A practical Bayesian stepped wedge design for community-based cluster-randomized clinical trials: the British Columbia Telehealth Trial

Supplementary Material

```
library(snowfall)

### statistical significance level ###

alpha = 0.1

### number of clusters/ sample size per cluster ###

p = 50; n = 20

### number of time periods (every 2 weeks) ###

t = p+1 # 1 for each cluster plus baseline

### true log RR in hosp/ER (control,treatment)  ###

theta.s = log(c(0.7,0.8,0.85,0.9,1))

### number of simulations ###

n.sim = 1000

start = Sys.time()

for (i in 1:length(theta.s)){ # loop over scenarios

set.seed(4206)

# sfSetMaxCPUs(number = 32) # need if using more than 32 cores

sfInit(parallel = TRUE, cpus = 10,type = "MPI")

sfLibrary(rjags); sfLibrary(boot)

################### function to loop over for simulation ######################

kevin = function(xxx,d,int = 2){ #int = time interval (2 or 3 weeks)

alpha = 0.1
```

```
### number of clusters/ sample size per cluster ###
p = 50; n = 20
### number of time periods (every 2 weeks) ###
t = p+1 # 1 for each practice plus baseline
## simulated truth for log RR ##
theta.s = log(c(0.7,0.8,0.85,0.9,1))
## expected hosp days/yr per clinic ##
E.yr = rep(n*.36,p) # for each cluster in a year
E.it = E.yr/52  # for each cluster in a 1 wk interval
## number of iterations for inference ##
n_iter = 5000
## treatment indicator variable, 1 = present ##
x = matrix(1,nrow = p,ncol = t)
x[lower.tri(x,diag = T)] = 0
## cluster effects and time trend ##
a = rnorm(p,0,0.5) # a = rep(0,p)
b = log(1:t) – log(t/2) # b = rep(0,t)
### simulating data ###
dat = array(NA,dim = c(p,t))
dat[,1] = rpois(p,exp(log(int*E.it)+rep(b[1], p ) + a ))# baseline, at t = 1: no treatment/device
for (i in 2:t){ # max cluster, p, is last to receive trt
dat[,i] = array(c(rpois((i-1),exp(log(int*E.it[1:(i-1)])+rep(theta.s[d], i-1)+rep(b[i], i-1 ) + a[1:(i-
 1)] )),rpois((p -(i-1)),exp(log(int*E.it[1:(p -(i-1))])+rep(b[i], p -(i-1) ) +a[i:p]))),dim = c(p))
```

```
 #theta.s[d] = 0 for type I error
}
dat.E = matrix(rep(int*E.it,t),nrow = p)
### calling JAGS in R ###
 jags_works = 0; attempt = 1
 while (jags_works == 0){
 jags = try( jags.model(file = "kevin_jags.txt", data = list("t" = t,"p" = p, "y" = dat,"E" =
 dat.E,"x" = x),inits = list("a" = rep(0,p),"b" = rep(0,t),"theta.x" = 0, "prec" = 0.5),quiet=T) )
 jags_works = as.numeric(length(jags) > 1 | attempt == 10)
 attempt = attempt + 1
 }
 update(jags,1000) # more burn-in
 my.samples = coda.samples(jags, variable.names = c("a","b","theta.x"), n.iter = n_iter)
 check.this = coda.samples(jags, variable.names = c("sigma.a"),n.iter = n_iter)
 samples = my.samples[[1]]
 trt.effect = exp(samples[,(p+t+1)])
 l = as.numeric(quantile(samples[,(p+t+1)],c(alpha/2))) # 90% BIC
 u = as.numeric(quantile(samples[,(p+t+1)],c(1-alpha/2)))
 R.h0 = ifelse(u < 0,1,0) # one-sided
 out = c(as.numeric(apply(samples,2,mean)),mean(trt.effect),R.h0,exp(l),exp(u),
 mean(check.this[[1]]))
 return(out)   }
###################### end func ###################################
```

```
sfExport("kevin")

## apply loop over func ##

list.results = sfSapply(1:n.sim,kevin,d = i)

sfStop()

results = matrix(unlist(list.results),ncol = n.sim, nrow = (p+t+6))

} # end loop over simulations

Sys.time() - start

###################### JAGS script ###################################

model{

        for ( j in 1:p ){ # cluster

        for ( k in 1:t ){ # time period

                y[j,k] ~ dpois(lambda[j,k])

                log(lambda[j,k]) <- log(E[j,k]) + a[j] + b[k] + x[j,k]*theta.x

          }                   }

         # priors

        theta.x ~ dnorm(0, 10)

        for (i in 1:p){       # cluster effect

                  a[i] ~ dnorm(0 , prec)          }

        for (i in 1:t){   # fixed effect

                  b[i] ~ dnorm(0 , 1/10)              }

        prec ~ dgamma(.1,1)

        sigma.a <- 1/prec

} # end model
```