Validation of Superpower

Aaron Caldwell & Daniël Lakens

2019-09-12

Contents

This is a compilation of validation documents for `Superpower` written in **Markdown** and compiled by **Bookdown**.

Chapter 1

Introduction

Below we have included various examples of the performance of `Superpower` against other R packages and statistical programs (such as GPower).

Chapter 2

One-way ANOVA Part 1

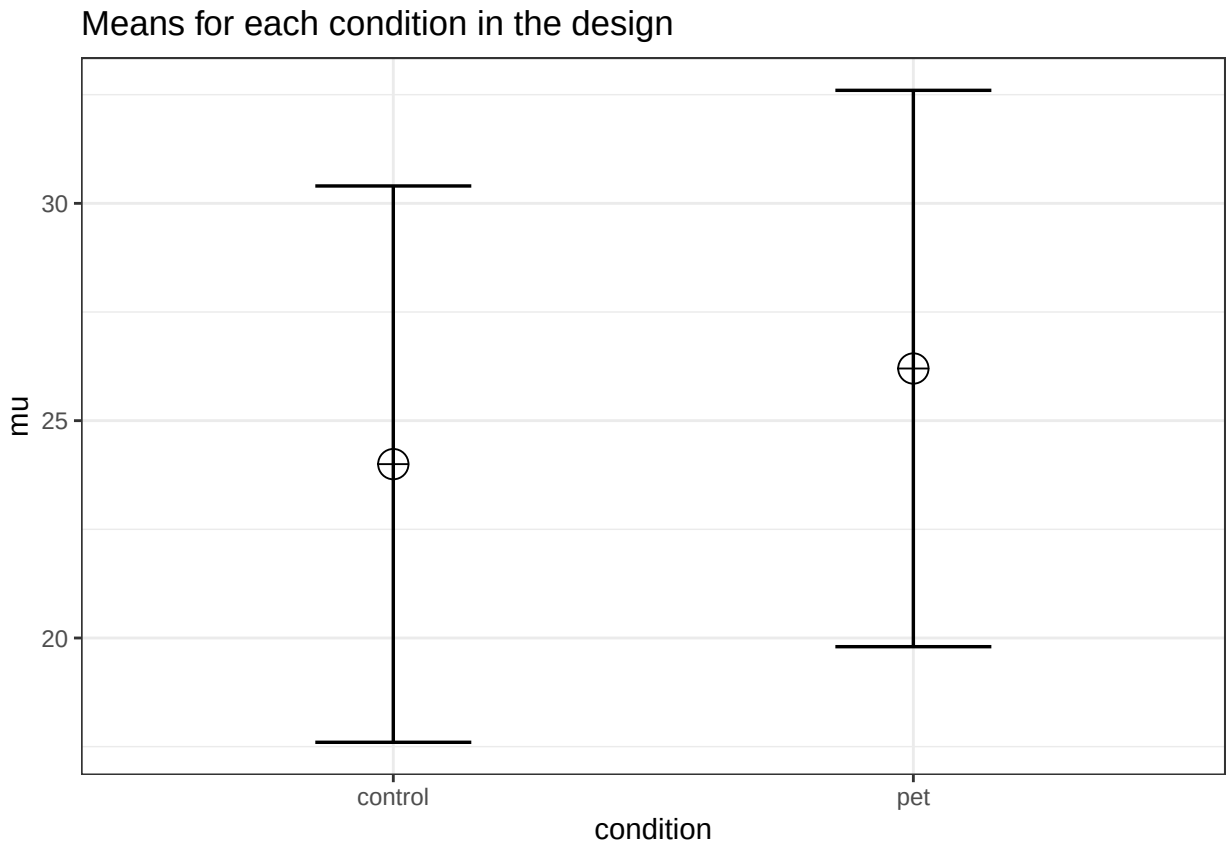
2.1 Validation of Power in One-Way ANOVA

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

2.2 Two conditions

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

```
string <- "2b"
n <- 100
# We are thinking of running 50 peope in each condition
mu <- c(24, 26.2)
# Enter means in the order that matches the labels below.
# In this case, control, cat, dog.
sd <- 6.4
labelnames <- c("condition", "control", "pet") #
# the label names should be in the order of the means specified above.
design_result <- ANOVA_design(design = string,
                   n = n,
                   mu = mu,
                   sd = sd,
                   labelnames = labelnames)
```




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

```
## Power and Effect sizes for ANOVA tests
##                 power effect_size
## anova_condition    65     0.03314
##
## Power and Effect sizes for contrasts
##                                    power effect_size
## p_condition_control_condition_pet    65      0.3381
```

The result shows that we have exactly the same power for the ANOVA, as we have for the t-test. This is because 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 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 power.t.test.

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

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

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

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

```
K <- 2
n <- 100
sd <- 6.4
r <- 0
#Calculate f when running simulation
f <- pwr.anova.test(n = n,
                    k = K,
                    power = 0.9,
                    sig.level = alpha_level)$f
f
```

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

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

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

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

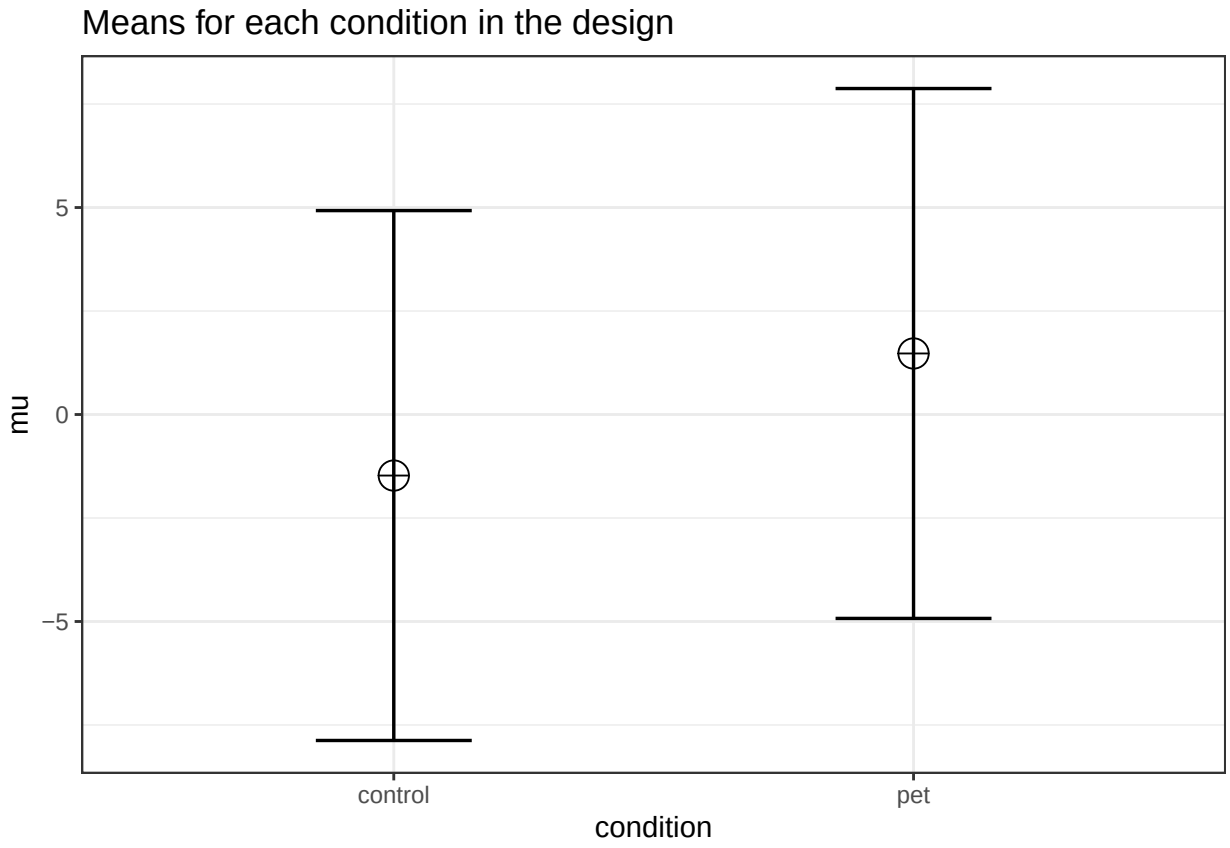
Expressed in eta-squared, we can detect values of eta-squared = 0.05 or larger.

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

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

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

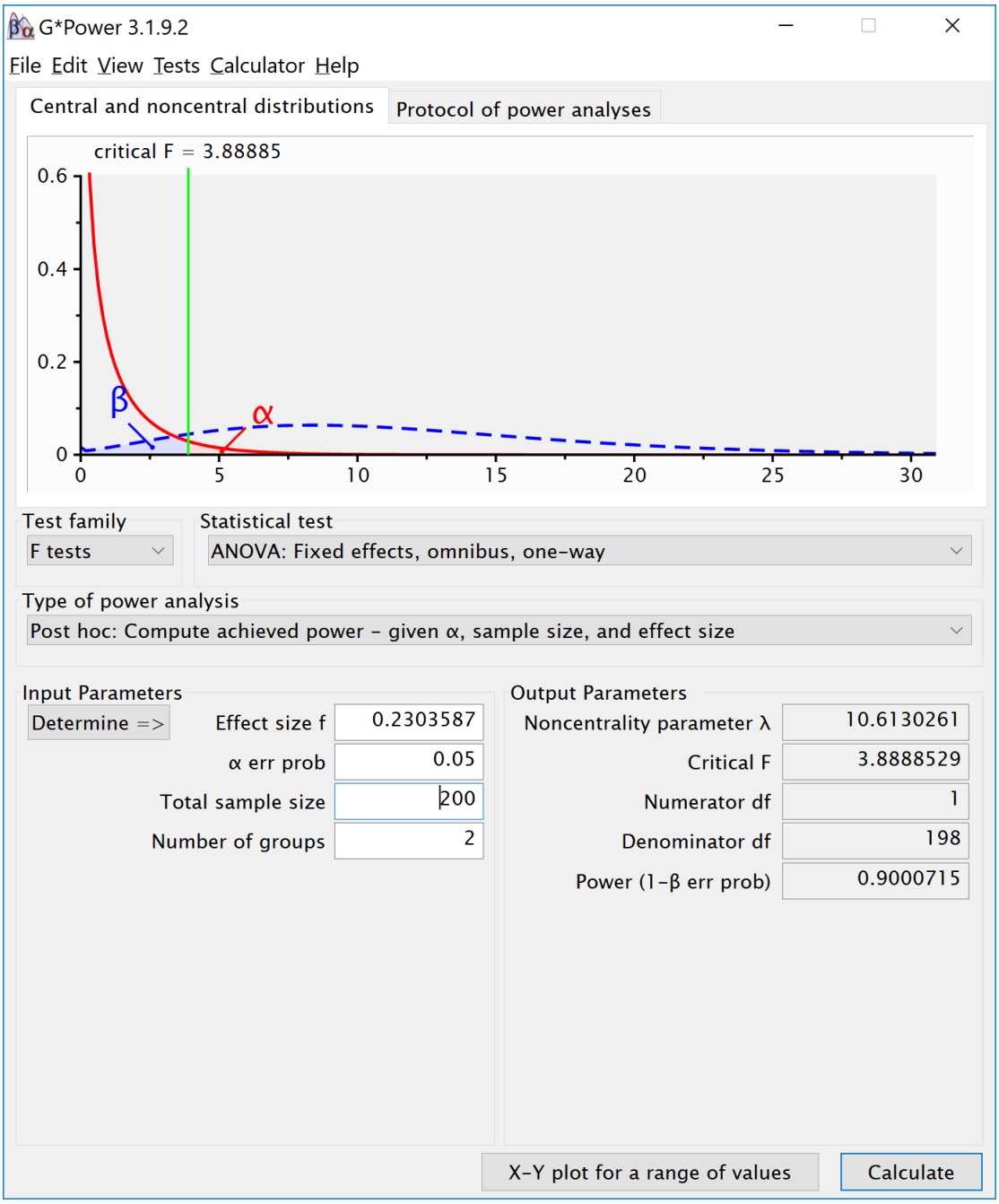
```
design_result <- ANOVA_design(design = string,
                    n = n,
                    mu = mu,
                    sd = sd,
                    labelnames = labelnames)
```




```
ANOVA_power(design_result, alpha_level = alpha_level, nsims = nsims)
```

```
## Power and Effect sizes for ANOVA tests
##                 power effect_size
## anova_condition    87     0.04984
##
## Power and Effect sizes for contrasts
##                                   power effect_size
## p_condition_control_condition_pet    87      0.4386
```

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.

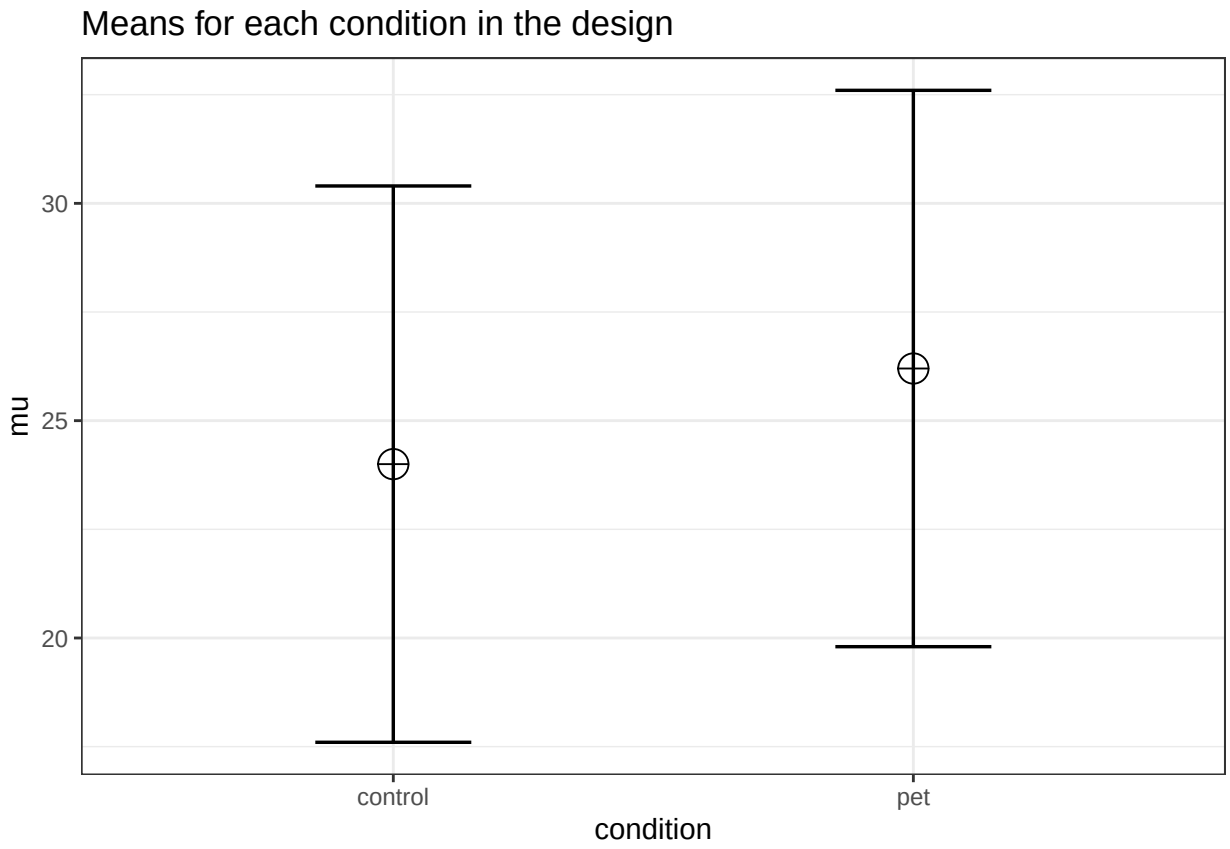
```
K <- 2
power <- 0.9
f <- 0.171875
```

```
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)
```

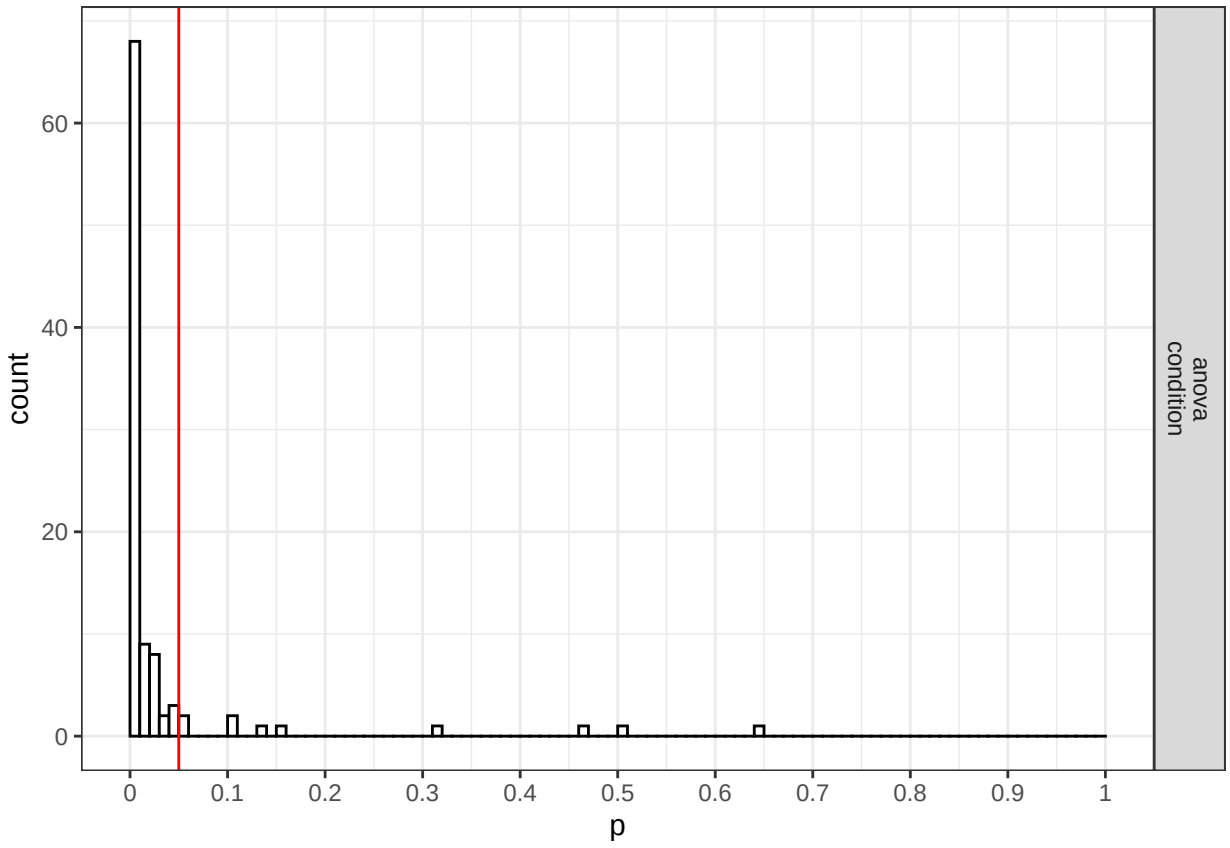



```
alpha_level <- 0.05
power_result <- ANOVA_power(design_result, alpha_level = alpha_level, nsims = nsims)
```

```
## Power and Effect sizes for ANOVA tests
##                 power effect_size
## anova_condition    90     0.02933
##
## Power and Effect sizes for contrasts
##                                    power effect_size
## p_condition_control_condition_pet    90      0.3315
```

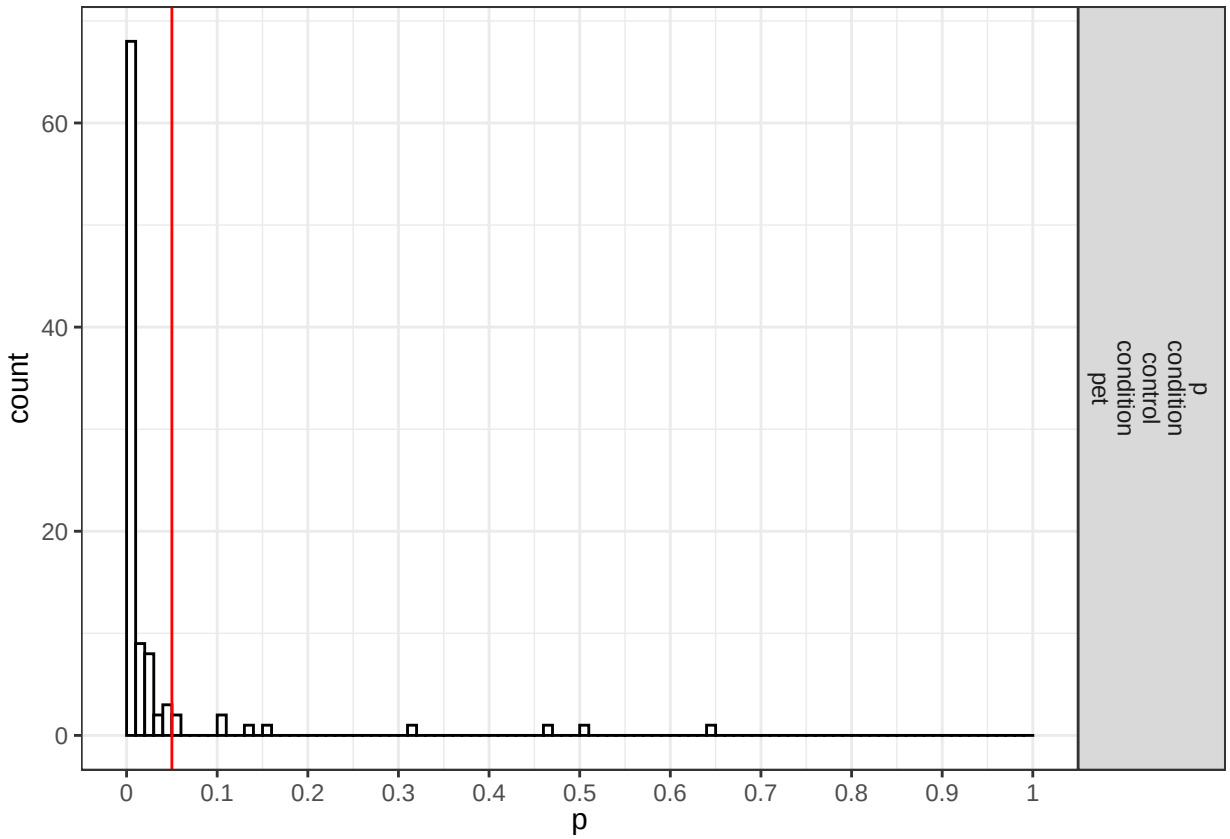
We stored the result from the power analysis in an object. This allows us to request plots (which are not printed automatically) showing the p-value distribution. If we request power_result$plot1 we get the p-value distribution for the ANOVA:

```
power_result$plot1
```



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

```
power_result$plot2
```


count
60
40
20
0
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
p
p
condition
control
condition
pet

Chapter 3

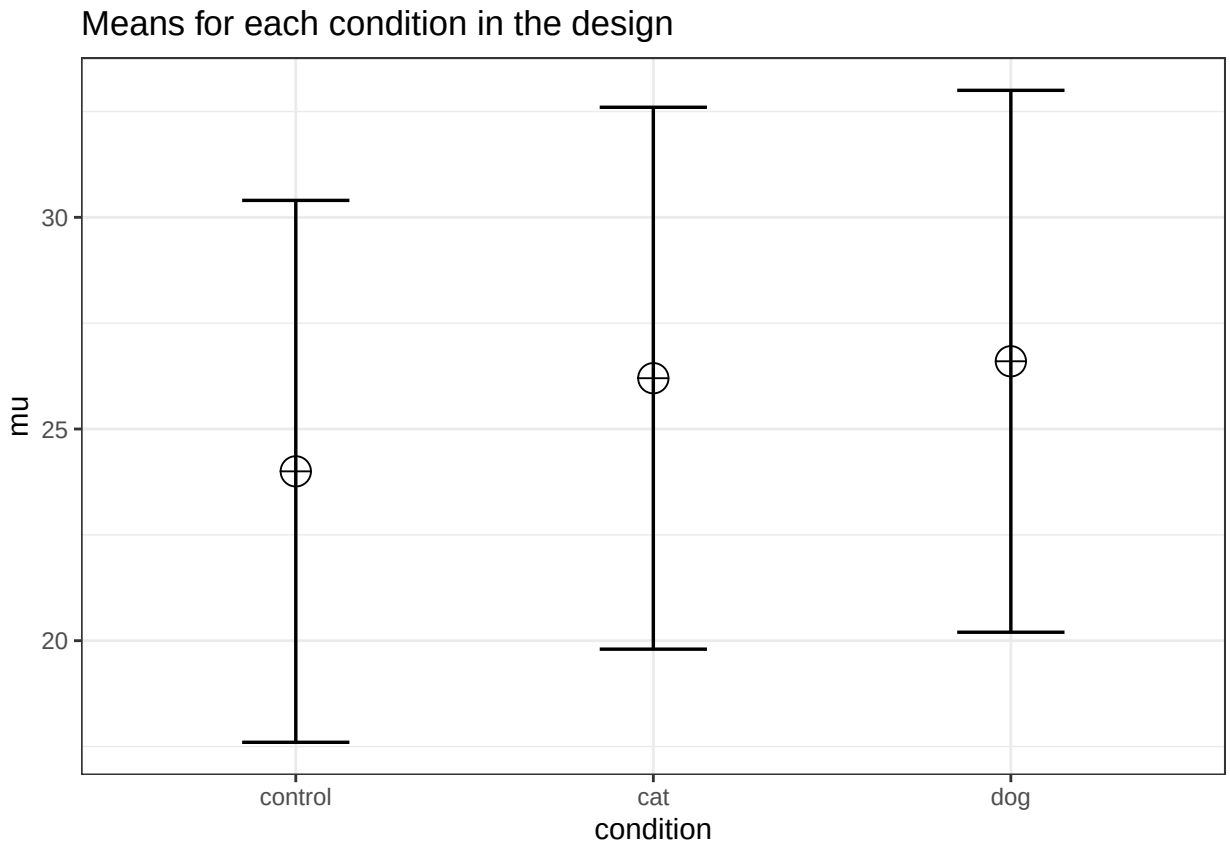
One-way ANOVA Part 2

3.1 Validation of Power in One-Way ANOVA with Brysbaert example

3.1.1 Three conditions

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

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




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

```
## Power and Effect sizes for ANOVA tests
##                 power effect_size
## anova_condition    49     0.04673
##
## Power and Effect sizes for contrasts
##                                    power effect_size
## p_condition_control_condition_cat     36     0.35331
## p_condition_control_condition_dog     59     0.43522
## p_condition_cat_condition_dog          7     0.07778
```

```
#should yield
#0.3983064
#0.5205162
#0.06104044
```

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

3.2 Power for simple effects

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

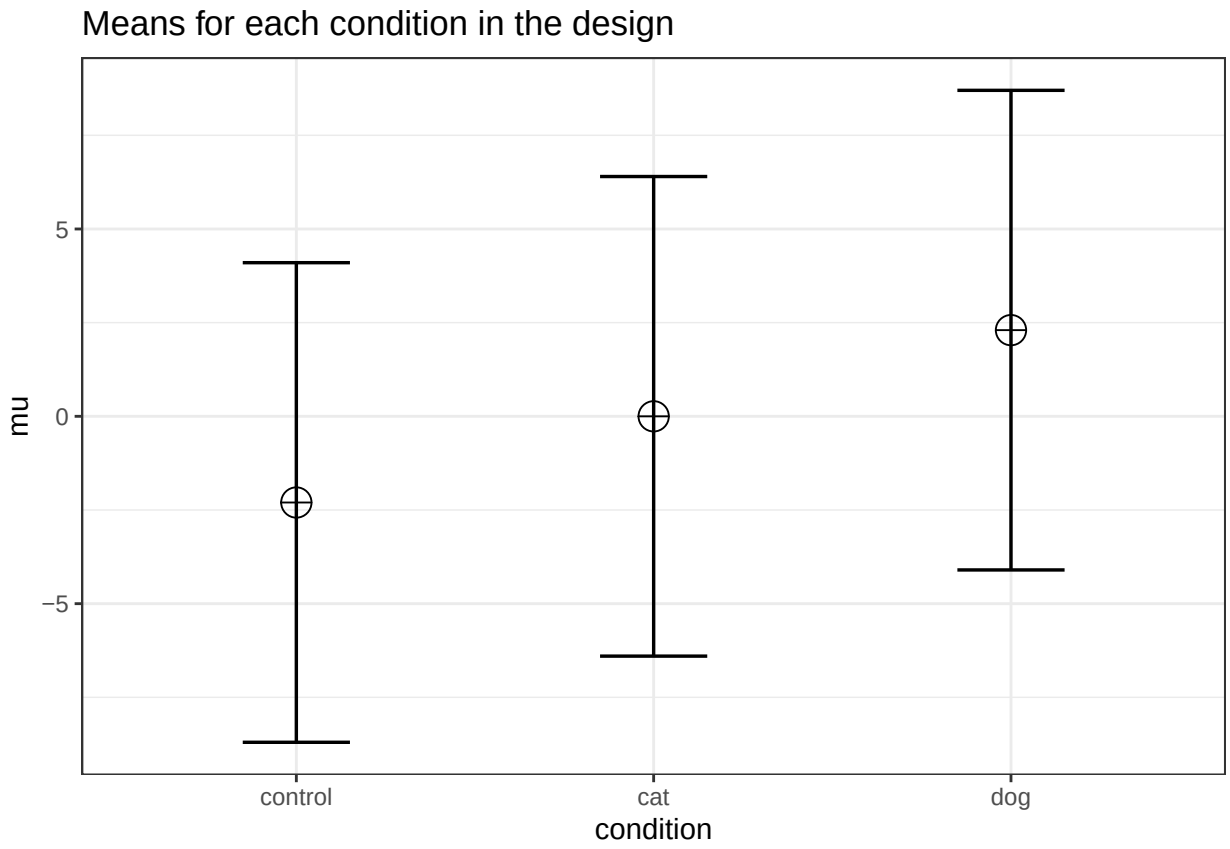
Expressed in eta-squared, we can detect values of eta-squared = 0.0793 or larger.

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

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

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

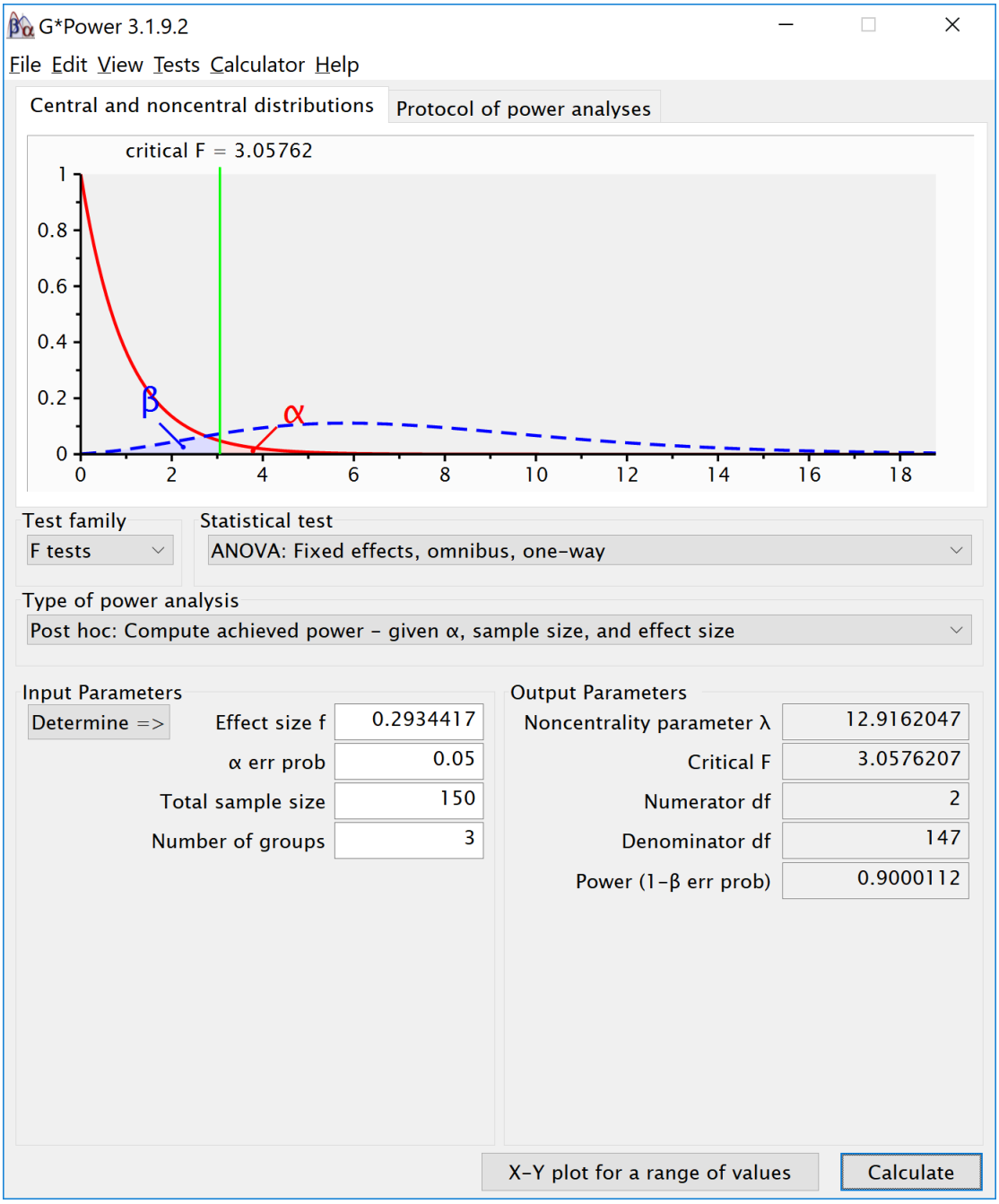
```
design_result <- ANOVA_design(design = string,
                    n = n,
                    mu = mu,
                    sd = sd,
                    labelnames = labelnames)
```




```
ANOVA_power(design_result, alpha_level = alpha_level, nsims = nsims)
```

```
## Power and Effect sizes for ANOVA tests
##                 power effect_size
## anova_condition    92      0.1056
##
## Power and Effect sizes for contrasts
##                                    power effect_size
## p_condition_control_condition_cat     50      0.4132
## p_condition_control_condition_dog     97      0.7993
## p_condition_cat_condition_dog         46      0.3803
```

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.

```
anova1f_3<-function(m1=NULL,m2=NULL,m3=NULL,s1=NULL,s2=NULL,s3=NULL,n1=NULL,n2=NULL,n3=NULL,alpha=.05){
x<-stats::rnorm(n1,m1,s1)
X<-x
MEAN<-m1
SD<-s1
Z <- (((X - mean(X, na.rm = TRUE))/stats::sd(X, na.rm = TRUE))) * SD
y<-MEAN + Z
group<-rep("A1",n1)
l1<-data.frame(y, group)
x<-stats::rnorm(n2,m2,s2)
X<-x
MEAN<-m2
SD<-s2
Z <- (((X - mean(X, na.rm = TRUE))/stats::sd(X, na.rm = TRUE))) * SD
y<-MEAN + Z
group<-rep("A2",n2)
l2<-data.frame(y, group)
x<-stats::rnorm(n3,m3,s3)
X<-x
```

```
MEAN<-m3
SD<-s3
Z <- (((X - mean(X, na.rm = TRUE))/stats::sd(X, na.rm = TRUE))) * SD
y<-MEAN + Z
group<-rep("A3",n3)
l3<-data.frame(y, group)
simdat<-rbind(l1,l2,l3)
anova<-stats::aov(y~group, data=simdat)
anova<-car::Anova(anova, type="III")
SSA<-anova[2,1] #column, row
SSwin<-anova[3,1]
dfwin<-anova[3,2]
dfbg<-anova[2,2]
eta2<-SSA/(SSA+SSwin)
f2<-eta2/(1-eta2)
lambda<-f2*dfwin
minusalpha<-1-alpha
Ft<-stats::qf(minusalpha, dfbg, dfwin)
power<-1-stats::pf(Ft, dfbg,dfwin,lambda)
list(Power = power)}
```

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)
```

```
## $Power
## [1] 0.4769468
```

```
#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)
```

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

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

Chapter 4

One-way ANOVA Part 3

We first repeat the simulation by Brysbaert:

```
# 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
# 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
alpha1 = .05 #alpha level for the omnibus ANOVA
alpha2 = .05 #alpha level for three post hoc one-tailed t-tests Bonferroni correction
# create progress bar in case it takes a while
#pb <- winProgressBar(title = "progress bar", min = 0, max = nSim, width = 300)
# 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
library(lsr)
for(i in 1:nSim){ #for each simulated experiment
 # setWinProgressBar(pb, i, title=paste(round(i/nSim*100, 1), "% done"))
  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]
  }
#close(pb)#close progress bar
# results are as predicted when omnibus ANOVA is significant, t-tests are significant between x and y p
#printing all unique tests (adjusted code by DL)
sum(p1<alpha1)/nSim
```

```
## [1] 0.77
```

```
sum(p2<alpha2)/nSim
```

```
## [1] 0.75
```

```
sum(p3<alpha2)/nSim
```

```
## [1] 0.74
```

```
sum(p4<alpha2)/nSim
```

```
## [1] 0.04
```

```
mean(pes1)
```

```
## [1] 0.04181189
```

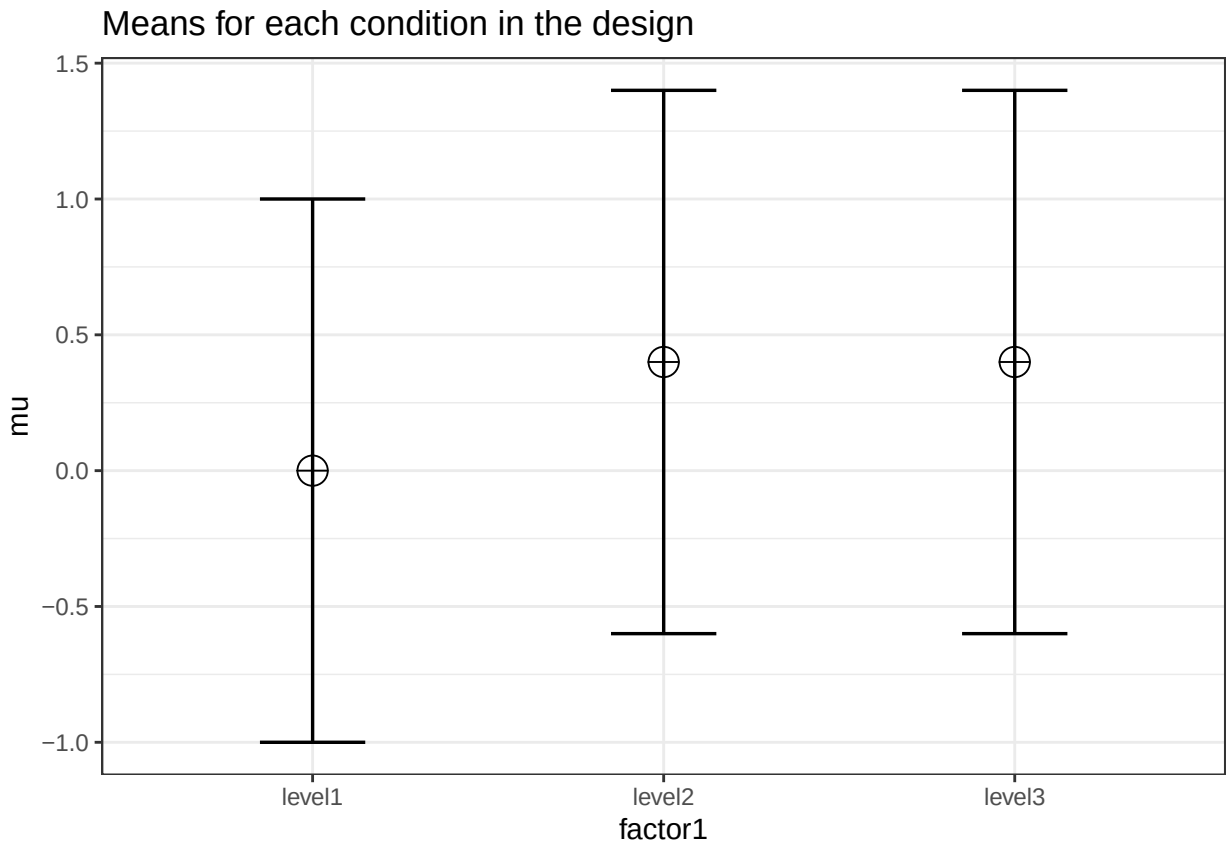
```
mean(pes2)
```

```
## [1] 0.04460323
```

4.1 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 =string,
                   n = n,
                   mu = mu,
                   sd = sd,
                   labelnames = c("factor1", "level1", "level2", "level3"))
```




```
ANOVA_power(design_result, nsims = nsims)
```

```
## Power and Effect sizes for ANOVA tests
##               power effect_size
## anova_factor1    80     0.04101
##
## Power and Effect sizes for contrasts
##                                  power effect_size
## p_factor1_level1_factor1_level2    77    0.398176
## p_factor1_level1_factor1_level3    71    0.402702
## p_factor1_level2_factor1_level3     5    0.002075
```

4.2 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
```

```
alpha1 = .05 #alpha level for the omnibus ANOVA
alpha2 = .05 #alpha level for three post hoc one-tailed t-tests Bonferroni correction
# create progress bar in case it takes a while
#pb <- winProgressBar(title = "progress bar", min = 0, max = nSim, width = 300)
# 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
library(lsr)
for(i in 1:nSim){ #for each simulated experiment
 # setWinProgressBar(pb, i, title=paste(round(i/nSim*100, 1), "% done"))
  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]
  }
#close(pb)#close progress bar
# results are as predicted when omnibus ANOVA is significant, t-tests are significant between x and y p
#printing all unique tests (adjusted code by DL)
sum(p1<alpha1)/nSim
```

```
## [1] 0.98
```

```
sum(p2<alpha2)/nSim
```

```
## [1] 0.93
```

```
sum(p3<alpha2)/nSim
```

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

```
sum(p4<alpha2)/nSim
```

```
## [1] 0.94
```

```
mean(pes1)
```

```
## [1] 0.04333334
```

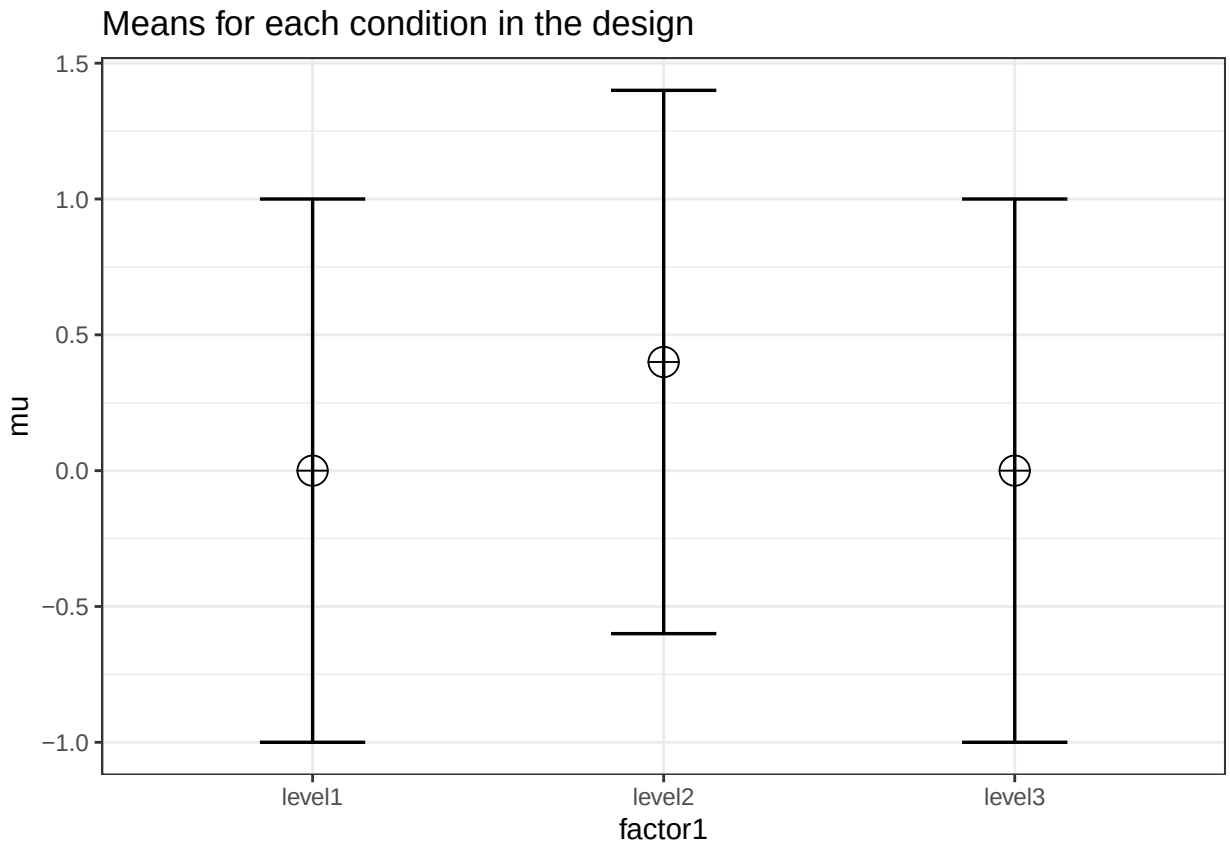
```
mean(pes2)
```

```
## [1] 0.04761062
```

4.3 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 =string,
                    n = n,
                    mu = mu,
                    sd = sd,
                    labelnames = c("factor1", "level1", "level2", "level3"))
```



```
ANOVA_power(design_result, nsims = nsims)
```

```
## Power and Effect sizes for ANOVA tests
##               power effect_size
## anova_factor1    92     0.03649
##
## Power and Effect sizes for contrasts
##                                  power effect_size
## p_factor1_level1_factor1_level2    88    0.391587
## p_factor1_level1_factor1_level3     2    0.003771
## p_factor1_level2_factor1_level3    92   -0.387947
```

4.4 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
alpha1 = .05 #alpha level for the omnibus ANOVA
alpha2 = .05 #alpha level for three post hoc one-tailed t-tests Bonferroni correction
# create progress bar in case it takes a while
#pb <- winProgressBar(title = "progress bar", min = 0, max = nSim, width = 300)
# 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
library(lsr)
for(i in 1:nSim){ #for each simulated experiment
 # setWinProgressBar(pb, i, title=paste(round(i/nSim*100, 1), "% done"))
  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]
  }
#close(pb)#close progress bar
# results are as predicted when omnibus ANOVA is significant, t-tests are significant between x and y p
#printing all unique tests (adjusted code by DL)
sum(p1<alpha1)/nSim
```

```
## [1] 0.59
```

```
sum(p2<alpha2)/nSim
```

```
## [1] 0.67
```

```
sum(p3<alpha2)/nSim
```

```
## [1] 0.27
```

```
sum(p4<alpha2)/nSim
```

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

```
mean(pes1)
```

```
## [1] 0.03544864
```

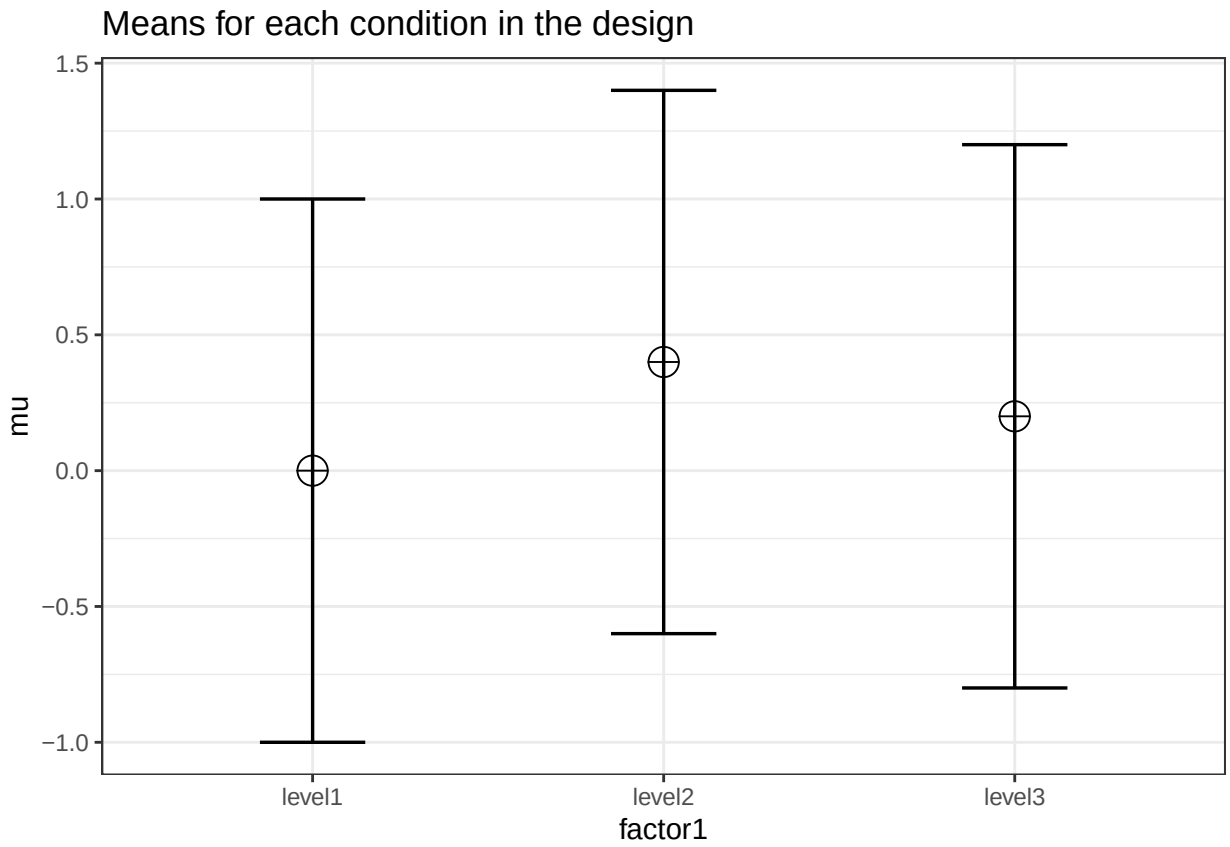
```
mean(pes2)
```

```
## [1] 0.04780688
```

4.5 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 =string,
                   n = n,
                   mu = mu,
                   sd = sd,
                   labelnames = c("factor1", "level1", "level2", "level3"))
```




```
ANOVA_power(design_result, nsims = nsims)
```

```
## Power and Effect sizes for ANOVA tests
##               power effect_size
## anova_factor1    55     0.03171
##
## Power and Effect sizes for contrasts
##                                  power effect_size
## p_factor1_level1_factor1_level2    66      0.3863
## p_factor1_level1_factor1_level3    20      0.1931
## p_factor1_level2_factor1_level3    25     -0.1947
```

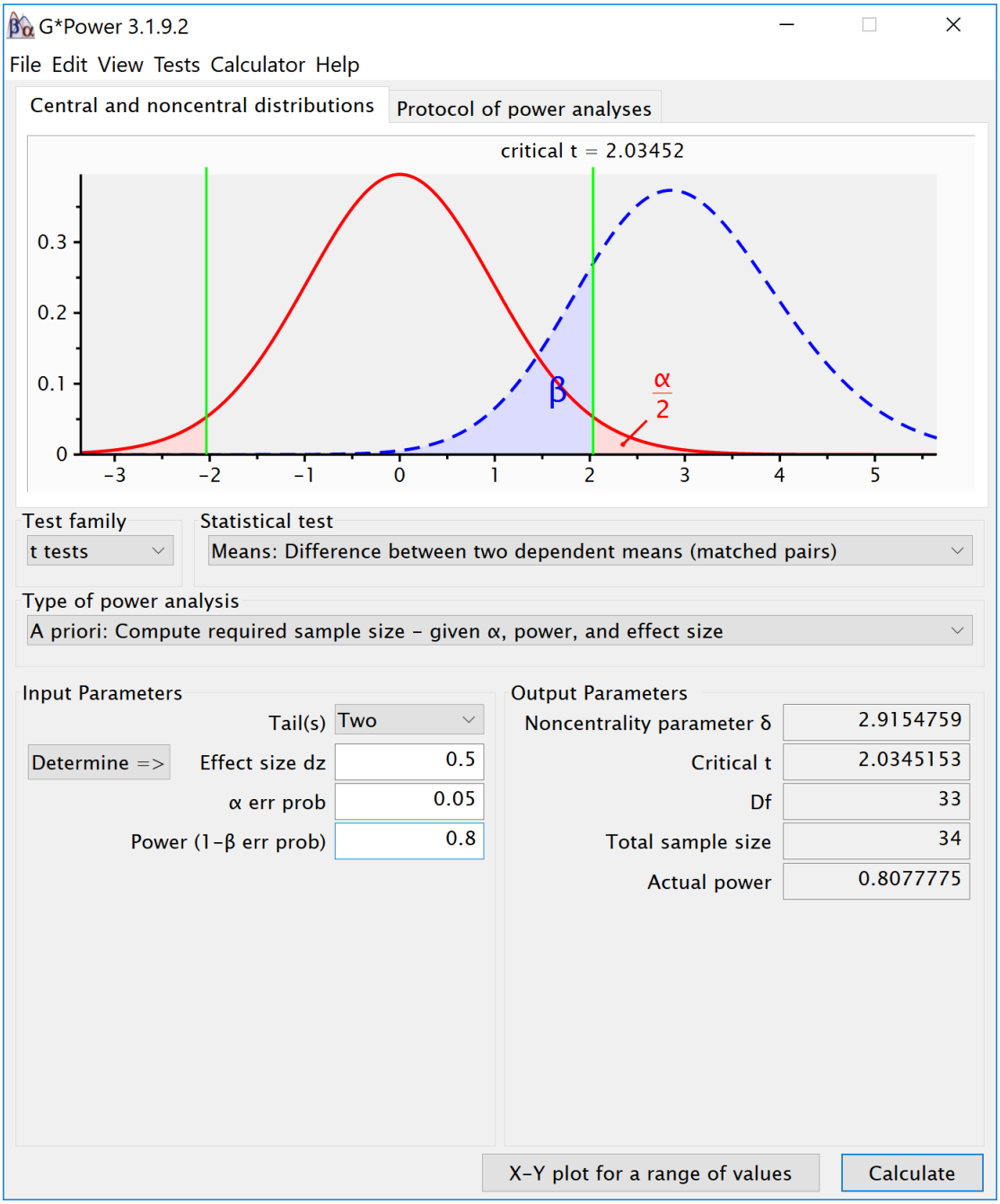
Chapter 5

Repeated Measures-ANOVA 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 difference compared to a between subject design is that repeated measures can be correlated, and in psychology, they often are. Let's first explore the impact of this correlation on the power of a repeated measures ANOVA.

5.1 Two conditions, medium effect size

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



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

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

The division by 2 in the equation is due to the fact that in a two-condition within design every participant provides two data-points. The extent to which this reduces the sample size compared to a between-subject design depends on the correlation (r) between the two dependent variables, as indicated by the 1-r part of

the equation. If the correlation is 0, a within-subject design needs half as many participants as a between-subject design (e.g., 64 instead 128 participants), simply because every participants provides 2 datapoints. The higher the correlation, the larger the relative benefit of within designs, and whenever the correlation is negative (up to -1) the relative benefit disappears.

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

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

Therefore, the relation between dz and d is $\sqrt{2(1-\rho)}$. As Cohen (1988) writes: "In other words, a given difference between population means for matched (dependent) samples is standardized by a value which is $\sqrt{2(1-\rho)}$ as large as would be the case were they independent. If we enter a correlation of 0.5 in the formula, we get $\sqrt{2(0.5)} = 1$. In other words, when the correlation is 0.5, d = dz. When there is a strong correlation between dependent variables, for example r = 0.9, we get $d = d_z\sqrt{2(1-0.9)}$, and a dz of 1 would be a d = 0.45. Reversely, $d_z = \frac{d}{\sqrt{2(1-r)}}$, so with a r = 0.9, a d of 1 would be a dz = 2.24. Some consider this increase in dz compared to d when observations are strongly correlated an 'inflation' when estimating effect sizes, but since the reduction in the standard deviation of the difference scores due to the correlation makes it easier to distinguish signal from noise in a hypothesis test, it leads to a clear power benefit.

```
# 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")
```

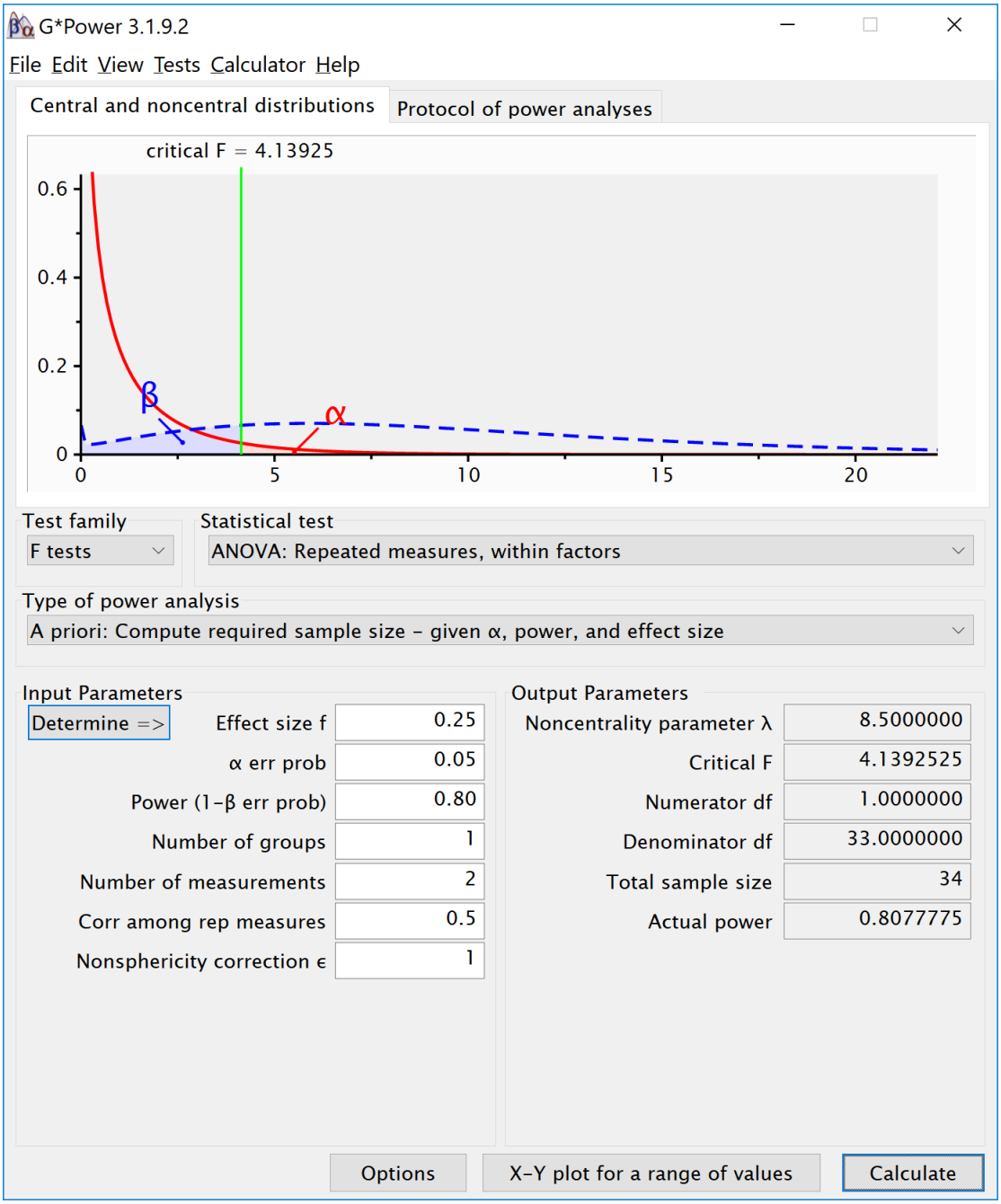
```
##
##      Two-sample t test power calculation
##
##               n = 2000, 4000, 8000
##               d = 0.05
##       sig.level = 0.05
##           power = 0.3524674, 0.6086764, 0.8853424
##     alternative = two.sided
##
## NOTE: n is number in *each* group
```

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

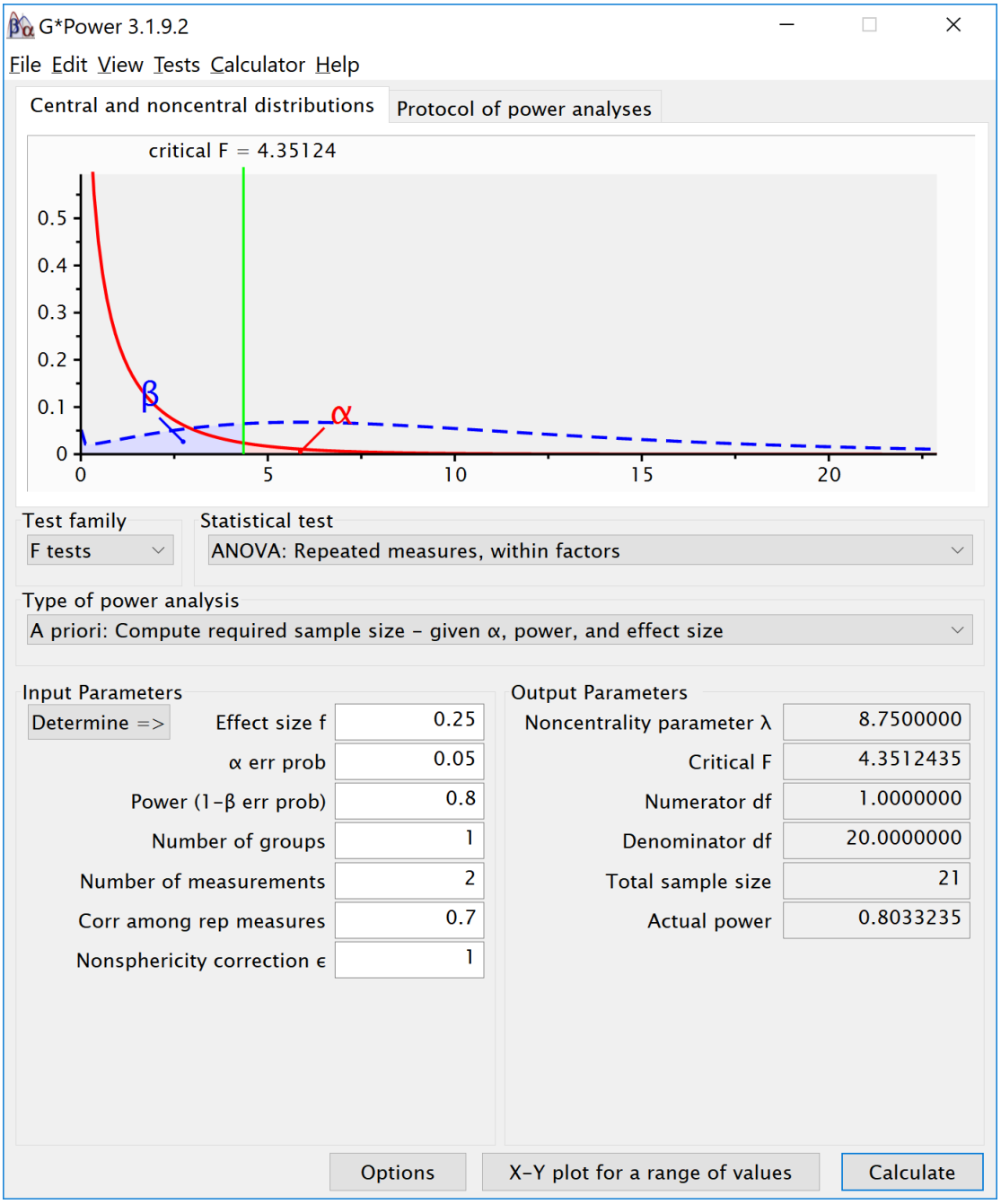
```
##
##      Paired t test power calculation
##
##               n = 1000, 2000, 4000
##               d = 0.05
```

```
##       sig.level = 0.05
##           power = 0.3520450, 0.6083669, 0.8852320
##     alternative = two.sided
##
## NOTE: n is number of *pairs*
```

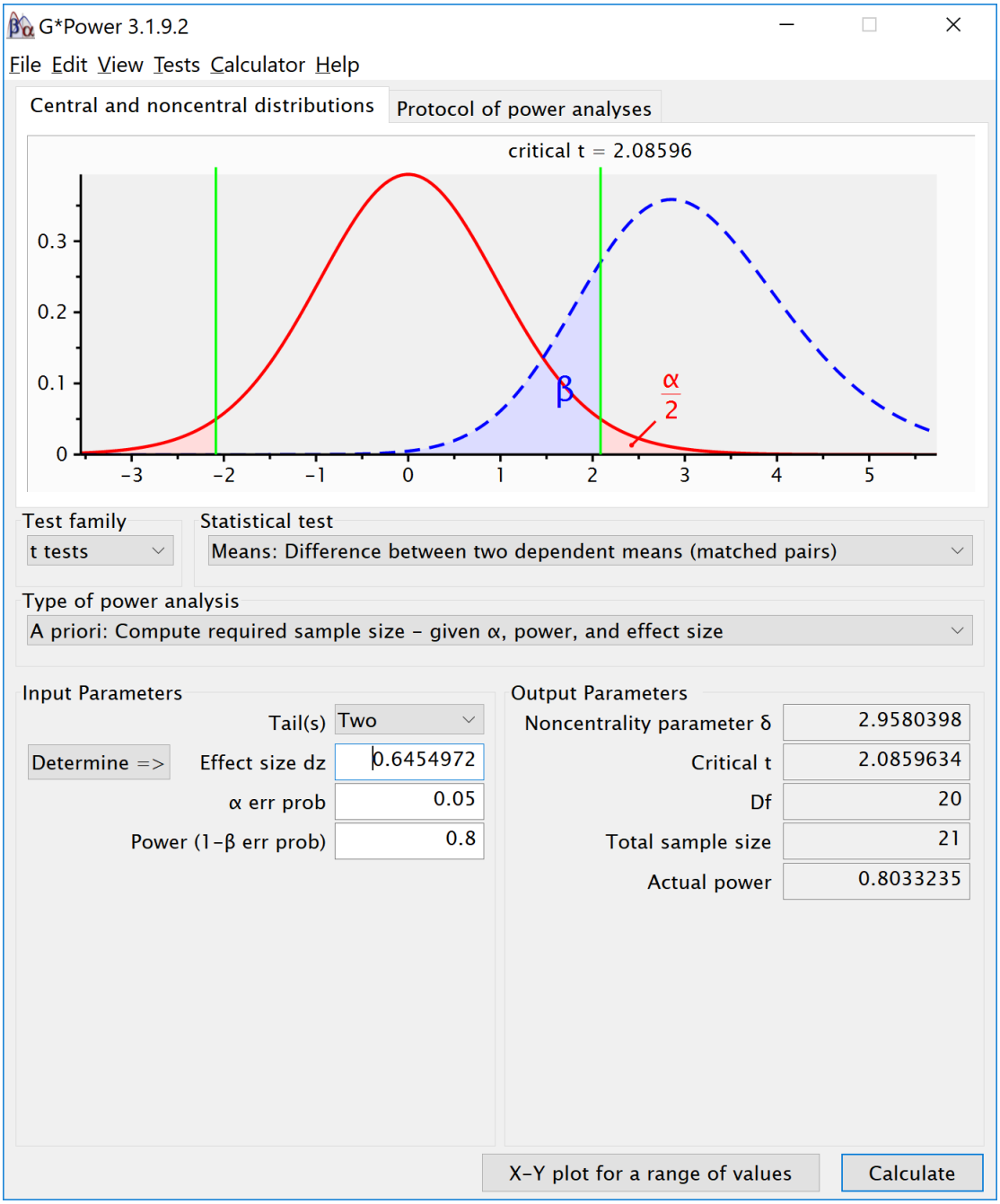
There is no equivalent fz for Cohen's f for a within subject ANOVA. For two groups, we can directly compute Cohen's f from Cohen's d for two groups, as Cohen (1988) describes, because f = 1/2d. For a d = 0.5, f = 0.25. In Gpower we can run a 2 group within-subject power analysis for ANOVA. We plan for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



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



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



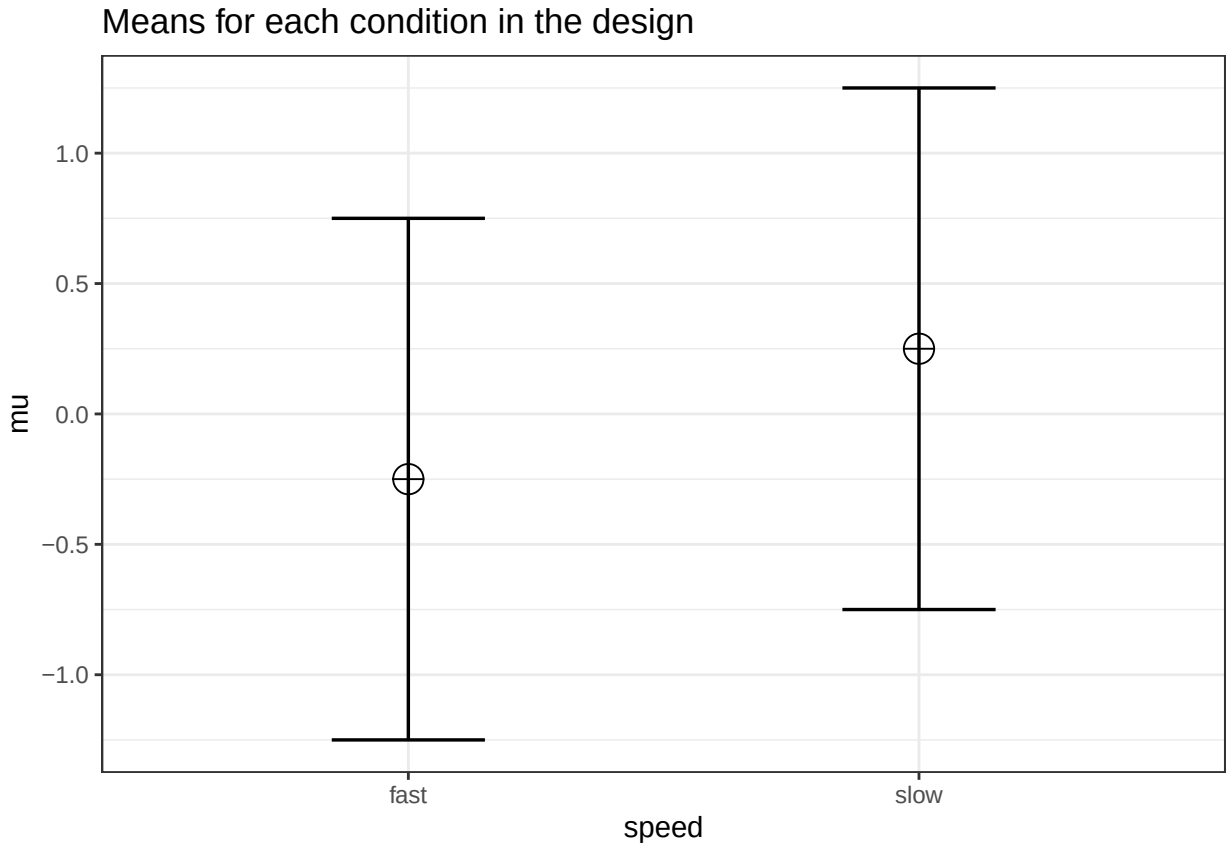
We can perform both these power analyses using simuations as well. We set groups to 2 for the simulation, n = 34 (which should give 80.777 power, according to the g*power program), a correlation among repeated measures of 0.5, and an alpha of 0.05. In this case, we simulate data with means -0.25 and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

```
K <- 2
n <- 34
sd <- 1
r <- 0.5
```

```
alpha = 0.05
f <- 0.25
f2 <- f^2
ES <- f2/(f2+1)
ES
```

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

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




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

```
## Power and Effect sizes for ANOVA tests
##             power effect_size
## anova_speed    82      0.2079
```

```
##
## Power and Effect sizes for contrasts
##                          power effect_size
## p_speed_fast_speed_slow    82      0.4971
##
## Within-Subject Factors Included: Check MANOVA Results
```

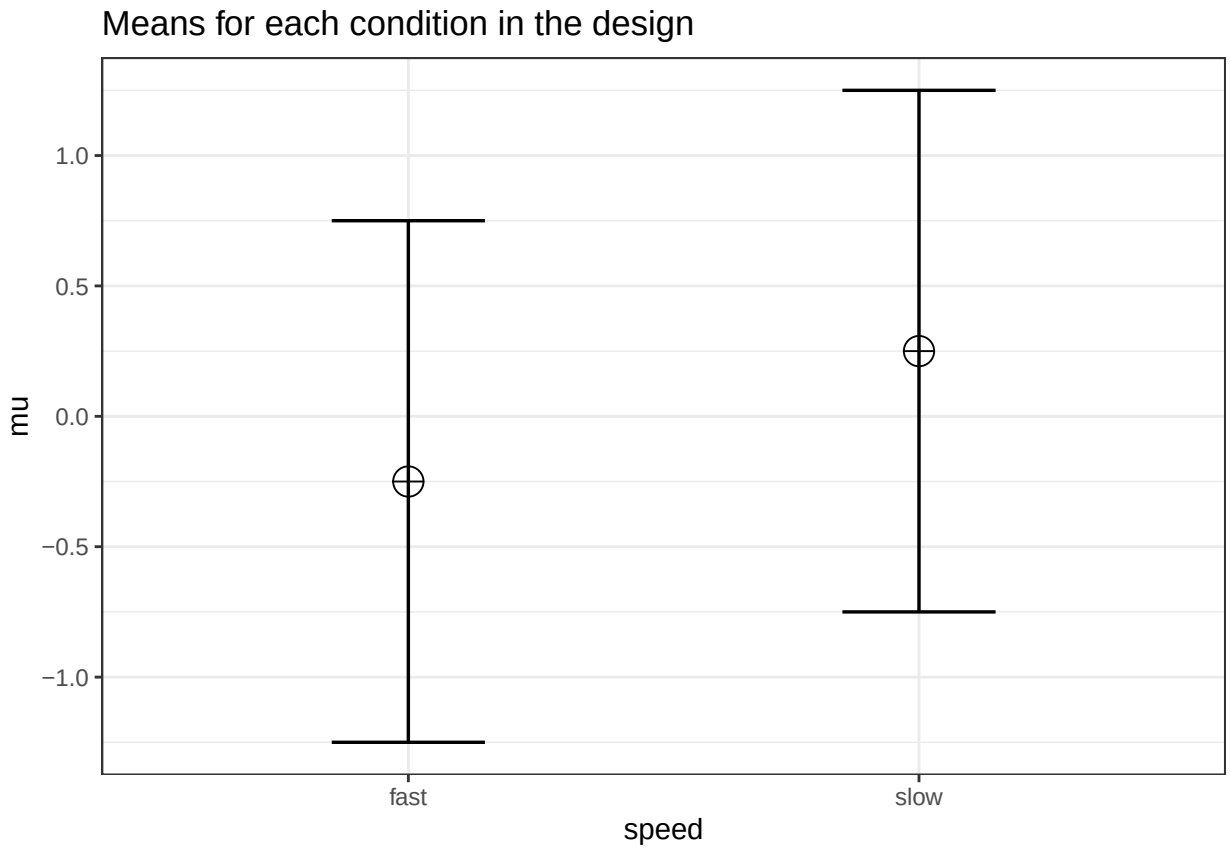
The results of the simulation are indeed very close to 80.777%. Note that the simulation calculates Cohen's dz effect sizes for paired comparisons - which here given the correlation of 0.5 is also 0.5 for a medium effect size.

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

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

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

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




```
alpha_level <- 0.05
design_result$sigmatrix
```

```
##      fast slow
## fast  1.0  0.7
## slow  0.7  1.0
```

```
ANOVA_power(design_result, nsims = nsims)
```

```
## Power and Effect sizes for ANOVA tests
##             power effect_size
## anova_speed    87      0.3235
##
## Power and Effect sizes for contrasts
##                          power effect_size
## p_speed_fast_speed_slow    87      0.6863
##
## Within-Subject Factors Included: Check MANOVA Results
```

```
#relation dz and f for within designs
f <- 0.5*0.6454972
# Entering this f in G*power, with a correlation of 0.5, yields the same as entering f = 0.25 and corre
```

Chapter 6

Repeated Measures-ANOVA Part 2

In a repeated measures design multiple observations are collected from the same participants. 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 & Delaney, 2004, p. 562, formula 47):

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

Where a is the number of within-subject levels.

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

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

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

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

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

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

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

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

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

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

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

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

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

I 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:

```
source("https://raw.githubusercontent.com/Lakens/ANOVA_power_simulation/master/calc_f_d_eta.R")
res <- calc_f_d_eta(mu = mu, sd = sd, variability = "maximum")
res$f
```

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

```
res$d
```

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

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

6.2 Three within conditions, medium effect size

We can perform power analyses for within designs using simuations. We set groups to 3 for the simulation, n = 20, and the correlation between dependent variables to 0.8. If the true effect size is f = 0.25, and the alpha level is 0.05, the power is 96.6%.

In this case, we simulate data with means -0.3061862, 0.0000000, and 0.3061862, and set the sd to 1.

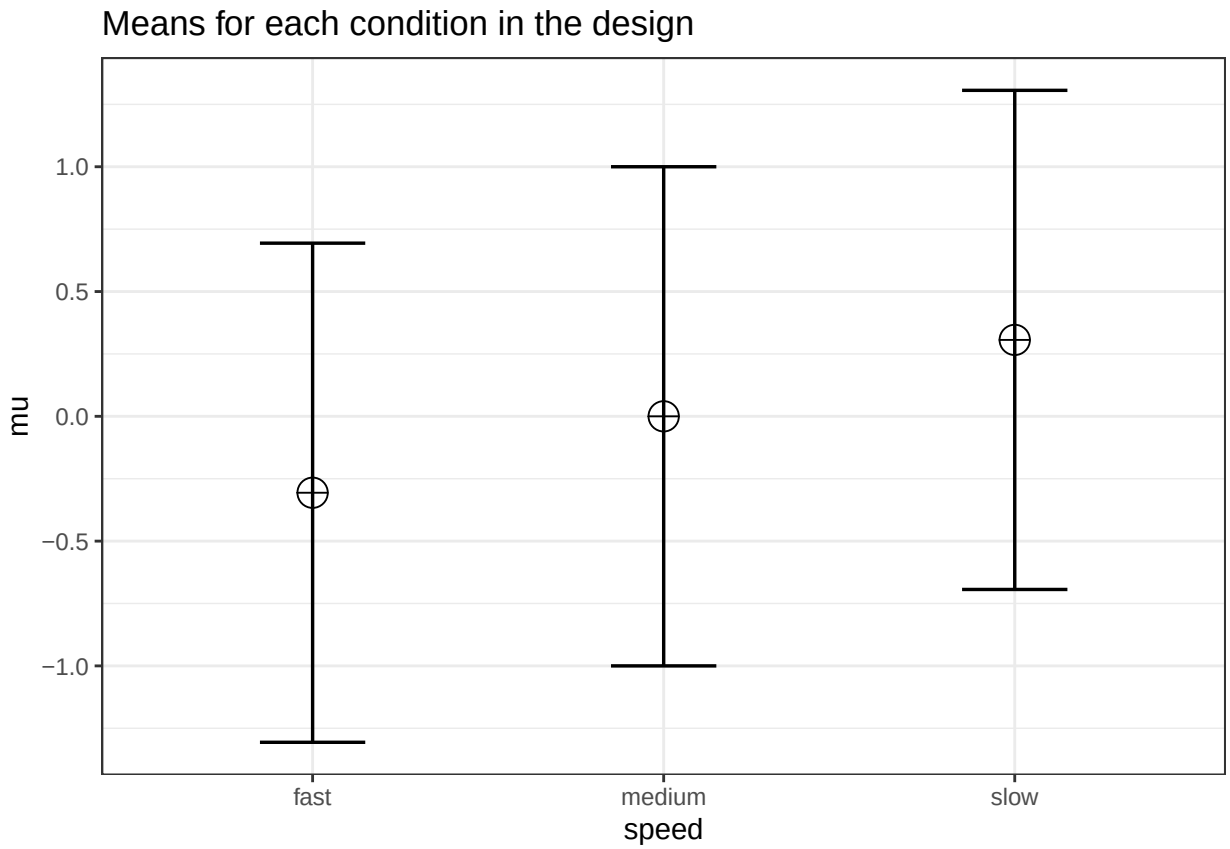
```
K <- 3
n <- 20
sd <- 1
r <- 0.8
alpha = 0.05
f <- 0.25
f2 <- f^2
ES <- f2/(f2+1)
ES
```

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

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

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

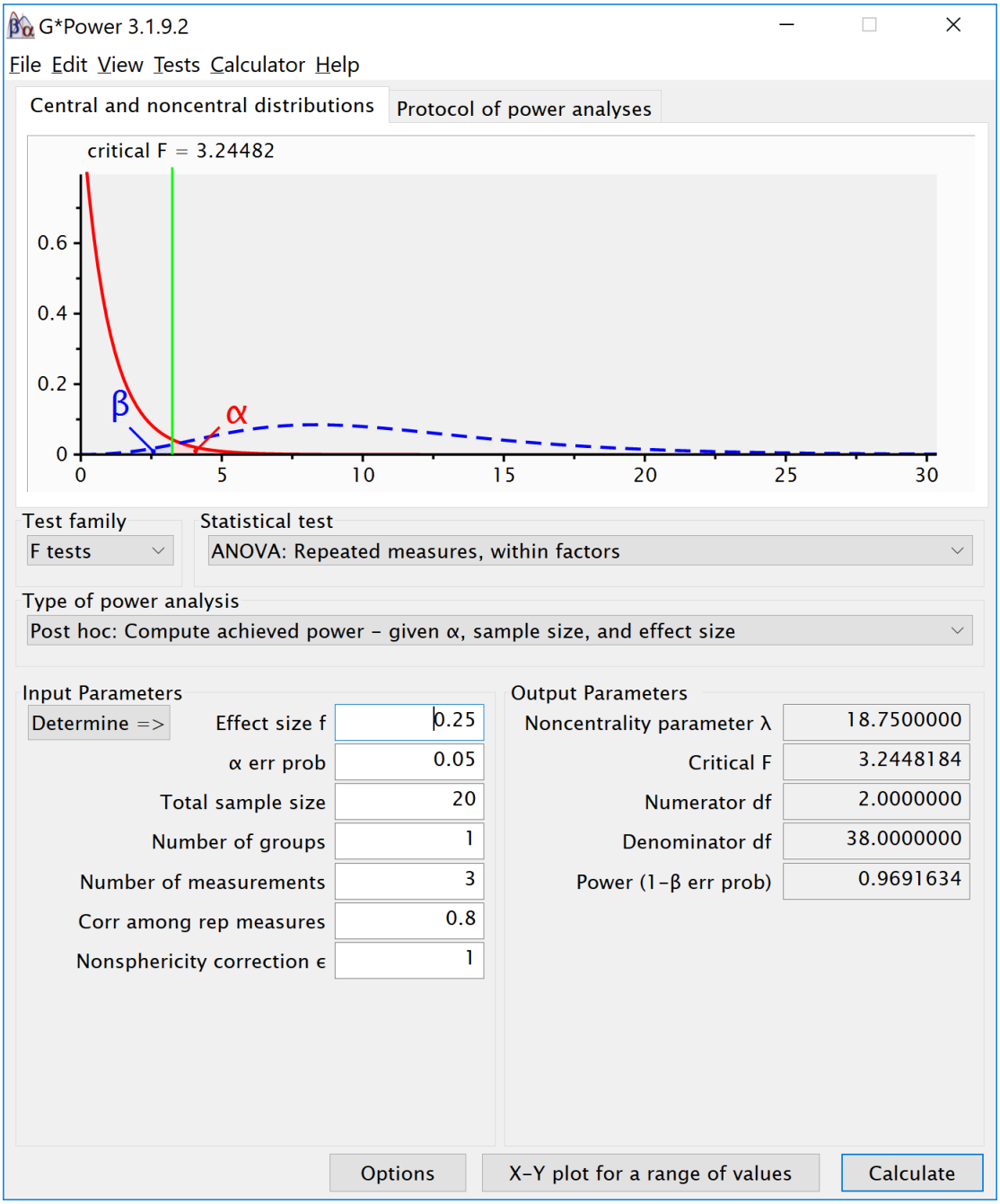
```
design =paste(K,"w",sep="")
labelnames <- c("speed", "fast", "medium", "slow")
design_result <- ANOVA_design(design = design,
                   n = n,
                   mu = mu,
                   sd = sd,
                   r = r,
                   labelnames = labelnames)
```



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

```
## Power and Effect sizes for ANOVA tests
##             power effect_size
## anova_speed    99      0.3507
##
## Power and Effect sizes for contrasts
##                            power effect_size
## p_speed_fast_speed_medium     60      0.5341
## p_speed_fast_speed_slow       99      1.0074
## p_speed_medium_speed_slow     57      0.4996
##
## Within-Subject Factors Included: Check MANOVA Results
```

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:

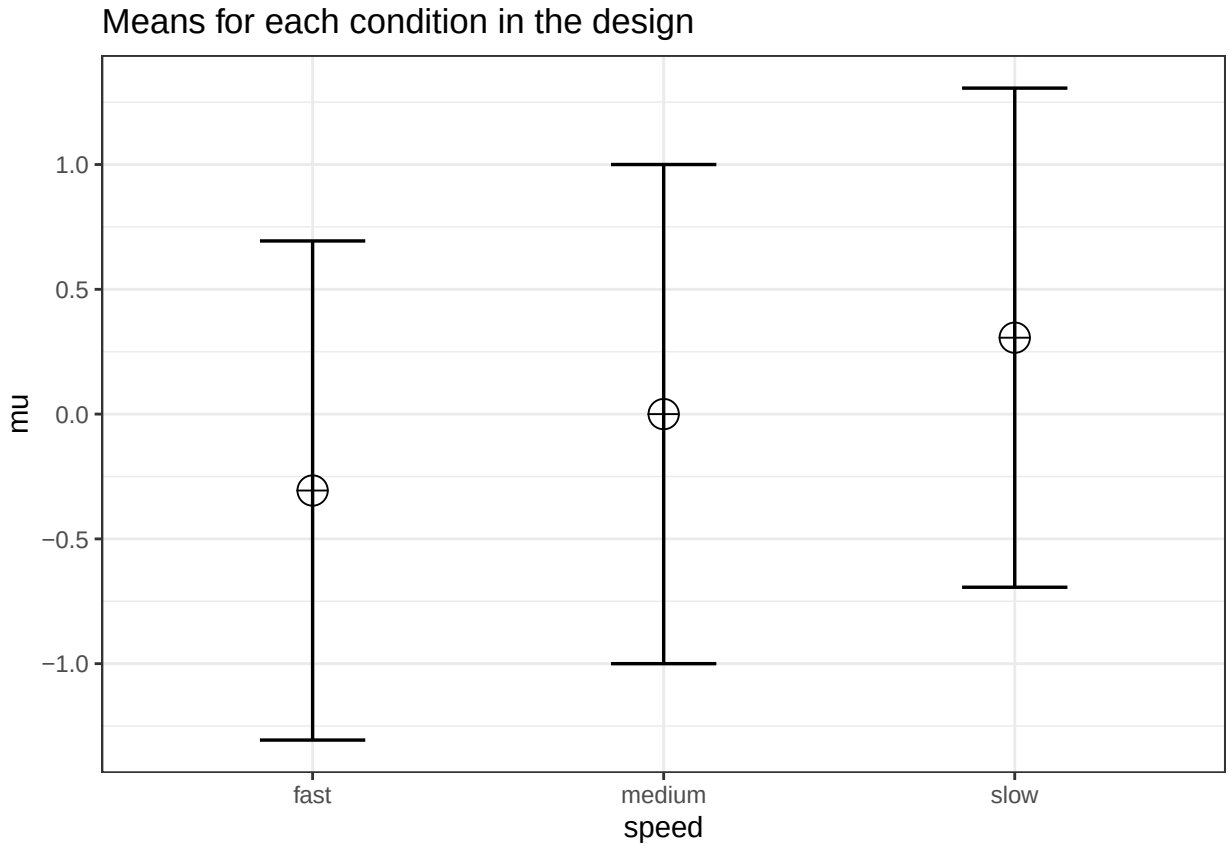


We can also validate this by creating the code to do a power analysis in R from scratch:

```
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] 0.9691634
```

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

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

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

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

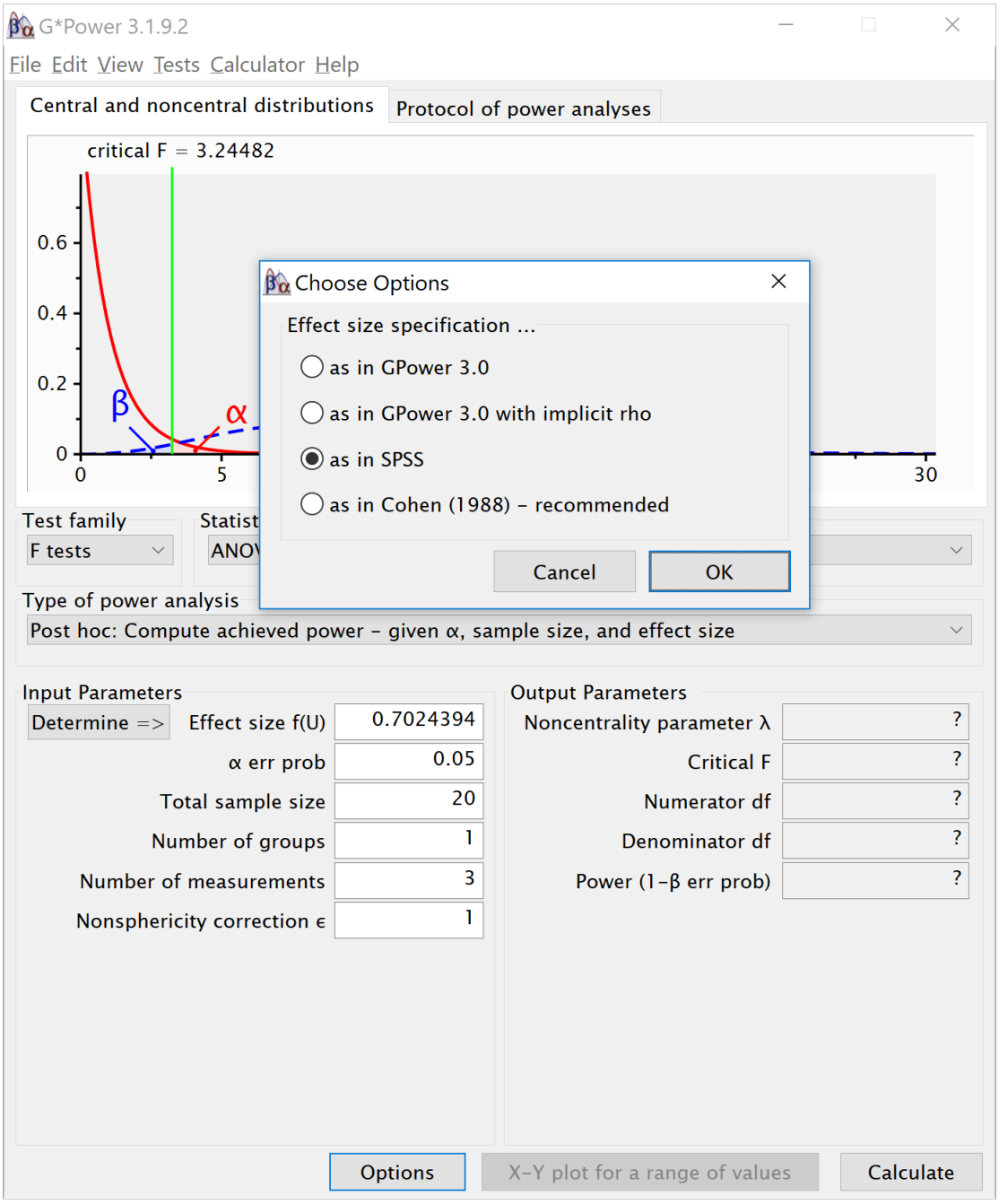
```
power_oneway_within(design_result)$Cohen_f
```

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

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

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

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


G*Power 3.1.9.2
File Edit View Tests Calculator Help
Central and noncentral distributions
Protocol of power analyses
critical F = 3.24482
0.6
0.4
0.2
0
β
α
0
5
30
Choose Options
Effect size specification ...
as in GPower 3.0
as in GPower 3.0 with implicit rho
as in SPSS
as in Cohen (1988) – recommended
Cancel
OK
Test family
F tests
Statist
ANOV
Type of power analysis
Post hoc: Compute achieved power – given α, sample size, and effect size
Input Parameters
Determine =>
Effect size f(U) 0.7024394
α err prob 0.05
Total sample size 20
Number of groups 1
Number of measurements 3
Nonsphericity correction ε 1
Output Parameters
Noncentrality parameter λ ?
Critical F ?
Numerator df ?
Denominator df ?
Power (1–β err prob) ?
Options
X-Y plot for a range of values
Calculate

Chapter 7

Repeated Measures-ANOVA 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 progress bar in case it takes a while
#pb <- winProgressBar(title = "progress bar", min = 0, max = nSim, width = 300)
# 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
# open library MASS
library('MASS')
# 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
 # setWinProgressBar(pb, i, title=paste(round(i/nSim*100, 1), "% done"))
  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
  }
#close(pb)#close progress bar
#printing all unique tests (adjusted code by DL)
sum(p1<alpha1)/nSim
```

```
## [1] 0.98
```

```
sum(p2<alpha2)/nSim
```

```
## [1] 0.95
```

```
sum(p3<alpha2)/nSim
```

```
## [1] 0.94
```

```
sum(p4<alpha2)/nSim
```

```
## [1] 0.02
```

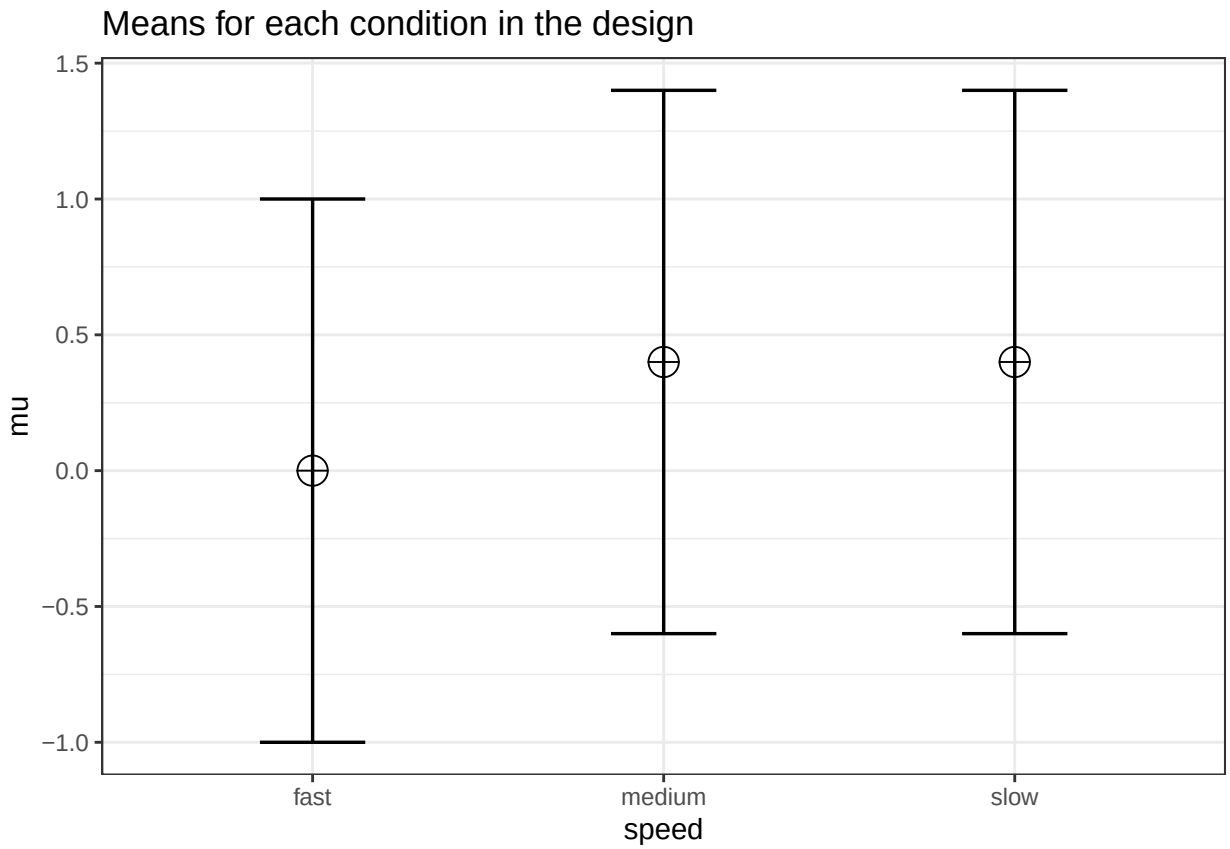
7.1 Reproducing Brysbaert

We can reproduce the same results as Brysbaeert 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)
```

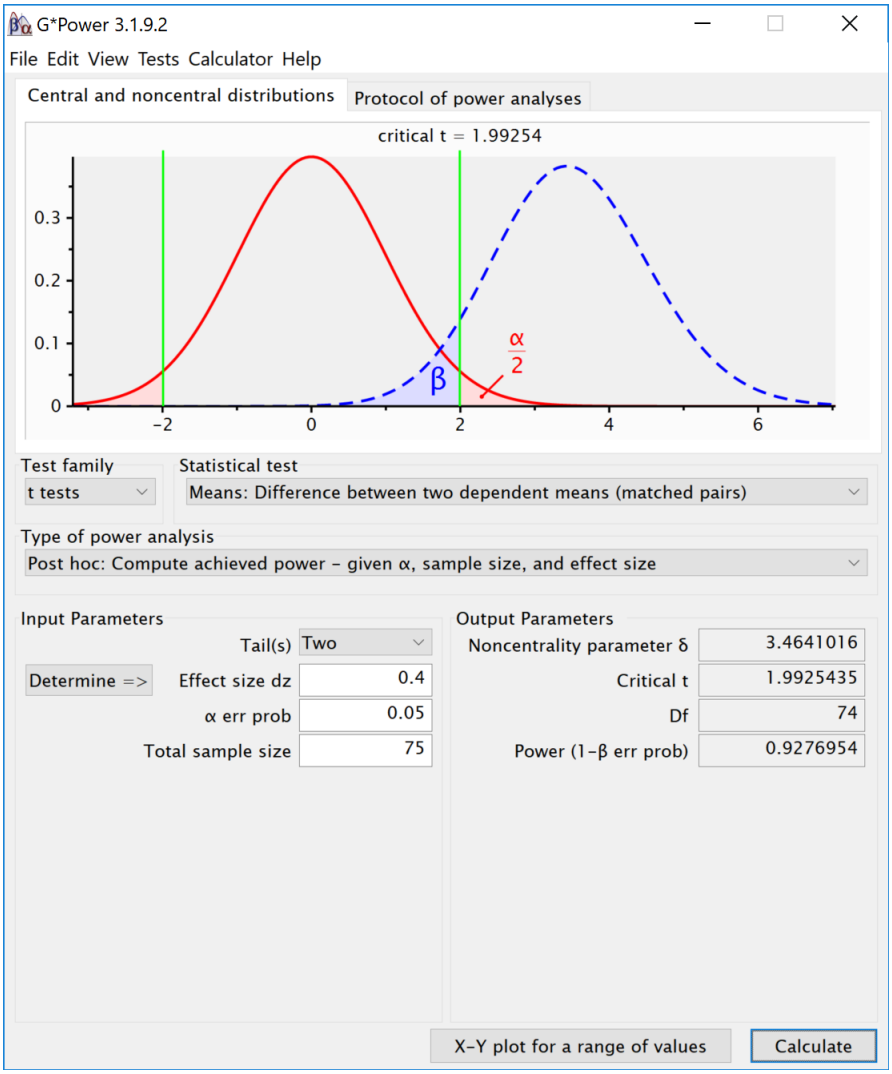



```
ANOVA_power(design_result, nsims = nsims)
```

```
## Power and Effect sizes for ANOVA tests
##             power effect_size
## anova_speed    94      0.1057
##
## Power and Effect sizes for contrasts
##                            power effect_size
## p_speed_fast_speed_medium     93    0.397304
## p_speed_fast_speed_slow       90    0.400646
## p_speed_medium_speed_slow      6    0.009926
##
## Within-Subject Factors Included: Check MANOVA Results
```

#Results

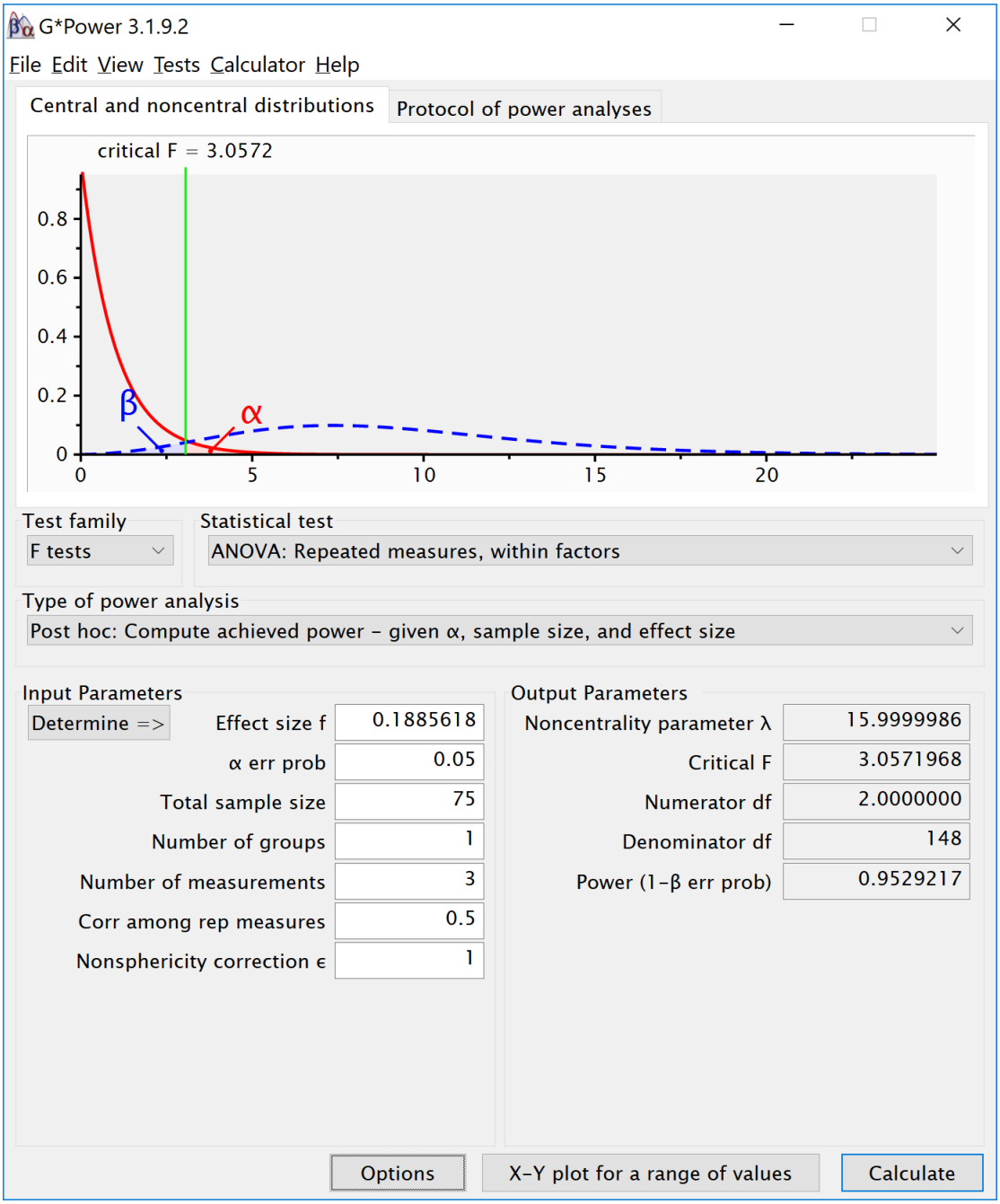
The results of the simulation are very similar. Power for the ANOVA F-test is around 95.2%. For the three paired t-tests, power is around 92.7. This is in line with the a-priori power analysis when using g*power:



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

```
# Our simulation is based onthe following means and sd:
# mu <- c(0, 0.4, 0.4)
# sd <- 1
f <- sqrt(sum((mu-mean(mu))^2)/length(mu))/sd #Cohen, 1988, formula 8.2.1 and 8.2.2
# We can see why f = 0.5*d.
# Imagine 2 group, mu = 1 and 2
# Grand mean is 1.5, we have sqrt(sum(0.5^2 + 0.5^2)/2), or sqrt(0.5/2), = 0.5.
# For Cohen's d we use the difference, 2-1 = 1.
```

The Cohen´s f is 0.1885618. We can enter the f (using the default 'as in G*Power 3.0' in the option window) and enter a sample size of 75, number of groups as 1, number of measurements as 3, correlation as 0.5. This yields:



7.2 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
```

```r
# 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 progress bar in case it takes a while
#pb <- winProgressBar(title = "progress bar", min = 0, max = nSim, width = 300)
# 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
# open library MASS
library('MASS')
# 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
 # setWinProgressBar(pb, i, title=paste(round(i/nSim*100, 1), "% done"))
  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
  }
#close(pb)#close progress bar
#printing all unique tests (adjusted code by DL)
sum(p1<alpha1)/nSim
```

```
## [1] 0.99
```

```r
sum(p2<alpha2)/nSim
```

```
## [1] 0.95
```

```r
sum(p3<alpha2)/nSim
```

```
## [1] 0.98
```

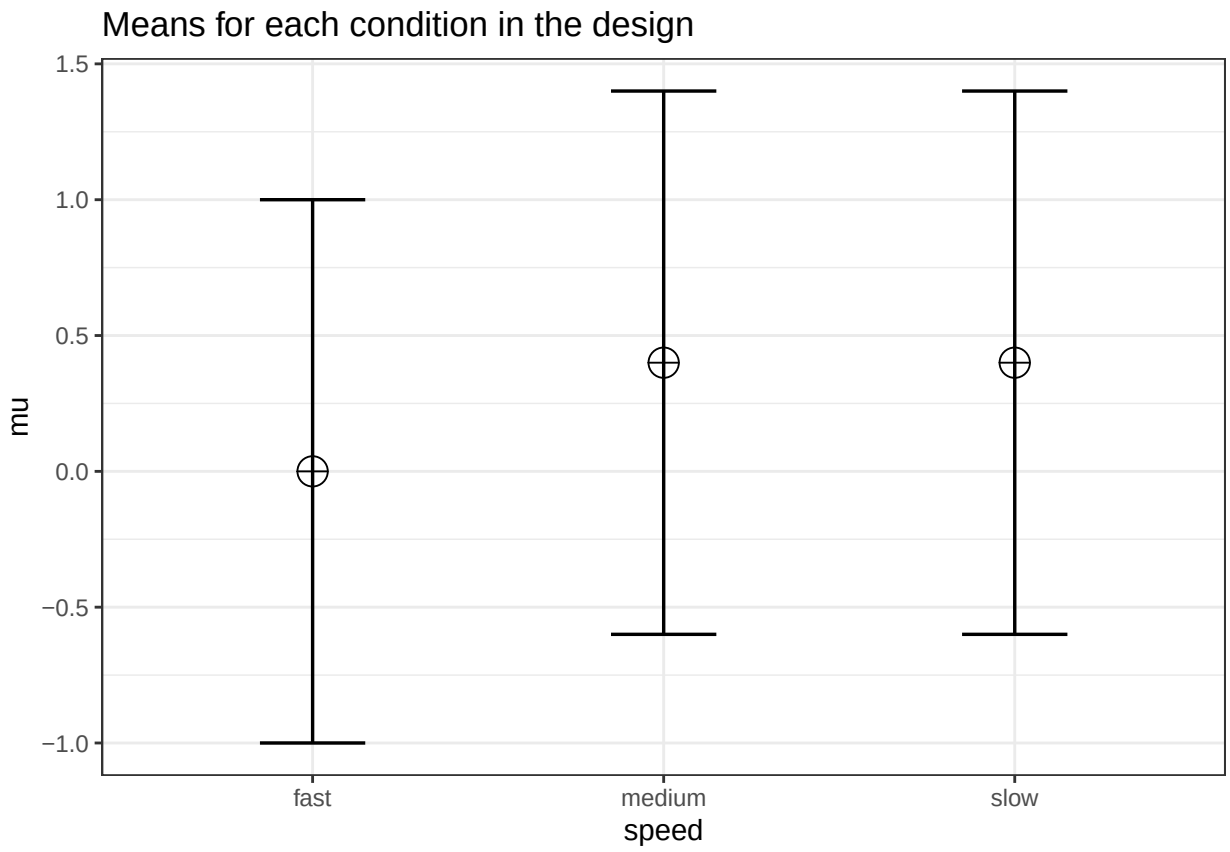
```r
sum(p4<alpha2)/nSim
```

```
## [1] 0.07
```

```
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 within design, and run the simulation

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

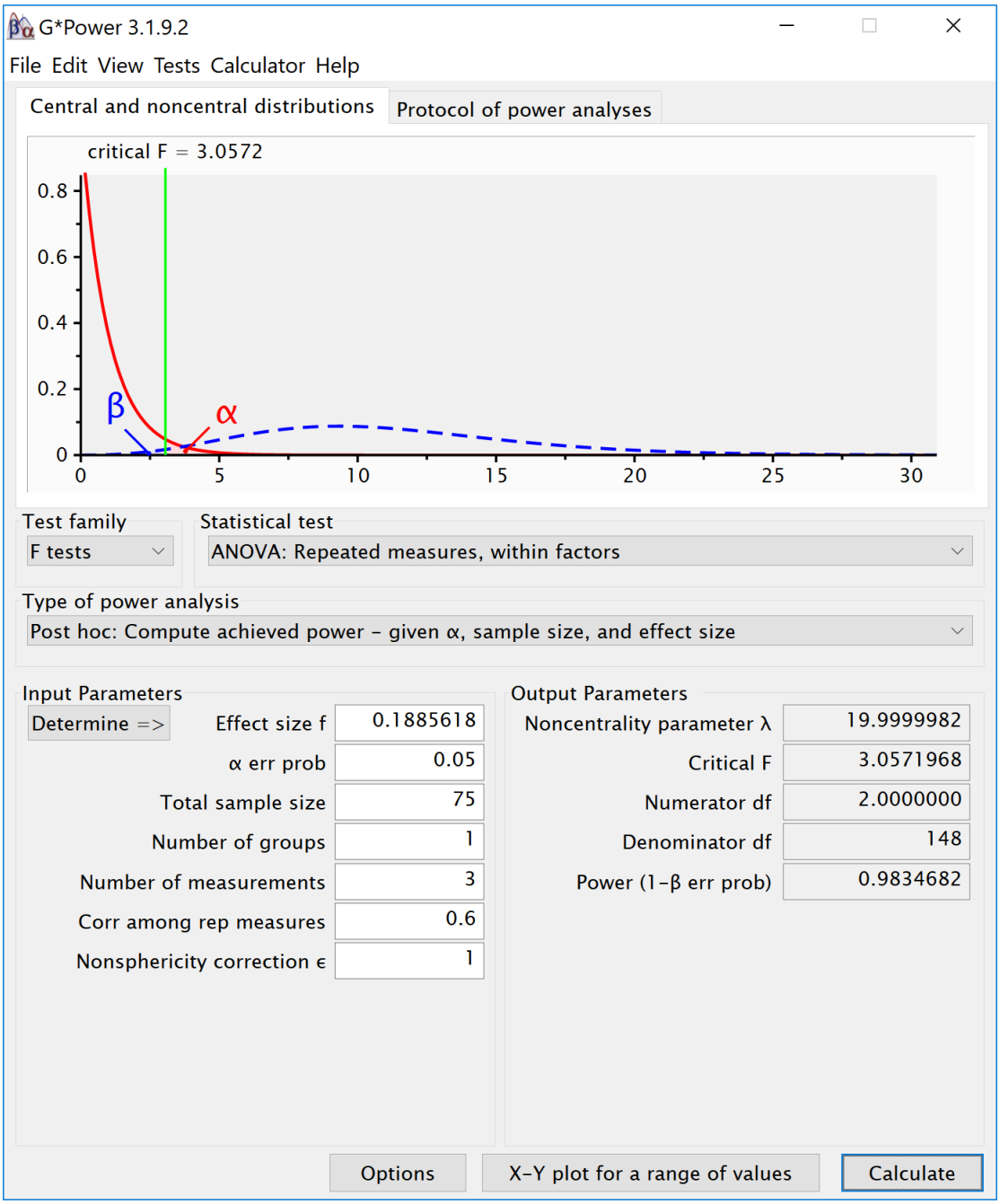



```
ANOVA_power(design_result, nsims = nsims)
```

```
## Power and Effect sizes for ANOVA tests
##             power effect_size
## anova_speed   100      0.1247
##
## Power and Effect sizes for contrasts
##                           power effect_size
```

```
## p_speed_fast_speed_medium    97   0.4462847
## p_speed_fast_speed_slow      98   0.4479337
## p_speed_medium_speed_slow     5   0.0005669
##
## Within-Subject Factors Included: Check MANOVA Results
```

Again, this is similar to g*power for the ANOVA:

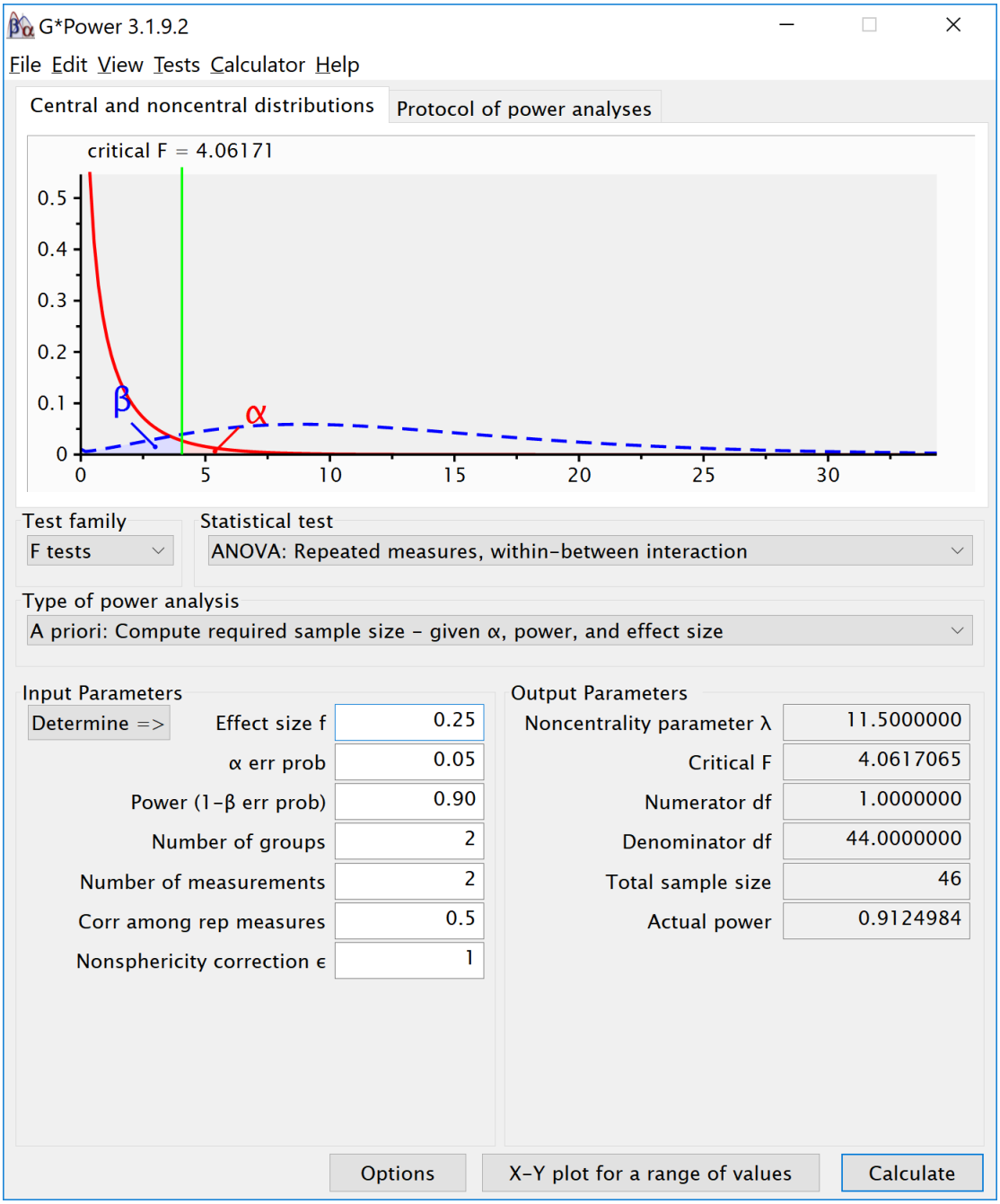


Chapter 8

Mixed ANOVA Part 1

8.1 Two by two ANOVA, within-between design

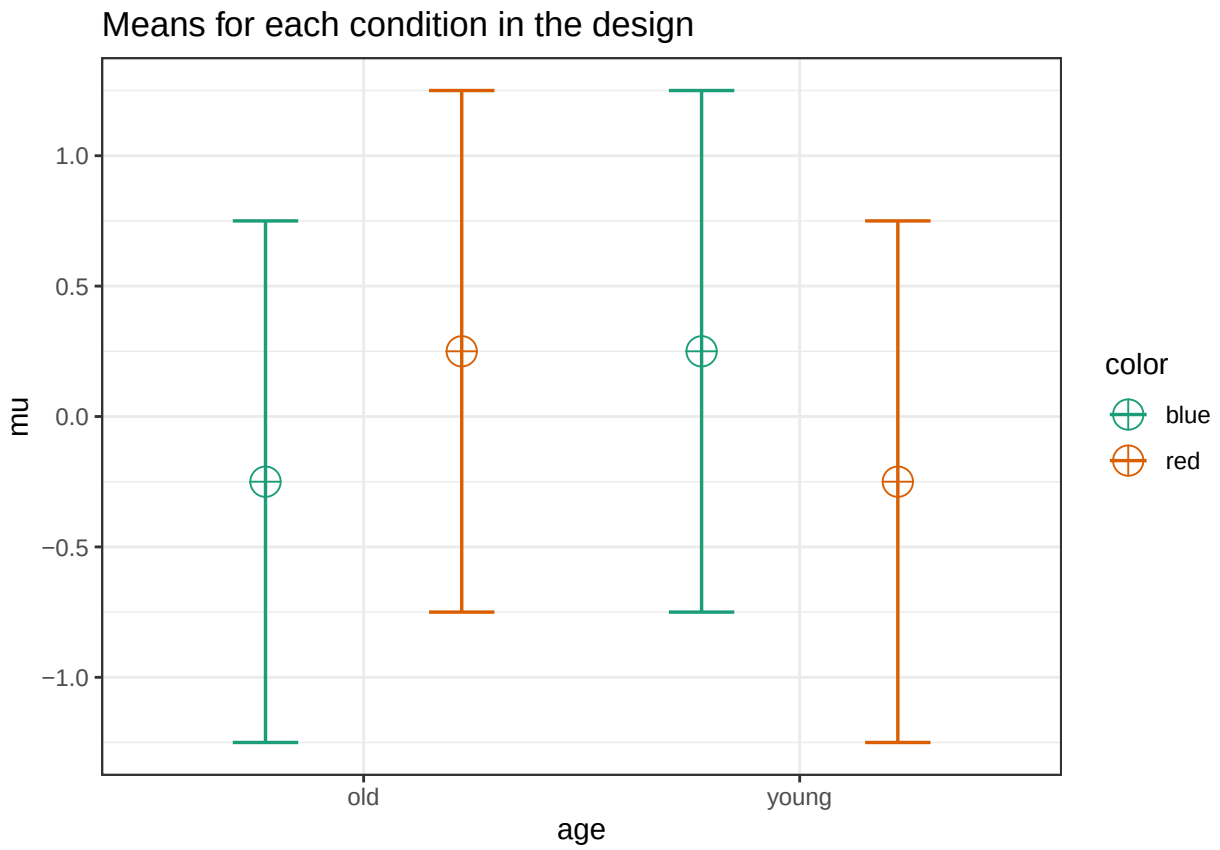
We can simulate a Two-Way ANOVA with a specific alpha, sample size and effect size, to achieve a specified statistical power. We wil try to reproduce the power analysis by g*power for an F-test, ANOVA: Repeated measures, within-between interaction.



For the 2-way interaction, the result should be a power of 91.25% is we have a total samplesize of 46. Since we have 2 groups in the between factor that means the sample size per group is 2 (and both these groups collect 2 repeated measures).

```
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)
```

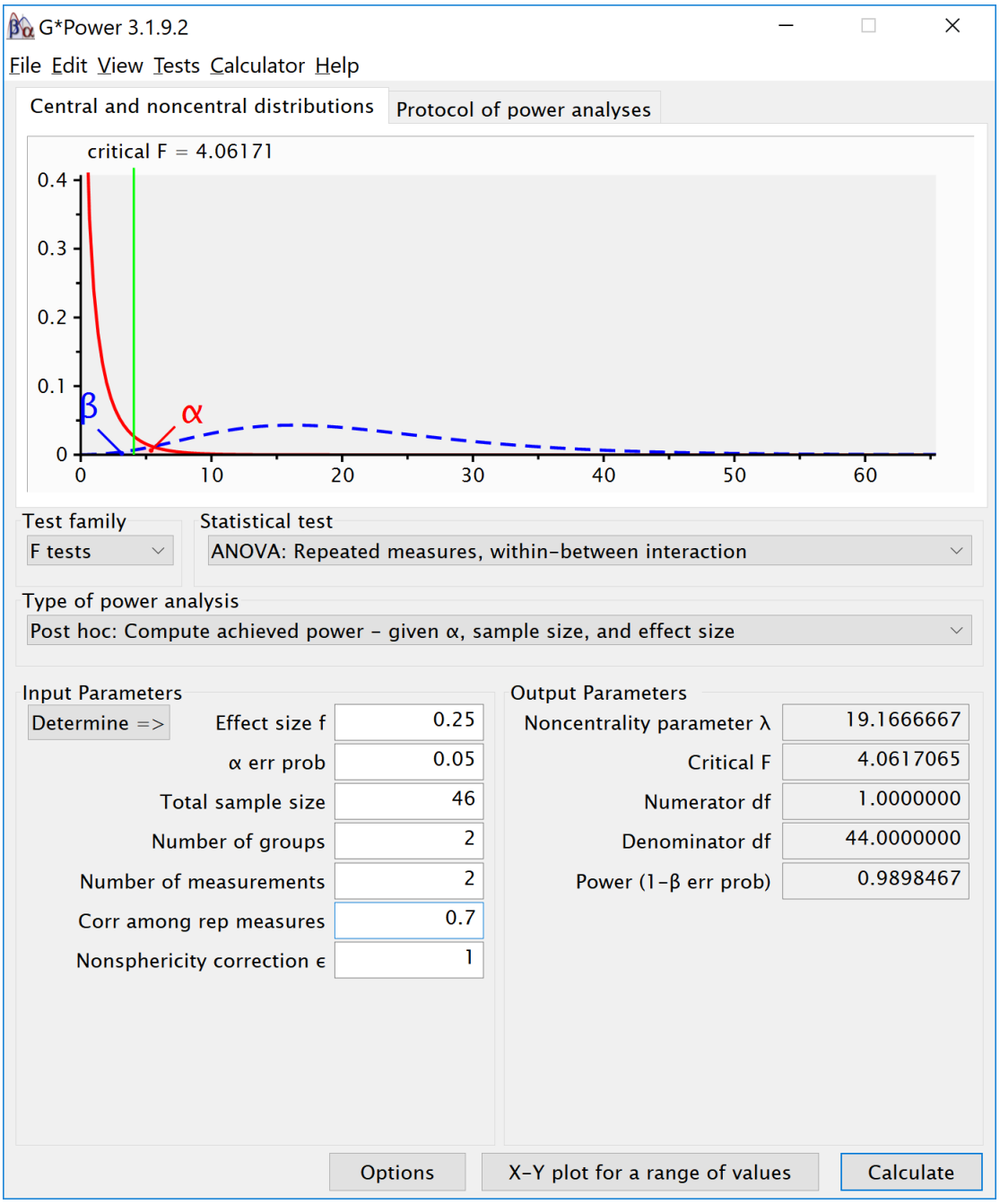



```
simulation_result <- ANOVA_power(design_result, alpha = 0.05, nsims = nsims)
```

```
## Power and Effect sizes for ANOVA tests
##                 power effect_size
## anova_color         7     0.02477
## anova_age           5     0.01691
## anova_color:age    93     0.22241
##
## Power and Effect sizes for contrasts
##                                           power effect_size
## p_age_old_color_blue_age_old_color_red       40     0.54317
## p_age_old_color_blue_age_young_color_blue    63     0.51745
## p_age_old_color_blue_age_young_color_red      2     0.01617
## p_age_old_color_red_age_young_color_blue      8    -0.02424
## p_age_old_color_red_age_young_color_red      71    -0.53890
## p_age_young_color_blue_age_young_color_red   40    -0.50118
##
## Within-Subject Factors Included: Check MANOVA Results
```

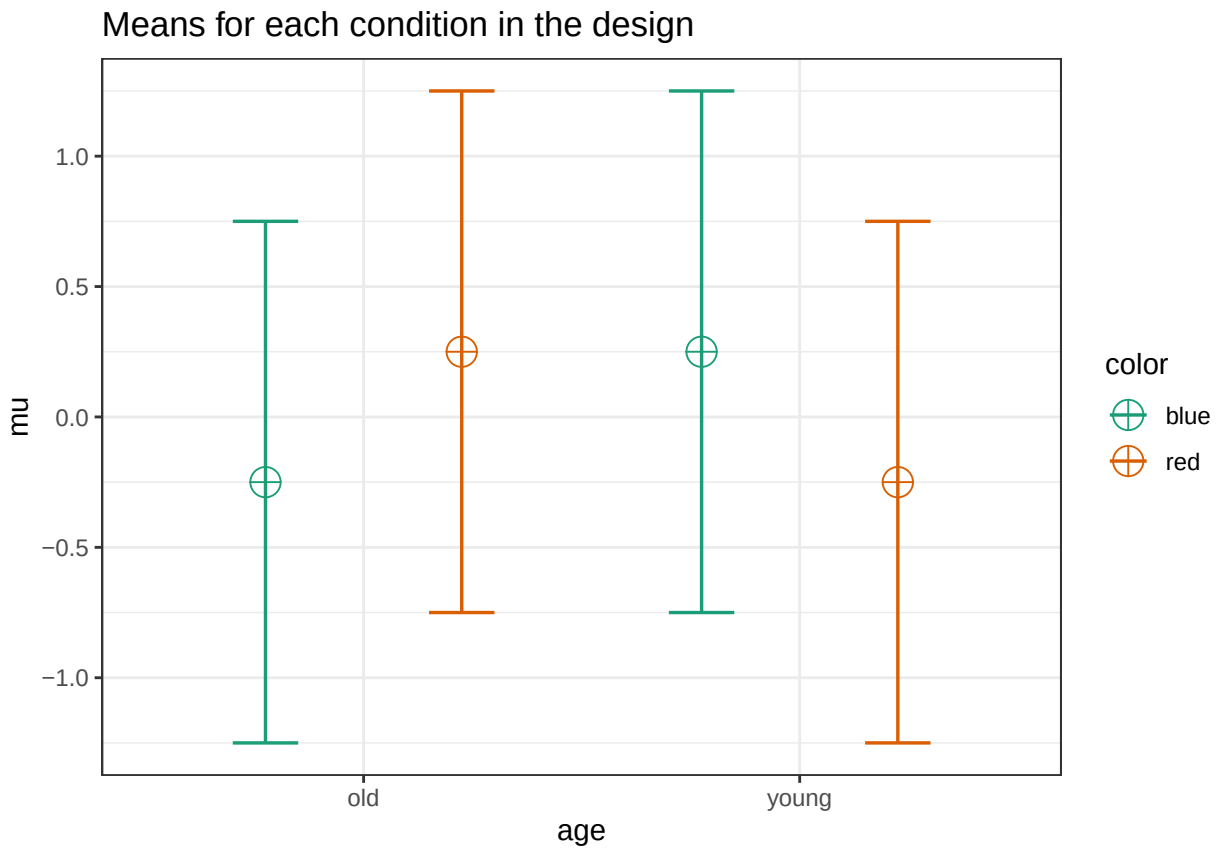
8.2 Two by two ANOVA, within-between design Variation 1

We can simulate the same Two-Way ANOVA increasing the correlation to 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 = 0.05, nsims = nsims)
```

```
## Power and Effect sizes for ANOVA tests
##                 power effect_size
## anova_color         7     0.02453
## anova_age           5     0.02467
## anova_color:age    98     0.31881
##
## Power and Effect sizes for contrasts
##                                              power effect_size
## p_age_old_color_blue_age_old_color_red          43     0.55101
## p_age_old_color_blue_age_young_color_blue       81     0.70382
## p_age_old_color_blue_age_young_color_red         8     0.04100
## p_age_old_color_red_age_young_color_blue         5    -0.01757
## p_age_old_color_red_age_young_color_red         85    -0.66518
## p_age_young_color_blue_age_young_color_red      35    -0.48696
##
## Within-Subject Factors Included: Check MANOVA Results
```

Chapter 9

Mixed ANOVA Part 2

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

We can reproduce the simulation coded by Ben Amsel

```
knitr::opts_chunk$set(echo=TRUE, warning=FALSE, message=FALSE)

# define the parameters
mu = c(700, 670, 670, 700) # true effects (in this case, a double dissociation)
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 parameters mu and Sigma created earlier
# and bind to the existing df

require(MASS)
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

require(dplyr)
require(car)
require(broom)

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 = 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) = c('X1','X2','Interaction')
```

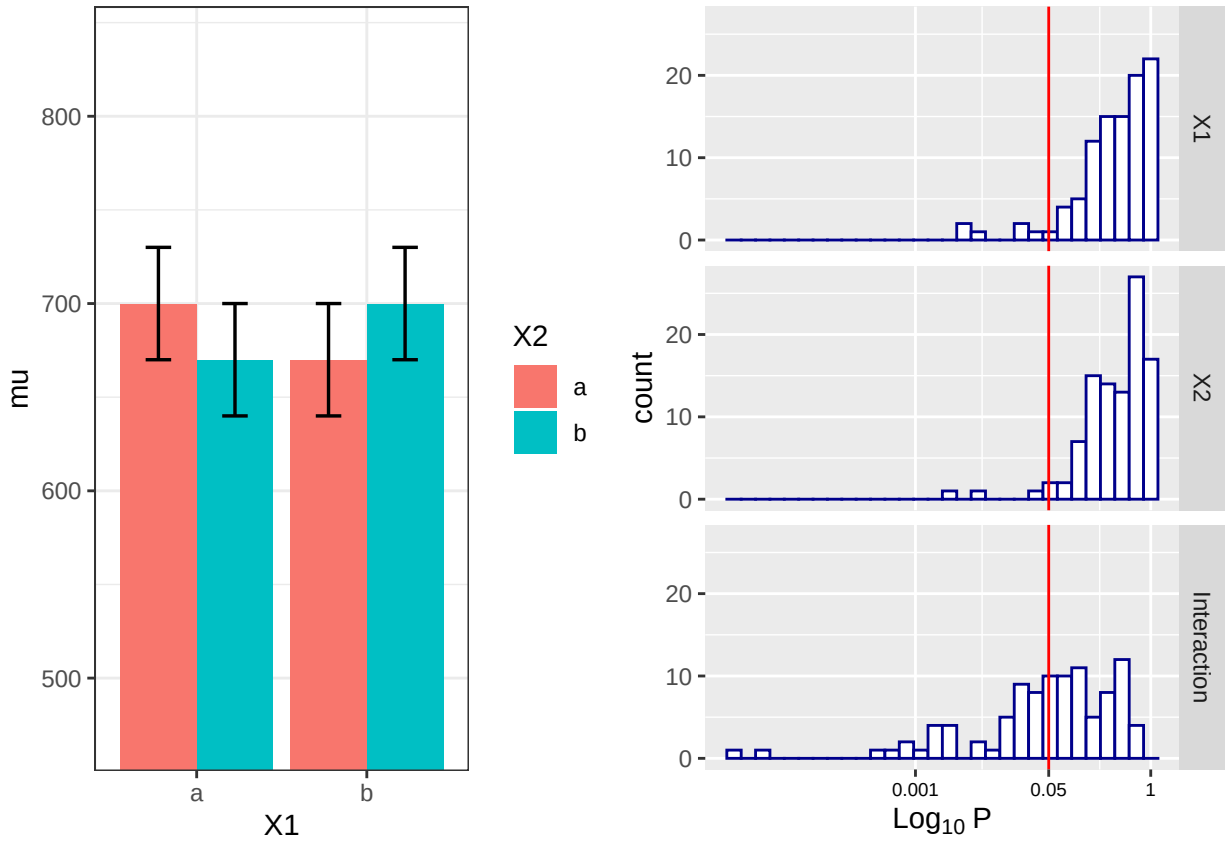
The empirical power is easy to compute, it's just the proportion of simulation runs where p $<$. 05.

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

```
##          X1          X2 Interaction
##           6           4          45
```

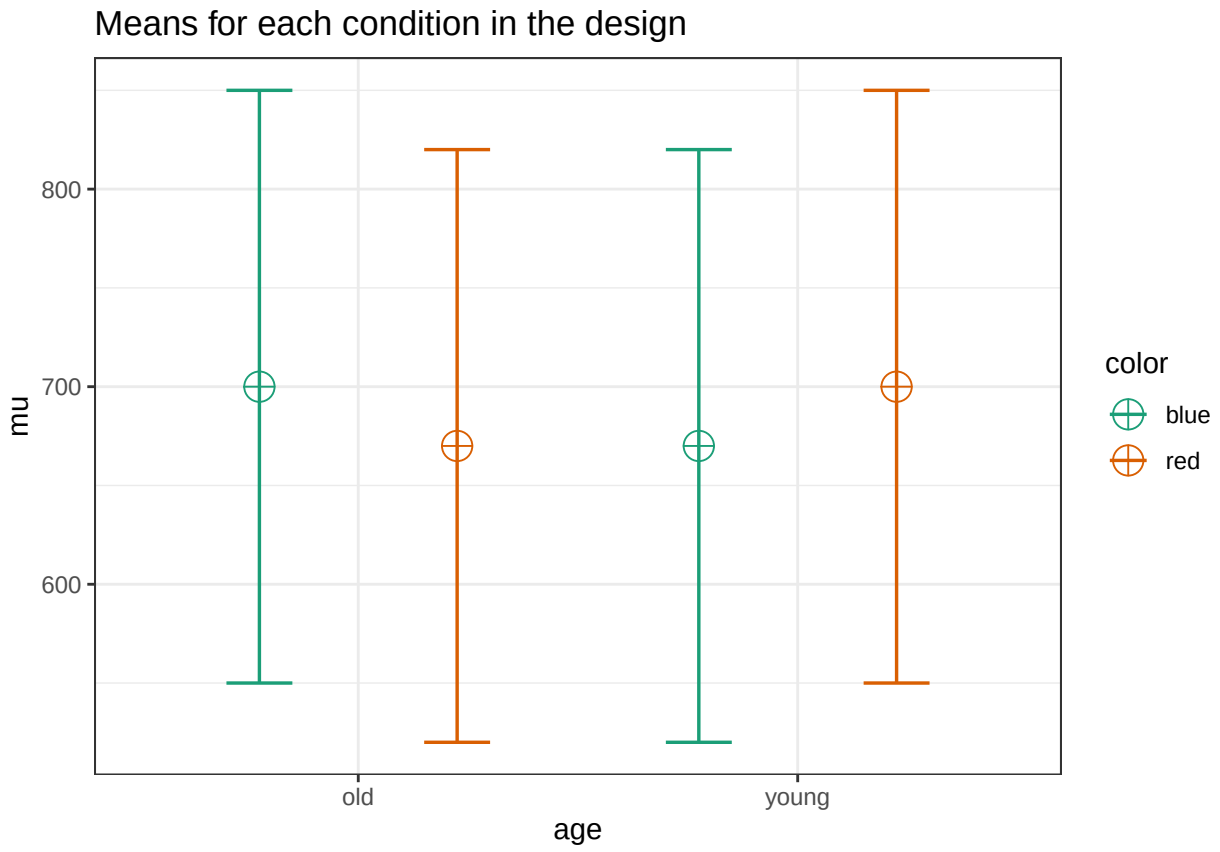
Visualize the distributions of p-values

```
# plot the known effects
require(ggplot2)
require(gridExtra)

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
require(reshape2)
plotData <- melt(p, 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:

```
mu = c(700, 670, 670, 700) # true effects (in this case, a double dissociation)
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 = 0.05, nsims = nsims)
```

```
## Power and Effect sizes for ANOVA tests
##                 power effect_size
## anova_age           6     0.04097
## anova_color         8     0.04684
## anova_age:color    49     0.16492
##
## Power and Effect sizes for contrasts
##                                           power effect_size
## p_age_old_color_blue_age_old_color_red       37    -0.30464
## p_age_old_color_blue_age_young_color_blue    29    -0.27634
## p_age_old_color_blue_age_young_color_red      4     0.00226
## p_age_old_color_red_age_young_color_blue      8     0.03465
## p_age_old_color_red_age_young_color_red      27     0.31187
## p_age_young_color_blue_age_young_color_red   26     0.26932
##
## Within-Subject Factors Included: Check MANOVA Results
```

The simulations yield closely matching results.

9.2 Examine variation of means and correlation

```
# define the parameters
mu = c(700, 670, 690, 750) # true effects (in this case, a double dissociation)
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 parameters mu and Sigma created earlier
# and bind to the existing df

require(MASS)
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
```

```
require(dplyr)
require(car)
require(broom)

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 = 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) = c('X1','X2','Interaction')
```

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

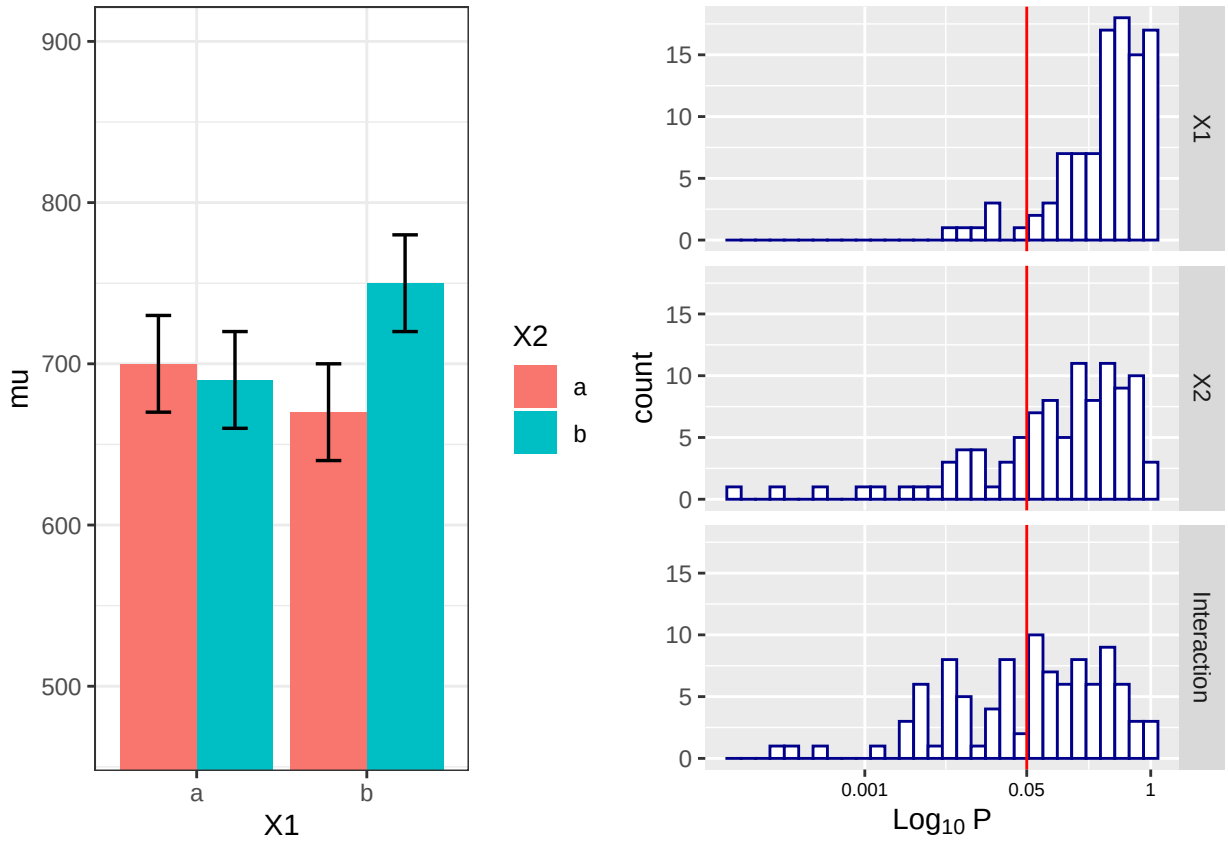
```
power.res = apply(as.matrix(p), 2,
  function(x) round(mean(ifelse(x < .05, 1, 0) * 100),2))
power.res
```

```
##          X1          X2 Interaction
##           7          28          42
```

Visualize the distributions of p-values

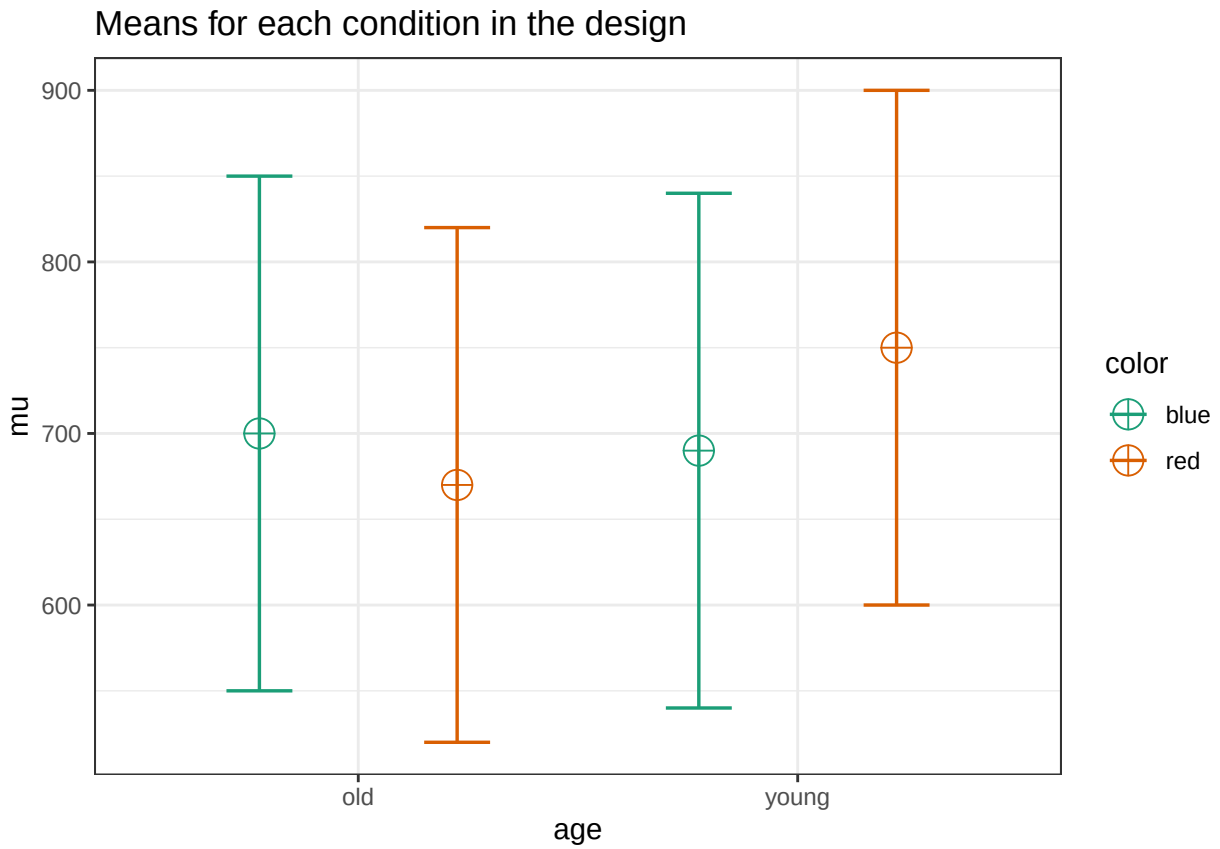
```
# plot the known effects
require(ggplot2)
require(gridExtra)

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
require(reshape2)
plotData <- melt(p, 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:

```
mu = c(700, 670, 690, 750) # true effects (in this case, a double dissociation)
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 = 0.05, nsims = nsims)
```

```
## Power and Effect sizes for ANOVA tests
##                 power effect_size
## anova_age          28     0.10912
## anova_color         6     0.05297
## anova_age:color    52     0.17050
##
## Power and Effect sizes for contrasts
##                                           power effect_size
## p_age_old_color_blue_age_old_color_red       18    -0.20444
## p_age_old_color_blue_age_young_color_blue     8    -0.07423
## p_age_old_color_blue_age_young_color_red     29     0.30459
## p_age_old_color_red_age_young_color_blue      7     0.12526
## p_age_old_color_red_age_young_color_red      60     0.49868
## p_age_young_color_blue_age_young_color_red   39     0.38081
##
## Within-Subject Factors Included: Check MANOVA Results
```

Chapter 10

Mixed ANOVA Part 3

10.1 Two by two ANOVA, within design

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

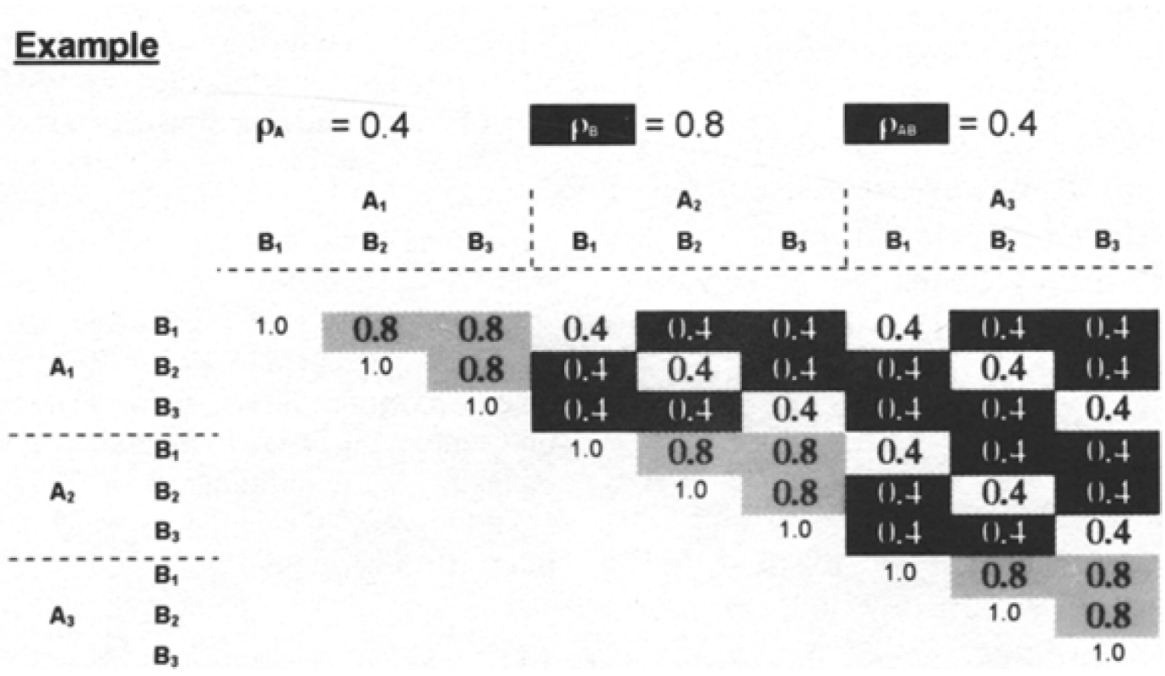


Figure 1. Representation of a correlation matrix for a 3 (A) × 3 (B) RM ANOVA: General form and numeric example. ρ_A and ρ_B represent the average correlation among the A and B (pooled) trials, respectively, and ρ_{AB} represents the average correlation among the AB coefficients having dissimilar levels.

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

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

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

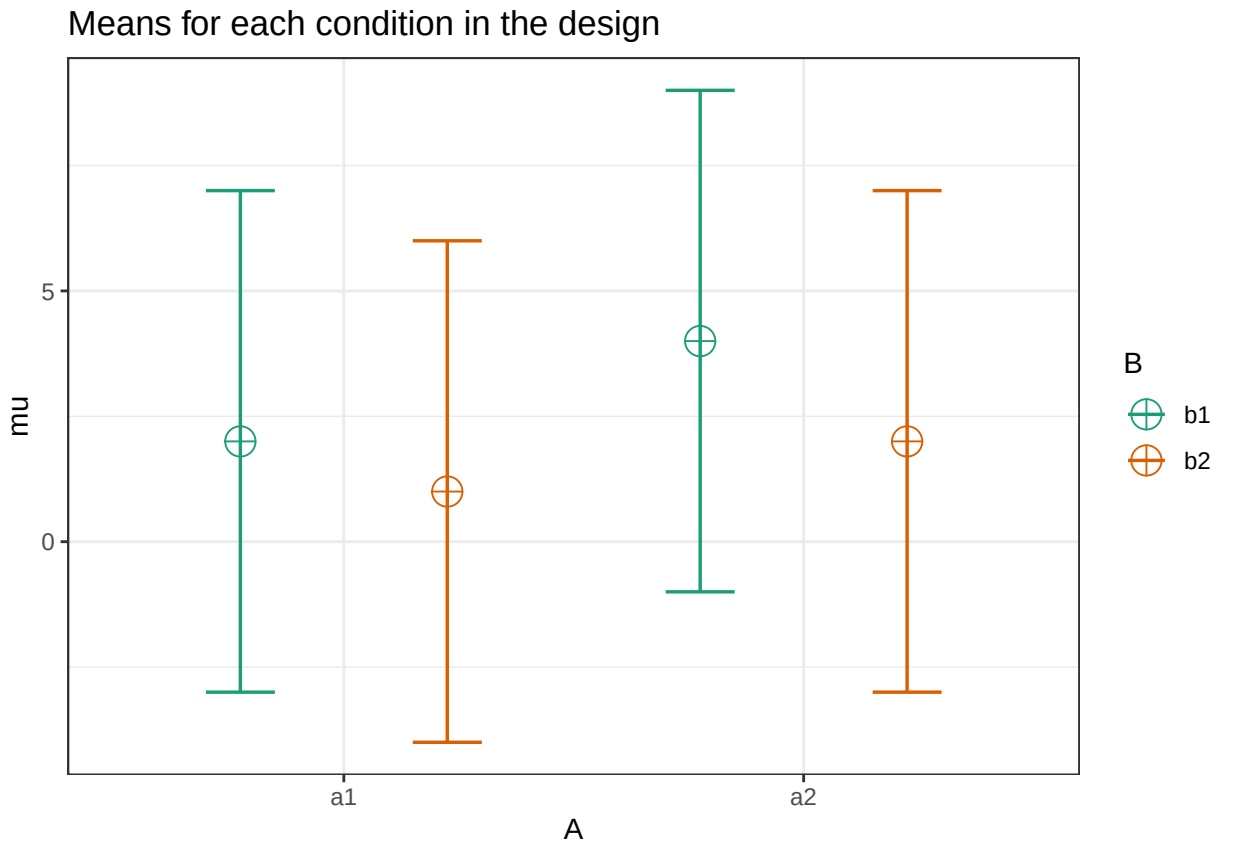
For the interaction between A and B: $\sigma_e^2 = \sigma^2(1 - \rho_{\max}) - \sigma^2(\overline{\rho}_{\min} - \overline{\rho}_{AB})$

10.2 Simple example: 2x2 within design

It is difficult to just come up with a positive definite covariance matrix. The best way to achieve this is to get the correlations from a pilot study. Indeed, it should be rather difficult to know which correlations to fill in without some pilot data.

We try to get the formulas in Potvin and Schutz (2000) working. **Below, I manage for the main effects, but not for the interaction**.

```
mu = c(2,1,4,2)
n <- 20
sd <- 5
r <- c(
  0.8, 0.4, 0.4,
       0.4, 0.4,
            0.8
  )
string = "2w*2w"
alpha_level <- 0.05

labelnames = c("A", "a1", "a2", "B", "b1", "b2")
design_result <- ANOVA_design(design = string,
                              n = n,
                              mu = mu,
                              sd = sd,
                              r = r,
                              labelnames = labelnames)
```




```
simulation_result <- ANOVA_power(design_result, alpha = 0.05, nsims = 1000)
```

```
## Power and Effect sizes for ANOVA tests
##           power effect_size
## anova_A    28.4     0.13259
## anova_B    77.5     0.32575
## anova_A:B  15.5     0.09107
##
## Power and Effect sizes for contrasts
##                         power effect_size
## p_A_a1_B_b1_A_a1_B_b2  26.5    -0.32449
## p_A_a1_B_b1_A_a2_B_b1  37.7     0.40289
## p_A_a1_B_b1_A_a2_B_b2   6.9     0.02044
## p_A_a1_B_b2_A_a2_B_b1  67.4     0.59278
## p_A_a1_B_b2_A_a2_B_b2  14.6     0.20703
## p_A_a2_B_b1_A_a2_B_b2  74.6    -0.65794
##
## Within-Subject Factors Included: Check MANOVA Results
```

Result simulation after 100000 simulations

simulation_result <- ANOVA_power(design_result, alpha = 0.05, nsims = 100000) Power and Effect sizes for ANOVA tests power effect size anova_A 26.849 0.0984 anova_B 64.091 0.2452 anova_A:B 26.875 0.0983

Power and Effect sizes for contrasts power effect size p_A_a1_B_b1_A_a1_B_b2 27.052 -0.3298 p_A_a1_B_b1_A_a2_B_b1 39.637 0.4162 p_A_a1_B_b1_A_a2_B_b2 4.983 -0.0005 p_A_a1_B_b2_A_a2_B_b1 64.252 0.5699 p_A_a1_B_b2_A_a2_B_b2 13.479 0.2077 p_A_a2_B_b1_A_a2_B_b2 76.622 -0.6597

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

```
k <- 1 #one group (because all factors are within)
rho_A <- 0.5 #mean r for factor A
rho_B <- 0.8 #mean r for factor B
rho_AB <- 0.4 #mean r for factor AB
alpha <- 0.05
sigma <- sd
m_A <- 2 #levels factor A
variance_e_A <- sigma^2 * (1 - rho_A) + sigma^2 * (m_A - 1) * (rho_B - rho_AB) #Variance A
variance_e_A
```

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

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

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

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

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

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

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

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

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

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

```
          df2,
          lambda_A,
          lower.tail = FALSE)
lambda_B <- n * m_B * sum((colMeans(mean_mat) - mean(colMeans(mean_mat))) ^ 2) / variance_e_B
lambda_B
```

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

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

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

```
pow_B
```

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

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

```
#This (or the variance calculation above) does not work.
lambda_AB <- n * sum((
  mean_mat - rowMeans(mean_mat) - colMeans(mean_mat) + mean(mean_mat)
  ) ^ 2) / variance_e_AB
lambda_AB
```

```
## [1] 38
```

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

```
          df1,
          df2,
          lambda_AB,
          lower.tail = FALSE)
pow
```

```
## [1] 0.9999458
```

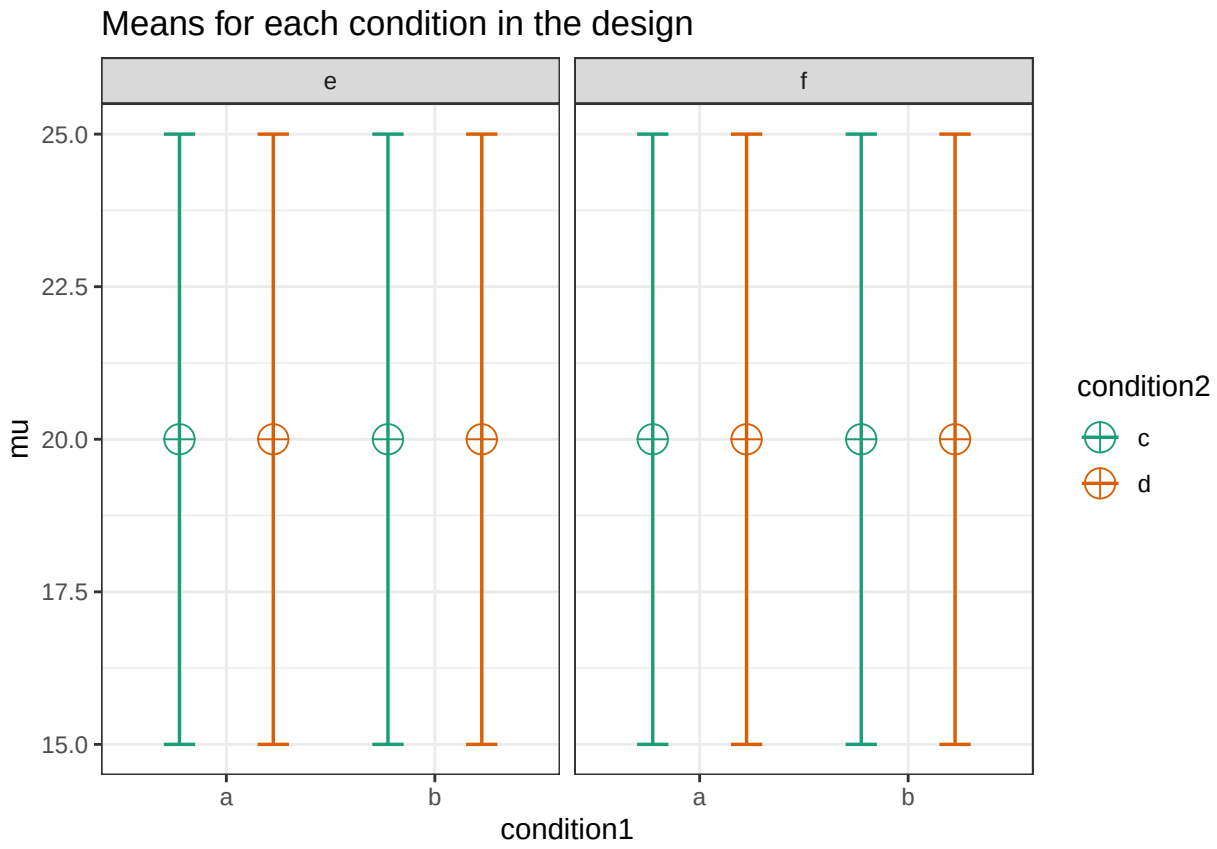
Maybe the simulation is not correct for the interaction, or the formula is not correctly programmed.

Chapter 11

Error Control in Exploratory ANOVA

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

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

```
## Power and Effect sizes for ANOVA tests
##                                       power effect_size
## anova_condition1                          5    0.002715
## anova_condition2                          4    0.002567
## anova_condition3                          2    0.002612
## anova_condition1:condition2               1    0.001701
## anova_condition1:condition3               6    0.002686
## anova_condition2:condition3               6    0.002528
## anova_condition1:condition2:condition3     3    0.001849
##
## Power and Effect sizes for contrasts
##                                                                                 power
## p_condition1_a_condition2_c_condition3_e_condition1_a_condition2_c_condition3_f     8
## p_condition1_a_condition2_c_condition3_e_condition1_a_condition2_d_condition3_e     3
## p_condition1_a_condition2_c_condition3_e_condition1_a_condition2_d_condition3_f     7
## p_condition1_a_condition2_c_condition3_e_condition1_b_condition2_c_condition3_e     6
## p_condition1_a_condition2_c_condition3_e_condition1_b_condition2_c_condition3_f     5
## p_condition1_a_condition2_c_condition3_e_condition1_b_condition2_d_condition3_e     7
## p_condition1_a_condition2_c_condition3_e_condition1_b_condition2_d_condition3_f     5
```

```
## p_condition1_a_condition2_c_condition3_f_condition1_a_condition2_d_condition3_e     5
## p_condition1_a_condition2_c_condition3_f_condition1_a_condition2_d_condition3_f     3
## p_condition1_a_condition2_c_condition3_f_condition1_b_condition2_c_condition3_e     4
## p_condition1_a_condition2_c_condition3_f_condition1_b_condition2_c_condition3_f     4
## p_condition1_a_condition2_c_condition3_f_condition1_b_condition2_d_condition3_e     6
## p_condition1_a_condition2_c_condition3_f_condition1_b_condition2_d_condition3_f     7
## p_condition1_a_condition2_d_condition3_e_condition1_a_condition2_d_condition3_f     1
## p_condition1_a_condition2_d_condition3_e_condition1_b_condition2_c_condition3_e     6
## p_condition1_a_condition2_d_condition3_e_condition1_b_condition2_c_condition3_f     1
## p_condition1_a_condition2_d_condition3_e_condition1_b_condition2_d_condition3_e     2
## p_condition1_a_condition2_d_condition3_e_condition1_b_condition2_d_condition3_f     3
## p_condition1_a_condition2_d_condition3_f_condition1_b_condition2_c_condition3_e     5
## p_condition1_a_condition2_d_condition3_f_condition1_b_condition2_c_condition3_f     2
## p_condition1_a_condition2_d_condition3_f_condition1_b_condition2_d_condition3_e     1
## p_condition1_a_condition2_d_condition3_f_condition1_b_condition2_d_condition3_f     3
## p_condition1_b_condition2_c_condition3_e_condition1_b_condition2_c_condition3_f     4
## p_condition1_b_condition2_c_condition3_e_condition1_b_condition2_d_condition3_e     4
## p_condition1_b_condition2_c_condition3_e_condition1_b_condition2_d_condition3_f     7
## p_condition1_b_condition2_c_condition3_f_condition1_b_condition2_d_condition3_e     1
## p_condition1_b_condition2_c_condition3_f_condition1_b_condition2_d_condition3_f     2
## p_condition1_b_condition2_d_condition3_e_condition1_b_condition2_d_condition3_f     5
##                                                                                 effect_size
## p_condition1_a_condition2_c_condition3_e_condition1_a_condition2_c_condition3_f  0.034099563
## p_condition1_a_condition2_c_condition3_e_condition1_a_condition2_d_condition3_e  0.013527418
## p_condition1_a_condition2_c_condition3_e_condition1_a_condition2_d_condition3_f  0.016821856
## p_condition1_a_condition2_c_condition3_e_condition1_b_condition2_c_condition3_e  0.017997083
## p_condition1_a_condition2_c_condition3_e_condition1_b_condition2_c_condition3_f -0.000004627
## p_condition1_a_condition2_c_condition3_e_condition1_b_condition2_d_condition3_e  0.007240448
## p_condition1_a_condition2_c_condition3_e_condition1_b_condition2_d_condition3_f -0.005670795
## p_condition1_a_condition2_c_condition3_f_condition1_a_condition2_d_condition3_e -0.017784563
## p_condition1_a_condition2_c_condition3_f_condition1_a_condition2_d_condition3_f -0.014882624
## p_condition1_a_condition2_c_condition3_f_condition1_b_condition2_c_condition3_e -0.013385432
## p_condition1_a_condition2_c_condition3_f_condition1_b_condition2_c_condition3_f -0.032383085
## p_condition1_a_condition2_c_condition3_f_condition1_b_condition2_d_condition3_e -0.024270671
## p_condition1_a_condition2_c_condition3_f_condition1_b_condition2_d_condition3_f -0.037354599
## p_condition1_a_condition2_d_condition3_e_condition1_a_condition2_d_condition3_f  0.001973644
## p_condition1_a_condition2_d_condition3_e_condition1_b_condition2_c_condition3_e  0.002576426
## p_condition1_a_condition2_d_condition3_e_condition1_b_condition2_c_condition3_f -0.015970973
## p_condition1_a_condition2_d_condition3_e_condition1_b_condition2_d_condition3_e -0.006416174
## p_condition1_a_condition2_d_condition3_e_condition1_b_condition2_d_condition3_f -0.021936820
## p_condition1_a_condition2_d_condition3_f_condition1_b_condition2_c_condition3_e  0.000534329
## p_condition1_a_condition2_d_condition3_f_condition1_b_condition2_c_condition3_f -0.015715868
## p_condition1_a_condition2_d_condition3_f_condition1_b_condition2_d_condition3_e -0.007152002
## p_condition1_a_condition2_d_condition3_f_condition1_b_condition2_d_condition3_f -0.021430237
## p_condition1_b_condition2_c_condition3_e_condition1_b_condition2_c_condition3_f -0.017112689
## p_condition1_b_condition2_c_condition3_e_condition1_b_condition2_d_condition3_e -0.009162072
## p_condition1_b_condition2_c_condition3_e_condition1_b_condition2_d_condition3_f -0.023933237
## p_condition1_b_condition2_c_condition3_f_condition1_b_condition2_d_condition3_e  0.009882990
## p_condition1_b_condition2_c_condition3_f_condition1_b_condition2_d_condition3_f -0.005943129
## p_condition1_b_condition2_d_condition3_e_condition1_b_condition2_d_condition3_f -0.014443698
```

When there is no true effect, we formally do not have 'power' (which is defined as the probability of finding $p < \alpha$ if there is a true effect to be found) so the power column should be read as the 'Type 1 error rate'. Because we have saved the power simulation in the 'power_result' object, we can perform calculations on

the 'sim_data' dataframe that is stored. This dataframe contains the results for the nsims simulations (e.g., 10000 rows if you ran 10000 simulations) and stores the p-values and effect size estimates for each ANOVA. The first 7 columns are the p-values for the ANOVA, first the main effects of condition 1, 2, and 3, then three two-way interactions, and finally the threeway interaction.

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

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

```
##                         anova_condition1
##                                        5
##                         anova_condition2
##                                        4
##                         anova_condition3
##                                        2
##              anova_condition1:condition2
##                                        1
##              anova_condition1:condition3
##                                        6
##              anova_condition2:condition3
##                                        6
## anova_condition1:condition2:condition3
##                                        3
```

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

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

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

```
## [1] 24
```

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

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

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

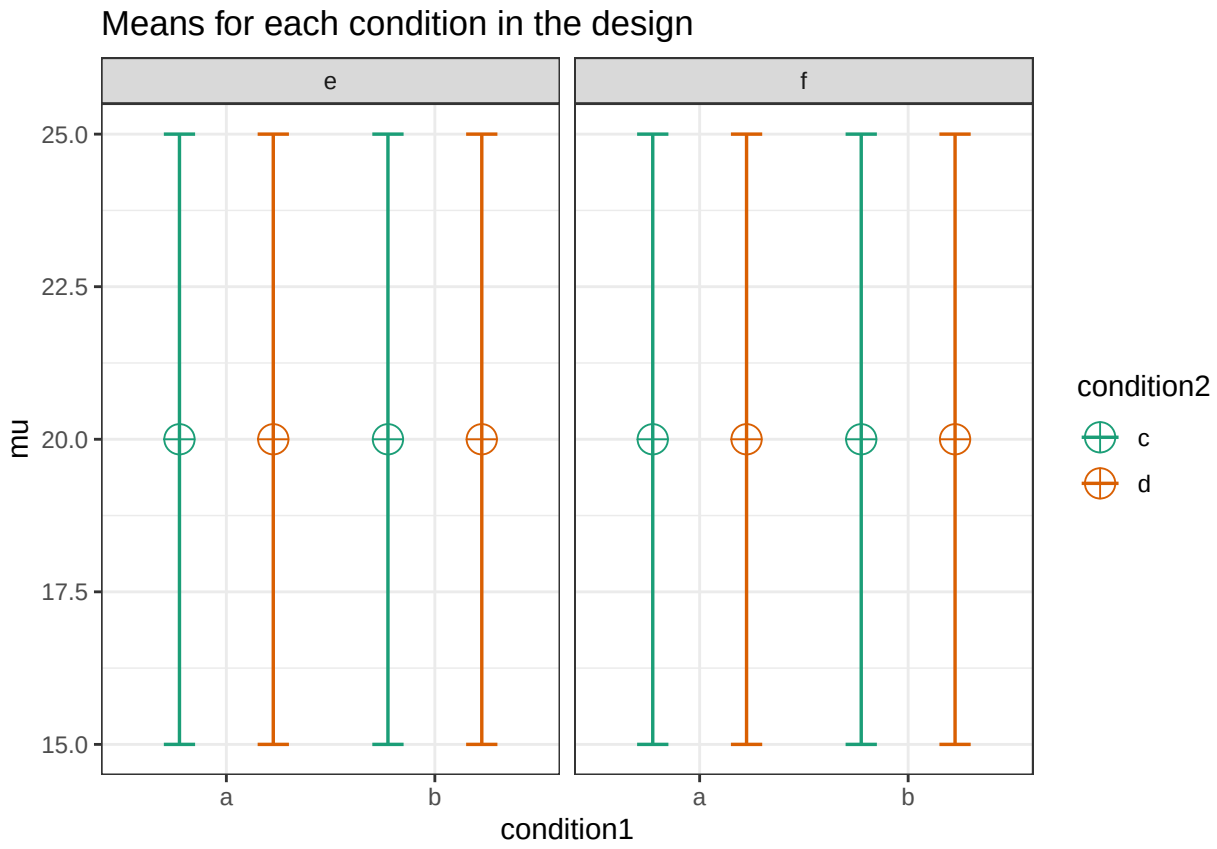
The adjustment methods include the Bonferroni correction ("bonferroni") in which the p-values are multiplied by the number of comparisons. Less conservative corrections are also included by Holm (1979) ("holm"), Hochberg (1988) ("hochberg"), Hommel (1988) ("hommel"), Benjamini & Hochberg (1995) ("BH"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

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

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

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

```
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
#We set the alpha level at 0.05.
power_result <- ANOVA_power(design_result,
                            alpha_level = alpha_level,
                            p_adjust = p_adjust,
                            nsims = nsims)
```

```
## Power and Effect sizes for ANOVA tests
##                                       power effect_size
## anova_condition1                          0    0.001852
## anova_condition2                          0    0.002161
## anova_condition3                          1    0.002694
## anova_condition1:condition2               0    0.002381
## anova_condition1:condition3               0    0.001987
## anova_condition2:condition3               0    0.002424
## anova_condition1:condition2:condition3     0    0.002549
##
## Power and Effect sizes for contrasts
##                                                                                 power
## p_condition1_a_condition2_c_condition3_e_condition1_a_condition2_c_condition3_f     1
## p_condition1_a_condition2_c_condition3_e_condition1_a_condition2_d_condition3_e     0
## p_condition1_a_condition2_c_condition3_e_condition1_a_condition2_d_condition3_f     0
## p_condition1_a_condition2_c_condition3_e_condition1_b_condition2_c_condition3_e     0
## p_condition1_a_condition2_c_condition3_e_condition1_b_condition2_c_condition3_f     0
## p_condition1_a_condition2_c_condition3_e_condition1_b_condition2_d_condition3_e     0
## p_condition1_a_condition2_c_condition3_e_condition1_b_condition2_d_condition3_f     0
```

```
## p_condition1_a_condition2_c_condition3_f_condition1_a_condition2_d_condition3_e     0
## p_condition1_a_condition2_c_condition3_f_condition1_a_condition2_d_condition3_f     0
## p_condition1_a_condition2_c_condition3_f_condition1_b_condition2_c_condition3_e     0
## p_condition1_a_condition2_c_condition3_f_condition1_b_condition2_c_condition3_f     0
## p_condition1_a_condition2_c_condition3_f_condition1_b_condition2_d_condition3_e     0
## p_condition1_a_condition2_c_condition3_f_condition1_b_condition2_d_condition3_f     0
## p_condition1_a_condition2_d_condition3_e_condition1_a_condition2_d_condition3_f     1
## p_condition1_a_condition2_d_condition3_e_condition1_b_condition2_c_condition3_e     0
## p_condition1_a_condition2_d_condition3_e_condition1_b_condition2_c_condition3_f     0
## p_condition1_a_condition2_d_condition3_e_condition1_b_condition2_d_condition3_e     0
## p_condition1_a_condition2_d_condition3_e_condition1_b_condition2_d_condition3_f     0
## p_condition1_a_condition2_d_condition3_f_condition1_b_condition2_c_condition3_e     0
## p_condition1_a_condition2_d_condition3_f_condition1_b_condition2_c_condition3_f     0
## p_condition1_a_condition2_d_condition3_f_condition1_b_condition2_d_condition3_e     0
## p_condition1_a_condition2_d_condition3_f_condition1_b_condition2_d_condition3_f     0
## p_condition1_b_condition2_c_condition3_e_condition1_b_condition2_c_condition3_f     0
## p_condition1_b_condition2_c_condition3_e_condition1_b_condition2_d_condition3_e     0
## p_condition1_b_condition2_c_condition3_e_condition1_b_condition2_d_condition3_f     0
## p_condition1_b_condition2_c_condition3_f_condition1_b_condition2_d_condition3_e     0
## p_condition1_b_condition2_c_condition3_f_condition1_b_condition2_d_condition3_f     0
## p_condition1_b_condition2_d_condition3_e_condition1_b_condition2_d_condition3_f     0
##                                                                                 effect_size
## p_condition1_a_condition2_c_condition3_e_condition1_a_condition2_c_condition3_f    0.021965
## p_condition1_a_condition2_c_condition3_e_condition1_a_condition2_d_condition3_e    0.031111
## p_condition1_a_condition2_c_condition3_e_condition1_a_condition2_d_condition3_f    0.031687
## p_condition1_a_condition2_c_condition3_e_condition1_b_condition2_c_condition3_e    0.011195
## p_condition1_a_condition2_c_condition3_e_condition1_b_condition2_c_condition3_f   -0.003282
## p_condition1_a_condition2_c_condition3_e_condition1_b_condition2_d_condition3_e    0.002246
## p_condition1_a_condition2_c_condition3_e_condition1_b_condition2_d_condition3_f    0.021404
## p_condition1_a_condition2_c_condition3_f_condition1_a_condition2_d_condition3_e    0.008657
## p_condition1_a_condition2_c_condition3_f_condition1_a_condition2_d_condition3_f    0.009576
## p_condition1_a_condition2_c_condition3_f_condition1_b_condition2_c_condition3_e   -0.010283
## p_condition1_a_condition2_c_condition3_f_condition1_b_condition2_c_condition3_f   -0.024728
## p_condition1_a_condition2_c_condition3_f_condition1_b_condition2_d_condition3_e   -0.019057
## p_condition1_a_condition2_c_condition3_f_condition1_b_condition2_d_condition3_f    0.001137
## p_condition1_a_condition2_d_condition3_e_condition1_a_condition2_d_condition3_f    0.002556
## p_condition1_a_condition2_d_condition3_e_condition1_b_condition2_c_condition3_e   -0.021909
## p_condition1_a_condition2_d_condition3_e_condition1_b_condition2_c_condition3_f   -0.034872
## p_condition1_a_condition2_d_condition3_e_condition1_b_condition2_d_condition3_e   -0.029285
## p_condition1_a_condition2_d_condition3_e_condition1_b_condition2_d_condition3_f   -0.010442
## p_condition1_a_condition2_d_condition3_f_condition1_b_condition2_c_condition3_e   -0.020592
## p_condition1_a_condition2_d_condition3_f_condition1_b_condition2_c_condition3_f   -0.033982
## p_condition1_a_condition2_d_condition3_f_condition1_b_condition2_d_condition3_e   -0.027786
## p_condition1_a_condition2_d_condition3_f_condition1_b_condition2_d_condition3_f   -0.007902
## p_condition1_b_condition2_c_condition3_e_condition1_b_condition2_c_condition3_f   -0.014459
## p_condition1_b_condition2_c_condition3_e_condition1_b_condition2_d_condition3_e   -0.008649
## p_condition1_b_condition2_c_condition3_e_condition1_b_condition2_d_condition3_f    0.010618
## p_condition1_b_condition2_c_condition3_f_condition1_b_condition2_d_condition3_e    0.005174
## p_condition1_b_condition2_c_condition3_f_condition1_b_condition2_d_condition3_f    0.025110
## p_condition1_b_condition2_d_condition3_e_condition1_b_condition2_d_condition3_f    0.019101
```

If we now calculate the overall Type 1 error rate:

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

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

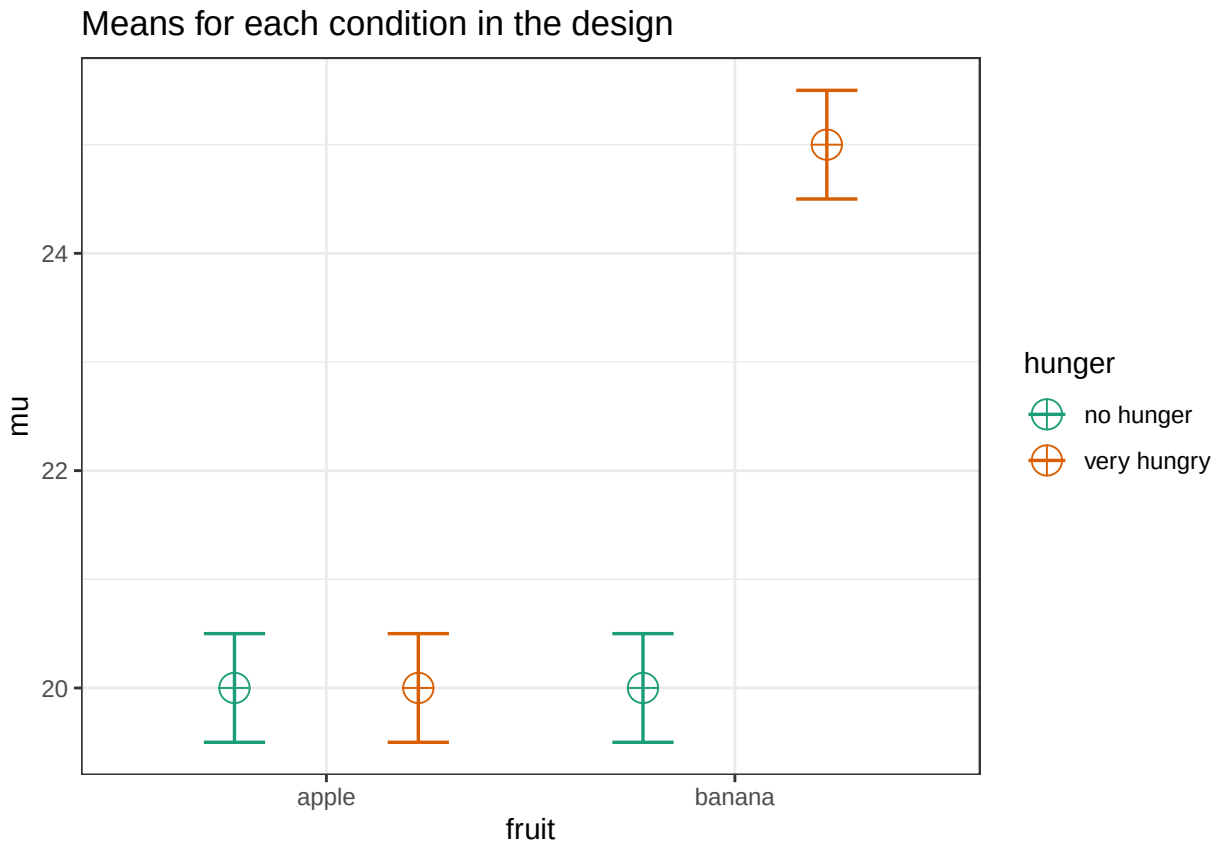
We see it is close to 5%. Note that error rates have variation, and even in a few thousand simulations, the error rate in the sample of studies can easily be half a percentage point higher or lower. But *in the long run* the error rate should equal the alpha level. Furthermore, note that the Holm-bonferroni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

Chapter 12

Power in Interactions

In the 17th Data Colada blog post titled No-way Interactions Uri Simonsohn 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
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 <- 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)
```



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
#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
  #simtot: how many simulations to run
  #Here we will store results
      p1=c()     #p-values for Study 1
```

```
      p2=c()    #p-values for Study 2
  for(i in 1:simtot) {
    #draw data 4 samples
    y1=rnorm(n=max(n1,n2),mean=d1)
    y2=rnorm(n=max(n1,n2))
    y3=rnorm(n=max(n1,n2),mean=d2)
    y4=rnorm(n=max(n1,n2))

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

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

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

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

      }

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

    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: 0.88
```

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

```
##
## Study 1 is powered to: 0.31
## Study 2 is powered to: 0.32
```

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

```
##
```

```
## Study 1 is powered to: 0.59
## Study 2 is powered to: 0.59
```

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

```
##
## Study 1 is powered to: 0.4
## Study 2 is powered to: 0.41
```

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

```
##
## Study 1 is powered to: 0.71
## Study 2 is powered to: 0.7
```

```
  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: 0.85
## Study 2 is powered to: 0.56
```

And we can reproduce the results using the ANOVA_power function.

```
alpha_level <- 0.05 #We set the alpha level at 0.05.
power_result <- ANOVA_power(design_result, alpha_level = alpha_level, nsims = nsims)
```

```
## Power and Effect sizes for ANOVA tests
##                    power effect_size
## anova_fruit          100      0.8677
## anova_hunger         100      0.8695
## anova_fruit:hunger   100      0.8666
##
## Power and Effect sizes for contrasts
##                                                                  power
## p_fruit_apple_hunger_no hunger_fruit_apple_hunger_very hungry        4
## p_fruit_apple_hunger_no hunger_fruit_banana_hunger_no hunger         3
## p_fruit_apple_hunger_no hunger_fruit_banana_hunger_very hungry     100
## p_fruit_apple_hunger_very hungry_fruit_banana_hunger_no hunger       5
## p_fruit_apple_hunger_very hungry_fruit_banana_hunger_very hungry   100
## p_fruit_banana_hunger_no hunger_fruit_banana_hunger_very hungry    100
##                                                                  effect_size
## p_fruit_apple_hunger_no hunger_fruit_apple_hunger_very hungry        0.05925
```

```
## p_fruit_apple_hunger_no hunger_fruit_banana_hunger_no hunger          0.02399
## p_fruit_apple_hunger_no hunger_fruit_banana_hunger_very hungry       10.20519
## p_fruit_apple_hunger_very hungry_fruit_banana_hunger_no hunger       -0.03362
## p_fruit_apple_hunger_very hungry_fruit_banana_hunger_very hungry      9.98146
## p_fruit_banana_hunger_no hunger_fruit_banana_hunger_very hungry      10.14086
```

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)
```

```
##
## Study 1 is powered to: 0.87
## Study 2 is powered to: 0.88
```

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

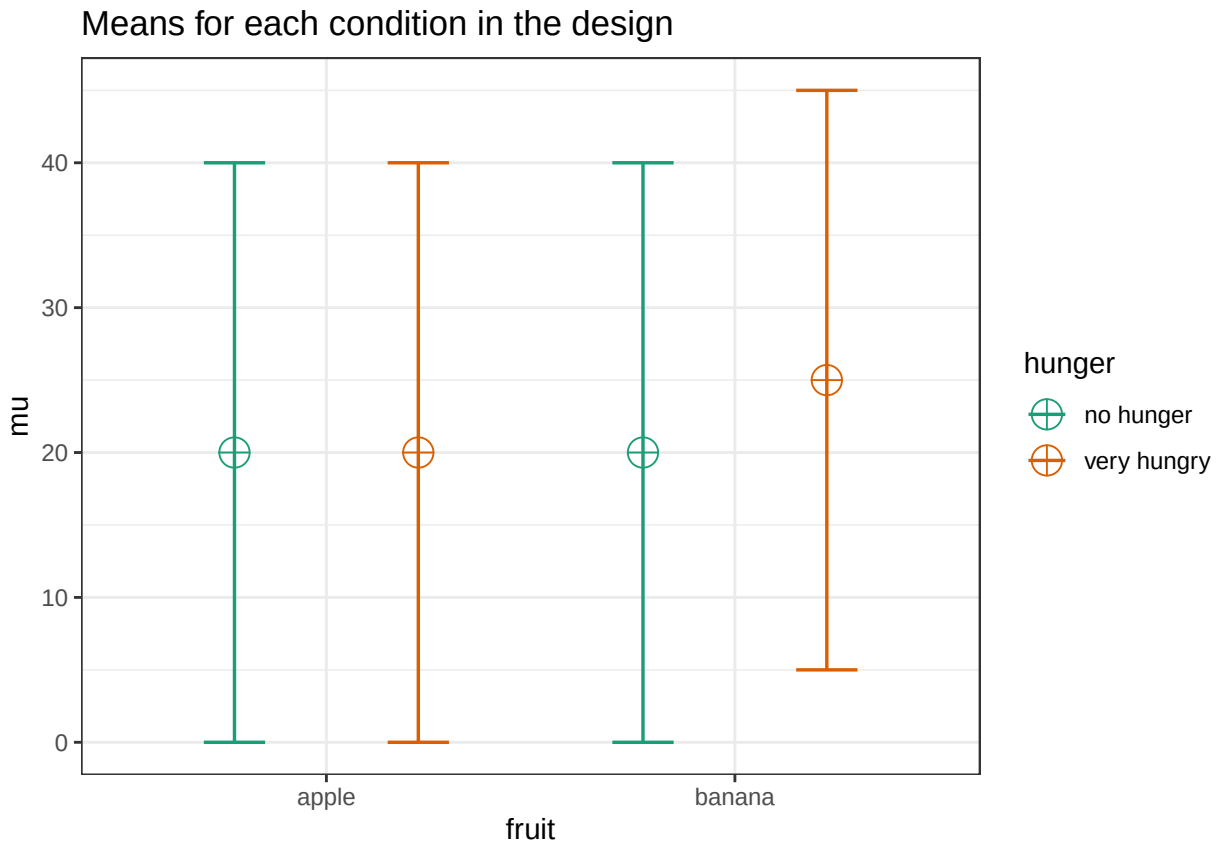
```
##
## Study 1 is powered to: 0.31
## Study 2 is powered to: 0.32
```

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

```
##
## Study 1 is powered to: 0.58
## Study 2 is powered to: 0.59
```

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)
```

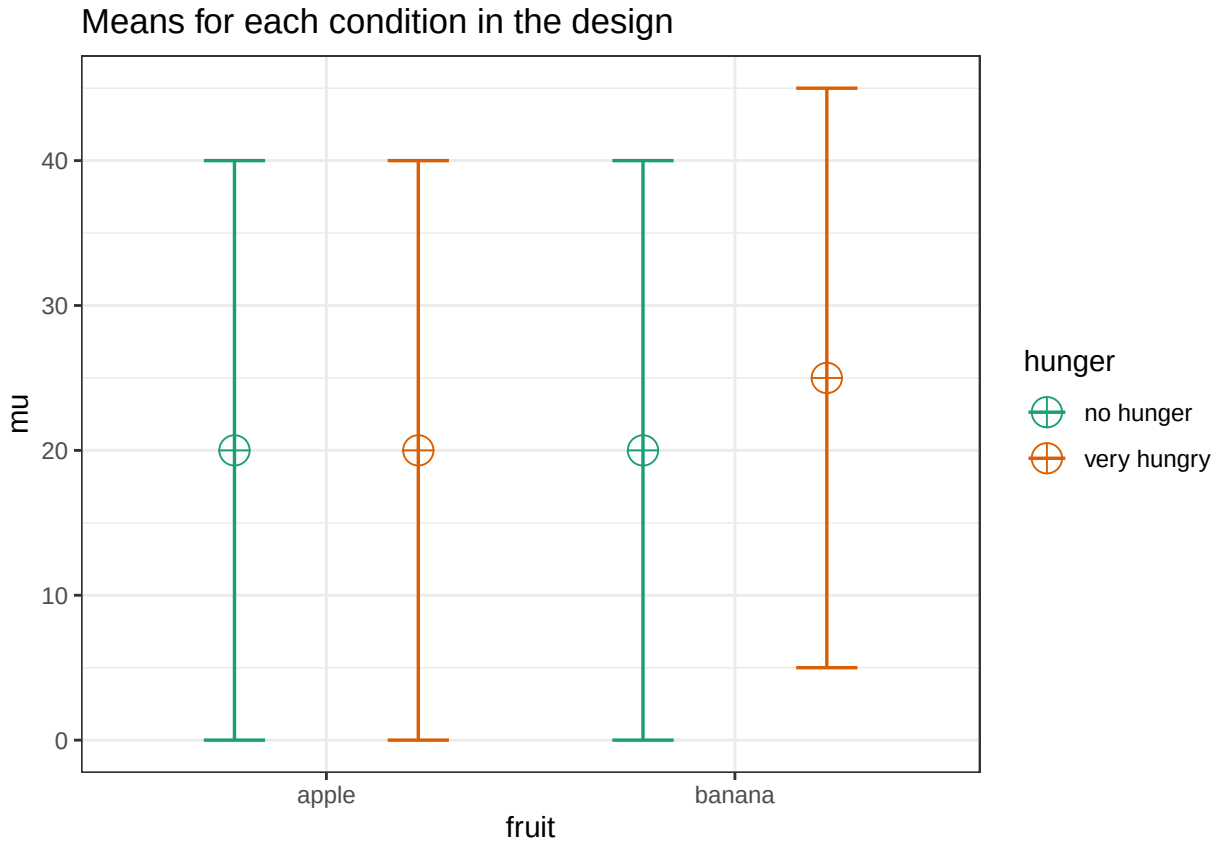



```
alpha_level <- 0.05 #We set the alpha level at 0.05.
power_result <- ANOVA_power(design_result, alpha_level = alpha_level, nsims = nsims)
```

```
## Power and Effect sizes for ANOVA tests
##                    power effect_size
## anova_fruit           27    0.005035
## anova_hunger          32    0.005649
## anova_fruit:hunger    31    0.005571
##
## Power and Effect sizes for contrasts
##                                                                  power
## p_fruit_apple_hunger_no hunger_fruit_apple_hunger_very hungry         4
## p_fruit_apple_hunger_no hunger_fruit_banana_hunger_no hunger          3
## p_fruit_apple_hunger_no hunger_fruit_banana_hunger_very hungry       50
## p_fruit_apple_hunger_very hungry_fruit_banana_hunger_no hunger        5
## p_fruit_apple_hunger_very hungry_fruit_banana_hunger_very hungry     54
## p_fruit_banana_hunger_no hunger_fruit_banana_hunger_very hungry      56
##                                                                  effect_size
## p_fruit_apple_hunger_no hunger_fruit_apple_hunger_very hungry       0.003055
## p_fruit_apple_hunger_no hunger_fruit_banana_hunger_no hunger       -0.013895
## p_fruit_apple_hunger_no hunger_fruit_banana_hunger_very hungry      0.241859
## p_fruit_apple_hunger_very hungry_fruit_banana_hunger_no hunger     -0.016907
## p_fruit_apple_hunger_very hungry_fruit_banana_hunger_very hungry    0.238201
## p_fruit_banana_hunger_no hunger_fruit_banana_hunger_very hungry     0.255458
```

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

```
## Power and Effect sizes for ANOVA tests
##                     power effect_size
## anova_fruit            58    0.004884
## anova_hunger           64    0.005216
## anova_fruit:hunger     51    0.004574
##
## Power and Effect sizes for contrasts
##                                                                  power
## p_fruit_apple_hunger_no hunger_fruit_apple_hunger_very hungry        6
```

```
## p_fruit_apple_hunger_no hunger_fruit_banana_hunger_no hunger         7
## p_fruit_apple_hunger_no hunger_fruit_banana_hunger_very hungry      92
## p_fruit_apple_hunger_very hungry_fruit_banana_hunger_no hunger       5
## p_fruit_apple_hunger_very hungry_fruit_banana_hunger_very hungry    83
## p_fruit_banana_hunger_no hunger_fruit_banana_hunger_very hungry     86
##                                                                 effect_size
## p_fruit_apple_hunger_no hunger_fruit_apple_hunger_very hungry       0.012670
## p_fruit_apple_hunger_no hunger_fruit_banana_hunger_no hunger        0.009070
## p_fruit_apple_hunger_no hunger_fruit_banana_hunger_very hungry      0.260997
## p_fruit_apple_hunger_very hungry_fruit_banana_hunger_no hunger     -0.003496
## p_fruit_apple_hunger_very hungry_fruit_banana_hunger_very hungry    0.248100
## p_fruit_banana_hunger_no hunger_fruit_banana_hunger_very hungry     0.252196
```

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

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

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

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

```
rowMeans(mean_mat)
```

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

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

```
colMeans(mean_mat)
```

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

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

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

+ 1.25 [the marginal mean in row 2, 22.5, minus the grand mean of 21.25] + 1.25 [the marginal mean in column 2, 22.5, minus the grand mean of 21.25]). 25 - (21.25 + (22.5-21.25) + (22.5-21.25)) = 1.25.

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

```
a1 <- mean_mat[1,1] - (mean(mean_mat) + (mean(mean_mat[1,]) - mean(mean_mat)) + (mean(mean_mat[,1]) - m
a2 <- mean_mat[1,2] - (mean(mean_mat) + (mean(mean_mat[1,]) - mean(mean_mat)) + (mean(mean_mat[,2]) - m
b1 <- mean_mat[2,1] - (mean(mean_mat) + (mean(mean_mat[2,]) - mean(mean_mat)) + (mean(mean_mat[,1]) - m
b2 <- mean_mat[2,2] - (mean(mean_mat) + (mean(mean_mat[2,]) - mean(mean_mat)) + (mean(mean_mat[,2]) - m
SS_ab <- n * sum(c(a1, a2, b1, b2)^2)
```

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

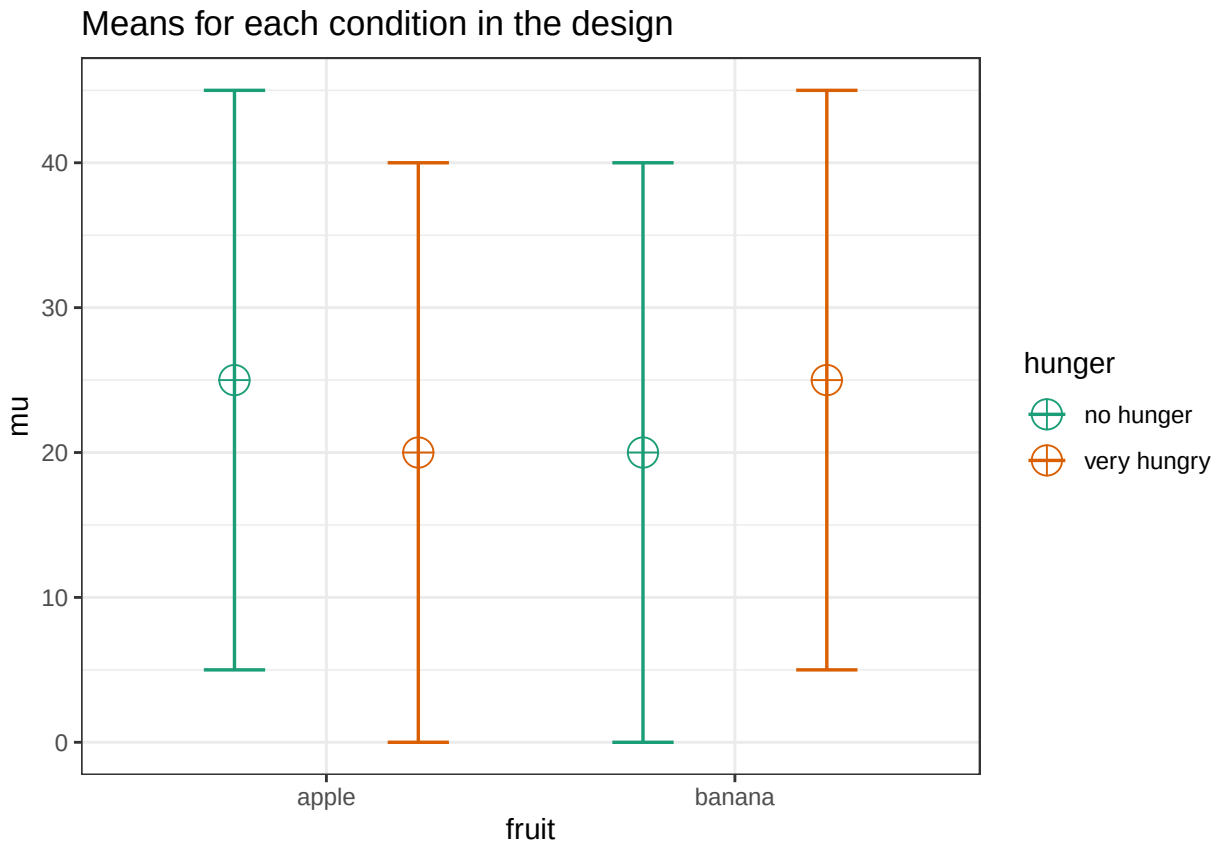
Interactions can have more power than main effects if the effect size of the interaction is larger than the effect size of the main effects. An example of this is a cross-over interaction. For example, let's take a 2x2 matrix of means with a crossover interaction:

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

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

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

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




```
alpha_level <- 0.05 #We set the alpha level at 0.05.
power_result <- ANOVA_power(design_result, alpha_level = alpha_level, nsims = nsims)
```

```
## Power and Effect sizes for ANOVA tests
##                    power effect_size
## anova_fruit            9   0.0009299
## anova_hunger           6   0.0008283
## anova_fruit:hunger   100   0.0160483
##
## Power and Effect sizes for contrasts
##                                                                  power
## p_fruit_apple_hunger_no hunger_fruit_apple_hunger_very hungry       91
## p_fruit_apple_hunger_no hunger_fruit_banana_hunger_no hunger        85
## p_fruit_apple_hunger_no hunger_fruit_banana_hunger_very hungry       6
## p_fruit_apple_hunger_very hungry_fruit_banana_hunger_no hunger       6
## p_fruit_apple_hunger_very hungry_fruit_banana_hunger_very hungry    84
## p_fruit_banana_hunger_no hunger_fruit_banana_hunger_very hungry     83
##                                                                  effect_size
## p_fruit_apple_hunger_no hunger_fruit_apple_hunger_very hungry      -0.256673
## p_fruit_apple_hunger_no hunger_fruit_banana_hunger_no hunger       -0.248062
## p_fruit_apple_hunger_no hunger_fruit_banana_hunger_very hungry     -0.005798
## p_fruit_apple_hunger_very hungry_fruit_banana_hunger_no hunger      0.007876
## p_fruit_apple_hunger_very hungry_fruit_banana_hunger_very hungry    0.250525
## p_fruit_banana_hunger_no hunger_fruit_banana_hunger_very hungry     0.242007
```

We can also reproduce the power analysis using the anlytic function:

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

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

```
power_analytic$power_B
```

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

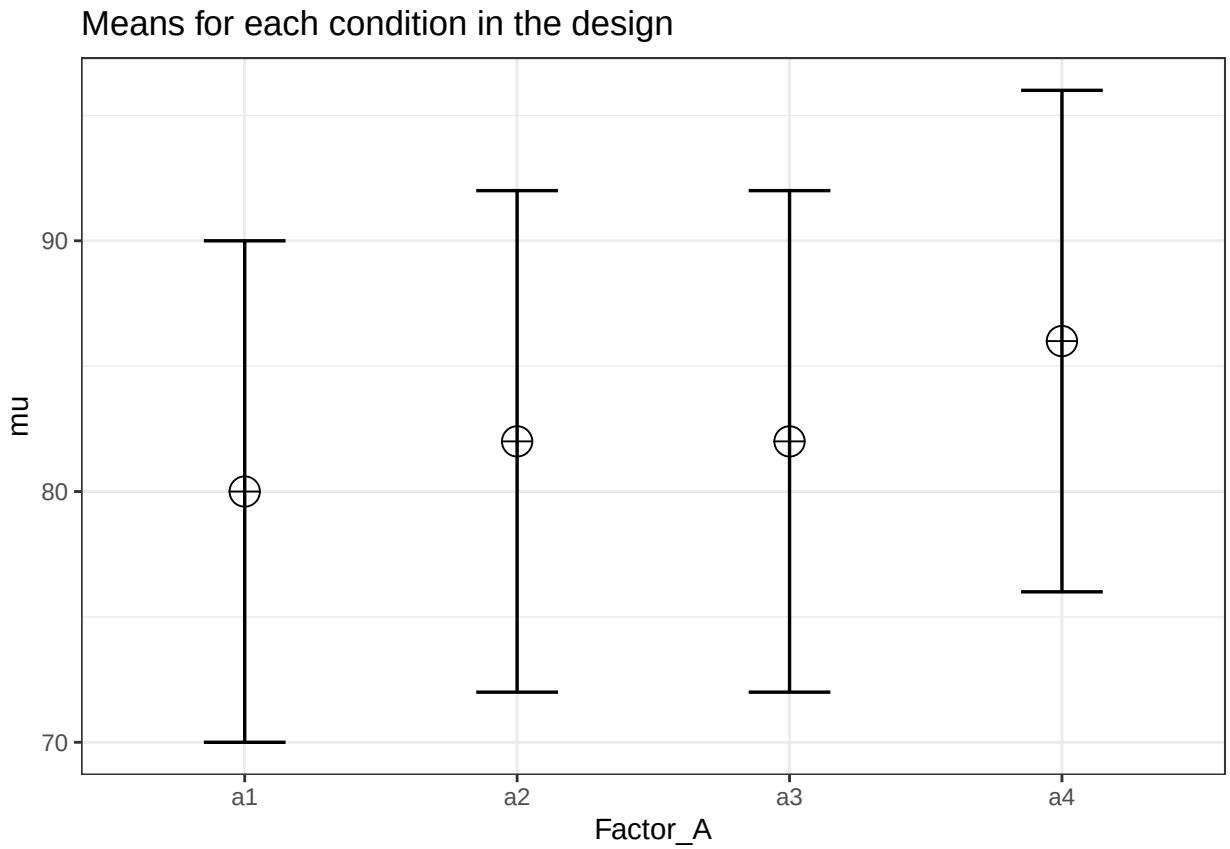
Chapter 13

Analytic Power Functions

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

13.1 One-Way Between Subject ANOVA

```
string <- "4b"
n <- 60
mu <- c(80, 82, 82, 86) #All means are equal - so there is no real difference.
# Enter means in the order that matches the labels below.
sd <- 10
labelnames <- c("Factor_A", "a1", "a2", "a3", "a4") #
# the label names should be in the order of the means specified above.
design_result <- ANOVA_design(design = string,
                   n = n,
                   mu = mu,
                   sd = sd,
                   labelnames = labelnames)
```




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

```
## Power and Effect sizes for ANOVA tests
##                power effect_size
## anova_Factor_A    75     0.05081
##
## Power and Effect sizes for contrasts
##                            power effect_size
## p_Factor_A_a1_Factor_A_a2    17    0.166420
## p_Factor_A_a1_Factor_A_a3    14    0.161871
## p_Factor_A_a1_Factor_A_a4    87    0.554659
## p_Factor_A_a2_Factor_A_a3     6   -0.004014
## p_Factor_A_a2_Factor_A_a4    57    0.387251
## p_Factor_A_a3_Factor_A_a4    51    0.388607
```

We can also calculate power analytically with our own function.

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

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

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

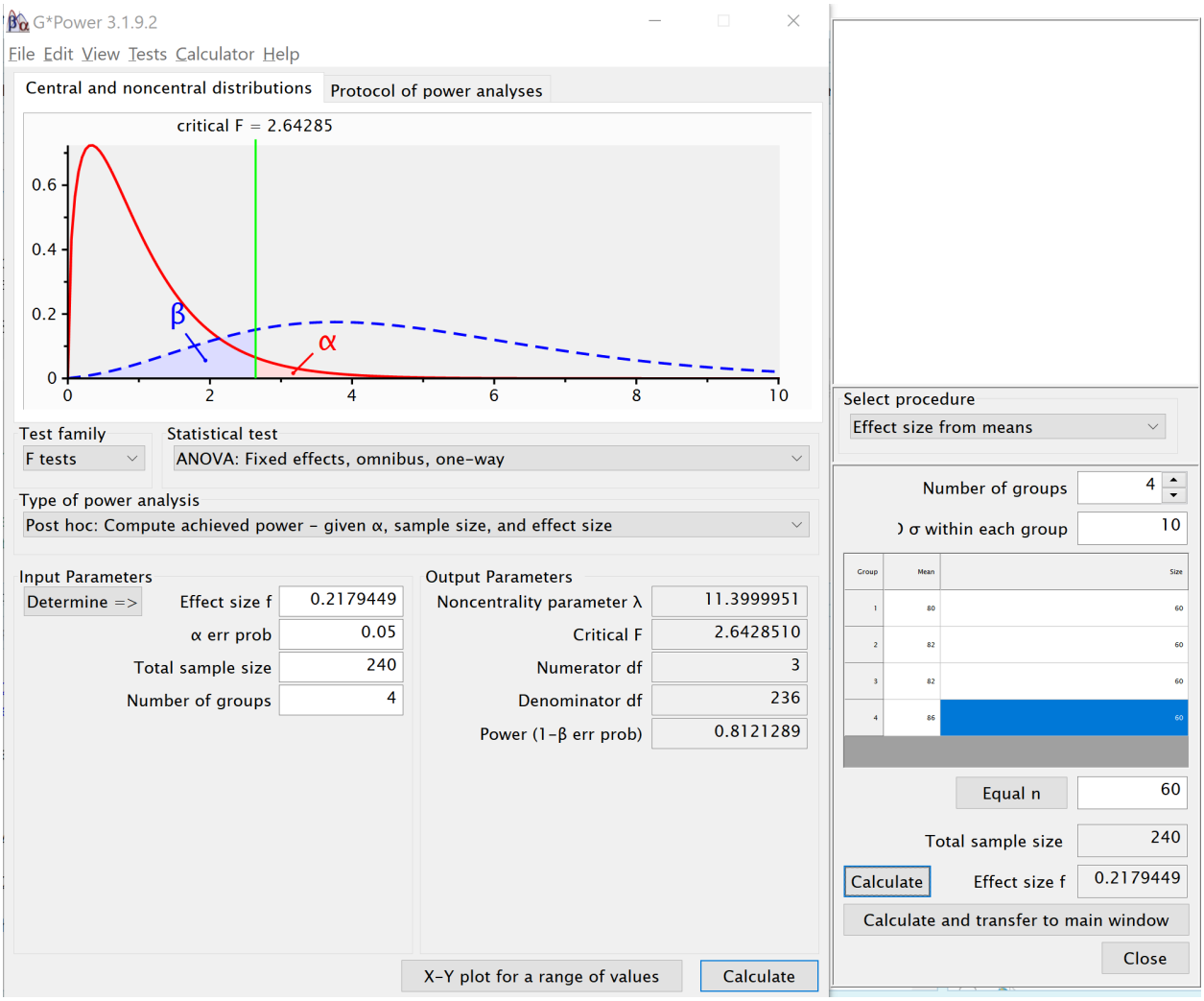
```
anova1f_4(m1 = 80, m2 = 82, m3 = 82, m4 = 86,
          s1 = 10, s2 = 10, s3 = 10, s4 = 10,
          n1 = 60, n2 = 60, n3 = 60, n4 = 60,
          alpha = .05)
```

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

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

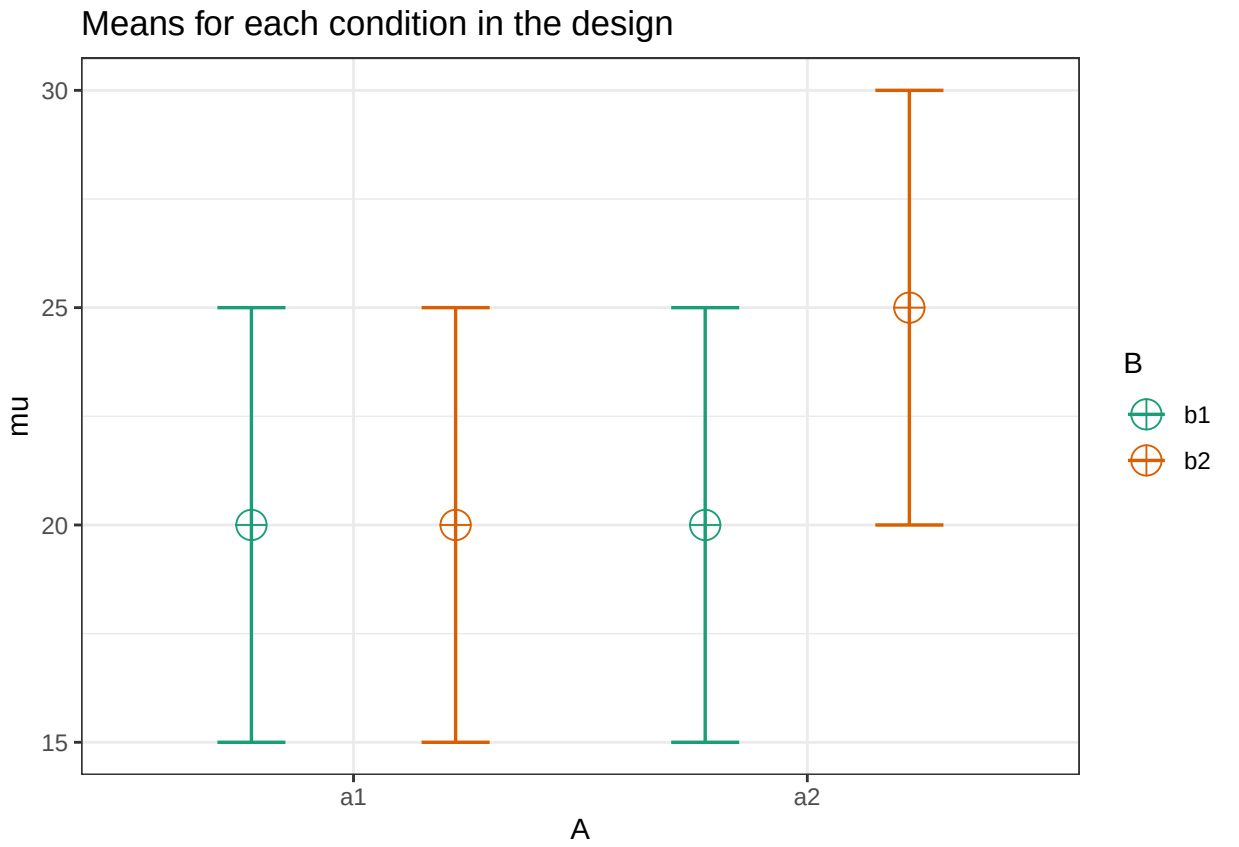
```
##
##      Balanced one-way analysis of variance power calculation
##
##               k = 4
##               n = 60
##               f = 0.2179449
##       sig.level = 0.05
##           power = 0.8121289
##
## NOTE: n is number in each group
```

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



13.2 Two-way Between Subject Interaction

```
string <- "2b*2b"
n <- 20
mu <- c(20, 20, 20, 25)
# Enter means in the order that matches the labels below.
sd <- 5
labelnames <- c("A", "a1", "a2", "B", "b1", "b2") #
# the label names should be in the order of the means specified above.
design_result <- ANOVA_design(design = string,
                   n = n,
                   mu = mu,
                   sd = sd,
                   labelnames = labelnames)
```




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

```
## Power and Effect sizes for ANOVA tests
##           power effect_size
## anova_A      61     0.07402
## anova_B      67     0.07842
## anova_A:B    62     0.06883
##
## Power and Effect sizes for contrasts
##                         power effect_size
## p_A_a1_B_b1_A_a1_B_b2     5     0.03784
## p_A_a1_B_b1_A_a2_B_b1     3     0.01744
## p_A_a1_B_b1_A_a2_B_b2    93     1.06236
## p_A_a1_B_b2_A_a2_B_b1     6    -0.01497
## p_A_a1_B_b2_A_a2_B_b2    85     1.01365
## p_A_a2_B_b1_A_a2_B_b2    86     1.03545
```

```
power_res <- power_twoway_between(design_result) #using default alpha level of .05
power_res$power_A
```

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

```
power_res$power_B
```

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

```
power_res$power_AB
```

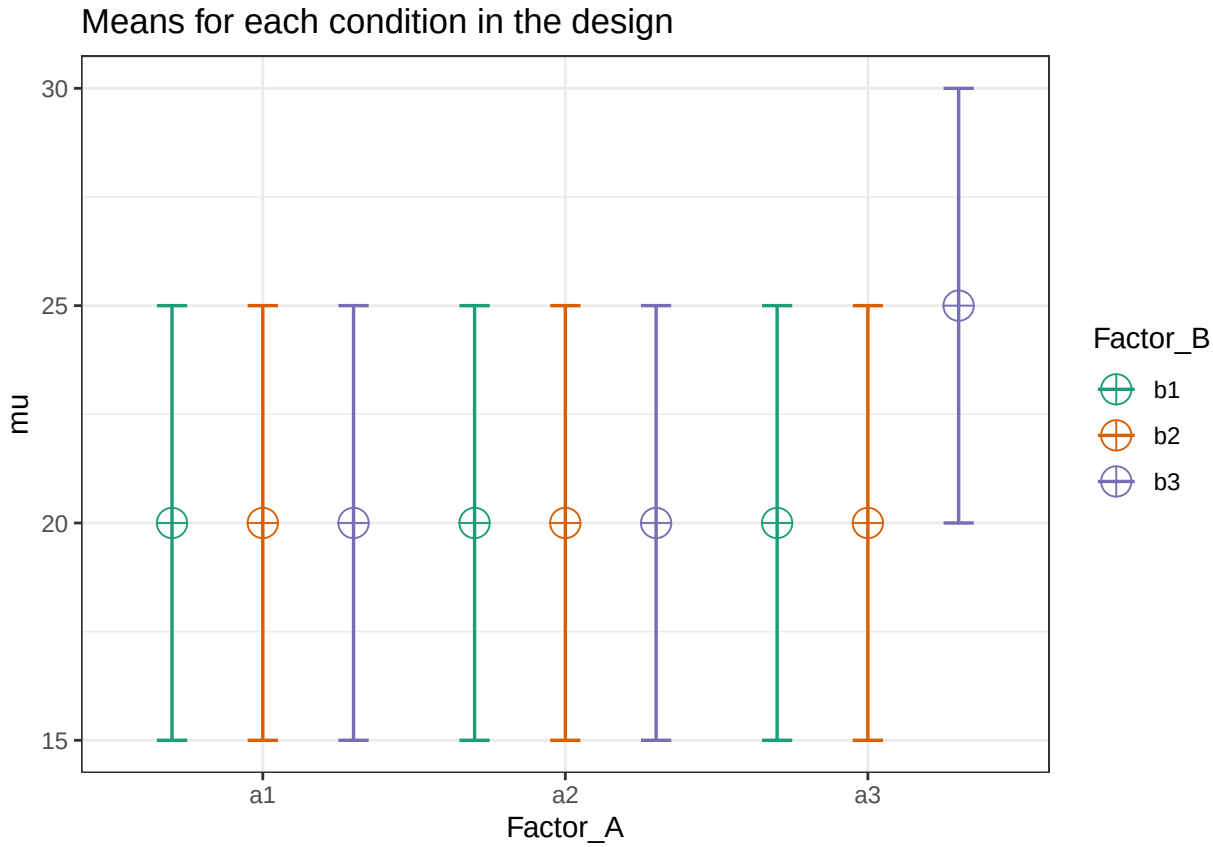
```
## [1] 0.5978655
```

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

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

13.3 3x3 Between Subject ANOVA

```
string <- "3b*3b"
n <- 20
mu <- c(20, 20, 20, 20, 20, 20, 20, 20, 25) #All means are equal - so there is no real difference.
# Enter means in the order that matches the labels below.
sd <- 5
labelnames <- c("Factor_A", "a1", "a2", "a3", "Factor_B", "b1", "b2", "b3") #
# the label names should be in the order of the means specified above.
design_result <- ANOVA_design(design = string,
                   n = n,
                   mu = mu,
                   sd = sd,
                   labelnames = labelnames)
```




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

```
## Power and Effect sizes for ANOVA tests
##                          power effect_size
## anova_Factor_A              44     0.03703
## anova_Factor_B              49     0.03887
## anova_Factor_A:Factor_B     68     0.07409
##
## Power and Effect sizes for contrasts
##                                                    power effect_size
## p_Factor_A_a1_Factor_B_b1_Factor_A_a1_Factor_B_b2     1   0.0423477
## p_Factor_A_a1_Factor_B_b1_Factor_A_a1_Factor_B_b3     3   0.0547665
## p_Factor_A_a1_Factor_B_b1_Factor_A_a2_Factor_B_b1    10   0.0184574
## p_Factor_A_a1_Factor_B_b1_Factor_A_a2_Factor_B_b2     7  -0.0003622
## p_Factor_A_a1_Factor_B_b1_Factor_A_a2_Factor_B_b3    11  -0.0116560
## p_Factor_A_a1_Factor_B_b1_Factor_A_a3_Factor_B_b1    10  -0.0037645
## p_Factor_A_a1_Factor_B_b1_Factor_A_a3_Factor_B_b2     3   0.0180993
## p_Factor_A_a1_Factor_B_b1_Factor_A_a3_Factor_B_b3    88   1.0926958
## p_Factor_A_a1_Factor_B_b2_Factor_A_a1_Factor_B_b3     2   0.0175707
## p_Factor_A_a1_Factor_B_b2_Factor_A_a2_Factor_B_b1     6  -0.0252172
## p_Factor_A_a1_Factor_B_b2_Factor_A_a2_Factor_B_b2     3  -0.0389791
## p_Factor_A_a1_Factor_B_b2_Factor_A_a2_Factor_B_b3     4  -0.0528683
## p_Factor_A_a1_Factor_B_b2_Factor_A_a3_Factor_B_b1     5  -0.0499965
## p_Factor_A_a1_Factor_B_b2_Factor_A_a3_Factor_B_b2     4  -0.0204242
## p_Factor_A_a1_Factor_B_b2_Factor_A_a3_Factor_B_b3    88   1.0374540
```

```
## p_Factor_A_a1_Factor_B_b3_Factor_A_a2_Factor_B_b1     2  -0.0388015
## p_Factor_A_a1_Factor_B_b3_Factor_A_a2_Factor_B_b2     5  -0.0642951
## p_Factor_A_a1_Factor_B_b3_Factor_A_a2_Factor_B_b3     3  -0.0703655
## p_Factor_A_a1_Factor_B_b3_Factor_A_a3_Factor_B_b1     5  -0.0658323
## p_Factor_A_a1_Factor_B_b3_Factor_A_a3_Factor_B_b2     3  -0.0419601
## p_Factor_A_a1_Factor_B_b3_Factor_A_a3_Factor_B_b3    88   1.0199386
## p_Factor_A_a2_Factor_B_b1_Factor_A_a2_Factor_B_b2     7  -0.0237147
## p_Factor_A_a2_Factor_B_b1_Factor_A_a2_Factor_B_b3     4  -0.0283884
## p_Factor_A_a2_Factor_B_b1_Factor_A_a3_Factor_B_b1    12  -0.0263331
## p_Factor_A_a2_Factor_B_b1_Factor_A_a3_Factor_B_b2     3   0.0031495
## p_Factor_A_a2_Factor_B_b1_Factor_A_a3_Factor_B_b3    92   1.0512691
## p_Factor_A_a2_Factor_B_b2_Factor_A_a2_Factor_B_b3     7  -0.0067561
## p_Factor_A_a2_Factor_B_b2_Factor_A_a3_Factor_B_b1     6  -0.0074707
## p_Factor_A_a2_Factor_B_b2_Factor_A_a3_Factor_B_b2     8   0.0166257
## p_Factor_A_a2_Factor_B_b2_Factor_A_a3_Factor_B_b3    90   1.1003260
## p_Factor_A_a2_Factor_B_b3_Factor_A_a3_Factor_B_b1     3  -0.0006323
## p_Factor_A_a2_Factor_B_b3_Factor_A_a3_Factor_B_b2     6   0.0261821
## p_Factor_A_a2_Factor_B_b3_Factor_A_a3_Factor_B_b3    91   1.0834066
## p_Factor_A_a3_Factor_B_b1_Factor_A_a3_Factor_B_b2     7   0.0212508
## p_Factor_A_a3_Factor_B_b1_Factor_A_a3_Factor_B_b3    90   1.0959735
## p_Factor_A_a3_Factor_B_b2_Factor_A_a3_Factor_B_b3    92   1.0689520
```

```
power_res <- power_twoway_between(design_result) #using default alpha level of .05
power_res$power_A
```

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

```
power_res$power_B
```

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

```
power_res$power_AB
```

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

13.4 Two by two ANOVA, within design

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

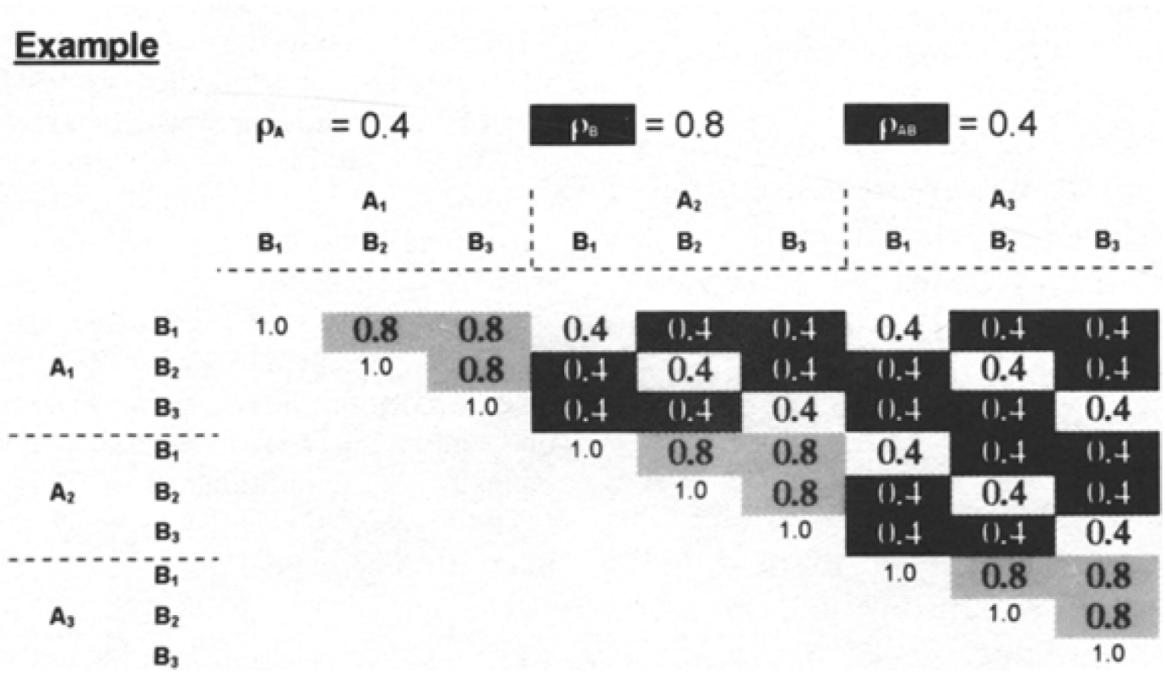


Figure 1. Representation of a correlation matrix for a 3 (A) × 3 (B) RM ANOVA: General form and numeric example. ρ_A and ρ_B represent the average correlation among the A and B (pooled) trials, respectively, and ρ_{AB} represents the average correlation among the AB coefficients having dissimilar levels.

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

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

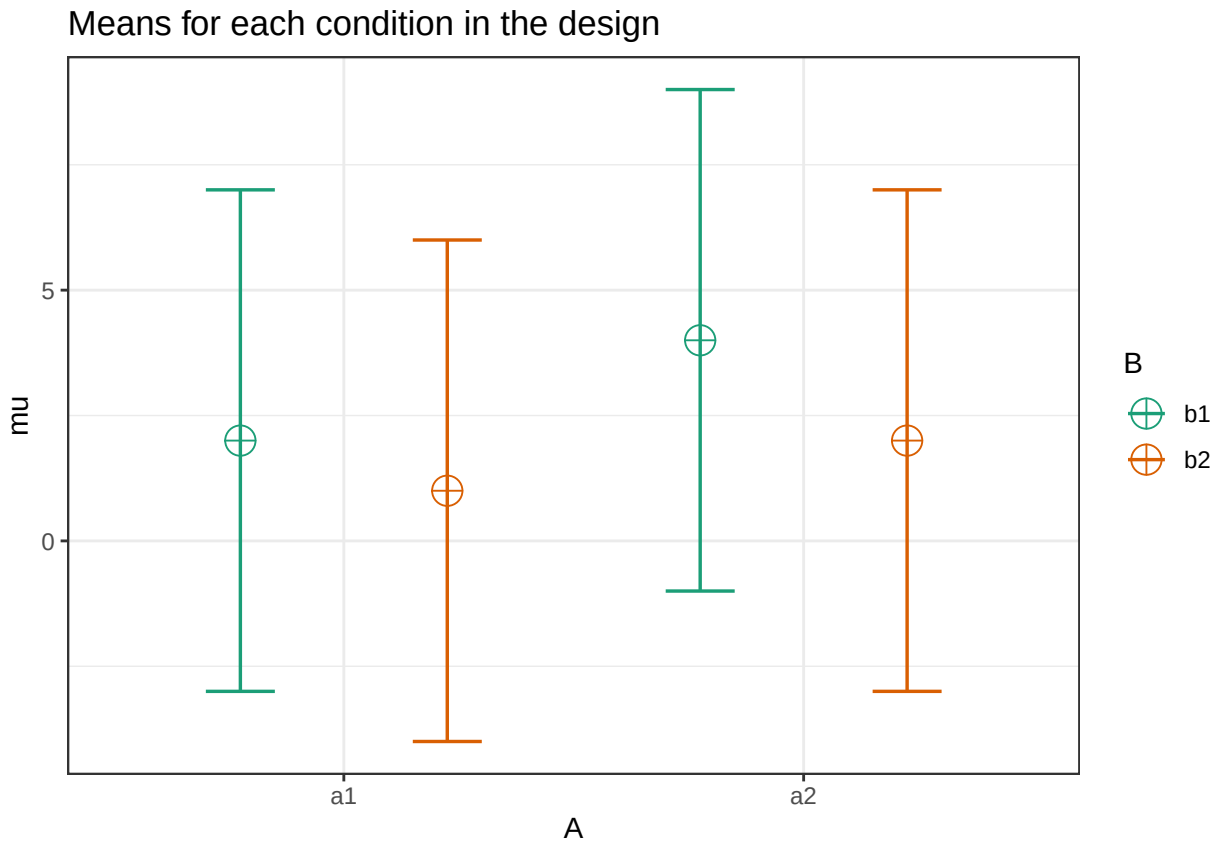
For the main effectof B: $\sigma_e^2 = \sigma^2(1 - \overline{\rho}_B) + \sigma^2(p-1)(\overline{\rho}_A - \overline{\rho}_{AB})$

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

We first simulate a within subjects 2x2 ANOVA design.

```
mu = c(2,1,4,2)
n <- 20
sd <- 5
r <- c(
  0.8, 0.5, 0.4,
       0.4, 0.5,
            0.8
  )
string = "2w*2w"
labelnames = c("A", "a1", "a2", "B", "b1", "b2")
```

```
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)
```

```
## Power and Effect sizes for ANOVA tests
##           power effect_size
## anova_A      28      0.1346
## anova_B      61      0.2615
## anova_A:B    26      0.1190
##
## Power and Effect sizes for contrasts
##                         power effect_size
## p_A_a1_B_b1_A_a1_B_b2    32   -0.342046
## p_A_a1_B_b1_A_a2_B_b1    43    0.403180
## p_A_a1_B_b1_A_a2_B_b2     6   -0.001266
## p_A_a1_B_b2_A_a2_B_b1    63    0.558218
## p_A_a1_B_b2_A_a2_B_b2    17    0.202298
## p_A_a2_B_b1_A_a2_B_b2    75   -0.633718
##
## Within-Subject Factors Included: Check MANOVA Results
```

We can use the `ANOVA_exact` function to evaluate this design.

```
power_res <- ANOVA_exact(design_result = design_result)
```

```
## Power and Effect sizes for ANOVA tests
##     power partial_eta_squared cohen_f non_centrality
## A   26.92              0.0952  0.3244              2
## B   64.23              0.2400  0.5620              6
## A:B 26.92              0.0952  0.3244              2
##
## Power and Effect sizes for contrasts
##                        power effect_size
## p_A_a1_B_b1_A_a1_B_b2 26.92     -0.3162
## p_A_a1_B_b1_A_a2_B_b1 39.70      0.4000
## p_A_a1_B_b1_A_a2_B_b2  5.00      0.0000
## p_A_a1_B_b2_A_a2_B_b1 64.23      0.5477
## p_A_a1_B_b2_A_a2_B_b2 13.60      0.2000
## p_A_a2_B_b1_A_a2_B_b2 76.52     -0.6325
```

```
power_res$power_A
```

```
## NULL
```

```
power_res$power_B
```

```
## NULL
```

```
power_res$power_AB
```

```
## NULL
```

We can use the code by Abelson (2019) to produce the same results.

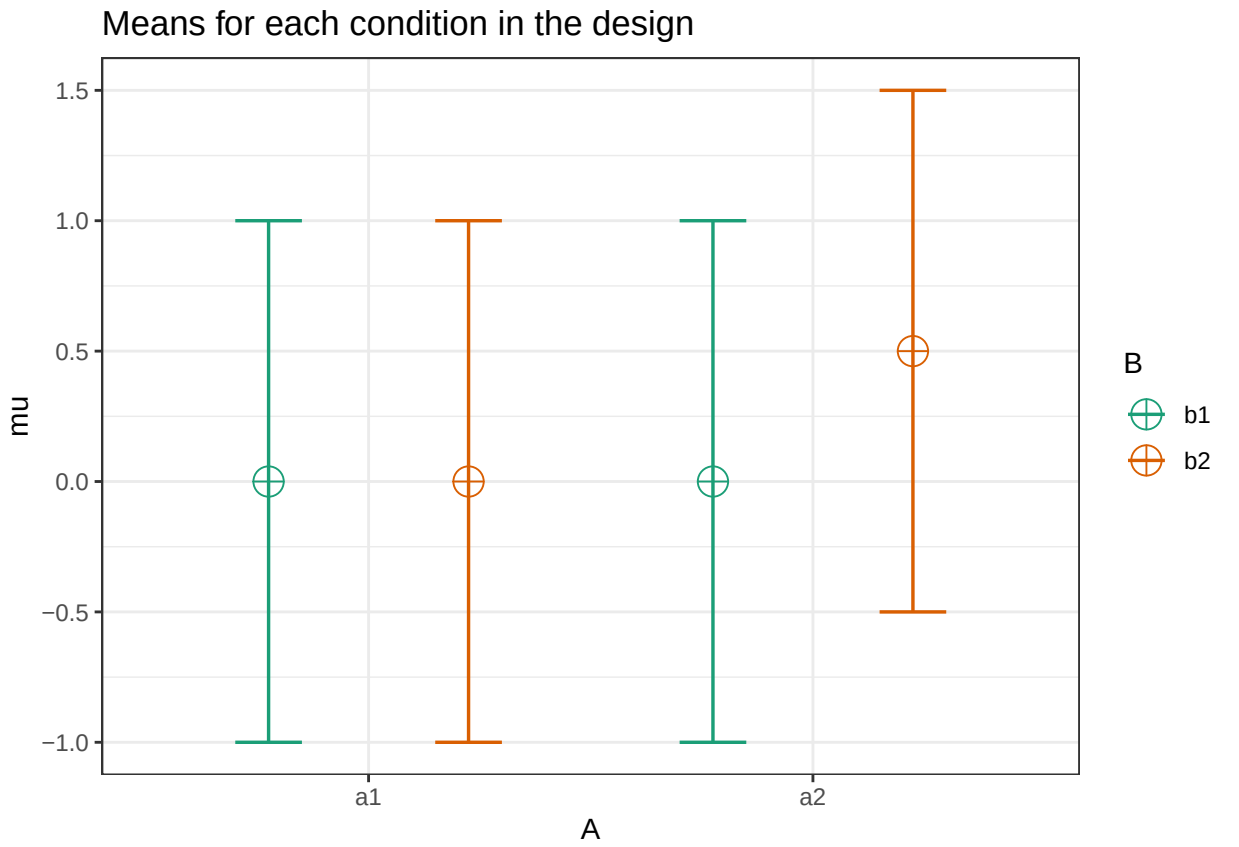
```
win2F(m1.1=2, m2.1=1, m1.2=4, m2.2=2,
      s1.1=5, s2.1=5, s1.2=5, s2.2=5,
      r12=0.8, r13=0.5, r14=0.4,
               r23=0.4, r24=0.5,
                        r34=0.8,
      n=20)
```

Chapter 14

Power curves

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 can be calculated based on simulations for any design, we will use the analytic solution to calculate the power of ANOVA designs because these calculations are much faster. The basic approach is to calculate power for a specific pattern of means, a specific effect size, a given alpha level, and a specific pattern of correlations. This is one example:

```
#2x2 design
string = "2w*2w"
mu = c(0,0,0,0.5)
n <- 20
sd <- 1
r <- 0.5
labelnames = c("A", "a1", "a2", "B", "b1", "b2")
design_result <- ANOVA_design(design = string,
                              n = n,
                              mu = mu,
                              sd = sd,
                              r = r,
                              labelnames = labelnames)
```




```
power_res <- ANOVA_exact(design_result)
```

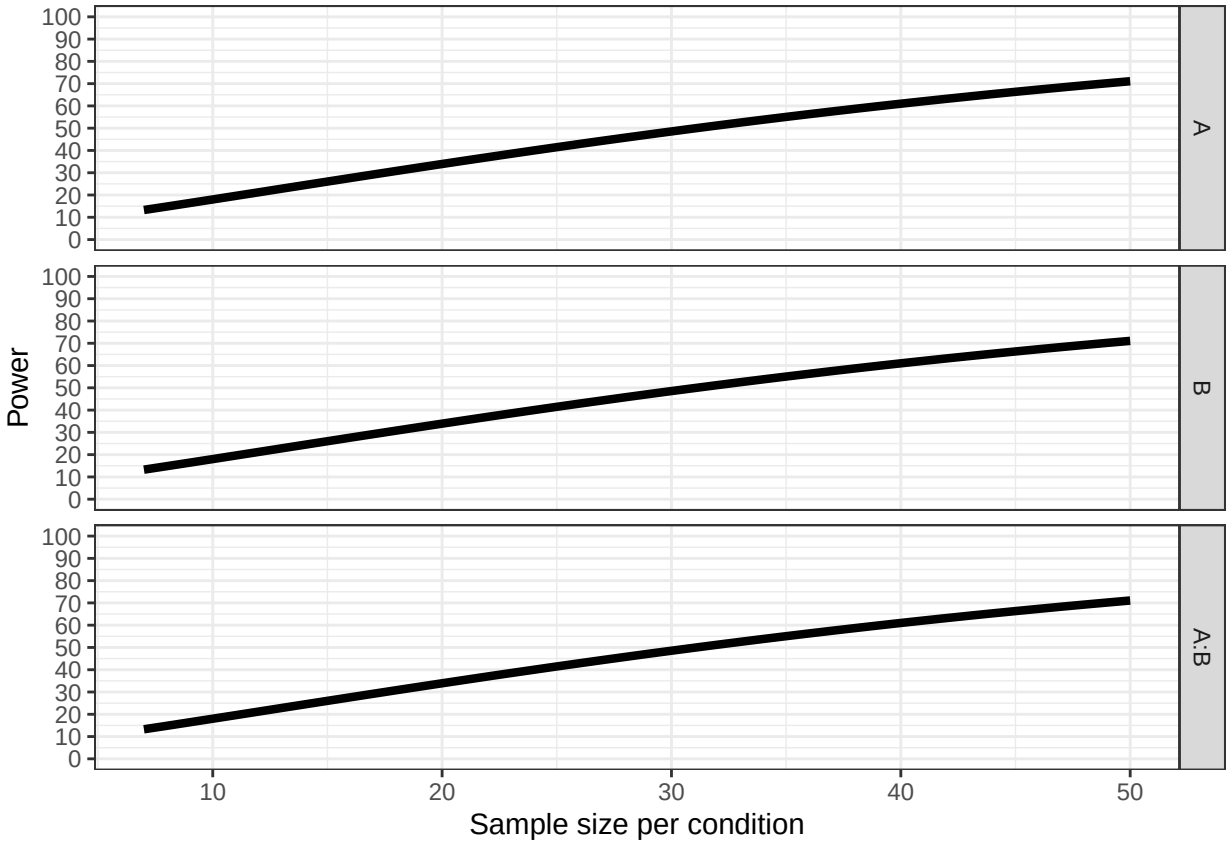
```
## Power and Effect sizes for ANOVA tests
##     power partial_eta_squared cohen_f non_centrality
## A   32.36              0.1163  0.3627            2.5
## B   32.36              0.1163  0.3627            2.5
## A:B 32.36              0.1163  0.3627            2.5
##
## Power and Effect sizes for contrasts
##                         power effect_size
## p_A_a1_B_b1_A_a1_B_b2  5.00         0.0
## p_A_a1_B_b1_A_a2_B_b1  5.00         0.0
## p_A_a1_B_b1_A_a2_B_b2 56.45         0.5
## p_A_a1_B_b2_A_a2_B_b1  5.00         0.0
## p_A_a1_B_b2_A_a2_B_b2 56.45         0.5
## p_A_a2_B_b1_A_a2_B_b2 56.45         0.5
```

```
power_res$main_results
```

```
##     power partial_eta_squared cohen_f non_centrality
## A   32.36              0.1163  0.3627            2.5
## B   32.36              0.1163  0.3627            2.5
## A:B 32.36              0.1163  0.3627            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 = 2 to max_, a variable you can specify in the function).

```
p_a <- plot_power(design_result,
                  max_n = 50)
```

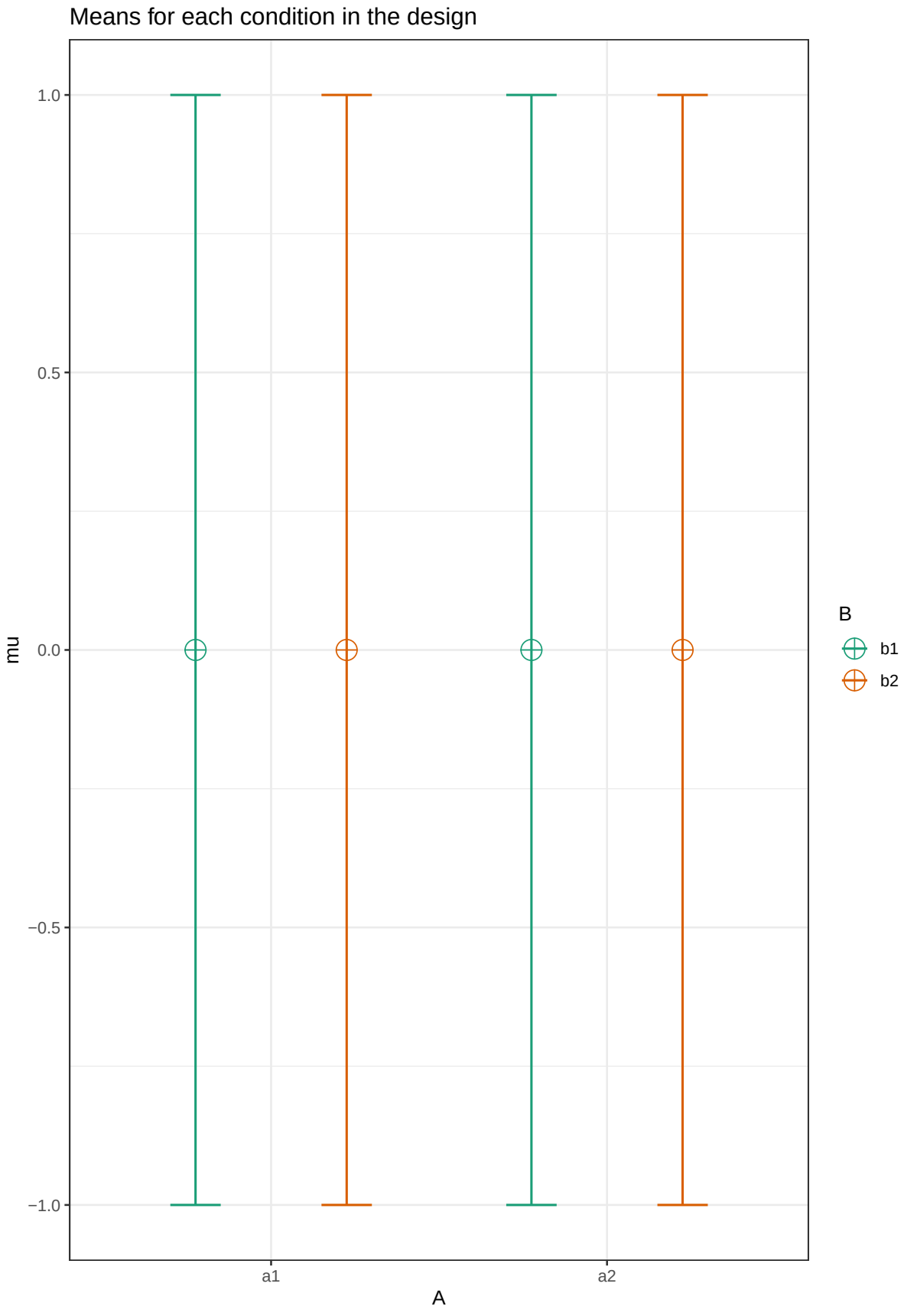


Chapter 15

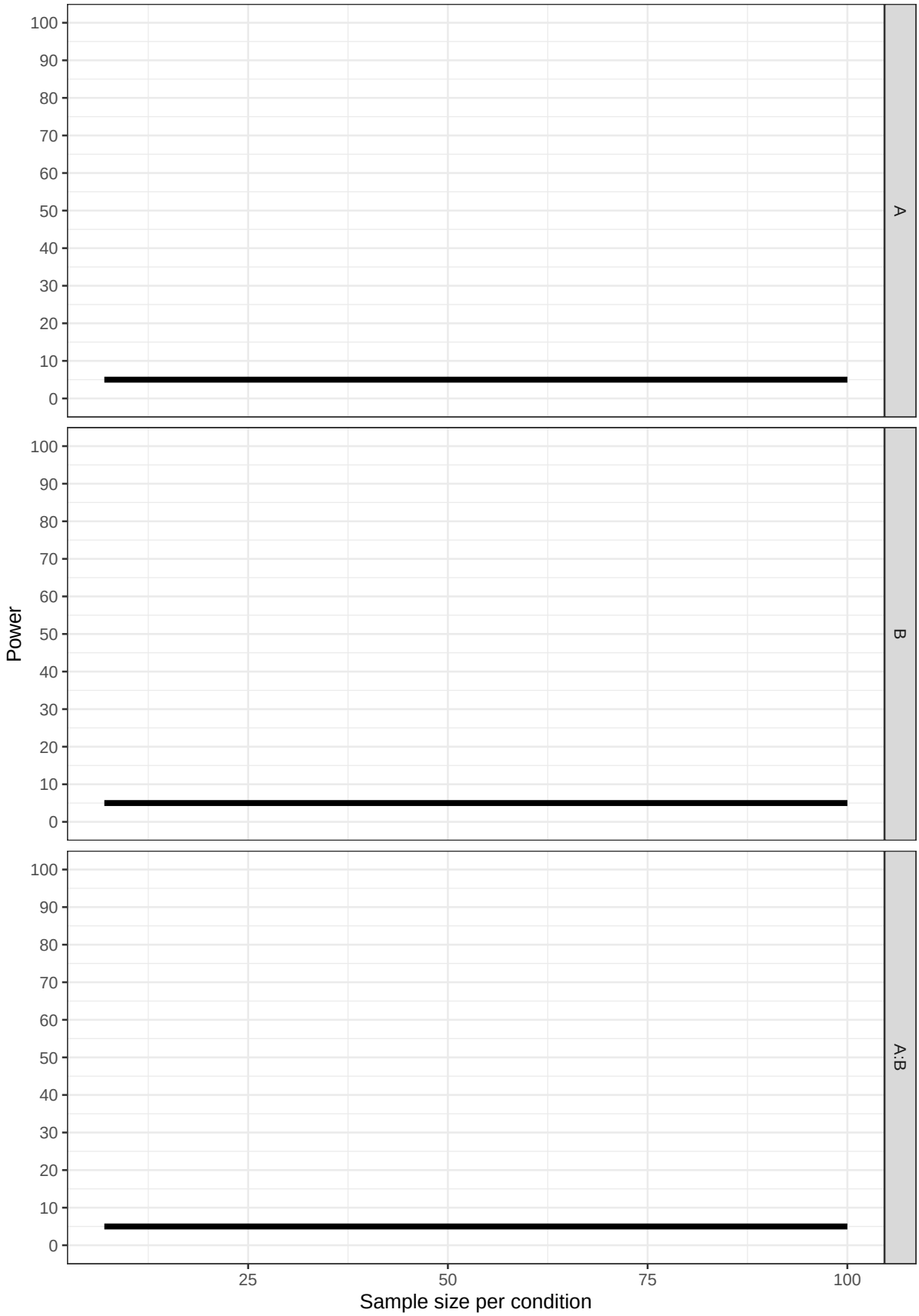
Explore increase in effect size for moderated interactions.

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.

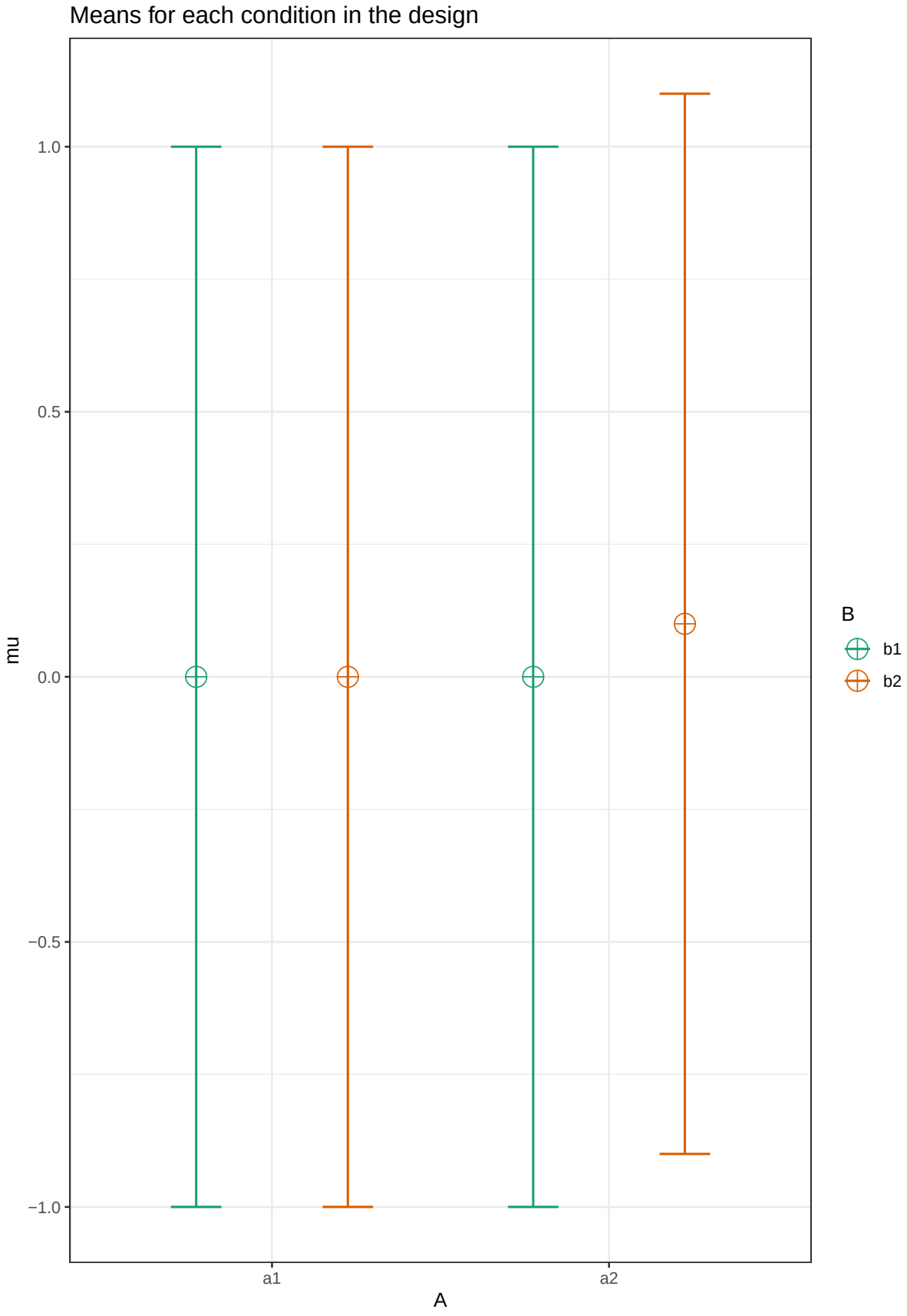
```
string <- "2w*2w"
labelnames = c("A", "a1", "a2", "B", "b1", "b2")
design_result <- ANOVA_design(design = string,
                              n = 20,
                              mu = c(0,0,0,0.0),
                              sd = 1,
                              r = 0.5,
                  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


Means for each condition in the design
1.0
0.5
0.0
−0.5
−1.0
mu
a1
a2
A
B
b1
b2

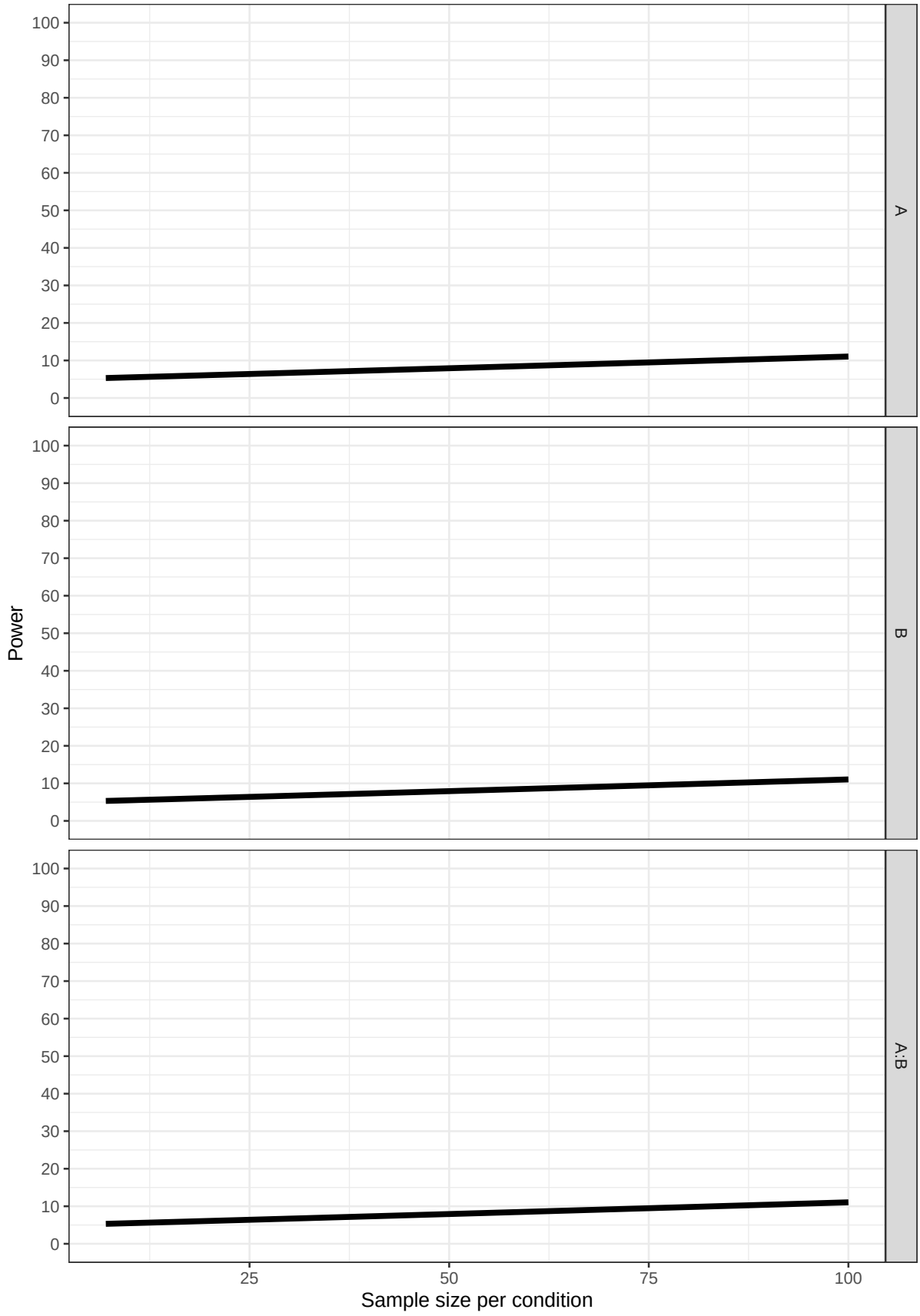
```
p_a <- plot_power(design_result,
                  max_n = 100)
```


Power
100
90
80
70
60
50
40
30
20
10
0
A
B
A:B
25
50
75
100
Sample size per condition

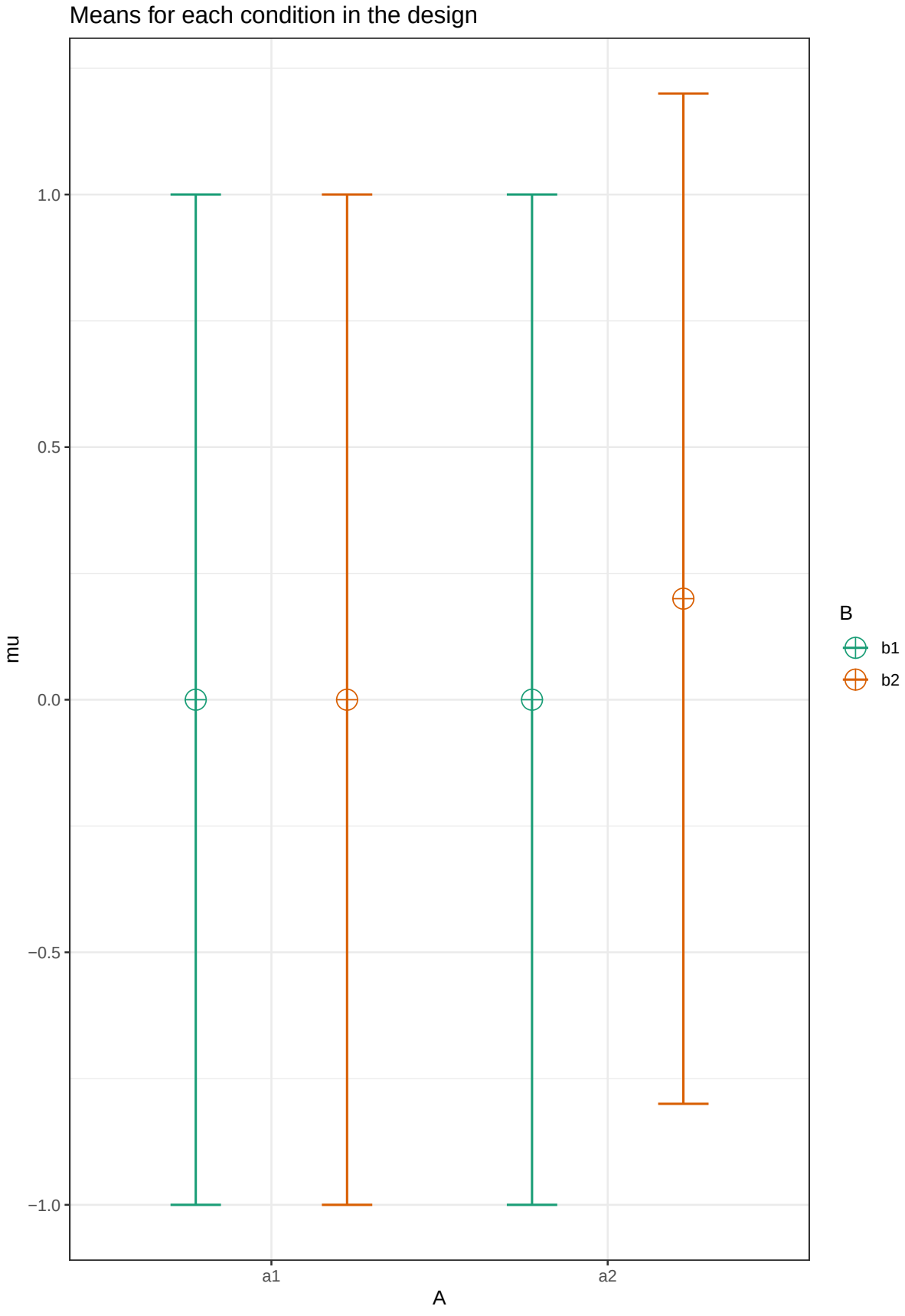
```
design_result <- ANOVA_design(design = string,
                              n = 20,
                              mu = c(0,0,0,0.1),
                              sd = 1,
                              r = 0.5,
                              labelnames = labelnames)
```


Means for each condition in the design
mu
1.0
0.5
0.0
−0.5
−1.0
a1
a2
A
B
b1
b2

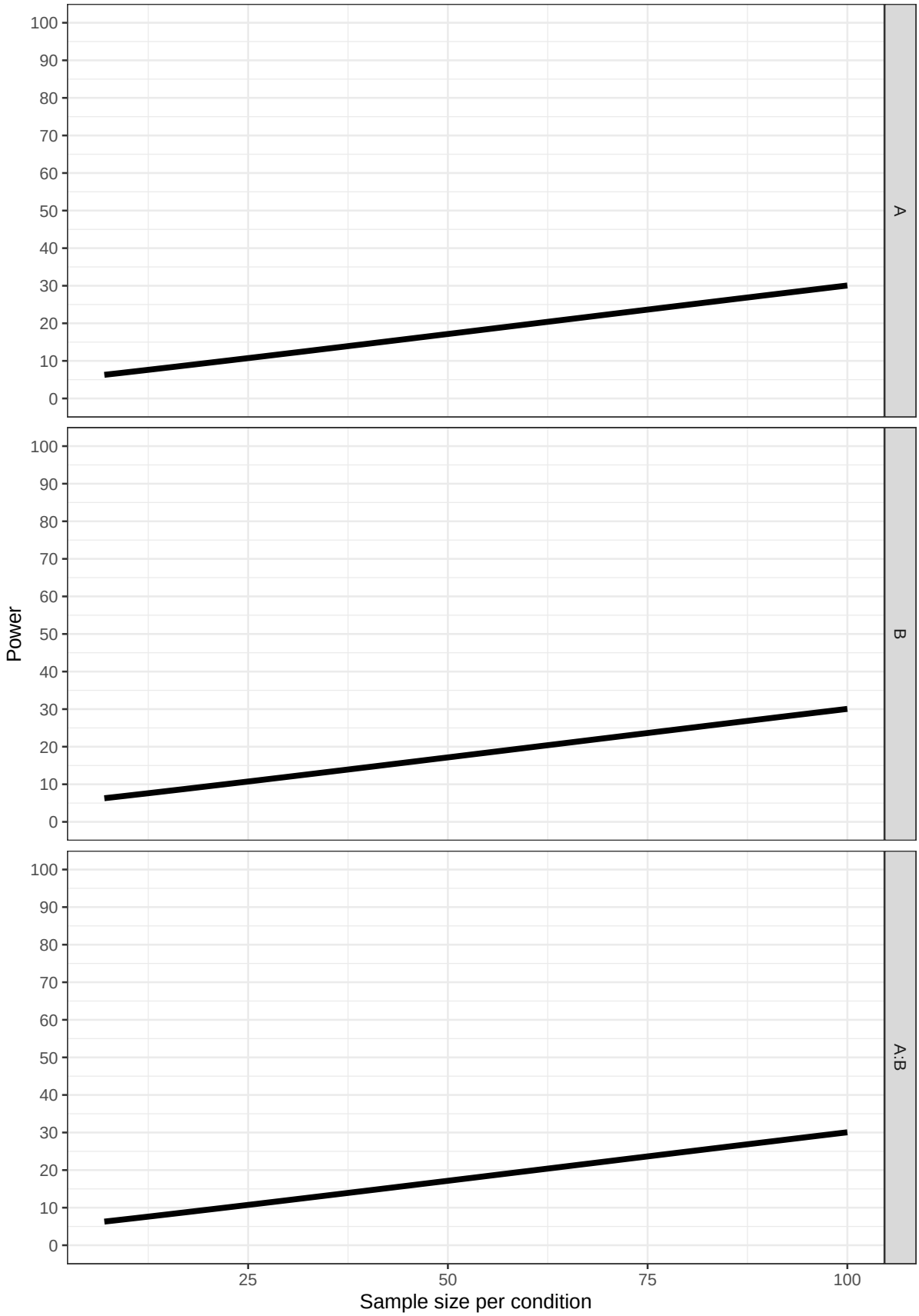
```
p_b <- plot_power(design_result,
                  max_n = 100)
```


Power
100
90
80
70
60
50
40
30
20
10
0
A
B
A:B
25
50
75
100
Sample size per condition

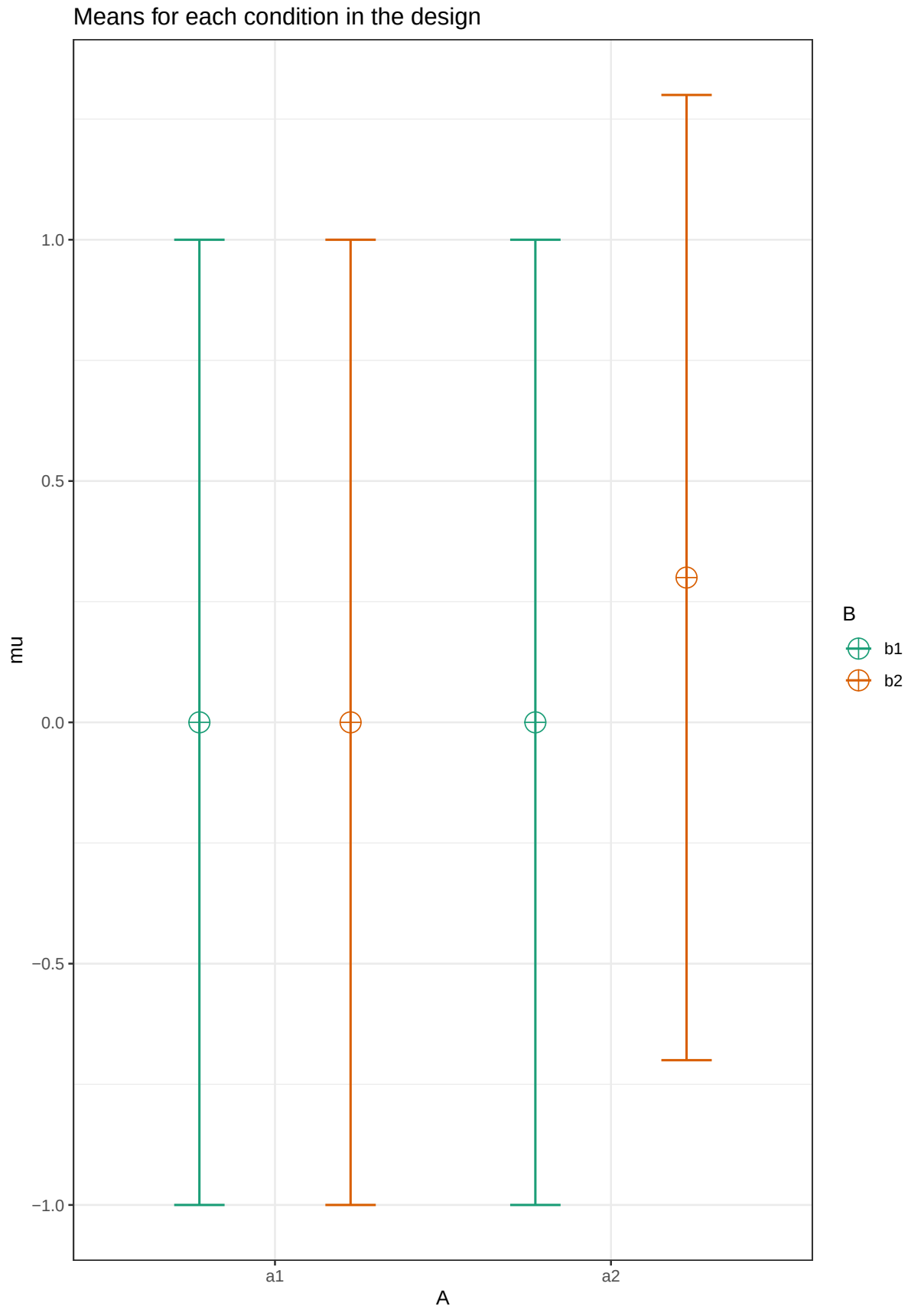
```
design_result <- ANOVA_design(design = string,
                              n = 20,
                              mu = c(0,0,0,0.2),
                              sd = 1,
                              r = 0.5,
                              labelnames = labelnames)
```


Means for each condition in the design
mu
1.0
0.5
0.0
−0.5
−1.0
a1
a2
A
B
b1
b2

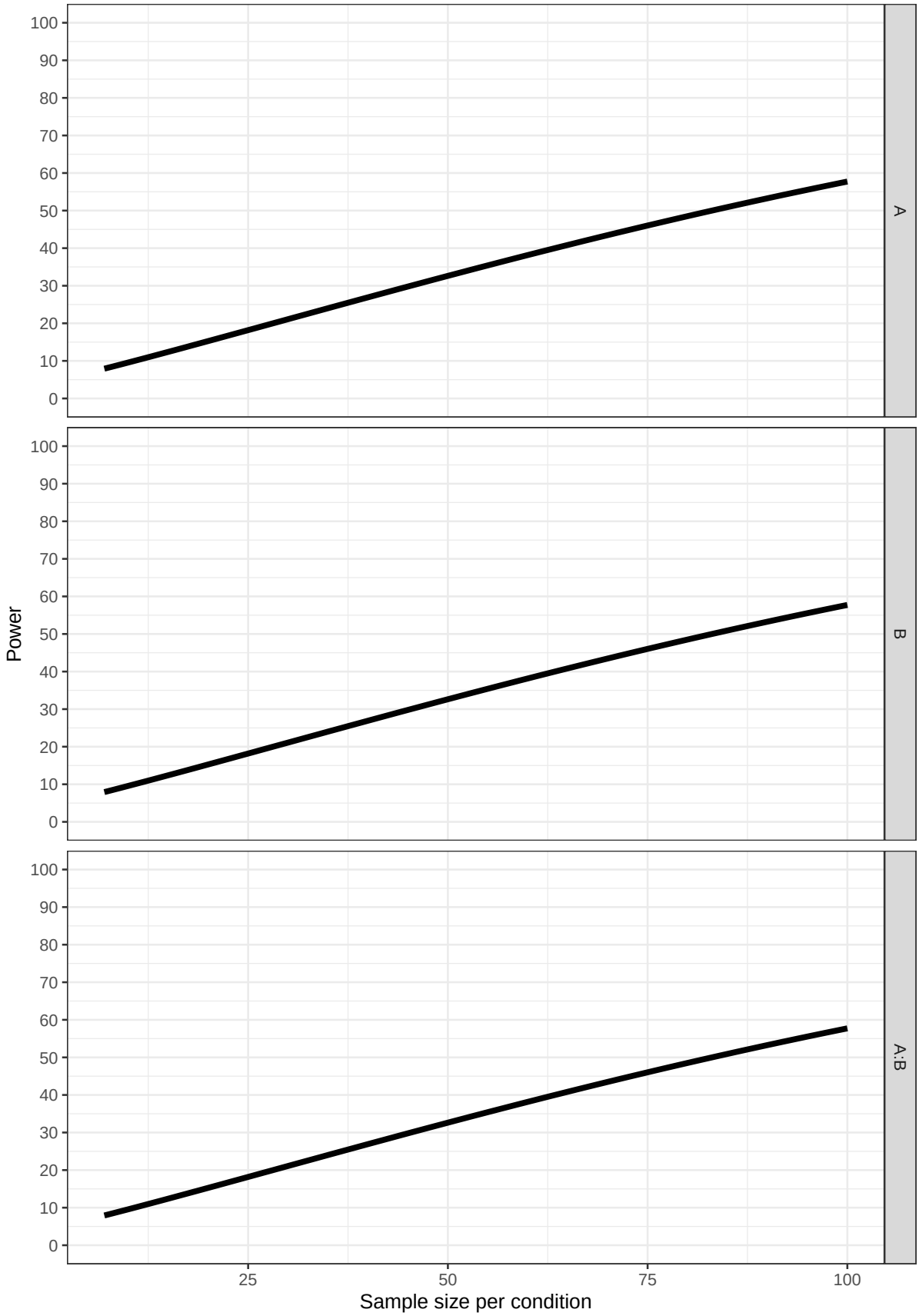
```
p_c <- plot_power(design_result,
                  max_n = 100)
```


Power
Sample size per condition
A
B
A:B
0
10
20
30
40
50
60
70
80
90
100
25
50
75
100

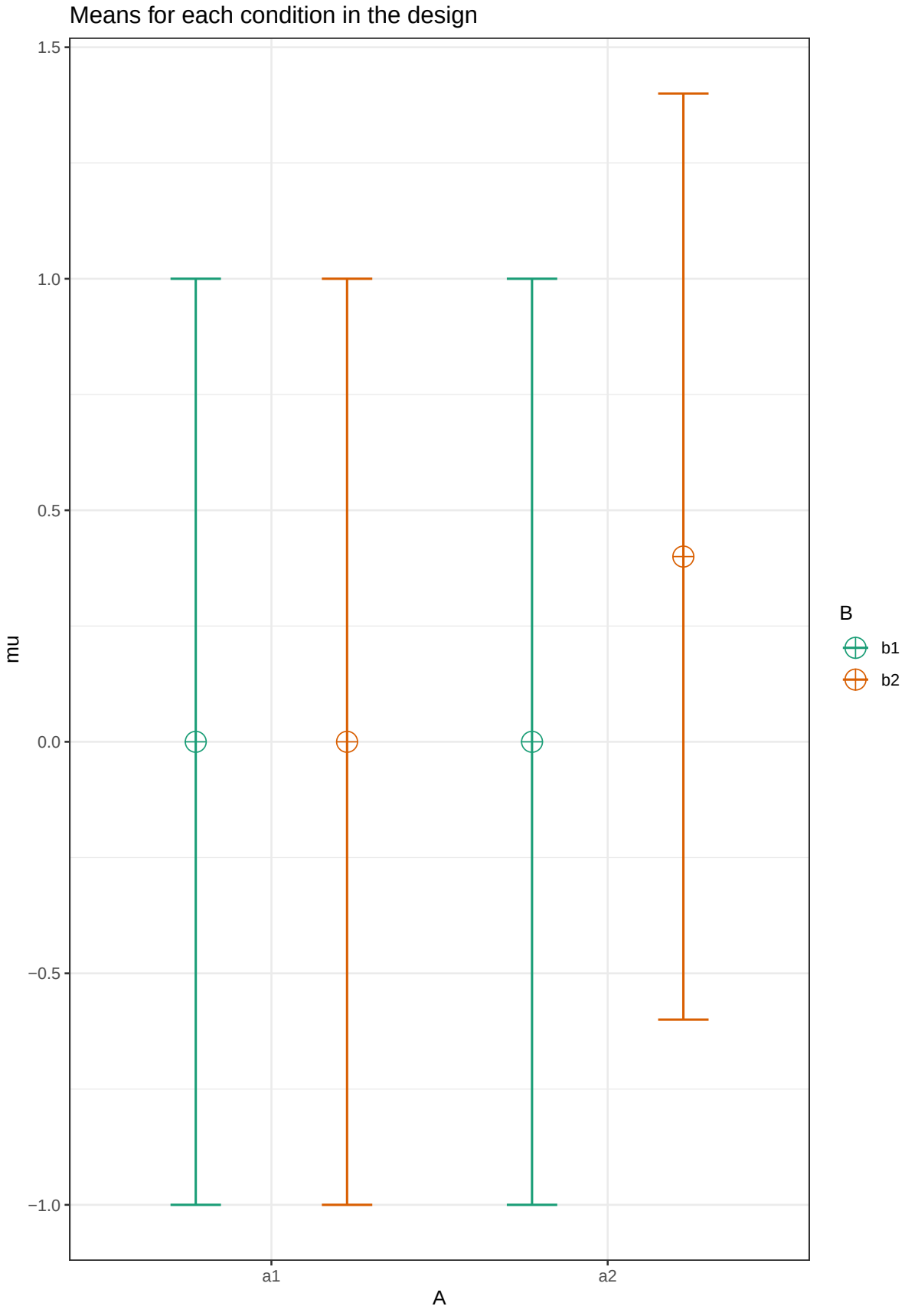
```
design_result <- ANOVA_design(design = string,
                              n = 20,
                              mu = c(0,0,0,0.3),
                              sd = 1,
                              r = 0.5,
                              labelnames = labelnames)
```


Means for each condition in the design
mu
1.0
0.5
0.0
−0.5
−1.0
a1
a2
A
B
b1
b2

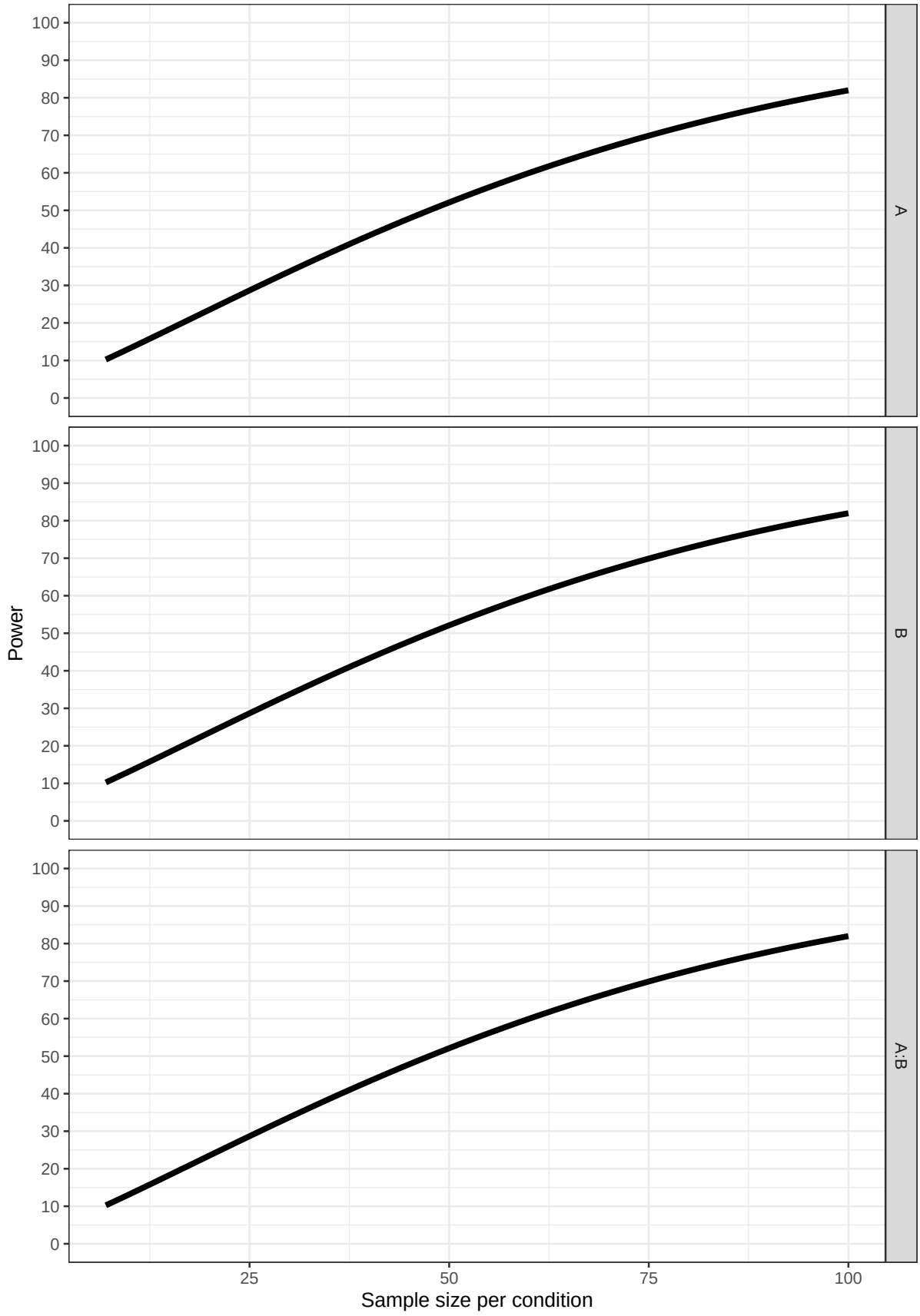
```
p_d <- plot_power(design_result,
                  max_n = 100)
```


Power
100
90
80
70
60
50
40
30
20
10
0
A
B
A:B
25
50
75
100
Sample size per condition

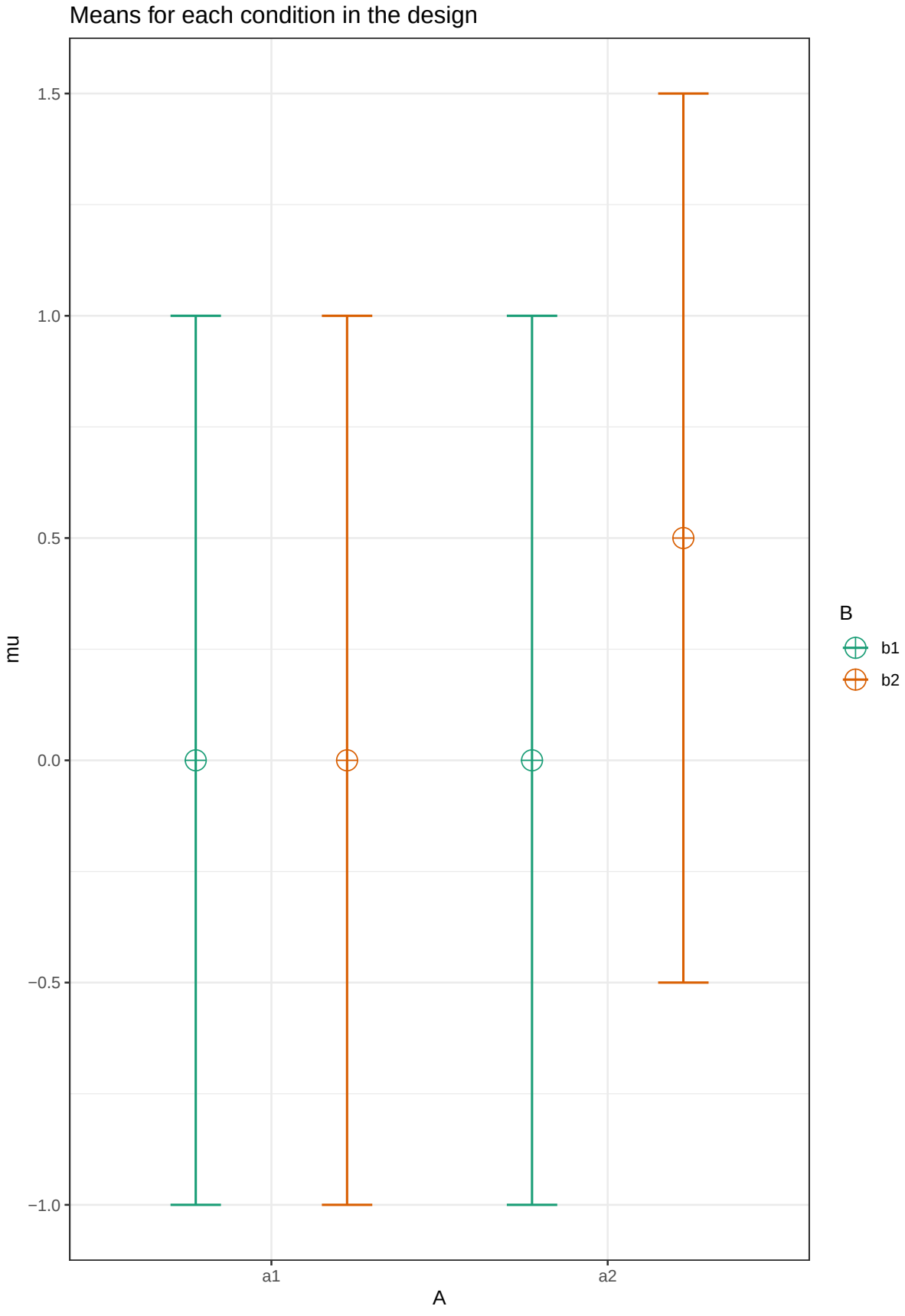
```
design_result <- ANOVA_design(design = string,
                              n = 20,
                              mu = c(0,0,0,0.4),
                              sd = 1,
                              r = 0.5,
                              labelnames = labelnames)
```


Means for each condition in the design
1.5
1.0
0.5
0.0
−0.5
−1.0
mu
a1
a2
A
B
b1
b2

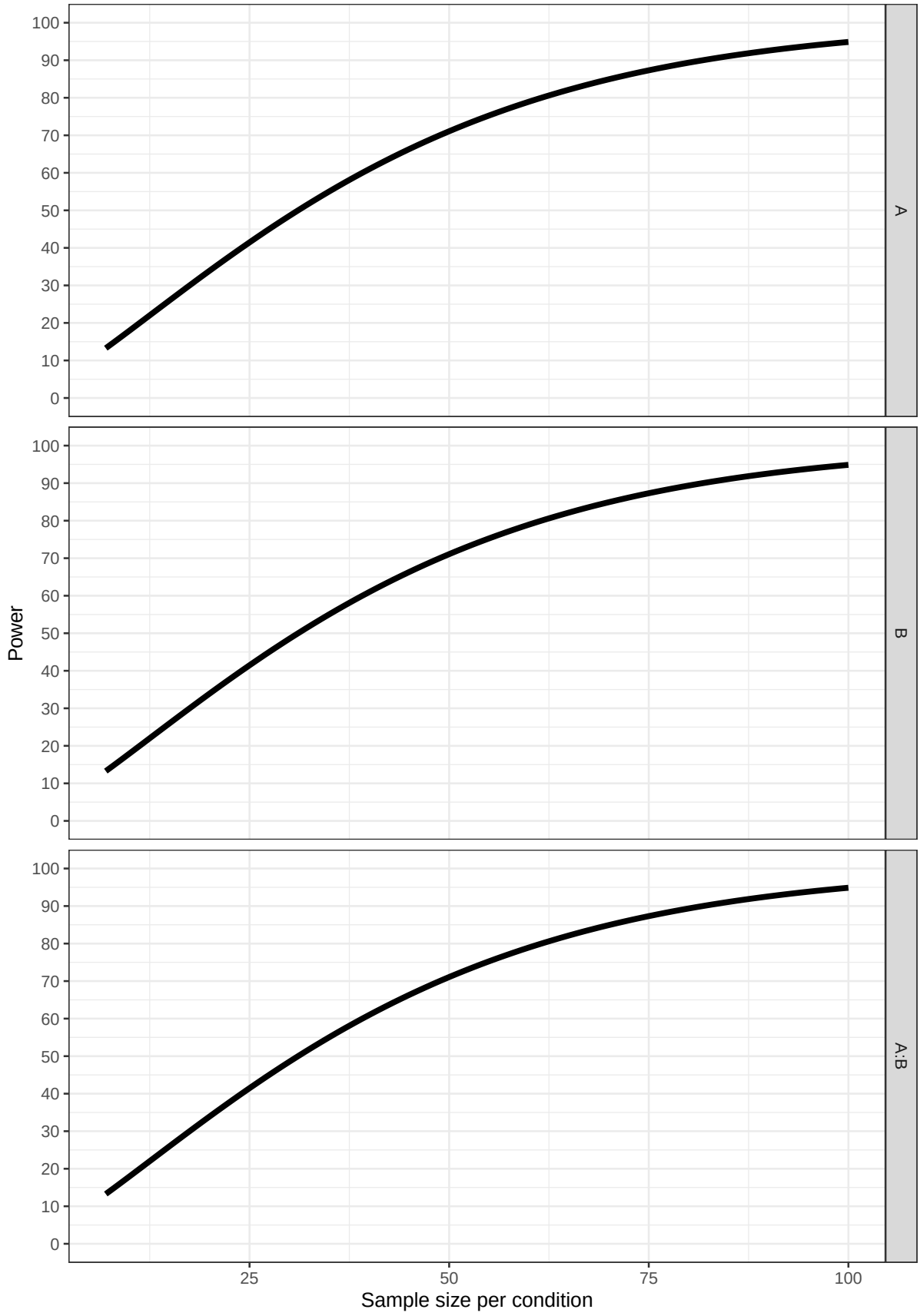
```
p_e <- plot_power(design_result,
                  max_n = 100)
```


Power
100
90
80
70
60
50
40
30
20
10
0
A
B
A:B
25
50
75
100
Sample size per condition

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


Means for each condition in the design
1.5
1.0
0.5
0.0
−0.5
−1.0
mu
a1
a2
A
B
b1
b2

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

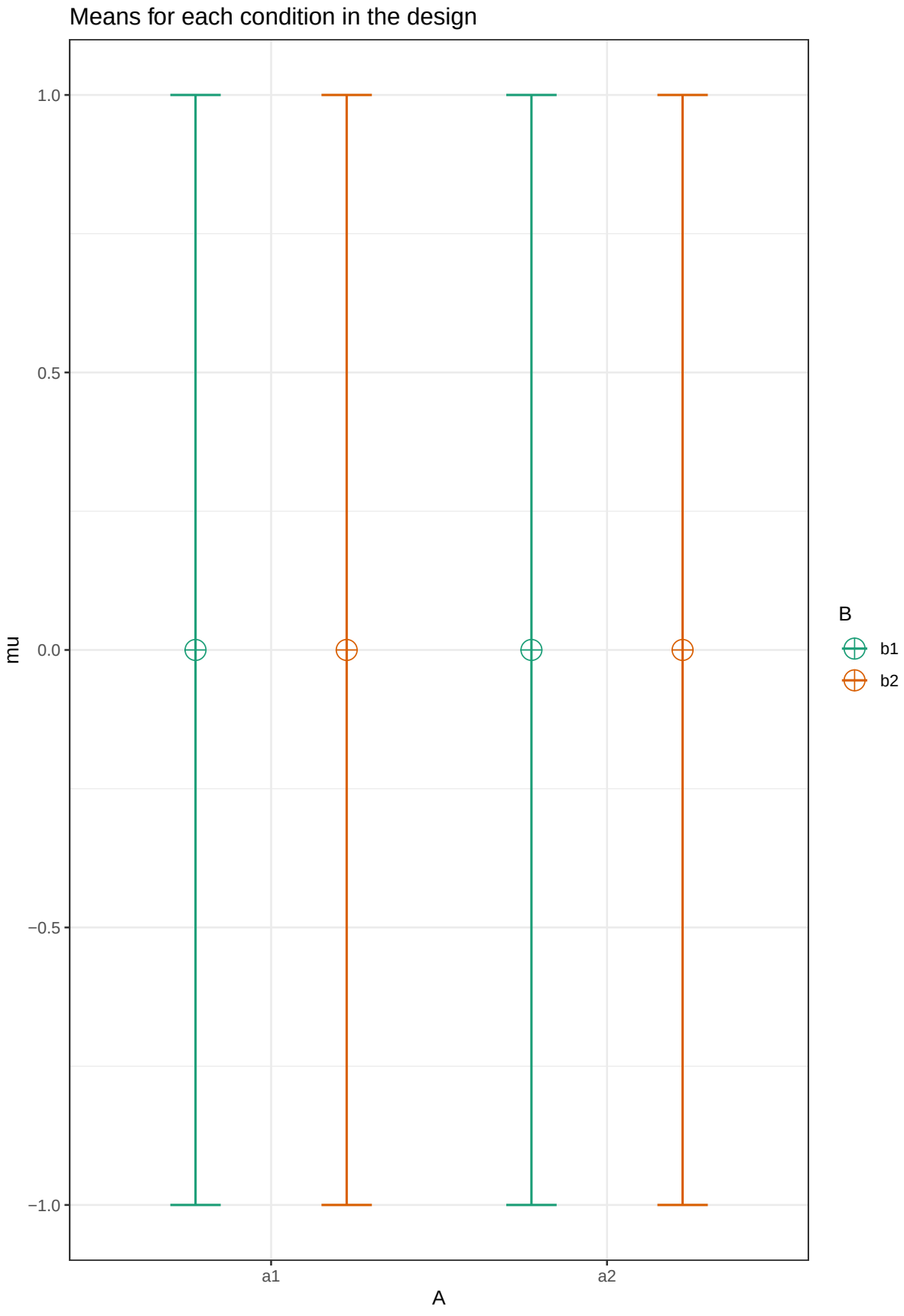

Power
100
90
80
70
60
50
40
30
20
10
0
A
B
A:B
25
50
75
100
Sample size per condition

```
# Create long format dataframe
#zzz <- 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)
#zzz <- cbind(zzz,seq(1,length(zzz$design)))
#colnames(zzz)[1] <- "design"
#colnames(zzz)[6] <- "ID"
#zzz <- melt(zzz, id.vars = c("ID", "design", "n_vec"), measure.vars = c("power_A", "power_B", "power_A
# Plot data using facets, split by factors and interaction, and design
#ggplot(data=zzz, aes(x = n_vec, y = value)) +
#  geom_line( size=1.5) +
#  scale_x_continuous(limits = c(0, max(zzz$n_vec))) +
#  scale_y_continuous(limits = c(0, 100)) +
#  theme_bw() +
#  labs(x="Sample size", y = "Power") +
#  facet_grid(design~variable)
```

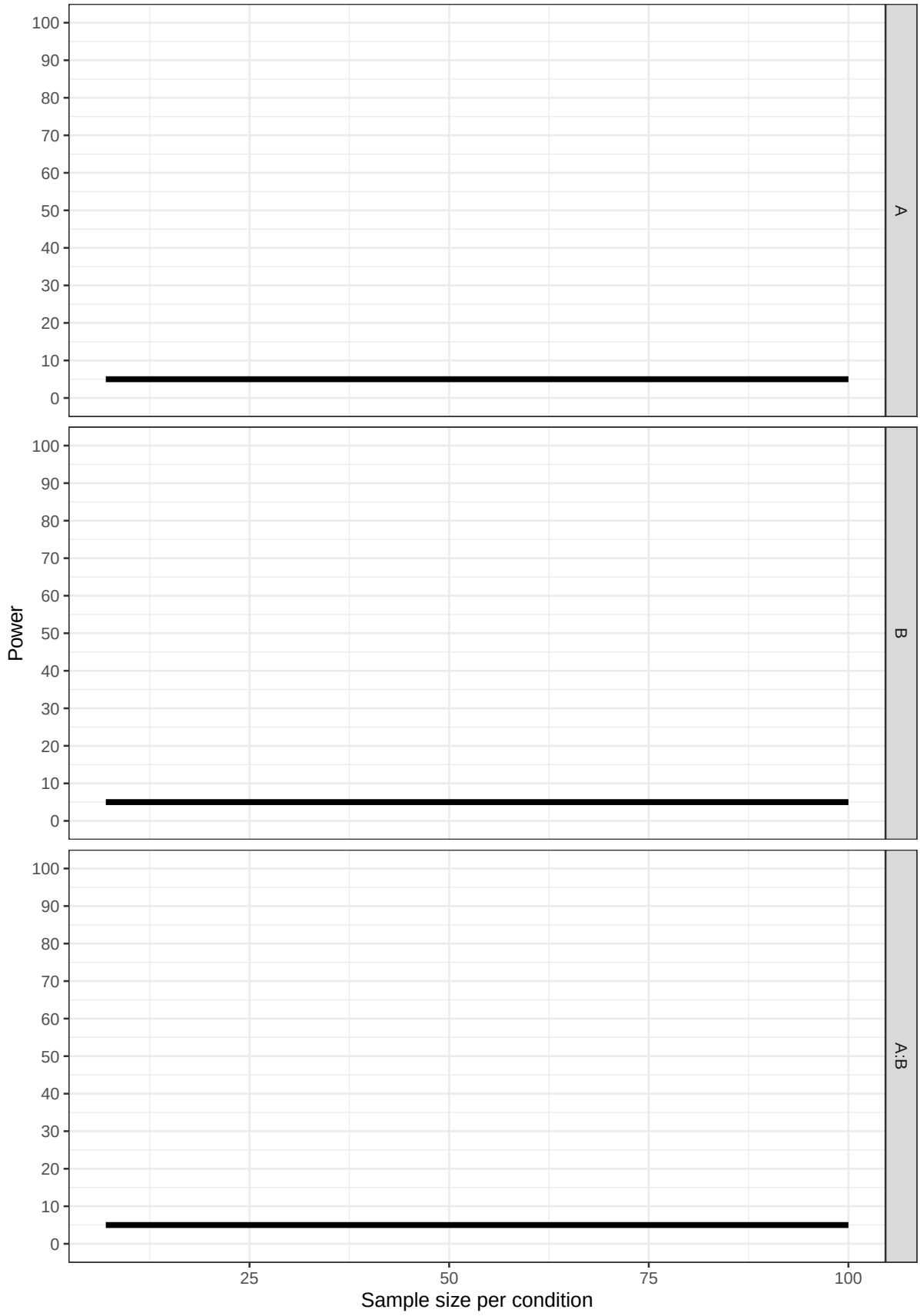
15.1 Explore increase in effect size for cross-over interactions.

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

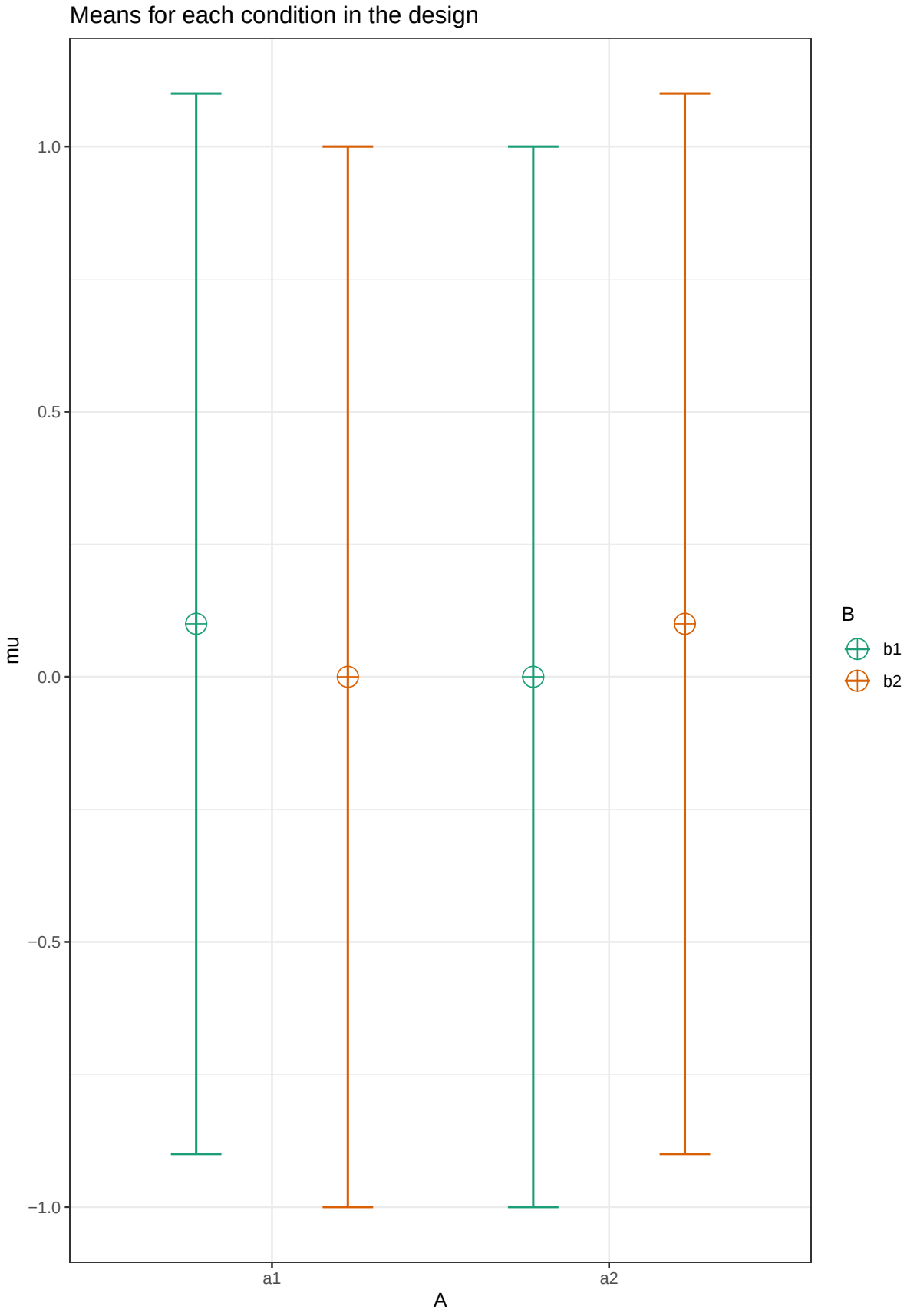
```
design_result <- ANOVA_design(design = string,
                              n = 20,
                              mu = c(0,0,0,0.0),
                              sd = 1,
                              r = 0.5,
                              labelnames = labelnames)
```


Means for each condition in the design
1.0
0.5
0.0
−0.5
−1.0
mu
a1
a2
A
B
b1
b2

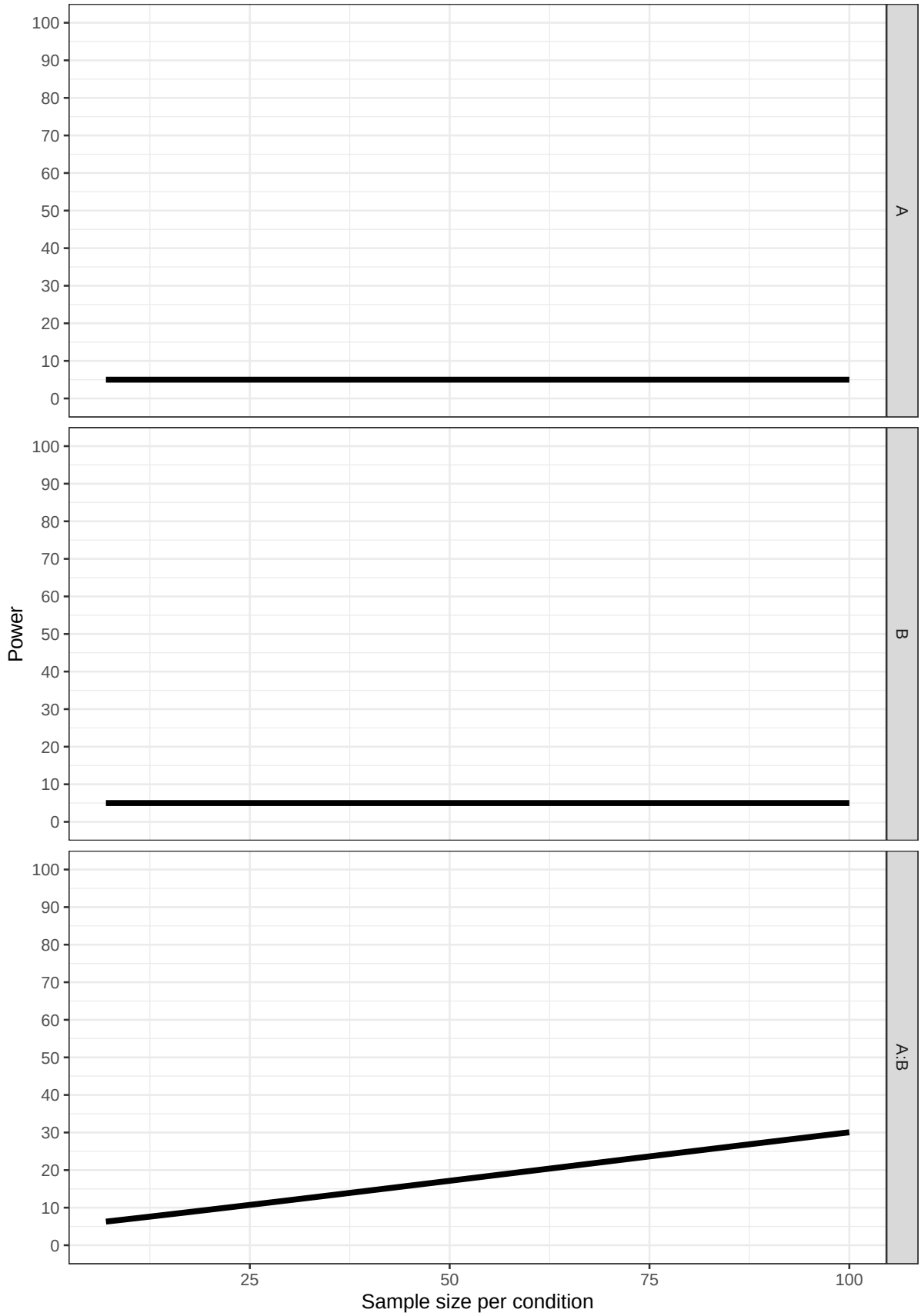
```
p_a <- plot_power(design_result,
                  max_n = 100)
```


Power
100
90
80
70
60
50
40
30
20
10
0
A
B
A:B
25
50
75
100
Sample size per condition

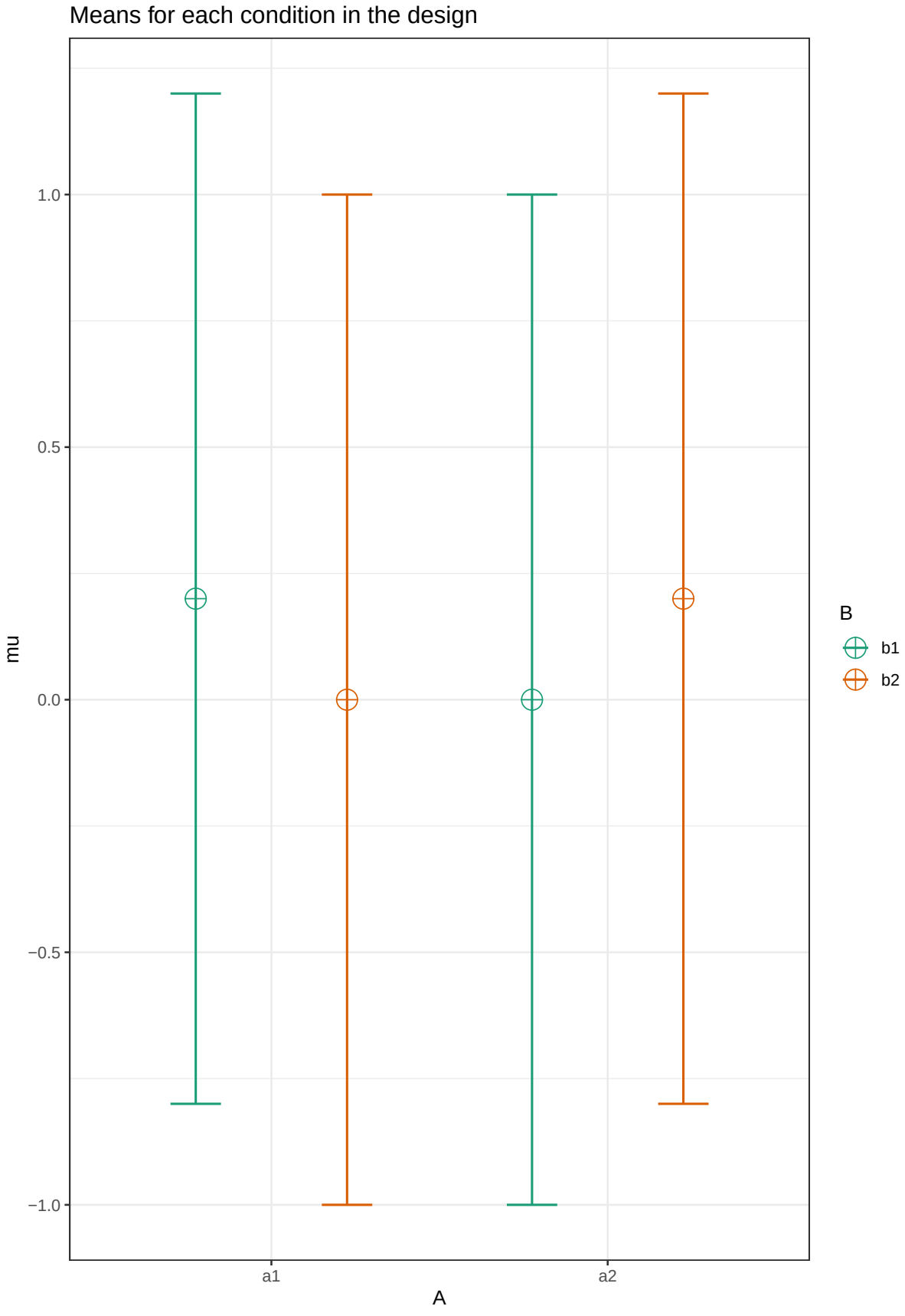
```
design_result <- ANOVA_design(design = string,
                              n = 20,
                              mu = c(0.1,0,0,0.1),
                              sd = 1,
                              r = 0.5,
                              labelnames = labelnames)
```


Means for each condition in the design
mu
1.0
0.5
0.0
−0.5
−1.0
a1
a2
A
B
b1
b2

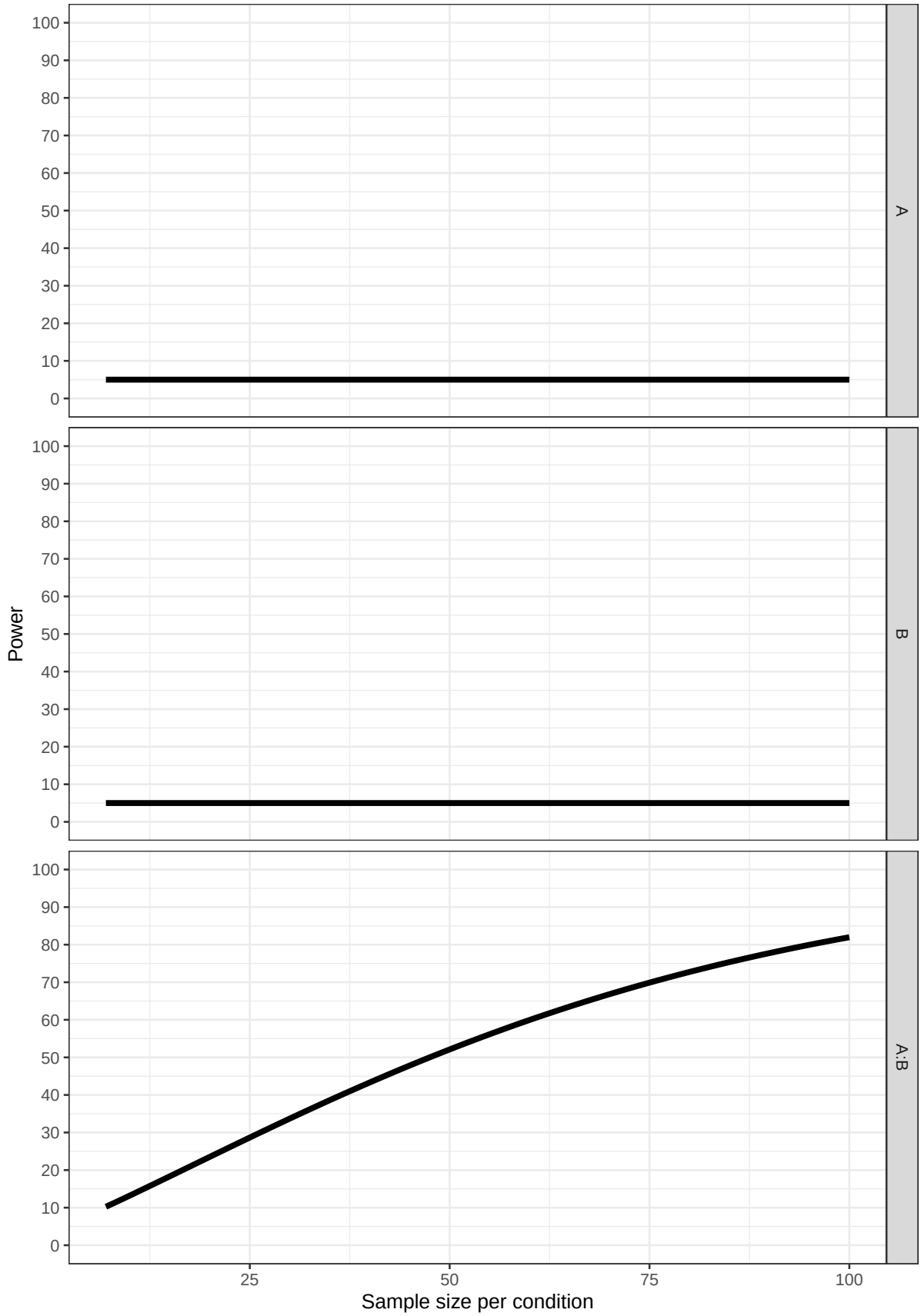
```
p_b <- plot_power(design_result,
                  max_n = 100)
```


Power
100
90
80
70
60
50
40
30
20
10
0
A
B
A:B
25
50
75
100
Sample size per condition

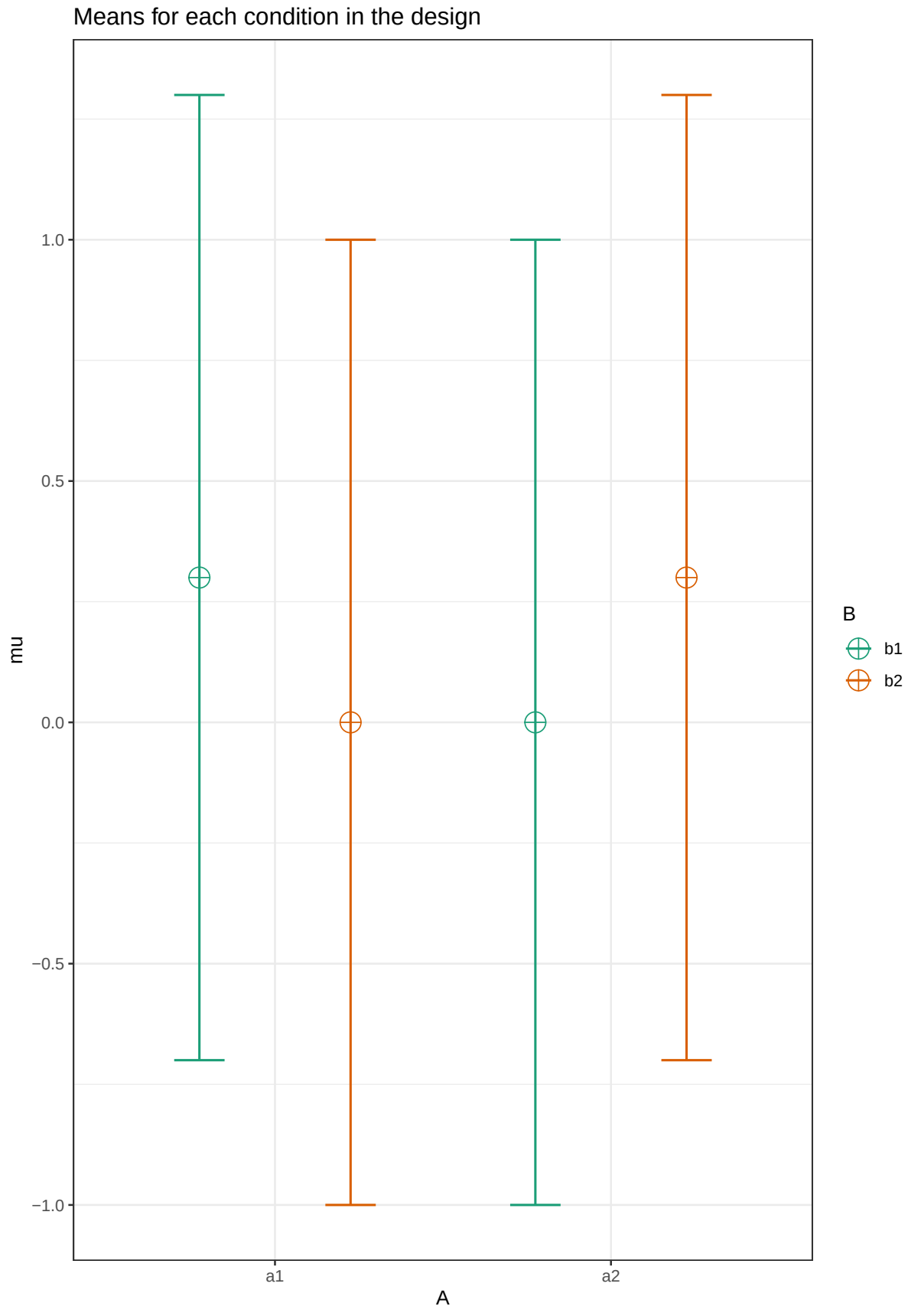
```
design_result <- ANOVA_design(design = string,
                              n = 20,
                              mu = c(0.2,0,0,0.2),
                              sd = 1,
                              r = 0.5,
                              labelnames = labelnames)
```


Means for each condition in the design
mu
1.0
0.5
0.0
−0.5
−1.0
a1
a2
A
B
b1
b2

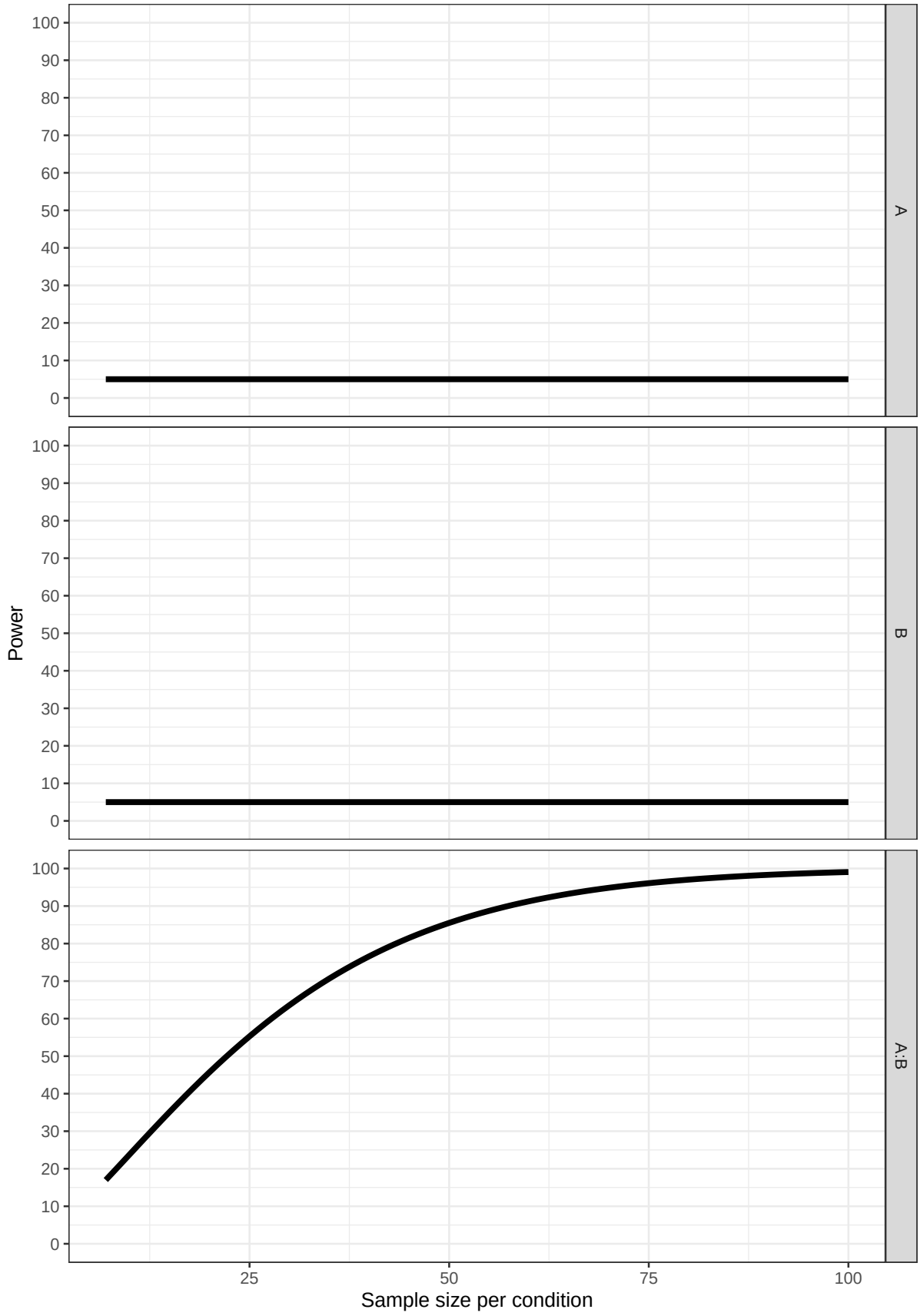
```
p_c <- plot_power(design_result,
                  max_n = 100)
```


Power
100
90
80
70
60
50
40
30
20
10
0
A
B
A:B
25
50
75
100
Sample size per condition

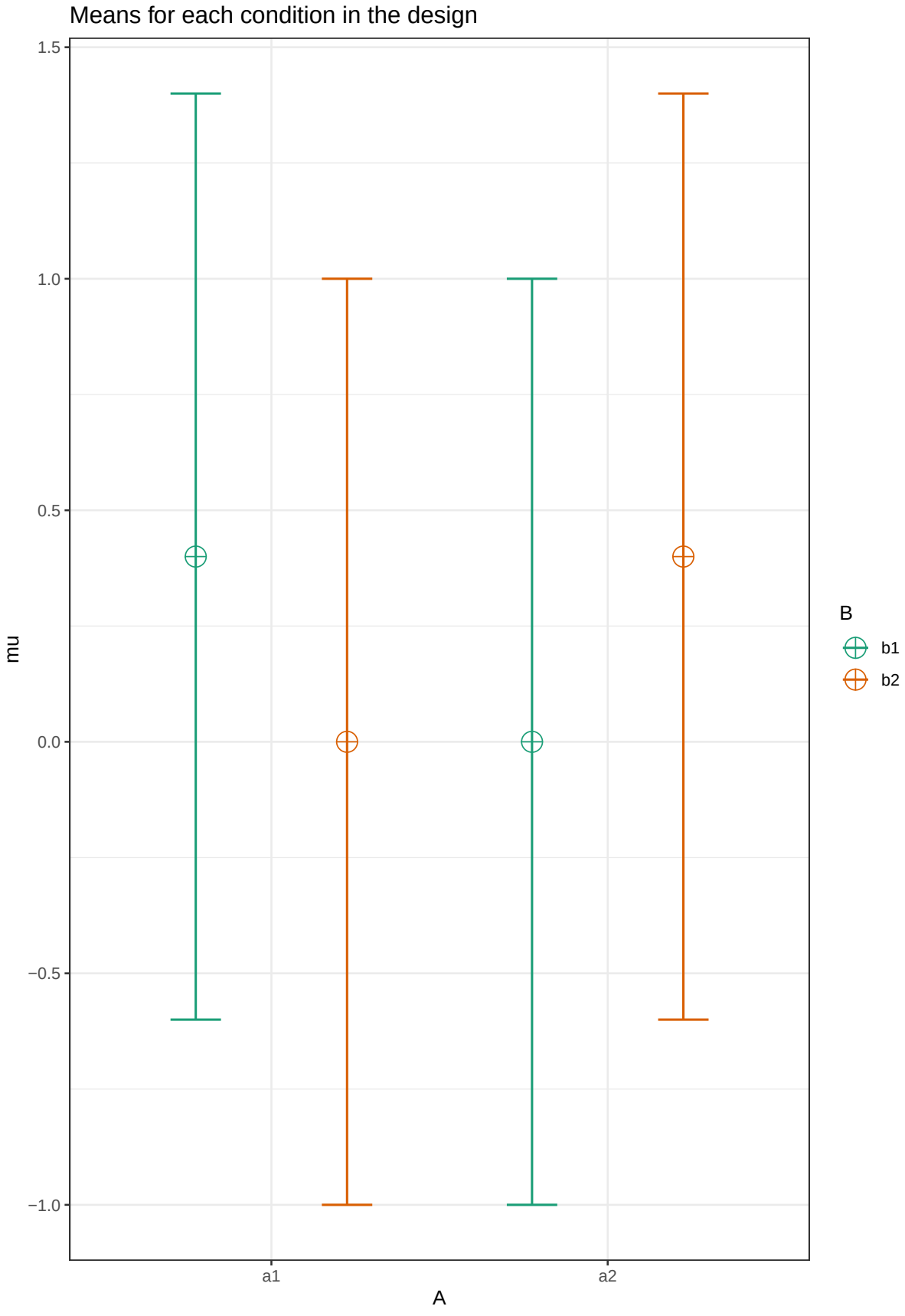
```
design_result <- ANOVA_design(design = string,
                              n = 20,
                              mu = c(0.3,0,0,0.3),
                              sd = 1,
                              r = 0.5,
                              labelnames = labelnames)
```


Means for each condition in the design
mu
1.0
0.5
0.0
−0.5
−1.0
a1
a2
A
B
b1
b2

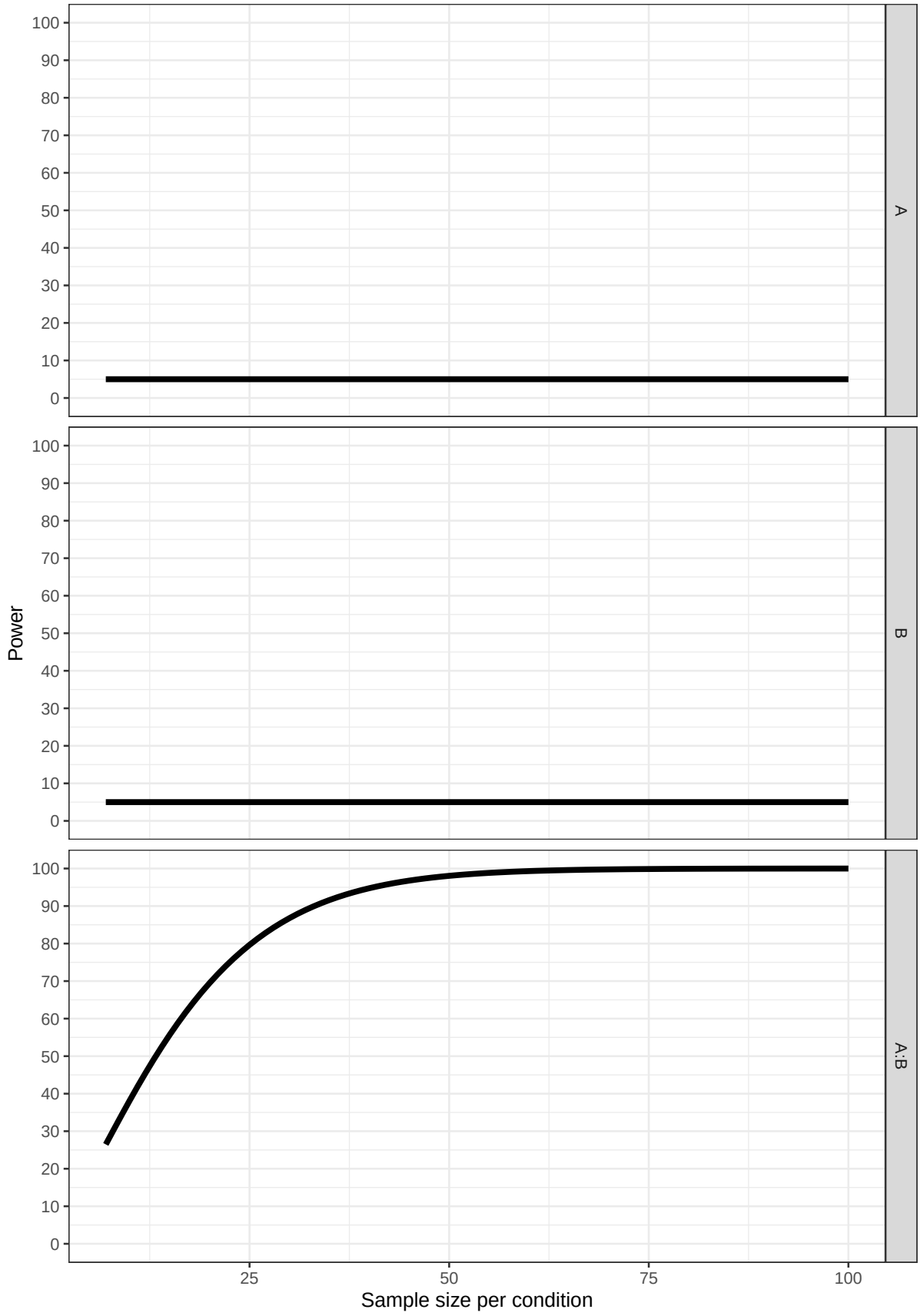
```
p_d <- plot_power(design_result,
                  max_n = 100)
```


Power
Sample size per condition
A
B
A:B
0
10
20
30
40
50
60
70
80
90
100
25
50
75
100

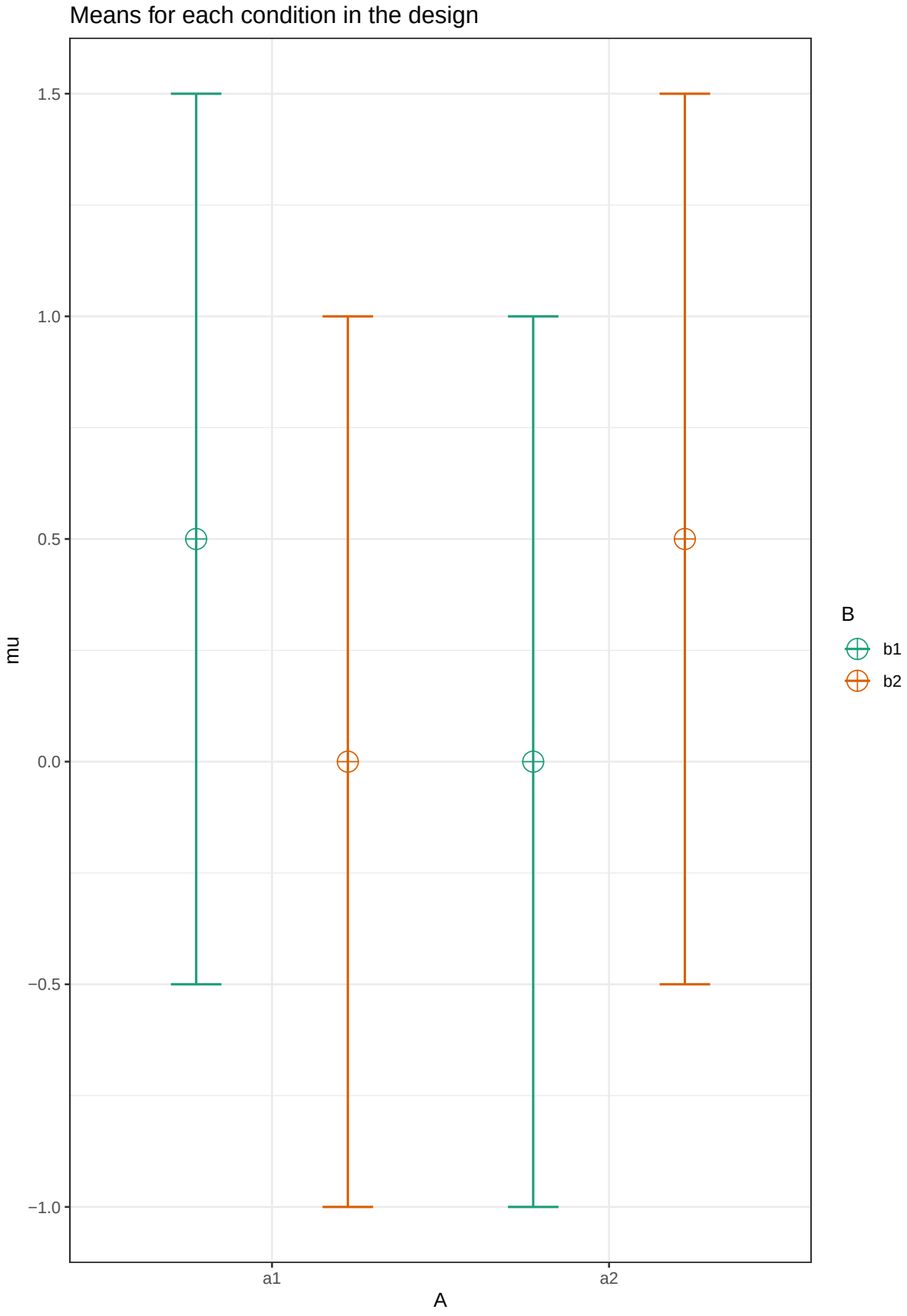
```
design_result <- ANOVA_design(design = string,
                              n = 20,
                              mu = c(0.4,0,0,0.4),
                              sd = 1,
                              r = 0.5,
                              labelnames = labelnames)
```


Means for each condition in the design
1.5
1.0
0.5
0.0
−0.5
−1.0
mu
a1
a2
A
B
b1
b2

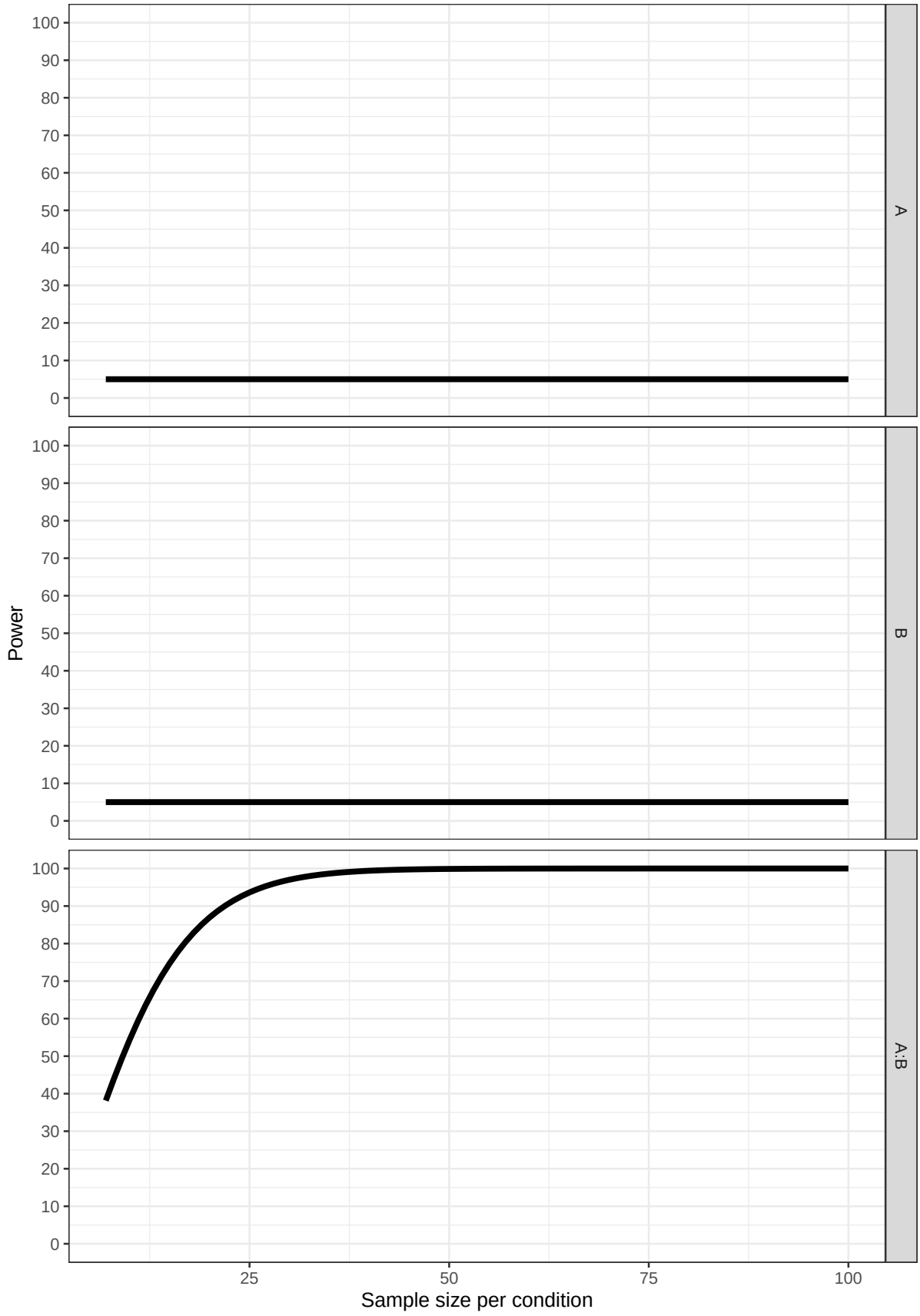
```
p_e <- plot_power(design_result,
                  max_n = 100)
```


Power
100
90
80
70
60
50
40
30
20
10
0
A
B
A:B
25
50
75
100
Sample size per condition

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


Means for each condition in the design
1.5
1.0
0.5
0.0
−0.5
−1.0
mu
a1
a2
A
B
b1
b2

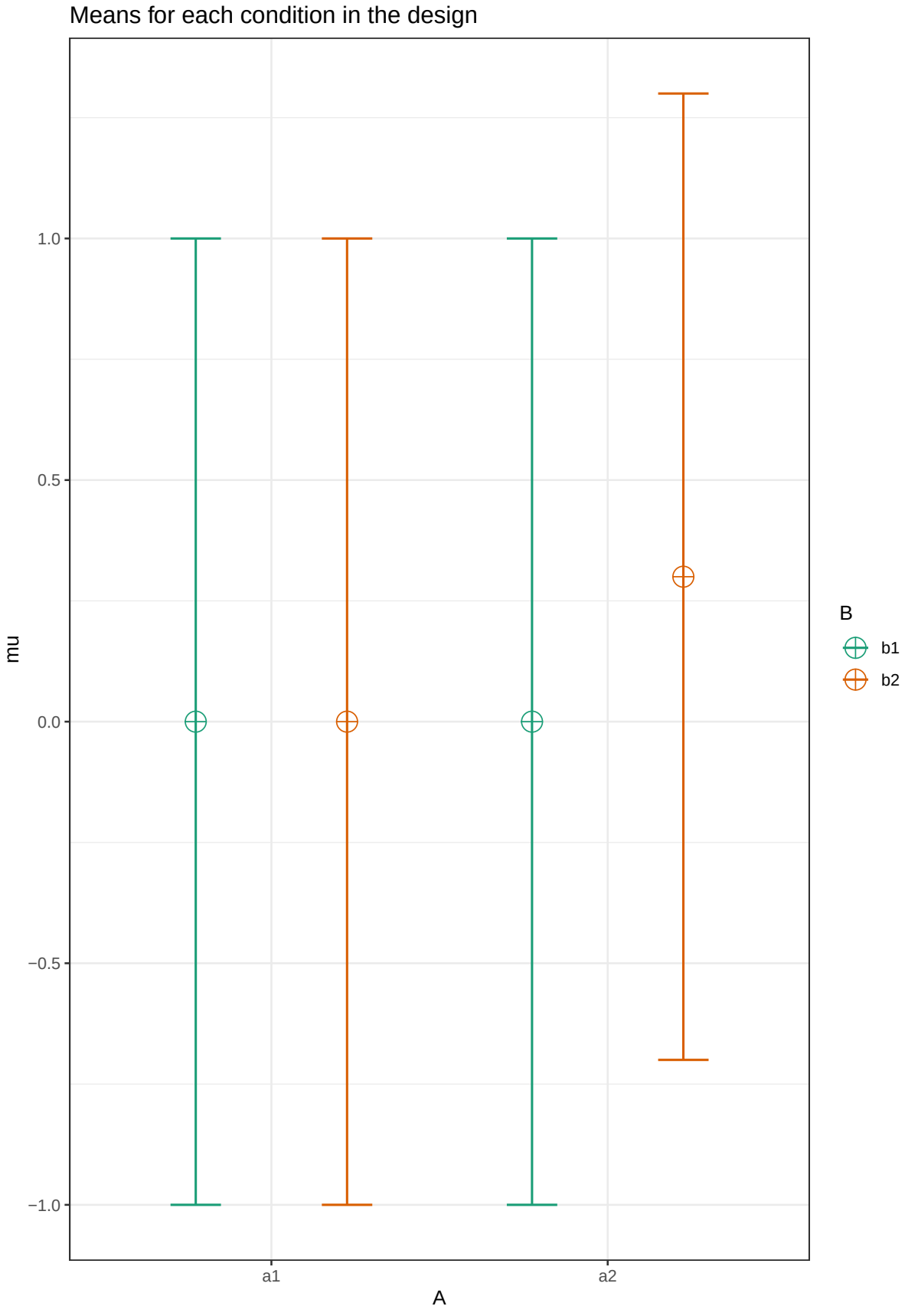
```
p_f <- plot_power(design_result,
                  max_n = 100)
```


Power
100
90
80
70
60
50
40
30
20
10
0
A
B
A:B
25
50
75
100
Sample size per condition

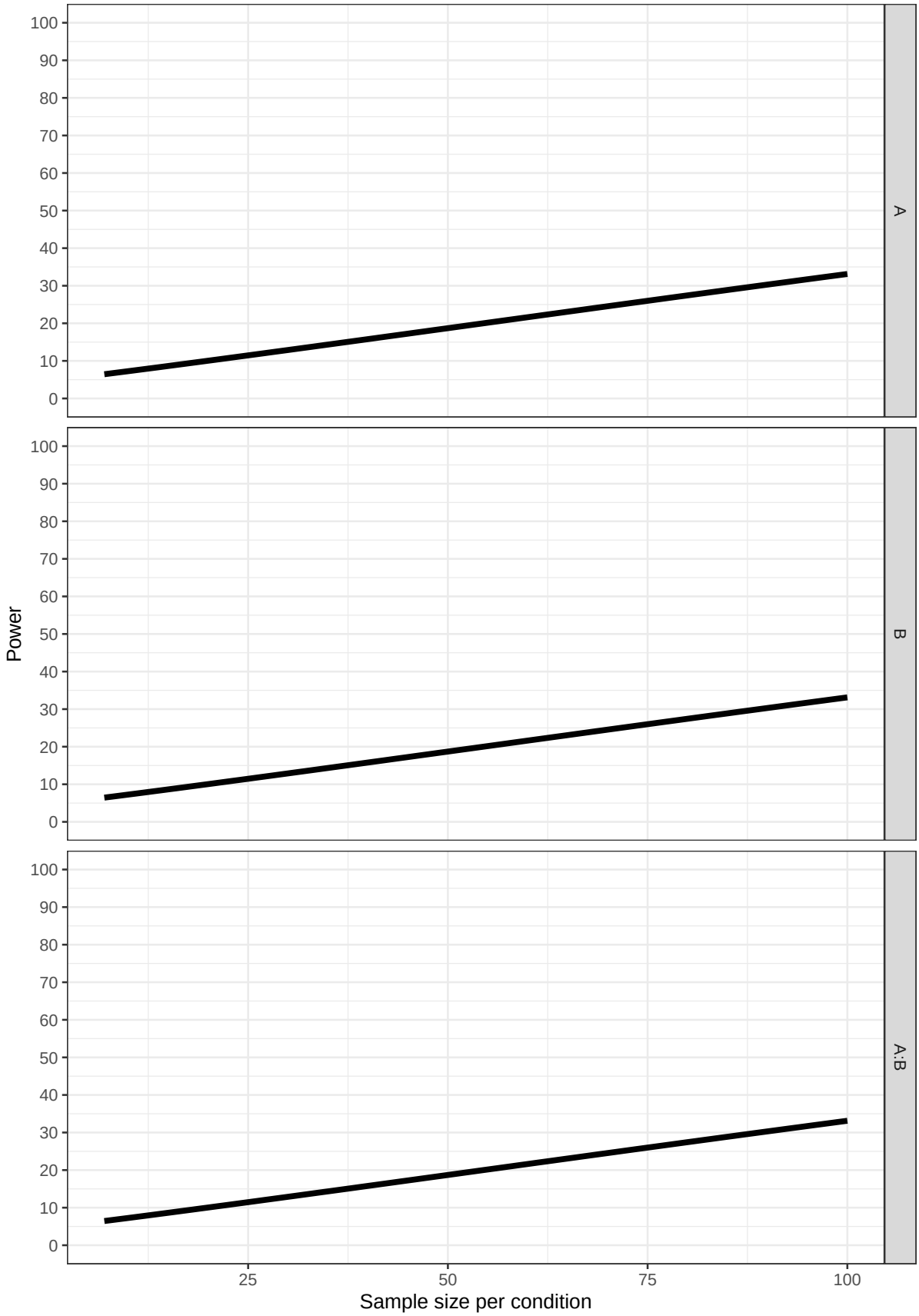
15.2 Explore increase in correlation in moderated interactions.

The design has means 0, 0, 0, 0.3. The standard deviation is set to 1. The correlation between all variables increases from 0.5 to 0.9.

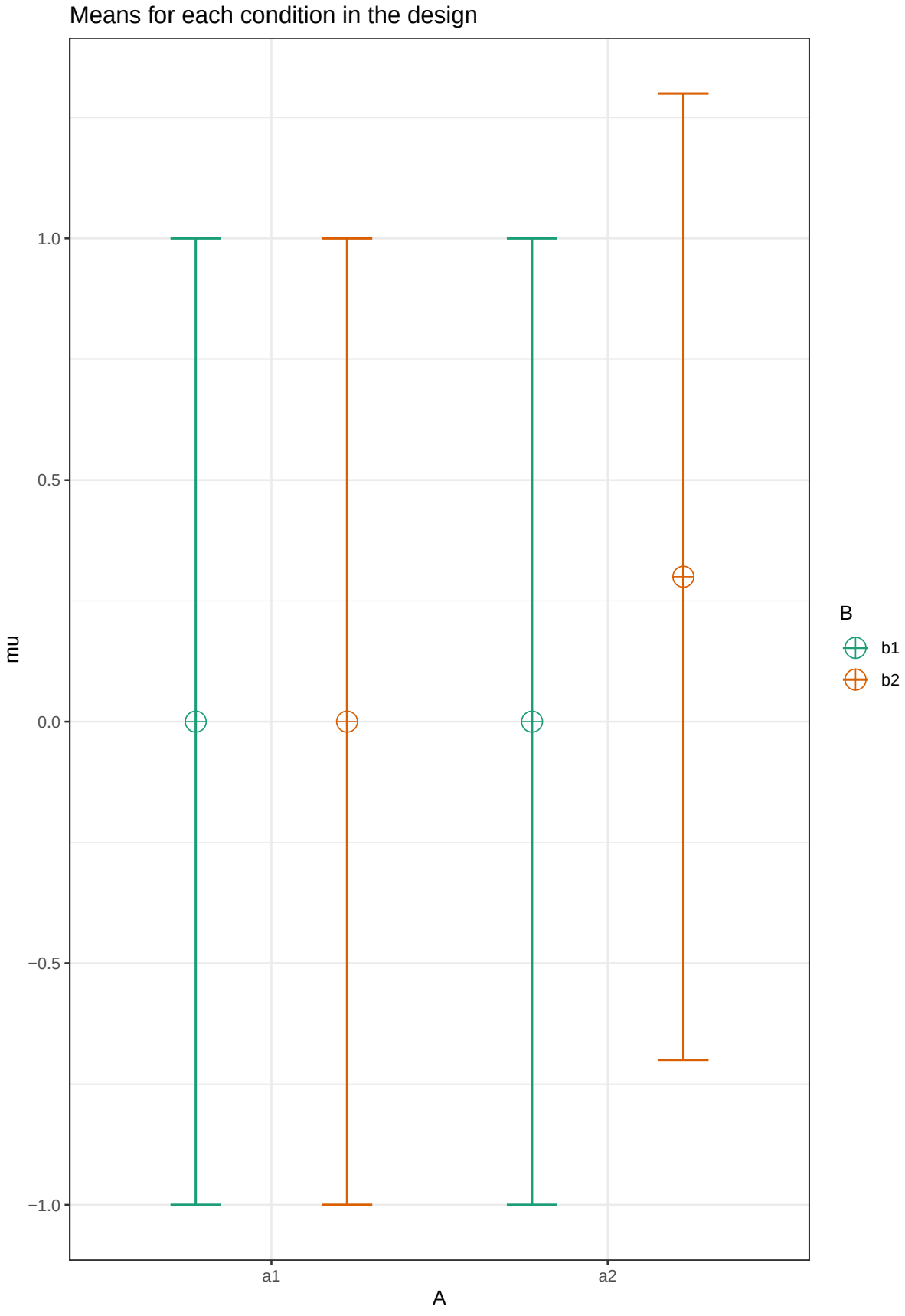
```
string <- "2w*2w"
labelnames = c("A", "a1", "a2", "B", "b1", "b2")
design_result <- ANOVA_design(design = string,
                              n = 20,
                              mu = c(0,0,0,0.3),
                              sd = 1,
                              r = 0.0,
                              labelnames = labelnames)
```


Means for each condition in the design
mu
1.0
0.5
0.0
−0.5
−1.0
a1
a2
A
B
b1
b2

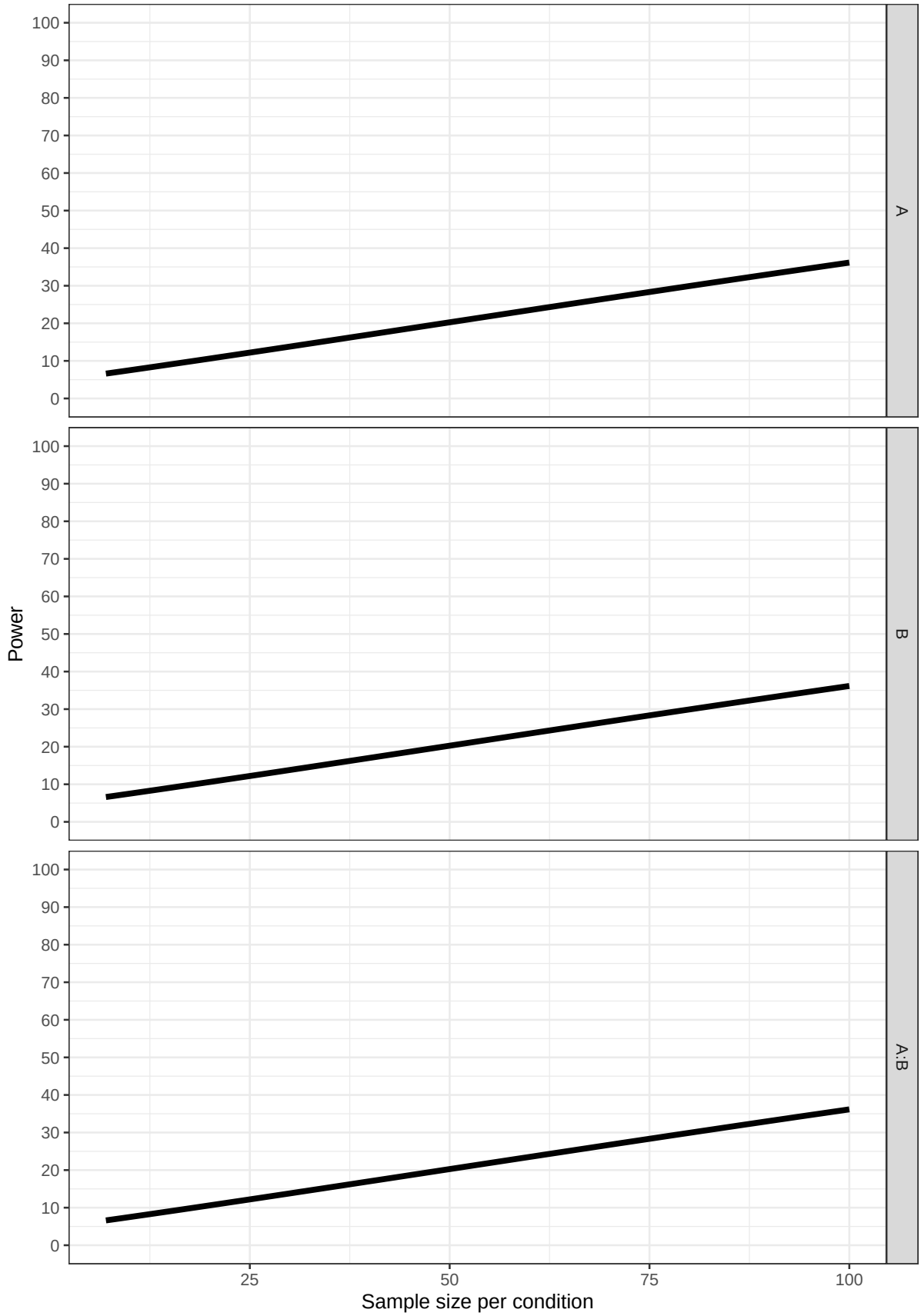
```
p_a <- plot_power(design_result,
                  max_n = 100)
```


Power
100
90
80
70
60
50
40
30
20
10
0
A
B
A:B
25
50
75
100
Sample size per condition

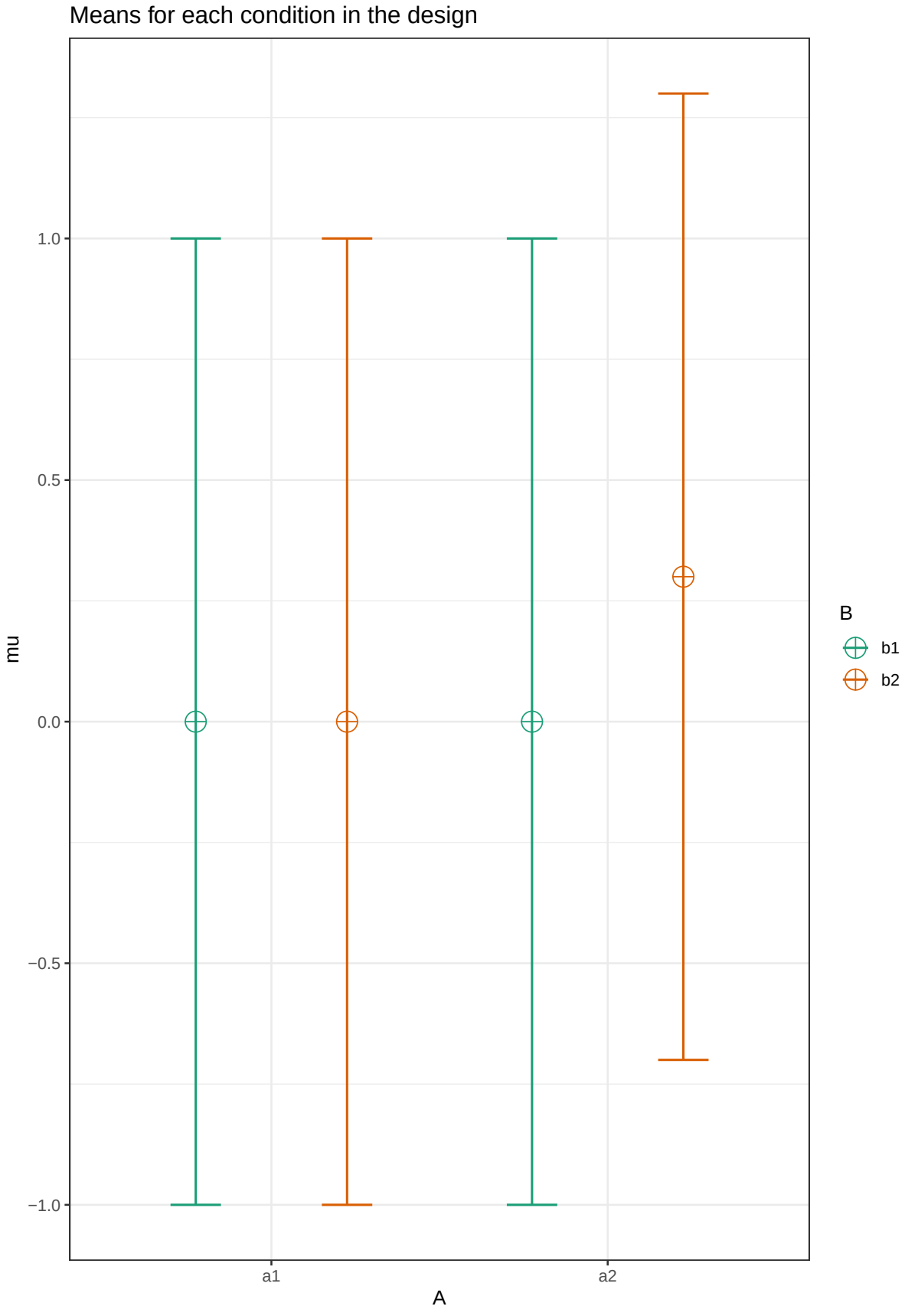
```
design_result <- ANOVA_design(design = string,
                              n = 20,
                              mu = c(0,0,0,0.3),
                              sd = 1,
                              r = 0.1,
                              labelnames = labelnames)
```


Means for each condition in the design
mu
1.0
0.5
0.0
−0.5
−1.0
a1
a2
A
B
b1
b2

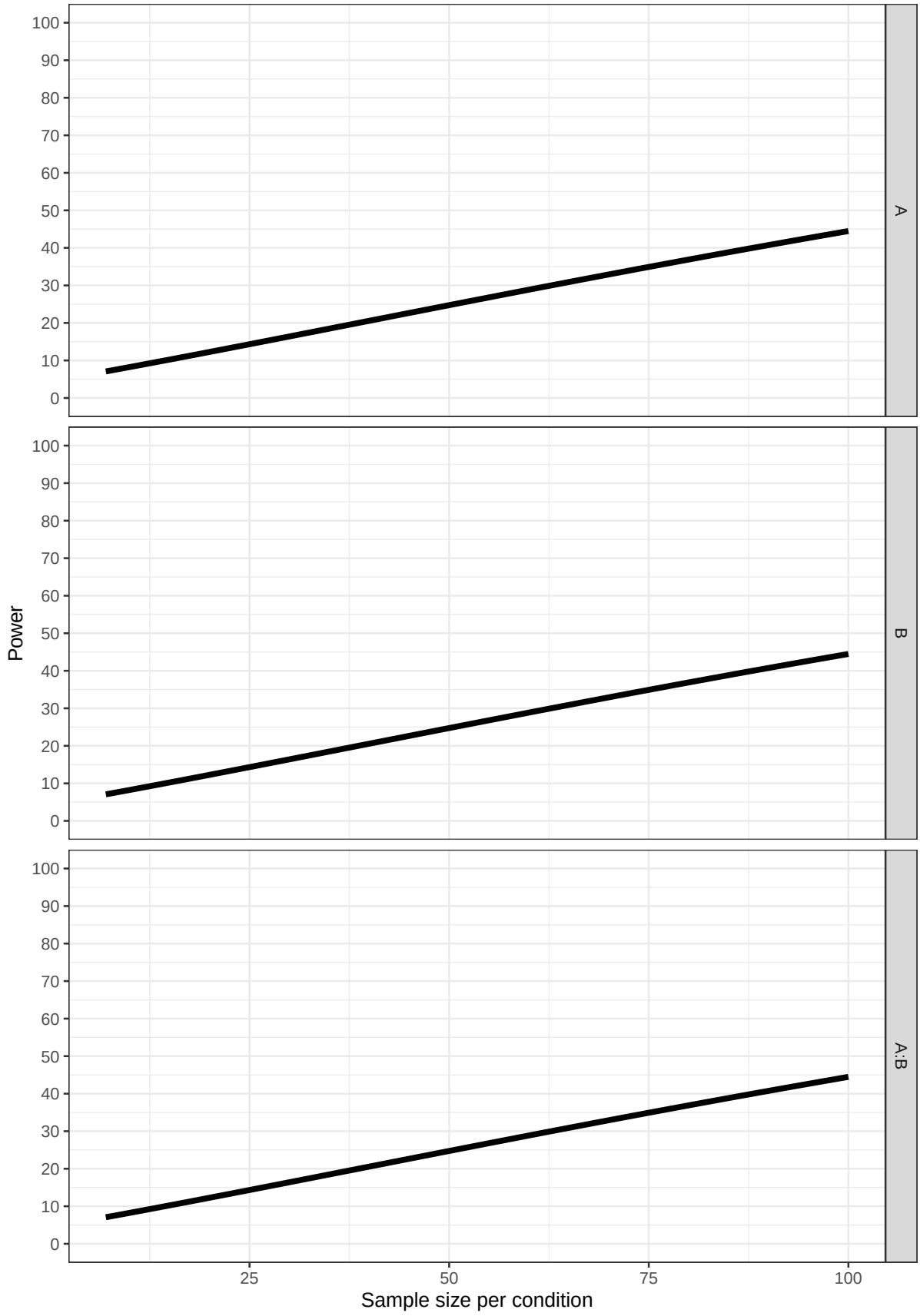
```
p_b <- plot_power(design_result,
                  max_n = 100)
```


Power
100
90
80
70
60
50
40
30
20
10
0
A
B
A:B
25
50
75
100
Sample size per condition

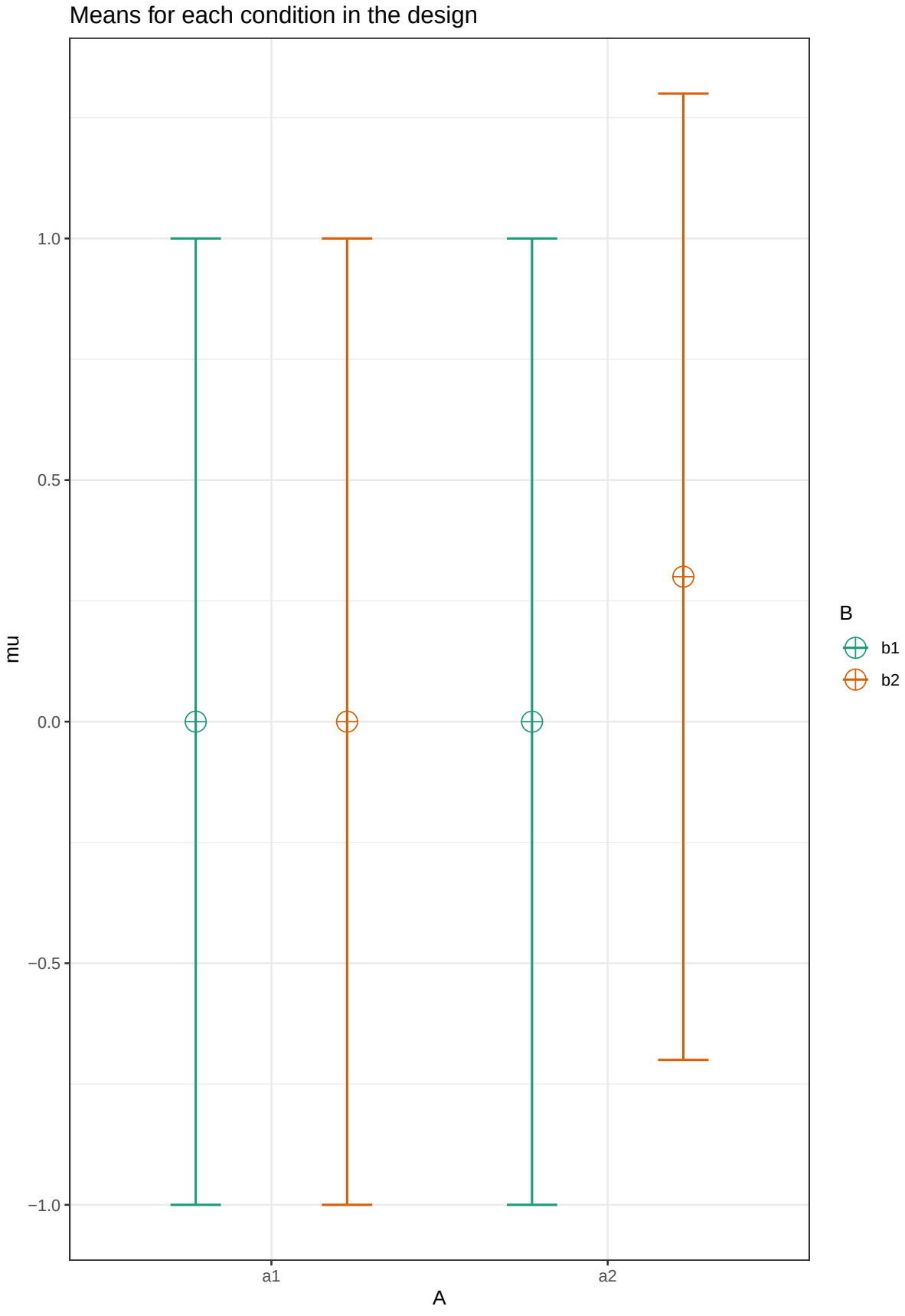
```
design_result <- ANOVA_design(design = string,
                              n = 20,
                              mu = c(0,0,0,0.3),
                              sd = 1,
                              r = 0.3,
                              labelnames = labelnames)
```


Means for each condition in the design
mu
1.0
0.5
0.0
−0.5
−1.0
a1
a2
A
B
b1
b2

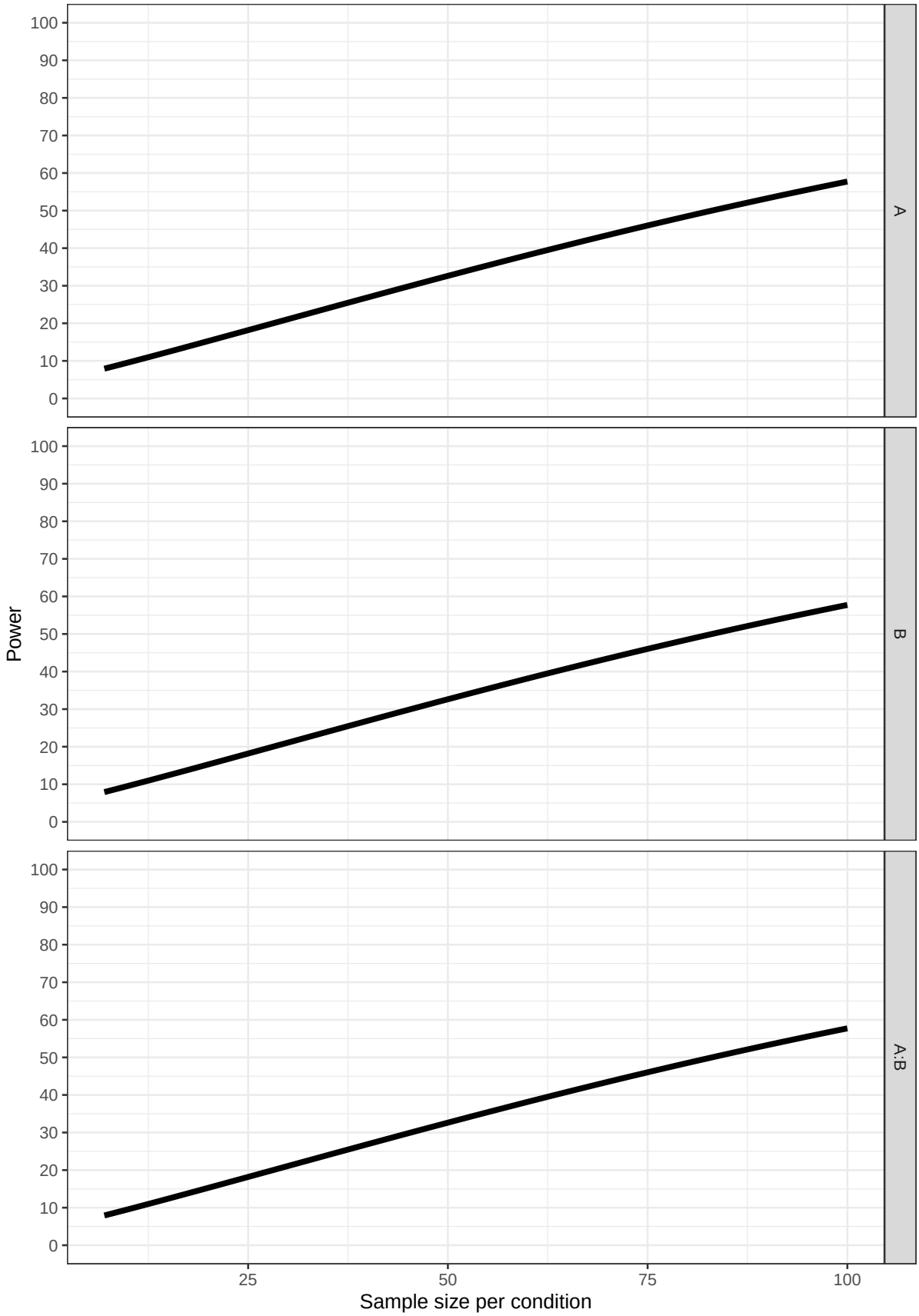
```
p_c <- plot_power(design_result,
                        max_n = 100)
```


Power
100
90
80
70
60
50
40
30
20
10
0
A
B
A:B
25
50
75
100
Sample size per condition

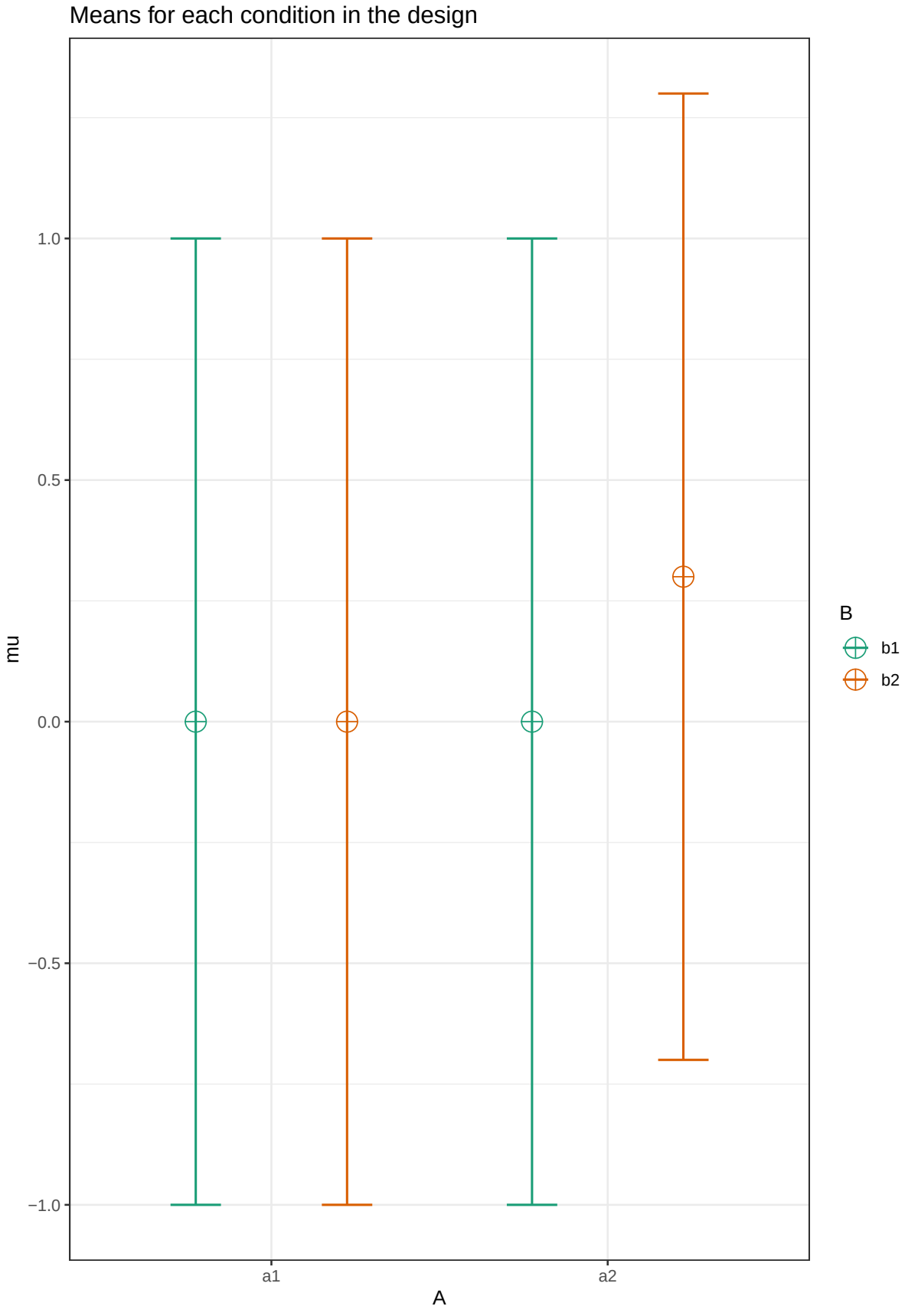
```
design_result <- ANOVA_design(design = string,
                              n = 20,
                              mu = c(0,0,0,0.3),
                              sd = 1,
                              r = 0.5,
                              labelnames = labelnames)
```


Means for each condition in the design
mu
1.0
0.5
0.0
−0.5
−1.0
a1
a2
A
B
b1
b2

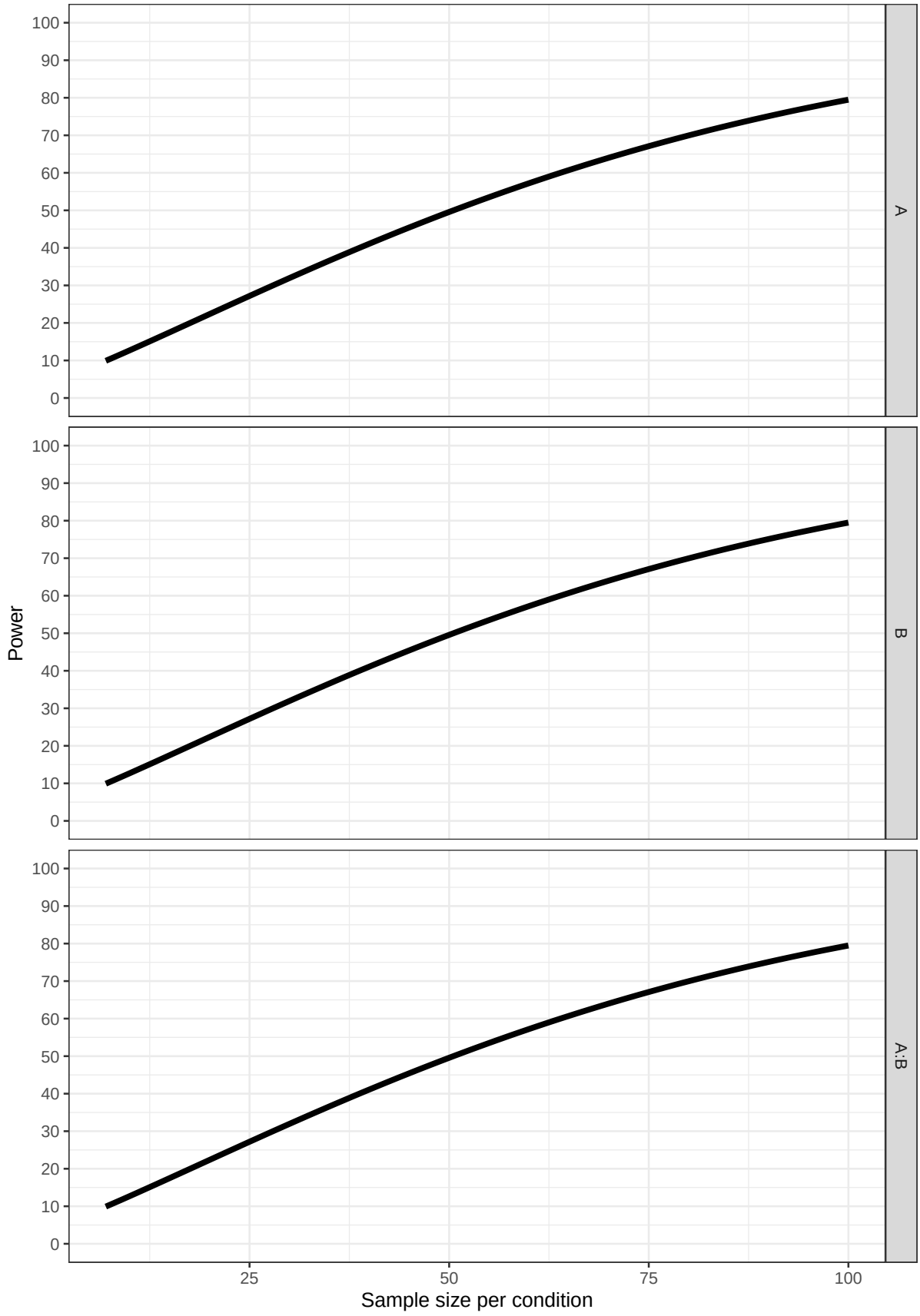
```
p_d <- plot_power(design_result,
                        max_n = 100)
```


Power
100
90
80
70
60
50
40
30
20
10
0
A
B
A:B
25
50
75
100
Sample size per condition

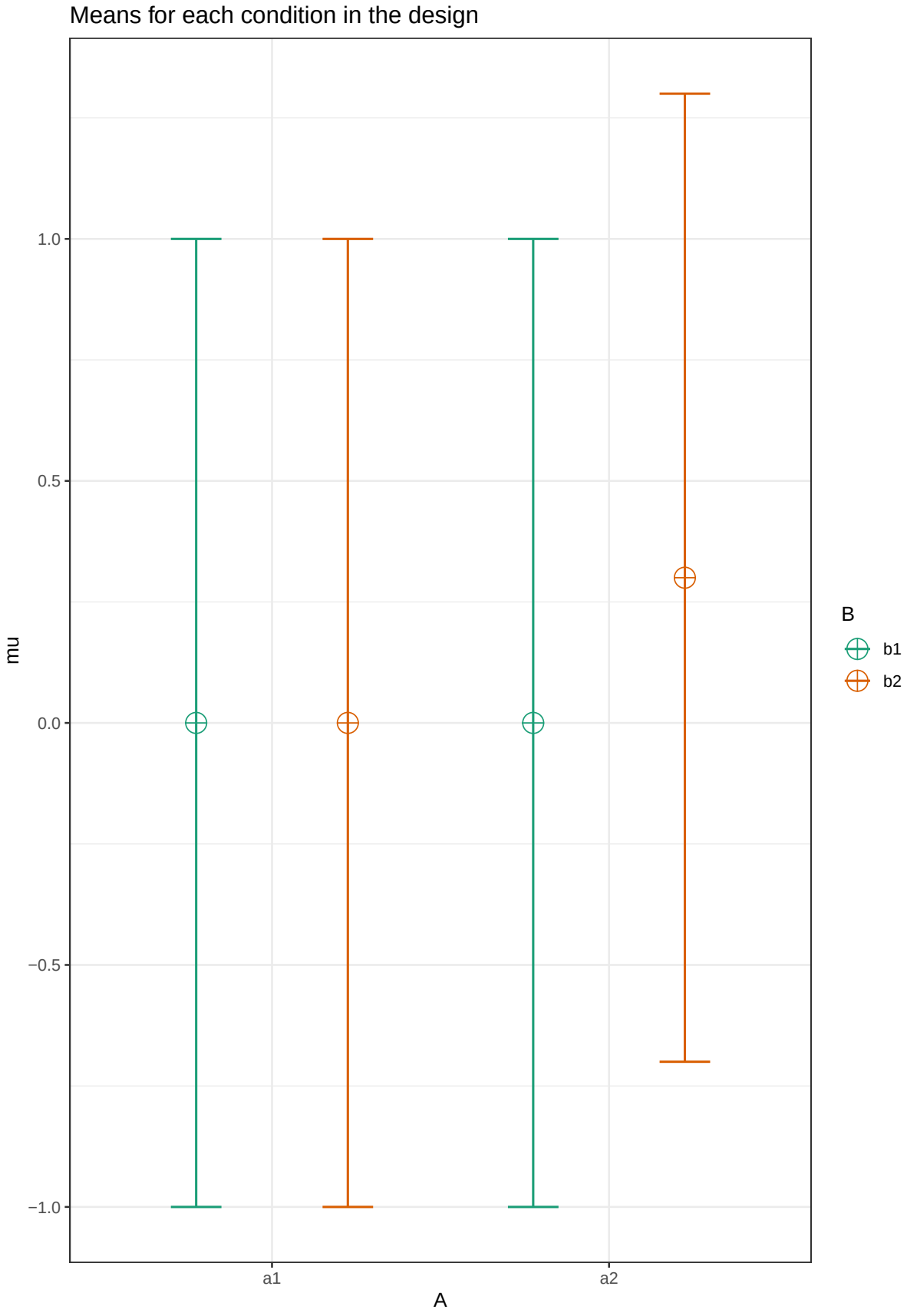
```
design_result <- ANOVA_design(design = string,
                              n = 20,
                              mu = c(0,0,0,0.3),
                              sd = 1,
                              r = 0.7,
                              labelnames = labelnames)
```


Means for each condition in the design
mu
1.0
0.5
0.0
−0.5
−1.0
a1
a2
A
B
b1
b2

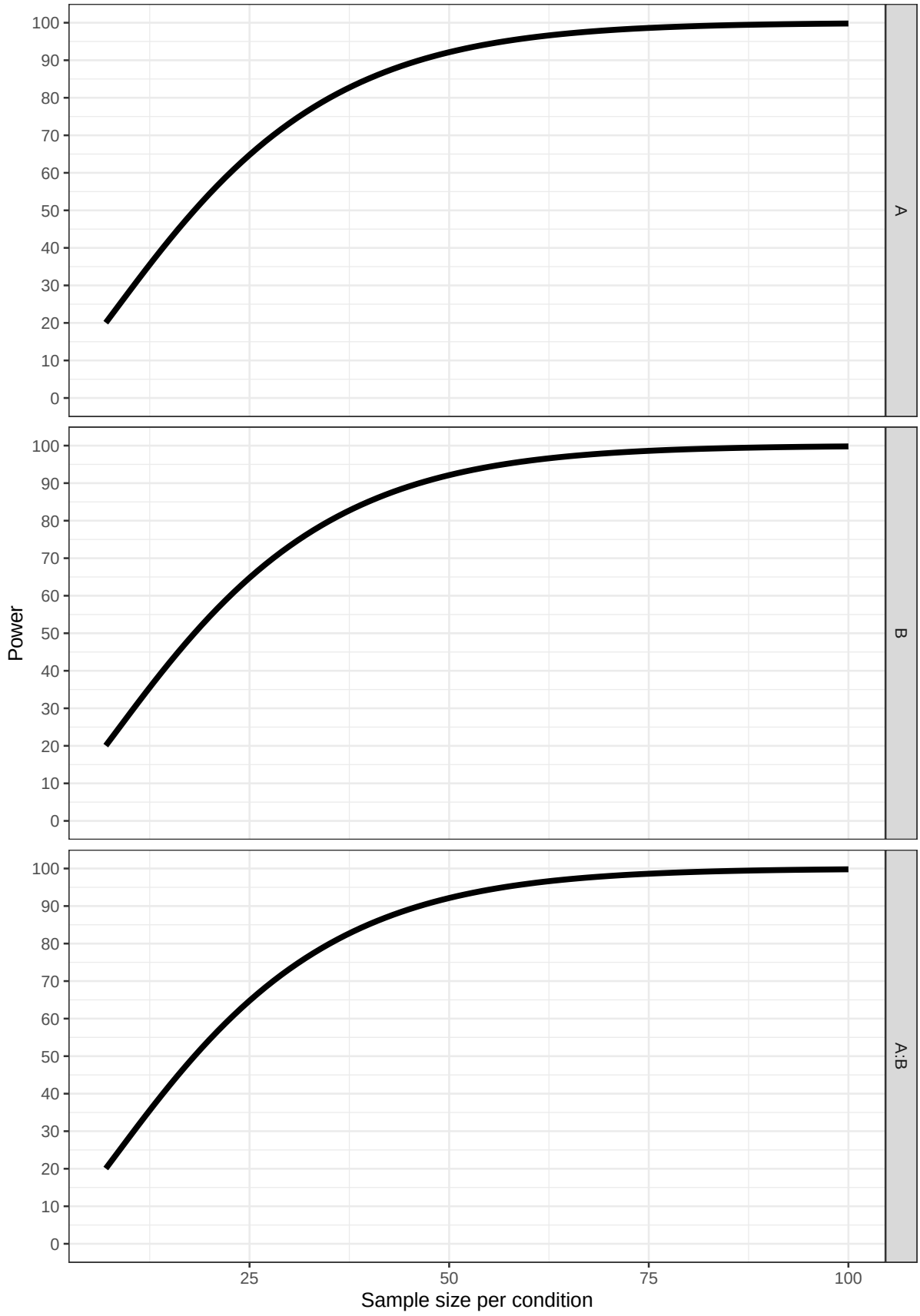
```
p_e <- plot_power(design_result,
                        max_n = 100)
```


Power
Sample size per condition
A
B
A:B
0
10
20
30
40
50
60
70
80
90
100
25
50
75
100

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


Means for each condition in the design
mu
1.0
0.5
0.0
−0.5
−1.0
a1
a2
A
B
b1
b2

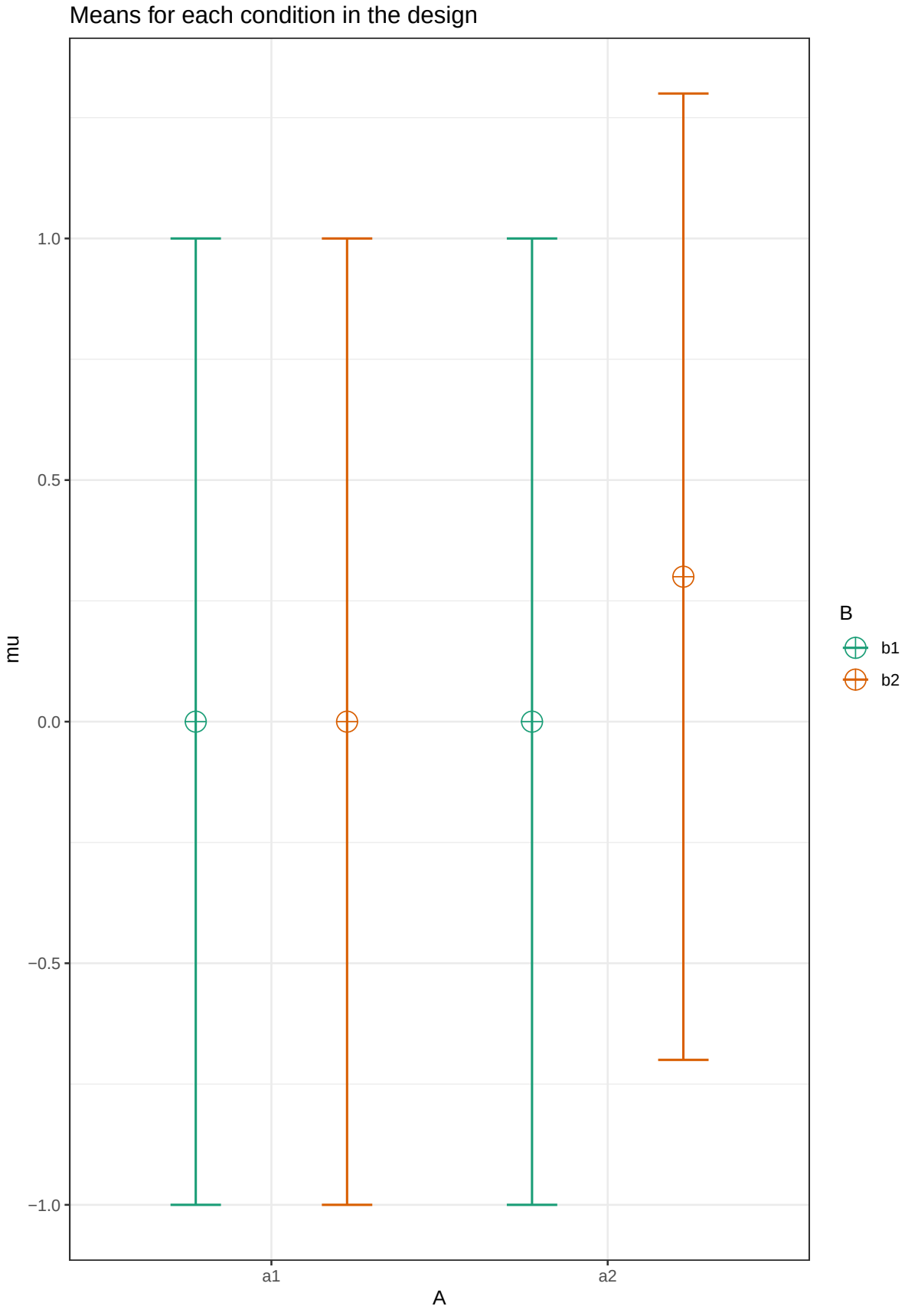
```
p_f <- plot_power(design_result,
                  max_n = 100)
```


Power
100
90
80
70
60
50
40
30
20
10
0
A
B
A:B
25
50
75
100
Sample size per condition

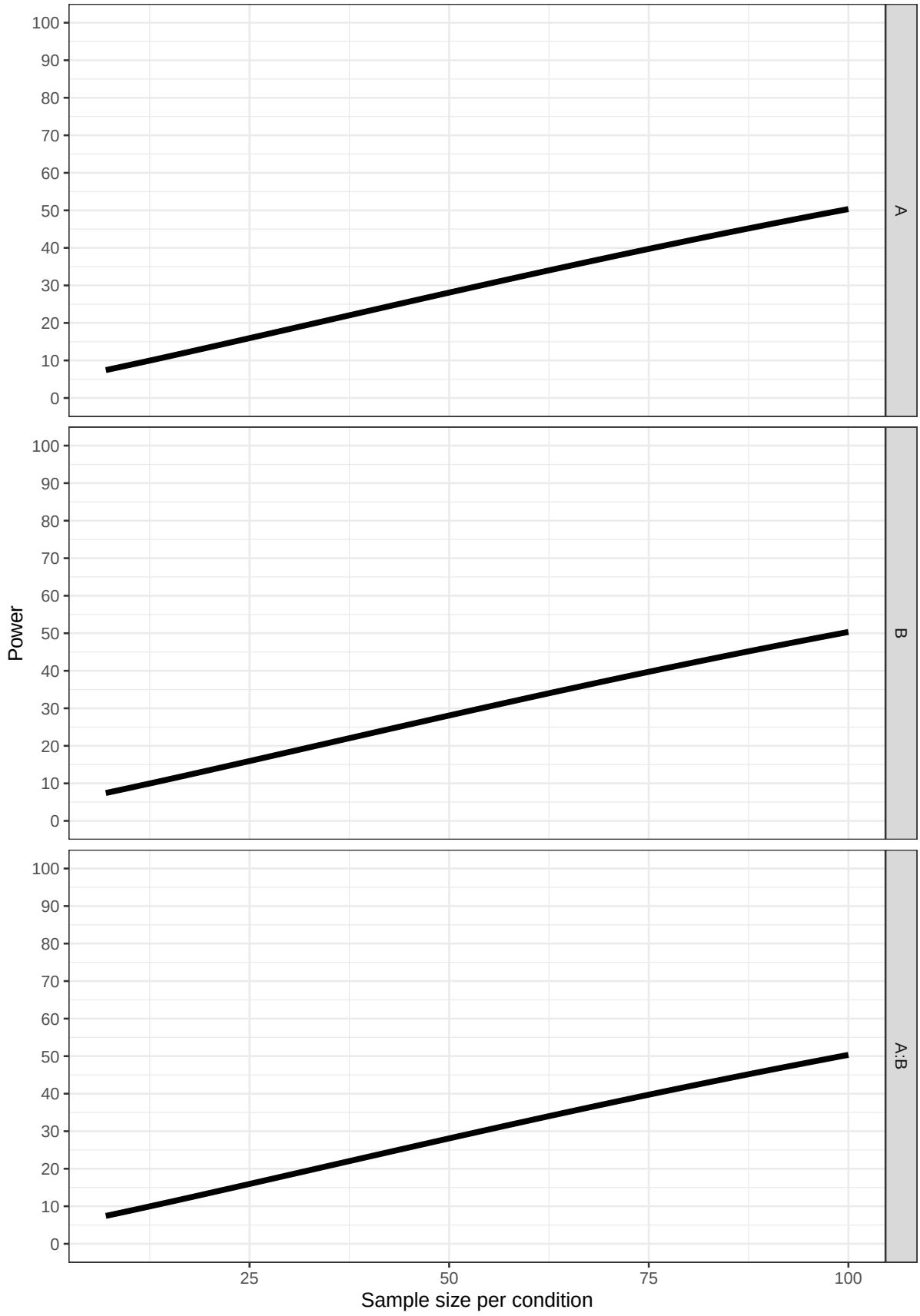
15.3 Increasing correlation in on factor decreases power in second factor

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., rB - rAB > 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 of factor B decreases.

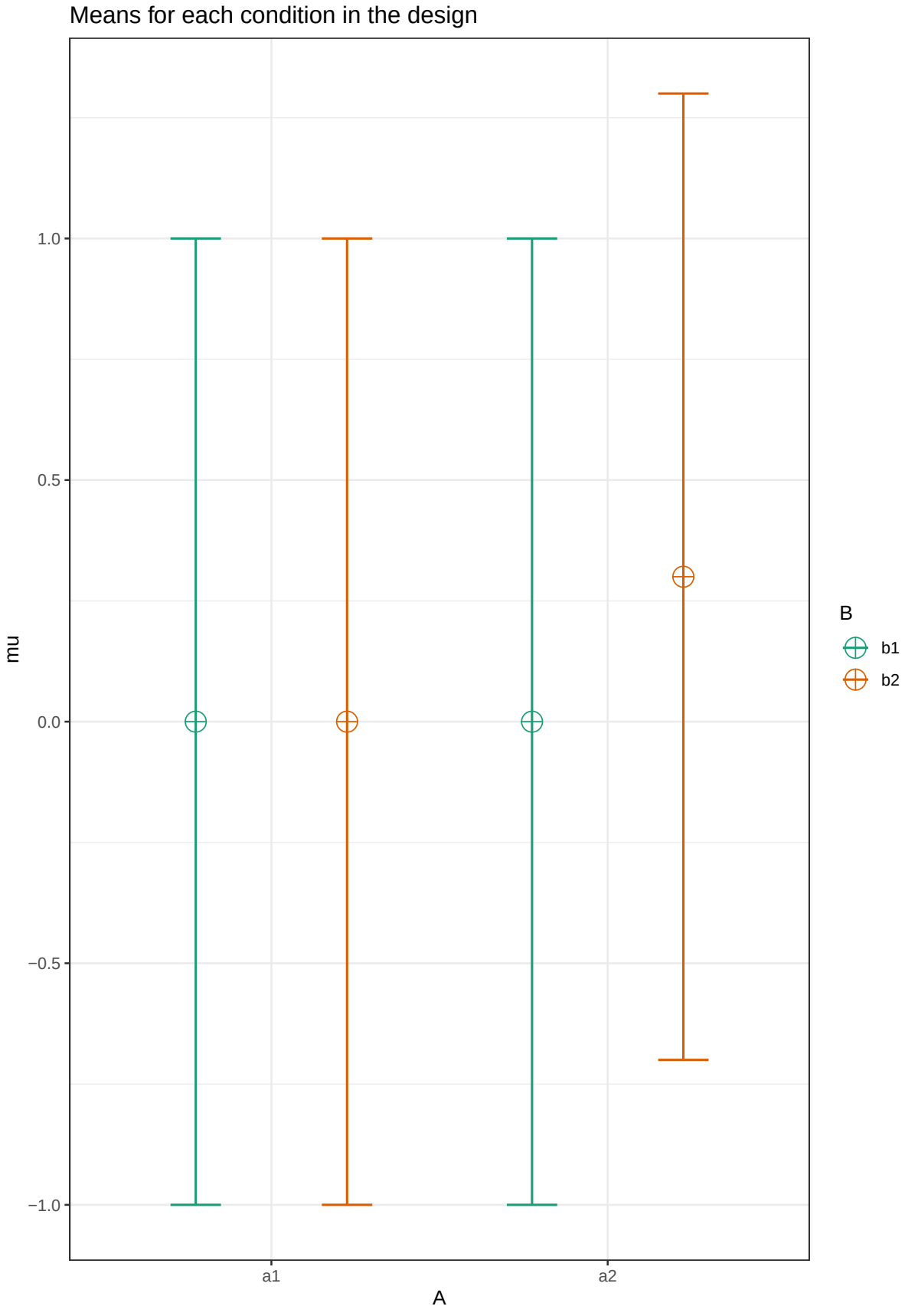
```
string <- "2w*2w"
labelnames = c("A", "a1", "a2", "B", "b1", "b2")
design_result <- ANOVA_design(design = string,
                              n = 20,
                              mu = c(0,0,0,0.3),
                              sd = 1,
                              r <- c(
                                0.4, 0.4, 0.4,
                                0.4, 0.4,
                                0.4),
                              labelnames = labelnames)
```


Means for each condition in the design
mu
1.0
0.5
0.0
−0.5
−1.0
a1
a2
A
B
b1
b2

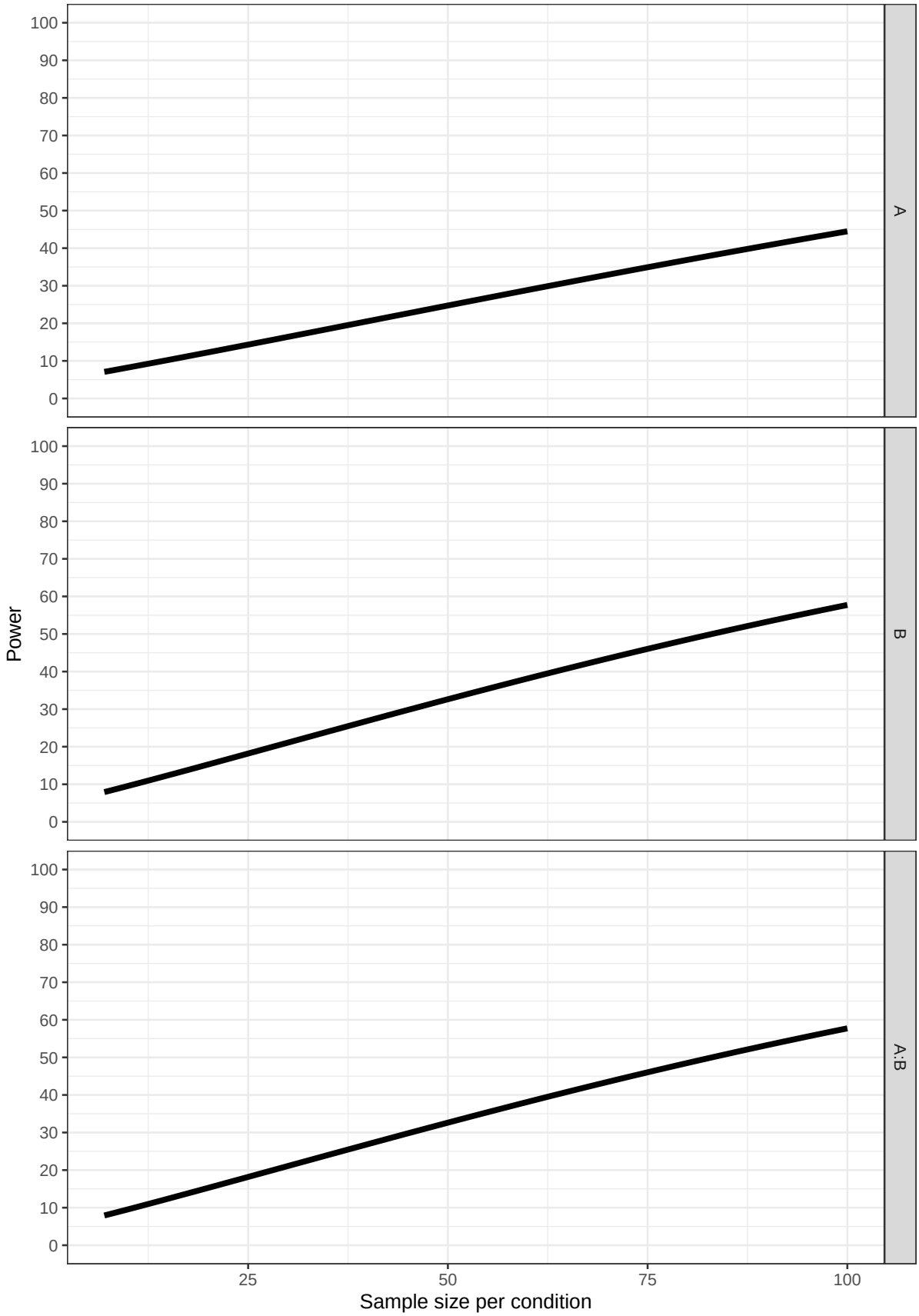
```
p_a <- plot_power(design_result,
                      max_n = 100)
```


Power
100
90
80
70
60
50
40
30
20
10
0
A
B
A:B
25
50
75
100
Sample size per condition

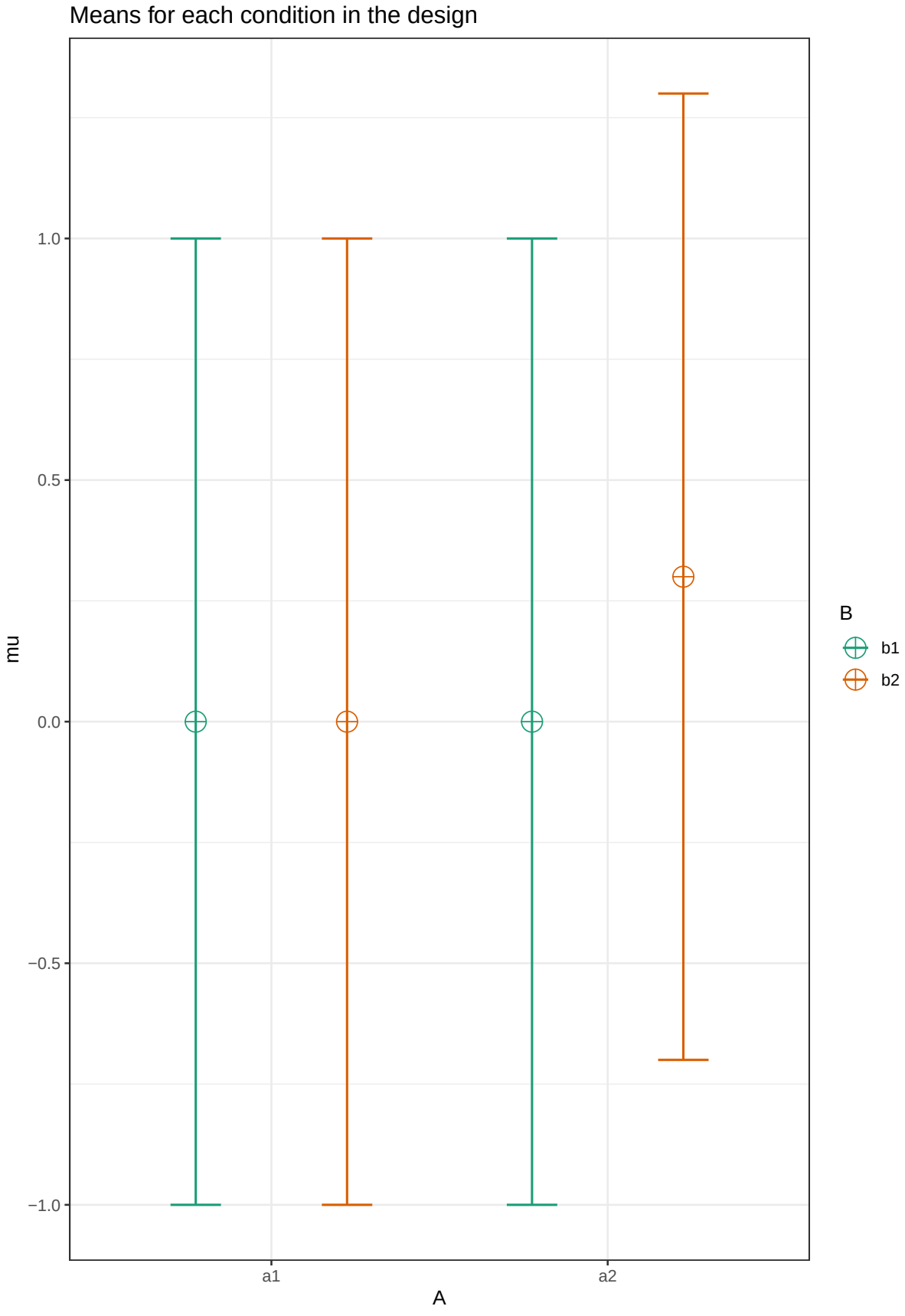
```
design_result <- ANOVA_design(design = string,
                              n = 20,
                              mu = c(0,0,0,0.3),
                              sd = 1,
                              r <- c(
                                0.5, 0.4, 0.4,
                                0.4, 0.4,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


Means for each condition in the design
mu
1.0
0.5
0.0
−0.5
−1.0
a1
a2
A
B
b1
b2

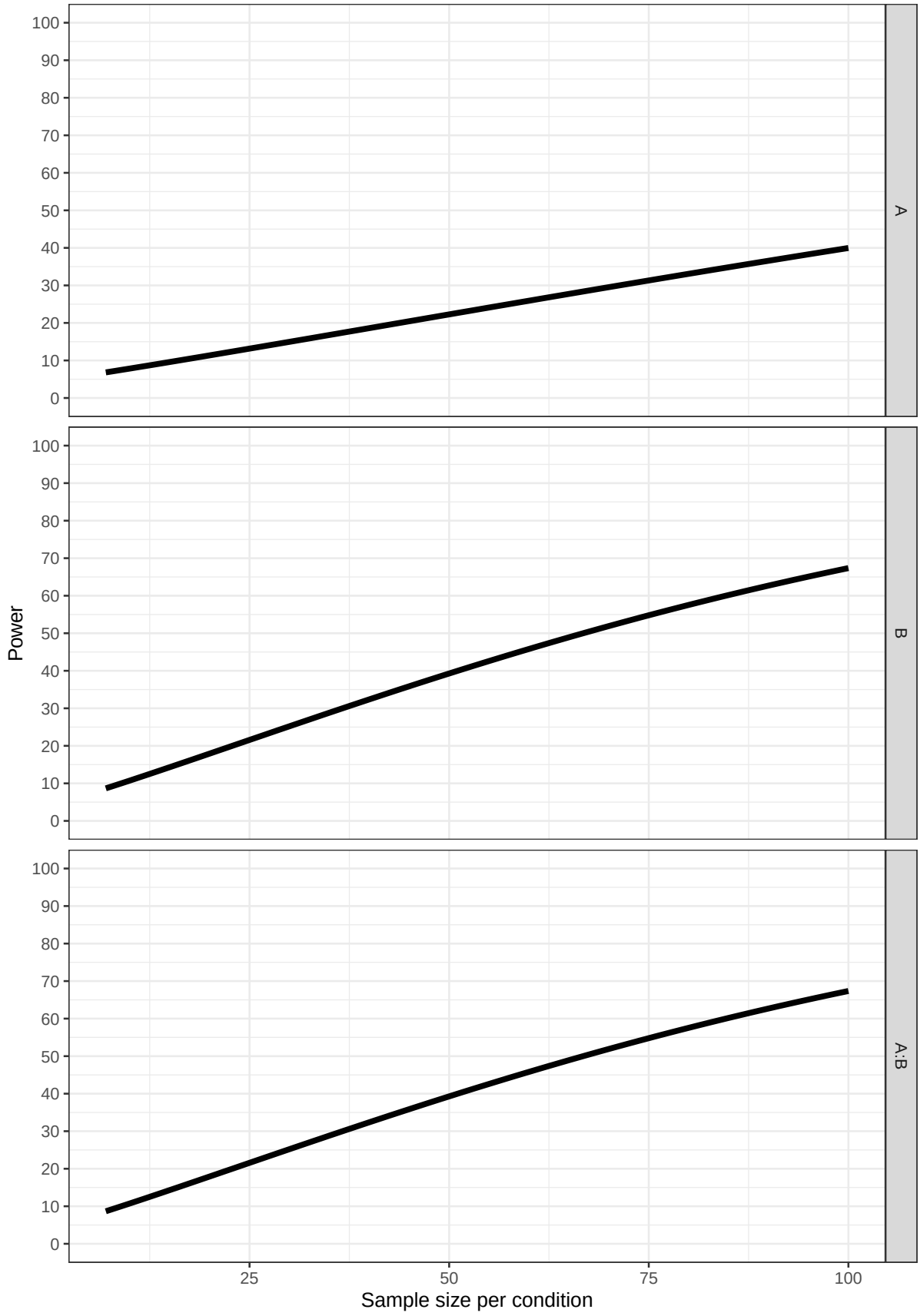
```
p_b <- plot_power(design_result,
                      max_n = 100)
```


Power
100
90
80
70
60
50
40
30
20
10
0
A
B
A:B
25
50
75
100
Sample size per condition

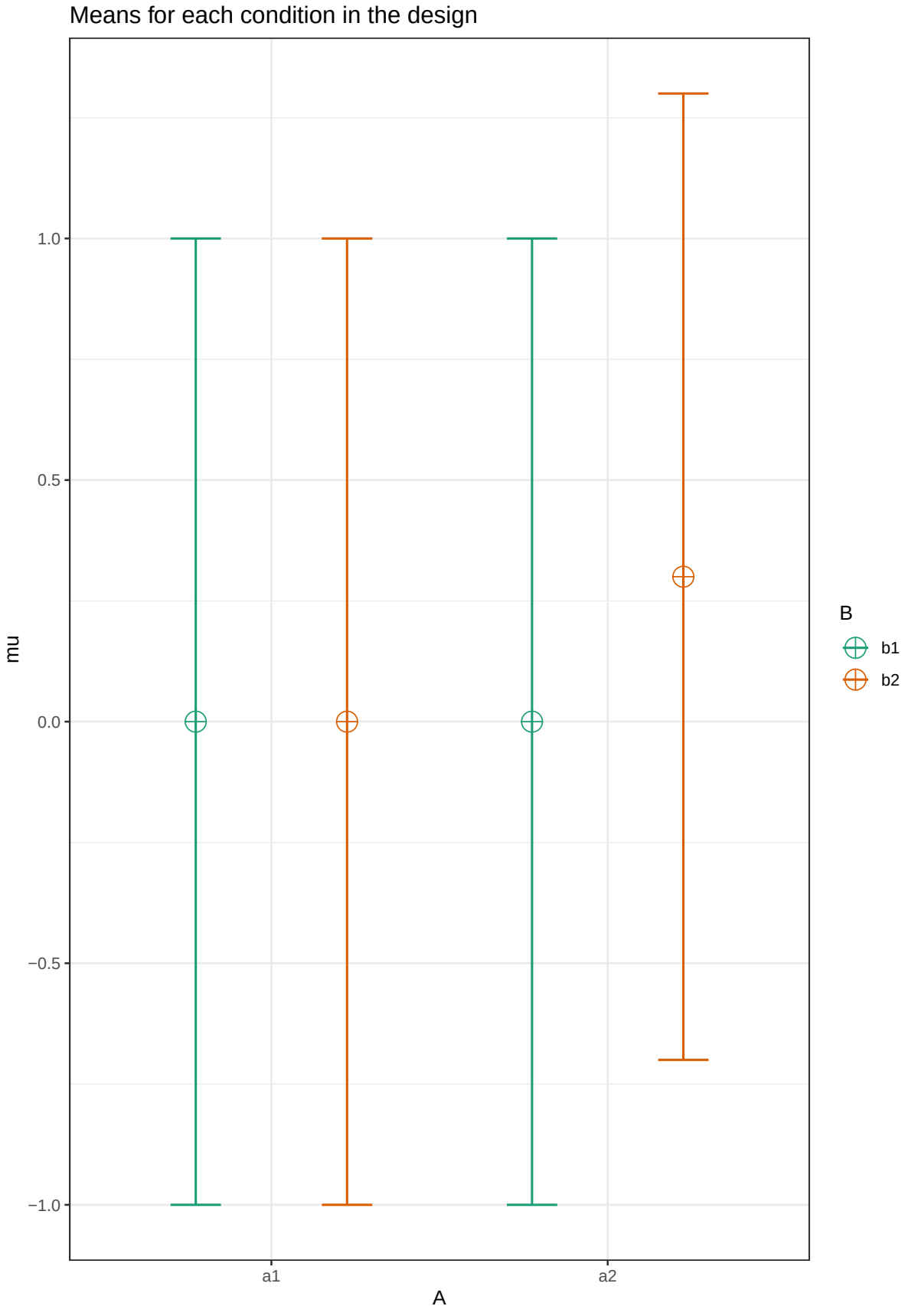
```
design_result <- ANOVA_design(design = string,
                              n = 20,
                              mu = c(0,0,0,0.3),
                              sd = 1,
                              r <- c(
                                0.6, 0.4, 0.4,
                                0.4, 0.4,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


Means for each condition in the design
mu
1.0
0.5
0.0
−0.5
−1.0
a1
a2
A
B
b1
b2

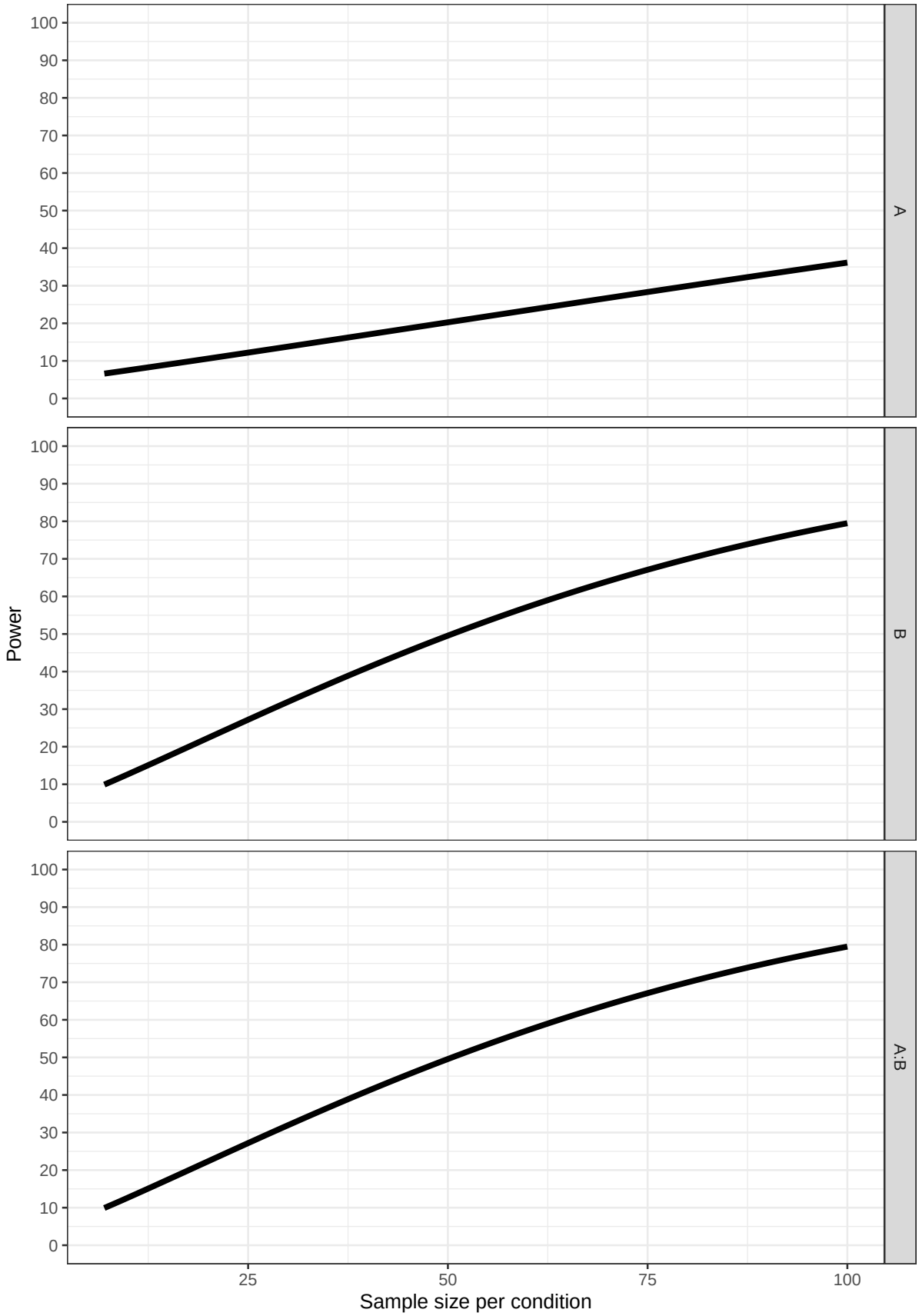
```
p_c <- plot_power(design_result,
                      max_n = 100)
```


Power
100
90
80
70
60
50
40
30
20
10
0
A
B
A:B
25
50
75
100
Sample size per condition

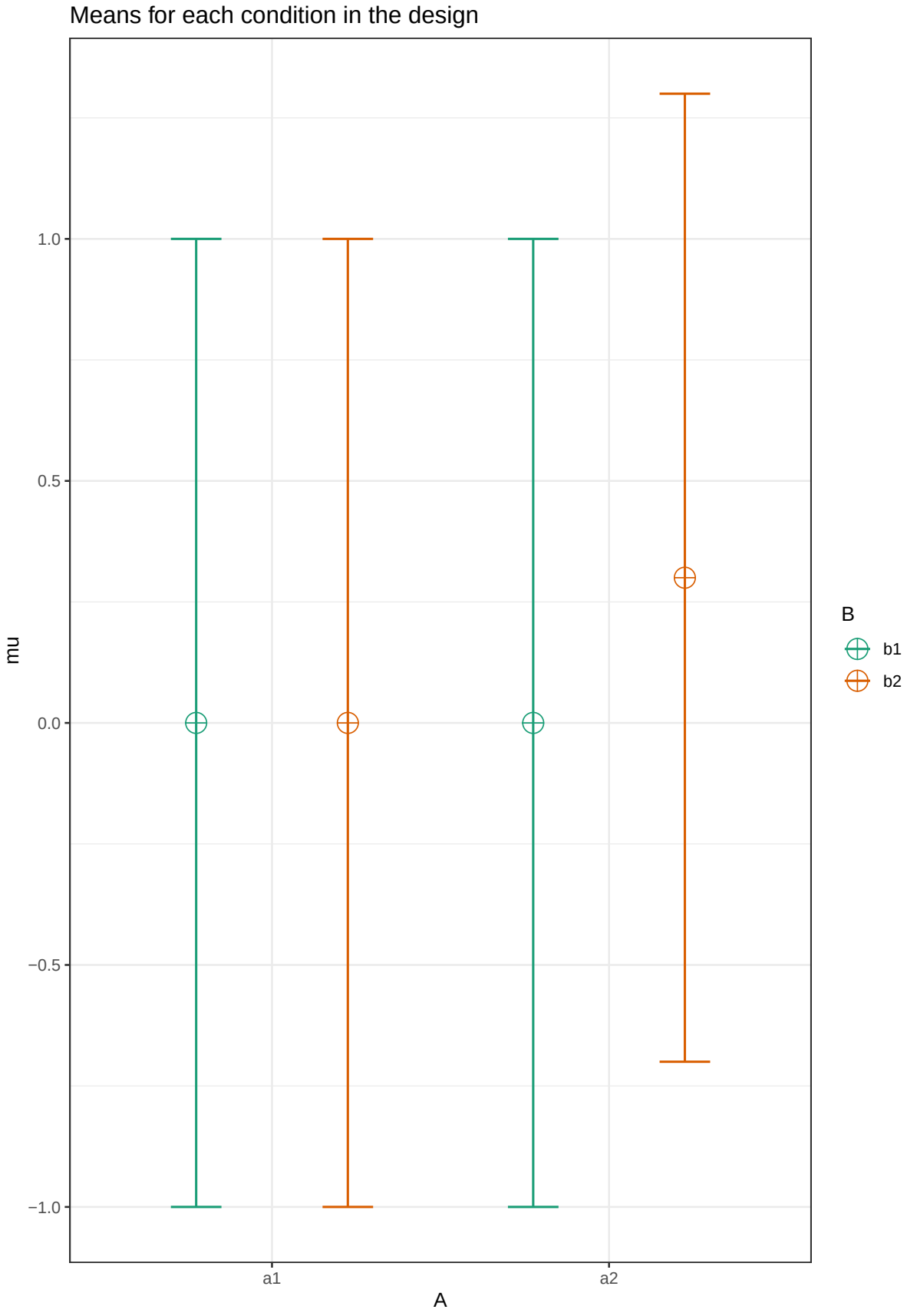
```
design_result <- ANOVA_design(design = string,
                              n = 20,
                              mu = c(0,0,0,0.3),
                              sd = 1,
                              r <- c(
                                0.7, 0.4, 0.4,
                                0.4, 0.4,
                                0.7),
                              labelnames = labelnames)
```


Means for each condition in the design
mu
1.0
0.5
0.0
−0.5
−1.0
a1
a2
A
B
b1
b2

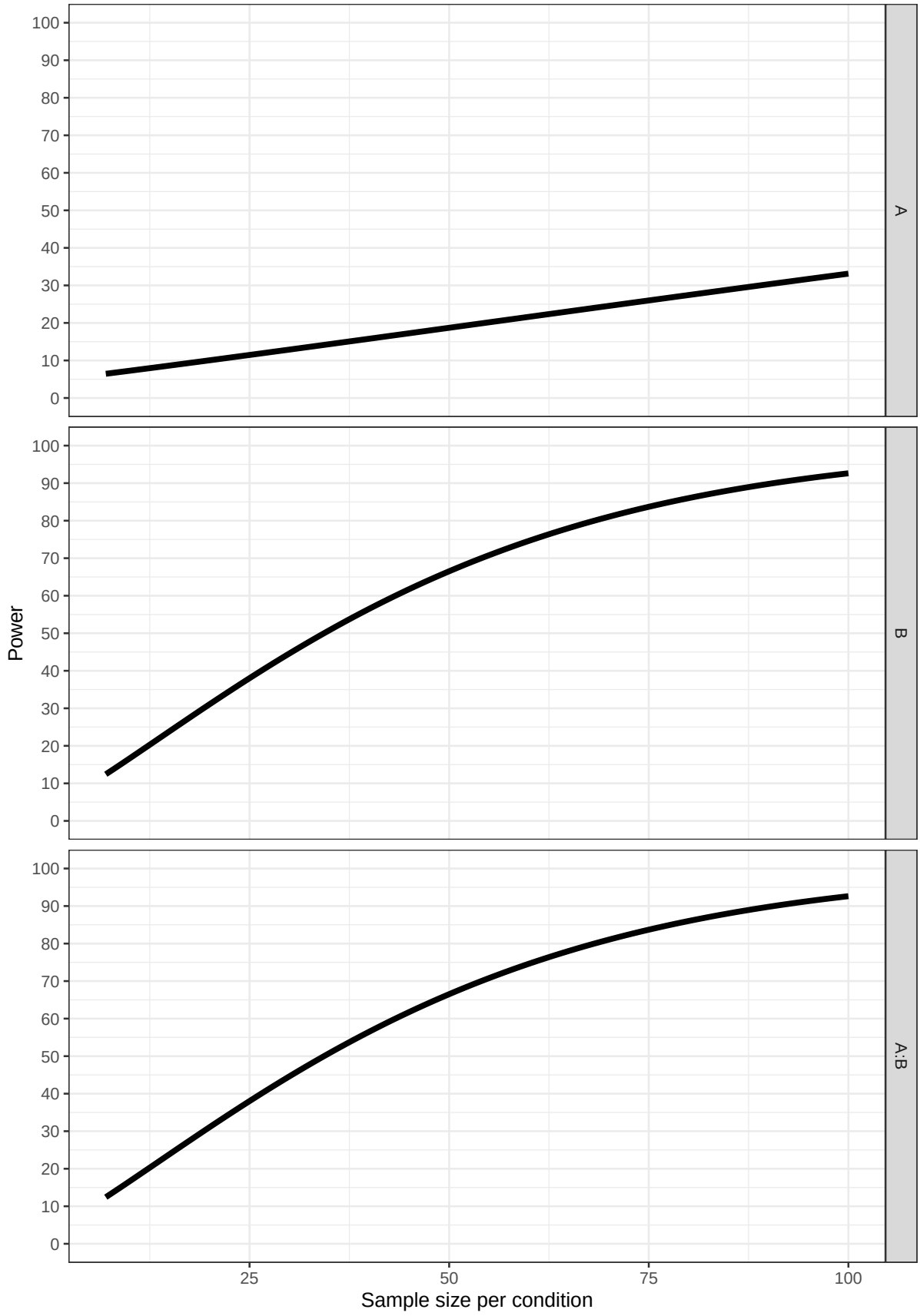
```
p_d <- plot_power(design_result,
                        max_n = 100)
```


Power
Sample size per condition
A
B
A:B
0
10
20
30
40
50
60
70
80
90
100
25
50
75
100

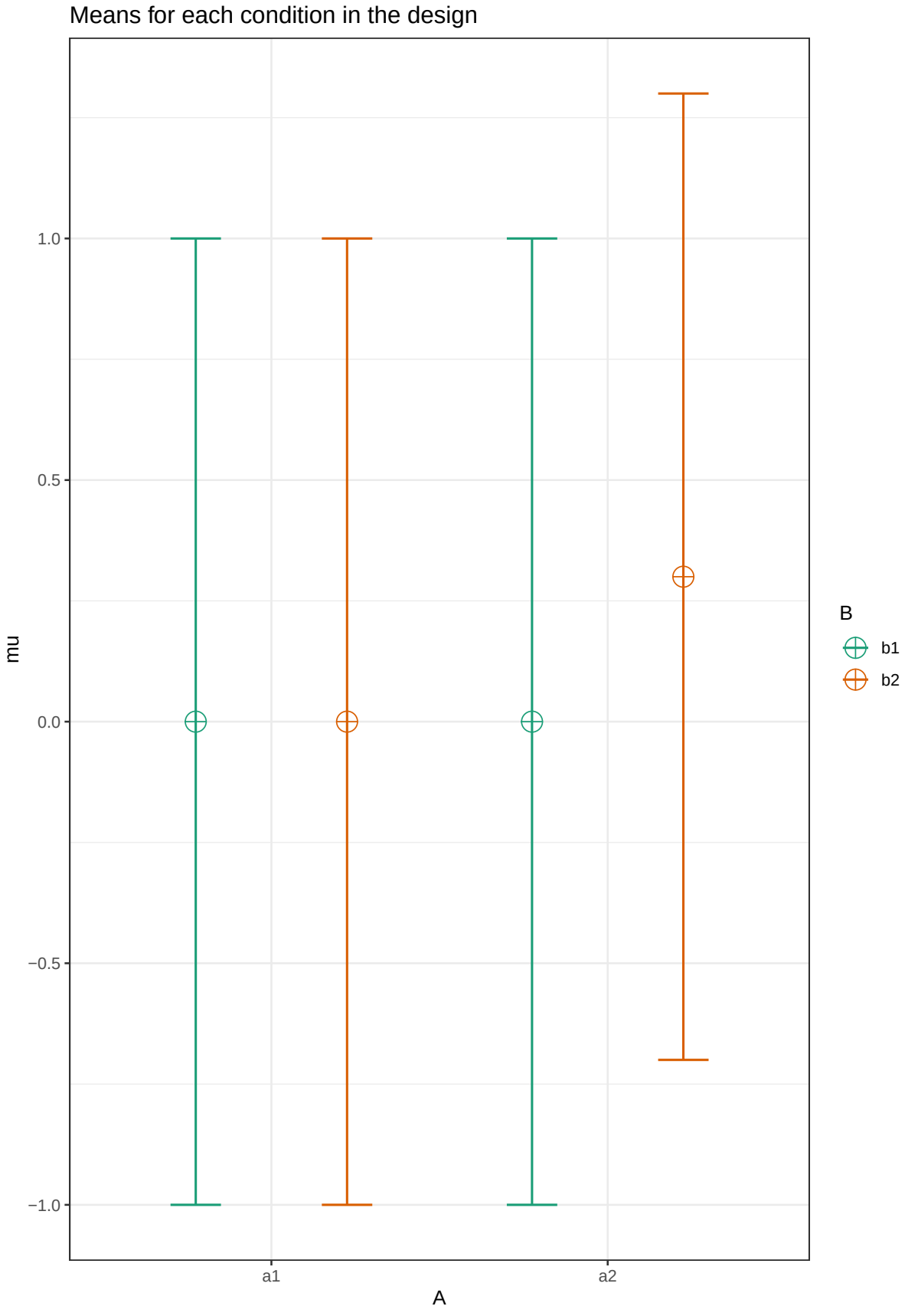
```
design_result <- ANOVA_design(design = string,
                              n = 20,
                              mu = c(0,0,0,0.3),
                              sd = 1,
                              r <- c(
                                0.8, 0.4, 0.4,
                                0.4, 0.4,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


Means for each condition in the design
mu
1.0
0.5
0.0
−0.5
−1.0
a1
a2
A
B
b1
b2

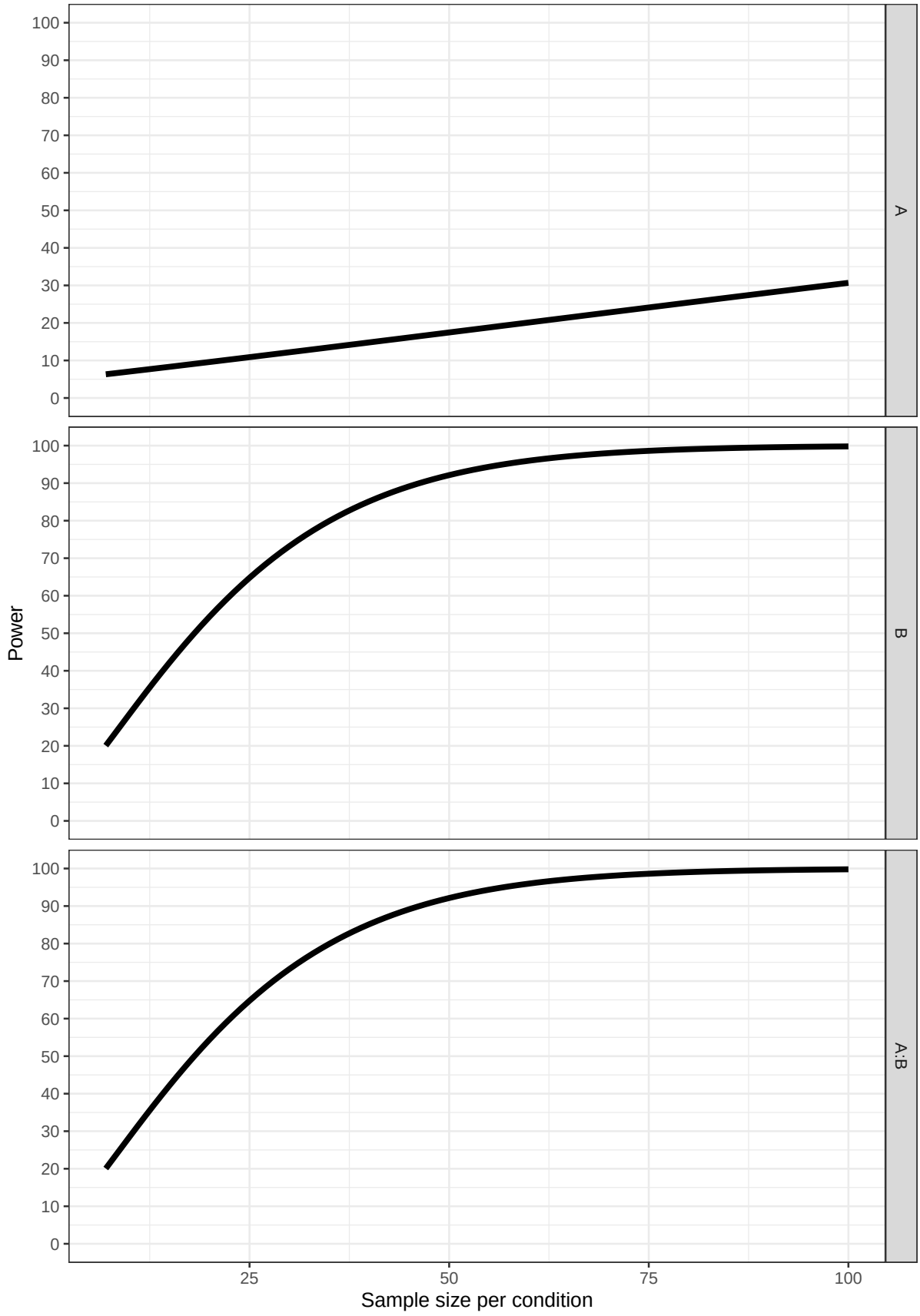
```
p_e <- plot_power(design_result,
                      max_n = 100)
```


Power
100
90
80
70
60
50
40
30
20
10
0
A
B
A:B
25
50
75
100
Sample size per condition

```
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)
```


Means for each condition in the design
mu
1.0
0.5
0.0
−0.5
−1.0
a1
a2
A
B
b1
b2

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


Power
100
90
80
70
60
50
40
30
20
10
0
A
B
A:B
25
50
75
100
Sample size per condition

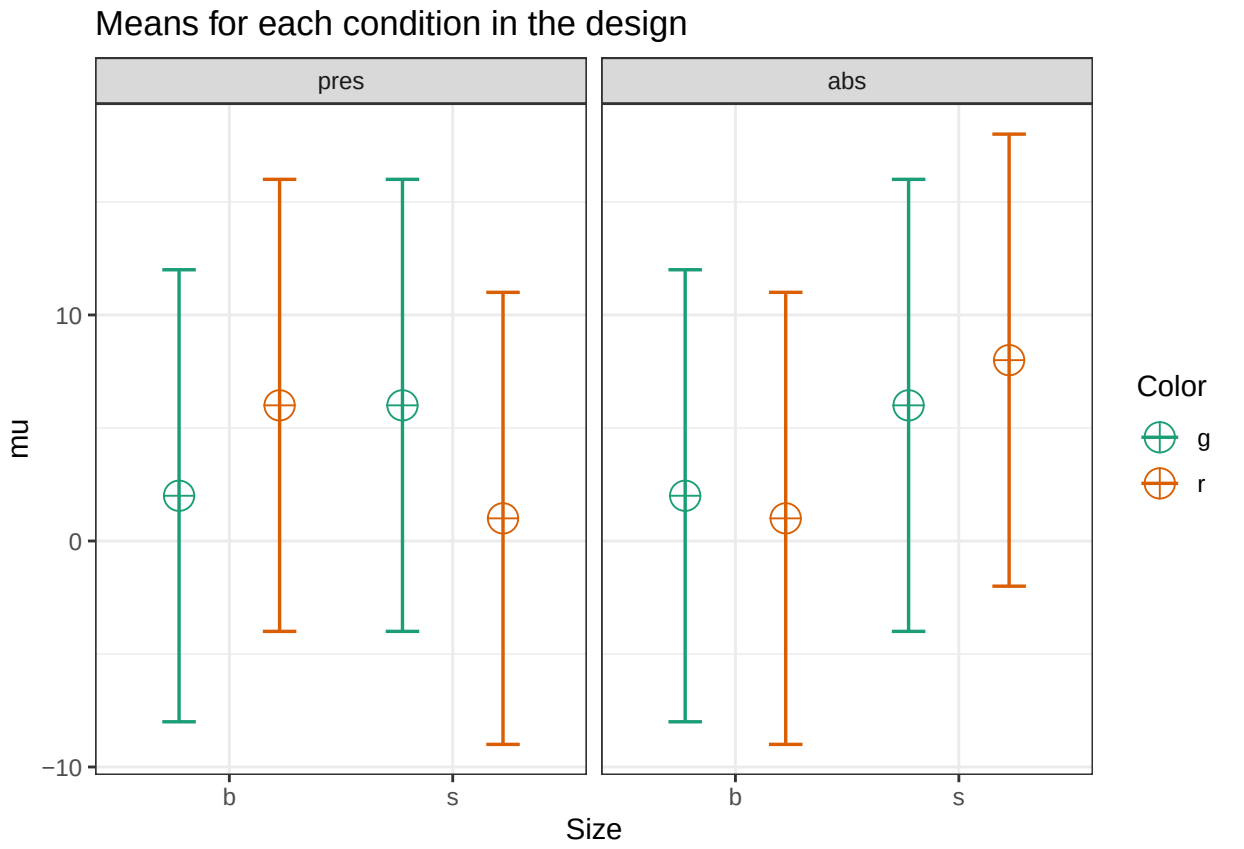
Chapter 16

Analytic Power for Three-way Interactions

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

Let's start with a 2x2x2 between subjects design. We collect 50 participants in each between participant condition (so 400 participants in total - 50x2x2x2).

```
# 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", #describe the design
                              n = 50, #sample size per group
                              mu = c(2, 2, 6, 1, 6, 6, 1, 8), #pattern of means
                              sd = 10, #standard deviation
                              labelnames = labelnames) #names of labels
```




```
# Power based on simulations
ANOVA_power(design_result, nsims = nsims)
```

```
## Power and Effect sizes for ANOVA tests
##                       power effect_size
## anova_Size               63    0.019255
## anova_Color               9    0.003026
## anova_Load                9    0.003036
## anova_Size:Color         32    0.008320
## anova_Size:Load          82    0.023807
## anova_Color:Load          8    0.003160
## anova_Size:Color:Load    79    0.024160
##
## Power and Effect sizes for contrasts
##                                                    power effect_size
## p_Size_b_Color_g_Load_pres_Size_b_Color_g_Load_abs     6    0.005541
## p_Size_b_Color_g_Load_pres_Size_b_Color_r_Load_pres   49    0.376156
## p_Size_b_Color_g_Load_pres_Size_b_Color_r_Load_abs     8   -0.102060
## p_Size_b_Color_g_Load_pres_Size_s_Color_g_Load_pres   55    0.406926
## p_Size_b_Color_g_Load_pres_Size_s_Color_g_Load_abs    53    0.411704
## p_Size_b_Color_g_Load_pres_Size_s_Color_r_Load_pres    5   -0.101535
## p_Size_b_Color_g_Load_pres_Size_s_Color_r_Load_abs    81    0.591370
## p_Size_b_Color_g_Load_abs_Size_b_Color_r_Load_pres    38    0.371205
## p_Size_b_Color_g_Load_abs_Size_b_Color_r_Load_abs      8   -0.107769
## p_Size_b_Color_g_Load_abs_Size_s_Color_g_Load_pres    49    0.399480
```

```
## p_Size_b_Color_g_Load_abs_Size_s_Color_g_Load_abs      44    0.407804
## p_Size_b_Color_g_Load_abs_Size_s_Color_r_Load_pres     10   -0.107625
## p_Size_b_Color_g_Load_abs_Size_s_Color_r_Load_abs      74    0.585699
## p_Size_b_Color_r_Load_pres_Size_b_Color_r_Load_abs     59   -0.474239
## p_Size_b_Color_r_Load_pres_Size_s_Color_g_Load_pres    10    0.026659
## p_Size_b_Color_r_Load_pres_Size_s_Color_g_Load_abs     12    0.033579
## p_Size_b_Color_r_Load_pres_Size_s_Color_r_Load_pres    67   -0.480681
## p_Size_b_Color_r_Load_pres_Size_s_Color_r_Load_abs     20    0.214595
## p_Size_b_Color_r_Load_abs_Size_s_Color_g_Load_pres     73    0.506063
## p_Size_b_Color_r_Load_abs_Size_s_Color_g_Load_abs      71    0.512447
## p_Size_b_Color_r_Load_abs_Size_s_Color_r_Load_pres      5   -0.003292
## p_Size_b_Color_r_Load_abs_Size_s_Color_r_Load_abs      89    0.689687
## p_Size_s_Color_g_Load_pres_Size_s_Color_g_Load_abs      6    0.007411
## p_Size_s_Color_g_Load_pres_Size_s_Color_r_Load_pres    71   -0.511627
## p_Size_s_Color_g_Load_pres_Size_s_Color_r_Load_abs     20    0.190130
## p_Size_s_Color_g_Load_abs_Size_s_Color_r_Load_pres     74   -0.516444
## p_Size_s_Color_g_Load_abs_Size_s_Color_r_Load_abs      17    0.183468
## p_Size_s_Color_r_Load_pres_Size_s_Color_r_Load_abs     93    0.696016
```

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

```
## [1] 0.7033333
```

```
power_analytic$power_B
```

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

```
power_analytic$power_C
```

```
## [1] 0.07895539
```

```
power_analytic$power_AB
```

```
## [1] 0.3217471
```

```
power_analytic$power_AC
```

```
## [1] 0.8491491
```

```
power_analytic$power_BC
```

```
## [1] 0.07895539
```

```
power_analytic$power_ABC
```

```
## [1] 0.8491491
```

```
power_analytic$eta_p_2_A
```

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

```
power_analytic$eta_p_2_B
```

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

```
power_analytic$eta_p_2_C
```

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

```
power_analytic$eta_p_2_AB
```

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

```
power_analytic$eta_p_2_AC
```

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

```
power_analytic$eta_p_2_BC
```

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

```
power_analytic$eta_p_2_ABC
```

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

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

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

```
## [1] 0.8491491
```

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

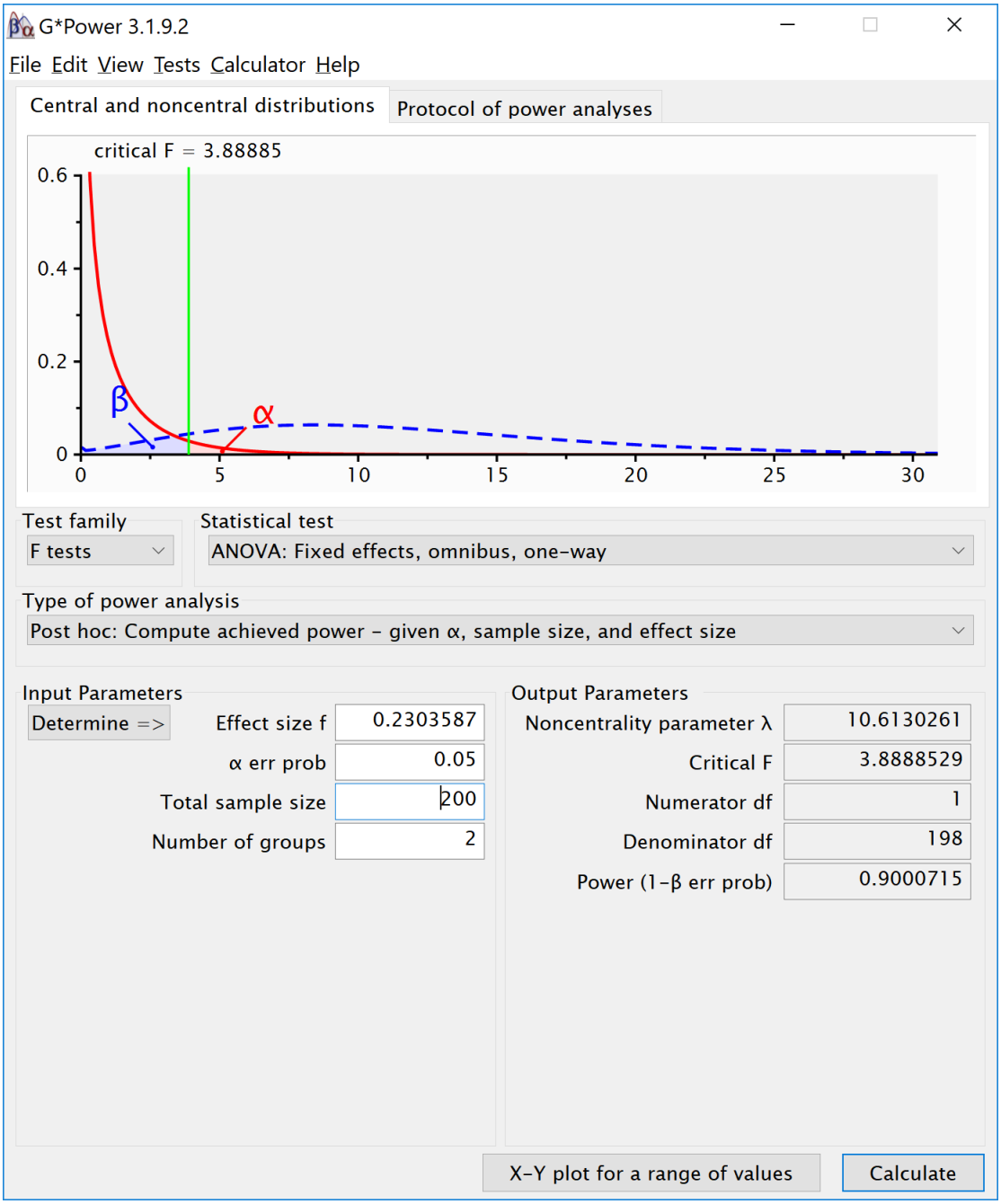
```
## [1] 0.15
```

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

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

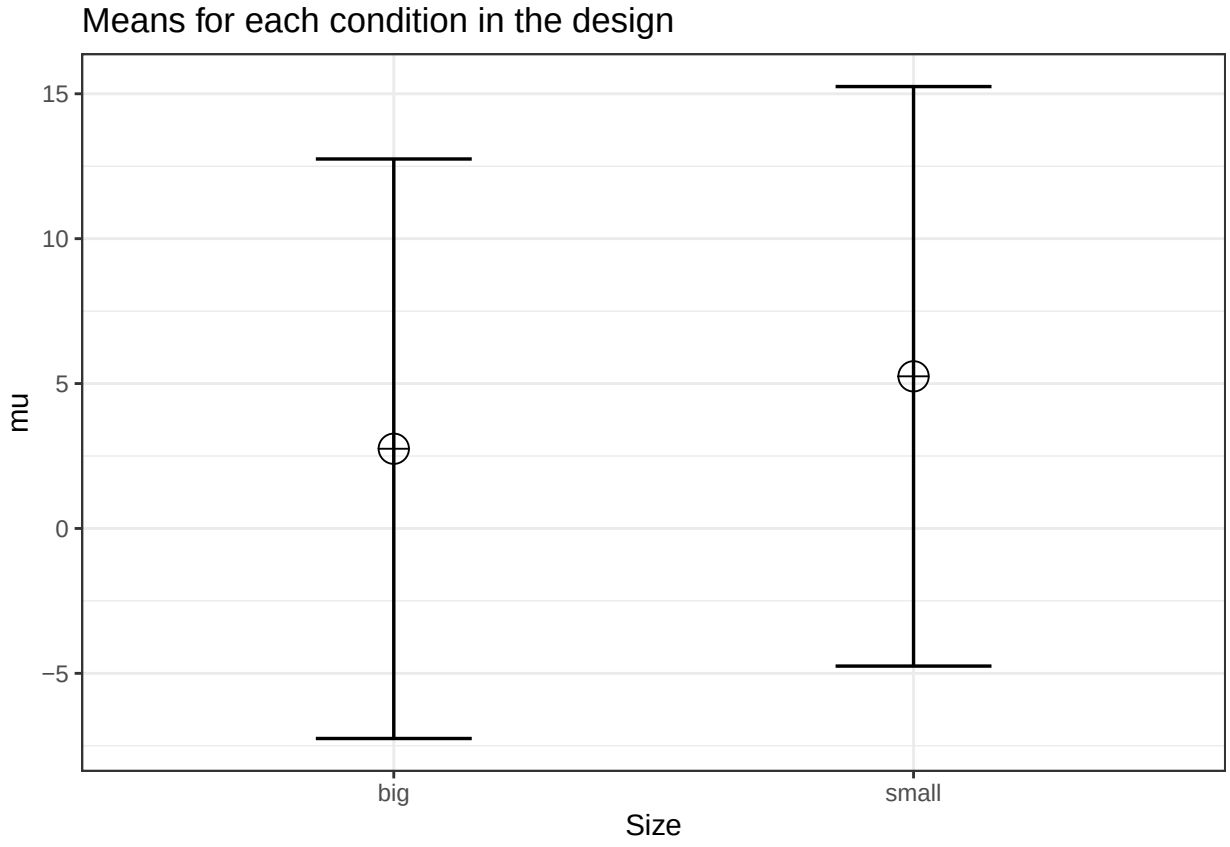
```
# We can double check the critical F value
power_analytic$F_critical_AC
```

```
## [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 repeating the simulation 1000.000 times instead of 100.000.

```
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)
```




```
# Power based on simulations
ANOVA_power(design_result, nsims = nsims)
```

```
## Power and Effect sizes for ANOVA tests
##            power effect_size
```

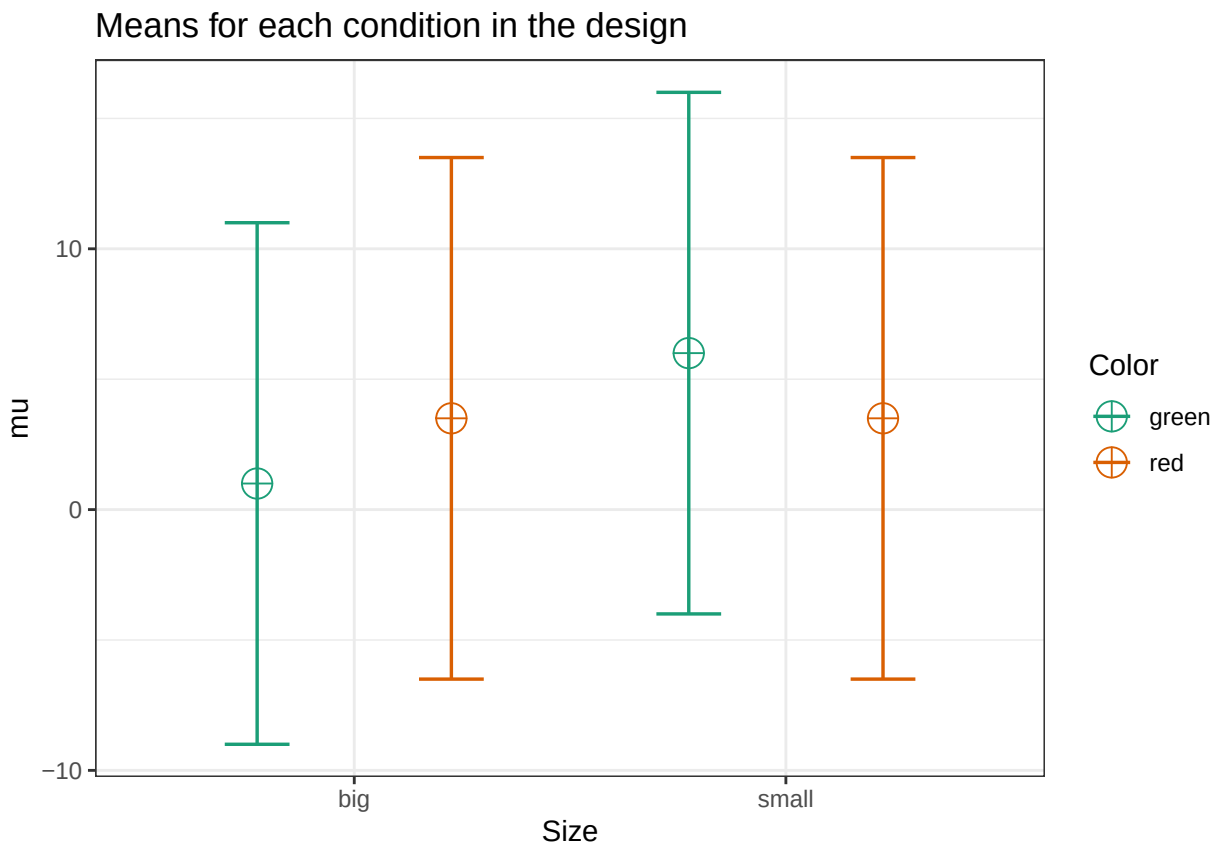
```
## anova_Size    72     0.01741
##
## Power and Effect sizes for contrasts
##                         power effect_size
## p_Size_big_Size_small    72      0.2526
```

```
# Power based on analytical solution
power_oneway_between(design_result)$power #using default alpha level of .05
```

```
## [1] 0.7033333
```

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)
```



```
# Power based on simulations
ANOVA_power(design_result, nsims = nsims)
```

```
## Power and Effect sizes for ANOVA tests
##                  power effect_size
## anova_Size          71     0.01622
## anova_Color          6     0.00259
## anova_Size:Color    76     0.01958
##
## Power and Effect sizes for contrasts
##                                                 power effect_size
## p_Size_big_Color_green_Size_big_Color_red          48      0.2848
## p_Size_big_Color_green_Size_small_Color_green      98      0.5037
## p_Size_big_Color_green_Size_small_Color_red        39      0.2594
## p_Size_big_Color_red_Size_small_Color_green        34      0.2207
## p_Size_big_Color_red_Size_small_Color_red           6     -0.0283
## p_Size_small_Color_green_Size_small_Color_red      41     -0.2479
```

```
# Power based on analytical solution
power_res <- power_twoway_between(design_result) #using default alpha level of .05
power_res$power_A
```

```
## [1] 0.7033228
```

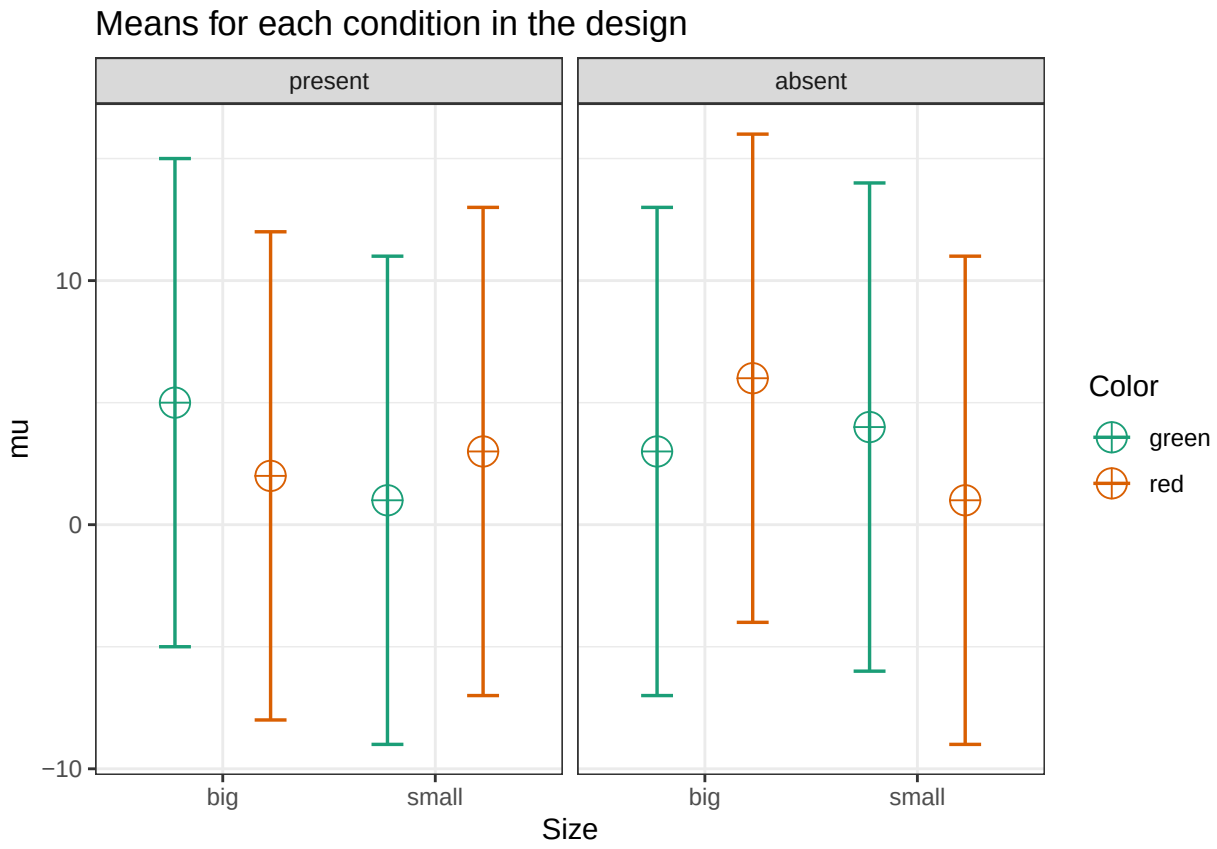
```
power_res$power_B
```

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

```
power_res$power_AB
```

```
## [1] 0.7033228
```

```
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)
```




```
# Power for the given N in the design_result
ANOVA_power(design_result, nsims = nsims)
```

```
## Power and Effect sizes for ANOVA tests
##                                 power effect_size
## anova_Size                         44    0.010531
## anova_Color                        10    0.003422
## anova_CognitiveLoad                13    0.004081
## anova_Size:Color                    7    0.002911
## anova_Size:CognitiveLoad            2    0.002452
## anova_Color:CognitiveLoad          12    0.003282
## anova_Size:Color:CognitiveLoad     84    0.021377
##
## Power and Effect sizes for contrasts
##                                                                                                 power
## p_Size_big_Color_green_CognitiveLoad_present_Size_big_Color_green_CognitiveLoad_absent              18
## p_Size_big_Color_green_CognitiveLoad_present_Size_big_Color_red_CognitiveLoad_present               31
## p_Size_big_Color_green_CognitiveLoad_present_Size_big_Color_red_CognitiveLoad_absent                9
## p_Size_big_Color_green_CognitiveLoad_present_Size_small_Color_green_CognitiveLoad_present          50
## p_Size_big_Color_green_CognitiveLoad_present_Size_small_Color_green_CognitiveLoad_absent           8
## p_Size_big_Color_green_CognitiveLoad_present_Size_small_Color_red_CognitiveLoad_present            17
## p_Size_big_Color_green_CognitiveLoad_present_Size_small_Color_red_CognitiveLoad_absent             57
## p_Size_big_Color_green_CognitiveLoad_absent_Size_big_Color_red_CognitiveLoad_present                7
## p_Size_big_Color_green_CognitiveLoad_absent_Size_big_Color_red_CognitiveLoad_absent                33
## p_Size_big_Color_green_CognitiveLoad_absent_Size_small_Color_green_CognitiveLoad_present           18
```

```
## p_Size_big_Color_green_CognitiveLoad_absent_Size_small_Color_green_CognitiveLoad_absent        8
## p_Size_big_Color_green_CognitiveLoad_absent_Size_small_Color_red_CognitiveLoad_present        5
## p_Size_big_Color_green_CognitiveLoad_absent_Size_small_Color_red_CognitiveLoad_absent        21
## p_Size_big_Color_red_CognitiveLoad_present_Size_big_Color_red_CognitiveLoad_absent           47
## p_Size_big_Color_red_CognitiveLoad_present_Size_small_Color_green_CognitiveLoad_present      14
## p_Size_big_Color_red_CognitiveLoad_present_Size_small_Color_green_CognitiveLoad_absent       16
## p_Size_big_Color_red_CognitiveLoad_present_Size_small_Color_red_CognitiveLoad_present         6
## p_Size_big_Color_red_CognitiveLoad_present_Size_small_Color_red_CognitiveLoad_absent        18
## p_Size_big_Color_red_CognitiveLoad_absent_Size_small_Color_green_CognitiveLoad_present      72
## p_Size_big_Color_red_CognitiveLoad_absent_Size_small_Color_green_CognitiveLoad_absent       15
## p_Size_big_Color_red_CognitiveLoad_absent_Size_small_Color_red_CognitiveLoad_present        30
## p_Size_big_Color_red_CognitiveLoad_absent_Size_small_Color_red_CognitiveLoad_absent         70
## p_Size_small_Color_green_CognitiveLoad_present_Size_small_Color_green_CognitiveLoad_absent  34
## p_Size_small_Color_green_CognitiveLoad_present_Size_small_Color_red_CognitiveLoad_present   16
## p_Size_small_Color_green_CognitiveLoad_present_Size_small_Color_red_CognitiveLoad_absent     5
## p_Size_small_Color_green_CognitiveLoad_absent_Size_small_Color_red_CognitiveLoad_present     7
## p_Size_small_Color_green_CognitiveLoad_absent_Size_small_Color_red_CognitiveLoad_absent      45
## p_Size_small_Color_red_CognitiveLoad_present_Size_small_Color_red_CognitiveLoad_absent       25
##                                                                                          effect_siz
## p_Size_big_Color_green_CognitiveLoad_present_Size_big_Color_green_CognitiveLoad_absent      -0.2018
## p_Size_big_Color_green_CognitiveLoad_present_Size_big_Color_red_CognitiveLoad_present       -0.2753
## p_Size_big_Color_green_CognitiveLoad_present_Size_big_Color_red_CognitiveLoad_absent         0.1033
## p_Size_big_Color_green_CognitiveLoad_present_Size_small_Color_green_CognitiveLoad_present   -0.3987
## p_Size_big_Color_green_CognitiveLoad_present_Size_small_Color_green_CognitiveLoad_absent    -0.0759
## p_Size_big_Color_green_CognitiveLoad_present_Size_small_Color_red_CognitiveLoad_present     -0.1859
## p_Size_big_Color_green_CognitiveLoad_present_Size_small_Color_red_CognitiveLoad_absent      -0.4153
## p_Size_big_Color_green_CognitiveLoad_absent_Size_big_Color_red_CognitiveLoad_present        -0.0783
## p_Size_big_Color_green_CognitiveLoad_absent_Size_big_Color_red_CognitiveLoad_absent          0.3067
## p_Size_big_Color_green_CognitiveLoad_absent_Size_small_Color_green_CognitiveLoad_present   -0.1988
## p_Size_big_Color_green_CognitiveLoad_absent_Size_small_Color_green_CognitiveLoad_absent     0.1249
## p_Size_big_Color_green_CognitiveLoad_absent_Size_small_Color_red_CognitiveLoad_present      0.0133
## p_Size_big_Color_green_CognitiveLoad_absent_Size_small_Color_red_CognitiveLoad_absent      -0.2169
## p_Size_big_Color_red_CognitiveLoad_present_Size_big_Color_red_CognitiveLoad_absent         0.3801
## p_Size_big_Color_red_CognitiveLoad_present_Size_small_Color_green_CognitiveLoad_present    -0.1174
## p_Size_big_Color_red_CognitiveLoad_present_Size_small_Color_green_CognitiveLoad_absent     0.2013
## p_Size_big_Color_red_CognitiveLoad_present_Size_small_Color_red_CognitiveLoad_present      0.0886
## p_Size_big_Color_red_CognitiveLoad_present_Size_small_Color_red_CognitiveLoad_absent      -0.1377
## p_Size_big_Color_red_CognitiveLoad_absent_Size_small_Color_green_CognitiveLoad_present    -0.5046
## p_Size_big_Color_red_CognitiveLoad_absent_Size_small_Color_green_CognitiveLoad_absent     -0.1779
## p_Size_big_Color_red_CognitiveLoad_absent_Size_small_Color_red_CognitiveLoad_present      -0.2891
## p_Size_big_Color_red_CognitiveLoad_absent_Size_small_Color_red_CognitiveLoad_absent       -0.5204
## p_Size_small_Color_green_CognitiveLoad_present_Size_small_Color_green_CognitiveLoad_absent  0.3214
## p_Size_small_Color_green_CognitiveLoad_present_Size_small_Color_red_CognitiveLoad_present   0.2117
## p_Size_small_Color_green_CognitiveLoad_present_Size_small_Color_red_CognitiveLoad_absent   -0.0179
## p_Size_small_Color_green_CognitiveLoad_absent_Size_small_Color_red_CognitiveLoad_present   -0.1115
## p_Size_small_Color_green_CognitiveLoad_absent_Size_small_Color_red_CognitiveLoad_absent    -0.3397
## p_Size_small_Color_red_CognitiveLoad_present_Size_small_Color_red_CognitiveLoad_absent     -0.2275
```

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

```
## [1] 0.415306
```

```
power_analytic$power_B
```

```
## [1] 0.05715533
```

```
power_analytic$power_C
```

```
## [1] 0.1161827
```

```
power_analytic$power_AB
```

```
## [1] 0.05715533
```

```
power_analytic$power_AC
```

```
## [1] 0.05715533
```

```
power_analytic$power_BC
```

```
## [1] 0.05715533
```

```
power_analytic$power_ABC
```

```
## [1] 0.7833036
```

```
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
```

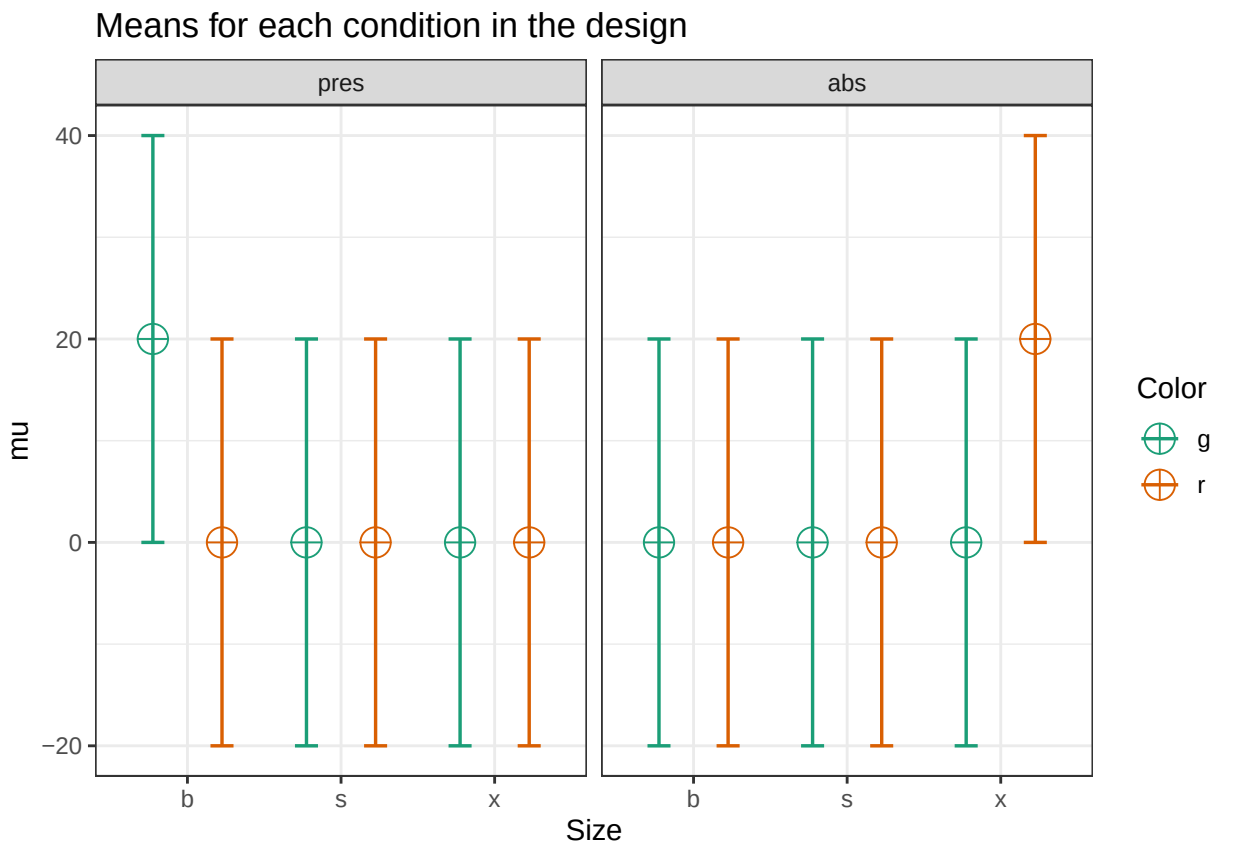
```
power_analytic$eta_p_2_ABC
```

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

In addition, ## Cohen's f for Three-way Interactions

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 sizI 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", #describe the design
                              n = 10, #sample size per group
                              mu = c(20, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 20), #pattern of means
                              sd = 20, #standard deviation
                              labelnames = labelnames) #names of labels
```



```
# Power based on simulations
# ANOVA_power(design_result, nsims = nsims)
#Analytical power calculation
power_analytic <- power_threeway_between(design_result)
power_analytic$power_A
```

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

```
power_analytic$power_B
```

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

```
power_analytic$power_C
```

```
## [1] 0.3478125
```

```
power_analytic$power_AB
```

```
## [1] 0.2414317
```

```
power_analytic$power_AC
```

```
## [1] 0.4928984
```

```
power_analytic$power_BC
```

```
## [1] 0.7752085
```

```
power_analytic$power_ABC
```

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

```
power_analytic$eta_p_2_A
```

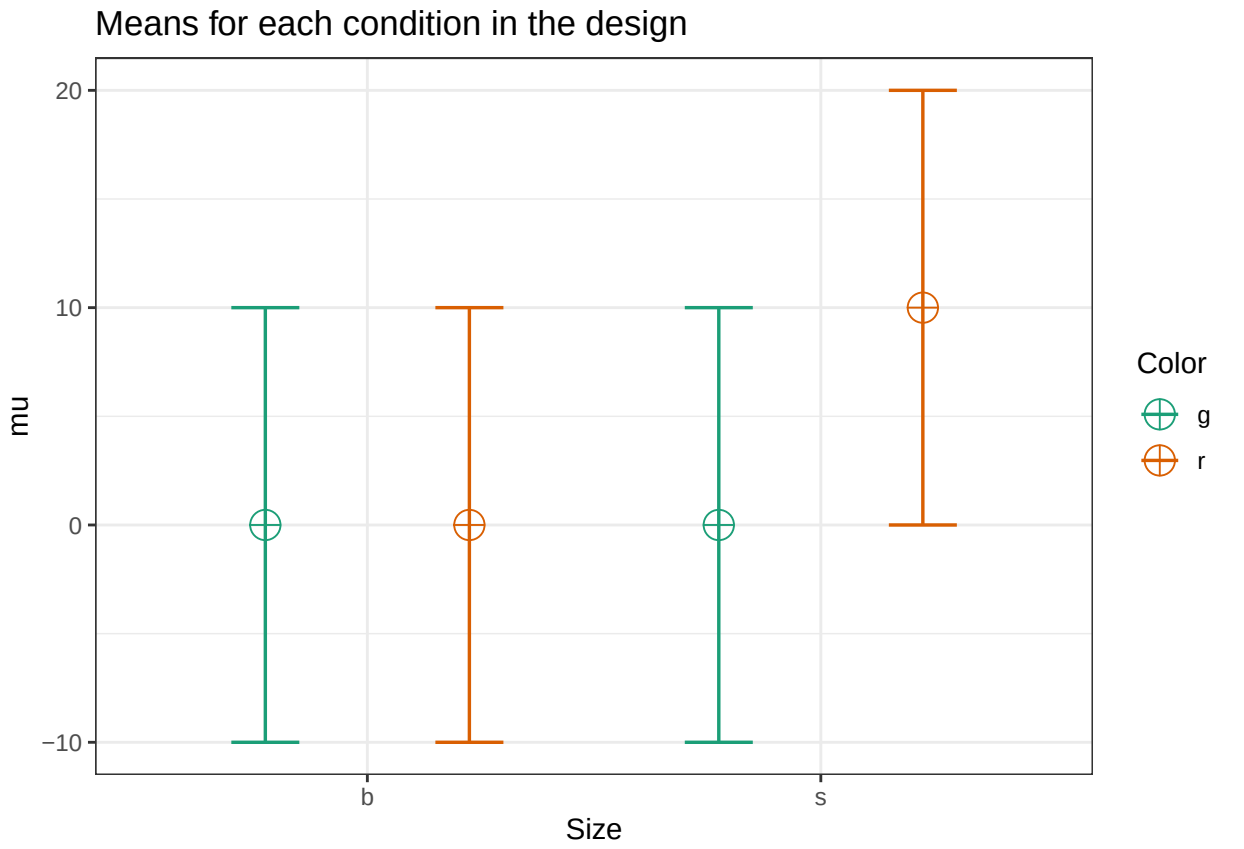
```
## [1] 0
```

```
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", #describe the design
                              n = 10, #sample size per group
                              mu = c(0, 0, 0, 10), #pattern of means
                              sd = 10, #standard deviation
                              labelnames = labelnames) #names of labels
```




```
# Power based on simulations
# ANOVA_power(design_result, nsims = nsims)
#Analytical power calculation
power_analytic <- power_twoway_between(design_result)
power_analytic$power_A
```

```
## [1] 0.3371329
```

```
power_analytic$eta_p_2_A
```

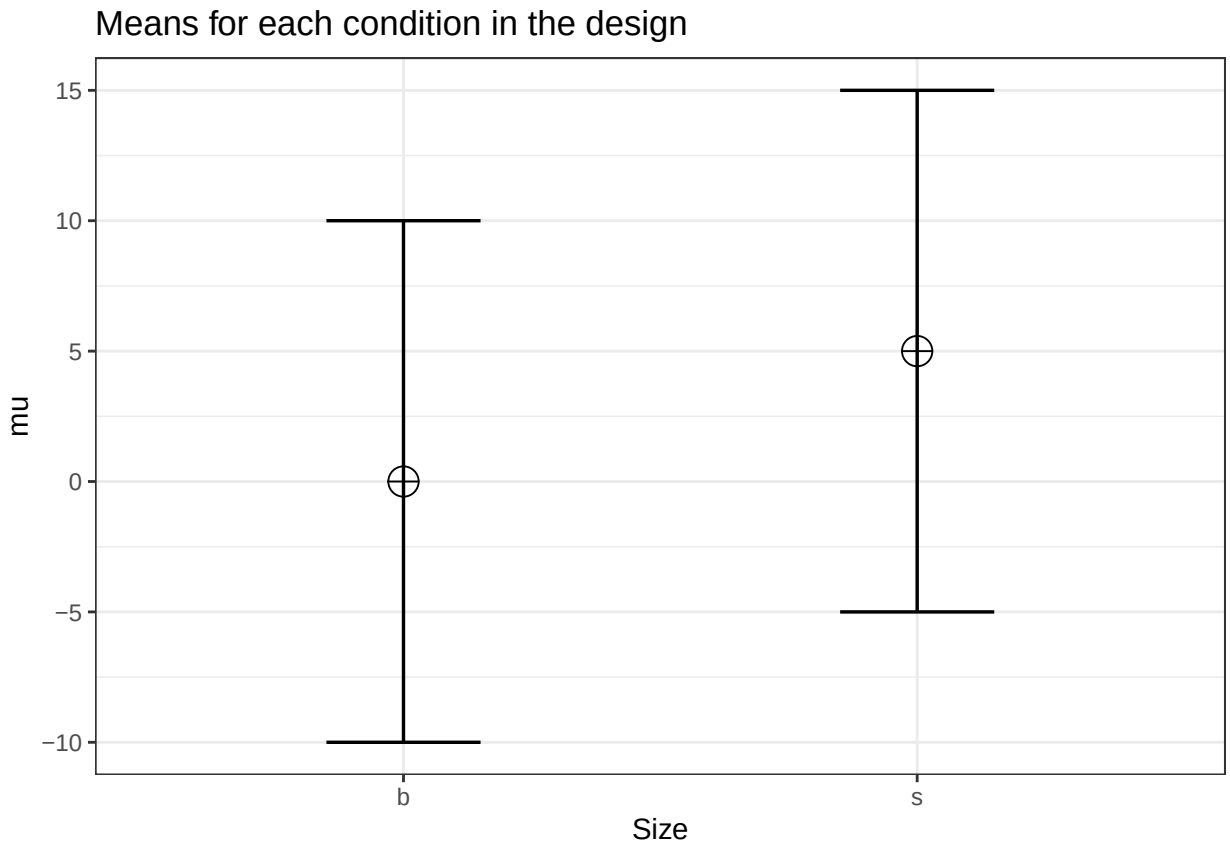
```
## [1] 0.05882353
```

```
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", #describe the design
                              n = 10, #sample size per group
                              mu = c(0, 5), #pattern of means
                              sd = 10, #standard deviation
                              labelnames = labelnames) #names of labels
```




```
# Power based on simulations
# ANOVA_power(design_result, nsims = nsims)
#Analytical power calculation
power_analytic <- power_oneway_between(design_result)
power_analytic$power
```

```
## [1] 0.1850957
```

```
power_analytic$eta_p_2
```

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

```
power_analytic$Cohen_f
```

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

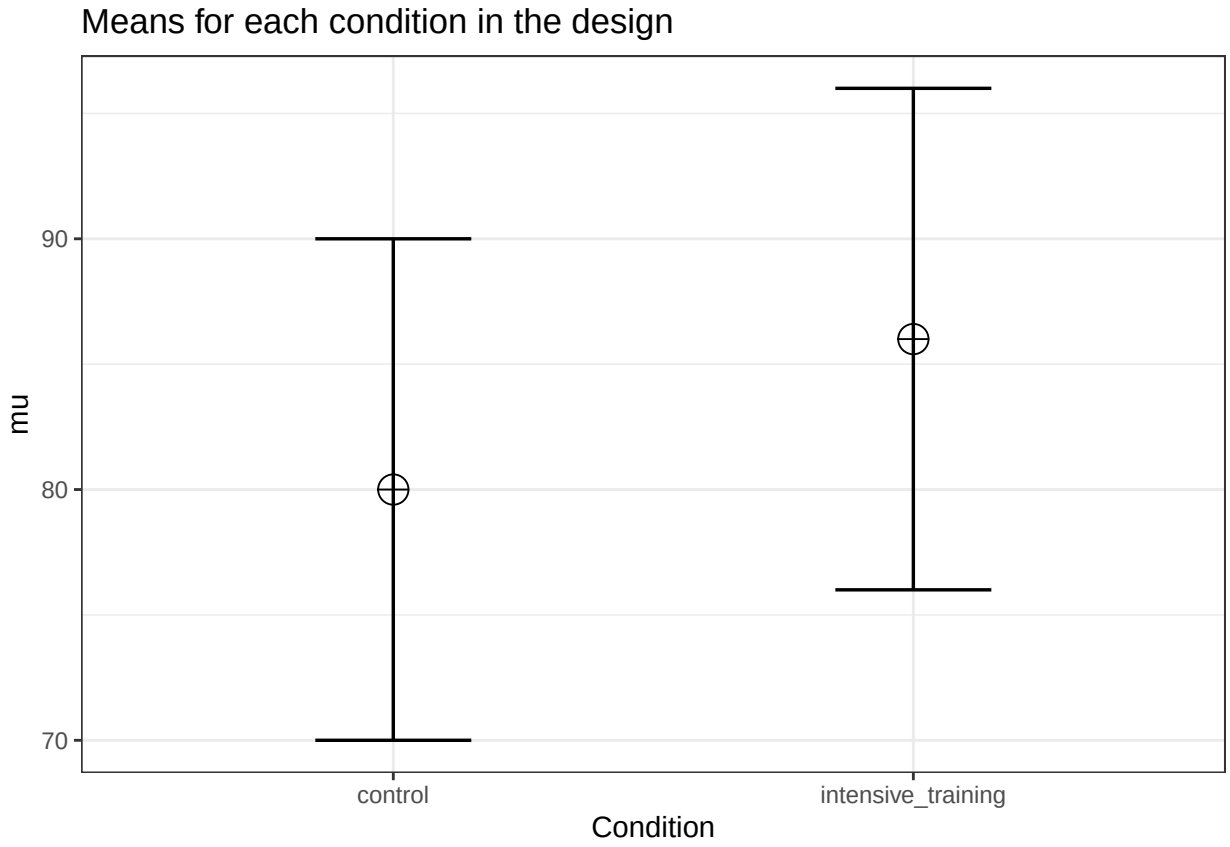
Chapter 17

Power for Design Variations

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




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

```
## [1] 0.8438754
```

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

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

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

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

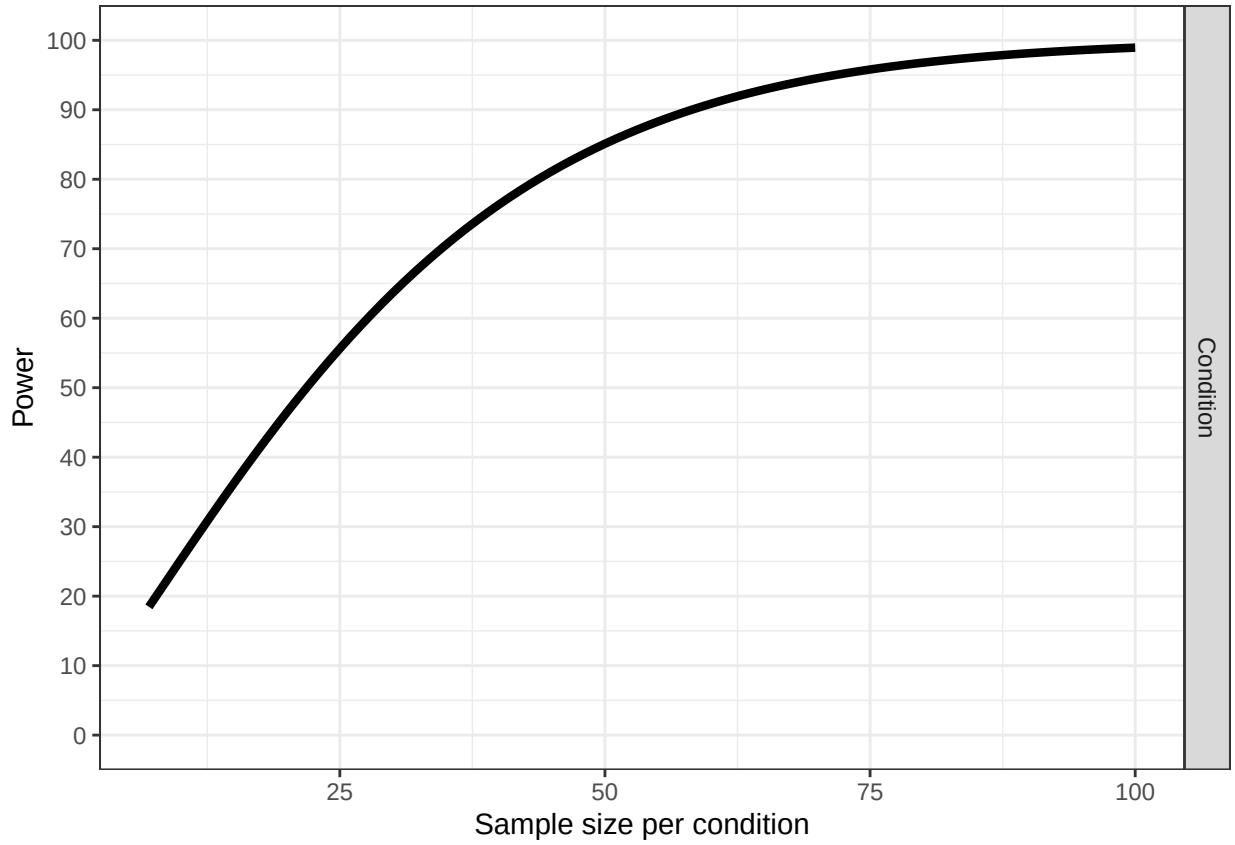
```
ANOVA_power(design_result, nsims = nsims)
```

```
## Power and Effect sizes for ANOVA tests
##                 power effect_size
## anova_Condition    86     0.09609
##
## Power and Effect sizes for contrasts
##                                                  power effect_size
## p_Condition_control_Condition_intensive_training    86      0.6237
```

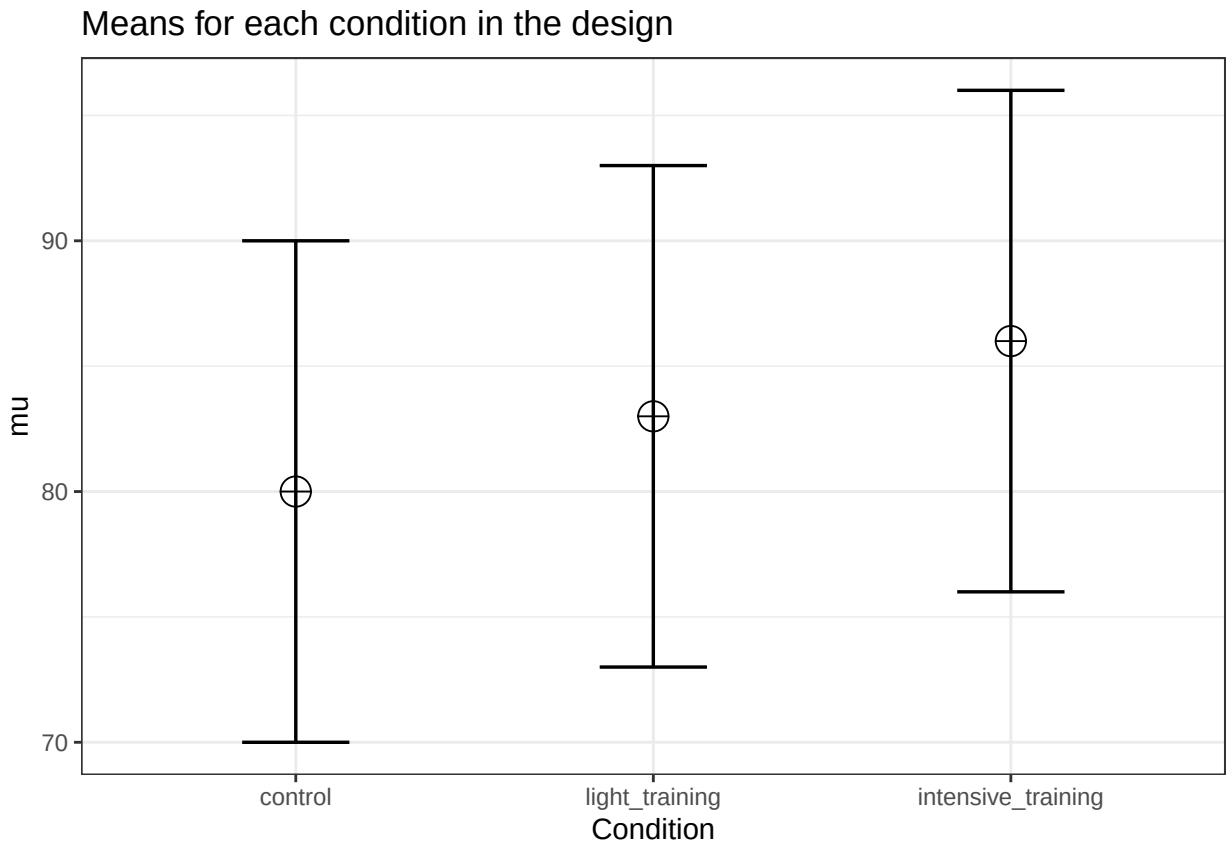
We now addd a condition. Let's assume the 'light training' condition falls in between the other two means.

And we can see power across sample sizes

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




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




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

```
## [1] 0.7616545
```

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

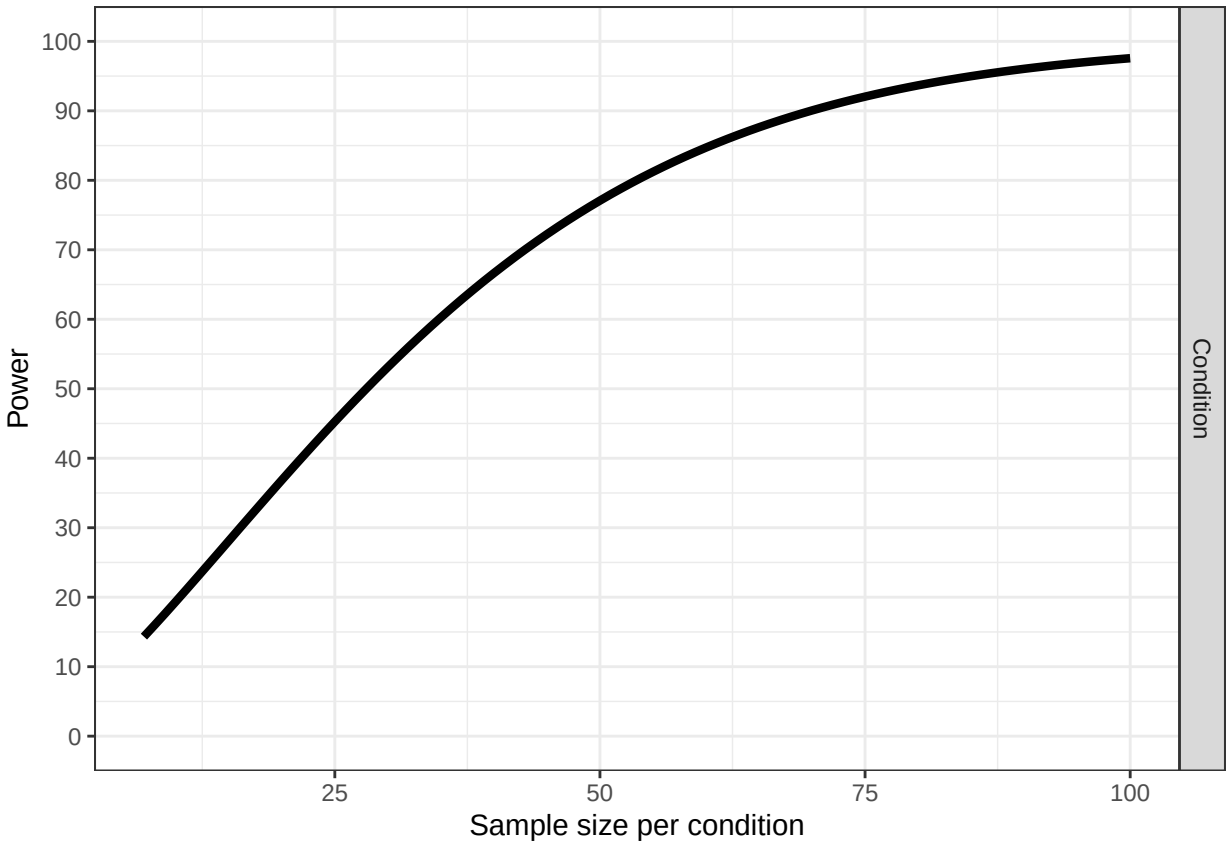
```
## [1] 0.244949
```

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

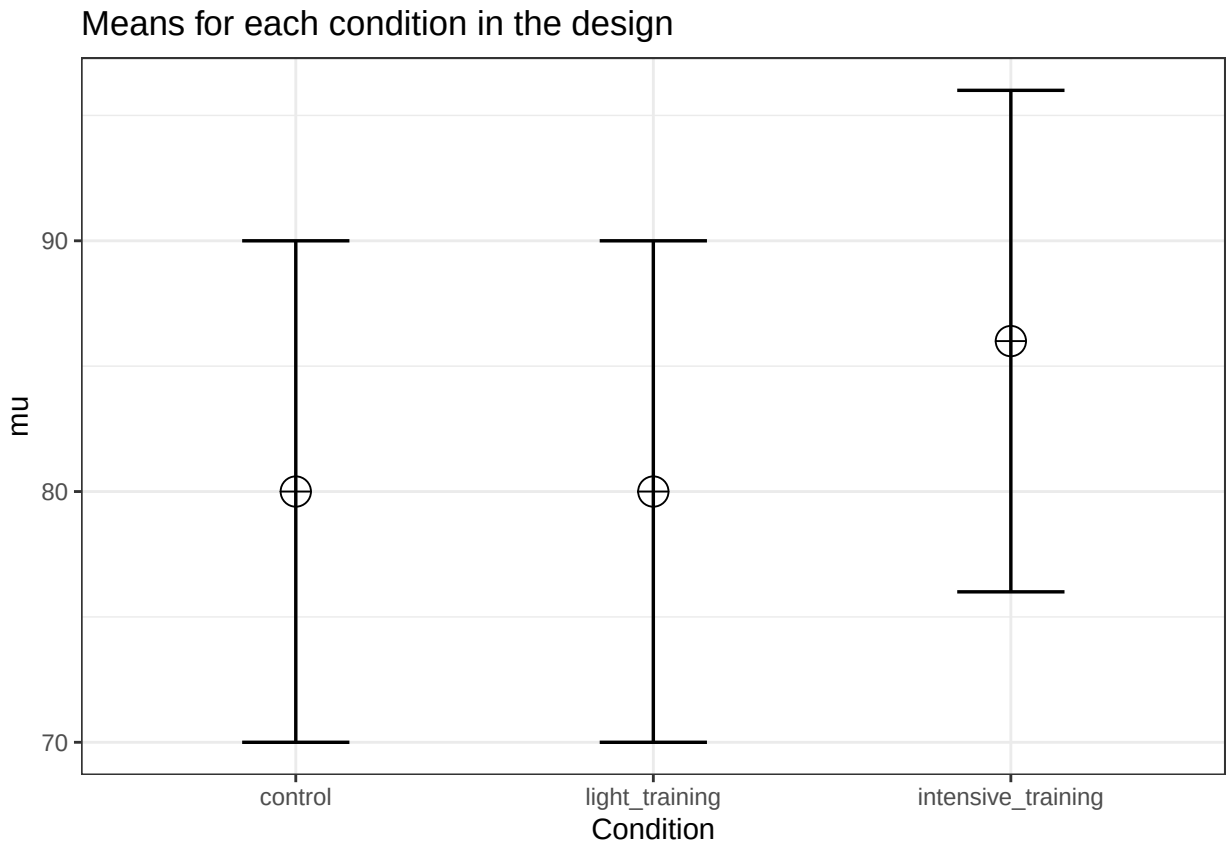
```
## [1] 0.05660377
```

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

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




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




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

```
## [1] 0.8762941
```

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

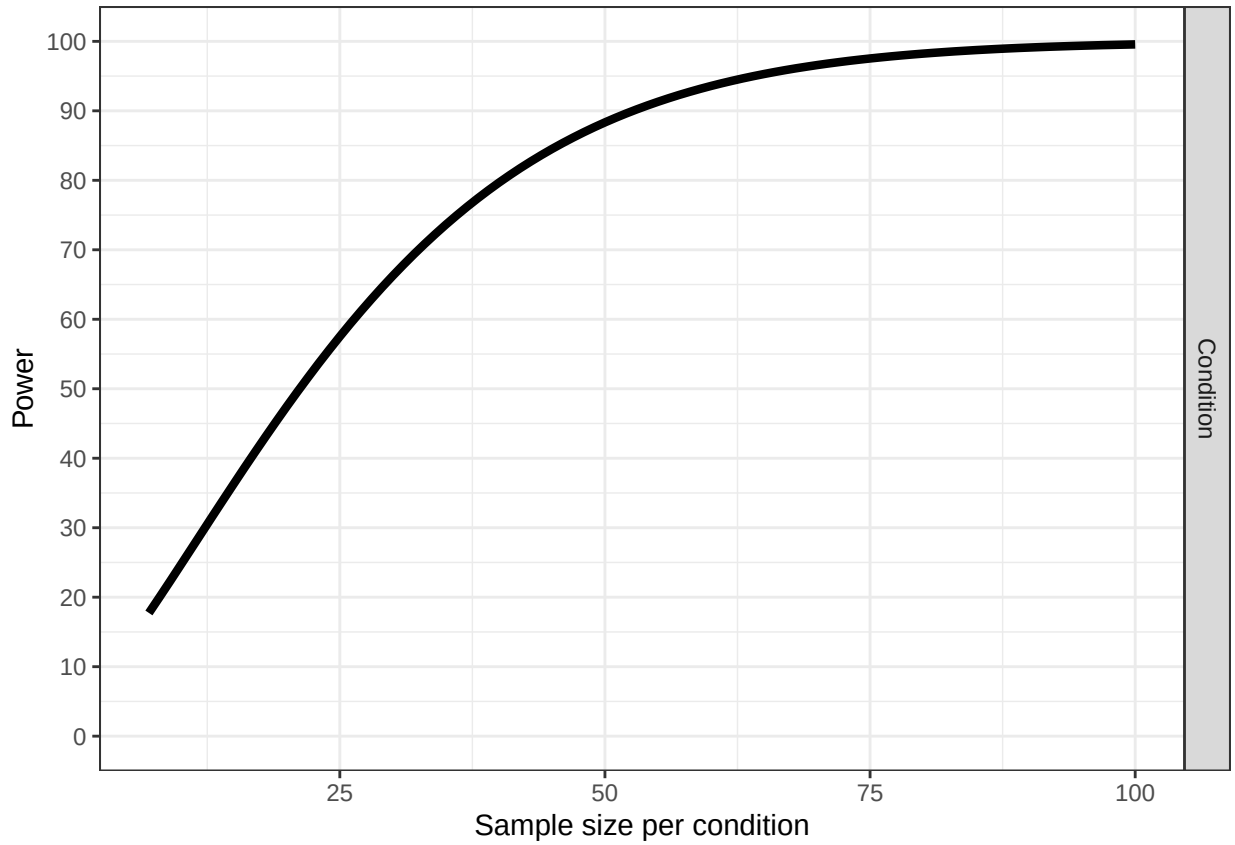
```
## [1] 0.2828427
```

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

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

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

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



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

```
#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 also reduces the power. Let's again do the calculations step by step when the extra group has a mean as extreme as one of the two original conditions.

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

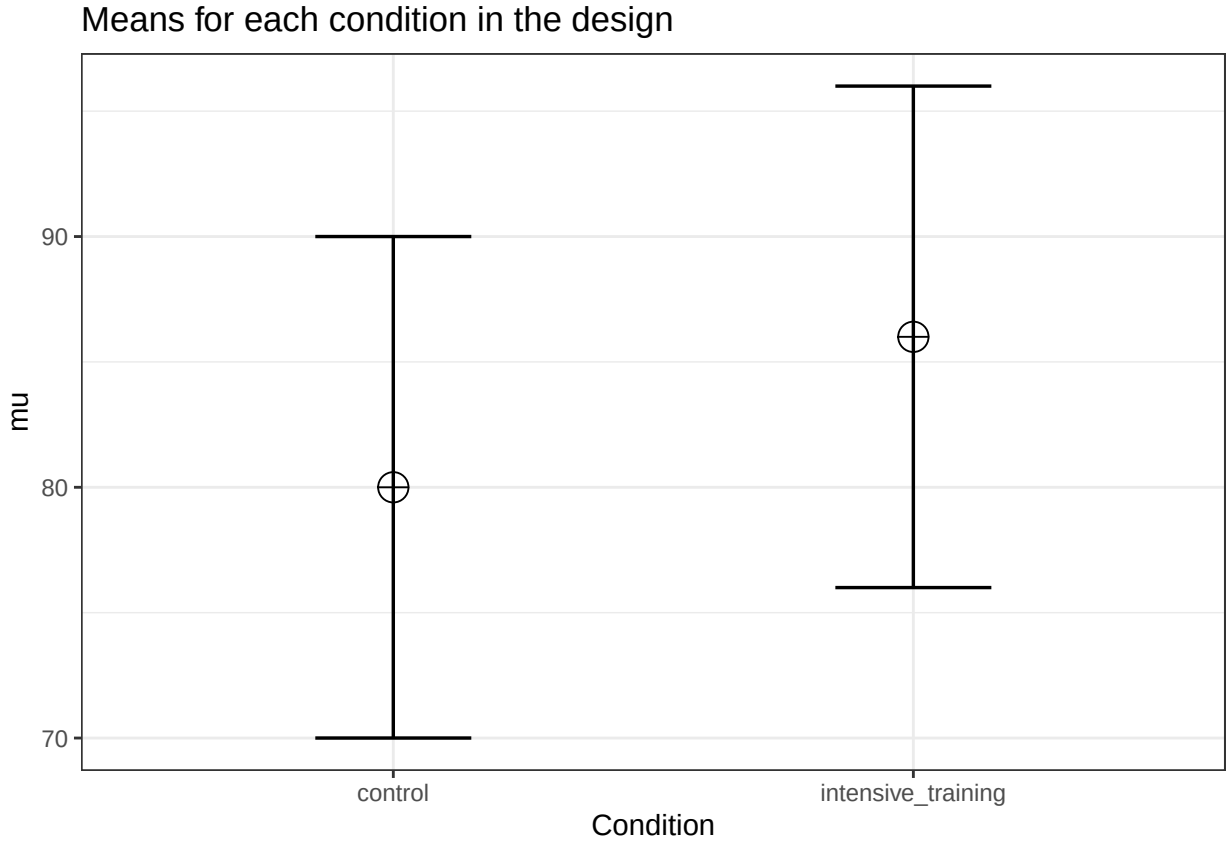
```
## [1] 0.2828427
```

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

17.1 Within Designs

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

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



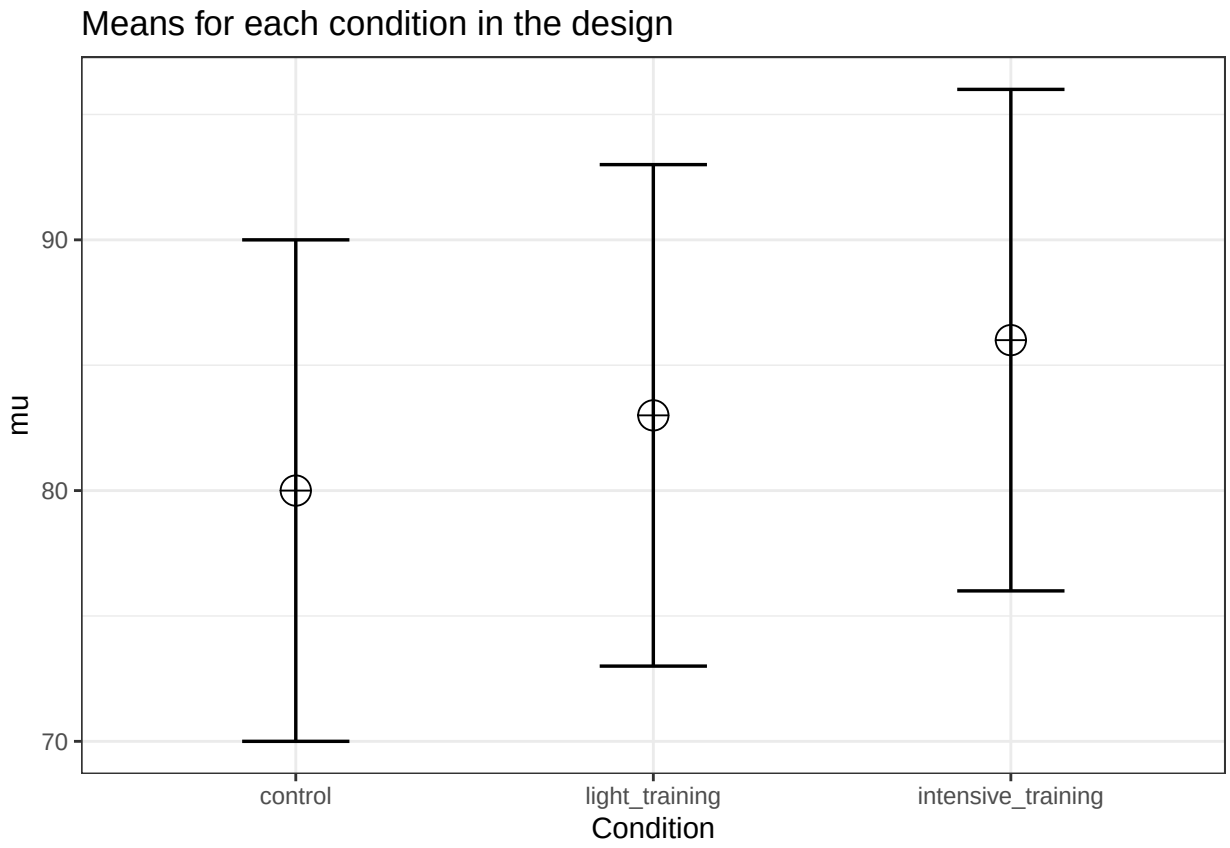

```
power_oneway_within(design_result)$power
```

```
## [1] 0.8366436
```

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) #All means are equal - so there is no real difference.
sd <- 10
labelnames <- c("Condition", "control", "light_training","intensive_training") #
```

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

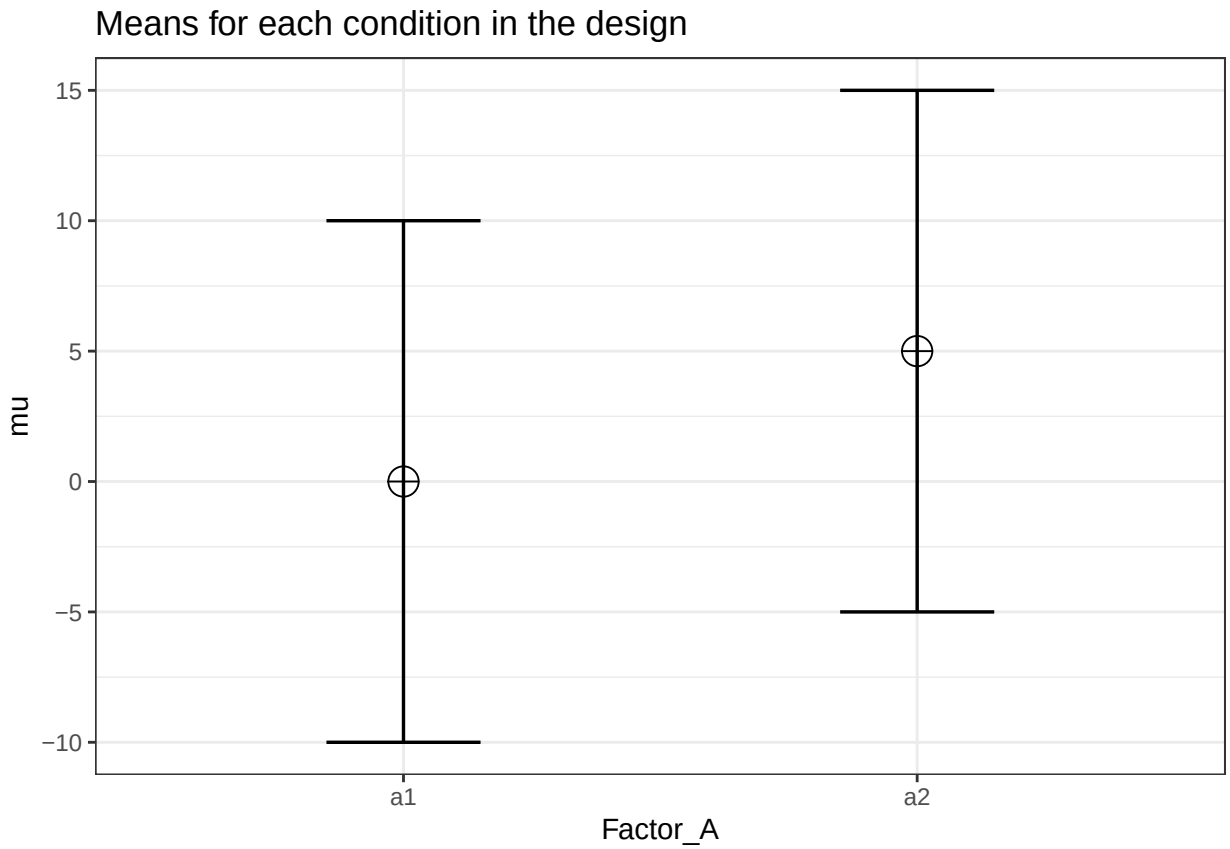



```
power_oneway_within(design_result)$power
```

```
## [1] 0.7570841
```

When we add a a condition in a within design where we expect the mean to be identical to the grand mean, we again see that the power decreases. This similarly shows that adding a condition that equals the grand mean to a within subject design does not come for free, but has a power cost.

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




```
power_oneway_within(design_result)$power
```

```
## [1] 0.7539647
```

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

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

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

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

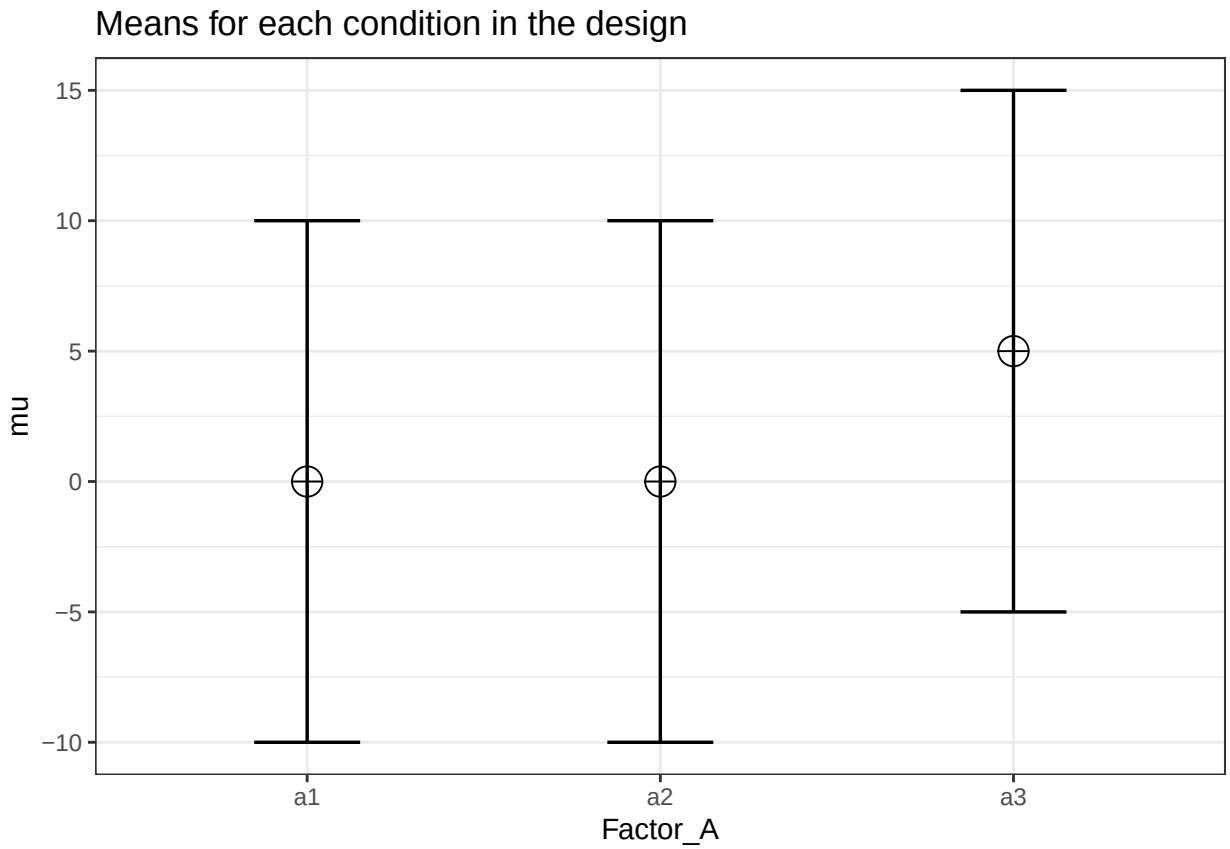
```
power_oneway_within(design_result)$lambda
```

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

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

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

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




```
power_oneway_within(design_result)$power
```

```
## [1] 0.7937037
```

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

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

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

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

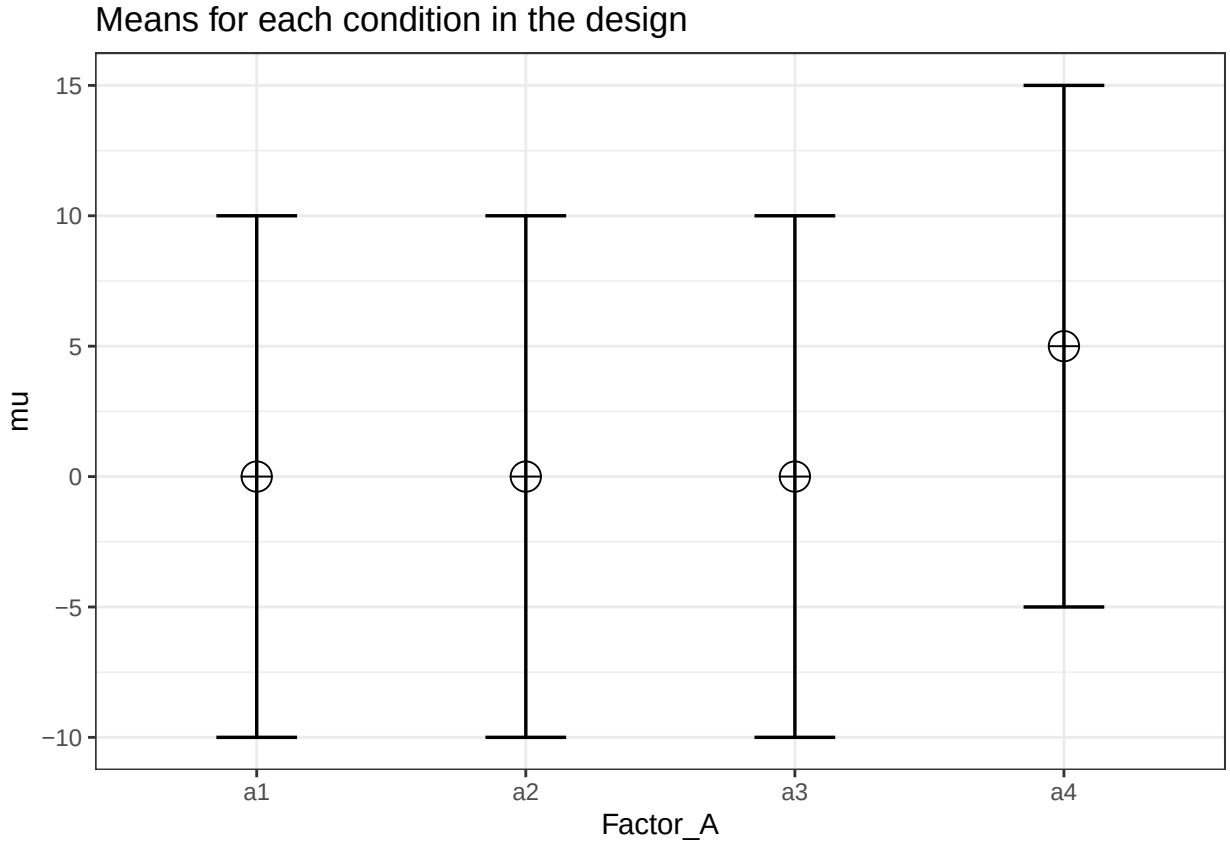
```
power_oneway_within(design_result)$lambda
```

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

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

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

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




```
power_oneway_within(design_result)$power
```

```
## [1] 0.7940126
```

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

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

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

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

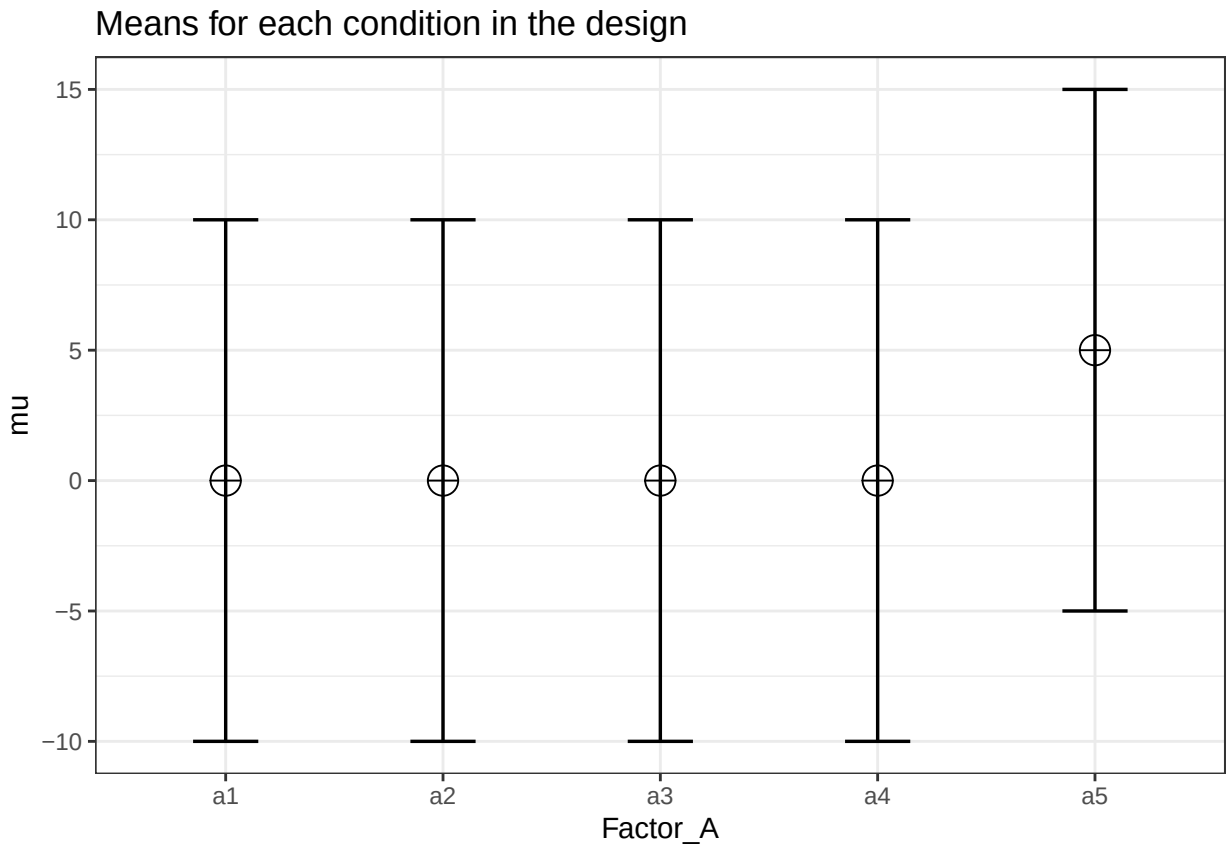
```
power_oneway_within(design_result)$lambda
```

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

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

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

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




```
power_oneway_within(design_result)$power
```

```
## [1] 0.7838682
```

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

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

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

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

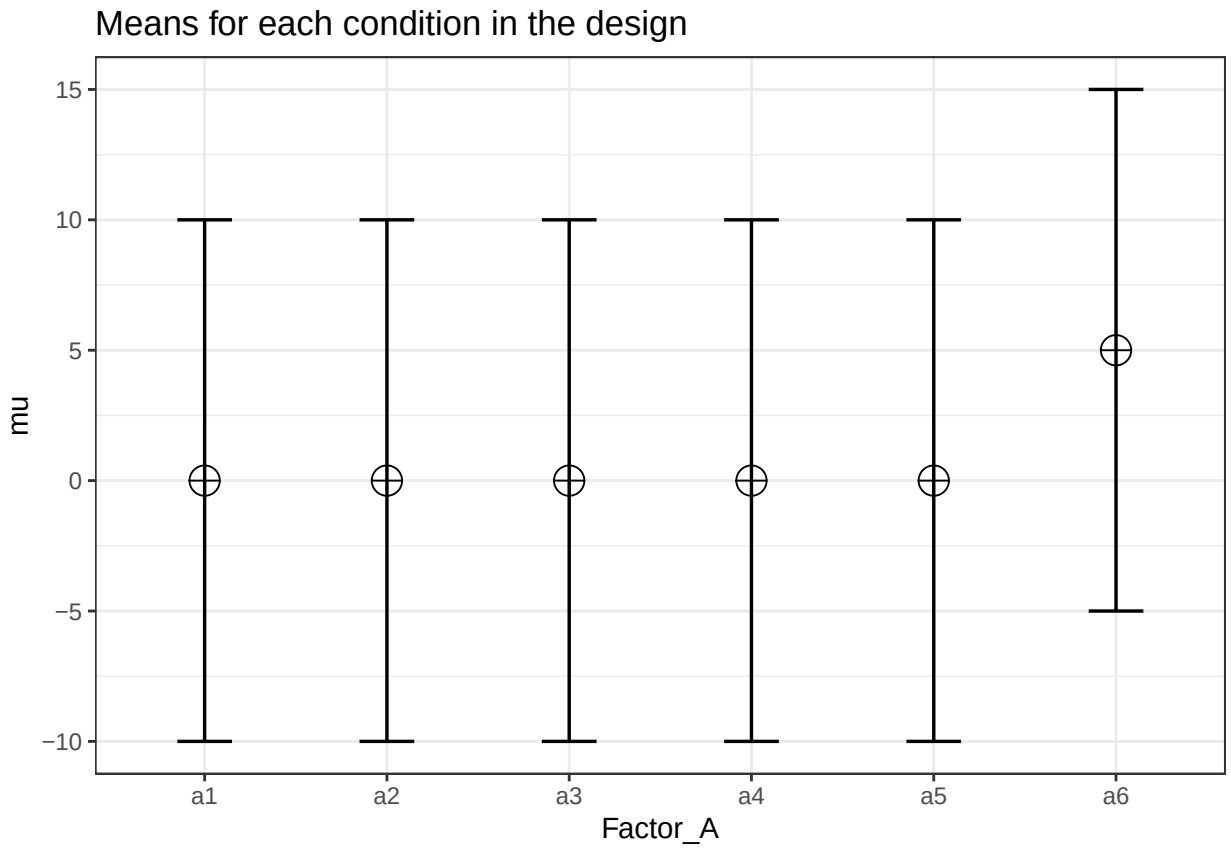
```
power_oneway_within(design_result)$lambda
```

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

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

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

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




```
power_oneway_within(design_result)$power
```

```
## [1] 0.7699592
```

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

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

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

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

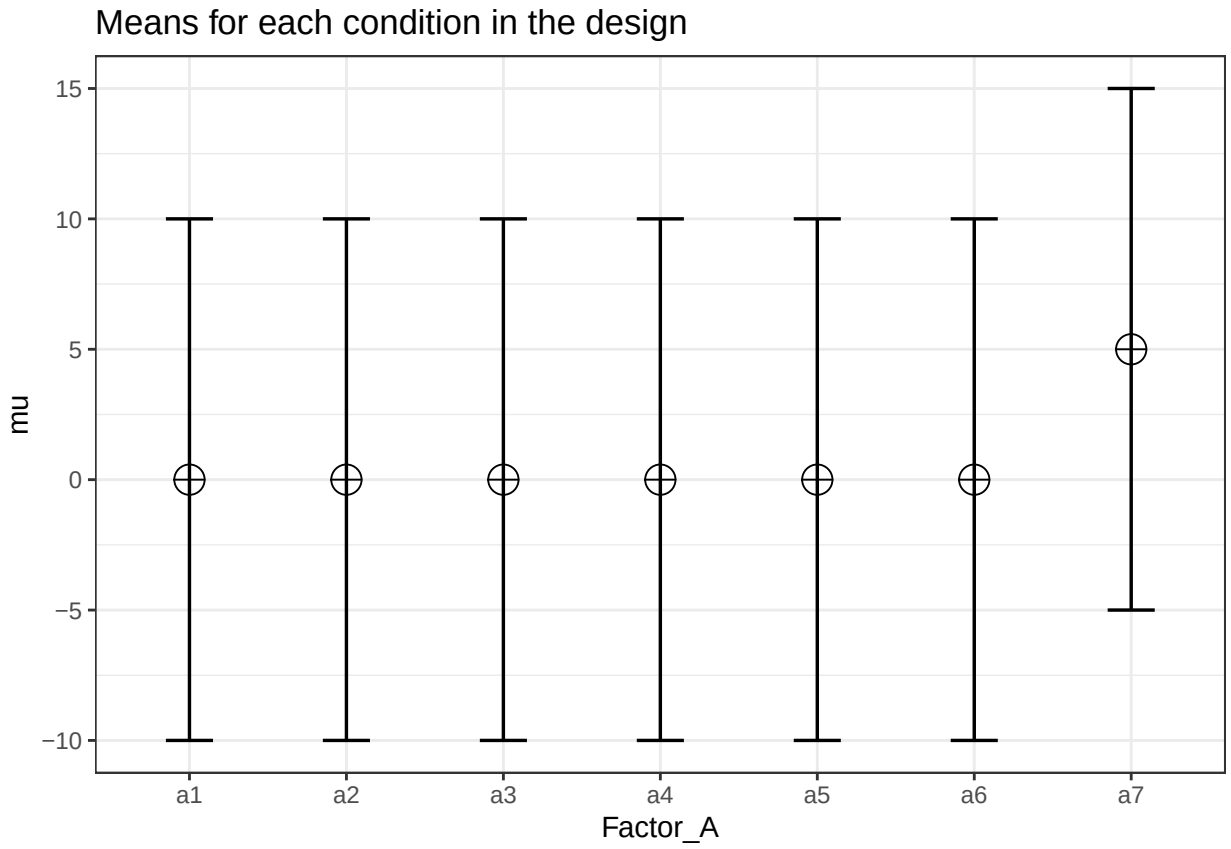
```r
power_oneway_within(design_result)$lambda
```

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

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

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

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



```
power_oneway_within(design_result)$power
```

```
## [1] 0.754601
```

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

```
## [1] 0.1749636
```

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

```
## [1] 0.2718301
```

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

```
## [1] 12.85714
```

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

```
## [1] 2.151016
```

This set of designs where we increase the number of conditions demonstrates a common pattern where the power initially increases, but then starts to decrease. Again, the exact pattern (and when the power starts to decrease) depends on the effect size and sample size. Note also that the effect size (Cohen's f) decreases as we add conditions, but the increased sample size compensates for this when calculating power. When using power analysis software such as GPower, 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).

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

Chapter 18

Setting the Correlation Matrix

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

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

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

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

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

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

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

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

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

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

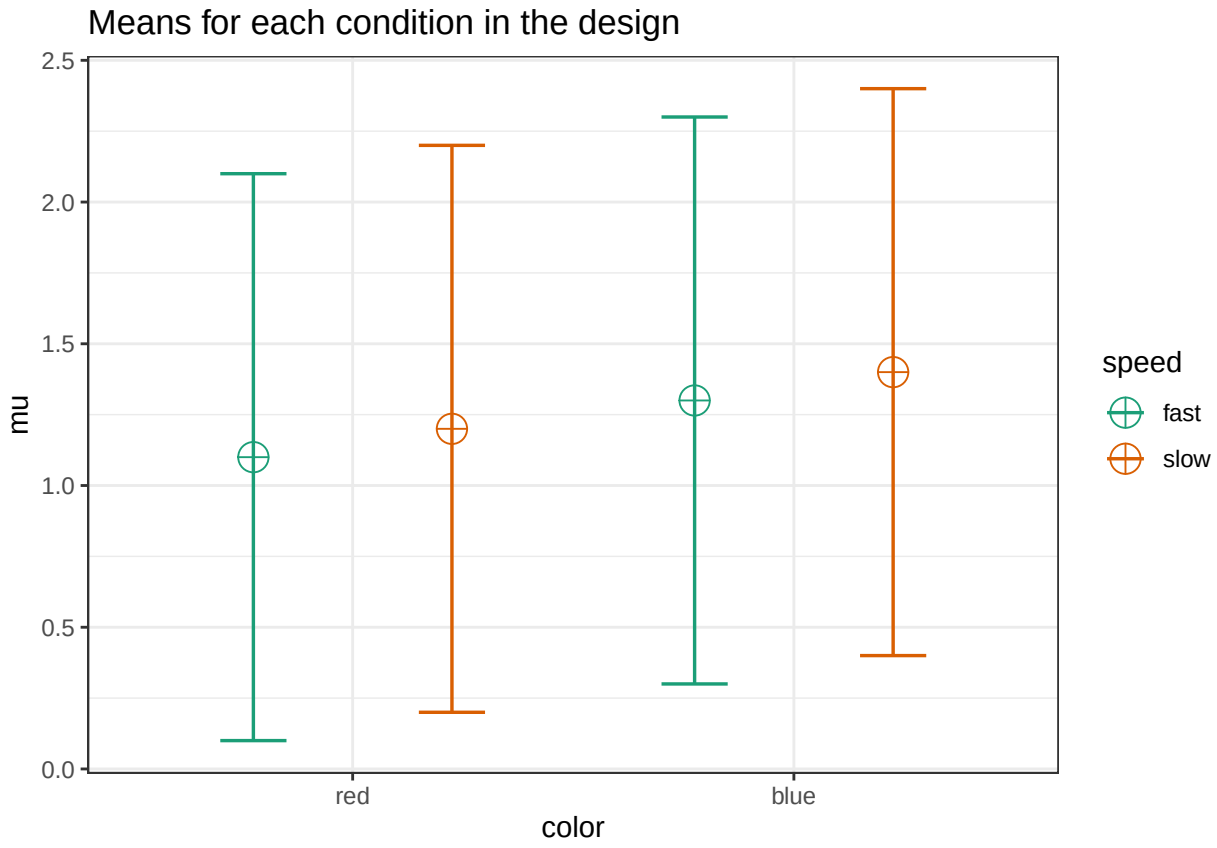
a1_b1 a1_b2 a2_b1 a2_b2

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 0.9 1.00

The diagonal is generated dynamically (based on the standard deviation).

We would enter this correlation matrix 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"))
```



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

```
design_result$cor_mat
```

```
##           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
```

Chapter 19

Validation of effect size estimates for One-Way ANOVA

Using the formulas below, we can calculate the means for between designs with one factor (One-Way ANOVA). Using the formula also used in Albers & Lakens (2018), we can determine the means that should yield a specified effect sizes (expressed in Cohen's f).

```
mu_from_ES <- function(K, ES){ # provides the vector of population means for a given population ES and
  f2 <- ES / (1 - ES)
  if (K == 2) {
  a <- sqrt(f2)
  muvec <- c(-a, a)
  }
  if (K == 3) {
  a <- sqrt(3 * f2 / 2)
  muvec <- c(-a, 0, a)
  }
  if (K == 4) {
  a <- sqrt(f2)
  muvec <- c(-a,-a, a, a)
  } # note: function gives error when K not 2,3,4. But we don't need other K.
  return(muvec)
}
```

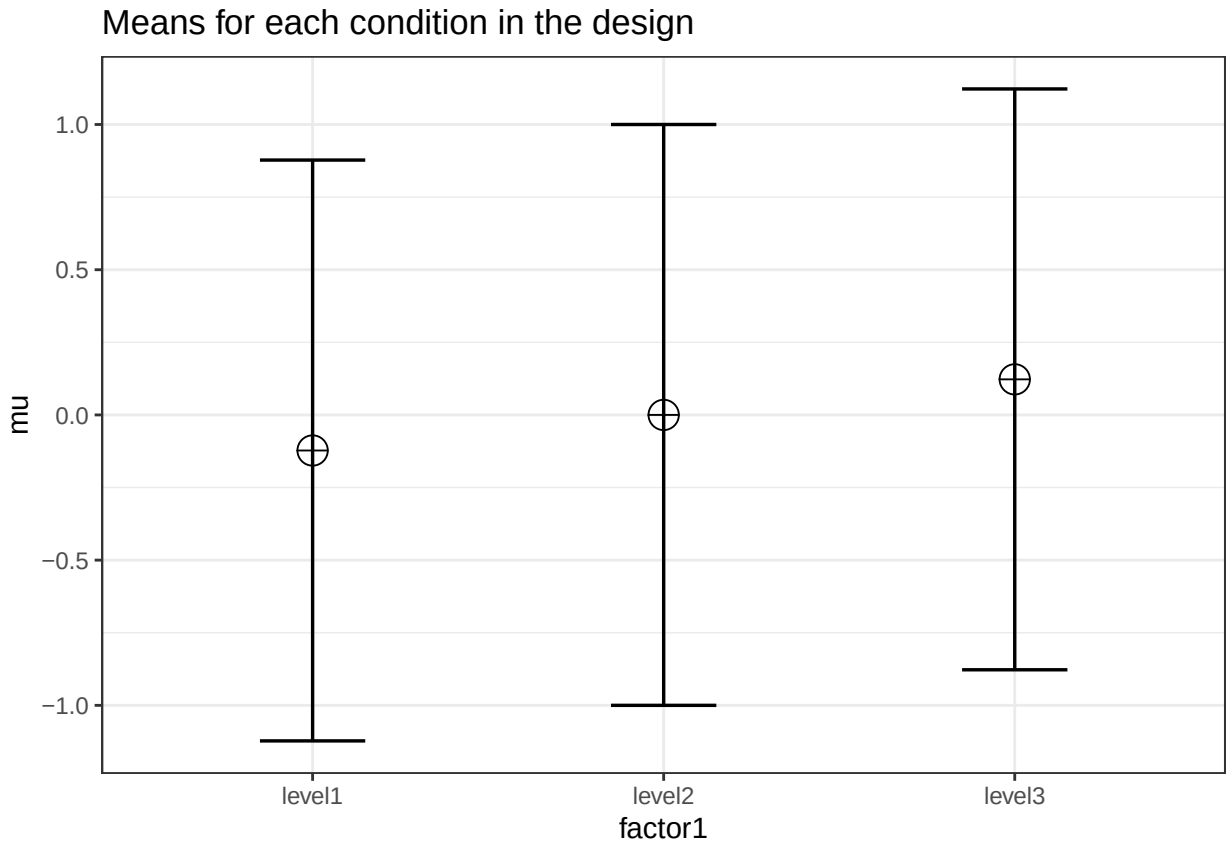
Eta-squared (idential to partial eta-squared for One-Way ANOVA's) has benchmarks of .0099, .0588, and .1379 for small, medium, and large effect sizes (cohen, 1988).

19.1 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"))
```




```
ANOVA_power(design_result, nsims = nsims)
```

```
## Power and Effect sizes for ANOVA tests
##               power effect_size
## anova_factor1   100     0.01004
##
## Power and Effect sizes for contrasts
##                                 power effect_size
## p_factor1_level1_factor1_level2   100      0.1218
```

```
## p_factor1_level1_factor1_level3   100       0.2452
## p_factor1_level2_factor1_level3   100       0.1234
```

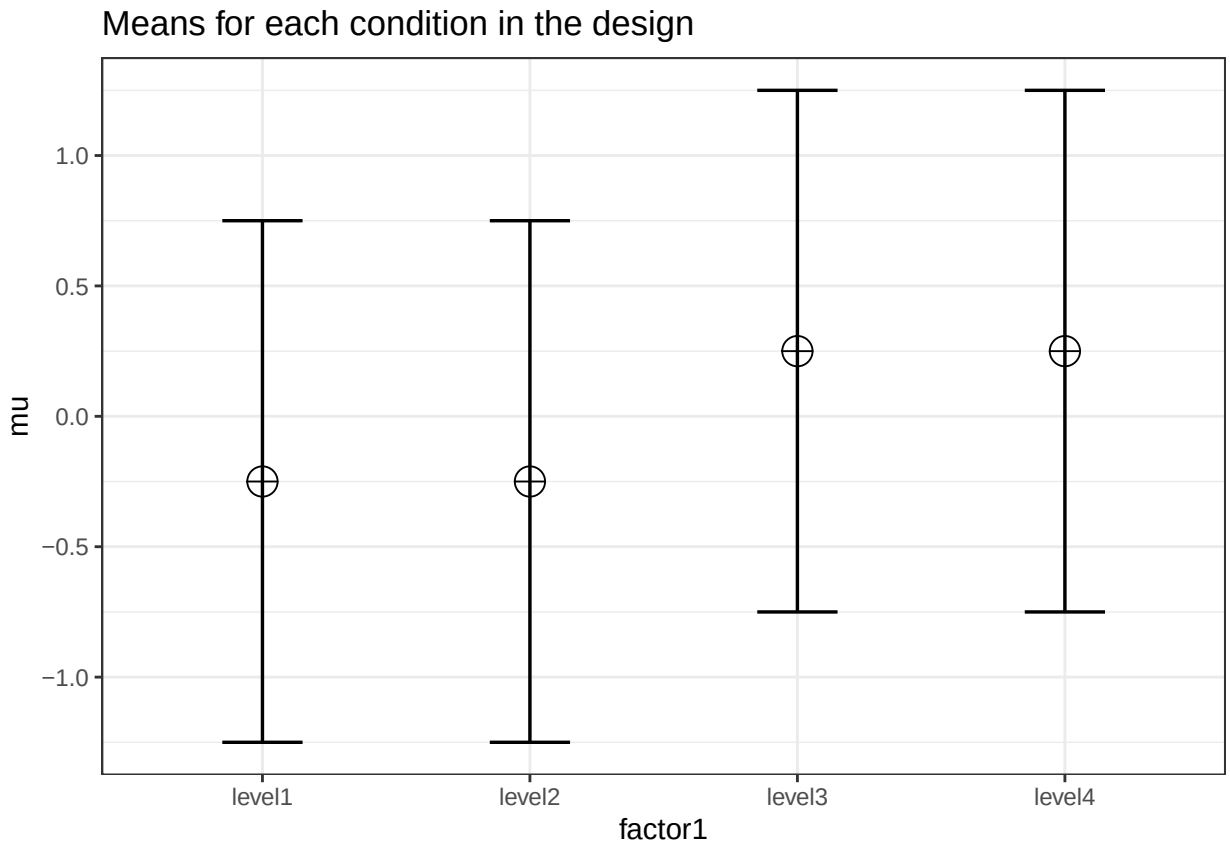
The resulting effect size estimate from the simulation is very close to 0.0099

19.2 Four conditions, medium effect size

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

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

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



```
ANOVA_power(design_result, nsims = nsims)
```

```
## Power and Effect sizes for ANOVA tests
##               power effect_size
## anova_factor1   100     0.05903
##
## Power and Effect sizes for contrasts
##                                  power effect_size
## p_factor1_level1_factor1_level2      8   -0.001316
## p_factor1_level1_factor1_level3    100    0.497462
## p_factor1_level1_factor1_level4    100    0.501763
## p_factor1_level2_factor1_level3    100    0.498742
## p_factor1_level2_factor1_level4    100    0.503064
## p_factor1_level3_factor1_level4      6    0.004152
```

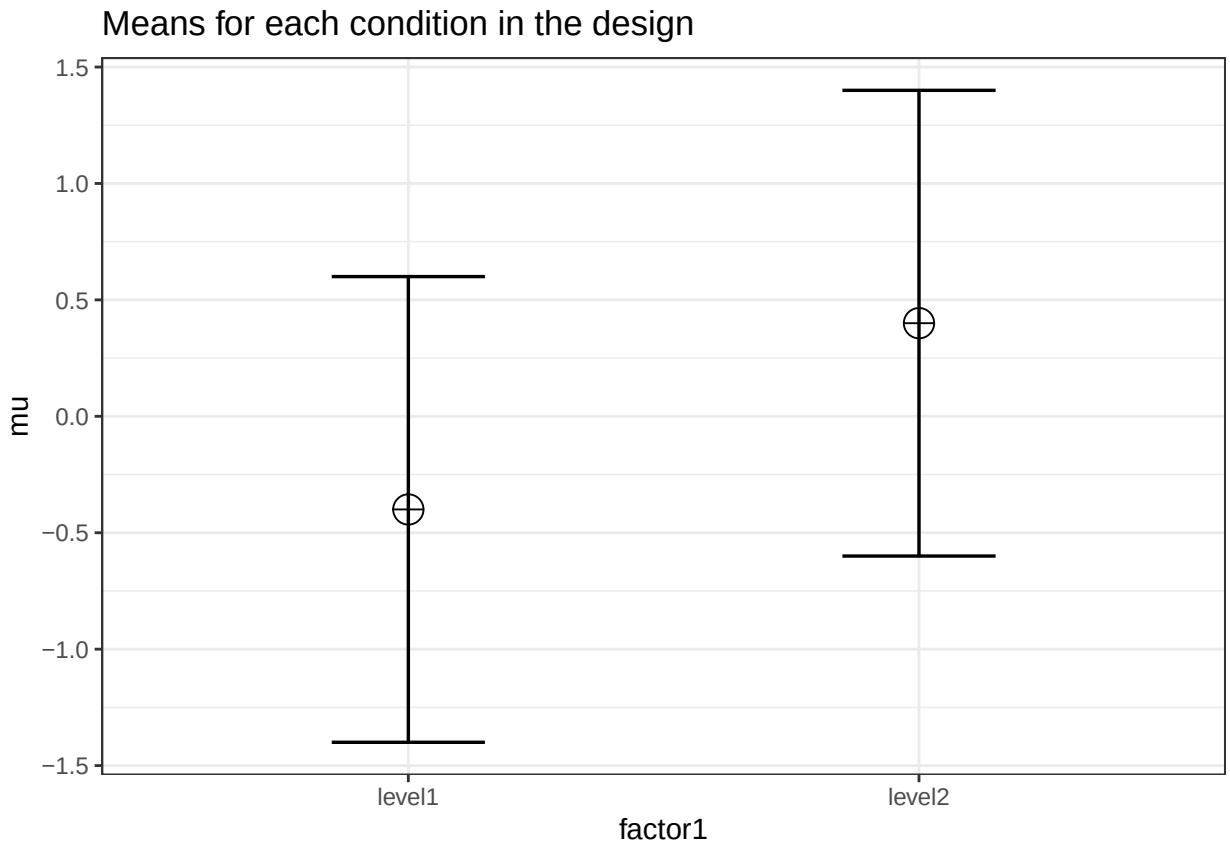
Results are very close to 0.588.

19.3 Two conditions, large effect size

We can simulate a one-factor 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"))
```




```
ANOVA_power(design_result, nsims = nsims)
```

```
## Power and Effect sizes for ANOVA tests
##               power effect_size
## anova_factor1   100      0.1378
##
## Power and Effect sizes for contrasts
##                                  power effect_size
## p_factor1_level1_factor1_level2   100      0.7993
```

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