Demographics: qualitative responses

Qualitative responses: subfields

For Q18, after asking respondents for their broad domain of study, we asked them this free-response question:
What is your primary field? (e.g. astrophysics, neuroscience)
One response is preferred, but if multiple are needed, please separate with commas.

I plan to "classify" these free responses into standardized categories. The taxonomy of academic disciplines that I'll be using is here:
https://digitalcommons.elsevier.com/en_US/dc-disciplines-taxonomy

I downloaded the PDF and did a very tiny amount of curation to make this list of academic fields and subfields machine-readable. (I left a txt file with my curation notes in the curation_of_disciplines/ folder in the Dryad data deposit.)

This script expects that your data folder contains a folder called curation_of_disciplines that in turn contains the lightly curated taxonomy of disciplines (digital_commons_disciplines.txt).

I'll use a fuzzy string matching algorithm to classify the responses–essentially, looking for which item in the taxonomy most closely matches the user's entry. Hopefully, there will be very few entries that aren't a good match for anything in the taxonomy, and I can deal with these stragglers manually.

Note that while I am preforming classification, there's no "real" machine learning here.

Note from the future: this code ended up being pretty ugly, with a lot of nested loops. Maybe I could have written less code if I'd used the `fuzzyjoin` package. Oh well. I went for a rudimentary solution, and it worked well enough.

Import packages and utilities

```r
project_root <- here::here() # requires that you be somewhere in the
# project directory (not above it)
# packages
suppressMessages(source(file.path(project_root, "scripts/packages.R")))
# functions and objects used across scripts
suppressMessages(source(file.path(project_root, "scripts/utils.R")))
```

Define funtions

get_all_hits

- Arguments:
  - `query`: A survey response, e.g. "neuroscience" or "AI".
  - `tax_df`: The taxonomy, as a data frame with columns `Level1`, `Level2`, and `Level3`. It wasn't really necessary to include this as an argument since it doesn't change, but I like to be explicit about what goes into the function.
- Details:
  - For a given query (survey response), return a data frame with the similarity scores between that query and all strings in the DC taxonomy. Each row in the taxonomy gets a row score, which is the minimum distance between the query and any of the three strings in that row. Row score was computed with `get_row_results()`.
- Outputs:
  - A data frame with columns `min_dist`, `level`, and `index`. `index` is just row number for that row in the taxonomy/in this data frame (they have the same number of rows). `level` indicates which level the winning string came from. `index` will be useful later. In the for loop at the end of this script, this df is called `q_scores`.

```r
get_all_hits <- function(query, tax_df) {
    n_tax <- nrow(tax_df)

  # Pre-allocate data frame rows for computational efficiency
  query_scores <- data.frame(
    min_dist = rep(NA_real_, n_tax),
    level = rep(NA_character_, n_tax),
    index = seq_len(n_tax),
    stringsAsFactors = FALSE
  )

  # Get metadata for all hits with the minimum distance score
  for (i in seq_len(n_tax)) {
    results <- get_row_results(query, tax_df, i)
    query_scores$min_dist[i] <- as.numeric(results[[1]])
    query_scores$level[i] <- results[[2]]
  }

  return(query_scores)
}
```

get_row_results

- Arguments:
  - `query`: A survey response, e.g. "neuroscience" or "AI".
  - `tax_df`: see get_all_hits
  - `row_num`: Row number of the row in `tax_df` you want to analyze.
- Details:
  - For a given query (survey response), and a given row in the taxonomy, record which column's entry had the lowest distance score. Breaks ties by just choosing the first occurrence of a minimum value. So it prefers Level1 over Level2, and Level2 over Level3. I'm betting that 99% of ties are basically just noise, and not real matches.
- Outputs:
  - A vector with two elements: first, the lowest distance score between the query and any string in that row of the taxonomy, and second, the name of the column that the winning string came from (e.g. `Level1`).

```
get_row_results <- function(query, tax_df, row_num) {

  strL1 <- tax_df[row_num, "Level1"]
  strL2 <- tax_df[row_num, "Level2"]
  strL3 <- tax_df[row_num, "Level3"]

  # compute Levenshtein distances for each level
  dist1 <- as.numeric(
    adist(query, strL1)
  )
```

```r
  dist2 <- as.numeric(
    ifelse(
      !is.na(strL2),
      adist(query, strL2),
      Inf
    )
  )
  dist3 <- as.numeric(
    ifelse(
      !is.na(strL3),
      adist(query, strL3),
      Inf
    )
  )

  dists <- c(dist1, dist2, dist3)
  min_dist <- min(dists, na.rm = TRUE)

  # which.min gets the index of the lowest element.
  # In the event of a tie, it chooses the first one.
  # So it will choose level1 over level2, and level2 over level3.
  level <- c("Level1", "Level2", "Level3")[
    which.min(dists)
  ]

  # This was a little test I ran to see how common ties are.
  # Turns out ties are extremely common. They are all, or nearly
  # all, cases where all 3 levels are really different from the query.
  # I'm willing to assume that a tie means the taxonomy strings are all
  # really different from the query, and I'm not going to fret over the
  # fact that this function is practically breaking the tie at random,
  # just choosing the first minimum in the list.
#   has_tied_min <- function(x, na.rm = FALSE) {
#   m <- min(x, na.rm = na.rm)
#   sum(x == m, na.rm = na.rm) > 1L
# }
  #if (has_tied_min(dists)) {message(query, " | ", tax_df[row_num,], " | ", dists)}

  return(
    c(min_dist, level)
  )
}
```

get_candidates

- Arguments:
  - `q_scores`: A data frame with columns `min_dist`, `level`, and `index`, produced by `get_all_hits()`.
- Details:
  - Given the minimum distance scores for every row/query pair, find the global minimum across all rows. Handles ties by reporting multiple candidates.
- Outputs:
  - A data frame with the same columns as the input data frame. It has been merely been subsetted so the only rows remaining are those with the minimum distance.

```r
get_candidates <- function(q_scores) {

  best_dist <- min(q_scores$min_dist)
  cands <- q_scores %>% filter(min_dist == best_dist)
  return(cands)
}
```

get_winner_df

- Arguments:
  - `cands_df`: A data frame with columns `min_dist`, `level`, and `index`, produced by `get_candidates()`.
  - `tax_df`: see `get_all_hits()`.
- Details:
  - Given a set of candidate rows, choose a smaller number of rows to keep. Chooses final picks by preferring Level 3 matches over Level 2 matches, and preferring Level 2 over Level 1. Returns a subset of the taxonomy, not just a bunch of indices and distance scores.
- Outputs:
  - A subset of the taxonomy; the final rows for manual inspection (as a data frame).

```r
get_winner_df <- function(cands_df, tax_df) {
  winners <- ""
  lvl3_cands <- subset(cands_df, level == "Level3")
  lvl2_cands <- subset(cands_df, level == "Level2")
  lvl1_cands <- subset(cands_df, level == "Level1")

  if (nrow(lvl3_cands) > 0) {
    winners <- tax_df[lvl3_cands$index, ]
  } else if (nrow(lvl2_cands) > 0) {
    keep_idx <- lvl2_cands$index[is.na(tax_df$Level3[lvl2_cands$index])]
    winners <- tax_df[keep_idx, , drop = FALSE]
  } else {
    keep_idx <- lvl1_cands$index[
      is.na(tax_df$Level2[lvl1_cands$index]) &
        is.na(tax_df$Level3[lvl1_cands$index])
    ]
    winners <- tax_df[keep_idx, , drop = FALSE]
  }
  return(winners)
}
```

Load data

```r
other_quant <- load_qualtrics_data("clean_data/other_quant.tsv")
status <- load_qualtrics_data("clean_data/contributor_status_Q3.tsv")
qual <- load_qualtrics_data("qual_responses.tsv")

# do not read with my load_qualtrics_data function bc it has header = TRUE
tax <- read.csv(
  file = file.path(DATA_PATH, "curation_of_disciplines/digital_commons_disciplines.txt"),
  header = FALSE,
  sep = "\t",
  check.names = FALSE,
  stringsAsFactors = FALSE
)

data <- cbind(status, other_quant)
nrow(data)
```

```
[1] 332
```

```r
head(data)
```

```
   Past Future           campus      favorite_solution field_of_study
1  True   True UC Santa Barbara  Sustainability grants    Math and CS
2  True   True UC Santa Barbara       Containerization  Life sciences
3  True   True UC Santa Barbara Computing environments     Humanities
4  True   True UC Santa Barbara  Sustainability grants    Math and CS
5  True   True UC Santa Barbara     Documentation help  Life sciences
6 False   True UC Santa Barbara                           Math and CS
          job_category staff_categories
1              Faculty
2             Post-Doc
3 Other research staff
4              Faculty
5              Faculty
6 Other research staff
```

```r
tmp <- data$job_category[nzchar(data$job_category)]
job_count <- data.frame(table(tmp))
names(job_count) <- c("Job", "Count")

academics <- sum(subset(job_count, Job != "Non-research Staff")[, "Count"])
```

```r
qual_fields <- qual$subfield[nzchar(qual$subfield)]
length(qual_fields)
```

```
[1] 174
```

```r
academics
```

```
[1] 188
```

This question was not mandatory, but 174/188 academics answered it.

```r
head(qual_fields)
```

```
[1] "AI and neuroscience"               "Plant Biology / Ecology"
[3] "Digital humanities, History"       "Computer science, neuroscience"
[5] "Evolutionary Genomics"             "Medical Imaging, Vision Science"
```

Let's standardize the capitalization in this vector. The capitalization will now match my taxonomy, which is important since the fuzzy string matching algorithm I'll be using is case-sensitive.

```r
qual_fields <- unname(
    sapply(
        qual_fields,
        function(x) tools::toTitleCase(tolower(x)) )
    )

head(qual_fields)
```

```
[1] "Ai and Neuroscience"                "Plant Biology / Ecology"
[3] "Digital Humanities, History"        "Computer Science, Neuroscience"
[5] "Evolutionary Genomics"              "Medical Imaging, Vision Science"
```

Hmm. The correction of "AI" to "Ai" is unfortunate, but let's see how it goes.

Here's our taxonomy:

```r
head(tax)
```

```
                                                               V1
1                                                    Architecture
2                          Architecture: Architectural Engineering
3               Architecture: Architectural History and Criticism
4                           Architecture: Architectural Technology
5                           Architecture: Construction Engineering
6 Architecture: Cultural Resource Management and Policy Analysis
```

Let's tidy this data frame.

```r
tax <- tax %>%
  separate(
    col   = names(tax)[1],
    into  = c("Level1", "Level2", "Level3"),
    sep   = ": ",
    fill  = "right",   # any missing pieces become NA
    extra = "merge"    # if there were >2 colons, they'd all merge into Level3
  )

head(tax)
```

```
        Level1                                           Level2 Level3
1 Architecture                                             <NA>   <NA>
2 Architecture                   Architectural Engineering   <NA>
3 Architecture          Architectural History and Criticism   <NA>
4 Architecture                    Architectural Technology   <NA>
5 Architecture                    Construction Engineering   <NA>
6 Architecture Cultural Resource Management and Policy Analysis   <NA>
```

You can take my word for it that I looked back and forth between the data frame and the data file, and made sure that the data frame looks good. (Correct # of rows, row numbers match line numbers, etc.)

Cool! Now we are ready for fuzzy string matching. Let's start with a very rudimentary method, and we can do something fancier if this appears insufficient.

We are just using the adist() function in base R to calculate the Levenshtein distance between pairs of strings–the minimum number of substitutions needed to turn 'string a' into 'string b'.

Since we invited people to separate multiple disciplines with a comma, we'll need to parse those and track participant IDs. Several people used a slash instead of a comma, so we'll use a regex that looks for that, too.

```
# I think the data need to be a tibble for unnest to work

responses_df <- tibble(
  participantID = seq_along(qual_fields),
  response = strsplit(qual_fields, "\\s*[,/]\\s*", perl = TRUE)
  # \\s* matches zero or more whitespace (first backslash is an escape)
  # [,/] will match either a comma or a slash
  # With base R strsplit(), \\s only works if you set perl = TRUE
) %>%
  unnest_longer(response) %>% # takes a column of type list, where the entries
  # are themselves vectors, and turns each element of those vectors
  # into its own row, duplicating the other fields appropriately.
  mutate(response = trimws(response)) %>%     # strip whitespace around each field
  filter(response != "")                     # drop empty entries (from e.g. A,,B)

responses_df
```

```
# A tibble: 192 x 2
   participantID response
           <int> <chr>
```

```
 1             1 Ai and Neuroscience
 2             2 Plant Biology
 3             2 Ecology
 4             3 Digital Humanities
 5             3 History
 6             4 Computer Science
 7             4 Neuroscience
 8             5 Evolutionary Genomics
 9             6 Medical Imaging
10             6 Vision Science
# i 182 more rows
```

Again, you can take my word for it that I visually compared this df to the raw survey data and it looks good. Now let's get to the meat of the script: for each discipline entered by a survey participant, get rows that contain the best possible fuzzy match (allowing multiple rows for ties).

This code takes a minute to run.

```
results <- vector("list", nrow(responses_df))

for (i in seq_len(nrow(responses_df))) {
  current_query <- responses_df$response[i]
  current_participantID <- responses_df$participantID[i]

  q_scores <- get_all_hits(current_query, tax)
  candidates <- get_candidates(q_scores)
  winnerdf <- get_winner_df(candidates, tax)

  # add participant info
  winnerdf$participantID <- current_participantID
  winnerdf$response <- current_query
  results[[i]] <- winnerdf
}

final_results <- dplyr::bind_rows(results) %>%
  # move participantID to be the first column
  dplyr::relocate(participantID, response)

head(final_results, n=50)
```

```
   participantID                    response                          Level1
```

1	1	Ai and Neuroscience	Life Sciences
2	2	Plant Biology	Life Sciences
3	2	Ecology	Life Sciences
4	2	Ecology	Medicine and Health Sciences
5	2	Ecology	Medicine and Health Sciences
6	2	Ecology	Physical Sciences and Mathematics
7	3	Digital Humanities	Arts and Humanities
8	3	History	Arts and Humanities
9	4	Computer Science	Physical Sciences and Mathematics
10	4	Neuroscience	Medicine and Health Sciences
11	5	Evolutionary Genomics	Education
12	5	Evolutionary Genomics	Life Sciences
13	6	Medical Imaging	Medicine and Health Sciences
14	6	Vision Science	Engineering
15	7	Physics	Physical Sciences and Mathematics
16	8	Linguistics	Social and Behavioral Sciences
17	9	Information Science	Physical Sciences and Mathematics
18	9	Information Science	Social and Behavioral Sciences
19	10	Geography	Social and Behavioral Sciences
20	11	Physics	Physical Sciences and Mathematics
21	12	Ai	Law
22	13	Marine Conservation	Social and Behavioral Sciences
23	13	Marine Conservation	Social and Behavioral Sciences
24	14	Neuroscience	Medicine and Health Sciences
25	15	Physical Oceanography	Medicine and Health Sciences
26	16	Environmental Data Science	Physical Sciences and Mathematics
27	17	Computer Science	Physical Sciences and Mathematics
28	18	Environmental Sciences	Physical Sciences and Mathematics
29	19	Mathematics	Physical Sciences and Mathematics
30	20	Ecology	Life Sciences
31	20	Ecology	Medicine and Health Sciences
32	20	Ecology	Medicine and Health Sciences
33	20	Ecology	Physical Sciences and Mathematics
34	21	Biomedical Physics	Physical Sciences and Mathematics
35	22	Political Science	Social and Behavioral Sciences
36	23	Health Policy	Social and Behavioral Sciences
37	24	Cfd Engineering	Engineering
38	25	High Energy Physics	Physical Sciences and Mathematics
39	26	Oral History	Arts and Humanities
40	27	Neuroscience	Medicine and Health Sciences
41	28	Psychology	Social and Behavioral Sciences
42	29	Computer Science	Physical Sciences and Mathematics
43	30	Computer Science	Physical Sciences and Mathematics

```
44            31                  Radiology      Medicine and Health Sciences
45            32             Urban Planning    Social and Behavioral Sciences
46            33                    Ecology                     Life Sciences
47            33                    Ecology      Medicine and Health Sciences
48            33                    Ecology      Medicine and Health Sciences
49            33                    Ecology Physical Sciences and Mathematics
50            34               Astrophysics               Arts and Humanities
                                                    Level2               Level3
1                            Neuroscience and Neurobiology Systems Neuroscience
2                                           Plant Sciences        Plant Biology
3                                          Animal Sciences              Zoology
4                                      Medical Specialties             Oncology
5                                      Medical Specialties              Urology
6                                           Earth Sciences              Geology
7                                       Digital Humanities                 <NA>
8                                                  History                 <NA>
9                                        Computer Sciences                 <NA>
10                                        Medical Sciences        Neurosciences
11                                     Education Economics                 <NA>
12                                   Genetics and Genomics                 <NA>
13                                        Medical Sciences      Medical Anatomy
14               Biomedical Engineering and Bioengineering       Vision Science
15                                                 Physics                 <NA>
16                                             Linguistics                 <NA>
17                                       Computer Sciences Information Security
18                         Library and Information Science Information Literacy
19                                               Geography                 <NA>
20                                                 Physics                 <NA>
21                                                    <NA>                 <NA>
22                                           Communication   Mass Communication
23 Public Affairs, Public Policy and Public Administration       Transportation
24                                        Medical Sciences        Neurosciences
25                              Rehabilitation and Therapy     Physical Therapy
26                                  Environmental Sciences                 <NA>
27                                       Computer Sciences                 <NA>
28                                  Environmental Sciences                 <NA>
29                                             Mathematics                 <NA>
30                                         Animal Sciences              Zoology
31                                     Medical Specialties             Oncology
32                                     Medical Specialties              Urology
33                                          Earth Sciences              Geology
34                                          Earth Sciences      Mineral Physics
35                                       Political Science                 <NA>
```

```
36 Public Affairs, Public Policy and Public Administration        Health Policy
37                                                      <NA>               <NA>
38                                            Earth Sciences    Mineral Physics
39                                                   History       Oral History
40                                          Medical Sciences      Neurosciences
41                                                Psychology               <NA>
42                                         Computer Sciences               <NA>
43                                         Computer Sciences               <NA>
44                                       Medical Specialties          Radiology
45 Public Affairs, Public Policy and Public Administration      Urban Studies
46                                           Animal Sciences            Zoology
47                                       Medical Specialties           Oncology
48                                       Medical Specialties            Urology
49                                            Earth Sciences            Geology
50                                                Philosophy        Metaphysics
```

Okay, well I got results, but there's a pretty high rate of crappy matches. And then there's some that only a sophisticated algorithm would catch, and a couple that I'M not even sure how to classify. I think experimenting with better algorithms, or "real ML" classifiers, would probably take longer than just manually sifting through these. My code is ugly, and I hate that my procedure is not reproducible, but it's fast. So I'm just going to manually review these classifications in Microsoft Excel.

```
write_df_to_file(final_results, "curation_of_disciplines/qual_fields_guesses.tsv")
```

```
sessionInfo()
```

```
R version 4.4.2 (2024-10-31)
Platform: aarch64-apple-darwin20
Running under: macOS 26.1

Matrix products: default
BLAS:   /Library/Frameworks/R.framework/Versions/4.4-arm64/Resources/lib/libRblas.0.dylib
LAPACK: /Library/Frameworks/R.framework/Versions/4.4-arm64/Resources/lib/libRlapack.dylib;

locale:
[1] C.UTF-8/C.UTF-8/C.UTF-8/C/C.UTF-8/C.UTF-8

time zone: America/Los_Angeles
tzcode source: internal

attached base packages:
```

```
[1] tools     grid      stats     graphics  grDevices datasets  utils
[8] methods   base

other attached packages:
 [1] treemapify_2.5.6      tidyr_1.3.1           svglite_2.2.1
 [4] stringr_1.5.1         scales_1.4.0          readr_2.1.5
 [7] pwr_1.3-0             patchwork_1.3.2       ordinal_2023.12-4.1
[10] lme4_1.1-37           Matrix_1.7-1          languageserver_0.3.16
[13] here_1.0.1            gtools_3.9.5          ggforce_0.5.0
[16] FSA_0.10.0            fpc_2.2-13            forcats_1.0.0
[19] factoextra_1.0.7      ggplot2_3.5.2         emmeans_1.11.2
[22] dplyr_1.1.4           corrplot_0.95         ComplexHeatmap_2.22.0
[25] cluster_2.1.8.1       BiocManager_1.30.26

loaded via a namespace (and not attached):
 [1] Rdpack_2.6.4        rlang_1.1.6         magrittr_2.0.3
 [4] clue_0.3-66         GetoptLong_1.0.5    matrixStats_1.5.0
 [7] compiler_4.4.2      flexmix_2.3-20      systemfonts_1.2.3
[10] png_0.1-8           callr_3.7.6         vctrs_0.6.5
[13] pkgconfig_2.0.3     shape_1.4.6.1       crayon_1.5.3
[16] fastmap_1.2.0       utf8_1.2.6          rmarkdown_2.29
[19] ggfittext_0.10.2    tzdb_0.5.0          ps_1.9.1
[22] nloptr_2.2.1        purrr_1.1.0         xfun_0.53
[25] modeltools_0.2-24   jsonlite_2.0.0      tweenr_2.0.3
[28] parallel_4.4.2      prabclus_2.3-4      R6_2.6.1
[31] stringi_1.8.7       RColorBrewer_1.1-3  boot_1.3-31
[34] diptest_0.77-2      numDeriv_2016.8-1.1 estimability_1.5.1
[37] Rcpp_1.1.0          iterators_1.0.14    knitr_1.50
[40] IRanges_2.40.1      splines_4.4.2       nnet_7.3-19
[43] tidyselect_1.2.1    yaml_2.3.10         doParallel_1.0.17
[46] codetools_0.2-20    processx_3.8.6      lattice_0.22-6
[49] tibble_3.3.0        withr_3.0.2         evaluate_1.0.4
[52] polyclip_1.10-7     xml2_1.4.0          circlize_0.4.16
[55] mclust_6.1.1        kernlab_0.9-33      pillar_1.11.0
[58] renv_1.1.5          foreach_1.5.2       stats4_4.4.2
[61] reformulas_0.4.1    generics_0.1.4      rprojroot_2.1.1
[64] S4Vectors_0.44.0    hms_1.1.3           minqa_1.2.8
[67] xtable_1.8-4        class_7.3-22        glue_1.8.0
[70] robustbase_0.99-4-1 mvtnorm_1.3-3       rbibutils_2.3
[73] colorspace_2.1-1    nlme_3.1-166        cli_3.6.5
[76] textshaping_1.0.1   gtable_0.3.6        DEoptimR_1.1-4
[79] digest_0.6.37       BiocGenerics_0.52.0 ucminf_1.2.2
[82] ggrepel_0.9.6       rjson_0.2.23        farver_2.1.2
```

```
[85] htmltools_0.5.8.1   lifecycle_1.0.4     GlobalOptions_0.1.2
[88] MASS_7.3-61
```