Data wrangling for 'Comparison of wild mushroom use by ethnic groups surrounding the Upper Ouémé Reserve Forest in Benin, West Africa.'

anonymous for submission

Setup

```
library(magrittr) # for %>%,  %T>%, %<>%
library(rlang) # for %|%, %||%
library(knitr)
library(ggplot2)
```

1 Load data files

The interview file is a compilation of all data collected in interviews. Interviewee names have been removed for the publicly available version.

```
data.dir <- here::here("data")
interview.file <- file.path(
  data.dir,
  "interviews_anon.xlsx"
)

biodata <- readxl::read_excel(interview.file, sheet = "Biodata") %>%
  magrittr::set_names(make.names(names(.)))
interviews <- readxl::read_excel(interview.file, sheet = "Interview") %>%
  magrittr::set_names(make.names(names(.)))
focusgroups <- readxl::read_excel(interview.file, sheet = "Focus group") %>%
  magrittr::set_names(make.names(names(.)))
specimen_id <- readxl::read_excel(
  interview.file,
  sheet = "Specimen ID",
  col_types = "text") %>%
  magrittr::set_names(make.names(names(.)))
```

The species key handles synonymy and defines the formal citation and the abbreviation for each species.

```
species.file <- file.path(data.dir, "species.csv")
specieskey <- readr::read_csv(species.file, col_types = "cccc")
```

The symbology files give abbreviations and aesthetics to use when plotting the different ethnic groups and villages.

```
groups.file <- here::here("data", "groups.csv")
groups <- readr::read_csv(groups.file, col_types = "cccc")
villages.file <- here::here("data", "villages.csv")
villages <- readr::read_csv(villages.file, col_types = "ccccii")
villagegroups.file <- here::here("data", "villagegroups.csv")
villagegroups <-
  readr::read_csv(villagegroups.file, col_types = "cc") %>%
  dplyr::left_join(groups, by = c("Group")) %>%
  dplyr::left_join(villages, by = c("Village"), suffix = c("Group", "Village")) %>%
  dplyr::mutate(VillageGroup = paste0(Village, Group))
```

The installation history file gives the length of time each group has been present in each village.

```
installation_file <- file.path(data.dir, "installation.csv")
installation <- readr::read_csv(installation_file, col_types = "cci")
```

```
specimens <-
    readxl::read_xls(
      here::here("data", "specimens.xls"),
      col_types = c(
        "text",
        "date",
        rep("text", 5),
        rep("logical", 3),
        rep("numeric", 10)
        )
    )
```

Load the sequences.

```
seqs <-
  here::here("data", "fullseqs.fasta") %>%
  seqinr::read.fasta()
seqs <- tibble::tibble(
  specimen_ID = seqinr::getName(seqs),
  seq = unlist(seqinr::getSequence(seqs, as.string = TRUE)),
  Species.photo = stringr::str_extract(specimen_ID, "\\d+")
)
```

2 Data checks

2.1 Species list

List "extra" names from the species key. There should not be any.

```
dplyr::anti_join(specieskey, interviews, by = c("original" = "Scientific.name")) %>%
  dplyr::anti_join(focusgroups, by = c("original" = "Scientific.name"))
```

```
## # A tibble: 0 x 4
## # ... with 4 variables: original <chr>, canonicalName <chr>, Abbrev <chr>,
## #   scientificName <chr>
```

2.2 Focus group data

Ensure all species present in the focus group data are also present in the species key.

```
focusgroups %>%
  dplyr::anti_join(specieskey, by = c("Scientific.name" = "original")) %>%
  dplyr::pull(Scientific.name) %>%
  unique() %>%
  sort() %T>%
  {stopifnot(length(.) == 0)}
```

```
## character(0)
```

Replace the names in the focus groups data with the canonical names from the species list.

```
focusgroups %<>%
  dplyr::left_join(specieskey, by = c("Scientific.name" = "original")) %>%
  dplyr::select(-Scientific.name)
```

2.2.1 Village

Unique values for `Village`:

```
unique(focusgroups$Village)
```

```
## [1] "Angaradebou" "Bio sika"    "Gando"       "Sonnoumon"
```

These are consistent, but Bio *S*ika should be capitalized. Make them factors, using abbreviations.

```
focusgroups$Village %<>%
  stringr::str_to_title() %>%
  factor(levels = villages$Village.Long, labels = villages$Village)
```

2.2.2 Ethnic group

Unique values for `Ethnic.group`:

```
unique(focusgroups$Ethnic.group)
```

```
## [1] "Yom"    "Lokpa"  "Gando"  "Bariba"
```

These are consistent. Make them factors, using abbreviations.

```
focusgroups$Ethnic.group %<>%
  factor(levels = groups$Group.Long, labels = groups$Group)
```

2.3 Interview data

Ensure all species present in the interviews are also present in the species key.

```
interviews %>%
  dplyr::select(Scientific.name) %>%
  unique() %>%
  dplyr::filter(!Scientific.name %in% specieskey$original) %>%
  {stopifnot(nrow(.) == 0)}
```

Add canonical names for each species.

```
interviews %<>%
  dplyr::left_join(specieskey, by = c("Scientific.name" = "original")) %>%
  dplyr::select(-Scientific.name)
```

2.3.1 Recognize

What are the values for `Recognize`?

```
unique(interviews$Recognize)
```

```
## [1] "y" "n" NA
```

Assume that missing data (`NA`) means that they do not recognize it.

```
interviews %<>%
  dplyr::mutate(Recognize = (Recognize == "y") %|% FALSE)
```

2.3.2 Preference

Unique values for `Preference`:

```
unique(interviews$Preference)
```

```
## [1] "4"   "3"   "2"   "1"   "n/a" "0"   "?"   NA
```

Force them to be integers; this will map `"n/a"` and `"?"` to `NA`.

```
interviews %<>% dplyr::mutate(Preference = as.integer(Preference))
```

```
## Warning in mask$eval_all_mutate(dots[[i]]): NAs introduced by coercion
```

Now the values are:

```
unique(interviews$Preference)
```

```
## [1]  4  3  2  1 NA  0
```

2.4 Biographical data

2.4.1 Village

Unique values for `Village`:

```
unique(biodata$Village)
```

```
## [1] "Gando"       "Sonnoumon"   "Faba"        "Bio Sika"    "Angaradebou"
```

```
unique(installation$Village)
```

```
## [1] "Angaradebou" "Bio Sika"    "Faba"        "Gando"       "Sonoummon"
```

These are consistent. Make them factors, using abbreviations.

```
biodata$Village %<>%
  factor(levels = villages$Village.Long, labels = villages$Village)
installation$Village %<>%
  factor(levels = villages$Village.Long, labels = villages$Village)
```

2.4.2 Ethnic group

Unique values for `Ethnic.group`:

```
unique(biodata$Ethnic.group)
```

```
## [1] "Gando"                 "Bariba"                "Nagot"
## [4] "Peuhl/Fulfulde/Fulani" "Lokpa"                 "Yom"
```

```
unique(installation$Ethnic.group)
```

```
## [1] "Yom"    "Lokpa"  "Bariba" "Gando"
```

These are consistent. Make them factors, using abbreviations.

```
biodata$Ethnic.group %<>%
  factor(levels = groups$Group.Long, labels = groups$Group)
installation$Ethnic.group %<>%
  factor(levels = groups$Group.Long, labels = groups$Group)
```

Now join the installation history data to the main data.

```
biodata %<>%
  dplyr::left_join(installation, by = c("Village", "Ethnic.group"))
```

Add a factor which uniquely identifies both Village and Ethnic group for each interviewee.

```
biodata %<>%
  dplyr::mutate(
    VillageGroup = paste0(Village, Ethnic.group),
    VillageGroup = factor(VillageGroup,
                          levels = villagegroups$VillageGroup)
  )
```

2.4.3 Age

Unique values for `Age`

```
sort(table(biodata$Age))
```

```
##
##  17  18  22  24  25  27  32  33  37  38  39  43  52  55  59 >50  50  60 >70  30
##   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   2   2   3   4   4
##  45  35  40 >60
##   4   8   8  20
```

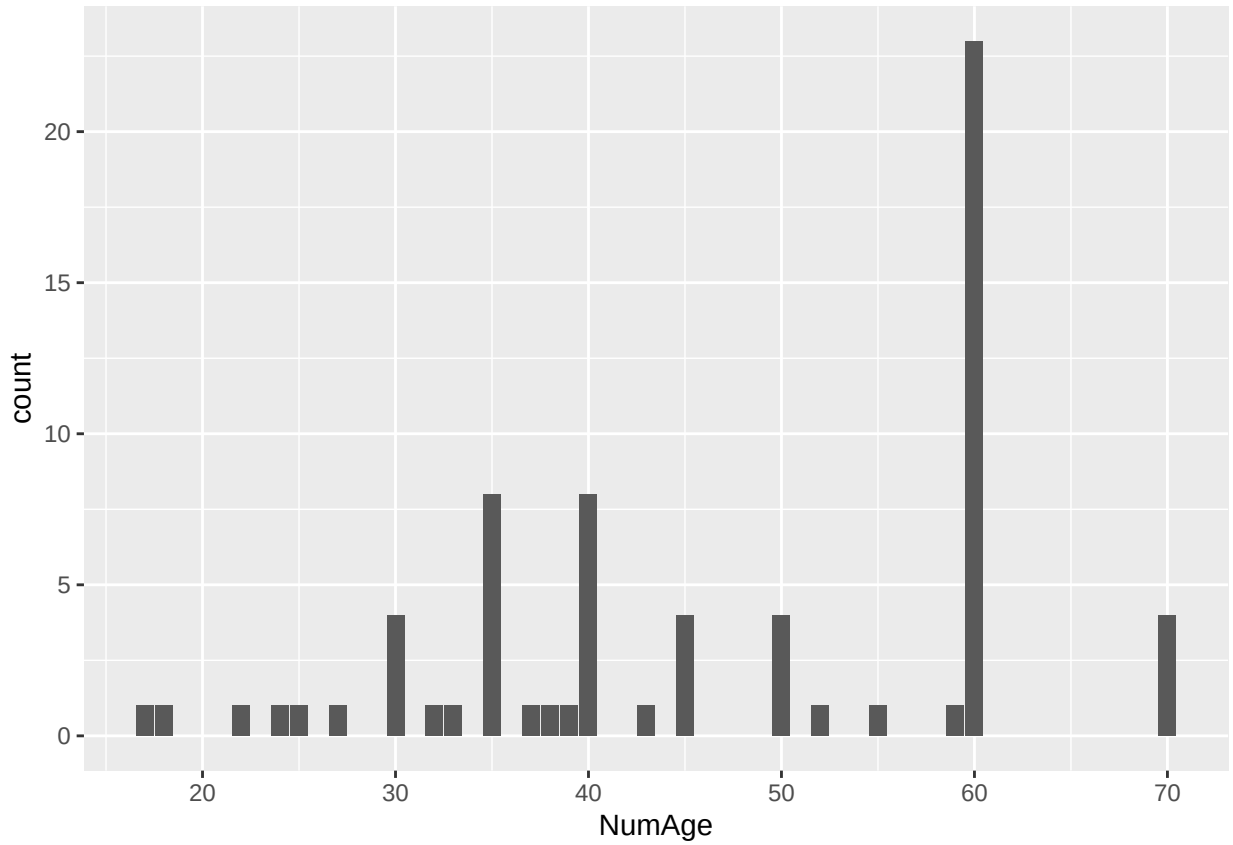
There are values of "> 50", "> 60", and "> 70"; in particular "> 60" is by far the most common value. All other values which are represented more than once are multiples of 5. This suggests that respondents frequently round their ages.

Previous works have divided respondants into age categories "young" <18, "adult" 18-35, and "old" 35+. We have only a very small number of "young" respondents, so we will simply split into <35 and 35+.

```
biodata %<>%
  dplyr::mutate(
    NumAge = readr::parse_number(Age),
    Age.group = cut(
      NumAge,
      c(0, 35, Inf),
      c("<35", "35+"),
      right = FALSE,
      ordered_result = TRUE
    )
  )
```

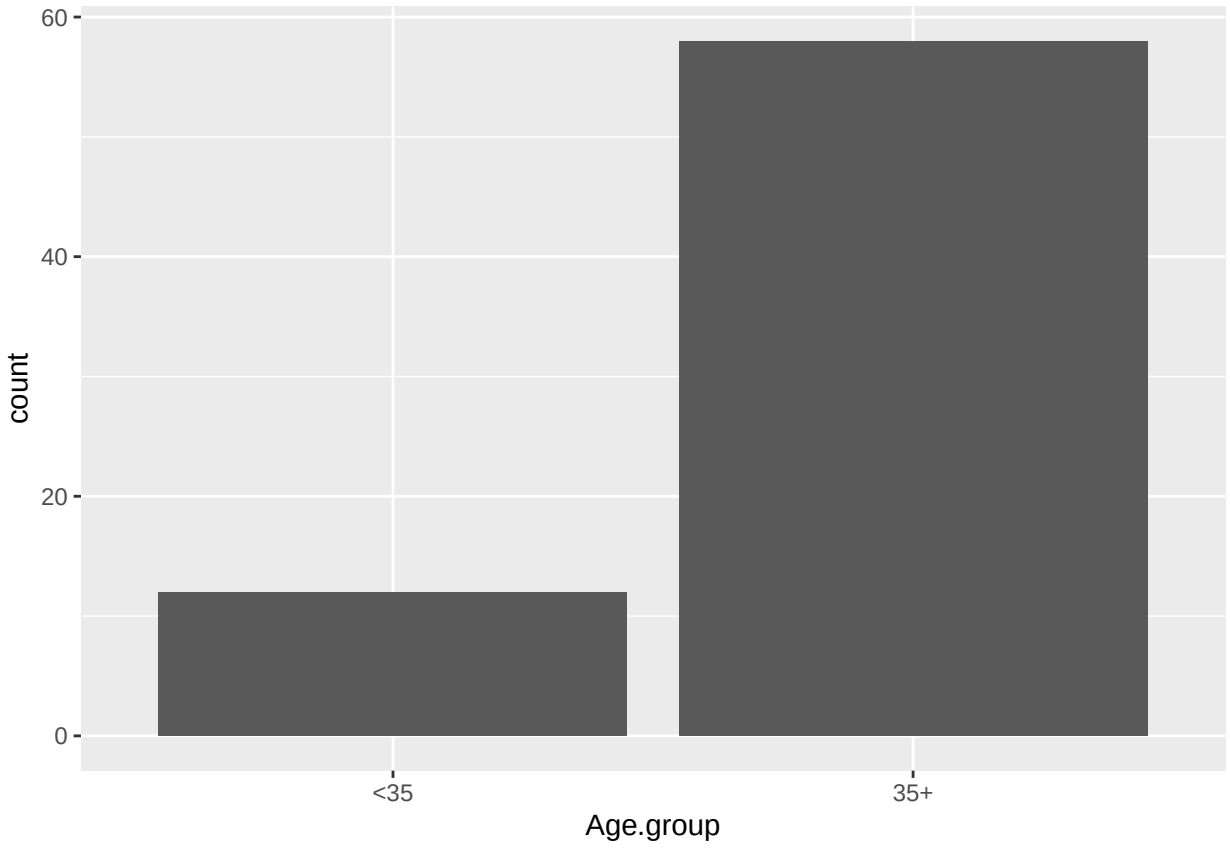
Here is a plot of the age distribution before binning: (The ">" values are plotted at the age given, even though it is a minimum.)

```
ggplot(biodata, aes(x = NumAge)) +
  geom_bar()
```



And after binning:

```
ggplot(biodata, aes(Age.group)) +
  geom_bar()
```



2.4.4 Gender

Values of `Sex`:

```
unique(biodata$Sex)
```

```
## [1] "F" "M"
```

No problems here, make it a factor.

```
biodata$Sex %<>% factor()
```

Let's call it `Gender` instead of `Sex`.

```
biodata %<>% dplyr::rename(Gender = Sex)
```

2.5 Vernacular names

Compile all the vernacular names used in the interviews, and count how many times each was given.

```
nametable <-
  # add a column to indicate whether the name came from an interview
  # or focus group
```

```
  list(
    interviews %>% dplyr::mutate(setting = "interview"),
    focusgroups %>% dplyr::mutate(setting = "focus group")
  ) %>%
  # select relevant columns and join the tables
  purrr::map_dfr(
    dplyr::select,
    canonicalName, scientificName, Abbrev, Vernacular.name,
    Language, Meaning.of.the.name, setting
  ) %>%
  # force to lower-case
  dplyr::mutate_at("Vernacular.name", tolower) %>%
  # n is the number of times the name was given in interviews
  # asterisks (star) indicate that the name was given in focus groups
  dplyr::group_by_at(dplyr::vars(!setting)) %>%
  dplyr::summarize(
    n = sum(setting == "interview"),
    star = strrep("*", sum(setting == "focus group"))
  ) %>%
  dplyr::ungroup() %>%
  # sort
  dplyr::arrange(Vernacular.name, Language, Abbrev) %>%
  # a missing name is not a name
  dplyr::filter(
    !is.na(Vernacular.name),
    Vernacular.name != "n/a",
    Vernacular.name != "no name"
  )
```

Write the name table to a file; this file was used as a base for further manual name curation.

```
name_table_file <- file.path(data.dir, "name_table.csv")
if (!file.exists(name_table_file))
  readr::write_csv(nametable, name_table_file)
```

Read the file with manual name curation.

```
fixed_name_table_file <- file.path(data.dir, "name_table_Boni-SB.xls")
fixed_name_table <- readxl::read_xls(fixed_name_table_file) %>%
  dplyr::mutate_at("n", as.integer) %>%
  dplyr::mutate_at("star", tidyr::replace_na, "")
```

Make sure that both files cover all the same rows. These are written to files so that an external diff tool can be used to address any mismatches.

```
match_cols <- c("canonicalName", "scientificName", "Abbrev",
                "Vernacular.name", "Language", "Meaning.of.the.name",
                "n", "star")
nametable_test <-
  nametable %>%
  dplyr::arrange(Language, Vernacular.name, Abbrev,
                 Meaning.of.the.name, n, star) %>%
```

```
  dplyr::select(dplyr::all_of(match_cols)) %T>%
  readr::write_csv(file.path(data.dir, "oldnametable.csv"))
fixed_name_table_test <-
  fixed_name_table %>%
  dplyr::arrange(Language, Vernacular.name, Abbrev,
                 Meaning.of.the.name, n, star) %>%
  dplyr::select(dplyr::all_of(match_cols)) %T>%
  readr::write_csv(file.path(data.dir, "newnametable.csv"))

stopifnot(isTRUE(all.equal(nametable_test, fixed_name_table_test)))
```

Create a table using just the curated names.

```
fixed_name_table <-
  dplyr::select(
    fixed_name_table,
    "canonicalName", "scientificName", "Abbrev",
    Alternate.name = "Vernacular.name",
    Vernacular.name = "New.Vernacular.name",
    "Language",
    Meaning.of.the.name = "New.Meaning.of.the.name",
    "n", "star", "explanation":"technique") %>%
  # some rows contain two different names
  tidyr::separate_rows(Vernacular.name, Meaning.of.the.name, sep = ";") %>%
  # remove white space
  dplyr::mutate_at(c("Vernacular.name", "Meaning.of.the.name"), trimws) %>%
  {dplyr::left_join(
    dplyr::select(., -"Alternate.name") %>%
      dplyr::group_by(dplyr::across(c(-n, -star))) %>%
      dplyr::summarize(n = sum(n), star = paste0(star, collapse = "")),
    dplyr::select(., Language, Vernacular.name, Alternate.name) %>%
      dplyr::filter(Vernacular.name != Alternate.name) %>%
      dplyr::group_by(Language, Vernacular.name) %>%
      dplyr::summarize_all(paste, collapse = ", "),
    c("Language", "Vernacular.name")
  )} %>%
  dplyr::mutate(species = glue::glue("{canonicalName} ({n}{star})")) %>%
  dplyr::ungroup() %>%
  dplyr::group_by(Vernacular.name, Language, Meaning.of.the.name,
           dplyr::across(explanation:technique)) %>%
  dplyr::summarize(species = paste(species, collapse = "; "))
```

Output a file including names which still have multiple entries.

```
problem_name_table <-
  dplyr::group_by(fixed_name_table, Vernacular.name, Language) %>%
  dplyr::filter(dplyr::n() > 1) %>%
  dplyr::arrange(Language, Vernacular.name) %T>%
  openxlsx::write.xlsx(here::here("output", "problem_names.xlsx"))
```

```
## Note: zip::zip() is deprecated, please use zip::zipr() instead
```

Format the names table for output.

```
fixed_name_table %<>%
  tidyr::pivot_longer(
    cols = unknown:technique,
    names_to = "characteristics"
  ) %>%
  dplyr::filter(!is.na(value)) %>%
  dplyr::mutate(
    type = dplyr::case_when(
      characteristics %in% c("color", "shape", "size", "texture", "changes") ~ "physical",
      characteristics %in% c("habitat", "growth habit") ~ "ecological",
      characteristics %in% c("edibility", "technique") ~ "practical",
      characteristics == "unknown" ~ "unknown"
    )) %>%
  dplyr::select(-value) %>%
  dplyr::ungroup()
```

3 Combine and save data

3.1 Biographical and interview data

Include only the four Ethnic groups with enough data to be included in the statistical analysis.

```
row_data <- interviews %>%
  dplyr::select(ID, Recognize, Preference, Abbrev) %>%
  dplyr::left_join(
    biodata %>%
      dplyr::select(ID, Village, Ethnic.group, VillageGroup,
                    Age.group, Gender, Installation),
    by = "ID") %>%
  dplyr::filter(Ethnic.group %in% c("Gan", "Yom", "Lok", "Bar"))
```

3.1.1 Find data which is only in specimens.

Some species were only shown as specimens, and were not available on every day. These are removed from the analysis.

```
specimen.only <- interviews %>%
  dplyr::select(Specimen.photo, Abbrev) %>%
  dplyr::group_by(Abbrev) %>%
  dplyr::filter(all(Specimen.photo == "Specimen")) %>%
  dplyr::pull(Abbrev) %>%
  unique()

row_data %<>% dplyr::filter(!Abbrev %in% specimen.only)
```

3.1.2 Create data for analyses

Sum the number of species recognized by each respondent to create the data for the GLM analysis.

```
knowledge <- row_data %>%
  dplyr::group_by(ID, Village, Ethnic.group, Age.group, Gender, Installation) %>%
  dplyr::summarize(Recognize = sum(Recognize, na.rm = TRUE)) %>%
  dplyr::ungroup() %>%
  dplyr::mutate_at("Ethnic.group", forcats::fct_drop)
```

Make the community matrices for CCA analysis. `knowledge.matrix` has interviewees as rows, and species as columns, where the value is 1 if the interviewee recognized the species, or 0 if they did not. `preference.matrix` has the same row and column setup, but the value is the degree of preference the interviewee has for the species (or 0 if they do not use/know it).

In both cases, we must also remove any rows which are empty, and columns with only one non-zero entry. In the knowledge matrix, this means people which do not know any mushrooms, and mushrooms which only one person knows. In the preference matrix, this means people who do not use any mushrooms (with preference >= 1) and mushrooms which nobody considers useful (preference >= 1).

```
knowledge.matrix <-
  row_data %>%
  dplyr::filter(Recognize) %>%
  dplyr::group_by(Abbrev, ID) %>%
  dplyr::summarize_at("Recognize", max) %>%
  tidyr::pivot_wider(
    names_from = Abbrev,
    values_from = Recognize,
    values_fill = 0
  ) %>%
  dplyr::arrange(ID) %>%
  tibble::column_to_rownames("ID") %>%
  as.matrix() %>%
  magrittr::extract(rowSums(.) > 0, colSums(.) > 2)
```

```
preference.matrix <-
  row_data %>%
  dplyr::filter(Recognize, !is.na(Preference)) %>%
  dplyr::group_by(Abbrev, ID) %>%
  dplyr::summarize_at("Preference", mean) %>%
  tidyr::pivot_wider(
    names_from = Abbrev,
    values_from = Preference,
    values_fill = 0
  ) %>%
  dplyr::arrange(ID) %>%
  tibble::column_to_rownames("ID") %>%
  as.matrix() %>%
  magrittr::extract(
    rowSums(.) > 0,
    apply(., 2, function(col) sum(col > 0) > 2)
  )
```

The CCAs also require biographical data for the interviewees.

```
knowledge.biodata <- tibble::tibble(ID = row.names(knowledge.matrix)) %>%
  dplyr::left_join(biodata, by = "ID")
preference.biodata <- tibble::tibble(ID = row.names(preference.matrix)) %>%
  dplyr::left_join(biodata, by = "ID")
```

Write the data as CSVs.

```
if (!dir.exists(here::here("output"))) dir.create(here::here("output"))
readr::write_csv(knowledge, here::here("output", "knowledge_data.csv"))
write.table(knowledge.matrix, here::here("output", "knowledge_matrix.csv"))
write.table(preference.matrix, here::here("output", "preference_matrix.csv"))
readr::write_csv(knowledge.biodata, here::here("output", "knowledge_biodata.csv"))
readr::write_csv(preference.biodata, here::here("output", "preference_biodata.csv"))
readr::write_csv(fixed_name_table, here::here("output", "nametable.csv"))
```

And as RDS for easy loading in R.

```
saveRDS(interviews, here::here("output", "interviews.rds"))
saveRDS(focusgroups, here::here("output", "focusgroups.rds"))
saveRDS(knowledge, here::here("output", "knowledge.rds"))
saveRDS(knowledge.matrix, here::here("output", "knowledge_matrix.rds"))
saveRDS(preference.matrix, here::here("output", "preference_matrix.rds"))
saveRDS(knowledge.biodata, here::here("output", "knowledge_biodata.rds"))
saveRDS(preference.biodata, here::here("output", "preference_biodata.rds"))
saveRDS(fixed_name_table, here::here("output", "nametable.rds"))
```