Weathering the Storm Conjoint Log

Bailee Donahue and Mark J. C. Crescenzi

8/22/2022

Conjoint Analysis

This document provides a log of the conjoint analysis in the main text of Weathering the Storm: Discordant Learning About Reputations for Reliability.

```
setwd("~/Downloads/dataverse_files")


library(MASS)
library(reshape2)
library(reshape)
```

```
##
## Attaching package: 'reshape'
```

```
## The following objects are masked from 'package:reshape2':
##
##     colsplit, melt, recast
```

```
library(countrycode)
library(states)
```

```
## Warning: replacing previous import 'vctrs::data_frame' by 'tibble::data_frame'
## when loading 'dplyr'
```

```
library(pltesim)
library(readstata13)
library(plyr)
```

```
##
## Attaching package: 'plyr'
```

```
## The following objects are masked from 'package:reshape':
##
##     rename, round_any
```

```
library(foreign)
##install.packages("cjoint")
library(cjoint)
```

```
## Loading required package: sandwich
```

```
## Loading required package: lmtest
```

```
## Loading required package: zoo
```

```
##
## Attaching package: 'zoo'
```

```
## The following objects are masked from 'package:base':
##
##     as.Date, as.Date.numeric

## Loading required package: ggplot2

## Loading required package: survey

## Loading required package: grid

## Loading required package: Matrix

##
## Attaching package: 'Matrix'

## The following object is masked from 'package:reshape':
##
##     expand

## Loading required package: survival

##
## Attaching package: 'survey'

## The following object is masked from 'package:graphics':
##
##     dotchart

## cjoint: AMCE Estimator for Conjoint Experiments
## Version: 2.1.0
## Authors: Soubhik Barari, Elissa Berwick, Jens Hainmueller, Daniel Hopkins, Sean Liu, Anton Strezhnev
```

```
##install.packages("FindIt")
library(FindIt)
```

```
## Loading required package: arm

## Loading required package: lme4

##
## arm (Version 1.11-2, built: 2020-7-27)

## Working directory is /Users/baileedonahue/Downloads/dataverse_files
```

```
#install.packages("snow")
library(snow)
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.0 --

## v tibble  3.0.2     v dplyr   1.0.0
## v tidyr   1.1.0     v stringr 1.4.0
## v readr   1.3.1     v forcats 0.5.0
## v purrr   0.3.4

## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::arrange()   masks plyr::arrange()
## x purrr::compact()   masks plyr::compact()
## x dplyr::count()     masks plyr::count()
## x tidyr::expand()    masks Matrix::expand(), reshape::expand()
## x dplyr::failwith()  masks plyr::failwith()
## x dplyr::filter()    masks stats::filter()
## x dplyr::id()        masks plyr::id()
```

```
## x dplyr::lag()       masks stats::lag()
## x dplyr::mutate()    masks plyr::mutate()
## x tidyr::pack()      masks Matrix::pack()
## x dplyr::rename()    masks plyr::rename(), reshape::rename()
## x dplyr::select()    masks MASS::select()
## x dplyr::summarise() masks plyr::summarise()
## x dplyr::summarize() masks plyr::summarize()
## x tidyr::unpack()    masks Matrix::unpack()
## x tibble::view()     masks cjoint::view()
```

```
cl <- makeCluster(8,"SOCK")



set.seed(384297)

conjoint<- read.csv("conjoint_final.csv")
```

Conjoint Analysis Figure 2

```
## Government Type - Democracy as the basline category
conjoint$Gov <- factor(conjoint$gov,
                       levels=c("Democracy", "Anocracy", "Autocracy"))
## Power - Top 15 Most Powerful as baseline category
conjoint$Pow <- factor(conjoint$pow,
                       levels =c("top 15 most powerful", "top 25 most powerful", "top 50 most powerful"
## Region - Middle East North Africa
conjoint$Reg<- factor(conjoint$reg,
                      levels = c("Middle East North Africa", "subSaharan Africa",
                                 "Asia", "South America", "Europe"))
## Alliance Type - Defensive
conjoint$All <- factor(conjoint$all,
                       levels = c("Defensive Alliance", "Offensive Alliance", "Neutrality Pact"))
## Reputation for Reliability - Good Reputation as the baseline category
conjoint$Rep <- factor(conjoint$rep,
                       levels = c("Good Reputation", "Middling Reputation","Bad Reputation"))

## Human Rights record - Good HR record as the baseline

conjoint$HR <- factor(conjoint$hr,
                      levels = c("Good HR",
                                 "Mediocre HR",
                                 "Bad HR"))




bootstrap_cluster<- function(conjoint){
  i <- sample(unique(conjoint$random_id),length(unique(conjoint$random_id)),replace=TRUE)
  row.nums <- NULL
  for (j in 1:length(i)){
    row.nums <- c(row.nums, which(conjoint$random_id==i[j]))
  }
```

```
  return(conjoint[row.nums,])
}

######## Main model ########

#Main analysis
conjoint_boot2 <- function(...){
  temp <- bootstrap_cluster(conjoint)
  mod.temp <- lm(selected ~  Rep + Gov + Pow + Reg + All + HR , data=temp)
  return(coef(mod.temp))
}

clusterExport(cl, list("conjoint", "conjoint_boot2", "bootstrap_cluster"))

main.boot <- parSapply(cl, rep(1,1500), conjoint_boot2)

plot.main.boot <- cbind(apply(main.boot, 1, mean), apply(main.boot, 1, quantile, c(0.025)), apply(main.

## This generates the baseline categories
plot_main_2<- rbind(rep(0,5), plot.main.boot[1:2,], rep(0,5), plot.main.boot[3:4,], rep(0,5), plot.main

## This generates Labels for the Graph
Labels_Conjoint <- c("Good Reputation", "Middling Reputation", "Bad Reputation",
                     "Democracy", "Anocracy", "Autocracy", "Top 15 most powerful",
                     "Top 25 most powerful", "Top 50 most powerful", "Middle East North Africa",
                     "SubSaharan Africa", "Asia", "South America", "Europe",
                     "Defensive Alliance", "Offensive Alliance",  "Neutrality Pact",
                     "Good Human Rights Record", "Mediocre Human Rights Record", "Bad Human Rights Reco
```

Code to Produce Figure 2

```
pdf(file = "WTS_Conjoint_Fig2", height=5, width=6.5)
#dev.new(height=9, width=6.5)
par(mar=c(6.1,1.1,4.1,0), oma=c(0,4,0,0))
plot(plot_main_2[,1], nrow(plot_main_2):1, type="n", axes=FALSE, xlab="Average marginal component effec
for (i in 1:nrow(plot_main_2)){
  points(plot_main_2[i,1], nrow(plot_main_2)-(i-1), pch=16)
  lines(plot_main_2[i,c(2,5)], rep(nrow(plot_main_2)-(i-1),2))
  lines(plot_main_2[i,c(3,4)], rep(nrow(plot_main_2)-(i-1),2), lwd=2)
  text(-0.5, nrow(plot_main_2)-(i-1), Labels_Conjoint[i], cex=0.7)
}
abline(v=0)
axis(1)
dev.off()
```

```
## pdf
##   2
```

AMIE Analysis Main Text

```
## For Causal Anova

## Set Levels as explicitly ordered factors so that Find It Causal Anova will run
```

```r
conjoint$Gov <- factor(conjoint$Gov, ordered = TRUE,
                       levels=c("Democracy", "Anocracy", "Autocracy"))
conjoint$Pow <- factor(conjoint$Pow, ordered = TRUE,
                       levels =c("top 15 most powerful", "top 25 most powerful", "top 50 most powerful
conjoint$Reg<- factor(conjoint$Reg, ordered = TRUE,
                      levels = c("Middle East North Africa", "subSaharan Africa",
                                 "Asia", "South America", "Europe"))
conjoint$All <- factor(conjoint$All, ordered = TRUE,
                       levels = c("Defensive Alliance", "Offensive Alliance", "Neutrality Pact"))
conjoint$Rep <- factor(conjoint$Rep, ordered = TRUE,
                       levels = c("Good Reputation", "Middling Reputation","Bad Reputation"))
conjoint$HR <- factor(conjoint$HR, ordered = TRUE,
                      levels = c("Good HR",
                                 "Mediocre HR",
                                 "Bad HR"))

conjoint$select <- factor(conjoint$selected, ordered = FALSE,
                          levels = c("0", "1"))

conjoint <- as.data.frame(conjoint)

set.seed(890909)

## Run Causal ANOVA


fit4 <- FindIt(model.treat= selected ~ Gov + Pow +
                 Reg + All + HR + Rep,
               nway=3, treat.type="multiple", data = conjoint)
```

```
## [1] -15  10
## [1] -10   0
## [1] -7.5 -2.5
## [1] -8 -3
## [1] -7 -4
## [1] -7.25 -4.25
## [1] -6.25 -5.25
## [1] -6.0 -5.5
## [1] -6.05 -5.55
## [1] -5.95 -5.65
## [1] -5.875 -5.575
## [1] -5.825 -5.625
## [1] -5.775 -5.675
## [1] -5.75 -5.70
## [1] -5.735 -5.715
## [1] -5.737 -5.717
## [1] -5.7295 -5.7245
```

```r
print("This model has finished running!")
```

```
## [1] "This model has finished running!"
```

```r
## Save this model because it takes a really long time to run
save(fit4, file="Three_Way_Interaction.RData")
```

```
full.FindIt <- function(object){
  ## This generates the full factoral di.
  comb.list <- apply(object$treat.orig,2,unique)
  full <- as.data.frame(expand.grid(comb.list))
  invisible(full)
}

full <- full.FindIt(fit4) # This generates a full factoral design matrix
pred.dat <- predict(fit4, newdata=full, sort=FALSE)$data #Predicted potential outcomes
#Save model
save(pred.dat, file="Predicted_Alliance_Conjoint.RData")
```

##Below generates an INT Function. ## This version is used to be consistent with the way AMTIE's have previously been calculated in the experimental reputations literature see Kertzer, Renshon, Yarhi-Milo (2019). ## This version of the INT function is taken from the FindIt package version 0.6, by Naoki Egami, Marc Ratkovic and Kosuke Imai. Newer versions of the package use a slightly different estimation methods.

```
INT <- function (object, target.data, column, dist = "target", base,
                 sort = TRUE, compare = FALSE, order = 2)
{
  if (all(class(object) != "FindIt")) {
    warning("the class of object needs to be FindIt.")
  }
  Restriction <- NULL
  measure <- "TIE"
  if (compare == TRUE) {
    if (missing(order)) {
      warning("Need to specify the order")
    }
  }
  coefs <- object$coefs.orig
  names(coefs) <- c("Intercept", object$name.t)
  outcome.type <- object$type
  if (dist == "unique") {
    data <- cbind(object$y.orig, object$treat.orig)
    colnames(data)[-1] <- colnames(object$treat.orig)
    data <- unique(data, MARGIN = 1)
  }
  if (dist == "sample") {
    data <- cbind(object$y.orig, object$treat.orig)
    colnames(data)[-1] <- colnames(object$treat.orig)
  }
  if (dist == "target") {
    data <- target.data
    print("Using Data representing the target population.")
  }
  else {
    print("Using the Data used to fit the model.")
  }
  if (!missing(column) & !missing(base)) {
    for (i in 1:length(column)) {
      ind.column <- which(colnames(data) == column[i])
      data[, ind.column] <- relevel(data[, ind.column],
```

```
                                    base[i])
  }
}
if (!missing(column)) {
  ind <- c()
  for (i in 1:length(column)) {
    ind[i] <- which(colnames(data) == column[i])
  }
  column <- column[order(ind, decreasing = FALSE)]
}
base <- 1
Marginal <- function(data, base, Restriction) {
  Effect.list <- list()
  Name.list <- list()
  range.mar <- c()
  for (i in 1:(ncol(data) - 1)) {
    columnM <- colnames(data)[i + 1]
    A <- tapply(data[, 1], data[, columnM], mean, simplify = FALSE)
    Effect1 <- unlist(A)
    if (base == "min") {
      base <- which(Effect1 == min(Effect1))
    }
    TEffect <- Effect1 - Effect1[base]
    range.mar[i] <- max(TEffect) - min(TEffect)
    Effect.list[[i]] <- TEffect
    Name.list[[i]] <- paste(colnames(data)[(i + 1)],
                            names(TEffect), sep = "_")
  }
  names(range.mar) <- colnames(data)[2:ncol(data)]
  name <- unlist(Name.list)
  Treatment.Effect1 <- unlist(Effect.list)
  Treatment.Effect1 <- as.data.frame(Treatment.Effect1)
  colnames(Treatment.Effect1) <- "AMTE"
  rownames(Treatment.Effect1) <- name
  output <- list(Treatment.Effect1 = Treatment.Effect1,
                 range = range.mar)
  return(output)
}
if (compare == FALSE & missing(column)) {
  Treatment.Effect1 <- Marginal(data = data, base = base,
                                Restriction = Restriction)$Treatment.Effect1
  Treatment.Effect <- Treatment.Effect1
}
if (compare == TRUE & order == 1) {
  Marginal.range <- Marginal(data = data, base = base,
                             Restriction = Restriction)$range
  print("Range of Marginal Effects")
  return(Marginal.range)
}
comb.prem <- function(data, column, Restriction, order) {
  A <- tapply(data[, 1], data[, column], mean, simplify = FALSE)
  A2 <- tapply(data[, 1], data[, column], mean, simplify = TRUE)
  A[is.na(A2)] <- NA
```

```
  Effect1 <- unlist(A)
  if (base == "min") {
    base <- which(Effect1 == min(Effect1))
  }
  Comb.Effect <- Effect1 - Effect1[base]
  Treatment.Effect <- cbind(Comb.Effect, expand.grid(dimnames(A)))
  Treatment.Effect <- na.omit(Treatment.Effect)
  Comb.Effect <- Treatment.Effect[, 1]
  Combination.name <- Treatment.Effect[, -1]
  Treatment.Effect.print <- Treatment.Effect
  for (j in 2:ncol(Treatment.Effect)) {
    Treatment.Effect[, j] <- paste(colnames(Treatment.Effect)[j],
                                   Treatment.Effect[, j], sep = "_")
  }
  Treatment.Effect1 <- Marginal(data = data, base = base,
                                Restriction = Restriction)$Treatment.Effect1
  Main.comb <- c()
  for (i in 1:nrow(Treatment.Effect)) {
    main <- c()
    for (j in 1:(ncol(Treatment.Effect) - 1)) {
      main[j] <- Treatment.Effect1[rownames(Treatment.Effect1) ==
                                     Treatment.Effect[i, (j + 1)], 1]
    }
    Main.comb[i] <- sum(main)
  }
  Sum.Mar.Effect <- Main.comb - Main.comb[base]
  if (order == 2) {
    TIE <- round(Comb.Effect - Sum.Mar.Effect, digits = 8)
  }
  if (order == 3) {
    data.column.three <- data[, column]
    Two.way.comb.list <- list()
    for (j in 1:3) {
      column.com.three <- c()
      Two.way.comb <- list()
      column.com.three <- colnames(data.column.three)[c(combn(length(data.column.three),
                                                              2)[, j])]
      A.Three <- tapply(data[, 1], data[, column.com.three],
                        mean, simplify = FALSE)
      A2.Three <- tapply(data[, 1], data[, column.com.three],
                         mean, simplify = TRUE)
      A.Three[is.na(A2.Three)] <- NA
      EffectTwo <- unlist(A.Three)
      Two.way.comb <- EffectTwo - EffectTwo[base]
      Two.way.comb2 <- cbind(Two.way.comb, expand.grid(dimnames(A.Three)))
      for (k in 2:3) {
        Two.way.comb2[, k] <- paste(colnames(Two.way.comb2)[k],
                                    Two.way.comb2[, k], sep = "_")
      }
      colnames(Two.way.comb2) <- c("Two.way.comb",
                                   "first", "second")
      Two.way.comb.list[[j]] <- na.omit(Two.way.comb2)
    }
```

```
    Two.way.comb.Final <- do.call(rbind, Two.way.comb.list)
    Two.Way.sum <- c()
    for (i in 1:nrow(Treatment.Effect)) {
      two.sum <- c()
      for (j in 1:nrow(Two.way.comb.Final)) {
        ind.two <- sum(is.element(Two.way.comb.Final[j,
                                                     2:3], Treatment.Effect[i, 2:4]))
        if (ind.two == 2) {
          two.sum[j] <- Two.way.comb.Final[j, 1]
        }
        else {
          two.sum[j] <- 0
        }
      }
      Two.Way.sum[i] <- sum(two.sum)
    }
    Sum.Two.Effect <- Two.Way.sum - Two.Way.sum[base]
    TIE <- round(Comb.Effect - Sum.Two.Effect + Sum.Mar.Effect,
                 digits = 8)
  }
  range.change <- max(TIE) - min(TIE)
  Order.data <- cbind(Sum.Mar.Effect, Comb.Effect)
  Order.data <- Order.data[order(Order.data[, 1], decreasing = TRUE),
                           ]
  Order.data <- as.data.frame(Order.data)
  Order.data$order.main <- seq(1:nrow(Order.data))
  Order.comb <- Order.data[order(Order.data[, 2], decreasing = TRUE),
                           "order.main"]
  Order.diff <- Order.comb - seq(1:nrow(Order.data))
  max.order.diff <- max(abs(Order.diff))
  return(list(A = A, Treatment.Effect = Treatment.Effect,
              Comb.Effect = Comb.Effect, Combination.name = Combination.name,
              Sum.Mar.Effect = Sum.Mar.Effect, TIE = TIE, range.change = range.change,
              max.order.diff = max.order.diff))
}
if (compare == FALSE) {
  if (!missing(column)) {
    X <- comb.prem(data = data, column = column, Restriction = Restriction,
                   order = order)
    A <- X$A
    Treatment.Effect <- X$Treatment.Effect
    Comb.Effect <- X$Comb.Effect
    Combination.name <- X$Combination.name
    Sum.Mar.Effect <- X$Sum.Mar.Effect
    TIE <- X$TIE
    range.change <- X$range.change
  }
}
else {
  data.column <- data[, -1]
  column.com <- list()
  Range.com <- list()
  for (j in 1:choose(length(data.column), order)) {
```

```
    column.com[[j]] <- colnames(data.column)[c(combn(length(data.column),
                                                    order)[, j])]
    if (measure == "TIE") {
      Range.com[[j]] <- comb.prem(data = data, column = column.com[[j]],
                                  Restriction = Restriction, order = order)$range.change
    }
    if (measure == "Order.Diff") {
      Range.com[[j]] <- comb.prem(data = data, column = column.com[[j]],
                                  Restriction = Restriction, order = order)$max.order.diff
    }
  }
  column.com <- as.data.frame(matrix(unlist(column.com),
                                     ncol = order, byrow = TRUE))
  Compare <- as.data.frame(cbind(column.com, unlist(Range.com)))
  Compare <- Compare[order(Compare[, (order + 1)], decreasing = TRUE),
                     ]
  if (measure == "TIE") {
    colnames(Compare) <- c(colnames(Compare)[1:order],
                           "range.TIE")
  }
  if (measure == "Order.Diff") {
    colnames(Compare) <- c(colnames(Compare)[1:order],
                           "max.Order.diff")
  }
  return(Compare)
}
if (!missing(column)) {
  if (length(column) == 2) {
    Treatment.coefs.name <- Treatment.Effect[, 2]
    if (ncol(Treatment.Effect) >= 3) {
      for (i in 1:nrow(Treatment.Effect)) {
        for (j in 3:ncol(Treatment.Effect)) {
          Treatment.coefs.name[i] <- paste(Treatment.coefs.name[i],
                                           Treatment.Effect[i, j], sep = ".")
        }
      }
    }
    Treatment.coefs.name <- gsub(" ", ".", Treatment.coefs.name)
  }
}
if (missing(column)) {
  Treatment.coefs.name <- rownames(Treatment.Effect1)
  Treatment.coefs.name <- gsub(" ", ".", Treatment.coefs.name)
}
if (missing(column)) {
  Coefficients <- c()
  for (i in 1:length(Treatment.coefs.name)) {
    if (Treatment.coefs.name[i] %in% names(coefs)) {
      Coefficients[i] <- coefs[names(coefs) == Treatment.coefs.name[i]]
    }
    else {
      Coefficients[i] <- NA
    }
```

```
  }
  if (outcome.type == "binary") {
    Coefficients <- Coefficients/2
  }
}
if (!missing(column)) {
  if (length(column) == 2) {
    Coefficients <- c()
    for (i in 1:length(Treatment.coefs.name)) {
      if (Treatment.coefs.name[i] %in% names(coefs)) {
        Coefficients[i] <- coefs[names(coefs) == Treatment.coefs.name[i]]
      }
      else {
        Coefficients[i] <- NA
      }
    }
    if (outcome.type == "binary") {
      Coefficients <- Coefficients/2
    }
  }
}
if (!missing(column)) {
  if (length(column) == 2) {
    Treatment.Effect.print <- cbind(Comb.Effect, TIE,
                                    Combination.name)
    colnames(Treatment.Effect.print) <- c("ATCE", "AMTIE",
                                          colnames(Combination.name))
    Main.matrix <- cbind(Sum.Mar.Effect, TIE, Combination.name)
    colnames(Main.matrix) <- c("Sum of AMTEs", "AMTIE",
                               colnames(Combination.name))
    Inequality.Cor <- cor(Sum.Mar.Effect, TIE)
    Relative.change <- cbind(TIE, Combination.name)
    colnames(Relative.change) <- c("AMTIE", colnames(Combination.name))
    Relative.change <- Relative.change[order(Relative.change[,
                                                             1], decreasing = TRUE), ]
  }
  if (length(column) >= 3) {
    Treatment.Effect.print <- cbind(Comb.Effect, TIE,
                                    Combination.name)
    colnames(Treatment.Effect.print) <- c("ATCE", "AMTIE",
                                          colnames(Combination.name))
    Main.matrix <- cbind(Sum.Mar.Effect, TIE, Combination.name)
    colnames(Main.matrix) <- c("Sum of AMTEs", "AMTIE",
                               colnames(Combination.name))
    Relative.change <- cbind(TIE, Combination.name)
    colnames(Relative.change) <- c("AMTIE", colnames(Combination.name))
    Relative.change <- Relative.change[order(Relative.change[,
                                                             1], decreasing = TRUE), ]
  }
}
else {
  Treatment.Effect.print <- Treatment.Effect1
  Man.matrix <- NULL
```

```
  }
  Treatment.Effect.print <- as.data.frame(Treatment.Effect.print)
  if (!missing(column)) {
    Main.matrix <- as.data.frame(Main.matrix)
  }
  if (sort == TRUE) {
    Treatment.Effect.print1 <- Treatment.Effect.print
    Treatment.Effect.print <- Treatment.Effect.print[order(Treatment.Effect.print[,
                                                                                  1], decreasing = TRUE
    if (!missing(column)) {
      Main.matrix <- Main.matrix[order(Main.matrix$Sum,
                                       decreasing = TRUE), ]
      Treatment.Effect.print <- cbind(Treatment.Effect.print[,
                                                             1:2], Treatment.Effect.print[, 3:ncol(Trea
    }
  }
  if (!missing(column)) {
    if (sort) {
      return(list(`Range of AMTIE` = range.change, AMTIE = Relative.change,
                  ATCE = Treatment.Effect.print, `Sum of AMTEs` = Main.matrix))
    }
    else {
      return(list(`Range of AMTIE` = range.change, AMTIE = Relative.change,
                  ATCE = Treatment.Effect.print, `Sum of AMTEs` = Main.matrix))
    }
  }
  else {
    return(list(Treatment.Effect = Treatment.Effect.print))
  }
}


one_way <- INT(fit4, target.data=pred.dat, compare=TRUE, order=1) #This generates the main effects pres
```

```
## [1] "Using Data representing the target population."
## [1] "Range of Marginal Effects"
```

```
##Calculate barplot data
sorted.one_way <- sort(one_way, decreasing=TRUE)
one_way.names <- names(sorted.one_way)


one_way_names_new <- one_way.names
one_way_names_new <- sub("HR", "Human Rights",one_way_names_new)
one_way_names_new <- sub("Rep", "Reputation",one_way_names_new)
one_way_names_new <- sub("Pow", "Power",one_way_names_new)
one_way_names_new <- sub("Gov", "Government",one_way_names_new)
one_way_names_new <- sub("Reg", "Region",one_way_names_new)
one_way_names_new <- sub("All", "Alliance", one_way_names_new)
```

Code to Produce Figure 3

```
## This code produces figure 3 in the main text
oneway_barplot <- c(as.data.frame(sorted.one_way)$sorted.one_way)
```

```r
oneway_barplot_rev <- rev(oneway_barplot)

barplot.names.oneway <- one_way_names_new
barplot.names.reversed.oneway <- rev(barplot.names.oneway)


pdf("Conjoint_AMIE_Fig3", height=5, width=7.5)

par(mar = c(4,8,1,1))
pos <- barplot(height=oneway_barplot_rev, horiz=TRUE, xlim=c(0,.4), names.arg = barplot.names.reversed.
axis(side=2, at=pos,labels=FALSE, tck=-0.02)
axis(side=1)
box()
abline(v=seq(0,0.4,by=0.05), lty=3, col=rgb(0,0,0,0.5))

dev.off()
```

```
## pdf
##   2
```