Appendix G: Technical appendix

TA Trikalinos, V Rofeberg

June 18, 2018

Contents

1 Network meta-analysis models

We provide a detailed description of the models used in the main analysis. Sensitivity analyses fitted the analogous model in the

1.1 Notation

Let $k = 1, \ldots, K$ index studies and $(j = 1, \ldots, J)$ index treatments in a network meta-analysis. Write x_{kj} for the number of events and N_{kj} for the total number of people who received treatment j in trial k.[1] The proportion of events with treatment j in trial k is estimated by

$$p_{jk} = \frac{x_{kj}}{N_{kj}}. \tag{1}$$

Write

$$\theta_{kj} = \text{logit}(p_{kj}) \tag{2}$$

[1] If trial k compares a strict subset of the J treatments, say j_1, j_2, and j_3, $j \in \{j_1, j_2, j_3\} = \mathcal{J}_k \subset \{1, 2, ..., J\} = \mathcal{J}$.

for the logit-transformed estimate, where $\text{logit}(x) \coloneqq \log(x) - \log(1-x)$; then the conditional (sampling) variance of θ_{kj} is

$$\sigma^2_{kj} = \frac{1}{N_{kj}\ p_{kj}\ (1-p_{kj})}. \tag{3}$$

Encode what treatment was assigned in an arm in a trial using the $(J-1)$-long row vector

$$\mathbf{T}_{kj} = \begin{cases} (0, \ldots, 1_{[j]}, \ldots, 0) & \text{if } j < J \\ (0, \ldots, 0) & \text{if } j = J \end{cases}. \tag{4}$$

where has $1_{[j]}$ means '1 at the j-th position'. This can be used as a row in a design matrix encoded such that treatment J is the reference treatment.

1.2 Model

Network meta-analysis is mathematically equivalent to a 2-level hierarchical model.

1.3 Observational part

The first level (observational part) models the conditional distribution of data within each trial as

$$\theta_{kj} \sim N(\mu_{kj}, \sigma^2_{kj}) \tag{5}$$

$$\mu_{kj} = \boldsymbol{T}_{kj}\boldsymbol{\delta}_k + \alpha_k, \tag{6}$$

where $\boldsymbol{\delta}_k = (\delta_{k1}, \ldots, \delta_{k,J-1})'$ is a column vector of basic parameters, α_k is a study-specific intercept, and $'$ denotes transpose. The α_k can be interpreted as the log-odds of the probability of the outcome under treatment J. Each δ_{kj} can be interpreted as the difference in the log-odds of the probability of the outcome between treatment $j < J$ and treatment J in study k.

The model in (6) explicitly encodes a *consistency* assumption between direct and indirect effects. [2]

1.4 Structural part

The structural part of the model prescribes how the study-specific parameters are related.

[2] Model (6) reduces the number of treatment effects in the network's graph from $|\mathcal{E}| \geq J-1$ to the $J-1$ in $\boldsymbol{\delta}_k$. All other effects are recovered as convex combinations of the elements of $\boldsymbol{\delta}_k$.

The intercepts α_k

The α_k can be interpreted as the log-odds of the probability of the outcome under treatment J. We examined two variants in modeling the intercepts

1. As fixed constants.

2. As random effects
$$\alpha_k \sim N(\alpha, \tau_\alpha^2), \tag{7}$$
where α is the between-study mean for treatment J and τ_α^2 the between-study variance.

The treatment effects $\boldsymbol{\delta}_k$

We considered two variants.

1. Under an *equal effect* model
$$\boldsymbol{\delta}_k = \boldsymbol{\delta}, \tag{8}$$
for all k. In this case the hierarchical model degenerates to a heteroskedastic regression model.

2. Under a *random effects* model, treatment-specific effects are modeled with a multivariate normal distribution
$$\boldsymbol{\delta}_k \sim N(\boldsymbol{\delta}, \boldsymbol{\Sigma}), \tag{9}$$
where $\boldsymbol{\delta} = (\delta_1, \ldots, \delta_{J-1})'$ is a column vector of between-study means and covariance matrix
$$\boldsymbol{\Sigma} = \tau^2 \boldsymbol{\Omega}. \tag{10}$$
In (10), $\boldsymbol{\Omega}$ is a square correlation matrix. τ^2 is a between-study variance parameter which we assume to be the same for all treatment effects. This *homogeneity of variances* assumption is almost forced, because there are at most 7 trials per edge in the network graphs, and most edges have 3 or fewer trials. The homogeneity of variances assumption together with (6) imply that $\boldsymbol{\Omega}$ has a compound-symmetry structure with all off-diagonal elements equal to 0.5 and all diagonal elements equal to 1. [3]

1.5 Estimation

We fit all model variants by maximizing the associated likelihoods.

We consider the sampling variances in (3) to be known and equal to their sample estimates. This is a typical, though rarely explained, assumption in meta-analysis.

Thus, we need to estimate J between-study means, namely, α and $\boldsymbol{\delta}$, and 2 between-study variance parameters, namely, τ_α^2 and τ^2.

[3] To show, start from (6) and take variances.

1.6 Comparison of direct and indirect estimates of an effect

Consider a network with $J + 1$ treatments, where treatment j in the set of studies that include a comparison of interest (say, j vs. i) is considered as j^*. Implementing the models previously described, the direct estimate for the comparison of interest is then the contrast between treatment j^* and treatment i, and the indirect estimate is the contrast between treatment j and treatment i. The consistency factor is then the contrast between treatment j^* and j.

1.7 Computation

We fit models with *metafor*, a wrapper for *lme4* in R. Graph operations were done with *igraph* in R.

2 Code

We outline the strategy for the coding in Section 2.1, by describing the logic for the analysis of the largest network for recurrence in Figure 1. We provide the full code in 2.2 and 2.3.

2.1 Illustrative code

```
# This example code illustrates the coding strategy in the complete
# code appendix for the main analysis of recurrence in the largest
# network in Figure 1. The complete code does all these steps
# automatically for all networks (i.e., connected subgraphs) and
# for all outcomes, handling exceptions as needed.

# Read necessary packages.
library(metafor)

# Read the data in to a dataframe
data <- read.csv("_d_RecurBCC_cleandata_subgraph1.csv")
data <- as.data.frame(data)

# Create a copy of the "treatment" variable, and pick a reference
# treatment. This is unimportant in terms of fitting the main
# analysis. However, code such as this will be used to change
# the reference treatment to easily get specific results.
data[, "trt"] <-  data[, "treatment"]
reference_treatment_name <-
    sort(levels(data[, "trt"])[as.numeric(data.ab[, "trt"])])[1]
data <-
    within(data, trt<-relevel(trt, ref=reference_treatment_name))

# Estimate the empirical log-odds as logit(responders/sampleSize).
# The escalc() function of metafor does this handling edge cases
data[, c("yi", "vi")] <-
    escalc(measure="PLO", xi=data[, "responders"],
           ni=data[, "sampleSize"])

# Run the analysis in metafor using restricted maximum likelihood.
# Because this analysis has at least 3 treatments and at least
# 4 RCTs in a comparison, we fit a random effects model.
# Otherwise, we would fit a fixed effects model.
# In the complete code, we count the treatments and the trials per
# comparison by manipulating a mathematical graph object, and
# fork the analysis accordingly (not shown in this example).
# This command fits the normal-normal models described in the
# theory section of the technical appendix.
```

```
results <- rma.mv(yi=yi, vi=vi, mods= as.formula(" ~ 1 + trt"),
                      data=data,
                      random= as.formula(" ~ 1 | study / trt"),
                      method="REML")


# League tables:

# Because this analysis is modeling the empirical log-odds of the
# event in each arm, we can estimate not only the mean effect
# across studies (i.e.,  the mean difference in the log-odds)
# but also the mean frequency of the event with each treatment.
# We can get these results for the above analysis as "predictions"
# of the mean proportion in each treatment.
# Note that in hierarchical models many different predictions can
# be obtained. This code with metafor gets us the predictions we
# need.

  ## To wit, first, set up an elementary design matrix to obtain
  ## predictions; This is an identity matrix I of the appropriate rank,
  ## stacked on a row-vector of 0's
  ##
  ##                           | 1, 0, ..., 0, 0 |
  ##              | I |        | 0, 1, ..., 0, 0 |
  ##  contrasts = +---+ =      | ...             |
  ##              | 0 |        | 0, 0, ..., 0, 1 |
  ##                           +-----------------+
  ##                           | 0, 0, ..., 0, 0 |
  ##
  ##
  ## The last row of this matrix will get us the mean log-odds for the
  ## event in the reference treatment, which corresponds to the intercept
  ## of the model in rows 39-42.The t-th row of the identity matrix obtains
  ## the mean log-odds for the t-th non-reference treatment.
  ## This design matrix works because we specified "intercept=TRUE" in line 77.
  n_treatments <- length(results$b)
  contrasts <- diag(n_treatments-1)
  contrasts <- rbind( rep(0, n_treatments-1), contrasts )
  logit_means <- predict(results, newmods=contrasts, intercept=TRUE )

  ## Then, we have to back-calculate the mean proportions from the mean odds
  ## The next line is, admittedly, an inefficient manipulation of objects in R.
  ## Coerce the list.rms object to a dataframe by taking the output of print()!
  logit_means <- as.data.frame(print.list.rma(logit_means))
  ## Proportion mean and confidence interval with pointwise transformation.
  pred <- exp(logit_means)/(1+exp(logit_means))
```

```
85
86  # Relative effects tables:
87
88    ## The coefficients in the "results" object in lines 39-42
89    ## correspond to a column of the relative effects table, namely the
90    ## column that corresponds to the reference treatment chosen in lines 21-22.
91
92    ## There are several ways to obtain the full relative effects table.
93    ## an efficient one would be to create an appropriate elementary
94    ## design matrix with each row corresponding to each relative
95    ## effect and run a command analogous to line 77.
96
97    ## However, we obtained the relative effects table the following way:
98    ## 1. by switching the reference treatment in the "trt" factor
99    ##    using code analogous to that in lines 21-22
100   ## 2. by re-running the model in 39-42 for the new reference
101   ##    treatment, to obtain another column of the relative effects
102   ##    table
103   ##
104   ## 1 and 2 are accomplished in a dedicated function in the R code.
105   ##
106   ## While this roundabout is not the optimal approach
107   ## (is not numerically stable in edge cases) we believe that it
108   ## did not introduce problems in the current analysis.
109
110
111 # "Split-node" analyses:
112
113   ## The name is unfortunate, because it refers to "nodes" in the directed
114   ## acyclic graphs describing an MCMC sampling algorithm in Bayesian models.
115   ## It would correspond to parameters for the edges in network figures such
116   ## as Figures 1-4 in the paper in a network meta-analysis model.
117   ##
118   ## Here is what this analysis does: Consider the small network
119   ##    A --- B
120   ##     \  /  \  .
121   ##      C ---D
122   ##
123   ##
124   ## Consistency assumes that the indirect effect for the comparison of
125   ## say, treatment A vs treatment C (i.e., the effect we estimate
126   ## without considering any A vs C trials) is the same as the direct
127   ## effect from the A vs C trials.
128   ## In the augmented network below, in the A vs C trials we
129   ## renamed C as C* :
130   ##
```

```
##    A --- B
##   /    /  \  .
##  C*  C ---D
##
## We can prove (but do not show the proof) that running a network
## meta-analysis in this augmented network gets us
##   1. an estimate of the direct effect of treatment A vs treatment C
##      in the form of the effect (A vs C*)
##   2. an estimate of the indirect effect of treatment A vs treatment C
##      based on all other data in the network in the form of the
##      effect (A vs C).
##
## We can then get the difference of direct and indirect effects
## from the relative effects table for the augmented network.
##
## The catch is that one has to do several such analyses, one for every
## informationally-independent cycle in the network (many authors
## of network meta-analyses refer to cycles in a graph as "loops").
## The algorithm to do this involves some maths on graphs. While the
## intuition behind the maths is straightforward, we do not describe it here.
```

2.2 Function definitions

```
################## Frequentist NMA Function #####################################
analyze_network_frequentist <- function(data.ab, xi, ni,
                                        study_column_name="study",
                                        treatment_column_name="treatment",
                                        reference_treatment_name=NA,
                                        rates_in_reference_treatment=c(0.01, 0.05, 0.10),
                                        digits=2) {

  data.ab <- as.data.frame(data.ab)

  ## a copy of the trt factor, to re-level
  data.ab[, "trt"] <-  data.ab[, treatment_column_name]



  es_var <- escalc(measure="PLO", xi=xi, ni=ni )
  data.ab[, c("yi", "vi")] <-  es_var

  ## RELEVEL TRT CODE HERE

  ## if there is no reference treatment name, select the lexicographically first
  if ( is.na(reference_treatment_name)) {
    reference_treatment_name <- sort(levels(data.ab[, "trt"])[as.numeric(data.ab[, "trt"])])[1]
    cat(paste("No reference treatment given; setting reference = ", reference_treatment_name,"\n", sep=""))
  }

  data.ab <- within(data.ab, trt <- relevel(trt, ref=reference_treatment_name))

  ## count multiplicity of edges to see if we will need a random effects model
  ## do this in igraph

  gnet <- extract_graph_from_network(mtc.network(data.ab=net_data))
  edge_multiplicity <- count_multiple(gnet, eids=E(gnet))
  how_many_trts <- length(V(gnet))
  how_many_edges <- length(edge_multiplicity)
  max_comparisons_in_edges <- max(edge_multiplicity)

  ## The FE formula and the RE formula
  fe_formula_string <- " ~ 1 + trt"
  re_formula_string <- paste(" ~ 1 | ", study_column_name, "/", "trt", sep='')

  if( how_many_trts >= 3 & max_comparisons_in_edges>=4 ) {
    cat(paste("running a RE analysis: max number of studies per comparison= ",
    max_comparisons_in_edges, "\n",sep=""))
    results <- rma.mv(yi, vi, mods= as.formula(fe_formula_string),
                      data=data.ab,
                      random= as.formula(re_formula_string) ,
                      method="REML")
  } else {
    cat(paste("running a FE analysis: max number of studies per comparison= ",
    max_comparisons_in_edges, "\n",sep=""))


    results <- try({
      rma.mv(yi, vi, mods= as.formula(fe_formula_string),
```

```
            data=data.ab,
            random= as.formula(re_formula_string) ,
            method="REML")} )
    if (is.error(results)) {
      new_fe_formula_string <- paste(" ~ 1 + trt * ", study_column_name, sep='')
      results <- rma.uni(yi, vi, mods= as.formula(new_fe_formula_string),
                         data=data.ab, method="FE")

    }


  }


  league_table <- get_league_table(network_results_with_correct_reference=results,
  reference_treatment_name= paste("reference (", reference_treatment_name, ")", sep=""),
                                   treatment_column_name="trt")

  relative_effect_table <- get_relative_effect_table(data.ab, xi, ni,
                                                     study_column_name=study_column_name,
                                                     treatment_column_name=treatment_column_name,
                                                     #reference_treatment_name=NA,
                                                     digits=digits)

  ret_list <- list(
    "relative_effect_table_re" <- relative_effect_table,
    "league_table" <- league_table)

  return(ret_list)

}



################## Node-Splitting Analysis Function ####################################
nodesplitting<-function(trt1,trt2,outcome){

  cleandata<-clean_data(outcome)
  coarse_network <-make_coarse_network(cleandata)
  g <- extract_simplified_graph_from_network(coarse_network)

  V(g)$label <- gsub("_", ",", unlist(V(g)$name))

  graph_list <- decompose(g)
  data_node_name_char <-
     levels(coarse_network$data.ab[,"treatment"])[as.numeric(coarse_network$data.ab[,"treatment"])]
  treatments_name_char <-
     levels(coarse_network$treatments[,"id"])[as.numeric(coarse_network$treatments[,"id"])]

  for (a_graph_index in 1:length(graph_list)) {
    cat(paste("Now examining network", a_graph_index, "of", length(graph_list),
       " -- outcome:", outcome_sheet, "...\n", sep=' '))


    ## Check in network_ma_results_path if object exists
    ##       the name pattern is:
    ##       res_network_ma.outcome_stem.subgraph_agraphindex.r
```

```
##       and the results objects are dumplists that you simply
##       source back to read them.
##              Aside note: name can be split by dots using:
##              unlist(strsplit("a.b.c", "[.]"))



#splitnodefilename<-paste("splitnode",outcome,trt1,trt2,"coarse",
#    paste("subgraph_",a_graph_index, sep='' ), 'csv', sep='.')


some_node_names <- unlist(V(graph_list[[a_graph_index]])$name)

selector_data <- is.element(data_node_name_char, some_node_names)
selector_treatments <- is.element(treatments_name_char, some_node_names)

net_data <- coarse_network$data.ab[selector_data, ]
# so far so good
# a problem is that the net_data$treatment is a factor which knows the full levels
# before we dropped the nodes.
# gemtc checks the factors levels, not the levels with >0
# clumsy fix: make it a factor again
net_data$treatment <- factor(net_data$treatment)

total_network<-mtc.network(net_data)
totalretable<-get_relative_effect_table(data.ab=total_network$data.ab,
                                        xi=total_network$responders,
                                        ni=total_network$sampleSize,
                                        study_column_name="study",
                                        # reference_treatment_name=NA,
                                        treatment_column_name="treatment")
#potential studies have one arm = trt1
potential<-net_data[which(net_data$treatment==trt1),"study"]
#for each of these potential studies, see if another arm = trt2
for (i in 1:length(potential)){
  rows<-which(net_data$study==potential[i])
  if (length(which(net_data$treatment[rows]==trt2))==1){
    #split the trt1 node in studies that have both trt1 and trt2
    net_data$treatment[which(net_data$treatment==trt1 &
    net_data$study==potential[i])]<-paste(trt1,"_tmp",sep="")
  }
}
new_network <- mtc.network(data=net_data)
#get comparison between trt1 and trt1_tmp
#re<-relative.effect(mtc.run(mtc.model(new_network)),trt1,paste(trt1,"_tmp",sep=""))
cat(paste(trt1," vs. ",trt2,sep=""))
retable<-get_relative_effect_table(data.ab=new_network$data.ab,
                                   xi=new_network$data.ab$responders,
                                   ni=new_network$data.ab$sampleSize,
                                   study_column_name="study",
                                   #reference_treatment_name=NA,
                                   treatment_column_name="treatment")

#pull trt1_tmp vs. trt1 from the relative effect table
re<-retable[trt1,paste(trt1,"_tmp",sep="")]
est<-strsplit(re," ")[[1]][1]
est<-as.numeric(gsub("[^0-9.]","",est))
```

```
    lower<-strsplit(re," ")[[1]][2]
    lower<-as.numeric(gsub("[^0-9.]", "", lower))
    upper<-strsplit(re," ")[[1]][3]
    upper<-as.numeric(gsub("[^0-9.]", "", upper))
    #pull the direct estimate from the relative effect table
    direct<-retable[paste(trt1,"_tmp",sep=""),trt2]
    #pull the indirect estimate from the relative effect table
    indirect<-retable[trt1,trt2]
    #pull the network estimate from the original network relative effect table
    network<-totalretable[trt1,trt2]

    #calculating p-value from relative.effect output
    SE<-(upper-lower)/(2*1.96)
    Z<-est/SE
    p<-exp(-0.717*Z-0.416*(Z^2))

    #create a table to store values
    t<-mat.or.vec(4,3)
    t[1,1]<-paste(trt1," vs. ",trt2,sep="")
    t[2,1]<-"direct"
    t[3,1]<-"indirect"
    t[4,1]<-"network"
    t[1,2]<-p
    t[2,3]<-direct
    t[3,3]<-indirect
    t[4,3]<-network
    write.csv(t,paste("splitnode",outcome,trt1,trt2,"coarse",
        paste("subgraph_",a_graph_index, sep='' ), 'csv', sep='.'))
  }

}


################### League Table Function ############################
get_league_table  <- function(network_results_with_correct_reference, reference_treatment_name=NA,
                              treatment_column_name="treatment"
) {

  cat("in function get_league_table\n")

  res <- network_results_with_correct_reference

  treatment_names  <- gsub(paste("^", treatment_column_name, sep=''), "", rownames(res$b))
  treatment_names[1] <- reference_treatment_name
  treatment_names <- gsub("_plus_", "+", treatment_names)
  treatment_names <- gsub("_or_", "|", treatment_names)
  treatment_names <- gsub("_", ",", treatment_names)

  n_treatments <- length(res$b)

  contrasts <- diag(n_treatments-1)
  contrasts <- rbind( rep(0, n_treatments-1), contrasts )
  # league_table_list <- as.list(rep(NA, length(rates_in_reference_treatment)) )

  tmp_pred <- predict(res, newmods=contrasts, intercept=TRUE )

  # coerse the list.rms object to a dataframe by print() !!!
```

```
  tmp_pred <- as.data.frame(print.list.rma(tmp_pred))

  # turn to probabilities
  tmp_pred <- invlogit(tmp_pred)

  #  Select the colums with the results.
  #     RE models return means and predictions for a new setting
  #     FE models return only the means
  which_tmp_pred_columns <- is.element(colnames(tmp_pred),c("pred", "ci.lb", "ci.ub", "cr.lb", "cr.ub"))
  tmp_pred <- tmp_pred[, which_tmp_pred_columns]

  # name the rows and save
  rownames(tmp_pred) <- treatment_names

  Estimate_Interval<-paste(format(round(100*tmp_pred$pred,1),nsmall=1)," (",
                           format(round(100*tmp_pred$ci.lb,1),nsmall=1),",
                           ",format(round(100*tmp_pred$ci.ub,1),nsmall=1),")",sep="")
  Estimate_Interval <- gsub("( 0.0,", paste("(<0.5,", sep=""), Estimate_Interval,fixed=TRUE)
  Estimate_Interval <- gsub("^ 0.0", paste("<", 0.5, sep=""), Estimate_Interval)
  pred<-as.data.frame(Estimate_Interval,stringsAsFactors=FALSE)

  # IF THIS WAS A RE ANALYSIS, DO PREDICTION
  if(is.element("cr.lb", colnames(tmp_pred))) {
    Prediction_Interval <- paste(format(round(100*tmp_pred$pred,1),nsmall=1)," (",
                                 format(round(100*tmp_pred$cr.lb,1),nsmall=1),",
                                 ",format(round(100*tmp_pred$cr.ub,1),nsmall=1),")",sep="")
    Prediction_Interval <- gsub("( 0.0,", paste("(<0.5,", sep=""), Prediction_Interval,fixed=TRUE)
    Prediction_Interval <- gsub("^ 0.0", paste("<", 0.5, sep=""), Prediction_Interval)
    pred<-as.data.frame(cbind(Estimate_Interval,Prediction_Interval),stringsAsFactors=FALSE)
  }


  rownames(pred) <- treatment_names

  return(pred)

}

################### Relative Effect Table Function ############################
get_relative_effect_table<-function(data.ab, xi, ni,
                                    study_column_name="study",
                                    treatment_column_name="treatment",
                                    digits=2){

  cat("in function get_relative_effect_table\n")

  data.ab <- as.data.frame(data.ab)

  ## a copy of the trt factor, to re-level
  data.ab[, "trt"] <-  data.ab[, treatment_column_name]

  es_var <- escalc(measure="PLO", xi=xi, ni=ni )
  data.ab[, c("yi", "vi")] <-  es_var

  ## set up the empty table
  relative_effect_table <- matrix(,nrow=length(levels(data.ab$treatment)),
                                  ncol=length(levels(data.ab$treatment)))
```

```
relative_effect_table <- as.data.frame(relative_effect_table)

reference_treatment_name_list <- unique(levels(data.ab[,"trt"])[as.numeric(data.ab[,"trt"])])
reference_treatment_name_list <- reference_treatment_name_list[order(reference_treatment_name_list)]


## count multiplicity of edges to see if we will need a random effects model
## do this in igraph

gnet <- extract_graph_from_network(mtc.network(data.ab=net_data))
edge_multiplicity <- count_multiple(gnet, eids=E(gnet))
how_many_trts <- length(V(gnet))
how_many_edges <- length(edge_multiplicity)
max_comparisons_in_edges <- max(edge_multiplicity)

## The FE formula and the RE formula
fe_formula_string <- " ~ 1 + trt"
re_formula_string <- paste(" ~ 1 | ", study_column_name, "/", "trt", sep='')

run_FE <- FALSE
run_RE <- FALSE

if( how_many_trts >= 3 & max_comparisons_in_edges>=4 ) {
  cat(paste("running a RE analysis: max number of studies per comparison= ",
  max_comparisons_in_edges, "\n",sep=""))
  run_RE <- TRUE
} else {
  cat(paste("running a FE analysis: max number of studies per comparison= ",
  max_comparisons_in_edges, "\n",sep=""))
  run_FE <- TRUE   # superfluous
}



y<-0
for (ref in reference_treatment_name_list) {
  y<-y+1

  data.ab <- within(data.ab, trt <- relevel(trt, ref=ref))

  if(run_RE==TRUE) {
    results <- rma.mv(yi, vi, mods= as.formula(fe_formula_string),
                      data=data.ab,
                      random= as.formula(re_formula_string) ,
                      method="REML")
  } else {



    results <- try({
      rma.mv(yi, vi, mods= as.formula(fe_formula_string),
             data=data.ab,
             random= as.formula(re_formula_string) ,
             method="REML")} )
    if (is.error(results)) {
      new_fe_formula_string <- paste(" ~ 1 + trt * ", study_column_name, sep='')
      results <- rma.uni(yi, vi, mods= as.formula(new_fe_formula_string),
```

```
                          data=data.ab, method="FE")}


    }


    # exponentiate before packing results, to get odds ratios,
    est <- exp(results$b)
    low <- exp(results$ci.lb)
    high <- exp(results$ci.ub)

    est_str <- paste(round(est, digits=digits))
    est_str <- gsub("^0$", paste("<", 0.5*10^(-digits), sep=""), est_str)

    low_str <- paste(round(low, digits=digits))
    low_str <- gsub("^0$", paste("<", 0.5*10^(-digits), sep=""), low_str)

    high_str <- paste(round(high, digits=digits))
    high_str <- gsub("^0$", paste("<", 0.5*10^(-digits), sep=""), high_str)

    relative_effect_ref <- paste(est_str, " (",
                                 low_str, ", ",
                                 high_str, ")", sep="")

    relative_effect_ref <- as.data.frame(relative_effect_ref, stringsAsFactors=FALSE)

    treatment_names  <- rownames(results$b)
    treatment_names  <- gsub("trt","", treatment_names)
    treatment_names[1] <- ref

    treatment_names <- gsub("_plus_", "+", treatment_names)
    treatment_names <- gsub("_or_", "|", treatment_names)
    treatment_names <- gsub("_", ",", treatment_names)


    rownames(relative_effect_ref) <- treatment_names
    relative_effect_ref[1, 1] <- ref
    relative_effect_table[, y] <- relative_effect_ref[order(rownames(relative_effect_ref)),]

  }

  colnames(relative_effect_table) <- treatment_names[order(treatment_names)]
  rownames(relative_effect_table) <- treatment_names[order(treatment_names)]
  return(relative_effect_table)
}
```

2.3 Analysis

```
library(metafor)
library(gemtc)
library(igraph)
source("NMA_functions.R")

total_list<-list("_d_HCBCC_cleandata_subgraph1.csv",
                 "_d_HCBCC_cleandata_subgraph2.csv",
                 "_d_HCBCC_cleandata_subgraph3.csv",
                 "_d_RecurBCC_cleandata_subgraph1.csv",
                 "_d_RecurBCC_cleandata_subgraph2.csv",
                 "_d_CosmeticObserverBCC_cleandata_subgraph1.csv",
                 "_d_CosmeticObserverBCC_cleandata_subgraph2.csv",
                 "_d_CosmeticPatientBCC_cleandata_subgraph1.csv",
                 "_d_CosmeticPatientBCC_cleandata_subgraph2.csv",
                 "_d_CosmeticPatientBCC_cleandata_subgraph3.csv",
                 "_d_CosmeticPatientBCC_cleandata_subgraph4.csv"
                 )
outcome_list<-list("HCBCC_subgraph_1","HCBCC_subgraph_2","HCBCC_subgraph_3",
                   "RecurrBCC_subgraph_1","RecurrBCC_subgraph_2",
                   "CosmeticObserverBCC_subgraph_1","CosmeticObserverBCC_subgraph_2",
                   "CosmeticPatientBCC_subgraph_1","CosmeticPatientBCC_subgraph_2",
                   "CosmeticPatientBCC_subgraph_3","CosmeticPatientBCC_subgraph_4")

for (i in 1:length(total_list)){
outcome_subgraph<-total_list[[i]]
outcome<-outcome_list[[i]]
net_data<-read.csv(outcome_subgraph)
leaguecsvfilename<-paste("league_table",outcome, 'csv', sep='.')
freqretablecsvfilename<-paste("freq_relative_effect_table",outcome,'csv', sep='.')
results_list <-analyze_network_frequentist(data.ab=net_data,
                              xi=net_data$responders,
                              ni=net_data$sampleSize,
                              study_column_name="study",
                              reference_treatment_name=NA,
                              treatment_column_name="treatment")
write.csv(results_list[[2]],leaguecsvfilename)
write.csv(results_list[[1]],freqretablecsvfilename)
}

treatment1<-c("A","A","C1","E1","E1","E1","F2",
              "A","A","A","C1","C1","C1","D1","E1","E1",
              "A","A","E1",
              "A","A","C1"
              )

treatment2<-c("C1","E1","E2","E2","F2","J","J",
              "E1","E2","F2","D1","E1","E2","F2","E2","F2",
              "E1","F2","F2",
              "C1","E1","E1"
              )
outcome_list<-list("HCBCC","HCBCC","HCBCC","HCBCC","HCBCC","HCBCC","HCBCC",
                   "RecurrBCC","RecurrBCC","RecurrBCC","RecurrBCC","RecurrBCC",
                   "RecurrBCC","RecurrBCC","RecurrBCC","RecurrBCC",
                   "CosmeticObserverBCC","CosmeticObserverBCC","CosmeticObserverBCC",
                   "CosmeticPatientBCC","CosmeticPatientBCC","CosmeticPatientBCC")
```

```
data_list<-list("_d_HCBCC_cleandata_subgraph1.csv","_d_HCBCC_cleandata_subgraph1.csv",
                "_d_HCBCC_cleandata_subgraph1.csv","_d_HCBCC_cleandata_subgraph1.csv",
                "_d_HCBCC_cleandata_subgraph1.csv","_d_HCBCC_cleandata_subgraph1.csv",
                "_d_HCBCC_cleandata_subgraph1.csv",
              "_d_RecurBCC_cleandata_subgraph1.csv","_d_RecurBCC_cleandata_subgraph1.csv",
              "_d_RecurBCC_cleandata_subgraph1.csv","_d_RecurBCC_cleandata_subgraph1.csv",
              "_d_RecurBCC_cleandata_subgraph1.csv","_d_RecurBCC_cleandata_subgraph1.csv",
              "_d_RecurBCC_cleandata_subgraph1.csv","_d_RecurBCC_cleandata_subgraph1.csv",
              "_d_RecurBCC_cleandata_subgraph1.csv",
              "_d_CosmeticObserverBCC_cleandata_subgraph2.csv",
              "_d_CosmeticObserverBCC_cleandata_subgraph2.csv",
              "_d_CosmeticObserverBCC_cleandata_subgraph2.csv",
              "_d_CosmeticPatientBCC_cleandata_subgraph1.csv",
              "_d_CosmeticPatientBCC_cleandata_subgraph1.csv",
              "_d_CosmeticPatientBCC_cleandata_subgraph1.csv"
              )


# Run the analyses

for (i in 1:length(outcome_list)){
  trt1<-treatment1[i]
  trt2<-treatment2[i]
  outcome<-outcome_list[[i]]
  data<-data_list[[i]]
  net_data<-read.csv(data)
  nodesplit_results<-nodesplitting(trt1,trt2,outcome,net_data)
}
```