```
library(rstan)
```

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

```
## Loading required package: ggplot2
```

```
## rstan (Version 2.21.8, 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)
```

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

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library(RWiener)


#original parameter values
th =  4.36
ndt =   1.1
beta =  .5
theta = .015
alpha = -0.55


stim1 = read.csv('Switching-Gambles.csv')


stim <- rbind(stim1, stim1)
stim$complexindex <- ifelse(1:nrow(stim) <= 60, 1, -1)
```

```
# gamble characteristics
  stim$eva = stim$payoffa1*stim$proba1+stim$payoffa2*stim$proba2
  stim$evb = stim$payoffb1*stim$probb1+stim$payoffb2*stim$probb2
  stim$evd = stim$evb-stim$eva
  stim$sda = sqrt((stim$payoffa1-stim$eva)^2*stim$proba1 + (stim$payoffa2-stim$ev
a)^2*stim$proba2)
  stim$sdb = sqrt((stim$payoffb1-stim$evb)^2*stim$probb1 + (stim$payoffb2-stim$ev
b)^2*stim$probb2)
  stim$sdd = stim$sdb - stim$sda
```

```
stim <- stim %>%
  rowwise() %>%
  mutate(
    evd = evd * (-complexindex),
    sdd = sdd * (-complexindex)
  )

stim$num <- seq_len(nrow(stim))
stim
```

```
## # A tibble: 120 × 16
## # Rowwise:
##      num payoffa1 payoffa2 proba1 proba2 payoffb1 payoffb2 probb1 probb2
##    <int>    <int>    <int>  <dbl>  <dbl>    <int>    <int>  <dbl>  <dbl>
##  1     1        9       48   0.25   0.75       91        2   0.14   0.86
##  2     2       22       52   0.61   0.39        4       99   0.89   0.11
##  3     3       35       66   0.55   0.45       96        3   0.24   0.76
##  4     4       48       67   0.92   0.08       84        4   0.26   0.74
##  5     5       45       36   0.95   0.05       92        1   0.18   0.82
##  6     6       38       67   0.03   0.97        7       92   0.57   0.43
##  7     7       17       77   0.53   0.47        6       95   0.72   0.28
##  8     8       67       20   0.63   0.37        9       98   0.75   0.25
##  9     9       36       18   0.96   0.04        2       57   0.7    0.3
## 10    10       61       43   0.31   0.69       87        7   0.3    0.7
## # ℹ 110 more rows
## # ℹ 7 more variables: complexindex <dbl>, eva <dbl>, evb <dbl>, evd <dbl>,
## #   sda <dbl>, sdb <dbl>, sdd <dbl>
```

```
stim2 <- stim
stim3 <-  stim
stim4 <-  stim

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

    stim2$simchosum[n] = 0
}
```

```
## Warning: Unknown or uninitialised column: `simchosum`.
```

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

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

```
## Warning: Unknown or uninitialised column: `simchosum`.
```

```
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
}

parameters {

    real alpha;
    real theta;
    real threshold;
    real ndt;
}
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));
    threshold_sbj = log(1 + exp(threshold));
    ndt_sbj = log(1 + exp(ndt));

    for (n in 1:N) {
        drift_t[n] = theta_sbj * (evd[n] + alpha * sdd[n]);
        drift_ll[n] = drift_t[n]*cho[n];
        threshold_t[n] = threshold_sbj;
        ndt_t[n] = ndt_sbj;
    }
}
model {
  alpha ~ normal(0, 5);
    theta ~ normal(1,5);
    threshold ~ normal(1,3);
    ndt ~ normal(0,1);


    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 = runif(1,-5,5),
                         theta = runif(1,-20,1),
                         threshold = runif(1,-0.5,5),
                         ndt = runif(1,-1.5, 0)
    )
  }
  return(initList)
}
```

```
# Set the number of iterations
n_iter <- 100


`%+=%` = 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)

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

th_dev <- numeric(n_iter)
theta_dev <- numeric(n_iter)
ndt_dev <- numeric(n_iter)
alpha_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 * (stim$evd[n] + alpha * stim$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","threshold_sbj","ndt_sbj",'theta_sbj')
  dataList  = list(cho = stim$simcho,rt = stim$simrt, N=120,  L = 1, starting_point=0.5, evd = stim$evd, sdd = stim$sdd)





  # 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', 'theta_sbj', 'threshold_sbj', 'ndt_sbj'))


  # 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)



  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)-alpha)/alpha


  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)-alpha)/alpha




}
```

```
## 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
                 )
```

```
#create a table to show all means and true values
df_mean <- data.frame(parameter = c('th', "theta", "ndt", "alpha"),
                      true_value = c(th, theta,ndt, alpha),
                      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_bias = c(mean(df_summary$bias_th), mean(df_summary$bias
_theta),mean(df_summary$bias_ndt),mean(df_summary$bias_alpha)),
                      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))
                 )
df_mean
```

```
##   parameter true_value mean_recovered    mean_bias mean_deviation
## 1        th      4.360     4.34855131 -0.002625846     0.04028745
## 2     theta      0.015     0.01388881 -0.074079284     0.22029846
## 3       ndt      1.100     1.08852986 -0.010427399     0.08093723
## 4     alpha     -0.550    -0.75984759  0.381541069    -0.49768928
```

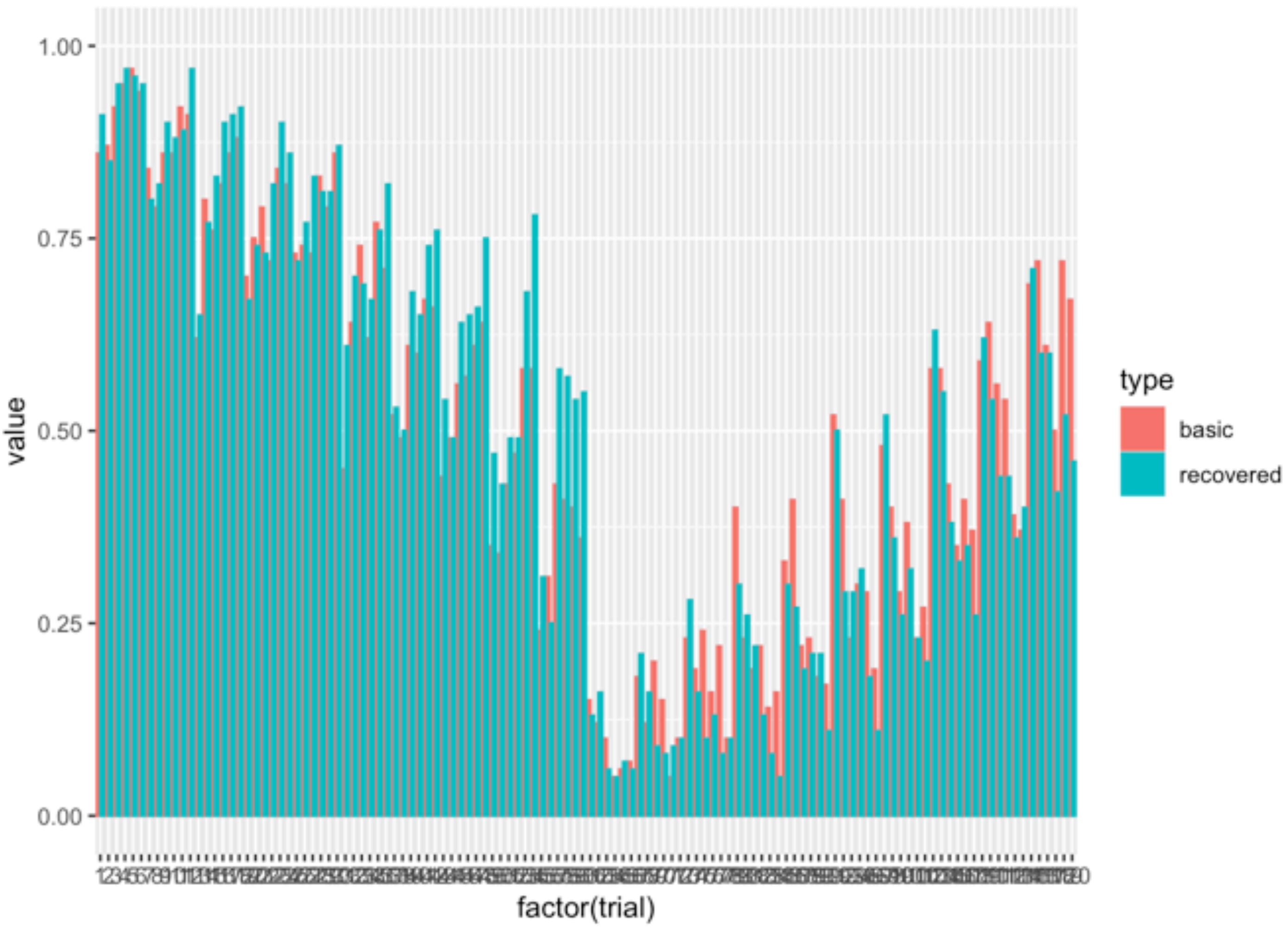
```
df_median <- data.frame(parameter = c('th', "theta", "ndt", "alpha"),
                      true_value = c(th, theta,ndt, alpha),
                      median_recovered = c(median(df_summary$recovered_th), media
n(df_summary$recovered_theta),median(df_summary$recovered_ndt),median(df_summary$r
ecovered_alpha))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

df_median
```

```
##   parameter true_value median_recovered
## 1        th      4.360       4.34514321
## 2     theta      0.015       0.01386356
## 3       ndt      1.100       1.07425977
## 4     alpha     -0.550      -0.64464209
```

```
#check whether the risky choice proportion can be successfully recovered by the mean-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_ndt), beta, mean(df_summary$recovered_theta) * (stim3$evd[n] + mean(df_summary$recovered_alpha) * stim3$sdd[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", 120), rep("recovered", 120))
df <- data.frame(trial = rep(stim2$num),
                 value = c(stim2$simchosum/n_iter, stim4$simchosum/n_iter),
                 type = rep(label))
#display the first n trials
subset_data <- df[df$trial <= 120, ]
```

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



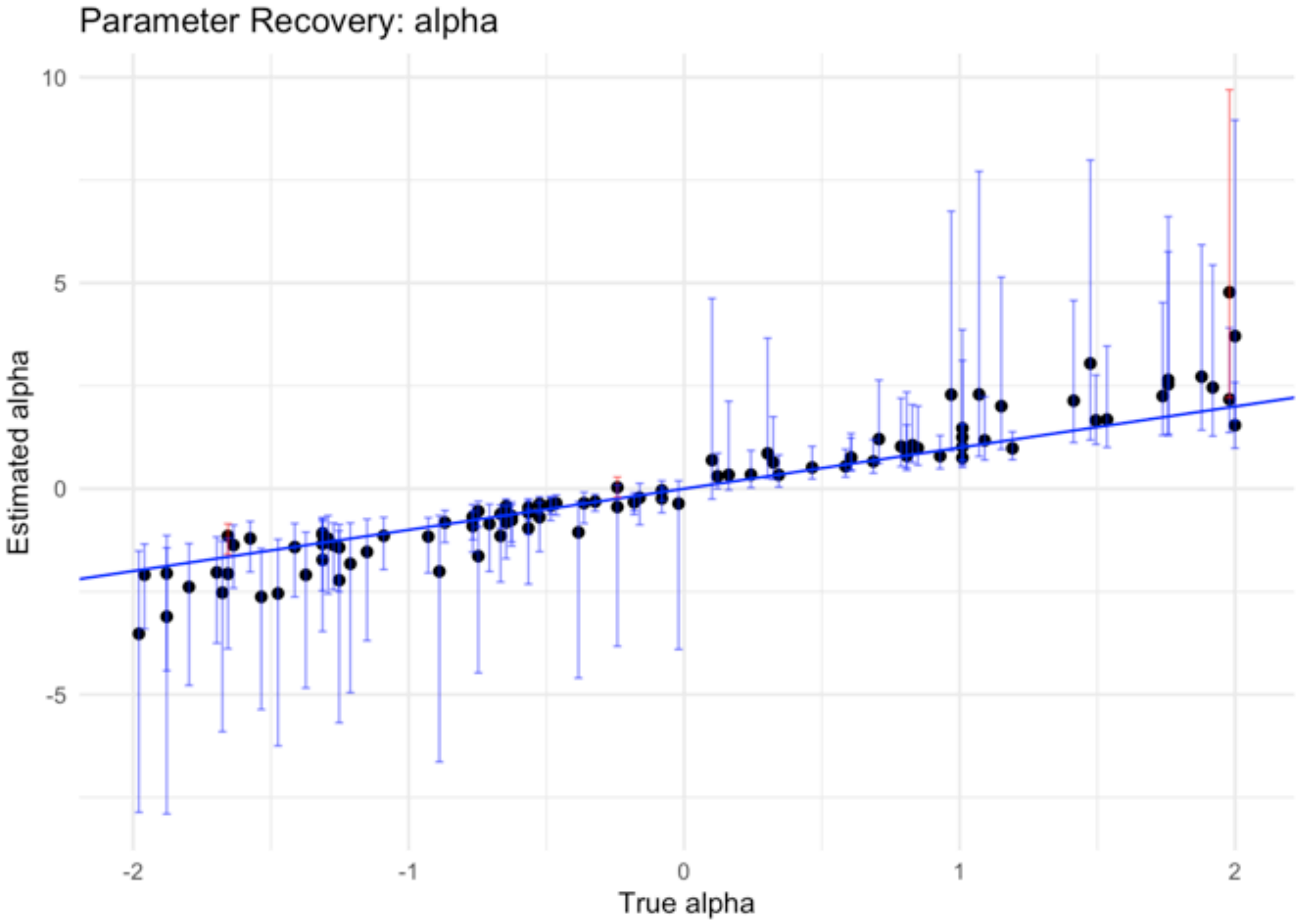
```
library(rstan)
library(RWiener)
th =  4.36
ndt =   1.1
beta =  .5
theta = .015





stim1 = read.csv('Switching-Gambles.csv')


stim <- rbind(stim1, stim1)
stim$complexindex <- ifelse(1:nrow(stim) <= 60, 1, -1)


# gamble characteristics
  stim$eva = stim$payoffa1*stim$proba1+stim$payoffa2*stim$proba2

  stim$evb = stim$payoffb1*stim$probb1+stim$payoffb2*stim$probb2
  stim$evd = stim$evb-stim$eva
  stim$sda = sqrt((stim$payoffa1-stim$eva)^2*stim$proba1 + (stim$payoffa2-stim$ev
a)^2*stim$proba2)
  stim$sdb = sqrt((stim$payoffb1-stim$evb)^2*stim$probb1 + (stim$payoffb2-stim$ev
b)^2*stim$probb2)
  stim$sdd = stim$sdb - stim$sda


stim <- stim %>%
  rowwise() %>%
  mutate(
    evd = evd * (-complexindex),
    sdd = sdd * (-complexindex)
  )

stim$num <- seq_len(nrow(stim))


stim2 <- stim
stim3 <-  stim
stim4 <-  stim

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

    stim2$simchosum[n] = 0
}

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

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

```
# Set the number of iterations
n_iter <- 100


`%+=%` = 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)

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

th_dev <- numeric(n_iter)
```

```
theta_dev <- numeric(n_iter)
ndt_dev <- numeric(n_iter)
alpha_dev <- numeric(n_iter)

# Storage for results
results_df <- data.frame(
  True_alpha = numeric(n_iter),
  Estimated_alpha = numeric(n_iter),
  CI_alpha_Lower = numeric(n_iter),
  CI_alpha_Upper = numeric(n_iter)
)

alpha_set <- numeric(n_iter)
# Run the model for n_iter iterations
for (i in 1:n_iter) {

  # Set the range (minimum and maximum values)
  min_value <- -2
  max_value <- 2


   # Generate a single random non-zero value within the range
  alpha <- 0
  while (alpha == 0) {
    alpha <- sample(c(seq(min_value, -0.0001, length.out = 100), seq(0.0001, max_v
alue, length.out = 100)), 1)
  }
  alpha_set[i] = alpha


  for(n in 1:nrow(stim)){
    cres <- rwiener(1,th, ndt, beta, theta * (stim$evd[n] + alpha * stim$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","threshold_sbj","ndt_sbj",'theta_sbj')
  dataList  = list(cho = stim$simcho,rt = stim$simrt, N=120,  L = 1, starting_poin
t=0.5, evd = stim$evd, sdd = stim$sdd)

```

```
  # 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', 'theta_sbj', 'threshold_sb
j', 'ndt_sbj'))
  extracted_params <- rstan::extract(dsamples)
  Estimated_alpha = mean(extracted_params$alpha)
  CI_alpha = quantile(extracted_params$alpha, probs = c(0.025, 0.975))

  # 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)



  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)-alpha)/alpha


  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)-alpha)/alpha

    # Store the results in the data frame
  results_df[i, ] <- c(
    alpha,
    Estimated_alpha,
    CI_alpha[1],
    CI_alpha[2]
  )


}
```

```
library(ggplot2)
# Create scatterplots for True vs. Estimated Intercepts with color-coded error bar
s
ggplot(results_df, aes(x = True_alpha, y = Estimated_alpha)) +
  geom_point(shape = 16, size = 2, color = "black", fill = "white") +
  geom_abline(intercept = 0, slope = 1, color = "blue") +
  geom_errorbar(
    aes(ymin = results_df$CI_alpha_Lower, ymax = results_df$CI_alpha_Upper),
    width = 0.03,
     color = ifelse(results_df$CI_alpha_Lower > results_df$True_alpha | results_df
$CI_alpha_Upper < results_df$True_alpha, "red", "blue"),
    linetype = "solid",
    linewidth = 0.4,
    alpha = 0.5
  ) +
  labs(
    title = "Parameter Recovery: alpha",
    x = "True alpha",
    y = "Estimated alpha"
  )+
  #ylim(-4, 4) +
  theme_minimal()  # Change to a minimal theme
```


Parameter Recovery: alpha
Estimated alpha
True alpha