Paper AS02

Size matters - sample size calculation with SAS and R

Fenja Seither, mainanalytics GmbH, Sulzbach, Germany

ABSTRACT

Sample size determination is a very important step in planning a clinical study. This paper intends to provide the reader with details on how to use SAS® and R to estimate the number of patients to be enrolled.

INTRODUCTION

"How many patients need to be enrolled?" – this is one of the first and most important questions during the planning of clinical studies. A sample size is required that assures accurate results without wasting limited resources. There is specific software available for sample size calculations, but this paper will focus on SAS and R methodology. It will illustrate based on five examples how to use SAS PROC PROWER as well as different R functions to determine sample sizes.

EXAMPLE 1 – DICHOTOMOUS ENDPOINTS

As a first example we will look at a fictive study, aiming to evaluate the efficacy of Treatment A versus Treatment B where the main endpoint is a dichotomous variable ("Response" or "No response"). To determine the number of patients that need to be enrolled, we assume a response rate of 55% for treatment A and 45% for treatment B. The aim is to estimate the sample size needed to detect a difference in the response rate of 10% with a power of 80% on a two-sided significance level of 5%.

When using PROC POWER, we need to specify the TWOSAMPLEFREQ TEST=pchi option to request the Pearson's chi-square test for proportion difference, enter the assumed response rates into the GROUPPROPORTION option and set POWER and ALPHA as described above. The NPERGROUP option is set to missing, as we want the number of patients per group to be estimated.

```
PROC POWER;
  TWOSAMPLEFREQ TEST = pchi
  GROUPPROPORTIONS = (0.55 0.45)
  ALPHA = 0.05
  SIDES = 2
  POWER = 0.80
  NPERGROUP = .;
RUN;
```

To estimate the sample size for our fictive study with R, we can use the pwr.2p.test function from the pwr package. First, we need to specify the effect size, which is not needed in PROC POWER. Fortunately, the function ES.h is available in the pwr package that computes effect size h for two proportions by simply entering the assumed proportions. For sample size estimation we additionally specify power and significance level. There is no need to set number of patients per group to NULL within the function call, as this is done automatically once the power is set.

```
pwr.2p.test(h = ES.h(p1 = 0.55, p2 = 0.45), alternative = "two.sided",
  sig.level = 0.05, power = 0.80)
```

EXAMPLE 2 – CONTINUOUS ENDPOINTS

The second example is a fictive study, which will evaluate the difference in the average change in a continuous variable from baseline between Treatment A and Treatment B. We assume an average change from baseline of 15% with a standard deviation of 1.5% for Treatment A and of 12% with a standard deviation of 1.9% for Treatment B. We want to estimate the number of patients to be included in the study to detect a difference of 3% in the average change from baseline with a power of 80% and a two-sided 5% significance level.

Before sample size determination with SAS or R, we need to compute the pooled standard deviation as:

$$\sigma = \sqrt{\frac{std(Treatment\ A)^2 + std(Treatment\ B)^2}{2}} \approx 1.711724$$

With this result, we have all information we need to call PROC POWER. To estimate sample size based on a pooled t-test we set the TWOSAMPLEMEANS option with MEANDIFF = 3 (because we want to detect a difference in means of 3%). All other options are set analogously to the first example leading to following SAS code to be used for sample size determination:

```
PROC POWER;
  TWOSAMPLEMEANS
  MEANDIFF = 3
  STDDEV = 1.711724
  ALPHA = 0.05
  SIDES = 2
  POWER = 0.8
  NPERGROUP = .;
RUN;
```

To estimate the sample size based on pooled t-test with R, we will use the pwr.t.test function. Once again, we need to specify the effect size. Unfortunately, in this scenario there is no function in place to do so, which is why we need to compute effect size manually as

$$d = \frac{Difference\ in\ means}{Pooled\ standard\ deviation} \approx 1.752619$$

Together with the power and significance level specified as described above, we need to use following R code to estimate the sample size for our second example:

```
pwr.t.test(d = 1.752619, power = 0.80, sig.level = 0.05 , alternative = "two.sided")
```

EXAMPLE 3 – COMPARE SURVIVAL TIMES

The third example is a study aiming to compare survival times with a power of 80% and a two-sided significance level of 5%. Let's assume that under standard Treatment A 41% of the patients survive beyond five years and the new Treatment B increases survival beyond five years to 60%. This translates into an event rate of 0.59 for treatment A and 0.4 for treatment B respectively. With these values, we can now calculate the hazards for Treatment A and Treatment B as well as the hazard ratio using

$$Hazard = \frac{-ln(1 - Event\ rate)}{t}$$

This reveals a hazard of 0.178 for Treatment A and 0.102 for Treatment B. We assume an accrual time of two years and a follow-up time of six years.
To estimate the sample size needed to detect a difference in the survival times based on the log-rank test with SAS, we need to specify TWOSAMPLESURVIVAL TEST=logrank. Moreover, we define the hazards as computed above in the GROUPSURVEXPHAZARDS option (first entry for control Treatment A, second entry for experimental Treatment B) and set assumed follow-up time, accrual time, significance level and power via FOLLOWUPTIME, ACCRUALTIME, ALPHA and POWER.

```
PROC POWER;
  TWOSAMPLESURVIVAL TEST = logrank
  GROUPSURVEXPHAZARDS = 0.178 | 0.102
  FOLLOWUPTIME = 6
  ACCRUALTIME = 2
  POWER = 0.80
  ALPHA = 0.05
  SIDES = 2
  NPERGROUP = .;
RUN;
```

To determine the sample size for the third example with R, we can use the getSampleSizeSurvival function from the rpact package as follows:

```
getSampleSizeSurvival(alpha = 0.05, sided = 2, beta = 0.2, lambda1 = 0.102,
  lambda2 = 0.178, accrualTime = 2, followUpTime = 6)
```

Thereby we set significance level, accrual time and follow-up time via alpha, accrualTime and followUpTime. Different from other R functions, the power must be specified via beta = 1-power. The hazards will be handed using lamda1 (experimental treatment B) and lambda2 (control treatment A).

EXAMPLE 4 – DOSE FINDING STUDIES

The previous examples are based on rather simple study designs. But what if it gets more complex? Let's think of a fictive and simplified dose-finding study with a binary endpoint ("Response" versus "No response"). The aim of this study is to show a significant non-flat dose-response curve across three different doses of Treatment A and placebo. The analysis will be performed using a MCPMod approach, which is an increasingly popular statistical method combining aspects of multiple comparison procedures and modeling techniques (please refer to [1] and [2] for details). The main idea is to simultaneously evaluate several possible dose-response models (patterns). For sample size determination we need to pre-specify comparator models. To keep this example simple, we will only include a linear comparator model(for a real clinical study one would use a diverse range of dose response patterns). Moreover, we assume a response rate of 20% with placebo, a maximum response rate of treatment A of 40% and the same number of patients being included in each dose group.

Unfortunately, there is no SAS procedure available for sample size estimation for dose-finding studies, meaning that we would need to do all computations manually. As this would be very lengthy, no SAS sample code will be provided at this point.

In contrast, the R function sampSizeMCT from the DoseFinding package R offers a possibility to determine the number of patients to be enrolled in the dose-finding study above. This function is based on a bisection search algorithm, which iterates the sample size so that the pre-defined power is achieved.

As a first step we need to create an R object models from the Mods class using the mods function which is also included in the DoseFinding package. We need to hand over a vector containing the coding for the three doses of treatment A (1,2,3) and for placebo (0) in the doses argument, the placebo effect placEFF as well as the maximum effect maxEFF. As mentioned above we only include a linear comparator model, which we achieve with the linear=NULL statement. Based on this object, we can now call the sampSizeMCT function specifying:

upperN: Upper bound for the target sample size (needed for bisection search algorithm)
altModels: Object of Mods class as described above
alRatio: Relative patient allocations to the dose groups
power of 80%
Two-sided significance level of 5%

```
models <- Mods(doses = c(0,1,2,3), placEff=0.2, maxEff=0.4, linear = NULL)
sampSizeMCT(upperN=80, contMat = optContr(models, w=1), sigma = 1,
       altModels=models, alRatio = rep(1, 4), power = 0.8, alpha = 0.05)
```

EXAMPLE 5 – MORE COMPLEX STUDY DESIGNS

Let's think of more complex designs, for example studies including interim analyses. In such cases there might be no formula for sample size computation at all or it might be very difficult to adapt an existing formula to fit the needs. In this case a "guess and check" approach can be used, which is based on simulation-based power calculations. Let's assume analogous to the previous examples a power of 80%. The underlying idea is to select a sample size and run simulations to estimate the power. If the power is estimated to be higher than 80%, we select a new and lower sample size and again estimate the power. This procedure will be repeated until the estimated power approximately equals 80%. To estimate the power, we repeatedly simulate our entire study and determine the proportion of studies in which the null hypothesis is rejected – this is our estimated power.

Due to the complexity of such studies, no sample code will be provided at this point. But we should keep in mind, that simulation can be computationally demanding, and runtime of underlying SAS programs might be much higher compared to R scripts.

CONCLUSION

This paper shows only a very limited number of examples and therefore a general advice which programming language to prefer can't be given. In any case the decision whether to use SAS or R for sample size calculation should be based on the study design, effort to write SAS or R programs, and of course personal preferences. For simple and standard study designs, both SAS and R provide efficient methods for sample size calculation. Therefore,

users should choose their preferred programming language in these situations. When it comes to more complex designs, SAS has some limitations compared to R, which is why R might be the better choice in such situations.

REFERENCES

[1] Bretz F., Pinheiro J., Branson M. Combining multiple comparisons and modelling techniques in dose-response studies. Biometrics. 2005;61:738–748
[2] Pinheiro J., Bjornkamp B., Glimm E., Bretz F. Model-based dose finding under model uncertainty using general parametric models. Stat. Med. 2013;33:1646–1661.

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the author at:
Fenja Seither
mainanalytics GmbH
Otto-Volger-Strasse 3c
65843 Sulzbach/Taunus, Germany
Work Phone: +49 (6196) 766 84 44
Email: fenja.seither@mainanalytics.de
Web: www.mainanalytics.de