

Study 3

Contents

Import data	2
Preprocessing	2
Demographics	3
Analysis	4

This file reproduces the preprocessing and analysis steps of Study 3. The data are automatically imported from Github and necessary packages will be downloaded and installed if they are not yet available.

```
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         ggribes_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             Brodningnag_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_study3.csv')
```

Preprocessing

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

# compute correct binary choice
df = df %>% mutate(redness_correct = ifelse(saturation < 50, 0, 1))
```

```

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

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

df[, "redness_rating_comparable"] = ifelse((df$condition == "unidirectional") &
                                           (df$redness_dichotom == 0),
                                           100 / df$redness_rating,
                                           df$redness_rating
)

df$saturation_factor = factor(df$saturation, levels = c(15, 25, 35, 45, 55, 65, 75, 85))

df$log_redness_rating = log(df$redness_rating_comparable)
# divide saturation by 100 to enable more efficient sampling
df$saturation = df$saturation/100
df$redness_rating_comparable = df$redness_rating_comparable/100

```

Demographics

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

```
##      vars      n mean  sd median trimmed  mad min max range skew kurtosis   se
## X1      1 1424 20.81 1.72      21   20.71 1.48  18  25     7  0.5    -0.12 0.05
```

```

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: 3 x 6
##   gender      N    Min    Max  Mean    Sd
##   <chr> <int> <int> <int> <dbl> <dbl>
## 1 d         1     24     24   24    NA
## 2 m        15     19     25  21.3  1.49
## 3 w        20     18     24  20.3  1.72

```

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

```
##   student  n
## 1         0  1
## 2         1 35
```

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

```
##   psycho  n
## 1         0 22
## 2         1 13
```

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

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

Analysis

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

```
(
  fit = aov_ez(
    dv = "log_redness_rating",
    within = 'saturation_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_redness_rating
##           Effect            df  MSE          F ges p.value
## 1           condition            1, 34 1.67 23.31 *** .123  <.001
## 2 saturation_factor 1.60, 54.51 4.03 56.05 *** .567  <.001
## 3 condition:saturation_factor 1.60, 54.51 4.03 12.20 *** .222  <.001
## ---
## 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_redness_rating ~ saturation_factor * condition + File,
    whichRandom = 'File',
    whichModels = 'top',
    data = df
  )
)

```

bayesian fit

```

## Bayes factor top-down analysis
## -----
## When effect is omitted from condition + saturation_factor + condition:saturation_factor + File , BF :
## [1] Omit condition:saturation_factor : 6.672491e-52 ±3.76%
## [2] Omit saturation_factor          : 1.61072e-198 ±3.38%
## [3] Omit condition                  : 0.003861875 ±2.81%
##
## Against denominator:
##   log_redness_rating ~ condition + saturation_factor + condition:saturation_factor + File
## ---
## Bayes factor type: BFlinearModel, JZS

```

```

# BF saturation_factor
bf_1 = lmBF(log_redness_rating ~ condition + File,
            whichRandom = 'File',
            data = df)
bf_2 = lmBF(
  log_redness_rating ~ saturation_factor + condition + File,
  whichRandom = 'File',
  data = df
)
(bf_saturation_factor = bf_2 / bf_1)

```

```

## Bayes factor analysis
## -----
## [1] saturation_factor + condition + File : 2.565474e+171 ±2.06%
##
## Against denominator:
##   log_redness_rating ~ condition + File
## ---
## Bayes factor type: BFlinearModel, JZS

```

```

print(paste0('logBF ', bf_saturation_factor@bayesFactor$bf))

```

```

## [1] "logBF 394.684194035786"

```

```

# BF condition
bf_1 = lmBF(
  log_redness_rating ~ saturation_factor + File,
  whichRandom = 'File',
  data = df
)
bf_2 = lmBF(
  log_redness_rating ~ saturation_factor + condition + File,
  whichRandom = 'File',
  data = df
)
(bf_condition = bf_2 / bf_1)

## Bayes factor analysis
## -----
## [1] saturation_factor + condition + File : 209.3591 ±1.74%
##
## Against denominator:
##   log_redness_rating ~ saturation_factor + File
## ---
## Bayes factor type: BFlinearModel, JZS

```

```

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

```

```

## [1] "logBF 5.34405098567186"

```

```

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

```

```

## Bayes factor analysis
## -----
## [1] saturation_factor * condition + File : 1.56056e+51 ±9.96%
##
## Against denominator:
##   log_redness_rating ~ saturation_factor + condition + File
## ---
## Bayes factor type: BFlinearModel, JZS

```

```

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

```

```

## [1] "logBF 117.876884177071"

```

```

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

ggplot(grid, aes(
  x = saturation_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(0.1, 1, 10, 100),
    limits = c(0.07, 100)
  ) +
  geom_line(linetype = 'dashed') +
  labs(y = 'Redness judgement', x = 'redness saturation (in %)') +
  scale_color_manual(
    values = c("darkgrey", "black"),
    name = "method",
    labels = c("unidirectional", "standard")
  ) +
  scale_x_discrete(labels = substring(grid$saturation_factor, 2)) +
  theme_classic() +
  theme(
    legend.position = c(0.2, 0.85),
    legend.background = element_rect(color = "black")
  )

```

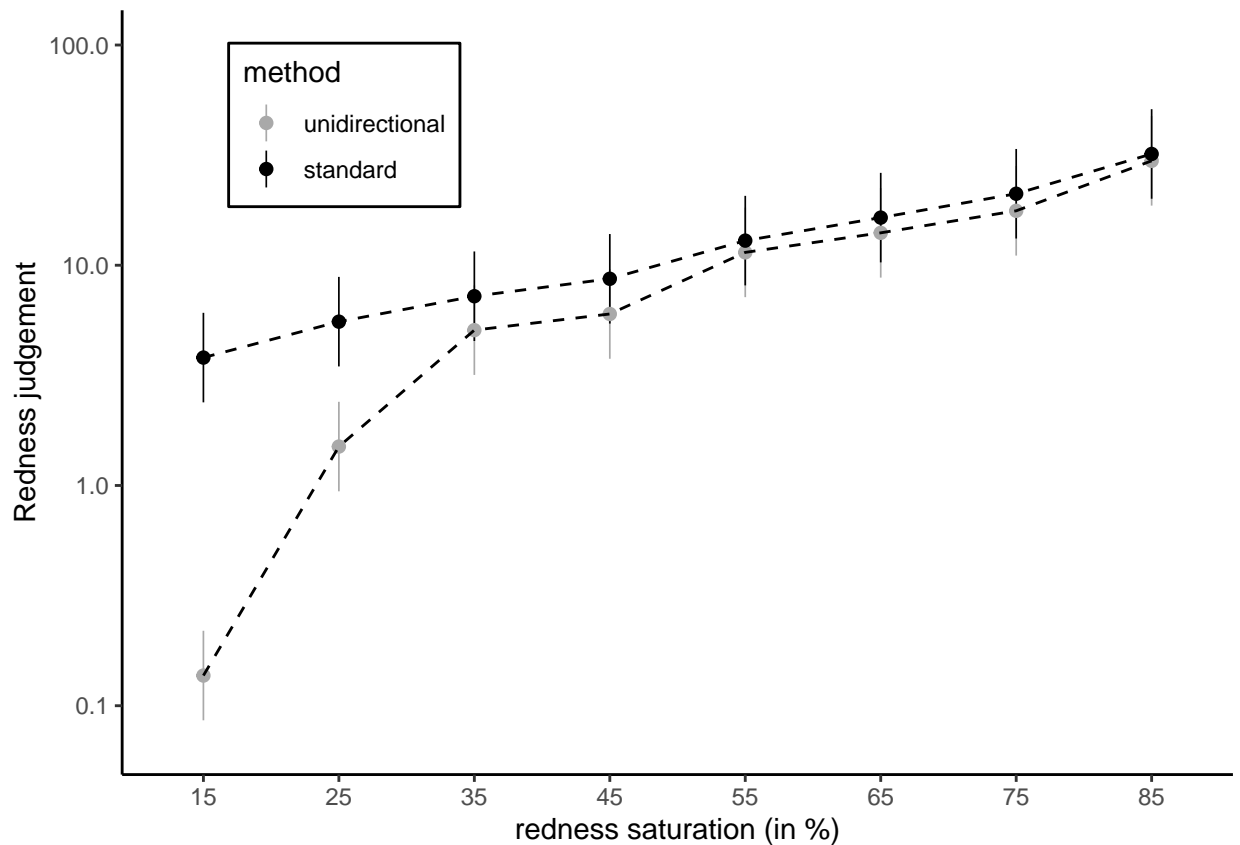


Figure 4

```
ggsave(
  paste0("final_plots/study3_figure3.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(
  redness_rating_comparable ~ a * saturation ^ b ,
  a ~ (1 | File),
  b ~ (1 | File),
  nl = TRUE
))
```

Fit power law


```
## redness_rating_comparable ~ a * saturation^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 = 15),
      prior = prior_settings,
      file = 'study3_power_standard_x'
    )
)
```

```
## Family: gaussian
## Links: mu = identity; sigma = identity
## Formula: redness_rating_comparable ~ a * saturation^b
## a ~ (1 | File)
## b ~ (1 | File)
## Data: df[df$condition == "standard", ] (Number of observations: 720)
## 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	1-95% CI	u-95% CI	Rhat	Bulk_ESS	Tail_ESS
sd(a_Intercept)	0.24	0.05	0.17	0.35	1.00	4279	7324
sd(b_Intercept)	1.01	0.21	0.69	1.49	1.00	5274	8608

```
##
## Population-Level Effects:
##
```

	Estimate	Est.Error	1-95% CI	u-95% CI	Rhat	Bulk_ESS	Tail_ESS
a_Intercept	0.45	0.06	0.33	0.56	1.00	3082	5810
b_Intercept	1.99	0.23	1.52	2.42	1.00	4071	7693

```
##
## Family Specific Parameters:
##
```

	Estimate	Est.Error	1-95% CI	u-95% CI	Rhat	Bulk_ESS	Tail_ESS
sigma	0.04	0.00	0.03	0.04	1.00	25386	14722

```
##
## 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') +

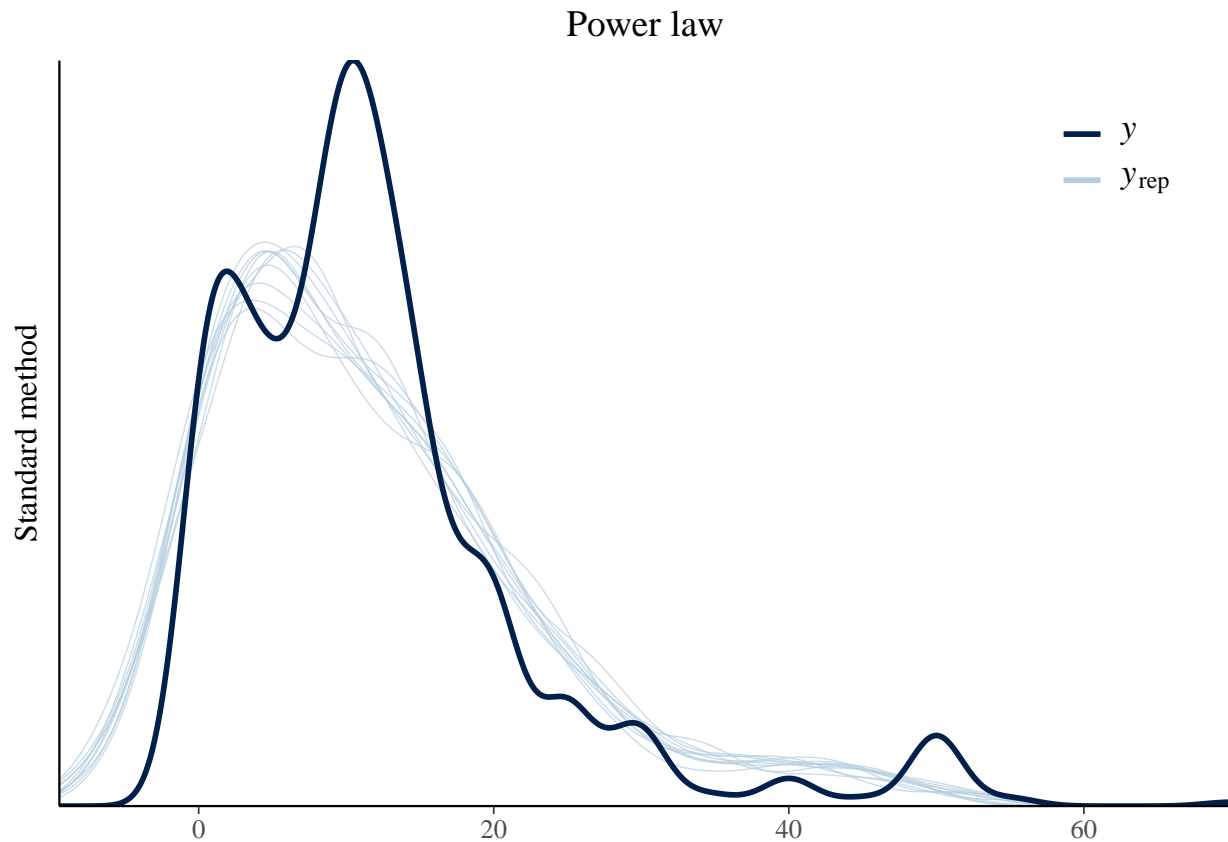
```

```

scale_x_continuous(labels=c(0,20,40,60), breaks=c(0,0.2,0.4,0.6)) +
theme(
  plot.title = element_text(hjust = 0.5),
  legend.position = c(0.9, 0.9)
)
)

```

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



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

Model metrics (LOOCV, WAIC, marginal likelihood)

```

## Warning: Found 3 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 720 log-likelihood matrix
##
##           Estimate   SE
## elpd_loo    1323.4 43.2

```

```
## p_loo          68.2 13.4
## looic         -2646.7 86.3
## -----
## Monte Carlo SE of elpd_loo is NA.
##
## Pareto k diagnostic values:
##               Count Pct.    Min. n_eff
## (-Inf, 0.5]   (good)   711  98.8%   2210
## (0.5, 0.7]    (ok)      6   0.8%    715
## (0.7, 1]      (bad)     2   0.3%     44
## (1, Inf)      (very bad) 1   0.1%     11
## See help('pareto-k-diagnostic') for details.
```

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

```
## Warning:
## 34 (4.7%) p_waic estimates greater than 0.4. We recommend trying loo instead.

##
## Computed from 20000 by 720 log-likelihood matrix
##
##           Estimate   SE
## elpd_waic  1325.1 42.6
## p_waic      66.5 12.8
## waic       -2650.2 85.2
##
## 34 (4.7%) 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

## Bridge sampling estimate of the log marginal likelihood: 1269.12
## Estimate obtained in 6 iteration(s) via method "normal".
```

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

```
(formula = redness_rating_comparable ~ 0 + Intercept + saturation + (1 + saturation |File))
```

Fit linear model

```

## redness_rating_comparable ~ 0 + Intercept + saturation + (1 +
##   saturation | 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 = 'study3_linear_standard_x'))

## Family: gaussian
## Links: mu = identity; sigma = identity
## Formula: redness_rating_comparable ~ 0 + Intercept + saturation + (1 + saturation | File)
## Data: df[df$condition == "standard", ] (Number of observations: 720)
## 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 1-95% CI u-95% CI Rhat Bulk_ESS
## sd(Intercept)      0.05      0.01   0.03   0.08 1.00     6634
## sd(saturation)     0.14      0.03   0.10   0.21 1.00     5951
## cor(Intercept,saturation) -0.98    0.02  -1.00  -0.92 1.00     7999
##
##           Tail_ESS
## sd(Intercept)    10211
## sd(saturation)     9412
## cor(Intercept,saturation) 9592
##
## Population-Level Effects:
##           Estimate Est.Error 1-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
## Intercept    -0.07      0.01  -0.09  -0.04 1.00     4745     7905
## saturation    0.37      0.04   0.30   0.44 1.00     4667     7078
##
## Family Specific Parameters:
##           Estimate Est.Error 1-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
## sigma      0.05      0.00   0.05   0.05 1.00    33352    15061
##
## 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(0,20,40,60), breaks=c(0,0.2,0.4,0.6)) +
  theme(
    plot.title = element_text(hjust = 0.5),

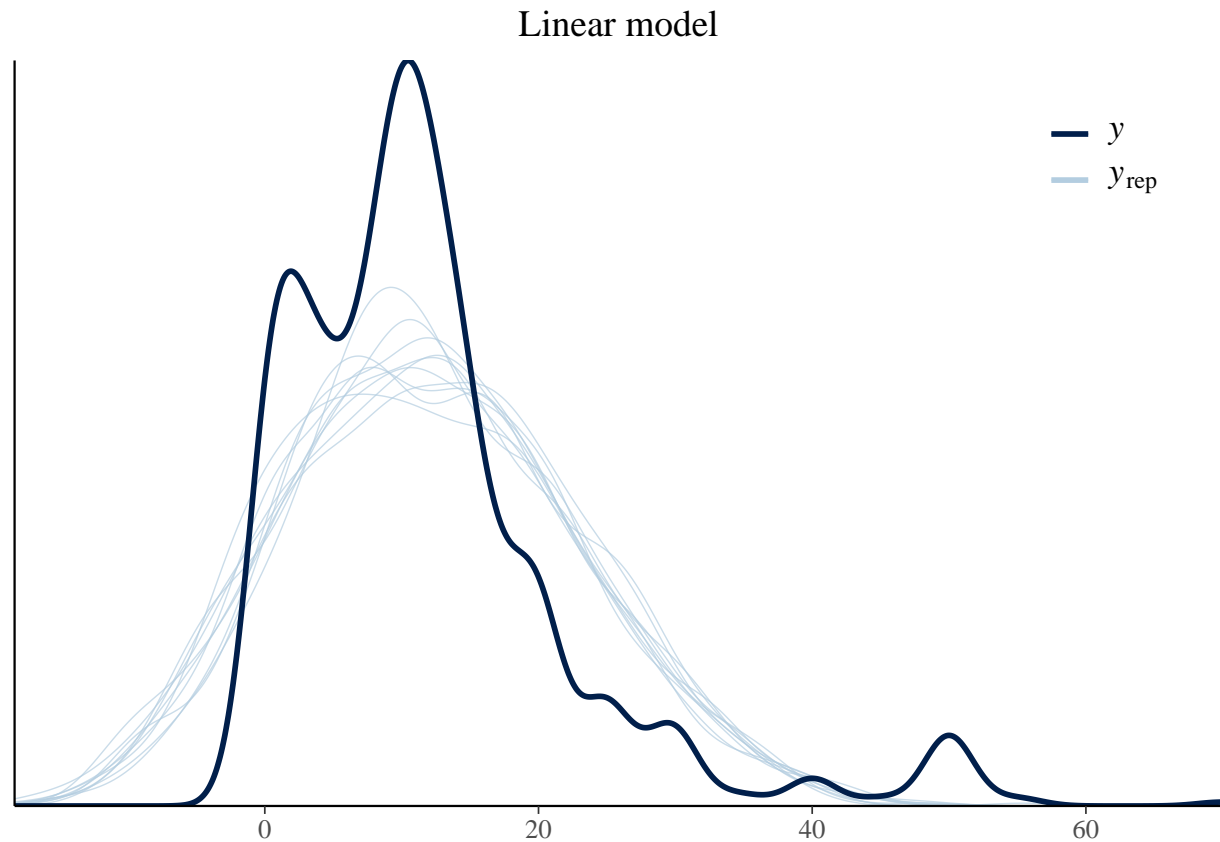
```

```

    legend.position = c(0.9, 0.9)
  )
)

```

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



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

Model metrics (LOOCV, WAIC, marginal likelihood)

```
## Warning: Found 1 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 720 log-likelihood matrix
##
##      Estimate   SE
## elpd_loo  1107.0 42.7
## p_loo     46.7  7.5
## looic     -2214.0 85.4
## -----
```

```
## Monte Carlo SE of elpd_loo is NA.
##
## Pareto k diagnostic values:
##           Count Pct.    Min. n_eff
## (-Inf, 0.5] (good)   719  99.9%   2469
## (0.5, 0.7]  (ok)      0    0.0%    <NA>
## (0.7, 1]    (bad)      1    0.1%   109
## (1, Inf)    (very bad) 0    0.0%    <NA>
## See help('pareto-k-diagnostic') for details.
```

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

```
## Warning:
## 24 (3.3%) p_waic estimates greater than 0.4. We recommend trying loo instead.
```

```
##
## Computed from 20000 by 720 log-likelihood matrix
##
##           Estimate   SE
## elpd_waic   1107.2 42.7
## p_waic       46.5  7.5
## waic        -2214.4 85.4
##
## 24 (3.3%) 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
## Iteration: 7
```

```
## Bridge sampling estimate of the log marginal likelihood: 1069.344
## Estimate obtained in 7 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: 7
## 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: 7
## 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: 1
```

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

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

```
## [1] "Mean logBF:199.738655575083Std logBF:0.0210697356413098"
```

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(
  redness_rating_comparable ~ a * saturation ^ b ,
  a ~ (1 | File),
  b ~ (1 | File),
  nl = TRUE
))
```

Fit power law

```
## redness_rating_comparable ~ a * saturation^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 = 15),
      prior = prior_settings,
      file = 'study3_power_unidirectional_x'
    )
)
```

```
## Family: gaussian
```



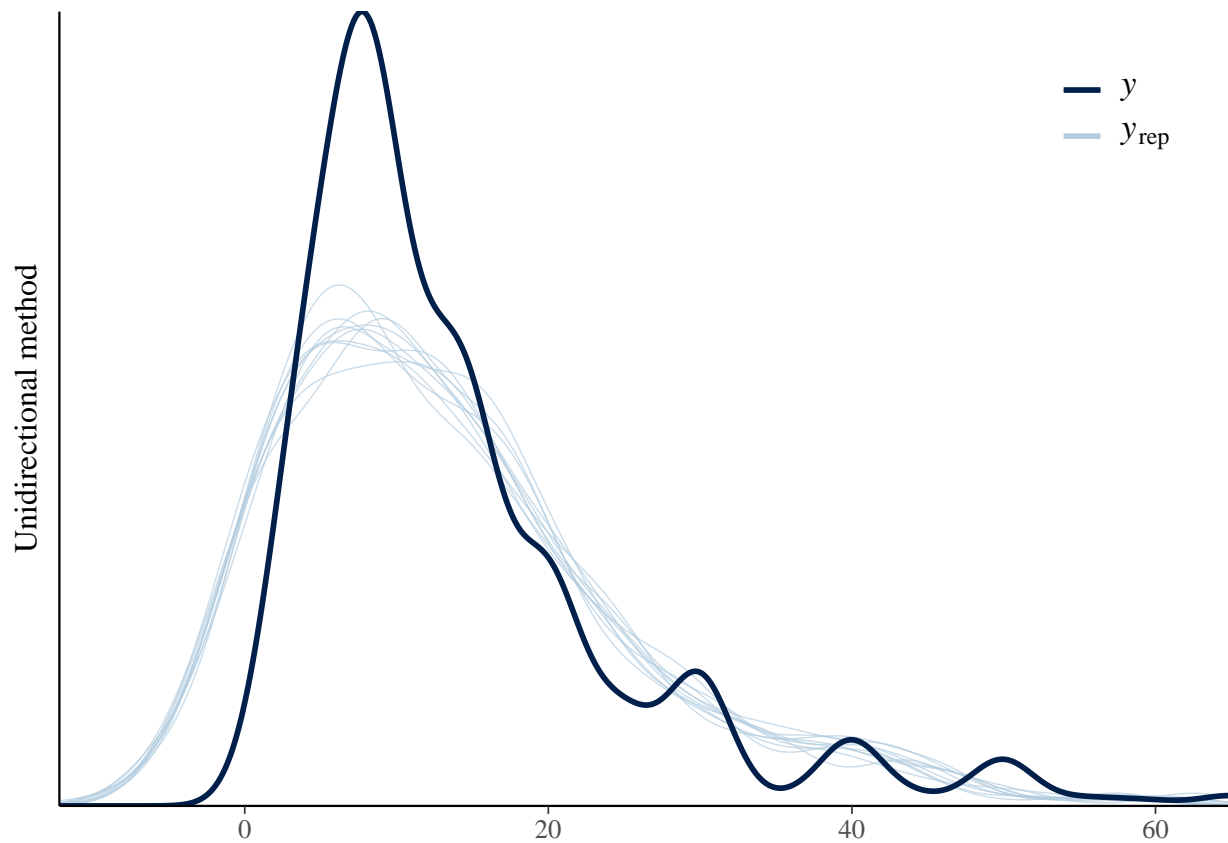
```

## Links: mu = identity; sigma = identity
## Formula: redness_rating_comparable ~ a * saturation^b
##      a ~ (1 | File)
##      b ~ (1 | File)
## Data: df[df$condition == "unidirectional", ] (Number of observations: 704)
## 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 1-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
## sd(a_Intercept)    0.24     0.05    0.17    0.35 1.00     4555     7435
## sd(b_Intercept)    0.91     0.18    0.63    1.33 1.00     4553     7719
##
## Population-Level Effects:
##      Estimate Est.Error 1-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
## a_Intercept    0.46     0.06    0.34    0.57 1.00     3542     5503
## b_Intercept    1.73     0.21    1.31    2.13 1.00     4438     7068
##
## Family Specific Parameters:
##      Estimate Est.Error 1-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
## sigma    0.04     0.00    0.03    0.04 1.00     20334     13326
##
## 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) +
    scale_x_continuous(labels=c(0,20,40,60), breaks=c(0,0.2,0.4,0.6)) +
    labs(y = 'Unidirectional method') +
    theme(legend.position = c(0.9, 0.9))
)

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

```



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

Model metrics (LOOCV, WAIC, marginal likelihood)

```
## Warning: Found 1 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 704 log-likelihood matrix
##
##      Estimate   SE
## elpd_loo  1307.9 36.9
## p_loo      62.5  9.5
## looic     -2615.8 73.8
## -----
## Monte Carlo SE of elpd_loo is NA.
##
## Pareto k diagnostic values:
##      Count Pct.   Min. n_eff
## (-Inf, 0.5] (good)   697  99.0%   1522
## (0.5, 0.7]  (ok)     6   0.9%    269
## (0.7, 1]    (bad)     1   0.1%     20
```

```
## (1, Inf) (very bad) 0 0.0% <NA>
## See help('pareto-k-diagnostic') for details.
```

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

```
## Warning:
## 34 (4.8%) p_waic estimates greater than 0.4. We recommend trying loo instead.
```

```
##
## Computed from 20000 by 704 log-likelihood matrix
##
##           Estimate   SE
## elpd_waic  1309.0 36.7
## p_waic      61.4  9.2
## waic       -2617.9 73.3
##
## 34 (4.8%) 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: 1246.292
## Estimate obtained in 7 iteration(s) via method "normal".
```

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

```
(formula = redness_rating_comparable ~ 0 + Intercept + saturation + (1 + saturation | File))
```

Fit linear model

```
## redness_rating_comparable ~ 0 + Intercept + saturation + (1 +
## saturation | 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 = 'study3_linear_unidirectional_x'
    )
)
```

```
## Family: gaussian
## Links: mu = identity; sigma = identity
## Formula: redness_rating_comparable ~ 0 + Intercept + saturation + (1 + saturation | File)
## Data: df[df$condition == "unidirectional", ] (Number of observations: 704)
## 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	1-95% CI	u-95% CI	Rhat	Bulk_ESS
sd(Intercept)	0.06	0.01	0.04	0.09	1.00	4600
sd(saturation)	0.18	0.03	0.12	0.25	1.00	4326
cor(Intercept,saturation)	-1.00	0.00	-1.00	-0.98	1.00	9694

```
##
```

	Tail_ESS
sd(Intercept)	9041
sd(saturation)	8110
cor(Intercept,saturation)	12046

```
##
## Population-Level Effects:
##
```

	Estimate	Est.Error	1-95% CI	u-95% CI	Rhat	Bulk_ESS	Tail_ESS
Intercept	-0.05	0.02	-0.08	-0.02	1.00	3188	5861
saturation	0.39	0.04	0.30	0.47	1.00	3094	5580

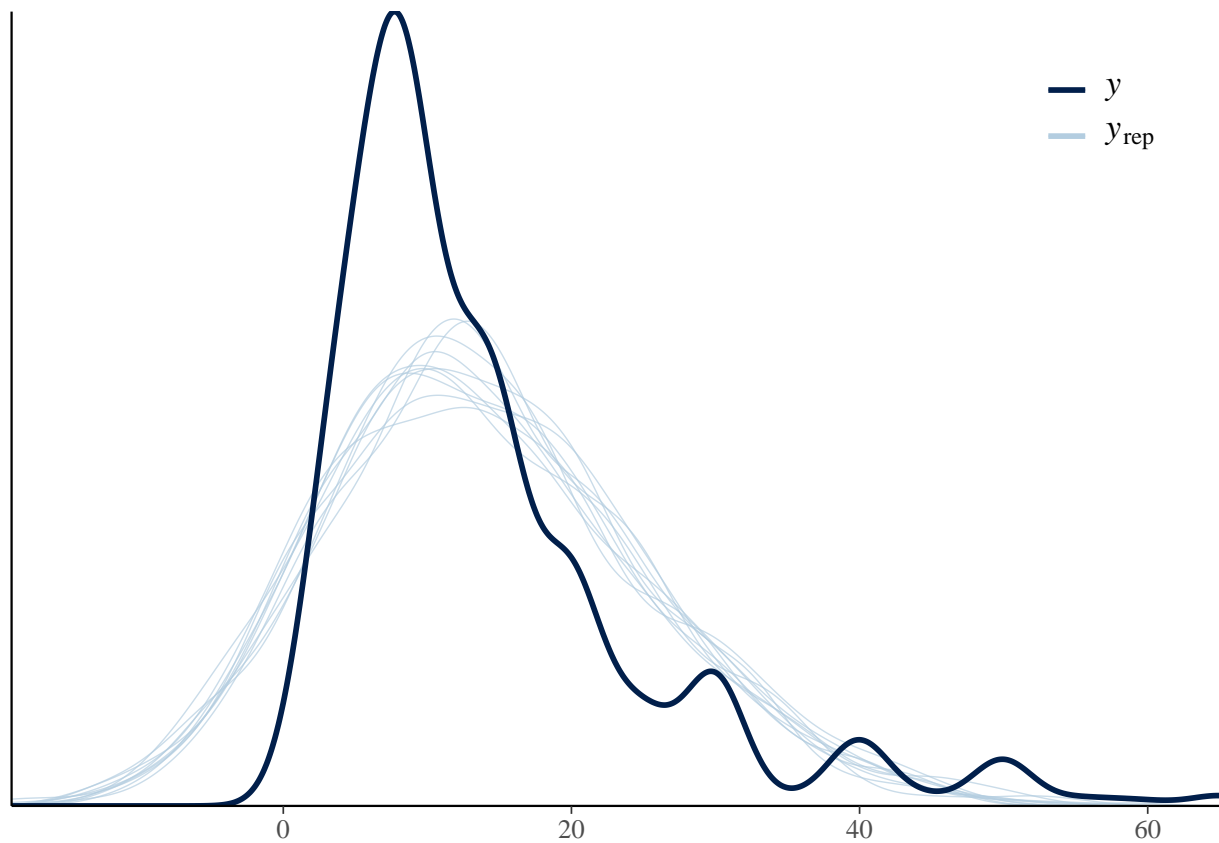
```
##
## Family Specific Parameters:
##
```

	Estimate	Est.Error	1-95% CI	u-95% CI	Rhat	Bulk_ESS	Tail_ESS
sigma	0.05	0.00	0.05	0.05	1.00	29565	15028

```
##
## 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(0,20,40,60), breaks=c(0,0.2,0.4,0.6)) +
  theme(legend.position = c(0.9, 0.9)))
```

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



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

Model metrics (LOOCV, WAIC, marginal likelihood)

```
##
## Computed from 20000 by 704 log-likelihood matrix
##
##      Estimate   SE
## elpd_loo  1082.9 35.9
## p_loo      41.1  6.0
## looic     -2165.9 71.8
## -----
## Monte Carlo SE of elpd_loo is 0.1.
##
## Pareto k diagnostic values:
##
##      Count Pct.   Min. n_eff
## (-Inf, 0.5] (good)   701  99.6%  1493
## (0.5, 0.7]  (ok)     3   0.4%   233
## (0.7, 1]    (bad)    0   0.0%   <NA>
## (1, Inf)    (very bad) 0   0.0%   <NA>
##
## All Pareto k estimates are ok (k < 0.7).
## See help('pareto-k-diagnostic') for details.
```

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

```
## Warning:
## 21 (3.0%) p_waic estimates greater than 0.4. We recommend trying loo instead.

##
## Computed from 20000 by 704 log-likelihood matrix
##
##           Estimate   SE
## elpd_waic    1083.2 35.8
## p_waic        40.8  5.9
## waic         -2166.5 71.7
##
## 21 (3.0%) 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: 1041.458
## 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: 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: 1
## Iteration: 2
## Iteration: 3
## Iteration: 4
## Iteration: 5
## Iteration: 1
## Iteration: 2
## Iteration: 3
## Iteration: 4
## Iteration: 5
## 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: 1
## Iteration: 2
## Iteration: 3
## Iteration: 4
## Iteration: 5
## Iteration: 1
## Iteration: 2
## Iteration: 3
## Iteration: 4
## Iteration: 5
## Iteration: 6
## Iteration: 7

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

## [1] "Mean logBF:204.815870047476Std logBF:0.0278751284377827"

```

```

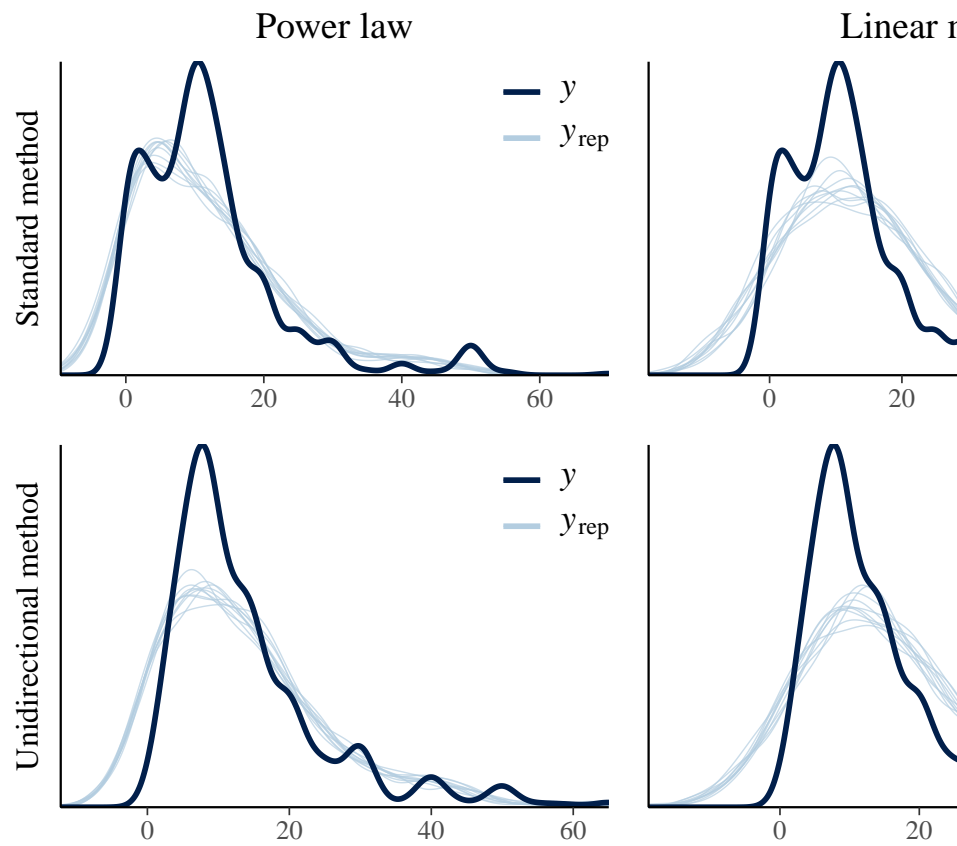
pp_grid = (
  grid.arrange(
    pow_standard_plot,
    lin_standard_plot,
    pow_unidirectional_plot,

```

```

    lin_unidirectional_plot
  )
)

```



```

ggsave(
  paste0("final_plots/study3_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(saturation,condition) %>% summarise(median_rating = median(redness_rating_comp))
```

Fit power law (standard condition)

```
## 'summarise()' regrouping output by 'saturation' (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 * saturation ^ b , a ~ 1, b ~ 1 , nl = TRUE))

## median_rating ~ a * saturation^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 = 'study3_power_standard_agg_x'
    )
)

## Family: gaussian
## Links: mu = identity; sigma = identity
## Formula: median_rating ~ a * saturation^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 1-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
## a_Intercept    0.33     0.04   0.25   0.40 1.00    5929    6659
## b_Intercept    1.72     0.28   1.14   2.25 1.00    5903    6051
##
## Family Specific Parameters:
##      Estimate Est.Error 1-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
## sigma    0.03     0.01   0.01   0.05 1.00    6154    8064
##
## 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 * saturation ^ b , a ~ 1, b ~ 1 , nl = TRUE))
```

Fit power law (unidirectional condition)

```
## median_rating ~ a * saturation^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 = 'study3_power_unidirectional_agg_x'
    )
)
```

```
## Family: gaussian
## Links: mu = identity; sigma = identity
## Formula: median_rating ~ a * saturation^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 1-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
## a_Intercept    0.34     0.04   0.26   0.42 1.00    6117    5750
## b_Intercept    1.56     0.26   1.03   2.08 1.00    6311    5676
##
## Family Specific Parameters:
##      Estimate Est.Error 1-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
## sigma    0.03     0.01   0.02   0.06 1.00    6197    6391
##
## 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).
```