ModelReportBaseline2

Fiona Zhu

18/03/2020

Load the packages

```r
library(tidyverse)
```

```
## -- Attaching packages -------------------------------------------------------------------------------

## v ggplot2 3.3.0     v purrr   0.3.3
## v tibble  2.1.3     v dplyr   0.8.5
## v tidyr   1.0.2     v stringr 1.4.0
## v readr   1.3.1     v forcats 0.5.0

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

```r
library(rstan)
```

```
## Loading required package: StanHeaders

## rstan (Version 2.19.3, 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
```

```r
library(readr)
```

load all experiment data

```r
AllExpData <- read_csv(paste0("data/AllExpData.csv"))
```

```
## Warning: Missing column names filled in: 'X1' [1]

## Parsed with column specification:
## cols(
##   X1 = col_double(),
##   X.1 = col_double(),
##   X = col_double(),
##   trialnum = col_double(),
##   group = col_double(),
##   interval = col_double(),
##   targetDur = col_double(),
##   phyTargetDur = col_double(),
##   RP = col_double(),
##   NSub = col_double(),
##   NT = col_double(),
##   valid = col_double(),
##   Exp = col_character(),
##   NB = col_double(),
##   repError = col_double()
## )
```

```
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, '/Baseline2')
```

Define the Rstan models and functions to plot

Baseline: Model for short and long groups

```
stancodeBaseline <- '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 xmean[3];  // mean of the target duration in each group
}

parameters {
//hyperparameters
real<lower=0> p_wf;   //Weber Fraction of local prior
real<lower=0> wf;     //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
real<lower=0> sig_m_square;   //square of sigma of distribution of motor noise
//vector[n_s] MotorNoise_s; //Motor noise of short group
//vector[n_l] MotorNoise_l; //Motor noise of long group
//vector[n_mix] MotorNoise_mix; //Motor noise of mix group

//vector[n_mix] Prior_mix; //internal prior of mix group
}
```

```
model {
real  w_s[n_s];   //  weight of stimuli in short group
real  w_l[n_l];   //  weight of stimuli in short group
//real  prior_s[n_s]; // mean of internal prior of short group
//real  prior_l[n_l]; // mean of internal prior of long group

//hyperpriors
mu_s ~ normal(xmean[1], p_wf^2 * xmean[1]^2);  // mean prior of short group
mu_l ~ normal(xmean[2], p_wf^2 * xmean[2]^2);   // mean prior of long group



//short groups
for (i in 1:n_s)
{
    w_s[i] = (mu_s[i]^2 * p_wf^2)/(mu_s[i]^2 * p_wf^2+ X_s[i]^2*wf^2);    // weight of current stimuls
    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^2 * X_s[i]
}

//long groups
for (i in 1:n_l)
{
    w_l[i] = (mu_l[i]^2 * p_wf^2)/(mu_l[i]^2 * p_wf^2+ X_l[i]^2*wf^2);  // weight of current stimuls
    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^2 * X_l[i]^2*wf^2) / mu_l[i]^2 * p_wf^2 + X_l[i]^2*wf^2);

}
}


generated quantities {
  vector[n_s] ynew_s;
  vector[n_l] ynew_l;
  vector[n_mix] ynew_mix;
  vector[n_s] mu_s_new;   // mean of internal prior of short group
  vector[n_l] mu_l_new;   // mean of internal prior of long group
  vector[n_mix] mu_mix_new;   // mean of internal prior of mix group

real  w_new_s[n_s];   //  weight of stimuli in short group
real  w_new_l[n_l];   //  weight of stimuli in long group
real  w_new_mix[n_mix];   //  weight of stimuli in mix group


  for (i in 1:n_s)  //prediction of short group
  {
    mu_s_new[i] = normal_rng(xmean[1], p_wf^2 * xmean[1]^2);  // mean prior of short group
    w_new_s[i] = (mu_s_new[i]^2 * p_wf^2)/(mu_s_new[i]^2 * p_wf^2+ X_s[i]^2*wf^2);    // weight of curre
    ynew_s[i] = normal_rng(mu_s_new[i] *  w_new_s[i]+ (1- w_new_s[i])* X_s[i], sig_m_square + (mu_s_new
  }

  for (i in 1:n_l)   //prediction of long group
  {
     mu_l_new[i] = normal_rng(xmean[2], p_wf^2 * xmean[2]^2);   // mean prior of long group
     w_new_l[i] = (mu_l_new[i]^2 * p_wf^2)/(mu_l_new[i]^2 * p_wf^2+ X_l[i]^2*wf^2);  // weight of curren
    ynew_l[i] = normal_rng(mu_l_new[i] *  w_new_l[i]+ (1- w_new_l[i])* X_l[i],
    sig_m_square + (mu_l_new[i]^2 * p_wf^2 * X_l[i]^2*wf^2) / mu_l_new[i]^2 * p_wf^2 + X_l[i]^2*wf^2);
  }

  for (i in 1:n_mix)   //prediction of mix group
  {
    if(X_mix[i] >= 1) {

    mu_mix_new[i] = normal_rng(xmean[2], p_wf^2 * xmean[2]^2);   // mean prior of long group
    w_new_mix[i] = (mu_mix_new[i]^2 * p_wf^2)/(mu_mix_new[i]^2 * p_wf^2+ X_mix[i]^2*wf^2);  // weight o
    ynew_mix[i] = normal_rng(mu_mix_new[i] *  w_new_mix[i]+ (1- w_new_mix[i])* X_mix[i],
    sig_m_square + (mu_mix_new[i]^2 * p_wf^2 * X_mix[i]^2*wf^2) / mu_mix_new[i]^2 * p_wf^2 + X_mix[i]^2

    }
   else{
        mu_mix_new[i] = normal_rng(xmean[1], p_wf^2 * xmean[1]^2);  // mean prior of short group
         w_new_mix[i] = (mu_mix_new[i]^2 * p_wf^2)/(mu_mix_new[i]^2 * p_wf^2+ X_mix[i]^2*wf^2);    // w
    ynew_mix[i] = normal_rng(mu_mix_new[i] *  w_new_mix[i]+ (1- w_new_mix[i])* X_mix[i], sig_m_square +
    }

  }

}

'

# compile models
stanmodeBaseline <- stan_model(model_code = stancodeBaseline, model_name="Baseline")
```

```
## Trying to compile a simple C file
```

```
## Running \
##   /mnt/shared/spack-v0.13/opt/spack/linux-debian10-skylake_avx512/gcc-8.3.0/r-3.6.1-a53piuzvsjwnz2iq
##   CMD SHLIB foo.c
## /bin/gcc-8 -I"/mnt/shared/spack-v0.13/opt/spack/linux-debian10-skylake_avx512/gcc-8.3.0/r-3.6.1-a53p
## In file included from /dss/dsshome1/lxc01/ru24tub2/R/debian10/3.6/RcppEigen/include/Eigen/Core:88,
##                  from /dss/dsshome1/lxc01/ru24tub2/R/debian10/3.6/RcppEigen/include/Eigen/Dense:1,
##                  from /dss/dsshome1/lxc01/ru24tub2/R/debian10/3.6/StanHeaders/include/stan/math/prim
##                  from <command-line>:
## /dss/dsshome1/lxc01/ru24tub2/R/debian10/3.6/RcppEigen/include/Eigen/src/Core/util/Macros.h:613:1: er
##  namespace Eigen {
##  ^~~~~~~~~
## /dss/dsshome1/lxc01/ru24tub2/R/debian10/3.6/RcppEigen/include/Eigen/src/Core/util/Macros.h:613:17: e
##  namespace Eigen {
##                  ^
## In file included from /dss/dsshome1/lxc01/ru24tub2/R/debian10/3.6/RcppEigen/include/Eigen/Dense:1,
##                  from /dss/dsshome1/lxc01/ru24tub2/R/debian10/3.6/StanHeaders/include/stan/math/prim
##                  from <command-line>:
## /dss/dsshome1/lxc01/ru24tub2/R/debian10/3.6/RcppEigen/include/Eigen/Core:96:10: fatal error: complex
##  #include <complex>
##           ^~~~~~~~~
## compilation terminated.
## make: *** [/mnt/shared/spack-v0.13/opt/spack/linux-debian10-skylake_avx512/gcc-8.3.0/r-3.6.1-a53piuz
```

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

```
funFitBaseLineStan <- 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$targetDur, n_mix = n_mix, "xmean" =xmean$targetMean)  #data pas

      # fit models
      subfit <- sampling(rstanModel, stan_data, chains = 4, iter = 2000)

      #parameters <- c("a_s", "b_s", "a_l", "b_l", "p_wf","ynew_s","ynew_l", "ynew_mix")
      parameters <- c("p_wf","wf", "sig_m_square", "w_new_l", "w_new_s", "w_new_mix", "ynew_s","ynew_l"
      fitpar <- summary(subfit, pars = parameters)$summary



      list_of_draws <- rstan::extract(subfit, pars = parameters)
      p_wf =  mean(list_of_draws$p_wf)
      wf =  mean(list_of_draws$wf)
      sig_m_square =  mean(list_of_draws$sig_m_square)

      ynew_s_list <- list_of_draws$ynew_s
      w_new_s_list <- list_of_draws$w_new_s
      pred_y_s <- {}
      w_new_s <- {}
      for (n in 1:n_s){
        pred_y_s[n] <-  mean(ynew_s_list[,n] )
        w_new_s[n] <-  mean(w_new_s_list[,n] )
```

```r
      }
      PredY_s_list$w = w_new_s
      PredY_s_list$predY = pred_y_s
      PredYlist_s <- rbind2(PredYlist_s, PredY_s_list)

      pred_y_l <- {}
      w_new_l <- {}
      ynew_l_list <- list_of_draws$ynew_l
      w_new_l_list <- list_of_draws$w_new_l
      for (n in 1:n_l){
        pred_y_l[n] <-  mean(ynew_l_list[,n] )
         w_new_l[n] <-  mean(w_new_l_list[,n] )
      }
      PredY_l_list$predY = pred_y_l
      PredY_l_list$w = w_new_l
      PredYlist_l <- rbind2(PredYlist_l, PredY_l_list)

      pred_y_mix <- {}
      w_new_mix <- {}
      ynew_mix_list <- list_of_draws$ynew_mix
      w_new_mix_list <- list_of_draws$w_new_mix
      for (n in 1:n_mix){
        pred_y_mix[n] <-  mean(ynew_mix_list[,n] )
        w_new_mix[n] <-  mean(w_new_mix_list[,n] )
      }
      PredY_mix_list$predY = pred_y_mix
      PredY_mix_list$w = w_new_mix
      PredYlist_mix <- rbind2(PredYlist_mix, PredY_mix_list)

      Baypar = data.frame(
        Nsub = subNo,
        Exp = expName,
        p_wf = p_wf,
        wf = wf,
        sig_m_square = sig_m_square
      )
      Bayparlist <- rbind2(Bayparlist, Baypar)
    }
  }
  write.csv(Bayparlist, file = paste0(modelResultPath, "/BaseLine_parlist_Stan.csv"))
  write.csv(PredYlist_s, file = paste0(modelResultPath, "/PredY_s_", filename,".csv"))
  write.csv(PredYlist_l, file = paste0(modelResultPath, "/PredY_l_", filename,".csv"))
  write.csv(PredYlist_mix, file = paste0(modelResultPath, "/PredY_mix_", filename,".csv"))

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

run Baseline RStan Models

display the model restults

load the model result data

```
PredY_l_Baseline <- read_csv(paste0(modelResultPath, "/PredY_l_Baseline2.csv"))
```

```
## Warning: Missing column names filled in: 'X1' [1]
```

```
## Parsed with column specification:
## cols(
##   X1 = col_double(),
##   NSub = col_double(),
##   targetDur = col_double(),
##   RP = col_double(),
##   Exp = col_character(),
##   group = col_double(),
##   predY = col_double(),
##   w = col_double()
## )
```

```
PredY_s_Baseline <- read_csv(paste0(modelResultPath, "/PredY_s_Baseline2.csv"))
```

```
## Warning: Missing column names filled in: 'X1' [1]
```

```
## Parsed with column specification:
## cols(
##   X1 = col_double(),
##   NSub = col_double(),
##   targetDur = col_double(),
##   RP = col_double(),
##   Exp = col_character(),
##   group = col_double(),
##   w = col_double(),
##   predY = col_double()
## )
```

```
PredY_mix_Baseline <- read_csv(paste0(modelResultPath, "/PredY_mix_Baseline2.csv"))
```

```
## Warning: Missing column names filled in: 'X1' [1]
```

```
## Parsed with column specification:
## cols(
##   X1 = col_double(),
##   NSub = col_double(),
##   targetDur = col_double(),
##   RP = col_double(),
##   Exp = col_character(),
##   group = col_double(),
##   predY = col_double(),
##   w = col_double()
## )
```

```
AllDat_Bayparlist <- read_csv(paste0(modelResultPath, "/BaseLine_parlist_Stan.csv"))
```

```
## Warning: Missing column names filled in: 'X1' [1]
```

```
## Parsed with column specification:
```

```
## cols(
##   X1 = col_double(),
##   Nsub = col_double(),
##   Exp = col_character(),
##   p_wf = col_double(),
##   wf = col_double(),
##   sig_m_square = col_double()
## )
```

```
PredY_Baseline <- rbind(PredY_l_Baseline, PredY_s_Baseline)
PredY_Baseline <- rbind(PredY_Baseline, PredY_mix_Baseline)
PredY_Baseline$group = factor(PredY_Baseline$group, labels = c("short", "long", "mix"))
```

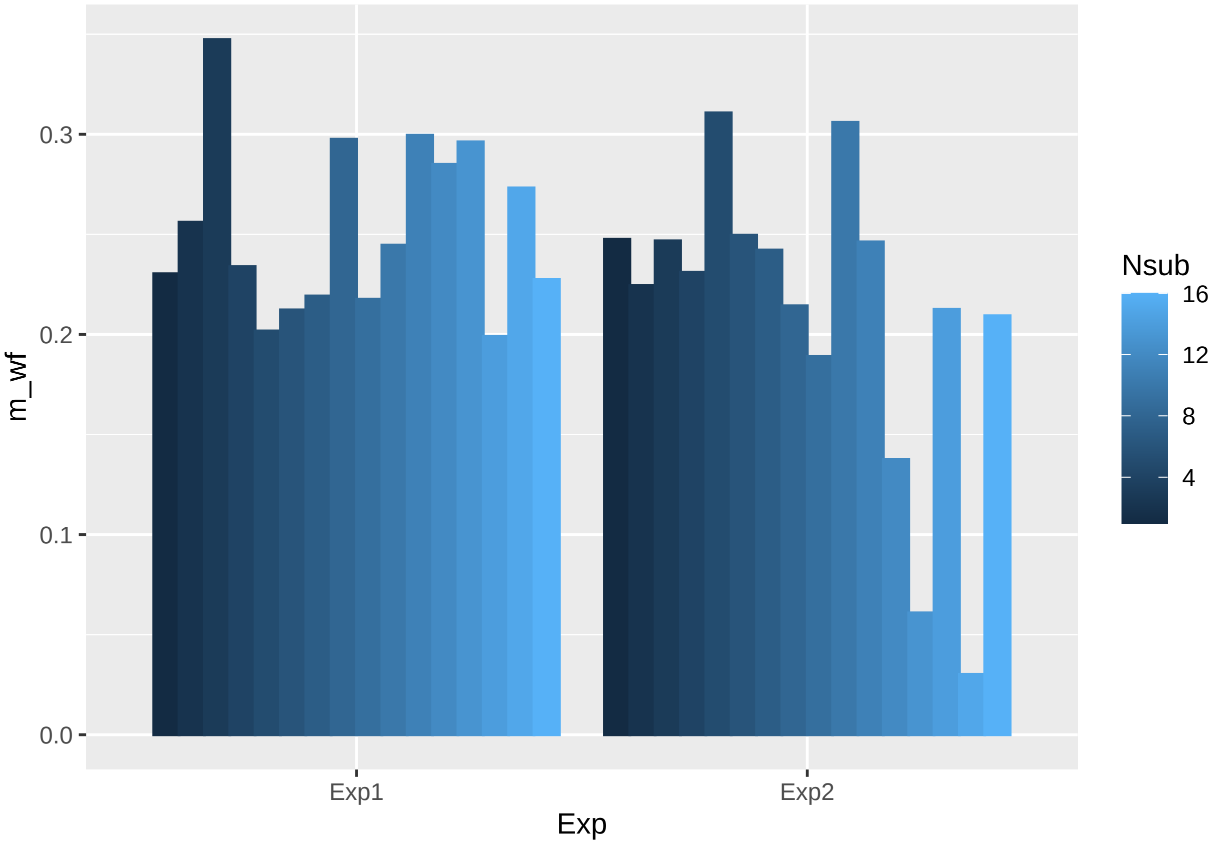
Analysis on the Rstan model parameters

```
m_Baypar <- dplyr::group_by(AllDat_Bayparlist, Exp, Nsub) %>%
  dplyr::summarize( m_sig_m_square = mean(sig_m_square), m_wf = mean(wf),
            m_p_wf = mean(p_wf))
m_Baypar
```

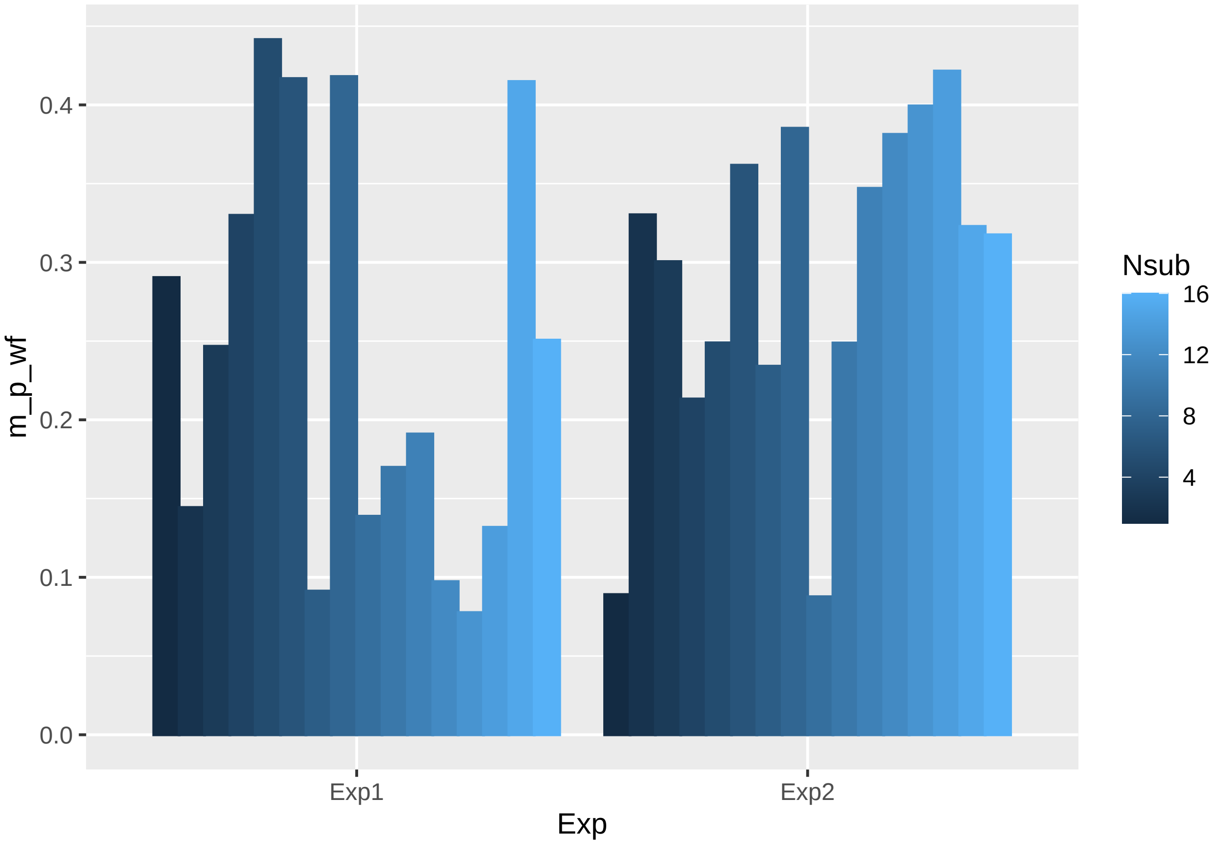
```
## # A tibble: 32 x 5
## # Groups:   Exp [2]
##    Exp    Nsub m_sig_m_square  m_wf m_p_wf
##    <chr> <dbl>          <dbl> <dbl>  <dbl>
##  1 Exp1      1         0.0996 0.230 0.290
##  2 Exp1      2         0.0788 0.256 0.144
##  3 Exp1      3         0.0872 0.347 0.247
##  4 Exp1      4         0.109  0.234 0.330
##  5 Exp1      5         0.116  0.202 0.442
##  6 Exp1      6         0.0642 0.212 0.417
##  7 Exp1      7         0.0730 0.219 0.0912
##  8 Exp1      8         0.0645 0.298 0.418
##  9 Exp1      9         0.0948 0.218 0.139
## 10 Exp1     10         0.118  0.245 0.170
## # ... with 22 more rows
```

p_wf in models

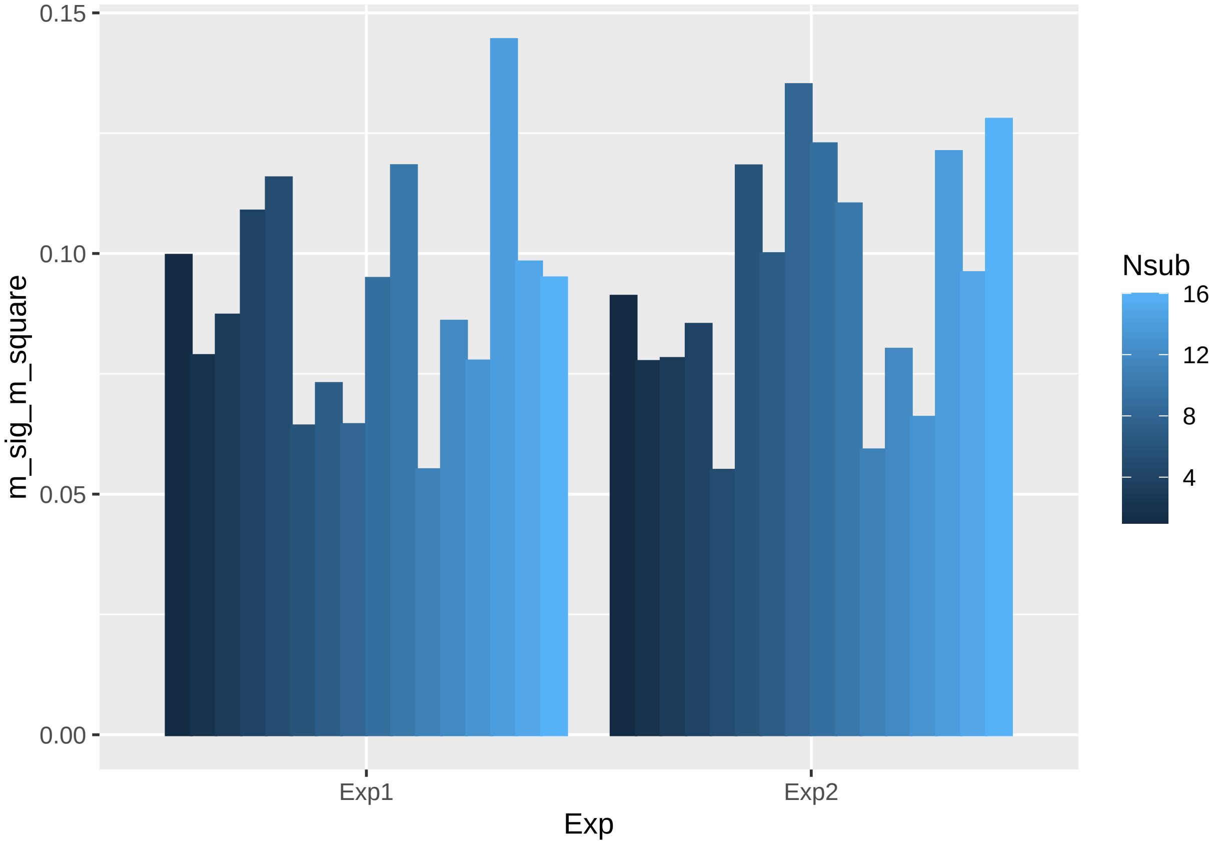
```
ggplot(m_Baypar, aes(x = Exp, y = m_wf, color = Nsub, fill = Nsub, group = Nsub)) +
    geom_bar(stat = "identity",
             position = position_dodge())
```



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



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



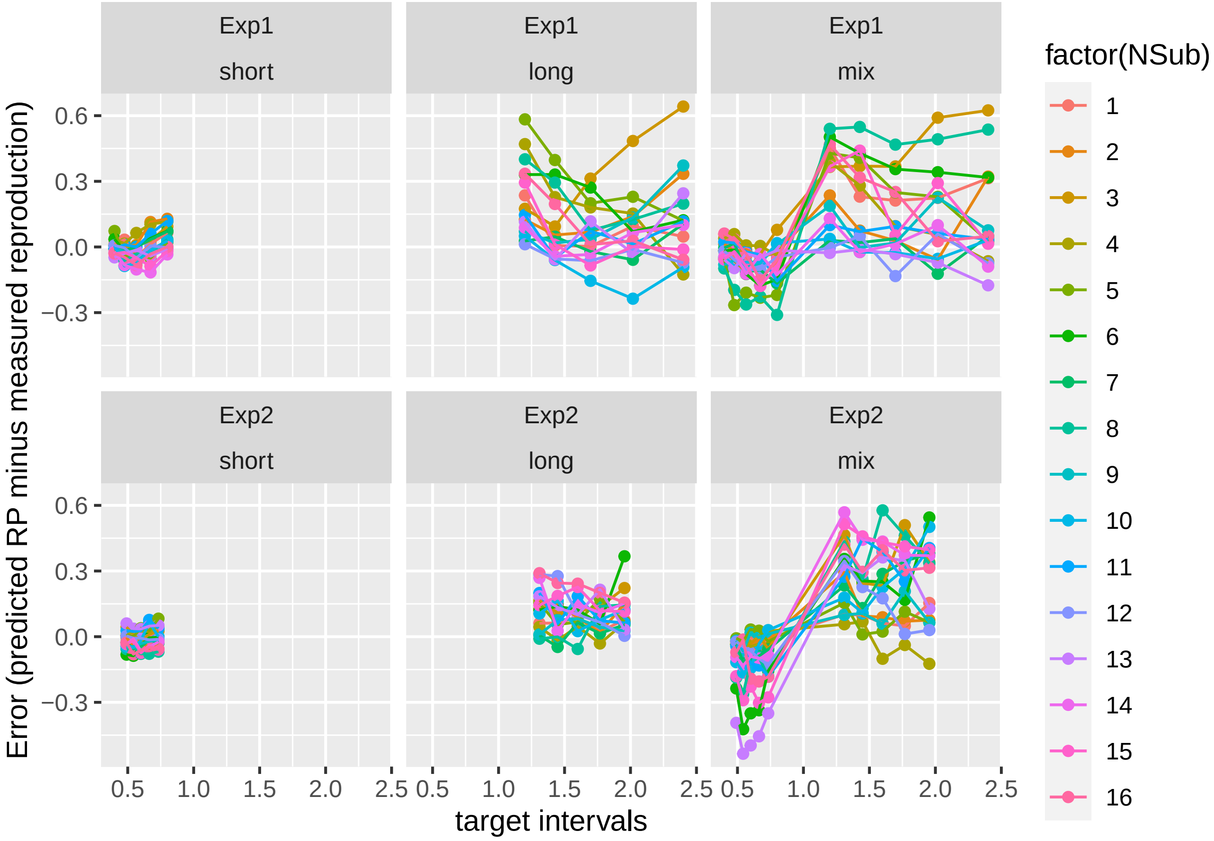
Prediction results (short blocks and long blocks)

```
predY <- dplyr::group_by(PredY_Baseline, targetDur, Exp, NSub, group) %>%
  dplyr::summarize(m_RP = mean(RP), n = n(), m_predY = mean(predY))
predY$m_rpErr = predY$m_predY - predY$m_RP
predY$m_relativeErr = predY$m_rpErr / predY$targetDur
predY
```

```
## # A tibble: 640 x 9
## # Groups:   targetDur, Exp, NSub [320]
##    targetDur Exp    NSub group  m_RP     n m_predY  m_rpErr m_relativeErr
##        <dbl> <chr> <dbl> <fct> <dbl> <int>   <dbl>    <dbl>         <dbl>
##  1       0.4 Exp1      1 short 0.531    28   0.544  0.0131         0.0327
##  2       0.4 Exp1      1 mix   0.487    15   0.545  0.0577         0.144
##  3       0.4 Exp1      2 short 0.486    28   0.473 -0.0130        -0.0326
##  4       0.4 Exp1      2 mix   0.459    14   0.472  0.0132         0.0329
##  5       0.4 Exp1      3 short 0.489    29   0.496  0.00665        0.0166
##  6       0.4 Exp1      3 mix   0.512    15   0.495 -0.0165        -0.0412
##  7       0.4 Exp1      4 short 0.496    28   0.531  0.0347         0.0868
##  8       0.4 Exp1      4 mix   0.490    15   0.530  0.0394         0.0986
##  9       0.4 Exp1      5 short 0.497    29   0.570  0.0729         0.182
## 10       0.4 Exp1      5 mix   0.624    16   0.570 -0.0543        -0.136
## # ... with 630 more rows
```

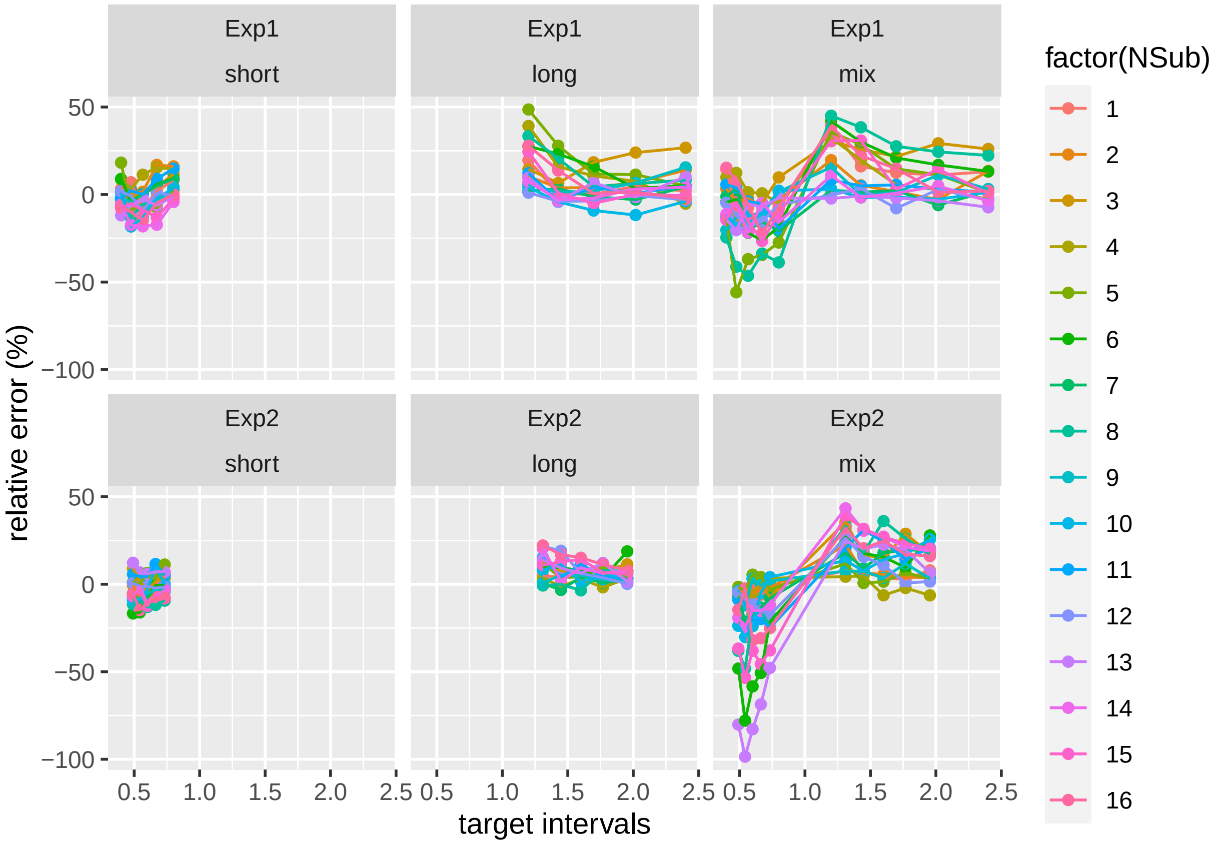
The predication of short and long blocks

```
#plot Error in predication
  ggplot(data=predY, aes(x= targetDur, y=m_rpErr, group = factor(NSub), color= factor(NSub))) +
    geom_point()+geom_line()+
    labs(x="target intervals", y="Error (predicted RP minus measured reproduction)")+
  facet_wrap(Exp~group)
```



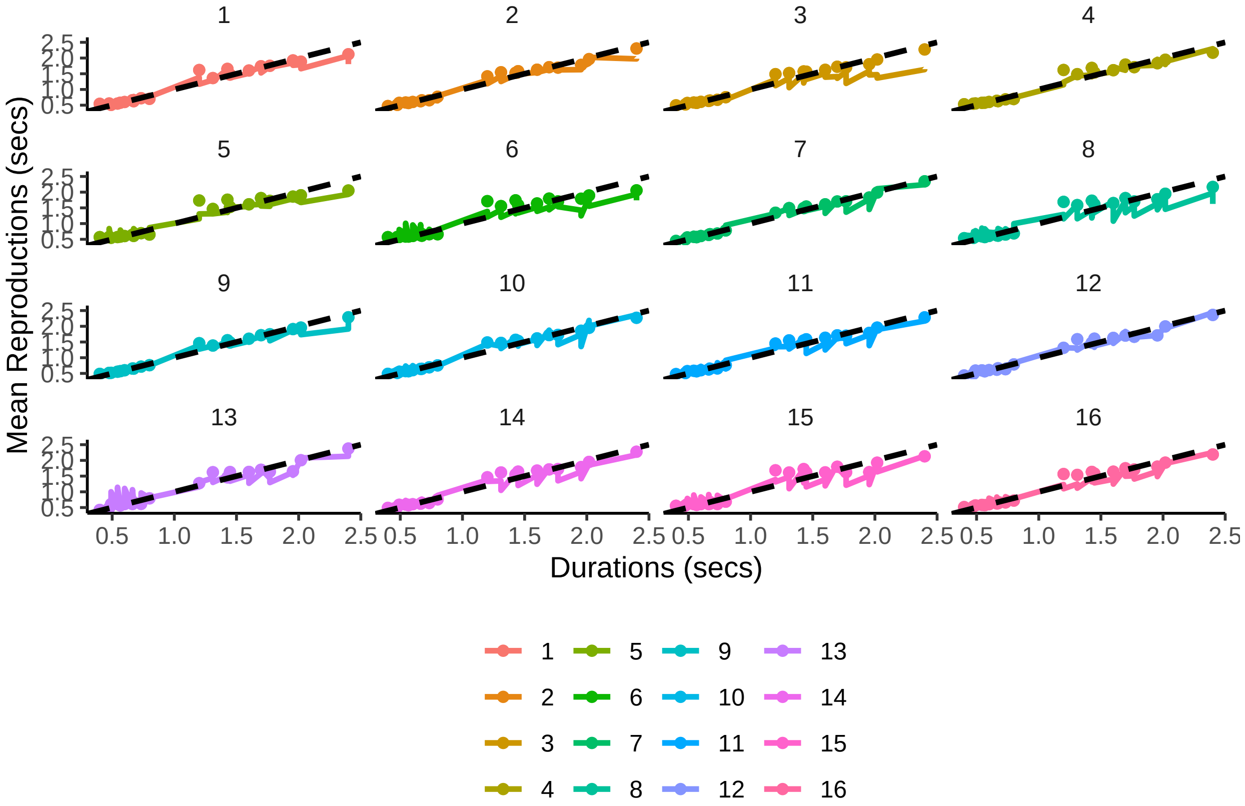
```
#plot relative Error for mixed blocks
 fig_rerr_model <-  ggplot(data=predY, aes(x= targetDur, y=m_relativeErr*100, group = factor(NSub), colo
     geom_point()+ geom_line()+
    labs(x="target intervals", y="relative error (%)")+
  facet_wrap(Exp~group)

fig_rerr_model
```



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

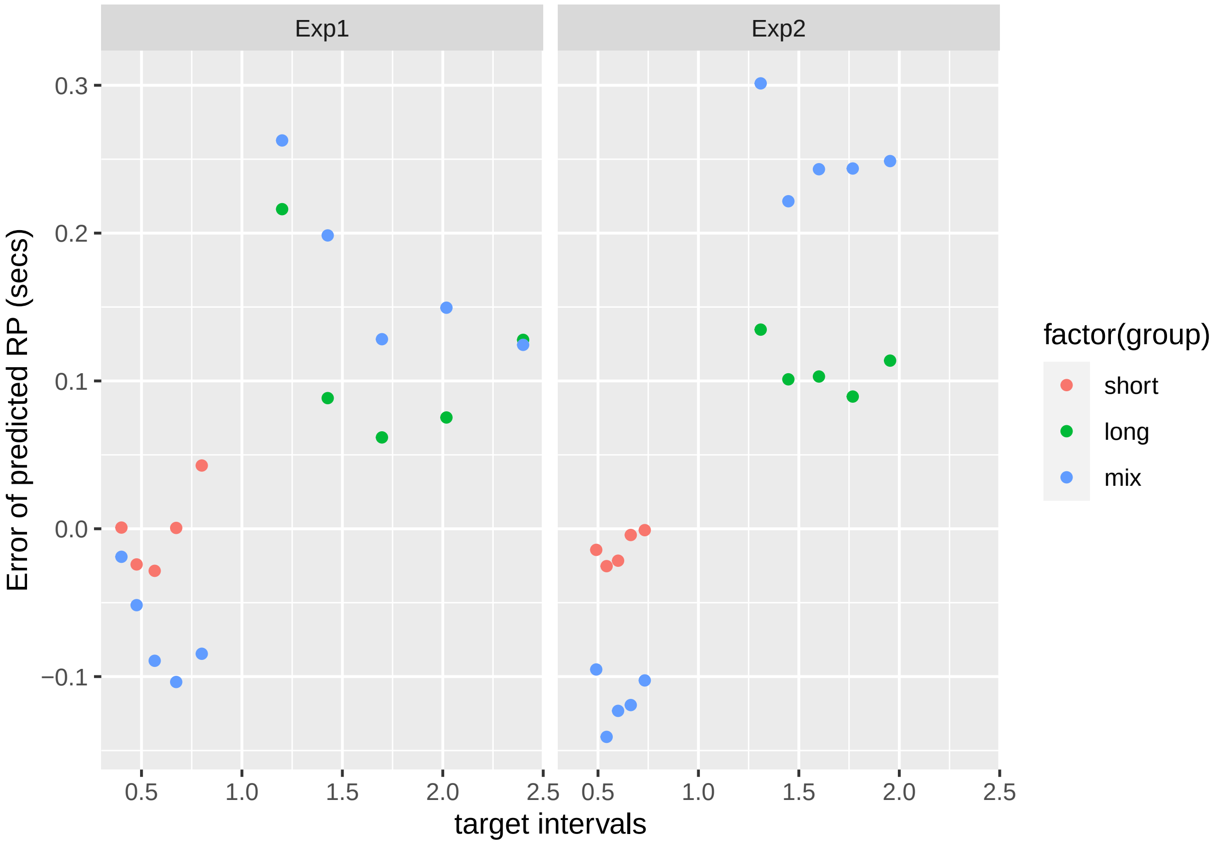
```
#plot the average of the predicted Y under the mixed condition
fig_mpredY = ggplot(predY) +
  geom_point(aes(targetDur, m_predY, group = factor(NSub), color = factor(NSub))) +
  geom_line(aes(targetDur, m_RP, group = factor(NSub), color = factor(NSub)),  size = 1) +
  #geom_errorbar(aes(ymin = m_m_predY-se_m_predY, 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~.)
fig_mpredY
```



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

```
m_predY <- predY%>%
  dplyr::group_by(targetDur, Exp, group) %>%
  dplyr::summarize(
    n = n(),
    m_predY = mean(m_predY),
    m_RP = mean(m_RP)
  )
m_predY$m_rpErr =m_predY$m_predY-m_predY$m_RP
```

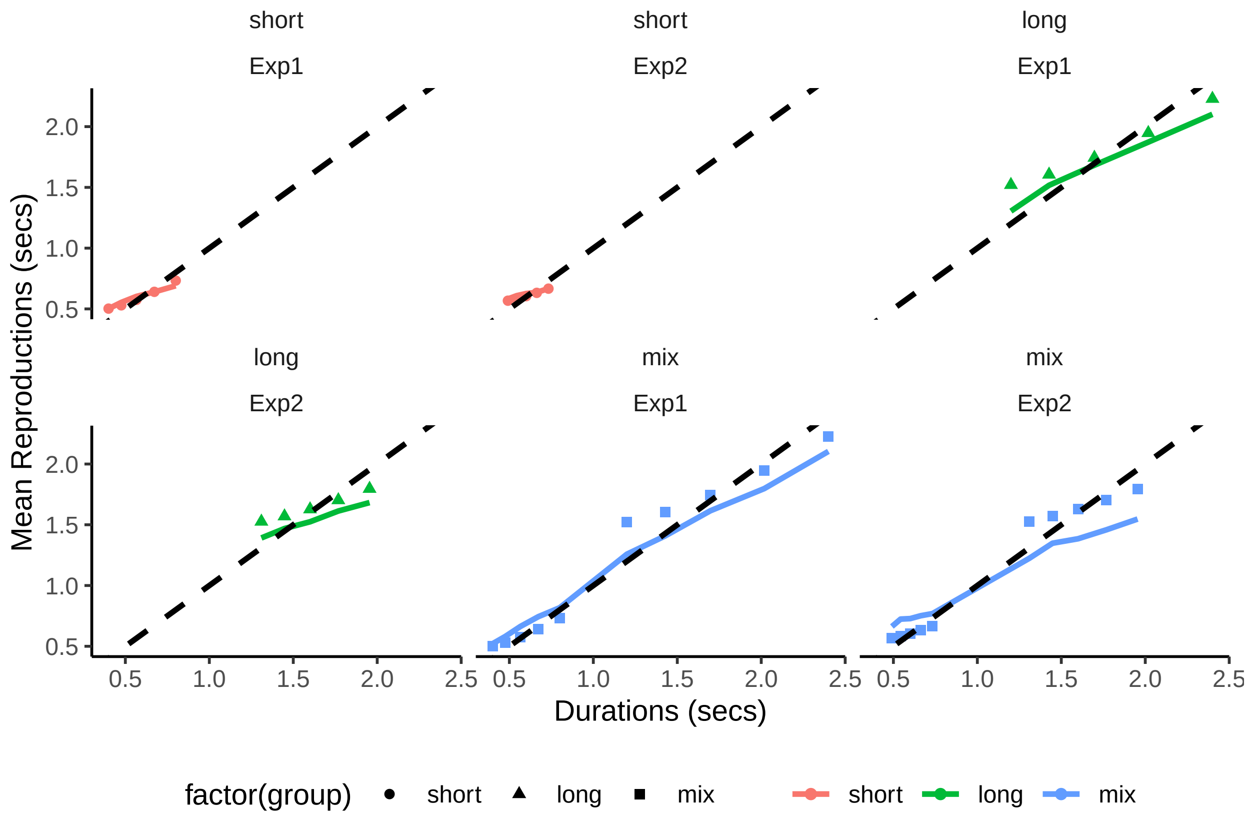
```
#plot Error in predication
  ggplot(data=m_predY, aes(x= targetDur, y=m_rpErr,
                           color = factor(group))) +
    geom_point()+  facet_wrap(~Exp) +
    labs(x="target intervals", y="Error of predicted RP (secs)")
```



```
#plot the average of the predicted Y under the mixed condition
fig_m_predY = ggplot(m_predY) +
  geom_point(aes(targetDur, m_predY, group = factor(group), color = factor(group), shape = factor(group
  geom_line(aes(targetDur, m_RP, group = factor(group), shape = factor(group), color = factor(group)),
  #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(group~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
```

```
## Warning: Ignoring unknown aesthetics: shape
```

```
fig_m_predY
```

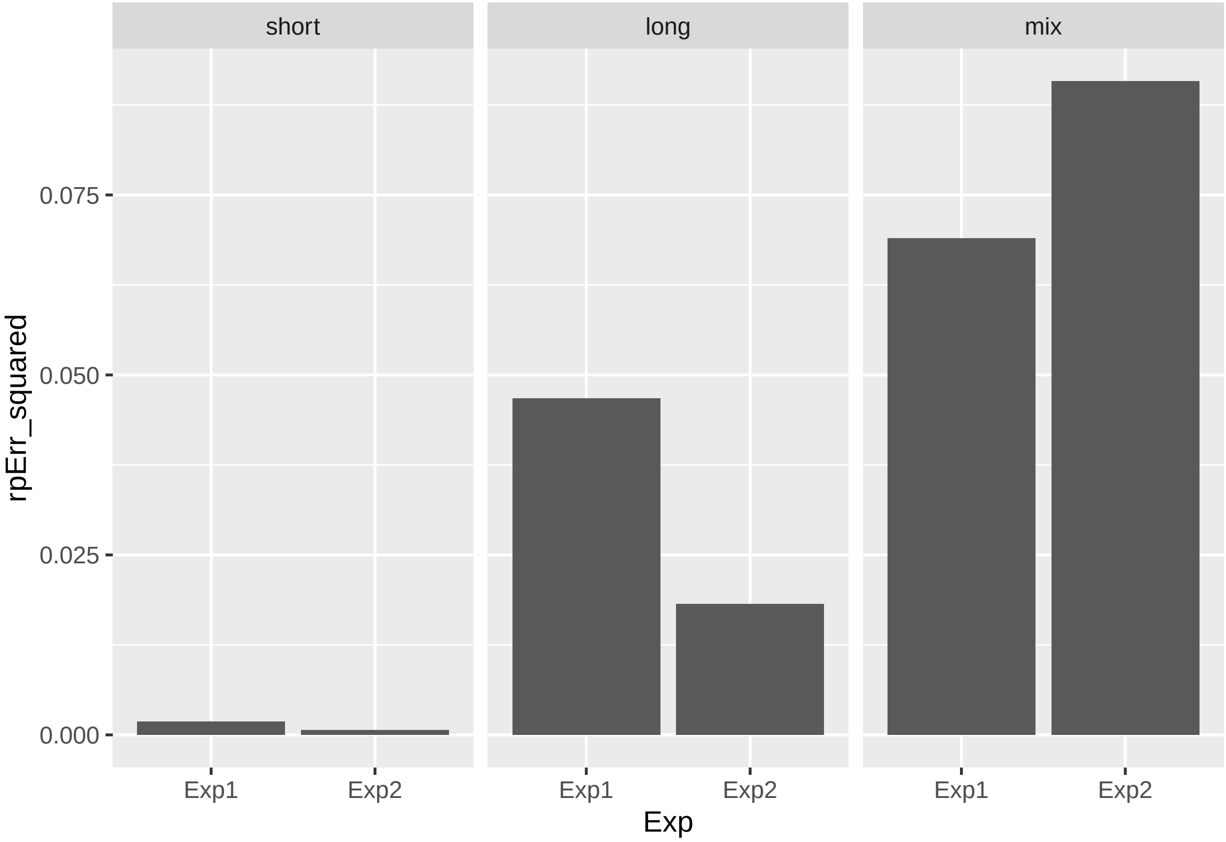


```
m_predY$rpErr_squared <- m_predY$m_rpErr^2

fig_rpErr_model <- ggplot(m_predY, aes(x = Exp, y = rpErr_squared)) +
    geom_bar(stat = "identity",
             position = position_dodge()) +
    theme(legend.position="bottom")+
    facet_wrap(~group)

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

fig_rpErr_model
```


short
long
mix
rpErr_squared
0.000
0.025
0.050
0.075
Exp1
Exp2
Exp1
Exp2
Exp1
Exp2
Exp