RP Power Analysis Framework

Rethink Priorities Survey Team

11/8/2021

Intro and goals

The purpose of this document is to propose the fundamentals for a workflow for power analyses. Although the initial purpose here was to perform power analyses for Bayesian analyses, the overarching framework is applicable to any analytic approach. Especially given that the Bayesian power analyses can be very time consuming (at the moment), in some cases it may make sense to run frequentist approaches: in many cases the estimates from Bayesian and frequentist approaches will tend to converge, and I suspect general estimates from frequentist approaches would be quite similar to those of Bayesian approaches, if the goal of the analysis (what you want to make an inference about) is the same.

By using the general approach laid out here, I think we could develop a library of common analyses and inference goals that we can add to over time. Helpful additions to the toolkit would be both in terms of clear and flexible ways to generate hypothetical data sets, and in adding analytic designs that can be used.

Definition of power

In frequentist null hypothesis significance testing, power is typically defined as the probability that we can reject the null hypothesis, if the alternative hypothesis is indeed true (i.e., the true positive rate). For Bayesian analyses, there is some esoteric discussion about whether power is even a thing we should be discussing: there is uncertainty about the data-generating process, but only one data set. However, before we conduct an experiment, there are of course many hypothetical future data sets we might observe. We broaden the idea of power to indicate the probability that our proposed sample yields information that allows us to make some specific kind of inference about the data-generating process.

These inferences can take many forms, such as:

- probability of determining that there is a non-zero difference between two conditions
- probability of detecting an effect in a regression model of a specific magnitude
- probability of detecting some 'smallest effect size of interest'
- probability of achieving a desired level of precision around a particular parameter estimate

General framework

Power analysis proceeds in 4 primary steps:

1. Generate a large number of data sets for the analysis. The data sets generated should be specific to the goal of the study, e.g., if one wishes to know power to detect an effect size of .2 SD, then one should generate data in line with that effect size (for the true positive and false negative rates), and for a null hypothesis (for the false positive and true negative rates)

2. Run the proposed analysis over the many data sets as efficiently as possible. Each analysis' output should be kept as flexible as possible while not using too much memory space, so that we can assess multiple inference goals on the same output (e.g., it is possible to get the full posterior distribution for an analysis so it can be assessed and summarised in many ways)
3. Summarize the output returned in Step 2 to determine the proportion of times different decision criteria are met. This stage can show the likelihood of achieving a range of goals given various sample sizes, and display rates of misleading evidence. In frequentist analyses, it is possible that steps 2 and 3 occur together (e.g., the p-value is returned along with the other output)
4. Assess the output and determine whether the proposed analyses and inference goals are realistic and likely to yield informative results. If not, one may need to think of alternative analyses, sample sizes, inference goals, or ways of generating the data that might be more powerful, and return to step 1.

Upfront consideration of limitations

I am always somewhat hesitant about power analyses because ultimately, if we are simulating data using quite simplistic rules, then we know that this does not completely represent what will come out in the real data. There are almost always surprises in real data sets that might make the ultimate analyses we wish to perform different from what we initially set out to investigate. I think that it is okay that our power analyses might not always be perfect or something of a blunt instrument, so long as we recognise they are not guarantees but more a way of determining the basic plausibility of achieving certain goals.

Likewise when we see real data and real interesting patterns emerge, we are likely to go further in modeling and investigating these patterns than some of the more simple analyses and comparisons we conduct in the initial power analysis. I suspect power analyses are not so good for determining all the intricate in depth things we might plumb in a dataset, but more for assessing the tractability of a broad inference goal.

Finally, it might be considered how far we wish to go in the initial data simulation step to think about all sorts of hypothetical data sets. As models become more complex, the number of different parameters that might vary - with possible effects on power - starts to balloon. E.g., even in a simple repeated measures example, do we wish to vary not only the effect size but all sorts of different correlations from pre- to post-treatment within subjects? If we are simulating ordinal data, then power might change depending on how we initially suggest binning the outcomes, but there are infinitely many ways we might think the data might look... Discussion and consideration of how far we should go with these things is welcome and could be useful!

Step 1: Generate a large number of data sets for the analysis

Confirm that we can generate the basic data we want

Because ultimately we might be generating some rather large data files, we increase the memory limit alotted to R first:

```
memory.limit(100000)
```

```
## [1] 1e+05
```

Now we want to generate a hypothetical data set. I will use a repeated measures ANOVA type design, with Group Control and Group Treatment measured at Time Pre and Time Post. I will also consider the data as normally distributed data, and as a Likert-type binned response.

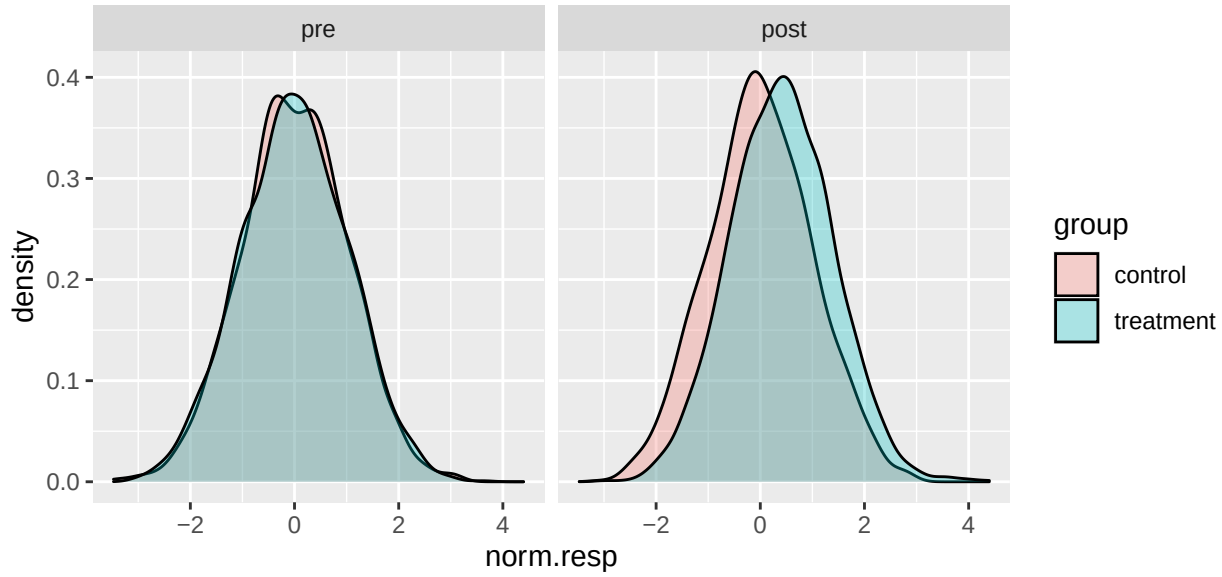
I used some of Willem's code ideas for generating the repeated measures data. **I think there are likely to be other useful packages for generating simulated data for more complex designs, or more simply than this for simple designs, that we could look into** The data-making functions below are

qutie specific to this use case - the point for this step is simply to make a function for making a single data set that we can then run many iterations of.

```
library(tidyverse)
library(MASS)
```

```
data.maker.rmanova <- function(effect.size, sim = 1, max.n, base.mean = 0,
                               base.sd = 1, r.within = .5, intended.effect.size) {

  # Parameters
  treatpre <- base.mean
  treatpost <- base.mean + effect.size
  sd <- base.sd
  n <- max.n
  r <- r.within

  # Prepare parameters
  treat.mus <- c(treatpre, treatpost)
  treat.sigma <- matrix(ncol = 2, nrow = 2,
                        c(sd^2, (sd^2 * r),
                          (sd^2 * r), sd^2))

  # Simulate once with empirical = TRUE
  treat.samples <- mvrnorm(n, mu = treat.mus, Sigma = treat.sigma)

  # Prepare data
  colnames(treat.samples) <- c("treatpre", "treatpost")
  treat.samples <- as_tibble(treat.samples)
  treat.samples$sample.size <- seq(from = 1, to = max.n)

  treat.long <-
    treat.samples %>%
    pivot_longer(cols = c('treatpre', 'treatpost'),
                 names_to = "identifier", values_to = "norm.resp") %>%
    mutate(group = case_when(grepl("treat", identifier) == TRUE ~ "treatment",
                             grepl('control', identifier) == TRUE ~ "control"),
           time = case_when(grepl("pre", identifier) == TRUE ~ "pre",
                            grepl('post', identifier) == TRUE ~ "post"),
           time = factor(time, levels = c("pre", "post"), labels = c("pre", "post")),
           sim = sim,
           ppn = sample.size)

  # Generate for control group

  # Parameters
  controlpre <- base.mean
  controlpost <- base.mean

  # Prepare parameters
  control.mus <- c(controlpre, controlpost)
  control.sigma <- matrix(ncol = 2, nrow = 2,
                          c(sd^2, (sd^2 * r),
                            (sd^2 * r), sd^2))
```

```r
  # Before using the function as is, simulate once with empirical = TRUE
  control.samples <- mvrnorm(n, mu = control.mus, Sigma = control.sigma)

  # Prepare data
  colnames(control.samples) <- c("controlpre", "controlpost")
  control.samples <- as_tibble(control.samples)
  control.samples$sample.size <- seq(from = 1, to = max.n)

  control.long <-
    control.samples %>%
    pivot_longer(cols = c('controlpre', 'controlpost'),
                 names_to = "identifier", values_to = "norm.resp") %>%
    mutate(group = case_when(grepl("treat", identifier) == TRUE ~ "treatment",
                             grepl('control', identifier) == TRUE ~ "control"),
           time = case_when(grepl("pre", identifier) == TRUE ~ "pre",
                            grepl('post', identifier) == TRUE ~ "post"),
           time = factor(time, levels = c("pre", "post"), labels = c("pre", "post")),
           sim = sim,
           ppn = sample.size + max.n)

  full.data <- bind_rows(control.long, treat.long)

  full.data$effect.size <- effect.size
  full.data$intended.effect.size <- intended.effect.size

  return(full.data)
}

# test that the function works to make one data set before making many!
test.data <- data.maker.rmanova(effect.size = .4, sim = 1,
                                max.n = 2000, base.mean = 0,
                                base.sd = 1, r.within = .6,
                                intended.effect.size = .4)

ggplot(data = test.data) +
  geom_density(aes(x = norm.resp, fill = group), alpha = .3) +
  facet_wrap(~time) +
  theme(
    aspect.ratio = 1
  )
```



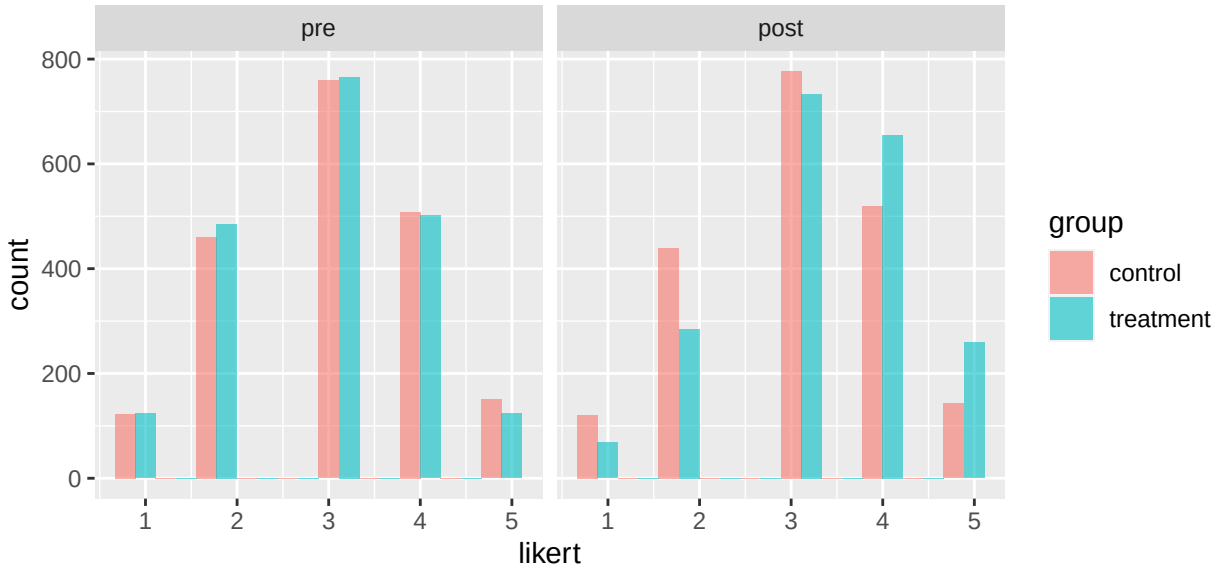
We can see from the plot that the function appears to be working. Note that, as Willem does in his example, for a full check you would also want to change the mvnorm code to include 'empirical = TRUE' so that it simulates exactly the right group difference, and check that this occurs in your data set. There might be all sorts of other diagnostics or plots we might check with other types of data that we generate.

The function below is essentially the same as above, but it bins the normally distributed data into 5 categories, with the numbers associated with each bin borrowed from the default used for ordinal binning in 'declare design'.

```
data.maker.rmanova.ordinal <- function(effect.size, sim = 1, max.n,
                                       base.mean = 0, base.sd = 1,
                                       r.within = .6, intended.effect.size) {

  # Parameters
  treatpre <- base.mean
  treatpost <- base.mean + effect.size
  sd <- base.sd
  n <- max.n
  r <- r.within

  # Prepare parameters
  treat.mus <- c(treatpre, treatpost)
  treat.sigma <- matrix(ncol = 2, nrow = 2,
                        c(sd^2, (sd^2 * r),
                          (sd^2 * r), sd^2))

  # Before using the function as is, simulate once with empirical = TRUE
```

```r
treat.samples <- mvrnorm(n, mu = treat.mus, Sigma = treat.sigma)

# Prepare data
colnames(treat.samples) <- c("treatpre", "treatpost")
treat.samples <- as_tibble(treat.samples)
treat.samples$sample.size <- seq(from = 1, to = max.n)

treat.long <-
  treat.samples %>%
  pivot_longer(cols = c('treatpre', 'treatpost'),
               names_to = "identifier", values_to = "norm.resp") %>%
  mutate(group = case_when(grepl("treat", identifier) == TRUE ~ "treatment",
                           grepl('control', identifier) == TRUE ~ "control"),
         time = case_when(grepl("pre", identifier) == TRUE ~ "pre",
                          grepl('post', identifier) == TRUE ~ "post"),
         time = factor(time, levels = c("pre", "post"), labels = c("pre", "post")),
         likert = case_when(norm.resp < -1.5 ~ 1,
                            norm.resp < -.5 ~ 2,
                            norm.resp < .5 ~ 3,
                            norm.resp < 1.5 ~ 4,
                            norm.resp >= 1.5 ~ 5),
         sim = sim,
         ppn = sample.size)

# Generate for control group

# Parameters
controlpre <- base.mean
controlpost <- base.mean

# Prepare parameters
control.mus <- c(controlpre, controlpost)
control.sigma <- matrix(ncol = 2, nrow = 2,
                        c(sd^2, (sd^2 * r),
                          (sd^2 * r), sd^2))

# Before using the function as is, simulate once with empirical = TRUE
control.samples <- mvrnorm(n, mu = control.mus, Sigma = control.sigma)

# Prepare data
colnames(control.samples) <- c("controlpre", "controlpost")
control.samples <- as_tibble(control.samples)
control.samples$sample.size <- seq(from = 1, to = max.n)

control.long <-
  control.samples %>%
  pivot_longer(cols = c('controlpre', 'controlpost'),
               names_to = "identifier", values_to = "norm.resp") %>%
  mutate(group = case_when(grepl("treat", identifier) == TRUE ~ "treatment",
                           grepl('control', identifier) == TRUE ~ "control"),
         time = case_when(grepl("pre", identifier) == TRUE ~ "pre",
                          grepl('post', identifier) == TRUE ~ "post"),
         time = factor(time, levels = c("pre", "post"), labels = c("pre", "post")),
```

```
            likert = case_when(norm.resp < -1.5 ~ 1,
                               norm.resp < -.5 ~ 2,
                               norm.resp < .5 ~ 3,
                               norm.resp < 1.5 ~ 4,
                               norm.resp >= 1.5 ~ 5),
            sim = sim,
            ppn = sample.size + max.n)

  full.data <- bind_rows(control.long, treat.long)

  full.data$effect.size <- effect.size
  full.data$intended.effect.size <- intended.effect.size

  return(full.data)
}

test.data.ordinal <- data.maker.rmanova.ordinal(effect.size = .4,
                                                sim = 1, max.n = 2000, base.mean = 0,
                                                base.sd = 1, r.within = .6,
                                                intended.effect.size = .4)

ggplot(data = test.data.ordinal) +
  geom_histogram(aes(x = likert, fill = group), alpha = .6,
                 position = position_dodge(), bins = 10) +
  facet_wrap(~time) +
  theme(
    aspect.ratio = 1
  )
```



Again, it should be possible to see that the function has worked, with the post-treatment bins indicating higher ratings among the treatment than the control group, whereas they are essentilly the same before treatment.

Efficiently generate many data sets

Now we just need to run this function many times over. This could be done using loops, but a very useful set of R functions in the tidyverse is the purrr package of map functions. Even better, a package called furrr is available to run such map functions in parallel to further reduce time. This doesn't matter so much here because this will be quite quick anyway, but is important when we run the analyses over the many data sets. For furrr to do this, we need to tell it to plan for 'multisession' and it is also helpful to give it a seed number:

```
library(furrr)

plan(multisession)
options <- furrr_options(seed = 48238)
```

Consider what you want your data sets to represent. As an example below, we can generate a null data set (difference between groups = 0) and various effect size options (differences of .15, .30, .45). However, we could do all sorts of things, including many other effect sizes, a distribution of effect sizes around a specific number, just a single effect size, or change other aspects of the data like the correlation between the repeated observations.

```
# set up effect sizes to pass to the map function
tibble.00 <- tibble(effect.size = rep(0, 500),
```

```
                    sim = seq(1:500))
tibble.15 <- tibble(effect.size = rep(.15, 500),
                    sim = seq(1:500))
tibble.30 <- tibble(effect.size = rep(.3, 500),
                    sim = seq(1:500))
tibble.45 <- tibble(effect.size = rep(.45, 500),
                    sim = seq(1:500))

t1 <- Sys.time()
sim.data.00 <- future_map2_dfr(.x = tibble.00$effect.size,
                               .y = tibble.00$sim, .f = data.maker.rmanova, max.n = 2000,
                               base.mean = 0, base.sd = 1,
                               r.within = .5, intended.effect.size = 0, .options = options)
t2 <- Sys.time()
t2 - t1
```

```
## Time difference of 4.696517 secs
```

```
t1 <- Sys.time()
sim.data.00 <- future_map2_dfr(.x = tibble.00$effect.size,
                               .y = tibble.00$sim, .f = data.maker.rmanova, max.n = 2000,
                               base.mean = 0, base.sd = 1,
                               r.within = .5, intended.effect.size = 0, .options = options)
t2 <- Sys.time()
t2 - t1
```

```
## Time difference of 3.287238 secs
```

```
sim.data.15 <- future_map2_dfr(.x = tibble.15$effect.size,
                               .y = tibble.15$sim, .f = data.maker.rmanova, max.n = 2000,
                               base.mean = 0, base.sd = 1,
                               r.within = .5, intended.effect.size = .15, .options = options)

sim.data.30 <- future_map2_dfr(.x = tibble.30$effect.size,
                               .y = tibble.30$sim, .f = data.maker.rmanova,max.n = 2000,
                               base.mean = 0, base.sd = 1,
                               r.within = .5, intended.effect.size = .3, .options = options)

sim.data.45 <- future_map2_dfr(.x = tibble.45$effect.size,
                               .y = tibble.45$sim, .f = data.maker.rmanova, max.n = 2000,
                               base.mean = 0, base.sd = 1,
                               r.within = .5, intended.effect.size = .45, .options = options)

sim.data.00 <- sim.data.00 %>% group_by(effect.size, sim) %>% group_split()
sim.data.15 <- sim.data.15 %>% group_by(effect.size, sim) %>% group_split()
sim.data.30 <- sim.data.30 %>% group_by(effect.size, sim) %>% group_split()
sim.data.45 <- sim.data.45 %>% group_by(effect.size, sim) %>% group_split()
```

Now, we have several hundred simulated data sets representing our hypothetical outcome data, and can perform analyses on them:

```r
head(sim.data.45[[1]])
```

```
## # A tibble: 6 x 9
##   sample.size identifier  norm.resp group   time    sim   ppn effect.size
##         <int> <chr>           <dbl> <chr>   <fct> <int> <dbl>       <dbl>
## 1           1 controlpre     0.652  control pre       1  2001        0.45
## 2           1 controlpost   -1.14   control post      1  2001        0.45
## 3           2 controlpre    -1.80   control pre       1  2002        0.45
## 4           2 controlpost   -2.61   control post      1  2002        0.45
## 5           3 controlpre     0.744  control pre       1  2003        0.45
## 6           3 controlpost    0.0285 control post      1  2003        0.45
## # ... with 1 more variable: intended.effect.size <dbl>
```

Limitation: For this analysis with repeated measures, the time for the actual analyses of these data sets is too long and below I will just show examples of analysing the first 100 data sets of each effect size (which already still took a very long time). This might be improved with cloud computing. Other Bayesian designs actually can run quickly enough even over 1000 data sets, and this is also likely feasible for many frequentist models to do even more easily without any issue or cloud computing needed

Step 2: Run the proposed analysis over the many data sets as efficiently as possible