Analyzing Qualtrics Conjoint Data in R

Priscilla Torres

9/26/2021

Load packages needed

Tidyverse is a syntax in R and a suite of packages that runs off of the pipe operator. For more info see: https://www.tidyverse.org/packages/

```
rm(list=ls()) # clears environment

library(foreign)
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.1 --

## v ggplot2 3.3.3     v purrr   0.3.4
## v tibble  3.1.2     v dplyr   1.0.6
## v tidyr   1.1.3     v stringr 1.4.0
## v readr   1.4.0     v forcats 0.5.1

## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
library(estimatr)
library(cregg)
```

Set the working directory to wherever you have your data stored and load the data as a CSV file. Note that many of the first few columns (variables) are metadata that we will not need for the analysis.

```
getwd() # how you can check what your working directory is set to
```

```
## [1] "/Users/priscilla/Dropbox/GSS Lab/GSS Workshop/Conjoint Tutorial"
```

```
setwd("/Users/priscilla/Dropbox/GSS Lab/GSS Workshop/Conjoint Tutorial") # how you can set your working

conjoint <- read.csv("conjoint.csv") # load csv file/data that we downloaded from Qualtrics

names(conjoint) # let's see which variables Qualtrics included in the data output
```

```
##  [1] "StartDate"
##  [2] "EndDate"
##  [3] "Status"
##  [4] "IPAddress"
##  [5] "Progress"
##  [6] "Duration..in.seconds."
##  [7] "Finished"
##  [8] "RecordedDate"
##  [9] "ResponseId"
## [10] "RecipientLastName"
## [11] "RecipientFirstName"
## [12] "RecipientEmail"
## [13] "ExternalReference"
## [14] "LocationLatitude"
## [15] "LocationLongitude"
## [16] "DistributionChannel"
## [17] "UserLanguage"
## [18] "C1"
## [19] "C2"
## [20] "C3"
## [21] "vers_CBCONJOINT"
## [22] "X6cfb4a95.fd09.4026.95d7.6dd27d85212a.1.1_CBCONJOINT"
## [23] "f2550540.8b2d.4116.bd4f.9e7d67d39926.1.1_CBCONJOINT"
## [24] "fe83fea4.1983.4e36.8201.40eb6016c125.1.1_CBCONJOINT"
## [25] "X6cfb4a95.fd09.4026.95d7.6dd27d85212a.1.2_CBCONJOINT"
## [26] "f2550540.8b2d.4116.bd4f.9e7d67d39926.1.2_CBCONJOINT"
## [27] "fe83fea4.1983.4e36.8201.40eb6016c125.1.2_CBCONJOINT"
## [28] "X6cfb4a95.fd09.4026.95d7.6dd27d85212a.2.1_CBCONJOINT"
## [29] "f2550540.8b2d.4116.bd4f.9e7d67d39926.2.1_CBCONJOINT"
## [30] "fe83fea4.1983.4e36.8201.40eb6016c125.2.1_CBCONJOINT"
## [31] "X6cfb4a95.fd09.4026.95d7.6dd27d85212a.2.2_CBCONJOINT"
## [32] "f2550540.8b2d.4116.bd4f.9e7d67d39926.2.2_CBCONJOINT"
## [33] "fe83fea4.1983.4e36.8201.40eb6016c125.2.2_CBCONJOINT"
## [34] "X6cfb4a95.fd09.4026.95d7.6dd27d85212a.3.1_CBCONJOINT"
## [35] "f2550540.8b2d.4116.bd4f.9e7d67d39926.3.1_CBCONJOINT"
## [36] "fe83fea4.1983.4e36.8201.40eb6016c125.3.1_CBCONJOINT"
## [37] "X6cfb4a95.fd09.4026.95d7.6dd27d85212a.3.2_CBCONJOINT"
## [38] "f2550540.8b2d.4116.bd4f.9e7d67d39926.3.2_CBCONJOINT"
## [39] "fe83fea4.1983.4e36.8201.40eb6016c125.3.2_CBCONJOINT"
## [40] "revision_CBCONJOINT"
```

```r
# Now, let's just isolate the variables that we will need in our analysis
conjoint2 <- subset(conjoint, select = c("C1",
                                         "C2",
                                         "C3",
                                         "X6cfb4a95.fd09.4026.95d7.6dd27d85212a.1.1_CBCONJOINT",
                                         "f2550540.8b2d.4116.bd4f.9e7d67d39926.1.1_CBCONJOINT",
                                         "fe83fea4.1983.4e36.8201.40eb6016c125.1.1_CBCONJOINT",

                                         "X6cfb4a95.fd09.4026.95d7.6dd27d85212a.1.2_CBCONJOINT",
                                         "f2550540.8b2d.4116.bd4f.9e7d67d39926.1.2_CBCONJOINT",
                                         "fe83fea4.1983.4e36.8201.40eb6016c125.1.2_CBCONJOINT",

                                         "X6cfb4a95.fd09.4026.95d7.6dd27d85212a.2.1_CBCONJOINT",
                                         "f2550540.8b2d.4116.bd4f.9e7d67d39926.2.1_CBCONJOINT",
                                         "fe83fea4.1983.4e36.8201.40eb6016c125.2.1_CBCONJOINT",

                                         "X6cfb4a95.fd09.4026.95d7.6dd27d85212a.2.2_CBCONJOINT",
                                         "f2550540.8b2d.4116.bd4f.9e7d67d39926.2.2_CBCONJOINT",
                                         "fe83fea4.1983.4e36.8201.40eb6016c125.2.2_CBCONJOINT",

                                         "X6cfb4a95.fd09.4026.95d7.6dd27d85212a.3.1_CBCONJOINT",
                                        "f2550540.8b2d.4116.bd4f.9e7d67d39926.3.1_CBCONJOINT",
                                        "fe83fea4.1983.4e36.8201.40eb6016c125.3.1_CBCONJOINT",

                                        "X6cfb4a95.fd09.4026.95d7.6dd27d85212a.3.2_CBCONJOINT",
                                        "f2550540.8b2d.4116.bd4f.9e7d67d39926.3.2_CBCONJOINT",
                                        "fe83fea4.1983.4e36.8201.40eb6016c125.3.2_CBCONJOINT"))

# The variable names are very clunky and convoluted. The code below allows us to change the variable na
names(conjoint2)[names(conjoint2) == "X6cfb4a95.fd09.4026.95d7.6dd27d85212a.1.1_CBCONJOINT"] <-"name_1a
names(conjoint2)[names(conjoint2) == "f2550540.8b2d.4116.bd4f.9e7d67d39926.1.1_CBCONJOINT"] <-"yrs_1a"
names(conjoint2)[names(conjoint2) == "fe83fea4.1983.4e36.8201.40eb6016c125.1.1_CBCONJOINT"] <-"exp_1a"

names(conjoint2)[names(conjoint2) == "X6cfb4a95.fd09.4026.95d7.6dd27d85212a.1.2_CBCONJOINT"] <-"name_1b
names(conjoint2)[names(conjoint2) == "f2550540.8b2d.4116.bd4f.9e7d67d39926.1.2_CBCONJOINT"] <-"yrs_1b"
names(conjoint2)[names(conjoint2) == "fe83fea4.1983.4e36.8201.40eb6016c125.1.2_CBCONJOINT"] <-"exp_1b"

names(conjoint2)[names(conjoint2) == "X6cfb4a95.fd09.4026.95d7.6dd27d85212a.2.1_CBCONJOINT"] <-"name_2a
names(conjoint2)[names(conjoint2) == "f2550540.8b2d.4116.bd4f.9e7d67d39926.2.1_CBCONJOINT"] <-"yrs_2a"
names(conjoint2)[names(conjoint2) == "fe83fea4.1983.4e36.8201.40eb6016c125.2.1_CBCONJOINT"] <-"exp_2a"

names(conjoint2)[names(conjoint2) == "X6cfb4a95.fd09.4026.95d7.6dd27d85212a.2.2_CBCONJOINT"] <-"name_2b
names(conjoint2)[names(conjoint2) == "f2550540.8b2d.4116.bd4f.9e7d67d39926.2.2_CBCONJOINT"] <-"yrs_2b"
names(conjoint2)[names(conjoint2) == "fe83fea4.1983.4e36.8201.40eb6016c125.2.2_CBCONJOINT"] <-"exp_2b"

names(conjoint2)[names(conjoint2) == "X6cfb4a95.fd09.4026.95d7.6dd27d85212a.3.1_CBCONJOINT"] <-"name_3a
names(conjoint2)[names(conjoint2) == "f2550540.8b2d.4116.bd4f.9e7d67d39926.3.1_CBCONJOINT"] <-"yrs_3a"
names(conjoint2)[names(conjoint2) == "fe83fea4.1983.4e36.8201.40eb6016c125.3.1_CBCONJOINT"] <-"exp_3a"

names(conjoint2)[names(conjoint2) == "X6cfb4a95.fd09.4026.95d7.6dd27d85212a.3.2_CBCONJOINT"] <-"name_3b
names(conjoint2)[names(conjoint2) == "f2550540.8b2d.4116.bd4f.9e7d67d39926.3.2_CBCONJOINT"] <-"yrs_3b"
names(conjoint2)[names(conjoint2) == "fe83fea4.1983.4e36.8201.40eb6016c125.3.2_CBCONJOINT"] <-"exp_3b"
```

If we look at the data, we can see that excess information is included in the first two rows that we will not need for the analysis.

```r
# this allows us to drop the first two rows because they are not relevant to our data
conjoint2 <- conjoint2[-c(1, 2), ]
```

Part of the challenge of analyzing conjoint data is that we need it to be in long-form. Recall that we assigned the conjoint task to the respondents three different times in our experiment (each survey respondent was asked three different times to choose between two different peacekeepers to send to the peacekeeping operation). Consequently, we will need each row in the dataset to represent: the respondent-task number-peacekeeper profile.

Each row currently represents the respondent only. Consequently, it would be helpful to create an indicator variable for each respondent.

```r
# this allows us to create an indicator number for each respondent, which we will call "result.id"
conjoint2$result.id <- 1:nrow(conjoint2)
```

Next, we will need to subset the data frame and create a variable that indicates which task (1, 2, or 3) the dataframe represents.

```r
#subsets the data to the first task
a <- subset(conjoint2, select = c("C1",
                                  "name_1a",
                                  "yrs_1a",
                                  "exp_1a",
                                  "name_1b",
                                  "yrs_1b",
                                  "exp_1b",
                                  "result.id"))

#subsets the data to the second task
b <- subset(conjoint2, select = c("C2",
                                  "name_2a",
                                  "yrs_2a",
                                  "exp_2a",
                                  "name_2b",
                                  "yrs_2b",
                                  "exp_2b",
                                  "result.id"))

# subsets the data to the third task
c <- subset(conjoint2, select = c("C3",
                                  "name_3a",
                                  "yrs_3a",
                                  "exp_3a",
                                  "name_3b",
                                  "yrs_3b",
                                  "exp_3b",
                                  "result.id"))

# create a variable indicating which task the dataframe represents
a$task <- 1

b$task <- 2

c$task <- 3
```

Now we will further subset each dataframe by peacekeeper choice. Additionally, we need to create a variable that indicates whether the individual peacekeeper profile (i.e. the combination of traits), was selected as the "winning" profile in the discrete choice set.

```r
# this subsets to: respondent- task 1- peacekeeper1

a_1 <- subset(a, select = c("C1",
                                   "name_1a",
                                   "yrs_1a",
                                   "exp_1a",
                                   "result.id",
                                   "task"))

# create the winner variable
a_1 <- a_1 %>%
  mutate(winner = case_when(
  C1 == "1" ~ 1,
  C1 == "2" ~ 0
)) # the code creates a new variable called "winner" and states that if C1 (task one variable) indicate
```

We will repeat this for task 1, peacekeeper 2 profile and repeat for each of the other 2 tasks and 2 peacekeepers.

```
# this subsets to: respondent- task 1- peacekeeper2
a_2 <- subset(a, select = c("C1",
                                  "name_1b",
                                  "yrs_1b",
                                  "exp_1b",
                                  "result.id",
                                  "task"))

a_2 <- a_2 %>%
  mutate(winner = case_when(
  C1 == "2" ~ 1,
  C1 == "1" ~ 0
))

# we will change the names of the name, years and experience variables to match between a_1 and a_2 so
names(a_2)[names(a_2) == "name_1b"] <-"name_1a"
names(a_2)[names(a_2) == "yrs_1b"] <-"yrs_1a"
names(a_2)[names(a_2) == "exp_1b"] <-"exp_1a"

# this merges the two a dataframes (the one that represents task 1, peacekeeper 1 and task 1, peacekeep
a <- full_join(a_1, a_2)
```

```
## Joining, by = c("C1", "name_1a", "yrs_1a", "exp_1a", "result.id", "task", "winner")
```

```
# we then change the variable names for dataframe a to ones that will match the consequent variables in
names(a)[names(a) == "name_1a"] <-"name"
names(a)[names(a) == "yrs_1a"] <-"yrs"
names(a)[names(a) == "exp_1a"] <-"exp"
```

```
b_1 <- subset(b, select = c("C2",
                                  "name_2a",
                                  "yrs_2a",
                                  "exp_2a",
                                  "result.id",
                                  "task"))

b_1 <- b_1 %>%
  mutate(winner = case_when(
  C2 == "1" ~ 1,
  C2 == "2" ~ 0
))

b_2 <- subset(b, select = c("C2",
                                  "name_2b",
                                  "yrs_2b",
                                  "exp_2b",
                                  "result.id",
                                  "task"))

b_2 <- b_2 %>%
  mutate(winner = case_when(
  C2 == "2" ~ 1,
  C2 == "1" ~ 0
))
```

```r
names(b_2)[names(b_2) == "name_2b"] <-"name_2a"
names(b_2)[names(b_2) == "yrs_2b"] <-"yrs_2a"
names(b_2)[names(b_2) == "exp_2b"] <-"exp_2a"

b <- full_join(b_1, b_2)
```

```
## Joining, by = c("C2", "name_2a", "yrs_2a", "exp_2a", "result.id", "task", "winner")
```

```r
names(b)[names(b) == "name_2a"] <-"name"
names(b)[names(b) == "yrs_2a"] <-"yrs"
names(b)[names(b) == "exp_2a"] <-"exp"
```

```r
c_1 <- subset(c, select = c("C3",
                                  "name_3a",
                                  "yrs_3a",
                                  "exp_3a",
                                  "result.id",
                                  "task"))

c_1 <- c_1 %>%
  mutate(winner = case_when(
  C3 == "1" ~ 1,
  C3 == "2" ~ 0
))

c_2 <- subset(c, select = c("C3",
                                  "name_3b",
                                  "yrs_3b",
                                  "exp_3b",
                                  "result.id",
                                  "task"))

c_2 <- c_2 %>%
  mutate(winner = case_when(
  C3 == "2" ~ 1,
  C3 == "1" ~ 0
))

names(c_2)[names(c_2) == "name_3b"] <-"name_3a"
names(c_2)[names(c_2) == "yrs_3b"] <-"yrs_3a"
names(c_2)[names(c_2) == "exp_3b"] <-"exp_3a"

c <- full_join(c_1, c_2)
```

```
## Joining, by = c("C3", "name_3a", "yrs_3a", "exp_3a", "result.id", "task", "winner")
```

```r
names(c)[names(c) == "name_3a"] <-"name"
names(c)[names(c) == "yrs_3a"] <-"yrs"
names(c)[names(c) == "exp_3a"] <-"exp"
```

Now that we have dataframes a, b and c (representing tasks 1, 2 and 3), we need to merge the three dataframes.

To do this, we will use a full join to first merge a and b. Then, we will merge the new data frame with c.

Our new dataframe, called "conjoint3" is now in long form. For our purposes, this means that each row represents the peacekeeper number (1 or 2) by task for each respondent (i.e. peacekeeper 1, task 1, for respondent 3).

```
merge1 <- full_join(a, b)
```

```
## Joining, by = c("name", "yrs", "exp", "result.id", "task", "winner")
```

```
conjoint3 <- full_join(merge1, c)
```

```
## Joining, by = c("name", "yrs", "exp", "result.id", "task", "winner")
```

```
conjoint3 <- subset(conjoint3, select = c("name",
                                  "yrs",
                                  "exp",
                                  "result.id",
                                  "task",
                                  "winner"))
```

Now that we have our data in the correct format, we can analyze it!

There are several different ways to analyze conjoint data. In political science, the Average Marginal Components Effect (AMCE) and the Marginal Means (MM) are the two most common ways to analyze conjoint data. For a more complete discussion of the two different methods, see Hainmueller, Hopkins and Yamamoto (2014) and Leeper, Hobolt and Tilley (2020).

To analyze the AMCE, see the code below:

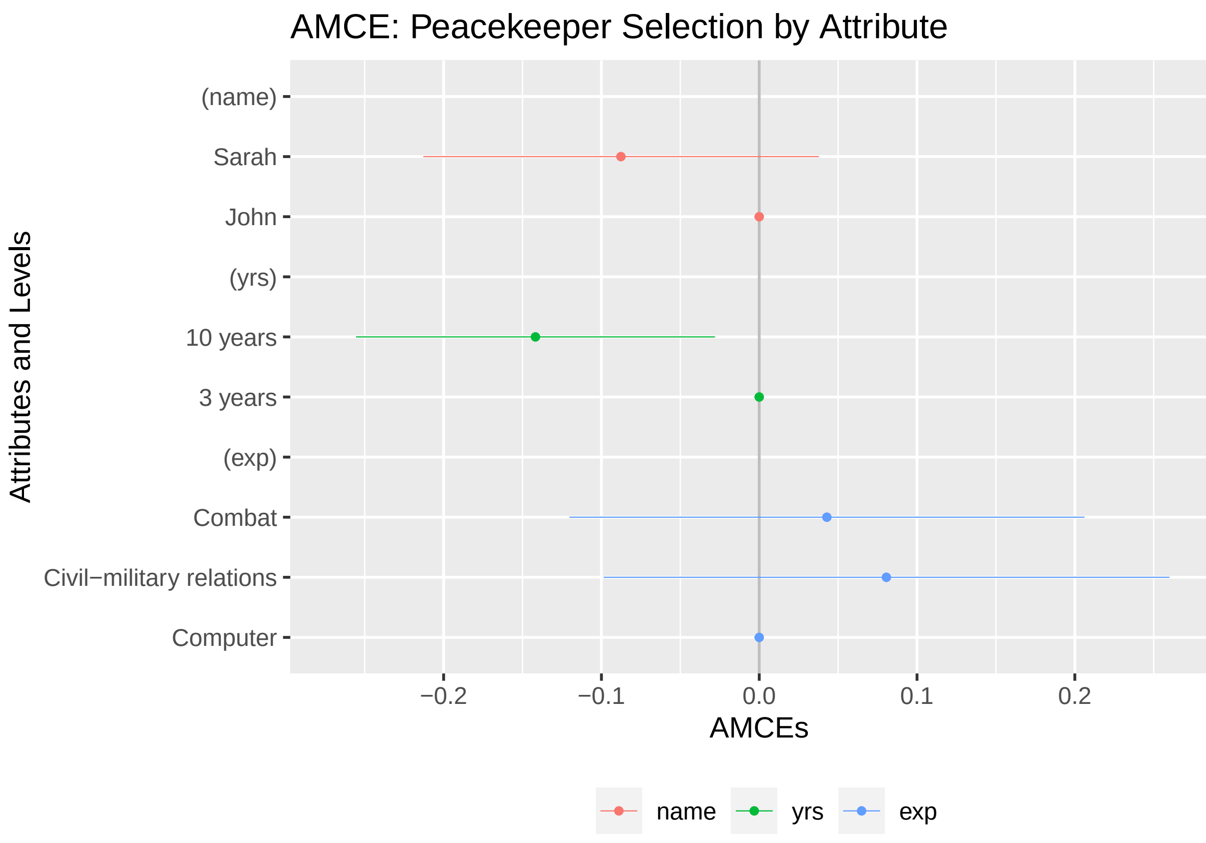
Note that AMCE requires the establishment of a baseline (a point of comparison against the other elements of the same attribute). This is what is referred to as "baselines" down below.

```
library(cregg) # package that we will need for the analysis

#baselines for AMCE
baselines <- list()
baselines$name <- "John"
baselines$yrs <- "3 years"
baselines$exp <- "Computer"

#establish factor order so conjoint levels in correct order: NOTE that the conjoint attributes MUST be
conjoint3$name <- factor(conjoint3$name, levels=c("John", "Sarah"))
conjoint3$yrs <- factor(conjoint3$yrs, levels= c("3 years", "10 years"))
conjoint3$exp <- factor(conjoint3$exp, levels= c("Computer", "Civil-military relations", "Combat"))

f1 <- winner ~ name + yrs + exp # base model

# amce model with respondent fixed effects
amces <- cj(conjoint3, f1, id = ~result.id, estimate="amce")
```

```
## Warning in logLik.svyglm(x): svyglm not fitted by maximum likelihood.
```

Now we can plot the results:

```
plot(amces, vline = 0) + ggtitle("AMCE: Peacekeeper Selection by Attribute") + labs(color = "Attributes
```



AMCE results-1.pdf

```
head(amces[c("feature", "level", "estimate", "std.error")], 8L)
```

```
##   feature                    level    estimate  std.error
## 1    name                     John  0.00000000         NA
## 2    name                    Sarah -0.08764306 0.06385339
## 3     yrs                  3 years  0.00000000         NA
## 4     yrs                 10 years -0.14181897 0.05793138
## 5     exp                 Computer  0.00000000         NA
## 6     exp Civil-military relations  0.08059066 0.09140696
## 7     exp                   Combat  0.04286869 0.08319160
```

To analyze the MM, use the code below:

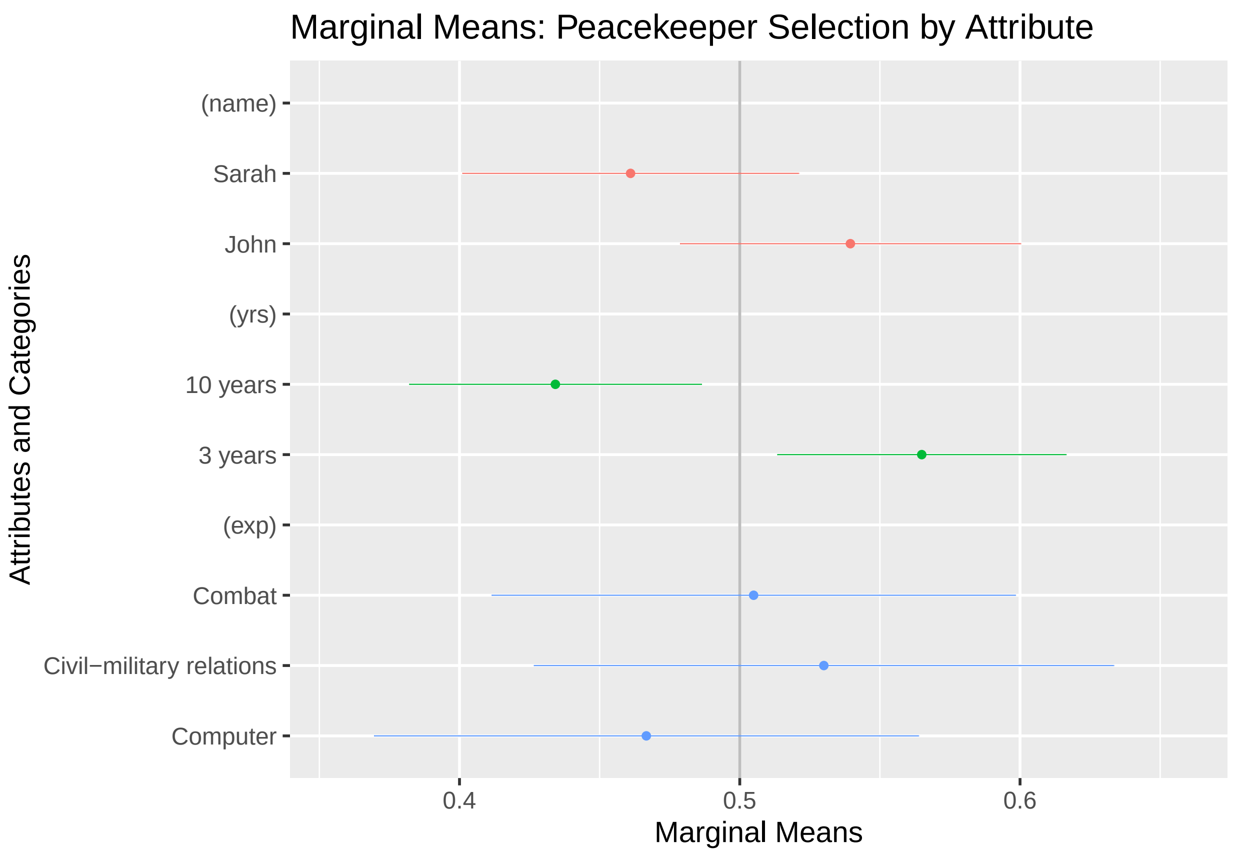
```
library(cregg)

#### Marginal Means: General ####
f1 <- winner ~ name + yrs + exp # base model

#### Marginal means estimates for each conjoint attribute ####
margmeans <- cj(conjoint3, f1, id = ~result.id, estimate = "mm")
```

```
## Warning in logLik.svyglm(x): svyglm not fitted by maximum likelihood.

## Warning in logLik.svyglm(x): svyglm not fitted by maximum likelihood.

## Warning in logLik.svyglm(x): svyglm not fitted by maximum likelihood.
```

Now we can plot the results:

Note that MM does not require the establishment of a baseline, but rather allows you to compare the effect within attributes (i.e. we can see which attribute drives the effect "more", i.e. 3 years versus 10 years of experience).

```
plot(margmeans, vline = 0.5) + ggtitle("Marginal Means: Peacekeeper Selection by Attribute") + labs(colo
```



MM-1.pdf

```
head(margmeans[c("feature", "level", "estimate", "std.error")], 8L) # plotting marginal means
```

```
##   feature                    level  estimate  std.error
## 1    name                     John 0.5394737 0.03101336
## 2    name                    Sarah 0.4610390 0.03062352
## 3     yrs                  3 years 0.5649351 0.02629573
## 4     yrs                 10 years 0.4342105 0.02659949
## 5     exp                 Computer 0.4666667 0.04957067
## 6     exp Civil-military relations 0.5300000 0.05278690
## 7     exp                   Combat 0.5049505 0.04768427
```