My Project

Bill Last Updated:

20 January, 2020

Contents

Preface: Motivation

This is my MS project (in progress). 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

Practice: Learning on the Battle Field

1.1 R code

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

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

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


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

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

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

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

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

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



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

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

n_counting<-0  # help counting the number of pairs

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

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

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


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

PT_L="Pass"
TG_L=0
FD_L=0

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

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

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

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

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

      n_counting<-n_counting+1
  }
}

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

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

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

Remarks

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

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

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

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

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

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

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

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

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

```
summary(mylogit3)
```

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

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

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

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

1.2 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 2

Project Draft

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

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

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

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

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

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

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

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

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

2.1 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
```

```
# Select the first list from "Xl", and then add other 3 lists--> 4000 * 3
Xs = Xl[[1]]
for(k in 2:K) Xs = rbind(Xs, Xl[[k]])

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

a1
```

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

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

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

```
    } else{
      aa = runif(1)
      if(aa<alpha){
        ub[k] = uk
        llk0 = llk1
      }
    }
  }
  return(ub)
}

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

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

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

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

2.2 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/

2.3 References

1. Data

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

Chapter 3

Trying

3.1 The Basic Idea

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

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).

3.2 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.

```
y<-c(1,1,1,0,0,1,1,0,1,0) ## observations
n=length(y) # the number of observations
Expit<-function(x){exp(x)/(1+exp(x))}

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_

  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
```

3.3 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)
```