National Institute for Health and Care Excellence

Draft for consultation

Acne vulgaris: management

TSU NMA software code (mild to moderate acne)

NICE guideline tbc

Supplement 3

December 2020

Draft for consultation

Supplementary material was developed by the National Guideline Alliance which is part of the Royal College of Obstetricians and Gynaecologists

ISBN:

Contents

TSU NMA software code (mild to moderate acne)

Efficacy (% change in total lesion count from baseline)

A.1: Efficacy, base-case model (OpenBUGS)

```
# Arm and Trial-level data
# Random effects model for multi-arm trials
# Fixed class effects
model{                              # *** PROGRAM STARTS
for(i in 1:ns.a){                   # LOOP THROUGH STUDIES WITH ARM DATA
   w[i,1] <- 0              # adjustment for multi-arm trials is zero for control arm
   delta[i,1] <- 0          # treatment effect is zero for control arm
   mu[i] ~ dnorm(0,.0001)          # vague priors for all trial baselines
   }

# trials reporting percent CFB
for(i in 1:ns.a1){                    # LOOP THROUGH STUDIES WITH %CFB ARM DATA
   for (k in 1:na[i]) {            #  LOOP THROUGH ARMS
      pCFB.se[i,k] <- pCFB.sd[i,k]/sqrt(n[i,k]) # calculate standard error
      pCFB.var[i,k] <- pow(pCFB.se[i,k],2)   # calculate variances
      pCFB.prec[i,k] <- 1/pCFB.var[i,k]      # set precisions
      pCFB[i,k] ~ dnorm(theta[i,k],pCFB.prec[i,k]) # normal likelihood

      theta[i,k] <- mu[i] + delta[i,k]  # model for linear predictor

      #Deviance contribution
      dev[i,k] <- (pCFB[i,k]-theta[i,k])*(pCFB[i,k]-theta[i,k])*pCFB.prec[i,k]
      }
   resdev[i] <- sum(dev[i,1:na[i]]
   )
```

```
}
# trials reporting CFB + B      # LOOP THROUGH STUDIES WITH CFB+B ARM DATA
for(i in (ns.a1+1):(ns.a1+ns.a2)){
   for (k in 1:na[i]) {         #  LOOP THROUGH ARMS
      x.se[i,k] <- x.sd[i,k]/sqrt(n[i,k])    # calculate standard error
      x.var[i,k] <- pow(x.se[i,k],2)   # calculate variances
      x.prec[i,k] <- 1/x.var[i,k]      # set precisions
      x[i,k] ~ dnorm(mu.X[i,k],x.prec[i,k]) # indpt normal likelihood for baseline mean
      mu.X[i,k] ~ dnorm(0,.0001)       # flat prior for baseline mean in likelihood

      CFB.se[i,k] <- CFB.sd[i,k]/sqrt(n[i,k])     # calculate standard error
      CFB.var[i,k] <- pow(CFB.se[i,k],2)   # calculate variances
      CFB.prec[i,k] <- 1/CFB.var[i,k]      # set precisions
      mu.CFB[i,k] <- mu.X[i,k]*(theta[i,k]/100)# calculate mean for CFB likelihood
      CFB[i,k] ~ dnorm(mu.CFB[i,k],CFB.prec[i,k]) # indpt normal likelihood for baseline mean

      theta[i,k] <- mu[i] + delta[i,k]  # model for linear predictor

      #Deviance contribution
      dev[i,k] <- (CFB[i,k]-mu.CFB[i,k])*(CFB[i,k]-mu.CFB[i,k])*CFB.prec[i,k]
      }
   resdev[i] <- sum(dev[i,1:na[i]]
   )
}
# trials reporting B + F
for(i in (ns.a1+ns.a2+1):ns.a){          # LOOP THROUGH STUDIES WITH B+F ARM DATA
   for (k in 1:na[i]) {           #  LOOP THROUGH ARMS
      #Calculate standard errors
      x.se[i,k] <- x.sd[i,k]/sqrt(n[i,k])
      y.se[i,k] <- y.sd[i,k]/sqrt(n[i,k])
      #Set precision matrix
```

```
        Sigma[i,k,1,1]<-pow(x.se[i,k],2)
        Sigma[i,k,2,2]<-pow(y.se[i,k],2)
        Sigma[i,k,1,2]<-corr[i]*x.se[i,k]*y.se[i,k]
        Sigma[i,k,2,1]<-Sigma[i,k,1,2]
        Prec[i,k,1:2,1:2]<-inverse(Sigma[i,k,1:2,1:2])
        #Set up vector for baseline and follow-up means
        y.XY[i,k,1]<-x[i,k]
        y.XY[i,k,2]<-y[i,k]

        # Bivariate normal likelihood for baseline and follow-up
        y.XY[i,k,1:2]~dmnorm(mu.XY[i,k,1:2],Prec[i,k,1:2,1:2])
        mu.XY[i,k,2]<- mu.XY[i,k,1]*(1-(theta[i,k]/100))
        mu.XY[i,k,1] ~ dnorm(0,.0001)       # flat prior for baseline mean in likelihood

        theta[i,k] <- mu[i] + delta[i,k]  # model for linear predictor

        #Deviance contribution
        for (j in 1:2){
           diff[i,k,j]<- y.XY[i,k,j]-mu.XY[i,k,j]
           z[i,k,j]<- inprod(Prec[i,k,j,1:2],diff[i,k,1:2])
           }
        dev[i,k]<-inprod(diff[i,k,1:2],z[i,k,1:2])
        }
    resdev[i] <- sum(dev[i,1:na[i]])
    }
# 2-arm trials reporting contrasts (e.g., split-face trials)
for(i in (ns.a+1):(ns.a+ns.t2)){       # LOOP THROUGH STUDIES WITH TRIAL DATA
    w[i,1] <- 0              # adjustment for multi-arm trials is zero for control arm
    delta[i,1] <- 0         # treatment effect is zero for control arm
    var[i,2] <- pow(se.T[i,2],2)   # calculate variances
    prec[i,2] <- 1/var[i,2]      # set precisions
```

```
  y.T[i,2] ~ dnorm(delta[i,2],prec[i,2]) # normal likelihood
  # Deviance contribution
  dev[i,2] <- (y.T[i,2]-delta[i,2])* (y.T[i,2]-delta[i,2])* prec[i,2]
  # summed residual deviance contribution for this trial
  resdev[i] <- dev[i,2]
  }
#RE Model (ARM AND TRIAL DATA)
for(i in 1:ns){                    # LOOP THROUGH STUDIES WITH ARM DATA
  for (k in 2:na[i]) {             # LOOP THROUGH ARMS
    # trial-specific RE distributions
    delta[i,k] ~ dnorm(md[i,k],taud[i,k])
    # mean of RE distributions, with multi-arm trial correction
    md[i,k] <-  d[t[i,k]] - d[t[i,1]] + sw[i,k]
    # precision of RE distributions (with multi-arm trial correction)
    taud[i,k] <- tau *2*(k-1)/k
    # adjustment, multi-arm RCTs
    w[i,k] <- (delta[i,k] - d[t[i,k]] + d[t[i,1]])
    # cumulative adjustment for multi-arm trials
    sw[i,k] <- sum(w[i,1:k-1])/(k-1)
   }
}

totresdev <- sum(resdev[])           #Total Residual Deviance
# Reference treatment currently Placebo (ref=1)
d[ref]<-0      # treatment effect is zero for reference treatment
D[class[ref]]<-0
# priors for mean class effect
for (j in 2:nc){
  D[j]~dnorm(0,.0001)
  }
# treatment effect = mean class effect
```

```
for (j in 2:nt){
   d[j] <- D[class[j]]
   }
#
sd ~ dunif(0,25)     # vague prior for between-trial SD
tau <- pow(sd,-2)   # between-trial precision = (1/between-trial variance)
#
# pairwise mean differences for all possible pair-wise comparisons
for (c in 1:(nt-1)) {
  for (k in (c+1):nt)  { mean.diff[c,k] <- d[k]-d[c] }
   }
# pairwise differences for classes
for (c in 1:(nc-1)){
   for (k in (c+1):nc){
      diffClass[c,k] <- D[k] - D[c]
      }
   }
# rank all classes
# ranking on relative scale
for (k in 1:nc){
   # rk[k] <- rank(D[],k)           # assumes lower values are "good"
   rk[k] <- nc+1-rank(D[],k)           # assumes higher values are "good"
   best[k] <- equals(rk[k],1)      #calculate probability that treat k is best
   # calculate probability that treat k is h-th best
   for (h in 1:nc){ prob[h,k] <- equals(rk[k],h) }
   }
# ranking on relative scale - males
for (k in 1:18){ D.m[k] <- D[k]}
for (k in 19:(nc-2)){ D.m[k] <- D[k+2]}
for (k in 1:(nc-2)){
   rk.m[k] <- (nc-2)+1-rank(D.m[],k)           # assumes higher values are "good"
```

```
    best.m[k] <- equals(rk.m[k],1)      #calculate probability that treat k is best
    # calculate probability that treat k is h-th best
    for (h in 1:nc){ prob.m[h,k] <- equals(rk.m[k],h) }
    }
}                                   # *** PROGRAM ENDS
```

A.2: Efficacy, bias-adjusted model: small study effects (OpenBUGS)

```
# Arm and Trial-level data
# Random effects model for multi-arm trials
# Fixed class effects
model{                              # *** PROGRAM STARTS
for(i in 1:ns) {
        Nsum[i]<- sum(n[i,1:na[i]])
}
for(i in 1:ns.a){                   # LOOP THROUGH STUDIES WITH ARM DATA
    w[i,1] <- 0              # adjustment for multi-arm trials is zero for control arm
    delta[i,1] <- 0            # treatment effect is zero for control arm
    beta[i,1] <- 0            # No bias on baseline arm
    mu[i] ~ dnorm(0,.0001)          # vague priors for all trial baselines
    }

# trials reporting percent CFB
for(i in 1:ns.a1){                     # LOOP THROUGH STUDIES WITH %CFB ARM DATA
    for (k in 1:na[i]) {            #  LOOP THROUGH ARMS
        pCFB.se[i,k] <- pCFB.sd[i,k]/sqrt(n[i,k]) # calculate standard error
        pCFB.var[i,k] <- pow(pCFB.se[i,k],2)   # calculate variances
        pCFB.prec[i,k] <- 1/pCFB.var[i,k]      # set precisions
        pCFB[i,k] ~ dnorm(theta[i,k],pCFB.prec[i,k]) # normal likelihood

        theta[i,k] <- mu[i] + delta[i,k] + beta[i,k]*X[i,k]/sqrt(Nsum[i])  # model for linear predictor
```

```

     #Deviance contribution
     dev[i,k] <- (pCFB[i,k]-theta[i,k])*(pCFB[i,k]-theta[i,k])*pCFB.prec[i,k]
     }
  resdev[i] <- sum(dev[i,1:na[i]]
  )
}
# trials reporting CFB + B      # LOOP THROUGH STUDIES WITH CFB+B ARM DATA
for(i in (ns.a1+1):(ns.a1+ns.a2)){
  for (k in 1:na[i]) {          #  LOOP THROUGH ARMS
     x.se[i,k] <- x.sd[i,k]/sqrt(n[i,k])    # calculate standard error
     x.var[i,k] <- pow(x.se[i,k],2)   # calculate variances
     x.prec[i,k] <- 1/x.var[i,k]      # set precisions
     x[i,k] ~ dnorm(mu.X[i,k],x.prec[i,k]) # indpt normal likelihood for baseline mean
     mu.X[i,k] ~ dnorm(0,.0001)       # flat prior for baseline mean in likelihood

     CFB.se[i,k] <- CFB.sd[i,k]/sqrt(n[i,k])     # calculate standard error
     CFB.var[i,k] <- pow(CFB.se[i,k],2)   # calculate variances
     CFB.prec[i,k] <- 1/CFB.var[i,k]      # set precisions
     mu.CFB[i,k] <- mu.X[i,k]*(theta[i,k]/100)# calculate mean for CFB likelihood
     CFB[i,k] ~ dnorm(mu.CFB[i,k],CFB.prec[i,k]) # indpt normal likelihood for baseline mean

     theta[i,k] <- mu[i] + delta[i,k] + beta[i,k]*X[i,k]/sqrt(Nsum[i])  # model for linear predictor

     #Deviance contribution
     dev[i,k] <- (CFB[i,k]-mu.CFB[i,k])*(CFB[i,k]-mu.CFB[i,k])*CFB.prec[i,k]
     }
  resdev[i] <- sum(dev[i,1:na[i]]
  )
}
# trials reporting B + F
```

```
for(i in (ns.a1+ns.a2+1):ns.a){          # LOOP THROUGH STUDIES WITH B+F ARM DATA
  for (k in 1:na[i]) {            #  LOOP THROUGH ARMS
    #Calculate standard errors
    x.se[i,k] <- x.sd[i,k]/sqrt(n[i,k])
    y.se[i,k] <- y.sd[i,k]/sqrt(n[i,k])
    #Set precision matrix
    Sigma[i,k,1,1]<-pow(x.se[i,k],2)
    Sigma[i,k,2,2]<-pow(y.se[i,k],2)
    Sigma[i,k,1,2]<-corr[i]*x.se[i,k]*y.se[i,k]
    Sigma[i,k,2,1]<-Sigma[i,k,1,2]
    Prec[i,k,1:2,1:2]<-inverse(Sigma[i,k,1:2,1:2])
    #Set up vector for baseline and follow-up means
    y.XY[i,k,1]<-x[i,k]
    y.XY[i,k,2]<-y[i,k]

    # Bivariate normal likelihood for baseline and follow-up
    y.XY[i,k,1:2]~dmnorm(mu.XY[i,k,1:2],Prec[i,k,1:2,1:2])
    mu.XY[i,k,2]<- mu.XY[i,k,1]*(1-(theta[i,k]/100))
    mu.XY[i,k,1] ~ dnorm(0,.0001)       # flat prior for baseline mean in likelihood

    theta[i,k] <- mu[i] + delta[i,k] + beta[i,k]*X[i,k]/sqrt(Nsum[i]) # model for linear predictor

    #Deviance contribution
    for (j in 1:2){
      diff[i,k,j]<- y.XY[i,k,j]-mu.XY[i,k,j]
      z[i,k,j]<- inprod(Prec[i,k,j,1:2],diff[i,k,1:2])
      }
    dev[i,k]<-inprod(diff[i,k,1:2],z[i,k,1:2])
    }
  resdev[i] <- sum(dev[i,1:na[i]])
  }
```

```
# 2-arm trials reporting contrasts (e.g., split-face trials)
for(i in (ns.a+1):(ns.a+ns.t2)){       # LOOP THROUGH STUDIES WITH TRIAL DATA
   w[i,1] <- 0              # adjustment for multi-arm trials is zero for control arm
   delta[i,1] <- 0          # treatment effect is zero for control arm
   var[i,2] <- pow(se.T[i,2],2)   # calculate variances
   prec[i,2] <- 1/var[i,2]      # set precisions
   y.T[i,2] ~ dnorm(theta[i,2],prec[i,2]) # normal likelihood
   theta[i,2] <- delta[i,2] + beta[i,2]*X[i,2]/sqrt(Nsum[i])        # model for linear predictor.
   # Deviance contribution
   dev[i,2] <- (y.T[i,2]-theta[i,2])* (y.T[i,2]-theta[i,2])* prec[i,2]
   # summed residual deviance contribution for this trial
   resdev[i] <- dev[i,2]
   }
#RE Model (ARM AND TRIAL DATA)
for(i in 1:ns){                  # LOOP THROUGH STUDIES WITH ARM DATA
   for (k in 2:na[i]) {            # LOOP THROUGH ARMS
      # trial-specific RE distributions
      delta[i,k] ~ dnorm(md[i,k],taud[i,k])
      # model for bias parameter beta
      beta[i,k] ~ dnorm(b, prec.b)
      # mean of RE distributions, with multi-arm trial correction
      md[i,k] <-  d[t[i,k]] - d[t[i,1]] + sw[i,k]
      # precision of RE distributions (with multi-arm trial correction)
      taud[i,k] <- tau *2*(k-1)/k
      # adjustment, multi-arm RCTs
      w[i,k] <- (delta[i,k] - d[t[i,k]] + d[t[i,1]])
      # cumulative adjustment for multi-arm trials
      sw[i,k] <- sum(w[i,1:k-1])/(k-1)
    }
}

```

```
totresdev <- sum(resdev[])           #Total Residual Deviance
# Reference treatment currently Placebo (ref=1)
d[ref]<-0       # treatment effect is zero for reference treatment
D[class[ref]]<-0
# priors for mean class effect
for (j in 2:nc){
   D[j]~dnorm(0,.0001)
   }
# treatment effect = mean class effect
for (j in 2:nt){
   d[j] <- D[class[j]]
   }
#
sd ~ dunif(0,25)     # vague prior for between-trial SD
tau <- pow(sd,-2)   # between-trial precision = (1/between-trial variance)
# bias model prior for variance
sd.b ~ dunif(0,1000)
prec.b <- pow(sd.b,-2)
# bias model prior for mean
b ~ dnorm(0,.0001)
#
# pairwise mean differences for all possible pair-wise treatment comparisons
for (c in 1:(nt-1)) {
  for (k in (c+1):nt)  { mean.diff[c,k] <- d[k]-d[c] }
   }
# pairwise differences for classes
for (c in 1:(nc-1)){
   for (k in (c+1):nc){
      diffClass[c,k] <- D[k] - D[c]
      }
   }
```

```
#Adjusted estimates for n = 1670
diffClass1670[1,1] <- 0
diffClass1670[1,2] <- diffClass[1,2]
for (k in 3:nc){
   diffClass1670[1,k] <- diffClass[1,k] + b/sqrt(1670)
   diffClass1670[2,k] <- diffClass[2,k] + b/sqrt(1670)
   }
for (c in 3:(nc-1)){
   for (k in (c+1):nc){
      diffClass1670[c,k] <- diffClass[c,k]
      }
   }
# rank all classes
# ranking on relative scale
for (k in 1:nc){
   # rk[k] <- rank(diffClass1670[1,],k)      # assumes lower values are "good"
   rk[k] <- nc+1-rank(diffClass1670[1,],k)   # assumes higher values are "good"
   best[k] <- equals(rk[k],1)      #calculate probability that treat k is best
   # calculate probability that treat k is h-th best
   for (h in 1:nc){ prob[h,k] <- equals(rk[k],h) }
   }
# ranking on relative scale - males
for (k in 1:18){ D.m[k] <- diffClass1670[1,k]}
for (k in 19:(nc-2)){ D.m[k] <- diffClass1670[1,k+2]}
for (k in 1:(nc-2)){
   rk.m[k] <- (nc-2)+1-rank(D.m[],k)           # assumes higher values are "good"
   best.m[k] <- equals(rk.m[k],1)      #calculate probability that treat k is best
   # calculate probability that treat k is h-th best
   for (h in 1:nc){ prob.m[h,k] <- equals(rk.m[k],h) }
   }
}                              # *** PROGRAM ENDS
```

A.3: Efficacy, node-splitting, class-level

A.3.1: R Code (requires R2OpenBUGS package)

```
##################################################################################
#
# Node-splitting for Acne Guideline - Efficacy at Class Level
# R script to run node-split for the MTC Random study effects, fixed
# class effects model using OpenBUGS
#
# Uses R2OpenBUGS package
#
# EFficacy
#  1. Need to include in the working directory the following files:
#       efficacy_class.txt --- text file with data
#       rse fce node-splitR2_v2_efficacy_class.txt --- text file holding BUGS code
#
#   2. Output files will be
#       data.txt --- holds all data as used by BUGS
#       log.odc and log.txt --- hold WinBUGS output
#       inits1.txt --- holds initial values as read by BUGS
#       script.txt --- BUGS script file with all commands to execute
#
#   3. Output files for each node should be transferred to a new directory
#     as they will be overwritten in each new run
#
#   4. You may need to edit the OpenBUGS location 'bd'
#
#   5. You will need to edit the working directory 'pathname'
#     to suit your computer settings
#
#   6. Run script file
#
```

```
##########################################################################################
#
# Declare the directory where OpenBUGS is found in this computer
bd <- "C:/Program Files (x86)/OpenBUGS/OpenBUGS323/OpenBUGS.exe"
#
# Declare working directory
pathname <- "C:/Acne/M2M/Efficacy"
setwd(pathname)
#
# load package to call OpenBUGS
library(R2OpenBUGS)
#
# LOAD DATA MANIPULATING FUNCTIONS:
#
PairXY <- function(treat, na, pair)
  # Check if pair(X,Y) in row i of data
  # and reorder treatments in trial as appropriate
{
  N <- nrow(treat)
  multi <- rep(NA,length(na))
  split.ind <- matrix(nrow=nrow(treat),ncol=ncol(treat))
  split.ind1 <- matrix(nrow=nrow(treat),ncol=ncol(treat))
  split.ind2 <- matrix(nrow=nrow(treat),ncol=ncol(treat))
  spliti <- rep(NA,length(na))
  split1i <- rep(NA,length(na))
  split2i <- rep(NA,length(na))
  pair1 <- matrix(nrow=nrow(treat),ncol=ncol(treat))
  pair2 <- matrix(nrow=nrow(treat),ncol=ncol(treat))
  k.ind <- matrix(nrow=nrow(treat),ncol=ncol(treat))
  for (i in 1:N) {
    # is trial i a multiarm trial?
```

```
multi[i] <- 1*(na[i]>2)
for (k in 1:na[i]){
  # which arms contain a treatment in the pair?
  split.ind[i,k] <- 1*(treat[i,k]==pair[1])+1*(treat[i,k]==pair[2])
  # which arms contain the treatment in pair[1]?
  split.ind1[i,k] <- 1*(treat[i,k]==pair[1])
  # which arms contain the treatment in pair[2]?
  split.ind2[i,k] <- 1*(treat[i,k]==pair[2])
}
# does trial i contain multiples of pair[1]?
split1i[i] <- 1*(sum(split.ind1[i,1:na[i]])>1)
# does trial i contain multiples of pair[2]?
split2i[i] <- 1*(sum(split.ind2[i,1:na[i]])>1)
# does trial i contain both treatments in the pair?
# (minus duplicates in multiarm trials that have one treatment (only) in pair)
spliti[i] <- 1*((sum(split.ind[i,1:na[i]])-split1i[i]*(sum(split.ind1[i,1:na[i]])-split1i[i])-
split2i[i]*(sum(split.ind2[i,1:na[i]])-split2i[i]))>1)
for (k in 1:na[i]) {
  # which arms contain the first element in the pair
  pair1[i,k] <- k*(1*(treat[i,k]==pair[1]))
  # which arms contain the second element in the pair
  pair2[i,k] <- k*(1*(treat[i,k]==pair[2]))
}
for (k in 1:na[i]) {
  # reposition order of arms within a trial according to node being split
  # k.ind ensures a treatment in the pair is in the baseline arm, where the
  # multi-arm trial contains both treatments in the pair


  # multi-arm trial contains both treatments in the pair
  # If a multi-arm trial does not contain the node, arm order stays the same
  k.ind[i,k]<-(k*((1-multi[i])+multi[i]*(1-spliti[i])+multi[i]*spliti[i]*(1*(split.ind[i,1]==1)))
        # If a multi-arm trial contains the node, the treatment in pair[1] is not duplicated in trial,
        # the baseline arm does not contain a treatment in the node, and the treatment
        # in arm k is pair[1], make this treatment baseline treatment
        + multi[i]*spliti[i]*(1-split1i[i])*(1-(1*(split.ind[i,1]==1)))*(1*(treat[i,k]==pair[1]))

        # If a multi-arm trial contains the node, the treatment in pair[1] is duplicated in trial,
        # the treatment in pair[2] is not duplicated in trial, the baseline arm does not contain
        # a treatment in the node, and the treatment in arm k is pair[2], make this treatment
baseline treatment
        + multi[i]*spliti[i]*split1i[i]*(1-split2i[i])*(1-(1*(split.ind[i,1]==1)))*(1*(treat[i,k]==pair[2]))

        # If a multi-arm trial contains the node, the treatment in pair[1] is not duplicated in trial,
        # the baseline arm does not contain a treatment in the node, and k is baseline arm,
        # move treatment to come after baseline treatment
        + sum(pair1[i,1:na[i]])*(1-split1i[i])*multi[i]*spliti[i]*(1-(1*(split.ind[i,1]==1)))*(1*(k==1))

        # If a multi-arm trial contains the node, the treatment in pair[1] is duplicated in trial,
        # the treatment in pair[2] is not duplicated in trial, the baseline arm does not contain a
        # treatment in the node, and k is baseline arm, move treatment to come after baseline
treatment
        + sum(pair2[i,1:na[i]])*split1i[i]*(1-split2i[i])*multi[i]*spliti[i]*(1-
(1*(split.ind[i,1]==1)))*(1*(k==1))

        # If a multi-arm trial contains the node, the treatment in pair[1] are not duplicated in
trial,
        # the baseline arm does not contain a treatment in the node, k is NOT baseline arm,
        # and treatment in arm k is NOT pair[1], arm order stays the same
        + k*multi[i]*spliti[i]*(1-split1i[i])*(1-(1*(split.ind[i,1]==1)))*(1-(1*(k==1)))*(1-
(1*(treat[i,k]==pair[1])))

        # If a multi-arm trial contains the node, the treatment in pair[1] is duplicated in trial,
        # the treatment in pair[2] is not duplicated in trial, the baseline arm does not contain a treatment in the node, k is NOT baseline arm,
        # and treatment in arm k is NOT pair[2], arm order stays the same
        + k*multi[i]*spliti[i]*split1i[i]*(1-split2i[i])*(1-(1*(split.ind[i,1]==1)))*(1-(1*(k==1)))*(1-(1*(treat[i,k]==pair[2]))))
    }
  }
  k.ind
}
################################################################################
#
# load data for MTC
MTCData <- read.table("efficacy_class.txt", header=TRUE)
n <- data.matrix(MTCData[,c("n1", "n2", "n3", "n4", "n5")])
x <- data.matrix(MTCData[,c("x1", "x2", "x3", "x4", "x5")])
x.sd <- data.matrix(MTCData[,c("x.sd1", "x.sd2", "x.sd3", "x.sd4", "x.sd5")])
y <- data.matrix(MTCData[,c("y1", "y2", "y3", "y4")])
y.sd <- data.matrix(MTCData[,c("y.sd1", "y.sd2", "y.sd3", "y.sd4")])
CFB <- data.matrix(MTCData[,c("CFB1", "CFB2", "CFB3", "CFB4", "CFB5")])
CFB.sd <- data.matrix(MTCData[,c("CFB.sd1", "CFB.sd2", "CFB.sd3", "CFB.sd4", "CFB.sd5")])
pCFB <- data.matrix(MTCData[,c("pCFB1", "pCFB2", "pCFB3", "pCFB4")])
pCFB.sd <- data.matrix(MTCData[,c("pCFB.sd1", "pCFB.sd2", "pCFB.sd3", "pCFB.sd4")])
y.T <- data.matrix(cbind(rep(NA,length(n[,1])),MTCData[,c("y.T2")]))
se.T <- data.matrix(cbind(rep(NA,length(n[,1])),MTCData[,c("se.T2")]))
corr <- data.matrix(MTCData[,"corr"])
c <- data.matrix(MTCData[,c("c1", "c2", "c3", "c4", "c5")])
na <- data.matrix(MTCData[,"na"])
#Class when running model at class level
class <- 1:max(c,na.rm = TRUE)
nt <- max(c, na.rm=TRUE)
nc <- max(class)
```

```
ns <- nrow(n)
ns.a <- 84  #studies reporting arm-level data
ns.a1 <- 38   #pCFB studies
ns.a2 <- 13   #CFB studies
ns.t2 <- 6    #2-arm studies reporting contrasts
ref <- 1    #reference treatment
#
initv1 <- list(direct=0,  D=c(NA,rep(0,nc-1)), mu=rep(0,ns.a), sd=1)
initv2 <- list(direct=0.05,  D=c(NA,rep(-1.2,nc-1)), mu=rep(0.5,ns.a), sd=3)
###############################################################################
#
# Check which notes to split
#
library(gemtc)
ns.data<-mtc.data.studyrow(MTCData,
                        armVars=c('treatment'='c'),
                        nArmsVar='na',
                        studyVars=c(),
                        studyNames=MTCData$studyid,
                        treatmentNames=NA,
                        patterns=c('%s', '%s%d'))
net<-mtc.network(data.ab=ns.data,description="Efficacy_trt")
## Print which nodes to split
splitcomps<-mtc.nodesplit.comparisons(net)
print(splitcomps)
#
###############################################################################
#  NODE-SPLITTING ROUTINE
###############################################################################
#
#
```

```
# Define nodes to split
pair<-splitcomps
pair
# Run node split models
for(j in 1:length(pair[,1])){
  print(pair[j,])

  k.ind <- PairXY(treat=c,na=na[,1],pair=as.numeric(pair[j,]))

  # Setup subdirectory to hold results for each node-split
  dir.create(paste("REFCEnode",pair[j,1],"_",pair[j,2],sep=""))

  # Build data file: stored in the working directory as "data.txt"
  bugs.data(list("n"=n,"x"=x,"x.sd"=x.sd,"y"=y,"y.sd"=y.sd,
                 "CFB"=CFB,"CFB.sd"=CFB.sd,"pCFB"=pCFB,"pCFB.sd"=pCFB.sd,
                 "y.T"=y.T,"se.T"=se.T,"corr" = corr[,1],
                 "t"=c, "class"=class,
                 "na" = na[,1], "ns.a" = ns.a, "ns.a1" = ns.a1, "ns.a2" = ns.a2,
                 "nt" = nt, "nc" = nc, "ns" = ns, "ns.t2" = ns.t2,
                 "ref" = ref, "pair" = as.numeric(pair[j,]), "k.ind" = k.ind) )

  # Call OpenBUGS
  #
  bugs(data = "data.txt",
      inits = list(initv1,initv2),
      #inits = list(initv1),
      parameters.to.save = c("direct", "d", "prob","totresdev","indirect","sd"),
      model.file = "rse fce node-splitR2_v2_efficacy_class.txt",
      n.chains = 2,
      n.iter = 120000,
      n.burnin = 40000,
    n.thin = 1,
    OpenBUGS.pgm = bd,
    debug = FALSE,
    save.history = TRUE,
    useWINE=FALSE)
 #
 # Copy input and output files to relevant directory
 file.copy("data.txt", paste("REFCEnode",pair[j,1],"_",pair[j,2],"/data.txt",sep=""),
overwrite=TRUE)
 file.copy(paste(tempdir(),"/log.odc",sep=""),
paste(pathname,"/REFCEnode",pair[j,1],"_",pair[j,2],"/log.odc",sep=""), overwrite=TRUE)
 file.copy(paste(tempdir(),"/log.txt",sep=""),
paste(pathname,"/REFCEnode",pair[j,1],"_",pair[j,2],"/log.txt",sep=""), overwrite=TRUE)
 file.copy(paste(tempdir(),"/inits1.txt",sep=""),
paste(pathname,"/REFCEnode",pair[j,1],"_",pair[j,2],"/inits1.txt",sep=""), overwrite=TRUE)
 file.copy(paste(tempdir(),"/script.txt",sep=""),
paste(pathname,"/REFCEnode",pair[j,1],"_",pair[j,2],"/script.txt",sep=""), overwrite=TRUE)
 #
 # REPEAT FOR ALL OTHER NODES
}
```

A.3.2 OpenBUGS Code

```
model{                          # *** PROGRAM STARTS
for(i in 1:ns.a){                # LOOP THROUGH STUDIES WITH ARM DATA
   w[i,1] <- 0             # adjustment for multi-arm trials is zero for control arm
   delta[i,1] <- 0           # treatment effect is zero for control arm
   mu[i] ~ dnorm(0,.0001)          # vague priors for all trial baselines
   }

# trials reporting percent CFB
for(i in 1:ns.a1){                   # LOOP THROUGH STUDIES WITH %CFB ARM DATA
   for (k in 1:na[i]) {            #  LOOP THROUGH ARMS
      pCFB.se[i,k.ind[i,k]] <- pCFB.sd[i,k.ind[i,k]]/sqrt(n[i,k.ind[i,k]])  # calculate standard error
      pCFB.var[i,k.ind[i,k]] <- pow(pCFB.se[i,k.ind[i,k]],2)   # calculate variances
      pCFB.prec[i,k.ind[i,k]] <- 1/pCFB.var[i,k.ind[i,k]]      # set precisions
      pCFB[i,k.ind[i,k]] ~ dnorm(theta[i,k],pCFB.prec[i,k.ind[i,k]]) # normal likelihood

      theta[i,k] <- mu[i] + delta[i,k]  # model for linear predictor

      #Deviance contribution
      dev[i,k] <- (pCFB[i,k.ind[i,k]]-theta[i,k])*(pCFB[i,k.ind[i,k]]-
theta[i,k])*pCFB.prec[i,k.ind[i,k]]

      split[i,k] <- equals(t[i,k.ind[i,1]], pair[1]) * equals(t[i,k.ind[i,k]], pair[2]) -
equals(t[i,k.ind[i,1]], pair[2]) * equals(t[i,k.ind[i,k]], pair[1])
      }
   resdev[i] <- sum(dev[i,1:na[i]])
}
# trials reporting CFB + B       # LOOP THROUGH STUDIES WITH CFB+B ARM DATA
for(i in (ns.a1+1):(ns.a1+ns.a2)){
   for (k in 1:na[i]) {           #  LOOP THROUGH ARMS
      x.se[i,k.ind[i,k]] <- x.sd[i,k.ind[i,k]]/sqrt(n[i,k.ind[i,k]])     # calculate standard error
      x.var[i,k.ind[i,k]] <- pow(x.se[i,k.ind[i,k]],2)   # calculate variances
      x.prec[i,k.ind[i,k]] <- 1/x.var[i,k.ind[i,k]]      # set precisions
      x[i,k.ind[i,k]] ~ dnorm(mu.X[i,k],x.prec[i,k.ind[i,k]]) # indpt normal likelihood for baseline
mean
      mu.X[i,k] ~ dnorm(0,.0001)I(0,)                    # flat prior for baseline mean in likelihood

      CFB.se[i,k.ind[i,k]] <- CFB.sd[i,k.ind[i,k]]/sqrt(n[i,k.ind[i,k]])      # calculate standard error
      CFB.var[i,k.ind[i,k]] <- pow(CFB.se[i,k.ind[i,k]],2)   # calculate variances
      CFB.prec[i,k.ind[i,k]] <- 1/CFB.var[i,k.ind[i,k]]      # set precisions
      mu.CFB[i,k] <- mu.X[i,k]*(theta[i,k]/100)# calculate mean for CFB likelihood
      CFB[i,k.ind[i,k]] ~ dnorm(mu.CFB[i,k],CFB.prec[i,k.ind[i,k]]) # indpt normal likelihood for
baseline mean

      theta[i,k] <- mu[i] + delta[i,k]  # model for linear predictor
```

```

         #Deviance contribution
       dev[i,k] <- (CFB[i,k.ind[i,k]]-mu.CFB[i,k])*(CFB[i,k.ind[i,k]]-
mu.CFB[i,k])*CFB.prec[i,k.ind[i,k]]
       split[i,k] <- equals(t[i,k.ind[i,1]], pair[1]) * equals(t[i,k.ind[i,k]], pair[2]) -
equals(t[i,k.ind[i,1]], pair[2]) * equals(t[i,k.ind[i,k]], pair[1])
       }
   resdev[i] <- sum(dev[i,1:na[i]])
   }
# trials reporting B + F
for(i in (ns.a1+ns.a2+1):ns.a){           # LOOP THROUGH STUDIES WITH B+F ARM DATA
   for (k in 1:na[i]) {            #  LOOP THROUGH ARMS
       #Calculate standard errors
       x.se[i,k.ind[i,k]] <- x.sd[i,k.ind[i,k]]/sqrt(n[i,k.ind[i,k]])
       y.se[i,k.ind[i,k]] <- y.sd[i,k.ind[i,k]]/sqrt(n[i,k.ind[i,k]])
       #Set precision matrix
       Sigma[i,k.ind[i,k],1,1]<-pow(x.se[i,k.ind[i,k]],2)
       Sigma[i,k.ind[i,k],2,2]<-pow(y.se[i,k.ind[i,k]],2)
       Sigma[i,k.ind[i,k],1,2]<-corr[i]*x.se[i,k.ind[i,k]]*y.se[i,k.ind[i,k]]
       Sigma[i,k.ind[i,k],2,1]<-Sigma[i,k.ind[i,k],1,2]
       Prec[i,k.ind[i,k],1:2,1:2]<-inverse(Sigma[i,k.ind[i,k],1:2,1:2])
       #Set up vector for baseline and follow-up means
       y.XY[i,k.ind[i,k],1]<-x[i,k.ind[i,k]]
       y.XY[i,k.ind[i,k],2]<-y[i,k.ind[i,k]]

       # Bivariate normal likelihood for baseline and follow-up
       y.XY[i,k.ind[i,k],1:2]~dmnorm(mu.XY[i,k,1:2],Prec[i,k.ind[i,k],1:2,1:2])
       mu.XY[i,k,2]<- mu.XY[i,k,1]*(1-(theta[i,k]/100))
       mu.XY[i,k,1] ~ dnorm(0,.0001)I(0,)                 # flat prior for baseline mean in likelihood

       theta[i,k] <- mu[i] + delta[i,k]  # model for linear predictor

```

```
        #Deviance contribution
        for (j in 1:2){
                    diff[i,k,j]<- y.XY[i,k.ind[i,k],j]-mu.XY[i,k,j]
                    z[i,k,j]<- inprod(Prec[i,k.ind[i,k],j,1:2],diff[i,k,1:2])
                    }
        dev[i,k]<-inprod(diff[i,k,1:2],z[i,k,1:2])

        split[i,k] <- equals(t[i,k.ind[i,1]], pair[1]) * equals(t[i,k.ind[i,k]], pair[2]) -
equals(t[i,k.ind[i,1]], pair[2]) * equals(t[i,k.ind[i,k]], pair[1])
        }
    resdev[i] <- sum(dev[i,1:na[i]])
}
# 2-arm trials reporting contrasts (e.g., split-face trials)
for(i in (ns.a+1):(ns.a+ns.t2)){       # LOOP THROUGH STUDIES WITH TRIAL DATA
    w[i,1] <- 0              # adjustment for multi-arm trials is zero for control arm
    delta[i,1] <- 0          # treatment effect is zero for control arm
    var[i,2] <- pow(se.T[i,2],2)   # calculate variances
    prec[i,2] <- 1/var[i,2]      # set precisions
    y.T[i,2] ~ dnorm(delta[i,2],prec[i,2]) # normal likelihood
    #Deviance contribution
    dev[i,2] <- (y.T[i,2]-delta[i,2])* (y.T[i,2]-delta[i,2])* prec[i,2]
    split[i,2] <- equals(t[i,1], pair[1]) * equals(t[i,2], pair[2]) - equals(t[i,1], pair[2]) * equals(t[i,2],
pair[1])

    #  summed residual deviance contribution for this trial
    resdev[i] <- dev[i,2]
    }
#RE Model (ARM AND TRIAL DATA)
for(i in 1:ns){                  # LOOP THROUGH STUDIES WITH ARM DATA
    for (k in 2:na[i]) {           # LOOP THROUGH ARMS
            # trial-specific RE distributions
            delta[i,k] ~ dnorm(md[i,k],taud[i,k])
```

```
          # mean of RE distributions, with multi-arm trial correction
          md[i,k] <-  (d[t[i,k.ind[i,k]]] - d[t[i,k.ind[i,1]]])*(1-abs(split[i,k])) + direct * split[i,k] + sw[i,k]
          # precision of RE distributions (with multi-arm trial correction)
          taud[i,k] <- tau *2*(k-1)/k
          # adjustment, multi-arm RCTs
          w[i,k] <- delta[i,k] - ((d[t[i,k.ind[i,k]]] - d[t[i,k.ind[i,1]]])*(1-abs(split[i,k])) + direct *
split[i,k] )
          # cumulative adjustment for multi-arm trials
          sw[i,k] <- sum(w[i,1:k-1])/(k-1)
          }
   }

totresdev <- sum(resdev[])          #Total Residual Deviance
# Reference treatment currently Placebo (ref=1)
d[ref]<-0      # treatment effect is zero for reference treatment
D[class[ref]]<-0
# priors for mean class effect
for (j in 2:nc){
   D[j]~dnorm(0,.0001)
   }
# treatment effect = mean class effect
for (j in 2:nt){
   d[j] <- D[class[j]]
   }
direct ~ dnorm(0,.0001)      # vague prior for direct comparison parameter
indirect <- mean.diff[pair[1], pair[2]]
#calculate difference between direct and lor
diff.ns <- direct - indirect
# calculate p-value
prob <- step(diff.ns)
#
```

```
sd ~ dunif(0,25)     # vague prior for between-trial SD
tau <- pow(sd,-2)   # between-trial precision = (1/between-trial variance)
#
# pairwise mean differences for all possible pair-wise comparisons
for (c in 1:(nt-1)) {
   for (k in (c+1):nt)  {
         mean.diff[c,k] <- d[k]-d[c]
         mean.diff[k,c] <- -mean.diff[c,k]
         }
   }
}                               # *** PROGRAM ENDS
```

Discontinuation for any reason

A.4: Discontinuation for any reason, base-case model (WinBUGS)

```
model{
for(i in 1:ns){                  # LOOP OVER ALL STUDIES
   mu[i] ~ dnorm(0,.0001)          # vague priors for all trial baselines
   for (k in 1:na[i]){             # LOOP OVER ARMS
         r[i,k] ~ dbin(p[i,k],n[i,k])  # binomial likelihood
         logit(p[i,k]) <- mu[i] + d[t[i,k]] - d[t[i,1]] # model for linear predictor
         rhat[i,k] <- p[i,k] * n[i,k]  # expected value of the numerators
         #Deviance contribution
         dev[i,k] <- 2 * (r[i,k] * (log(r[i,k])-log(rhat[i,k])) + (n[i,k]-r[i,k]) * (log(n[i,k]-r[i,k]) -
log(n[i,k]-rhat[i,k])))
         }
   # Summed residual deviance contribution for this trial
   resdev[i] <- sum(dev[i,1:na[i]])
   }
totresdev <- sum(resdev[])        # Total Residual Deviance
#
# Reference treatment
```

```
d[ref]<-0       # treatment effect is zero for reference treatment
D[class[ref]]<-0
#
# vague prior for class effects
for (j in 2:nc){
   D[j] ~ dnorm(0, .0001)
   }
for (j in 2:nt){
   d[j] <- D[class[j]]
   }
#
# pairwise ORs and LORs for all possible pair-wise treatment comparisons
for (c in 1:(nt-1)){
   for (k in (c+1):nt){
         or[c,k] <- exp(d[k] - d[c])
         lor[c,k] <- (d[k]-d[c])
         }
   }
#
# pairwise differences for classes
for (c in 1:(nc-1)){
   for (k in (c+1):nc){
         diffClass[c,k] <- D[k] - D[c]
         orClass[c,k] <- exp(D[k] - D[c])
         }
    }
# ranking on relative scale
for (k in 1:nc){
   rkClass[k] <- rank(D[],k)
   bestClass[k] <- equals(rkClass[k],1) # Smallest is best (i.e. rank 1)
   # prob class k is h-th best, prob[1,k]=best[k]
   for (h in 1:nc) { probClass[h,k] <- equals(rkClass[k],h) }
   }
#
# ranking on relative scale - males
for (k in 1:19){ D.m[k] <- D[k]}
for (k in 20:(nc-2)){ D.m[k] <- D[k+2]}
for (k in 1:(nc-2)){
   rk.m[k] <- rank(D.m[],k)          # assumes lower values are "good"
   best.m[k] <- equals(rk.m[k],1)     #calculate probability that treat k is best
   # calculate probability that treat k is h-th best
   for (h in 1:nc){ prob.m[h,k] <- equals(rk.m[k],h) }
   }
}                                # *** PROGRAM ENDS
```

A.5: Discontinuation for any reason, node-splitting, class-level

A.5.1: R Code (requires R2OpenBUGS package)

```
################################################################################
# Node-splitting for Acne Guideline - Discontinuation (any)
# R script to run node-split for the MTC Random study effects, fixed
# class effects model using OpenBUGS
#
# Uses R2OpenBUGS package
#
# Discontinuation (any reason)
#  1. Need to include in the working directory the following files:
#           Disc any_UK.txt --- text file with data
#           fse fce node-splitR2_v3.txt --- text file holding BUGS code
#
#   2. Output files will be
#           coda1.txt  --- holds coda output
#           codaIndex.txt --- holds indexes to coda output
#          data.txt --- holds all data as used by BUGS
#          log.odc and log.txt --- hold WinBUGS output
#          inits1.txt --- holds initial values as read by BUGS
#          script.txt --- BUGS script file with all commands to execute
#
#   3. Output files for each node should be transferred to a new directory
#      as they will be overwritten in each new run
#
#   4. You may need to edit the OpenBUGS location 'bd'
#
#   5. You will need to edit the working directory 'pathname'
#      to suit your computer settings
#
#   6. Run script file
#
#   7. To repeat for other node-splits need to change variable 'pair'
#      and edit output file names
#
################################################################################
#
# Declare the directory where OpenBUGS is found in this computer
bd <- "C:/Program Files (x86)/OpenBUGS/OpenBUGS323/OpenBUGS.exe"
#
# Declare working directory
pathname <- "C:/Acne/M2M/Disc Any/"
setwd(pathname)
#
# load package to call OpenBUGS
library(R2OpenBUGS)
#
# LOAD DATA MANIPULATING FUNCTIONS:
```

```
#
PairXY <- function(treat, na, pair)
  # Check if pair(X,Y) in row i of data
  # and reorder treatments in trial as appropriate
{
  N <- nrow(treat)
  multi <- rep(NA,length(na))
  split.ind <- matrix(nrow=nrow(treat),ncol=ncol(treat))
  split.ind1 <- matrix(nrow=nrow(treat),ncol=ncol(treat))
  split.ind2 <- matrix(nrow=nrow(treat),ncol=ncol(treat))
  spliti <- rep(NA,length(na))
  split1i <- rep(NA,length(na))
  split2i <- rep(NA,length(na))
  pair1 <- matrix(nrow=nrow(treat),ncol=ncol(treat))
  pair2 <- matrix(nrow=nrow(treat),ncol=ncol(treat))
  k.ind <- matrix(nrow=nrow(treat),ncol=ncol(treat))
  for (i in 1:N) {
    # is trial i a multiarm trial?
    multi[i] <- 1*(na[i]>2)
    for (k in 1:na[i]){
      # which arms contain a treatment in the pair?
      split.ind[i,k] <- 1*(treat[i,k]==pair[1])+1*(treat[i,k]==pair[2])
      # which arms contain the treatment in pair[1]?
      split.ind1[i,k] <- 1*(treat[i,k]==pair[1])
      # which arms contain the treatment in pair[2]?
      split.ind2[i,k] <- 1*(treat[i,k]==pair[2])
    }
    # does trial i contain multiples of pair[1]?
    split1i[i] <- 1*(sum(split.ind1[i,1:na[i]])>1)
    # does trial i contain multiples of pair[2]?
    split2i[i] <- 1*(sum(split.ind2[i,1:na[i]])>1)
  # does trial i contain both treatments in the pair?
  # (minus duplicates in multiarm trials that have one treatment (only) in pair)
  spliti[i] <- 1*((sum(split.ind[i,1:na[i]])-split1i[i]*(sum(split.ind1[i,1:na[i]])-split1i[i])-split2i[i]*(sum(split.ind2[i,1:na[i]])-split2i[i]))>1)
  for (k in 1:na[i]) {
    # which arms contain the first element in the pair
    pair1[i,k] <- k*(1*(treat[i,k]==pair[1]))
    # which arms contain the second element in the pair
    pair2[i,k] <- k*(1*(treat[i,k]==pair[2]))
  }
  for (k in 1:na[i]) {
    # reposition order of arms within a trial according to node being split
    # k.ind ensures a treatment in the pair is in the baseline arm, where the
    # multi-arm trial contains both treatments in the pair
		# If a multi-arm trial does not contain the node, arm order stays the same
    k.ind[i,k]<-(k*((1-multi[i])+multi[i]*(1-spliti[i])+multi[i]*spliti[i]*(1*(split.ind[i,1]==1)))

		# If a multi-arm trial contains the node, the treatment in pair[1] is not duplicated in trial,
		# the baseline arm does not contain a treatment in the node, and the treatment
		# in arm k is pair[1], make this treatment baseline treatment
		 + multi[i]*spliti[i]*(1-split1i[i])*(1-(1*(split.ind[i,1]==1)))*(1*(treat[i,k]==pair[1]))

		# If a multi-arm trial contains the node, the treatment in pair[1] is duplicated in trial,
		# the treatment in pair[2] is not duplicated in trial, the baseline arm does not contain
		# a treatment in the node, and the treatment in arm k is pair[2], make this treatment baseline treatment
		+ multi[i]*spliti[i]*split1i[i]*(1-split2i[i])*(1-(1*(split.ind[i,1]==1)))*(1*(treat[i,k]==pair[2]))

		# If a multi-arm trial contains the node, the treatment in pair[1] is not duplicated in trial,
                # the baseline arm does not contain a treatment in the node, and k is baseline arm,
                # move treatment to come after baseline treatment
                + sum(pair1[i,1:na[i]])*(1-split1i[i])*multi[i]*spliti[i]*(1-(1*(split.ind[i,1]==1)))*(1*(k==1))

                # If a multi-arm trial contains the node, the treatment in pair[1] is duplicated in trial,
                # the treatment in pair[2] is not duplicated in trial, the baseline arm does not contain a
                # treatment in the node, and k is baseline arm, move treatment to come after baseline treatment
                + sum(pair2[i,1:na[i]])*split1i[i]*(1-split2i[i])*multi[i]*spliti[i]*(1-(1*(split.ind[i,1]==1)))*(1*(k==1))

                # If a multi-arm trial contains the node, the treatment in pair[1] are not duplicated in trial,
                # the baseline arm does not contain a treatment in the node, k is NOT baseline arm,
                # and treatment in arm k is NOT pair[1], arm order stays the same
                + k*multi[i]*spliti[i]*(1-split1i[i])*(1-(1*(split.ind[i,1]==1)))*(1-(1*(k==1)))*(1-(1*(treat[i,k]==pair[1])))

                # If a multi-arm trial contains the node, the treatment in pair[1] is duplicated in trial,
                # the treatment in pair[2] is not duplicated in trial, the baseline arm does not contain a treatment in the node, k is NOT baseline arm,
                # and treatment in arm k is NOT pair[2], arm order stays the same
                + k*multi[i]*spliti[i]*split1i[i]*(1-split2i[i])*(1-(1*(split.ind[i,1]==1)))*(1-(1*(k==1)))*(1-(1*(treat[i,k]==pair[2]))))
    }
  }
  k.ind
}
#################################################################################
```

```
# load data for MTC
MTCData <- read.table("Disc any_UK.txt", header=TRUE)
r <- data.matrix(MTCData[,c("r1", "r2", "r3", "r4", "r5")])
n <- data.matrix(MTCData[,c("n1", "n2", "n3", "n4", "n5")])
c <- data.matrix(MTCData[,c("c1", "c2", "c3", "c4", "c5")])
na <- data.matrix(MTCData[,"na"])
#Class when running model at class level
class <- 1:max(c,na.rm = TRUE)
nt <- max(c, na.rm=TRUE)
nc <- max(class)
ns <- nrow(r)
ref <- 1    #reference treatment
#
# define initial values
initv1 <- list(direct=0,  D=c(NA,rep(0,nc-1)), mu=rep(0,ns))
initv2 <- list(direct=0.05,  D=c(NA,rep(-1.2,nc-1)), mu=rep(0.5,ns))
#############################################################################
#
# Check which notes to split
#
library(gemtc)
ns.data<-mtc.data.studyrow(MTCData,
            armVars=c('treatment'='c', 'responders'='r', 'sampleSize'='n'),
            nArmsVar='na',
            studyVars=c(),
            studyNames=MTCData$studyid,
            treatmentNames=NA,
            patterns=c('%s', '%s%d'))
net<-mtc.network(data.ab=ns.data,description="Disc any_trt")
## Print which nodes to split
splitcomps<-mtc.nodesplit.comparisons(net)
```

```
print(splitcomps)
#
################################################################################
##
#  NODE-SPLITTING ROUTINE
################################################################################
##
#
#
# Define nodes to split
pair<-splitcomps
pair
# Run node split models
for(j in 1:length(pair[,1])){
  print(pair[j,])

  k.ind <- PairXY(treat=c,na=na[,1],pair=as.numeric(pair[j,]))

  # Setup subdirectory to hold results for each node-split
  dir.create(paste("REFCEnode",pair[j,1],"_",pair[j,2],sep=""))
  # Build data file: stored in the working directory as "data.txt"
  bugs.data(list("r"=r,"n"=n,"t"=c, "class"=class,
               "na" = na[,1], "nt" = nt, "nc" = nc, "ns" = ns, "ref" = ref,
               "pair" = as.numeric(pair[j,]), "k.ind" = k.ind))

  # Call OpenBUGS
  #
  bugs(data = "data.txt",
     inits = list(initv1,initv2),
     #inits = list(initv1),
     parameters.to.save = c("direct", "d", "prob","totresdev","indirect"),
     model.file = "fse fce node-splitR2_v3.txt",
  n.chains = 2,
  n.iter = 120000,          #including burn-in iterations
  n.burnin = 40000,
  n.thin = 1,
  OpenBUGS.pgm = bd,
  debug = FALSE,
  save.history = TRUE,
  useWINE=FALSE)
 #
 # Copy input and output files to relevant directory
 file.copy("data.txt", paste("REFCEnode",pair[j,1],"_",pair[j,2],"/data.txt",sep=""),
overwrite=TRUE)
 file.copy(paste(tempdir(),"/log.odc",sep=""),
paste(pathname,"/REFCEnode",pair[j,1],"_",pair[j,2],"/log.odc",sep=""), overwrite=TRUE)
 file.copy(paste(tempdir(),"/log.txt",sep=""),
paste(pathname,"/REFCEnode",pair[j,1],"_",pair[j,2],"/log.txt",sep=""), overwrite=TRUE)
 file.copy(paste(tempdir(),"/inits1.txt",sep=""),
paste(pathname,"/REFCEnode",pair[j,1],"_",pair[j,2],"/inits1.txt",sep=""), overwrite=TRUE)
 file.copy(paste(tempdir(),"/script.txt",sep=""),
paste(pathname,"/REFCEnode",pair[j,1],"_",pair[j,2],"/script.txt",sep=""), overwrite=TRUE)
 #
 # REPEAT FOR ALL OTHER NODES
}
```

A.5.2 OpenBUGS Code

```
model{
for(i in 1:ns){                 # LOOP OVER ALL STUDIES
  delta[i,1] <- 0  # treatment effect is zero for control arm
  mu[i] ~ dnorm(0,.0001)         # vague priors for all trial baselines
  for (k in 1:na[i]){            # LOOP OVER ARMS
        r[i,k.ind[i,k]] ~ dbin(p[i,k],n[i,k.ind[i,k]])  # binomial likelihood
        logit(p[i,k]) <- mu[i] + delta[i,k] # model for linear predictor
        rhat[i,k] <- p[i,k] * n[i,k.ind[i,k]]  # expected value of the numerators
        #Deviance contribution
```

```
          dev[i,k] <- 2 * (r[i,k.ind[i,k]] * (log(r[i,k.ind[i,k]])-log(rhat[i,k])) +  (n[i,k.ind[i,k]]-r[i,k.ind[i,k]]) * (log(n[i,k.ind[i,k]]-r[i,k.ind[i,k]]) - log(n[i,k.ind[i,k]]-rhat[i,k])))
          split[i,k] <- equals(t[i,k.ind[i,1]], pair[1]) * equals(t[i,k.ind[i,k]], pair[2]) - equals(t[i,k.ind[i,1]], pair[2]) * equals(t[i,k.ind[i,k]], pair[1])
          }
   # Summed residual deviance contribution for this trial
   resdev[i] <- sum(dev[i,1:na[i]])
   for (k in 2:na[i]) {           # FE model for treatment effects
          delta[i,k] <- (d[t[i,k.ind[i,k]]] - d[t[i,k.ind[i,1]]])*(1-abs(split[i,k])) + direct * split[i,k]
          }
   }
totresdev <- sum(resdev[])       # Total Residual Deviance
#
d[ref]<-0       # treatment effect is zero for reference treatment
D[class[ref]]<-0
# vague prior for class effects
for (j in 2:nc){
   D[j] ~ dnorm(0, .0001)
   }
for (j in 2:nt){
   d[j] <- D[class[j]]
   }
direct ~ dnorm(0,.0001)      # vague prior for direct comparison parameter
indirect <- lor[pair[1], pair[2]]
#calculate difference between direct and lor
diff <- direct - indirect
# calculate p-value
prob <- step(diff)
#
# pairwise ORs and LORs for all possible pair-wise comparisons
for (c in 1:(nt-1)){
   for (k in (c+1):nt){
          or[c,k] <- exp(d[k] - d[c])
          lor[c,k] <- (d[k]-d[c])
          lor[k,c] <- -lor[c,k]
          }
   }
}                        # *** PROGRAM ENDS

```

Discontinuation due to side effects

A.6: Discontinuation due to side effects, base-case model (WinBUGS)

```
model{
for(i in 1:ns){             # LOOP OVER ALL STUDIES
  mu[i] ~ dnorm(0,.0001)       # vague priors for all trial baselines
  for (k in 1:na[i]){          # LOOP OVER ARMS
        r[i,k] ~ dbin(p[i,k],n[i,k])  # binomial likelihood
        logit(p[i,k]) <- mu[i] + d[t[i,k]] - d[t[i,1]] # model for linear predictor
        rhat[i,k] <- p[i,k] * n[i,k]  # expected value of the numerators
        #Deviance contribution
        dev[i,k] <- 2 * (r[i,k] * (log(r[i,k])-log(rhat[i,k])) + (n[i,k]-r[i,k]) * (log(n[i,k]-r[i,k]) -
log(n[i,k]-rhat[i,k])))
        }
  # Summed residual deviance contribution for this trial
  resdev[i] <- sum(dev[i,1:na[i]])
  }
totresdev <- sum(resdev[])      # Total Residual Deviance
#
# Reference treatment
d[ref]<-0     # treatment effect is zero for reference treatment
D[class[ref]]<-0
#
# vague prior for class effects
```

```
for (j in 2:nc){
   D[j] ~ dnorm(0, .0001)
   }
for (j in 2:nt){
   d[j] <- D[class[j]]
   }
#
# pairwise ORs and LORs for all possible pair-wise treatment comparisons
for (c in 1:(nt-1)){
   for (k in (c+1):nt){
         or[c,k] <- exp(d[k] - d[c])
         lor[c,k] <- (d[k]-d[c])
         }
   }
#
# pairwise differences for classes
for (c in 1:(nc-1)){
   for (k in (c+1):nc){
         diffClass[c,k] <- D[k] - D[c]
         orClass[c,k] <- exp(D[k] - D[c])
         }
    }
# ranking on relative scale
for (k in 1:nc){
   rkClass[k] <- rank(D[],k)
   bestClass[k] <- equals(rkClass[k],1) # Smallest is best (i.e. rank 1)
   # prob class k is h-th best, prob[1,k]=best[k]
   for (h in 1:nc) { probClass[h,k] <- equals(rkClass[k],h) }
   }
#
# ranking on relative scale - males
```

```
for (k in 1:11){ D.m[k] <- D[k]}
for (k in 12:(nc-2)){ D.m[k] <- D[k+2]}
for (k in 1:(nc-2)){
   rk.m[k] <- rank(D.m[],k)           # assumes lower values are “good”
   best.m[k] <- equals(rk.m[k],1)      #calculate probability that treat k is best
   # calculate probability that treat k is h-th best
   for (h in 1:nc){ prob.m[h,k] <- equals(rk.m[k],h) }
   }
}                              # *** PROGRAM ENDS
```

A.7: Discontinuation due to side effects, node-splitting, class-level

A.7.1: R Code (requires R2OpenBUGS package)

```
#########################################################################
# Node-splitting for Acne Guideline - Discontinuation (due to SE)
# R script to run node-split for the MTC Fixed study effects, fixed
# class effects model using OpenBUGS
#
# Uses R2OpenBUGS package
#
# Discontinuation (due to SE)
#  1. Need to include in the working directory the following files:
#           Disc se.txt --- text file with data
#           fse fce node-splitR2_v3.txt --- text file holding BUGS code
#
#   2. Output files will be
#           data.txt --- holds all data as used by BUGS
#           log.odc and log.txt --- hold WinBUGS output
#           inits1.txt --- holds initial values as read by BUGS
#           script.txt --- BUGS script file with all commands to execute
#
#   3. Output files for each node should be transferred to a new directory
#      as they will be overwritten in each new run
#
#   4. You may need to edit the OpenBUGS location 'bd'
#
#   5. You will need to edit the working directory 'pathname'
#      to suit your computer settings
#
#   6. Run script file
#
##########################################################################
#
# Declare the directory where OpenBUGS is found in this computer
bd <- "C:/Program Files (x86)/OpenBUGS/OpenBUGS323/OpenBUGS.exe"
#
# Declare working directory
pathname <- "C:/ Acne/M2M/Disc SE/"
setwd(pathname)
#
# load package to call OpenBUGS
library(R2OpenBUGS)
#
# LOAD DATA MANIPULATING FUNCTIONS:
#
PairXY <- function(treat, na, pair)
  # Check if pair(X,Y) in row i of data
  # and reorder treatments in trial as appropriate
{
  N <- nrow(treat)
  multi <- rep(NA,length(na))
  split.ind <- matrix(nrow=nrow(treat),ncol=ncol(treat))
  split.ind1 <- matrix(nrow=nrow(treat),ncol=ncol(treat))
  split.ind2 <- matrix(nrow=nrow(treat),ncol=ncol(treat))
  spliti <- rep(NA,length(na))
  split1i <- rep(NA,length(na))
  split2i <- rep(NA,length(na))
  pair1 <- matrix(nrow=nrow(treat),ncol=ncol(treat))
  pair2 <- matrix(nrow=nrow(treat),ncol=ncol(treat))
  k.ind <- matrix(nrow=nrow(treat),ncol=ncol(treat))
  for (i in 1:N) {
    # is trial i a multiarm trial?
    multi[i] <- 1*(na[i]>2)
    for (k in 1:na[i]){
      # which arms contain a treatment in the pair?
      split.ind[i,k] <- 1*(treat[i,k]==pair[1])+1*(treat[i,k]==pair[2])
      # which arms contain the treatment in pair[1]?
      split.ind1[i,k] <- 1*(treat[i,k]==pair[1])
      # which arms contain the treatment in pair[2]?
      split.ind2[i,k] <- 1*(treat[i,k]==pair[2])
    }
    # does trial i contain multiples of pair[1]?
    split1i[i] <- 1*(sum(split.ind1[i,1:na[i]])>1)
    # does trial i contain multiples of pair[2]?
    split2i[i] <- 1*(sum(split.ind2[i,1:na[i]])>1)
    # does trial i contain both treatments in the pair?
    # (minus duplicates in multiarm trials that have one treatment (only) in pair)
    spliti[i] <- 1*((sum(split.ind[i,1:na[i]])-split1i[i]*(sum(split.ind1[i,1:na[i]])-split1i[i])-
split2i[i]*(sum(split.ind2[i,1:na[i]])-split2i[i]))>1)
    for (k in 1:na[i]) {
      # which arms contain the first element in the pair
      pair1[i,k] <- k*(1*(treat[i,k]==pair[1]))
      # which arms contain the second element in the pair
      pair2[i,k] <- k*(1*(treat[i,k]==pair[2]))
```

```
}
for (k in 1:na[i]) {
  # reposition order of arms within a trial according to node being split
  # k.ind ensures a treatment in the pair is in the baseline arm, where the
  # multi-arm trial contains both treatments in the pair
  # If a multi-arm trial does not contain the node, arm order stays the same
  k.ind[i,k]<-(k*((1-multi[i])+multi[i]*(1-spliti[i])+multi[i]*spliti[i]*(1*(split.ind[i,1]==1)))

        # If a multi-arm trial contains the node, the treatment in pair[1] is not duplicated in trial,
        # the baseline arm does not contain a treatment in the node, and the treatment
        # in arm k is pair[1], make this treatment baseline treatment
        + multi[i]*spliti[i]*(1-split1i[i])*(1-(1*(split.ind[i,1]==1)))*(1*(treat[i,k]==pair[1]))

        # If a multi-arm trial contains the node, the treatment in pair[1] is duplicated in trial,
        # the treatment in pair[2] is not duplicated in trial, the baseline arm does not contain
        # a treatment in the node, and the treatment in arm k is pair[2], make this treatment baseline treatment
        + multi[i]*spliti[i]*split1i[i]*(1-split2i[i])*(1-(1*(split.ind[i,1]==1)))*(1*(treat[i,k]==pair[2]))

        # If a multi-arm trial contains the node, the treatment in pair[1] is not duplicated in trial,
        # the baseline arm does not contain a treatment in the node, and k is baseline arm,
        # move treatment to come after baseline treatment
        + sum(pair1[i,1:na[i]])*(1-split1i[i])*multi[i]*spliti[i]*(1-(1*(split.ind[i,1]==1)))*(1*(k==1))

        # If a multi-arm trial contains the node, the treatment in pair[1] is duplicated in trial,
        # the treatment in pair[2] is not duplicated in trial, the baseline arm does not contain a
```

```
            # treatment in the node, and k is baseline arm, move treatment to come after baseline treatment
            + sum(pair2[i,1:na[i]])*split1i[i]*(1-split2i[i])*multi[i]*spliti[i]*(1-(1*(split.ind[i,1]==1)))*(1*(k==1))

            # If a multi-arm trial contains the node, the treatment in pair[1] are not duplicated in trial,
            # the baseline arm does not contain a treatment in the node, k is NOT baseline arm,
            # and treatment in arm k is NOT pair[1], arm order stays the same
            + k*multi[i]*spliti[i]*(1-split1i[i])*(1-(1*(split.ind[i,1]==1)))*(1-(1*(k==1)))*(1-(1*(treat[i,k]==pair[1])))

            # If a multi-arm trial contains the node, the treatment in pair[1] is duplicated in trial,
            # the treatment in pair[2] is not duplicated in trial, the baseline arm does not contain a treatment in the node, k is NOT baseline arm,
            # and treatment in arm k is NOT pair[2], arm order stays the same
            + k*multi[i]*spliti[i]*split1i[i]*(1-split2i[i])*(1-(1*(split.ind[i,1]==1)))*(1-(1*(k==1)))*(1-(1*(treat[i,k]==pair[2]))))
    }
  }
  k.ind
}
#########################################################################
# load data for MTC
MTCData <- read.table("Disc se_UK.txt", header=TRUE)
r <- data.matrix(MTCData[,c("r1", "r2", "r3", "r4", "r5")])
n <- data.matrix(MTCData[,c("n1", "n2", "n3", "n4", "n5")])
c <- data.matrix(MTCData[,c("c1", "c2", "c3", "c4", "c5")])
na <- data.matrix(MTCData[,"na"])
#Class when running model at class level
class <- 1:max(c,na.rm = TRUE)
nt <- max(c, na.rm=TRUE)
```

```
nc <- max(class)
ns <- nrow(r)
ref <- 1    #reference treatment
#
# define initial values
initv1 <- list(direct=0,  D=c(NA,rep(0,nc-1)), mu=rep(0,ns))
initv2 <- list(direct=0.05,  D=c(NA,rep(-1.2,nc-1)), mu=rep(0.5,ns))
################################################################################
#
# Check which notes to split
#
library(gemtc)
ns.data<-mtc.data.studyrow(MTCData,
                    armVars=c('treatment'='c', 'responders'='r', 'sampleSize'='n'),
                    nArmsVar='na',
                    studyVars=c(),
                    studyNames=MTCData$study,
                    treatmentNames=NA,
                    patterns=c('%s', '%s%d'))
net<-mtc.network(data.ab=ns.data,description="Disc se_trt")
## Print which nodes to split
splitcomps<-mtc.nodesplit.comparisons(net)
print(splitcomps)
#
################################################################################
##
#  NODE-SPLITTING ROUTINE
################################################################################
##
#
#
# Define nodes to split
```

```
pair<-splitcomps
pair
# Run node split models
for(j in 1:length(pair[,1])){
 print(pair[j,])

 k.ind <- PairXY(treat=c,na=na[,1],pair=as.numeric(pair[j,]))

 # Setup subdirectory to hold results for each node-split
 dir.create(paste("REFCEnode",pair[j,1],"_",pair[j,2],sep=""))

 # Build data file: stored in the working directory as "data.txt"
 bugs.data(list("r"=r,"n"=n,"t"=c, "class"=class,
               "na" = na[,1], "nt" = nt, "nc" = nc, "ns" = ns, "ref" = ref,
               "pair" = as.numeric(pair[j,]), "k.ind" = k.ind))

 # Call OpenBUGS
 #
 bugs(data = "data.txt",
     inits = list(initv1,initv2),
     #inits = list(initv1),
     parameters.to.save = c("direct", "d", "prob","totresdev","indirect"),
     model.file = "fse fce node-splitR2_v3.txt",
     n.chains = 2,
     n.iter = 120000,
     n.burnin = 40000,
     n.thin = 1,
     OpenBUGS.pgm = bd,
     debug = FALSE,
     save.history = TRUE,
     useWINE=FALSE)
 #
 # Copy input and output files to relevant directory
 file.copy("data.txt", paste("REFCEnode",pair[j,1],"_",pair[j,2],"/data.txt",sep=""), overwrite=TRUE)
 file.copy(paste(tempdir(),"/log.odc",sep=""), paste(pathname,"/REFCEnode",pair[j,1],"_",pair[j,2],"/log.odc",sep=""), overwrite=TRUE)
 file.copy(paste(tempdir(),"/log.txt",sep=""), paste(pathname,"/REFCEnode",pair[j,1],"_",pair[j,2],"/log.txt",sep=""), overwrite=TRUE)
 file.copy(paste(tempdir(),"/inits1.txt",sep=""), paste(pathname,"/REFCEnode",pair[j,1],"_",pair[j,2],"/inits1.txt",sep=""), overwrite=TRUE)
 file.copy(paste(tempdir(),"/script.txt",sep=""), paste(pathname,"/REFCEnode",pair[j,1],"_",pair[j,2],"/script.txt",sep=""), overwrite=TRUE)
 #
 # REPEAT FOR ALL OTHER NODES
}
```

A.7.2 OpenBUGS Code

```
model{
for(i in 1:ns){                    # LOOP OVER ALL STUDIES
   delta[i,1] <- 0  # treatment effect is zero for control arm
   mu[i] ~ dnorm(0,.0001)          # vague priors for all trial baselines
   for (k in 1:na[i]){             # LOOP OVER ARMS
         r[i,k.ind[i,k]] ~ dbin(p[i,k],n[i,k.ind[i,k]])  # binomial likelihood
         logit(p[i,k]) <- mu[i] + delta[i,k] # model for linear predictor
         rhat[i,k] <- p[i,k] * n[i,k.ind[i,k]]  # expected value of the numerators
         #Deviance contribution
         dev[i,k] <- 2 * (r[i,k.ind[i,k]] * (log(r[i,k.ind[i,k]])-log(rhat[i,k])) +  (n[i,k.ind[i,k]]-r[i,k.ind[i,k]]) * (log(n[i,k.ind[i,k]]-r[i,k.ind[i,k]]) - log(n[i,k.ind[i,k]]-rhat[i,k])))
         split[i,k] <- equals(t[i,k.ind[i,1]], pair[1]) * equals(t[i,k.ind[i,k]], pair[2]) - equals(t[i,k.ind[i,1]], pair[2]) * equals(t[i,k.ind[i,k]], pair[1])
         }
   # Summed residual deviance contribution for this trial
   resdev[i] <- sum(dev[i,1:na[i]])
   for (k in 2:na[i]) {            # FE model for treatment effects
```

```
            delta[i,k] <- (d[t[i,k.ind[i,k]]] - d[t[i,k.ind[i,1]]])*(1-abs(split[i,k])) + direct * split[i,k]
            }
    }
totresdev <- sum(resdev[])        # Total Residual Deviance
#
d[ref]<-0       # treatment effect is zero for reference treatment
D[class[ref]]<-0
# vague prior for class effects
for (j in 2:nc){
    D[j] ~ dnorm(0, .0001)
    }
for (j in 2:nt){
    d[j] <- D[class[j]]
    }
direct ~ dnorm(0,.0001)      # vague prior for direct comparison parameter
indirect <- lor[pair[1], pair[2]]
#calculate difference between direct and lor
diff <- direct - indirect
# calculate p-value
prob <- step(diff)
#
# pairwise ORs and LORs for all possible pair-wise comparisons
for (c in 1:(nt-1)){
    for (k in (c+1):nt){
            or[c,k] <- exp(d[k] - d[c])
            lor[c,k] <- (d[k]-d[c])
            lor[k,c] <- -lor[c,k]
            }
    }
}                              # *** PROGRAM ENDS
```