Pre-registered code

Heather Kappes, Crystal Hall, Rebecca Johnson, Simone Zhang

12/28/2020

Contents

Load packages and setup

```
library(xlsx)
library(dplyr)
library(stringr)
library(haven)
library(ggplot2)
library(tidyverse)
library(data.table)
library(viridis)
library(kableExtra)
library(qualtRics)
library(car)
```

```
theme_new <- function(base_size = 16, base_family = "Helvetica"){
  theme_bw(base_size = base_size, base_family = base_family) %+replace%
    theme(
      panel.grid = element_blank(),
      panel.border = element_rect(fill = NA, colour = "black", size=1),
      panel.background = element_rect(fill = "white", colour = "black"),
      strip.background = element_rect(fill = NA),
      axis.text.x = element_text(color = "black"),
      axis.text.y = element_text(color = "black")
    )
}


code_agg_scales <- function(data, prefix_string,
                            verbose = FALSE,
                            FUN){

  ## get all columns with that prefix
  cols_withprefix = grep(sprintf("^%s", prefix_string),
                         colnames(data),
                         value = TRUE)
  if(verbose) print(sprintf("Coding scale based on: %s", paste(cols_withprefix,
                                                               collapse = ";")))
  ## apply coding function to all columns with that pattern
  data[, cols_withprefix] = apply(data[, cols_withprefix], 2, FUN)
  ## sum those columns and average by the number of columns considered
  sum_across = rowSums(data[, cols_withprefix])
  avg_across = (1/length(cols_withprefix)) * sum_across
  # return the vector of averages
  return(avg_across)
}
```

Load and clean data

```
# Load qualtrics data
raw_data= read_survey("../data/Beliefs about funding models _rjrevis_December 1, 2020_11.06.csv")
```

```
##
## -- Column specification --------------------------------------------------------
## cols(
##   .default = col_character(),
##   StartDate = col_datetime(format = ""),
##   EndDate = col_datetime(format = ""),
##   Progress = col_double(),
##   `Duration (in seconds)` = col_double(),
##   Finished = col_logical(),
##   RecordedDate = col_datetime(format = ""),
##   RecipientLastName = col_logical(),
##   RecipientFirstName = col_logical(),
##   RecipientEmail = col_logical(),
```

```
##   ExternalReference = col_logical(),
##   LocationLatitude = col_double(),
##   LocationLongitude = col_double(),
##   w_rankfair_1 = col_double(),
##   w_rankfair_2 = col_double(),
##   w_rankfair_3 = col_double(),
##   m_rankfair_1 = col_double(),
##   m_rankfair_2 = col_double(),
##   m_rankfair_3 = col_double(),
##   `Create New Field or Choose From Dropdown...` = col_logical()
## )
## i Use `spec()` for the full column specifications.
```

```
# Load prolific dem
prol_dem = read.csv("../data/prolific_export_5fc0ff8b87f2d307d12a6324.csv")

print(sprintf("Out of %s unique ids in prolific df, %s are found in qualtrics df",
        length(unique(prol_dem$participant_id)),
        length(intersect(unique(prol_dem$participant_id),
                    unique(raw_data$prol_id)))))

prol_notfound = setdiff(prol_dem$participant_id,
                        raw_data$prol_id)

prol_dem = prol_dem %>%
        mutate(is_notfound = ifelse(participant_id %in% prol_notfound, TRUE, FALSE),
               is_missing_completetime = ifelse(completed_date_time == "", TRUE, FALSE))

## see that mostof the prolific not found are missing a complete day/time
## but there's also 1 missing completedatetime who's found in qualtrics
table(prol_dem$is_notfound, prol_dem$is_missing_completetime)

## rename prolific cols to distinguish b/t qualtrics cols
colnames(prol_dem) = sprintf("%s_prolific",
                             colnames(prol_dem))

## construct combined timing variables
time_var = grep("\\_time",
                colnames(raw_data),
                value = TRUE)
raw_data[, time_var][is.na(raw_data[,
                    time_var])] <- ""
raw_data$timetrad_raw = apply(raw_data[, time_var],
             1,
             function(x) paste(x, collapse = ""))

## construct label for all choice combos
fr_cols = grep("fr\\_1$", colnames(raw_data),
               value = TRUE)
raw_data$choice = gsub("\\_fr_1",
                       "",
                names(raw_data[, fr_cols])[max.col(!is.na(raw_data[, fr_cols]), "first")])
```

```r
# Create flags
## from looking at labels:
### [w/m]_rankfair_1 == lottery
### [w/m]_rankfair_2 == fcfs
### [w/m]_rankfair_3 == points
### and then values within are ranking
raw_data = raw_data %>%
        mutate(is_prol_r = ifelse(grepl("^5", prol_id),
                                  TRUE, FALSE),
               is_any_barriers = ifelse(Cond %in% c("Min", "Wom"), TRUE,
                                  FALSE),
                points_rank= case_when(
                          !is.na(w_rankfair_3) ~ w_rankfair_3,
                          !is.na(m_rankfair_3) ~ m_rankfair_3),
                lottery_rank = case_when(
                          !is.na(w_rankfair_1) ~ w_rankfair_1,
                          !is.na(m_rankfair_1) ~ m_rankfair_1),
                fcfs_rank = case_when(
                          !is.na(w_rankfair_2) ~ w_rankfair_2,
                          !is.na(m_rankfair_2) ~ m_rankfair_2),
               ## reverse code points so that same direction as binary
               points_rank_rev = 3-points_rank,
                is_points_first = case_when(points_rank == 1 ~ TRUE,
                                    TRUE ~ FALSE),
               timetrad_weeks =
              case_when(grepl("as quickly", timetrad_raw) ~ 0,
                        grepl("2 weeks", timetrad_raw) ~ 2,
                        grepl("1 month", timetrad_raw) ~ 4,
                        grepl("6 weeks", timetrad_raw) ~ 6),

              ## just world scale
              jw_combined = code_agg_scales(raw_data,
                            prefix_string = "jw_scale",
                            FUN = function(x){
                            case_when(grepl("agreement", x) ~ 6,
                                      grepl("disagreement", x) ~ 1,
                                      TRUE ~ as.numeric(x))}))



## left join prolific cols onto raw data
raw_data_wp = merge(raw_data,
                    prol_dem,
                    by.x = "prol_id",
                    by.y= "participant_id_prolific",
                    all.x = TRUE)

## construct flag for missing all free responses
raw_data_wp$is_missing_all_fr = ifelse(rowSums(is.na(raw_data_wp[, fr_cols])) ==
                        length(fr_cols),
                        TRUE, FALSE)

## just for pre-analysis plan, simulate vars we didnt measure
```

```
## in pilot round: race/ethnicity, political affiliation,
## importance of prioritizing group
raw_data_wp <- raw_data_wp %>%
        mutate(is_minority = sample(c(TRUE, FALSE),
                                    nrow(raw_data_wp),
                                    replace = TRUE,
                                    prob = c(0.2, 0.8)),
               political_affil = sample(c("D", "R", "I", "P"),
                                    nrow(raw_data_wp),
                                    replace = TRUE),
               imp_prior = sample(seq(-3, 3, by = 1),
                                    nrow(raw_data_wp),
                                    replace = TRUE))


## subset to analytic df
analytic_df = raw_data_wp %>%
          filter(is_prol_r)

## make colnames lowercase and remove spaces/punctuation
colnames(analytic_df) = gsub("\\s+|\\(|\\)",
                             "",
                             tolower(colnames(analytic_df)))
```

Summarize descriptive stats

Duration/randomization/display logics

```
# Distribution of duration across conditions
## median duration is 6 minutes
## slightly lower for the people who read about historical barriers
quantile(analytic_df$durationinseconds)
analytic_df %>%
  group_by(is_any_barriers) %>%
  summarise(quant = paste(quantile(durationinseconds),
                      collapse = "; "))
```

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

```
# Conditions
table(analytic_df$cond)


# Check that free response options were displayed correctly
rank_cols = grep("\\_rank$", colnames(analytic_df),
                 value = TRUE)
check_logic <- function(one_fr){

  ## first filter to those who filled out the fr
  fill_resp = analytic_df %>%
```

```
          filter(!is.na(!!sym(one_fr)))

  ## then, what the first choice should be
  choices = unlist(strsplit(gsub("\\_fr\\_1", "", one_fr),
                            split = ""))
  first = choices[1]
  second = choices[2]

  ## make sure rank matches choices
  ### first
  if((first == "p" & all(fill_resp$points_rank == 1)) |
     (first == "f" & all(fill_resp$fcfs_rank == 1)) |
     (first == "l" & all(fill_resp$lottery_rank == 1))){
    print(sprintf("passed first choice for: %s",
                  one_fr))
  } else{
    print(sprintf("failed first choice for: %s",
                  one_fr))
  }

  ### second
  if((second == "p" & all(fill_resp$points_rank == 2)) |
     (second == "f" & all(fill_resp$fcfs_rank == 2)) |
     (second == "l" & all(fill_resp$lottery_rank == 2))){
    print(sprintf("passed second choice for: %s",
                  one_fr))
  } else{
    print(sprintf("failed second choice for: %s",
                  one_fr))
  }
  return(NULL)

}

checking <- lapply(fr_cols, check_logic)
```

Demographics

```
# Demographics
dem_vars = c("employment.status_prolific",
             "sex_prolific",
             "first.language_prolific",
             "is_minority",
             "political_affil")

lapply(analytic_df[, dem_vars],
       function(x) table(x))
```

Descriptives on outcome

```
## first, show raw distribution of ranks
dist_rank_plot <- ggplot(analytic_df, aes(x = points_rank,
                         group = cond,
                         fill = cond)) +
  geom_histogram(bins = 3, position = "dodge") +
  theme_new() +
  scale_fill_viridis(discrete = TRUE) +
  xlab("Ranking of points system (1 = first; 3 = last)") +
  theme(legend.position = c(0.7, 0.7)) +
  labs(fill = "")

## since uneven counts between any barriers and
## no barriers, get proportions
prop_choose = analytic_df %>%
        group_by(points_rank, is_any_barriers) %>%
        summarise(num = n()) %>%
        left_join(analytic_df %>%
                group_by(is_any_barriers) %>%
                summarise(denom = n())) %>%
        mutate(prop_choose = num/denom) %>%
        ungroup()
```

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

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

```
## Joining, by = "is_any_barriers"
```

```
prop_choose_plot <- ggplot(prop_choose, aes(x = points_rank,
                         y = prop_choose,
                         group = is_any_barriers,
                         fill = is_any_barriers)) +
  geom_bar(stat = "identity",
           position = "dodge") +
  theme_new() +
  scale_fill_viridis(discrete = TRUE) +
  xlab("Ranking of points system (1 = first; 3 = last)") +
  theme(legend.position = c(0.7, 0.7)) +
  labs(fill = "Any barriers info?")


## distribution of time by choices
## (p = points; f = fcfs; l = lottery,
## order is the ranking)
time_plot <- ggplot(analytic_df, aes(x = factor(timetrad_weeks))) +
  geom_bar(stat = "count") +
  facet_wrap(~choice) +
  xlab("Number of weeks willing to make biz wait for firstchoice") +
  theme_new()
```

Analytic

Hyp 1: barriers tx causes (1) higher ranking of points system (lower rank) and (2) higher likelihood of ranking points first

```
## continuous regressing rank of points
## reverse coded so that pos coef = rank higher
summary(lm(points_rank_rev ~ is_any_barriers,
           data = analytic_df))

## binary regressing points as first
## use lpm
summary(lm(is_points_first ~ is_any_barriers,
           data = analytic_df))
```

Hyp 2: barriers tx causes respondents to rate it more important to prioritize that group

```
summary(lm(imp_prior ~ is_any_barriers,
           data = analytic_df))
```

Hyp 3: differences in tx effect between minority barriers treatment and women's barrier treatment

1. Filter to respondents randomized to those two
2. Reg is comparing those two

```
## when women is the tx, slightly more likely
## to rank points higher/first than when minority is tx
summary(lm(points_rank_rev ~ cond,
           data = analytic_df %>%
             filter(cond != "No_info")))

summary(lm(is_points_first ~ cond,
           data = analytic_df %>%
             filter(cond != "No_info")))
```

Hyp 4: interaction between treatment and respondent's status in underserved group

Similar to above, filters to those randomized to one of the barrier conditions

```
barriers_r <- analytic_df %>%
        filter(cond != "No_info") %>%
        mutate(is_woman = ifelse(sex_prolific == "Female",
                                 TRUE, FALSE))
```

```r
## women
summary(lm(points_rank_rev ~ cond*is_woman,
           data = barriers_r))

## minority
summary(lm(points_rank_rev ~ cond*is_minority,
           data = barriers_r))
```

Hyp 5: interaction b/t other respondent demographics and treatment

Uses all respondents and pools the barriers treatment into single treatment

```r
## political affiliation
summary(lm(points_rank_rev ~ is_any_barriers*political_affil,
           data = analytic_df))

## just world
summary(lm(points_rank_rev ~ is_any_barriers*jw_combined,
           data = analytic_df))
```