Session 3

Sample Size Estimation

Matt Denwood

2024-01-30

A note on coding styles

My code looks a bit different to Giles's code, e.g.:

```r
# If necessary:
## install.packages(c("tidyverse","pbapply"))

library("tidyverse")
library("pbapply")
```

A note on coding styles

My code looks a bit different to Giles's code, e.g.:

```r
# If necessary:
## install.packages(c("tidyverse","pbapply"))

library("tidyverse")
library("pbapply")
```

REMEMBER: the coding style is not important as long as the output is the same!

Background to sample size calculations

Power calculation

Power is defined as the proportion of experiments that can be expected to give p-values of $<=$ 0.05 (or whatever alpha is chosen), conditional on the specified parameters.

Power calculations can be done using:

1. Approximation methods, e.g. power.t.test:

```
power.t.test(n = 150, delta = 0.25, sd = 1)
##
##      Two-sample t test power calculation
##
##               n = 150
##           delta = 0.25
##              sd = 1
##       sig.level = 0.05
##           power = 0.5785239
##     alternative = two.sided
##
## NOTE: n is number in *each* group
```

2. Numerical methods i.e. by simulation:

A function to simulate data, then calculate and return a p-value:

```
p_fun <- function(parameters){
  stopifnot(is.data.frame(parameters), nrow(parameters)==1L, "Size" %in% names(parameters))
  sample1 <- rnorm(parameters$Size, mean=0, sd=1)
  sample2 <- rnorm(parameters$Size, mean=0.25, sd=1)
  parameters |>
    mutate(P_val = t.test(sample1, sample2)$p.value)
}
```

```
p_fun(tibble(Size = 150L))
## # A tibble: 1 x 2
##    Size  P_val
##   <int>  <dbl>
## 1   150 0.0171
```

There is randomness so this will be different every time it is run:

```
p_fun(tibble(Size = 150L))
## # A tibble: 1 x 2
##    Size  P_val
##   <int>  <dbl>
## 1   150 0.0109
p_fun(tibble(Size = 150L))
## # A tibble: 1 x 2
##    Size  P_val
##   <int>  <dbl>
## 1   150 0.0119
```

So we must run it several times (e.g. 1000):

```r
tibble(Iteration = seq_len(1000L), Size = 150L) |>
  group_split(Iteration, Size) |>
  lapply(p_fun) |>
  bind_rows() ->
  pvals
```

So we must run it several times (e.g. 1000):

```r
tibble(Iteration = seq_len(1000L), Size = 150L) |>
  group_split(Iteration, Size) |>
  lapply(p_fun) |>
  bind_rows() ->
  pvals
```

And we calculate the power like so:

```r
pvals |>
  group_by(Size) |>
  summarise(Power = sum(P_val <= 0.05) / n(), .groups="drop")
## # A tibble: 1 x 2
##    Size Power
##   <int> <dbl>
## 1   150 0.581
```

Sample size estimation

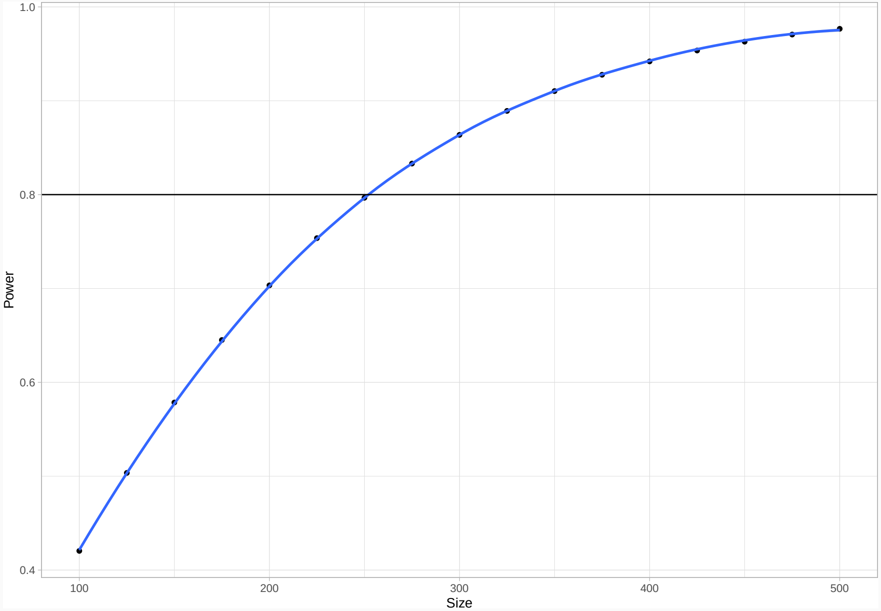
The goal is typically to find the minimum sample size that corresponds to $>=$ 80% power, for a specified set of parameters. This can be done in one of two ways:

1. Using approximation methods directly i.e.:

```
power.t.test(n = NULL, delta = 0.25, sd = 1, power = 0.8)
##
##      Two-sample t test power calculation
##
##               n = 252.1281
##           delta = 0.25
##              sd = 1
##       sig.level = 0.05
##           power = 0.8
##     alternative = two.sided
##
## NOTE: n is number in *each* group
```

2. By trying different sample sizes (using either approximation methods or simulation):

```r
tibble(Size = seq(100, 500, by=25)) |>
  group_split(Size) |>
  lapply(function(parameters){
    parameters |>
      mutate(Power = power.t.test(n = parameters$Size, delta = 0.25, sd = 1)$power)
  }) |>
  bind_rows() ->
  power_estimates
```


1.0
0.8
0.6
0.4
Power
100
200
300
400
500
Size

Sample size calculation for LCM

Determining the objective

Let’s take a simple 2-test, 2-population Hui-Walter model as an example.

- There is (usually) no hypothesis to test, so we don’t have a concept of power
- Our objective is to have narrow posterior (95%) credible intervals (“precision”*)
- But for which parameter?
- Note: this is not exactly the same as the usual statistical definition of precision

Types of parameter

In general:

- Experimental parameter: structural things that we can control
- Parameter of interest: things we want to estimate
- Nuisance parameters: everything else

Types of parameter

In general:

- Experimental parameter: structural things that we can control
- Parameter of interest: things we want to estimate
- Nuisance parameters: everything else

For t-tests:

- Experimental parameter: sample size
- Parameter of interest: difference in means
- Nuisance parameters: standard deviation

In general:

- Experimental parameter: structural things that we can control
- Parameter of interest: things we want to estimate
- Nuisance parameters: everything else

For Hui-Walter models:

- Experimental parameters: number of samples from each population (and maybe number of populations)
- Parameters of interest: sensitivities, specificities
- Nuisance parameters: prevalences (and maybe correlation terms for >2 tests)

Determining the objective

We have 4 parameters of interest: 2 x Se, 2 x Sp. Which are most important?

- Is sensitivity the sole objective? NB: high-prevalence population will be prioritised.
- Is specificity the sole objective? NB: low-prevalence population will be prioritised.
- Are sensitivity and specificity equally important? If so, do we average the relative or absolute size of their 95% CI?

Determining the objective

We have 4 parameters of interest: 2 x Se, 2 x Sp. Which are most important?

- Is sensitivity the sole objective? NB: high-prevalence population will be prioritised.
- Is specificity the sole objective? NB: low-prevalence population will be prioritised.
- Are sensitivity and specificity equally important? If so, do we average the relative or absolute size of their 95% CI?
- Would Youden's index be easier?

Determining the objective

We have 4 parameters of interest: 2 x Se, 2 x Sp. Which are most important?

- Is sensitivity the sole objective? NB: high-prevalence population will be prioritised.
- Is specificity the sole objective? NB: low-prevalence population will be prioritised.
- Are sensitivity and specificity equally important? If so, do we average the relative or absolute size of their 95% CI?
- Would Youden's index be easier?
- One or both tests?

Determining the objective

We have 4 parameters of interest: 2 x Se, 2 x Sp. Which are most important?

- Is sensitivity the sole objective? NB: high-prevalence population will be prioritised.
- Is specificity the sole objective? NB: low-prevalence population will be prioritised.
- Are sensitivity and specificity equally important? If so, do we average the relative or absolute size of their 95% CI?
- Would Youden's index be easier?
- One or both tests?

My suggestion: define "precision" as the average width of 95% CI for Youden's index (for either one or both tests)

Simulating data

This is best as a function that takes population-level and test-level inputs:

```r
simulation_fun <- function(populations, tests){
  stopifnot(
    is.data.frame(populations),
    nrow(populations) >= 2L,
    c("N","Prev") %in% names(populations),
    populations$N >= 1L,
    populations$N %% 1 == 0,
    populations$Prev >= 0, populations$Prev <= 1
  )
  stopifnot(
    is.data.frame(tests),
    nrow(tests) >= 2L,
    c("Se","Sp") %in% names(tests),
    tests$Se >= 0, tests$Se <= 1,
    tests$Sp >= 0, tests$Sp <= 1
  )
  ## Do some stuff like from session 2 and return the simulated dataset
  ## See the exercise for a complete function!
}
```

Output looks like:

```
tests <- tribble(
  ~Se, ~Sp,
  0.8, 0.99,
  0.9, 0.95
)

populations <- tribble(
  ~N, ~Prev,
  100, 0.1,
  100, 0.4
)

(data <- simulation_fun(populations, tests))
##      [,1] [,2]
## [1,]   89   59
## [2,]    0    4
## [3,]    3    6
## [4,]    8   31
```

Analying data

Also best as a function taking the data, burnin and sample iterations as inputs:

```r
analysis_fun <- function(data, burnin=1000L, sample=5000L){
  stopifnot(is.matrix(data), nrow(data)==4L, ncol(data)==2L, data>=0L)
  ## Do some stuff like from session 2 to analyse the data
  ## See the exercise for a complete function
  results |>
    summary(vars=c("youden","se","sp")) |>
    as.data.frame() |>
    rownames_to_column("Variable")
}
```

Output looks like:

```
analysis_fun(data)
## Loading required namespace: rjags
##    Variable   Lower95    Median   Upper95      Mean
## 1 youden[1] 0.7381103 0.8629339 0.9911514 0.8596757
## 2 youden[2] 0.7697796 0.8792681 0.9705061 0.8752323
## 3     se[1] 0.7546444 0.8753223 0.9994181 0.8720793
## 4     se[2] 0.8169770 0.9123494 0.9979675 0.9073182
## 5     sp[1] 0.9643916 0.9908675 0.9999995 0.9875964
## 6     sp[2] 0.9296448 0.9709498 0.9999884 0.9679141
##           SD      Mode        MCerr MC%ofSD SSeff
## 1 0.06836868 0.8648971 0.0009361798     1.4  5333
## 2 0.05299179 0.8851866 0.0006097953     1.2  7552
## 3 0.06719288 0.8790966 0.0009231673     1.4  5298
## 4 0.04892656 0.9228860 0.0005505624     1.1  7897
## 5 0.01141568 0.9959492 0.0001759331     1.5  4210
## 6 0.02073107 0.9775405 0.0002922083     1.4  5033
##           AC.10      psrf
## 1  7.717975e-03 1.0006418
## 2  7.472875e-03 0.9999681
## 3  1.175842e-02 1.0006085
## 4  1.007350e-02 1.0001019
## 5  2.444691e-02 1.0000588
## 6 -7.212734e-05 1.0013366
```

A quick note on label switching

I now recommend this method of specifying minimally informative priors:

```
model{
  ### Rest of the model as usual

  for(t in 1:2){
    se[t] ~ dbeta(2,1)
    sp[t] ~ dbeta(2,1)
    youden[t] <- se[t]+sp[t]-1.0
    AcceptTest[t] ~ dbern(ifelse(youden[t] >= 0.0, 1, 0))
  }
 #data# AcceptTest
}

AcceptTest <- c(1,1)
```

A quick note on label switching

I now recommend this method of specifying minimally informative priors:

```
model{
  ### Rest of the model as usual

  for(t in 1:2){
    se[t] ~ dbeta(2,1)
    sp[t] ~ dbeta(2,1)
    youden[t] <- se[t]+sp[t]-1.0
    AcceptTest[t] ~ dbern(ifelse(youden[t] >= 0.0, 1, 0))
  }
 #data# AcceptTest
}

AcceptTest <- c(1,1)
```

More on this during my presentation tomorrow!

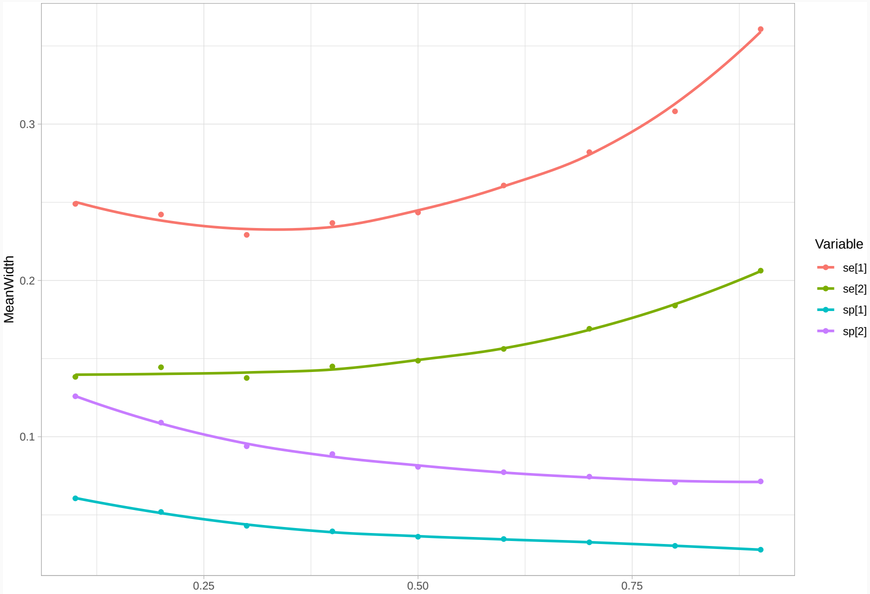
Combining the two

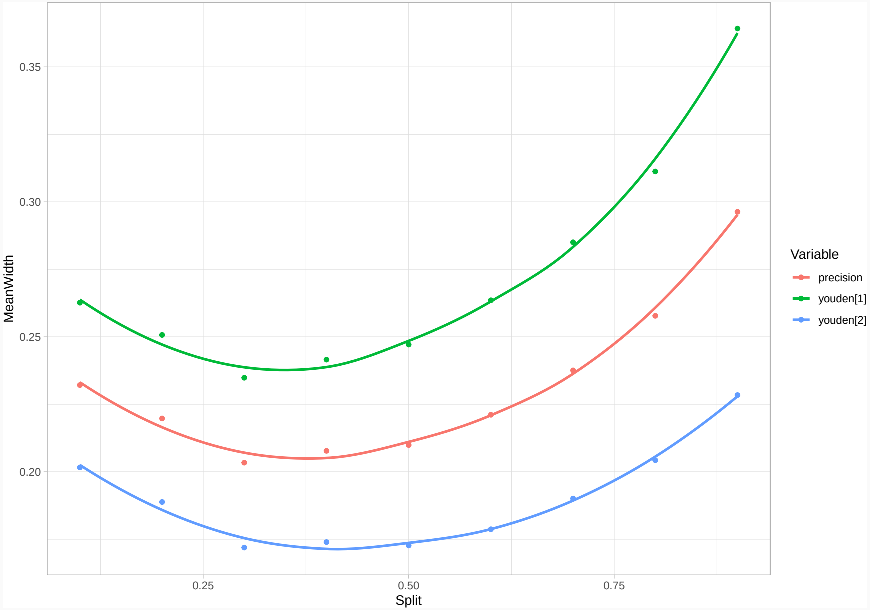
```
summary_fun <- function(parameters, iterations, cl=NULL, burnin=1000L, sample=5000L){
  stopifnot(is_tibble(parameters))
  stopifnot(iterations >= 1L)

  parameters |>
    mutate(ParameterSet = row_number()) |>
    expand_grid(Iteration = seq_len(iterations)) |>
    rowwise() |>
    group_split() |>
    pblapply(function(x){
      simulation_fun(x$Populations[[1]], x$Tests[[1]]) |>
        analysis_fun(burnin=burnin, sample=sample) |>
        bind_cols(x)
    }, cl=cl) |>
    bind_rows() |>
    mutate(WidthCI = Upper95 - Lower95) |>
    group_by(ParameterSet, Variable) |>
    summarise(MeanEst = mean(Mean), MeanWidth = mean(WidthCI), MeanLCI = mean(Lower95),
    ↪  MeanUCI = mean(Upper95), .groups="drop") |>
    full_join(parameters |> mutate(ParameterSet = row_number()), by="ParameterSet")
}
```

Output looks like:

```
parameters <- tibble(Populations=list(populations), Tests=list(tests))
summary_fun(parameters, 10L)
## # A tibble: 6 x 8
##   ParameterSet Variable  MeanEst MeanWidth MeanLCI MeanUCI
##          <int> <chr>       <dbl>     <dbl>   <dbl>   <dbl>
## 1            1 se[1]       0.833    0.293    0.687   0.980
## 2            1 se[2]       0.910    0.180    0.814   0.994
## 3            1 sp[1]       0.978    0.0512   0.949   1.00
## 4            1 sp[2]       0.942    0.107    0.888   0.995
## 5            1 youden[1]   0.812    0.305    0.660   0.965
## 6            1 youden[2]   0.852    0.223    0.737   0.960
## # i 2 more variables: Populations <list>, Tests <list>
```

Visualisaing the results





It is easier to see a balance between these using Youden’s index or our “precision”.

Exercises

1. Either have a go at writing your own functions for simulation_fun and analysis_fun (based on the code from earlier today), or just look at the function code given in the HTML file (although make sure you read it through so that you understand it!).
2. Re-create the "visualising the results" plot I have given above (using either your own functions or the functions I have provided below). Test #1 has Se/Sp of 0.8/0.99 and Test #2 has Se/Sp of 0.9/0.95. Prevalences in the two populations are 10% and 40%, and the total sample size is 500.
3. Assuming that the optimum distribution of tests between the two populations is always around 40% / 60% for these test/prevalence estimates, create a plot showing how precision increases with total sample size of between 100 and 1000.
4. How much does the precision also depend on the parameters of interest (diagnostic test performance) and nuisance parameters (prevalences)?