Power Analysis with Superpower

Aaron R. Caldwell & Daniël Lakens

2019-12-11

Contents

This is a compilation of documents for Superpower R package written in **Markdown** (Allaire et al., 2019) and compiled by **Bookdown** (Xie, 2019).

Preface

This book is still being developed. You may find typos in the book.

The goal of `Superpower` is to easily simulate factorial designs and empirically calculate power using a simulation approach. The R package is intended to be utilized for prospective (a priori) power analysis. Please don't be silly and compute post hoc power.

This package, and book, expect readers to have some familiarity with R (2019). However, we have created two Shiny apps (for the `ANOVA_power` & `ANOVA_exact` functions respectively) to help use `Superpower` if you are not familiar with R.

In this book we will display a variety of ways the `Superpower` package can be used for power analysis and sample size planning for factorial experimental designs. We also included various examples of the performance of `Superpower` against other R packages (e.g., `pwr2ppl` by Aberson (2019) and `pwr` by Champely (2018)) and statistical programs (such as G*Power Faul et al. (2007), MOREpower Campbell and Thompson (2012), and SAS's `PROC GLMPOWER` (2015)). All uses of the `ANOVA_power` function have been run with 10000 iterations (`nsims = 10000`). If you have any issues using `Superpower` or want to expand its capabilities please raise the issue on our GitHub repository.

Chapter 1

The Experimental Design

This section introduces how `Superpower` allows the user to specify their experimental design. Currently, this pacakge only provides power analysis for ANOVAs (and MANOVA for repeated measures) so all designs are currently defined by the `ANOVA_design` function.

1.1 ANOVA_design function

Currently the `ANOVA_design` function can create designs with up to three factors, for both within, between, and mixed designs. It requires the following input: design, n, mu, sd, r, and optionally allows you to set labelnames.

1. `design`: string that specifies the design (see below).
2. `n`: the sample size for each between subject condition.
3. `mu`: a vector with the means for each condition.
4. `sd`: the population standard deviation. Assumes homogeneity of variances (only one standard deviation can be provided).
5. `r`: the correlation(s) for within designs (or 0 for between designs).
6. `labelnames`: This is an optonal vector of words that indicates factor names and level names (see below).
7. A final optional setting is to specify if you want to automatically print a plot or not (`plot` = TRUE or FALSE)

1.1.1 Specifying the design using `design`

The `design` option is used to specify the design. Every factor is specified with a number, indicating the number of levels of the factor, and a letter, b or w, to indicate whether the factor is manipulated between or within participants. For

example, a `2b` design has two between-participant groups. A `12w` design has one factor with 12 levels, all manipulated within-participants. A `2b*3w` is a design with two factors (a 2b factor and a 3w factor), the first of which has 2 between participant levels (2b), and the second of which has 3 within participants levels (3w). **If there are multiple factors (the functions take up to three different factors) separate factors with a * (asterisk)**. An example of a `2b*3w` design is a group of people in one condition who get a drug, and a group of people in another condition who get a placebo (hence `2b`), and we measure their health before they take the pill, one day after they take the pill, and a week after they take the pill (hence the `3w`).

1.1.2 Specifying the means using `mu`

Note that for each cell in the design, a mean must be provided. Thus, for a `2b*3w` design, **6** (i.e., 2*3=6) means need to be entered.

Means need to be entered in the correct order. ANOVA_design outputs a plot so you can check if you entered all means as you intended. Always carefully check if the plot that is generated matches your expectations.

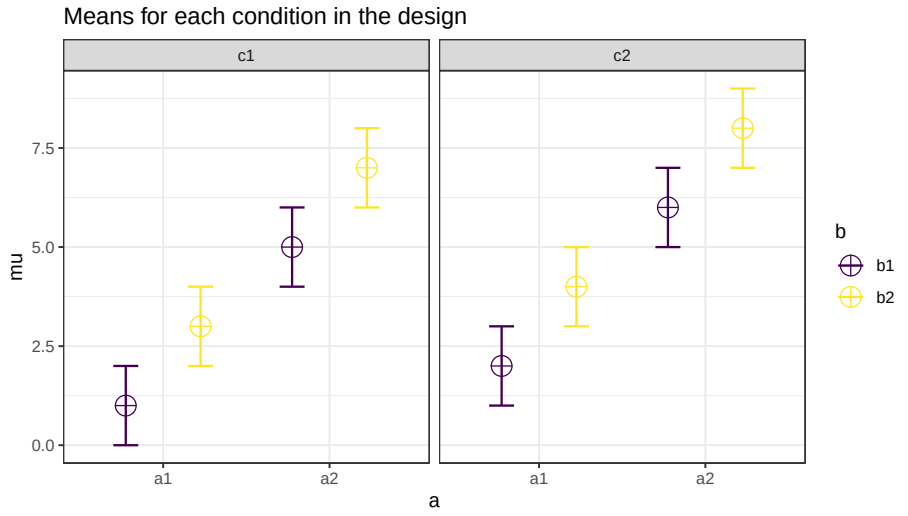
The general principle is that the code generates factors, indicated by the factor names you entered in the labelnames variable, (i.e., *condition* and *time*). Levels are indicated by factor names and levels (e.g., control_time1, control_time2, control_time3, etc).

If your design has just one factor, just enter the means in the same order as the labelnames (see below). For more factors, note the general pattern in the example below. Means are entered in the following order for a 3 factors design (each with 2 levels):

1. a1 b1 c1
2. a1 b1 c2
3. a1 b2 c1
4. a1 b2 c2
5. a2 b1 c1
6. a2 b1 c2
7. a2 b2 c1
8. a2 b2 c2

So if you enter the means 1, 2, 3, 4, 5, 6, 7, 8 the first 4 means correspond to level 1 of factor 1, the second 4 means correspond to level 2 of factor 1. Within the first 4 means, the first 2 correspond to level 1 of factor 2, and within those 2 means, the first corresponds to level 1 of factor 3.

The plot below visualizes means from 1 to 8 being entered in a vector: `mu = c(1, 2, 3, 4, 5, 6, 7, 8)` so you can see how the basic ordering works.



1.1.3 Specifying label names

To make sure the plots and tables with simulation results are easy to interpret, it really helps to name all factors and levels. You can enter the labels in the 'labelnames' variable. You can also choose not to specify names. Then all factors are indicated by letters (a, b, c) and all levels by numbers (a1, a2, a3).

For the 2x3 design we have been using as an example, where there are 2 factors (condition and time of measurement), the first with 2 levels (placebo vs. medicine) and the second with three levels (time1, time2, and time3) we would enter the labels as follows:

```
c("condition", "placebo", "medicine", "time", "time1", "time2",
"time3")
```

As you can see, you follow the order of the design (`2b*3w`), and first write the **FACTOR** label (condition) followed by the levels of that factor (placebo and medicine). Then you write the second factor name (time) followed by the three labels for each **LEVEL** (time1, time2, time3). **Do not use spaces or special characters in the names (so not "time 1" or "time_1" but "time1").**

Some examples:

1. One within factor (time with 2 levels), 2w: `c("time", "morning", "evening")`
2. Two between factors (time and group, each with 2 levels), 2b*2b: `c("time", "morning", "evening", "group", "control", "experimental")`
3. Two between factors (time and group, first with 4 levels, second with 2 levels), 4b*2b: `c("time", "morning", "afternoon" ,"evening", "night", "group", "control", "experimental")`

1.1.4 Specifying the correlation

Depending on whether factors are manipulated within or between, variables are correlated, or not. You can set the correlation for within-participant factors. You can either assume all factors have the same correlation (e.g., r = 0.7), or enter the correlations for each pair of observations separately by specifying a correlation matrix.

In a 2x2 design, with factors A and B, each with 2 levels, there are 6 possible comparisons that can be made.

1. A1 vs. A2
2. A1 vs. B1
3. A1 vs. B2
4. A2 vs. B1
5. A2 vs. B2
6. B1 vs. B2

The number of possible comparisons is the product of the levels of all factors squared minus the product of all factors, divided by two. For a 2x2 design where each factor has two levels, this is:

```
(((2 * 2) ^ 2) - (2 * 2))/2
```

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

The number of possible comparisons increases rapidly when adding factors and levels for each factor. For example, for a 2x2x4 design it is:

```
(((2 * 2 * 4) ^ 2) - (2 * 2 * 4))/2
```

```
## [1] 120
```

Each of these comparisons can have their own correlation if the factor is manipulated within subjects (if the factor is manipulated between subjects the correlation is 0). These correlations determine the covariance matrix. Potvin and Schutz (2000) surveyed statistical tools for power analysis and conclude that most software packages are limited to one factor repeated measure designs and do not provide power calculations for within designs with multiple factor (which is still true for software such as G*Power). Furthermore, software solutions which were available at the time (DATASIM by Bradley, Russel, & Reeve, 1996) required researchers to assume correlations were of the same magnitude for all within factors, which is not always realistic. If you do not want to assume

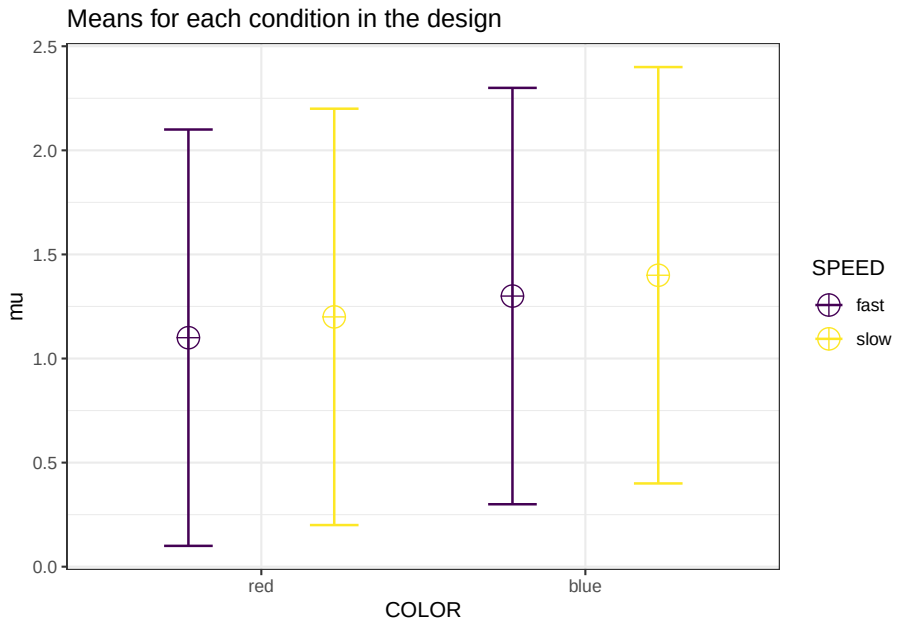
equal correlations for all paired comparisons, you can specify the correlation for each possible comparison.

The order in which the correlations are entered in the vector should match the covariance matrix. The order for a 2x2 design is given in the 6 item list above. The general pattern is that the matrix is filled from top to bottom, and left to right, illustrated by the increasing correlations in the table below. The diagonal is generated dynamically (based on the standard deviation).

Matrix	a1_b1	a1_b2	a2_b1	a2_b2
a1_b1	1.00	0.91	0.92	0.93
a1_b2	0.91	1.00	0.94	0.95
a2_b1	0.92	0.94	1.00	0.96
a2_b2	0.93	0.95	0.90	1.00

We would enter this correlation matrix in the `ANOVA_design` function as:

```
design_result <- ANOVA_design(design = "2w*2w",
                              n = 80,
                              mu = c(1.1, 1.2,
                                     1.3, 1.4),
                              sd = 1,
                              r <- c(0.91, 0.92,
                                     0.93, 0.94,
                                     0.95, 0.96),
                              labelnames = c("COLOR",
                                             "red", "blue",
                                             "SPEED",
                                             "fast", "slow"),
                              plot = TRUE)
```



We can check the correlation matrix by asking for it from the design_result object to check if it was entered the way we wanted:

```
design_result$cor_mat
```

Table 1.2: Correlation Matrix

	red_fast	red_slow	blue_fast	blue_slow
red_fast	1.00	0.91	0.92	0.93
red_slow	0.91	1.00	0.94	0.95
blue_fast	0.92	0.94	1.00	0.96
blue_slow	0.93	0.95	0.96	1.00

We should also check the covariance-variance matrix to ensure the `ANOVA_design` function working properly. The variance should be the diagonal element while the off-diagonal elements should be equal to `covariance = correlation*variance` or $cov_{x,y} = \frac{\sum_{i=1}^{N}(x_i-\bar{x})(y_i-\bar{y})}{N-1}$. In this case, it is identical to the correlation matrix because the variance is equal to 1.

```
design_result$sigmatrix
```

Table 1.3: Covariance-Variance Matrix

	red_fast	red_slow	blue_fast	blue_slow
red_fast	1.00	0.91	0.92	0.93
red_slow	0.91	1.00	0.94	0.95
blue_fast	0.92	0.94	1.00	0.96
blue_slow	0.93	0.95	0.96	1.00

1.1.5 Specifying the sample size

You can set the sample size **per condition** by setting a value for `n`. The assumption is that you will collect equal sample sizes in all conditions [expanding Superpower to allow different sample sizes in each group is a planned future option].

This means for the `2w*2w` design above there is a total of 80 participants/subjects with a total of 320 observations.

1.1.6 Specifying the standard deviation

You can set the standard deviation(s) by setting a value of `sd`. Currently, `Superpower` allows you to **violate the assumption of homogeneity of variance**. This will affect the type I error rate if the differences in `sd` between conditions are extreme. Note that there is always some uncertainty in which values you can expect in the study you are planning. It is therefore useful to perform sensitivity analyses (e.g., running the simulation with the expected standard deviation, but also with more conservative or even worst-case-scenario values).

Chapter 2

One-Way ANOVA

2.1 Part 1

Using the formula also used by Albers and Lakens (2018), we can determine the means that should yield a specified effect sizes (expressed in Cohen's f). Eta-squared (identical to partial eta-squared for one-way ANOVA's) has benchmarks of .0099, .0588, and .1379 for small, medium, and large effect sizes (Cohen, 1988). Athough these benchmarks are quite arbitrary, and researchers should only use such benchmarks for power analyses as a last resort, we will demonstrate an a priori power analysis for these values.

2.1.1 Two conditions

Imagine we aim to design a study to test the hypothesis that giving people a pet to take care of will increase their life satisfaction. We have a control condition, and a condition where people get a pet, and randomly assign participants to either condition. We can simulate a one-way ANOVA with a specified alpha, sample size, and effect size, on see the statistical power we would have for the ANOVA and the follow-up comparisons. We expect pets to increase life-satisfaction compared to the control condition. Based on work by Pavot and Diener (1993) we believe that we can expect responses on the life-satifaction scale to have a mean of approximately 24 in our population, with a standard deviation of 6.4. We expect having a pet increases life satisfaction with approximately 2.2 scale points for participants who get a pet. 200 participants in total, with 100 participants in each condition. But before we proceed with the data collection, we examine the statistical power our design would have to detect the differences we predict.

```
string <- "2b"
n <- 100
# We are thinking of running 50 peope in each condition
mu <- c(24, 26.2)
# Enter means in the order that matches the labels below.
# In this case, control, cat, dog.
sd <- 6.4
labelnames <- c("condition", "control", "pet") #
# the label names should be in the order of the means specified above.
design_result <- ANOVA_design(design = string,
                   n = n,
                   mu = mu,
                   sd = sd,
                   labelnames = labelnames)
alpha_level <- 0.05
# You should think carefully about how to justify your alpha level.
# We will give some examples later, but for now, use 0.05.
```

```
simulation_result <- ANOVA_power(design_result,
                                 alpha_level = alpha_level,
                                 nsims = nsims,
                                 verbose = FALSE)
```

Table 2.1: Simulated ANOVA Result

	power	effect_size
anova_condition	67.2	0.0331418

```
exact_result <- ANOVA_exact(design_result,
                            alpha_level = alpha_level,
                            verbose = FALSE)
```

Table 2.2: Exact ANOVA Result

	power	partial_eta_squared	cohen_f	non_centrality
condition	67.68572	0.0289748	0.1727409	5.908203

The result shows that we have exactly the same power for the ANOVA, as we have for the *t*-test. When there are only two groups, these tests are mathematically identical. In a study with 100 participants, we would have quite low power (around 67.7%). An ANOVA with 2 groups is identical to a *t*-test. For our example, Cohen's d (the standardized mean difference) is 2.2/6.4, or d =

0.34375 for the difference between the control condition and pets, which we can use to easily compute the expected power for these simple comparisons using the `pwr` (2018) package.

```
pwr.t.test(d = 2.2/6.4,
           n = 100,
           sig.level = 0.05,
           type = "two.sample",
           alternative = "two.sided")$power
```

```
## [1] 0.6768572
```

We can also directly compute Cohen's f from Cohen's d for two groups, as Cohen (1988) describes, because $f = 1/2$d. So $f = 0.5$*$0.34375 = 0.171875$. And indeed, power analysis using the pwr package yields the same result using the `pwr.anova.test` as the `power.t.test`.

```
K <- 2
n <- 100
f <- 0.171875
pwr.anova.test(n = n,
               k = K,
               f = f,
               sig.level = alpha_level)$power
```

```
## [1] 0.6768572
```

This analysis tells us that running the study with 100 participants in each condition is too likely to *not* yield a significant test result, even if our expected pattern of differences is true. This is not optimal.

Let's mathematically explore which pattern of means we would need to expect to habe 90% power for the ANOVA with 50 participants in each group. We can use the pwr package in R to compute a sensitivity analysis that tells us the effect size, in Cohen's *f*, that we are able to detect with 3 groups and 50 partiicpants in each group, in order to achive 90% power with an alpha level of 5%.

```
K <- 2
n <- 100
sd <- 6.4
r <- 0
#Calculate f when running simulation
f <- pwr.anova.test(n = n,
                    k = K,
```

```
                    power = 0.9,
                    sig.level = alpha_level)$f
f
```

```
## [1] 0.2303587
```

This sensitivity analysis shows we have 90% power in our planned design to detect effects of Cohen's f of 0.2303587. Benchmarks by Cohen (1988) for small, medium, and large Cohen's f values are 0.1, 0.25, and 0.4, which correspond to eta-squared values of small (.0099), medium (.0588), and large (.1379), in line with d = .2, .5, or .8. So, at least based on these benchmarks, we have 90% power to detect effects that are slightly below a medium effect benchmark.

```
f2 <- f^2
ES <- f2 / (f2 + 1)
ES
```

```
## [1] 0.0503911
```

Expressed in eta-squared, we can detect values of eta-squared = 0.05 or larger.

```
mu <- mu_from_ES(K = K, ES = ES)
mu <- mu * sd
mu
```

```
## [1] -1.474295  1.474295
```

We can compute a pattern of means, given a standard deviation of 6.4, that would give us an effect size of $f = 0.23$, or eta-squared of 0.05. We should be able to accomplish this is the means are -1.474295 and 1.474295. We can use these values to confirm the ANOVA has 90% power.

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

```
simulation_result <- ANOVA_power(design_result,
                                 alpha_level = alpha_level,
                                 nsims = nsims,
                                 verbose = FALSE)
```

Table 2.3: Simulated ANOVA Result

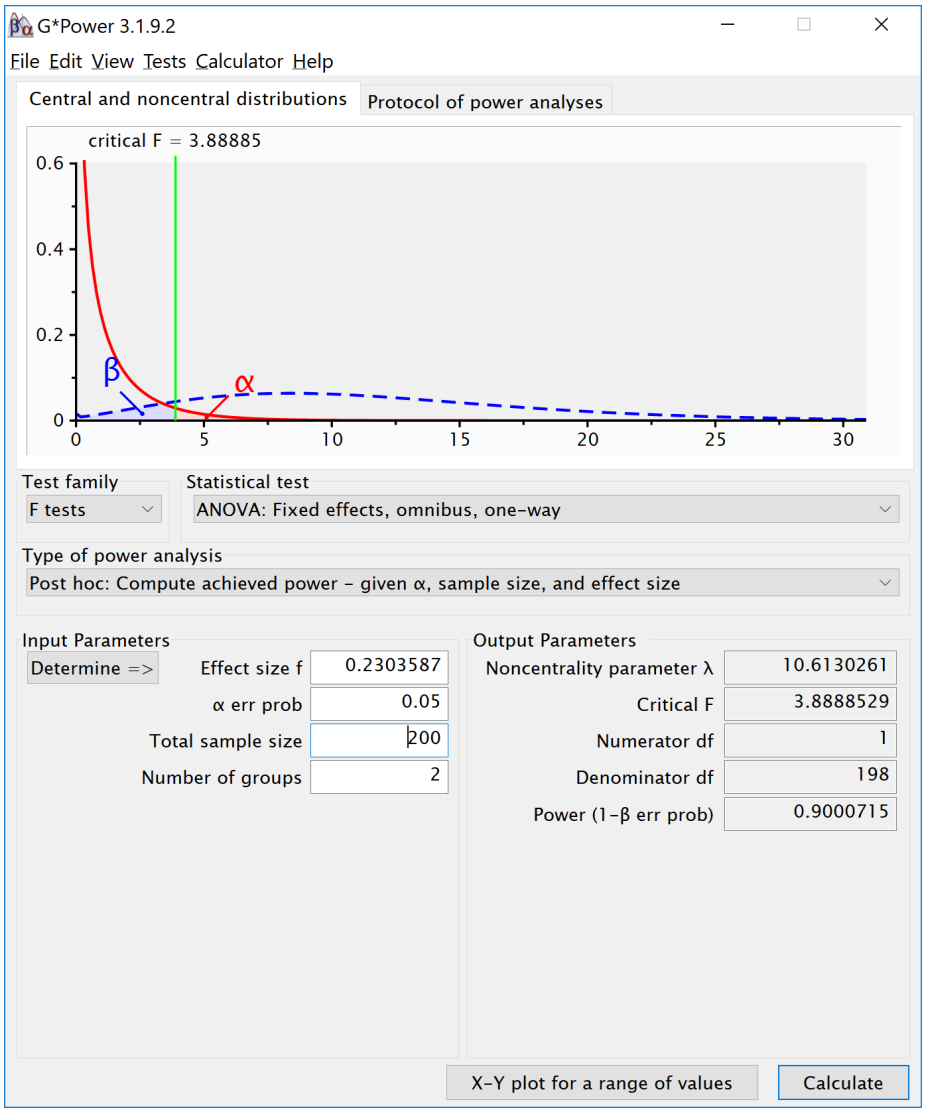
	power	effect_size
anova_condition	90.15	0.0544974

```
exact_result <- ANOVA_exact(design_result,
                            alpha_level = alpha_level,
                            verbose = FALSE)
```

Table 2.4: Exact ANOVA Result

	power	partial_eta_squared	cohen_f	non_centrality
condition	90.00714	0.0508742	0.2315192	10.61302

The simulation confirms that for the F-test for the ANOVA we have 90% power. This is also what Gpower tells us what would happen based on a post-hoc power analysis with an f of 0.2303587, 2 groups, 200 participants in total (100 in each between subject condition), and an alpha of 5%.



If we return to our expected means, how many participants do we need for sufficient power? Given the expected difference and standard deviation, d = 0.34375, and $f = 0.171875$. We can perform an a priori power analysis for this simple case, which tells us we need 179 participants in each group (we can't split people in parts, and thus always round a power analysis upward), or 358 in total.

```
K <- 2
power <- 0.9
f <- 0.171875
```

```
pwr.anova.test(power = power,
               k = K,
               f = f,
               sig.level = alpha_level)
```

```
##
##      Balanced one-way analysis of variance power calculation
##
##               k = 2
##               n = 178.8104
##               f = 0.171875
##       sig.level = 0.05
##           power = 0.9
##
## NOTE: n is number in each group
```

If we re-run the simulation with this sample size, we indeed have 90% power.

```
string <- "2b"
n <- 179
mu <- c(24, 26.2)
# Enter means in the order that matches the labels below.
# In this case, control, pet.
sd <- 6.4
labelnames <- c("condition", "control", "pet") #
# the label names should be in the order of the means specified above.
design_result <- ANOVA_design(
design = string,
n = n,
mu = mu,
sd = sd,
labelnames = labelnames
)
alpha_level <- 0.05
```

```
simulation_result <- ANOVA_power(design_result,
                                 alpha_level = alpha_level,
                                 nsims = nsims,
                                 verbose = FALSE)
```

```
exact_result <- ANOVA_exact(design_result,
                            alpha_level = alpha_level,
                            verbose = FALSE)
```

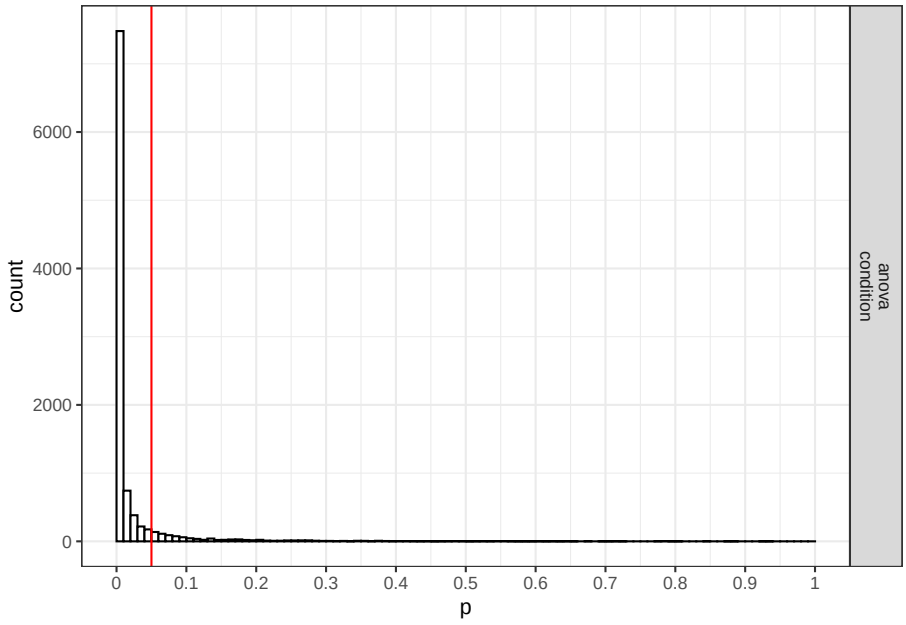
Table 2.5: Simulated ANOVA Result

	power	effect_size
anova_condition	89.99	0.0315708

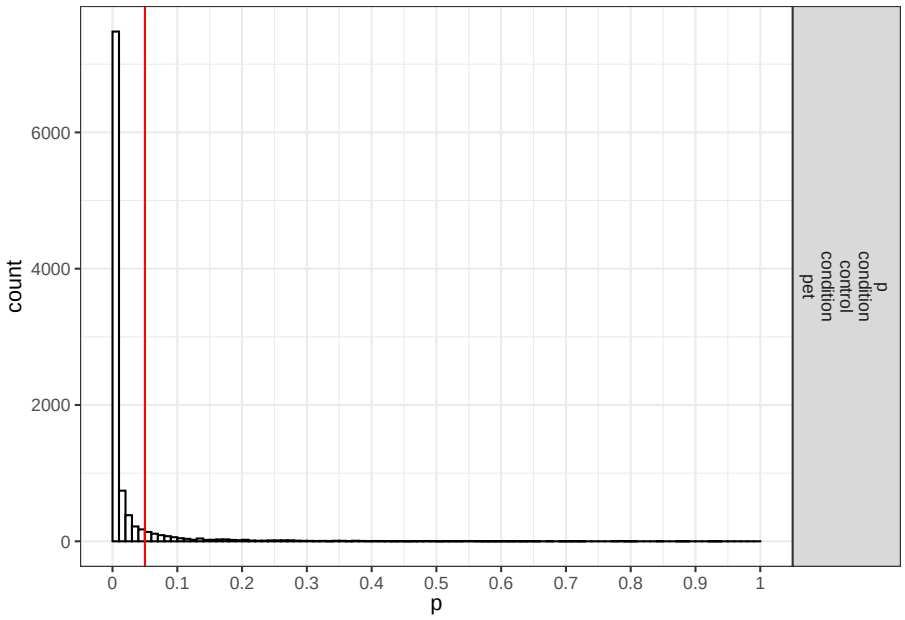
Table 2.6: Exact ANOVA Result

	power	partial_eta_squared	cohen_f	non_centrality
condition	90.03028	0.0288499	0.1723571	10.57568

We stored the result from the power analysis in an object. This allows us to request plots (which are not printed automatically) showing the p-value distribution. If we request `simulation_result$plot1` we get the p-value distribution for the ANOVA:



If we request `simulation_result$plot2` we get the p-value distribution for the paired comparisons (in this case only one):


count
6000
4000
2000
0
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
p
p
condition
control
condition
pet

Part 2

2.1.2 Three conditions

Imagine we aim to design a study to test the hypothesis that giving people a pet to take care of will increase their life satisfaction. We have a control condition, a 'cat' pet condition, and a 'dog' pet condition. We can simulate a One-Way ANOVA with a specified alpha, sample size, and effect size, on see the statistical power we would have for the ANOVA and the follow-up comparisons. We expect all pets to increase life-satisfaction compared to the control condition. Obviously, we also expect the people who are in the 'dog' pet condition to have even greater life-satisfaction than people in the 'cat' pet condition. Based on work by Pavot and Diener (1993) we believe that we can expect responses on the life-satifaction scale to have a mean of approximately 24 in our population, with a standard deviation of 6.4. We expect having a pet increases life satisfaction with approximately 2.2 scale points for participants who get a cat, and 2.6 scale points for participants who get a dog. We initially consider collecting data from 150 participants in total, with 50 participants in each condition. But before we proceed with the data collection, we examine the statistical power our design would have to detect the differences we predict.

```
string <- "3b"
n <- 50
# We are thinking of running 50 peope in each condition
mu <- c(24, 26.2, 26.6)
# Enter means in the order that matches the labels below.
# In this case, control, cat, dog.
sd <- 6.4
labelnames <- c("condition", "control", "cat", "dog") #
# the label names should be in the order of the means specified above.
design_result <- ANOVA_design(
design = string,
n = n,
mu = mu,
sd = sd,
labelnames = labelnames
)
alpha_level <- 0.05
```

```
# You should think carefully about how to justify your alpha level.
# We will give some examples later, but for now, use 0.05.
simulation_result <- ANOVA_power(design_result,
                                 alpha_level = alpha_level,
                                 nsims = nsims,
                                 verbose = FALSE)
```

Table 2.7: Simulated ANOVA Result

	power	effect_size
anova_condition	47.26	0.0433997

```
exact_result <- ANOVA_exact(design_result,
                            alpha_level = alpha_level,
                            verbose = FALSE)
```

Table 2.8: Exact ANOVA Result

	power	partial_eta_squared	cohen_f	non_centrality
condition	47.69468	0.0315259	0.180422	4.785156

The result shows that you would have quite low power with 50 participants, both for the overall ANOVA (just around 50% power), as for the follow up comparisons (approximately 40% power for the control vs cat condition, around 50% for the control vs dogs condition, and a really low power (around 6%, just above the Type 1 error rate of 5%) for the expected difference between cats and dogs.

2.1.3 Power for simple effects

We are typically not just interested in the ANOVA, but also in follow up comparisons. In this case, we would perform a *t*-test comparing the control condition against the cat and dog condition, and we would compare the cat and dog conditions against each other, in independent *t*-tests.

For our example, Cohen's d (the standardized mean difference) is 2.2/6.4, or d = 0.34375 for the difference between the control condition and cats, 2.6/6.4 of d = 0.40625 for the difference between the control condition and dogs, and 0.4/6.4 or d = 0.0625 for the difference between cats and dogs as pets.

We can easily compute the expected power for these simple comparisons using the pwr package.

```
pwr.t.test(
  d = 2.2 / 6.4,
  n = 50,
  sig.level = 0.05,
  type = "two.sample",
  alternative = "two.sided"
  )$power
```

```
## [1] 0.3983064
```

```
  pwr.t.test(
  d = 2.6 / 6.4,
  n = 50,
  sig.level = 0.05,
  type = "two.sample",
  alternative = "two.sided"
  )$power
```

```
## [1] 0.5205162
```

```
  pwr.t.test(
  d = 0.4 / 6.4,
  n = 50,
  sig.level = 0.05,
  type = "two.sample",
  alternative = "two.sided"
  )$power
```

```
## [1] 0.06104044
```

This analysis tells us that running the study with 50 participants in each condition is more likely to *not* yield a significant test result, even if our expected pattern of differences is true, than that we will observe a p-value smaller than our alpha level. This is not optimal.

Let's mathematically explore which pattern of means we would need to expect to habe 90% power for the ANOVA with 50 participants in each group. We can use the pwr package in R to compute a sensitivity analysis that tells us the effect size, in Cohen's f, that we are able to detect with 3 groups and 50 partiicpants in each group, in order to achive 90% power with an alpha level of 5%.

```
K <- 3
n <- 50
sd <- 6.4
r <- 0
#Calculate f when running simulation
f <- pwr.anova.test(n = n,
                    k = K,
                    power = 0.9,
                    sig.level = alpha_level)$f
f
```

```
## [1] 0.2934417
```

This sensitivity analysis shows we have 90% power in our planned design to detect effects of Cohen's f of 0.2934417. Benchmarks by Cohen (1988) for small, medium, and large Cohen's f values are 0.1, 0.25, and 0.4, which correspond to eta-squared values of small (.0099), medium (.0588), and large (.1379), in line with d = .2, .5, or .8. So, at least based on these benchmarks, we have 90% power to detect effects that are somewhat sizeable.

```
f2 <- f^2
ES <- f2 / (f2 + 1)
ES
```

```
## [1] 0.07928127
```

Expressed in eta-squared, we can detect values of eta-squared = 0.0793 or larger.

```
mu <- mu_from_ES(K = K, ES = ES)
mu <- mu * sd
mu
```

```
## [1] -2.300104  0.000000  2.300104
```

We can compute a pattern of means, given a standard deviation of 6.4, that would give us an effect size of f = 0.2934, or eta-squared of 0.0793. We should be able to accomplish this is the means are -2.300104, 0.000000, and 2.300104. We can use these values to confirm the ANOVA has 90% power.

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

```
simulation_result <- ANOVA_power(design_result,
                                 alpha_level = alpha_level,
                                 nsims = nsims,
                                 verbose = FALSE)
```

```
exact_result <- ANOVA_exact(design_result,
                            alpha_level = alpha_level,
                            verbose = FALSE)
```

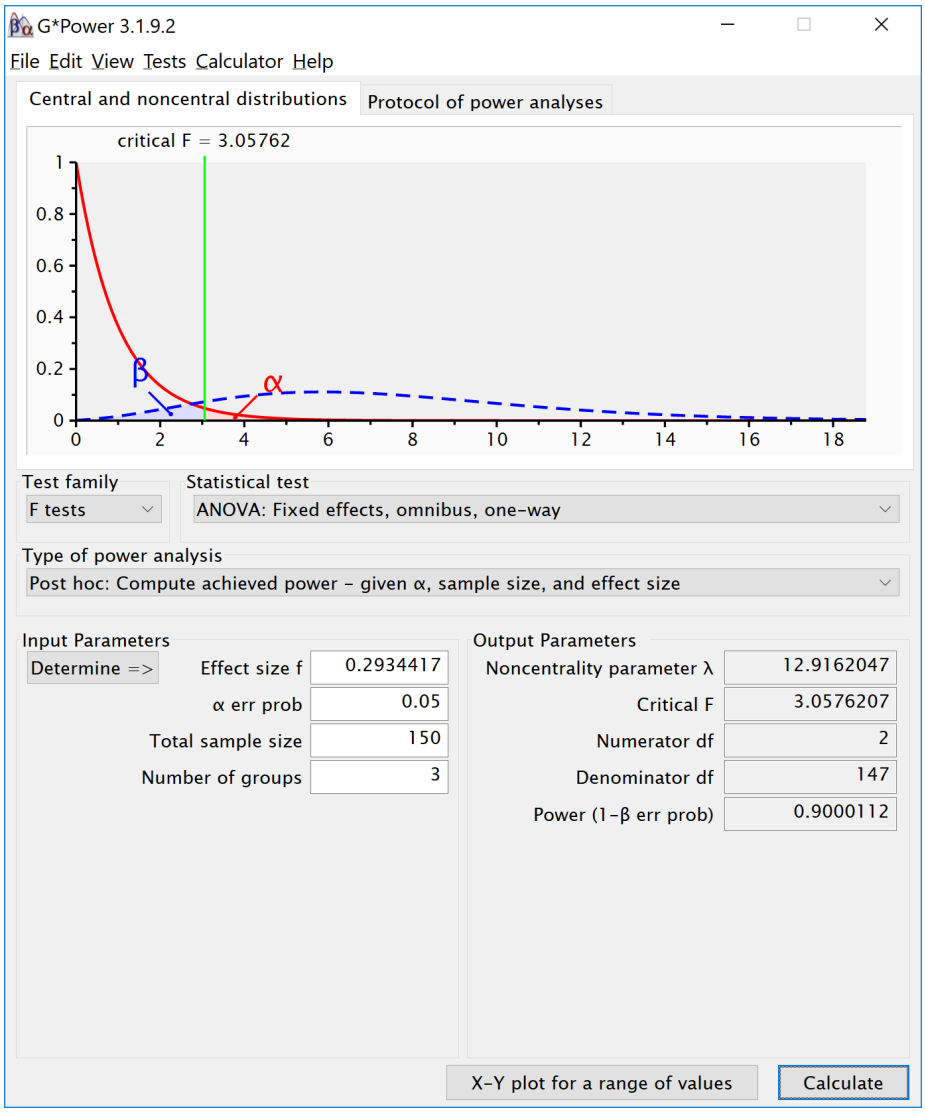
Table 2.9: Simulated ANOVA Result

	power	effect_size
anova_condition	90.3	0.090741

Table 2.10: Exact ANOVA Result

	power	partial_eta_squared	cohen_f	non_centrality
condition	90.00112	0.0807686	0.2964209	12.9162

The simulation confirms that for the F-test for the ANOVA we have 90% power. This is also what g*power tells us what would happen based on a post-hoc power analysis with an f of 0.2934417, 3 groups, 150 participants in total (50 in each between subject condition), and an alpha of 5%.



We can also compute the power for the ANOVA and simple effects in `R` with the `pwr` package. The calculated effect sizes and power match those from the simulation.

```
K <- 3
n <- 50
sd <- 6.4
f <- 0.2934417
pwr.anova.test(
n = n,
```

```
k = K,
f = f,
sig.level = alpha_level
)$power
```

```
## [1] 0.9000112
```

```
d <- 2.300104 / 6.4
d
```

```
## [1] 0.3593912
```

```
pwr.t.test(
d = 2.300104 / 6.4,
n = 50,
sig.level = 0.05,
type = "two.sample",
alternative = "two.sided"
)$power
```

```
## [1] 0.4284243
```

```
d <- 2 * 2.300104 / 6.4
d
```

```
## [1] 0.7187825
```

```
pwr.t.test(
d = d,
n = 50,
sig.level = 0.05,
type = "two.sample",
alternative = "two.sided"
)$power
```

```
## [1] 0.9450353
```

We can also compare the results against the analytic solution by Aberson (2019). First, load the function for a 3-way ANOVA from the `pwr2ppl` package. Then we use the function to calculate power.

```
#Initial example, low power
anova1f_3(
m1 = 24,
m2 = 26.2,
m3 = 26.6,
s1 = 6.4,
s2 = 6.4,
s3 = 6.4,
n1 = 50,
n2 = 50,
n3 = 50,
alpha = .05
)
```

```
## Sample size overall = 150
```

```
## Power  = 0.4769 for eta-squared = 0.0315
```

```
#From: Aberson, Christopher L.
#   Applied Power Analysis for the Behavioral Sciences, 2nd Edition.
# $Power [1] 0.4769468
#Later example, based on larger mean difference
anova1f_3(
m1 = -2.300104,
m2 = 0,
m3 = 2.300104,
s1 = 6.4,
s2 = 6.4,
s3 = 6.4,
n1 = 50,
n2 = 50,
n3 = 50,
alpha = .05
)
```

```
## Sample size overall = 150
```

```
## Power  = 0.9 for eta-squared = 0.0808
```

```
# $Power [1] 0.9000112
```

2.2 Effect Size Estimates for One-Way ANOVA

Using the formulas below, we can calculate the means for a one-way ANOVA. Using the formula from Albers and Lakens (2018), we can determine the means that should yield a specified effect sizes (expressed in Cohen's f).

Eta-squared (idential to partial eta-squared for one-way ANOVA's) has benchmarks of .0099, .0588, and .1379 for small, medium, and large effect sizes (Cohen, 1988).

2.2.1 Three conditions, small effect size

We can simulate a one-factor anova setting means to achieve a certain effect size. Eta-squared is biased. Thus, the eta-squared we calculate based on the observed data overestimates the population effect size. This bias is largest for smaller sample sizes. Thus, to test whether the simulation yields the expected effect size, we use extremele large sample sizes in each between subject condition (n = 5000). This simulation should yield a small effect size (0.099)

```
K <- 3
ES <- .0099
mu <- mu_from_ES(K = K, ES = ES)
n <- 5000
sd <- 1
r <- 0
string = paste(K,"b",sep = "")
```

```
design_result <- ANOVA_design(
  design = string,
  n = n,
  mu = mu,
  sd = sd,
  r = r,
  labelnames = c("factor1", "level1", "level2", "level3")
  )
```

```
simulation_result <- ANOVA_power(design_result,
                                 alpha_level = alpha_level,
                                 nsims = nsims,
                                 verbose = FALSE)
```

```
exact_result <- ANOVA_exact(design_result,
                            alpha_level = alpha_level,
                            verbose = FALSE)
```

Table 2.11: Simulated ANOVA Result

	power	effect_size
anova_factor1	100	0.0100165

Table 2.12: Exact ANOVA Result

	power	partial_eta_squared	cohen_f	non_centrality
factor1	100	0.009902	0.100005	149.9848

The resulting effect size estimate from the simulation is very close to 0.0099

2.2.2 Four conditions, medium effect size

This simulation should yield a medium effect size (0.588) across four independent conditions.

```
K <- 4
ES <- .0588
mu <- mu_from_ES(K = K, ES = ES)
n <- 5000
sd <- 1
r <- 0
string = paste(K,"b",sep = "")
```

```
design_result <- ANOVA_design(
  design = string,
  n = n,
  mu = mu,
  sd = sd,
  r = r,
  labelnames = c("factor1", "level1", "level2", "level3", "level4")
  )
```

```
simulation_result <- ANOVA_power(design_result,
                                 alpha_level = alpha_level,
                                 nsims = nsims,
                                 verbose = FALSE)
```

```
exact_result <- ANOVA_exact(design_result,
                            alpha_level = alpha_level,
                            verbose = FALSE)
```

Table 2.13: Simulated ANOVA Result

	power	effect_size
anova_factor1	100	0.058913

Table 2.14: Exact ANOVA Result

	power	partial_eta_squared	cohen_f	non_centrality
factor1	100	0.0588111	0.2499719	1249.469

Results are very close to 0.588.

2.2.3 Two conditions, large effect size

We can simulate a one-way ANOVA that should yield a large effect size (0.1379) across two conditions.

```
K <- 2
ES <- .1379
mu <- mu_from_ES(K = K, ES = ES)
n <- 5000
sd <- 1
r <- 0
string = paste(K,"b",sep = "")
```

```
design_result <- ANOVA_design(design = string,
                    n = n,
                    mu = mu,
                    sd = sd,
                    r = r,
                    labelnames = c("factor1", "level1", "level2"))
```

```
simulation_result <- ANOVA_power(design_result,
                                 alpha_level = alpha_level,
                                 nsims = nsims,
                                 verbose = FALSE)
```

Table 2.15: Simulated ANOVA Result

	power	effect_size
anova_factor1	100	0.1380489

```
exact_result <- ANOVA_exact(design_result,
                            alpha_level = alpha_level,
                            verbose = FALSE)
```

Table 2.16: Exact ANOVA Result

	power	partial_eta_squared	cohen_f	non_centrality
factor1	100	0.1379238	0.3999878	1599.582

The results are very close to is simulation should yield a small effect size (0.1379).

Chapter 3

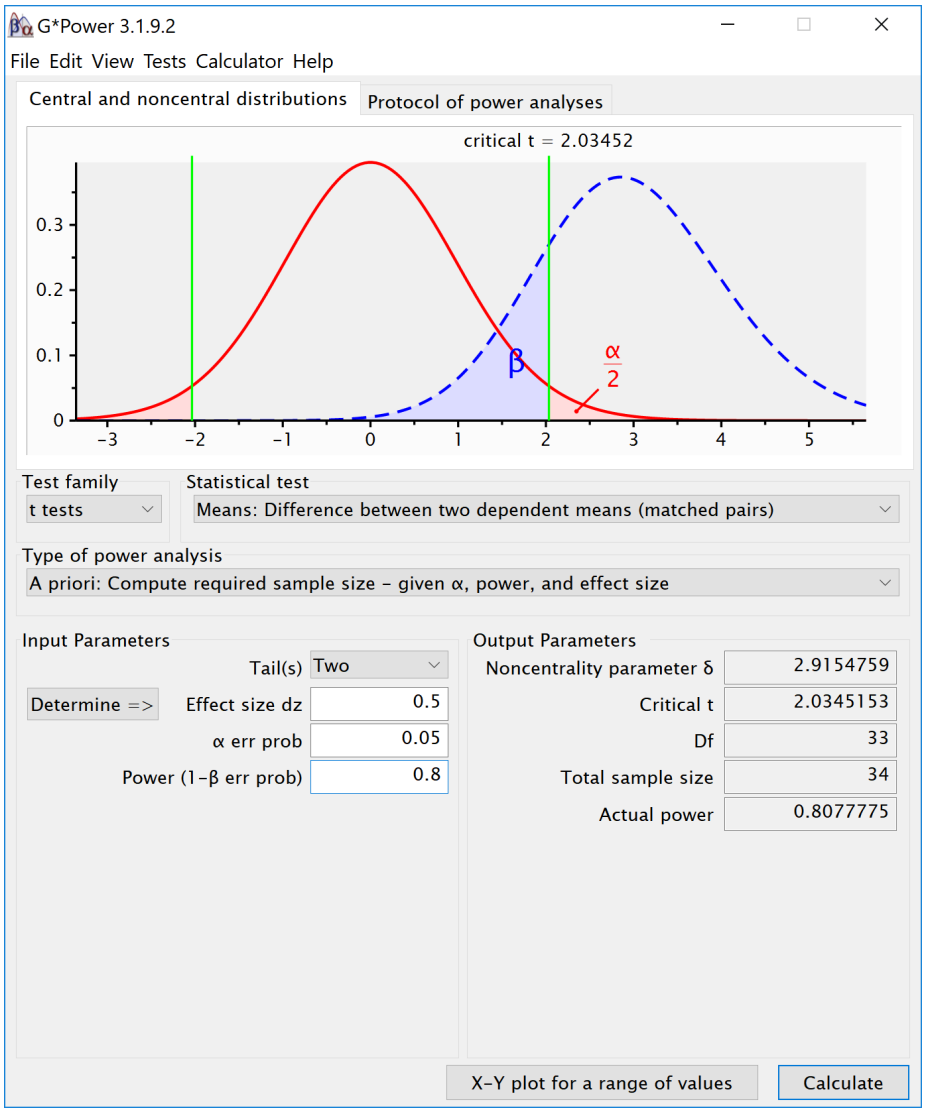
Repeated Measures ANOVA

3.1 Part 1

In a repeated measures design multiple observations are collected from the same participants. In the simplest case, where there are two repeated observations, a repeated measures ANOVA equals a dependent or paired t-test. The advantage of repeated measures designs is that they capitalize on the correlations between the repeated measurements. Let's first explore the impact of this correlation on the power of a repeated measures ANOVA.

3.1.1 Two conditions, medium effect size

To illustrate the effect of correated observations, we start by simulating data for a medium effect size for a dependent (or paired, or within-subject) t-test. Let's first look at Gpower Faul et al. (2007). If we want to perform an a-priori power analysis, we are asked to fill in the effect size dz. As Cohen (1988) writes, "The Z subscript is used to emphasize the fact that our raw score unit is no longer X or Y, but Z", where Z are the difference scores of X-Y.



Within designs can have greater power to detect differences than between designs because the values are correlated, and a within design requires less participants because each participant provides multiple observations. One difference between an independent *t*-test and a dependent *t*-test is that an independent *t*-test has 2(n-1) degrees of freedom, while a dependent *t*-test has (n-1) degrees of freedom. The sample size needed in a two-group within-design (NW) relative to the sample needed in two-group between-designs (NB), assuming normal distributions, and ignoring the difference in degrees of freedom between the two types of tests, is (from Maxwell et al. (2004), p. 561, formula 45):

$$N_W = \frac{N_B(1-\rho)}{2}$$

The division by 2 in the equation is due to the fact that in a two-condition within design every participant provides two data-points. The extent to which this reduces the sample size compared to a between-subject design depends on the correlation (r) between the two dependent variables, as indicated by the `1-r` part of the equation. If the correlation is 0, a within-subject design needs half as many participants as a between-subject design (e.g., 64 instead 128 participants), simply because every participants provides 2 datapoints. The higher the correlation, the larger the relative benefit of within designs, and whenever the correlation is negative (up to -1) the relative benefit disappears.

Whereas in an independent t-test the two observations are uncorrelated, in a within design the observations are correlated. This has an effect on the standard deviation of the difference scores. In turn, because the standardized effect size is the mean difference divided by the standard deviation of the difference scores, the correlation has an effect on the standardized mean difference in a within design, Cohen's dz. The relation, as Cohen (1988, formula 2.3.7) explains, is:

$\sigma_z = \sigma\sqrt{2(1-\rho)}$

Therefore, the relation between dz and d is $\sqrt{2(1-\rho)}$. A given difference between population means for matched (dependent) samples is standardized by a value which is $\sqrt{2(1-\rho)}$ as large as would be the case were they independent. If we enter a correlation of 0.5 in the formula, we get $\sqrt{2(0.5)} = 1$. In other words, when the correlation is 0.5, d = dz. When there is a strong correlation between dependent variables, for example r = 0.9, we get $d = d_z\sqrt{2(1-0.9)}$, and a dz of 1 would be a d = 0.45. Reversely, $d_z = \frac{d}{\sqrt{2(1-r)}}$, so with a r = 0.9, a d of 1 would be a dz = 2.24. Some consider this increase in dz compared to d when observations are strongly correlated an 'inflation' when estimating effect sizes, but since the reduction in the standard deviation of the difference scores due to the correlation makes it easier to distinguish signal from noise in a hypothesis test, it leads to a clear power benefit.

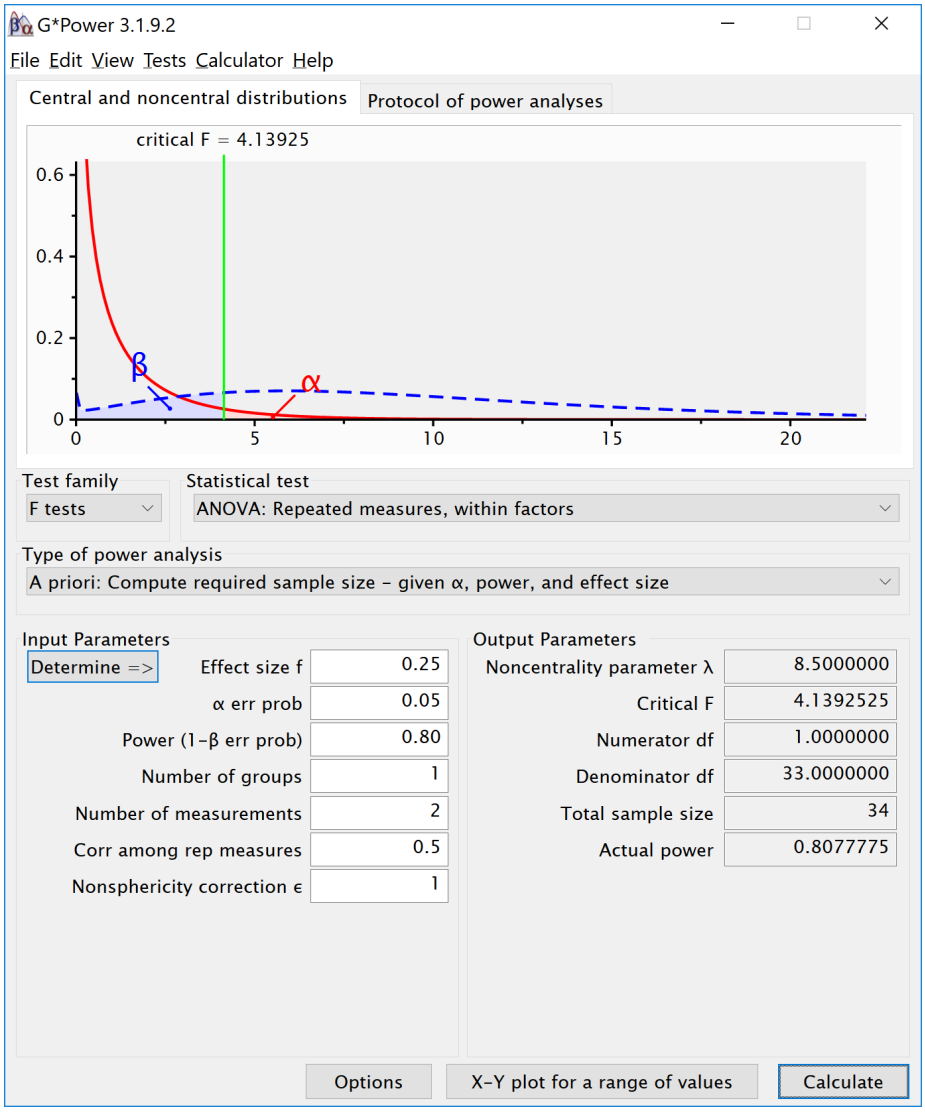
```r
# Check sample size formula Maxwell
# Power is pretty similar with n/2, same d (assuming r = 0.5).
# Small differences due to df = 2(n-1) vs df = n-1
pwr.t.test(d = 0.05,
           n = c(2000, 4000, 8000),
           sig.level = 0.05,
           type = "two.sample",
           alternative = "two.sided")
```

```
##
##      Two-sample t test power calculation
##
##               n = 2000, 4000, 8000
##               d = 0.05
##       sig.level = 0.05
##           power = 0.3524674, 0.6086764, 0.8853424
##     alternative = two.sided
##
## NOTE: n is number in *each* group
```

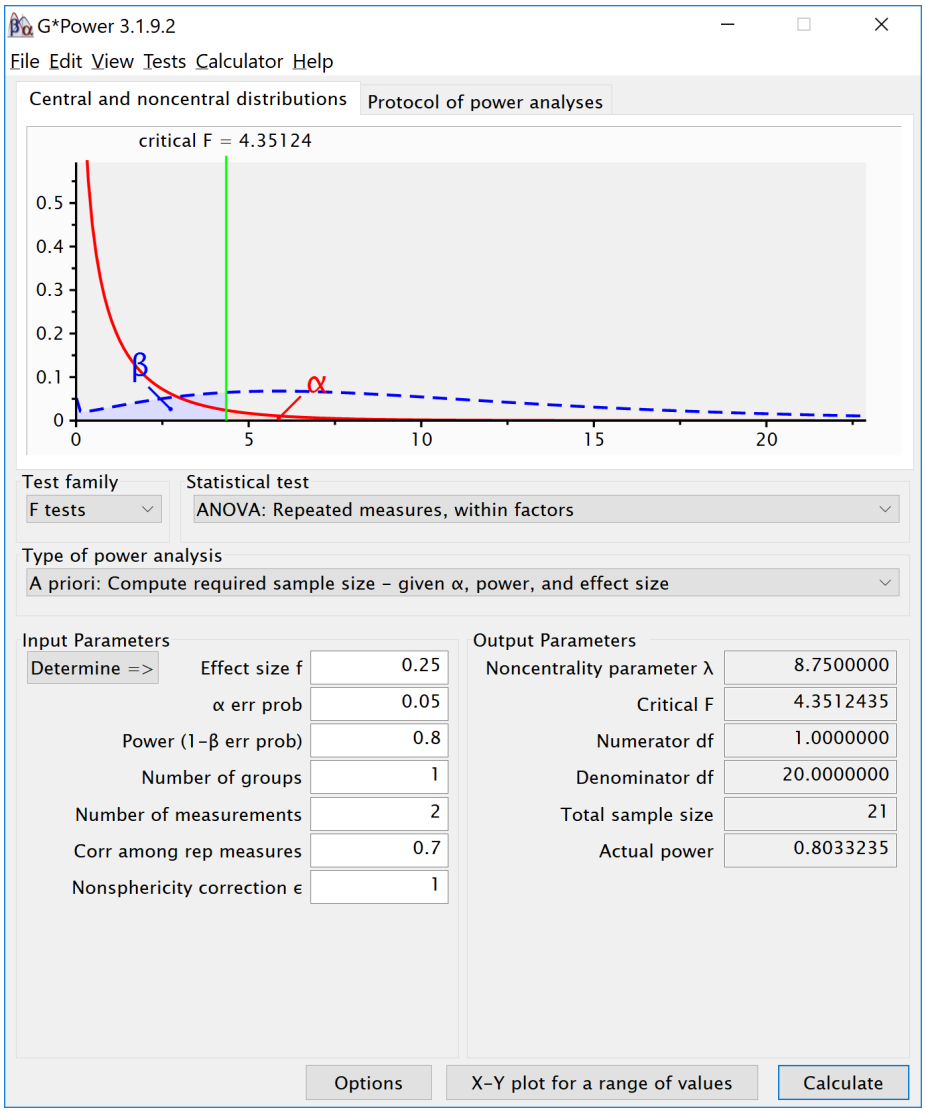
```r
pwr.t.test(d = 0.05,
           n = c(1000, 2000, 4000),
           sig.level = 0.05,
           type = "paired",
           alternative = "two.sided")
```

```
##
##      Paired t test power calculation
##
##               n = 1000, 2000, 4000
##               d = 0.05
##       sig.level = 0.05
##           power = 0.3520450, 0.6083669, 0.8852320
##     alternative = two.sided
##
## NOTE: n is number of *pairs*
```

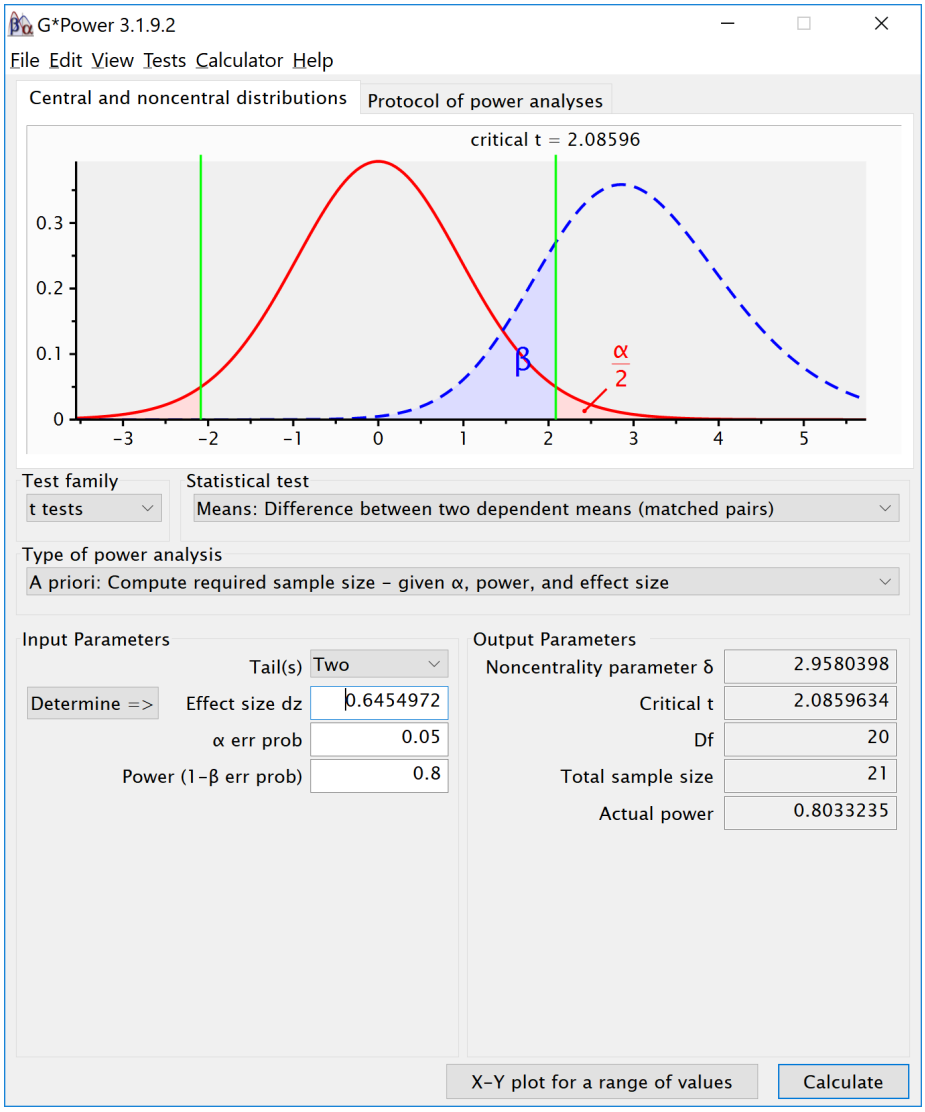
There is no equivalent "fz" for Cohen's f for a within subject ANOVA. For two groups, we can directly compute Cohen's f from Cohen's d for two groups, as Cohen (1988) describes, because $f = 1/2d$. For a $d = 0.5$, $f = 0.25$. In Gpower we can run a 2 group within-subject power analysis for ANOVA. We plan for 80% power, and reproduce the anaysis above for the dependent t-test. This works because the correlation is set to 0.5, when d = dz, and thus the transformation of f=1/2d works.



If we change the correlation to 0.7 and keep all other settings the same, the repeated measure a-priori power analysis yields a sample of 21. The correlation increases the power for the test.



To reproduce this analysis in Gpower with a dependent t-test we need to change dz following the formula above, $d_z = \frac{0.5}{\sqrt{2(1-0.7)}}$, which yields dz = 0.6454972. If we enter this value in Gpower for an a-priori power analysis, we get the exact same results (as we should, since an repeated measures ANOVA with 2 groups equals a dependent t-test). This example illustrates that the correlation between dependent variables always factors into a power analysis, both for a dependent t-test, and for a repeated measures ANOVA. Because a dependent t-test uses dz the correlation might be less visible, but given the relation between d and dz, the correlation is always taken into account and can greatly improve power for within designs compared to between designs.



We can perform both these power analyses using `Superpower` as well. We set groups to 2 for the simulation, n = 34 (which should give 80.777 power, according to the Faul et al. (2007) program), a correlation among repeated measures of 0.5, and an alpha of 0.05. In this case, we simulate data with means -0.25 and 0.25, and set the sd to 1. This means we have a mean difference of 0.5, and a Cohen's d of $0.5/1 = 0.5$. In the first example, we set the correlation to 0.5, and the result should be 80.77% power, and an effect size estimate of 0.5 for the simple effect. We also calculate partial eta-squared for the ANOVA, which equals $\frac{f^2}{f^2+1}$, or 0.05882353.

```
K <- 2
n <- 34
sd <- 1
r <- 0.5
alpha = 0.05
f <- 0.25
f2 <- f^2
ES <- f2/(f2 + 1)
ES
```

```
## [1] 0.05882353
```

```
mu <- mu_from_ES(K = K, ES = ES)
design = paste(K,"w",sep = "")
labelnames <- c("speed", "fast", "slow")
design_result <- ANOVA_design(design = design,
                  n = n, mu = mu, sd = sd, r = r,
                  labelnames = labelnames)
alpha_level <- 0.05
```

```
simulation_result <- ANOVA_power(design_result,
                                 alpha_level = alpha_level,
                                 nsims = nsims, verbose = FALSE)
```

Table 3.1: Simulated ANOVA Result

	power	effect_size
anova_speed	80.74	0.2149221

Table 3.2: Exact ANOVA Result

	power	partial_eta_squared	cohen_f	non_centrality
speed	80.77775	0.2048193	0.5075192	8.5

The results of the simulation are indeed very close to 80.777%. Note that the simulation calculates Cohen's dz effect sizes for paired comparisons - which here given the correlation of 0.5 is also 0.5 for a medium effect size.

We should see a larger dz if we increase the correlation, keeping the sample size the same, following the example in Gpower above. We repeat the simulation, and the only difference is a correlation between dependent variables of 0.7. This should yield an effect size dz = 0.6454972.

```
K <- 2
n <- 21
sd <- 1
r <- 0.7
alpha = 0.05
f <- 0.25
f2 <- f^2
ES <- f2/(f2 + 1)
ES
```

```
## [1] 0.05882353
```

```
mu <- mu_from_ES(K = K, ES = ES)
design = paste(K,"w",sep = "")
labelnames <- c("speed", "fast", "slow")
design_result <- ANOVA_design(design = design,
                   n = n, mu = mu, sd = sd, r = r,
                   labelnames = labelnames)
alpha_level <- 0.05
```

Table 3.3: Covariance-Variance Matrix

	fast	slow
fast	1.0	0.7
slow	0.7	1.0

```
simulation_result <- ANOVA_power(design_result,
                                 alpha_level = alpha_level,
                                 nsims = nsims,
                                 verbose = FALSE)
```

Table 3.4: Simulated ANOVA Result

	power	effect_size
anova_speed	80.48	0.3153414

```
exact_result <- ANOVA_exact(design_result,
                            alpha_level = alpha_level,
                            verbose = FALSE)
```

Table 3.5: Exact ANOVA Result

	power	partial_eta_squared	cohen_f	non_centrality
speed	80.33235	0.3043478	0.6614378	8.75

```
#relation dz and f for within designs
f <- 0.5*0.6454972
f
```

```
## [1] 0.3227486
```

Entering this f in Gpower, with a correlation of 0.5, yields the same as entering `f = 0.25` and `correlation = 0.7`.

3.2 Part 2

Here, we will examine a repeated measures experiment with 3 within-subject conditions, to illustrate how a repeated measures ANOVA extends a dependent *t*-test with 3 groups.

In the example for a two-group within design we provided a specific formula for the sample size benefit for two groups. The sample size needed in within-designs (NW) with more than 2 conditions, relative to the sample needed in between-designs (NB), assuming normal distributions and compound symmetry, and ignoring the difference in degrees of freedom between the two types of tests, is (from Maxwell et al. (2004), p. 562, formula 47):

$N_W = \frac{N_B(1-\rho)}{a}$

Where "a" is the number of within-subject levels.

3.2.1 The relation between Cohen's f and Cohen's d

Whereas in the case of a repeated measures ANOVA with 2 groups we could explain the principles of a power analysis by comparing our test against a *t*-test and Cohen's d, this becomes more difficult when we have more than 2 groups. It is more useful to explain how to directly calculate Cohen's *f*, the effect size used in power analyses for ANOVA. Cohen's *f* is calculated following Cohen (1988), formula 8.2.1 and 8.2.2:

$f = \sqrt{\frac{\frac{\sum(\mu-\overline{\mu})^2)}{N}}{\sigma}}$

Imagine we have a within-subject experiment with 3 conditions. We ask people what they mood is when their alarm clock wakes them up, when they wake up naturally on a week day, and when they wake up naturally on a weekend day. Based on pilot data, we expect the means (on a 7 point validated mood scale) are 3.8, 4.2, and 4.3. The standard deviation is 0.9, and the correlation between the dependent measurements is 0.7. We can calculate Cohen's *f* for the ANOVA, and Cohen's dz for the contrasts:

```
mu <- c(3.8, 4.2, 4.3)
sd <- 0.9
f <- sqrt(sum((mu - mean(mu)) ^ 2) / length(mu)) / sd
#Cohen, 1988, formula 8.2.1 and 8.2.2
  f
```

```
## [1] 0.2400274
```

```
  r <- 0.7
  (4.2 - 3.8) / 0.9 / sqrt(2 * (1 - r))
```

```
## [1] 0.5737753
```

```
  (4.3 - 3.8) / 0.9 / sqrt(2 * (1 - r))
```

```
## [1] 0.7172191
```

```
  (4.3 - 4.2) / 0.9 / sqrt(2 * (1 - r))
```

```
## [1] 0.1434438
```

The relation between Cohen's d or dz and Cohen's f becomes more difficult when there are multiple groups, because the relationship depends on the pattern of the means. Cohen (1988) presents calculations for three patterns, minimal variability (for example, for 5 means: -0.25, 0, 0, 0, 0.25), medium variability (for example, for 5 means: -0.25, -0.25, 0.25, 0.25, 0.25 or -0.25, -0.25, -0.25, 0.25, 0.25). For these three patterns, formula's are available that compute Cohen's f from Cohen's d, where d is the effect size calculated for the difference between the largest and smallest mean (if the largest mean is 0.25 and the smallest mean is -0.25, 0.25 - -0.25 = 0.5, so d is 0.5 divided by the standard deviation of 0.9). In our example, d would be (4.3-3.8)/0.9 = 0.5555556. If we divide this value by `sqrt(2*(1-r))` we have dz = 0.5555556/0.7745967 = 0.7172191.

We have created a custom function that will calculate f from d, based on a specification of one of the three patterns of means. Our pattern is most similar (but not identical) to a maximum variability pattern (two means are high, one is lower). So we could attempt to calculate f from d (0.5555556), by calculating d from the largest and smallest mean.

This function allows you to calculate f, d and eta squared following Cohen (1988), p 277. The patterns are: 1. Minimum variability: one mean at each end of d, the remaining k- 2 means all at the midpoint. 2. Intermediate variability: the k means equally spaced over d. 3. Maximum variability: the means all at the end points of d.

For each of these patterns, there is a fixed relationship between f and d for any given number of means, k.

Pattern 1 For any given range of means, d, the minimum standard deviation, f1, results when the remaining k - 2 means are concentrated at the mean of the means (0 when expressed in standard units), i.e., half-way between the largest and smallest.

Pattern 2 A pattern of medium variability results when the k meansare equally spaced over the range, and therefore at intervals of d/(k- 1).

Pattern 3 It is demonstrable and intuitively evident that for any given range the dispersion which yield~ the maximum standard deviation has the k means falling at both extremes of the range. When k is even, !k fall at - !d and the other !k fall at + !d; when k is odd, (k + I)/2 of the means fall at either end and the (k- 1)/2 remaining means at the other. With this pattern, for all even numbers of means, use formula (8.2.12). When k is odd, and there is thus one more mean at one extreme than at the other, use formula (8.2.13).

```
calc_f_d_eta <- function(mu, sd, variability){
  if (variability == "minimum") {
    k = length(mu)
    d <- (max(mu) - min(mu)) / sd
    f <- d * sqrt(1 / (2 * k))
    f2 <- f ^ 2
    ES <- f2 / (f2 + 1)
  }
  if (variability == "medium") {
    k = length(mu)
    d <- (max(mu) - min(mu)) / sd
    f <- (d / 2) * sqrt((k + 1) / (3 * (k - 1)))
    f2 <- f ^ 2
    ES <- f2 / (f2 + 1)
  }
  if (variability == "maximum") {
    k = length(mu)
    d <- (max(mu) - min(mu)) / sd
    f <- ifelse(k %% 2 == 0, .5 * d, d * (sqrt(k ^ 2 - 1) / (2 * k)))
    f2 <- f ^ 2
    ES <- f2 / (f2 + 1)
  }
  invisible(list(mu = mu,
                 sd = sd,
                 d = d,
                 f = f,
                 f2 = f2,
                 ES = ES))
}
res <- calc_f_d_eta(mu = mu, sd = sd, variability = "maximum")
res$f
```

```
## [1] 0.2618914
```

```
res$d
```

```
## [1] 0.5555556
```

We see the Cohen's f value is 0.2618914 and d = 0.5555556. The Cohen's f is not perfectly accurate - it is assuming the pattern of means is 3.8, 4.3, 4.3, and not 3.8, 4.2, 4.3. If the means and sd is known, it is best to calculate Cohen's f directly from these values.

3.2.2 Three within conditions, medium effect size

We can perform power analyses for within designs using simuations. We set groups to 3 for the simulation, n = 20, and the correlation between dependent variables to 0.8. If the true effect size is f = 0.25, and the alpha level is 0.05, the power is 96.6%. In this case, we simulate data with means -0.3061862, 0.0000000, and 0.3061862, and set the sd to 1.

```
K <- 3
n <- 20
sd <- 1
r <- 0.8
alpha = 0.05
f <- 0.25
f2 <- f^2
ES <- f2 / (f2 + 1)
ES
```

```
## [1] 0.05882353
```

```
mu <- mu_from_ES(K = K, ES = ES)

#Cohen, 1988, formula 8.2.1 and 8.2.2
sqrt(sum((mu - mean(mu)) ^ 2) / length(mu)) / sd
```

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

```
design = paste(K, "w", sep = "")
labelnames <- c("speed", "fast", "medium", "slow")
design_result <- ANOVA_design(design = design,
                    n = n,
                    mu = mu,
                    sd = sd,
```

```
                    r = r,
                    labelnames = labelnames)

alpha_level <- 0.05
```

```
simulation_result <- ANOVA_power(design_result,
                                 alpha_level = alpha_level,
                                 nsims = nsims,
                                 verbose = FALSE)
```

Table 3.6: Simulated ANOVA Result

	power	effect_size
anova_speed	97.04	0.345486

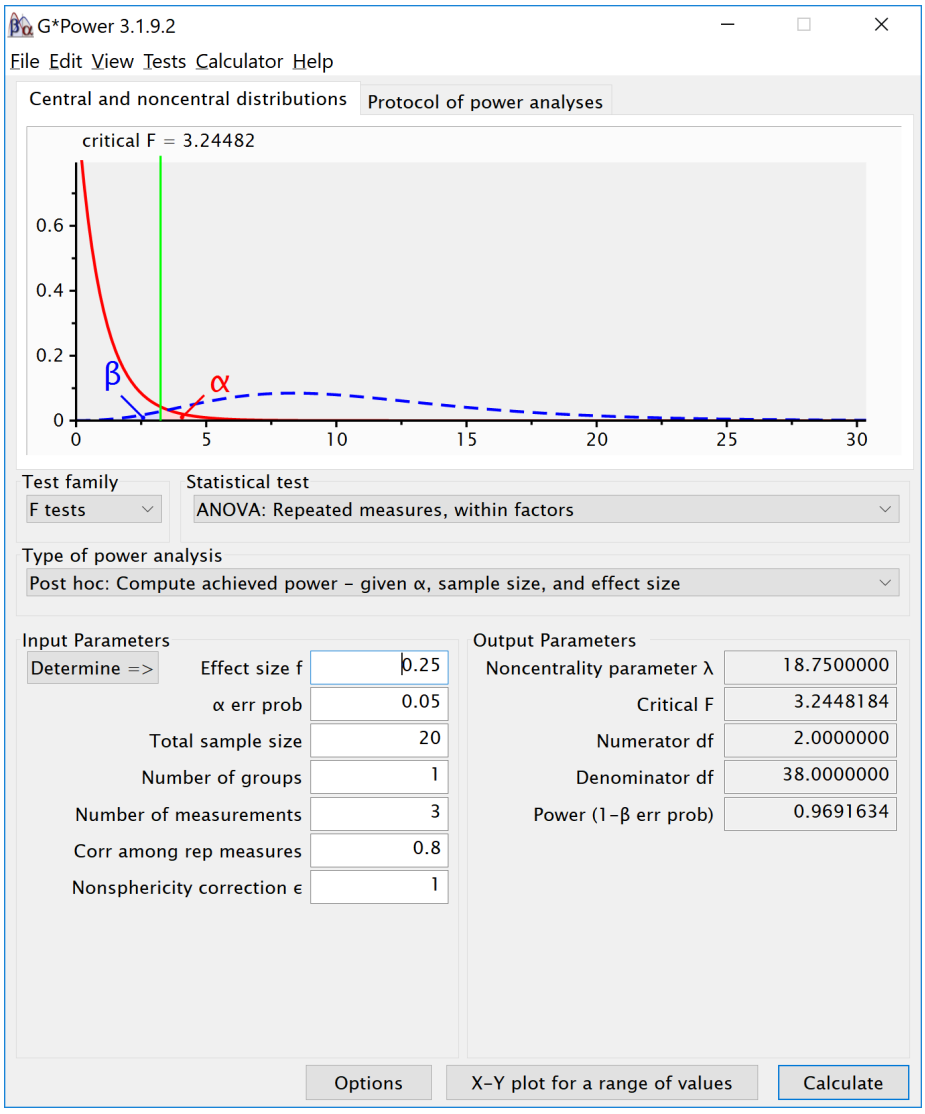
```
exact_result <- ANOVA_exact(design_result,
                            alpha_level = alpha_level,
                            verbose = FALSE)
```

Table 3.7: Exact ANOVA Result

	power	partial_eta_squared	cohen_f	non_centrality
speed	96.91634	0.3303965	0.7024394	18.75

The results of the simulation are indeed very close to 96.9%.

We can see this is in line with the power estimate from Gpower:



We can also validate this by creating the code to do a power analysis in R from scratch:

```
K <- 3 #three groups
n <- 20
sd <- 1
r <- 0.8
alpha = 0.05
f <- 0.25
f2 <- f^2
ES <- f2 / (f2 + 1)
ES
```

```
## [1] 0.05882353
```

```
mu <- mu_from_ES(K = K, ES = ES)
design = paste(K, "w", sep = "")
labelnames <- c("speed", "fast", "medium", "slow")
design_result <- ANOVA_design(design = design,
                   n = n,
                   mu = mu,
                   sd = sd,
                   r = r,
                   labelnames = labelnames)

power_oneway_within(design_result)$power
```

```
## [1] 96.91634
```

```
power_oneway_within(design_result)$eta_p_2
```

```
## [1] 0.05882353
```

```
power_oneway_within(design_result)$eta_p_2_SPSS
```

```
## [1] 0.3303965
```

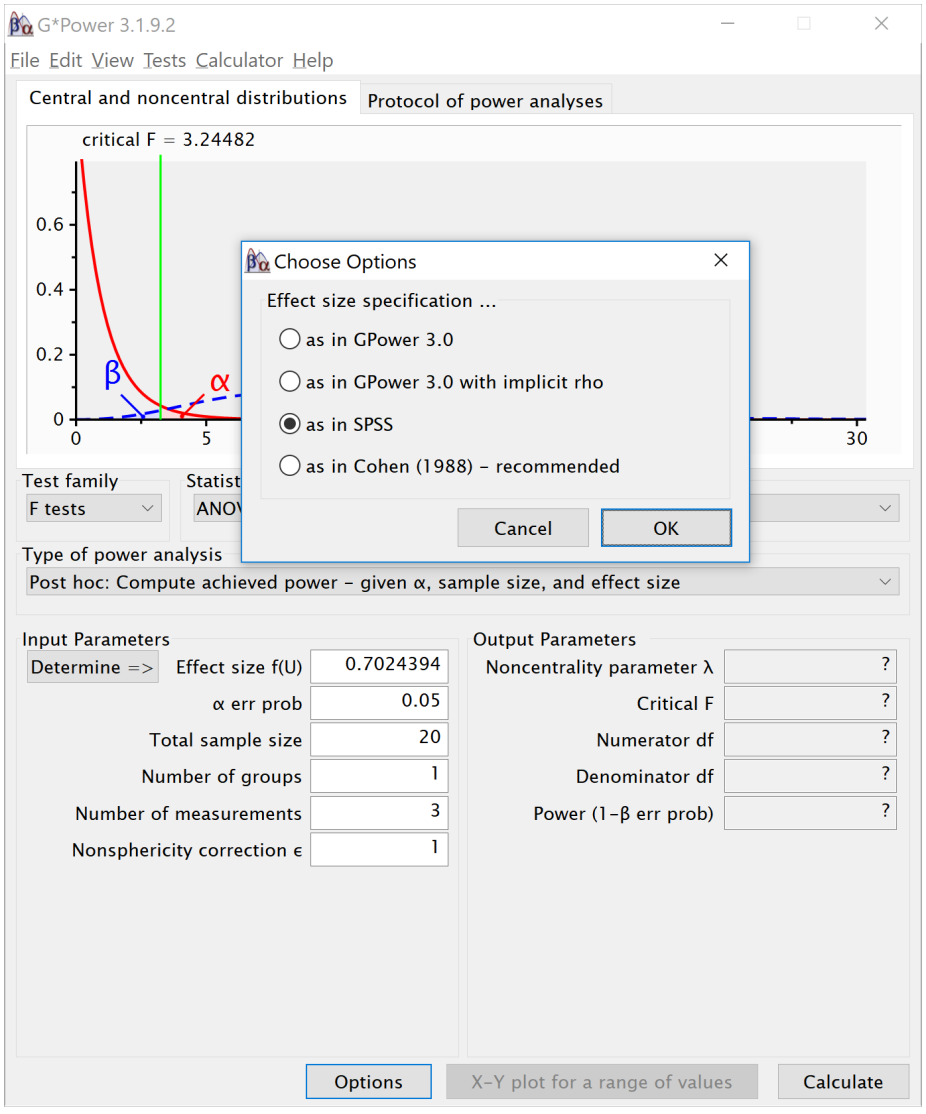
```
power_oneway_within(design_result)$Cohen_f
```

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

```
power_oneway_within(design_result)$Cohen_f_SPSS
```

```
## [1] 0.7024394
```

We can even check the calculation of Cohen's f SPSS style in GPower. We take the GPower settings as illustrated above. We click the 'Options' button, and check the radiobutton next to 'As in SPSS'. Click ok, and you will notice that the 'Corr among rep measures' field has disappeared. The correlation does not need to be entered seperately, but is incorporated in Cohen's f. The value of Cohen's f, which was 0.25, has changed into 0.7024394. This is the SPSS equivalent. The value is much larger. This value, and it's corresponding partial eta-squared, incorporate the correlation between observations.



3.3 Part 3

We first repeat the simulation by Brysbaert:

```
# give sample size
N = 75
# give effect size d
d1 = .4 #difference between the extremes
d2 = .4 #third condition goes with the highest extreme
# give the correlation between the conditions
r = .5
# give number of simulations
nSim = nsims
# give alpha levels
alpha1 = .05 #alpha level for the omnibus ANOVA
alpha2 = .05 #also adjusted from original by DL
```

```
# create vectors to store p-values
p1 <- numeric(nSim) #p-value omnibus ANOVA
p2 <- numeric(nSim) #p-value first post hoc test
p3 <- numeric(nSim) #p-value second post hoc test
p4 <- numeric(nSim) #p-value third post hoc test

# define correlation matrix
rho <- cbind(c(1, r, r), c(r, 1, r), c(r, r, 1))
# define participant codes
part <- paste("part",seq(1:N))
for (i in 1:nSim) {
  #for each simulated experiment

  data = mvrnorm(n = N,
  mu = c(0, 0, 0),
  Sigma = rho)
  data[, 2] = data[, 2] + d1
  data[, 3] = data[, 3] + d2
  datalong = c(data[, 1], data[, 2], data[, 3])
  conds = factor(rep(letters[24:26], each = N))
  partID = factor(rep(part, times = 3))
  output <- data.frame(partID, conds, datalong)
  test <- aov(datalong ~ conds + Error(partID / conds), data = output)
  tests <- (summary(test))
  p1[i] <- tests$'Error: partID:conds'[[1]]$'Pr(>F)'[[1]]
  p2[i] <- t.test(data[, 1], data[, 2], paired = TRUE)$p.value
  p3[i] <- t.test(data[, 1], data[, 3], paired = TRUE)$p.value
  p4[i] <- t.test(data[, 2], data[, 3], paired = TRUE)$p.value
```

```
}
```

```
#printing all unique tests (adjusted code by DL)
sum(p1 < alpha1) / nSim
sum(p2 < alpha2) / nSim
sum(p3 < alpha2) / nSim
sum(p4 < alpha2) / nSim
```

```
## [1] 0.7961
```

```
## [1] 0.7624
```

```
## [1] 0.7574
```

```
## [1] 0.0452
```

3.4 Part 4

3.4.1 2x2 ANOVA, within-within design

We can simulate a 2x2 ANOVA, both factors manipulated within participants, with a specific sample size and effect size, to achieve a desired statistical power.

As Potvin and Schutz (2000) explain, analytic procedures for a two-factor repeated measures ANOVA do not seem to exist. The main problem is quantifying the error variance (the denominator when calculating lambda or Cohen's f). Simulation based aproaches provide a solution.

We can reproduce the simulation coded by Ben Amsel

```
# define the parameters
# true effects (in this case, a double dissociation)
mu = c(700, 670, 670, 700)
sigma = 150  # population standard deviation
rho = 0.75 # correlation between repeated measures
nsubs = 25 # how many subjects?
nsims = nsims # how many simulation replicates?

# create 2 factors representing the 2 independent variables
cond = data.frame(X1 = rep(factor(letters[1:2]), nsubs * 2),
                  X2 = rep(factor(letters[1:2]), nsubs, each = 2))

# create a subjects factor
subject = factor(sort(rep(1:nsubs, 4)))

# combine above into the design matrix
dm = data.frame(subject, cond)
```

Build Sigma: the population variance-covariance matrix

```
# create k x k matrix populated with sigma
sigma.mat <- rep(sigma, 4)
S <-
  matrix(sigma.mat,
  ncol = length(sigma.mat),
  nrow = length(sigma.mat))

# compute covariance between measures
Sigma <- t(S) * S * rho

# put the variances on the diagonal
diag(Sigma) <- sigma^2
```

Run the simulation

```
# stack 'nsims' individual data frames into one large data frame
df = dm[rep(seq_len(nrow(dm)), nsims), ]

# add an index column to track the simulation run
df$simID = sort(rep(seq_len(nsims), nrow(dm)))

# sample the observed data from a multivariate normal distribution
# using MASS::mvrnorm with the mu and Sigma created earlier
# and bind to the existing df


make.y = expression(as.vector(t(mvrnorm(nsubs, mu, Sigma))))
df$y = as.vector(replicate(nsims, eval(make.y)))

# use do(), the general purpose complement to the specialized data
# manipulation functions available in dplyr, to run the ANOVA on
# each section of the grouped data frame created by group_by


mods <- df %>%
  group_by(simID) %>%
  do(model = aov(y ~ X1 * X2 + Error(subject / (X1 * X2)),
                 qr = FALSE, data = .))

# extract p-values for each effect and store in a data frame
p_val_1 = data.frame(
  mods %>% do(as.data.frame(tidy(.$model[[3]]))$p.value[1])),
  mods %>% do(as.data.frame(tidy(.$model[[4]]))$p.value[1])),
  mods %>% do(as.data.frame(tidy(.$model[[5]]))$p.value[1])))
colnames(p_val_1) = c('X1','X2','Interaction')
```

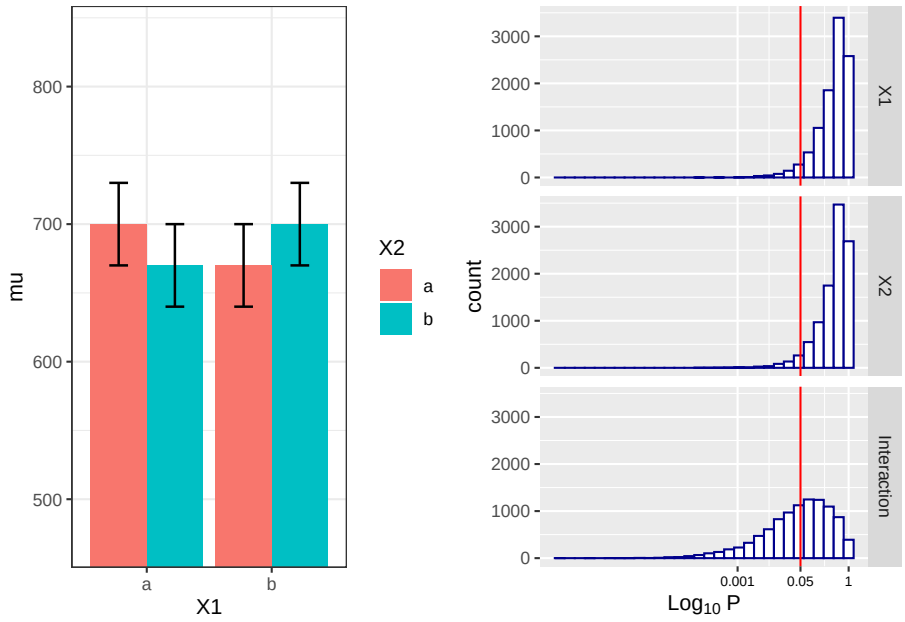
The empirical power is easy to compute, it's just the proportion of simulation runs where $p < .05$.

```
power.res = apply(as.matrix(p_val_1), 2,
  function(x) round(mean(ifelse(x < .05, 1, 0) * 100),2))
power.res
```

```
##          X1          X2 Interaction
##        4.80        4.66       47.50
```

Visualize the distributions of p-values

```
# plot the known effects


means = data.frame(cond[1:4,], mu, SE = sigma / sqrt(nsubs))
plt1 = ggplot(means, aes(y = mu, x = X1, fill = X2)) +
geom_bar(position = position_dodge(), stat = "identity") +
geom_errorbar(
aes(ymin = mu - SE, ymax = mu + SE),
position = position_dodge(width = 0.9),
size = .6,
width = .3
) +
coord_cartesian(ylim = c((.7 * min(mu)), 1.2 * max(mu))) +
theme_bw()

# melt the data into a ggplot friendly 'long' format

plotData <- melt(p_val_1, value.name = 'p')

# plot each of the p-value distributions on a log scale
options(scipen = 999) # 'turn off' scientific notation
plt2 = ggplot(plotData, aes(x = p)) +
scale_x_log10(breaks = c(1, 0.05, 0.001),
labels = c(1, 0.05, 0.001)) +
geom_histogram(colour = "darkblue", fill = "white") +
geom_vline(xintercept = 0.05, colour = 'red') +
facet_grid(variable ~ .) +
labs(x = expression(Log[10] ~ P)) +
theme(axis.text.x = element_text(color = 'black', size = 7))

# arrange plots side by side and print
grid.arrange(plt1, plt2, nrow = 1)
```



We can reproduce this simulation:

```
# true effects (in this case, a double dissociation)
mu = c(700, 670, 670, 700)
sigma = 150  # population standard deviation
n <- 25
sd <- 150
r <- 0.75
string = "2w*2w"
alpha_level <- 0.05
labelnames = c("age", "old", "young", "color", "blue", "red")
design_result <- ANOVA_design(design = string,
                              n = n,
                              mu = mu,
                              sd = sd,
                              r = r,
                              labelnames = labelnames)
```

```
simulation_result <- ANOVA_power(design_result,
                                 alpha_level = alpha_level,
                                 nsims = nsims,
                                 verbose = FALSE)
```

Table 3.8: Simulated ANOVA Result

	power	effect_size
anova_age	4.99	0.0396351
anova_color	5.17	0.0405810
anova_age:color	47.80	0.1641690

```
exact_result <- ANOVA_exact(design_result,
                            alpha_level = alpha_level,
                            verbose = FALSE)
```

Table 3.9: Exact ANOVA Result

	power	partial_eta_squared	cohen_f	non_centrality
age	5.00000	0.0000000	0.0000000	0
color	5.00000	0.0000000	0.0000000	0
age:color	48.40183	0.1428571	0.4082483	4

The simulations yield closely matching results.

3.4.2 Examine variation of means and correlation

```
# define the parameters
# true effects (in this case, a double dissociation)
mu = c(700, 670, 690, 750)
sigma = 150  # population standard deviation
rho = 0.4 # correlation between repeated measures
nsubs = 25 # how many subjects?
nsims = nsims # how many simulation replicates?

# create 2 factors representing the 2 independent variables
cond = data.frame(X1 = rep(factor(letters[1:2]), nsubs * 2),
X2 = rep(factor(letters[1:2]), nsubs, each = 2))

# create a subjects factor
subject = factor(sort(rep(1:nsubs, 4)))

# combine above into the design matrix
dm = data.frame(subject, cond)
```

Build Sigma: the population variance-covariance matrix

```
# create k x k matrix populated with sigma
sigma.mat <- rep(sigma, 4)
S <-
matrix(sigma.mat,
ncol = length(sigma.mat),
nrow = length(sigma.mat))

# compute covariance between measures
Sigma <- t(S) * S * rho

# put the variances on the diagonal
diag(Sigma) <- sigma ^ 2
```

Run the simulation

```
# stack 'nsims' individual data frames into one large data frame
df = dm[rep(seq_len(nrow(dm)), nsims), ]

# add an index column to track the simulation run
df$simID = sort(rep(seq_len(nsims), nrow(dm)))

# sample the observed data from a multivariate normal distribution
# using MASS::mvrnorm with the mu and Sigma created earlier
# and bind to the existing df


make.y = expression(as.vector(t(mvrnorm(nsubs, mu, Sigma))))
df$y = as.vector(replicate(nsims, eval(make.y)))

# use do(), the general purpose complement to the specialized data
# manipulation functions available in dplyr, to run the ANOVA on
# each section of the grouped data frame created by group_by


mods <- df %>%
  group_by(simID) %>%
  do(model = aov(y ~ X1 * X2 + Error(subject / (X1 * X2)),
                 qr = FALSE, data = .))

  # extract p-values for each effect and store in a data frame
  p_val_2 = data.frame(mods %>%
                    do(as.data.frame(tidy(.$model[[3]]))$p.value[1])),
  mods %>% do(as.data.frame(tidy(.$model[[4]]))$p.value[1])),
  mods %>% do(as.data.frame(tidy(.$model[[5]]))$p.value[1])))
  colnames(p_val_2) = c('X1', 'X2', 'Interaction')
```

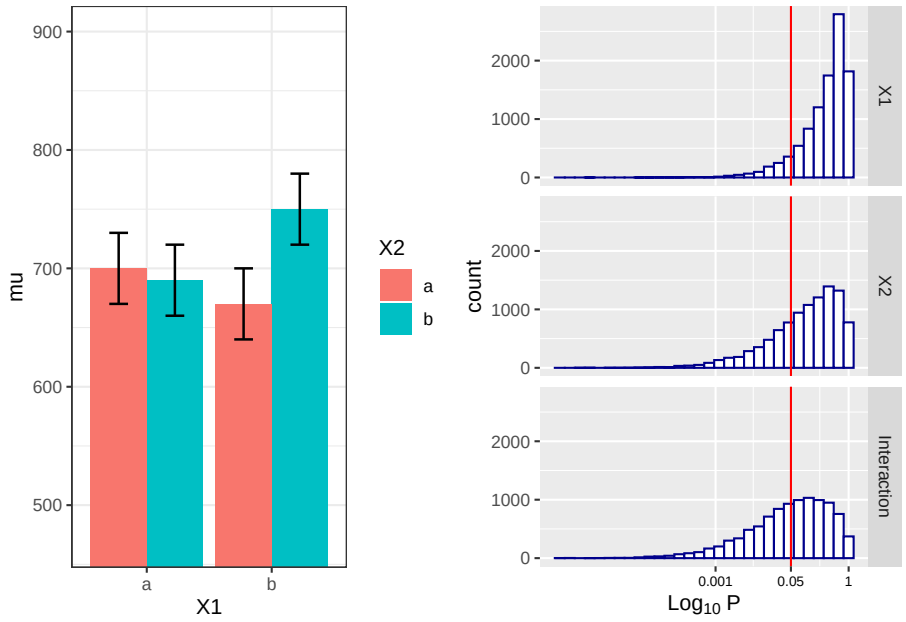
The empirical power is easy to compute, it's just the proportion of simulation runs where p <. 05.

```
power.res = apply(as.matrix(p_val_2), 2,
  function(x) round(mean(ifelse(x < .05, 1, 0) * 100),2))
power.res
```

```
##          X1          X2 Interaction
##        9.47       30.17       46.45
```

Visualize the distributions of p-values

```
means = data.frame(cond[1:4,], mu, SE = sigma / sqrt(nsubs))
plt1 = ggplot(means, aes(y = mu, x = X1, fill = X2)) +
geom_bar(position = position_dodge(), stat = "identity") +
geom_errorbar(
aes(ymin = mu - SE, ymax = mu + SE),
position = position_dodge(width = 0.9),
size = .6,
width = .3
) +
coord_cartesian(ylim = c((.7 * min(mu)), 1.2 * max(mu))) +
theme_bw()

# melt the data into a ggplot friendly 'long' format

plotData <- melt(p_val_2, value.name = 'p')

# plot each of the p-value distributions on a log scale
options(scipen = 999) # 'turn off' scientific notation
plt2 = ggplot(plotData, aes(x = p)) +
scale_x_log10(breaks = c(1, 0.05, 0.001),
labels = c(1, 0.05, 0.001)) +
geom_histogram(colour = "darkblue", fill = "white") +
geom_vline(xintercept = 0.05, colour = 'red') +
facet_grid(variable ~ .) +
labs(x = expression(Log[10] ~ P)) +
theme(axis.text.x = element_text(color = 'black', size = 7))

# arrange plots side by side and print
grid.arrange(plt1, plt2, nrow = 1)
```



We can reproduce this simulation:

```
# true effects (in this case, a double dissociation)
mu = c(700, 670, 690, 750)
sigma = 150  # population standard deviation
n <- 25
sd <- 150
r <- 0.4
string = "2w*2w"
alpha_level <- 0.05
labelnames = c("AGE", "old", "young",
               "COLOR", "blue", "red")

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

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

Table 3.10: Simulated ANOVA Result

	power	effect_size
anova_AGE	30.60	0.1143202
anova_COLOR	9.41	0.0541130
anova_AGE:COLOR	46.22	0.1595391

```
exact_result <- ANOVA_exact(design_result, alpha_level = alpha_level)
```

```
## Power and Effect sizes for ANOVA tests
##             power partial_eta_squared cohen_f non_centrality
## AGE       30.4009              0.0864  0.3074         2.2685
## COLOR      9.5071              0.0171  0.1318         0.4167
## AGE:COLOR 45.9803              0.1351  0.3953         3.7500
##
## Power and Effect sizes for pairwise comparisons (t-tests)
##                                              power effect_size
## p_AGE_old_COLOR_blue_AGE_old_COLOR_red     14.16       -0.18
## p_AGE_old_COLOR_blue_AGE_young_COLOR_blue   5.98       -0.06
## p_AGE_old_COLOR_blue_AGE_young_COLOR_red   30.91        0.30
## p_AGE_old_COLOR_red_AGE_young_COLOR_blue    9.00        0.12
## p_AGE_old_COLOR_red_AGE_young_COLOR_red    64.66        0.49
## p_AGE_young_COLOR_blue_AGE_young_COLOR_red 41.80        0.37
```

Table 3.11: Exact ANOVA Result

	power	partial_eta_squared	cohen_f	non_centrality
AGE	30.400885	0.0863588	0.3074437	2.2685185
COLOR	9.507147	0.0170648	0.1317616	0.4166667
AGE:COLOR	45.980305	0.1351351	0.3952847	3.7500000

3.5 Part 5

Credit for the non-centrality parameter for the interaction goes to Andrew Vigotsky

3.5.1 2x2 ANOVA, within design

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

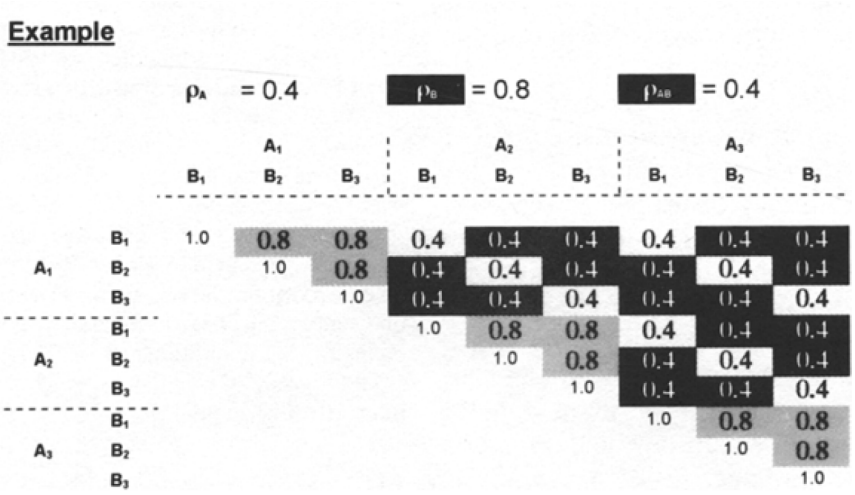


Figure 1. Representation of a correlation matrix for a 3 (A) × 3 (B) RM ANOVA: General form and numeric example. ρ_A and ρ_B represent the average correlation among the A and B (pooled) trials, respectively, and ρ_{AB} represents the average correlation among the AB coefficients having dissimilar levels.

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

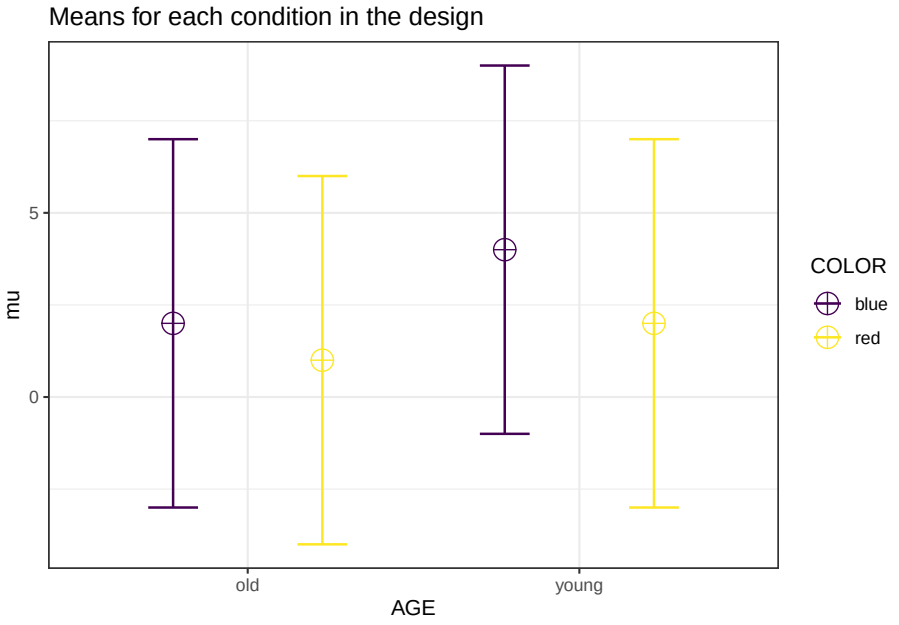
For the main effectof B: $\sigma_e^2 = \sigma^2(1 - \overline{\rho}_B) + \sigma^2(p-1)(\overline{\rho}_A - \overline{\rho}_{AB})$

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

Let's now compare the formulas in Potvin and Schutz (2000) with `Superpower` with a simple scenario.

```
mu = c(2,1,4,2)
n <- 20
sd <- 5
r <- .77
string = "2w*2w"
```

```
alpha_level <- 0.05


design_result <- ANOVA_design(design = string,
                              n = n,
                              mu = mu,
                              sd = sd,
                              r = r,
                              labelnames = labelnames,
                              plot = TRUE)
```




```
design_result$cor_mat
```

```
##            old_blue old_red young_blue young_red
## old_blue       1.00    0.77       0.77      0.77
## old_red        0.77    1.00       0.77      0.77
## young_blue     0.77    0.77       1.00      0.77
## young_red      0.77    0.77       0.77      1.00
```

```
exact_result <- ANOVA_exact(design_result,
                            alpha_level = alpha_level,
                            verbose = FALSE)
```

Table 3.12: Exact ANOVA Result

	power	partial_eta_squared	cohen_f	non_centrality
AGE	75.61412	0.2917342	0.6417938	7.8260870
COLOR	75.61412	0.2917342	0.6417938	7.8260870
AGE:COLOR	14.36376	0.0437637	0.2139313	0.8695652

Further, as we use the analytical solution below, the variance components should be equal to the corresponding MSE from the `ANOVA_exact` produced `anova_table` object.

```
exact_result$aov_result$anova_table
```

```
## Anova Table (Type 3 tests)
##
## Response: y
##           num Df den Df  MSE      F      pes  Pr(>F)
## AGE            1     19 5.75 7.8261 0.291734 0.01149 *
## COLOR          1     19 5.75 7.8261 0.291734 0.01149 *
## AGE:COLOR      1     19 5.75 0.8696 0.043764 0.36278
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

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

```
k <- 1 #one group (because all factors are within)

rho_A <- .77 #mean r for factor A

rho_B <- .77 #mean r for factor B

rho_AB <- .77 #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] 5.75
```

```
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] 5.75
```

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

```
## [1] 5.75
```

```
#Create a mean matrix
mean_mat <- t(matrix(mu, nrow = m_B,ncol = m_A))
mean_mat
```

```
##      [,1] [,2]
## [1,]    2    1
## [2,]    4    2
```

```
# 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] 7.826087
```

```
#calculate degrees of freedom 1 - ignoring the sphericity correction
df1 <- (m_A - 1)

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)
pow_A*100
```

```
## [1] 75.61412
```

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

```
## [1] 7.826087
```

```
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)


pow_B*100
```

```
## [1] 75.61412
```

```
#Peform double summation courtesy of Andrew Vigotsky
term <- 0
for (i in 1:nrow(mean_mat)) {
  for (j in 1:ncol(mean_mat)) {
    term <- (term + (mean_mat[i,j] -
                       mean(mean_mat[i,]) -
                       mean(mean_mat[,j]) + mean(mean_mat))^2)
  }
}
term
```

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

```
#Calculate lambda for interaction term
lambda_AB <- n*term/variance_e_AB
lambda_AB
```

```
## [1] 0.8695652
```

```
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_AB*100
```

```
## [1] 14.36376
```

We can now compile all the analytical results into a single table, and see that the results match those from `ANOVA_exact`.

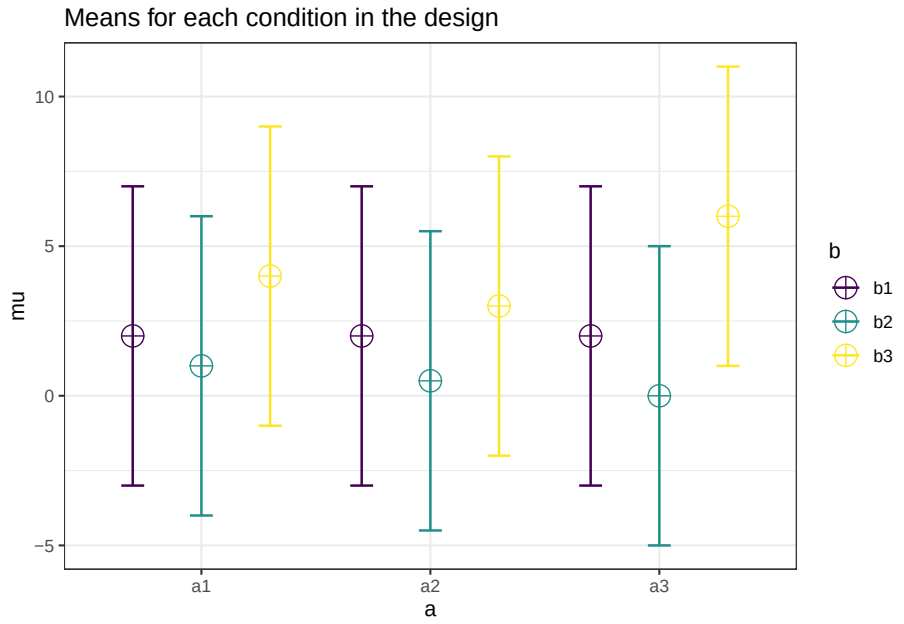
Table 3.13: Analytical Result

variance	lambda	power
5.75	7.8260870	75.61412
5.75	7.8260870	75.61412
5.75	0.8695652	14.36376

And we can see in the simple scenario matches `ANOVA_exact` and `ANOVA_power` output.

Let's now extend to one of the simlations presented by Potvin and Schutz (2000) with a 3x3 solution.

```
mu = c(2,1,4,
       2,.5,3,
       2,0,6)
n <- 20
sd <- 5
r <- c(1.0,0.8,0.8,0.4,0.4,0.4,0.4,0.4,0.4,
       0.8,1.0,0.8,0.4,0.4,0.4,0.4,0.4,0.4,
       0.8,0.8,1.0,0.4,0.4,0.4,0.4,0.4,0.4,
       0.4,0.4,0.4,1.0,0.8,0.8,0.4,0.4,0.4,
       0.4,0.4,0.4,0.8,1.0,0.8,0.4,0.4,0.4,
       0.4,0.4,0.4,0.8,0.8,1.0,0.4,0.4,0.4,
       0.4,0.4,0.4,0.4,0.4,0.4,1.0,0.8,0.8,
       0.4,0.4,0.4,0.4,0.4,0.4,0.8,1.0,0.8,
       0.4,0.4,0.4,0.4,0.4,0.4,0.8,0.8,1.0)


string = "3w*3w"
alpha_level <- 0.05

design_result <- ANOVA_design(design = string,
                              n = n,
                              mu = mu,
                              sd = sd,
                              r = r,
                              plot = TRUE)
```




```
design_result$cor_mat
```

```
##       a1_b1 a1_b2 a1_b3 a2_b1 a2_b2 a2_b3 a3_b1 a3_b2 a3_b3
## a1_b1   1.0   0.8   0.8   0.4   0.4   0.4   0.4   0.4   0.4
## a1_b2   0.8   1.0   0.8   0.4   0.4   0.4   0.4   0.4   0.4
## a1_b3   0.8   0.8   1.0   0.4   0.4   0.4   0.4   0.4   0.4
## a2_b1   0.4   0.4   0.4   1.0   0.8   0.8   0.4   0.4   0.4
## a2_b2   0.4   0.4   0.4   0.8   1.0   0.8   0.4   0.4   0.4
## a2_b3   0.4   0.4   0.4   0.8   0.8   1.0   0.4   0.4   0.4
## a3_b1   0.4   0.4   0.4   0.4   0.4   0.4   1.0   0.8   0.8
## a3_b2   0.4   0.4   0.4   0.4   0.4   0.4   0.8   1.0   0.8
## a3_b3   0.4   0.4   0.4   0.4   0.4   0.4   0.8   0.8   1.0
```

The design now matches the correlation matrix in Figure 1 of Potvin and Schutz (2000).

And we can estimate power with `ANOVA_exact`.

```
exact_result <- ANOVA_exact(design_result,
                            alpha_level = alpha_level,
                            verbose = FALSE)
```

Further, as we use the analytical solution below, the variance components should be equal to the corresponding MSE from the `ANOVA_exact` produced `anova_table` object.

Table 3.14: Exact ANOVA Result

	power	partial_eta_squared	cohen_f	non_centrality
a	9.441726	0.0156250	0.1259882	0.6031746
b	100.000000	0.7020906	1.5351629	89.5555556
a:b	90.092634	0.1778846	0.4651605	16.4444444

```
exact_result$aov_result$anova_table
```

```
## Anova Table (Type 3 tests)
##
## Response: y
##     num Df den Df MSE       F     pes          Pr(>F)
## a        2     38  35  0.3016 0.01563        0.741397
## b        2     38   5 44.7778 0.70209 0.0000000001018 ***
## a:b      4     76   5  4.1111 0.17788        0.004542 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
k <- 1 #one group (because all factors are within)

rho_A <- .4 #mean r for factor A

rho_B <- .8 #mean r for factor B

rho_AB <- .4 #mean r for factor AB

alpha <- 0.05

sigma <- sd

m_A <- 3 #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] 35
```

```
m_B <- 3 #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] 5
```

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

```
## [1] 5
```

```
#Create a mean matrix
mean_mat <- t(matrix(mu, nrow = m_B,ncol = m_A))
mean_mat
```

```
##      [,1] [,2] [,3]
## [1,]    2  1.0    4
## [2,]    2  0.5    3
## [3,]    2  0.0    6
```

```
# 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] 0.6031746
```

```
#calculate degrees of freedom 1 - ignoring the sphericity correction
df1 <- (m_A - 1)

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)
pow_A*100
```

```
## [1] 9.441726
```

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

```
## [1] 89.55556
```

```
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)


pow_B*100
```

```
## [1] 100
```

```
#Peform double summation courtesy of Andrew Vigotsky
term <- 0
for (i in 1:nrow(mean_mat)) {
  for (j in 1:ncol(mean_mat)) {
    term <- ((term + (mean_mat[i,j] -
                        mean(mean_mat[i,]) -
                        mean(mean_mat[,j]) + mean(mean_mat))^2))
  }
}
term
```

```
## [1] 4.111111
```

```
#Calculate lambda for interaction term
lambda_AB <- n*term/variance_e_AB
lambda_AB
```

```
## [1] 16.44444
```

```
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_AB*100
```

```
## [1] 90.09263
```

Again, when we compile all the analytical results into a single table we can see that the results match those from `ANOVA_exact`.

Table 3.15: Analytical Result

variance	lambda	power
35	0.6031746	9.441726
5	89.5555556	100.000000
5	16.4444444	90.092634

3.6 Multivariate ANOVA (MANOVA)

A large proportion of research with within-subjects manipulations, or repeated measures, rely upon the "univariate" approach (Maxwell et al., 2004). While this approach is valid, when corrections for sphercity are applied, it may not be the most powerful or informative analysis plan. Instead, researchers should consider a multivariate analysis (MANOVA). While the MANOVA is not "assumption free" it does not assume sphercity which makes it a very attractive analytical tool and the preferred method of analysis for some [Maxwell et al. (2004).

For a simple one-way repeated measures design, there are some simple guidelines for power analysis set forth by Maxwell et al. (2004) (pg. 750). All that is needed in the effect size calculated as:

$$d = \frac{\mu_{max} - \mu_{min}}{\sigma}$$

This assumes that each level has a common standard deviation (i.e., there is only 1 `sd` input for the design).

In addition, the non-centrality parameter of the F-statistic can be estimated from Vonesh and Schork (1986) equations as the following:

$$\delta^2 = \frac{n \cdot d^2}{2 \cdot (1 - \rho_{min})}$$

Let us assume we have a `2w` design with `mu = c(0,0.5)`, a common standard deviation of 1 (`sd=1`), and correlation between `a1` and `a2` of r = .4 and a total sample size of 15 (`n = 15`) participants. Power could then be calculated with the following `R` code.

```
mu = c(0,0.5)
rho = .4
sd = 2
n = 15
d = (max(mu)-min(mu))/sd
noncentrality = ((n*d^2) / (2*(1-min(rho))))
noncentrality
```

```
## [1] 0.78125
```

```
#Critical F
Ft <- qf((1 - .05), 1, 14)
Ft
```

```
## [1] 4.60011
```

```
#Power
power <- (1 - pf(Ft,
                 1,
                 14,
                 noncentrality)) * 100
power
```

```
## [1] 13.07682
```

Now we replicate in `Superpower`.

```
design_result <- ANOVA_design("2w",
                              n = n,
                              r = rho,
                              sd = sd,
                              mu = mu)

exact_result <- ANOVA_exact(design_result, verbose = FALSE)
```

Table 3.16: MANOVA Result

	power	pillai_trace	cohen_f	non_centrality
(Intercept)	8.401182	0.0233572	0.1546474	0.3348214
a	13.076818	0.0528541	0.2362278	0.7812500

The problem with this formula for determining power is that it is inexact and makes a number of assumptions (Maxwell et al. (2004), ppg. 752). In reality, it can only give a lower bound estimate of power for a given design, and the actual power may be much higher. This is problematic because it could lead to inefficient study design (e.g., determining you need 20 participants when adequate power could be achieved with less participants). This will become increasingly important with designs with multiple levels and possible violations of the assumption of sphericity.

3.7 Sphericity Assumption

In all the above examples, you will notice that the correlation is the sample between repeated measures. For most experiments in real life, this will not be the case. The correlations between levels may vary. This is problematic because the univariate (ANOVA) approach assumes sphericity, which means, for all intents and purposes, that the correlations between factor-levels are equal and the standard deviations at each level are equal as well. This assumption is tenuous at best and is typically "adjusted" for by applying a sphericity correction (e.g, Greenhouse-Geisser). How bad can it get? Well, let's simulate an example below.

```
design_result <- ANOVA_design("4w",
                              n = 29,
                              r = c(.05,.15,.25,.55, .65, .9
                                    ),
                              sd = 1,
                              mu= c(0,0,0,0))

#In order to simulate violations we MUST use ANOVA_power
power_result_s1 <- ANOVA_power(design_result, nsims = nsims, verbose = FALSE)
```

Table 3.17: Simulated ANOVA Result

	power	effect_size
anova_a	7.16	0.0344525

As we can see, the actual type I error rate far exceeds the typical 5%!

Now, let's pour gasoline on the fire and see what happens when we make the sphericity violation worse by varying the standard deviations.

```
design_result <- ANOVA_design("4w",
                              n = 29,
                              r = c(.05,.15,.25,.55, .65, .9
                                    ),
                              sd = c(1,3,5,7),
                              mu= c(0,0,0,0))

#In order to simulate violations we MUST use ANOVA_power
power_result_s2 <- ANOVA_power(design_result, nsims = nsims, verbose = FALSE)
```

Table 3.18: Simulated ANOVA Result

	power	effect_size
anova_a	8.37	0.0346856

These inflated error rates are obviously a problem, which begs the question what do we do about them?

In our experience, most researchers default to using a Greenhouse-Geisser adjustment for sphericity, but this may not be the most statistical efficient way of dealing with violations of sphercity. Further, as we can see from the simulation (`power_result_s2`) the MANOVA maintains the type I error rate at 5%.

Table 3.19: Simulated MANOVA Result

	power
manova_(Intercept)	5.16
manova_a	5.16

3.7.1 MANOVA or Sphericity Adjustment?

In addition to adjusting for sphericity, one could also simply use the multivariate approach to repeated measures. While it is tempting to simply say one approach is superior to another, that is not case when the sample size is small (Maxwell et al. (2004), ppg 775). Some general guidelines were proposed by Algina and Keselman (1997):

MANOVA when `levels <= 4, epsilon <= .9, n > levels + 15` and `5 <= levels <= 8, epsilon <= .85, n > levels + 30`.

However, `ANOVA_power` can make the solution easy. Simply simulate, like we have done above, for both the null situation (no differences) and with the hypothesized effect (to determine power), and see which approach best balances type I and II error rates.

As we saw above, the unadjusted repeated measures ANOVA has an elevated type I error rate, but the MANOVA analysis of the same data above approximately preserves the type I error rate. However, how does this perform relative to the sphericity corrections? Let's simulate again, and compare the results of the different corrections.

First we setup the design.

```
design_result <- ANOVA_design("4w",
                              n = 29,
                              r = c(.05,.15,.25,.55, .65, .9
                              ),
                              sd = c(1,3,5,7),
                              mu= c(0,0,0,0))
```

```
power_result_none <- ANOVA_power(design_result, nsims = nsims, verbose = FALSE)
```

Table 3.20: Simulated MANOVA Result

	power
manova_(Intercept)	5.16
manova_a	5.16

```
power_result_gg <- ANOVA_power(design_result, correction = "GG",
                               nsims = nsims, verbose = FALSE)
```

Table 3.21: Simulated ANOVA Result

	power	effect_size
anova_a	4.59	0.0339026

```
power_result_hf <- ANOVA_power(design_result, correction = "HF",
                               nsims = nsims, verbose = FALSE)
```

Table 3.22: Simulated ANOVA Result

	power	effect_size
anova_a	5.12	0.0346905

Both the sphericity corrections, Greenhouse-Geisser (`GG`) and Huynh-Feldt (`HF`), as well as the MANOVA were able to adequately control type I error rate. Howevever, Greenhouse-Geisser seemed to be a tad conservative. We can now directly compare the power of the MANOVA or `HF`-adjusted approach. We can adjust the study design to the alternative, or hypothesized, model with the predicted means of `0, 0.75, 1.5, 3` instead of the null model.

```
design_result_power <- ANOVA_design("4w",
                                    n = 29,
                                    r = c(.05,.15,.25,.55, .65, .9
                                    ),
                                    sd = c(1,3,5,7),
                                    mu= c(0,0.75,1.5,3))

power_result_hfeffect <- ANOVA_power(design_result_power,
                                         correction = "HF",
                                         nsims = nsims,
                                         verbose = FALSE)
```

Table 3.23: Simulated ANOVA Result

	power	effect_size
anova_a	60.48	0.1479382

Table 3.24: Simulated MANOVA Result

	power
manova_(Intercept)	49.60
manova_a	49.64

Well, it appears the MANOVA based approach has roughly ~10% lower power compared the HF-adjusted ANOVA. However, the study still appears to be underpowered so we can increase the sample size. We will continue to evaluate the MANOVA results because, as Algina and Keselman (1997) note, the power disadvantage of MANOVA is diminished with increased sample size.

```
design_result_power <- ANOVA_design("4w",
                                    n = 29,
                                    r = c(.05,.15,.25,.55, .65, .9),
                                    sd = c(1,3,5,7),
                                    mu= c(0,0.75,1.5,3))

power_result_hfeffect <- ANOVA_power(design_result_power,
                                     correction = "HF",
                                     nsims = nsims,
                                     verbose = FALSE)
```

Table 3.25: Simulated ANOVA Result

	power	effect_size
anova_a	84.69	0.1385088

Table 3.26: Simulated MANOVA Result

	power
manova_(Intercept)	74.11
manova_a	79.18

Again, the HF-adjusted analysis appears to be more powerful for *this very specific experimental design*. The difference in power between univariate and multivariate output is diminished when the sample size is increased.

Chapter 4

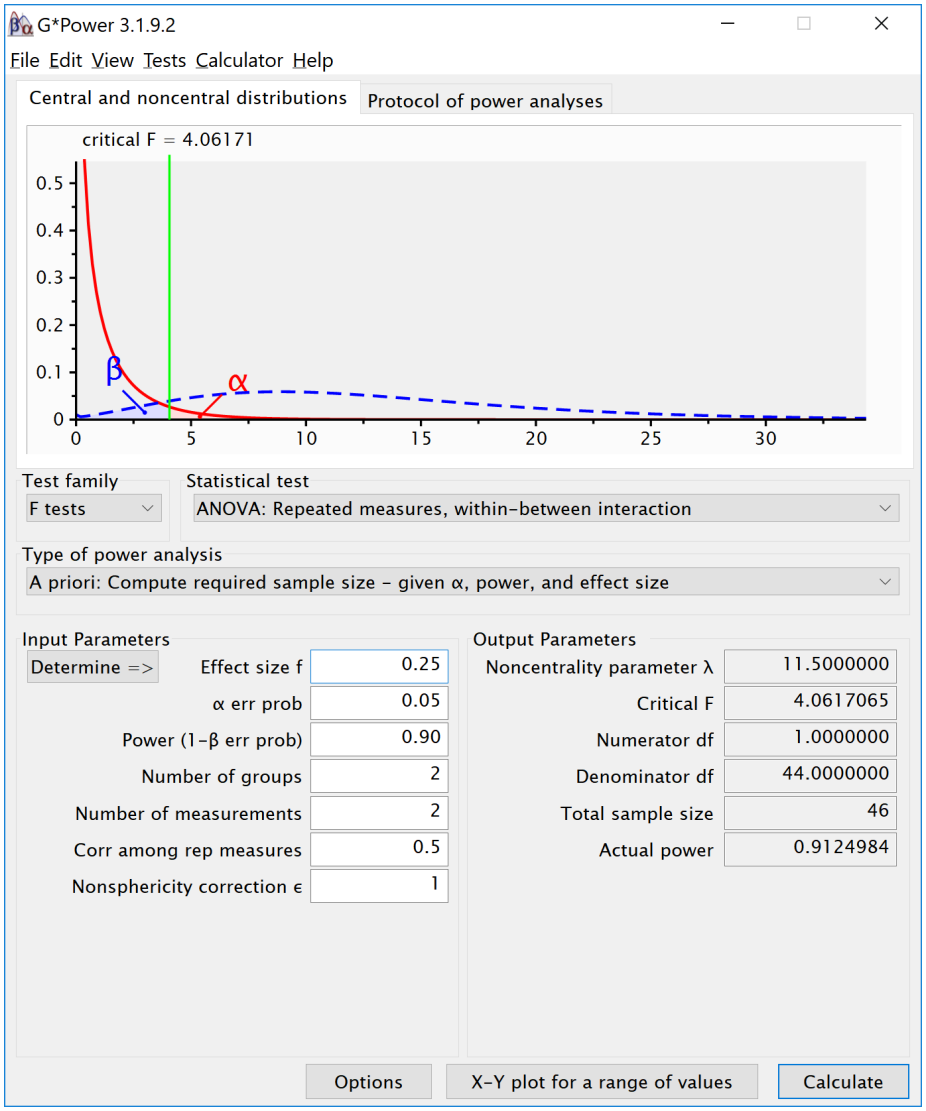
Mixed ANOVA

So far we have discussed the simple one-way ANOVA, various forms of the repeated measures ANOVA (and multivariate alternatives), but we have not yet looked at "mixed ANOVA" wherein there are between and within subjects factors. Therefore, in this chapter we will show how a power analysis for these designs is performed in `Superpower`. Further, we will introduce comparisons to SAS's `PROC GLMPOWER` which is a very powerful tool when designing mixed factorial experiments.

4.1 Simple Mixed Designs

We can simulate a two-way ANOVA with a specific alpha, sample size and effect size, to achieve a specified statistical power. We will try to reproduce the power analysis by GPower (Faul et al., 2007) for an F-test from an ANOVA with a repeated measures, within-between interaction effect. While GPower is a great tool it has limited options for mixed factorial ANOVAs.

Let us setup a simple 2x2 design. For the 2-way interaction, the result should be a power of 91.25% with at total sample size of 46. Since we have 2 groups in the between -subjects factor that means the sample size per group is 23 with two measurements per subject (i.e., `2w`).



Now, we can repeat the process in `Superpower`.

```
mu <- c(-0.25, 0.25, 0.25,-0.25)
n <- 23
sd <- 1
r <- 0.5
string = "2w*2b"
alpha_level <- 0.05
labelnames = c("age", "old", "young", "color", "blue", "red")
design_result <- ANOVA_design(
design = string,
n = n,
mu = mu,
sd = sd,
r = r,
labelnames = labelnames
)
```

```
simulation_result <- ANOVA_power(design_result,
                                 alpha_level = alpha_level,
                                 nsims = nsims,
                                 verbose = FALSE)
```

Table 4.1: Simulated ANOVA Result

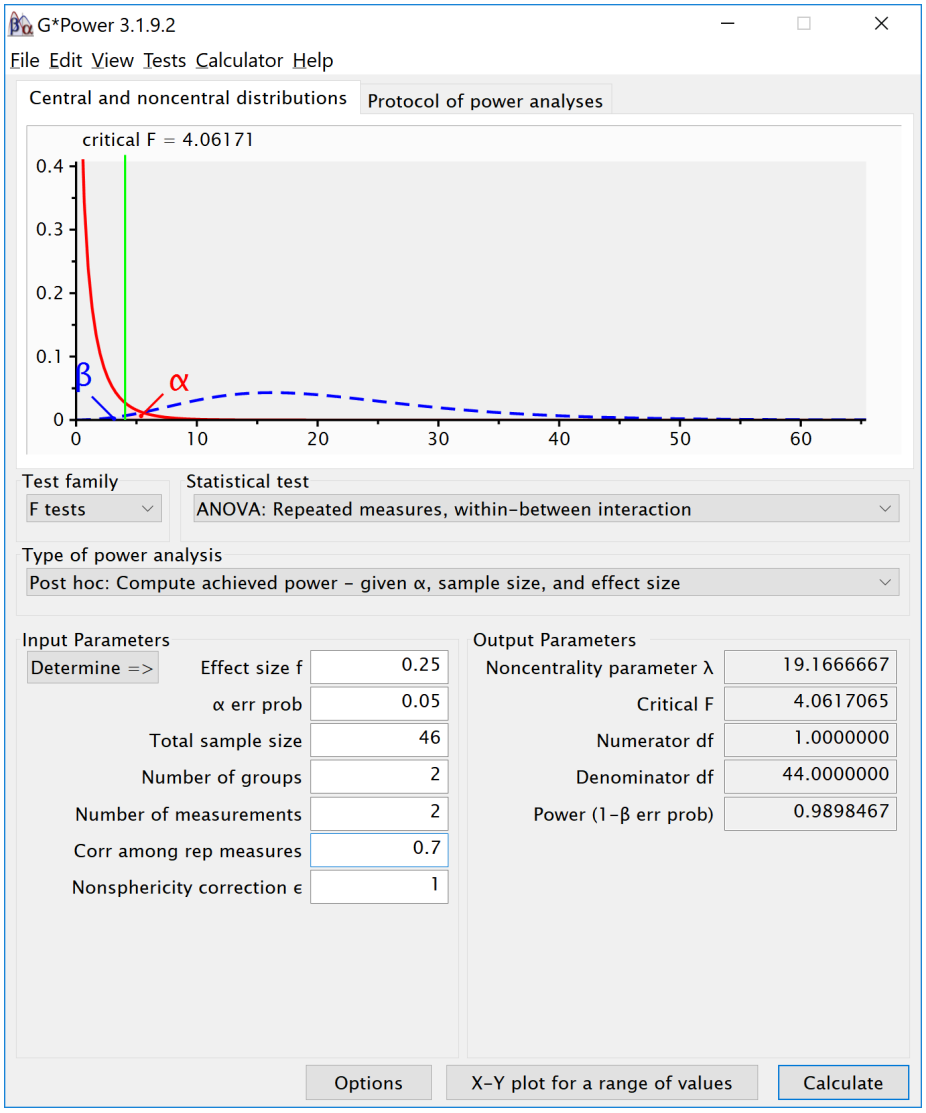
	power	effect_size
anova_color	4.45	0.0215448
anova_age	5.29	0.0221193
anova_color:age	91.26	0.2155901

```
exact_result <- ANOVA_exact(design_result,
                            alpha_level = alpha_level,
                            verbose = FALSE)
```

Table 4.2: Exact ANOVA Result

	power	partial_eta_squared	cohen_f	non_centrality
color	5.00000	0.0000000	0.0000000	0.0
age	5.00000	0.0000000	0.0000000	0.0
color:age	91.24984	0.2072072	0.5112374	11.5

Now, we can simulate the same two-way ANOVA but increasing the correlation to r=0.7.



```
mu <- c(-0.25, 0.25, 0.25, -0.25)
n <- 23
sd <- 1
r <- 0.7
string = "2w*2b"
alpha_level <- 0.05
labelnames = c("age", "old", "young", "color", "blue", "red")
design_result <- ANOVA_design(design = string,
                              n = n,
                              mu = mu,
                              sd = sd,
                              r = r,
                              labelnames = labelnames)
```

```
simulation_result <- ANOVA_power(design_result,
                                 alpha_level = alpha_level,
                                 nsims = nsims,
                                 verbose = FALSE)
```

Table 4.3: Simulated ANOVA Result

	power	effect_size
anova_color	4.93	0.0220656
anova_age	4.85	0.0224569
anova_color:age	99.00	0.3081551

```
exact_result <- ANOVA_exact(design_result,
                            alpha_level = alpha_level,
                            verbose = FALSE)
```

Table 4.4: Exact ANOVA Result

	power	partial_eta_squared	cohen_f	non_centrality
color	5.00000	0.0000000	0.0000000	0.00000
age	5.00000	0.0000000	0.0000000	0.00000
color:age	98.98467	0.3034301	0.6600046	19.16667

4.2 Complex Mixed Designs

Now, we are to the most complicated calculations. Similiar to the simple one-way repeated measures ANOVA, a mixed ANOVA assumes sphercity. Therefore, in most situations a multivariate approach, MANOVA, is recommended (Maxwell et al., 2004). To our knowledge, the only program that can accurately calculate power for mixed designs with greater than 2 levels is SAS's `PROC GLMPOWER` (SAS, 2015). The procedure utilizes approximate analytical solutions derived by Muller and Peterson (1984) and O'Brien and Shieh (1999). According to the documentation, these analytical solutions are very accurate for all but small N situations (sorry exercise scientists!). Eventually, this chapter will document the analytical solution in a step-by-step fashion, but for the time being we will just directly compare `Superpower` to `GLMPOWER` from a few examples provided by SAS.

4.2.1 2b*3w Design

Here we will use a modified example from SAS (2015) pg. 3739. In this hypothetical experiment Suppose you are planning an experiment to study the growth of two varieties of flowers over the course of 3 weeks. The planned data analysis is a two-way ANOVA with flower height as the outcome and a model consisting of the effects of time, flower variety, and their interaction.

First we can set up the dataframe in SAS. This is similiar to the `mu` command in `ANOVA_design`.

```
data Exemplary2;
input variety Height1 Height2 Height3;
datalines;
1 14 16 21
2 10 15 16
;
```

We can now solve for power in `PROC GLMPOWER`. In this case, we set up a multivariate repeated measures model, with a total of 40 flowers (20 per variety). We also setup the within-factor correlations with the CORRS command.

```
proc glmpower data=Exemplary2;
class Variety Exposure;
model Height1 Height2 Height3 = Variety;
repeated Time contrast;
power
mtest = pt
stddev = 5
ntotal = 40
```

```
power = .
MATRIX("MyCorrs")= (.75,
                    0.5625,.75)
CORRS= "MyCorrs";
run;
```

The power analysis results then get printed to the SAS ouput page.

The SAS System

The GLMPOWER Procedure
F Test for Multivariate Model

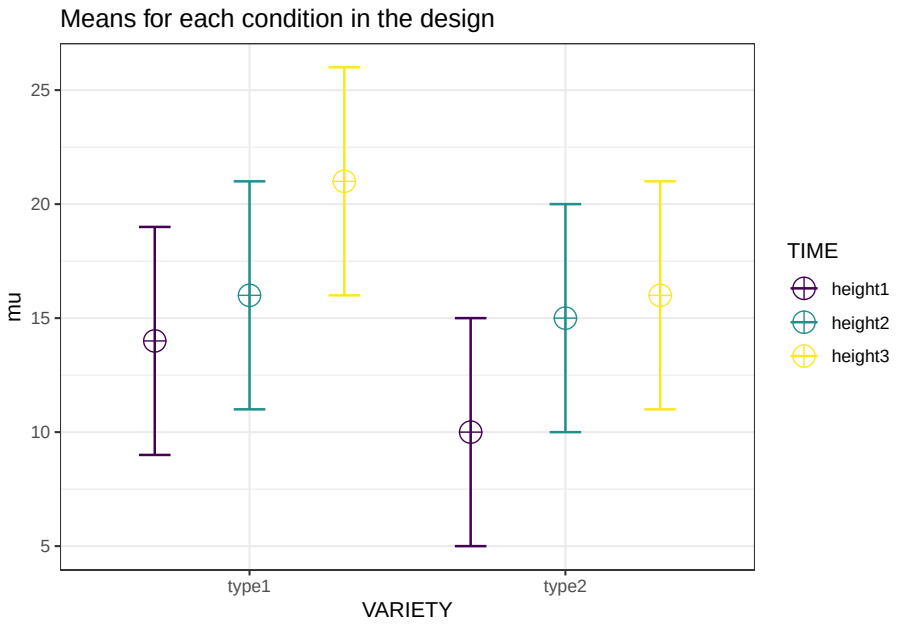
Fixed Scenario Elements	
Wilks/HLT/PT Method	O'Brien-Shieh
F Test	Pillai's Trace
Error Standard Deviation	5
Correlations	MyCorrs
Total Sample Size	40
Alpha	0.05

Computed Power						
Index	Transformation	Source	Effect	Num DF	Den DF	Power
1	Time	Intercept	Time	2	37	>.999
2	Time	variety	Time*variety	2	37	0.962
3	Mean(Dep)	Intercept	Intercept	1	38	>.999
4	Mean(Dep)	variety	variety	1	38	0.637

Now let's replicate this in `R` with `Superpower`. First, we setup the same design with `ANOVA_design`.

```
cor_1 <- matrix(c(1, .75, .5625,
                  .75, 1, .75,
                  .5625, .75, 1), nrow=3)

cor_2 <- cor_1*0

rho_mat <- cbind(rbind(cor_1,cor_2),
                      rbind(cor_2,cor_1))

design_result <- ANOVA_design("2b*3w",
                              n = 20,
```

```
                                sd = 5,
                                mu = c(14, 16, 21,
                                       10, 15, 16),
                                r = rho_mat,
                                labelnames = c("VARIETY",
                                               "type1", "type2",
                                               "TIME",
                                               "height1", "height2",
                                               "height3"),
                                plot = TRUE)
```




```
exact_result <- ANOVA_exact(design_result, verbose = FALSE,
                            correction = "none",
                            alpha_level = .05)
```

Table 4.5: MANOVA Result

	power	partial_eta_squared	cohen_f	non_centrality
VARIETY	63.65566	0.1287208	0.3843667	5.614035
TIME	100.00000	0.5877929	1.1941377	108.373333
VARIETY:TIME	84.08029	0.1273729	0.3820535	11.093333

You will notice a small discrepancy between the two power estimates for the main

effect of variety. This difference is due to the analytical solutions problems with small sample sizes. You will see in the example below that the results match when the total sample size is much greater.

4.2.2 2b*4w Design

Now we move onto the example from pg 3794 of SAS (2015).

As stated in the manual:

> Logan, Baron, and Kohout (1995) and Guo et al. (2013) study the effect of a dental intervention on the memory of pain after root canal therapy. The intervention is a sensory focus strategy, in which patients are instructed to pay attention only to the physical sensations in their mouth during the root canal procedure. Suppose you are interested in the long-term effects of this sensory focus intervention, because avoidance behavior has been shown to build along with memory of pain. You are planning a study to compare sensory focus to standard of care over a period of a year, asking patients to self-report their memory of pain immediately after the procedure and then again at 1 week, 6 months, and 12 months. You use a scale from 0 (no pain remembered) to 5 (maximum pain remembered).

This makes it a `2b*4w` design with treatment as a between subjects factor, with two levels (sensory focus versus standard of care), and time as a within-subject factor is time, with four levels (0, 1, 26, and 52 weeks). In this case, we differ from SAS (2015) and we will solve for power with a sample size of 300 per group (600 total) with an alpha of `.01`. In addition, we want to see what the impact of chaning the common standard deviation will have on power.

So in SAS we set up the data.

```
data Pain;
input Treatment $ PainMem0 PainMem1Wk PainMem6Mo PainMem12Mo;
datalines;
SensoryFocus 2.40 2.38 2.05 1.90
StandardOfCare 2.40 2.39 2.36 2.30
;
```

Then we can run the analysis in SAS. Note that in the example SAS (2015) are assuming a linear exponential covariance matrix, which we can mimic in `R`.

```
proc glmpower data=Pain;
class Treatment;
model PainMem0 PainMem1Wk PainMem6Mo PainMem12Mo = Treatment;
```

```
repeated Time contrast;
power
mtest = pt
alpha = 0.01
power = .
ntotal = 600
stddev = 0.92 1.04
matrix ("PainCorr") = lear(0.6, 0.8, 4, 0 1 26 52)
corrmat = "PainCorr";
run;
quit;
```

This produces a table with the result for a power analysis with 2 different common standard deviations.

The SAS System

The GLMPOWER Procedure
F Test for Multivariate Model

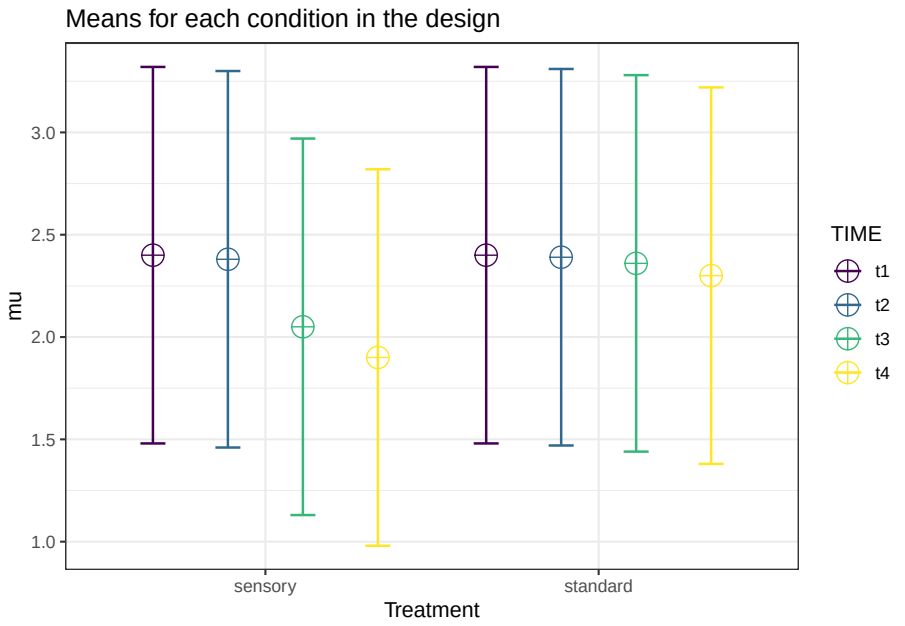
Fixed Scenario Elements	
Wilks/HLT/PT Method	O'Brien-Shieh
F Test	Pillai's Trace
Alpha	0.01
Correlation Matrix	PainCorr
Total Sample Size	600

Computed Power

Index	Transformation	Source	Std Dev	Effect	Num DF	Den DF	Power
1	Time	Intercept	0.92	Time	3	596	>.999
2	Time	Intercept	1.04	Time	3	596	>.999
3	Time	Treatment	0.92	Time*Treatment	3	596	0.996
4	Time	Treatment	1.04	Time*Treatment	3	596	0.976
5	Mean(Dep)	Intercept	0.92	Intercept	1	598	>.999
6	Mean(Dep)	Intercept	1.04	Intercept	1	598	>.999
7	Mean(Dep)	Treatment	0.92	Treatment	1	598	0.686
8	Mean(Dep)	Treatment	1.04	Treatment	1	598	0.552

We then replicate in R, first by setting up the "lear" correlation matrix.

```
cor_1 <- matrix(c(1,.6,.491,.399,
                  .6,1,.495,.402,
                  .491,.495,1,.491,
                  .399,.402,.491,1), nrow=4)

cor_2 <- cor_1*0

pain_cor_mat <- cbind(rbind(cor_1,cor_2),
                      rbind(cor_2,cor_1))

design_result <- ANOVA_design("2b*4w",
                              n = 300,
                              mu = c(2.4, 2.38, 2.05, 1.90,
                                     2.4, 2.39, 2.36, 2.30),
                              sd = .92,
                              r = pain_cor_mat,
                              labelnames = c("Treatment", "sensory", "standard",
                                             "TIME", "t1", "t2", "t3", "t4"),
                              plot = TRUE)
```

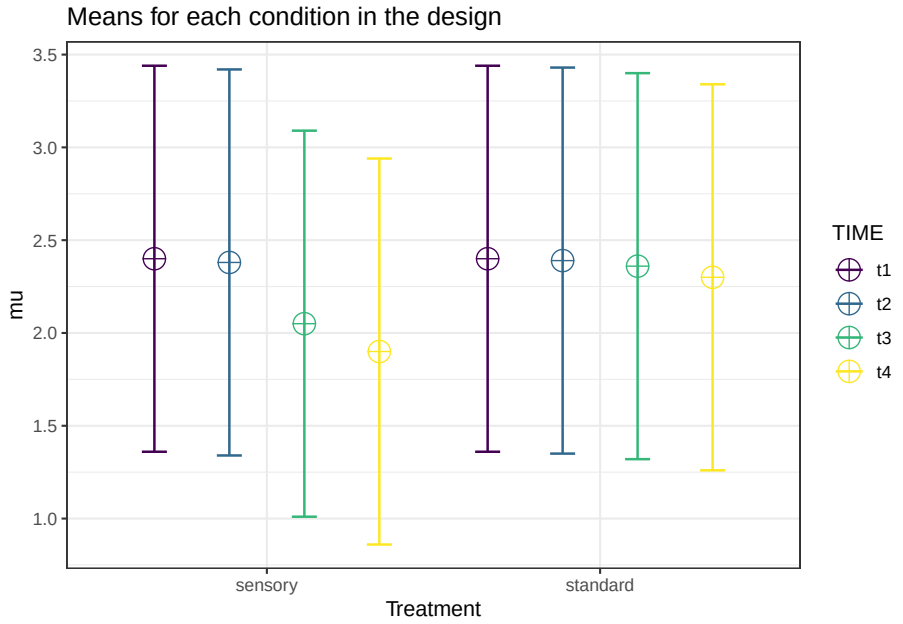



```
exact_result <- ANOVA_exact(design_result, verbose = FALSE,
                            alpha_level = .01)
```

Table 4.6: Simulated MANOVA Result

	power	pillai_trace	cohen_f	non_centrality
(Intercept)	100.00000	0.9094197	3.1685838	6003.874016
Treatment	68.59334	0.0155032	0.1254885	9.416918
TIME	99.99995	0.0998664	0.3330858	66.123912
Treatment:TIME	99.55718	0.0531965	0.2370344	33.486447

```
design_result <- ANOVA_design("2b*4w",
                              n = 300,
                              mu = c(2.4, 2.38, 2.05, 1.90,
                                     2.4, 2.39, 2.36, 2.30),
                              sd = 1.04,
                              r = pain_cor_mat,
                              labelnames = c("Treatment", "sensory", "standard",
                                             "TIME", "t1", "t2", "t3", "t4"),
                              plot = TRUE)
```




```
exact_result <- ANOVA_exact(design_result, verbose = FALSE,
                            alpha_level = .01)
```

As we can see, the results for this analysis match SAS perfectly.

Table 4.7: Simulated MANOVA Result

	power	pillai_trace	cohen_f	non_centrality
(Intercept)	100.00000	0.8870909	2.8029779	4698.297861
Treatment	55.22151	0.0121730	0.1110090	7.369156
TIME	99.99672	0.0798847	0.2946528	51.744895
Treatment:TIME	97.55110	0.0421158	0.2096843	26.204631

Chapter 5

Power for Three-way Interactions

There are almost no software solutions that allow researchers to perform power anaysis for more complex designs. Through simulation, it is relatively straightforward to examine the power for designs with multiple factors with many levels.

Let's start with a 2x2x2 between-subjects design. We collect 50 participants in each between participant condition (so 400 participants in total - $50(n) \times 2(levels) \times 2(levels) \times 2(levels) = 400$).

```
# With 2x2x2 designs,
# the names for paired comparisons can become very long.
# So here I abbreviate terms:
#   Size, Color, and Cognitive Load, have values:
# b = big, s = small, g = green,
# r = red, pres = present, abs = absent.
labelnames <- c("Size", "b", "s", "Color", "g", "r",
                "Load", "pres", "abs") #
design_result <- ANOVA_design(design = "2b*2b*2b",
                              #sample size per group
                              n = 50,
                              #pattern of means
                              mu = c(2, 2, 6, 1, 6, 6, 1, 8),
                              sd = 10, #standard deviation
                              labelnames = labelnames)
```

```
simulation_result <- ANOVA_power(design_result,
                                 alpha_level = alpha_level,
                                 nsims = nsims,
                                 verbose = FALSE)
```

Table 5.1: Simulated ANOVA Result

	power	effect_size
anova_Size	70.17	0.0181513
anova_Color	5.37	0.0026042
anova_Load	8.01	0.0031862
anova_Size:Color	32.24	0.0082572
anova_Size:Load	85.01	0.0247630
anova_Color:Load	8.10	0.0032246
anova_Size:Color:Load	84.88	0.0248926

```
exact_result <- ANOVA_exact(design_result,
                            alpha_level = alpha_level,
                            verbose = FALSE)
```

Table 5.2: Exact ANOVA Result

	power	partial_eta_squared	cohen_f	non_centrality
Size	70.330132	0.0156937	0.1262691	6.25
Color	5.000000	0.0000000	0.0000000	0.00
Load	7.895322	0.0006373	0.0252538	0.25
Size:Color	32.172729	0.0057070	0.0757614	2.25
Size:Load	84.912313	0.0224439	0.1515229	9.00
Color:Load	7.895322	0.0006373	0.0252538	0.25
Size:Color:Load	84.912313	0.0224439	0.1515229	9.00

```
#Analytical power calculation
power_analytic <- power_threeway_between(design_result)
power_analytic$power_A
```

```
## [1] 70.33333
```

```
power_analytic$power_B
```

```
## [1] 5
```

```
power_analytic$power_C
```

```
## [1] 7.895539
```

```
power_analytic$power_AB
```

```
## [1] 32.17471
```

```
power_analytic$power_AC
```

```
## [1] 84.91491
```

```
power_analytic$power_BC
```

```
## [1] 7.895539
```

```
power_analytic$power_ABC
```

```
## [1] 84.91491
```

```
power_analytic$eta_p_2_A
```

```
## [1] 0.01538462
```

```
power_analytic$eta_p_2_B
```

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

```
power_analytic$eta_p_2_C
```

```
## [1] 0.0006246096
```

```
power_analytic$eta_p_2_AB
```

```
## [1] 0.005593536
```

```
power_analytic$eta_p_2_AC
```

```
## [1] 0.02200489
```

```
power_analytic$eta_p_2_BC
```

```
## [1] 0.0006246096
```

```
power_analytic$eta_p_2_ABC
```

```
## [1] 0.02200489
```

We can also confirm the power analysis in GPower (Faul et al., 2007). GPower allows you to compute the power for a three-way interaction - if you know the Cohen's f value to enter. Cohen's f is calculated based on the means for the interaction, the sum of squares of the effect, and the sum of squares of the errors. This is quite a challenge by hand, but we can simulate the results, or use the analytical solution we programmed to get Cohen's f for the pattern of means that we specified.

```
# The power for the AC interaction (Size x Load) is 0.873535.
power_analytic$power_AC
```

```
## [1] 84.91491
```

```
# We can enter the Cohen's f for this interaction.
power_analytic$Cohen_f_AC
```

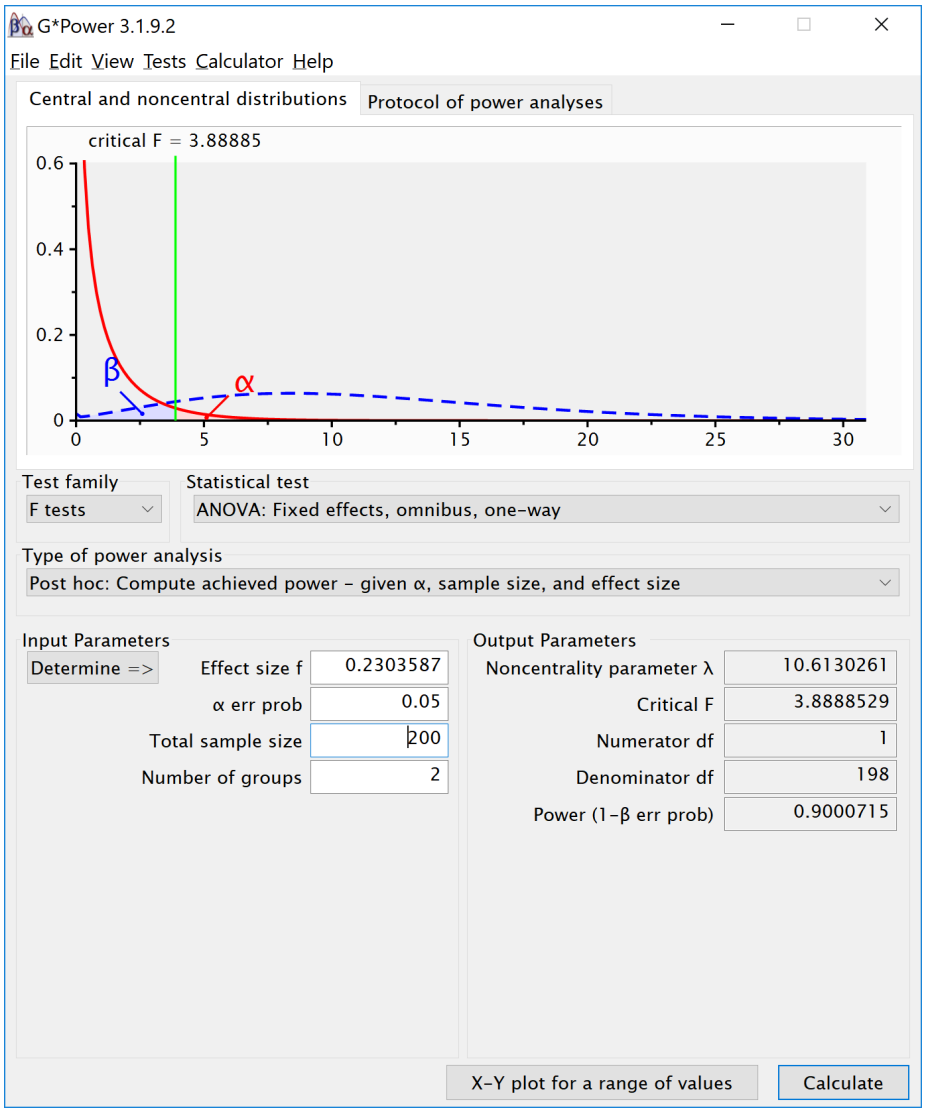
```
## [1] 0.15
```

```
# We can double check the calculated lambda
power_analytic$lambda_AC
```

```
## [1] 9
```

```
# We can double check the critical F value
power_analytic$F_critical_AC
```

```
## [1] 3.864929
```



A three-way ANOVA builds on the same principles as a one-way ANOVA. We look at whether the differences between groups are large, compared to the standard deviation. For the main effects we simply have 2 groups of 200 participants, and 2 means. If the population standard deviations are identical across groups, this is not in any way different from a one-way ANOVA. Indeed, we can show this by simulating a one-way ANOVA, where instead of 8 conditions, we have two conditions, and we average over the 4 groups of the other two factors. For example, for the main effect of size above can be computed analytically. There might be a small difference in the degrees of freedom of the two tests, or it is just random variation (And it will disappear when the number of iterations in

the simulation, `nsim`, is increased).

```
string <- "2b"
n <- 200
mu <- c(mean(c(2, 2, 6, 1)), mean(c(6, 6, 1, 8)))
sd <- 10
labelnames <- c("Size", "big", "small")
design_result <- ANOVA_design(design = string,
                   n = n,
                   mu = mu,
                   sd = sd,
                   labelnames = labelnames)
```

```
simulation_result <- ANOVA_power(design_result,
                                 alpha_level = alpha_level,
                                 nsims = nsims,
                                 verbose = FALSE)
```

Table 5.3: Simulated ANOVA Result

	power	effect_size
anova_Size	70.69	0.0180045

```
exact_result <- ANOVA_exact(design_result,
                            alpha_level = alpha_level,
                            verbose = FALSE)
```

Table 5.4: Exact ANOVA Result

	power	partial_eta_squared	cohen_f	non_centrality
Size	70.33333	0.0154607	0.1253137	6.25

```
# Power based on analytical solution
power_oneway_between(design_result)$power
```

```
## [1] 70.33333
```

Similarly, we can create a 2 factor design where we average over the third factor, and recreate the power analysis for the Two-Way interaction. For example, we can group over the Cognitive Load condition, and look at the Size by Color Interaction:

```
string <- "2b*2b"
n <- 100
mu <- c(mean(c(1, 1)), mean(c(6, 1)), mean(c(6, 6)), mean(c(1, 6)))
sd <- 10
labelnames <- c("Size", "big", "small", "Color", "green", "red")
design_result <- ANOVA_design(design = string,
                    n = n,
                    mu = mu,
                    sd = sd,
                    labelnames = labelnames)
```

```
simulation_result <- ANOVA_power(design_result,
                                 alpha_level = alpha_level,
                                 nsims = nsims,
                                 verbose = FALSE)
```

Table 5.5: Simulated ANOVA Result

	power	effect_size
anova_Size	70.93	0.0178407
anova_Color	5.20	0.0025269
anova_Size:Color	69.72	0.0176374

```
exact_result <- ANOVA_exact(design_result,
                            alpha_level = alpha_level,
                            verbose = FALSE)
```

Table 5.6: Exact ANOVA Result

	power	partial_eta_squared	cohen_f	non_centrality
Size	70.33227	0.0155376	0.1256297	6.25
Color	5.00000	0.0000000	0.0000000	0.00
Size:Color	70.33227	0.0155376	0.1256297	6.25

```
# Power based on analytical solution
power_res <- power_twoway_between(design_result)
power_res$power_A
```

```
## [1] 70.33228
```

```
power_res$power_B
```

```
## [1] 5
```

```
power_res$power_AB
```

```
## [1] 70.33228
```

```
string <- "2b*2b*2b"
n <- 50
mu <- c(5, 3, 2, 6, 1, 4, 3, 1)
sd <- 10
r <- 0.0
labelnames <- c("Size", "big", "small",
                "Color", "green", "red",
                "CognitiveLoad", "present", "absent")
design_result <- ANOVA_design(design = string,
                  n = n,
                  mu = mu,
                  sd = sd,
                  labelnames = labelnames)
```

```
simulation_result <- ANOVA_power(design_result,
                                 alpha_level = alpha_level,
                                 nsims = nsims,
                                 verbose = FALSE)
```

Table 5.7: Simulated ANOVA Result

	power	effect_size
anova_Size	42.71	0.0103728
anova_Color	5.52	0.0026837
anova_CognitiveLoad	11.10	0.0038783
anova_Size:Color	5.74	0.0026956
anova_Size:CognitiveLoad	6.05	0.0027663
anova_Color:CognitiveLoad	5.69	0.0027028
anova_Size:Color:CognitiveLoad	78.13	0.0213083

```
exact_result <- ANOVA_exact(design_result,
                            alpha_level = alpha_level,
                            verbose = FALSE)
```

Table 5.8: Exact ANOVA Result

	power	partial_eta_squared	cohen_f	non_centrality
Size	41.52809	0.0077519	0.0883883	3.0625
Color	5.71548	0.0001594	0.0126269	0.0625
CognitiveLoad	11.61777	0.0014329	0.0378807	0.5625
Size:Color	5.71548	0.0001594	0.0126269	0.0625
Size:CognitiveLoad	5.71548	0.0001594	0.0126269	0.0625
Color:CognitiveLoad	5.71548	0.0001594	0.0126269	0.0625
Size:Color:CognitiveLoad	78.32737	0.0189270	0.1388960	7.5625

```
#Analytical power calculation
power_analytic <- power_threeway_between(design_result)
power_analytic$power_A
```

```
## [1] 41.5306
```

```
power_analytic$power_B
```

```
## [1] 5.715533
```

```
power_analytic$power_C
```

```
## [1] 11.61827
```

```
power_analytic$power_AB
```

```
## [1] 5.715533
```

```
power_analytic$power_AC
```

```
## [1] 5.715533
```

```
power_analytic$power_BC
```

```
## [1] 5.715533
```

```
power_analytic$power_ABC
```

```
## [1] 78.33036
```

```
power_analytic$eta_p_2_A
```

```
## [1] 0.007598077
```

```
power_analytic$eta_p_2_B
```

```
## [1] 0.0001562256
```

```
power_analytic$eta_p_2_C
```

```
## [1] 0.001404275
```

```
power_analytic$eta_p_2_AB
```

```
## [1] 0.0001562256
```

```
power_analytic$eta_p_2_AC
```

```
## [1] 0.0001562256
```

```
power_analytic$eta_p_2_BC
```

```
## [1] 0.0001562256
```

```
power_analytic$eta_p_2_ABC
```

```
## [1] 0.01855544
```

The power for interactions depends on Cohen's *f*, the alpha level, the sample size, and the degrees of freedom.

```
# With 2x2x2 designs,
# the names for paired comparisons can become very long.
# So here the sample size abbreviate terms
# Size, Color, and Cognitive Load, have values:
# b = big, s = small, g = green,
# r = red, pres = present, abs = absent.
labelnames <- c("Size", "b", "s", "x", "Color", "g", "r",
                "Load", "pres", "abs") #
design_result <- ANOVA_design(design = "3b*2b*2b",
                              n = 15,
                              mu = c(20, 0, 0, 0, 0,
                                     0, 0, 0, 0, 0, 0, 20),

                              sd = 20,
                              labelnames = labelnames)

# Power based on exact simulations
exact_result <- ANOVA_exact(design_result,
                            verbose = FALSE)
```

Table 5.9: Exact ANOVA Result

	power	partial_eta_squared	cohen_f	non_centrality
Size	26.92604	0.0146628	0.1219875	2.5
Color	5.00000	0.0000000	0.0000000	0.0
Load	5.00000	0.0000000	0.0000000	0.0
Size:Color	67.93217	0.0427350	0.2112886	7.5
Size:Load	67.93217	0.0427350	0.2112886	7.5
Color:Load	60.38579	0.0289017	0.1725164	5.0
Size:Color:Load	26.92604	0.0146628	0.1219875	2.5

```
#Analytical power calculation
power_analytic <- power_threeway_between(design_result)
```

```
## Warning in sqrt(f_2_ABC): NaNs produced
```

```
## Warning in pf(F_critical_ABC, df_ABC, df_error, lambda_ABC, lower.tail = FALSE):
## NaNs produced
```

```
power_analytic$power_A
```

```
## [1] 5
```

```
power_analytic$power_B
```

```
## [1] 5
```

```
power_analytic$power_C
```

```
## [1] 48.6496
```

```
power_analytic$power_AB
```

```
## [1] 34.7961
```

```
power_analytic$power_AC
```

```
## [1] 67.97466
```

```
power_analytic$power_BC
```

```
## [1] 91.55713
```

```
power_analytic$power_ABC
```

```
## [1] NaN
```

```
power_analytic$eta_p_2_A
```

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

```
power_analytic$Cohen_f_A
```

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

We see that a pattern of means of 0, 0, 0, 0, 0, 0, 0, 20 for a 2x2x2 interaction equals a Cohen's f of 0.25.

```
labelnames <- c("Size", "b", "s", "Color", "g", "r")
design_result <- ANOVA_design(design = "2b*2b",
                              n = 10,
                              mu = c(0, 0, 0, 10),
                              sd = 10,
                              labelnames = labelnames)

# Power based on exact simulations
exact_result <- ANOVA_exact(design_result,
                            verbose = FALSE)
```

Table 5.10: Exact ANOVA Result

	power	partial_eta_squared	cohen_f	non_centrality
Size	33.71329	0.0649351	0.2635231	2.5
Color	33.71329	0.0649351	0.2635231	2.5
Size:Color	33.71329	0.0649351	0.2635231	2.5

```
#Analytical power calculation
power_analytic <- power_twoway_between(design_result)
power_analytic$power_A
```

```
## [1] 33.71329
```

```
power_analytic$eta_p_2_A
```

```
## [1] 0.05882353
```

```
power_analytic$Cohen_f_A
```

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

Cohen's f is twice as large for a 2x2 design with the same mean value in one of four cells. In a 2 factor between design.

```
labelnames <- c("Size", "b", "s")
design_result <- ANOVA_design(design = "2b",
                              n = 10,
                              mu = c(0, 5),
                              sd = 10,
                              labelnames = labelnames)
```

```
# Power based on exact simulations
exact_result <- ANOVA_exact(design_result,
                            verbose = FALSE)
```

Table 5.11: Exact ANOVA Result

	power	partial_eta_squared	cohen_f	non_centrality
Size	18.50957	0.0649351	0.2635231	1.25

```
#Analytical power calculation
power_analytic <- power_oneway_between(design_result)
power_analytic$power
```

```
## [1] 18.50957
```

```
power_analytic$eta_p_2
```

```
## [1] 0.05882353
```

```
power_analytic$Cohen_f
```

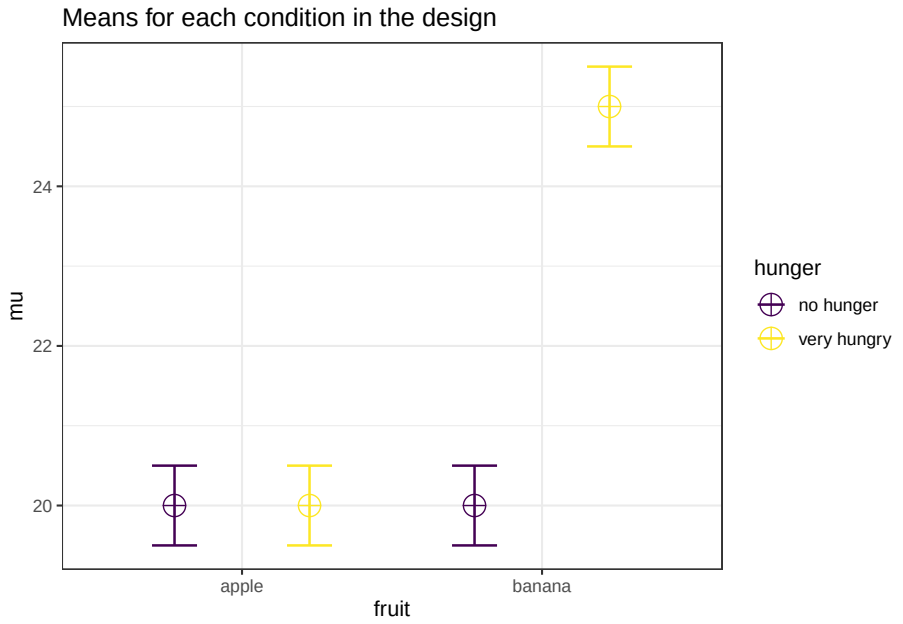
```
## [1] 0.25
```

Chapter 6

The 'No-Way' Interactions

In the 17th Data Colada blog post titled No-way Interactions Simonsohn (2014) discusses how a moderated interaction (the effect is there in one condition, but disappears in another condition) requires at least twice as many subjects per cell as a study that simply aims to show the simple effect. For example, see the plot below. Assume the score on the vertical axis is desire for fruit, as a function of the fruit that is available (an apple or a banana) and how hungry people are (not, or very). We see there is a difference between the participants desire for a banana compared to an apple, but only for participants who are very hungry. The point that is made is that you need twice as many participants in each cell to have power for the interaction, as you need for the simple effect.

```
string <- "2b*2b"
n <- 20
# All means are equal - so there is no real difference.
# Enter means in the order that matches the labels below.
mu <- c(20, 20, 20, 25)
sd <- 0.5
labelnames <- c("fruit", "apple", "banana",
                "hunger", "no hunger", "very hungry") #
# the label names should be in the order of the means specified above.
design_result <- ANOVA_design(design = string,
                    n = n,
                    mu = mu,
                    sd = sd,
                    labelnames = labelnames,
                    plot = TRUE)
```



We can reproduce the simulations in the Data Colada blog post, using the original code.

```
#R-Code
#
#Written by Uri Simonsohn, March 2014
#
#
#In DataColada[17]
# I propose that 2x2 interaction studies need 2x the sample size
#http://datacolada.org/2014/03/10/17-no-way-interactions
#In a companion ,pdf I show the simple math behind it
#
#
#Simulations are often more persuasive than math, so here it goes.
#I run simulations that compute power for 2 and 4 cell design,
# the latter testing the interaction
#################################################################################
#Create function that computes power of Studies 1 and 2,
# where Study 1  has 2 cells and tests a simple effect
#and Study 2 has 4 cells and tests the interaction
colada17 = function(d1,d2,n1,n2,simtot)
  {
  #n1: sample size, per cell, study 1
  #n2: sample size, per cell, study 2
```

```r
#d1: simple effect M1-M2
#d2: moderated effect M3-M4,
# full elimination of effect implies d2=0
#simtot: how many simulations to run
#Here we will store results
    p1 = c()    #p-values for Study 1
    p2 = c()    #p-values for Study 2
for (i in 1:simtot) {
  #draw data 4 samples
  y1 = rnorm(n = max(n1, n2), mean = d1)
  y2 = rnorm(n = max(n1, n2))
  y3 = rnorm(n = max(n1, n2), mean = d2)
  y4 = rnorm(n = max(n1, n2))

  #GET DATA READY FOR ANOVA
  y = c(y1, y2, y3, y4)          #the d.v.
  nrep = rep(n2, 4)
  A = rep(c(1, 1, 0, 0), times = nrep)
  B = rep(c(1, 0, 1, 0), times = nrep)

  #STUDY 1
    #Do a t-test on the first n1 observations
    p1.k = t.test(y1[1:n1], y2[1:n1], var.equal = TRUE)$p.value

  #STUDY 2
    #Do anova, keep p-value of the interaction
    p2.k = anova(lm(y ~ A * B))["A:B", "Pr(>F)"]

    #Store the results
    p1 = c(p1, p1.k)
    p2 = c(p2, p2.k)

    }

#What share off comparisons are significant
    #Simple test using estimate of variance from 2 cells only
    power1 = sum(p1 <= .05) / simtot
    #Interaction
    power2 = sum(p2 <= .05) / simtot

  cat("\nStudy 1 is powered to:",round(power1,2))
  cat("\nStudy 2 is powered to:",round(power2,2))

  }
```

```
#Same power for 2n regardless of n and d
```

```
  colada17(simtot = 2000, n1 = 20, n2 = 40, d1 = 1, d2 = 0)
```

```
##
## Study 1 is powered to: 0.85
```

```
##
## Study 2 is powered to: 0.89
```

```
  colada17(simtot = 2000, n1 = 50, n2 = 100, d1 = .3, d2 = 0)
```

```
##
## Study 1 is powered to: 0.33
```

```
##
## Study 2 is powered to: 0.32
```

```
  colada17(simtot = 2000, n1 = 150, n2 = 300, d1 = .25, d2 = 0)
```

```
##
## Study 1 is powered to: 0.56
```

```
##
## Study 2 is powered to: 0.58
```

```
#Need 4n if effect is 70% attenuated
  colada17(simtot = 2000, n1 = 25, n2 = 100, d1 = .5, d2 = .3 * .5)
```

```
##
## Study 1 is powered to: 0.4
```

```
##
## Study 2 is powered to: 0.4
```

```
  colada17(simtot = 2000, n1 = 50, n2 = 200, d1 = .5, d2 = .3 * .5)
```

```
##
## Study 1 is powered to: 0.71
```

```
##
## Study 2 is powered to: 0.68
```

```
  colada17(simtot = 2000, n1 = 22, n2 = 88, d1 = .41, d2 = .3 * .41)
```

```
##
## Study 1 is powered to: 0.28
```

```
##
## Study 2 is powered to: 0.25
```

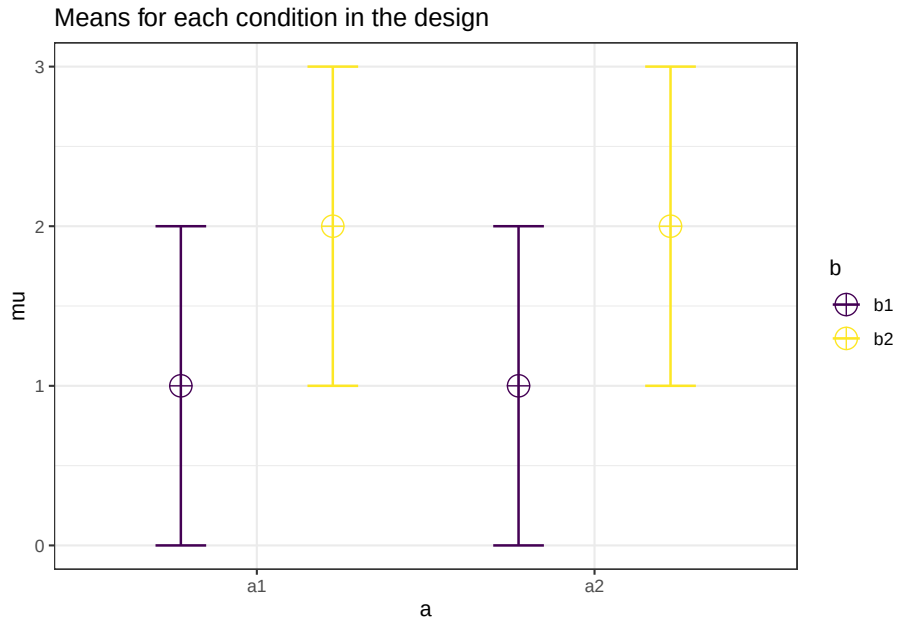
```
#underpowered if run with the same n
colada17(simtot = nsims, n1 = 20, n2 = 20, d1 = 1, d2 = 0)
```

```
##
## Study 1 is powered to: 0.87
```

```
##
## Study 2 is powered to: 0.59
```

And we can reproduce the results using the `ANOVA_exact` function.

```
#Study 1
string <- "2b*2b"
n <- 10
# All means are equal - so there is no real difference.
# Enter means in the order that matches the labels below.
mu <- c(1, 2, 1, 2)
sd <- 1

# the label names should be in the order of the means specified above.
design_result <- ANOVA_design(design = string,
                   n = n,
                   mu = mu,
                   sd = sd,
                   plot = TRUE)
```



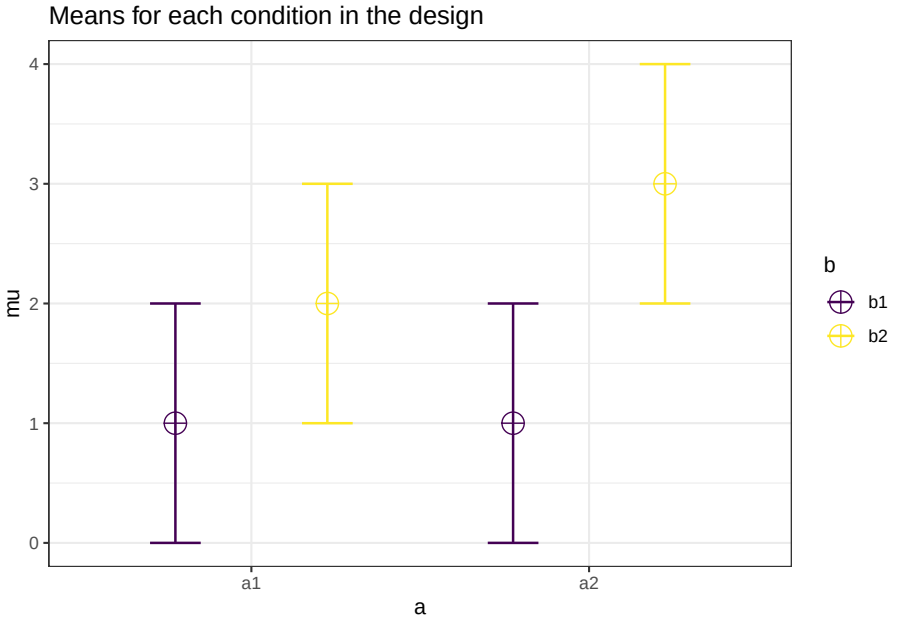

```
alpha_level <- 0.05 #We set the alpha level at 0.05.

exact_result <- ANOVA_exact(design_result,
                            alpha_level = alpha_level,
                            verbose = FALSE)

knitr::kable(exact_result$main_results[2,],
             caption = "ANOVA Results")%>%
  kable_styling(latex_options = "hold_position")
```

Table 6.1: ANOVA Results

	power	partial_eta_squared	cohen_f	non_centrality
b	86.79843	0.2173913	0.5270463	10

```
#Study 2
string <- "2b*2b"
n <- 20
# All means are equal - so there is no real difference.
# Enter means in the order that matches the labels below.
mu <- c(1, 2, 1, 3)
sd <- 1
```

```
# the label names should be in the order of the means specified above.
design_result <- ANOVA_design(design = string,
                      n = n,
                      mu = mu,
                      sd = sd,
                      plot = TRUE)
```




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

exact_result <- ANOVA_exact(design_result,
                            alpha_level = alpha_level,
                            verbose = FALSE)

knitr::kable(exact_result$main_results[3,],
             caption = "ANOVA Results")%>%
  kable_styling(latex_options = "hold_position")
```

Table 6.2: ANOVA Results

	power	partial_eta_squared	cohen_f	non_centrality
a:b	59.78655	0.0617284	0.2564946	5

We see we get the same power for the anova_fruit:hunger interaction and for the

simple effect p_fruit_apple_hunger_very hungry_fruit_banana_hunger_very hungry as the simulations by Uri Simonsohn in his blog post.

```
#Same power for 2n regardless of n and d
colada17(simtot = 10000, n1 = 20, n2 = 40, d1 = 1, d2 = 0)

colada17(simtot = 10000, n1 = 20, n2 = 40, d1 = 1, d2 = 0)
```

```
##
## Study 1 is powered to: 0.87
```

```
##
## Study 2 is powered to: 0.88
```

```
colada17(simtot = 10000, n1 = 50, n2 = 100, d1 = .3, d2 = 0)
```

```
##
## Study 1 is powered to: 0.32
```

```
##
## Study 2 is powered to: 0.33
```

```
colada17(simtot = 10000, n1 = 150, n2 = 300, d1 = .25, d2 = 0)
```

```
##
## Study 1 is powered to: 0.58
```

```
##
## Study 2 is powered to: 0.58
```

We can also reproduce the last example by adjusting the means and standard deviation. With 150 people, and a Cohen's d of 0.25 (the difference is 5, the sd 20, so $5/20 = 0.25$) we should reproduce the power for the simple effect.

```
string <- "2b*2b"
n <- 150
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 <- 20
labelnames <- c("fruit", "apple", "banana",
                "hunger", "no hunger", "very hungry") #
# the label names should be in the order of the means specified above.
```

```
design_result <- ANOVA_design(design = string,
                   n = n,
                   mu = mu,
                   sd = sd,
                   labelnames = labelnames)
alpha_level <- 0.05 #We set the alpha level at 0.05.

exact_result <- ANOVA_exact(design_result,
                            alpha_level = alpha_level,
                            verbose = FALSE)

knitr::kable(exact_result$main_results[2,],
             caption = "ANOVA Results")%>%
  kable_styling(latex_options = "hold_position")
```

Table 6.3: ANOVA Results

	power	partial_eta_squared	cohen_f	non_centrality
hunger	33.32955	0.0039171	0.0627094	2.34375

And changing the sample size to 300 should reproduce the power for the interaction in the ANOVA.

```
string <- "2b*2b"
n <- 300
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 <- 20
labelnames <- c("fruit", "apple", "banana",
                "hunger", "no hunger", "very hungry") #
# the label names should be in the order of the means specified above.
design_result <- ANOVA_design(design = string,
                   n = n,
                   mu = mu,
                   sd = sd,
                   labelnames = labelnames)
alpha_level <- 0.05 #We set the alpha level at 0.05.

exact_result <- ANOVA_exact(design_result, alpha_level = alpha_level)
```

```
## Power and Effect sizes for ANOVA tests
##                 power partial_eta_squared cohen_f non_centrality
## fruit         58.0592              0.0039  0.0626         4.6875
```

```
## hunger       58.0592              0.0039  0.0626         4.6875
## fruit:hunger 58.0592              0.0039  0.0626         4.6875
##
## Power and Effect sizes for pairwise comparisons (t-tests)
##                                                                    power
## p_fruit_apple_hunger_no hunger_fruit_apple_hunger_very hungry     5.00
## p_fruit_apple_hunger_no hunger_fruit_banana_hunger_no hunger      5.00
## p_fruit_apple_hunger_no hunger_fruit_banana_hunger_very hungry   86.37
## p_fruit_apple_hunger_very hungry_fruit_banana_hunger_no hunger    5.00
## p_fruit_apple_hunger_very hungry_fruit_banana_hunger_very hungry 86.37
## p_fruit_banana_hunger_no hunger_fruit_banana_hunger_very hungry  86.37
##                                                                  effect_size
## p_fruit_apple_hunger_no hunger_fruit_apple_hunger_very hungry           0.00
## p_fruit_apple_hunger_no hunger_fruit_banana_hunger_no hunger            0.00
## p_fruit_apple_hunger_no hunger_fruit_banana_hunger_very hungry          0.25
## p_fruit_apple_hunger_very hungry_fruit_banana_hunger_no hunger          0.00
## p_fruit_apple_hunger_very hungry_fruit_banana_hunger_very hungry        0.25
## p_fruit_banana_hunger_no hunger_fruit_banana_hunger_very hungry         0.25
```

```
knitr::kable(exact_result$main_results[3,],
             caption = "ANOVA Results")%>%
  kable_styling(latex_options = "hold_position")
```

Table 6.4: ANOVA Results

	power	partial_eta_squared	cohen_f	non_centrality
fruit:hunger	58.05922	0.003904	0.0626044	4.6875

Now if we look at the power analysis table for the last simulation, we see that the power for the ANOVA is the same for the main effect of fruit, the main effect of hunger, and the main effect of the interaction. All the effect sizes are equal as well. We can understand why if we look at the means in a 2x2 table:

```
mean_mat <- t(matrix(mu,
                     nrow = 2,
                     ncol = 2)) #Create a mean matrix
rownames(mean_mat) <- c("apple", "banana")
colnames(mean_mat) <- c("no hunger", "very hungry")
mean_mat
```

```
##        no hunger very hungry
## apple         20          20
## banana        20          25
```

The first main effect tests the marginal means if we sum over rows, 22.5 vs 20.

```
rowMeans(mean_mat)
```

```
##  apple banana
##   20.0   22.5
```

The second main effect tests the marginal means over the rows, which is also 22.5 vs 20.

```
colMeans(mean_mat)
```

```
##   no hunger very hungry
##        20.0        22.5
```

The interaction tests whether the average effect of hunger on liking fruit differs in the presence of bananas. In the presence of bananas the effect of hunger on the desireability of fruit is 5 scalepoints. The average effect (that we get from the marginal means) of hunger on fruit desireability is 2.5 (22.5-20). In other words, the interaction tests whether the difference effect between hunger and no hunger is different in the presence of an apple versus in the presence of a banana.

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.

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

```
b2 <- mean_mat[2,2] - (mean(mean_mat) +
                         (mean(mean_mat[2,]) - mean(mean_mat)) +
                         (mean(mean_mat[,2]) - mean(mean_mat)))

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

The sum of squares is dependent on the sample size, as can be seen in the code above. 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.

Interactions can have more power than main effects if the effect size of the interaction is larger than the effect size of the main effects. An example of this is a cross-over interaction. For example, let's take a 2x2 matrix of means with a crossover interaction:

```
mu <- c(25, 20, 20, 25)
mean_mat <- t(matrix(mu,
                     nrow = 2,
                     ncol = 2)) #Create a mean matrix
rownames(mean_mat) <- c("apple", "banana")
colnames(mean_mat) <- c("no hunger", "very hungry")
mean_mat
```

```
##        no hunger very hungry
## apple         25          20
## banana        20          25
```

Neither of the main effects is now significant, as the marginal means are 22.5 vs 22.5 for both main effects. The interaction is much stronger, however. We are testing whether the average effect of hunger on the desireability of fruit is different in the presence of bananas. Since the average effect is 0, and the effect of hunger on the desireability of bananas is 5, so the effect size is now twice as large.

```
string <- "2b*2b"
n <- 300
mu <- c(25, 20, 20, 25) #All means are equal - so there is no real difference.
# Enter means in the order that matches the labels below.
sd <- 20
labelnames <- c("fruit", "apple", "banana",
                "hunger", "no hunger", "very hungry") #
# the label names should be in the order of the means specified above.
```

```
design_result <- ANOVA_design(design = string,
                   n = n,
                   mu = mu,
                   sd = sd,
                   labelnames = labelnames)
alpha_level <- 0.05 #We set the alpha level at 0.05.

exact_result <- ANOVA_exact(design_result,
                            alpha_level = alpha_level,
                            verbose = FALSE)

knitr::kable(exact_result$main_results,
             caption = "ANOVA Results")%>%
  kable_styling(latex_options = "hold_position")
```

Table 6.5: ANOVA Results

	power	partial_eta_squared	cohen_f	non_centrality
fruit	5.00000	0.0000000	0.0000000	0.00
hunger	5.00000	0.0000000	0.0000000	0.00
fruit:hunger	99.10259	0.0154353	0.1252089	18.75

We can also reproduce the power analysis using the analytic functions in Superpower:

```
power_analytic <- power_twoway_between(design_result)
power_analytic$power_A
```

```
## [1] 5
```

```
power_analytic$power_B
```

```
## [1] 5
```

```
power_analytic$power_AB
```

```
## [1] 99.10259
```

Chapter 7

Effect of Varying Designs on Power

```
load("data/variation_data.RData")
```

Researchers might consider what the effects on the statistical power of their design is, when they add participants. Participants can be added to an additional condition, or to the existing design.

In a one-way ANOVA adding a condition means, for example, going from a 1x2 to a 1x3 design. For example, in addition to a control and intensive training condition, we add a light training condition.

```
string <- "2b"
n <- 50
#All means are equal - so there is no real difference.
mu <- c(80, 86)
sd <- 10
labelnames <- c("Condition", "control", "intensive_training")
design_result <- ANOVA_design(design = string,
                    n = n,
                    mu = mu,
                    sd = sd,
                    labelnames = labelnames)
# Power for the given N in the design_result
power_oneway_between(design_result)$power
```

```
## [1] 84.38754
```

```
power_oneway_between(design_result)$Cohen_f
```

```
## [1] 0.3
```

```
power_oneway_between(design_result)$eta_p_2
```

```
## [1] 0.08256881
```

```
simulation_result <- ANOVA_power(design_result,
                                 alpha_level = alpha_level,
                                 nsims = nsims,
                                 verbose = FALSE)
```

Table 7.1: Simulated ANOVA Result

	power	effect_size
anova_Condition	84.15	0.0911796

```
exact_result <- ANOVA_exact(design_result,
                            alpha_level = alpha_level,
                            verbose = FALSE)
```

Table 7.2: Exact ANOVA Result

	power	partial_eta_squared	cohen_f	non_centrality
Condition	84.38754	0.0841121	0.3030458	9

We now addd a condition. Let's assume the 'light training' condition falls in between the other two means.

And we can see power across sample sizes

```
# Plot power curve (from 5 to 100)
plot_power(design_result, max_n = 100)
```

```
string <- "3b"
n <- 50
mu <-
c(80, 83, 86) #All means are equal - so there is no real difference.
sd <- 10
```

```
labelnames <-
c("Condition",
"control",
"light_training",
"intensive_training")
design_result <- ANOVA_design(
design = string,
n = n,
mu = mu,
sd = sd,
labelnames = labelnames
)
# Power for the given N in the design_result
power_oneway_between(design_result)$power
```

```
## [1] 76.16545
```

```
power_oneway_between(design_result)$Cohen_f
```

```
## [1] 0.244949
```

```
power_oneway_between(design_result)$eta_p_2
```

```
## [1] 0.05660377
```

```
exact_result <- ANOVA_exact(design_result,
                            alpha_level = alpha_level,
                            verbose = FALSE)
```

Table 7.3: Exact ANOVA Result

	power	partial_eta_squared	cohen_f	non_centrality
Condition	76.16545	0.0576923	0.2474358	9

We see that adding a condition that falls between the other two means reduces our power. Let's instead assume that the 'light training' condition is not different from the control condition. In other words, the mean we add is as extreme as one of the existing means.

```
# Plot power curve (from 5 to 100)
plot_power(design_result, max_n = 100)
```

```
string <- "3b"
n <- 50
mu <-
c(80, 80, 86) #All means are equal - so there is no real difference.
sd <- 10
labelnames <-
c("Condition",
"control",
"light_training",
"intensive_training")
design_result <- ANOVA_design(
design = string,
n = n,
mu = mu,
sd = sd,
labelnames = labelnames
)

# Power for the given N in the design_result
power_oneway_between(design_result)$power
```

```
## [1] 87.62941
```

```
power_oneway_between(design_result)$Cohen_f
```

```
## [1] 0.2828427
```

```
power_oneway_between(design_result)$eta_p_2
```

```
## [1] 0.07407407
```

Now power has increased. This is not always true. The power is a function of many factors in the design, incuding the effect size (Cohen's f) and the total sample size (and the degrees of freedom and number of groups). But as we will see below, as we keep adding conditions, the power will reduce, even if initially, the power might increase.

```
# Plot power curve (from 5 to 100)
plot_power(design_result,max_n = 100)
```

It helps to think of these different designs in terms of either partial eta-squared, or Cohen's f (the one can easily be converted into the other).

```r
#Two groups
mu <- c(80, 86)
sd = 10
n <- 50 #sample size per condition
mean_mat <- t(matrix(mu,
nrow = 2,
ncol = 1)) #Create a mean matrix
# Using the sweep function to remove rowmeans from the matrix
mean_mat_res <- sweep(mean_mat, 2, rowMeans(mean_mat))
mean_mat_res
```

```
##      [,1] [,2]
## [1,]   -3    3
```

```r
MS_a <- n * (sum(mean_mat_res ^ 2) / (2 - 1))
MS_a
```

```
## [1] 900
```

```r
SS_A <- n * sum(mean_mat_res ^ 2)
SS_A
```

```
## [1] 900
```

```r
MS_error <- sd ^ 2
MS_error
```

```
## [1] 100
```

```r
SS_error <- MS_error * (n * 2)
SS_error
```

```
## [1] 10000
```

```r
eta_p_2 <- SS_A / (SS_A + SS_error)
eta_p_2
```

```
## [1] 0.08256881
```

```
f_2 <- eta_p_2 / (1 - eta_p_2)
f_2
```

```
## [1] 0.09
```

```
Cohen_f <- sqrt(f_2)
Cohen_f
```

```
## [1] 0.3
```

```
#Three groups
mu <- c(80, 83, 86)
sd = 10
n <- 50
mean_mat <- t(matrix(mu,
nrow = 3,
ncol = 1)) #Create a mean matrix
# Using the sweep function to remove rowmeans from the matrix
mean_mat_res <- sweep(mean_mat, 2, rowMeans(mean_mat))
mean_mat_res
```

```
##      [,1] [,2] [,3]
## [1,]   -3    0    3
```

```
MS_a <- n * (sum(mean_mat_res ^ 2) / (3 - 1))
MS_a
```

```
## [1] 450
```

```
SS_A <- n * sum(mean_mat_res ^ 2)
SS_A
```

```
## [1] 900
```

```
MS_error <- sd ^ 2
MS_error
```

```
## [1] 100
```

```
SS_error <- MS_error * (n * 3)
SS_error
```

```
## [1] 15000
```

```
eta_p_2 <- SS_A / (SS_A + SS_error)
eta_p_2
```

```
## [1] 0.05660377
```

```
f_2 <- eta_p_2 / (1 - eta_p_2)
f_2
```

```
## [1] 0.06
```

```
Cohen_f <- sqrt(f_2)
Cohen_f
```

```
## [1] 0.244949
```

The `SS_A` or the sum of squares for the main effect, is 900 for two groups, and the `SS_error` for the error term is 10000. When we add a group, `SS_A` is 900, and the `SS_error` is 15000. Because the added condition falls exactly on the grand mean (83), the sum of squared for this extra group is 0. In other words, it does nothing to increase the signal that there is a difference between groups. However, the sum of squares for the error, which is a function of the total sample size, is increased, which reduces the effect size. So, adding a condition that falls on the grand mean reduces the power for the main effect of the ANOVA. Obviously, adding such a group has other benefits, such as being able to compare the two means to a new third condition.

We already saw that adding a condition that has a mean as extreme as one of the existing groups also reduces the power. Let's again do the calculations step by step when the extra group has a mean as extreme as one of the two original conditions.

```
#Three groups
mu <- c(80, 80, 86)
sd = 10
n <- 50
mean_mat <- t(matrix(mu,
nrow = 3,
```

```
ncol = 1)) #Create a mean matrix
# Using the sweep function to remove rowmeans from the matrix
mean_mat_res <- sweep(mean_mat, 2, rowMeans(mean_mat))
mean_mat_res
```

```
##      [,1] [,2] [,3]
## [1,]   -2   -2    4
```

```
MS_a <- n * (sum(mean_mat_res ^ 2) / (3 - 1))
MS_a
```

```
## [1] 600
```

```
SS_A <- n * sum(mean_mat_res ^ 2)
SS_A
```

```
## [1] 1200
```

```
MS_error <- sd ^ 2
MS_error
```

```
## [1] 100
```

```
SS_error <- MS_error * (n * 3)
SS_error
```

```
## [1] 15000
```

```
eta_p_2 <- SS_A / (SS_A + SS_error)
eta_p_2
```

```
## [1] 0.07407407
```

```
f_2 <- eta_p_2 / (1 - eta_p_2)
f_2
```

```
## [1] 0.08
```

```
Cohen_f <- sqrt(f_2)
Cohen_f
```

```
## [1] 0.2828427
```

We see the sum of squares of the error stays the same - 15000 - because it is only determined by the standard error and the sample size, but not by the differences in the means. This is an increase of 5000 compared to the 2 group design. The sum of squares (the second component that determines the size of partial eta-squared) increases, which increases Cohen's *f*.

7.1 Within Designs

Now imagine our design described above was a within design. The means and sd remain the same. We collect 50 participants (instead of 100, or 50 per group, for the between design). Let's first assume the two samples are completely uncorrelated.

```
string <- "2w"
n <- 50
mu <-
c(80, 86) #All means are equal - so there is no real difference.
sd <- 10
labelnames <- c("Condition", "control", "intensive_training") #
design_result <- ANOVA_design(
design = string,
n = n,
mu = mu,
sd = sd,
labelnames = labelnames
)

power_oneway_within(design_result)$power
```

```
## [1] 83.66436
```

```
exact_result <- ANOVA_exact(design_result,
                            alpha_level = alpha_level,
                            verbose = FALSE)
```

We see power is ever so slightly less than for the between subject design. This is due to the loss in degrees of freedom, which is $2(n-1)$ for between designs, and

Table 7.4: Exact ANOVA Result

	power	partial_eta_squared	cohen_f	non_centrality
Condition	83.66436	0.1551724	0.4285714	9

$n - 1$ for within designs. But as the correlation increases, the power advantage of within designs becomes stronger.

```
string <- "3w"
n <- 50
mu <-
c(80, 83, 86) #All means are equal - so there is no real difference.
sd <- 10
labelnames <-
c("Condition",
"control",
"light_training",
"intensive_training") #

design_result <- ANOVA_design(
design = string,
n = n,
mu = mu,
sd = sd,
labelnames = labelnames
)

power_oneway_within(design_result)$power
```

```
## [1] 75.70841
```

```
exact_result <- ANOVA_exact(design_result,
                            alpha_level = alpha_level,
                            verbose = FALSE)
```

Table 7.5: Exact ANOVA Result

	power	partial_eta_squared	cohen_f	non_centrality
Condition	75.70841	0.0841121	0.3030458	9

When we add a a condition in a within design where we expect the mean to be identical to the grand mean, we again see that the power decreases. This

similarly shows that adding a condition that equals the grand mean to a within subject design does not come for free, but has a power cost.

```
n <- 30
sd <- 10
r <- 0.5
string <- "2w"
mu <- c(0, 5) #All means are equal - so there is no real difference.
labelnames <- c("Factor_A", "a1", "a2") #
design_result <-
ANOVA_design(
design = string,
n = n,
mu = mu,
sd = sd,
r = r,
labelnames = labelnames
)

power_oneway_within(design_result)$power
```

```
## [1] 75.39647
```

```
power_oneway_within(design_result)$Cohen_f
```

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

```
power_oneway_within(design_result)$Cohen_f_SPSS
```

```
## [1] 0.5085476
```

```
power_oneway_within(design_result)$lambda
```

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

```
power_oneway_within(design_result)$F_critical
```

```
## [1] 4.182964
```

```
string <- "3w"
mu <-
c(0, 0, 5) #All means are equal - so there is no real difference.
labelnames <- c("Factor_A", "a1", "a2", "a3")

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

power_oneway_within(design_result)$power
```

```
## [1] 79.37037
```

```
power_oneway_within(design_result)$Cohen_f
```

```
## [1] 0.2357023
```

```
power_oneway_within(design_result)$Cohen_f_SPSS
```

```
## [1] 0.4152274
```

```
power_oneway_within(design_result)$lambda
```

```
## [1] 10
```

```
power_oneway_within(design_result)$F_critical
```

```
## [1] 3.155932
```

```
string <- "4w"
mu <-
c(0, 0, 0, 5) #All means are equal - so there is no real difference.
labelnames <- c("Factor_A", "a1", "a2", "a3", "a4") #
design_result <-
ANOVA_design(
```

```
design = string,
n = n,
mu = mu,
sd = sd,
r = r,
labelnames = labelnames
)

power_oneway_within(design_result)$power
```

```
## [1] 79.40126
```

```
power_oneway_within(design_result)$Cohen_f
```

```
## [1] 0.2165064
```

```
power_oneway_within(design_result)$Cohen_f_SPSS
```

```
## [1] 0.3595975
```

```
power_oneway_within(design_result)$lambda
```

```
## [1] 11.25
```

```
power_oneway_within(design_result)$F_critical
```

```
## [1] 2.709402
```

```
string <- "5w"
mu <-
c(0, 0, 0, 0, 5)
#All means are equal - so there is no real difference.
labelnames <- c("Factor_A", "a1", "a2", "a3", "a4", "a5")

design_result <-
ANOVA_design(
design = string,
n = n,
mu = mu,
sd = sd,
r = r,
labelnames = labelnames
)
power_oneway_within(design_result)$power
```

```
## [1] 78.38682
```

```
power_oneway_within(design_result)$Cohen_f
```

```
## [1] 0.2
```

```
power_oneway_within(design_result)$Cohen_f_SPSS
```

```
## [1] 0.3216338
```

```
power_oneway_within(design_result)$lambda
```

```
## [1] 12
```

```
power_oneway_within(design_result)$F_critical
```

```
## [1] 2.44988
```

```
string <- "6w"
mu <- c(0, 0, 0, 0, 0, 5)
#All means are equal - so there is no real difference.
labelnames <- c("Factor_A", "a1", "a2", "a3", "a4", "a5", "a6") #
design_result <-
ANOVA_design(
design = string,
n = n,
mu = mu,
sd = sd,
r = r,
labelnames = labelnames
)
power_oneway_within(design_result)$power
```

```
## [1] 76.99592
```

```
power_oneway_within(design_result)$Cohen_f
```

```
## [1] 0.186339
```

```
power_oneway_within(design_result)$Cohen_f_SPSS
```

```
## [1] 0.2936101
```

```
power_oneway_within(design_result)$lambda
```

```
## [1] 12.5
```

```
power_oneway_within(design_result)$F_critical
```

```
## [1] 2.276603
```

```
string <- "7w"
mu <- c(0, 0, 0, 0, 0, 0, 5)
#All means are equal - so there is no real difference.
labelnames <- c("Factor_A", "a1", "a2", "a3",
                "a4", "a5", "a6", "a7")
design_result <-
ANOVA_design(
design = string,
n = n,
mu = mu,
sd = sd,
r = r,
labelnames = labelnames
)

power_oneway_within(design_result)$power
```

```
## [1] 75.4601
```

```
power_oneway_within(design_result)$Cohen_f
```

```
## [1] 0.1749636
```

```
power_oneway_within(design_result)$Cohen_f_SPSS
```

```
## [1] 0.2718301
```

```
power_oneway_within(design_result)$lambda
```

```
## [1] 12.85714
```

```
power_oneway_within(design_result)$F_critical
```

```
## [1] 2.151016
```

This set of designs where we increase the number of conditions demonstrates a common pattern where the power initially increases, but then starts to decrease. Again, the exact pattern (and when the power starts to decrease) depends on the effect size and sample size. Note also that the effect size (Cohen's f) decreases as we add conditions, but the increased sample size compensates for this when calculating power. When using power analysis software such as GPower (Faul et al., 2007), this is important to realize. You can't just power for a medium effect size, and then keep adding conditions under the assumption that the increased power you see in the program will become a reality. Increasing the number of conditions will reduce the effect size, and therefore, adding conditions will not automatically increase power (and might even decrease it).

Overall, the effect of adding conditions with an effect close to the grand mean reduces power quite strongly, and adding conditions with means close to the extreme of the current conditions will either slightly increase of decrease power.

Chapter 8

Error Control in Exploratory ANOVA

In a 2 x 2 x 2 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 I error in a single 2 x 2 x 2 ANOVA is $1 - (0.95)^7 = 30\%$.

```
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
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(design = string,
                   n = n,
                   mu = mu,
                   sd = sd,
                   labelnames = labelnames)
alpha_level <- 0.05
#We set the alpha level at 0.05.
```

```
simulation_result <- ANOVA_power(design_result,
                                 alpha_level = alpha_level,
                                 verbose = FALSE)
```

Table 8.1: Simulated ANOVA Result

	power	effect_size
anova_condition1	4.5	0.0023816
anova_condition2	5.3	0.0025314
anova_condition3	5.6	0.0026427
anova_condition1:condition2	5.2	0.0025443
anova_condition1:condition3	5.2	0.0025055
anova_condition2:condition3	6.6	0.0027795
anova_condition1:condition2:condition3	4.5	0.0024848

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 I error rate'. Because we have saved the power simulation in the 'simulation_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(simulation_result_8.1$sim_data[(1:7)]), 2,
    function(x) round(mean(ifelse(x < alpha_level, 1, 0) * 100),4))
```

```
##                         anova_condition1                         anova_condition2
##                                      4.5                                      5.3
##                         anova_condition3             anova_condition1:condition2
##                                      5.6                                      5.2
##              anova_condition1:condition3             anova_condition2:condition3
##                                      5.2                                      6.6
## anova_condition1:condition2:condition3
##                                      4.5
```

This is the type I 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 I error rate (e.g., a significant result for the three-way interaction, and for the main effect of condition 1) but such an ANOVA is counted only once. If we calculate this percentage from our simulations, we see the number is indeed very close to 1-(0.95)^7 = 30%.

```
sum(apply(as.matrix(simulation_result_8.1$sim_data[(1:7)]), 1,
    function(x)
      round(mean(ifelse(x < alpha_level, 1, 0) * 100),4)) > 0)/nsims*100
```

```
## [1] 3.22
```

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 I errors, which will not replicate if you try to build on them. Therefore, you need to control the type I 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 base 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.

```
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
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(design = string,
                  n = n,
                  mu = mu,
                  sd = sd,
                  labelnames = labelnames)
alpha_level <- 0.05
```

```
simulation_result <- ANOVA_power(design_result,
                                 alpha_level = alpha_level,
                                 p_adjust = p_adjust,
                                 verbose = FALSE)
```

```
sum(apply(as.matrix(simulation_result_8.2$sim_data[(1:7)]), 1,
    function(x)
      round(mean(ifelse(x < alpha_level, 1, 0) * 100),4)) > 0)/nsims*100
```

```
## [1] 0.66
```

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

Table 8.2: ANOVA Results

	power	effect_size
anova_condition1	1.2	0.0026148
anova_condition2	1.4	0.0026294
anova_condition3	0.6	0.0024602
anova_condition1:condition2	0.8	0.0026337
anova_condition1:condition3	0.9	0.0025830
anova_condition2:condition3	0.8	0.0025079
anova_condition1:condition2:condition3	1.1	0.0027855

Table 8.3: Pairwise Results

	power	effect_size
p_condition1_a_condition2_c_condition3_e_condition1_a_condition2_c_condition3_f	0.2	0.0091931
p_condition1_a_condition2_c_condition3_e_condition1_a_condition2_d_condition3_e	0.3	-0.0020647
p_condition1_a_condition2_c_condition3_e_condition1_a_condition2_d_condition3_f	0.1	0.0045378
p_condition1_a_condition2_c_condition3_e_condition1_b_condition2_c_condition3_e	0.2	0.0113580
p_condition1_a_condition2_c_condition3_e_condition1_b_condition2_c_condition3_f	0.2	0.0065425
p_condition1_a_condition2_c_condition3_e_condition1_b_condition2_d_condition3_e	0.2	-0.0060879
p_condition1_a_condition2_c_condition3_e_condition1_b_condition2_d_condition3_f	0.2	0.0029079
p_condition1_a_condition2_c_condition3_f_condition1_a_condition2_d_condition3_e	0.4	-0.0118354
p_condition1_a_condition2_c_condition3_f_condition1_a_condition2_d_condition3_f	0.1	-0.0041912
p_condition1_a_condition2_c_condition3_f_condition1_b_condition2_c_condition3_e	0.2	0.0022093
p_condition1_a_condition2_c_condition3_f_condition1_b_condition2_c_condition3_f	0.1	-0.0029714
p_condition1_a_condition2_c_condition3_f_condition1_b_condition2_d_condition3_e	0.1	-0.0155765
p_condition1_a_condition2_c_condition3_f_condition1_b_condition2_d_condition3_f	0.1	-0.0072470
p_condition1_a_condition2_d_condition3_e_condition1_a_condition2_d_condition3_f	0.2	0.0069110
p_condition1_a_condition2_d_condition3_e_condition1_b_condition2_c_condition3_e	0.3	0.0139610
p_condition1_a_condition2_d_condition3_e_condition1_b_condition2_c_condition3_f	0.1	0.0085668
p_condition1_a_condition2_d_condition3_e_condition1_b_condition2_d_condition3_e	0.1	-0.0037733
p_condition1_a_condition2_d_condition3_e_condition1_b_condition2_d_condition3_f	0.2	0.0054335
p_condition1_a_condition2_d_condition3_f_condition1_b_condition2_c_condition3_e	0.4	0.0066358
p_condition1_a_condition2_d_condition3_f_condition1_b_condition2_c_condition3_f	0.1	0.0012669
p_condition1_a_condition2_d_condition3_f_condition1_b_condition2_d_condition3_e	0.5	-0.0114851
p_condition1_a_condition2_d_condition3_f_condition1_b_condition2_d_condition3_f	0.6	-0.0020705
p_condition1_b_condition2_c_condition3_e_condition1_b_condition2_c_condition3_f	0.1	-0.0053269
p_condition1_b_condition2_c_condition3_e_condition1_b_condition2_d_condition3_e	0.2	-0.0179785
p_condition1_b_condition2_c_condition3_e_condition1_b_condition2_d_condition3_f	0.2	-0.0086868
p_condition1_b_condition2_c_condition3_f_condition1_b_condition2_d_condition3_e	0.3	-0.0129444
p_condition1_b_condition2_c_condition3_f_condition1_b_condition2_d_condition3_f	0.4	-0.0035748
p_condition1_b_condition2_d_condition3_e_condition1_b_condition2_d_condition3_f	0.1	0.0090723

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 I 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 (Hothorn et al., 2019).

Chapter 9

Analytic Power Functions

For some designs it is possible to calculate power analytically, using closed functions. Within the `Superpower` package we have included a number of these closed functions. As you will see below, each analytic function only serves a very narrow scenario while the simulations functions are much more flexible. In addition, we will compare these functions to other packages/software. Please note, that the analytic power functions are designed to reject designs that are not appropriate for the functions (i.e., a `3w` design will be rejected by the `power_oneway_between` function).

9.1 One-Way Between Subjects ANOVA

First, we can setup a one-way design with four levels, and perform a exact simulation power analysis with the `ANOVA_exact` function.

```
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.
sd <- 10
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(design = string,
                   n = n,
                   mu = mu,
                   sd = sd,
                   labelnames = labelnames)
```

```
exact_result <- ANOVA_exact(design_result,
                            alpha_level = alpha_level,
                            verbose = FALSE)
```

Table 9.1: Exact ANOVA Result

	power	partial_eta_squared	cohen_f	non_centrality
Factor_A	81.21291	0.0460792	0.2197842	11.4

We can also calculate power analytically with a `Superpower` function.

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

```
## [1] 81.21291
```

This is a generalized function for one-way ANOVA's for any number of groups. It is in part based on code from the `pwr2ppl` (Aberson, 2019) package (but Aberson's code allows for different n per condition, and different sd per condition).

```
pwr2ppl::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)
```

```
## Power  = 0.812 for eta-squared = 0.05
```

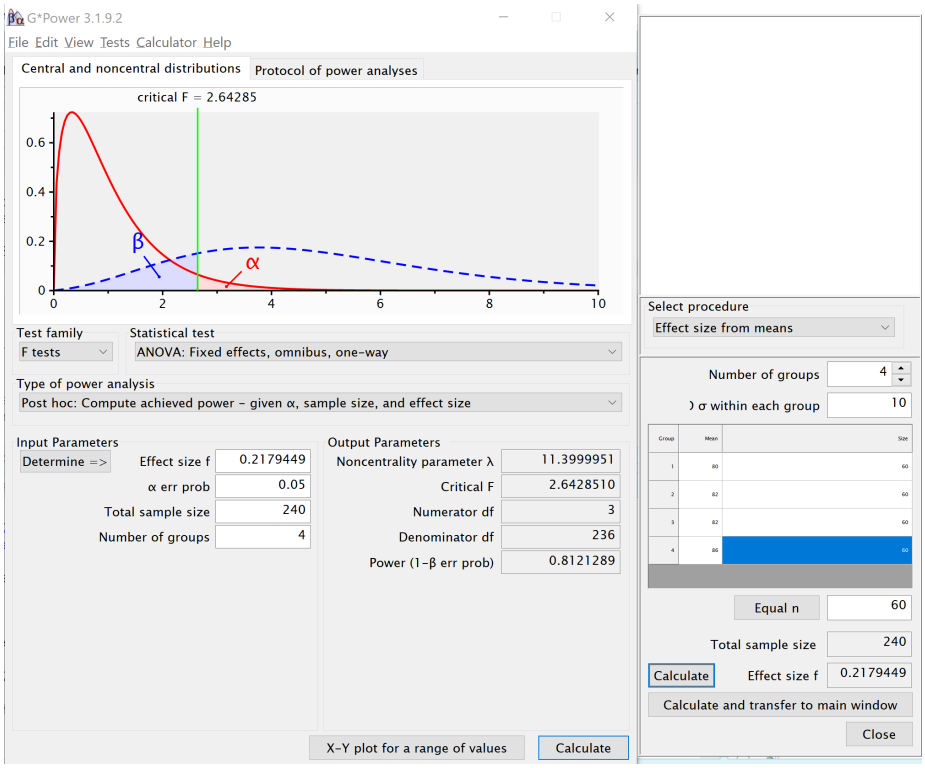
We can also use the function in the `pwr` package (Champely, 2018). Note that we need to calculate f to use this function, which is based on the means and sd, as illustrated in the formulas above.

```
pwr::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 (Faul et al., 2007) provides the option to calculate `f` from the means, sd and n for the cells. It can then be used to calculate power.



9.2 Two-way Between Subject Interaction

Now, we will setup a 2x2 between-subject ANOVA.

```
string <- "2b*2b"
n <- 20
mu <- c(20, 20, 20, 25)
# Enter means in the order that matches the labels below.
sd <- 5
```

```
labelnames <- c("A", "a1", "a2", "B", "b1", "b2")
# the label names should be in the order of the means specified above.

design_result <- ANOVA_design(design = string,
                   n = n,
                   mu = mu,
                   sd = sd,
                   labelnames = labelnames)



exact_result <- ANOVA_exact(design_result,
                            alpha_level = alpha_level,
                            verbose = FALSE)
```

Table 9.2: Exact ANOVA Result

	power	partial_eta_squared	cohen_f	non_centrality
A	59.78655	0.0617284	0.2564946	5
B	59.78655	0.0617284	0.2564946	5
A:B	59.78655	0.0617284	0.2564946	5

Now, let's use the analytic function `power_twoway_between`.

```
#using default alpha level of .05
power_res <- power_twoway_between(design_result)
power_res$power_A
```

```
## [1] 59.78655
```

```
power_res$power_B
```

```
## [1] 59.78655
```

```
power_res$power_AB
```

```
## [1] 59.78655
```

We can compare these results to (Aberson, 2019), as well.

```
pwr2ppl::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")
```

```
## Power for Main Effect Factor A = 0.598

## Power for Main Effect Factor B = 0.598

## Power for Interaction AxB = 0.598
```

9.3 3x3 Between Subject ANOVA

We can extend this function to a two-way design with 3 levels.

```
string <- "3b*3b"
n <- 20
#All means are equal - so there is no real difference.
mu <- c(20, 20, 20, 20, 20, 20, 20, 20, 25)
# Enter means in the order that matches the labels below.
sd <- 5
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(design = string,
                   n = n,
                   mu = mu,
                   sd = sd,
                   labelnames = labelnames)



exact_result <- ANOVA_exact(design_result,
                            alpha_level = alpha_level,
                            verbose = FALSE)
```

Table 9.3: Exact ANOVA Result

	power	partial_eta_squared	cohen_f	non_centrality
Factor_A	44.86306	0.0253325	0.1612169	4.444444
Factor_B	44.86306	0.0253325	0.1612169	4.444444
Factor_A:Factor_B	64.34127	0.0494132	0.2279952	8.888889

```
#using default alpha level of .05
power_res <- power_twoway_between(design_result)
power_res$power_A
```

```
## [1] 44.86306
```

```
power_res$power_B
```

```
## [1] 44.86306
```

```
power_res$power_AB
```

```
## [1] 64.34127
```

Chapter 10

Power Curve

Power is calculated for a specific value of an effect size, alpha level, and sample size. Because you often do not know the true effect size, it often makes more sense to think of the power curve as a function of the size of the effect. Although power curves could be constructed from Monte Carlo simulations (`ANOVA_power`) the `plot_power` function utilizes the `ANOVA_exact` function within its code because these "exact" simulations are much faster. The basic approach is to calculate power for a specific pattern of means, a specific effect size, a given alpha level, and a specific pattern of correlations. This is one example:

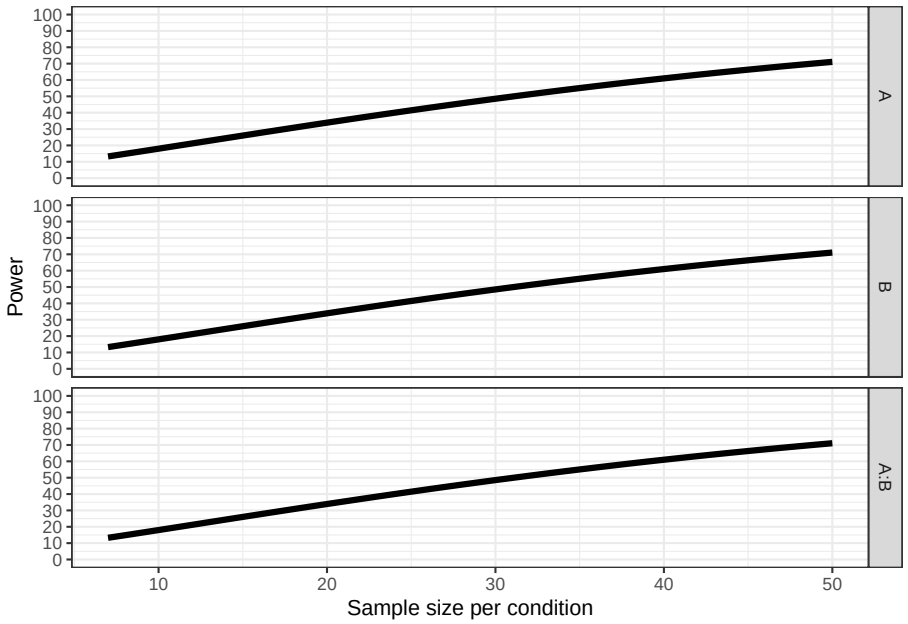
```
#2x2 design
string = "2w*2w"
mu = c(0,0,0,0.5)
n <- 20
sd <- 1
r <- 0.5
labelnames = c("A", "a1", "a2", "B", "b1", "b2")

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

exact_result <- ANOVA_exact(design_result,
                            alpha_level = alpha_level,
                            verbose = FALSE)
```

Table 10.1: Exact ANOVA Result

	power	partial_eta_squared	cohen_f	non_centrality
A	32.35926	0.1162791	0.3627381	2.5
B	32.35926	0.1162791	0.3627381	2.5
A:B	32.35926	0.1162791	0.3627381	2.5

We can make these calculations for a range of sample sizes, to get a power curve. We created a simple function that performs these calculations across a range of sample sizes (from n = 3 to max_, a variable you can specify in the function).

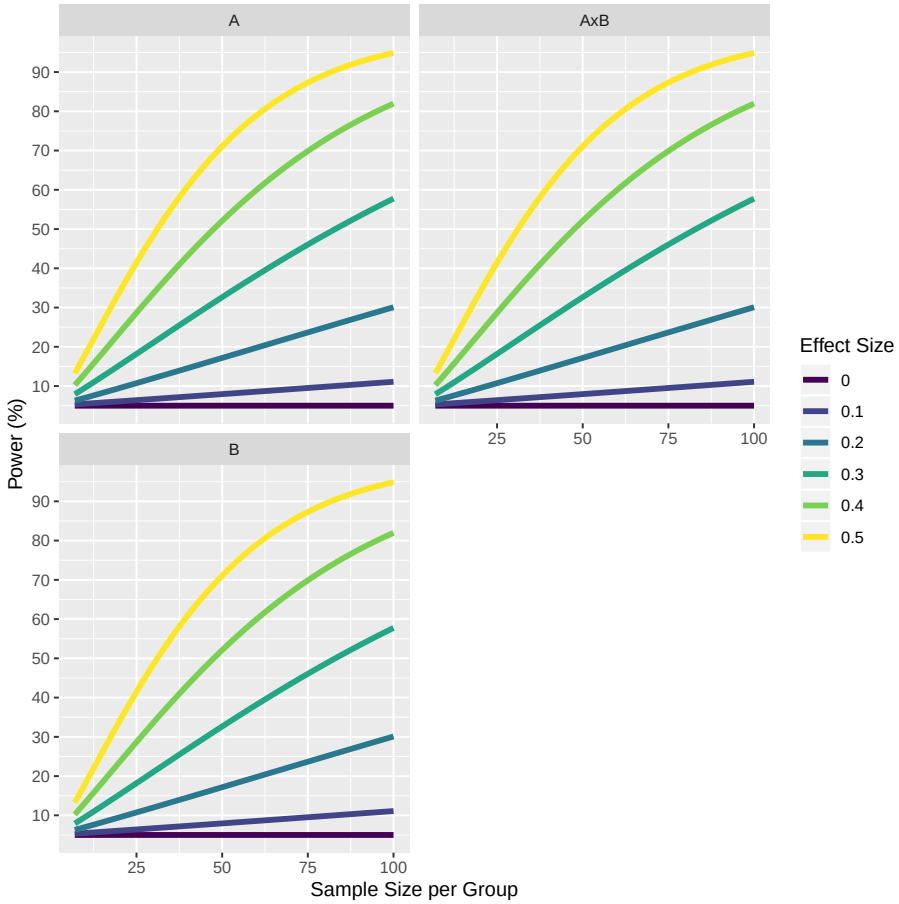
```
p_a <- plot_power(design_result,
                  max_n = 50)
p_a$plot_ANOVA
```



If we run many of these `plot_power` functions across small changes in the ANOVA_design we can compile a number of power curves that can be combined into a single plot. We do this below. The code to reprodcue these plots can be found on the GitHub repository for this book.

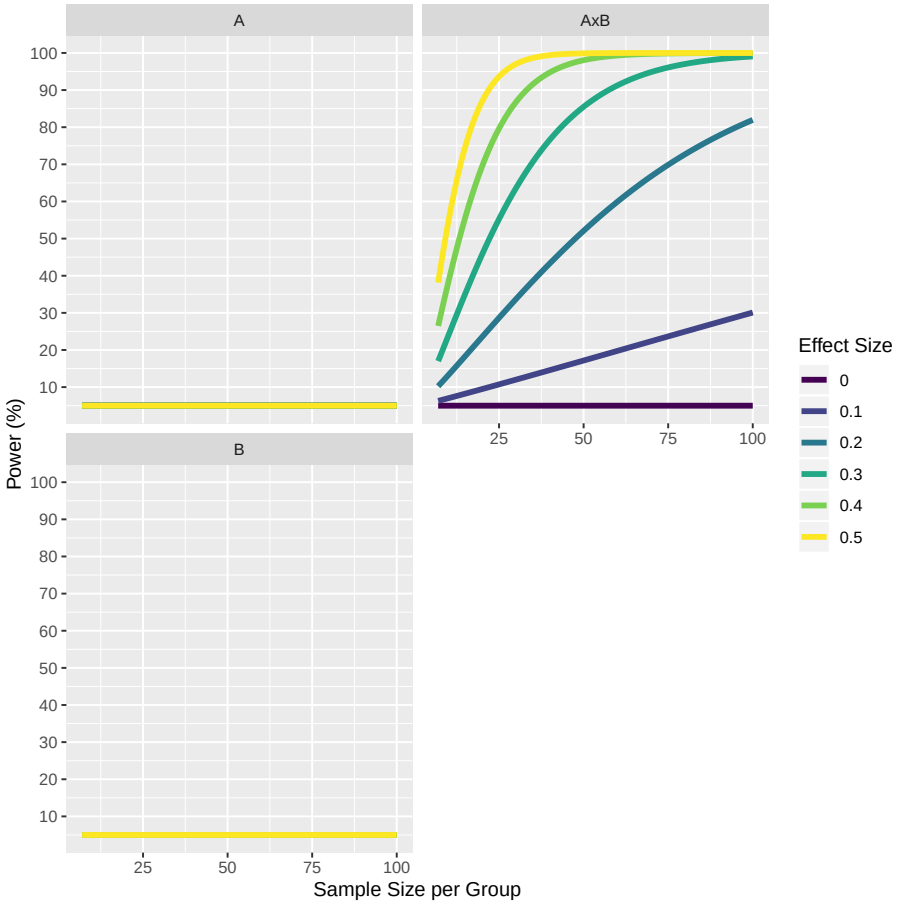
10.1 Explore increase in effect size for moderated interactions.

The design has means 0, 0, 0, 0, with one cell increasing by 0.1, up to 0, 0, 0, 0.5. The standard deviation is set to 1. The correlation between all variables is 0.5.



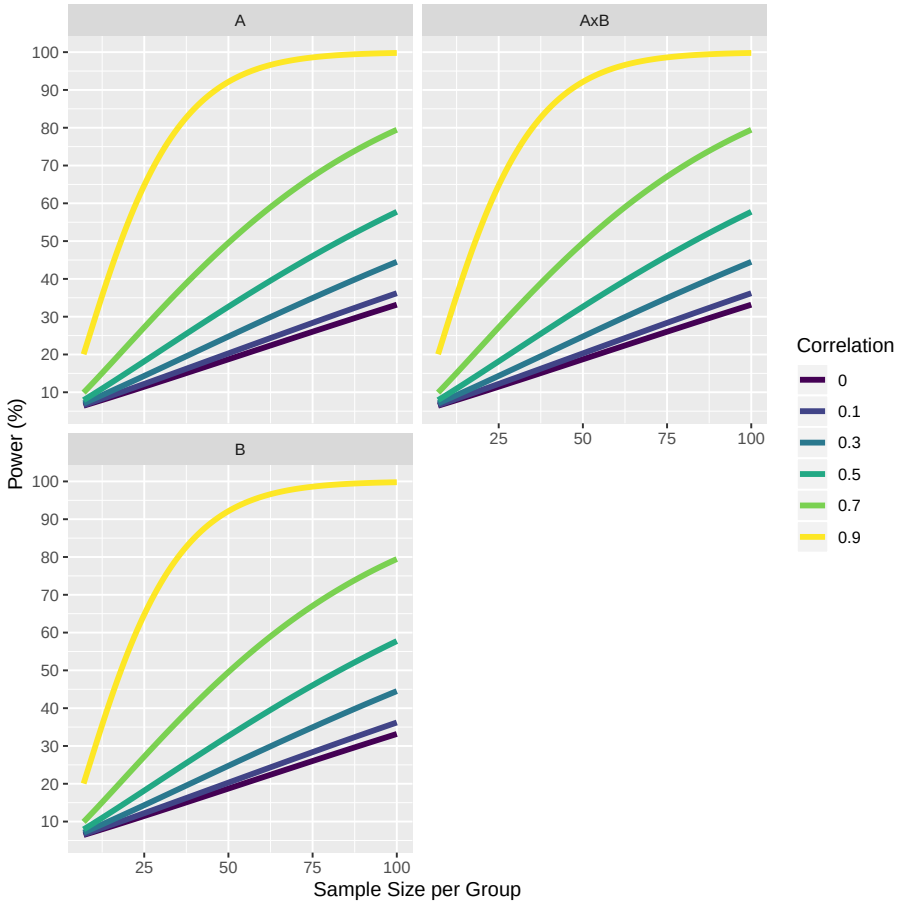
10.2 Explore increase in effect size for cross-over interactions.

The design has means 0, 0, 0, 0, with two cells increasing by 0.1, up to 0.5, 0, 0, 0.5. The standard deviation is set to 1. The correlation between all variables is 0.5.



10.3 Explore increase in correlation in moderated interactions.

The design has means 0, 0, 0, 0.3. The standard deviation is set to 1. The correlation between all variables increases from 0 to 0.9.

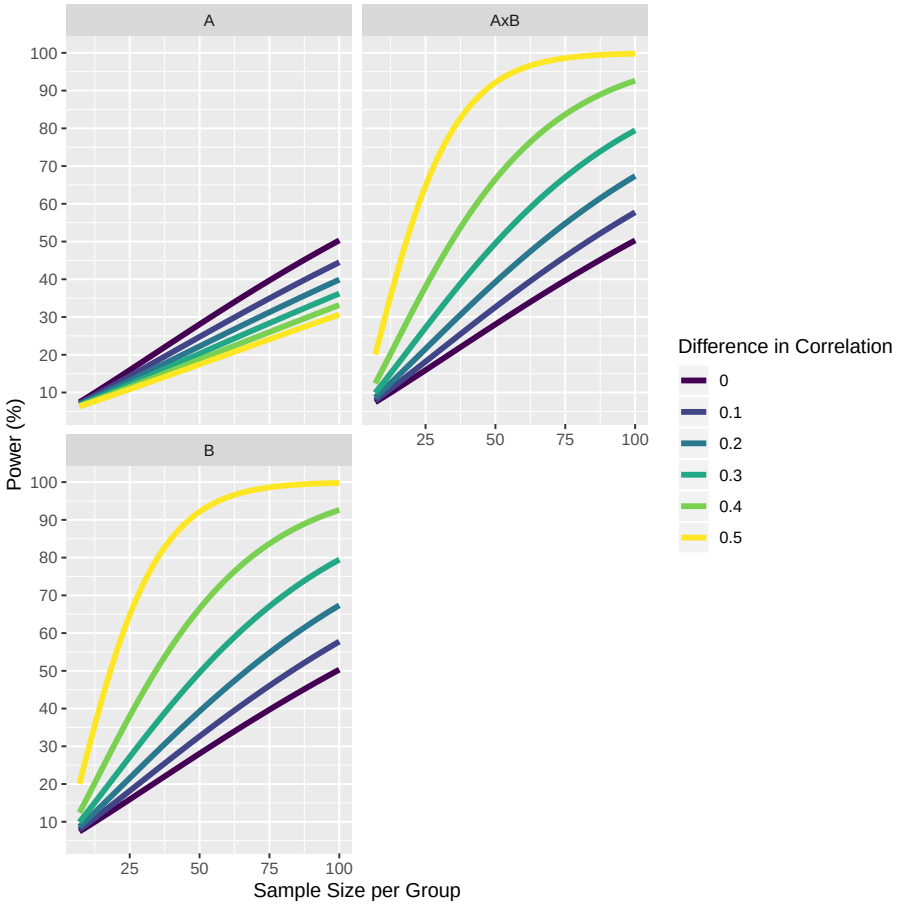


10.4 Increasing correlation in on factor decreases power in second factor

As Potvin and Schutz (2000) write:

> The more important finding with respect to the effect of r on power relates to the effect of the correlations associated with one factor on the power of the test of the main effect of the other factor. Specifically, if the correlations among the levels of B are larger than those within the AB matrix (i.e., r(B) - r(AB) > 0.0), there is a reduction in the power for the test of the A effect (and the test on B is similarly affected by the A correlations).

We see this in the plots below. As the correlation of the A factor increases from 0.4 to 0.9, we see the power for the main effect decreases.



Appendix 1: Direct Comparison to pwr2ppl

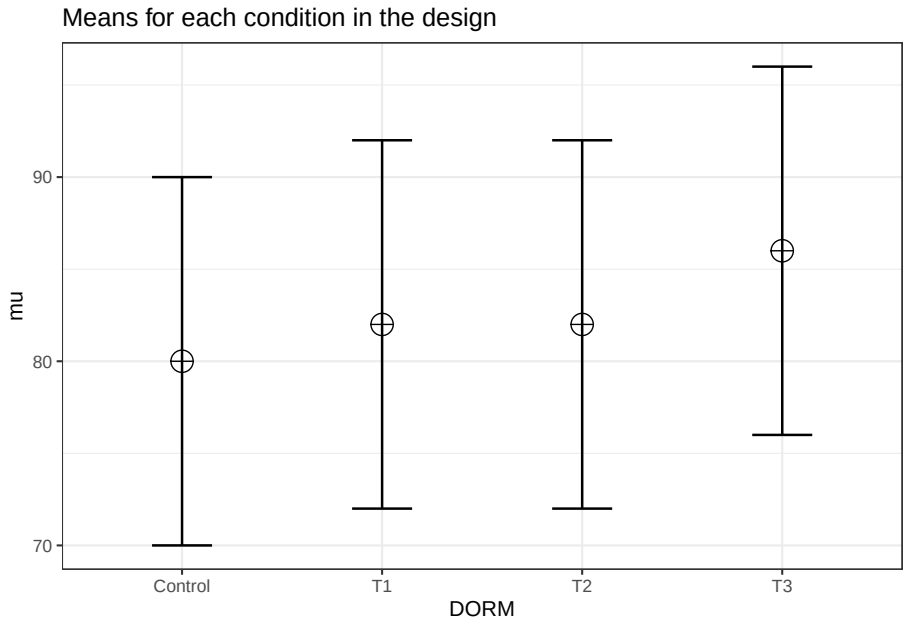
In this appendix we have included the performance of `Superpower` to the `pwr2ppl` package using Chris Aberson's examples in *Applied Power Analysis for the Behavioral Sciences* (2nd edition).

10.5 Examples from Chapter 5

10.5.1 Example 5.1/5.2

In this example, Aberson proposes a study expecting an average "score" of 80, 82, 82, and 86 for the control and three treatment groups respectively. The common standard deviation is 10 and the sample size per cell is 60.

```
design_result <- ANOVA_design(design = "4b",
                              n = 60,
                              sd = 10,
                              mu = c(80, 82, 82, 86),
                              labelnames = c("DORM",
                              "Control",
                              "T1",
                              "T2",
                              "T3"),
                              plot = TRUE)
```



Now we calculate the analytical result from `Superpower`.

```
analytical_result <- power_oneway_between(design_result)

analytical_result$power
```

```
## [1] 81.21291
```

The `ANOVA_exact` result.

```
exact_result <- ANOVA_exact(design_result, verbose = FALSE)

exact_result$main_results
```

```
##         power partial_eta_squared   cohen_f non_centrality
## DORM 81.21291          0.04607922 0.2197842           11.4
```

And these match `pwr2ppl`.

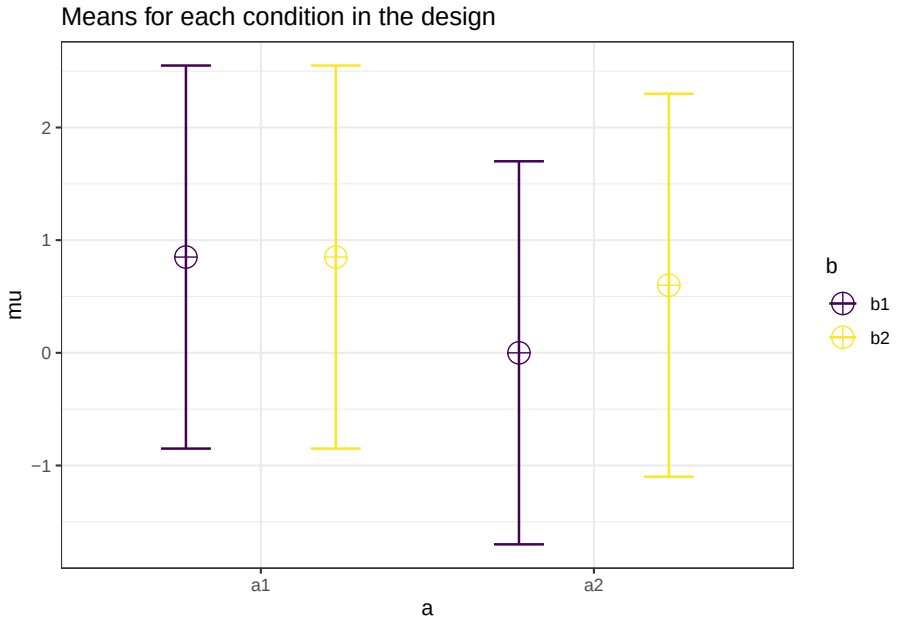
```
anova1f_4(m1 = 80, m2 = 82, m3 = 82, m4 = 86,
          s1 = 10, s2 = 10, s3 = 10, s4 = 10,
          n1 = 60, n2 = 60, n3 = 60, n4 = 60)
```

```
## Power  = 0.812 for eta-squared = 0.05
```

10.5.2 Example 5.3

Now a 2 x 2 between-subject ANOVA.

```
design_result <- ANOVA_design(design = "2b*2b",
                              n = 100,
                              sd = 1.7,
                              mu = c(.85, .85,
                                     0, .6),
                              plot = TRUE)
```



Now we calculate the analytical result from `Superpower`.

```
analytical_result <- power_twoway_between(design_result)

analytical_result$power_A
```

```
## [1] 89.75072
```

```
analytical_result$power_B
```

```
## [1] 42.10204
```

```
analytical_result$power_AB
```

```
## [1] 42.10204
```

The `ANOVA_exact` result.

```
exact_result <- ANOVA_exact(design_result, verbose = FALSE)

exact_result$main_results
```

```
##        power partial_eta_squared    cohen_f non_centrality
## a   89.75072         0.025751475 0.16257965      10.467128
## b   42.10204         0.007802747 0.08867981       3.114187
## a:b 42.10204         0.007802747 0.08867981       3.114187
```

And these match `pwr2ppl`. From Table 5.12.

```
anova2x2(m1.1 = 0.85, m1.2 = 0.85, m2.1 = 0.00, m2.2 = 0.60,
         s1.1 = 1.7, s1.2 = 1.7, s2.1 = 1.7, s2.2 = 1.7,
         n1.1 = 100, n1.2 = 100, n2.1 = 100, n2.2 = 100,
         alpha = .05)
```

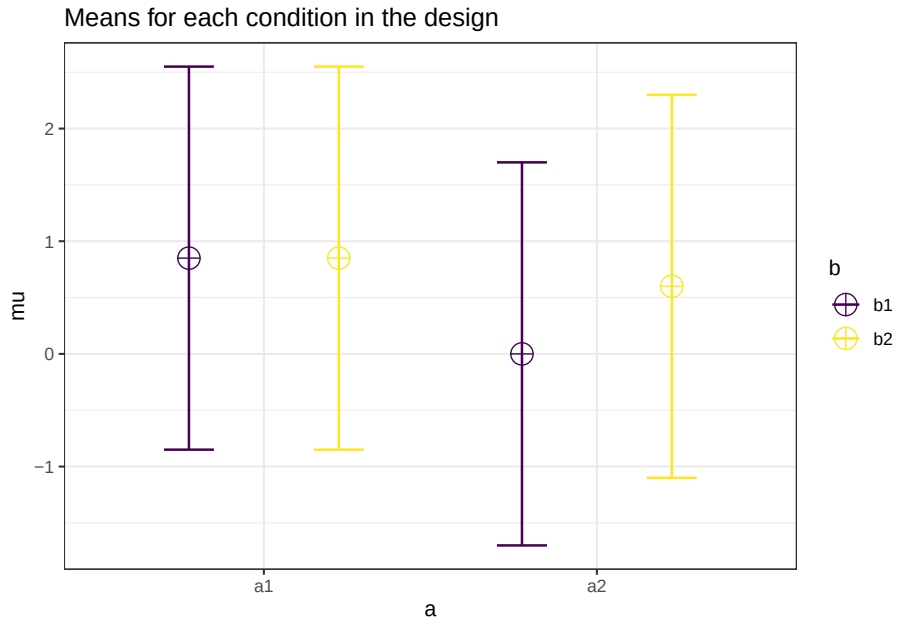
```
## Power for Main Effect Factor A = 0.898
```

```
## Power for Main Effect Factor B = 0.421
```

```
## Power for Interaction AxB = 0.421
```

Now we can increase the sample size to 250 per cell

```
design_result <- ANOVA_design(design = "2b*2b",
                              n = 250,
                              sd = 1.7,
                              mu = c(.85, .85,
                                     0, .6),
                              plot = TRUE)
```



Now we calculate the analytical result from `Superpower`.

```
analytical_result <- power_twoway_between(design_result)

analytical_result$power_A
```

```
## [1] 99.91852
```

```
analytical_result$power_B
```

```
## [1] 79.60496
```

```
analytical_result$power_AB
```

```
## [1] 79.60496
```

The `ANOVA_exact` result.

```
exact_result <- ANOVA_exact(design_result, verbose = FALSE)

exact_result$main_results
```

```
##        power partial_eta_squared   cohen_f non_centrality
## a   99.91852         0.025600317 0.1620892      26.167820
## b   79.60496         0.007756107 0.0884123       7.785467
## a:b 79.60496         0.007756107 0.0884123       7.785467
```

And these match `pwr2ppl`.

```
anova2x2(m1.1 = 0.85, m1.2 = 0.85, m2.1 = 0.00, m2.2 = 0.60,
         s1.1 = 1.7, s1.2 = 1.7, s2.1 = 1.7, s2.2 = 1.7,
         n1.1 = 250, n1.2 = 250, n2.1 = 250, n2.2 = 250,
         alpha = .05)
```

```
## Power for Main Effect Factor A = 0.999
```

```
## Power for Main Effect Factor B = 0.796
```

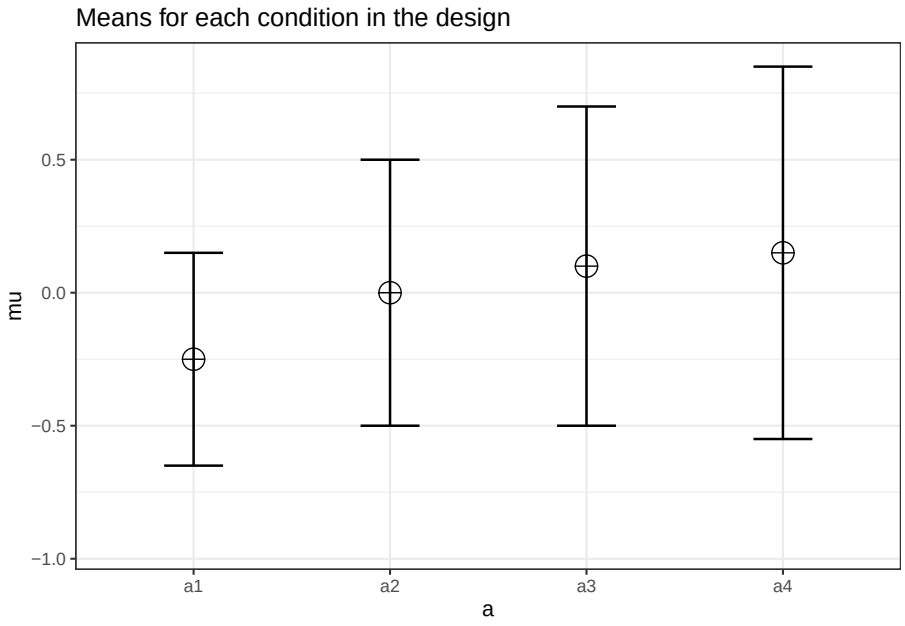
```
## Power for Interaction AxB = 0.796
```

10.6 Examples from Chapter 6

Repeated measures ANOVAs

10.6.1 Example from Table 6.2

```
design_result <- ANOVA_design(design = "4w",
                              n = 25,
                              sd = c(.4,.5,.6,.7),
                              mu = c(-.25, .00, .10, .15),
                              r = c(.50,
                                    .30,
                                    .15,
                                    .5,
                                    .30,
                                    .50),
                              plot = TRUE)
```




```
design_result$cor_mat
```

```
##      a1  a2  a3   a4
## a1 1.00 0.5 0.3 0.15
## a2 0.50 1.0 0.5 0.30
## a3 0.30 0.5 1.0 0.50
## a4 0.15 0.3 0.5 1.00
```

There is no analytical result from `Superpower` when the correlations vary.

Now we prouce 3 `ANOVA_exact` results representing no sphericity correction, Greenhouse-Geisser, and Huynh-Feldt corrected results.

```
exact_result <- ANOVA_exact(design_result, verbose = FALSE)

exact_result$main_results
```

```
##      power partial_eta_squared   cohen_f non_centrality
## a 80.94999           0.1404744 0.4042678       11.76713
```

```
exact_result <- ANOVA_exact(design_result,
                            correction = "GG",
                            verbose = FALSE)

exact_result$main_results
```

```
##      power partial_eta_squared   cohen_f non_centrality
## a 74.45876           0.1404744 0.4042678       9.585214
```

```
exact_result <- ANOVA_exact(design_result,
                            correction = "HF",
                            verbose = FALSE)

exact_result$main_results
```

```
##      power partial_eta_squared   cohen_f non_centrality
## a 78.14498           0.1404744 0.4042678       10.75258
```

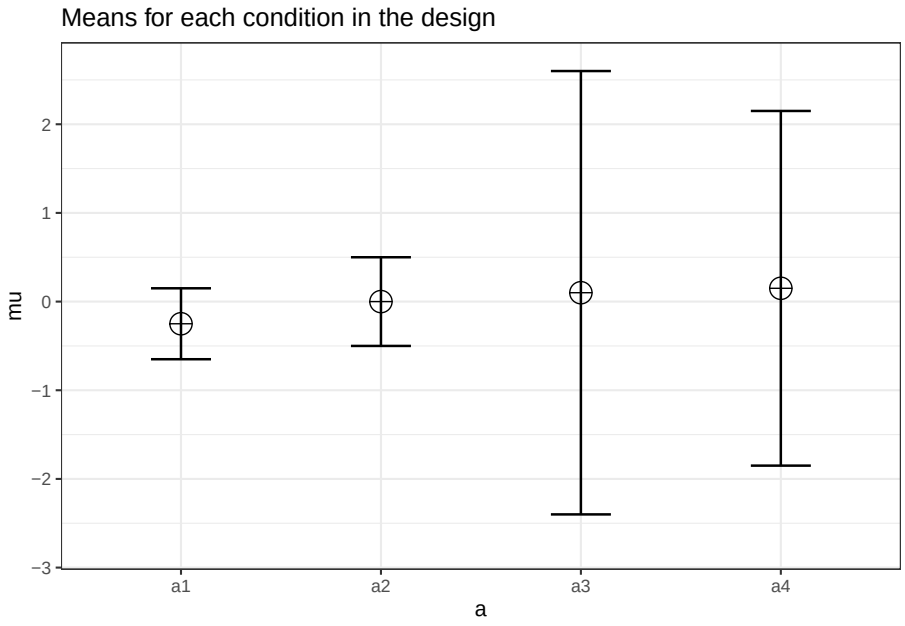
And these match `pwr2ppl`.

```
win1F(m1 = -.25, m2 = .00, m3 = .10, m4 = .15,
      s1 = .4, s2 = .5, s3 = .6, s4 = .7,
      r12 = .50, r13 = .30,
      r14 = .15, r23 = .5,
      r24 = .30, r34 = .50,
      n = 25)
```

```
## partial eta-squared = 0.14

## Power (Unadjusted) for n = 25 is 0.809

## Power H-F Adjusted (Epsilon = 0.914) for n = 25 is 0.782

## Power G-G Adjusted (Epsilon = 0.815) for n = 25 is 0.745
```

10.6.2 Example from Table 6.6

```
design_result <- ANOVA_design(design = "4w",
                              n = 100,
                              sd = c(.4,.5,2.5,2),
                              mu = c(-.25, .00, .10, .15),
                              r = c(.50,
                                    .30,
                                    .1,
                                    .5,
                                    .30,
                                    .40),
                              plot = TRUE)
```




```
design_result$cor_mat
```

```
##     a1  a2  a3  a4
## a1 1.0 0.5 0.3 0.1
## a2 0.5 1.0 0.5 0.3
## a3 0.3 0.5 1.0 0.4
## a4 0.1 0.3 0.4 1.0
```

There is no analytical result from `Superpower` when the correlations vary.

Now we prouce 3 `ANOVA_exact` results representing no sphericity correction, Greenhouse-Geisser, and Huynh-Feldt corrected results.

```
exact_result <- ANOVA_exact(design_result, verbose = FALSE)

exact_result$main_results
```

```
##      power partial_eta_squared   cohen_f non_centrality
## a 39.74802          0.01502077 0.1234902       4.529201
```

```
exact_result <- ANOVA_exact(design_result,
                            correction = "GG",
                            verbose = FALSE)

exact_result$main_results
```

```
##      power partial_eta_squared   cohen_f non_centrality
## a 31.78652          0.01502077 0.1234902       2.997994
```

```
exact_result <- ANOVA_exact(design_result,
                            correction = "HF",
                            verbose = FALSE)

exact_result$main_results
```

```
##      power partial_eta_squared   cohen_f non_centrality
## a 32.12295          0.01502077 0.1234902       3.059139
```

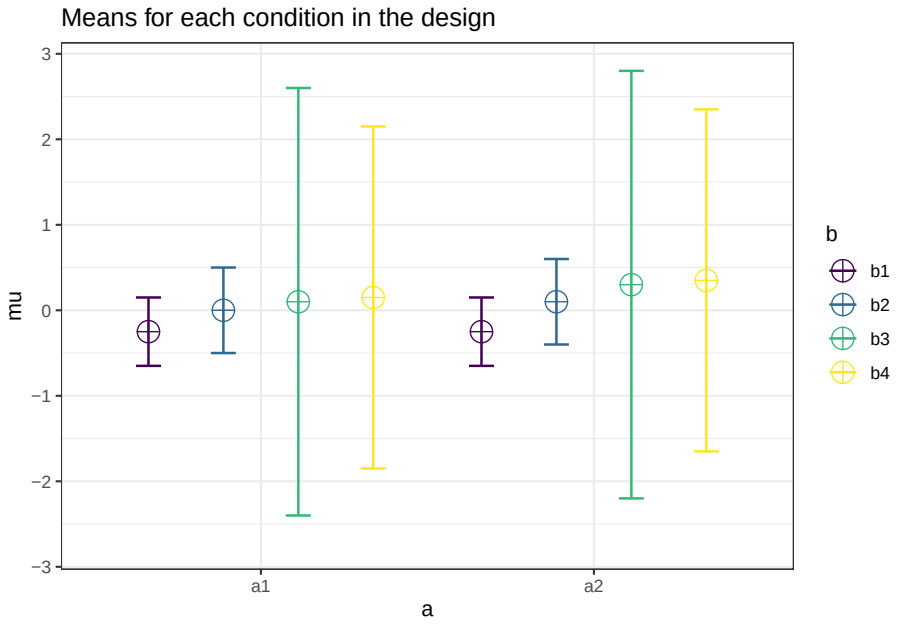
And these match `pwr2ppl`.

```
win1F(m1 = -.25, m2 = .00, m3 = .10, m4 = .15,
      s1 = .4, s2 = .5, s3 = 2.5, s4 = 2.0,
      r12 = .50, r13 = .30, r14 = .10,
      r23 = .5, r24 = .30, r34 = .40,
      n = 100)
```

```
## partial eta-squared = 0.015

## Power (Unadjusted) for n = 100 is 0.397

## Power H-F Adjusted (Epsilon = 0.675) for n = 100 is 0.321

## Power G-G Adjusted (Epsilon = 0.662) for n = 100 is 0.318
```

10.6.3 Example from Table 6.8

```
design_result <- ANOVA_design(design = "2w*4w",
                              n = 80,
                              sd = c(.4,0.5,
                                     2.5,2.0,
                                     0.4,0.5,
                                     2.5,2.0),
                              mu = c(-0.25,0.0,
                                     0.10,0.15,
                                     -0.25,0.10,
                                     0.30,0.35),
                              r = c(.5),
                              plot = TRUE)
```




```
design_result$cor_mat
```

```
##       a1_b1 a1_b2 a1_b3 a1_b4 a2_b1 a2_b2 a2_b3 a2_b4
## a1_b1   1.0   0.5   0.5   0.5   0.5   0.5   0.5   0.5
## a1_b2   0.5   1.0   0.5   0.5   0.5   0.5   0.5   0.5
## a1_b3   0.5   0.5   1.0   0.5   0.5   0.5   0.5   0.5
## a1_b4   0.5   0.5   0.5   1.0   0.5   0.5   0.5   0.5
## a2_b1   0.5   0.5   0.5   0.5   1.0   0.5   0.5   0.5
## a2_b2   0.5   0.5   0.5   0.5   0.5   1.0   0.5   0.5
## a2_b3   0.5   0.5   0.5   0.5   0.5   0.5   1.0   0.5
## a2_b4   0.5   0.5   0.5   0.5   0.5   0.5   0.5   1.0
```

There is no analytical result from `Superpower` for two-way within subjects designs.

Now we prouce 3 `ANOVA_exact` results representing no sphericity correction, Greenhouse-Geisser, and Huynh-Feldt corrected results.

```
#In comparision to pwr2ppl the main effects are "flipped"
# e.g. Superpower a = pwr2ppl "B"
exact_result <- ANOVA_exact(design_result, verbose = FALSE)

exact_result$main_results
```

```
##        power partial_eta_squared    cohen_f non_centrality
## a   27.24340         0.023198088 0.15410717      1.8761726
## b   74.84647         0.040077020 0.20432877      9.8948083
## a:b 10.23225         0.003471099 0.05901855      0.8255159
```

```
exact_result <- ANOVA_exact(design_result,
                            correction = "GG",
                            verbose = FALSE)

exact_result$main_results
```

```
##         power partial_eta_squared    cohen_f non_centrality
## a   27.243403         0.023198088 0.15410717      1.8761726
## b   58.540772         0.040077020 0.20432877      5.9072378
## a:b  9.130272         0.003471099 0.05901855      0.5072161
```

```
exact_result <- ANOVA_exact(design_result,
                            correction = "HF",
                            verbose = FALSE)

exact_result$main_results
```

```
##         power partial_eta_squared    cohen_f non_centrality
## a   27.243403         0.023198088 0.15410717      1.8761726
## b   59.182337         0.040077020 0.20432877      6.0353398
## a:b  9.174188         0.003471099 0.05901855      0.5187745
```

And these match `pwr2ppl`.

```
win2F(m1.1 = -.25, m2.1 = 0,
      m3.1 = .10, m4.1 = .15,
      m1.2 = -.25, m2.2 = .10,
      m3.2 = .30, m4.2 = .35,
      s1.1 = .4, s2.1 = .5,
      s3.1 = 2.5, s4.1 = 2.0,
      s1.2 = .4, s2.2 = .5,
      s3.2 = 2.5, s4.2 = 2.0,
      r = .5, n = 80)
```

```
## Partial eta-squared Factor A = 0.04
```

```
## Power Factor A (Unadjusted) for n = 80 is 0.748
```

```
## Power Factor A H-F Adjusted (Epsilon = 0.61) for n = 80 is 0.592

## Power Factor A G-G Adjusted (Epsilon = 0.597) for n = 80 is 0.585

## Partial eta-squared Factor B = 0.023

## Power Factor B (Unadjusted) for n = 80 is 0.272

## Power Factor B Adjusted - There is no adjustment when levels = 2

## Partial eta-squared AxB = 0.003

## Power AxB (Unadjusted) for n = 80 is 0.102

## Power AxB H-F Adjusted (Epsilon = 0.628) for n = 80 is 0.092

## Power AxB G-G Adjusted (Epsilon = 0.614) for n = 80 is 0.091
```

10.7 Example from Chapter 7

Mixed effect ANOVA

10.7.1 From Table 7.2

In this case we must write out an entire correlation matrix. This means the diagonal element is equal to 1 and the off-diagonal elements corresponding to between-subjects factors are equal to zero.

```
design_result <- ANOVA_design("2b*4w",
                              n = 50,
                              sd = c(.4, .5, 0.6, .7,
                                     .4, .5, .6, .7),
                              r = c(1.0,0.5,0.3,0.15,0.0,0.0,0.0,0.0,
                                    0.5,1.0,0.5,0.3,0.0,0.0,0.0,0.0,
                                    0.3,0.5,1.0,0.5,0.0,0.0,0.0,0.0,
                                    0.15,0.3,0.5,1.0,0.0,0.0,0.0,0.0,
                                    0.0,0.0,0.0,0.0,1.0,0.5,0.3,0.15,
                                    0.0,0.0,0.0,0.0,0.5,1.0,0.5,0.3,
                                    0.0,0.0,0.0,0.0,0.3,0.5,1.0,0.5,
                                    0.0,0.0,0.0,0.0,0.15,0.3,0.5,1.0),
                              mu = c(-.25, 0.0, 0.10, 0.15,
                                     -.25,-.25,-.25,-.25))

design_result$cor_mat
```

```
##       a1_b1 a1_b2 a1_b3 a1_b4 a2_b1 a2_b2 a2_b3 a2_b4
## a1_b1  1.00   0.5   0.3  0.15  0.00   0.0   0.0  0.00
## a1_b2  0.50   1.0   0.5  0.30  0.00   0.0   0.0  0.00
## a1_b3  0.30   0.5   1.0  0.50  0.00   0.0   0.0  0.00
## a1_b4  0.15   0.3   0.5  1.00  0.00   0.0   0.0  0.00
## a2_b1  0.00   0.0   0.0  0.00  1.00   0.5   0.3  0.15
## a2_b2  0.00   0.0   0.0  0.00  0.50   1.0   0.5  0.30
## a2_b3  0.00   0.0   0.0  0.00  0.30   0.5   1.0  0.50
## a2_b4  0.00   0.0   0.0  0.00  0.15   0.3   0.5  1.00
```

Now the results from `ANOVA_exact`.

```
exact_result <- ANOVA_exact(design_result,
                            correction = "none",
                            verbose = FALSE)

exact_result$main_results
```

```
##        power partial_eta_squared   cohen_f non_centrality
## a   86.42918          0.08878976 0.3121563       9.549274
## b   82.68405          0.03848397 0.2000607      11.767135
## a:b 82.68405          0.03848397 0.2000607      11.767135
```

```
exact_result <- ANOVA_exact(design_result,
                            correction = "GG",
                            verbose = FALSE)

exact_result$main_results
```

```
##        power partial_eta_squared   cohen_f non_centrality
## a   86.42918          0.08878976 0.3121563       9.549274
## b   76.47495          0.03848397 0.2000607       9.585214
## a:b 76.47495          0.03848397 0.2000607       9.585214
```

```
exact_result <- ANOVA_exact(design_result,
                            correction = "HF",
                            verbose = FALSE)

exact_result$main_results
```

```
##        power partial_eta_squared   cohen_f non_centrality
## a   86.42918          0.08878976 0.3121563       9.549274
## b   77.31933          0.03848397 0.2000607       9.848557
## a:b 77.31933          0.03848397 0.2000607       9.848557
```

And the results from `pwr2ppl`.

```
win1bg1(m1.1 = -.25, m2.1 = 0, m3.1 = 0.10, m4.1 = .15,
        m1.2 = -.25, m2.2 = -.25, m3.2 = -.25, m4.2 = -.25,
        s1.1 = .4, s2.1 = .5, s3.1 = 0.6, s4.1 = .7, s1.2 = .4,
        s2.2 = .5, s3.2 = .6, s4.2 = .7,
        n = 50,
        r1.2_1 = .5, r1.3_1 = .3, r1.4_1 = .15,
        r2.3_1 = .5, r2.4_1 = .3, r3.4_1 = .5,
        r1.2_2 = .5, r1.3_2 = .3, r1.4_2 = .15,
        r2.3_2 = .5, r2.4_2 = .3, r3.4_2 = .5)
```

```
## Partial eta-squared Factor A = 0.089

## Power Factor A (Between) for n = 50 is 0.864

## Partial eta-squared Factor B = 0.038

## Power Factor B (Within) for n = 50 is 0.827

## Power Factor B H-F Adjusted (Epsilon = 0.837), for n = 50 is 0.773

## Power Factor B G-G Adjusted (Epsilon = 0.815) for n = 50 is 0.765

## Partial eta-squared Factor AxB = 0.089

## Power AxB (Unadjusted) for n = 50 is 0.827

## Power AxB H-F Adjusted (Epsilon = 0.837) for n = 50 is 0.761

## Power AxB G-G Adjusted (Epsilon = 0.815) for n = 50 is 0.765
```

Appendix 2: Direct Comparison to MOREpower

MOREpower 6.0 by Campbell and Thompson (2012) is standalone software for power analysis for ANOVA, t-tests, correlations, and tests of proportions. It outperforms Gpower, in that it allows researchers to perform power analyses for a much wider range of designs. It allows a maximum of 9 levels per factor. Users can solve for N, power or effect size. An ANOVA effect size may be specified in terms of the effect-related variance explained (partial eta-squared) or in terms of a test statistic (F, mean square treatment [MST], or t).

Compared to `Superpower`, it does not provide all tests at once, it does not provide simple comparisons, it does not incorporate corrections for multiple comparisons, and it does not allow users to enter the means, sd's, and correlations.

We can replicate the independent t-test example in MOREpower:

```
string <- "2b"
n <- 100
mu <- c(24, 26.2)
sd <- 6.4
labelnames <- c("condition", "control", "pet") #
# the label names should be in the order of the means specified above.
design_result <- ANOVA_design(design = string,
                              n = n,
                              mu = mu,
                              sd = sd,
                              labelnames = labelnames)
alpha_level <- 0.05

exact_result <- ANOVA_exact(design_result,
```

```
                                alpha_level = alpha_level,
                                verbose = TRUE)
```

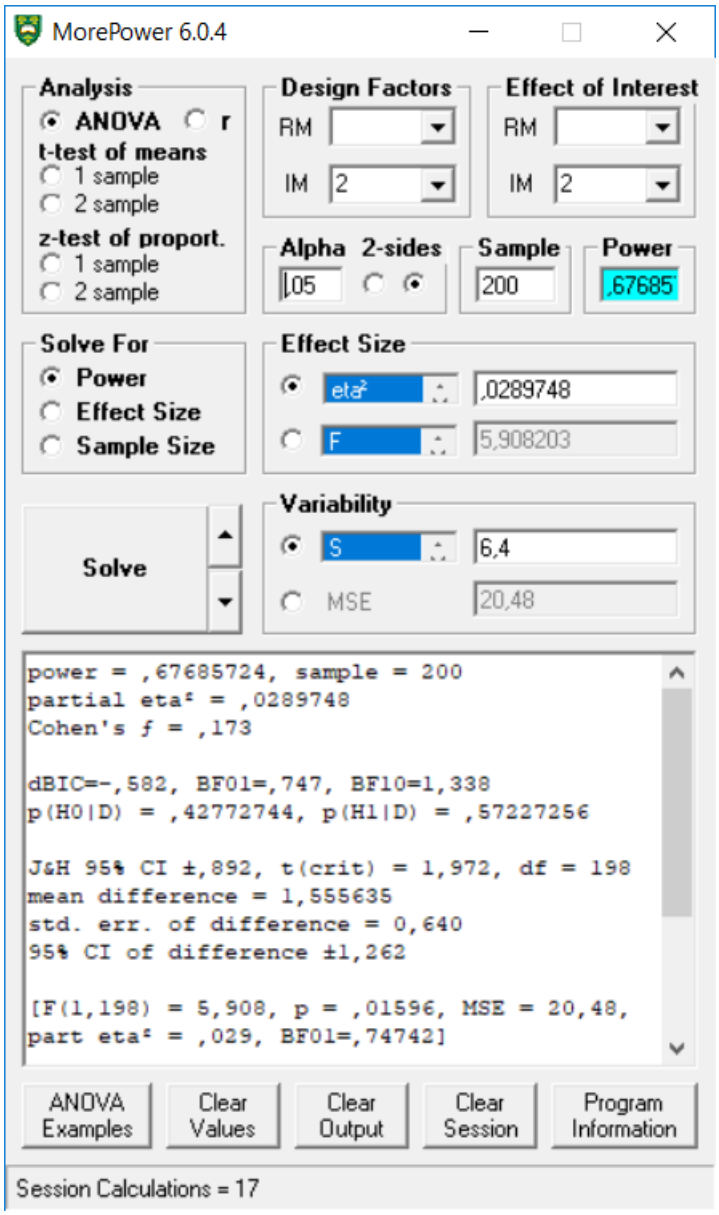
```
## Power and Effect sizes for ANOVA tests
##             power partial_eta_squared cohen_f non_centrality
## condition 67.6857               0.029  0.1727         5.9082
##
## Power and Effect sizes for pairwise comparisons (t-tests)
##                                    power effect_size
## p_condition_control_condition_pet 67.69        0.34
```

```
exact_result$main_result$partial_eta_squared
```

```
## [1] 0.02897482
```

```
exact_result$main_result$power
```

```
## [1] 67.68572
```



We see the power estimates almost perfectly align with MOREpower if we enter partial eta-squared of 0.02897482, sample = 200 (MOREpower requires entering the total N, not N per condition).

We can also replicate the dependent *t*-test example in MOREpower.

```
K <- 2
n <- 34
sd <- 1
r <- 0.5
alpha = 0.05
f <- 0.25
f2 <- f^2
ES <- f2/(f2 + 1)
ES
```

```
## [1] 0.05882353
```

```
mu <- mu_from_ES(K = K, ES = ES)
design = paste(K,"w",sep = "")
labelnames <- c("speed", "fast", "slow")

design_result <- ANOVA_design(design = design,
                   n = n,
                   mu = mu,
                   sd = sd,
                   r = r,
                   labelnames = labelnames)

alpha_level <- 0.05

exact_result <- ANOVA_exact(design_result,
                            alpha_level = alpha_level,
                            verbose = TRUE)
```

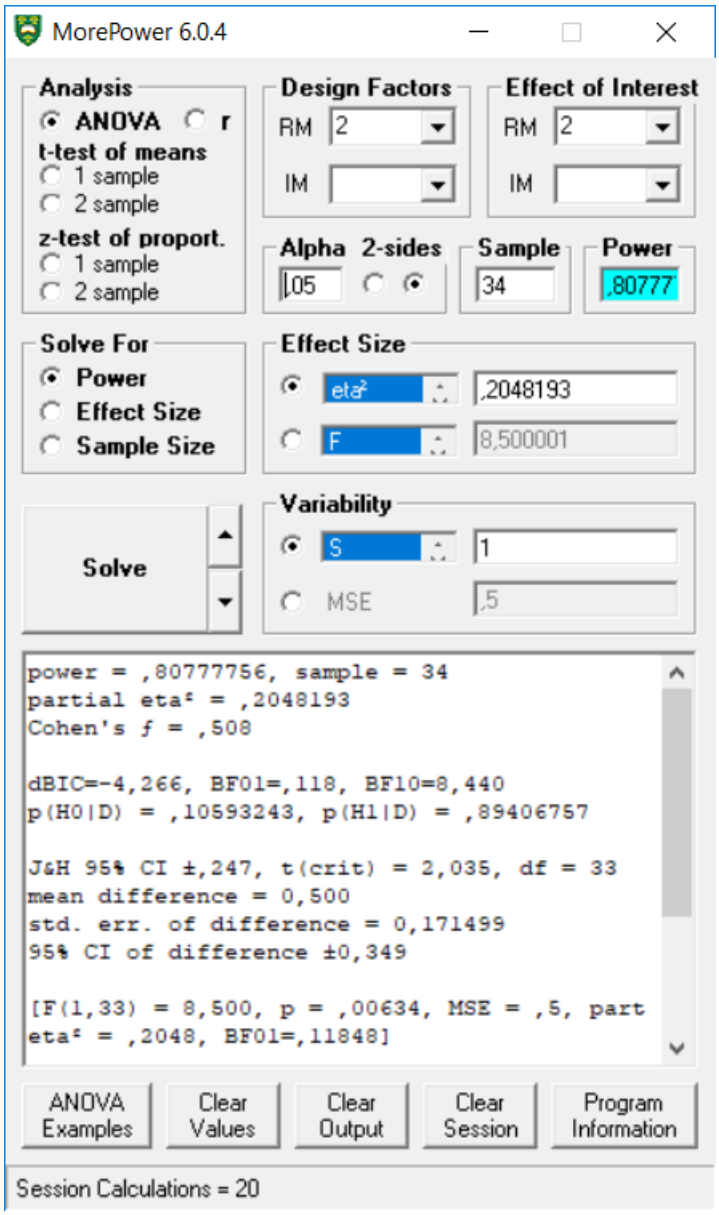
```
## Power and Effect sizes for ANOVA tests
##         power partial_eta_squared cohen_f non_centrality
## speed 80.7778              0.2048  0.5075            8.5
##
## Power and Effect sizes for pairwise comparisons (t-tests)
##                           power effect_size
## p_speed_fast_speed_slow 80.78         0.5
```

```
exact_result$main_result$partial_eta_squared
```

```
## [1] 0.2048193
```

```
exact_result$main_result$power
```

```
## [1] 80.77775
```



We can also replicate the 3-group within ANOVA example in MOREpower.

```
K <- 3
n <- 20
sd <- 1
```

```
r <- 0.8
alpha = 0.05
f <- 0.25
f2 <- f^2
ES <- f2 / (f2 + 1)
ES
```

```
## [1] 0.05882353
```

```
mu <- mu_from_ES(K = K, ES = ES)

design = paste(K, "w", sep = "")
labelnames <- c("speed", "fast", "medium", "slow")
design_result <- ANOVA_design(design = design,
                   n = n,
                   mu = mu,
                   sd = sd,
                   r = r,
                   labelnames = labelnames)

alpha_level <- 0.05

exact_result <- ANOVA_exact(design_result,
                            alpha_level = alpha_level,
                            verbose = TRUE)
```

```
## Power and Effect sizes for ANOVA tests
##         power partial_eta_squared cohen_f non_centrality
## speed 96.9163              0.3304  0.7024          18.75
##
## Power and Effect sizes for pairwise comparisons (t-tests)
##                            power effect_size
## p_speed_fast_speed_medium 53.79        0.48
## p_speed_fast_speed_slow   98.39        0.97
## p_speed_medium_speed_slow 53.79        0.48
```

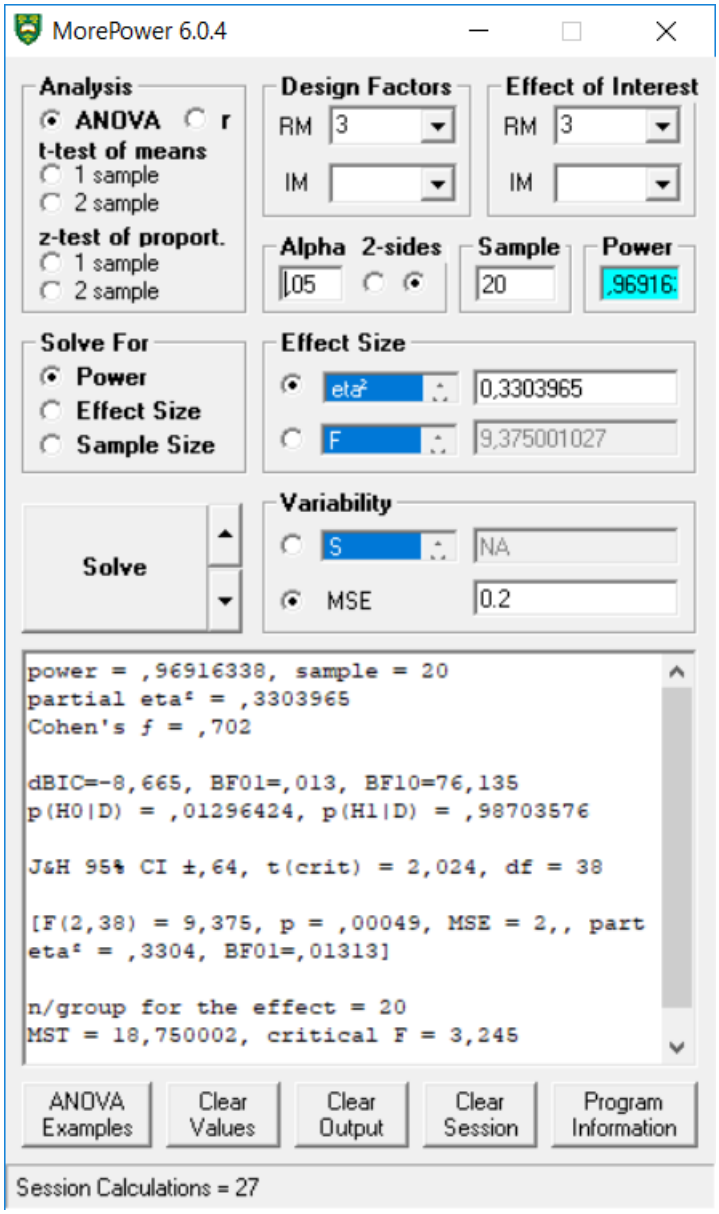
```
#MSE, for MOREpower
exact_result[["aov_result"]][["anova_table"]]
```

```
## Anova Table (Type 3 tests)
##
## Response: y
##       num Df den Df MSE     F    pes    Pr(>F)
## speed      2     38 0.2 9.375 0.3304 0.0004904 ***
```

```
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
#POWER
exact_result$main_result$power
```

```
## [1] 96.91634
```



We can reproduce the 2x2 ANOVA example in MOREpower.

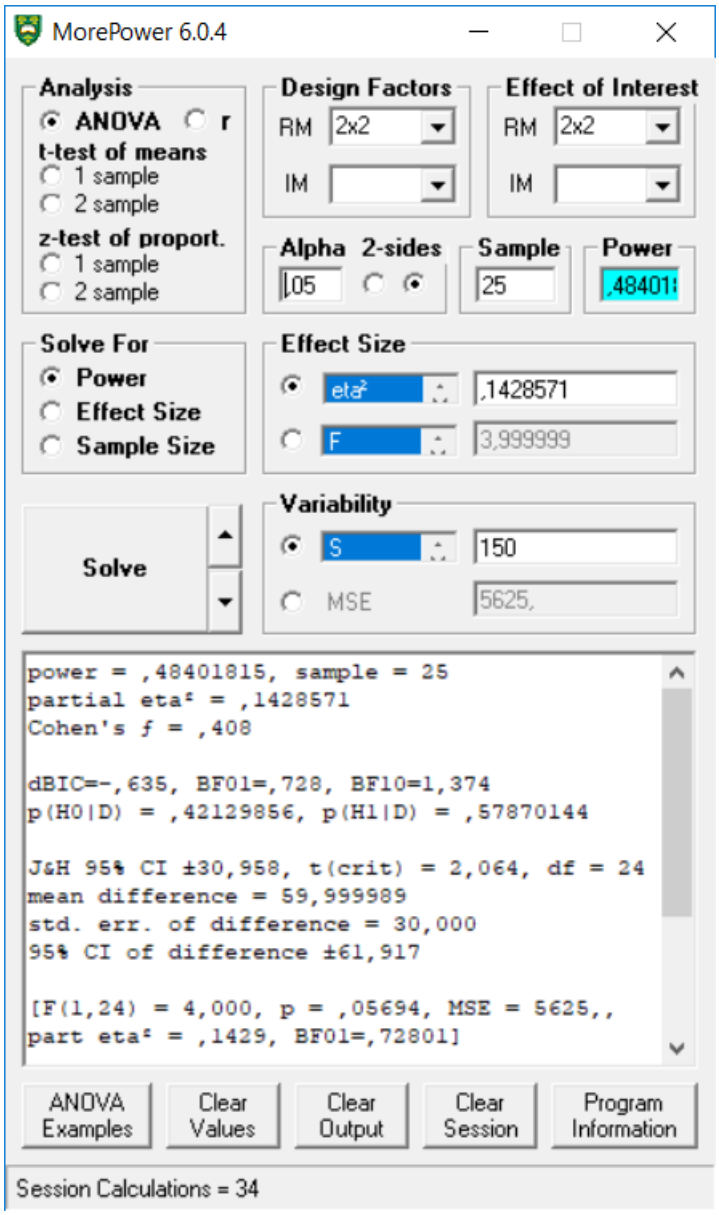
```
mu = c(700, 670, 670, 700)
sigma = 150  # population standard deviation
n <- 25
sd <- 150
r <- 0.75
string = "2w*2w"
alpha_level <- 0.05
labelnames = c("age", "old", "young", "color", "blue", "red")
design_result <- ANOVA_design(design = string,
                              n = n,
                              mu = mu,
                              sd = sd,
                              r = r,
                              labelnames = labelnames)

simulation_result <- ANOVA_exact(design_result,
                                 alpha_level = alpha_level,
                                 verbose = TRUE)
```

```
## Power and Effect sizes for ANOVA tests
##             power partial_eta_squared cohen_f non_centrality
## age        5.0000              0.0000  0.0000              0
## color      5.0000              0.0000  0.0000              0
## age:color 48.4018              0.1429  0.4082              4
##
## Power and Effect sizes for pairwise comparisons (t-tests)
##                                            power effect_size
## p_age_old_color_blue_age_old_color_red      27.4       -0.28
## p_age_old_color_blue_age_young_color_blue   27.4       -0.28
## p_age_old_color_blue_age_young_color_red     5.0        0.00
## p_age_old_color_red_age_young_color_blue     5.0        0.00
## p_age_old_color_red_age_young_color_red     27.4        0.28
## p_age_young_color_blue_age_young_color_red  27.4        0.28
```

```
simulation_result$main_result$partial_eta_squared[3]
```

```
## [1] 0.1428571
```

This result reproduces the analysis in MOREpower, for a 2x2 within design, with a sample of 25, and eta-squared of 0.1428571.



Let's replicate the 2x2x2 full between analysis (total N = 400, effects sizes differ for each test):

```
# With 2x2x2 designs,
# the names for paired comparisons can become very long.
# So here I abbreviate terms:
# Size, Color, and Cognitive Load, have values:
```

```
# b = big, s = small, g = green,
# r = red, pres = present, abs = absent.
labelnames <- c("Size", "b", "s", "Color", "g", "r",
                "Load", "pres", "abs") #
design_result <- ANOVA_design(design = "2b*2b*2b",
                              #sample size per group
                              n = 50,
                              #pattern of means
                              mu = c(2, 2, 6, 1, 6, 6, 1, 8),
                              sd = 10, #standard deviation
                              labelnames = labelnames)

exact_result <- ANOVA_exact(design_result,
                            alpha_level = alpha_level,
                            verbose = TRUE)
```

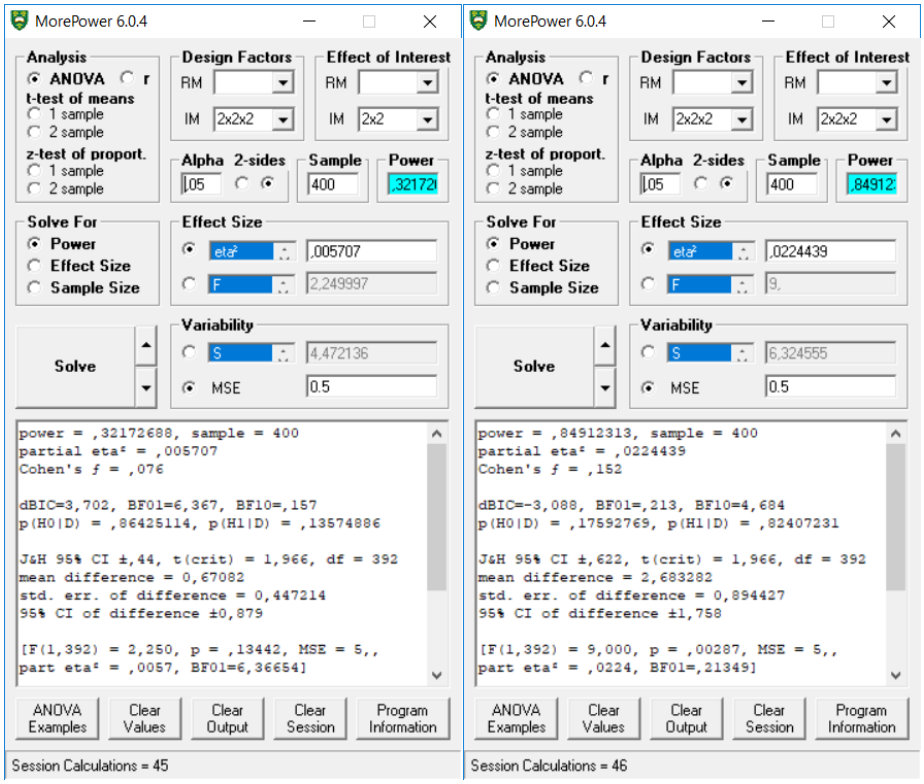
```
## Power and Effect sizes for ANOVA tests
##                   power partial_eta_squared cohen_f non_centrality
## Size            70.3301              0.0157  0.1263           6.25
## Color            5.0000              0.0000  0.0000           0.00
## Load             7.8953              0.0006  0.0253           0.25
## Size:Color      32.1727              0.0057  0.0758           2.25
## Size:Load       84.9123              0.0224  0.1515           9.00
## Color:Load       7.8953              0.0006  0.0253           0.25
## Size:Color:Load 84.9123              0.0224  0.1515           9.00
##
## Power and Effect sizes for pairwise comparisons (t-tests)
##                                                    power effect_size
## p_Size_b_Color_g_Load_pres_Size_b_Color_g_Load_abs  5.00         0.0
## p_Size_b_Color_g_Load_pres_Size_b_Color_r_Load_pres 50.82        0.4
## p_Size_b_Color_g_Load_pres_Size_b_Color_r_Load_abs  7.85        -0.1
## p_Size_b_Color_g_Load_pres_Size_s_Color_g_Load_pres 50.82        0.4
## p_Size_b_Color_g_Load_pres_Size_s_Color_g_Load_abs  50.82        0.4
## p_Size_b_Color_g_Load_pres_Size_s_Color_r_Load_pres  7.85        -0.1
## p_Size_b_Color_g_Load_pres_Size_s_Color_r_Load_abs  84.39         0.6
## p_Size_b_Color_g_Load_abs_Size_b_Color_r_Load_pres  50.82         0.4
## p_Size_b_Color_g_Load_abs_Size_b_Color_r_Load_abs    7.85        -0.1
## p_Size_b_Color_g_Load_abs_Size_s_Color_g_Load_pres  50.82         0.4
## p_Size_b_Color_g_Load_abs_Size_s_Color_g_Load_abs   50.82         0.4
## p_Size_b_Color_g_Load_abs_Size_s_Color_r_Load_pres   7.85        -0.1
## p_Size_b_Color_g_Load_abs_Size_s_Color_r_Load_abs   84.39         0.6
## p_Size_b_Color_r_Load_pres_Size_b_Color_r_Load_abs  69.69        -0.5
## p_Size_b_Color_r_Load_pres_Size_s_Color_g_Load_pres  5.00         0.0
## p_Size_b_Color_r_Load_pres_Size_s_Color_g_Load_abs   5.00         0.0
## p_Size_b_Color_r_Load_pres_Size_s_Color_r_Load_pres 69.69        -0.5
```

```
## p_Size_b_Color_r_Load_pres_Size_s_Color_r_Load_abs  16.77         0.2
## p_Size_b_Color_r_Load_abs_Size_s_Color_g_Load_pres  69.69         0.5
## p_Size_b_Color_r_Load_abs_Size_s_Color_g_Load_abs   69.69         0.5
## p_Size_b_Color_r_Load_abs_Size_s_Color_r_Load_pres   5.00         0.0
## p_Size_b_Color_r_Load_abs_Size_s_Color_r_Load_abs   93.39         0.7
## p_Size_s_Color_g_Load_pres_Size_s_Color_g_Load_abs   5.00         0.0
## p_Size_s_Color_g_Load_pres_Size_s_Color_r_Load_pres 69.69        -0.5
## p_Size_s_Color_g_Load_pres_Size_s_Color_r_Load_abs  16.77         0.2
## p_Size_s_Color_g_Load_abs_Size_s_Color_r_Load_pres  69.69        -0.5
## p_Size_s_Color_g_Load_abs_Size_s_Color_r_Load_abs   16.77         0.2
## p_Size_s_Color_r_Load_pres_Size_s_Color_r_Load_abs  93.39         0.7
```

```
exact_result$main_result$partial_eta_squared
```

```
## [1] 0.0156936598 0.0000000000 0.0006373486 0.0057070387 0.0224438903
## [6] 0.0006373486 0.0224438903
```

This result is nicely reproduced in MOREpower, both for the 2x2 effects, and the 2x2x2 between effects.



And finally, let's replicate the 2x2 full within analysis.

```
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"
labelnames = c("A", "a1", "a2", "B", "b1", "b2")
design_result <- ANOVA_design(design = string,
                              n = n,
                              mu = mu,
                              sd = sd,
                              r = r,
                              labelnames = labelnames)
exact_result <- ANOVA_exact(design_result,
                            alpha_level = alpha_level,
                            verbose = TRUE)
```

```
## Power and Effect sizes for ANOVA tests
##       power partial_eta_squared cohen_f non_centrality
## A   26.9175              0.0952  0.3244              2
## B   64.2259              0.2400  0.5620              6
## A:B 26.9175              0.0952  0.3244              2
##
## Power and Effect sizes for pairwise comparisons (t-tests)
##                         power effect_size
## p_A_a1_B_b1_A_a1_B_b2 26.92       -0.32
## p_A_a1_B_b1_A_a2_B_b1 39.70        0.40
## p_A_a1_B_b1_A_a2_B_b2  5.00        0.00
## p_A_a1_B_b2_A_a2_B_b1 64.23        0.55
## p_A_a1_B_b2_A_a2_B_b2 13.60        0.20
## p_A_a2_B_b1_A_a2_B_b2 76.52       -0.63
```

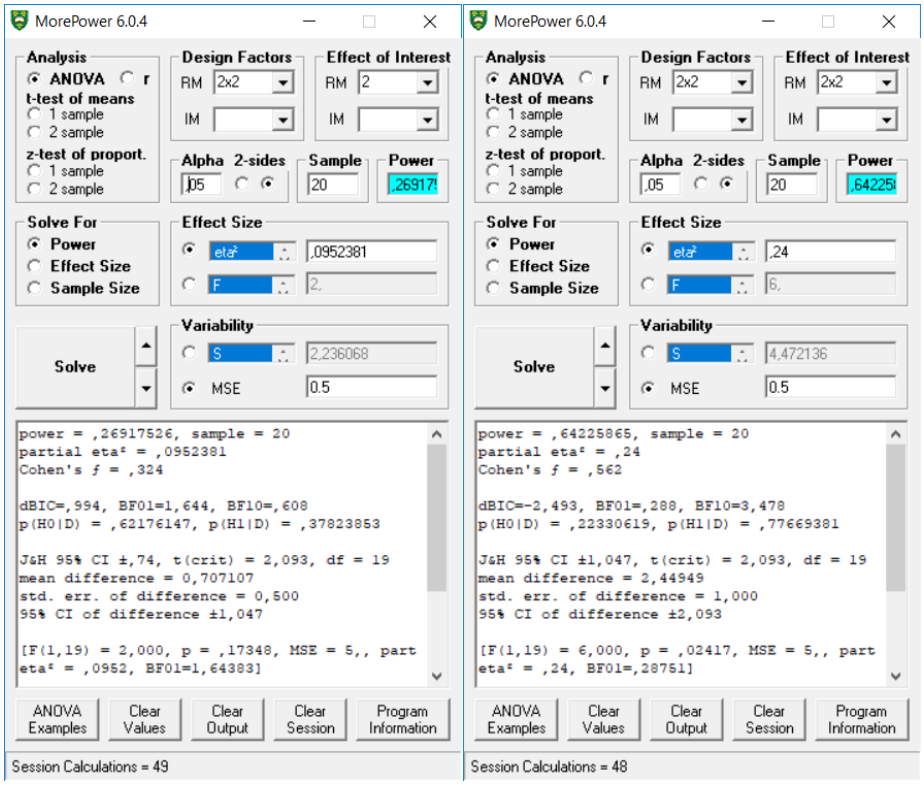
```
exact_result$main_result$partial_eta_squared
```

```
## [1] 0.0952381 0.2400000 0.0952381
```

```
exact_result$main_result$power
```

```
## [1] 26.91752 64.22587 26.91752
```

This result is also reproduced in MOREpower, both for the main effect, and the 2x2 within interaction effect.



Appendix 3: Comparison to Brysbaert

10.8 One-Way ANOVA

Now we will simply replicate the simulations of Brysbaert (2019), and compare those results to `Superpower`. Simulations to estimate the power of an ANOVA with three unrelated groups the effect between the two extreme groups is set to d = .4, the effect for the third group is d = .4 (see below for other situations) we use the built-in aov-test command give sample sizes (all samples sizes are equal).

```
# Simulations to estimate the power of an ANOVA
#with three unrelated groups
# the effect between the two extreme groups is set to d = .4,
# the effect for the third group is d = .4
#(see below for other situations)
# we use the built-in aov-test command
# give sample sizes (all samples sizes are equal)
N = 90
# give effect size d
d1 = .4 # difference between the extremes
d2 = .4 # third condition goes with the highest extreme
# give number of simulations
nSim = nsims
# give alpha levels
# alpha level for the omnibus ANOVA
alpha1 = .05
#alpha level for three post hoc one-tail t-tests Bonferroni correction
alpha2 = .05
```

```
# create vectors to store p-values
p1 <- numeric(nSim) #p-value omnibus ANOVA
```

```
p2 <- numeric(nSim) #p-value first post hoc test
p3 <- numeric(nSim) #p-value second post hoc test
p4 <- numeric(nSim) #p-value third post hoc test
pes1 <- numeric(nSim) #partial eta-squared
pes2 <- numeric(nSim) #partial eta-squared two extreme conditions
```

```
for (i in 1:nSim) {

x <- rnorm(n = N, mean = 0, sd = 1)
y <- rnorm(n = N, mean = d1, sd = 1)
z <- rnorm(n = N, mean = d2, sd = 1)
data = c(x, y, z)
groups = factor(rep(letters[24:26], each = N))
test <- aov(data ~ groups)
pes1[i] <- etaSquared(test)[1, 2]
p1[i] <- summary(test)[[1]][["Pr(>F)"]][[1]]
p2[i] <- t.test(x, y)$p.value
p3[i] <- t.test(x, z)$p.value
p4[i] <- t.test(y, z)$p.value
data = c(x, y)
groups = factor(rep(letters[24:25], each = N))
test <- aov(data ~ groups)
pes2[i] <- etaSquared(test)[1, 2]
}
```

```
# results are as predicted when omnibus ANOVA is significant,
# t-tests are significant between x and y plus x and z;
# not significant between y and z
# printing all unique tests (adjusted code by DL)
sum(p1 < alpha1) / nSim
sum(p2 < alpha2) / nSim
sum(p3 < alpha2) / nSim
sum(p4 < alpha2) / nSim
mean(pes1)
mean(pes2)
```

10.8.1 Three conditions replication

```
K <- 3
mu <- c(0, 0.4, 0.4)
n <- 90
sd <- 1
```

```
r <- 0
design = paste(K, "b", sep = "")
```

```
design_result <- ANOVA_design(
  design = design,
  n = n,
  mu = mu,
  sd = sd,
  labelnames = c("factor1", "level1", "level2", "level3")
  )
```

```
simulation_result <- ANOVA_power(design_result,
                                 alpha_level = alpha_level,
                                 nsims = nsims,
                                 verbose = FALSE)
```

Table 10.2: Simulated ANOVA Result

	power	effect_size
anova_factor1	78.81	0.0413822

```
exact_result <- ANOVA_exact(design_result,
                            alpha_level = alpha_level,
                            verbose = FALSE)
```

Table 10.3: Exact ANOVA Result

	power	partial_eta_squared	cohen_f	non_centrality
factor1	79.37239	0.0347072	0.1896182	9.6

10.8.2 Variation 1

```
# give sample sizes (all samples sizes are equal)
N = 145
# give effect size d
d1 = .4 #difference between the extremes
d2 = .0 #third condition goes with the highest extreme
# give number of simulations
nSim = nsims
```

```
# give alpha levels
#alpha level for the omnibus ANOVA
alpha1 = .05
#alpha level for three post hoc one-tail t-test Bonferroni correction
alpha2 = .05
```

```
# create vectors to store p-values
p1 <- numeric(nSim) #p-value omnibus ANOVA
p2 <- numeric(nSim) #p-value first post hoc test
p3 <- numeric(nSim) #p-value second post hoc test
p4 <- numeric(nSim) #p-value third post hoc test
pes1 <- numeric(nSim) #partial eta-squared
pes2 <- numeric(nSim) #partial eta-squared two extreme conditions
```

```
for (i in 1:nSim) {


x <- rnorm(n = N, mean = 0, sd = 1)
y <- rnorm(n = N, mean = d1, sd = 1)
z <- rnorm(n = N, mean = d2, sd = 1)
data = c(x, y, z)
groups = factor(rep(letters[24:26], each = N))
test <- aov(data ~ groups)
pes1[i] <- etaSquared(test)[1, 2]
p1[i] <- summary(test)[[1]][["Pr(>F)"]][[1]]
p2[i] <- t.test(x, y)$p.value
p3[i] <- t.test(x, z)$p.value
p4[i] <- t.test(y, z)$p.value
data = c(x, y)
groups = factor(rep(letters[24:25], each = N))
test <- aov(data ~ groups)
pes2[i] <- etaSquared(test)[1, 2]
}
```

```
# results are as predicted when omnibus ANOVA is significant,
# t-tests are significant between x and y plus x and z;
# not significant between y and z
# printing all unique tests (adjusted code by DL)
sum(p1 < alpha1) / nSim
sum(p2 < alpha2) / nSim
sum(p3 < alpha2) / nSim
sum(p4 < alpha2) / nSim
mean(pes1)
mean(pes2)
```

10.8.3 Three conditions replication

```
K <- 3
mu <- c(0, 0.4, 0.0)
n <- 145
sd <- 1
r <- 0
design = paste(K, "b", sep = "")
```

```
design_result <- ANOVA_design(
  design = design,
  n = n,
  mu = mu,
  sd = sd,
  labelnames = c("factor1", "level1", "level2", "level3")
  )
```

```
simulation_result <- ANOVA_power(design_result,
                                 alpha_level = alpha_level,
                                 nsims = nsims,
                                 verbose = FALSE)
```

Table 10.4: Simulated ANOVA Result

	power	effect_size
anova_factor1	94.61	0.0386982

```
exact_result <- ANOVA_exact(design_result,
                            alpha_level = alpha_level,
                            verbose = FALSE)
```

Table 10.5: Exact ANOVA Result

	power	partial_eta_squared	cohen_f	non_centrality
factor1	94.89169	0.034565	0.1892154	15.46667

10.8.4 Variation 2

```
# give sample sizes (all samples sizes are equal)
N = 82
# give effect size d
d1 = .4 #difference between the extremes
d2 = .2 #third condition goes with the highest extreme
# give number of simulations
nSim = nsims
# give alpha levels
#alpha level for the omnibus ANOVA
alpha1 = .05
#alpha level for three post hoc one-tail t-test Bonferroni correction
alpha2 = .05
```

```
# create vectors to store p-values
p1 <- numeric(nSim) #p-value omnibus ANOVA
p2 <- numeric(nSim) #p-value first post hoc test
p3 <- numeric(nSim) #p-value second post hoc test
p4 <- numeric(nSim) #p-value third post hoc test
pes1 <- numeric(nSim) #partial eta-squared
```

```
for (i in 1:nSim) {
#for each simulated experiment

x <- rnorm(n = N, mean = 0, sd = 1)
y <- rnorm(n = N, mean = d1, sd = 1)
z <- rnorm(n = N, mean = d2, sd = 1)
data = c(x, y, z)
groups = factor(rep(letters[24:26], each = N))
test <- aov(data ~ groups)
pes1[i] <- etaSquared(test)[1, 2]
p1[i] <- summary(test)[[1]][["Pr(>F)"]][[1]]
p2[i] <- t.test(x, y)$p.value
p3[i] <- t.test(x, z)$p.value
p4[i] <- t.test(y, z)$p.value
data = c(x, y)
groups = factor(rep(letters[24:25], each = N))
test <- aov(data ~ groups)
pes2[i] <- etaSquared(test)[1, 2]
}
```

```
sum(p1 < alpha1) / nSim
sum(p2 < alpha2) / nSim
sum(p3 < alpha2) / nSim
sum(p4 < alpha2) / nSim
```

```
mean(pes1)
mean(pes2)
```

10.8.5 Three conditions replication

```
K <- 3
mu <- c(0, 0.4, 0.2)
n <- 82
sd <- 1
design = paste(K, "b", sep = "")
```

```
design_result <- ANOVA_design(
  design = design,
  n = n,
  mu = mu,
  sd = sd,
  labelnames = c("factor1", "level1", "level2", "level3")
  )
```

```
simulation_result <- ANOVA_power(design_result,
                                 alpha_level = alpha_level,
                                 nsims = nsims,
                                 verbose = FALSE)
```

Table 10.6: Simulated ANOVA Result

	power	effect_size
anova_factor1	61.51	0.0335436

```
exact_result <- ANOVA_exact(design_result,
                            alpha_level = alpha_level,
                            verbose = FALSE)
```

Table 10.7: Exact ANOVA Result

	power	partial_eta_squared	cohen_f	non_centrality
factor1	61.94317	0.0262863	0.1643042	6.56

10.9 Repeated Measures

We can reproduce the same results as Brysbaert finds with his code:

```
design <- "3w"
n <- 75
mu <- c(0, 0.4, 0.4)
sd <- 1
r <- 0.5
labelnames <- c("speed", "fast", "medium", "slow")
```

We create the within design, and run the simulation

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

```
simulation_result <- ANOVA_power(design_result,
                                 alpha_level = alpha_level,
                                 nsims = nsims,
                                 verbose = FALSE)
```

Table 10.8: Simulated ANOVA Result

	power	effect_size
anova_speed	95.25	0.1075912

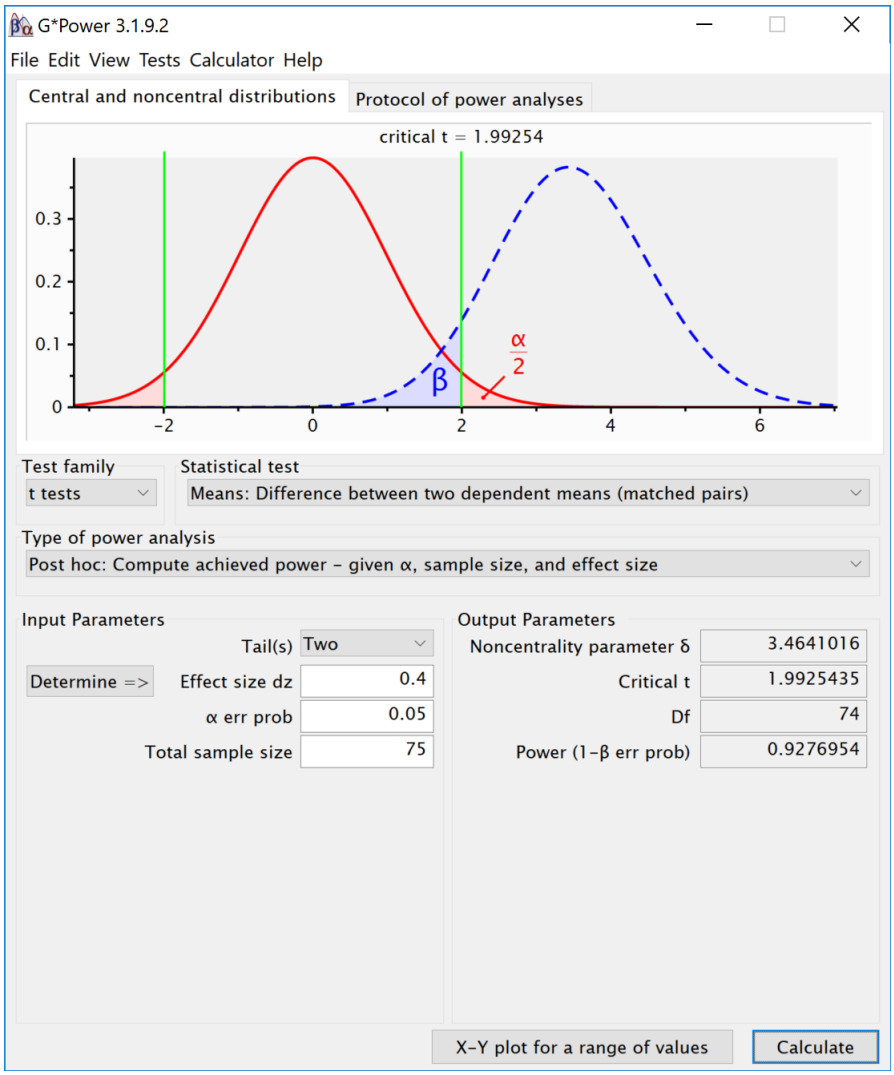
```
exact_result <- ANOVA_exact(design_result,
                            alpha_level = alpha_level,
                            verbose = FALSE)
```

Table 10.9: Exact ANOVA Result

	power	partial_eta_squared	cohen_f	non_centrality
speed	95.29217	0.097561	0.328798	16

Results

The results of the simulation are very similar. Power for the ANOVA F-test is around 95.2%. For the three paired t-tests, power is around 92.7. This is in line with the a-priori power analysis when using Gpower:

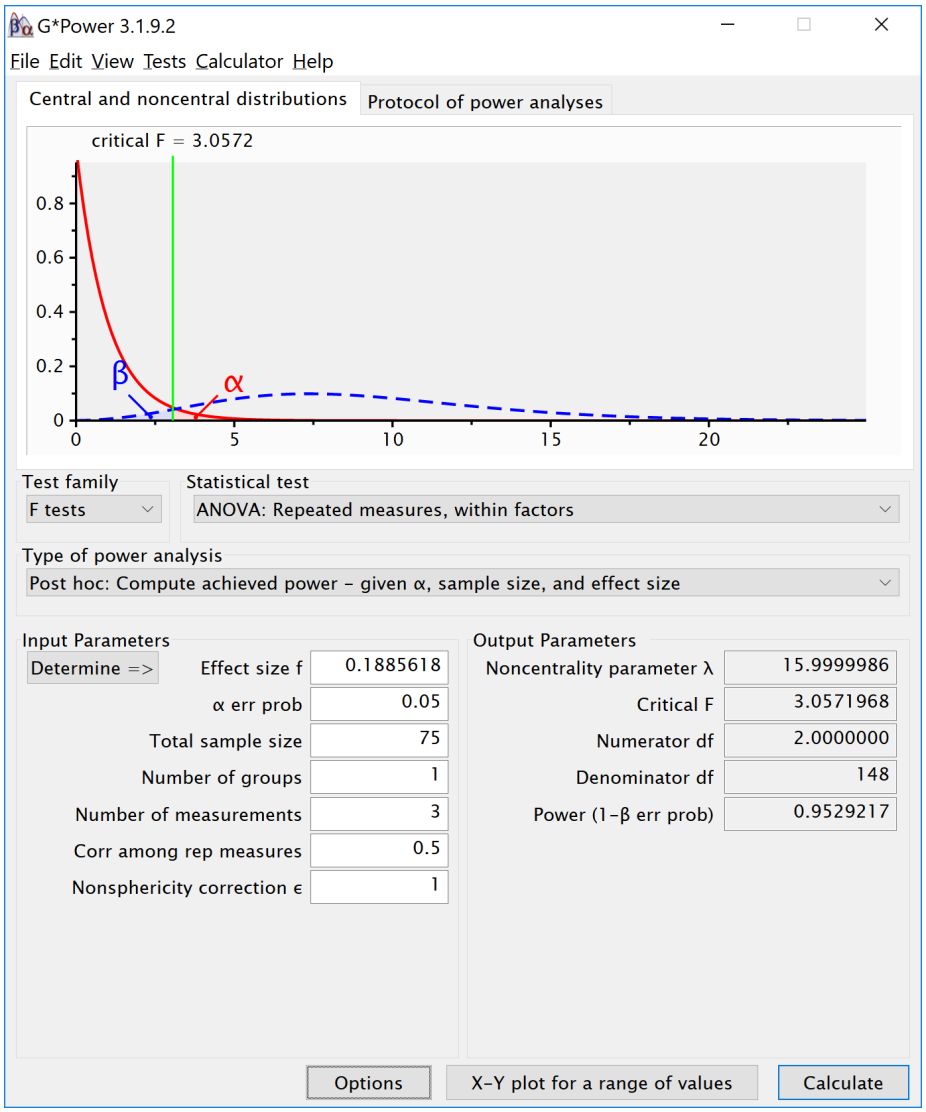


We can perform an post-hoc power analysis in Gpower. We can calculate Cohen´s f based on the means and sd, using our own custom formula.

```
# Our simulation is based onthe following means and sd:
# mu <- c(0, 0.4, 0.4)
# sd <- 1
# Cohen, 1988, formula 8.2.1 and 8.2.2
f <- sqrt(sum((mu - mean(mu)) ^ 2) / length(mu)) / sd
```

```
# We can see why f = 0.5*d.
# Imagine 2 group, mu = 1 and 2
# Grand mean is 1.5,
# we have sqrt(sum(0.5^2 + 0.5^2)/2), or sqrt(0.5/2), = 0.5.
# For Cohen's d we use the difference, 2-1 = 1.
```

The Cohen´s f is 0.1885618. We can enter the f (using the default 'as in G*Power 3.0' in the option window) and enter a sample size of 75, number of groups as 1, number of measurements as 3, correlation as 0.5. This yields:



10.9.1 Reproducing Brysbaert Variation 1: Changing Correlation

```
# give sample size
N = 75
# give effect size d
d1 = .4 #difference between the extremes
d2 = .4 #third condition goes with the highest extreme
# give the correlation between the conditions
r = .6 #increased correlation
# give number of simulations
nSim = nsims
# give alpha levels
alpha1 = .05 #alpha level for the omnibus ANOVA
alpha2 = .05 #also adjusted from original by DL
```

```
# create vectors to store p-values
p1 <- numeric(nSim) #p-value omnibus ANOVA
p2 <- numeric(nSim) #p-value first post hoc test
p3 <- numeric(nSim) #p-value second post hoc test
p4 <- numeric(nSim) #p-value third post hoc test

# define correlation matrix
rho <- cbind(c(1, r, r), c(r, 1, r), c(r, r, 1))
# define participant codes
part <- paste("part",seq(1:N))
for (i in 1:nSim) {

  #for each simulated experiment

  data = mvrnorm(n = N,
  mu = c(0, 0, 0),
  Sigma = rho)
  data[, 2] = data[, 2] + d1
  data[, 3] = data[, 3] + d2
  datalong = c(data[, 1], data[, 2], data[, 3])
  conds = factor(rep(letters[24:26], each = N))
  partID = factor(rep(part, times = 3))
  output <- data.frame(partID, conds, datalong)
  test <- aov(datalong ~ conds + Error(partID / conds),
              data = output)
  tests <- (summary(test))
  p1[i] <- tests$'Error: partID:conds'[[1]]$'Pr(>F)'[[1]]
  p2[i] <- t.test(data[, 1], data[, 2], paired = TRUE)$p.value
```

```
  p3[i] <- t.test(data[, 1], data[, 3], paired = TRUE)$p.value
  p4[i] <- t.test(data[, 2], data[, 3], paired = TRUE)$p.value
}
```

```
sum(p1 < alpha1) / nSim
sum(p2 < alpha2) / nSim
sum(p3 < alpha2) / nSim
sum(p4 < alpha2) / nSim
```

```
## [1] 0.9472
```

```
## [1] 0.9245
```

```
## [1] 0.0503
```

```
## [1] 0.923
```

```
design <- "3w"
n <- 75
mu <- c(0, 0.4, 0.4)
sd <- 1
r <- 0.6
labelnames <- c("SPEED",
                "fast", "medium", "slow")
```

We create the 3-level repeated measures design, and run the simulation.

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

```
simulation_result <- ANOVA_power(design_result,
                                 alpha_level = alpha_level,
                                 nsims = nsims,
                                 verbose = FALSE)
```

```
exact_result <- ANOVA_exact(design_result,
                            alpha_level = alpha_level,
                            verbose = FALSE)
```

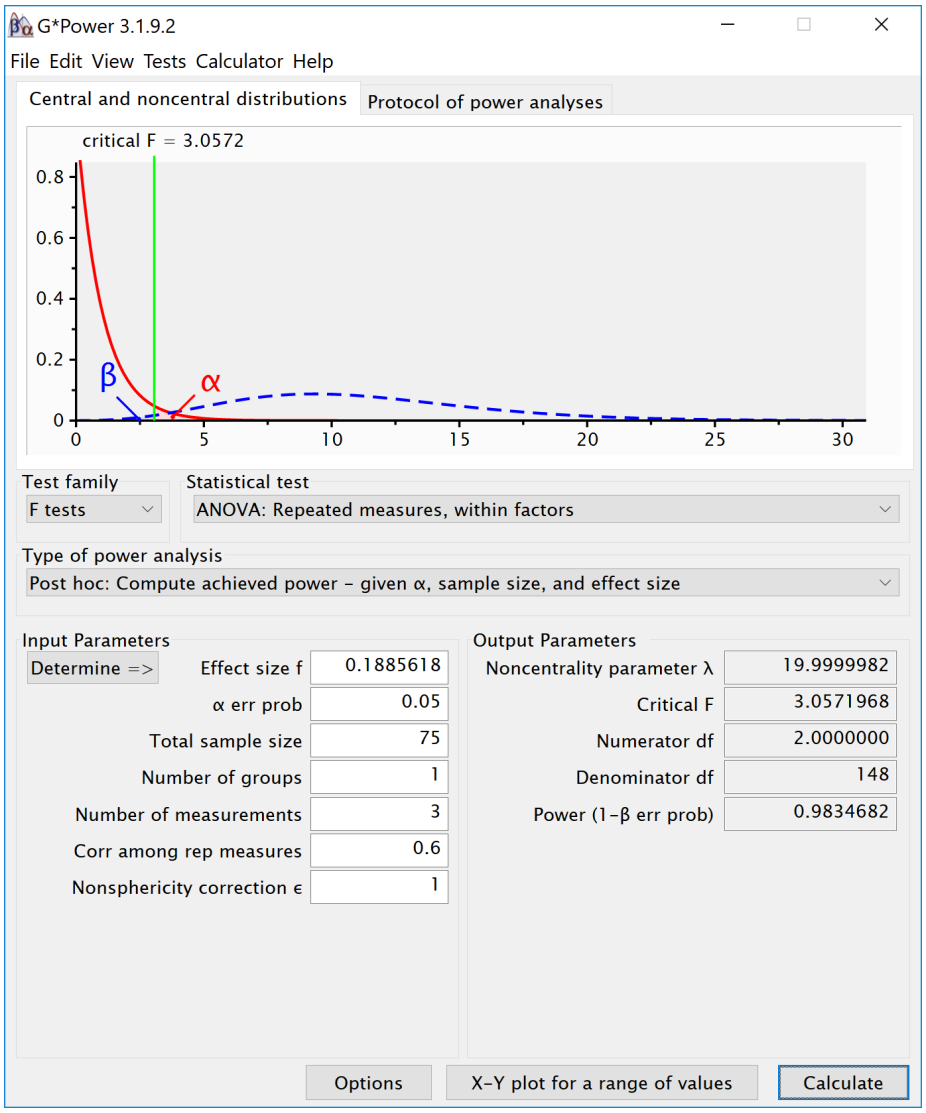
Table 10.10: Simulated ANOVA Result

	power	effect_size
anova_SPEED	98.37	0.1287432

Table 10.11: Exact ANOVA Result

	power	partial_eta_squared	cohen_f	non_centrality
SPEED	98.34682	0.1190476	0.3676073	20

Again, this is similar to GPower for the ANOVA:


G*Power 3.1.9.2
File Edit View Tests Calculator Help
Central and noncentral distributions
Protocol of power analyses
critical F = 3.0572
0.8
0.6
0.4
0.2
0
β
α
0
5
10
15
20
25
30
Test family
F tests
Statistical test
ANOVA: Repeated measures, within factors
Type of power analysis
Post hoc: Compute achieved power – given α, sample size, and effect size
Input Parameters
Determine =>
Effect size f 0.1885618
α err prob 0.05
Total sample size 75
Number of groups 1
Number of measurements 3
Corr among rep measures 0.6
Nonsphericity correction ε 1
Output Parameters
Noncentrality parameter λ 19.9999982
Critical F 3.0571968
Numerator df 2.0000000
Denominator df 148
Power (1-β err prob) 0.9834682
Options
X-Y plot for a range of values
Calculate

Bibliography

Aberson, C. (2019). *pwr2ppl: Power Analyses for Common Designs (Power to the People)*. R package version 0.1.1.

Albers, C. and Lakens, D. (2018). When power analyses based on pilot data are biased: Inaccurate effect size estimators and follow-up bias. *Journal of experimental social psychology*, 74:187–195.

Algina, J. and Keselman, H. (1997). Detecting repeated measures effects with univariate and multivariate statistics. *Psychological Methods*, 2(2):208.

Allaire, J., Xie, Y., McPherson, J., Luraschi, J., Ushey, K., Atkins, A., Wickham, H., Cheng, J., Chang, W., and Iannone, R. (2019). *rmarkdown: Dynamic Documents for R*. R package version 1.18.

Brysbaert, M. (2019). How many participants do we have to include in properly powered experiments? a tutorial of power analysis with reference tables. *Journal of cognition*, 2(1).

Campbell, J. and Thompson, V. A. (2012). Morepower 6.0 for anova with relational confidence intervals and bayesian analysis. *Behavior Research Methods*, 44:1255–1265.

Champely, S. (2018). *pwr: Basic Functions for Power Analysis*. R package version 1.2-2.

Cohen, J. (1988). *Statistical Power Analysis for the Behavioral Sciences*. Lawrence Erlbaum Associates.

Faul, F., Erdfelder, E., Lang, A.-G., and Buchner, A. (2007). G*power 3: A flexible statistical power analysis program for the social, behavioral, and biomedical sciences. *Behavior research methods*, 39(2):175–191.

Hothorn, T., Bretz, F., and Westfall, P. (2019). *multcomp: Simultaneous Inference in General Parametric Models*. R package version 1.4-10.

Maxwell, S., Delaney, H., and Kelley, K. (2004). *Designing Experiments and Analyzing Data: A Model Comparison Perspective*. Number v. 1 in Avec CD. Lawrence Erlbaum Associates.

Muller, K. E. and Peterson, B. L. (1984). Practical methods for computing power in testing the multivariate general linear hypothesis. *Computational Statistics & Data Analysis*, 2(2):143–158.

O'Brien, R. G. and Shieh, G. (1999). Pragmatic, unifying algorithm gives power probabilities for common f tests of the multivariate general linear hypothesis.

Pavot, W. and Diener, E. (1993). The affective and cognitive context of self-reported measures of subjective well-being. *Social Indicators Research*, 28(1):1–20.

Potvin, P. J. and Schutz, R. W. (2000). Statistical power for the two-factor repeated measures anova. *Behavior Research Methods, Instruments, & Computers*, 32(2):347–356.

R Core Team (2019). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria.

SAS (2015). The glmpower procedure. In *SAS/STAT 14.1 User's Guide*, chapter 48, pages 3738–3798. SAS Institute Inc, Cary, NC.

Simonsohn, U. (2014). No-way interactions.

Vonesh, E. F. and Schork, M. A. (1986). Sample sizes in the multivariate analysis of repeated measurements. *Biometrics*, pages 601–610.

Xie, Y. (2019). *bookdown: Authoring Books and Technical Documents with R Markdown*. R package version 0.16.