R_HW10_Count Response

Eom SangJun

2020 11 29

간질환자들을 대상으로 8 주 동안 발작의 횟수를 조사한 데이터를 이용해 Count Response Data 를 분석해보자.

무작위배정을 통해 31 명의 환자들은 Progabide 라는 약에 배정받았고 28 명의 환자는 Placebo 약(위약)에 배정받았다. 환자들은 2 주를 기준으로 총 4 번의 관찰되었고, 각 기간 동안의 발작횟수가 기록되었다.

Progabide 가 정말 발작의 비율을 줄여줄 수 있는지를 알아보자.

```
library(faraway)
data(epilepsy, package = 'faraway')
epilepsy$period <- rep(0:4, 59)
epilepsy$drug <- factor(c('placebo', 'treatment')[epilepsy$treat+1])
epilepsy$phase <- factor(c('baseline', 'experiment')[epilepsy$expind+1])
epilepsy[epilepsy$id < 2.5, ]
```

```
##    seizures id treat expind timeadj age period    drug      phase
## 1        11  1     0      0       8  31      0 placebo   baseline
## 2         5  1     0      1       2  31      1 placebo experiment
## 3         3  1     0      1       2  31      2 placebo experiment
## 4         3  1     0      1       2  31      3 placebo experiment
## 5         3  1     0      1       2  31      4 placebo experiment
## 6        11  2     0      0       8  30      0 placebo   baseline
## 7         3  2     0      1       2  30      1 placebo experiment
## 8         5  2     0      1       2  30      2 placebo experiment
## 9         3  2     0      1       2  30      3 placebo experiment
## 10        3  2     0      1       2  30      4 placebo experiment
```

→ Treat = 0 은 placebo 를 받았다는 것을 의미하며, expind 는 baseline 단계인지 아닌지를 나타낸다(0 인 경우 baseline 이다). 그리고 편의를 위해 period, drug, phase 세 변수를 추가해주었다.

이제 주당 발작횟수의 평균을 구하되, treatment 와 phase 로 범주화 시켜 살펴보자.

```
library(dplyr)
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
epilepsy %>%
  group_by(drug, phase) %>%
  summarise(rate=mean(seizures/timeadj)) %>%
  xtabs(formula = rate ~ phase + drug)
```

```
## `summarise()` regrouping output by 'drug' (override with `.groups` argumen
t)
```

```
##             drug
## phase         placebo treatment
##   baseline   3.848214  3.955645
##   experiment 4.303571  3.983871
```

→ treatment 를 살펴보면, 실험기간 동안 발작횟수가 오히려 증가했음을 확인할 수 있다.

또한 placebo 의 경우는 훨씬 더 많이 증가했다. Treatment period 동안 사실 drug 는 발작을 감소하는 역할을 했음에도 불구하고, 다른 요인으로 인해 발작이 늘어났을 수도 있다.

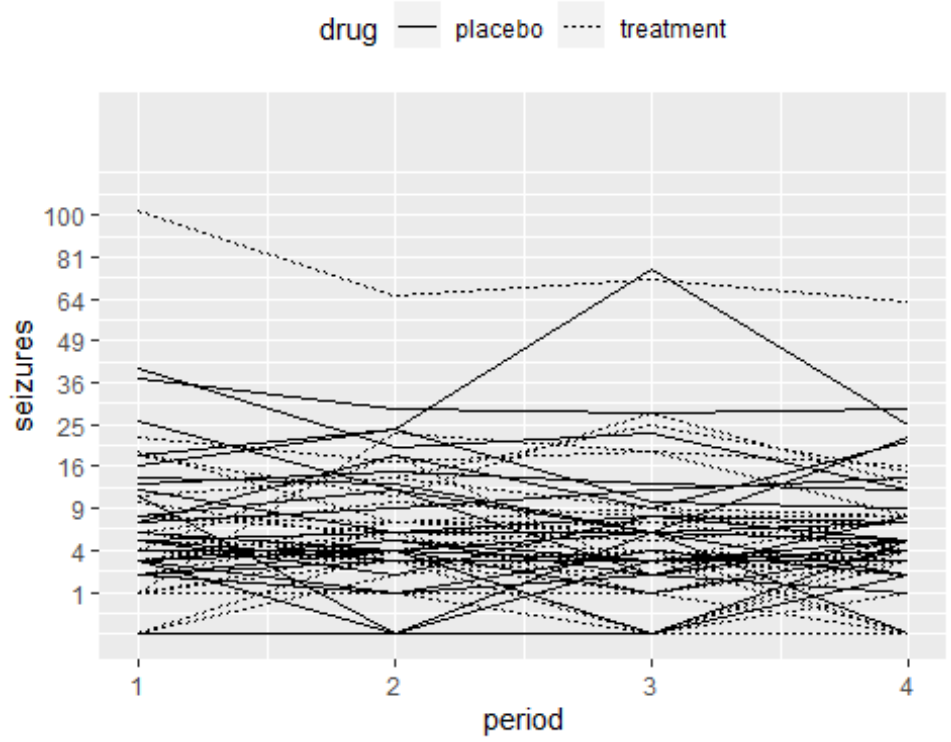
이제 우리는 treatment group 과 control group 간의 차이를 plot 을 통해 알아보자.

첫 번째 plot 은 오직 experiment period 동안의 두 그룹 간의 차이를 보여준다.

```
library(ggplot2)
ggplot(epilepsy, aes(x=period, y=seizures, linetype=drug, group=id)) +
  geom_line() +
  xlim(1,4) +
  scale_y_sqrt(breaks=(0:10)^2) +
  theme(legend.position = 'top', legend.direction='horizontal')
```

→ response 에 square root 를 씌워준 것은 variance 를 안정적으로 만들기 위해서이다. Count data 에서 종종 쓰이는 방법이다.

```
## Warning: Removed 59 row(s) containing missing values (geom_path).
```



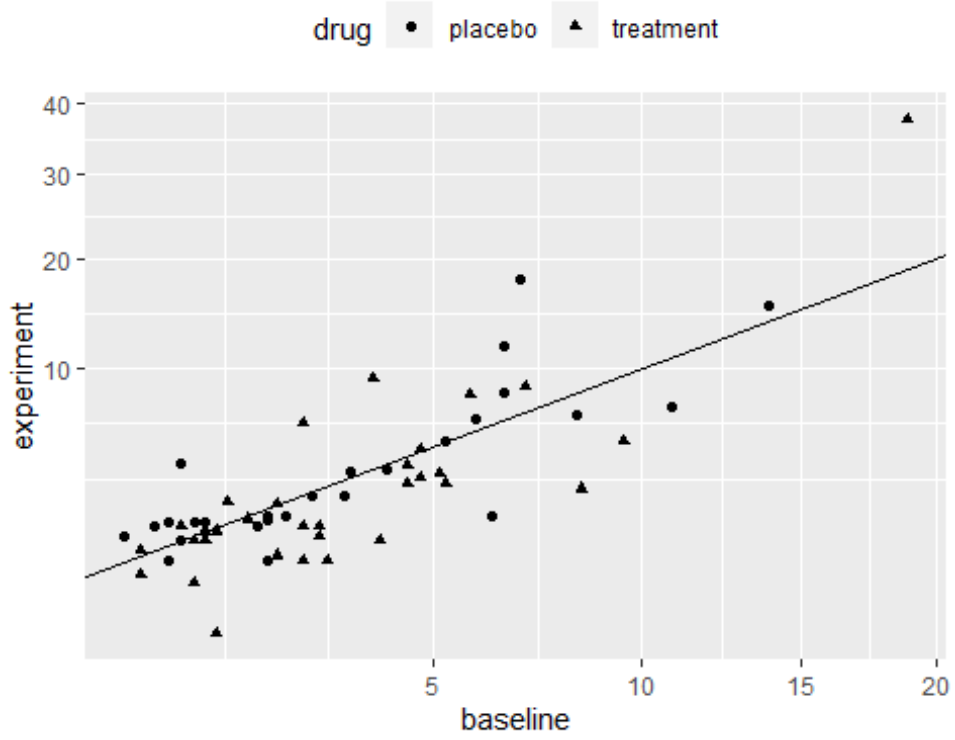
➔ 뚜렷한 차이를 보기 힘들다.

이번에는 두 그룹간의 baseline 을 비교하는 plot 을 그려보자.

```
ratesum <- epilepsy %>%
  group_by(id, phase, drug) %>%
  summarise(rate=mean(seizures/timeadj))

## `summarise()` regrouping output by 'id', 'phase' (override with `.groups`
argument)

library(tidyr)
comsum <- spread(ratesum, phase, rate)
ggplot(comsum, aes(x=baseline, y=experiment, shape = drug)) +
  geom_point() +
  scale_x_sqrt() +
  scale_y_sqrt() +
  geom_abline(intercept=0, slope=1) +
  theme(legend.position = 'top', legend.direction='horizontal')
```



➔ Treatment effect 가 뚜렷하게 나타나는 것 같지는 않다.

이제 GLMM model 을 fitting 해보자.

그 전에 너무 높은 발작율을 보이는 49 번 환자는 제외하도록 하자.

주의할 것은 case 를 함부로 제외하는 것은 위험하며, 하더라도 제외했다는 사실을 꼭 명시해야 한다는 점이다. 지금은 필요 없어 보이거나, outlier 처럼 보일지라도 다른 분석에서는 다를 수 있기 때문이다.

```
epilo <- filter(epilepsy, id != 49)
```

비록 observation 들의 group 화로 model 이 맞지 않을 수 있지만 GLM 으로 시작하는 것은 가치가 있다. Baseline 과 treatment period 간의 길이 차이가 존재하기 때문에, offset 을 설정해주도록 하자.

```
modglm <- glm(seizures ~offset(log(timeadj)) + expind + treat + I(expind*treat),
              family=poisson, data=epilo)
sumary(modglm)
```

```
##                     Estimate Std. Error z value  Pr(>|z|)
## (Intercept)         1.347609   0.034060 39.5656 < 2.2e-16
## expind              0.111836   0.046877  2.3857   0.01704
## treat              -0.106822   0.048630 -2.1966   0.02805
## I(expind * treat) -0.302384   0.069713 -4.3375 1.441e-05
##
## n = 290 p = 4
## Deviance = 2411.54986 Null Deviance = 2485.10988 (Difference = 73.56002)
```

→ 우리의 주 관심사는 interaction 이다. 모든 환자들은 baseline 에서는 treat 되지 않은 상태이다. 이는 treatment 에 대한 main effect 가 treatment 에 대한 반응을 잘 측정한다는 것인데, 이는 baseline period 를 포함하기 때문이다. 우리가 확인했던 것처럼, 우리는 baseline period 와 active period 간에 response 에 차이가 있을 것이라고 추측하였다. 그리고 Interaction term 은 이를 보여주는데, 위에서 볼 수 있듯이 매우 유의하다는 것을 알 수 있다. 그리고 coefficient 값은 negative 인데, 우리는 발작을 줄이고자 하는 것이 목적이므로 이에 잘 부합한다고 할 수 있다.

그러나 문제는 우리가 개인들 내에서의 response 가 correlated 되어있다는 것을 상정하지 않았다는 점이다.

추가적으로 overdispersion 문제 또한 고려해볼 수 있으나 여기서는 잠시 넘어가자.

Binary Case 와 마찬가지로 PQL 로 random effect 를 고려해보자.

```
library(MASS)
```

```
##
## Attaching package: 'MASS'
```

```
## The following object is masked from 'package:dplyr':
##
##     select
```

```
modpql <- glmmPQL(seizures ~ offset(log(timeadj)) + expind + treat +
                    I(expind*treat), random = ~1|id, family=poisson, data=epi
lo)
```

```
## iteration 1
```

```
## iteration 2
```

```
## iteration 3
```

```
## iteration 4
```

```
## iteration 5
```

```
summary(modpql)
```

```
## Linear mixed-effects model fit by maximum likelihood
##  Data: epilo
##   AIC BIC logLik
##    NA  NA     NA
##
## Random effects:
##  Formula: ~1 | id
##         (Intercept) Residual
## StdDev:   0.6819731 1.605408
##
## Variance function:
##  Structure: fixed weights
##  Formula: ~invwt
## Fixed effects: seizures ~ offset(log(timeadj)) + expind + treat + I(expind
 *      treat)
##                         Value  Std.Error  DF   t-value p-value
## (Intercept)        1.0807909 0.14370130 230  7.521094  0.0000
## expind             0.1118360 0.07576686 230  1.476055  0.1413
## treat             -0.0089374 0.20024399  56 -0.044632  0.9646
## I(expind * treat) -0.3023841 0.11268890 230 -2.683353  0.0078
##  Correlation:
##                   (Intr) expind treat
## expind            -0.278
## treat             -0.718  0.200
## I(expind * treat)  0.187 -0.672 -0.274
##
## Standardized Within-Group Residuals:
##        Min         Q1        Med         Q3        Max
## -2.2956801 -0.5651709 -0.1501522  0.3243903  6.3134910
##
## Number of Observations: 290
## Number of Groups: 58
```

→ Estimation 의 경우 GLM 과 비슷하지만, 역시 standard error 에서 차이를 보이는 것을 알 수 있다. PQL 이 더 크게 나온다.

그런데 binary response 에서와 마찬가지로 여기서도 추론의 정확도에 대해 의심해볼 필요가 있다. 특히 count 의 크기가 작은 경우에 말이다. 더 나아가 residual SD 는 model statement 에 나타나지 않는 점, AIC 의 부재 등을 주의할 필요가 있다.

그러나 여기서도 마찬가지로 interaction 은 significant 하게 negative 로 나타나는 것을 확인할 수 있다.

이번에는 Numerical Quadrature 를 사용해보자. 우리는 random effect 구조가 간단하기 때문에 Laplace 대신 Gauss-Hermite 방식을 사용하기로 했다. 따라서 nAGQ=25 로 설정하더라도, computation 속도에는 문제가 없을 것이다.

```
library(lme4)
```

```
## Loading required package: Matrix
```

```
modgh <- glmer(seizures ~ offset(log(timeadj)) + expind + treat + I(expind*tr
eat) +
                 (1|id), nAGQ = 25, family=poisson, data=epilo)
summary(modgh)
```

```
## Generalized linear mixed model fit by maximum likelihood (Adaptive
##   Gauss-Hermite Quadrature, nAGQ = 25) [glmerMod]
##  Family: poisson  ( log )
## Formula: seizures ~ offset(log(timeadj)) + expind + treat + I(expind *
##     treat) + (1 | id)
##    Data: epilo
##
##      AIC      BIC   logLik deviance df.resid
##    877.7    896.1   -433.9    867.7      285
##
## Scaled residuals:
##     Min      1Q  Median      3Q     Max
## -3.8724 -0.8482 -0.1722  0.5697  9.8941
##
## Random effects:
##  Groups Name        Variance Std.Dev.
##  id     (Intercept) 0.515    0.7176
## Number of obs: 290, groups:  id, 58
##
## Fixed effects:
##                     Estimate Std. Error z value Pr(>|z|)
## (Intercept)         1.035998   0.141256   7.334 2.23e-13 ***
## expind              0.111838   0.046877   2.386    0.017 *
## treat              -0.008152   0.196524  -0.041    0.967
## I(expind * treat) -0.302387   0.069714  -4.338 1.44e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Correlation of Fixed Effects:
##             (Intr) expind treat
## expind      -0.175
## treat       -0.718  0.126
## I(xpnd*trt)  0.118 -0.672 -0.173
```

→ 마찬가지로 interaction term 이 negative 로 significant 인 것을 확인할 수 있다. 방법들을 살펴보았을 때 interaction term 의 estimation 값이 대략 -0.302 로 모두 consistent 하다는 것을 확인할 수 있다. 따라서 이 값을 이용하면,

```
exp(-0.302)
```

```
## [1] 0.7393381
```

Drug 가 발작율을 26% 감소시킨다는 것을 알 수 있다.

그런데 subject SD 의 값이 drug effect 의 값보다 2 배보다 더 크다는 것을 알 수 있다. 이는 drug 로 인한 improvement 가 개인들간의 variation 보다 작다는 것을 의미한다.

Main effect term 을 해석하는 것은 interaction term 때문에 문제가 있을 수 있다. 예를 들어, treatment effect 는 baseline period 동안 response 에 대한 predicted difference 를 나타내는데, 문제는 baseline period 동안에는 treat 을 받은 사람이 아무도 없다는 점이다. 따라서 우리는 이 effect 가 통계적으로 유의하지 않다는 것을 다시금 확인할 수 있다.

우리는 또한 Bayesian approach 를 사용할 수 있다. 앞서 사용했던 stan 코드를 Poisson 식으로 변환하여 불러오고 data 를 준비해보자.

```
epilo$id[epilo$id == 59] <- 49
```

넘버링을 다시 해 준 이유는 subject 들이 연속적으로 numbering 이 되어있어야 하기 때문이다.

```
xm <- model.matrix( ~ expind + treat + I(expind*treat), epilo)
epildat <- with(epilo, list(Nobs=nrow(epilo), Nsubs=length(unique(id)),
                            Npreds = ncol(xm),
                            y = seizures,
                            subject = id,
                            x = xm,
                            offset = timeadj))
```

```
library(rstan)
```

```
## Warning: package 'rstan' was built under R version 4.0.3
```

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

```
## Warning: package 'StanHeaders' was built under R version 4.0.3
```

```
## rstan (Version 2.21.2, GitRev: 2e1f913d3ca3)

## For execution on a local, multicore CPU with excess RAM we recommend calli
ng
## options(mc.cores = parallel::detectCores()).
## To avoid recompilation of unchanged Stan programs, we recommend calling
## rstan_options(auto_write = TRUE)

## Do not specify '-march=native' in 'LOCAL_CPPFLAGS' or a Makevars file

rt <- stanc('glmmpois.stan')
sm <- stan_model(stanc_ret = rt, verbose = FALSE)
fit <- sampling(sm, data=epildat)

##
## SAMPLING FOR MODEL 'glmmpois' NOW (CHAIN 1).
## Chain 1:
## Chain 1: Gradient evaluation took 0.001 seconds
## Chain 1: 1000 transitions using 10 leapfrog steps per transition would tak
e 10 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: 4.257 seconds (Warm-up)
## Chain 1:                4.636 seconds (Sampling)
## Chain 1:                8.893 seconds (Total)
## Chain 1:
##
## SAMPLING FOR MODEL 'glmmpois' NOW (CHAIN 2).
## Chain 2:
## Chain 2: Gradient evaluation took 0 seconds
## Chain 2: 1000 transitions using 10 leapfrog steps per transition would tak
e 0 seconds.
## Chain 2: Adjust your expectations accordingly!
## Chain 2:
## Chain 2:
## Chain 2: Iteration:    1 / 2000 [  0%]  (Warmup)
```

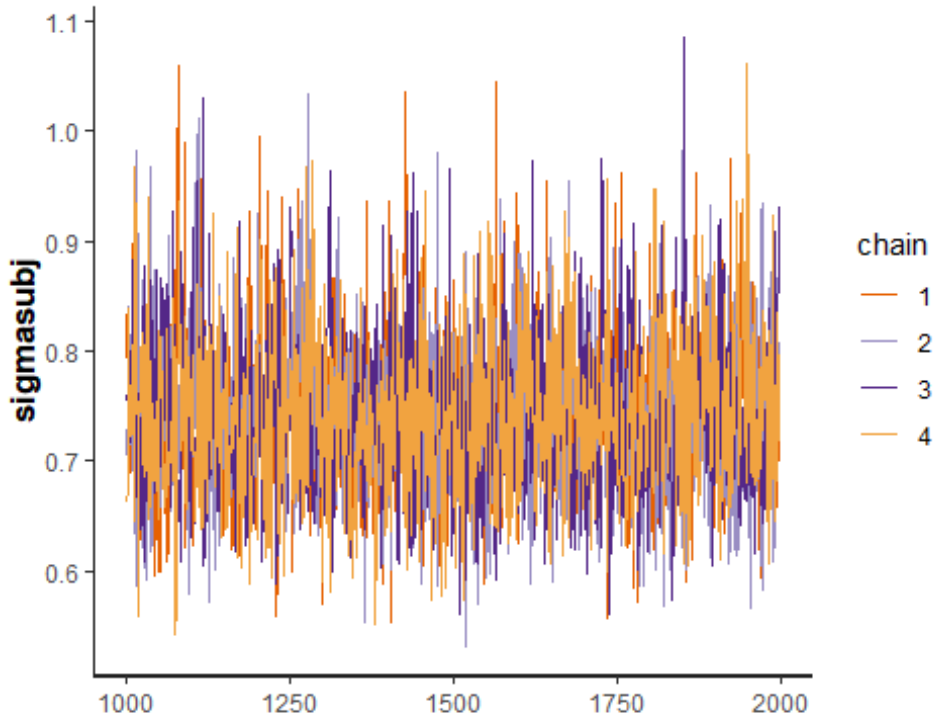
```
## Chain 2: Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 2: Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 2: Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 2: Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 2: Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 2: Iteration: 1001 / 2000 [ 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: 4.422 seconds (Warm-up)
## Chain 2:                3.804 seconds (Sampling)
## Chain 2:                8.226 seconds (Total)
## Chain 2:
##
## SAMPLING FOR MODEL 'glmmpois' NOW (CHAIN 3).
## Chain 3:
## Chain 3: Gradient evaluation took 0.001 seconds
## Chain 3: 1000 transitions using 10 leapfrog steps per transition would tak
e 10 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: 4.268 seconds (Warm-up)
## Chain 3:                3.976 seconds (Sampling)
## Chain 3:                8.244 seconds (Total)
## Chain 3:
##
## SAMPLING FOR MODEL 'glmmpois' NOW (CHAIN 4).
## Chain 4:
## Chain 4: Gradient evaluation took 0.001 seconds
## Chain 4: 1000 transitions using 10 leapfrog steps per transition would tak
e 10 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: 4.853 seconds (Warm-up)
## Chain 4:                4.116 seconds (Sampling)
## Chain 4:                8.969 seconds (Total)
## Chain 4:

## Warning: Bulk Effective Samples Size (ESS) is too low, indicating posterio
r means and medians may be unreliable.
## Running the chains for more iterations may help. See
## http://mc-stan.org/misc/warnings.html#bulk-ess

## Warning: Tail Effective Samples Size (ESS) is too low, indicating posterio
r variances and tail quantiles may be unreliable.
## Running the chains for more iterations may help. See
## http://mc-stan.org/misc/warnings.html#tail-ess

traceplot(fit, pars='sigmasubj', inc_warmup = FALSE)
```

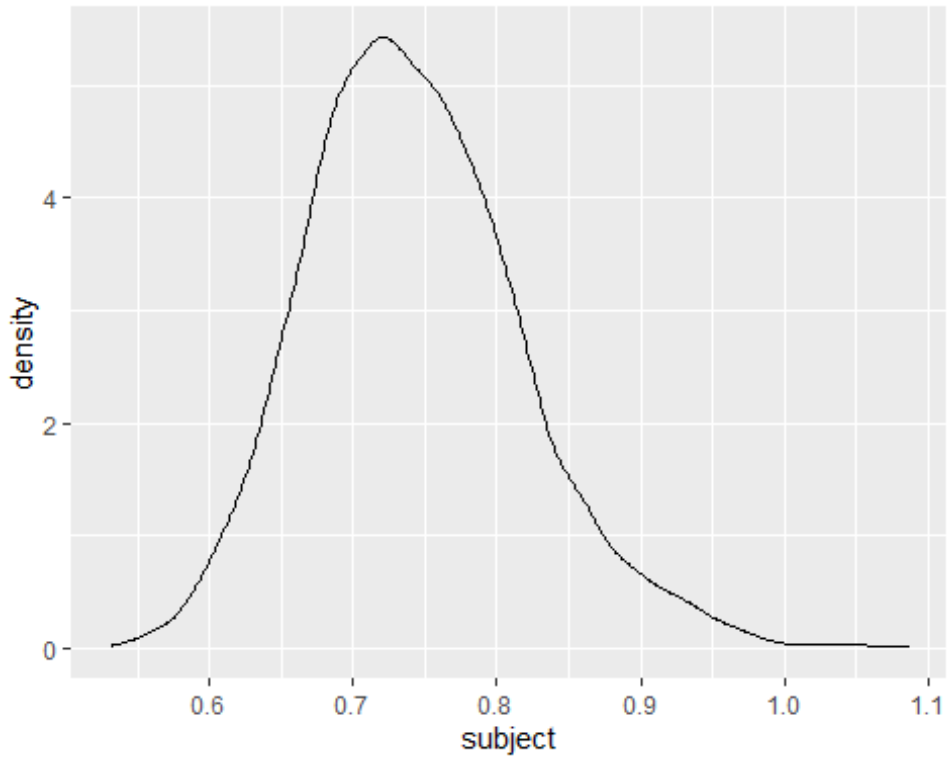


➔ 잘 converge 했음을 알 수 있다.

이번에는 Posterior density plot 을 그려보자.

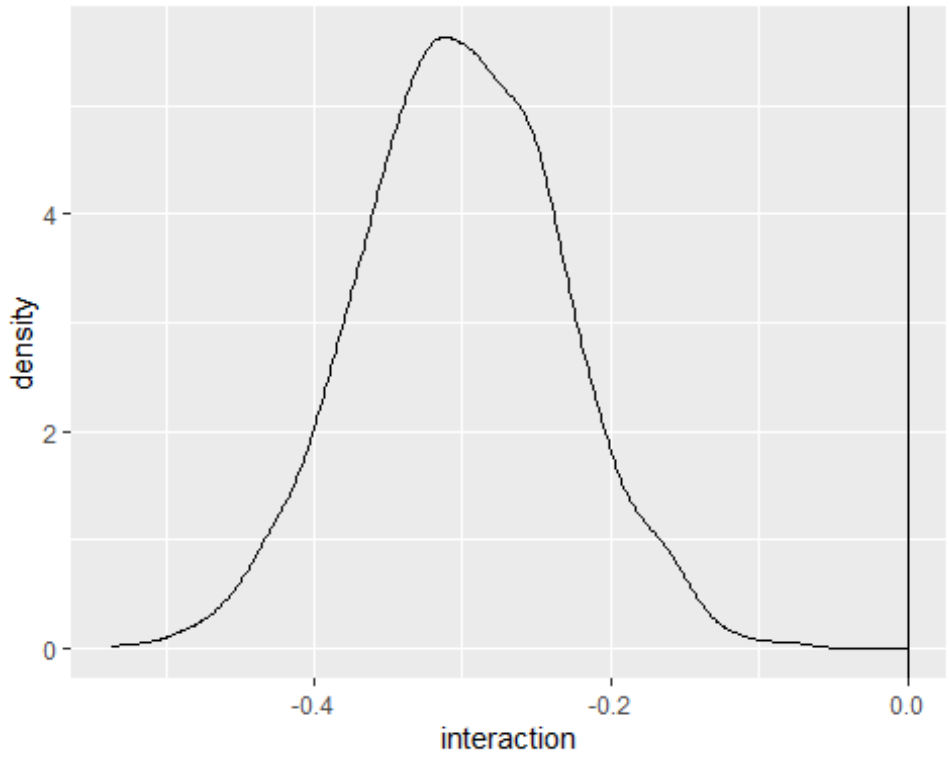
```
ipars <- data.frame(rstan::extract(fit, pars=c('sigmasubj', 'beta')))
colnames(ipars) <- c('subject', 'intercept', 'expind', 'treat','interaction')

ggplot(ipars, aes(x=subject)) + geom_density()
```



➔ Subject density 는 확실히 0 에서 벗어나 있다.

```
ggplot(ipars, aes(x=interaction)) + geom_density() + geom_vline(xintercept =
0)
```



➔ Interaction 또한 -0.3 을 중심으로 확연히 0 에서 벗어나 있다.

이제 Bayesian p-value 를 포함하여 mean 등을 나타내보자.

```
bayespval <- function(x) {p <- mean(x>0); 2*min(p, 1-p)}
smat <- apply(ipars, 2, function(x){
  c(mean(x), quantile(x, c(0.025, 0.975)), bayespval(x))
})
row.names(smat) <- c('mean', 'LCB', 'UCB', 'pvalue')
t(smat)
```

```
##                    mean         LCB        UCB pvalue
## subject       0.7424436  0.60977620  0.9118233 0.0000
## intercept     1.0589163  0.76589032  1.3633527 0.0000
## expind        0.1102354  0.01607901  0.1997727 0.0195
## treat        -0.0366894 -0.41648383  0.3360910 0.8415
## interaction -0.3011540 -0.43571383 -0.1630935 0.0000
```

➔ Posterior Mean 들이 likelihood 방식으로 구했던 estimation 값들과 거의 동일하다는 것을 확인할 수 있다.

같은 모델은 또한 INLA 를 통해 바로 fitting 될 수 있다.

```r
formula <- seizures ~ offset(log(timeadj)) + expind + treat + I(expind*treat)
 + f(id, model='iid')
library(INLA)
```

```
## Loading required package: sp
```

```
## Warning: package 'sp' was built under R version 4.0.3
```

```
## Loading required package: parallel
```

```
## Loading required package: foreach
```

```
## Warning: package 'foreach' was built under R version 4.0.3
```

```
## This is INLA_20.03.17 built 2020-11-27 06:31:09 UTC.
## See www.r-inla.org/contact-us for how to get help.
```

```r
result <- inla(formula, family='poisson', data=epilo)

sigmaalpha <- inla.tmarginal(function(x){
  1/sqrt(x)
}, result$marginals.hyperpar$'Precision for id')
restab <- sapply(result$marginals.fixed, function(x){inla.zmarginal(x, silent
 = TRUE)})
restab <- cbind(restab, inla.zmarginal(sigmaalpha, silent = TRUE))
colnames(restab) = c('mu', 'expind', 'treat', 'interaction', 'alpha')
data.frame(restab)
```

```
##                    mu     expind        treat interaction      alpha
## mean         1.036001  0.1117602 -0.008111234  -0.3023862  0.7257077
## sd          0.1431661 0.04688317    0.1991518  0.06972186 0.07192121
## quant0.025 0.7527861 0.01982029    -0.400497  -0.4393198  0.5998809
## quant0.25  0.9400873 0.07998416   -0.1418298  -0.3496021   0.674796
## quant0.5    1.035844  0.1115993 -0.008891942  -0.3025532  0.7202203
## quant0.75   1.131224  0.1432358    0.1240462  -0.2555381  0.7705279
## quant0.975   1.31581  0.2035587    0.3827135  -0.1660752  0.8821045
```

→ 마찬가지로 결과들이 앞선 결과들과 비슷하다. Interaction 의 credible interval 이 -0.44 에서 -0.17 인 것으로 보았을 때 Stan 에서와 마찬가지로 0 에서 확연히 벗어나 있음을 확인할 수 있다.