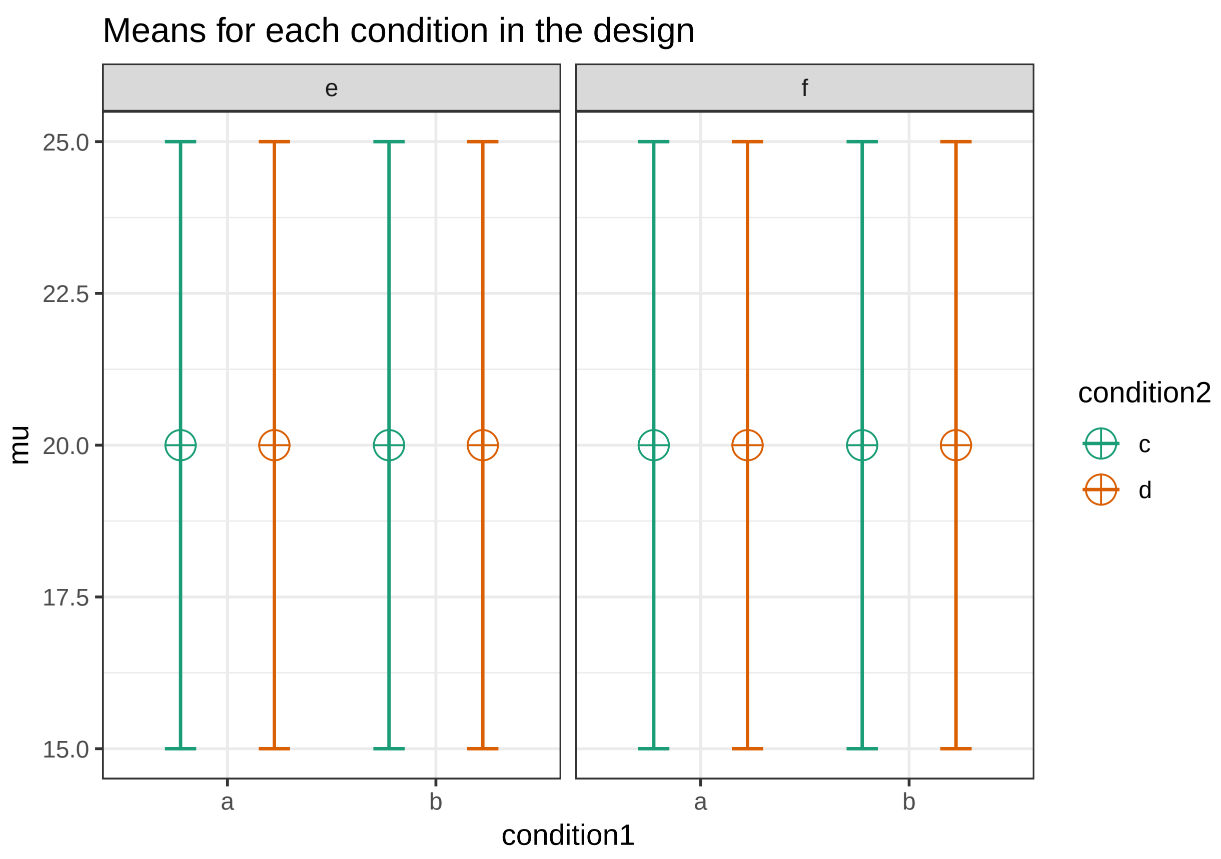
```r
knitr::opts_chunk$set(echo = TRUE)
nsims <- 100000 #set number of simulations
library(mvtnorm)
library(afex)
library(emmeans)
library(ggplot2)
library(gridExtra)
library(reshape2)
library(pwr)

# 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")
source("https://raw.githubusercontent.com/Lakens/ANOVA_power_simulation/master/mu_from_ES.R")
```

Error Control in Exploratory ANOVA

In a 2X2X2 design, an ANOVA will give the test results for three main effects, three two-way interactions, and one three-way interaction. That's 7 statistical tests. The probability of making at least one Type 1 error in a single 2x2x2 ANOVA is 1-(0.95)^7 = 30%.

```r
string <- "2b*2b*2b"
n <- 50
mu <- c(20, 20, 20, 20, 20, 20, 20, 20) #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("condition1", "a", "b", "condition2", "c", "d", "condition3", "e", "f") #
# 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_condition1                        5.052      0.0012
## anova_condition2                        4.968      0.0012
## anova_condition3                        5.065      0.0012
## anova_condition1:condition2             4.927      0.0011
## anova_condition1:condition3             4.888      0.0012
## anova_condition2:condition3             4.972      0.0012
## anova_condition1:condition2:condition3 5.080      0.0011
##
## Power and Effect sizes for contrasts
##                                                                          power effect size
## p_condition1_a_condition2_c_condition3_e_condition1_a_condition2_c_condition3_f 4.936     -0.0001
## p_condition1_a_condition2_c_condition3_e_condition1_a_condition2_d_condition3_e 4.997     -0.0001
## p_condition1_a_condition2_c_condition3_e_condition1_a_condition2_d_condition3_f 4.916     -0.0007
## p_condition1_a_condition2_c_condition3_e_condition1_b_condition2_c_condition3_e 4.880     -0.0003
## p_condition1_a_condition2_c_condition3_e_condition1_b_condition2_c_condition3_f 5.027     -0.0003
## p_condition1_a_condition2_c_condition3_e_condition1_b_condition2_d_condition3_e 5.014     -0.0006
## p_condition1_a_condition2_c_condition3_e_condition1_b_condition2_d_condition3_f 5.034     -0.0006
## p_condition1_a_condition2_c_condition3_f_condition1_a_condition2_d_condition3_e 4.983      0.0000
## p_condition1_a_condition2_c_condition3_f_condition1_a_condition2_d_condition3_f 4.893     -0.0005
## p_condition1_a_condition2_c_condition3_f_condition1_b_condition2_c_condition3_e 4.981     -0.0001
## p_condition1_a_condition2_c_condition3_f_condition1_b_condition2_c_condition3_f 4.924     -0.0002
## p_condition1_a_condition2_c_condition3_f_condition1_b_condition2_d_condition3_e 4.979     -0.0005
## p_condition1_a_condition2_c_condition3_f_condition1_b_condition2_d_condition3_f 4.988     -0.0005
## p_condition1_a_condition2_d_condition3_e_condition1_a_condition2_d_condition3_f 4.845     -0.0005
## p_condition1_a_condition2_d_condition3_e_condition1_b_condition2_c_condition3_e 4.931     -0.0002
## p_condition1_a_condition2_d_condition3_e_condition1_b_condition2_c_condition3_f 5.094     -0.0001
## p_condition1_a_condition2_d_condition3_e_condition1_b_condition2_d_condition3_e 5.026     -0.0005
## p_condition1_a_condition2_d_condition3_e_condition1_b_condition2_d_condition3_f 5.000     -0.0005
## p_condition1_a_condition2_d_condition3_f_condition1_b_condition2_c_condition3_e 5.035      0.0004
## p_condition1_a_condition2_d_condition3_f_condition1_b_condition2_c_condition3_f 4.964      0.0004
## p_condition1_a_condition2_d_condition3_f_condition1_b_condition2_d_condition3_e 4.900      0.0001
## p_condition1_a_condition2_d_condition3_f_condition1_b_condition2_d_condition3_f 4.970      0.0001
## p_condition1_b_condition2_c_condition3_e_condition1_b_condition2_c_condition3_f 4.997      0.0000
## p_condition1_b_condition2_c_condition3_e_condition1_b_condition2_d_condition3_e 4.910     -0.0003
## p_condition1_b_condition2_c_condition3_e_condition1_b_condition2_d_condition3_f 5.067     -0.0003
## p_condition1_b_condition2_c_condition3_f_condition1_b_condition2_d_condition3_e 4.981     -0.0003
## p_condition1_b_condition2_c_condition3_f_condition1_b_condition2_d_condition3_f 4.938     -0.0003
## p_condition1_b_condition2_d_condition3_e_condition1_b_condition2_d_condition3_f 5.100     -0.0001
```

When there is no true effect, we formally do not have 'power' (which is defined as the probability of finding $p < \alpha$ if there is a true effect to be found) so the power column should be read as the 'Type 1 error rate'. Because we have saved the power simulation in the 'power_result' object, we can perform calculations on the 'sim_data' dataframe that is stored. This dataframe contains the results for the nsims simulations (e.g., 10000 rows if you ran 10000 simulations) and stores the p-values and effect size estimates for each ANOVA. The first 7 columns are the p-values for the ANOVA, first the main effects of condition 1, 2, and 3, then three two-way interactions, and finally the threeway interaction.

We can calculate the number of significant results for each test (which should be 5%) by counting the number of significant p-values in each of the 7 rows:

```
apply(as.matrix(power_result$sim_data[(1:7)]), 2,
    function(x) round(mean(ifelse(x < alpha_level, 1, 0) * 100),4))
```

```
##                       anova_condition1                       anova_condition2                       a
##                                  5.052                                  4.968
##            anova_condition2:condition3 anova_condition1:condition2:condition3
##                                  4.972                                  5.080
```

This is the Type 1 error rate for each test. When we talk about error rate inflation due to multiple comparisons, we are talking about the probability that you conclude there is an effect, when there is actually no effect, when there is a significant effect for the main effect of condition 1, or condition 2, or condition 3, or for the two-way interaction between condition 1 and 2, or condition 1 and 3, or condition 2 and 3, or in the threeway interaction.

To calculate this error rate we do not just add the 7 error rates (so 7 * 5% - 35%). Instead, we calculate the probability that there will be at least one significant result in an ANOVA we perform. Some ANOVA results will have multiple significant results, just due to the Type 1 error rate (e.g., a significant result for the threeway interaction, and for the main effect of condition 1) but such an ANOVA is counted only once. Iwe calculate this percentage from our simulations, we see the number is indeed very close to 1-(0.95)^7 = 30%.

```r
sum(apply(as.matrix(power_result$sim_data[(1:7)]), 1,
    function(x) round(mean(ifelse(x < alpha_level, 1, 0) * 100),4)) > 0)/nsims*100
```

```
## [1] 29.997
```

The question is what we should do about this alpha inflation. It is undesirable if you perform exploratory ANOVA's and are fooled too often by Type 1 errors, which will not replicate if you try to build on them. Therefore, you need to control the Type 1 error rate.

In the simulation code, which relies on the afex package, there is the option to set p_adjust. In the simulation above, p_adjust was set to "none". This means no adjustment is mage to which p-values are considered to be significant, and the alpha level is used as it is set in the simulation (above this was 0.05).

Afex relies on the p.adjust functon in the stats package in R (more information is available here). From the package details:

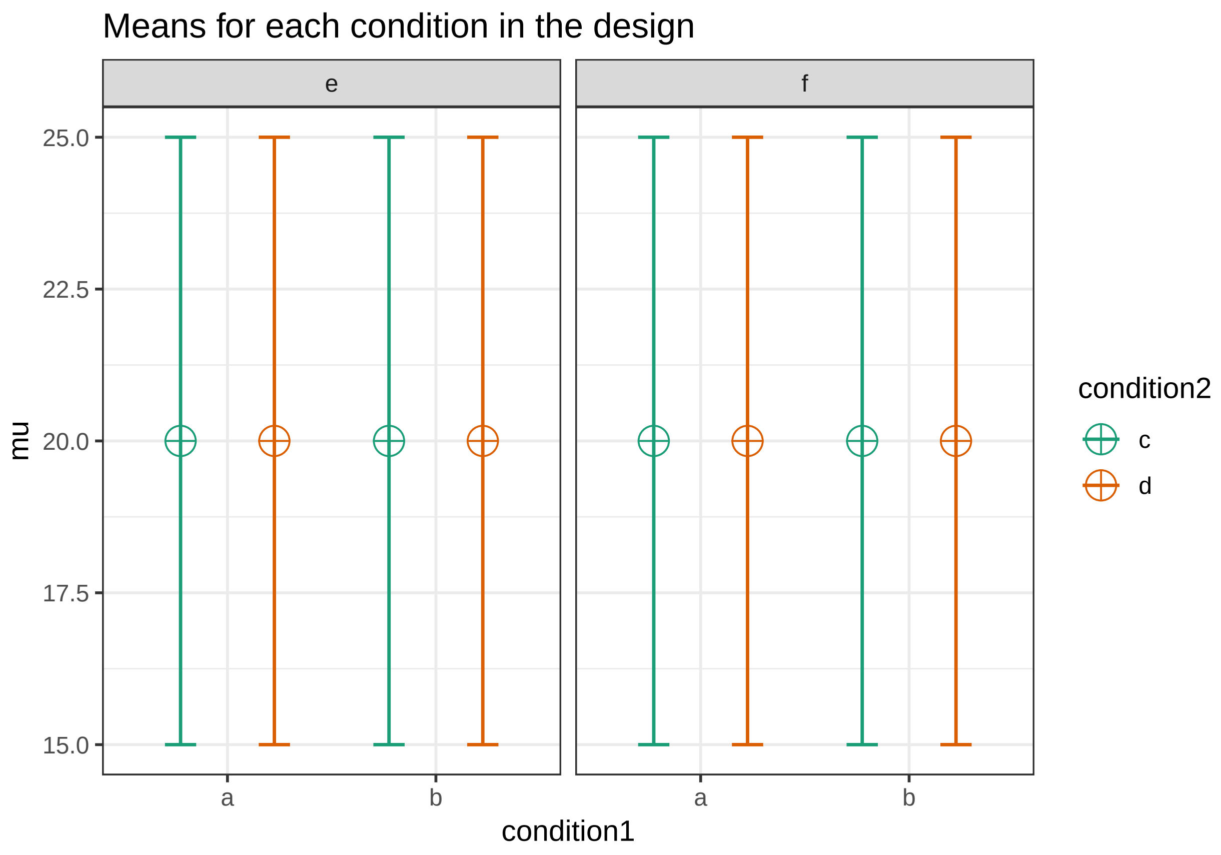
The adjustment methods include the Bonferroni correction ("bonferroni") in which the p-values are multiplied by the number of comparisons. Less conservative corrections are also included by Holm (1979) ("holm"), Hochberg (1988) ("hochberg"), Hommel (1988) ("hommel"), Benjamini & Hochberg (1995) ("BH" or its alias "fdr"), and Benjamini & Yekutieli (2001) ("BY"), respectively. A pass-through option ("none") is also included. The first four methods are designed to give strong control of the family-wise error rate. There seems no reason to use the unmodified Bonferroni correction because it is dominated by Holm's method, which is also valid under arbitrary assumptions.

Hochberg's and Hommel's methods are valid when the hypothesis tests are independent or when they are non-negatively associated (Sarkar, 1998; Sarkar and Chang, 1997). Hommel's method is more powerful than Hochberg's, but the difference is usually small and the Hochberg p-values are faster to compute.

The "BH" (aka "fdr") and "BY" method of Benjamini, Hochberg, and Yekutieli control the false discovery rate, the expected proportion of false discoveries amongst the rejected hypotheses. The false discovery rate is a less stringent condition than the family-wise error rate, so these methods are more powerful than the others.

Let's re-run the simulation twith the Holm-Bonferroni correction, which is simple and require no assumptions.

```r
string <- "2b*2b*2b"
n <- 50
mu <- c(20, 20, 20, 20, 20, 20, 20, 20) #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 = "holm"
# Changed to Holm-Bonferroni
labelnames <- c("condition1", "a", "b", "condition2", "c", "d", "condition3", "e", "f") #
# 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
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_condition1                        0.785      0.0011
## anova_condition2                        0.721      0.0011
## anova_condition3                        0.696      0.0012
## anova_condition1:condition2             0.770      0.0012
## anova_condition1:condition3             0.752      0.0012
## anova_condition2:condition3             0.707      0.0012
## anova_condition1:condition2:condition3 0.726      0.0012
##
## Power and Effect sizes for contrasts
##                                                                            power effect size
## p_condition1_a_condition2_c_condition3_e_condition1_a_condition2_c_condition3_f 4.944     -0.0002
## p_condition1_a_condition2_c_condition3_e_condition1_a_condition2_d_condition3_e 5.026      0.0000
## p_condition1_a_condition2_c_condition3_e_condition1_a_condition2_d_condition3_f 5.039     -0.0005
## p_condition1_a_condition2_c_condition3_e_condition1_b_condition2_c_condition3_e 5.003      0.0014
## p_condition1_a_condition2_c_condition3_e_condition1_b_condition2_c_condition3_f 5.035      0.0000
## p_condition1_a_condition2_c_condition3_e_condition1_b_condition2_d_condition3_e 4.922      0.0005
## p_condition1_a_condition2_c_condition3_e_condition1_b_condition2_d_condition3_f 4.953     -0.0005
## p_condition1_a_condition2_c_condition3_f_condition1_a_condition2_d_condition3_e 5.127      0.0003
## p_condition1_a_condition2_c_condition3_f_condition1_a_condition2_d_condition3_f 4.976     -0.0003
## p_condition1_a_condition2_c_condition3_f_condition1_b_condition2_c_condition3_e 5.059      0.0016
## p_condition1_a_condition2_c_condition3_f_condition1_b_condition2_c_condition3_f 5.166      0.0004
## p_condition1_a_condition2_c_condition3_f_condition1_b_condition2_d_condition3_e 5.024      0.0008
## p_condition1_a_condition2_c_condition3_f_condition1_b_condition2_d_condition3_f 4.872     -0.0002
## p_condition1_a_condition2_d_condition3_e_condition1_a_condition2_d_condition3_f 5.198     -0.0006
## p_condition1_a_condition2_d_condition3_e_condition1_b_condition2_c_condition3_e 5.050      0.0013
## p_condition1_a_condition2_d_condition3_e_condition1_b_condition2_c_condition3_f 5.075      0.0000
## p_condition1_a_condition2_d_condition3_e_condition1_b_condition2_d_condition3_e 5.129      0.0004
## p_condition1_a_condition2_d_condition3_e_condition1_b_condition2_d_condition3_f 5.044     -0.0005
## p_condition1_a_condition2_d_condition3_f_condition1_b_condition2_c_condition3_e 4.931      0.0019
## p_condition1_a_condition2_d_condition3_f_condition1_b_condition2_c_condition3_f 5.058      0.0006
## p_condition1_a_condition2_d_condition3_f_condition1_b_condition2_d_condition3_e 5.049      0.0011
## p_condition1_a_condition2_d_condition3_f_condition1_b_condition2_d_condition3_f 4.989      0.0001
## p_condition1_b_condition2_c_condition3_e_condition1_b_condition2_c_condition3_f 5.039     -0.0012
## p_condition1_b_condition2_c_condition3_e_condition1_b_condition2_d_condition3_e 5.079     -0.0009
## p_condition1_b_condition2_c_condition3_e_condition1_b_condition2_d_condition3_f 4.939     -0.0018
## p_condition1_b_condition2_c_condition3_f_condition1_b_condition2_d_condition3_e 5.007      0.0004
## p_condition1_b_condition2_c_condition3_f_condition1_b_condition2_d_condition3_f 4.918     -0.0005
## p_condition1_b_condition2_d_condition3_e_condition1_b_condition2_d_condition3_f 4.981     -0.0010
```

If we now calculate the overall Type 1 error rate:

```r
sum(apply(as.matrix(power_result$sim_data[(1:7)]), 1,
    function(x) round(mean(ifelse(x < alpha_level, 1, 0) * 100),4)) > 0)/nsims*100
```

```
## [1] 4.991
```

We see it is close to 5%. Note that error rates have variation, and even in a few thousand simulations, the error rate in the sample of studies can easily be half a percentage point higher or lower. But *in the long run* the error rate should equal the alpha level. Furthermore, note that the Holm-bonferroni method is slightly more powerful than the Bonferroni procedure (which is simply α divided by the numner of tests). There are more powerful procedures to control the Type 1 error rate, which require more assumptions. For a small number of tests, they Holm-Bonferroni procedure works well. Alternative procedure to control error rates can be found in the multcomp R package.