Lecture 18 Rscript, Poisson lognormal model

Example dataset

The data is given in `lymphocyte.csv`, which is available on Canvas. It consists of 84 lymphocyte counts (y), along with recorded dosage (d) and log cell counts (log_count). We will fit a Poisson regression to this data, while allowing for over-dispersion by including an normal error term in the log link function. This means the vector of link functions can be represented as

$$\log(\boldsymbol{\lambda}) = \mathbf{X}\boldsymbol{\beta} + \boldsymbol{\epsilon}, \quad \boldsymbol{\epsilon} \sim \mathcal{N}(0, \sigma^2\mathbf{I})$$

where $\mathbf{X}$ is the predictor matrix whose columns will contain an intercept, column for dosage and column of log cell counts.

Assume a flat prior for $\boldsymbol{\beta}$ and $\text{Ga}(\alpha, \gamma)$ prior for the precision.

Aims for this coding session.

We want to show that the data augmentation strategy we outlined in the notes does give proper Bayesian inference. To do this, we will compare our results to those obtained using stan. Secondly, we want to check the impact of inappropriate data augmentation.

```
#Inputs:
#y: vector of responses
#X: predictor matrix including intercept.
#sigma0:  initial value for residual standard deviation of link,
#iter: number of iterations
#burnin: number of initial iterations to throw out.
#a,b:   hyper-parameters of gamma prior for precision = inverse of the variance.

PoisLN.reg<-function(X,y,sigma0,iter,burnin,a,b){
n      <-length(y)
p      <-dim(X)[2]
loglambda0<-log(y+0.01)
sdpros <-1/sqrt(y)
XTX <-crossprod(X)
XTXinv <-solve(XTX)
H      <-XTXinv%*%t(X)
sigma2  <-sigma0^2

#function of updating
joint.fun <- function(y,x,xb,tau,a,b){
```

```
  p1<-dpois(y,exp(x),log=TRUE)
  p2<-dnorm(x,mean=xb,sd=1/sqrt(tau),log=TRUE)
  p3<-dgamma(tau,a,b,log=TRUE)
  return(p1+p2+p3)
}

#storing matrix
par<-matrix(0,iter,n+p+1)
library(MASS)

for(i in 1:iter){
#Update co-efficients and variance.
bhat  <- H%*%loglambda0
beta  <- mvrnorm(1,mu=bhat,Sigma=XTXinv*sigma2)
Xb    <- X%*%beta
SSE   <- sum((loglambda0-Xb)^2)
tau   <- rgamma(1,0.5*n+a,0.5*SSE+b)
sigma2<- 1/tau

#Update link (Sequence of independent Metropolis)
loglambda.cand <-rnorm(n,loglambda0,2.4*sdpros)
r  <- joint.fun(y,loglambda.cand,Xb,tau,a,b) - joint.fun(y,loglambda0,Xb,tau,a,b)
r[r>0]<-0
ind<-rbinom(n,1,exp(r) )
loglambda0<-ind*loglambda.cand+(1-ind)*loglambda0

par[i,]<-c(loglambda0,as.numeric(beta),sigma2)
}

par <- par[-c(1:burnin),]
colnames(par)<-c(paste('log(lambda)',1:n,sep=''),paste('beta',1:p,sep=''),'sigma2')
return(par)
}
```

Function one: Metropolis in Gibbs algorithm for fitting the Poisson lognormal model

```
lymphocyte<-read.csv(file='./lymphocyte.csv', head=TRUE)
y         <-lymphocyte$y
n         <-length(y)
X         <-cbind(rep(1,n),lymphocyte$d,lymphocyte$log_count)
N_i       <-11000

chain1<-PoisLN.reg(X=X,y=y,sigma0=1,iter=N_i,burnin=1000,a=0.001,b=0.001)
```

Fitting the model to the lymphocyte data

```
## Warning: package 'MASS' was built under R version 4.3.1
```

```
chain2<-PoisLN.reg(X=X,y=y,sigma0=0.2,iter=N_i,burnin=1000,a=0.001,b=0.001)
chain3<-PoisLN.reg(X=X,y=y,sigma0=5,iter=N_i,burnin=1000,a=0.001,b=0.001)
dim(chain1[5000+1:5000,])
```

```
## [1] 5000   88
```

```
dim(chain1[5001:10000,])
```

```
## [1] 5000   88
```

```
library(coda)
```

```
## Warning: package 'coda' was built under R version 4.3.1
```

```
rml1<-as.mcmc.list(as.mcmc((chain1[1:5000,])))
rml2<-as.mcmc.list(as.mcmc((chain2[1:5000,])))
rml3<-as.mcmc.list(as.mcmc((chain3[1:5000,])))
rml4<-as.mcmc.list(as.mcmc((chain1[5000+1:5000,])))
rml5<-as.mcmc.list(as.mcmc((chain2[5000+1:5000,])))
rml6<-as.mcmc.list(as.mcmc((chain3[5000+1:5000,])))
rml<-c(rml1,rml2,rml3,rml4,rml5,rml6)

#Gelman-Rubin diagnostic.
gelman.diag(rml)[[1]]
```

```
##                Point est. Upper C.I.
## log(lambda)1     1.000406   1.001155
## log(lambda)2     1.000739   1.001863
## log(lambda)3     1.000418   1.001158
## log(lambda)4     1.000626   1.001483
## log(lambda)5     1.000808   1.001836
## log(lambda)6     1.000552   1.001213
## log(lambda)7     1.000634   1.000963
## log(lambda)8     1.000763   1.001611
## log(lambda)9     1.001489   1.003560
## log(lambda)10    1.000212   1.000365
## log(lambda)11    1.000246   1.000748
## log(lambda)12    1.000540   1.001358
## log(lambda)13    1.000964   1.002573
## log(lambda)14    1.000449   1.000758
## log(lambda)15    1.000299   1.000680
## log(lambda)16    1.002653   1.006019
## log(lambda)17    1.000390   1.000867
## log(lambda)18    1.001521   1.003233
## log(lambda)19    1.001410   1.003530
## log(lambda)20    1.001147   1.002817
## log(lambda)21    1.000543   1.001471
## log(lambda)22    1.001144   1.002448
## log(lambda)23    1.001104   1.002975
## log(lambda)24    1.001731   1.004713
## log(lambda)25    1.001088   1.002576
## log(lambda)26    1.003898   1.009328
## log(lambda)27    1.001371   1.002401
## log(lambda)28    1.000375   1.001136
## log(lambda)29    1.001948   1.004604
## log(lambda)30    1.001542   1.003888
```

```
## log(lambda)31   1.002274   1.005239
## log(lambda)32   1.001145   1.002596
## log(lambda)33   1.001199   1.003073
## log(lambda)34   1.001198   1.003060
## log(lambda)35   1.001412   1.003079
## log(lambda)36   1.000430   1.001228
## log(lambda)37   1.000647   1.001575
## log(lambda)38   1.000669   1.001701
## log(lambda)39   1.002322   1.006076
## log(lambda)40   1.001232   1.002860
## log(lambda)41   1.001451   1.003877
## log(lambda)42   1.000915   1.002273
## log(lambda)43   1.001503   1.003443
## log(lambda)44   1.001513   1.003403
## log(lambda)45   1.000978   1.002556
## log(lambda)46   1.000817   1.001975
## log(lambda)47   1.000461   1.000677
## log(lambda)48   1.000804   1.001693
## log(lambda)49   1.001337   1.003023
## log(lambda)50   1.002983   1.007737
## log(lambda)51   1.001974   1.004545
## log(lambda)52   1.001229   1.002611
## log(lambda)53   1.001041   1.002423
## log(lambda)54   1.000690   1.001928
## log(lambda)55   1.000821   1.001587
## log(lambda)56   1.000664   1.001831
## log(lambda)57   1.000586   1.001717
## log(lambda)58   1.000605   1.001640
## log(lambda)59   1.001134   1.002735
## log(lambda)60   1.000500   1.000954
## log(lambda)61   1.001648   1.003704
## log(lambda)62   1.001238   1.003348
## log(lambda)63   1.000591   1.001651
## log(lambda)64   1.000878   1.002087
## log(lambda)65   1.001506   1.003909
## log(lambda)66   1.001553   1.003194
## log(lambda)67   1.001641   1.003497
## log(lambda)68   1.000610   1.001388
## log(lambda)69   1.002259   1.005602
## log(lambda)70   1.003060   1.007998
## log(lambda)71   1.000809   1.002287
## log(lambda)72   1.000689   1.001290
## log(lambda)73   1.001256   1.002944
## log(lambda)74   1.001876   1.004899
## log(lambda)75   1.001157   1.002839
## log(lambda)76   1.001211   1.002502
## log(lambda)77   1.001179   1.002668
## log(lambda)78   1.000344   1.000756
## log(lambda)79   1.001871   1.004514
## log(lambda)80   1.000977   1.002434
## log(lambda)81   1.000486   1.000705
## log(lambda)82   1.000600   1.001604
## log(lambda)83   1.001428   1.003269
## log(lambda)84   1.001308   1.003362
```

```
## beta1           1.001723   1.004556
## beta2           1.002187   1.005819
## beta3           1.001611   1.004293
## sigma2          1.000673   1.001762
```

```
#effective sample size.
effectiveSize(rml)
```

```
##  log(lambda)1  log(lambda)2  log(lambda)3  log(lambda)4  log(lambda)5
##      6339.210      5653.145      6429.627      6490.671      6625.916
##  log(lambda)6  log(lambda)7  log(lambda)8  log(lambda)9 log(lambda)10
##      6331.983      6604.808      6398.785      5189.711      6371.343
## log(lambda)11 log(lambda)12 log(lambda)13 log(lambda)14 log(lambda)15
##      6446.432      5051.186      6372.628      6493.856      6415.762
## log(lambda)16 log(lambda)17 log(lambda)18 log(lambda)19 log(lambda)20
##      5886.653      6719.253      6501.707      6615.775      6745.098
## log(lambda)21 log(lambda)22 log(lambda)23 log(lambda)24 log(lambda)25
##      6802.784      6529.217      5420.100      6308.015      6726.208
## log(lambda)26 log(lambda)27 log(lambda)28 log(lambda)29 log(lambda)30
##      5735.604      6390.557      6796.514      6289.982      4461.761
## log(lambda)31 log(lambda)32 log(lambda)33 log(lambda)34 log(lambda)35
##      6802.022      6650.617      6895.411      6303.266      6306.104
## log(lambda)36 log(lambda)37 log(lambda)38 log(lambda)39 log(lambda)40
##      6318.350      5930.243      6610.947      6565.058      6992.400
## log(lambda)41 log(lambda)42 log(lambda)43 log(lambda)44 log(lambda)45
##      6285.247      6385.153      6578.279      5491.899      6470.532
## log(lambda)46 log(lambda)47 log(lambda)48 log(lambda)49 log(lambda)50
##      6432.643      6425.921      6655.430      6382.979      6191.993
## log(lambda)51 log(lambda)52 log(lambda)53 log(lambda)54 log(lambda)55
##      5292.694      6412.635      6594.838      6051.509      6019.900
## log(lambda)56 log(lambda)57 log(lambda)58 log(lambda)59 log(lambda)60
##      6210.472      6284.821      6037.365      6306.588      6234.549
## log(lambda)61 log(lambda)62 log(lambda)63 log(lambda)64 log(lambda)65
##      6973.431      6818.539      6590.641      6527.195      5320.690
## log(lambda)66 log(lambda)67 log(lambda)68 log(lambda)69 log(lambda)70
##      6411.440      6068.756      6568.920      6225.111      6690.450
## log(lambda)71 log(lambda)72 log(lambda)73 log(lambda)74 log(lambda)75
##      6628.701      5210.671      6472.689      6368.613      6593.005
## log(lambda)76 log(lambda)77 log(lambda)78 log(lambda)79 log(lambda)80
##      5916.584      6602.692      6395.650      5248.584      6542.386
## log(lambda)81 log(lambda)82 log(lambda)83 log(lambda)84         beta1
##      6034.589      6616.283      6213.921      6666.671      5506.309
##         beta2         beta3        sigma2
##      5435.235      5804.194      3825.043
```

```
#Inputs:
#y: vector of responses
#X: predictor matrix including intercept.
#sigma0:  initial value for residual standard deviation of link,
#iter: number of iterations
#burnin: number of initial iterations to throw out.
```

```
#a,b:   hyper-parameters of gamma prior for precision.

BADPoisLN.reg<-function(X,y,sigma0,iter,burnin,a,b){
  n      <-length(y)
  p      <-dim(X)[2]
  loglambda0<-log(y+0.01)
  sdpros <-1/sqrt(y)
  XTX <-crossprod(X)
  XTXinv <-solve(XTX)
  H      <-XTXinv%*%t(X)
  sigma2  <-sigma0^2

  #storing matrix
  par<-matrix(0,iter,n+p+1)
  library(MASS)

  for(i in 1:iter){
    #Update co-efficients and variance.
    bhat  <- H%*%loglambda0
    beta  <- mvrnorm(1,mu=bhat,Sigma=XTXinv*sigma2)
    Xb    <- X%*%beta
    SSE   <- sum((loglambda0-Xb)^2)
    tau   <- rgamma(1,0.5*n+a,0.5*SSE+b)
    sigma2<- 1/tau

    #Update link, incorrect just using Gamma posterior
    loglambda0<-log(rgamma(n,a+y,b+1))

    par[i,]<-c(loglambda0,as.numeric(beta),sigma2)
  }

  par <- par[-c(1:burnin),]
  colnames(par)<-c(paste('log(lambda)',1:n,sep=''),paste('beta',1:p,sep=''),'sigma2')
  return(par)
}

chain4<-BADPoisLN.reg(X=X,y=y,sigma0=1,iter=N_i,burnin=1000,a=0.001,b=0.001)
chain5<-BADPoisLN.reg(X=X,y=y,sigma0=0.2,iter=N_i,burnin=1000,a=0.001,b=0.001)
chain6<-BADPoisLN.reg(X=X,y=y,sigma0=5,iter=N_i,burnin=1000,a=0.001,b=0.001)

library(coda)
bml1<-as.mcmc.list(as.mcmc((chain4[1:5000,])))
bml2<-as.mcmc.list(as.mcmc((chain5[1:5000,])))
bml3<-as.mcmc.list(as.mcmc((chain6[1:5000,])))
bml4<-as.mcmc.list(as.mcmc((chain4[5000+1:5000,])))
bml5<-as.mcmc.list(as.mcmc((chain5[5000+1:5000,])))
bml6<-as.mcmc.list(as.mcmc((chain6[5000+1:5000,])))
bml<-c(bml1,bml2,bml3,bml4,bml5,bml6)

#Gelman-Rubin diagnostic.
gelman.diag(bml)[[1]]
```

Function for Inappropriate data augmentation in Poisson-lognormal model

```
##                Point est. Upper C.I.
## log(lambda)1    1.0001698  1.0006807
## log(lambda)2    0.9998969  0.9999651
## log(lambda)3    0.9999268  1.0000121
## log(lambda)4    1.0003314  1.0010831
## log(lambda)5    1.0001299  1.0005880
## log(lambda)6    1.0001759  1.0006420
## log(lambda)7    0.9999913  1.0001984
## log(lambda)8    1.0000256  1.0002782
## log(lambda)9    1.0000749  1.0004652
## log(lambda)10   0.9998505  0.9998797
## log(lambda)11   1.0001164  1.0004919
## log(lambda)12   1.0002675  1.0009252
## log(lambda)13   0.9999069  1.0000216
## log(lambda)14   1.0003226  1.0009873
## log(lambda)15   1.0001039  1.0004574
## log(lambda)16   1.0001110  1.0005389
## log(lambda)17   1.0002163  1.0007422
## log(lambda)18   1.0001947  1.0006655
## log(lambda)19   0.9999585  1.0001647
## log(lambda)20   1.0001268  1.0004637
## log(lambda)21   1.0000783  1.0003762
## log(lambda)22   1.0000175  1.0002370
## log(lambda)23   1.0003002  1.0009202
## log(lambda)24   0.9999676  1.0001414
## log(lambda)25   0.9998803  0.9999587
## log(lambda)26   1.0001757  1.0006003
## log(lambda)27   1.0000604  1.0003798
## log(lambda)28   0.9999825  1.0001861
## log(lambda)29   0.9999626  1.0000663
## log(lambda)30   1.0000022  1.0002364
## log(lambda)31   0.9999162  1.0000213
## log(lambda)32   1.0004761  1.0013791
## log(lambda)33   1.0000353  1.0003082
## log(lambda)34   1.0005325  1.0016110
## log(lambda)35   0.9999574  1.0001785
## log(lambda)36   1.0001618  1.0006389
## log(lambda)37   1.0002943  1.0009089
## log(lambda)38   0.9998611  0.9999015
## log(lambda)39   1.0001062  1.0004730
## log(lambda)40   1.0001533  1.0005013
## log(lambda)41   1.0000295  1.0002925
## log(lambda)42   1.0001388  1.0006139
## log(lambda)43   1.0001060  1.0005102
## log(lambda)44   1.0001033  1.0005344
## log(lambda)45   1.0002769  1.0009703
## log(lambda)46   1.0000552  1.0003141
## log(lambda)47   1.0001631  1.0006516
## log(lambda)48   1.0002166  1.0007449
## log(lambda)49   1.0000454  1.0003141
## log(lambda)50   1.0001289  1.0004334
## log(lambda)51   0.9998718  0.9999404
## log(lambda)52   1.0003369  1.0011528
## log(lambda)53   0.9999729  1.0001115
```

```
## log(lambda)54  1.0003274  1.0010854
## log(lambda)55  1.0001925  1.0006901
## log(lambda)56  1.0000888  1.0003913
## log(lambda)57  0.9998880  0.9999269
## log(lambda)58  1.0000363  1.0003647
## log(lambda)59  0.9998523  0.9999028
## log(lambda)60  1.0002799  1.0009032
## log(lambda)61  0.9999880  1.0001500
## log(lambda)62  1.0000612  1.0003537
## log(lambda)63  0.9999613  1.0001724
## log(lambda)64  1.0002989  1.0010057
## log(lambda)65  1.0003465  1.0010949
## log(lambda)66  1.0000657  1.0003396
## log(lambda)67  0.9999717  1.0001615
## log(lambda)68  1.0002641  1.0007929
## log(lambda)69  0.9999147  1.0000224
## log(lambda)70  1.0000792  1.0004390
## log(lambda)71  0.9999504  1.0000121
## log(lambda)72  0.9999881  1.0002240
## log(lambda)73  1.0002025  1.0006432
## log(lambda)74  1.0002528  1.0007968
## log(lambda)75  1.0002171  1.0008559
## log(lambda)76  0.9998819  0.9999323
## log(lambda)77  1.0000015  1.0002366
## log(lambda)78  0.9999114  1.0000130
## log(lambda)79  1.0000180  1.0002934
## log(lambda)80  1.0000522  1.0003561
## log(lambda)81  1.0001976  1.0006855
## log(lambda)82  1.0002024  1.0006392
## log(lambda)83  1.0001412  1.0004711
## log(lambda)84  1.0002007  1.0007968
## beta1          1.0000924  1.0005366
## beta2          1.0001566  1.0007061
## beta3          1.0000718  1.0004840
## sigma2         1.0005482  1.0015579
```

```
#effective sample size.
effectiveSize(bml)
```

```
##  log(lambda)1  log(lambda)2  log(lambda)3  log(lambda)4  log(lambda)5
##      28756.59      29758.02      30000.00      30000.00      30086.92
##  log(lambda)6  log(lambda)7  log(lambda)8  log(lambda)9 log(lambda)10
##      30000.00      30091.24      30000.00      30000.00      30984.90
## log(lambda)11 log(lambda)12 log(lambda)13 log(lambda)14 log(lambda)15
##      31694.68      30704.08      28517.88      30000.00      30000.00
## log(lambda)16 log(lambda)17 log(lambda)18 log(lambda)19 log(lambda)20
##      29796.29      29752.31      30000.00      30000.00      30000.00
## log(lambda)21 log(lambda)22 log(lambda)23 log(lambda)24 log(lambda)25
##      30000.00      29892.62      30000.00      30000.00      30000.00
## log(lambda)26 log(lambda)27 log(lambda)28 log(lambda)29 log(lambda)30
##      30000.00      30000.00      30000.00      29006.12      30000.00
## log(lambda)31 log(lambda)32 log(lambda)33 log(lambda)34 log(lambda)35
##      29446.57      29340.79      30000.00      30000.00      30000.00
## log(lambda)36 log(lambda)37 log(lambda)38 log(lambda)39 log(lambda)40
```

```
##      30000.00      30752.43      29511.27      30000.00      29465.47
## log(lambda)41 log(lambda)42 log(lambda)43 log(lambda)44 log(lambda)45
##      29508.32      29417.83      30000.00      30000.00      30000.00
## log(lambda)46 log(lambda)47 log(lambda)48 log(lambda)49 log(lambda)50
##      30000.00      30000.00      30062.68      29358.89      30000.00
## log(lambda)51 log(lambda)52 log(lambda)53 log(lambda)54 log(lambda)55
##      29393.93      31032.51      30000.00      30000.00      30000.00
## log(lambda)56 log(lambda)57 log(lambda)58 log(lambda)59 log(lambda)60
##      30000.00      30000.00      30000.00      30533.36      30000.00
## log(lambda)61 log(lambda)62 log(lambda)63 log(lambda)64 log(lambda)65
##      30000.00      30064.01      30726.92      30000.00      30000.00
## log(lambda)66 log(lambda)67 log(lambda)68 log(lambda)69 log(lambda)70
##      28934.25      30000.00      30000.00      30000.00      30000.00
## log(lambda)71 log(lambda)72 log(lambda)73 log(lambda)74 log(lambda)75
##      30000.00      30095.59      31237.21      30000.00      30000.00
## log(lambda)76 log(lambda)77 log(lambda)78 log(lambda)79 log(lambda)80
##      30000.00      30264.37      30000.00      30000.00      30000.00
## log(lambda)81 log(lambda)82 log(lambda)83 log(lambda)84         beta1
##      30000.00      30000.00      31999.86      30000.00      30000.00
##         beta2         beta3        sigma2
##      30000.00      30000.00      28106.14
```

```
#Combining chains.

#Correct augmentation
chain.combine1<-rbind(chain1,chain2,chain3)
#Incorrect augmentation
chain.combine2<-rbind(chain4,chain5,chain6)


#Finding posterior means

colMeans(chain.combine1) #Correct augmentation.
```

Comparing results

```
##  log(lambda)1  log(lambda)2  log(lambda)3  log(lambda)4  log(lambda)5
##    4.67910679    3.77140102    4.32620598    4.62157064    5.18380233
##  log(lambda)6  log(lambda)7  log(lambda)8  log(lambda)9 log(lambda)10
##    4.70274320    4.74975369    4.77166353    3.77634159    4.45401021
## log(lambda)11 log(lambda)12 log(lambda)13 log(lambda)14 log(lambda)15
##    4.52846869    4.46504549    4.73958463    4.69683336    4.72010108
## log(lambda)16 log(lambda)17 log(lambda)18 log(lambda)19 log(lambda)20
##    3.88189654    4.37009166    4.62021670    4.92344171    4.74492271
## log(lambda)21 log(lambda)22 log(lambda)23 log(lambda)24 log(lambda)25
##    4.93216615    4.87244373    3.73063097    4.47400169    4.59177956
## log(lambda)26 log(lambda)27 log(lambda)28 log(lambda)29 log(lambda)30
##    4.49862973    5.07148688    5.13130572    4.85978811    3.58845128
## log(lambda)31 log(lambda)32 log(lambda)33 log(lambda)34 log(lambda)35
##    4.37751913    4.56364827    4.94023517    4.90960458    4.62701448
## log(lambda)36 log(lambda)37 log(lambda)38 log(lambda)39 log(lambda)40
```

```
##    4.64782100    3.88347378    4.39709399    4.50157178    4.75651454
## log(lambda)41 log(lambda)42 log(lambda)43 log(lambda)44 log(lambda)45
##    5.29523443    5.14999463    5.00973263    3.68183318    4.29286499
## log(lambda)46 log(lambda)47 log(lambda)48 log(lambda)49 log(lambda)50
##    4.42545185    4.81858836    4.81823766    4.67294940    4.77394765
## log(lambda)51 log(lambda)52 log(lambda)53 log(lambda)54 log(lambda)55
##    3.68310970    4.71011650    4.59161855    4.78410158    4.90998497
## log(lambda)56 log(lambda)57 log(lambda)58 log(lambda)59 log(lambda)60
##    4.91887181    4.67971982    3.93307053    4.35617897    4.33083009
## log(lambda)61 log(lambda)62 log(lambda)63 log(lambda)64 log(lambda)65
##    4.89501653    4.88346917    4.86505331    4.80679145    4.06744541
## log(lambda)66 log(lambda)67 log(lambda)68 log(lambda)69 log(lambda)70
##    4.40821178    4.90283860    4.74404716    4.56992624    4.85863027
## log(lambda)71 log(lambda)72 log(lambda)73 log(lambda)74 log(lambda)75
##    4.81756281    3.74104250    4.41756914    4.38338363    4.88430249
## log(lambda)76 log(lambda)77 log(lambda)78 log(lambda)79 log(lambda)80
##    4.74084895    5.06110854    4.88028055    3.81371274    4.55673371
## log(lambda)81 log(lambda)82 log(lambda)83 log(lambda)84         beta1
##    4.42799130    4.72598453    4.81001738    5.02127589   -0.74315497
##         beta2         beta3        sigma2
##    0.73439419    0.92434529    0.02137983
```

```
colMeans(chain.combine2) #Incorrect augmentation.
```

```
##  log(lambda)1  log(lambda)2  log(lambda)3  log(lambda)4  log(lambda)5
##    4.64830819    3.74868984    4.18074941    4.66716890    5.22649766
##  log(lambda)6  log(lambda)7  log(lambda)8  log(lambda)9 log(lambda)10
##    4.61890538    4.73921882    4.76545546    3.59900949    4.41195699
## log(lambda)11 log(lambda)12 log(lambda)13 log(lambda)14 log(lambda)15
##    4.55829197    4.29484979    4.71398807    4.64829798    4.67600893
## log(lambda)16 log(lambda)17 log(lambda)18 log(lambda)19 log(lambda)20
##    3.87980752    4.38627587    4.57843908    4.97148862    4.79075074
## log(lambda)21 log(lambda)22 log(lambda)23 log(lambda)24 log(lambda)25
##    4.94459543    4.89388328    3.64773164    4.53834862    4.61890415
## log(lambda)26 log(lambda)27 log(lambda)28 log(lambda)29 log(lambda)30
##    4.31089613    5.13847053    5.15543578    4.87771453    3.31221129
## log(lambda)31 log(lambda)32 log(lambda)33 log(lambda)34 log(lambda)35
##    4.33635045    4.56989562    4.96562903    4.89259868    4.56906464
## log(lambda)36 log(lambda)37 log(lambda)38 log(lambda)39 log(lambda)40
##    4.56721220    3.97870487    4.43552579    4.49312179    4.72156059
## log(lambda)41 log(lambda)42 log(lambda)43 log(lambda)44 log(lambda)45
##    5.34342527    5.18289625    5.06501986    3.67505609    4.30957519
## log(lambda)46 log(lambda)47 log(lambda)48 log(lambda)49 log(lambda)50
##    4.49325884    4.82361186    4.79969163    4.58870710    4.78309246
## log(lambda)51 log(lambda)52 log(lambda)53 log(lambda)54 log(lambda)55
##    3.54151634    4.79053202    4.64827402    4.70307469    4.89995534
## log(lambda)56 log(lambda)57 log(lambda)58 log(lambda)59 log(lambda)60
##    4.98558629    4.62915483    3.97775881    4.28227969    4.22522864
## log(lambda)61 log(lambda)62 log(lambda)63 log(lambda)64 log(lambda)65
##    4.90814553    4.90027811    4.87838916    4.83175854    4.18056689
## log(lambda)66 log(lambda)67 log(lambda)68 log(lambda)69 log(lambda)70
##    4.42495488    4.97929385    4.71293766    4.50444281    4.85492723
## log(lambda)71 log(lambda)72 log(lambda)73 log(lambda)74 log(lambda)75
##    4.79842043    3.77153776    4.44740772    4.32431533    4.87800002
```

```
## log(lambda)76 log(lambda)77 log(lambda)78 log(lambda)79 log(lambda)80
##    4.64834818    5.07204062    4.87840536    3.72355136    4.62918765
## log(lambda)81 log(lambda)82 log(lambda)83 log(lambda)84         beta1
##    4.48226092    4.72171567    4.78233514    5.03217129   -0.85572458
##         beta2         beta3        sigma2
##    0.74652624    0.94240677    0.04574563
```

```
#Posterior standard deviations

apply(chain.combine1,2,FUN=sd ) #Correct augmentation
```

```
##  log(lambda)1  log(lambda)2  log(lambda)3  log(lambda)4  log(lambda)5
##   0.081597352   0.108042850   0.091017168   0.082235027   0.067059568
##  log(lambda)6  log(lambda)7  log(lambda)8  log(lambda)9 log(lambda)10
##   0.081673897   0.079720058   0.078824518   0.108136152   0.087046431
## log(lambda)11 log(lambda)12 log(lambda)13 log(lambda)14 log(lambda)15
##   0.084651155   0.091445374   0.079161212   0.080407470   0.079226012
## log(lambda)16 log(lambda)17 log(lambda)18 log(lambda)19 log(lambda)20
##   0.104130907   0.087563833   0.082480631   0.073700517   0.078654765
## log(lambda)21 log(lambda)22 log(lambda)23 log(lambda)24 log(lambda)25
##   0.073810796   0.075897288   0.111703590   0.088142945   0.082010456
## log(lambda)26 log(lambda)27 log(lambda)28 log(lambda)29 log(lambda)30
##   0.090227841   0.071174095   0.068678605   0.076508251   0.117622586
## log(lambda)31 log(lambda)32 log(lambda)33 log(lambda)34 log(lambda)35
##   0.089352800   0.083396648   0.072144258   0.074348073   0.084211934
## log(lambda)36 log(lambda)37 log(lambda)38 log(lambda)39 log(lambda)40
##   0.081634906   0.106199158   0.087415371   0.085370146   0.078938425
## log(lambda)41 log(lambda)42 log(lambda)43 log(lambda)44 log(lambda)45
##   0.064762275   0.068563661   0.073411920   0.111583110   0.090966272
## log(lambda)46 log(lambda)47 log(lambda)48 log(lambda)49 log(lambda)50
##   0.088097703   0.076968187   0.076102085   0.082483386   0.077741065
## log(lambda)51 log(lambda)52 log(lambda)53 log(lambda)54 log(lambda)55
##   0.110439364   0.080023018   0.082998565   0.077022140   0.074479535
## log(lambda)56 log(lambda)57 log(lambda)58 log(lambda)59 log(lambda)60
##   0.073751340   0.081494478   0.104293332   0.090142519   0.092338845
## log(lambda)61 log(lambda)62 log(lambda)63 log(lambda)64 log(lambda)65
##   0.074388797   0.073697265   0.075996239   0.077960760   0.100210139
## log(lambda)66 log(lambda)67 log(lambda)68 log(lambda)69 log(lambda)70
##   0.088901932   0.074509508   0.078997755   0.084525185   0.074828569
## log(lambda)71 log(lambda)72 log(lambda)73 log(lambda)74 log(lambda)75
##   0.076721293   0.108643652   0.087138666   0.089288529   0.075117851
## log(lambda)76 log(lambda)77 log(lambda)78 log(lambda)79 log(lambda)80
##   0.079000861   0.070159128   0.076127493   0.105839662   0.084361095
## log(lambda)81 log(lambda)82 log(lambda)83 log(lambda)84         beta1
##   0.088139910   0.078967411   0.076896161   0.071131256   0.349128020
##         beta2         beta3        sigma2
##   0.042461918   0.063753675   0.005110326
```

```
apply(chain.combine2,2,FUN=sd ) #Incorrect augmentation
```

```
##  log(lambda)1  log(lambda)2  log(lambda)3  log(lambda)4  log(lambda)5
##   0.097691428   0.153082635   0.123677128   0.097279519   0.073274496
##  log(lambda)6  log(lambda)7  log(lambda)8  log(lambda)9 log(lambda)10
```

```
##   0.099240942   0.093596647   0.091862250   0.164789639   0.110593854
## log(lambda)11 log(lambda)12 log(lambda)13 log(lambda)14 log(lambda)15
##   0.102229587   0.117100641   0.095126677   0.097325043   0.097321026
## log(lambda)16 log(lambda)17 log(lambda)18 log(lambda)19 log(lambda)20
##   0.144352765   0.110842852   0.100967167   0.083445627   0.091233407
## log(lambda)21 log(lambda)22 log(lambda)23 log(lambda)24 log(lambda)25
##   0.084937075   0.086341684   0.160447692   0.103490611   0.099309287
## log(lambda)26 log(lambda)27 log(lambda)28 log(lambda)29 log(lambda)30
##   0.115313969   0.076470751   0.076011036   0.087194749   0.189147009
## log(lambda)31 log(lambda)32 log(lambda)33 log(lambda)34 log(lambda)35
##   0.115223202   0.101302141   0.083865711   0.086468877   0.102046970
## log(lambda)36 log(lambda)37 log(lambda)38 log(lambda)39 log(lambda)40
##   0.101841240   0.136598507   0.108721582   0.105702164   0.094782440
## log(lambda)41 log(lambda)42 log(lambda)43 log(lambda)44 log(lambda)45
##   0.069083119   0.074416465   0.079048592   0.160346228   0.116396257
## log(lambda)46 log(lambda)47 log(lambda)48 log(lambda)49 log(lambda)50
##   0.105896942   0.089819169   0.091162551   0.100617463   0.090698378
## log(lambda)51 log(lambda)52 log(lambda)53 log(lambda)54 log(lambda)55
##   0.169349646   0.091093629   0.097890745   0.095385040   0.086380044
## log(lambda)56 log(lambda)57 log(lambda)58 log(lambda)59 log(lambda)60
##   0.082449045   0.098683598   0.137457225   0.117611948   0.120448146
## log(lambda)61 log(lambda)62 log(lambda)63 log(lambda)64 log(lambda)65
##   0.085645983   0.086338880   0.086950698   0.088933157   0.124099122
## log(lambda)66 log(lambda)67 log(lambda)68 log(lambda)69 log(lambda)70
##   0.109290874   0.082657539   0.094811613   0.104829173   0.088647633
## log(lambda)71 log(lambda)72 log(lambda)73 log(lambda)74 log(lambda)75
##   0.090468091   0.150412284   0.108408815   0.114663516   0.086964999
## log(lambda)76 log(lambda)77 log(lambda)78 log(lambda)79 log(lambda)80
##   0.097820256   0.079081410   0.087107992   0.154925978   0.098431752
## log(lambda)81 log(lambda)82 log(lambda)83 log(lambda)84         beta1
##   0.105929477   0.094810438   0.091993278   0.080687124   0.452013310
##         beta2         beta3        sigma2
##   0.054901985   0.083065373   0.009018646
```

```
#95 % Credible intervals.
apply(chain.combine1,2,FUN=function(x){ quantile(x,c(0.025,0.975))} ) #Correct augmentation
```

```
##       log(lambda)1 log(lambda)2 log(lambda)3 log(lambda)4 log(lambda)5
## 2.5%      4.515128     3.561404     4.139866     4.458312     5.049255
## 97.5%     4.835868     3.987254     4.501112     4.781961     5.311822
##       log(lambda)6 log(lambda)7 log(lambda)8 log(lambda)9 log(lambda)10
## 2.5%      4.540779     4.592141     4.613504     3.559629      4.280339
## 97.5%     4.858450     4.904959     4.924545     3.983005      4.622720
##       log(lambda)11 log(lambda)12 log(lambda)13 log(lambda)14 log(lambda)15
## 2.5%       4.360704      4.280055      4.582230      4.534536      4.560751
## 97.5%      4.694959      4.638566      4.892777      4.850078      4.874157
##       log(lambda)16 log(lambda)17 log(lambda)18 log(lambda)19 log(lambda)20
## 2.5%       3.673679      4.195158      4.456010      4.778207      4.590075
## 97.5%      4.082565      4.542426      4.782595      5.064682      4.898917
##       log(lambda)21 log(lambda)22 log(lambda)23 log(lambda)24 log(lambda)25
## 2.5%       4.783934      4.720537      3.508631      4.300329      4.430775
## 97.5%      5.071489      5.019050      3.943178      4.646617      4.751589
##       log(lambda)26 log(lambda)27 log(lambda)28 log(lambda)29 log(lambda)30
## 2.5%       4.317182      4.934371      4.995135      4.708249       3.35103
```

```
## 97.5%      4.668056      5.209724      5.267056      5.007915       3.81069
##       log(lambda)31 log(lambda)32 log(lambda)33 log(lambda)34 log(lambda)35
## 2.5%       4.197019      4.393817      4.799332      4.763792      4.458629
## 97.5%      4.551735      4.724437      5.081281      5.051802      4.790262
##       log(lambda)36 log(lambda)37 log(lambda)38 log(lambda)39 log(lambda)40
## 2.5%       4.485471      3.675228      4.223284      4.330907      4.597040
## 97.5%      4.803736      4.087960      4.567916      4.667489      4.907181
##       log(lambda)41 log(lambda)42 log(lambda)43 log(lambda)44 log(lambda)45
## 2.5%       5.167097      5.015529      4.864776      3.457221      4.113782
## 97.5%      5.419809      5.281106      5.152643      3.899879      4.472659
##       log(lambda)46 log(lambda)47 log(lambda)48 log(lambda)49 log(lambda)50
## 2.5%       4.253800      4.666789      4.668869      4.510087      4.620633
## 97.5%      4.596435      4.965279      4.967108      4.830456      4.925172
##       log(lambda)51 log(lambda)52 log(lambda)53 log(lambda)54 log(lambda)55
## 2.5%       3.466038      4.552737      4.427723      4.630831      4.761575
## 97.5%      3.891907      4.867393      4.753262      4.932946      5.055921
##       log(lambda)56 log(lambda)57 log(lambda)58 log(lambda)59 log(lambda)60
## 2.5%       4.775235      4.514972      3.717926      4.179296      4.146071
## 97.5%      5.063685      4.835138      4.138568      4.531251      4.508871
##       log(lambda)61 log(lambda)62 log(lambda)63 log(lambda)64 log(lambda)65
## 2.5%       4.748619      4.739553      4.712855      4.652903      3.868563
## 97.5%      5.039110      5.027257      5.010381      4.955857      4.261361
##       log(lambda)66 log(lambda)67 log(lambda)68 log(lambda)69 log(lambda)70
## 2.5%       4.234600      4.756163      4.587617      4.403138      4.710039
## 97.5%      4.579448      5.047100      4.895425      4.729827      5.005105
##       log(lambda)71 log(lambda)72 log(lambda)73 log(lambda)74 log(lambda)75
## 2.5%       4.666043      3.526651      4.247027      4.207791      4.732323
## 97.5%      4.966881      3.950138      4.585781      4.554508      5.027059
##       log(lambda)76 log(lambda)77 log(lambda)78 log(lambda)79 log(lambda)80
## 2.5%       4.581083      4.920347      4.728951      3.602599      4.392194
## 97.5%      4.893339      5.196615      5.024935      4.018347      4.724224
##       log(lambda)81 log(lambda)82 log(lambda)83 log(lambda)84       beta1
## 2.5%       4.250883      4.568192      4.658192      4.881080 -1.43424867
## 97.5%      4.598821      4.880899      4.959835      5.159943 -0.06654378
##           beta2    beta3     sigma2
## 2.5%  0.6518329 0.800495 0.01313913
## 97.5% 0.8181780 1.049830 0.03292012
```

```
apply(chain.combine2,2,FUN=function(x){ quantile(x,c(0.025,0.975))} ) #Incorrect augmentation
```

```
##       log(lambda)1 log(lambda)2 log(lambda)3 log(lambda)4 log(lambda)5
## 2.5%      4.452952     3.438730     3.930953     4.471891     5.079579
## 97.5%     4.835439     4.036423     4.417024     4.852294     5.368660
##       log(lambda)6 log(lambda)7 log(lambda)8 log(lambda)9 log(lambda)10
## 2.5%      4.418104     4.551836     4.579777     3.259893      4.191968
## 97.5%     4.805963     4.920572     4.940805     3.908053      4.624603
##       log(lambda)11 log(lambda)12 log(lambda)13 log(lambda)14 log(lambda)15
## 2.5%       4.352469      4.061401      4.523952      4.450609      4.481594
## 97.5%      4.753034      4.519883      4.896173      4.832947      4.862113
##       log(lambda)16 log(lambda)17 log(lambda)18 log(lambda)19 log(lambda)20
## 2.5%       3.585451      4.162620      4.374658      4.805149      4.609323
## 97.5%      4.151365      4.597409      4.770727      5.131990      4.966872
##       log(lambda)21 log(lambda)22 log(lambda)23 log(lambda)24 log(lambda)25
## 2.5%       4.776041      4.722886      3.323388      4.330515      4.419794
```

```
## 97.5%      5.107874      5.059862      3.949544      4.736493      4.809916
##       log(lambda)26 log(lambda)27 log(lambda)28 log(lambda)29 log(lambda)30
## 2.5%       4.079131      4.985841      5.002129      4.704080      2.929157
## 97.5%      4.530016      5.286012      5.301658      5.046701      3.666688
##       log(lambda)31 log(lambda)32 log(lambda)33 log(lambda)34 log(lambda)35
## 2.5%       4.104034      4.366236      4.797257      4.722174      4.362886
## 97.5%      4.554333      4.761905      5.126645      5.058093      4.763293
##       log(lambda)36 log(lambda)37 log(lambda)38 log(lambda)39 log(lambda)40
## 2.5%       4.363127      3.703479      4.218171      4.281289      4.531564
## 97.5%      4.760366      4.236643      4.643094      4.695330      4.903377
##       log(lambda)41 log(lambda)42 log(lambda)43 log(lambda)44 log(lambda)45
## 2.5%       5.205231      5.036141      4.906672      3.349825      4.075265
## 97.5%      5.475598      5.325985      5.217237      3.977363      4.533365
##       log(lambda)46 log(lambda)47 log(lambda)48 log(lambda)49 log(lambda)50
## 2.5%       4.280907      4.643934      4.618004      4.385916      4.599805
## 97.5%      4.694785      4.996520      4.972957      4.780844      4.955306
##       log(lambda)51 log(lambda)52 log(lambda)53 log(lambda)54 log(lambda)55
## 2.5%       3.195978      4.606120      4.450846      4.511302      4.726779
## 97.5%      3.861666      4.965068      4.834507      4.885664      5.066066
##       log(lambda)56 log(lambda)57 log(lambda)58 log(lambda)59 log(lambda)60
## 2.5%       4.821541      4.431353      3.701213      4.044349      3.980062
## 97.5%      5.143668      4.819160      4.236984      4.506714      4.453497
##       log(lambda)61 log(lambda)62 log(lambda)63 log(lambda)64 log(lambda)65
## 2.5%       4.736634      4.727244      4.705980      4.654653      3.931286
## 97.5%      5.073579      5.065644      5.045928      5.003055      4.415737
##       log(lambda)66 log(lambda)67 log(lambda)68 log(lambda)69 log(lambda)70
## 2.5%       4.206491      4.814846      4.522424      4.293454      4.675074
## 97.5%      4.634685      5.139074      4.894555      4.705142      5.022803
##       log(lambda)71 log(lambda)72 log(lambda)73 log(lambda)74 log(lambda)75
## 2.5%       4.616767      3.468584      4.230307      4.095083      4.704916
## 97.5%      4.972035      4.056316      4.654033      4.544649      5.046694
##       log(lambda)76 log(lambda)77 log(lambda)78 log(lambda)79 log(lambda)80
## 2.5%       4.451105      4.915212      4.703724      3.407703      4.432138
## 97.5%      4.834728      5.223391      5.042926      4.016764      4.816874
##       log(lambda)81 log(lambda)82 log(lambda)83 log(lambda)84       beta1
## 2.5%       4.269332      4.531461      4.599617      4.870575 -1.73679708
## 97.5%      4.682477      4.902562      4.959187      5.188484  0.02939825
##           beta2     beta3     sigma2
## 2.5%  0.6382032 0.7805438 0.03076600
## 97.5% 0.8535832 1.1052255 0.06603682
```

Stan code for implementing the Poisson-lognormal model This code is avaiable on Canvas as `Poisson-lognormal.stan`.

```
//
// This Stan program defines a Poisson-lognormal regression
//
// Learn more about model development with Stan at:
//
//    http://mc-stan.org/users/interfaces/rstan.html
//    https://github.com/stan-dev/rstan/wiki/RStan-Getting-Started
//

data {
  int<lower=0> n;   //number of observations
  int<lower=0> P;   //number of parameters
```

```
  int<lower=0> y[n];          //response vector
  matrix[n,P] X;     //design matrix (includes intercept)
}


// The parameters accepted by the model.
// accepts two sets of parameters 'beta', and 'sigma'.
parameters {
  vector[P] beta; //vector of fixed effects of length P.
  vector[n] llambda; //vector of link function.
  real<lower=0> tau; //residual precision
}

transformed parameters {
  real<lower=0> sigma;
  sigma = pow(tau, -0.5); //residual standard deviation
}

// The model to be estimated. We model the output
// 'llambda' to be normal with mean X*beta and variance sigma.
// We assume y is Poisson with parameter exp(llambda)
// and  a vague gamma prior for tau = 1/sigma^2.
model {
  y ~ poisson(exp(llambda));  //likelihood
  llambda ~ normal(X*beta,sigma); //augmented variable
  tau ~ gamma(0.001,0.001); //prior
}
```

```
#fitting model using stan. You may need to install Rtools to get this to work. The stan file is availab
#from canvas as Poisson-lognormal.stan. This should go in the file.choose() position.
library(rstan)
```

```
## Warning: package 'rstan' was built under R version 4.3.1
```

```
## Loading required package: StanHeaders
```

```
## Warning: package 'StanHeaders' was built under R version 4.3.1
```

```
##
## rstan version 2.32.5 (Stan version 2.32.2)
```

```
## For execution on a local, multicore CPU with excess RAM we recommend calling
## options(mc.cores = parallel::detectCores()).
## To avoid recompilation of unchanged Stan programs, we recommend calling
## rstan_options(auto_write = TRUE)
## For within-chain threading using `reduce_sum()` or `map_rect()` Stan functions,
## change `threads_per_chain` option:
## rstan_options(threads_per_chain = 1)
```

```
##
## Attaching package: 'rstan'
```

```
## The following object is masked from 'package:coda':
##
##     traceplot
```

```
P<-dim(X)[2]
#Formatting inputs.
pois.reg<-stan(file='./Poisson-lognormal.stan',data=c('X','y','n','P'),iter=2000,chains=4)
```

```
##
## SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 1).
## Chain 1:
## Chain 1: Gradient evaluation took 4.6e-05 seconds
## Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 0.46 seconds.
## Chain 1: Adjust your expectations accordingly!
## Chain 1:
## Chain 1:
## Chain 1: Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 1: Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 1: Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 1: Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 1: Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 1: Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 1: Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 1: Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 1: Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 1: Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 1: Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 1: Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 1:
## Chain 1:  Elapsed Time: 0.522 seconds (Warm-up)
## Chain 1:                0.572 seconds (Sampling)
## Chain 1:                1.094 seconds (Total)
## Chain 1:
##
## SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 2).
## Chain 2:
## Chain 2: Gradient evaluation took 5e-06 seconds
## Chain 2: 1000 transitions using 10 leapfrog steps per transition would take 0.05 seconds.
## Chain 2: Adjust your expectations accordingly!
## Chain 2:
## Chain 2:
## Chain 2: Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 2: Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 2: Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 2: Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 2: Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 2: Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 2: Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 2: Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 2: Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 2: Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 2: Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 2: Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 2:
## Chain 2:  Elapsed Time: 0.53 seconds (Warm-up)
## Chain 2:                0.574 seconds (Sampling)
## Chain 2:                1.104 seconds (Total)
```

```
## Chain 2:
##
## SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 3).
## Chain 3:
## Chain 3: Gradient evaluation took 5e-06 seconds
## Chain 3: 1000 transitions using 10 leapfrog steps per transition would take 0.05 seconds.
## Chain 3: Adjust your expectations accordingly!
## Chain 3:
## Chain 3:
## Chain 3: Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 3: Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 3: Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 3: Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 3: Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 3: Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 3: Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 3: Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 3: Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 3: Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 3: Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 3: Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 3:
## Chain 3:  Elapsed Time: 0.55 seconds (Warm-up)
## Chain 3:                0.575 seconds (Sampling)
## Chain 3:                1.125 seconds (Total)
## Chain 3:
##
## SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 4).
## Chain 4:
## Chain 4: Gradient evaluation took 9e-06 seconds
## Chain 4: 1000 transitions using 10 leapfrog steps per transition would take 0.09 seconds.
## Chain 4: Adjust your expectations accordingly!
## Chain 4:
## Chain 4:
## Chain 4: Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 4: Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 4: Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 4: Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 4: Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 4: Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 4: Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 4: Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 4: Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 4: Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 4: Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 4: Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 4:
## Chain 4:  Elapsed Time: 0.545 seconds (Warm-up)
## Chain 4:                0.575 seconds (Sampling)
## Chain 4:                1.12 seconds (Total)
## Chain 4:
```

```
print(pois.reg)
```

```
## Inference for Stan model: anon_model.
## 4 chains, each with iter=2000; warmup=1000; thin=1;
## post-warmup draws per chain=1000, total post-warmup draws=4000.
##
##                mean se_mean    sd     2.5%      25%      50%      75%    97.5%
## beta[1]       -0.73    0.01  0.35    -1.44    -0.96    -0.72    -0.49    -0.07
## beta[2]        0.73    0.00  0.04     0.65     0.70     0.73     0.76     0.82
## beta[3]        0.92    0.00  0.06     0.80     0.88     0.92     0.96     1.05
## llambda[1]     4.68    0.00  0.08     4.51     4.62     4.68     4.73     4.83
## llambda[2]     3.77    0.00  0.10     3.57     3.70     3.77     3.84     3.97
## llambda[3]     4.33    0.00  0.09     4.14     4.27     4.33     4.39     4.50
## llambda[4]     4.62    0.00  0.08     4.46     4.56     4.62     4.68     4.78
## llambda[5]     5.18    0.00  0.07     5.06     5.14     5.18     5.23     5.31
## llambda[6]     4.70    0.00  0.08     4.55     4.65     4.71     4.76     4.85
## llambda[7]     4.75    0.00  0.08     4.60     4.70     4.75     4.80     4.90
## llambda[8]     4.77    0.00  0.08     4.61     4.72     4.77     4.83     4.92
## llambda[9]     3.78    0.00  0.11     3.56     3.71     3.78     3.85     3.99
## llambda[10]    4.45    0.00  0.09     4.28     4.40     4.45     4.51     4.62
## llambda[11]    4.53    0.00  0.08     4.36     4.47     4.53     4.58     4.68
## llambda[12]    4.47    0.00  0.09     4.29     4.41     4.47     4.53     4.63
## llambda[13]    4.74    0.00  0.08     4.58     4.69     4.74     4.80     4.90
## llambda[14]    4.70    0.00  0.08     4.54     4.64     4.70     4.75     4.85
## llambda[15]    4.72    0.00  0.08     4.56     4.67     4.72     4.77     4.88
## llambda[16]    3.88    0.00  0.10     3.68     3.81     3.88     3.96     4.09
## llambda[17]    4.37    0.00  0.09     4.20     4.31     4.37     4.43     4.54
## llambda[18]    4.62    0.00  0.08     4.46     4.57     4.62     4.67     4.77
## llambda[19]    4.92    0.00  0.07     4.78     4.87     4.92     4.97     5.07
## llambda[20]    4.75    0.00  0.08     4.60     4.70     4.75     4.80     4.90
## llambda[21]    4.93    0.00  0.07     4.78     4.88     4.93     4.98     5.08
## llambda[22]    4.87    0.00  0.08     4.72     4.82     4.87     4.92     5.02
## llambda[23]    3.73    0.00  0.11     3.52     3.66     3.74     3.81     3.94
## llambda[24]    4.47    0.00  0.09     4.30     4.41     4.47     4.53     4.65
## llambda[25]    4.59    0.00  0.08     4.42     4.53     4.59     4.65     4.75
## llambda[26]    4.50    0.00  0.09     4.32     4.44     4.50     4.56     4.67
## llambda[27]    5.07    0.00  0.07     4.93     5.02     5.07     5.12     5.21
## llambda[28]    5.13    0.00  0.07     5.00     5.08     5.13     5.18     5.26
## llambda[29]    4.86    0.00  0.08     4.71     4.81     4.86     4.91     5.01
## llambda[30]    3.59    0.00  0.11     3.37     3.51     3.59     3.67     3.81
## llambda[31]    4.38    0.00  0.09     4.21     4.32     4.38     4.44     4.55
## llambda[32]    4.57    0.00  0.08     4.40     4.51     4.57     4.62     4.72
## llambda[33]    4.94    0.00  0.07     4.80     4.89     4.94     4.99     5.08
## llambda[34]    4.91    0.00  0.08     4.76     4.86     4.91     4.96     5.06
## llambda[35]    4.63    0.00  0.08     4.47     4.58     4.63     4.69     4.79
## llambda[36]    4.65    0.00  0.08     4.47     4.59     4.65     4.70     4.81
## llambda[37]    3.88    0.00  0.10     3.68     3.81     3.88     3.95     4.09
## llambda[38]    4.40    0.00  0.09     4.22     4.34     4.40     4.46     4.57
## llambda[39]    4.50    0.00  0.09     4.33     4.44     4.50     4.56     4.67
## llambda[40]    4.76    0.00  0.08     4.61     4.70     4.76     4.81     4.91
## llambda[41]    5.29    0.00  0.06     5.17     5.25     5.29     5.34     5.42
## llambda[42]    5.15    0.00  0.07     5.01     5.10     5.15     5.19     5.28
## llambda[43]    5.01    0.00  0.07     4.87     4.96     5.01     5.05     5.15
## llambda[44]    3.68    0.00  0.11     3.47     3.61     3.68     3.76     3.89
## llambda[45]    4.29    0.00  0.09     4.11     4.23     4.29     4.35     4.47
## llambda[46]    4.43    0.00  0.09     4.25     4.36     4.43     4.49     4.60
```

```
## llambda[47]     4.82    0.00  0.08     4.66     4.77     4.82     4.87     4.97
## llambda[48]     4.82    0.00  0.08     4.67     4.77     4.82     4.87     4.96
## llambda[49]     4.67    0.00  0.08     4.51     4.62     4.67     4.73     4.83
## llambda[50]     4.77    0.00  0.08     4.62     4.72     4.77     4.83     4.93
## llambda[51]     3.68    0.00  0.11     3.46     3.61     3.69     3.76     3.90
## llambda[52]     4.71    0.00  0.08     4.55     4.65     4.71     4.76     4.86
## llambda[53]     4.59    0.00  0.09     4.42     4.53     4.59     4.65     4.75
## llambda[54]     4.78    0.00  0.08     4.63     4.73     4.79     4.84     4.93
## llambda[55]     4.91    0.00  0.07     4.76     4.86     4.91     4.97     5.06
## llambda[56]     4.92    0.00  0.08     4.77     4.87     4.92     4.97     5.06
## llambda[57]     4.68    0.00  0.08     4.52     4.62     4.68     4.74     4.84
## llambda[58]     3.93    0.00  0.10     3.73     3.86     3.94     4.01     4.13
## llambda[59]     4.35    0.00  0.09     4.17     4.29     4.36     4.42     4.53
## llambda[60]     4.33    0.00  0.09     4.14     4.27     4.33     4.39     4.51
## llambda[61]     4.89    0.00  0.08     4.74     4.84     4.89     4.95     5.04
## llambda[62]     4.88    0.00  0.07     4.74     4.83     4.88     4.93     5.02
## llambda[63]     4.86    0.00  0.08     4.71     4.81     4.86     4.92     5.01
## llambda[64]     4.80    0.00  0.08     4.65     4.76     4.80     4.86     4.95
## llambda[65]     4.07    0.00  0.10     3.87     4.00     4.07     4.13     4.26
## llambda[66]     4.41    0.00  0.09     4.23     4.35     4.41     4.47     4.58
## llambda[67]     4.90    0.00  0.08     4.75     4.85     4.90     4.96     5.05
## llambda[68]     4.74    0.00  0.08     4.59     4.69     4.75     4.80     4.89
## llambda[69]     4.57    0.00  0.09     4.40     4.51     4.57     4.63     4.73
## llambda[70]     4.86    0.00  0.08     4.71     4.81     4.86     4.91     5.01
## llambda[71]     4.82    0.00  0.07     4.66     4.77     4.82     4.87     4.96
## llambda[72]     3.74    0.00  0.11     3.52     3.67     3.74     3.82     3.96
## llambda[73]     4.42    0.00  0.09     4.24     4.36     4.42     4.48     4.59
## llambda[74]     4.38    0.00  0.09     4.21     4.32     4.38     4.44     4.56
## llambda[75]     4.88    0.00  0.07     4.74     4.84     4.89     4.93     5.03
## llambda[76]     4.74    0.00  0.08     4.59     4.69     4.75     4.80     4.89
## llambda[77]     5.06    0.00  0.07     4.92     5.01     5.06     5.11     5.20
## llambda[78]     4.88    0.00  0.08     4.73     4.83     4.88     4.93     5.03
## llambda[79]     3.82    0.00  0.11     3.60     3.74     3.82     3.89     4.02
## llambda[80]     4.56    0.00  0.09     4.39     4.50     4.56     4.61     4.73
## llambda[81]     4.43    0.00  0.09     4.26     4.37     4.43     4.49     4.60
## llambda[82]     4.73    0.00  0.08     4.57     4.67     4.73     4.78     4.88
## llambda[83]     4.81    0.00  0.08     4.66     4.76     4.81     4.86     4.95
## llambda[84]     5.02    0.00  0.07     4.88     4.97     5.02     5.07     5.16
## tau            49.87    0.29 12.14    31.23    41.06    48.23    56.73    78.39
## sigma           0.14    0.00  0.02     0.11     0.13     0.14     0.16     0.18
## lp__        32970.81    0.23  7.57 32954.99 32966.01 32971.00 32975.94 32984.72
##             n_eff Rhat
## beta[1]      1677    1
## beta[2]      1643    1
## beta[3]      1778    1
## llambda[1]   7796    1
## llambda[2]   5373    1
## llambda[3]   6633    1
## llambda[4]   7576    1
## llambda[5]   7231    1
## llambda[6]   5505    1
## llambda[7]   7806    1
## llambda[8]   7357    1
## llambda[9]   6158    1
```

```
## llambda[10]  7541    1
## llambda[11]  7605    1
## llambda[12]  5045    1
## llambda[13]  6916    1
## llambda[14]  8292    1
## llambda[15]  6920    1
## llambda[16]  6089    1
## llambda[17]  7528    1
## llambda[18]  8760    1
## llambda[19]  6321    1
## llambda[20]  6858    1
## llambda[21]  8204    1
## llambda[22]  6483    1
## llambda[23]  5693    1
## llambda[24]  7825    1
## llambda[25]  7889    1
## llambda[26]  5049    1
## llambda[27]  6767    1
## llambda[28]  7851    1
## llambda[29]  7612    1
## llambda[30]  4222    1
## llambda[31]  8162    1
## llambda[32]  9634    1
## llambda[33]  7356    1
## llambda[34]  6834    1
## llambda[35]  6875    1
## llambda[36]  6783    1
## llambda[37]  5387    1
## llambda[38]  6435    1
## llambda[39]  8611    1
## llambda[40]  7761    1
## llambda[41]  7289    1
## llambda[42]  8293    1
## llambda[43]  7574    1
## llambda[44]  4941    1
## llambda[45]  7110    1
## llambda[46]  7362    1
## llambda[47]  7401    1
## llambda[48]  7958    1
## llambda[49]  6799    1
## llambda[50]  8579    1
## llambda[51]  5695    1
## llambda[52]  5859    1
## llambda[53]  8976    1
## llambda[54]  6896    1
## llambda[55]  7208    1
## llambda[56]  7240    1
## llambda[57]  6951    1
## llambda[58]  5962    1
## llambda[59]  6967    1
## llambda[60]  8279    1
## llambda[61]  8140    1
## llambda[62]  7321    1
## llambda[63]  9757    1
```

```
## llambda[64]  7463    1
## llambda[65]  6912    1
## llambda[66]  8713    1
## llambda[67]  6691    1
## llambda[68]  8607    1
## llambda[69]  7023    1
## llambda[70]  6917    1
## llambda[71]  7836    1
## llambda[72]  5468    1
## llambda[73]  7607    1
## llambda[74]  5692    1
## llambda[75]  8842    1
## llambda[76]  5890    1
## llambda[77]  7444    1
## llambda[78]  7237    1
## llambda[79]  6726    1
## llambda[80]  6641    1
## llambda[81]  6466    1
## llambda[82]  7234    1
## llambda[83]  7499    1
## llambda[84]  8019    1
## tau          1743    1
## sigma        1820    1
## lp__         1087    1
##
## Samples were drawn using NUTS(diag_e) at Thu Sep 12 00:44:01 2024.
## For each parameter, n_eff is a crude measure of effective sample size,
## and Rhat is the potential scale reduction factor on split chains (at
## convergence, Rhat=1).
```

```
#Stan.
myresults<-extract(pois.reg)
chain.stan <- cbind(myresults$llambda,myresults$beta,myresults$sigma^2)
#Finding posterior means

colMeans(chain.stan)     #Stan.
```

```
##  [1]  4.67742738  3.77188629  4.32725146  4.62181625  5.18205714  4.70451494
##  [7]  4.75247004  4.77335139  3.77935459  4.45290806  4.52712653  4.46694949
## [13]  4.74065267  4.69693192  4.72016056  3.88493190  4.37021769  4.62020326
## [19]  4.92220802  4.74701648  4.93182199  4.87196778  3.73473916  4.47325792
## [25]  4.59024845  4.49813955  5.07146938  5.13052950  4.85985782  3.58970109
## [31]  4.37965227  4.56602588  4.93971433  4.90957786  4.63129026  4.64678069
## [37]  3.88169233  4.39678685  4.50204205  4.75640027  5.29384761  5.14789817
## [43]  5.00681706  3.68233842  4.29030192  4.42551447  4.81809234  4.81814847
## [49]  4.67369403  4.77225949  3.68384188  4.70743297  4.58973810  4.78383597
## [55]  4.91302638  4.91877319  4.67867200  3.93458105  4.35420172  4.33005247
## [61]  4.89260195  4.88195248  4.86451184  4.80454820  4.06574151  4.40882737
## [67]  4.90150272  4.74471505  4.57055403  4.86064120  4.81700918  3.74425010
## [73]  4.41931451  4.38385737  4.88463724  4.74343738  5.05961150  4.87996171
## [79]  3.81710811  4.55663889  4.42769428  4.72570828  4.81064827  5.02076692
## [85] -0.73094324  0.73333554  0.92199572  0.02120312
```

```r
#Posterior standard deviations

apply(chain.stan,2,FUN=sd ) #Stan
```

```
##  [1] 0.081682647 0.104500251 0.091215762 0.082792633 0.065005359 0.076980413
##  [7] 0.077402989 0.079073308 0.108210204 0.086288728 0.081925821 0.086774818
## [13] 0.080383561 0.081956551 0.080981452 0.104541298 0.087363388 0.080484658
## [19] 0.072298510 0.077068796 0.073905001 0.076480423 0.107905770 0.087742783
## [25] 0.083774956 0.089332529 0.069696090 0.067778525 0.075885095 0.114845120
## [31] 0.089325664 0.082293736 0.072694060 0.075851037 0.082074905 0.084427950
## [37] 0.103710669 0.089153156 0.085909642 0.076831878 0.061949893 0.067335539
## [43] 0.070932836 0.108512066 0.090163933 0.088859016 0.077901281 0.076150784
## [49] 0.081136286 0.078729941 0.112146070 0.081013872 0.085441369 0.077238859
## [55] 0.074467762 0.075552570 0.082802243 0.104227001 0.091157326 0.092431233
## [61] 0.075903252 0.073480693 0.075418119 0.076433140 0.100125256 0.086714347
## [67] 0.077888702 0.078106570 0.086275113 0.075747748 0.074698031 0.110174063
## [73] 0.087392580 0.088921973 0.074006926 0.078579086 0.072018805 0.076313777
## [79] 0.107792561 0.086092147 0.085891271 0.079208993 0.075239257 0.070095117
## [85] 0.349626685 0.042395022 0.063793973 0.004990872
```

```r
#95 % Credible intervals.
apply(chain.stan,2,FUN=function(x){ quantile(x,c(0.025,0.975))} ) #Stan
```

```
##           [,1]     [,2]     [,3]    [,4]     [,5]     [,6]     [,7]     [,8]
## 2.5%  4.512330 3.569716 4.142989 4.45954 5.057192 4.549917 4.595074 4.614374
## 97.5% 4.834604 3.970757 4.496474 4.78270 5.306588 4.854651 4.902506 4.923963
##           [,9]    [,10]    [,11]    [,12]    [,13]    [,14]    [,15]    [,16]
## 2.5%  3.562963 4.282935 4.364038 4.290361 4.579495 4.535437 4.556217 3.680361
## 97.5% 3.991894 4.622766 4.683685 4.630371 4.896578 4.854144 4.877241 4.086307
##          [,17]    [,18]    [,19]    [,20]    [,21]    [,22]    [,23]    [,24]
## 2.5%  4.196632 4.456954 4.777801 4.597897 4.781142 4.722837 3.522227 4.301071
## 97.5% 4.541188 4.772695 5.066081 4.897923 5.080282 5.019010 3.940127 4.645988
##          [,25]    [,26]    [,27]    [,28]    [,29]    [,30]    [,31]    [,32]
## 2.5%  4.424542 4.320647 4.933390 4.997806 4.707283 3.366704 4.206204 4.404155
## 97.5% 4.754552 4.672660 5.205514 5.261194 5.011181 3.806128 4.547608 4.722568
##          [,33]    [,34]    [,35]    [,36]    [,37]    [,38]    [,39]    [,40]
## 2.5%  4.796758 4.760069 4.467517 4.472050 3.682108 4.220646 4.331320 4.605599
## 97.5% 5.080637 5.057707 4.785731 4.811397 4.089290 4.568368 4.665595 4.906407
##          [,41]    [,42]    [,43]    [,44]    [,45]    [,46]    [,47]    [,48]
## 2.5%  5.173301 5.009707 4.872708 3.465797 4.106781 4.251864 4.662214 4.666945
## 97.5% 5.415132 5.277594 5.147174 3.888778 4.467190 4.596441 4.969322 4.962691
##          [,49]    [,50]    [,51]    [,52]    [,53]    [,54]    [,55]    [,56]
## 2.5%  4.512295 4.616478 3.463943 4.545438 4.421298 4.634217 4.762596 4.770254
## 97.5% 4.829928 4.925989 3.897447 4.863379 4.754612 4.931499 5.056035 5.063976
##          [,57]    [,58]    [,59]    [,60]    [,61]    [,62]    [,63]    [,64]
## 2.5%  4.515693 3.729981 4.172815 4.142795 4.742438 4.738578 4.714684 4.648787
## 97.5% 4.837980 4.131843 4.529169 4.508086 5.039798 5.022598 5.008304 4.951287
##          [,65]    [,66]    [,67]    [,68]    [,69]    [,70]    [,71]    [,72]
## 2.5%  3.873242 4.232750 4.749450 4.587312 4.398666 4.711925 4.664967 3.524872
## 97.5% 4.258515 4.579248 5.051474 4.894992 4.734355 5.008124 4.962137 3.961159
##          [,73]    [,74]    [,75]    [,76]    [,77]    [,78]    [,79]    [,80]
## 2.5%  4.244128 4.208409 4.740824 4.588297 4.920258 4.725149 3.603490 4.387663
```

```
## 97.5% 4.590628 4.555327 5.025086 4.893967 5.195850 5.026405 4.024975 4.726854
##          [,81]    [,82]    [,83]    [,84]       [,85]    [,86]    [,87]
## 2.5%  4.261000 4.572976 4.662613 4.884608 -1.43770682 0.654033 0.800155
## 97.5% 4.597803 4.879852 4.954448 5.158262 -0.06752384 0.816795 1.053213
##           [,88]
## 2.5%  0.01275594
## 97.5% 0.03202284
```