Fitting ddm

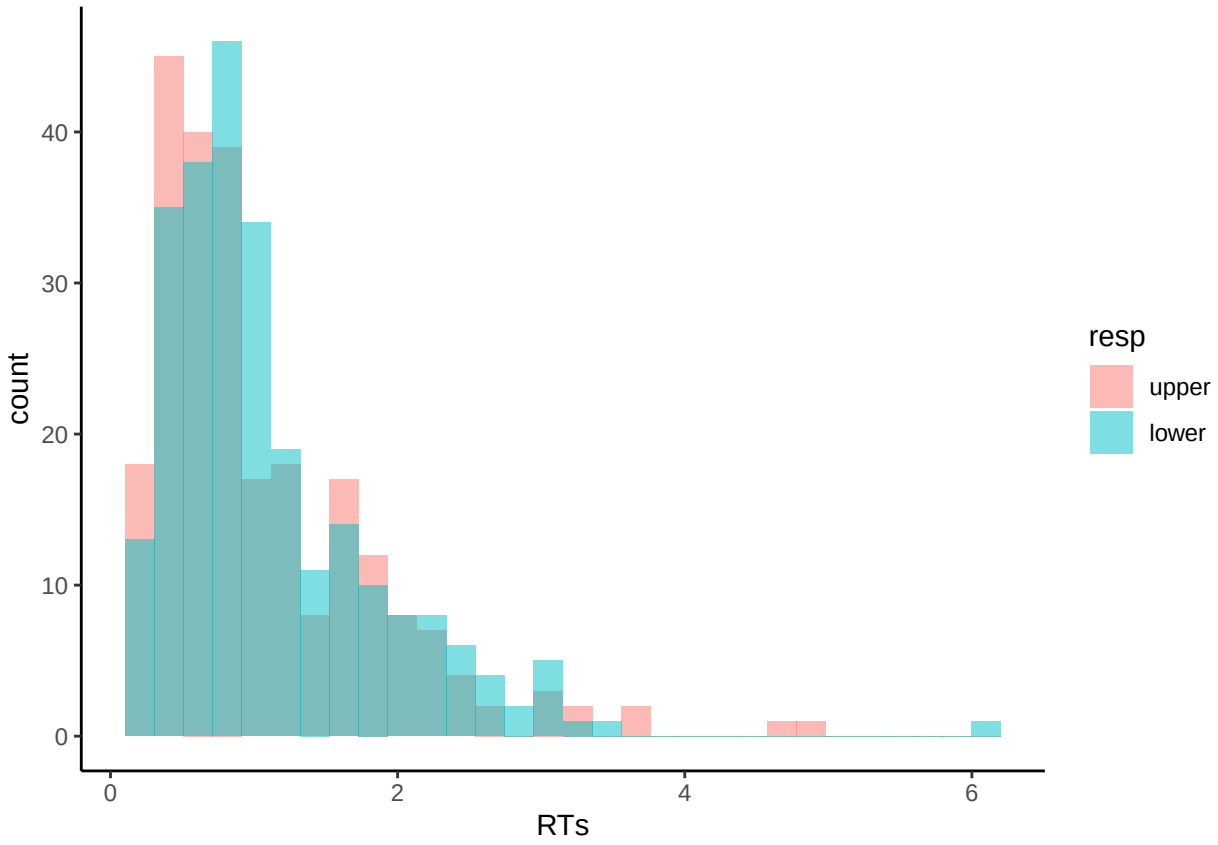
2023-10-22

Simulating and fitting data with the DDM

```
set.seed(123)
trials = 500
alpha = 2
delta = 0
beta = 0.5
tau = 0.1
#for later!
reals = data.frame(variable = c("alpha","beta","delta","tau"),reals = c(alpha,beta,delta,tau))


parameters = data.frame(alpha,delta,beta,tau, trials)

data = rwiener(n = trials,
        alpha = alpha,
        delta = delta,
        beta = beta,
        tau = tau)
```

Visualing the simulations

```
data %>% ggplot(aes(x = q, fill = resp), col = "black")+
  geom_histogram(position = "identity",alpha = 0.5)+
  theme_classic()+
  xlab("RTs")
```



Fitting the model in Stan

```
data_stan = list(trials = nrow(data),
                 RT = data$q,
                 resp = ifelse(data$resp == "lower",0,1),
                 minRT = min(data$q))


mod = cmdstanr::cmdstan_model(here::here("report","DDM","Stan Models","DDM.stan"))


fit <- mod$sample(
    data = data_stan,
    chains = 4,
    seed = 123,
    parallel_chains = 4,
    adapt_delta = 0.9,
    max_treedepth = 12)
```

```
## Running MCMC with 4 parallel chains...
##
## Chain 1 Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 2 Iteration:    1 / 2000 [  0%]  (Warmup)
```

```
## Chain 3 Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 4 Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 1 Iteration:  100 / 2000 [  5%]  (Warmup)
## Chain 2 Iteration:  100 / 2000 [  5%]  (Warmup)
## Chain 3 Iteration:  100 / 2000 [  5%]  (Warmup)
## Chain 4 Iteration:  100 / 2000 [  5%]  (Warmup)
## Chain 1 Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 2 Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 3 Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 4 Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 1 Iteration:  300 / 2000 [ 15%]  (Warmup)
## Chain 2 Iteration:  300 / 2000 [ 15%]  (Warmup)
## Chain 3 Iteration:  300 / 2000 [ 15%]  (Warmup)
## Chain 4 Iteration:  300 / 2000 [ 15%]  (Warmup)
## Chain 1 Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 2 Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 3 Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 4 Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 1 Iteration:  500 / 2000 [ 25%]  (Warmup)
## Chain 2 Iteration:  500 / 2000 [ 25%]  (Warmup)
## Chain 3 Iteration:  500 / 2000 [ 25%]  (Warmup)
## Chain 4 Iteration:  500 / 2000 [ 25%]  (Warmup)
## Chain 1 Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 2 Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 3 Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 4 Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 1 Iteration:  700 / 2000 [ 35%]  (Warmup)
## Chain 2 Iteration:  700 / 2000 [ 35%]  (Warmup)
## Chain 3 Iteration:  700 / 2000 [ 35%]  (Warmup)
## Chain 4 Iteration:  700 / 2000 [ 35%]  (Warmup)
## Chain 1 Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 2 Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 3 Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 4 Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 1 Iteration:  900 / 2000 [ 45%]  (Warmup)
## Chain 2 Iteration:  900 / 2000 [ 45%]  (Warmup)
## Chain 3 Iteration:  900 / 2000 [ 45%]  (Warmup)
## Chain 4 Iteration:  900 / 2000 [ 45%]  (Warmup)
## Chain 1 Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 1 Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 2 Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 2 Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 3 Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 3 Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 4 Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 4 Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 1 Iteration: 1100 / 2000 [ 55%]  (Sampling)
## Chain 2 Iteration: 1100 / 2000 [ 55%]  (Sampling)
## Chain 3 Iteration: 1100 / 2000 [ 55%]  (Sampling)
## Chain 4 Iteration: 1100 / 2000 [ 55%]  (Sampling)
## Chain 1 Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 2 Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 3 Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 4 Iteration: 1200 / 2000 [ 60%]  (Sampling)
```

```
## Chain 1 Iteration: 1300 / 2000 [ 65%]  (Sampling)
## Chain 3 Iteration: 1300 / 2000 [ 65%]  (Sampling)
## Chain 2 Iteration: 1300 / 2000 [ 65%]  (Sampling)
## Chain 4 Iteration: 1300 / 2000 [ 65%]  (Sampling)
## Chain 1 Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 3 Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 2 Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 4 Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 1 Iteration: 1500 / 2000 [ 75%]  (Sampling)
## Chain 3 Iteration: 1500 / 2000 [ 75%]  (Sampling)
## Chain 4 Iteration: 1500 / 2000 [ 75%]  (Sampling)
## Chain 2 Iteration: 1500 / 2000 [ 75%]  (Sampling)
## Chain 1 Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 3 Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 4 Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 2 Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 1 Iteration: 1700 / 2000 [ 85%]  (Sampling)
## Chain 3 Iteration: 1700 / 2000 [ 85%]  (Sampling)
## Chain 4 Iteration: 1700 / 2000 [ 85%]  (Sampling)
## Chain 2 Iteration: 1700 / 2000 [ 85%]  (Sampling)
## Chain 1 Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 3 Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 4 Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 2 Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 1 Iteration: 1900 / 2000 [ 95%]  (Sampling)
## Chain 3 Iteration: 1900 / 2000 [ 95%]  (Sampling)
## Chain 4 Iteration: 1900 / 2000 [ 95%]  (Sampling)
## Chain 1 Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 2 Iteration: 1900 / 2000 [ 95%]  (Sampling)
## Chain 3 Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 1 finished in 6.2 seconds.
## Chain 3 finished in 6.2 seconds.
## Chain 4 Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 4 finished in 6.2 seconds.
## Chain 2 Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 2 finished in 6.4 seconds.
##
## All 4 chains finished successfully.
## Mean chain execution time: 6.2 seconds.
## Total execution time: 6.6 seconds.
```

```
variables = c("alpha","tau","beta","delta")
```

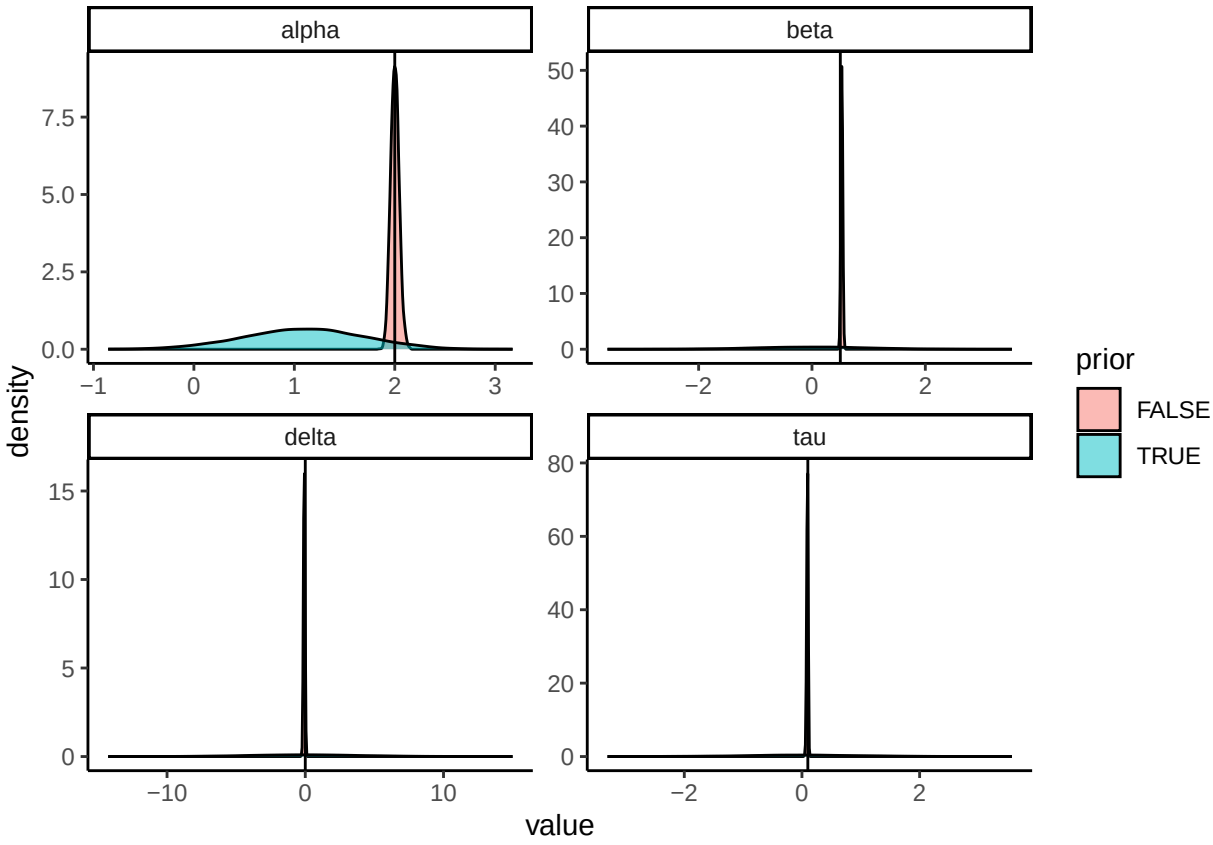
Lets look at the summary of the model

```
flextable::flextable(fit$summary(variables) %>%
                       mutate_if(is.numeric, round, digits = 2) %>%
                       inner_join(.,reals) %>%
                       head(4))
```

variable	mean	median	sd	mad	q5	q95	rhat	ess_bulk	ess_tail
alpha	2.00	2.00	0.04	0.04	1.93	2.07	1	3,074.18	2,607.01
tau	0.09	0.09	0.01	0.01	0.07	0.11	1	3,104.85	2,239.64
beta	0.53	0.53	0.02	0.02	0.50	0.55	1	2,902.54	2,713.75
delta	-0.08	-0.08	0.05	0.05	-0.17	0.01	1	2,606.39	2,732.52

Prior posterior updates

```
posteriors = as_draws_df(fit) %>% dplyr::select(any_of(names(parameters))) %>% mutate(prior = F)
priors = as_draws_df(fit) %>% dplyr::select(starts_with("prior_")) %>% rename_with(~gsub("^prior_", "",

rbind(posteriors,priors) %>%
  pivot_longer(cols = -prior) %>%
  ggplot(aes(x = value, fill = prior))+
  geom_density(alpha = 0.5)+
  theme_classic()+
  facet_wrap(~name, scales = "free")+
  geom_vline(data = parameters %>% dplyr::select(-trials) %>% pivot_longer(everything()), aes(xintercep
```



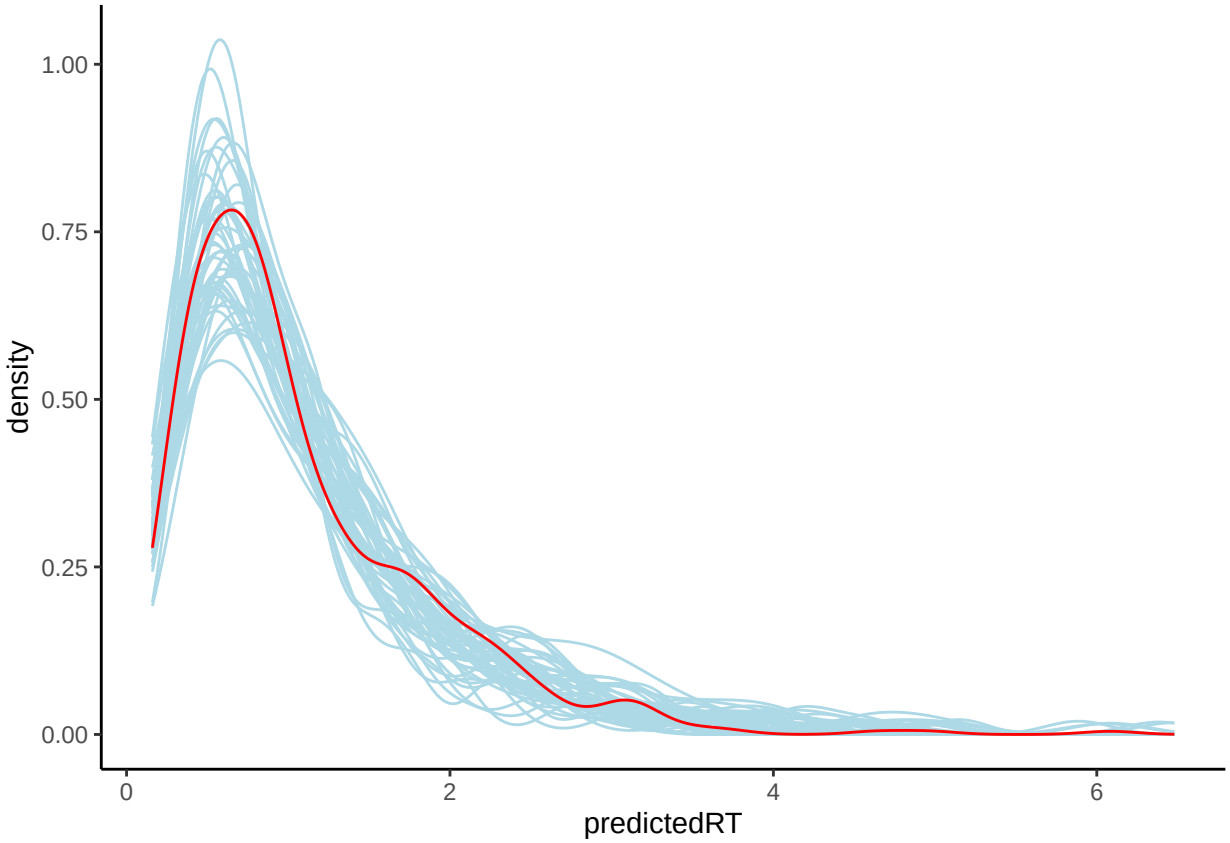
Posterior predictive checks

```r
library(posterior)

n_check = 50

get_pp = function(input){
  variables = c("alpha","tau","beta","delta")

  draww = input$draww

  parameters = as_draws_df(fit$draws(variables)) %>%
    dplyr::select(all_of(variables)) %>%
    mutate(draw = 1:nrow(.)) %>% slice(draww)

  parameters$times = input$times

  df = parameters %>%
    rowwise() %>%
    mutate(predictedRT = list(RWiener::rwiener(times,alpha,tau,beta,delta)[[1]]),
           predictedresp = list(RWiener::rwiener(times,alpha,tau,beta,delta)[[2]]),
           draw = draww)

  returndf = data.frame(predictedRT = unlist(df$predictedRT), predictedresp = unlist(df$predictedresp),

  return(list(returndf))
}

draww = rbinom(n_check,4000,extraDistr::rprop(n_check,1,0.5))


parameters = expand.grid(draww = draww,
                         times = 100) %>%
  mutate(id = 1:nrow(.))

data_list <- split(parameters, parameters$id)

possfit_model = possibly(.f = get_pp, otherwise = "Error")

results <- future_map(data_list, ~possfit_model(.x), .progress = TRUE, .options = furrr_options(seed =

error_indices <- which(results == "Error")

unique(error_indices)
```

```
## [1] 12
```

```r
results2 = results[results != "Error"]


rts = map_dfr(results2,1)

rts %>% ggplot()+
  geom_density(aes(x = predictedRT, group = draw), col = "lightblue")+
  geom_density(data = data, aes(x = q), col = "red")+
```

```
  theme_classic()
```



Lets do parameter recovery on this model!

First we make a function that does the following: - First we simulate data given some parameter values: - Then we fit the model using stan to get estimated parameter values - Then we extract these estimated and simulated values aswell as diagnostics for the models fit.

```
fit_model = function(parameters){

  # Simulate
  id = parameters$id

  data = rwiener(n = parameters$trials,
          alpha = parameters$alpha,
          delta = parameters$delta,
          beta = parameters$beta,
          tau = parameters$tau)

  data_stan = list(trials = nrow(data),
                   RT = data$q,
                   resp = ifelse(data$resp == "lower",0,1),
                   minRT = min(data$q))
```

```
  # Fit model
  mod = cmdstanr::cmdstan_model(here::here("report","DDM","Stan Models","DDM.stan"))


  fit <- mod$sample(
      data = data_stan,
      chains = 4,
      refresh = 0,
      parallel_chains = 4,
      adapt_delta = 0.9,
      max_treedepth = 12)

  #Extract parameters and diagnosics

  posteriors = as_draws_df(fit$summary()) %>% dplyr::filter(variable %in% names(parameters))
  diag = data.frame(fit$diagnostic_summary(), id)

  data = posteriors %>% mutate(num_div = diag$num_divergent,
                               tree_depth = diag$num_max_treedepth,
                               real_alpha = parameters$alpha,
                               real_delta = parameters$delta,
                               real_beta = parameters$beta,
                               real_tau = parameters$tau,
                               trials = parameters$trials,
                               id = id) %>% select(-contains("."))
  return(list(data, diag))
}
```

Defining ranges for our parameter estimates

```
trials = seq(50,200,by = 50)
alpha = seq(1,4,by = 1)
delta = seq(-3,3,by = 1)
beta = seq(0.2,0.8,by = 0.1)
tau = seq(0.1,0.3,by = 0.1)

replicate = 1:1

parameters = expand.grid(alpha = alpha,
                         delta = delta,
                         beta = beta,
                         tau = tau,
                         trials = trials,
                         replicate = replicate) %>%
  mutate(id = 1:nrow(.))

data_list <- split(parameters, parameters$id)

load(here::here("report","DDM","Workspace","DDM parameterrecovery.RData"))
```

Parallelizing the procedure to speed up the process!

```
#cores = availableCores()-1
#
# plan(multisession, workers = 4)
#
# possfit_model = possibly(.f = fit_model, otherwise = "Error")
#
# results <- future_map(data_list, ~possfit_model(.x), .progress = TRUE, .options = furrr_options(seed =
load(here::here("report","DDM","Workspace","DDM parameterrecovery.RData"))
```

Parameter recover results!

We start with the models that caused errors if any.

```
error_indices <- which(results == "Error")

unique(error_indices)
```

```
## integer(0)
```

```
results2 = results[results != "Error"]
```

None! As expected but good to see

Lets look at the divergences and max treedpeth of the models fit.

```
divergence = map_dfr(results2, 2)

divergence %>% median_qi(num_divergent)
```

```
## # A tibble: 1 x 6
##   num_divergent .lower .upper .width .point .interval
##           <dbl>  <dbl>  <dbl>  <dbl> <chr>  <chr>
## 1             0      0      0   0.95 median qi
```

There are also none which is good!

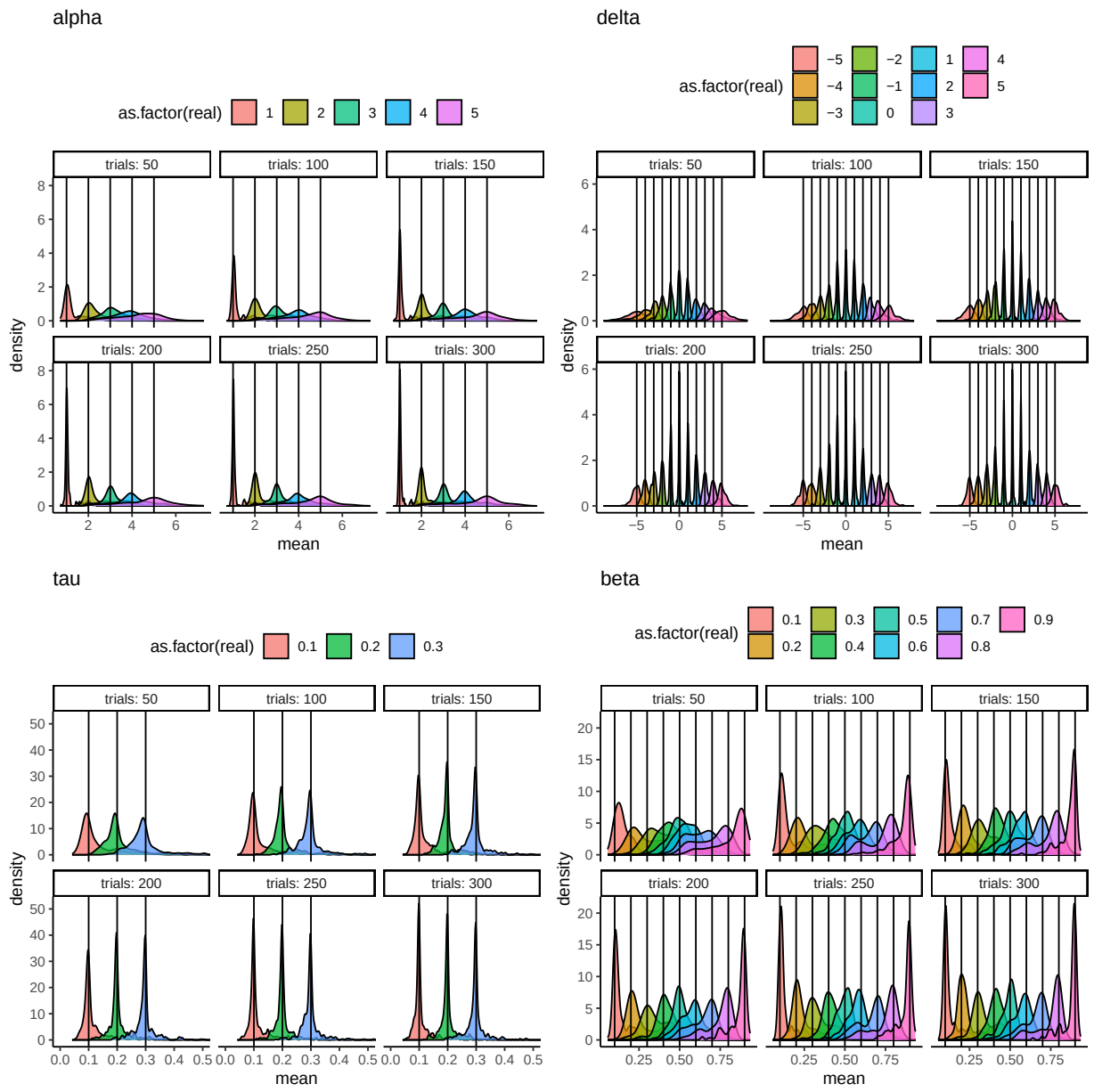
Visualizing Parameter recovery!

single variable plots vs trials!

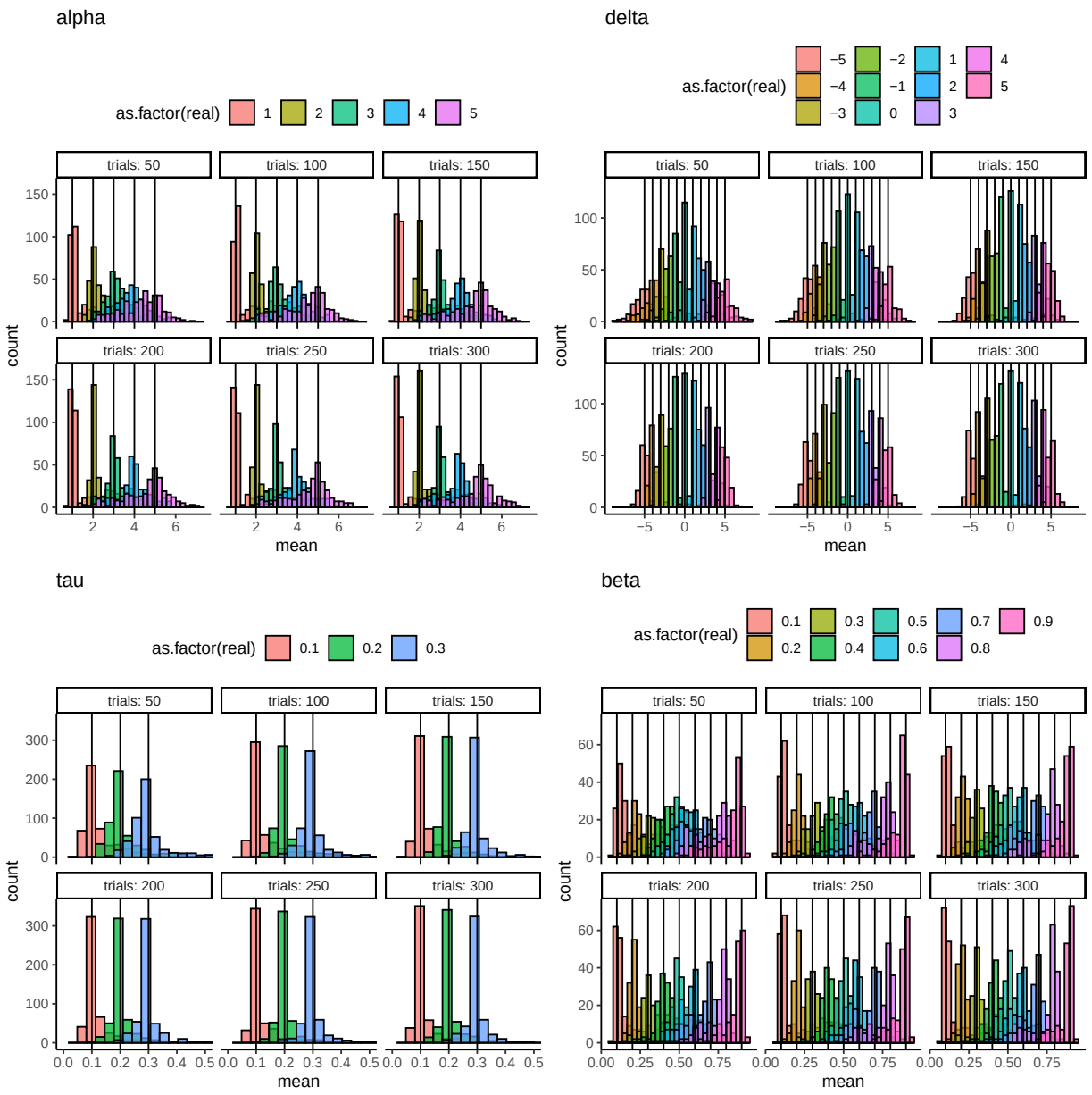
Now we can look at the parameter values! Estimated vs simulated

```
params = map_dfr(results2, 1)

variables = c("alpha","delta","tau","beta")

get_parameter_plots = function(variables, histogram){
```

```
  plot_list = list()
  for(variable1 in variables){
    plot = params %>% filter(variable == variable1) %>%
      mutate(real = get(paste0("real_",variable1))) %>%
      ggplot(aes(x = mean, fill = as.factor(real)))+
      {if(histogram)geom_histogram(alpha = 0.75, position = "identity", col = "black",bins = 30)}+
      {if(!histogram)geom_density(alpha = 0.75)}+
      theme_classic()+
      geom_vline(aes(xintercept = real))+
      facet_wrap(~trials, labeller = label_both)+
      {if(variable1 == "tau")coord_cartesian(xlim = c(0,.5))}+
      ggtitle(variable1)+
      theme(legend.position = "top")

    plot_list[[variable1]] = plot
  }
  return(plot_list)
}

plots = get_parameter_plots(variables = variables,histogram = FALSE)

(plots[["alpha"]]+plots[["delta"]])/(plots[["tau"]]+plots[["beta"]])
```

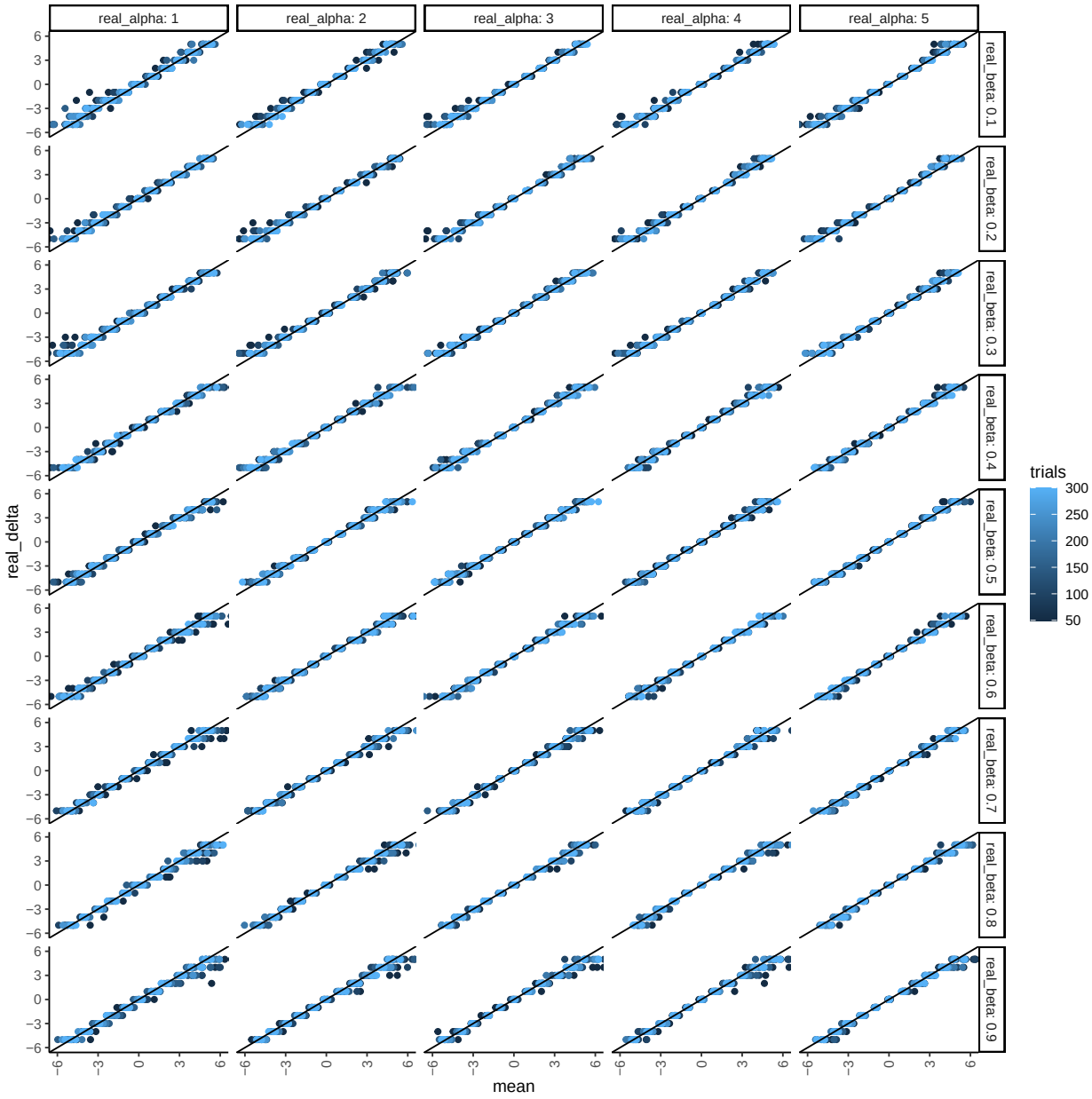



```
plots_hist = get_parameter_plots(variables = variables,histogram = TRUE)

(plots_hist[["alpha"]]+plots_hist[["delta"]])/(plots_hist[["tau"]]+plots_hist[["beta"]])
```

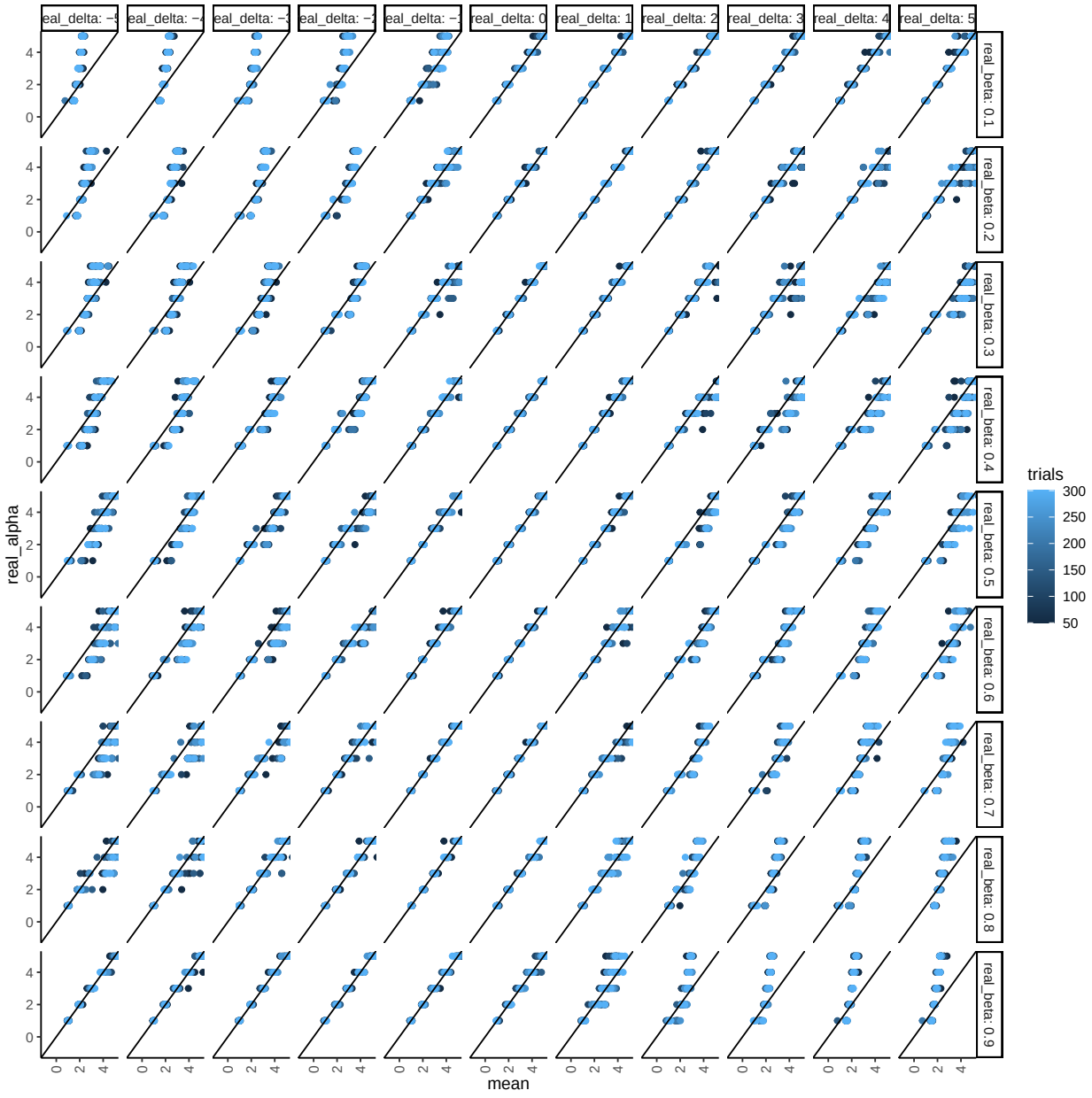


Scatter plots of interactions between variables

```
params %>%
  mutate_if(is.numeric, round, digits = 2) %>%
  filter(variable == "delta") %>%
        ggplot(aes(x = mean, y = real_delta, col = trials))+
        facet_grid(real_beta~real_alpha, labeller = label_both, scales = "free")+
        theme_classic()+
  geom_point(aes())+geom_abline(slope = 1, intercept = 0)+
  coord_cartesian(ylim = c(-6, 6), xlim = c(-6,6))+ theme(axis.text.x = element_text(angle = 90, vjust =
```




```
params %>%
  mutate_if(is.numeric, round, digits = 2) %>%
  filter(variable == "alpha") %>%
        ggplot(aes(x = mean, y = real_alpha, col = trials))+
        facet_grid(real_beta~real_delta, labeller = label_both, scales = "free")+
        theme_classic()+
  geom_point(aes())+geom_abline(slope = 1, intercept = 0)+
  coord_cartesian(ylim = c(-1, 5), xlim = c(-1,5))+ theme(axis.text.x = element_text(angle = 90, vjust =
```



We can do better lets model this to see the interactions!!

Lets start with the most obvious case of the drift rate! Here we need to realize that both signs basically mean the same and from the plots above we see that they are identical we can therefore take the absolute value of real and estimated values!

```
variables = "delta"


m1 = params %>%
    mutate(across(all_of(contains("real_")), as.factor)) %>%
    mutate(real = .[[paste0("real_", variables)]]) %>%
    mutate(real2 = as.numeric(as.character(real)), trials = as.factor(trials)) %>%
```

```
    mutate(across(all_of(contains("real_")), as.numeric)) %>%
    rename(Trial_number = trials) %>%
    mutate(Trial_number = as.numeric(Trial_number)) %>%
    mutate(real2 = abs(real2), mean = abs(mean)) %>%
  filter(variable == variables)


model_gam = gamlss::gamlss(mean ~ real+Trial_number,
                           data = m1)
```

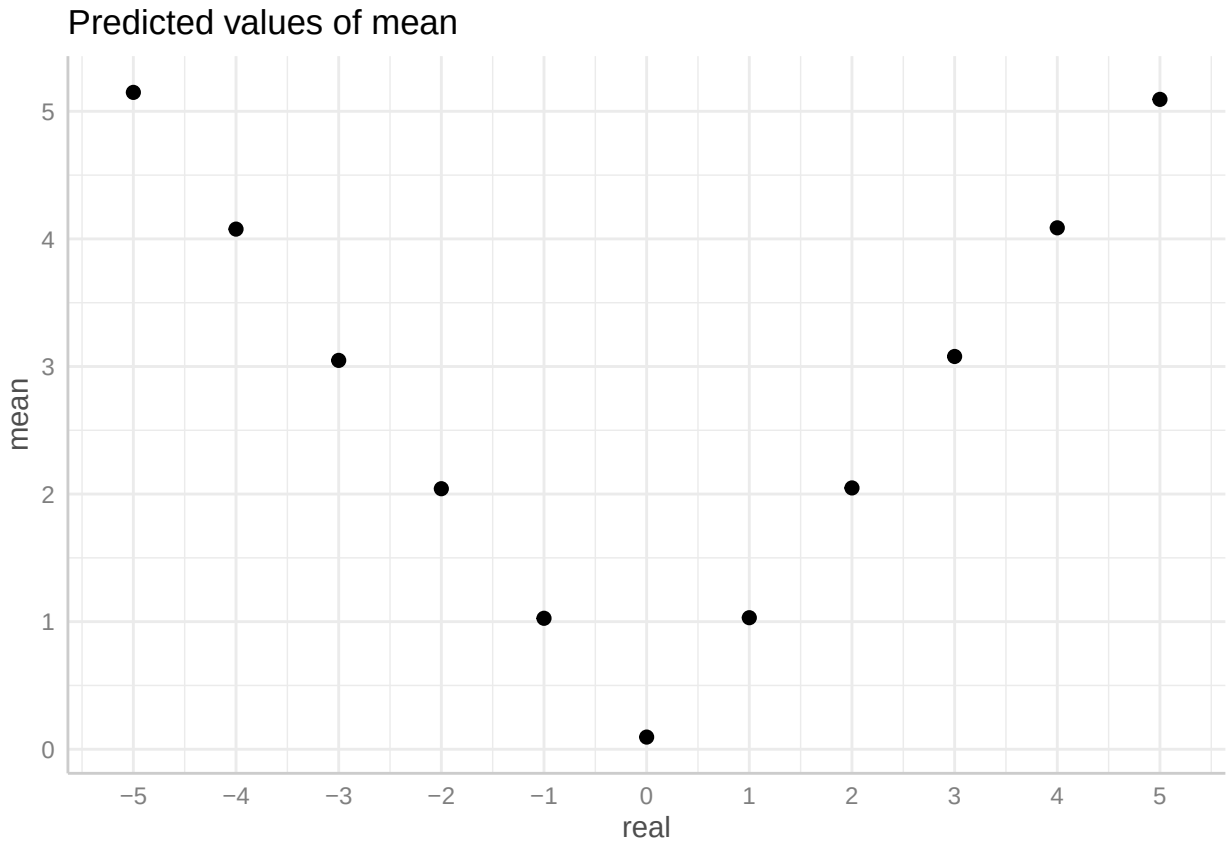
```
## GAMLSS-RS iteration 1: Global Deviance = 9579.915
## GAMLSS-RS iteration 2: Global Deviance = 9579.915
```

```
summary(model_gam)
```

```
## ******************************************************************
## Family:  c("NO", "Normal")
##
## Call:  gamlss::gamlss(formula = mean ~ real + Trial_number,      data = m1)
##
## Fitting method: RS()
##
## ------------------------------------------------------------------
## Mu link function:  identity
## Mu Coefficients:
##              Estimate Std. Error  t value Pr(>|t|)
## (Intercept)   5.22157    0.01711  305.198  < 2e-16 ***
## real-4       -1.07158    0.02058  -52.061  < 2e-16 ***
## real-3       -2.10061    0.02058 -102.055  < 2e-16 ***
## real-2       -3.10638    0.02058 -150.920  < 2e-16 ***
## real-1       -4.12192    0.02058 -200.258  < 2e-16 ***
## real0        -5.05315    0.02058 -245.501  < 2e-16 ***
## real1        -4.11750    0.02058 -200.044  < 2e-16 ***
## real2        -3.10034    0.02058 -150.626  < 2e-16 ***
## real3        -2.06985    0.02058 -100.561  < 2e-16 ***
## real4        -1.06151    0.02058  -51.572  < 2e-16 ***
## real5        -0.05469    0.02058   -2.657  0.00789 **
## Trial_number -0.02089    0.00257   -8.130 4.88e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## ------------------------------------------------------------------
## Sigma link function:  log
## Sigma Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) -0.881345   0.007491  -117.7   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## ------------------------------------------------------------------
## No. of observations in the fit:  8910
## Degrees of Freedom for the fit:  13
##       Residual Deg. of Freedom:  8897
```

```
##                       at cycle:  2
##
## Global Deviance:     9579.915
##              AIC:    9605.915
##              SBC:    9698.149
## ******************************************************************
```

```
plot(ggeffects::ggpredict(model_gam, terms = c("real")))+
  geom_point()
```



But this model doesn't really makes sense! It makes sense that the different simulated real values tightly correlate with the estimated values, but what do we make of the effect of Trial number?

The effect of Trial number basically here states that as we simulate with more trials the estimated mean gets shrunk a bit. Which makes sense given the estimated means (especially in higher simulated (real) values)

The right way to approach this, is to not model the mean of the normal distribution as trials increase but the standard deviation as trials increase:

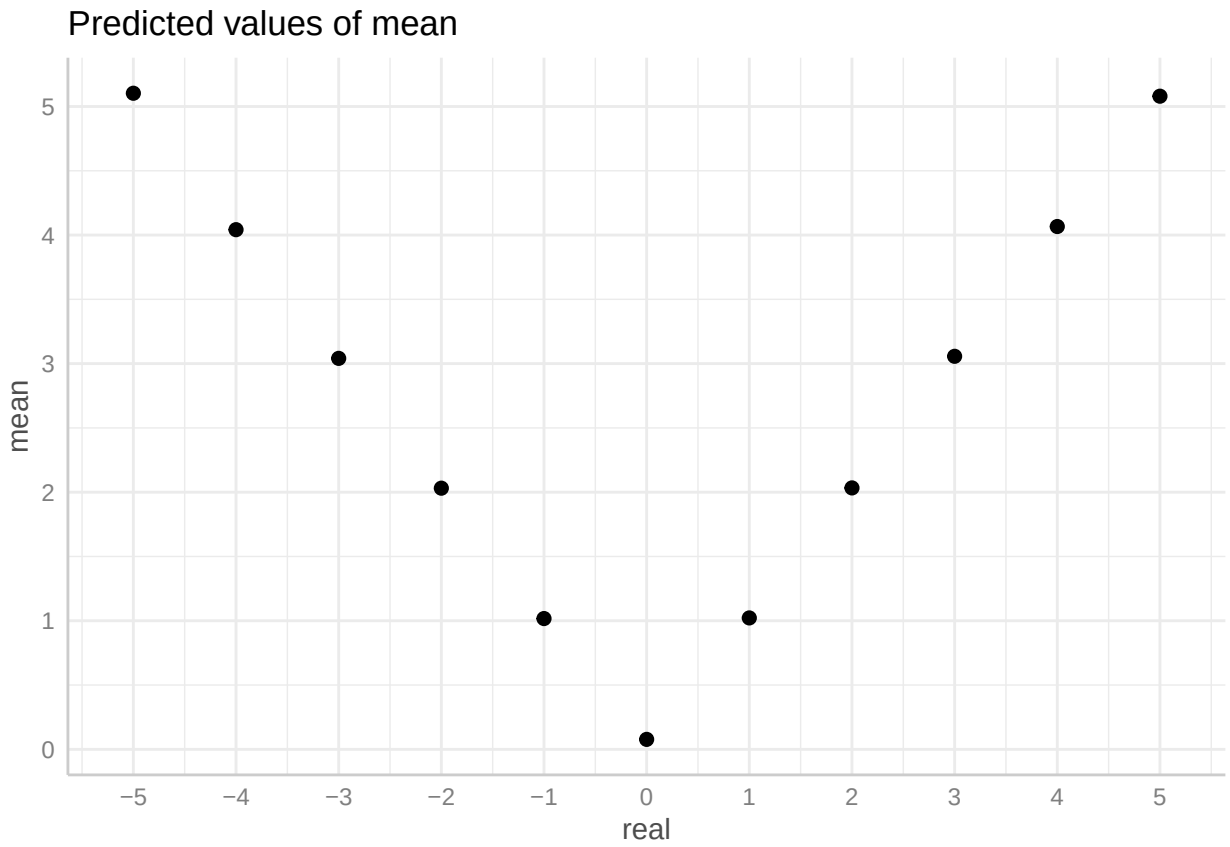
```
model_gam = gamlss::gamlss(mean ~ real,
                           sigma.formula = ~Trial_number,
                           data = m1)
```

```
## GAMLSS-RS iteration 1: Global Deviance = 7446.472
## GAMLSS-RS iteration 2: Global Deviance = 7407.654
## GAMLSS-RS iteration 3: Global Deviance = 7407.65
## GAMLSS-RS iteration 4: Global Deviance = 7407.65
```

```
summary(model_gam)
```

```
## ******************************************************************
## Family:  c("NO", "Normal")
##
## Call:  gamlss::gamlss(formula = mean ~ real, sigma.formula = ~Trial_number,
##     data = m1)
##
## Fitting method: RS()
##
## ------------------------------------------------------------------
## Mu link function:  identity
## Mu Coefficients:
##             Estimate Std. Error  t value Pr(>|t|)
## (Intercept)  5.10299    0.01157  440.992   <2e-16 ***
## real-4      -1.06117    0.01634  -64.959   <2e-16 ***
## real-3      -2.06218    0.01634 -126.172   <2e-16 ***
## real-2      -3.07179    0.01635 -187.931   <2e-16 ***
## real-1      -4.08593    0.01634 -250.001   <2e-16 ***
## real0       -5.02574    0.01634 -307.572   <2e-16 ***
## real1       -4.08102    0.01634 -249.692   <2e-16 ***
## real2       -3.06977    0.01634 -187.835   <2e-16 ***
## real3       -2.04604    0.01634 -125.229   <2e-16 ***
## real4       -1.03660    0.01634  -63.437   <2e-16 ***
## real5       -0.02294    0.01634   -1.403    0.161
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## ------------------------------------------------------------------
## Sigma link function:  log
## Sigma Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  -0.310124   0.016097  -19.27   <2e-16 ***
## Trial_number -0.198035   0.004071  -48.65   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## ------------------------------------------------------------------
## No. of observations in the fit:  8910
## Degrees of Freedom for the fit:  13
##       Residual Deg. of Freedom:  8897
##                       at cycle:  4
##
## Global Deviance:     7407.649
##             AIC:     7433.649
##             SBC:     7525.884
## ******************************************************************
```

```
plot(ggeffects::ggpredict(model_gam, terms = c("real")))+
  geom_point()
```



What we see here is that generally as we increase trials by 1 we decrease the standard deviation by $exp(-0.198)$ which translates into 0.18 % decrease in the standard deviation.

Now for the threshold parameter which was the trickiest to recover and depended heavily on the simulated value of both drift rate and starting bias!

```
variables = "alpha"


m1 = params %>%
    mutate(across(all_of(contains("real_")), as.factor)) %>%
    mutate(real = .[[paste0("real_", variables)]]) %>%
    mutate(real2 = as.numeric(as.character(real)), trials = as.factor(trials)) %>%
    mutate(across(all_of(contains("real_")), as.character)) %>%
    mutate(across(all_of(contains("real_")), as.numeric)) %>%
    rename(Trial_number = trials) %>%
    mutate(Trial_number = as.numeric(Trial_number)) %>%
  filter(variable == variables)


model_gam = gamlss::gamlss(mean ~ real* real_beta * real_delta,
                           sigma.formula = ~Trial_number,
                           data = m1)
```

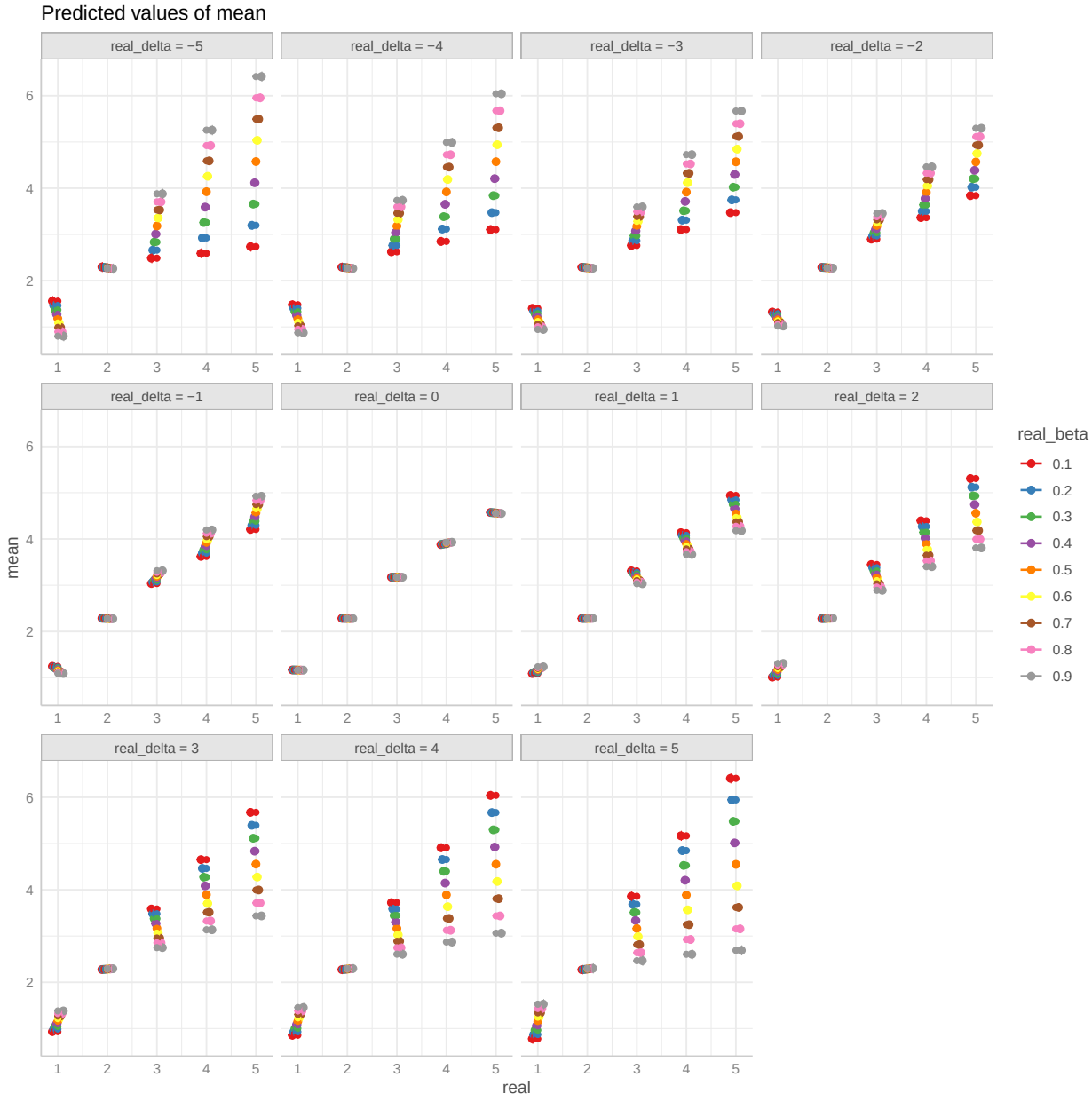
```
## GAMLSS-RS iteration 1: Global Deviance = 13898.45
## GAMLSS-RS iteration 2: Global Deviance = 13897.62
## GAMLSS-RS iteration 3: Global Deviance = 13897.62
```

```
summary(model_gam)
```

```
## ******************************************************************
## Family:  c("NO", "Normal")
##
## Call:  gamlss::gamlss(formula = mean ~ real * real_beta *
##     real_delta, sigma.formula = ~Trial_number, data = m1)
##
## Fitting method: RS()
##
## ------------------------------------------------------------------
## Mu link function:  identity
## Mu Coefficients:
##                               Estimate Std. Error t value Pr(>|t|)
## (Intercept)                  1.168e+00  2.722e-02  42.907  < 2e-16 ***
## real2                        1.116e+00  3.847e-02  29.015  < 2e-16 ***
## real3                        2.005e+00  3.848e-02  52.112  < 2e-16 ***
## real4                        2.705e+00  3.851e-02  70.239  < 2e-16 ***
## real5                        3.408e+00  3.854e-02  88.432  < 2e-16 ***
## real_beta                   -4.918e-03  4.834e-02  -0.102    0.919
## real_delta                  -9.689e-02  8.606e-03 -11.259  < 2e-16 ***
## real2:real_beta             -8.806e-05  6.837e-02  -0.001    0.999
## real3:real_beta              3.891e-03  6.837e-02   0.057    0.955
## real4:real_beta              6.885e-02  6.837e-02   1.007    0.314
## real5:real_beta             -2.155e-02  6.837e-02  -0.315    0.753
## real2:real_delta             9.366e-02  1.217e-02   7.693 1.59e-14 ***
## real3:real_delta             2.690e-01  1.217e-02  22.097  < 2e-16 ***
## real4:real_delta             4.197e-01  1.217e-02  34.499  < 2e-16 ***
## real5:real_delta             5.564e-01  1.217e-02  45.735  < 2e-16 ***
## real_beta:real_delta         1.884e-01  1.529e-02  12.318  < 2e-16 ***
## real2:real_beta:real_delta -1.803e-01  2.164e-02  -8.332  < 2e-16 ***
## real3:real_beta:real_delta -5.365e-01  2.163e-02 -24.806  < 2e-16 ***
## real4:real_beta:real_delta -8.416e-01  2.162e-02 -38.926  < 2e-16 ***
## real5:real_beta:real_delta -1.113e+00  2.162e-02 -51.460  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## ------------------------------------------------------------------
## Sigma link function:  log
## Sigma Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  -0.555401   0.017201 -32.289  < 2e-16 ***
## Trial_number -0.023900   0.004424  -5.402 6.75e-08 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## ------------------------------------------------------------------
## No. of observations in the fit:  8910
## Degrees of Freedom for the fit:  22
##       Residual Deg. of Freedom:  8888
##                       at cycle:  3
##
## Global Deviance:     13897.62
```

```
##              AIC:      13941.62
##              SBC:      14097.71
## ************************************************************************
```

```
plot(ggeffects::ggpredict(model_gam, terms = c("real","real_beta","real_delta [-5:5]")))+
  geom_point()
```



Other modelings opportunities!

The next thing one might think about is that when participants go through all these tasks they will inevitably become tried and lose focus / attention. We can think of a couple of ways to incorporate this into the modeling.

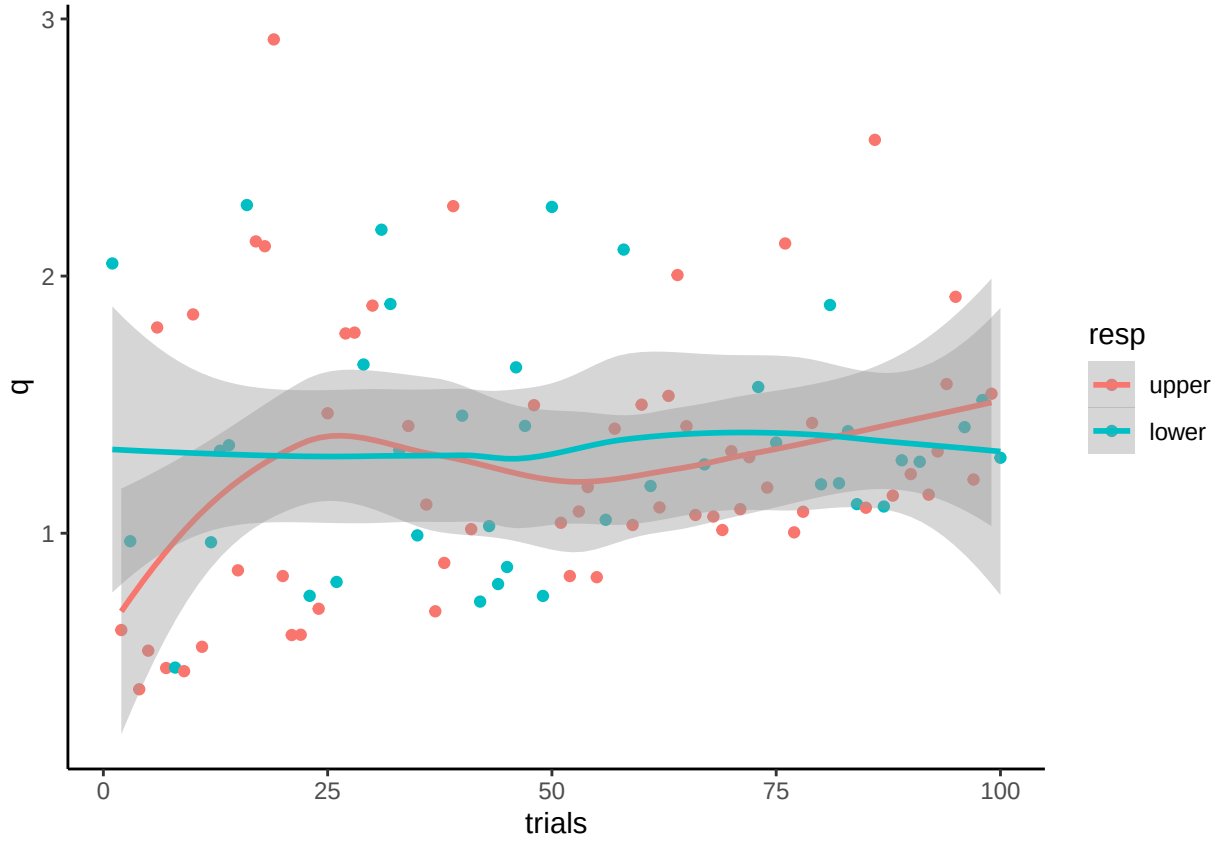
1) participants decision boundary decreases as trials increase.
2) participants' absolute drift rates decreases as trials increase.
3) participants' non-decision time increases as trials increases.

We would also expect that these effects would be somewhat mitigated by breaks (i.e a sudden shift in these parameters).

Lets just start with a linear decrease increase in these as trials increase.

```
parameters = data.frame(trials = 100,
                        alpha_0 = 2,
                        alpha_b1 = -0.01,
                        delta_0 = 0,
                        delta_b1 = 0,
                        beta = 0.5,
                        tau_0 = 0.2,
                        tau_b1 = 0.01)

fit_gdmm = function(parameters){

  trials = parameters$trials
  alpha_0 = parameters$alpha_0
  alpha_b1 = parameters$alpha_b1

  delta_0 = parameters$delta_0
  delta_b1 = parameters$delta_b1

  beta = parameters$beta
  tau_0 = parameters$tau_0
  tau_b1 = parameters$tau_b1


  alpha = array(NA,trials)
  delta = array(NA,trials)
  tau = array(NA,trials)

  resp = data.frame()
  for(i in 1:trials){
    alpha[i] = alpha_0 + alpha_b1 * i
    delta[i] = delta_0 + delta_b1 * i
    tau[i] = tau_0 + tau_b1 * i

    data = rwiener(n = 1,
                   beta = beta,
                   alpha = alpha[i],
                   tau = tau[i],
                   delta = delta[i])

    resp = rbind(resp,data)
  }
  resp$trials = 1:trials

  resp$alpha_0 = alpha_0
  resp$alpha_b1 = alpha_b1
```

```
  resp$delta_0 = delta_0
  resp$delta_b1 = delta_b1
  resp$tau_0 = tau_0
  resp$tau_b1 = tau_b1
  resp$beta = beta
  resp$id = parameters$id


  return(list(resp))
}


fit_gdmm(parameters)[[1]] %>%
  ggplot(aes(x = trials, y = q, col = resp))+
  geom_point()+
  theme_classic()+
  geom_smooth()
```



Now lets see this plot with different levels:

```
trials = 100
alpha_0 = 1
alpha_b1 = seq(-0.009, 0,length.out = 5)

delta_0 = 0
delta_b1 = 0
```

```r
beta = 0.5
tau_0 = 0.3
tau_b1 = seq(0, 0.001, length.out = 5)


parameters = expand.grid(alpha_0 = alpha_0,
                         alpha_b1 = alpha_b1,
                         delta_0 = delta_0,
                         delta_b1 = delta_b1,
                         beta = beta,
                         tau_0 = tau_0,
                         tau_b1 = tau_b1,
                         trials = trials) %>%
  mutate(id = 1:nrow(.))

data_list <- split(parameters, parameters$id)

#cores = availableCores()-120

plan(multisession, workers = 4)

possfit_model = possibly(.f = fit_gdmm, otherwise = "Error")

#results <- future_map(data_list, ~possfit_model(.x), .progress = TRUE, .options = furrr_options(seed =

#error_indices <- which(results == "Error")

#unique(error_indices)

#results2 = results[results != "Error"]
```

```r
# dd = map_dfr(results2,1)
#
# dd %>% ggplot(aes(x = trials, y = q)) +
#   geom_point()+
#   theme_classic()+
#   facet_grid(tau_b1~alpha_b1, labeller = label_both, scales = "free")+
#   geom_smooth(method = "lm")
```