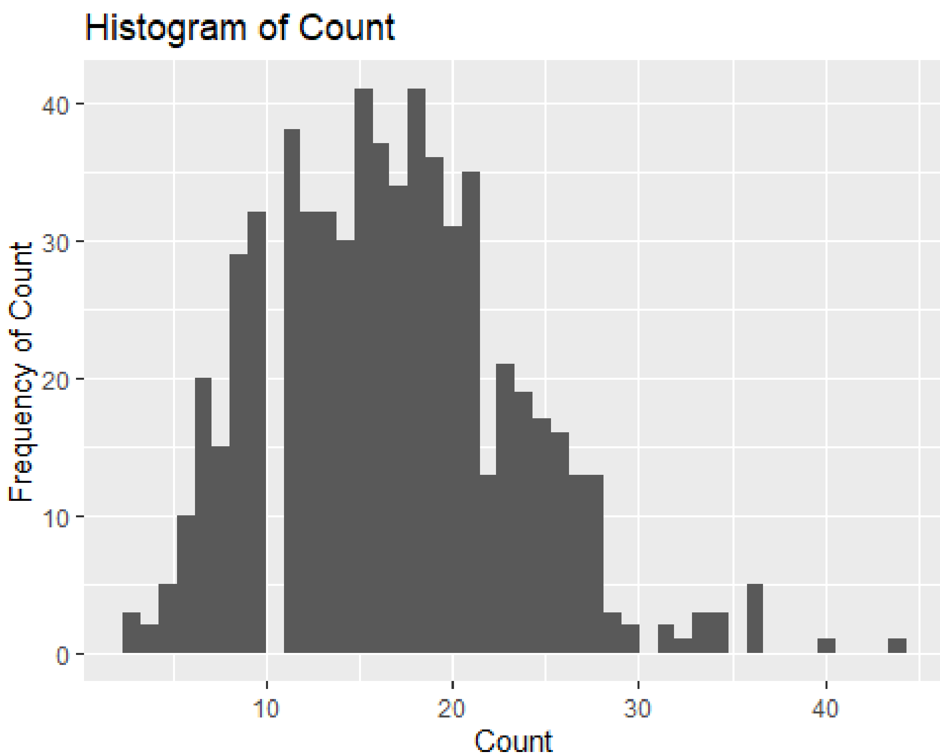
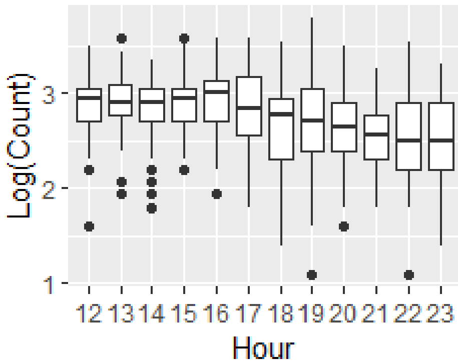
Analyzing 9-1-1 Call Data using Bayesian Regression

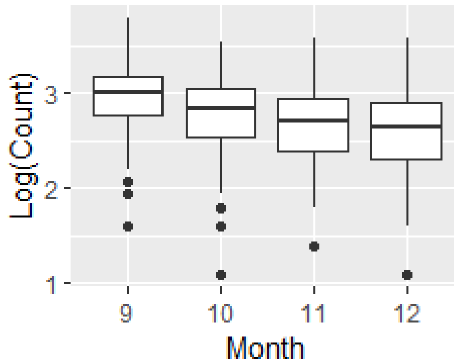
This is an analysis of Poisson and negative binomial regression models using both Bayesian and frequentist methods. The data is aggregated data from a Whitcom 9-1-1 public records request as well as additional information regarding home football games for Washington State University. Of the five columns in the data set (Count, Date, Hour, Month, Game.Day), all but Date are of interest for this analysis. Using Count as the response, we will fit four models using a combination of Hour, Month, and Game.Day. All in all, there will be sixteen models evaluated.

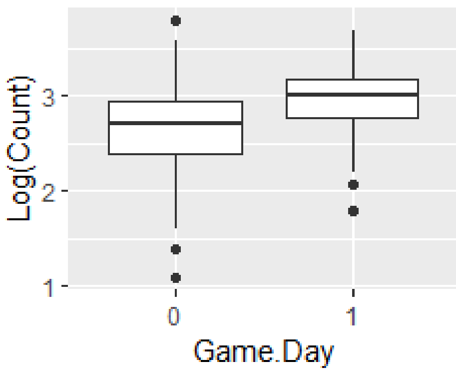
```
library(rjags)

## Loading required package: coda

## Linked to JAGS 4.3.0

## Loaded modules: basemod,bugs

library(ggplot2)
library(ggpubr)
library(MASS)

# Read in the data
data = read.csv('Data/Project_data.csv', header=TRUE)
data$Hour = as.factor(data$Hour)
data$Month = as.factor(data$Month)
data$Game.Day = as.factor(data$Game.Day)

# Histogram of Count
ggplot(data, aes(x=Count))+geom_histogram(bins=44)+
  ylab('Frequency of Count')+ggtitle('Histogram of Count')
```




```r
# Plot log(Count) against each predictor
ggarrange(
  ggplot(data, aes(x=Hour, y=log(Count)))+geom_boxplot()+
    xlab('Hour')+ylab('Log(Count)'),

  ggplot(data, aes(x=Month, y=log(Count)))+geom_boxplot()+
    xlab('Month')+ylab('Log(Count)'),

  ggplot(data, aes(x=Game.Day, y=log(Count)))+geom_boxplot()+
    xlab('Game.Day')+ylab('Log(Count)'))
```







It appears there may exist predictive relationships between the log of Count and each predictor.

```
# Create the response matrix
Y = data$Count

# Create the design matrix
X = model.matrix(Count~Hour+Month+Game.Day, data=data)
head(X)

##   (Intercept) Hour13 Hour14 Hour15 Hour16 Hour17 Hour18 Hour19 Hour20
Hour21
## 1           1      0      0      0      0      0      0      0      0
0
## 2           1      1      0      0      0      0      0      0      0
0
## 3           1      0      1      0      0      0      0      0      0
0
## 4           1      0      0      1      0      0      0      0      0
0
## 5           1      0      0      0      1      0      0      0      0
0
## 6           1      0      0      0      0      1      0      0      0
0
##   Hour22 Hour23 Month10 Month11 Month12 Game.Day1
## 1      0      0       0       0       0         1
## 2      0      0       0       0       0         1
```

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

set.seed(7)

# 80/20 training/testing split
train_rows = sample(nrow(data), 0.8*nrow(data))

# Matrix form
train_X = X[train_rows,]
test_X = X[-train_rows,]

train_Y = Y[train_rows]
test_Y = Y[-train_rows]

# Dataframe form
train_data = data[train_rows,]
test_data = data[-train_rows,]

# Poisson models and priors
p_jags = '
model{
  for(i in 1:n){

    # LIKELIHOODS

    # Model 1 (Count~Hour)
    Y1[i] ~ dpois(lambda1[i])
    log(lambda1[i]) <- inprod(X1[i,], beta1[])

    # Model 2 (Count~Hour+Month)
    Y2[i] ~ dpois(lambda2[i])
    log(lambda2[i]) <- inprod(X2[i,], beta2[])

    # Model 3 (Count~Hour+Game.Day)
    Y3[i] ~ dpois(lambda3[i])
    log(lambda3[i]) <- inprod(X3[i,], beta3[])

    # Model 4 (Count~Hour+Month+Game.Day)
    Y4[i] ~ dpois(lambda4[i])
    log(lambda4[i]) <- inprod(X4[i,], beta4[])
  }



  # PRIORS
```

```
  # Priors for Model 1
  for(p1 in 1:12){
    beta1[p1] ~ dnorm(0, 0.001)
  }

  # Priors for Model 2
  for(p2 in 1:15){
    beta2[p2] ~ dnorm(0, 0.001)
  }

  # Priors for Model 3
  for(p3 in 1:13){
    beta3[p3] ~ dnorm(0, 0.001)
  }

  # Priors for Model 4
  for(p4 in 1:16){
    beta4[p4] ~ dnorm(0, 0.001)
  }



  # PREDICTIONS USING THE TEST SET

    for(i in 1:m){

      # Model 1 Predictions
      log(lambda1_star[i]) <- inprod(X1_test[i,], beta1[])
        pred1[i] ~ dpois(lambda1_star[i])

        # Model 2 Predictions
        log(lambda2_star[i]) <- inprod(X2_test[i,], beta2[])
        pred2[i] ~ dpois(lambda2_star[i])

        # Model 3 Predictions
        log(lambda3_star[i]) <- inprod(X3_test[i,], beta3[])
        pred3[i] ~ dpois(lambda3_star[i])

        # Model 4 Predictions
        log(lambda4_star[i]) <- inprod(X4_test[i,], beta4[])
        pred4[i] ~ dpois(lambda4_star[i])
    }
}
'

# Data
p_data = list(n=nrow(train_X),
              m=nrow(test_X),
```

```r
              X1=train_X[,1:12],
              X2=train_X[,1:15],
              X3=train_X[,c(1:12, 16)],
              X4=train_X,

              X1_test=test_X[,1:12],
              X2_test=test_X[,1:15],
              X3_test=test_X[,c(1:12, 16)],
              X4_test=test_X,

              Y1=train_Y,
              Y2=train_Y,
              Y3=train_Y,
              Y4=train_Y)

# Initialize models
p_models = jags.model(file=textConnection(p_jags), data=p_data,
                      inits=list(.RNG.name = 'base::Wichmann-Hill',
                                 .RNG.seed =7))

## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 2032
##    Unobserved stochastic nodes: 568
##    Total graph size: 41104
##
## Initializing model

# Burn in
update(p_models, n.iter=1000)

# Number of iterations
iter = 1e4

# Sample
p_outputs = coda.samples(p_models,
                         variable.names = c('beta1', 'beta2',
                                            'beta3', 'beta4',
                                            'lambda1', 'lambda2',
                                            'lambda3', 'lambda4',
                                            'pred1', 'pred2',
                                            'pred3', 'pred4'),
                         n.iter = iter)

# Create 4 frequentist models
p1 = glm(Count~Hour, data=train_data, family='poisson')
p2 = glm(Count~Hour+Month, data=train_data, family='poisson')
p3 = glm(Count~Hour+Game.Day, data=train_data, family='poisson')
```

```r
p4 = glm(Count~Hour+Month+Game.Day, data=train_data, family='poisson')


# Extract the estimate of each Bayesian node
p_est = summary(p_outputs)$statistics[,1]

# Extract the 95% equi-tailed credible set of each node
p_quants = summary(p_outputs)$quantiles[,c(1,5)]


# Create a table of Bayesian coefficients with their credible set
p_beta1 = cbind(Estimate=p_est[1:12], p_quants[1:12,])

p_beta2 = cbind(Estimate=p_est[13:27], p_quants[13:27,])

p_beta3 = cbind(Estimate=p_est[28:40], p_quants[28:40,])

p_beta4 = cbind(p_est[41:56], p_quants[41:56,])

# Rename row names for better interpretation
rownames(p_beta1) = c('(Intercept)', 'Hour13', 'Hour14', 'Hour15', 'Hour16',
                      'Hour17', 'Hour18', 'Hour19', 'Hour20', 'Hour21',
                      'Hour22', 'Hour23')

rownames(p_beta2) = c('(Intercept)', 'Hour13', 'Hour14', 'Hour15', 'Hour16',
                      'Hour17', 'Hour18', 'Hour19', 'Hour20', 'Hour21',
                      'Hour22', 'Hour23', 'Month10', 'Month11', 'Month12')

rownames(p_beta3) = c('(Intercept)', 'Hour13', 'Hour14', 'Hour15', 'Hour16',
                      'Hour17', 'Hour18', 'Hour19', 'Hour20', 'Hour21',
                      'Hour22', 'Hour23', 'Game.Day')

rownames(p_beta4) = c('(Intercept)', 'Hour13', 'Hour14', 'Hour15', 'Hour16',
                      'Hour17', 'Hour18', 'Hour19', 'Hour20', 'Hour21',
                      'Hour22', 'Hour23', 'Month10', 'Month11', 'Month12',
                      'Game.Day')

# Print Bayesian coefficients with their credible sets
p_beta1

##                 Estimate        2.5%        97.5%
## (Intercept)  2.942620851  2.87257437  3.011388907
## Hour13      -0.004945450 -0.09936393  0.089815672
## Hour14      -0.089667594 -0.19371536  0.016635046
## Hour15       0.007390016 -0.09026404  0.106847978
## Hour16       0.024506690 -0.07092365  0.118258916
## Hour17      -0.034642500 -0.13233598  0.062635910
## Hour18      -0.204641979 -0.30987376 -0.103441382
## Hour19      -0.113044809 -0.21903480 -0.009888539
```

```
## Hour20      -0.309521262 -0.41665880 -0.201518591
## Hour21      -0.309227028 -0.41514291 -0.202240589
## Hour22      -0.292859235 -0.39787854 -0.188053597
## Hour23      -0.287710008 -0.39335266 -0.181559895
```

p_beta2

```
##                 Estimate        2.5%       97.5%
## (Intercept)  3.101231362  3.02509989  3.17309777
## Hour13      -0.008741151 -0.10313219  0.08612208
## Hour14      -0.081526226 -0.18100219  0.01945024
## Hour15       0.005592348 -0.08887491  0.09990170
## Hour16       0.018073521 -0.07385414  0.11032883
## Hour17      -0.035113494 -0.12860395  0.06213752
## Hour18      -0.215515432 -0.31607777 -0.11613118
## Hour19      -0.132089785 -0.23504108 -0.02676695
## Hour20      -0.292864970 -0.39772239 -0.18597796
## Hour21      -0.308105770 -0.40877779 -0.20742150
## Hour22      -0.296172721 -0.40058463 -0.19290069
## Hour23      -0.298484525 -0.40172863 -0.19371928
## Month10     -0.154200952 -0.21246416 -0.09624511
## Month11     -0.226754637 -0.28679946 -0.16771271
## Month12     -0.262667827 -0.31927433 -0.20613843
```

p_beta3

```
##                 Estimate        2.5%       97.5%
## (Intercept)  2.858590629  2.78648098  2.92726859
## Hour13      -0.017652015 -0.11184657  0.07809581
## Hour14      -0.086580171 -0.18965667  0.01904538
## Hour15      -0.002919066 -0.09931943  0.09697755
## Hour16       0.023880498 -0.06646064  0.11898009
## Hour17      -0.041752821 -0.13994982  0.05601537
## Hour18      -0.217268864 -0.31872138 -0.11327866
## Hour19      -0.136501112 -0.23910823 -0.03116623
## Hour20      -0.295711422 -0.40386500 -0.18755878
## Hour21      -0.317991644 -0.41968701 -0.21243387
## Hour22      -0.298609156 -0.40347264 -0.19254296
## Hour23      -0.300589512 -0.40546489 -0.19560144
## Game.Day     0.272029585  0.22749658  0.31548137
```

p_beta4

```
##                                 2.5%       97.5%
## (Intercept)  2.957794171  2.87388225  3.03579226
## Hour13      -0.014139694 -0.10448787  0.08160450
## Hour14      -0.078991353 -0.18388751  0.02625790
## Hour15      -0.003598471 -0.09814970  0.09305833
## Hour16       0.018594269 -0.07319965  0.11615175
## Hour17      -0.040167645 -0.13710844  0.06042960
## Hour18      -0.222317174 -0.32201503 -0.11918210
```

```
## Hour19      -0.145483010 -0.24914736 -0.04456522
## Hour20      -0.284629760 -0.39228944 -0.17557722
## Hour21      -0.314237260 -0.41671255 -0.20947911
## Hour22      -0.296995292 -0.39973330 -0.19112992
## Hour23      -0.297289228 -0.40295097 -0.19378172
## Month10     -0.074213358 -0.13455659 -0.01408321
## Month11     -0.184763643 -0.24516446 -0.12675892
## Month12     -0.116067482 -0.18157043 -0.05158707
## Game.Day     0.242713607  0.19243433  0.29374964
```

```r
# Print GLM coefficients with 95% confidence intervals
cbind(Estimate=coef(p1), confint(p1))
```

```
## Waiting for profiling to be done...

##                 Estimate       2.5 %      97.5 %
## (Intercept)  2.944438979  2.87665118  3.01072661
## Hour13      -0.006741599 -0.10066686  0.08723778
## Hour14      -0.090768588 -0.19339493  0.01135996
## Hour15       0.006397974 -0.09060525  0.10326533
## Hour16       0.023743343 -0.06996906  0.11752279
## Hour17      -0.036141005 -0.13301160  0.06060202
## Hour18      -0.205852054 -0.30742715 -0.10470564
## Hour19      -0.114498960 -0.21870083 -0.01089971
## Hour20      -0.310891305 -0.41890089 -0.20374386
## Hour21      -0.310154928 -0.41350256 -0.20735290
## Hour22      -0.293820827 -0.39807423 -0.19017351
## Hour23      -0.288906812 -0.39300607 -0.18540330
```

```r
cbind(Estimate=coef(p2), confint(p2))
```

```
## Waiting for profiling to be done...

##                 Estimate       2.5 %      97.5 %
## (Intercept)  3.105192007  3.02975267  3.17929863
## Hour13      -0.011975309 -0.10593695  0.08204045
## Hour14      -0.083297357 -0.18593849  0.01884613
## Hour15       0.002609741 -0.09442406  0.09950773
## Hour16       0.015514684 -0.07823746  0.10933385
## Hour17      -0.037500893 -0.13437725  0.05924791
## Hour18      -0.217514065 -0.31913038 -0.11632607
## Hour19      -0.135002700 -0.23931993 -0.03128670
## Hour20      -0.294227755 -0.40234489 -0.18697096
## Hour21      -0.310891391 -0.41424679 -0.20808152
## Hour22      -0.298349915 -0.40261457 -0.19469120
## Hour23      -0.301237216 -0.40545151 -0.19761716
## Month10     -0.154842314 -0.21387030 -0.09596761
## Month11     -0.227366427 -0.28668220 -0.16820917
## Month12     -0.263446248 -0.32165845 -0.20534945
```

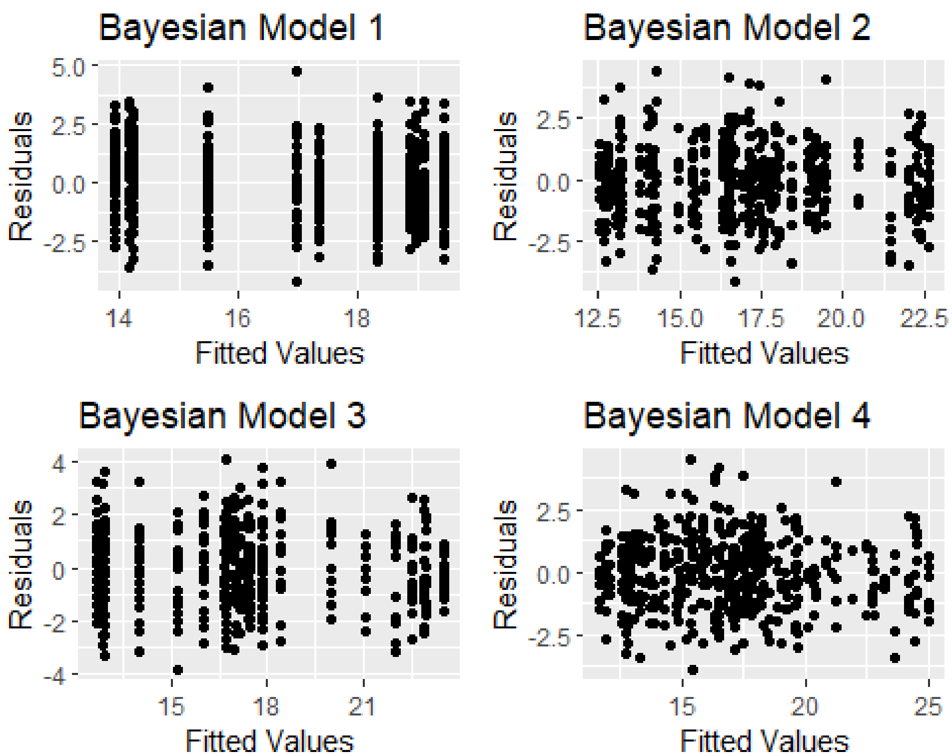
```r
cbind(Estimate=coef(p3), confint(p3))
```

```
## Waiting for profiling to be done...

##                Estimate       2.5 %      97.5 %
## (Intercept)  2.857902288  2.78839812  2.92596220
## Hour13      -0.015387387 -0.10932408  0.07860342
## Hour14      -0.085411560 -0.18804196  0.01672109
## Hour15      -0.001657116 -0.09866995  0.09521987
## Hour16       0.025547101 -0.06816580  0.11932706
## Hour17      -0.039989316 -0.13686211  0.05675592
## Hour18      -0.216326106 -0.31791665 -0.11516411
## Hour19      -0.135011655 -0.23927069 -0.03135460
## Hour20      -0.294072942 -0.40212095 -0.18688643
## Hour21      -0.316511460 -0.41986471 -0.21370376
## Hour22      -0.297669139 -0.40192458 -0.19401975
## Hour23      -0.299380864 -0.40349519 -0.19586212
## Game.Day1    0.272233268  0.22789301  0.31639676

cbind(Estimate=coef(p4), confint(p4))

## Waiting for profiling to be done...

##                Estimate       2.5 %      97.5 %
## (Intercept)  2.960284045  2.87832701  3.04101719
## Hour13      -0.014149184 -0.10811264  0.07986838
## Hour14      -0.080471848 -0.18311646  0.02167515
## Hour15      -0.004086963 -0.10112715  0.09281747
## Hour16       0.017358979 -0.07639458  0.11117956
## Hour17      -0.040464876 -0.13734320  0.05628590
## Hour18      -0.222972212 -0.32459248 -0.12178022
## Hour19      -0.146068739 -0.25041352 -0.04232487
## Hour20      -0.285210604 -0.39335712 -0.17792384
## Hour21      -0.315453179 -0.41881236 -0.21263948
## Hour22      -0.297678522 -0.40194387 -0.19401910
## Hour23      -0.298586476 -0.40280198 -0.19496521
## Month10     -0.074867288 -0.13629049 -0.01358229
## Month11     -0.185543823 -0.24549656 -0.12574728
## Month12     -0.117472250 -0.18389697 -0.05099900
## Game.Day1    0.242193943  0.19125180  0.29311039
```

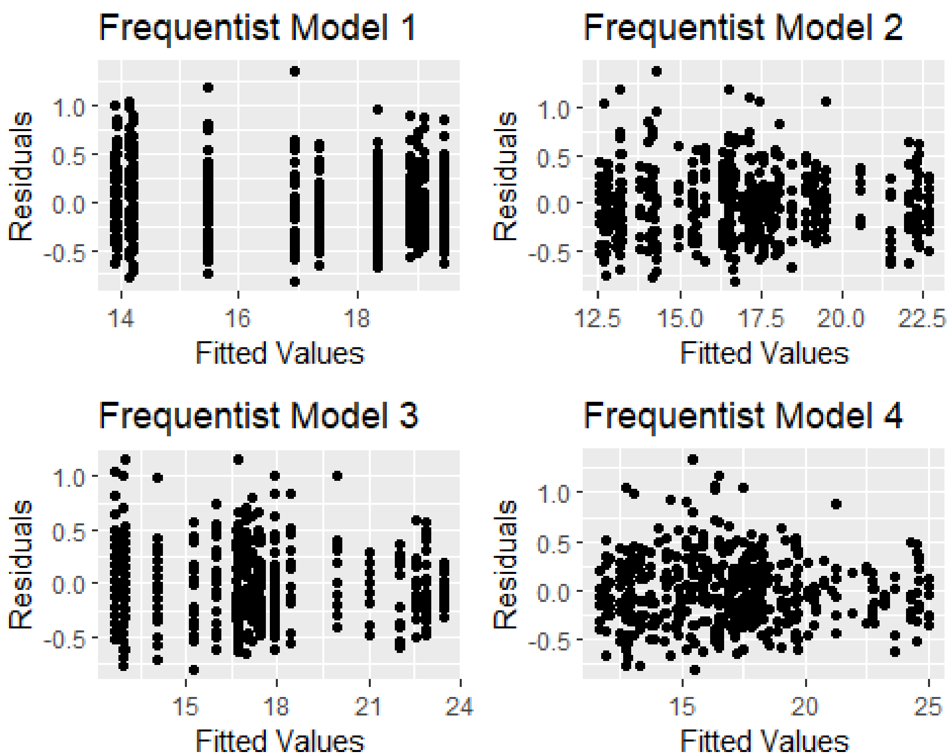
Poisson deviance residuals $d_i = sign(Y_i - \lambda_i)\sqrt{2[Y_i \log(Y_i/\lambda_i) - (Y_i - \lambda_i)]}$

```
# Extract fitted values for Bayesian models
p_fv1 = p_est[57:564]
p_fv2 = p_est[565:1072]
p_fv3 = p_est[1073:1580]
p_fv4 = p_est[1581:2088]

# Calculate deviance residuals for Bayesian models
pois_dev_res = function(fv){
  dr = sign(train_Y-fv)*sqrt(2*(train_Y*log(train_Y/fv)-(train_Y-fv)))
  return(dr)
```

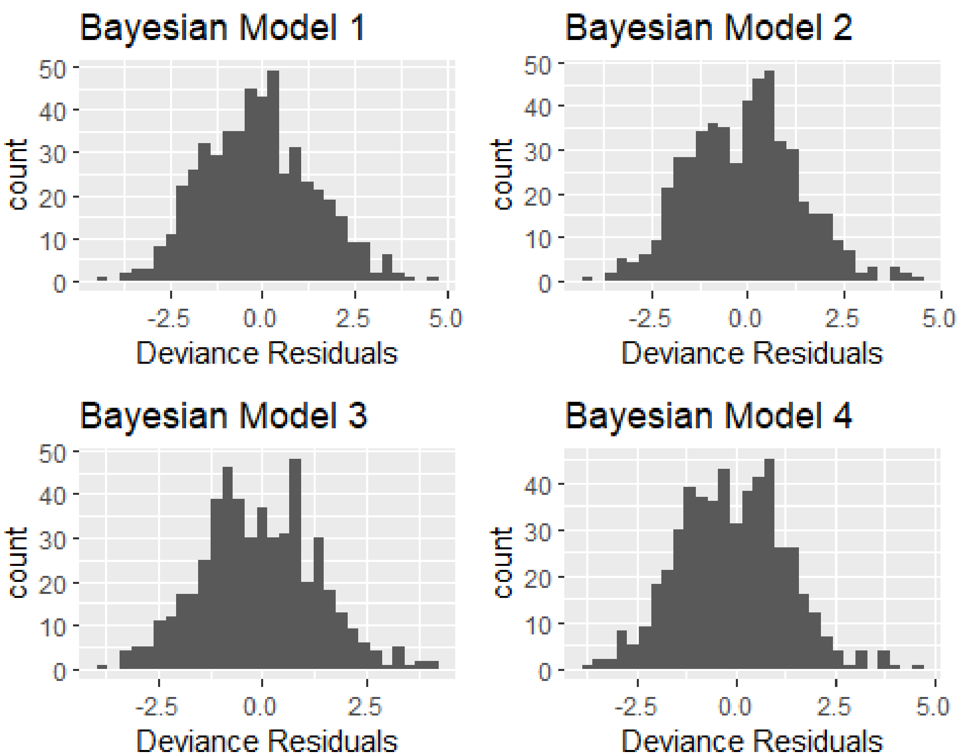
```
}

p_dr = data.frame(
  p_dr1 = pois_dev_res(p_fv1),
  p_dr2 = pois_dev_res(p_fv2),
  p_dr3 = pois_dev_res(p_fv3),
  p_dr4 = pois_dev_res(p_fv4),
  row.names = c(1:508))

# Residual analysis
# Residuals vs fitted values
ggarrange(
  ggplot(p_dr, aes(x=p_fv1, y=p_dr1))+geom_point()+
    xlab('Fitted Values')+ylab('Residuals')+
    ggtitle('Bayesian Model 1'),

  ggplot(p_dr, aes(x=p_fv2, y=p_dr2))+geom_point()+
    xlab('Fitted Values')+ylab('Residuals')+
    ggtitle('Bayesian Model 2'),

  ggplot(p_dr, aes(x=p_fv3, y=p_dr3))+geom_point()+
    xlab('Fitted Values')+ylab('Residuals')+
    ggtitle('Bayesian Model 3'),

  ggplot(p_dr, aes(x=p_fv4, y=p_dr4))+geom_point()+
    xlab('Fitted Values')+ylab('Residuals')+
    ggtitle('Bayesian Model 4'))
```



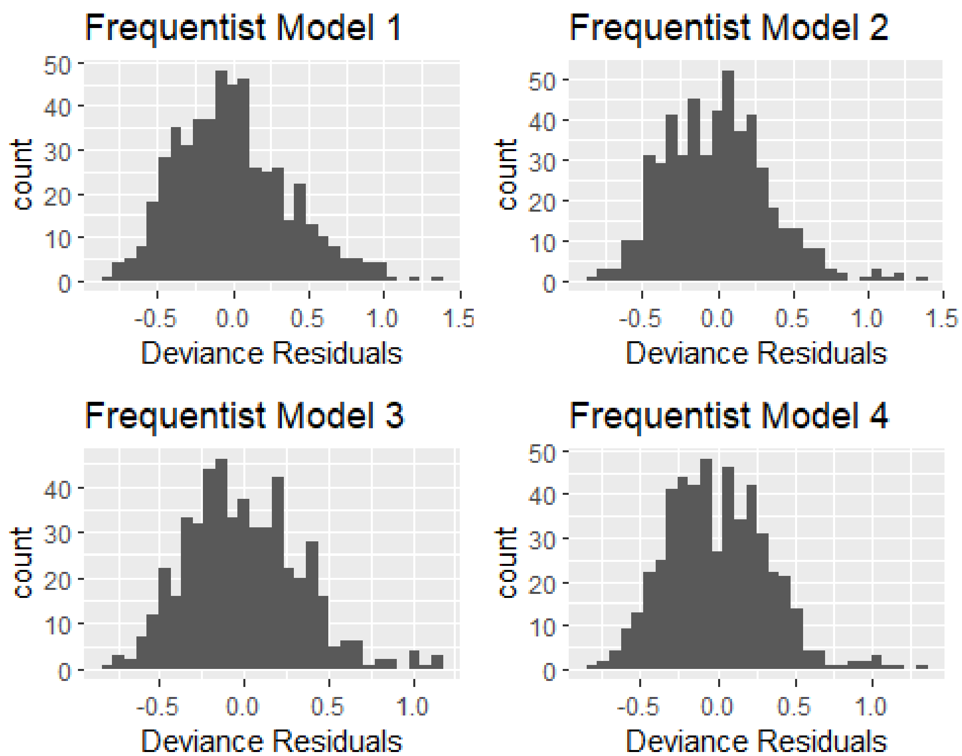

```r
ggarrange(
  ggplot(p1, aes(p1$fitted.values, p1$residuals))+
    geom_point()+xlab('Fitted Values')+
    ylab('Residuals')+ggtitle('Frequentist Model 1'),

  ggplot(p2, aes(p2$fitted.values, p2$residuals))+
    geom_point()+xlab('Fitted Values')+
    ylab('Residuals')+ggtitle('Frequentist Model 2'),

  ggplot(p3, aes(p3$fitted.values, p3$residuals))+
    geom_point()+xlab('Fitted Values')+
    ylab('Residuals')+ggtitle('Frequentist Model 3'),

  ggplot(p4, aes(p4$fitted.values, p4$residuals))+
    geom_point()+xlab('Fitted Values')+
    ylab('Residuals')+ggtitle('Frequentist Model 4'))
```



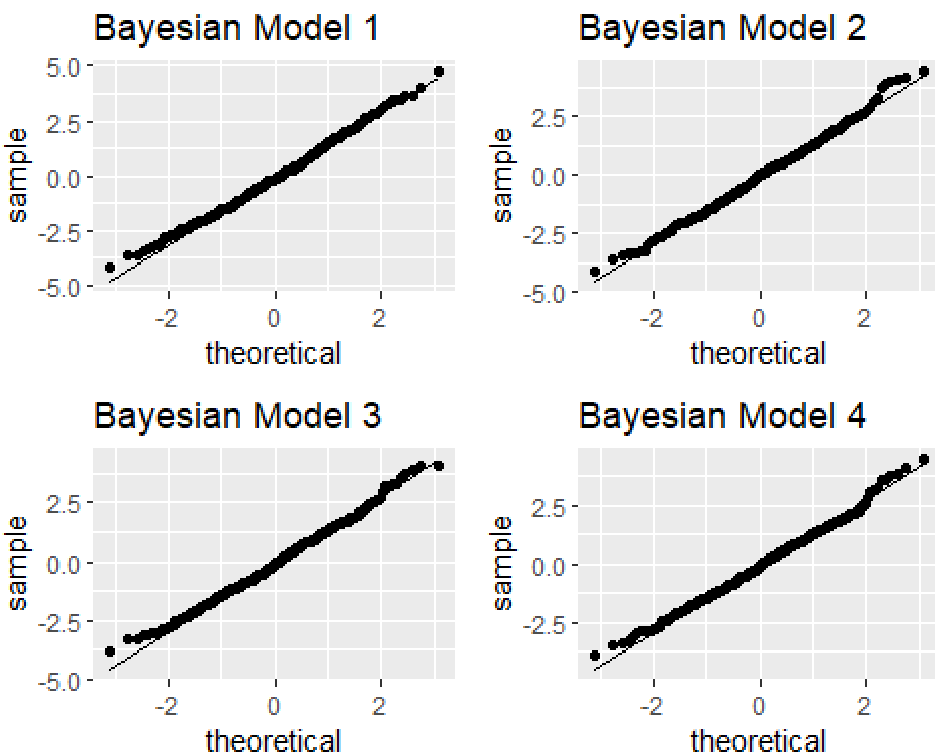

```
# Histogram of the Bayesian residuals
ggarrange(
  ggplot(p_dr, aes(x=p_dr1))+geom_histogram()+
    xlab('Deviance Residuals')+ggtitle('Bayesian Model 1'),

  ggplot(p_dr, aes(x=p_dr2))+geom_histogram()+
    xlab('Deviance Residuals')+ggtitle('Bayesian Model 2'),

  ggplot(p_dr, aes(x=p_dr3))+geom_histogram()+
    xlab('Deviance Residuals')+ggtitle('Bayesian Model 3'),

  ggplot(p_dr, aes(x=p_dr4))+geom_histogram()+
    xlab('Deviance Residuals')+ggtitle('Bayesian Model 4'))

## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



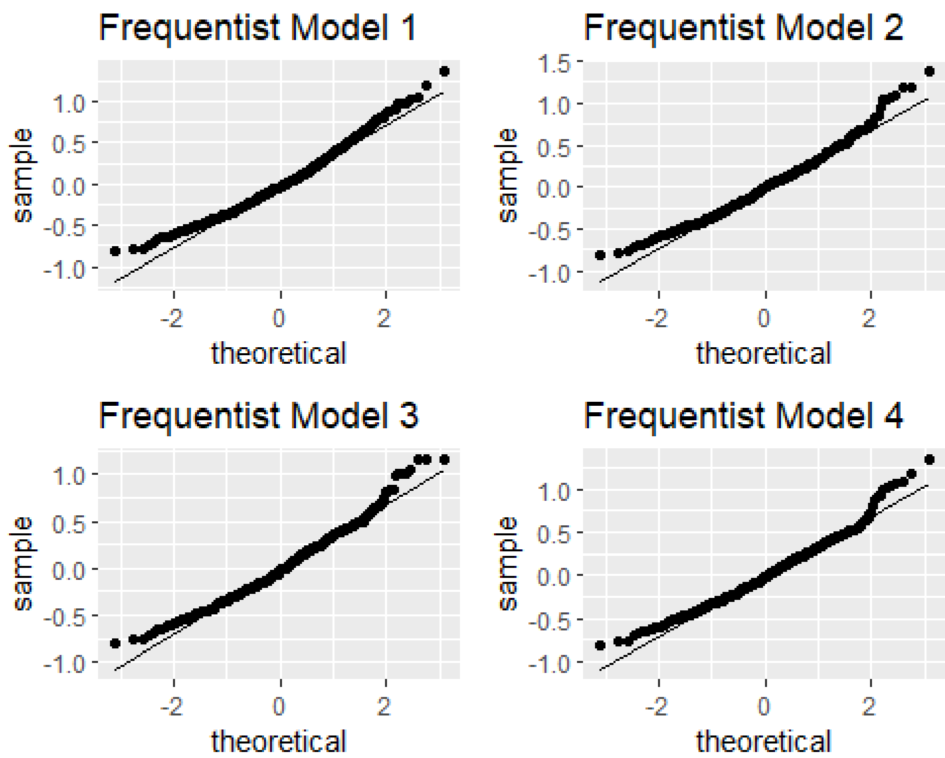

```
# Histogram of frequentist residuals
ggarrange(
  ggplot(p1, aes(x=p1$residuals))+geom_histogram()+
    xlab('Deviance Residuals')+ggtitle('Frequentist Model 1'),

  ggplot(p2, aes(x=p2$residuals))+geom_histogram()+
    xlab('Deviance Residuals')+ggtitle('Frequentist Model 2'),

  ggplot(p3, aes(x=p3$residuals))+geom_histogram()+
    xlab('Deviance Residuals')+ggtitle('Frequentist Model 3'),

  ggplot(p4, aes(x=p4$residuals))+geom_histogram()+
    xlab('Deviance Residuals')+ggtitle('Frequentist Model 4'))

## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```




```
# QQ plot of the Bayesian residuals
ggarrange(ggplot(p_dr, aes(sample=p_dr1))+stat_qq()+stat_qq_line()+
            ggtitle('Bayesian Model 1'),

          ggplot(p_dr, aes(sample=p_dr2))+stat_qq()+stat_qq_line()+
            ggtitle('Bayesian Model 2'),

          ggplot(p_dr, aes(sample=p_dr3))+stat_qq()+stat_qq_line()+
            ggtitle('Bayesian Model 3'),

          ggplot(p_dr, aes(sample=p_dr4))+stat_qq()+stat_qq_line()+
            ggtitle('Bayesian Model 4'))
```




```r
# Histogram of the frequentist residuals
ggarrange(ggplot(p1, aes(sample=p1$residuals))+stat_qq()+stat_qq_line()+
            ggtitle('Frequentist Model 1'),

          ggplot(p2, aes(sample=p2$residuals))+stat_qq()+stat_qq_line()+
            ggtitle('Frequentist Model 2'),

          ggplot(p3, aes(sample=p3$residuals))+stat_qq()+stat_qq_line()+
            ggtitle('Frequentist Model 3'),

          ggplot(p4, aes(sample=p4$residuals))+stat_qq()+stat_qq_line()+
            ggtitle('Frequentist Model 4'))
```




```r
# Calculate the deviance of each Bayesian model
p_d1 = sum(p_dr$p_dr1^2)
p_d2 = sum(p_dr$p_dr2^2)
p_d3 = sum(p_dr$p_dr3^2)
p_d4 = sum(p_dr$p_dr4^2)

# Summary table (deviance, p-value, dispersion)
p_gof = data.frame(
  Deviance_B = c(p_d1, p_d2, p_d3, p_d4),

  GOF_B = c(1-pchisq(p_d1, summary(p1)$df.residual),
            1-pchisq(p_d2, summary(p2)$df.residual),
            1-pchisq(p_d3, summary(p3)$df.residual),
            1-pchisq(p_d4, summary(p4)$df.residual)),

  Dispersion_B = c(p_d1/summary(p1)$df.residual,
                   p_d2/ summary(p1)$df.residual,
                   p_d3/ summary(p1)$df.residual,
                   p_d4/ summary(p1)$df.residual),

  Deviance_F = c(deviance(p1),
                 deviance(p2),
                 deviance(p3),
                 deviance(p4)),

  GOF_F = c(1-pchisq(deviance(p1), summary(p1)$df.residual),
```

```
            1-pchisq(deviance(p2), summary(p2)$df.residual),
            1-pchisq(deviance(p3), summary(p3)$df.residual),
            1-pchisq(deviance(p4), summary(p4)$df.residual)),

  Dispersion_B = c(deviance(p1)/summary(p1)$df.residual,
                   deviance(p2)/summary(p2)$df.residual,
                   deviance(p3)/summary(p3)$df.residual,
                   deviance(p4)/summary(p4)$df.residual),

  row.names = c('Hour', 'Hour_Month',
                'Hour_Game.Day', 'Hour_Month_Game.Day'))

p_gof

##                     Deviance_B GOF_B Dispersion_B Deviance_F GOF_F
## Hour                 1112.9389     0     2.243829  1112.9369     0
## Hour_Month           1019.6453     0     2.055737  1019.6337     0
## Hour_Game.Day         971.4373     0     1.958543   971.4301     0
## Hour_Month_Game.Day   933.1296     0     1.881310   933.1241     0
##                     Dispersion_B.1
## Hour                      2.243824
## Hour_Month                2.068222
## Hour_Game.Day             1.962485
## Hour_Month_Game.Day       1.896594
```

All eight models failed the deviance goodness-of-fit test and showed evidence of being overdispersed. One method of addressing overdispersion is using negative binomial regression.

```
# Bayesian negative binomial regression models adapted from:
# https://georgederpa.github.io/teaching/countModels.html
#
# Accessed on: November 24, 2020
#
#
# Formulas for deviance residuals, log-likelihood, and BIC for
# the negative binomial distribution adapated from formulas
# listed in the following PDF:
# https://ncss-wpengine.netdna-ssl.com/wp-
content/themes/ncss/pdf/Procedures/NCSS/Negative_Binomial_Regression.pdf
#
# Accessed on: November 29, 2020

# Negative binomial models and priors
nb_jags = '
model{
  for(i in 1:n){

    # LIKELIHOODS
```

```
  # Model 1 (Count~Hour)
  Y1[i] ~ dnegbin(p1[i], r1)
  log(lambda1[i]) <- inprod(X1[i,], beta1[])
  p1[i] <- r1/(r1+lambda1[i])

    # Model 2 (Count~Hour+Month)
    Y2[i] ~ dnegbin(p2[i], r2)
  log(lambda2[i]) <- inprod(X2[i,], beta2[])
  p2[i] <- r2/(r2+lambda2[i])

    # Model 3 (Count~Hour+Game.Day)
    Y3[i] ~ dnegbin(p3[i], r3)
  log(lambda3[i]) <- inprod(X3[i,], beta3[])
  p3[i] <- r3/(r3+lambda3[i])

    # Model 4 (Count~Hour+Month+Game.Day)
    Y4[i] ~ dnegbin(p4[i], r4)
  log(lambda4[i]) <- inprod(X4[i,], beta4[])
  p4[i] <- r4/(r4+lambda4[i])
}



# PRIORS

# Priors for Model 1
r1 ~ dunif(1, 100)
for(i in 1:12){
  beta1[i] ~ dnorm(0, 0.001)
}

# Priors for Model 2
r2 ~ dunif(1, 100)
for(i in 1:15){
  beta2[i] ~ dnorm(0, 0.001)
}

# Priors for Model 3
r3 ~ dunif(1, 100)
for(i in 1:13){
  beta3[i] ~ dnorm(0, 0.001)
}

# Priors for Model 4
r4 ~ dunif(1, 100)
for(i in 1:16){
  beta4[i] ~ dnorm(0, 0.001)
}
```

```
  # PREDICTIONS USING THE TEST SET

    for(i in 1:m){

      # Model 1 Predictions
      log(lambda1_star[i]) <- inprod(X1_test[i,], beta1[])
        p1_star[i] <- r1/(r1+lambda1_star[i])
        pred1[i] ~ dnegbin(p1_star[i], r1)

        # Model 2 Predictions
        log(lambda2_star[i]) <- inprod(X2_test[i,], beta2[])
        p2_star[i] <- r2/(r2+lambda2_star[i])
        pred2[i] ~ dnegbin(p2_star[i], r2)

        ## Model 3 Predictions
        log(lambda3_star[i]) <- inprod(X3_test[i,], beta3[])
        p3_star[i] <- r3/(r3+lambda3_star[i])
        pred3[i] ~ dnegbin(p3_star[i], r3)

        # Model 4 Predictions
        log(lambda4_star[i]) <- inprod(X4_test[i,], beta4[])
        p4_star[i] <- r4/(r4+lambda4_star[i])
        pred4[i] ~ dnegbin(p4_star[i], r4)
    }
}
'

# Data
nb_data = list(n=nrow(train_X),
               m=nrow(test_X),

               X1=train_X[,1:12],
               X2=train_X[,1:15],
               X3=train_X[,c(1:12, 16)],
               X4=train_X,

               X1_test=test_X[,1:12],
               X2_test=test_X[,1:15],
               X3_test=test_X[,c(1:12, 16)],
               X4_test=test_X,

               Y1=train_Y,
               Y2=train_Y,
               Y3=train_Y,
               Y4=train_Y)
```

```r
# Initialize models
nb_models = jags.model(file=textConnection(nb_jags), data=nb_data,
                       inits=list(.RNG.name = 'base::Wichmann-Hill',
                                  .RNG.seed =7))

## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 2032
##    Unobserved stochastic nodes: 572
##    Total graph size: 41446
##
## Initializing model

# Burn in
update(nb_models, n.iter=1000)

# Sample
nb_outputs = coda.samples(nb_models,
                          variable.names = c('beta1', 'beta2',
                                             'beta3', 'beta4',
                                             'lambda1', 'lambda2',
                                             'lambda3', 'lambda4',
                                             'pred1', 'pred2',
                                             'pred3', 'pred4',
                                             'r1', 'r2',
                                             'r3', 'r4'),
                          n.iter = iter)

# Create 4 frequentist models
nb1 = glm.nb(Count~Hour, data=train_data)
nb2 = glm.nb(Count~Hour+Month, data=train_data)
nb3 = glm.nb(Count~Hour+Game.Day, data=train_data)
nb4 = glm.nb(Count~Hour+Month+Game.Day, data=train_data)


# Extract estimate of each Bayesian node
nb_est = summary(nb_outputs)$statistics[,1]

# Extract Bayesian 95% equi-tailed credible sets
nb_quants = summary(nb_outputs)$quantiles[,c(1,5)]

# Create a table of coefficients with their credible set
nb_beta1 = cbind(Estimate=nb_est[1:12], nb_quants[1:12,])

nb_beta2 = cbind(Estimate=nb_est[13:27], nb_quants[13:27,])

nb_beta3 = cbind(Estimate=nb_est[28:40], nb_quants[28:40,])
```

```r
nb_beta4 = cbind(nb_est[41:56], nb_quants[41:56,])

# Rename row names for better interpretation
rownames(nb_beta1) = c('(Intercept)', 'Hour13', 'Hour14', 'Hour15', 'Hour16',
                       'Hour17', 'Hour18', 'Hour19', 'Hour20', 'Hour21',
                       'Hour22', 'Hour23')

rownames(nb_beta2) = c('(Intercept)', 'Hour13', 'Hour14', 'Hour15', 'Hour16',
                       'Hour17', 'Hour18', 'Hour19', 'Hour20', 'Hour21',
                       'Hour22', 'Hour23', 'Month10', 'Month11', 'Month12')

rownames(nb_beta3) = c('(Intercept)', 'Hour13', 'Hour14', 'Hour15', 'Hour16',
                       'Hour17', 'Hour18', 'Hour19', 'Hour20', 'Hour21',
                       'Hour22', 'Hour23', 'Game.Day')

rownames(nb_beta4) = c('(Intercept)', 'Hour13', 'Hour14', 'Hour15', 'Hour16',
                       'Hour17', 'Hour18', 'Hour19', 'Hour20', 'Hour21',
                       'Hour22', 'Hour23', 'Month10', 'Month11', 'Month12',
                       'Game.Day')

nb_beta1

##                 Estimate       2.5%       97.5%
## (Intercept)  2.945956561  2.8501333  3.04913398
## Hour13      -0.007702856 -0.1525525  0.13083108
## Hour14      -0.091715678 -0.2436317  0.06088671
## Hour15       0.004687277 -0.1428907  0.14971154
## Hour16       0.021901023 -0.1259354  0.16163502
## Hour17      -0.037374588 -0.1825136  0.10808454
## Hour18      -0.208109587 -0.3583839 -0.06354914
## Hour19      -0.116624234 -0.2753911  0.03742164
## Hour20      -0.311503137 -0.4683523 -0.15678665
## Hour21      -0.309784759 -0.4576187 -0.16190761
## Hour22      -0.294763432 -0.4423637 -0.14737080
## Hour23      -0.290703878 -0.4371789 -0.14452850

nb_beta2

##                 Estimate       2.5%        97.5%
## (Intercept)  3.116355953  3.0053552  3.225979442
## Hour13      -0.017483831 -0.1573635  0.119708926
## Hour14      -0.090901710 -0.2385957  0.060300282
## Hour15      -0.001608053 -0.1484079  0.143059216
## Hour16       0.013509080 -0.1274968  0.151238396
## Hour17      -0.036967859 -0.1762720  0.105216678
## Hour18      -0.222801668 -0.3700270 -0.073179409
## Hour19      -0.146833390 -0.3000294  0.003106644
## Hour20      -0.302444994 -0.4551150 -0.155925716
## Hour21      -0.320962323 -0.4639451 -0.173659792
## Hour22      -0.310708804 -0.4584439 -0.165582601
```

```
## Hour23      -0.308864734 -0.4587167 -0.159789695
## Month10     -0.158556204 -0.2450749 -0.071856281
## Month11     -0.234613623 -0.3193864 -0.149379886
## Month12     -0.271511606 -0.3534394 -0.187445169
```

```
nb_beta3
```

```
##                 Estimate       2.5%         97.5%
## (Intercept)  2.8582438234  2.7619261  2.9580968147
## Hour13      -0.0152006020 -0.1533823  0.1166490073
## Hour14      -0.0859180714 -0.2306254  0.0620140541
## Hour15       0.0001342486 -0.1419478  0.1414802653
## Hour16       0.0272369996 -0.1060675  0.1630312241
## Hour17      -0.0387009109 -0.1800896  0.1007914797
## Hour18      -0.2181378068 -0.3626486 -0.0759936871
## Hour19      -0.1444794149 -0.2952555  0.0005390693
## Hour20      -0.2929693335 -0.4431782 -0.1487326361
## Hour21      -0.3196993937 -0.4615336 -0.1778701986
## Hour22      -0.3070895977 -0.4506681 -0.1594076479
## Hour23      -0.3079636369 -0.4553227 -0.1663394979
## Game.Day     0.2775275766  0.2141724  0.3408365215
```

```
nb_beta4
```

```
##                             2.5%       97.5%
## (Intercept)  2.965507092  2.8538003  3.07750122
## Hour13      -0.015633912 -0.1475658  0.11696541
## Hour14      -0.085283369 -0.2262403  0.05831890
## Hour15      -0.006792115 -0.1443675  0.13276158
## Hour16       0.018119686 -0.1088513  0.15011443
## Hour17      -0.038011856 -0.1741755  0.09695838
## Hour18      -0.228967914 -0.3666803 -0.09168618
## Hour19      -0.160650446 -0.3033978 -0.01770115
## Hour20      -0.288928019 -0.4321319 -0.14143492
## Hour21      -0.322875478 -0.4629820 -0.18378739
## Hour22      -0.312880782 -0.4497295 -0.17247506
## Hour23      -0.309369885 -0.4496991 -0.16992463
## Month10     -0.074558344 -0.1637909  0.01248068
## Month11     -0.189735555 -0.2722486 -0.10825826
## Month12     -0.121146744 -0.2118714 -0.03145375
## Game.Day     0.246587793  0.1774711  0.31702508
```

```
# Print GLM coefficients with 95% confidence intervals
cbind(Estimate=coef(nb1), confint(nb1))
```

```
## Waiting for profiling to be done...
```

```
##                Estimate      2.5 %      97.5 %
## (Intercept)  2.944438979  2.8417659  3.04764851
## Hour13      -0.006741599 -0.1508646  0.13735452
## Hour14      -0.090768588 -0.2456736  0.06416725
```

```
## Hour15       0.006397974 -0.1425428  0.15539581
## Hour16       0.023743343 -0.1206710  0.16816026
## Hour17      -0.036141005 -0.1839769  0.11169439
## Hour18      -0.205852054 -0.3567783 -0.05506722
## Hour19      -0.114498960 -0.2710796  0.04210163
## Hour20      -0.310891305 -0.4682076 -0.15381197
## Hour21      -0.310154928 -0.4613573 -0.15920644
## Hour22      -0.293820827 -0.4465428 -0.14132915
## Hour23      -0.288906812 -0.4415247 -0.13651389
```

```
cbind(Estimate=coef(nb2), confint(nb2))
```

```
## Waiting for profiling to be done...
```

```
##                 Estimate      2.5 %      97.5 %
## (Intercept)  3.105586057  2.9957839  3.21564715
## Hour13      -0.008244344 -0.1458560  0.12935337
## Hour14      -0.080733634 -0.2288283  0.06731980
## Hour15       0.006349080 -0.1358402  0.14856416
## Hour16       0.021538772 -0.1163044  0.15939233
## Hour17      -0.027018539 -0.1682252  0.11417424
## Hour18      -0.214695680 -0.3591614 -0.07040536
## Hour19      -0.137443567 -0.2873529  0.01238576
## Hour20      -0.292372276 -0.4433239 -0.14172325
## Hour21      -0.311776794 -0.4566403 -0.16719115
## Hour22      -0.301030871 -0.4473778 -0.15495465
## Hour23      -0.298123482 -0.4444990 -0.15201267
## Month10     -0.155997251 -0.2413145 -0.07071792
## Month11     -0.232338454 -0.3171410 -0.14760056
## Month12     -0.269287844 -0.3524635 -0.18618891
```

```
cbind(Estimate=coef(nb3), confint(nb3))
```

```
## Waiting for profiling to be done...
```

```
##                  Estimate      2.5 %       97.5 %
## (Intercept)  2.8567477729  2.7590197  2.954484714
## Hour13      -0.0137563763 -0.1477959  0.120273580
## Hour14      -0.0841813176 -0.2285313  0.060085182
## Hour15       0.0008544508 -0.1376598  0.139376033
## Hour16       0.0282255003 -0.1060011  0.162469628
## Hour17      -0.0371556593 -0.1747279  0.100388915
## Hour18      -0.2167359356 -0.3576780 -0.075986754
## Hour19      -0.1423735185 -0.2885383  0.003662098
## Hour20      -0.2917786823 -0.4390566 -0.144845484
## Hour21      -0.3167679053 -0.4582219 -0.175607595
## Hour22      -0.3055505437 -0.4484304 -0.162964459
## Hour23      -0.3070601354 -0.4498717 -0.164539199
## Game.Day1    0.2776555258  0.2149357  0.340417302
```

```
cbind(Estimate=coef(nb4), confint(nb4))
```

```
## Waiting for profiling to be done...

##                  Estimate      2.5 %      97.5 %
## (Intercept)  2.9596691631  2.8469489  3.07231954
## Hour13      -0.0106520122 -0.1418888  0.12058220
## Hour14      -0.0794722721 -0.2208872  0.06182941
## Hour15      -0.0005396158 -0.1361472  0.13506212
## Hour16       0.0234302601 -0.1079662  0.15484596
## Hour17      -0.0321532832 -0.1668639  0.10252454
## Hour18      -0.2222290286 -0.3603803 -0.08428603
## Hour19      -0.1541511793 -0.2974702 -0.01100283
## Hour20      -0.2825127634 -0.4270979 -0.13830094
## Hour21      -0.3162374926 -0.4549750 -0.17780536
## Hour22      -0.3055689943 -0.4456876 -0.16576114
## Hour23      -0.3031745121 -0.4433205 -0.16333277
## Month10     -0.0744623263 -0.1591460  0.01021312
## Month11     -0.1893812587 -0.2713198 -0.10749426
## Month12     -0.1205447133 -0.2104571 -0.03064710
## Game.Day1    0.2467946526  0.1767296  0.31689725
```

Coefficients should match the associated Poisson model (or be approximately the same for the Bayesian models due to the MCMC sampling). However, the variance assumption of Poisson regression is now loosened allowing for a better fit.

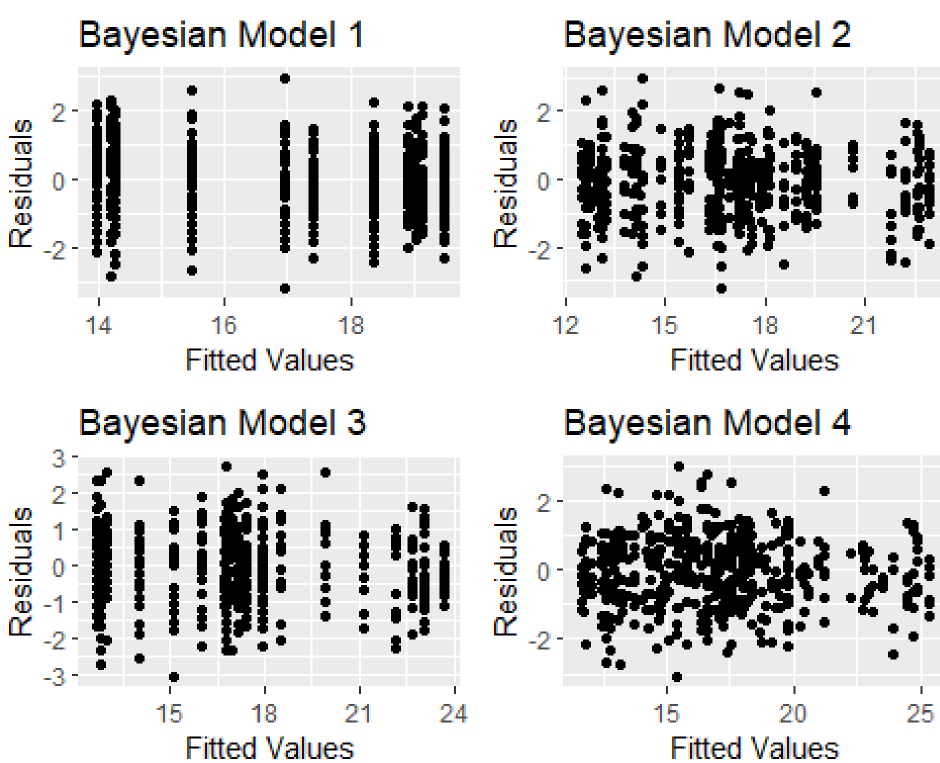
```
# Extract fitted values for Bayesian models
nb_fv1 = nb_est[57:564]
nb_fv2 = nb_est[565:1072]
nb_fv3 = nb_est[1073:1580]
nb_fv4 = nb_est[1581:2088]

# Extract r for each NB(p,r) distribution
r1 = nb_est[2601]
r2 = nb_est[2602]
r3 = nb_est[2603]
r4 = nb_est[2404]
```

Negative binomial, NB(p,r), deviance residuals

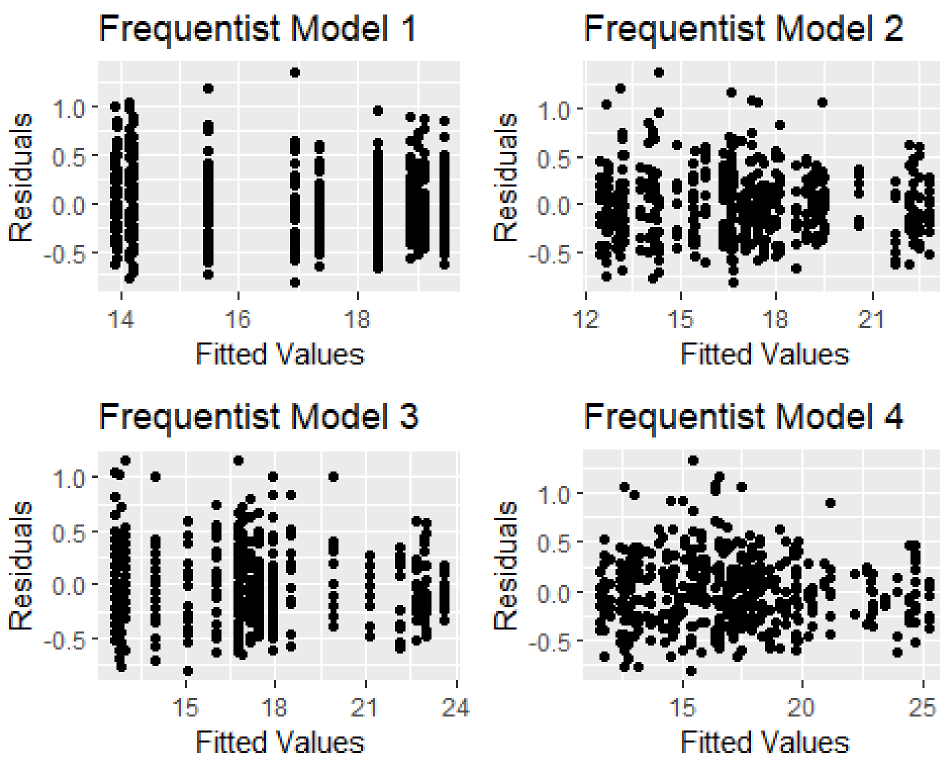
$$d_i = sign(Y_i - \lambda_i)\sqrt{2[Y_i \log(Y_i/\lambda_i) - (Y_i + r)\log(\frac{1 + Y_i/r}{1 + \lambda_i/r})]}$$

```
# Calculate deviance residuals
nb_dev_res = function(fv, r){

  dr = sign(train_Y-fv)*sqrt(2*(train_Y*log(train_Y/fv)
            -(train_Y+r)*log((1+train_Y/r)/(1+fv/r))))

  return(dr)
}
```

```
# Calculate deviance residuals for Bayesian models
nb_dr = data.frame(
  nb_dr1 = nb_dev_res(nb_fv1, r1),
  nb_dr2 = nb_dev_res(nb_fv2, r2),
  nb_dr3 = nb_dev_res(nb_fv3, r3),
  nb_dr4 = nb_dev_res(nb_fv4, r4),
  row.names = c(1:508))

# Residual analysis
# Residuals vs fitted values
ggarrange(
  ggplot(nb_dr, aes(x=nb_fv1, y=nb_dr1))+geom_point()+
    xlab('Fitted Values')+ylab('Residuals')+
    ggtitle('Bayesian Model 1'),

  ggplot(nb_dr, aes(x=nb_fv2, y=nb_dr2))+geom_point()+
    xlab('Fitted Values')+ylab('Residuals')+
    ggtitle('Bayesian Model 2'),

  ggplot(nb_dr, aes(x=nb_fv3, y=nb_dr3))+geom_point()+
    xlab('Fitted Values')+ylab('Residuals')+
    ggtitle('Bayesian Model 3'),

  ggplot(nb_dr, aes(x=nb_fv4, y=nb_dr4))+geom_point()+
    xlab('Fitted Values')+ylab('Residuals')+
    ggtitle('Bayesian Model 4'))
```

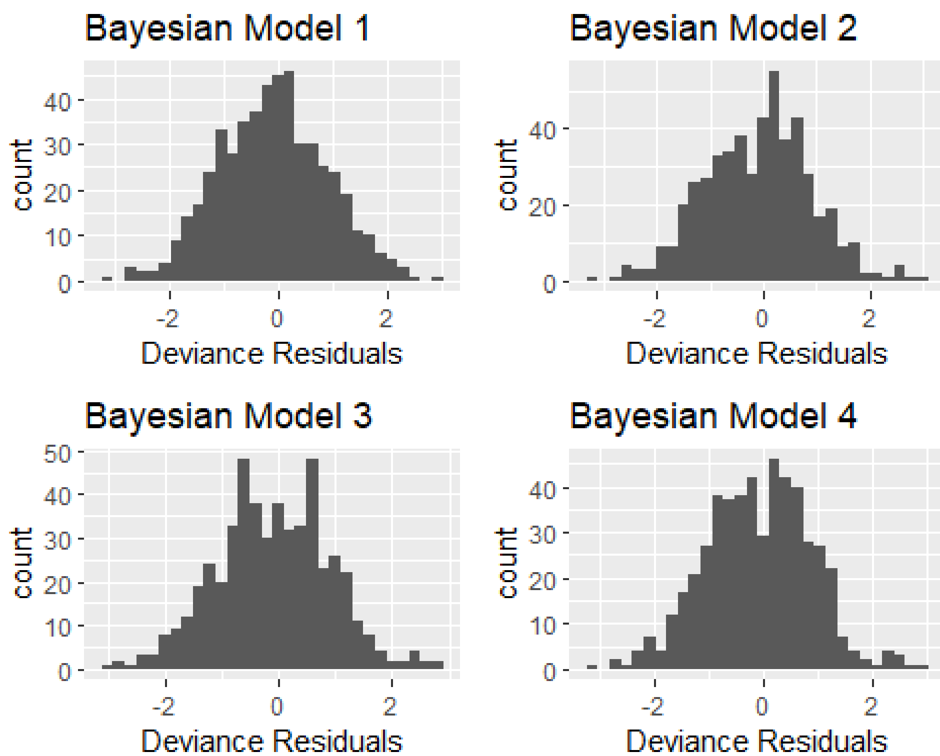


```
ggarrange(
  ggplot(nb1, aes(nb1$fitted.values, nb1$residuals))+
    geom_point()+xlab('Fitted Values')+
    ylab('Residuals')+ggtitle('Frequentist Model 1'),

  ggplot(nb2, aes(nb2$fitted.values, nb2$residuals))+
    geom_point()+xlab('Fitted Values')+
    ylab('Residuals')+ggtitle('Frequentist Model 2'),

  ggplot(nb3, aes(nb3$fitted.values, nb3$residuals))+
    geom_point()+xlab('Fitted Values')+
    ylab('Residuals')+ggtitle('Frequentist Model 3'),

  ggplot(nb4, aes(nb4$fitted.values, nb4$residuals))+
    geom_point()+xlab('Fitted Values')+
    ylab('Residuals')+ggtitle('Frequentist Model 4'))
```



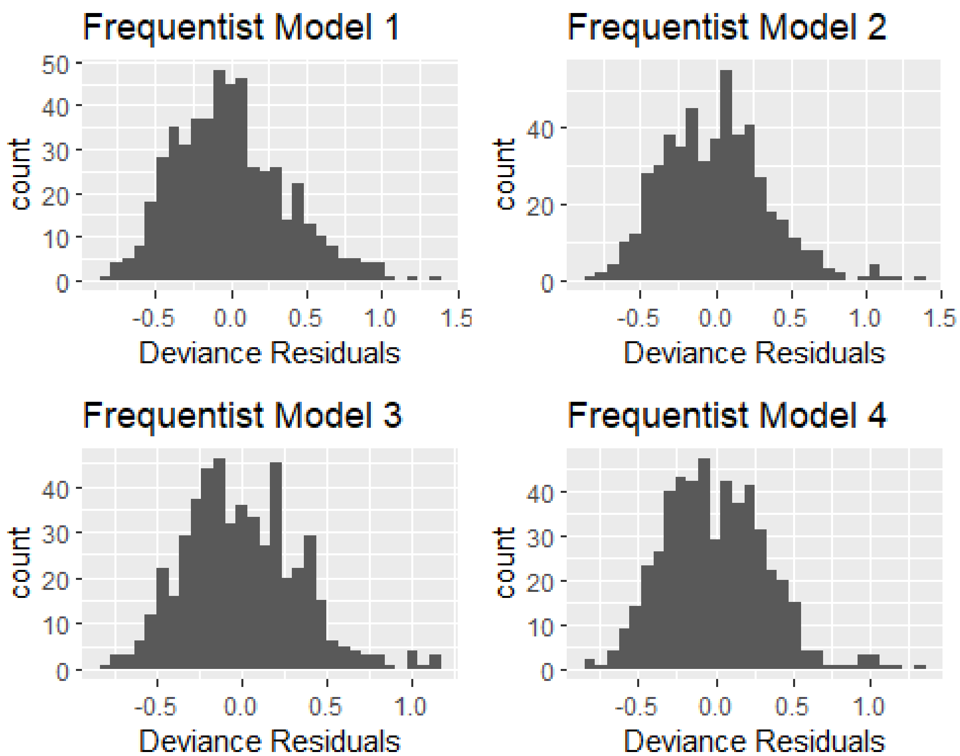

```
# Histogram of the Bayesian residuals
ggarrange(
  ggplot(nb_dr, aes(x=nb_dr1))+geom_histogram()+
    xlab('Deviance Residuals')+ggtitle('Bayesian Model 1'),

  ggplot(nb_dr, aes(x=nb_dr2))+geom_histogram()+
    xlab('Deviance Residuals')+ggtitle('Bayesian Model 2'),

  ggplot(nb_dr, aes(x=nb_dr3))+geom_histogram()+
```

```
    xlab('Deviance Residuals')+ggtitle('Bayesian Model 3'),

  ggplot(nb_dr, aes(x=nb_dr4))+geom_histogram()+
    xlab('Deviance Residuals')+ggtitle('Bayesian Model 4'))

## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

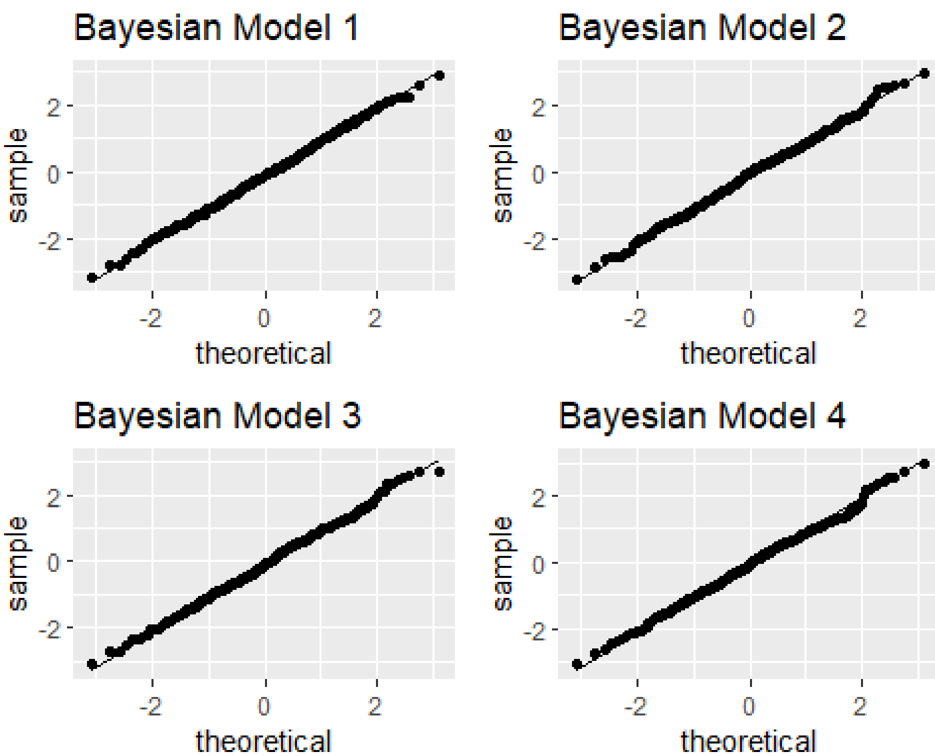



```
# Histogram of the frequentist residuals
ggarrange(
  ggplot(nb1, aes(x=nb1$residuals))+geom_histogram()+
    xlab('Deviance Residuals')+ggtitle('Frequentist Model 1'),

  ggplot(nb2, aes(x=nb2$residuals))+geom_histogram()+
    xlab('Deviance Residuals')+ggtitle('Frequentist Model 2'),

  ggplot(nb3, aes(x=nb3$residuals))+geom_histogram()+
    xlab('Deviance Residuals')+ggtitle('Frequentist Model 3'),

  ggplot(nb4, aes(x=nb4$residuals))+geom_histogram()+
    xlab('Deviance Residuals')+ggtitle('Frequentist Model 4'))

## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

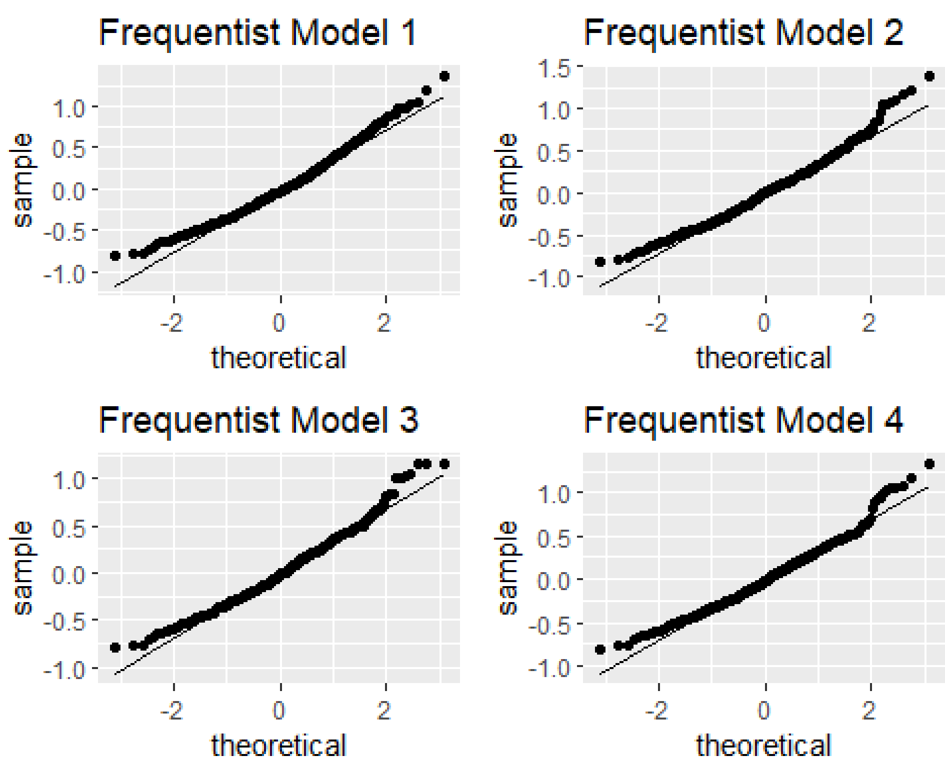
```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```




```
# QQ plot of the Bayesian residuals
ggarrange(ggplot(nb_dr, aes(sample=nb_dr1))+stat_qq()+stat_qq_line()+
            ggtitle('Bayesian Model 1'),

          ggplot(nb_dr, aes(sample=nb_dr2))+stat_qq()+stat_qq_line()+
            ggtitle('Bayesian Model 2'),

          ggplot(nb_dr, aes(sample=nb_dr3))+stat_qq()+stat_qq_line()+
            ggtitle('Bayesian Model 3'),

          ggplot(nb_dr, aes(sample=nb_dr4))+stat_qq()+stat_qq_line()+
            ggtitle('Bayesian Model 4'))
```




```
# QQ plot of the frequentist residuals
ggarrange(ggplot(nb1, aes(sample=nb1$residuals))+stat_qq()+stat_qq_line()+
            ggtitle('Frequentist Model 1'),

          ggplot(nb2, aes(sample=nb2$residuals))+stat_qq()+stat_qq_line()+
            ggtitle('Frequentist Model 2'),

          ggplot(nb3, aes(sample=nb3$residuals))+stat_qq()+stat_qq_line()+
            ggtitle('Frequentist Model 3'),

          ggplot(nb4, aes(sample=nb4$residuals))+stat_qq()+stat_qq_line()+
            ggtitle('Frequentist Model 4'))
```



Log-likelihood

$$LL_i = \log(\Gamma(Y_i + r)) - \log(\Gamma(r)) - \log(\Gamma(Y_i + 1)) - r\log(1 + \lambda_i/r) - Y_i\log(1 + \lambda_i/r) + Y_i\log(1/r)) + Y_i\log(\lambda_i)$$

BIC

$BIC = -2\sum_{i=1}^{n} LL_i + p\log(n)$, where p=number of predicting variables.

```
# Calculate BIC
bayes_bic = function(fv, num_params, r){

  ll = log(gamma(train_Y+r))-log(gamma(r))-log(gamma(train_Y+1))-
    r*log(1+fv/r)-train_Y*log(1+fv/r)+train_Y*log(1/r)+
    train_Y*log(fv)

  return(-2*sum(ll)+num_params*log(508))
}


# Table of BIC values
bic = data.frame(
  BIC_B = c(bayes_bic(nb_fv1, 12, r1),
            bayes_bic(nb_fv2, 15, r2),
            bayes_bic(nb_fv3, 13, r3),
            bayes_bic(nb_fv4, 16, r4)),
```

```
  BIC_F = c(AIC(nb1, k=log(508)),
            AIC(nb2, k=log(508)),
            AIC(nb3, k=log(508)),
            AIC(nb4, k=log(508))),

  row.names = c('Hour', 'Hour_Month',
                'Hour_Game.Day', 'Hour_Month_Game.Day'))

bic

##                        BIC_B    BIC_F
## Hour                3317.759 3323.949
## Hour_Month          3290.887 3296.984
## Hour_Game.Day       3253.688 3259.874
## Hour_Month_Game.Day 3252.775 3257.635

# Calculate the deviance of each Bayesian model
nb_d1 = sum(nb_dr$nb_dr1^2)
nb_d2 = sum(nb_dr$nb_dr2^2)
nb_d3 = sum(nb_dr$nb_dr3^2)
nb_d4 = sum(nb_dr$nb_dr4^2)

# Summary table (deviance, p-value, dispersion)
nb_gof = data.frame(
  Deviance_B = c(nb_d1, nb_d2, nb_d3, nb_d4),

  GOF_B = c(1-pchisq(nb_d1, summary(nb1)$df.residual),
            1-pchisq(nb_d2, summary(nb2)$df.residual),
            1-pchisq(nb_d3, summary(nb3)$df.residual),
            1-pchisq(nb_d4, summary(nb4)$df.residual)),

  Deviance_F = c(deviance(nb1),
                 deviance(nb2),
                 deviance(nb3),
                 deviance(nb4)),

  GOF_F = c(1-pchisq(deviance(nb1), summary(nb1)$df.residual),
            1-pchisq(deviance(nb2), summary(nb2)$df.residual),
            1-pchisq(deviance(nb3), summary(nb3)$df.residual),
            1-pchisq(deviance(nb4), summary(nb4)$df.residual)),

  row.names = c('Hour', 'Hour_Month',
                'Hour_Game.Day', 'Hour_Month_Game.Day'))

nb_gof

##                     Deviance_B     GOF_B Deviance_F     GOF_F
## Hour                  514.8099 0.2706230   519.7349 0.2227395
## Hour_Month            511.4881 0.2733389   519.0913 0.2010619
```

```
## Hour_Game.Day         514.8000 0.2603701   519.4384 0.2161151
## Hour_Month_Game.Day   483.2372 0.6024672   518.9991 0.1930801

# Bayesian predictions
nb_pred1 = nb_est[2089:2216]
nb_pred2 = nb_est[2217:2344]
nb_pred3 = nb_est[2345:2472]
nb_pred4 = nb_est[2473:2600]

# GLM predictions
nb1_pred = predict(nb1, test_data, type='response')
nb2_pred = predict(nb2, test_data, type='response')
nb3_pred = predict(nb3, test_data, type='response')
nb4_pred = predict(nb4, test_data, type='response')

# Mean Square Prediction Error and Precision Error
mspe = function(pred){
  mspe = mean((pred-test_data$Count)^2)
  return(mspe)
}

precision = function(pred){
  precision = sum((pred-test_data$Count)^2)/
    sum((test_data$Count-mean(test_data$Count))^2)

  return(precision)
}

nb_errors = data.frame(

  MSPE_B = c(mspe(nb_pred1), mspe(nb_pred2),
             mspe(nb_pred3), mspe(nb_pred4)),

  MSPE_F = c(mspe(nb1_pred), mspe(nb2_pred),
             mspe(nb3_pred), mspe(nb4_pred)),

  Precison_B = c(precision(nb_pred1), precision(nb_pred2),
                 precision(nb_pred3), precision(nb_pred4)),

  Precison_F = c(precision(nb1_pred), precision(nb2_pred),
                 precision(nb3_pred), precision(nb4_pred)),

  row.names = c('Hour', 'Hour_Month',
                'Hour_Game.Day', 'Hour_Month_Game.Day'))

nb_errors

##                       MSPE_B   MSPE_F Precison_B Precison_F
## Hour                45.17452 45.22076  0.9672709  0.9682612
```

```
## Hour_Month           38.08498 37.97594  0.8154707  0.8131360
## Hour_Game.Day        38.80083 38.72979  0.8307983  0.8292772
## Hour_Month_Game.Day 36.23542 36.23058  0.7758682  0.7757645
```