Recognition Behavioral Data Analysis

Andrew Graves

May 22, 2019

Load Packages

```
library(tidyverse)
library(lme4)
library(effects)
# Requires additional packages installed: psycho, EGAnet
```

Initialize vectors

```
# User must set working directory where subject data folder
# are located on your local machine as wd:
wd <- "C:/Users/Andrew Graves/Documents/University of Virginia/Research/first_author/recog_beh_data"
setwd(wd)

subj_id <- 1:269
recog_vector <- seq(1, 3)
data_grid <- expand.grid(subj_id, recog_vector)

# Orthogonalize contrasts
options(contrasts = c("contr.sum", "contr.poly"))

# Set ggplot theme and colors
theme_set(theme_light())
theme_update(text = element_text(family = "serif", size = 20),
    plot.title = element_text(hjust = 0.5), plot.caption = element_text(hjust = 0),
    axis.title.y = element_text(margin = margin(t = 0, r = 20,
        b = 0, l = 0)), strip.background = element_rect(fill = "black"))

my_purple <- rgb(51, 0, 102, maxColorValue = 255)
my_green <- rgb(0, 102, 51, maxColorValue = 255)
```

Read data

```
read_recog_data <- function(i, j) {

    setwd(wd)
    subj_dir <- paste0(i, "/")
    subj_file <- paste0(i, "_recognition", j, ".txt")

    if (file.exists(paste0(subj_dir, subj_file))) {
```

```
        i %>% as.character() %>% setwd()

        data_file <- subj_file %>% read_delim(delim = " ")
        return(data_file)

    }
}

# map_data <- map2(data_grid$Var1, data_grid$Var2,
# read_recog_data) raw_data <- do.call(rbind, map_data)
# setwd(wd) save(raw_data, file = 'raw_data')
load("raw_data")
```

Tidy data for modelling

```
raw_data$stim_class <- raw_data$stimType %>% str_detect("face") %>%
    if_else("face", "flower")

raw_data$stim_mem <- raw_data$stimType %>% str_detect("Old") %>%
    if_else("old", "new")

raw_data$stim_id <- paste0(raw_data$stim_class, raw_data$stimNumber)

recog_data <- raw_data %>% filter(ac != 99, rt > 250) %>% mutate(log_rt = log(rt/1000)) %>%
    select(-hand, -resp, -stimNumber, -stimType) %>% rename(subj = subNo,
    resp = respType)

factor_data <- recog_data %>% select(subj, resp, ac, stim_class,
    stim_mem, stim_id) %>% map_dfr(factor)

model_data <- recog_data %>% select(-subj, -resp, -ac, -stim_class,
    -stim_mem, -stim_id) %>% bind_cols(factor_data)
```

Calculate Cronbach's alpha for reliability

```
stim_classes <- levels(model_data$stim_class)

get_alpha <- function(stim_class_arg) {
    alpha <- model_data %>% filter(stim_class == stim_class_arg) %>%
        select(subj, ac, stim_id) %>% spread(key = stim_id, value = ac) %>%
        select(-subj) %>% mutate_all(as.character) %>% mutate_all(as.numeric) %>%
        VIM::kNN() %>% select(1:60) %>% # psych::splitHalf(n.sample = 100000)
    psych::alpha(check.keys = TRUE)
}

alpha_list <- map(stim_classes, get_alpha)
paste("Face alpha:", format(alpha_list[[1]]$total$raw_alpha,
    digits = 2))
```

```
## [1] "Face alpha: 0.84"
```

```
paste("Flower alpha:", format(alpha_list[[2]]$total$raw_alpha,
    digits = 2))
```

```
## [1] "Flower alpha: 0.86"
```

Compute accuracy across blocks

```
model_data %>% group_by(block) %>% summarize(block_accuracy = format(mean(as.numeric(ac) -
    1), digits = 2, nsmall = 2))
```

```
## # A tibble: 3 x 2
##   block block_accuracy
##   <int> <chr>
## 1     1 0.70
## 2     2 0.70
## 3     3 0.69
```

Get signal statistsics

```
signal_stats <- recog_data %>% group_by(subj, stim_class) %>%
    summarize(ht = sum(resp == "ht"), fa = sum(resp == "fp"),
        ms = sum(resp == "fn"), cr = sum(resp == "cr"))

face_stats <- signal_stats %>% filter(stim_class == "face")

flower_stats <- signal_stats %>% filter(stim_class == "flower")

flower_indices <- psycho::dprime(flower_stats$ht, flower_stats$fa,
    flower_stats$ms, flower_stats$cr, n_targets = 60, n_distractors = 60) %>%
    as_tibble()
colnames(flower_indices) <- paste0("flower_", colnames(flower_indices))

face_indices <- psycho::dprime(face_stats$ht, face_stats$fa,
    face_stats$ms, face_stats$cr, n_targets = 60, n_distractors = 60) %>%
    as_tibble()
colnames(face_indices) <- paste0("face_", colnames(face_indices))

indices <- face_indices %>% bind_cols(flower_indices)
```

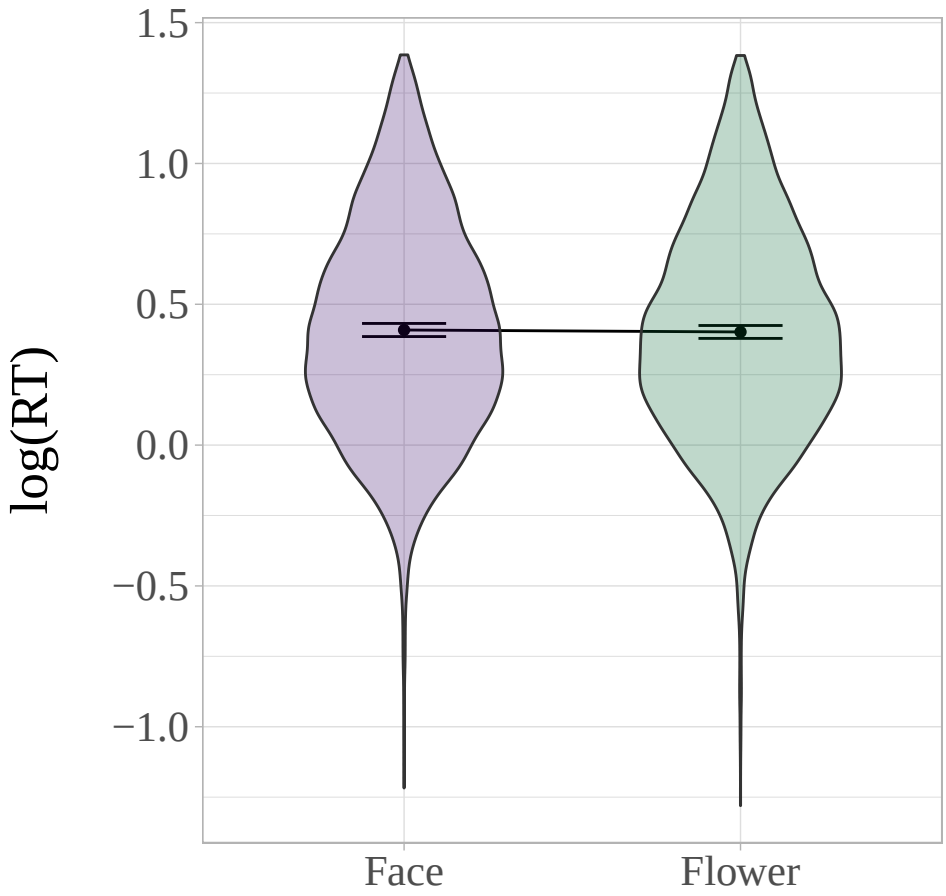
Model data

```
fixed_vec <- c("1", "stim_class", "stim_mem", "stim_class+stim_mem",
    "stim_class*stim_mem")
old_new_vec <- c("1", "stim_class")

old_item_data <- model_data %>% filter(stim_mem == "old") %>%
    mutate(old = ac)

new_item_data <- model_data %>% filter(stim_mem == "new") %>%
    mutate(old = if_else(ac == 1, 0, 1))
```

```r
# log(RT)
run_lmer <- function(i) {

    lmer(as.formula(paste("log_rt ~ ", i, "+ (stim_class|subj) + (1|stim_id)")),
        data = model_data, REML = FALSE)

}

# lmer_models <- map(fixed_vec, run_lmer) save(lmer_models,
# file = 'lmer_models')
load("lmer_models")

# Accuracy
run_glmer <- function(i) {

    glmer(as.formula(paste("ac ~ ", i, "+ (stim_class|subj) + (1|stim_id)")),
        data = model_data, family = "binomial")

}

# glmer_models <- map(fixed_vec, run_glmer)
# save(glmer_models, file = 'glmer_models')
load("glmer_models")

# Hit rate
run_glmer_old <- function(i) {

    glmer(as.formula(paste("old ~ ", i, "+ (stim_class|subj) + (1|stim_id)")),
        data = old_item_data, family = "binomial")

}

# glmer_old_models <- map(old_new_vec, run_glmer_old)
# save(glmer_old_models, file = 'glmer_old_models')
load("glmer_old_models")

# False alarm rate
run_glmer_new <- function(i) {

    glmer(as.formula(paste("old ~ ", i, "+ (stim_class|subj) + (1|stim_id)")),
        data = new_item_data, family = "binomial")

}

# glmer_new_models <- map(old_new_vec, run_glmer_new)
# save(glmer_new_models, file = 'glmer_new_models')
load("glmer_new_models")
```

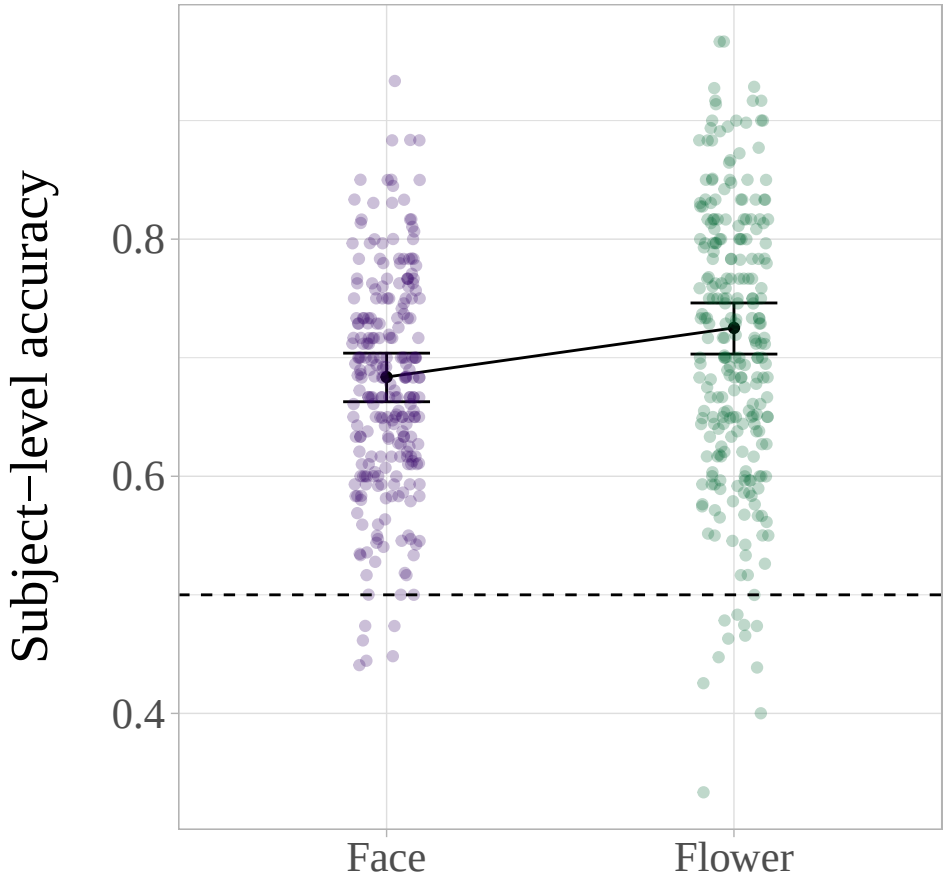
Plot model data

```r
get_effects <- function(i) {

    effect_data <- data.frame(effect("stim_class", i))

    effect_data %>% ggplot(aes(x = stim_class, y = fit, group = 1)) +
        geom_point() + geom_line() + geom_errorbar(aes(ymin = lower,
        ymax = upper), width = 0.25) + theme(legend.position = "none") +
        labs(x = "") + scale_x_discrete(labels = c("Face", "Flower"))

}

# log(RT)
get_effects(lmer_models[[2]]) + geom_violin(data = model_data,
    aes(y = log_rt, group = stim_class, fill = stim_class), alpha = 0.25,
    width = 0.6) + scale_fill_manual(values = c(my_purple, my_green)) +
    labs(y = "log(RT)")
```




```r
# Accuracy
subj_ac <- model_data %>% mutate(ac_num = as.numeric(as.character(ac))) %>%
    group_by(subj, stim_class) %>% summarize(mean_ac = mean(ac_num))

get_effects(glmer_models[[2]]) + geom_jitter(data = subj_ac,
    aes(y = mean_ac, color = stim_class), width = 0.1, alpha = 0.25) +
```

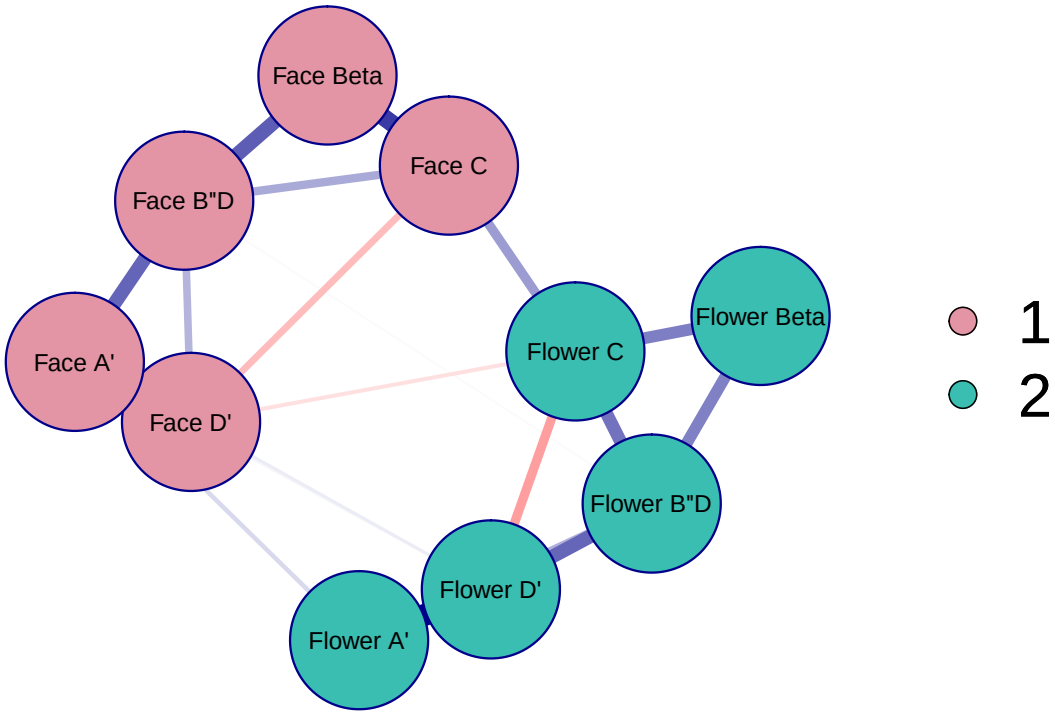
```
    scale_color_manual(values = c(my_purple, my_green)) + geom_hline(yintercept = 0.5,
    linetype = 2) + labs(y = "Subject-level accuracy")
```



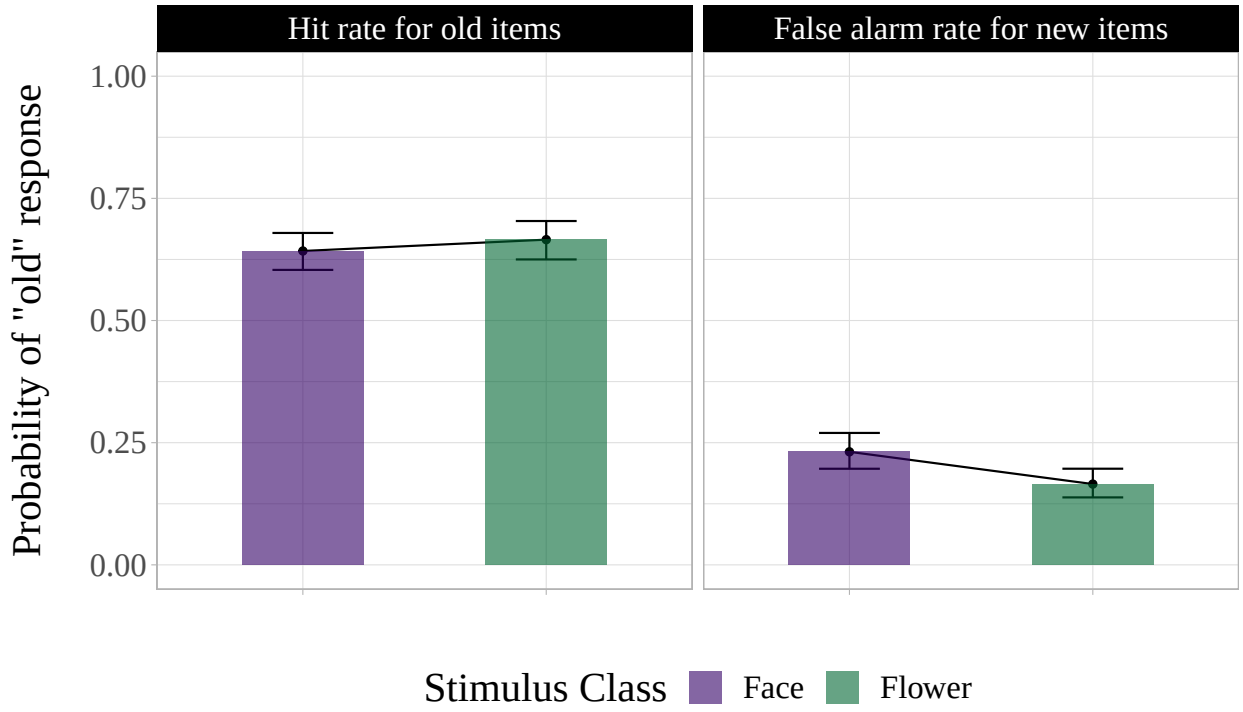

```
# Compute BF
approx_bf <- function(i) {

    exp((bic_0 - i)/2)

}

lmer_bic <- map(lmer_models, BIC)
bic_0 <- lmer_bic[[1]]
lmer_bf <- map(lmer_bic, approx_bf)
stim_lmer_bf <- lmer_bf[[2]]

glmer_bic <- map(glmer_models, BIC)
bic_0 <- glmer_bic[[1]]
glmer_bf <- map(glmer_bic, approx_bf)
stim_glmer_bf <- glmer_bf[[2]]

glmer_old_bic <- map(glmer_old_models, BIC)
bic_0 <- glmer_old_bic[[1]]
glmer_bf <- map(glmer_old_bic, approx_bf)
```

```
stim_glmer_old_bf <- glmer_bf[[2]]

glmer_new_bic <- map(glmer_new_models, BIC)
bic_0 <- glmer_new_bic[[1]]
glmer_bf <- map(glmer_new_bic, approx_bf)
stim_glmer_new_bf <- glmer_bf[[2]]
```

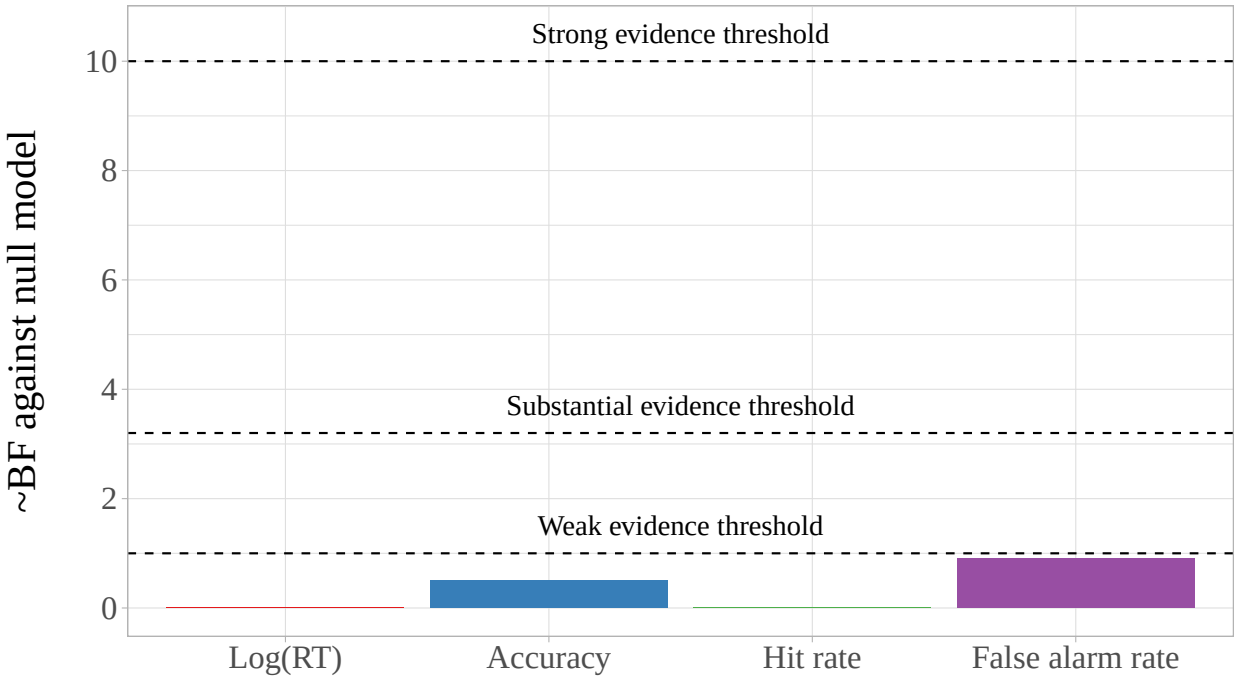
```
# EGA
labels <- indices %>% names() %>% str_replace_all(c(prime = "'",
    `_` = " ", pp = "''")) %>% str_to_title()

indices %>% EGAnet::EGA(model = "glasso", plot.EGA = FALSE) %>%
    plot(vsize = 12.75, border.width = 1.5, labels = labels,
        groups = groups, label.scale = FALSE, legend.mode = "groups",
        legend.cex = 1.25, palette = "pastel", theme = "Borkulo")
```




```
# Hit rate/ false alarm rate
data.frame(effect("stim_class", glmer_old_models[[2]])) %>% bind_rows(data.frame(effect("stim_class",
    glmer_new_models[[2]]))) %>% mutate(stim_mem = rep(c("Hit rate for old items",
    "False alarm rate for new items"), each = 2)) %>% ggplot(aes(x = stim_class,
    y = fit, group = 1)) + geom_point() + geom_line() + geom_errorbar(aes(ymin = lower,
    ymax = upper), width = 0.25) + scale_x_discrete(labels = c("Face",
    "Flower")) + geom_bar(stat = "identity", group = 1, alpha = 0.6,
    width = 0.5, aes(fill = stim_class)) + facet_wrap(~fct_rev(stim_mem)) +
    scale_fill_manual(labels = c("Face", "Flower"), values = c(my_purple,
        my_green)) + ylim(0, 1) + labs(x = "", y = "Probability of \"old\" response",
    fill = "Stimulus Class") + theme(axis.text.x = element_blank(),
    legend.position = "bottom")
```




```
# ~BF plot
model_vec <- factor(c("Log(RT)", "Accuracy", "Hit rate", "False alarm rate"),
    levels = c("Log(RT)", "Accuracy", "Hit rate", "False alarm rate"))

threshold_data <- data.frame(labels = c("Weak evidence threshold",
    "Substantial evidence threshold", "Strong evidence threshold"),
    x_pos = 2.5, y_pos = c(1.5, 3.7, 10.5))

rbind(stim_lmer_bf, stim_glmer_bf, stim_glmer_old_bf, stim_glmer_new_bf) %>%
    as_tibble() %>% cbind(model_vec) %>% ggplot(aes(x = model_vec,
    y = V1, fill = model_vec)) + geom_bar(stat = "identity") +
    geom_hline(yintercept = c(1, 3.2, 10), linetype = 2) + geom_text(data = threshold_data,
    aes(label = labels, x = x_pos, y = y_pos), inherit.aes = FALSE,
    family = "serif", size = 5) + scale_y_continuous(breaks = seq(0,
    10, 2)) + scale_fill_brewer(palette = "Set1") + labs(x = "",
    y = "~BF against null model") + theme(legend.position = "none")
```


Strong evidence threshold
Substantial evidence threshold
Weak evidence threshold
~BF against null model
10
8
6
4
2
0
Log(RT)
Accuracy
Hit rate
False alarm rate