The powerCalc-Package: Commented Examples

Thomas Fabbro

10. May 2014

```
> library(power)
```

1 Starting with a minimal example

How sensitive is the estimation of the sample size to changes in the effect size?

In a first step we crate an object of class powPar. We investigate the power for a range of effect sizes (0.5 to 1.5) and a range of sample sizes (20 to 60).

```
> psi <- powPar(theta = seq(from = 0.5, to = 1.5, by = 0.05),
+               n = seq(from = 20, to = 60, by = 2))
```

In a second step we write a function that calculates the power. The function should take only one argument, the object of class powPar we created in the last step. Two methods called *n* and *theta* allow to extract the elements of the object individually during evaluation.

```
> powFun <- function(psi)
+ {
+   return(power.t.test(n = n(psi)/2,
+                       delta = theta(psi),
+                       sig.level = 0.05)$power)
+ }
```

In the third step we create an object of class powCalc using a function with the same name. The function *powCalc* evaluates the function we created in the last step using the powPar object of the first step.

```
> calc <- powCalc(psi, statistic = powFun)
```

The “hidden” result of the last steps is a matrix that contains the power of all combinations of *n* and *theta*. Now we need to define for which theta we would like to know the sample size with a certain power.

```
> ex <- powEx(theta = 1, power = 0.9)
```

A method called merge allows us to create one single object of class power containing the information from the calculation together with the example.

```
> pow <- merge(calc, ex)
```

Now we are done and can extract the information we need. But before we start we have a look at the relationship between the power and the sample size for the theta we are mainly interested in.

```
> inspect(pow)
```

Finally we can generate a "power plot" showing the sensitivity of the sample size estimation to the effect size, theta.

```
> plot(pow,
+      xlab = "Effect size",
+      ylab = "Total sample size",
+      label.pos = c(0,1))
```

To be able to extract the estimate from the power-object for the integration in Sweave documents there is a method called *tex*. The call

```
> tex(pow, type = "nEval")

[1] 46
```

returns the string "46" and can be used as 46.

2 Using a resampling approach

To illustrate the resampling approach we take the same example as before but we compare the two groups with a wilcoxon test. Note the difference in the function we define. Whereas the function return the power of a combination of *theta* and *n* it returns a logical, indicating if the test was statistically significant or not.

```
> powFun.resample <- function(psi)
+ {
+   x <- rnorm(n(psi)/2)
+   y <- rnorm(n(psi)/2) + theta(psi)
+   return(wilcox.test(x = x, y = y)$p.value < 0.05)
+ }
```

Also the call of the function powCalc has to be changed slightly. Whereas in the minimal example the "power function" was evaluated only once per combination of *theta* and *n* it will now be evaluated several times [1] to be able to calculate the power as the proportion of significant evaluations.

[1]Note: the number of iterations (n.iter = 99) is to small and only for illustrative purpose!

```
> calc.resample <- powCalc(psi, statistic = powFun.resample, n.iter = 99)
```

The next two steps are the same as in the minimal example.

```
> ex <- powEx(theta = 1, power = 0.9)
> pow.resample <- merge(calc.resample, ex)
```

As in the minimal example we will now be able to inspect the estimation. The choose a sample size based on the evaluations there are two different methods (they can be addressed using the argument "method" of the function *powEx*. The default method for the resampling approach is called "lm". Therefor a linear regression is fit to the transformed values and used for estimating the sample size where the power is equal to the value asked. To see if the fit of the regression model is reasonable it is important to have a look at the inspection plot.

```
> inspect(pow.resample)
```

The power plot can be produced in exactly the same manner as in the minimal example. Especially useful for the resampling approach is the call

```
> tex(pow.resample, type = "sampling")
```

that returns after evaluation in latex "$n_{i=1,...,21} = 20, ..., 60$". It allows to see the step width used for estimation. Also the call

```
> tex(pow.resample, type = "n.iter")
```

returning the number of iterations for each combination of *theta* and *n* might be useful.

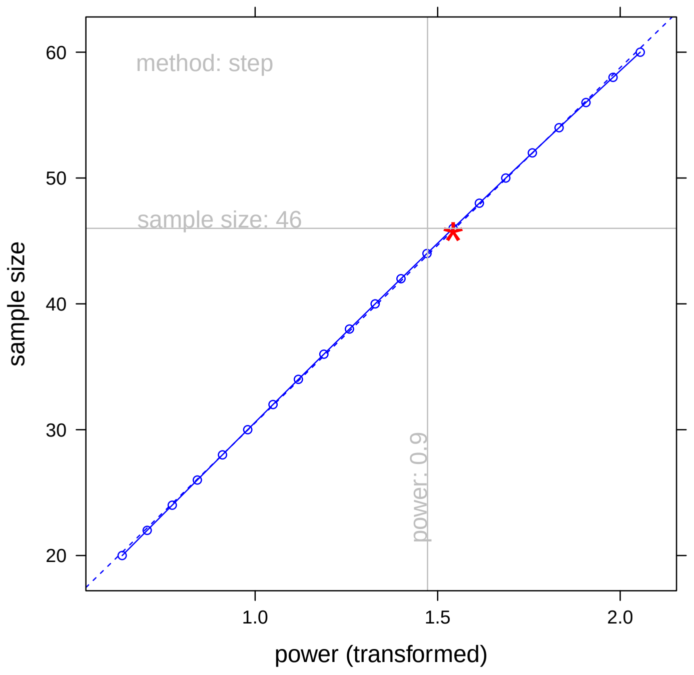

Figure 1: We see on the x-axis the observed range of power and on the y-axis the range of sample sizes. The dots represent the individual values that were calculated. The red star shows which sample size, 46, was chosen for reporting. The method "step" always takes the first sample size that is larger than the power chosen (note: the power for n = 44 is 0.8997).

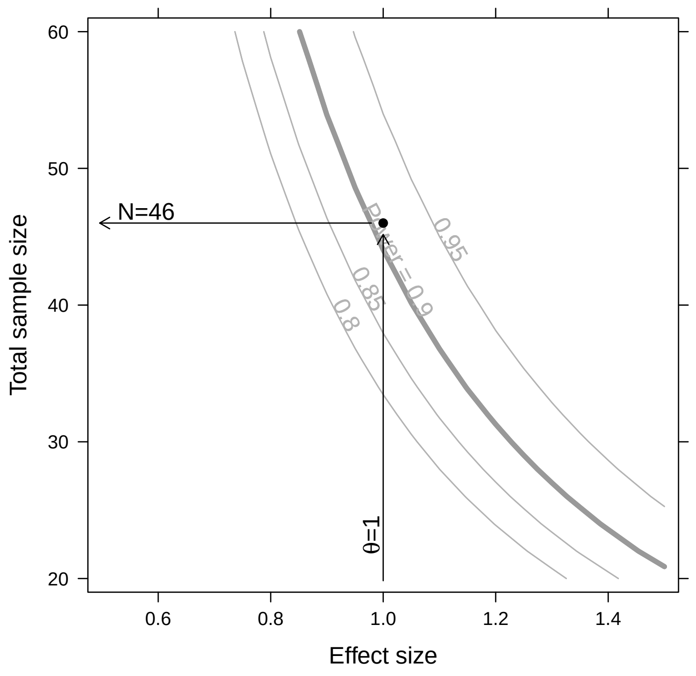

Figure 2:

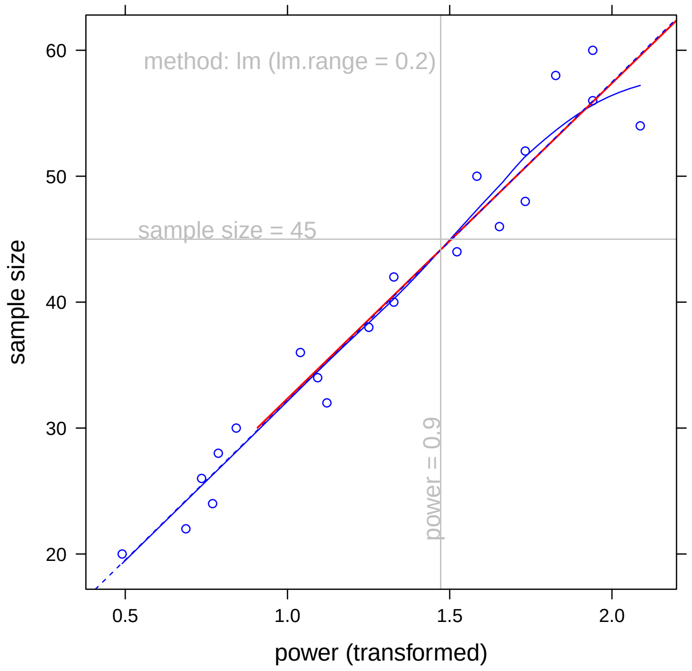

Figure 3: We see on the x-axis the observed range of power and on the y-axis the range of sample sizes. The dots represent the individual values that were calculated. The red line shows the regression line used for estimation. By default only the data in the neighbourhood of the power in focus is used for estimating the regression line (the size of the neighbourhood can be chosen using the argument lm.range of the function *powEx*).