```r
knitr::opts_chunk$set(echo = TRUE)
nsims <- 1000 #set number of simulations
library(mvtnorm)
library(afex)
```

```
## Loading required package: lme4

## Loading required package: Matrix

## ************
## Welcome to afex. For support visit: http://afex.singmann.science/

## - Functions for ANOVAs: aov_car(), aov_ez(), and aov_4()
## - Methods for calculating p-values with mixed(): 'KR', 'S', 'LRT', and 'PB'
## - 'afex_aov' and 'mixed' objects can be passed to emmeans() for follow-up tests
## - NEWS: library('emmeans') now needs to be called explicitly!
## - Get and set global package options with: afex_options()
## - Set orthogonal sum-to-zero contrasts globally: set_sum_contrasts()
## - For example analyses see: browseVignettes("afex")
## ************

##
## Attaching package: 'afex'

## The following object is masked from 'package:lme4':
##
##     lmer
```

```r
library(emmeans)
library(ggplot2)
library(gridExtra)
library(reshape2)
library(pwr)

# Install functions from GitHub by running the code below:
source("https://raw.githubusercontent.com/Lakens/ANOVA_power_simulation/master/ANOVA_design.R")
source("https://raw.githubusercontent.com/Lakens/ANOVA_power_simulation/master/ANOVA_power.R")
source("https://raw.githubusercontent.com/chrisaberson/pwr2ppl/master/R/anova1f_4.R")
source("https://raw.githubusercontent.com/chrisaberson/pwr2ppl/master/R/anova2x2.R")
source("https://raw.githubusercontent.com/Lakens/ANOVA_power_simulation/master/helper_functions/power_o
source("https://raw.githubusercontent.com/Lakens/ANOVA_power_simulation/master/helper_functions/power_t
source("https://raw.githubusercontent.com/Lakens/ANOVA_power_simulation/master/helper_functions/power_2
```

Analytic power functions

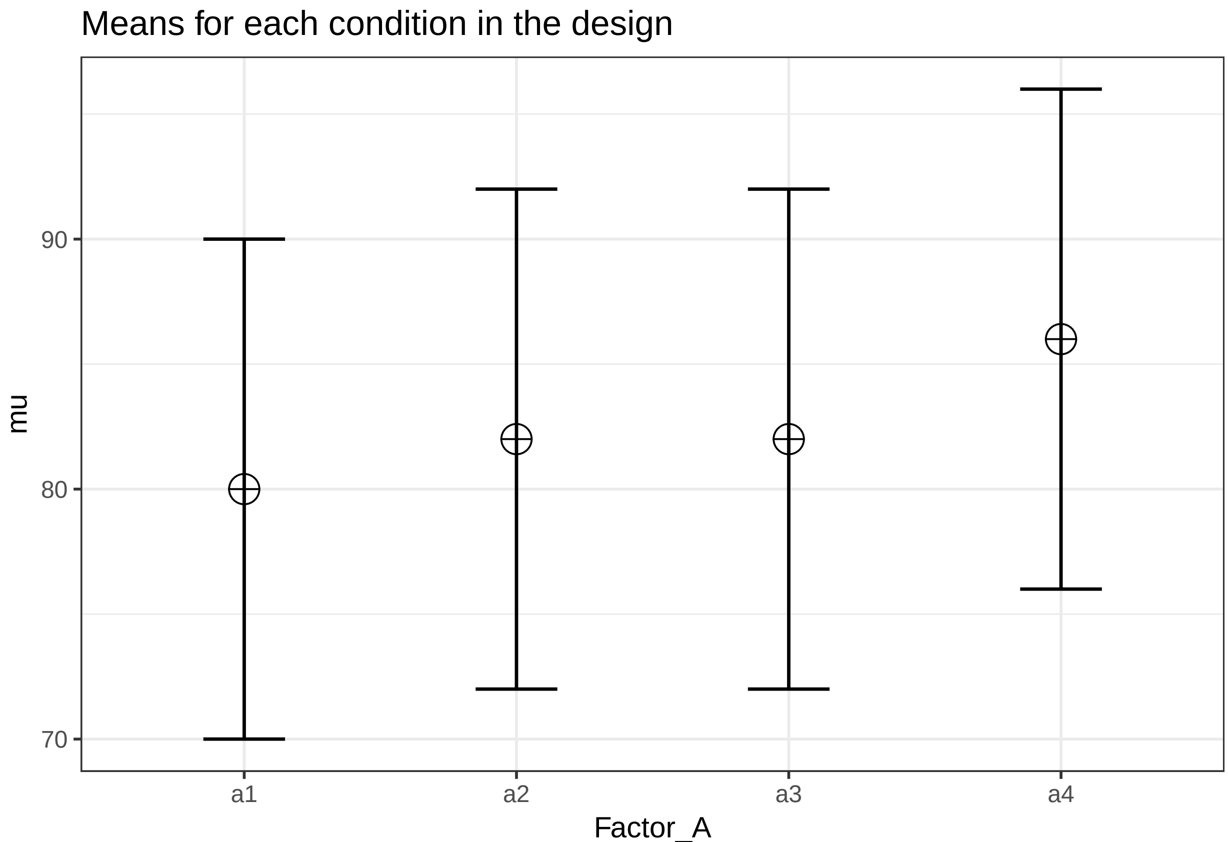
For some designs it is possible to calculate power analytically, using closed functions.

One-Way Between Subject ANOVA

```r
string <- "4b"
n <- 60
mu <- c(80, 82, 82, 86) #All means are equal - so there is no real difference.
# Enter means in the order that matches the labels below.
```

```r
sd <- 10
r <- 0
# (note that since we simulate a between design, the correlation between variables
# will be 0, regardless of what you enter here, but the value must be set).
p_adjust = "none"
# "none" means we do not correct for multiple comparisons
labelnames <- c("Factor_A", "a1", "a2", "a3", "a4") #
# the label names should be in the order of the means specified above.

design_result <- ANOVA_design(string = string,
                              n = n,
                              mu = mu,
                              sd = sd,
                              r = r,
                              p_adjust = p_adjust,
                              labelnames = labelnames)
```




```r
power_result <- ANOVA_power(design_result, alpha_level = 0.05, nsims = nsims)
```

```
## Power and Effect sizes for ANOVA tests
##                 power effect size
## anova_Factor_A  82.6      0.0536
##
## Power and Effect sizes for contrasts
##                            power effect size
## p_Factor_A_a1_Factor_A_a2  16.5      0.1862
```

```
## p_Factor_A_a1_Factor_A_a3  17.0      0.1987
## p_Factor_A_a1_Factor_A_a4  91.1      0.6032
## p_Factor_A_a2_Factor_A_a3   4.1      0.0133
## p_Factor_A_a2_Factor_A_a4  59.1      0.4175
## p_Factor_A_a3_Factor_A_a4  58.6      0.4035
```

We can also caculate power analytically with our own function.

```
power_oneway_between(design_result)$power #using default alpha level of .05
```

```
## [1] 0.8121291
```

This is a generalized function for One-Way ANOVA's for any number of groups. It is in part based on code provided with the excellent book by Aberson (2019) Applied Power Analysis for the Behavioral Sciences (but Aberson's code allows for different n per condition, and different sd per condition).

```
anova1f_4(m1=80, m2=82, m3=82, m4=86,
          s1=10, s2=10, s3=10, s4=10,
          n1=60, n2=60, n3=60, n4=60,
          alpha=.05)
```

```
## [1] "Power = 0.812"
```

We can also use the function in the pwr package. Note that we need to calculate f to use this function, which is based on the means and sd, as illustrated in the formulas above.

```
pwr.anova.test(n = 60,
               k = 4,
               f <- 0.2179449,
               sig.level = 0.05)
```

```
##
##      Balanced one-way analysis of variance power calculation
##
##               k = 4
##               n = 60
##               f = 0.2179449
##       sig.level = 0.05
##           power = 0.8121289
##
## NOTE: n is number in each group
```

Finally, G*Power provides the option to calculate f from the means, sd and n for the cells. It can then be used to calculate power.

Twoway Between Subject Interaction

```
string <- "2b*2b"
n <- 20
mu <- c(20, 20, 20, 25) #All means are equal - so there is no real difference.
# Enter means in the order that matches the labels below.
sd <- 5
r <- 0
# (note that since we simulate a between design, the correlation between variables
# will be 0, regardless of what you enter here, but the value must be set).
p_adjust = "none"
```

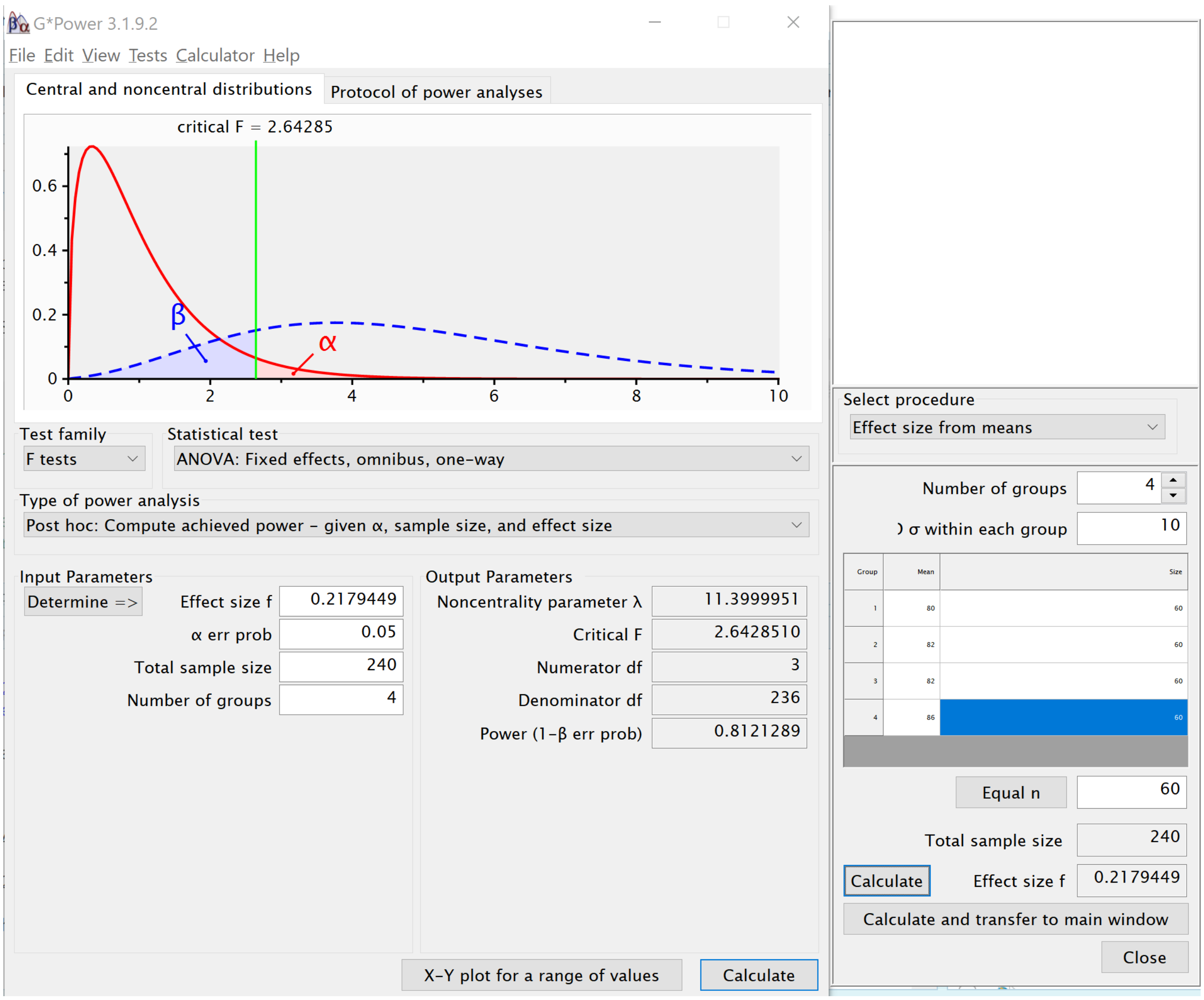
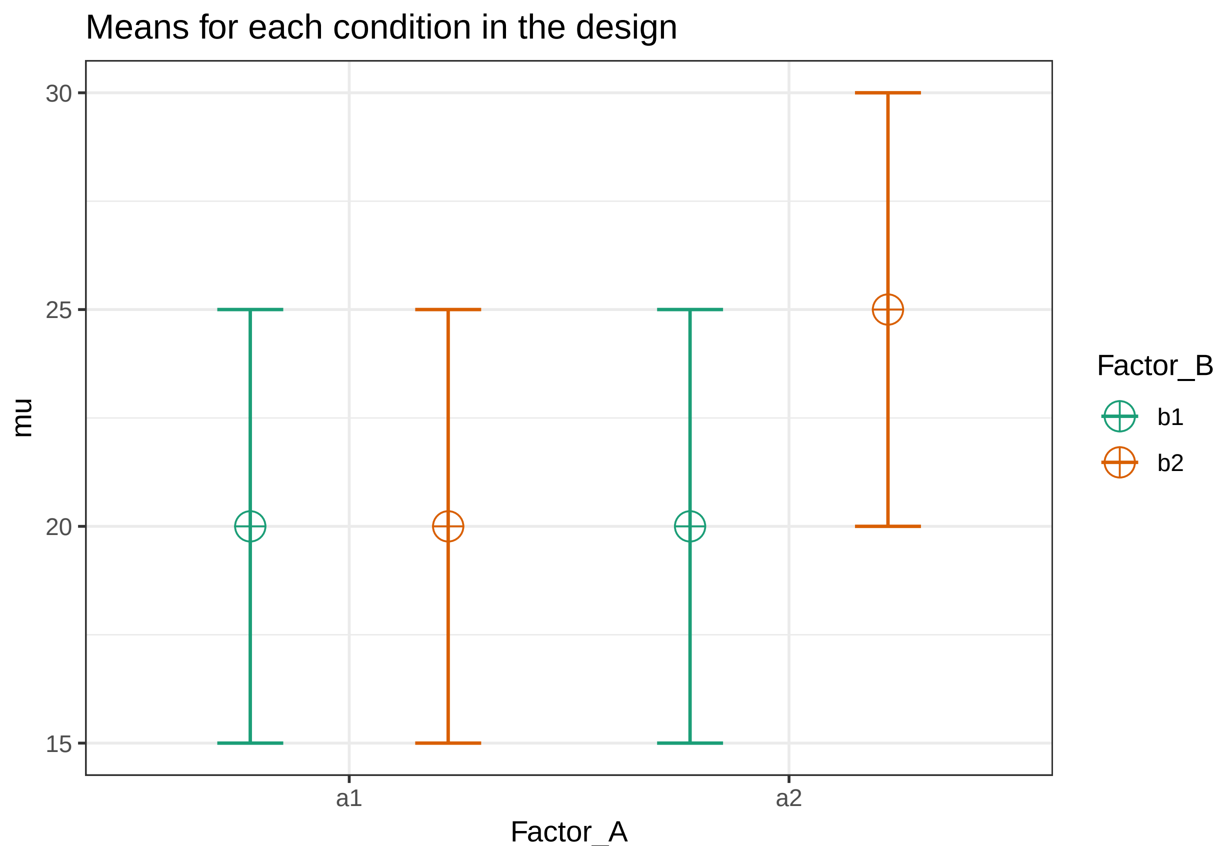

G*Power 3.1.9.2
File Edit View Tests Calculator Help
Central and noncentral distributions
Protocol of power analyses
critical F = 2.64285
Test family
F tests
Statistical test
ANOVA: Fixed effects, omnibus, one-way
Type of power analysis
Post hoc: Compute achieved power – given α, sample size, and effect size
Input Parameters
Determine =>
Effect size f
0.2179449
α err prob
0.05
Total sample size
240
Number of groups
4
Output Parameters
Noncentrality parameter λ
11.3999951
Critical F
2.6428510
Numerator df
3
Denominator df
236
Power (1–β err prob)
0.8121289
X-Y plot for a range of values
Calculate
Select procedure
Effect size from means
Number of groups
4
σ within each group
10
Equal n
60
Total sample size
240
Calculate
Effect size f
0.2179449
Calculate and transfer to main window
Close

Figure 1:

```r
# "none" means we do not correct for multiple comparisons
labelnames <- c("Factor_A", "a1", "a2", "Factor_B", "b1", "b2") #
# the label names should be in the order of the means specified above.

design_result <- ANOVA_design(string = string,
                   n = n,
                   mu = mu,
                   sd = sd,
                   r = r,
                   p_adjust = p_adjust,
                   labelnames = labelnames)
```




```r
power_result <- ANOVA_power(design_result, alpha_level = 0.05, nsims = nsims)
```

```
## Power and Effect sizes for ANOVA tests
##                         power effect size
## anova_Factor_A           60.3      0.0632
## anova_Factor_B           59.3      0.0619
## anova_Factor_A:Factor_B  61.0      0.0637
##
## Power and Effect sizes for contrasts
##                                                  power effect size
## p_Factor_A_a1_Factor_B_b1_Factor_A_a1_Factor_B_b2   4.1     -0.0059
## p_Factor_A_a1_Factor_B_b1_Factor_A_a2_Factor_B_b1   4.6      0.0036
## p_Factor_A_a1_Factor_B_b1_Factor_A_a2_Factor_B_b2  87.1      1.0343
## p_Factor_A_a1_Factor_B_b2_Factor_A_a2_Factor_B_b1   5.1      0.0095
```

```
## p_Factor_A_a1_Factor_B_b2_Factor_A_a2_Factor_B_b2  88.3      1.0340
## p_Factor_A_a2_Factor_B_b1_Factor_A_a2_Factor_B_b2  87.7      1.0278
```

```
power_res <- power_twoway_between(design_result) #using default alphe level of .05

power_res$power_A
```

```
## [1] 0.5978655
```

```
power_res$power_B
```

```
## [1] 0.5978655
```

```
power_res$power_AB
```

```
## [1] 0.5978655
```

We can use the function by Aberson, 2019, as well.

```
anova2x2(m1.1=20,
         m1.2=20,
         m2.1=20,
         m2.2=25,
         s1.1=5,
         s1.2=5,
         s2.1=5,
         s2.2=5,
         n1.1=20,
         n1.2=20,
         n2.1=20,
         n2.2=20,
         alpha=.05,
         all="OFF")
```

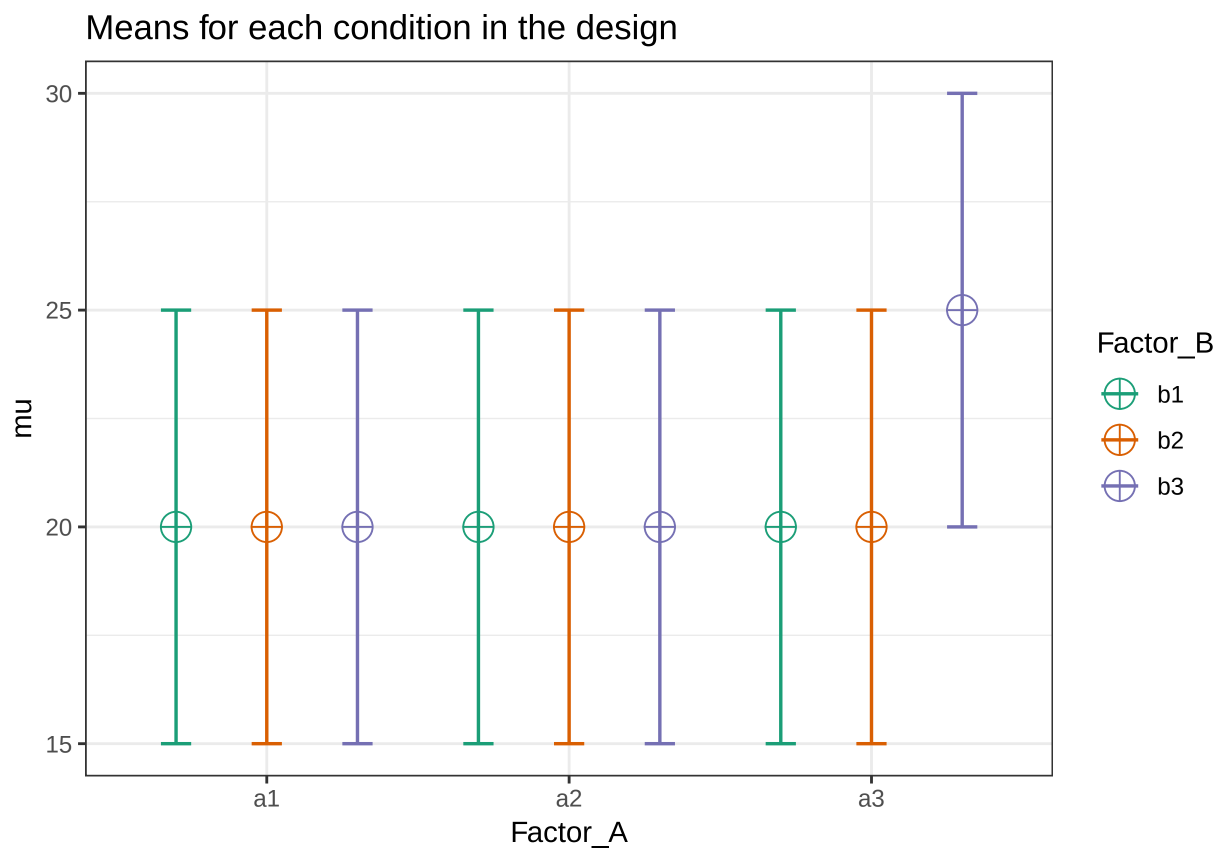
```
## [1] "Power for Main Effect Factor A =  0.598"
## [1] "Power for Main Effect Factor B =  0.598"
## [1] "Power for Interaction AxB =  0.598"
```

3x3 Between Subject ANOVA

```
string <- "3b*3b"
n <- 20
mu <- c(20, 20, 20, 20, 20, 20, 20, 20, 25) #All means are equal - so there is no real difference.
# Enter means in the order that matches the labels below.
sd <- 5
r <- 0
# (note that since we simulate a between design, the correlation between variables
# will be 0, regardless of what you enter here, but the value must be set).
p_adjust = "none"
# "none" means we do not correct for multiple comparisons
labelnames <- c("Factor_A", "a1", "a2", "a3", "Factor_B", "b1", "b2", "b3") #
# the label names should be in the order of the means specified above.

design_result <- ANOVA_design(string = string,
                   n = n,
```

```
                    mu = mu,
                    sd = sd,
                    r = r,
                    p_adjust = p_adjust,
                    labelnames = labelnames)
```




```
power_result <- ANOVA_power(design_result, alpha_level = 0.05, nsims = nsims)
```

```
## Power and Effect sizes for ANOVA tests
##                         power effect size
## anova_Factor_A            44.8      0.0317
## anova_Factor_B            45.1      0.0313
## anova_Factor_A:Factor_B  63.6      0.0649
##
## Power and Effect sizes for contrasts
##                                                     power effect size
## p_Factor_A_a1_Factor_B_b1_Factor_A_a1_Factor_B_b2   4.0     -0.0024
## p_Factor_A_a1_Factor_B_b1_Factor_A_a1_Factor_B_b3   5.4     -0.0035
## p_Factor_A_a1_Factor_B_b1_Factor_A_a2_Factor_B_b1   4.7     -0.0219
## p_Factor_A_a1_Factor_B_b1_Factor_A_a2_Factor_B_b2   4.4     -0.0071
## p_Factor_A_a1_Factor_B_b1_Factor_A_a2_Factor_B_b3   4.3     -0.0047
## p_Factor_A_a1_Factor_B_b1_Factor_A_a3_Factor_B_b1   4.0      0.0020
## p_Factor_A_a1_Factor_B_b1_Factor_A_a3_Factor_B_b2   3.7     -0.0120
## p_Factor_A_a1_Factor_B_b1_Factor_A_a3_Factor_B_b3  87.6      1.0169
## p_Factor_A_a1_Factor_B_b2_Factor_A_a1_Factor_B_b3   4.6     -0.0010
## p_Factor_A_a1_Factor_B_b2_Factor_A_a2_Factor_B_b1   4.8     -0.0172
```

```
## p_Factor_A_a1_Factor_B_b2_Factor_A_a2_Factor_B_b2   6.5     -0.0033
## p_Factor_A_a1_Factor_B_b2_Factor_A_a2_Factor_B_b3   5.0     -0.0016
## p_Factor_A_a1_Factor_B_b2_Factor_A_a3_Factor_B_b1   5.4      0.0052
## p_Factor_A_a1_Factor_B_b2_Factor_A_a3_Factor_B_b2   4.0     -0.0094
## p_Factor_A_a1_Factor_B_b2_Factor_A_a3_Factor_B_b3  86.2      1.0193
## p_Factor_A_a1_Factor_B_b3_Factor_A_a2_Factor_B_b1   3.9     -0.0162
## p_Factor_A_a1_Factor_B_b3_Factor_A_a2_Factor_B_b2   4.9     -0.0022
## p_Factor_A_a1_Factor_B_b3_Factor_A_a2_Factor_B_b3   5.5     -0.0013
## p_Factor_A_a1_Factor_B_b3_Factor_A_a3_Factor_B_b1   4.8      0.0051
## p_Factor_A_a1_Factor_B_b3_Factor_A_a3_Factor_B_b2   4.9     -0.0093
## p_Factor_A_a1_Factor_B_b3_Factor_A_a3_Factor_B_b3  87.6      1.0206
## p_Factor_A_a2_Factor_B_b1_Factor_A_a2_Factor_B_b2   5.5      0.0134
## p_Factor_A_a2_Factor_B_b1_Factor_A_a2_Factor_B_b3   4.7      0.0164
## p_Factor_A_a2_Factor_B_b1_Factor_A_a3_Factor_B_b1   5.0      0.0201
## p_Factor_A_a2_Factor_B_b1_Factor_A_a3_Factor_B_b2   5.9      0.0103
## p_Factor_A_a2_Factor_B_b1_Factor_A_a3_Factor_B_b3  89.5      1.0403
## p_Factor_A_a2_Factor_B_b2_Factor_A_a2_Factor_B_b3   5.5      0.0018
## p_Factor_A_a2_Factor_B_b2_Factor_A_a3_Factor_B_b1   6.9      0.0072
## p_Factor_A_a2_Factor_B_b2_Factor_A_a3_Factor_B_b2   5.3     -0.0065
## p_Factor_A_a2_Factor_B_b2_Factor_A_a3_Factor_B_b3  86.8      1.0257
## p_Factor_A_a2_Factor_B_b3_Factor_A_a3_Factor_B_b1   4.4      0.0074
## p_Factor_A_a2_Factor_B_b3_Factor_A_a3_Factor_B_b2   5.2     -0.0070
## p_Factor_A_a2_Factor_B_b3_Factor_A_a3_Factor_B_b3  88.0      1.0257
## p_Factor_A_a3_Factor_B_b1_Factor_A_a3_Factor_B_b2   4.7     -0.0146
## p_Factor_A_a3_Factor_B_b1_Factor_A_a3_Factor_B_b3  87.1      1.0129
## p_Factor_A_a3_Factor_B_b2_Factor_A_a3_Factor_B_b3  88.1      1.0325
```

```r
power_res <- power_twoway_between(design_result) #using default alphe level of .05

power_res$power_A
```

```
## [1] 0.4486306
```

```r
power_res$power_B
```

```
## [1] 0.4486306
```

```r
power_res$power_AB
```

```
## [1] 0.6434127
```

Two by two ANOVA, within design

Potvin & Schutz (2000) simulate a wide range of repeated measure designs. The give an example of a 3x3 design, with the following correlation matrix:

Variances were set to 1 (so all covariance matrices in their simulations were identical). In this specific example, the white fields are related to the correlation for the A main effect (these cells have the same level for B, but different levels of A). The grey cells are related to the main effect of B (the cells have the same level of A, but different levels of B). Finally, the black cells are related to the AxB interaction (they have different levels of A and B). The diagonal (all 1) relate to cells with the same levels of A and B.

Potvin & Schulz (2000) examine power for 2x2 within ANOVA designs and develop approximations of the error variance. For a design with 2 within factors (A and B) these are:

For the main effect of A: $\sigma_e^2 = \sigma^2(1 - \overline{\rho}_A) + \sigma^2(q-1)(\overline{\rho}_B - \overline{\rho}_{AB})$

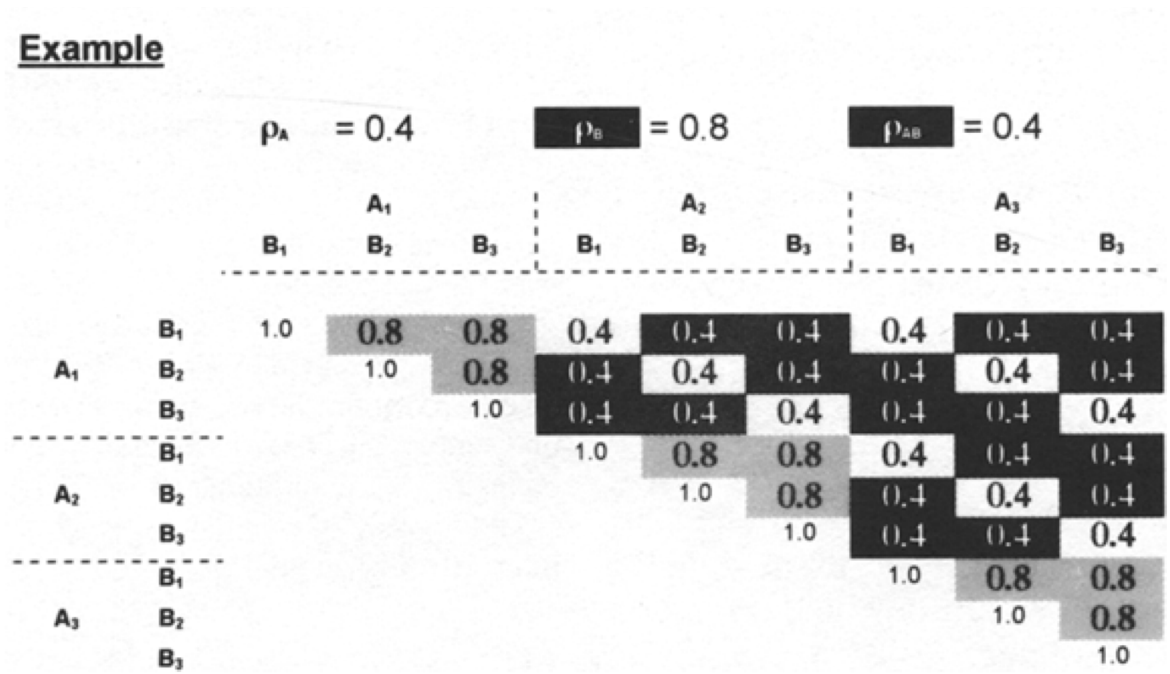


Figure 1. Representation of a correlation matrix for a 3 (A) × 3 (B) RM ANOVA: General form and numeric example. ρ_A and ρ_B represent the average correlation among the A and B (pooled) trials, respectively, and ρ_{AB} represents the average correlation among the AB coefficients having dissimilar levels.

Figure 2:

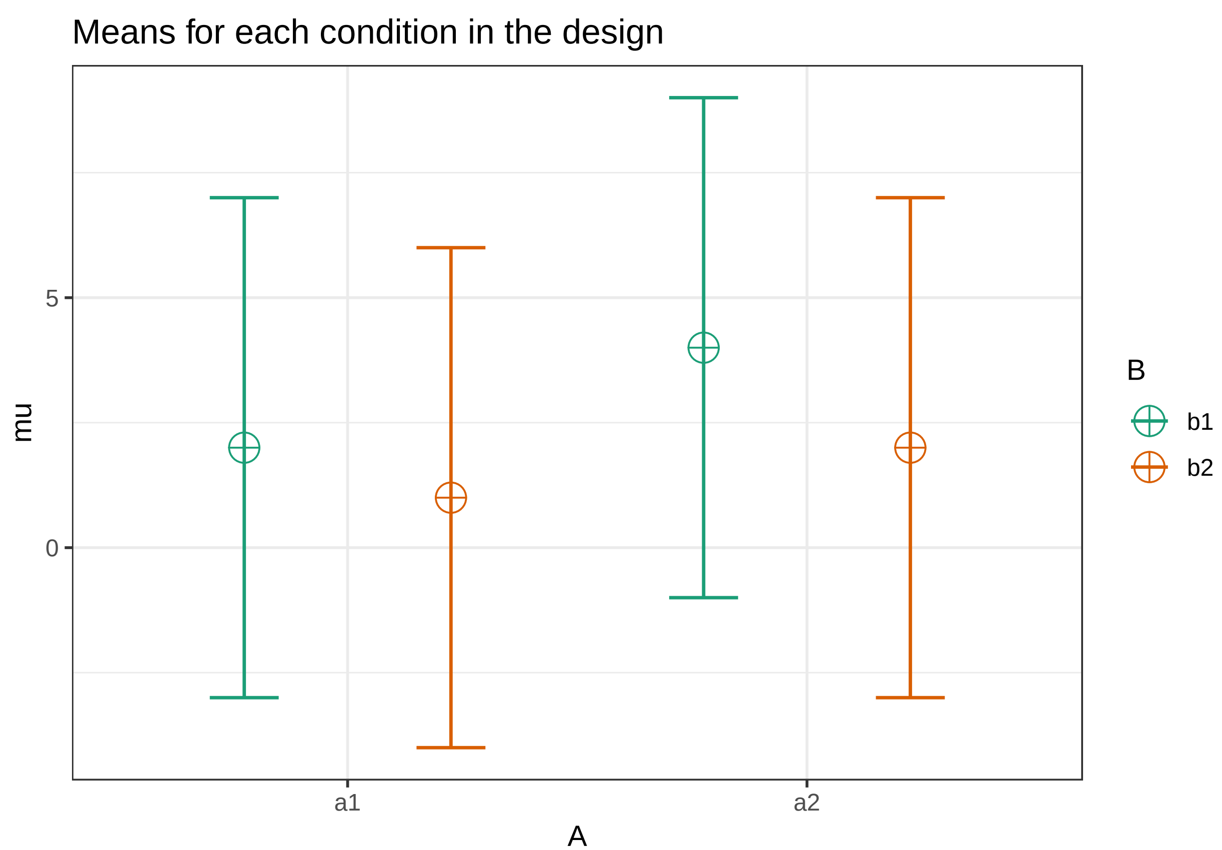
For the main effectof B: $\sigma_e^2 = \sigma^2(1-\overline{\rho}_B) + \sigma^2(p-1)(\overline{\rho}_A - \overline{\rho}_{AB})$

For the interaction between A and B: $\sigma_e^2 = \sigma^2(1-\rho_{\max}) - \sigma^2(\overline{\rho}_{\min} - \overline{\rho}_{AB})$

We first simulate a within subjects 2x2 ANOVA design.

```
mu = c(2,1,4,2)
n <- 20
sd <- 5
r <- c(
  0.8, 0.5, 0.4,
       0.4, 0.5,
            0.8
  )

string = "2w*2w"
alpha_level <- 0.05
p_adjust = "none"
labelnames = c("A", "a1", "a2", "B", "b1", "b2")
design_result <- ANOVA_design(string = string,
                              n = n,
                              mu = mu,
                              sd = sd,
                              r = r,
                              p_adjust = p_adjust,
                              labelnames = labelnames)
```



```
simulation_result <- ANOVA_power(design_result, alpha = 0.05, nsims = nsims)
```

```
## Power and Effect sizes for ANOVA tests
##             power effect size
## anova_A      26.2      0.0970
## anova_B      66.8      0.2542
## anova_A:B    25.7      0.0969
##
## Power and Effect sizes for contrasts
##                         power effect size
## p_A_a1_B_b1_A_a1_B_b2   28.3     -0.3456
## p_A_a1_B_b1_A_a2_B_b1   38.3      0.4099
## p_A_a1_B_b1_A_a2_B_b2    5.5     -0.0069
## p_A_a1_B_b2_A_a2_B_b1   65.9      0.5769
## p_A_a1_B_b2_A_a2_B_b2   12.5      0.2071
## p_A_a2_B_b1_A_a2_B_b2   79.1     -0.6736
```

Result simulation after 100000 simulations

simulation__result <- ANOVA__power(design__result, alpha = 0.05, nsims = 100000) Power and Effect sizes for ANOVA tests power effect size anova__A 26.849 0.0984 anova__B 64.091 0.2452 anova__A:B 26.875 0.0983

Power and Effect sizes for contrasts power effect size p__A__a1__B__b1__A__a1__B__b2 27.052 -0.3298 p__A__a1__B__b1__A__a2__B__b1 39.637 0.4162 p__A__a1__B__b1__A__a2__B__b2 4.983 -0.0005 p__A__a1__B__b2__A__a2__B__b1 64.252 0.5699 p__A__a1__B__b2__A__a2__B__b2 13.479 0.2077 p__A__a2__B__b1__A__a2__B__b2 76.622 -0.6597

We can try to use the formula in Potvin & Schutz (2000).

```
mean_mat <- t(matrix(mu,
                     nrow = 2,
                     ncol = 2)) #Create a mean matrix
rownames(mean_mat) <- c("a1", "a2")
colnames(mean_mat) <- c("b1", "b2")
mean_mat
```

```
##    b1 b2
## a1  2  1
## a2  4  2
```

```
a1 <- mean_mat[1,1] - (mean(mean_mat) + (mean(mean_mat[1,]) - mean(mean_mat)) + (mean(mean_mat[,1]) - m
a2 <- mean_mat[1,2] - (mean(mean_mat) + (mean(mean_mat[1,]) - mean(mean_mat)) + (mean(mean_mat[,2]) - m
b1 <- mean_mat[2,1] - (mean(mean_mat) + (mean(mean_mat[2,]) - mean(mean_mat)) + (mean(mean_mat[,1]) - m
b2 <- mean_mat[2,2] - (mean(mean_mat) + (mean(mean_mat[2,]) - mean(mean_mat)) + (mean(mean_mat[,2]) - m
c(a1, a2, b1, b2)
```

```
## [1] -0.25  0.25  0.25 -0.25
```

```
k <- 1 #one group (because all factors are within)
rho_A <- 0.5 #mean r for factor A
rho_B <- 0.8 #mean r for factor B
rho_AB <- 0.4 #mean r for factor AB
alpha <- 0.05
sigma <- sd

m_A <- 2 #levels factor A
variance_e_A <- sigma^2 * (1 - rho_A) + sigma^2 * (m_A - 1) * (rho_B - rho_AB) #Variance A
variance_e_A
```

```
## [1] 22.5
```

```
m_B <- 2 #levels factor B
variance_e_B <- sigma^2 * (1 - rho_B) + sigma^2 * (m_B - 1) * (rho_A - rho_AB) #Variance B
variance_e_B
```

```
## [1] 7.5
```

```
variance_e_AB <- sigma^2 * (1 - max(rho_A, rho_B)) - sigma^2 * (min(rho_A, rho_B) - rho_AB) #Variance A
variance_e_AB
```

```
## [1] 2.5
```

```
# Potving & Schutz, 2000, formula 2, p. 348
# For main effect A
lambda_A <- n * m_A * sum((rowMeans(mean_mat)-mean(rowMeans(mean_mat)))^2)/variance_e_A
lambda_A
```

```
## [1] 2
```

```
df1 <- (m_A - 1) #calculate degrees of freedom 1 - ignoring the * e sphericity correction
df2 <- (n - k) * (m_A - 1) #calculate degrees of freedom 2
F_critical <- qf(alpha, # critical F-vaue
                 df1,
                 df2,
                 lower.tail=FALSE)

pow_A <- pf(qf(alpha, #power
             df1,
             df2,
             lower.tail = FALSE),
          df1,
          df2,
          lambda_A,
          lower.tail = FALSE)

lambda_B <- n * m_B * sum((colMeans(mean_mat)-mean(colMeans(mean_mat)))^2)/variance_e_B
lambda_B
```

```
## [1] 6
```

```
df1 <- (m_B - 1) #calculate degrees of freedom 1
df2 <- (n - k) * (m_B - 1) #calculate degrees of freedom 2
F_critical <- qf(alpha, # critical F-vaue
                 df1,
                 df2,
                 lower.tail=FALSE)

pow_B <- pf(qf(alpha, #power
             df1,
             df2,
             lower.tail = FALSE),
          df1,
          df2,
          lambda_B,
          lower.tail = FALSE)
```

```
lambda_AB <- n * sqrt(sum(c(a1, a2, b1, b2)^2)/length(mu))/ variance_e_AB
lambda_AB
```

```
## [1] 2
```

```
df1 <- (m_A - 1)*(m_B - 1)  #calculate degrees of freedom 1
df2 <- (n - k) * (m_A - 1) * (m_B - 1) #calculate degrees of freedom 2
F_critical <- qf(alpha, # critical F-vaue
                 df1,
                 df2,
                 lower.tail=FALSE)

pow_AB <- pf(qf(alpha, #power
             df1,
             df2,
             lower.tail = FALSE),
          df1,
          df2,
          lambda_AB,
          lower.tail = FALSE)
pow_A
```

```
## [1] 0.2691752
```

```
pow_B
```

```
## [1] 0.6422587
```

```
pow_AB
```

```
## [1] 0.2691752
```

We see the 26.9 and 64.2, and 26.9 correspond to the results of the simulation quite closely.

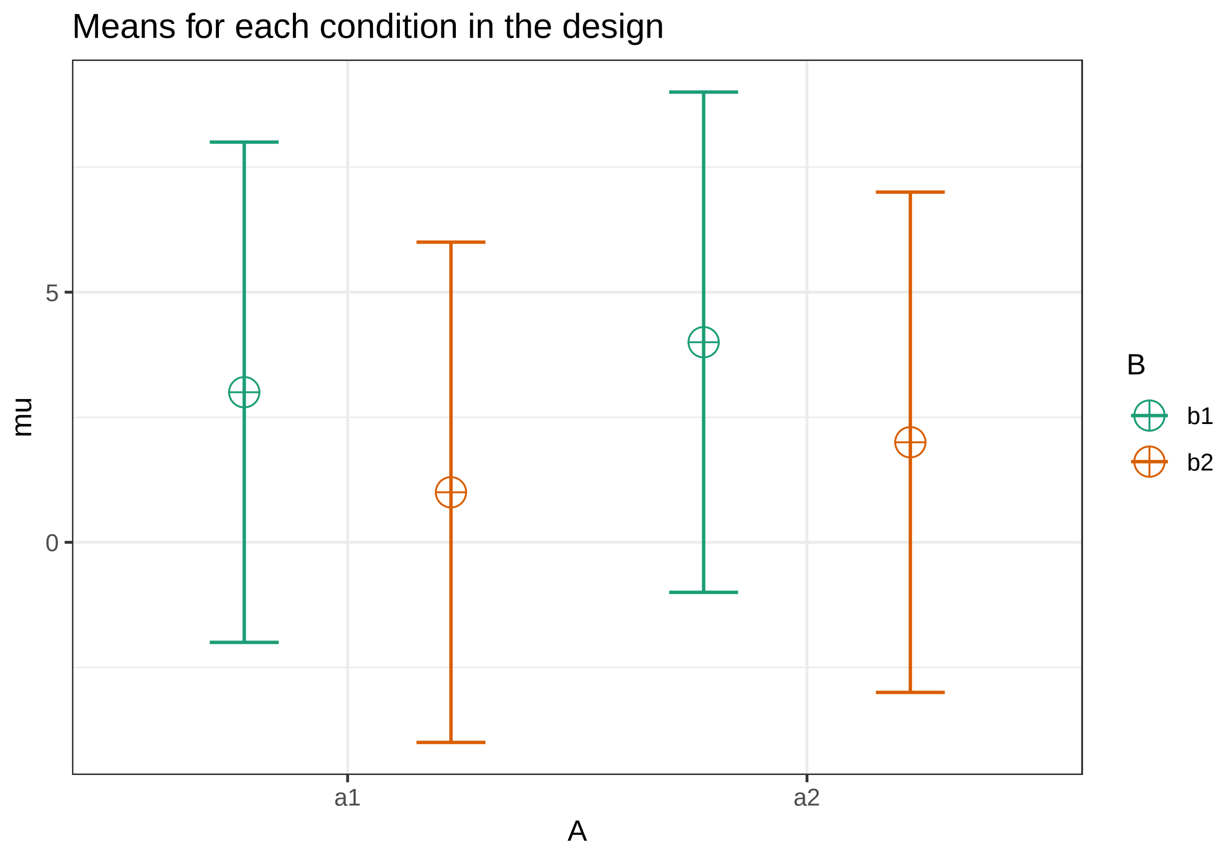
Variation 2x2 within design

We first simulate a within subjects 2x2 ANOVA design.

```
mu = c(3,1,4,2)
n <- 20
sd <- 5
r <- c(
  0.8, 0.5, 0.5,
       0.5, 0.5,
            0.8
  )

string = "2w*2w"
alpha_level <- 0.05
p_adjust = "none"
labelnames = c("A", "a1", "a2", "B", "b1", "b2")
design_result <- ANOVA_design(string = string,
                              n = n,
                              mu = mu,
                              sd = sd,
```

```
                          r = r,
                          p_adjust = p_adjust,
                          labelnames = labelnames)
```




```
simulation_result <- ANOVA_power(design_result, alpha = 0.05, nsims = nsims)
```

```
## Power and Effect sizes for ANOVA tests
##           power effect size
## anova_A    16.2      0.0566
## anova_B    96.4      0.4653
## anova_A:B   5.7      0.0253
##
## Power and Effect sizes for contrasts
##                       power effect size
## p_A_a1_B_b1_A_a1_B_b2  74.7     -0.6557
## p_A_a1_B_b1_A_a2_B_b1  14.9      0.2088
## p_A_a1_B_b1_A_a2_B_b2  13.7     -0.2090
## p_A_a1_B_b2_A_a2_B_b1  71.9      0.6218
## p_A_a1_B_b2_A_a2_B_b2  13.6      0.2052
## p_A_a2_B_b1_A_a2_B_b2  77.0     -0.6621
```

Now the analytic solution.

```
mean_mat <- t(matrix(mu,
                     nrow = 2,
                     ncol = 2)) #Create a mean matrix
rownames(mean_mat) <- c("a1", "a2")
```

```
colnames(mean_mat) <- c("b1", "b2")
mean_mat
```

```
##    b1 b2
## a1  3  1
## a2  4  2
```

```
a1 <- mean_mat[1,1] - (mean(mean_mat) + (mean(mean_mat[1,]) - mean(mean_mat)) + (mean(mean_mat[,1]) - me
a2 <- mean_mat[1,2] - (mean(mean_mat) + (mean(mean_mat[1,]) - mean(mean_mat)) + (mean(mean_mat[,2]) - me
b1 <- mean_mat[2,1] - (mean(mean_mat) + (mean(mean_mat[2,]) - mean(mean_mat)) + (mean(mean_mat[,1]) - me
b2 <- mean_mat[2,2] - (mean(mean_mat) + (mean(mean_mat[2,]) - mean(mean_mat)) + (mean(mean_mat[,2]) - me
c(a1, a2, b1, b2)
```

```
## [1] 0 0 0 0
```

```
k <- 1 #one group (because all factors are within)
rho_A <- 0.5 #mean r for factor A
rho_B <- 0.8 #mean r for factor B
rho_AB <- 0.4 #mean r for factor AB
alpha <- 0.05
sigma <- sd

m_A <- 2 #levels factor A
variance_e_A <- sigma^2 * (1 - rho_A) + sigma^2 * (m_A - 1) * (rho_B - rho_AB) #Variance A
variance_e_A
```

```
## [1] 22.5
```

```
m_B <- 2 #levels factor B
variance_e_B <- sigma^2 * (1 - rho_B) + sigma^2 * (m_B - 1) * (rho_A - rho_AB) #Variance B
variance_e_B
```

```
## [1] 7.5
```

```
variance_e_AB <- sigma^2 * (1 - max(rho_A, rho_B)) - sigma^2 * (min(rho_A, rho_B) - rho_AB) #Variance A
variance_e_AB
```

```
## [1] 2.5
```

```
# Potving & Schutz, 2000, formula 2, p. 348
# For main effect A
lambda_A <- n * m_A * sum((rowMeans(mean_mat)-mean(rowMeans(mean_mat)))^2)/variance_e_A
lambda_A
```

```
## [1] 0.8888889
```

```
df1 <- (m_A - 1) #calculate degrees of freedom 1 - ignoring the * e sphericity correction
df2 <- (n - k) * (m_A - 1) #calculate degrees of freedom 2
F_critical <- qf(alpha, # critical F-vaue
                 df1,
                 df2,
                 lower.tail=FALSE)

pow_A <- pf(qf(alpha, #power
             df1,
             df2,
             lower.tail = FALSE),
          df1,
```

```r
          df2,
          lambda_A,
          lower.tail = FALSE)

lambda_B <- n * m_B * sum((colMeans(mean_mat)-mean(colMeans(mean_mat)))^2)/variance_e_B
lambda_B
```

```
## [1] 10.66667
```

```r
df1 <- (m_B - 1) #calculate degrees of freedom 1
df2 <- (n - k) * (m_B - 1) #calculate degrees of freedom 2
F_critical <- qf(alpha, # critical F-vaue
                 df1,
                 df2,
                 lower.tail=FALSE)

pow_B <- pf(qf(alpha, #power
             df1,
             df2,
             lower.tail = FALSE),
          df1,
          df2,
          lambda_B,
          lower.tail = FALSE)

lambda_AB <- n * sqrt(sum(c(a1, a2, b1, b2)^2)/length(mu))/ variance_e_AB
lambda_AB
```

```
## [1] 0
```

```r
df1 <- (m_A - 1)*(m_B - 1)  #calculate degrees of freedom 1
df2 <- (n - k) * (m_A - 1) * (m_B - 1) #calculate degrees of freedom 2
F_critical <- qf(alpha, # critical F-vaue
                 df1,
                 df2,
                 lower.tail=FALSE)

pow_AB <- pf(qf(alpha, #power
             df1,
             df2,
             lower.tail = FALSE),
          df1,
          df2,
          lambda_AB,
          lower.tail = FALSE)
pow_A
```

```
## [1] 0.1457747
```

```r
pow_B
```

```
## [1] 0.8722533
```

```r
pow_AB
```

```
## [1] 0.05
```