Candidate Data

Lei Shi

2023-11-28

```
library(dplyr)
library(ggplot2)
library(tidyverse)
library(glmnet)
setwd("/Users/leishi/Documents/Research/ForwardScreening")
source("auxillary_functions.R")
```

Analyze candidate experiment

The second dataset is taken from the paper: https://www.cambridge.org/core/journals/political-analysis/article/causal-inference-in-conjoint-analysis-understanding-multidimensional-choices-via-stated-preference-experiments/414DA03BAA2ACE060FFE005F53EFF8C8. The data is again a conjoint survey experiment with fake profiles of US presidental candidates. The respondents give a rating (which is the outcome) for each profile based on eight factors: military, religion, education, income, race, age, gender, profession. In the original data these factors are of multiple levels, but it is easy and reasonble to collapse the factors into binary, except for profession. So I removed profession and only used 7 of the factors. In this case, $K = 7$ and $N = 3456$ after removing missingness.

```
# import dataset
library(haven)
candidate <- read_dta("CandidateExperiment/candidate.dta")
head(candidate, 5)
```

```
## # A tibble: 5 x 11
##   resID     atmilitary atreligion ated    atprof  atinc   atrace  atage  atmale
##   <dbl+lbl> <dbl+lbl>  <dbl+lbl>  <dbl+l> <dbl+l> <dbl+l> <dbl+l> <dbl+> <dbl+l>
## 1 383 [A2N~ 1 [Did No~ 6 [Mormon] 3 [Com~ 6 [Car~ 6 [5.1~ 1 [Whi~ 6 [75] 2 [Fem~
## 2 383 [A2N~ 2 [Served] 1 [None]   1 [No ~ 4 [Hig~ 3 [65K] 6 [Asi~ 4 [60] 1 [Mal~
## 3 383 [A2N~ 1 [Did No~ 3 [Cathol~ 5 [Sma~ 5 [Far~ 1 [32K] 2 [Nat~ 5 [68] 2 [Fem~
## 4 383 [A2N~ 2 [Served] 4 [Mainli~ 5 [Sma~ 3 [Doc~ 2 [54K] 1 [Whi~ 6 [75] 1 [Mal~
## 5 383 [A2N~ 2 [Served] 1 [None]   2 [Bap~ 3 [Doc~ 6 [5.1~ 2 [Nat~ 2 [45] 2 [Fem~
## # i 2 more variables: selected <dbl>, rating <dbl>
```

```
factorial_candidate = candidate %>%
  dplyr::select(rating, atmilitary, atreligion, ated, atinc, atrace, atage, atmale) %>%
  mutate(
    atmilitary = atmilitary - 1,
    atreligion = case_when(
      atreligion == 1 ~ 0,
      atreligion != 1 ~ 1
    ),
    ated = case_when(
      ated %in% c(1,5) ~ 0,
      ated %in% c(2,3,4,6) ~ 1
```

```
    ),
    atinc = case_when(
      atinc %in% c(1,2,3,4) ~ 0,
      atinc %in% c(5,6) ~ 1
    ),
    atrace = case_when(
      atrace %in% c(1,5) ~ 0,
      atrace %in% c(2,3,4,6) ~ 1
    ),
    atage = case_when(
      atage %in% c(1,2,3) ~ 0,
      atage %in% c(4,5,6) ~ 1
    ),
    atmale = atmale - 1
  ) %>%
  rename(
    y = rating,
    F1 = atmilitary,
    F2 = atreligion,
    F3 = ated,
    F4 = atinc,
    F5 = atrace,
    F6 = atage,
    F7 = atmale
  )  %>% drop_na()
head(factorial_candidate, 5)
```

```
## # A tibble: 5 x 8
##       y    F1    F2    F3    F4    F5    F6    F7
##   <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
## 1 0.333     0     1     1     1     0     1     1
## 2 0.5       1     0     0     0     1     1     0
## 3 0.667     0     1     0     0     1     1     1
## 4 0.667     1     1     0     0     0     1     0
## 5 0.333     1     0     1     1     1     0     1
```

```
num_pop = nrow(factorial_candidate)
num_pop
```

```
## [1] 3456
```

LASSO

We again study forward screening + LASSO + strong heredity:

```
source("auxillary_functions.R")
forward.select.opts <- list()

# heredity.proceed.init <- list()
heredity.proceed.init <- list(
  criterion = "strong"
)

# model.selection.init
model.selection.opts <- list(
```

```
  correction.type = "hc0",
  robust.flag = TRUE,
  best_lambda_choice = "lambda.min"
)
model.selection.init <- list(
  method = "LASSO",
  model.selection.opts = model.selection.opts
)

forward.select(factorial_data = factorial_candidate,
               alpha.vec = rep(0.20/7, 7),
               wt = NULL,
               level = 7L,
               forward.select.opts = forward.select.opts,
               heredity.proceed.init = heredity.proceed.init,
               model.selection.init = model.selection.init
)
```

```
## $selected_model
##  [1] "F2"       "F3"       "F4"       "F6"       "F7"       "F2.F3"
##  [7] "F3.F6"    "F4.F6"    "F4.F7"    "F6.F7"    "F4.F6.F7"
##
## $fit.model
## $fit.model[[1]]
##
## Call:  glmnet(x = X, y = y, weights = wt, alpha = 1, lambda = best_lambda)
##
##   Df %Dev   Lambda
## 1  5 2.46 0.005247
##
## $fit.model[[2]]
##
## Call:  glmnet(x = X, y = y, weights = wt, alpha = 1, lambda = best_lambda)
##
##   Df %Dev   Lambda
## 1 10 3.28 0.004356
##
## $fit.model[[3]]
##
## Call:  glmnet(x = X, y = y, weights = wt, alpha = 1, lambda = best_lambda)
##
##   Df %Dev    Lambda
## 1 11 3.63 9.605e-05
##
## $fit.model[[4]]
## NULL
##
## $fit.model[[5]]
## NULL
##
## $fit.model[[6]]
## NULL
##
## $fit.model[[7]]
```

```
## NULL
```

A sparse subset of effects are selected. Some main effects, some two-ways and one three-way. High-order effects are zero.

For a comparison, we drop the forward screening and heredity structure:

```
# prepare full data
source("auxillary_functions.R")
data.full <- data.frame(2 * factorial_candidate[, -1] - 1)
data.full <- model.matrix(as.formula(paste0("~.^", 7)), data = data.full)
data.full <- data.frame(data.full[, -1])
data.full <- cbind(factorial_candidate["y"], data.full)

wt = factorial_candidate %>% group_by(across(c(-y))) %>%
  mutate(num=n()) %>% ungroup() %>%
  mutate(inv_num = num_pop/num) %>%
  dplyr::select(inv_num)
wt = wt$inv_num

pre_selected_model = colnames(factor.design(7, trt_group_size = rep(1,2^7), interaction = 7, centering =
# pre_selected_model = paste0("F", 1:7)

model.selection.opts <- setOpts(model.selection.init, "model.selection.opts", list())
model.selection.opts$alpha <- 0.05/7
model.selection.opts$test_model <- paste0("F", 1:7)
model.selection.opts$best_lambda_choice <- "lambda.min"
model.selection.opts$test_model = pre_selected_model



model.selection(data.full,
                pre_selected_model = pre_selected_model,
                wt = wt,
                method = "LASSO",
                model.selection.opts = model.selection.opts)
```

```
## $post_working_model
## [1] "F3"    "F6"    "F1.F4"
##
## $fit.model
##
## Call:  glmnet(x = X, y = y, weights = wt, alpha = 1, lambda = best_lambda)
##
##   Df %Dev  Lambda
## 1  3  1.5 0.01602
```

```
# fit.lm = lm("y~.", data = data.full, weights = wt)
# print(sort(abs(fit.lm$coefficients), decreasing = T))
```

Again, the selected model is hard to interpret.