Complete study program to reproduce all steps from data ingest through to results dissemination for a study to K10 (psychological distress) and SOFAS (functioning) to AQoL-6D and CHU-9D health utility

Matthew P Hamilton[3,1,*] Caroline X Gao[1,2,3]

30 June 2023

[1] Orygen, Parkville, Australia
[2] Centre for Youth Mental Health; University of Melbourne, Parkville, Australia
[3] School of Public Health and Preventive Medicine, Monash University, Clayton, Australia

[*] Correspondence: Matthew P Hamilton <matthew.hamilton1@monash.edu>

1 About this code

When using this code it is important to note:

1) 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 knit if you are using the kdown version of this code).

2) The code is implemented in R using ready4 modules for

- labeling, validating and summarising youth mental health datasets from the youthvars package;
- scoring health utility from the scorz package;
- specifying and testing statistical models from the specific package;
- generating reports from the ready4show package; and
- sharing data via online data repositories from the ready4use package; and
- implementing a utility mapping study from the TTU package.

3) The code in this program is highly abstracted which means that its detailed workings are not exposed. However, as all code is open source you are able to scrutinise the underlying code if you wish. The websites, manuals and repositories of the ready4 framework libraries used in this analysis are a good starting point.

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. Note, if you do not already have cmdstandr installed and configured you will also need to perform a minor additional configuration step as outlined on https://mc-stan.org/cmdstanr/ .

```r
# install.packages("cmdstanr", repos = c("https://mc-stan.org/r-packages/", getOption("repos")))
# devtools::install_github("ready4-dev/ready4")
# devtools::install_github("ready4-dev/TTU")
```

Next load the libraries required to run this program.

```r
library(ready4)
library(ready4show)
library(ready4use)
library(youthvars)
library(scorz)
library(TTU)
```

3 Ingest, transform, label and score data

3.1 Specify data locations

To start, we need to specify the local and remote locations for storing project input and output data. The real study dataset (for study implementation and reproduction) can be read from a local location and fake data that closely resembles the study dataset (which are useful for algorithm testing and demonstration) can be read from either local or remote (i.e. online repository) locations (note we have not yet publicly released a fake dataset to use with this program - contact us if you need one). Note, if using real data, the local output directory **must be a secure location** as copies of the input dataset will be saved in this directory. Non-confidential analyses outputs will be shared via online repositories. Different repositories will be used for outputs created with real or fake data and which map to AQOL-6D or CHU-9D health utility.

```
#path_to_data_1L_chr <- "****Uncomment and replace with path to local data****"
```

```
consent_1L_chr <- "Y"
cores_1L_int <- 1L # # Default value. For Mac users, consider using cores_1L_int = parallel::detectCores() (but not yet -
# there is a bug in the multicore implementation that needs to be fixed!).
is_real_1L_lgl <- F # Replace with T if working with real data
name_of_dataset_1L_chr <- "synthetic" # Only update if you are using fake data and want to use a different data source.
path_to_output_1L_chr <- normalizePath("../Data")
repositories_ls <- purrr::map2(c(rep("TTU",2),rep("fakes",3)),
                                c("https://doi.org/10.7910/DVN/FDRUXH", "https://doi.org/10.7910/DVN/N4NEHL",
                                  "https://doi.org/10.7910/DVN/LYBMBO", "https://doi.org/10.7910/DVN/3R5TS3",
                                  "https://doi.org/10.7910/DVN/HJXYKQ"),
                                ~ Ready4useRepos(dv_nm_1L_chr = .x, dv_ds_nm_1L_chr = .y,
                                                 dv_server_1L_chr = "dataverse.harvard.edu")) %>%
  stats::setNames(c("aqol_real_r4", "chu_real_r4","aqol_fake_r4", "chu_fake_r4","records_fake_r4"))
```

3.2 Ingest data

Based on the path value specified above, we can now ingest either the real study dataset or the synthetic (fake) data. We perform a number of data transformations (using the `transform_csnl_example_ds` function) to prepare the dataset for analysis and then use the `Ready4useDyad` module to pair the dataset with its data dictionary and label dataset variables.

```
A <- Ready4useDyad(ds_tb = {if(!is.na(path_to_data_1L_chr)){readRDS(path_to_data_1L_chr)}else{
  repositories_ls$records_fake_r4 %>%
    ingest(fls_to_ingest_chr = c(name_of_dataset_1L_chr), metadata_1L_lgl = F)}} %>% transform_csnl_example_ds(),
    dictionary_r3 = repositories_ls$aqol_real_r4 %>%
    ingest(fls_to_ingest_chr = c("ymh_clinical_aqol_chu_dict_r3"), metadata_1L_lgl = F)) %>% renew(type_1L_chr = "label")
```

We provide details of the unique identifier variable using the `YouthvarsProfile` module.

```
A <- YouthvarsProfile(a_Ready4useDyad = A, id_var_nm_1L_chr = "fkClientID")
```

3.3 Create utility mapping projects

We supply the dataset along with its metadata to two `TTUProject` modules - one to be used for mapping to adolescent AQoL-6D health utility and the other to map to CHU-9D health utility. Details about the former instrument can be automatically retrieved through use of `ScorzAqol6Adol` module. We change the default variable name for AQoL-6D health utility to the format preferred for predictors (all caps, no underscore) as this will be used as a candidate predictor for CHU-9D health utility later on. We manually record information about the CHU-9D instrument using the more generic `ScorzProfile` module. At the conclusion of this step we delete the object `A` that we previously created as it is no longer required.

```
projects_ls <- list(AQOL_r4 = TTUProject(a_ScorzProfile = ScorzAqol6Adol(a_YouthvarsProfile = A,
                                                                         total_wtd_var_nm_1L_chr = "AQOL6D")) %>%
                      renewSlot("a_ScorzProfile@instrument_dict_r3", .@a_ScorzProfile@instrument_dict_r3 %>%
                                  renew(filter_cdn_1L_chr = "var_nm_chr!='aqol6d_total_w'",
                                        var_nm_chr = "AQOL6D",  var_ctg_chr = "utility overall score (final weighted)",
                                        var_desc_chr = "AQOL-6D (weighted total)",
                                        var_type_chr = "numeric") %>% dplyr::arrange(var_ctg_chr, var_nm_chr)),
                    CHU_r4 = TTUProject(a_ScorzProfile = ScorzProfile(a_YouthvarsProfile = A,
                                                                      country_1L_chr = "Australia",
                                                                      domain_wtd_var_nms_chr =
                                                                        renew(A@a_Ready4useDyad@dictionary_r3,
                                                                                "startsWith(var_nm_chr,'chu9_')") %>%
                                                                        dplyr::pull(var_nm_chr) %>% as.character(),
                                                                      instrument_dict_r3 =
                                                                        renew(A@a_Ready4useDyad@dictionary_r3,
                                                                              filter_cdn_1L_chr =
                                                                                "startsWith(var_nm_chr,'chu9_')|var_nm_chr=='CHU9D'"),
                                                                      instrument_nm_1L_chr = "Child Health Utility (9 Dimension)",
                                                                      instrument_short_nm_1L_chr = "CHU-9D",
                                                                      itm_labels_chr = c("worry", "sadness", "pain", "tiredness",
                                                                                         "annoyance", "school", "sleep",
                                                                                         "daily routine", "activities"),
                                                                      itm_prefix_1L_chr = "chu9_",
                                                                      total_wtd_var_nm_1L_chr = "CHU9D",
                                                                      total_unwtd_var_nm_1L_chr = "CHU_c")))
rm(A)
```

3.4 Score health utility scores

We apply the multi-instrument total scoring algorithms associated with the `ScorzAqol6Adol` and `ScorzProfile` modules. The `ScorzAqol6Adol` scoring algorithm will calculate both weighted (utility) and unweighted totals. The `ScorzProfile` algorithm calculates only also the unweighted total of all CHU-9D item responses. The weighted CHU-9D total (utility) were already pre-calculated in the dataset we ingested.

```
projects_ls <- projects_ls %>% purrr::map(~renew(.x, what_1L_chr = "utility"))
```

We update the dataset copy with CHU-9D MAUI data so that it includes the AQoL totals that we calculated in the previous step (we need both instrument totals present in each dataset to facilitate comparison / mapping).

```
projects_ls$CHU_r4 <- renewSlot(projects_ls$CHU_r4,
                                "a_ScorzProfile@a_YouthvarsProfile@a_Ready4useDyad@ds_tb",
                                dplyr::left_join(procureSlot(projects_ls$CHU_r4,
                                                             "a_ScorzProfile@a_YouthvarsProfile@a_Ready4useDyad@ds_tb"),
                                                 procureSlot(projects_ls$AQOL_r4,
                                                             "a_ScorzProfile@a_YouthvarsProfile@a_Ready4useDyad@ds_tb") %>%
                                                   dplyr::select(projects_ls$AQOL_r4@a_ScorzProfile@a_YouthvarsProfile@id_var_nm_1L_chr,
                                                                 projects_ls$AQOL_r4@a_ScorzProfile@total_wtd_var_nm_1L_chr)))
```

Our source dataset included total weighted utility scores for both utility instruments. We can use the pre-calculated AQoL-6D weighted total (utility) to verify the scores we have just generated using the `ScorzAqol6Adol`. We therefore calculate the difference the pre-calculated and `ScorzAqol6Adol` scored AQoL-6D weighted totals.

```
projects_ls <- projects_ls %>% purrr::map(~renewSlot(.x, "a_ScorzProfile@a_YouthvarsProfile@a_Ready4useDyad@ds_tb",
                                                     procureSlot(.x, "a_ScorzProfile@a_YouthvarsProfile@a_Ready4useDyad@ds_tb") %>%
                                                       dplyr::mutate(difference_aqol_calcs = AQOL6D - validation_aqol_w)))
```

```
# To manually inspect the validation statistic, uncomment and run the following lines
# projects_ls$AQOL_r4@a_ScorzProfile@a_YouthvarsProfile@a_Ready4useDyad@ds_tb %>% dplyr::select(AQOL6D,
# validation_aqol_w, difference_aqol_calcs)
# projects_ls$AQOL_r4@a_ScorzProfile@a_YouthvarsProfile@a_Ready4useDyad@ds_tb$difference_aqol_calcs %>% summary()
```

4 Describe and analyse data

4.1 Specify modelling parameters

We begin by reorganising some of the metadata we have already specified about our dataset.

```
projects_ls <- projects_ls %>% purrr::map(~renew(.x, what_1L_chr = "parameters"))
```

We ingest from our online repositories tables describing the candidate predictors for each utility instrument that we will assess.

```
projects_ls <- projects_ls %>%
  purrr::map2(repositories_ls[c("aqol_real_r4", "chu_real_r4")],
              ~ renew(.x, "use_renew_mthd", fl_nm_1L_chr = "predictors_r3", type_1L_chr = "predictors_lup",
                      y_Ready4useRepos = .y, what_1L_chr = "parameters"))
```

We now add the required parameters for allowable range for utility scores, the variable names of candidate predictors to be explored in the primary analysis, candidate covariates, and the variables to use when preparing descriptive statistics.

```
projects_ls <- projects_ls %>%
  purrr::map2(c(0.03,-0.1059), # Check that this is minimum value for CHU-9D
              ~ renew(.x, c(.y, 1), type_1L_chr = "range", what_1L_chr = "parameters") %>%
                renew(c("K10", "SOFAS"),  type_1L_chr = "predictors_vars", what_1L_chr = "parameters") %>% #"MLT",
                renew(c("d_age", "d_gender", "d_ATSI", "d_CALD", "d_studying_working", "c_p_diag_grouped", "c_days_oor",
                        "s_IRSD","s_remoteness"), #"c_clinical_staging_s",
                      type_1L_chr = "covariates", what_1L_chr = "parameters") %>%
                renew(c("d_age", "d_gender", "d_ATSI", "d_CALD", "d_employment_type", "d_studying", "d_studying_working",
                        "c_days_cut_back", "c_days_unable", "c_days_oor", "c_p_diag_grouped", "s_IRSD", "s_remoteness"),
                      type_1L_chr = "descriptives", what_1L_chr = "parameters") %>% #"c_clinical_staging_s",
                renew(!is_real_1L_lgl, type_1L_chr = "is_fake", what_1L_chr = "parameters"))
```

4.2 Create local workspace

We create sub-directories where data generated by our analysis will be written and perform checks to confirm that our dataset and parameters are internally consistent.

```
projects_ls <- projects_ls %>%
  purrr::map2(c("AQOL","CHU"), ~ renew(.x, consent_1L_chr = consent_1L_chr, paths_chr = paste0(path_to_output_1L_chr,"/",.y),
                                       what_1L_chr = "project"))
```

4.3 Create descriptive tables and figures

We now generate tables and charts that describe our dataset. These are saved in a sub-directory of our output data directory.

```
projects_ls <- projects_ls %>%
  purrr::map2(c(F,T), ~ author(.x, consent_1L_chr = consent_1L_chr, digits_1L_int = 3L, items_as_domains_1L_lgl = .y,
                               what_1L_chr = "descriptives"))
```

4.4 Undertake exploratory modelling

4.4.1 Identify preferred model types

For each utility instrument, we next compare the performance of different model types. This step saves model objects and plots to a sub-directory of our output directory. When modelling, we constrain utility values to the range 0.00001 to 0.99999 (i.e. converting values outside this range to the nearest lower or upper bound) as this allows us to explore models with log transformations.

```
projects_ls <- projects_ls %>%
  purrr::map(~ investigate(.x, consent_1L_chr = consent_1L_chr, depnt_var_max_val_1L_dbl = 0.9999,
                           depnt_var_min_val_1L_dbl = 0.0001, session_ls = sessionInfo()))
```

You can inspect the performance of models by uncommenting and running the following:

```
# Not run
# exhibitSlot(projects_ls$AQOL_r4, "c_SpecificProject", what_1L_chr = "mdl_cmprsn", type_1L_chr = "results")
# exhibitSlot(projects_ls$CHU_r4, "c_SpecificProject", what_1L_chr = "mdl_cmprsn", type_1L_chr = "results")
```

To get the reference codes for the top performing models of each type you can uncomment and run:

```
# projects_ls %>% purrr::map(~procureSlot(.x,"c_SpecificProject", use_procure_mthd_1L_lgl = T, what_1L_chr = "prefd_mdls"))
```

If the default selections are not the models you plan on using you can instead look up the reference codes for other models from a table displayed when uncommenting and running:

```
# Not run
# projects_ls$AQOL_r4@b_SpecificParameters@candidate_mdls_lup[,1:2]
```

After inspecting the comparisons of the models we have tested, we now specify the preferred model types to use from this point onwards, using the relevant reference codes.

```
projects_ls <- projects_ls %>%
  purrr::map(~ renew(.x, c("GLM_GSN_LOG", "OLS_CLL"), type_1L_chr = "models", what_1L_chr = "results"))
```

4.4.2 Identify preferred predictors and covariates

If using factor variables as predictors you may wish to manually control the names of the dummy variables that will be generated for each factor as shorter names will be easier to read in summary tables. You can inspect the default dummy variable names that will be used for each factor candidate covariate by uncommenting and running:

```
# Not run
# manufacture(projects_ls$CHU_r4@c_SpecificProject@a_YouthvarsProfile@a_Ready4useDyad, type_1L_chr = "dummys",
# what_1L_chr = "factors", restrict_to_chr = projects_ls$CHU_r4@c_SpecificProject@b_SpecificParameters@candidate_covars_chr)
```

We rename some of the default values for the dummy variables that will be created.

```
projects_ls <- projects_ls %>%
  purrr::map(~ renewSlot(.x, "c_SpecificProject",
                         renew(.x@c_SpecificProject, list(cpdiaggrouped = c("NotApp","Other","SubstanceUse"),
                                                          dstudyingworking = c("Both","Study","Work"),
                                                          sremoteness = c("MajorCities","OuterRegional","RemoteVeryRemote")),
                               what_1L_chr = "dummys")))
```

Next we assess multiple versions of our preferred model type - one single predictor model for each of our candidate predictors and the same models with candidate covariates added. A number of model/plot objects are saved to a sub-directory of our output directory. We specify the criterion that for each utility measure, candidate covariates must be significant parameters in all preferred models (the “all” value for `signft_covars_cndn_1L_chr`) to be considered for subsequent steps. To relax this condition, supply “any” instead of “all”.

```
projects_ls <- projects_ls %>%
  purrr::map(~ investigate(.x, consent_1L_chr = consent_1L_chr, depnt_var_min_val_1L_dbl = 0.0001, signft_covars_cndn_1L_chr = "all"))
```

Uncommenting and running the following code will allow inspection of the performance of the predictors.

```
# Not run
# exhibit(projects_ls$AQOL_r4@c_SpecificProject, what_1L_chr = "predr_cmprsn", type_1L_chr = "results")
# exhibit(projects_ls$CHU_r4@c_SpecificProject, what_1L_chr = "predr_cmprsn", type_1L_chr = "results")
```

Uncommenting and running the following code will allow inspection of additional performance indicators of the predictors.

```
# Not run
# exhibit(projects_ls$AQOL_r4@c_SpecificProject, type_1L_chr = "results", what_1L_chr = "fxd_sngl_cmprsn")
# exhibit(projects_ls$CHU_r4@c_SpecificProject, type_1L_chr = "results", what_1L_chr = "fxd_sngl_cmprsn")
```

Uncommenting the following code will allow identification of significant covariates.

```
# Not run
# exhibit(projects_ls$AQOL_r4@c_SpecificProject, type_1L_chr = "results", what_1L_chr = "fxd_full_cmprsn")
# exhibit(projects_ls$CHU_r4@c_SpecificProject, type_1L_chr = "results", what_1L_chr = "fxd_full_cmprsn")
```

Covariates that were significant in all models for each instrument can be returned by uncommenting the following:

```
# Not run
# purrr::map(projects_ls, ~procure(.x@c_SpecificProject, type_1L_chr = "results", what_1L_chr = "signt_covars"))
```

4.4.3 Assess performance of final model specification

After reviewing the output of the previous step, we specify the covariates we wish to add to the models. In the below example, we use the covariates that were automatically identified as meeting the criterion we specified earlier.

```
projects_ls <- projects_ls %>%
  purrr::map(~ renew(.x, procure(.x@c_SpecificProject, type_1L_chr = "results", what_1L_chr = "signt_covars"),
                     type_1L_chr = "covariates", what_1L_chr = "results"))
```

Alternatively, we could have over-ridden the default selections and specified our own values, for example:

```
# Not run
# projects_ls <- projects_ls %>%
#   purrr::map(~ renew(.x,c("cdaysoor", "dage", if(inherits(.x@a_ScorzProfile,"ScorzAqol6Adol")){ "dgender"}),
# type_1L_chr = "covariates", what_1L_chr = "results"))
```

We now assess the multivariate models. More model/plot objects are saved to a sub-directory of our output directory.

```
projects_ls <- projects_ls %>%
  purrr::map(~ investigate(.x, consent_1L_chr = consent_1L_chr, depnt_var_min_val_1L_dbl = 0.00001))
```

4.5 Create bayesian models for use in prediction

We next reformulate the models we finalised in the previous step as bayesian models that we will use when making predictions.

4.5.1 Primary analysis

We first modify the default parameters used when creating the models to address potential convergence issues. Supplying a non-default value for `adapt_delta` that is close to but less than 1 (as we have done below) can reduce the risk of divergent transitions, though it may result in slower sampling speeds (i.e., it will take longer to execute subsequent steps with sampling).

```
projects_ls <- projects_ls %>%
  purrr::map(~ renewSlot(.x, "c_SpecificProject@b_SpecificParameters@control_ls", list(adapt_delta = 0.99)))
```

For our primary analysis, we use a baysesian formulation of the models we previously selected. A series of large model files are written to a sub-directory of the local output data directory. If not using multiple cores, this step will typically take a very long time (~20 hours) to execute.

```
projects_ls <- projects_ls %>%
  purrr::map(~ investigate(.x, combinations_1L_lgl = T, consent_1L_chr = consent_1L_chr, max_nbr_of_covars_1L_int = 2L,
                           depnt_var_min_val_1L_dbl = 0.00001))
```

This is a good time to save a copy of our project modules.

```
projects_ls <- projects_ls %>% purrr::map(~author(.x, consent_1L_chr = consent_1L_chr, what_1L_chr = "self"))
```

4.5.2 Exploring differences in utility instruments

To explore potential differences in the two utility instruments, we first create an additional project and workspace.

```
dir.create(paste0(path_to_output_1L_chr,"/Differences"))
A <- TTUProject(a_ScorzProfile = projects_ls$AQOL_r4@a_ScorzProfile) %>%
  renew(what_1L_chr = "parameters") %>%
  renew(projects_ls$AQOL_r4@b_SpecificParameters@predictors_lup, type_1L_chr = "predictors_lup", what_1L_chr = "parameters") %>% ##
  renew(projects_ls$AQOL_r4@b_SpecificParameters@candidate_predrs_chr, type_1L_chr = "predictors_vars", what_1L_chr = "parameters") %>%
  renew(projects_ls$AQOL_r4@b_SpecificParameters@candidate_covars_chr, type_1L_chr = "covariates", what_1L_chr = "parameters") %>%
  renew(projects_ls$AQOL_r4@b_SpecificParameters@fake_1L_lgl, type_1L_chr = "is_fake", what_1L_chr = "parameters") %>%
  renew(consent_1L_chr = consent_1L_chr, paths_chr = paste0(path_to_output_1L_chr,"/Differences"), what_1L_chr = "project")
```

```
A <- investigate(A, consent_1L_chr = consent_1L_chr, depnt_var_max_val_1L_dbl = 0.9999, session_ls = sessionInfo()) %>%
  renew(c("GLM_GSN_LOG", "OLS_CLL"), type_1L_chr = "models", what_1L_chr = "results") %>%
  investigate(consent_1L_chr = consent_1L_chr, signft_covars_cndn_1L_chr = "all")
```

We can inspect the initial results by uncommenting and running the following commands.

```
# exhibit(A@c_SpecificProject, what_1L_chr = "predr_cmprsn", type_1L_chr = "results")
```

```
# exhibit(A@c_SpecificProject, type_1L_chr = "results", what_1L_chr = "fxd_sngl_cmprsn")
```

```
# exhibit(A@c_SpecificProject, type_1L_chr = "results", what_1L_chr = "fxd_full_cmprsn")
```

```
# procure(A@c_SpecificProject, type_1L_chr = "results", what_1L_chr = "signt_covars")
```

We could choose the covariates already identified as significant in all models with the best performing candidate predictor. To do so we would uncomment and run the following:

```
# Not run
# A <- renew(A, procure(A@c_SpecificProject, type_1L_chr = "results", what_1L_chr = "signt_covars"),
# type_1L_chr = "covariates", what_1L_chr = "results")
```

However, in this case we have opted to specify a custom selection.

```
 A <- renew(A, c("cdaysoor", "dage", "dgender"), type_1L_chr = "covariates", what_1L_chr = "results")
```

We now explore how well these covariates help predict differences between the two utility instruments.

```r
A <- investigate(A, consent_1L_chr = consent_1L_chr)
```

Based on the above analysis of what predicts differences between AQoL-6D and CHU-9D in our sample, we can now run secondary analyses that specify and test models that map between CHU-9D and AQoL-6D.

```r
projects_ls$AQOL_r4 <- investigate(projects_ls$AQOL_r4, combinations_1L_lgl = F, consent_1L_chr = consent_1L_chr,
                                   scndry_anlys_params_ls = make_scndry_anlys_params(candidate_predrs_chr = c("CHU9D"),
                                                                                     candidate_covar_nms_chr = c("K10","SOFAS"),
                                                                                     prefd_covars_chr = NA_character_) %>%
                                     make_scndry_anlys_params(candidate_predrs_chr = c("CHU9D"),
                                                              candidate_covar_nms_chr = c("K10", "dage", "dgender"),
                                                              prefd_covars_chr = c("K10", "dage", "dgender")) %>%
                                     make_scndry_anlys_params(candidate_predrs_chr = c("CHU9D"),
                                                              candidate_covar_nms_chr = c("SOFAS", "dage", "dgender"),
                                                              prefd_covars_chr = c("SOFAS", "dage", "dgender")))
```

The corresponding step for mapping AQoL-6D and CHU-9D would be the following code. However, this does not currently execute and is commented out until the relevant bug is fixed.

```r
# projects_ls$CHU_r4 <- investigate(projects_ls$CHU_r4, combinations_1L_lgl = F, consent_1L_chr = consent_1L_chr,
#                                   scndry_anlys_params_ls = make_scndry_anlys_params(candidate_predrs_chr = c("AQOL6D"),
#                                                                                     candidate_covar_nms_chr = c("K10","SOFAS"),
#                                                                                     prefd_covars_chr = NA_character_) %>%
#                                     make_scndry_anlys_params(candidate_predrs_chr = c("AQOL6D"),
#                                                              candidate_covar_nms_chr = c("K10", "dage", "dgender"),
#                                                              prefd_covars_chr = c("K10", "dage", "dgender")) %>%
#                                     make_scndry_anlys_params(candidate_predrs_chr = c("AQOL6D"),
#                                                              candidate_covar_nms_chr = c("SOFAS", "dage", "dgender"),
#                                                              prefd_covars_chr = c("SOFAS", "dage", "dgender")))
```

This is another good time to save our work to date.

```r
dir.create(paste0(path_to_output_1L_chr,"/Project"))
saveRDS(projects_ls,paste0(path_to_output_1L_chr,"/Project/project_ls.RDS"))
```

5 Report and disseminate findings

5.1 Create shareable models

The model objects created and saved in our working directory by the preceding steps are not suitable for public dissemination. They are both too large in file size and, more importantly, include copies of our source dataset. We can overcome these limitations by creating shareable versions of the models. Two types of shareable version are created - copies of the original model objects in which fake data overwrites the original source data and tables of model coefficients.

```
projects_ls <- projects_ls %>% purrr::map(~ renewSlot(.x, "c_SpecificProject",
                                                     authorData(procureSlot(.x, "c_SpecificProject"),
                                                                consent_1L_chr = consent_1L_chr, depnt_var_min_val_1L_dbl = 0.0001))
                                          #author(.x, consent_1L_chr = consent_1L_chr, what_1L_chr = "models")
                                          # Needs update to TTU, before this code can be reformed.
                                        )
```

6 Specify study reporting metadata

We update our project modules so that they are ready to render and share reports.

```
projects_ls <- projects_ls %>% purrr::map(~ renew(.x, what_1L_chr = "reporting"))
```

We add metadata relevant to the reports that we will be generating. Note, when rendering real reports it may be necessary to update the default values in the CSVs for authors and their institutional affiliations.

```
projects_ls <- projects_ls %>%
  purrr::map2(1:2, ~{
    y_Ready4useRepos <- repositories_ls[ifelse(.x@b_SpecificParameters@fake_1L_lgl,3,1):ifelse(.x@b_SpecificParameters@fake_1L_lgl,4,2)][[
  renew(.x, {if(.x@b_SpecificParameters@fake_1L_lgl){ready4show::authors_tb}else{read.csv("../Data/CSVs/Authors.csv") %>%
                              tibble::as_tibble() %>% ready4show::ready4show_authors()}}, type_1L_chr = "authors",
        what_1L_chr = "reporting") %>%
  renew({if(.x@b_SpecificParameters@fake_1L_lgl){ready4show::institutes_tb }else{
                              read.csv("../Data/CSVs/Institutes.csv") %>% tibble::as_tibble() %>% ready4show::ready4show_institutes()}},
  renew(c(3L,3L), type_1L_chr = "digits", what_1L_chr = "reporting") %>%
  renew(c("PDF","PDF"), type_1L_chr = "formats", what_1L_chr = "reporting") %>%
  renew(paste0(ifelse(.x@b_SpecificParameters@fake_1L_lgl,
               "A hypothetical study using fake data to map",
               "Mapping"),
               " K10 psychological distress and SOFAS functioning measures to ",
               ifelse(inherits(.x@a_ScorzProfile,"ScorzAqol6Adol"),
                      "AQoL-6D",
                      "CHU-9D"),
               " health utility using data from a sample of young people presenting to primary mental health services"),
       type_1L_chr = "title", what_1L_chr = "reporting") %>%
  renew(y_Ready4useRepos, type_1L_chr = "repos", what_1L_chr = "reporting")
  })
```

6.1 Describe and share models

6.1.1 Author model catalogues

```
projects_ls <- projects_ls %>% purrr::map(~author(.x, consent_1L_chr = consent_1L_chr, download_tmpl_1L_lgl = T,
                                                  what_1L_chr = "catalogue"))
```

6.1.2 Author manuscripts

We add additional meta-data necessary for rendering a manuscript from template and then create the abstracts and plots that will be used in the automatically authored manuscripts.

```
projects_ls <- projects_ls %>%
  purrr::map2(list(c("AQoL-6D", "Adolescent AQoL Six Dimension"),
                   c("CHU-9D", "Child Health Utility Nine Dimension")),
              ~ {
                renew(.x, "Quality Adjusted Life Years (QALYs) are often used in economic evaluations, yet utility weights for deriving th
                      type_1L_chr = "background", what_1L_chr = "reporting") %>%
                  renew("None declared", type_1L_chr = "conflicts", what_1L_chr = "reporting") %>%
                  renew("Nothing should be concluded from this study as it is purely hypothetical.",
                        ifelse(.x@b_SpecificParameters@fake_1L_lgl, "Nothing should be concluded from this study as it is purely hypotheti
                               ifelse(inherits(.x@a_ScorzProfile,"ScorzAqol6Adol"),
                                      "REPLACE WITH TEXT TO BE WRITTEN ON REVIEW OF REAL AQOL-6D STUDY RESULTS",
                                      "REPLACE WITH TEXT TO BE WRITTEN ON REVIEW OF REAL CHU-9D STUDY RESULTS")),
                        type_1L_chr = "conclusion", what_1L_chr = "reporting") %>%
                  renew(ifelse(.x@b_SpecificParameters@fake_1L_lgl, "The study was reviewed and granted approval by no-one.",
                               "REPLACE WITH DETAILS OF ETHICS APPROVAL FOR REAL STUDY"),
                        type_1L_chr = "ethics", what_1L_chr = "reporting") %>%
                  renew(ifelse(.x@b_SpecificParameters@fake_1L_lgl, "The study was funded by no-one.",
                               "REPLACE WITH DETAILS OF FUNDING FOR REAL STUDY"), type_1L_chr = "funding", what_1L_chr = "reporting") %>%
                  renew(c("anxiety", "AQoL","CHU9D", "psychological distress", "QALYs", "utility mapping"), type_1L_chr = "keywords",
                        what_1L_chr = "reporting") %>%
                    renew(ifelse(.x@b_SpecificParameters@fake_1L_lgl, "The study sample is fake data.",
                                 "REPLACE WITH DETAILS OF SAMPLE FOR REAL STUDY"), type_1L_chr = "sample", what_1L_chr = "reporting")  %>%
                    renew(.y, type_1L_chr = "naming", what_1L_chr = "reporting") %>%
                    renew("use_renew_mthd", type_1L_chr = "abstract", what_1L_chr = "reporting") %>%
                    author(consent_1L_chr = consent_1L_chr, items_as_domains_1L_lgl == (.y[1]=="CHU-9D"), what_1L_chr = "plots")
                })
```

We download a program for automated authoring of a scientific summary from a template and run it to author a first draft of the manuscript.

```
projects_ls <- projects_ls %>% purrr::map(~author(.x, consent_1L_chr = consent_1L_chr, download_tmpl_1L_lgl = T,
                                                   what_1L_chr = "manuscript"))
```

We can copy the RMarkdown files that created the template manuscript to a new directory (called "Manuscript_Submission") so that we can then manually edit those files to produce a manuscript that we can submit for publication.

```
projects_ls <- projects_ls %>% purrr::map(~author(.x, consent_1L_chr = consent_1L_chr, type_1L_chr = "copy",
                                                   what_1L_chr = "manuscript"))
```

Alternatively we could download a program for rendering the submission version of the manuscript. To do so you need to uncomment the below code, update the references to the program you wish to download and then run it.

```
# Not run
# projects_ls <- projects_ls %>% purrr::map(~renew(.x, c("URL of GitHub repository with", "Program version number"),
# type_1L_chr = "template-manuscript", what_1L_chr = "reporting"))
```

We can now configure the output to be generated by the manuscript authoring program. The below commands will specify a Microsoft Word format manuscript and a PDF technical appendix. Unlike the template manuscript, the figures and tables will be positioned after (and not within) the main body of the manuscript. Note that the Word version of the manuscript generated by these values will require some minor formatting edits (principally to the display of tables and numbering of sections).

```
projects_ls <- projects_ls %>% purrr::map(~renew(.x, F, type_1L_chr = "figures-body", what_1L_chr = "reporting") %>%
                                           renew(F, type_1L_chr = "tables-body", what_1L_chr = "reporting") %>%
                                           renew(c("Word","PDF"), type_1L_chr = "formats", what_1L_chr = "reporting"))
```

If working with real data, the following commands will generate additional artefacts (high quality plots, summary table of dependency libraries) required for a manuscript submission.

```
projects_ls <- projects_ls %>% purrr::map(~{
  if(.x@b_SpecificParameters@fake_1L_lgl){
    .x
  }else{
    author(.x, consent_1L_chr = consent_1L_chr, type_1L_chr = "plots", what_1L_chr = "manuscript") %>%
                                                    author(consent_1L_chr = consent_1L_chr, type_1L_chr = "dependencies",
                                                           what_1L_chr = "manuscript")
  }
} )
```

Once any edits to the RMarkdown files for creating the submission manuscript have been finalised, we can run the following command to author the manuscript. If we are using a custom manuscript authoring program downloaded from an online repository the download_tmpl_1L_lgl argument will need to be set to `T`.

```
projects_ls <- projects_ls %>% purrr::map(~author(.x, consent_1L_chr = consent_1L_chr, download_tmpl_1L_lgl = F,
                                                  type_1L_chr="submission", what_1L_chr = "manuscript"))
```

We next generate the Supplementary Information for the submission manuscript.

```
projects_ls <- projects_ls %>% purrr::map(~author(.x, consent_1L_chr = consent_1L_chr,
                                                  supplement_fl_nm_1L_chr = "TA_PDF", type_1L_chr="submission",
                                                  what_1L_chr = "supplement"))
```

6.1.3 Share outputs

We can now share non-confidential elements (ie no copies of individual records) of the outputs that we have created via our study online repository. To run this step you will need write permissions to the online repository. In the below step we are sharing model catalogues, details of the utility instrument, the shareable mapping models (designed to be used in conjunction with the youthu package) and (only if we are using fake data) our manuscript files, including supplementary information. This step has to be run interactively as it will ask for permission (often twice for each request) both to upload each file to the relevant online repository and to publish the dataset.

```
projects_ls <- projects_ls %>% purrr::map(~share(.x, types_chr = c("auto", "submission"),
                                                 what_chr = c("catalogue", "instrument", "models",
                                                              {if(.x@b_SpecificParameters@fake_1L_lgl){
                                                                c("manuscript","supplement")
                                                                }else{character(0)}})))
```

7 Tidy workspace

The preceding steps saved many objects (mostly R model objects) that have embedded within them copies of the source dataset. We can now purge such copies from our output data directory.

Purging files should only be undertaken once you are happy with the reports you have generated as many of the files to be purged are required for report authoring. Once you are sure that you wish to purge the files, uncomment and run the following.

```r
#projects_ls <- projects_ls %>% purrr::map(~author(.x, what_1L_chr = "purge"))
```

Note, the small number of project copies we explicitly made in this script will not be purged by the preceding step, so these need to be deleted manually. They are located at:

- Data > AQOL > Fake / Real (Depending on whether data is fake or real) > Output > TTUProject.RDS
- Data > CHU > Fake / Real (Depending on whether data is fake or real) > Output > TTUProject.RDS
- Data > Project > project_ls
- Data > Project > project.RDS