ModelReport

Fiona Zhu

23/03/2020

Load the packages

customize theme

```
theme_new <- theme_bw() +
  theme(panel.border = element_blank(),
        panel.grid.major = element_blank(),
        panel.grid.minor = element_blank(),
        axis.line = element_line(colour = "black"),
        strip.background = element_rect(color = "white", fill = "white"),
        panel.grid = element_blank())
```

```
options(mc.cores = parallel::detectCores())
rstan_options (auto_write=TRUE)
# flag for saving figures
saveFigure = TRUE
# flag for generating CSV
generateSCV = TRUE
# flag for running rstan model and saving the results
runModels = FALSE
# path of model result
rstanmodelPath = 'RSTANMODELS'
modelResultPath = paste0(rstanmodelPath, '/Version2_2')
```

load experimental data

RStan Models

Definition of the function to merge rstan result data

To further uncover the underlying structure of the prior, we hypothesize that in addition to two local priors, there is a general global prior.Participants may combine both local and global priors for the final reproduction. There are multiple possibilities for integrating those local and global priors with the sensory inputs. For examples, the sensory input could first integrate with the local prior, and then integrate with the global prior (a hierarchical local-global model, see Figure 7B in D1 proposal).

Our Hypothesis

- H1: Short and long is independent
- H3: A hierarchical local-global model In H3, global and local priors are integrated first. That means local prior is integrated with sensory input firstly, then global prior integrates with sensory input.

The sensory input (D_s) first integrates with the local prior (P_L) to a posterior (D_L), which further integrates with the global prior (P_G) to generate a final posterior for reproduction (D_r).

- H4: Global prior (the dual integration model)

integration of local priors firstly, then integration of the global prior

Both local and global priors independently integrate with the sensory inputs to generate two posteriors (D_L) and (D_G), the latter two are combined together for reproduction (D_r).

Merg different version of model results

```
# this R script is designed for the analysis of the result of Rstan model.
mergeData <- function(cvsfiles, filename, versionlist){
  merge.data.all = {}
  for(version in versionlist){
    merge.data = {}
    dataDir <- paste0(rstanmodelPath, "/", version )
    merge.data <- read.csv(file.path(dataDir, cvsfiles[1]),  header=T)
    merge.data$model = modellist[1]
    if (length(cvsfiles) >= 2) {
      for (i in 2:length(cvsfiles)){
        new.data = read.csv(file.path(dataDir, cvsfiles[i]),  header=T)
        new.data$model = modellist[i]
        merge.data = rbind(merge.data,new.data)
      }
    }
    merge.data$version=version
    merge.data.all = rbind(merge.data.all, merge.data)

  }
  savedataDir <- file.path(paste0(dataDir, "/AllDat_", filename, ".csv"))
  write.csv(file=savedataDir, merge.data.all)
}
```

H2: A hierarchical local-global model

```
stancodeH2 <- 'data {
int<lower=0> n_s;  //number of the short group baseline data points
int<lower=0> n_l;  //number of the long group baseline data points
int<lower=0> n_mix;  //number of the mix group baseline data points
real<lower=0> Y_s[n_s];   //measured reproductive duration (short group)
real<lower=0> X_s[n_s];   //stimulus duration (short group)
real<lower=0> Y_l[n_l];   //measured reproductive duration (long group)
real<lower=0> X_l[n_l];   //stimulus duration (long group)
real<lower=0> X_mix[n_mix];   //stimulus duration (mix group)
real<lower=0> Y_mix[n_mix];   //measured reproductive duration (mixed)
real xmean[3];  // mean of the target duration in each group
}

parameters {
//hyperparameters
real<lower=0, upper =1> p_wf_s;   //Weber Fraction of local prior
real<lower=0, upper =1> wf_s;     //Weber Fraction of sensory noise
real<lower=0, upper =1> p_wf_l;   //Weber Fraction of local prior
real<lower=0, upper =1> wf_l;     //Weber Fraction of sensory noise
real<lower=0, upper =1> p_wf_m;   //Weber Fraction of local prior
real<lower=0, upper =1> wf_m;     //Weber Fraction of sensory noise
vector[n_s] mu_s;   // mean of internal prior of short group
vector[n_l] mu_l;   // mean of internal prior of long group
vector[n_mix] mu_m;   // mean of global prior of mix group
vector[n_mix] mu_m_s;   // mean of local prior of mix group
vector[n_mix] mu_m_l;   // mean of local prior of mix group
```

```
real<lower=0, upper=9> sig_m_square;   //square of sigma of distribution of motor noise

}


transformed parameters {
  real<lower=0> sig_mix_square = p_wf_m^2 * xmean[3]^2;  //square of sigma of distribution of global pri
}

model {
real  w_s[n_s];   //  weight of stimuli in short group
real  w_l[n_l];   //  weight of stimuli in short group
real p_mix[n_mix];
real p_sig_mix[n_mix];
real w_mix[n_mix];
real d_mix_hat[n_mix];
real var_mix[n_mix];
real g_w_mix[n_mix];
real y_mix_mean[n_mix];

//hyperpriors
mu_s ~ normal(xmean[1], p_wf_s^2 * xmean[1]^2);  // mean prior of short group
mu_l ~ normal(xmean[2], p_wf_l^2 * xmean[2]^2);   // mean prior of long group
mu_m~ normal(xmean[3], sig_mix_square);   // mean prior of mix group (mean of global prior)
mu_m_s~ normal(xmean[1], p_wf_s^2 * xmean[1]^2);   // mean prior of mix group (mean of global prior)
mu_m_l~ normal(xmean[2], p_wf_l^2 * xmean[2]^2);   // mean prior of mix group (mean of global prior)



//short groups
for (i in 1:n_s)
{
  w_s[i] = (mu_s[i]^2 * p_wf_s^2)/(mu_s[i]^2 * p_wf_s^2+ X_s[i]^2*wf_s^2);    // weight of current stimu
  Y_s[i] ~ normal(mu_s[i] *  w_s[i]+ (1- w_s[i])* X_s[i], sig_m_square + (mu_s[i]^2 * p_wf_s^2 * X_s[i]
}

//long groups
for (i in 1:n_l)
{
    w_l[i] = (mu_l[i]^2 * p_wf_l^2)/(mu_l[i]^2 * p_wf_l^2+ X_l[i]^2*wf_l^2);  // weight of current stimu
    Y_l[i] ~ normal(mu_l[i] *  w_l[i]+ (1- w_l[i])* X_l[i],
    sig_m_square + (mu_l[i]^2 * p_wf_l^2 * X_l[i]^2*wf_l^2) / mu_l[i]^2 * p_wf_l^2 + X_l[i]^2*wf_l^2);

}

//mix groups
for (m in 1:n_mix) {
      // H3 part1 integration of local priors firstly
      if (X_mix[m] < 1){
        p_mix[m] = mu_m_s[m];
        p_sig_mix[m] = p_mix[m] * p_wf_s;
      }else{
        p_mix[m] = mu_m_l[m];
```

```
        p_sig_mix[m] = p_mix[m] * p_wf_l;
      }

      w_mix[m] = p_sig_mix[m]^2 / (p_sig_mix[m]^2 +  (X_mix[m]*wf_m)^2);
      d_mix_hat[m] = w_mix[m] * X_mix[m] + (1-w_mix[m])*p_mix[m];

      // H3 part2  posterior variances
      var_mix[m] = p_sig_mix[m]^2 * (X_mix[m]*wf_m)^2 / (p_sig_mix[m]^2 + (X_mix[m]*wf_m)^2);

      // H3 part3 integration of the global prior

      g_w_mix[m] = sig_mix_square / (sig_mix_square+ var_mix[m]);
      y_mix_mean[m] = g_w_mix[m] * d_mix_hat[m]  + (1-g_w_mix[m]) * xmean[3];
      Y_mix[m] ~ normal(y_mix_mean[m],  sig_m_square + (sig_mix_square *  var_mix[m]) / (sig_mix_square+

  }
}

generated quantities {
  vector[n_mix] ynew_mix;

  vector[n_mix] p_mix_new;   // mean of internal prior of mix group
  vector[n_mix] p_sig_mix_new;
  vector[n_mix] w1_new;
  vector[n_mix] d_mix_hat_new;
  vector[n_mix] var_mix_new;
  vector[n_mix] w2_new;
  vector[n_mix] y_mix_mean_new;


  for (m in 1:n_mix)   //prediction of mix group
  {
    if(X_mix[m] >= 1) {
     p_mix_new[m] = mu_m_s[m];
     p_sig_mix_new[m] = p_mix_new[m] * p_wf_s;
    }
   else{
     p_mix_new[m] = mu_m_l[m];
     p_sig_mix_new[m] = p_mix_new[m] * p_wf_l;
   }

     w1_new[m] = p_sig_mix_new[m]^2 / (p_sig_mix_new[m]^2 +  (X_mix[m]*wf_m)^2);
     d_mix_hat_new[m] = w1_new[m] * X_mix[m] + (1-w1_new[m])*p_mix_new[m];

      // H3 part2  posterior variances
      var_mix_new[m] = p_sig_mix_new[m]^2 * (X_mix[m]*wf_m)^2 / (p_sig_mix_new[m]^2 + (X_mix[m]*wf_m)^2

      // H3 part3 integration of the global prior

      w2_new[m] = sig_mix_square / (sig_mix_square+ var_mix_new[m]);
      y_mix_mean_new[m] = w2_new[m] * d_mix_hat_new[m]  + (1-w2_new[m]) * xmean[3];
      ynew_mix[m] =  normal_rng(y_mix_mean_new[m],  sig_m_square + (sig_mix_square *  var_mix_new[m]) /
```

```
  }

}
'



  # compile models
if (runModels == TRUE){
stanmodelH2 <- stan_model(model_code = stancodeH2, model_name="stanmodelH2")
  }
```

H3: Global prior (the dual integration model)

```
stancodeH3 <- 'data {
int<lower=0> n_s;  //number of the short group baseline data points
int<lower=0> n_l;  //number of the long group baseline data points
int<lower=0> n_mix;  //number of the mix group baseline data points
real<lower=0> Y_s[n_s];   //measured reproductive duration (short group)
real<lower=0> X_s[n_s];   //stimulus duration (short group)
real<lower=0> Y_l[n_l];   //measured reproductive duration (long group)
real<lower=0> X_l[n_l];   //stimulus duration (long group)
real<lower=0> X_mix[n_mix];   //stimulus duration (mix group)
real<lower=0> Y_mix[n_mix];   //measured reproductive duration (mixed)
real xmean[3];  // mean of the target duration in each group
}

parameters {
//hyperparameters
real<lower=0, upper =1> p_wf_s;   //Weber Fraction of local prior
real<lower=0, upper =1> wf_s;     //Weber Fraction of sensory noise
real<lower=0, upper =1> p_wf_l;   //Weber Fraction of local prior
real<lower=0, upper =1> wf_l;     //Weber Fraction of sensory noise
real<lower=0, upper =1> p_wf_m;   //Weber Fraction of local prior
real<lower=0, upper =1> wf_m;     //Weber Fraction of sensory noise
real<lower=0, upper =1> g_wf;    //Weber Fraction of global prior
vector[n_s] mu_s;   // mean of internal prior of short group
vector[n_l] mu_l;   // mean of internal prior of long group
vector[n_mix] mu_m;   // mean of global prior of mix group
vector[n_mix] mu_m_s;   // mean of local prior of mix group
vector[n_mix] mu_m_l;   // mean of local prior of mix group

real<lower=0, upper=9> sig_m_square;   //square of sigma of distribution of motor noise

}


transformed parameters {
  real<lower=0> sig_mix_square = p_wf_m^2 * xmean[3]^2;  //square of sigma of distribution of global pri
}
```

```
model {
real w_s[n_s];   //  weight of stimuli in short group
real w_l[n_l];   //  weight of stimuli in short group
real p_mix[n_mix];
real p_sig_mix[n_mix];
real w1[n_mix];
real pp_mix[n_mix];
real pp_mix_var[n_mix];
real w2[n_mix];
real y_mix_mean[n_mix];

//hyperpriors
mu_s ~ normal(xmean[1], p_wf_s^2 * xmean[1]^2);  // mean prior of short group
mu_l ~ normal(xmean[2], p_wf_l^2 * xmean[2]^2);   // mean prior of long group
mu_m~ normal(xmean[3], sig_mix_square);   // mean prior of mix group (mean of global prior)
mu_m_s~ normal(xmean[1], p_wf_s^2 * xmean[1]^2);   // mean prior of mix group (mean of global prior)
mu_m_l~ normal(xmean[2], p_wf_l^2 * xmean[2]^2);   // mean prior of mix group (mean of global prior)



//short groups
for (i in 1:n_s)
{
  w_s[i] = (mu_s[i]^2 * p_wf_s^2)/(mu_s[i]^2 * p_wf_s^2+ X_s[i]^2*wf_s^2);    // weight of current stimu
  Y_s[i] ~ normal(mu_s[i] *  w_s[i]+ (1- w_s[i])* X_s[i], sig_m_square + (mu_s[i]^2 * p_wf_s^2 * X_s[i]
}

//long groups
for (i in 1:n_l)
{
    w_l[i] = (mu_l[i]^2 * p_wf_l^2)/(mu_l[i]^2 * p_wf_l^2+ X_l[i]^2*wf_l^2);  // weight of current stimu
    Y_l[i] ~ normal(mu_l[i] *  w_l[i]+ (1- w_l[i])* X_l[i],
    sig_m_square + (mu_l[i]^2 * p_wf_l^2 * X_l[i]^2*wf_l^2) / mu_l[i]^2 * p_wf_l^2 + X_l[i]^2*wf_l^2);

}

//mix groups
for (m in 1:n_mix) {
     // first integration D_local
       if(X_mix[m] >= 1) {
         p_mix[m] = mu_m_s[m];   // local prior
         p_sig_mix[m] = p_mix[m] * p_wf_s;  //sigma of D_long
        }
       else{
         p_mix[m] = mu_m_l[m];   // local prior
         p_sig_mix[m] = p_mix[m] * p_wf_l;  //sigma of D_short
       }

      w1[m] = sig_mix_square^2 / (sig_mix_square^2 + p_sig_mix[m]^2);
      pp_mix[m] = p_mix[m] * w1[m] + (1- w1[m]) * xmean[3];  // integration with global prior
      pp_mix_var[m] = sig_mix_square^2* p_sig_mix[m]^2/(sig_mix_square^2 + p_sig_mix[m]^2); // integrati
```

```
      w2[m] = pp_mix_var[m]  / (pp_mix_var[m]  + X_mix[m]^2* g_wf^2);
      y_mix_mean[m] = w2[m] * X_mix[m] + (1-w2[m])*pp_mix[m];
      Y_mix[m] ~ normal(y_mix_mean[m],
      sig_m_square + (pp_mix_var[m] *  (p_sig_mix[m])^2 ) / (pp_mix_var[m]+ (p_sig_mix[m])^2 ));
  }
}



generated quantities {
  vector[n_mix] ynew_mix;
  vector[n_mix] p_mix_new;   // mean of internal prior of mix group
  vector[n_mix] p_sig_mix_new;
  vector[n_mix] w1_new;
  vector[n_mix] pp_mix_new;
  vector[n_mix] pp_mix_var_new;
  vector[n_mix] w2_new;
  vector[n_mix] y_mix_mean_new;


  for (m in 1:n_mix)   //prediction of mix group
  {
      // first integration D_local
       if(X_mix[m] >= 1) {
         p_mix_new[m] = mu_m_s[m];   // local prior
         p_sig_mix_new[m] = p_mix_new[m] * p_wf_s;  //sigma of D_long
        }
       else{
         p_mix_new[m] = mu_m_l[m];   // local prior
         p_sig_mix_new[m] = p_mix_new[m] * p_wf_l;  //sigma of D_short
       }

      w1_new[m] = sig_mix_square^2 / (sig_mix_square^2 + p_sig_mix_new[m]^2);
      pp_mix_new[m] = p_mix_new[m] * w1_new[m] + (1- w1_new[m]) * xmean[3];  // integration with global
      pp_mix_var_new[m] = sig_mix_square^2* p_sig_mix_new[m]^2/(sig_mix_square^2 + p_sig_mix_new[m]^2);


      w2_new[m] = pp_mix_var_new[m]  / (pp_mix_var_new[m]  + X_mix[m]^2* g_wf^2);
      y_mix_mean_new[m] = w2_new[m] * X_mix[m] + (1-w2_new[m])*pp_mix_new[m];
      ynew_mix[m] = normal_rng(y_mix_mean_new[m],
      sig_m_square + (pp_mix_var_new[m] *  (p_sig_mix_new[m])^2 ) / (pp_mix_var_new[m]+ (p_sig_mix_new[
  }
}
'


  # compile models
if (runModels == TRUE){
  stanmodelH3 <- stan_model(model_code = stancodeH3, model_name="stanmodelH3")
}
```

predicte the parameters of Bayesian

definisiton of the function to predicte the parameters of Bayesian by runing Rstan model

```
funFitBayesianStanH2 <- function(data, rstanModel, filename){
  Bayfit = {}
  Bayparlist = {}
  subList <- unique(data$NSub)
  fitparList = {}
  PredYlist_l = {}
  PredYlist_s = {}
  PredYlist_mix = {}
  expList <- unique(data$Exp)

  for (expName in expList) {
    subdata <- data %>% filter(valid > 0 & Exp == expName)
    subList <- unique(data$NSub)

    for (subNo in subList) {
      xmean <- data %>% filter(valid > 0 & Exp == expName & NSub == subNo )  %>% dplyr::group_by(group)

      subdata <- data %>% filter(valid > 0  & NSub == subNo & Exp == expName)
      data_s<- subdata %>% filter(group == 1)  # short groups only
      data_l <- subdata %>% filter(group == 2)  # long groups only
      data_mix <- subdata %>% filter(group == 3)  # mixed groups
      PredY_s_list <- data_s[c('NSub','targetDur', 'RP','Exp','group')]
      PredY_l_list <- data_l[c('NSub','targetDur', 'RP','Exp','group')]
      PredY_mix_list <- data_mix[c('NSub','targetDur', 'RP','Exp','group')]
      n_s <- length(data_s$RP)
      n_l <- length(data_l$RP)
      n_mix <- length(data_mix$RP)

      stan_data = list(Y_s=data_s$RP, n_s=n_s, X_s = data_s$targetDur,
                       Y_l=data_l$RP, n_l=n_l, X_l = data_l$targetDur,
                       X_mix = data_mix$targetDur, n_mix = n_mix,
                       Y_mix = data_mix$RP,
                       "xmean" =xmean$targetMean)  #data passed to stan

      # fit models
      subfit <- sampling(rstanModel, stan_data, chains = 4, iter = 4000,
                         control = list(adapt_delta = 0.99,
                                        max_treedepth = 15))


      #parameters <- c("p_wf_s","wf_s", "p_wf_l","wf_l","sig_m_square", "p_wf_m", "wf_m", "sig_mix_squa
      parameters <- c("p_wf_s","wf_s", "p_wf_l","wf_l","sig_m_square", "p_wf_m", "wf_m", "sig_mix_square
      fitpar <- summary(subfit, pars = parameters)$summary



      list_of_draws <- rstan::extract(subfit, pars = parameters)
      p_wf_s =  mean(list_of_draws$p_wf_s)
      wf_s =  mean(list_of_draws$wf_s)
      p_wf_l =  mean(list_of_draws$p_wf_l)
      wf_l =  mean(list_of_draws$wf_l)
      wf_m =  mean(list_of_draws$wf_m)
```

```r
      p_wf_m =  mean(list_of_draws$p_wf_m)
      sig_m_square =  mean(list_of_draws$sig_m_square)
      sig_mix_square =  mean(list_of_draws$sig_mix_square)


      pred_y_mix <- {}
      w1_list_mix <- {}
      w2_list_mix <- {}
      ynew_mix_list <- list_of_draws$ynew_mix
      w1_list <- list_of_draws$w1_new
      w2_list <- list_of_draws$w2_new
      for (n in 1:n_mix){
        pred_y_mix[n] <-  mean(ynew_mix_list[,n] )
        w1_list_mix[n] <-  mean(w1_list[,n] )
        w2_list_mix[n] <-  mean(w2_list[,n] )
      }
      PredY_mix_list$predY = pred_y_mix
      PredY_mix_list$w1 = w1_list_mix
      PredY_mix_list$w2 = w2_list_mix
      PredYlist_mix <- rbind2(PredYlist_mix, PredY_mix_list)

      Baypar = data.frame(
        Nsub = subNo,
        Exp = expName,
        p_wf_s = p_wf_s,
        wf_s = wf_s,
        p_wf_l = p_wf_l,
        wf_l = wf_l,
        wf_m = wf_m,
        p_wf_m = p_wf_m,
        sig_m_square = sig_m_square,
        sig_mix_square =sig_mix_square
      )
      Bayparlist <- rbind2(Bayparlist, Baypar)
    }
  }
  write.csv(Bayparlist, file = paste0(modelResultPath, "/Bayparlist_", filename,".csv"))
  write.csv(PredYlist_mix, file = paste0(modelResultPath, "/PredY_mix_", filename,".csv"))

  return(list("Bayparlist" = Bayparlist))
}
```

```r
funFitBayesianStanH3 <- function(data, rstanModel, filename){
  Bayfit = {}
  Bayparlist = {}
  subList <- unique(data$NSub)
  fitparList = {}
  PredYlist_l = {}
  PredYlist_s = {}
  PredYlist_mix = {}
  expList <- unique(data$Exp)

  for (expName in expList) {
    subdata <- data %>% filter(valid > 0 & Exp == expName)
```

```r
subList <- unique(data$NSub)

for (subNo in subList) {
  xmean <- data %>% filter(valid > 0 & Exp == expName & NSub == subNo )  %>% dplyr::group_by(group)

  subdata <- data %>% filter(valid > 0  & NSub == subNo & Exp == expName)
  data_s<- subdata %>% filter(group == 1)  # short groups only
  data_l <- subdata %>% filter(group == 2)  # long groups only
  data_mix <- subdata %>% filter(group == 3)  # mixed groups
  PredY_s_list <- data_s[c('NSub','targetDur', 'RP','Exp','group')]
  PredY_l_list <- data_l[c('NSub','targetDur', 'RP','Exp','group')]
  PredY_mix_list <- data_mix[c('NSub','targetDur', 'RP','Exp','group')]
  n_s <- length(data_s$RP)
  n_l <- length(data_l$RP)
  n_mix <- length(data_mix$RP)

  stan_data = list(Y_s=data_s$RP, n_s=n_s, X_s = data_s$targetDur,
                   Y_l=data_l$RP, n_l=n_l, X_l = data_l$targetDur,
                   X_mix = data_mix$targetDur, n_mix = n_mix,
                   Y_mix = data_mix$RP,
                   "xmean" =xmean$targetMean)  #data passed to stan

  # fit models
  subfit <- sampling(rstanModel, stan_data, chains = 4, iter = 4000,
                     control = list(adapt_delta = 0.99,
                                    max_treedepth = 15))


   parameters <- c("g_wf", "p_wf_s","wf_s", "p_wf_l","wf_l","sig_m_square", "p_wf_m", "wf_m", "sig_
  fitpar <- summary(subfit, pars = parameters)$summary



  list_of_draws <- rstan::extract(subfit, pars = parameters)
  p_wf_s =  mean(list_of_draws$p_wf_s)
  wf_s =  mean(list_of_draws$wf_s)
  p_wf_l =  mean(list_of_draws$p_wf_l)
  wf_l =  mean(list_of_draws$wf_l)
  wf_m =  mean(list_of_draws$wf_m)
  p_wf_m =  mean(list_of_draws$p_wf_m)
  sig_m_square =  mean(list_of_draws$sig_m_square)
  sig_mix_square =  mean(list_of_draws$sig_mix_square)


  pred_y_mix <- {}
  w1_list_mix <- {}
  w2_list_mix <- {}
  ynew_mix_list <- list_of_draws$ynew_mix
  w1_list <- list_of_draws$w1_new
  w2_list <- list_of_draws$w2_new

  for (n in 1:n_mix){
    pred_y_mix[n] <-  mean(ynew_mix_list[,n] )
```

```
        w1_list_mix[n] <-  mean(w1_list[,n] )
        w2_list_mix[n] <-  mean(w2_list[,n] )
      }
      PredY_mix_list$predY = pred_y_mix
      PredY_mix_list$w1 = w1_list_mix
      PredY_mix_list$w2 = w2_list_mix
      PredYlist_mix <- rbind2(PredYlist_mix, PredY_mix_list)

      Baypar = data.frame(
        Nsub = subNo,
        Exp = expName,
        p_wf_s = p_wf_s,
        wf_s = wf_s,
        p_wf_l = p_wf_l,
        wf_l = wf_l,
        wf_m = wf_m,
        p_wf_m = p_wf_m,
        sig_m_square = sig_m_square,
        sig_mix_square =sig_mix_square
      )
      Bayparlist <- rbind2(Bayparlist, Baypar)
    }
  }
  write.csv(Bayparlist, file = paste0(modelResultPath, "/Bayparlist_", filename,".csv"))
  write.csv(PredYlist_mix, file = paste0(modelResultPath, "/PredY_mix_", filename,".csv"))

  return(list("Bayparlist" = Bayparlist))
}
```

run the model

run H2

run H3

display the model restults

Merge the Result data

To preprocess the model result data, and merge different model version data together.

```
needmerge=1
versionlist =c('Version2_2')
modellist = c('H2','H3')

if (needmerge == 1){
  predY_mix_filename <- paste0("PredY_mix_", modellist, ".csv")
  #predY_s_filename <- paste0("PredY_s_", modellist, ".csv")
  #predY_l_filename <- paste0("PredY_l_", modellist, ".csv")
  BayParlist_filename <- paste0("Bayparlist_", modellist, ".csv")
  mergeData(predY_mix_filename, 'predY_mix', versionlist)
  #mergeData(predY_s_filename, 'predY_s', versionlist)
  #mergeData(predY_l_filename, 'predY_l', versionlist)
  mergeData(BayParlist_filename, 'Bayparlist', versionlist)
}
```

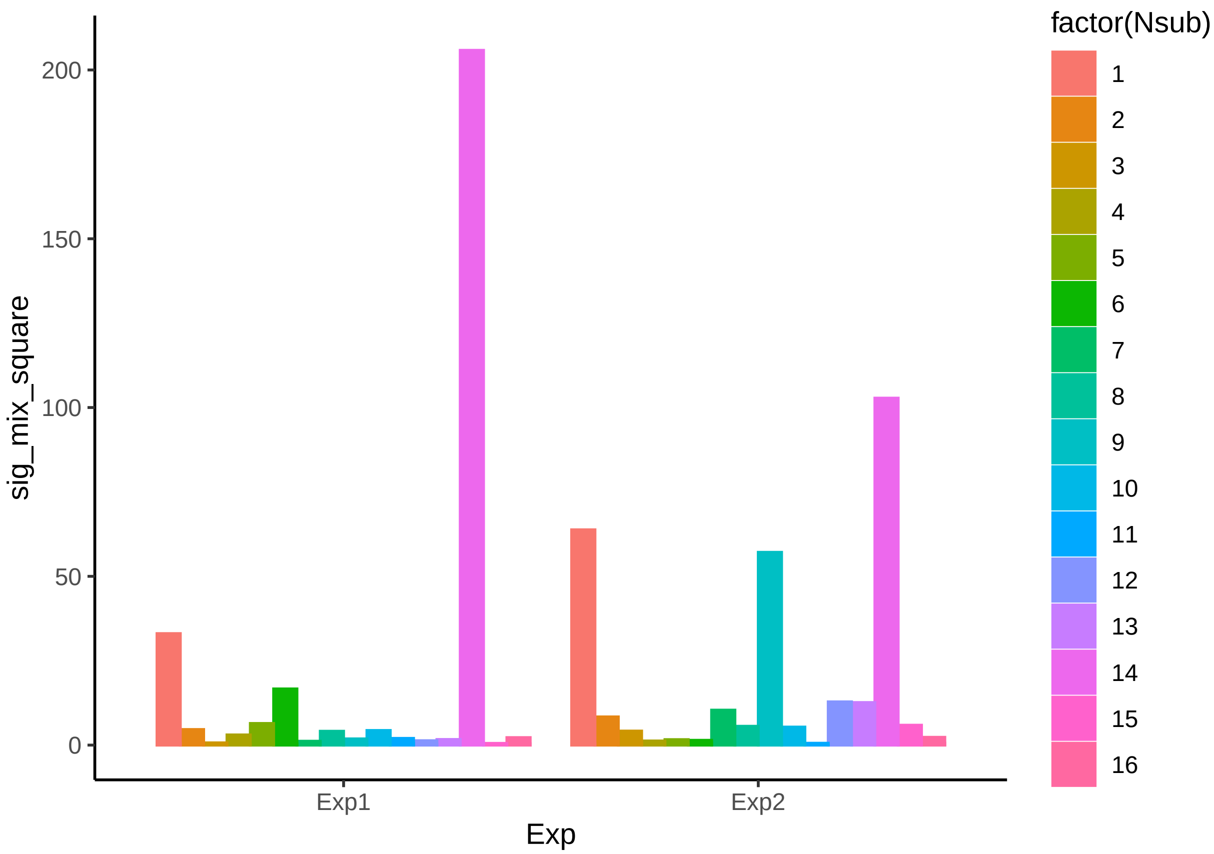
load the model result data

Analysis on the Rstan model parameters

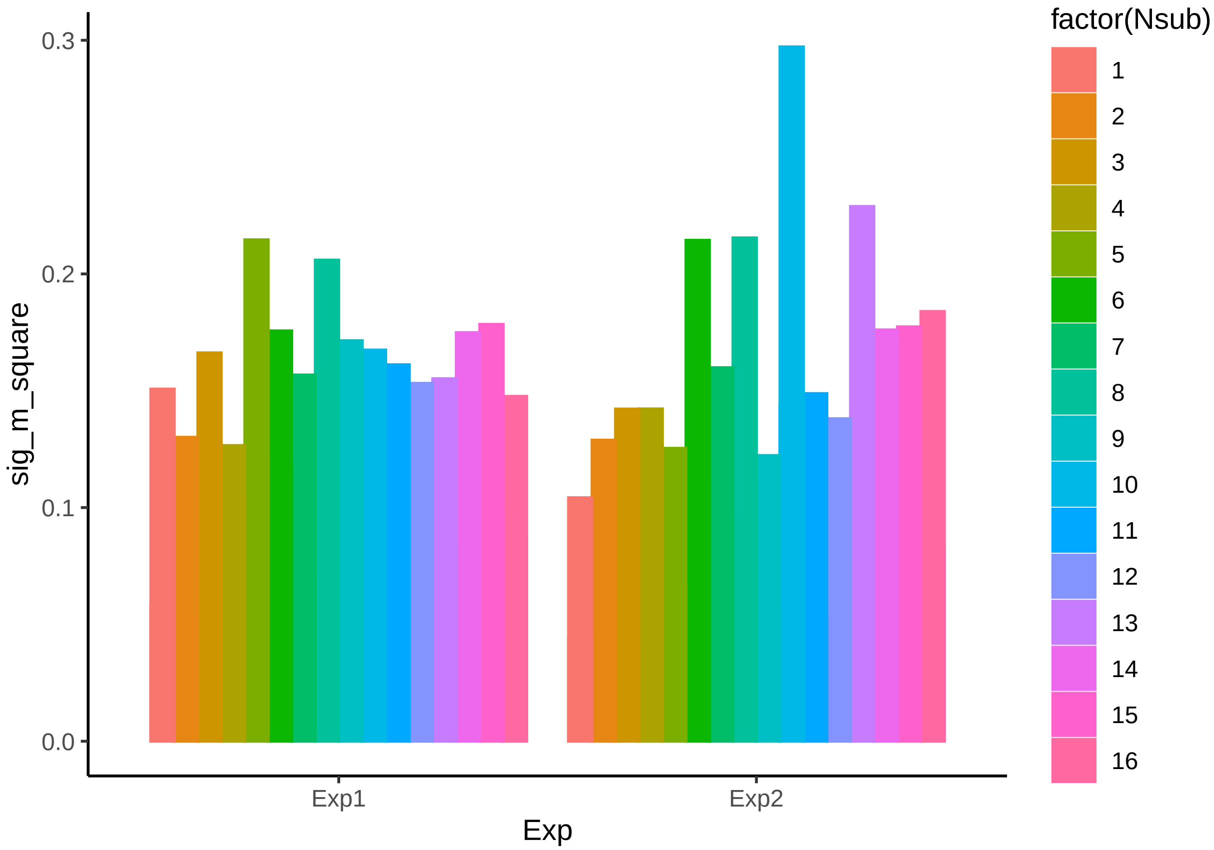
Parameters

```
  AllDat_Bayparlist$model <- factor(AllDat_Bayparlist$model, labels = c( "Hierarchical local-global mode
m_Baypar <- group_by(AllDat_Bayparlist, Exp, Nsub, model,version) %>%
  summarize(p_wf_s = mean(p_wf_s), wf_s = mean(wf_s),
            p_wf_l = mean(p_wf_l), wf_l = mean(wf_l),
            wf_m = mean(wf_m),  p_wf_m = mean(p_wf_m),
            sig_m_square =mean(sig_m_square),
            sig_mix_square = mean(sig_mix_square))

# %>%
# group_by(Exp, model,version) %>%
#   summarize(m_as = mean(m_as), n = n(), m_al = mean(m_al),
#             m_bs = mean(m_bs), m_bl = mean(m_bl),
#             m_wf = mean(m_wf))
```

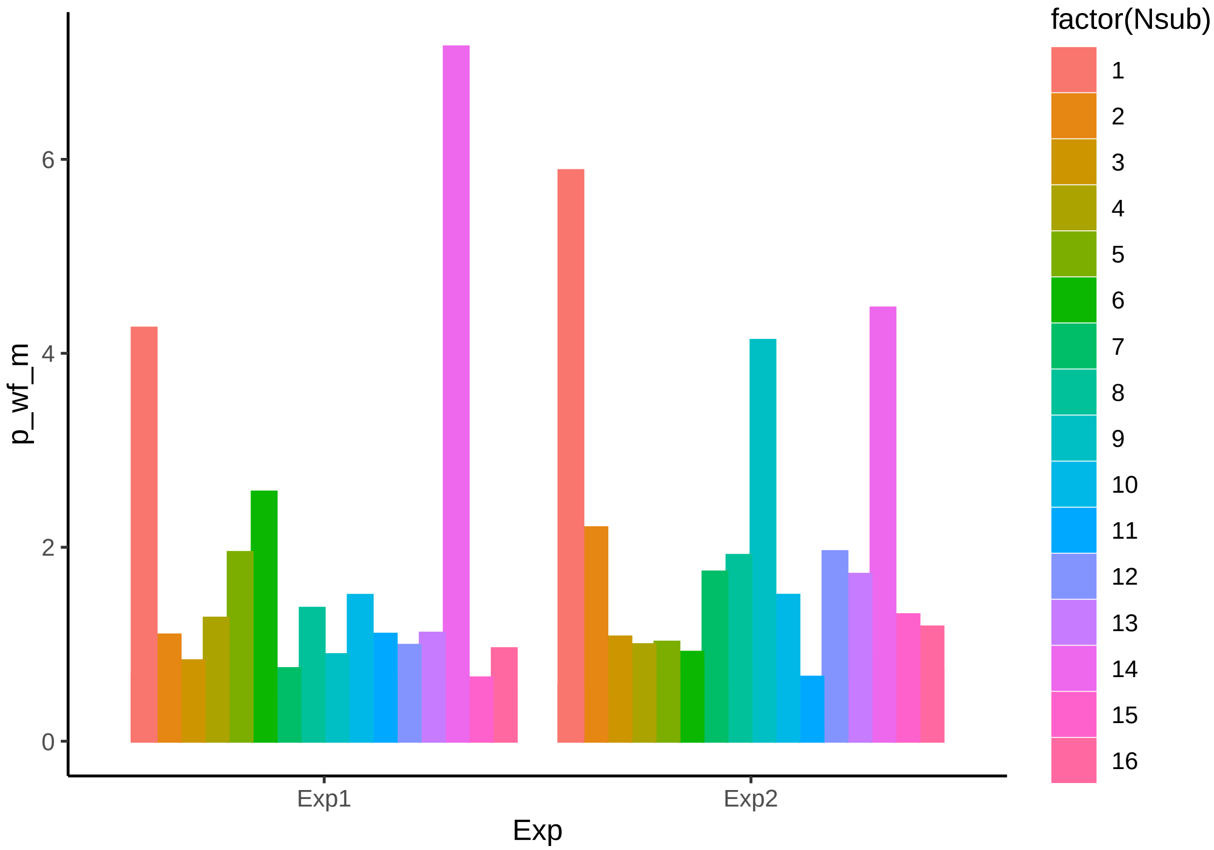
```
ggplot(m_Baypar, aes(x = Exp, y = sig_mix_square, color = factor(Nsub), fill = factor(Nsub), group = fa
    geom_bar(stat = "identity",
             position = position_dodge()) +
  theme_new
```



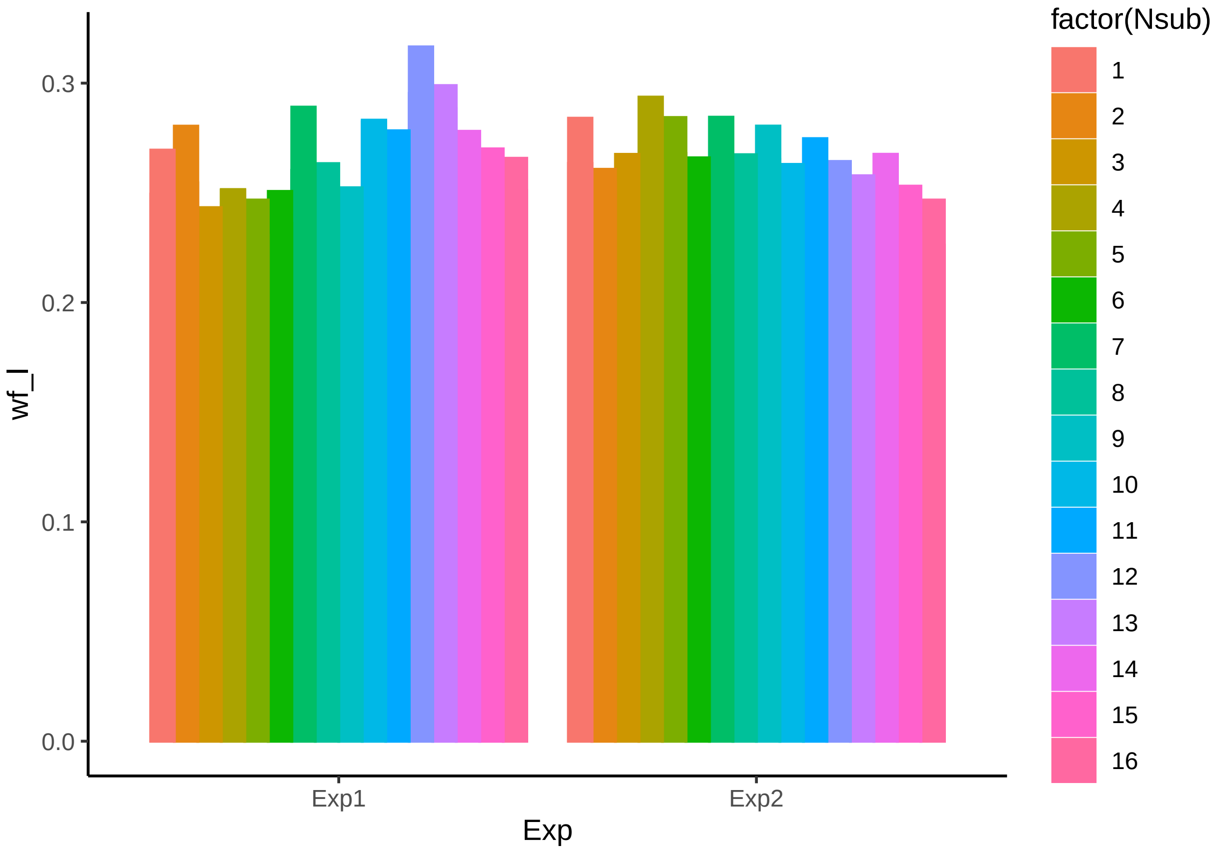

```
ggplot(m_Baypar, aes(x = Exp, y = sig_m_square, color = factor(Nsub), fill = factor(Nsub), group = fact
    geom_bar(stat = "identity",
             position = position_dodge()) +
  theme_new
```



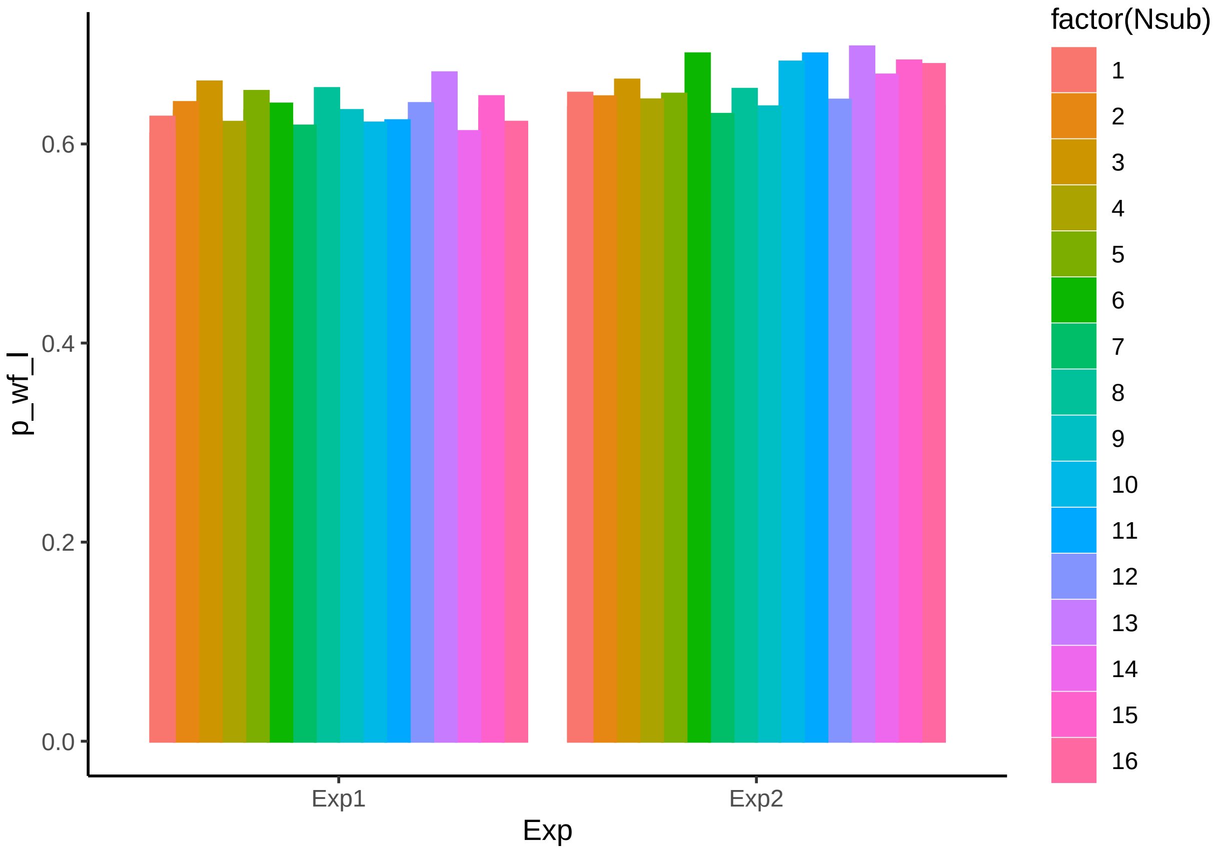

```
ggplot(m_Baypar, aes(x = Exp, y = p_wf_m, color = factor(Nsub), fill = factor(Nsub), group = factor(Nsu
    geom_bar(stat = "identity",
             position = position_dodge()) +
  theme_new
```



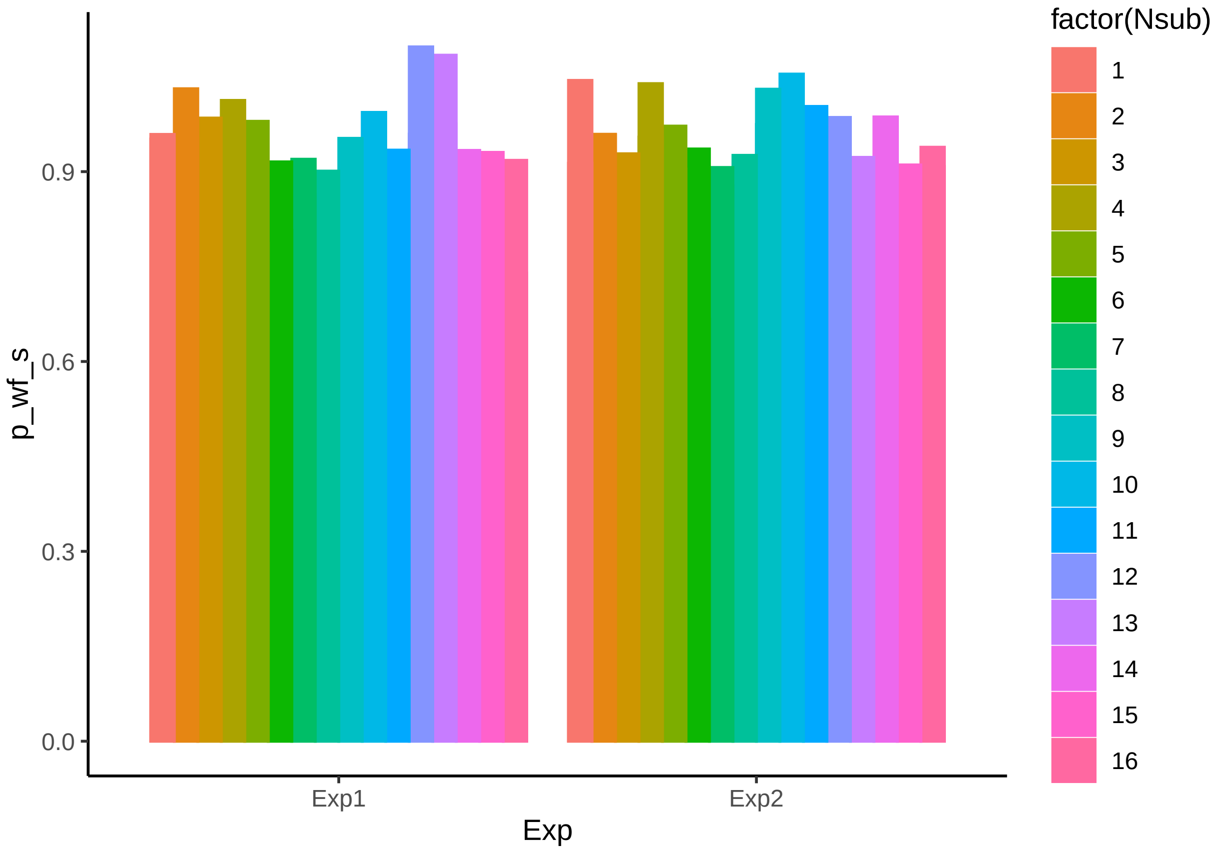

```
ggplot(m_Baypar, aes(x = Exp, y = wf_l, color = factor(Nsub), fill = factor(Nsub), group = factor(Nsub)
    geom_bar(stat = "identity",
             position = position_dodge()) +
  theme_new
```



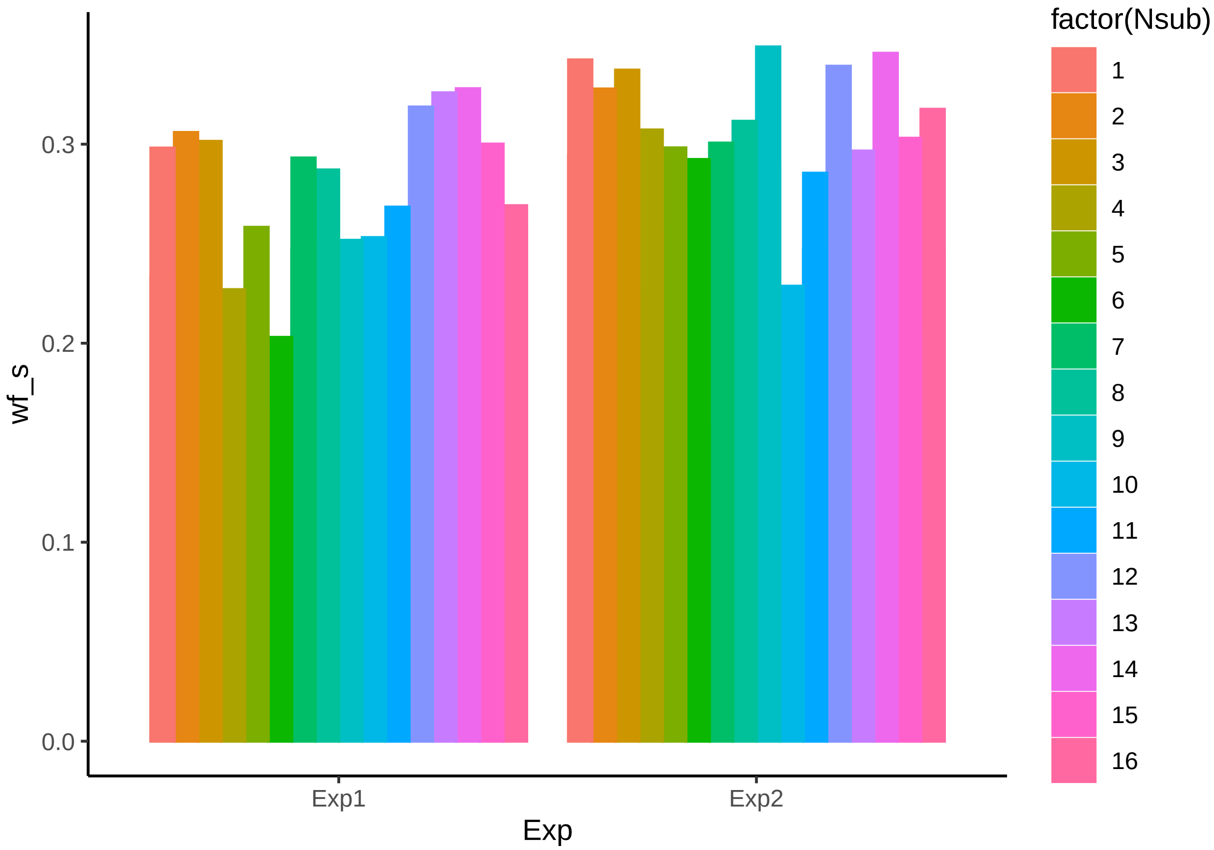

```
ggplot(m_Baypar, aes(x = Exp, y = p_wf_l, color = factor(Nsub), fill = factor(Nsub), group = factor(Nsu
    geom_bar(stat = "identity",
             position = position_dodge()) +
  theme_new
```




```
ggplot(m_Baypar, aes(x = Exp, y = p_wf_s, color = factor(Nsub), fill = factor(Nsub), group = factor(Nsu
    geom_bar(stat = "identity",
             position = position_dodge()) +
  theme_new
```




```
ggplot(m_Baypar, aes(x = Exp, y = wf_s, color = factor(Nsub), fill = factor(Nsub), group = factor(Nsub)
    geom_bar(stat = "identity",
             position = position_dodge()) +
  theme_new
```

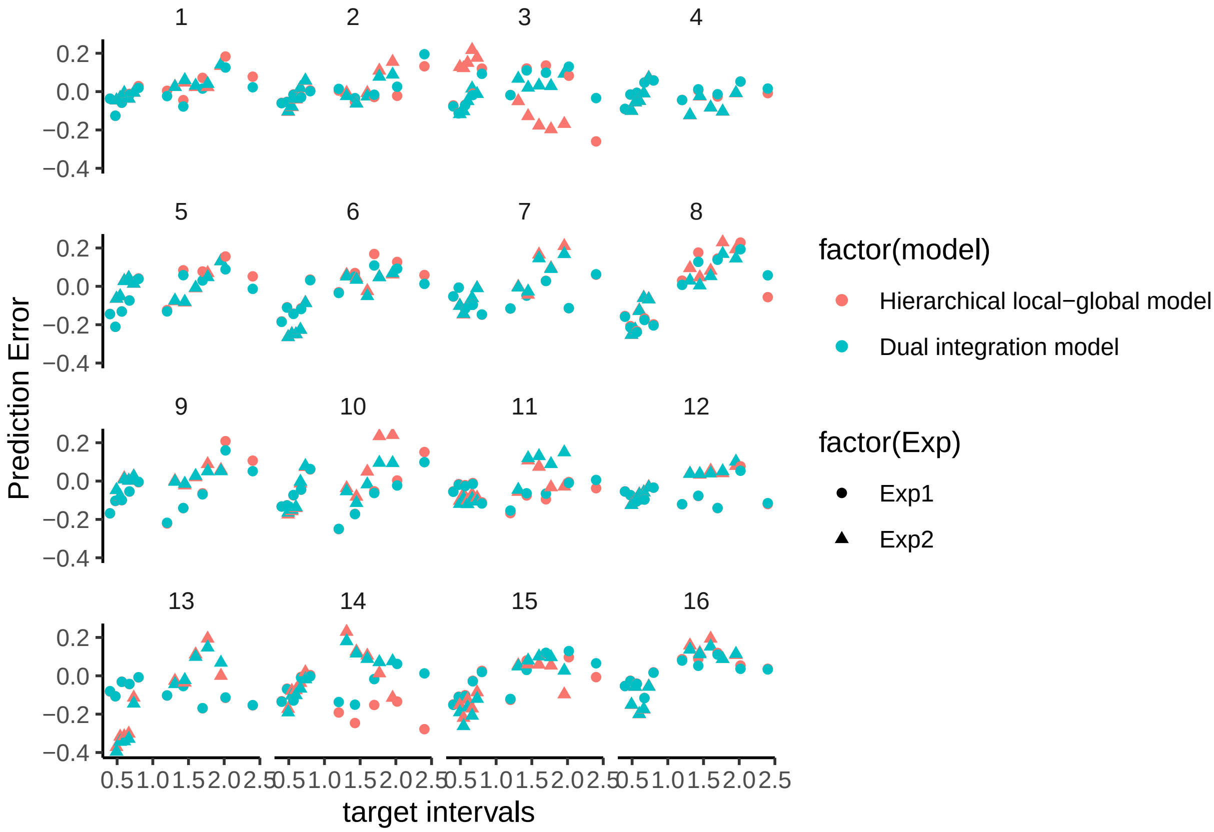


Prediction results (mixed block)

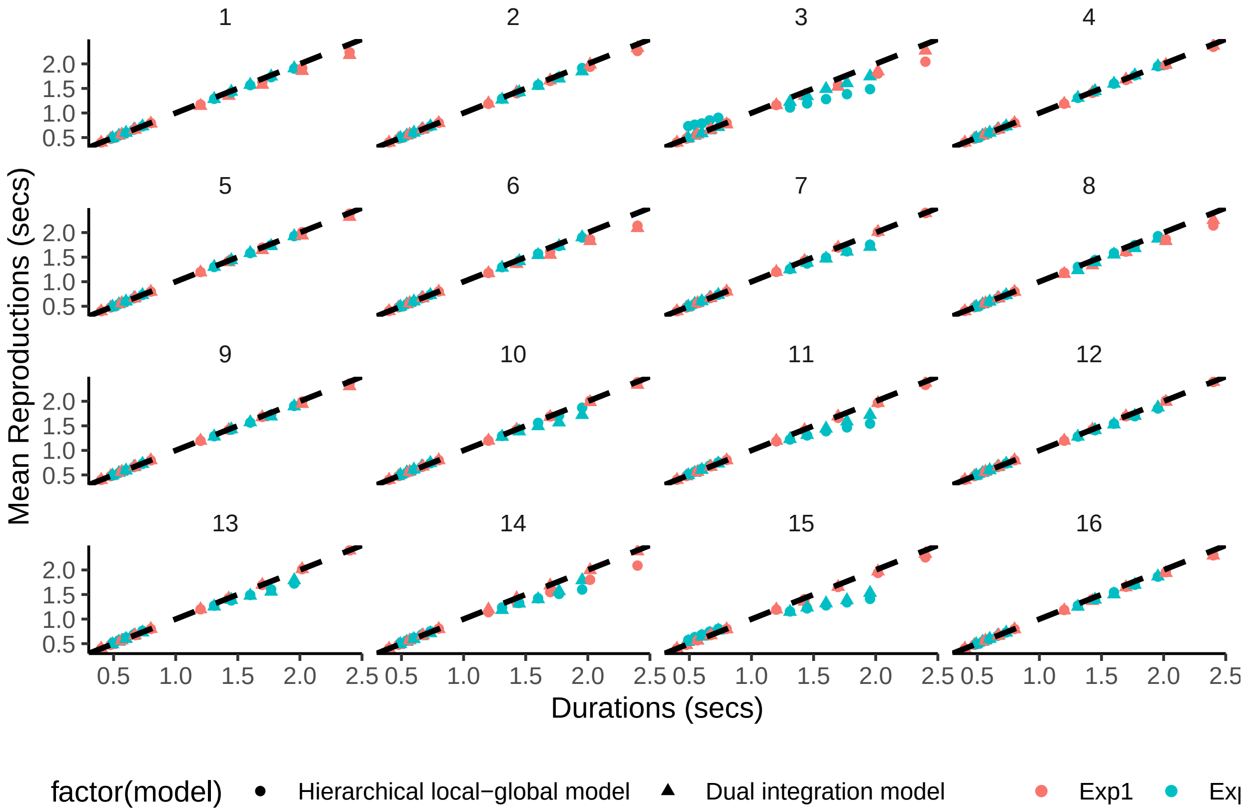
```
AllDat_predY_mix$model <- factor(AllDat_predY_mix$model, labels = c( "Hierarchical local-global model",

predY_mix <- group_by(AllDat_predY_mix, targetDur, Exp, NSub, model,version) %>%
  summarize(m_RP = mean(RP), n = n(), sd_RP = sd(RP)/ sqrt(n-1),m_predY = mean(predY), sd_predY = sd(pr
predY_mix$m_rpErr = predY_mix$m_predY - predY_mix$m_RP
predY_mix$m_relativeErr = predY_mix$m_rpErr / predY_mix$targetDur
```

The predication of mix blocks

```
#plot Error in predication
fig_prederr <- ggplot(data=predY_mix, aes(x= targetDur, y=m_rpErr, shape =factor(Exp), color= factor(mo
    geom_point()+
    labs(x="target intervals", y="Prediction Error")+
  facet_wrap(~NSub) +
  theme_new

ggsave(file.path(modelResultPath,'fig_prederr_mix.png'), fig_prederr, width = 7, height = 5)
fig_prederr
```

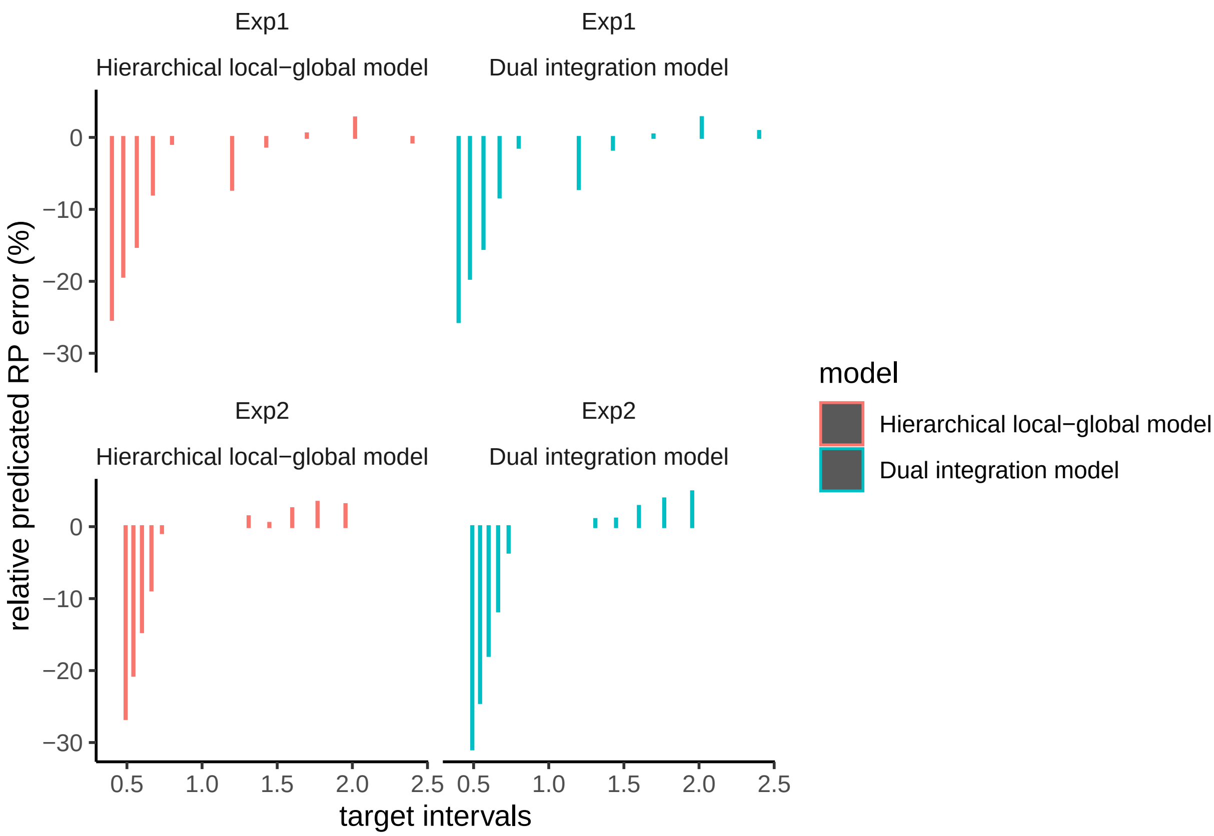



```
#plot the average of the predicted Y under the mixed condition
fig_mpredY = ggplot(predY_mix) +
  geom_point(aes(targetDur, m_predY, group = factor(NSub), color = factor(Exp), shape = factor(model) ))
  #geom_line(aes(targetDur, m_RP, group = factor(NSub), color = factor(NSub)),  size = 1) +
  #geom_errorbar(aes(ymin = m_m_predY-se_m_predY, ymax = m_m_predY + se_m_predY), width = 0.05) +
  geom_abline(slope = 1, linetype = 2, size = 1) + # add diagonal line
  facet_wrap(~Exp) +
  guides(color = guide_legend(title = element_blank())) + # remove legend title
  theme_classic() +
  theme(strip.background = element_blank()) + # remove subtitle background
  labs(x = "Durations (secs)", y = "Mean Reproductions (secs)", size =15) + theme(legend.position="bott
  facet_wrap(NSub~.)

ggsave(file.path(modelResultPath,'fig_mpredY_mix.png'), fig_mpredY, width = 7, height = 5)
fig_mpredY
```

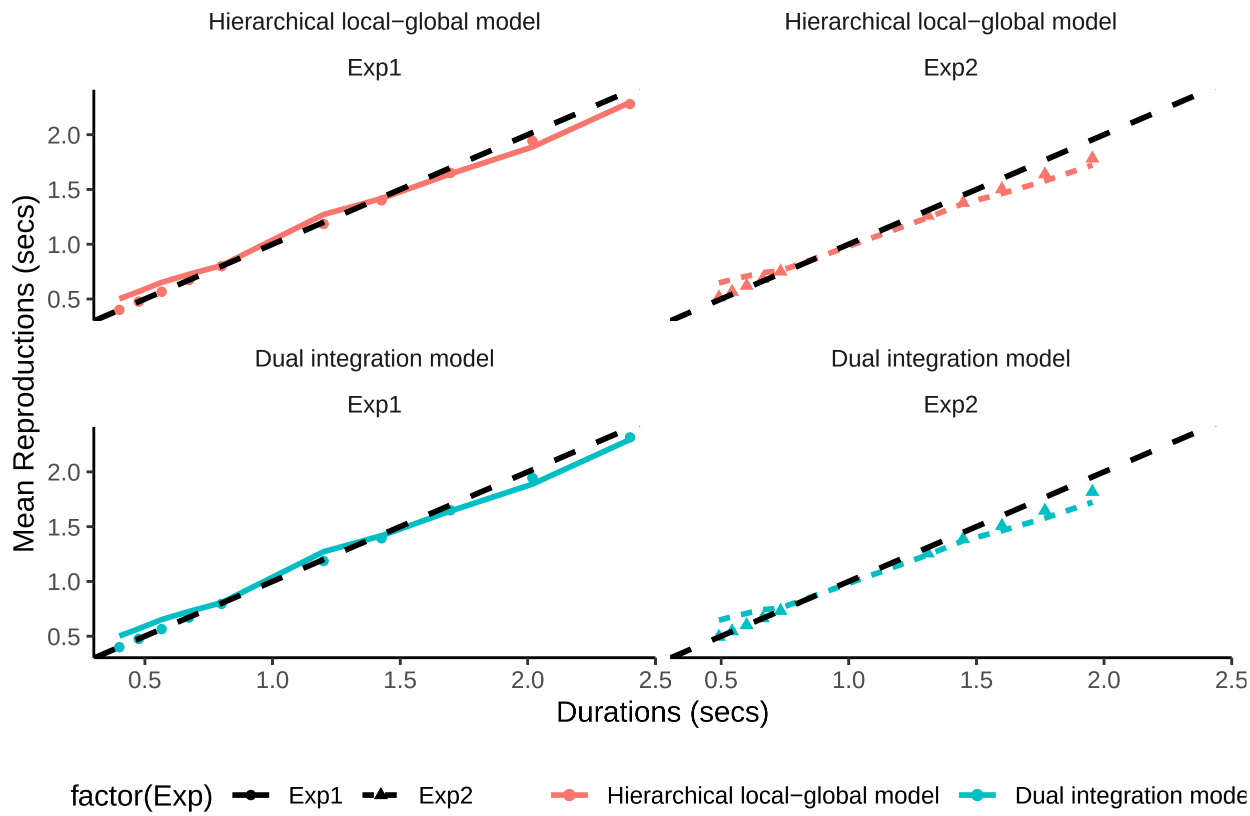



```
m_predY_mix <- predY_mix %>%
  group_by(targetDur, Exp, model,version) %>%
  summarize(
    n = n(),
    m_m_predY = mean(m_predY),
    se_m_predY = sd(m_predY) / sqrt(n - 1),
    m_m_RP = mean(m_RP),
    se_m_RP = sd(m_RP) / sqrt(n-1)
  )
m_predY_mix$m_rpErr = m_predY_mix$m_m_predY - m_predY_mix$m_m_RP
m_predY_mix$m_relativeErr = m_predY_mix$m_rpErr / m_predY_mix$targetDur
```

```
#plot relative Error for mixed blocks
 fig_rerr_model <-  ggplot(data=m_predY_mix, aes(x= targetDur, y=m_relativeErr*100, group = model, colo
    geom_bar(stat="identity")+
    labs(x="target intervals", y="relative predicated RP error (%)")+
  facet_wrap(Exp~model) +
  theme_new

ggsave(file.path(modelResultPath,'fig_rerr_model_mix.png'), fig_rerr_model, width = 7, height = 5)
fig_rerr_model
```

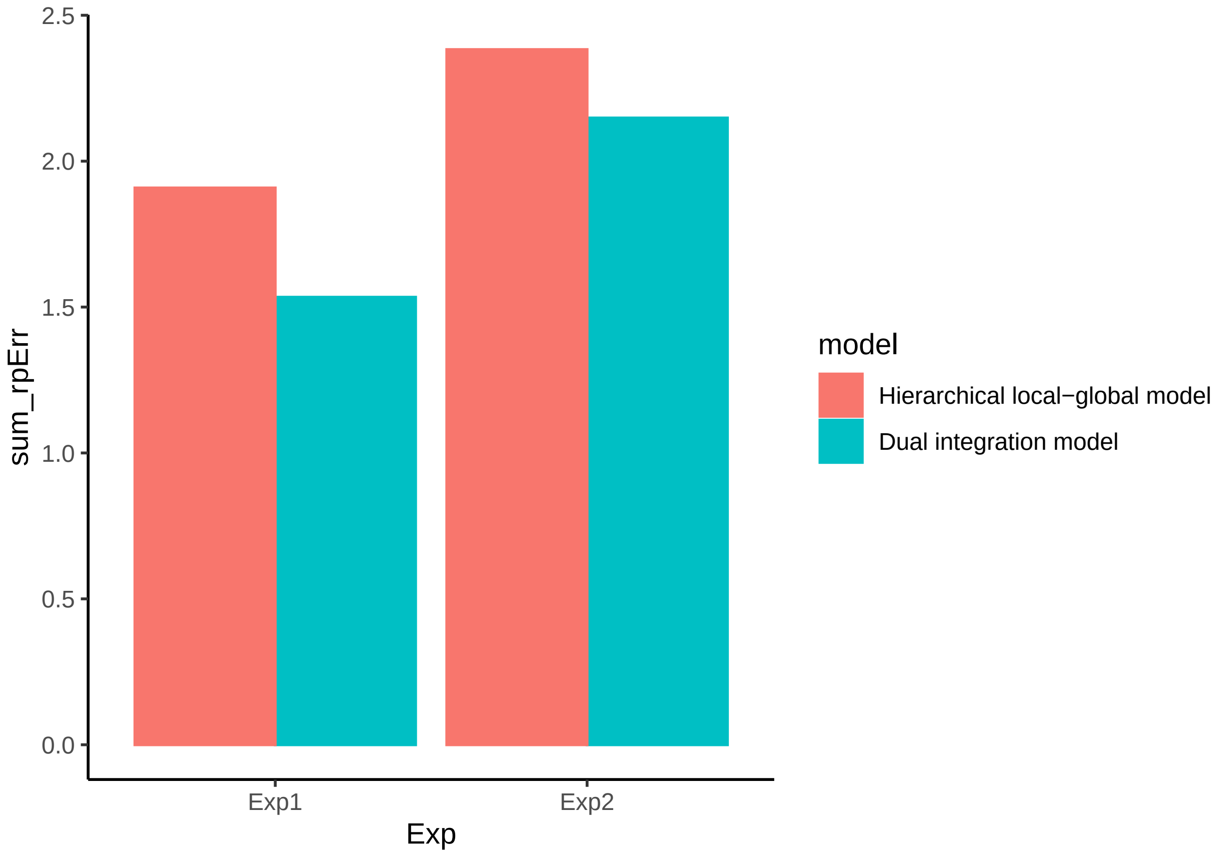



```r
#plot the average of the predicted Y under the mixed condition
fig_mPredY_mix = ggplot(m_predY_mix) +
  geom_point(aes(targetDur, m_m_predY, group = model, color = model, shape = factor(Exp))) +
  geom_line(aes(targetDur, m_m_RP, group = model, color = model, linetype = factor(Exp)),  size = 1) +
  #geom_errorbar(aes(ymin = m_m_predY-se_m_predY, ymax = m_m_predY + se_m_predY), width = 0.05) +
  geom_abline(slope = 1, linetype = 2, size = 1) + # add diagonal line
  facet_wrap(model~Exp) +
  guides(color = guide_legend(title = element_blank())) + # remove legend title
  theme_classic() +
  theme(strip.background = element_blank()) + # remove subtitle background
  labs(x = "Durations (secs)", y = "Mean Reproductions (secs)", size =15) + theme(legend.position="bott
fig_mPredY_mix
```




```
ggsave(file.path(modelResultPath,'fig_mPredY_mix.png'), fig_mPredY_mix, width = 7, height = 5)
```

```
predY_mix$rpErr_squared <- predY_mix$m_rpErr^2
m_predY_Err <- group_by(predY_mix, Exp, model,version) %>%
  summarize(sum_rpErr = sum(rpErr_squared),
            n = n())

fig_rpErr_model <- ggplot(m_predY_Err, aes(x = Exp, y = sum_rpErr, color = model, fill = model, group =
    geom_bar(stat = "identity",
             position = position_dodge()) +
  theme_new

ggsave(file.path(modelResultPath,'fig_rpErr_model_mix.png'), fig_rpErr_model, width = 7, height = 5)

fig_rpErr_model
```



2.5
2.0
1.5
1.0
0.5
0.0
sum_rpErr
Exp1
Exp2
Exp
model
Hierarchical local−global model
Dual integration model