Plot and examine chains: 3 regions (FP + reg)

JD

March 4, 2021

1 Simple model and plots

This file collects draws and generates plots and info about parameters.

```
burnin <- 200

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(rstan)

## Loading required package:  StanHeaders
## Loading required package:  ggplot2
## Need help getting started?  Try the R Graphics Cookbok:
## https://r-graphics.org
## rstan (Version 2.21.1, 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)

c1 <- read.csv("chain1/chain-0.csv")

dataf <- select(c1, starts_with("mu_rt"))

dataf <- dataf[burnin:length(dataf[, 1]), ]

str(dataf)

## 'data.frame': 2794 obs. of  12 variables:
##  $ mu_rt__0 : num  259 259 249 251 251 ...
##  $ mu_rt__1 : num  320 320 302 314 314 ...
##  $ mu_rt__2 : num  745 744 705 718 716 ...
##  $ mu_rt__3 : num  259 259 249 251 251 ...
##  $ mu_rt__4 : num  339 339 316 338 338 ...
##  $ mu_rt__5 : num  753 752 713 723 721 ...
##  $ mu_rt__6 : num  265 264 252 255 255 ...
##  $ mu_rt__7 : num  322 322 304 315 315 ...
##  $ mu_rt__8 : num  768 775 735 746 744 ...
##  $ mu_rt__9 : num  265 264 252 255 255 ...
##  $ mu_rt__10: num  341 341 318 339 339 ...
##  $ mu_rt__11: num  766 770 730 740 739 ...
```

```
dataf2 <- select(c1, starts_with("mu_reg"))

dataf2 <- dataf2[burnin:length(dataf2[, 1]), ]

str(dataf2)
```

```
## 'data.frame': 2794 obs. of  12 variables:
##  $ mu_reg__0 : num  0.0318 0.0318 0.0318 0.0341 0.0341 ...
##  $ mu_reg__1 : num  0.0502 0.0502 0.0502 0.0516 0.0516 ...
##  $ mu_reg__2 : num  0.481 0.481 0.481 0.485 0.485 ...
##  $ mu_reg__3 : num  0.0187 0.0187 0.0187 0.0201 0.0201 ...
##  $ mu_reg__4 : num  0.0504 0.0504 0.0504 0.0518 0.0518 ...
##  $ mu_reg__5 : num  0.467 0.468 0.468 0.476 0.476 ...
##  $ mu_reg__6 : num  0.255 0.156 0.156 0.16 0.16 ...
##  $ mu_reg__7 : num  0.242 0.242 0.242 0.247 0.247 ...
##  $ mu_reg__8 : num  0.474 0.469 0.469 0.476 0.476 ...
##  $ mu_reg__9 : num  0.306 0.206 0.206 0.211 0.211 ...
##  $ mu_reg__10: num  0.273 0.273 0.273 0.278 0.278 ...
##  $ mu_reg__11: num  0.503 0.507 0.507 0.516 0.516 ...
```

```
c2 <- read.csv("chain2/chain-0.csv")

dataf.c2 <- select(c2, starts_with("mu_rt"))

dataf.c2 <- dataf.c2[burnin:length(dataf.c2[, 1]), ]

dataf <- rbind(dataf, dataf.c2)

str(dataf)
```

```
## 'data.frame': 5588 obs. of  12 variables:
##  $ mu_rt__0 : num  259 259 249 251 251 ...
##  $ mu_rt__1 : num  320 320 302 314 314 ...
##  $ mu_rt__2 : num  745 744 705 718 716 ...
##  $ mu_rt__3 : num  259 259 249 251 251 ...
##  $ mu_rt__4 : num  339 339 316 338 338 ...
##  $ mu_rt__5 : num  753 752 713 723 721 ...
##  $ mu_rt__6 : num  265 264 252 255 255 ...
##  $ mu_rt__7 : num  322 322 304 315 315 ...
##  $ mu_rt__8 : num  768 775 735 746 744 ...
##  $ mu_rt__9 : num  265 264 252 255 255 ...
##  $ mu_rt__10: num  341 341 318 339 339 ...
##  $ mu_rt__11: num  766 770 730 740 739 ...
```

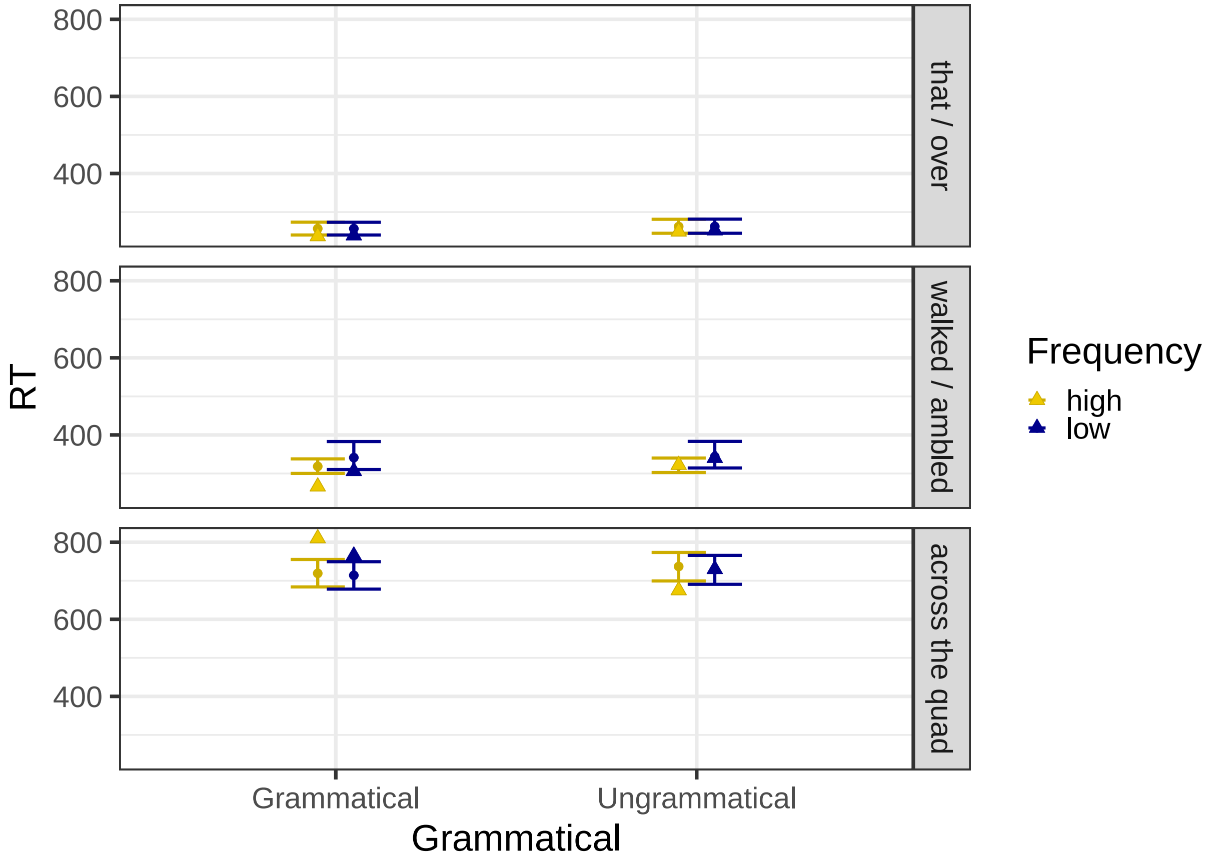
```
dataf2.c2 <- select(c2, starts_with("mu_reg"))

dataf2.c2 <- dataf2.c2[burnin:length(dataf2.c2[, 1]), ]
```

```
dataf2 <- rbind(dataf2, dataf2.c2)

str(dataf2)

## 'data.frame': 5588 obs. of  12 variables:
##  $ mu_reg__0 : num  0.0318 0.0318 0.0318 0.0341 0.0341 ...
##  $ mu_reg__1 : num  0.0502 0.0502 0.0502 0.0516 0.0516 ...
##  $ mu_reg__2 : num  0.481 0.481 0.481 0.485 0.485 ...
##  $ mu_reg__3 : num  0.0187 0.0187 0.0187 0.0201 0.0201 ...
##  $ mu_reg__4 : num  0.0504 0.0504 0.0504 0.0518 0.0518 ...
##  $ mu_reg__5 : num  0.467 0.468 0.468 0.476 0.476 ...
##  $ mu_reg__6 : num  0.255 0.156 0.156 0.16 0.16 ...
##  $ mu_reg__7 : num  0.242 0.242 0.242 0.247 0.247 ...
##  $ mu_reg__8 : num  0.474 0.469 0.469 0.476 0.476 ...
##  $ mu_reg__9 : num  0.306 0.206 0.206 0.211 0.211 ...
##  $ mu_reg__10: num  0.273 0.273 0.273 0.278 0.278 ...
##  $ mu_reg__11: num  0.503 0.507 0.507 0.516 0.516 ...

ndraws <- length(dataf[, 1])

data.all <- data.frame(region = factor(rep(rep(c("that / over", "walked / ambled",
    "across the quad"), each = ndraws), 4), levels = c("that / over", "walked / ambled",
    "across the quad")), grammatical = c(rep("Grammatical", ndraws * 6), rep("Ungrammatical",
    ndraws * 6)), frequency = c(rep("high", ndraws * 3), rep("low", ndraws *
    3), rep("high", ndraws * 3), rep("low", ndraws * 3)), RT = c(dataf[, 1],
    dataf[, 2], dataf[, 3], dataf[, 4], dataf[, 5], dataf[, 6], dataf[, 7],
    dataf[, 8], dataf[, 9], dataf[, 10], dataf[, 11], dataf[, 12]), x = rep(c(237,
    266, 810, 239, 306, 765, 249, 322, 675, 252, 340, 730), each = ndraws))

str(data.all)

## 'data.frame': 67056 obs. of  5 variables:
##  $ region     : Factor w/ 3 levels "that / over",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ grammatical: Factor w/ 2 levels "Grammatical",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ frequency  : Factor w/ 2 levels "high","low": 1 1 1 1 1 1 1 1 1 1 ...
##  $ RT         : num  259 259 249 251 251 ...
##  $ x          : num  237 237 237 237 237 237 237 237 237 237 ...
```

Prepare data for plots.

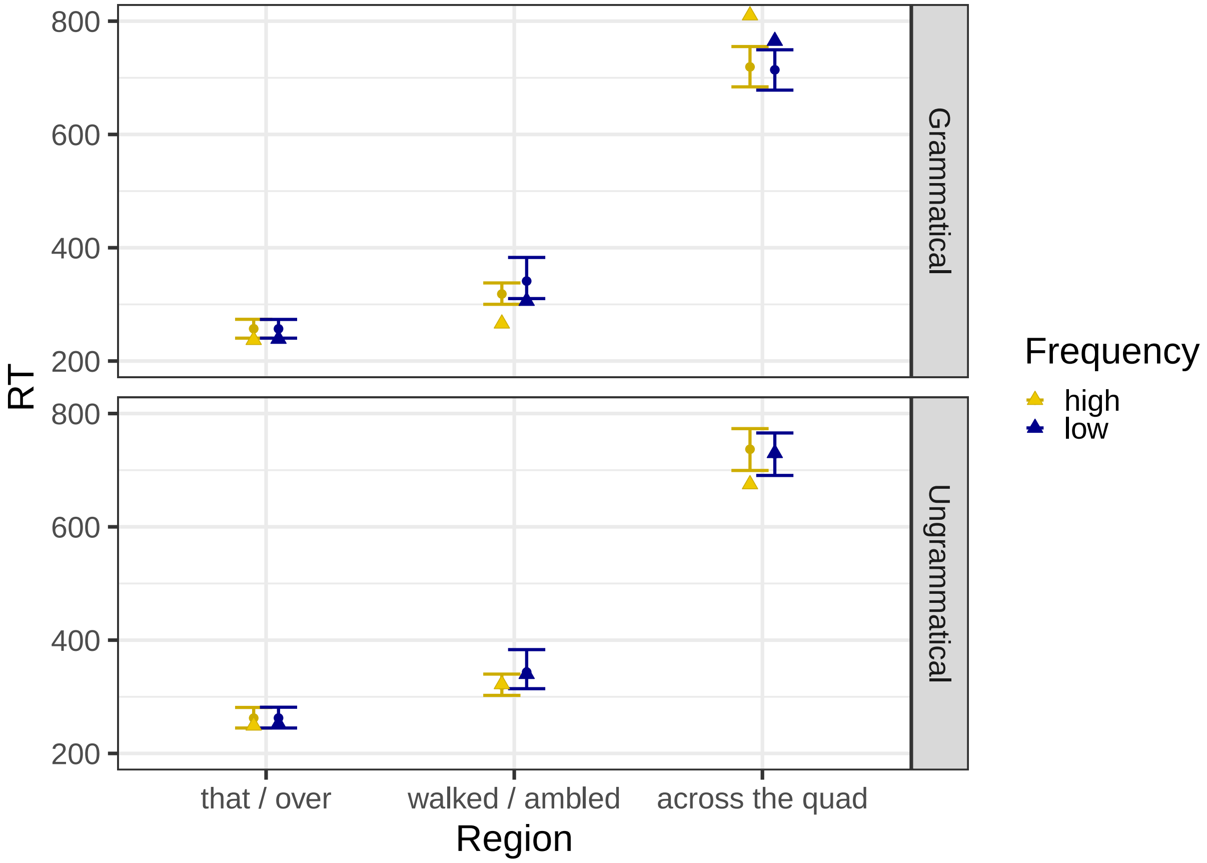
```
library(ggplot2)

library(dplyr)

dodge <- position_dodge(width = 0.2)

data.to.plot <- data.all %>% group_by(region, grammatical, frequency) %>% summarise(Region = first(regi
    Grammatical = first(grammatical), Frequency = first(frequency), CF1 = quantile(RT,
        probs = c(0.05, 0.95))[1], CF2 = quantile(RT, probs = c(0.05, 0.95))[2],
    RT = mean(RT), Observed = first(x))

## `summarise()` has grouped output by 'region', 'grammatical'. You can override using the
`.groups` argument.

data.to.plot
```

```
## # A tibble: 12 x 10
## # Groups:   region, grammatical [6]
##    region grammatical frequency Region Grammatical Frequency   CF1   CF2
##    <fct>  <fct>       <fct>     <fct>  <fct>       <fct>     <dbl> <dbl>
##  1 that ~ Grammatical high      that ~ Grammatical high       240.  274.
##  2 that ~ Grammatical low       that ~ Grammatical low        240.  273.
##  3 that ~ Ungrammati~ high      that ~ Ungrammati~ high       245.  281.
##  4 that ~ Ungrammati~ low       that ~ Ungrammati~ low        245.  282.
##  5 walke~ Grammatical high      walke~ Grammatical high       300.  338.
##  6 walke~ Grammatical low       walke~ Grammatical low        310.  383.
##  7 walke~ Ungrammati~ high      walke~ Ungrammati~ high       302.  340.
##  8 walke~ Ungrammati~ low       walke~ Ungrammati~ low        314.  383.
##  9 acros~ Grammatical high      acros~ Grammatical high       684.  755.
## 10 acros~ Grammatical low       acros~ Grammatical low        678.  749.
## 11 acros~ Ungrammati~ high      acros~ Ungrammati~ high       700.  773.
## 12 acros~ Ungrammati~ low       acros~ Ungrammati~ low        691.  766.
## # ... with 2 more variables: RT <dbl>, Observed <dbl>

g1 <- ggplot(data.to.plot, aes(Grammatical, RT, color = Frequency, fill = Frequency))
g1 <- g1 + geom_point(position = dodge, size = I(3)) + geom_errorbar(aes(ymin = CF1,
    ymax = CF2), position = dodge, width = 0.3, size = I(1.2)) + scale_shape_manual(values = 21:24) +
    scale_color_manual(values = c("gold3", "blue4")) + scale_fill_manual(values = c("gold2",
    "blue4")) + theme_bw(30)  # + theme(legend.justification = c(0.98, 0.9), legend.position = c(0.74,
g1 <- g1 + geom_point(aes(x = Grammatical, y = Observed, fill = Frequency),
    pch = 24, position = dodge, size = 4) + facet_grid(Region ~ .)
```



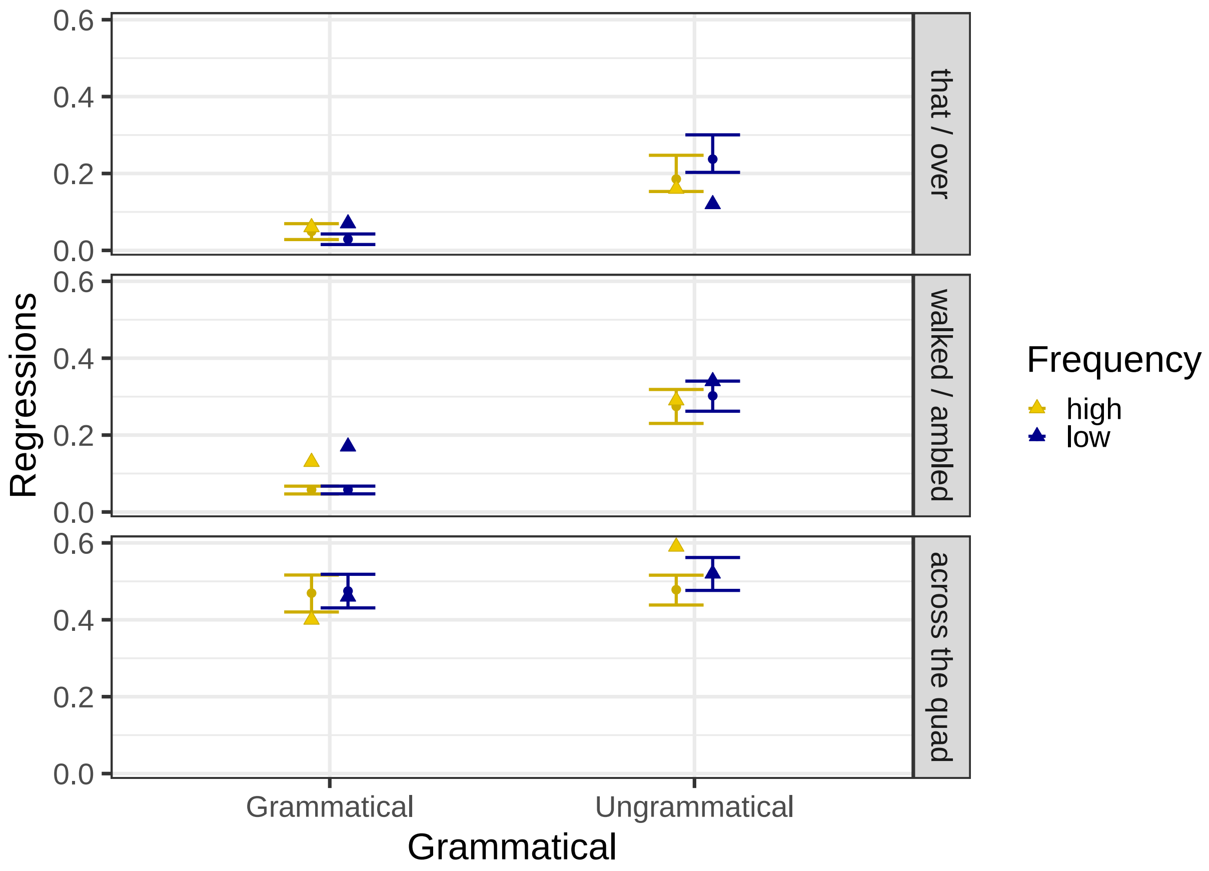
```
ggsave("staub-firstpass.pdf", width = 20, height = 15)
```

```
g1 <- ggplot(data.to.plot, aes(Region, RT, color = Frequency, fill = Frequency))
g1 <- g1 + geom_point(position = dodge, size = I(3)) + geom_errorbar(aes(ymin = CF1,
    ymax = CF2), position = dodge, width = 0.3, size = I(1.2)) + scale_shape_manual(values = 21:24) +
    scale_color_manual(values = c("gold3", "blue4")) + scale_fill_manual(values = c("gold2",
    "blue4")) + theme_bw(30)  # + theme(legend.justification = c(0.98, 0.9), legend.position = c(0.74,
g1 <- g1 + coord_cartesian(ylim = c(200, 800)) + geom_point(aes(x = Region,
    y = Observed, fill = Frequency), pch = 24, position = dodge, size = 4) +
    facet_grid(Grammatical ~ .)
```



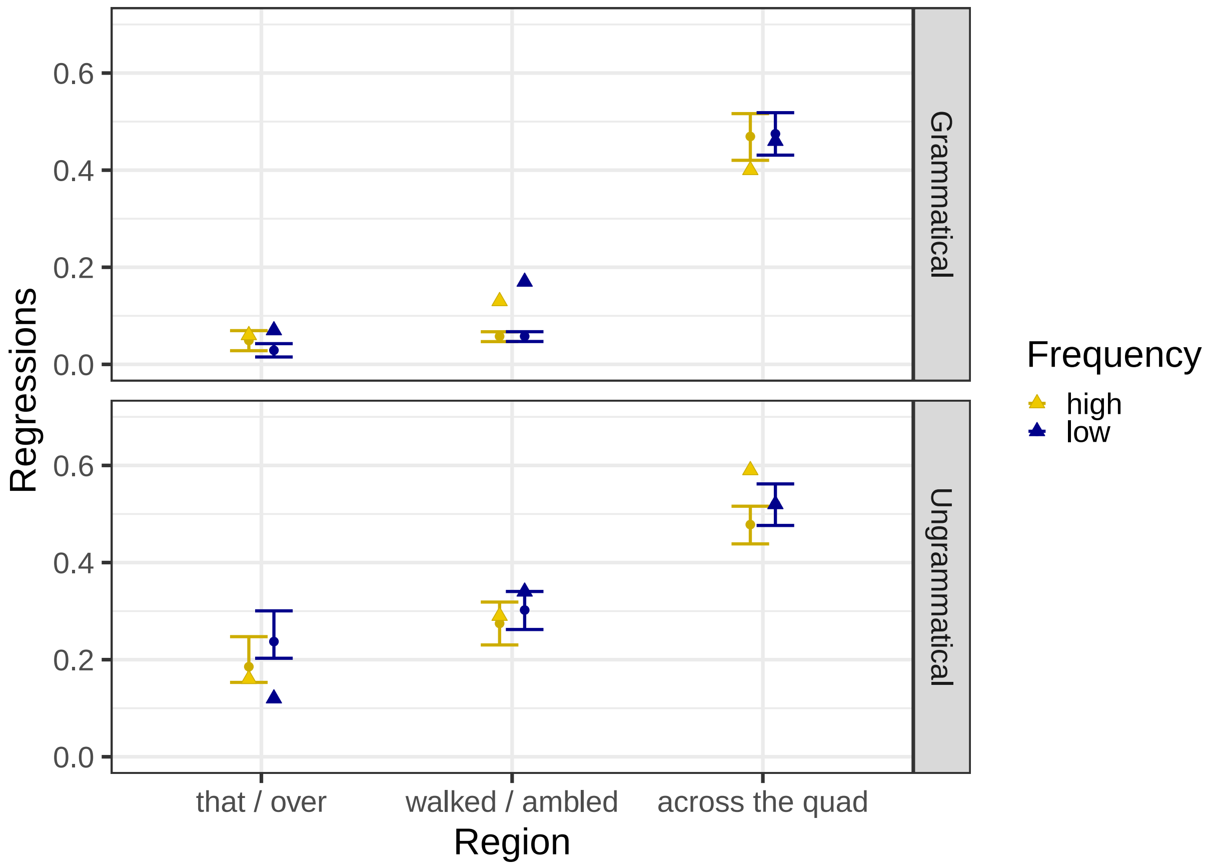
```
ggsave("staub-firstpass-inonegraph.pdf", width = 20, height = 15)
```

```
ndraws <- length(dataf2[, 1])

data.all <- data.frame(region = factor(rep(rep(c("that / over", "walked / ambled",
    "across the quad"), each = ndraws), 4), levels = c("that / over", "walked / ambled",
    "across the quad")), grammatical = c(rep("Grammatical", ndraws * 6), rep("Ungrammatical",
    ndraws * 6)), frequency = c(rep("high", ndraws * 3), rep("low", ndraws *
    3), rep("high", ndraws * 3), rep("low", ndraws * 3)), Reg = c(dataf2[,
    1], dataf2[, 2], dataf2[, 3], dataf2[, 4], dataf2[, 5], dataf2[, 6], dataf2[,
    7], dataf2[, 8], dataf2[, 9], dataf2[, 10], dataf2[, 11], dataf2[, 12]),
    x = rep(c(0.06, 0.13, 0.4, 0.07, 0.17, 0.46, 0.16, 0.29, 0.59, 0.12, 0.34,
        0.52), each = ndraws))

str(data.all)

## 'data.frame': 67056 obs. of  5 variables:
##  $ region     : Factor w/ 3 levels "that / over",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ grammatical: Factor w/ 2 levels "Grammatical",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ frequency  : Factor w/ 2 levels "high","low": 1 1 1 1 1 1 1 1 1 1 ...
##  $ Reg        : num  0.0318 0.0318 0.0318 0.0341 0.0341 ...
##  $ x          : num  0.06 0.06 0.06 0.06 0.06 0.06 0.06 0.06 0.06 0.06 ...

# data.all <- subset(data.all, RT > 50 & RT < 3000)

library(ggplot2)

library(dplyr)

data.to.plot <- data.all %>% group_by(region, grammatical, frequency) %>% summarise(Region = first(regi
    Grammatical = first(grammatical), Frequency = first(frequency), CF1 = quantile(Reg,
        probs = c(0.05, 0.95))[1], CF2 = quantile(Reg, probs = c(0.05, 0.95))[2],
    Regressions = mean(Reg), Observed = first(x))

## `summarise()` has grouped output by 'region', 'grammatical'. You can override using the
## `.groups` argument.

data.to.plot

## # A tibble: 12 x 10
## # Groups:   region, grammatical [6]
##    region grammatical frequency Region Grammatical Frequency    CF1    CF2
##    <fct>  <fct>       <fct>     <fct>  <fct>       <fct>      <dbl>  <dbl>
##  1 that ~ Grammatical high      that ~ Grammatical high      0.0282 0.0695
##  2 that ~ Grammatical low       that ~ Grammatical low       0.0152 0.0427
##  3 that ~ Ungrammati~ high      that ~ Ungrammati~ high      0.153  0.247
##  4 that ~ Ungrammati~ low       that ~ Ungrammati~ low       0.203  0.301
##  5 walke~ Grammatical high      walke~ Grammatical high      0.0469 0.0672
##  6 walke~ Grammatical low       walke~ Grammatical low       0.0471 0.0673
##  7 walke~ Ungrammati~ high      walke~ Ungrammati~ high      0.230  0.319
##  8 walke~ Ungrammati~ low       walke~ Ungrammati~ low       0.262  0.340
##  9 acros~ Grammatical high      acros~ Grammatical high      0.420  0.516
## 10 acros~ Grammatical low       acros~ Grammatical low       0.431  0.518
## 11 acros~ Ungrammati~ high      acros~ Ungrammati~ high      0.438  0.516
## 12 acros~ Ungrammati~ low       acros~ Ungrammati~ low       0.476  0.562
## # ... with 2 more variables: Regressions <dbl>, Observed <dbl>

g1 <- ggplot(data.to.plot, aes(Grammatical, Regressions, color = Frequency,
    fill = Frequency))
g1 <- g1 + geom_point(position = dodge, size = I(3)) + geom_errorbar(aes(ymin = CF1,
    ymax = CF2), position = dodge, width = 0.3, size = I(1.2)) + scale_shape_manual(values = 21:24) +
    scale_color_manual(values = c("gold3", "blue4")) + scale_fill_manual(values = c("gold2",
    "blue4")) + theme_bw(30)  # + theme(legend.justification = c(0.98, 0.9), legend.position = c(0.74,
g1 <- g1 + geom_point(aes(x = Grammatical, y = Observed, fill = Frequency),
    pch = 24, position = dodge, size = 4) + facet_grid(Region ~ .)
```



```
ggsave("staub-regressions.pdf", width = 20, height = 15)
```

```
g1 <- ggplot(data.to.plot, aes(Region, Regressions, color = Frequency, fill = Frequency))
g1 <- g1 + geom_point(position = dodge, size = I(3)) + geom_errorbar(aes(ymin = CF1,
    ymax = CF2), position = dodge, width = 0.3, size = I(1.2)) + scale_shape_manual(values = 21:24) +
    scale_color_manual(values = c("gold3", "blue4")) + scale_fill_manual(values = c("gold2",
    "blue4")) + theme_bw(30)  # + theme(legend.justification = c(0.98, 0.9), legend.position = c(0.74,
g1 <- g1 + coord_cartesian(ylim = c(0, 0.7)) + geom_point(aes(x = Region, y = Observed,
    fill = Frequency), pch = 24, position = dodge, size = 4) + facet_grid(Grammatical ~
    .)
```



```
ggsave("staub-regressions-inonegraph.pdf", width = 20, height = 15)
```

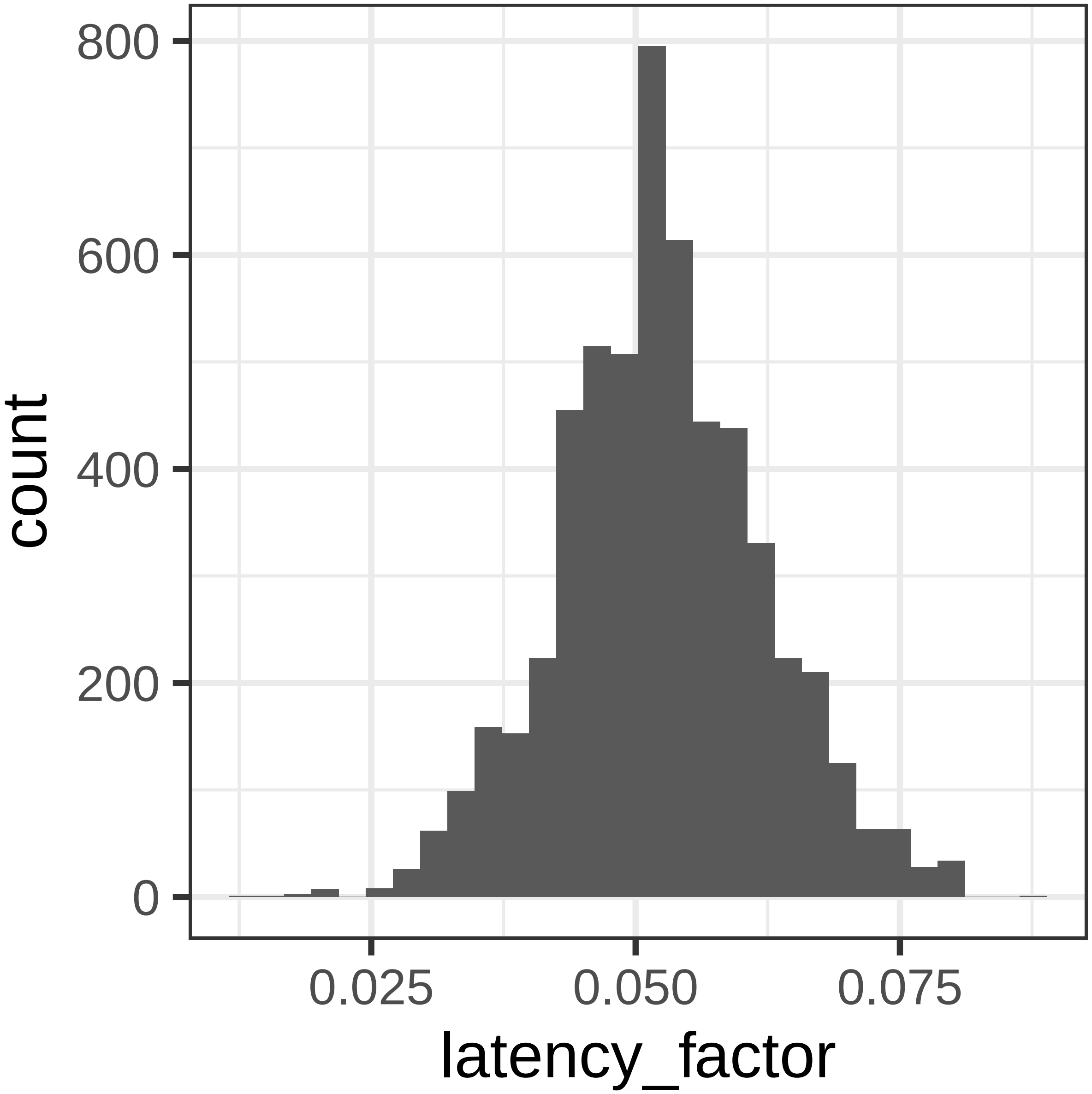
2 Parameters

2.1 LF

```
############# PARAMS###########
draws <- createdraws("lf")

## Note: Using an external vector in selections is ambiguous.
## i Use `all_of(param)` instead of `param` to silence this message.
## i See <https://tidyselect.r-lib.org/reference/faq-external-vector.html>.
## This message is displayed once per session.

str(draws)

##  num [1:2794, 1:2] 0.0556 0.0556 0.0404 0.0483 0.0483 ...

Rhat(draws)

## [1] 1.011642
```

Mean etc.

```r
tail(draws)

##                [,1]       [,2]
## [2789,] 0.05125586 0.04595704
## [2790,] 0.05125586 0.04595704
## [2791,] 0.05125586 0.04595704
## [2792,] 0.05551180 0.04595704
## [2793,] 0.05551180 0.04595704
## [2794,] 0.05551180 0.04595704

mean(c(draws[, 1:2]))

## [1] 0.0524086

median(c(draws[, 1:2]))

## [1] 0.05165914

sd(c(draws[, 1:2]))

## [1] 0.009714798

g1 <- ggplot(data.frame(latency_factor = c(draws[, 1:2])), aes(latency_factor))
g1 <- g1 + geom_histogram() + theme_bw(28)
g1

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



```
ggsave("staub-lf.pdf", width = 20, height = 12)

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

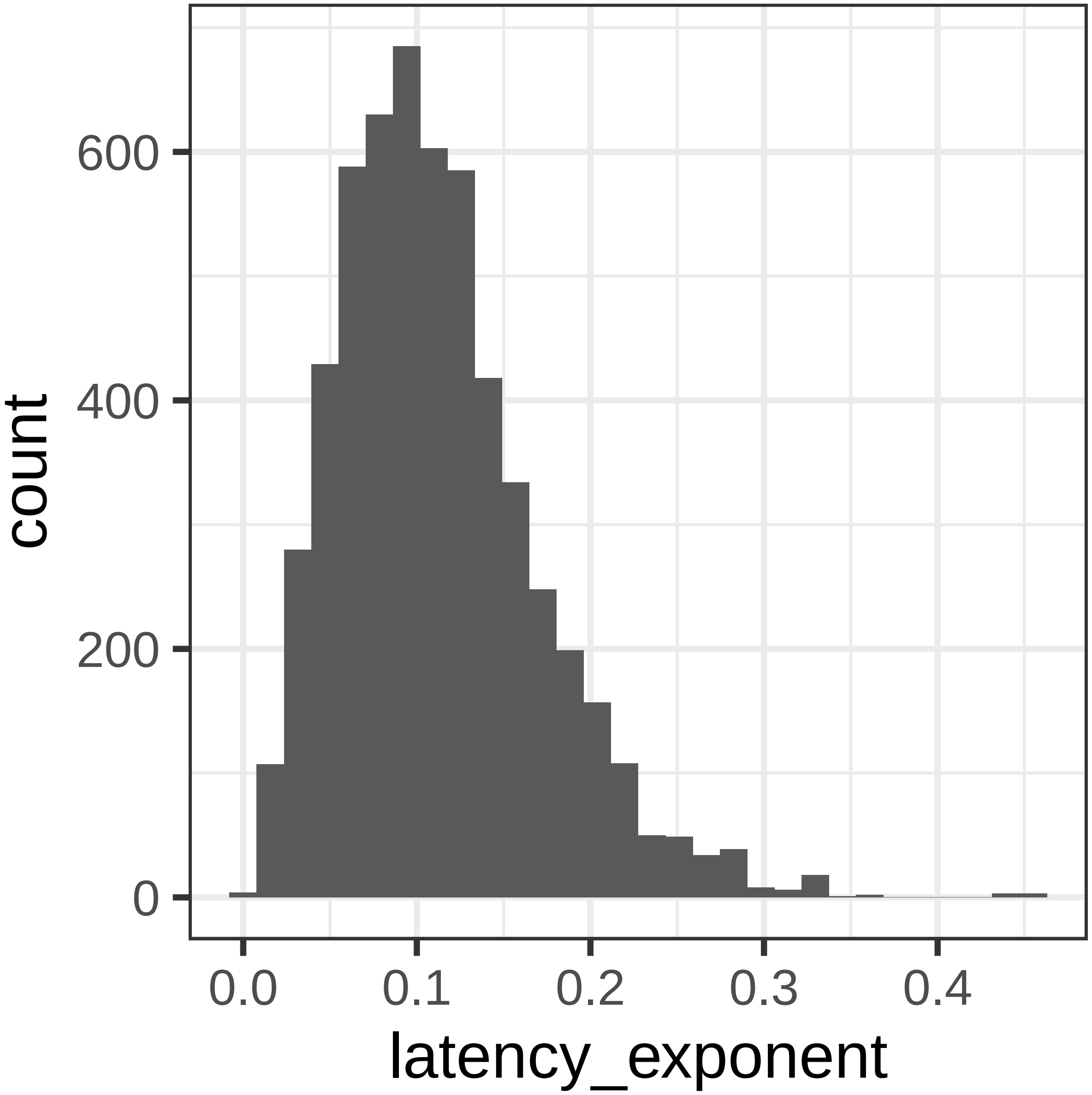
2.2 LE

```
############# PARAMS###########
draws <- createdraws("le")

str(draws)

##  num [1:2794, 1:2] 0.1 0.1 0.1 0.131 0.131 ...
```

```
Rhat(draws)

## [1] 1.002964
```

Mean etc.

```
tail(draws)

##              [,1]       [,2]
## [2789,] 0.1217190 0.07315806
## [2790,] 0.1566882 0.09953382
## [2791,] 0.1220705 0.09953382
## [2792,] 0.1220705 0.09953382
## [2793,] 0.1220705 0.09953382
## [2794,] 0.1220705 0.09953382

mean(c(draws[, 1:2]))

## [1] 0.1119095

median(c(draws[, 1:2]))

## [1] 0.1031822

sd(c(draws[, 1:2]))

## [1] 0.05774515

g1 <- ggplot(data.frame(latency_exponent = c(draws[, 1:2])), aes(latency_exponent))
g1 <- g1 + geom_histogram() + theme_bw(28)
g1

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



```
ggsave("staub-le.pdf", width = 20, height = 12)

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

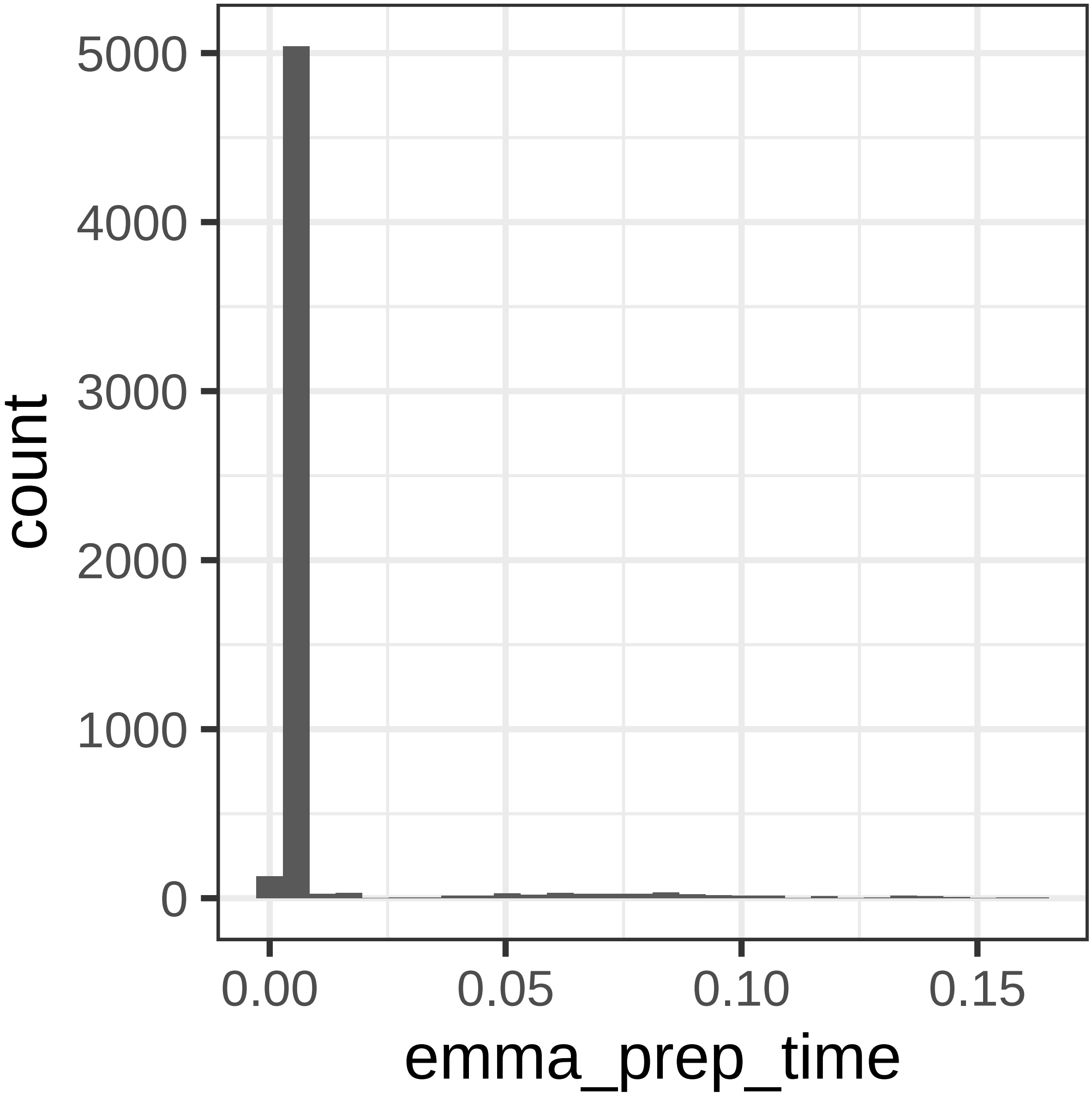
2.3 Emma preparation time

```
############# PARAMS###########
draws <- createdraws("emma_prep_time")

str(draws)

##  num [1:2794, 1:2] 0.00924 0.00407 0.00407 0.00428 0.00689 ...
```

```
Rhat(draws)

## [1] 1.021713
```

Mean etc.

```
tail(draws)

##                [,1]         [,2]
## [2789,] 0.004869950 0.0006885783
## [2790,] 0.003709092 0.0006885783
## [2791,] 0.003709092 0.0006885783
## [2792,] 0.005699268 0.0046673239
## [2793,] 0.005699268 0.0046673239
## [2794,] 0.005699268 0.0046673239

mean(c(draws[, 1:2]))

## [1] 0.01054056

median(c(draws[, 1:2]))

## [1] 0.005892357

sd(c(draws[, 1:2]))

## [1] 0.02030516

g1 <- ggplot(data.frame(emma_prep_time = c(draws[, 1:2])), aes(emma_prep_time))
g1 <- g1 + geom_histogram(xlim = c(0, 0.05)) + theme_bw(28)

## Warning: Ignoring unknown parameters: xlim

g1

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



```
ggsave("staub-e.pdf", width = 20, height = 12)

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

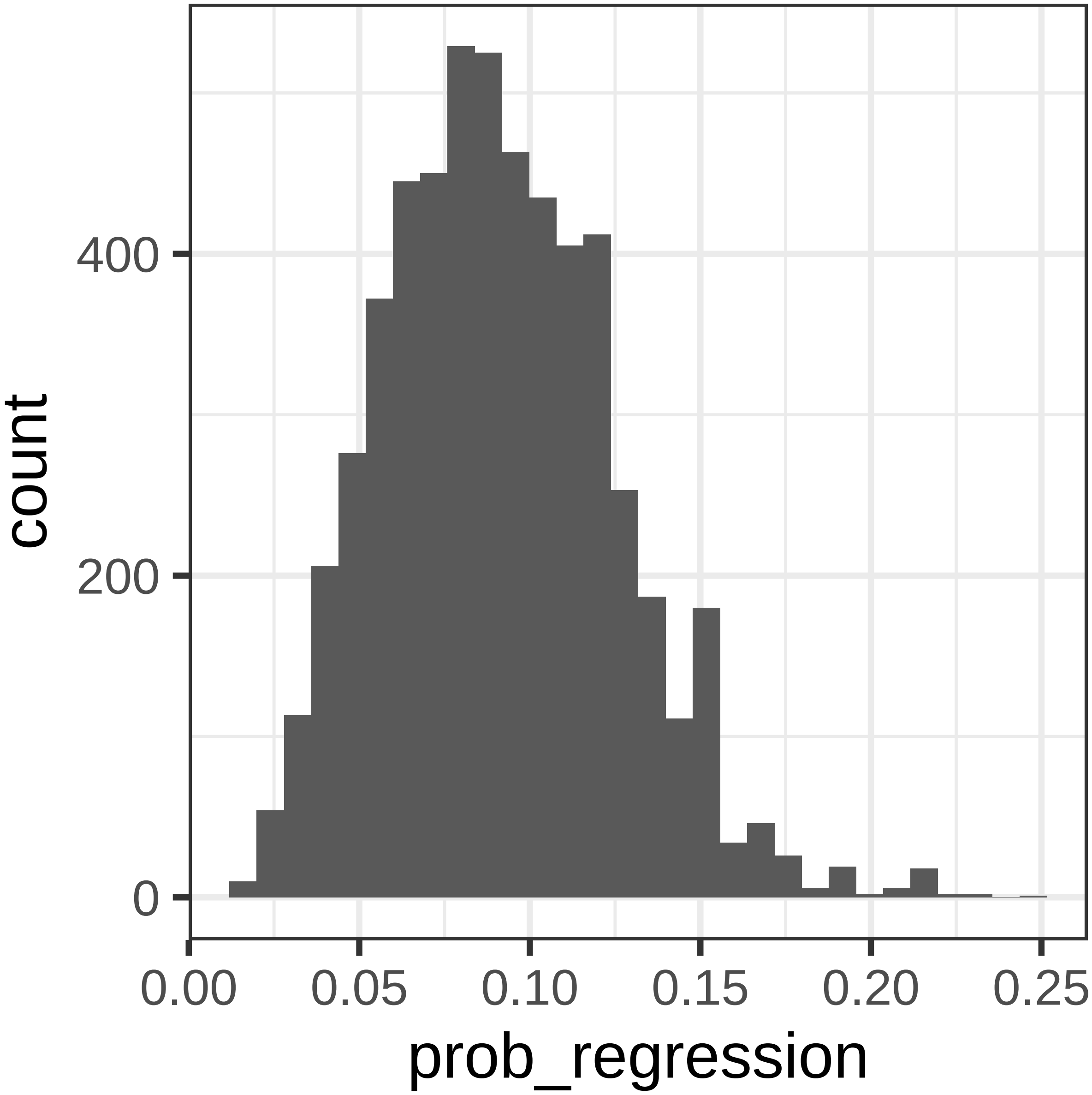
2.4 Prob regression

```
############# PARAMS###########
draws <- createdraws("prob_regression")

str(draws)

##  num [1:2794, 1:2] 0.122 0.122 0.122 0.122 0.122 ...
```

```
Rhat(draws)

## [1] 1.016652
```

Mean etc.

```
tail(draws)

##              [,1]       [,2]
## [2789,] 0.1451989 0.09932388
## [2790,] 0.1451989 0.09932388
## [2791,] 0.1451989 0.09932388
## [2792,] 0.1451989 0.09932388
## [2793,] 0.1405039 0.11544639
## [2794,] 0.1405039 0.11544639

mean(c(draws[, 1:2]))

## [1] 0.09149012

median(c(draws[, 1:2]))

## [1] 0.0885908

sd(c(draws[, 1:2]))

## [1] 0.03379763

g1 <- ggplot(data.frame(prob_regression = c(draws[, 1:2])), aes(prob_regression))
g1 <- g1 + geom_histogram() + theme_bw(28)
g1

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



```
ggsave("staub-p.pdf", width = 20, height = 12)

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

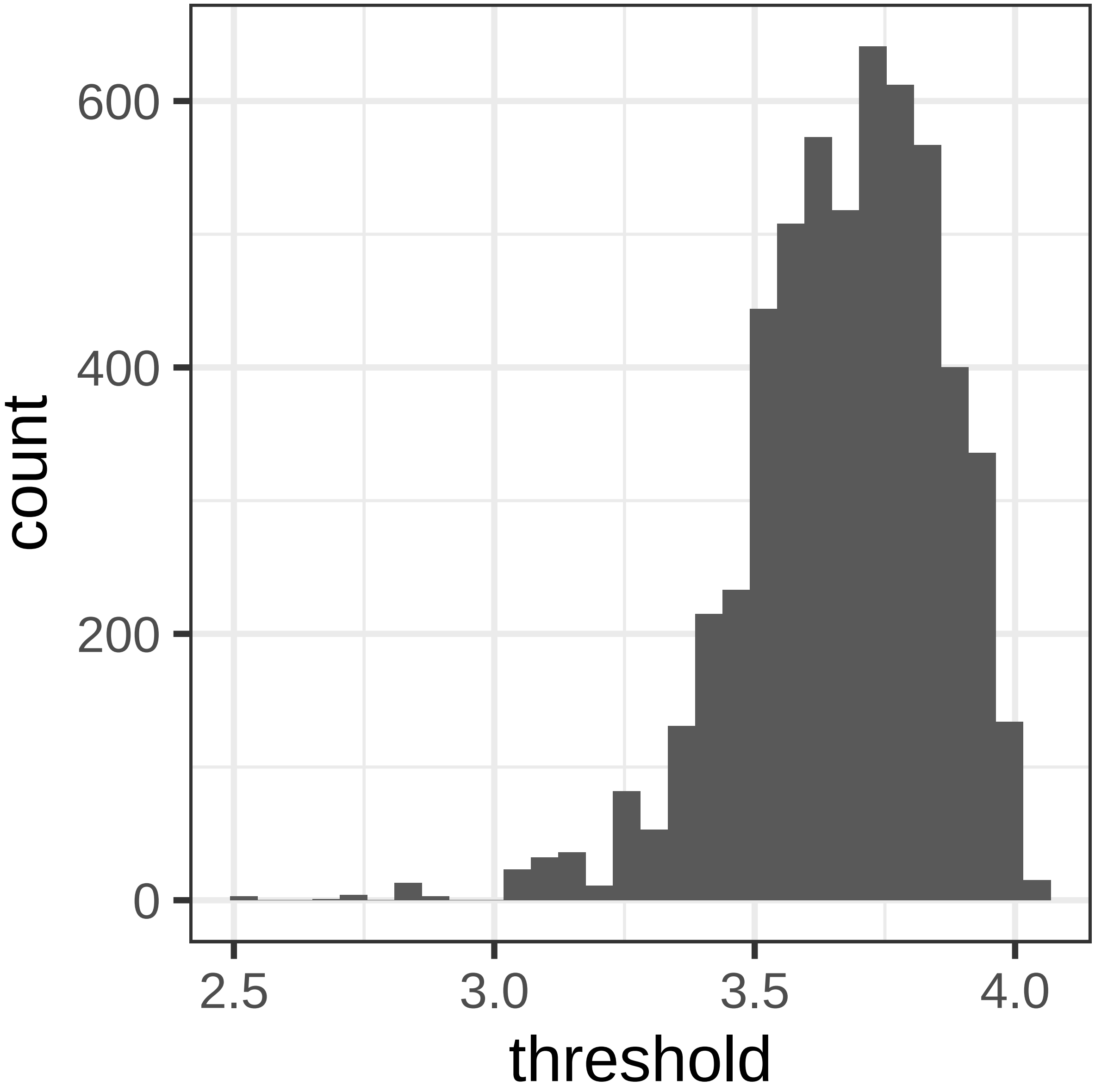
2.5 Threshold

```
############# PARAMS###########
draws <- createdraws("threshold")

str(draws)

##  num [1:2794, 1:2] 3.45 3.45 3.45 3.48 3.48 ...
```

```
Rhat(draws)

## [1] 1.024723
```

Mean etc.

```
tail(draws)

##              [,1]    [,2]
## [2789,] 3.529291 3.56383
## [2790,] 3.529291 3.56383
## [2791,] 3.529291 3.56383
## [2792,] 3.529291 3.56383
## [2793,] 3.529291 3.56383
## [2794,] 3.529291 3.56383

mean(c(draws[, 1:2]))

## [1] 3.669984

median(c(draws[, 1:2]))

## [1] 3.690112

sd(c(draws[, 1:2]))

## [1] 0.1926334

g1 <- ggplot(data.frame(threshold = c(draws[, 1:2])), aes(threshold))
g1 <- g1 + geom_histogram() + theme_bw(28)
g1

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



```
ggsave("staub-t.pdf", width = 20, height = 12)

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

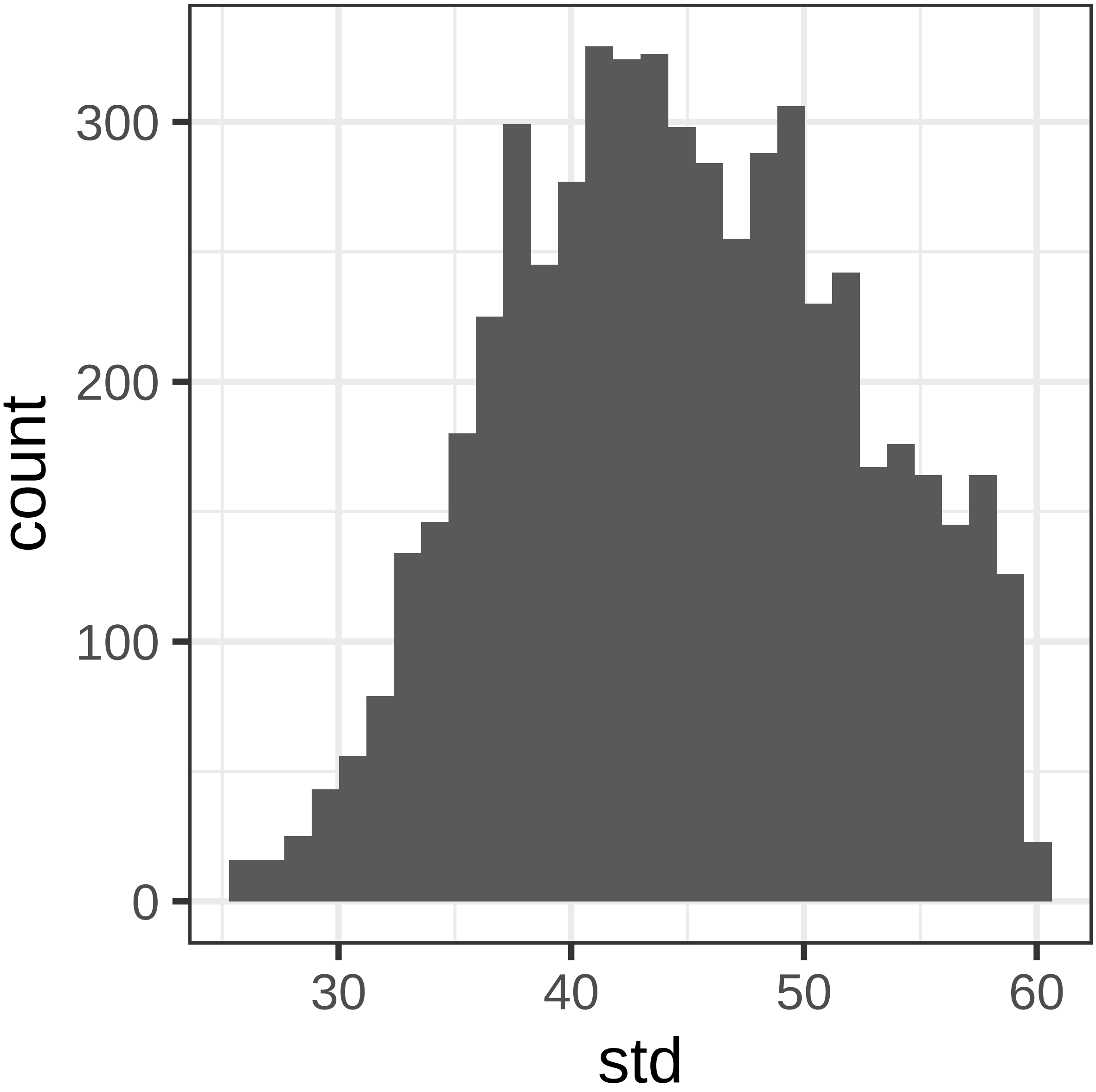
2.6 Std

```
############# PARAMS###########
draws <- createdraws("std")

str(draws)

##  num [1:2794, 1:2] 45.6 49.4 43.7 28.7 32.6 ...
```

```
Rhat(draws)

## [1] 1.000528
```

Mean etc.

```
tail(draws)

##              [,1]    [,2]
## [2789,] 35.62718 33.0111
## [2790,] 35.62718 33.0111
## [2791,] 35.62718 33.0111
## [2792,] 38.58657 39.7666
## [2793,] 38.58657 39.7666
## [2794,] 35.67267 39.7666

mean(c(draws[, 1:2]))

## [1] 44.73057

median(c(draws[, 1:2]))

## [1] 44.43172

sd(c(draws[, 1:2]))

## [1] 7.442251

g1 <- ggplot(data.frame(std = c(draws[, 1:2])), aes(std))
g1 <- g1 + geom_histogram() + theme_bw(28)
g1

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



```
ggsave("staub-std.pdf", width = 20, height = 12)

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