카운트 자료의 분석

```r
rm(list=ls(all=TRUE))
pacman::p_load(rjags, runjags, ggmcmc, tidyverse)
setwd('C:/Users/dnskd/Desktop/20Spring')
knitr::opts_chunk$set(dev="CairoPNG", dpi = 110)
```

mers data ——————————————————-

```r
mers = read.table('./Bayesian/week8/mers.txt', header = T)
head(mers)
```

```
##    day diag_new diag_sum checkout_new checkout_sum death_new death_sum car
e
## 1 5/20        2        2            0            0         0         0
2
## 2 5/21        1        3            0            0         0         0
3
## 3 5/22        0        3            0            0         0         0
3
## 4 5/23        0        3            0            0         0         0
3
## 5 5/24        0        3            0            0         0         0
3
## 6 5/25        0        3            0            0         0         0
3
##   quarantin quaratine_off
## 1         3             0
## 2        64             0
## 3        58             0
## 4        61             0
## 5        62             0
## 6        62             0
```

```r
day = as.character(mers$day)
diag = mers$diag_new
y = diag; n = length(y)
n.day = c(1:n)
plot(n.day, y, xlab = "day", ylab = "n_diag", type = 'l')
points(n.day, y)

x1 <- c(rep(1, 19), rep(0, n-19))
x2 <- c(c(1:19), rep(0, n-19))
x3 <- c(rep(0, 19), rep(1, n-19))
x4 <- c(rep(0, 19), c(1:(n-19)))
X = cbind(x1, x2, x3, x4)
```

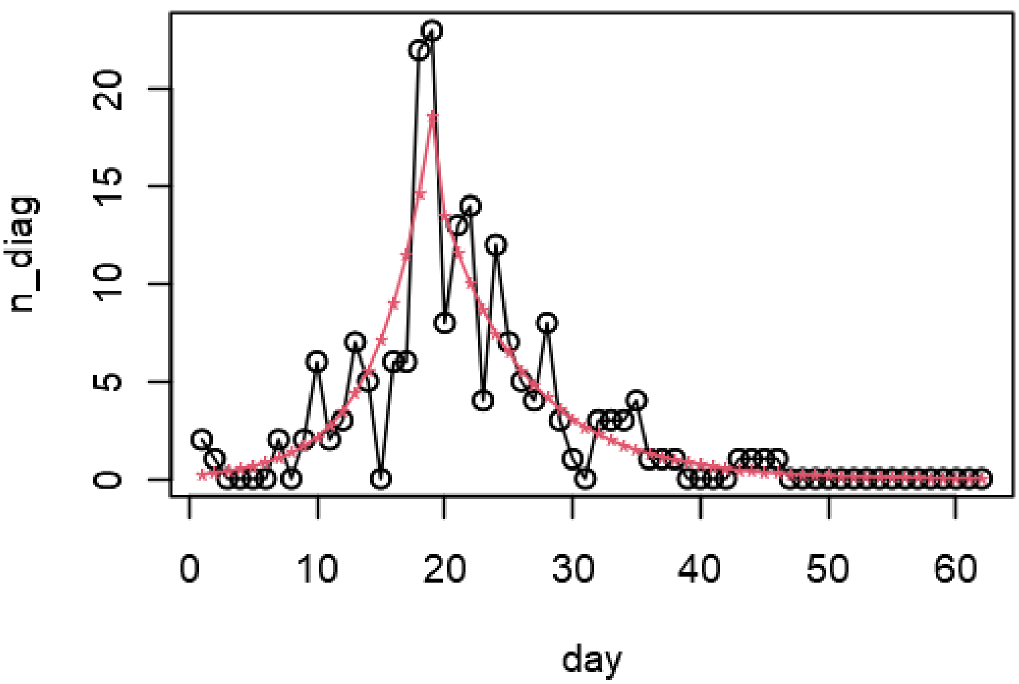
```r
data = data.frame(y, X)
p = ncol(X)

modelString="
model
{
  for(i in 1:length(y)){
  y[i] ~ dpois(lambda[i])
  log(lambda[i]) <- inprod(X[i,], beta[])
  }
  for (i in 1:p){beta[i] ~ dnorm(mu.beta[i], Tau.beta[i])}
}
"

writeLines(modelString, "./Bayesian/week8/model_pois.txt")

# prior parameters
mu.beta = rep(0, p)
Tau.beta = rep(0.01, p)
dataList = list(p = p, y = y, X = X, mu.beta = mu.beta, Tau.beta = Tau.beta)
initsList = list(beta = mu.beta)

require(rjags)
jagsModel.pois = jags.model(file = "./Bayesian/week8/model_pois.txt", data =
dataList, inits = initsList,
                            n.chains = 3, n.adapt = 1000)

## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 62
##    Unobserved stochastic nodes: 4
##    Total graph size: 510
##
## Initializing model

update(jagsModel.pois, n.iter = 3000)
codaSamples = coda.samples(jagsModel.pois, variable.names = c("beta"), thin =
 1, n.chains = 3, n.iter = 10000)
coda::gelman.diag(codaSamples)

## Potential scale reduction factors:
##
##         Point est. Upper C.I.
## beta[1]          1       1.00
## beta[2]          1       1.00
## beta[3]          1       1.01
## beta[4]          1       1.00
```

```
##
## Multivariate psrf
##
## 1
```

```
summary(codaSamples)
```

```
##
## Iterations = 4001:14000
## Thinning interval = 1
## Number of chains = 3
## Sample size per chain = 10000
##
## 1. Empirical mean and standard deviation for each variable,
##    plus standard error of the mean:
##
##            Mean      SD  Naive SE Time-series SE
## beta[1] -1.6593 0.45163 2.608e-03      0.0196162
## beta[2]  0.2412 0.02830 1.634e-04      0.0012143
## beta[3]  2.7502 0.14883 8.593e-04      0.0020386
## beta[4] -0.1477 0.01535 8.863e-05      0.0002096
##
## 2. Quantiles for each variable:
##
##            2.5%     25%     50%     75%   97.5%
## beta[1] -2.5859 -1.9594 -1.6458 -1.3489 -0.8121
## beta[2]  0.1871  0.2219  0.2406  0.2602  0.2985
## beta[3]  2.4565  2.6498  2.7513  2.8517  3.0375
## beta[4] -0.1787 -0.1580 -0.1472 -0.1372 -0.1192
```

```
mcmcSamples = as.matrix(codaSamples)
beta.hat = apply(mcmcSamples, 2, mean)
lambda.hat = exp(X %*% beta.hat)
points(n.day, y, xlab = "day", ylab = "n_diag")
lines(n.day, lambda.hat, col = 2)
points(n.day, lambda.hat, col = 2, pch = "*")
```




```
jagsModel.pois = jags.model(file = "./Bayesian/week8/model_pois.txt", data =
dataList, inits = initsList,
                            n.chains = 3, n.adapt = 1000)

## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 62
##    Unobserved stochastic nodes: 4
##    Total graph size: 510
##
## Initializing model

dic.pois = dic.samples(jagsModel.pois, 10000); dic.pois

## Mean deviance:  192.9
## penalty 3.951
## Penalized deviance: 196.8

X2 = cbind(x1, x2, x2*x2, x3, x4, x4*x4)
p = ncol(X2)
mu.beta = rep(0, p)
```

```r
Tau.beta = rep(0.01, p)
dataList = list(p = p, y = y, X = X2, mu.beta = mu.beta, Tau.beta = Tau.beta)
initsList = list(beta = mu.beta)
jagsModel.pois2 = jags.model(file = "./Bayesian/week8/model_pois.txt", data =
 dataList, inits = initsList, n.chains = 3, n.adapt = 1000)
```

```
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 62
##    Unobserved stochastic nodes: 6
##    Total graph size: 640
##
## Initializing model
```

```r
dic.pois = dic.samples(jagsModel.pois2, 10000); dic.pois
```

```
## Mean deviance:  188.7
## penalty 5.725
## Penalized deviance: 194.4
```

KL data ———————————————————————-

```r
rm(list = ls())
KL = read.csv("./Bayesian/week8/KL1.csv")
head(KL)
```

```
##   team1 team2 score1 score2 hometeam
## 1  광주  대구      1      0     광주
## 2  상주  강원      1      2     상주
## 3  울산  포항      2      1     울산
## 4  서울  수원      1      1     서울
## 5  인천  제주      0      1     인천
## 6  전북  전남      2      1     전북
```

```r
table(KL$team1)
```

```
##
## 강원 광주 대구 부산 상주 서울 수원 아산 울산 인천 전남 전북 제주 포항
##   19   19   19    2   20   19   19    1   19   19   19   19   19   19
```

```r
table(KL$team2)
```

```
##
## 강원 광주 대구 부산 상주 서울 성남 수원 아산 울산 인천 전남 전북 제주 포항
##   19   19   19    1   20   19    1   19    1   19   19   19   19   19   19
```

```r
# 4(부산) 7(성남) 9(아산) 제거
KL = KL %>% filter(team1 != "부산" & team1 != "아산" &
                     team2 != "부산" & team2 != "성남" & team2 != "아산")
name.team1 = data.frame(num.team1 = 1:12, team1 = c("강원", "광주", "대구", "
상주", "서울",
                                                     "수원", "울산", "인천", "
전남", "전북", "제주", "포항"))
name.team2 = data.frame(num.team2 = 1:12, team2 = c("강원", "광주", "대구", "
상주", "서울",
                                                     "수원", "울산", "인천", "
전남", "전북", "제주", "포항") )
KL = merge(KL, name.team1, by = c("team1"))
KL = merge(KL, name.team2, by = c("team2"))

modelString="
model{
  for(i in 1:n){
  y1[i] ~ dpois(lambda1[i])
  y2[i] ~ dpois(lambda2[i])
  log(lambda1[i]) = mu + home + a[ht[i]] + d[at[i]]
  log(lambda2[i]) = mu + home + a[at[i]] + d[ht[i]]
  }

  for(k in 2:K){
  a[k] ~ dnorm(0, 0.0001)
  d[k] ~ dnorm(0, 0.0001)
  }
  a[1] = -sum(a[2:K])
  d[1] = -sum(d[2:K])

  mu ~ dnorm(0, 0.0001)
  home ~ dnorm(0, 0.0001)
}
"

writeLines(modelString, "./Bayesian/week8/model_soccer_pois.txt")

y1 = KL$score1; y2 = KL$score2
ht = KL$num.team1; at = KL$num.team2
```

```r
n = nrow(KL)
K = length(unique(KL$team1))
initsList = list(mu = 0, home = 0)
dataList = list(n = n, K = K, at = at, ht = ht, y1 = y1, y2 = y2)
jagsModel = jags.model("./Bayesian/week8/model_soccer_pois.txt", inits = init
sList, data = dataList, n.chains = 3, n.adapt = 1000)

## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 456
##    Unobserved stochastic nodes: 24
##    Total graph size: 1210
##
## Initializing model

update(jagsModel, n.iter = 3000)
codaSamples = coda.samples(jagsModel, variable.names = c("a", "d", "mu", "hom
e"), n.chains = 3, n.iter = 10000)
```

inference ———————————————————

```r
coda::gelman.diag(codaSamples)

## Potential scale reduction factors:
##
##        Point est. Upper C.I.
## a[1]        1.00        1.0
## a[2]        1.00        1.0
## a[3]        1.00        1.0
## a[4]        1.00        1.0
## a[5]        1.00        1.0
## a[6]        1.00        1.0
## a[7]        1.00        1.0
## a[8]        1.00        1.0
## a[9]        1.00        1.0
## a[10]       1.00        1.0
## a[11]       1.00        1.0
## a[12]       1.00        1.0
## d[1]        1.00        1.0
## d[2]        1.00        1.0
## d[3]        1.00        1.0
## d[4]        1.00        1.0
## d[5]        1.00        1.0
## d[6]        1.00        1.0
## d[7]        1.00        1.0
## d[8]        1.00        1.0
## d[9]        1.00        1.0
```

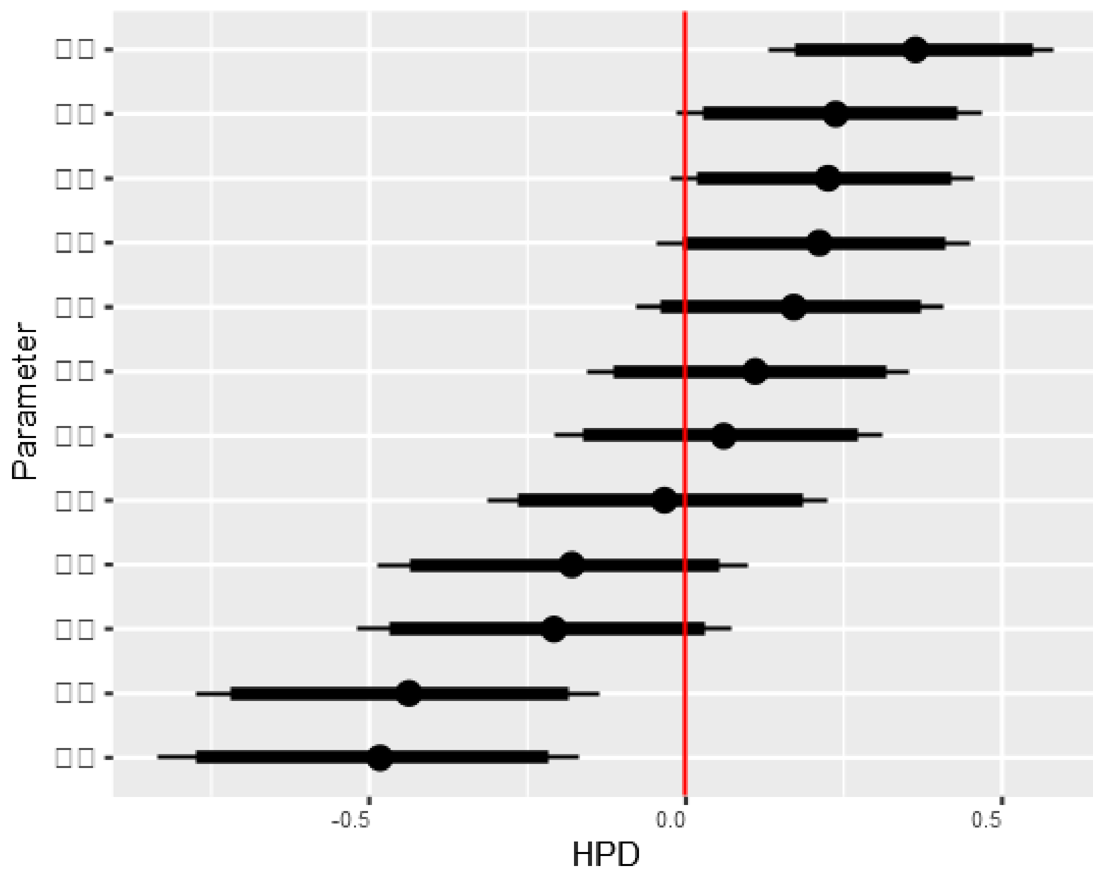
```
## d[10]       1.00        1.0
## d[11]       1.00        1.0
## d[12]       1.00        1.0
## home        7.83       19.8
## mu          7.83       19.8
##
## Multivariate psrf
##
## 5.42
```

```r
summary(codaSamples)
```

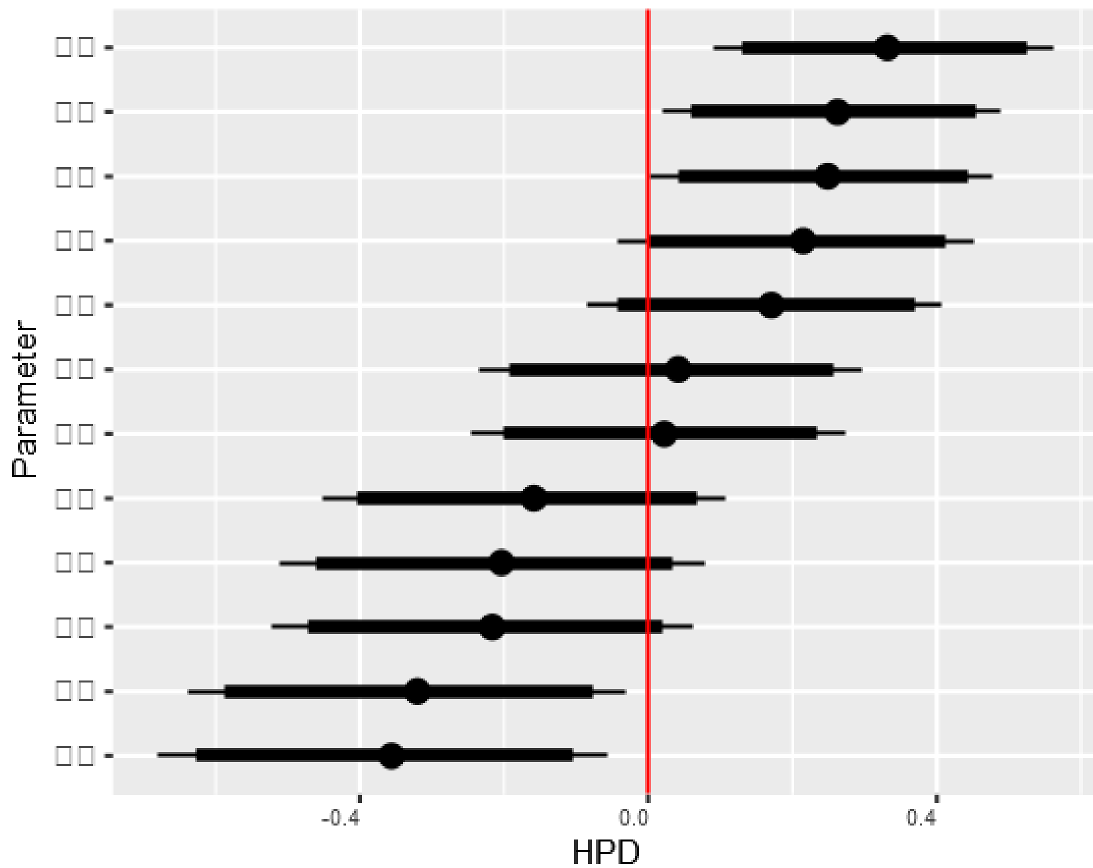
```
##
## Iterations = 4001:14000
## Thinning interval = 1
## Number of chains = 3
## Sample size per chain = 10000
##
## 1. Empirical mean and standard deviation for each variable,
##    plus standard error of the mean:
##
##            Mean     SD  Naive SE Time-series SE
## a[1]   0.20807 0.1265 0.0007302      0.0004035
## a[2]  -0.44199 0.1635 0.0009441      0.0019430
## a[3]  -0.03587 0.1375 0.0007939      0.0014937
## a[4]  -0.21156 0.1515 0.0008750      0.0018764
## a[5]   0.10773 0.1306 0.0007543      0.0014127
## a[6]   0.22229 0.1225 0.0007073      0.0012560
## a[7]  -0.18350 0.1490 0.0008604      0.0017840
## a[8]  -0.48840 0.1690 0.0009759      0.0021681
## a[9]   0.05807 0.1329 0.0007671      0.0014506
## a[10]  0.36177 0.1147 0.0006624      0.0011957
## a[11]  0.16898 0.1246 0.0007193      0.0013116
## a[12]  0.23441 0.1228 0.0007090      0.0013238
## d[1]   0.24572 0.1217 0.0007029      0.0003889
## d[2]   0.16774 0.1247 0.0007197      0.0013714
## d[3]   0.03887 0.1353 0.0007812      0.0015509
## d[4]   0.26011 0.1196 0.0006904      0.0012997
## d[5]  -0.20743 0.1495 0.0008634      0.0019180
## d[6]  -0.22010 0.1497 0.0008641      0.0017817
## d[7]  -0.16318 0.1434 0.0008277      0.0016876
## d[8]   0.02005 0.1321 0.0007628      0.0015200
## d[9]   0.33065 0.1204 0.0006950      0.0013260
## d[10] -0.35914 0.1597 0.0009222      0.0019826
## d[11] -0.32477 0.1559 0.0008999      0.0019507
## d[12]  0.21149 0.1255 0.0007248      0.0013386
## home  -4.65196 5.0166 0.0289634      0.5178245
## mu     4.89190 5.0161 0.0289607      0.5193695
##
```

```
## 2. Quantiles for each variable:
##
##               2.5%      25%      50%      75%    97.5%
## a[1]    -0.047131  0.12333  0.21107  0.29431  0.44898
## a[2]    -0.774510 -0.55108 -0.43757 -0.32792 -0.13736
## a[3]    -0.314653 -0.12864 -0.03369  0.05994  0.22397
## a[4]    -0.521300 -0.31026 -0.20829 -0.10734  0.07266
## a[5]    -0.157008  0.02069  0.11002  0.19717  0.35408
## a[6]    -0.024120  0.13995  0.22491  0.30546  0.45683
## a[7]    -0.487652 -0.28120 -0.18007 -0.08190  0.09946
## a[8]    -0.834276 -0.60017 -0.48326 -0.37418 -0.16798
## a[9]    -0.206128 -0.03037  0.06000  0.14960  0.31100
## a[10]    0.131140  0.28493  0.36322  0.44097  0.58148
## a[11]   -0.078925  0.08534  0.17068  0.25396  0.40719
## a[12]   -0.013535  0.15360  0.23705  0.31811  0.46780
## d[1]     0.002169  0.16427  0.24887  0.32895  0.47632
## d[2]    -0.085012  0.08558  0.17074  0.25231  0.40574
## d[3]    -0.235417 -0.04994  0.04163  0.13155  0.29525
## d[4]     0.018443  0.18059  0.26256  0.34194  0.48807
## d[5]    -0.510164 -0.30639 -0.20359 -0.10512  0.07762
## d[6]    -0.523968 -0.31829 -0.21646 -0.11806  0.06317
## d[7]    -0.452438 -0.25885 -0.15859 -0.06416  0.10721
## d[8]    -0.245267 -0.06740  0.02215  0.11001  0.27319
## d[9]     0.091567  0.25005  0.33158  0.41317  0.56223
## d[10]   -0.680061 -0.46523 -0.35606 -0.24901 -0.05575
## d[11]   -0.638928 -0.42848 -0.32020 -0.21696 -0.03118
## d[12]   -0.043631  0.12779  0.21478  0.29730  0.45081
## home  -13.264399 -7.92476 -6.22504  1.05013  2.92011
## mu     -2.679549 -0.80564  6.46540  8.15999 13.50799

mcmcSamples = as.matrix(codaSamples)
para.hat = apply(mcmcSamples, 2, mean)
para.hat2 = apply(mcmcSamples, 2, sd)

ind.a = which(unlist(lapply(strsplit(varnames(codaSamples), split = ""), func
tion(x) x[1]))=="a")
ind.d = which(unlist(lapply(strsplit(varnames(codaSamples), split = ""), func
tion(x) x[1]))=="d")

aa = sort(c("강원", "광주", "대구", "상주", "서울",
            "수원", "울산", "인천", "전남", "전북", "제주", "포항"))

P1 = c()
P2 = c()
for(i in 1:K){
  P1[i] = paste("a", "[",i,"]", sep = "")
  P2[i] = paste("d", "[",i,"]", sep = "")
}
```

```r
P1 = data.frame(Parameter = P1, Label = aa)
P2 = data.frame(Parameter = P2, Label = aa)

ggs1 = ggs(codaSamples[, ind.a], par_labels = P1)
p1 = ggs_caterpillar(ggs1)
p1 = p1 + geom_vline(xintercept = 0, col = 'red')
plot(p1)
```




```r
ggs2 = ggs(codaSamples[,ind.d], par_labels = P2)
p2 = ggs_caterpillar(ggs2, horizontal = TRUE)
p2 = p2 + geom_vline(xintercept = 0, col = 'red')
plot(p2)
```




```
tail(KL)

##     team2 team1 score1 score2 hometeam num.team1 num.team2
## 223  포항   제주       3      0      제주        11        12
## 224  포항   광주       0      4      광주         2        12
## 225  포항   대구       2      1      대구         3        12
## 226  포항   울산       2      1      울산         7        12
## 227  포항   인천       2      0      인천         8        12
## 228  포항   전남       1      3      전남         9        12
```

Prediction ————————————————-

```
y1[227:228] = y2[227:228] = NA
initsList = list(mu = 0, home = 0)
dataList = list(n = n, K = K, at = at, ht = ht, y1 = y1, y2 = y2)
jagsModel.pois = jags.model("./Bayesian/week8/model_soccer_pois.txt", inits =
 initsList, data = dataList, n.chains = 3, n.adapt = 1000)
```

```
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 452
##    Unobserved stochastic nodes: 28
##    Total graph size: 1210
##
## Initializing model
```

```
update(jagsModel.pois, n.iter = 3000)
variable.names = c("a", "d", "mu", "home", "y1[227]", "y1[228]", "y2[227]", "
y2[228]")
codaSamples = coda.samples(jagsModel.pois, variable.names = variable.names,
n.chains = 3, n.iter = 10000)

summary(codaSamples)
```

```
##
## Iterations = 4001:14000
## Thinning interval = 1
## Number of chains = 3
## Sample size per chain = 10000
##
## 1. Empirical mean and standard deviation for each variable,
##    plus standard error of the mean:
##
##              Mean     SD  Naive SE Time-series SE
## a[1]      0.20767 0.1278 0.0007381      0.0004028
## a[2]     -0.44027 0.1659 0.0009581      0.0020739
## a[3]     -0.03613 0.1365 0.0007882      0.0015439
## a[4]     -0.20917 0.1494 0.0008628      0.0017546
## a[5]      0.10612 0.1292 0.0007459      0.0013655
## a[6]      0.22575 0.1219 0.0007038      0.0012794
## a[7]     -0.18071 0.1486 0.0008580      0.0017529
## a[8]     -0.51997 0.1721 0.0009935      0.0021707
## a[9]      0.07281 0.1348 0.0007785      0.0015289
## a[10]     0.36170 0.1161 0.0006702      0.0011899
## a[11]     0.16650 0.1266 0.0007309      0.0013557
## a[12]     0.24570 0.1255 0.0007245      0.0013165
## d[1]      0.24489 0.1223 0.0007059      0.0003930
## d[2]      0.16515 0.1253 0.0007235      0.0013475
## d[3]      0.03400 0.1348 0.0007780      0.0015291
## d[4]      0.25824 0.1217 0.0007026      0.0013067
## d[5]     -0.20762 0.1486 0.0008580      0.0017303
## d[6]     -0.21957 0.1500 0.0008662      0.0017720
## d[7]     -0.16543 0.1436 0.0008290      0.0017535
## d[8]      0.04831 0.1335 0.0007707      0.0015177
## d[9]      0.32130 0.1219 0.0007038      0.0013573
```

```
## d[10]   -0.35767 0.1611 0.0009299      0.0020631
## d[11]   -0.32809 0.1564 0.0009030      0.0020392
## d[12]    0.20648 0.1288 0.0007434      0.0014866
## home    -3.01014 2.2633 0.0130669      0.8670726
## mu       3.25032 2.2631 0.0130662      0.8650813
## y1[227]  0.95470 0.9970 0.0057562      0.0060618
## y1[228]  1.71023 1.3432 0.0077548      0.0084750
## y2[227]  1.73157 1.3451 0.0077661      0.0082142
## y2[228]  2.28400 1.5615 0.0090153      0.0096215
##
## 2. Quantiles for each variable:
##
##              2.5%      25%      50%      75%    97.5%
## a[1]    -0.050165  0.12232  0.20953  0.29565  0.45186
## a[2]    -0.773845 -0.55062 -0.43639 -0.32549 -0.12626
## a[3]    -0.310844 -0.12797 -0.03328  0.05765  0.22183
## a[4]    -0.513017 -0.30906 -0.20569 -0.10685  0.07531
## a[5]    -0.156169  0.02129  0.10931  0.19451  0.35094
## a[6]    -0.019330  0.14534  0.22752  0.30846  0.46170
## a[7]    -0.484742 -0.27775 -0.17713 -0.07927  0.10093
## a[8]    -0.866263 -0.63464 -0.51752 -0.40130 -0.19283
## a[9]    -0.197561 -0.01541  0.07474  0.16368  0.32990
## a[10]    0.125030  0.28525  0.36403  0.44009  0.58308
## a[11]   -0.086987  0.08210  0.16873  0.25287  0.40982
## a[12]   -0.003992  0.16181  0.24730  0.33165  0.48666
## d[1]    -0.001011  0.16401  0.24665  0.32891  0.47775
## d[2]    -0.089897  0.08129  0.16765  0.25124  0.40333
## d[3]    -0.236400 -0.05621  0.03642  0.12665  0.29288
## d[4]     0.011862  0.17867  0.26038  0.34004  0.49137
## d[5]    -0.506113 -0.30560 -0.20525 -0.10654  0.07655
## d[6]    -0.525934 -0.31736 -0.21583 -0.11801  0.06505
## d[7]    -0.456056 -0.26072 -0.16154 -0.06616  0.10406
## d[8]    -0.222890 -0.03949  0.05185  0.13899  0.30211
## d[9]     0.076541  0.24094  0.32231  0.40370  0.55497
## d[10]   -0.682446 -0.46527 -0.35449 -0.24638 -0.05145
## d[11]   -0.646969 -0.43039 -0.32493 -0.21989 -0.03486
## d[12]   -0.055603  0.12126  0.20953  0.29339  0.45288
## home    -7.723601 -4.40130 -3.13760 -1.92980  2.22283
## mu      -1.988531  2.17314  3.37844  4.64097  7.95280
## y1[227]  0.000000  0.00000  1.00000  1.00000  3.00000
## y1[228]  0.000000  1.00000  2.00000  2.00000  5.00000
## y2[227]  0.000000  1.00000  2.00000  3.00000  5.00000
## y2[228]  0.000000  1.00000  2.00000  3.00000  6.00000
```

```
KL2 = cbind(c(1:nrow(KL)) , KL)
KL2 %>% filter(team1 == "서울" & team2 == "제주")
```

```
##   c(1:nrow(KL)) team2 team1 score1 score2 hometeam num.team1 num.team2
## 1           193  제주  서울      0      0     서울         5        11
## 2           197  제주  서울      0      0     서울         5        11
## 3           201  제주  서울      3      2     서울         5        11
```

```
KL2 %>% filter(team1 == "전북" & team2 == "수원")
```

```
##   c(1:nrow(KL)) team2 team1 score1 score2 hometeam num.team1 num.team2
## 1           103  수원  전북      2      0     전북        10         6
## 2           110  수원  전북      2      3     전북        10         6
```

음이항 로그 선형 모형

```
modelString = "
model{
  for(i in 1:n){
  y1[i] ~ dnegbin(pi1[i], r1[i])
  pi1[i] = r1[i]/(r1[i] + lambda1[i])
  y2[i] ~ dnegbin(pi2[i], r2[i])
  pi2[i] = r2[i]/(r2[i] + lambda2[i])
  log(lambda1[i]) = mu + home + a[ht[i]] + d[at[i]]
  log(lambda2[i]) = mu + a[at[i]] + d[ht[i]]
  r1[i] ~ dgamma(0.001, 0.001)
  r2[i] ~ dgamma(0.001, 0.001)
  }

  for(k in 2:K){
  a[k] ~ dnorm(0, 0.0001)
  d[k] ~ dnorm(0, 0.0001)
  }
  a[1] = -sum(a[2:K])
  d[1] = -sum(d[2:K])

  mu ~ dnorm(0, 0.0001)
  home ~ dnorm(0, 0.0001)
}
"

writeLines(modelString, "./Bayesian/week8/model_soccer_nb.txt")


jagsModel.pois = jags.model("./Bayesian/week8/model_soccer_pois.txt", inits =
 initsList, data = dataList, n.chains = 3, n.adapt = 1000)
```

```
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 452
##    Unobserved stochastic nodes: 28
##    Total graph size: 1210
##
## Initializing model
```

```
jagsModel.nb = jags.model("./Bayesian/week8/model_soccer_nb.txt", inits = ini
tsList, data = dataList, n.chains = 3, n.adapt = 1000)
```

```
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 452
##    Unobserved stochastic nodes: 484
##    Total graph size: 2843
##
## Initializing model
```

```
dic.pois = dic.samples(jagsModel.pois, 10000)
dic.nb = dic.samples(jagsModel.nb, 10000)
dic.pois; dic.nb
```

```
## Mean deviance:  1291
## penalty 23.06
## Penalized deviance: 1314
```

```
## Mean deviance:  1150
## penalty 174.8
## Penalized deviance: 1325
```