04: Generating unaggregated fake data

Shravan Vasishth / Bruno Nicenboim

25-29 March 2019

Generating fake data for repeated measures designs

Consider again the reading time data from the experiment by Grodner and Gibson, 2005. This is the data from their Experiment 1. You can download the paper from **here**.

Recall that in this paper, we are interested in the reading time differences between object and subject relatives at the relative clause verb. The expectation from theory is that object relatives (objgap) have longer reading times than subject relatives (subjgap). The explanation for the longer reading times in objgap vs subjgap lies in working memory constraints: it is more difficult to figure out who did what to whom in object relatives than subject relatives.

First, load the data-set provided, and do the preprocessing shown. This gives us the relevant data.

```
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
```

```
gg05e1 <- read.table("data/GrodnerGibson2005E1.csv",sep=",", header=T)
gge1 <- gg05e1 %>% filter(item != 0)

gge1 <- gge1 %>% mutate(word_positionnew = ifelse(item != 15 & word_position > 10,
                                                  word_position-1, word_position))
#there is a mistake in the coding of word position,
#all items but 15 have regions 10 and higher coded
#as words 11 and higher

## get data from relative clause verb:
gge1crit <- subset(gge1, ( condition == "objgap" & word_position == 6 ) |
            ( condition == "subjgap" & word_position == 4 ))
gge1crit<-gge1crit[,c(1,2,3,6)]
head(gge1crit)
```

```
##    subject item condition rawRT
## 6        1    1    objgap   320
## 19       1    2   subjgap   424
## 34       1    3    objgap   309
## 49       1    4   subjgap   274
## 68       1    5    objgap   333
## 80       1    6   subjgap   266
```

```
gge1crit$so<-ifelse(gge1crit$condition=="objgap",1,-1)
```

Each of the 42 participants see multiple (eight) instances of subject and object relatives. So each subject delivers 16 data points in total.

```
xtabs(~subject+condition,
          gge1crit)
```

```
##        condition
## subject objgap subjgap
##      1       8       8
##      2       8       8
##      3       8       8
##      4       8       8
##      5       8       8
##      6       8       8
##      7       8       8
##      8       8       8
##      9       8       8
##      10      8       8
##      11      8       8
##      12      8       8
##      13      8       8
##      14      8       8
##      15      8       8
##      16      8       8
##      17      8       8
##      18      8       8
##      19      8       8
##      20      8       8
##      21      8       8
##      22      8       8
##      23      8       8
##      24      8       8
##      25      8       8
##      26      8       8
##      27      8       8
##      28      8       8
##      29      8       8
##      30      8       8
##      31      8       8
##      32      8       8
##      33      8       8
##      34      8       8
##      35      8       8
##      36      8       8
##      37      8       8
##      38      8       8
##      39      8       8
##      40      8       8
##      41      8       8
##      42      8       8
```

So, from each participant, we have **repeated** measures.

Aggregating subject data

It is common in psycholinguistics and psychology to aggregate such data by subjects, collapsing the data by subject so that we have *one* data point per condition per subject (instead of eight):

```
gge1bysubj<-aggregate(rawRT~subject+condition,mean,
                      data=gge1crit)
```

Look at the data:

```
head(gge1bysubj)
```

```
##   subject condition   rawRT
## 1       1    objgap 335.000
## 2       2    objgap 419.375
## 3       3    objgap 247.250
## 4       4    objgap 616.375
## 5       5    objgap 395.750
## 6       6    objgap 330.250
```

```
# one row for each subject for each condition:
dim(gge1bysubj)
```

```
## [1] 84  3
```

```
xtabs(~subject+condition,
      gge1bysubj)
```

```
##        condition
## subject objgap subjgap
##      1       1       1
##      2       1       1
##      3       1       1
##      4       1       1
##      5       1       1
##      6       1       1
##      7       1       1
##      8       1       1
##      9       1       1
##      10      1       1
##      11      1       1
##      12      1       1
##      13      1       1
##      14      1       1
##      15      1       1
##      16      1       1
##      17      1       1
##      18      1       1
##      19      1       1
##      20      1       1
##      21      1       1
##      22      1       1
##      23      1       1
##      24      1       1
##      25      1       1
##      26      1       1
##      27      1       1
```

```
##      28      1       1
##      29      1       1
##      30      1       1
##      31      1       1
##      32      1       1
##      33      1       1
##      34      1       1
##      35      1       1
##      36      1       1
##      37      1       1
##      38      1       1
##      39      1       1
##      40      1       1
##      41      1       1
##      42      1       1
```

Note that

- Each subject's responses are assumed to be **independent** of the others' (a debatable assumption!)
- We have *two* data points from each subject: one for objgap and one for subjgap. The responses from each subject are therefore *repeated* measures, hence correlated (not independent. How to generate fake data for repeated measures data?

Generating fake data for repeated measures aggregated data

```
gge1bysubj$so<-ifelse(gge1bysubj$condition=="objgap",1,-1)
nsubj<-length(unique(gge1bysubj$subject))
N<-dim(gge1bysubj)[1]
rtfake<-rep(NA,N)
beta0<-6
beta1<-0.01
so<-gge1bysubj$so
u <- rnorm(nsubj,mean=0,sd=.2)
sigma<-.5
for(i in 1:N){
  rtfake[i] <-  rlnorm(1,beta0 + beta1*so[i] + u[gge1bysubj[i,]$subject],sigma)
}
gge1bysubj$rtfake <- rtfake
```

Now we are in a position to analyze fake data just like we analyzed real data:

```
library(brms)
```

```
## Loading required package: Rcpp
```

```
## Loading required package: ggplot2
```

```
## Loading 'brms' package (version 2.5.0). Useful instructions
## can be found by typing help('brms'). A more detailed introduction
## to the package is available through vignette('brms_overview').
## Run theme_set(theme_default()) to use the default bayesplot theme.
```

```
priors <- c(set_prior("normal(0, 10)", class = "Intercept"),
                      set_prior("normal(0, 1)", class = "b",
                                coef = "so"),
```

```
                      set_prior("normal(0, 1)", class = "sd"),
                      set_prior("normal(0, 1)", class = "sigma"))
```

With real data:

```
## real data:
m_gg<-brm(rawRT~so + (1|subject),gge1bysubj,family=lognormal(),
    prior=priors,control = list(adapt_delta = 0.99,max_treedepth=15))
summary(m_gg)
```

With fake data:

```
## real data:
m_ggfake<-brm(rtfake~so + (1|subject),
              gge1bysubj,family=lognormal(),
    prior=priors,control= list(adapt_delta = 0.99,max_treedepth=15))
summary(m_ggfake)
```

Next, we will generate fake data **assuming no aggregation**.

Generating fake data assuming no aggregation

Recall that the maximal model we would fit for these data is as follows. We assume that we have j subjects and k items. y is the dependent measure, and x is the predictor (it's called ''so" in our case).

$$y_{kj} = \alpha + u_{0j} + w_{0k} + (\beta + u_{1j} + w_{1k}) * x_{kj} + \varepsilon_{kj} \tag{1}$$

where $\varepsilon_{kj} \sim Normal(0, \sigma)$ and

$$\Sigma_u = \begin{pmatrix} \sigma^2_{u0} & \rho_u \sigma_{u0} \sigma_{u1} \\ \rho_u \sigma_{u0} \sigma_{u1} & \sigma^2_{u1} \end{pmatrix} \quad \Sigma_w = \begin{pmatrix} \sigma^2_{w0} & \rho_w \sigma_{w0} \sigma_{w1} \\ \rho_w \sigma_{w0} \sigma_{w1} & \sigma^2_{w1} \end{pmatrix} \tag{2}$$

$$\begin{pmatrix} u_0 \\ u_1 \end{pmatrix} \sim \mathcal{N}\left(\begin{pmatrix} 0 \\ 0 \end{pmatrix}, \Sigma_u \right), \quad \begin{pmatrix} w_0 \\ w_1 \end{pmatrix} \sim \mathcal{N}\left(\begin{pmatrix} 0 \\ 0 \end{pmatrix}, \Sigma_w \right) \tag{3}$$

```
## adjustments to intercepts and slopes by subject, no correlation:
## unaggregated data:
head(gge1crit)
```

```
##    subject item condition rawRT so
## 6        1    1    objgap   320  1
## 19       1    2   subjgap   424 -1
## 34       1    3    objgap   309  1
## 49       1    4   subjgap   274 -1
## 68       1    5    objgap   333  1
## 80       1    6   subjgap   266 -1
```

```
## number of rows:
(N<-dim(gge1crit)[1])
```

```
## [1] 672
```

```
library(MASS)
```

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

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

```
## just making up some sd's here for now:
sigma_u0<-.5
sigma_u1<- .2
sigma_w0<- .2
sigma_w1<- .2

rho_u<-rho_w<-0 # uncorrelated intercepts and slopes

Sigma_u<-matrix(c(sigma_u0^2,sigma_u0*sigma_u1*rho_u,
                  sigma_u0*sigma_u1*rho_u,sigma_u1^2),
                ncol=2)
Sigma_u
```

```
##      [,1] [,2]
## [1,] 0.25 0.00
## [2,] 0.00 0.04
```

```
u<-mvrnorm(42,c(0,0),Sigma_u)

Sigma_w<-matrix(c(sigma_w0^2,sigma_w0*sigma_w1*rho_w,
                  sigma_w0*sigma_w1*rho_w,sigma_w1^2),
                ncol=2)
Sigma_w
```

```
##      [,1] [,2]
## [1,] 0.04 0.00
## [2,] 0.00 0.04
```

```
w<-mvrnorm(16,c(0,0),Sigma_w)

u[1,1] # first subject's intercept adjustment
```

```
## [1] -0.05131294
```

```
u[1,2] # first subject's slope adjustment
```

```
## [1] 0.1520017
```

```
u[2,1] # second subject's intercept adjustment
```

```
## [1] -0.7509992
```

```
u[2,2] # second subject's slope adjustment
```

```
## [1] -0.1004808
```

```
w[1,1] # first item's intercept adjustment
```

```
## [1] -0.1773309
```

```
w[1,2] # first item's slope adjustment
```

```
## [1] -0.3157483
```

```
w[2,1] # second item's intercept adjustment
```

```
## [1] 0.07826965
```

```
w[2,2] # second item's slope adjustment
```

```
## [1] 0.1800446
```

```
for(i in 1:N){
  rtfake[i] <- rlnorm(1,beta0 +
                u[gge1crit[i,]$subject,1] +
                w[gge1crit[i,]$item,1] +
                (beta1+
                   u[gge1crit[i,]$subject,2] +
                   w[gge1crit[i,]$item,2])*gge1crit[i,]$so,sigma)
}

gge1crit$rtfake <- rtfake
```

Now we can fit a maximal model to the fake data:

```
priorsmax <- c(set_prior("normal(0, 10)", class = "Intercept"),
                     set_prior("normal(0, 1)", class = "b",
                               coef = "so"),
                     set_prior("normal(0, 1)", class = "sd"),
                     set_prior("normal(0, 1)", class = "sigma"),
                     set_prior("lkj(2)", class="cor"))
```

With real unaggregated data:

```
m_gg2<-brm(rawRT~so + (1+so|subject)+(1+so|item),
               gge1crit,
    family=lognormal(),
    prior=priorsmax,control= list(adapt_delta = 0.99,max_treedepth=15))
summary(m_gg2)
```

With fake unaggregated data:

```
## real data:
m_ggfake2<-brm(rtfake~so + (1+so|subject)+(1+so|item),
               gge1crit,
    family=lognormal(),
    prior=priorsmax,
    control= list(adapt_delta = 0.99,max_treedepth=15))
summary(m_ggfake2)
```

We can write a function to generate fake data, using the Grodner and Gibson data frame as a starting point:

```
summary(m_gg2)
```

```
##  Family: lognormal
##   Links: mu = identity; sigma = identity
## Formula: rawRT ~ so + (1 + so | subject) + (1 + so | item)
##    Data: gge1crit (Number of observations: 672)
## Samples: 4 chains, each with iter = 2000; warmup = 1000; thin = 1;
##          total post-warmup samples = 4000
##
## Group-Level Effects:
```

```
## ~item (Number of levels: 16)
##                   Estimate Est.Error l-95% CI u-95% CI Eff.Sample Rhat
## sd(Intercept)         0.04      0.02     0.00     0.09       1335 1.00
## sd(so)                0.04      0.02     0.00     0.09       1114 1.00
## cor(Intercept,so)     0.28      0.41    -0.64     0.91       2167 1.00
##
## ~subject (Number of levels: 42)
##                   Estimate Est.Error l-95% CI u-95% CI Eff.Sample Rhat
## sd(Intercept)         0.33      0.04     0.26     0.42        827 1.00
## sd(so)                0.11      0.02     0.08     0.16       1595 1.00
## cor(Intercept,so)     0.51      0.17     0.13     0.78       2010 1.00
##
## Population-Level Effects:
##           Estimate Est.Error l-95% CI u-95% CI Eff.Sample Rhat
## Intercept     5.88      0.05     5.78     5.99        464 1.01
## so            0.06      0.03     0.01     0.11       1442 1.00
##
## Family Specific Parameters:
##       Estimate Est.Error l-95% CI u-95% CI Eff.Sample Rhat
## sigma     0.36      0.01     0.34     0.38       3202 1.00
##
## Samples were drawn using sampling(NUTS). For each parameter, Eff.Sample
## is a crude measure of effective sample size, and Rhat is the potential
## scale reduction factor on split chains (at convergence, Rhat = 1).
```

Here is a function that generates fake data:

```
genfake<-function(dat=gge1crit,
                  nsubj=42,nitem=16,
                  beta0=5.88,beta1=0.06,
                  sigma_u0=.33,
                  sigma_u1=.11,
                  sigma_w0=.04,
                  sigma_w1=.04,
                  rho_u=0.51,
                  rho_w=.29,
                  N=dim(gge1crit)[1]){
  Sigma_u<-matrix(c(sigma_u0^2,sigma_u0*sigma_u1*rho_u,
                    sigma_u0*sigma_u1*rho_u,sigma_u1^2),
                  ncol=2)
  ## generate by subject intercepts and slopes:
  u<-mvrnorm(nsubj,c(0,0),Sigma_u)
  Sigma_w<-matrix(c(sigma_w0^2,sigma_w0*sigma_w1*rho_w,
                    sigma_w0*sigma_w1*rho_w,sigma_w1^2),
                  ncol=2)
  ## generate by item intercepts and slopes:
  w<-mvrnorm(nitem,c(0,0),Sigma_w)

  ## generate log normally distributed data:
    for(i in 1:N){
    rtfake[i] <- rlnorm(1,beta0 +
      u[dat[i,]$subject,1] +
      w[dat[i,]$item,1] +
      (beta1+
         u[dat[i,]$subject,2] +
```

```
        w[dat[i,]$item,2])*dat[i,]$so,
      sigma)
    }
  rtfake
}

## generate and save 50 data-sets:
matrixfakerts<-matrix(rep(NA,672*50),ncol=50)

for(i in 1:50){
matrixfakerts[,i]<-genfake()
}

## true discovery rate:
tval<-rep(NA,50)
fakedesign<-gge1crit[,c(1,2,3,5)]
library(lme4)
```

```
## Warning: package 'lme4' was built under R version 3.5.2
```

```
for(i in 1:50){
  fakedat<-data.frame(fakedesign,fakert=matrixfakerts[,i])
  ## frequentist version of Bayesian hierarchical model:
  m<-lmer(fakert~so+(1+so||subject)+(1+so||item),fakedat)
  if(!isSingular(m)){
  tval[i]<-summary(m)$coefficients[2,3]
  }
}
```

```
## Warning in checkConv(attr(opt, "derivs"), opt$par, ctrl =
## control$checkConv, : Model failed to converge with max|grad| = 0.00328183
## (tol = 0.002, component 1)
```

```
## Warning in checkConv(attr(opt, "derivs"), opt$par, ctrl =
## control$checkConv, : Model failed to converge with max|grad| = 0.00480494
## (tol = 0.002, component 1)
```

```
## Warning in checkConv(attr(opt, "derivs"), opt$par, ctrl =
## control$checkConv, : Model failed to converge with max|grad| = 0.00295183
## (tol = 0.002, component 1)
```

```
## Warning in checkConv(attr(opt, "derivs"), opt$par, ctrl =
## control$checkConv, : Model failed to converge with max|grad| = 0.0022976
## (tol = 0.002, component 1)
```

```
mean(abs(tval)>2,na.rm=TRUE)
```

```
## [1] 0.5
```

Exercise

Load the following data we saw in the exercises in 04_01 and subset the relevant data:

```
chineseRC<-read.table("data/gibsonwu2012data.txt")
crit<-subset(chineseRC,region=="headnoun")
```

```r
crit$region<-factor(crit$region)
head(crit[,c(1,2,3,7)])
```

```
##     subj item     type   rt
## 94     1   13  obj-ext 1561
## 221    1    6 subj-ext  959
## 341    1    5  obj-ext  582
## 461    1    9  obj-ext  294
## 621    1   14 subj-ext  438
## 753    1    4 subj-ext  286
```

```r
crit<-crit[,c(1,2,3,7)]
head(crit)
```

```
##     subj item     type   rt
## 94     1   13  obj-ext 1561
## 221    1    6 subj-ext  959
## 341    1    5  obj-ext  582
## 461    1    9  obj-ext  294
## 621    1   14 subj-ext  438
## 753    1    4 subj-ext  286
```

Tasks

- Code the two levels of the two-level factor called type using sum coding (± 1); call the predictor x.
- Fit a **maximal** linear model to the data investigating the effect of the predictor x (the predictor type that is now sum-coded)
- Using estimates of the parameters from the linear mixed model as a starting point, generate fake data 100 times and find out the prospective power (True Discovery Rate) assuming the observed effect as the true effect. Number of subjects should be 40 and the number of items 16.