Model and R code description for SCTLD Etiology Elicitation

Model - M1

We propose exploring a simple hierarchical framework to summarize the information from elicitation method M1. Using this approach we assume that each of the experts are random draws/observation from a population of experts each of whom has a belief about the support for each etiological hypothesis. We are interested in the population-level parameters for each weight as a measure of the current state of knowledge. Thus, our model, conditional on some true current state of knowledge regarding the etiology of SCTLD is simply:

$$\begin{aligned} y_{i,j} &\sim ddirch\left(\overrightarrow{\alpha}\right) \\ \alpha_j &\sim dgamma\left(1,1\right), \end{aligned}$$

where $y_{i,j}$ is the normalized weight (i.e., between 0 and 1) for the i^{th} expert for the j^{th} of J hypotheses, and α_j is a concentration parameter for the Dirichlet distribution associated with the j^{th} hypothesis. The benefit of this formulation appears to be multi-fold. First, it allows the precision to properly increase as the number of experts surveyed increases. It also can easily handle any number of points used to assign weights by the experts because $y_{i,j}$ must be normalized to lie between 0 and 1. Note for estimation in NIMBLE values should not be exactly zero or 1 which will result in an infinite value in the Dirichlet density within the package. To account for this we add or substract a small value 1e-12 to any expert elicited probabilities that were exactly 0 or 1, respectively. Our method also provides us a way to assess the amount of learning that has happened since the discovery of SCTLD because we know that at maximum uncertainty we would set $\alpha_j = 1$ (i.e., each hypothesis has equal weight). Thus, the expected value for each hypothesis weight (i.e., population-level) is

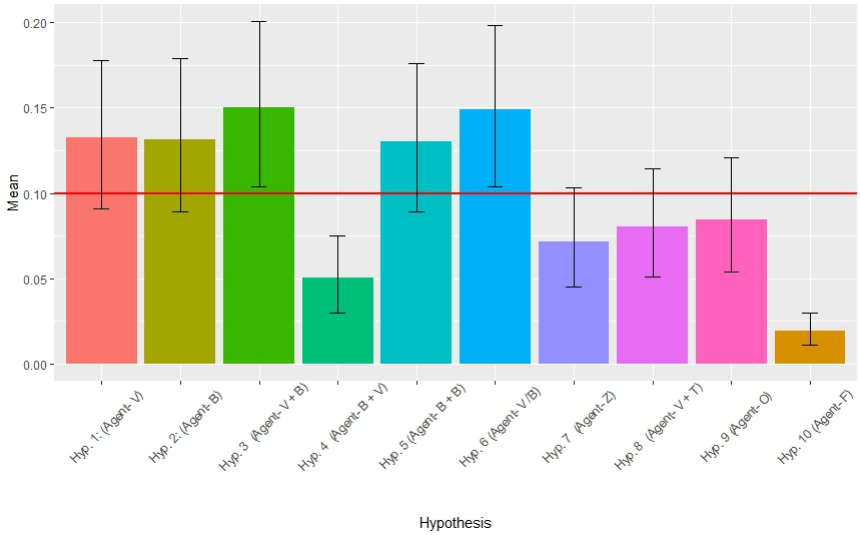
$$\bar{W}_j = \frac{\alpha_j}{\sum_j \alpha_j},$$

and the amount of learning or relative evidence, E_j, for each hypothesis could be estimated as:

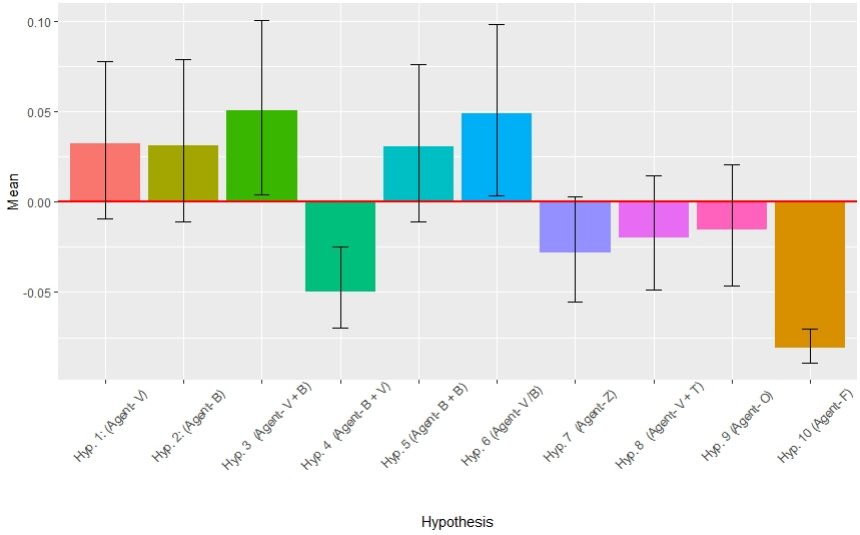
$$E_j = \bar{W}_j - \frac{1}{J}.$$

The relative evidence will take on positive values where there is evidence for a hypothesis and negative values if evidence against a hypothesis based on the experts' assessments.

Example plots from our SCTLD elicitation are shown below. The first plot shows the estimated $\bar{W}_j$ with the red line indicating maximum uncertainty.



The second plot shows the relative evidence for each hypothesis.



Code

Below we describe the code we used to run this model on an example data set to demonstrate its potential application for our problem.

Data Preparation

We begin by loading the necessary libraries that will be used in the subsequent analyses.

```
library(nimble)
library(dplyr)
library(tidyr)
```

Next we generate data similar to what was provided by the experts. It is used as an evaluation of the approach.

```
#load data
all_data=readRDS("data_M1_nimble.RDS")

hyp.names <- c("Hyp. 1 (Agent- V)", "Hyp. 2 (Agent- B)",
               "Hyp. 3 (Agent- V + B)", "Hyp. 4 (Agent- B + V)",
               "Hyp. 5 (Agent- B + B)", "Hyp. 6 (Agent- V /B)",
               "Hyp. 7 (Agent- Z)", "Hyp. 8 (Agent- V + T)",
               "Hyp. 9 (Agent- O)", "Hyp. 10 (Agent- F)")

#data summary
summary <- all_data%>%
  filter(Round=="R2")%>%
  select(name,hypothesis,data)%>%
  pivot_wider(names_from=hypothesis,values_from=data)
summary <- data.frame(summary)
experts=as.matrix(summary[,2:11])#this removed the name column
dimnames(experts)=NULL
sum(experts[1,])#sum to 100

#account for exactly zero or one values = Inf density estimates
experts[experts == 0] <- 0.000000001
experts[experts == 1] <- 0.999999999

#create probability from point assignments
for(i in 1:nrow(experts)){
  experts[i,] <- experts[i,]/sum(experts[i,])
}

sum(experts[1,])#now sums to 1

#Number of hypotheses
num.hyp<- length(experts[1,])

#Number of experts
```

```
num.experts <- length(experts[,1])
```

Estimation

We begin estimation by setting up the necessary inputs for use in Nimble. These include the data, the constants and the initial, which each have to be setup as a list structure.

```
simData<-list( y = experts
)

##Create model constants
simConst<-list( n = num.experts, m = num.hyp
)

##Create initial values
simInits <- list(
  #assume all hypotheses are equally likely
  alpha = rep(1, simConst$m)
)
```

Next we specify the parameters for running the MCMC algorithm.

```
nchains <- 3
reps <- 10000
nburnin <- 0.1*reps
```

The code for Nimble leverages the BUGS pseudo-language. For our model for elicitation method M1, it is as follows:

```
library(nimble)
simcode <- nimbleCode({

  # Priors
  for(i in 1:m){
    alpha[i] ~ dgamma(1, 1)
  }

  # Likelihood
  for(i in 1:n){
    y[i,1:m] ~ ddirch(alpha[1:m])
  }

  #Derived parameters
```

```
  for(i in 1:m){
    W[i] <- alpha[i]/sum(alpha[1:m])#E[wt] for each hypothesis
    learning[i] <- W[i] - 1/m #how much change from equal weight
  }

})
```

We then compile the code.

```
simtest<-nimbleModel(code= simcode, name="test",
                     constants = simConst,
                     data = simData, inits = simInits)
```

Finally, the MCMC estimation procedure is ran for the specified model.

```
mcmcout1<-nimbleMCMC(model=simtest, nchains=nchains,
                     nburnin = nburnin,
                     niter=reps, summary=TRUE, WAIC=F,
                     monitors = c("alpha", "W", "learning"
                     ))
```

Results

We save the results of the MCMC estimation procedure.

```
#saveRDS(mcmcout1,"m1test_ellen.RDS")
```

Next we create a function to create diagnostic plots and examine them for evidence of non-convergence of the MCMC chains.

```
library(coda)
library(ggplot2)
mcmc.coda <- mcmc.list(lapply(mcmcout1$samples, mcmc))

#Function to create diagnostic plots and Gelman-Rubin statistics
diag.plots <- function(x,nchains){
  xx <- colnames(mcmc.coda$chain1)[
    grep(x,colnames(mcmc.coda$chain1))]
  for(j in xx){
    if(mean(mcmc.coda[,j][[1]])==0){
      next   #skip if parm set to zero
    }
    plot(mcmc.coda[,j], main=j)
    print(j)
```

```
    print(gelman.diag(mcmc.coda[,j]))
  }
}

for(i in c("alpha")){
  diag.plots(i)
}
```

Finally we create bar plots of the posterior mean and the 95% credible bounds for the weights and for the "Bayesian learning".

```
df <- as.data.frame(mcmcout1$summary$all.chains)
df <- df[grep("W",row.names(df)),]
for(i in 1:num.hyp){
  df$hyp[i] <- hyp.names[i]
}


p1 <- ggplot(data=df, aes(y=Mean,x=reorder(hyp,1:10),fill=hyp)) +
  geom_bar(stat="identity", show.legend = F) +
  xlab("Hypothesis") +
  geom_hline(yintercept = 1/num.hyp,col="red", size=1) +
  geom_errorbar(aes(ymin=df[,4], ymax=df[,5]), width=.2,
                position=position_dodge(.9))+
  theme(axis.text.x = element_text(angle = 45,
                        margin = unit(c(10, 0, 0, 0), "mm")))

p1
df <- as.data.frame(mcmcout1$summary$all.chains)
df <- df[grep("learn",row.names(df)),]
for(i in 1:num.hyp){
  df$hyp[i] <- hyp.names[i]
}


#Following corresponds to Figure 2 in Manuscript
p2 <- ggplot(data=df, aes(y=Mean,x=reorder(hyp,1:10),fill=hyp)) +
  geom_bar(stat="identity", show.legend = F) +
  xlab("Hypothesis") +
  geom_hline(yintercept = 0,col="red", size=1) +
  geom_errorbar(aes(ymin=df[,4], ymax=df[,5]), width=.2,
                position=position_dodge(.9))+
  theme(axis.text.x = element_text(angle = 45,
                          margin = unit(c(10, 0, 0, 0), "mm")))
```

```
p2
```

Model - M2

The elicitation method requires a different perspective because each study is evaluated independently, and there is a desire to track the accumulation of evidence through time. We propose to use a hurdle model to capture the evidence for or against a particular study through time as scientific studies are conducted. We first calculate the hurdle probability of a study demonstrating no support for a hypothesis (i.e., elicited value of 0):

$$zeros_{i,j,k} \sim dbern\left(pzero_{j,k}\right),$$

where $zeros_{i,j,k}$ is an indicator of a zero value provided by the i^{th} expert for the j^{th} hypothesis and the k^{th}study (note: studies are ordered chronologically), and $pzero_{j,k}$ is the probability there is no scientific support for the j^{th} hypothesis provided by the k^{th}study. For all instances where the experts did believe there was support for a hypothesis provided by a study, we normalized and converted the elicited scores to probability space (i.e., between 0 and 1). We then use the probit link function and assume these probabilities arise from the probit transformation of a latent measure of learning/evidence:

$$p_{i,j,k} = \Phi\left(L_{i,j,k}\right) \forall p_{i,j,k} > 0,$$

where $L_{i,j,k}$ is the learning/evidence determined by the i^{th} expert for the j^{th} hypothesis and the k^{th}study (note: studies are ordered chronologically), $\mathbf{\Phi}$ represents the CDF function for the standard normal distribution, and $p_{i,j,k}$ is the elicited normalized score. We then use these values to estimate the posterior distribution of $L_{j,k}$:

$$L_{i,j,k} \sim dnorm\left(\mu_{j,k}, \sigma_{j,k}\right),$$

where $\mu_{j,k}$ is the mean for the j^{th} hypothesis and the k^{th}study and $\sigma_{j,k}$ is the associated standard deviation. The hurdle model is necessary with this approach because zero values for $p_{i,j,k}$ result in a -Inf density estimate in our probit model. Using the hurdle model permits a cumulative summary of knowledge across experts through time. The estimated posterior for the total learning/evidence for a hypothesis across experts, given the k^{th}study is completed, is the sum of all the past and current evidence, which can simply be calculated from several derived parameters and sampling from the posteriors of $pzero_{j,k}$, $\mu_{j,k}$ and $\sigma^2_{i,j,k}$:

$$pzerot_{j,k} = \Phi\left[\left(\frac{1}{k}\right)\sum_{k=1}^{k}\Phi^{-1}\left(pzero_{j,k}\right)\right],$$
$$E\left[T_{j,k}\right] = \left(\frac{1}{k+1}\right)\sum_{k=0}^{k}\mu_{j,k}.$$

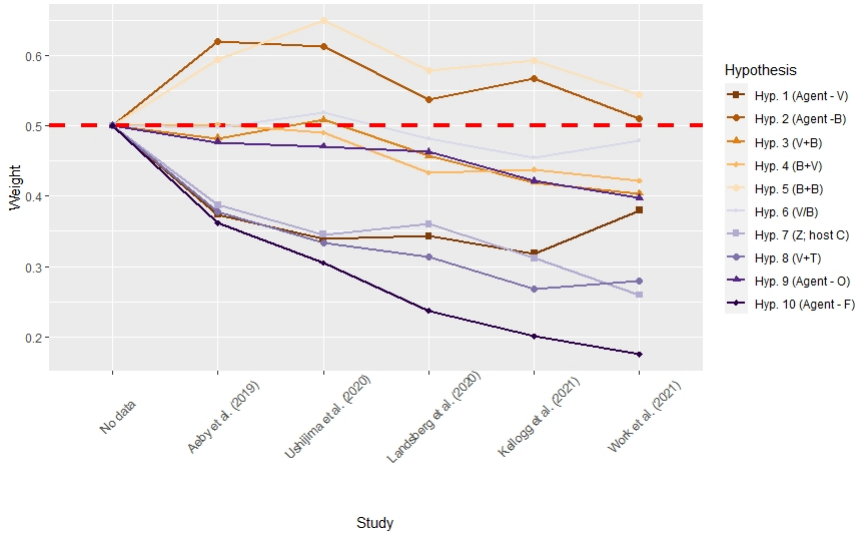
Note for estimating the expectation of T, we start with $k = 0$ because this represents the state of complete uncertainty or lack of knowledge, representing a common starting point for all hypotheses. On the probit scale this is presented as $T_{j,0} = 0 \forall i, j$. This not the case for $pzerot_{j,k}$, which is tied to each study and therefore we start at $k = 1$. With this structure, as evidence for a hypothesis accumulates, the values of T increase and those of $pzerot$ decrease. Conversely, as evidence against a hypothesis mounts this results in decreasing T and increasing $pzerot$ values. Also, because we are using $\mu_{j,k}$ in the estimation of $T_{j,k}$, we are estimating the posterior distribution of the expectation of $T_{j,k}$. Finally, we can estimate the weight or probability of interest as follows:

$$W_{j,k} = \Phi\left(E\left[T_{j,k}\right]\right) \times \left(1 - pzerot_{j,k}\right),$$

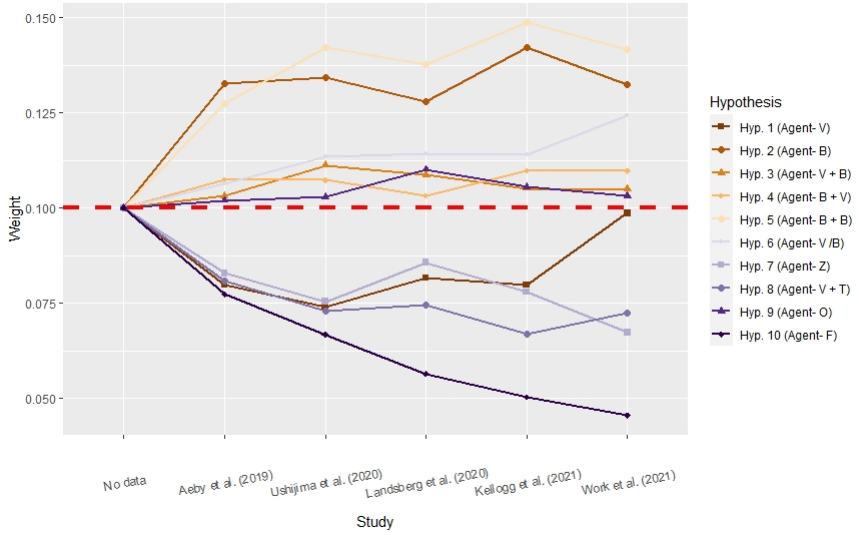
where $W_{j,k}$ is the current weight of evidence for the j^{th} hypothesis given the k^{th} study is completed. Additional weights based on study quality could be easily incorporated into the estimation of $T_{j,k}$. Given some experts may feel that a study does not address a particular hypothesis the following recursive, mixture formulation could be used to address this aspect of the elicitation process:

$$E\left[T_{j,k}\right] = \begin{cases} k = 0 \rightarrow & 0 \\ k > 0 \rightarrow & \left(\frac{1}{k+1}\right)\left[\left(\sum_{l=0}^{k-1}\mu_{j,l}\right) \times \left(1 + \frac{a_{j,k}}{N_{j,k}}\right) + \mu_{j,k} \times \left(\frac{N_{j,k} - a_{j,k}}{N_{j,k}}\right)\right] \end{cases},$$

where $N_{j,k}$ is the number of experts who assigned a weight > 0 and $a_{j,k}$ is the number of experts that specified the k^{th}study does not address the j^{th} hypothesis. This mixture formulation weights past knowledge against current knowledge based on the proportion of experts that feel a study addresses a specific hypothesis. Then $W_{j,k}$ is calculated as before. A plot of results from the SCTLD data is shown below where the dashed lined represents complete uncertainty surrounding the etiology of SCTLD.



A plot of the normalized weights is also provided.



Code

The code to implement this approach is described below.

Data Preparation

We begin by loading the necessary libraries that will be used in the subsequent analyses.

```
library(DirichletReg)
library(nimble)
library(coda)
library(ggplot2)
library(knitr)
library(dplyr)
```

Next we generate data similar to what was provided by the experts. It is used as an evaluation of the approach.

```
#load data
all_data=readRDS("all_data_combined_M2_nimble.RDS")

#data formatting
#filter to Yes's
summary <- all_data%>%
  filter(predictions=="Yes (E=1)")
summary$data=ifelse(summary$data=="N/A","NA",summary$data)
summary$data=ifelse(summary$data=="n/a","NA",summary$data)

#put studies in chronological order that they were published
studynames=c("Aeby et al. (2019)","Ushijima et al. (2020)",
             "Landsberg et al. (2020)","Kellogg et al. (2021)",
             "Work et al. (2021)")
hypothesisnames=c("Hyp. 1 (Agent- V)", "Hyp. 2 (Agent- B)",
                  "Hyp. 3 (Agent- V + B)",
                 "Hyp. 4 (Agent- B + V)", "Hyp. 5 (Agent- B + B)",
                 "Hyp. 6 (Agent- V /B)", "Hyp. 7 (Agent- Z)",
                 "Hyp. 8 (Agent- V + T)", "Hyp. 9 (Agent- O)",
                 "Hyp. 10 (Agent- F)")

expertnames=unique(summary$name)
finalarray=array(NA,dim=c(length(expertnames),
                    length(hypothesisnames),length(studynames)))
#create data array
for (i in 1:length(studynames)){
  for (j in 1:length(hypothesisnames)){
    for (k in 1:length(expertnames)){
      value=summary[summary$name==
                  expertnames[k]&summary$studies==studynames[i] &
                  summary$hypotheses==hypothesisnames[j],"data"]
```

```
      value=if(identical(value, character(0))){NA_character_} else
        {value}
      value=as.numeric(as.character(value))
      finalarray[k,j,i]=value
    }}}

experts=finalarray/100
max(experts,na.rm=TRUE)
min(experts,na.rm=TRUE)

#Number of hypotheses
num.hyp<- 10

#Number of experts
num.experts <- 15

#Number of studies
num.studies <- 5

#Account for zero values
indices <-which(experts==0, arr.ind = T)
zeros <- array(0,dim=dim(experts))
zeros[indices] <- 1  #indicator of zeros in the data array
nzeros <- apply(zeros,c(2,3),sum,na.rm=T)

#Account for non-zero values
L.experts <- qnorm(experts)  #convert to probit scale

#Count the number of NAs
nna <- apply(L.experts, c(2,3),  ##this counts NA and non NAs
             function(x) sum(is.na(x)))

nonna <- apply(L.experts, c(2,3),
               function(x) sum(!is.na(x)))

L.experts[indices] <- NA  #set zeros to NA...so won't affect
                ##likelihood and done after counting NAs above
```

Estimation

We begin estimation by setting up the necessary inputs for use in Nimble. These include the data, the constants and the initial, which each have to be setup as a list structure.

```
##Create data
simData<-list( y = L.experts, zeros = zeros
)

##Create constants
simConst<-list( n = num.experts, m = num.hyp,
                 nstudies = num.studies,
                nna = nna, nonna = nonna,
                nonzeros= num.experts - nzeros
)

##Create initial values
simInits <- list(
  #initial value - all hypotheses are equal
  mu = matrix(0, simConst$m, simConst$nstudies),
  sd = matrix(1, simConst$m, simConst$nstudies),
  pi = matrix(0.05, simConst$m, simConst$nstudies)
)
```

Next we specify the parameters for running the MCMC algorithm.

```
nchains <- 3
reps <- 10000
nburnin <- 0.1*reps
```

The code for Nimble leverages the BUGS pseudo-language. For our model for elicitation method M2, it is as follows:

```
simcode <- nimbleCode({

  # Priors

  for(i in 1:m){
    for(j in 1:nstudies){
      p[i,j] ~ dbeta(1, 100)  # prob value=0
      mu[i,j] ~ dnorm(0, 1)
      sd[i,j] ~ dunif(0, 1)
    }
  }


  # Likelihood

  for(i in 1:n){
    for(j in 1:m){
      for(k in 1:nstudies){
```

```
          zeros[i,j,k] ~ dbern(p[j,k])
          y[i,j,k] ~ dnorm(mu[j,k],sd=sd[j,k])
      }
    }
  }

  #Derived parameters

  T1[1,1:m] <- 0
  We[1,1:m] <-0.5
  # pi[1:m,1:nstudies] <- qnorm(p[1:m, 1:nstudies])
  for(i in 1:m){    #probability of 0
    for(j in 1:nstudies){
      pi[i,j] <- qnorm(p[i,j])
    }
  }

  for(i in 1:m){    #probability of 0
      pi.use[i,1] <- pi[i,1]
    for(j in 2:nstudies){
      pi.use[i,j] <- sum(pi[i,1:j])
    }
  }

  for(i in 1:m){
    for(j in 1:nstudies){
      # mean probability of zero | completed studies
      pzero[i,j] <- pnorm(pi.use[i,j]/j)

      #latent evidence | completed studies  * prob >0
      T1[j+1,i] <- T1[j,i] +  T1[j,i]*nna[i,j]/nonzeros[i,j] +
            mu[i,j]*nonna[i,j]/nonzeros[i,j]
      #Study weight
      We[j+1,i] <- pnorm(1/(j+1)*T1[j+1,i])*(1 - pzero[i,j])
    }
  }
  for(i in 1:m){
    for(j in 1:(nstudies+1)){
     normW[j,i] <- We[j,i]/sum(We[j,1:m])
    }
  }
})

simtest<-nimbleModel(code= simcode, name="test",
                     constants = simConst,
```

```
                        data = simData,
                        inits = simInits)
```

We then compile the code.
Finally, the MCMC estimation procedure is ran for the specified model.

```
mcmcout1<-nimbleMCMC(model=simtest, nchains=nchains,
                        nburnin = nburnin,
                        niter=reps, summary=TRUE, WAIC=F,
                        monitors = c("mu", "sd", "p", "T1",
                           "We", "pzero", "pi","pi.use","normW"
                        ))
```

Results

We save the results of the MCMC estimation procedure.

```
saveRDS(mcmcout1$samples,"m2.RDS")
saveRDS(mcmcout1$summary,"m2summary.RDS")
```

Next, we create a function to create diagnostic plots and examine them for evidence of non-convergence of the MCMC chains.

```
mcmc.coda <- mcmc.list(lapply(mcmcout1$samples, mcmc))

#Function to create diagnostic plots and Gelman-Rubin statistics
diag.plots <- function(x,nchains){
  xx <- colnames(mcmc.coda$chain1)[
    grep(x,colnames(mcmc.coda$chain1))]
  for(j in xx){
    if(mean(mcmc.coda[,j][[1]])==0){
      next   #skip if parm set to zero
    }
    plot(mcmc.coda[,j], main=j)
    print(j)
    print(gelman.diag(mcmc.coda[,j]))
  }
}

#Note: The diagnostic plots may take a while to run
for(i in c("p", "mu","sd")){
  diag.plots(i)
}
```

We create a plot of the posterior means for the weights for each hypothesis across the time series of studies.

```
df <- as.data.frame(mcmcout1$summary$all.chains)
df <- df[grep("We",row.names(df)),]
df$hyp <- rep(hypothesisnames,each = num.studies+1)
df$study <- factor(rep(c("No data",studynames),num.hyp),
                   levels=c("No data",studynames))
df$hyp <-factor(df$hyp, levels=hypothesisnames)


p1 <- ggplot(data=df, aes(y=Mean, x=study, group= hyp,color=hyp,
                          shape=hyp)) +
  geom_line(size=1)  + xlab("Study") + ylab("Weight") +
  geom_hline(yintercept = 0.5,col="red", size=1.5,
             linetype="dashed") +
  geom_point(size=2)+
  scale_shape_manual(values = rep(15:20, len = num.hyp),
                     name="Hypothesis")+
  scale_color_brewer(palette = "PuOr", name="Hypothesis")+
  theme(axis.text.x = element_text(angle = 10,
            margin = unit(c(10, 0, 0, 0), "mm")))

p1
```

We create a plot of the normalized posterior means for the weights for each hypothesis across the time series of studies.

```
df2 <- as.data.frame(mcmcout1$summary$all.chains)
df2 <- df2[grep("normW",row.names(df2)),]
df2$hyp <- rep(hypothesisnames,each = num.studies+1)
df2$study <- factor(rep(c("No data",studynames),num.hyp),
                    levels=c("No data",studynames))
df2$hyp <-factor(df2$hyp, levels=hypothesisnames)

#Plot corresponds to Figure 3 in the main manuscript
p2 <- ggplot(data=df2, aes(y=Mean, x=study,
                           group= hyp,color=hyp, shape=hyp)) +
  geom_line(size=1)  + xlab("Study") + ylab("Weight") +
  geom_hline(yintercept = 0.1,col="red", size=1.5,
             linetype="dashed") +
  geom_point(size=2)+
  scale_shape_manual(values = rep(15:20, len = num.hyp),
                     name="Hypothesis")+
  scale_color_brewer(palette = "PuOr", name="Hypothesis")+
  # scale_x_discrete(labels= c("No data",studynames)) +
```

```
  theme(axis.text.x = element_text(angle = 10,
                    margin = unit(c(10, 0, 0, 0), "mm")))

p2
```