R_HW9_Binary Responses

Eom SangJun

2020 11 27

몸의 Balance 를 Response 로 갖고 surface 와 vision 등을 independent variable 로 갖는 데이터를 이용하여 분석을 진행하자.

```
library(faraway)
data(ctsib, package='faraway')
```

우선 Response 가 Binary 가 아니라 Ordinal 이기 때문에 Binary 로 변환해주자.

1 은 Balance 가 Stable 함을 나타내며, 0 은 그렇지 못함을 나타낸다.

```
ctsib$stable <- ifelse(ctsib$CTSIB==1, 1, 0)
```

Surface 와 Vision 의 상황이 주어졌을 때 단순히 기술적으로 mean response 를 나타내면 다음과 같다.

80 으로 나눠준 이유는 xtabs 는 each combination 의 value 들을 모두 더해주는데, subject 의 수는 40 이며 총 2 번씩 관측했기 때문이다.

```
xtabs(stable ~ Surface + Vision, ctsib)/80
```

```
##         Vision
## Surface closed   dome   open
##    foam 0.0000 0.0000 0.1250
##    norm 0.2125 0.2750 0.8125
```

```
library(dplyr)
```

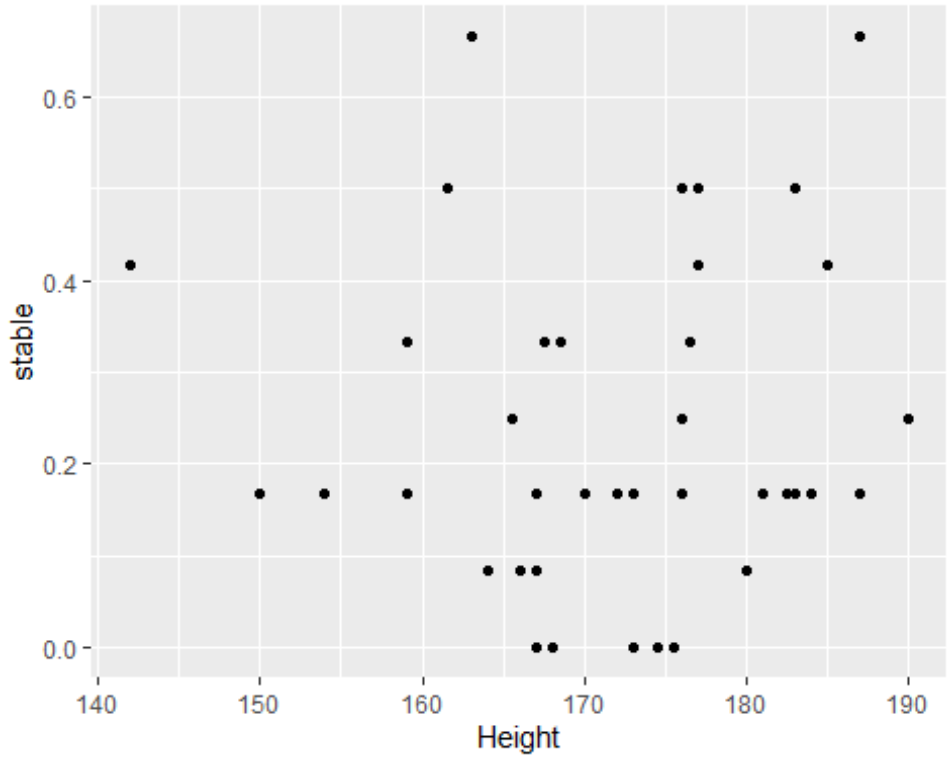
```
##
## Attaching package: 'dplyr'
```

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

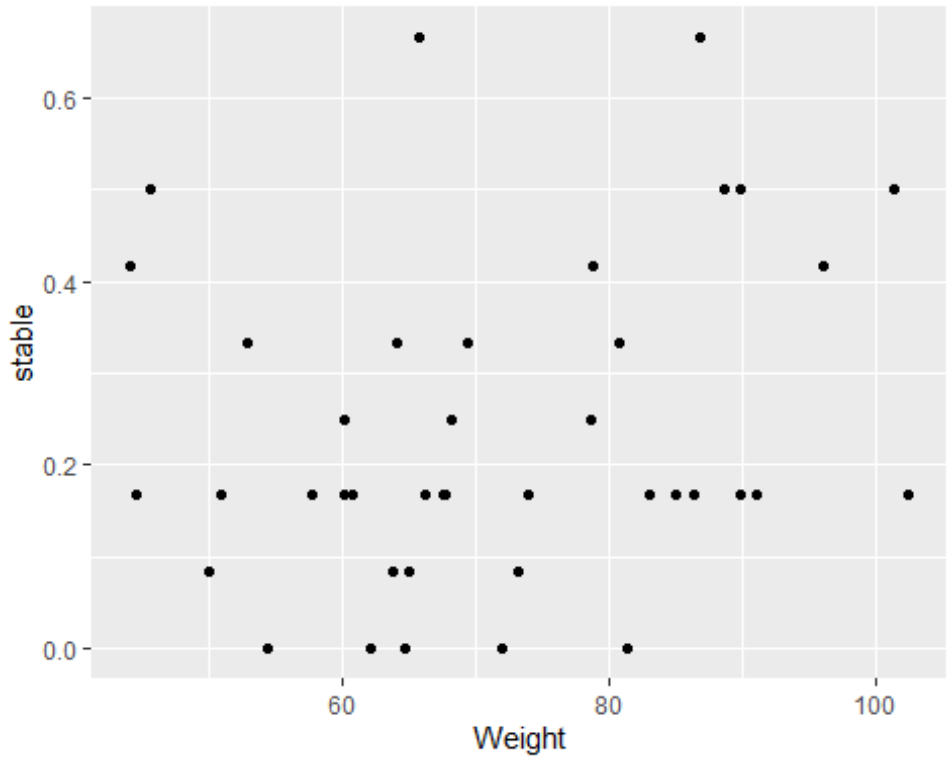
```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

subject 별로 group 화를 시켜줘서 살펴볼 수도 있다. 이 때 Response 의 값은 12 번의 관측 상황(6 개의 combination * 2 번)에서 stable 한 것의 평균이다.

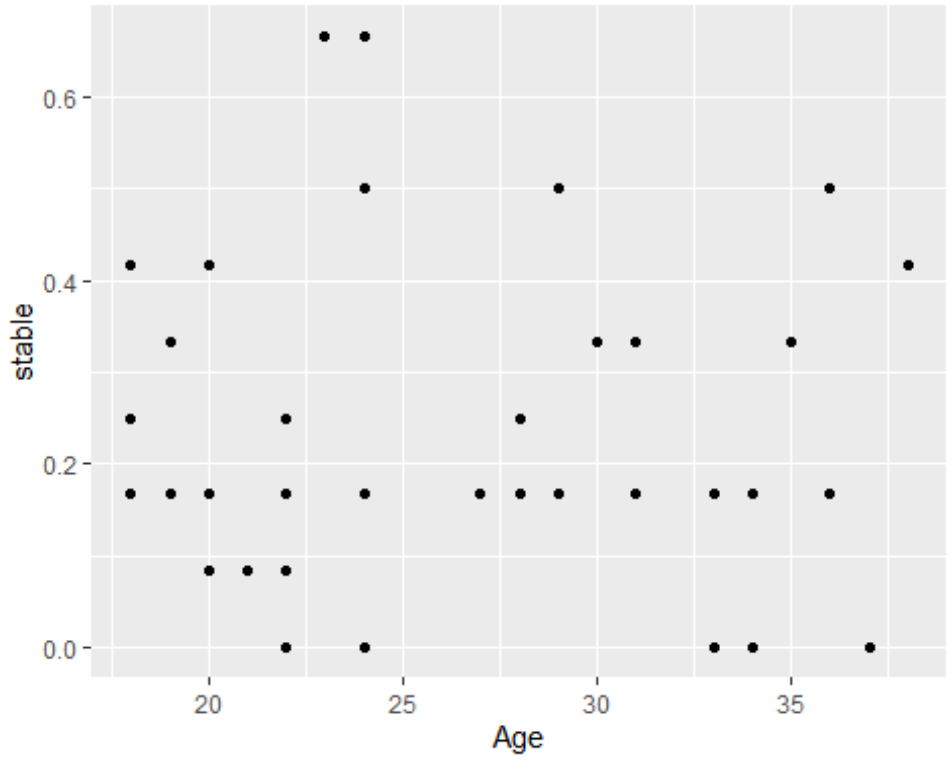
```
subsum <- ctsib %>%
  group_by(Subject) %>%
  summarise(Height=Height[1], Weight = Weight[1], stable = mean(stable), Age
= Age[1], Sex = Sex[1])

## `summarise()` ungrouping output (override with `.groups` argument)

library(ggplot2)
ggplot(subsum, aes(x=Height, y=stable)) + geom_point()
```



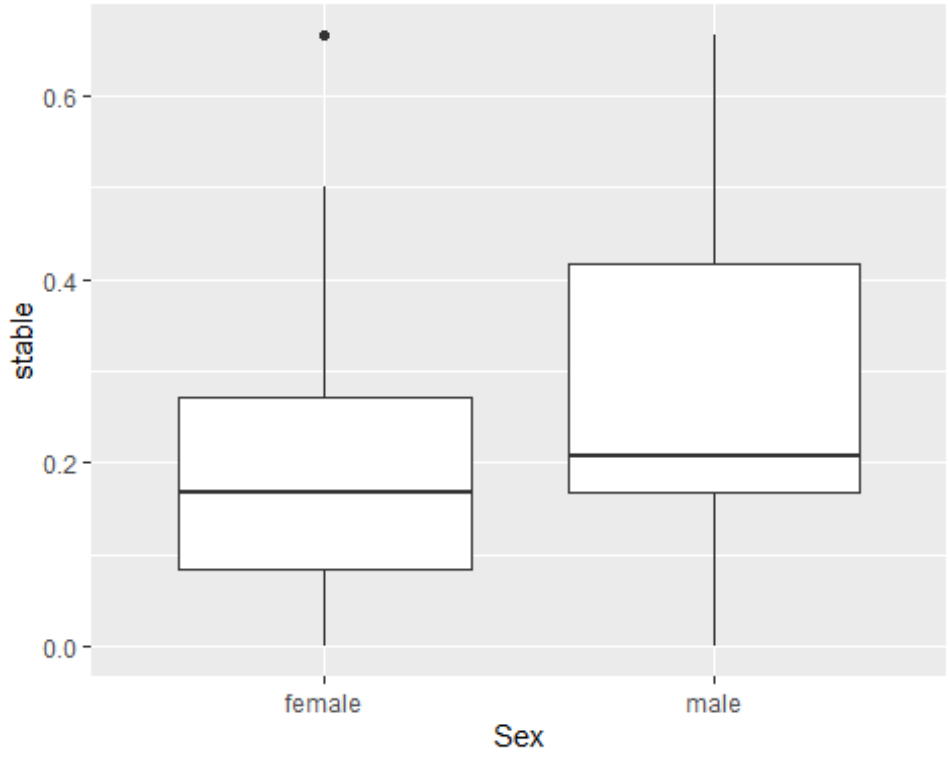

```
ggplot(subsum, aes(x=Weight, y=stable)) + geom_point()
```




```
ggplot(subsum, aes(x=Age, y=stable)) + geom_point()
```



```
ggplot(subsum, aes(x=Sex, y=stable)) + geom_boxplot()
```



subject 를 무시하고 GLM 을 fitting 시키면 다음과 같다.

```
gf <- glm(stable ~ Sex + Age + Height + Weight + Surface + Vision, binomial,
data = ctsib)
sumary(gf)
```

```
##                Estimate Std. Error z value  Pr(>|z|)
## (Intercept)  7.2774488  3.8039871  1.9131 0.0557339
## Sexmale      1.4015773  0.5162309  2.7150 0.0066272
## Age          0.0025212  0.0243073  0.1037 0.9173896
## Height      -0.0964134  0.0268369 -3.5926 0.0003274
## Weight       0.0435030  0.0180016  2.4166 0.0156652
## Surfacenorm  3.9675152  0.4471789  8.8723 < 2.2e-16
## Visiondome   0.3637528  0.3832178  0.9492 0.3425157
## Visionopen   3.1875007  0.4160007  7.6622 1.827e-14
##
## n = 480 p = 8
## Deviance = 295.20261 Null Deviance = 526.25381 (Difference = 231.05120)
```

결과를 보면, 480 개의 observation 이 나온다. 하지만 사실상 우리는 response 들이 서로 correlate 되어있는 40 개의 subject 데이터를 가지고 있을 뿐이다.

따라서 방금 위의 식은 standard error 를 과소측정하게 되고, 이로 인해 experiment effect 의 significance 를 과대평가하게 된다.

그럼 이제 subject factor 를 fixed effect 로 넣어보자.

```
gfs <- glm(stable ~ Sex + Age + Height + Weight + Surface + Vision + factor(Subject),
           binomial, data = ctsib)
```

```
## Warning: glm.fit: algorithm did not converge
```

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

그러나 이 방법 또한 식별의 문제가 있다. 이는 네 가지의 subject-specific measure 와 subject factor 를 완전히 분리할 수 없기 때문에 발생한다.

또한 이러한 문제를 해결하게 되더라도, subject factor 를 fixed effect 로 적절하게 다루기는 힘들다. 우리는 개별 subject 자체에 관심이 있는 것이 아니라

네 가지의 subject measure 가 어떻게 stability 에 영향을 미치는 가에 관심이 있는 것이다. 피험자는 우리의 target population 에서 random 으로 추출한 random sample 이라고 간주할 수 있다.

우리는 개별 subject 가 아니라 population 에 내재되어 있는 variability 를 알고 싶으며 이 variability 는 일반적으로 measurable variable 에 의해서는 설명이 되지 않는다.

R 에서 GLMM 을 fitting 하는 방법에는 여러가지가 있는데, 우선 그 중 MASS Package 의 PQL 방식을 살펴보자.

```
library(MASS)
```

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

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

```
modpql <- glmmPQL(stable ~ Sex + Age + Height + Weight + Surface + Vision,
                  random = ~1|Subject, family = binomial, data = ctsib)
```

```
## iteration 1
```

```
## iteration 2
```

```
## iteration 3
```

```
## iteration 4
```

```
## iteration 5

## iteration 6

## iteration 7
```

```
summary(modpql)
```

```
## Linear mixed-effects model fit by maximum likelihood
##  Data: ctsib
##   AIC BIC logLik
##    NA  NA     NA
##
## Random effects:
##  Formula: ~1 | Subject
##         (Intercept)  Residual
## StdDev:    3.060712 0.5906232
##
## Variance function:
##  Structure: fixed weights
##  Formula: ~invwt
## Fixed effects: stable ~ Sex + Age + Height + Weight + Surface + Vision
##                 Value Std.Error  DF   t-value p-value
## (Intercept) 15.571494 13.498304 437  1.153589  0.2493
## Sexmale      3.355340  1.752614  35  1.914478  0.0638
## Age         -0.006638  0.081959  35 -0.080992  0.9359
## Height      -0.190819  0.092023  35 -2.073601  0.0455
## Weight       0.069467  0.062857  35  1.105155  0.2766
## Surfacenorm  7.724078  0.573578 437 13.466492  0.0000
## Visiondome   0.726464  0.325933 437  2.228873  0.0263
## Visionopen   6.485257  0.543980 437 11.921876  0.0000
##  Correlation:
##             (Intr) Sexmal Age    Height Weight Srfcnr Visndm
## Sexmale      0.488
## Age         -0.164  0.110
## Height      -0.963 -0.388  0.041
## Weight       0.368 -0.374 -0.168 -0.555
## Surfacenorm  0.051  0.116  0.023 -0.114  0.055
## Visiondome  -0.003  0.011  0.004 -0.017  0.011  0.087
## Visionopen   0.056  0.125  0.026 -0.116  0.049  0.788  0.377
##
## Standardized Within-Group Residuals:
##           Min            Q1           Med            Q3           Max
## -7.3825387632 -0.2333403349 -0.0233564301 -0.0004216629  9.9310682797
##
## Number of Observations: 480
## Number of Groups: 40
```

subject effect 의 Standard Deviation 은 3.06 이다. Logistic Regression 에서 해석했던 것과 마찬가지로 이 값에 exponential 을 취해주면, 21.3 의 value 를 얻게된다.

우리는 개인들이 본래 가지고 있는 stability 간에 큰 variation 이 있다는 것을 확인할 수 있다.

또한 이 variation 은 treatment effect 에 필적할만한 크기를 가지고 있다.

residual standard deviation 의 경우 fitting process 과정에서 나온 부산물같은 것이므로, model 의 statement 에는 존재하지 않는다.

우리는 surface 와 vision effect 가 매우 강하다는 것을 확인할 수 있고, 반면 다른 요인들은 marginal 하게 유의한 p-value 를 가지고 있음을 알 수 있다.

그러나 이 추론은 linearized model 과 다소 미심쩍은 가정에 기반한 것이기 때문에 결과를 믿기는 힘들다.

더 나아가 베르누이 response 는 regression coefficient 들에 대해 biased 된 estimates 로 이어질 수 있다.

따라서 다른 estimation method 들도 확인해보는 것이 좋다.

Numerical Integration-based methods 는 lme4 package 에 내장되어 있다.

이 method 는 기본적으로 laplace approximation 을 이용한다.

```
library(lme4)
```

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

```
modlap <- glmer(stable ~ Sex + Age + Height + Weight + Surface + Vision + (1|
Subject),
                family = binomial, data = ctsib)
```

Laplace method 는 Gauss-Hermite approximation 의 특별한 경우이기 때문에, 이 approach 를 시도해보는 것이 가장 좋다.

여기서 우리는 maximum allowable number of quadrature points 로 25 를 사용했다.

```
modgh <- glmer(stable ~ Sex + Age + Height + Weight + Surface + Vision + (1|S
ubject),
               nAGQ = 25, family = binomial, data = ctsib)
```

```
## Warning in checkConv(attr(opt, "derivs"), opt$par, ctrl = control$checkCon
v, : Model failed to converge with max|grad| = 0.289247 (tol = 0.002, compone
```

```
nt 1)

## Warning in checkConv(attr(opt, "derivs"), opt$par, ctrl = control$checkCon
v, : Model is nearly unidentifiable: large eigenvalue ratio
##  - Rescale variables?

summary(modgh)

## Generalized linear mixed model fit by maximum likelihood (Adaptive
##   Gauss-Hermite Quadrature, nAGQ = 25) [glmerMod]
##  Family: binomial  ( logit )
## Formula: stable ~ Sex + Age + Height + Weight + Surface + Vision + (1 |
##     Subject)
##    Data: ctsib
##
##      AIC      BIC   logLik deviance df.resid
##    247.9    285.5   -115.0    229.9      471
##
## Scaled residuals:
##     Min      1Q  Median      3Q     Max
## -4.8841 -0.1386 -0.0197 -0.0007  4.9021
##
## Random effects:
##  Groups  Name        Variance Std.Dev.
##  Subject (Intercept) 7.194    2.682
## Number of obs: 480, groups:  Subject, 40
##
## Fixed effects:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept) 16.168969  12.717602   1.271   0.2036
## Sexmale      3.096710   1.695980   1.826   0.0679 .
## Age         -0.006669   0.076455  -0.087   0.9305
## Height      -0.192248   0.088926  -2.162   0.0306 *
## Weight       0.075156   0.059099   1.272   0.2035
## Surfacenorm  7.285354   1.055122   6.905 5.03e-12 ***
## Visiondome   0.675906   0.527362   1.282   0.2000
## Visionopen   6.088820   0.972368   6.262 3.80e-10 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Correlation of Fixed Effects:
##             (Intr) Sexmal Age    Height Weight Srfcnr Visndm
## Sexmale      0.509
## Age         -0.166  0.094
## Height      -0.960 -0.429  0.047
## Weight       0.386 -0.324 -0.169 -0.570
## Surfacenorm  0.166  0.256 -0.013 -0.281  0.147
## Visiondome   0.007  0.034  0.000 -0.044  0.027  0.113
## Visionopen   0.169  0.265 -0.008 -0.280  0.138  0.829  0.382
```

```
## convergence code: 0
## Model failed to converge with max|grad| = 0.289247 (tol = 0.002, component
 1)
## Model is nearly unidentifiable: large eigenvalue ratio
##  - Rescale variables?
```

우리는 simple random structure 를 사용했기 때문에 quardrature points 의 수를 넉넉하게 잡을 수 있는 것이다.

만약 우리가 좀 더 복잡한 모델을 사용하고 싶다면, 적절한 시간 내에 계산을 하기 위해 더 작은 숫자를 설정해주어야 할 것이다.

예를 들어, 작은 숫자에서 시작해서 estimates 가 별로 변하지 않거나 computation 의 시간이 너무 길어질 때까지 숫자를 조금씩 키워나가는 방식을 사용할 수 있다.

우리는 결과에서 다른 모델과 비교하기 위해 AIC/BIC 값을 얻을 수 있다는 것을 확인할 수 있다.

이는 PQL 에서는 얻을 수 없는데 그 이유는 PQL 은 true likelihood method 가 아니기 때문이다.

다만, PQL 에서의 parameter estimate 값들은 비슷하다는 것을 확인할 수 있다.

우리는 subject-spesific variable 들 중 significant 한 것이 하나도 없는 지를 확인하고 싶을 수 있다. 이는 anova function 을 이용해서 확인할 수 있다.

```
modgh2 <- glmer(stable ~ Surface + Vision + (1|Subject), nAGQ = 25,
                family = binomial, data = ctsib)
anova(modgh, modgh2)
```

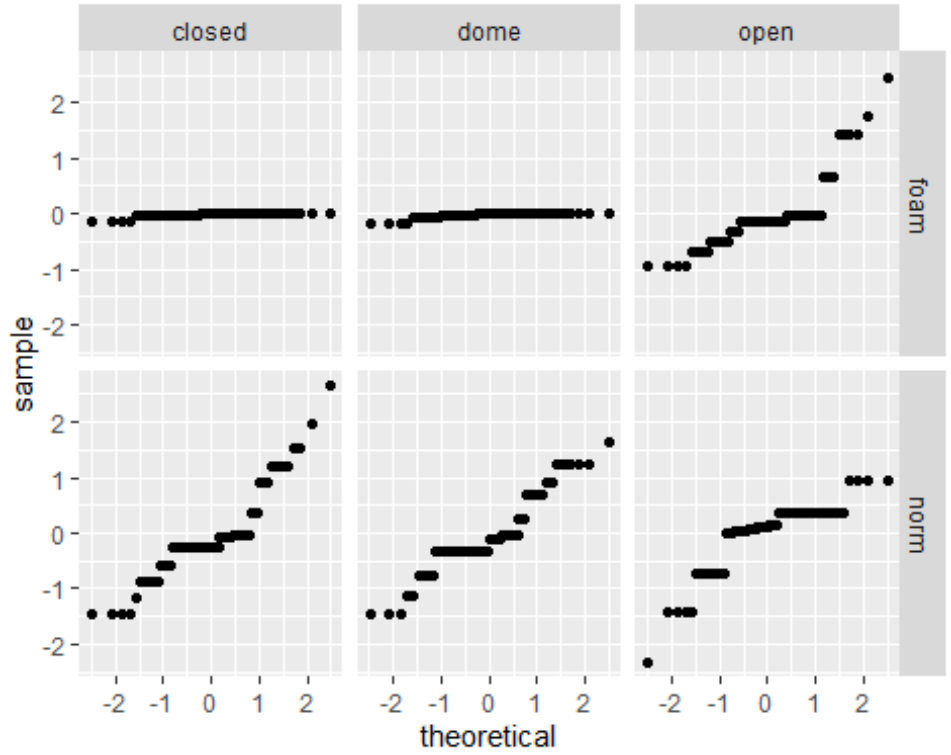
```
## Data: ctsib
## Models:
## modgh2: stable ~ Surface + Vision + (1 | Subject)
## modgh: stable ~ Sex + Age + Height + Weight + Surface + Vision + (1 |
## modgh:     Subject)
##        npar    AIC    BIC  logLik deviance  Chisq Df Pr(>Chisq)
## modgh2    5 247.30 268.17 -118.65   237.30
## modgh     9 247.93 285.49 -114.96   229.93 7.3725  4     0.1175
```

우리가 사용한 데이터가 balanced experiment 이고 size 도 작지 않기 때문에 결과를 나름 신뢰할 수 있다.

우리는 단지 treatment effect 만 fixed effect 로 보는 것이 적절하다는 것을 확인할 수 있다. (p-value 가 0.05 이상이기 때문)

이제 diagnostics 를 체크해보자.

```
dd <- fortify.merMod(modgh2)

ggplot(dd, aes(sample=.resid)) + stat_qq() + facet_grid(Surface ~ Vision)
```



➔ 처음 두 개의 residual 을 살펴보면, 0 에 가깝다는 것을 알 수 있다. 이는 처음 두 개의 조건이 보편적으로 불안정한 조건이었고 결과값 또한 대부분 그러했기 때문이다.

➔ 가장 안정적인 조건인 normal & open 조건에서는 더 큰 positive residual 은 보이지 않는 다는 것도 확인할 수 있다.

➔ 만약 이 plot 이 이단성(heteroscedascity)을 나타낸다고 해석한다면, 그것은 잘못된 것일 수도 있다. 왜냐하면, 차이에 대해 더 신빙성 있는 설명들이 있기 때문이다.

우리는 INLA package 를 이용해서 Bayesian approach 를 이러한 모델들을 fitting 하는 데에 사용할 수 있다. 설명의 편의성을 위해 surface 와 vision 만을 fixed effect 라고 가정하자. 기본적으로 noninformative prior 를 사용할 것이다.

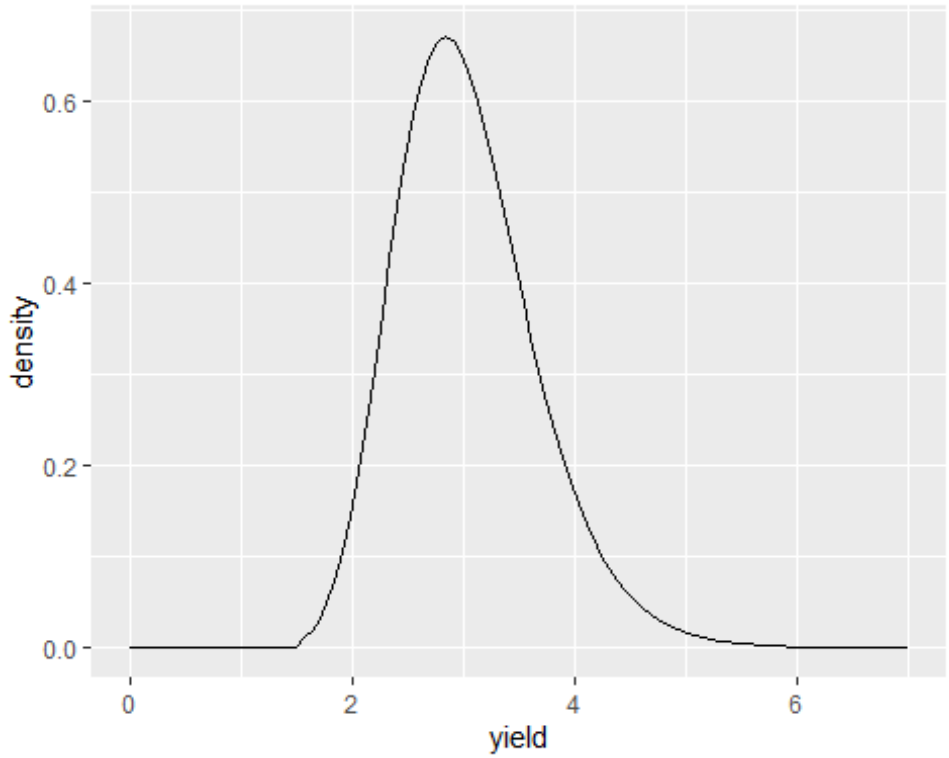
```
library(INLA)

## 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.
```

```
formula <- stable ~ Surface + Vision + f(Subject, model='iid')
result <- inla(formula, family = 'binomial', data = ctsib)
```

subject random variable 에 대해 standard error 를 구해보자.

```
sigmaalpha <- inla.tmarginal(function(x){1/sqrt(x)}, result$marginals.hyperpa
r$`Precision for Subject`)

x <- seq(0, 7, length.out = 100)
sdf <- data.frame(yield = x, density = inla.dmarginal(x, sigmaalpha))
ggplot(sdf, aes(x=yield, y=density)) + geom_line()
```



➔ Subject Standard Deviation 의 posterior density

➔ 분포가 확실히 0 에서 떨어져 있기 때문에 subject effect 가 확실히 있다는 것을 알 수 있다. 다만, effect 의 size 에 대해서는 아직 확실하지 않다.

우리는 posterior 들의 numerical summary 에 대해서도 확인해볼 수 있다.

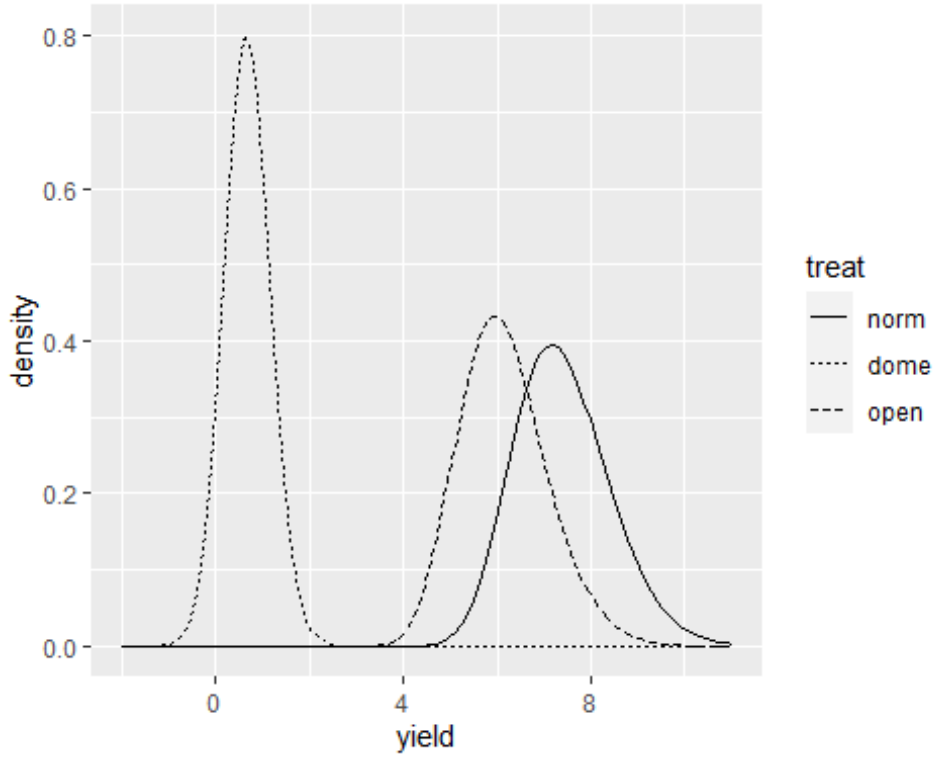
```
restab <- sapply(result$marginals.fixed, function(x){inla.zmarginal(x, silent
=TRUE)})
restab <- cbind(restab, inla.zmarginal(sigmaalpha, silent=TRUE))
colnames(restab) = c('mu', 'norm', 'dome', 'open', 'alpha')
data.frame(restab)
```

```
##                    mu     norm       dome      open     alpha
## mean       -10.33068 7.385507  0.6661077  6.149311   3.02545
## sd          1.498667 1.024956  0.4995412 0.9395766 0.6291306
## quant0.025 -13.59668 5.579706 -0.2964232   4.50383  1.977447
## quant0.25  -11.27352 6.656569  0.3247137  5.481381  2.576283
## quant0.5   -10.21621 7.312065  0.6579071   6.07801  2.959742
## quant0.75  -9.269399 8.033124  0.9969314  6.738514  3.401821
## quant0.975 -7.746339 9.581812   1.662586  8.175024  4.442759
```

Posterior mean 들을 확인해보면, glmer-based 의 fit 들과 비슷하다는 것을 확인할 수 있다.

Plot 을 그려보자.

```
x <- seq(-2, 11, length.out = 100)
rden <- sapply(result$marginals.fixed, function(y){inla.dmarginal(x, y)})[,-1]
ddf <- data.frame(yield = rep(x,3), density = as.vector(rden),
                  treat = gl(3, 100, labels = c('norm', 'dome', 'open')))
ggplot(ddf, aes(x=yield, y=density, linetype = treat)) + geom_line()
```



➔ Norm level 과 open level 은 확연히 각각의 reference level 과는 차이를 보인다는 것을 알 수 있다. (density 가 0 에서 멀리 있다.)

➔ 반면 density 가 0 에 겹치는 것을 보았을 때 dome 과 closed levels of vision 간에는 차이가 많지 않을 수 있다.

Bayesian P-value 의 값은 다음과 같다. 2 를 곱해주는 이유는 양측 검정이므로.

```
2*inla.pmarginal(0, result$marginals.fixed$Visiondome)
```

```
## [1] 0.1796335
```

물론 Bayesian P-value 는 원래의 p-value 와 동일한 의미를 지니지는 않는다. 다만, 이는 posterior density 가 얼마나 0 과 관련이 있는 지를 나타낸다. P-value 값을 보았을 때 level 간의 차이가 많지 않다는 것을 확인할 수 있다.

우선 model matrix of fixed effects 를 먼저 만들어주자.

```
xm <- model.matrix(~ Sex + Age + Height + Weight + Surface + Vision, ctsib)
stabledat <- with(ctsib, list(Nobs = nrow(ctsib),
                              Nsubs = length(unique(ctsib$Subject)),
```

```
                                Npreds = ncol(xm),
                                y=stable,
                                subject = Subject,
                                x = xm))

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
```

이제 앞서 만든 것들을 이용해서 MCMC Sampling 을 진행하자.

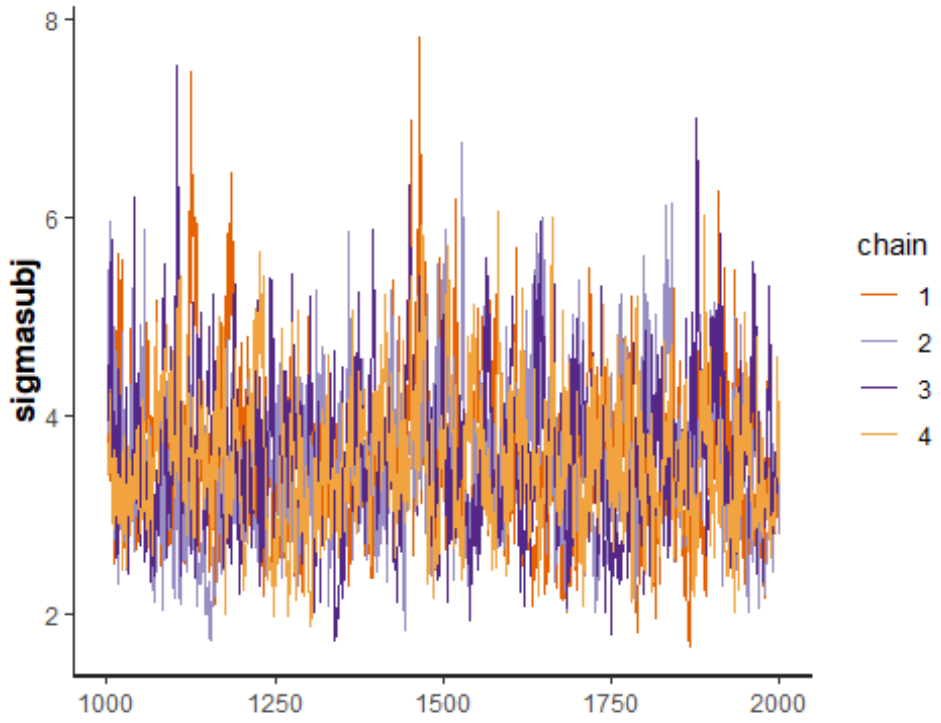
```
rt <- stanc('glmmbin.stan')
sm <- stan_model(stanc_ret = rt, verbose = FALSE)
fit <- sampling(sm, data=stabledat)

##
## SAMPLING FOR MODEL 'glmmbin' NOW (CHAIN 1).
## Chain 1:
## Chain 1: Gradient evaluation took 0 seconds
## Chain 1: 1000 transitions using 10 leapfrog steps per transition would tak
e 0 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: 89.104 seconds (Warm-up)
## Chain 1:                108.884 seconds (Sampling)
## Chain 1:                197.988 seconds (Total)
## Chain 1:
##
## SAMPLING FOR MODEL 'glmmbin' 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: 95.218 seconds (Warm-up)
## Chain 2:                105.639 seconds (Sampling)
## Chain 2:                200.857 seconds (Total)
## Chain 2:
##
## SAMPLING FOR MODEL 'glmmbin' 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: 95.109 seconds (Warm-up)
## Chain 3:                113.57 seconds (Sampling)
## Chain 3:                208.679 seconds (Total)
## Chain 3:
##
## SAMPLING FOR MODEL 'glmmbin' NOW (CHAIN 4).
## Chain 4:
## Chain 4: Gradient evaluation took 0 seconds
## Chain 4: 1000 transitions using 10 leapfrog steps per transition would tak
e 0 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: 87.364 seconds (Warm-up)
## Chain 4:                105.862 seconds (Sampling)
## Chain 4:                193.226 seconds (Total)
## Chain 4:

## Warning: There were 179 transitions after warmup that exceeded the maximum
 treedepth. Increase max_treedepth above 10. See
## http://mc-stan.org/misc/warnings.html#maximum-treedepth-exceeded

## Warning: Examine the pairs() plot to diagnose sampling problems
```

제대로 됐는지, plot 을 그려서 확인해보자. Subject 의 standard deviation 이 문제가 있을 가능성이 제일 크므로 이를 확인해보자.

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



➔ 문제 없이 잘 됐다는 것을 확인할 수 있다.

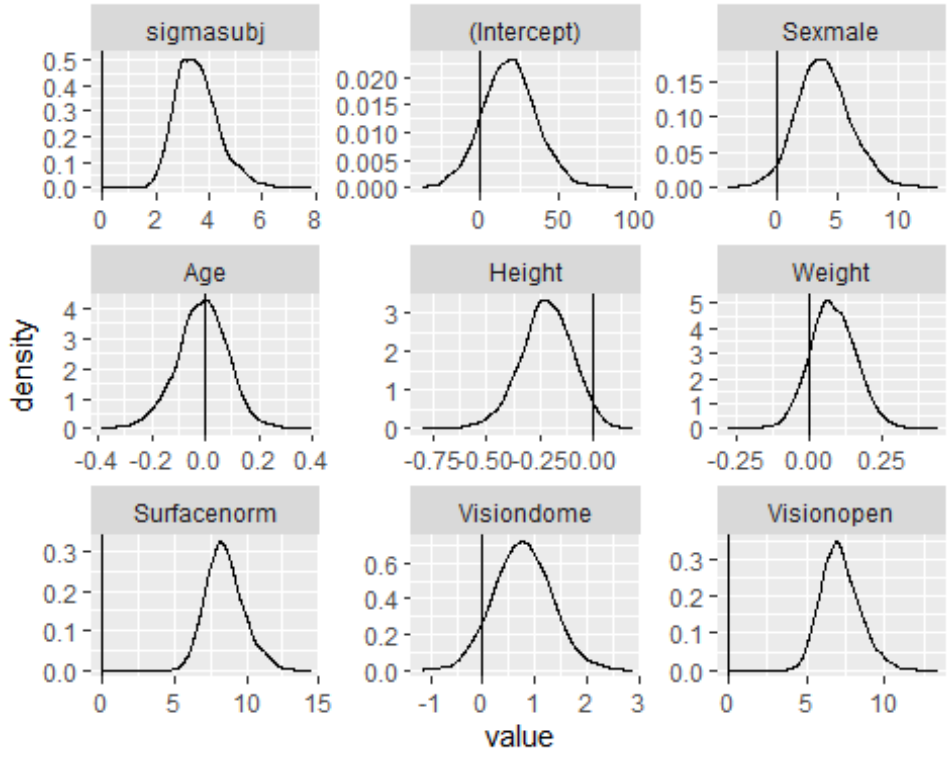
Parameter 들의 summary 는 다음과 같다.

```
print(fit, pars=c('sigmasubj', 'beta'))
```

```
## Inference for Stan model: glmmbin.
## 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% n_eff Rhat
## sigmasubj  3.55    0.03  0.79   2.23  2.97  3.47  4.01  5.35   584 1.01
## beta[1]   18.79    0.65 17.57 -15.61  7.03 18.46 29.96 54.67   720 1.02
## beta[2]    3.84    0.08  2.24  -0.40  2.35  3.75  5.25  8.44   795 1.01
## beta[3]   -0.01    0.00  0.10  -0.21 -0.07 -0.01  0.05  0.17  1067 1.00
## beta[4]   -0.22    0.00  0.12  -0.48 -0.30 -0.22 -0.14  0.00   761 1.01
## beta[5]    0.08    0.00  0.08  -0.07  0.03  0.08  0.14  0.24   956 1.01
## beta[6]    8.50    0.05  1.31   6.24  7.59  8.40  9.28 11.38   648 1.01
## beta[7]    0.77    0.01  0.56  -0.31  0.40  0.76  1.14  1.88  4071 1.00
## beta[8]    7.18    0.05  1.23   5.06  6.32  7.07  7.94  9.88   651 1.01
##
## Samples were drawn using NUTS(diag_e) at Fri Nov 27 20:30:23 2020.
## 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).
```

이제 Posterior Distribution 은 검사해보자. 우선 우리가 관심이 있는 parameter 들을 빼내고, 편의대로 이름을 재지정해주자. 다음으로 reshape2 package 를 이용해 데이터들을 posterior distribution 을 그리기 좋은 형태로 rearrange 해주자.

```
ipars <- data.frame(extract(fit, pars=c('sigmasubj', 'beta')))
colnames(ipars)[-1] <- colnames(xm)
library(reshape2)

## Warning: package 'reshape2' was built under R version 4.0.3

rdf <- melt(ipars)

## No id variables; using all as measure variables

ggplot(rdf, aes(x=value)) +
  geom_density() +
  facet_wrap(~ variable, scales='free') +
  geom_vline(xintercept = 0)
```



➔ Subject 와 관련된 effect 들은 주로 0 과 density 가 걸쳐있는 것을 확인할 수 있다.

추가적으로 어떤 subject 가 제일 불안정하고 어떤 subject 가 제일 안정적인지도 확인할 수 있다.

```r
ppars <- data.frame(extract(fit, pars='subeff'))
sort(colMeans(ppars))
```

```
##    subeff.3   subeff.38   subeff.37   subeff.14   subeff.30    subeff.4
## -6.57938610 -4.82150021 -4.47547132 -4.05165302 -3.66127947 -3.43369993
##   subeff.11   subeff.40   subeff.34    subeff.5   subeff.23    subeff.9
## -2.97638013 -2.66503909 -2.48099257 -2.23557284 -2.12485444 -1.50398449
##    subeff.8   subeff.12    subeff.7   subeff.24   subeff.22   subeff.13
## -1.38981666 -1.33436845 -1.29121035 -1.16487559 -1.09751163 -0.63414170
##    subeff.2   subeff.28   subeff.20    subeff.1   subeff.36   subeff.39
## -0.49738711 -0.09109082  0.50215053  0.50521180  0.57373272  0.71962793
##   subeff.18   subeff.10   subeff.26   subeff.16   subeff.15    subeff.6
##  0.73721407  0.88500250  0.90674242  0.91173349  1.66871047  1.87451407
##   subeff.35   subeff.19   subeff.31   subeff.32   subeff.33   subeff.21
##  1.94065848  2.41372850  2.41872168  2.46094628  2.99757572  3.31336022
##   subeff.17   subeff.29   subeff.25   subeff.27
##  3.41475536  5.84166402  6.66627662  6.83096481
```

3 번 subject 가 제일 불안정하고 27 번이 제일 안정적이다.