

University of Potsdam, Germany

SMLP 2017, Frequentist foundations exercises for Monday and Tuesday

1 Exercises on the binomial random variable

Simulate the toss of one coin. We will assume that an output of 1 means heads and an output of 0 means tails.

Run the following code:

```
rbinom(n = 1, size = 1, prob = 0.5)
## [1] 1
```

Run this command repeatedly 10 times. Now change n to 10:

```
rbinom(n = 10, size = 1, prob = 0.5)
## [1] 1 0 1 0 0 1 1 0 1 1
```

Run the above code repeatedly.

Now, calculate the average number of heads in 10 coin tosses:

```
mean(rbinom(n = 10, size = 1, prob = 0.5))
## [1] 0.4
```

Run the above code repeatedly. You will see that the mean number of heads can be different in each run.

2 Exercises on the normal random variable

1. Sample 1000 data points from a normal distribution with mean 500 and standard deviation 50.
2. Calculate the mean of the 1000 data points.
3. Calculate the standard deviation of the 1000 data points (function in R: sd).

Answer the following questions:

1. What is the probability of obtaining a value of **1.5 or less** from a random variable that has as pdf a normal distribution of mean 0 and sd 1? Use pnorm to find out.

- Without writing any further R code, can you say what the probability is of obtaining a value of **1.5 or more** from the above random variable?
- What is the value x in $P(X < x) = 0.75$ when $X \sim N(0, 1)$? That is, what is the value x such that the probability of obtaining a value equal to or less than x is 0.75? Use `qnorm` to find out.

3 The sampling distribution of the sample mean

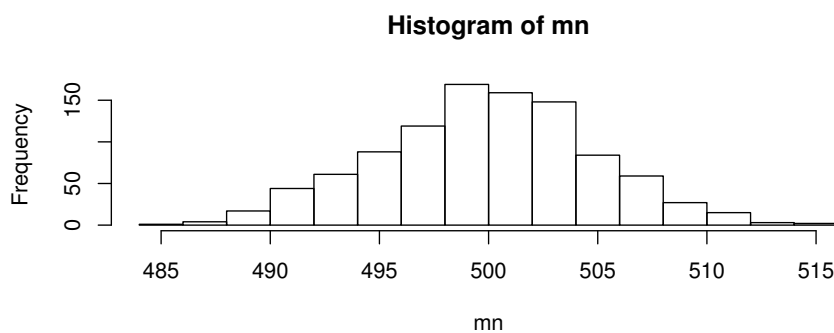
Recall the discussion about the sampling distribution of the means in lecture 1

Do the following 1000 times: take a random sample of size 100 from a normal distribution with mean 500 and standard deviation 50, compute the mean from each sample, and save it in a vector called `mn`. We will use a for-loop for this:

```
nsim <- 1000
mn <- rep(NA, nsim)
for (i in 1:nsim) {
  mn[i] <- mean(rnorm(n = 100, mean = 500, sd = 50))
}
```

Next, plot the histogram of the distribution of sample means stored in `mn`.

```
hist(mn)
```



What are the

- mean
- standard deviation

of this distribution of sample means?

Recall that the standard deviation of the sampling distribution of means is called the **standard error**. This standard error should be approximately 5; why is that?

3.1 Sampling from an exponential

Instead of sampling from a normal distribution with mean 500 and standard deviation 50, now sample from an exponential distribution with mean 1 and sd 1. You can take a sample of size 100 from an exponential as follows:

```
samp <- rexp(n = 100)
```

The mean of this exponential distribution is 1, and standard deviation is 1.

1. Do the following 1000 times: take a sample of size $n=100$ from the exponential, and compute the mean each time and store it in a vector `mn`, as in the previous example.
2. Compute the mean and standard error of the sampling distribution of sample means.
3. What should the standard error be, given the standard deviation of the exponential and the sample size? Recall the fact that standard error is (standard deviation) / \sqrt{n} .

3.2 Sampling from a uniform

Instead of sampling from a normal distribution with mean 500 and standard deviation 50, now sample from a uniform distribution with mean 1 and sd 1, as shown below. You can take a sample of size 100 from an exponential as follows:

```
samp <- runif(n = 100, min = 0, max = 1)
```

The mean of this distribution is 0.5, and standard deviation is about .28.

1. Take 1000 samples like these, and compute the mean each time and store it in a vector `mn` as in the previous example.
2. Compute the mean and standard error of the sampling distribution of sample means.
3. What should the standard error be, given the standard deviation of the uniform distribution and the sample size?

3.3 Practising basic skills in computing probability, quantiles, and testing the central limit theorem

Now you should be able to do the following given some random variable X :

1. Compute the probabilities: $P(X < x)$ or $P(X > x)$ or $P(x_1 < X < x_2)$.
Example: Suppose $X \sim \text{Normal}(0, 1)$. Find the probabilities:
 $P(X < 0.4)$ or $P(X > 0.8)$, and $P(0.4 < X < 0.8)$
2. Compute the quantile x such that $P(X < x) = p$ or $P(X > x) = p$.
Example: Suppose $X \sim \text{Normal}(0, 1)$. Find the quantiles:
 $P(X < x) = 0.4$ or $P(X > x) = 0.8$.

3. Demonstrate the Central Limit Theorem through simulation, with X being a random variable from the poisson distribution, with parameter $\lambda=2$ (earlier we tested the normal, the exponential, and the uniform).

Sampling from the Poisson can be done like this:

```
samp <- rpois(100, lambda = 2)
```

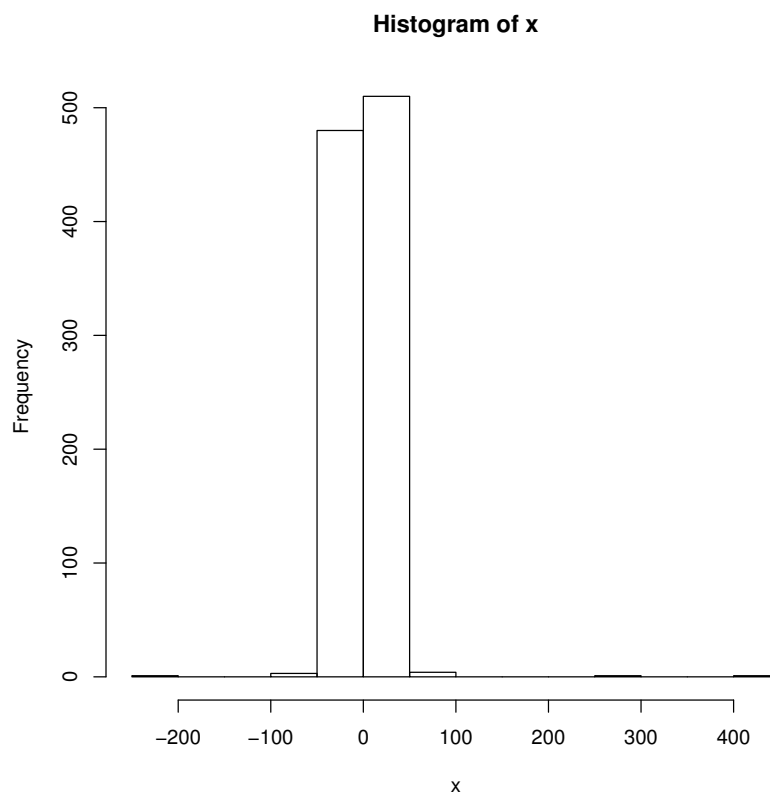
3.4 Sampling from the Cauchy distribution

The central limit theorem only applies when the mean and variance of a distribution are defined. The Cauchy distribution is exceptional in that the mean and variance are undefined.

Here is what samples from the Cauchy look like:

```
x <- rcauchy(1000)
hist(x)
summary(x)
```

##	Min.	1st Qu.	Median	Mean	3rd Qu.	Max.
##	-219.6000	-0.8921	0.0465	0.2805	1.0910	400.1000



Notice the massive spread of the observed values.

Try sampling repeatedly from the Cauchy and computing the sampling distribution of the sample means. Is the sampling distribution normally distributed?

4 Paired t-test

We have repeated measures data on noun pronunciation durations, in seconds:

```
dataN2 <- read.table("data/dataN2.txt", header = T)
head(dataN2)

##   Sentence Speaker_id N2_dur.2 N2_dur.1
## 1         1           1 0.4965026 0.6144392
## 2         1           2 0.4797888 0.5873895
## 3         1           3 0.5471585 0.6945130
## 4         1           4 0.3783597 0.5684208
## 5         1           5 0.5671948 0.4404005
## 6         1           6 0.5183090 0.5465097
```

Incorrect analysis:

```
## significant effect:
with(dataN2, t.test(N2_dur.2, N2_dur.1, paired = TRUE))

##
## Paired t-test
##
## data: N2_dur.2 and N2_dur.1
## t = 2.22, df = 335, p-value = 0.02709
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  0.002320133 0.038405219
## sample estimates:
## mean of the differences
##                0.02036268
```

The above t-test was incorrect because we have multiple rows of (dependent) data from the same subject.

We need to aggregate the multiple measurements from each subject until we have one data point from each subject for each combination of vowel and language.

First, convert data to "long" form:

```
N2dur1data<-data.frame(item=dataN2$Sentence,
                      subj=dataN2$Speaker_id,
                      cond="a",
                      dur=dataN2$N2_dur.1)
N2dur2data<-data.frame(item=dataN2$Sentence,
                      subj=dataN2$Speaker_id,
                      cond="b",
```

```
dur=dataN2$N2_dur.2)

N2data<-rbind(N2dur1data,N2dur2data)
```

```
head(N2data)

##   item subj cond      dur
## 1    1    1    a 0.6144392
## 2    1    2    a 0.5873895
## 3    1    3    a 0.6945130
## 4    1    4    a 0.5684208
## 5    1    5    a 0.4404005
## 6    1    6    a 0.5465097
```

Then aggregate so that we have only one data point per subject for each condition:

```
N2data_bysubj <- aggregate(dur ~ subj + cond, mean, data = N2data)
```

Create a vector for each condition:

```
conda <- subset(N2data_bysubj, cond == "a")
condb <- subset(N2data_bysubj, cond == "b")
```

```
## not significant:
t.test(condb$dur, conda$dur, paired = TRUE)

##
## Paired t-test
##
## data: condb$dur and conda$dur
## t = 1.8355, df = 13, p-value = 0.08941
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
## -0.003604625 0.044329976
## sample estimates:
## mean of the differences
## 0.02036268
```

Alternative syntax:

```
## alternative syntax:
t.test(dur ~ cond, paired = TRUE, N2data_bysubj)

##
## Paired t-test
##
## data: dur by cond
```

```
## t = -1.8355, df = 13, p-value = 0.08941
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -0.044329976  0.003604625
## sample estimates:
## mean of the differences
## -0.02036268
```

Now do a by-items t-test:

```
## STEP 1: Aggregate over items: N2data_byitem<-aggregate(dur~item+cond,mean,
## data=N2data)
```

```
## STEP 2: Create a vector for condition a and b:
## conda<-subset(N2data_byitem,cond=='a')
## condb<-subset(N2data_byitem,cond=='b') conda<-... condb<-... Do a by
## items paired t-test: t.test(condb$dur,conda$dur,paired=TRUE)
```

5 By subject and by item paired t-tests on Gibson and Wu data

Run two t-tests on the Gibson & Wu data, one t-test with the data aggregated by-subjects and another one with the data aggregated by-items. Which kind of t-test do you need here (paired or unpaired)?

6 OPTIONAL: Main effects and interactions using t-tests

In the 2x2 design for the noise and degree data, test the main effects of noise and degree, and the interaction between noise and degree, by running three paired t-tests by-subject, and three paired t-tests by-item.

7 Generating fake data for one sample t-tests, to check for power and Type I error

It is easy to generate fake data, e.g., from a log-normal or normal:

```
y <- rnorm(100, mean = 0, sd = 0.5)
```

You can study Type I error by repeatedly sampling and computing the p-value:

```
nsim <- 1000
pvals <- rep(NA, nsim)
for (i in 1:nsim) {
```

```

y <- rnorm(100, mean = 0, sd = 10)
pvals[i] <- t.test(y)$p.value
}
## approximately 5%
mean(pvals < 0.05)

## [1] 0.052

```

In the above case, what is the power if the true effect is 1? And when it is 3?

8 Run a linear model

Run a linear model on the noisedeg dataset, with two predictors, noise level and degree of tilt. Make sure you go through all the following steps:

1. Load dataset and change the two independent variables' names from their factor levels to the numerical values 0 and 1.

```

noisedeg <- read.table("data/noisedeg.txt", header = TRUE)

noisedeg$deg_R <- ifelse(noisedeg$deg == "0", 0, 1)
noisedeg$noise_R <- ifelse(noisedeg$noise == "no.noise", 0, 1)
head(noisedeg)

##      rt subj deg      noise deg_R noise_R
## 1 420   s1  0 no.noise      0      0
## 2 420   s2  0 no.noise      0      0
## 3 480   s3  0 no.noise      0      0
## 4 420   s4  0 no.noise      0      0
## 5 540   s5  0 no.noise      0      0
## 6 360   s6  0 no.noise      0      0

```

2. Run the linear model.

```

m1 <- lm(rt ~ deg_R + noise_R, data = noisedeg)

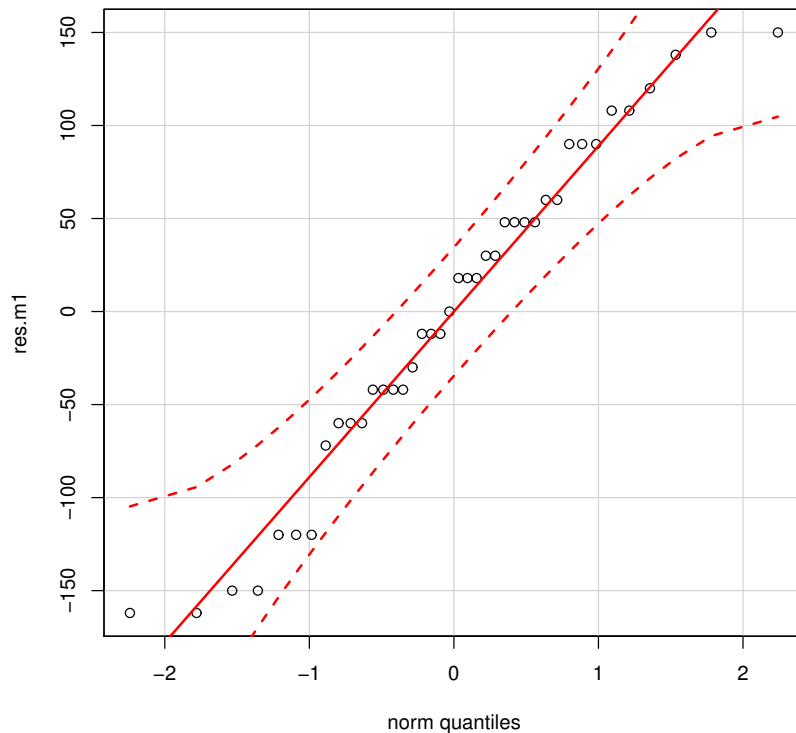
```

3. Check that the residuals are more or less normally distributed

```

## residuals:
res.m1 <- residuals(m1)
library(car)
qqPlot(res.m1)

```

4. Look at the coefficients and interpret the results

```
round(summary(m1)$coefficients, 3)
```

	Estimate	Std. Error	t value	Pr(> t)
## (Intercept)	432	25.08	17.225	0.000
## deg_R	108	28.96	3.729	0.001
## noise_R	90	28.96	3.108	0.004

If time, you can follow the same procedure to run a model in which you include the main effects as well as interaction between degree of tilt and noise. Here is how you can obtain this model:

```
m2 <- lm(rt ~ deg_R * noise_R, data = noisedeg)
```

9 Generating fake data for linear models

A statistical model expresses the generative process that led to the data. In this exercise, we will turn the problem of data analysis on its head: we will define the generative process, and then look at how well the model can recover the parameters.

Here is a simple data generation process.

```

n <- 100
beta1 <- 6
beta2 <- -0.07
sigma <- 0.5
x <- rep(c(-0.5, 0.5), each = n/2)

y <- beta1 + beta2 * x + rnorm(n, mean = 0, sd = sigma)

```

This corresponds to the simple linear model with one categorical predictor coded ± 0.5 . When we fit such a model to the data, we are expressing a belief about how we think the data might be generated:

```

summary(m <- lm(y ~ x))

##
## Call:
## lm(formula = y ~ x)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.11027 -0.28200 -0.00075  0.26563  1.28993
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  6.04984     0.04797 126.121  <2e-16
## x           -0.11097     0.09594  -1.157    0.25
##
## Residual standard error: 0.4797 on 98 degrees of freedom
## Multiple R-squared:  0.01347, Adjusted R-squared:  0.003402
## F-statistic: 1.338 on 1 and 98 DF,  p-value: 0.2502

```

Repeatedly generate fake data as shown above; do this 1000 times. Each time the fake data is generated, fit a linear model to the data (as shown above), and record the parameters from the model. You can extract the three parameters like this:

```

coef(m)

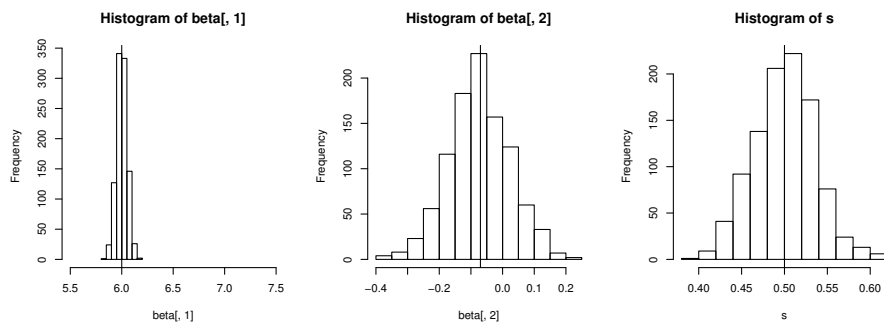
## (Intercept)          x
##  6.0498398 -0.1109718

summary(m)$sigma

## [1] 0.4796866

```

Then plot the distributions of the estimated parameters, and check whether the true parameter lies within the distributions. You should be able to generate a plot like this:



Hint:

To do this exercise, you need to write a loop:

```
nsim<-1000
for(i in 1:nsim){
  ## generate data
  ## fit linear model
  ## extract parameters and store in a vector
}
```

10 Generating bivariate data

This exercise gives you a feel for generating bivariate data. Understanding how a variance-covariance matrix expresses bi- or multivariate data is important for understanding linear mixed models' random effects components.

Generate a bivariate distribution as follows:

1. First, define the two variances, and the correlation:

```
var1 <- 100
var2 <- 200
rho <- 0.9
```

2. Next, use the function `mvrnorm` from the library `MASS` to generate 100 bivariate data points from a bivariate normal with means 0 and variance covariance matrix Σ composed from the above variances and correlation:

```

library(MASS)

Sigma <- matrix(c(var1,
                  sqrt(var1)*sqrt(var2)*rho,
                  sqrt(var1)*sqrt(var2)*rho,
                  var2),ncol=2)

Sigma

##           [,1]      [,2]
## [1,] 100.0000 127.2792
## [2,] 127.2792 200.0000

u<-mvrnorm(n=100,
           mu=c(0,0),Sigma=Sigma)
head(u)

##           [,1]      [,2]
## [1,] -6.700386 -12.90360
## [2,] -18.486165 -24.01280
## [3,] 13.827755 12.76741
## [4,] -16.475079 -39.65779
## [5,] -25.177050 -36.98865
## [6,] -7.398715 -10.21822

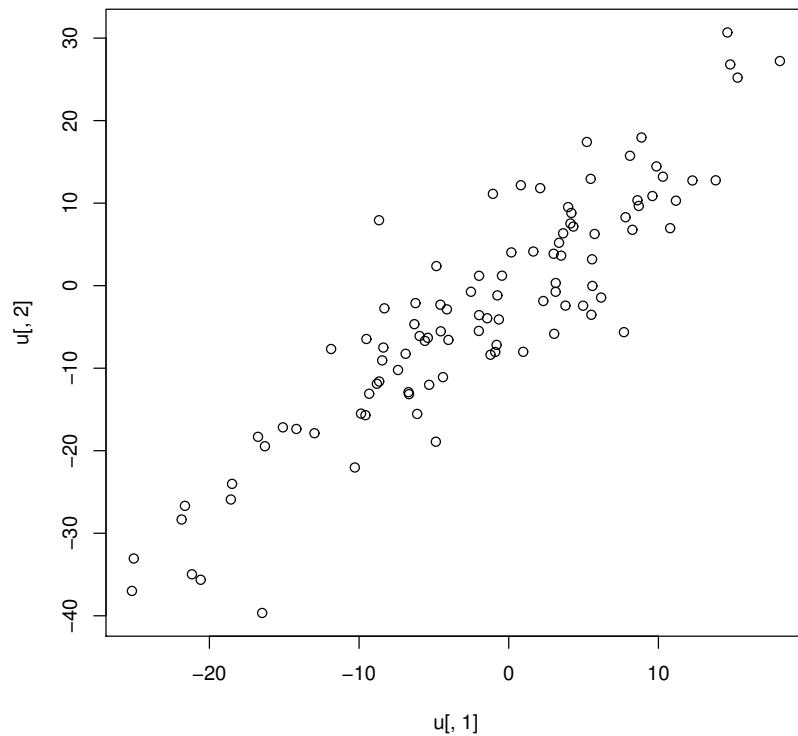
```

3. Now plot the relationship between the two distributions:

```

plot(u[, 1], u[, 2])

```



4. Next, change the variances to 10 and 20 respectively, instead of 100 and 200, and plot the relationship between the two distributions. What changes?
5. Finally, change the correlation in the variance covariance matrix to 0 and then -0.9 , and replot the relationship between the two distributions for these two cases. What is different in the two cases where correlation is 0 vs -0.9 ?

11 The connection between the paired t-test and linear mixed models

This is not an exercise as much as an observation. The paired t-test is identical to a varying intercepts linear mixed model. The generative process implied is the same.

To see this, generate paired data as follows:

```
## by subject intercepts:
bi <- rnorm(100, sd = 0.3)
## random error:
epsilon1 <- rnorm(100, mean = 0, sd = 0.4)
epsilon2 <- rnorm(100, mean = 0, sd = 0.4)
# observed data:
y1 <- 6 - 0.07 * 0.5 + bi + epsilon1
```

```

y2 <- 6 + 0.07 * 0.5 + bi + epsilon2
t.test(y2, y1, paired = TRUE)

##
## Paired t-test
##
## data: y2 and y1
## t = 2.1229, df = 99, p-value = 0.03626
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  0.006423978 0.190233314
## sample estimates:
## mean of the differences
##                0.09832865

```

This paired t-test corresponds to the following linear mixed model:

```

dat <- data.frame(subj = rep(1:100, 2), y = c(y1, y2), x = rep(c(-0.5, 0.5),
  each = 100))
head(dat)

##      subj      y      x
## 1      1 6.029388 -0.5
## 2      2 6.136514 -0.5
## 3      3 6.534361 -0.5
## 4      4 6.460267 -0.5
## 5      5 5.906542 -0.5
## 6      6 6.608565 -0.5

m <- lmer(y ~ x + (1 | subj), dat)
summary(m)

## Linear mixed model fit by REML ['lmerMod']
## Formula: y ~ x + (1 | subj)
## Data: dat
##
## REML criterion at convergence: 240.9
##
## Scaled residuals:
##      Min       1Q   Median       3Q      Max
## -2.19571 -0.53562 -0.05003  0.51356  2.12755
##
## Random effects:
## Groups   Name      Variance Std.Dev.
## subj    (Intercept) 0.1123   0.3352
## Residual              0.1073   0.3275
## Number of obs: 200, groups: subj, 100
##
## Fixed effects:
##              Estimate Std. Error t value

```

```
## (Intercept)  5.98430    0.04074  146.89
## x           0.09833    0.04632    2.12
##
## Correlation of Fixed Effects:
##   (Intr)
## x 0.000
```

12 Generating fake data for linear mixed models and studying power using simulation

First, load the function for generating fake data. This function assumes that we have a two-condition repeated measures design; later on, we will generalize this to any arbitrary design.

```
source("R/gen_fake_lnorm.R")
```

This function assumes that the data are generated from a maximal model for data like the Gibson and Wu 2013 data-set discussed in the slides. The true underlying parameter values assumed here are taken from the estimates we got from the Gibson and Wu data:

Random effects:

Groups	Name	Variance	Std.Dev.	Corr
subj	(Intercept)	5.993e-02	0.2448145	
	x	1.418e-02	0.1190643	-1.00
item	(Intercept)	3.314e-02	0.1820318	
	x	1.946e-07	0.0004412	1.00
Residual		2.645e-01	0.5143264	

Number of obs: 547, groups: subj, 37; item, 15

Fixed effects:

	Estimate	Std. Error	t value
(Intercept)	6.06180	0.06572	92.24
x	-0.07250	0.04831	-1.50

The only exception is the correlations of the random effects; these are assumed to be -0.6 for both subjects and items, rather than the ± 1 you see above.

First, generate fake data repeatedly as shown in the lecture, and calculate power for the case where the effect (the fixed effect slope) is the default value of -0.07 . You can ignore the convergence error warnings for now.

```
nsim<-100
tval<-rep(NA,nsim)
for(i in 1:nsim){
  dat<-gen_fake_lnorm(nsubj=20,nitem=16,
                      beta=c(6,-0.07),
                      ranefsd=c(0.25,0.12,
                                0.18,0.0004),
```

```

                                corr=c(-.6,-.6),
                                sigma_e=0.51)
mfake<-lmer(log(rt)~x+(1+x|subj)+(1+x|item),dat)
tval[i]<-summary(mfake)$coefficients[2,3]
}
## power:
mean(abs(tval)>2)

## [1] 0.14

```

Next, compute power with the following changes. Each change should lead to an increase in power.

- increase subject sample size to 40, with number of items 16.
- increase item sample size to 32, with number of subjects 20.
- increase item sample size to 32, with number of subjects 40.

The practical application of this approach is the following. If you have a hypothesis that the effect of interest lies between, say, -0.03 and -0.11, you can plot a power function for this range of effect sizes, given your assumptions about the different parameters (based on a previous study or studies, as in this case). This allows you to determine what sample size you need (how many subjects, items) to achieve a reasonably high power.

Holding number of items constant at 16, how many subjects do you need to have 80% power in the above case, where the true effect lies between -0.03 and -0.11? (You can either ignore convergence failures for now, or fit a model with no correlation parameters.) Plot the power function for each of the two effect sizes, with the number of subjects ranging from 20 to 200 on the x-axis, and power on the y-axis. This now gives you a basis for deciding what sample size you should use to achieve reasonably high power.