ACRM 2023 Longitudinal Data Analysis Workshop

Keith Lohse, PhD, PStat, and Allan Kozlowski, PhD, BsC (PT)

Practical Session 4: Simulation-based Approaches to Statistical Power in Mixed-Effect Models

1.A Refresher about Statistical Power

In previous sessions, we explored how to visualize longitudinal data, how to build mixed-effect regression models, and how to conduct statistical significance tests in the context of linear and non-linear mixed-effect regression. In this session, we will focus on statistical power as a complement to null-hypothesis testing.

As before, we will use several packages related to data management, visualization, and mixed-effect models.

```
# Loading the essential libraries.
library("tidyverse"); library("lme4"); library("MASS")
library("car"); library("lmerTest");library("nlme"); library("patchwork")
```

If you have not already installed these packages, you will need to use the install.packages() function first. This can take some time and will require an internet connection.

```
# If these packages are not installed already, run the following code:
install.packages("tidyverse"); install.packages("lme4");
install.packages("car"); install.packages("lmerTest"); install.packages("nlme")
```

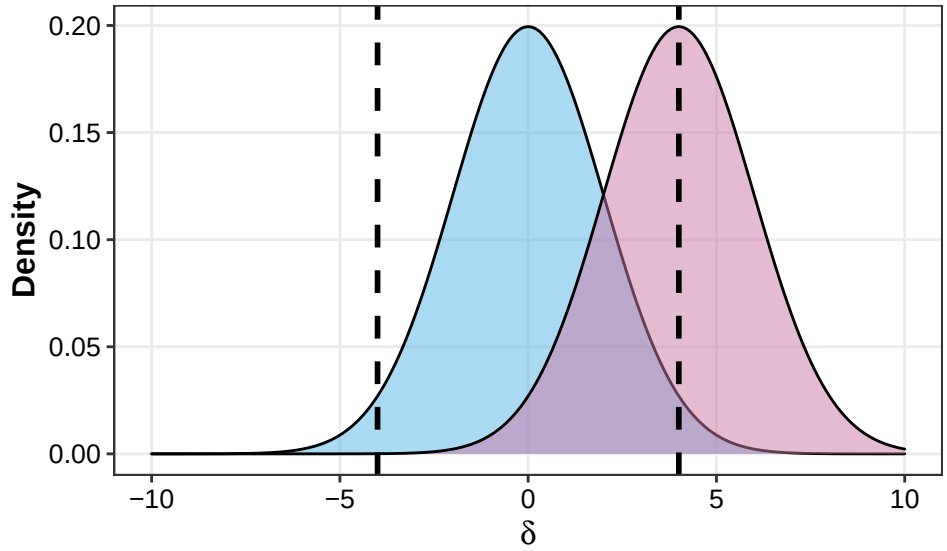
Before we get into the details of hypothesis testing with longitudinal data, let's briefly refresh the concept of statistical power. Calculating statistical power requires us to first figure out what would happen under the *null hypothesis*. In this case we assume that the true mean difference is zero (H0: δ=0). If the null hypothesis were true, we would expect a distribution of mean differences like the one in blue. The distribution is centered on 0, but the difference we observe will vary from study to study. Specifically, 95% of the null distribution is between the vertical black lines, putting 2.5% in each tail. If a mean difference falls outside of those black lines (i.e., p<0.05 if the null is true), then we would call the result "statistically significant" and reject the null hypothesis. We use α=0.05 as a cutoff by convention, but this limit could be set to any value and conventions will vary by scientific discipline. Critically though, by setting this α threshold allows us to control the Type I error rate. That is, if we set α=0.05, that means we will only make a mistake and declare an effect statistically significant 5% time when the null is true (i.e., we will make "false alarms" 5% of the time).

However, we also need to consider what might happen if the null is *not* true. For instance, what if the true mean difference was δ=4 instead of δ=0? If this *alternative hypothesis* was true, then we would expect a distribution of mean differences like the one in red. Under this assumption, 50% of our results are above the statistical significance threshold and 50% are below the threshold. Thus, if δ=4 is true, we will fail to find a statistically significant effect 50% of the time (i.e., we will miss 50% of the time). This miss rate is referred to as the Type II error rate and represented by β.

Statistical power is 1-β. Put another way, statistical power is the probability of obtaining a statistically significant result if a given alternative hypothesis is true. In the figure, statistical power is not very good, because if the true mean difference is δ=4, we will only correctly reject the null hypothesis 50% of the time!

```r
differences <- c(seq(-10, 10, 0.1))
null_dist <- dnorm(x=differences, mean=0, sd=2)
alt_dist <- dnorm(x=differences, mean=4, sd=2)

DATA <- data.frame(differences, null_dist, alt_dist)
```

```r
cbPalette <- c("#999999", "#E69F00", "#56B4E9", "#009E73", "#CC79A7", "#F0E442",
               "#555555",  "#D55E00","#0072B2", "#03c478", "#661100", "#f3f319",
               "#222222","#FBD7A2", "#6699CC", "#99edcc", "#b804a2", "#F9E999")

ggplot(data=DATA, aes(x=differences)) +
  geom_density(aes(y=null_dist), fill=cbPalette[3], stat="identity", alpha=0.5)+
  geom_density(aes(y=alt_dist), fill=cbPalette[5], stat="identity", alpha=0.5)+
  geom_vline(xintercept = c(-4, 4), lty=2, lwd=1, col="black")+
  scale_x_continuous(name = expression(delta)) +
  scale_y_continuous(name = "Density") +
  theme_bw()+
  theme(axis.text=element_text(size=10, color="black"),
        legend.text=element_text(size=10, color="black"),
        legend.title=element_text(size=10, face="bold"),
        axis.title=element_text(size=12, face="bold"),
        plot.title=element_text(size=12, face="bold", hjust=0.5),
        panel.grid.minor = element_blank(),
        strip.text = element_text(size=10, face="bold"),
        legend.position = "none")
```



This figure highlights why we can't solely focus on getting statistically significant results. The alternative hypothesis might very well be true, but if we do not have enough people/observations, then we may lack the statistical power to detect it. Thus, anytime you are designing a study you need to consider statistical power, how high you want statistical power to be, and what changes you might be able to make to improve statistical power. E.g.,

1. Increasing the size of the effect (e.g., giving a drug at a high dose compared to a moderate dose).

2. Reducing the variability in the data (e.g., through inclusion/exclusion criteria or controlling for relevant covariates).
3. Increasing the number of observations (e.g., recruiting more participants and/or taking more observations per participant)

In the context of longitudinal studies, we are commonly most interested in *cross-level interactions*. For instance, did people receiving the experimental treatment change at the same rate as people receiving the control treatment? Or, do people with more severe baseline impairment show similar trajectories compared to people with less severe baseline impairment? These Group x Time interactions can take many forms and we cannot possibly cover them all here. However, we will try to provide some fundamental tools and show the logic of using simulation based approaches to statistical power. Users can then adapt this code to their specific case, changing things like the number of observations, the amount and pattern of missing data, the (non)linearity of their trajectories, and the magnitude of the effects they are predicting.

2. A Simulation Approach to Longitudinal Data

There are many tools that provide *analytical* solutions to statistical power in mixed models (e.g., https://glimmpse.samplesizeshop.org/; https://jakewestfall.shinyapps.io/pangea/). However, I think the simulation approach is very useful because it allows users to generate *empirical* solutions for very complex cases that are not easily accounted for by analytical packages (e.g., unequal spacing of time points, different types of missing data, differing random effects). The mathematics of mixed-effect regression are very complex and there is not always complete agreement on *how* effects should be estimated or if p-values should even be calculated (see: https://stat.ethz.ch/pipermail/r-help/2006-May/094765.html). Obviously, this makes a formal calculation of statistical power difficult, as you cannot confidently test $p<0.05$ if not everyone agrees on how p should be calculated.

Far from being impossible however, I would encourage you to think about statistical power in mixed-models as "squishy". For instance, flexibility in the structure of the random effects or choices in the methods of estimation can lead to different p-values. However, there are definitely some *incorrect* approaches (e.g., under-specified or mis-specified random effects) and being able to simulate your own data then allows you to test the effect of your different modelling choices on the outcome.

In this section, we will simulate a relatively simple longitudinal data structure with 4-5 observations for 10 individuals. The data are missing at random with the exception of every participant having the first observation. In the code chunk below, we set the number of individuals, the values for the fixed effects, and the values for the random effects.

```
N <- 10 # set number of individuals

# Fixed Effects
beta0 <- 50.0 # population intercept
beta1 <- 1.0  # population slope

# Random Effects and Errors
tau0  <- 10 # intercept SD,
tau1  <- 2.0 # slope SD,
tau01 <- 0.5 # correlation between slope and intercept,
sigma <- 5 # true error SD


# number of possible observations per person
max_obs <- 5
min_obs <- 4
```

With these starting values in place, we can simulate a random number of observations per person, constrained to between the minimum and maximum number of observations we set above. Finally, we bind the subject identifiers together with the time variable into a dataframe called *DATA*. The first six rows of the dataframe are shown.

```
# simulate MISSING AT RANDOM observations for each individual
set.seed(42)
p <- round(runif(n=N, min=min_obs, max=max_obs))

# simulate observations per person (everyone has 1st observation)
time <- unlist(sapply(p, function(x) c(1, sort(sample(x=2:max_obs, size = x-1, replace=FALSE)))))

# set up data frame
DATA <- data.frame(id=rep(1:N, times=p), time=time)

head(DATA)
```

```
##   id time
## 1  1    1
## 2  1    2
## 3  1    3
## 4  1    4
## 5  1    5
## 6  2    1
```

Next, we will create the actual data. To do this, we will take the scalar variance and covariance values that we set above, and then use those values to obtain correlated random effects.

```
mu  <- c(0,0) # random effects are assumed to be normally distributed with a mean of 0
S   <- matrix(c(1, tau01, tau01, 1), nrow=2) # correlation matrix for the randomw slope and intercept
taus <- c(tau0, tau1) # vector of random effect variances
S   <- diag(taus) %*% S %*% diag(taus) # rescaling our correlation matrix to be a covariance matrix
U   <- mvrnorm(N, mu=mu, Sigma=S) # Creating random deviates for the random slope and intercept, equal
```

Next, at the level of individual observations, we will simulate random errors. Note you could also simulate correlated residuals in this step, but this is an advanced topic. For the moment, we will assume that these errors are independent of each other with a mean = 0 and a standard deviation equal to *sigma* that we defined above.

```
# simulate (uncorrelated) residuals
# you could instead simulate correlated residuals, but that takes a bit more work
set.seed(42)
DATA$eij <- rnorm(n=nrow(DATA), mean=0, sd=sigma)

head(DATA)
```

```
##   id time        eij
## 1  1    1  6.8547922
## 2  1    2 -2.8234909
## 3  1    3  1.8156421
## 4  1    4  3.1643130
## 5  1    5  2.0213416
## 6  2    1 -0.5306226
```

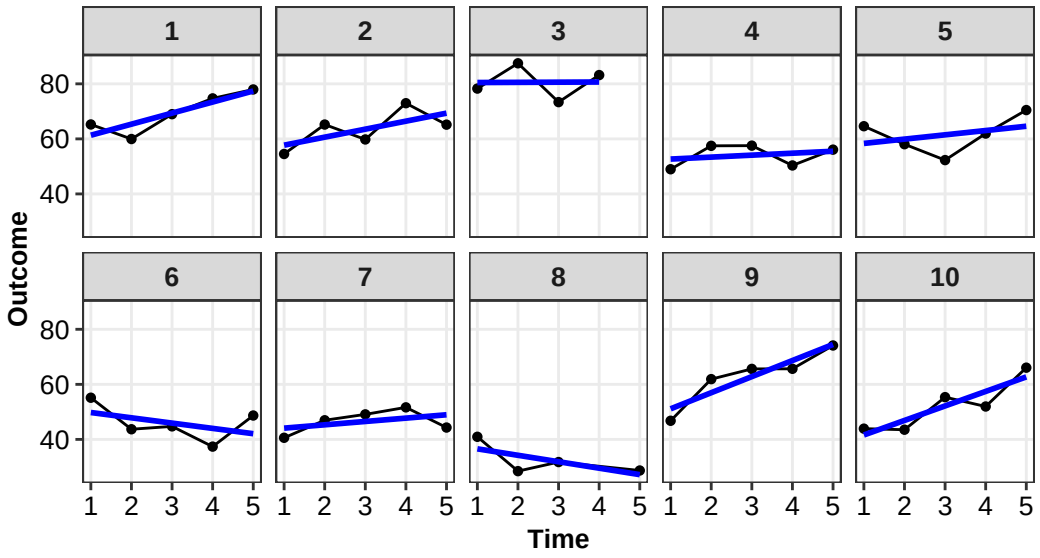
Finally, we can combine the fixed effects, the random deviates (for each participant), and the random errors (for each observation) to simulate longitudinal data with time nested within participants. We will print the first six rows of the data and then generate a plot showing all of the data for these 10 participants.

```
DATA$yij <- (beta0 + rep(U[,1], times=p)) +
  (beta1 + rep(U[,2], times=p)) * DATA$time + DATA$eij

head(DATA)
```

```
##   id time        eij      yij
## 1  1    1  6.8547922 65.24221
## 2  1    2 -2.8234909 59.95065
## 3  1    3  1.8156421 68.97651
## 4  1    4  3.1643130 74.71190
## 5  1    5  2.0213416 77.95566
## 6  2    1 -0.5306226 54.51284
```

```
cbPalette <- c("#999999", "#E69F00", "#56B4E9", "#009E73", "#CC79A7", "#F0E442",
               "#555555",  "#D55E00","#0072B2", "#03c478", "#661100", "#f3f319",
               "#222222","#FBD7A2", "#6699CC", "#99edcc", "#b804a2", "#F9E999")

# Lattice plot of the example data ----
ggplot(data=DATA, aes(x=time, y=yij)) +
  geom_point(shape=16, col="black")+
  geom_line(col="black")+
  stat_smooth(aes(group=id), col="blue", se=FALSE,
              method="lm")+
  scale_x_continuous(name = "Time") +
  scale_y_continuous(name = "Outcome") +
  facet_wrap(~id, ncol=5) +
  theme_bw()+
  theme(axis.text=element_text(size=10, color="black"),
        legend.text=element_text(size=10, color="black"),
        legend.title=element_text(size=10, face="bold"),
        axis.title=element_text(size=10, face="bold"),
        plot.title=element_text(size=12, face="bold", hjust=0.5),
        panel.grid.minor = element_blank(),
        strip.text = element_text(size=10, face="bold"),
        legend.position = "none")
```

```
## `geom_smooth()` using formula 'y ~ x'
```



You can tinker with the values for the fixed effects, random effects, and errors to see the effect this has on the data. For the moment though, the goal is just to show that we can simulate longitudinal data by reversing the process of data analysis. When analyzing data, we start with data and try to find the best fitting fixed effects and random effects, by minimizing the residuals (which are our best estimate of the true errors in the population). When simulating data, we start with the *true values* of the fixed effects in the population that we invented. We can then generate individual "participants" consistent with the *true* random effects (fixed effect + random deviate), and generate individual data points based on the underlying pattern for each participant (fixed effect + random deviate + error).

3. Simulating Two Different Populations and Resampling to Estimate Power

3.1 Setting the population parameters.

Next, we will build on the same example above to simulate two different populations. The first population ($POP1$) will have true intercept of 50 (sd = 10) and a true slope of 0 (sd=5).

```
# 1.0 Set the parameters for Population 1 ----
N <- 10000 # set number of individuals

beta0 <- 50.0 # true intercept
beta1 <- 0.0 # true slope

sigma <- 5  # true error SD
tau0  <- 10    # true intercept SD
tau1  <- 5  # true slope SD
tau01 <- 0.5  # slope-intercept correlation

# number of possible observations
max_obs <- 5
min_obs <- 4
```

```r
# simulate MISSING AT RANDOM observations for each individual
set.seed(1)
p <- round(runif(n=N, min=min_obs, max=max_obs))

# simulate observations per person (everyone has 1st observation)
time <- unlist(sapply(p, function(x) c(1, sort(sample(x=2:max_obs, size = x-1, replace=FALSE)))))

# set up data frame
POP1 <- data.frame(id=factor(rep(1:N, times=p)), time=time) %>%
  mutate(id = factor(paste("s1", id, sep="_")),
         group="A")

# simulate (correlated) random effects for intercepts and slopes
mu  <- c(0,0)
S   <- matrix(c(1, tau01, tau01, 1), nrow=2)
taus <- c(tau0, tau1)
S   <- diag(taus) %*% S %*% diag(taus)
U   <- mvrnorm(N, mu=mu, Sigma=S)

# simulate (uncorrelated) residuals
# you can simulate correlated residuals, but that takes a bit more work
set.seed(2)
POP1$eij <- rnorm(n=nrow(POP1), mean=0, sd=sigma)

POP1$yij <- (beta0 + rep(U[,1], times=p)) +
  (beta1 + rep(U[,2], times=p)) * POP1$time + POP1$eij
```

The second population ($POP2$) will have true intercept of 50 (sd = 10) and a true slope of 2.5 (sd=5).

```r
# 2.0 Set the parameters for Population 2 ----
N <- 10000 # set number of individuals

beta0 <- 50.0 # true intercept
beta1 <- 2.5 # true slope

sigma <- 5  # true error SD
tau0  <- 10    # true intercept SD
tau1  <- 5  # true slope SD
tau01 <- 0.5  # slope-intercept correlation

# number of possible observations
max_obs <- 5
min_obs <- 4

# simulate MISSING AT RANDOM observations for each individual
set.seed(1)
p <- round(runif(n=N, min=min_obs, max=max_obs))

# simulate observations per person (everyone has 1st observation)
time <- unlist(sapply(p, function(x) c(1, sort(sample(x=2:max_obs, size = x-1, replace=FALSE)))))

# set up data frame
POP2 <- data.frame(id=factor(rep(1:N, times=p)), time=time)%>%
```

```
  mutate(id = factor(paste("s2", id, sep="_")),
         group="B")

# simulate (correlated) random effects for intercepts and slopes
mu  <- c(0,0)
S   <- matrix(c(1, tau01, tau01, 1), nrow=2)
taus <- c(tau0, tau1)
S   <- diag(taus) %*% S %*% diag(taus)
U   <- mvrnorm(N, mu=mu, Sigma=S)

# simulate (uncorrelated) residuals
# you can simulate correlated residuals, but that takes a bit more work
set.seed(2)
POP2$eij <- rnorm(n=nrow(POP2), mean=0, sd=sigma)

POP2$yij <- (beta0 + rep(U[,1], times=p)) +
  (beta1 + rep(U[,2], times=p)) * POP1$time + POP1$eij
```

3.2 Repeatedly sample from the population (with replacement).

With these two populations in place, we will now repeatedly sample from each population (with replacement) to simulated experiments. In this case, I am going to take samples of two different sizes to ($n/group = 20$ and $n/group = 60$) to illustrate the effects that sample size has on statistical power. For each sample size, we will run $k = 1000$ simulations. All of these parameters can be controlled in the code below so that you can simulate samples of whatever sizes you wish (and more than two sample size if you wish) and obtain answers with varying degrees of precision. Setting $k = 1000$ should give us good precision, but it will take time. If you wanted to take a lot of sample size at once, it is a good idea to reduce k so that the code will run faster (e.g., $k = 500$ or $k = 100$ if a lot of different sample sizes are being tested). Then, once decide how many sample sizes are really needed (some may be obviously too small or two big) then you can increase k again to obtain precise results for the most interesting cases.

```
# set sample sizes
sample_sizes = c(20, 60)

# set number of iterations at each sample size
k = 1000
```

With the sample sizes and number of iterations in place for our simulations, next we will create a dataframe to store the results from our mixed model. In this case, we are fitting a relatively simply model with a Group x Time interaction and a random intercept of time: $yij\ 1 + time * group + (1 + time|id)$.

You could adjust the code to fit a more complicated model depending on the kind of data you generate in the previous steps. Our "ground truth" was a linear effect of time, so that is all we will model here. If you had used polynomial effects (or truly nonlinear effects) in the population, then you would fit a similar model here.

Because the outputs are being stored as *lists* the code is very flexible and can handle models with many different kinds of parameters.

```
# initialize null variables to populate:
sample_size = NULL
iteration = NULL
random_effects=NULL
```

```
fixed_effects=NULL
anova_results=NULL

count=0
set.seed(1)
for (size in sample_sizes){
  #print(size)

  for (i in c(1:k)) {
    count=count+1
    #print(count)

    # Sample from each population
    SAMP1 <- POP1[POP1$id %in% sample(x=unique(POP1$id), size=size, replace=FALSE),]
    SAMP2 <- POP2[POP2$id %in% sample(x=unique(POP2$id), size=size, replace=FALSE),]

    # Binding the two different samples together
    SAMPLE <- rbind(SAMP1, SAMP2)

    # Specify the model you want to fit
    mod <- lmer(yij~1+time*group+(1+time|id),
            data=SAMPLE,
            REML=TRUE)

    sample_size[[count]] = size
    iteration[[count]] = count
    random_effects[[count]] = data.frame(VarCorr(mod))
    fixed_effects[[count]] = data.frame(fixef(mod))
    anova_results[[count]] = data.frame(anova(mod))
  }
}
```

```
## boundary (singular) fit: see help('isSingular')
```

The sample size, iteration number, variance-covariance matrix for the random effects, parameter estimates and standard errors for the fixed effects, and the ANOVA table summarizing the results are all stored as lists from every iteration. To make these results a little easier to work with for plotting and exporting, we will gather the smaller *lists* into a master list, and the flatten out parts of that list into *data frames*:

```
# Bind all of our lists together into one big list
SIM_RESULTS <- list(sample_size=sample_size,
                    iteration=iteration,
                    random_effects=random_effects,
                    fixed_effects=fixed_effects,
                    anova_results=anova_results)

# Flatten out the iteration number and sample size into their own data frame
SAMP <- data.frame(iteration = as.character(unlist(SIM_RESULTS$iteration)),
                   sample_size = unlist(SIM_RESULTS$sample_size))


# Tidying the random effects output
RE_DATA <- bind_rows(SIM_RESULTS$random_effects, .id = "iteration") %>%
```

```
  pivot_wider(values_from = vcov:sdcor, names_from = grp:var2, names_sep="_") %>%
  left_join(SAMP, by="iteration") %>%
  relocate(iteration, sample_size)

# Tidying the fixed effects output
FE_DATA <- bind_rows(SIM_RESULTS$fixed_effects, .id = "iteration") %>%
  rownames_to_column(var="parameter") %>%
  mutate(parameter=str_split(parameter, "\\.{2,}", simplify = TRUE)[,1]) %>%
  pivot_wider(values_from = fixef.mod., names_from = parameter, names_sep="_") %>%
  left_join(SAMP, by="iteration") %>%
  relocate(iteration, sample_size)

# Tidying the ANOVA results
ANOVA_DATA <- bind_rows(SIM_RESULTS$anova_results, .id = "iteration")   %>%
  rownames_to_column(var="parameter") %>%
  mutate(parameter=str_split(parameter, "\\.{2,}", simplify = TRUE)[,1]) %>%
  left_join(SAMP, by="iteration") %>%
  relocate(iteration, sample_size)
```

Let's look at the first few rows of the cumulative random-effects data:

```
head(RE_DATA)
```

```
## # A tibble: 6 x 10
##   iteration sample_size `vcov_id_(Intercept)_~` vcov_id_time_NA `vcov_id_(Inte~`
##   <chr>           <dbl>                   <dbl>           <dbl>            <dbl>
## 1 1                  20                    95.6            21.4             20.4
## 2 2                  20                   124.             28.2             35.6
## 3 3                  20                   117.             19.4             22.2
## 4 4                  20                   118.             21.6             28.8
## 5 5                  20                   157.             21.7              8.04
## 6 6                  20                    61.7            18.4             19.5
## # ... with 5 more variables: vcov_Residual_NA_NA <dbl>,
## #   `sdcor_id_(Intercept)_NA` <dbl>, sdcor_id_time_NA <dbl>,
## #   `sdcor_id_(Intercept)_time` <dbl>, sdcor_Residual_NA_NA <dbl>
```

The first few rows of the cumulative fixed effects data:

```
head(FE_DATA)
```

```
## # A tibble: 6 x 6
##   iteration sample_size `(Intercept)`   time groupB `time:groupB`
##   <chr>           <dbl>         <dbl>  <dbl>  <dbl>         <dbl>
## 1 1                  20          50.8 -0.275 -1.12           3.39
## 2 2                  20          49.9 -1.79  -0.637          4.69
## 3 3                  20          50.2 -1.76  -0.242          3.12
## 4 4                  20          51.1 -1.32  -2.35           3.06
## 5 5                  20          50.6  0.195  1.02           3.70
## 6 6                  20          50.9  0.720 -4.51           1.58
```

And the first few rows of the cumulative ANOVA data:

```r
head(ANOVA_DATA)
```

```
##   iteration sample_size  parameter      Sum.Sq     Mean.Sq NumDF    DenDF
## 1         1          20       time  99.1342565  99.1342565     1 38.47797
## 2         1          20      group   2.9066550   2.9066550     1 38.37306
## 3         1          20 time:group 141.2381551 141.2381551     1 38.47797
## 4         2          20       time  10.9498976  10.9498976     1 38.29036
## 5         2          20      group   0.7198527   0.7198527     1 38.21514
## 6         2          20 time:group 194.2291131 194.2291131     1 38.29036
##      F.value     Pr..F.
## 1 3.24646582 0.07941682
## 2 0.09518764 0.75935168
## 3 4.62529159 0.03784676
## 4 0.39708094 0.53234342
## 5 0.02610433 0.87249697
## 6 7.04341557 0.01152311
```

3.3. Estimated Statistical Power

From these cumulative statistics, we can estimate statistical power for each of our effects. For instance, we can plot the p-values from the F-tests for the main-effect of $Group$, the main effect of $Time$ and the $GroupxTime$ interaction. You can see that with $n/group = 20$ we do not have good statistical power to detect our anticipated effect, but with $n/group = 60$, we are approaching 80% statistical power (which is a commonly used benchmark).

```r
# Plots showing the estimation of the true parameters ----
cbPalette <- c("#D55E00", "#56B4E9", "#009E73", "#000000",
               "#F0E442", "#0072B2", "#E69F00", "#CC79A7",
               "#999933", "#882255", "#661100", "#6699CC")


head(ANOVA_DATA)
```

```
##   iteration sample_size  parameter      Sum.Sq     Mean.Sq NumDF    DenDF
## 1         1          20       time  99.1342565  99.1342565     1 38.47797
## 2         1          20      group   2.9066550   2.9066550     1 38.37306
## 3         1          20 time:group 141.2381551 141.2381551     1 38.47797
## 4         2          20       time  10.9498976  10.9498976     1 38.29036
## 5         2          20      group   0.7198527   0.7198527     1 38.21514
## 6         2          20 time:group 194.2291131 194.2291131     1 38.29036
##      F.value     Pr..F.
## 1 3.24646582 0.07941682
## 2 0.09518764 0.75935168
## 3 4.62529159 0.03784676
## 4 0.39708094 0.53234342
## 5 0.02610433 0.87249697
## 6 7.04341557 0.01152311
```

```r
# distribution of p-values ----
A<-ggplot(data=ANOVA_DATA, aes(x=Pr..F.)) +
  geom_histogram(aes(fill=Pr..F.<0.05), col="black", binwidth = 0.02)+
```

```
  scale_x_continuous(name = NULL, limits=c(-0.5,1)) +
  scale_y_continuous(name = "Frequency") +
  ggtitle(label="Distribution of P-Values")+
  facet_wrap(~sample_size+parameter, ncol=3, scales = "free")+
  scale_fill_manual(values=cbPalette)+
  scale_colour_manual(values=cbPalette)+
  theme_bw()+
  theme(axis.text=element_text(size=10, color="black"),
        legend.text=element_text(size=10, color="black"),
        legend.title=element_text(size=10, face="bold"),
        axis.title=element_text(size=10, face="bold"),
        plot.title=element_text(size=12, face="bold", hjust=0.5),
        panel.grid.minor = element_blank(),
        strip.text = element_text(size=10, face="bold"),
        legend.position = "none")

# statistical power ----
ANOVA_DATA %>% group_by(sample_size, parameter) %>%
  summarize(sig=sum(Pr..F.<0.05)/k,
            ns=sum(Pr..F.>=0.05)/k) %>%
  pivot_longer(cols=sig:ns, names_to = "result", values_to = "freq")
```

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

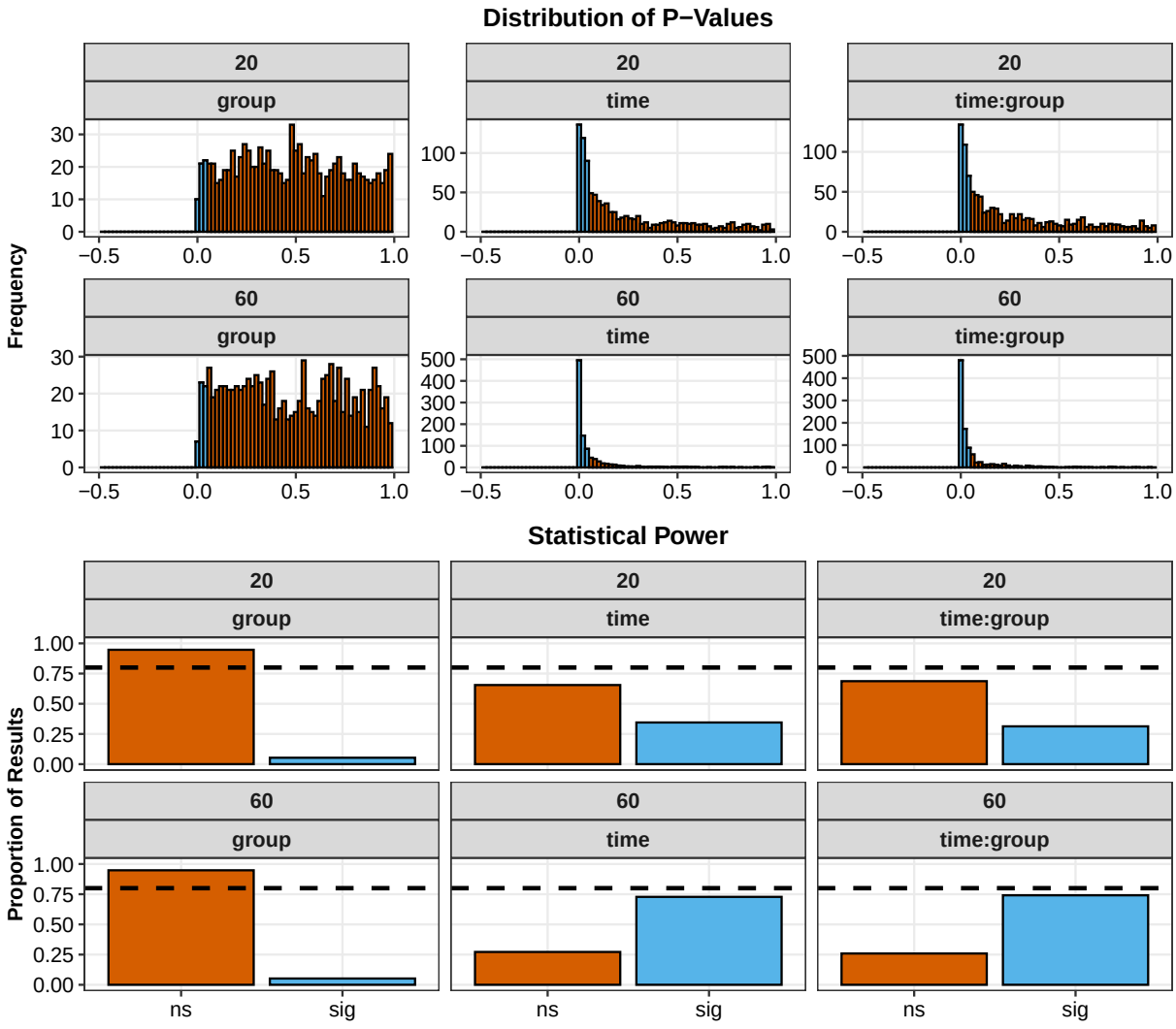
```
## # A tibble: 12 x 4
## # Groups:   sample_size [2]
##    sample_size parameter  result  freq
##          <dbl> <chr>      <chr>  <dbl>
##  1          20 group      sig    0.053
##  2          20 group      ns     0.947
##  3          20 time       sig    0.345
##  4          20 time       ns     0.655
##  5          20 time:group sig    0.313
##  6          20 time:group ns     0.687
##  7          60 group      sig    0.052
##  8          60 group      ns     0.948
##  9          60 time       sig    0.728
## 10          60 time       ns     0.272
## 11          60 time:group sig    0.741
## 12          60 time:group ns     0.259
```

```
B<-ggplot(data=ANOVA_DATA %>% group_by(sample_size, parameter) %>%
         summarize(sig=sum(Pr..F.<0.05)/k,
                   ns=sum(Pr..F.>=0.05)/k) %>%
         pivot_longer(cols=sig:ns, names_to = "result", values_to = "freq"),
       aes(x=result, y=freq)) +
  geom_bar(aes(fill=result), col="black", stat="identity")+
  geom_hline(yintercept=0.8, lty=2, lwd=1)+
  ggtitle(label="Statistical Power")+
  scale_x_discrete(name = NULL) +
  scale_y_continuous(name = "Proportion of Results", limits=c(0,1)) +
  facet_wrap(~sample_size+parameter, ncol=3)+
```

```
  scale_fill_manual(values=cbPalette)+
  scale_colour_manual(values=cbPalette)+
  theme_bw()+
  theme(axis.text=element_text(size=10, color="black"),
        legend.text=element_text(size=10, color="black"),
        legend.title=element_text(size=10, face="bold"),
        axis.title=element_text(size=10, face="bold"),
        plot.title=element_text(size=12, face="bold", hjust=0.5),
        panel.grid.minor = element_blank(),
        strip.text = element_text(size=10, face="bold"),
        legend.position = "none")
```

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

```
(A)/(B)
```

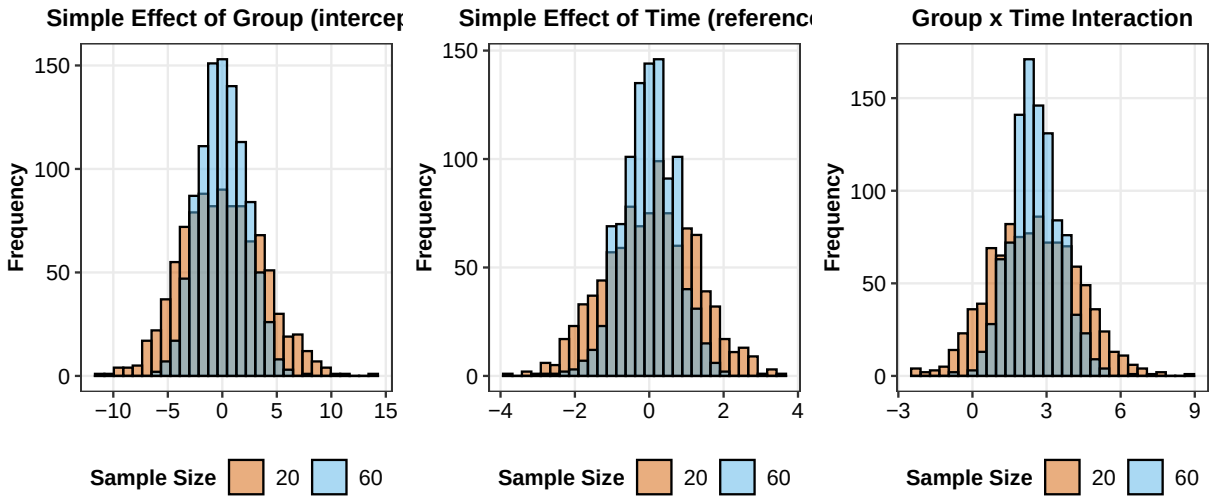


3.4. Sampling Distribution of Fixed-Effect Estimates

We can also use these cumulative statistics to see the distribution of the estimates for the fixed effects from all of our models. Note that each distribution is centered on the true value that we programmed in, but the width of each distribution will depend on the variance and the size of the sample we chose.

```
# Plots showing the estimation of the true parameters ----
cbPalette <- c("#D55E00", "#56B4E9", "#009E73", "#000000",
               "#F0E442", "#0072B2", "#E69F00", "#CC79A7",
               "#999933", "#882255", "#661100", "#6699CC")


# Group Effect at baseline ----
FE1 <- ggplot(data=FE_DATA, aes(x=groupB)) +
  geom_histogram(aes(fill=factor(sample_size)), col="black", bins=30,
                 position="identity", alpha=0.5)+
  #geom_vline(xintercept=beta0, lty=2, lwd=1, col="black")+
  scale_x_continuous(name = NULL) +
  scale_y_continuous(name = "Frequency") +
  ggtitle(label="Simple Effect of Group (intercept)")+
  scale_fill_manual(values=cbPalette)+
  scale_colour_manual(values=cbPalette)+
  labs(fill="Sample Size")+
  #facet_wrap(~sample_size, ncol=1)+
  theme_bw()+
  theme(axis.text=element_text(size=10, color="black"),
        legend.text=element_text(size=10, color="black"),
        legend.title=element_text(size=10, face="bold"),
        axis.title=element_text(size=10, face="bold"),
        plot.title=element_text(size=11, face="bold", hjust=0.5),
        panel.grid.minor = element_blank(),
        strip.text = element_text(size=10, face="bold"),
        legend.position = "bottom")

# Effect of Time in Reference Groups ----
FE2 <- ggplot(data=FE_DATA, aes(x=time)) +
  geom_histogram(aes(fill=factor(sample_size)), col="black", bins=30,
                 position="identity", alpha=0.5)+
  #geom_vline(xintercept=beta0, lty=2, lwd=1, col="black")+
  scale_x_continuous(name = NULL) +
  scale_y_continuous(name = "Frequency") +
  ggtitle(label="Simple Effect of Time (reference)")+
  scale_fill_manual(values=cbPalette)+
  scale_colour_manual(values=cbPalette)+
  labs(fill="Sample Size")+
  #facet_wrap(~sample_size, ncol=1)+
  theme_bw()+
  theme(axis.text=element_text(size=10, color="black"),
        legend.text=element_text(size=10, color="black"),
        legend.title=element_text(size=10, face="bold"),
        axis.title=element_text(size=10, face="bold"),
        plot.title=element_text(size=11, face="bold", hjust=0.5),
        panel.grid.minor = element_blank(),
        strip.text = element_text(size=10, face="bold"),
```

```
        legend.position = "bottom")

# Difference between slopes ----
FE3 <- ggplot(data=FE_DATA, aes(x=`time:groupB`)) +
  geom_histogram(aes(fill=factor(sample_size)), col="black", bins=30,
                 position="identity", alpha=0.5)+
  #geom_vline(xintercept=beta1, lty=2, lwd=1, col="black")+
  scale_x_continuous(name = NULL) +
  scale_y_continuous(name = "Frequency") +
  ggtitle(label="Group x Time Interaction")+
  scale_fill_manual(values=cbPalette)+
  scale_colour_manual(values=cbPalette)+
  labs(fill="Sample Size")+
  #facet_wrap(~sample_size, ncol=1)+
  theme_bw()+
  theme(axis.text=element_text(size=10, color="black"),
        legend.text=element_text(size=10, color="black"),
        legend.title=element_text(size=10, face="bold"),
        axis.title=element_text(size=10, face="bold"),
        plot.title=element_text(size=11, face="bold", hjust=0.5),
        panel.grid.minor = element_blank(),
        strip.text = element_text(size=10, face="bold"),
        legend.position = "bottom")

FE1|FE2|FE3
```



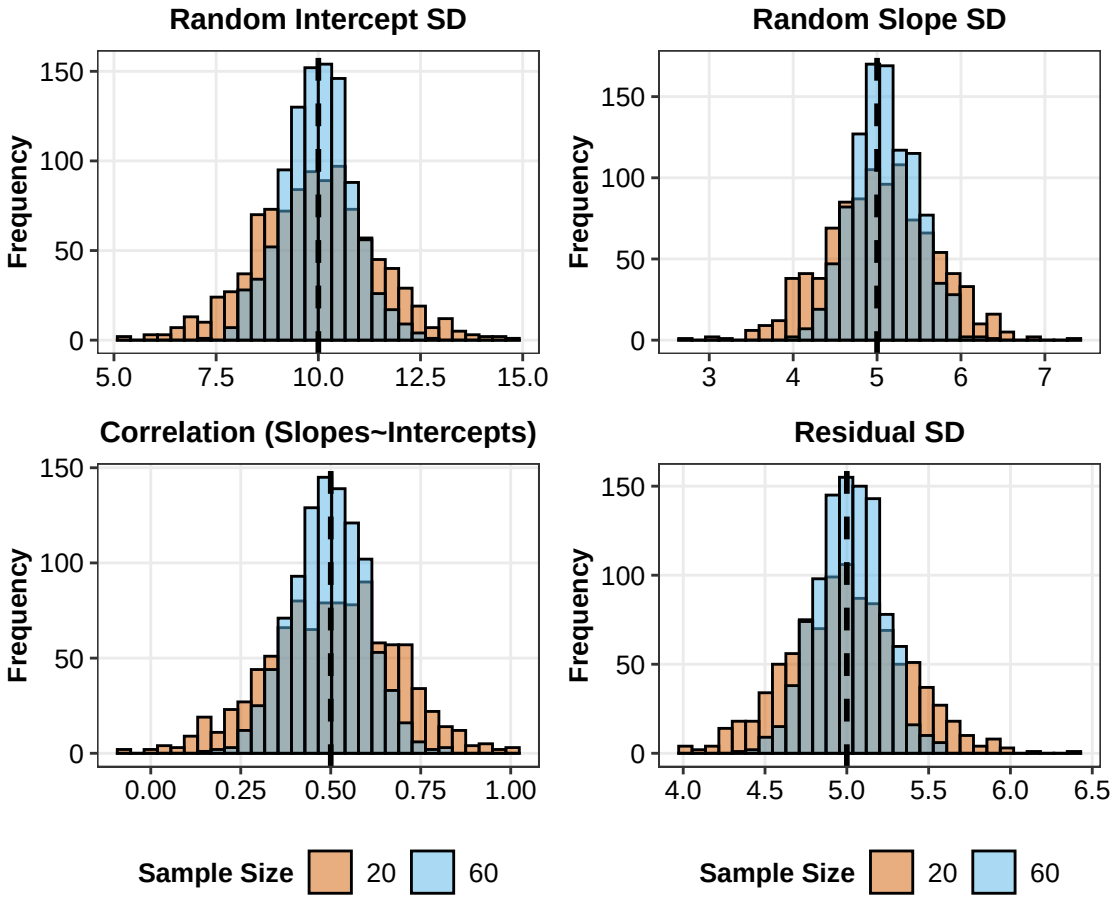
3.5. Sampling Distribution of Random-Effect Estimates

Similarly we can see our estimates of the variance components from the random effects data. As above, that each distribution is centered on the true value of the variance (or covariance) that we programmed in, but the width of each distribution will depend on the variance and the size of the sample we chose.

```
# Plots showing the estimation of the true parameters ----
cbPalette <- c("#D55E00", "#56B4E9", "#009E73", "#000000",
```

```
               "#F0E442", "#0072B2", "#E69F00", "#CC79A7",
               "#999933", "#882255", "#661100", "#6699CC")


# tau0  true standard deviation of the random intercept ----
RE1<-ggplot(data=RE_DATA, aes(x=`sdcor_id_(Intercept)_NA`)) +
  geom_histogram(aes(fill=factor(sample_size)), col="black", bins=30,
                 position="identity", alpha=0.5)+
  geom_vline(xintercept=tau0, lty=2, lwd=1, col="black")+
  scale_x_continuous(name = NULL) +
  scale_y_continuous(name = "Frequency") +
  ggtitle(label="Random Intercept SD")+
  scale_fill_manual(values=cbPalette)+
  scale_colour_manual(values=cbPalette)+
  labs(fill="Sample Size")+
  #facet_wrap(~sample_size, ncol=1)+
  theme_bw()+
  theme(axis.text=element_text(size=10, color="black"),
        legend.text=element_text(size=10, color="black"),
        legend.title=element_text(size=10, face="bold"),
        axis.title=element_text(size=10, face="bold"),
        plot.title=element_text(size=11, face="bold", hjust=0.5),
        panel.grid.minor = element_blank(),
        strip.text = element_text(size=10, face="bold"),
        legend.position = "none")

# tau1 true standard deviation of the random slope ----
RE2 <- ggplot(data=RE_DATA, aes(x=`sdcor_id_time_NA`)) +
  geom_histogram(aes(fill=factor(sample_size)), col="black", bins=30,
                 position="identity", alpha=0.5)+
  geom_vline(xintercept=tau1, lty=2, lwd=1, col="black")+
  scale_x_continuous(name = NULL) +
  scale_y_continuous(name = "Frequency") +
  ggtitle(label="Random Slope SD")+
  scale_fill_manual(values=cbPalette)+
  scale_colour_manual(values=cbPalette)+
  labs(fill="Sample Size")+
  #facet_wrap(~sample_size, ncol=1)+
  theme_bw()+
  theme(axis.text=element_text(size=10, color="black"),
        legend.text=element_text(size=10, color="black"),
        legend.title=element_text(size=10, face="bold"),
        axis.title=element_text(size=10, face="bold"),
        plot.title=element_text(size=11, face="bold", hjust=0.5),
        panel.grid.minor = element_blank(),
        strip.text = element_text(size=10, face="bold"),
        legend.position = "none")

# tau01 true correlation between random-slopes/intercepts ----
RE3<-ggplot(data=RE_DATA, aes(x=`sdcor_id_(Intercept)_time`)) +
  geom_histogram(aes(fill=factor(sample_size)), col="black", bins=30,
                 position="identity", alpha=0.5)+
  geom_vline(xintercept=tau01, lty=2, lwd=1, col="black")+
```

```
  scale_x_continuous(name = NULL) +
  scale_y_continuous(name = "Frequency") +
  ggtitle(label="Correlation (Slopes~Intercepts)")+
  scale_fill_manual(values=cbPalette)+
  scale_colour_manual(values=cbPalette)+
  labs(fill="Sample Size")+
  #facet_wrap(~sample_size, ncol=1)+
  theme_bw()+
  theme(axis.text=element_text(size=10, color="black"),
        legend.text=element_text(size=10, color="black"),
        legend.title=element_text(size=10, face="bold"),
        axis.title=element_text(size=10, face="bold"),
        plot.title=element_text(size=11, face="bold", hjust=0.5),
        panel.grid.minor = element_blank(),
        strip.text = element_text(size=10, face="bold"),
        legend.position = "bottom")


# sigma true standard deviation of the residuals ----
RE4<-ggplot(data=RE_DATA, aes(x=sdcor_Residual_NA_NA)) +
  geom_histogram(aes(fill=factor(sample_size)), col="black", bins=30,
                 position="identity", alpha=0.5)+
  geom_vline(xintercept=sigma, lty=2, lwd=1, col="black")+
  scale_x_continuous(name = NULL) +
  scale_y_continuous(name = "Frequency") +
  ggtitle(label="Residual SD")+
  scale_fill_manual(values=cbPalette)+
  scale_colour_manual(values=cbPalette)+
  labs(fill="Sample Size")+
  #facet_wrap(~sample_size, ncol=1)+
  theme_bw()+
  theme(axis.text=element_text(size=10, color="black"),
        legend.text=element_text(size=10, color="black"),
        legend.title=element_text(size=10, face="bold"),
        axis.title=element_text(size=10, face="bold"),
        plot.title=element_text(size=11, face="bold", hjust=0.5),
        panel.grid.minor = element_blank(),
        strip.text = element_text(size=10, face="bold"),
        legend.position = "bottom")

(RE1|RE2)/(RE3|RE4)
```


Random Intercept SD
Random Slope SD
Correlation (Slopes~Intercepts)
Residual SD
Frequency
150
100
50
0
5.0
7.5
10.0
12.5
15.0
3
4
5
6
7
0.00
0.25
0.50
0.75
1.00
4.0
4.5
5.0
5.5
6.0
6.5
Sample Size
20
60