bayesiansprt

Contents

1 Introduction

This vignette will provide some examples of using the package `bayesiansprt` (under GPL-3 licence) to provide the results for sequential probability ratio test (SPRT) under Bayesian setup. It also provides the results based on the frequentist approach.

2 Efficacy and futility boundaries

In SPRT, we can define the thresholds, that is, efficacy and futility boundaries, for accepting the null and alternative hypothesis with respect to the number of events in the treatment group. The `SPRTboundary` function provides the efficacy and futility boundaries for the sequential ratio probability test under the frequentist and Bayesian setup. Here is the example of calculating the boundaries under the Bayesian setup. Under the frequentist context, we do not need to provide the `optimum.w`, `w`, `a.prior`, `b.prior` as these parameters are related to Bayesian priors.

2.1 Bayesian Example

```
#################
# Load packages #
#################

library(dplyr)
library(bayesiansprt)

##################
# Specify values #
##################

method = "Bayes"
n.event = 60
n.start = 30
n.increment = 5
ve.null = 0.3
ve.alt = 0.75
type1 = 0.025
power = 0.9
ratio.alloc = 1
optimum.w = TRUE
w = 0.5
b.prior = 1


########################
# Calculate boundaries #
########################

boundary.result = SPRTboundary(method, n.event, n.start, n.increment, ve.null,
                               ve.alt, type1, power, ratio.alloc, optimum.w, w,
                               b.prior)

boundary.result$boundary
#>    m lower upper
#> 1 30     5    11
#> 2 35     7    12
#> 3 40     9    14
#> 4 45    10    15
#> 5 50    11    17
#> 6 55    13    19
#> 7 60    14    20
```

2.2 Frequentist Example

```
##################
# Specify values #
##################

method = "Freq"

########################
# Calculate boundaries #
########################

boundary.result = SPRTboundary(method, n.event, n.start, n.increment, ve.null,
                               ve.alt, type1, power, ratio.alloc, optimum.w, w,
                               b.prior)

boundary.result$boundary
#>    m lower upper
#> 1 30     5    12
#> 2 35     6    13
#> 3 40     8    15
#> 4 45     9    16
#> 5 50    11    18
#> 6 55    12    19
#> 7 60    14    21
```

3 SPRT

Now, based on the boundaries with interim time-points and weights of hyper-prior, we can generate data and calculate the probability corresponding to the boundaries (r_1 and r_2), estimated type-I error and power, cumulative probability, and average number of events. In this example, we have shown the results under null hypothesis.

```
##########
# Result #
##########

# Under null hypothesis
SPRTonVE(n.sim = 10000, boundary.val = boundary.result$boundary,
         ve.null = boundary.result$ve.null, ve.alt = boundary.result$ve.alt,
         ve = boundary.result$ve.null, ratio.alloc = boundary.result$ratio.alloc,
         seed = 134)
#> $boundary
#>    m lower upper
#> 1 30     5    12
#> 2 35     6    13
#> 3 40     8    15
#> 4 45     9    16
#> 5 50    11    18
#> 6 55    12    19
#> 7 60    14    21
#>
#> $prob.rejH0
#>   prob_dec
#> 1   0.0093
#>
#> $cum.prob
#>      Total_cp cp_rejectH0 cp_acceptH0
#> [1,]   0.6296      0.0042      0.6254
#> [2,]   0.7646      0.0051      0.7595
#> [3,]   0.8023      0.0069      0.7954
#> [4,]   0.8610      0.0071      0.8539
#> [5,]   0.8825      0.0084      0.8741
#> [6,]   0.9177      0.0086      0.9091
#> [7,]   0.9294      0.0093      0.9201
#>
#> $prob.each.n
#>      x_less_r1 x_greater_r2
#> [1,]    0.0042       0.6254
#> [2,]    0.0029       0.7500
#> [3,]    0.0045       0.7437
#> [4,]    0.0019       0.8244
#> [5,]    0.0035       0.8213
#> [6,]    0.0019       0.8819
#> [7,]    0.0028       0.8703
#>
#> $percent.stop
#>    percent.stop
#> 30       0.6296
#> 35       0.1350
#> 40       0.0377
#> 45       0.0587
#> 50       0.0215
#> 55       0.0352
#> 60       0.0823
#>
#> $summary.stat
#>         Summary
#> Min.    30.0000
#> 1st Qu. 30.0000
#> Median  30.0000
#> Mean    35.7115
#> 3rd Qu. 35.0000
#> Max.    60.0000
#> sd       9.6130
#>
#> $time
#> Time difference of 10.03249 secs
#>
#> $seed
#> [1] 134
```

4 Optimum weight for Bayesian prior

The main attractive feature of this package is to get optimum weight for interim analysis under the Bayesian setup. Using the SPRT rules, we can obtain the threshold for the number of events in the treat group to make the decision at each interim or final analysis. Due to discrete nature of the binomial distribution, the critical region for the efficacy boundary could be the same for multiple values of total number of events. Therefore, in SPRT we should find the optimum interim time-points where we can make decisions the earliest with high power. So, choosing an interim time-point with the minimum number of events for a given critical region of the efficacy boundary should provide better power. In the following example, we show that how we can optimize the weight and use this for calculating results under Bayesian paradigm.

```
method = "Bayes"
n.sim = 10000
n.event = 60
n.start = 30
n.increment = 5
ve.null = 0.3
ve.alt = 0.75
type1 = 0.025
power = 0.9
ratio.alloc = 1
optimum.w = TRUE
b.prior = 1
seed = 134

##########
# Result #
##########

opt.w = SPRToptimumW(n.sim = n.sim, n.event = n.event, n.start = n.start,
                     n.increment = n.increment, ve.null = ve.null, ve.alt = ve.alt,
                     type1 = type1, power = power, ratio.alloc = ratio.alloc,
                     w = seq(0.1, 0.9, by = 0.1), b.prior = b.prior, seed = seed)

results = outputSPRT(method = method, n.sim = n.sim, n.event = n.event, n.start = n.start,
                     n.increment = n.increment, ve.null = ve.null, ve.alt = ve.alt,
                     type1 = type1, power = power, ratio.alloc = ratio.alloc,
                     optimum.w = optimum.w, w = opt.w$weight, b.prior = b.prior,
                     seed = seed)
results
#> $boundary
#>   n_s r1 r2
#> 1  30  5 11
#> 2  35  7 13
#> 3  40  9 14
#> 4  45 10 16
#> 5  50 12 17
#> 6  55 13 19
#> 7  60 15 20
#>
#> $cumulative.prob
#>   n_s r1 r2 P(RejectH0|H0) P(RejectH0|H1)
#> 1  30  5 11         0.0042         0.4246
#> 2  35  7 13         0.0084         0.6080
#> 3  40  9 14         0.0134         0.7438
#> 4  45 10 16         0.0141         0.7748
#> 5  50 12 17         0.0165         0.8425
#> 6  55 13 19         0.0166         0.8587
#> 7  60 15 20         0.0174         0.8928
#>
#> $type1.power
#>          prob_dec
#> Under.H0   0.0174
#> Under.H1   0.8928
#>
#> $prob.VE
#>   n_s P(x<=r1|H0) P(x>r2|H0) P(x<=r1|H1) P(x>r2|H1)
#> 1  30      0.0042     0.7576      0.4246     0.0273
#> 2  35      0.0080     0.7500      0.5976     0.0152
#> 3  40      0.0107     0.8377      0.7298     0.0217
#> 4  45      0.0058     0.8244      0.7148     0.0123
#> 5  50      0.0085     0.8867      0.8140     0.0144
#> 6  55      0.0045     0.8819      0.8019     0.0091
#> 7  60      0.0054     0.9223      0.8687     0.0116
#>
#> $percent.stop
#>   n_s Avg(%)|H0 Avg(%)|H1
#> 1  30    0.7618    0.4519
#> 2  35    0.0461    0.1855
#> 3  40    0.0688    0.1428
#> 4  45    0.0217    0.0321
#> 5  50    0.0342    0.0699
#> 6  55    0.0114    0.0168
#> 7  60    0.0560    0.1010
#>
```

```
#> $summary.stat
#>         Under.H0 Under.H1
#> Min.     30.0000  30.0000
#> 1st Qu.  30.0000  30.0000
#> Median   30.0000  35.0000
#> Mean     33.8930  37.6850
#> 3rd Qu.  30.0000  40.0000
#> Max.     60.0000  60.0000
#> sd        8.3049   9.8209
#>
#> $time
#>          Time (in seconds)
#> Under.H0          8.749508
#> Under.H1          6.790189
#>
#> $method
#> [1] "Bayes"
#>
#> $n.event
#> [1] 60
#>
#> $n.start
#> [1] 30
#>
#> $n.increment
#> [1] 5
#>
#> $ve.null
#> [1] 0.3
#>
#> $ve.alt
#> [1] 0.75
#>
#> $type1
#> [1] 0.025
#>
#> $power
#> [1] 0.9
#>
#> $ratio.alloc
#> [1] 1
#>
#> $optimum.w
#> [1] TRUE
#>
#> $w
#> [1] 0.7
#>
#> $a.prior
#> [1] 0.534296
#>
#> $b.prior
#> [1] 1
```

```
#>
#> $seed
#> [1] 134
```