ERLDMM

2023-10-22

R Markdown

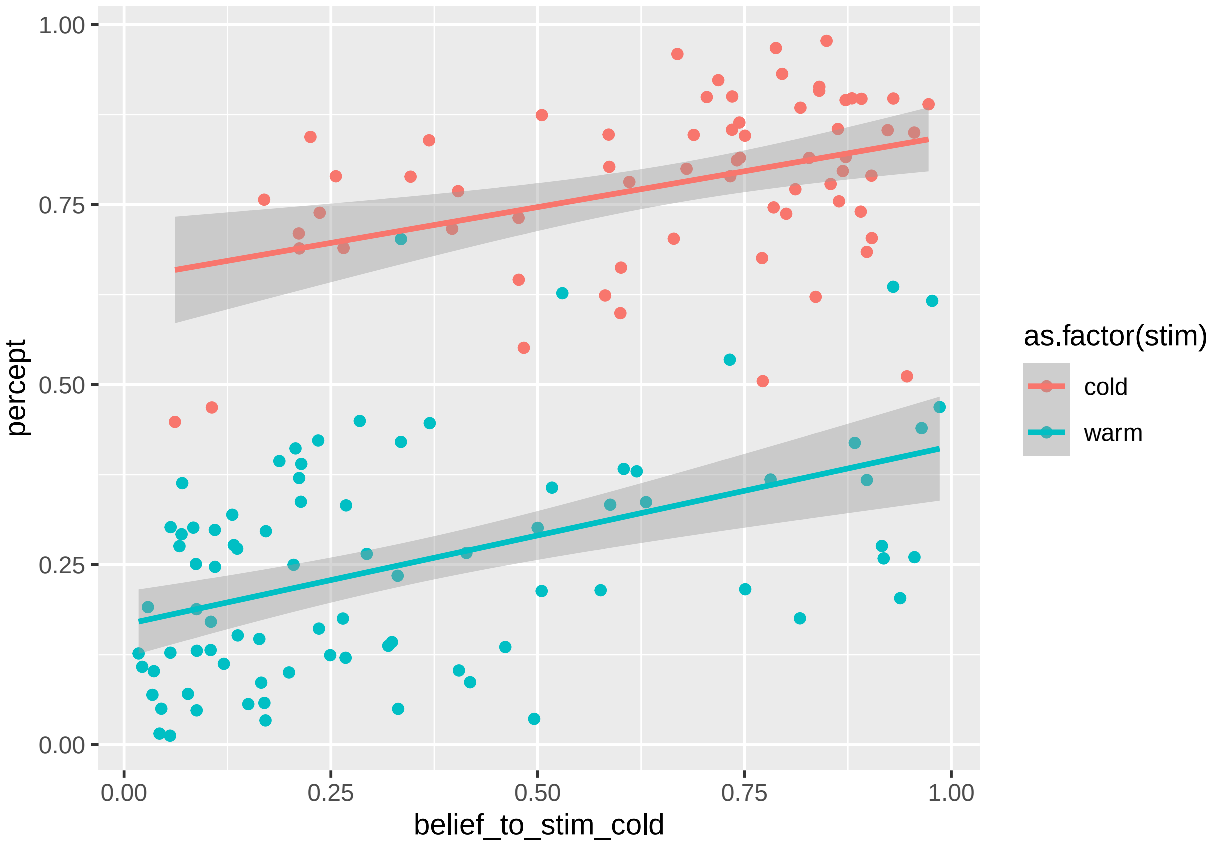
```
agent_expect = function(parameters){
  trials_per_reversal = parameters$trials_per_reversal

  get_u = function(trials_per_reversal){
    return(u = c(rbinom(trials_per_reversal,1,0.8),
                 rbinom(trials_per_reversal,1,0.2),
                 rbinom(trials_per_reversal,1,0.8),
                 rbinom(trials_per_reversal,1,0.2),
                 rbinom(trials_per_reversal,1,0.8)))
  }

  u = get_u(trials_per_reversal)

  N = length(u)


  stim = rbinom(N,1,0.5)
  cue = ifelse(stim == u, 1,0)

  stim2 = ifelse(stim == 1, "cold","warm")
  cue2 = ifelse(cue == 1, "high-tone","low-tone")

  stim = ifelse(stim == 1, 0.8, 0.2)

  alpha = parameters$alpha
  delta = parameters$delta
  beta = parameters$beta
  tau = parameters$tau
  lr = parameters$lr
  e0 = parameters$e0
  zeta = parameters$zeta
  nu = parameters$nu
  prec_per = parameters$prec_per

  expectation = array(NA, N+1)
  uncertainty = array(NA, N)
  real_resp = array(NA, N)
  mu_per = array(NA, N)
  percept = array(NA, N)
  belief_to_stim_cold = array(NA, N)
```

```
expectation[1] = e0

resp = data.frame()
for(i in 1:N){

  belief_to_stim_cold[i] = ifelse(cue[i] == 1, expectation[i], 1-expectation[i])

  mu_per[i] = (1-nu)*stim[i]+nu*belief_to_stim_cold[i]

  percept[i] = extraDistr::rprop(1, prec_per, mu_per[i])

  uncertainty[i] = (expectation[i]-(1-expectation[i]))*delta

  resp1 = rwiener(n = 1,
        alpha = alpha,
        delta = uncertainty[i],
        beta = beta,
        tau = tau)

  expectation[i+1] = expectation[i]+lr*(u[i]-expectation[i])

  real_resp[i] = rbinom(1,1,(expectation[i]^zeta)/((expectation[i]^zeta)+(1-expectation[i])^zeta))

  resp = rbind(resp,resp1)
}

resp$u = u
resp$expectation = expectation[1:N]
resp$uncertainty = uncertainty[1:N]
resp$real_resp = real_resp

resp$trial = 1:N

resp %>% ggplot(aes(x = trial, y = expectation))+geom_line()+geom_point(aes(x = trial, y = u))

resp = resp %>% mutate(resp2 = ifelse(resp == "upper",1,0))

resp = resp %>% mutate(correct = ifelse(real_resp == u, 1, 0))

resp$percept = ifelse(percept < 0.001, 0.001, ifelse(percept > 0.999, 0.999, percept))

resp$belief_to_stim_cold = belief_to_stim_cold

resp$stim = stim2

resp$cue = cue2

resp$stim2 = stim

resp$cue2 = cue


return(resp)
```

```
}
```

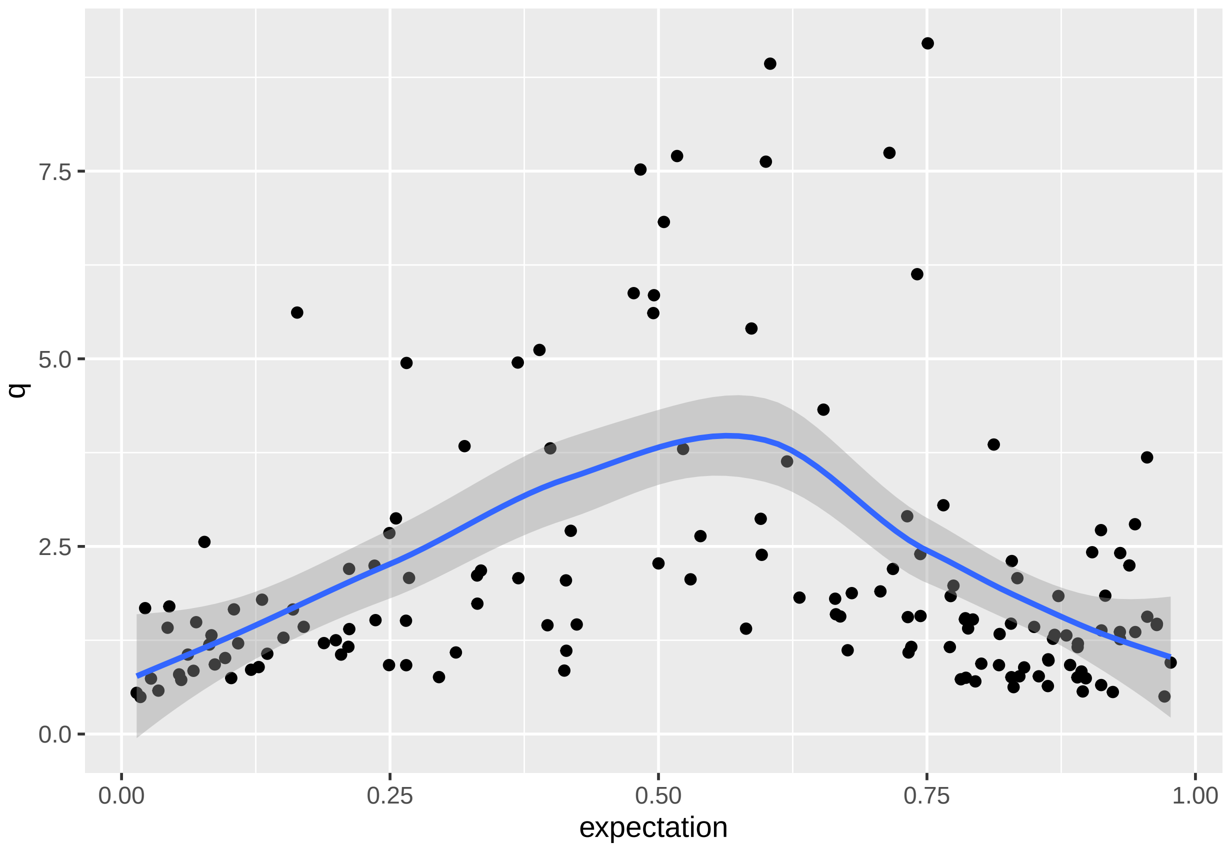
```
resp = agent_expect(parameters = data.frame(trials_per_reversal = 30,
                              alpha = 4,
                              lr = 0.2,
                              delta = 2,
                              beta = 0.5,
                              tau = 0.1,
                              e0 = 0.5,
                              zeta = 3,
                              nu = 0.2,
                              prec_per = 10,
                              id = 2))

resp %>% ggplot(aes(x = belief_to_stim_cold, y = percept, col = as.factor(stim)))+geom_point()+geom_smo
```

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



```
resp %>% ggplot(aes(x = expectation, y = q))+geom_point()+geom_smooth()
```

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



```
data_stan = list(Nu = nrow(resp %>% filter(resp == "upper")),
                 Nl = nrow(resp %>% filter(resp == "lower")),
                 RTu = resp %>% filter(resp == "upper") %>% .$q,
                 RTl = resp %>% filter(resp == "lower") %>% .$q,
                 minRT = min(resp$q),
                 run_estimation = 1,
                 trials = nrow(resp),
                 stim = resp$stim2,
                 cue = resp$cue2,
                 percept = resp$percept,
                 u = resp$u,
                 indexupper = resp %>% filter(resp == "upper") %>% .$trial,
                 indexlower = resp %>% filter(resp == "lower") %>% .$trial,
                 resp = c(resp$resp2,0))


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



 # fit1 <- mod$sample(
 #     data = data_stan,
 #     chains = 4,
 #     parallel_chains = 4,
 #     adapt_delta = 0.8,
 #     max_treedepth = 10,
 #     refresh = 10
 #     )
#
#
#
# fit1$save_object(here::here("models","ERLDDM_model.RDS"))

fit1 <- readRDS(here::here("models","ERLDDM_model.RDS"))

flextable::flextable(data.frame(fit1$summary()) %>% mutate_if(is.numeric, round, digits = 2) %>% head(8
```

```
## 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___	-18,439.66	-18,439.40	1.86	1.78	-18,443.10	-18,437.20	1	1,911.49	2,768.84
alpha	3.46	3.46	0.02	0.02	3.42	3.50	1	2,733.87	3,024.84
beta	0.52	0.52	0.00	0.00	0.52	0.52	1	6,062.77	3,302.40
delta	1.32	1.32	0.01	0.01	1.29	1.34	1	3,847.72	3,035.11
tau_raw	-0.87	-0.87	0.08	0.07	-1.00	-0.74	1	3,344.54	3,088.31
lr	0.23	0.23	0.00	0.00	0.23	0.24	1	4,077.73	3,030.62
nu	0.14	0.14	0.03	0.04	0.09	0.20	1	4,600.47	2,425.57
prec_per	10.99	10.92	1.22	1.22	9.06	13.04	1	5,390.46	3,227.71

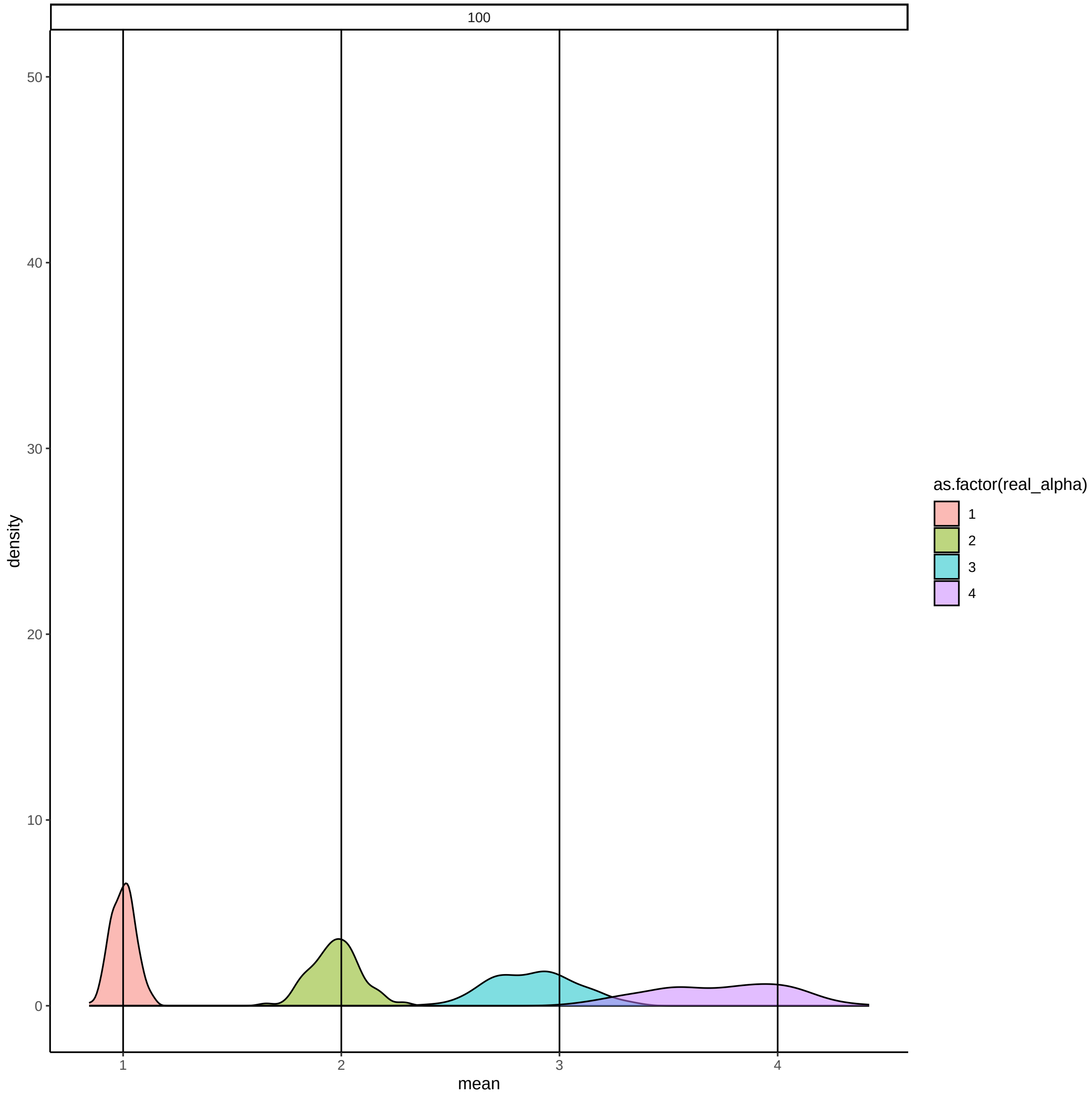
```
parameter_recovery_expect = function(parameters){

  resp = agent_expect(parameters)

  data_stan = list(Nu = nrow(resp %>% filter(resp == "upper")),
                   Nl = nrow(resp %>% filter(resp == "lower")),
                   RTu = resp %>% filter(resp == "upper") %>% .$q,
                   RTl = resp %>% filter(resp == "lower") %>% .$q,
                   minRT = min(resp$q),
                   run_estimation = 1,
                   trials = nrow(resp),
                   stim = resp$stim,
                   percept = resp$percept,
                   u = resp$u,
                   indexupper = resp %>% filter(resp == "upper") %>% .$trial,
                   indexlower = resp %>% filter(resp == "lower") %>% .$trial,
                   resp = c(resp$resp2,0))


  mod = cmdstanr::cmdstan_model(here::here("stan_scripts","ERLDDM.stan"))
  fit1 <- mod$sample(
      data = data_stan,
      chains = 4,
      parallel_chains = 4,
      adapt_delta = 0.9,
      max_treedepth = 12,
      refresh = 100
      )


  posteriors = as_draws_df(fit1$summary()) %>% dplyr::filter(variable %in% names(parameters))

  diag = data.frame(fit1$diagnostic_summary(), id = parameters$id)

  data = posteriors %>% mutate(real_alpha = parameters$alpha,
                               real_delta = parameters$delta,
                               real_beta = parameters$beta,
                               real_tau = parameters$tau,
                               real_lr = parameters$lr,
                               trials = parameters$n_reversals*parameters$trials_per_reversal,
                               real_prec_per = parameters$prec_per,
                               real_nu = parameters$nu,
                               id = parameters$id)
  return(list(data, diag))

}
```

```
n_reversals = seq(5,length.out = 1)
#n_reversals = seq(5,length.out = 1)

trials_per_reversal = seq(20, length.out = 1)
#trials_per_reversal = seq(20, length.out = 1)


alpha = seq(1,4, length.out = 4)

lr = seq(0.1,0.4, length.out = 4)

zeta = seq(3, length.out = 1)

delta  = seq(-2,2,length.out = 4)

beta = seq(0.5,length.out = 1)

tau = seq(0.1, length.out = 1)

e0 = seq(0.5, length.out = 1)

prec_per = seq(1,10, length.out = 3)

nu = seq(0.1,0.4, length.out = 4)
```

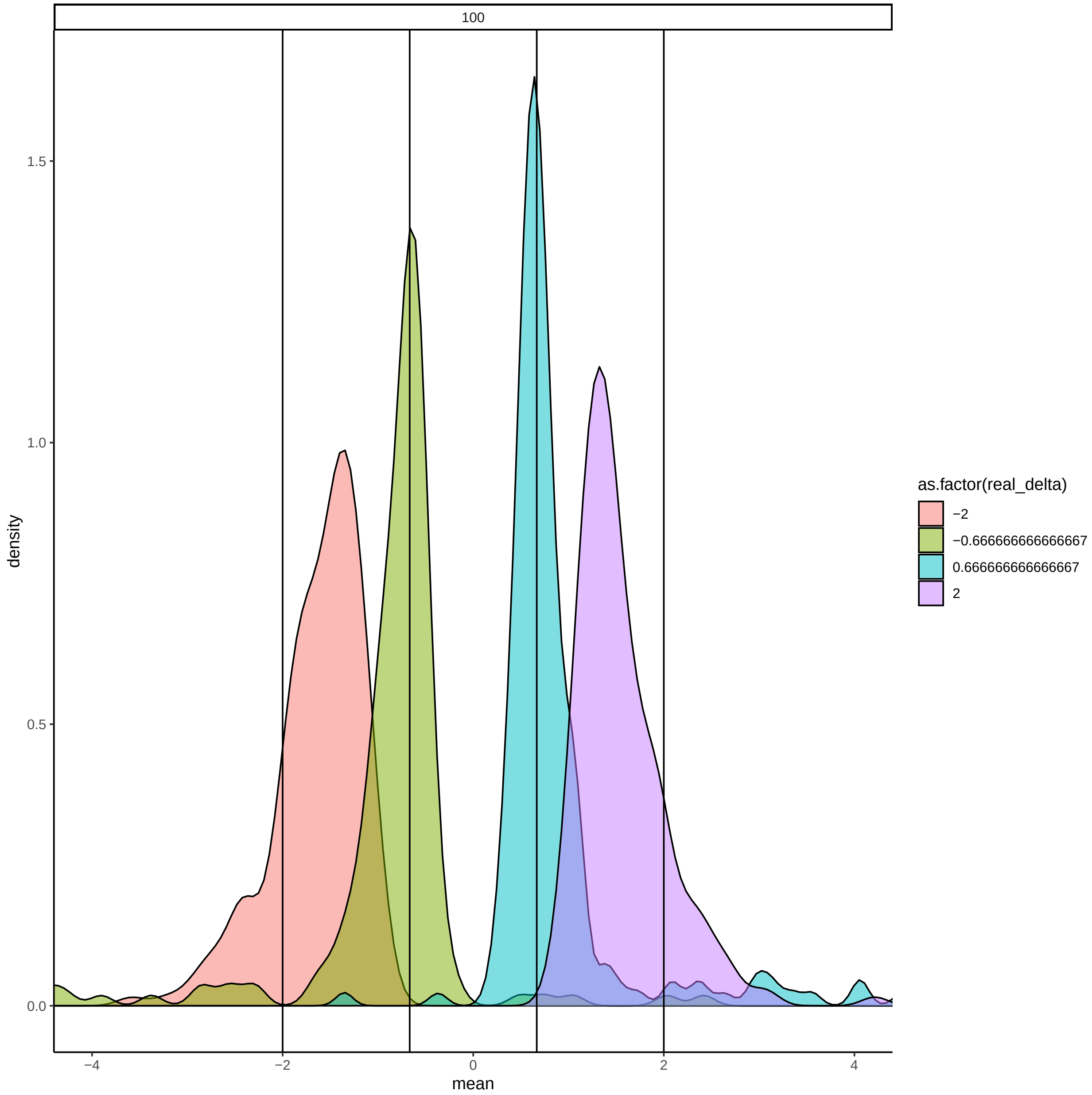
```
replicate = 1

parameters = expand.grid(n_reversals = n_reversals,
                         lr= lr,
                         zeta = zeta,
                         alpha = alpha,
                         delta = delta,
                         beta = beta,
                         tau = tau,
                         prec_per = prec_per,
                         nu = nu,
                         e0 = e0,
                         replicate = replicate,
                         trials_per_reversal = trials_per_reversal) %>%
  mutate(id = 1:nrow(.))

data_list <- split(parameters, parameters$id)
```

```
# qq = parameter_recovery_expect(data_list[[50]])
#
# cores = availableCores()-1
#
# plan(multisession, workers = 4)
#
# possfit_model = possibly(.f = parameter_recovery_expect, 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"]
```

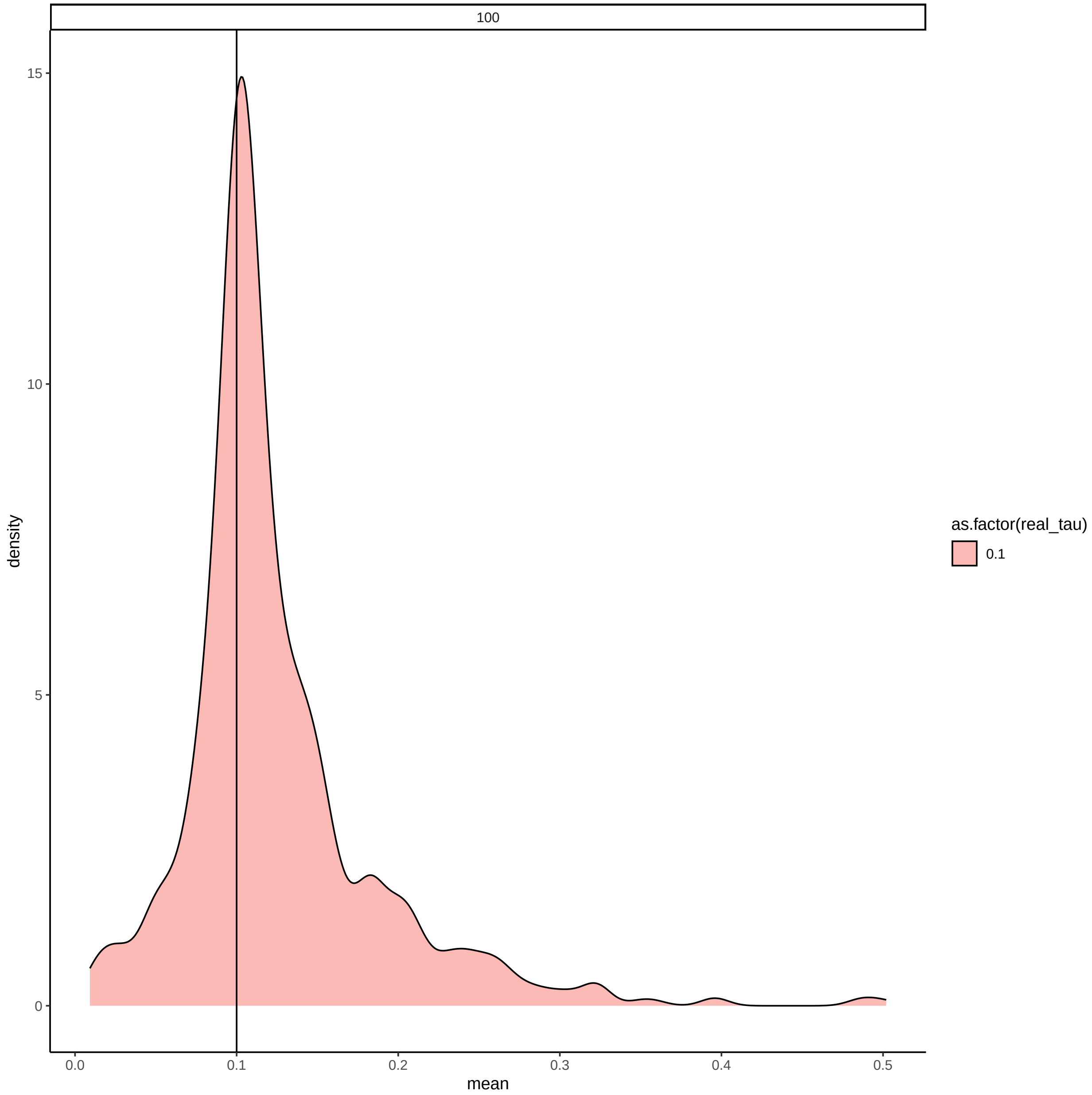
```
load(here::here("workspace_data","ERLDMM.RData"))


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)+coord_cartesian(ylim = c(0,50))
```



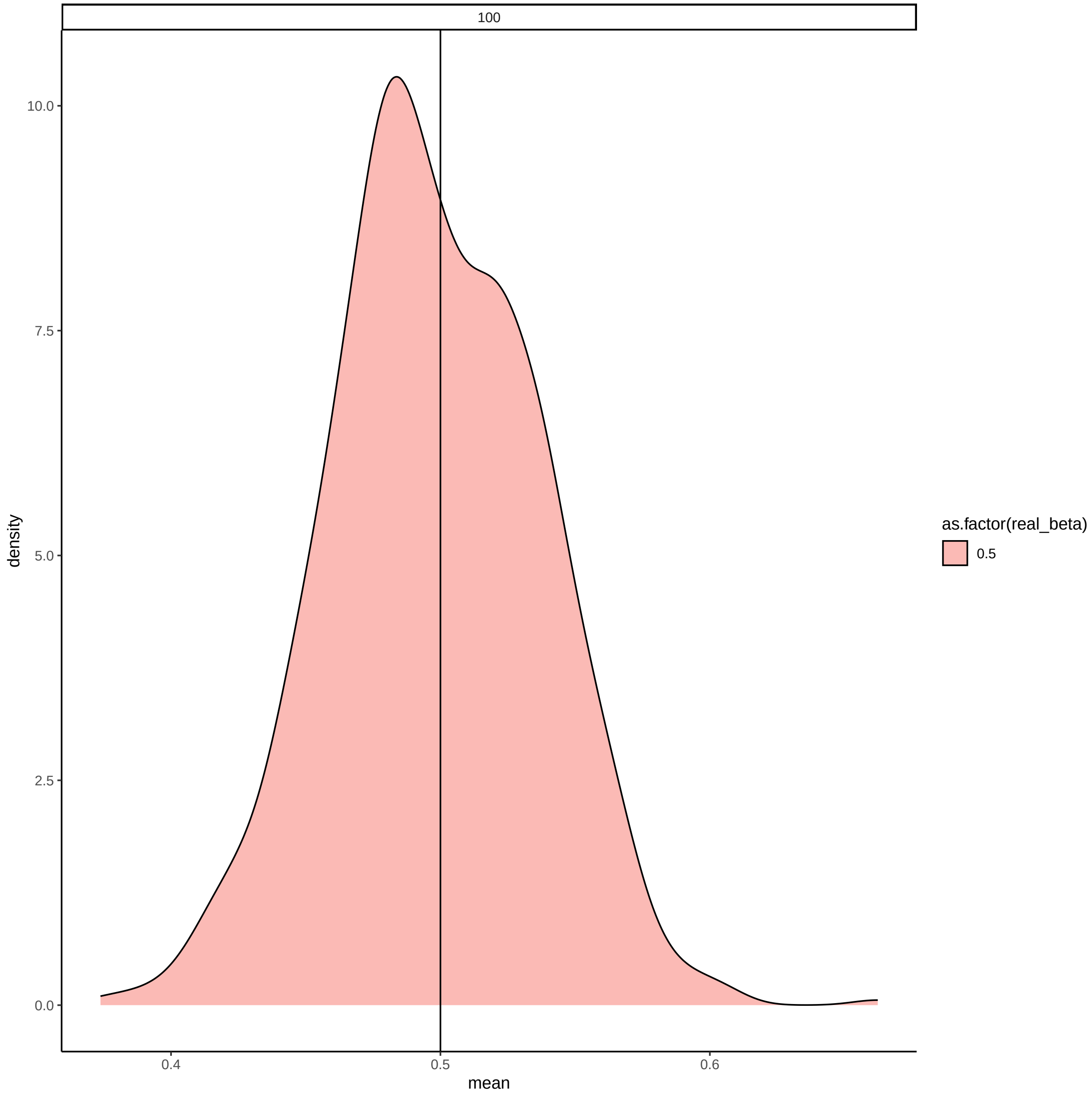
```
params %>% filter(variable == "delta") %>%
  ggplot(aes(x = mean, fill = as.factor(real_delta)))+
  geom_density(alpha = 0.5)+
  theme_classic()+
  geom_vline(aes(xintercept = real_delta))+
  facet_wrap(~trials)+
  coord_cartesian(xlim = c(-4,4))
```



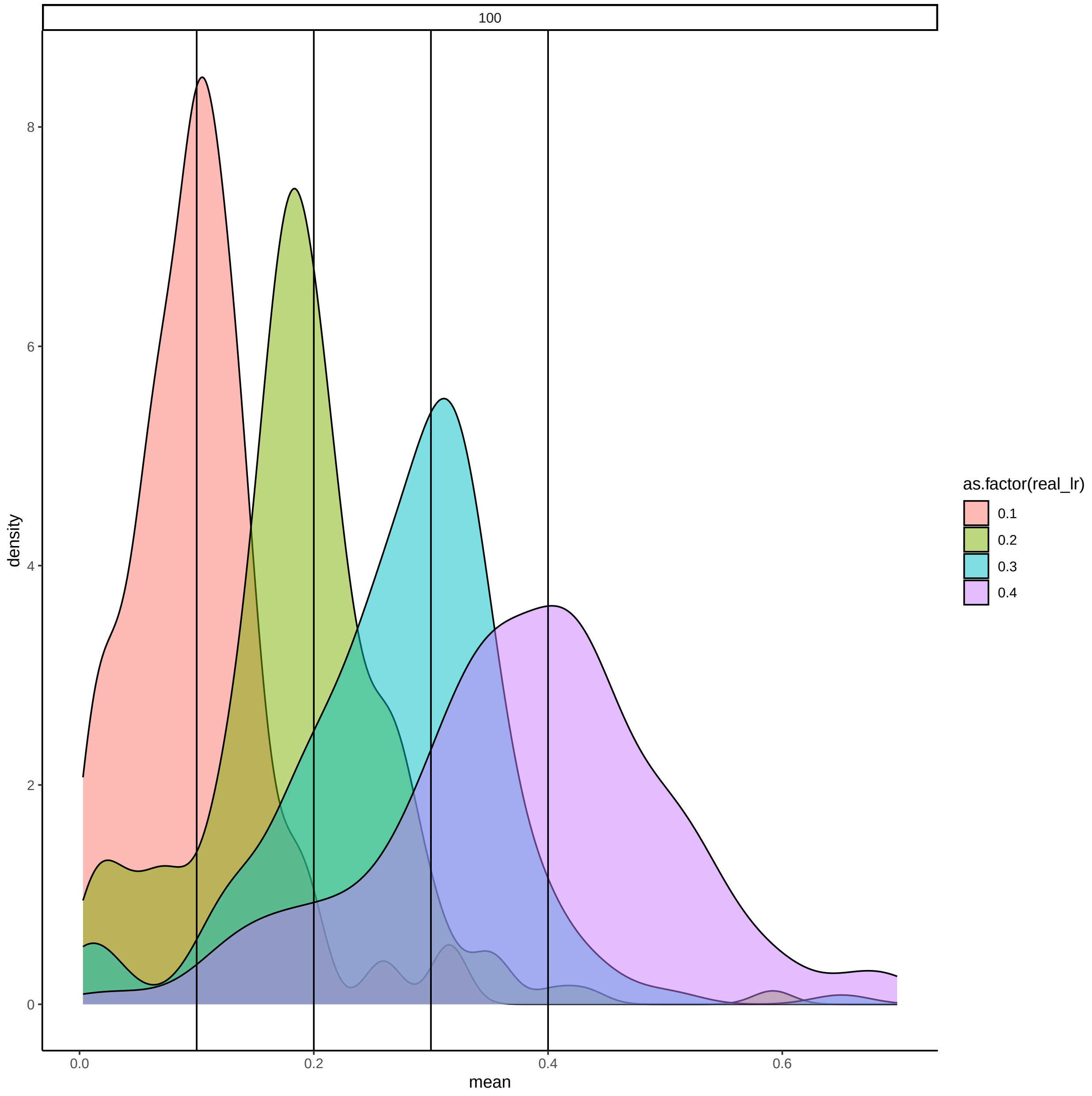
```
params %>% filter(variable == "tau") %>%
  ggplot(aes(x = mean, fill = as.factor(real_tau)))+
  geom_density(alpha = 0.5)+
  theme_classic()+
  geom_vline(aes(xintercept = real_tau))+
  facet_wrap(~trials)
```



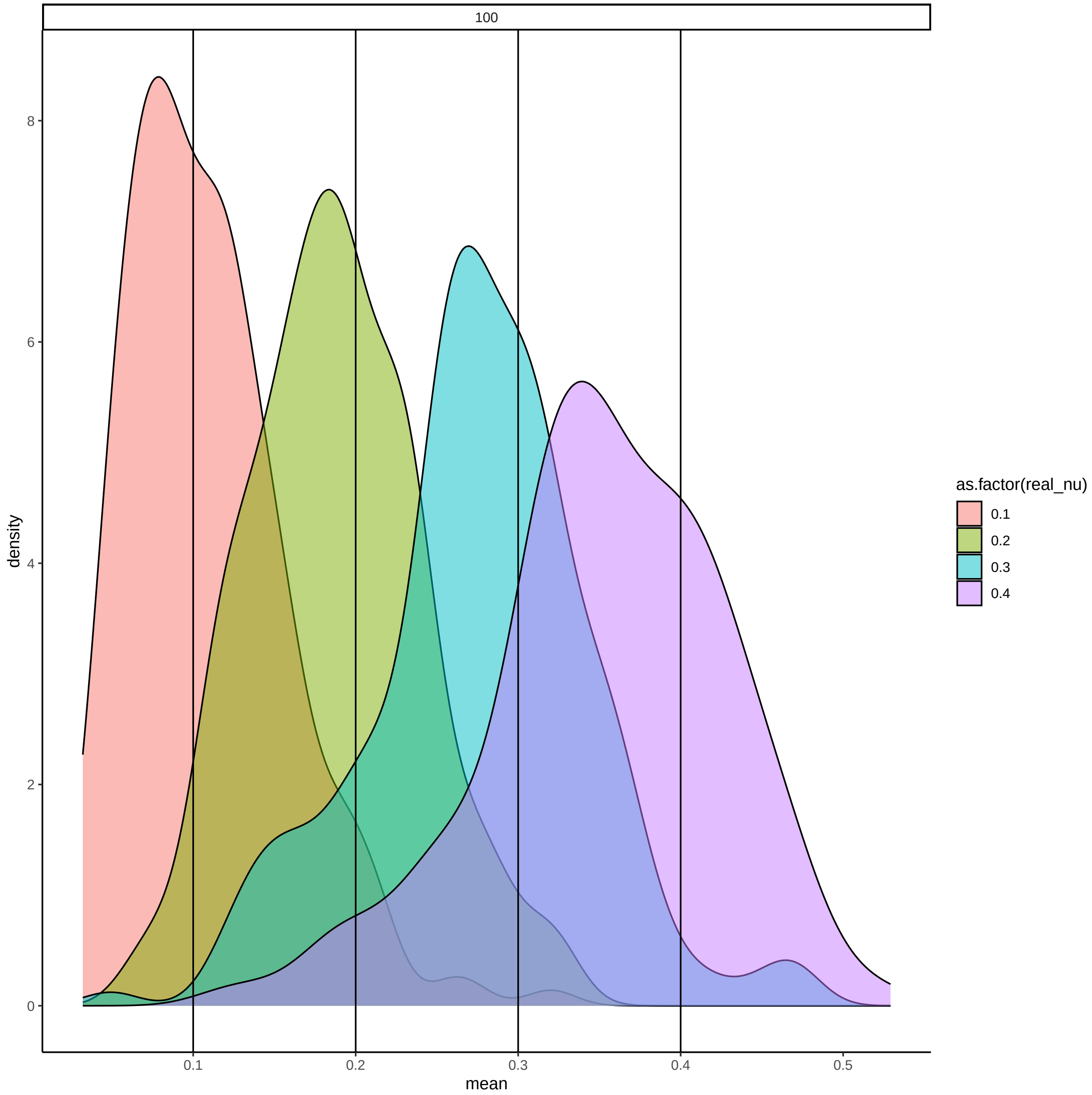
```
params %>% filter(variable == "beta") %>%
  ggplot(aes(x = mean, fill = as.factor(real_beta)))+
  geom_density(alpha = 0.5)+
  theme_classic()+
  geom_vline(aes(xintercept = real_beta))+
  facet_wrap(~trials)
```



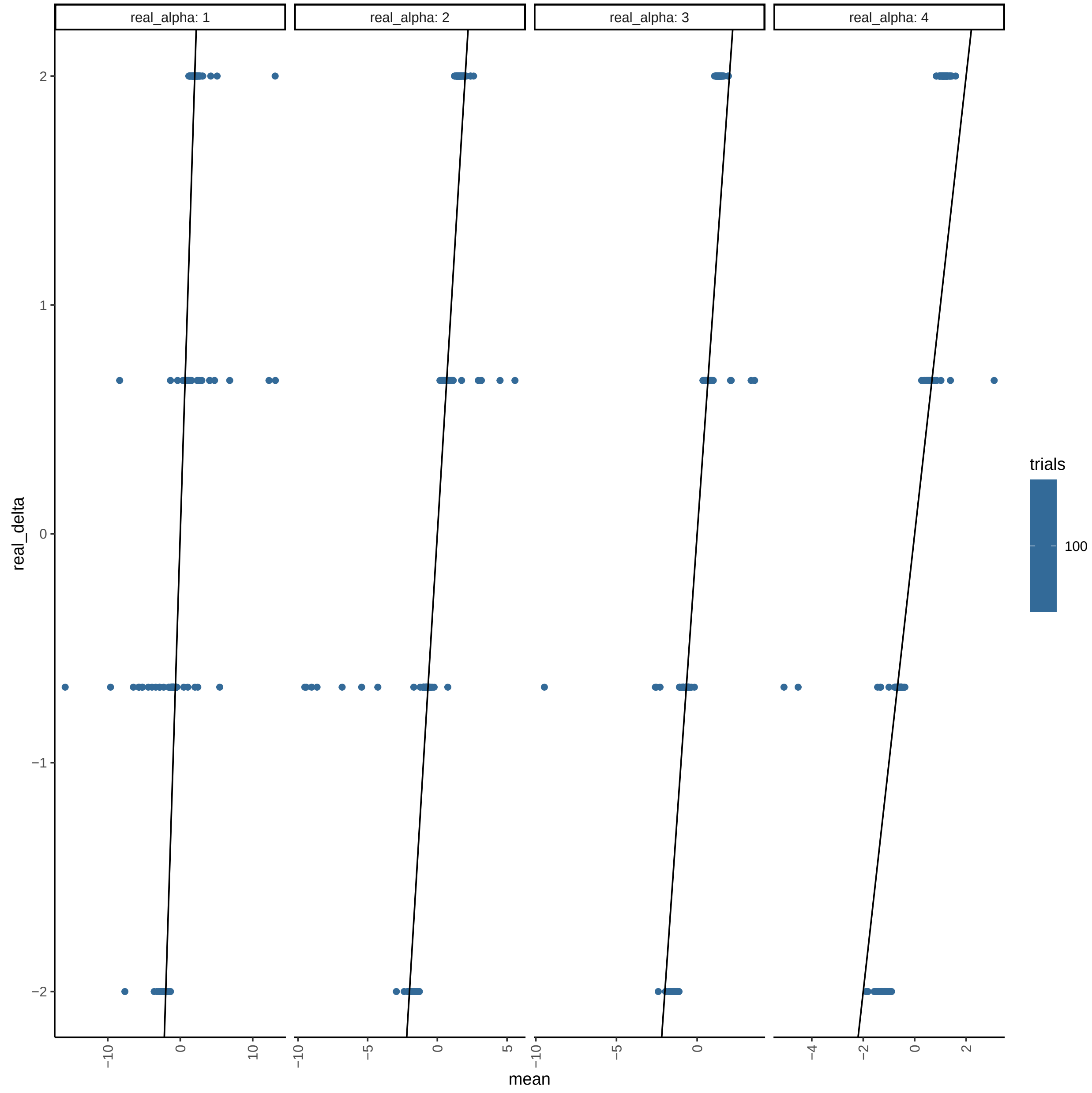
```
params %>% filter(variable == "lr") %>%
  ggplot(aes(x = mean, fill = as.factor(real_lr)))+
  geom_density(alpha = 0.5)+
  theme_classic()+
  geom_vline(aes(xintercept = real_lr))+
  facet_wrap(~trials)
```



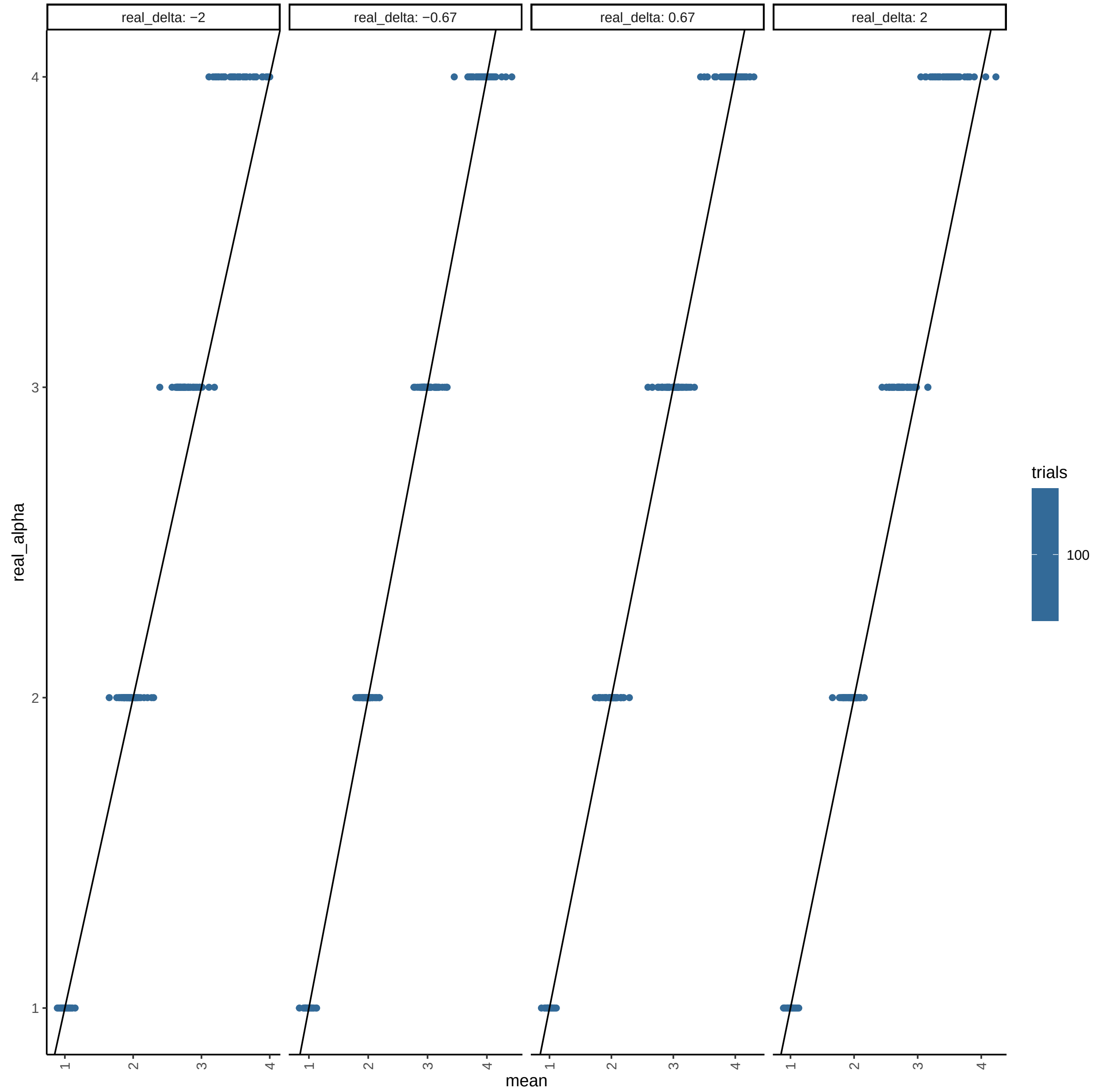
```
params %>% filter(variable == "nu") %>%
  ggplot(aes(x = mean, fill = as.factor(real_nu)))+
  geom_density(alpha = 0.5)+
  theme_classic()+
  geom_vline(aes(xintercept = real_nu))+
  facet_wrap(~trials)
```



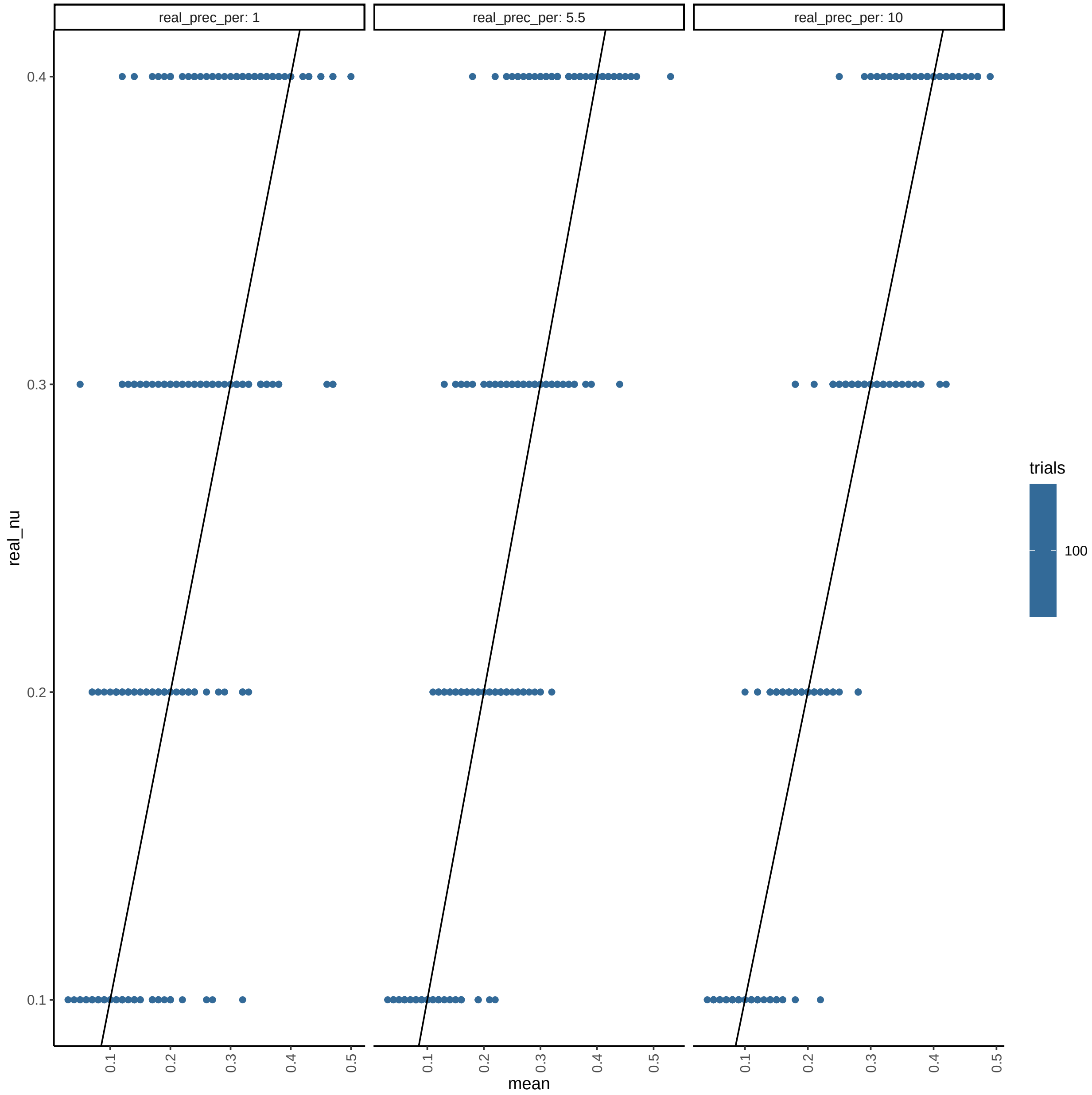
```
params %>%
  mutate_if(is.numeric, round, digits = 2) %>%
  filter(variable == "delta") %>%
        ggplot(aes(x = mean, y = real_delta, col = trials))+
        facet_grid(~real_alpha, labeller = label_both, scales = "free")+
        theme_classic()+
  geom_point(aes())+geom_abline(slope = 1, intercept = 0)+
  theme(axis.text.x = element_text(angle = 90, vjust = 0.5, hjust=1))
```



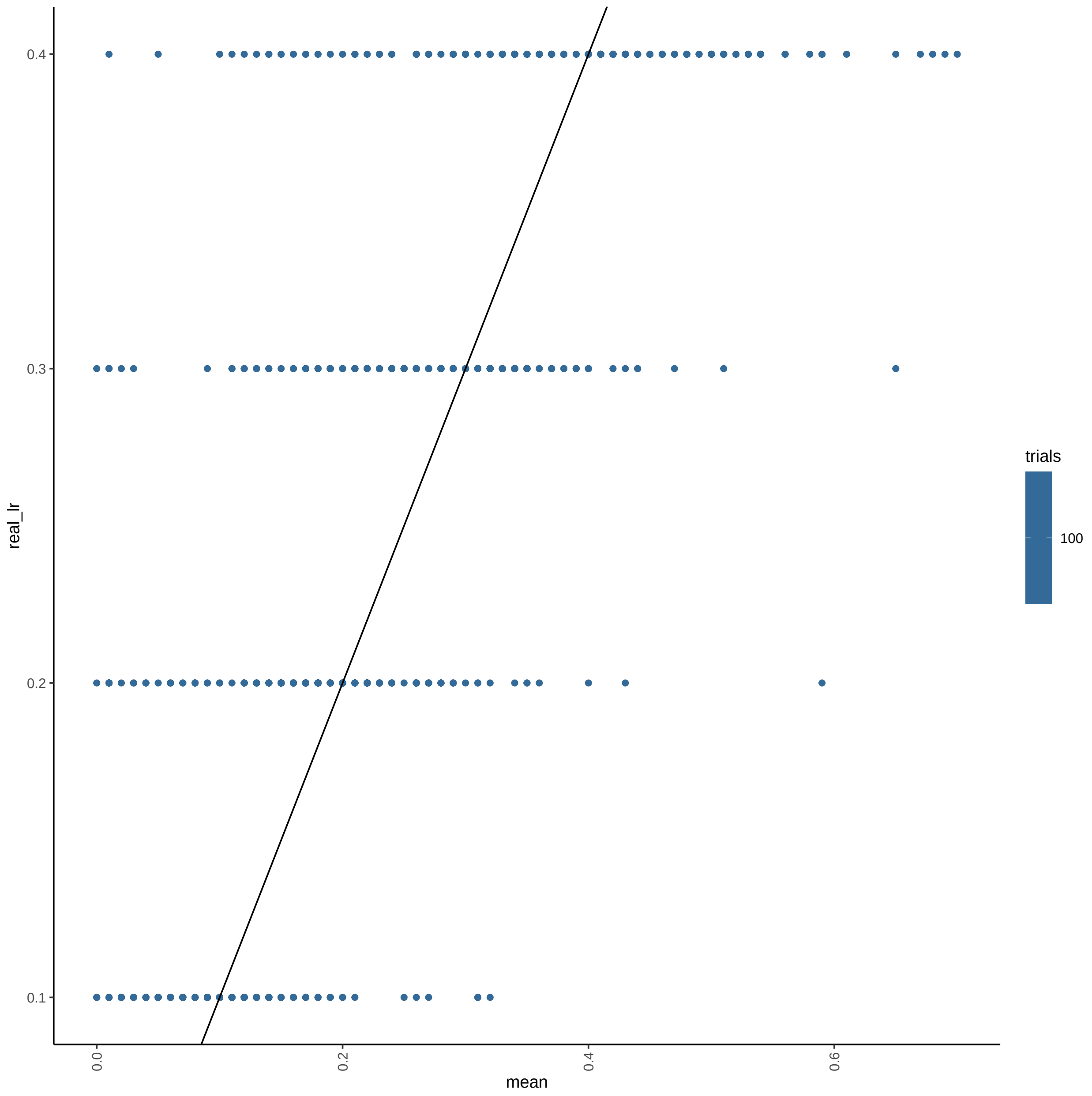
```
params %>%
  mutate_if(is.numeric, round, digits = 2) %>%
  filter(variable == "alpha") %>%
        ggplot(aes(x = mean, y = real_alpha, col = trials))+
        facet_grid(~real_delta, labeller = label_both, scales = "free")+
        theme_classic()+
  geom_point(aes())+geom_abline(slope = 1, intercept = 0)+
  theme(axis.text.x = element_text(angle = 90, vjust = 0.5, hjust=1))
```



```
params %>%
  mutate_if(is.numeric, round, digits = 2) %>%
  filter(variable == "nu") %>%
        ggplot(aes(x = mean, y = real_nu, col = trials))+
        facet_grid(~real_prec_per, labeller = label_both, scales = "free")+
        theme_classic()+
  geom_point(aes())+geom_abline(slope = 1, intercept = 0)+theme(axis.text.x = element_text(angle = 90,
```



```
params %>%
  mutate_if(is.numeric, round, digits = 2) %>%
  filter(variable == "lr") %>%
        ggplot(aes(x = mean, y = real_lr, col = trials))+
        theme_classic()+
  geom_point(aes())+geom_abline(slope = 1, intercept = 0)+theme(axis.text.x = element_text(angle = 90,
```


trials
100
real_lr
0.1
0.2
0.3
0.4
mean
0.0
0.2
0.4
0.6