Doi: https://doi.org/10.1027/1614-2241/a000180

Program 1
Code for fitting the BITS model without slopes using R and JAGS (BITS1)

```
                                    #text that follows the # symbol are comments and will not be executed
install.packages(runjags)           #this command installs the runjags package
library(runjags)                    #loads runjags into the environment
                                    #y is the dependent variable with 10 observations, 5/phase
y <- c(78, 83, 86, 118, 72, 116, 129, 177, 152, 121)
                                    #the <- symbol means 'gets the value.' This is equivalent to the '=' sign
T <- length(y)                      #T is the total length of y, i.e. 10
P <- 2                              #P is assigned the value 2, i.e. the number of phases
Tb <- 5          #Baseline has 5 observations. Change 5 to any number according to your data
                                    #beta1 is the mean of the first Tb observations and beta2 - the rest
                                    #these betas will be used as starting values in autorunjags
beta1 <- mean(y[1:Tb])
beta2 <- mean(y[(Tb + 1):T])
                                    #Model is defined
BITS.model1 <- "model {
                #yhat for time-point 1 is set to the intercept or the estimated mean of the first phase
              #this is because yhat is the expected value of y and the expected value of y is the mean
  yhat[1] <- beta[1, 1]
                                    #similarly the sixth time-point (i.e. the first time-point of phase 2)
                                    #is assigned to the estimated mean of the second phase
  yhat[(Tb + 1)] <- beta[2, 1]
                                    #for loop begins at 2 and runs till Tb (for baseline phase)
   for (i in 2:Tb) {
     yhat[i] <- beta[1, 1]          #the expected value of yhat in the baseline is beta[1, 1]
                                    #y is drawn from a distribution with expected value yhat
    #and autocorrelation rho which is multiplied by the error of the previous time-point (i - 1)
                                    #tau is the precision = 1/(standard deviation) = 1/sigma
     y[i] ~ dnorm(yhat[i] + rho * (y[i - 1] - yhat[i - 1]), tau)
   }
                                    #for loop begins at Tb+1 and runs till T (for treatment phase)
   for (i in (Tb + 2):T) {
     yhat[i] <- beta[2, 1]
     y[i] ~ dnorm(yhat[i] + rho * (y[i - 1] - yhat[i - 1]), tau)
   }
   y[1] ~ dnorm(yhat[1], tau) #equation 1 for baseline
   y[(Tb + 1)] ~ dnorm(yhat[(Tb + 1)], tau)          #equation 1 for treatment
   es <- (mu[2] – mu[1])/sigma          #standardized mean difference effect size
                                    #Prior specifications
                                    #For both phases
   for (i in 1:P){
                                    #the intercepts for baseline and treatment phases are drawn
```

```
                                        #from distributions with means mu[1] and mu[2], respectively
                                                                                # and precision 0.01
      beta[i, 1] ~ dnorm(mu[i], 0.01)
                                #because mu is the number of vocal responses in five-minute intervals,
        #the expected value is set at 40 with a standard deviation of 20 (precision = 1/20 = .05)
                                        #change the values inside () to reflect your belief or literature
                            #You can set the lower limit of mu to 0 by adding I(0, ) after dnorm(40, .05)
      mu[i] ~ dnorm(40, .05)
    }
                                #Change values inside the parentheses to reflect your belief or literature
  sigma ~ dunif(0.1, 5)                 #standard deviation can uniformly vary between 0.1 to 5
  tau <- pow(sigma, -2)                                                  #tau is precision or sigma^-2
rho ~ dunif(-1, 1)                      #autocorrelation can vary uniformly between -1 and 1
}"                                                                       #end of model definition
                                                                #Begin running the model with the data
results <- autorun.jags(                #autorun.jags runs the model until convergence is indicated
  model = BITS.model1,                                          #the model is BITS.model1 defined above
  data = list(y = y, T = T, P = P, Tb = Tb),          #input data are the y vector of observations,
       #the total number of time-points T, the number of baseline observations Tb, and phases P
  monitor = c("beta", "sigma", "rho", "es"),
                                # parameters to be monitored i.e., checked for convergence and estimated
  n.chains = 4,               #four chains are run. Change this number and observe the trace plot
  startsample = 30000,                                          #run 30000 iterations of each chain
  inits = function() {                                          #initialize/assign starting values
   #change the specs to see how starting values affect the estimates before and after burning-in.
                                                                #Once burned-in they should not.
    list(
      beta = rbind(rnorm(1, beta1, 1), rnorm(1, beta2, 1)),
                                                        #intercept starting values around the phase means
      sigma = runif(1, 0.1, 5),             #standard deviation can be any value between -1 and 5
      rho = runif(1, -1, 1)                                     #autocorrelations between -1 and 1
    )
  },
  method = "rjparallel"                                                 #run the chains in parallel
)                                                                       #end autorun.jags
                                        # combine all chains into a single chain for convenience
results$draws <- combine.mcmc(results$mcmc)
results                                                                 #displays the results
results.tab <- summary(results)                 #summarizes results for plotting convenience
plot(results$mcmc, trace = TRUE, density = TRUE, smooth = FALSE,
     auto.layout = TRUE)                                        #draw density and traceplots

############BLOCK 2 for plotting the intercepts#####################################
lims <- c(min(c(results$draws[,"beta[1,1]"], results$draws[,"beta[2,1]"])),
        max(c(results$draws[,"beta[1,1]"], results$draws[,"beta[2,1]"])))
                                                        #to help set the limits of the plotting area
```

```
x11()                                                    #opens a new plot window
par(mar=c(3.5,3.5,2,1),mgp=c(2,0.7,0), mfcol = c(2, 1))
                                                         #divides the region into 2 rows and 1 column
plot(density(results$draws[,"beta[1,1]"]), main = "Intercept Phase 1",
     xlab = expression(beta[11]), ylab =" ", xlim = lims)        #plots the density of intercept 1
abline(v= results$HPD["beta[1,1]", c(1, 3)], lty = "dashed")
                                                         #adds vertical lines to indicate 95% HDI
plot(density(results$draws[,"beta[2,1]"]), main = "Intercept Phase 2",
     xlab = expression(beta[21]), ylab =" ", xlim = lims)        #plots the density of intercept 2
abline(v= results$HPD["beta[2,1]", c(1, 3)], lty = "dashed")                              #95% HDI
plot(density(results$draws[,"es"]), main = "Effect Size", xlab = " ", ylab =" ")          #plots es
abline(v= results$HPD["es", lty = "dashed")
p.calc <- length(which(results$draws[,"beta[1,1]"] > min(results$draws[,"beta[2,1]"]))) /
          length(results$draws[,"beta[1,1]"])                                        #compute p.calc
p.calc                                                                                #display p.calc
```

Program 2
Code for fitting the BITS model with slopes using R and JAGS (BITS2)

```
library(runjags)
y <- c(85, 85, 83, 83, 82, 83, 84, 85, 84, 84,
       94, 96, 96, 98, 98, 97, 97, 98, 100, 100)
T <- length(y)
P <- 2
Tb <- 10
beta1 <- mean(y[1:Tb])
beta2 <- mean(y[(Tb + 1):T])
                                                        #Model is defined:
BITS.model2 <- "model {
  yhat[1] <- beta[1, 1]
  yhat[Tb+1] <- beta[2, 1]
                                                        #baseline phase
    for (i in 2:Tb) {
                                                #slope beta[1, 2] is added to the equation
      yhat[i] <- beta[1, 1] + beta[1, 2] * i
      y[i] ~ dnorm(yhat[i] + rho * (y[i - 1] - yhat[i - 1]), tau)
    }
                                                        #treatment phase
    for (i in (Tb + 2):T) {
                                        #slope is multiplied by (t-tb) as given in equation 7 such that
                                    #(i-Tb)  = x represents the xth time-point in the intervention phase
      yhat[i] <- beta[2, 1] + beta[2, 2] * i
      y[i] ~ dnorm(yhat[i] + rho * (y[i - 1] - yhat[i - 1]), tau)
    }
    y[1] ~ dnorm(yhat[1], tau)
    y[(Tb + 1)] ~ dnorm(yhat[(Tb + 1)], tau)
                                                        #Prior specifications
    for (i in 1:P){
                                            #the slopes for baseline and treatment phases are drawn
                                    #from distributions with means mu.s[1] and mu.s[2], respectively
                                                        # and precision 0.01
      beta[i, 1] ~ dnorm(mu[i], 0.01)
      beta[i, 2] ~ dnorm(mu.s[i], 0.01)
      mu[i] ~ dnorm(40, 0.5)
      mu.s[i] ~ dnorm(0, 0.1)
    }
    sigma ~ dunif(0.1, 5)
    tau <- pow(sigma, -2)
    rho ~ dunif(-1, 1)
  }"                                                    #end of model definition
                                                    #Begin running the model with the data
results <- autorun.jags(
```

```
  model = BITS.model2,
  data = list(y = y, T = T, P = P, Tb = Tb),
  monitor = c("beta", "sigma", "rho"),
  n.chains = 4,
  startsample = 30000,
  inits = function() {
    list(
      beta = rbind(c(rnorm(1, beta1, 1), 1),
                   c(rnorm(1, beta2, 1), 1)),
      sigma = runif(1, 0.1, 5),
      rho = runif(1, -1, 1)
    )
  },
  method = "rjparallel"
)
results$draws <- combine.mcmc(results$mcmc)
results
results.tab <- summary(results)
p.calc.int <- length(which(results$draws[,"beta[1,1]"] > min(results$draws[,"beta[2,1]"])))/
                length(results$draws[,"beta[1,1]"])
p.calc.slope <- length(which(results$draws[,"beta[1,2]"] > min(results$draws[,"beta[2,2]"])))/
                  length(results$draws[,"beta[2,2]"])
int.lims <- c(min(c(results$draws[,"beta[1,1]"], results$draws[,"beta[2,1]"])),
                max(c(results$draws[,"beta[1,1]"], results$draws[,"beta[2,1]"])))
slope.lims <- c(min(c(results$draws[,"beta[1,2]"], results$draws[,"beta[2,2]"])),
                  max(c(results$draws[,"beta[1,2]"], results$draws[,"beta[2,2]"])))
x11()
par(mar=c(3.5,3.5,2,1),mgp=c(2,0.7,0), mfcol = c(2, 2))
plot(density(results$draws[,"beta[1,1]"]), main = "Intercept Phase 1",
    xlab = expression(beta[11]), ylab =" ", xlim = int.lims)
abline(v= results$HPD["beta[1,1]", c(1, 3)], lty = "dashed")
plot(density(results$draws[,"beta[2,1]"]), main = "Intercept Phase 2",
    xlab = expression(beta[21]), ylab =" ", xlim = int.lims)
abline(v= results$HPD["beta[2,1]", c(1, 3)], lty = "dashed")
plot(density(results$draws[,"beta[1,2]"]), main = "Slope Phase 1",
    xlab = expression(beta[12]), ylab =" ", xlim = slope.lims)
abline(v= results$HPD["beta[1,2]", c(1, 3)], lty = "dashed")
plot(density(results$draws[,"beta[2,2]"]), main = "Slope Phase 2",
    xlab = expression(beta[22]), ylab =" ", xlim = slope.lims)
abline(v= results$HPD["beta[2,2]", c(1, 3)], lty = "dashed")
```

Program 3
Multiple Baseline Design Code for BITS1 Model

```
library(runjags)
                                                    #Model is defined:
BITS.model1.MBD <- "model {
  yhat[1] <- beta[1, 1]
  yhat[(Tb + 1)] <- beta[2, 1]
  for (i in 2:Tb) {
    yhat[i] <- beta[1, 1]
    y[i] ~ dnorm(yhat[i] + rho * (y[i - 1] - yhat[i - 1]), tau)
  }
  for (i in (Tb + 2):T) {
    yhat[i] <- beta[2, 1]
    y[i] ~ dnorm(yhat[i] + rho * (y[i - 1] - yhat[i - 1]), tau)
  }
  y[1] ~ dnorm(yhat[1], tau)
  y[(Tb + 1)] ~ dnorm(yhat[(Tb + 1)], tau)
  es <- (beta[2, 1] - beta[1, 1])/sigma
  for (i in 1:P){
    beta[i, 1] ~ dnorm(mu[i], 0.01)
    mu[i] ~ dnorm(40, .05)
  }
  sigma ~ dunif(0.1, 5)
  tau <- pow(sigma, -2)
  rho ~ dunif(-1, 1)
}"
                                                    # end of model definition
                                    #data is a dataframe with the first, second, and third columns
                                                    #indicating the subject, the values, and the phase
data <- data.frame(cbind(c(rep(1, 10), rep(2, 16), rep(3, 20)),
           c(78, 83, 86, 118, 72, 116, 129, 177, 152, 121,
            70, 91, 110, 101, 96, 95, 134, 94, 106, 138, 123, 155, 135, 134, 164, 115,
            66, 113, 121, 106, 135, 72, 70, 105, 131, 96,
            122, 96, 136, 100, 137, 114, 100, 124, 109, 111),
            c(rep(c(1, 2), each = 5), rep(c(1, 2), each = 8), rep(c(1, 2), each = 10))))
colnames(data) <- c("subject", "DV", "phase")
P <- 2
                                    #change nSubject to reflect the number of subjects in your data
                                                    #change nbase to reflect the baselengths in your data
                                                    #change nTime to reflect the lengths in your data
nSubject <- 3                                       #There are 3 subjects in this dataset
nbase <- c(5, 8, 10)                                #Baseline length for each subject
nTime <- c(10, 16, 20)                              #total length for each subject
pcalc.agg <- matrix(c(1:nSubject, rep(NA, nSubject)), nSubject, 2)
                                                    #empty matrix to store pcalcs
```

```
colnames(pcalc.agg) <- c("subject", "p.calc")
results.agg <- data.frame()
                                                    #begin for loop with one iteration per subject
for (i in 1:nSubject){
  y <- data[data$subject==i,2]
  T <- nTime[i]
  Tb <- nbase[i]
  beta1 <- mean(y[1:Tb])
  beta2 <- mean(y[(Tb + 1):T])
  results <- autorun.jags(
    model = BITS.model1.MBD,
    data = list(y = y, T = T, P = P, Tb = Tb),
    monitor = c("beta", "sigma", "rho", "es"),
    n.chains = 3,
    startsample = 30000,
    inits = function() {
      list(
        beta = rbind(rnorm(1, beta1, 1), rnorm(1, beta2, 1)),
        sigma = runif(1, 0.1, 5),
        rho = runif(1, -1, 1)
      )
    },
    method = "rjparallel"
  )
  results$draws <- combine.mcmc(results$mcmc)
  results.tab <- summary(results)
  pcalc.agg[i,2] <- length(intersect(results$draws[,"beta[1,1]"],
                          results$draws[,"beta[2,1]"]))/length(results$draws[,"beta[1,1]"])
  results.agg <- rbind.data.frame(results.agg, data.frame(cbind("subject" = i, results$HPD,
                              results$summary$statistics)))
}
write.csv(results.agg, 'MBD-results.csv')                    #write the results to a csv file
write.csv(pcalc.agg, 'p-calc-MBD.csv', row.names = FALSE)   #write the p-values to a csv file
```

Program 4
BITS1 Code for ABAB Design

```
library(runjags)
y <- c(78, 83, 86, 118, 72, 116, 129, 177, 152, 121,
       93, 93, 100, 112, 122, 176, 156, 159, 147, 135)
T <- length(y)
P <- 4
Tp <- c(5, 5, 5, 5)                                          #length of each phase
Tt <- c(0, cumsum(Tp))        #cumulative sum tells the time-point when the new phase begins
beta <- c()
for (l in 1:P){
  beta <- c(beta, mean(y[(Tt[l] + 1):Tt[l + 1]]))            #starting values for each phase
}
                                                                          #Model is defined:
BITS.model1.ABAB<- "model {
    for (l in 1:P){                                                  #loop over the 4 phases
      yhat[(Tt[l] + 1)] <- beta[l]
      y[(Tt[l] + 1)] ~ dnorm(yhat[(Tt[l] + 1)], tau)
        for (i in (Tt[l] + 2):Tt[l + 1]){
          yhat[i] <- beta[l]
          y[i] ~ dnorm(yhat[i] + rho * (y[i - 1] - yhat[i - 1]), tau)
        }
        beta[l] ~ dnorm(mu[l], 0.01)
        mu[l] ~ dnorm(40, .05)
    }
   sigma ~ dunif(0.1, 5)
   tau <- pow(sigma, -2)
   rho ~ dunif(-1, 1)
}"                                                                  #end of model definition
                                                          #Begin running the model with the data
results <- autorun.jags(
  model = BITS.model1.ABAB,
  data = list(y = y, Tt = Tt, P = P),
  monitor = c("beta", "sigma", "rho", "es"),
  n.chains = 4,
  startsample = 30000,
  inits = function() {
    list(
      beta = apply(as.matrix(beta), 1, function(x)rnorm(1, x, 1)),
      sigma = runif(1, 0.1, 5),
      rho = runif(1, -1, 1)
    )
  },
  method = "rjparallel"
)
```

```
results$draws <- combine.mcmc(results$mcmc)
results.agg <- data.frame(cbind(results$HPD, results$summary$statistics))
write.csv(results.agg, "ABAB-results.csv")
```