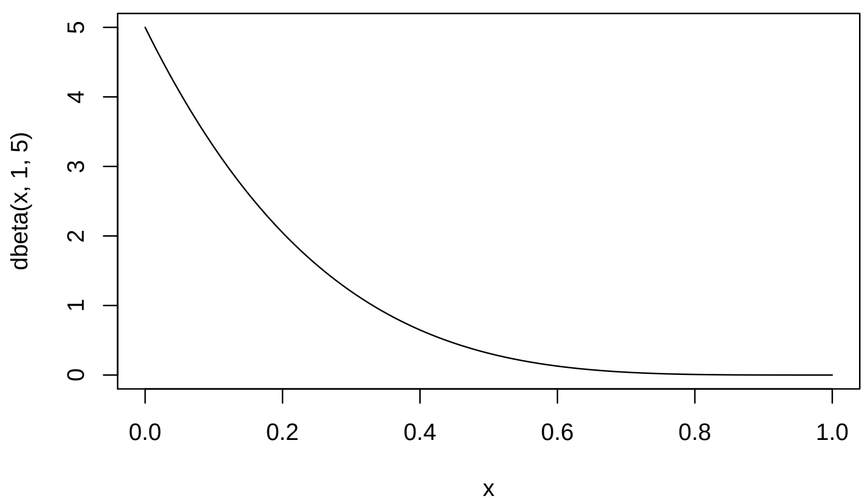
$$\hat{p} + / - 1.96\sqrt{\frac{\hat{p}(1-\hat{p})}{n}}$$

(2) Mean of Beta distribution:

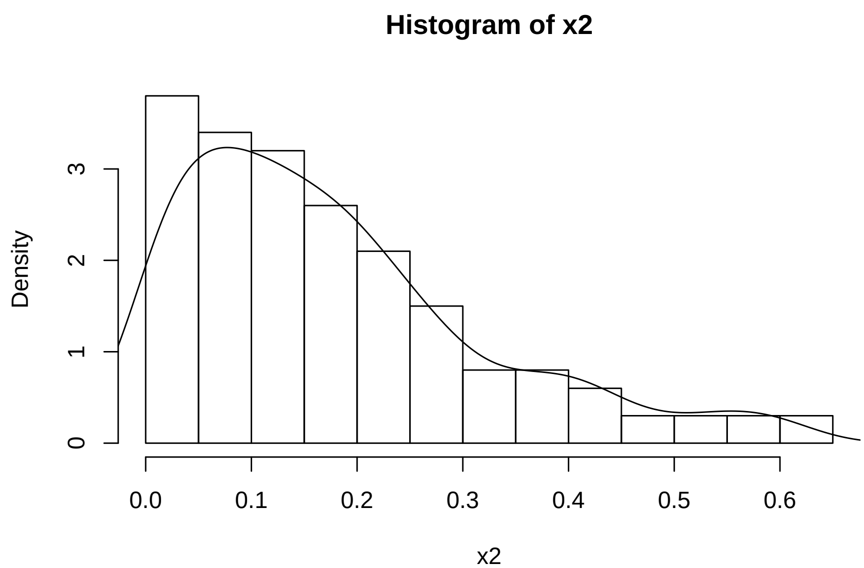
$$\frac{\alpha}{\alpha+\beta}$$

(3) Plot of Beta distribution

```
# Method 1
x<-seq(0,1,length=200)
plot(x,dbeta(x,1,5),type = "l")
```




```
# Method 2
x2<-rbeta(200,1,5)
hist(x2,prob = TRUE)
lines(density(x2))
```



11.6 Poisson distribution

Pmf of Poisson distribution:

$$Pois(\lambda) \sim \frac{\lambda^k e^{-\lambda}}{k!}$$

We can replace k with the notation of y, and assume that we observe n y_i:

$$y_i \sim \frac{\lambda^{y_i} e^{-\lambda}}{y_i!}$$

$$f(y|\lambda) = \frac{\lambda^{\sum y_i} e^{-n\lambda}}{\prod_{i=1}^{n} y_i!}$$

We assume that λ follows Gamma distribution (i.e.,Gamma prior):

$$\lambda \sim \Gamma(\alpha, \beta)$$

The pdf for Gamma distribution is:

$$\frac{\beta^\alpha}{\Gamma(\alpha)} x^{\alpha-1} e^{-\beta x}$$

Thus, the posterior is as follows:

$$\begin{aligned} f(\lambda|y) &\propto f(y|\lambda)f(\lambda) \\ &= \frac{\lambda^{\sum y_i} e^{-n\lambda}}{\prod_{i=1}^{n} y_i!} \times \frac{\beta^\alpha}{\Gamma(\alpha)} \lambda^{\alpha-1} e^{-\beta\lambda} \\ &\propto \lambda^{\sum y_i} e^{-n\lambda} \times \lambda^{\alpha-1} e^{-\beta\lambda} \\ &= \lambda^{(\alpha+\sum y_i)-1} e^{-(\beta+n)\lambda} \end{aligned}$$

Thus, the posterior is:

$$\Gamma(\alpha + \sum y_i, \beta + n)$$

As we know that, the mean of prior for Gamma is $\frac{\alpha}{\beta}$. Thus, we can get the mean for the posterior for Gamma is:

$$\begin{aligned} &= \frac{\alpha + \sum y_i}{\beta + n} \\ &= \frac{\beta}{\beta + n} \frac{\alpha}{\beta} + \frac{n}{\beta + n} \frac{\sum y_i}{n} \end{aligned}$$

To determine the prior of α and β:

(1) Prior mean $\frac{\alpha}{\beta}$

(a) Prior std. dev. $\frac{\sqrt{\alpha}}{\beta}$
(b) Effective sample size β

(2) Vague prior Small $\varepsilon > 0$: $\Gamma(\varepsilon, \varepsilon)$. Thus, the posterior mean is primarily driven by the data:

$$\frac{\varepsilon + \sum y_i}{\varepsilon + n} \approx \frac{\sum y_i}{n}$$

(1) As we know, beta prior lead the Bernoulli trial to a beta posterior. That is, we know $f(\theta|y) = \frac{f(y|\theta)f(\theta)}{f(y)}$. What is the prior predictive distribution of $f(y)$?

(2) If Beta is Beta (3,3), what is the prior predictive probablity that we wil observe $y = 0$ in the next trial?

11.7 Exponential data

For instance, suppose that on average you need to wait for 10 minutes for a fast food delivery, and thus we can assume that $y \sim Exp(\lambda)$. Furthermore, we assume that the prior λ follows Gamma distribution $Gamma(\alpha, \beta)$, thus it is with a mean of $\frac{\alpha}{\beta} = \frac{1}{10}$.

$$if \;\; \Gamma(100, 1000)$$

(Note that, it has a mean of $\frac{100}{1000} = \frac{1}{10}$).

Thus, we can get:

$$\begin{aligned} f(\lambda|y) &\propto f(y|\lambda)f(\lambda) \\ &\propto \lambda e^{-\lambda y}\lambda^{\alpha-1}e^{-\beta\lambda} \\ &\propto \lambda^{(\alpha+1)-1}e^{-(\beta+y)\lambda} \end{aligned}$$

Thus, we get

$$\lambda|y \sim \Gamma(\alpha+1, \beta+y)$$

Thus, if we observe a data point that we need to wait for 12 minutes for a fast food delivery, we can update the posterior:

$$\lambda|y \sim \Gamma(101, 1012)$$

Thus, the mean for the posterior is

$$\frac{101}{1012} = \frac{1}{10.02}$$

Note:

(1) Typically, we know that the pdf of Gamma is $\frac{\beta^\alpha}{\Gamma(\alpha)}x^{\alpha-1}e^{-\beta x}$. We replace x with λ since now the random variable of x is to represent the parameter λ in the exponential distribution.

(2) In the above, we drop the constant part ($\frac{\beta^\alpha}{\Gamma(\alpha)}$) in the Gamma distribution as long as it does not include x (i.e., λ).

(3) Suppose that you have 4 observations in total, then

$$\begin{aligned} f(\lambda|y) &\propto f(y_1|\lambda)f(y_2|\lambda)f(y_3|\lambda)f(y_4|\lambda)f(\lambda) \\ &\propto \lambda e^{-\lambda y_1}\lambda e^{-\lambda y_2}\lambda e^{-\lambda y_3}\lambda e^{-\lambda y_4}\lambda^{\alpha-1}e^{-\beta\lambda} \\ &\propto \lambda^{(\alpha+4)-1}e^{-(\beta+\sum_{i=1}^{4} y_i)\lambda} \end{aligned}$$

Thus, the generalized form is as follows:

$$\lambda^{(\alpha+n)-1}e^{-(\beta+\sum_{i=1}^{n} y_i)\lambda}$$

11.8 Normal likelihood

11.8.1 When variance is known

$$x_i \sim N(\mu, \sigma_0^2)$$

(σ_0 is assumed to be known. Thus, the only unknown parameter is μ.)

The conjugate prior for normal distribution is normal distribution itself.

$$f(\mu|x) \sim f(x|\mu)f(\mu)$$

Assume that

$$\mu \sim N(m_0, s_0^2)$$

$$\mu|x \sim N\left(\frac{\frac{n\bar{x}}{\sigma_o^2} + \frac{m_o}{S_0^2}}{\frac{n}{\sigma_0^2} + \frac{1}{s_0^2}}, \frac{1}{\frac{n}{\sigma_0^2} + \frac{1}{s_0^2}}\right)$$

Thus, where

$$\begin{aligned}
\frac{\frac{n\bar{x}}{\sigma_o^2} + \frac{m_o}{S_0^2}}{\frac{n}{\sigma_0^2} + \frac{1}{s_0^2}} &= \frac{\frac{n\bar{x}}{\sigma_o^2}}{\frac{n}{\sigma_0^2} + \frac{1}{s_0^2}} + \frac{\frac{m_o}{S_0^2}}{\frac{n}{\sigma_0^2} + \frac{1}{s_0^2}} \\
&= \frac{n}{n + \frac{\sigma_0^2}{s_0^2}}\bar{x} + \frac{\frac{\sigma_0^2}{S_0^2}}{n + \frac{\sigma_0^2}{s_0^2}}m_o
\end{aligned}$$

Note:

(1) As we can see, the posterior mean is a weighted mean – a combination of prior mean and sample mean.

(2) When n is larger, the sample mean $\bar{x}$ gets more weight.

(3) For the prior mean m_0, the smaller the prior variance s_0^2 is, the prior mean gets more weight. If the prior variance s_0^2 is big, the prior mean will get less weight in the final posterior mean.

11.8.2 When variance is unknown

$$x_i|\mu, \sigma^2 \sim N(\mu, \sigma^2)$$

$$\mu|\sigma^2 \sim N(m, \frac{\sigma^2}{w})$$

Side note

$w = \frac{\sigma^2}{\sigma_\mu^2}$ effective sample size

$$\sigma^2 \sim \Gamma^{-1}(\alpha, \beta)$$

Thus, we can get that,

$$\sigma^2|x \sim \Gamma^{-1}(\alpha + \frac{n}{2}, \beta + \frac{1}{2}\sum_{i=1}^{n}(x_i - \bar{x})^2 + \frac{nw}{2(n+w)}(\bar{x} - m)^2)$$

$$\mu|\sigma^2, x \sim N(\frac{n\bar{x} + wm}{n+w}, \frac{\sigma^2}{n+w})$$

Where,

$\frac{n\bar{x}+wm}{n+w} = \frac{w}{n+w}m + \frac{n}{n+w}\bar{x}$

$$\mu|x \sim t - distribution$$

11.9 Non-informative priors

11.9.1 Bernoulli

$Y_i \sim B(\theta)$

$$\theta \sim U[0, 1] = Beta(1, 1)$$

(Effective sample size is 1+1=2)

If we get $Beta(\frac{1}{2}, \frac{1}{2})$ and $Beta(0.001, 0.001)$ have less impact on the posterior.

Improper prior, for instance, THe prior

$$Beta(0, 0)$$

$$f(\theta) \propto \theta^{-1}(1-\theta)^{-1}$$

In this case,

$$f(\theta|y) \propto \theta^{y-1}(1-\theta)^{n-y-1} \sim Beta(y, n-y)$$

Posterior mean: $\frac{y}{n} = \hat{\theta}$

11.9.2 Gaussian

$$Y_i \sim N(\mu, \sigma^2)$$

Vague prior:

$$\mu \sim N(0, 1000000^2)$$

or,

$$f(\mu) \sim 1$$

$$\begin{aligned} f(\mu|y) &\propto f(y|\mu)f(\mu) \\ &\propto exp(-\frac{1}{2\sigma^2}\sum(y_i-\mu)^2) \times 1 \\ &\propto exp(-\frac{1}{2\frac{\sigma^2}{n}}\sum(y_i-\bar{y})^2) \end{aligned}$$

Thus,

$$\mu|y \sim N(\bar{y}, \frac{\sigma^2}{n})$$

This is exactly the same as the esitimate from MLE estimate.

NOTE

In case that the variance is unknown,

$$f(\sigma^2) \propto \frac{1}{\sigma^2}$$

This is equivalent to the following:

$$\Gamma^{-1}(0, 1)$$

Thus the posteiro for σ^2

$$\sigma^2|y \sim \Gamma^{-1}(\frac{n-1}{2}, \frac{1}{2}\sum(y_i-\bar{y})^2)$$

11.10 Jeffreys Priors

Jeffreys Prior

$$f(\theta) \propto \sqrt{I(\theta)}$$

For instance,

11.10.1 Gaussian

$$Y_i \sim (\mu, \sigma^2) \rightarrow f(\mu) \propto 1, f(\sigma^2) \propto \frac{1}{\sigma^2}$$

11.10.2 Bernoulli

$$Y_i \sim B(\theta) \rightarrow f(\theta) \propto \theta^{-\frac{1}{2}}(1-\theta)^{-\frac{1}{2}} \sim Beta(\frac{1}{2}, \frac{1}{2})$$

11.10.3 Side Note: Fisher Information

The Fisher information (for one paramter):

$$?(\theta) = E[(\frac{d}{d\theta} log(f(X|\theta)))^2]$$

The Expectation is with respect to X with a PDF $f(x|\theta)$.

For instance, suppose the $X|\theta \sim N(\theta, 1)$

$$\begin{aligned}
f(x|\theta) &= \frac{1}{\sqrt{2\pi}} exp[-\frac{1}{2}(x-\theta)^2] \\
log(f(x|\theta)) &= log(\frac{1}{\sqrt{2\pi}}) - \frac{1}{2}(x-\theta)^2 \\
\frac{d}{d\theta} log(f(x|\theta)) &= x - \theta \\
(\frac{d}{d\theta} log(f(x|\theta)))^2 &= (x-\theta)^2 \\
E[(\frac{d}{d\theta} log(f(x|\theta)))^2] &= (x-\theta)^2 = Var(X) = 1
\end{aligned}$$

11.10.4 Prior predictive distribution

Prior distribution vs. prior predictive distribution

https://stats.stackexchange.com/questions/394648/differences-between-prior-distribution-and-prior-predictive-distribution

Stochastic Processes

https://www.youtube.com/watch?v=TuTmC8aOQJE

11.11 Linear regression

11.11.1 Review

For single covariate of x:

$$E[y] = \beta_0 + \beta_1 x$$

Thus,

$$Y \sim N(\beta_0 + \beta_1 x, \sigma^2)$$

For multiple covariates,

$$E[y] = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + ... + \beta_k x_k$$

Side Note It is interesting to think y as the expected value of of the combination of βx.

11.11.2 When σ^2 is known

If we use a Jeffreys prior and assume σ^2 is known, we will get β that has the same mean as the standard OLS. Assuming we only has one covariate:

$$\beta_1|y \sim N(\frac{\sum_{i=1}^n (x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=1}^n (x_i - \bar{x})^2}, \frac{\sigma^2}{\sum_{i=1}^n (x_i - \bar{x})^2})$$

For mutiple covariates, we can use matrix notation,

$$\beta|y \sim N((X^t X)^{-1} X^t y, (X^t X)^{-1} \sigma^2)$$

11.11.3 When σ^2 is unknown

When both β and σ^2 are unknown, the standard prior is the non-informative Jeffreys prior:

$$f(\beta, \sigma^2) \propto \frac{1}{\sigma^2}$$

The posterior mean for β is the same as standard OLS estimates. The posterior for β conditional on σ^2 is the same normal distribution when σ^2 is known. However, the marginal posteior distribution for β, with σ^2 integrated out, is a t distribution.

The t distribution has the mean of $(X^tX)^{-1}X^ty$ and variance matrix, $s^2(X^tX)^{-1}$, where $s^2 = \sum_{i=1}^{n}(y_i - \hat{y}_i)^2/(n-k-1)$

The variance of σ^2 is an inverse gamma distribution

$$\sigma^2|y \sim \Gamma^{-1}(\frac{n-k-1}{2}, \frac{n-k-1}{2}s^2)$$

Chapter 12

Bayesian - 2

12.1 Components of Bayesian models

$$y_i = \mu + \epsilon_i$$

Where,

$$\epsilon_i \sim N(0, \sigma^2)$$

$$y_i \sim N(\mu, \sigma^2)$$

(Thus, y_i is = to a fixed μ plus with a ϵ_i, whereas $y_i \sim N(\mu, \sigma^2)$. These two expressions are not exactly the same, but they are connected.)

Likelihood: $P(y|\theta)$ $(P(y, \theta) = P(\theta)P(y|\theta))$

Prior: $P(\theta)$

Posterior:

$$P(\theta|y) = \frac{P(\theta, y)}{P(y)} = \frac{P(\theta, y)}{\int P(\theta, y) d\theta} = \frac{P(\theta)P(y|\theta)}{\int P(\theta, y) d\theta} = \frac{P(\theta)P(y|\theta)}{\int P(\theta)P(y|\theta) d\theta}$$

Markts

(1) The only random variables in frequentist models are the data. In contrast, Bayesian paradigm also uses probability to describe one's uncertainty about unknown model parameters.

(2) Consider the following model for binary outcome y:

$y_i|\theta_i \sim Bern(\theta_i), i = 1, 2, 3...6$

$\theta_i|\alpha \sim Beta(a, b_0), i = 1, 2, 3...6$

$\alpha \sim Exp(r_0)$

Thus, the joint distribution of all variable:

$$\prod_{i=1}^{6}[\theta_i^{y_i}(1-\theta_i)^{1-y_i}\frac{\Gamma(a+b_0)}{\Gamma(a)\Gamma(b_0)}\theta_i^{a-1}(1-\theta_i)^{b_0-1}]r_0e^{-r_0\alpha}$$

(Question: Why not write it as a_i?)

Chapter 13

Trying

13.1 The Basic Idea

$$L(\beta, D|Y) = \int \prod_{i=1}^{n} f_{y_i|u}(y_i|b,\beta) f_{b_i}(b_i|D) db_i$$

Notations :

y: Variable for the fixed effect

b: Variable for the random effect

β: Parameters for the fixed effect

D: Parameters for the random effect

The dimension of the integral is equal to the levels of the random factors (i.e., the number of observations).

13.2 Model and R Code

Covarance Matrix for n observations:

$$V = \sigma^2 \begin{bmatrix} 1 & \rho & \rho^2 & ... & \rho^{n-1} & \rho^n \\ \rho & 1 & \rho & ... & \rho^{n-2} & \rho^{n-1} \\ \rho^2 & \rho & 1 & ... & \rho^{n-3} & \rho^{n-2} \\ ... & & & & & \\ \rho^n & \rho^{n-1} & \rho^{n-2} & ... & \rho & 1 \end{bmatrix}$$

The inverse matrix is as follows:

$$Q = V^{-1} = \frac{1}{\sigma^2(1-\rho)} \begin{bmatrix} 1 & -\rho & 0 & ... & 0 & 0 \\ -\rho & 1+\rho^2 & -\rho & ... & 0 & 0 \\ 0 & -\rho & 1+\rho^2 & ... & 0 & 0 \\ ... \\ 0 & 0 & 0 & ... & 1+\rho^2 & -\rho \\ 0 & 0 & 0 & ... & -\rho & 1 \end{bmatrix}$$

$$N(-\sum_{j\neq k} Q_{kj} b_j^{(m)} Q_{kk}^{-1}, Q_{kk}^{-1})$$

$$lnL(\beta,\theta;Y,b) = \ell = lnf_{Y|b}(Y|b,\beta) + lnf_b(b|\theta)$$

$$a^{(m+1)} = a^{(m)} + \tau(a^{(m)})^{-1} S(a^{(m)})$$

Where,

$$\tau(a) = -E(\frac{\partial^2 \ell}{\partial\alpha\partial\alpha'}|Y)$$

$$S(a) = E(\frac{\partial \ell}{\partial \alpha}|Y)$$

Note that, α is a combination of two sets of parameters.

$$\alpha = \begin{pmatrix} \beta \\ b \end{pmatrix}$$

$$\ell = \sum_{i=1}^{n} \{[y_i ln(\frac{e^{\beta^T x_i + b_i}}{1+e^{\beta^T x_i + b_i}}) + (1-y_i) ln(1 - \frac{e^{\beta^T x_i + b_i}}{1+e^{\beta^T x_i + b_i}})] + lnf_b(b_i|\theta)\}$$

$$\frac{\partial lnf(Y|b,\beta)}{\partial \beta} = X^{'}(Y - E(Y|b))$$

$$\frac{\partial lnf(Y|b,\beta)}{\partial \beta \partial \beta'} = -X^{'}(Y - E(Y|b))$$

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

The Newton Raphson algorithm needs the second order.

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

Using the Newton Raphson, the following code calculates the basic logistic model, without any random effects. As we can see, it produces the same result as the R generic function of GLM. Note that, the function of Expit and the variables of y and are from the last block of R code.

```
#n<-10
x_intercept<-rep(1,n)
x_intercept<-as.matrix(x_intercept)

tolerance=1e-3
max_its=2000;iteration=1;difference=2
W<-matrix(0,n,n)
beta_old<-0.4

while(difference>tolerance & iteration<max_its)
  {
  # The first order
  f_firstorder<-t(x_intercept)%*%(y-Expit(x_intercept%*%beta_old))

  # The second order
  diag(W) = Expit(x_intercept%*%beta_old)*(1-Expit(x_intercept%*%beta_old))

  f_secondorder<--t(x_intercept)%*%W%*%x_intercept

  # 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

glm(y~1, family=binomial)$coefficients
```

```
#install.packages("CVTuningCov")
library(CVTuningCov) # Will be used to generate AR1 matrix

set.seed(123)
y<-c(1,1,1,0,0,1,1,0,1,0) ## observations

n=length(y) # the number of observations


#Establish the exp function
Expit<-function(x){exp(x)/(1+exp(x))}
#Y: observations
#b: random effect
#beta_0:fixed effect->intercept (or, mean of Y)

log_pdf_function<-function(Y,b,beta_0)
  {mean_prob<-Expit(beta_0+b)
  dbinom(Y,1,mean_prob,log = TRUE)
  }

b_records<-rep(0,n)  #Initial values for the random effect
rho_records<-0.5 #Initial value for rho
sigma_recoards<-2  #Initial value for sigma
mean_0<-0 # Initial mean value for normal distribution (of the random effect)
beta<-0.5 # Initial value for the intercept of Y


f_random<-function(sigma_recoards, rho_records,beta)
{

co_matrix<-(sigma_recoards^2)*AR1(n,rho_records) # covariance matrix
co_matrix_inverse<-solve(co_matrix)  # inverse covariance matrix


for (k in 1:n)
 {
  # Variance for the random effect
  sd_0<-1/(co_matrix_inverse[k,k])
```

```
  for(j in 1:n)
      { # Make sure that k is not equal to j, otherwise 0
        Q_kj<-ifelse(j!=k,co_matrix_inverse[k,j],0)
        # Calculate the mean for the random effect; sum of mean in a loop
        mean_0<-mean_0-(Q_kj/co_matrix_inverse[k,k])*b_records[j]
      }
  # Draw a random number from the normal distribution for the random effect
  b_candidate<-rnorm(1,mean_0,sd_0)

  current_lp<-log_pdf_function(y[k],b_records[k],beta)
  candidate_lp<-log_pdf_function(y[k],b_candidate,beta)

  Smaller_value<-min(exp(candidate_lp-current_lp),1)
  # Draw a random number from the uniform distribution
  Random_probability<-runif(1)
  # Update b (i.e., random variable)
  b_records[k]<-ifelse(Random_probability<Smaller_value,b_candidate,b_records[k])
}

return(b_records)
}

# Print result
#b_records<-f_random(1,0.8,0.3)
```

In the following, I will try to add the random effect.

```
x_intercept<-rep(1,n)
x_intercept<-as.matrix(x_intercept)

#We need to set random starting values.

tolerance=1e-3
max_its=2000;iteration=1;difference=2
W<-matrix(0,n,n)
beta_old<-0.4

while(difference>tolerance & iteration<max_its)
  {
  b_records<-f_random(2,0.9,beta_old)
  print("first")
  print(beta_old)
  print(b_records)
  # The first order
```

```
  f_firstorder<-t(x_intercept)%*%(y-Expit(x_intercept%*%beta_old+b_records))
  print(f_firstorder)
  # The second order
  diag(W) = Expit(x_intercept%*%beta_old+b_records)*(1-Expit(x_intercept%*%beta_old+b_r

  f_secondorder<--t(x_intercept)%*%W%*%x_intercept

  # 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
```

13.3 glmmTMB package

```
# https://becarioprecario.bitbucket.io/inla-gitbook/ch-intro.html
#https://cran.r-project.org/web/packages/glmmTMB/vignettes/covstruct.html
#install.packages("glmmTMB")
library(glmmTMB)

times <- factor(1:n)
levels(times)
group <- factor(rep(1,n))
dat0 <- data.frame(y,times,group)

glmmTMB(y ~ ar1(times + 0 | group), data=dat0)
```