fitting ddm

2023-10-22

R Markdown

Lets try and fit this in Stan!

```
trials = 500
alpha = 2
delta = 0
beta = 0.5
tau = 0.1


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

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

data_stan = list(Nu = nrow(data %>% filter(resp == "upper")),
                 Nl = nrow(data %>% filter(resp == "lower")),
                 RTu = data %>% filter(resp == "upper") %>% .$q,
                 RTl = data %>% filter(resp == "lower") %>% .$q,
                 minRT = min(data$q),
                 run_estimation = 1)


mod = cmdstanr::cmdstan_model(here::here("stan_scripts","HDDM.stan"))


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

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

## Chain 1 Informational Message: The current Metropolis proposal is about to be rejected because of th

## Chain 1 Exception: wiener_lpdf: Boundary separation is 0, but must be positive finite! (in '/tmp/Rtm
```

```
## Chain 1 If this warning occurs sporadically, such as for highly constrained variable types like cova
## Chain 1 but if this warning occurs often then your model may be either severely ill-conditioned or mi
## Chain 1
## Chain 1 Informational Message: The current Metropolis proposal is about to be rejected because of the
## Chain 1 Exception: wiener_lpdf: Boundary separation is 0, but must be positive finite! (in '/tmp/Rtm
## Chain 1 If this warning occurs sporadically, such as for highly constrained variable types like cova
## Chain 1 but if this warning occurs often then your model may be either severely ill-conditioned or mi
## Chain 1
## Chain 1 Informational Message: The current Metropolis proposal is about to be rejected because of the
## Chain 1 Exception: wiener_lpdf: Boundary separation is 0, but must be positive finite! (in '/tmp/Rtm
## Chain 1 If this warning occurs sporadically, such as for highly constrained variable types like cova
## Chain 1 but if this warning occurs often then your model may be either severely ill-conditioned or mi
## Chain 1
## Chain 1 Informational Message: The current Metropolis proposal is about to be rejected because of the
## Chain 1 Exception: wiener_lpdf: Boundary separation is 0, but must be positive finite! (in '/tmp/Rtm
## Chain 1 If this warning occurs sporadically, such as for highly constrained variable types like cova
## Chain 1 but if this warning occurs often then your model may be either severely ill-conditioned or mi
## Chain 1
## Chain 2 Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 2 Informational Message: The current Metropolis proposal is about to be rejected because of the
## Chain 2 Exception: wiener_lpdf: Boundary separation is 0, but must be positive finite! (in '/tmp/Rtm
## Chain 2 If this warning occurs sporadically, such as for highly constrained variable types like cova
## Chain 2 but if this warning occurs often then your model may be either severely ill-conditioned or mi
## Chain 2
```

```
## Chain 2 Informational Message: The current Metropolis proposal is about to be rejected because of the
## Chain 2 Exception: wiener_lpdf: Boundary separation is 0, but must be positive finite! (in '/tmp/Rtmp
## Chain 2 If this warning occurs sporadically, such as for highly constrained variable types like covar
## Chain 2 but if this warning occurs often then your model may be either severely ill-conditioned or mi
## Chain 2
## Chain 2 Informational Message: The current Metropolis proposal is about to be rejected because of the
## Chain 2 Exception: wiener_lpdf: Random variable  = 0.195651, but must be greater than nondecision tim
## Chain 2 If this warning occurs sporadically, such as for highly constrained variable types like covar
## Chain 2 but if this warning occurs often then your model may be either severely ill-conditioned or mi
## Chain 2
## Chain 2 Informational Message: The current Metropolis proposal is about to be rejected because of the
## Chain 2 Exception: wiener_lpdf: Random variable  = 0.195651, but must be greater than nondecision tim
## Chain 2 If this warning occurs sporadically, such as for highly constrained variable types like covar
## Chain 2 but if this warning occurs often then your model may be either severely ill-conditioned or mi
## Chain 2
## Chain 3 Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 3 Informational Message: The current Metropolis proposal is about to be rejected because of the
## Chain 3 Exception: wiener_lpdf: Boundary separation is 0, but must be positive finite! (in '/tmp/Rtmp
## Chain 3 If this warning occurs sporadically, such as for highly constrained variable types like covar
## Chain 3 but if this warning occurs often then your model may be either severely ill-conditioned or mi
## Chain 3
## Chain 4 Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 4 Informational Message: The current Metropolis proposal is about to be rejected because of the
## Chain 4 Exception: wiener_lpdf: Boundary separation is 0, but must be positive finite! (in '/tmp/Rtmp
```

```
## Chain 4 If this warning occurs sporadically, such as for highly constrained variable types like cova

## Chain 4 but if this warning occurs often then your model may be either severely ill-conditioned or m

## Chain 4

## Chain 4 Informational Message: The current Metropolis proposal is about to be rejected because of th

## Chain 4 Exception: wiener_lpdf: Boundary separation is 0, but must be positive finite! (in '/tmp/Rtm

## Chain 4 If this warning occurs sporadically, such as for highly constrained variable types like cova

## Chain 4 but if this warning occurs often then your model may be either severely ill-conditioned or m

## Chain 4

## Chain 4 Informational Message: The current Metropolis proposal is about to be rejected because of th

## Chain 4 Exception: wiener_lpdf: Boundary separation is 0, but must be positive finite! (in '/tmp/Rtm

## Chain 4 If this warning occurs sporadically, such as for highly constrained variable types like cova

## Chain 4 but if this warning occurs often then your model may be either severely ill-conditioned or m

## Chain 4

## Chain 2 Iteration:  100 / 2000 [  5%]  (Warmup)
## Chain 3 Iteration:  100 / 2000 [  5%]  (Warmup)
## Chain 4 Iteration:  100 / 2000 [  5%]  (Warmup)
## Chain 2 Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 3 Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 4 Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 2 Iteration:  300 / 2000 [ 15%]  (Warmup)
## Chain 3 Iteration:  300 / 2000 [ 15%]  (Warmup)
## Chain 4 Iteration:  300 / 2000 [ 15%]  (Warmup)
## Chain 2 Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 3 Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 4 Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 2 Iteration:  500 / 2000 [ 25%]  (Warmup)
## Chain 3 Iteration:  500 / 2000 [ 25%]  (Warmup)
## Chain 4 Iteration:  500 / 2000 [ 25%]  (Warmup)
## Chain 2 Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 3 Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 4 Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 2 Iteration:  700 / 2000 [ 35%]  (Warmup)
## Chain 3 Iteration:  700 / 2000 [ 35%]  (Warmup)
## Chain 4 Iteration:  700 / 2000 [ 35%]  (Warmup)
## Chain 2 Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 3 Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 4 Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 3 Iteration:  900 / 2000 [ 45%]  (Warmup)
```

```
## Chain 2 Iteration:  900 / 2000 [ 45%]  (Warmup)
## Chain 4 Iteration:  900 / 2000 [ 45%]  (Warmup)
## Chain 3 Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 3 Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 2 Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 2 Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 4 Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 4 Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 3 Iteration: 1100 / 2000 [ 55%]  (Sampling)
## Chain 2 Iteration: 1100 / 2000 [ 55%]  (Sampling)
## Chain 4 Iteration: 1100 / 2000 [ 55%]  (Sampling)
## Chain 3 Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 2 Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 4 Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 3 Iteration: 1300 / 2000 [ 65%]  (Sampling)
## Chain 2 Iteration: 1300 / 2000 [ 65%]  (Sampling)
## Chain 4 Iteration: 1300 / 2000 [ 65%]  (Sampling)
## Chain 2 Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 3 Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 4 Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 3 Iteration: 1500 / 2000 [ 75%]  (Sampling)
## Chain 1 Iteration:  100 / 2000 [  5%]  (Warmup)
## Chain 2 Iteration: 1500 / 2000 [ 75%]  (Sampling)
## Chain 4 Iteration: 1500 / 2000 [ 75%]  (Sampling)
## Chain 3 Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 1 Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 2 Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 4 Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 1 Iteration:  300 / 2000 [ 15%]  (Warmup)
## Chain 2 Iteration: 1700 / 2000 [ 85%]  (Sampling)
## Chain 3 Iteration: 1700 / 2000 [ 85%]  (Sampling)
## Chain 4 Iteration: 1700 / 2000 [ 85%]  (Sampling)
## Chain 1 Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 2 Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 3 Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 4 Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 1 Iteration:  500 / 2000 [ 25%]  (Warmup)
## Chain 2 Iteration: 1900 / 2000 [ 95%]  (Sampling)
## Chain 3 Iteration: 1900 / 2000 [ 95%]  (Sampling)
## Chain 4 Iteration: 1900 / 2000 [ 95%]  (Sampling)
## Chain 1 Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 2 Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 3 Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 4 Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 2 finished in 7.1 seconds.
## Chain 3 finished in 7.1 seconds.
## Chain 4 finished in 7.1 seconds.
## Chain 1 Iteration:  700 / 2000 [ 35%]  (Warmup)
## Chain 1 Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 1 Iteration:  900 / 2000 [ 45%]  (Warmup)
## Chain 1 Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 1 Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 1 Iteration: 1100 / 2000 [ 55%]  (Sampling)
## Chain 1 Iteration: 1200 / 2000 [ 60%]  (Sampling)
```

```
## Chain 1 Iteration: 1300 / 2000 [ 65%]  (Sampling)
## Chain 1 Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 1 Iteration: 1500 / 2000 [ 75%]  (Sampling)
## Chain 1 Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 1 Iteration: 1700 / 2000 [ 85%]  (Sampling)
## Chain 1 Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 1 Iteration: 1900 / 2000 [ 95%]  (Sampling)
## Chain 1 Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 1 finished in 12.2 seconds.
##
## All 4 chains finished successfully.
## Mean chain execution time: 8.4 seconds.
## Total execution time: 12.5 seconds.
```

Lets look at the summary of the model

```
flextable::flextable(fit$summary() %>% mutate_if(is.numeric, round, digits = 2) %>% head(6))
```

```
## Warning: fonts used in ‘flextable‘ are ignored because the ‘pdflatex‘ engine is
## used and not ‘xelatex‘ or ‘lualatex‘. You can avoid this warning by using the
## ‘set_flextable_defaults(fonts_ignore=TRUE)‘ command or use a compatible engine
## by defining ‘latex_engine: xelatex‘ in the YAML header of the R Markdown
## document.
```

variable	mean	median	sd	mad	q5	q95	rhat	ess_bulk	ess_tail
lp__	-763.96	-763.67	1.42	1.24	-766.68	-762.28	1	1,780.82	2,355.58
alpha	1.96	1.96	0.04	0.04	1.90	2.03	1	2,885.07	2,539.75
beta	0.47	0.47	0.02	0.02	0.45	0.50	1	2,698.50	2,912.42
delta	0.03	0.03	0.06	0.06	-0.06	0.12	1	2,338.20	2,893.84
tau_raw	0.32	0.33	0.24	0.24	-0.08	0.71	1	2,941.11	3,136.97
tau	0.11	0.11	0.01	0.01	0.09	0.13	1	2,941.11	3,136.97

Prior posterior updates

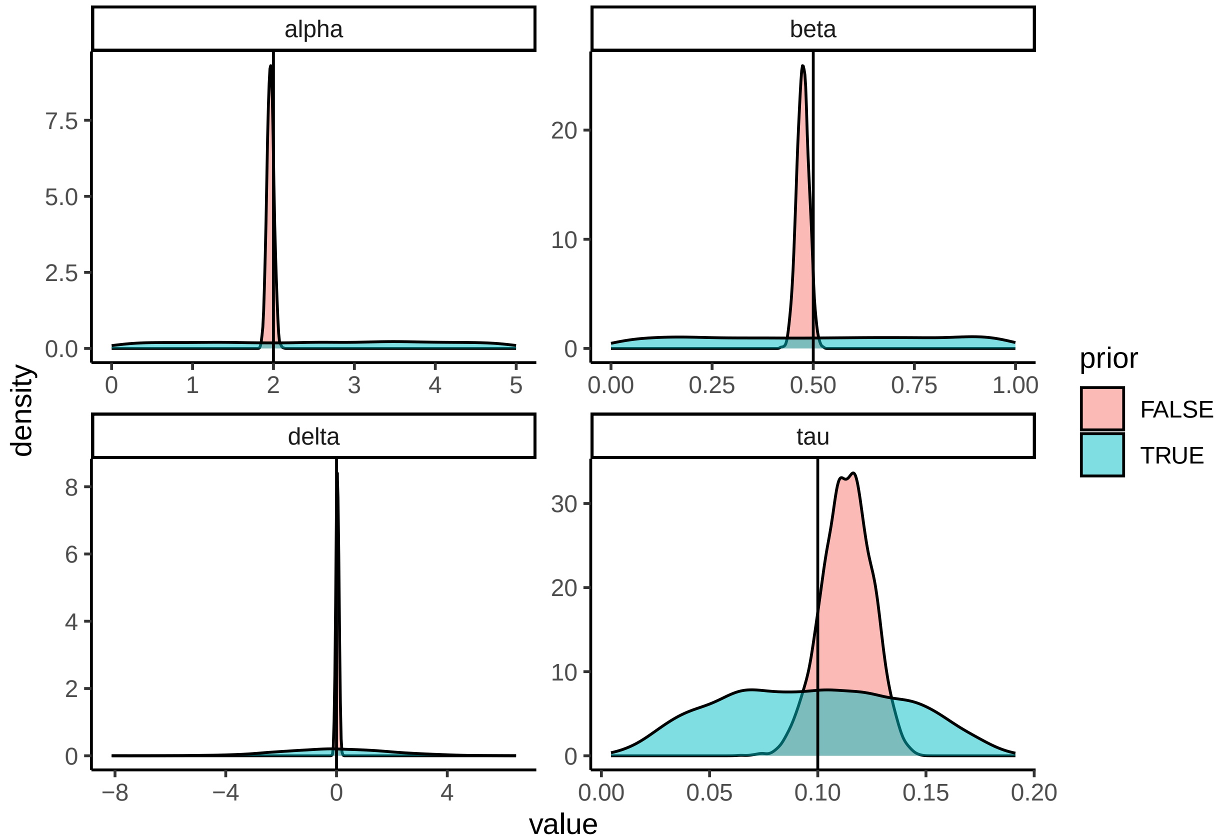
```
posteriors = as_draws_df(fit) %>% dplyr::select(any_of(names(parameters))) %>% mutate(prior = F)
```

```
## Warning: Dropping 'draws_df' class as required metadata was removed.
```

```
priors = as_draws_df(fit) %>% dplyr::select(starts_with("prior_")) %>% rename_with(~gsub("^prior_", "",
```

```
## Warning: Dropping 'draws_df' class as required metadata was removed.
```

```
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 %>% select(-trials) %>% pivot_longer(everything()), aes(xintercept = valu
```



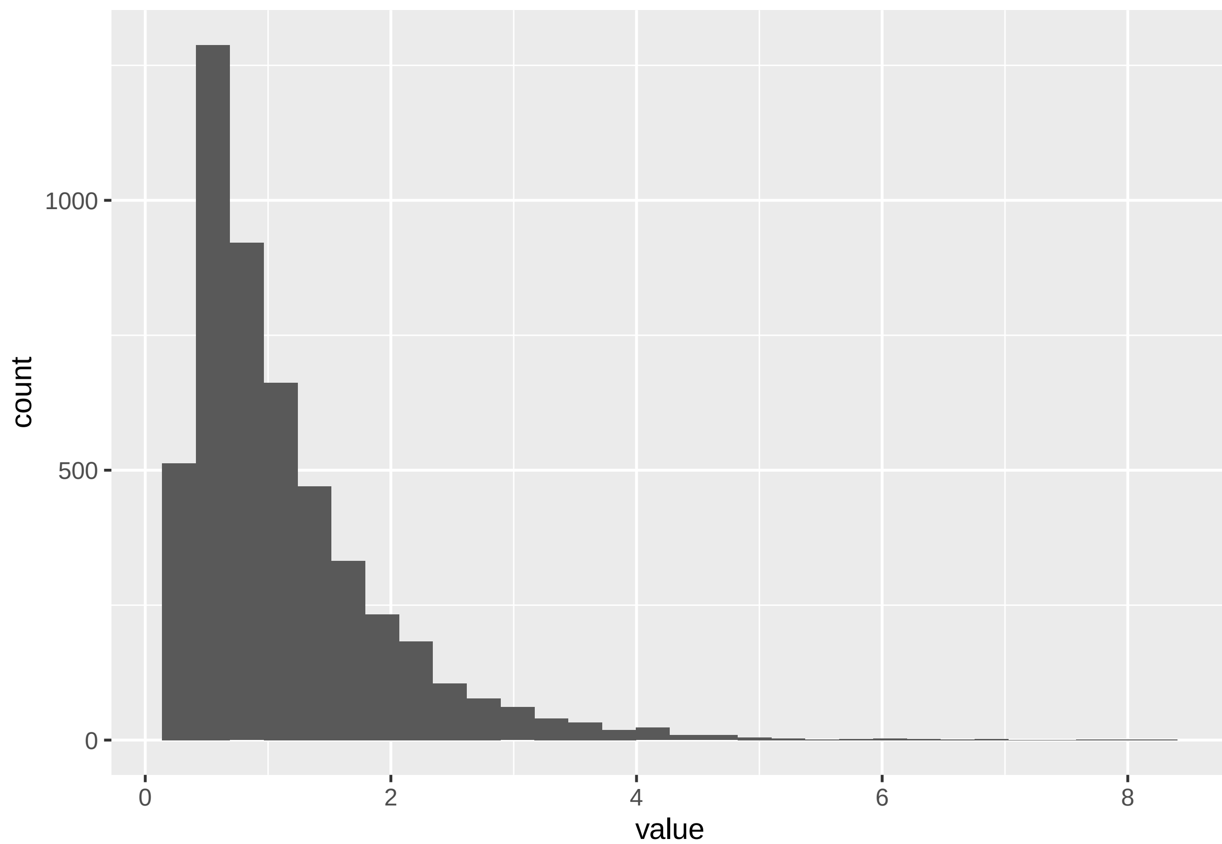
Posterior predictive checks

```
rt_pattern <- "out\\[\\d+,1\\]"
choice_pattern <- "out\\[\\d+,2\\]"


Rts = fit$summary() %>%
  filter(grepl(rt_pattern, variable))


Rts = as_draws_df(fit$draws()) %>%
  select(matches(rt_pattern))
```

```
## Warning: Dropping 'draws_df' class as required metadata was removed.
```

```
Rts %>% slice(sample(1:4000,10)) %>% pivot_longer(everything()) %>% ggplot(aes(x = value))+geom_histogra
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

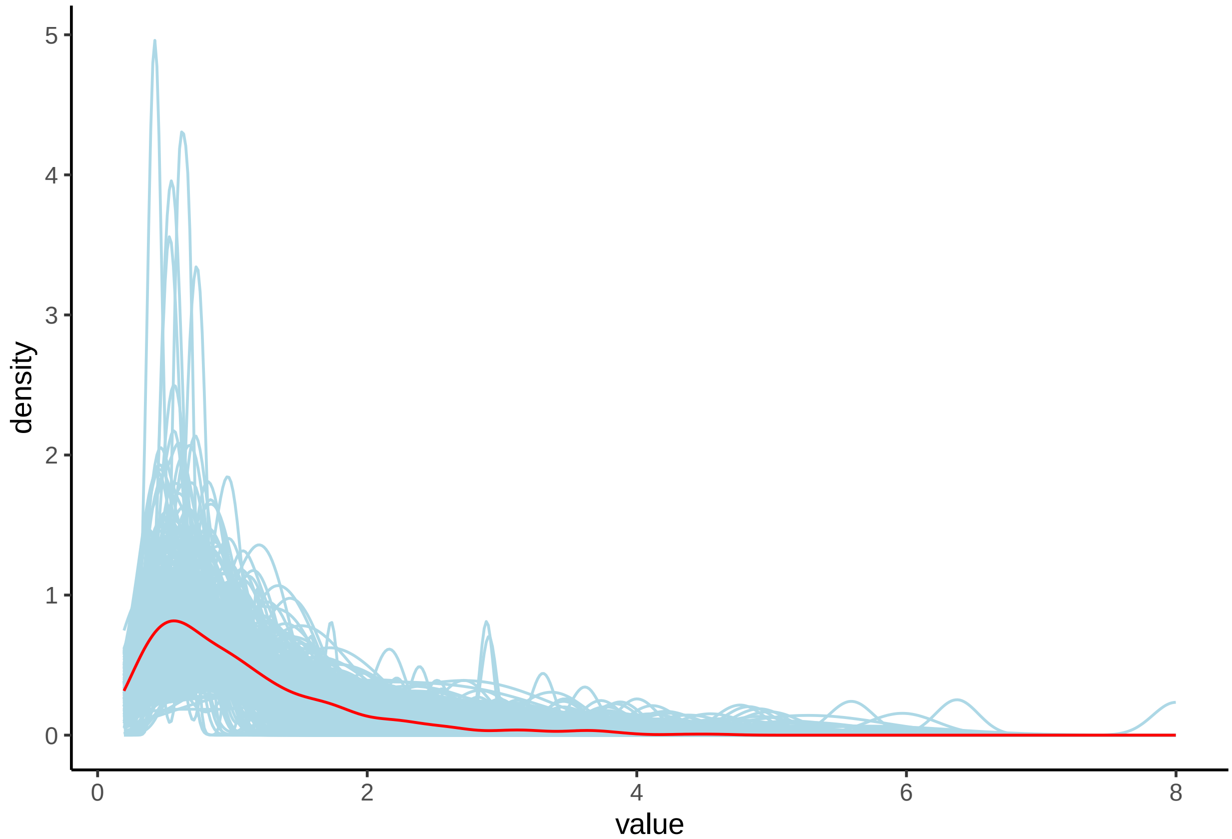



```r
choice = fit$summary() %>%
  filter(grepl(choice_pattern, variable))

choice = as_draws_df(fit$draws()) %>%
  select(matches(choice_pattern))
```

```
## Warning: Dropping 'draws_df' class as required metadata was removed.
```

```r
draws = 10
#pp_check
as_draws_df(fit$draws()) %>%
  select(matches(rt_pattern)) %>% slice(sample(1:4000,draws)) %>%
  pivot_longer(everything()) %>% mutate(estimated = TRUE, slice = rep(1:trials,draws)) %>%
  ggplot()+
  geom_density(aes(x = value, group = slice), color = "lightblue")+
  geom_density(data = data %>% mutate(estimated = FALSE), aes(x = q), color = "red")+
  theme_classic()
```

```
## Warning: Dropping 'draws_df' class as required metadata was removed.
```



Lets do some parameter recovery on this model!

fit model

```
fit_model = function(parameters){
  id = parameters$id

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

  data_stan = list(Nu = nrow(data %>% filter(resp == "upper")),
                   Nl = nrow(data %>% filter(resp == "lower")),
                   RTu = data %>% filter(resp == "upper") %>% .$q,
                   RTl = data %>% filter(resp == "lower") %>% .$q,
                   minRT = min(data$q),
                   run_estimation = 1,
                   RTbound = 0)


  mod = cmdstanr::cmdstan_model(here::here("stan_scripts","HDDM.stan"))
```

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



  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))
}
```

```
trials = seq(50,100,by = 10)
alpha = seq(1,4,by = 1)
delta = seq(-3,3,by = 1)
beta = seq(0.2,0.8,by = 0.1)
tau = seq(0.1,0.4,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)
```

```
#cores = availableCores()-1
#
# plan(multisession, workers = 10)
#
# possfit_model = possibly(.f = fit_model, 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"]
#
# results = NULL
```

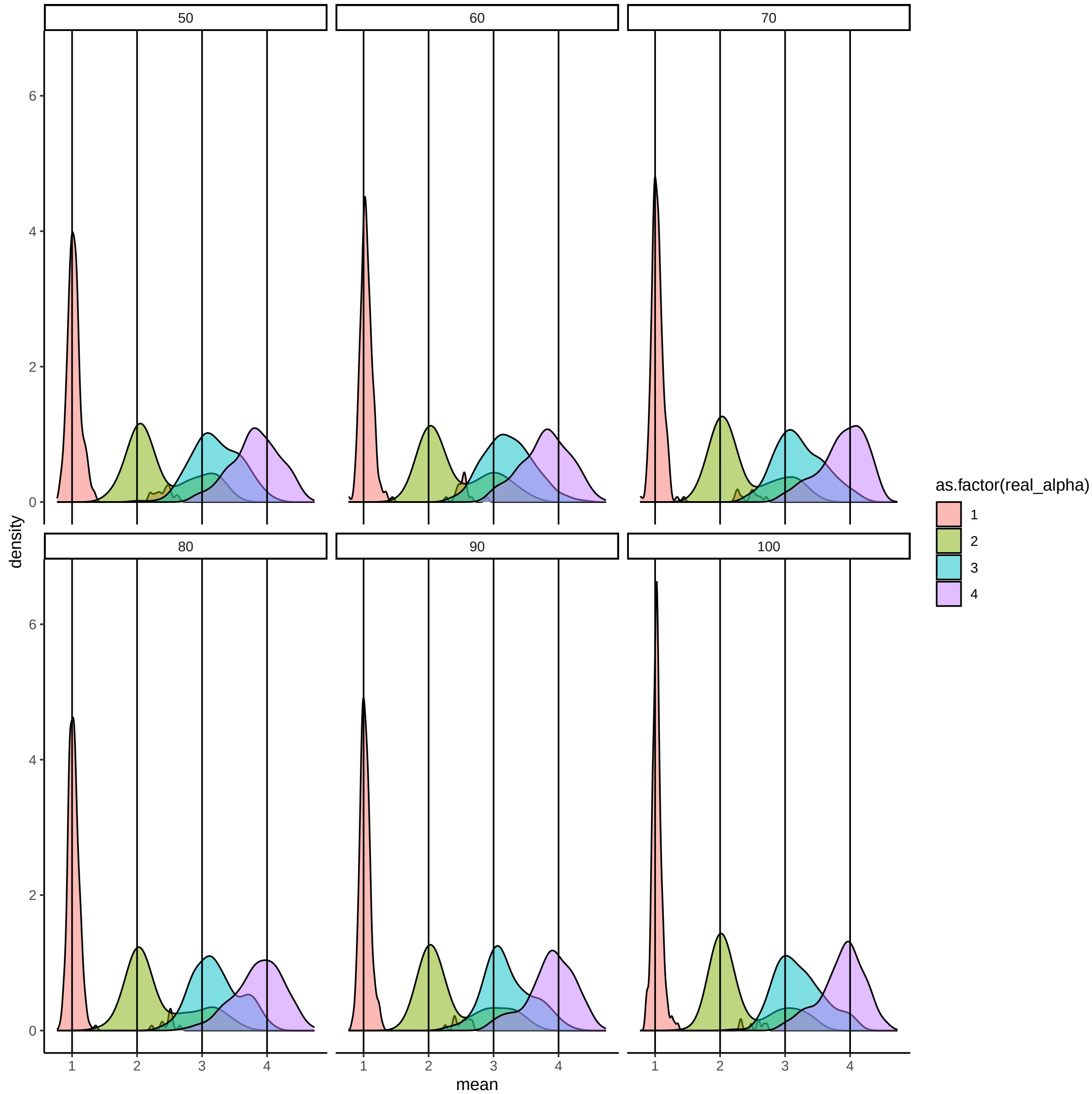
lets look at the divergences

```
load(here::here("workspace_data","ddm_parameterrecovery.RData"))

divergence = map_dfr(results2, 2)

divergence %>% median_qi(num_divergent)
```

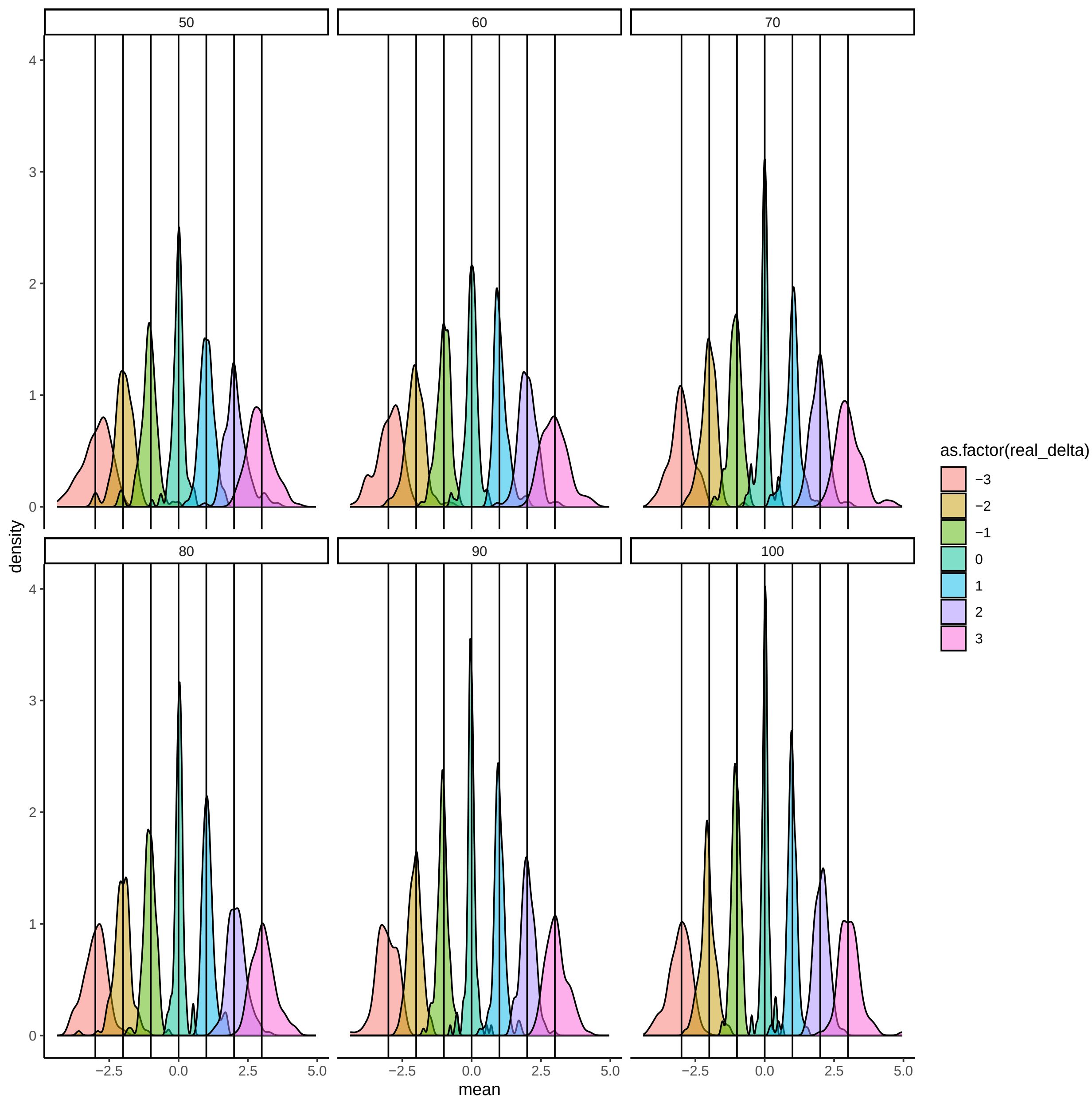
```
## # A tibble: 1 x 6
##   num_divergent .lower .upper .width .point .interval
##           <dbl>  <dbl>  <dbl>  <dbl> <chr>  <chr>
## 1             0      0      0   0.95 median qi
```

there are none which is good now at the parameter values

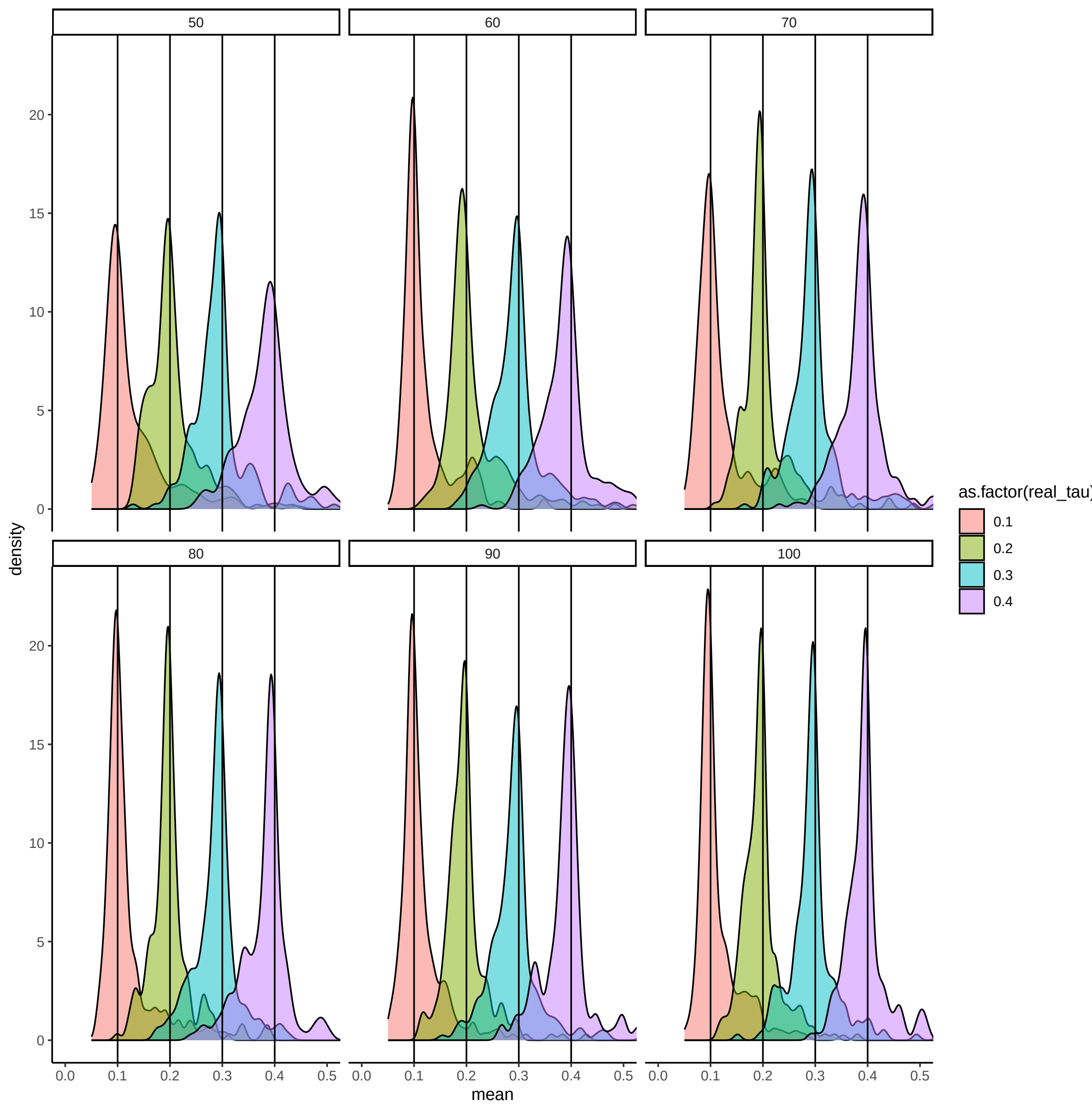
```
params = map_dfr(results2, 1)

params %>% filter(variable == "alpha") %>%
  ggplot(aes(x = mean, fill = as.factor(real_alpha)))+
  geom_density(alpha = 0.5)+
  theme_classic()+
  geom_vline(aes(xintercept = real_alpha))+
  facet_wrap(~trials)
```



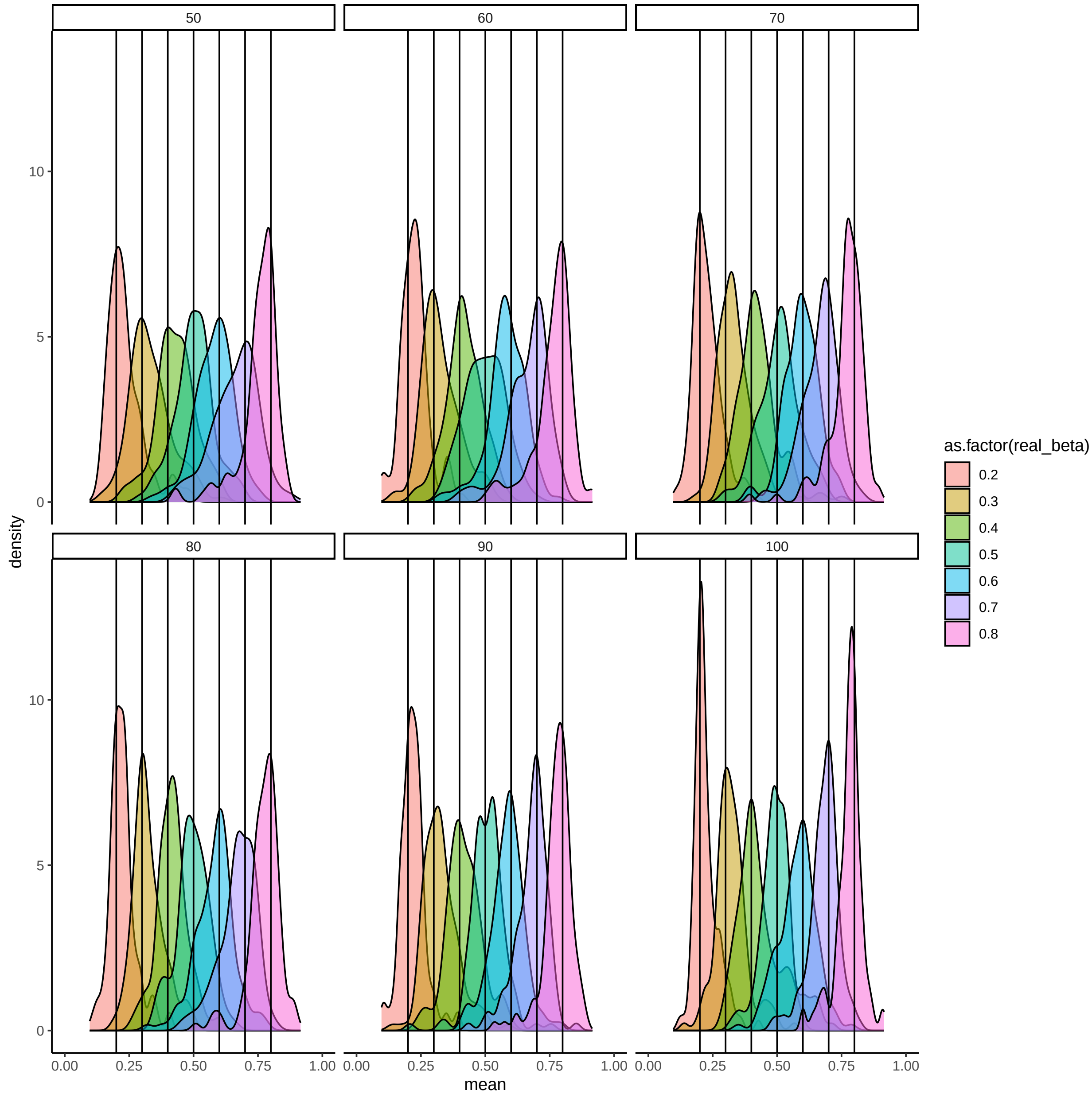

```r
params %>% filter(variable == "delta") %>%
  ggplot(aes(x = mean, fill = as.factor(real_delta)))+
  geom_density(alpha = 0.5)+
  theme_classic()+
  geom_vline(aes(xintercept = real_delta))+
  facet_wrap(~trials)
```



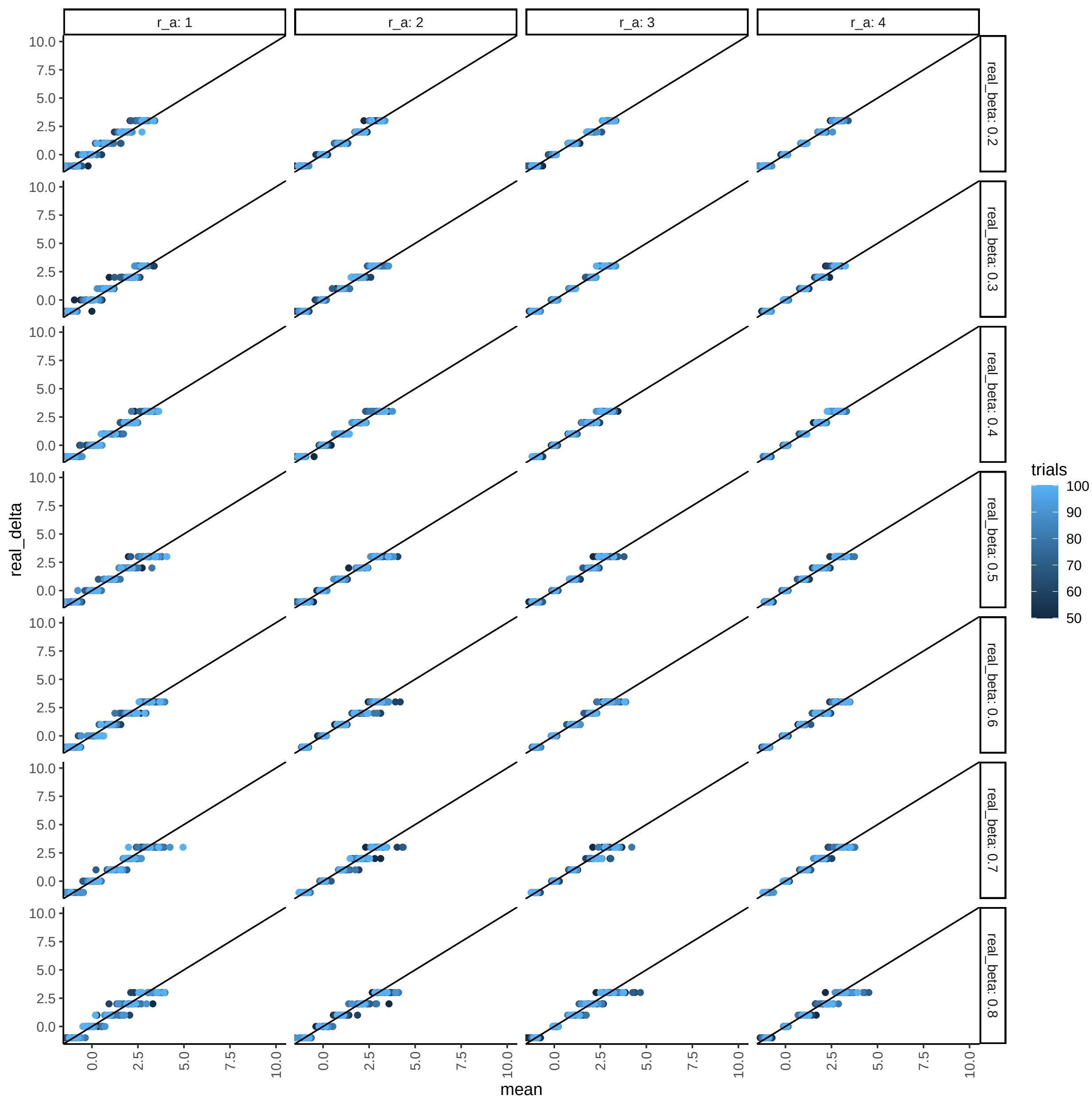

```r
params %>% filter(variable == "tau") %>%
  ggplot(aes(x = mean, fill = as.factor(real_tau)))+
  geom_density(alpha = 0.5)+
  theme_classic()+
  geom_vline(aes(xintercept = real_tau))+
  facet_wrap(~trials)+
  coord_cartesian(xlim = c(0,.5))
```



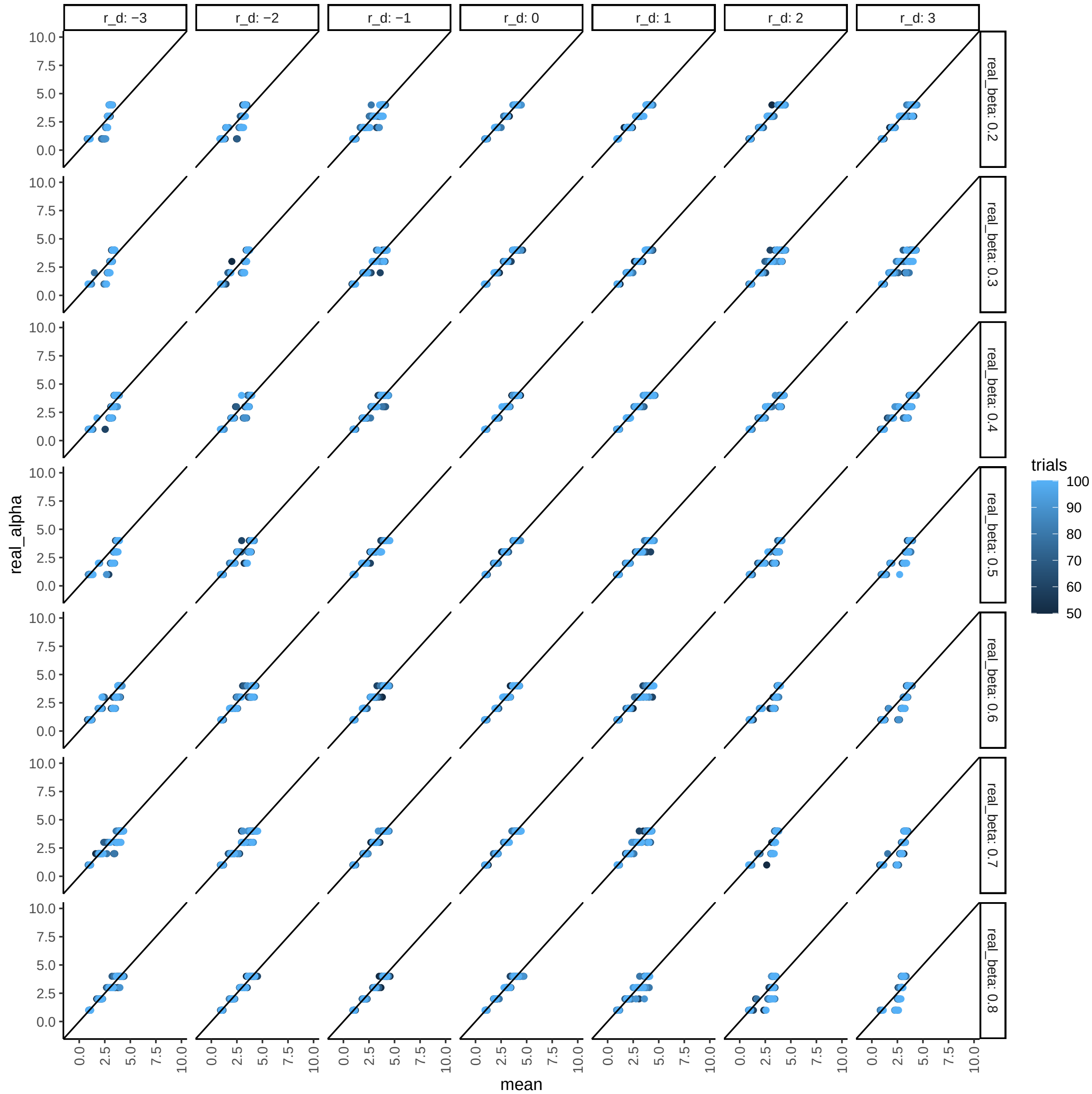

```r
params %>% filter(variable == "beta") %>%
  ggplot(aes(x = mean, fill = as.factor(real_beta)))+
  geom_density(alpha = 0.5)+
  theme_classic()+
  geom_vline(aes(xintercept = real_beta))+
  facet_wrap(~trials)+
  coord_cartesian(xlim = c(0,1))
```




```
params %>%
  mutate_if(is.numeric, round, digits = 2) %>%
  filter(variable == "delta") %>%
  dplyr::rename(r_a = real_alpha) %>%
        ggplot(aes(x = mean, y = real_delta, col = trials))+
        facet_grid(real_beta~r_a, labeller = label_both, scales = "free")+
        theme_classic()+
  geom_point(aes())+geom_abline(slope = 1, intercept = 0)+
  coord_cartesian(ylim = c(-1, 10), xlim = c(-1,10))+ theme(axis.text.x = element_text(angle = 90, vjus
```




```
params %>%
  mutate_if(is.numeric, round, digits = 2) %>%
  filter(variable == "alpha") %>%
  dplyr::rename(r_d = real_delta) %>%
        ggplot(aes(x = mean, y = real_alpha, col = trials))+
        facet_grid(real_beta~r_d, labeller = label_both, scales = "free")+
        theme_classic()+
  geom_point(aes())+geom_abline(slope = 1, intercept = 0)+
  coord_cartesian(ylim = c(-1, 10), xlim = c(-1,10))+ theme(axis.text.x = element_text(angle = 90, vjus
```



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.

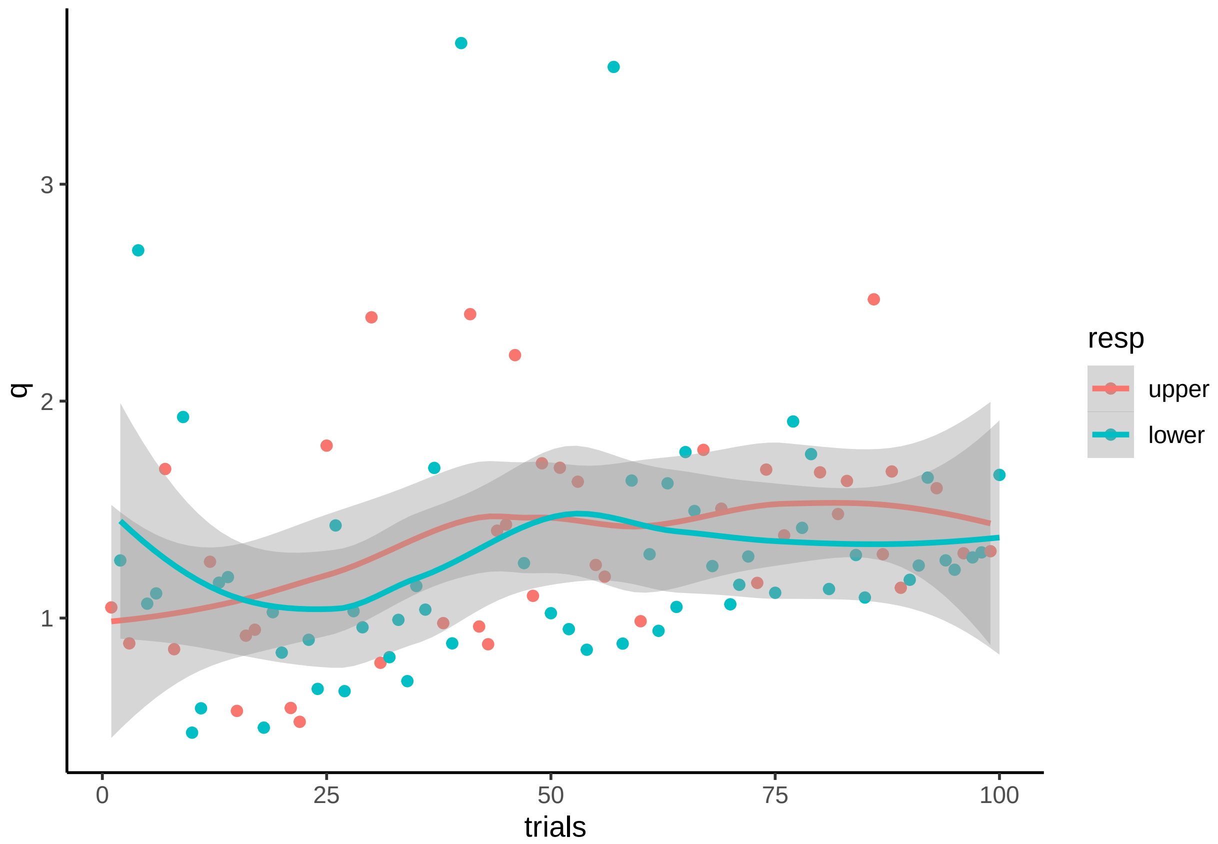
We would also expect that these effects would be somewhat mitigated by breaks (i.e a sudden shift in these parameters).

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

```r
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()
```

```
## 'geom_smooth()' using method = 'loess' and formula = 'y ~ x'
```



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

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

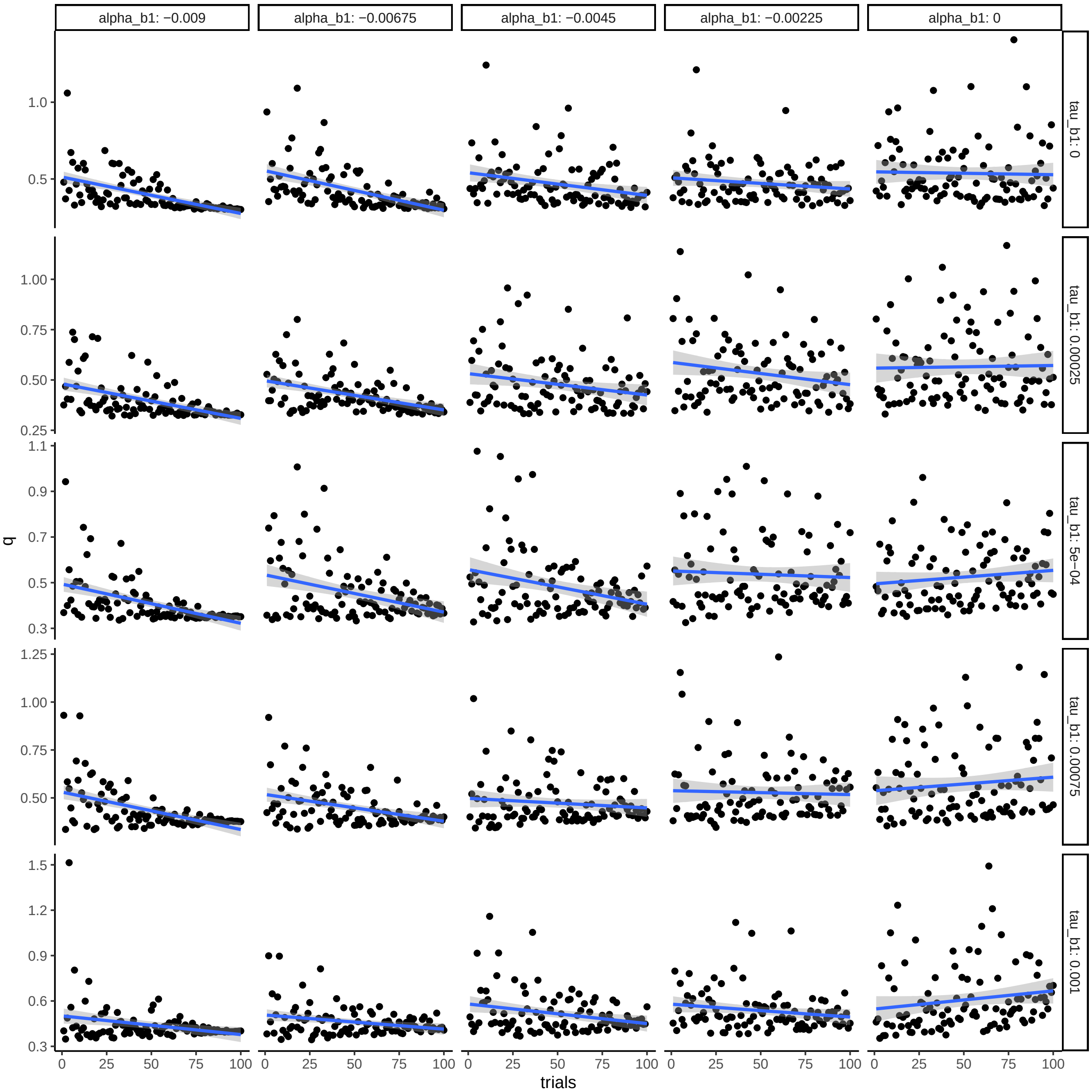
```r
parameters = expand.grid(alpha_0 = alpha_0,
                         alpha_b1 = alpha_b1,
                         delta_0 = delta_0,
                         delta_b1 = delta_b1,
                         beta = beta,
                         tau_0 = tau_0,
                         tau_b1 = tau_b1,
                         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)
```

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

```r
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")
```

```
## `geom_smooth()` using formula = 'y ~ x'
```



alpha_b1: −0.009
alpha_b1: −0.00675
alpha_b1: −0.0045
alpha_b1: −0.00225
alpha_b1: 0
tau_b1: 0
tau_b1: 0.00025
tau_b1: 5e−04
tau_b1: 0.00075
tau_b1: 0.001
q
trials