#Meta-analysis R script

```
# Calculate overall effect size and overall publication bias (Random effect model)
library(meta)
data<-read.csv("mice_all.csv", header = TRUE, row.names = 1, check.names = FALSE)
data
m<-metacont(n1, mean1, sd1,n2, mean2, sd2, data=data, sm="SMD")
m

metabias(m,plotit=TRUE,k.min=2, method.bias="Egger") #Egger's test
metabias(m, k=45, k.min=2, method.bias="Egger") #Egger's test

library(metafor)
library(Matrix)
library(metadat)
trans<-read.csv("micedata_merge.csv",header =T)
RR <-log(trans$mean1/trans$mean2)
var.RR <- (trans$sd1)^2/((trans$n1)*(trans$mean1)^2) +
(trans$sd2)^2/((trans$n2)*(trans$mean2)^2)
trans$yi <- RR + (1/2) *((trans$sd1)^2/((trans$n1)*(trans$mean1)^2) -
(trans$sd2)^2/((trans$n2)*(trans$mean2)^2))
trans$vi <- var.RR + (1/2) *((trans$sd1)^4/((trans$n1)^2*(trans$mean1)^4) +
(trans$sd2)^4/((trans$n2)^2*(trans$mean2)^4))
trans$Geary_T <- (trans$mean1)*((4*(trans$n1)^(3/2))/(1+4*(trans$n1)))/(trans$sd1)
trans$Geary_C <- (trans$mean2)*((4*(trans$n2)^(3/2))/(1+4*(trans$n2)))/(trans$sd2)
write.csv(trans,"micedata_model.csv")


#Literature publication bias
trans<-read.csv("micedata_model.csv",header =T)
trans$vi[trans$vi == 0] <- 0.0000001
res <- rma(yi = yi, vi = vi, data = trans, method=c("REML","DL"))
regtest(res)
funnel(res, legend=TRUE)
trimfill(res)
funnel(trimfill(res), legend=TRUE)

#Fail-Safe N Analysis
fsn(yi, vi, data=trans)


#Calculate likelihood ratio tests
```

```
library(metafor)
trans<-read.csv("micedata_model.csv",header =T)
trans$vi[trans$vi == 0] <- 0.00001
m1<-rma.mv(yi, vi, mods=~endpoints_type, random=list(~1|factor(study)),
method="ML",data=trans)
m2<-rma.mv(yi, vi, mods=~1,random=list(~1|factor(study)),method="ML",data=trans)
anova(m1,m2)
m1<-rma.mv(yi, vi, mods=~endpoints1, random=list(~1|factor(study)), method="ML",data=trans)
m2<-rma.mv(yi, vi, mods=~1,random=list(~1|factor(study)),method="ML",data=trans)
anova(m1,m2)
m1<-rma.mv(yi, vi, mods=~size_class, random=list(~1|factor(study)), method="ML",data=trans)
m2<-rma.mv(yi, vi, mods=~1,random=list(~1|factor(study)),method="ML",data=trans)
anova(m1,m2)
m1<-rma.mv(yi, vi, mods=~concentration_class, random=list(~1|factor(study)),
method="ML",data=trans)
m2<-rma.mv(yi, vi, mods=~1,random=list(~1|factor(study)),method="ML",data=trans)
anova(m1,m2)
m1<-rma.mv(yi, vi, mods=~system, random=list(~1|factor(study)), method="ML",data=trans)
m2<-rma.mv(yi, vi, mods=~1,random=list(~1|factor(study)),method="ML",data=trans)
anova(m1,m2)
m1<-rma.mv(yi, vi, mods=~cell.line..L..primary.cells..P., random=list(~1|factor(study)),
method="ML",data=trans)
m2<-rma.mv(yi, vi, mods=~1,random=list(~1|factor(study)),method="ML",data=trans)
anova(m1,m2)
m1<-rma.mv(yi, vi, mods=~exposure_type, random=list(~1|factor(study)),
method="ML",data=trans)
m2<-rma.mv(yi, vi, mods=~1,random=list(~1|factor(study)),method="ML",data=trans)
anova(m1,m2)
m1<-rma.mv(yi, vi, mods=~polymer_type, random=list(~1|factor(study)),
method="ML",data=trans)
m2<-rma.mv(yi, vi, mods=~1,random=list(~1|factor(study)),method="ML",data=trans)
anova(m1,m2)
m1<-rma.mv(yi, vi, mods=~exposure_time_class, random=list(~1|factor(study)),
method="ML",data=trans)
m2<-rma.mv(yi, vi, mods=~1,random=list(~1|factor(study)),method="ML",data=trans)
anova(m1,m2)
m1<-rma.mv(yi, vi, mods=~gender, random=list(~1|factor(study)), method="ML",data=trans)
m2<-rma.mv(yi, vi, mods=~1,random=list(~1|factor(study)),method="ML",data=trans)
anova(m1,m2)

# Calculate multivariate mixed-effects models and subgroup analysis
rmod.mix<-rma.mv(yi~endpoints_type-1,vi,random=list(~1|factor(study)), method="REML",
data=trans)
rmod.mix
```

```
rmod.mix2<-rma.mv(yi~endpoints_type-1,vi,random=list(~endpoints_type|factor(study)),
method="REML", data=trans)
rmod.mix2
rmod.mix3<-rma.mv(yi~endpoints1-1,vi,random=list(~endpoints1|factor(title)), method="REML",
data=trans)
rmod.mix3

rmod.mix<-rma.mv(yi~size_class-1,vi,random=list(~1|study), method="REML", data=trans)
rmod.mix
rmod.mix<-rma.mv(yi~concentration_class-1,vi,random=list(~1|study), method="REML",
data=trans)
rmod.mix
rmod.mix<-rma.mv(yi~exposure_time_class-1,vi,random=list(~1|study), method="REML",
data=trans)
rmod.mix
rmod.mix<-rma.mv(yi~lifestage_class-1,vi,random=list(~1|study), method="REML", data=trans)
rmod.mix
rmod.mix<-rma.mv(yi~exposure_type-1,vi,random=list(~1|study), method="REML", data=trans)
rmod.mix
rmod.mix<-rma.mv(yi~gender-1,vi,random=list(~1|study), method="REML", data=trans)
rmod.mix


#Calculate correlation of multivariate
rmod.mix2<-rma.mv(yi~0+paste(endpoints_detail,size_class),vi,random=list(~1|factor(study),~1|s
pecies),method="REML", data=trans,subset = rep>=3)
rmod.mix2
rmod.mix2<-rma.mv(yi~0+paste(endpoints,size_class),vi,random=list(~1|factor(study),~1|species)
,method="REML", data=trans,subset = rep>=3)
rmod.mix2
rmod.mix2<-rma.mv(yi~0+paste(endpoints_detail,concentration_class),vi,random=list(~1|factor(s
tudy),~1|species),method="REML", data=trans,subset = rep>=3)
rmod.mix2
rmod.mix2<-rma.mv(yi~0+paste(concentration_class,size_class),vi,random=list(~1|factor(study),
~1|species),method="REML", data=trans,subset = rep>=3)
rmod.mix2
rmod.mix2<-rma.mv(yi~0+paste(endpoints,concentration_class),vi,random=list(~1|factor(study),~
1|species),method="REML", data=trans,subset = rep>=3)
rmod.mix2


#Forest plot
library(forestplot)
library(dplyr)
```

```
rs_forest2 <- read.csv("forest_data.csv",header = TRUE)
rs_forest2 <- tibble::tibble(rs_forest2)
options(repr.plot.width = 20, repr.plot.height = 14)
rs_forest2 |>
    forestplot(labeltext = rs_forest2[,c(1:3)],
    is.summary=c(TRUE,FALSE,TRUE,FALSE,FALSE,FALSE,FALSE,FALSE,FALSE,TRUE,
                 FALSE,FALSE,FALSE,FALSE,TRUE,FALSE,FALSE,FALSE,FALSE,TRUE,
                 FALSE,FALSE,FALSE,FALSE,FALSE,TRUE,FALSE,FALSE,FALSE,FALSE,
                 FALSE,TRUE,FALSE,FALSE,FALSE,TRUE,FALSE,FALSE,TRUE,FALSE,
                 FALSE),
                    boxsize = c(0.4, 0.4, 0.8, 0.4, 0.8, 0.4, 0.4, 0.4, 0.4, 0.4, 0.4, 0.8,
                                0.4, 0.4, 0.4, 0.4, 0.8, 0.4, 0.4, 0.4, 0.4, 0.8,
                                0.4, 0.4, 0.4, 0.4, 0.4, 0.8, 0.4, 0.4, 0.4, 0.4,
                                0.4, 0.8, 0.4, 0.4, 0.4, 0.8, 0.4, 0.4, 0.8, 0.4,
                                0.4),
                    lineheight = unit(8,'mm'),
                    line.margin= unit(8,'mm'),
                    colgap = unit(1,'mm'),
                    zero = 0,
                    lwd.zero = 4,
                    lwd.ci = 2,
                    xlab="Effect Size",
                    lwd.xaxis=2,
                    col=fpColors(box='#CD853F', summary="#006400",lines = '#8B4513',zero =
'grey'),
                    txt_gp=fpTxtGp(label=gpar(cex=1.25),
                                   ticks=gpar(cex=1.3),
                                   xlab=gpar(cex=1.8),
                                   title=gpar(cex=1.1)),
                    lty.ci = "solid",
                    graph.pos = 2) |>

       fp_add_lines(h_3 = gpar(lty = 1)) |>
       fp_decorate_graph(grid = TRUE) |>
       fp_set_zebra_style("#F5F9F9")|>
       fp_add_header(Toxicity.metrics = c("","Toxicity"),
                     No..of.cases.No..of.studies. = c("No. of cases","(No. of studies)"),
                     OR = c("OR","(95%CI)"))



#dose-response relationship of NPs with different features in different conditions
# load packages
library(dplyr)
```

```
library(metafor)
library(RColorBrewer)
library(svglite)

col_vector<-c('#0084DE' , '#ffc000', '#228a22', '#e0220e', '#7f7f7f', '#d600d6', '#e97a27','#4700b0',
'#00d09a','#9a6324' ,
                        '#008080', '#bcf60c', '#ffd8b1', '#fffac8', '#800000', '#e6beff','#aaffc3' , '#fabebe',
'#f032e6', '#808080', '#ffffff', '#000000')

palette(col_vector)

df = read.csv("3.11micedata_model.csv",header =T)

df$vi[df$vi == 0] <- 0.00001
df$sd1[df$sd1 == 0] <- 0.00001
df$invse <- 1/df$vi
df$log.invse <- log(df$invse)

#system
df$system <- factor(df$system)
polyplot_system <- function(d, p) {
   d = as.data.frame(d)
   model = rma.mv(yi, vi, mods = ~log(concentration.µg.L.), random = list(~1|study/Number),
                          data = d[d$system == p,])
   minx = min(log(d$concentration.µg.L.[which(d$system == p)]), na.rm = T)
   maxx = max(log(d$concentration.µg.L.[which(d$system == p)]), na.rm = T)
   lines(x = c(minx,maxx), y = c(model$b[1]+model$b[2]*minx, model$b[1]+model$b[2]*maxx),
            col = as.numeric(d$system[d$system == p]), lwd = 2)
}

levels(df$system)
svglite("Plots/regplot_system2.svg", height = 5,
            width = 5, pointsize = 20)
par(mar=c(4,4,4,8), xpd = TRUE, mfrow = c(1,1), las =1, bty = "l",
      cex = 0.7)
palette(col_vector[c(1:9)])
plot(yi ~ log(concentration.µg.L.), data = df, pch = 21,
        col = adjustcolor(as.numeric(df$system), alpha.f = 0.05),
        bg = adjustcolor(as.numeric(df$system), alpha.f = 0.05), cex = log.invse/7,
        ylab = "", xlab = "", xlim = c(-3, 17), ylim = c(-1,2))
mtext("ln (Risk Ratio)", side = 2, line = 2.5, cex = 0.7, las = 3)
mtext("ln (Concentration in µg per L)", side = 1, line = 2, cex = 0.7)
lines(c(-3,17),y= c(0,0), col = adjustcolor("grey60", alpha.f= 0.9), lwd = 1.5, lty = 2)
text(x = 15, y = -0.2,"No effect", col = adjustcolor("grey60", alpha.f= 0.9), cex = 0.8)
```

```
lapply(levels(df$system), polyplot_system, d = df)
legend(18,1.5, levels(df$system), fill = c(1:9), cex = 0.8, text.font = 3, bty = "n")
dev.off()

#celltype
polyplot_celltype <- function(d, p) {
  d = as.data.frame(d)
  model = rma.mv(yi, vi, mods = ~log(concentration.µg.L.), random = list(~1|study/Number),
                 data = d[d$cell.line..L..primary.cells..P. == p,])
  minx = min(log(d$concentration.µg.L.[which(d$cell.line..L..primary.cells..P. == p)]), na.rm =
T)
  maxx = max(log(d$concentration.µg.L.[which(d$cell.line..L..primary.cells..P. == p)]), na.rm =
T)
  lines(x = c(minx,maxx), y = c(model$b[1]+model$b[2]*minx, model$b[1]+model$b[2]*maxx),
        col = as.numeric(d$cell.line..L..primary.cells..P.[d$cell.line..L..primary.cells..P. == p]),
lwd = 2)
}
df$cell.line..L..primary.cells..P. <- factor(df$cell.line..L..primary.cells..P.)
levels(df$cell.line..L..primary.cells..P.)
svglite("Plots/regplot_celltype2.svg", height = 5,
        width = 5, pointsize = 20)
par(mar=c(4,4,4,8), xpd = TRUE, mfrow = c(1,1), las =1, bty = "l",
    cex = 0.7)
palette(col_vector[c(1:9)])
plot(yi ~ log(concentration.µg.L.), data = df, pch = 21,
     col = adjustcolor(as.numeric(df$cell.line..L..primary.cells..P.), alpha.f = 0.1),
     bg = adjustcolor(as.numeric(df$cell.line..L..primary.cells..P.), alpha.f = 0.1), cex =
log.invse/7,
     ylab = "", xlab = "", xlim = c(-3, 17), ylim = c(-1,2))
mtext("ln (Risk Ratio)", side = 2, line = 2.5, cex = 0.7, las = 3)
mtext("ln (Concentration in µg per L)", side = 1, line = 2, cex = 0.7)
lines(c(-3,17),y= c(0,0), col = adjustcolor("grey60", alpha.f= 0.9), lwd = 1.5, lty = 2)
text(x = 15, y = -0.2,"No effect", col = adjustcolor("grey60", alpha.f= 0.9), cex = 0.8)
lapply(levels(df$cell.line..L..primary.cells..P.), polyplot_celltype, d = df)
legend(18,1,c("in vitro", "in vivo"), fill = c(1:2), cex = 0.8, text.font = 3, bty = "n")
dev.off()

#exposure type
polyplot_exposuretype <- function(d, p) {
  d = as.data.frame(d)
  model = rma.mv(yi, vi, mods = ~log(concentration.µg.L.), random = list(~1|study/Number),
                 data = d[d$exposure_type == p,])
  minx = min(log(d$concentration.µg.L.[which(d$exposure_type == p)]), na.rm = T)
  maxx = max(log(d$concentration.µg.L.[which(d$exposure_type == p)]), na.rm = T)
```

```
   lines(x = c(minx,maxx), y = c(model$b[1]+model$b[2]*minx, model$b[1]+model$b[2]*maxx),
         col = as.numeric(d$exposure_type[d$exposure_type == p]), lwd = 2)
}
df$exposure_type <- factor(df$exposure_type)
levels(df$exposure_type)
svglite("Plots/regplot_exposuretype2.svg", height = 5,
         width = 5, pointsize = 20)
par(mar=c(4,4,4,8), xpd = TRUE, mfrow = c(1,1), las =1, bty = "l",
    cex = 0.7)
palette(col_vector[c(1:9)])
plot(yi ~ log(concentration.µg.L.), data = df, pch = 21,
      col = adjustcolor(as.numeric(df$exposure_type), alpha.f = 0.05),
      bg = adjustcolor(as.numeric(df$exposure_type), alpha.f = 0.05), cex = log.invse/7,
      ylab = "", xlab = "", xlim = c(-3, 17), ylim = c(-1,2))
mtext("ln (Risk Ratio)", side = 2, line = 2.5, cex = 0.7, las = 3)
mtext("ln (Concentration in µg per L)", side = 1, line = 2, cex = 0.7)
lines(c(-3,17),y= c(0,0), col = adjustcolor("grey60", alpha.f= 0.9), lwd = 1.5, lty = 2)
text(x = 15, y = -0.2,"No effect", col = adjustcolor("grey60", alpha.f= 0.9), cex = 0.8)
lapply(levels(df$exposure_type), polyplot_exposuretype, d = df)
legend(18,1, levels(df$exposure_type), fill = c(1:4), cex = 0.8, text.font = 3, bty = "n")
dev.off()

#size class
polyplot_size <- function(d, p) {
   d = as.data.frame(d)
   model = rma.mv(yi, vi, mods = ~log(concentration.µg.L.), random = list(~1|study/Number),
                  data = d[d$size_class == p,])
   minx = min(log(d$concentration.µg.L.[which(d$size_class == p)]), na.rm = T)
   maxx = max(log(d$concentration.µg.L.[which(d$size_class == p)]), na.rm = T)
   lines(x = c(minx,maxx), y = c(model$b[1]+model$b[2]*minx, model$b[1]+model$b[2]*maxx),
         col = as.numeric(d$size_class[d$size_class == p]), lwd = 2)
}
df$size_class <- factor(df$size_class)
svglite("Plots/regplot_npsize2.svg", height = 5,
         width = 5, pointsize = 20)

opar <- par(mar=c(4,4,4,8), xpd = TRUE, mfrow = c(1,1), bty = "l", las = 1, cex = 0.7)
palette(col_vector[c(11,5)])
m.size <- rma.mv(yi, vi, mods = ~log(size_of_plastic..nm.), random = list(~1|study/Number), data
= df)
m.size
# plot regression plot
regplot(m.size,bty = "l", ylab = "", xlab = "    ", psize = df$log.invse/7,
         col = adjustcolor(as.numeric(as.factor(df$size_class)), alpha.f = 0.2),
```

```
            bg = adjustcolor(as.numeric(as.factor(df$size_class)), alpha.f = 0.1),
            lcol = "grey60", ylim = c(-1,2))
regplot(m.size,bty = "l", ylab = "", xlab = "    ", psize = df$log.invse/7,
            col = adjustcolor(as.numeric(as.factor(df$size_class)), alpha.f = 0.2),
            bg = adjustcolor(as.numeric(as.factor(df$size_class)), alpha.f = 0.1),
            lcol = "grey60", ylim = c(-1,2))
mtext("ln (Risk Ratio)", side = 2, line = 2.5, cex = 0.7, las = 3)
mtext("ln (NP size in nm)", side = 1, line = 2, cex = 0.7)
legend(5,1,    c("<50nm", "50-100nm"), fill = c(1:2), cex = 0.8, border = c(1:2), bty = "n")
dev.off()

#exposure time
polyplot_exptime <- function(d, p) {
   d = as.data.frame(d)
   model = rma.mv(yi, vi, mods = ~log(concentration.µg.L.), random = list(~1|study/Number),
                          data = d[d$exposure_time_class == p,])
   minx = min(log(d$concentration.µg.L.[which(d$exposure_time_class == p)]), na.rm = T)
   maxx = max(log(d$concentration.µg.L.[which(d$exposure_time_class == p)]), na.rm = T)
   lines(x = c(minx,maxx), y = c(model$b[1]+model$b[2]*minx, model$b[1]+model$b[2]*maxx),
            col = as.numeric(d$exposure_time_class[d$exposure_time_class == p]), lwd = 2)
}
df$exposure_time_class <- factor(df$exposure_time_class)
levels(df$exposure_time_class)
svglite("Plots/regplot_exptime2.svg", height = 5,
            width = 5, pointsize = 20)
par(mar=c(4,4,4,8), xpd = TRUE, mfrow = c(1,1), las =1, bty = "l",
      cex = 0.7)
palette(col_vector[c(1:9)])
plot(yi ~ log(concentration.µg.L.), data = df, pch = 21,
        col = adjustcolor(as.numeric(df$exposure_time_class), alpha.f = 0.1),
        bg = adjustcolor(as.numeric(df$exposure_time_class), alpha.f = 0.1), cex = log.invse/7,
        ylab = "", xlab = "", xlim = c(-3, 17), ylim = c(-1,2))
mtext("ln (Risk Ratio)", side = 2, line = 2.5, cex = 0.7, las = 3)
mtext("ln (Concentration in µg per L)", side = 1, line = 2, cex = 0.7)
lines(c(-3,17),y= c(0,0), col = adjustcolor("grey60", alpha.f= 0.9), lwd = 1.5, lty = 2)
text(x = 15, y = -0.2,"No effect", col = adjustcolor("grey60", alpha.f= 0.9), cex = 0.8)
lapply(levels(df$exposure_time_class), polyplot_exptime, d = df)
legend(18,1, c("< 24h", "1 ~ 15d", "15 ~ 30d", "30 ~ 45d", "> 45d"), fill = c(1:5), cex = 0.8,
text.font = 3, bty = "n")
dev.off()

#gender
polyplot_gender <- function(d, p) {
   d = as.data.frame(d)
```

```
  model = rma.mv(yi, vi, mods = ~log(concentration.µg.L.), random = list(~1|study/Number),
                 data = d[d$gender == p,])
  minx = min(log(d$concentration.µg.L.[which(d$gender == p)]), na.rm = T)
  maxx = max(log(d$concentration.µg.L.[which(d$gender == p)]), na.rm = T)
  lines(x = c(minx,maxx), y = c(model$b[1]+model$b[2]*minx, model$b[1]+model$b[2]*maxx),
        col = as.numeric(d$gender[d$gender == p]), lwd = 2)
}
df$gender <- factor(df$gender)
levels(df$gender)
svglite("Plots/regplot_gender2.svg", height = 5,
        width = 5, pointsize = 20)
par(mar=c(4,4,4,8), xpd = TRUE, mfrow = c(1,1), las =1, bty = "l",
    cex = 0.7)
palette(col_vector[c(1:9)])
plot(yi ~ log(concentration.µg.L.), data = df, pch = 21,
     col = adjustcolor(as.numeric(df$gender), alpha.f = 0.1),
     bg = adjustcolor(as.numeric(df$gender), alpha.f = 0.1), cex = log.invse/7,
     ylab = "", xlab = "", xlim = c(-3, 17), ylim = c(-1,2))
mtext("ln (Risk Ratio)", side = 2, line = 2.5, cex = 0.7, las = 3)
mtext("ln (Concentration in µg per L)", side = 1, line = 2, cex = 0.7)
lines(c(-3,17),y= c(0,0), col = adjustcolor("grey60", alpha.f= 0.9), lwd = 1.5, lty = 2)
text(x = 15, y = -0.2,"No effect", col = adjustcolor("grey60", alpha.f= 0.9), cex = 0.8)
lapply(levels(df$gender), polyplot_gender, d = df)
legend(18,1, levels(df$gender), fill = c(1:3), cex = 0.8, text.font = 3, bty = "n")
dev.off()

#NP type
polyplot_polytype <- function(d, p) {
  d = as.data.frame(d)
  model = rma.mv(yi, vi, mods = ~log(concentration.µg.L.), random = list(~1|study/Number),
                 data = d[d$polymer_type == p,])
  minx = min(log(d$concentration.µg.L.[which(d$polymer_type == p)]), na.rm = T)
  maxx = max(log(d$concentration.µg.L.[which(d$polymer_type == p)]), na.rm = T)
  lines(x = c(minx,maxx), y = c(model$b[1]+model$b[2]*minx, model$b[1]+model$b[2]*maxx),
        col = as.numeric(d$polymer_type[d$polymer_type == p]), lwd = 2)
}
df$polymer_type <- factor(df$polymer_type)
levels(df$polymer_type)
svglite("Plots/regplot_polytype2.svg", height = 5,
        width = 5, pointsize = 20)
par(mar=c(4,4,4,8), xpd = TRUE, mfrow = c(1,1), las =1, bty = "l",
    cex = 0.7)
palette(col_vector[c(1:9)])
plot(yi ~ log(concentration.µg.L.), data = df, pch = 21,
```

```
        col = adjustcolor(as.numeric(df$polymer_type), alpha.f = 0.1),
        bg = adjustcolor(as.numeric(df$polymer_type), alpha.f = 0.1), cex = log.invse/7,
        ylab = "", xlab = "", xlim = c(-3, 17), ylim = c(-1,2))
mtext("ln (Risk Ratio)", side = 2, line = 2.5, cex = 0.7, las = 3)
mtext("ln (Concentration in µg per L)", side = 1, line = 2, cex = 0.7)
lines(c(-3,17),y= c(0,0), col = adjustcolor("grey60", alpha.f= 0.9), lwd = 1.5, lty = 2)
text(x = 15, y = -0.2,"No effect", col = adjustcolor("grey60", alpha.f= 0.9), cex = 0.8)
m.Ps <- rma.mv(yi, vi, mods = ~log(concentration.µg.L.), random = list(~1|study/Number),
                        data = df[df$polymer_type == "PS",])
lines(x = c(0, 15), y = c(m.Ps$b[1]+m.Ps$b[2]*0 ,m.Ps$b[1]+m.Ps$b[2]*15),
        col = as.numeric(df$polymer_type[df$polymer_type == "PS"]), lwd = 2)
m.Psc <- rma.mv(yi, vi, mods = ~log(concentration.µg.L.), random = list(~1|study/Number),
                        data = df[df$polymer_type == "PS-COOH",])
lines(x = c(0, 15), y = c(m.Psc$b[1]+m.Psc$b[2]*0 ,m.Psc$b[1]+m.Psc$b[2]*15),
        col = as.numeric(df$polymer_type[df$polymer_type == "PS-COOH"]), lwd = 2)
m.Psn <- rma.mv(yi, vi, mods = ~log(concentration.µg.L.), random = list(~1|study/Number),
                        data = df[df$polymer_type == "PS-NH2",])
lines(x = c(10, 13), y = c(m.Psn$b[1]+m.Psn$b[2]*10 ,m.Psn$b[1]+m.Psn$b[2]*13),
        col = as.numeric(df$polymer_type[df$polymer_type == "PS-NH2"]), lwd = 2)
legend(18,1, c("PS", "PS-COOH", "PS-NH2"), fill = c(1:3), cex = 0.8, text.font = 3, bty = "n")
dev.off()
```

#RF&XGBoost

```
from sklearn.ensemble import RandomForestRegressor
from xgboost import XGBRegressor as XGBR
from sklearn.model_selection import cross_val_score,cross_val_predict
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import mean_squared_error
from mpl_toolkits.mplot3d import Axes3D
from sklearn.metrics import r2_score
from sklearn.metrics import mean_squared_error as MSE
from sklearn.metrics import mean_absolute_error as MAE
from sklearn.metrics import log_loss
import matplotlib
import matplotlib.pyplot as plt
import matplotlib.cm as cm
import shap
import seaborn as sns
import numpy as np
import pandas as pd
```

```
import os
import sklearn

data = pd.read_excel(r"ML_rawdata.xlsx")
feature_names = ['cell type', 'system', 'exposure type', 'polymer type', 'NP size',
                 'concentration', 'exposure time', 'gender', 'endpoints']
X = np.asarray(data.iloc[:,2:])
y = np.asarray(data.iloc[:,1])
X = pd.DataFrame(X, columns=feature_names)
Xtrain, Xtest, Ytrain, Ytest = train_test_split(X, y, test_size=0.2, random_state=42)


#RF
#Random_state
score_5cv_all = []
for i in range(0, 200, 1):
    rfc =RandomForestRegressor(random_state=i)
    score_5cv =cross_val_score(rfc, Xtrain, Ytrain, cv=5).mean()
    score_5cv_all.append(score_5cv)
    pass
score_max_5cv = max(score_5cv_all)
print("Best_5cv score:{}".format(score_max_5cv),
"random_5cv:{}".format(score_5cv_all.index(score_max_5cv)))
random_state_5cv = range(0, 200)[score_5cv_all.index(max(score_5cv_all))]
print(random_state_5cv)

score_5cv_all = []
for i in range(1, 150, 1):
    rfc = RandomForestRegressor(n_estimators=i
    , random_state=random_state_5cv)
    score_5cv = cross_val_score(rfc, Xtrain, Ytrain, cv=5).mean()
    score_5cv_all.append(score_5cv)
    pass
score_max_5cv = max(score_5cv_all)
print("Best_5cv
score:{}".format(score_max_5cv),"n_est_5cv:{}".format(score_5cv_all.index(score_max_5cv)))
n_est_5cv = range(1,400)[score_5cv_all.index(score_max_5cv)]
print(n_est_5cv)

score_5cv_all = []
for i in range(1, 100, 1):
    rfc = RandomForestRegressor(n_estimators=n_est_5cv ,random_state=random_state_5cv
    ,max_depth=i)
    score_5cv = cross_val_score(rfc, Xtrain, Ytrain, cv=5).mean()
    score_5cv_all.append(score_5cv)
```

```
        pass
score_max_5cv = max(score_5cv_all)
print("Best_5cv score:{}".format(score_max_5cv),
"max_depth_5cv:{}".format(score_5cv_all.index(score_max_5cv)))
max_depth_5cv = range(1,100)[score_5cv_all.index(score_max_5cv)]
print(max_depth_5cv)

rfc = RandomForestRegressor(n_estimators=98 ,random_state=199,max_depth=11)
CV_score = cross_val_score(rfc, Xtrain, Ytrain, cv=5).mean()
CV_predictions = cross_val_predict(rfc, Xtrain, Ytrain, cv=5)
rmse = np.sqrt(mean_squared_error(Ytrain,CV_predictions))
print("5cv:",CV_score)
print("rmse_5CV",rmse)
expvspred_5cv = {'Exp': Ytrain, 'Pred':CV_predictions}
pd.DataFrame(expvspred_5cv).to_excel('Random_forest_5fcv_predictions.xlsx')

rfc = RandomForestRegressor(n_estimators=98 ,random_state=199,max_depth=11)
regressor = rfc.fit(Xtrain, Ytrain)
test_predictions = regressor.predict(Xtest)
score_test = regressor.score(Xtest,Ytest)
rmse = np.sqrt(mean_squared_error(Ytest,test_predictions))
print("test:",score_test)
print("rmse_test",rmse)
expvspred_test = {'Exp':Ytest,'Pred':test_predictions}
pd.DataFrame(expvspred_test).to_excel('RF_test_predictions.xlsx')

def result_figure(y_test_train, pred_train, y_test_test, pred_test, x_label, y_label, data_neme,
model_name, path):
        os.makedirs("{}/figure".format(path), exist_ok=True)
        os.makedirs("{}/csv".format(path), exist_ok=True)
        r2_train = r2_score(y_test_train, pred_train)
        r2_test = r2_score(y_test_test, pred_test)

        mse_train = MSE(y_test_train, pred_train)
        mse_test = MSE(y_test_test, pred_test)
        mae_train = MAE(y_test_train, pred_train)
        mae_test = MAE(y_test_test, pred_test)



pd.concat([pd.DataFrame(pred_train,columns=["prediction"]) ,pd.DataFrame(y_test_train,column
s=['true'])
        ,pd.DataFrame([r2_train],columns=["r^2"]) ,pd.DataFrame([np.sqrt(mse_train)],columns=['R
MSE']) ,pd.DataFrame([mae_train],columns=['MAE'])],axis
```

```
=1).to_csv("{}/csv/{}_{}_kf.csv".format(path, data_neme, model_name), index=None)


pd.concat([pd.DataFrame(pred_test,columns=["prediction"]) ,pd.DataFrame(y_test_test,columns=['true'])
,pd.DataFrame([r2_test],columns=["r^2"]) ,pd.DataFrame([np.sqrt(mse_test)],columns=['RMSE']) ,pd.DataFrame([mae_test],columns=['MAE'])],axis
=1).to_csv("{}/csv/{}_{}_test.csv".format(path, data_neme, model_name), index=None)

plt.figure(figsize=[20,20], dpi=300)
matplotlib.rcParams['font.family'] = "Arial"
xmax = int(max(list(pred_train)+list(y_test_train)+list(pred_test)+list(y_test_test)))+1
xmin = int(min(list(pred_train)+list(y_test_train)+list(pred_test)+list(y_test_test)))

plt.scatter(y_test_train, pred_train,c="royalblue", s=200 ,edgecolor="k",alpha=0.7, label="Train Set")
plt.scatter(y_test_test, pred_test,c="tomato", s=200 ,edgecolor="k",alpha=0.7, label="Test Set")

line1 = plt.scatter(-4, -4,c="royalblue", s=1200 ,edgecolor="k",alpha=0.9)
line2 = plt.scatter(-4, -4,c="tomato", s=1200 ,edgecolor="k",alpha=0.9)

plt.plot(range(xmin,xmax+1),range(xmin,xmax+1),c='k',linewidth=3,linestyle="--", alpha=0.9)

plt.xlabel(x_label,fontsize=50)
plt.ylabel(y_label,fontsize=50, fontdict={"family":"Arial"})
labels = [""]+[str(i) for i in range(xmin+1,xmax+1)]
plt.yticks(ticks=range(xmin,xmax+1), labels=labels, fontsize=40)
plt.xticks(ticks=range(xmin,xmax+1), labels=range(xmin,xmax+1), fontsize=40)
plt.xlim(xmin,xmax)
plt.ylim(xmin,xmax)
plt.legend(fontsize=50, handles=[line1, line2], labels=['Train Set','Test Set'], framealpha=0)
plt.savefig("{}/figure/{}_{}.png".format(path, data_neme, model_name))
plt.show()

result_figure(Ytrain, CV_predictions, Ytest, test_predictions, "Actual Values", "Predicted Values", "Presentation_data", "RF", "./results_RF")


#XGBoost
#Random_state
score_5cv_all = []
for i in range(0, 200, 1):
```

```
        rfc =XGBR(random_state=i)
        score_5cv =cross_val_score(rfc, Xtrain, Ytrain, cv=5).mean()
        score_5cv_all.append(score_5cv)
        pass
score_max_5cv = max(score_5cv_all)
print("Best_5cv score:{}".format(score_max_5cv),
"random_5cv:{}".format(score_5cv_all.index(score_max_5cv)))
random_state_5cv = range(0, 200)[score_5cv_all.index(max(score_5cv_all))]
print(random_state_5cv)

#learning_rate
score_5cv_all = []
for i in np.arange(0.01, 0.5, 0.01):
        rfc = XGBR(learning_rate=i ,random_state=random_state_5cv)
        score_5cv = cross_val_score(rfc, Xtrain, Ytrain, cv=5).mean()
        score_5cv_all.append(score_5cv)
        pass
score_max_5cv = max(score_5cv_all)
print("Best_5cv
score:{}".format(score_max_5cv),"lr_5cv:{}".format(score_5cv_all.index(score_max_5cv)))
n_lr_5cv = np.arange(0.01,0.5,0.01)[score_5cv_all.index(score_max_5cv)]
print(n_lr_5cv)

#N_estimators
score_5cv_all = []
for i in range(1, 400, 1):
        rfc = XGBR(n_estimators=i ,learning_rate=n_lr_5cv,random_state=random_state_5cv)
        score_5cv = cross_val_score(rfc, Xtrain, Ytrain, cv=5).mean()
        score_5cv_all.append(score_5cv)
        pass
score_max_5cv = max(score_5cv_all)

print("Best_5cv score:{}".format(score_max_5cv),
"n_est_5cv:{}".format(score_5cv_all.index(score_max_5cv)))
n_est_5cv = range(1,400)[score_5cv_all.index(score_max_5cv)]
print(n_est_5cv)

#Max_depth
score_5cv_all = []
for i in range(1, 300, 1):
        rfc =
XGBR(n_estimators=n_est_5cv ,learning_rate=n_lr_5cv,random_state=random_state_5cv,max_de
pth=i)
        score_5cv = cross_val_score(rfc, Xtrain, Ytrain, cv=5).mean()
```

```
        CV_predictions = cross_val_predict(rfc, Xtrain, Ytrain, cv=5)
        score_5cv_all.append(score_5cv)
        pass
score_max_5cv = max(score_5cv_all)
print("Best_5cv score:{}".format(score_max_5cv),
"max_depth_5cv:{}".format(score_5cv_all.index(score_max_5cv)))
max_depth_5cv = range(1,300)[score_5cv_all.index(score_max_5cv)]
print(max_depth_5cv )

#Gamma
score_5cv_all = []
for i in np.arange(0,5,0.05):
        rfc =
XGBR(n_estimators=n_est_5cv ,learning_rate=n_lr_5cv,random_state=random_state_5cv ,max_d
epth=max_depth_5cv ,gamma=i)
        score_5cv = cross_val_score(rfc, Xtrain, Ytrain, cv=5).mean()
        CV_predictions = cross_val_predict(rfc, Xtrain, Ytrain, cv=5)
        score_5cv_all.append(score_5cv)
        pass

score_max_5cv = max(score_5cv_all)
print("Best_5cv
score:{}".format(score_max_5cv),"gamma_5cv:{}".format(score_5cv_all.index(score_max_5cv))
)
max_gamma_5cv = np.arange(0,5,0.05)[score_5cv_all.index(score_max_5cv)]
print(max_gamma_5cv)

#Alpha
score_5cv_all = []
for i in np.arange(0,5,0.05):
        rfc = XGBR(n_estimators=n_est_5cv,
learning_rate=n_lr_5cv,random_state=random_state_5cv ,max_depth=max_depth_5cv ,gamma=m
ax_gamma_5cv ,alpha=i)
        score_5cv = cross_val_score(rfc, Xtrain, Ytrain, cv=5).mean()
        CV_predictions = cross_val_predict(rfc, Xtrain, Ytrain, cv=5)
        score_5cv_all.append(score_5cv)
        pass
score_max_5cv = max(score_5cv_all)
print("Best_5cv
score:{}".format(score_max_5cv),"alpha_5cv:{}".format(score_5cv_all.index(score_max_5cv)))
max_alpha_5cv = np.arange(0,5,0.05)[score_5cv_all.index(score_max_5cv)]
print(max_alpha_5cv)

XGB =
```

```
XGBR(learning_rate=0.09999999999999999 ,n_estimators=75,random_state=0 ,max_depth=6 ,ga
mma =0 ,alpha = 0)
CV_score = cross_val_score(XGB, Xtrain, Ytrain, cv=5).mean()
CV_predictions = cross_val_predict(XGB, Xtrain, Ytrain, cv=5)
rmse = np.sqrt(mean_squared_error(Ytrain,CV_predictions))
print("5cv:",CV_score)
print("RMSE_5CV",rmse)
expvspred_5cv = {'Exp': Ytrain, 'Pred':CV_predictions}
pd.DataFrame(expvspred_5cv).to_excel('./XGBoost_5fcv_predictions.xlsx')

'''
Test set validation
'''
XGB =
XGBR(learning_rate=0.09999999999999999 ,n_estimators=75,random_state=0 ,max_depth=6 ,ga
mma =0 ,alpha = 0)
XGBOOST = XGB.fit(Xtrain, Ytrain)
test_predictions = XGBOOST.predict(Xtest)
score_test = XGBOOST.score(Xtest,Ytest)
rmse = np.sqrt(mean_squared_error(Ytest,test_predictions))
print("test:",score_test)
print("rmse_test",rmse)
expvspred_test = {'Exp':Ytest,'Pred':test_predictions}
pd.DataFrame(expvspred_test).to_excel('./XGB_test_predictions.xlsx')
result_figure(Ytrain, CV_predictions, Ytest, test_predictions, "Actual Values", "Predicted Values",
"Presentation_data", "XGBoost", "./results_xgb")
```