```
library(rstan)
library(dplyr)
options(mc.cores = parallel::detectCores())
rstan_options(auto_write = TRUE)


library(RWiener)


#original parameter values
th =  4.56
ndt =   1.32
beta =  .5
theta = .01
alpha = -0.36
delta_theta = 0.002

dat <- read.csv('final_data.csv')

dat <- dat %>%
  filter(test_part == 'cc' | test_part == 'ss',
         skew %in% c("rl", "lr", 'ns'),
         subject == '4xf5xlk2') %>%
  select(P_A1, O_A1, P_A2, O_A2, P_B1, O_B1, P_B2, O_B2, eva, evb, evd, sda, sdb,
sdd, test_part)

dat$con = ifelse(dat$test_part == 'cc', 1, -1)


columns_to_modify <- c('P_A1',  'P_A2', 'P_B1', 'P_B2')


for(column in columns_to_modify) {
  dat[[column]] <- dat[[column]] / 100
}

dat$evd_cat <- cut(dat$evd,
                   breaks = c(-Inf, -15, -5, 5, 15, Inf),
                   labels = c("-20", "-10", "0", "10", "20"),
                   right = FALSE)

dat$sdd_cat <- cut(dat$sdd,
                   breaks = c(-Inf, 6, 11, 16),
                   labels = c('5', "10", "15"),
                   right = FALSE)

stim <- dat[order(dat$evd_cat, dat$sdd_cat, dat$con), ]

stim$drift <- (theta + delta_theta * stim$con) * (stim$evd + alpha * stim$sdd)
```

```
stim2 = stim
stim3 = stim



for(n in 1:nrow(stim2)){

    stim2$simchosum[n] = 0
}

stim4 = stim
for(n in 1:nrow(stim4)){

    stim4$simchosum[n] = 0
}
```

```
sim_ddm <- "
data {
    int<lower=1> N;                                 // number of data items
    int<lower=1> L;                                 // number of participants
    // int<lower=1, upper=L> participant[N];            // level (participant)

    int<lower=-1,upper=1> cho[N];               // accuracy (-1, 1)
    real<lower=0> rt[N];                            // rt
    real evd[N];
    real sdd[N];
    real<lower=0, upper=1> starting_point;          // starting point diffusion mo
del not to estimate
    int<lower=-1,upper=1> con[N];
}

parameters {

    real alpha_sbj;
    real theta_v;
    real threshold_v;
    real ndt_v;
    real delta_theta;
}
transformed parameters {
    real drift_ll[N];                               // trial-by-trial drift rate f
or likelihood (incorporates accuracy)
    real drift_t[N];                                // trial-by-trial drift rate f
or predictions
    real<lower=0> threshold_t[N];                   // trial-by-trial threshold
    real<lower=0> ndt_t[N];                         // trial-by-trial ndt

    real<lower=0> theta_sbj;
    real<lower=0> threshold_sbj;
    real<lower=0> ndt_sbj;
```

```
    theta_sbj = log(1 + exp(theta_v));
    threshold_sbj = log(1 + exp(threshold_v));
    ndt_sbj = log(1 + exp(ndt_v));

    for (n in 1:N) {
        drift_t[n] = (theta_sbj + delta_theta*con[n])* (evd[n] + alpha_sbj * sdd[
n]);
        drift_ll[n] = drift_t[n]*cho[n];
        threshold_t[n] = threshold_sbj;
        ndt_t[n] = ndt_sbj;
    }
}
model {
  alpha_sbj ~ normal(0, 5);
    theta_v ~ normal(1,5);
    threshold_v ~ normal(1,3);
    ndt_v ~ normal(0,1);
    delta_theta ~ normal(0, 5);


    rt ~ wiener(threshold_t, ndt_t, starting_point, drift_ll);
}
generated quantities {
    vector[N] log_lik;

    {for (n in 1:N) {
        log_lik[n] = wiener_lpdf(rt[n] | threshold_t[n], ndt_t[n], starting_point,
drift_ll[n]);
    }
}
}

"
```

```
initFunc <-function (i) {
  initList=list()
  for (ll in 1:i){
    initList[[ll]] = list(
                          alpha_sbj = runif(1,-5,5),
                          theta_v = runif(1,-20,1),
                          threshold_v = runif(1,-0.5,5),
                          ndt_v = runif(1,-1.5, 0),
                          delta_theta = runif(1,-2,2)
    )
  }
  return(initList)
}
```

```
# Set the number of iterations
n_iter <- 20


`%+=%` = function(e1,e2) eval.parent(substitute(e1 <- e1 + e2))



# Create empty vectors to store the outcome parameters for each iteration
th_recover <- numeric(n_iter)
theta_recover <- numeric(n_iter)
ndt_recover <- numeric(n_iter)
alpha_recover <- numeric(n_iter)
delta_theta_recover <- numeric(n_iter)


th_bias <- numeric(n_iter)
theta_bias <- numeric(n_iter)
ndt_bias <- numeric(n_iter)
alpha_bias <- numeric(n_iter)
delta_theta_bias <- numeric(n_iter)


th_dev <- numeric(n_iter)
theta_dev <- numeric(n_iter)
ndt_dev <- numeric(n_iter)
alpha_dev <- numeric(n_iter)
delta_theta_dev <- numeric(n_iter)


# Run the model for n_iter iterations
for (i in 1:n_iter) {


  for(n in 1:nrow(stim)){
```

```
    cres <- rwiener(1,th, ndt, beta, (theta + delta_theta * stim$con[n]) * (stim$e
vd[n] + alpha * stim$sdd[n]))
    stim$simrt[n] <- as.numeric(cres[1])
    stim$simcho[n] <- ifelse(cres[2]=="upper",1,-1)
  }



  for(n in 1:nrow(stim2)){

    stim2$simchosum[n]  %+=% ifelse(stim$simcho[n]==1,1,0)
    }




  parameters = c("alpha_sbj","threshold_sbj","ndt_sbj",'theta_sbj', 'delta_theta')
  dataList  = list(cho = stim$simcho,rt = stim$simrt, N=90,  L = 1, starting_point
=0.5, evd = stim$evd, sdd = stim$sdd, con = stim$con)




  # Run the diffusion model for the current iteration
  dsamples <- stan(model_code = sim_ddm,
                data=dataList,
                pars=parameters,
                iter=1000,
                chains=4,#If not specified, gives random inits
                init=initFunc(4),
                warmup = 500,  # Stands for burn-in; Default = iter/2
                refresh = 0
                )

  samples <- rstan::extract(dsamples, pars = c('alpha_sbj', 'theta_sbj', 'threshol
d_sbj', 'ndt_sbj', 'delta_theta'))


  # Store the outcome parameters for the current iteration
  th_recover[i] <- mean(samples$threshold_sbj)
  theta_recover[i] <- mean(samples$theta_sbj)
  ndt_recover[i] <- mean(samples$ndt_sbj)
  alpha_recover[i] <- mean(samples$alpha_sbj)
  delta_theta_recover[i] <- mean(samples$delta_theta)


  th_bias[i] <- (mean(samples$threshold_sbj)-th)/th
  theta_bias[i] <- (mean(samples$theta_sbj)-theta)/theta
  ndt_bias[i] <- (mean(samples$ndt_sbj)-ndt)/ndt
```

```
  alpha_bias[i] <- (mean(samples$alpha_sbj)-alpha)/alpha
  delta_theta_bias[i] <- (mean(samples$delta_theta)-delta_theta)/delta_theta


  th_dev[i] <- abs(mean(samples$threshold_sbj)-th)/th
  theta_dev[i] <- abs(mean(samples$theta_sbj)-theta)/theta
  ndt_dev[i] <- abs(mean(samples$ndt_sbj)-ndt)/ndt
  alpha_dev[i] <- abs(mean(samples$alpha_sbj)-alpha)/alpha
  delta_theta_dev[i] <- abs(mean(samples$delta_theta)-delta_theta)/delta_theta




}
```

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

```
## Running /Library/Frameworks/R.framework/Resources/bin/R CMD SHLIB foo.c
## clang -mmacosx-version-min=10.13 -I"/Library/Frameworks/R.framework/Resources/i
nclude" -DNDEBUG   -I"/Library/Frameworks/R.framework/Versions/4.2/Resources/libra
ry/Rcpp/include/"  -I"/Library/Frameworks/R.framework/Versions/4.2/Resources/libra
ry/RcppEigen/include/"  -I"/Library/Frameworks/R.framework/Versions/4.2/Resources/
library/RcppEigen/include/unsupported"  -I"/Library/Frameworks/R.framework/Version
s/4.2/Resources/library/BH/include" -I"/Library/Frameworks/R.framework/Versions/4.
2/Resources/library/StanHeaders/include/src/"  -I"/Library/Frameworks/R.framework/
Versions/4.2/Resources/library/StanHeaders/include/"  -I"/Library/Frameworks/R.fra
mework/Versions/4.2/Resources/library/RcppParallel/include/"  -I"/Library/Framewor
ks/R.framework/Versions/4.2/Resources/library/rstan/include" -DEIGEN_NO_DEBUG  -DB
OOST_DISABLE_ASSERTS  -DBOOST_PENDING_INTEGER_LOG2_HPP  -DSTAN_THREADS  -DBOOST_NO
_AUTO_PTR  -include '/Library/Frameworks/R.framework/Versions/4.2/Resources/librar
y/StanHeaders/include/stan/math/prim/mat/fun/Eigen.hpp'  -D_REENTRANT -DRCPP_PARAL
LEL_USE_TBB=1   -I/usr/local/include   -fPIC  -Wall -g -O2  -c foo.c -o foo.o
## In file included from <built-in>:1:
## In file included from /Library/Frameworks/R.framework/Versions/4.2/Resources/li
brary/StanHeaders/include/stan/math/prim/mat/fun/Eigen.hpp:13:
## In file included from /Library/Frameworks/R.framework/Versions/4.2/Resources/li
brary/RcppEigen/include/Eigen/Dense:1:
## In file included from /Library/Frameworks/R.framework/Versions/4.2/Resources/li
brary/RcppEigen/include/Eigen/Core:88:
## /Library/Frameworks/R.framework/Versions/4.2/Resources/library/RcppEigen/includ
e/Eigen/src/Core/util/Macros.h:628:1: error: unknown type name 'namespace'
## namespace Eigen {
## ^
## /Library/Frameworks/R.framework/Versions/4.2/Resources/library/RcppEigen/includ
e/Eigen/src/Core/util/Macros.h:628:16: error: expected ';' after top level declara
tor
## namespace Eigen {
##                ^
##                ;
## In file included from <built-in>:1:
## In file included from /Library/Frameworks/R.framework/Versions/4.2/Resources/li
brary/StanHeaders/include/stan/math/prim/mat/fun/Eigen.hpp:13:
## In file included from /Library/Frameworks/R.framework/Versions/4.2/Resources/li
brary/RcppEigen/include/Eigen/Dense:1:
## /Library/Frameworks/R.framework/Versions/4.2/Resources/library/RcppEigen/includ
e/Eigen/Core:96:10: fatal error: 'complex' file not found
## #include <complex>
##          ^~~~~~~~~
## 3 errors generated.
## make: *** [foo.o] Error 1
```

```
#create a summary df of all parameters
df_summary <- data.frame(original_th = th,
                 recovered_th = th_recover,
                 bias_th = th_bias,
                 deviation_th = th_dev,
                 original_theta = theta,
                 recovered_theta = theta_recover,
                 bias_theta = theta_bias,
                 deviation_theta = theta_dev,
                 original_ndt = ndt,
                 recovered_ndt = ndt_recover,
                 bias_ndt = ndt_bias,
                 deviation_ndt = ndt_dev,
                 original_alpha = alpha,
                 recovered_alpha = alpha_recover,
                 bias_alpha = alpha_bias,
                 deviation_alpha = alpha_dev,
                 original_delta_theta = delta_theta,
                 recovered_delta_theta = delta_theta_recover,
                 bias_delta_theta = delta_theta_bias,
                 deviation_delta_theta = delta_theta_dev
                 )
```

```
#create a table to show all means and true values
df_mean <- data.frame(parameter = c('th', "theta", "ndt", "alpha", 'delta_theta'),
                      true_value = c(th, theta,ndt, alpha, delta_theta),
                      mean_recovered = c(mean(df_summary$recovered_th), mean(df_su
mmary$recovered_theta),mean(df_summary$recovered_ndt),mean(df_summary$recovered_al
pha), mean(df_summary$recovered_delta_theta)),
                      mean_bias = c(mean(df_summary$bias_th), mean(df_summary$bias
_theta),mean(df_summary$bias_ndt),mean(df_summary$bias_alpha), mean(df_summary$bia
s_delta_theta)),
                      mean_deviation = c(mean(df_summary$deviation_th), mean(df_su
mmary$deviation_theta),mean(df_summary$deviation_ndt),mean(df_summary$deviation_al
pha), mean(df_summary$deviation_delta_theta))
                 )
df_mean
```

```
##     parameter true_value mean_recovered     mean_bias mean_deviation
## 1          th      4.560    4.562158642  0.0004733865     0.04242890
## 2       theta      0.010    0.007345112 -0.2654887788     0.40999227
## 3         ndt      1.320    1.268747189 -0.0388278874     0.09423101
## 4       alpha     -0.360   -0.540735721  0.5020436697    -1.20532921
## 5 delta_theta      0.002    0.001622808 -0.1885957955     0.83556092
```

```
df_median <- data.frame(parameter = c('th', "theta", "ndt", "alpha", 'delta_thet
a'),
                        true_value = c(th, theta,ndt, alpha, delta_theta),
                        median_recovered = c(median(df_summary$recovered_th), media
n(df_summary$recovered_theta),median(df_summary$recovered_ndt),median(df_summary$r
ecovered_alpha), median(df_summary$recovered_delta_theta))
                        )

df_median
```

```
##     parameter true_value median_recovered
## 1          th      4.560      4.622587130
## 2       theta      0.010      0.007467987
## 3         ndt      1.320      1.246827249
## 4       alpha     -0.360     -0.557378886
## 5 delta_theta      0.002      0.001397025
```

```
#check whether the risky choice proportion can be successfully recovered by the me
an-variance model
#firstly, use recovered parameter values to simulation choice data
for (i in 1:n_iter) {

  for(n in 1:nrow(stim3)){
      cres <- rwiener(1,mean(df_summary$recovered_th), mean(df_summary$recovered_n
dt), beta, (mean(df_summary$recovered_theta) + mean(df_summary$recovered_delta_the
ta)* stim3$con[n] ) * (stim3$evd[n] + mean(df_summary$recovered_alpha) * stim3$sd
d[n]))
      stim3$simrt[n] <- as.numeric(cres[1])
      stim3$simcho[n] <- ifelse(cres[2]=="upper",1,-1)

  }
  for(n in 1:nrow(stim4)){

    stim4$simchosum[n]  %+=% ifelse(stim3$simcho[n]==1,1,0)
    }
}
```

```
#create summary dataframe
label <- c(rep("basic", 90), rep("recovered", 90))
df <- data.frame(trial = rep(1:90, 2),
                 value = c(stim2$simchosum/n_iter, stim4$simchosum/n_iter),
                 type = rep(label))
#display the first n trials
subset_data <- df[df$trial <= 90, ]
```

```
library(ggplot2)
ggplot(subset_data, aes(x = factor(trial), y = value, fill = type, colour = type))
+
  geom_bar(stat = "identity", position = "dodge")+
  ylim(0,1)
```