Familiarity Facilitates Feature-based Face Processing Slope and Set Size 1 estimates

Matteo Visconti di Oleggio Castello, Kelsey G. Wheeler, Carlo Cipolli, M. Ida Gobbini

Contents

Setup 1

Functions 2

Test Functions 3

Bootstrapping of the data 5

Compute average estimates and CIs 6

Set Size 1 Estimates 8

This R markdown file will generate estimates of intercept, slopes, and Set Size 1 for the models on reaction times.

Setup

Load some useful libraries, return version information, and load the data.

```
# return version information
version
```

```
##                _
## platform       x86_64-apple-darwin13.4.0
## arch           x86_64
## os             darwin13.4.0
## system         x86_64, darwin13.4.0
## status
## major          3
## minor          2.3
## year           2015
## month          12
## day            10
## svn rev        69752
## language       R
## version.string R version 3.2.3 (2015-12-10)
## nickname       Wooden Christmas-Tree
```

```
packages <- c('dplyr',
              'ggplot2',
              'doParallel',
```

```
              'foreach',
              'assertthat',
              'knitr',
              'broom')

for (package in packages) {
  require(package, character.only=T)
  cat(paste(package, packageVersion(package), '\n'))
}
```

```
## dplyr 0.4.3
## ggplot2 2.1.0
## doParallel 1.0.10
## foreach 1.4.3
## assertthat 0.1
## knitr 1.12.3
## broom 0.4.0
```

```
data <- read.csv('data.csv')
# set order of levels for plotting
data$orientation <- factor(data$orientation,
                           levels=c('Upright', 'Inverted'))
data$target_presence <- factor(data$target_presence,
                               levels=c('Target Present', 'Target Absent'))
# get correct trials
data_correct <- data %>% filter(correct == 1)
```

Set up some variables that will be used later.

```
nproc <- 4  # change this to use more/less processors for parallel use
seed <- 42  # seed for rng to obtain reproducible results in different runs
alpha <- .05  # significance level for confidence intervals
nbs <- 10000  # number of bootstrapping repetitions
```

Functions

Set up some functions that will be used later.

```
# run linear regression on a dataframe
run_lm <- function(df) {
    tidy(lm(RT ~ set_size, data=df))
}

# this function computes the following estimates for each bootstrapped sample
# 1. average intercept/slope across subjects
# 2. average effect size for familiarity on the intercept
# 3. average effect size for familiarity on the slopes
compute_estimates <- function(df) {
    lmsubj <-
        df %>%
        group_by(familiarity, target_presence, orientation, subid) %>%
```

```
        do(run_lm(.)) %>%
        select(-std.error, -statistic, -p.value)

    # compute average of the estimates
    lmsubj_byterm <-
        lmsubj %>%
        group_by(familiarity, target_presence, orientation, term) %>%
        summarise(estimate=mean(estimate))

    # compute average effect size for familiarity on the intercept
    lmsubj_int <-
        lmsubj %>%
        filter(term == "(Intercept)") %>%
        ungroup() %>%
        arrange(subid, target_presence, orientation, familiarity) %>%
        group_by(subid, target_presence, orientation)

    diffint <-
        lmsubj_int %>%
        mutate(diffint=(estimate - lag(estimate))) %>%
        filter(!is.na(diffint)) %>%
        select(-estimate, -familiarity) %>%
        group_by(target_presence, orientation) %>%
        summarise(diffint=mean(diffint))

    # compute average effect size for familiarity on the slope
    lmsubj_slope <-
        lmsubj %>%
        filter(term == 'set_size') %>%
        ungroup() %>%
        arrange(subid, target_presence, orientation, familiarity) %>%
        group_by(subid, target_presence, orientation)

    diffslope <-
        lmsubj_slope %>%
        mutate(diffslope=(estimate - lag(estimate))) %>%
        filter(!is.na(diffslope)) %>%
        select(-estimate, -familiarity) %>%
        group_by(target_presence, orientation) %>%
        summarise(diffslope=mean(diffslope))

    return(list(lmsubj_byterm, diffint, diffslope))
}
```

Test Functions

Test the function `compute_estimates`:

```
# make up some data
df_fake <- expand.grid(
  set_size=sort(unique(data$set_size)),
  familiarity=unique(data$familiarity),
  target_presence=unique(data$target_presence),
```

```
  orientation=unique(data$orientation)
)

# create data with known intercept and slope
df_fake$intercept <- sort(rep(seq(500, 1200, 100), 3))
df_fake$slope <- sort(rep(seq(10, 80, 10), 3))
df_fake$RT <- df_fake$set_size * df_fake$slope + df_fake$intercept
df_fake$subid <- 'sub001'

suppressWarnings(out <- compute_estimates(df_fake))

# check that we obtain what we expected
estimates <- out[[1]]
nestimates <- nrow(estimates)

for (i in 1:nestimates) {
  this_estimate <- estimates[[i, 'estimate']]
  this_familiarity <- estimates[[i, 'familiarity']]
  this_target_presence <- estimates[[i, 'target_presence']]
  this_orientation <- estimates[[i, 'orientation']]

  fake_value <- filter(df_fake,
                       set_size == 2,
                       familiarity == this_familiarity,
                       target_presence == this_target_presence,
                       orientation == this_orientation)

  if (estimates[[i, 'term']] == 'set_size') {
    assert_that(isTRUE(all.equal(this_estimate, fake_value$slope)))
  } else {
    assert_that(isTRUE(all.equal(this_estimate, fake_value$intercept)))
  }
}

# check that the difference on the intercepts is correct
diffint <- out[[2]]
ndiffint <- nrow(diffint)

for (i in 1:ndiffint) {
  # get values from original ds
  this_target_presence <- as.character(diffint[i, ]$target_presence)
  this_orientation <- as.character(diffint[i, ]$orientation)
  fake_values <-
    df_fake %>%
    filter(target_presence == this_target_presence,
           orientation == this_orientation)

  familiar_value <-
    fake_values %>%
    filter(familiarity == 'Familiar',
           set_size == 2) %>%
    select(intercept) %>%
    as.numeric()
```

```
  stranger_value <-
    fake_values %>%
    filter(familiarity == 'Stranger',
           set_size == 2) %>%
    select(intercept) %>%
    as.numeric()

  this_diffint <- diffint[[i, 'diffint']]

  assert_that(isTRUE(all.equal(this_diffint,
                               stranger_value - familiar_value)))
}

# check difference of slopes
diffslope <- out[[3]]
ndiffslope <- nrow(diffslope)

for (i in 1:ndiffslope) {
  # get values from original ds
  this_target_presence <- as.character(diffslope[i, ]$target_presence)
  this_orientation <- as.character(diffslope[i, ]$orientation)
  fake_values <-
    df_fake %>%
    filter(target_presence == this_target_presence,
           orientation == this_orientation)

  familiar_value <-
    fake_values %>%
    filter(familiarity == 'Familiar',
           set_size == 2) %>%
    select(slope) %>%
    as.numeric()

  stranger_value <-
    fake_values %>%
    filter(familiarity == 'Stranger',
           set_size == 2) %>%
    select(slope) %>%
    as.numeric()

  this_diffslope <- diffslope[[i, 'diffslope']]

  assert_that(isTRUE(all.equal(this_diffslope,
                               stranger_value - familiar_value)))
}
```

Bootstrapping of the data

Note: this will take a while to run (> 20m on a MacbookAir i7 1.7 GHz)

```
# aux function to add index to dataframe
addindex <- function(df, i) {
    df$index <- i
    df
}
# function to append results to a list without having to loop at the end
mycombine <- function(x, y) {
  list(rbind(x[[1]], y[[1]]),
       rbind(x[[2]], y[[2]]),
       rbind(x[[3]], y[[3]]))
}

cl <- makeCluster(nproc)
registerDoParallel(cl)
# push libraries to workers
clusterCall(cl, function() library(magrittr))
clusterCall(cl, function() library(plyr))
clusterCall(cl, function() library(dplyr))
clusterCall(cl, function() library(broom))
bstrap <- data.frame()
# set seed for reproducibility
set.seed(seed)
bstrap <- foreach(i = 1:nbs, .combine=mycombine) %dopar% {
    tmp <-
        data_correct %>%
        group_by(orientation, target_presence, familiarity,
                 set_size, target_sex, subid) %>%
        sample_frac(1, replace=T) %>%
        group_by(orientation, target_presence, familiarity, set_size) %>%
        compute_estimates

    tmp <- lapply(tmp, addindex, i)
    tmp
}
stopCluster(cl)
```

Compute average estimates and CIs

```
estimates <- bstrap[[1]]
diffint <- bstrap[[2]]
diffslope <- bstrap[[3]]

cis.estimates <-
    estimates %>%
    group_by(familiarity, target_presence, orientation, term) %>%
    summarise(low=quantile(estimate, alpha/2), high=quantile(estimate, 1-alpha/2))

cis.diffint <-
    diffint %>%
    group_by(target_presence, orientation) %>%
    summarise(low=quantile(diffint, alpha/2), high=quantile(diffint, 1-alpha/2))
```

```
cis.diffslope <-
    diffslope %>%
    group_by(target_presence, orientation) %>%
    summarise(low=quantile(diffslope, alpha/2), high=quantile(diffslope, 1-alpha/2))

# get averages
avgs <- compute_estimates(data_correct)
cis.estimates <- merge(avgs[[1]], cis.estimates)
cis.diffint <- merge(avgs[[2]], cis.diffint)
cis.diffslope <- merge(avgs[[3]], cis.diffslope)

# Put everything in a single dataframe,
# add condition variable
cis.diffint$cond <- paste(cis.diffint$target_presence,
                          cis.diffint$orientation, sep=', ')
cis.diffslope$cond <- paste(cis.diffslope$target_presence,
                            cis.diffslope$orientation, sep=', ')

# reorder the factors for cond
order_cond <- c('Target Present, Upright', 'Target Present, Inverted',
                'Target Absent, Upright', 'Target Absent, Inverted')
cis.diffint$cond <- factor(cis.diffint$cond, rev(order_cond))
cis.diffslope$cond <- factor(cis.diffslope$cond, rev(order_cond))

# let's put the two together
cis.diffint$term <- "Intercept"
cis.diffslope$term <- "Slope"

cis.diffint <- rename(cis.diffint, value=diffint)
cis.diffslope <- rename(cis.diffslope, value=diffslope)

cis.diff <- rbind(cis.diffint, cis.diffslope)
```

Raw estimates

```
kable(cis.estimates, digits=0)
```

familiarity	target_presence	orientation	term	estimate	low	high
Familiar	Target Absent	Inverted	(Intercept)	523	498	548
Familiar	Target Absent	Inverted	set_size	210	203	216
Familiar	Target Absent	Upright	(Intercept)	440	419	462
Familiar	Target Absent	Upright	set_size	184	178	189
Familiar	Target Present	Inverted	(Intercept)	579	548	610
Familiar	Target Present	Inverted	set_size	121	112	130
Familiar	Target Present	Upright	(Intercept)	545	522	568
Familiar	Target Present	Upright	set_size	87	80	94
Stranger	Target Absent	Inverted	(Intercept)	519	495	544
Stranger	Target Absent	Inverted	set_size	237	230	243
Stranger	Target Absent	Upright	(Intercept)	458	437	480
Stranger	Target Absent	Upright	set_size	210	204	215
Stranger	Target Present	Inverted	(Intercept)	666	635	698
Stranger	Target Present	Inverted	set_size	116	107	125

familiarity	target_presence	orientation	term	estimate	low	high
Stranger	Target Present	Upright	(Intercept)	574	548	600
Stranger	Target Present	Upright	set_size	108	101	116

Effect size on intercept and slopes

```
kable(select(cis.diff, -cond), digits=0)
```

target_presence	orientation	value	low	high	term
Target Absent	Inverted	-4	-38	32	Intercept
Target Absent	Upright	18	-13	49	Intercept
Target Present	Inverted	88	43	131	Intercept
Target Present	Upright	29	-5	64	Intercept
Target Absent	Inverted	27	18	36	Slope
Target Absent	Upright	26	19	34	Slope
Target Present	Inverted	-4	-17	8	Slope
Target Present	Upright	22	11	32	Slope

Set Size 1 Estimates

Compute Set Size 1 estimates and 95% bootstrapped confidence intervals.

```
avg_ss1 <-
  cis.estimates %>%
  select(-low, -high) %>%
  group_by(familiarity, target_presence, orientation) %>%
  mutate(set_size1=estimate + lag(estimate)) %>%
  select(-estimate, -term) %>%
  na.omit()

bstrap.ss1 <-
  estimates %>%
  group_by(familiarity, target_presence, orientation, index) %>%
  mutate(set_size1=estimate + lag(estimate)) %>%
  na.omit() %>%
  select(-estimate, -term)

cis.ss1 <-
  bstrap.ss1 %>%
  group_by(familiarity, target_presence, orientation) %>%
  summarise(low=quantile(set_size1, alpha/2),
            high=quantile(set_size1, 1 - alpha/2))

cis.ss1 <- merge(avg_ss1, cis.ss1)

kable(cis.ss1, digits=0)
```

familiarity	target_presence	orientation	set_size1	low	high
Familiar	Target Absent	Inverted	733	714	752
Familiar	Target Absent	Upright	623	607	640
Familiar	Target Present	Inverted	699	677	722
Familiar	Target Present	Upright	632	615	649
Stranger	Target Absent	Inverted	756	737	775
Stranger	Target Absent	Upright	668	652	685
Stranger	Target Present	Inverted	783	759	806
Stranger	Target Present	Upright	683	663	702

Compute effect size on Set Size 1 estimates for Stranger vs. Familiar faces.

```
avg_ss1_es <-
  avg_ss1 %>%
  ungroup() %>%
  arrange(target_presence, orientation, familiarity) %>%
  group_by(target_presence, orientation) %>%
  mutate(set_size1_es=set_size1 - lag(set_size1)) %>%
  select(-familiarity, -set_size1) %>%
  na.omit()

bstrap.ss1_es <-
  bstrap.ss1 %>%
  ungroup() %>%
  arrange(index, target_presence, orientation, familiarity) %>%
  group_by(index, target_presence, orientation) %>%
  mutate(set_size1_es=set_size1 - lag(set_size1)) %>%
  select(-familiarity, -set_size1) %>%
  na.omit()

cis.ss1_es <-
  bstrap.ss1_es %>%
  group_by(target_presence, orientation) %>%
  summarise(low=quantile(set_size1_es, alpha/2),
            high=quantile(set_size1_es, 1 - alpha/2))

cis.ss1_es <- merge(avg_ss1_es, cis.ss1_es)

kable(cis.ss1_es, digits=0)
```

target_presence	orientation	set_size1_es	low	high
Target Absent	Inverted	23	-4	50
Target Absent	Upright	45	21	68
Target Present	Inverted	83	50	116
Target Present	Upright	51	26	77

Now compute difference of effect sizes across upright and inverted.

```
avg_ss1_es_upinv <-
  avg_ss1_es %>%
```

```
  ungroup() %>%
  arrange(target_presence, orientation) %>%
  group_by(target_presence) %>%
  mutate(set_size1_es=set_size1_es - lag(set_size1_es)) %>%
  na.omit() %>%
  select(-orientation)

cis.ss1_es_ss1 <-
  bstrap.ss1_es %>%
  ungroup() %>%
  arrange(index, target_presence, orientation) %>%
  group_by(index, target_presence) %>%
  mutate(set_size1_es=set_size1_es - lag(set_size1_es)) %>%
  na.omit() %>%
  select(-orientation) %>%
  group_by(target_presence) %>%
  summarise(low=quantile(set_size1_es, alpha/2),
            high=quantile(set_size1_es, 1 - alpha/2))

cis.ss1_es_ss1 <- merge(avg_ss1_es_upinv, cis.ss1_es_ss1)

kable(cis.ss1_es_ss1, digits=0)
```

target_presence	set_size1_es	low	high
Target Absent	-22	-57	14
Target Present	33	-10	74