Study 2

Contents

This file reproduces the preprocessing and analysis steps of Study 2. The data are automatically imported from Github and necessary packages will be downloaded and installed if they are not yet available.

```
rm(list=ls())
if (!require("pacman"))
  install.packages("pacman")
required_packages = c(
  "tidyverse",
  "afex",
  'brms',
  'BayesFactor',
  'emmeans',
  'gridExtra'
)
pacman::p_load(required_packages,
               update = F,
               character.only = T
)
```

```
sessionInfo()
```

```
## R version 4.0.3 (2020-10-10)
## Platform: x86_64-apple-darwin17.0 (64-bit)
## Running under: macOS Big Sur 10.16
##
## Matrix products: default
## BLAS:   /Library/Frameworks/R.framework/Versions/4.0/Resources/lib/libRblas.dylib
## LAPACK: /Library/Frameworks/R.framework/Versions/4.0/Resources/lib/libRlapack.dylib
##
## locale:
## [1] en_US.UTF-8/en_US.UTF-8/en_US.UTF-8/C/en_US.UTF-8/en_US.UTF-8
##
## attached base packages:
## [1] stats     graphics  grDevices utils     datasets  methods   base
##
```

```
## other attached packages:
##  [1] gridExtra_2.3          emmeans_1.5.3          BayesFactor_0.9.12-4.2
##  [4] coda_0.19-4            brms_2.14.4            Rcpp_1.0.5
##  [7] afex_0.28-0            lme4_1.1-26            Matrix_1.2-18
## [10] forcats_0.5.0          stringr_1.4.0          dplyr_1.0.2
## [13] purrr_0.3.4            readr_1.4.0            tidyr_1.1.2
## [16] tibble_3.0.4           ggplot2_3.3.2          tidyverse_1.3.0
## [19] pacman_0.5.1
##
## loaded via a namespace (and not attached):
##   [1] readxl_1.3.1         backports_1.2.1      plyr_1.8.6
##   [4] igraph_1.2.6         splines_4.0.3        crosstalk_1.1.0.1
##   [7] TH.data_1.0-10       rstantools_2.1.1     inline_0.3.17
##  [10] digest_0.6.27        htmltools_0.5.0      rsconnect_0.8.16
##  [13] lmerTest_3.1-3       fansi_0.4.1          magrittr_2.0.1
##  [16] openxlsx_4.2.3       modelr_0.1.8         RcppParallel_5.0.2
##  [19] matrixStats_0.57.0   xts_0.12.1           sandwich_3.0-0
##  [22] prettyunits_1.1.1    colorspace_2.0-0     rvest_0.3.6
##  [25] haven_2.3.1          xfun_0.19            callr_3.5.1
##  [28] crayon_1.3.4         jsonlite_1.7.2       survival_3.2-7
##  [31] zoo_1.8-8            glue_1.4.2           gtable_0.3.0
##  [34] MatrixModels_0.4-1   V8_3.4.0             car_3.0-10
##  [37] pkgbuild_1.1.0       rstan_2.21.3         abind_1.4-5
##  [40] scales_1.1.1         mvtnorm_1.1-1        DBI_1.1.0
##  [43] miniUI_0.1.1.1       xtable_1.8-4         foreign_0.8-80
##  [46] StanHeaders_2.21.0-6 stats4_4.0.3         DT_0.16
##  [49] htmlwidgets_1.5.3    httr_1.4.2           threejs_0.3.3
##  [52] ellipsis_0.3.1       pkgconfig_2.0.3      loo_2.4.1
##  [55] dbplyr_2.0.0         tidyselect_1.1.0     rlang_0.4.9
##  [58] reshape2_1.4.4       later_1.1.0.1        munsell_0.5.0
##  [61] cellranger_1.1.0     tools_4.0.3          cli_2.2.0
##  [64] generics_0.1.0       broom_0.7.2          ggridges_0.5.2
##  [67] evaluate_0.14        fastmap_1.0.1        yaml_2.2.1
##  [70] processx_3.4.5       knitr_1.30           fs_1.5.0
##  [73] zip_2.1.1            pbapply_1.4-3        nlme_3.1-149
##  [76] mime_0.9             projpred_2.0.2       xml2_1.3.2
##  [79] compiler_4.0.3       bayesplot_1.7.2      shinythemes_1.1.2
##  [82] rstudioapi_0.13      curl_4.3             gamm4_0.2-6
##  [85] reprex_0.3.0         statmod_1.4.35       stringi_1.5.3
##  [88] ps_1.5.0             Brobdingnag_1.2-6    lattice_0.20-41
##  [91] nloptr_1.2.2.2       markdown_1.1         shinyjs_2.0.0
##  [94] vctrs_0.3.5          pillar_1.4.7         lifecycle_0.2.0
##  [97] bridgesampling_1.0-0 estimability_1.3     data.table_1.13.4
## [100] httpuv_1.5.4         R6_2.5.0             promises_1.1.1
## [103] rio_0.5.16           codetools_0.2-16     boot_1.3-25
## [106] colourpicker_1.1.0   MASS_7.3-53          gtools_3.8.2
## [109] assertthat_0.2.1     withr_2.3.0          shinystan_2.5.0
## [112] multcomp_1.4-15      mgcv_1.8-33          parallel_4.0.3
## [115] hms_0.5.3            grid_4.0.3           minqa_1.2.4
## [118] rmarkdown_2.6        carData_3.0-4        numDeriv_2016.8-1.1
## [121] shiny_1.5.0          lubridate_1.7.9.2    base64enc_0.1-3
## [124] dygraphs_1.1.1.6
```

Import data

```
df = read.csv('https://raw.githubusercontent.com/mertensu/thinking-in-ratios/master/data_total_study2.cs
```

Preprocessing

```
# replace zero ratings with 0.001
df[df$brightness_rating==0,'brightness_rating'] = 0.001
# df = df[df$brightness_rating!=0, ]


# compute correct binary choice
df = df %>% mutate(brightness_dichotom_correct = ifelse(cd < 10, 0, 1))

df$correct_response = NA
df[df$condition == 'unidirectional', 'correct_response'] = df[df$condition == 'unidirectional', 'brightn

# remove wrong binary choice(s) in unidirectional condition
df = df %>% filter(!(correct_response == F &
                       condition == 'unidirectional'))

df[, "brightness_rating_stevens"] = ifelse((df$condition == "unidirectional") &
                                             (df$brightness_dichotom_correct == 0),
                                           100 / df$brightness_rating,
                                           df$brightness_rating
)
df[, "brightness_rating"] = ifelse((df$condition == "standard") &
                                     (df$brightness_dichotom_correct == 0),
                                   10 * (10 / df$brightness_rating),
                                   df$brightness_rating
)

df$log_brightness_rating = log(df$brightness_rating_stevens)

df$cd_factor = factor(df$cd, levels = c(1, 1.8, 3.2, 5.7, 17.9, 32.0, 57.2, 100.0))

df$brightness_rating_stevens = df$brightness_rating_stevens/100
df$cd = df$cd/100
```

Demographics

```
psych::describe(df$age)
```

```
##    vars    n  mean   sd median trimmed  mad min max range skew kurtosis   se
## X1    1 1359 23.77 5.83     23   22.67 2.97  18  46    28 2.57     6.88 0.16
```

```
df %>% distinct(File, .keep_all = T) %>% group_by(gender) %>% summarise(
  N = n(),
  Min =
    min(age),
  Max =
    max(age),
  Mean =
    mean(age),
  Sd =
    sd(age)
)
```

```
## `summarise()` ungrouping output (override with `.groups` argument)
```

```
## # A tibble: 2 x 6
##   gender     N   Min   Max  Mean    Sd
##   <chr>  <int> <int> <int> <dbl> <dbl>
## 1 m          5    21    46  28   10.2
## 2 w         29    18    43  23.0  4.75
```

```
df %>% distinct(File, .keep_all = T) %>% count(student)
```

```
##   student  n
## 1       0  3
## 2       1 31
```

```
df %>% distinct(File, .keep_all = T) %>% filter(student == 1) %>% count(psycho)
```

```
##   psycho  n
## 1      0 21
## 2      1 10
```

```
df %>% distinct(File, .keep_all = T) %>% count(condition)
```

```
##        condition  n
## 1       standard 16
## 2 unidirectional 18
```

Analysis

ANOVA I (mixed 2(method) x 8(luminance))

```
(
  fit = aov_ez(
    dv = "log_brightness_rating",
    within = 'cd_factor',
    between = "condition",
```

```
    id = "File",
    data = df
  )
)
```

frequentist fit

```
## Converting to factor: condition
```

```
## Warning: More than one observation per cell, aggregating the data using mean
## (i.e, fun_aggregate = mean)!
```

```
## Contrasts set to contr.sum for the following variables: condition
```

```
## Anova Table (Type 3 tests)
##
## Response: log_brightness_rating
##                Effect          df  MSE          F  ges p.value
## 1           condition       1, 32 0.14  21.73 *** .090   <.001
## 2           cd_factor 2.02, 64.70 0.39 176.75 *** .825   <.001
## 3 condition:cd_factor 2.02, 64.70 0.39     4.71 * .112    .012
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '+' 0.1 ' ' 1
##
## Sphericity correction method: GG
```

```
df$File = factor(df$File)
df$condition = factor(df$condition)
bfs = anovaBF(
  log_brightness_rating ~ cd_factor * condition + File,
  whichRandom = 'File',
  whichModels = 'top',
  data = df
)

# BF cd_factor
bf_1 = lmBF(log_brightness_rating ~ condition + File,
            whichRandom = 'File',
            data = df)
bf_2 = lmBF(
  log_brightness_rating ~ cd_factor + condition + File,
  whichRandom = 'File',
  data = df
)
(bf_cd_factor = bf_2 / bf_1)
```

bayesian fit

```
## Bayes factor analysis
## --------------
## [1] cd_factor + condition + File : 1.169377e+326 ±2.9%
##
## Against denominator:
##   log_brightness_rating ~ condition + File
## ---
## Bayes factor type: BFlinearModel, JZS
```

```
print(paste0('logBF ', bf_cd_factor@bayesFactor$bf))
```

```
## [1] "logBF 750.79921133321"
```

```
# BF condition
bf_1 = lmBF(log_brightness_rating ~ cd_factor + File,
            whichRandom = 'File',
            data = df)
bf_2 = lmBF(
  log_brightness_rating ~ cd_factor + condition + File,
  whichRandom = 'File',
  data = df
)
(bf_condition = bf_2 / bf_1)
```

```
## Bayes factor analysis
## --------------
## [1] cd_factor + condition + File : 32.11403 ±2.74%
##
## Against denominator:
##   log_brightness_rating ~ cd_factor + File
## ---
## Bayes factor type: BFlinearModel, JZS
```

```
print(paste0('logBF ', bf_condition@bayesFactor$bf))
```

```
## [1] "logBF 3.46929288932154"
```

```
# BF interaction
bf_1 = lmBF(
  log_brightness_rating ~ cd_factor + condition + File,
  whichRandom = 'File',
  data = df
)
bf_2 = lmBF(
  log_brightness_rating ~ cd_factor * condition + File,
  whichRandom = 'File',
  data = df
)
(bf_interaction = bf_2 / bf_1)
```

```
## Bayes factor analysis
```

```
## --------------
## [1] cd_factor * condition + File : 373924496607 ±1.78%
##
## Against denominator:
##   log_brightness_rating ~ cd_factor + condition + File
## ---
## Bayes factor type: BFlinearModel, JZS
```

```
print(paste0('logBF ', bf_interaction@bayesFactor$bf))
```

```
## [1] "logBF 26.6473197332501"
```

```
scaleFUN <- function(x)
  sprintf("%.1f", x)

grid = data.frame(emmeans(fit,  ~ cd_factor + condition))

ggplot(grid, aes(
  x = cd_factor,
  y = exp(emmean),
  group = condition
)) +
  geom_pointrange(aes(
    ymin = exp(lower.CL),
    ymax = exp(upper.CL),
    color = condition
  ), size =
    0.3) +
  scale_y_continuous(
    trans = 'log2',
    breaks = c(1.0, 10.0, 100.0),
    limits = c(1.0, 100.0),
    labels = scaleFUN
  ) +
  geom_line(linetype = 'dashed') +
  labs(y = 'Brightness judgement') +
  xlab(expression(paste("cd/", m ^ 2, sep = ""))) +
  scale_color_manual(
    values = c("darkgrey", "black"),
    name = "method",
    labels = c("unidirectional", "standard")
  ) +
  scale_x_discrete(labels = substring(grid$cd_factor, 2)) +
  theme_classic() +
  theme(
    legend.position = c(0.2, 0.8),
    legend.background = element_rect(color = "black")
  )
```

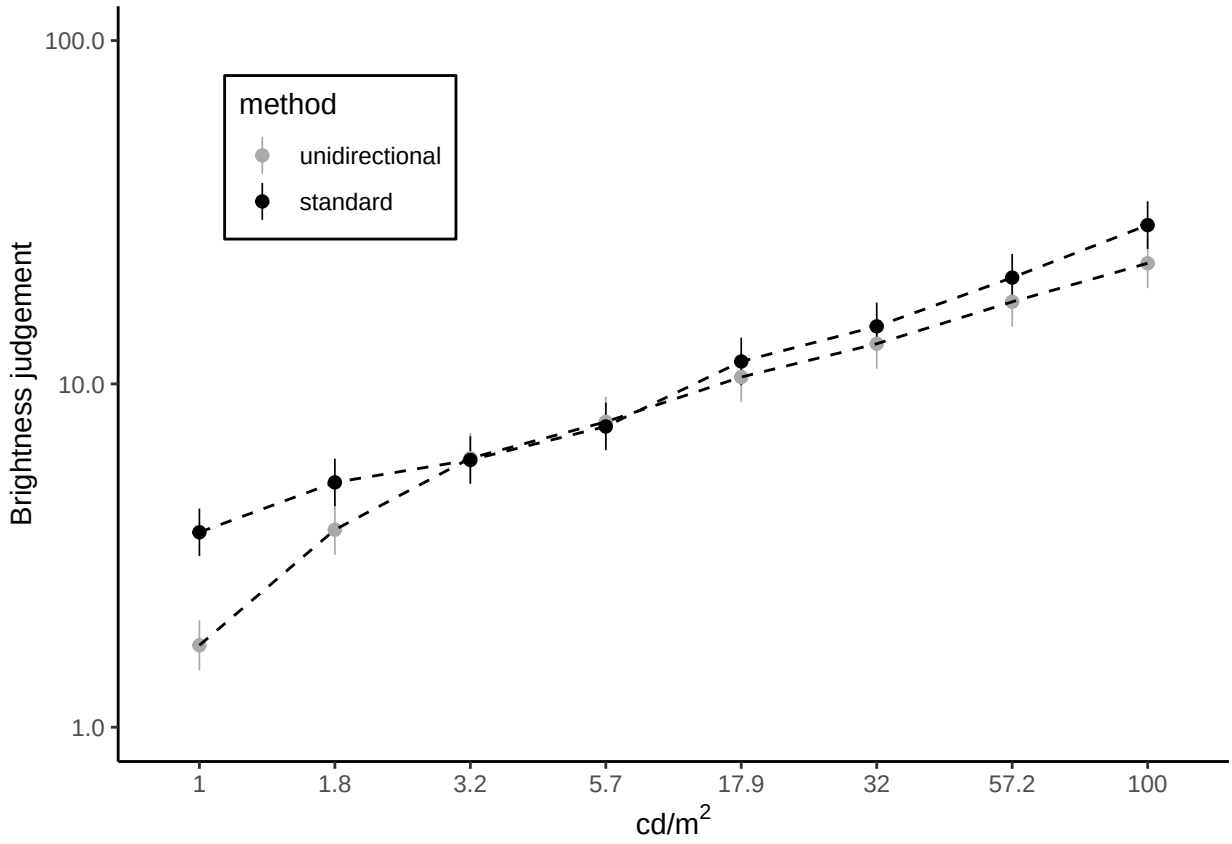


Figure 2

```
ggsave(
  paste0("final_plots/study2_figure2.png"),
  dpi = 600,
  height = 4,
  width = 5,
  units = "in"
)
```

Bayesian mixed effects models (standard condition)

```
prior_settings <- c(set_prior('normal(0, 5)', nlpar = "a"),
                    set_prior('normal(1.0, 0.5)', nlpar = "b", lb = 0))
```

```
(formula = bf(brightness_rating_stevens ~ a * cd ^ b, a ~ (1 |File), b ~ (1 | File), nl = TRUE))
```

Fit power law

```
## brightness_rating_stevens ~ a * cd^b
## a ~ (1 | File)
## b ~ (1 | File)
```

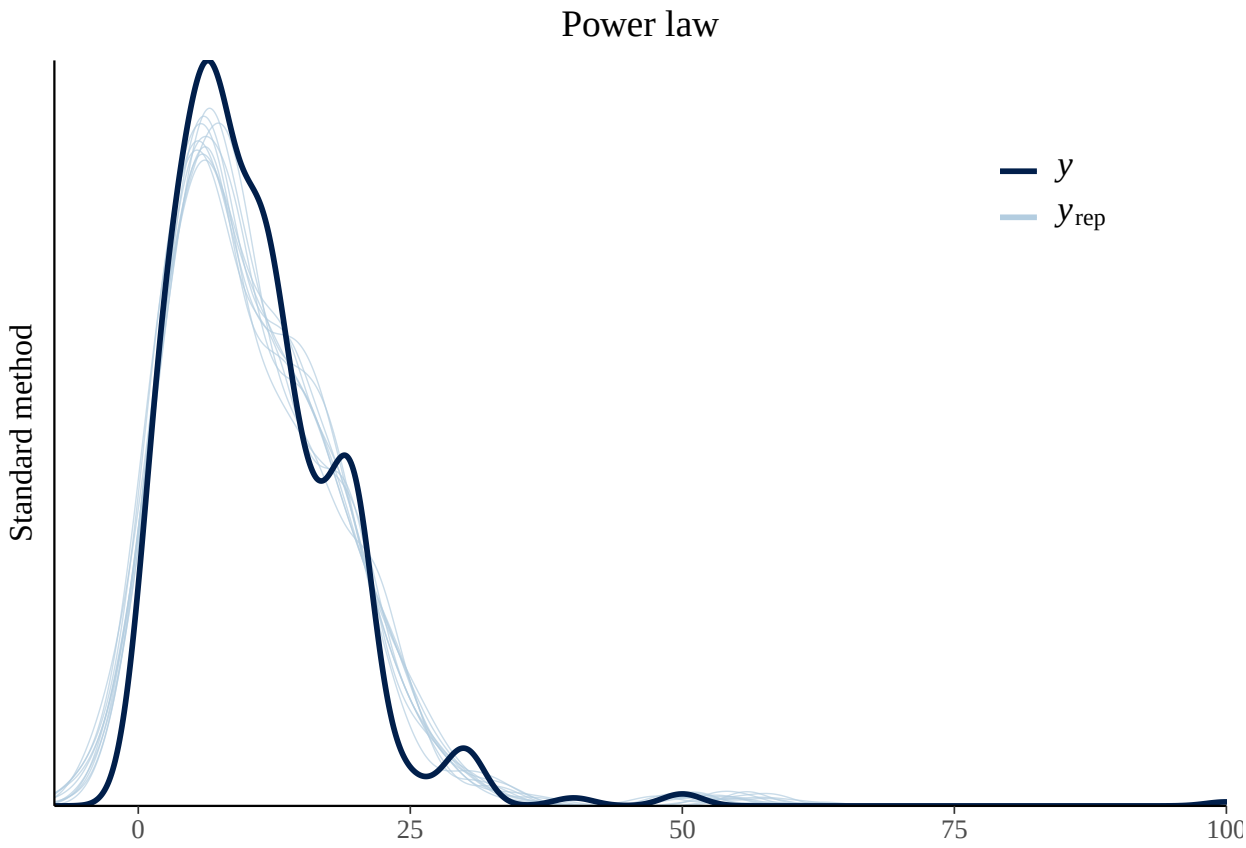
```
# make_stancode(formula, data = df[df$condition == 'standard', ], prior = prior_settings)
```

```
(power_law_standard <-
  brm(formula,
    data = df[df$condition == 'standard', ],
    save_all_pars = T,
    sample_prior = 'yes',
    warmup = 2000,
    iter = 7000,
    cores = 4,
    control = list(adapt_delta = 0.96, max_treedepth = 20),
    prior = prior_settings,
    file = 'study2_power_standard_x'))
```

```
##  Family: gaussian
##   Links: mu = identity; sigma = identity
## Formula: brightness_rating_stevens ~ a * cd^b
##          a ~ (1 | File)
##          b ~ (1 | File)
##    Data: df[df$condition == "standard", ] (Number of observations: 640)
## Samples: 4 chains, each with iter = 7000; warmup = 2000; thin = 1;
##          total post-warmup samples = 20000
##
## Group-Level Effects:
## ~File (Number of levels: 16)
##                 Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
## sd(a_Intercept)     0.09      0.02     0.06     0.14 1.00     5851     9960
## sd(b_Intercept)     0.18      0.04     0.12     0.27 1.00     5892     8201
##
## Population-Level Effects:
##             Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
## a_Intercept     0.23      0.02     0.19     0.28 1.00     2723     5081
## b_Intercept     0.44      0.05     0.35     0.53 1.00     3523     6853
##
## Family Specific Parameters:
##       Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
## sigma     0.03      0.00     0.03     0.03 1.00    22498    13684
##
## Samples were drawn using sampling(NUTS). For each parameter, Bulk_ESS
## and Tail_ESS are effective sample size measures, and Rhat is the potential
## scale reduction factor on split chains (at convergence, Rhat = 1).
```

```
(
  pow_standard_plot = pp_check(power_law_standard) +
    labs(title = 'Power law', y = 'Standard method') +
    scale_x_continuous(labels = c(0, 25, 50, 75, 100)) +
    theme(
      plot.title = element_text(hjust = 0.5),
      legend.position = c(0.85, 0.85),
    )
)
```

```
## Using 10 posterior samples for ppc type 'dens_overlay' by default.
```




```
brms::loo(power_law_standard)
```

Model metrics (LOOCV, WAIC, marginal likelihood)

```
## Warning: Found 4 observations with a pareto_k > 0.7 in model
## 'power_law_standard'. It is recommended to set 'moment_match = TRUE' in order to
## perform moment matching for problematic observations.

##
## Computed from 20000 by 640 log-likelihood matrix
##
##          Estimate    SE
## elpd_loo   1254.8 142.1
## p_loo        89.2  51.9
## looic     -2509.6 284.2
## ------
## Monte Carlo SE of elpd_loo is NA.
##
## Pareto k diagnostic values:
##                          Count Pct.    Min. n_eff
## (-Inf, 0.5]   (good)     636   99.4%   3143
##  (0.5, 0.7]   (ok)         0    0.0%   <NA>
##    (0.7, 1]   (bad)        2    0.3%   57
```

```
##    (1, Inf)   (very bad)   2    0.3%   2
## See help('pareto-k-diagnostic') for details.
```

```
brms::waic(power_law_standard)
```

```
## Warning:
## 15 (2.3%) p_waic estimates greater than 0.4. We recommend trying loo instead.

##
## Computed from 20000 by 640 log-likelihood matrix
##
##           Estimate    SE
## elpd_waic   1228.0 168.9
## p_waic       116.0  79.0
## waic       -2456.0 337.8
##
## 15 (2.3%) p_waic estimates greater than 0.4. We recommend trying loo instead.
```

```
brms::bridge_sampler(power_law_standard)
```

```
## Iteration: 1
## Iteration: 2
## Iteration: 3
## Iteration: 4
## Iteration: 5
## Iteration: 6
## Iteration: 7

## Bridge sampling estimate of the log marginal likelihood: 1223.806
## Estimate obtained in 7 iteration(s) via method "normal".
```

```
prior_settings <-
  c(
    set_prior('normal(0,1)', class = 'b', coef = 'cd'),
    set_prior('normal(0,5)', class = 'b', coef = 'Intercept')
  )
```

```
(formula = brightness_rating_stevens ~ 0 + Intercept + cd + (1 + cd | File))
```

Fit linear model

```
## brightness_rating_stevens ~ 0 + Intercept + cd + (1 + cd | File)
```

```
# make_stancode(formula, data = df[df$condition == 'standard', ], prior = prior_settings)
```

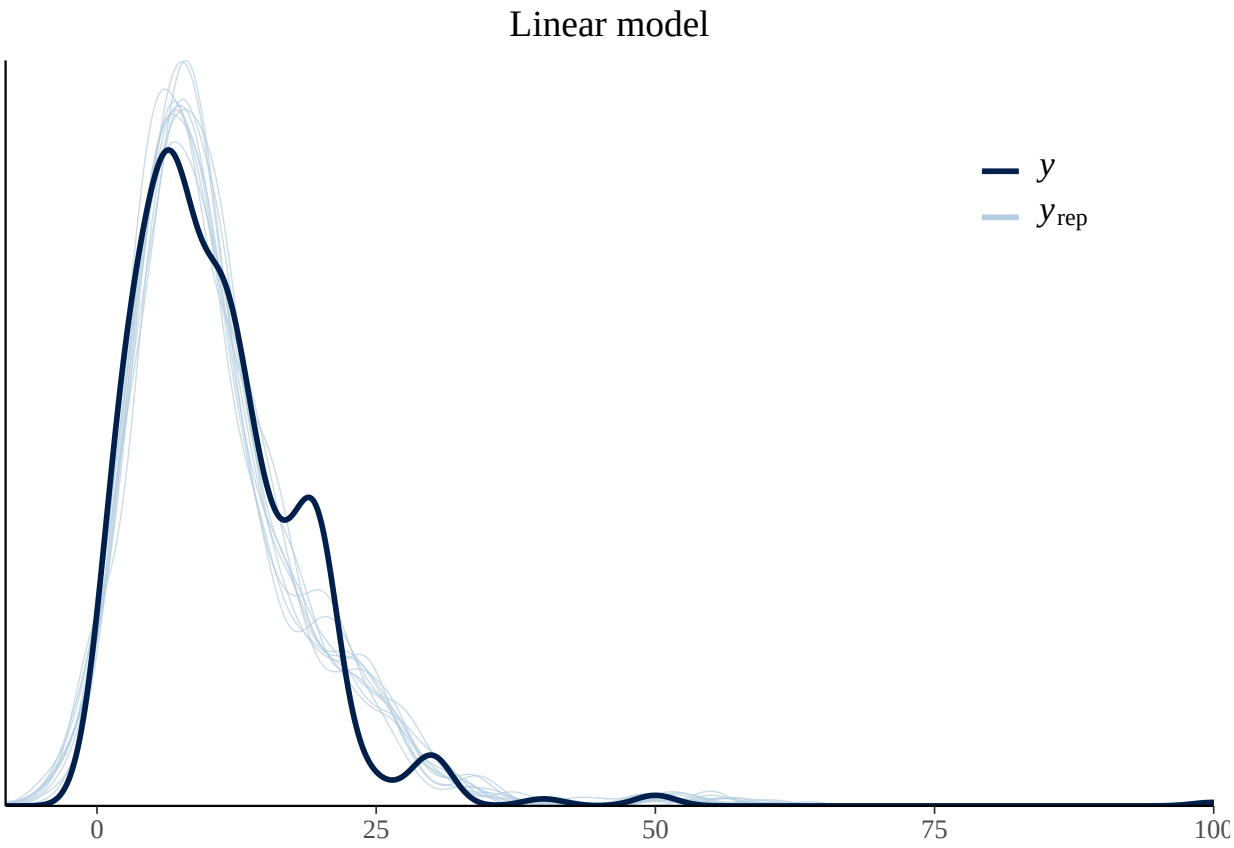
```
(linear_standard <-
  brm(
    formula,
    data = df[df$condition == 'standard', ],
    save_all_pars = T,
    sample_prior = 'yes',
    warmup = 2000,
    cores = 4,
    prior = prior_settings,
    iter = 7000,
    file = 'study2_linear_standard_x'))
```

```
## Warning: There were 9 divergent transitions after warmup. Increasing adapt_delta
## above 0.8 may help. See http://mc-stan.org/misc/warnings.html#divergent-
## transitions-after-warmup
```

```
##  Family: gaussian
##   Links: mu = identity; sigma = identity
## Formula: brightness_rating_stevens ~ 0 + Intercept + cd + (1 + cd | File)
##    Data: df[df$condition == "standard", ] (Number of observations: 640)
## Samples: 4 chains, each with iter = 7000; warmup = 2000; thin = 1;
##          total post-warmup samples = 20000
##
## Group-Level Effects:
## ~File (Number of levels: 16)
##                   Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
## sd(Intercept)         0.01      0.00     0.01     0.02 1.00     9842    11547
## sd(cd)                0.10      0.02     0.07     0.14 1.00     8200    10097
## cor(Intercept,cd)    -0.91      0.09    -1.00    -0.66 1.00     8033    10322
##
## Population-Level Effects:
##           Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
## Intercept     0.05      0.00     0.05     0.06 1.00     7984     9466
## cd            0.20      0.03     0.15     0.25 1.00     7436     8469
##
## Family Specific Parameters:
##       Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
## sigma     0.04      0.00     0.03     0.04 1.00    26295    14071
##
## Samples were drawn using sampling(NUTS). For each parameter, Bulk_ESS
## and Tail_ESS are effective sample size measures, and Rhat is the potential
## scale reduction factor on split chains (at convergence, Rhat = 1).
```

```
(
  lin_standard_plot = pp_check(linear_standard) + labs(title = 'Linear model') +
    scale_x_continuous(labels = c(0, 25, 50, 75, 100)) +
    theme(
      plot.title = element_text(hjust = 0.5),
      legend.position = c(0.85, 0.85),
    )
)
```

```
## Using 10 posterior samples for ppc type 'dens_overlay' by default.
```




```
brms::loo(linear_standard)
```

Model metrics (LOOCV, WAIC, marginal likelihood)

```
## Warning: Found 2 observations with a pareto_k > 0.7 in model 'linear_standard'.
## It is recommended to set 'moment_match = TRUE' in order to perform moment
## matching for problematic observations.

##
## Computed from 20000 by 640 log-likelihood matrix
##
##          Estimate    SE
## elpd_loo   1189.7 115.2
## p_loo        69.8  40.1
## looic     -2379.4 230.5
## ------
## Monte Carlo SE of elpd_loo is NA.
##
## Pareto k diagnostic values:
##                          Count Pct.    Min. n_eff
## (-Inf, 0.5]   (good)     636   99.4%   2172
##  (0.5, 0.7]   (ok)         2    0.3%   1016
##    (0.7, 1]   (bad)        1    0.2%   32
```

```
##    (1, Inf)   (very bad)   1    0.2%   1
## See help('pareto-k-diagnostic') for details.
```

```
brms::waic(linear_standard)
```

```
## Warning:
## 13 (2.0%) p_waic estimates greater than 0.4. We recommend trying loo instead.

##
## Computed from 20000 by 640 log-likelihood matrix
##
##           Estimate    SE
## elpd_waic   1179.1 126.3
## p_waic        80.3  51.3
## waic       -2358.3 252.5
##
## 13 (2.0%) p_waic estimates greater than 0.4. We recommend trying loo instead.
```

```
brms::bridge_sampler(linear_standard)
```

```
## Iteration: 1
## Iteration: 2
## Iteration: 3
## Iteration: 4
## Iteration: 5
## Iteration: 6

## Bridge sampling estimate of the log marginal likelihood: 1169.19
## Estimate obtained in 6 iteration(s) via method "normal".
```

Model comparison (Bayes factors) Compute logBF in favor of power law. Takes a few minutes to run, hence commented out.

```
bfs_standard = vector('numeric', length = 5)
for (i in 1:5) {
  bfs_standard[i] = log(1 / bayes_factor(linear_standard, power_law_standard)$bf)
}
```

```
## Iteration: 1
## Iteration: 2
## Iteration: 3
## Iteration: 4
## Iteration: 5
## Iteration: 6
## Iteration: 7
## Iteration: 1
## Iteration: 2
## Iteration: 3
## Iteration: 4
## Iteration: 5
## Iteration: 6
```

```
## Iteration: 1
## Iteration: 2
## Iteration: 3
## Iteration: 4
## Iteration: 5
## Iteration: 6
## Iteration: 7
## Iteration: 1
## Iteration: 2
## Iteration: 3
## Iteration: 4
## Iteration: 5
## Iteration: 6
## Iteration: 1
## Iteration: 2
## Iteration: 3
## Iteration: 4
## Iteration: 5
## Iteration: 6
## Iteration: 7
## Iteration: 1
## Iteration: 2
## Iteration: 3
## Iteration: 4
## Iteration: 5
## Iteration: 6
## Iteration: 1
## Iteration: 2
## Iteration: 3
## Iteration: 4
## Iteration: 5
## Iteration: 6
## Iteration: 1
## Iteration: 2
## Iteration: 3
## Iteration: 4
## Iteration: 5
## Iteration: 1
## Iteration: 2
## Iteration: 3
## Iteration: 4
## Iteration: 5
## Iteration: 6
## Iteration: 7
## Iteration: 8
## Iteration: 1
## Iteration: 2
## Iteration: 3
## Iteration: 4
## Iteration: 5
```

```
print(paste0('Mean logBF:', mean(bfs_standard), 'Std logBF:', sd(bfs_standard)))
```

```
## [1] "Mean logBF:54.6418551193861Std logBF:0.0427925389990544"
```

Bayesian mixed-effects models (unidirectional condition)

```
prior_settings <- c(set_prior('normal(0, 5)', nlpar = "a"),
                    set_prior('normal(1.0, 0.5)', nlpar = "b", lb = 0)
                    )
```

```
(formula = bf(brightness_rating_stevens ~ a * cd ^ b, a ~ (1 |File), b ~ (1 | File),  nl = TRUE))
```

Fit power law

```
## brightness_rating_stevens ~ a * cd^b
## a ~ (1 | File)
## b ~ (1 | File)
```

```
# make_stancode(formula, data = df[df$condition == 'unidirectional', ], prior = prior_settings)
```

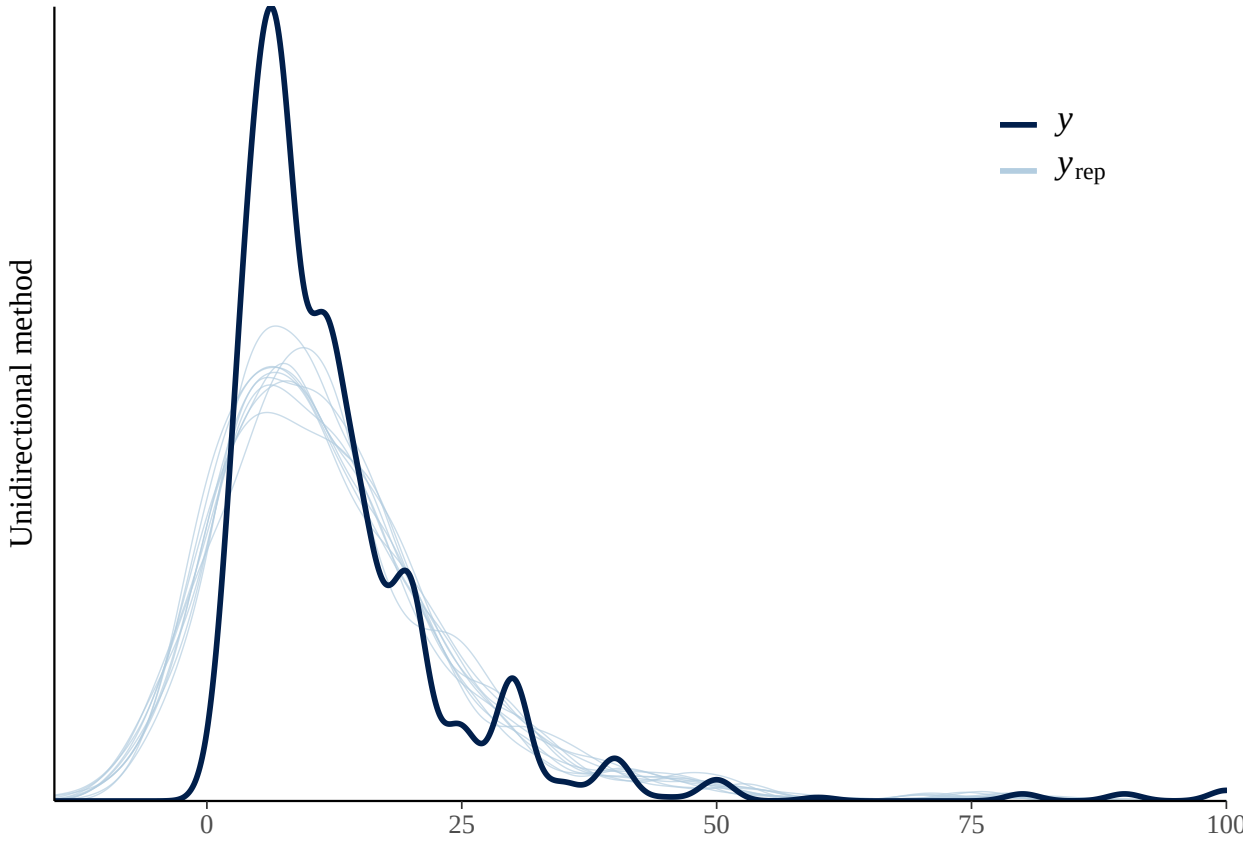
```
(power_law_unidirectional <-
  brm(formula,
    data = df[df$condition == 'unidirectional', ],
    save_all_pars = T,
    sample_prior = 'yes',
    warmup = 2000,
    iter = 7000,
    cores = 4,
    control = list(adapt_delta = 0.96, max_treedepth = 20),
    prior = prior_settings,
    file = 'study2_power_unidirectional_x'))
```

```
##  Family: gaussian
##   Links: mu = identity; sigma = identity
## Formula: brightness_rating_stevens ~ a * cd^b
##          a ~ (1 | File)
##          b ~ (1 | File)
##    Data: df[df$condition == "unidirectional", ] (Number of observations: 719)
## Samples: 4 chains, each with iter = 7000; warmup = 2000; thin = 1;
##          total post-warmup samples = 20000
##
## Group-Level Effects:
## ~File (Number of levels: 18)
##                 Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
## sd(a_Intercept)     0.17      0.03     0.12     0.24 1.00     3967     7473
## sd(b_Intercept)     0.30      0.06     0.21     0.45 1.00     4096     7561
##
## Population-Level Effects:
##             Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
## a_Intercept     0.32      0.04     0.24     0.40 1.00     2575     4882
## b_Intercept     0.55      0.07     0.41     0.70 1.00     3468     5553
##
## Family Specific Parameters:
```

```
##       Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
## sigma     0.05      0.00     0.05     0.05 1.00    19667    15132
##
## Samples were drawn using sampling(NUTS). For each parameter, Bulk_ESS
## and Tail_ESS are effective sample size measures, and Rhat is the potential
## scale reduction factor on split chains (at convergence, Rhat = 1).
```

```
(
  pow_unidirectional_plot = pp_check(power_law_unidirectional) +
    labs(y = 'Unidirectional method') +
    scale_x_continuous(labels = c(0, 25, 50, 75, 100), breaks=c(0,0.25,0.5,0.75,1)) +
    theme(legend.position = c(0.85, 0.85))
)
```

```
## Using 10 posterior samples for ppc type 'dens_overlay' by default.
```




```
brms::loo(power_law_unidirectional)
```

Model metrics (LOOCV, WAIC, marginal likelihood)

```
## Warning: Found 7 observations with a pareto_k > 0.7 in model
## 'power_law_unidirectional'. It is recommended to set 'moment_match = TRUE' in
## order to perform moment matching for problematic observations.
```

```
##
## Computed from 20000 by 719 log-likelihood matrix
##
##          Estimate    SE
## elpd_loo   1074.0 104.1
## p_loo       119.3  37.4
## looic     -2148.0 208.2
## ------
## Monte Carlo SE of elpd_loo is NA.
##
## Pareto k diagnostic values:
##                          Count Pct.    Min. n_eff
## (-Inf, 0.5]   (good)     709   98.6%   1665
##  (0.5, 0.7]   (ok)         3    0.4%   305
##    (0.7, 1]   (bad)        2    0.3%   88
##    (1, Inf)   (very bad)   5    0.7%   3
## See help('pareto-k-diagnostic') for details.
```

```
brms::waic(power_law_unidirectional)
```

```
## Warning:
## 23 (3.2%) p_waic estimates greater than 0.4. We recommend trying loo instead.

##
## Computed from 20000 by 719 log-likelihood matrix
##
##           Estimate    SE
## elpd_waic   1069.0 106.7
## p_waic       124.2  40.1
## waic       -2138.0 213.4
##
## 23 (3.2%) p_waic estimates greater than 0.4. We recommend trying loo instead.
```

```
brms::bridge_sampler(power_law_unidirectional)
```

```
## Iteration: 1
## Iteration: 2
## Iteration: 3
## Iteration: 4
## Iteration: 5
## Iteration: 6
## Iteration: 7

## Bridge sampling estimate of the log marginal likelihood: 1052.984
## Estimate obtained in 7 iteration(s) via method "normal".
```

```
prior_settings <-
  c(
    set_prior('normal(0,1)', class = 'b', coef = 'cd'),
    set_prior('normal(0,5)', class = 'b', coef = 'Intercept')
  )
```

```
(formula = brightness_rating_stevens ~ 0 + Intercept + cd + (1 + cd | File))
```

Fit linear model

```
## brightness_rating_stevens ~ 0 + Intercept + cd + (1 + cd | File)
```

```
# make_stancode(formula, data = df[df$condition == 'unidirectional', ], prior = prior_settings)
```

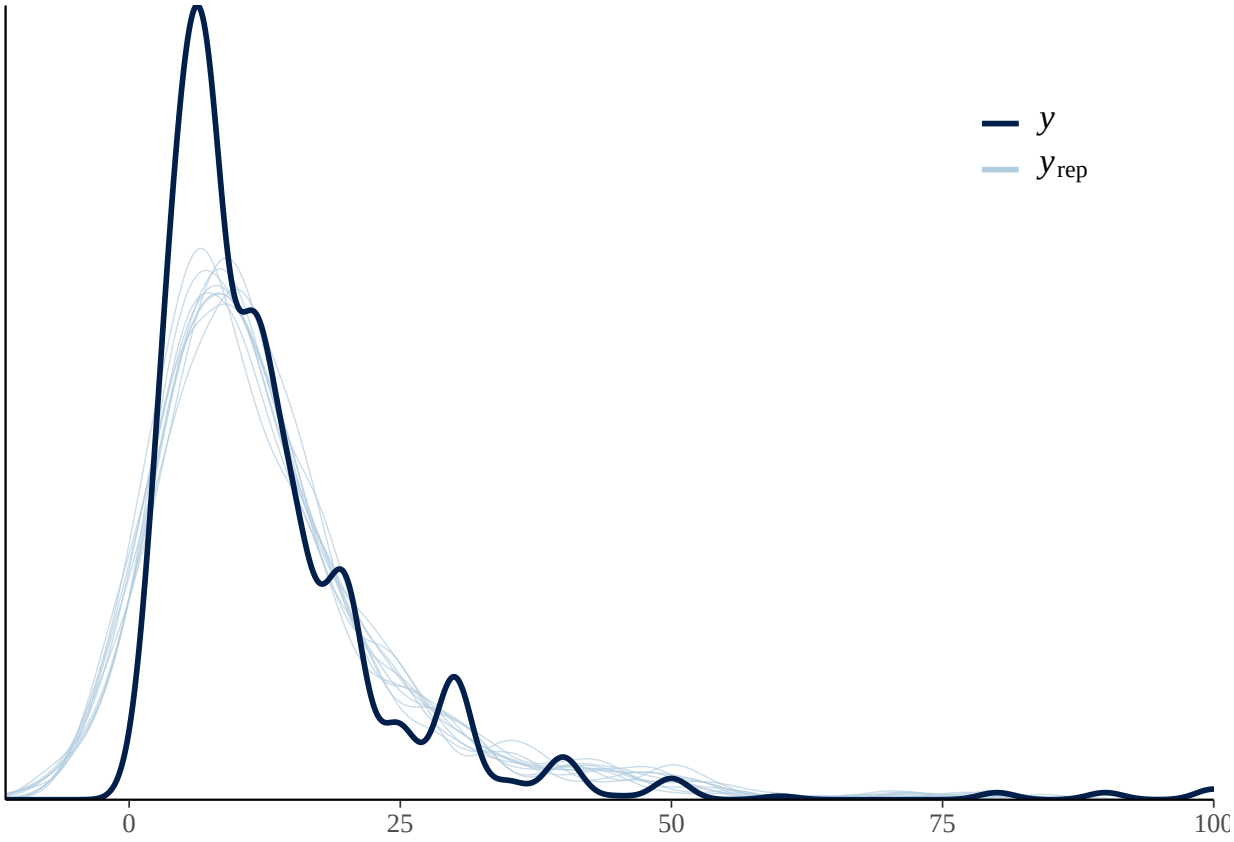
```
(linear_unidirectional <-
  brm(
    formula,
    data = df[df$condition == 'unidirectional', ],
    save_all_pars = T,
    sample_prior = 'yes',
    warmup = 2000,
    cores = 4,
    prior = prior_settings,
    iter = 7000,
    file = 'study2_linear_unidirectional_x'))
```

```
## Warning: There were 1 divergent transitions after warmup. Increasing adapt_delta
## above 0.8 may help. See http://mc-stan.org/misc/warnings.html#divergent-
## transitions-after-warmup
```

```
##  Family: gaussian
##   Links: mu = identity; sigma = identity
## Formula: brightness_rating_stevens ~ 0 + Intercept + cd + (1 + cd | File)
##    Data: df[df$condition == "unidirectional", ] (Number of observations: 719)
## Samples: 4 chains, each with iter = 7000; warmup = 2000; thin = 1;
##          total post-warmup samples = 20000
##
## Group-Level Effects:
## ~File (Number of levels: 18)
##                   Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
## sd(Intercept)         0.02      0.00     0.01     0.03 1.00     8165    11658
## sd(cd)                0.17      0.03     0.12     0.25 1.00     6883    10056
## cor(Intercept,cd)    -0.98      0.02    -1.00    -0.92 1.00    11423    12818
##
## Population-Level Effects:
##           Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
## Intercept     0.05      0.01     0.04     0.06 1.00     5306     8291
## cd            0.28      0.04     0.20     0.36 1.00     4713     8208
##
## Family Specific Parameters:
##       Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
## sigma     0.05      0.00     0.05     0.05 1.00    41479    13515
##
## Samples were drawn using sampling(NUTS). For each parameter, Bulk_ESS
## and Tail_ESS are effective sample size measures, and Rhat is the potential
## scale reduction factor on split chains (at convergence, Rhat = 1).
```

```
(
  lin_unidirectional_plot = pp_check(linear_unidirectional) +
    scale_x_continuous(labels = c(0, 25, 50, 75, 100), breaks=c(0,0.25,0.5,0.75,1)) +
    theme(legend.position = c(0.85, 0.85))
)
```

```
## Using 10 posterior samples for ppc type 'dens_overlay' by default.
```




```
brms::loo(linear_unidirectional)
```

Model metrics (LOOCV, WAIC, marginal likelihood)

```
## Warning: Found 6 observations with a pareto_k > 0.7 in model
## 'linear_unidirectional'. It is recommended to set 'moment_match = TRUE' in order
## to perform moment matching for problematic observations.

##
## Computed from 20000 by 719 log-likelihood matrix
##
##          Estimate    SE
## elpd_loo   1077.9  99.4
## p_loo        96.1  30.3
```

```
## looic     -2155.8 198.9
## ------
## Monte Carlo SE of elpd_loo is NA.
##
## Pareto k diagnostic values:
##                          Count Pct.    Min. n_eff
## (-Inf, 0.5]   (good)     708   98.5%   2491
##  (0.5, 0.7]   (ok)         5    0.7%   561
##    (0.7, 1]   (bad)        1    0.1%   36
##    (1, Inf)   (very bad)   5    0.7%   5
## See help('pareto-k-diagnostic') for details.
```

```
brms::waic(linear_unidirectional)
```

```
## Warning:
## 18 (2.5%) p_waic estimates greater than 0.4. We recommend trying loo instead.

##
## Computed from 20000 by 719 log-likelihood matrix
##
##           Estimate    SE
## elpd_waic   1075.6 101.0
## p_waic        98.4  32.0
## waic       -2151.3 202.0
##
## 18 (2.5%) p_waic estimates greater than 0.4. We recommend trying loo instead.
```

```
brms::bridge_sampler(linear_unidirectional)
```

```
## Iteration: 1
## Iteration: 2
## Iteration: 3
## Iteration: 4
## Iteration: 5

## Bridge sampling estimate of the log marginal likelihood: 1066.34
## Estimate obtained in 5 iteration(s) via method "normal".
```

Model comparison (Bayes factors) Compute logBF in favor of power law. Takes a few minutes to run, hence commented out.

```
bfs_unidirectional = vector('numeric', length = 5)
for (i in 1:5) {
  bfs_unidirectional[i] = log(1 / bayes_factor(linear_unidirectional, power_law_unidirectional)$bf)
}
```

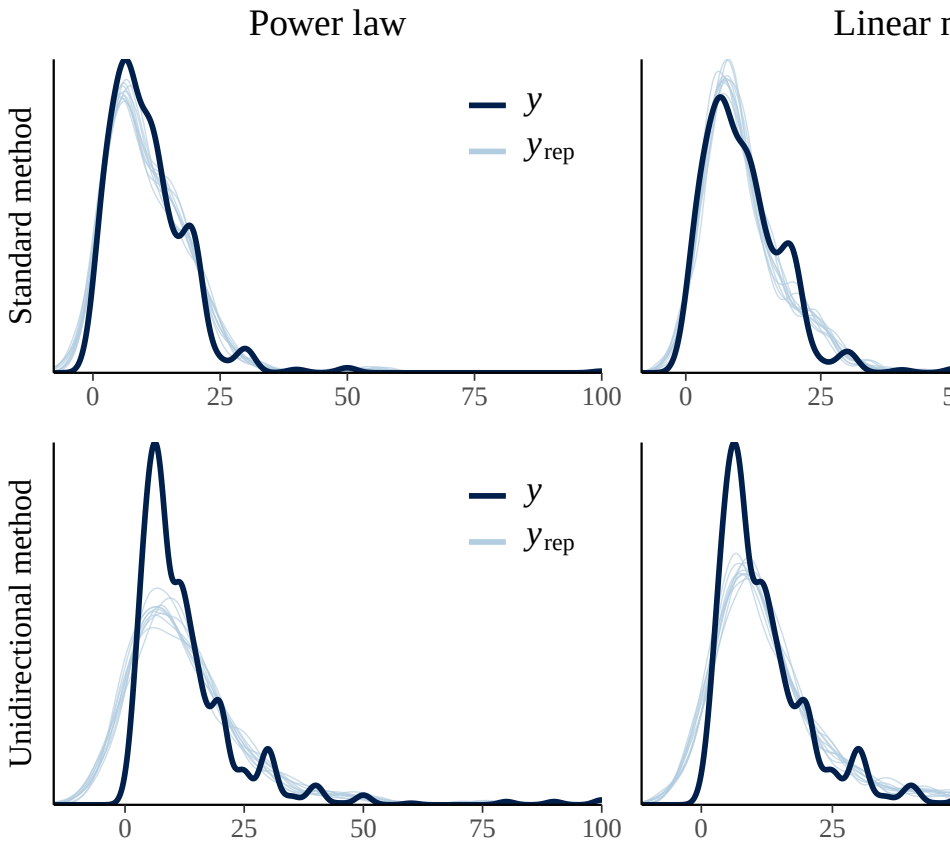
```
## Iteration: 1
## Iteration: 2
## Iteration: 3
## Iteration: 4
## Iteration: 5
```

```
## Iteration: 6
## Iteration: 1
## Iteration: 2
## Iteration: 3
## Iteration: 4
## Iteration: 5
## Iteration: 6
## Iteration: 1
## Iteration: 2
## Iteration: 3
## Iteration: 4
## Iteration: 5
## Iteration: 6
## Iteration: 1
## Iteration: 2
## Iteration: 3
## Iteration: 4
## Iteration: 1
## Iteration: 2
## Iteration: 3
## Iteration: 4
## Iteration: 5
## Iteration: 6
## Iteration: 7
## Iteration: 1
## Iteration: 2
## Iteration: 3
## Iteration: 4
## Iteration: 5
## Iteration: 1
## Iteration: 2
## Iteration: 3
## Iteration: 4
## Iteration: 5
## Iteration: 6
## Iteration: 7
## Iteration: 8
## Iteration: 1
## Iteration: 2
## Iteration: 3
## Iteration: 4
## Iteration: 5
## Iteration: 1
## Iteration: 2
## Iteration: 3
## Iteration: 4
## Iteration: 5
## Iteration: 6
## Iteration: 1
## Iteration: 2
## Iteration: 3
## Iteration: 4
## Iteration: 5
## Iteration: 6
```

```
print(paste0('Mean logBF:', mean(bfs_unidirectional), 'Std logBF:', sd(bfs_unidirectional)))
```

```
## [1] "Mean logBF:-13.3761701085566Std logBF:0.0298489725894633"
```

```
pp_grid = (
  grid.arrange(
    pow_standard_plot,
    lin_standard_plot,
    pow_unidirectional_plot,
    lin_unidirectional_plot
  )
)
```



Posterior predictive plot (Figure 3)

```
ggsave(
  paste0("final_plots/study2_pp_plot.png"),
  plot = pp_grid,
  dpi = 600,
  height = 6,
  width = 7,
  units = "in"
)
```

Bayesian mixed-effects models (aggregated data)

```
df_agg = df %>% group_by(cd,condition) %>% summarise(median_rating = median(brightness_rating_stevens))
```

Fit power law (standard condition)

```
## `summarise()` regrouping output by 'cd' (override with `.groups` argument)
```

```
prior_settings <- c(set_prior('normal(0, 5)', nlpar = "a"),
            set_prior('normal(1.0, 0.5)', nlpar = "b", lb = 0))
```

```
(formula = bf(median_rating ~ a * cd ^ b , a ~ 1, b ~ 1 , nl = TRUE))
```

```
## median_rating ~ a * cd^b
## a ~ 1
## b ~ 1
```

```
# make_stancode(formula, data = df_agg[df_agg$condition == 'standard', ], prior = prior_settings)
```

```
(
  power_law_standard_agg <-
    brm(
      formula,
      data = df_agg[df_agg$condition == 'standard',],
      save_all_pars = T,
      sample_prior = 'yes',
      warmup = 2000,
      iter = 7000,
      cores = 4,
      control = list(adapt_delta = 0.96, max_treedepth = 20),
      prior = prior_settings,
      file = 'study2_power_standard_agg_x'
    )
)
```

```
##  Family: gaussian
##   Links: mu = identity; sigma = identity
## Formula: median_rating ~ a * cd^b
##          a ~ 1
##          b ~ 1
##    Data: df_agg[df_agg$condition == "standard", ] (Number of observations: 8)
## Samples: 4 chains, each with iter = 7000; warmup = 2000; thin = 1;
##          total post-warmup samples = 20000
##
## Population-Level Effects:
##             Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
## a_Intercept     0.20      0.01     0.18     0.23 1.00     6778     5687
## b_Intercept     0.38      0.04     0.31     0.46 1.00     6929     6449
##
```

```
## Family Specific Parameters:
##       Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
## sigma     0.01      0.01     0.01     0.03 1.00     5258     5513
##
## Samples were drawn using sampling(NUTS). For each parameter, Bulk_ESS
## and Tail_ESS are effective sample size measures, and Rhat is the potential
## scale reduction factor on split chains (at convergence, Rhat = 1).
```

```
prior_settings <- c(set_prior('normal(0, 5)', nlpar = "a"),
            set_prior('normal(1.0, 0.5)', nlpar = "b", lb = 0))
```

```
(formula = bf(median_rating ~ a * cd ^ b , a ~ 1, b ~ 1 , nl = TRUE))
```

Fit power law on aggregated data (unidirectional condition)

```
## median_rating ~ a * cd^b
## a ~ 1
## b ~ 1
```

```
# make_stancode(formula, data = df_agg[df_agg$condition == 'unidirectional', ], prior = prior_settings)
```

```
(
  power_law_unidirectional_agg <-
    brm(
      formula,
      data = df_agg[df_agg$condition == 'unidirectional',],
      save_all_pars = T,
      sample_prior = 'yes',
      warmup = 2000,
      iter = 7000,
      cores = 4,
      control = list(adapt_delta = 0.96, max_treedepth = 20),
      prior = prior_settings,
      file = 'study2_power_unidirectional_agg_x'
    )
)
```

```
##  Family: gaussian
##   Links: mu = identity; sigma = identity
## Formula: median_rating ~ a * cd^b
##          a ~ 1
##          b ~ 1
##    Data: df_agg[df_agg$condition == "unidirectional", ] (Number of observations: 8)
## Samples: 4 chains, each with iter = 7000; warmup = 2000; thin = 1;
##          total post-warmup samples = 20000
##
## Population-Level Effects:
##             Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
```

```
## a_Intercept     0.28      0.02     0.25     0.33 1.00     6197     5744
## b_Intercept     0.52      0.09     0.39     0.73 1.00     6073     5480
##
## Family Specific Parameters:
##       Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
## sigma     0.02      0.01     0.01     0.05 1.00     5536     5761
##
## Samples were drawn using sampling(NUTS). For each parameter, Bulk_ESS
## and Tail_ESS are effective sample size measures, and Rhat is the potential
## scale reduction factor on split chains (at convergence, Rhat = 1).
```

Supplemental analysis

```
df_supp = read.csv('https://raw.githubusercontent.com/mertensu/thinking-in-ratios/master/data_total_sup
```

```
df_supp[df_supp$brightness_rating == 0, 'brightness_rating'] = 0.001

df_supp = df_supp %>% mutate(brightness_dichotom_correct = ifelse(cd < 10, 0, 1))


df_supp$correct_response = NA
df_supp[,'correct_response'] = df_supp['brightness_dichotom_correct'] == df_supp[, 'brightness_dichotom

df_supp$cd_factor = factor(df_supp$cd, levels = c(1, 1.8, 3.2, 5.7, 17.9, 32.0, 57.2, 100.0))


# use same column name as in Study 2
df_supp$brightness_rating_stevens = df_supp$brightness_rating
df_supp$log_brightness_rating = log(df_supp$brightness_rating_stevens)

df_supp$brightness_rating_stevens = df_supp$brightness_rating_stevens/100
df_supp$cd = df_supp$cd/100
```

```
psych::describe(df_supp$age)
```

```
##    vars   n  mean   sd median trimmed  mad min max range skew kurtosis   se
## X1    1 760 21.37 2.37     21   21.14 1.48  18  27     9 0.82    -0.24 0.09
```

```
df_supp %>% distinct(File, .keep_all = T) %>% group_by(gender) %>% summarise(
  N = n(),
  Min =
    min(age),
  Max =
    max(age),
  Mean =
    mean(age),
  Sd =
    sd(age)
)
```

```
## `summarise()` ungrouping output (override with `.groups` argument)
```

```
## # A tibble: 2 x 6
##   gender     N   Min   Max  Mean    Sd
##   <chr>  <int> <int> <int> <dbl> <dbl>
## 1 m          4    18    25  21.8  2.99
## 2 w         15    19    27  21.3  2.37
```

```
df_supp %>% distinct(File, .keep_all = T) %>% count(student)
```

```
##   student  n
## 1       1 19
```

```
df_supp %>% distinct(File, .keep_all = T) %>% filter(student == 1) %>% count(psycho)
```

```
##   psycho  n
## 1      0 12
## 2      1  7
```

```
df_supp %>% distinct(File, .keep_all = T) %>% count(condition)
```

```
##         condition  n
## 1 binary_standard 19
```

```
df_s2 = df
df_s2$File = paste0(df_s2$File,'study2')
df_tot = rbind(df_supp,df_s2)
df_tot = df_tot %>% filter(condition != 'unidirectional')
```

```
(
  fit = aov_ez(
    dv = "log_brightness_rating",
    within = 'cd_factor',
    between = "condition",
    id = "File",
    data = df_tot
  )
)
```

```
## Converting to factor: condition
```

```
## Warning: More than one observation per cell, aggregating the data using mean
## (i.e, fun_aggregate = mean)!
```

```
## Contrasts set to contr.sum for the following variables: condition
```

```
## Anova Table (Type 3 tests)
##
## Response: log_brightness_rating
##                Effect          df  MSE          F   ges p.value
## 1           condition       1, 33 0.40       0.13 <.001    .716
## 2           cd_factor 1.60, 52.89 0.90 111.11 ***  .724   <.001
```

```
## 3 condition:cd_factor 1.60, 52.89 0.90       0.24  .006    .739
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '+' 0.1 ' ' 1
##
## Sphericity correction method: GG
```

```
df_tot$File = factor(df_tot$File)
df_tot$condition = factor(df_tot$condition)
bfs = anovaBF(log_brightness_rating ~ cd_factor * condition + File,
              whichRandom = 'File',whichModels = 'top',data=df_tot)

# BF cd_factor
bf_1 = lmBF(log_brightness_rating ~ condition + File, whichRandom = 'File',data=df_tot)
bf_2 = lmBF(log_brightness_rating ~ cd_factor + condition + File, whichRandom = 'File',data=df_tot)
(bf_cd_factor = bf_2 / bf_1)
```

```
## Bayes factor analysis
## --------------
## [1] cd_factor + condition + File : 1.74591e+279 ±1.44%
##
## Against denominator:
##   log_brightness_rating ~ condition + File
## ---
## Bayes factor type: BFlinearModel, JZS
```

```
print(paste0('logBF ', bf_cd_factor@bayesFactor$bf))
```

```
## [1] "logBF 642.978517059165"
```

```
# BF condition
bf_1 = lmBF(log_brightness_rating ~ cd_factor + File, whichRandom = 'File',data=df_tot)
bf_2 = lmBF(log_brightness_rating ~ cd_factor + condition + File, whichRandom = 'File',data=df_tot)
(bf_condition = bf_2 / bf_1)
```

```
## Bayes factor analysis
## --------------
## [1] cd_factor + condition + File : 0.1689223 ±3.18%
##
## Against denominator:
##   log_brightness_rating ~ cd_factor + File
## ---
## Bayes factor type: BFlinearModel, JZS
```

```
print(paste0('logBF ', bf_condition@bayesFactor$bf))
```

```
## [1] "logBF -1.77831633339497"
```

```
# BF interaction
bf_1 = lmBF(log_brightness_rating ~ cd_factor + condition + File, whichRandom = 'File',data=df_tot)
bf_2 = lmBF(log_brightness_rating ~ cd_factor * condition + File, whichRandom = 'File',data=df_tot)
(bf_int = bf_2 / bf_1)
```

```
## Bayes factor analysis
## --------------
## [1] cd_factor * condition + File : 0.001997562 ±3.99%
##
## Against denominator:
##   log_brightness_rating ~ cd_factor + condition + File
## ---
## Bayes factor type: BFlinearModel, JZS
```

```
print(paste0('logBF ', bf_int@bayesFactor$bf))
```

```
## [1] "logBF -6.21582797405972"
```

```
prior_settings <-
  c(set_prior('normal(0, 5)', nlpar = "a"),
    set_prior('normal(1.0, 0.5)', nlpar = "b", lb = 0))
```

```
(formula = bf(brightness_rating_stevens ~ a * cd ^ b , a ~ (1 |File), b ~ (1 | File), nl = TRUE))
```

Fit power law

```
## brightness_rating_stevens ~ a * cd^b
## a ~ (1 | File)
## b ~ (1 | File)
```

```
# make_stancode(formula, data = df_supp, prior = prior_settings)
```

```
(
  power_law_supp <-
    brm(
      formula,
      data = df_supp,
      save_all_pars = T,
      sample_prior = 'yes',
      warmup = 2000,
      iter = 7000,
      cores = 4,
      control = list(adapt_delta = 0.96, max_treedepth = 20),
      prior = prior_settings,
      file = 'power_binary_standard'
    )
)
```

```
##  Family: gaussian
##   Links: mu = identity; sigma = identity
## Formula: brightness_rating_stevens ~ a * cd^b
##          a ~ (1 | File)
```

```
##          b ~ (1 | File)
##    Data: df_supp (Number of observations: 760)
## Samples: 4 chains, each with iter = 7000; warmup = 2000; thin = 1;
##          total post-warmup samples = 20000
##
## Group-Level Effects:
## ~File (Number of levels: 19)
##                 Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
## sd(a_Intercept)     0.09      0.02     0.06     0.13 1.00     3549     6546
## sd(b_Intercept)     0.18      0.03     0.12     0.26 1.00     4340     7621
##
## Population-Level Effects:
##             Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
## a_Intercept     0.24      0.02     0.19     0.28 1.00     2495     4228
## b_Intercept     0.42      0.04     0.34     0.51 1.00     3223     5098
##
## Family Specific Parameters:
##       Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
## sigma     0.03      0.00     0.03     0.03 1.00    18911    14253
##
## Samples were drawn using sampling(NUTS). For each parameter, Bulk_ESS
## and Tail_ESS are effective sample size measures, and Rhat is the potential
## scale reduction factor on split chains (at convergence, Rhat = 1).
```