```r
knitr::opts_chunk$set(echo = TRUE)
nsims <- 100000 #set number of simulations
library(mvtnorm)
library(afex)
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")
```

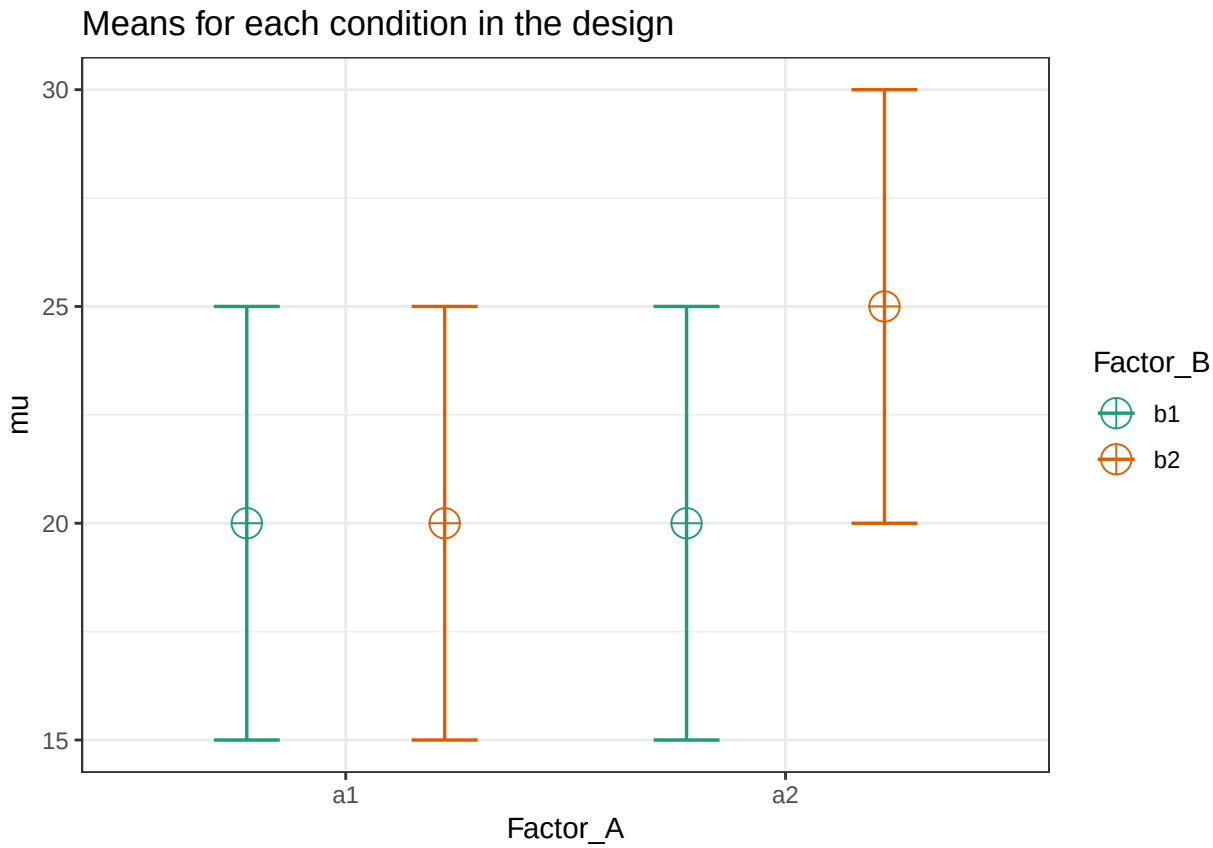
Analytic power functions

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

2x2 Between Subject Interaction

```r
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"
# "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)
```




```
alpha_level <- 0.05 #We set the alpha level at 0.05.

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

```
## Power and Effect sizes for ANOVA tests
##                          power effect size
## anova_Factor_A          59.738      0.0622
## anova_Factor_B          59.851      0.0623
## anova_Factor_A:Factor_B 59.837      0.0622
##
## Power and Effect sizes for contrasts
##                                                      power effect size
## p_Factor_A_a1_Factor_B_b1_Factor_A_a1_Factor_B_b2  4.956     -0.0002
## p_Factor_A_a1_Factor_B_b1_Factor_A_a2_Factor_B_b1  5.039     -0.0008
## p_Factor_A_a1_Factor_B_b1_Factor_A_a2_Factor_B_b2 86.889      1.0201
## p_Factor_A_a1_Factor_B_b2_Factor_A_a2_Factor_B_b1  4.898     -0.0006
## p_Factor_A_a1_Factor_B_b2_Factor_A_a2_Factor_B_b2 86.925      1.0200
## p_Factor_A_a2_Factor_B_b1_Factor_A_a2_Factor_B_b2 86.724      1.0208
```

Mathematically the interaction effect is computed as the difference between a cell mean and the grand mean, the marginal mean in row i and the grand mean, and the marginal mean in column j and grand mean. For example, for the very hungry-banana condition this is 25 (the value in the cell) - (21.25 [the grand mean] + 1.25 [the marginal mean in row 2, 22.5, minus the grand mean of 21.25] + 1.25 [the marginal mean in column 2, 22.5, minus the grand mean of 21.25]). 25 - (21.25 + (22.5-21.25) + (22.5-21.25)) = 1.25.

We can repeat this for every cell, and get for no hunger-apple: 20 - (21.25 + (20-21.25) + (20-21.25)) = 1.25, for very hungry apple: 20 - (21.25 + (22.5-21.25) + (20-21.25)) = 1.25, and no hunger-banana: 20 - (21.25 + (20-21.25) + (22.5-21.25)) = 1.25. These values are used to calculate the sum of squares.

```
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 20 20
## a2 20 25
```

```
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]  1.25 -1.25 -1.25  1.25
```

```
SS_ab <- n * sum(c(a1, a2, b1, b2)^2)
```

The sum of squares is dependent on the sample size. The larger the sample size, the larger the sum of squares, and therefore (all else equal) the larger the F-statistic, and the smaller the p-value. We see from the simulations that all three tests have the same effect size, and therefore the same power.

We calculate Cohen's f following the

$$f = \frac{\sigma_m}{\sigma}$$

. The calculation of σ_m depends on the design. For a 2x2 interaction it is based on the additive effect, or the residuals after removing main effects of factor A and B.

```
#We calculate Cohen's f following the f$\sigma_m
f <- sqrt(sum(c(a1, a2, b1, b2)^2)/length(mu))/sd #based on G*power manual page 28

#Analytic power formula for interaction
k1 <- 2 #levels in factor 1
k2 <- 2 #levels in factor 1
m <- 1 #number of measurement per group (1 because between design)
e <- 1 #non-spericity correction
r <- 0.0 #correlation between dependent variables (0 in a between design)
alpha <- 0.05 #alpha level for each test
df1 <- (k1-1) * (k2-1) * e #df for effect in interaction
df2 <- (n * (k1*k2) - (k1*k2)) * e #df_error
lambda <- (k1*k2) * n * (m/(1 + (m - 1) * r)) * f^2 # lambda
F_critical <- qf(alpha, df1, df2, lower.tail=FALSE) # Critical F-Value
pow <- pf(F_critical, df1, df2, lambda, lower.tail = FALSE) # power
pow #power
```

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

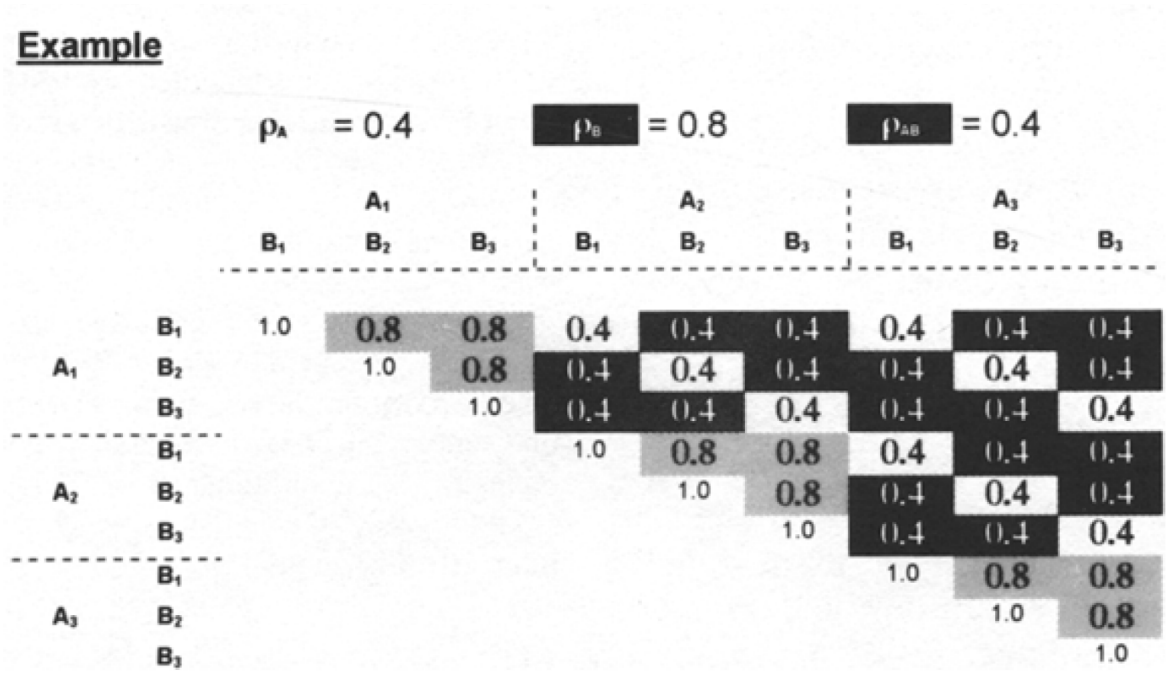


Figure 1. Representation of a correlation matrix for a 3 (A) × 3 (B) RM ANOVA: General form and numeric example. ρ_A and ρ_B represent the average correlation among the A and B (pooled) trials, respectively, and ρ_{AB} represents the average correlation among the AB coefficients having dissimilar levels.

Figure 1:

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: $\sigma_e^2 = \sigma^2(1 - \overline{\rho}_A) + \sigma^2(q-1)(\overline{\rho}_B - \overline{\rho}_{AB})$

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

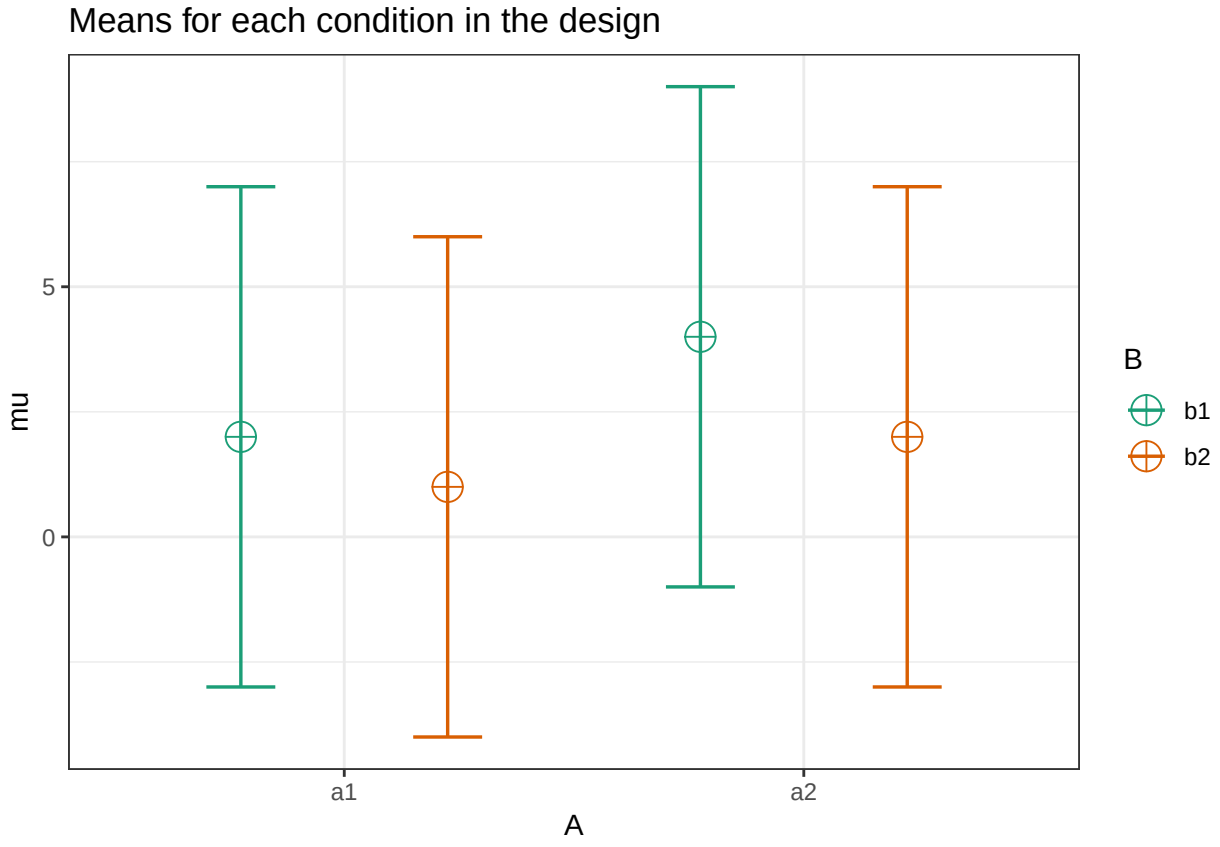
For the interaction between A and B: $\sigma_e^2 = \sigma^2(1 - \rho_{\max}) - \sigma^2(\overline{\rho}_{\min} - \overline{\rho}_{AB})$

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   26.972      0.0984
## anova_B   64.231      0.2448
## anova_A:B 26.866      0.0982
```

```
##
## Power and Effect sizes for contrasts
##                          power effect size
## p_A_a1_B_b1_A_a1_B_b2 27.014     -0.3297
## p_A_a1_B_b1_A_a2_B_b1 39.820      0.4176
## p_A_a1_B_b1_A_a2_B_b2  4.972      0.0003
## p_A_a1_B_b2_A_a2_B_b1 64.220      0.5719
## p_A_a1_B_b2_A_a2_B_b2 13.565      0.2087
## p_A_a2_B_b1_A_a2_B_b2 76.624     -0.6591
```

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).

```r
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
```

```r
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] -0.25  0.25  0.25 -0.25
```

```r
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
```

```r
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
```

```r
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
```

```r
# 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
```

```r
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
```

```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] 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.

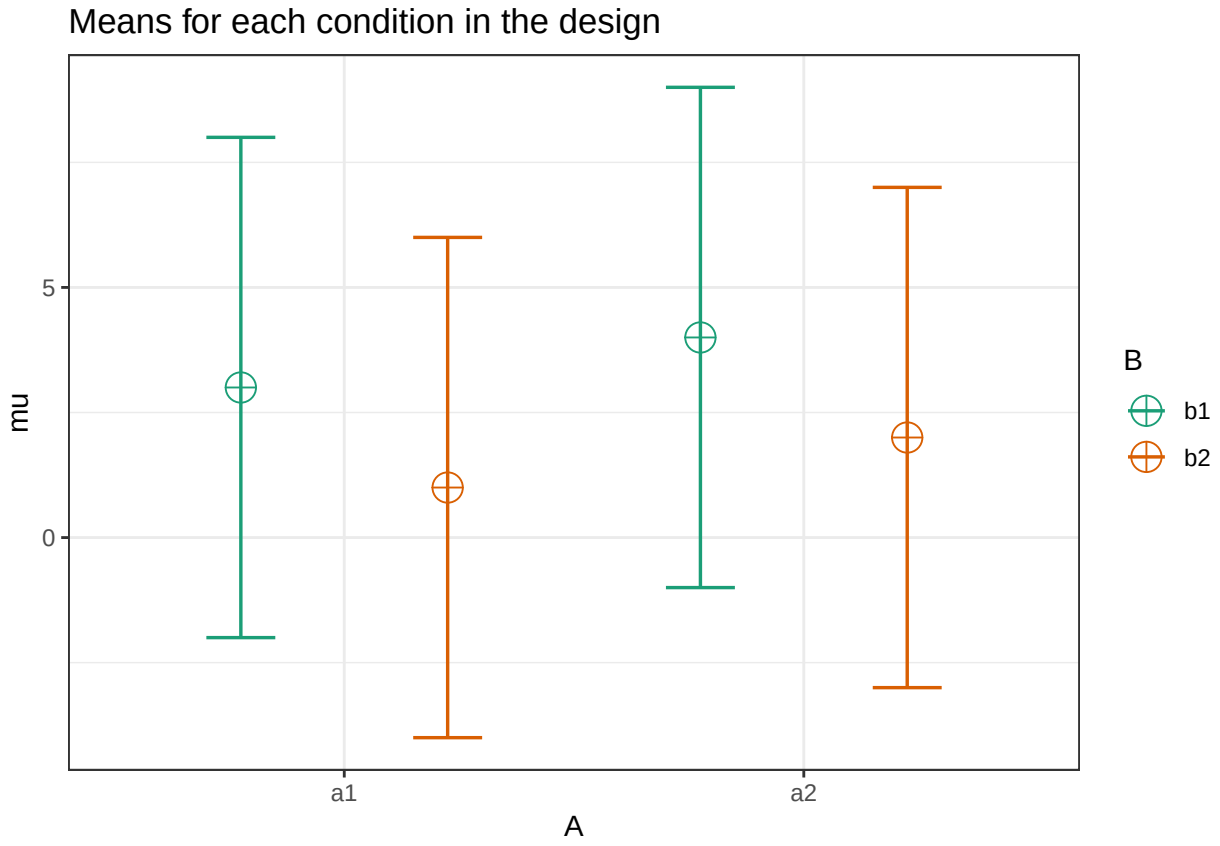
Variation 2x2 within design

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   15.815      0.0567
## anova_B   96.657      0.4641
## anova_A:B  5.068      0.0241
##
## Power and Effect sizes for contrasts
##                        power effect size
```

```
## p_A_a1_B_b1_A_a1_B_b2 76.372     -0.6580
## p_A_a1_B_b1_A_a2_B_b1 13.489      0.2086
## p_A_a1_B_b1_A_a2_B_b2 13.626     -0.2082
## p_A_a1_B_b2_A_a2_B_b1 71.970      0.6252
## p_A_a1_B_b2_A_a2_B_b2 13.606      0.2084
## p_A_a2_B_b1_A_a2_B_b2 76.770     -0.6592
```

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
```

```r
# Potvin & 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
```

```r
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] 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
```

```r
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
```