

Acne vulgaris: management

TSU NMA Software code (moderate to severe acne)

NICE guideline tbc

Supplement 7

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 All NICE guidance is subject to regular review and may be updated or withdrawn.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 (moderate to severe acne)	5
Efficacy (% change in total lesion count from baseline)	5
A.1: Efficacy, base-case model (OpenBUGS)	5
A.2: Efficacy, node-splitting, class-level.....	10
Discontinuation for any reason	23
A.3: Discontinuation for any reason, base-case model (WinBUGS).....	23
A.4: Discontinuation for any reason, node-splitting, class-level.....	25
Discontinuation due to side effects	35
A.5: Discontinuation due to side effects, base-case model (WinBUGS).....	35
A.6: Discontinuation due to side effects, bias-adjusted model: Domain 4, Outcome measurement (efficacy) (WinBUGS).....	36
A.7: Discontinuation due to side effects, node-splitting, treatment-level.....	38

1 TSU NMA software code (moderate to 2 severe 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
```

A.2: Efficacy, node-splitting, class-level

A.2.1: R Code (requires R2OpenBUGS package)

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

```
1 # 5. You will need to edit the working directory 'pathname'
2 #   to suit your computer settings
3 #
4 # 6. Run script file
5 #
6 #
7 #####
8 #
9 # Declare the directory where OpenBUGS is found in this computer
10 bd <- "C:/Program Files (x86)/OpenBUGS/OpenBUGS323/OpenBUGS.exe"
11 #
12 # Declare working directory
13 pathname <- "C:/Acne/M2S/Efficacy"
14 setwd(pathname)
15 #
16 # load package to call OpenBUGS
17 library(R2OpenBUGS)
18 #
19 # LOAD DATA MANIPULATING FUNCTIONS:
20 #
21 PairXY <- function(treat, na, pair)
22 # Check if pair(X,Y) in row i of data
23 # and reorder treatments in trial as appropriate
24 {
25   N <- nrow(treat)
26   multi <- rep(NA,length(na))
27   split.ind <- matrix(nrow=nrow(treat),ncol=ncol(treat))
28   split.ind1 <- matrix(nrow=nrow(treat),ncol=ncol(treat))
29   split.ind2 <- matrix(nrow=nrow(treat),ncol=ncol(treat))
30   spliti <- rep(NA,length(na))
31   split1i <- rep(NA,length(na))
```

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

```
1 # k.ind ensures a treatment in the pair is in the baseline arm, where the
2 # multi-arm trial contains both treatments in the pair
3
4 # multi-arm trial contains both treatments in the pair
5 # If a multi-arm trial does not contain the node, arm order stays the same
6 k.ind[i,k]<-(k*((1-multi[i])+multi[i]*(1-split[i])+multi[i]*split[i]*(1*(split.ind[i,1]==1))))
7
8 # If a multi-arm trial contains the node, the treatment in pair[1] is not duplicated in trial,
9 # the baseline arm does not contain a treatment in the node, and the treatment
10 # in arm k is pair[1], make this treatment baseline treatment
11 + multi[i]*split[i]*(1-split1[i])*(1-(1*(split.ind[i,1]==1)))*(1*(treat[i,k]==pair[1]))
12
13 # If a multi-arm trial contains the node, the treatment in pair[1] is duplicated in trial,
14 # the treatment in pair[2] is not duplicated in trial, the baseline arm does not contain
15 # a treatment in the node, and the treatment in arm k is pair[2], make this treatment
16 baseline treatment
17 + multi[i]*split[i]*split1[i]*(1-split2[i])*(1-(1*(split.ind[i,1]==1)))*(1*(treat[i,k]==pair[2]))
18
19 # If a multi-arm trial contains the node, the treatment in pair[1] is not duplicated in trial,
20 # the baseline arm does not contain a treatment in the node, and k is baseline arm,
21 # move treatment to come after baseline treatment
22 + sum(pair1[i,1:na[i]])*(1-split1[i])*multi[i]*split[i]*(1-(1*(split.ind[i,1]==1)))*(1*(k==1))
23
24 # If a multi-arm trial contains the node, the treatment in pair[1] is duplicated in trial,
25 # the treatment in pair[2] is not duplicated in trial, the baseline arm does not contain a
26 # treatment in the node, and k is baseline arm, move treatment to come after baseline
27 treatment
28 + sum(pair2[i,1:na[i]])*split1[i]*(1-split2[i])*multi[i]*split[i]*(1-
29 (1*(split.ind[i,1]==1)))*(1*(k==1))
30
31 # If a multi-arm trial contains the node, the treatment in pair[1] are not duplicated in
32 trial,
```

```
1      # the baseline arm does not contain a treatment in the node, k is NOT baseline arm,
2      # and treatment in arm k is NOT pair[1], arm order stays the same
3      + k*multi[i]*spliti[i]*(1-split1i[i]*(1-(1*(split.ind[i,1]==1)))*(1-(1*(k==1)))*(1-
4 (1*(treat[i,k]==pair[1])))
5
6      # If a multi-arm trial contains the node, the treatment in pair[1] is duplicated in trial,
7      # the treatment in pair[2] is not duplicated in trial, the baseline arm does not contain a
8 treatment in the node, k is NOT baseline arm,
9      # and treatment in arm k is NOT pair[2], arm order stays the same
10     + k*multi[i]*spliti[i]*split1i[i]*(1-split2i[i]*(1-(1*(split.ind[i,1]==1)))*(1-(1*(k==1)))*(1-
11 (1*(treat[i,k]==pair[2])))
12   }
13 }
14 k.ind
15 }
16 #####
17 #
18 # load data for MTC
19 MTCData <- read.table("efficacy_class.txt", header=TRUE)
20 n <- data.matrix(MTCData[,c("n1", "n2", "n3", "n4")])
21 x <- data.matrix(MTCData[,c("x1", "x2", "x3", "x4")])
22 x.sd <- data.matrix(MTCData[,c("x.sd1", "x.sd2", "x.sd3", "x.sd4")])
23 y <- data.matrix(MTCData[,c("y1", "y2", "y3", "y4")])
24 y.sd <- data.matrix(MTCData[,c("y.sd1", "y.sd2", "y.sd3", "y.sd4")])
25 CFB <- data.matrix(MTCData[,c("CFB1", "CFB2", "CFB3", "CFB4")])
26 CFB.sd <- data.matrix(MTCData[,c("CFB.sd1", "CFB.sd2", "CFB.sd3", "CFB.sd4")])
27 pCFB <- data.matrix(MTCData[,c("pCFB1", "pCFB2", "pCFB3", "pCFB4")])
28 pCFB.sd <- data.matrix(MTCData[,c("pCFB.sd1", "pCFB.sd2", "pCFB.sd3", "pCFB.sd4")])
29 y.T <- data.matrix(cbind(rep(NA,length(n[,1])),MTCData[,c("y.T2","y.T3","y.T4")]))
30 se.T <- data.matrix(cbind(rep(NA,length(n[,1])),MTCData[,c("se.T2","se.T3","se.T4")]))
31 V <- data.matrix(MTCData[, "V"])
32 corr <- data.matrix(MTCData[, "corr"])
```

```
1 c <- data.matrix(MTCData[,c("c1", "c2", "c3", "c4")])
2 na <- data.matrix(MTCData[, "na"])
3 #Class when running model at class level
4 class <- 1:max(c, na.rm = TRUE)
5 nt <- max(c, na.rm=TRUE)
6 nc <- max(class)
7 ns <- nrow(n)
8 ns.a <- 48 #studies reporting arm-level data
9 ns.a1 <- 29 #pCFB studies
10 ns.a2 <- 11 #CFB studies
11 ns.t2 <- 7 #2-arm studies reporting contrasts
12 ref <- 1 #reference treatment
13 #
14 initv1 <- list(direct=0, D=c(NA,rep(0,nc-1)), mu=rep(0,ns.a), sd=1)
15 initv2 <- list(direct=0.05, D=c(NA,rep(-1.2,nc-1)), mu=rep(0.5,ns.a), sd=3)
16 #####
17 #
18 # Check which notes to split
19 #
20 library(gemtc)
21 ns.data<-mtc.data.studyrow(MTCData,
22                             armVars=c('treatment'='c'),
23                             nArmsVar='na',
24                             studyVars=c(),
25                             studyNames=MTCData$study,
26                             treatmentNames=NA,
27                             patterns=c('%s', '%s%d'))
28 net<-mtc.network(data.ab=ns.data,description="Efficacy_trt")
29 ## Print which nodes to split
30 splitcomps<-mtc.nodesplit.comparisons(net)
31 print(splitcomps)
```

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



```
1   inits = list(initv1,initv2),
2   #inits = list(initv1),
3   parameters.to.save = c("direct", "d", "prob","totresdev","indirect","sd"),
4   model.file = "rse fce node-splitR2_v2_efficacy_class.txt",
5   n.chains = 2,
6   n.iter = 120000,
7   n.burnin = 40000,
8   n.thin = 1,
9   OpenBUGS.pgm = bd,
10  debug = FALSE,
11  save.history = TRUE,
12  useWINE=FALSE)
13  #
14  # Copy input and output files to relevant directory
15  file.copy("data.txt", paste("REFCENode",pair[j,1],"_",pair[j,2],"/data.txt",sep=""),
16  overwrite=TRUE)
17  file.copy(paste(tempdir(),"/log.odc",sep=""),
18  paste(pathname,"/REFCENode",pair[j,1],"_",pair[j,2],"/log.odc",sep=""), overwrite=TRUE)
19  file.copy(paste(tempdir(),"/log.txt",sep=""),
20  paste(pathname,"/REFCENode",pair[j,1],"_",pair[j,2],"/log.txt",sep=""), overwrite=TRUE)
21  file.copy(paste(tempdir(),"/inits1.txt",sep=""),
22  paste(pathname,"/REFCENode",pair[j,1],"_",pair[j,2],"/inits1.txt",sep=""), overwrite=TRUE)
23  file.copy(paste(tempdir(),"/script.txt",sep=""),
24  paste(pathname,"/REFCENode",pair[j,1],"_",pair[j,2],"/script.txt",sep=""), overwrite=TRUE)
25  #
26  # REPEAT FOR ALL OTHER NODES
27  }
```

2A.2.2 OpenBUGS Code

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

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

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

```
1
2   theta[i,k] <- mu[i] + delta[i,k] # model for linear predictor
3
4   #Deviance contribution
5   for (j in 1:2){
6       diff[i,k,j]<- y.XY[i,k.ind[i,k],j]-mu.XY[i,k,j]
7       z[i,k,j]<- inprod(Prec[i,k.ind[i,k],j, 1:2],diff[i,k, 1:2])
8   }
9   dev[i,k]<-inprod(diff[i,k, 1:2],z[i,k, 1:2])
10
11   split[i,k] <- equals(t[i,k.ind[i,1]], pair[1]) * equals(t[i,k.ind[i,k]], pair[2]) -
12 equals(t[i,k.ind[i,1]], pair[2]) * equals(t[i,k.ind[i,k]], pair[1])
13   }
14   resdev[i] <- sum(dev[i,1:na[i]])
15 }
16 # 2-arm trials reporting contrasts (e.g., split-face trials)
17 for(i in (ns.a+1):(ns.a+ns.t2)){ # LOOP THROUGH STUDIES WITH TRIAL DATA
18   w[i,1] <- 0 # adjustment for multi-arm trials is zero for control arm
19   delta[i,1] <- 0 # treatment effect is zero for control arm
20   var[i,2] <- pow(se.T[i,2],2) # calculate variances
21   prec[i,2] <- 1/var[i,2] # set precisions
22   y.T[i,2] ~ dnorm(delta[i,2],prec[i,2]) # normal likelihood
23 #Deviance contribution
24   dev[i,2] <- (y.T[i,2]-delta[i,2])*
25   (y.T[i,2]-delta[i,2])* prec[i,2]
26   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],
27 pair[1])
28   # summed residual deviance contribution for this trial
29   resdev[i] <- dev[i,2]
30   }
31 # 4-arm trials reporting contrasts
32 # No k.ind for Thiboutot 2002 because code can't handle double nodes
```

```
1 for(i in (ns.a+ns.t2+1):ns){ # LOOP THROUGH STUDIES WITH TRIAL DATA
2   w[i,1] <- 0 # adjustment for multi-arm trials is zero for control arm
3   delta[i,1] <- 0 # treatment effect is zero for control arm
4   for (k in 2:na[i]) { # LOOP THROUGH ARMS
5     var[i,k] <- pow(se.T[i,k],2) # calculate variances
6     prec[i,k] <- 1/var[i,k] # set precisions
7   }
8   for(k in 2:na[i]){
9     split[i,k] <- equals(t[i,1], pair[1]) * equals(t[i,k], pair[2]) - equals(t[i,k], pair[2]) *
10    equals(t[i,k], pair[1])
11   }
12
13   for (k in 1:(na[i]-1)){ # set variance-covariance matrix
14     for (j in 1:(na[i]-1)){
15       Sigma2[i,j,k] <- V[i]*(1-equals(j,k)) + var[i,(k+1)]*equals(j,k)
16     }
17   }
18   Omega2[i,1:(na[i]-1),1:(na[i]-1)] <- inverse(Sigma2[i,,]) # Precision matrix
19
20   # multivariate normal likelihood for 4-arm trials
21   y.T[i,2:na[i]] ~ dmnorm(delta[i,2:na[i]],Omega2[i,1:(na[i]-1),1:(na[i]-1)])
22
23   # Deviance contribution for trial i
24   for (k in 1:(na[i]-1)){ # multiply vector & matrix
25     ydiff[i,k] <- y.T[i,(k+1)] - delta[i,(k+1)]
26     z2[i,k] <- inprod(Omega2[i,k,1:(na[i]-1)], ydiff[i,1:(na[i]-1)])
27   }
28   resdev[i] <- inprod(ydiff[i,1:(na[i]-1)], z2[i,1:(na[i]-1)])
29 }
30 #RE Model (ARM AND TRIAL DATA)
31 for(i in 1:ns){ # LOOP THROUGH STUDIES WITH ARM DATA
```

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

```
1 #
2 #calculate difference between direct and lor
3 diff.ns <- direct - indirect
4 # calculate p-value
5 prob <- step(diff.ns)
6 #
7 sd ~ dunif(0,25) # vague prior for between-trial SD
8 tau <- pow(sd,-2) # between-trial precision = (1/between-trial variance)
9 #
10 # pairwise mean differences for all possible pair-wise comparisons
11 for (c in 1:(nt-1)) {
12   for (k in (c+1):nt) {
13     mean.diff[c,k] <- d[k]-d[c]
14     mean.diff[k,c] <- -mean.diff[c,k]
15   }
16 }
17 } # *** PROGRAM ENDS
```

1B Discontinuation for any reason

1A.3: Discontinuation for any reason, base-case model (WinBUGS)

20 *Note: Same code run separately for female and male populations*

```
21 model{
22   for(i in 1:ns){ # LOOP OVER ALL STUDIES
23     mu[i] ~ dnorm(0,.0001) # vague priors for all trial baselines
24     for (k in 1:na[i]){ # LOOP OVER ARMS
25       r[i,k] ~ dbin(p[i,k],n[i,k]) # binomial likelihood
26       logit(p[i,k]) <- mu[i] + d[t[i,k]] - d[t[i,1]] # model for linear predictor
27       rhat[i,k] <- p[i,k] * n[i,k] # expected value of the numerators
28       #Deviance contribution
29       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]) -
30 log(n[i,k]-rhat[i,k])))
```

```
1     }
2   # Summed residual deviance contribution for this trial
3   resdev[i] <- sum(dev[i,1:na[i]])
4   for (k in 2:na[i]) {     # RE model for treatment effects
5     delta[i,k] ~ dnorm(md[i,k],taud[i,k]) # trial-specific LOR distributions
6     # mean of LOR distributions (with multi-arm trial correction)
7     md[i,k] <- d[t[i,k]] - d[t[i,1]] + sw[i,k]
8     # precision of LOR distributions (with multi-arm trial correction)
9     taud[i,k] <- tau *2*(k-1)/k
10    # adjustment for multi-arm RCTs
11    w[i,k] <- (delta[i,k] - d[t[i,k]] + d[t[i,1]])
12    # cumulative adjustment for multi-arm trials
13    sw[i,k] <- sum(w[i,1:k-1])/(k-1)
14  }
15 }
16 totesdev <- sum(resdev[]) # Total Residual Deviance
17 #
18 # Reference treatment
19 d[ref]<-0 # treatment effect is zero for reference treatment
20 D[class[ref]]<-0
21 #
22 # vague prior for class effects
23 for (j in 2:nc){
24   D[j] ~ dnorm(0, .0001)
25 }
26 for (j in 2:nt){
27   d[j] <- D[class[j]]
28 }
29 #
30 sd ~ dunif(0,5) # vague prior for between-trial SD
31 tau <- pow(sd,-2) # between-trial precision = (1/between-trial variance)
```



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

2A.4: Discontinuation for any reason, node-splitting, class-level

2A.4.1: R Code (requires R2OpenBUGS package)

```
28 #####
29 # Node-splitting for Acne Guideline - Discontinuation (any)
30 # R script to run node-split for the MTC Random study effects, fixed
```

```
1 # class effects model using OpenBUGS
2 #
3 # Uses R2OpenBUGS package
4 #
5 # Discontinuation (any reason)
6 # 1. Need to include in the working directory the following files:
7 #     Disc any_UK.txt --- text file with data
8 #     rse fce node-splitR2_v3.txt --- text file holding BUGS code
9 #
10 # 2. Output files will be
11 #     coda1.txt --- holds coda output
12 #     codaIndex.txt --- holds indexes to coda output
13 #     data.txt --- holds all data as used by BUGS
14 #     log.odc and log.txt --- hold WinBUGS output
15 #     inits1.txt --- holds initial values as read by BUGS
16 #     script.txt --- BUGS script file with all commands to execute
17 #
18 # 3. Output files for each node should be transferred to a new directory
19 #     as they will be overwritten in each new run
20 #
21 # 4. You may need to edit the OpenBUGS location 'bd'
22 #
23 # 5. You will need to edit the working directory 'pathname'
24 #     to suit your computer settings
25 #
26 # 6. Run script file
27 #
28 # 7. To repeat for other node-splits need to change variable 'pair'
29 #     and edit output file names
30 #
31 #####
```

```
1 #
2 # Declare the directory where OpenBUGS is found in this computer
3 bd <- "C:/Program Files (x86)/OpenBUGS/OpenBUGS323/OpenBUGS.exe"
4 #
5 # Declare working directory
6 pathname <- "C:/Acne/M2S/Disc Any/"
7 setwd(pathname)
8 #
9 # load package to call OpenBUGS
10 library(R2OpenBUGS)
11 #
12 # LOAD DATA MANIPULATING FUNCTIONS:
13 #
14 PairXY <- function(treat, na, pair)
15 # Check if pair(X,Y) in row i of data
16 # and reorder treatments in trial as appropriate
17 {
18 N <- nrow(treat)
19 multi <- rep(NA,length(na))
20 split.ind <- matrix(nrow=nrow(treat),ncol=ncol(treat))
21 split.ind1 <- matrix(nrow=nrow(treat),ncol=ncol(treat))
22 split.ind2 <- matrix(nrow=nrow(treat),ncol=ncol(treat))
23 spliti <- rep(NA,length(na))
24 split1i <- rep(NA,length(na))
25 split2i <- rep(NA,length(na))
26 pair1 <- matrix(nrow=nrow(treat),ncol=ncol(treat))
27 pair2 <- matrix(nrow=nrow(treat),ncol=ncol(treat))
28 k.ind <- matrix(nrow=nrow(treat),ncol=ncol(treat))
29 for (i in 1:N) {
30 # is trial i a multiarm trial?
31 multi[i] <- 1*(na[i]>2)
```

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

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

37

```
1           # If a multi-arm trial contains the node, the treatment in pair[1] is duplicated in
2 trial,
3           # the treatment in pair[2] is not duplicated in trial, the baseline arm does not
4 contain a treatment in the node, k is NOT baseline arm,
5           # and treatment in arm k is NOT pair[2], arm order stays the same
6           + k*multi[i]*spliti[i]*split1i[i]*(1-split2i[i]*(1-(1*(split.ind[i,1]==1)))*(1-(1*(k==1)))*(1-
7 (1*(treat[i,k]==pair[2]))))
8     }
9   }
10  k.ind
11 }
12 #####
13 #
14 # load data for MTC
15 MTCData <- read.table("Disc any_UK.txt", header=TRUE)
16 r <- data.matrix(MTCData[,c("r1", "r2", "r3", "r4")])
17 n <- data.matrix(MTCData[,c("n1", "n2", "n3", "n4")])
18 c <- data.matrix(MTCData[,c("c1", "c2", "c3", "c4")])
19 na <- data.matrix(MTCData[, "na"])
20 #Class when running model at class level
21 class <- 1:max(c,na.rm = TRUE)
22 nt <- max(c, na.rm=TRUE)
23 nc <- max(class)
24 ns <- nrow(r)
25 ref <- 1 #reference treatment
26 #
27 # define initial values
28 initv1 <- list(direct=0, D=c(NA,rep(0,nc-1)), mu=rep(0,ns), sd=1)
29 initv2 <- list(direct=0.05, D=c(NA,rep(-1.2,nc-1)), mu=rep(0.5,ns), sd=3)
30 #####
31 #
32 # Check which notes to split
33 #
```

```
1 library(gemtc)
2 ns.data<-mtc.data.studyrow(MTCDData,
3     armVars=c('treatment'='c', 'responders'='r', 'sampleSize'='n'),
4     nArmsVar='na',
5     studyVars=c(),
6     studyNames=MTCDData$studyid,
7     treatmentNames=NA,
8     patterns=c('%s', '%s%d'))
9 net<-mtc.network(data.ab=ns.data,description="Disc any_trt")
10 ## Print which nodes to split
11 splitcomps<-mtc.nodesplit.comparisons(net)
12 print(splitcomps)
13 #
14 #####
15 ##
16 # NODE-SPLITTING ROUTINE
17 #####
18 ##
19 #
20 #
21 # Define nodes to split
22 pair<-splitcomps
23 pair
24 # Run node split models
25 for(j in 1:length(pair[,1])){
26   print(pair[j,])
27
28   k.ind <- PairXY(treat=c,na=na[,1],pair=as.numeric(pair[j,]))
29
30   # Setup subdirectory to hold results for each node-split
31   dir.create(paste("REFCENode",pair[j,1],"_",pair[j,2],sep=""))
32   # Build data file: stored in the working directory as "data.txt"
Acne vulgaris Supplement 7: TSU NMA software code (December 2020)
```

```
1  bugs.data(list("r"=r,"n"=n,"t"=c, "class"=class,
2          "na" = na[,1], "nt" = nt, "nc" = nc, "ns" = ns, "ref" = ref,
3          "pair" = as.numeric(pair[j,]), "k.ind" = k.ind))
4
5  # Call OpenBUGS
6  #
7  bugs(data = "data.txt",
8    inits = list(initv1,initv2),
9    #inits = list(initv1),
10 parameters.to.save = c("direct", "d", "prob","totresdev","indirect","sd"),
11 model.file = "rse fce node-splitR2_v3.txt",
12 n.chains = 2,
13 n.iter = 120000,      #including burn-in iterations
14 n.burnin = 40000,
15 n.thin = 1,
16 OpenBUGS.pgm = bd,
17 debug = FALSE,
18 save.history = TRUE,
19 useWINE=FALSE)
20 #
21 # Copy input and output files to relevant directory
22 file.copy("data.txt", paste("REFCENode",pair[j,1],"_",pair[j,2],"/data.txt",sep=""),
23 overwrite=TRUE)
24 file.copy(paste(tempdir(),"/log.odc",sep=""),
25 paste(pathname,"/REFCENode",pair[j,1],"_",pair[j,2],"/log.odc",sep=""), overwrite=TRUE)
26 file.copy(paste(tempdir(),"/log.txt",sep=""),
27 paste(pathname,"/REFCENode",pair[j,1],"_",pair[j,2],"/log.txt",sep=""), overwrite=TRUE)
28 file.copy(paste(tempdir(),"/inits1.txt",sep=""),
29 paste(pathname,"/REFCENode",pair[j,1],"_",pair[j,2],"/inits1.txt",sep=""), overwrite=TRUE)
30 file.copy(paste(tempdir(),"/script.txt",sep=""),
31 paste(pathname,"/REFCENode",pair[j,1],"_",pair[j,2],"/script.txt",sep=""), overwrite=TRUE)
32 #
33 # REPEAT FOR ALL OTHER NODES
```


1 }

A.4.2 OpenBUGS Code

```
3 model{
4   for(i in 1:ns){           # LOOP OVER ALL STUDIES
5     delta[i,1] <- 0 # treatment effect is zero for control arm
6     mu[i] ~ dnorm(0,.0001) # vague priors for all trial baselines
7     for (k in 1:na[i]){     # LOOP OVER ARMS
8       r[i,k.ind[i,k]] ~ dbin(p[i,k],n[i,k.ind[i,k]]) # binomial likelihood
9       logit(p[i,k]) <- mu[i] + delta[i,k] # model for linear predictor
10      rhat[i,k] <- p[i,k] * n[i,k.ind[i,k]] # expected value of the numerators
11      #Deviance contribution
12      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]]-
13      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])))
14      split[i,k] <- equals(t[i,k.ind[i,1]], pair[1]) * equals(t[i,k.ind[i,k]], pair[2]) -
15      equals(t[i,k.ind[i,1]], pair[2]) * equals(t[i,k.ind[i,k]], pair[1])
16    }
17    # Summed residual deviance contribution for this trial
18    resdev[i] <- sum(dev[i,1:na[i]])
19    for (k in 2:na[i]) {     # RE model for treatment effects
20      delta[i,k] ~ dnorm(md[i,k],taud[i,k]) # trial-specific LOR distributions
21      # mean of LOR distributions (with multi-arm trial correction)
22      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]
23      # precision of LOR distributions (with multi-arm trial correction)
24      taud[i,k] <- tau *2*(k-1)/k
25      # adjustment for multi-arm RCTs
26      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 *
27      split[i,k] )
28      # cumulative adjustment for multi-arm trials
29      sw[i,k] <- sum(w[i,1:k-1])/(k-1)
30    }
31  }
```

```
32 totresdev <- sum(resdev[]) # Total Residual Deviance
```

```
1 #
2 d[ref]<-0    # treatment effect is zero for reference treatment
3 D[class[ref]]<-0
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 direct ~ dnorm(0,.0001)    # vague prior for direct comparison parameter
12 indirect <- lor[pair[1], pair[2]]
13 #calculate difference between direct and lor
14 diff <- direct - indirect
15 # calculate p-value
16 prob <- step(diff)
17 #
18 sd ~ dunif(0,5)    # vague prior for between-trial SD
19 tau <- pow(sd,-2) # between-trial precision = (1/between-trial variance)
20 #
21 # pairwise ORs and LORs for all possible pair-wise comparisons
22 for (c in 1:(nt-1)){
23   for (k in (c+1):nt){
24     or[c,k] <- exp(d[k] - d[c])
25     lor[c,k] <- (d[k]-d[c])
26     lor[k,c] <- -lor[c,k]
27   }
28 }
29 }          # *** PROGRAM ENDS
30
```

Discontinuation due to side effects

A.5: Discontinuation due to side effects, base-case model (WinBUGS)

3 *Note: Same code run separately for female and male populations*

```
4 model{
5   for(i in 1:ns){           # LOOP OVER ALL STUDIES
6     mu[i] ~ dnorm(0,.0001)  # vague priors for all trial baselines
7     for (k in 1:na[i]){    # LOOP OVER ARMS
8       r[i,k] ~ dbin(p[i,k],n[i,k]) # binomial likelihood
9       logit(p[i,k]) <- mu[i] + d[t[i,k]] - d[t[i,1]] # model for linear predictor
10      rhat[i,k] <- p[i,k] * n[i,k] # expected value of the numerators
11      #Deviance contribution
12      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]) -
13      log(n[i,k]-rhat[i,k])))
14    }
15    # Summed residual deviance contribution for this trial
16    resdev[i] <- sum(dev[i,1:na[i]])
17  }
18  totesdev <- sum(resdev[]) # Total Residual Deviance
19  #
20  # Reference treatment
21  d[ref]<-0 # treatment effect is zero for reference treatment
22  D[class[ref]]<-0
23  #
24  # vague prior for class effects
25  for (j in 2:nc){
26    D[j] ~ dnorm(0, .0001)
27  }
28  for (j in 2:nt){
29    d[j] <- D[class[j]]
30  }
31  #
```

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

**2A.6: Discontinuation due to side effects, bias-adjusted model: Domain 4,
26 Outcome measurement (efficacy) (WinBUGS)**

```
27 model{
28 for(i in 1:ns){ # LOOP OVER ALL STUDIES
29   mu[i] ~ dnorm(0,.0001) # vague priors for all trial baselines
30   beta[i,1] <- 0 #No bias on baseline arm
31   for (k in 1:na[i]){ # LOOP OVER ARMS
```

```
1      r[i,k] ~ dbin(p[i,k],n[i,k]) # binomial likelihood
2      logit(p[i,k]) <- mu[i] + d[t[i,k]] - d[t[i,1]] + beta[i,k]*X[i,k]*step(bias[i,b.ind]-2) # model for
3 linear predictor, Bias for high ROB or some concerns
4      rhat[i,k] <- p[i,k] * n[i,k] # expected value of the numerators
5      #Deviance contribution
6      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]) -
7 log(n[i,k]-rhat[i,k])))
8      }
9      for(k in 2:na[i]){
10         # model for bias parameter beta
11         beta[i,k] ~ dnorm(b, prec.b)
12      }
13      # Summed residual deviance contribution for this trial
14      resdev[i] <- sum(dev[i,1:na[i]])
15      }
16      totresdev <- sum(resdev[]) # Total Residual Deviance
17      #
18      # Reference treatment currently Placebo
19      d[ref]<-0 # treatment effect is zero for reference treatment
20      D[class[ref]]<-0
21      # vague prior for class effects
22      for (j in 2:nc){
23         D[j] ~ dnorm(0, .0001)
24      }
25      for (j in 2:nt){
26         d[j] <- D[class[j]]
27      }
28      # bias model prior for variance
29      sd.b ~ dunif(0,5)
30      prec.b <- pow(sd.b,-2)
31      # bias model prior for mean
32      b ~ dnorm(0,.0001)
```

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

2A.7: Discontinuation due to side effects, node-splitting, treatment-level

2A.7.1: R Code (requires R2OpenBUGS package)

```
28 #####
29 # Node-splitting for Acne Guideline - Discontinuation (due to SE)
30 # R script to run node-split for the MTC Fixed study effects, fixed
```

```
1 # class effects model using OpenBUGS
2 #
3 # Uses R2OpenBUGS package
4 #
5 # Discontinuation (due to SE)
6 # 1. Need to include in the working directory the following files:
7 #     Disc se.txt --- text file with data
8 #     fse fce node-splitR2_v3.txt --- text file holding BUGS code
9 #
10 # 2. Output files will be
11 #     data.txt --- holds all data as used by BUGS
12 #     log.odc and log.txt --- hold WinBUGS output
13 #     inits1.txt --- holds initial values as read by BUGS
14 #     script.txt --- BUGS script file with all commands to execute
15 #
16 # 3. Output files for each node should be transferred to a new directory
17 #     as they will be overwritten in each new run
18 #
19 # 4. You may need to edit the OpenBUGS location 'bd'
20 #
21 # 5. You will need to edit the working directory 'pathname'
22 #     to suit your computer settings
23 #
24 # 6. Run script file
25 #
26 #####
27 #
28 # Declare the directory where OpenBUGS is found in this computer
29 bd <- "C:/Program Files (x86)/OpenBUGS/OpenBUGS323/OpenBUGS.exe"
30 #
31 # Declare working directory
```

```
1 pathname <- "C:/Acne/M2S/Disc SE/"
2 setwd(pathname)
3 #
4 # load package to call OpenBUGS
5 library(R2OpenBUGS)
6 #
7 # LOAD DATA MANIPULATING FUNCTIONS:
8 #
9 PairXY <- function(treat, na, pair)
10 # Check if pair(X,Y) in row i of data
11 # and reorder treatments in trial as appropriate
12 {
13 N <- nrow(treat)
14 multi <- rep(NA,length(na))
15 split.ind <- matrix(nrow=nrow(treat),ncol=ncol(treat))
16 split.ind1 <- matrix(nrow=nrow(treat),ncol=ncol(treat))
17 split.ind2 <- matrix(nrow=nrow(treat),ncol=ncol(treat))
18 spliti <- rep(NA,length(na))
19 split1i <- rep(NA,length(na))
20 split2i <- rep(NA,length(na))
21 pair1 <- matrix(nrow=nrow(treat),ncol=ncol(treat))
22 pair2 <- matrix(nrow=nrow(treat),ncol=ncol(treat))
23 k.ind <- matrix(nrow=nrow(treat),ncol=ncol(treat))
24 for (i in 1:N) {
25 # is trial i a multiarm trial?
26 multi[i] <- 1*(na[i]>2)
27 for (k in 1:na[i]){
28 # which arms contain a treatment in the pair?
29 split.ind[i,k] <- 1*(treat[i,k]==pair[1])+1*(treat[i,k]==pair[2])
30 # which arms contain the treatment in pair[1]?
31 split.ind1[i,k] <- 1*(treat[i,k]==pair[1])
```



```
1   # which arms contain the treatment in pair[2]?
2   split.ind2[i,k] <- 1*(treat[i,k]==pair[2])
3   }
4   # does trial i contain multiples of pair[1]?
5   split1i[i] <- 1*(sum(split.ind1[i,1:na[i]])>1)
6   # does trial i contain multiples of pair[2]?
7   split2i[i] <- 1*(sum(split.ind2[i,1:na[i]])>1)
8   # does trial i contain both treatments in the pair?
9   # (minus duplicates in multiarm trials that have one treatment (only) in pair)
10  spliti[i] <- 1*((sum(split.ind[i,1:na[i]])-split1i[i]*(sum(split.ind1[i,1:na[i]])-split1i[i])-
11 split2i[i]*(sum(split.ind2[i,1:na[i]])-split2i[i]))>1)
12  for (k in 1:na[i]) {
13    # which arms contain the first element in the pair
14    pair1[i,k] <- k*(1*(treat[i,k]==pair[1]))
15    # which arms contain the second element in the pair
16    pair2[i,k] <- k*(1*(treat[i,k]==pair[2]))
17  }
18  for (k in 1:na[i]) {
19    # reposition order of arms within a trial according to node being split
20    # k.ind ensures a treatment in the pair is in the baseline arm, where the
21    # multi-arm trial contains both treatments in the pair
22    # If a multi-arm trial does not contain the node, arm order stays the same
23    k.ind[i,k]<-(k*((1-multi[i])+multi[i]*(1-spliti[i])+multi[i]*spliti[i]*(1*(split.ind[i,1]==1))))
24
25    # If a multi-arm trial contains the node, the treatment in pair[1] is not duplicated in
26    trial,
27
28    # the baseline arm does not contain a treatment in the node, and the treatment
29    # in arm k is pair[1], make this treatment baseline treatment
30
31    + multi[i]*spliti[i]*(1-split1i[i])*(1-(1*(split.ind[i,1]==1)))*(1*(treat[i,k]==pair[1]))
32
31    # If a multi-arm trial contains the node, the treatment in pair[1] is duplicated in
32    trial,
```

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

```
1           + k*multi[i]*spliti[i]*split1i[i]*(1-split2i[i])*(1-(1*(split.ind[i,1]==1)))*(1-(1*(k==1)))*(1-
2 (1*(treat[i,k]==pair[2])))
3   }
4   }
5   k.ind
6 }
7 #####
8 #
9 # load data for MTC
10 MTCData <- read.table("Disc se_UK_treat.txt", header=TRUE)
11 r <- data.matrix(MTCData[,c("r1", "r2", "r3", "r4")])
12 n <- data.matrix(MTCData[,c("n1", "n2", "n3", "n4")])
13 t <- data.matrix(MTCData[,c("t1", "t2", "t3", "t4")])
14 na <- data.matrix(MTCData[, "na"])
15 #Class when running model at treatment level
16 class <-c(1,1, 1,2,2,3,3, 3,3,4,5,6, 7,7,7,7,7, 8,9,10,11,11,
17 12,13,14,15,15,16,16,16,17,17, 18)
18 nt <- max(t, na.rm=TRUE)
19 nc <- max(class)
20 ns <- nrow(r)
21 ref <- 1 #reference treatment
22 #
23 # define initial values
24 initv1 <- list(direct=0, D=c(NA,rep(0,nc-1)), mu=rep(0,ns))
25 initv2 <- list(direct=0.05, D=c(NA,rep(-1.2,nc-1)), mu=rep(0.5,ns))
26 #####
27 #
28 # Check which notes to split
29 #
30 library(gemtc)
31 ns.data<-mtc.data.studyrow(MTCData,
32 armVars=c('treatment'='t', 'responders'='r', 'sampleSize'='n'),
```

```
1         nArmsVar='na',
2         studyVars=c(),
3         studyNames=MTCDData$study,
4         treatmentNames=NA,
5         patterns=c('%s', '%s%d'))
6 net<-mtc.network(data.ab=ns.data,description="Disc se_trt")
7 ## Print which nodes to split
8 splitcomps<-mtc.nodesplit.comparisons(net)
9 print(splitcomps)
10 #
11 #####
12 ##
13 # NODE-SPLITTING ROUTINE
14 #####
15 ##
16 #
17 # Define nodes to split
18 pair<-splitcomps
19 pair
20 # Run node split models
21 for(j in 1:length(pair[,1])){
22   print(pair[j,])
23
24   k.ind <- PairXY(treat=t,na=na[,1],pair=as.numeric(pair[j,]))
25
26   # Setup subdirectory to hold results for each node-split
27   dir.create(paste("FEFCNode",pair[j,1],"_",pair[j,2],sep=""))
28
29   # Build data file: stored in the working directory as "data.txt"
30   bugs.data(list("r"=r,"n"=n,"t"=t, "class"=class,
31                 "na" = na[,1], "nt" = nt, "nc" = nc, "ns" = ns, "ref" = ref,
32                 "pair" = as.numeric(pair[j,]), "k.ind" = k.ind))
```

```
1
2 # Call OpenBUGS
3 #
4 bugs(data = "data.txt",
5     inits = list(initv1,initv2),
6     #inits = list(initv1),
7     parameters.to.save = c("direct", "d", "prob","totresdev","indirect"),
8     model.file = "fse fce node-splitR2_v3.txt",
9     n.chains = 2,
10    n.iter = 120000,      #including burn-in iterations
11    n.burnin = 40000,
12    n.thin = 1,
13    OpenBUGS.pgm = bd,
14    debug = TRUE,
15    save.history = TRUE,
16    useWINE=FALSE)
17 #
18 # Copy input and output files to relevant directory
19 file.copy("data.txt", paste("FEFCENode",pair[j,1],"_",pair[j,2],"/data.txt",sep=""),
20 overwrite=TRUE)
21 file.copy(paste(tempdir(),"/log.odc",sep=""),
22 paste(pathname,"/FEFCENode",pair[j,1],"_",pair[j,2],"/log.odc",sep=""), overwrite=TRUE)
23 file.copy(paste(tempdir(),"/log.txt",sep=""),
24 paste(pathname,"/FEFCENode",pair[j,1],"_",pair[j,2],"/log.txt",sep=""), overwrite=TRUE)
25 file.copy(paste(tempdir(),"/inits1.txt",sep=""),
26 paste(pathname,"/FEFCENode",pair[j,1],"_",pair[j,2],"/inits1.txt",sep=""), overwrite=TRUE)
27 file.copy(paste(tempdir(),"/script.txt",sep=""),
28 paste(pathname,"/FEFCENode",pair[j,1],"_",pair[j,2],"/script.txt",sep=""), overwrite=TRUE)
29 #
30 # REPEAT FOR ALL OTHER NODES
31 }
```

3A.7.2 OpenBUGS Code

```
33 model{
    Acne vulgaris Supplement 7: TSU NMA software code (December 2020)
```

```
1 for(i in 1:ns){          # LOOP OVER ALL STUDIES
2   delta[i,1] <- 0 # treatment effect is zero for control arm
3   mu[i] ~ dnorm(0,.0001)    # vague priors for all trial baselines
4   for (k in 1:na[i]){      # LOOP OVER ARMS
5     r[i,k.ind[i,k]] ~ dbin(p[i,k],n[i,k.ind[i,k]]) # binomial likelihood
6     logit(p[i,k]) <- mu[i] + delta[i,k] # model for linear predictor
7     rhat[i,k] <- p[i,k] * n[i,k.ind[i,k]] # expected value of the numerators
8     #Deviance contribution
9     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]]-
10    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])))
11     split[i,k] <- equals(t[i,k.ind[i,1]], pair[1]) * equals(t[i,k.ind[i,k]], pair[2]) -
12    equals(t[i,k.ind[i,1]], pair[2]) * equals(t[i,k.ind[i,k]], pair[1])
13   }
14   # Summed residual deviance contribution for this trial
15   resdev[i] <- sum(dev[i,1:na[i]])
16   for (k in 2:na[i]) {    # FE model for treatment effects
17     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]
18   }
19 }
20 totresdev <- sum(resdev[]) # Total Residual Deviance
21 #
22 d[ref]<-0 # treatment effect is zero for reference treatment
23 D[class[ref]]<-0
24 # vague prior for class effects
25 for (j in 2:nc){
26   D[j] ~ dnorm(0, .0001)
27 }
28 for (j in 2:nt){
29   d[j] <- D[class[j]]
30 }
31 direct ~ dnorm(0,.0001) # vague prior for direct comparison parameter
32 indirect <- lor[pair[1], pair[2]]
```

```
1 #calculate difference between direct and lor
2 diff <- direct - indirect
3 # calculate p-value
4 prob <- step(diff)
5 #
6 # pairwise ORs and LORs for all possible pair-wise comparisons
7 for (c in 1:(nt-1)){
8   for (k in (c+1):nt){
9     or[c,k] <- exp(d[k] - d[c])
10    lor[c,k] <- (d[k]-d[c])
11    lor[k,c] <- -lor[c,k]
12  }
13 }
14 }          # *** PROGRAM ENDS
15
```