Hands-on session: Open research and statistics

Shravan Vasishth[1] & Daniel Schad[1]

[1] University of Potsdam

Author Note

Funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) – Project number 317633480 – SFB 1287, project Q.

Correspondence concerning this article should be addressed to Shravan Vasishth, University of Potsdam. E-mail: vasishth@uni-potsdam.de

Abstract

Hands-on session on open research and statistics

Keywords: keywords

Word count: X

Hands-on session: Open research and statistics

Introduction

Consider the subject and object relative clauses shown in (1):

(1) a. **Subject Relative (SR):** The boy who hugged the girl chased the woman

b. **Object Relative (OR):** The brother who the sister followed kissed the woman

Grodner and Gibson (2005) hypothesized that subject relatives are easier to process than object relatives. We have their data, so we are going to plan a workflow for data analysis, and for a future study.

This is the data from their Experiment 1. You can download the paper from **here**.

In the Grodner and Gibson (2005) paper, the interest is 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 (roughly, it is more difficult to figure out who did what to whom in object relatives than subject relatives because in object relatives, one has difficulty in figuring out which of the nouns is the subject of the relative clause verb).

Load and preprocess data

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

```
library(dplyr)
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
```

Check what the data look like

Each of the 42 participants see multiple (eight) instances of subject and object relatives:

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

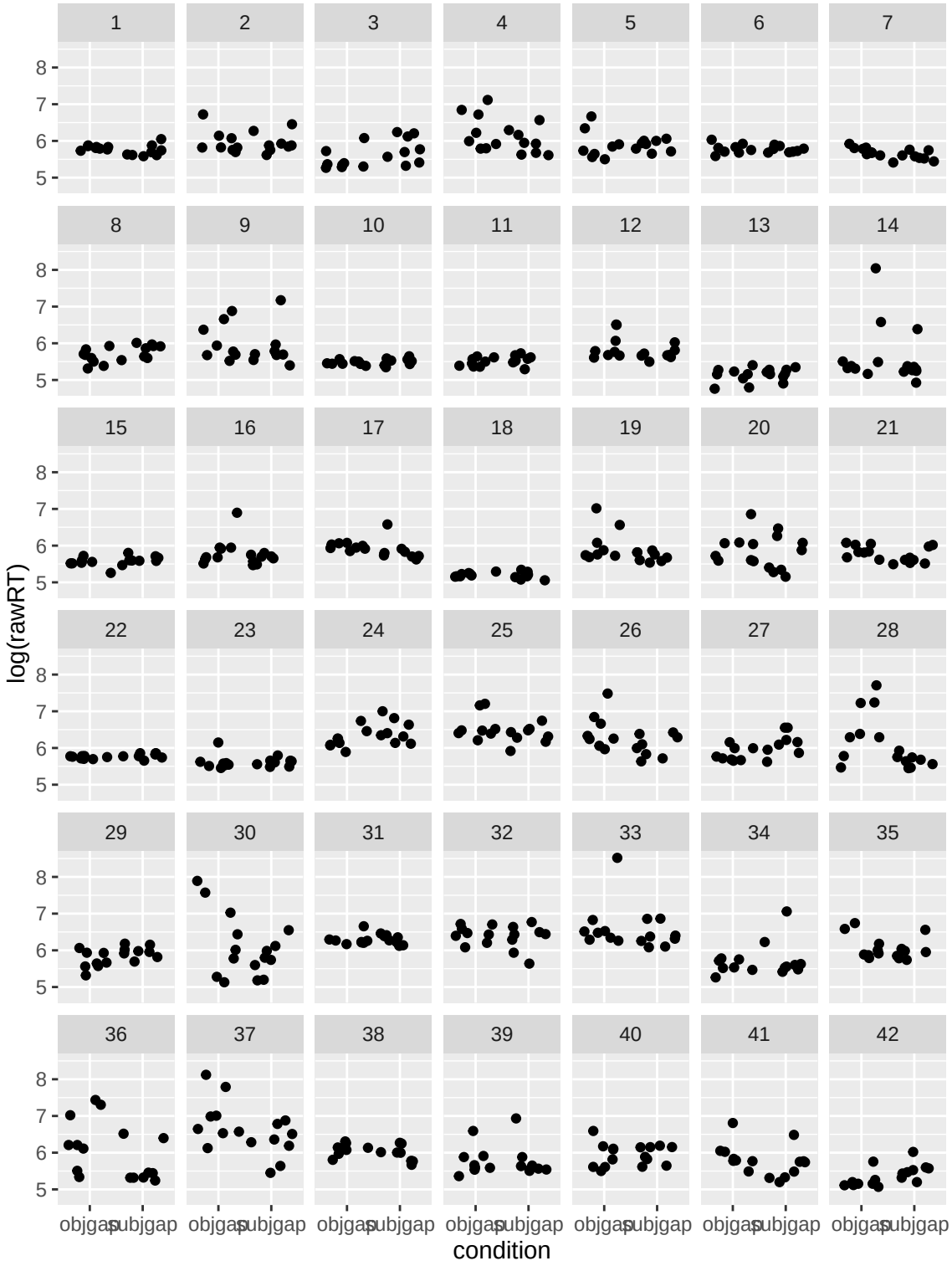
```
##         condition
## subject objgap subjgap
```

```
##      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
```

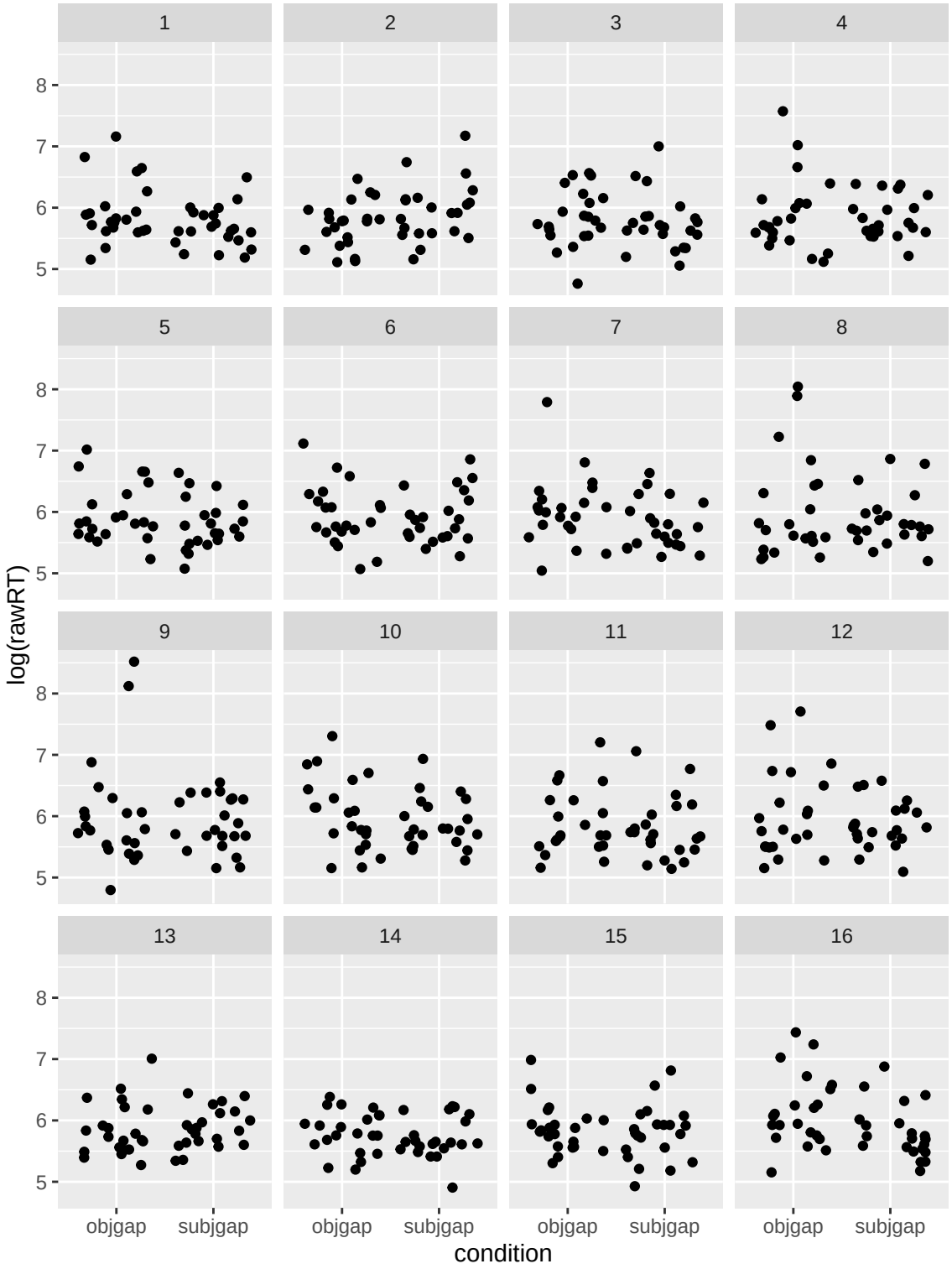
So, from each participant, we have **repeated** measures, which are therefore **not** independent (because they come from the same subject).

```
library(ggplot2)
p <- ggplot(gge1crit, aes(x=condition, y=log(rawRT))) + geom_point(position="jitter")+
  facet_wrap( ~ subject, nrow=6)
p
```



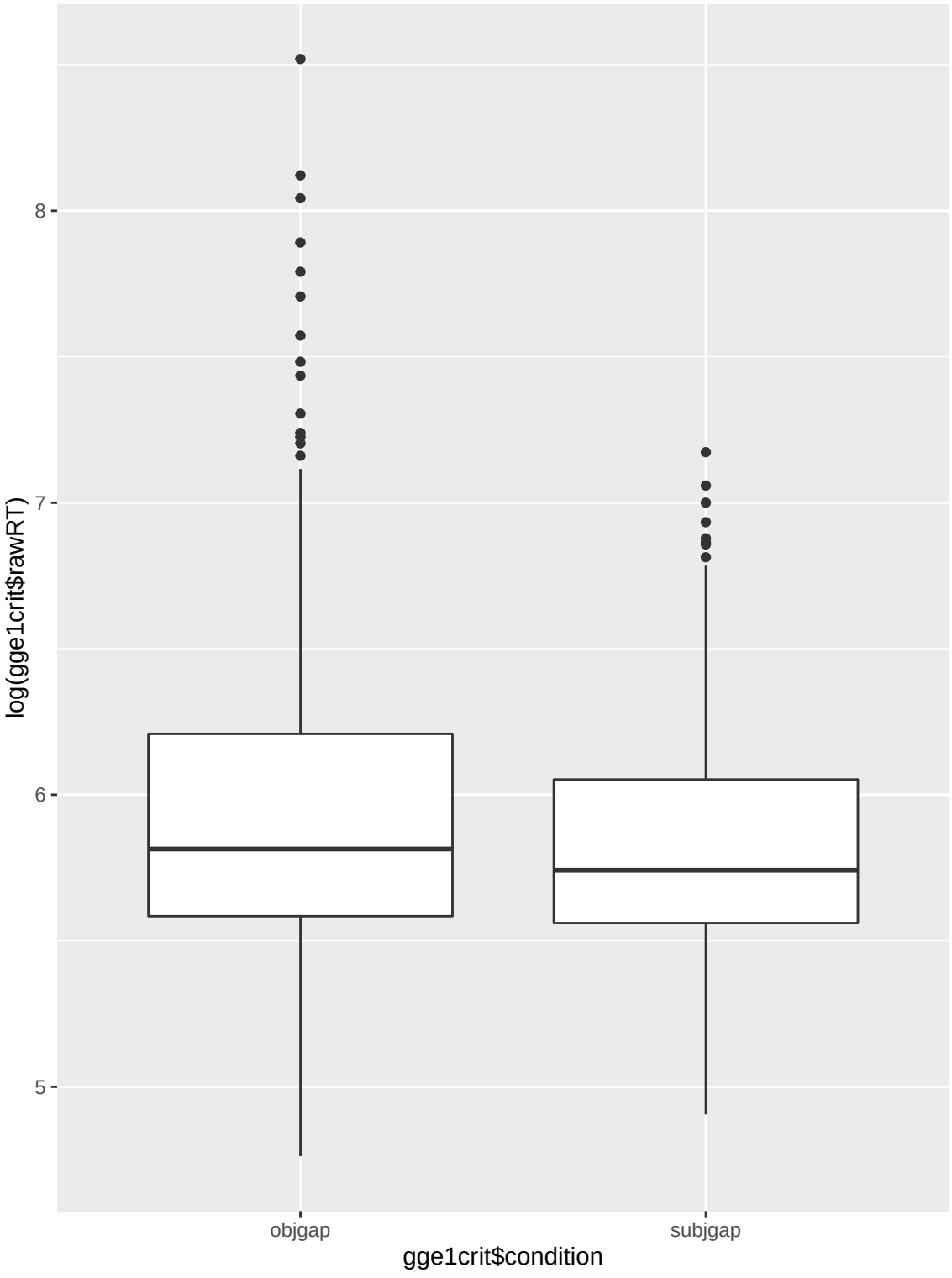
We can do the same for items:

```r
library(ggplot2)

p2 <- ggplot(gge1crit, aes(x=condition, y=log(rawRT))) + geom_point(position="jitter")+

  facet_wrap( ~ item, nrow=4)

p2
```



Boxplots by condition:

```
qplot(gge1crit$condition,log(gge1crit$rawRT),geom="boxplot")
```



There are some unusually long reading times but only in object relatives. Is this

meaningful or not?

Generating fake data

We will now generate fake data repeatedly resembling Grodner and Gibson's data:

Step 1: Fit a linear mixed model to the data

See Schad, Hohenstein, Vasishth, and Kliegl (2020) for a tutorial on contrast coding.

```
## sum contrast coding, +1 for OR and -1 for SR:
gge1crit$so<-ifelse(gge1crit$condition=="objgap",1,-1)
library(lme4)
m<-lmer(log(rawRT)~so+(1+so|subject) + (1+so|item),gge1crit,
        control=lmerControl(calc.derivs=FALSE))
summary(m)
```

```
## Linear mixed model fit by REML ['lmerMod']
## Formula: log(rawRT) ~ so + (1 + so | subject) + (1 + so | item)
##    Data: gge1crit
## Control: lmerControl(calc.derivs = FALSE)
##
## REML criterion at convergence: 693.6
##
## Scaled residuals:
##    Min     1Q Median     3Q    Max
## -2.947 -0.528 -0.123  0.296  6.143
##
## Random effects:
##  Groups   Name        Variance Std.Dev. Corr
##  subject  (Intercept) 0.10103  0.3178
```

```
##           so          0.01228  0.1108   0.58
##  item     (Intercept) 0.00172  0.0415
##           so          0.00196  0.0443   1.00
##  Residual             0.12984  0.3603
## Number of obs: 672, groups:  subject, 42; item, 16
##
## Fixed effects:
##             Estimate Std. Error t value
## (Intercept)   5.8831     0.0520  113.09
## so            0.0620     0.0247    2.51
##
## Correlation of Fixed Effects:
##    (Intr)
## so 0.469
```

```
## extract estimates of fixed-effects parameters:
beta<-summary(m)$coefficients[,1]
## extract standard deviation estimate:
sigma_e<-attr(VarCorr(m),"sc")
## assemble variance covariance matrix for subjects:
subj_ranefsd<-attr(VarCorr(m)$subj,"stddev")
subj_ranefcorr<-attr(VarCorr(m)$subj,"corr")
## choose some intermediate values for correlations:
corr_matrix<-(diag(2) + matrix(rep(1,4),ncol=2))/2
Sigma_u<-SIN::sdcor2cov(stddev=subj_ranefsd,
                        corr=corr_matrix)


## assemble variance covariance matrix for items:
```

```
item_ranefsd<-attr(VarCorr(m)$item,"stddev")
Sigma_w<-SIN::sdcor2cov(stddev=item_ranefsd,
                        corr=corr_matrix)
```

Step 2: Generate fake data using estimates, compute power (and Type I error):

Calculate power for a future study. We load a function to generate fake data:

```
source("../R/gen_fake_lnorm.R")
nsim<-100
tvals<-c()
for(i in 1:nsim){
fakedat<-gen_fake_lnorm(nitem=16,nsubj=40,
               alpha=beta[1],beta=beta[2],
               Sigma_u=Sigma_u,Sigma_w=Sigma_w,
               sigma_e=sigma_e)
m<-lmer(log(rt)~so+(1+so|subj)+(1+so|item),
        fakedat,
        control=lmerControl(calc.derivs=FALSE))
tvals[i]<-summary(m)$coefficients[2,3]
}
## this is only valid if the true effect is 0.06 on log ms scale:
mean(abs(tvals)>2)
```

```
## [1] 0.7
```

Write a function for computing power and Type I error (this is an example of code refactoring):

```
compute_power_type2<-function(nsim=100,b=0.06,
                          nsubj=40,nitem=16){
  tvals<-c()
for(i in 1:nsim){
fakedat<-gen_fake_lnorm(nitem=nitem,nsubj=nsubj,
                alpha=beta[1],beta=b,
                Sigma_u=Sigma_u,Sigma_w=Sigma_w,sigma_e=sigma_e)
m<-lmer(log(rt)~so+(1+so|subj)+(1+so|item),
        fakedat,
        control=lmerControl(calc.derivs=FALSE))
tvals[i]<-summary(m)$coefficients[2,3]
}
mean(abs(tvals)>2)
}
```

How many subjects would I need **in a future study** to have approximately 80% power:

```
library(MASS)
compute_power_type2(nsubj=100)
```

```
## [1] 0.93
```

```
compute_power_type2(nsubj=200)
```

```
## [1] 0.99
```

Calculate Type I error (sanity check only). By setting the slope to be 0, we are able to compute Type I error:

```
## null hypothesis is true:
compute_power_type2(b=0)
```

```
## [1] 0.01
```

Can we recover parameters under repeated sampling? This is informative about whether we should even be trying to fit a maximal model given the sample sizes of subjects and items that we plan to have:

```
nsim<-100
int<-slope<-stddev<-sigma_u0<-sigma_u1<-sigma_w0<-
  sigma_w1<-rho_u<-rho_w<-c()
for(i in 1:nsim){
fakedat<-gen_fake_lnorm(nitem=16,nsubj=40,
                alpha=beta[1],beta=beta[2],
                Sigma_u=Sigma_u,Sigma_w=Sigma_w,
                sigma_e=sigma_e)
m<-lmer(log(rt)~so+(1+so|subj)+(1+so|item),fakedat,
        control=lmerControl(calc.derivs=FALSE))
## extract parameter estimates:
int[i]<-summary(m)$coefficients[1,1]
slope[i]<-summary(m)$coefficients[2,1]
stddev[i]<-summary(m)$sigma
subj_ranefsd<-attr(VarCorr(m)$subj,"stddev")
sigma_u0[i]<-subj_ranefsd[1]
sigma_u1[i]<-subj_ranefsd[2]
subj_ranefcorr<-attr(VarCorr(m)$subj,"corr")
rho_u[i]<-subj_ranefcorr[1,2]
## assemble variance covariance matrix for items:
```

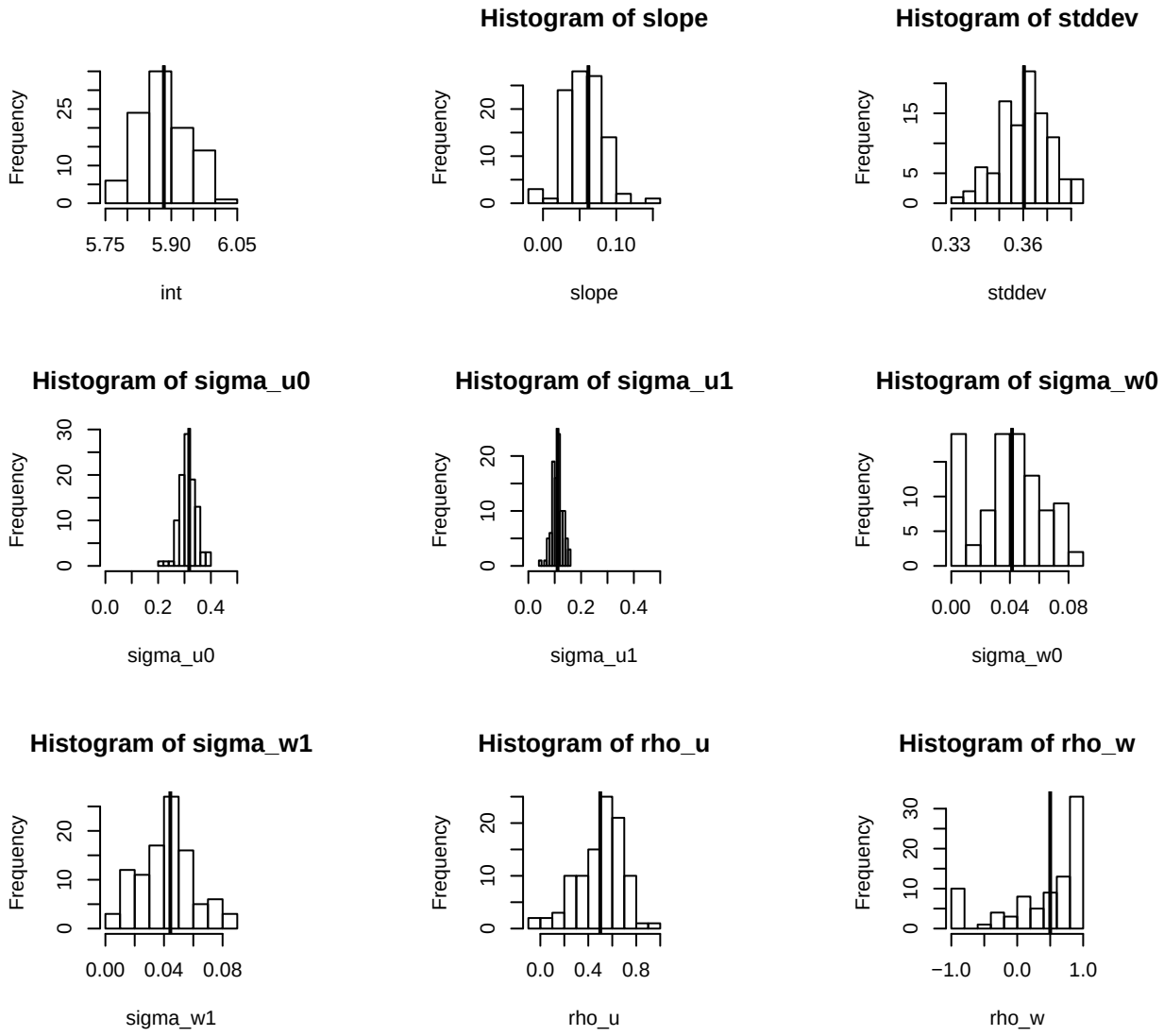
```
item_ranefsd<-attr(VarCorr(m)$item,"stddev")
sigma_w0[i]<-item_ranefsd[1]
sigma_w1[i]<-item_ranefsd[2]
item_ranefcorr<-attr(VarCorr(m)$item,"corr")
rho_w[i]<-item_ranefcorr[1,2]
}
```

One could automate the above steps by writing a function that extracts the parameters.

Check if the repeatedly estimated parameters match the true values used to generate the data:

```
op<-par(mfrow=c(3,3),pty="s")
hist(int,main="")
abline(v=beta[1],lwd=2)
hist(slope)
abline(v=beta[2],lwd=2)
hist(stddev)
abline(v=sigma_e,lwd=2)
hist(sigma_u0,xlim=c(0,0.5))
abline(v=sqrt(Sigma_u[1,1]),lwd=2)
hist(sigma_u1,xlim=c(0,0.5))
abline(v=sqrt(Sigma_u[2,2]),lwd=2)
hist(sigma_w0)
abline(v=sqrt(Sigma_w[1,1]),lwd=2)
hist(sigma_w1)
abline(v=sqrt(Sigma_w[2,2]),lwd=2)
hist(rho_u)
```

```
abline(v=0.5,lwd=2)

hist(rho_w)

abline(v=0.5,lwd=2)
```



Notice that the correlation between the item varying intercept and slope, and the correlations, are not recovered accurately by lmer. For the sample size used in Grodner and Gibson (2005). There wouldn't be much point in fitting a maximal model (Barr, Levy, Scheepers, & Tily, 2013) here.

Now we can plan our analysis in advance **for this planned sample size**: we can

specify the model formula to be:

```
log(rt)~so+(1+so|subj)+(1+so||item)
```

This information then goes into a pre-registration.

How to release data and code

The basic principle is always to think about what information the user will need to reproduce the analyses. This is minimally:

- the data used in the paper
- the R code that generated all plots and analyses
- documentation of the experiment design and predictions
- any additional data and code needed

One simple approach is to

- create a package

```
usethis::create_package("mypackage")
```

- put it on osf.io or github or both

I will demonstrate data installation in class.

Exercise 1

Load the following data and subset the relevant data as shown:

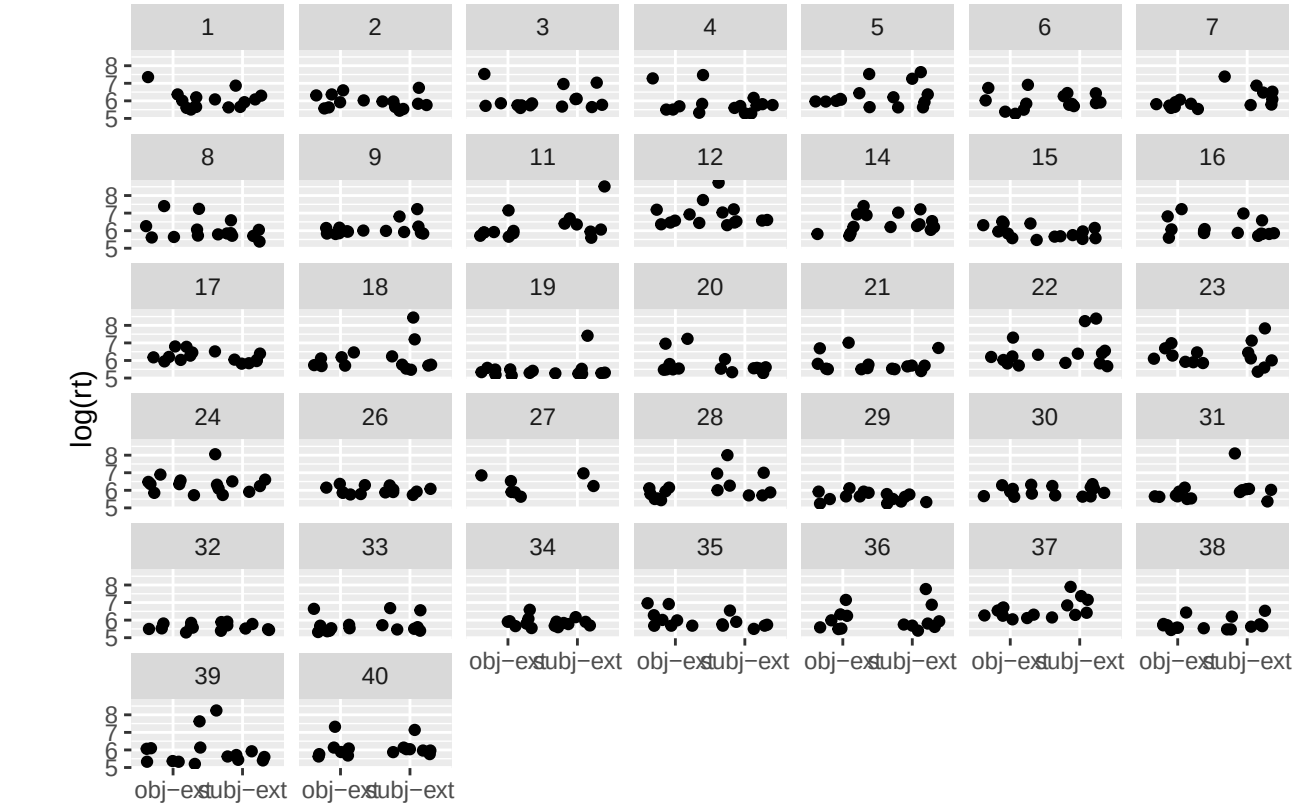
```
chineseRC<-read.table("../data/gibsonwu2012data.txt",header=TRUE)
## isolate the critical region:
crit<-subset(chineseRC,region=="headnoun")
crit$region<-factor(crit$region)
```

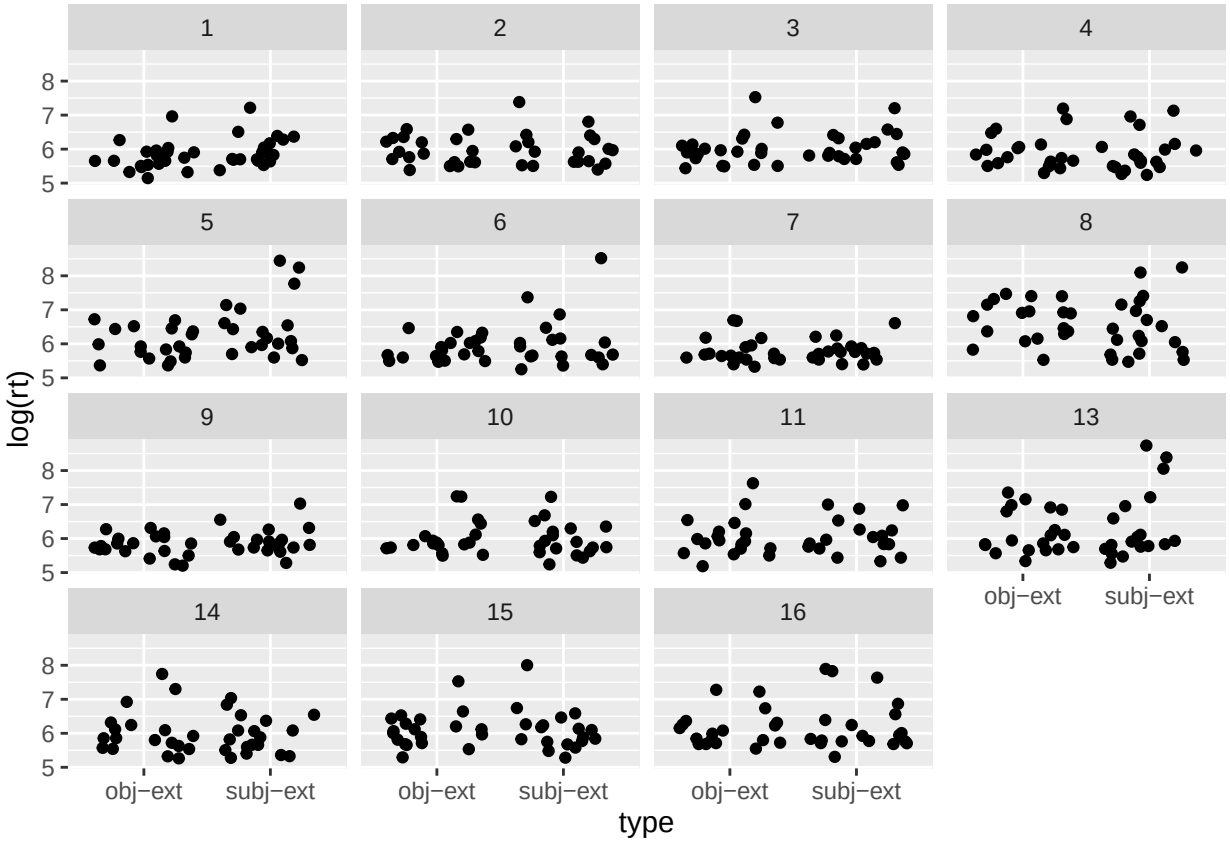
```
crit<-crit[,c(1,2,3,7)]
head(crit)
```

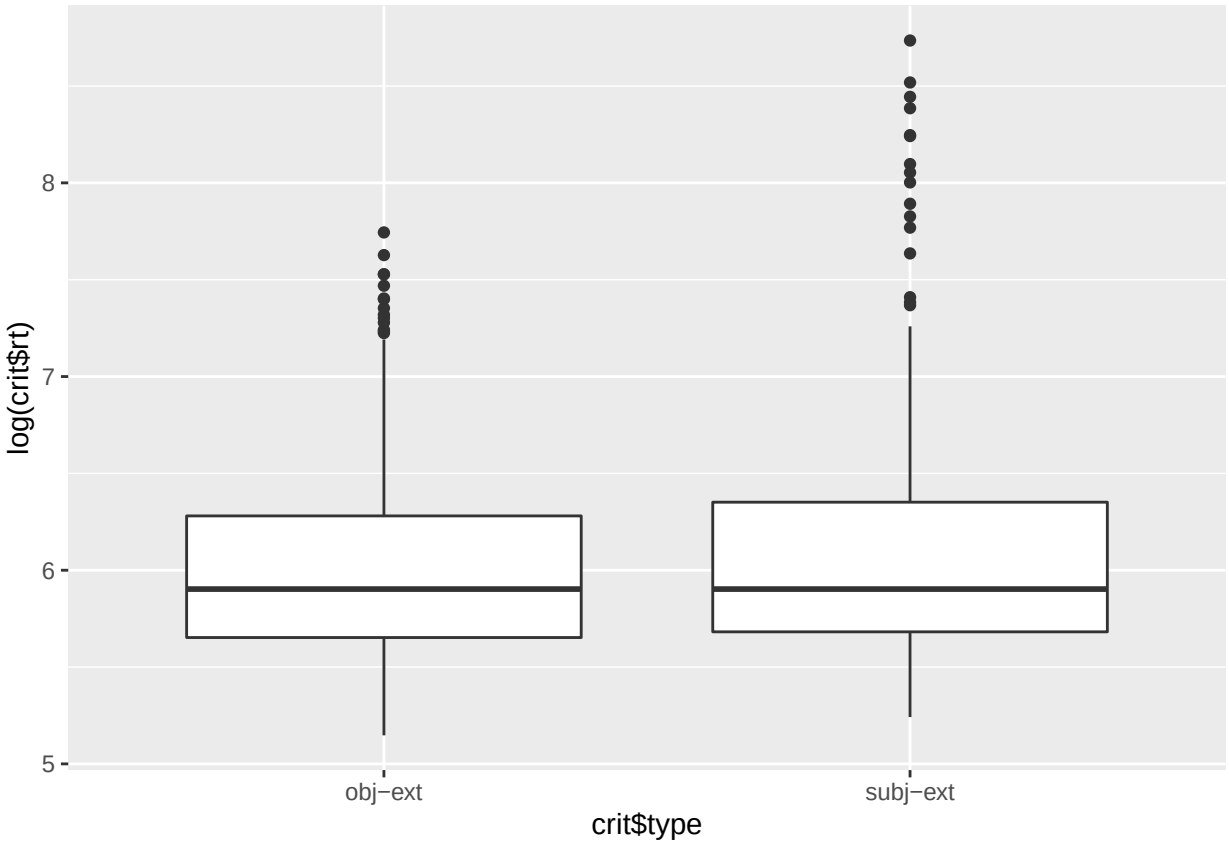
```
##     subj item     type   rt
## 94     1   13  obj-ext 1561
## 221    1    6 subj-ext  959
## 341    1    5  obj-ext  582
## 461    1    9  obj-ext  294
## 621    1   14 subj-ext  438
## 753    1    4 subj-ext  286
```

Tasks

- Code the two levels of the two-level factor called type using sum coding (± 1); call the predictor so, and code obj-ext as +1, subj-ext as -1.
- Carry out the above steps, and establish what sample size (number of subjects) you need to detect an effect, with power 0.80, whether the estimate of the effect (the slope of the fixed effect) is significant. All the code is provided above for this, you just need to reuse it, adapting the code as needed.


1
2
3
4
5
6
7
8
9
11
12
14
15
16
17
18
19
20
21
22
23
24
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
log(rt)
obj−ext
subj−ext
type


1
2
3
4
5
6
7
8
9
10
11
13
14
15
16
log(rt)
obj−ext
subj−ext
type



Exercise 2

This exercise demonstrates why it's important to pre-specify your analysis before you collect your data.

Load the following data, from Expt 1 of this Frontiers paper: https://www.frontiersin.org/articles/10.3389/fpsyg.2019.02210/full

```
priming<-read.csv("../data/Frontiers/analysis_script/E1_RTAll.csv",
                  header=TRUE)
head(priming)
```

```
##   Sub Type ID Position      RT ACC
## 1   1    1  1        1 0.43369   1
## 2   1    1  2        1 0.35046   1
```

```
## 3   1    1  3        1 0.33372   1
## 4   1    1  4        1 0.36713   1
## 5   1    1  5        1 0.33372   1
## 6   1    1  6        1 0.36707   1
```

```
## Type is subject vs object RC probably, not sure which is which:
#xtabs(~Sub+Type,priming)
## ID is the item id:
#xtabs(~Sub+ID,priming)


## reported subject and item numbers check out:
length(unique(priming$Sub))
```

```
## [1] 46
```

```
length(unique(priming$ID))
```

```
## [1] 64
```

```
## head noun, critical region:
headnoun<-subset(priming,Position==4)


## convert to ms:
headnoun$RT<-headnoun$RT*1000


## some imbalance, probably due to data removal:
#xtabs(~ID+Type,priming)
#summary(priming)
```

Here is the published result:

"the ORC sentence was found to read significantly faster than the SRC sentence at the head noun region (W4, β=-0.03, SE=0.01, t=-2.68)

Task for half the class: demonstrate that this claim is false.

Task for the other half: demonstrate that this claim is true.

The moral here is: it is not at all obvious which model should be fit, and what the conclusion should be. Depending on what result you want, you can report either an OR advantage, or a null result. This is a pretty common situation to be in.

Note also that it is not obvious what would happen with the untrimmed data (in Gibson and Wu 2013 no trimming was done).

Finally, note that the authors should have checked whether they had sufficient power to detect the effect.

References

```
r_refs(file = "r-references.bib")
```

Barr, D. J., Levy, R., Scheepers, C., & Tily, H. J. (2013). Random effects structure for confirmatory hypothesis testing: Keep it maximal. *Journal of Memory and Language*, *68*(3), 255–278.

Grodner, D., & Gibson, E. (2005). Consequences of the serial nature of linguistic input for sentential complexity. *Cognitive Science*, *29*, 261–291.

Schad, D. J., Hohenstein, S., Vasishth, S., & Kliegl, R. (2020). How to capitalize on a priori contrasts in linear (mixed) models: A tutorial. *Journal of Memory and Language*, *110*. https://doi.org/https://doi.org/10.1016/j.jml.2019.104038