A hands-on example of Bayesian mixed models with *brms*

Andrey Anikin

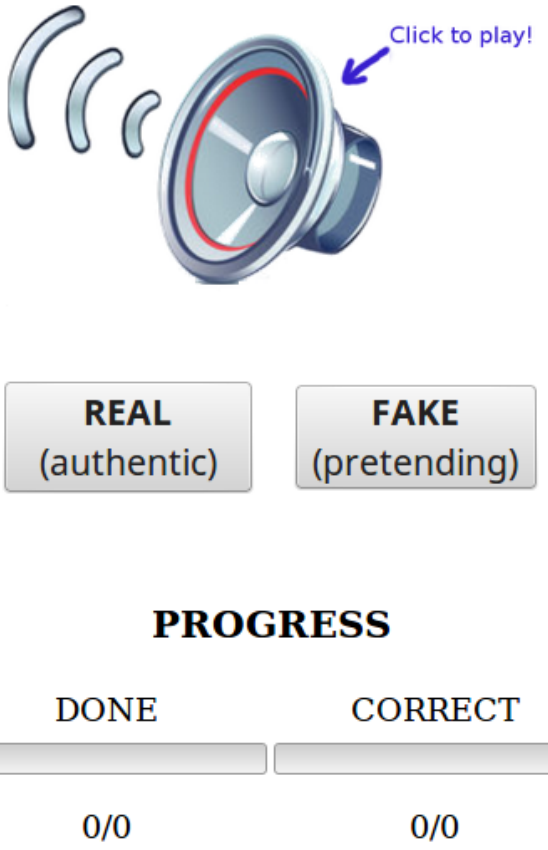
Lund University Cognitive Science
andrey.anikin@lucs.lu.se

Research question

Authentic vs. acted emotional vocalizations

Experiment

- 139 authentic sounds (“ut”)
- 139 actor portrayals, including
  - 1 corpus with 14 sounds by professional actors (“hawk”)
  - 5 corpora with 125 sounds by amateurs
- Listeners hear a mix and rate each as “real” or “fake”



We want to know...

- Are authentic sounds more likely to be rated as “real” vs. actor portrayals?
- Are professional actors better than amateurs?
- Are professional actors as convincing as “the real thing”?

Data (subset)

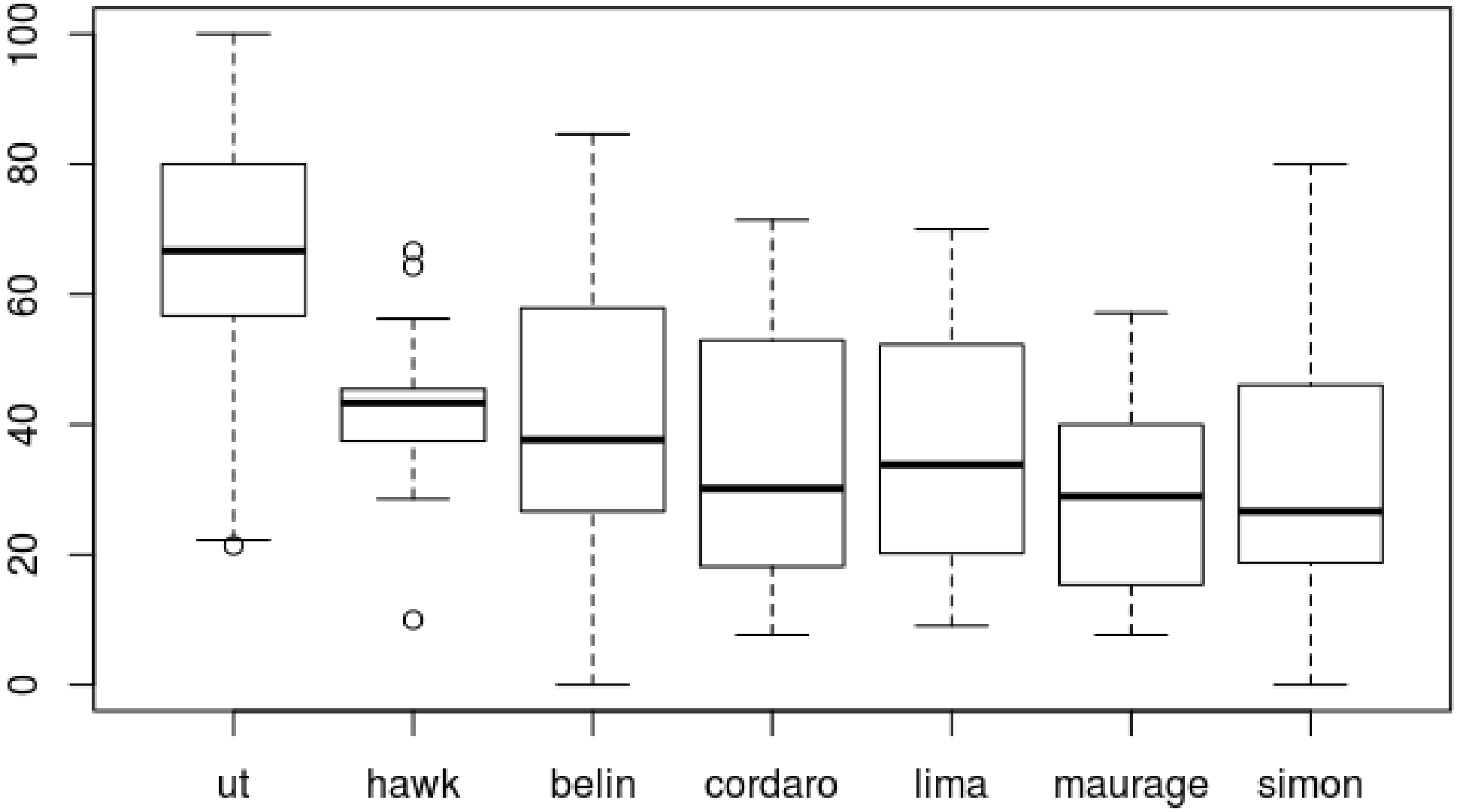
- 3900 real / fake judgments of 278 sounds from 7 corpora by 46 participants

```
> head(df)

     id                                sound corpus  real
n6X2yZ    belin_pain_60.mp3                  belin FALSE
n6X2yZ    ut_achievement_pregn_11-f.mp3         ut  TRUE
n6X2yZ    ut_sadness_sad-cry_50-m.mp3           ut FALSE
n6X2yZ    lima_amusement_M_6.mp3               lima FALSE
n6X2yZ    belin_pain_06.mp3                  belin FALSE
n6X2yZ    ut_anger_13-m-roar-scream.mp3         ut  TRUE
...
```

Descriptives

```
> aggregate(real ~ corpus, df, mean)

   corpus      real
1      ut 0.6671819   # authentic
2    hawk 0.4427861   # professional actors
3   belin 0.3905473   # amateurs
4 cordaro 0.3897436   # amateurs
5    lima 0.3517787   # amateurs
6 maurage 0.2914980   # amateurs
7   simon 0.3406863   # amateurs
```

Descriptives



Model specification

Non-Bayesian (GLMM) with lme4	Bayesian with brms

`mod0 = lme4::glmer(`
`  real ~ corpus+(1|sound)+(1|id),`
`  data = df,`
`  family = 'binomial'`
`)`	

Model specification

Non-Bayesian (GLMM) with lme4	Bayesian with brms

`mod0 = lme4::glmer(`
`  real ~ corpus+(1|sound)+(1|id),`
`  data = df,`
`  family = 'binomial'`
`)`
	`mod = brms::brm(`
	`  real ~ corpus+(1|sound)+(1|id),`
	`  data = df,`
	`  ...`
	`)`

Model specification

Non-Bayesian (GLMM) with lme4	Bayesian with brms

`mod0 = lme4::glmer(`
`  real ~ corpus+(1|sound)+(1|id),`
`  data = df,`
`  family = '`**`binomial`**`'`
`)`
	`mod = brms::brm(`
	`  real ~ corpus+(1|sound)+(1|id),`
	`  data = df,`
	`  family = '`**`bernoulli`**`',`
	`  ...`
	`)`

Model specification

Non-Bayesian (GLMM) with lme4	Bayesian with brms

`mod0 = lme4::glmer(`
` real ~ corpus+(1|sound)+(1|id),`
` data = df,`
` family = '`**`binomial`**`'`
`)`
	`mod = brms::brm(`
	` real ~ corpus+(1|sound)+(1|id),`
	` data = df,`
	` family = '`**`bernoulli`**`',`
	` `**`prior = set_prior('normal(0, 3)'),`**
	` `**`...`**
	`)`

Model specification

Non-Bayesian (GLMM) with lme4	Bayesian with brms

`mod0 = lme4::glmer(`
` real ~ corpus+(1|sound)+(1|id),`
` data = df,`
` family = '`**`binomial`**`'`
`)`
	`mod = brms::brm(`
	` real ~ corpus+(1|sound)+(1|id),`
	` data = df,`
	` family = '`**`bernoulli`**`',`
	` `**`prior = set_prior('normal(0, 3)'),`**
	` `**`iter = 1000,`**
	` `**`...`**
	`)`

Model specification

Non-Bayesian (GLMM) with lme4	Bayesian with brms

`mod0 = lme4::glmer(`
` real ~ corpus+(1|sound)+(1|id),`
` data = df,`
` family = '`**`binomial`**`'`
`)`
	`mod = brms::brm(`
	` real ~ corpus+(1|sound)+(1|id),`
	` data = df,`
	` family = '`**`bernoulli`**`',`
	` `**`prior = set_prior('normal(0, 3)'),`**
	` `**`iter = 1000,`**
	` `**`chains = 4,`**
	` `**`...`**
	`)`

Model specification

Non-Bayesian (GLMM) with lme4	Bayesian with brms

`mod0 = lme4::glmer(`
`  real ~ corpus+(1|sound)+(1|id),`
`  data = df,`
`  family = '`**`binomial`**`'`
`)`
	`mod = brms::brm(`
	`  real ~ corpus+(1|sound)+(1|id),`
	`  data = df,`
	`  family = '`**`bernoulli`**`',`
	`  prior = set_prior('normal(0, 3)'),`
	`  iter = 1000,`
	`  chains = 4,`
	`  cores = 4`
	`)`

Model specification

Non-Bayesian (GLMM) with lme4	Bayesian with brms

mod0 = lme4::glmer(
  real ~ corpus+(1|sound)+(1|id),
  data = df,
  family = '**binomial**'
)
 	
# running time: **6 s** 	
	mod = brms::brm(
	  real ~ corpus+(1|sound)+(1|id),
	  data = df,
	  family = '**bernoulli**',
	  prior = set_prior('normal(0, 3)'),
	  iter = 1000,
	  chains = 4,
	  cores = 4
	)
	# running time: 40s compilation +
	  50 s sampling = **1.5 min**

Model diagnostics: has the model converged?

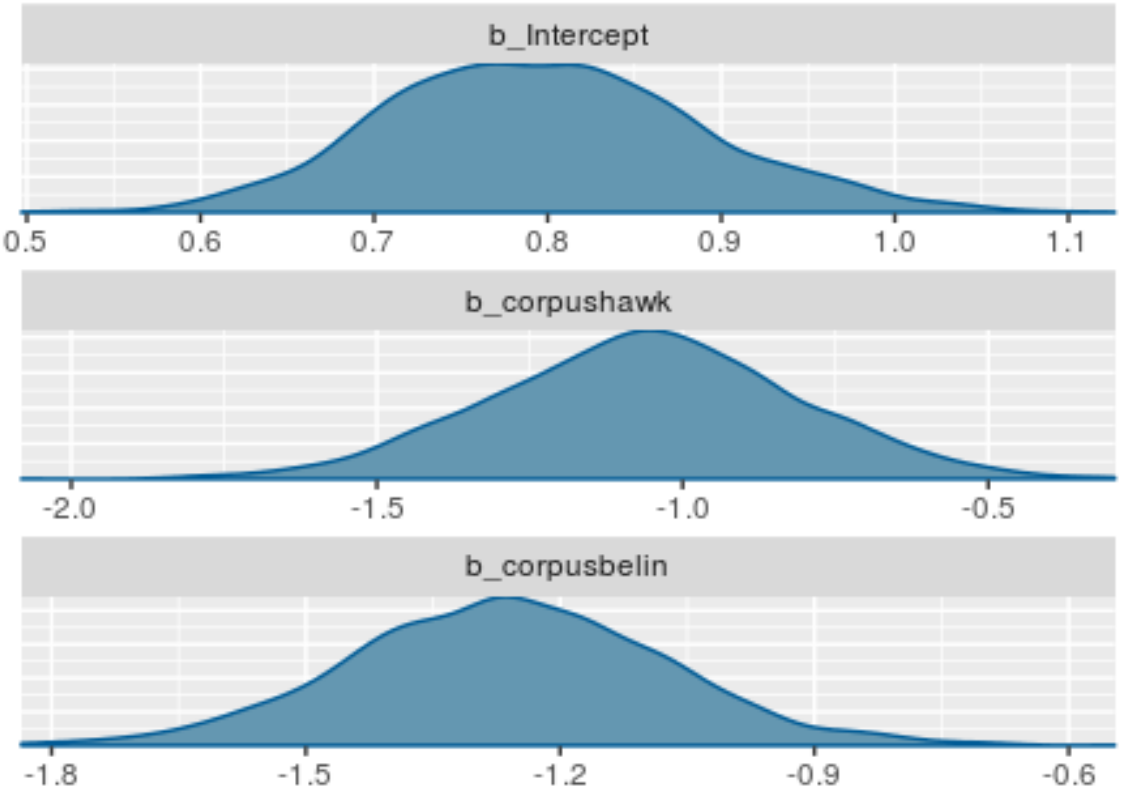
```
> summary(mod)
...
```

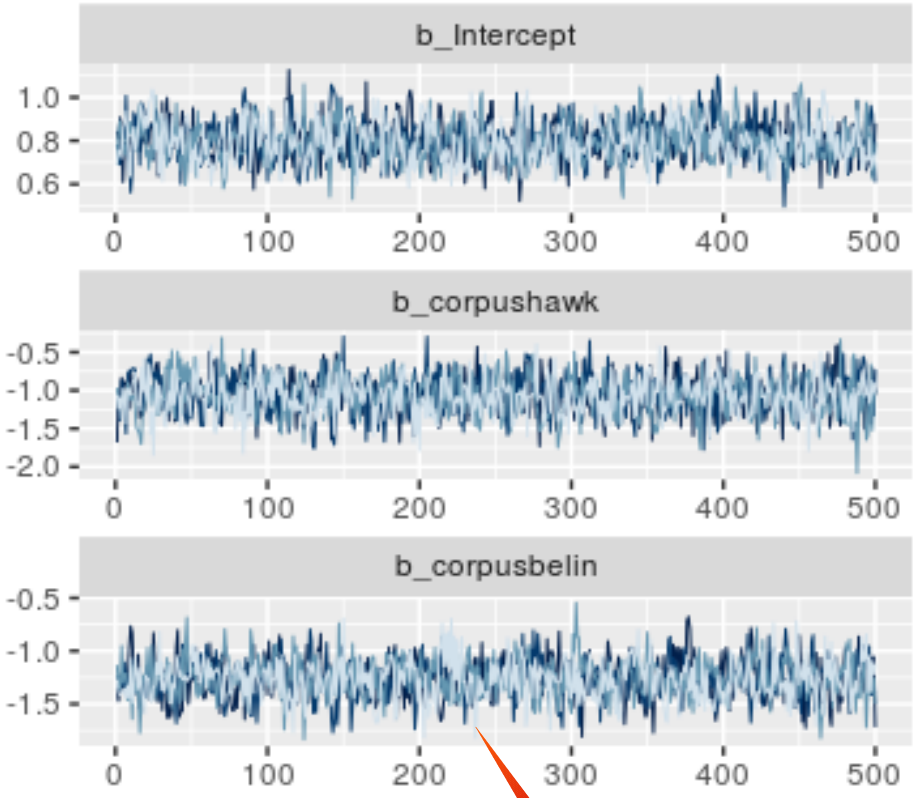
Model diagnostics: has the model converged?

```
> summary(mod)
...

Group-Level Effects:
~id (Number of levels: 46)
              Estimate Est.Error l-95% CI u-95% CI Eff.Sample Rhat
sd(Intercept)     0.31      0.07     0.19     0.45        636 1.01

~sound (Number of levels: 278)
              Estimate Est.Error l-95% CI u-95% CI Eff.Sample Rhat
sd(Intercept)     0.68      0.06     0.57     0.79        838 1.00

Population-Level Effects:
              Estimate Est.Error l-95% CI u-95% CI Eff.Sample Rhat
Intercept         0.80      0.09     0.63     0.99       1109 1.00
corpushawk       -1.06      0.25    -1.56    -0.59       1341 1.00
corpusbelin      -1.26      0.19    -1.63    -0.89        992 1.00
corpuscordaro    -1.34      0.26    -1.85    -0.80       1083 1.00
corpuslima       -1.44      0.17    -1.77    -1.11       1159 1.00
corpusmaurage    -1.75      0.23    -2.20    -1.30       1440 1.00
corpussimon      -1.54      0.19    -1.93    -1.17       1163 1.00
```

Model diagnostics: has the model converged?

```
> plot(mod)
```

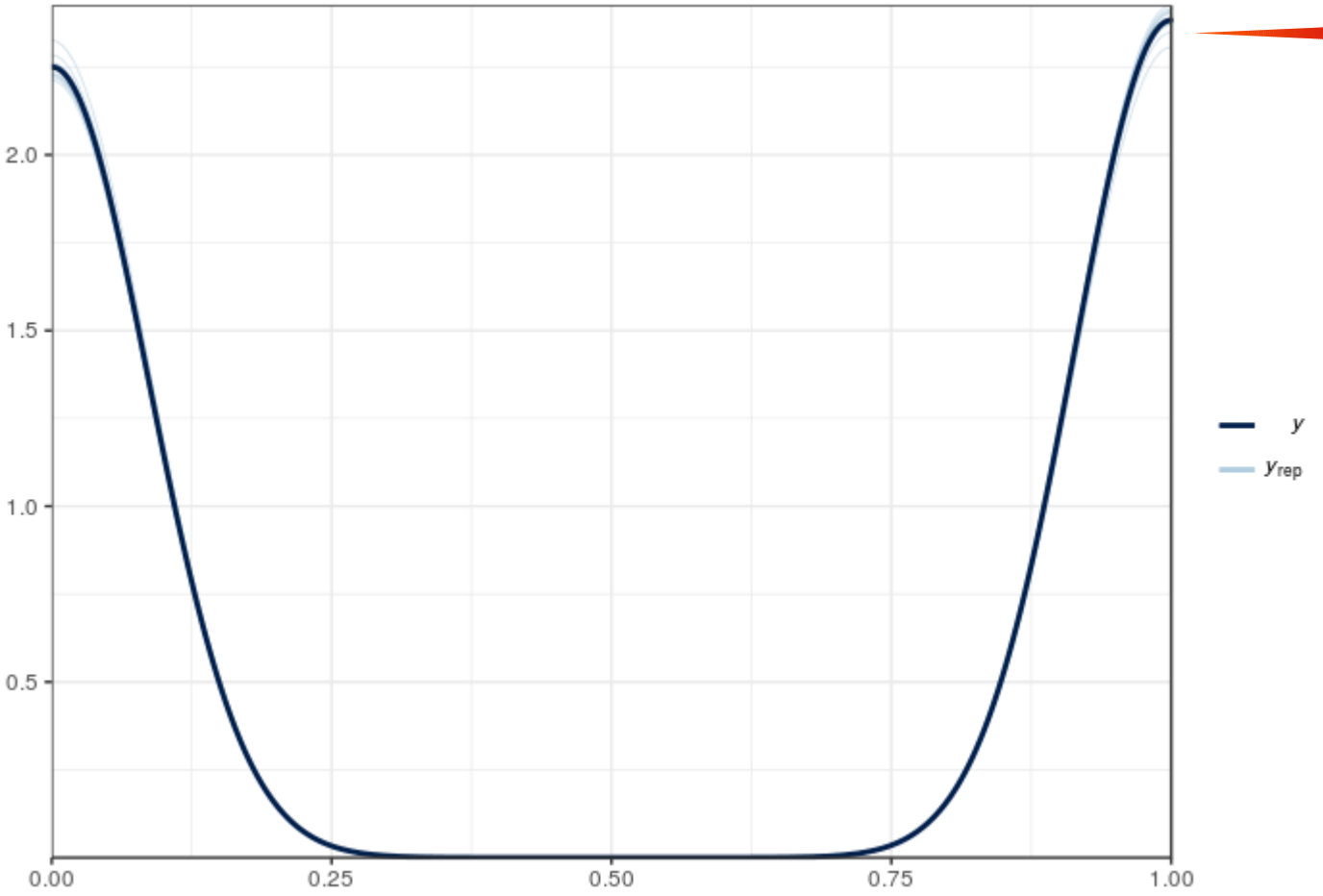




Nice, hairy caterpillars

Model diagnostics: is it a reasonable fit?

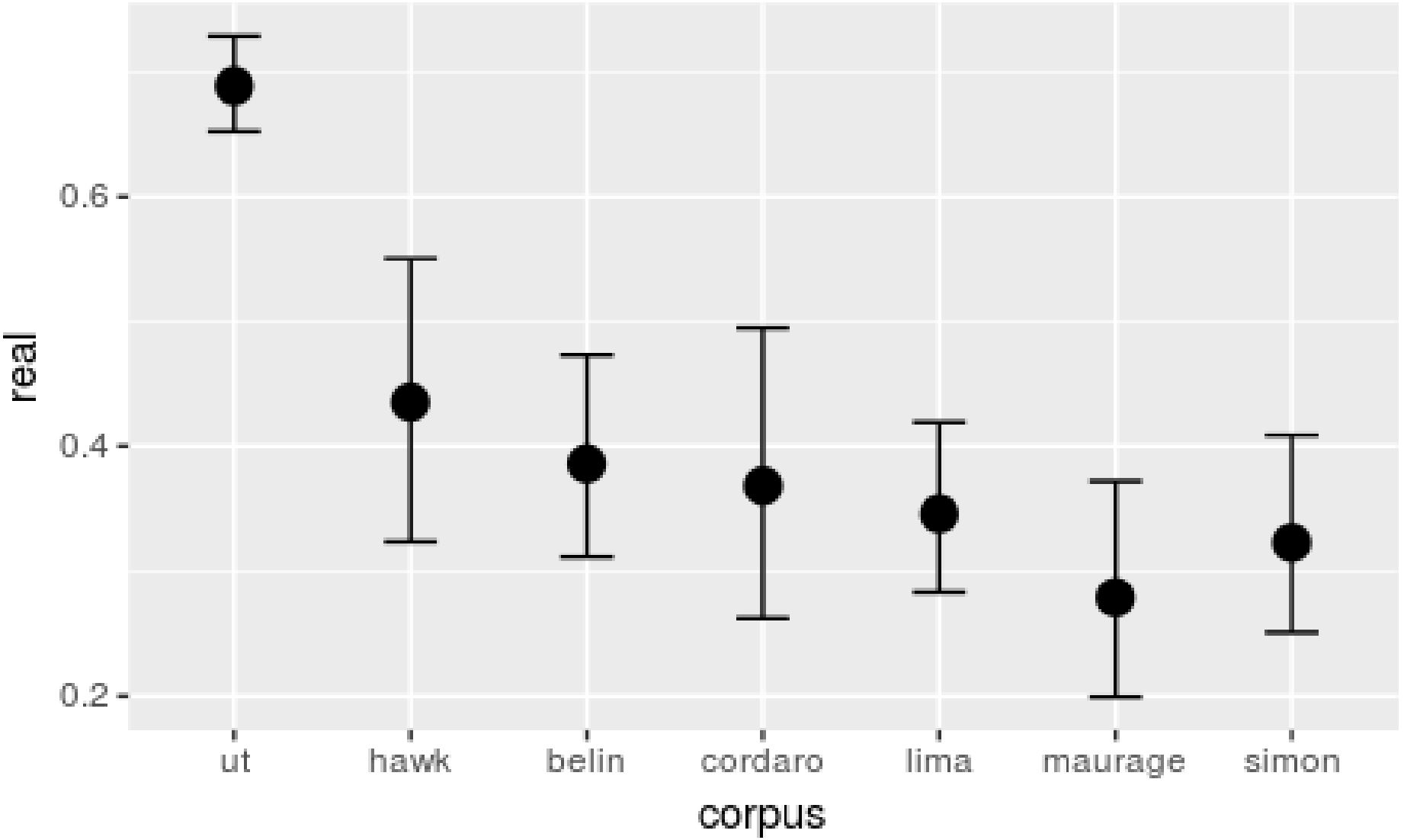
```
> pp = brms::pp_check(mod)
> pp + theme_bw()
```



Similar density plots of observed and predicted values

Default plot of model predictions

```
> brms::marginal_effects(mod)
```



Custom plot of model predictions

```
> newdata = data.frame(corpus = levels(df$corpus))
```

Custom plot of model predictions

```
> newdata = data.frame(corpus = levels(df$corpus))
> fit = fitted(
>   mod,
>   newdata = newdata,
>   ...
> )
```

Custom plot of model predictions

```
> newdata = data.frame(corpus = levels(df$corpus))
> fit = fitted(
>   mod,
>   newdata = newdata,
>   re_formula = NA,    # ignore random effects
>   ...
> )
```

Custom plot of model predictions

```
> newdata = data.frame(corpus = levels(df$corpus))
> fit = fitted(
>   mod,
>   newdata = newdata,
>   re_formula = NA,    # ignore random effects
>   summary = TRUE      # mean and 95% CI
> )
```

Custom plot of model predictions

```
> newdata = data.frame(corpus = levels(df$corpus))
> fit = fitted(
>   mod,
>   newdata = newdata,
>   re_formula = NA,    # ignore random effects
>   summary = TRUE      # mean and 95% CI
> ) * 100               # convert to %
```

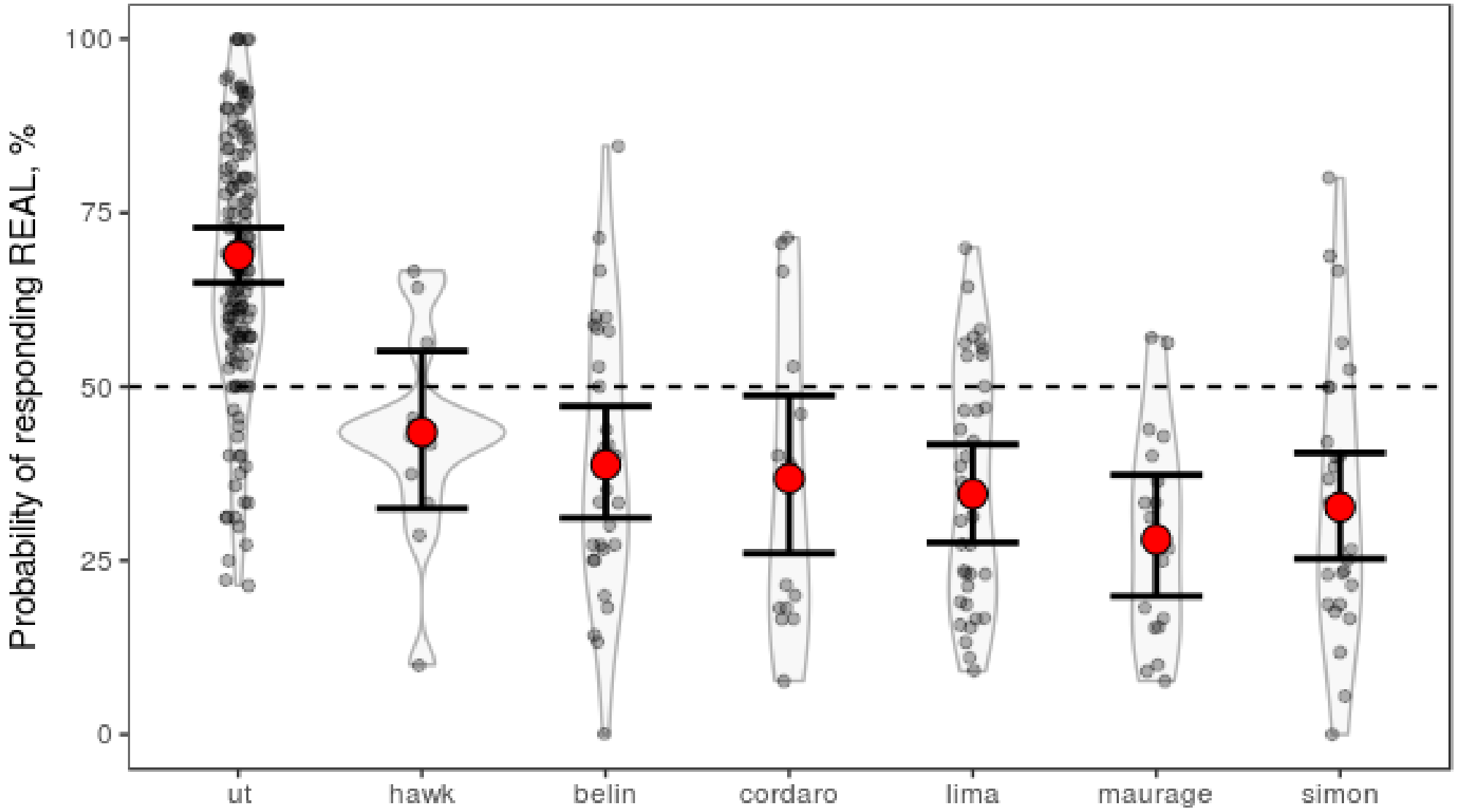
Custom plot of model predictions

```
> newdata = data.frame(corpus = levels(df$corpus))
> fit = fitted(
>   mod,
>   newdata = newdata,
>   re_formula = NA,    # ignore random effects
>   summary = TRUE      # mean and 95% CI
> ) * 100               # convert to %
> colnames(fit) = c('fit', 'se', 'lwr', 'upr')
> df_plot = cbind(newdata, fit)
```

Custom plot of model predictions

```
> df_plot

   corpus       fit        se       lwr       upr
1      ut 68.86003 2.030859 64.91156 72.85869
2    hawk 43.43550 5.780774 32.49832 55.09837
3   belin 38.77180 4.140586 31.12392 47.18532
4 cordaro 36.80961 5.865695 26.04502 48.72115
5    lima 34.57693 3.586463 27.55386 41.71141
6 maurage 28.03637 4.401277 19.87059 37.30708
7   simon 32.68807 3.915151 25.28420 40.48484
```

Custom plot of model predictions



Contrasts between corpora

```
> fit1 = as.data.frame(fitted(
    mod,
    newdata = data.frame(corpus = levels(df$corpus)),
    re_formula = NA,
    summary = FALSE  # extract the full MCMC
))
> colnames(fit1) = newdata$corpus
```

Contrasts between corpora

```
> head(fit1)

         ut      hawk     belin   cordaro      lima   maurage     simon
1 0.6991368 0.3017015 0.3754336 0.3122634 0.3364265 0.3658070 0.3380636
2 0.6919216 0.4318584 0.3402173 0.2790131 0.3921006 0.2571805 0.3082657
3 0.7124336 0.3810847 0.4205503 0.3073799 0.3349322 0.2701446 0.4140096
4 0.7063214 0.5108651 0.3773467 0.3065392 0.4512227 0.3557162 0.4012695
5 0.6479099 0.4183722 0.3395259 0.2441611 0.2657999 0.2506801 0.3163448
6 0.6881893 0.4754693 0.3902508 0.3028129 0.2871305 0.3141214 0.3555258
...

> nrow(fit1)
[1] 2000
```

Contrasts between corpora

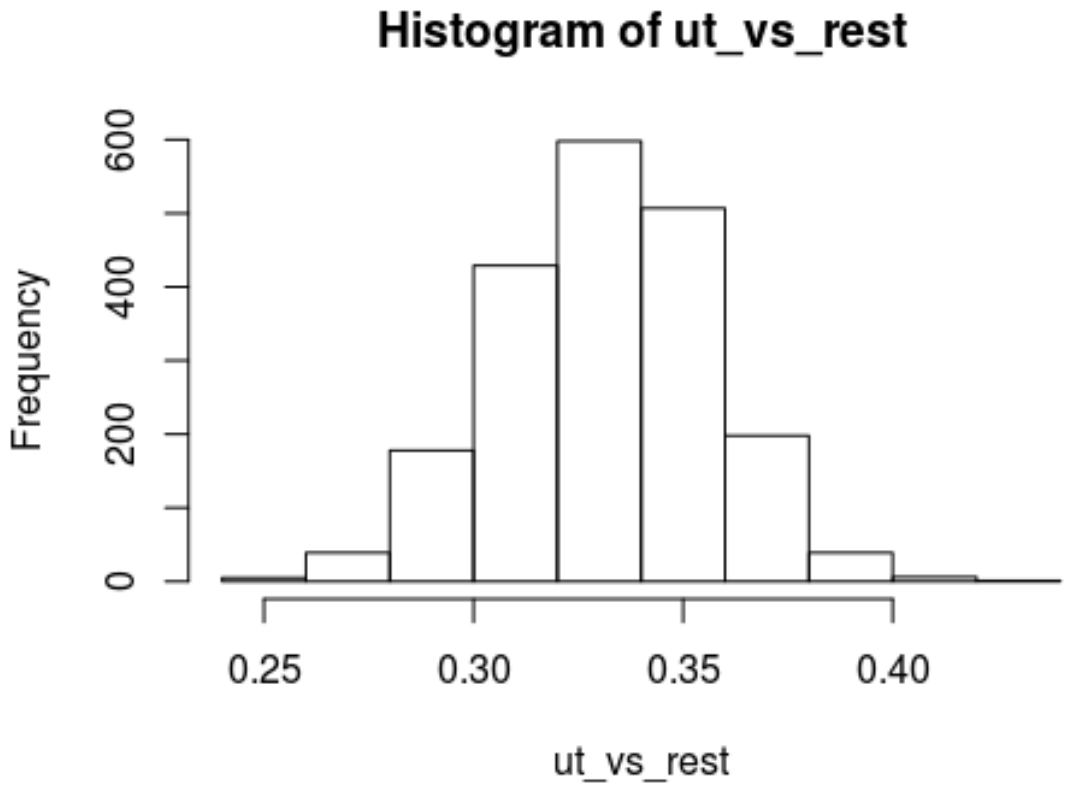
- Q1: Are authentic sounds more likely to be rated as “real” vs. actor portrayals?

```
> ut_vs_rest = fit1$ut -
    (
      fit1$belin +
      fit1$cordaro +
      fit1$hawk +
      fit1$lima +
      fit1$maurage +
      fit1$simon
    ) / 6
```

Contrasts between corpora

- Q1: Are authentic sounds more likely to be rated as “real” vs. actor portrayals?

```
> hist(ut_vs_rest)
```



Contrasts between corpora

- Q1: Are authentic sounds more likely to be rated as “real” vs. actor portrayals?

```
> quantile(ut_vs_rest, probs = c(.5, .025, .975))

      50%      2.5%     97.5% 
0.3319703 0.2835615 0.3816895 
```

Thus: yes, and the most credible difference in perceived authenticity is 33.2%, 95% CI [28.4, 38.2]

Contrasts between corpora

- Q2: Are professional actors better than amateurs?

```
> hawk_vs_rest = fit1$hawk - (fit1$belin +
fit1$cordaro + fit1$lima + fit1$maurage +
fit1$simon) / 5
```

Contrasts between corpora

- Q2: Are professional actors better than amateurs?

```
> hawk_vs_rest = fit1$hawk - (fit1$belin +
fit1$cordaro + fit1$lima + fit1$maurage +
fit1$simon) / 5
> quantile(hawk_vs_rest, probs = c(.5, .025, .975))

       50%        2.5%       97.5%
0.09309276 -0.02252535  0.21356418
```

Thus: possibly, but not much, and the evidence is not very strong: 9.3% [-2.3, 21.4]

Contrasts between corpora

- Q2: Are professional actors better than amateurs?

```
> mean(hawk_vs_rest > 0)

[1] 0.939
```

Thus: we are 93.9% confident that professional actors are better than amateurs

Contrasts between corpora

- Q3: Are professional actors as convincing as “the real thing”?

```
> ut_vs_hawk = fit1$ut - fit1$hawk
> quantile(ut_vs_hawk, probs = c(.5, .025, .975))

      50%      2.5%     97.5% 
0.2543908 0.1372629 0.3693648 
```

Thus: definitely not: sounds recorded in real life are judged as 25.4% [13.7, 36.9] more authentic than sounds by professional actors

Demystifying *brms* code for contrasts

1. Extract MCMC for regression coefficients

```
> coda = brms::posterior_samples(mod)
```

Demystifying *brms* code for contrasts

1. Extract MCMC for regression coefficients

```
> coda = brms::posterior_samples(mod)
> # colnames(coda)
> coda = coda[, 1:7]
```

Demystifying *brms* code for contrasts

1. Extract MCMC for regression coefficients

```
> coda = brms::posterior_samples(mod)
> # colnames(coda)
> coda = coda[, 1:7]
> head(coda)
```

	b_Intercept	b_corpushawk	b_corpusbelin	b_corpuscordaro	b_corpuslima	b_corpusmaurage	b_corpussimon
1	0.8208295	-1.0877844	-1.039162	-1.013646	-1.418948	-1.918795	-1.575574
2	0.7763032	-0.9574446	-1.114927	-1.483529	-1.549794	-1.901897	-1.410622
3	0.7956587	-1.3054393	-1.428899	-1.648397	-1.356121	-1.763765	-1.457054
4	0.7773173	-1.0152087	-1.437142	-1.574973	-1.589480	-1.727606	-1.548478
5	0.8811827	-1.2514199	-1.284633	-1.219970	-1.392294	-1.857699	-1.649942
6	0.7042098	-1.1067592	-1.268842	-1.028635	-1.177143	-2.019502	-1.441486

...

Demystifying *brms* code for contrasts

2. Convert to MCMC for outcome variable

```
> antilogit = function(x) 1 / (1 + exp(-x))

> fit2 = data.frame(
    ut =      antilogit(coda[, 1]),
    hawk =    antilogit(coda[, 1] + coda[, 2]),
    belin =   antilogit(coda[, 1] + coda[, 3]),
    cordaro = antilogit(coda[, 1] + coda[, 4]),
    lima =    antilogit(coda[, 1] + coda[, 5]),
    maurage = antilogit(coda[, 1] + coda[, 6]),
    simon =   antilogit(coda[, 1] + coda[, 7])
 )
```

Demystifying *brms* code for contrasts

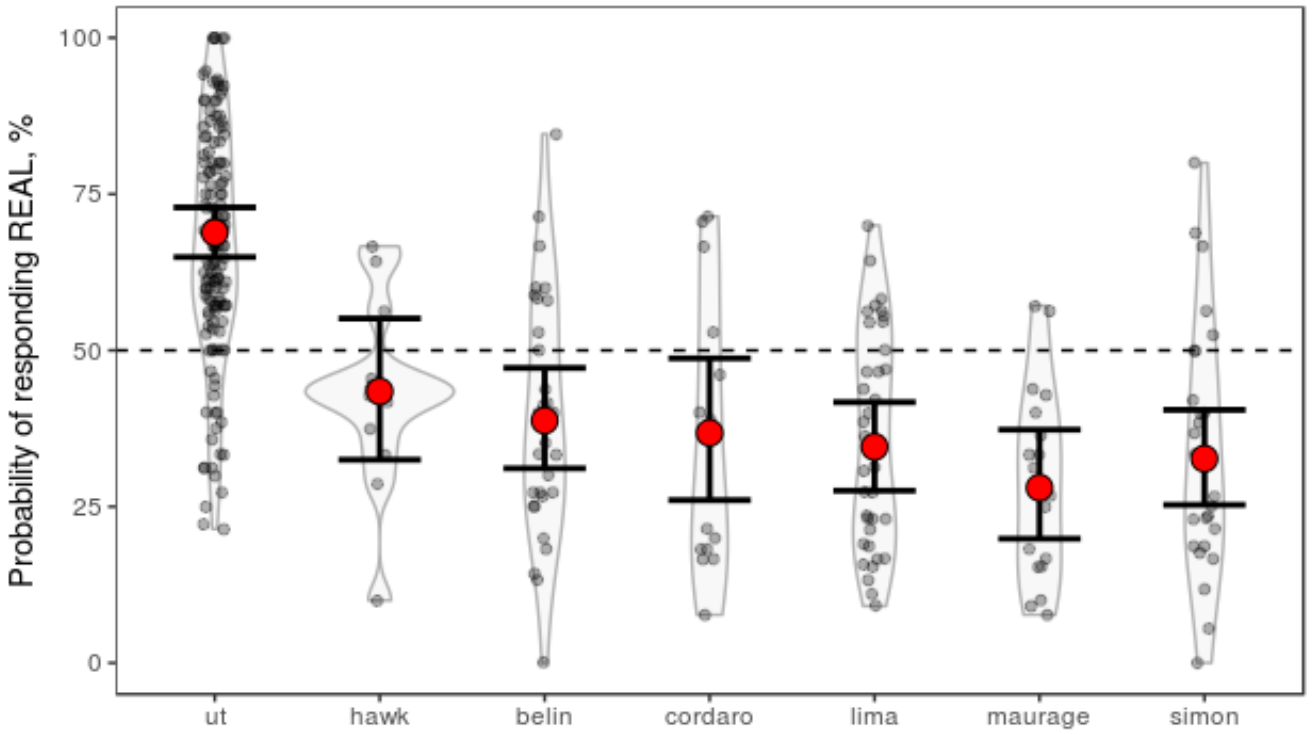
3. Compare to fit1

```
> head(fit1)  # extracted with fitted(...)
         ut      hawk     belin   cordaro      lima   maurage     simon
1 0.6944124 0.4336548 0.4456326 0.4519446 0.3547743 0.2501213 0.3197884
2 0.6848828 0.4548381 0.4161438 0.3302122 0.3157244 0.2449751 0.3465319
3 0.6890451 0.3752449 0.3467763 0.2988587 0.3634405 0.2752581 0.3404262
4 0.6851016 0.4408060 0.3407791 0.3105273 0.3074297 0.2788267 0.3162281
5 0.7070672 0.4084837 0.4004836 0.4161040 0.3749331 0.2735836 0.3167476
6 0.6691205 0.4007000 0.3624764 0.4195976 0.3839222 0.2116026 0.3236001

> head(fit2)  # extracted manually from MCMC
         ut      hawk     belin   cordaro      lima   maurage     simon
1 0.6944124 0.4336548 0.4456326 0.4519446 0.3547743 0.2501213 0.3197884
2 0.6848828 0.4548381 0.4161438 0.3302122 0.3157244 0.2449751 0.3465319
3 0.6890451 0.3752449 0.3467763 0.2988587 0.3634405 0.2752581 0.3404262
4 0.6851016 0.4408060 0.3407791 0.3105273 0.3074297 0.2788267 0.3162281
5 0.7070672 0.4084837 0.4004836 0.4161040 0.3749331 0.2735836 0.3167476
6 0.6691205 0.4007000 0.3624764 0.4195976 0.3839222 0.2116026 0.3236001
```

Reporting Bayesian analysis Methods

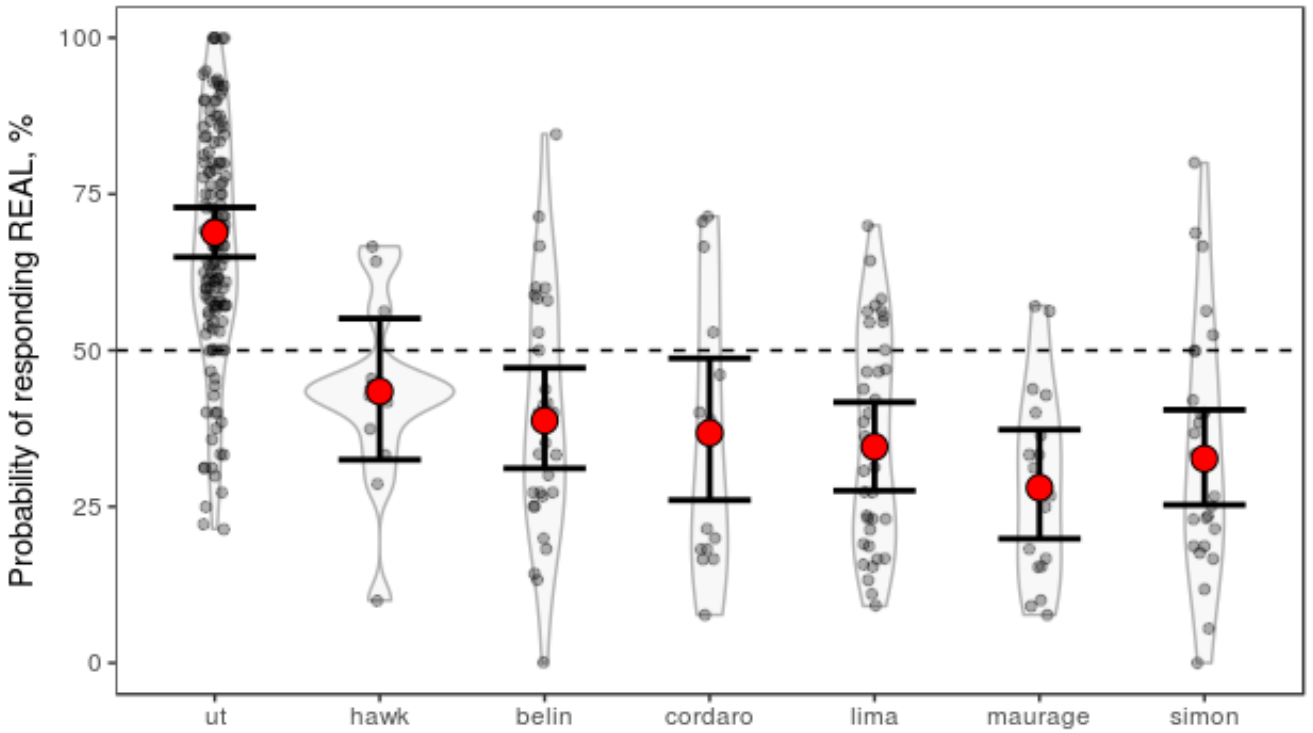
- Describe the model like any GLMM (trial-level data, group-level intercepts, etc)
- “All Bayesian models were created in Stan computational framework (http://mc-stan.org/) accessed with *brms* package (Bürkner, 2017). To improve convergence and guard against overfitting, we specified mildly informative conservative priors.”

Reporting Bayesian analysis: Figure caption



“Fig. 1 Bla-bla... Solid red points show fitted values: the mean of posterior distribution and 95% credible intervals”

Reporting Bayesian analysis: Figure caption



“Fig. 1 Bla-bla... Solid red points show fitted values: the mean of posterior distribution and ~~95% credible intervals~~ 95% CI”

Reporting Bayesian analysis: Results

- Authentic sounds were 33.2% (95% CI [28.4, 38.2]) more likely to be rated as “real” compared to sounds produced intentionally
- It is possible that portrayals by professional actors were marginally (9.3% [-2.3, 21.4]) more realistic than those by amateurs, but this difference was not statistically robust
- Compared to authentic sounds, portrayals by professional actors were still 25.4% [13.7, 36.9] less likely to be rated as “real”

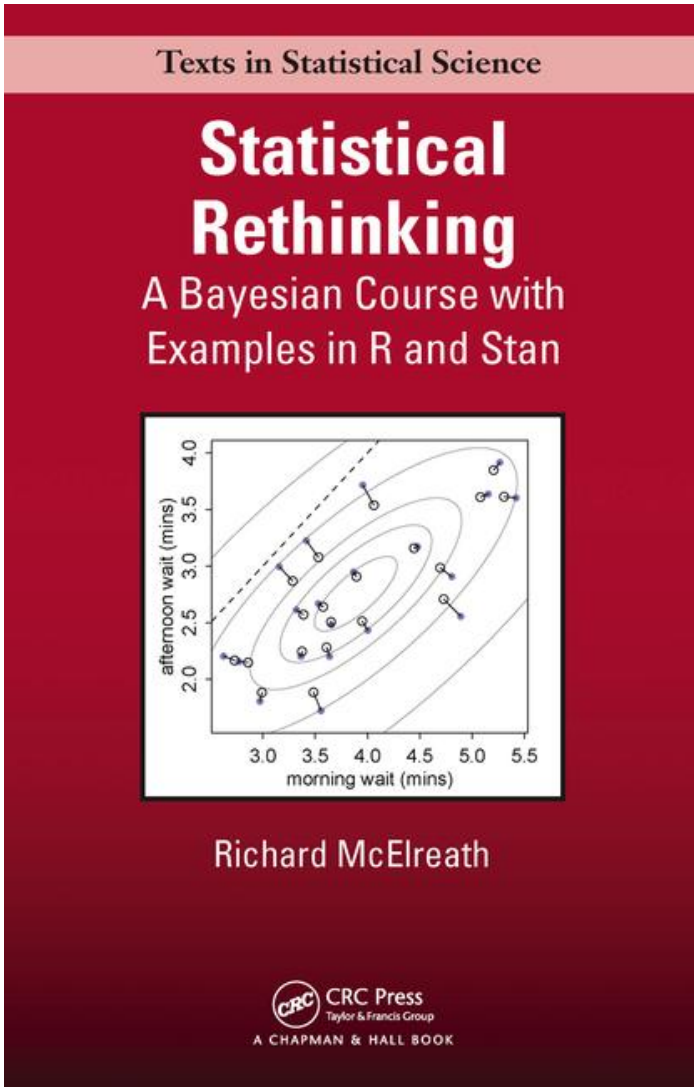
brms makes life easy, but...

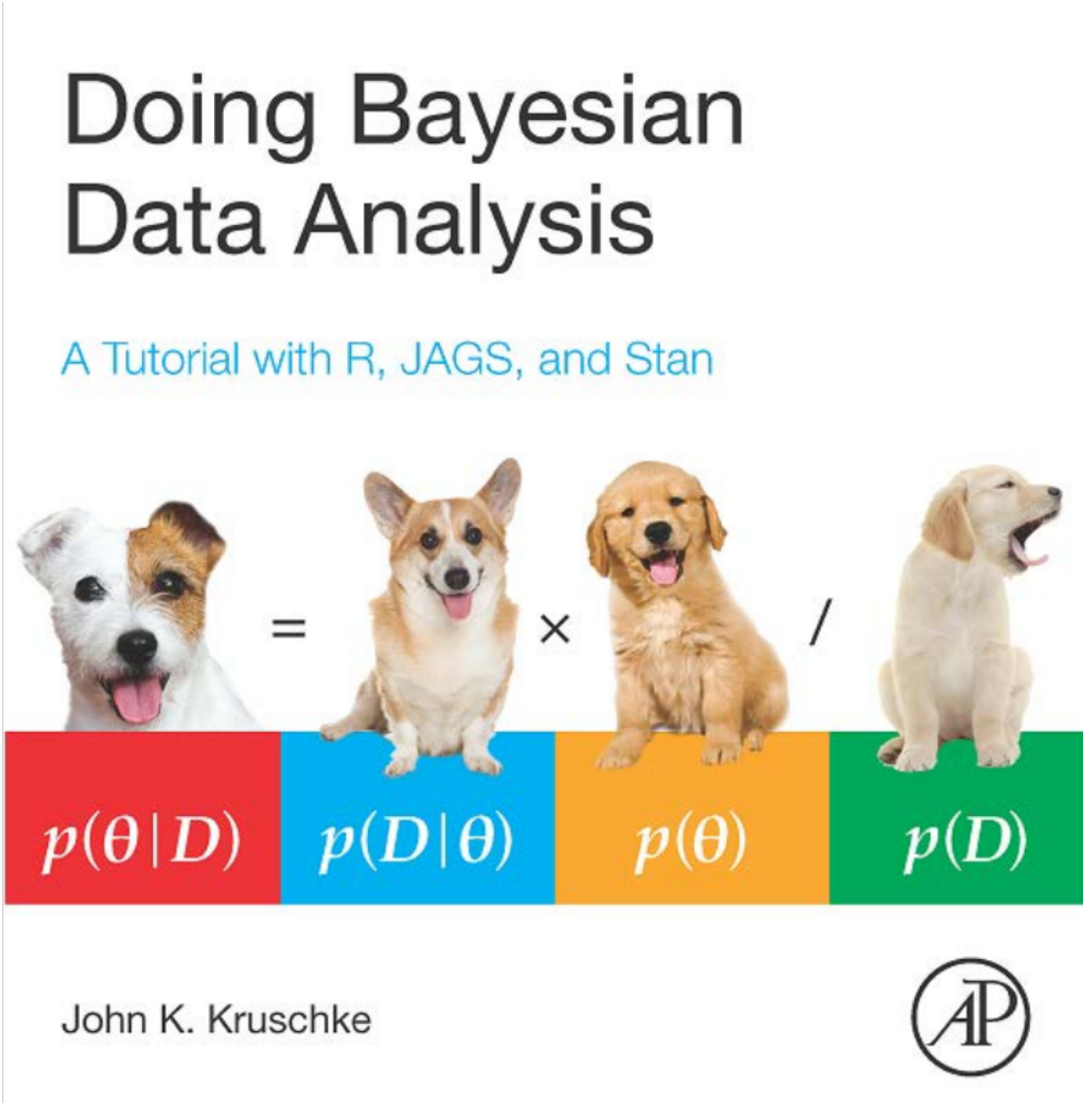
```
library(brms)
df = read.csv('real-fake_indiv-answers.csv')
df = droplevels(df_master[df_master$noisy == T, ])
df$corpus = factor(df$corpus, levels = c('ut', 'hawk', 'belin', 'cordaro', 'lima', 'maurage', 'simon'))


# model specification
mod = brm(real ~ corpus + (1|sound) + (1|id), data = df, family = 'bernoulli',  prior = set_prior('normal(0, 3)'),
  iter = 1000, chains = 4, cores = 4)


# model inspection
summary(mod)
plot(mod)
pp_check(mod)


# get fitted values
newdata = data.frame(corpus = levels(df$corpus))
fit = fitted(mod, newdata = newdata, re_formula = NA) * 100
colnames(fit) = c('fit', 'se', 'lwr', 'upr')
df_plot = cbind(newdata, fit)  # and plot with ggplot


# get contrasts
fit1 = as.data.frame(fitted(mod, newdata = newdata, re_formula = NA, summary = FALSE))
colnames(fit1) = newdata$corpus


ut_vs_rest = fit1$ut - (fit1$belin + fit1$cordaro + fit1$hawk + fit1$lima + fit1$maurage + fit1$simon) / 6
quantile(ut_vs_rest, probs = c(.5, .025, .975))


hawk_vs_rest = fit1$hawk - (fit1$belin + fit1$cordaro + fit1$lima + fit1$maurage + fit1$simon) / 5
quantile(hawk_vs_rest, probs = c(.5, .025, .975))


ut_vs_hawk = fit1$ut - fit1$hawk
quantile(ut_vs_hawk, probs = c(.5, .025, .975))
```

...but a few things are worth learning first

- What is MCMC?
- What is a prior and why should I use one?
- Does my model make sense? (convergence, posterior prediction)
- Fitted values (brms::fitted.brmsfit) vs. posterior prediction (brms::predict.brmsfit)
- Shrinkage (e.g. brms::horseshoe)
- How can I extract and customize STAN code (brms::stancode)?

How to learn all this





Download

Slides, dataset, R code, and the original article:

http://cogsci.se/publications.html

(the link at the very bottom called
anikin_bayes@lund_2018.zip)

Thank you!