Crossover Trial Power Simulation en mass and Plotting

Eamonn O'Brien

10 August, 2019

Contents

List of Figures

List of Tables

Function to simulate crossover trial dataset including simulating missing data

```
# 'miss' is the percentage of data that is missing Entering
# n=10 for example will mean there are a total of 20
# patients, 10 in each treatment sequence So n is the number
# in each treatment sequence

ABBA <- function(n, intercept, main.effect, order.effect, interaction,
    sdb, sdw, miss) {

    beta <- c(intercept, main.effect, order.effect, interaction)
    Patient <- as.factor(rep(1:(2 * n), rep(2, 2 * n)))
    Treatment <- c(rep(c("Treatment1", "Treatment2"), n), rep(c("Treatment2",
        "Treatment1"), n))
    Order <- rep(c("First", "Second"), 2 * n)
    Data <- data.frame(Patient, Treatment, Order)
    FMat <- model.matrix(~Treatment * Order, data = Data)
    RMat <- model.matrix(~0 + Patient, data = Data)
    Response <- FMat %*% beta + RMat %*% rnorm(2 * n, 0, sdb) +
        rnorm(4 * n, 0, sdw)
    Data$Response <- Response
    df <- as.data.frame(Data)
    df$Response[sample(nrow(df), round(nrow(df) * miss, 0))] <- NA  # may set some data to NA here
    return(data.frame(df))

}
```

Show a simulated data sets

```
ABBA(n = 10, intercept = 1, main.effect = 0, order.effect = 0,
    interaction = 0, sdb = 1, sdw = 1, miss = 0.1)  # show a simulated data set
```

```
   Patient  Treatment  Order     Response
1        1 Treatment1  First -0.628299353
2        1 Treatment2 Second  0.221549439
3        2 Treatment1  First           NA
4        2 Treatment2 Second  0.040931281
5        3 Treatment1  First  1.933669046
6        3 Treatment2 Second  0.872015003
7        4 Treatment1  First           NA
8        4 Treatment2 Second  1.223881509
9        5 Treatment1  First -0.008849202
10       5 Treatment2 Second  2.383102656
11       6 Treatment1  First  3.141529208
12       6 Treatment2 Second  2.419993504
13       7 Treatment1  First  2.356041867
14       7 Treatment2 Second  2.339049694
15       8 Treatment1  First  0.556519847
16       8 Treatment2 Second  0.423579019
17       9 Treatment1  First  0.867064802
18       9 Treatment2 Second  0.251235438
19      10 Treatment1  First  0.248375366
20      10 Treatment2 Second  0.173867029
21      11 Treatment2  First  1.529374819
22      11 Treatment1 Second  2.016164519
23      12 Treatment2  First  0.094417475
24      12 Treatment1 Second  3.528769792
25      13 Treatment2  First  2.608733449
26      13 Treatment1 Second  0.277662867
27      14 Treatment2  First  0.707797881
28      14 Treatment1 Second  0.644027362
29      15 Treatment2  First           NA
30      15 Treatment1 Second  0.360789799
31      16 Treatment2  First  3.040231651
32      16 Treatment1 Second           NA
33      17 Treatment2  First  1.454980021
34      17 Treatment1 Second  2.866452762
35      18 Treatment2  First -1.192388142
36      18 Treatment1 Second  0.549853448
37      19 Treatment2  First  0.152603097
38      19 Treatment1 Second  2.285969651
39      20 Treatment2  First  0.651062836
40      20 Treatment1 Second  0.743150161
```

Function to use prior function repeatedly

```
require(nlme)

IHC.power <- function(a, b, c, d, e, f, g, n.sims = sims, miss) {

    treatment <- order <- interaction <- rep(NA, n.sims)  # capture output

    for (s in 1:n.sims) {

        # create a data set using above function
        fake <- ABBA(n = a, intercept = b, main.effect = c, order.effect = d,
            interaction = e, sdb = f, sdw = g, miss = miss)

        # analyse
        possibleError <- tryCatch(lme(Response ~ Treatment *
            Order, random = ~1 | Patient, data = fake, na.action = "na.omit"),
            error = function(e) e)

        ### http://stackoverflow.com/questions/8093914
        ### /skip-to-next-value-of-loop-upon-error-in-r-trycatch

        if (!inherits(possibleError, "error")) {

            modelint <- possibleError

            z <- as.matrix(summary(modelint)$tTable)
            treatment[s] <- z[2, 5][[1]]
            order[s] <- z[3, 5][[1]]
            interaction[s] <- z[4, 5][[1]]

        }
    }

    A <- mean(treatment < 0.05)
    B <- mean(order < 0.05)
    C <- mean(interaction < 0.05)

    c(A, B, C)

}
```

Function to use function repeatedly and vary the inputs, print power

```
# execute the functions numerous times varying inputs
sims <- 1000

# these are the parameters that vary
n <-   c(10, 20, 30, 40 ,50)   # double to find the actual no. of patients
m <-   c(0, 0.10, 0.25)        # missing data %
sdw <- c(1.5, 2.0)             # within person sd


# set up array to capture output
dnam = list(  N=n, Miss.perc= m, SD.within=sdw , Power=c("main.effect","order","interaction" ))
pwpr <- array( NA, dim=sapply( dnam, length ), dimnames=dnam )
str(pwpr)
```

```
 logi [1:5, 1:3, 1:2, 1:3] NA NA NA NA NA NA ...
 - attr(*, "dimnames")=List of 4
  ..$ N        : chr [1:5] "10" "20" "30" "40" ...
  ..$ Miss.perc: chr [1:3] "0" "0.1" "0.25"
  ..$ SD.within: chr [1:2] "1.5" "2"
  ..$ Power    : chr [1:3] "main.effect" "order" "interaction"
```

```
# run the simulations, set up the truth: n; intercept; ;main.effect; order.effect; interaction; sdb; sd
system.time(

  for (i in 1:length(n))
    for (j in 1:length(m))
      for (k in 1:length(sdw))

        pwpr[i,j,k,] <- IHC.power( a=n[i], b=8, c=.9, d=0, e=0, f=1, g=sdw[k], miss=m[j], n.sims=sims )

)
```

```
   user  system elapsed
1008.52    0.27 1013.56
```

Print power for scenarios

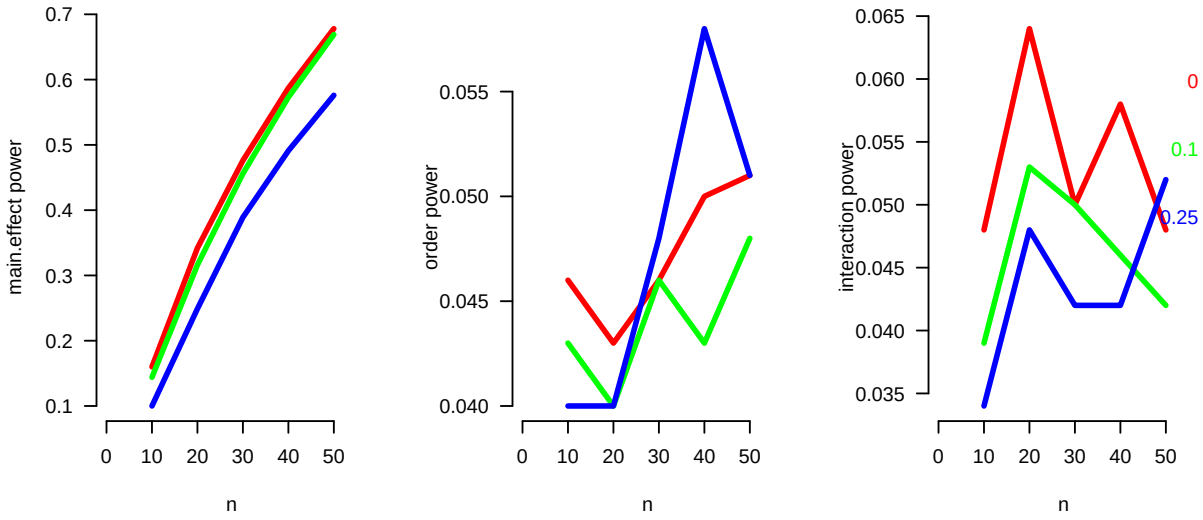
```
print(pwpr, digits = 4)
```

```
, , SD.within = 1.5, Power = main.effect

    Miss.perc
N        0   0.1  0.25
  10 0.160 0.144 0.100
  20 0.342 0.316 0.249
  30 0.476 0.456 0.389
  40 0.587 0.573 0.491
  50 0.678 0.669 0.576

, , SD.within = 2, Power = main.effect

    Miss.perc
N        0   0.1  0.25
  10 0.114 0.122 0.097
  20 0.206 0.190 0.176
  30 0.322 0.278 0.230
  40 0.438 0.401 0.340
  50 0.464 0.505 0.385

, , SD.within = 1.5, Power = order

    Miss.perc
N        0   0.1  0.25
  10 0.046 0.043 0.040
  20 0.043 0.040 0.040
  30 0.046 0.046 0.048
  40 0.050 0.043 0.058
  50 0.051 0.048 0.051

, , SD.within = 2, Power = order

    Miss.perc
N        0   0.1  0.25
  10 0.043 0.035 0.029
  20 0.051 0.053 0.041
  30 0.039 0.050 0.031
  40 0.051 0.041 0.057
  50 0.050 0.043 0.041

, , SD.within = 1.5, Power = interaction

    Miss.perc
N        0   0.1  0.25
  10 0.048 0.039 0.034
  20 0.064 0.053 0.048
  30 0.050 0.050 0.042
  40 0.058 0.046 0.042
  50 0.048 0.042 0.052
```

```
, , SD.within = 2, Power = interaction

    Miss.perc
N        0   0.1  0.25
  10 0.047 0.032 0.034
  20 0.054 0.046 0.038
  30 0.045 0.045 0.045
  40 0.059 0.047 0.045
  50 0.044 0.033 0.049
```

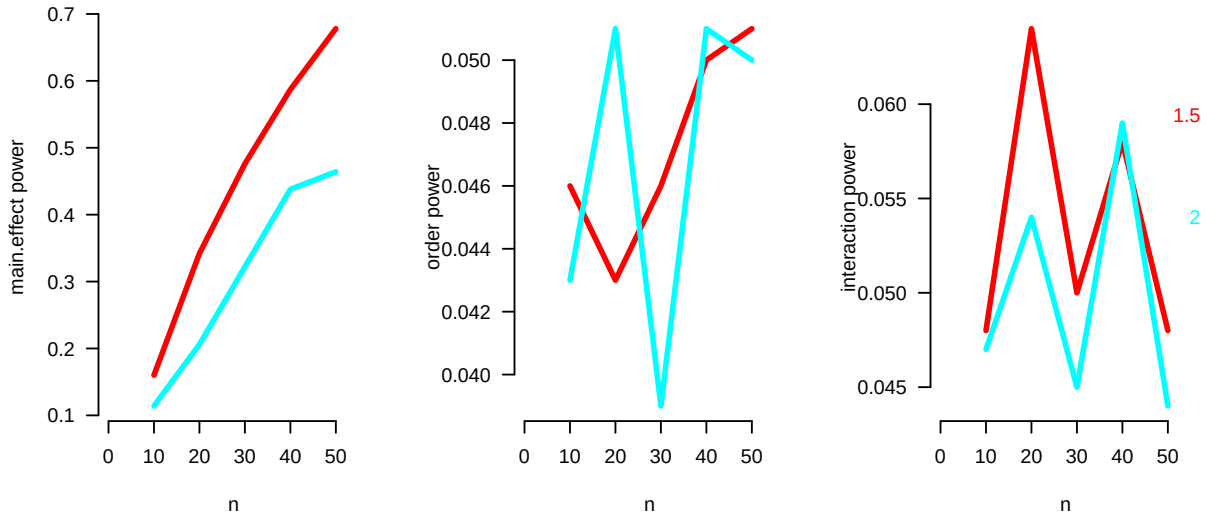
Plot power in which proportion of missing data effect on power (main/order/interaction) is explored

```
# hold within SD constant at first value cycle through main, order and interaction
lR <- length(m)  # this will vary in each plot
clr <- rainbow( lR )
wR <- (2*lR-1:lR)/(2*lR)


par(mfrow=c(1,3))
for( wh in dimnames(pwpr)[[4]] )  # (pwpr[,,"1",wh]),
  matplot( n, (pwpr[,,"1.5",wh]),  # within person SD held constant
           type="l", lty=1, lwd=3, col=clr,
           ylab=paste0(wh," power"), xlim=c(0,max(n)*1.1), bty="n", las=1 )

text( par("usr")[rep(2,lR)],
      par("usr")[4]*wR + par("usr")[3]*(1-wR),
      dimnames(pwpr)[[2]], adj=1, col=clr )
```



Plot power in which within person SD effect on power (main/order/interaction) is explored

```
# hold missing constant at first value cycle through main, order and interaction
lR <- length(sdw)  # this will vary in each plot
clr <- rainbow( lR )
wR <- (2*lR-1:lR)/(2*lR)


par(mfrow=c(1,3))
for( wh in dimnames(pwpr)[[4]] )  # (pwpr[,,"1",wh]),
  matplot( n, (pwpr[,"0",,wh]),   # missing % held constant
           type="l", lty=1, lwd=3, col=clr,
           ylab=paste0(wh," power"), xlim=c(0,max(n)*1.1), bty="n", las=1 )

text( par("usr")[rep(2,lR)],
      par("usr")[4]*wR + par("usr")[3]*(1-wR),
      dimnames(pwpr)[[3]], adj=1, col=clr )
```



References

http://www.r-bloggers.com/statistical-aspects-of-two-way-cross-over-studies/
http://stackoverflow.com/questions/8289463/extracting-fixed-effects-and-standard-errors-from-several-lme-objects-in-r
http://www.r-bloggers.com/simulated-powerprecision-analysis/
http://biostatmatt.com/archives/2315

Computing Environment

```
sessionInfo()
```

```
R version 3.6.1 (2019-07-05)
Platform: x86_64-w64-mingw32/x64 (64-bit)
Running under: Windows 10 x64 (build 17134)

Matrix products: default

locale:
[1] LC_COLLATE=English_United Kingdom.1252
[2] LC_CTYPE=English_United Kingdom.1252
[3] LC_MONETARY=English_United Kingdom.1252
[4] LC_NUMERIC=C
[5] LC_TIME=English_United Kingdom.1252

attached base packages:
[1] stats     graphics  grDevices utils     datasets
[6] methods   base

other attached packages:
[1] nlme_3.1-140 knitr_1.23

loaded via a namespace (and not attached):
 [1] compiler_3.6.1  magrittr_1.5    formatR_1.7
 [4] tools_3.6.1     htmltools_0.3.6 yaml_2.2.0
 [7] Rcpp_1.0.1      stringi_1.4.3   rmarkdown_1.14
[10] grid_3.6.1      stringr_1.4.0   xfun_0.8
[13] digest_0.6.20   lattice_0.20-38 evaluate_0.14
```

This took 1008.99 seconds to execute.