

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

Disclaimer

The recommendations in this guideline represent the view of NICE, arrived at after careful consideration of the evidence available. When exercising their judgement, professionals are expected to take this guideline fully into account, alongside the individual needs, preferences and values of their patients or service users. The recommendations in this guideline are not mandatory and the guideline does not override the responsibility of healthcare professionals to make decisions appropriate to the circumstances of the individual patient, in consultation with the patient and/or their carer or guardian.

Local commissioners and/or providers have a responsibility to enable the guideline to be applied when individual health professionals and their patients or service users wish to use it. They should do so in the context of local and national priorities for funding and developing services, and in light of their duties to have due regard to the need to eliminate unlawful discrimination, to advance equality of opportunity and to reduce health inequalities. Nothing in this guideline should be interpreted in a way that would be inconsistent with compliance with those duties.

NICE guidelines cover health and care in England. Decisions on how they apply in other UK countries are made by ministers in the [Welsh Government](#), [Scottish Government](#), and [Northern Ireland Executive](#). All NICE guidance is subject to regular review and may be updated or withdrawn.

Copyright

© NICE 2020. All rights reserved. Subject to [Notice of Rights](#).

ISBN:

Contents

TSU NMA software code (mild to moderate acne)	5
Efficacy (% change in total lesion count from baseline)	5
A.1: Efficacy, base-case model (OpenBUGS)	5
A.2: Efficacy, bias-adjusted model: small study effects (OpenBUGS).....	10
A.3: Efficacy, node-splitting, class-level.....	16
Discontinuation for any reason	28
A.4: Discontinuation for any reason, base-case model (WinBUGS).....	28
A.5: Discontinuation for any reason, node-splitting, class-level.....	30
Discontinuation due to side effects	39
A.6: Discontinuation due to side effects, base-case model (WinBUGS).....	39
A.7: Discontinuation due to side effects, node-splitting, class-level.....	41

1 TSU NMA software code (mild to moderate 2 acne)

Efficacy (% change in total lesion count from baseline)

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

```
5 # Arm and Trial-level data
6 # Random effects model for multi-arm trials
7 # Fixed class effects
8 model{                                # *** PROGRAM STARTS
9   for(i in 1:ns.a){                  # LOOP THROUGH STUDIES WITH ARM DATA
10    w[i,1] <- 0                        # adjustment for multi-arm trials is zero for control arm
11    delta[i,1] <- 0                    # treatment effect is zero for control arm
12    mu[i] ~ dnorm(0,.0001)            # vague priors for all trial baselines
13  }
14
15 # trials reporting percent CFB
16 for(i in 1:ns.a1){                  # LOOP THROUGH STUDIES WITH %CFB ARM DATA
17   for (k in 1:na[i]) {              # LOOP THROUGH ARMS
18     pCFB.se[i,k] <- pCFB.sd[i,k]/sqrt(n[i,k]) # calculate standard error
19     pCFB.var[i,k] <- pow(pCFB.se[i,k],2) # calculate variances
20     pCFB.prec[i,k] <- 1/pCFB.var[i,k] # set precisions
21     pCFB[i,k] ~ dnorm(theta[i,k],pCFB.prec[i,k]) # normal likelihood
22
23     theta[i,k] <- mu[i] + delta[i,k] # model for linear predictor
24
25     #Deviance contribution
26     dev[i,k] <- (pCFB[i,k]-theta[i,k])*(pCFB[i,k]-theta[i,k])*pCFB.prec[i,k]
27   }
28   resdev[i] <- sum(dev[i,1:na[i]])
29 }
```

```
1 }
2 # trials reporting CFB + B # LOOP THROUGH STUDIES WITH CFB+B ARM DATA
3 for(i in (ns.a1+1):(ns.a1+ns.a2)){
4   for (k in 1:na[i]) { # LOOP THROUGH ARMS
5     x.se[i,k] <- x.sd[i,k]/sqrt(n[i,k]) # calculate standard error
6     x.var[i,k] <- pow(x.se[i,k],2) # calculate variances
7     x.prec[i,k] <- 1/x.var[i,k] # set precisions
8     x[i,k] ~ dnorm(mu.X[i,k],x.prec[i,k]) # indpt normal likelihood for baseline mean
9     mu.X[i,k] ~ dnorm(0,.0001) # flat prior for baseline mean in likelihood
10
11     CFB.se[i,k] <- CFB.sd[i,k]/sqrt(n[i,k]) # calculate standard error
12     CFB.var[i,k] <- pow(CFB.se[i,k],2) # calculate variances
13     CFB.prec[i,k] <- 1/CFB.var[i,k] # set precisions
14     mu.CFB[i,k] <- mu.X[i,k]*(theta[i,k]/100)# calculate mean for CFB likelihood
15     CFB[i,k] ~ dnorm(mu.CFB[i,k],CFB.prec[i,k]) # indpt normal likelihood for baseline mean
16
17     theta[i,k] <- mu[i] + delta[i,k] # model for linear predictor
18
19     #Deviance contribution
20     dev[i,k] <- (CFB[i,k]-mu.CFB[i,k])*(CFB[i,k]-mu.CFB[i,k])*CFB.prec[i,k]
21   }
22   resdev[i] <- sum(dev[i,1:na[i]])
23 }
24 }
25 # trials reporting B + F
26 for(i in (ns.a1+ns.a2+1):ns.a){ # LOOP THROUGH STUDIES WITH B+F ARM DATA
27   for (k in 1:na[i]) { # LOOP THROUGH ARMS
28     #Calculate standard errors
29     x.se[i,k] <- x.sd[i,k]/sqrt(n[i,k])
30     y.se[i,k] <- y.sd[i,k]/sqrt(n[i,k])
31     #Set precision matrix
```

```
1   Sigma[i,k,1,1]<-pow(x.se[i,k],2)
2   Sigma[i,k,2,2]<-pow(y.se[i,k],2)
3   Sigma[i,k,1,2]<-corr[i]*x.se[i,k]*y.se[i,k]
4   Sigma[i,k,2,1]<-Sigma[i,k,1,2]
5   Prec[i,k,1:2,1:2]<-inverse(Sigma[i,k,1:2,1:2])
6   #Set up vector for baseline and follow-up means
7   y.XY[i,k,1]<-x[i,k]
8   y.XY[i,k,2]<-y[i,k]
9
10  # Bivariate normal likelihood for baseline and follow-up
11  y.XY[i,k,1:2]~dmnorm(mu.XY[i,k,1:2],Prec[i,k,1:2,1:2])
12  mu.XY[i,k,2]<- mu.XY[i,k,1]*(1-(theta[i,k]/100))
13  mu.XY[i,k,1] ~ dnorm(0,.0001)    # flat prior for baseline mean in likelihood
14
15  theta[i,k] <- mu[i] + delta[i,k] # model for linear predictor
16
17  #Deviance contribution
18  for (j in 1:2){
19    diff[i,k,j]<- y.XY[i,k,j]-mu.XY[i,k,j]
20    z[i,k,j]<- inprod(Prec[i,k,j,1:2],diff[i,k,1:2])
21  }
22  dev[i,k]<-inprod(diff[i,k,1:2],z[i,k,1:2])
23  }
24  resdev[i] <- sum(dev[i,1:na[i]])
25  }
26 # 2-arm trials reporting contrasts (e.g., split-face trials)
27 for(i in (ns.a+1):(ns.a+ns.t2)){    # LOOP THROUGH STUDIES WITH TRIAL DATA
28   w[i,1] <- 0          # adjustment for multi-arm trials is zero for control arm
29   delta[i,1] <- 0     # treatment effect is zero for control arm
30   var[i,2] <- pow(se.T[i,2],2) # calculate variances
31   prec[i,2] <- 1/var[i,2]    # set precisions
```

```
1  y.T[i,2] ~ dnorm(delta[i,2],prec[i,2]) # normal likelihood
2  # Deviance contribution
3  dev[i,2] <- (y.T[i,2]-delta[i,2])* (y.T[i,2]-delta[i,2])* prec[i,2]
4  # summed residual deviance contribution for this trial
5  resdev[i] <- dev[i,2]
6  }
7  #RE Model (ARM AND TRIAL DATA)
8  for(i in 1:ns){          # LOOP THROUGH STUDIES WITH ARM DATA
9    for (k in 2:na[i]) {   # LOOP THROUGH ARMS
10     # trial-specific RE distributions
11     delta[i,k] ~ dnorm(md[i,k],taud[i,k])
12     # mean of RE distributions, with multi-arm trial correction
13     md[i,k] <- d[t[i,k]] - d[t[i,1]] + sw[i,k]
14     # precision of RE distributions (with multi-arm trial correction)
15     taud[i,k] <- tau *2*(k-1)/k
16     # adjustment, multi-arm RCTs
17     w[i,k] <- (delta[i,k] - d[t[i,k]] + d[t[i,1]])
18     # cumulative adjustment for multi-arm trials
19     sw[i,k] <- sum(w[i,1:k-1])/(k-1)
20   }
21 }
22
23 totresdev <- sum(resdev[])      #Total Residual Deviance
24 # Reference treatment currently Placebo (ref=1)
25 d[ref]<-0      # treatment effect is zero for reference treatment
26 D[class[ref]]<-0
27 # priors for mean class effect
28 for (j in 2:nc){
29   D[j]~dnorm(0,.0001)
30 }
31 # treatment effect = mean class effect
```



```
1 for (j in 2:nt){
2   d[j] <- D[class[j]]
3 }
4 #
5 sd ~ dunif(0,25) # vague prior for between-trial SD
6 tau <- pow(sd,-2) # between-trial precision = (1/between-trial variance)
7 #
8 # pairwise mean differences for all possible pair-wise comparisons
9 for (c in 1:(nt-1)) {
10  for (k in (c+1):nt) { mean.diff[c,k] <- d[k]-d[c] }
11 }
12 # pairwise differences for classes
13 for (c in 1:(nc-1)){
14   for (k in (c+1):nc){
15     diffClass[c,k] <- D[k] - D[c]
16   }
17 }
18 # rank all classes
19 # ranking on relative scale
20 for (k in 1:nc){
21   # rk[k] <- rank(D[,k]) # assumes lower values are "good"
22   rk[k] <- nc+1-rank(D[,k]) # assumes higher values are "good"
23   best[k] <- equals(rk[k],1) #calculate probability that treat k is best
24   # calculate probability that treat k is h-th best
25   for (h in 1:nc){ prob[h,k] <- equals(rk[k],h) }
26 }
27 # ranking on relative scale - males
28 for (k in 1:18){ D.m[k] <- D[k]}
29 for (k in 19:(nc-2)){ D.m[k] <- D[k+2]}
30 for (k in 1:(nc-2)){
31   rk.m[k] <- (nc-2)+1-rank(D.m[,k]) # assumes higher values are "good"
```

```
1 best.m[k] <- equals(rk.m[k],1) #calculate probability that treat k is best
2 # calculate probability that treat k is h-th best
3 for (h in 1:nc){ prob.m[h,k] <- equals(rk.m[k],h) }
4 }
5 } # *** PROGRAM ENDS
6
```

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

```
8 # Arm and Trial-level data
9 # Random effects model for multi-arm trials
10 # Fixed class effects
11 model{ # *** PROGRAM STARTS
12 for(i in 1:ns) {
13     Nsum[i]<- sum(n[i,1:na[i]])
14 }
15 for(i in 1:ns.a){ # LOOP THROUGH STUDIES WITH ARM DATA
16     w[i,1] <- 0 # adjustment for multi-arm trials is zero for control arm
17     delta[i,1] <- 0 # treatment effect is zero for control arm
18     beta[i,1] <- 0 # No bias on baseline arm
19     mu[i] ~ dnorm(0,.0001) # vague priors for all trial baselines
20 }
21
22 # trials reporting percent CFB
23 for(i in 1:ns.a1){ # LOOP THROUGH STUDIES WITH %CFB ARM DATA
24     for (k in 1:na[i]) { # LOOP THROUGH ARMS
25         pCFB.se[i,k] <- pCFB.sd[i,k]/sqrt(n[i,k]) # calculate standard error
26         pCFB.var[i,k] <- pow(pCFB.se[i,k],2) # calculate variances
27         pCFB.prec[i,k] <- 1/pCFB.var[i,k] # set precisions
28         pCFB[i,k] ~ dnorm(theta[i,k],pCFB.prec[i,k]) # normal likelihood
29
30         theta[i,k] <- mu[i] + delta[i,k] + beta[i,k]*X[i,k]/sqrt(Nsum[i]) # model for linear predictor
```

```
1
2   #Deviance contribution
3   dev[i,k] <- (pCFB[i,k]-theta[i,k])*(pCFB[i,k]-theta[i,k])*pCFB.prec[i,k]
4   }
5   resdev[i] <- sum(dev[i,1:na[i]])
6 )
7 }
8 # trials reporting CFB + B   # LOOP THROUGH STUDIES WITH CFB+B ARM DATA
9 for(i in (ns.a1+1):(ns.a1+ns.a2)){
10  for (k in 1:na[i]) {      # LOOP THROUGH ARMS
11    x.se[i,k] <- x.sd[i,k]/sqrt(n[i,k]) # calculate standard error
12    x.var[i,k] <- pow(x.se[i,k],2) # calculate variances
13    x.prec[i,k] <- 1/x.var[i,k] # set precisions
14    x[i,k] ~ dnorm(mu.X[i,k],x.prec[i,k]) # indpt normal likelihood for baseline mean
15    mu.X[i,k] ~ dnorm(0,.0001) # flat prior for baseline mean in likelihood
16
17    CFB.se[i,k] <- CFB.sd[i,k]/sqrt(n[i,k]) # calculate standard error
18    CFB.var[i,k] <- pow(CFB.se[i,k],2) # calculate variances
19    CFB.prec[i,k] <- 1/CFB.var[i,k] # set precisions
20    mu.CFB[i,k] <- mu.X[i,k]*(theta[i,k]/100)# calculate mean for CFB likelihood
21    CFB[i,k] ~ dnorm(mu.CFB[i,k],CFB.prec[i,k]) # indpt normal likelihood for baseline mean
22
23    theta[i,k] <- mu[i] + delta[i,k] + beta[i,k]*X[i,k]/sqrt(Nsum[i]) # model for linear predictor
24
25    #Deviance contribution
26    dev[i,k] <- (CFB[i,k]-mu.CFB[i,k])*(CFB[i,k]-mu.CFB[i,k])*CFB.prec[i,k]
27    }
28    resdev[i] <- sum(dev[i,1:na[i]])
29  )
30 }
31 # trials reporting B + F
```

```
1 for(i in (ns.a1+ns.a2+1):ns.a){      # LOOP THROUGH STUDIES WITH B+F ARM DATA
2   for (k in 1:na[i]) {              # LOOP THROUGH ARMS
3     #Calculate standard errors
4     x.se[i,k] <- x.sd[i,k]/sqrt(n[i,k])
5     y.se[i,k] <- y.sd[i,k]/sqrt(n[i,k])
6     #Set precision matrix
7     Sigma[i,k,1,1]<-pow(x.se[i,k],2)
8     Sigma[i,k,2,2]<-pow(y.se[i,k],2)
9     Sigma[i,k,1,2]<-corr[i]*x.se[i,k]*y.se[i,k]
10    Sigma[i,k,2,1]<-Sigma[i,k,1,2]
11    Prec[i,k,1:2,1:2]<-inverse(Sigma[i,k,1:2,1:2])
12    #Set up vector for baseline and follow-up means
13    y.XY[i,k,1]<-x[i,k]
14    y.XY[i,k,2]<-y[i,k]
15
16    # Bivariate normal likelihood for baseline and follow-up
17    y.XY[i,k,1:2]~dmnorm(mu.XY[i,k,1:2],Prec[i,k,1:2,1:2])
18    mu.XY[i,k,2]<- mu.XY[i,k,1]*(1-(theta[i,k]/100))
19    mu.XY[i,k,1] ~ dnorm(0,.0001)      # flat prior for baseline mean in likelihood
20
21    theta[i,k] <- mu[i] + delta[i,k] + beta[i,k]*X[i,k]/sqrt(Nsum[i]) # model for linear predictor
22
23    #Deviance contribution
24    for (j in 1:2){
25      diff[i,k,j]<- y.XY[i,k,j]-mu.XY[i,k,j]
26      z[i,k,j]<- inprod(Prec[i,k,j,1:2],diff[i,k,1:2])
27    }
28    dev[i,k]<-inprod(diff[i,k,1:2],z[i,k,1:2])
29  }
30  resdev[i] <- sum(dev[i,1:na[i]])
31 }
```

```
1 # 2-arm trials reporting contrasts (e.g., split-face trials)
2 for(i in (ns.a+1):(ns.a+ns.t2)){ # LOOP THROUGH STUDIES WITH TRIAL DATA
3   w[i,1] <- 0 # adjustment for multi-arm trials is zero for control arm
4   delta[i,1] <- 0 # treatment effect is zero for control arm
5   var[i,2] <- pow(se.T[i,2],2) # calculate variances
6   prec[i,2] <- 1/var[i,2] # set precisions
7   y.T[i,2] ~ dnorm(theta[i,2],prec[i,2]) # normal likelihood
8   theta[i,2] <- delta[i,2] + beta[i,2]*X[i,2]/sqrt(Nsum[i]) # model for linear predictor.
9   # Deviance contribution
10  dev[i,2] <- (y.T[i,2]-theta[i,2])*(y.T[i,2]-theta[i,2])* prec[i,2]
11  # summed residual deviance contribution for this trial
12  resdev[i] <- dev[i,2]
13 }
14 #RE Model (ARM AND TRIAL DATA)
15 for(i in 1:ns){ # LOOP THROUGH STUDIES WITH ARM DATA
16   for (k in 2:na[i]) { # LOOP THROUGH ARMS
17     # trial-specific RE distributions
18     delta[i,k] ~ dnorm(md[i,k],taud[i,k])
19     # model for bias parameter beta
20     beta[i,k] ~ dnorm(b, prec.b)
21     # mean of RE distributions, with multi-arm trial correction
22     md[i,k] <- d[t[i,k]] - d[t[i,1]] + sw[i,k]
23     # precision of RE distributions (with multi-arm trial correction)
24     taud[i,k] <- tau *2*(k-1)/k
25     # adjustment, multi-arm RCTs
26     w[i,k] <- (delta[i,k] - d[t[i,k]] + d[t[i,1]])
27     # cumulative adjustment for multi-arm trials
28     sw[i,k] <- sum(w[i,1:k-1])/(k-1)
29   }
30 }
31
```

```
1  totresdev <- sum(resdev[])      #Total Residual Deviance
2  # Reference treatment currently Placebo (ref=1)
3  d[ref]<-0    # treatment effect is zero for reference treatment
4  D[class[ref]]<-0
5  # priors for mean class effect
6  for (j in 2:nc){
7    D[j]~dnorm(0,.0001)
8  }
9  # treatment effect = mean class effect
10 for (j in 2:nt){
11   d[j] <- D[class[j]]
12 }
13 #
14 sd ~ dunif(0,25)  # vague prior for between-trial SD
15 tau <- pow(sd,-2) # between-trial precision = (1/between-trial variance)
16 # bias model prior for variance
17 sd.b ~ dunif(0,1000)
18 prec.b <- pow(sd.b,-2)
19 # bias model prior for mean
20 b ~ dnorm(0,.0001)
21 #
22 # pairwise mean differences for all possible pair-wise treatment comparisons
23 for (c in 1:(nt-1)) {
24   for (k in (c+1):nt) { mean.diff[c,k] <- d[k]-d[c] }
25 }
26 # pairwise differences for classes
27 for (c in 1:(nc-1)){
28   for (k in (c+1):nc){
29     diffClass[c,k] <- D[k] - D[c]
30   }
31 }
```

```
1 #Adjusted estimates for n = 1670
2 diffClass1670[1,1] <- 0
3 diffClass1670[1,2] <- diffClass[1,2]
4 for (k in 3:nc){
5   diffClass1670[1,k] <- diffClass[1,k] + b/sqrt(1670)
6   diffClass1670[2,k] <- diffClass[2,k] + b/sqrt(1670)
7 }
8 for (c in 3:(nc-1)){
9   for (k in (c+1):nc){
10    diffClass1670[c,k] <- diffClass[c,k]
11   }
12 }
13 # rank all classes
14 # ranking on relative scale
15 for (k in 1:nc){
16   # rk[k] <- rank(diffClass1670[1,],k) # assumes lower values are "good"
17   rk[k] <- nc+1-rank(diffClass1670[1,],k) # assumes higher values are "good"
18   best[k] <- equals(rk[k],1) #calculate probability that treat k is best
19   # calculate probability that treat k is h-th best
20   for (h in 1:nc){ prob[h,k] <- equals(rk[k],h) }
21 }
22 # ranking on relative scale - males
23 for (k in 1:18){ D.m[k] <- diffClass1670[1,k]}
24 for (k in 19:(nc-2)){ D.m[k] <- diffClass1670[1,k+2]}
25 for (k in 1:(nc-2)){
26   rk.m[k] <- (nc-2)+1-rank(D.m[,k]) # assumes higher values are "good"
27   best.m[k] <- equals(rk.m[k],1) #calculate probability that treat k is best
28   # calculate probability that treat k is h-th best
29   for (h in 1:nc){ prob.m[h,k] <- equals(rk.m[k],h) }
30 }
31 } # *** PROGRAM ENDS
```

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

A.3.1: R Code (requires R2OpenBUGS package)

```
3 #####
4 #
5 # Node-splitting for Acne Guideline - Efficacy at Class Level
6 # R script to run node-split for the MTC Random study effects, fixed
7 # class effects model using OpenBUGS
8 #
9 # Uses R2OpenBUGS package
10 #
11 # Efficacy
12 # 1. Need to include in the working directory the following files:
13 #     efficacy_class.txt --- text file with data
14 #     rse fce node-splitR2_v2_efficacy_class.txt --- text file holding BUGS code
15 #
16 # 2. Output files will be
17 #     data.txt --- holds all data as used by BUGS
18 #     log.odc and log.txt --- hold WinBUGS output
19 #     inits1.txt --- holds initial values as read by BUGS
20 #     script.txt --- BUGS script file with all commands to execute
21 #
22 # 3. Output files for each node should be transferred to a new directory
23 #     as they will be overwritten in each new run
24 #
25 # 4. You may need to edit the OpenBUGS location 'bd'
26 #
27 # 5. You will need to edit the working directory 'pathname'
28 #     to suit your computer settings
29 #
30 # 6. Run script file
31 #
```



```
1 #####
2 #
3 # Declare the directory where OpenBUGS is found in this computer
4 bd <- "C:/Program Files (x86)/OpenBUGS/OpenBUGS323/OpenBUGS.exe"
5 #
6 # Declare working directory
7 pathname <- "C:/Acne/M2M/Efficacy"
8 setwd(pathname)
9 #
10 # load package to call OpenBUGS
11 library(R2OpenBUGS)
12 #
13 # LOAD DATA MANIPULATING FUNCTIONS:
14 #
15 PairXY <- function(treat, na, pair)
16 # Check if pair(X,Y) in row i of data
17 # and reorder treatments in trial as appropriate
18 {
19 N <- nrow(treat)
20 multi <- rep(NA,length(na))
21 split.ind <- matrix(nrow=nrow(treat),ncol=ncol(treat))
22 split.ind1 <- matrix(nrow=nrow(treat),ncol=ncol(treat))
23 split.ind2 <- matrix(nrow=nrow(treat),ncol=ncol(treat))
24 spliti <- rep(NA,length(na))
25 split1i <- rep(NA,length(na))
26 split2i <- rep(NA,length(na))
27 pair1 <- matrix(nrow=nrow(treat),ncol=ncol(treat))
28 pair2 <- matrix(nrow=nrow(treat),ncol=ncol(treat))
29 k.ind <- matrix(nrow=nrow(treat),ncol=ncol(treat))
30 for (i in 1:N) {
31 # is trial i a multiarm trial?
```

```
1 multi[i] <- 1*(na[i]>2)
2 for (k in 1:na[i]){
3   # which arms contain a treatment in the pair?
4   split.ind[i,k] <- 1*(treat[i,k]==pair[1])+1*(treat[i,k]==pair[2])
5   # which arms contain the treatment in pair[1]?
6   split.ind1[i,k] <- 1*(treat[i,k]==pair[1])
7   # which arms contain the treatment in pair[2]?
8   split.ind2[i,k] <- 1*(treat[i,k]==pair[2])
9 }
10 # does trial i contain multiples of pair[1]?
11 split1i[i] <- 1*(sum(split.ind1[i,1:na[i]])>1)
12 # does trial i contain multiples of pair[2]?
13 split2i[i] <- 1*(sum(split.ind2[i,1:na[i]])>1)
14 # does trial i contain both treatments in the pair?
15 # (minus duplicates in multiarm trials that have one treatment (only) in pair)
16 spliti[i] <- 1*((sum(split.ind[i,1:na[i]])-split1i[i]*(sum(split.ind1[i,1:na[i]])-split1i[i])-
17 split2i[i]*(sum(split.ind2[i,1:na[i]])-split2i[i]))>1)
18 for (k in 1:na[i]) {
19   # which arms contain the first element in the pair
20   pair1[i,k] <- k*(1*(treat[i,k]==pair[1]))
21   # which arms contain the second element in the pair
22   pair2[i,k] <- k*(1*(treat[i,k]==pair[2]))
23 }
24 for (k in 1:na[i]) {
25   # reposition order of arms within a trial according to node being split
26   # k.ind ensures a treatment in the pair is in the baseline arm, where the
27   # multi-arm trial contains both treatments in the pair
28
29   # multi-arm trial contains both treatments in the pair
30   # If a multi-arm trial does not contain the node, arm order stays the same
31   k.ind[i,k]<-(k*((1-multi[i])+multi[i]*(1-spliti[i])+multi[i]*spliti[i]*(1*(split.ind[i,1]==1))))
```

```
1
2     # If a multi-arm trial contains the node, the treatment in pair[1] is not duplicated in trial,
3     # the baseline arm does not contain a treatment in the node, and the treatment
4     # in arm k is pair[1], make this treatment baseline treatment
5     + multi[i]*spliti[i]*(1-split1i[i])*(1-(1*(split.ind[i,1]==1)))*(1*(treat[i,k]==pair[1]))
6
7     # If a multi-arm trial contains the node, the treatment in pair[1] is duplicated in trial,
8     # the treatment in pair[2] is not duplicated in trial, the baseline arm does not contain
9     # a treatment in the node, and the treatment in arm k is pair[2], make this treatment
10    baseline treatment
11    + multi[i]*spliti[i]*split1i[i]*(1-split2i[i])*(1-(1*(split.ind[i,1]==1)))*(1*(treat[i,k]==pair[2]))
12
13    # If a multi-arm trial contains the node, the treatment in pair[1] is not duplicated in trial,
14    # the baseline arm does not contain a treatment in the node, and k is baseline arm,
15    # move treatment to come after baseline treatment
16    + sum(pair1[i,1:na[i]])*(1-split1i[i])*multi[i]*spliti[i]*(1-(1*(split.ind[i,1]==1)))*(1*(k==1))
17
18    # If a multi-arm trial contains the node, the treatment in pair[1] is duplicated in trial,
19    # the treatment in pair[2] is not duplicated in trial, the baseline arm does not contain a
20    # treatment in the node, and k is baseline arm, move treatment to come after baseline
21    treatment
22    + sum(pair2[i,1:na[i]])*split1i[i]*(1-split2i[i])*multi[i]*spliti[i]*(1-
23    (1*(split.ind[i,1]==1)))*(1*(k==1))
24
25    # If a multi-arm trial contains the node, the treatment in pair[1] are not duplicated in
26    trial,
27    # the baseline arm does not contain a treatment in the node, k is NOT baseline arm,
28    # and treatment in arm k is NOT pair[1], arm order stays the same
29    + k*multi[i]*spliti[i]*(1-split1i[i])*(1-(1*(split.ind[i,1]==1)))*(1-(1*(k==1)))*(1-
30    (1*(treat[i,k]==pair[1])))
31
32    # If a multi-arm trial contains the node, the treatment in pair[1] is duplicated in trial,
```

```
1      # the treatment in pair[2] is not duplicated in trial, the baseline arm does not contain a
2 treatment in the node, k is NOT baseline arm,
3      # and treatment in arm k is NOT pair[2], arm order stays the same
4      + k*multi[i]*spliti[j]*split1i[j]*(1-split2i[j]*(1-(1*(split.ind[i,1]==1)))*(1-(1*(k==1)))*(1-
5 (1*(treat[i,k]==pair[2]))))
6    }
7  }
8  k.ind
9  }
10 #####
11 #
12 # load data for MTC
13 MTCData <- read.table("efficacy_class.txt", header=TRUE)
14 n <- data.matrix(MTCData[,c("n1", "n2", "n3", "n4", "n5")])
15 x <- data.matrix(MTCData[,c("x1", "x2", "x3", "x4", "x5")])
16 x.sd <- data.matrix(MTCData[,c("x.sd1", "x.sd2", "x.sd3", "x.sd4", "x.sd5")])
17 y <- data.matrix(MTCData[,c("y1", "y2", "y3", "y4")])
18 y.sd <- data.matrix(MTCData[,c("y.sd1", "y.sd2", "y.sd3", "y.sd4")])
19 CFB <- data.matrix(MTCData[,c("CFB1", "CFB2", "CFB3", "CFB4", "CFB5")])
20 CFB.sd <- data.matrix(MTCData[,c("CFB.sd1", "CFB.sd2", "CFB.sd3", "CFB.sd4",
21 "CFB.sd5")])
22 pCFB <- data.matrix(MTCData[,c("pCFB1", "pCFB2", "pCFB3", "pCFB4")])
23 pCFB.sd <- data.matrix(MTCData[,c("pCFB.sd1", "pCFB.sd2", "pCFB.sd3", "pCFB.sd4")])
24 y.T <- data.matrix(cbind(rep(NA,length(n[,1])),MTCData[,c("y.T2")]))
25 se.T <- data.matrix(cbind(rep(NA,length(n[,1])),MTCData[,c("se.T2")]))
26 corr <- data.matrix(MTCData[, "corr"])
27 c <- data.matrix(MTCData[,c("c1", "c2", "c3", "c4", "c5")])
28 na <- data.matrix(MTCData[, "na"])
29 #Class when running model at class level
30 class <- 1:max(c,na.rm = TRUE)
31 nt <- max(c, na.rm=TRUE)
32 nc <- max(class)
```

```
1 ns <- nrow(n)
2 ns.a <- 84 #studies reporting arm-level data
3 ns.a1 <- 38 #pCFB studies
4 ns.a2 <- 13 #CFB studies
5 ns.t2 <- 6 #2-arm studies reporting contrasts
6 ref <- 1 #reference treatment
7 #
8 initv1 <- list(direct=0, D=c(NA,rep(0,nc-1)), mu=rep(0,ns.a), sd=1)
9 initv2 <- list(direct=0.05, D=c(NA,rep(-1.2,nc-1)), mu=rep(0.5,ns.a), sd=3)
10 #####
11 #
12 # Check which notes to split
13 #
14 library(gemtc)
15 ns.data<-mtc.data.studyrow(MTCDData,
16                             armVars=c('treatment'='c'),
17                             nArmsVar='na',
18                             studyVars=c(),
19                             studyNames=MTCDData$studyid,
20                             treatmentNames=NA,
21                             patterns=c('%s', '%s%d'))
22 net<-mtc.network(data.ab=ns.data,description="Efficacy_trt")
23 ## Print which nodes to split
24 splitcomps<-mtc.nodesplit.comparisons(net)
25 print(splitcomps)
26 #
27 #####
28 # NODE-SPLITTING ROUTINE
29 #####
30 #
31 #
```

```
1 # Define nodes to split
2 pair<-splitcomps
3 pair
4 # Run node split models
5 for(j in 1:length(pair[,1])){
6   print(pair[j,])
7
8   k.ind <- PairXY(treat=c,na=na[,1],pair=as.numeric(pair[j,]))
9
10  # Setup subdirectory to hold results for each node-split
11  dir.create(paste("REFCENode",pair[j,1],"_",pair[j,2],sep=""))
12
13  # Build data file: stored in the working directory as "data.txt"
14  bugs.data(list("n"=n,"x"=x,"x.sd"=x.sd,"y"=y,"y.sd"=y.sd,
15              "CFB"=CFB,"CFB.sd"=CFB.sd,"pCFB"=pCFB,"pCFB.sd"=pCFB.sd,
16              "y.T"=y.T,"se.T"=se.T,"corr" = corr[,1],
17              "t"=c, "class"=class,
18              "na" = na[,1], "ns.a" = ns.a, "ns.a1" = ns.a1, "ns.a2" = ns.a2,
19              "nt" = nt, "nc" = nc, "ns" = ns, "ns.t2" = ns.t2,
20              "ref" = ref, "pair" = as.numeric(pair[j,]), "k.ind" = k.ind )
21
22  # Call OpenBUGS
23  #
24  bugs(data = "data.txt",
25       inits = list(initv1,initv2),
26       #inits = list(initv1),
27       parameters.to.save = c("direct", "d", "prob","totresdev","indirect","sd"),
28       model.file = "rse fce node-splitR2_v2_efficacy_class.txt",
29       n.chains = 2,
30       n.iter = 120000,
31       n.burnin = 40000,
```

```
1     n.thin = 1,
2     OpenBUGS.pgm = bd,
3     debug = FALSE,
4     save.history = TRUE,
5     useWINE=FALSE)
6 #
7 # Copy input and output files to relevant directory
8 file.copy("data.txt", paste("REFCENode",pair[j,1],"_",pair[j,2],"/data.txt",sep=""),
9 overwrite=TRUE)
10 file.copy(paste(tempdir(),"/log.odc",sep=""),
11 paste(pathname,"/REFCENode",pair[j,1],"_",pair[j,2],"/log.odc",sep=""), overwrite=TRUE)
12 file.copy(paste(tempdir(),"/log.txt",sep=""),
13 paste(pathname,"/REFCENode",pair[j,1],"_",pair[j,2],"/log.txt",sep=""), overwrite=TRUE)
14 file.copy(paste(tempdir(),"/inits1.txt",sep=""),
15 paste(pathname,"/REFCENode",pair[j,1],"_",pair[j,2],"/inits1.txt",sep=""), overwrite=TRUE)
16 file.copy(paste(tempdir(),"/script.txt",sep=""),
17 paste(pathname,"/REFCENode",pair[j,1],"_",pair[j,2],"/script.txt",sep=""), overwrite=TRUE)
18 #
19 # REPEAT FOR ALL OTHER NODES
20 }
```

2A.3.2 OpenBUGS Code

```
22 model{                                # *** PROGRAM STARTS
23 for(i in 1:ns.a){                      # LOOP THROUGH STUDIES WITH ARM DATA
24     w[i,1] <- 0                         # adjustment for multi-arm trials is zero for control arm
25     delta[i,1] <- 0                    # treatment effect is zero for control arm
26     mu[i] ~ dnorm(0,.0001)             # vague priors for all trial baselines
27 }
28
29 # trials reporting percent CFB
30 for(i in 1:ns.a1){                     # LOOP THROUGH STUDIES WITH %CFB ARM DATA
31     for (k in 1:na[i]) {               # LOOP THROUGH ARMS
32         pCFB.se[i,k.ind[i,k]] <- pCFB.sd[i,k.ind[i,k]]/sqrt(n[i,k.ind[i,k]]) # calculate standard error
33         pCFB.var[i,k.ind[i,k]] <- pow(pCFB.se[i,k.ind[i,k]],2) # calculate variances
```

```
1     pCFB.prec[i,k.ind[i,k]] <- 1/pCFB.var[i,k.ind[i,k]]    # set precisions
2     pCFB[i,k.ind[i,k]] ~ dnorm(theta[i,k],pCFB.prec[i,k.ind[i,k]]) # normal likelihood
3
4     theta[i,k] <- mu[i] + delta[i,k] # model for linear predictor
5
6     #Deviance contribution
7     dev[i,k] <- (pCFB[i,k.ind[i,k]]-theta[i,k])*(pCFB[i,k.ind[i,k]]-
8 theta[i,k])*pCFB.prec[i,k.ind[i,k]]
9
10    split[i,k] <- equals(t[i,k.ind[i,1]], pair[1]) * equals(t[i,k.ind[i,k]], pair[2]) -
11 equals(t[i,k.ind[i,1]], pair[2]) * equals(t[i,k.ind[i,k]], pair[1])
12    }
13    resdev[i] <- sum(dev[i,1:na[i]])
14 }
15 # trials reporting CFB + B    # LOOP THROUGH STUDIES WITH CFB+B ARM DATA
16 for(i in (ns.a1+1):(ns.a1+ns.a2)){
17   for (k in 1:na[i]) {      # LOOP THROUGH ARMS
18     x.se[i,k.ind[i,k]] <- x.sd[i,k.ind[i,k]]/sqrt(n[i,k.ind[i,k]])    # calculate standard error
19     x.var[i,k.ind[i,k]] <- pow(x.se[i,k.ind[i,k]],2) # calculate variances
20     x.prec[i,k.ind[i,k]] <- 1/x.var[i,k.ind[i,k]]    # set precisions
21     x[i,k.ind[i,k]] ~ dnorm(mu.X[i,k],x.prec[i,k.ind[i,k]]) # indpt normal likelihood for baseline
22 mean
23     mu.X[i,k] ~ dnorm(0,.0001)I(0,)          # flat prior for baseline mean in likelihood
24
25     CFB.se[i,k.ind[i,k]] <- CFB.sd[i,k.ind[i,k]]/sqrt(n[i,k.ind[i,k]])    # calculate standard error
26     CFB.var[i,k.ind[i,k]] <- pow(CFB.se[i,k.ind[i,k]],2) # calculate variances
27     CFB.prec[i,k.ind[i,k]] <- 1/CFB.var[i,k.ind[i,k]]    # set precisions
28     mu.CFB[i,k] <- mu.X[i,k]*(theta[i,k]/100)# calculate mean for CFB likelihood
29     CFB[i,k.ind[i,k]] ~ dnorm(mu.CFB[i,k],CFB.prec[i,k.ind[i,k]]) # indpt normal likelihood for
30 baseline mean
31
32     theta[i,k] <- mu[i] + delta[i,k] # model for linear predictor
```



```
1
2     #Deviance contribution
3     dev[i,k] <- (CFB[i,k.ind[i,k]]-mu.CFB[i,k])*(CFB[i,k.ind[i,k]]-
4 mu.CFB[i,k])*CFB.prec[i,k.ind[i,k]]
5     split[i,k] <- equals(t[i,k.ind[i,1]], pair[1]) * equals(t[i,k.ind[i,k]], pair[2]) -
6 equals(t[i,k.ind[i,1]], pair[2]) * equals(t[i,k.ind[i,k]], pair[1])
7     }
8     resdev[i] <- sum(dev[i,1:na[i]])
9     }
10 # trials reporting B + F
11 for(i in (ns.a1+ns.a2+1):ns.a){     # LOOP THROUGH STUDIES WITH B+F ARM DATA
12   for (k in 1:na[i]) {           # LOOP THROUGH ARMS
13     #Calculate standard errors
14     x.se[i,k.ind[i,k]] <- x.sd[i,k.ind[i,k]]/sqrt(n[i,k.ind[i,k]])
15     y.se[i,k.ind[i,k]] <- y.sd[i,k.ind[i,k]]/sqrt(n[i,k.ind[i,k]])
16     #Set precision matrix
17     Sigma[i,k.ind[i,k],1,1]<-pow(x.se[i,k.ind[i,k]],2)
18     Sigma[i,k.ind[i,k],2,2]<-pow(y.se[i,k.ind[i,k]],2)
19     Sigma[i,k.ind[i,k],1,2]<-corr[i]*x.se[i,k.ind[i,k]]*y.se[i,k.ind[i,k]]
20     Sigma[i,k.ind[i,k],2,1]<-Sigma[i,k.ind[i,k],1,2]
21     Prec[i,k.ind[i,k],1:2,1:2]<-inverse(Sigma[i,k.ind[i,k],1:2,1:2])
22     #Set up vector for baseline and follow-up means
23     y.XY[i,k.ind[i,k],1]<-x[i,k.ind[i,k]]
24     y.XY[i,k.ind[i,k],2]<-y[i,k.ind[i,k]]
25
26     # Bivariate normal likelihood for baseline and follow-up
27     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])
28     mu.XY[i,k,2]<- mu.XY[i,k,1]*(1-(theta[i,k]/100))
29     mu.XY[i,k,1] ~ dnorm(0,.0001)|(0,)           # flat prior for baseline mean in likelihood
30
31     theta[i,k] <- mu[i] + delta[i,k] # model for linear predictor
32
```

```
1 #Deviance contribution
2 for (j in 1:2){
3     diff[i,k,j]<- y.XY[i,k.ind[i,k],j]-mu.XY[i,k,j]
4     z[i,k,j]<- inprod(Prec[i,k.ind[i,k],j,1:2],diff[i,k,1:2])
5 }
6 dev[i,k]<-inprod(diff[i,k,1:2],z[i,k,1:2])
7
8 split[i,k] <- equals(t[i,k.ind[i,1]], pair[1]) * equals(t[i,k.ind[i,k]], pair[2]) -
9 equals(t[i,k.ind[i,1]], pair[2]) * equals(t[i,k.ind[i,k]], pair[1])
10 }
11 resdev[i] <- sum(dev[i,1:na[i]])
12 }
13 # 2-arm trials reporting contrasts (e.g., split-face trials)
14 for(i in (ns.a+1):(ns.a+ns.t2)){ # LOOP THROUGH STUDIES WITH TRIAL DATA
15     w[i,1] <- 0 # adjustment for multi-arm trials is zero for control arm
16     delta[i,1] <- 0 # treatment effect is zero for control arm
17     var[i,2] <- pow(se.T[i,2],2) # calculate variances
18     prec[i,2] <- 1/var[i,2] # set precisions
19     y.T[i,2] ~ dnorm(delta[i,2],prec[i,2]) # normal likelihood
20     #Deviance contribution
21     dev[i,2] <- (y.T[i,2]-delta[i,2])* (y.T[i,2]-delta[i,2])* prec[i,2]
22     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],
23 pair[1])
24
25 # summed residual deviance contribution for this trial
26 resdev[i] <- dev[i,2]
27 }
28 #RE Model (ARM AND TRIAL DATA)
29 for(i in 1:ns){ # LOOP THROUGH STUDIES WITH ARM DATA
30     for (k in 2:na[i]) { # LOOP THROUGH ARMS
31         # trial-specific RE distributions
32         delta[i,k] ~ dnorm(md[i,k],taud[i,k])
```

```
1      # mean of RE distributions, with multi-arm trial correction
2      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]
3      # precision of RE distributions (with multi-arm trial correction)
4      taud[i,k] <- tau *2*(k-1)/k
5      # adjustment, multi-arm RCTs
6      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 *
7 split[i,k] )
8      # cumulative adjustment for multi-arm trials
9      sw[i,k] <- sum(w[i,1:k-1])/(k-1)
10     }
11  }
12
13  totesdev <- sum(resdev[])      #Total Residual Deviance
14  # Reference treatment currently Placebo (ref=1)
15  d[ref]<-0      # treatment effect is zero for reference treatment
16  D[class[ref]]<-0
17  # priors for mean class effect
18  for (j in 2:nc){
19    D[j]~dnorm(0,.0001)
20  }
21  # treatment effect = mean class effect
22  for (j in 2:nt){
23    d[j] <- D[class[j]]
24  }
25  direct ~ dnorm(0,.0001)      # vague prior for direct comparison parameter
26  indirect <- mean.diff[pair[1], pair[2]]
27  #calculate difference between direct and lor
28  diff.ns <- direct - indirect
29  # calculate p-value
30  prob <- step(diff.ns)
31  #
```

```
1 sd ~ dunif(0,25) # vague prior for between-trial SD
2 tau <- pow(sd,-2) # between-trial precision = (1/between-trial variance)
3 #
4 # pairwise mean differences for all possible pair-wise comparisons
5 for (c in 1:(nt-1)) {
6   for (k in (c+1):nt) {
7     mean.diff[c,k] <- d[k]-d[c]
8     mean.diff[k,c] <- -mean.diff[c,k]
9   }
10 }
11 } # *** PROGRAM ENDS
```

Discontinuation for any reason

1A.4: Discontinuation for any reason, base-case model (WinBUGS)

```
14 model{
15   for(i in 1:ns){ # LOOP OVER ALL STUDIES
16     mu[i] ~ dnorm(0,.0001) # vague priors for all trial baselines
17     for (k in 1:na[i]){ # LOOP OVER ARMS
18       r[i,k] ~ dbin(p[i,k],n[i,k]) # binomial likelihood
19       logit(p[i,k]) <- mu[i] + d[t[i,k]] - d[t[i,1]] # model for linear predictor
20       rhat[i,k] <- p[i,k] * n[i,k] # expected value of the numerators
21       #Deviance contribution
22       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]) -
23 log(n[i,k]-rhat[i,k])))
24     }
25     # Summed residual deviance contribution for this trial
26     resdev[i] <- sum(dev[i,1:na[i]])
27   }
28   totresdev <- sum(resdev[]) # Total Residual Deviance
29 #
30 # Reference treatment
```

```
1 d[ref]<-0    # treatment effect is zero for reference treatment
2 D[class[ref]]<-0
3 #
4 # vague prior for class effects
5 for (j in 2:nc){
6   D[j] ~ dnorm(0, .0001)
7 }
8 for (j in 2:nt){
9   d[j] <- D[class[j]]
10 }
11 #
12 # pairwise ORs and LORs for all possible pair-wise treatment comparisons
13 for (c in 1:(nt-1)){
14   for (k in (c+1):nt){
15     or[c,k] <- exp(d[k] - d[c])
16     lor[c,k] <- (d[k]-d[c])
17   }
18 }
19 #
20 # pairwise differences for classes
21 for (c in 1:(nc-1)){
22   for (k in (c+1):nc){
23     diffClass[c,k] <- D[k] - D[c]
24     orClass[c,k] <- exp(D[k] - D[c])
25   }
26 }
27 # ranking on relative scale
28 for (k in 1:nc){
29   rkClass[k] <- rank(D[,k])
30   bestClass[k] <- equals(rkClass[k],1) # Smallest is best (i.e. rank 1)
31   # prob class k is h-th best, prob[1,k]=best[k]
```

```
1   for (h in 1:nc) { probClass[h,k] <- equals(rkClass[k],h) }
2   }
3   #
4   # ranking on relative scale - males
5   for (k in 1:19){ D.m[k] <- D[k]}
6   for (k in 20:(nc-2)){ D.m[k] <- D[k+2]}
7   for (k in 1:(nc-2)){
8     rk.m[k] <- rank(D.m[,k])      # assumes lower values are "good"
9     best.m[k] <- equals(rk.m[k],1)  #calculate probability that treat k is best
10    # calculate probability that treat k is h-th best
11    for (h in 1:nc){ prob.m[h,k] <- equals(rk.m[k],h) }
12  }
13 }                                # *** PROGRAM ENDS
```

1A.5: Discontinuation for any reason, node-splitting, class-level

1A.5.1: R Code (requires R2OpenBUGS package)

```
16 #####
17 # Node-splitting for Acne Guideline - Discontinuation (any)
18 # R script to run node-split for the MTC Random study effects, fixed
19 # class effects model using OpenBUGS
20 #
21 # Uses R2OpenBUGS package
22 #
23 # Discontinuation (any reason)
24 # 1. Need to include in the working directory the following files:
25 #     Disc any_UK.txt --- text file with data
26 #     fse fce node-splitR2_v3.txt --- text file holding BUGS code
27 #
28 # 2. Output files will be
29 #     coda1.txt --- holds coda output
30 #     codaIndex.txt --- holds indexes to coda output
```

```
1 #      data.txt --- holds all data as used by BUGS
2 #      log.odc and log.txt --- hold WinBUGS output
3 #      inits1.txt --- holds initial values as read by BUGS
4 #      script.txt --- BUGS script file with all commands to execute
5 #
6 # 3. Output files for each node should be transferred to a new directory
7 #   as they will be overwritten in each new run
8 #
9 # 4. You may need to edit the OpenBUGS location 'bd'
10 #
11 # 5. You will need to edit the working directory 'pathname'
12 #   to suit your computer settings
13 #
14 # 6. Run script file
15 #
16 # 7. To repeat for other node-splits need to change variable 'pair'
17 #   and edit output file names
18 #
19 #####
20 #
21 # Declare the directory where OpenBUGS is found in this computer
22 bd <- "C:/Program Files (x86)/OpenBUGS/OpenBUGS323/OpenBUGS.exe"
23 #
24 # Declare working directory
25 pathname <- "C:/Acne/M2M/Disc Any/"
26 setwd(pathname)
27 #
28 # load package to call OpenBUGS
29 library(R2OpenBUGS)
30 #
31 # LOAD DATA MANIPULATING FUNCTIONS:
```

```
1 #
2 PairXY <- function(treat, na, pair)
3   # Check if pair(X,Y) in row i of data
4   # and reorder treatments in trial as appropriate
5   {
6     N <- nrow(treat)
7     multi <- rep(NA,length(na))
8     split.ind <- matrix(nrow=nrow(treat),ncol=ncol(treat))
9     split.ind1 <- matrix(nrow=nrow(treat),ncol=ncol(treat))
10    split.ind2 <- matrix(nrow=nrow(treat),ncol=ncol(treat))
11    spliti <- rep(NA,length(na))
12    split1i <- rep(NA,length(na))
13    split2i <- rep(NA,length(na))
14    pair1 <- matrix(nrow=nrow(treat),ncol=ncol(treat))
15    pair2 <- matrix(nrow=nrow(treat),ncol=ncol(treat))
16    k.ind <- matrix(nrow=nrow(treat),ncol=ncol(treat))
17    for (i in 1:N) {
18      # is trial i a multiarm trial?
19      multi[i] <- 1*(na[i]>2)
20      for (k in 1:na[i]){
21        # which arms contain a treatment in the pair?
22        split.ind[i,k] <- 1*(treat[i,k]==pair[1])+1*(treat[i,k]==pair[2])
23        # which arms contain the treatment in pair[1]?
24        split.ind1[i,k] <- 1*(treat[i,k]==pair[1])
25        # which arms contain the treatment in pair[2]?
26        split.ind2[i,k] <- 1*(treat[i,k]==pair[2])
27      }
28      # does trial i contain multiples of pair[1]?
29      split1i[i] <- 1*(sum(split.ind1[i,1:na[i]])>1)
30      # does trial i contain multiples of pair[2]?
31      split2i[i] <- 1*(sum(split.ind2[i,1:na[i]])>1)
```



```
1 # does trial i contain both treatments in the pair?
2 # (minus duplicates in multiarm trials that have one treatment (only) in pair)
3 split1[i] <- 1*((sum(split.ind[i,1:na[i]])-split1[i]*sum(split.ind1[i,1:na[i]])-split1[i]*
4 split2[i]*sum(split.ind2[i,1:na[i]])-split2[i])>1)
5 for (k in 1:na[i]) {
6 # which arms contain the first element in the pair
7 pair1[i,k] <- k*(1*(treat[i,k]==pair[1]))
8 # which arms contain the second element in the pair
9 pair2[i,k] <- k*(1*(treat[i,k]==pair[2]))
10 }
11 for (k in 1:na[i]) {
12 # reposition order of arms within a trial according to node being split
13 # k.ind ensures a treatment in the pair is in the baseline arm, where the
14 # multi-arm trial contains both treatments in the pair
15 # If a multi-arm trial does not contain the node, arm order stays the same
16 k.ind[i,k]<-(k*((1-multi[i])+multi[i]*(1-split1[i])+multi[i]*split1[i]*(1*(split.ind[i,1]==1))))
17
18 # If a multi-arm trial contains the node, the treatment in pair[1] is not duplicated in
19 trial,
20 # the baseline arm does not contain a treatment in the node, and the treatment
21 # in arm k is pair[1], make this treatment baseline treatment
22 + multi[i]*split1[i]*(1-split1[i])*(1-(1*(split.ind[i,1]==1)))*(1*(treat[i,k]==pair[1]))
23
24 # If a multi-arm trial contains the node, the treatment in pair[1] is duplicated in
25 trial,
26 # the treatment in pair[2] is not duplicated in trial, the baseline arm does not
27 contain
28 # a treatment in the node, and the treatment in arm k is pair[2], make this
29 treatment baseline treatment
30 + multi[i]*split1[i]*split1[i]*(1-split2[i])*(1-
31 (1*(split.ind[i,1]==1)))*(1*(treat[i,k]==pair[2]))
32
33 # If a multi-arm trial contains the node, the treatment in pair[1] is not duplicated in
34 trial,
```

```
1          # the baseline arm does not contain a treatment in the node, and k is baseline
2 arm,
3          # move treatment to come after baseline treatment
4          + sum(pair1[i,1:na[i]])*(1-split1i[i])*multi[i]*spliti[i]*(1-
5 (1*(split.ind[i,1]==1)))*(1*(k==1))
6
7          # If a multi-arm trial contains the node, the treatment in pair[1] is duplicated in
8 trial,
9          # the treatment in pair[2] is not duplicated in trial, the baseline arm does not
10 contain a
11          # treatment in the node, and k is baseline arm, move treatment to come after
12 baseline treatment
13          + sum(pair2[i,1:na[i]])*split1i[i]*(1-split2i[i])*multi[i]*spliti[i]*(1-
14 (1*(split.ind[i,1]==1)))*(1*(k==1))
15
16          # If a multi-arm trial contains the node, the treatment in pair[1] are not duplicated
17 in trial,
18          # the baseline arm does not contain a treatment in the node, k is NOT baseline
19 arm,
20          # and treatment in arm k is NOT pair[1], arm order stays the same
21          + k*multi[i]*spliti[i]*(1-split1i[i]*(1-(1*(split.ind[i,1]==1)))*(1-(1*(k==1)))*(1-
22 (1*(treat[i,k]==pair[1]))))
23
24          # If a multi-arm trial contains the node, the treatment in pair[1] is duplicated in
25 trial,
26          # the treatment in pair[2] is not duplicated in trial, the baseline arm does not
27 contain a treatment in the node, k is NOT baseline arm,
28          # and treatment in arm k is NOT pair[2], arm order stays the same
29          + k*multi[i]*spliti[i]*split1i[i]*(1-split2i[i]*(1-(1*(split.ind[i,1]==1)))*(1-(1*(k==1)))*(1-
30 (1*(treat[i,k]==pair[2]))))
31    }
32  }
33  k.ind
34 }
35 #####
36 #
```

```
1 # load data for MTC
2 MTCDData <- read.table("Disc any_UK.txt", header=TRUE)
3 r <- data.matrix(MTCDData[,c("r1", "r2", "r3", "r4", "r5")])
4 n <- data.matrix(MTCDData[,c("n1", "n2", "n3", "n4", "n5")])
5 c <- data.matrix(MTCDData[,c("c1", "c2", "c3", "c4", "c5")])
6 na <- data.matrix(MTCDData[, "na"])
7 #Class when running model at class level
8 class <- 1:max(c,na.rm = TRUE)
9 nt <- max(c, na.rm=TRUE)
10 nc <- max(class)
11 ns <- nrow(r)
12 ref <- 1 #reference treatment
13 #
14 # define initial values
15 initv1 <- list(direct=0, D=c(NA,rep(0,nc-1)), mu=rep(0,ns))
16 initv2 <- list(direct=0.05, D=c(NA,rep(-1.2,nc-1)), mu=rep(0.5,ns))
17 #####
18 #
19 # Check which notes to split
20 #
21 library(gemtc)
22 ns.data<-mtc.data.studyrow(MTCDData,
23     armVars=c('treatment'='c', 'responders'='r', 'sampleSize'='n'),
24     nArmsVar='na',
25     studyVars=c(),
26     studyNames=MTCDData$studyid,
27     treatmentNames=NA,
28     patterns=c('%s', '%s%d'))
29 net<-mtc.network(data.ab=ns.data,description="Disc any_trt")
30 ## Print which nodes to split
31 splitcomps<-mtc.nodesplit.comparisons(net)
```

```
1 print(splitcomps)
2 #
3 #####
4 ##
5 # NODE-SPLITTING ROUTINE
6 #####
7 ##
8 #
9 #
10 # Define nodes to split
11 pair<-splitcomps
12 pair
13 # Run node split models
14 for(j in 1:length(pair[,1])){
15   print(pair[j,])
16
17   k.ind <- PairXY(treat=c,na=na[,1],pair=as.numeric(pair[j,]))
18
19   # Setup subdirectory to hold results for each node-split
20   dir.create(paste("REFCENode",pair[j,1],"_",pair[j,2],sep=""))
21   # Build data file: stored in the working directory as "data.txt"
22   bugs.data(list("r"=r,"n"=n,"t"=c, "class"=class,
23                 "na" = na[,1], "nt" = nt, "nc" = nc, "ns" = ns, "ref" = ref,
24                 "pair" = as.numeric(pair[j,]), "k.ind" = k.ind))
25
26   # Call OpenBUGS
27   #
28   bugs(data = "data.txt",
29         inits = list(initv1,initv2),
30         #inits = list(initv1),
31         parameters.to.save = c("direct", "d", "prob","totresdev","indirect"),
32         model.file = "fse fce node-splitR2_v3.txt",
```

```
1   n.chains = 2,
2   n.iter = 120000,      #including burn-in iterations
3   n.burnin = 40000,
4   n.thin = 1,
5   OpenBUGS.pgm = bd,
6   debug = FALSE,
7   save.history = TRUE,
8   useWINE=FALSE)
9   #
10  # Copy input and output files to relevant directory
11  file.copy("data.txt", paste("REFCENode",pair[j,1],"_",pair[j,2],"/data.txt",sep=""),
12  overwrite=TRUE)
13  file.copy(paste(tempdir(),"/log.odc",sep=""),
14  paste(pathname,"/REFCENode",pair[j,1],"_",pair[j,2],"/log.odc",sep=""), overwrite=TRUE)
15  file.copy(paste(tempdir(),"/log.txt",sep=""),
16  paste(pathname,"/REFCENode",pair[j,1],"_",pair[j,2],"/log.txt",sep=""), overwrite=TRUE)
17  file.copy(paste(tempdir(),"/inits1.txt",sep=""),
18  paste(pathname,"/REFCENode",pair[j,1],"_",pair[j,2],"/inits1.txt",sep=""), overwrite=TRUE)
19  file.copy(paste(tempdir(),"/script.txt",sep=""),
20  paste(pathname,"/REFCENode",pair[j,1],"_",pair[j,2],"/script.txt",sep=""), overwrite=TRUE)
21  #
22  # REPEAT FOR ALL OTHER NODES
23 }
```

2A.5.2 OpenBUGS Code

```
25 model{
26   for(i in 1:ns){      # LOOP OVER ALL STUDIES
27     delta[i,1] <- 0 # treatment effect is zero for control arm
28     mu[i] ~ dnorm(0,.0001) # vague priors for all trial baselines
29     for (k in 1:na[i]){ # LOOP OVER ARMS
30       r[i,k.ind[i,k]] ~ dbin(p[i,k],n[i,k.ind[i,k]]) # binomial likelihood
31       logit(p[i,k]) <- mu[i] + delta[i,k] # model for linear predictor
32       rhat[i,k] <- p[i,k] * n[i,k.ind[i,k]] # expected value of the numerators
33     }
34     #Deviance contribution
```

```
1      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]]-
2 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])))
3      split[i,k] <- equals(t[i,k.ind[i,1]], pair[1]) * equals(t[i,k.ind[i,k]], pair[2]) -
4 equals(t[i,k.ind[i,1]], pair[2]) * equals(t[i,k.ind[i,k]], pair[1])
5      }
6      # Summed residual deviance contribution for this trial
7      resdev[i] <- sum(dev[i,1:na[i]])
8      for (k in 2:na[i]) {      # FE model for treatment effects
9          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]
10         }
11     }
12 totresdev <- sum(resdev[])      # Total Residual Deviance
13 #
14 d[ref]<-0      # treatment effect is zero for reference treatment
15 D[class[ref]]<-0
16 # vague prior for class effects
17 for (j in 2:nc){
18     D[j] ~ dnorm(0, .0001)
19 }
20 for (j in 2:nt){
21     d[j] <- D[class[j]]
22 }
23 direct ~ dnorm(0,.0001)      # vague prior for direct comparison parameter
24 indirect <- lor[pair[1], pair[2]]
25 #calculate difference between direct and lor
26 diff <- direct - indirect
27 # calculate p-value
28 prob <- step(diff)
29 #
30 # pairwise ORs and LORs for all possible pair-wise comparisons
31 for (c in 1:(nt-1)){
32     for (k in (c+1):nt){
```

```
1      or[c,k] <- exp(d[k] - d[c])
2      lor[c,k] <- (d[k]-d[c])
3      lor[k,c] <- -lor[c,k]
4      }
5  }
6 }      # *** PROGRAM ENDS
7
```

Discontinuation due to side effects

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

```
10 model{
11 for(i in 1:ns){      # LOOP OVER ALL STUDIES
12   mu[i] ~ dnorm(0,.0001)      # vague priors for all trial baselines
13   for (k in 1:na[i]){      # LOOP OVER ARMS
14     r[i,k] ~ dbin(p[i,k],n[i,k]) # binomial likelihood
15     logit(p[i,k]) <- mu[i] + d[t[i,k]] - d[t[i,1]] # model for linear predictor
16     rhat[i,k] <- p[i,k] * n[i,k] # expected value of the numerators
17     #Deviance contribution
18     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]) -
19 log(n[i,k]-rhat[i,k])))
20   }
21   # Summed residual deviance contribution for this trial
22   resdev[i] <- sum(dev[i,1:na[i]])
23 }
24 totresdev <- sum(resdev[])      # Total Residual Deviance
25 #
26 # Reference treatment
27 d[ref]<-0      # treatment effect is zero for reference treatment
28 D[class[ref]]<-0
29 #
30 # vague prior for class effects
```

```
1 for (j in 2:nc){
2   D[j] ~ dnorm(0, .0001)
3 }
4 for (j in 2:nt){
5   d[j] <- D[class[j]]
6 }
7 #
8 # pairwise ORs and LORs for all possible pair-wise treatment comparisons
9 for (c in 1:(nt-1)){
10  for (k in (c+1):nt){
11    or[c,k] <- exp(d[k] - d[c])
12    lor[c,k] <- (d[k]-d[c])
13  }
14 }
15 #
16 # pairwise differences for classes
17 for (c in 1:(nc-1)){
18  for (k in (c+1):nc){
19    diffClass[c,k] <- D[k] - D[c]
20    orClass[c,k] <- exp(D[k] - D[c])
21  }
22 }
23 # ranking on relative scale
24 for (k in 1:nc){
25  rkClass[k] <- rank(D[,k])
26  bestClass[k] <- equals(rkClass[k],1) # Smallest is best (i.e. rank 1)
27  # prob class k is h-th best, prob[1,k]=best[k]
28  for (h in 1:nc) { probClass[h,k] <- equals(rkClass[k],h) }
29 }
30 #
31 # ranking on relative scale - males
```



```
1 for (k in 1:11){ D.m[k] <- D[k]}
2 for (k in 12:(nc-2)){ D.m[k] <- D[k+2]}
3 for (k in 1:(nc-2)){
4   rk.m[k] <- rank(D.m[,k])      # assumes lower values are "good"
5   best.m[k] <- equals(rk.m[k],1) #calculate probability that treat k is best
6   # calculate probability that treat k is h-th best
7   for (h in 1:nc){ prob.m[h,k] <- equals(rk.m[k],h) }
8 }
9 }                               # *** PROGRAM ENDS
```

1A.7: Discontinuation due to side effects, node-splitting, class-level

1A.7.1: R Code (requires R2OpenBUGS package)

```
12 #####
13 # Node-splitting for Acne Guideline - Discontinuation (due to SE)
14 # R script to run node-split for the MTC Fixed study effects, fixed
15 # class effects model using OpenBUGS
16 #
17 # Uses R2OpenBUGS package
18 #
19 # Discontinuation (due to SE)
20 # 1. Need to include in the working directory the following files:
21 #     Disc se.txt --- text file with data
22 #     fse fce node-splitR2_v3.txt --- text file holding BUGS code
23 #
24 # 2. Output files will be
25 #     data.txt --- holds all data as used by BUGS
26 #     log.odc and log.txt --- hold WinBUGS output
27 #     inits1.txt --- holds initial values as read by BUGS
28 #     script.txt --- BUGS script file with all commands to execute
29 #
30 # 3. Output files for each node should be transferred to a new directory
```

```
1 # as they will be overwritten in each new run
2 #
3 # 4. You may need to edit the OpenBUGS location 'bd'
4 #
5 # 5. You will need to edit the working directory 'pathname'
6 # to suit your computer settings
7 #
8 # 6. Run script file
9 #
10 #####
11 #
12 # Declare the directory where OpenBUGS is found in this computer
13 bd <- "C:/Program Files (x86)/OpenBUGS/OpenBUGS323/OpenBUGS.exe"
14 #
15 # Declare working directory
16 pathname <- "C:/ Acne/M2M/Disc SE/"
17 setwd(pathname)
18 #
19 # load package to call OpenBUGS
20 library(R2OpenBUGS)
21 #
22 # LOAD DATA MANIPULATING FUNCTIONS:
23 #
24 PairXY <- function(treat, na, pair)
25 # Check if pair(X,Y) in row i of data
26 # and reorder treatments in trial as appropriate
27 {
28 N <- nrow(treat)
29 multi <- rep(NA,length(na))
30 split.ind <- matrix(nrow=nrow(treat),ncol=ncol(treat))
31 split.ind1 <- matrix(nrow=nrow(treat),ncol=ncol(treat))
```

```
1 split.ind2 <- matrix(nrow=nrow(treat),ncol=ncol(treat))
2 spliti <- rep(NA,length(na))
3 split1i <- rep(NA,length(na))
4 split2i <- rep(NA,length(na))
5 pair1 <- matrix(nrow=nrow(treat),ncol=ncol(treat))
6 pair2 <- matrix(nrow=nrow(treat),ncol=ncol(treat))
7 k.ind <- matrix(nrow=nrow(treat),ncol=ncol(treat))
8 for (i in 1:N) {
9   # is trial i a multiarm trial?
10  multi[i] <- 1*(na[i]>2)
11  for (k in 1:na[i]){
12    # which arms contain a treatment in the pair?
13    split.ind[i,k] <- 1*(treat[i,k]==pair[1])+1*(treat[i,k]==pair[2])
14    # which arms contain the treatment in pair[1]?
15    split.ind1[i,k] <- 1*(treat[i,k]==pair[1])
16    # which arms contain the treatment in pair[2]?
17    split.ind2[i,k] <- 1*(treat[i,k]==pair[2])
18  }
19  # does trial i contain multiples of pair[1]?
20  split1i[i] <- 1*(sum(split.ind1[i,1:na[i]])>1)
21  # does trial i contain multiples of pair[2]?
22  split2i[i] <- 1*(sum(split.ind2[i,1:na[i]])>1)
23  # does trial i contain both treatments in the pair?
24  # (minus duplicates in multiarm trials that have one treatment (only) in pair)
25  spliti[i] <- 1*((sum(split.ind[i,1:na[i]])-split1i[i]*(sum(split.ind1[i,1:na[i]])-split1i[i])-
26 split2i[i]*(sum(split.ind2[i,1:na[i]])-split2i[i]))>1)
27  for (k in 1:na[i]) {
28    # which arms contain the first element in the pair
29    pair1[i,k] <- k*(1*(treat[i,k]==pair[1]))
30    # which arms contain the second element in the pair
31    pair2[i,k] <- k*(1*(treat[i,k]==pair[2]))
```

```
1   }
2   for (k in 1:na[i]) {
3     # reposition order of arms within a trial according to node being split
4     # k.ind ensures a treatment in the pair is in the baseline arm, where the
5     # multi-arm trial contains both treatments in the pair
6     # If a multi-arm trial does not contain the node, arm order stays the same
7     k.ind[i,k]<-(k*((1-multi[i])+multi[i]*(1-split[i])+multi[i]*split[i]*(1*(split.ind[i,1]==1))))
8
9     # If a multi-arm trial contains the node, the treatment in pair[1] is not duplicated in
10    trial,
11    # the baseline arm does not contain a treatment in the node, and the treatment
12    # in arm k is pair[1], make this treatment baseline treatment
13    + multi[i]*split[i]*(1-split1[i])*(1-(1*(split.ind[i,1]==1)))*(1*(treat[i,k]==pair[1]))
14
15    # If a multi-arm trial contains the node, the treatment in pair[1] is duplicated in
16    trial,
17    # the treatment in pair[2] is not duplicated in trial, the baseline arm does not
18    contain
19    # a treatment in the node, and the treatment in arm k is pair[2], make this
20    treatment baseline treatment
21    + multi[i]*split[i]*split1[i]*(1-split2[i])*(1-
22    (1*(split.ind[i,1]==1)))*(1*(treat[i,k]==pair[2]))
23
24    # If a multi-arm trial contains the node, the treatment in pair[1] is not duplicated in
25    trial,
26    # the baseline arm does not contain a treatment in the node, and k is baseline
27    arm,
28    # move treatment to come after baseline treatment
29    + sum(pair1[i,1:na[i]])*(1-split1[i])*multi[i]*split[i]*(1-
30    (1*(split.ind[i,1]==1)))*(1*(k==1))
31
32    # If a multi-arm trial contains the node, the treatment in pair[1] is duplicated in
33    trial,
34    # the treatment in pair[2] is not duplicated in trial, the baseline arm does not
35    contain a
```

```
1           # treatment in the node, and k is baseline arm, move treatment to come after
2 baseline treatment
3           + sum(pair2[i,1:na[i]])*split1i[i]*(1-split2i[i])*multi[i]*spliti[i]*(1-
4 (1*(split.ind[i,1]==1)))*(1*(k==1))
5
6           # If a multi-arm trial contains the node, the treatment in pair[1] are not duplicated
7 in trial,
8           # the baseline arm does not contain a treatment in the node, k is NOT baseline
9 arm,
10          # and treatment in arm k is NOT pair[1], arm order stays the same
11          + k*multi[i]*spliti[i]*(1-split1i[i])*(1-(1*(split.ind[i,1]==1)))*(1-(1*(k==1)))*(1-
12 (1*(treat[i,k]==pair[1])))
13
14          # If a multi-arm trial contains the node, the treatment in pair[1] is duplicated in
15 trial,
16          # the treatment in pair[2] is not duplicated in trial, the baseline arm does not
17 contain a treatment in the node, k is NOT baseline arm,
18          # and treatment in arm k is NOT pair[2], arm order stays the same
19          + k*multi[i]*spliti[i]*split1i[i]*(1-split2i[i])*(1-(1*(split.ind[i,1]==1)))*(1-(1*(k==1)))*(1-
20 (1*(treat[i,k]==pair[2])))
21    }
22  }
23  k.ind
24 }
25 #####
26 #
27 # load data for MTC
28 MTCDData <- read.table("Disc se_UK.txt", header=TRUE)
29 r <- data.matrix(MTCDData[,c("r1", "r2", "r3", "r4", "r5")])
30 n <- data.matrix(MTCDData[,c("n1", "n2", "n3", "n4", "n5")])
31 c <- data.matrix(MTCDData[,c("c1", "c2", "c3", "c4", "c5")])
32 na <- data.matrix(MTCDData[, "na"])
33 #Class when running model at class level
34 class <- 1:max(c,na.rm = TRUE)
35 nt <- max(c, na.rm=TRUE)
Acne vulgaris Supplement 3: TSU NMA software code (December 2020)
```

```
1 nc <- max(class)
2 ns <- nrow(r)
3 ref <- 1 #reference treatment
4 #
5 # define initial values
6 initv1 <- list(direct=0, D=c(NA,rep(0,nc-1)), mu=rep(0,ns))
7 initv2 <- list(direct=0.05, D=c(NA,rep(-1.2,nc-1)), mu=rep(0.5,ns))
8 #####
9 #
10 # Check which nodes to split
11 #
12 library(gemtc)
13 ns.data<-mtc.data.studyrow(MTCDData,
14                             armVars=c('treatment'='c', 'responders'='r', 'sampleSize'='n'),
15                             nArmsVar='na',
16                             studyVars=c(),
17                             studyNames=MTCDData$study,
18                             treatmentNames=NA,
19                             patterns=c('%s', '%s%d'))
20 net<-mtc.network(data.ab=ns.data,description="Disc se_trt")
21 ## Print which nodes to split
22 splitcomps<-mtc.nodesplit.comparisons(net)
23 print(splitcomps)
24 #
25 #####
26 ##
27 # NODE-SPLITTING ROUTINE
28 #####
29 ##
30 #
31 #
32 # Define nodes to split
```

```
1 pair<-splitcomps
2 pair
3 # Run node split models
4 for(j in 1:length(pair[,1])){
5   print(pair[j,])
6
7   k.ind <- PairXY(treat=c,na=na[,1],pair=as.numeric(pair[j,]))
8
9   # Setup subdirectory to hold results for each node-split
10  dir.create(paste("REFCENode",pair[j,1],"_",pair[j,2],sep=""))
11
12  # Build data file: stored in the working directory as "data.txt"
13  bugs.data(list("r"=r,"n"=n,"t"=c, "class"=class,
14              "na" = na[,1], "nt" = nt, "nc" = nc, "ns" = ns, "ref" = ref,
15              "pair" = as.numeric(pair[j,]), "k.ind" = k.ind))
16
17  # Call OpenBUGS
18  #
19  bugs(data = "data.txt",
20       inits = list(initv1,initv2),
21       #inits = list(initv1),
22       parameters.to.save = c("direct", "d", "prob","totresdev","indirect"),
23       model.file = "fse fce node-splitR2_v3.txt",
24       n.chains = 2,
25       n.iter = 120000,
26       n.burnin = 40000,
27       n.thin = 1,
28       OpenBUGS.pgm = bd,
29       debug = FALSE,
30       save.history = TRUE,
31       useWINE=FALSE)
```

```
1 #
2 # Copy input and output files to relevant directory
3 file.copy("data.txt", paste("REFCENode",pair[j,1],"_",pair[j,2],"/data.txt",sep=""),
4 overwrite=TRUE)
5 file.copy(paste(tempdir(),"/log.odc",sep=""),
6 paste(pathname,"/REFCENode",pair[j,1],"_",pair[j,2],"/log.odc",sep=""), overwrite=TRUE)
7 file.copy(paste(tempdir(),"/log.txt",sep=""),
8 paste(pathname,"/REFCENode",pair[j,1],"_",pair[j,2],"/log.txt",sep=""), overwrite=TRUE)
9 file.copy(paste(tempdir(),"/inits1.txt",sep=""),
10 paste(pathname,"/REFCENode",pair[j,1],"_",pair[j,2],"/inits1.txt",sep=""), overwrite=TRUE)
11 file.copy(paste(tempdir(),"/script.txt",sep=""),
12 paste(pathname,"/REFCENode",pair[j,1],"_",pair[j,2],"/script.txt",sep=""), overwrite=TRUE)
13 #
14 # REPEAT FOR ALL OTHER NODES
15 }
```

1A.7.2 OpenBUGS Code

```
17 model{
18 for(i in 1:ns){          # LOOP OVER ALL STUDIES
19   delta[i,1] <- 0 # treatment effect is zero for control arm
20   mu[i] ~ dnorm(0,.0001) # vague priors for all trial baselines
21   for (k in 1:na[i]){    # LOOP OVER ARMS
22     r[i,k.ind[i,k]] ~ dbin(p[i,k],n[i,k.ind[i,k]]) # binomial likelihood
23     logit(p[i,k]) <- mu[i] + delta[i,k] # model for linear predictor
24     rhat[i,k] <- p[i,k] * n[i,k.ind[i,k]] # expected value of the numerators
25     #Deviance contribution
26     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]]-
27 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])))
28     split[i,k] <- equals(t[i,k.ind[i,1]], pair[1]) * equals(t[i,k.ind[i,k]], pair[2]) -
29 equals(t[i,k.ind[i,1]], pair[2]) * equals(t[i,k.ind[i,k]], pair[1])
30   }
31   # Summed residual deviance contribution for this trial
32   resdev[i] <- sum(dev[i,1:na[i]])
33   for (k in 2:na[i]) {    # FE model for treatment effects
```



```
1      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]
2    }
3  }
4  totresdev <- sum(resdev[])    # Total Residual Deviance
5  #
6  d[ref]<-0    # treatment effect is zero for reference treatment
7  D[class[ref]]<-0
8  # vague prior for class effects
9  for (j in 2:nc){
10   D[j] ~ dnorm(0, .0001)
11 }
12 for (j in 2:nt){
13   d[j] <- D[class[j]]
14 }
15 direct ~ dnorm(0,.0001)    # vague prior for direct comparison parameter
16 indirect <- lor[pair[1], pair[2]]
17 #calculate difference between direct and lor
18 diff <- direct - indirect
19 # calculate p-value
20 prob <- step(diff)
21 #
22 # pairwise ORs and LORs for all possible pair-wise comparisons
23 for (c in 1:(nt-1)){
24   for (k in (c+1):nt){
25     or[c,k] <- exp(d[k] - d[c])
26     lor[c,k] <- (d[k]-d[c])
27     lor[k,c] <- -lor[c,k]
28   }
29 }
30 }          # *** PROGRAM ENDS
31
```