Power Analysis for CSI online typing with patients with aphasia

Kirsten Stark

06 Dezember, 2021

load packages

```
library(tidyr)
library(dplyr)
library(devtools)
library(MASS)
library(lme4)
library(lmerTest)
library(simr)
library(pbkrtest)
library(testthat)
library(ggplot2)

rm(list = ls())

today <- Sys.Date()
today <- format(today, format="%d%m%y")

# Set number of iterations
n_iter = 1000
```

Load data from Lorenz, Doering, van Scherpenberg, Pino, Abdel Rahman, & Obrig (2021)

In this lab-based study, people with Aphasia did a CSI task with 3 repetitions in two subsequent weeks. Both testing sessions had different items. Additionally, participants also did a CSI task with compounds (not relevant here).
The data loaded here are already cleaned for errors and participants.

```
# load data
df <- read.csv2(here::here("data", "power-analysis",
                "Doeringetal_PWA_naming_simple_nouns_data_for_modelling.csv"))

# subset the relevant columns
df <- df %>%
  filter(error==1) %>%
    # repet_trig = repetition (151,152,153 is 1,2,3),
```

```
    # wh = testing session 1 and 2,
    # cat_nr = 18 categories (à 5 members each)
  dplyr::select(c(subject, Ordinal_position, RT,
                  repet_trig, wh, item_id, cat_nr)) %>%
  droplevels() %>%
  dplyr::rename(repet = repet_trig,
                OrdPos = Ordinal_position) %>%
  # factorize colums
  mutate(subject = as.factor(subject),
         repet = as.factor(repet),
         wh = as.factor(wh),
         item_id = as.factor(item_id),
         cat_nr = as.factor(cat_nr))
```

The data structure is somewhat different from the planned experiment. Our experiment will have no repetition, but three testing sessions with the same items. Does the variance differ between the repetitions within a session and the two testing sessions?

```
df %>% group_by(wh, repet, OrdPos) %>%
  summarize(sd = sd(RT, na.rm = T))
```

```
## `summarise()` has grouped output by 'wh', 'repet'. You can override using the `.groups` argument.
```

```
## # A tibble: 30 x 4
## # Groups:   wh, repet [6]
##    wh    repet OrdPos    sd
##    <fct> <fct> <chr>  <dbl>
##  1 1     151   OP1     694.
##  2 1     151   OP2     896.
##  3 1     151   OP3     929.
##  4 1     151   OP4     706.
##  5 1     151   OP5    1048.
##  6 1     152   OP1     830.
##  7 1     152   OP2     907.
##  8 1     152   OP3     726.
##  9 1     152   OP4     766.
## 10 1     152   OP5     926.
## # ... with 20 more rows
```

```
df %>% group_by(wh)%>% summarize(sd = sd(RT, na.rm = T))
```

```
## # A tibble: 2 x 2
##   wh       sd
##   <fct> <dbl>
## 1 1      859.
## 2 2      958.
```

```
df %>% group_by(repet)%>% summarize(sd = sd(RT, na.rm = T))
```

```
## # A tibble: 3 x 2
##   repet    sd
```

```
##   <fct> <dbl>
## 1 151   1003.
## 2 152    865.
## 3 153    857.
```

```
df %>% group_by(wh, repet)%>%summarize(sd = sd(RT, na.rm = T))
```

```
## `summarise()` has grouped output by 'wh'. You can override using the `.groups` argument.
```

```
## # A tibble: 6 x 3
## # Groups:   wh [2]
##   wh    repet    sd
##   <fct> <fct> <dbl>
## 1 1     151    870.
## 2 1     152    833.
## 3 1     153    872.
## 4 2     151   1112.
## 5 2     152    895.
## 6 2     153    840.
```

```
df %>% group_by(OrdPos)%>%summarize(sd = sd(RT,na.rm=T))
```

```
## # A tibble: 5 x 2
##   OrdPos    sd
##   <chr>  <dbl>
## 1 OP1     776.
## 2 OP2     880.
## 3 OP3     944.
## 4 OP4     968.
## 5 OP5     971.
```

Overall, the variances are not too different (but still). The variance in the second session seems to be somewhat higher and the variances seem to increase with ordinal position.

Comparison 1: Power analysis for the ordinal position effect

Subset the data to the first session and first repetition

```
df_Ord <- df %>% filter(wh == 1, repet == 151) %>% droplevels()
```

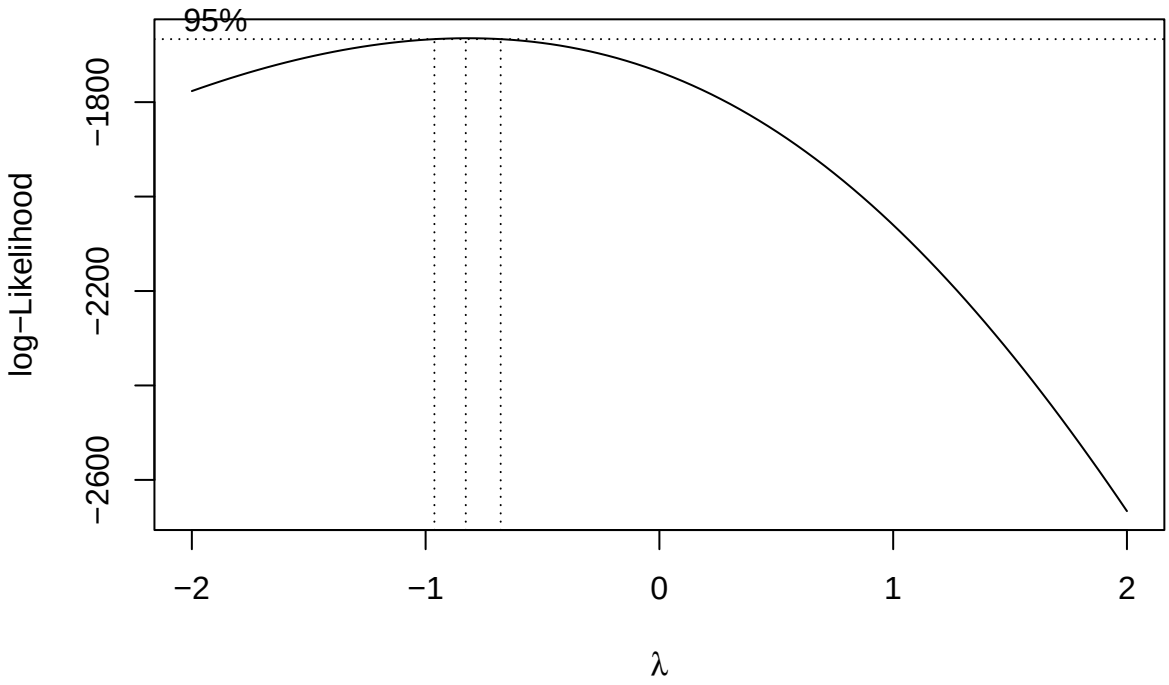
1) Set up models based on the structure of the online CSI experiment

Unfortunately, simr does not work properly with GLMMs. Therefore, for the power analysis, we will set up an LMM with transformed RTs
a) center OrdPos (continuous predictor)

```
# center continuous predictor
df_Ord %>% mutate(OrdPos_num = case_when(OrdPos == "OP1" ~1,
                                         OrdPos == "OP2" ~2,
                                         OrdPos == "OP3" ~3,
                                         OrdPos == "OP4" ~4,
                                         OrdPos == "OP5" ~5)) %>%
                  mutate(OrdPos.c=
                    scale(as.numeric(as.character(OrdPos_num)),
                     center = TRUE, scale = FALSE)) -> df_Ord
```

b) Check distribution of RTs

```
# Boxcox plot suggests inverse transformation:
boxcox(df_Ord$RT ~ df_Ord$OrdPos)
```




```
# # check distribution of RTs (by eyeballing)
# # 1) density plot of RTs
# qplot(data=df_Ord, RT, geom="density", na.rm=TRUE)+ theme_bw()
# # 2) plot data against real normal distribution -> is it way off?
# qqnorm(df_Ord$RT); qqline(df_Ord$RT)
#
# # check distribution of logRTs (by eyeballing)
# df_Ord$lRT <- log(df_Ord$RT)
# # 1) density plot of logRTs
```

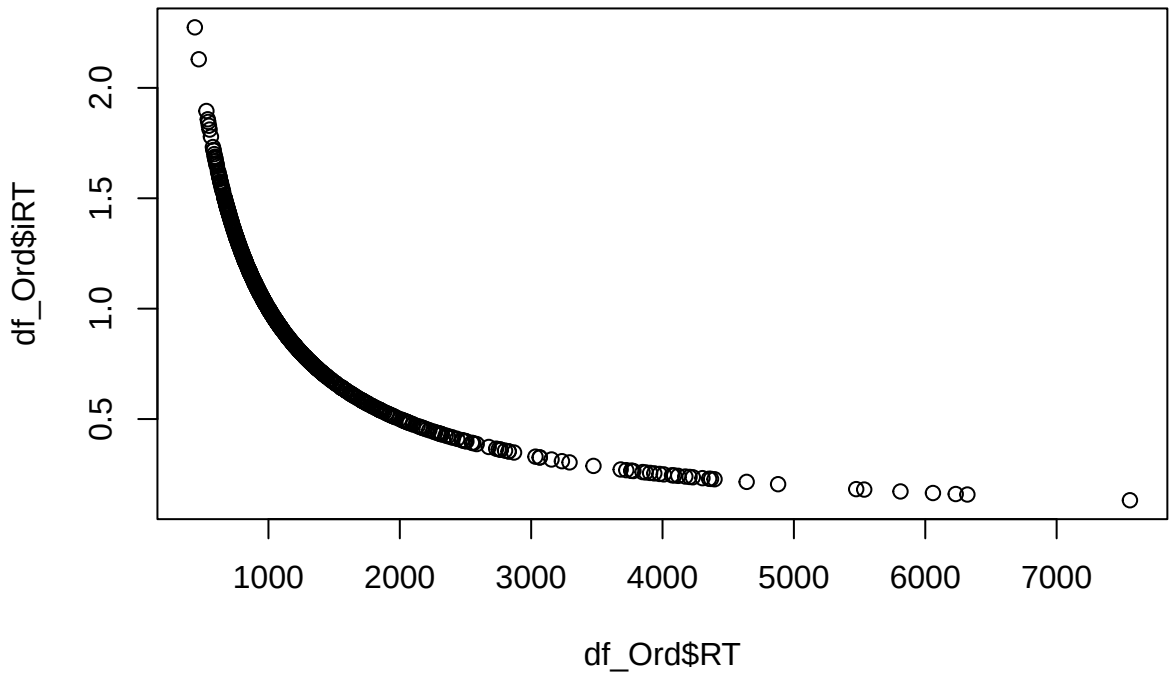
```
# qplot(data=df_Ord, lRT, geom="density", na.rm=TRUE)+ theme_bw()
# # 2) plot data against real normal distribution -> is it way off?
# qqnorm(df_Ord$lRT); qqline(df_Ord$lRT)
# ### data kind of normally distributed. Log RTs fit better
#
# # check distribution of 1/RT (by eyeballing)
# df_Ord$iRT <- 1/df_Ord$RT
# # 1) density plot of logRTs
# qplot(data=df_Ord, iRT, geom="density", na.rm=TRUE)+ theme_bw()
# # 2) plot data against real normal distribution -> is it way off?
# qqnorm(df_Ord$iRT); qqline(df_Ord$iRT)
# ### data kind of normally distributed. Inverse RTs fit well
```

Check the goodness of fit of an inverse transformation (with RT divided by 1000: 1/(x/1000) = 1000/x):

```
library(fitdistrplus)
```

```
## Lade nötiges Paket: survival
```

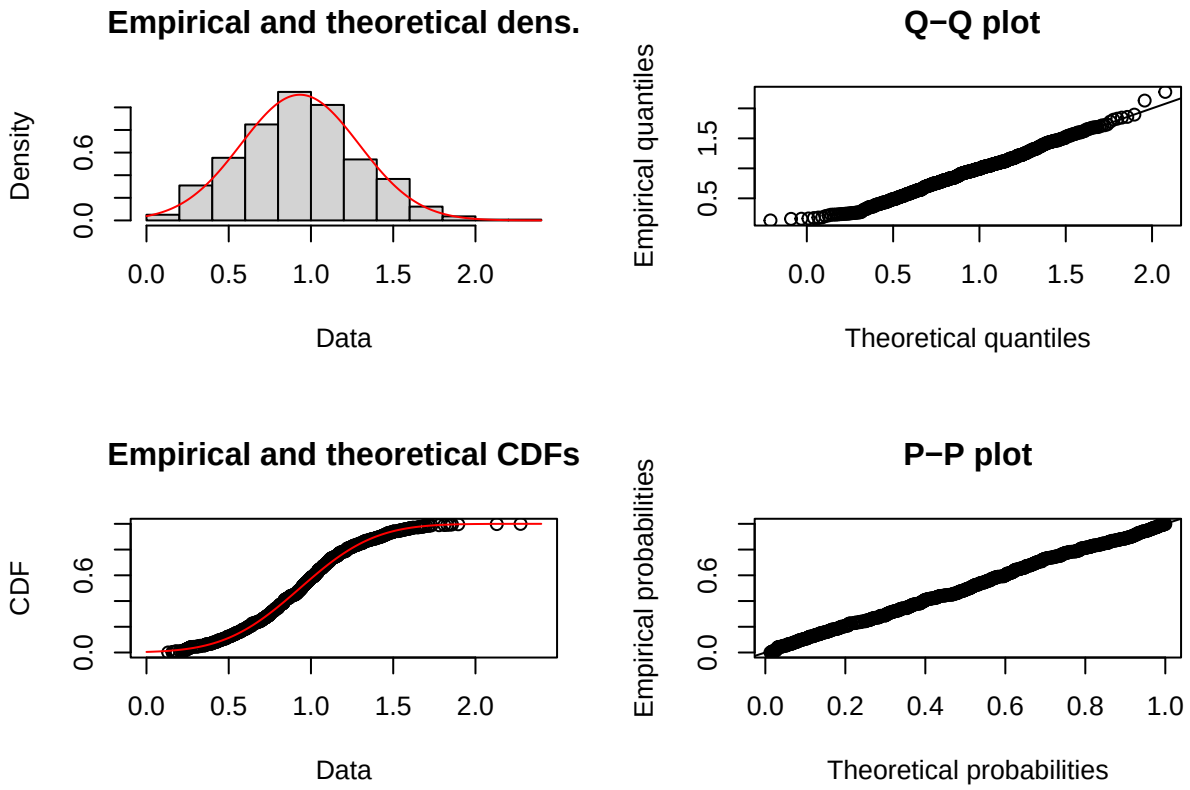
```
# fit.normal<- fitdist(df_Ord$RT, distr = "norm", method = "mle")
# summary(fit.normal)
# plot(fit.normal)
df_Ord$iRT <-1000/df_Ord$RT
# plot this transformation distribution
plot(df_Ord$RT, df_Ord$iRT)
```



```r
# check fit
fit.normal_inv<- fitdist(df_Ord$iRT, distr = "norm", method = "mle")
summary(fit.normal_inv)
```

```
## Fitting of the distribution ' norm ' by maximum likelihood
## Parameters :
##       estimate  Std. Error
## mean 0.9329607 0.013612620
## sd   0.3588675 0.009625239
## Loglikelihood:  -273.9248   AIC:  551.8497   BIC:  560.9375
## Correlation matrix:
##      mean sd
## mean    1  0
## sd      0  1
```

```r
plot(fit.normal_inv)
```



An inverse transformation fits the data well. We will thus use this kind of transformation

c) Set up the models
Model 1: Linear model with continuous predictor "Ordinal position" and inversely transformed RT data

```r
lmm1 <- lmer(iRT ~ OrdPos.c + (OrdPos.c|subject) +(OrdPos.c|cat_nr) ,
             data = df_Ord, REML = FALSE,
```

```
            control=lmerControl(optimizer = "bobyqa"))
isSingular(lmm1)
```

```
## [1] FALSE
```

```
summary(lmm1)
```

```
## Linear mixed model fit by maximum likelihood . t-tests use Satterthwaite's
##   method [lmerModLmerTest]
## Formula: iRT ~ OrdPos.c + (OrdPos.c | subject) + (OrdPos.c | cat_nr)
##    Data: df_Ord
## Control: lmerControl(optimizer = "bobyqa")
##
##      AIC      BIC   logLik deviance df.resid
##    354.3    395.2   -168.1    336.3      686
##
## Scaled residuals:
##      Min       1Q   Median       3Q      Max
## -2.94617 -0.64056  0.07423  0.61505  3.04551
##
## Random effects:
##  Groups   Name        Variance  Std.Dev. Corr
##  subject  (Intercept) 0.0334551 0.18291
##           OrdPos.c    0.0003164 0.01779  -0.58
##  cat_nr   (Intercept) 0.0106732 0.10331
##           OrdPos.c    0.0003476 0.01864  -0.24
##  Residual             0.0840919 0.28999
## Number of obs: 695, groups:  subject, 18; cat_nr, 17
##
## Fixed effects:
##             Estimate Std. Error       df t value Pr(>|t|)
## (Intercept)  0.92215    0.05157 25.89346  17.881 4.33e-16 ***
## OrdPos.c    -0.03048    0.01013 11.39100  -3.008   0.0115 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Correlation of Fixed Effects:
##          (Intr)
## OrdPos.c -0.259
```

2) Extend dataset

We already know that we want 24 categories (we use the same stimuli as in Stark, van Scherpenberg et al., 2021). So we extend along categories.
Additionally, we also extend along participants

```
lmm1_2 <- extend(lmm1, along="cat_nr", n=24)
m2data <- getData(lmm1_2)
## ok, data were indeed extended to n categories ;-)
str(m2data)
```

```
## 'data.frame':    1056 obs. of  10 variables:
##  $ subject   : Factor w/ 18 levels "1","2","3","4",..: 1 1 1 1 2 2 2 3 3 3 ...
##  $ OrdPos    : chr  "OP5" "OP1" "OP2" "OP3" ...
##  $ RT        : num  7558 735 678 860 1208 ...
##  $ repet     : Factor w/ 1 level "151": 1 1 1 1 1 1 1 1 1 1 ...
##  $ wh        : Factor w/ 1 level "1": 1 1 1 1 1 1 1 1 1 1 ...
##  $ item_id   : Factor w/ 84 levels "1","2","3","4",..: 20 16 17 18 18 20 16 18 19 20 ...
##  $ cat_nr    : Factor w/ 24 levels "a","b","c","d",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ OrdPos_num: num  5 1 2 3 2 4 5 1 2 3 ...
##  $ OrdPos.c  : num [1:1056, 1] 2.0201 -1.9799 -0.9799 0.0201 -0.9799 ...
##   ..- attr(*, "dimnames")=List of 2
##   .. ..$ : NULL
##   .. ..$ : NULL
##  $ iRT       : num  0.132 1.36 1.476 1.163 0.828 ...
```

```
str(df_Ord)
```

```
## 'data.frame':    695 obs. of  10 variables:
##  $ subject   : Factor w/ 18 levels "1","2","3","4",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ OrdPos    : chr  "OP5" "OP1" "OP1" "OP1" ...
##  $ RT        : num  7558 652 735 728 552 ...
##  $ repet     : Factor w/ 1 level "151": 1 1 1 1 1 1 1 1 1 1 ...
##  $ wh        : Factor w/ 1 level "1": 1 1 1 1 1 1 1 1 1 1 ...
##  $ item_id   : Factor w/ 84 levels "1","2","3","4",..: 20 1 16 50 25 55 2 26 56 51 ...
##  $ cat_nr    : Factor w/ 17 levels "1","2","3","4",..: 4 1 4 11 6 12 1 6 12 11 ...
##  $ OrdPos_num: num  5 1 1 1 1 1 2 2 2 2 ...
##  $ OrdPos.c  : num [1:695, 1] 2.02 -1.98 -1.98 -1.98 -1.98 ...
##   ..- attr(*, "scaled:center")= num 2.98
##  $ iRT       : num  0.132 1.533 1.36 1.373 1.811 ...
```

```
lmm24_18 <- extend(lmm1_2, along="subject", n=18)
 m2data <- getData(lmm24_18)
# ## ok, data were indeed extended to n subjects ;-)
str(m2data)
```

```
## 'data.frame':    1056 obs. of  10 variables:
##  $ subject   : Factor w/ 18 levels "a","b","c","d",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ OrdPos    : chr  "OP5" "OP1" "OP2" "OP3" ...
##  $ RT        : num  7558 735 678 860 652 ...
##  $ repet     : Factor w/ 1 level "151": 1 1 1 1 1 1 1 1 1 1 ...
##  $ wh        : Factor w/ 1 level "1": 1 1 1 1 1 1 1 1 1 1 ...
##  $ item_id   : Factor w/ 84 levels "1","2","3","4",..: 20 16 17 18 1 2 3 4 5 50 ...
##  $ cat_nr    : Factor w/ 24 levels "a","b","c","d",..: 1 1 1 1 2 2 2 2 2 3 ...
##  $ OrdPos_num: num  5 1 2 3 1 2 3 4 5 1 ...
##  $ OrdPos.c  : num [1:1056, 1] 2.0201 -1.9799 -0.9799 0.0201 -1.9799 ...
##   ..- attr(*, "dimnames")=List of 2
##   .. ..$ : NULL
##   .. ..$ : NULL
##  $ iRT       : num  0.132 1.36 1.476 1.163 1.533 ...
```

```
# str(df_Ord)
```

```
lmm24_30 <- extend(lmm1_2, along="subject", n=30)
 m2data <- getData(lmm24_30)
# ## ok, data were indeed extended to n subjects ;-)
str(m2data)
```

```
## 'data.frame':    1763 obs. of  10 variables:
##  $ subject   : Factor w/ 30 levels "a","b","c","d",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ OrdPos    : chr  "OP5" "OP1" "OP2" "OP3" ...
##  $ RT        : num  7558 735 678 860 652 ...
##  $ repet     : Factor w/ 1 level "151": 1 1 1 1 1 1 1 1 1 1 ...
##  $ wh        : Factor w/ 1 level "1": 1 1 1 1 1 1 1 1 1 1 ...
##  $ item_id   : Factor w/ 84 levels "1","2","3","4",..: 20 16 17 18 1 2 3 4 5 50 ...
##  $ cat_nr    : Factor w/ 24 levels "a","b","c","d",..: 1 1 1 1 2 2 2 2 2 3 ...
##  $ OrdPos_num: num  5 1 2 3 1 2 3 4 5 1 ...
##  $ OrdPos.c  : num [1:1763, 1] 2.0201 -1.9799 -0.9799 0.0201 -1.9799 ...
##   ..- attr(*, "dimnames")=List of 2
##   .. ..$ : NULL
##   .. ..$ : NULL
##  $ iRT       : num  0.132 1.36 1.476 1.163 1.533 ...
```

```
# str(df_Ord)
```

To be sure nothing went wrong with the category extension, we do another power analysis for 17 categories (category 5 was deleted whilst subsetting the data):

```
lmm17_30 <- extend(lmm1, along="subject", n=30)
 m2data <- getData(lmm17_30)
# ## ok, data were indeed extended to n subjects ;-)
str(m2data)
```

```
## 'data.frame':    1147 obs. of  10 variables:
##  $ subject   : Factor w/ 30 levels "a","b","c","d",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ OrdPos    : chr  "OP5" "OP1" "OP1" "OP1" ...
##  $ RT        : num  7558 652 735 728 552 ...
##  $ repet     : Factor w/ 1 level "151": 1 1 1 1 1 1 1 1 1 1 ...
##  $ wh        : Factor w/ 1 level "1": 1 1 1 1 1 1 1 1 1 1 ...
##  $ item_id   : Factor w/ 84 levels "1","2","3","4",..: 20 1 16 50 25 55 2 26 56 51 ...
##  $ cat_nr    : Factor w/ 17 levels "1","2","3","4",..: 4 1 4 11 6 12 1 6 12 11 ...
##  $ OrdPos_num: num  5 1 1 1 1 1 2 2 2 2 ...
##  $ OrdPos.c  : num [1:1147, 1] 2.02 -1.98 -1.98 -1.98 -1.98 ...
##   ..- attr(*, "dimnames")=List of 2
##   .. ..$ : NULL
##   .. ..$ : NULL
##  $ iRT       : num  0.132 1.533 1.36 1.373 1.811 ...
```

```
# str(df_Ord)
```

3a) Specify effect sizes: Use effect from Stark et al 2021

As a first attempt, we use the effect size from Stark, van Scherpenberg et al., 2021, Exp. 1 as this is the smallest effect size that we have: 31 ms (We take the most conservative calculation)

Model 1: Linear model with continuous predictor "Ordinal position" and inversely transformed RT data

What effect size is 31 ms in 1/(RT/1000) depends on the position: We take the grand mean

```
mean(df_Ord$RT)
```

```
## [1] 1337.724
```

```
# calculate the effect size in the transformed scale
Ord1 <- 1000/(mean(df_Ord$RT))
Ord2 <- 1000/(mean(df_Ord$RT)+31)
eff1 <- Ord2 - Ord1
```

```
# use this as the fixed effect of interest
fixef(lmm24_30)
```

```
## (Intercept)    OrdPos.c
##  0.92214551 -0.03048293
```

```
fixef(lmm24_30)["OrdPos.c"] <- eff1
```

4a) Run the power analysis based on effect size from Stark et al (2021)

```
set.seed(99)
```

Model 1: Linear model with continuous predictor "Ordinal position" and inversely transformed RT data

```
PowerLMM1_Stark <- powerSim(lmm24_30, test=fixed("OrdPos.c","t"), nsim = n_iter) # increase to nsim > 1
```

```
#lastResult()$warnings
#lastResult()$errors
```

```
PowerLMM1_Stark
```

```
## Power for predictor 'OrdPos.c', (95% confidence interval):
##       61.20% (58.10, 64.23)
##
## Test: t-test with Satterthwaite degrees of freedom (package lmerTest)
##       Effect size for OrdPos.c is -0.017
##
## Based on 1000 simulations, (7 warnings, 0 errors)
## alpha = 0.05, nrow = 1763
##
## Time elapsed: 0 h 6 m 11 s
```

3b) Specify effect sizes: Use effect from Lorenz et al (2021)

Now we use the experimental effect size from Lorenz et al (2021), but with 24 categories

```
fixef(lmm24_18)
```

```
## (Intercept)    OrdPos.c
##  0.92214551 -0.03048293
```

4b) Run the power analysis based on effect size from Lorenz et al (2021)

```
set.seed(99)
```

Model 1: Linear model with continuous predictor "Ordinal position" and inversely transformed RT data

```
PowerLMM1_Lorenz <- powerSim(lmm24_18, test=fixed("OrdPos.c","t"), nsim = n_iter) # increase to nsim >
lastResult()$warnings
lastResult()$errors
```

```
#lastResult()$warnings
#lastResult()$errors
```

```
PowerLMM1_Lorenz
```

```
## Power for predictor 'OrdPos.c', (95% confidence interval):
##       92.70% (90.91, 94.23)
##
## Test: t-test with Satterthwaite degrees of freedom (package lmerTest)
##       Effect size for OrdPos.c is -0.030
##
## Based on 1000 simulations, (2 warnings, 0 errors)
## alpha = 0.05, nrow = 1056
##
## Time elapsed: 0 h 4 m 16 s
##
## nb: result might be an observed power calculation
```

3c) Specify effect sizes: Use effect from Stark et al 2021

Now we use the effect size from Stark Stark, van Scherpenberg et al., 2021, Exp. 1, but with the original 17 categories.

Model 1: Linear model with continuous predictor "Ordinal position" and inversely transformed RT data

What effect size is 31 ms in 1/(RT/1000) depends on the position: We take the grand mean

```
# use this as the fixed effect of interest
fixef(lmm17_30)
```

```
## (Intercept)    OrdPos.c
##  0.92214551 -0.03048293
```

```
fixef(lmm17_30)["OrdPos.c"] <- eff1
```

4c) Run the power analysis based on effect size from Stark et al (2021) with 17 categories

```
set.seed(99)
```

Model 1: Linear model with continuous predictor "Ordinal position" and inversely transformed RT data

```
PowerLMM1_Stark_17cat <- powerSim(lmm17_30, test=fixed("OrdPos.c","t"), nsim = n_iter) # increase to ns
```

```
#lastResult()$warnings
#lastResult()$errors
```

```
PowerLMM1_Stark_17cat
```

```
## Power for predictor 'OrdPos.c', (95% confidence interval):
##       47.10% (43.97, 50.25)
##
## Test: t-test with Satterthwaite degrees of freedom (package lmerTest)
##       Effect size for OrdPos.c is -0.017
##
## Based on 1000 simulations, (7 warnings, 0 errors)
## alpha = 0.05, nrow = 1147
##
## Time elapsed: 0 h 4 m 28 s
```

5a) Power analyses at a range of sample sizes with effect from Stark, van Scherpenberg et al, 2021

Try out different sample sizes:

```
lmm24_60 <- extend(lmm1_2, along="subject", n=60)
fixef(lmm24_60)["OrdPos.c"] <- eff1
pc_lmm24_60 <- powerCurve(lmm24_60, along = "subject",
                          breaks = c(20, 24, 30, 40, 60), nsim = n_iter)
lastResult()$warnings
lastResult()$errors
```

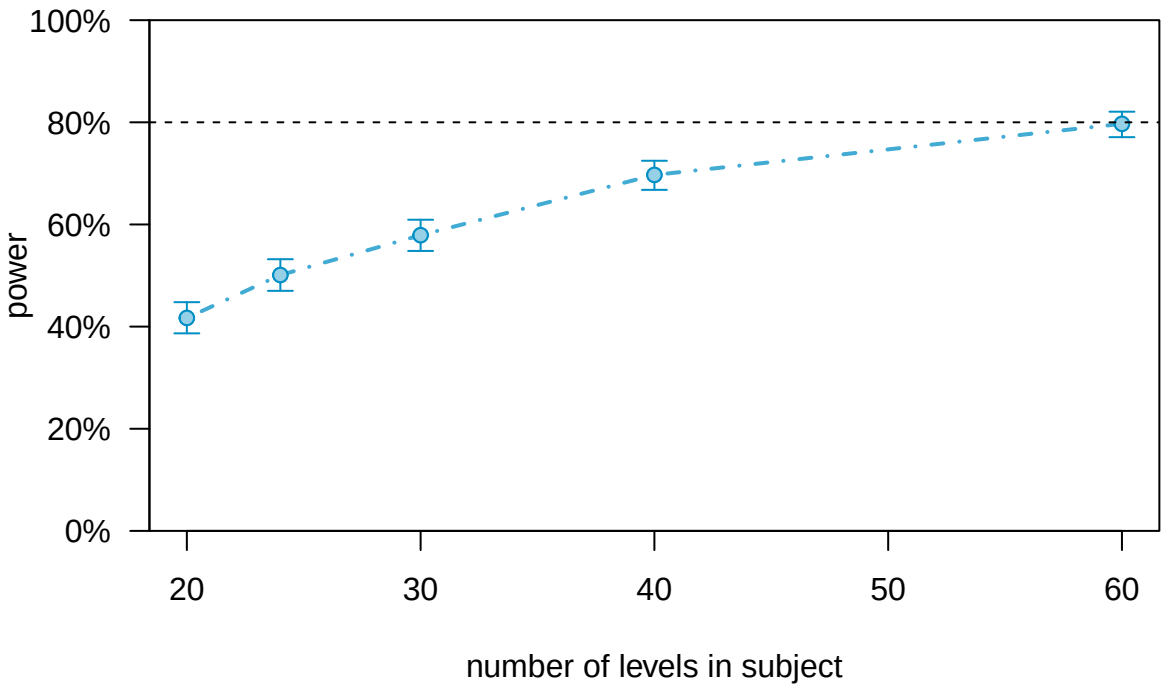
```
#lastResult()$warnings
#lastResult()$errors
```

```
pc_lmm24_60
```

```
## Power for predictor 'OrdPos.c', (95% confidence interval),
## by number of levels in subject:
##      20: 41.70% (38.62, 44.83) - 1193 rows
##      24: 50.10% (46.95, 53.24) - 1437 rows
```

```
##      30: 57.90% (54.77, 60.98) - 1763 rows
##      40: 69.70% (66.75, 72.54) - 2391 rows
##      60: 79.70% (77.07, 82.15) - 3549 rows
##
## Time elapsed: 12 h 13 m 9 s
```

```
plot(pc_lmm24_60)
```



5b) Power analyses at a range of sample sizes with effect size from Lorenz et al (2021)

Try out different sample sizes)

Several sample sizes

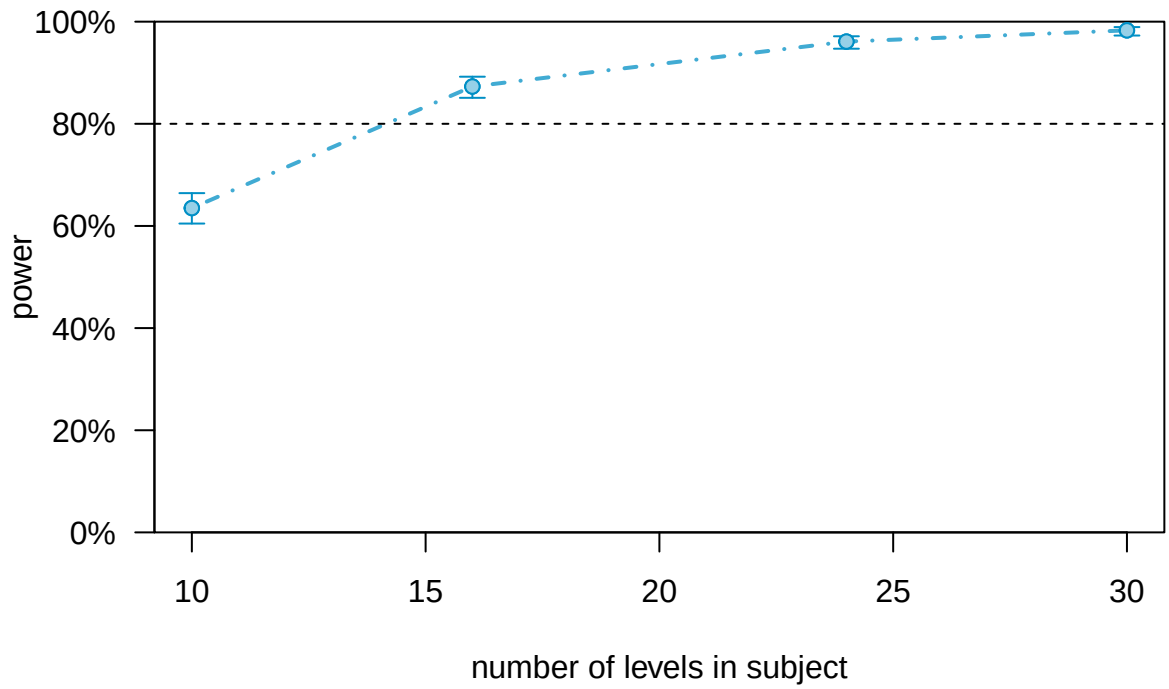
```
lmm24_18 <- extend(lmm1_2, along="subject", n=30)
pc_lmm24_18 <- powerCurve(lmm24_18, along = "subject",
                          breaks = c(10, 16, 24, 30), nsim = n_iter)
```

```
#lastResult()$warnings
#lastResult()$errors
```

```
pc_lmm24_18
```

```
## Power for predictor 'OrdPos.c', (95% confidence interval),
## by number of levels in subject:
##       10: 63.50% (60.43, 66.49) - 579 rows
##       16: 87.30% (85.08, 89.30) - 942 rows
##       24: 96.10% (94.71, 97.21) - 1437 rows
##       30: 98.30% (97.29, 99.01) - 1763 rows
##
## Time elapsed: 2 h 24 m 3 s
```

```
plot(pc_lmm24_18)
```



write results

```
# powersim_results_cont <- rbind(summary(PowerLMM1),
#                           summary(powersim_24_cont_log),
#                           summary(powersim_40_cont_log_015),
#                           summary(powersim_24_cont_log_015),
#                           summary(powersim_40_cont),
#                           summary(powersim_24_cont),
#                           summary(powersim_40_cont_18ms),
#                           summary(powersim_24_cont_18ms),
#                           summary(powersim_30_cont_18ms))
#
```

```r
# powersim_results_cont <- cbind(powersim_results_cont, n = c(40,24,40,24,40,24,40,24,30), RT_transform
#
# write.table(powersim_results_cont, "powersim_results_cont.csv", append = FALSE, sep = ";", row.names =
```