ModelReport V5

Fiona Zhu

05/03/2020

Load the packages

```
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.2.1 --
```

```
## v ggplot2 3.2.1       v purrr   0.3.2
## v tibble  2.0.1       v dplyr   0.8.0.1
## v tidyr   0.8.3       v stringr 1.4.0
## v readr   1.3.1       v forcats 0.4.0
```

```
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
library(rstan)
```

```
## Loading required package: StanHeaders
```

```
## rstan (Version 2.18.2, GitRev: 2e1f913d3ca3)
```

```
## For execution on a local, multicore CPU with excess RAM we recommend calling
## options(mc.cores = parallel::detectCores()).
## To avoid recompilation of unchanged Stan programs, we recommend calling
## rstan_options(auto_write = TRUE)
```

```
##
## Attaching package: 'rstan'
```

```
## The following object is masked from 'package:tidyr':
##
##     extract
```

```
library(readr)
library(plyr)
```

```
## ------------------------------------------------------------------------------
```

```
## You have loaded plyr after dplyr - this is likely to cause problems.
## If you need functions from both plyr and dplyr, please load plyr first, then dplyr:
## library(plyr); library(dplyr)
```

```
## ------------------------------------------------------------------------------
```

```
##
## Attaching package: 'plyr'
```

```
## The following objects are masked from 'package:dplyr':
##
##     arrange, count, desc, failwith, id, mutate, rename, summarise,
##     summarize
```

```
## The following object is masked from 'package:purrr':
##
```

```
##     compact
```

```
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, '/Version5')
```

load experimental data

//TODO ynew_mix[sub, m] = normal_rng(mu, sigma)

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,"/data" )
    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(rstanmodelPath, "/AllDat_", filename, ".csv"))
  write.csv(file=savedataDir, merge.data.all)
}
```

H3: A hierarchical local-global model

```
stancodeH3 <- '//H3: Prior is dynamic updated (the distribution of the prior is updated)
//In H3, global and local priors are integrated first. That means local prior is integrated with sensory
// then global prior integrates with sensory input.
data {
int nsubjs;
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
real Y_s[nsubjs, n_s];   //measured reproductive duration (short group)
real X_s[nsubjs, n_s];   //stimulus duration (short group)
real Y_l[nsubjs, n_l];   //measured reproductive duration (long group)
real X_l[nsubjs, n_l];   //stimulus duration (long group)

int<lower=0> n_mix; //number of the up mixed data points
real X_mix[nsubjs, n_mix]; //measured reproductive duration (mixed)
real Y_mix[nsubjs, n_mix]; //measured reproductive duration (mixed)
int<lower=0> n_mix_te; //number of the up mixed data points (test the model)
real X_mix_te[nsubjs, n_mix_te]; //measured reproductive duration (mixed)
}

parameters {
real<lower=0, upper = 1> p_wf;  //Weber Fraction of prior
real<lower=0, upper = 1> wf;  //Weber Fraction of prior
real<lower=0> sigma_s;   //sd for the short group
real<lower=0> sigma_l;   //sd for the long group
real<lower=0> sigma_mix;   //sd for the mix group
real<lower=0, upper = 1> g_wf;  //Webber Fraction for the global prior
vector[nsubjs] a_s;
vector[nsubjs] b_s;
vector[nsubjs] a_l;
vector[nsubjs] b_l;
}


model {
real p_mix[nsubjs, n_mix];
real p_sig_mix[nsubjs, n_mix];
real w_mix[nsubjs, n_mix];
real d_mix_hat[nsubjs, n_mix];
real var_mix[nsubjs, n_mix];
real g_w_mix[nsubjs, n_mix];
```

```
real y_mix_mean[nsubjs, n_mix];
real sig_mix[nsubjs, n_mix];

//hyperpriors
a_s ~ normal(0, 1);   // short group intercept
b_s ~ normal(1, 1);   // short group slope
a_l ~ normal(0, 1);   // long group intercept
b_l ~ normal(1, 1);   // long group slope

sigma_s ~ cauchy(0, 1);
sigma_l ~ cauchy(0, 1);
sigma_mix ~ cauchy(0, 1);

for (sub in 1:nsubjs) {
  for (i in 1:n_s)
  {
    if(X_s[sub, i] > 0){
        Y_s[sub, i] ~ normal(a_s[sub] + b_s[sub] * X_s[sub, i], sigma_s); //short groups
    }
  }
}

for (sub in 1:nsubjs) {
  for (j in 1:n_l)
  {
    if(X_l[sub, j] > 0){
        Y_l[sub, j] ~ normal(a_l[sub] + b_l[sub] * X_l[sub, j], sigma_l); //long groups
    }
  }
}

for (sub in 1:nsubjs) {
  for (m in 1:n_mix)
  {
    if (X_mix[sub, m] > 0){
      // H3 part1 integration of local priors firstly
      if (X_mix[sub, m] < 1){
        p_mix[sub, m] = a_s[sub] + b_s[sub] * mean(X_s[sub,]);
      }else{
        p_mix[sub, m] = a_l[sub] + b_l[sub] * mean(X_l[sub,]);
      }
      p_sig_mix[sub, m] = p_mix[sub, m] * p_wf;
      w_mix[sub, m] = p_sig_mix[sub, m]^2 / (p_sig_mix[sub, m]^2 +  (X_mix[sub, m]*wf)^2);
      d_mix_hat[sub, m] = w_mix[sub, m] * X_mix[sub, m] + (1-w_mix[sub, m])*p_mix[sub, m];

      // H3 part2  posterior variances
      var_mix[sub, m] = p_sig_mix[sub, m]^2 * (X_mix[sub, m]*wf)^2 / (p_sig_mix[sub, m]^2 + (X_mix[sub,

      // H3 part3 integration of the global prior
      sig_mix[sub, m] = mean(X_mix[sub,])*g_wf;  //global prior of up mixed block
      g_w_mix[sub, m] = sig_mix[sub, m]^2 / (sig_mix[sub, m]^2 + var_mix[sub, m]);
      y_mix_mean[sub, m] = g_w_mix[sub, m] * d_mix_hat[sub, m]  + (1-g_w_mix[sub, m]) * mean(X_mix[sub,
      Y_mix[sub, m] ~ normal(y_mix_mean[sub, m], sigma_mix);
```

```
    }
  }
}

}

generated quantities {
real p_mix_new[nsubjs, n_mix];
real p_sig_mix_new[nsubjs, n_mix];
real w_mix_new[nsubjs, n_mix];
real d_mix_hat_new[nsubjs, n_mix];
real var_mix_new[nsubjs, n_mix];
real g_w_mix_new[nsubjs, n_mix];
real sig_mix_new[nsubjs, n_mix];
real ynew_mix[nsubjs, n_mix];


for (sub in 1:nsubjs) {
  for (m in 1:n_mix_te)  //prediction of up mixed range
  {
    if (X_mix_te[sub, m] > 0){
      // H3 part1 integration of local priors firstly
      if (X_mix_te[sub, m] < 1){
        p_mix_new[sub, m] = a_s[sub] + b_s[sub] * mean(X_s[sub,]);
      }else{
        p_mix_new[sub, m] = a_l[sub] + b_l[sub] * mean(X_l[sub,]);
      }
      p_sig_mix_new[sub, m] = p_mix_new[sub, m] * p_wf;
      w_mix_new[sub, m] = p_sig_mix_new[sub, m]^2 / (p_sig_mix_new[sub, m]^2 +  (X_mix_te[sub, m]*wf)^2
      d_mix_hat_new[sub, m] = w_mix_new[sub, m] * X_mix_te[sub, m] + (1-w_mix_new[sub, m])*p_mix_new[su

      // H3 part2  posterior variances
      var_mix_new[sub, m] = p_sig_mix_new[sub, m]^2 * (X_mix_te[sub, m]*wf)^2 / (p_sig_mix_new[sub, m]^

      // H3 part3 integration of the global prior
      sig_mix_new[sub, m] = mean(X_mix_te[sub,])*g_wf;  //global prior of up mixed block
      g_w_mix_new[sub, m] = sig_mix_new[sub, m]^2 / (sig_mix_new[sub, m]^2 + var_mix_new[sub, m]);

      ynew_mix[sub, m] = g_w_mix_new[sub, m] * d_mix_hat_new[sub, m]  + (1-g_w_mix_new[sub, m]) * mean(
    }
  }
}

}
'
```

H4: Global prior (the dual integration model)

```
stancodeH4 <- '//H4: The dual integration model
//integration of local priors firstly, then integration of the global prior
data {
int nsubjs;
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
real Y_s[nsubjs, n_s];   //measured reproductive duration (short group)
real X_s[nsubjs, n_s];   //stimulus duration (short group)
real Y_l[nsubjs, n_l];   //measured reproductive duration (long group)
real X_l[nsubjs, n_l];   //stimulus duration (long group)

int<lower=0> n_mix; //number of the mixed data points
real X_mix[nsubjs, n_mix]; //measured target duration (mixed group)
real Y_mix[nsubjs, n_mix]; //measured reproductive duration (mixed group)
int<lower=0> n_mix_te; //number of the mixed data points (test the model)
real X_mix_te[nsubjs, n_mix_te]; //measured reproductive duration (up mixed)
}

parameters {
real<lower=0, upper = 1> p_wf;  //Weber Fraction of prior
real<lower=0, upper = 1> wf;  //Weber Fraction of prior
real<lower=0> sigma_s;   //sd for the short group
real<lower=0> sigma_l;   //sd for the long group
real<lower=0> sigma_mix;   //sd for the mix group
real<lower=0, upper = 1> g_wf;  //Webber Fraction for the global prior
vector[nsubjs] a_s;
vector[nsubjs] b_s;
vector[nsubjs] a_l;
vector[nsubjs] b_l;
}

model {
real w_mix[nsubjs, n_mix];
real p_sig_mix[nsubjs, n_mix];
real p_mix[nsubjs, n_mix];
real pp_mix[nsubjs, n_mix];
real pp_mix_var[nsubjs, n_mix];
real pp_w[nsubjs, n_mix];
real sig_mix[nsubjs, n_mix];


//hyperpriors
a_s ~ normal(0, 1);   // short group intercept
b_s ~ normal(1, 1);   // short group slope
a_l ~ normal(0, 1);   // long group intercept
b_l ~ normal(1, 1);   // long group slope

sigma_s ~ cauchy(0, 1);
sigma_l ~ cauchy(0, 1);
sigma_mix ~ cauchy(0, 1);
//priors


//likelihood
for (sub in 1:nsubjs) {
  for (i in 1:n_s)
  {
    if(X_s[sub, i] > 0){
```

```
      Y_s[sub, i] ~ normal(a_s[sub] + b_s[sub] * X_s[sub, i], sigma_s); //short groups
    }
  }
}

for (sub in 1:nsubjs) {
  for (j in 1:n_l)
  {
    if(X_l[sub, j] > 0){
      Y_l[sub, j] ~ normal(a_l[sub] + b_l[sub] * X_l[sub, j], sigma_l); //long groups
    }
  }
}

for (sub in 1:nsubjs) {
  for (m in 1:n_mix)  //integration of local priors firstly
  {
    if(X_mix[sub, m] > 0){
      // first integration
      if(X_mix[sub, m] < 1){
        p_mix[sub, m] = a_s[sub] + b_s[sub] * mean(X_s[sub,]);
      }
      else{
        p_mix[sub, m] = a_l[sub] + b_l[sub] * mean(X_l[sub,]);
      }
      sig_mix[sub, m] = mean(X_mix[sub,])*g_wf;  //TODO global prior of up mixed block    // g_mean
      p_sig_mix[sub, m] = p_mix[sub, m] * p_wf;
      w_mix[sub, m] = sig_mix[sub, m]^2 / (sig_mix[sub, m]^2 + p_sig_mix[sub, m]^2); //pp_w_mix
      pp_mix[sub, m] = p_mix[sub, m] * w_mix[sub, m] + (1- w_mix[sub, m]) * mean(X_mix[sub, ]);
      pp_mix_var[sub, m] = sig_mix[sub, m]^2* p_sig_mix[sub, m]^2/(sig_mix[sub, m]^2 + p_sig_mix[sub, m
      pp_w[sub, m] = pp_mix_var[sub, m]  / (pp_mix_var[sub, m]  + X_mix[sub, m]^2 * wf^2);
      Y_mix[sub, m] ~ normal(pp_w[sub, m] * X_mix[sub, m] + (1-pp_w[sub, m])*pp_mix[sub, m], sigma_mix)
    }
  }
 }
}


generated quantities {
real ynew_mix[nsubjs, n_mix];
real w_mix_new[nsubjs, n_mix];
real p_sig_mix_new[nsubjs, n_mix];
real p_mix_new[nsubjs, n_mix];
real pp_mix_new[nsubjs, n_mix];
real pp_mix_var_new[nsubjs, n_mix];
real pp_w_new[nsubjs, n_mix];
real sig_mix_new[nsubjs, n_mix];

for (sub in 1:nsubjs) {
  for (m in 1:n_mix_te)  //prediction of up mixed range
  {
    // first integration
    if(X_mix_te[sub, m] < 1){
```

```
      p_mix_new[sub, m] = a_s[sub] + b_s[sub] * mean(X_s[sub,]);
    }
    else{
      p_mix_new[sub, m] = a_l[sub] + b_l[sub] * mean(X_l[sub,]);
    }
    sig_mix_new[sub, m] = mean(X_mix_te[sub,])*g_wf;  //global prior of up mixed block
    p_sig_mix_new[sub, m] = p_mix_new[sub, m] * p_wf;
    w_mix_new[sub, m] = sig_mix_new[sub, m]^2 / (sig_mix_new[sub, m]^2 + p_sig_mix_new[sub, m]^2);
    pp_mix_new[sub, m] = p_mix_new[sub, m] * w_mix_new[sub, m] + (1- w_mix_new[sub, m]) * mean(X_mix_te

    pp_mix_var_new[sub, m] = sig_mix_new[sub, m]^2* p_sig_mix_new[sub, m]^2/(sig_mix_new[sub, m]^2 + p_s
    pp_w_new[sub, m] = pp_mix_var_new[sub, m]  / (pp_mix_var_new[sub, m]  + X_mix_te[sub, m]^2* wf^2);
    ynew_mix[sub, m] = pp_w_new[sub, m] * X_mix_te[sub, m] + (1-pp_w_new[sub, m])*pp_mix_new[sub, m];

   }
}

}
'
```

predicte the parameters of Bayesian

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

```
funFitBayesianStanV4 <- function(data, rstanModel, filename){
  Bayparlist = {}
  PredYlist_mix  = {}
  expList <- unique(data$Exp)
  for (expName in expList) {
    subdata <- data %>% filter(valid > 0 & Exp == expName)
    subList <- unique(subdata$NSub)
    nsubjs <- length(subList)

    # reform the data set
    Y_s <- {}
    n_s <- 0
    X_s <- {}
    Y_l <- {}
    n_l <- 0
    X_l <- {}
    Y_mix <- {}
    n_mix <- 0
    X_mix <- {}
    X_mix_te <- {}
    n_mix_te <- {}
    testdat_mix <- {}
    list_Y_s <- list()
    list_X_s <- list()
    list_Y_l <- list()
    list_X_l <- list()
    list_Y_mix <- list()
    list_X_mix <- list()

    for(i in 1 : nsubjs){
```

```
  dat <- subdata %>% filter(NSub == i)
  data_s<- dat %>% filter(group == 1)  # short groups
  data_l <- dat %>% filter(group == 2)  # long groups
  data_mix <- dat %>% filter(group == 3)  # mixed groups
  list_Y_s[[1]] <- data.frame(Y_s)
  list_Y_s[[2]] <- data.frame(t(data_s$RP))
  Y_s <- rbind.fill(list_Y_s)
  list_X_s[[1]] <- data.frame(X_s)
  list_X_s[[2]] <- data.frame(t(data_s$targetDur))
  X_s <- rbind.fill(list_X_s)
  list_Y_l[[1]] <- data.frame(Y_l)
  list_Y_l[[2]] <- data.frame(t(data_l$RP))
  Y_l <- rbind.fill(list_Y_l)
  list_X_l[[1]] <- data.frame(X_l)
  list_X_l[[2]] <- data.frame(t(data_l$targetDur))
  X_l <- rbind.fill(list_X_l)
  list_Y_mix[[1]] <- data.frame(Y_mix)
  list_Y_mix[[2]] <- data.frame(t(data_mix$RP))
  Y_mix <- rbind.fill(list_Y_mix)
  list_X_mix[[1]] <- data.frame(X_mix)
  list_X_mix[[2]] <- data.frame(t(data_mix$targetDur))
  X_mix <- rbind.fill(list_X_mix)
  testdat_mix <- rbind.data.frame(testdat_mix, data.frame(data_mix[c('NSub','targetDur', 'RP','Exp'
}

  Y_s[is.na(Y_s)] = 0
  X_s[is.na(X_s)] = 0
  Y_l[is.na(Y_l)] = 0
  X_l[is.na(X_l)] = 0
  X_mix[is.na(X_mix)] = 0
  Y_mix[is.na(Y_mix)] = 0
  X_mix_te <- X_mix
  n_s <- ncol(X_s)
  n_l <- ncol(X_l)
  n_mix <-  ncol(X_mix)
  n_mix_te <- n_mix

  stan_data<- list("nsubjs"=nsubjs, "Y_s"=Y_s, "n_s"=n_s, "X_s"=X_s, "Y_l"= Y_l, "X_l"=X_l, "n_l" =


   # fit models
  subfit <- sampling(rstanModel, stan_data, chains =4)
  ## Launch shinystan and save the resulting shinystan object
  #sf_sso <- launch_shinystan(subfit)
  #rdspath <- paste0("reproduction_outlier/data/sso_", subNo,"_",expName)
  #save_rds(subfit, rdspath)

  # print(subfit)
  # plot(subfit)
  # stan_trace(subfit)
  # stan_trace(subfit, inc_warmup = TRUE)
  # pairs(subfit)
```

```
parameters <- c("a_s","b_s", "a_l","b_l", "sigma_s", "sigma_l", "sigma_mix", "wf", "p_wf", "g_wf"
fitpar <- summary(subfit, pars = parameters)$summary

# fitpar <- summary(subfit, pars = parameters)$summary
#list_of_draws <- rstan::extract(subfit, pars = parameters)
list_of_draws <- rstan::extract(subfit, pars = parameters)
 # Extracting the parameters
a_s_list =  list_of_draws$a_s
b_s_list = list_of_draws$b_s
a_l_list = list_of_draws$a_l
b_l_list = list_of_draws$b_l
wf = mean(list_of_draws$wf)
p_wf = mean(list_of_draws$p_wf)
g_wf = mean(list_of_draws$g_wf)
sigma_s = mean(list_of_draws$sigma_s)
sigma_l = mean(list_of_draws$sigma_l)
sigma_mix = mean(list_of_draws$sigma_mix)

a_s = c(rep(0, nsubjs))
b_s = c(rep(0, nsubjs))
a_l = c(rep(0, nsubjs))
b_l = c(rep(0, nsubjs))
ynew_mix_lsit <- list_of_draws$ynew_mix
ynew_mix = matrix(rep(0, nsubjs*length(ynew_mix_lsit[1,1,])), nrow = length(ynew_mix_lsit[1,1,]))

# Constructing MAP-estimates of parameters
for (i in 1: nsubjs)
{
  a_s[i] = density(a_s_list[,i])$x[which(density(a_s_list[,i])$y ==  max(density(a_s_list[,i])$y))
  b_s[i] = density(b_s_list[,i])$x[which(density(b_s_list[,i])$y ==  max(density(b_s_list[,i])$y))
  a_l[i] = density(a_l_list[,i])$x[which(density(a_l_list[,i])$y ==  max(density(a_l_list[,i])$y))
  b_l[i] = density(b_l_list[,i])$x[which(density(b_l_list[,i])$y ==  max(density(b_l_list[,i])$y))
  for (j in 1: length(ynew_mix_lsit[1,1,])){
    if(X_mix_te[i,j] >0){
                ynew_mix[j,i] =  density(ynew_mix_lsit[,i,j])$x[which(density(ynew_mix_lsit[,i,j]))
    }
  }
}

t_X_mix_te <- as.data.frame(t(X_mix_te))
colnames(t_X_mix_te) <- paste0('sub', 1:nsubjs)
predY <- as.data.frame(t_X_mix_te) %>% gather(NSub, targetDur, sub1:sub16) %>% separate(NSub, c("k
predY$key = NULL

t_Y_mix <- as.data.frame(t(Y_mix))
colnames(t_Y_mix) <- paste0('sub', 1:nsubjs)
predResult <- data.frame(t_Y_mix) %>% gather(NSub, RP, paste0('sub', 1:nsubjs))
predResult <- cbind(predY, predResult$RP)

colnames(ynew_mix) <- paste0('sub', 1:nsubjs)
ynew_mix <- data.frame(ynew_mix) %>% gather(NSub, predY, paste0('sub', 1:nsubjs))

predResult <- cbind(predResult, ynew_mix$predY)
```

```
    colnames(predResult) <- c("NSub", "targetDur","RP", "predY")
    predResult$Exp = expName
    PredYlist_mix <- rbind2(PredYlist_mix, predResult)

    write.csv(predResult, file = paste0(modelResultPath, "/data/predResult_", expName, "_", filename,



    # merge the parameters
    Baypar <- data.frame("a_s" = a_s, "b_s" = b_s, "a_l" = a_l, "b_l" = b_l,  "sigma_s" =sigma_s, "sig
    Baypar$Exp = expName
    Baypar$wf = wf
    Baypar$p_wf = p_wf
    Baypar$g_wf = g_wf

    Bayparlist <- rbind2(Bayparlist, Baypar)
    write.csv(Baypar, file = paste0(modelResultPath, "/data/Baypar_", expName, "_", filename, ".csv"))
  }

  write.csv(Bayparlist, file = paste0(modelResultPath, "/data/Bayparlist_Stan_", filename,".csv"))
  write.csv(PredYlist_mix, file = paste0(modelResultPath, "/data/PredY_mix_Stan_", filename,".csv"))

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

run the model

run H3

```
# compile models
if (runModels){
  stanmodeH3 <- stan_model(model_code = stancodeH3, model_name="stanmodelH3")
}
```

run H4

display the model restults

Merge the Result data

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

```
needpreprocess=1
versionlist =c('Version5')
modellist = c('H3', 'H4')

if (needpreprocess == 1){

  predY_mix_filename <- paste0("PredY_mix_Stan_", modellist, ".csv")
  BayParlist_filename <- paste0("Bayparlist_Stan_", modellist, ".csv")
  mergeData(predY_mix_filename, 'predY_mix', versionlist)
  mergeData(BayParlist_filename, 'Bayparlist', versionlist)
}
```

load the model result data

Analysis on the Rstan model parameters

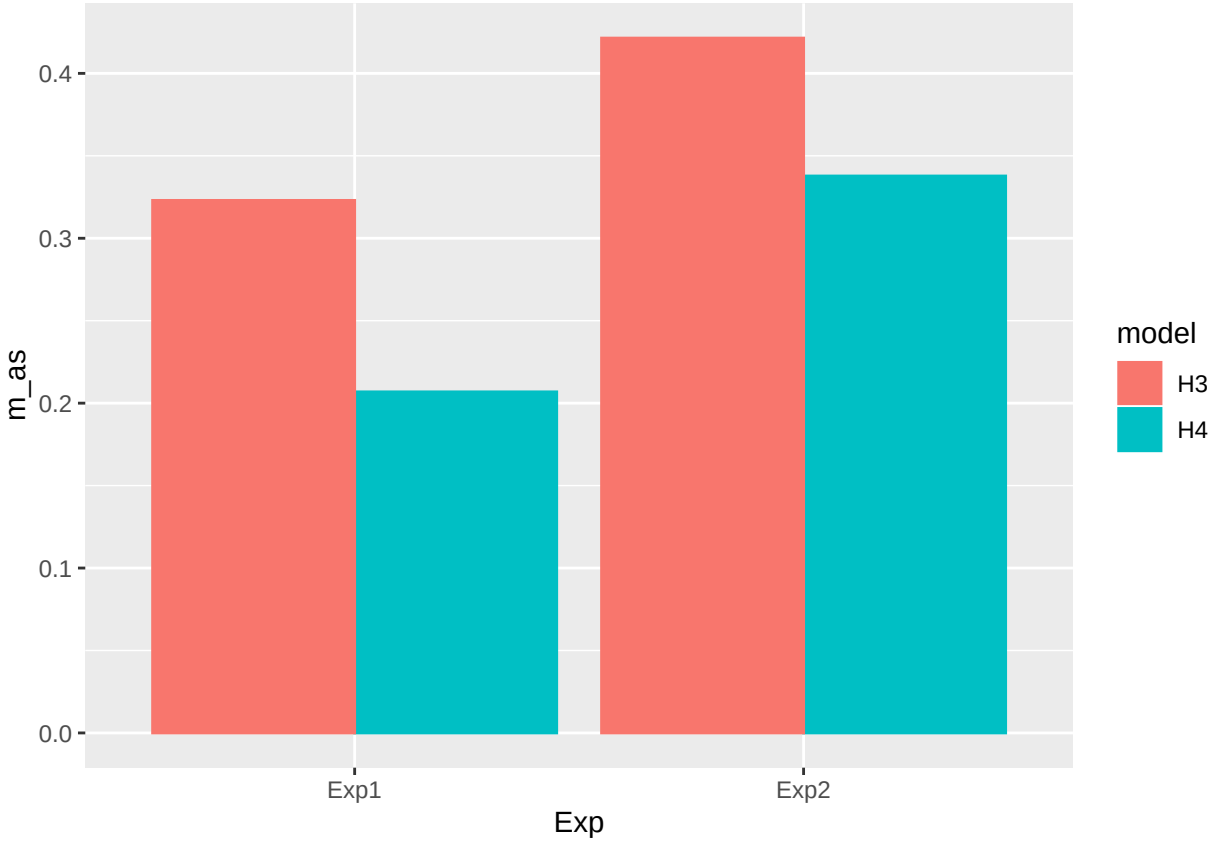
Parameters

```
 # AllDat_Bayparlist$model <- factor(AllDat_Bayparlist$model, labels = c( "Hierarchical local-global mo
m_Baypar <- dplyr::group_by(AllDat_Bayparlist, Exp, model, version) %>%
  dplyr::summarize(m_as = mean(a_s), m_al = mean(a_l),
            m_bs = mean(b_s), m_bl = mean(b_l),
            m_wf = mean(wf), m_gwf = mean(g_wf), m_pwf = mean(g_wf) )
m_Baypar
```

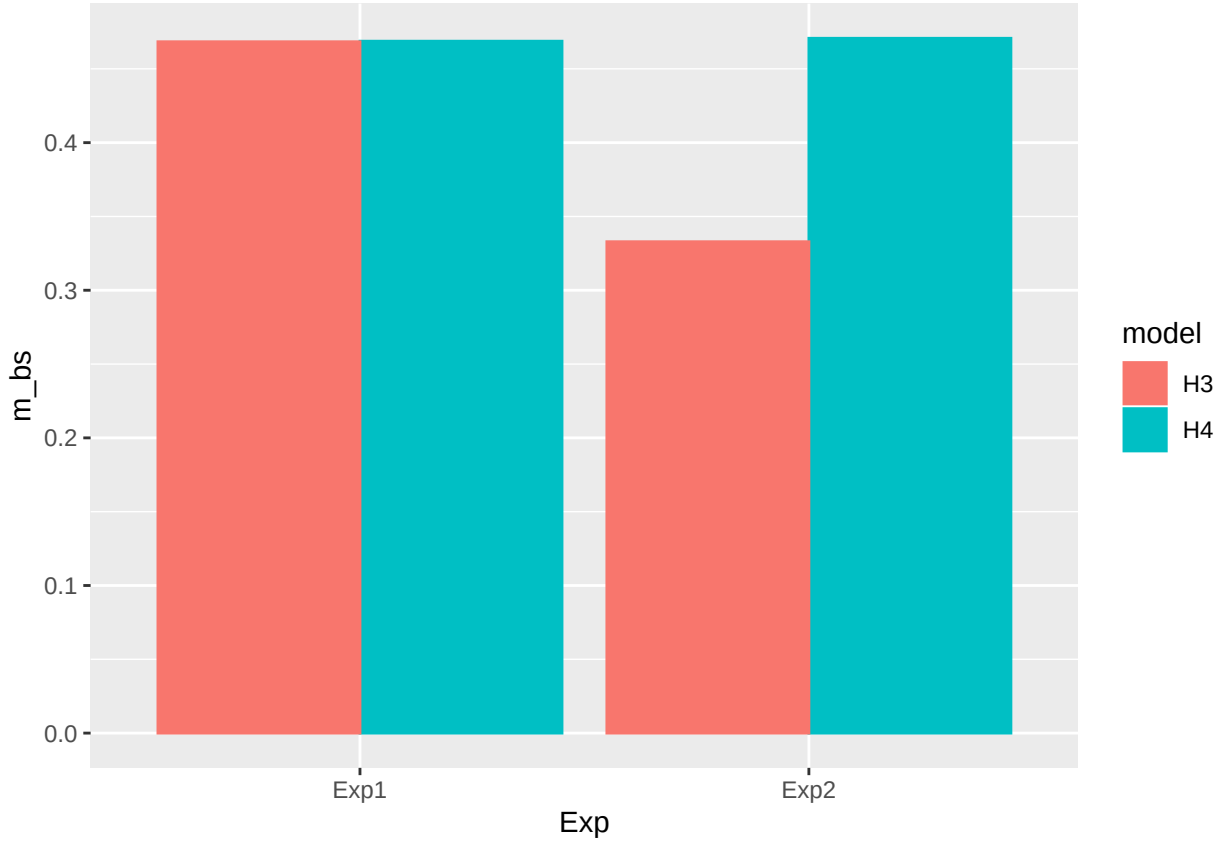
```
## # A tibble: 4 x 10
## # Groups:   Exp, model [4]
##   Exp   model version   m_as   m_al  m_bs  m_bl     m_wf    m_gwf    m_pwf
##   <chr> <chr> <chr>    <dbl>  <dbl> <dbl> <dbl>    <dbl>    <dbl>    <dbl>
## 1 Exp1  H3    Version5 0.323  0.433 0.468 0.715 0.00900  0.150    0.150
## 2 Exp1  H4    Version5 0.207 -0.274 0.469 0.771 0.000225 0.346    0.346
## 3 Exp2  H3    Version5 0.421  0.516 0.333 0.618 0.0874   0.0105   0.0105
## 4 Exp2  H4    Version5 0.338  0.204 0.471 0.617 0.0115   0.000573 0.000573
```

wf in models

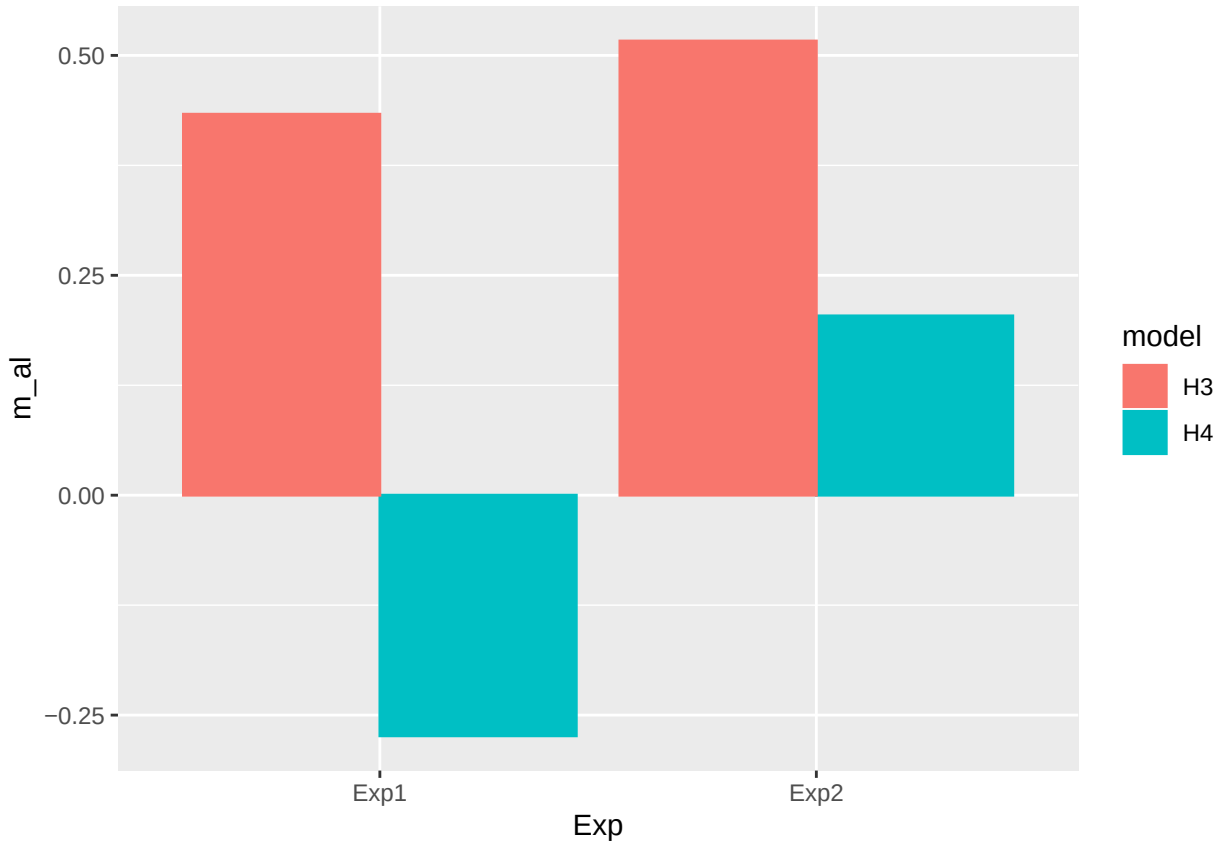
```
ggplot(m_Baypar, aes(x = Exp, y = m_as, color = model, fill = model, group = model)) +
    geom_bar(stat = "identity",
             position = position_dodge())
```



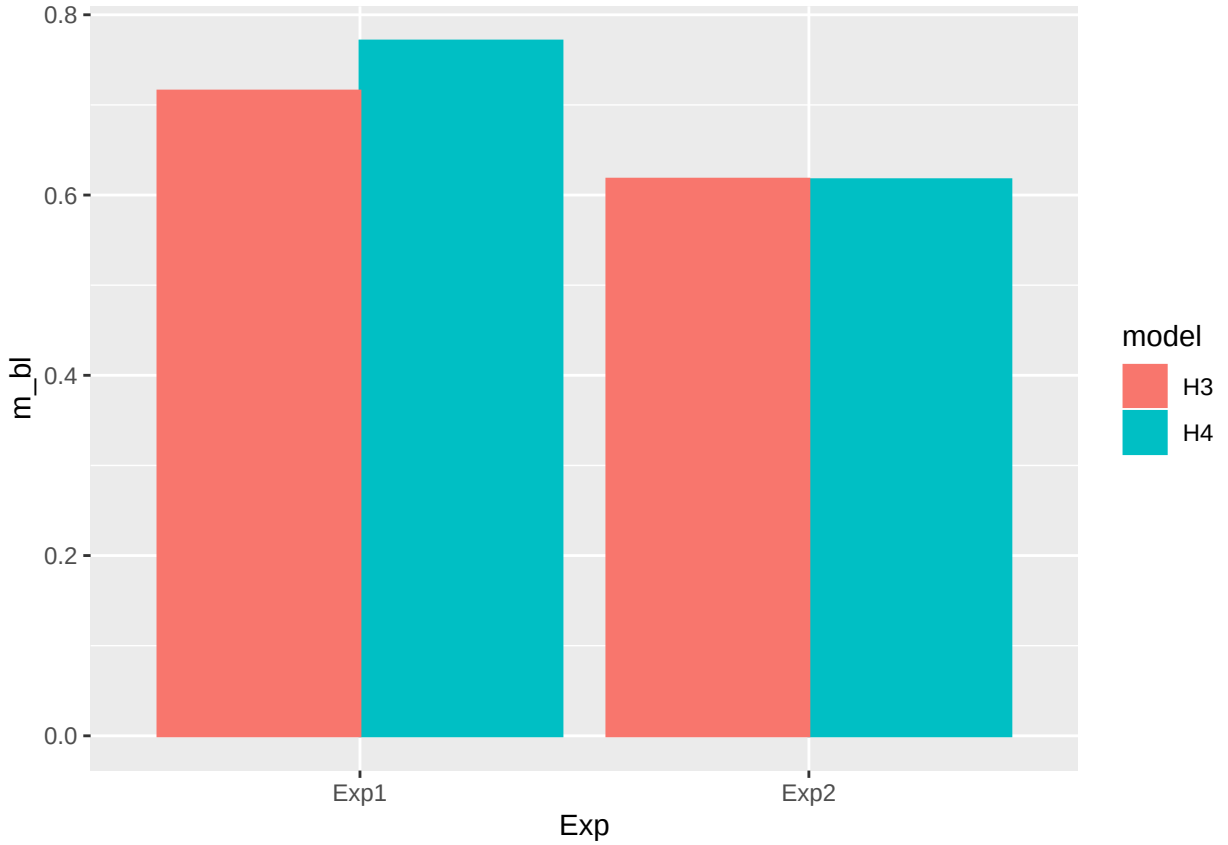
```
ggplot(m_Baypar, aes(x = Exp, y = m_bs, color = model, fill = model, group = model)) +
    geom_bar(stat = "identity",
             position = position_dodge())
```



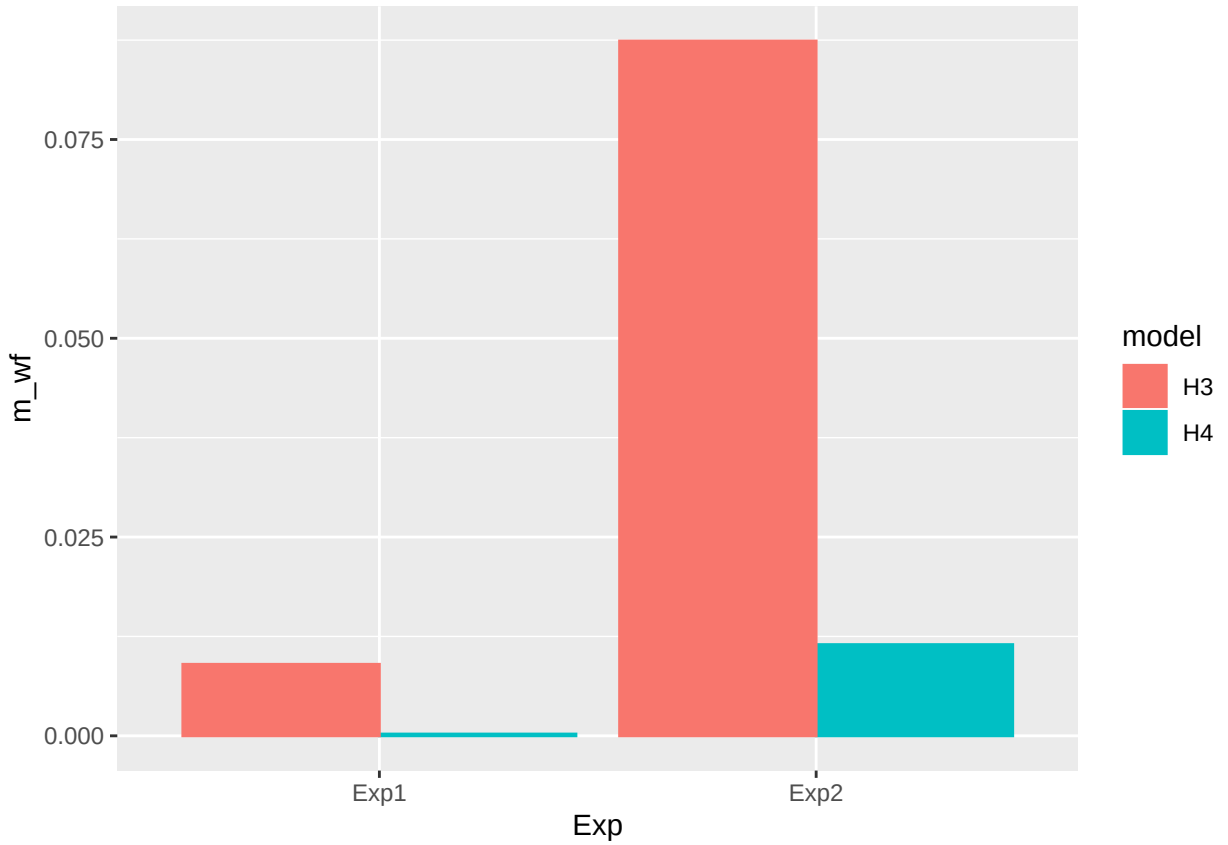

```
ggplot(m_Baypar, aes(x = Exp, y = m_al, color = model, fill = model, group = model)) +
    geom_bar(stat = "identity",
             position = position_dodge())
```



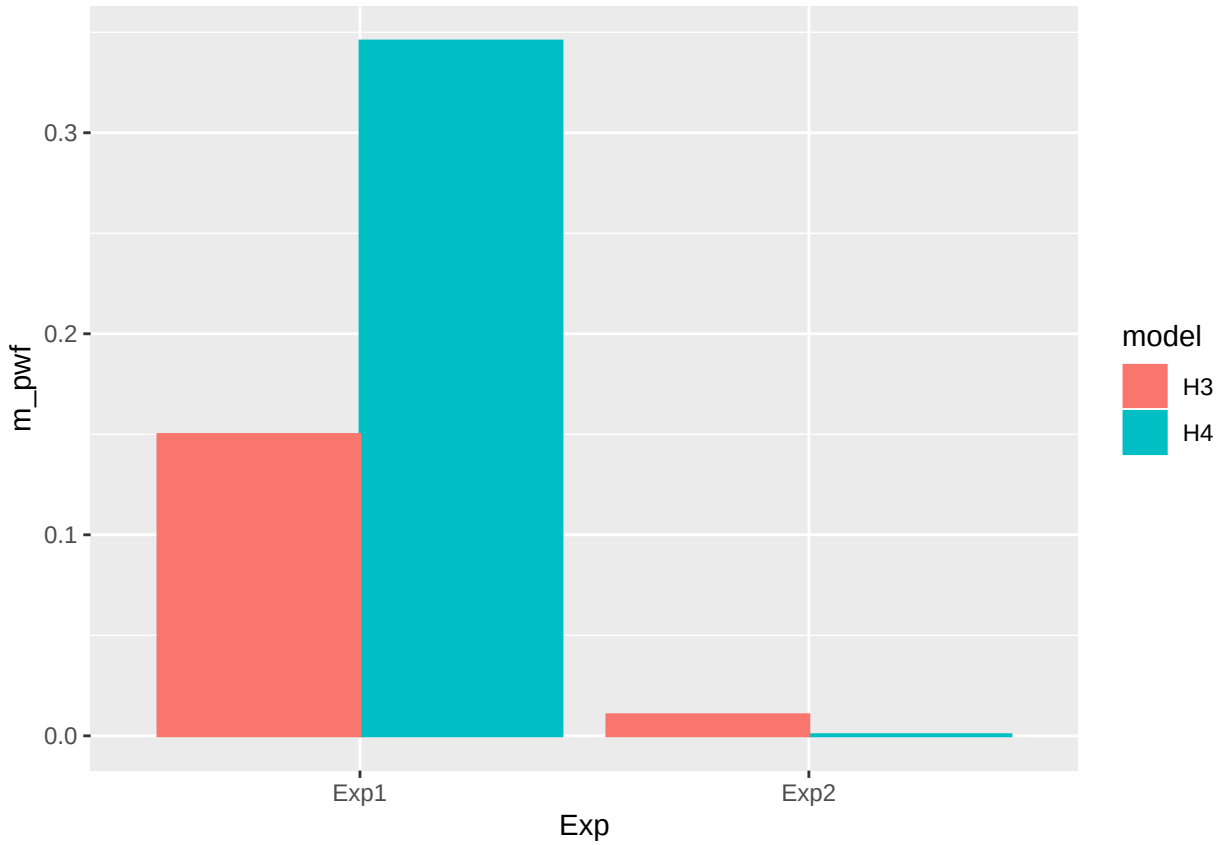

```
ggplot(m_Baypar, aes(x = Exp, y = m_bl, color = model, fill = model, group = model)) +
    geom_bar(stat = "identity",
             position = position_dodge())
```



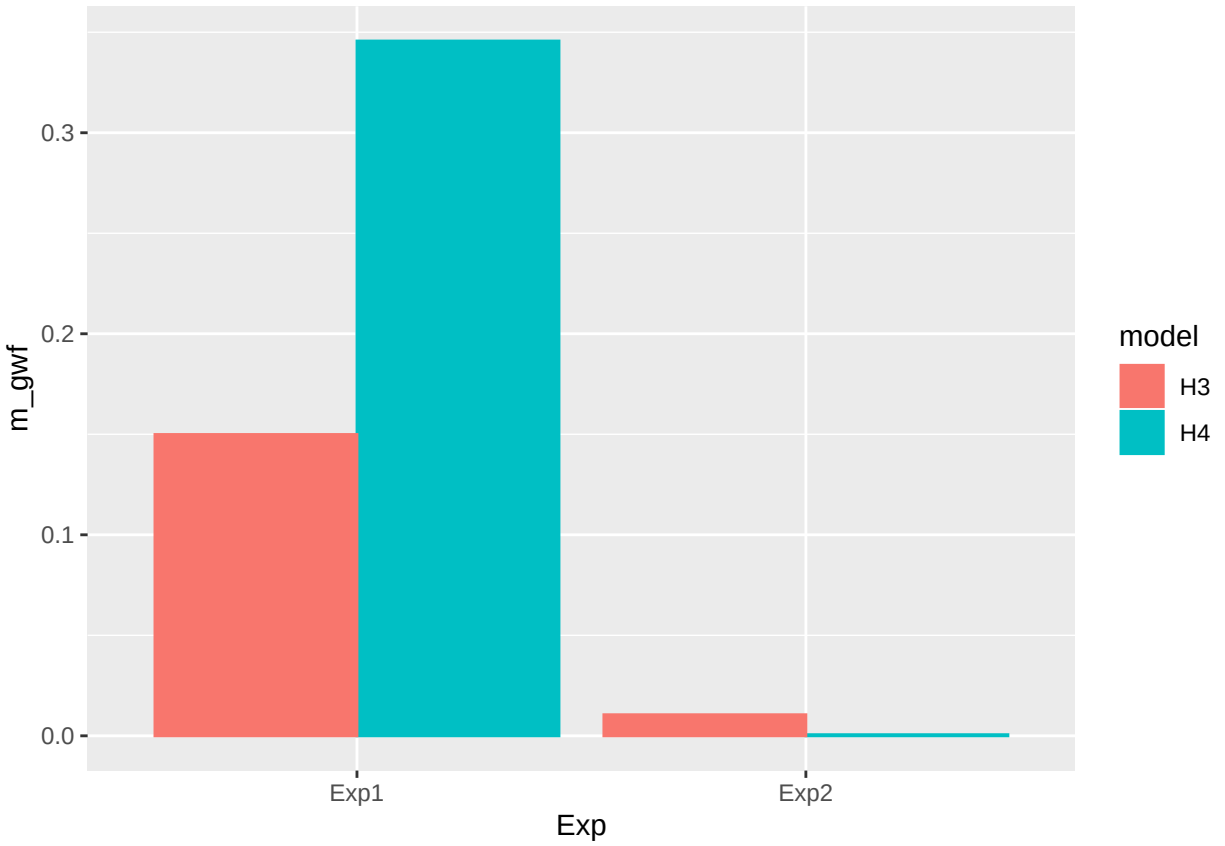

```
ggplot(m_Baypar, aes(x = Exp, y = m_wf, color = model, fill = model, group = model)) +
    geom_bar(stat = "identity",
             position = position_dodge())
```




```
ggplot(m_Baypar, aes(x = Exp, y = m_pwf, color = model, fill = model, group = model)) +
    geom_bar(stat = "identity",
             position = position_dodge())
```




```
ggplot(m_Baypar, aes(x = Exp, y = m_gwf, color = model, fill = model, group = model)) +
    geom_bar(stat = "identity",
             position = position_dodge())
```



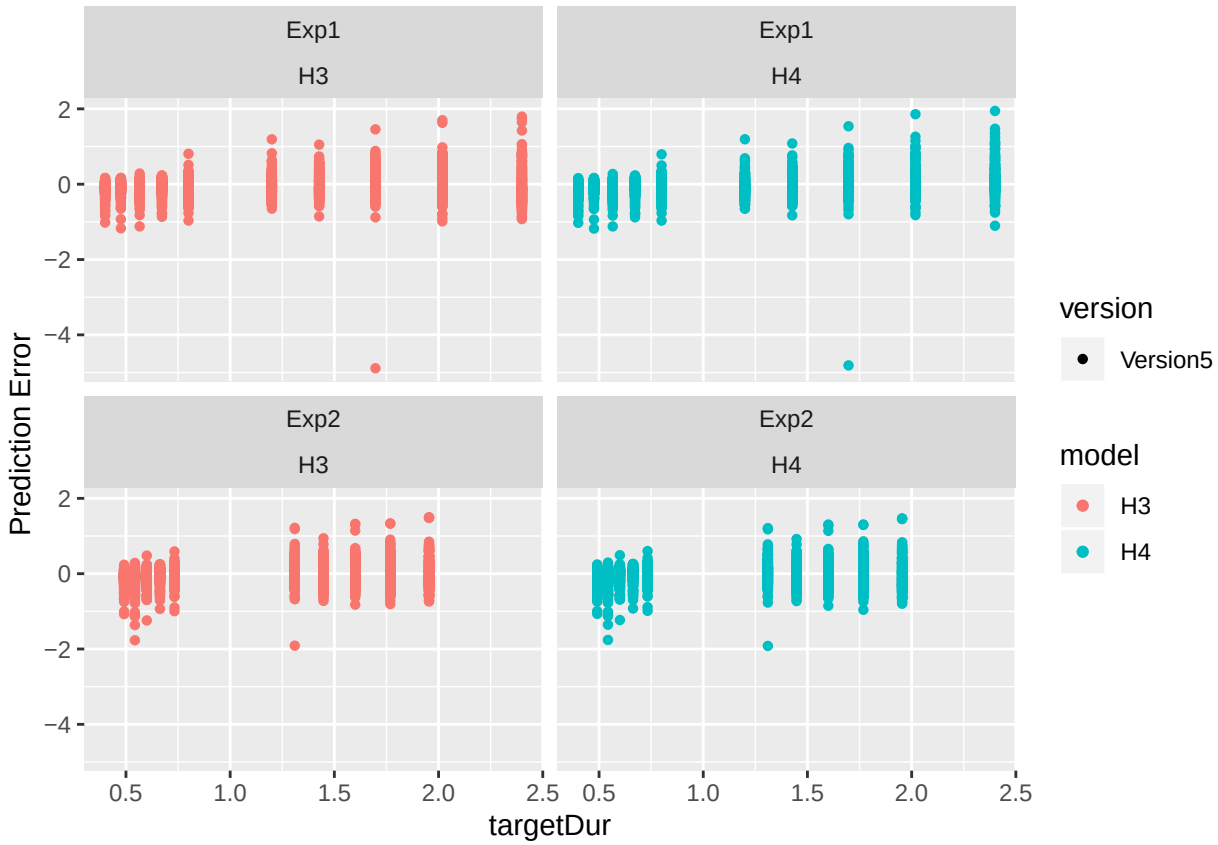
Prediction results (mixed block)

load the oberserved and predicted Y

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

plot model results in scatter plot

```
fig1 = ggplot(AllDat_predY_mix, aes(targetDur, err, color = model, shape = version)) +
  geom_point() +
  ylab('Prediction Error')+
  facet_wrap(Exp~model)

fig1
```



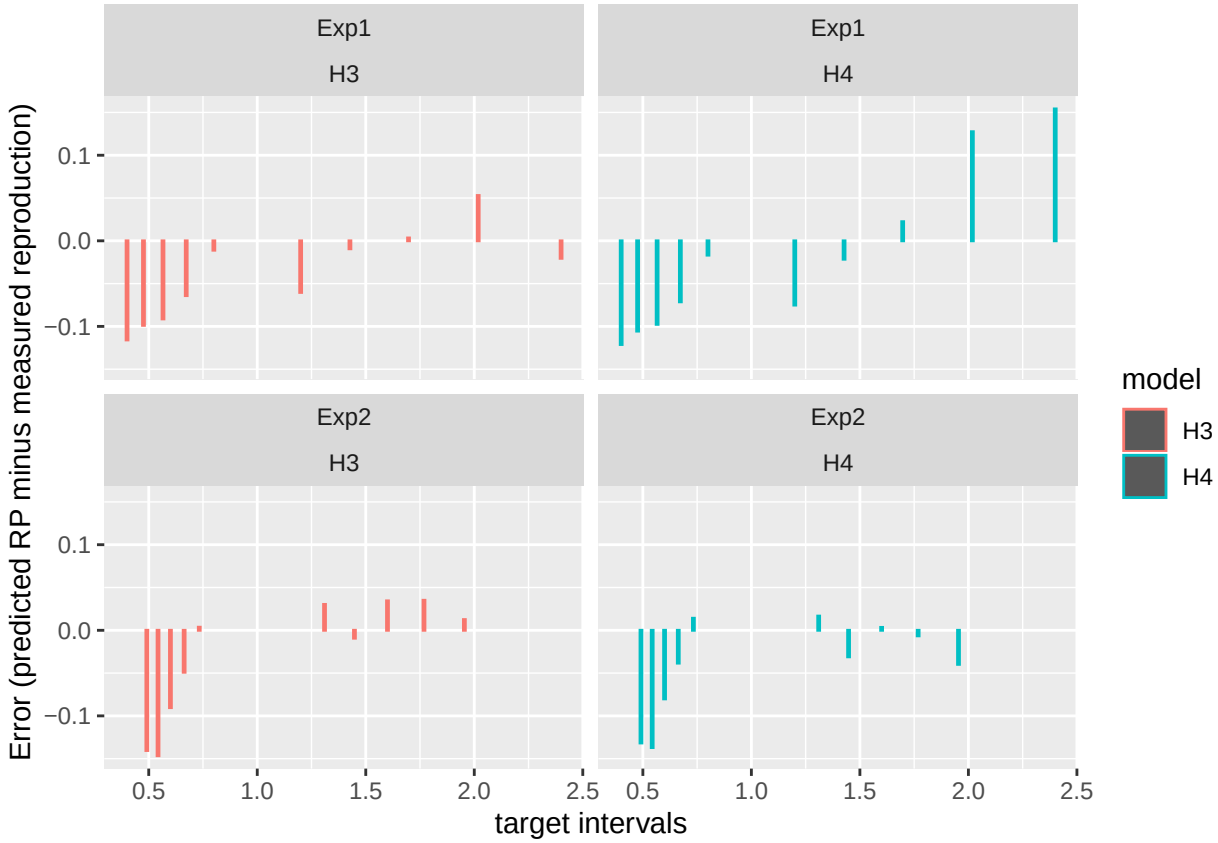

```
predY_mix <- dplyr::group_by(AllDat_predY_mix, targetDur, Exp, NSub, model,version) %>%
    dplyr::summarize(m_RP = mean(RP),
                     n = n(), m_predY =mean(predY),
                     ) %>%
  dplyr::group_by(targetDur, Exp, model,version) %>%
  dplyr::summarize(
    n = n(),
    m_m_predY = 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)
  )

predY_mix$m_rpErr = predY_mix$m_m_predY - predY_mix$m_m_RP
predY_mix$m_relativeErr = predY_mix$m_rpErr / predY_mix$targetDur
predY_mix
```

```
## # A tibble: 40 x 11
## # Groups:   targetDur, Exp, model [40]
##    targetDur Exp   model version     n m_m_predY se_m_predY m_m_RP se_m_RP
##        <dbl> <chr> <chr> <chr>   <int>     <dbl>      <dbl>  <dbl>   <dbl>
##  1     0.4   Exp1  H3    Versio~    16     0.405  0.000829   0.521  0.0169
##  2     0.4   Exp1  H4    Versio~    16     0.400  0.0000444  0.521  0.0169
##  3     0.476 Exp1  H3    Versio~    16     0.482  0.00104    0.581  0.0241
##  4     0.476 Exp1  H4    Versio~    16     0.476  0.0000697  0.581  0.0241
##  5     0.491 Exp2  H3    Versio~    16     0.522  0.000136   0.663  0.0321
##  6     0.491 Exp2  H4    Versio~    16     0.531  0.000157   0.663  0.0321
```

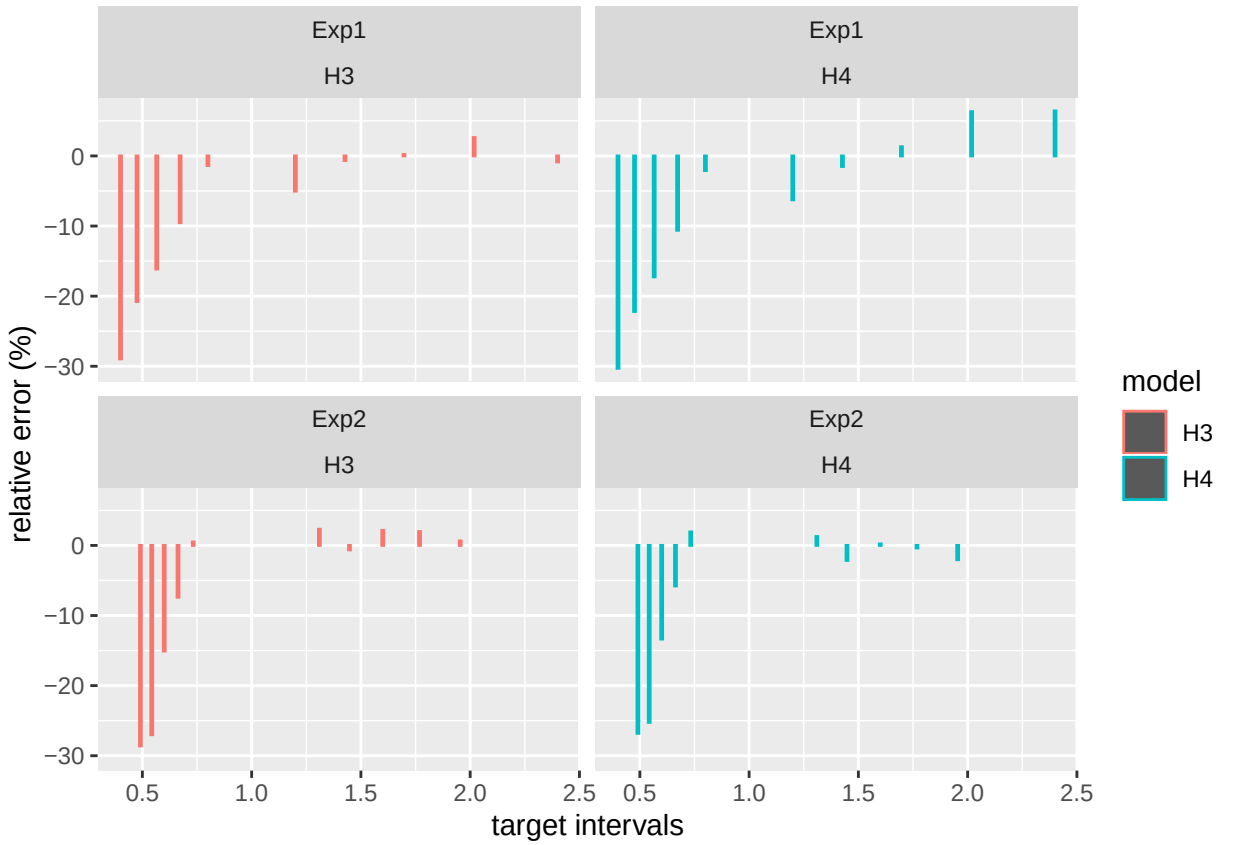
```
##  7     0.543 Exp2  H3    Versio~    16     0.577  0.000115   0.724  0.0433
##  8     0.543 Exp2  H4    Versio~    16     0.587  0.000130   0.724  0.0433
##  9     0.566 Exp1  H3    Versio~    16     0.572  0.00147    0.663  0.0190
## 10     0.566 Exp1  H4    Versio~    16     0.566  0.000115   0.663  0.0190
## # ... with 30 more rows, and 2 more variables: m_rpErr <dbl>,
## #   m_relativeErr <dbl>
```

The predication of mixed block

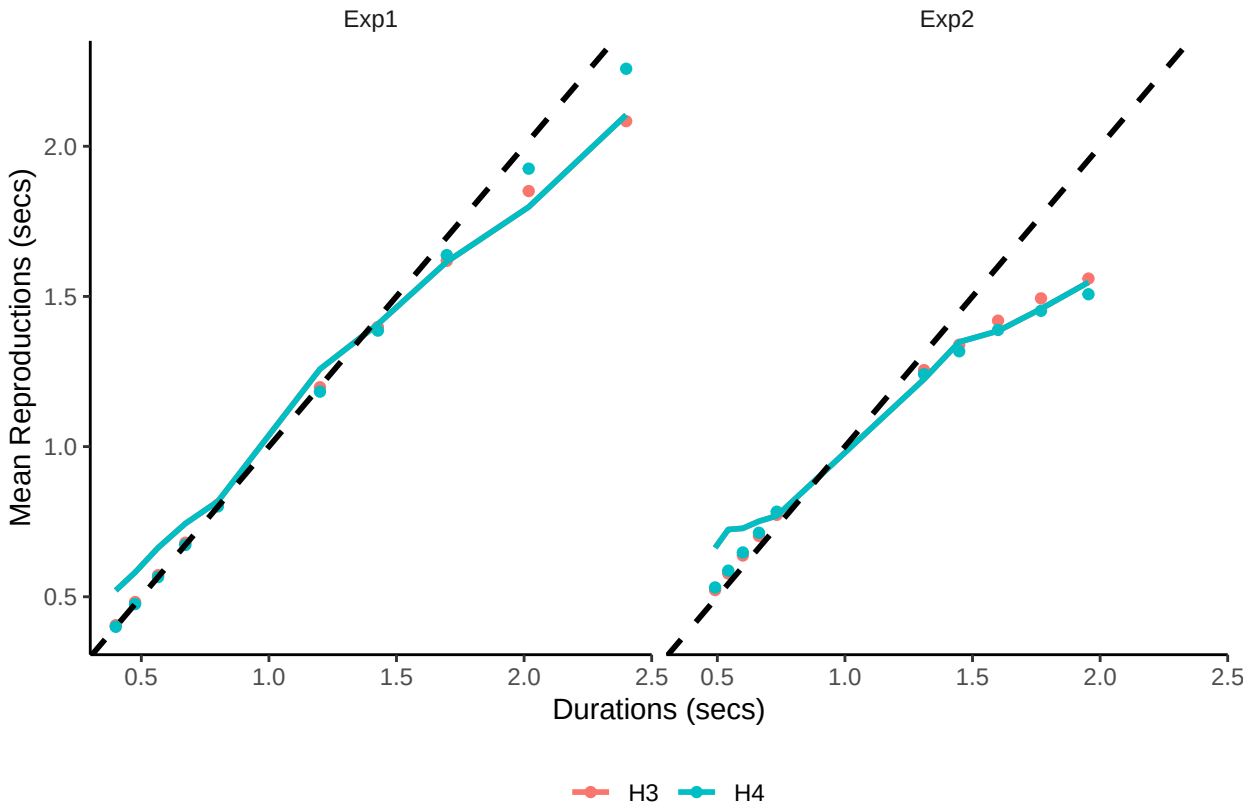
```
#plot Error in mixed blocks
  ggplot(data=predY_mix, aes(x= targetDur, y=m_rpErr, group = model, color= model)) +
    geom_bar(stat="identity")+
    labs(x="target intervals", y="Error (predicted RP minus measured reproduction)")+
  facet_wrap(Exp~model)
```




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

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

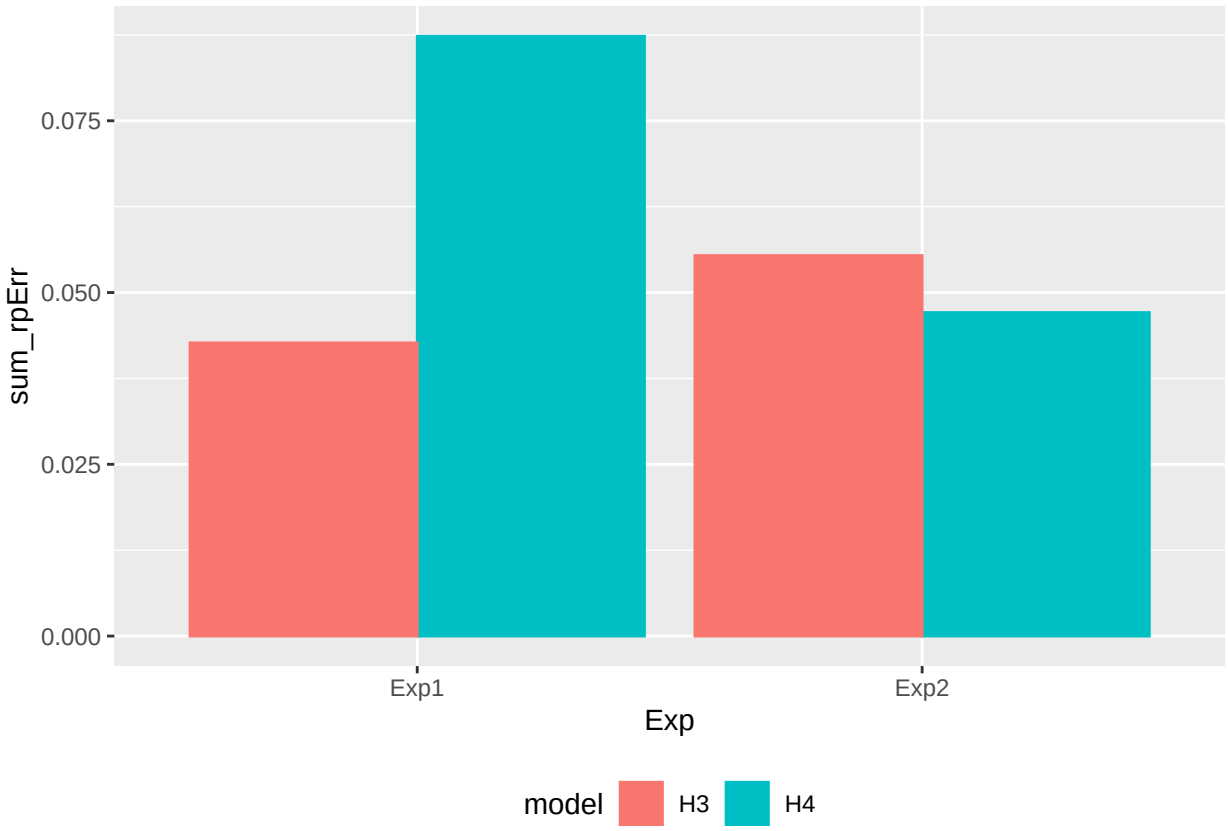



```
#plot the average of the predicted Y under the mixed condition
fig_mPredY_mix = ggplot(predY_mix) +
  geom_point(aes(targetDur, m_m_predY, group = model, color = model)) +
  geom_line(aes(targetDur, m_m_RP, group = model, color = model),  size = 1) +
  #geom_errorbar(aes(ymin = m_m_predY-se_m_predY, 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
fig_mPredY_mix
```




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

```
predY_mix$rpErr_squared <- predY_mix$m_rpErr^2
m_predY_Err <- dplyr::group_by(predY_mix, Exp, model,version) %>%
  dplyr::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(legend.position="bottom")

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

fig_rpErr_model
```


0.075
0.050
0.025
0.000
sum_rpErr
Exp1
Exp2
Exp
model
H3
H4