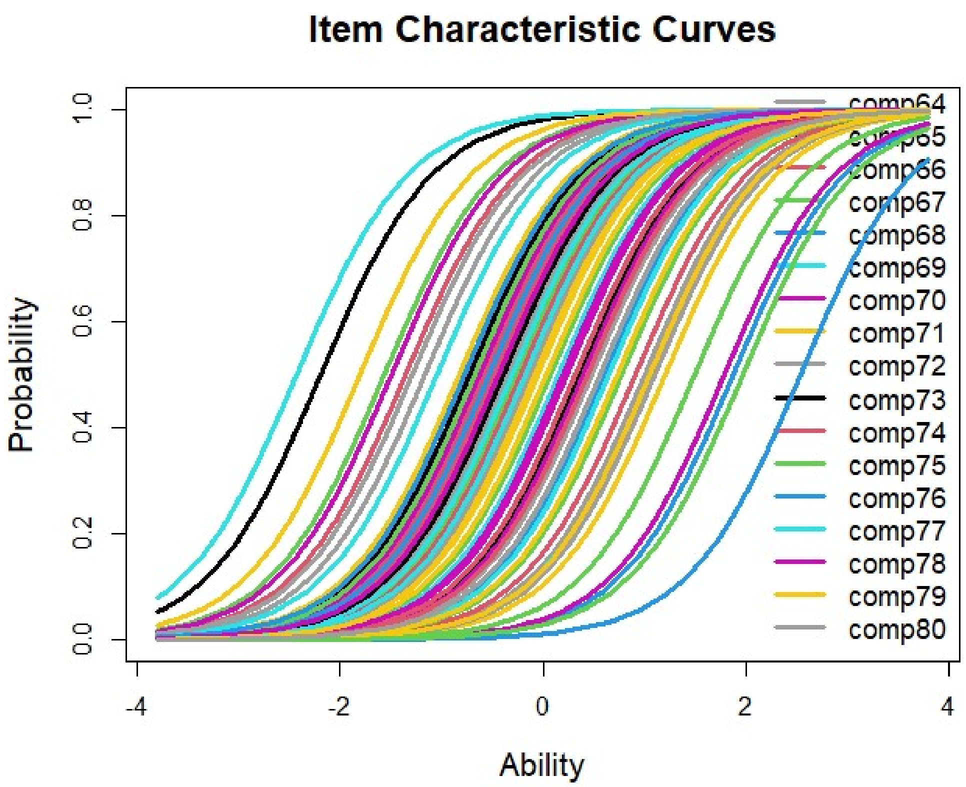
Simulations word completion experiment (CT) Menut et al.

First we analyze the data of the experiment with a simple Rasch model (difficulty of items varies). Datafile is Completion_lmm.xlsx.

```
library(readxl)
library(ltm)
Completion_lmm <- read_excel("Completion_lmm.xlsx")
mydata <- Completion_lmm[ -c(1:2) ]
fit1 <- rasch(mydata)
summary(fit1)
plot(fit1, legend = TRUE, cx = "bottomright", lwd = 3, cex.main = 1.5, cex.lab = 1.3, cex = 1.1)
```

This is a plot of how the analysis looks like (ordinate = ability of the participant, probability = chances of getting the item right; different lines = different stimuli – easy and more difficult ones). Each stimulus has the same discrimination (i.e., they form parallel curves).

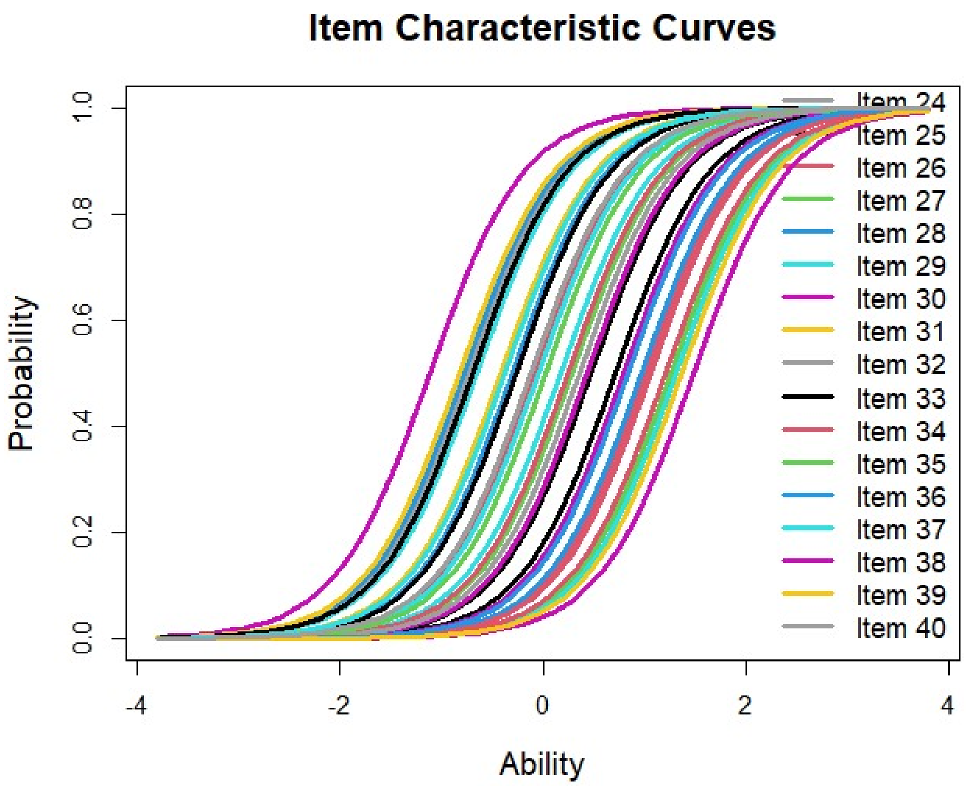


Next we generate new stimuli according to a similar Rasch model. We assume that the 90 participants are equally spread between z-values -4 and +4. We also assume that items are equally spread between -2.5 and +2.5 (data from the previous analysis).

We generate the first 40 stimuli for the non-shared condition.

```
ppar <- runif(90,min = -4, max = 4)
spar <- runif(40,min =-2.5, max = 2.5)
mydata2 <- sim.rasch(ppar, spar)
fit2 <- rasch(mydata2)
plot(fit2, legend = TRUE, cx = "bottomright", lwd = 3, cex.main = 1.5, cex.lab = 1.3, cex = 1.1)
```

This is how the data look like:



Next we generate the second 40 stimuli according to the same equation for the shared condition (i.e., d = .0)

```
d = 0.0
spar <- runif(40,min =-2.5-d, max = 2.5-d)
mydata3 <- sim.rasch(ppar, spar)
```

We also have to define the proficiency of the participants. To some extent, this is given by the variable ppar (ability of the participants). However, if we use this variable in our MME analysis, there is no variance in participant intercepts left. It is also unlikely that we have a proficiency measure that correlates 1.00 with true language proficiency. A correlation of r = .7 seems reasonable. So, we use a proficiency measure that correlates more or less r = .7 with the ability used to generate Rasch. This is done by adding noise to the ppar variable.

```
prof = ppar  + rnorm(90,0,1.7)
```

Now we have everything to generate a data file to be analyzed with MME. For this we define a file output, which has the data of both conditions in long format (not shown here).

For the mixed effects analysis we can use the R lines:

```
fit3 <- glmer(Value ~ Proficiency*Condition + (Condition|Part) +( 1 | Stim), data=output, family=binomial)
summary(fit3)
```

Which gives the following output.

```
Generalized linear mixed model fit by maximum likelihood (Laplace Approximation) ['glmerMod']
 Family: binomial  ( logit )
Formula: Value ~ Proficiency * Condition + (Condition | Part) + (1 | Stim)
   Data: output

    AIC     BIC      logLik     deviance     df.resid
 6102.5  6157.6     -3043.3     6086.5       7192

Scaled residuals:
    Min      1Q   Median      3Q     Max
-11.8982  -0.4441  -0.1260   0.4221   8.8589

Random effects:
 Groups Name            Variance    Std.Dev.    Corr
 Part   (Intercept)     2.129790    1.45938
        Conditionshared 0.003549    0.05957     -1.00
 Stim   (Intercept)     2.381688    1.54327
Number of obs: 7200, groups:  Part, 90; Stim, 80

Fixed effects:
                            Estimate    Std. Error    z value    Pr(>|z|)
(Intercept)                 -0.23920    0.29327       -0.816     0.415
Proficiency                  0.50993    0.05973        8.538     <2e-16 ***
Conditionshared             -0.08749    0.35168       -0.249     0.804
Proficiency:Conditionshared  0.03676    0.03054        1.204     0.229
```

```
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Correlation of Fixed Effects:
            (Intr) Prfcnc Cndtns
Proficiency  0.065
Conditnshrd -0.609 -0.004
Prfcncy:Cnd -0.009 -0.314  0.004
convergence code: 0
Model failed to converge with max|grad| = 0.00269215 (tol = 0.002, component 1)
```

For comparison, this is the jamovi output:

Model Info

Info	Value	Comment
Model Type	Logistic	Model for binary y
Call	glm	Value ~ 1 + Proficiency + Condition + Condition:Proficiency + (1 | Stim) + (1 + Condition | Part)
Link function	Logit	Log of the odd of y=1 over y=0
Direction	P(y=1)/P(y=0)	P(Value =) / P(Value =)
Distribution	Binomial	Dichotomous event distribution of y
LogLikel.	-3043.263	Less is better
R-squared	0.217	Marginal
R-squared	0.666	Conditional
AIC	6102.530	Less is better
BIC	6157.581	Less is better
Deviance	5398.469	Conditional
Residual DF	7192.000	
Converged	yes	
Optimizer	bobyqa	

Note. boundary (singular) fit: see ?isSingular

Fixed Effect Omnibus tests

	X^2	df	p
Proficiency	86.4751	1.00	< .001
Condition	0.0819	1.00	0.775
Condition ✻ Proficiency	1.4486	1.00	0.229

Fixed Effects Parameter Estimates

					95% Exp(B) Confidence Interval			
Names	**Effect**	**Estimate**	**SE**	**exp(B)**	**Lower**	**Upper**	**z**	**p**
(Intercept)	(Intercept)	-0.4723	0.2320	0.624	0.396	0.982	-2.036	0.042
Proficiency	Proficiency	0.5283	0.0568	1.696	1.517	1.896	9.299	< .001
Condition1	shared - nonshared	-0.1007	0.3518	0.904	0.454	1.802	-0.286	0.775
Proficiency ✻ Condition1	Proficiency ✻ shared - nonshared	0.0368	0.0305	1.037	0.977	1.101	1.204	0.229

Random Components

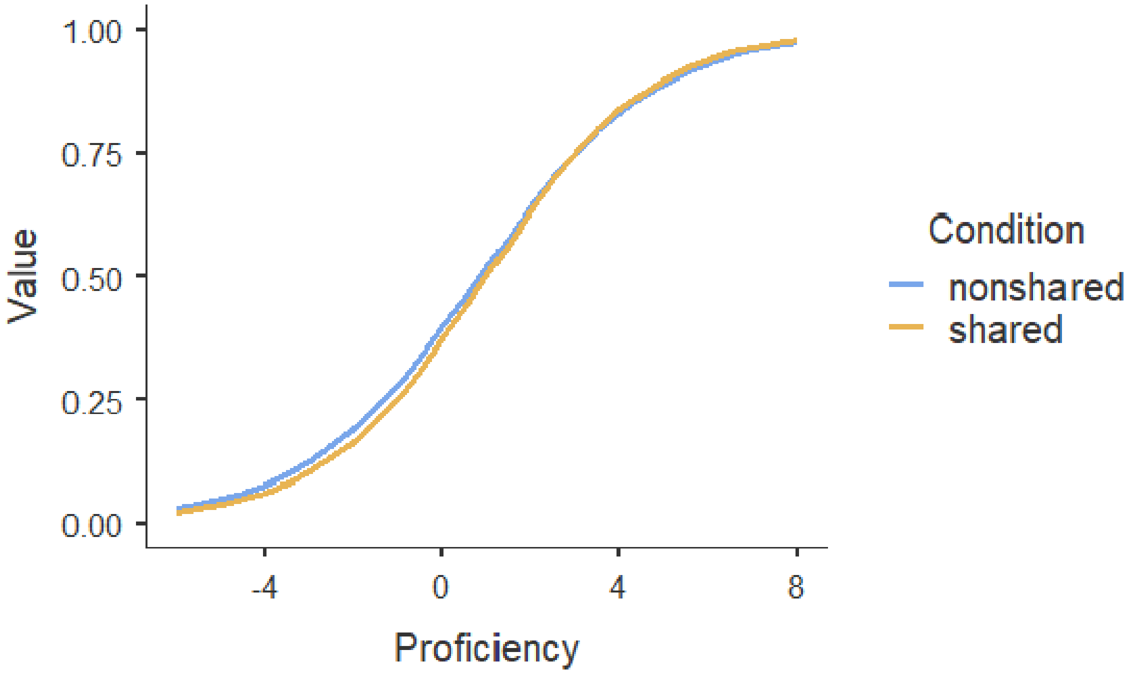
Groups	Name	SD	Variance
Part	(Intercept)	1.4296	2.04379
	Condition1	0.0596	0.00355
Stim	(Intercept)	1.5432	2.38161
Residuals		1.0000	1.00000

Note. Number of Obs: 7200 , groups: Part 90, Stim 80

Random Parameters correlations

Groups	Param.1	Param.2	Corr.
Part	(Intercept)	Condition1	-1.00

Effects Plots



Compare the jamovi output to the output of the experiment itself.

Model Info

Info	Value	Comment
Model Type	Logistic	Model for binary y
Call	glm	response ~ 1 + suffix + sPLexT + suffix:sPLexT + (1 + suffix | id) + (1 | items)
Link function	Logit	Log of the odd of y=1 over y=0

Model Info

Info	Value	Comment
Direction	P(y=1)/P(y=0)	P(response =) / P(response =)
Distribution	Binomial	Dichotomous event distribution of y
LogLikel.	-3142.166	Less is better
R-squared	0.255	Marginal
R-squared	0.687	Conditional
AIC	6300.330	Less is better
BIC	6355.563	Less is better
Deviance	5545.818	Conditional
Residual DF	7352.000	
Converged	yes	
Optimizer	bobyqa	

Fixed Effect Omnibus tests

	X^2	df	p
suffix	0.0971	1.00	0.755
sPLexT	126.7759	1.00	< .001
suffix ✻ sPLexT	0.4753	1.00	0.491

Fixed Effects Parameter Estimates

					95% Exp(B) Confidence Interval			
Names	Effect	Estimate	SE	exp(B)	Lower	Upper	z	p
(Intercept)	(Intercept)	-0.26033	0.23435	0.771	0.487	1.22	-1.111	0.267
suffix1	unshared - shared	-0.11905	0.38207	0.888	0.420	1.88	-0.312	0.755
sPLexT	sPLexT	0.14578	0.01295	1.157	1.128	1.19	11.259	< .001
suffix1 ✻ sPLexT	unshared - shared ✻ sPLexT	0.00598	0.00868	1.006	0.989	1.02	0.689	0.491

Random Components

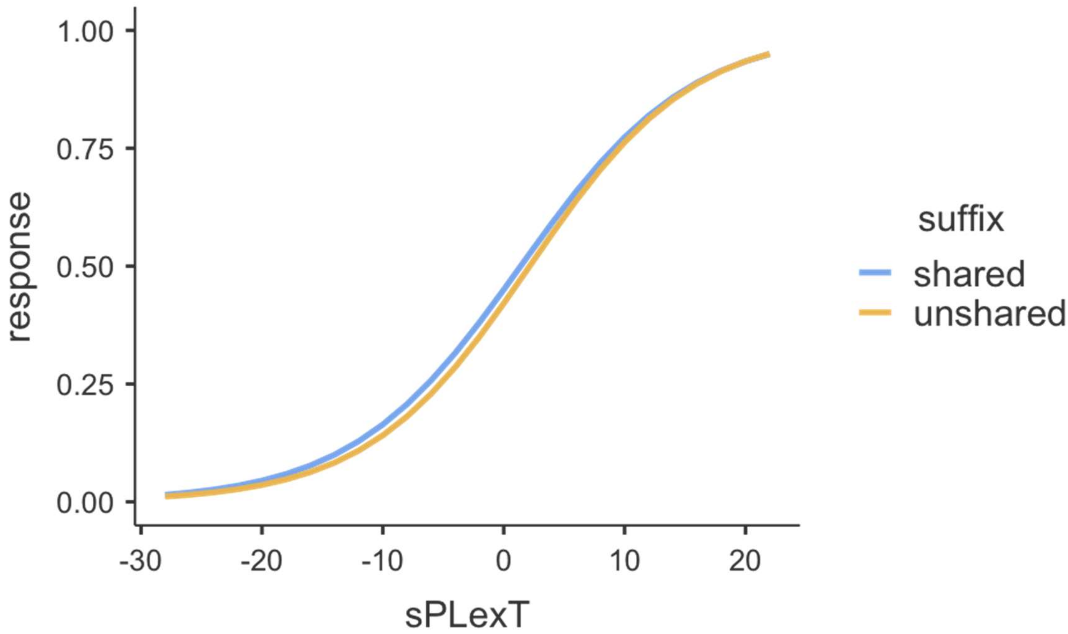
Groups	Name	SD	Variance
id	(Intercept)	1.309	1.714
	suffix1	0.374	0.140
items	(Intercept)	1.670	2.790
Residuals		1.000	1.000

Note. Number of Obs: 7360 , groups: id 92, items 80

Random Parameters correlations

Groups	Param.1	Param.2	Corr.
id	(Intercept)	suffix1	0.237

Effects Plots



Now that we have the basic model, we can run simulations. In the program below, we have 400 datasets simulated and analyzed.

#Simulate and look at the distribution of z-values

```
nSim = 400
zProfi <- numeric(nSim)
zCond <- numeric(nSim)
zInter <- numeric(nSim)

# create progress bar in case it takes a while
pb <- winProgressBar(title = "progress bar", min = 0, max = nSim, width = 300)

for(i in 1:nSim){ #for each simulated experiment
```

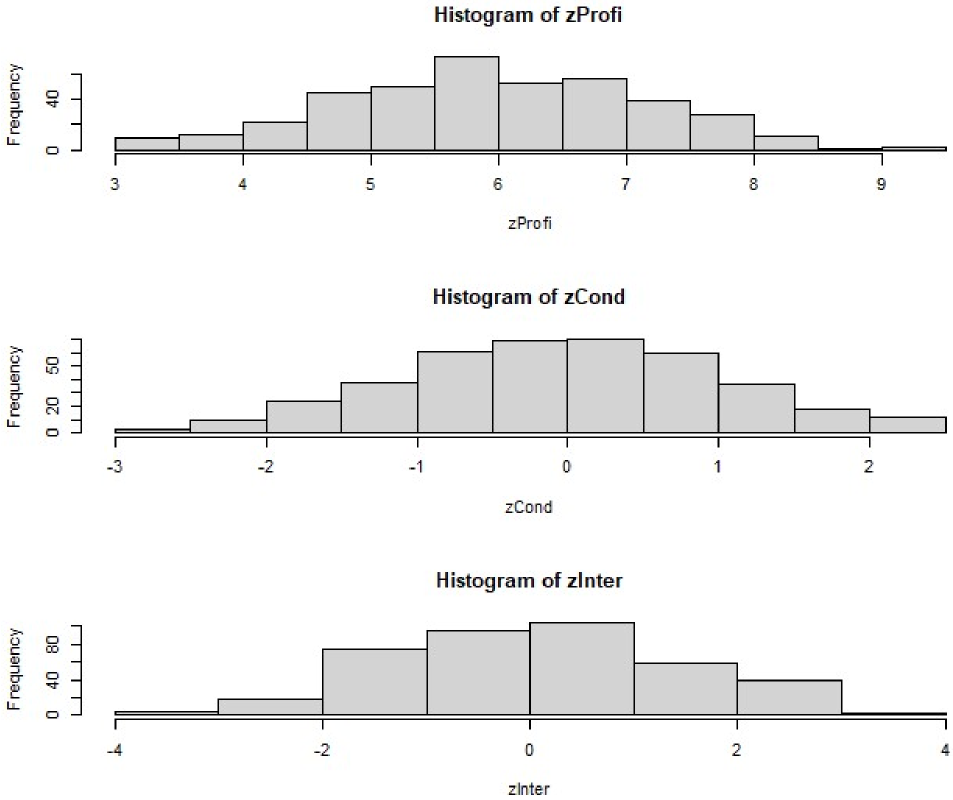
```
  setWinProgressBar(pb, i, title=paste(round(i/nSim*100, 1), "% done"))
  ppar <- runif(90,min = -4, max = 4)
  prof = ppar  + rnorm(90,0,1.7)
  proficiency <- rep(prof, times=40)
  spar <- runif(40,min =-2.5, max = 2.5)
  mydata2 <- sim.rasch(ppar, spar)
  output1 <- melt(mydata2, id.vars=rownames(mydata2))
  condition <- rep("nonshared", times = 40*90)
  output1 <- cbind(output1,proficiency,condition)

  d=0.0
  spar <- runif(40,min = -2.5-d, max = 2.5-d)
  mydata3 <- sim.rasch(ppar, spar)
  output2 <- melt(mydata3, id.vars=rownames(mydata3))
  condition <- rep("shared", times = 40*90)
  output2 <- cbind(output2,proficiency,condition)
  output2[,2] <- output2[,2] + 40

  output <- rbind(output1,output2)
  colnames(output) <- c("Part","Stim", "Value", "Proficiency", "Condition")
  fit3 <- glmer(Value ~ Proficiency*Condition + (Condition|Part) +( 1 | Stim), data=output, family=binomial)
  zProfi[i] <- coef(summary(fit3))[2,"z value"]
  zCond[i] <- coef(summary(fit3))[3,"z value"]
  zInter[i] <- coef(summary(fit3))[4,"z value"]
}
close(pb)#close progress bar

par(mfrow=c(3,1))
hist(zProfi)
hist(zCond)
hist(zInter)
```

This gives the following outcome for proficiency, suffix condition, and the interaction (z-values). We see that the effect of proficiency is significant in each simulation ($z > 1.96$). There is no effect of condition beyond the alpha level (power ≈ .05) or the interaction (power ≈ .05).



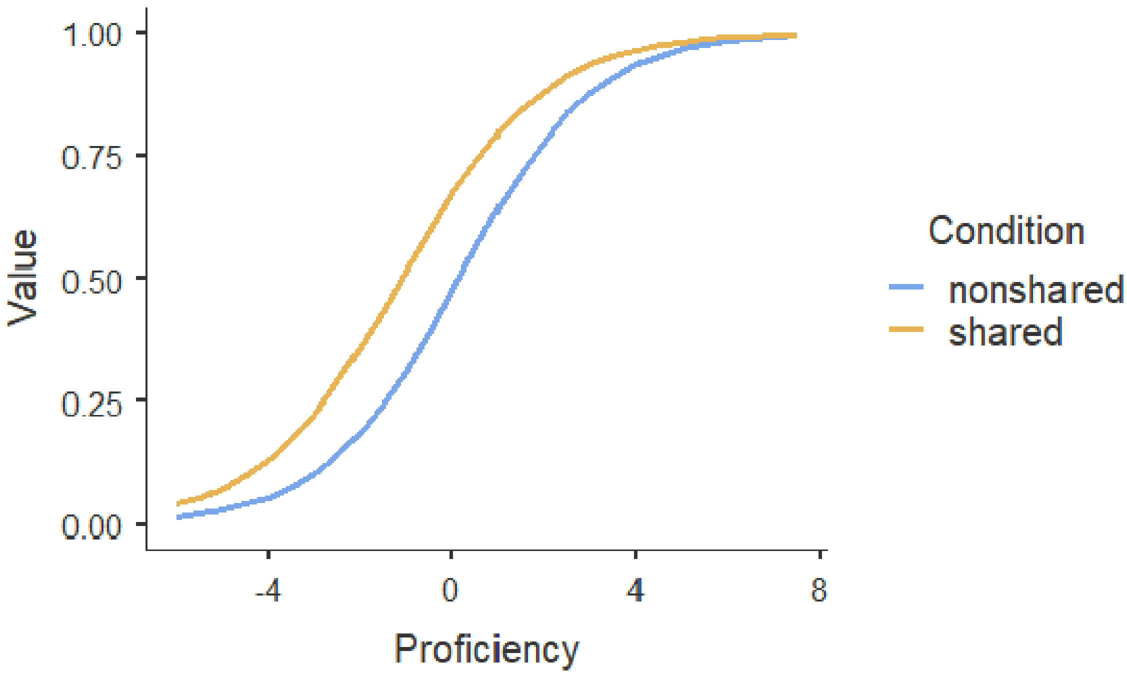
We can now shift the item values in the shared condition by d = .4 or d = .6. Below are the powers we get. Bottom line is that we have 23% chance of finding a shift of d = .4 and 69% to find a shift of d = .8.

Power (p < .05)	**d = .0**	**d = .4**	**d=.8**	**d=1.0**
Proficiency	1.00	1.00	1.00	1.00
Shared	0.05	0.23	0.69	0.87
P * S	0.07	0.02	0.05	0.05
Shared or P*S	0.11	0.25	0.70	0.87

Below you see how the effects look like if the data of a d = 1.0 experiment are analyzed.

d = 1.0

Effects Plots



The programs used are available at https://osf.io/cv8ny/files/ in the folder **Simulations power**.