Power calculations

Andrew Roth

2024-05-21

Introduction

We have seen how to load our data and plot it in R. The next thing to learn is how to do statistics in R. Since R started life as a programming language for statistics, it is not surprising it has a lot of statistical capabilities.

Setup

The first thing we will look at is how to do power calculations. The are a few different packages for power calculations in R. For the purposes of this tutorial I will work with the pwrss library. You will need to install it using the `install.packages` function before proceeding.

Assuming you have installed the package, let's load it.

```
library(pwrss)
```

```
##
## Attaching package: 'pwrss'

## The following object is masked from 'package:stats':
##
##     power.t.test
```

Let's assume we plan to use a t-test to compare two group. We can use the `pwrss.t.2means` function to do our power calculation. Recall we saw how to do this in class using an online tool to compare two independent samples.

Sample size calculation

First, let's find out what sample size we need. Let's assume a power of 0.8, effect size of 1, standard deviation of 1 for both groups, mean of 10 for the control group.

To do the calculation we specify the following arguments to `pwrss.t.2means`: - `mu1` - The mean of the control group. - `mu2` - The mean of the test group. Here this mu1 + effect size. - `sd1` - Standard deviation of the control group. By default this also the standard deviation of the test group. - `alpha` - The singificance level. - `power` - Power to reject the null hypothesis.

```
pwrss.t.2means(mu1=10, mu2=11, sd1=1, alpha=0.05, power=0.8)
```

```
##  Difference between Two means
##  (Independent Samples t Test)
##  H0: mu1 = mu2
##  HA: mu1 != mu2
##  ------------------------------
##   Statistical power = 0.8
##   n1 = 17
##   n2 = 17
##  ------------------------------
##  Alternative = "not equal"
##  Degrees of freedom = 32
##  Non-centrality parameter = -2.915
##  Type I error rate = 0.05
##  Type II error rate = 0.2
```

We see that we will need a sample size of 17 in each group. By default the `pwrss.t.2means` assumes the groups are of the same size. If we want to consider an unbalanced design we can specify the `kappa` argument which is the ratio of control to test group sizes. Suppose we wanted to have twice as many observations in the test group, we could use the following.

```
pwrss.t.2means(mu1=10, mu2=11, sd1=1, alpha=0.05, power=0.8, kappa=0.5)
```

```
##  Difference between Two means
##  (Independent Samples t Test)
##  H0: mu1 = mu2
##  HA: mu1 != mu2
##  ------------------------------
##   Statistical power = 0.8
##   n1 = 13
##   n2 = 25
##  ------------------------------
##  Alternative = "not equal"
##  Degrees of freedom = 36
##  Non-centrality parameter = -2.924
##  Type I error rate = 0.05
##  Type II error rate = 0.2
```

Here we would a sample size of 13 for the control and 25 for the test group. Note it this case our overall n has gone from 34 with the balanced design to 38 for the unbalanced design.

We can also assume a different standard deviation for the test group by passing the `sd2` argument. In the next example I will assume we use the same n for both groups, but the test group has a standard deviation of 2.

```
pwrss.t.2means(mu1=10, mu2=11, sd1=1, sd2=2, alpha=0.05, power=0.8)
```

```
##  Difference between Two means
##  (Independent Samples t Test)
##  H0: mu1 = mu2
##  HA: mu1 != mu2
##  ------------------------------
##   Statistical power = 0.8
##   n1 = 41
##   n2 = 41
##  ------------------------------
##  Alternative = "not equal"
##  Degrees of freedom = 80
##  Non-centrality parameter = -2.864
##  Type I error rate = 0.05
##  Type II error rate = 0.2
```

Unsurprisingly if we assume more variability in the test group our sample size needs to be larger.

Retrospective power calculation

We can also figure out our power given a fixed sample size. To do this we specify the `n2` argument and omit the `power` argument. The `n2` argument is interpreted as the n of the test group which by default is assumed to be the same as the control group. Let's try with a test group size of 10, so overall n=20.

```
pwrss.t.2means(mu1=10, mu2=11, sd1=1, alpha=0.05, n2=10)
```

```
##  Difference between Two means
##  (Independent Samples t Test)
##  H0: mu1 = mu2
##  HA: mu1 != mu2
##  ------------------------------
##   Statistical power = 0.562
##   n1 = 10
##   n2 = 10
##  ------------------------------
##  Alternative = "not equal"
##  Degrees of freedom = 18
##  Non-centrality parameter = -2.236
##  Type I error rate = 0.05
##  Type II error rate = 0.438
```

We that our power is quite low with n=20.

Exploring power

So far there is not much of an advantage over using the online system. Let's see an example that takes advantage of the power of R. We are going to plot out how our power varies with the number of samples.

We will use ggplot as usual so let's load it up.

```
library(ggplot2)
```

Now we will use some new syntax. I want to get all values between 2 and 20, and I will pass them as an argument eventually. R makes it easy to get a vector like this as follows.

```
2:20
```

```
##  [1]  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20
```

The `:` means give me all vector with values starting at the number on the left up to including the values on the right.

This is helpful because the `pwrss.t.2means` allows us to pass multiple values for `n2`. Let's take a look.

```
pwrss.t.2means(mu1=10, mu2=11, sd1=1, alpha=0.05, n2=2:20)
```

```
##  Difference between Two means
##  (Independent Samples t Test)
##  H0: mu1 = mu2
##  HA: mu1 != mu2
##  ------------------------------
##   Statistical power = 0.095 0.159 0.223 0.286 0.347 0.406 0.461 0.513 0.562 0.607 0.649 0.687 0.721
##   n1 = 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20
##   n2 = 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20
##  ------------------------------
##  Alternative = "not equal"
##  Degrees of freedom = 2 4 6 8 10 12 14 16 18 20 22 24 26 28 30 32 34 36 38
##  Non-centrality parameter = -1 -1.225 -1.414 -1.581 -1.732 -1.871 -2 -2.121 -2.236 -2.345 -2.449 -2.
##  Type I error rate = 0.05
##  Type II error rate = 0.905 0.841 0.777 0.714 0.653 0.594 0.539 0.487 0.438 0.393 0.351 0.313 0.279
```

This is a bit of a mess to look at. Before going further it is helpful to understand what `pwrss.t.2means` returns. The output of the function is a type of R object called a "list". A list is a lot like a data frame. There are named entries in the list, like columns of a data frame. Lists are a bit more flexible however since the entries do not need to be of the same length. Let's save the result of `pwrss.t.2means` and see what the named entries are.

```
x <- pwrss.t.2means(mu1=10, mu2=11, sd1=1, alpha=0.05, n2=2:20)
```

```
##  Difference between Two means
##  (Independent Samples t Test)
##  H0: mu1 = mu2
##  HA: mu1 != mu2
##  ------------------------------
##   Statistical power = 0.095 0.159 0.223 0.286 0.347 0.406 0.461 0.513 0.562 0.607 0.649 0.687 0.721
##   n1 = 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20
##   n2 = 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20
##  ------------------------------
##  Alternative = "not equal"
##  Degrees of freedom = 2 4 6 8 10 12 14 16 18 20 22 24 26 28 30 32 34 36 38
##  Non-centrality parameter = -1 -1.225 -1.414 -1.581 -1.732 -1.871 -2 -2.121 -2.236 -2.345 -2.449 -2.
##  Type I error rate = 0.05
##  Type II error rate = 0.905 0.841 0.777 0.714 0.653 0.594 0.539 0.487 0.438 0.393 0.351 0.313 0.279
```

```
names(x)
```

```
## [1] "parms" "test"  "df"    "ncp"   "power" "n"
```

In the previous code I have run `pwrss.t.2means` and saved the result in a variable called `x`. Next I see what is stored in `x` using the `names` function. We see there are six entries. Let's take a look at the one for power.

```
x$power
```

```
##  [1] 0.09520176 0.15879087 0.22318803 0.28629549 0.34735370 0.40580695
##  [7] 0.46123885 0.51336253 0.56200665 0.60709780 0.64864255 0.68671060
## [13] 0.72141975 0.75292303 0.78139779 0.80703672 0.83004056 0.85061238
## [19] 0.86895303
```

Here I have accessed the value for power the same way as I do columns of a data frame.

We are almost ready to make our plot. The last step is to create a data frame to use.

```
power_data <- data.frame(n=2*2:20, power=x$power)
head(power_data)
```

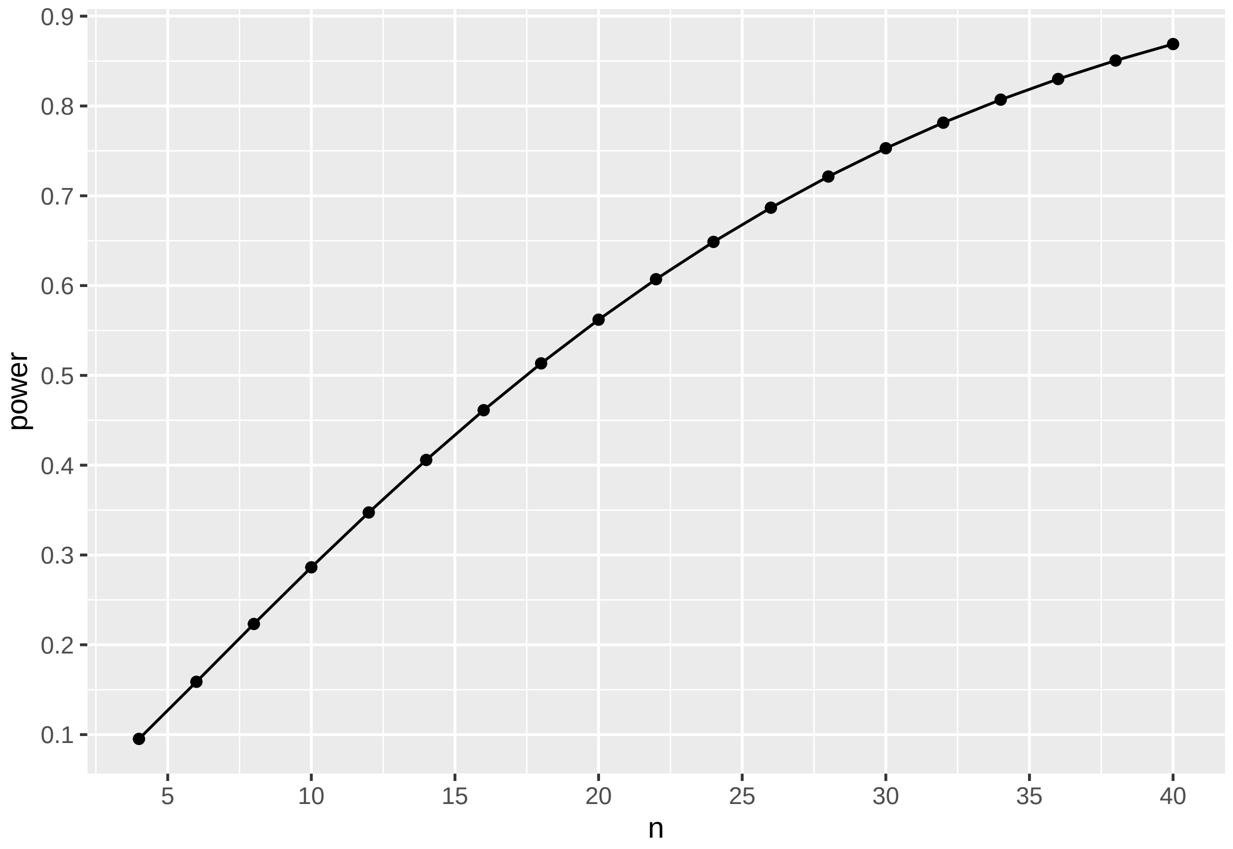
```
##    n      power
## 1  4 0.09520176
## 2  6 0.15879087
## 3  8 0.22318803
## 4 10 0.28629549
## 5 12 0.34735370
## 6 14 0.40580695
```

We now have a data frame with two columns. - `n` - The total number of samples. Note I created this using `2 * 2:20`. This first creates the vector with entries from 2 to 20, then multiplies by 2 recognising we have balanced class sizes. - `power` - The power we would have for a given `n`

All the pieces are now ready. Let's plot the result.

```r
ggplot(power_data, aes(x=n, y=power)) +
  geom_point() +
  geom_line() +
  ylim(0, 1) +
  scale_x_continuous(breaks = scales::pretty_breaks(n = 10)) +
  scale_y_continuous(breaks = scales::pretty_breaks(n = 10))
```

```
## Scale for y is already present.
## Adding another scale for y, which will replace the existing scale.
```



We can see that around a sample size of n=34, n=17 in each group, we achieve a power of 0.8. This lines up with our previous sample size calculation!

I used a few not plotting features in the previous code:

- `geom_line` - Adds a line connecting the points.
- `ylim(0, 1)` - Sets the y-axis limits to be between 0 and 1.
- `scale_x_continuous(breaks = scales::pretty_breaks(n = 10))` - Creates 10 axis label marks on the x-axis.
- `scale_y_continuous(breaks = scales::pretty_breaks(n = 10))` - Same as above.

These changes are primarily to make it easier to see when power goes past 0.8.

Conclusions

We have seen how we can do power calculations in this tutorial. We focused on the comparison of two group means. There are other power calculations that can be done, but the principles will be similar. For details read the pwrss vignette.