Immigrants

Lei Shi

2023-11-27

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

Analyze immigrants data

The first dataset is taken from the paper: https://www.cambridge.org/core/journals/political-analysis/article/estimating-and-using-individual-marginal-component-effects-from-conjoint-experiments/FE284F17AB91A18673CC33276FF45D34. It is a conjoint survey experiment, which generates fake profiles of immigrants into US and evaluate the effect of six factors: age, gender, race, education, language, prior trips to US. The outcome is a rate of acceptance by respondents. In this case, $K = 6$ and the total sample size is 27833 after removing missing responses.

Preprocessing of the dataset

```
library(haven)
data_03_conjoint_processed = read_dta("replication_materials (1)/data/data_03_conjoint_processed.dta")
# View(data_03_conjoint_processed)
head(data_03_conjoint_processed, 5)
```

```
## # A tibble: 5 x 15
##   respid scenario profile  rate age_int age_bin   sex_bin    race_cat   race_bin
##    <dbl>    <dbl>   <dbl> <dbl>   <dbl> <dbl+lbl> <dbl+lbl>  <dbl+lbl>  <dbl+lb>
## 1      2        1       1     5      48 1 [40+]   2 [Female] 2 [Black]  1 [Non-~
## 2      2        1       2     5      34 2 [39-]   1 [Male]   3 [Hispan~ 1 [Non-~
## 3      2        2       1     5      49 1 [40+]   2 [Female] 3 [Hispan~ 1 [Non-~
## 4      2        2       2     5      36 2 [39-]   2 [Female] 3 [Hispan~ 1 [Non-~
## 5      2        3       1     5      33 2 [39-]   2 [Female] 1 [White]  2 [Whit~
## # i 6 more variables: educ_cat <dbl+lbl>, educ_bin <dbl+lbl>,
## #   lang_cat <dbl+lbl>, lang_bin <dbl+lbl>, trip_cat <dbl+lbl>,
## #   trip_bin <dbl+lbl>
```

```
factorial_immigrants = data_03_conjoint_processed %>%
  dplyr::select(rate, age_bin, sex_bin, race_bin, educ_bin, lang_bin, trip_bin) %>%
  rename(y = rate,
         F1 = age_bin,
         F2 = sex_bin,
         F3 = race_bin,
         F4 = educ_bin,
         F5 = lang_bin,
```

```
         F6 = trip_bin) %>%
  mutate(F1 = F1 - 1,
         F2 = F2 - 1,
         F3 = F3 - 1,
         F4 = F4 - 1,
         F5 = F5 - 1,
         F6 = F6 - 1) %>%
  drop_na()

head(factorial_immigrants, 5)
```

```
## # A tibble: 5 x 7
##       y    F1    F2    F3    F4    F5    F6
##   <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
## 1     5     0     1     0     0     0     1
## 2     5     1     0     0     1     0     1
## 3     5     0     1     0     1     1     0
## 4     5     1     1     0     0     1     0
## 5     5     1     1     1     1     0     1
```

```
num_pop = nrow(factorial_immigrants)
num_pop
```

```
## [1] 27833
```

```
# forward.select.opts
```

LASSO selection

We first test out our forward screening procedure with LASSO as a selector and strong heredity as the forward structure.

```
# 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_immigrants,
               alpha.vec = rep(0.05/6, 6),
               wt = NULL,
```

```
                level = 6L,
                forward.select.opts = forward.select.opts,
                heredity.proceed.init = heredity.proceed.init,
                model.selection.init = model.selection.init
)
```

```
## $selected_model
##  [1] "F1"    "F2"    "F4"    "F5"    "F6"    "F1.F4" "F1.F6" "F2.F5" "F4.F6"
## [10] "F5.F6"
##
## $fit.model
## $fit.model[[1]]
##
## Call:  glmnet(x = X, y = y, weights = wt, alpha = 1, lambda = best_lambda)
##
##   Df %Dev   Lambda
## 1  5 9.53 0.008696
##
## $fit.model[[2]]
##
## Call:  glmnet(x = X, y = y, weights = wt, alpha = 1, lambda = best_lambda)
##
##   Df %Dev   Lambda
## 1 10 9.62 0.009544
##
## $fit.model[[3]]
##
## Call:  glmnet(x = X, y = y, weights = wt, alpha = 1, lambda = best_lambda)
##
##   Df %Dev  Lambda
## 1 10 9.63 0.00343
##
## $fit.model[[4]]
## NULL
##
## $fit.model[[5]]
## NULL
##
## $fit.model[[6]]
## NULL
```

The results are nice! Selected five main effects and five two-way interactions. The interaction terms follow the strong heredity structure. It actually verifies that in this example, the three-way and higher-order interactions are zero (instead of assuming this to be true).

As a comparison, we perform LASSO directly on the full data, without forward screening and any heredity principle:

```
# prepare full data
data.full <- data.frame(2 * factorial_immigrants[, -1] - 1)
data.full <- model.matrix(as.formula(paste0("~.^", 6)), data = data.full)
data.full <- data.frame(data.full[, -1])
data.full <- cbind(factorial_immigrants["y"], data.full)

wt = factorial_immigrants %>% 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(6, trt_group_size = rep(1,2^6), interaction = 6, centering =

model.selection.opts <- setOpts(model.selection.init, "model.selection.opts", list())
model.selection.opts$alpha <- 0.05/6
model.selection.opts$test_model <- paste0("F", 1:6)
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] "F1"              "F2"              "F4"              "F5"
##  [5] "F6"              "F1.F3"           "F1.F4"           "F1.F6"
##  [9] "F2.F5"           "F3.F4"           "F3.F6"           "F4.F6"
## [13] "F5.F6"           "F1.F2.F5"        "F1.F3.F4"        "F1.F4.F5"
## [17] "F2.F3.F5"        "F2.F3.F6"        "F2.F5.F6"        "F3.F4.F5"
## [21] "F1.F2.F3.F5"     "F1.F2.F4.F5"     "F1.F2.F4.F6"     "F1.F3.F4.F5"
## [25] "F1.F4.F5.F6"     "F1.F2.F3.F4.F5" "F1.F2.F3.F4.F6" "F1.F2.F3.F5.F6"
## [29] "F2.F3.F4.F5.F6"
##
## $fit.model
##
## Call:  glmnet(x = X, y = y, weights = wt, alpha = 1, lambda = best_lambda)
##
##   Df %Dev Lambda
## 1 31 9.72 0.0183
```

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

We see that while a sparse model is selected, for some levels the structure between effects break down. For example, the interaction "F1.F2.F4.F6" is selected but none of its three-way parents belongs to the working model. Nevertheless, the first three layers of the selected effects follow the weak heredity structure, which partially verify the effect heredity principle for lower order effects.

Bonferroni corrected marginal t test

Next, we test out forward screening based on Bonferroni corrected t test with 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
)
model.selection.init <- list(
  method = "Bonferroni",
  model.selection.opts = model.selection.opts
)

forward.select(factorial_data = factorial_immigrants,
               alpha.vec = rep(0.05/6, 6),
               wt = NULL,
               level = 6L,
               forward.select.opts = forward.select.opts,
               heredity.proceed.init = heredity.proceed.init,
               model.selection.init = model.selection.init
)
```

```
## [1] "test: 6 q.tail: 3.19695022913125"
## [1] "test: 3 q.tail: 2.99131611518378"
```

```
## $selected_model
## [1] "F4"    "F5"    "F6"    "F5.F6"
##
## $fit.model
## $fit.model[[1]]
##
## Call:
## lm(formula = form, data = dataIn, weights = wt)
##
## Coefficients:
## (Intercept)           F1           F2           F3           F4           F5
##    5.816164     0.034252     0.049713    -0.002947     0.349178     0.321000
##          F6
##    0.756315
##
##
## $fit.model[[2]]
##
## Call:
## lm(formula = form, data = dataIn, weights = wt)
##
## Coefficients:
## (Intercept)           F4           F5           F6        F4.F5        F4.F6
##    5.816164     0.349178     0.321000     0.756315     0.003804     0.030215
##       F5.F6
##    0.075613
##
##
## $fit.model[[3]]
```

```
## NULL
##
## $fit.model[[4]]
## NULL
##
## $fit.model[[5]]
## NULL
##
## $fit.model[[6]]
## NULL
```

A sparser model is selected, compared to LASSO. But the results are subject to tuning.

For a comparison, the following codes run Bonferroni corrected t tests on the full dataset, without any correction procedure:

```
# prepare full data
data.full <- data.frame(2 * factorial_immigrants[, -1] - 1)
data.full <- model.matrix(as.formula(paste0("~.^", 6)), data = data.full)
data.full <- data.frame(data.full[, -1])
data.full <- cbind(factorial_immigrants["y"], data.full)

wt = factorial_immigrants %>% 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(6, trt_group_size = rep(1,2^6), interaction = 6, centering =

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

```
## [1] "test: 63 q.tail: 3.35499966939159"
```

```
## $post_working_model
## [1] "F4"    "F5"    "F6"    "F5.F6"
##
## $fit.model
##
## Call:
## lm(formula = form, data = dataIn, weights = wt)
##
## Coefficients:
##       (Intercept)                 F1                 F2                 F3
##          5.816164           0.034252           0.049713          -0.002947
##                F4                 F5                 F6             F1.F2
##          0.349178           0.321000           0.756315           0.003423
##             F1.F3              F1.F4              F1.F5              F1.F6
##          0.037113           0.026359          -0.009531           0.018569
##             F2.F3              F2.F4              F2.F5              F2.F6
##          0.009559           0.002332          -0.025018           0.002617
##             F3.F4              F3.F5              F3.F6              F4.F5
##          0.027904           0.015945           0.033581           0.003804
```

```
##            F4.F6            F5.F6         F1.F2.F3         F1.F2.F4
##         0.030215         0.075613        -0.005975        -0.006418
##         F1.F2.F5         F1.F2.F6         F1.F3.F4         F1.F3.F5
##        -0.034686        -0.013688        -0.029949        -0.007805
##         F1.F3.F6         F1.F4.F5         F1.F4.F6         F1.F5.F6
##        -0.014354         0.034432        -0.001723        -0.001597
##         F2.F3.F4         F2.F3.F5         F2.F3.F6         F2.F4.F5
##         0.009419         0.019034        -0.026819        -0.008559
##         F2.F4.F6         F2.F5.F6         F3.F4.F5         F3.F4.F6
##        -0.001397         0.031801        -0.027410        -0.010167
##         F3.F5.F6         F4.F5.F6      F1.F2.F3.F4      F1.F2.F3.F5
##         0.006671         0.001860         0.002563        -0.036583
##      F1.F2.F3.F6      F1.F2.F4.F5      F1.F2.F4.F6      F1.F2.F5.F6
##        -0.003345        -0.051046        -0.022440         0.009173
##      F1.F3.F4.F5      F1.F3.F4.F6      F1.F3.F5.F6      F1.F4.F5.F6
##         0.019447         0.015860        -0.001544        -0.022012
##      F2.F3.F4.F5      F2.F3.F4.F6      F2.F3.F5.F6      F2.F4.F5.F6
##        -0.011751        -0.014273         0.012527         0.018366
##      F3.F4.F5.F6   F1.F2.F3.F4.F5   F1.F2.F3.F4.F6   F1.F2.F3.F5.F6
##         0.018399        -0.029916        -0.029213        -0.020838
##   F1.F2.F4.F5.F6   F1.F3.F4.F5.F6   F2.F3.F4.F5.F6 F1.F2.F3.F4.F5.F6
##         0.012938        -0.013504        -0.024203         0.011201
```

However, the selected model is the same as forward screening. So might not be a good example here showing the superiority of forward screening. Again, this might be subject to the tuning choices.