

Power Analysis with Superpower

Aaron R. Caldwell, Daniël Lakens, Chelsea M. Parlett-Pelleriti, and Guy Prochilo

2021-12-08

Contents

Preface	7
1 Introduction to Power Analysis	9
1.1 Overview of Power Analysis	9
1.2 Error and Power	14
1.3 Effect size	15
1.4 Sample effect size vs. population effect size	16
1.5 Safeguard Effect Size	20
1.6 <i>Post hoc</i> power analysis	21
1.7 The Minimal Detectable Effect Size	25
2 The Experimental Design	29
2.1 ANOVA_design function	29
3 One-Way ANOVA	37
3.1 Introduction	37
3.2 Effect Size Estimates for One-Way ANOVA	57
4 Repeated Measures ANOVA	63
4.1 Part 1	63
4.2 Part 2	76
4.3 Part 3	85
4.4 Part 4	87
4.5 Part 5	97
4.6 Multivariate ANOVA (MANOVA)	110

5	Mixed ANOVA	113
5.1	Simple Mixed Designs	113
5.2	Complex Mixed Designs	119
6	Power for Three-way Interactions	127
7	The ‘No-Way’ Interactions	145
8	Effect of Varying Designs on Power	161
8.1	Within Designs	172
9	Error Control in Exploratory ANOVA	185
10	Analytic Power Functions	193
10.1	One-Way Between Subjects ANOVA	193
10.2	Two-way Between Subject Interaction	196
10.3	3x3 Between Subject ANOVA	198
11	Power Curve	201
11.1	Explore increase in effect size for moderated interactions.	204
11.2	Explore increase in effect size for cross-over interactions.	205
11.3	Explore increase in correlation in moderated interactions.	206
11.4	Increasing correlation in on factor decreases power in second factor	207
11.5	Code to Reproduce Power Curve Figures	208
12	Violations of Assumptions	217
12.1	Violation of Heterogeneity Assumption	217
12.2	Violation of the sphericity assumption	220
13	Estimated Marginal Means	225
13.1	A note of caution	226
13.2	Pairwise Comparisons	226
13.3	Customized <code>emmeans</code> contrasts	228
13.4	Equivalence and non-superiority/-inferiority tests	231
13.5	Joint tests	231
13.6	Monte Carlo Simulations	231

14 Beyond Superpower I: Mixed Models with <code>simr</code>	235
14.1 How does <code>simr</code> work?	235
14.2 To Be Continued...	237
15 Beyond Superpower II: Custom Simulations	239
15.1 A Clinical Trial with a Binary Outcome	239
15.2 Binary RCT with a Interim Analysis	245
15.3 Conclusion on a Binary RCT with Interim Analyses	251
Appendix 1: Direct Comparison to <code>pwr2ppl</code>	253
15.4 Examples from Chapter 5	253
15.5 Examples from Chapter 6	258
15.6 Example from Chapter 7	265
Appendix 2: Direct Comparison to <code>MOREpower</code>	269
Appendix 3: Comparison to Brysbaert	287
15.7 One-Way ANOVA	287
15.8 Repeated Measures	295

This is a compilation of documents for Superpower R package written in **Markdown** (Allaire et al., 2021) and compiled by **Bookdown** (Xie, 2021).

Preface

This book is still being developed. If you have any comments to improve this book, let us know.

This book was compiled with R version 4.1.1 (2021-08-10).


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. Calculating post hoc power is not a useful thing to do for single studies.

This package, and book, expect readers to have some familiarity with R (2021). 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. Reading through the examples in this book, and reproducing them in the Shiny apps, is probably the easiest way to get started with power analyses in **Superpower**.

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 (2021) and **pwr** by Champely (2020)) 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.

Contributors

Daniël Lakens  <https://orcid.org/0000-0002-0247-239X>.

Aaron R. Caldwell  <https://orcid.org/0000-0002-4541-6283>.

Chelsea M. Parlett-Pelleriti  <https://cparlettpelleriti.github.io/>

Guy Prochilo [screenshots/github5.png] <http://www.guyprochilo.com/>

Funding

This work was funded by VIDI Grant 452-17-013 from the Netherlands Organisation for Scientific Research.

Chapter 1

Introduction to Power Analysis

In this first chapter we will introduce some basic concepts of power analysis. If you are already familiar with these concepts please feel free to move onto the next chapter where we discuss the specifics of how **Superpower** can be used.

1.1 Overview of Power Analysis

The goal of Neyman-Pearson hypothesis testing, often divisively labeled null hypothesis significance testing (NHST), is to determine whether the null hypothesis (H_0) can be rejected. *Statistical power* is the probability of rejecting the null hypothesis when it is false. And a *power analysis* is a sample size planning procedure performed for a future *yet-to-be-conducted* study that will tell us how many observations are required to reject the null hypothesis with sufficiently high probability.

For example, if you want to plan a study with 80% power, a power analysis will answer the question:

“If I were to repeat my experiment many thousands of times, what sample size will allow me to reject the null hypothesis on 80% of these occasions?”

Let’s illustrate this with an example. Imagine we have conducted a literature review and have estimated that the true standardized mean difference between two groups is $\delta = 0.50$. That is, the mean of each group differs by 0.50 standard deviations. Assuming that our estimate of this population effect size is accurate,

how many observations per group should we collect to reject a null hypothesis of no difference with 80% probability?

Probability is the long-run frequency at which an event occurs (i.e., after many repeated samples). Therefore, if we want to determine the sample size that allows us to reject the null hypothesis with 80% probability, we can run a simulation of our study at different sample sizes where each study is repeated many thousands of times. Power will be the proportion of these studies, at each sample size, that reject the null hypothesis.

```
# Define our power analysis simulation function
power_sim <- function(n, d, nsims, seed = TRUE, print_plot = TRUE){

  if(seed == TRUE){set.seed(2)}

  # For sample size `n`, we want to perform `nsims`
  ## number of simulations of our experiment using effect size `d`
  replicate(n = nsims, exp = {
    x1 = rnorm(n, d, 1)
    x2 = rnorm(n, 0, 1)
    t.test(x = x1, y = x2, var.equal = TRUE)$p.value}) -> ps

  # Power for a given `n` is the long-run probability
  ## of rejecting the null hypothesis when it is true.
  # In our simulation, it is the sum of all `nsims` simulations
  ## of our study that yield  $p < .05$  divided by the number of `nsims` performed.
  power = sum(ps < .05) / nsims
  result = cbind(n, power)

  # Visualize the results as a plot
  ps %>%
    as.data.frame() %>%
    ggplot(aes(x = .)) +
    geom_histogram(binwidth = 0.01) +
    scale_x_continuous(breaks = scales::pretty_breaks(n = 15)) +
    geom_vline(xintercept = 0.05,
               linetype = "dashed",
               color = "red") +
    xlab(label = expression(italic(p) ~ "value")) +
    ylab(label = "Frequency") +
    labs(caption = paste(
      "Power =",
      power * 100,
      "% \n",
      sum(ps < .05),
      "of",
```

```

    nsims,
    "simulations yield p < .05"
  )) +
  ggtitle(label = expr(paste(
    "N = ", !!(n * 2),
    " (", n[1], " = ", !!n, "; ", n[2], " = ", !!n, ")")
  ))) -> p1

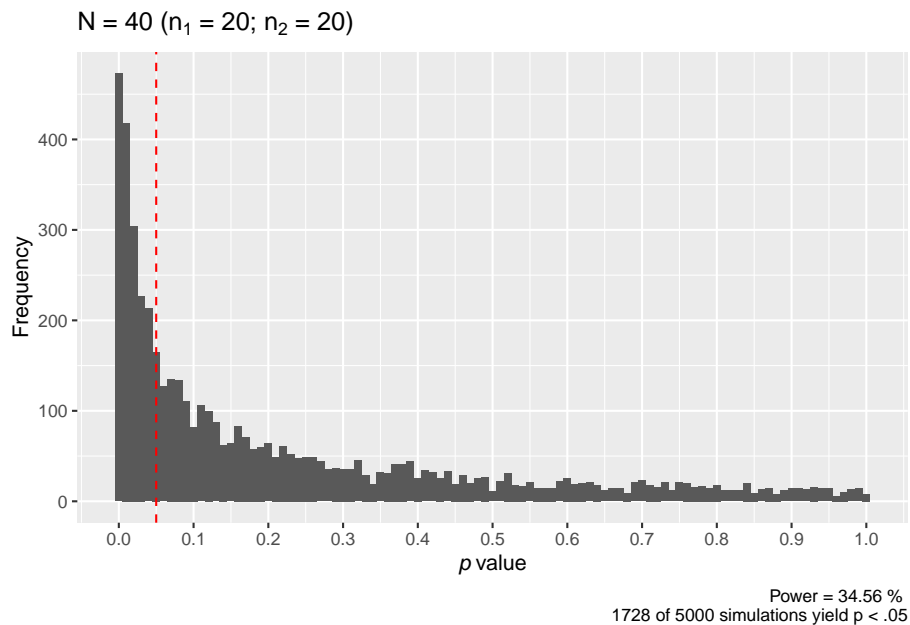
# Print our plot
if(print_plot == TRUE){print(p1)}

# Return the result variable
return(result)
}

```

We can use this function to compute power for any given sample size we like. For example, let's determine the power of detecting $\delta = 0.50$ if we were to collect $n = 20$ observations per group with 5000 simulations.

```
power_sim(n = 20, d = 0.50, nsims = 5000, seed = TRUE)
```

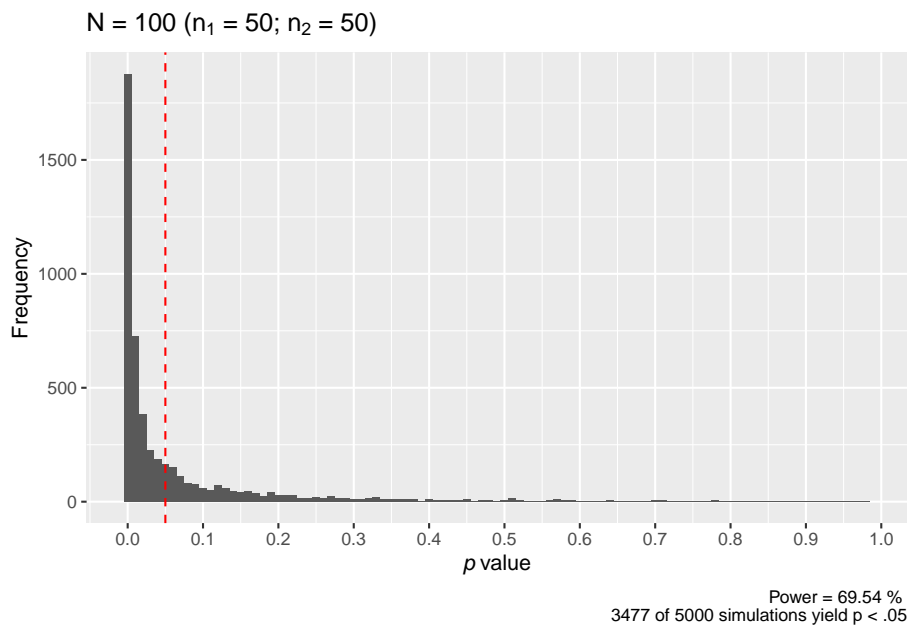


```
##      n power
## [1,] 20 0.3456
```

In the above plot, the red dashed line marks $p = .05$. As we can see, the majority of our simulated studies report p values greater than .05. In fact, only 1728 of our 5000 simulated studies rejected the null hypothesis with this sample size. Recall that probability is the long-run frequency at which an event occurs (i.e., after many repeated samples), and power is the probability of rejecting the null hypothesis when it is false. If we divide the 1728 correct rejections by the 5000 total simulations we get a final power estimate for this sample size: 34.56%. That is, after many, many repetitions of our study with a sample size of $n = 20$ per group, we only reject the null hypothesis 34.56% of the time. If we were planning a study to detect $\delta = 0.50$, there is not a high probability that we will reject the null hypothesis with only $n = 20$ per group. We probably wouldn't want to use this sample size.

Let's try $n = 50$ per group.

```
power_sim(n = 50, d = 0.50, nsims = 5000, seed = TRUE)
```

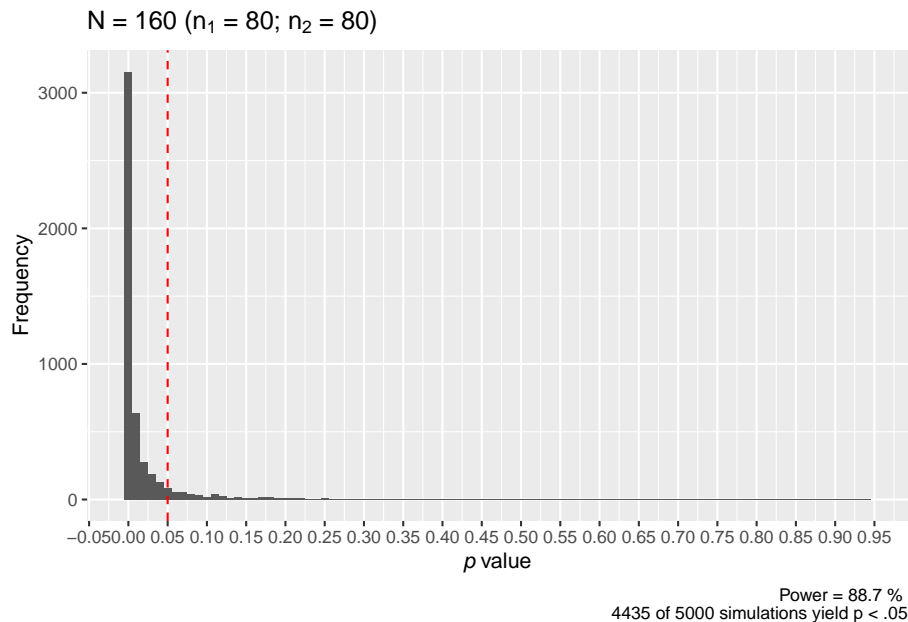


```
##      n power
## [1,] 50 0.6954
```

As we can see in the plot of our results, 3477 studies of our 5000 simulated studies rejected the null hypothesis. This corresponds to a power of 69.54%. Is rejecting the null hypothesis around 70% of the time satisfactory for you? It isn't for most scientists.

Let's try again with $n = 80$ per group.

```
power_sim(n = 80, d = 0.50, nsims = 5000, seed = TRUE)
```



```
##      n power
## [1,] 80 0.887
```

This time 4435 of our simulated 5000 studies rejected the null hypothesis, corresponding to a power of 88.7%. This is usually more than enough power for most scientists to consider satisfactory. We might even consider examining power for lower sample sizes.

And that's it. That's all there is to power analysis. Power is simply the probability of rejecting the null hypothesis when it is false. And a power analysis will tell us how many observations are needed to reject this null hypothesis with a sufficiently high probability. Analytic solutions have been determined for many of the common tests you will encounter in psychological science, so simulations like those above are not always necessary. The analytic solution for the t test is implemented by the `pwr::pwr.t.test()` function as part of the `pwr` package. In fact, if we input the parameters for our last power analysis into `pwr::pwr.t.test()`, we will obtain almost identical power estimates for $n = 80$ per group:

```
pwr::pwr.t.test(n = 80, d = 0.50, sig.level = 0.05)
```

```
##
```

```
##      Two-sample t test power calculation
##
##              n = 80
##              d = 0.5
##      sig.level = 0.05
##      power     = 0.8816025
##      alternative = two.sided
##
## NOTE: n is number in each group
```

For $n = 80$ per group our simulation reported power = 88.7%, and the analytic solution reported power = 88.2%. If we were to increase the number of simulations we performed (perhaps to many hundreds of thousands), these two results would converge even closer.

Power analysis is essential before embarking on any research study. As we will discuss in subsequent section, with high power you are more likely to obtain findings that are trustworthy and you are more likely to draw correct inferences from your results. This is important, not only for getting your findings published, but also for building a reliable literature that is built on cumulative knowledge.

1.2 Error and Power

In this Neyman-Pearson framework there are two kinds of errors that we must account for when designing an experimental study. These are:

1. Type I Error: rejecting the null hypothesis when it is true (also known as false positives or α).
2. Type II Error: failing to reject the null hypothesis when it is true (also known as false negatives or β).

Recall that power is the probability of rejecting the null hypothesis when it is false. This corresponds to $1 - \beta$. Power will depend crucially on sample size, effect size, and the Type I error rates that you are willing to accept.

The typical power analysis parameters in psychology are as follows:

- $\beta = .20$
 - This means we are willing to make a Type II error 20% of the time (i.e., 80% power).
- $\alpha = .05$

- This means we are willing to make a Type I error only 5% of the time (i.e., significance $< .05$).
- $1 - \beta = .80$:
 - This means we want to be able to correctly reject the null hypothesis, with a effect size *at least as large* as hypothesized, when it is false at least 80% of the time.

The values of each of these parameters reflect our relative concerns regarding the cost of Type I and Type II errors. That is, we regard Type I errors as *four times* more serious than making a Type II error:

$$\frac{\beta}{\alpha} = \frac{.20}{.05} = 4$$

This may not always be the case and we may want to design a study where the errors are balanced:

$$\frac{\beta}{\alpha} = \frac{.05}{.05} = 1$$

And other times we may be willing to increase our Type I error rate slightly in order to reduce the Type II error rate. For example, we may raise our Type I error rate slightly in order to reduce the ratio of errors.

$$\frac{\beta}{\alpha} = \frac{.15}{.075} = 2$$

1.3 Effect size

For any given α level, the power of a test will increase as the effect size increases. An effect size is a measure of the magnitude of some phenomenon. This might include a measure of the difference between two groups (e.g., the mean), the strength of association between variables (e.g., r : the Pearson's correlation coefficient), or the linear relationship between a dependent variable and one or more predictors (e.g., b : the unstandardized regression coefficient).

A very common way to quantify effect size in psychological science is Cohen's d . This quantity represents the standardized difference between two means. It is computed as:

$$d = \frac{M_1 - M_2}{SD_{pool}}$$

Where M_1 is the mean of group one, M_2 is the mean of group two, and SD_{pool} is the pooled standard deviation of group one and two. Cohen's d is sometimes given descriptive labels of small, moderate, and large that correspond to 0.2, 0.5, and 0.8, respectively. However, these descriptives can differ across different measurements. We would strongly advocate against using these “default” interpretation scales and instead understand your phenomena/measurements well enough to understand when an effect is meaningful (Caldwell and Vigotsky, 2020). Some additional common effect sizes alongside their descriptive magnitudes are given in the following table:

Effect	Small	Moderate	Large
$*d^*$	0.20	0.50	0.80
$*f^*$	0.10	0.24	0.37
$*f^*$	0.10	0.25	0.40
η^2	0.01	0.06	0.14

1.4 Sample effect size vs. population effect size

When we run a study what we are doing is collecting a sample of the total population. It follows that any effect sizes computed on the sample data are an *estimate of the true population effect size*. This is an important consideration to keep in mind when performing power analyses. As Albers and Lakens (2018) demonstrate, researchers are very likely to miss out on interesting effects when studies are powered to detect previously observed effect sizes.

To make this clear, imagine two groups with a known population standardized mean difference of *Cohen's* $\delta = 0.5$. (We use the Greek symbol δ to refer to the population effect size, and the lower case d to refer to the sample effect size). Now imagine that we have take a random sample of 20 measurements from each of these groups. We can simulate this in R using `rnorm()`:

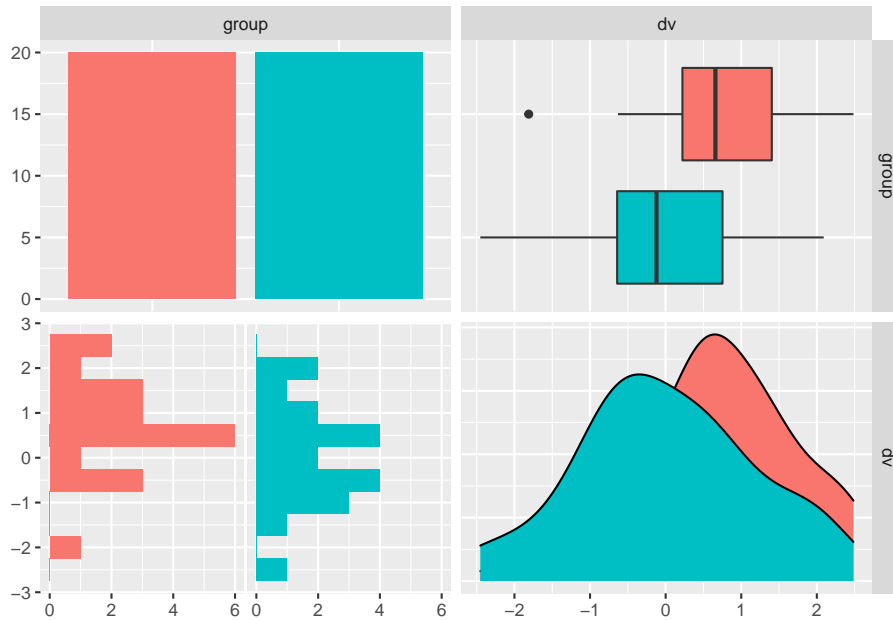
```
# Set random seed for reproducibility
set.seed(2)

# Simulate datasets with mean difference of 0.5
group_1 = rnorm(n = 20, mean = 0.5, sd = 1)
group_2 = rnorm(n = 20, mean = 0, sd = 1)

# Visualize the data
data.frame(group_1, group_2) %>%
  tidyr::pivot_longer(
    names_to = "group",
    values_to = "dv",
    cols = everything()) %>%
  GGally::ggpairs(ggplot2::aes(colour = group),
```



```
lower = list(combo = wrap(ggally_facethist,
                          binwidth = 0.5)))
```



The above plot visualizes Group 1 (red) and Group 2 (teal) from left to right as: (1) counts, (2) box plots, (3) rotated histograms, and (4) density distributions. We can see clearly that the mean of Group 1 is greater than Group 2.

Now let's compute the sample Cohen's d using the formula we described earlier. We will also compute a 95% CI on Cohen's d using the exact method implemented by `MBESS::ci.smd()`. The 95% CI will give us all plausible values of Cohen's δ :

```
# Determine N for each group
n1 = length(group_1)
n2 = length(group_2)

# Determine standard deviation for each group
sd1 = sd(group_1)
sd2 = sd(group_2)

# Compute pooled standard deviation
sd_pool = sqrt((sd1 * sd1 * (n1 - 1) + sd2 * sd2 * (n2 - 1)) / (n1 + n2 - 2))

# Determine the difference between the means
m_diff = mean(group_1) - mean(group_2)
```

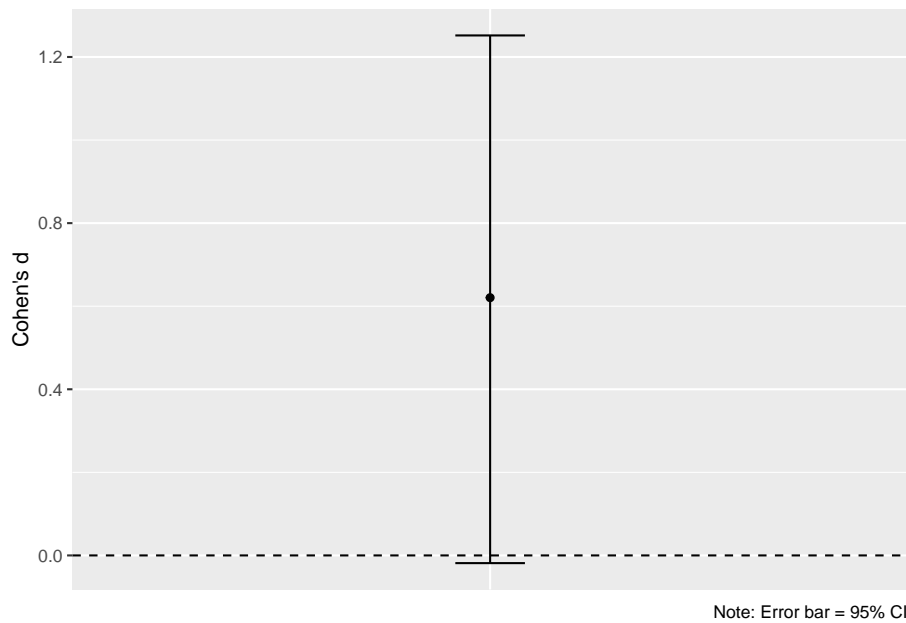
```
# Compute Cohen's d
d = m_diff / sd_pool

# Compute 95% CI on Cohen's d
(d_ci = MBESS::ci.smd(smd = d, n.1 = n1, n.2 = n2, conf.level = 0.95))
```

```
## $Lower.Conf.Limit.smd
## [1] -0.01853137
##
## $smd
## [1] 0.6205506
##
## $Upper.Conf.Limit.smd
## [1] 1.251871
```

Recall that another name for Cohen's d is the standardized mean difference (here, `smd`). Cohen's d on our data is 0.62, and the 95% CI ranges from -0.02 to 1.25. Let's visualize this as a plot (because we love plots):

```
# Plot the result
d_ci %>%
  as.data.frame() %>%
  ggplot(aes(x = "Cohen's d")) +
  geom_point(aes(y = smd)) +
  geom_errorbar(aes(ymin = Lower.Conf.Limit.smd,
                    ymax = Upper.Conf.Limit.smd), width = 0.1) +
  geom_hline(yintercept = 0, linetype = "dashed") +
  theme(axis.title.x = element_blank(),
        axis.text.x = element_blank(),
        axis.ticks.x = element_blank()) +
  ylab(label = "Cohen's d") +
  labs(caption = "Note: Error bar = 95% CI")
```



So what do these results tell us? Although we know with certainty that the population effect size is a Cohen's δ of 0.5, our sample effect size is $d = 0.62$. Furthermore, the wide confidence interval on d means that our sample effect size has considerable uncertainty. All values in the confidence interval can be considered plausible values for the true population effect size. Indeed, in this unusual situation where we know the population effect size with certainty, we can see that the confidence interval on the sample effect size actually captures the true value (this will not always be the case).

The important take-home message for power analysis is that we want to use **population effect size** and **not the sample effect size** as our effect size input. When we identify sample effect sizes in the literature (which may be through individual studies, meta-analysis, or when appropriate, through internal pilot studies) we must be mindful that the sample effect size may not be equal to the true population effect size. If we were to use the sample effect size in the above example ($d = 0.62$) as our estimate of the population effect size in a power analysis, this would have resulted in much less power than we desired. The sample effect size has **overestimated** the true population effect size. Uncertainty in the population effect size is captured by the 95% CI on the sample effect size. It is wise to take this into consideration when using previously reported effect sizes for a power analysis.

1.5 Safeguard Effect Size

Our estimate of the population effect size is an educated guess based on the best information available to us (much like the written hypotheses we have for a study). It's accuracy can have critical implications for the sample size required to attain a certain level of power. For this reason, researchers should routinely consider varying their effect size estimate and assessing the impact this has on their power analysis. If we are using sample effect sizes reported in the literature to estimate the population effect size, one excellent recommendation is to formally consider the uncertainty of sample effect sizes. That is, take into account the 95% CI on these reported effects.

Consider once again the data reported in the previous section. If you skipped ahead, we simulated a study where the true population effect size was Cohen's $\delta = 0.5$. However, we obtained a sample estimate of this effect size as Cohen's $d = 0.62$ in a sample of $N = 40$ observations. Were we to use $d = 0.62$ as our power analysis input, this would result in much less power than anticipated to detect an effect size of $\delta = 0.5$. So what should we do?

Perugini et al. (2014) suggest we take into account the uncertainty of the sample effect size. To do so, they suggest for the researcher to compute the lower absolute limit of a 60% CI on the sample effect. This value would then be used as a *safeguard effect size* and inputted into a power analysis. Why 60%? Well, the lower limit of a 60% CI corresponds to a one-sided 80% CI. As Perugini et al. (2014) describe, this implies a 20% risk that the true population effect size is lower than this limit, and correspondingly, 80% assurance that the population effect size is equal to or greater than this limit. We focus on the lower boundary because of the asymmetry of overestimating compared to underestimating the population effect size in a power analysis.

Let's do this for our previous sample estimate of Cohen's d . We can compute a 60% CI by changing the `conf.level` parameter to 0.60 in `MBESS::ci.smd()`:

```
(d_safeguard = MBESS::ci.smd(smd = d, n.1 = n1, n.2 = n2, conf.level = 0.60))
```

```
## $Lower.Conf.Limit.smd
## [1] 0.3437032
##
## $smd
## [1] 0.6205506
##
## $Upper.Conf.Limit.smd
## [1] 0.8892221
```

Using our sample Cohen's $d = 0.62$ in a sample of $N = 40$, the lower absolute limit of a 60% CI on this effect is 0.34. We would therefore use the safeguard

sample effect size as our best estimate of the population effect size in our power analysis. While we have somewhat underestimated the population effect size in this scenario, we have also ensured we have sufficient power to detect the population effect size. Sufficient power means we are more likely to draw correct inferences from our data.

1.6 *Post hoc* power analysis

All authors of this book are in agreement: *post hoc* power analysis is useless. Many statisticians have made this abundantly clear (Gelman, 2019; Althouse, 2021).

Sometimes editors or reviewers of manuscripts will insist that an author perform a post-experiment power calculation in order to interpret non-significant findings. These *post hoc* or *observed* power calculations are requested on the basis of explaining the observed data rather than planning some future experiment. These analyses are intended to answer the question: “*On the basis of my observed effect size, did I have enough power to reject my null hypothesis?*”. To avoid needless suspense, the answer is always: no.

We will discuss two reasons why *post hoc* power analyses are a problematic practice.

1.6.1 The sample effect size is *not* the population effect size

As discussed at length in previous sections, the sample effect size is *not* the population effect size. For this simple reason alone, any *post hoc* power calculation will be an inaccurate estimate of the true power of an experiment. To illustrate, consider once more our simulated data set. If you skipped ahead, we simulated random sample of $N = 40$ from two populations ($n_1 = 20$, $n_2 = 20$) where the true population effect size was Cohen’s $\delta = 0.5$. However, based on random sampling alone, we obtained a sample effect size of Cohen’s $d = 0.62$.

If we perform a *t*-test on these data, we will find that the data are compatible with the null hypothesis of no difference. That is, $p > .05$. This is exactly the kind of scenario targeted by proponents of *post hoc* power analysis. Let’s perform the test:

```
t.test(x = group_1, y = group_2, var.equal = TRUE)
```

```
##
## Two Sample t-test
##
```

```
## data: group_1 and group_2
## t = 1.9624, df = 38, p-value = 0.05707
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
## -0.02199628 1.41348186
## sample estimates:
##      mean of x      mean of y
## 0.6954610335 -0.0002817572
```

We obtain $p = .057$. Now let's compute a *post hoc* power calculation on the sample effect size. As will be explained in later sections, we can compute different kinds of power analyses by omitting a single parameter in a power function. Here, we omit `power` and ask `pwr::pwr.t.test()` to estimate `power` on the basis of $n = 20$ per group (total $N = 40$) using $d = 0.62$ and $\alpha = .05$ (two-sided):

```
# Remind ourselves of the value of Cohen's d for these data:
(d)
```

```
## [1] 0.6205506
```

```
# Perform the post hoc power analysis
## by excluding `power`.
## Note: `n` refers to sample size 'per group':
pwr::pwr.t.test(n = 20, d = d, sig.level = 0.05)
```

```
##
##      Two-sample t test power calculation
##
##              n = 20
##              d = 0.6205506
##      sig.level = 0.05
##      power = 0.4811947
##      alternative = two.sided
##
## NOTE: n is number in *each* group
```

Our estimate of *post hoc* power is 48%. However, we know that the true population effect size is $\delta = 0.5$. With this knowledge, we can compute the *true* power of our test as:

```
# Define delta
delta = 0.5

# Perform the post hoc power analysis by excluding `power`:
pwr::pwr.t.test(n = 20, d = delta, sig.level = 0.05)
```

```
##
##      Two-sample t test power calculation
##
##              n = 20
##              d = 0.5
##      sig.level = 0.05
##      power = 0.337939
##      alternative = two.sided
##
## NOTE: n is number in each group
```

The true power based on the population effect size is 34%. This is much lower than our *post hoc* power analysis has lead us to believe. As such, *post hoc* power analysis will not tell you whether you had enough power to reject the null hypothesis. That is, unless you are willing to assume your sample effect size *is* the population effect size. This is something a single study cannot tell you. And this is something that is clearly incorrect in our example.

1.6.2 *Post hoc* power is merely a transformation of your obtained p value

Post hoc power analyses are simply a re-expression of the obtained p value. As Lenth (2007) describes, to compute *post hoc* power all we need (beyond our chosen α) is the p value of the test and associated degrees of freedom. For this reason, *post hoc* power does not provide more information than that which is already provided by our p value.

Recall from above that our obtained p value in our simulated data is .057. Lenth (2007) provides a simple R function to compute *post hoc* power from the obtained p value from an independent groups t test as follows:

```
# Define power function
power <- function(p_val, deg_f, alpha){

  delta = qt(p = 1 - p_val / 2, df = deg_f)
  crit_val = qt(p = 1 - alpha / 2, df = deg_f)
  power = 1 - pt(q = crit_val, df = deg_f, ncp = delta) + pt(-crit_val, deg_f, delta)

  return(power)
}
```

Let's apply this function to our obtained p value and associated degrees of freedom:

```
# Apply power function to our data
power(p_val = 0.05707, deg_f = 38, alpha = .05)
```

```
## [1] 0.4812096
```

Using only the obtained p value and test degrees of freedom we obtain a *post hoc* power of 48%. This is identical to our power analysis based on the observed effect size (with error due to rounding of our p value). We could do this with any combination of obtained p value and degrees of freedom, and obtain exactly the result of a *post hoc* power analysis.

In general, the power of a test yielding exactly $p = .05$ will be around 50% (and inches ever closer to exactly 50% as degrees of freedom approaches infinity). If $p > .05$, power (in general) will be less than 50%. To demonstrate, let's test this with our function using degrees of freedom from five to infinity against p values of .05, .10, and .20:

```
v = list(seq(20,200,20), Inf)
map_df(v, ~ data.frame(.,
  power(p_val = .05, deg_f = ., alpha = .05),
  power(p_val = 0.10, deg_f = ., alpha = .05),
  power(p_val = 0.20, deg_f = ., alpha = .05))) %>%
  `colnames<-`(c("Degrees of Freedom",
    "Power ($p$ = .05)",
    "Power ($p$ = .10)",
    "Power ($p$ = .20)")) %>%
  knitr::kable(escape = FALSE) %>%
  kableExtra::kable_styling(bootstrap_options = "striped",
    full_width = T,
    position = "left")
```

Degrees of Freedom	Power ($p = .05$)	Power ($p = .10$)	Power ($p = .20$)
20	0.5101846	0.3754240	0.2432919
40	0.5050221	0.3759611	0.2463210
60	0.5033422	0.3761399	0.2473313
80	0.5025099	0.3762290	0.2478362
100	0.5020131	0.3762824	0.2481391
120	0.5016829	0.3763180	0.2483409
140	0.5014475	0.3763434	0.2484851
160	0.5012713	0.3763624	0.2485932
180	0.5011344	0.3763772	0.2486773
200	0.5010250	0.3763890	0.2487446
Inf	0.5000443	0.3764951	0.2493496

As we can see, whenever a test is non-significant (as it was in our example) the power will usually be less than 50%. Therefore, whenever you are motivated to perform a *post hoc* power analysis to explain a non-significant result, it is an empty question of whether *post hoc* power is high. The answer, as we alluded to earlier, is always *no*. In our simulated dataset we already knew that $p > .05$, so of course this must mean we had low power. Otherwise we would not have obtained $p > .05$! *Post hoc* power is a mere transformation of the p value and provides us with no new information than that we already knew.

Power calculations are useful for designing experiments and planning studies. They are not useful once the data is collected and the final analyses have been completed. Keep this in mind when performing your own power analyses, or attempting to explain non-significant results.

1.7 The Minimal Detectable Effect Size

In an ideal world all studies would proceed with sample sizes that give sufficient power to detect population effects. However, we operate within a world of limited resources, and there are often times where researchers are unable to increase sample sizes to attain a specific power. If an experiment is performed in this context, rather than calculating something like *post hoc* power, it is much more informative to compute the smallest effect size that could be detected in a study of this particular sample size. This requires fixing the α level and sample size, and computing the critical test statistic for the proposed (or conducted) statistical analysis.

We will once again draw on the simulated dataset we introduced above. This dataset was generated by drawing a random sample of $N = 40$ from two populations ($n_1 = 20$, $n_2 = 20$) where the true population effect size was Cohen's $\delta = 0.5$. However, the obtained sample effect size was Cohen's $d = 0.62$. Furthermore, using an independent groups t -test ($\alpha = .05$; two-sided) we found that this effect was non-significant ($p = .057$). Let's compute what minimum t value was necessary to attain statistical significance. This is the critical t value, which can be computed with the `qt()` function in R:

```
n1 = length(group_1)
n2 = length(group_2)
alpha = .05
deg_f = n1 + n2 - 2
(t_crit = qt(p = 1 - (alpha/2), df = deg_f))
```

```
## [1] 2.024394
```

The critical t value for our test is 2.024. This means that any obtained t statistic lower than this value will yield a statistically non-significant result (i.e., $p > .05$). Our t statistic was 1.96, so it is unsurprising that our result was non-significant.

There is an algebraic solution to converting an obtained independent groups t statistic (two sample t -test) to a Cohen's d value:

$$d = t * \sqrt{\frac{1}{n_1} + \frac{1}{n_2}}$$

Here, t is the obtained t statistic from an independent groups t test, and n_1 and n_2 are the sample sizes of each group, respectively. We can confirm that $d = 0.62$ in our data on the basis of the t statistic and sample sizes alone:

```
# Store the t test result
result = t.test(x = group_1, y = group_2, var.equal = TRUE)

# Extract the t value
(t_val = as.numeric(result$statistic))
```

```
## [1] 1.962353
```

```
# Store the sample size of each group
n1 = length(group_1)
n2 = length(group_2)

# Compute Cohen's d from the t statistic
## and sample sizes alone
t_val * sqrt((1/n1 + 1/n2))
```

```
## [1] 0.6205506
```

$d = 0.62$. Magic. We can take advantage of this formula and, instead of inputting the obtained t statistic, we can input the critical t statistic for a study of this size. In doing so, we will obtain a critical d value:

```
# Remind ourselves of what t_crit is
(t_crit)
```

```
## [1] 2.024394
```

```
# Compute the critical d value
t_crit * sqrt((1/n1 + 1/n2))
```

```
## [1] 0.6401696
```

In a study of this size the critical d value is 0.64. This means that any obtained d statistic lower or equal to this value will yield a non-significant result (i.e., $p > .05$). As we obtained $d = 0.62$ as our sample effect size, it is unsurprising that our result is non-significant. The critical d value of 0.64 is the minimal detectable effect size in a study of this size.

What does this all mean? Well, if a researcher believes that the true population effect size is less than 0.62 (perhaps, $\delta = 0.5$), they should be un-surprised if they find a non-significant finding in $N = 40$. The minimum effect size they could have hoped to have detected was $d = 0.66$. The non-significant result is explained by the fact that a study of this sample size cannot detect effect sizes below the critical d value. In later chapters we will discuss how to use **Superpower** to address these issues using tools such as the `plot_power` function and the `morey_plot` functions.

Chapter 2

The Experimental Design

Now that we have introduced the basics of what constitutes a power analysis we can move onto how to use **Superpower** as tool for power analyses. The first step is to define and specify the details of the experimental design. Currently, **Superpower** only provides power analysis capabilities for ANOVAs (and MANOVA for repeated measures) so all designs are currently defined by the `ANOVA_design` function.

2.1 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. Can be specified as a single value (when all conditions have the same sample size), or as a vector of values specifying the sample size for each 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). Can be specified as a single value (when all conditions have the same standard deviation), or as a vector of values specifying the standard deviation for each condition.
5. **r**: the correlation(s) for within designs (or 0 for between designs). Can be specified as a single value (when all pairs of variables have the same correlation), or as a correlation matrix specifying the correlation between each pair.
6. **labelnames**: This is an optional 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`)

2.1.1 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`).

2.1.2 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).

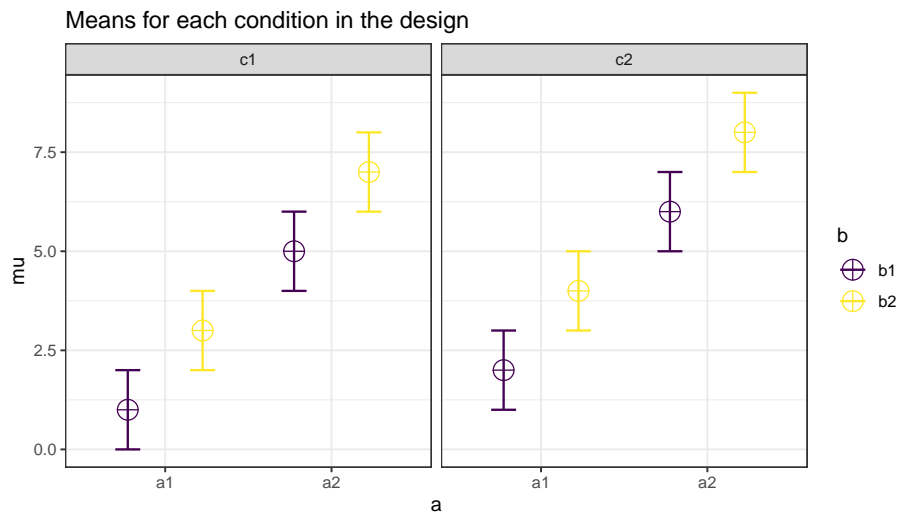
If your design has just one factor, just enter the means in the same order as the `labelnames` (see below). For more factors, note the general pattern in the example below. Means are entered in the following order for a 3 factors design (each with 2 levels):

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.



2.1.3 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")`

2.1.4 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
```


Table 2.1: Correlation Matrix

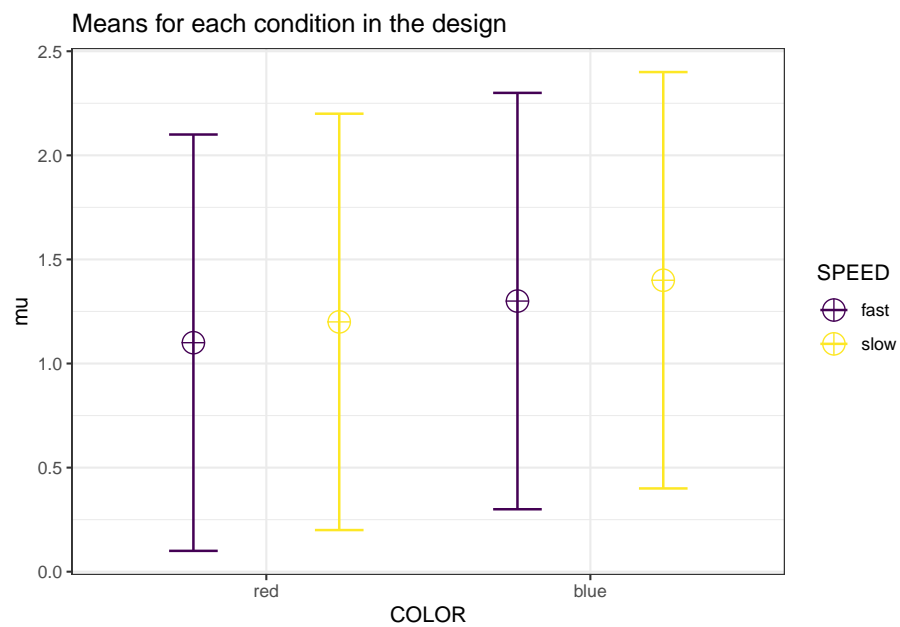
	a1_b1	a1_b2	a2_b1	a2_b2
a1_b1	1.00	0.91	0.92	0.93
a1_b2	0.91	1.00	0.94	0.95
a2_b1	0.92	0.94	1.00	0.96
a2_b2	0.93	0.95	0.96	1.00

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).

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 printing it from the `design_result` object to check if it was entered the way we wanted:

```
design_result$cor_mat
```

Table 2.2: 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 is 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$sigmatix
```

Table 2.3: 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

2.1.5 Specifying the sample size

You can set the sample size **per condition** by setting a value for `n`. It is beneficial to have a fully balanced design (the same sample size in each condition) in case of any violations of assumptions (such as homogeneity of variances), but this might not always be possible in real life, and it can be more efficient to collect a larger sample size in some conditions (e.g., if the population from which other participants are drawn is more difficult to collect). Superpower allows you to enter a single value (the same sample size in each condition) or a vector of values (different sample sizes in each condition).

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

for a **2b*2b** between subjects design, we could also specify the sample size as **n** = **c**(60, 80, 80, 60) if we plan to collect 60 participants in 2 conditions and 80 participants in 2 other conditions.

2.1.6 Specifying the standard deviation

You can set the standard deviation(s) by setting a value of **sd**. You can enter either a single standard deviation for all conditions, or you can **violate the assumption of homogeneity of variance** by entering a vector with different standard deviations for each conditions (e.g., **sd** = **c**(0.8, 1.2, 0.92, 1.12) for a **2b*2b** design. Violation the homogeneity of variances assumption 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 3

One-Way ANOVA

3.1 Introduction

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). Although 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.

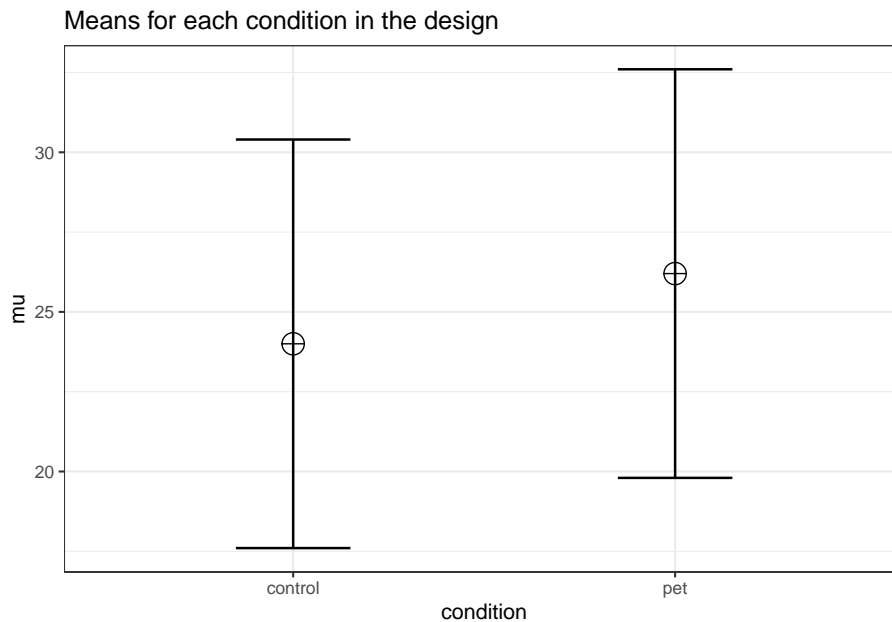
3.1.1 Two conditions

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-satisfaction 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. There are 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 100 people in each condition
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
# You should think carefully about how to justify your alpha level.
# We will use 0.05 in most examples, but do not support using a 0.05 alpha level for a

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

```

Table 3.1: Simulated ANOVA Result

	power	effect_size
anova_condition	67.79	0.0335176

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

Table 3.2: 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 (note: in the simulation, values will differ slightly, but with a large number of simulations, power estimates will be identical), 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` (2020) 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 $f = 1/2d$. 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 `pwr.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 very 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 have 90% power for the ANOVA with 100 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 2 groups and 100 participants in each group, in order to achieve 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
```

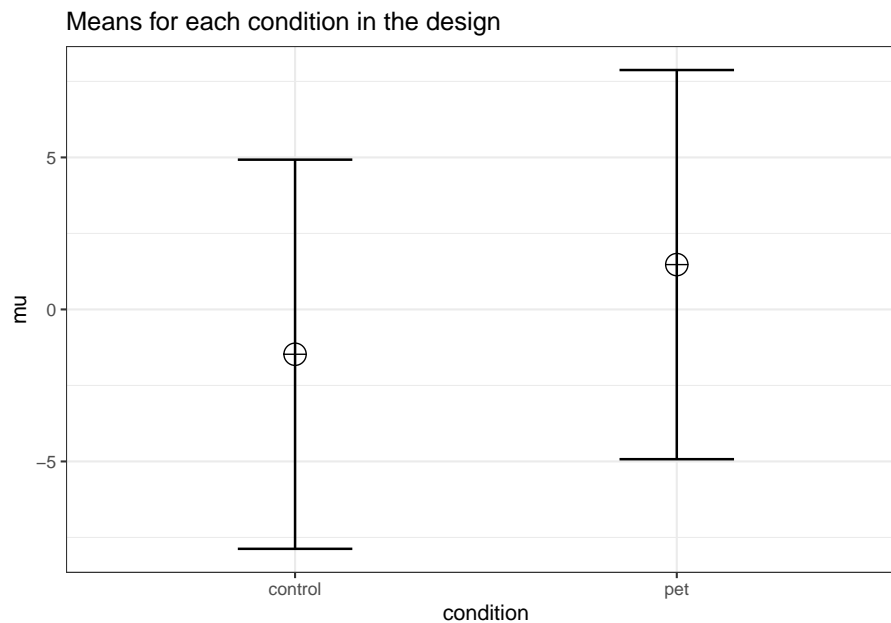
Expressed in eta-squared, we can detect values of eta-squared = 0.05 or larger.


```
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 if 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 3.3: Simulated ANOVA Result

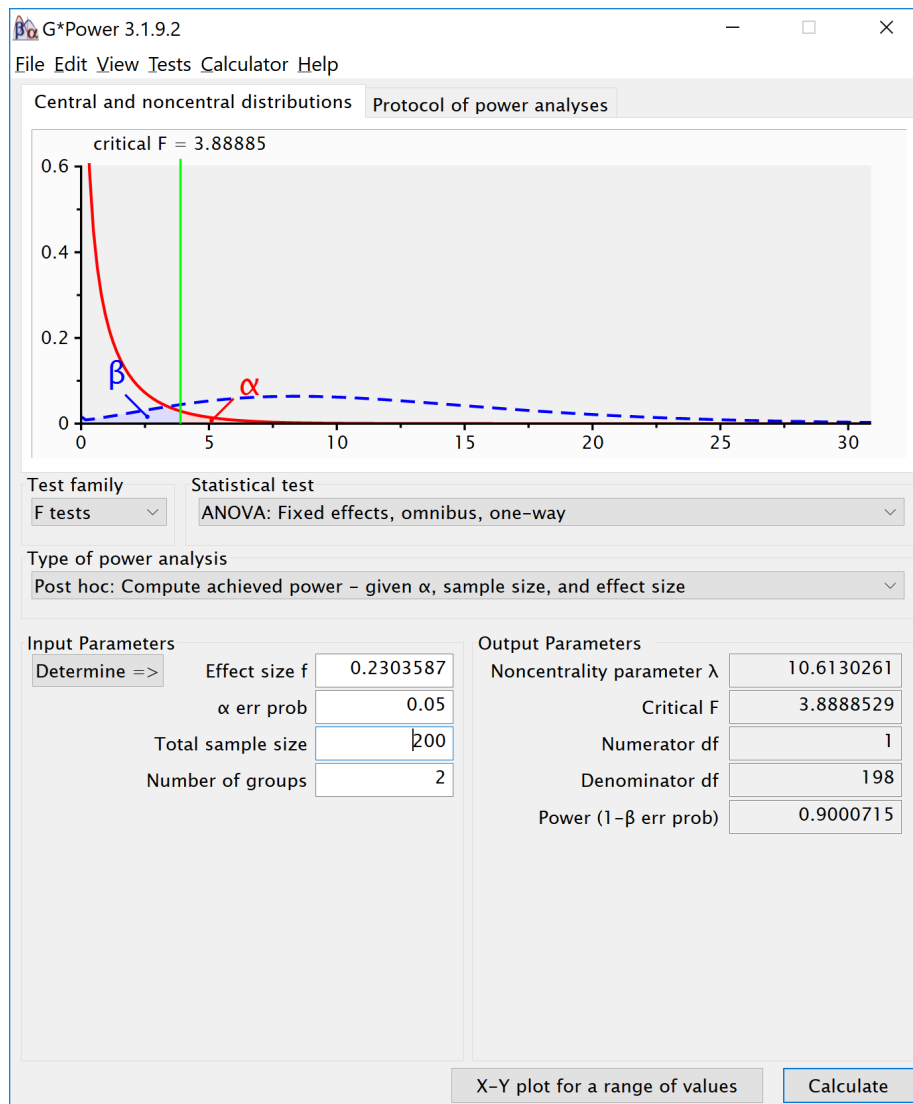
	power	effect_size
anova_condition	89.64	0.0549452

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

Table 3.4: 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 g*power 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.

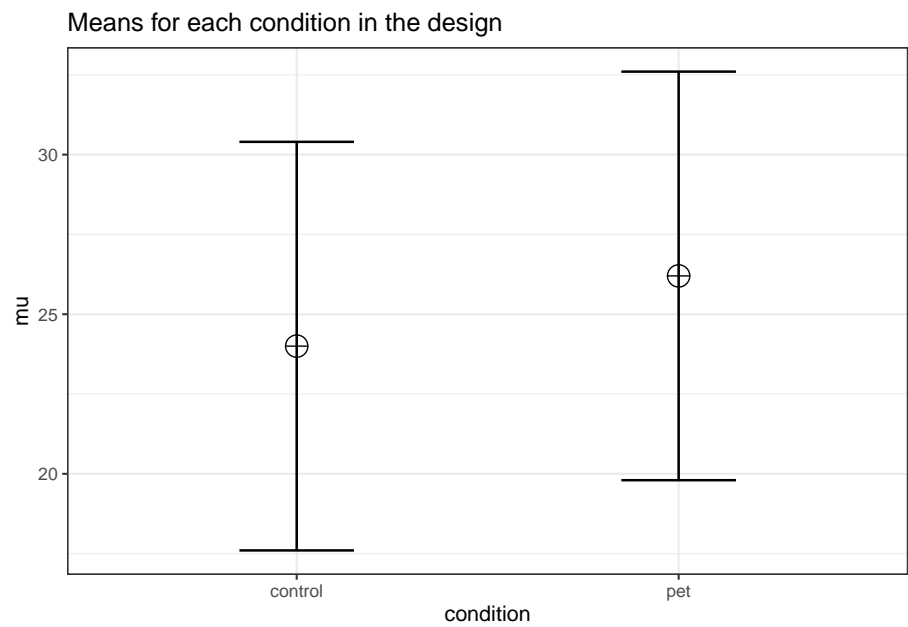
```
K <- 2
power <- 0.9
f <- 0.171875
```

```
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)
```

Table 3.5: Simulated ANOVA Result

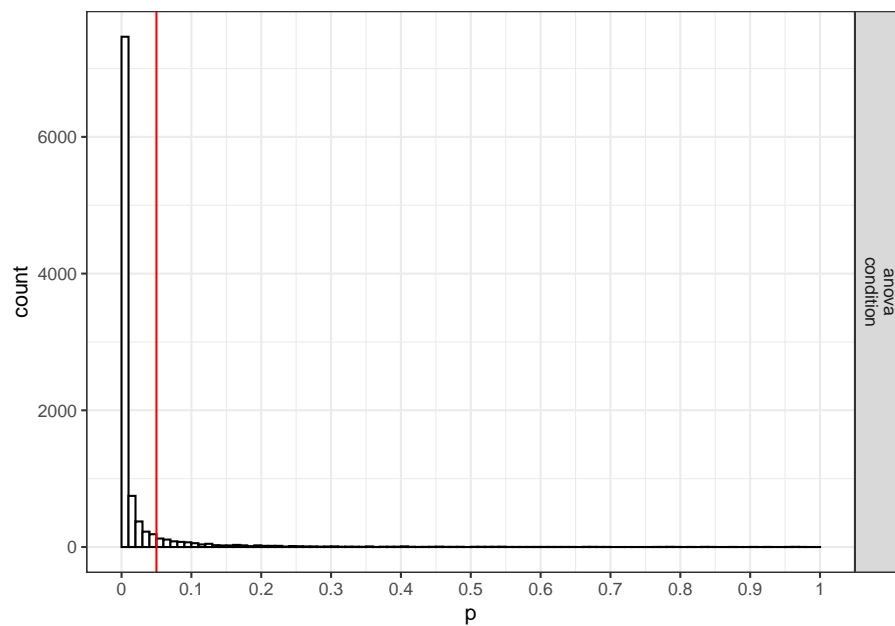
	power	effect_size
anova_condition	89.99	0.0315309

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

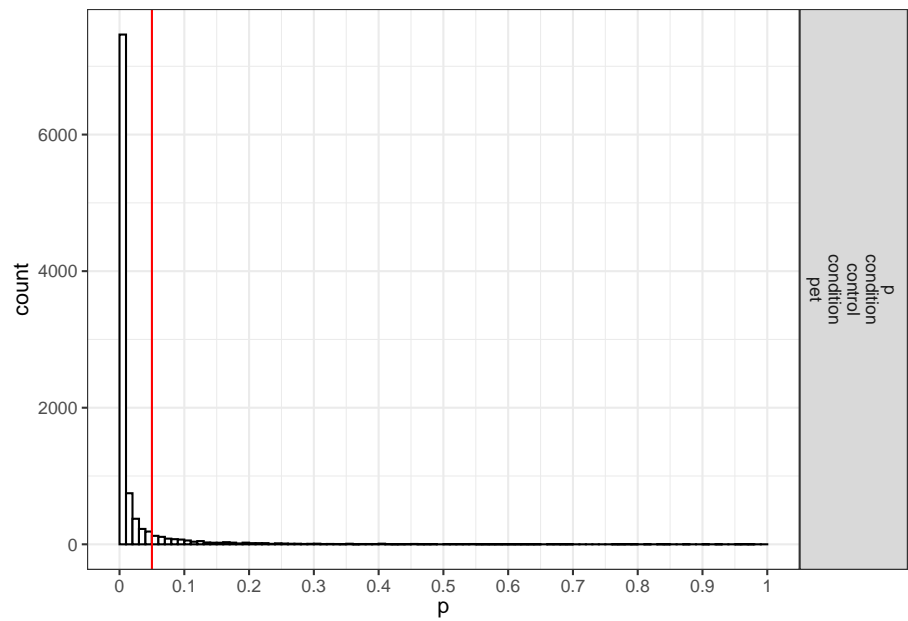
Table 3.6: Exact ANOVA Result

	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):

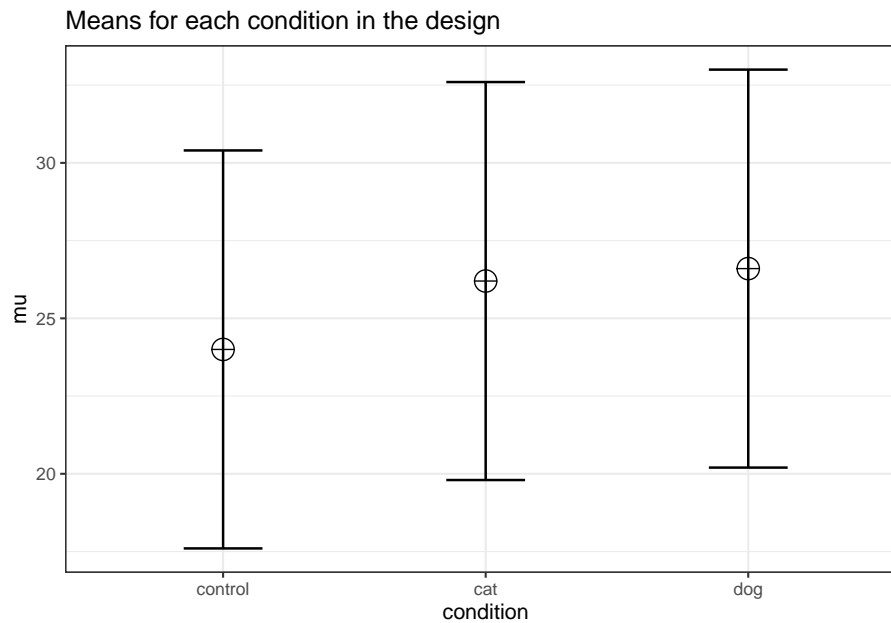


Part 2

3.1.2 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, to 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-satisfaction 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 people 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 3.7: Simulated ANOVA Result

	power	effect_size
anova_condition	48.22	0.0439651

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

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 really low power (around 6%, just above the Type 1 error rate of 5%) for the expected difference between cats and dogs.

Table 3.8: Exact ANOVA Result

	power	partial_eta_squared	cohen_f	non_centrality
condition	47.69468	0.0315259	0.180422	4.785156

3.1.3 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 have 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 participants in each group, in order to achieve 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
```

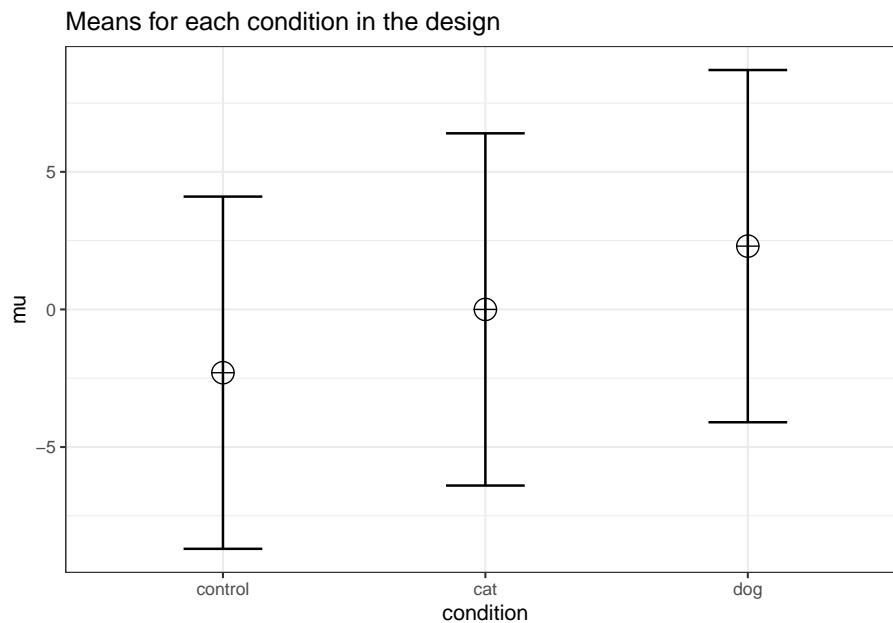
Expressed in eta-squared, we can detect values of eta-squared = 0.0793 or larger.

```
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 if 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)
```

Table 3.9: Simulated ANOVA Result

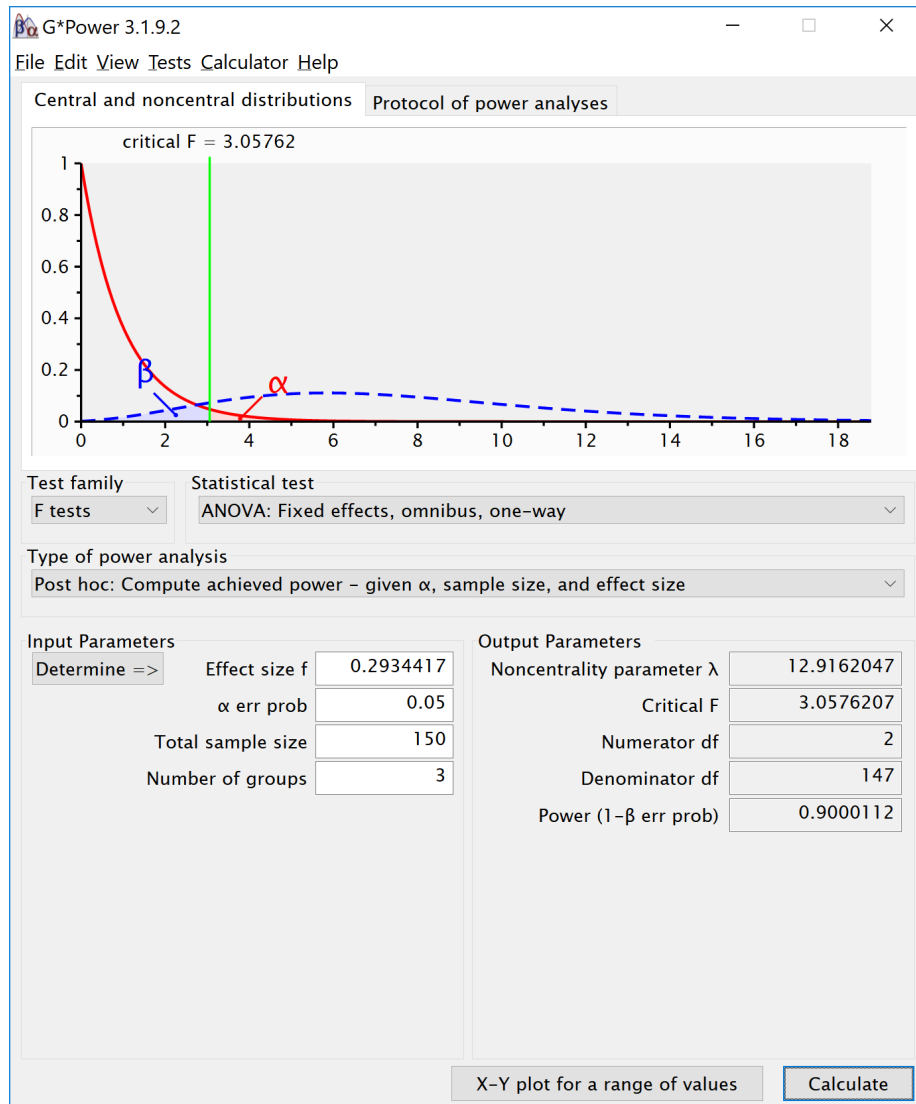
	power	effect_size
anova_condition	90.01	0.0903282

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

Table 3.10: 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
```


3.2 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).

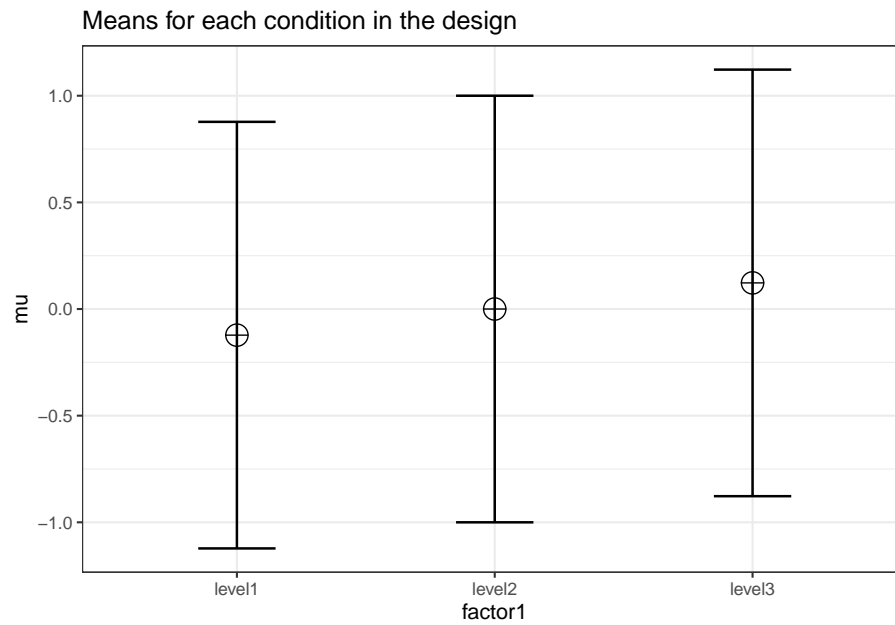
Eta-squared (identical to partial eta-squared for one-way ANOVA's) has benchmarks of .0099, .0588, and .1379 for small, medium, and large effect sizes (Cohen, 1988).

3.2.1 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)
```

Table 3.11: Simulated ANOVA Result

	power	effect_size
anova_factor1	100	0.0100136

```
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
factor1	100	0.009902	0.100005	149.9848

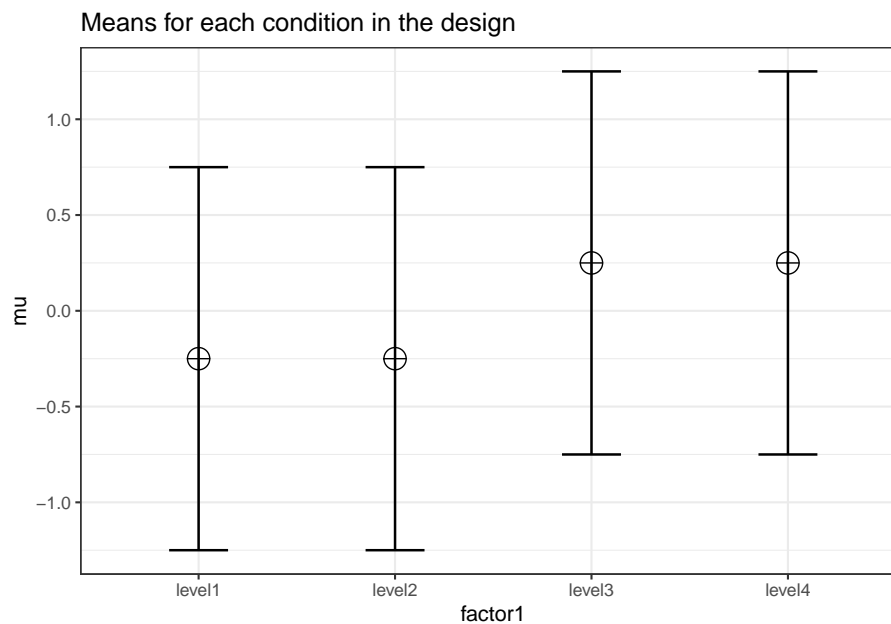
The resulting effect size estimate from the simulation is very close to 0.0099

3.2.2 Four conditions, medium effect size

This simulation should yield a medium effect size (0.0588) 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)
```

Table 3.13: Simulated ANOVA Result

	power	effect_size
anova_factor1	100	0.0589616

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

Table 3.14: Exact ANOVA Result

	power	partial_eta_squared	cohen_f	non centrality
factor1	100	0.0588111	0.2499719	1249.469

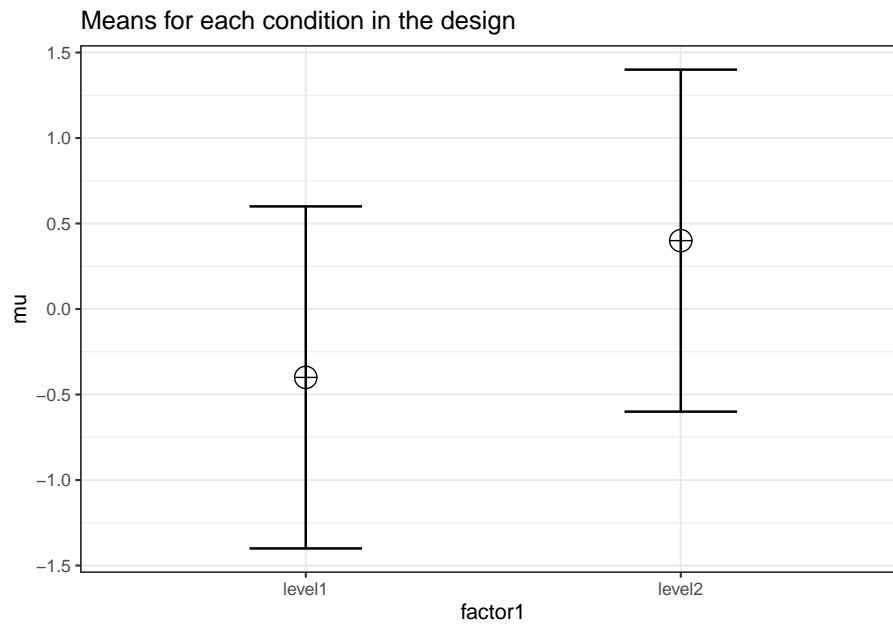
Results are very close to 0.0588.

3.2.3 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 3.15: Simulated ANOVA Result

	power	effect_size
anova_factor1	100	0.1379965

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

Table 3.16: 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 4

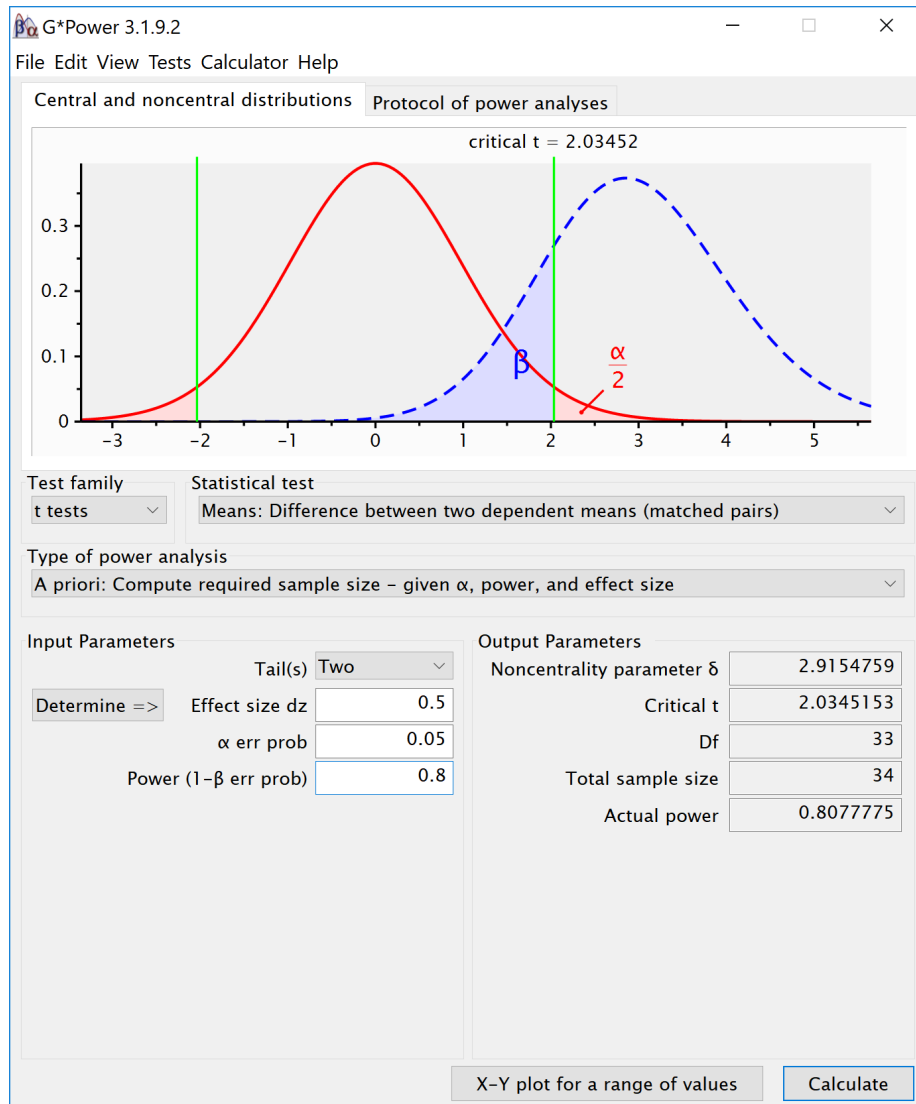
Repeated Measures ANOVA

4.1 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.

4.1.1 Two conditions, medium effect size

To illustrate the effect of correlated 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 g*power Faul et al. (2007). If we want to perform an a-priori power analysis, we are asked to fill in the effect size d_z . 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 d_z . The relation, as Cohen (1988, formula 2.3.7) explains, is:

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

Therefore, the relation between d_z 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, $d = d_z$. 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 d_z 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 $d_z = 2.24$. Some consider this increase in d_z 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.

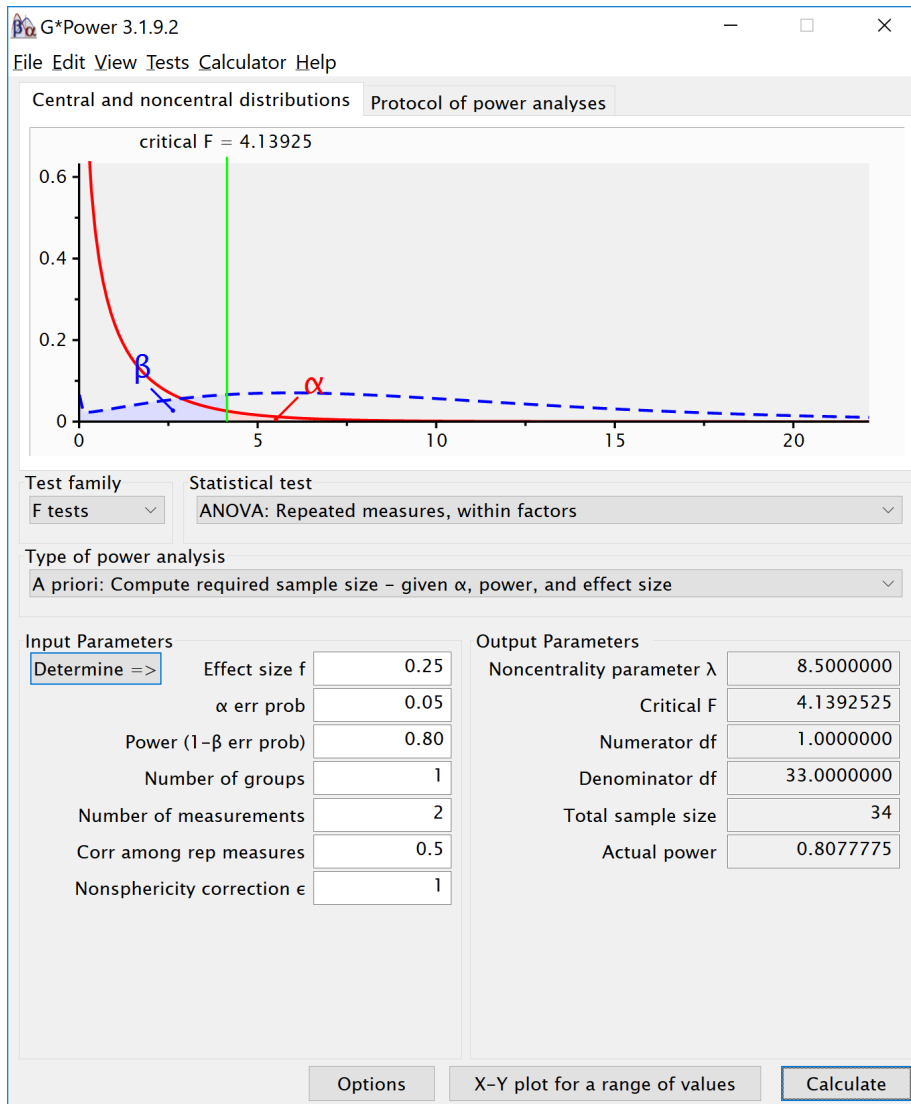
```
# Check sample size formula Maxwell
# Power is pretty similar with n/2, same d (assuming r = 0.5).
# Small differences due to df = 2(n-1) vs df = n-1
pwr.t.test(d = 0.05,
           n = c(2000, 4000, 8000),
           sig.level = 0.05,
           type = "two.sample",
           alternative = "two.sided")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

```
##
##      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
```

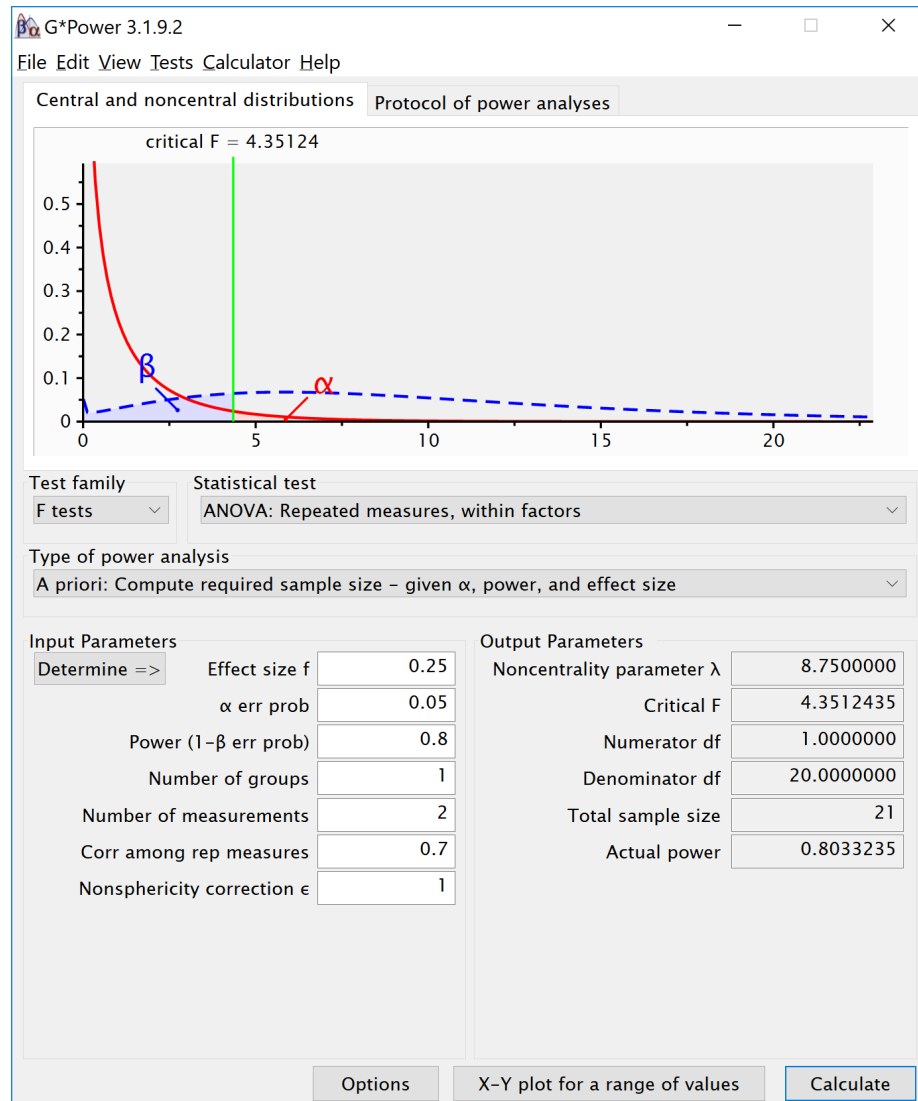
```
pwr.t.test(d = 0.05,
           n = c(1000, 2000, 4000),
           sig.level = 0.05,
           type = "paired",
           alternative = "two.sided")
```

```
##
##      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
```

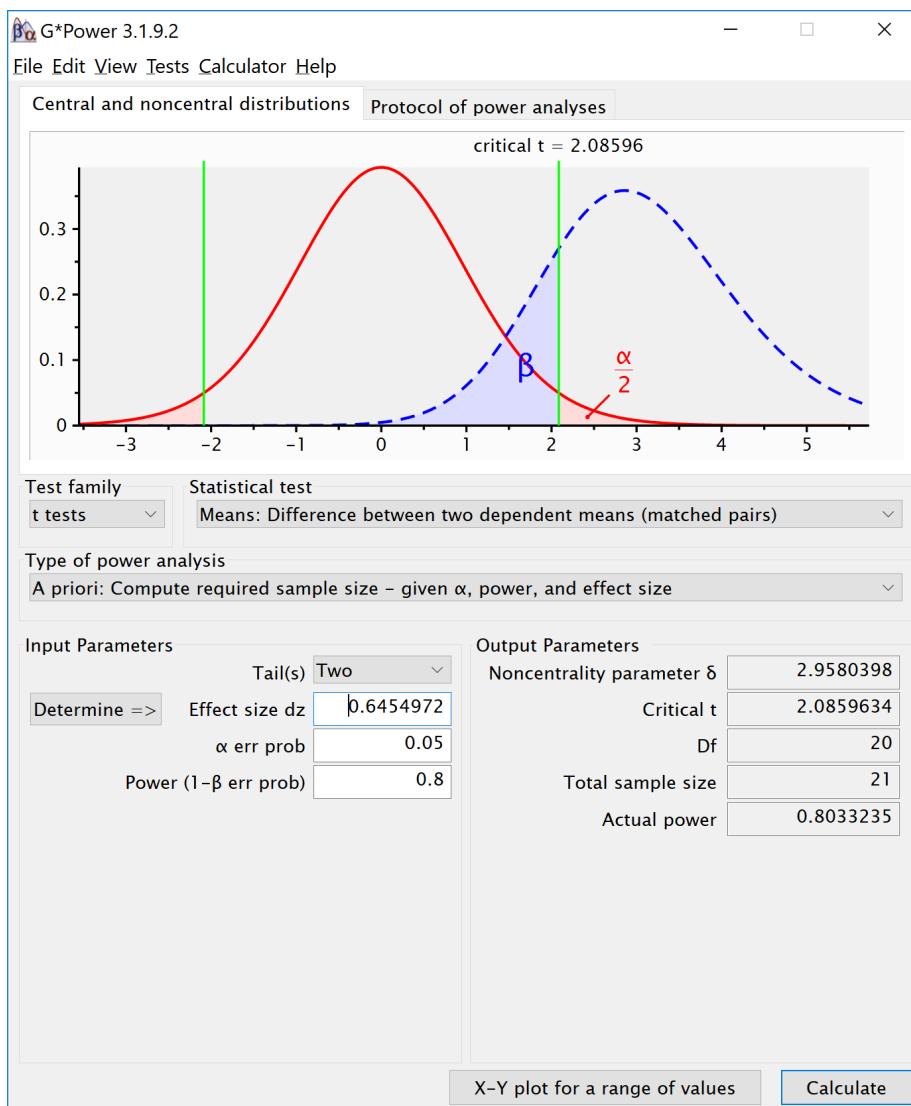
There is no equivalent “fz” for Cohen’s f for a within subject ANOVA. For two groups, we can directly compute Cohen’s f from Cohen’s d for two groups, as Cohen (1988) describes, because $f = 1/2d$. For a $d = 0.5$, $f = 0.25$. In g*power we can run a 2 group within-subject power analysis for ANOVA. We plan for 80% power, and reproduce the analysis 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 g*power with a dependent t -test we need to change d_z following the formula above, $d_z = \frac{0.5}{\sqrt{2(1-0.7)}}$, which yields $d_z = 0.6454972$. If we enter this value in g*power 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 d_z the correlation might be less visible, but given the relation between d and d_z , 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)
```

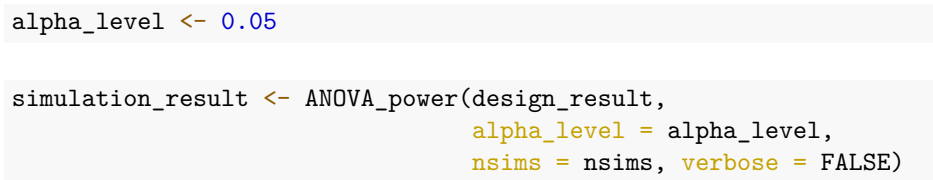


Table 4.1: Simulated ANOVA Result

	power	effect_size
anova_speed	81.02	0.2160964

Table 4.2: Exact ANOVA Result

	power	partial_eta_squared	cohen_f	non_centrality
speed	80.77775	0.2048193	0.5075192	8.5

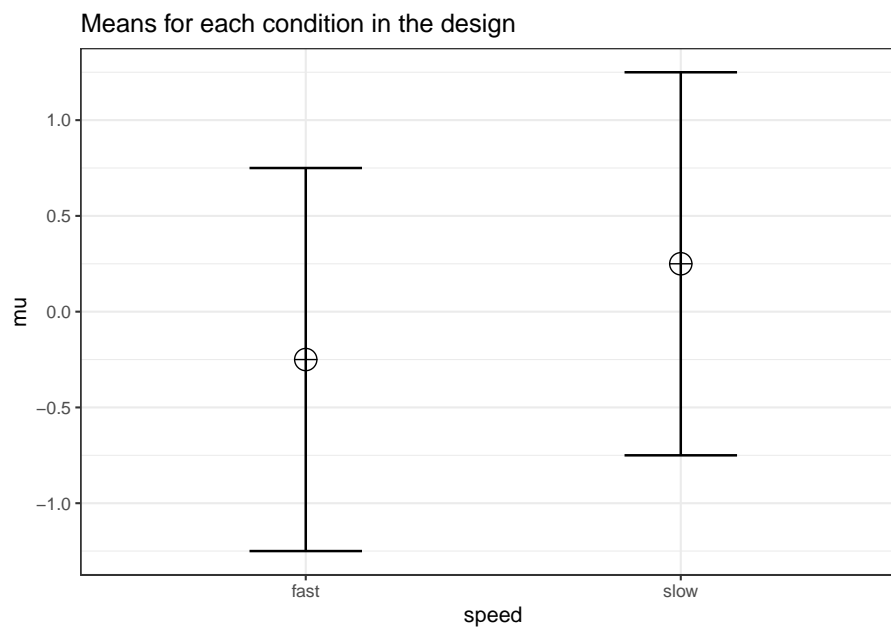
The results of the simulation are indeed very close to 80.777%. Note that the simulation calculates Cohen's d_z effect sizes for paired comparisons - which here given the correlation of 0.5 is also 0.5 for a medium effect size.

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

```
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 4.3: 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 4.4: Simulated ANOVA Result

	power	effect_size
anova_speed	79.99	0.3158324

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

Table 4.5: 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 g*power, with a correlation of 0.5, yields the same as entering f = 0.25 and correlation = 0.7.

4.2 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.

4.2.1 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:

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

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 d_z 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 d_z 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 $\text{sqrt}(2*(1-r))$ we have $d_z = 0.5555556/0.7745967 = 0.7172191$.

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, f_1 , 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 means are 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 yields the maximum standard deviation has the k means falling at both extremes of the range. When k is even, $k/2$ fall at $-d$ and the other $k/2$ fall at $+d$; when k is odd, $(k+1)/2$ 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 sds are known, it is best to calculate Cohen's f directly from these values.

4.2.2 Three within conditions, medium effect size

We can perform power analyses for within designs using simulations. 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
```

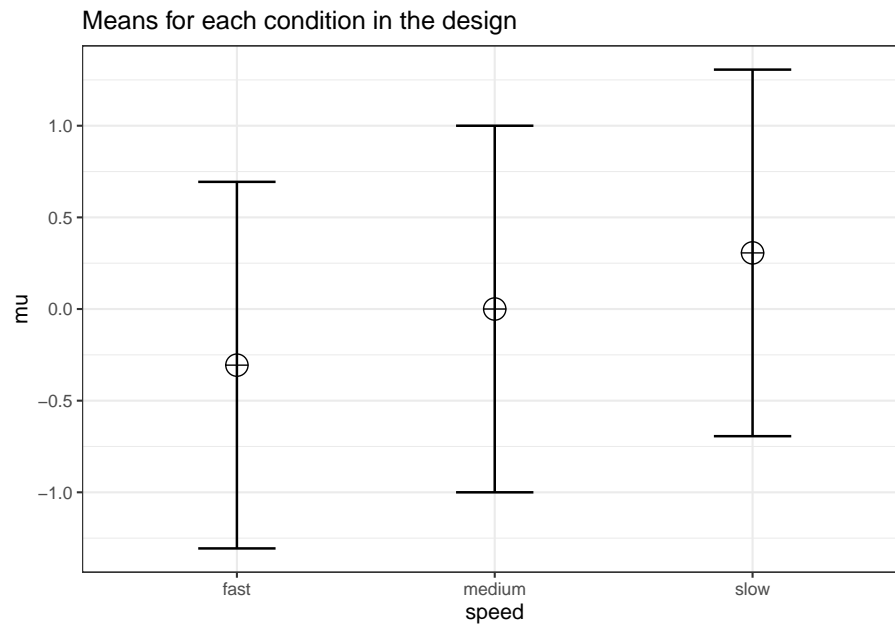
```
## [1] 0.05882353
```

```
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 4.6: Simulated ANOVA Result

	power	effect_size
anova_speed	96.89	0.347031

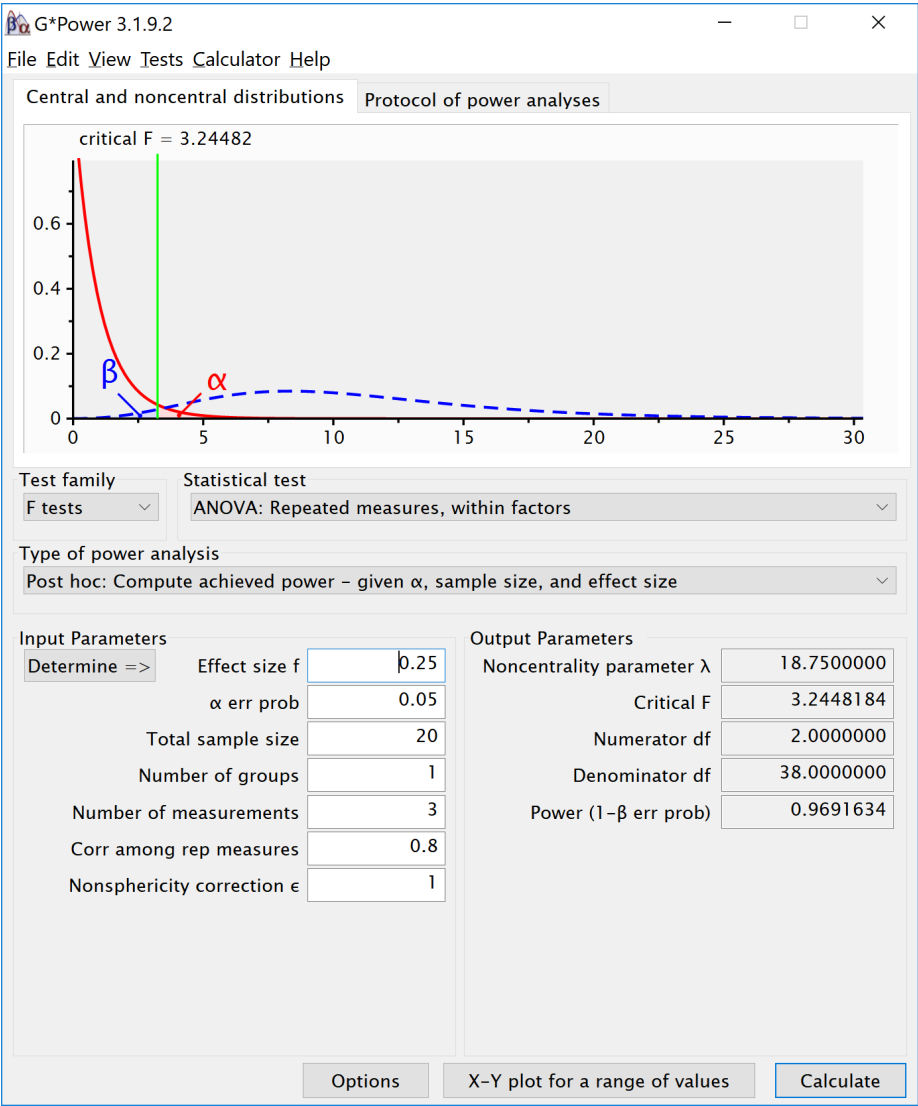
```
exact_result <- ANOVA_exact(design_result,
                             alpha_level = alpha_level,
                             verbose = FALSE)
```

Table 4.7: 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:

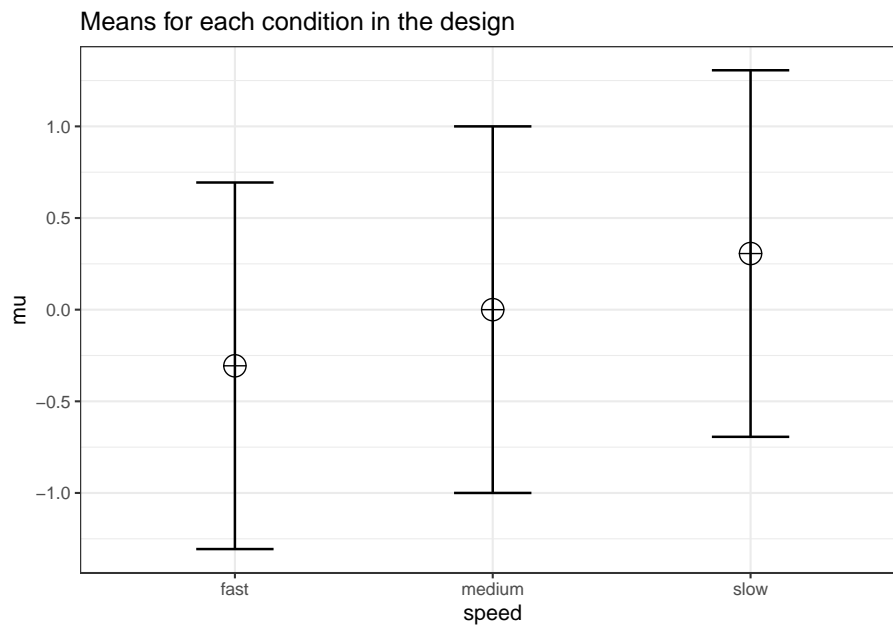


We can also validate this by creating the code to do a power analysis in R from scratch:

```
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
```

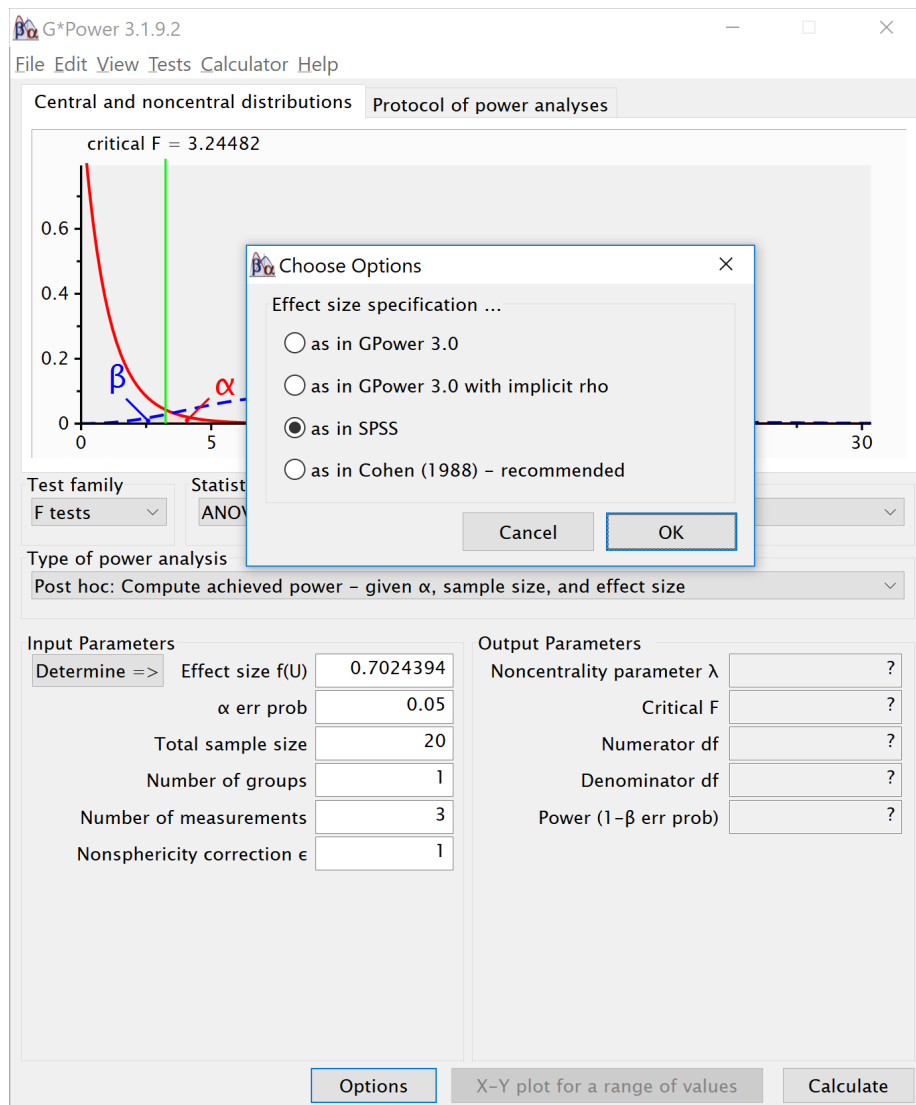
```
power_oneway_within(design_result)$Cohen_f
```

```
## [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 g*power. We take the g*power settings as illustrated above. We click the 'Options' button, and check the radio button 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 separately, 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.



4.3 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] 0.792
```

```
## [1] 0.7576
```

```
## [1] 0.7599
```

```
## [1] 0.048
```

4.4 Part 4

4.4.1 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 approaches 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
##          4.81          4.60          48.34
```

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()

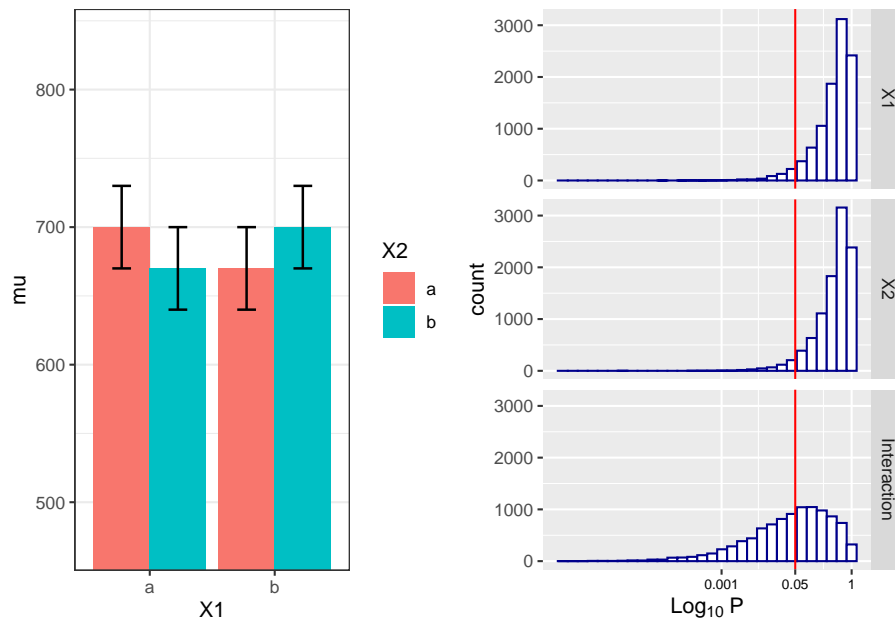
# melt the data into a ggplot friendly 'long' format

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))

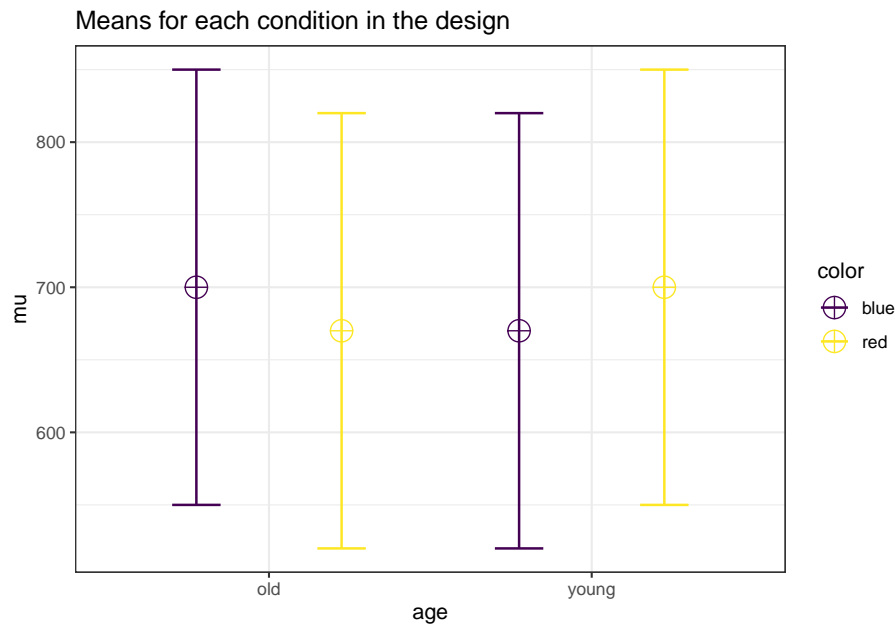
# arrange plots side by side and print
grid.arrange(plt1, plt2, nrow = 1)

```



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 4.8: Simulated ANOVA Result

	power	effect_size
anova_age	4.78	0.0393606
anova_color	4.73	0.0388339
anova_age:color	48.65	0.1665309

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

The simulations yield closely matching results.

4.4.2 Examine variation of means and correlation

Table 4.9: Exact ANOVA Result

	power	partial_eta_squared	cohen_f	non_centralilty
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

```

# 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
##          9.64          30.44          45.95

```

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()

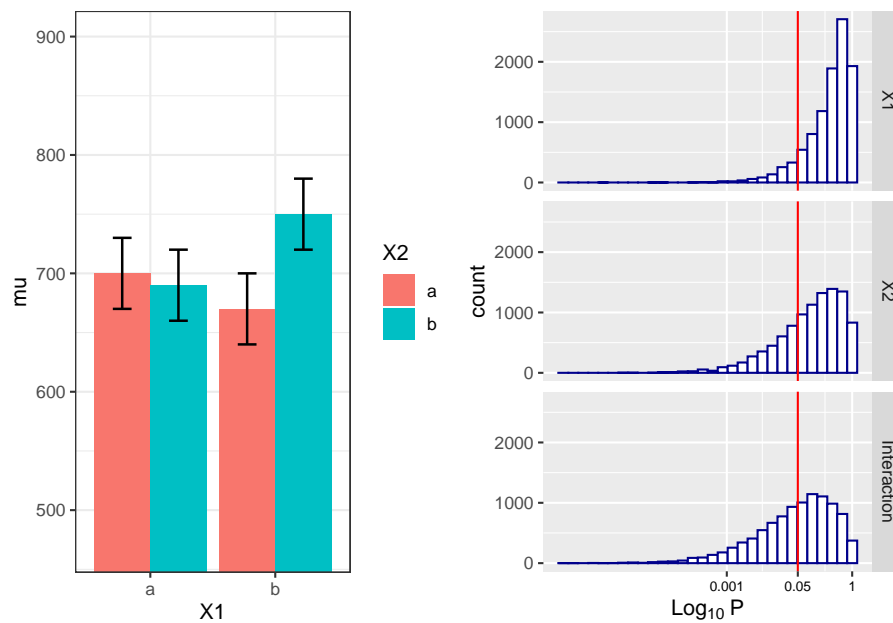
# melt the data into a ggplot friendly 'long' format

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)

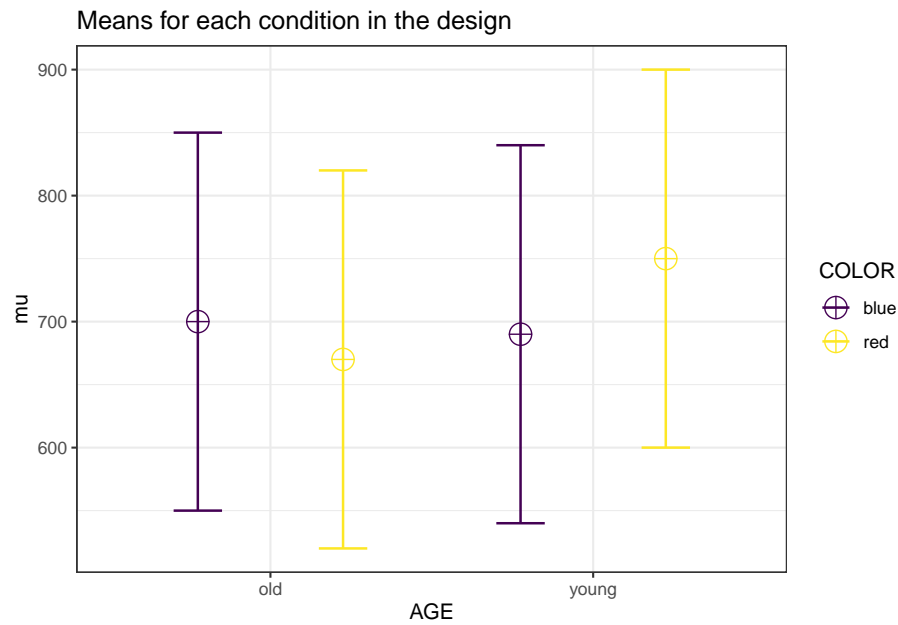
```



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 4.10: Simulated ANOVA Result

	power	effect_size
anova AGE	30.05	0.1143297
anova COLOR	9.45	0.0551819
anova AGE:COLOR	46.26	0.1591529

```
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 4.11: Exact ANOVA Result

	power	partial_eta_squared	cohen_f	non_centralilty
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

4.5 Part 5

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

4.5.1 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:

Example

$\rho_A = 0.4$			$\rho_B = 0.8$			$\rho_{AB} = 0.4$			
A ₁			A ₂			A ₃			
	B ₁	B ₂	B ₃	B ₁	B ₂	B ₃	B ₁	B ₂	B ₃
A ₁	B ₁	1.0	0.8	0.8	0.4	0.4	0.4	0.4	0.4
	B ₂		1.0	0.8	0.4	0.4	0.4	0.4	0.4
	B ₃			1.0	0.4	0.4	0.4	0.4	0.4
A ₂	B ₁			1.0	0.8	0.8	0.4	0.4	0.4
	B ₂				1.0	0.8	0.4	0.4	0.4
	B ₃					1.0	0.4	0.4	0.4
A ₃	B ₁						1.0	0.8	0.8
	B ₂							1.0	0.8
	B ₃								1.0

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:

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

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

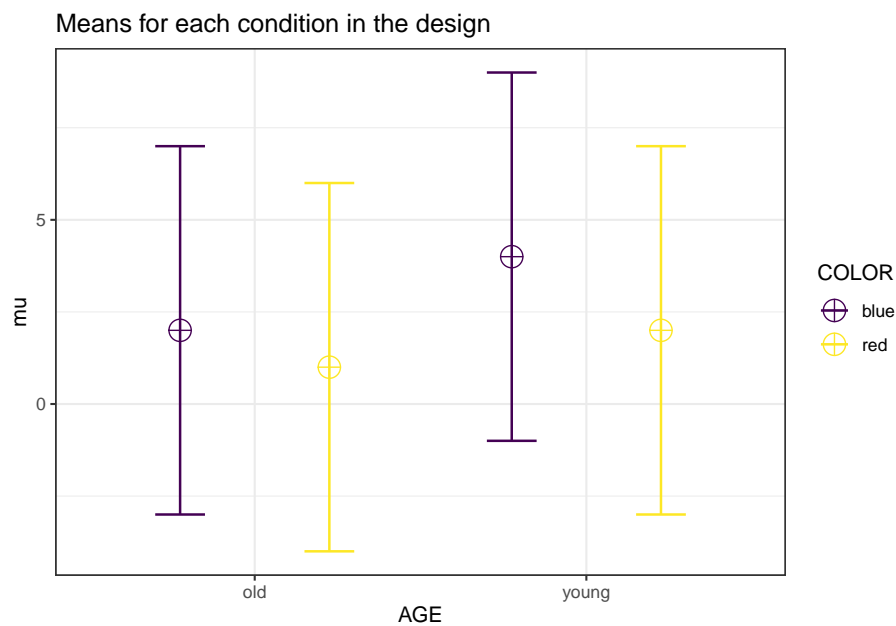
$$\text{For the interaction between A and B: } \sigma_e^2 = \sigma^2(1 - \rho_{\max}) - \sigma^2(\bar{\rho}_{\min} - \bar{\rho}_{AB})$$

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 4.12: 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-value
```

```

        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-value
        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
```

```
#Perform 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-value
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 4.13: Analytical Result

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

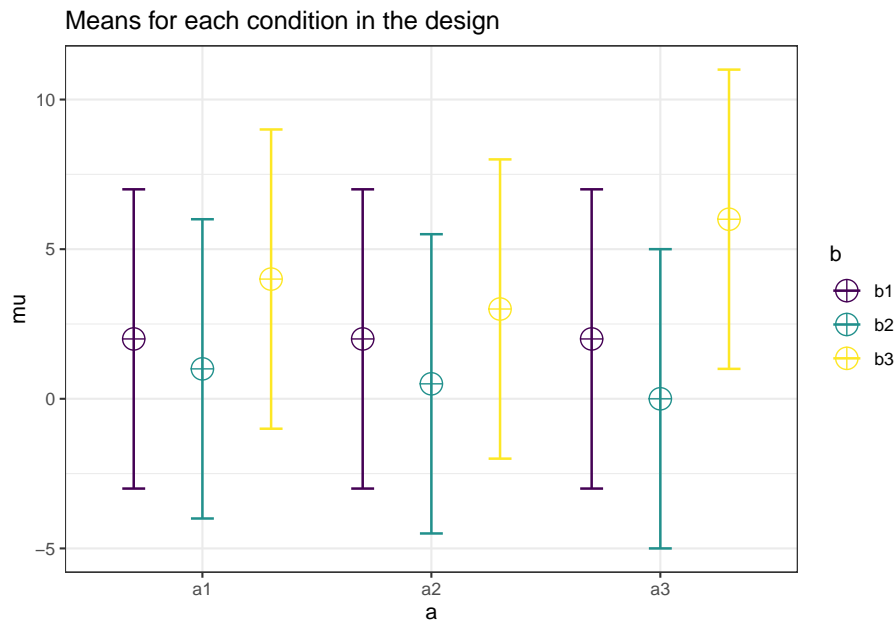
And we can see in the simple scenario matches `ANOVA_exact` and `ANOVA_power` output.

Let's now extend to one of the simulations 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 4.14: 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 0.01562      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-value
                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-value
                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
```

```
#Perform 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-value
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 4.15: Analytical Result

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

4.6 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 sphericity 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 sphericity which makes it a very attractive analytical tool and the preferred method of analysis for some situations (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
```

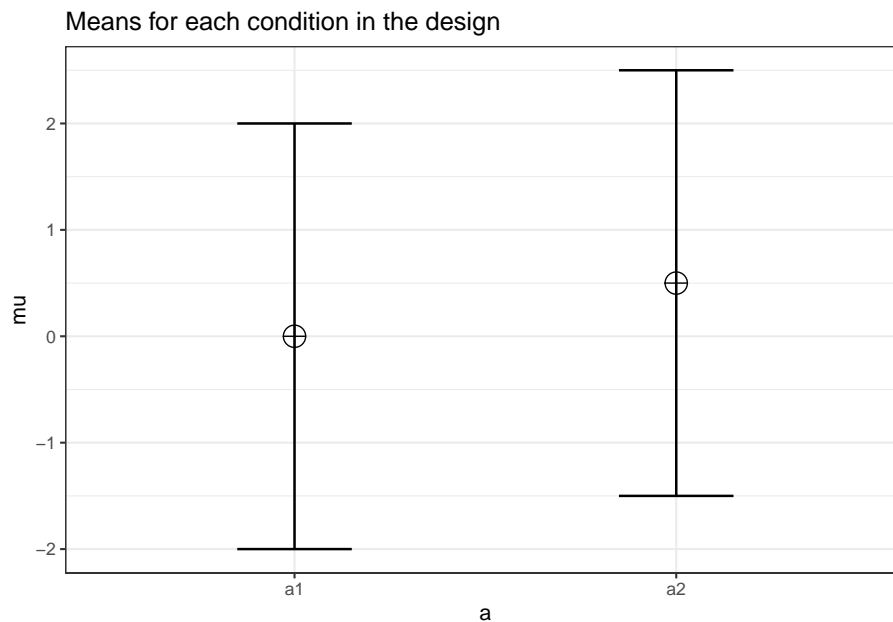
```
## [1] 4.60011
```

```
#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 4.16: 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 in designs with multiple levels and possible violations of the assumption of sphericity.

Chapter 5

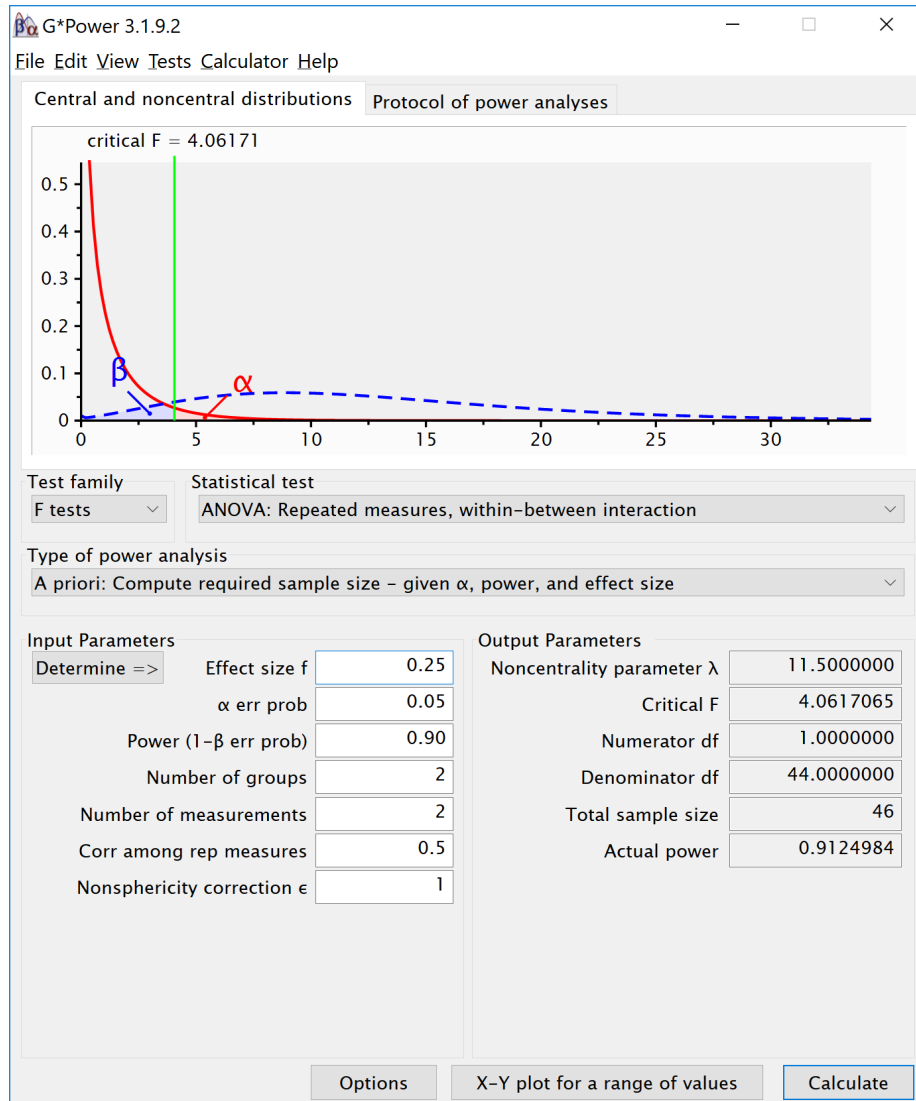
Mixed ANOVA

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.

5.1 Simple Mixed Designs

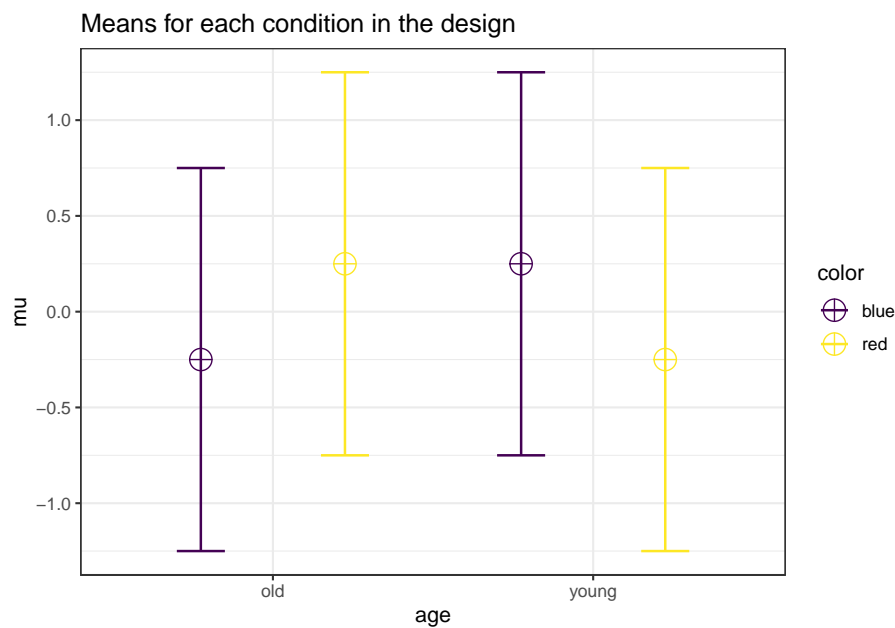
We can simulate a two-way ANOVA with a specific alpha, sample size and effect size, to achieve a specified statistical power. We will try to reproduce the power analysis in *g*power* (Faul et al., 2007) for an F-test from an ANOVA with a repeated measures, within-between interaction effect. While *g*power* 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 5.1: Simulated ANOVA Result

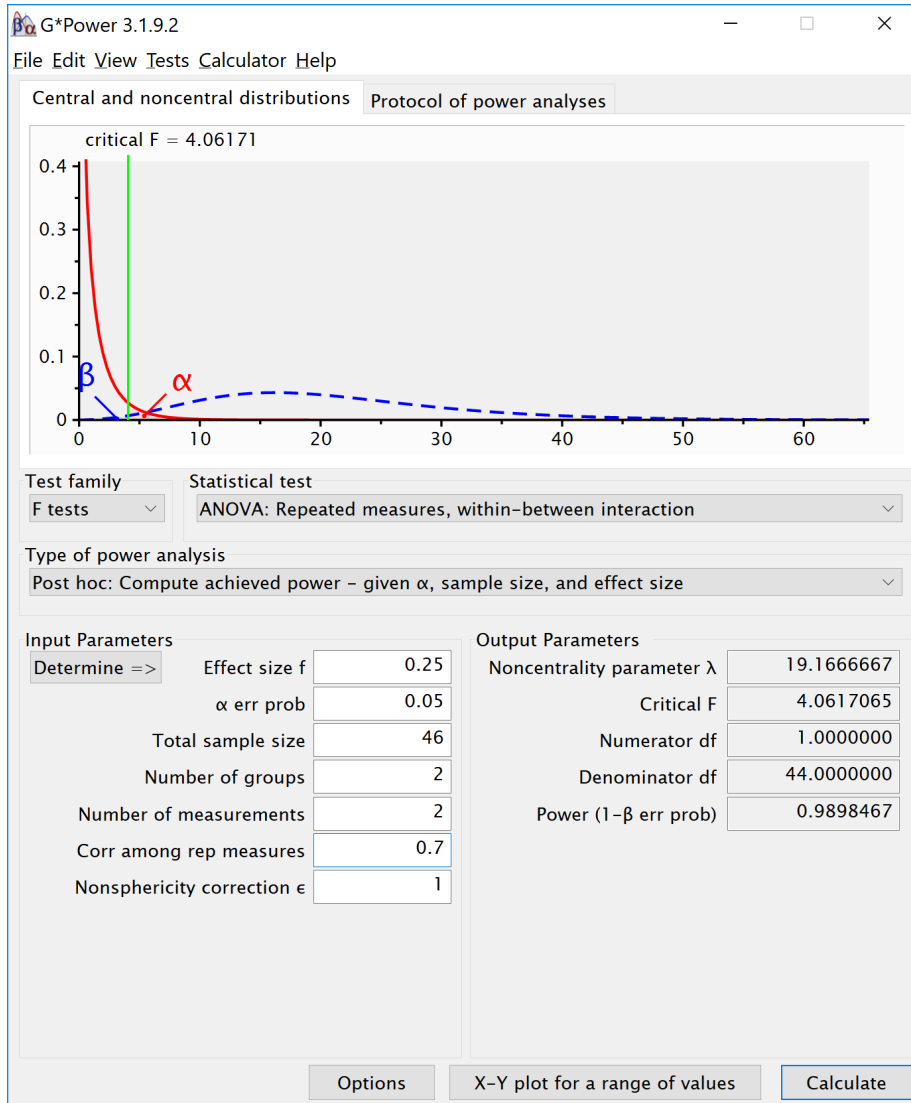
	power	effect_size
anova_color	4.89	0.0219823
anova_age	5.20	0.0225908
anova_color:age	91.22	0.2151983

```
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
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

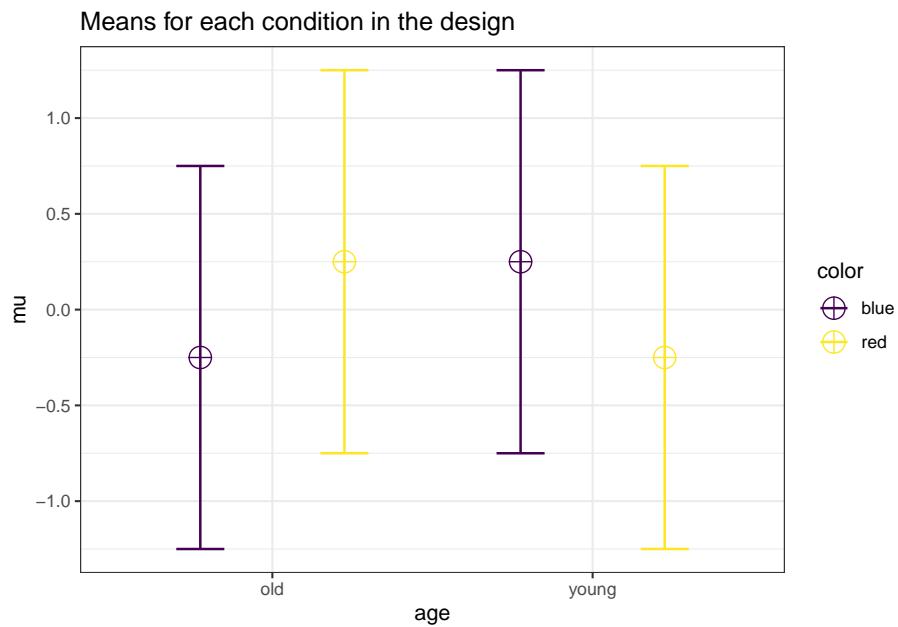
Now, we can simulate the same two-way ANOVA but increasing the correlation to $r=0.7$.



```

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)

```

```

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

```

Table 5.3: Simulated ANOVA Result

	power	effect_size
anova_color	5.14	0.0220381
anova_age	4.78	0.0219031
anova_color:age	99.03	0.3066459

Table 5.4: 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

5.2 Complex Mixed Designs

Now, we are to the most complicated calculations. Similiar to the simple one-way repeated measures ANOVA, a mixed ANOVA assumes sphericity. 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.

5.2.1 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 with 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.

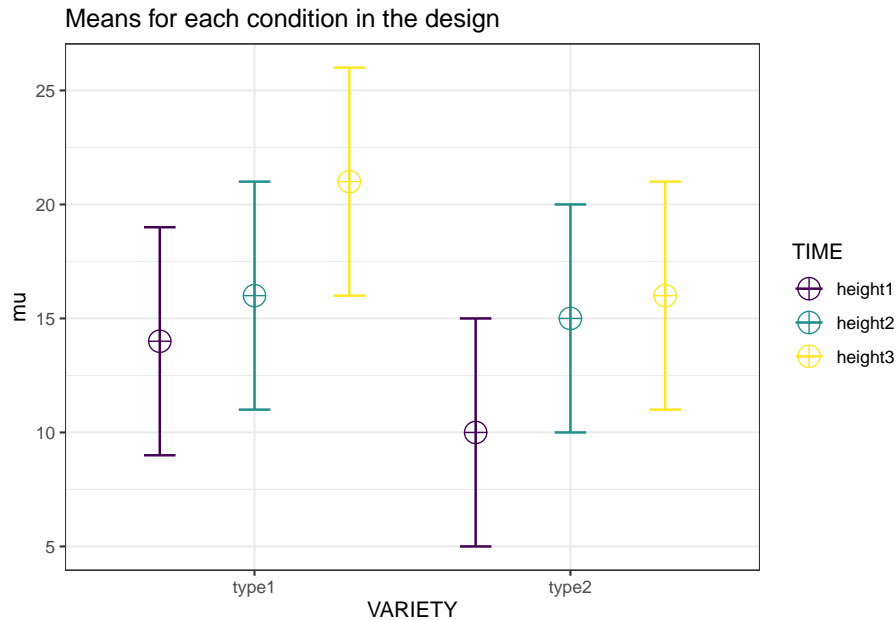
```
proc glmpower data=Exemplary2;  
class Variety Exposure;  
model Height1 Height2 Height3 = Variety;  
repeated Time contrast;  
power  
mtest = pt  
stddev = 5  
ntotal = 40  
power = .  
MATRIX("MyCorrs")= (.75,  
                     0.5625,.75)  
CORRS= "MyCorrs";  
run;
```

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


```

                                "height3"),
                                plot = TRUE)

```



```

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

```

Table 5.5: 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.

5.2.2 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 $2b \times 4w$ 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 changing 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

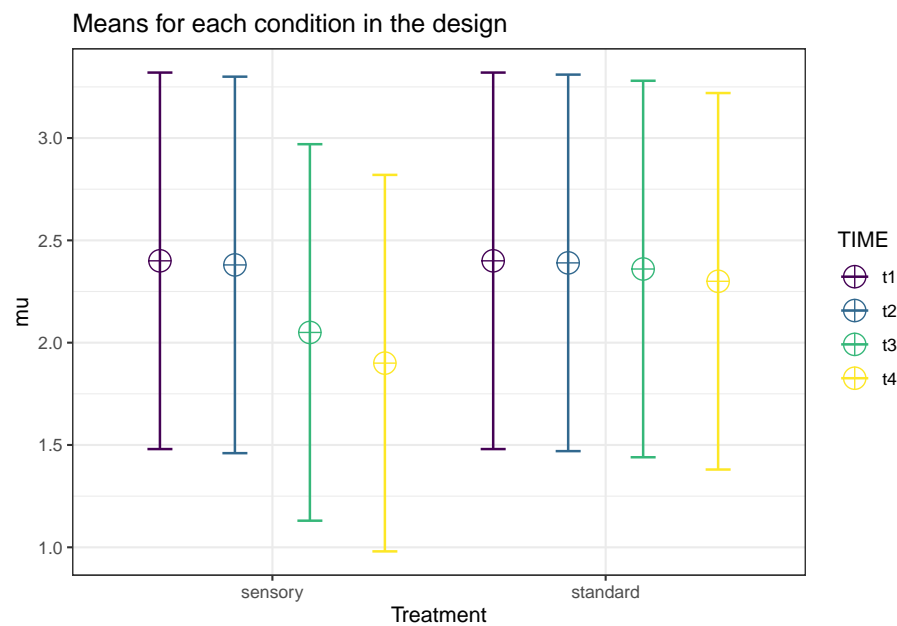
We then replicate in R, first by setting up the “lear” correlation matrix.

```
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)
```

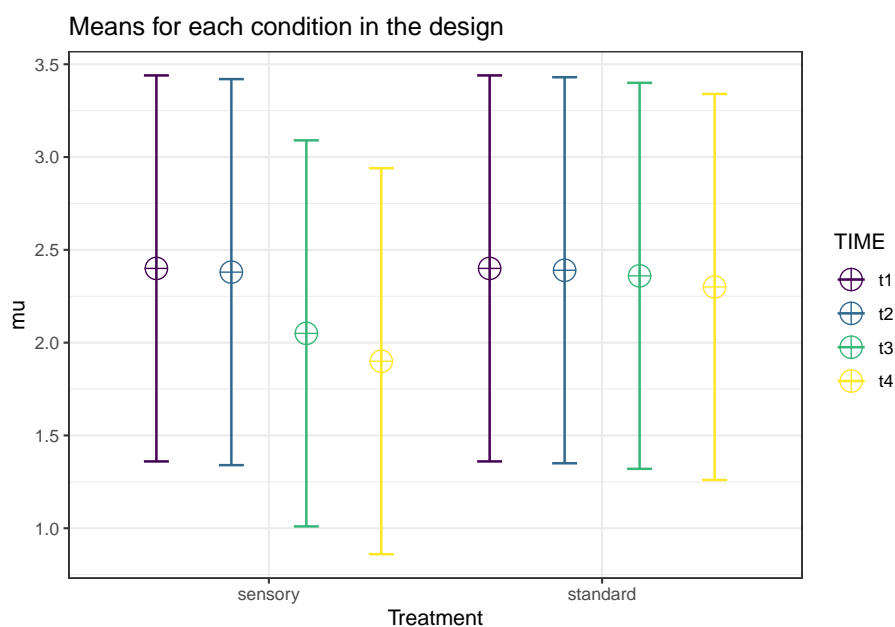


```
exact_result <- ANOVA_exact(design_result, verbose = FALSE,  
  alpha_level = .01)
```

Table 5.6: 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)
```

Table 5.7: 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

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

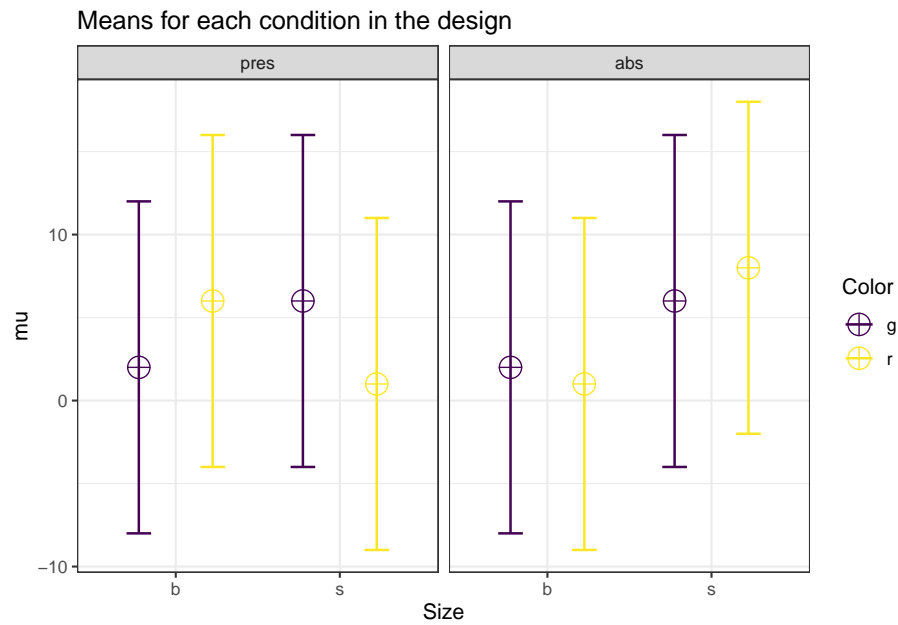
Chapter 6

Power for Three-way Interactions

There are almost no software solutions that allow researchers to perform power analysis 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 6.1: Simulated ANOVA Result

	power	effect_size
anova_Size	70.17	0.0179159
anova_Color	4.86	0.0025370
anova_Load	8.31	0.0032647
anova_Size:Color	31.89	0.0081783
anova_Size:Load	85.45	0.0248185
anova_Color:Load	7.80	0.0031794
anova_Size:Color:Load	85.60	0.0248853

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


Table 6.2: 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 *g*power* (Faul et al., 2007). *g*power* 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
```

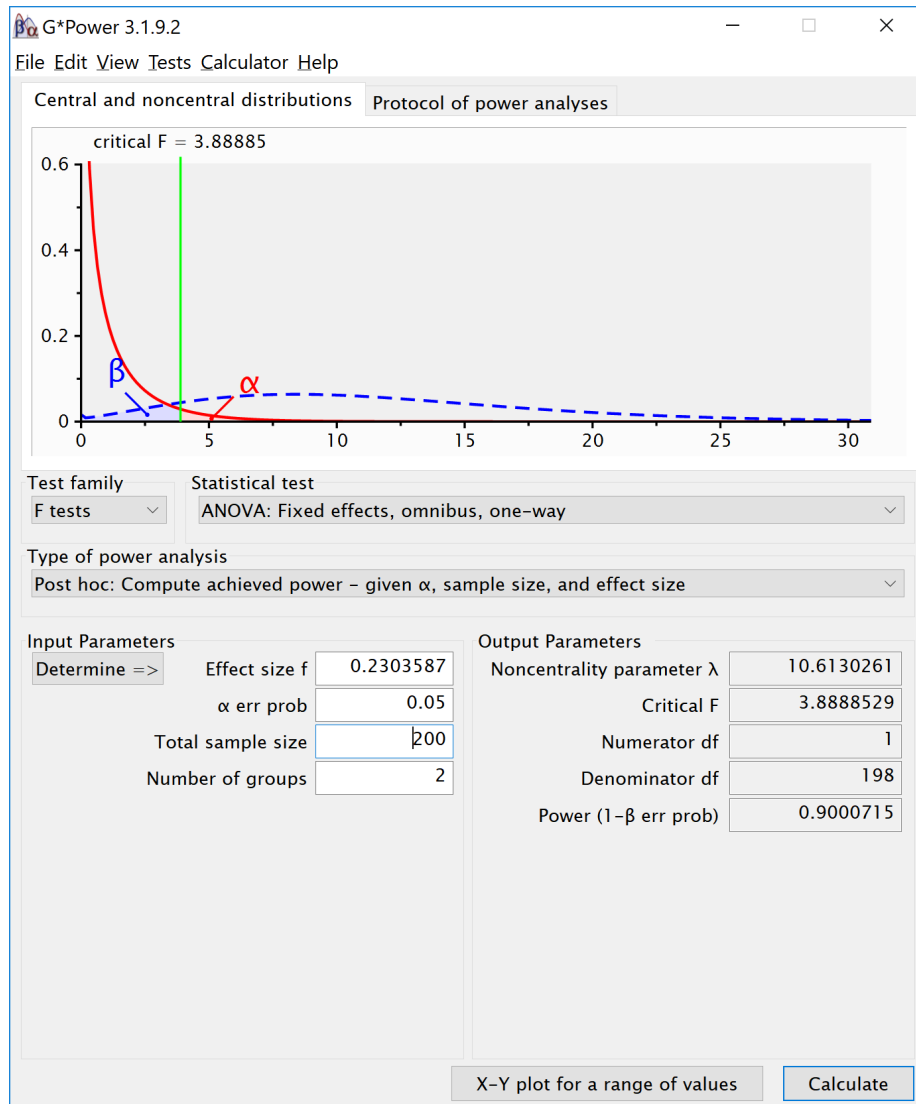
```
## [1] 0.15
```

```
# 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
```

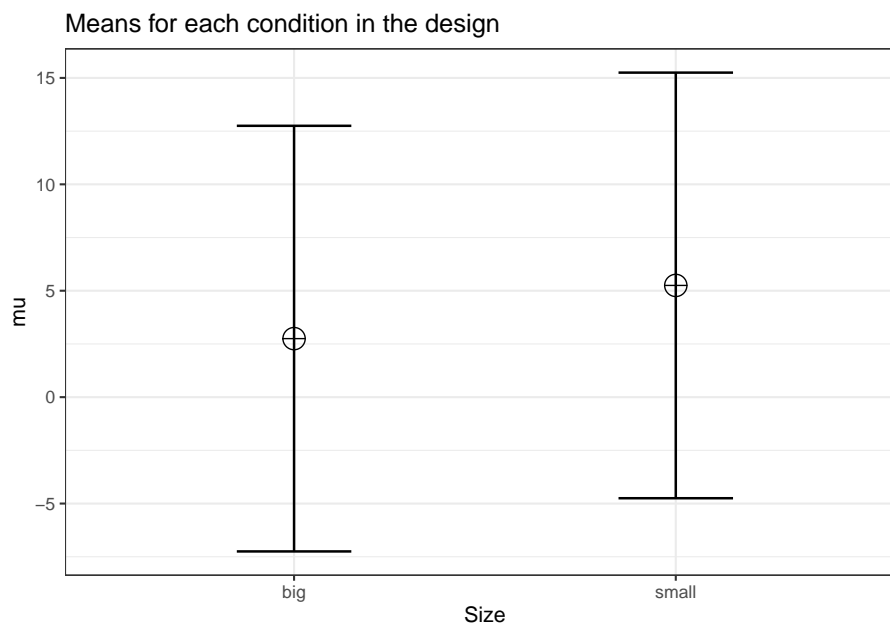
```
## [1] 3.864929
```



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 6.3: Simulated ANOVA Result

	power	effect_size
anova_Size	69.63	0.0177096

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

Table 6.4: Exact ANOVA Result

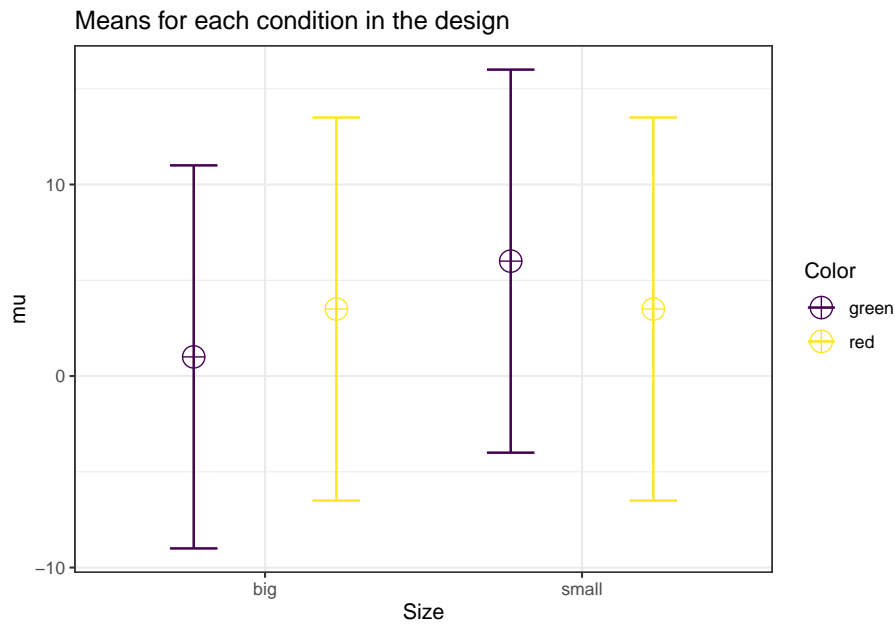
	power	partial_eta_squared	cohen_f	non_centrality
Size	70.33333	0.0154607	0.1253137	6.25

```
# 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 6.5: Simulated ANOVA Result

	power	effect_size
anova_Size	70.41	0.0179067
anova_Color	5.10	0.0025282
anova_Size:Color	70.80	0.0179360

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

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

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

Table 6.6: 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

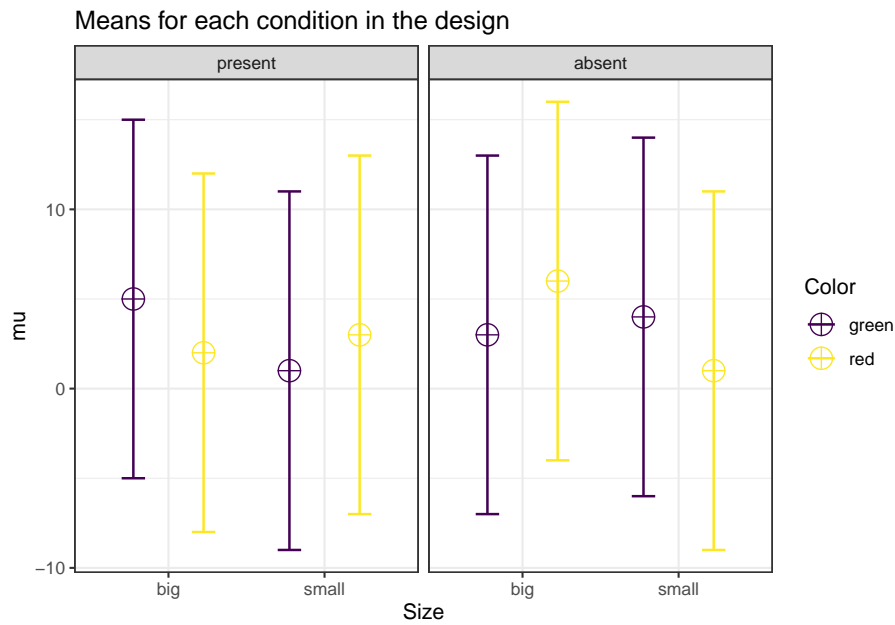
```
power_res$power_B
```

```
## [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 6.7: Simulated ANOVA Result

	power	effect_size
anova_Size	41.15	0.0101529
anova_Color	5.97	0.0027337
anova_CognitiveLoad	11.55	0.0039201
anova_Size:Color	5.67	0.0027030
anova_Size:CognitiveLoad	5.45	0.0026596
anova_Color:CognitiveLoad	5.54	0.0027225
anova_Size:Color:CognitiveLoad	78.61	0.0214761

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

Table 6.8: 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
```

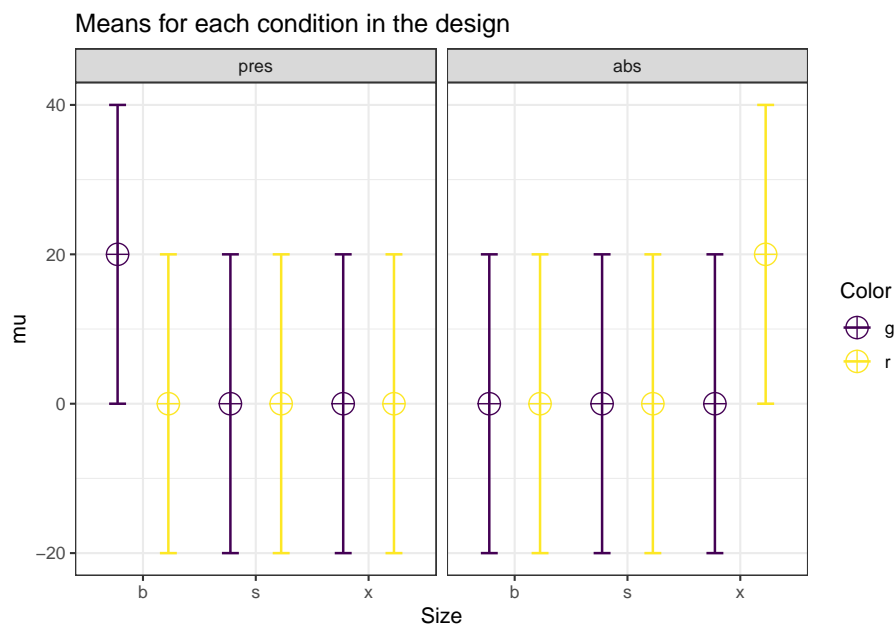
```
## [1] 0.01855544
```

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, 0, 20),
                             sd = 20,
                             labelnames = labelnames)

```



```

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

```

```

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

```

Table 6.9: 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

```
## [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
```

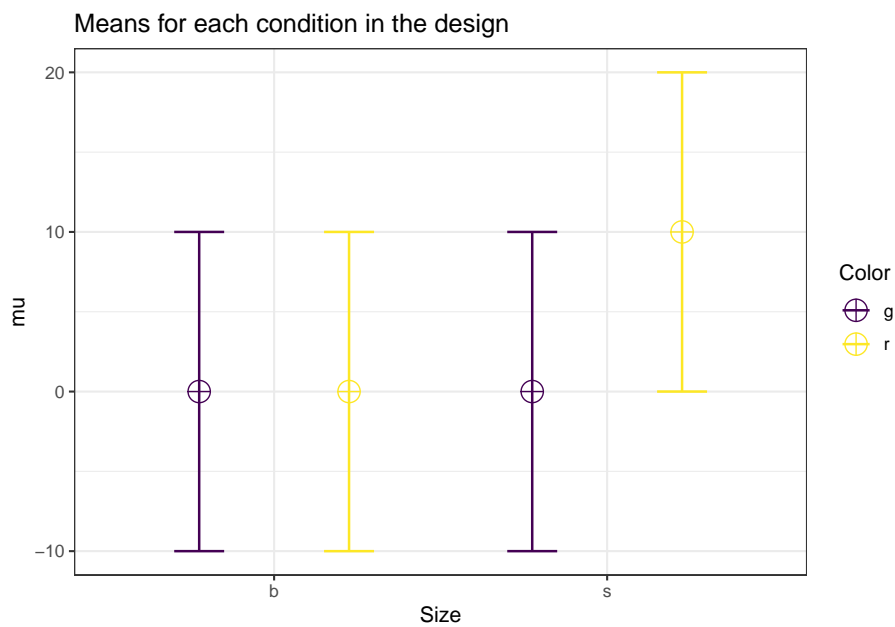
```
## [1] 0
```

```
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 6.10: 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
```

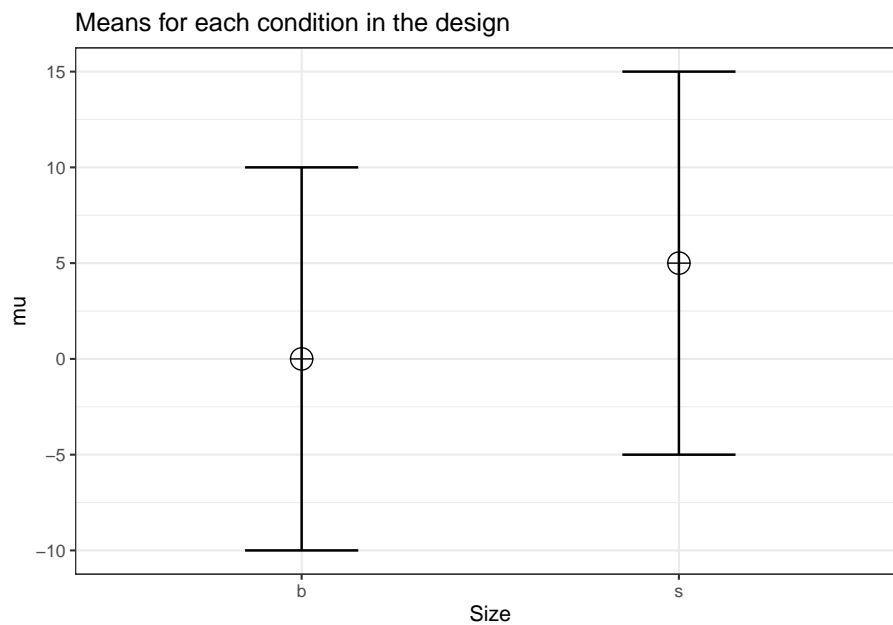
```
## [1] 0.05882353
```

```
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 6.11: 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
```

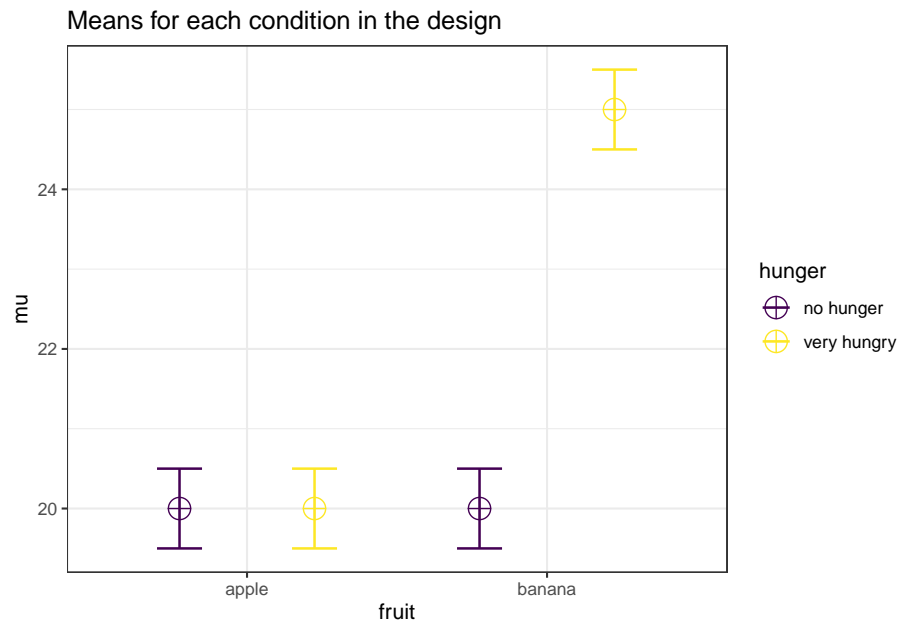
```
## [1] 0.25
```


Chapter 7

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
```

```

#d1: simple effect M1-M2
#d2: moderated effect M3-M4,
# full elimination of effect implies d2=0
#simtoto: 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:simtoto) {
  #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) / simtoto
#Interaction
power2 = sum(p2 <= .05) / simtoto

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: 0.88
```

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

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

```
##  
## Study 1 is powered to: 0.3
```

```
##  
## Study 2 is powered to: 0.31
```

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

```
##  
## Study 1 is powered to: 0.57
```

```
##  
## Study 2 is powered to: 0.57
```

#Need 4n if effect is 70% attenuated

```
colada17(simtot = 2000, n1 = 25, n2 = 100, d1 = .5, d2 = .3 * .5)
```

```
##  
## Study 1 is powered to: 0.41
```

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

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

```
##  
## Study 1 is powered to: 0.69
```

```
##  
## Study 2 is powered to: 0.7
```

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

```
##
## Study 1 is powered to: 0.26
```

```
##
## Study 2 is powered to: 0.27
```

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

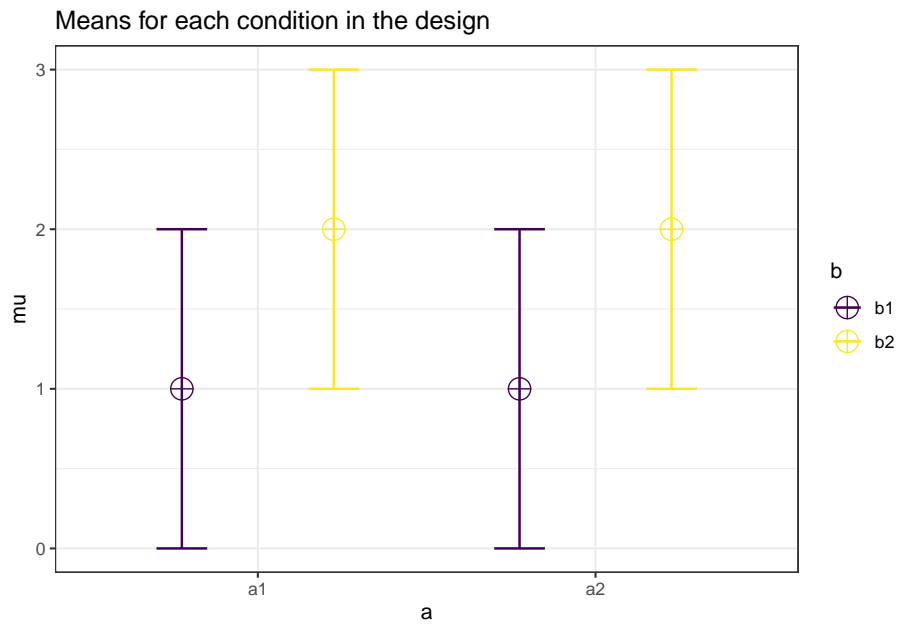
```
##
## Study 1 is powered to: 0.87
```

```
##
## Study 2 is powered to: 0.6
```

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)

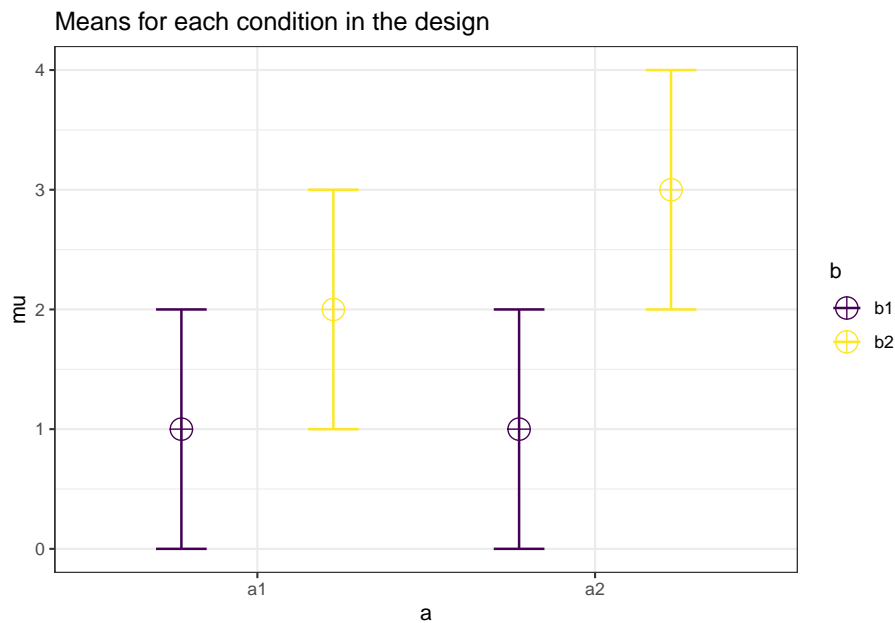
knitr::kable(exact_result$main_results[2,],
              caption = "ANOVA Results")%>%
  kable_styling(latex_options = "hold_position")
```

Table 7.1: ANOVA Results

	power	partial_eta_squared	cohen_f	non_centralilty
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 7.2: ANOVA Results

	power	partial_eta_squared	cohen_f	non_centralilty
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: 0.86
```

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

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

```
##
## Study 1 is powered to: 0.31
```

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

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

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

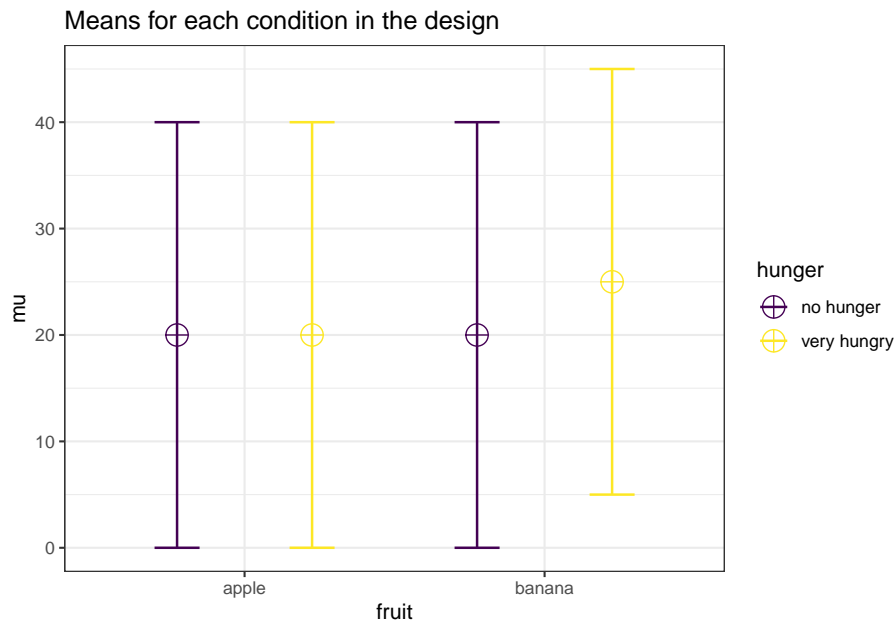
```
##
## Study 2 is powered to: 0.58
```

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 7.3: ANOVA Results

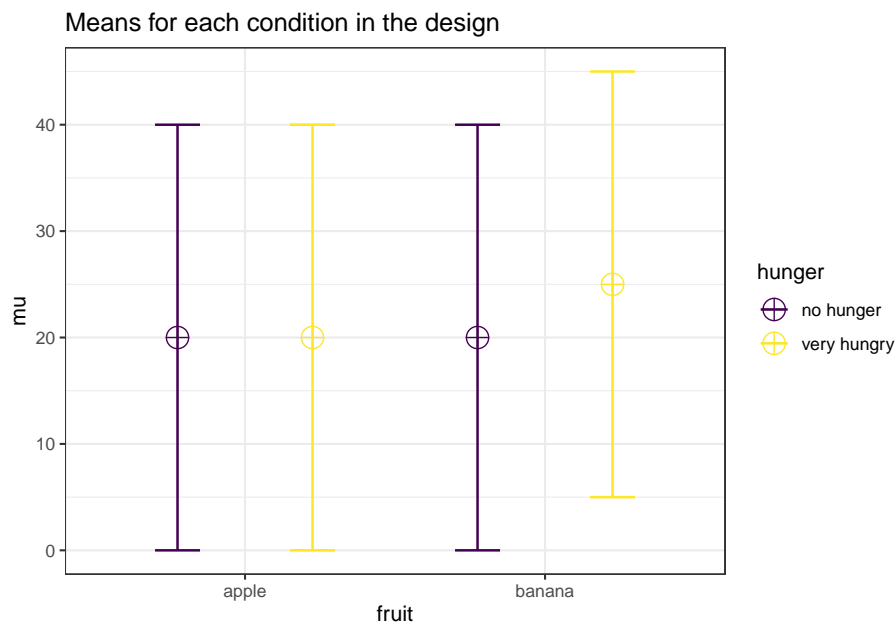
	power	partial_eta_squared	cohen_f	non centrality
hunger	33.32955	0.0039171	0.0627094	2.34375

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)
##
## p_fruit_apple_hunger_no hunger_fruit_apple_hunger_very hungry      power
## 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 7.4: 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, 20 (apple) vs 22.5 (banana).

```
rowMeans(mean_mat)
```

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

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

```
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:

$$\underbrace{22.5}_{\text{cell value}} - \left(\underbrace{21.25}_{\text{grand mean}} + \underbrace{(22.5 - 21.25)}_{\text{row2 mean - grand mean}} + \underbrace{(22.5 - 21.25)}_{\text{col2 mean - grand mean}} \right) = 1.25$$

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

```

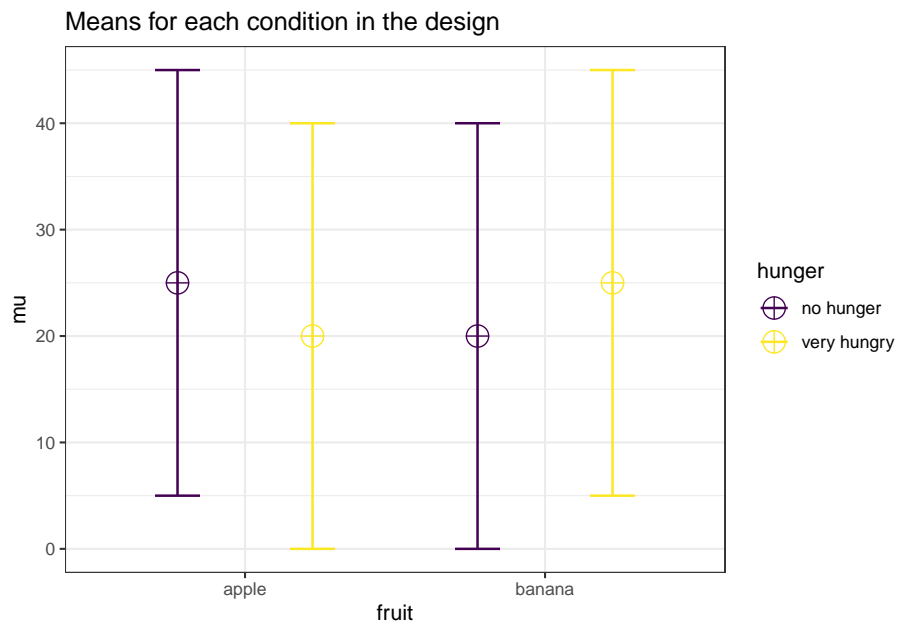
Neither of the main effects is now significant, as the marginal means are 22.5 vs 22.5 for both main effects. The interaction is much stronger, however. We are testing whether the average effect of hunger on the desirability of fruit is different in the presence of bananas. Since the average effect is 0, and the effect of hunger on the desirability 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 marginal 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")
```

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

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

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

Table 7.5: 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

```
power_analytic$power_B
```

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

```
power_analytic$power_AB
```

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

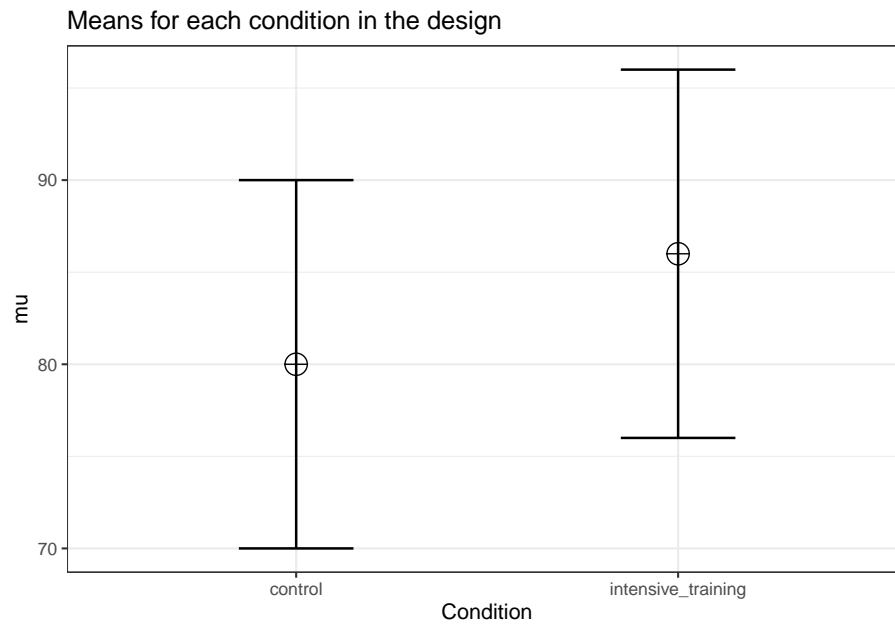

Chapter 8

Effect of Varying Designs on Power

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
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)
```

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

Table 8.1: Simulated ANOVA Result

	power	effect_size
anova_Condition	84.15	0.0911796

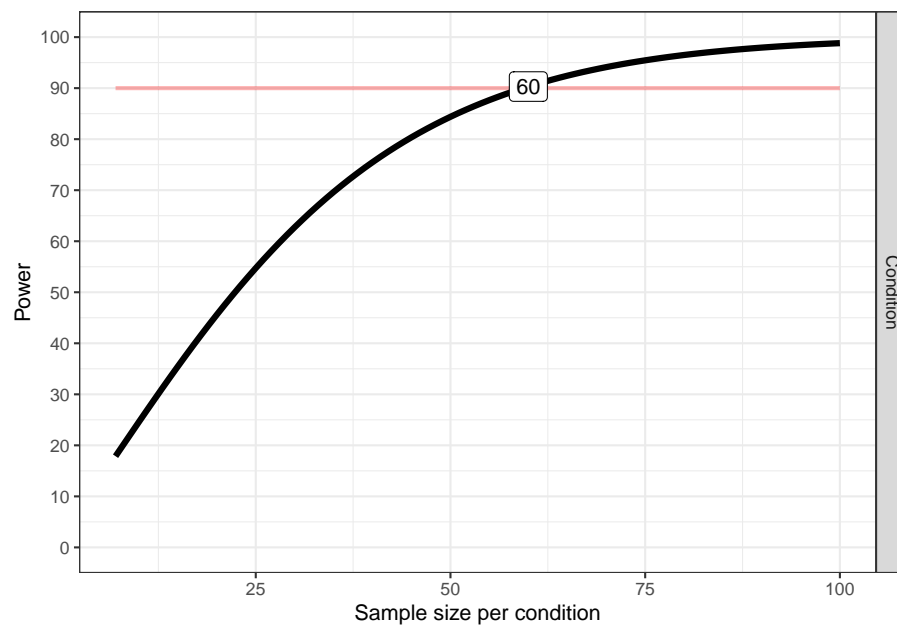
Table 8.2: Exact ANOVA Result

	power	partial_eta_squared	cohen_f	non_centrality
Condition	84.38754	0.0841121	0.3030458	9

We now add 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)
```

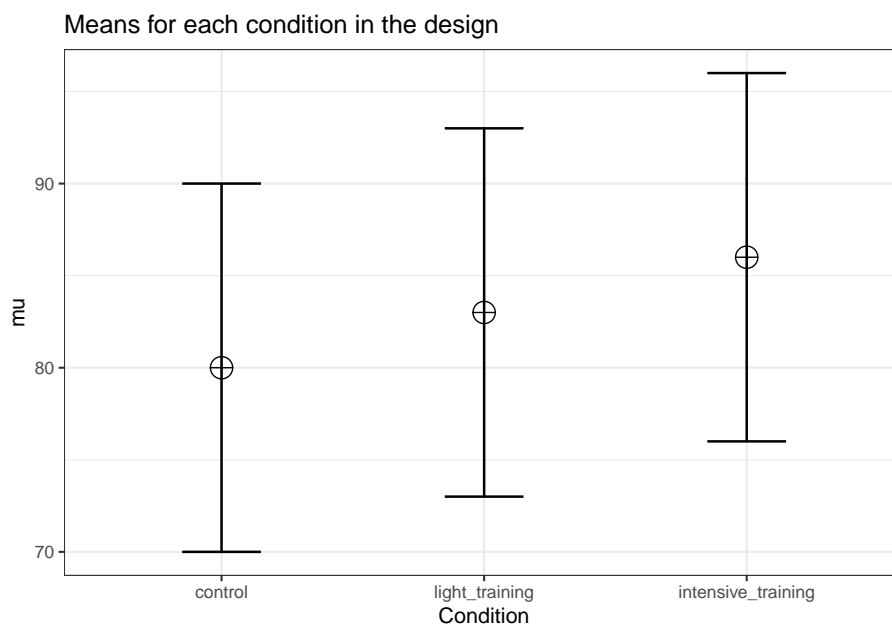


```
## Achieved Power and Sample Size for ANOVA-level effects
##   variable          label  n achieved_power desired_power
## 1 Condition Desired Power Achieved 60          90.31      90
```

```

string <- "3b"
n <- 50
mu <- c(80, 83, 86)
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

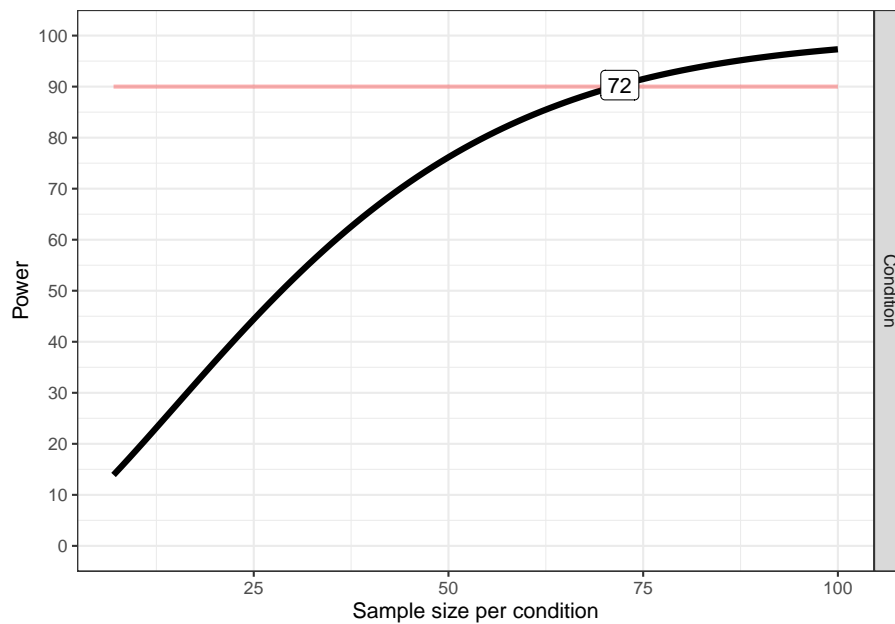
exact_result <- ANOVA_exact(design_result,
                             alpha_level = alpha_level,
                             verbose = FALSE)
```

Table 8.3: Exact ANOVA Result

	power	partial_eta_squared	cohen_f	non_centralilty
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.

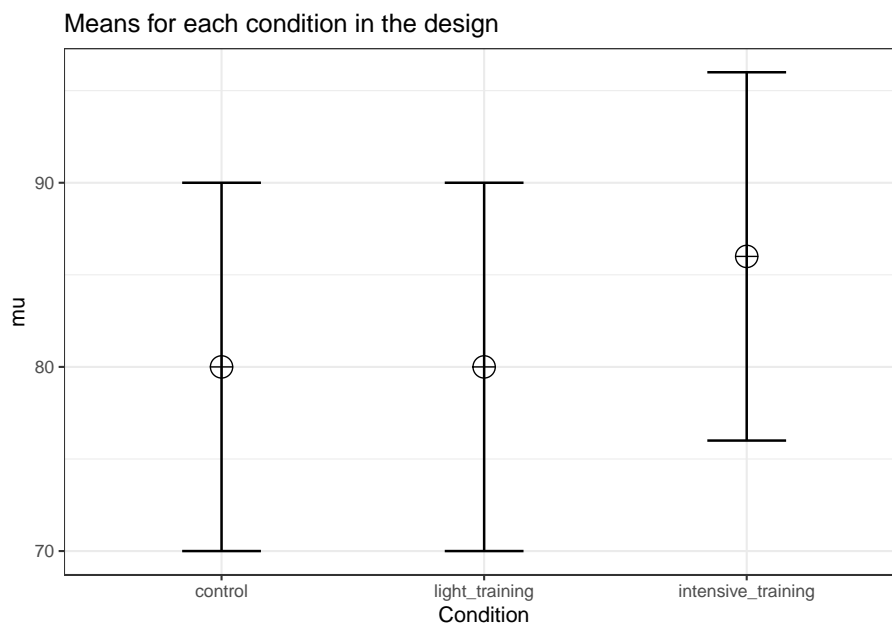
```
# Plot power curve (from 5 to 100)
plot_power(design_result, max_n = 100)
```



```
## Achieved Power and Sample Size for ANOVA-level effects
```

```
##      variable                label  n achieved_power desired_power
## 1 Condition Desired Power Achieved 72          90.29          90
```

```
string <- "3b"
n <- 50
mu <- c(80, 80, 86)
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
```

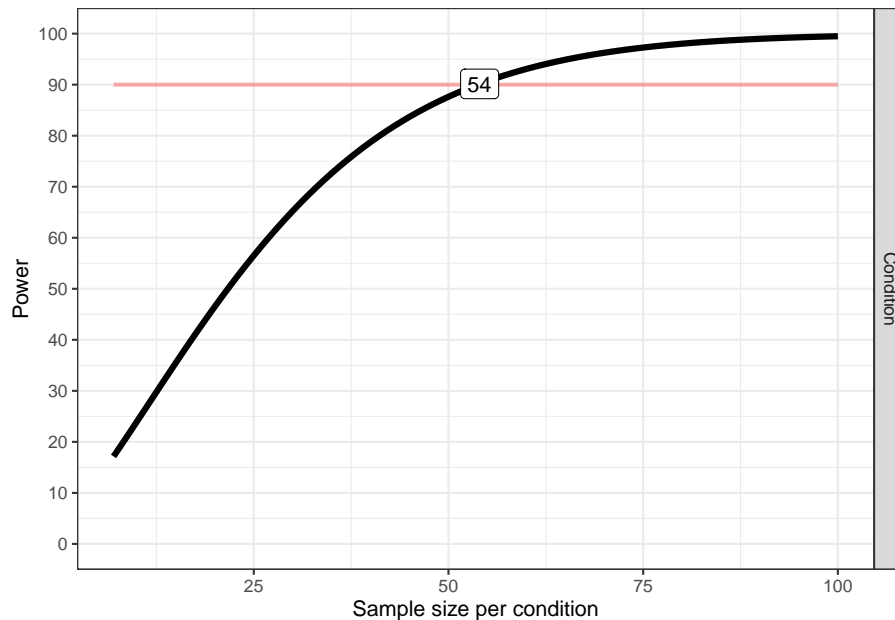
```
## [1] 0.2828427
```

```
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, including 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)
```



```
## Achieved Power and Sample Size for ANOVA-level effects
##   variable          label  n achieved_power desired_power
## 1 Condition Desired Power Achieved 54          90.15      90
```

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).

```
#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

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

## [1] 900

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 * 2)
SS_error

## [1] 10000

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 increases 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
```

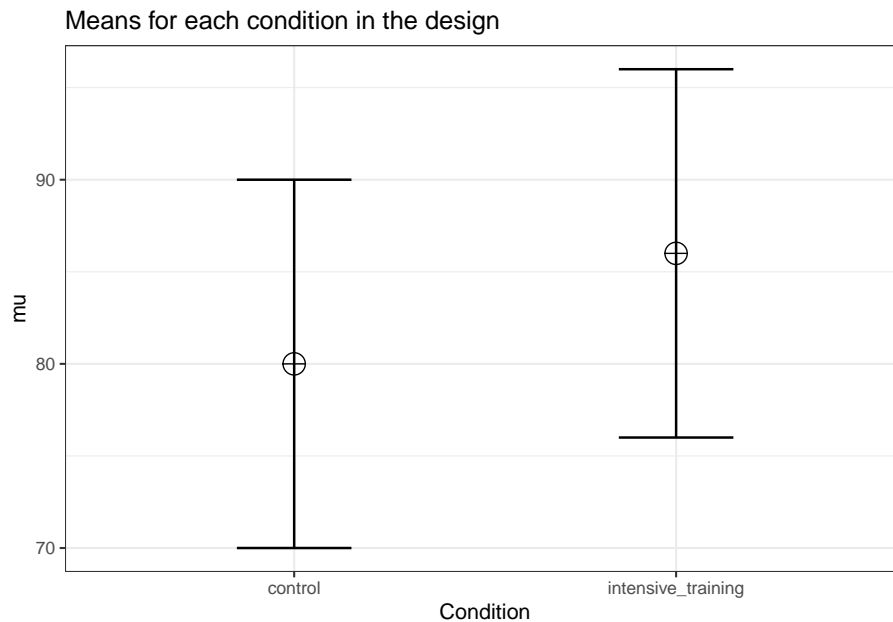
```
## [1] 0.2828427
```

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 .

8.1 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)
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
```

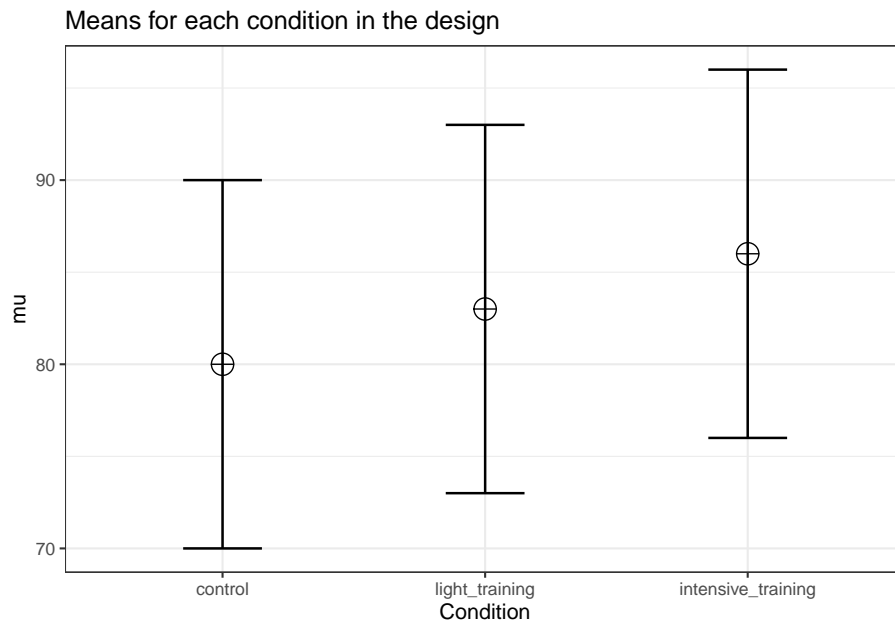
```
exact_result <- ANOVA_exact(design_result,
                             alpha_level = alpha_level,
                             verbose = FALSE)
```

Table 8.4: Exact ANOVA Result

	power	partial_eta_squared	cohen_f	non_centrality
Condition	83.66436	0.1551724	0.4285714	9

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 $n - 1$ 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)
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)
```

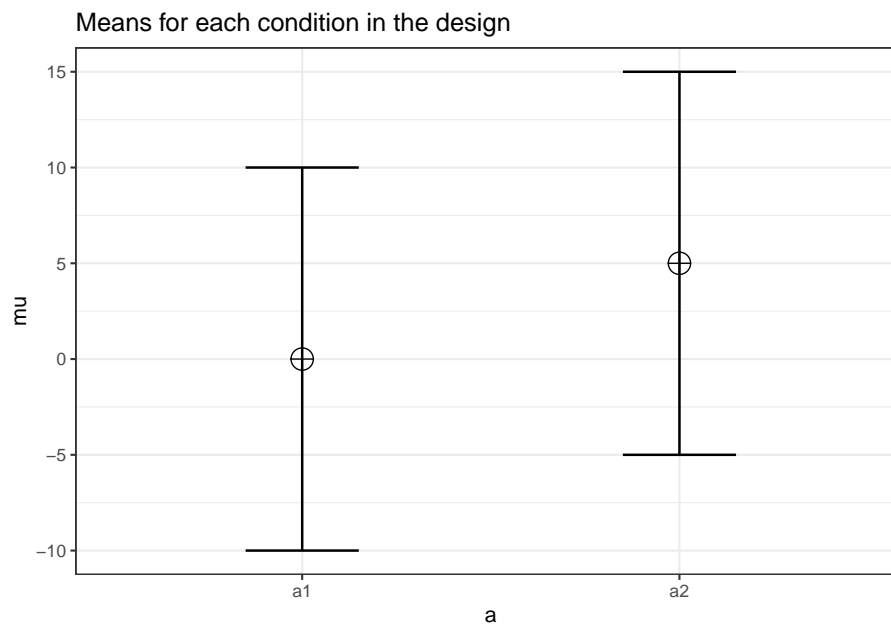
Table 8.5: Exact ANOVA Result

	power	partial_eta_squared	cohen_f	non_centrality
Condition	75.70841	0.0841121	0.3030458	9

When we add 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)
design_result <- ANOVA_design(
  design = string,
  n = n,
  mu = mu,
  sd = sd,
  r = r
)
```



```
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
```

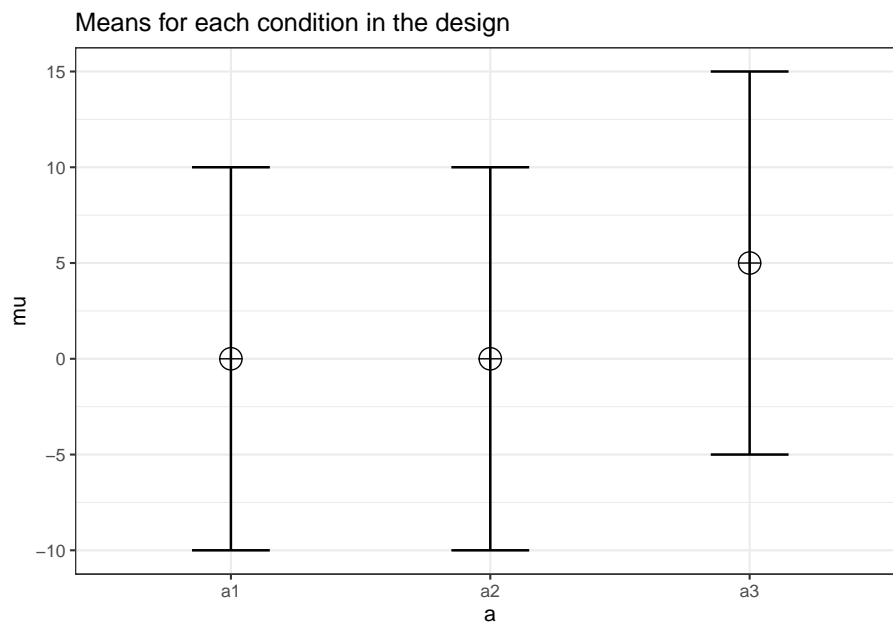
```
power_oneway_within(design_result)$lambda
```

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

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

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

```
string <- "3w"  
mu <- c(0, 0, 5)  
  
design_result <- ANOVA_design(  
  design = string,  
  n = n,  
  mu = mu,  
  sd = sd,  
  r = r  
)
```



```
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
```

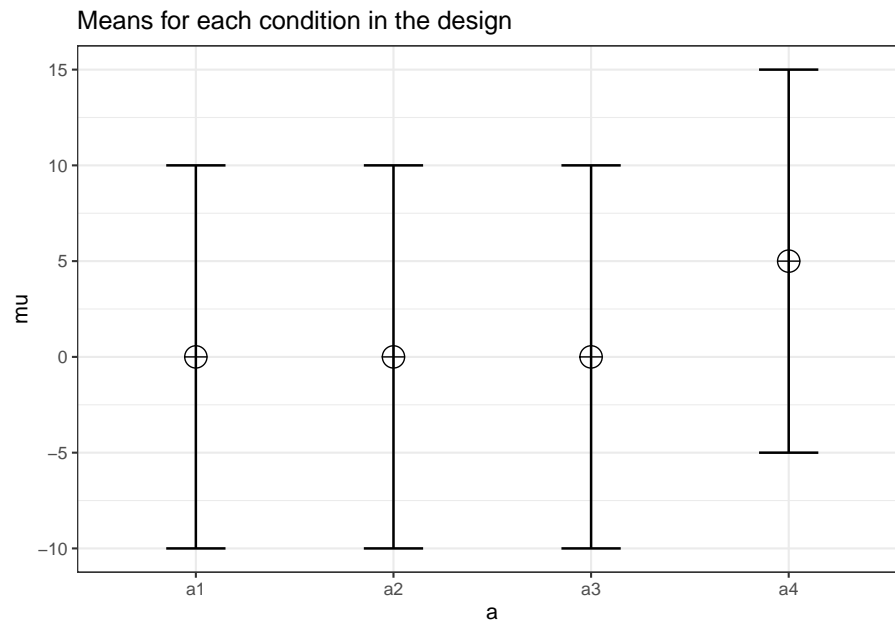
```
power_oneway_within(design_result)$lambda
```

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

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

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

```
string <- "4w"  
mu <- c(0, 0, 0, 5)  
design_result <- ANOVA_design(  
  design = string,  
  n = n,  
  mu = mu,  
  sd = sd,  
  r = r  
)
```



```
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
```

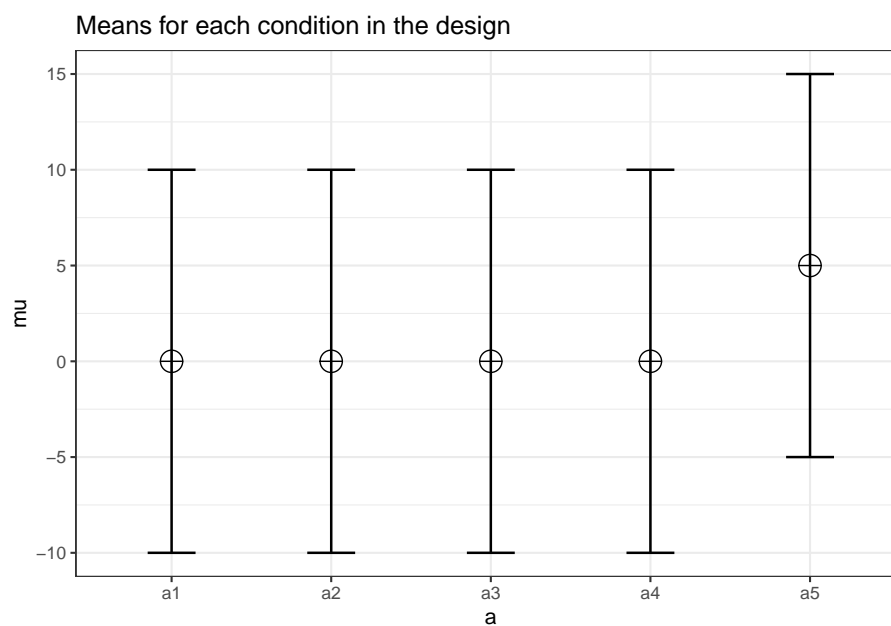
```
## [1] 11.25
```

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

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

```
string <- "5w"
mu <- c(0, 0, 0, 0, 5)

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



```
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
```

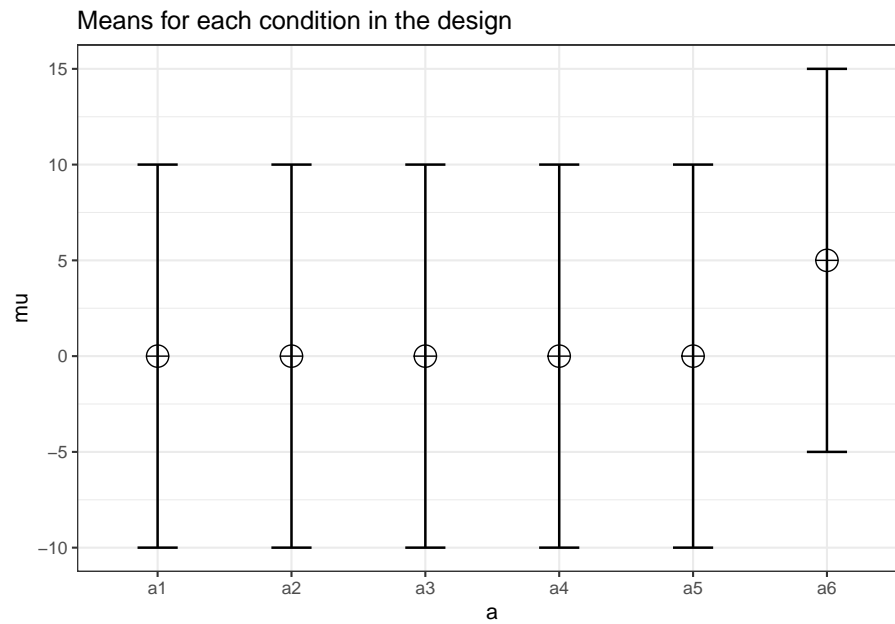
```
power_oneway_within(design_result)$lambda
```

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

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

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

```
string <- "6w"  
mu <- c(0, 0, 0, 0, 0, 5)  
design_result <- ANOVA_design(  
  design = string,  
  n = n,  
  mu = mu,  
  sd = sd,  
  r = r  
)
```



```
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
```

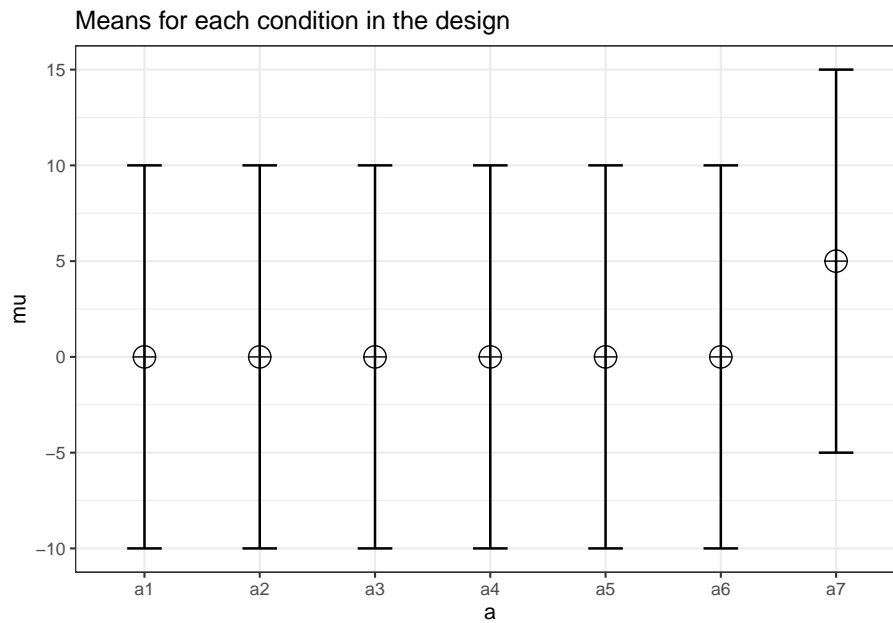
```
power_oneway_within(design_result)$lambda
```

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

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

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

```
string <- "7w"  
mu <- c(0, 0, 0, 0, 0, 0, 5)  
design_result <- ANOVA_design(  
  design = string,  
  n = n,  
  mu = mu,  
  sd = sd,  
  r = r  
)
```



```
print(paste0("power: ", power_oway_within(design_result)$power, "\nCohen's F: ", pow
        "\nCohen's F (SPSS): ", power_oway_within(design_result)$Cohen_f_SPSS,
        "\nlambda: ", power_oway_within(design_result)$lambda,
        "\nCritical F: ",
        power_oway_within(design_result)$F_critical
    ))
```

```
## [1] "power: 75.4600950274621\nCohen's F: 0.174963553055941\nCohen's F (SPSS): 0.271
```

```
# power_oway_within(design_result)$power
# power_oway_within(design_result)$Cohen_f
# power_oway_within(design_result)$Cohen_f_SPSS
# power_oway_within(design_result)$lambda
# power_oway_within(design_result)$F_critical
```

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 *g*power* (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).

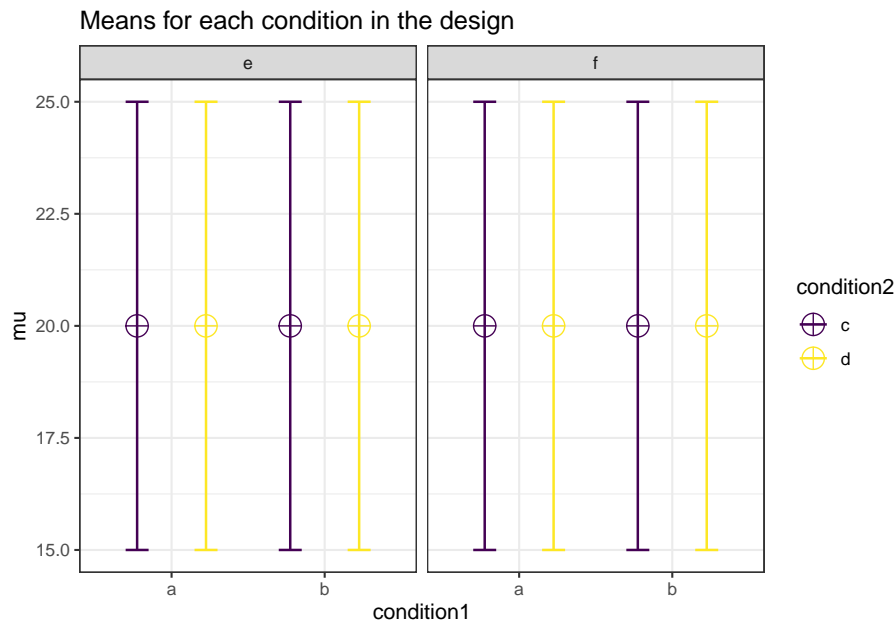
Overall, the effect of adding conditions with an effect close to the grand mean reduces power quite strongly, and adding conditions with means close to the extreme of the current conditions will either slightly increase or decrease power.

Chapter 9

Error Control in Exploratory ANOVA

In a $2 \times 2 \times 2$ 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 $2 \times 2 \times 2$ 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 9.1: Simulated ANOVA Result

	power	effect_size
anova_condition1	4.6	0.0024976
anova_condition2	3.7	0.0023969
anova_condition3	5.5	0.0026112
anova_condition1:condition2	4.4	0.0025654
anova_condition1:condition3	5.3	0.0026206
anova_condition2:condition3	4.6	0.0024337
anova_condition1:condition2:condition3	5.5	0.0026119

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)),4))
```

##	anova_condition1	anova_condition2
##	0.046	0.037
##	anova_condition3	anova_condition1:condition2
##	0.055	0.044
##	anova_condition1:condition3	anova_condition2:condition3
##	0.053	0.046
##	anova_condition1:condition2:condition3	
##	0.055	

This is the type I error rate for each test. The familywise error rate refers to the probability of one or more of the tests in the ANOVA being significant, even though none of the 7 effects exist (3 main effects, 3 two-way interactions, and 1 three-way 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)),4)) > 0) / nsims
```

```
## [1] 0.0275
```

But what we should do about this alpha inflation? We do not want to perform exploratory ANOVAs and observe more type I errors than expected, because these significant effects will often 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 made to alpha level for each test (above this was 0.05).

Afex relies on the `p.adjust` function 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.

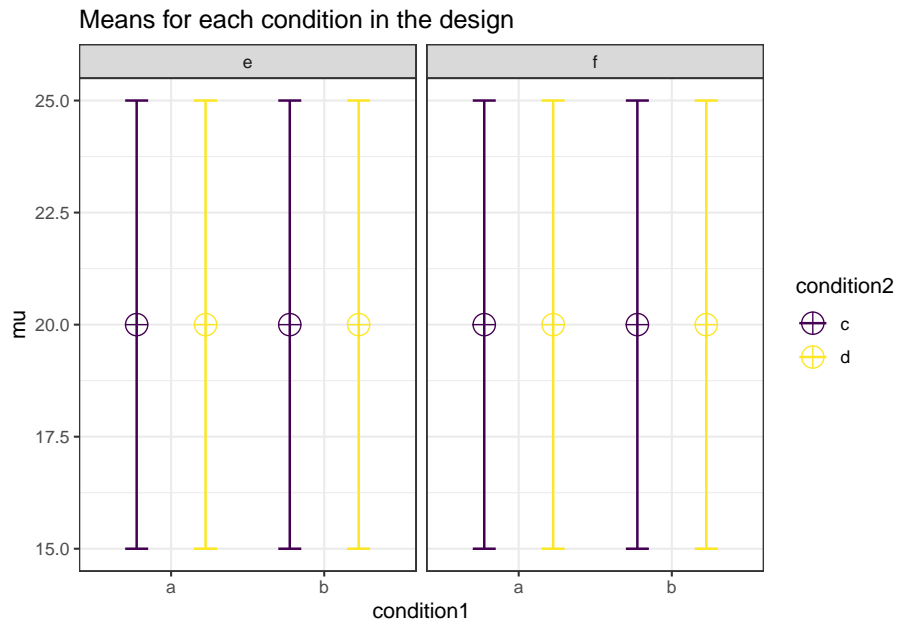
Hochberg’s and Hommel’s methods are valid when the hypothesis tests are independent or when they are non-negatively associated (Sarkar, 1998; Sarkar and Chang, 1997). Hommel’s method is more powerful than Hochberg’s, but the difference is usually small and the Hochberg p-values are faster to compute.

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 with 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)
```

Table 9.2: ANOVA Results

	power	effect_size
anova_condition1	1.3	0.0027223
anova_condition2	1.2	0.0027230
anova_condition3	0.5	0.0024579
anova_condition1:condition2	0.9	0.0026822
anova_condition1:condition3	1.0	0.0027651
anova_condition2:condition3	0.7	0.0023383
anova_condition1:condition2:condition3	0.6	0.0024676

Table 9.3: Pairwise Results

	power
p_condition1_a_condition2_c_condition3_e_condition1_a_condition2_c_condition3_f	0.0
p_condition1_a_condition2_c_condition3_e_condition1_a_condition2_d_condition3_e	0.1
p_condition1_a_condition2_c_condition3_e_condition1_a_condition2_d_condition3_f	0.1
p_condition1_a_condition2_c_condition3_e_condition1_b_condition2_c_condition3_e	0.6
p_condition1_a_condition2_c_condition3_e_condition1_b_condition2_c_condition3_f	0.0
p_condition1_a_condition2_c_condition3_e_condition1_b_condition2_d_condition3_e	0.2
p_condition1_a_condition2_c_condition3_e_condition1_b_condition2_d_condition3_f	0.1
p_condition1_a_condition2_c_condition3_f_condition1_a_condition2_d_condition3_e	0.1
p_condition1_a_condition2_c_condition3_f_condition1_a_condition2_d_condition3_f	0.2
p_condition1_a_condition2_c_condition3_f_condition1_b_condition2_c_condition3_e	0.2
p_condition1_a_condition2_c_condition3_f_condition1_b_condition2_c_condition3_f	0.2
p_condition1_a_condition2_c_condition3_f_condition1_b_condition2_d_condition3_e	0.2
p_condition1_a_condition2_c_condition3_f_condition1_b_condition2_d_condition3_f	0.2
p_condition1_a_condition2_d_condition3_e_condition1_a_condition2_d_condition3_f	0.1
p_condition1_a_condition2_d_condition3_e_condition1_b_condition2_c_condition3_e	0.3
p_condition1_a_condition2_d_condition3_e_condition1_b_condition2_c_condition3_f	0.2
p_condition1_a_condition2_d_condition3_e_condition1_b_condition2_d_condition3_e	0.0
p_condition1_a_condition2_d_condition3_e_condition1_b_condition2_d_condition3_f	0.1
p_condition1_a_condition2_d_condition3_f_condition1_b_condition2_c_condition3_e	0.0
p_condition1_a_condition2_d_condition3_f_condition1_b_condition2_c_condition3_f	0.0
p_condition1_a_condition2_d_condition3_f_condition1_b_condition2_d_condition3_e	0.0
p_condition1_a_condition2_d_condition3_f_condition1_b_condition2_d_condition3_f	0.0
p_condition1_b_condition2_c_condition3_e_condition1_b_condition2_c_condition3_f	0.2
p_condition1_b_condition2_c_condition3_e_condition1_b_condition2_d_condition3_e	0.1
p_condition1_b_condition2_c_condition3_e_condition1_b_condition2_d_condition3_f	0.3
p_condition1_b_condition2_c_condition3_f_condition1_b_condition2_d_condition3_e	0.0
p_condition1_b_condition2_c_condition3_f_condition1_b_condition2_d_condition3_f	0.0
p_condition1_b_condition2_d_condition3_e_condition1_b_condition2_d_condition3_f	0.2

```

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

## [1] 0.0061

```

We see it is close to 5%. Note that error rates have variation, and even in a ten thousand simulations, the error rate in the sample of studies can easily be half a percentage point higher or lower (see this blog). 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 number of tests). There are more powerful procedures to control the type I error rate, which require making more assumptions. For a small number of tests, the Holm-Bonferroni procedure works well. Alternative procedures to control error rates can be found in the multcomp R package (Hothorn et al., 2021).

Chapter 10

Analytic Power Functions

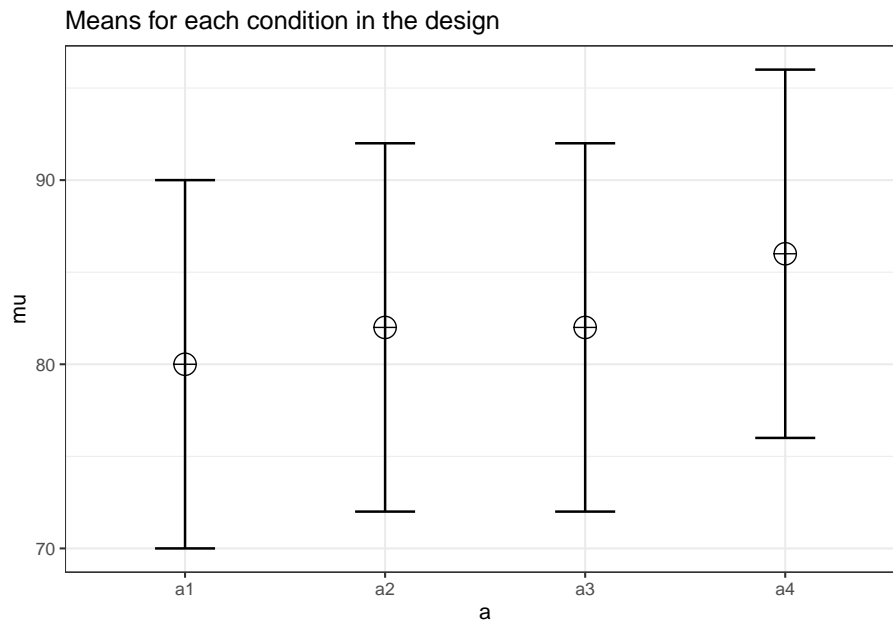
For some designs it is possible to calculate power analytically, using closed functions. Within the **Superpower** package we have included a number of these closed functions. As you will see below, each analytic function only serves a very narrow scenario while the simulations functions are much more 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).

10.1 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)

# Enter means in the order that matches the labels below.
sd <- 10
design_result <- ANOVA_design(design = string,
                             n = n,
                             mu = mu,
                             sd = sd)
```



```
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	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 `pwr2pp1` (Aberson, 2021) 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)
```

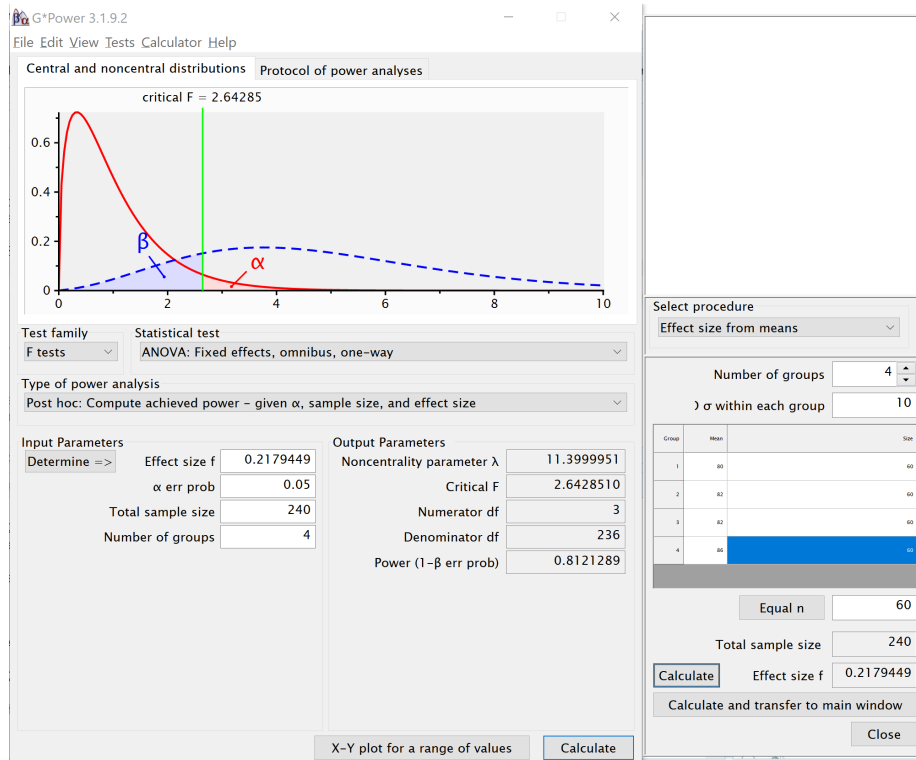
```
## Power = 0.812 for eta-squared = 0.05
```

We can also use the function in the **pwr** package (Champely, 2020). 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.

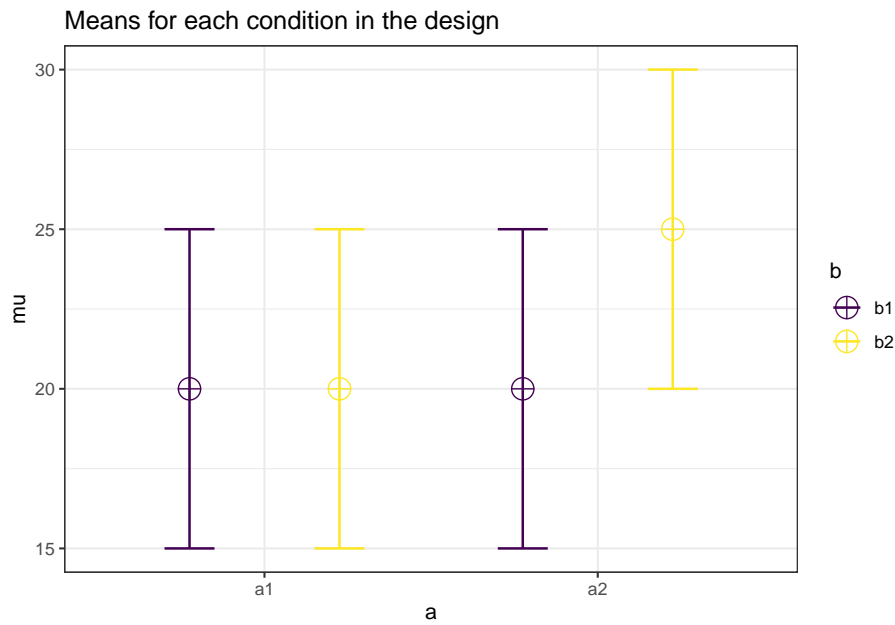


10.2 Two-way Between Subject Interaction

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

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



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

Table 10.2: Exact ANOVA Result

	power	partial_eta_squared	cohen_f	non_centralilty
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_tway_between`.

```
#using default alpha level of .05
power_res <- power_tway_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, 2021), 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
```

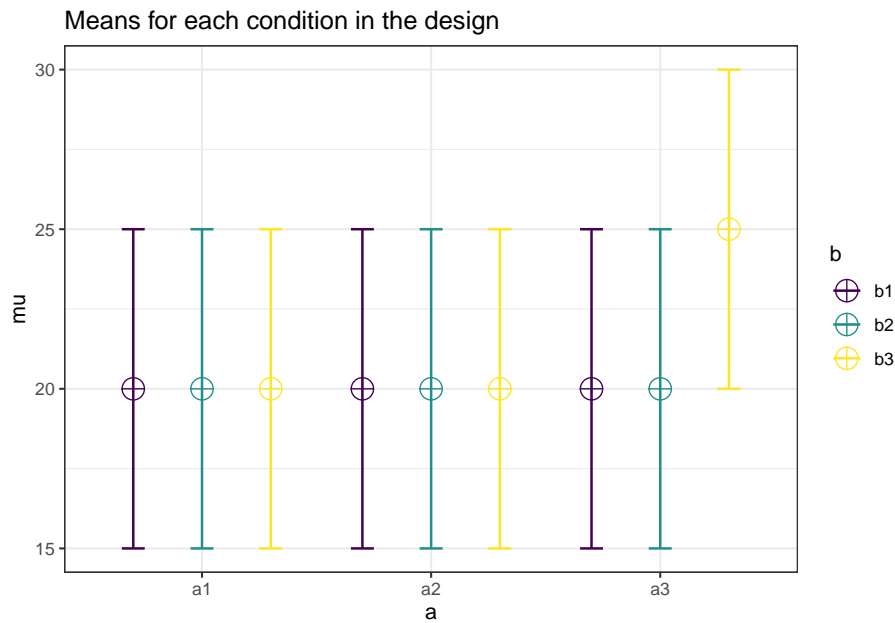
```
## Power for Interaction AxB = 0.598
```

10.3 3x3 Between Subject ANOVA

We can extend this function to a two-way design with 3 levels.

```
string <- "3b*3b"
n <- 20
mu <- c(20, 20, 20, 20, 20, 20, 20, 20, 25)
# Enter means in the order that matches the labels below.
sd <- 5

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



```
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_centralilty
a	44.86306	0.0253325	0.1612169	4.444444
b	44.86306	0.0253325	0.1612169	4.444444
a: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
```

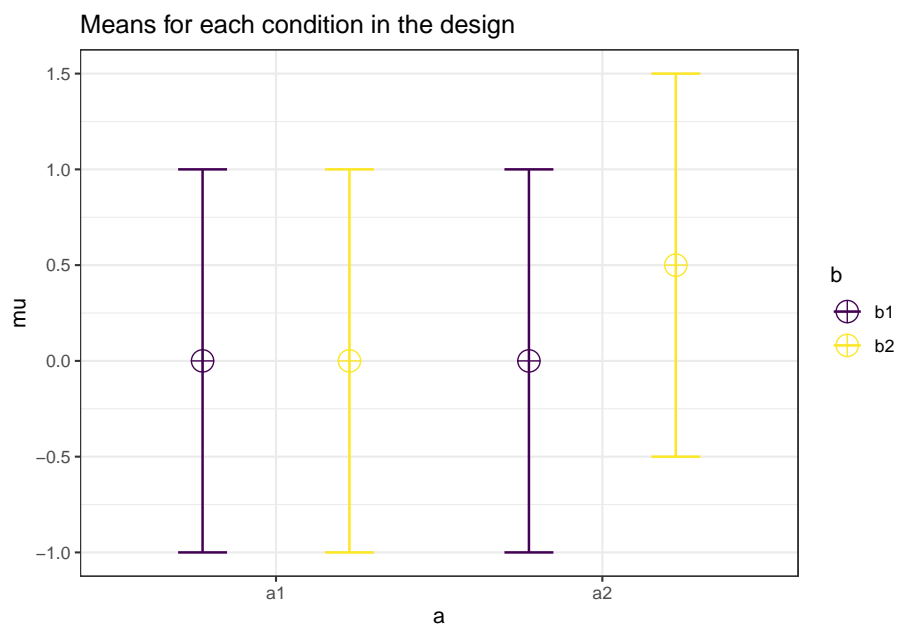

Chapter 11

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

design_result <- ANOVA_design(design = "2w*2w",
                             n = 20,
                             mu = c(0,0,0,0.5),
                             sd = 1,
                             r = 0.5)
```



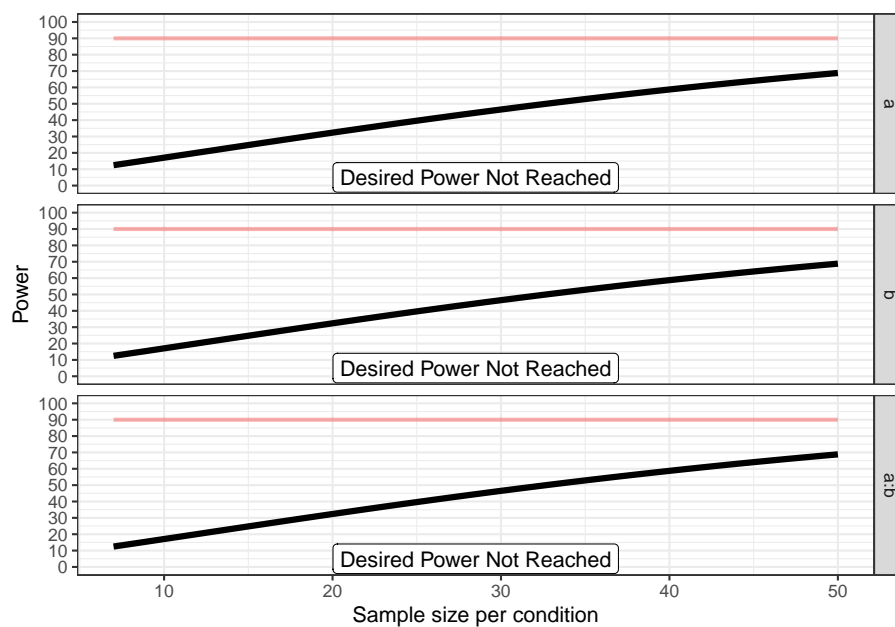
```
exact_result <- ANOVA_exact(design_result,
                             alpha_level = alpha_level,
                             verbose = FALSE)
```

Table 11.1: Exact ANOVA Result

	power	partial_eta_squared	cohen_f	non_central
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,
                  plot = TRUE,
                  verbose = TRUE)
```

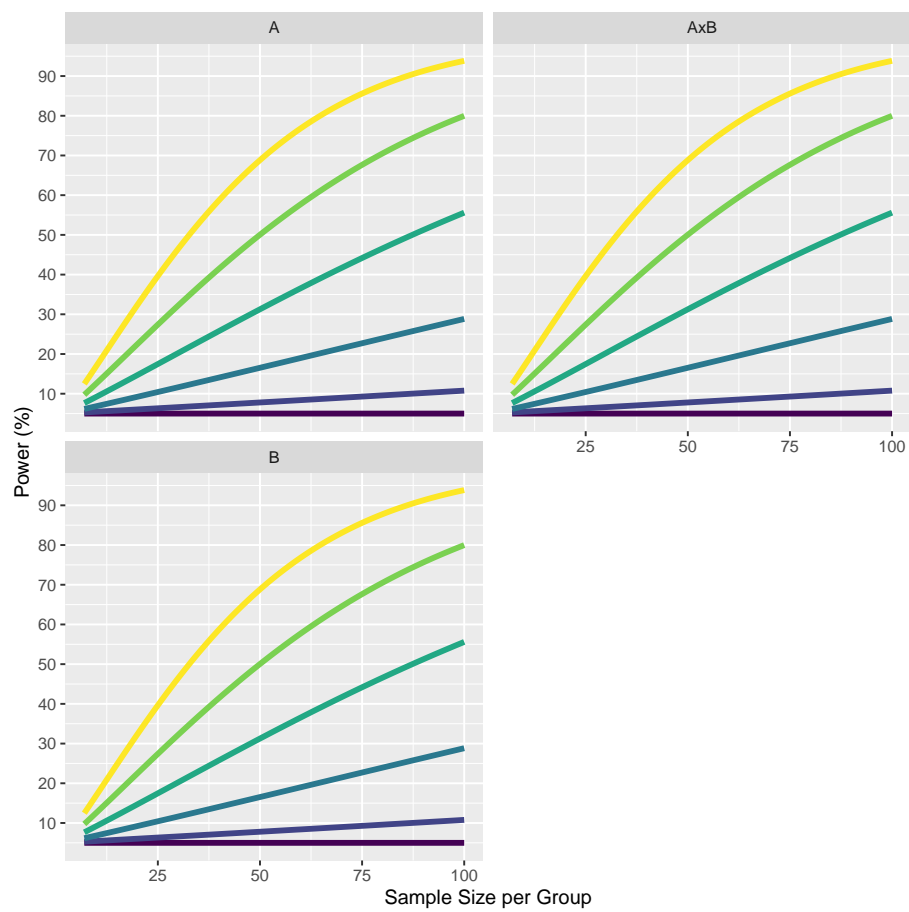


```
## Achieved Power and Sample Size for ANOVA-level effects
##   variable                label  n achieved_power desired_power
## 1         a Desired Power Not Reached 50          68.82          90
## 2         b Desired Power Not Reached 50          68.82          90
## 3        a:b Desired Power Not Reached 50          68.82          90
```

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 reproduce these plots can be found on the GitHub repository for this book.

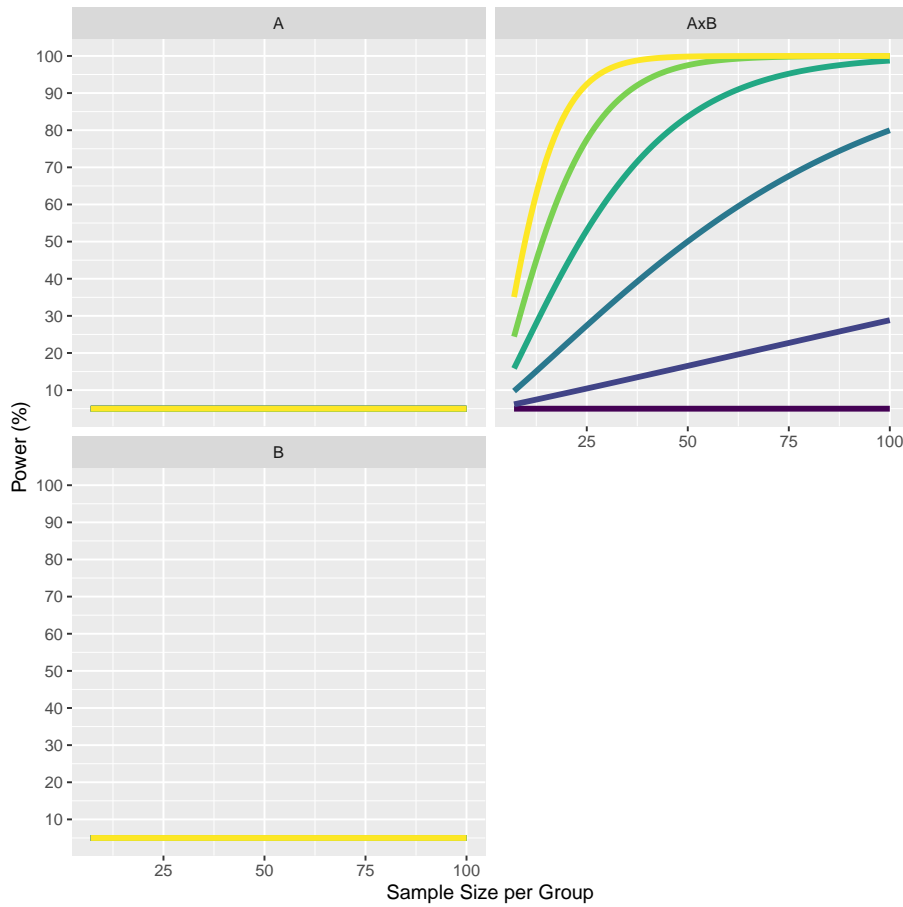
11.1 Explore increase in effect size for moderated interactions.

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.



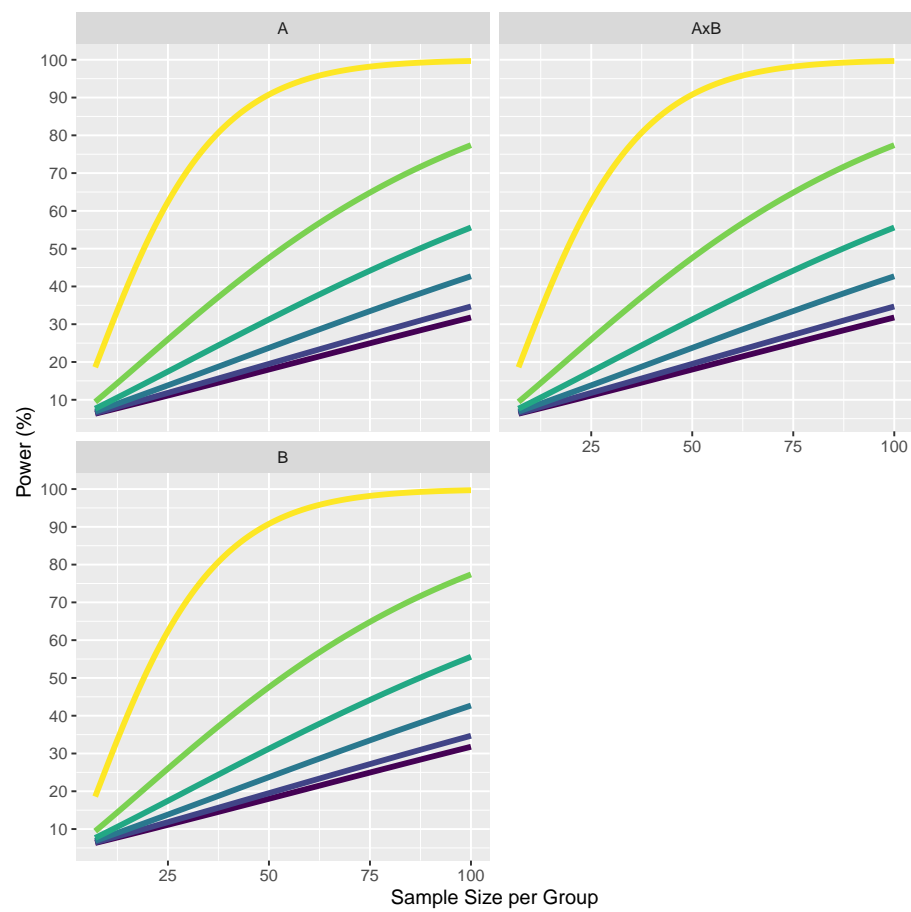
11.2 Explore increase in effect size for cross-over interactions.

The design has means 0, 0, 0, 0, with two cells increasing by 0.1, up to 0.5, 0, 0.5. The standard deviation is set to 1. The correlation between all variables is 0.5.



11.3 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.

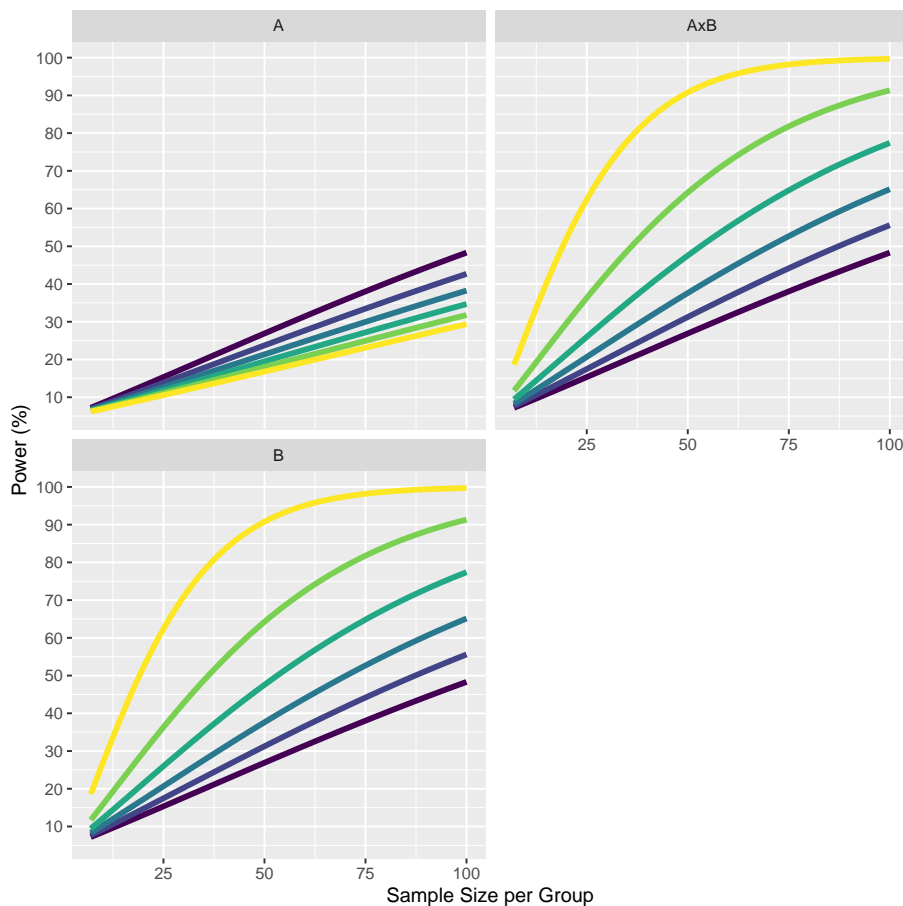


11.4 Increasing correlation in on factor decreases power in second factor

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.



11.5 Code to Reproduce Power Curve Figures

```

nsims = 10000
library(Superpower)
library(pwr)
library(tidyverse)
library(viridis)

string <- "2w*2w"
labelnames = c("A", "a1", "a2", "B", "b1", "b2")
design_result <- ANOVA_design(design = string,
                             n = 20,
                             mu = c(0,0,0,0.0),
                             sd = 1,
                             r = 0.5,
                             labelnames = labelnames)

p_a <- plot_power(design_result,
                  max_n = 100)
p_a$power_df$effect <- 0

design_result <- ANOVA_design(design = string,
                             n = 20,
                             mu = c(0,0,0,0.1),
                             sd = 1,
                             r = 0.5,
                             labelnames = labelnames)

p_b <- plot_power(design_result,
                  max_n = 100)

p_b$power_df$effect <- 0.1

design_result <- ANOVA_design(design = string,
                             n = 20,
                             mu = c(0,0,0,0.2),
                             sd = 1,
                             r = 0.5,
                             labelnames = labelnames)

p_c <- plot_power(design_result,
                  max_n = 100)

p_c$power_df$effect <- 0.2

design_result <- ANOVA_design(design = string,
                             n = 20,

```



```

                                mu = c(0,0,0,0.3),
                                sd = 1,
                                r = 0.5,
                                labelnames = labelnames)
p_d <- plot_power(design_result,
                  max_n = 100)

p_d$power_df$effect <- 0.3

design_result <- ANOVA_design(design = string,
                              n = 20,
                              mu = c(0,0,0,0.4),
                              sd = 1,
                              r = 0.5,
                              labelnames = labelnames)
p_e <- plot_power(design_result,
                  max_n = 100)

p_e$power_df$effect <- 0.4

design_result <- ANOVA_design(design = string,
                              n = 20,
                              mu = c(0,0,0,0.5),
                              sd = 1,
                              r = 0.5,
                              labelnames = labelnames)
p_f <- plot_power(design_result,
                  max_n = 100)

p_f$power_df$effect <- 0.5

plot_data <- rbind(p_a$power_df, p_b$power_df, p_c$power_df,
                  p_d$power_df, p_e$power_df, p_f$power_df)%>%
  mutate(AxB = `A:B`) %>% select(n, effect, A, B, AxB)

long_data = plot_data %>%
  gather("A", "B", "AxB", key = factor, value = power)

plot1 <- ggplot(long_data, aes(x = n, y = power,
                              color = as.factor(effect))) +
  geom_line(size = 1.5) +
  scale_y_continuous(breaks = seq(0,100,10)) +
  labs(color = "Effect Size",
       x = "Sample Size per Group",

```

```

      y = "Power (%)") +
      scale_color_viridis_d() + facet_wrap( ~ factor, ncol=2)

####

design_result <- ANOVA_design(design = string,
                             n = 20,
                             mu = c(0,0,0,0.0),
                             sd = 1,
                             r = 0.5,
                             labelnames = labelnames)
p_a <- plot_power(design_result,
                  max_n = 100)
p_a$power_df$effect <- 0

design_result <- ANOVA_design(design = string,
                             n = 20,
                             mu = c(0.1,0,0,0.1),
                             sd = 1,
                             r = 0.5,
                             labelnames = labelnames)
p_b <- plot_power(design_result,
                  max_n = 100)

p_b$power_df$effect <- 0.1

design_result <- ANOVA_design(design = string,
                             n = 20,
                             mu = c(0.2,0,0,0.2),
                             sd = 1,
                             r = 0.5,
                             labelnames = labelnames)
p_c <- plot_power(design_result,
                  max_n = 100)

p_c$power_df$effect <- 0.2

design_result <- ANOVA_design(design = string,
                             n = 20,
                             mu = c(0.3,0,0,0.3),
                             sd = 1,
                             r = 0.5,
                             labelnames = labelnames)
p_d <- plot_power(design_result,
                  max_n = 100)

```

```

p_d$power_df$effect <- 0.3

design_result <- ANOVA_design(design = string,
                             n = 20,
                             mu = c(0.4,0,0,0.4),
                             sd = 1,
                             r = 0.5,
                             labelnames = labelnames)
p_e <- plot_power(design_result,
                  max_n = 100)

p_e$power_df$effect <- 0.4

design_result <- ANOVA_design(design = string,
                             n = 20,
                             mu = c(0.5,0,0,0.5),
                             sd = 1,
                             r = 0.5,
                             labelnames = labelnames)
p_f <- plot_power(design_result,
                  max_n = 100)

p_f$power_df$effect <- 0.5

plot_data <- rbind(p_a$power_df, p_b$power_df, p_c$power_df,
                  p_d$power_df, p_e$power_df, p_f$power_df)%>%
  mutate(AxB = `A:B`) %>% select(n, effect, A, B, AxB)

long_data = plot_data %>%
  gather("A", "B", "AxB", key = factor, value = power)

plot2 <- ggplot(long_data, aes(x = n, y = power,
                              color = as.factor(effect))) +
  geom_line(size = 1.5) +
  scale_y_continuous(breaks = seq(0,100,10)) +
  labs(color = "Effect Size",
       x = "Sample Size per Group",
       y = "Power (%)") +
  scale_color_viridis_d() + facet_wrap( ~ factor, ncol=2)

##

string <- "2w*2w"
labelnames = c("A", "a1", "a2", "B", "b1", "b2")

```

```
design_result <- ANOVA_design(design = string,
                             n = 20,
                             mu = c(0,0,0,0.3),
                             sd = 1,
                             r = 0.0,
                             labelnames = labelnames)

p_a <- plot_power(design_result,
                 max_n = 100)

p_a$power_df$correlation <- 0.0

design_result <- ANOVA_design(design = string,
                             n = 20,
                             mu = c(0,0,0,0.3),
                             sd = 1,
                             r = 0.1,
                             labelnames = labelnames)

p_b <- plot_power(design_result,
                 max_n = 100)

p_b$power_df$correlation <- 0.1

design_result <- ANOVA_design(design = string,
                             n = 20,
                             mu = c(0,0,0,0.3),
                             sd = 1,
                             r = 0.3,
                             labelnames = labelnames)

p_c <- plot_power(design_result,
                 max_n = 100)

p_c$power_df$correlation <- 0.3

design_result <- ANOVA_design(design = string,
                             n = 20,
                             mu = c(0,0,0,0.3),
                             sd = 1,
                             r = 0.5,
                             labelnames = labelnames)

p_d <- plot_power(design_result,
                 max_n = 100)
```

```

p_d$power_df$correlation <- 0.5

design_result <- ANOVA_design(design = string,
                             n = 20,
                             mu = c(0,0,0,0.3),
                             sd = 1,
                             r = 0.7,
                             labelnames = labelnames)
p_e <- plot_power(design_result,
                  max_n = 100)

p_e$power_df$correlation <- 0.7

design_result <- ANOVA_design(design = string,
                             n = 20,
                             mu = c(0,0,0,0.3),
                             sd = 1,
                             r = 0.9,
                             labelnames = labelnames)
p_f <- plot_power(design_result,
                  max_n = 100)

p_f$power_df$correlation <- 0.9

plot_data <- rbind(p_a$power_df, p_b$power_df,
                  p_c$power_df, p_d$power_df, p_e$power_df, p_f$power_df)%>%
  mutate(AxB = `A:B`) %>% select(n,correlation,A,B,AxB)

long_data = plot_data %>%
  gather("A", "B", "AxB", key = factor, value = power)

plot3 <- ggplot(long_data, aes(x = n, y = power,
                              color = as.factor(correlation))) +
  geom_line(size = 1.5) +
  scale_y_continuous(breaks = seq(0,100,10)) +
  labs(color = "Correlation",
       x = "Sample Size per Group",
       y = "Power (%)") +
  scale_color_viridis_d() + facet_wrap( ~ factor, ncol=2)
#####

string <- "2w*2w"
labelnames = c("A", "a1", "a2", "B", "b1", "b2")

```

```

design_result <- ANOVA_design(design = string,
                             n = 20,
                             mu = c(0,0,0,0.3),
                             sd = 1,
                             r <- c(
                               0.4, 0.4, 0.4,
                               0.4, 0.4,
                               0.4),
                             labelnames = labelnames)
p_a <- plot_power(design_result,
                  max_n = 100)

p_a$power_df$corr_diff <- 0

design_result <- ANOVA_design(design = string,
                             n = 20,
                             mu = c(0,0,0,0.3),
                             sd = 1,
                             r <- c(
                               0.5, 0.4, 0.4,
                               0.4, 0.4,
                               0.5),
                             labelnames = labelnames)
p_b <- plot_power(design_result,
                  max_n = 100)

p_b$power_df$corr_diff <- 0.1

design_result <- ANOVA_design(design = string,
                             n = 20,
                             mu = c(0,0,0,0.3),
                             sd = 1,
                             r <- c(
                               0.6, 0.4, 0.4,
                               0.4, 0.4,
                               0.6),
                             labelnames = labelnames)
p_c <- plot_power(design_result,
                  max_n = 100)

p_c$power_df$corr_diff <- 0.2

design_result <- ANOVA_design(design = string,
                             n = 20,

```

```

mu = c(0,0,0,0.3),
sd = 1,
r <- c(
  0.7, 0.4, 0.4,
  0.4, 0.4,
  0.7),
labelnames = labelnames)
p_d <- plot_power(design_result,
  max_n = 100)

p_d$power_df$corr_diff <- 0.3

design_result <- ANOVA_design(design = string,
  n = 20,
  mu = c(0,0,0,0.3),
  sd = 1,
  r <- c(
    0.8, 0.4, 0.4,
    0.4, 0.4,
    0.8),
  labelnames = labelnames)
p_e <- plot_power(design_result,
  max_n = 100)

p_e$power_df$corr_diff <- 0.4

design_result <- ANOVA_design(design = string,
  n = 20,
  mu = c(0,0,0,0.3),
  sd = 1,
  r <- c(
    0.9, 0.4, 0.4,
    0.4, 0.4,
    0.9),
  labelnames = labelnames)
p_f <- plot_power(design_result,
  max_n = 100)

p_f$power_df$corr_diff <- 0.5

plot_data <- rbind(p_a$power_df, p_b$power_df, p_c$power_df,
  p_d$power_df, p_e$power_df, p_f$power_df)%>%
  mutate(AxB = `A:B`) %>% select(n,corr_diff,A,B,AxB)

long_data = plot_data %>%

```

```
gather("A", "B", "AxB", key = factor, value = power)

plot4 <- ggplot(long_data, aes(x = n, y = power,
                              color = as.factor(corr_diff))) +
  geom_line(size = 1.5) +
  scale_y_continuous(breaks = seq(0,100,10)) +
  labs(color = "Difference in Correlation",
       x = "Sample Size per Group",
       y = "Power (%)") +
  scale_color_viridis_d() + facet_wrap( ~ factor, ncol=2)
```


Chapter 12

Violations of Assumptions

So far we have shown how simulations can be useful for power analyses for ANOVA designs where all assumptions of the statistical tests are met. An ANOVA is quite robust against violations of the normality assumption, which means the Type 1 error rate remains close to the alpha level specified in the test. Violations of the homogeneity of variances assumption can be more impactful, especially when sample sizes are unequal between conditions. When the equal variances assumption is violated for a one-way ANOVA, Welch's F -test is a good default.

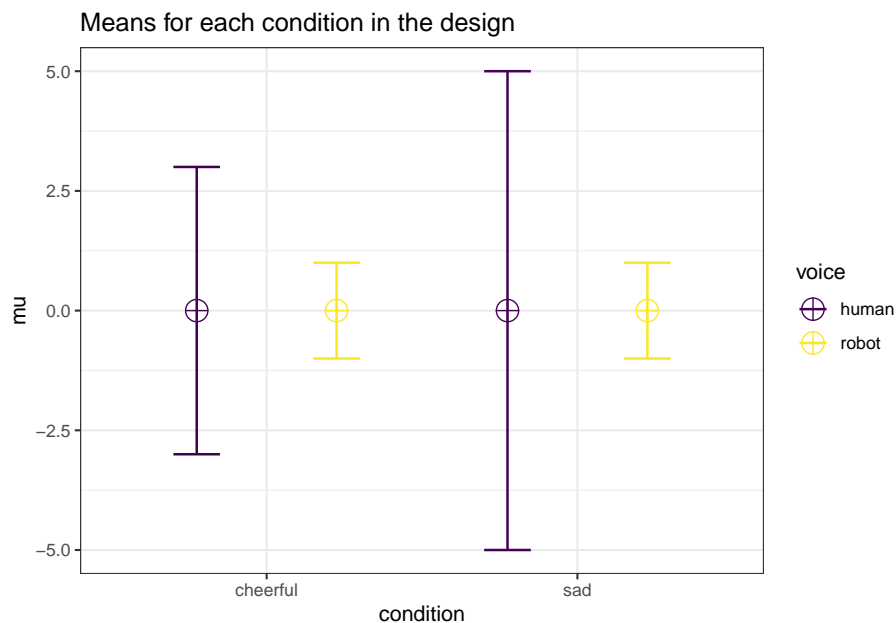
When the sphericity assumption in within designs is violated (when the variances of the differences between all pairs are not equal) a sphericity correction can be applied (e.g., the Greenhouse-Geisser or Huynh-Feldt correction) or a Multivariate ANOVA (MANOVA) can be performed. Alternative approaches for ANOVA designs with multiple between factors exist, such as heteroskedasticity robust standard errors. Superpower allows researchers to perform power analyses for unequal variances (or correlations) applying sphericity corrections, or a MANOVA. We aim to include the option to perform Welch's F -test in the future.

12.1 Violation of Heterogeneity Assumption

Although some recommendations have been provided to assist researchers to choose an approach to deal with violations of the homogeneity assumption (Algina and Keselman, 1997), it is often unclear if these violations of the homogeneity assumption are consequential for a given study. So far we have used simulations in Superpower to simulate patterns of means where there is a true effect, but we can also simulate a null effect. Such Monte Carlo simulation studies are used in published articles to examine the Type 1 error rate under a

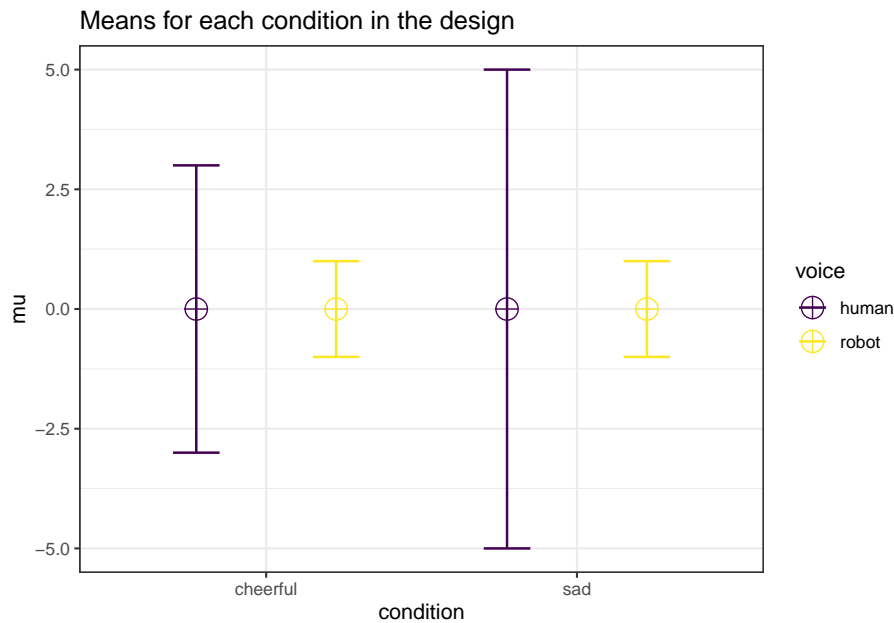
range of assumptions and while performing different tests. Superpower makes it easy to perform such simulations studies for the specific scenario a researcher is faced with, and can help to make a decision whether violations of assumptions are something to worry about, and whether choices to deal with violations are sufficient.

```
design_result_violation <- ANOVA_design(
  design = "2b*2b",
  n = c(20, 80, 40, 80),
  mu = c(0, 0, 0, 0),
  sd = c(3, 1, 5, 1),
  labelnames = c("condition", "cheerful", "sad", "voice", "human", "robot"))
```



```
# power_result_violation <- ANOVA_power(
#   design_result_violation,
#   nsims = 10000,
#   verbose = TRUE)

design_result_violation_2 <- ANOVA_design(
  design = "2b*2b",
  n = c(80, 80, 80, 80),
  mu = c(0, 0, 0, 0),
  sd = c(3, 1, 5, 1),
  labelnames = c("condition", "cheerful", "sad", "voice", "human", "robot"))
```



```
# power_result_violation_2 <- ANOVA_power_unequal_n(
#   design_result_violation_2,
#   nsims = 10000,
#   verbose = TRUE)
```

As an example, let's revisit our earlier 2x2 between subjects design. Balanced designs (the same sample size in each condition) reduce the impact of violations of the homogeneity assumption, but let's assume that for some reason sample sizes varied between 20 and 80 per cell, and the population standard violations varied extremely across conditions (from 1 to 5). We can use Superpower to estimate the impact of violating the homogeneity assumption by simulating a null effect (the means in all conditions are the same) and examining the Type 1 error rate. We can specify a design with unequal sample sizes and unequal variances as illustrated in the code below.

```
design_result_violation <- ANOVA_design(
  design = "2b*2b",
  n = c(20, 80, 40, 80),
  mu = c(0, 0, 0, 0),
  sd = c(3, 1, 5, 1),
  labelnames = c("condition", "cheerful", "sad", "voice", "human", "robot"))
)

power_result_violation <- ANOVA_power(
```

```
design_result_violation,
nsims = 100000)
```

Based on this simulation, the Type 1 error rate for the main effects and interactions for the ANOVA are approximately 14.5%. Under these assumptions it is clear that the Type 1 error rate is too high. One solution would be to make sure that an experiment has equal sample sizes. If we re-run the simulation with equal sample sizes, the Type 1 error rate is reduced to 5.36%, which is acceptable, even though the standard deviations are not equal.

12.2 Violation of the sphericity assumption

We can also use Superpower to estimate the impact of violating the assumption of sphericity on the type 1 error rate for a simple one-way repeated measures ANOVA. The sphericity assumption entails that the variances of the differences between all possible pairs of within-subject conditions (i.e., levels of the independent variable) are equal. This assumption is often tenuous at best in real life, and is typically “adjusted” for by applying a sphericity correction (e.g., Greenhouse-Geisser or Huynh-Feldt adjusted output).

An inquiring researcher may wonder, how much will this violation inflate the type 1 error rate? We can simulate an example (below) wherein there are no differences between the repeated measures (i.e., the null-hypothesis is true) and the correlations between the levels vary from $r = .05$ to $r = .90$, and the standard deviations from 1 to 7. This leads to unequal variances of the differences between conditions, and thus violates the sphericity assumption. We can simulate the consequences by specifying the design in the ANOVA_design function.

```
design_result_violation_3 <- ANOVA_design(
  design = "4w",
  n = 29,
  r = c(.05, .15, .25, .55, .65, .9),
  sd = c(1,3,5,7),
  mu = c(0, 0, 0, 0)
)
```

We then perform the Monte Carlo simulation using the ANOVA_power function.

```
power_result_violation_3 <- ANOVA_power(design_result_violation_3,
  alpha_level = 0.05,
  nsims = 10000)
```

The simulated type 1 error rate for the univariate ANOVA is 8.25%. The multivariate ANOVA (MANOVA) does not assume sphericity and therefore should be robust to this pattern of correlations. We can then check the MANOVA results and see this analysis approach maintains the Type 1 error rate at 4.91%. Alternatively, we could re-run the simulation with a Greenhouse-Geisser (`correction = "GG"`) or Huynh-Feldt (`correction = "HF"`) corrections for sphericity.

```
power_result_violation_4 <- ANOVA_power(design_result_violation_4,
                                       alpha_level = 0.05,
                                       nsims = nsims,
                                       correction = "GG",
                                       verbose = FALSE)
power_result_violation_5 <- ANOVA_power(design_result_violation_4,
                                       alpha_level = 0.05,
                                       nsims = nsims,
                                       correction = "HF",
                                       verbose = FALSE)
```

The simulated type 1 error rate for the univariate ANOVA with a Greenhouse-Geisser correction is now 4.82% and it is 5.36% with a Huynh-Feldt correction. After we determine which analysis approach best preserves the Type 1 error rate, given assumptions about the data generation model (e.g., the standard deviations and correlations) we could then re-run the simulation with the pattern of means we want to detect and estimate the power for the given design. This way, we control our Type 1 error rate, and can estimate our statistical power for an analysis that handles violations of the sphericity assumption.

12.2.1 MANOVA or Sphericity Adjustment?

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 sphericity. 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 be used to examine whether such general guidelines actually make sense in a specific case. Researchers can simulate both the expected reality when the null hypothesis is true (no differences between means, to determine the Type 1 error rate) and when the alternative hypothesis is true (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 and corrections control Type 1 error rates well.

Table 12.1: Simulated MANOVA Result

	power
<code>manova_(Intercept)</code>	4.82
<code>manova_a</code>	4.91

Table 12.2: Simulated ANOVA Result

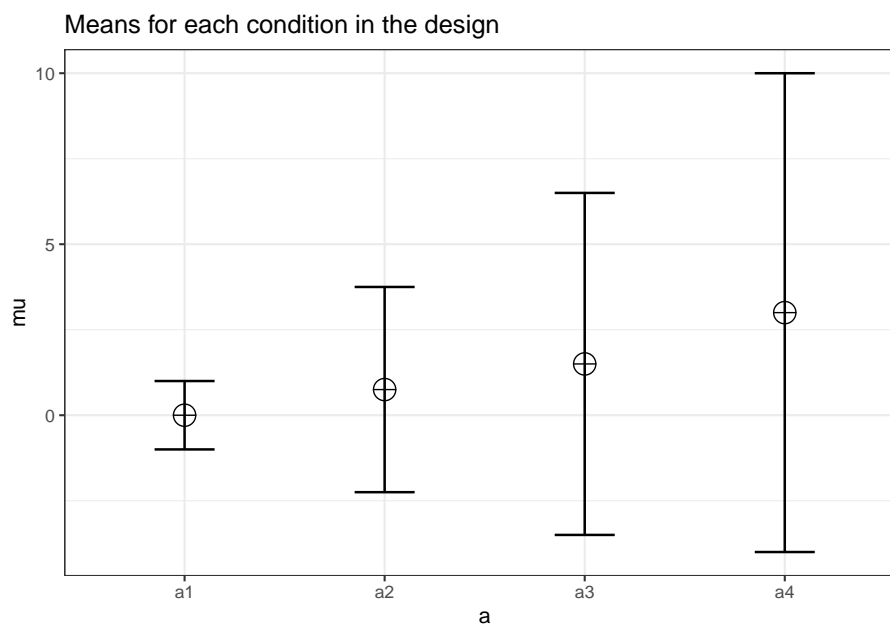
	power	effect_size
<code>anova_a</code>	4.82	0.034076

Table 12.3: Simulated ANOVA Result

	power	effect_size
<code>anova_a</code>	5.36	0.0344436

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. However, Greenhouse-Geisser seemed to be more 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 12.4: Simulated ANOVA Result

	power	effect_size
anova_a	61.22	0.1496984

Table 12.5: Simulated MANOVA Result

	power
manova_(Intercept)	50.07
manova_a	50.92

It appears the MANOVA based approach has roughly ~10% less power compared the HF-adjusted ANOVA. 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, and thus researchers could continue to simulate the effect of a larger sample size on the statistical power under different approaches, and make a final decision on the best approach to use in their study, based on assumptions about the data generating process.

Chapter 13

Estimated Marginal Means

Thank you to Fredrick Aust for developing the `emmeans_power` function

In many cases researchers may not be interested in the ANOVA-level effects, but rather in the power to detect a specific comparisons within the data. For example, you may have hypothesis about equivalence rather than a difference between groups. In these cases, we then need to obtain the “estimated marginal means” (EMMs), also known as the least squared means (`lsmeans` for SAS users), which can be done in R with the **emmeans** package and this is what **Superpower** uses “under the hood”. The EMMs refer to the mean of a group or set of groups within a statistical model.

For **Superpower**, you can estimate EMMs power for your designs in three different ways: `ANOVA_power`, `ANOVA_exact`, `emmeans_power`. Both `ANOVA_power` and `ANOVA_exact` allow for EMMs comparisons to be run along side the ANOVA-level comparisons. To do this, you will need to set a few parameters within the function.

1. `emm`: tell the function to run `emmeans` or not; default is set to `FALSE`.
2. `contrast_type`: This input will determine the contrast type for the `emmeans` comparisons. The default is “pairwise”. Possible input is limited to “pairwise”, “revpairwise”, “eff”, “consec”, “poly”, “del.eff”, “trt.vs.ctrl”, “trt.vs.ctrl1”, “trt.vs.ctrlk”, and “mean_chg”. See `help(“contrast-methods”)` with the **emmeans** package loaded for more information.
3. `emm_model`: `emmeans` accepts univariate and multivariate models. This will only make a difference if a within-subjects factor is included in the design. Setting this option to “multivariate” will result in the “multivariate” model being selected for the `emmeans` comparisons. *This is generally recommended if you are interested in the estimated marginal means.*
4. `emm_comp`: This selects the factors to be included for the `emmeans`. The default is to take the `frml2` object from the results of `ANOVA_design`, and

with the default `contrast_type = "pairwise"`, results in *all* the pairwise comparisons being performed. The simple effects can also be performed by including `|` in the `emm_comp` formula. For example, with two factors (e.g., `a` and `b`) `emm_comp = "a+b"` results in all pairwise comparisons being performed while `emm_comp = "a|b"` will result in pairwise comparisons across all levels of `a` **within** each level of `b`.

5. `emm_p_adjust` (ANOVA_power only): This sets the correction for multiple comparisons for the EMMs; default is "none". When `emmeans` is installed you can check `?summary.emmGrid` for more details on acceptable methods for EMMs. Please note that these corrections can only be applied in the `ANOVA_power` function at this time.

If you want to go beyond the simple comparisons that can be accomplished with `ANOVA_power` or `ANOVA_exact` then you can use the `emmeans_power` function directly. In this chapter, we will show how to use all three functions on their own or in combination with each other.

13.1 A note of caution

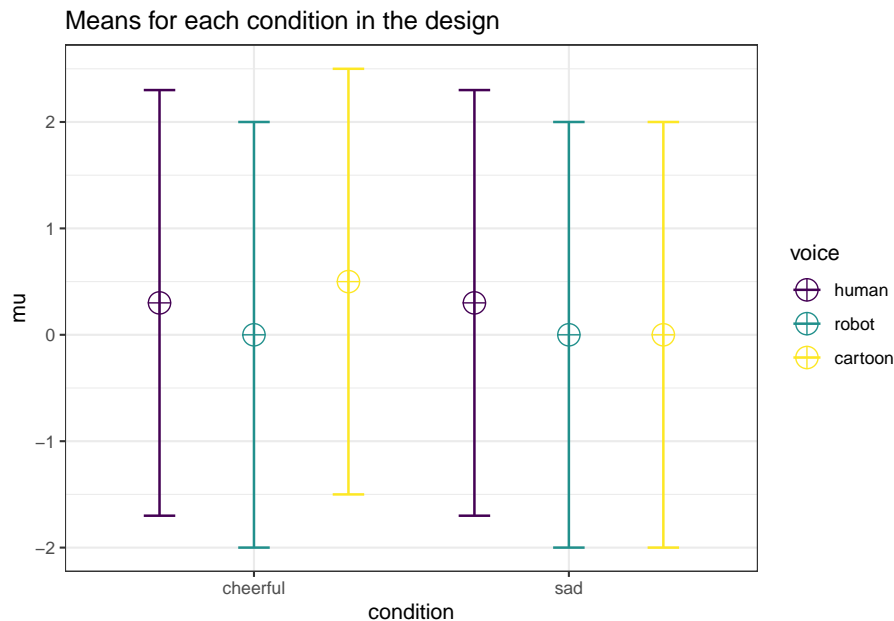
When conducting exact ANOVA power analyses with `Superpower` it is possible to calculate the power for both the omnibus F -tests and planned contrasts or post-hoc comparisons. All power analyses for `emmeans`-objects are based on the F - and t -values from the analyses of the dataset simulated by `ANOVA_exact()` assuming two-sided testing. Thus, the `emmeans_power()` does not honor adjustments of the testing procedure due to either one-sided testing (including two one-sided tests) or corrections for multiple comparisons via the `adjust` option in `emmeans`. As noted below, for the Bonferroni-adjustment this limitation can be overcome by manually adjusting `alpha_level`.

13.2 Pairwise Comparisons

This is the most common form of using EMMs. In our experience, many people will first inspect the ANOVA-level effects and then go into the individual group comparisons.

First, we can setup a 2×3 repeated measures design. When calling `ANOVA_exact()` pairwise comparisons of expected marginal means are added by setting `emm = TRUE` and `contrast_type = "pairwise"` (default). The `emm_comp` parameter will automatically be set to `condition*voice`. If you wanted to look at the main effect of `condition` or `voice` all you need to do is set `emm_comp` to that specific factor rather than the interaction.

```
# Set up a within design with 2 factors, each with 2 and 3 levels
design_result <- ANOVA_design(
  design = "2w*3w",
  n = 40,
  mu = c(0.3, 0, 0.5, 0.3, 0, 0),
  sd = 2,
  r = 0.8,
  labelnames = c("condition", "cheerful", "sad", "voice", "human", "robot", "cartoon"),
  plot = TRUE
)
```



```
exact_result <- ANOVA_exact(
  design_result,
  alpha_level = 0.05,
  verbose = FALSE,
  emm = TRUE,
  contrast_type = "pairwise"
)
```

The result contains the power calculations for both the omnibus F -tests and pairwise post-hoc comparisons.

```
knitr::kable(exact_result$main_results[1:2])
```

	power	partial_eta_squared
condition	29.08380	0.0507099
voice	50.12584	0.0621242
condition:voice	41.62968	0.0507099

```
knitr::kable((exact_result$emm_results[1:2]))
```

contrast	power
condition_cheerful voice_cartoon - condition_sad voice_cartoon	68.37160
condition_cheerful voice_cartoon - condition_cheerful voice_human	16.41134
condition_cheerful voice_cartoon - condition_sad voice_human	16.41134
condition_cheerful voice_cartoon - condition_cheerful voice_robot	68.37160
condition_cheerful voice_cartoon - condition_sad voice_robot	68.37160
condition_sad voice_cartoon - condition_cheerful voice_human	30.99652
condition_sad voice_cartoon - condition_sad voice_human	30.99652
condition_sad voice_cartoon - condition_cheerful voice_robot	5.00000
condition_sad voice_cartoon - condition_sad voice_robot	5.00000
condition_cheerful voice_human - condition_sad voice_human	5.00000
condition_cheerful voice_human - condition_cheerful voice_robot	30.99652
condition_cheerful voice_human - condition_sad voice_robot	30.99652
condition_sad voice_human - condition_cheerful voice_robot	30.99652
condition_sad voice_human - condition_sad voice_robot	30.99652
condition_cheerful voice_robot - condition_sad voice_robot	5.00000

The output also contains the `emmeans`-object on which these power calculations are based. By manipulating this object it is possible to tailor the power analyses to the contrasts desired for the planned study. That is, based on the dataset simulated with `ANOVA_exact()` we can write out the analysis code for `emmeans`-contrasts, just as we would if we were to analyse the empirical data, and use the output to perform the corresponding power analysis.

13.3 Customized `emmeans` contrasts

The `emmeans` reference grid and contrasts are included in the output of `ANOVA_exact()`. This can be accessed through `exact_result$emmeans`

By using `emmeans_power()` on the contrasts, we can reproduce the results of the previous power analysis for the pairwise comparisons.

```
knitr::kable(emmeans_power(exact_result$emmeans$contrasts)[1:2])
```

contrast	power
condition_cheerful_voice_cartoon - condition_sad_voice_cartoon	68.37160
condition_cheerful_voice_cartoon - condition_cheerful_voice_human	16.41134
condition_cheerful_voice_cartoon - condition_sad_voice_human	16.41134
condition_cheerful_voice_cartoon - condition_cheerful_voice_robot	68.37160
condition_cheerful_voice_cartoon - condition_sad_voice_robot	68.37160
condition_sad_voice_cartoon - condition_cheerful_voice_human	30.99652
condition_sad_voice_cartoon - condition_sad_voice_human	30.99652
condition_sad_voice_cartoon - condition_cheerful_voice_robot	5.00000
condition_sad_voice_cartoon - condition_sad_voice_robot	5.00000
condition_cheerful_voice_human - condition_sad_voice_human	5.00000
condition_cheerful_voice_human - condition_cheerful_voice_robot	30.99652
condition_cheerful_voice_human - condition_sad_voice_robot	30.99652
condition_sad_voice_human - condition_cheerful_voice_robot	30.99652
condition_sad_voice_human - condition_sad_voice_robot	30.99652
condition_cheerful_voice_robot - condition_sad_voice_robot	5.00000

Now, we can manipulate the `emmeans` reference grid to perform additional power analyses. In the following example, we calculate the power for the contrasts between sad and cheerful condition for each voice.

```
simple_condition_effects <- emmeans(
  exact_result$emmeans$emmeans,
  specs = ~ condition | voice
)

knitr::kable(emmeans_power(pairs(simple_condition_effects))[1:2])
```

contrast	voice
condition_cheerful - condition_sad	voice_cartoon
condition_cheerful - condition_sad	voice_human
condition_cheerful - condition_sad	voice_robot

We may also calculate the power for testing all condition means against an arbitrary constant. **Remember**, these are two-tailed tests, so it is not testing against the null that the individual cells in the design are more or less than 0.5, but rather different from 0.5 at all.

```
knitr::kable(emmeans_power(test(simple_condition_effects, null = 0.5))[1:2])
```

condition	voice
condition_cheerful	voice_cartoon
condition_sad	voice_cartoon
condition_cheerful	voice_human
condition_sad	voice_human
condition_cheerful	voice_robot
condition_sad	voice_robot

Finally, we can calculate the power for custom contrasts between any linear combination of conditions.

```
custom_contrast <- contrast(
  exact_result$emmeans$emmeans,
  list(robot_vs_sad_human = c(0, 0, 0, 1, -0.5, -0.5))
)

knitr::kable(emmeans_power(custom_contrast))
```

contrast	power	partial_eta_squared	cohen_f	non_centrality
robot_vs_sad_human	39.34971	0.0714286	0.2773501	3

Although `emmeans_power()` currently ignores adjustments for multiple comparisons, it is possible to calculate the power for Bonferroni-corrected tests by adjusting `alpha_level`.

```
n_contrasts <- nrow(as.data.frame(simple_condition_effects))

knitr::kable(emmeans_power(
  pairs(simple_condition_effects),
  alpha_level = 0.05 / n_contrasts
)[1:3])
```

contrast	voice	power
condition_cheerful - condition_sad	voice_cartoon	40.1555499
condition_cheerful - condition_sad	voice_human	0.8333333
condition_cheerful - condition_sad	voice_robot	0.8333333

Similarly, if we want to calculate power for a one-sided test, we can use the “hack” of doubling the `alpha_level` parameter.

```
knitr::kable(emmeans_power(
  pairs(simple_condition_effects)[1],
  alpha_level = 2 * 0.05
))
```

contrast	voice	power	partial_eta_squared	cohen_f
condition_cheerful - condition_sad	voice_cartoon	79.14507	0.1381215	0.4003204

Note, that because power is calculated from the squared *t*-value, power is only calculated correctly if the alternative hypothesis is true in the simulated dataset. That is, the difference of the condition means is *consistent with the tested directional hypothesis*.

13.4 Equivalence and non-superiority/-inferiority tests

Because `emmeans` can perform equivalence, non-superiority, and -inferiority tests, `emmeans_power()` can calculate the corresponding power for these tests.

```
knitr::kable(emmeans_power(
  pairs(simple_condition_effects, side = "equivalence", delta = 0.3)[2]
)[1:2])
```

	contrast	voice
2	condition_cheerful - condition_sad	voice_human

Note, that because power is calculated from the squared t -value, power is only calculated correctly if the alternative hypothesis is true in the simulated dataset. That is, the difference between the condition means is consistent with the tested directional hypothesis (smaller than `delta`).

13.5 Joint tests

Another useful application of `emmeans_power()` is to joint tests. Lets assume we plan to test the main effect of voice for each of the two conditions separately using `joint_tests()`. We can then calculate the power for each Bonferroni-corrected F -test as follows.

```
voice_by_condition <- joint_tests(
  exact_result$emmeans$emmeans,
  by = "condition"
)

knitr::kable(emmeans_power(voice_by_condition, alpha_level = 0.05 / 2)[1:3])
```

	model term	condition	power
1	voice	condition_cheerful	21.80102
3	voice	condition_sad	10.39583

13.6 Monte Carlo Simulations

Now, many of those reading may be anxious about wanting to apply corrections for multiple comparisons beyond a simple Bonferroni alpha reduction. This can be accomplished with the `ANOVA_power` function. The procedure is very similar to the one outlined in the pairwise comparisons section above. However, we

will need to set the the type of multiple comparisons adjustment we would like (`emm_p_adjust`) and the total number of simulations (`nsims`). Remember, the total number of simulations will determine the accuracy of the power estimation (higher `nsims` = higher accuracy). But, this comes at the cost of increased computation time. So, again, we can setup the same design and pass this onto the `ANOVA_power` function. In this particular case I have decided to set the `emm_p_adjust` to “holm” for a Holm-Bonferroni correction. This out of personal preference for this stepdown procedure since it preserves the alpha but is more powerful than the normal Bonferroni adjustment itself. However, I would like to warn users that some adjustments, such as Tukey or Scheffe, are not appropriate for designs with within-subject factors (i.e., the do not fully preserve the Type 1 error rate).

```
design_result <- ANOVA_design(
  design = "2w*3w",
  n = 40,
  mu = c(0.3, 0, 0.5, 0.3, 0, 0),
  sd = 2,
  r = 0.8,
  labelnames = c("condition", "cheerful", "sad", "voice", "human", "robot", "cartoon")
  plot = FALSE
)

monte_result1 <- ANOVA_power(
  design_result,
  alpha_level = 0.05,
  verbose = FALSE,
  emm = TRUE,
  contrast_type = "pairwise",
  emm_p_adjust = "holm",
  nsims = 10000, #set total number of simulations
  seed = 27042020, #set seed for reproducibility
)
```

We can now inspect the results contained within the `monte_result` object.

```
knitr::kable(monte_result1$emm_results[1:2])
```


	contrast	power
p_1	condition_cheerful,voice_cartoon - condition_sad,voice_cartoon	29.21
p_2	condition_cheerful,voice_cartoon - condition_cheerful,voice_human	2.77
p_3	condition_cheerful,voice_cartoon - condition_sad,voice_human	2.59
p_4	condition_cheerful,voice_cartoon - condition_cheerful,voice_robot	29.70
p_5	condition_cheerful,voice_cartoon - condition_sad,voice_robot	29.40
p_6	condition_sad,voice_cartoon - condition_cheerful,voice_human	7.58
p_7	condition_sad,voice_cartoon - condition_sad,voice_human	7.42
p_8	condition_sad,voice_cartoon - condition_cheerful,voice_robot	0.46
p_9	condition_sad,voice_cartoon - condition_sad,voice_robot	0.55
p_10	condition_cheerful,voice_human - condition_sad,voice_human	0.47
p_11	condition_cheerful,voice_human - condition_cheerful,voice_robot	7.43
p_12	condition_cheerful,voice_human - condition_sad,voice_robot	7.22
p_13	condition_sad,voice_human - condition_cheerful,voice_robot	7.67
p_14	condition_sad,voice_human - condition_sad,voice_robot	7.49
p_15	condition_cheerful,voice_robot - condition_sad,voice_robot	0.51

Chapter 14

Beyond Superpower I: Mixed Models with `simr`

While the ANOVA is a flexible tool, there are many hypotheses that cannot be tested with an ANOVA. One of the most common situations is the introduction of “random” effects (i.e., mixed models), or the distribution of the dependent variable may not be “normal” (e.g., it may be a binary, 0 or 1, response). For these cases the `simr` R package can be very useful (Green and MacLeod, 2016). This is a flexible power analysis tool that can be used for linear mixed models *and* for generalized linear mixed models. Unfortunately, the documentation for this package, in the way of informative vignettes for new users, is sorely lacking. Therefore, we have added some basic information on how to perform power analyses from scratch using this package.

14.1 How does `simr` work?

The `simr` package is quite elegant in its simplicity. Essentially, it builds a model using the `makeLmer` or `makeGlm` functions and then simulates data (using `lme4` under-the-hood) to estimate the power of the model given the specified parameters. As a user all you have to do is specify the data and the variance components. So, the tricky part is that a user *must* have a dataset for the `simr` package to work with. However, this can be randomly generated data.

For example, from the `simr` vignettes, we can imagine a study where a researcher has three levels for a random effect. For a basic scientist this could be different Petri dishes of cells/tissue, or, for a education researcher, this could be different schools. We then have a variable, “x”, and we want to see our power to detect this effect on outcome “y”. The data we need for `simr` could be generated with the following code.

```
library(simr)
# Some covariate that we have measured at 10 different levels
x <- rep(1:10)
# Three random grouping variables
# (e.g., Petri dishes)
g <- c('a', 'b', 'c')

# Comine into 1 dataset
X <- expand.grid(x=x, g=g)
```

Now we can define our model a little bit further by assigned the fixed and random effects. For this model we need to provide 2 slope coefficients, and the random intercept variance.

```
# fixed intercept and slope for the model
b <- c(2, -0.1)
# random intercept *variance*
V1 <- 0.5

# residual variance
s <- 1
```

We then need to build a mixed model using the `makeLmer` function. In this case we define a new outcome variable (`y`) then assign a fixed effect for `x`, and provide the random effect structure as random intercept only (`(1|g)`). The other model information we created above is also passed onto this function. We can print the model information to verify that the details are correct.

```
model1 <- makeLmer(y ~ x + (1|g),
                  fixef=b,
                  VarCorr=V1,
                  sigma=s,
                  data=X)
print(model1)

## Linear mixed model fit by REML ['lmerMod']
## Formula: y ~ x + (1 | g)
## Data: X
## REML criterion at convergence: 93.2416
## Random effects:
## Groups Name Std.Dev.
## g (Intercept) 0.7071
## Residual 1.0000
## Number of obs: 30, groups: g, 3
```

```
## Fixed Effects:
## (Intercept)          x
##           2.0         -0.1
```

Now we can pass this information onto the `powerSim` function for a power analysis. Please note that the total number of observations and data structure (3 levels for the random effect) will be the same in these simulations.

```
sim1 = powerSim(model1,
                 nsim=20,
                 progress=FALSE)
```

```
sim1
```

```
## Power for predictor 'x', (95% confidence interval):
##           20.00% ( 5.73, 43.66)
##
## Test: Kenward Roger (package pbkrtest)
##           Effect size for x is -0.10
##
## Based on 20 simulations, (0 warnings, 0 errors)
## alpha = 0.05, nrow = 30
##
## Time elapsed: 0 h 0 m 1 s
```

14.2 To Be Continued...

Chapter 15

Beyond Superpower II: Custom Simulations

Sometimes it may be better just to simulate your planned study. If your study has peculiar features that current software does not support custom simulation may be your only possible solution. This approach is flexible and you can customize it as much as you are willing to code. The downside is that the process will be more time consuming and it is possible to introduce errors in your code.

In this chapter, we will provide a few vignettes on how to implement simulation-based power analyses for a variety of outcomes and designs. Overall, all you need for these simulations is an accurate data generating process and the ability to repeat that process many times (preferably a few thousand iterations). This is the same process that **Superpower** performs under-the-hood when you are running simulations with **ANOVA_power**. The ability to write your own simulations and data generating mechanisms is a powerful tool that allows you to test your analyses, and assumptions about those analyses, prior to any data being collected.

15.1 A Clinical Trial with a Binary Outcome

This section was inspired by a Twitter thread from Andrew Althouse, a PhD statistician currently at the University of Pittsburgh Medical School, on creating a simulation of Randomized Clinical/Control Trial (RCT) data. We wanted to expand on his suggestions and provide some of my own recommendations. In Andrew's thread he explicitly used base R and only default **stats** functions that are native to R. While these are great tools, they are limited and most researchers wanting to simulate a study will find these tools lacking when trying

to simulate their own particular study. This is *not* a criticism of Dr. Althouse’s thread, it is a wonderful thread. However, in this section we will use a mostly “tidy” approach for generating the data.

In this section we will use `simstudy` as a way to pseudo-replicate the results from Dr. Althouse. This will serve as a simple introduce to simulating experimental designs and how this can be utilized for power analysis purposes. All the code here will be “functional” which means we will define new functions that will generate the data. This way you can run multiple variations on your simulation with only a few more lines of code (rather than copy and pasting the whole simulation again and again).

R Packages

For this simulation I will try to restrict my use of packages to as few as possible. To simulate the data again I will need the `simstudy` package and I will also `tidyverse` set of packages to make the code “tidy”. In addition, I will need the

```
library(simstudy)
library(tidyverse)
library(data.table)
```

15.1.1 Proposed Study and Approach

According to Dr. Althouse’s documentation he was trying to simulate the following study:

- This code will mimic a 2-group parallel-arm randomized trial using 1:1 allocation of patients to treatment 1 versus treatment 2
- For this example, we will use a binary outcome of “death”
- Patients receiving treatment 1 will have a 40% probability of death
- Patients receiving treatment 2 will have a 30% probability of death
- Analysis will be performed using a logistic regression model
- We will run 100 simulated RCT’s and report the odds ratio, 95% confidence interval, and p-value for each simulated trial
- The “true” treatment effect for a treatment that reduces probability of outcome from 40% to 30% is about $OR = 0.642$
- The power of a trial with $N=1000$ patients and exactly 1:1 allocation under these assumptions is about 91-92%

Some people reading this may already feel a tad overwhelmed. But trust me, everyone has to start somewhere and all the authors of this book were in your shoes not too long ago! Also, this is a fairly straightforward process. The process is only made easier when you use the many tools available in R. We are going to create “functions” which essentially means if I want to run the same

simulation again (but maybe change 1 or 2 parameters) this can be done with only a few lines of code. This is efficient and if you are serious about writing your own simulations I would highly recommend writing your own functions for the simulation.

15.1.2 Step 1: Create a Data Generating Function

This part looks more complex than it is. All we are doing is making a function that I will call `gen_bidat` (short for “generate binomial data”). One of the things Althouse mentions in his code is that his allocation in the simulation is inappropriate. We can get around that by incorporating the functions from `simstudy` (which randomly assigns group) and keeps things organized through the `%>%` (known as a “pipe”) function. We also will import the `data.table` package because the `simstudy` package generates data in a `data.table` format.

```
# Define function
# Parameters
## N = sample size
## props = proportions in each group (length is equal to number of groups)
gen_bidat = function(props,N){
  # Create data with N participants and balanced assignment to treatment group
  df = genData(N) %>% # generate data with N participants
    trtAssign(n = length(props),
              balanced = TRUE,
              grpName = "trt") # Randomly assign treatment (tr) in a balanced fashion
  # Get the number of groups by the number or proportions entered
  grps = length(props)
  # generate condition for each group
  # This creates a conditional output
  # I.e., the odds of the outcome is determined by treatment group
  for(i in 1:grps){ # run loop once for each group in the study
    grp = i-1 # the simulation runs from 1 to the total number of groups
    # the i-1 starts the loop at zero
    # We then assign the group (grp) by which number in the loop we are in
    con_run = paste0("trt == ", grp)
    # We then grab the assign the correct proportion to the groups in order
    # We have to create the object dc first (i == 1)
    # All iterations of the loop add to dc rather than creating a new dc
    if (i == 1) {
      dc = defCondition(
        condition = con_run,
        formula = props[i],
        dist = "binary",
        link = "identity"
      )
    }
  }
}
```

```

    } else{
      dc = defCondition(
        dc,
        condition = con_run,
        formula = props[i],
        dist = "binary",
        link = "identity"
      )
    }

  }

  # Now we generate the outcome based on the group (condition)
  dat <- addCondition(condDefs = dc,
                     dtOld = df,
                     newvar = "y")

  return(dat)
}

# Test run
gen_bidat(c(.2,.35),N=10)

```

```

##      id y trt
##  1:  1 0   0
##  2:  2 0   1
##  3:  3 0   1
##  4:  4 0   0
##  5:  5 1   0
##  6:  6 0   0
##  7:  7 1   1
##  8:  8 0   1
##  9:  9 1   0
## 10: 10 0   1

```

In some cases the function above is enough. We may just want to generate a data set reflective of study we have designed. We can then “play” with the data set to plan analyses for future study. However, that is not what we are after today and we will move onto the power analysis.

15.1.3 Step 2: Simulation

Now we can get to the fun part and run a simulation. All this means is that we run the simulated data (above) for a number of iterations or repetitions (typically for thousands of iterations). We can make it a power analysis by counting the number of positive results (e.g., below the significance threshold).

For the power analysis we created a `pwr_bidat` function that performs a power analysis with a certain number of repetitions (`reps` argument). Notice below that we are using the function we just created (`gen_bidat`) within the `pwr_bidat` function. That is why the user must supply the same information as the last function (the proportions, `props`, and the sample size, `N`). In addition, there are two arguments needed for the power analysis: `alpha` and `conf.level`. The `alpha` argument sets the alpha-level for the analyses (i.e., the significance cutoff). While the `conf.level` argument sets the confidence level (e.g., 95%) for the confidence intervals for the power analysis. We can calculate confidence intervals for a simulation because we have a number of “successes” over a number of attempts (total number of `reps`). We can use the `prop.test` function which provides confidence intervals for proportions. This is helpful when for when we are running a small number of simulations and want an idea of what estimates of power are reasonable.

```
pwr_bidat = function(props,N,
                     reps=100,
                     alpha=.05,
                     conf.level = .95){

  # Create 1 to reps simulated data sets
  sims = replicate(n = reps,
                  gen_bidat(props, N = N),
                  simplify = FALSE)
  # Run an equivalent number of analyses
  sims2 = purrr::map(sims, ~ glm(y ~ trt,
                                data = .x,
                                family = binomial(link = "logit")))
  # Get the summary coefficients from our models
  sims3 = purrr::map(sims2, ~ summary(.x)$coefficients)
  # Put all the results into a data frame (tibble)
  sims4 = purrr::map(sims3,
                    ~ tibble::rownames_to_column(as.data.frame(.x),
                                                  "coef"))

  # Combine all the data frames into one
  simsdf = bind_rows(sims4, .id = "nrep")
  # Summarize results by coefficient
  simspow = simsdf %>%
    group_by(coef) %>%
    # Calculate the power (number of results with p-value < alpha)
    summarize(
      estimate = mean(Estimate),
      power = mean(`Pr(>|z|)` < alpha),
      # Calculate confidence intervals
      power.lci = prop.test(sum(`Pr(>|z|)` < alpha), reps)$conf.int[1],
      power.uci = prop.test(sum(`Pr(>|z|)` < alpha), reps)$conf.int[2],
```

Table 15.1: Result from Power Simulation

Coefficients	Average Log Odds	Power	Lower C.I.	Upper C.I.
(Intercept)	-0.43	1.0	0.95	1.00
trt	-0.43	0.9	0.82	0.95

```

    .groups = 'drop'
  )

  # Return table of results
  return(simpow)
}

set.seed(01292020)
pwr_res = pwr_bdat(c(.4, .3), N=1000)

```

Now that we have the results we can create a table as output using the `kable` function.

```

# Create pretty table to print results
knitr::kable(pwr_res %>%
  rename(Coefficients = coef,
         `Average Log Odds` = estimate,
         Power = power,
         `Lower C.I.` = power.lci,
         `Upper C.I.` = power.uci),
  digits = 2,
  caption = "Result from Power Simulation")

```

Based on these results we can conclude that a study of 1000 patients randomly assigned to one of two treatment groups wherein 30% of participants die in treatment group #1 and 40% perish in treatment group #2 will have approximately 95% power [0.82,0.95]. Notice, that compared to Dr. Althouse's thread, we estimated the power at 95% (thread noted power at ~92.3%). This is to be expected when simulating data and is why a high number of repetitions are needed to get an accurate estimate of power.

15.1.4 Why more complicated code?

Well, here is why we create functions: it is easier on yourself in the future. Say, we run three separate simulations with minor differences so I go about with the Althouse approach (maybe even copy and paste the code a few times). Later, I notice a flaw in my code, or maybe there is a small change that alters all three

Table 15.2: Another Study of Binary Outcomes

coef	estimate	power	power.lci	power.uci
(Intercept)	-3.3449342	1.00	0.9538987	1.0000000
trt	-0.6826144	0.68	0.5782080	0.7677615

simulations. Well, if I take the time to create the customized functions then all I have to do is change the code in one place (where I defined the function) rather than with every chunk of code that includes the simulation code.

Also, it is easier to produce variations on the same design. Let's imagine we are doing a study on a treatment for an infectious disease that has a hospitalization rate of at least 3.5% people that get infected (so treatment occurs immediately upon diagnosis and only the most severe are hospitalized). The study investigators want to know if 2000 patients are enough to detect if the proposed treatment reduces the hospitalization rate at least by half (1.75%).

All I have to do is use the same function I have created above but change the arguments in the function.

```
pwr_2 = pwr_bdat(c(.035,.0175),N=2000)

knitr::kable(pwr_2,
              caption = "Another Study of Binary Outcomes")
```

Now, we have the new results in only 4 lines of code! Based on those results we would likely advise the investigators that they will need a larger sample size to conclude effectiveness for their proposed treatment.

15.1.5 Conclusions on the Binary RCT Analysis

In this section we have demonstrated how to create 2 R functions that 1) generate data and 2) generate a simulation based power analysis. Simulation is not necessary for a simple analysis such as this where analytic solutions exist in programs like **GPower** or **Superpower**. However, as we will detail in the next sections, simulation becomes *very* useful when the design becomes complicated (or when we want to violate the assumptions of the models we use).

15.2 Binary RCT with a Interim Analysis

Now we can move onto a more complicated, but very useful, RCT design that involves analyzing the data at multiple sample sizes. The process of looking at the data early (i.e., before the full sample size is collected) is called a “interim”

analysis, but is sometimes called a “sequential” analysis as well (Lakens, 2014; Lakens et al., 2021). Regardless of what the process is called the goal is the same: more efficient study designs. There is an obvious advantage of including interim analyses is the ability to stop a study early. Therefore the sample size is lower and you can save resources. Why don’t we do this for every study? Well, adjusting for the multiple looks at the data can be difficult, and if we do not adjust for the multiple possible stopping points the Type I error rate will be inflated (e.g., an alpha of 0.05 can be inflated to ~0.35 with unlimited looks at the data).

15.2.1 Proposed Study and Approach

The approach mostly remains the same as the previous example but we have to account for a interim analysis.

- This code will mimic a 2-group parallel-arm randomized trial using 1:1 allocation of patients to treatment 1 versus treatment 2 with a maximum sample size of 1000 patients.
- For this example, we will use a binary outcome of “death”
- Patients receiving treatment 1 will have a 40% probability of death
- Patients receiving treatment 2 will have a 30% probability of death
- Analysis will be performed using a logistic regression model
- We will run 100 simulated RCT’s and report the odds ratio, 95% confidence interval, and p-value for each simulated trial
- The “true” treatment effect for a treatment that reduces probability of outcome from 40% to 30% is about $OR = 0.642$
- There will be a total of 3 analyses at 50%, 75%, and 100% of the sample size.
- The alpha level will be adjusted with a O’Brien-Fleming correction (alpha spending function).

Now, for this simulation we will need 1 more R package: ‘rpact’. This is a *very* useful package if you are interested in sequential analyses.

To properly plan the study we have to define the number of looks at the data (`nLooks`), when these analyses would be scheduled as proportion of the total sample size (`analyses_scheduled`), and we can create an empty vector for the what the adjusted alpha should be (`efficacy_thresholds`).

```
library(rpact)
# here is where you put the maximum number of analyses
nLooks <- 3
# Here is where you list the information fraction (% of total N)
## (e.g. here 50%, 75% and 100% information)
```

```

analyses_scheduled <- (c(0.50, 0.75, 1))
efficacy_thresholds <- numeric(nLooks)

```

Now, we setup the analysis procedure using the `getDesignGroupSequential` function from `rpact`. Notice that we set the if the hypothesis test is one or two sided (`sided = 2` for a two sided hypothesis in this case), and the overall alpha level is set with `alpha` argument. We then supply when the analyses will take place with the `informationRates` argument, and the type of correction to the alpha level for each interim analysis is set with `typeOfDesign`. Since we want a O'Brien-Fleming correction we set it to `typeOfDesign = "asOF"`.

Once the design is created we can extract it with a simple for loop for the number of analyses to extract the alpha level for each interim analysis.

```

# Set up design in rpact
design <- getDesignGroupSequential(sided = 2,
                                  alpha = 0.05,
                                  informationRates = analyses_scheduled,
                                  typeOfDesign = "asOF")

for(j in 1:nLooks){
  efficacy_thresholds[j] = design$stageLevels[j]
}

```

Now we can create our own function again. We'll call it `pwr_bidat2`. Essentially this does the same thing as the function in the last example. However, it takes three looks at the data when there are 500, 750, and 1000 patients. So a for loop will split up the sample and add interim analyses at these points. The loop will stop for each repetition of the simulation when either the observed p -value is less than the corrected alpha or when the total sample size is reached. Those results are then counted and save in the `res_tab` portion of the function's output.

```

pwr_bidat2 = function(props, N,
                      interim_n = c(500,750,1000),
                      reps = 100,
                      alpha = efficacy_thresholds,
                      conf.level = .95){
  #Create empty data frame for the results of each repetition
  res = data.frame(N = rep(NA,reps),
                   alpha = rep(NA,reps),
                   success = rep(NA,reps),
                   stop = rep(NA,reps),
                   p.value = rep(NA,reps),
                   estimate = rep(NA,reps))

  # For loop for each repetition of the simulation

```

```

for(i in 1:reps){
  # Generate full data set
  dat = gen_bidat(props, N = N)
  # Set logical indicator for stopping the trial
  stop_trial = FALSE
  # Run loop for each potential stopping point
  for(j in 1:length(interim_n)){
    # Set current N and alpha level
    val = interim_n[j]
    alpha_j = alpha[j]
    # Only continue analysis if stopping criteria not met
    if(stop_trial != TRUE){
      # Get interim data points
      dat_i = dat[1:val,]
      # Build model
      mod_i = glm(y ~ trt,
                  data = dat_i,
                  family = binomial(link = "logit"))
      # Get results from the model
      sum_i = summary(mod_i)$coefficients[2,]

      # Determine if the trial can stop
      if(sum_i[4] < alpha_j){
        stop_trial = TRUE
        alpha_f = alpha_j
        n_f = val
      }
    }
  }

  # Get final alpha and N if no "success" at final N
  if(stop_trial == FALSE){
    n_f = val
    alpha_f = alpha_j
  }

  # Store results from this repetition
  res$N[i] = n_f
  res$alpha[i] = alpha_f
  res$success[i] = stop_trial
  res$stop[i] = ifelse(stop_trial, n_f, "fail")
  res$p.value[i] = sum_i[4]
  res$estimate[i] = sum_i[1]
}

# Create summary of results
simspow = res %>%

```


Table 15.3: Power for Interim Analysis Design

med_est	power	power.lci	power.uci
0.4617294	0.85	0.7614692	0.9108509

```

summarize(
  med_est = median(estimate),
  power = mean(p.value < alpha),
  # Calculate confidence intervals
  power.lci = prop.test(sum(p.value < alpha), reps)$conf.int[1],
  power.uci = prop.test(sum(p.value < alpha), reps)$conf.int[2]
)

stoppow = res %>%
  group_by(stop) %>%
  summarize(
    prop = n()/reps,
    prop.lci = prop.test(n(), reps)$conf.int[1],
    prop.uci = prop.test(n(), reps)$conf.int[2],
    mean.est = mean(estimate),
    .groups = 'drop'
  )

return(list(sims = res,
            power_tab = simspow,
            stop_tab = stoppow))
}

```

Now, we can actually perform the simulation and with the same parameters from the last example.

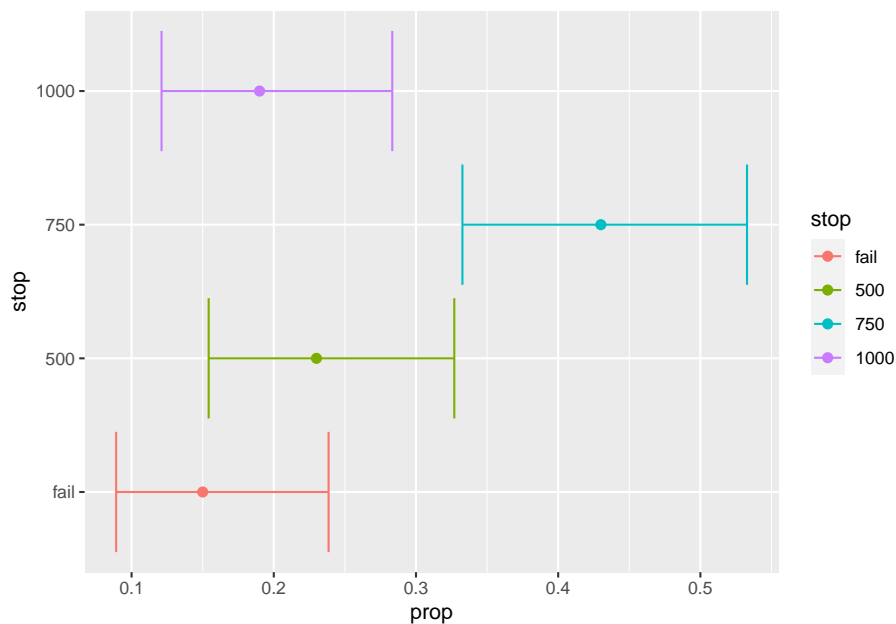
```

set.seed(20210324)
test = pwr_bidat2(props = c(.3, .4),
                  N = 1000)
knitr::kable(test$power_tab,
              caption = "Power for Interim Analysis Design")

```

As we can see from the summary table above, the overall statistical power is slightly reduced (at least with 100 simulations). Below, we can plot the proportion of results that are below the corrected alpha level at 1000, 750, and 500 patients (as well as the proportion where there was a failure to find a significant result).

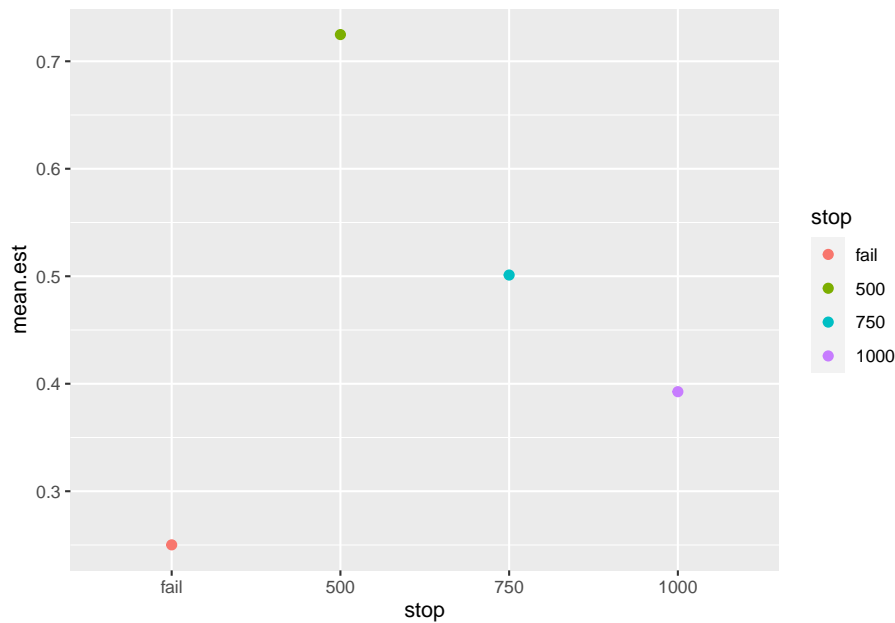
```
test$stop_tab %>%
  mutate(stop = factor(stop,
                        levels = c("fail", "500", "750", "1000"))) %>%
  ggplot(aes(x = stop, color = stop, y = prop)) +
  geom_point(size=2) +
  geom_errorbar(aes(ymin = prop.lci,
                   ymax = prop.uci)) +
  coord_flip()
```



So how often can we expect the trial to end early? Well, according to this simulation we can expect the trial, under these conditions and stopping criteria, to end at $N = 500$ approximately 30% of the time and $N = 750$ about 35% of the time. So we have a very high chance of being able to save resources by using interim analyses. Note, that the confidence intervals are awfully wide from these simulations so we want to be re-run the simulation with a higher number of repetitions.

Another thing to point out is the effect size estimate. We can also plot the effect of treatment at every possible stopping point of the study.

```
test$stop_tab %>%
  mutate(stop = factor(stop,
                        levels = c("fail", "500", "750", "1000"))) %>%
  ggplot(aes(x = stop, color = stop, y = mean.est)) +
  geom_point(size = 2)
```



We can see that the effect size estimate, log-odds, are inflated when the study is stopped early (roughly double the average effect size at $N=500$). This is one of the disadvantages of early stopping: inflated effect size estimates. What is the reason for this peculiar pattern? In order to obtain a significant result at an early stopping point the observed effect size must be quite large. Therefore, interim analyses are selecting for very large effect sizes in order to stop the study early.

15.3 Conclusion on a Binary RCT with Interim Analyses

As we have documented, planning for interim analyses can be implemented with a few modifications to our existing code from the previous example. For this example we switched to a for loop to make it easier to perform the interim analyses and record the stopping point (and the relevant statistics at the stopping point). The appropriate stopping criteria (i.e., the adjusted alpha level) can be obtained from specialized R packages like `rpact`.

Appendix 1: Direct Comparison to pwr2ppl

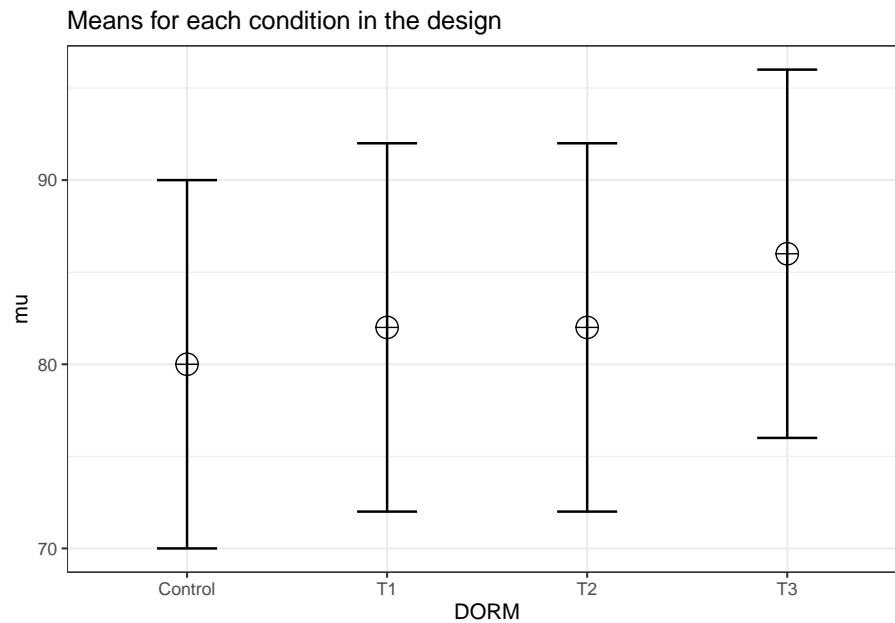
In this appendix we have included the performance of **Superpower** to the **pwr2ppl** package using Chris Aberson's examples in *Applied Power Analysis for the Behavioral Sciences* (2nd edition).

15.4 Examples from Chapter 5

15.4.1 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.

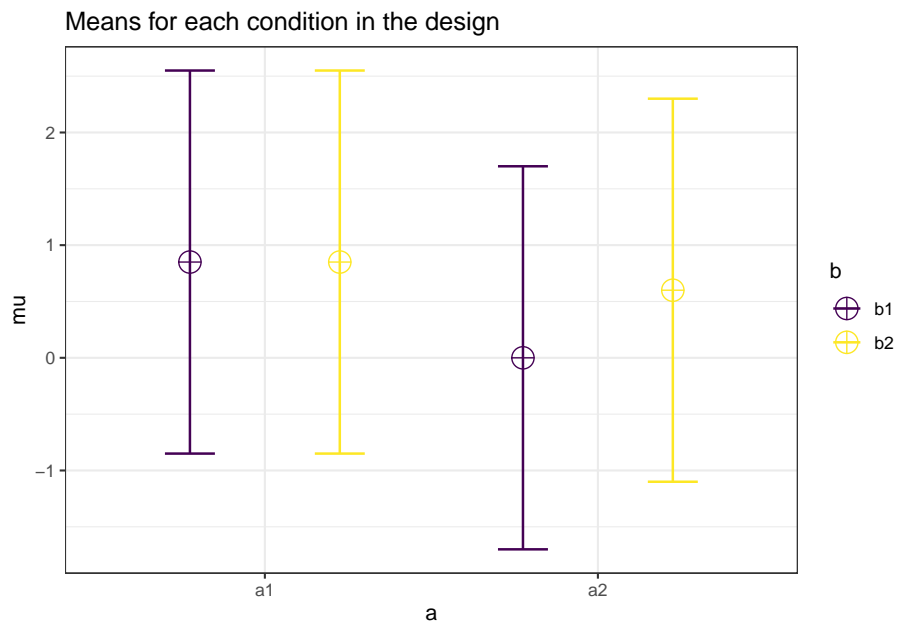
```
anova1f_4(m1 = 80, m2 = 82, m3 = 82, m4 = 86,
          s1 = 10, s2 = 10, s3 = 10, s4 = 10,
          n1 = 60, n2 = 60, n3 = 60, n4 = 60)
```

```
## Power = 0.812 for eta-squared = 0.05
```

15.4.2 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)
```

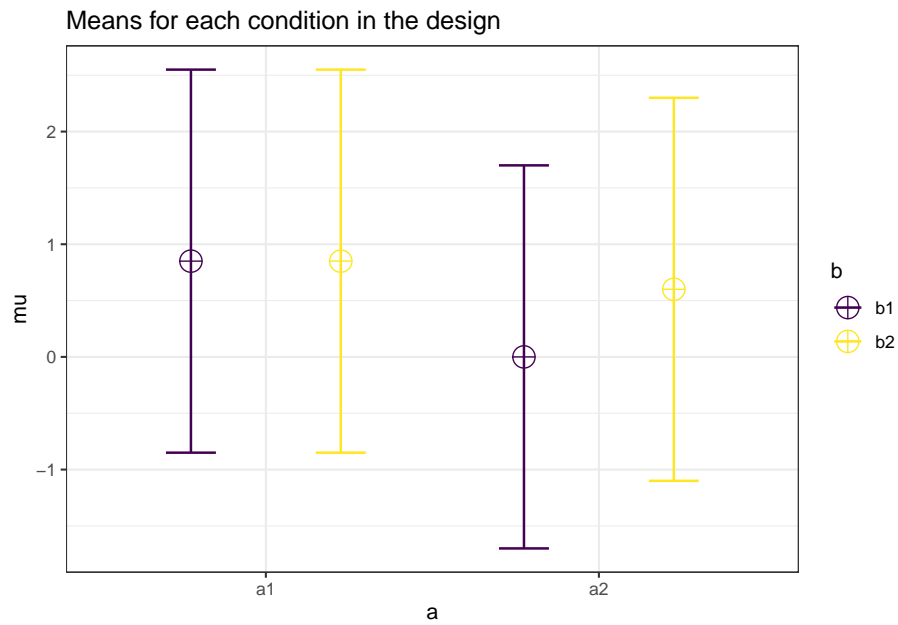
```
## Power for Main Effect Factor A = 0.898
```

```
## 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
```

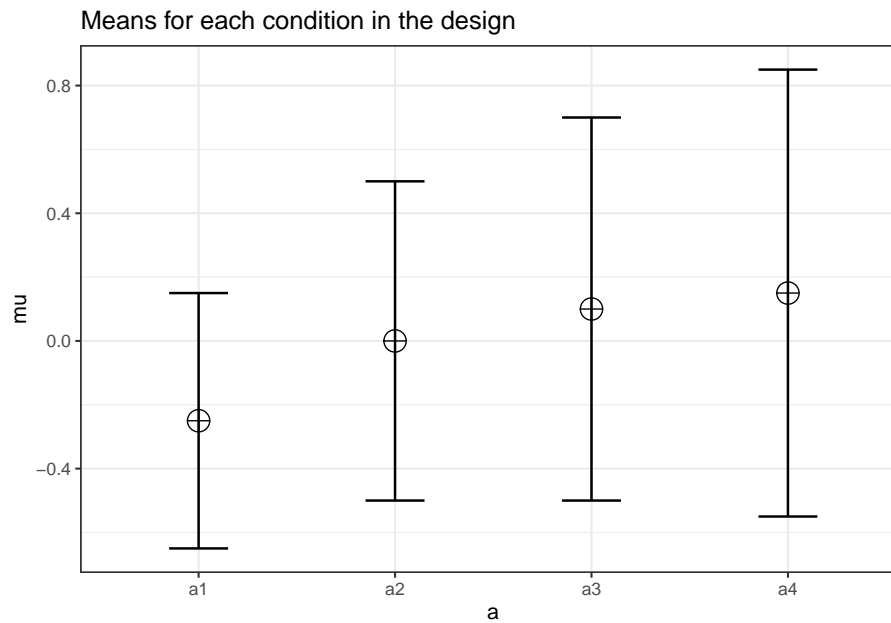
```
## Power for Interaction AxB = 0.796
```

15.5 Examples from Chapter 6

Repeated measures ANOVAs

15.5.1 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 produce 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
```

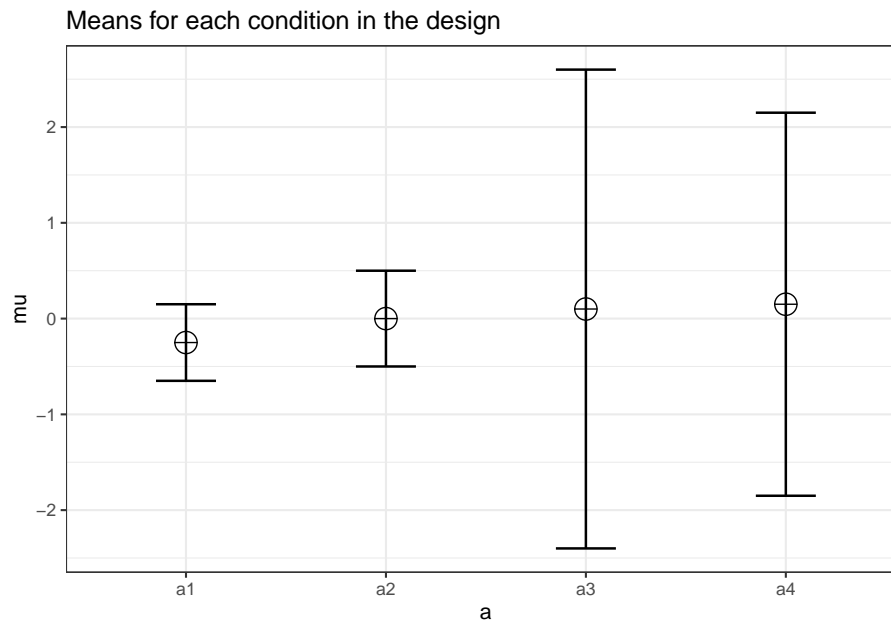
```
## Power (Unadjusted) for n = 25 is 0.809
```

```
## 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
```

15.5.2 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 produce 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
```

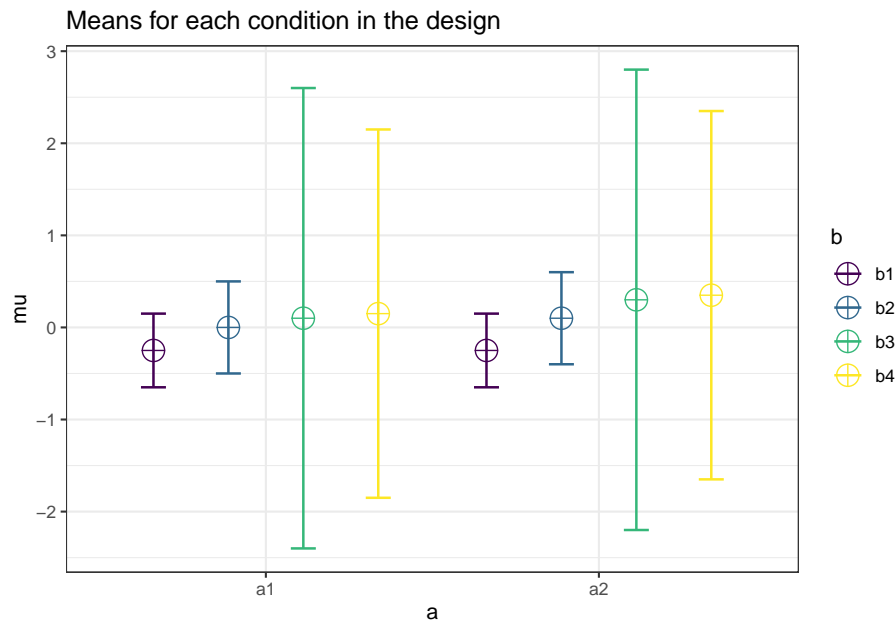
```
## Power (Unadjusted) for n = 100 is 0.397
```

```
## 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
```

15.5.3 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 produce 3 **ANOVA_exact** results representing no sphericity correction, Greenhouse-Geisser, and Huynh-Feldt corrected results.

```
#In comparison 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
```

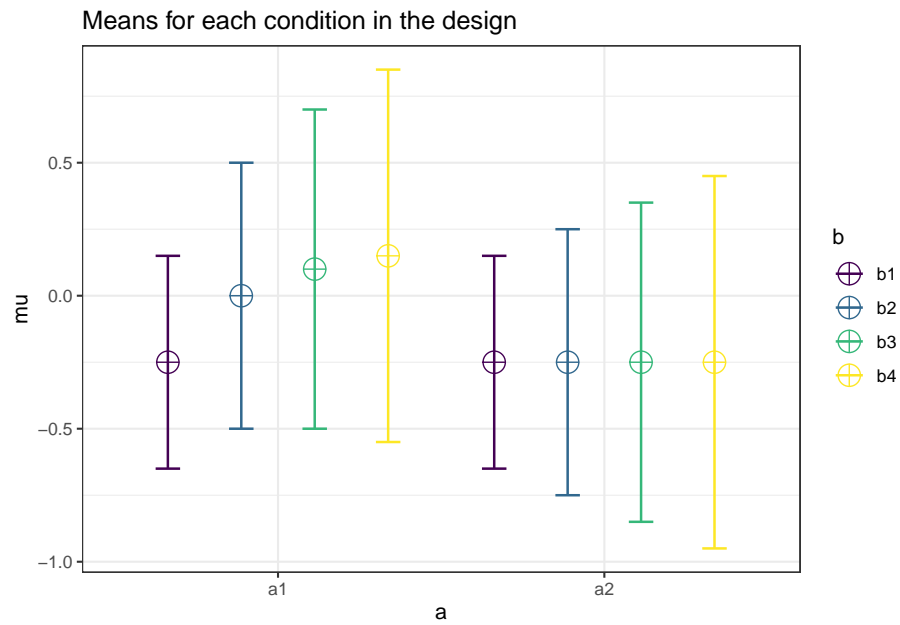
15.6 Example from Chapter 7

Mixed effect ANOVA

15.6.1 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

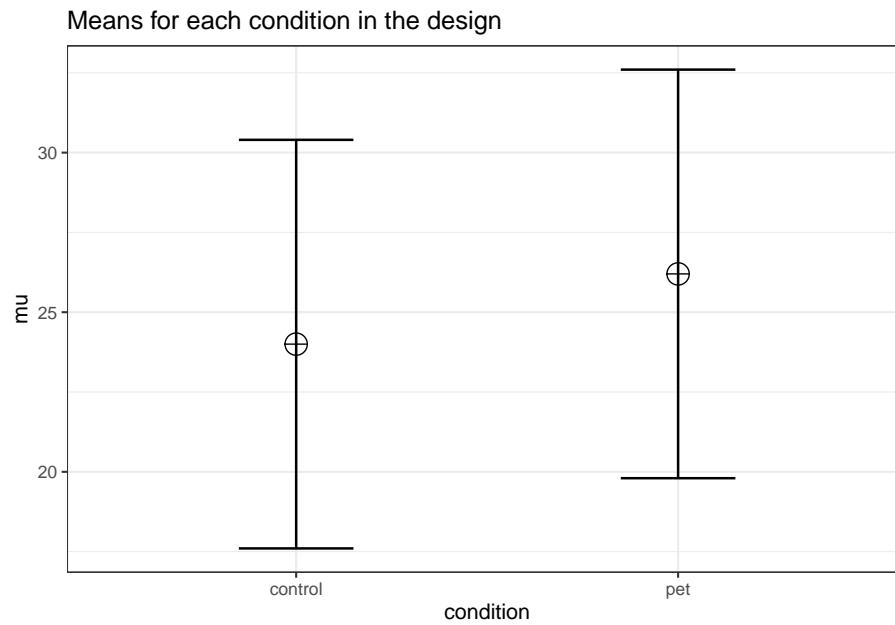
Appendix 2: Direct Comparison to MOREpower

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
```

MorePower 6.0.4

Analysis

☒ ANOVA ☐ t-test of means

☐ 1 sample ☐ 2 sample

z-test of proport.

☐ 1 sample ☐ 2 sample

Design Factors

RM

IM

Effect of Interest

RM

IM

Alpha 2-sides

☐ ☒

Sample

Power

Solve For

☒ Power ☐ Effect Size ☐ Sample Size

Effect Size

☒ η^2

☐ F

Variability

☒ S

☐ MSE

Solve

power = ,67685724, sample = 200
 partial η^2 = ,0289748
 Cohen's f = ,173

dBIC=-,582, BF01=,747, BF10=1,338
 p(H0|D) = ,42772744, p(H1|D) = ,57227256

J&H 95% CI \pm ,892, t(crit) = 1,972, df = 198
 mean difference = 1,555635
 std. err. of difference = 0,640
 95% CI of difference \pm 1,262

[F(1,198) = 5,908, p = ,01596, MSE = 20,48,
 part η^2 = ,029, BF01=,74742]

ANOVA Examples Clear Values Clear Output Clear Session Program Information

Session Calculations = 17

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

```

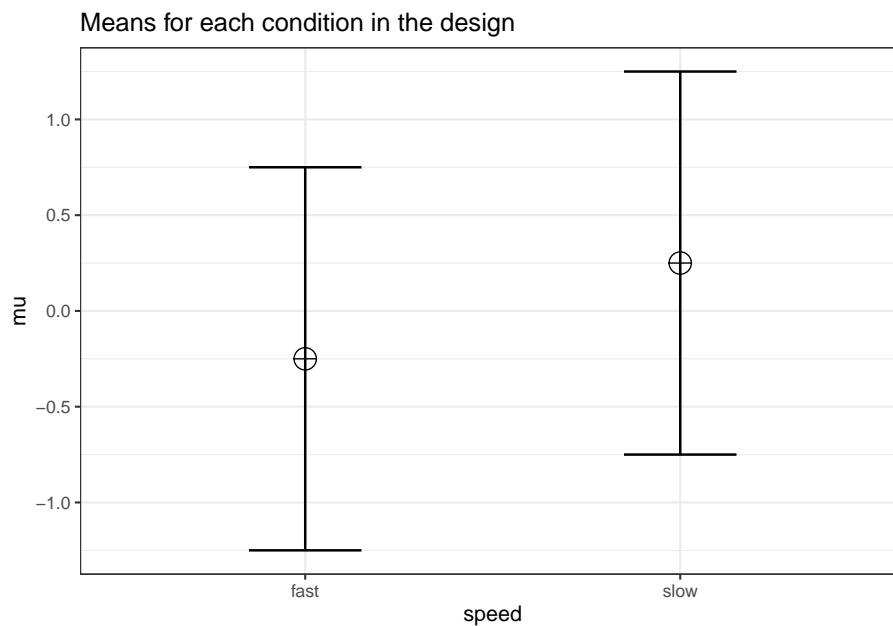
```
## [1] 0.05882353
```

```

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
```

MorePower 6.0.4

Analysis
☒ ANOVA ☐ t-test of means
☐ 1 sample ☐ 2 sample
 z-test of proport.
☐ 1 sample ☐ 2 sample

Design Factors
 RM 2
 IM

Effect of Interest
 RM 2
 IM

Alpha 2-sides .05 ☐ ☒
Sample 34 **Power** .80777

Solve For
☒ Power ☐ Effect Size ☐ Sample Size

Effect Size
☒ η^2 .2048193
☐ F 8,500001

Variability
☒ S 1
☐ MSE .5

Solve

power = .80777756, sample = 34
 partial η^2 = .2048193
 Cohen's f = .508

dBIC=-4,266, BF01=.118, BF10=8,440
 p(H0|D) = .10593243, p(H1|D) = .89406757

J&H 95% CI \pm .247, t(crit) = 2,035, df = 33
 mean difference = 0,500
 std. err. of difference = 0,171499
 95% CI of difference \pm 0,349

[F(1,33) = 8,500, p = .00634, MSE = .5, part η^2 = .2048, BF01=.11848]

ANOVA Examples Clear Values Clear Output Clear Session Program Information

Session Calculations = 20

We can also replicate the 3-group within ANOVA example in MOREpower.

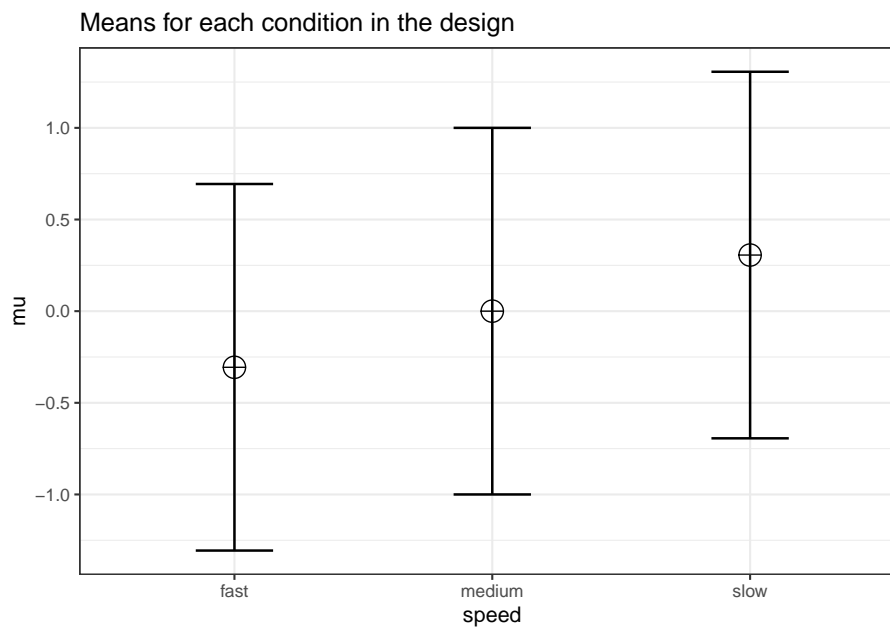
```
K <- 3
n <- 20
sd <- 1
r <- 0.8
alpha = 0.05
```

```
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
```

MorePower 6.0.4

Analysis

☒ ANOVA ☐ t-test of means

☐ 1 sample ☐ 2 sample

z-test of proport.

☐ 1 sample ☐ 2 sample

Design Factors

RM 3

IM

Effect of Interest

RM 3

IM

Alpha 2-sides

0.05

Sample

20

Power

.96916

Solve For

☒ Power ☐ Effect Size ☐ Sample Size

Effect Size

☒ η^2 0.3303965

☐ F 9.375001027

Variability

☐ S NA

☒ MSE 0.2

Solve

power = .96916338, sample = 20
 partial η^2 = .3303965
 Cohen's f = .702

dBIC=-8.665, BF01=.013, BF10=76.135
 p(H0|D) = .01296424, p(H1|D) = .98703576

J&H 95% CI \pm .64, t(crit) = 2.024, df = 38

[F(2,38) = 9.375, p = .00049, MSE = 2., part η^2 = .3304, BF01=.01313]

n/group for the effect = 20
 MST = 18.750002, critical F = 3.245

ANOVA Examples Clear Values Clear Output Clear Session Program Information

Session Calculations = 27

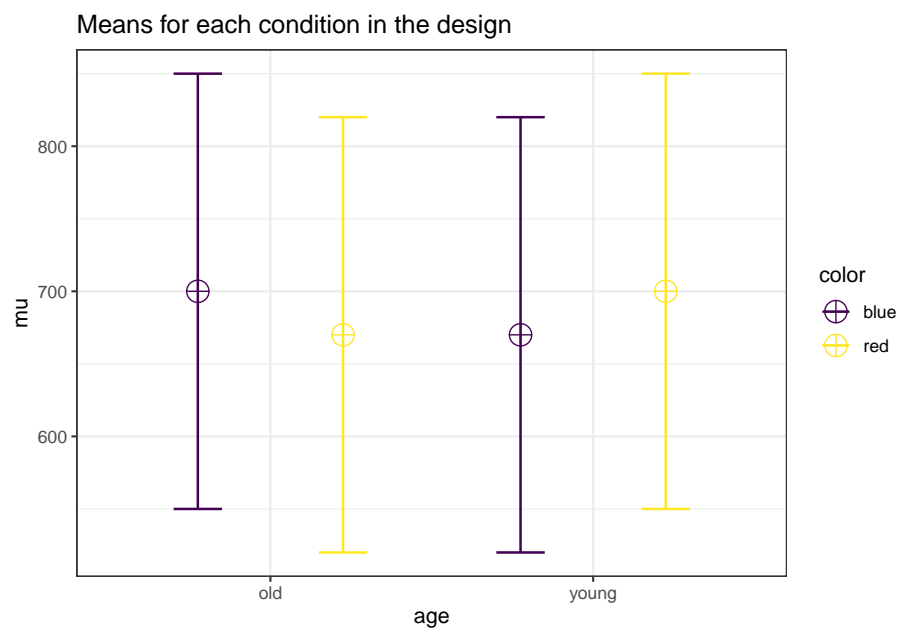
We can reproduce the 2x2 ANOVA example in MOREpower.

```
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.

MorePower 6.0.4

Analysis
☒ ANOVA ☐ t-test of means
☐ 1 sample ☐ 2 sample
 z-test of proport.
☐ 1 sample ☐ 2 sample

Design Factors
 RM
 IM

Effect of Interest
 RM
 IM

Alpha 2-sides ☐ ☒
Sample **Power**

Solve For
☒ Power ☐ Effect Size ☐ Sample Size

Effect Size
☒ η^2
☐ F

Variability
☒ S
☐ MSE

Solve

```

power = ,48401815, sample = 25
partial eta² = ,1428571
Cohen's f = ,408

dBIC=-,635, BF01=,728, BF10=1,374
p(H0|D) = ,42129856, p(H1|D) = ,57870144

J&H 95% CI ±30,958, t(crit) = 2,064, df = 24
mean difference = 59,99989
std. err. of difference = 30,000
95% CI of difference ±61,917

[F(1,24) = 4,000, p = ,05694, MSE = 5625,,
part eta² = ,1429, BF01=,72801]
  
```

ANOVA Examples Clear Values Clear Output Clear Session Program Information

Session Calculations = 34

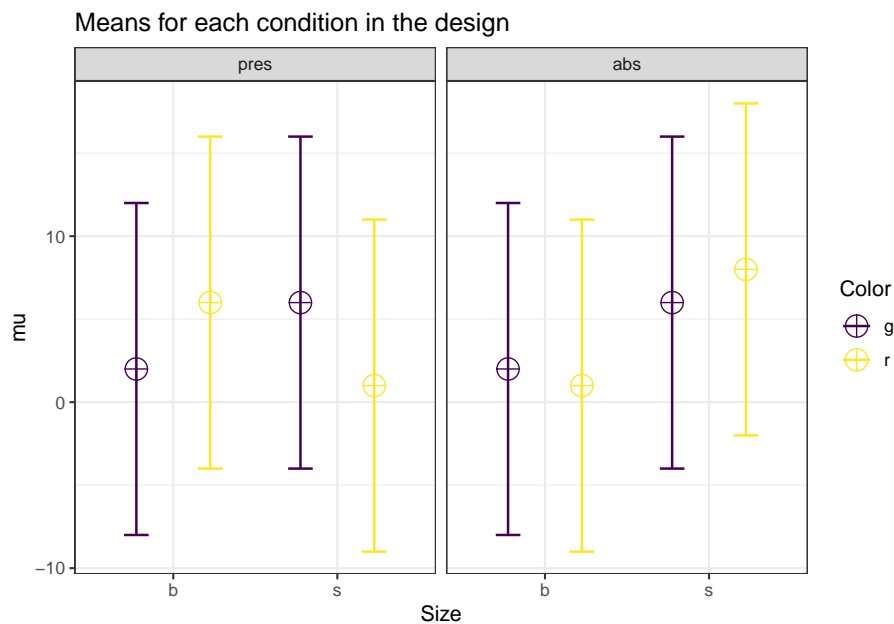
Let's replicate the 2x2x2 full between analysis (total N = 400, effects sizes differ for each test):

```

# 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

```

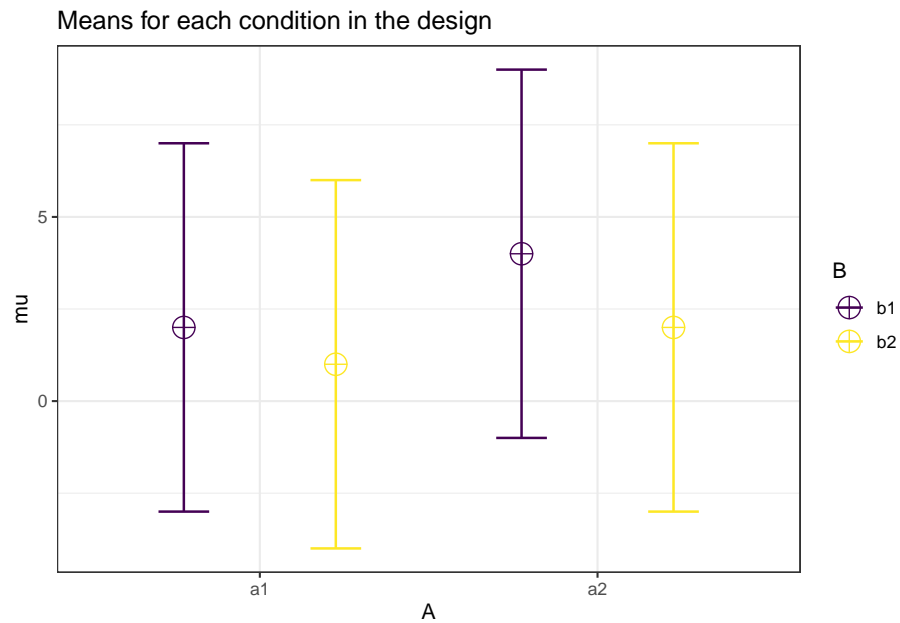
```
exact_result$main_result$partial_eta_squared
```

```

## [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.



```
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.

The image displays two side-by-side screenshots of the MorePower 6.0.4 software interface, showing the ANOVA power calculation window. Both windows are set to 'ANOVA' analysis, 'RM' design factors, and 'RM' effect of interest. The 'Alpha' is set to .05, '2-sides' is selected, 'Sample' size is 20, and 'Power' is .2691752 (left) and .6422587 (right). The 'Solve For' option is 'Power'. The 'Effect Size' is set to 'eta^2' with values .0952381 (left) and .24 (right). The 'Variability' is set to 'S' with values 2.236068 (left) and 4.472136 (right). The 'MSE' is set to 0.5 in both. The 'Solve' button is highlighted in both. The output text at the bottom of each window shows the calculated power, sample size, and other statistical values.

Left Window (Session Calculations = 49):

- power = .26917526, sample = 20
- partial eta² = .0952381
- Cohen's f = .324
- dBIC=,994, BF01=1,644, BF10=,608
- p(H0|D) = .62176147, p(H1|D) = .37823853
- J&H 95% CI ±,74, t(crit) = 2,093, df = 19
- mean difference = 0,707107
- std. err. of difference = 0,500
- 95% CI of difference ±1,047
- [F(1,19) = 2,000, p = .17348, MSE = 5,, part eta² = .0952, BF01=1,64383]

Right Window (Session Calculations = 48):

- power = .64225865, sample = 20
- partial eta² = .24
- Cohen's f = .562
- dBIC=-2,493, BF01=,288, BF10=3,478
- p(H0|D) = .22330619, p(H1|D) = .77669381
- J&H 95% CI ±1,047, t(crit) = 2,093, df = 19
- mean difference = 2,44949
- std. err. of difference = 1,000
- 95% CI of difference ±2,093
- [F(1,19) = 6,000, p = .02417, MSE = 5,, part eta² = .24, BF01=,28751]

Appendix 3: Comparison to Brysbaert

15.7 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)

```

15.7.1 Three conditions replication

```

K <- 3
mu <- c(0, 0.4, 0.4)
n <- 90
sd <- 1

```

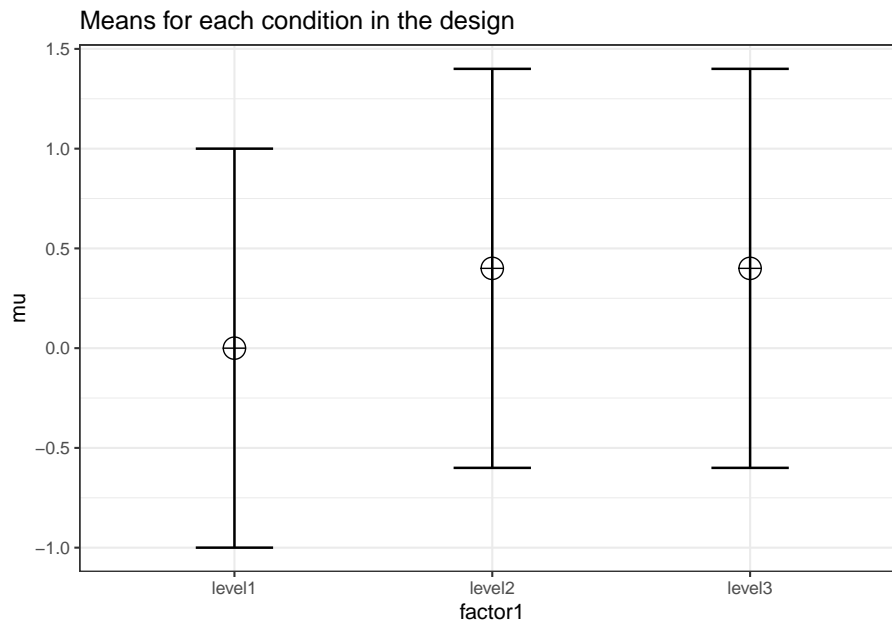


```

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 15.4: Simulated ANOVA Result

	power	effect_size
anova_factor1	79.6	0.0417248

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

Table 15.5: Exact ANOVA Result

	power	partial_eta_squared	cohen_f	non_centralilty
factor1	79.37239	0.0347072	0.1896182	9.6

15.7.2 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)

```

15.7.3 Three conditions replication

```

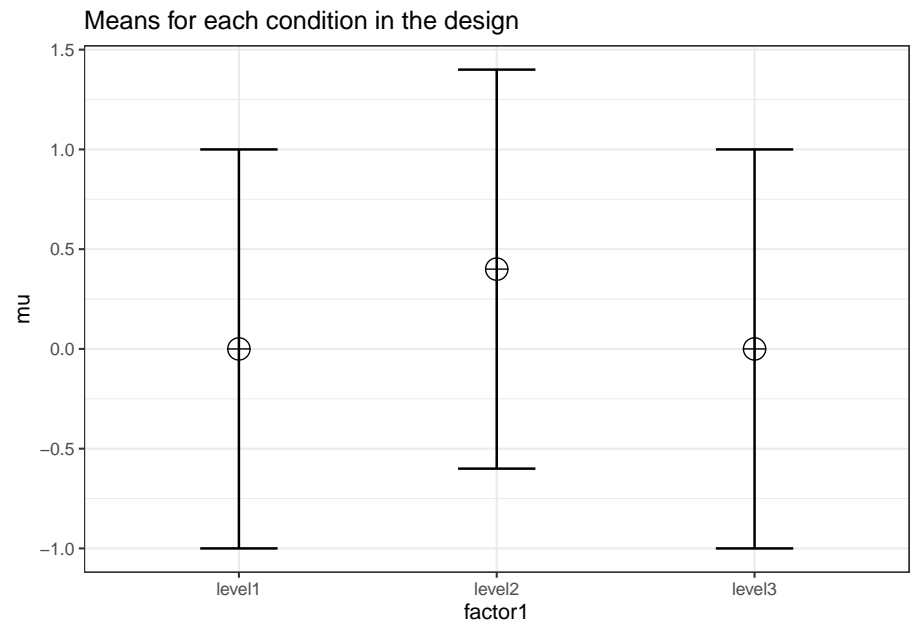
K <- 3
mu <- c(0, 0.4, 0.0)
n <- 145
sd <- 1
r <- 0
design = paste(K, "b", sep = "")

```

```

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 15.6: Simulated ANOVA Result

	power	effect_size
anova_factor1	94.75	0.0386366

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

Table 15.7: Exact ANOVA Result

	power	partial_eta_squared	cohen_f	non_centrality
factor1	94.89169	0.034565	0.1892154	15.46667

15.7.4 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)

```

15.7.5 Three conditions replication

```

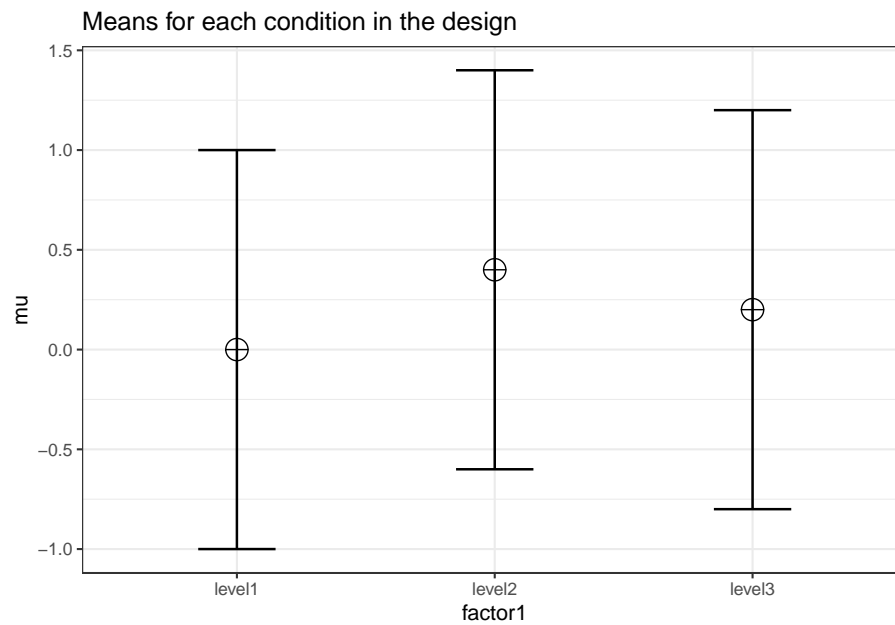
K <- 3
mu <- c(0, 0.4, 0.2)
n <- 82
sd <- 1
design = paste(K, "b", sep = "")

```

```

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 15.8: Simulated ANOVA Result

	power	effect_size
anova_factor1	62.9	0.0340814

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

Table 15.9: Exact ANOVA Result

	power	partial_eta_squared	cohen_f	non_centrality
factor1	61.94317	0.0262863	0.1643042	6.56

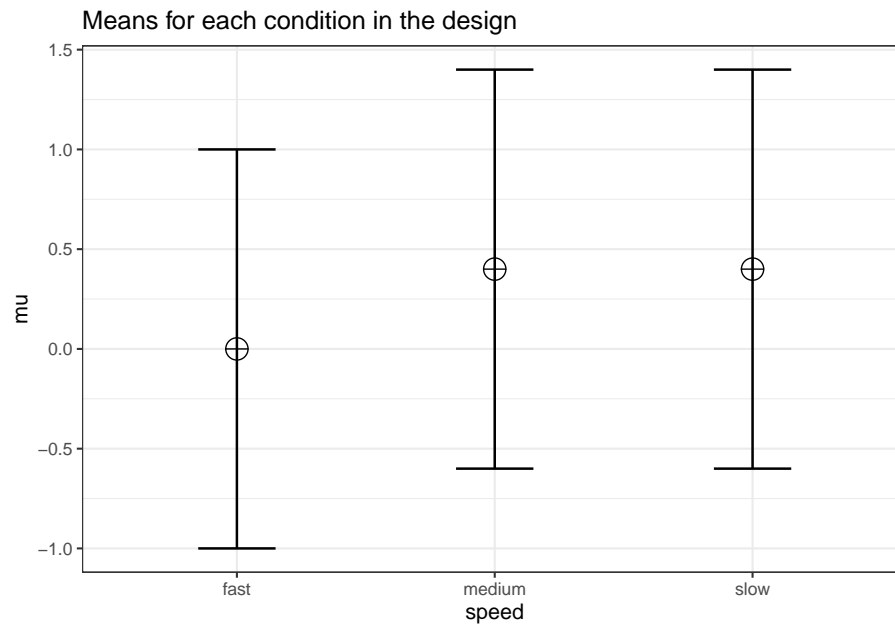
15.8 Repeated Measures

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 15.10: Simulated ANOVA Result

	power	effect_size
anova_speed	95.11	0.1070021

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

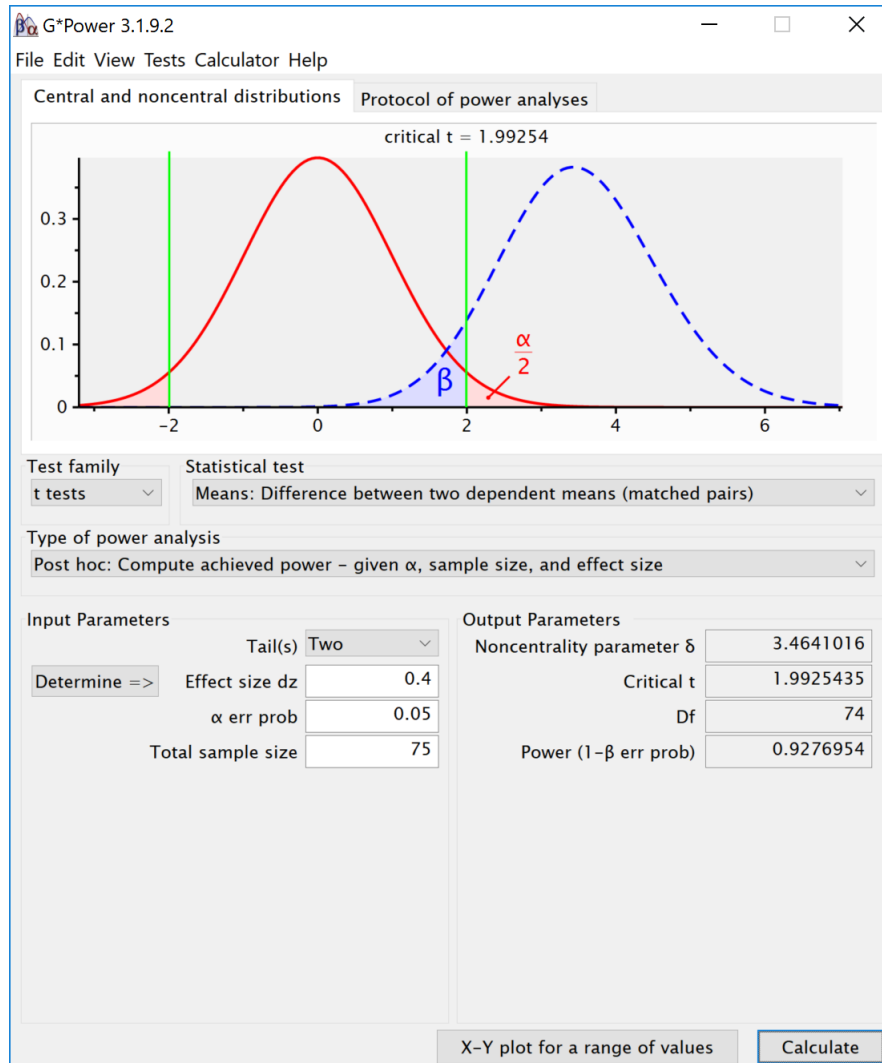
Table 15.11: Exact ANOVA Result

	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:

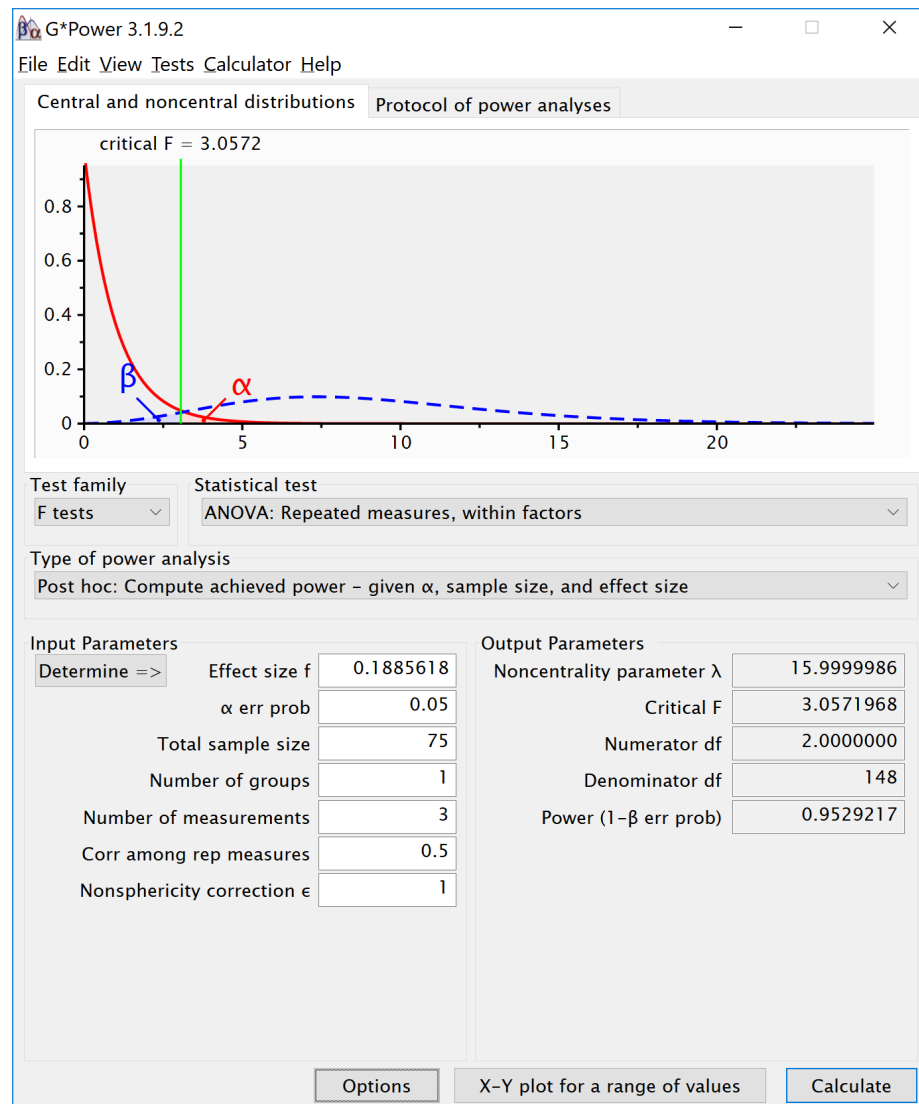


We can perform an post-hoc power analysis in Gpower. We can calculate Cohen's f based on the means and sd, using our own custom formula.

```
# Our simulation is based on the 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  $f = 0.5 \cdot d$ .
# Imagine 2 group,  $\mu = 1$  and  $2$ 
# Grand mean is  $1.5$ ,
# we have  $\sqrt{(\text{sum}(0.5^2 + 0.5^2)/2)}$ , 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:



15.8.1 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] 0.9479
```

```
## [1] 0.9201
```

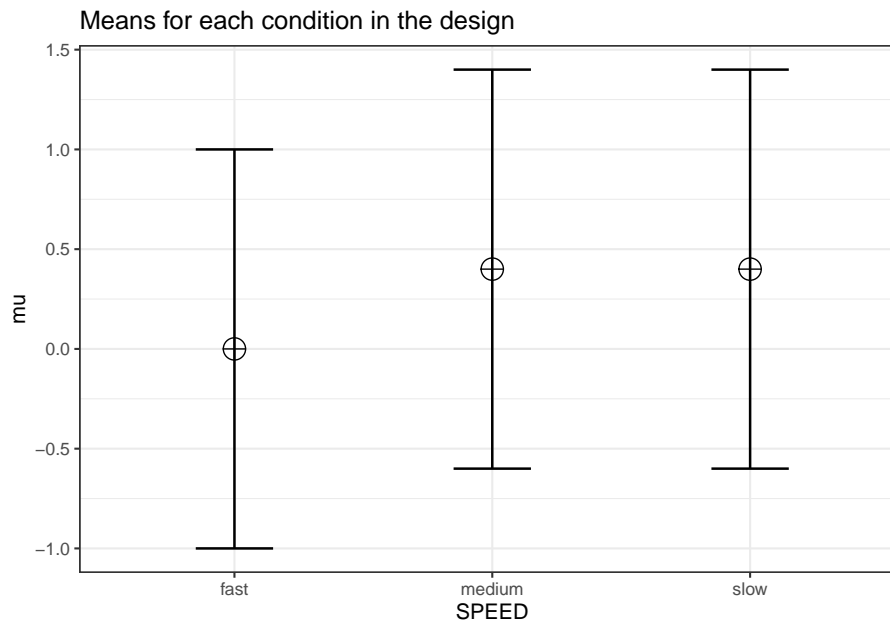
```
## [1] 0.0494
```

```
## [1] 0.9217
```

```
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)
```

Table 15.12: Simulated ANOVA Result

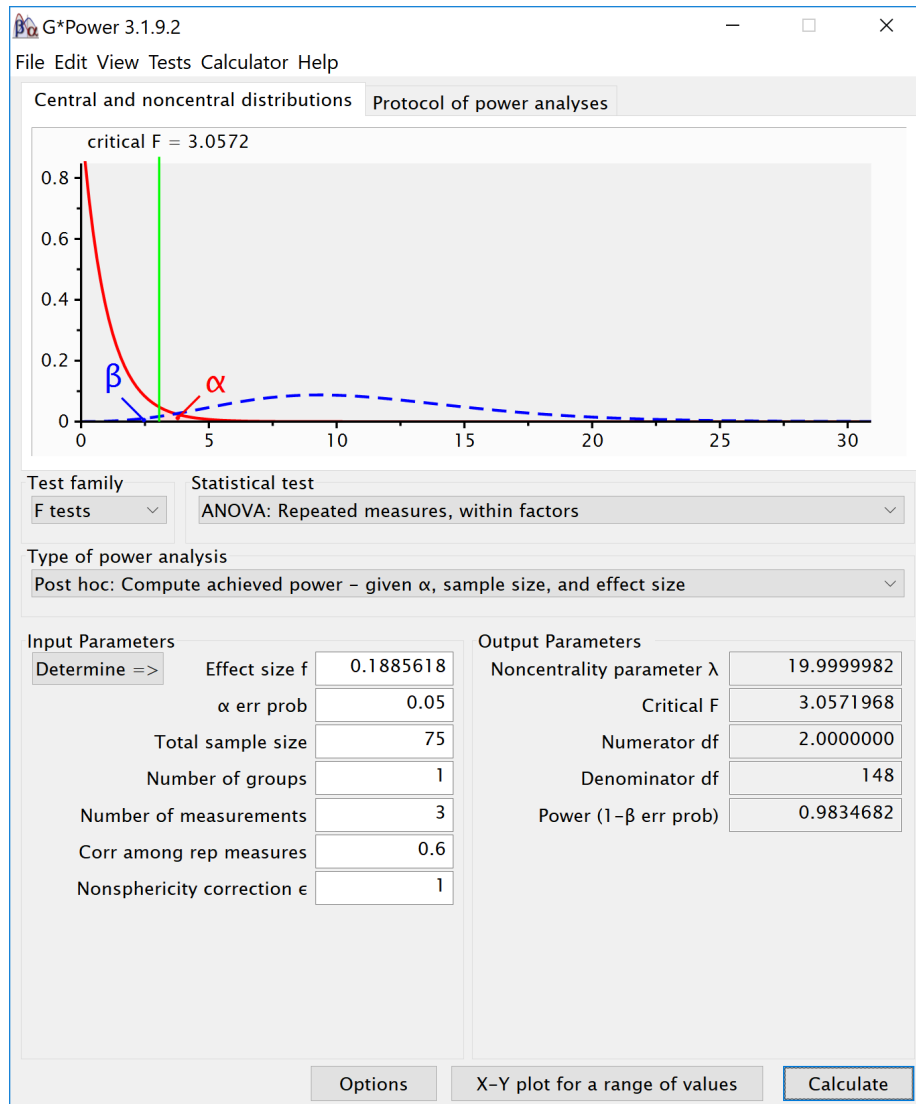
	power	effect_size
anova_SPEED	98.39	0.1285941

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

Table 15.13: Exact ANOVA Result

	power	partial_eta_squared	cohen_f	non_centralilty
SPEED	98.34682	0.1190476	0.3676073	20

Again, this is similar to GPower for the ANOVA:



Bibliography

- Aberson, C. (2021). *pwr2ppl: Power Analyses for Common Designs (Power to the People)*. R package version 0.2.0.
- 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. J. (1997). Detecting repeated measures effects with univariate and multivariate statistics. *Psychological Methods*, 2(2):208–218.
- Allaire, J., Xie, Y., McPherson, J., Luraschi, J., Ushey, K., Atkins, A., Wickham, H., Cheng, J., Chang, W., and Iannone, R. (2021). *rmarkdown: Dynamic Documents for R*. R package version 2.11.
- Althouse, A. D. (2021). Post hoc power: Not empowering, just misleading. *Journal of Surgical Research*, 259:A3–A6.
- 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).
- Caldwell, A. and Vigotsky, A. D. (2020). A case against default effect sizes in sport and exercise science. *PeerJ*, 8:e10314.
- 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. (2020). *pwr: Basic Functions for Power Analysis*. R package version 1.3-0.
- 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.

- Gelman, A. (2019). Don't calculate post-hoc power using observed estimate of effect size. *Annals of surgery*, 269(1):e9–e10.
- Green, P. and MacLeod, C. J. (2016). SIMR : an r package for power analysis of generalized linear mixed models by simulation. *Methods in Ecology and Evolution*, 7(4):493–498.
- Hothorn, T., Bretz, F., and Westfall, P. (2021). *multcomp: Simultaneous Inference in General Parametric Models*. R package version 1.4-17.
- Lakens, D. (2014). Performing high-powered studies efficiently with sequential analyses. *European Journal of Social Psychology*, 44(7):701–710.
- Lakens, D., Pahlke, F., and Wassmer, G. (2021). Group sequential designs: A tutorial.
- Lenth, R. V. (2007). Statistical power calculations. *Journal of animal science*, 85(suppl_13):E24–E29.
- 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.
- Perugini, M., Gallucci, M., and Costantini, G. (2014). Safeguard power as a protection against imprecise power estimates. *Perspectives on Psychological Science*, 9(3):319–332.
- 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 (2021). *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. (2021). *bookdown: Authoring Books and Technical Documents with R Markdown*. R package version 0.24.