Introduction to hybridpower

Joonsuk Park

2020-08-06

`hybridpower` is an R package that implements the idea of Bayesian-classical (BC) hybrid power analysis that was introduced by Pek & Park (2019). Specifically, it allows users to compute the BC hybrid power for popular statistical testing procedures such as t-test and ANOVA. For compatibility purposes, it can also compute classical power, sometimes by performing large numbers of Monte Carlo (MC) simulations. In this document, we provide the technical details about the package, as well as showcasing how to use each submodule.

Note that `hybridpower` does not compute sample size for a given level of power, contrary to other existing power analysis packages. To obtain a recommended sample size for a target level of power, one can rely on a method such as linear interpolation.

Organization of the package

The package follows the object-oriented paramdigm (OOP). That is, there is a parent class for all the BC hybrid power analysis submodules. The parent class is called HybridPower. Every children class for tests like t-test or sign test inherits HybridPower. This design natually fits the structure of power analysis because every power analysis requires inputs such as sample size, significance level, alternative hypothesis, and so on. Plus, in case of BC hybrid power analysis, prior distributions need to be specified and the power analysis submodules share common types of possible prior distributions such as normal or prior. The user does not need to interact with the parent class, however, and only need to deal with a children class that computes power for a specific testing procedure.

Initializing a BC hybrid power analysis instance

Before computing power, one needs to initialize a power analysis instance. Common inputs to the power analysis submodules include sample size, alpha (significance level), effect size, and the alternative hypothesis if necessary. To compute classical power only, it is not required to provide prior distributions. For example, in case of t-test, the following is a minimal example of doing classical power analysis:

```r
x_classical <- hp_ttest$new(
  ns = seq(10, 90, 10),
  d = 0.5
)
x_classical$classical_power()
```

The default values of the alternative hypothesis and the significance level are 'two.sided' and 0.05, respectively. One can change this by explicitly providing a value while initializing. Or one can change the values stored in the instances and assign new values to them, for example,

```r
x_classical$d <- 0.1
x_classical$alt <- 'one.sided'
x_classical$alpha <- 0.01
```

That can be done for other types of classes, too.

To compute BC hybrid power, one also needs to provide type of the prior and the parameters for the prior. Types of possible priors for the classes are listed in the aforementioned manuscript. The following is an example of doing BC hybrid power analysis:

```r
x_hybrid <- hp_ttest$new(
  ns = seq(10, 90, 10),
  n_prior=1000,
  prior = 'normal',
  prior_mu = 0.1,
  prior_sigma = 0.1,
  alpha=0.05,
  d=0.1,
  sd=.1
)
x_hybrid$hybrid_power()
x_hybrid$assurance()
x_hybrid$boxplot()
x_hybrid$power_quantiles()
x_hybrid$assurance_level()
```

In the instance `x_hybrid`, note that parameters regarding the prior are added. `n_prior` is the number of draws from the prior; the default value of it is 100, but the user will need to increase it to obtain more accurate power. The type of the prior is normal and the parameters are specified in `prior_mu` and `prior_sigma`. After creating the instance, one needs to run the `$hybrid_power` method to generate BC hybrid power and store them in `$output` to run other methods such as the following:

- `assurance()` returns the assurance values for the sample sizes.
- `boxplot()` returns boxplots of the generated BC hybrid power values.
- `power_quantiles()` returns quantiles of the hybrid power values. By default, percentiles corresponding to 0, 25, 50, 75, and 100 are returned. One can explicitly provide the thresholds while initializing the instance by, for example,

```r
quantiles = c(.1, .5, .9)
```

- `boxplot()` displays boxplots of the generated BC hybrid power values.
- `assurance_level()` returns what is called the `assurance levels` in the literature (Du & Wang, 2016). See the article for details. One can provide thresholds for that during initialization of the instance, e.g.,

```r
assurance_level_props = c(.1, .5, .8)
```

These methods can be used for other tests than t-test, of course. From now on, we introduce each submodule in turn.

t-test

We just saw an example of doing BC hybrid power analysis for t-test. In this section, we provide more details about the class for t-test, `hp_ttest`.

The supported types of experimental designs are 'one.sample', 'two.sample' and 'paired', as in case of t.test() function in the stats library. The default value of `sd`, the within-group population standard deviation of the data, is 1. If this is unchanged, the effect size, d, is naturally interpreted as a Cohen's d. Otherwise, d is simply an unstandardized between-group mean difference. If two distinct values are provided as `sd`, it is interpreted as indicating an unequal variances design and power analyses are done by doing MC simulations via Welch t-tests. For accurate results, one will need to increase the value of n_MC in the instance, whose default value is 100.

One-way ANOVA

The class name for ANOVA is `hp_oneway_anova`. The effect sizes are provided in terms of unstandardized group means and standard deviations. You can specify as many groups as you would like to. Priors are specified in terms of the means, not the standard deviations (they are assumed to be fixed). The following is an example instance:

```r
x_hybrid <- hp_oneway_anova$new(
  ns = seq(10, 90, 10),
  mu = c(2, 2.2),
  prior = 'normal',
  prior_mu = c(2, 2.5),
  prior_sigma = c(0, .2),
  sd = 1,
  design='fe',
  n_prior = 1000,
  n_MC = 1000,
  quantiles = c(.2, .5, .8),
  assurance_level_props = c(.5, .8)
)
```

As in case of t-test, `sd` denotes the within-group population standard deviation. The design variable, `design`, can take one of the two options: 'fe' for fixed-effects and 'rm' for repeated-measures design. In case of 'rm', one can specify two more parameters during initialization, within-subject correlation ('rho') and sphericity correction factor ('epsilon'). Without explicit initialization, they are set to 0 and 1, respectively; to avoid this, their values must be provided.

If `prior_sigma` is simply a number, or a vector with identical values, then it is assumed that variances are equal across the groups. If `prior_sigma` contains distinct values, on the other hand, the equal variances assumption is assumed to be violated and MC simulations are done to compute power in the context of Welch ANOVA. This is automatically detected and the user doesn't have to do anything other than specifying the values of `sd`.

Simple regression

The class name for simple regression is `hp_slr`. Type of the effect size is proportion of explained variance, i.e., r^2, which takes on values between 0 and 1. Below is an example:

```r
x_hybrid <- hp_slr$new(
  ns = seq(10, 90, 10),
  r2 = 0.1,
  n_prior=1000,
  prior = 'truncnorm',
  prior_mu = .2,
  prior_sigma = .2,
  prior_lower = 0,
  prior_upper = 0.5,
  assurance_level_props = c(.5, .8)
)
```

In this example we used the truncated normal prior. The limits can be set by the user or automatically set at 0 and 1. However, we suggest users use a tighter range than 0 and 1 to better quantify the degree of uncertainty about r^2.

Bivariate correlation

The class name for bivariate correlation is `hp_cor`. The input effect size is `rho`, the population Pearson correlation coefficient. Below is an example:

```r
x_hybrid <- hp_cor$new(
  ns = seq(10, 90, 10),
  n_prior=1000,
  rho = .5,
  prior_mu = .3,
  prior_sigma = .1,
  prior = 'truncnorm',
  alt = 'two.sided'
)
```

rho can take values between -1 and 1, as it should. The user can also choose either 'one.sided' or 'two.sided' as the alternative hypothesis.

Sign test

The class name for sign test is `hp_sign`. The input effect size is `p_1`, the population proportion under the alternative. While computing power, the large sample approximation formula proposed by Noether (1987) is used; one can override this by performing MC simulations by providing the `MC` $= T$ option during initialization of the instance. However, it can significantly increase the execution time especially in case of hybrid power calculation because simulations are increased by a factor of the product of n_prior and n_MC, not n_MC alone.

Below is an example:

```r
x_hybrid <- hp_sign$new(
  prior='uniform',
  prior_lower = 0.2,
  prior_upper = 0.4,
  ns = seq(10, 90, 10),
  n_prior=1000,
  n_MC = 1000,
  p_0 = 0.5,
  p_1 = 0.3,
  MC=F,
  quantiles = c(.2, .5, .8),
  assurance_level_props = c(.5, .8)
)
```

The population proportion under the null is 0.5, as it is in most cases. That under the alternative is 0.3, and the prior is Uniform(0.2, 0.4). Of course, it takes more time to conduct a BC hybrid power analysis when MC simulations are done to compute the value.

Chi-square test of goodness of fit

```r
x_hybrid <- hp_chisq$new(
  prior='beta',
  parallel = T,
  ns = seq(10, 90, 10),
  n_prior=1000,
  prior_a = 1,
  prior_b = 2,
  p_0 = c(1/2, 1/2),
  p_1 = c(1/3, 2/3),
  assurance_level_props = c(.5, .8)
)
```

Proportion tests

```r
x <- hp_prop$new(
  parallel=T,
  ns = seq(30, 90, 10),
  n_prior=10,
  prior = 'truncnorm',
  prior_pi_1_mu = .6,
  prior_pi_1_sd = .1,
  prior_pi_2_mu = .5,
  prior_pi_2_sd = .1,
  c = 0.5,
  n_MC = 1000,
  alt = 'two.sided',
  exact=F,
  pi_1 = 0.5,
  pi_2 = 0.7,
  assurance_level_props = c(.5, .8)
)
```