

Data cleanup, part 2

In this script, I create some cleaner data frames for the more complex questions in our quantitative data set, to minimize the data wrangling at the top of each analysis script.

Input:

deidentified_no_qual.tsv (produced by data_cleanup.R)
pii.tsv (produced by data_cleanup.R)

Output:

clean_data/
 challenges_Q9.tsv
 contributor_roles_Q4.tsv
 contributor_status_Q3.tsv
 future_contributors_Q15.tsv
 hosting_services_Q8.tsv
 importance_Q2.tsv
 motivations_Q6.tsv
 project_size_Q5.tsv
 project_types_Q7.tsv
 solutions_Q10.tsv
 other_quant.tsv

There are certain patterns in the way Qualtrics displays questions, so I've organized this into groups of similarly formatted questions.

Load packages

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

```
# functions and objects used across scripts
suppressMessages(source(file.path(project_root, "scripts/utils.R")))
```

Functions

strip_descriptions

- Arguments:
 - df: A data frame. Presumably, all entries are strings, and at least one column has entries that contain a colon.
- Details:
 - For each entry in a data frame, strips all text following the colon, if there is one.
- Outputs:
 - A new data frame with shortened entries.
- Example:

```
t <- data.frame(
  col1 = c("A:1", "A:1", "", NA, "A:1"),
  col2 = c("B:2", "", "B:2", NA, NA)
)
> t
  col1 col2
1  A:1  B:2
2  A:1
3      B:2
4 <NA> <NA>
5  A:1 <NA>
> strip_descriptions(t)
  col1 col2
[1,] "A"  "B"
[2,] "A"  NA
[3,] NA   "B"
[4,] NA   NA
[5,] "A"  NA
```

```
strip_descriptions <- function(df) {
  new_df <- apply(df, MARGIN = c(1, 2), FUN = function(x) {
    strsplit(x, ":")[[1]][1]
  })
  return(new_df)
}
```

rename_cols_based_on_entries

- Arguments:
 - df: A data frame. Presumably, each column contains only one meaningful value.
- Details:
 - Rename columns based on the meaningful value in that column. If a column is entirely NA or empty strings, function returns “?”.
- Outputs:
 - A new data frame with meaningful columns.
- Example:

```
t <- data.frame(
  col1 = c("A", "A", "", NA, "A"),
  col2 = c("B", "", "B", NA, NA)
)
> rename_cols_based_on_entries(t)
   A    B
1  A    B
2  A
3      B
4 <NA> <NA>
5  A <NA>
> get_unique_vals(t, 1)
[1] "A"
```

```
rename_cols_based_on_entries <- function(df) {
  colnames(df) <- sapply(seq_len(ncol(df)), function(x) {
    get_unique_vals(df, x)
  })
  as.data.frame(df)
}
```

```
# Propose a column name based on the entries in that column.
get_unique_vals <- function(df, col_num) {
  unique_vals <- unique(df[, col_num])
  if (length(unique_vals) == 1 && (is.na(unique_vals) | unique_vals == "")) {
    return("?")
  }
  unique_vals <- unique_vals[!(is.na(unique_vals) | unique_vals == "")]
  stopifnot(length(unique_vals) == 1)
  return(unique_vals)
}
```

make_df_binary

- Arguments:
 - df: A data frame containing strings and/or NAs.
 - cols: Optional. A character vector of column names, or numeric indices of columns you want to modify.
- Details:
 - Takes a data frame where some entries are meaningful strings and others are empty strings or NAs, and converts the meaningful strings to 1s and the empty strings and NAs to 0s. Importantly, the string “Non-applicable” is not considered a “meaningful string”; these entries are converted to 0s.
- Outputs:
 - A new data frame where all entries are 0s or 1s.
- Example:

	Job	Improve	Tools	Customize	Network	Give back	Skills	Fun	Other
1									
2									
3		Improve	Tools	Customize		Give back	Skills	Fun	Other
4									
5									
6	Job	Improve	Tools	Customize	Network	Give back	Skills	Fun	Other
7	Job	Improve	Tools			Give back	Skills		
8						Give back	Skills	Fun	Other
9									
10	Job	Improve	Tools			Give back	Skills		

Becomes:

	Job	Improve	Tools	Customize	Network	Give back	Skills	Fun	Other
1	0		0	0	0	0	0	0	0
2	0		0	0	0	0	0	0	0
3	0		1	1	0	1	1	1	1
4	0		0	0	0	0	0	0	0
5	0		0	0	0	0	0	0	0
6	1		1	1	1	1	1	1	1
7	1		1	0	0	1	1	0	0
8	0		0	0	0	1	1	1	1
9	0		0	0	0	0	0	0	0
10	1		1	0	0	1	1	0	0

```
make_df_binary <- function(df, cols = NULL) {  
  # Determine columns to modify  
  if (is.null(cols)) {  
    cols_to_modify <- names(df)  
  } else if (is.numeric(cols)) {  
    cols_to_modify <- names(df)[cols]  
  } else if (is.character(cols)) {  
    cols_to_modify <- cols  
  } else {  
    stop(  
      "`cols` must be NULL, a character vector of column names, or numeric indices."  
    )  
  }  
  
  df <- df %>%  
    # Convert "Non-applicable" to NA  
    mutate(across(  
      all_of(cols_to_modify),  
      ~ ifelse(.x == "Non-applicable", NA, .x)  
    )) %>%  
    # Turn empty strings into NAs, and turn non-empty strings into 1s  
    mutate(across(all_of(cols_to_modify), ~ ifelse(.x == "", NA, 1))) %>%  
    # Convert all NAs to 0s  
    mutate(across(all_of(cols_to_modify), ~ ifelse(is.na(.x), 0, .x)))  
  
  return(df)  
}
```

shorten_long_responses

- Arguments:
 - `df`: A data frame with at least one column containing (presumably long) strings. Any column with strings to be replaced must be a character column, not a factor.
 - `codes`: A list where keys are the beginning of the long string to be replaced, and values are the short string to replace it with.
- Details:
 - Takes a data frame where some entries are long strings, and replaces these with pre-specified shorter strings. You don't need to specify the entire long string to be replaced; just specify the first part of it, and the function will identify matches.
- Outputs:
 - A new data frame where your pre-specified string substitutions have been made.
- Example:

```
hosting_services_18 hosting_services_19 hosting_services_20
A custom website (e.g. a lab website)      In the supplementary components of a journal article
A custom website (e.g. a lab website)
```

```
A custom website (e.g. a lab website)      In the supplementary components of a journal article
```

Becomes:

```
hosting_services_18 hosting_services_19 hosting_services_20
Custom Website      Article Supplement
Custom Website
```

```
Custom Website      Article Supplement
```

```
shorten_long_responses <- function(df, codes) {
  new_df <- df
  for (keyword in names(codes)) {
    new_df <- shorten_long_response(new_df, keyword, codes[[keyword]])
  }
  return(new_df)
}
shorten_long_response <- function(df, keyword, replacement) {
  pattern <- paste0("^", stringr::fixed(keyword))
```

```
df <- df %>%
  mutate(across(
    where(is.character),
    ~ ifelse(str_starts(.x, keyword), replacement, .x)
  ))
return(df)
}
```

Load data

```
data <- load_qualtrics_data("deidentified_no_qual.tsv")
pii <- load_qualtrics_data("pii.tsv")
```

Question type 1: No way to tell what columns mean

These are the more annoying Qualtrics outputs. The column names are e.g. `challenges_1`, `challenges_2`, etc., and the entries are not informative, e.g. `Never`, `Infrequently`, etc. Presumably, `challenges_1` corresponds to the first option, `challenges_2` corresponds to the second option, etc., but we still need to check, especially since the column numbering from Qualtrics can be strange. I am manually comparing the answers in this data frame to those in the Qualtrics interface, which shows the whole response, i.e. “Limited time for writing new code”, not just “challenges_1”. (Data table, under the Data & Analysis tab.)

To be extra confident that I am comparing the same rows between the two tables, I am looking at responses associated with a particular email. After I have visually confirmed the correspondences between the column names and the survey options, I go back to the data frame that does not contain PII. This part must be done manually and interactively, and it only needs to be done once for each question, just to get the corresponding codes. Therefore, I have commented this code out, since I’ve already done it.

Clean up challenges columns

```
challenges <- data %>%
  select(
    starts_with("challenges")
  )
```

```
head(challenges)
```

	challenges_1	challenges_2	challenges_3	challenges_4	challenges_5	challenges_6
1	Always	Always	Always	Always	Always	Always
2	Frequently	Occasionally	Occasionally	Occasionally	Occasionally	Rarely
3	Frequently	Always	Occasionally	Always	Occasionally	Frequently
4	Always	Always	Frequently	Occasionally	Frequently	Always
5	Always	Always	Rarely	Occasionally	Frequently	Never
6						

	challenges_7	challenges_8	challenges_9	challenges_10	challenges_11
1	Always	Always	Always	Always	Always
2	Frequently	Occasionally	Frequently	Frequently	Frequently
3	Frequently	Occasionally	Occasionally	Rarely	Rarely
4	Occasionally	Rarely	Rarely	Frequently	Rarely
5	Never	Never	Never	Always	Occasionally
6					

	challenges_12	challenges_13	challenges_14
1	Always	Always	Always
2	Frequently	Frequently	Occasionally
3	Always	Always	Always
4	Occasionally	Frequently	Frequently
5	Occasionally	Rarely	Always
6			

STOP!!! Manually compare this data frame to the Data table in Qualtrics.

```
# emails <- pii %>%
#   select(starts_with("stay_in_touch_email"))

# t <- cbind(emails, challenges)

#I run this next line repeatedly with different emails,
#to make sure that this person's response to "challenges_1"
#matches their response to "Limited time for writing new code", etc.

# subset(t, startsWith(stay_in_touch_email, "PERSON_EMAIL_HERE"))
```

My assumption above was correct; the options are ordered as expected. Let's rename the columns accordingly.


```

challenge_codes <- c(
  "Coding time" = "challenges_1",
  "Documentation time" = "challenges_2",
  "Managing issues" = "challenges_3",
  "Attracting users" = "challenges_4",
  "Recognition" = "challenges_5",
  "Hiring" = "challenges_6",
  "Security" = "challenges_7",
  "Finding peers" = "challenges_8",
  "Finding mentors" = "challenges_9",
  "Education time" = "challenges_10",
  "Educational resources" = "challenges_11",
  "Legal" = "challenges_12",
  "Finding funding" = "challenges_13",
  "Securing funding" = "challenges_14"
)
challenges <- rename(challenges, any_of(challenge_codes))

head(challenges)

```

	Coding time	Documentation time	Managing issues	Attracting users	Recognition
1	Always	Always	Always	Always	Always
2	Frequently	Occasionally	Occasionally	Occasionally	Occasionally
3	Frequently	Always	Occasionally	Always	Occasionally
4	Always	Always	Frequently	Occasionally	Frequently
5	Always	Always	Rarely	Occasionally	Frequently
6					
	Hiring	Security	Finding peers	Finding mentors	Education time
1	Always	Always	Always	Always	Always
2	Rarely	Frequently	Occasionally	Frequently	Frequently
3	Frequently	Frequently	Occasionally	Occasionally	Rarely
4	Always	Occasionally	Rarely	Rarely	Frequently
5	Never	Never	Never	Never	Always
6					
	Educational resources	Legal	Finding funding	Securing funding	
1	Always	Always	Always	Always	
2	Frequently	Frequently	Frequently	Occasionally	
3	Rarely	Always	Always	Always	
4	Rarely	Occasionally	Frequently	Frequently	
5	Occasionally	Occasionally	Rarely	Always	
6					

Clean up importance columns

```
importance <- data %>%
  select(
    starts_with("importance_opensrc")
  )
head(importance)
```

	importance_opensrc_1	importance_opensrc_2	importance_opensrc_3
1	Very important	Very important	Very important
2	Very important	Moderately important	Important
3	Very important	Very important	Very important
4	Very important	Slightly important	Important
5	Very important	Important	Very important
6	Very important	Non-applicable	Important

	importance_opensrc_4	importance_opensrc_5
1	Very important	Very important
2	Important	Non-applicable
3	Very important	Non-applicable
4	Important	Non-applicable
5	Very important	Non-applicable
6	Important	Non-applicable

STOP!!! Manually compare this data frame to the Data table in Qualtrics.

```
# t <- cbind(emails, importance)
# subset(t, startsWith(stay_in_touch_email, "PERSON_EMAIL_HERE"))
```

```
importance_codes <- c(
  "Research" = "importance_opensrc_1",
  "Teaching" = "importance_opensrc_2",
  "Learning" = "importance_opensrc_3",
  "Professional Development" = "importance_opensrc_4",
  "Job" = "importance_opensrc_5"
)
importance <- rename(importance, any_of(importance_codes))
head(importance)
```

	Research	Teaching	Learning	Professional Development
1	Very important	Very important	Very important	Very important

2	Very important	Moderately important	Important	Important
3	Very important	Very important	Very important	Very important
4	Very important	Slightly important	Important	Important
5	Very important	Important	Very important	Very important
6	Very important	Non-applicable	Important	Important

Job

1	Very important
2	Non-applicable
3	Non-applicable
4	Non-applicable
5	Non-applicable
6	Non-applicable

Clean up project size columns

```
size <- data %>%
  select(
    starts_with("project_size")
  )
```

STOP!!! Manually compare this data frame to the Data table in Qualtrics.

```
# t <- cbind(emails, size)
# subset(t, startsWith(stay_in_touch_email, "PERSON_EMAIL_HERE"))
```

```
size_codes <- c(
  "Small" = "project_size_1",
  "Medium" = "project_size_2",
  "Large" = "project_size_3"
)
size <- rename(size, any_of(size_codes))
head(size)
```

	Small	Medium	Large
1	Relatively frequently	Occasionally	Relatively infrequently
2	Occasionally	Relatively infrequently	Never
3	Occasionally	Relatively infrequently	Never
4	Relatively frequently	Relatively infrequently	Never
5	Relatively frequently	Occasionally	Relatively infrequently
6			

Clean up solutions columns

```
solutions <- data %>%
  select(
    starts_with("solution_offerings")
  )
```

STOP!!! Manually compare this data frame to the Data table in Qualtrics.

```
# t <- cbind(emails, solutions)
# subset(t, startsWith(stay_in_touch_email, "PERSON_EMAIL_HERE"))
```

```
solution_codes <- c(
  "Computing environments" = "solution_offerings_1",
  "Publicity" = "solution_offerings_2",
  "Containerization" = "solution_offerings_3",
  "Documentation help" = "solution_offerings_4",
  "A learning community" = "solution_offerings_5",
  "Event planning" = "solution_offerings_6",
  "Mentoring programs" = "solution_offerings_7",
  "Education" = "solution_offerings_8",
  "Legal support" = "solution_offerings_9",
  "Industry partnerships" = "solution_offerings_10",
  "Sustainability grants" = "solution_offerings_11",
  "Help finding funding" = "solution_offerings_12"
)
solutions <- rename(solutions, any_of(solution_codes))
head(solutions)
```

	Computing environments	Publicity	Containerization	Documentation help
1	Very useful	Very useful	Very useful	Very useful
2	Useful	Very useful	Very useful	Not very useful
3	Very useful	Very useful	Very useful	Very useful
4	Not very useful	Useful	Useful	Very useful
5	Useful	Not very useful	Useful	Very useful
6				
	A learning community	Event planning	Mentoring programs	Education
1	Very useful	Very useful	Very useful	Very useful
2	Useful	Non-applicable	Very useful	Very useful
3	Useful	Useful	Useful	Not very useful

4	Not very useful	Useful	Not very useful	Not very useful
5	Not very useful	Not very useful	Useful	Very useful
6				
	Legal support	Industry partnerships	Sustainability grants	
1	Very useful	Very useful	Very useful	
2	Very useful	Useful	Very useful	
3	Very useful	Very useful	Very useful	
4	Useful	Not very useful	Very useful	
5	Useful	Useful	Very useful	
6				
	Help finding funding			
1	Very useful			
2	Useful			
3	Very useful			
4	Very useful			
5	Useful			
6				

Clean up contributor status columns

```
status <- data %>% select(
  starts_with("contributor_status")
)
```

STOP!!! Manually compare this data frame to the Data table in Qualtrics.

```
# t <- cbind(emails, status)
# subset(t, startsWith(stay_in_touch_email, "PERSON_EMAIL_HERE"))
```

```
status_codes <- c(
  "Past" = "contributor_status_1",
  "Future" = "contributor_status_2"
)
status <- rename(status, any_of(status_codes))
head(status)
```

	Past	Future
1	True	True
2	True	True
3	True	True

```
4 True True
5 True True
6 False True
```

Question type 2: Rename columns based on entries

In these cases, we can tell what the column means because the entries are either the response itself or an empty string. These are essentially dummy variables with yes/no outcomes. We don't need to manually match the data frame to the responses in the Qualtrics interface. Even better, we don't need a list of codes.

For contributor_roles_Q4 and project_types_Q7, we included a colon in the survey response, so I am using my strip_descriptions function to strip everything after the colon because that seems easier than typing out a list of codes.

Clean up contributor roles columns

```
roles <- data %>% select(starts_with("contributor_role"))

roles <- strip_descriptions(roles)

# Change the column names to more useful labels
roles <- rename_cols_based_on_entries(roles)

# Shorten this one long role
names(roles) <- gsub(
  "^Other.*",
  "Other",
  names(roles)
)

roles <- make_df_binary(roles)
head(roles)
```

	Maintainer	Contributor	Bug/Issue Reporter	Community Manager	Educator	Other
1	1	1	1	1	1	0
2	0	1	0	0	0	0
3	1	1	1	0	1	0
4	1	1	1	0	1	0
5	1	1	1	0	1	0

6	0	0	0	0	0	0
	Supervisor	IT/Systems administrator	UI/UX Designer	Technical support		
1	1		0	0		1
2	0		0	0		0
3	0		0	1		0
4	1		0	0		0
5	1		0	0		0
6	0		0	0		0

Clean up project types columns

```
types <- data %>% select(starts_with("project_types"))

types <- strip_descriptions(types)

# Change the column names to more useful labels
types <- rename_cols_based_on_entries(types)

# Shorten this one long role
names(types) <- gsub(
  "^Other.*",
  "Other",
  names(types)
)

types <- make_df_binary(types)
head(types)
```

	Applications	Other	Website code	Plug-ins or extensions
1	1	0	0	1
2	0	0	0	0
3	1	0	1	1
4	1	0	1	1
5	1	0	0	1
6	0	0	0	0

	Libraries, packages, or frameworks	Automation scripts	Hardware
1		1	1
2		1	0
3		1	1
4		1	0

5	0	1	0
6	0	0	0

For these next questions, the responses were often rather long and complex, so I am shortening their names with `shorten_long_responses`.

Clean up `future_contributors` columns

```
future <- data %>%
  select(
    starts_with("future_contributors")
  )

future_codes <- c(
  "Accessible conferences" = "Conferences and hackathons",
  "Access to free" = "Computing environments",
  "Educational materials" = "Educational materials",
  "An open source discussion group" = "Learning community",
  "Dedicated grants" = "Dedicated grants",
  "Networking opportunities" = "Industry networking",
  "Job" = "Academic job opportunities",
  "Other " = "Other",
  "Assistance identifying" = "Help finding funding",
  "Legal" = "Legal support",
  "A mentor" = "Mentoring programs"
)

future <- shorten_long_responses(future, future_codes)
future <- rename_cols_based_on_entries(future)
future <- make_df_binary(future)

head(future)
```

	Conferences and hackathons	Computing environments	Educational materials
1	0	0	0
2	0	0	0
3	0	0	0
4	0	0	0
5	0	0	0
6	0	1	1

	Learning community	Dedicated grants	Industry networking
1	0	0	0
2	0	0	0
3	0	0	0
4	0	0	0
5	0	0	0
6	1	0	0

	Academic job opportunities	Other Help finding funding	Legal support
1	0	0	0
2	0	0	0
3	0	0	0
4	0	0	0
5	0	0	0
6	0	0	0

	Mentoring programs
1	0
2	0
3	0
4	0
5	0
6	0

Clean up hosting_services columns

```

hosting <- data %>%
  select(
    starts_with("hosting_services")
  )

hosting_codes <- c(
  "Other" = "Other",
  "OSF" = "OSF",
  "A custom website" = "Custom Website",
  "In the supplementary" = "Article Supplement"
)
hosting <- shorten_long_responses(hosting, hosting_codes)

hosting <- rename_cols_based_on_entries(hosting)
# Manual inspection reveals no one selected Vivli
names(hosting) <- gsub("\\?", "Vivli", names(hosting))

```

```
hosting <- make_df_binary(hosting)
head(hosting)
```

	Bitbucket	Codeberg	GitHub	Gitea	GitLab	Launchpad	SourceForge	Other	Zenodo
1	0	0	1	0	1	0	0	0	1
2	0	0	1	0	0	0	0	0	1
3	0	0	1	0	0	0	0	1	1
4	0	0	1	0	0	0	0	0	1
5	0	0	1	0	0	0	0	0	1
6	0	0	0	0	0	0	0	0	0

	Dryad	Figshare	OSF	Mendeley	Data	Vivli	Dataverse	Custom	Website	Thingiverse
1	0	1	1		0	0	0		1	0
2	1	0	0		0	0	0		1	0
3	0	0	0		0	0	0		0	0
4	1	1	1		0	0	0		1	0
5	1	1	0		0	0	0		0	0
6	0	0	0		0	0	0		0	0

	Article	Supplement
1		1
2		0
3		0
4		1
5		0
6		0

Clean up motivations columns

```
motivations <- data %>%
  select(
    starts_with("motivations")
  )

codenames <- c(
  "Developing open-source" = "Job",
  "To improve the tools" = "Improve Tools",
  "To customize existing" = "Customize",
  "To build a network" = "Network",
  "To give back to" = "Give back",
  "To improve my skills" = "Skills",
  "Because it's fun" = "Fun",
```

```

  "Other " = "Other"
)

motivations <- shorten_long_responses(motivations, codenames)
motivations <- rename_cols_based_on_entries(motivations)
motivations <- make_df_binary(motivations)
head(motivations)

```

	Job	Improve	Tools	Customize	Network	Give back	Skills	Fun	Other
1	1		1	1	1	1	1	1	0
2	0		1	1	1	0	1	0	0
3	0		1	1	0	0	1	1	0
4	1		1	1	0	1	0	0	0
5	0		1	1	0	1	1	1	0
6	0		0	0	0	0	0	0	0

Question type 3: Simple multiple choice

These are questions with just a single column, where the entries indicate which choice the respondent selected. I could just save them as-is, but I think it will make my analysis scripts tidier to replace the longer responses with codes here.

```

other_quant <- data %>%
  select(
    starts_with(
      c(
        "campus",
        "favorite_solution",
        "field_of_study",
        "job_category",
        "staff_categories"
      )
    )
  )
head(other_quant)

```

	campus
1	UC Santa Barbara
2	UC Santa Barbara

3 UC Santa Barbara
 4 UC Santa Barbara
 5 UC Santa Barbara
 6 UC Santa Barbara

favorite_solution
 1 Dedicated grants for open-source project sustainability
 2 Assistance creating (i.e. Docker) containers for your open-source software
 3 Access to free, feature-rich computing environments
 4 Dedicated grants for open-source project sustainability
 5 Assistance writing documentation
 6

field_of_study
 1 Mathematical and computational sciences
 2 Life sciences
 3 Humanities
 4 Mathematical and computational sciences
 5 Life sciences
 6 Mathematical and computational sciences

job_category
 1 Faculty
 2 Post-Doc
 3 Other research staff (e.g., research scientist, research software engineer)
 4 Faculty
 5 Faculty
 6 Other research staff (e.g., research scientist, research software engineer)
 staff_categories

1
 2
 3
 4
 5
 6

```
solution_codes2 <- c(
  "Access to free" = "Computing environments",
  "Assistance promoting your" = "Publicity",
  "Assistance creating" = "Containerization",
  "Assistance writing" = "Documentation help",
  "An open source discussion" = "A learning community",
  "Assistance with event" = "Event planning",
  "A mentor" = "Mentoring programs",
  "Educational materials" = "Education",
```

```

"Legal and licensing" = "Legal support",
"Assistance building industry" = "Industry partnerships",
"Dedicated grants" = "Sustainability grants",
"Assistance identifying potential" = "Help finding funding"
)
other_quant <- shorten_long_responses(other_quant, solution_codes2)

```

```

field_codes <- c(
  "Mathematical and computational sciences" = "Math and CS"
)
other_quant <- shorten_long_responses(other_quant, field_codes)

```

```

job_codes <- c(
  "Other research staff" = "Other research staff"
)
other_quant <- shorten_long_responses(other_quant, job_codes)

```

```

staff_codes <- c(
  "Academic and Research Support" = "Academic and Research Support",
  "Other \\\(Please specify" = "Other",
  "Finance" = "Finance"
)
other_quant <- shorten_long_responses(other_quant, staff_codes)
head(other_quant)

```

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

staff_categories

1

2

3

4

5

6

Check row numbers and save

We want to make sure that all data frames have the same number of rows. Because each row corresponds to a participant, it's critical that the row numbers match, so that we can compare responses from the same participant on different questions. I'm putting all my data frames in one list, and then applying functions to each data frame in the list.

```
dfs <- list(
  "challenges_Q9.tsv" = challenges,
  "contributor_roles_Q4.tsv" = roles,
  "contributor_status_Q3.tsv" = status,
  "future_contributors_Q15.tsv" = future,
  "hosting_services_Q8.tsv" = hosting,
  "importance_Q2.tsv" = importance,
  "motivations_Q6.tsv" = motivations,
  "project_size_Q5.tsv" = size,
  "project_types_Q7.tsv" = types,
  "solutions_Q10.tsv" = solutions,
  "other_quant.tsv" = other_quant
)
sapply(dfs, function(current_df) nrow(current_df))
```

challenges_Q9.tsv	contributor_roles_Q4.tsv
332	332
contributor_status_Q3.tsv	future_contributors_Q15.tsv
332	332
hosting_services_Q8.tsv	importance_Q2.tsv
332	332
motivations_Q6.tsv	project_size_Q5.tsv
332	332
project_types_Q7.tsv	solutions_Q10.tsv
332	332
other_quant.tsv	
332	

Good! Same number of rows. Now let's save them to files in a folder. We will use the `write_df_to_file` function, which is in `scripts/utils.R`.

```
output_dir <- file.path(Sys.getenv("DATA_PATH"), "clean_data")

if (!dir.exists(output_dir)) {
```

```

    dir.create(output_dir)
  }

# invisible() suppresses noisy output text to the terminal
invisible(
  sapply(seq(length(dfs)), function(i) {
    write_df_to_file(
      dfs[[i]],
      file.path("clean_data", names(dfs[i]))
    )
  })
)

```

```
sessionInfo()
```

```

R version 4.4.2 (2024-10-31)
Platform: aarch64-apple-darwin20
Running under: macOS Sequoia 15.5

```

```
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] en_US.UTF-8/en_US.UTF-8/en_US.UTF-8/C/en_US.UTF-8/en_US.UTF-8
```

```
time zone: America/Los_Angeles
```

```
tzcode source: internal
```

```
attached base packages:
```

```

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

```

```
other attached packages:
```

```

[1] treemap_2.4-4      tidyr_1.3.1        stringr_1.5.1
[4] scales_1.4.0       readr_2.1.5        pwr_1.3-0
[7] patchwork_1.3.0    ordinal_2023.12-4.1 mvabund_4.2.1
[10] languageserver_0.3.16 here_1.0.1          gtools_3.9.5
[13] fpc_2.2-13         forcats_1.0.0      factoextra_1.0.7
[16] ggplot2_3.5.2      emmeans_1.11.1     dplyr_1.1.4
[19] corrplot_0.95      cluster_2.1.8.1

```

loaded via a namespace (and not attached):

[1] tidyselect_1.2.1	gridBase_0.4-7	farver_2.1.2
[4] fastmap_1.2.0	promises_1.3.3	digest_0.6.37
[7] mime_0.13	estimability_1.5.1	lifecycle_1.0.4
[10] statmod_1.5.0	processx_3.8.6	magrittr_2.0.3
[13] kernlab_0.9-33	compiler_4.4.2	rlang_1.1.6
[16] igraph_2.1.4	yaml_2.3.10	data.table_1.17.6
[19] knitr_1.50	mclust_6.1.1	xml2_1.3.8
[22] RColorBrewer_1.1-3	withr_3.0.2	purrr_1.0.4
[25] numDeriv_2016.8-1.1	nnet_7.3-19	grid_4.4.2
[28] stats4_4.4.2	xtable_1.8-4	colorspace_2.1-1
[31] MASS_7.3-61	prabclus_2.3-4	cli_3.6.5
[34] mvtnorm_1.3-3	rmarkdown_2.29	generics_0.1.4
[37] robustbase_0.99-4-1	tzdb_0.5.0	modeltools_0.2-24
[40] parallel_4.4.2	vctrs_0.6.5	Matrix_1.7-1
[43] jsonlite_2.0.0	callr_3.7.6	hms_1.1.3
[46] ggrepel_0.9.6	diptest_0.77-1	tweedie_2.3.5
[49] glue_1.8.0	DEoptimR_1.1-3-1	ps_1.9.1
[52] stringi_1.8.7	gtable_0.3.6	later_1.4.2
[55] tibble_3.2.1	pillar_1.10.2	htmltools_0.5.8.1
[58] R6_2.6.1	ucminf_1.2.2	rprojroot_2.0.4
[61] evaluate_1.0.3	shiny_1.11.0	lattice_0.22-6
[64] renv_1.1.4	httpuv_1.6.16	class_7.3-22
[67] Rcpp_1.0.14	flexmix_2.3-20	nlme_3.1-166
[70] xfun_0.52	pkgconfig_2.0.3	