An R Markdown program to create the experimental design for a Discrete Choice Experiment (DCE) exploring online help seeking in socially anxious young people

Pilot survey design sub-routine (reproduction)

Matthew P Hamilton[1,*]

09 June 2022

[1] Orygen, Parkville, Australia

[*] Correspondence: Matthew P Hamilton <matthew.hamilton@orygen.org.au>

Suggested citation: "Matthew Hamilton (2022). dce_sa_design: An R Markdown program to create the experimental design for a Discrete Choice Experiment (DCE) exploring online help seeking in socially anxious young people. Zenodo. https://doi.org/10.5281/zenodo.6626256"

1 About this code

1.1 Motivation

This program was used to generate the pilot survey choice efficient design and choice cards for a Discrete Choice Experiment study that is currently being written up. Future versions of this program will include details of the parent study.

1.2 Status

This code has been minimally adapted from when it was first applied. It has not been formatted for consistency or ease of use and a number of sections have copied almost unchanged from online examples provided by other authors to demonstrate the third party functions (in particular from the idefix package) used in this program. Future releases of this program aim to adopt a more consistent and integrated approach that should make the program easier to follow and adapt.

1.3 Use

When using this code it is important to note that some of the steps in this program involve interactivity - they generate a prompt that a user must respond to before proceeding. Therefore, **this code should be run step by step** (i.e run one chunk at a time and do not try to run the program by knitting the R Markdown version of this code). Although it would be possible to add work-arounds to the interactivity issue, running the program by knitting the RMD version is still not recommended as it will prevent the documents generated by this program from rendering properly.

2 Install and load required libraries

If you do not already have the required libraries to run this program installed, you can do so by un-commenting and running the following lines.

```
# utils::install.packages("idefix")
# utils::install.packages("kableExtra")
# utils::install.packages("knitr")
# utils::install.packages("magick")
# utils::install.packages("magrittr")
# utils::install.packages("shiny")
# utils::install.packages("stringr")
# utils::install.packages("webshot")
# utils::install.packages("xfun")
```

Next load the libraries required to run this program.

```
library(magrittr)
```

3 Create custom functions

Next we create a number of functions that we will use in subsequent parts of this program.

```
make_block_choice_tbs_ls <- function(block_ind,
                                     choices_tb){
  purrr::map(block_ind,
             ~ dplyr::filter(choices_tb, startsWith(Choice, paste0("set",.x,"."))))
}
make_choice_card <- function(choice_card_sng_tb){
  formatted_tb <- t(choice_card_sng_tb) %>%
    tibble::as_tibble(rownames = "Attribute") %>%
    dplyr::filter(Attribute != "Choice") %>%
    dplyr::rename(`Social Anxiety App 1` = V1,
                  `Social Anxiety App 2` = V2)
  row_names <- formatted_tb %>% dplyr::pull(Attribute)
  formatted_tb <- formatted_tb %>% dplyr::select(-Attribute)
  formatted_tb <- formatted_tb %>%
    as.data.frame()
  rownames(formatted_tb) <- row_names

  formatted_tb %>%
   knitr::kable(escape = F) %>%
    kableExtra::kable_styling(bootstrap_options = c("striped", "hover", "condensed", "responsive"),
                              full_width = F,
                              position = "left") %>%
    kableExtra::column_spec(1,bold = T, border_right = T) %>%
    kableExtra::column_spec(2:3,
                            # width = "40em",
                            color = "black", border_right = T)
}
make_one_block_choice_cards_ls <- function(block_choice_tbs_ls){
  purrr::map(1:length(block_choice_tbs_ls),
```

```
            ~ make_choice_card(block_choice_tbs_ls %>% purrr::pluck(.x)))
}
save_one_block_choice_cards <- function(block_choice_cards_ls,
                                        save_path_stub,
                                        output_type = ".png"){

  purrr::walk2(block_choice_cards_ls,
               1:length(block_choice_cards_ls),
               ~ save_choice_card_as(.x,
                                     .y,
                                     save_path_stub = save_path_stub,
                                     output_type = output_type)
               )
}
save_choice_card_as <- function(choice_kab,
                                choice_nbr,
                                save_path_stub,
                                output_type){
  file_path = paste0(save_path_stub,
                     "/choice_",
                     choice_nbr,
                     output_type)
  if(output_type==".png"){
    kableExtra::as_image(choice_kab,
                         file = file_path)
  }else{
    kableExtra::save_kable(choice_kab, file = file_path)
  }
}
```


4 Specify choice attributes and levels

We can now define the attributes and levels for the choices to be included in the survey.

```
attribute_levels <- list(Outcomes = c("Provides knowledge and skills to  manage future situations",
                                      "Addresses current symptoms",
                                      "Addresses current symptoms and provides knowledge and skills  to manage future situations"),
```

```
                          Information_sharing = c("No information is shared with your treating clinician",
                                                  "Information is shared with your treating clinician in accordance with app policy",
                                                  "Information is shared with your treating clinician based on settings you control"),
                          Social = c("No discussions with other app users",
                                     "Unmoderated discussions with other app users",
                                     "Discussions with other app users moderated by trained peers",
                                     "Discussions with other app users moderated by mental health clinicians",
                                     "Discussions with other app users moderated by both trained peers and mental health clinicians"),
                          Endorsers = c("App has no endorsers",
                                        "App is endorsed by respected non experts",
                                        "App is endorsed by youth mental health experts"),
                          Cost = c(0,
                                   5,
                                   15,
                                   30,
                                   60))
```


5 Create candidate choices matrix

We now create a design matrix of the full factorial of all attribute / level combinations. To do this we create a list specifying the number of attributes for each level and record whether coefficients for each attribute are Continuous or Dummy.

```
at_lvls <- purrr::map_int(attribute_levels,~length(.x)) %>%
  unname()
c_type <- c("D","D","D","D","C")
con_lvls <- purrr::keep(attribute_levels, is.numeric) %>%
  unname()
candidate_des_mat <- idefix::Profiles(lvls = at_lvls,
                                      coding = c_type,
                                      c.lvls = con_lvls)
```

6 Specify coefficient priors

We now specify our prior expectation of the values of coefficients for each attribute and an opt out constant. The coefficients supplied in this section were based on study authors' perception of participant feedback at a number of focus groups.

```
mu <- c(-0.15, # Opt out constant
        0.5,1, # Outcomes
        0.2,0.4, # Information sharing
        0.1,0.2,0.3,0.4, # Social
        0.1,0.2,# Endorsers
        -0.05) # Cost
v <- diag(length(mu)) ## Prior variance matrix
```

7 Create matrices of parameter value draws

We now create a matrix comprised of ten draws for each parameter and then split the matrix into two: one for the alternative specific constant, the other for the coefficients.

```
set.seed(1987)
pd <- MASS::mvrnorm(n = 10, mu = mu, Sigma = v)
p.d <- list(matrix(pd[,1], ncol = 1), pd[,2:12])
```


8 Specifiy additional features about our survey

We now provide additional detail about our intended survey design, specifying the total number of choice sets (30 - two blocks of 15), the number of alternatives (3 - including opt out), the position of the opt out option (third of three options) and the alternative specific constants (zero for each active option, one for the opt out option).

```
n_sets <- 30
n_alts <- 3
no_choice_idx <- 3
no_choice_lgl <- TRUE
alt_cte <- c(0,0,1)
```

9 Create efficient pilot survey design

We can now create the initial efficient design to be used in the pilot survey. Note this step can take a long time (about an hour).

```
no_app_optout_ls <- idefix::Modfed(cand.set = candidate_des_mat,
                                   n.sets = n_sets,
                                   n.alts = n_alts,
                                   no.choice = no_choice_lgl,
                                   alt.cte = alt_cte,
                                   parallel = FALSE,
                                   par.draws = p.d)
```

10 Translate design matrix into survey choices

The first step to translating the design matrix into survey content is to add text that specifies the currency and frequency of payment associated with the cost attribute.

```
lvl_names <- attribute_levels %>%
  purrr::discard(is.numeric) %>%
  append(list(paste("$",
                    as.character(attribute_levels[[5]]),
                    " per month")) %>%
           stats::setNames("Cost"))
```

We can now create a survey list object.

```
survey_ls <- idefix::Decode(des = no_app_optout_ls$design,
                            lvl.names = lvl_names,
                            coding = c_type,
                            c.lvls = con_lvls,
                            alt.cte = alt_cte,
                            n.alts = n_alts,
                            no.choice = no_choice_idx)
```

That list object contains a table of survey choice sets. We create a copy of that object, dropping rows relating to the opt-out choice.

```
choices_tb <- tibble::as_tibble(survey_ls$design, rownames = "Choice") %>%
  dplyr::rename(Outcomes = V1,
                `Information sharing` = V2,
                Social = V3,
```

```
                Endorsers = V4,
                Cost = V5) %>%
  dplyr::filter(!startsWith(Choice, "no"))
```

11 Assign survey choices to survey blocks

We randomly assign all thirty choice sets to one of two blocks (15 choice sets per block).

```
block_1_ind <- sample(1:30,15) %>% sort()
block_2_ind <- setdiff(1:30,block_1_ind)
blocks_choice_tbs_ls_ls <- purrr::map(list(block_1_ind,
                                   block_2_ind),
                              ~ make_block_choice_tbs_ls(.x,choices_tb))
```

12 Create choice cards for each block

We can now generate HTML choice cards for each block.

```
choice_cards_by_block_ls <- purrr::map(blocks_choice_tbs_ls_ls,
                                       ~ make_one_block_choice_cards_ls(.x))
```

13 Save work locally

We now back-up our work to a local directory.

```
if(!dir.exists("../Data/Pilot"))
  dir.create("../Data/Pilot")
dir.create("../Data/Pilot/block_1")
dir.create("../Data/Pilot/block_2")
saveRDS(no_app_optout_ls,"../Data/Pilot/no_app_optout_ls.rds")
saveRDS(survey_ls,"../Data/Pilot/survey_ls.rds")
saveRDS(choices_tb,"../Data/Pilot/choices_tb.rds")
purrr::walk2(choice_cards_by_block_ls,
```

```
            paste0("../Data/Pilot/block_",c(1:length(choice_cards_by_block_ls))),
            ~ {
              save_one_block_choice_cards(.x,
                                          .y,
                                          output_type = ".html")

              files_chr <- dir(.y, full.names = TRUE)
              fl_nm_1L_chr <- paste0(.y,"pilot.zip")
              utils::zip(zipfile = fl_nm_1L_chr,
                         files = files_chr)
                  })
```