An R Markdown program to analyse responses to a Discrete Choice Experiment (DCE) survey exploring the online help seeking preferences of socially anxious young people

Complete Analysis Program (reproduction)

Matthew P Hamilton[1,*]

27 October 2022

[1] Orygen, Parkville, Australia

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

Suggested citation: "Matthew Hamilton (2022). dce_sa_analysis: An R Markdown program to analyse responses to a Discrete Choice Experiment (DCE) survey exploring online help seeking in socially anxious young people. Zenodo. https://doi.org/10.5281/zenodo.7223286. Version 0.0.9.1"

1 About this code

1.1 Purpose

This program analyses response data for a Discrete Choice Experiment study that is currently being written up. Future versions of this program will include details of the parent study.

The analysis algorithm implemented by this code performs the following tasks:

- ingest and pre-process survey datasets;
- generate descriptive statistics and plots;
- develop and test choice models; and
- apply choice models to multiple analytic tasks.

1.2 Status

This code is the same version as was used to analyse study data. The only items that have been modified are those that remove references to the local directory structure on which study data was stored.

1.3 Use

This code can be run either by "knitting" the parent RMD file or by manually executing each code chunk. The code as is currently optimised for being knit in one step, which means that in a number of instance the value "Y" has been supplied to the `consent_1L_chr` argument of program functions. Supplying this value overrides the default behaviour of functions that write files which is to prompt users for their active consent prior to to write files and directories to their machine. If running this code interactively, we recommend omitting the `consent_1L_chr` argument as this will provide you with greater transparency about what files and directories are being written to your machine.

2 Prepare workspace

2.1 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.

```
# devtools::install_github("ready4-dev/ready4")
# devtools::install_github("ready4-dev/mychoice") # add lwgeom to imports
```

Next we load the libraries required to run this program.

```
library(ready4)
library(mychoice)
library(ready4use)
```

2.2 Specify data directories

We begin by specifying where our input data can be located and where we wish to write our outputs to. You must supply these details or the rest of this code will not work.

```
paths_ls <- list(input_data_dir_1L_chr = "PROVIDE DETAILS HERE",
                 output_data_dir_1L_chr = "PROVIDE DETAILS HERE",
                 raw_data_fl_nms_chr = "PROVIDE DETAILS HERE")
```

2.3 Reproducibility

We now set a seed to aid reproducibility.

```
set.seed(1001)
```

Having set the seed, it is now likely that if you run the syntax described in this document on your own installation of R you will get identical results to those reported in this document. However, if you do not, it may be that you have a different version of R, or of some of the packages that we used to produce this analysis. We therefore save a record of the software that we have on the machine used for this analysis so this can be made available for comparisons.

```
session_ls <- sessionInfo()
```

3 Create custom functions

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

4 Data ingest and processing

4.1 Create preprocessing log

We create an object into which we will save key milestones from our pre-processing algorithm.

```
preprocessing_log_ls <- list(session_ls = session_ls)
```

4.2 Ingest and validate data

4.2.1 Ingest data on choice features

We next retrieve data on the choices that participants were presented with from the code we previously ran to create the choice cards.

```
X <- Ready4useRepos(dv_nm_1L_chr = "springtolife",
                    dv_ds_nm_1L_chr = "https://doi.org/10.7910/DVN/VGPIPS",
                    dv_server_1L_chr = "dataverse.harvard.edu")
dce_design_ls <- ingest(X,
                        fls_to_ingest_chr = "DDD_final_dce_design_ls",
                        metadata_1L_lgl = F)
```

4.2.2 Import raw responses

We have three databases of participant responses - one from individuals that were sent a survey invite because they had participated in the Entourage pilot study and two of respondents to publicly advertised invitations. The reason for having two databases of responses to public invitations is that during the initial phases of our survey it became apparent that an individual or individuals was/were trying to game the survey in response to the survey participation incentive. To address this we temporarily paused data collection before relaunching the survey with additional security features. One of these additional features was the addition of one free text question in the relaunched public invitation version, which was used solely for the purposes of helping to screen out potential non-genuine respondents. Other than this additional screening question, the surveys were identical for all three sets of respondents.

The following commands ingest and merge the three databases in their raw format.

```
records_ls <- bind_tables_from_loc_files(paste0(paths_ls$input_data_dir_1L_chr,
                                                paste0("/",paths_ls$raw_data_fl_nms_chr)),
                                         force_numeric_1L_lgl = T, force_tb_1L_lgl = T, heading_rows_1L_int = 3L) %>%
  make_records_ls(choice_vars_pfx_1L_chr = "DCE_B")
```

```
preprocessing_log_ls <- list(raw_case_nbrs_1L_int = nrow(records_ls$ds_tb))
```

4.2.3 Drop missing choice responses

We next drop all records for which no choice card questions were answered.

```
records_ls$ds_tb <- remove_no_choice_responses(records_ls$ds_tb, choice_card_pfx_1L_chr = records_ls$choice_vars_pfx_1L_chr)
```

```
preprocessing_log_ls$raw_choice_resps_1L_int <- nrow(records_ls$ds_tb)
```

4.2.4 Restrict each dataset to valid responses

We next want to restrict response data to valid responses by dropping responses assessable as non-genuine. To do this we combine qualitative and automated checks.

Qualitative assessments were undertaken by one study investigator who reviewed participant free text responses, creating a dataset (`qltv_ax_tb`) in which flags ("green" or "red") were added to any individual responses that the reviewer had high degrees of confidence were likely to be genuine or not-genuine. Examples of where responses were red-flagged were for free-text entries that were either nonsensical or which poorly addressed the survey question with highly generic responses that may have been cut-and-pasted from published resources on social anxiety.

Automated checks added red flags if:

- a participant's stated age was inconsistent with the age calculated by combining their stated date of birth and the date of their response;
- the survey was completed in under 5 minutes;
- the participants submitted an email address which included 5 or more consecutive digits;
- different individuals supplied identical free text respondents; and
- the geo-code associated with a respondents' IP address was from outside Australia.

```
records_ls <- add_flags(records_ls,
                        qltv_ax_tb = {if(file.exists(paste0(paths_ls$input_data_dir_1L_chr,"/qltv_ax_tb.RDS"))){
                          qltv_ax_tb <- readRDS(paste0(paths_ls$input_data_dir_1L_chr,"/qltv_ax_tb.RDS"))}else{
                            qltv_ax_tb <- NULL}},
                        # Variables for age inconsistency red flag
                        age_var_nm_1L_chr = "Ent_DCE_Age",
                        date_stamp_format_1L_chr = "DMY",
                        date_stamp_var_nm_1L_chr = "StartDate",
```

```
                        dob_format_1L_chr = "DMY",
                        dob_var_nm_1L_chr = "Ent_DCE_DOB",
                        # Variables for too short completion time red flag
                        attempt_dur_min_1L_dbl = 300, #
                        attempt_dur_var_nm_1L_chr = "Duration (in seconds)",
                        # Variables for unusual email address red flag
                        email_max_cnstv_digits_1L_int = 5,
                        email_var_nm_1L_chr = "Ent_DCE_TY_1",
                        # Variables for ineligbile location red flag
                        lat_lng_var_nms_chr = c("LocationLongitude", "LocationLatitude"),
                        valid_countries_chr = "Australia",
                        # Variables for duplicate responses red flag
                        unique_by_case_var_nms_chr = "Ent_DCE_SAsupport",
                        # Variables for qualitative assessment flags
                        flags_max_1L_int = 0,
                        qltv_ax_green_flag_1L_chr = "Green",
                        qltv_ax_red_flag_1L_chr = "Red",
                        qltv_ax_var_nm_1L_chr = "Colour_code")
```



```
records_ls$ds_tb <- rlang::exec(add_red_flags, records_ls$ds_tb,
                                !!!(records_ls$flags_ls %>% purrr::list_modify("flags_max_1L_int" = NULL,
                                                                               "qltv_ax_green_flag_1L_chr" = NULL)))
```

```
preprocessing_log_ls <- make_flags_smry_ls(records_ls$ds_tb,
                                           qltv_ax_green_flag_1L_chr = records_ls$flags_ls$qltv_ax_green_flag_1L_chr,
                                           qltv_ax_var_nm_1L_chr = records_ls$flags_ls$qltv_ax_var_nm_1L_chr,
                                           preprocessing_log_ls = preprocessing_log_ls,
                                           flags_max_1L_int = records_ls$flags_ls$flags_max_1L_int)
```

We then dropped responses that had more than one red flag, unless it had been green-flagged in the qualitative assessment.

```
records_ls$ds_tb <- records_ls$ds_tb %>%
     remove_red_flag_cases(flags_max_1L_int = records_ls$flags_ls$flags_max_1L_int,
                           qltv_ax_green_flag_1L_chr = records_ls$flags_ls$qltv_ax_green_flag_1L_chr,
                           qltv_ax_var_nm_1L_chr = records_ls$flags_ls$qltv_ax_var_nm_1L_chr)
```

```
preprocessing_log_ls$raw_unflagged_1L_int <- nrow(records_ls$ds_tb)
```

4.3 Pre-process merged dataset of valid responses

The merged dataset of valid responses we created in the last step is not yet ready for analysis. To prepare the dataset for the analyses that we want to run we have to add information about the design of the survey, perform some transformations on the individual characteristics of survey respondents and reformat our data.

4.3.1 Prepare respondent characteristic data

We first need to create a number of new, derived or recoded or variables relating to respondent characteristics. We therefore appy a transformation function that performs the following tasks:

- creates a dichotomous age variable (under 20 years of age);
- creates dummy gender variables (Male and Other, Prefer Not To Say) with Female as the base gender category;
- calculate total SIAS scores and create Normal Range (under 34 total SIAS score) and Social Anxiety (SIAS scores above 43) dummy SIAS variables with Social Phobia (34-42 total SIAS score) the base category;
- create a SEIFA disadvantage quartile variable (based on participant postcode) and dummy variables for SEIFA disadvantage quartiles 1,2 and 4 (base category, SEIFA disadvantage quartile 3);

- create a State and Territory variable (based on participant postcode) and dummy variables for Victoria, Queensland, South Australia, Western Australia, ACT and Tasmania (base category New South Wales); and
- drop unnecessary variables.

```
preprocessing_log_ls$transformation_fn_ls <- list(fn = transform_repln_ds_for_analysis,
                                                  args_ls = list(consent_1L_chr = "Y",
                                                                 write_to_1L_chr = paths_ls$Replication))
```

```
records_ls$ds_tb <- rlang::exec(preprocessing_log_ls$transformation_fn_ls$fn,
                                records_ls$ds_tb,
                                !!!preprocessing_log_ls$transformation_fn_ls$args_ls)
```

4.3.2 Create database of respondent choices

The next step is to extract participant's responses to the choice cards as a matrix.

The design matrix we loaded in a previous step is not yet in the correct order. That is because we randomly sampled from that design matrix to create the choice cards in each block. We therefore need to reorder our design matrix to match how choices were presented to participants in the survey. Our survey included an opt out choice option on each choice card. We need to identify that variable prior to analysi by adding an alternative specific constant. Our database of valid responses is currently in a wide format - each case contains data on all choices presented in each choice card. For the analyses we plan on running, our choice response matrix needs to be in long format - each choice in a choice set needs to be its own case. We also need to selectively apply transformations to attribute variables depending on whether they are continuous or factors. In our survey, only the cost attribute is a continuous variable. All the factor attributes require the creation of dummy variables.

```
records_ls$case_choices_mat <- make_case_choices_mat(records_ls$ds_tb,
                                                     block_idxs_ls = dce_design_ls$choice_cards_ls[[length(dce_design_ls$choice_cards_ls)]
                                                     choice_sets_ls = dce_design_ls$choice_sets_ls,
                                                     design_mat = dce_design_ls$efnt_dsn_ls[[length(dce_design_ls$efnt_dsn_ls)]]$design,
                                                     choice_vars_pfx_1L_chr = records_ls$choice_vars_pfx_1L_chr)
```

```
preprocessing_log_ls$cases_wide_1L_int <- nrow(records_ls$case_choices_mat)
```

4.3.3 Specify modelling parameters

We now specify detail about the variables that we plan to use in our choice modelling.

```
mdl_params_ls <- tibble::tibble(domain_chr = c(rep("demographic",5),
                                               rep("clinical",4),
                                               rep("spatial",13),
                                               rep("study",5)),
                                concept_chr = c(rep("age",2),
                                                rep("gender",3),
                                                rep("SIAS",4),
                                                rep("area disadvantage",4),
                                                rep("urbanicity",2),
                                                rep("jurisdiction",7),
                                                "pilot participant",
                                                rep("incomplete choice tasks",2),
                                                rep("time taken",2)),
                                type_chr = c("continuous", "logical",
                                             "factor", "dummy", "dummy",
                                             "continuous", "factor", "dummy", "dummy",
                                             "factor",
```

```
                                        "dummy", "dummy","dummy",
                                        "factor", "logical",
                                        "factor", "dummy", "dummy","dummy", "dummy",
                                        "dummy", "dummy",
                                        "logical",
                                        "continuous","logical",
                                        "continuous", "logical"),
                            short_name_chr = c("rc_age", "der_age_u20",
                                               "der_gender", "der_gender_male", "der_gender_opnts",
                                               "der_SIAS_ttl", "der_SIAS_ctg", "der_SIAS_nr", "der_SIAS_sa",
                                               "der_SEIFA_Quartile",
                                               "der_SEIFA_Q1", "der_SEIFA_Q2", "der_SEIFA_Q4",
                                               "der_SOS", "der_urban",
                                               "der_STE", "der_STE_VIC","der_STE_QLD","der_STE_SA","der_STE_WA",
                                               "der_STE_ACT","der_STE_TAS",
                                               "rc_pilot_participant",
                                               "der_Missing_Tasks","der_All_Tasks",
                                               "rc_time_taken_secs", "der_under_ten_mins"),
                            long_name_chr = c("age","aged 15-19",
                                              "gender", "male gender", "other gender or prefer not to say",
                                              "total SIAS score", "overal SIAS category", "SIAS normal range", "SIAS social anxiety",
                                              "SEIFA disadvantage (by postcode, national comparisons) quartile",
                                              "SEIFA Quartile 1", "SEIFA Quartile 2", "SEIFA Quartile 4",
                                              "Section of State", "resides in an urban area",
                                              "State or Territory", "Victoria", "Queensland","South Australia","Western Australia",
                                              "Australian Capital Territory", "Tasmania",
                                              "Entourage pilot programme participant",
                                              "number of incomplete choice tasks","completed all choice tasks",
                                              "time taken to complete survey in seconds", "completed survey in under ten minutes"
                                              )) %>%
  dplyr::mutate(false_value_chr = dplyr::case_when(long_name_chr == "aged 15-19" ~ "aged 20-25",
                                                   long_name_chr == "resides in an urban area" ~ "resides in a rural or remote area",
                                                   long_name_chr == "Entourage pilot programme participant" ~ "responded to public adverti
                                                   long_name_chr == "completed all choice tasks" ~ "did not complete all choice tasks",
                                                   long_name_chr == "completed survey in under ten minutes" ~ "took at least ten minutes t
                                                   T ~ NA_character_)) %>%
  dplyr::mutate(units_chr = dplyr::case_when(short_name_chr == "rc_time_taken_secs" ~ "seconds",
                                             T ~ NA_character_)) %>%
```

```
make_mdl_params_ls(as_selection_ls_1L_lgl = T,
                   concepts_chr = c("age","gender","SIAS",
                                    "urbanicity", "jurisdiction", "area disadvantage",
                                    "pilot participant", "incomplete choice tasks", "time taken" ),
                   types_chr = c("logical","dummy","dummy",
                                 "logical", "dummy", "dummy","logical",
                                 "logical",
                                 "logical"),
                   dce_design_ls = dce_design_ls,
                   records_ls = records_ls)
```

4.3.4 Create database of respondents

We next create a composite, indexed database that combines the respondents' characteristics that we will use in choice modelling, the choice sets presented to respondents and their choices.

```
records_ls$ds_dfidx <- make_choice_mdlng_ds(case_choices_mat = records_ls$case_choices_mat,
                                            candidate_predrs_tb = mdl_params_ls$candidate_predrs_tb,
                                            card_id_var_nm_1L_chr = records_ls$card_id_var_nm_1L_chr,
                                            choice_sets_ls = dce_design_ls$choice_sets_ls,
                                            ds_tb = records_ls$ds_tb,
                                            person_card_uid_var_nm_1L_chr = records_ls$person_card_uid_var_nm_1L_chr,
                                            person_uid_var_nm_1L_chr = records_ls$person_uid_var_nm_1L_chr,
                                            concepts_chr = character(0),
                                            types_chr = character(0),
                                            as_selection_ls_1L_lgl = F)
```

5 Describe data

We begin by creating an object to store the descriptive statistics we generate.

```
descriptives_ls <- list()
```

We create a summary table of descriptive statistics.

```
descriptives_ls$summary_tb <- records_ls$ds_tb %>%
  make_smry_tb(var_metadata_tb = mdl_params_ls$candidate_predrs_tb, # Investigate alternative call formulations
               digits_1L_int = 2L)
```

We compare selected descriptive statistics from our sample with expected population values.

```
descriptives_ls$prpn_cmprsns_ls <- add_age_and_area_cmprsns(list(),
                                          ds_tb = records_ls$ds_tb,
                                          consent_1L_chr = "Y",
                                          write_to_1L_chr = paths_ls$Replication) %>%
  add_cut_pnts_cmprsn(cmprsn_nm_1L_chr = "SEIFA_cmprsn_tb",
                      ds_tb = records_ls$ds_tb,
                      grouping_var_nms_chr = c("der_SEIFA_Quartile", "SEIFA Disadvantage Quartile"),
                      expected_dbl = 25,
                      is_pc_1L_lgl = T) %>%
  add_cut_pnts_cmprsn(cmprsn_nm_1L_chr = "urban_cmprsn_tb",
                      ds_tb = records_ls$ds_tb,
                      grouping_var_nms_chr = c("der_urban", "Resides in urban area"),
                      expected_dbl = (make_sos_lup(write_to_1L_chr = paths_ls$Replication) %>% dplyr::pull(share_dbl))[1:2] * 100,
                      is_pc_1L_lgl = T)
```

We generate summary tables and plots of choice responses by attribute.

```
descriptives_ls$choice_smrys_ls <- make_choice_smrys(dce_design_ls,
                                                     mdl_params_ls = mdl_params_ls,
                                                     records_ls = records_ls,
                                                     text_size_1L_int = 7,
                                                     wrapping_1L_int = 20)
```

6 Fit choice models

We first create objects to log store all the models that we will fit and to log key modelling milestones.

```
mdls_ls <- list()
```

```
mdlng_log_ls <- list(session_ls = session_ls)
```

6.1 Model how Social Anxiety app attributes shape the choice behaviour of young people

In the first set of models we fit, we are only interested in exploring the role of the attributes of alternatives on participant choices and ignore respondent heterogeneity. We therefore begin by fitting a basic multinomial logit model. We create two versions of this model - one fitted using the gmnl package and the other using the mlogit package. The produce almost identical results, but each class of output has different distinctive methods defined for it, which means that each type has ease of use advantages over the other depending on the purpose to which the model will be applied.

```
mdls_ls <- add_choice_mdls(mdls_ls,
                           dce_design_ls = dce_design_ls,
                           mdl_params_ls = mdl_params_ls,
                           records_ls = records_ls,
                           purpose_chr = "attributes")
```

6.2 Model the heterogeneity of young people's preferences for Social Anxiety app attributes

The next set of models to estimate are those that include random parameters to explore how preferences vary due to respondent heterogeneity.

We fit a mixed logit model that includes random parameters. At this stage we are looking at overall heterogeneity for each parameter and are not exploring the role of specific individual characteristics. Again we create two versions of this model - one with the gmnl package and the other with the mlogit package. We also fit a Generalized Multinomial Logit Model using the gmnl package.

```
mdls_ls <- add_choice_mdls(mdls_ls = mdls_ls,
                           dce_design_ls = dce_design_ls,
                           mdl_params_ls = mdl_params_ls,
                           records_ls = records_ls,
                           purpose_chr = "heterogeneity")
```

6.3 Model interactions between respondent characteristics and Social Anxiety app attributes

We next want to investigate the role of individual respondent characteristics on the coefficients for choice attributes.

Our first step is to explore which characteristics are influential for each choice feature. We do this by sequentially estimating models that interact all respondent characteristics with a specified choice feature. We then identify which respondent characteristics are significant below a specified threshold (in this case 0.05). We include a specified maximum number (in our case 3) of characteristics that were most frequently found to be significant predictors for two or more choice features in models that includes all choice features. For the final models, we excluded the characteristics that relate to respondents' participation in the survey as these are not something that could be meaningfully used in real world applications of the choice model.

```
mdls_ls <- add_choice_mdls(mdls_ls = mdls_ls,
                           dce_design_ls = dce_design_ls,
                           mdl_params_ls = mdl_params_ls,
                           records_ls = records_ls,
                           exclude_chr = c("pilot participant","incomplete choice tasks", "time taken"),
                           max_concepts_1L_int = 2L,
                           min_threshold_1L_int = 2L,
                           purpose_chr = "interactions",
                           significant_at_1L_dbl = 0.05)
```


6.4 Model Social Anxiety app preference based sub-groups of young people

We are also interested if it is possible to identify subgroups that are distinguishable based on the nature of their preferences. To do this we estimate Latent Class models.

6.4.1 Basic Latent Class Models

The first model we estimate seeks to identify two different preference based classes. The second adds respondent characteristic predictors, to identify what characteristics predict membership of which class.

```
mdls_ls <- add_choice_mdls(mdls_ls = mdls_ls,
                           dce_design_ls = dce_design_ls,
                           mdl_params_ls = mdl_params_ls,
                           include_int = 1:2,
                           records_ls = records_ls,
                           purpose_chr = "classes",
                           nbr_of_clss_1L_int = 2L)
```

6.4.2 Mixed-Mixed Multinomial Logit Model

We next fit a model that adds correlated random parameters to the latent classes.

```
mdls_ls <- add_choice_mdls(mdls_ls = mdls_ls,
                           dce_design_ls = dce_design_ls,
                           mdl_params_ls = mdl_params_ls,
                           include_int = 4,
                           records_ls = records_ls,
                           purpose_chr = "classes",
                           nbr_of_clss_1L_int = 2L,
                           iterlim = 500,
                           method = "bfgsr") # Method that produces fewest NANs
```

6.5 Model Willingness to Pay of young people for distinct attributes of a Social Anxiety App

Our final group of models are designed to facilitate estimates of Willingness To Pay (WTP) values for Social Anxiety App attributes.

6.5.1 Scaled Multinomial Logit Model

We begin by estimating a Scaled Multinomial Logit Model model that we can use later to generate WTP estimates directly from WTP space.

```
mdls_ls <- add_choice_mdls(mdls_ls = mdls_ls,
                           dce_design_ls = dce_design_ls,
                           mdl_params_ls = mdl_params_ls,
                           include_int = 1,
                           records_ls = records_ls,
                           purpose_chr = "wtp",
                           iterlim = 500,
                           method = "bhhh")
```

6.5.2 Generalised Multinomial Logit Model

We next generate an GMNL model with fixed Cost parameters and correlated random parameters that allows WTP to vary across individuals.

```r
mdls_ls <- add_choice_mdls(mdls_ls = mdls_ls,
                           dce_design_ls = dce_design_ls,
                           mdl_params_ls = mdl_params_ls,
                           include_int = 3,
                           records_ls = records_ls,
                           purpose_chr = "wtp")
```

7 Apply choice models

7.1 Calculate Willingness To Pay

Our next step is to calculate Willingness To Pay values for the different attributes of social anxiety apps. We can do this both from utililty space or from willingness to pay space.

```
analysis_ls <- add_analysis(mdls_ls = mdls_ls,
                            records_ls = records_ls,
                            what_chr = "wtp")
```

7.2 Predict market share

A core motivation for the Entourage DCE was to explore the potential viability of a user-pays financing model for a future roll-out of the Entourage App. This concluding section of the report aims to address this issue by using the models previously estimated to predict what market share Entourage could secure compared to a competing app representative of current offerings and not using an app at all. We do this using two alternative approaches. The first approach is based on the (strong) assumption that our sample is representative of our target market and therefore ignores respondent characteristics. The second includes individual characteristic predictors to make predictions that account for the characteristics of our target market.

7.2.1 Predict from existing choice situations

We will use the multinomial logit model we estimated earlier to make predictions about the market share of each option in a given choice situation, assuming that the preferences from our sample are representative of those of our target market.

The first set of predictions we make are of the market shares of the choice situations presented to survey respondents.

```
analysis_ls <- add_analysis(mdls_ls,
                            analysis_ls = analysis_ls,
                            dce_design_ls = dce_design_ls,
                            records_ls = records_ls,
                            what_chr = "share")
```

7.2.2 Predict from custom choice situations

The previous step had the significant limitation of predictions being confined to choice situations that were included in the survey. To explore the situations that might apply in the roll-out of the Entourage app we need to generate some new choice situations. Our first step is to generate a choice

situation of the Entourage app as it is currently configured compared to an app representative of current market offerings and a no-choice option. We can then predict the potential market shares of the options in the new hypothetical choice situation. We make predictions using both the multinomial logit and mixed logit models.

```
analysis_ls <- add_new_choice_cmprsn(analysis_ls,
                                     dce_design_ls = dce_design_ls,
                                     mdl_params_ls = mdl_params_ls,
                                     new_choices_ls = list(list(Cost = 25, Endorsers = "expert", Information_sharing = NA_character_,
                                                                Outcomes = NA_character_, Social = "peer_clinician_mod"),
                                                           list(Cost = 0, Endorsers = NA_character_, Information_sharing = NA_character_,
                                                                Outcomes = NA_character_, Social = NA_character_)),
                                     records_ls = records_ls,
                                     altv_nms_chr = c("Entourage", "Competing App", "No App"),
                                     with_chr = c("mnl_mlogit_mdl","mixl_mlogit_mdl"))
```

We can then see how the predicted market share changes in response to variations in this choice situation (in the following example, we add a user controlled information sharing setting to the Entourage app, which results in an extra 6% of predicted market share).

```
analysis_ls <- add_new_choice_cmprsn(analysis_ls,
                                     dce_design_ls = dce_design_ls,
                                     mdl_params_ls = mdl_params_ls,
                                     new_choices_ls = list(list(Cost = 25, Endorsers = "expert", Information_sharing = "user_settings",
                                                                Outcomes = NA_character_, Social = "peer_clinician_mod"),
                                                           list(Cost = 0, Endorsers = NA_character_, Information_sharing = NA_character_,
                                                                Outcomes = NA_character_, Social = NA_character_)),
                                     records_ls = records_ls,
                                     altv_nms_chr = c("Entourage", "Competing App", "No App"),
                                     with_chr = c("mnl_mlogit_mdl","mixl_mlogit_mdl"))
```

We can do this iteratively for a range of values, for example by varying the price attribute of the Entourage app.

```
analysis_ls <- add_cost_comparison(analysis_ls,
                                   choices_ls = analysis_ls$new_data_ls$comparison_1_ls$new_choices_ls,
                                   dce_design_ls = dce_design_ls,
                                   mdls_ls = mdls_ls,
                                   mdl_params_ls = mdl_params_ls,
                                   records_ls = records_ls,
                                   cost_range_dbl = 25:50,
```

```
                                altv_nms_chr = c("Entourage", "Competing App", "No App"),
                                altv_to_modify_1L_int = 1,
                                with_chr = c("mnl_mlogit_mdl","mixl_mlogit_mdl"))
```

8 Write output files

We now save our output. If running this code interactively, we recommend omitting the `consent_1L_chr` argument.

```
paths_ls <- write_preprocessing_outp(paths_ls,
                                     mdl_params_ls = mdl_params_ls,
                                     preprocessing_log_ls = preprocessing_log_ls,
                                     records_ls = records_ls,
                                     consent_1L_chr = "Y")
```

```
write_obj_with_prompt(descriptives_ls,
                      obj_nm_1L_chr = "descriptives_ls",
                      outp_dir_1L_chr = paths_ls$Results,
                      consent_1L_chr = "Y")
```

```
paths_ls <- write_choice_mdlng_ws(paths_ls) # Not giving prompt.
```

```
write_obj_with_prompt(mdlng_log_ls,
                      obj_nm_1L_chr = "mdlng_log_ls",
                      outp_dir_1L_chr = paths_ls$Replication,
                      consent_1L_chr = "Y")
```

```
write_obj_with_prompt(mdls_ls,
                      obj_nm_1L_chr = "mdls_ls",
                      outp_dir_1L_chr = paths_ls$Models,
                      consent_1L_chr = "Y")
```

```
write_obj_with_prompt(analysis_ls,
                      obj_nm_1L_chr = "analysis_ls",
                      outp_dir_1L_chr = paths_ls$Results,
                      consent_1L_chr = "Y")
```