Package 'FPSurveyR'

February 3, 2025

Title Farm Practices Survey -
an R package containing functions for data analysis and statistical notice production

Version 1.2.0

Description This package contains all the required functions for pre-processing the Farm Practises Survey data exported from MS Access by the June team, analysing the data to produce means/ratios and associated confidence intervals, and undertaking post-processing to format the data correctly for output as part of a RAP using parameterised R markdown to produce the statistical report on GOV.UK.

License MIT + file LICENSE

Encoding UTF-8

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.2

Depends R (>= 3.5.0)

Suggests knitr,
rmarkdown,
testthat (>= 3.0.0)

VignetteBuilder knitr

Imports magrittr, rlang,
dplyr, tidyr,
readr, openxlsx, readxl, RODBC,
lubridate,
srvyr, purrr, forcats,
reshape2, janitor, data.table,
here, cli,
glue, english, snakecase,
scales,
afcharts

Config/testthat/edition 3

Contents

`fps_add_empty_factors` *FPS: Analysis: Add empty factors*

Description

This function ensures that all expected factor levels, including "All farms," are present in a dataset. If a factor level is missing, an empty row (filled with `NA`) is added to the dataset and the data are re-ordered to preserve the dataset dimensions.

Usage

```
fps_add_empty_factors(.data, factor_col = "cat", factor, factors_list_full)
```

Arguments

`.data`	A data frame containing the analysis data.
`factor_col`	A string specifying the column name in `.data` containing the factor categories. Defaults to `"cat"`.
`factor`	A string specifying the name of the factor to check for missing levels.

factors_list_full
: A named list of all expected factor levels, where each name corresponds to a factor, and the value is a vector of levels. The data are also re-ordered (by row) depending on the order of the supplied factor levels

Details

If a category from factors_list_full is missing in the specified factor_col, this function adds an empty row for that category at the correct location. A warning message is displayed for each missing category that is added.

Value

A re-oredered data frame with rows added for any missing factor levels, filled with NA values.

Author(s)

Tom Pearson

Examples

```
#data <- data.frame(cat = c("A", "B"), value = c(10, 20))
#factors_list_full <- list(my_factor = c("A", "B", "C"))
#updated_data <- fps_add_empty_factors(data, factor_col = "cat", factor = "my_factor", factors_list_full = factors
```

---

fps_analyse_by_factor *FPS: Analysis: Analyse by factor*

Description

Computes means or ratios for the Farm Practices Survey (FPS) data using a survey design object. This function is designed for use with fps_prepare_results.

Usage

```
fps_analyse_by_factor(
  design,
  factor,
  variable,
  denominator = NULL,
  ratio = FALSE,
  factor_col = "cat"
)
```

Arguments

design	A survey design object created using the survey or srvyr packages.
factor	A character string specifying the name of the grouping variable in the dataset.
variable	A character string specifying the name of the binary variable to analyze.
denominator	(Optional) A character string specifying the denominator variable for ratio calculations. Required if ratio = TRUE.
ratio	A logical value indicating whether to calculate ratios (TRUE) or means (FALSE). Defaults to FALSE.
factor_col	A character string specifying the column name for the factor in the output. Defaults to "cat".

Details

- This function assumes the variable is binary (e.g. 1 for presence and 0 for absence).
- Confidence intervals are computed at a 95% confidence level by default.

Value

A data frame containing the computed statistics, including:

- Unweighted counts.
- Weighted means or ratios.
- Confidence intervals.

Author(s)

Tom Pearson (adapted to srvyr methods)

Examples

```
#result <-
#fps_analyse_by_factor(
#  design = fps_design, #Assuming `fps_design` is a survey design object
#  factor = "fps_gor",
#  variable = "Q1_1",
#  ratio = FALSE)
```

fps_analysis *FPS: Analysis: High-level analysis*

Description

This function performs high-level analysis for the Farm Practices Survey (FPS). It processes survey data, sets the survey design, and prepares results using lower-level functions (`fps_process_factors`, `fps_prepare_results`, and `fps_analyse_by_factor`). The analysis outputs include formatted results and tables, written to XLSX files for quality assurance and subsequent use in the RAP process.

Usage

```
fps_analysis(
  .data,
  factor_col = "cat",
  questions_list,
  standard_factors_list,
  adhoc_factors_list = NULL,
  adhoc_factors_levels_list = NULL,
  ratio = FALSE,
  results_fp,
  tables_fp
)
```

Arguments

`.data`	A data frame containing the survey data to be analysed. The data frame must include relevant columns for defining the survey design (e.g. `cph_no`, `post_strat`, `num_pop`).
`factor_col`	A character string specifying the column containing the categorical variable(s) used for grouping in the analysis. Defaults to `"cat"`. The specified column should exist in `.data`.
`questions_list`	A named list where each element corresponds to a survey question. Each element must be a character vector consisting of: • The question identifier prefixed with `"answered_"` (e.g. `"answered_Q1"`). • Response options associated with the question.
`standard_factors_list`	A named list of standard factors used for grouping analysis. Each element of the list should be a named vector or a list defining factor levels.
`adhoc_factors_list`	An optional named list of ad-hoc factors to be included in the analysis. These factors are added dynamically to `standard_factors_list`. Defaults to `NULL`.
`adhoc_factors_levels_list`	An optional named list specifying the levels for the ad-hoc factors. This must align with `adhoc_factors_list` to ensure consistency. Defaults to `NULL`.

ratio	Logical. If TRUE, the analysis calculates ratios for the specified questions using a denominator derived from the variable names in questions_list. If FALSE, means are calculated. Defaults to FALSE.
results_fp	A character string specifying the file path where intermediate analysis results will be saved as XLSX files. The path must exist, and the function appends the results filenames.
tables_fp	A character string specifying the file path where the final formatted tables will be saved as XLSX files. The path must exist, and the function appends the tables filenames.

Details

The function performs the following steps:

- Validates input arguments and ensures correct structure for all inputs.
- Iterates through the questions specified in questions_list to subset data for each question.
- Dynamically processes factors, including standard and ad-hoc factors, using fps_process_factors.
- Sets the survey design using srvyr::as_survey_design.
- Computes analysis results and writes intermediate outputs using fps_prepare_results.
- Formats results into tables compatible with further RAP processing and ensures missing factor levels are explicitly represented using fps_add_empty_factors.
- Writes the final tables to XLSX files in the location specified by tables_fp.
- Factors with missing data are assigned NA values in outputs.

Value

A named list (table_list) containing the formatted results for each question. Each element is a nested list of data frames, one for each factor analysed for the respective question.

Author(s)

Tom Pearson

See Also

fps_process_factors, fps_prepare_results, fps_analyse_by_factor

Examples

```
#fps_analysis(
#  .data = s1_data,
#  factor_col = "cat",
#  questions_list = s1_questions_list,
#  standard_factors_list = factors_list,
#  adhoc_factors_list = s1_adhoc_factors_list,
#  adhoc_factors_levels_list = s1_adhoc_factors_levels_list,
#  ratio = FALSE,
#  results_fp = "results/s1_results",
```

```
#  tables_fp = "tables/s1_tables")
```

`fps_calc_weights` *FPS: Pre-processing: Calculate weights*

Description

This function calculates survey weights for Farm Practices Survey (FPS) data by determining the number of observations per stratum, joining it to the population data, and calculating weights as the ratio of the population size to the number of observations. The population column is then joined back into the dataset, and the strata are converted to factors.

Usage

```
fps_calc_weights(
  .data,
  pop_df,
  pop_col = "npop",
  unique_ref_col = "cph_no",
  join_col = "post_strat"
)
```

Arguments

`.data`	A data frame containing survey data. Must include the columns specified by `unique_ref_col` and `join_col`.
`pop_df`	A data frame containing population data by stratum. Must include columns specified by `join_col` and `pop_col`.
`pop_col`	A string specifying the column in `pop_df` containing population sizes for each stratum. Default is `"npop"`.
`unique_ref_col`	A string specifying the column in `.data` containing unique identifiers for observations. Default is `"cph_no"`.
`join_col`	A string specifying the column in `.data` and `pop_df` used to join the datasets (strata column). Default is `"post_strat"`.

Value

A data frame with the population column (`num_pop`) added to the input data, and strata (`join_col`) converted to factors. Weights are calculated but not included in the output, as the `svyr` package automatically computes these when using `fpc`.

Author(s)

Tom Pearson

Examples

```
#survey_data <- data.frame(
#  cph_no = 1:10,
#  post_strat = rep(c("A", "B"), each = 5))
#
#population_data <- data.frame(
#  post_strat = c("A", "B"),
#  npop = c(100, 200))
```

`fps_chart` *FPS: Charts: Render image file imitation of GOV.UK auto-barchart for QA*

Description

This function generates custom bar charts for Farm Practices Survey (FPS) data using ggplot2. The charts mimic GOV.UK auto-barchart formatting and are intended for QA purposes.

Usage

```
fps_chart(.data, questions, years, yaxis, fill, stacked = TRUE)
```

Arguments

`.data`	A named list of data frames, where each list element corresponds to a survey question and contains its associated data. Each data frame must include columns `year`, `value`, and any variables specified by `yaxis` and `fill`.
`questions`	A character vector of one or more question identifiers (e.g. "Q1", "Q2") to extract and visualize data for. Each identifier must match a name in `.data`.
`years`	A numeric vector of survey years to include in the chart.
`yaxis`	A character string specifying the variable for the y-axis (e.g. `"year"` or `"response"`).
`fill`	A character string specifying the variable to use for fill colors (e.g. `"year"` or `"response"`).
`stacked`	Logical. If `TRUE`, the chart will use a stacked bar format; if `FALSE`, it will use a dodged bar format. Defaults to `TRUE`.

Details

The function:

- Extracts and combines data for the specified questions.
- Filters data by the given years.
- Formats data for GOV.UK-style barcharts with appropriate aesthetics and color schemes.
- Dynamically adjusts graph parameters, such as text size and bar positioning, based on the input data.

Value

A ggplot object representing the bar chart.

Author(s)

Tom Pearson

Examples

```
#fps_chart(
#  .data = intro_tbl_list,
#  questions = "Q1",
#  years = c(2023, 2024),
#  yaxis = "year",
#  fill = "response",
#  stacked = TRUE)
```

---

fps_data_to_binary *FPS: Pre-processing: Data to binary*

Description

This function dynamically converts specified columns in a dataset to binary indicator columns based on a named list of question values. For each column, a new binary column is created for each value in the corresponding vector of options.

Usage

```
fps_data_to_binary(.data, question_values, nonbinary_questions = NULL)
```

Arguments

`.data`	A data frame containing the input dataset.
`question_values`	A named list where the names are column names in the dataset, and the values are numeric vectors of all possible options for those columns. If the question is non-binary, then an empty vector or `NULL` must be provided.
`nonbinary_questions`	A character vector of questions to exclude from processing to binary (e.g. `"Q1_TEXT"`). Default is `NULL`.

Details

For character columns, the function uses regular expressions to identify matches. For numeric columns, direct equality checks are performed. The function assumes all options provided in `question_values` are exhaustive for the corresponding columns.

Value

A data frame with the original data and additional binary indicator columns (e.g. "Q1_v1").

Author(s)

Tom Pearson

Examples

```
# s8_data <- data.frame(
#   Q56 = c(1, 2, 3, 4, 5),
#   Q57 = c("1,2", "3,4", "6", "1,3", "6,4"),
#   Q57_TEXT = c("example1", "example2", "example3", "example4", "example5"),
#   Q58 = c(10, 25, 50, 1, 10)
# )
#
# question_values <- list(
#   Q56 = 1:5, # Exhaustive options for Q56
#   Q57 = 1:6,  # Exhaustive options for Q57
#   Q57_TEXT = c(),
#   Q58 = c()
# )
#
# fps_data_to_binary(s8_data, question_values, nonbinary_questions = c("Q57_TEXT", "Q58"))
```

`fps_delete_old_versions` *FPS: Render: Delete old versions (of reports/charts)*

Description

This function deletes old versions of reports and charts in the specified output directory if the `delete` parameter is set to `TRUE`. The deletion is based on a time period defined by `delete_unit` and `delete_number`. The function is designed to be used prior to rendering a new report to ensure only the current reports/charts are being used

Usage

```
fps_delete_old_versions(
  out_dir,
  section = NULL,
  delete = FALSE,
  delete_unit = "hours",
  delete_number = 12
)
```

Arguments

out_dir	Character. The base directory containing the report and charts subdirectories.
section	Character (optional). The specific section or subdirectory under report and charts to process. Defaults to NULL, meaning no specific section is targeted.
delete	Logical. If TRUE, old files are deleted. Defaults to FALSE.
delete_unit	Character. The time unit for the deletion threshold (e.g. "secs", "mins", "hours", "days", "weeks"). See difftime. Defaults to "hours".
delete_number	Numeric. The number of delete_unit units defining the threshold for deletion. Defaults to 12.

Author(s)

Tom Pearson

See Also

difftime

Examples

```
#fps_delete_old_versions(out_dir = "output", section = "01_nutrient_management",
#                        delete = TRUE, delete_unit = "days", delete_number = 30)
```

---

fps_dodged *FPS: Charts: Dodged auto-barchart for GOV.UK*

Description

This function processes survey data to generate a table for use on GOV.UK as a dodged chart using the {barchart} tag. The output displays the proportion of responses for specified survey years, either by response or by year. It allows for the visualization of how responses evolve across survey years and can pivot the chart either by response or by year.

Usage

```
fps_dodged(.data, questions, svy_years, pivot_from_response = FALSE)
```

Arguments

.data	A tbl_list for a specific section containing survey data for different questions.
questions	A character vector of questions (e.g. "Q1", "Q2", etc.) for which the data should be processed. The function supports multiple questions but assumes the data is comparable across them.
svy_years	A vector of survey years to include in the chart (e.g. c(2020, 2021, 2022)).

pivot_from_response
: A logical value. If TRUE, pivots the data from the "response" variable to the response values. If FALSE (default), pivots the data from the "year" variable to create columns for each year.

Details

- The function works with survey data stored in tbl_list for each section. Each table should represent a survey question.
- It processes multiple questions by combining the data for all specified questions into one dataset.
- The function handles two types of pivoting:
  - Pivoting by "response" (pivot_from_response = TRUE): This will create one column per response category.
  - Pivoting by "year" (pivot_from_response = FALSE): This will create one column per survey year.
- It filters data for the specified survey years and formats the responses as percentages.

Value

A table formatted as a knitr object displaying the proportion of responses in a dodged format. The chart can be pivoted either by response or by year, depending on the value of pivot_from_response.

Author(s)

Tom Pearson

Examples

```
#fps_dodged(survey_data, questions = c("Q1", "Q2"), svy_years = c(2020, 2021), pivot_from_response = TRUE)
```

---

fps_format_factors *FPS: Pre-processing: Format factors*

Description

This function formats the factor variables in a dataset based on standard FPS conventions. It renames specific columns for region, farm size, farm type, and stratification to their standard names, applies specific transformations to values in these columns, and ensures consistent formatting.

Usage

```
fps_format_factors(
  .data,
  region_col = "fps_nuts_name",
  farm_size_col = "fps_slr_name",
  farm_type_col = "fps_robust",
  strat_col = "post_strat"
)
```

Arguments

.data	A data frame containing the data to format.
region_col	A character string specifying the column name for the region. Default is "fps_nuts_name".
farm_size_col	A character string specifying the column name for the farm size. Default is "fps_slr_name".
farm_type_col	A character string specifying the column name for the farm type. Default is "fps_robust".
strat_col	A numeric or character string specifying the column name for the stratification. Default is "post_strat".

Value

A data frame with formatted factor variables and renamed columns.

Author(s)

Tom Pearson

Examples

```
#data <- data.frame(
#  fps_gor = c("North West", "South East (England)", "Yorkshire"),
#  fps_slr_name = c("small", "medium", "large"),
#  fps_robust = c("Pigspoultry", "Dairy", "Arable"),
#  post_strat = c("1", "2", "3"))
#formatted_data <- fps_format_factors(data)
```

---

fps_format_md *FPS: Render: Format Markdown files for GOV.UK*

Description

This function processes .md files in a specified output directory, removing YAML headers and replacing image tags to conform to GOV.UK formatting standards.

Usage

```
fps_format_md(
  out_dir_md = ".",
  out_dir_figs = ".",
  section = NULL,
  chart_ext = "svg",
  date = tolower(format(Sys.Date(), "%d%b%y"))
)
```

Arguments

out_dir_md	Character. The directory containing the markdown files. Defaults to the current directory ("..").
out_dir_figs	Character. The directory containing the image files. Defaults to the current directory ("..").
section	Character (optional). A specific subdirectory or section within the output directories to process. Defaults to NULL.
chart_ext	Character. The file extension of chart images (e.g. "svg", "png"). Defaults to "svg".
date	Character. The date string used to filter .md files. Defaults to today's date in the format "%d%b%y" (e.g. "14jan25").

Details

- YAML headers in the .md files are removed.
- Image tags are updated to match GOV.UK standards.
- Only .md (non-README) files created on the specified date are processed.

Value

None. The function modifies .md files in place as a side effect.

Author(s)

Tom Pearson

Examples

```
#fps_format_md(out_dir_md = "output/report", out_dir_figs = "output/charts", section = "section_name")
```

fps_make_allfarms_table

FPS: Dataset: Make "All farms" table

Description

This function processes a list of tables and integrates them with a corresponding lookup list to create "All Farms" summary tables. It is primarily used for % questions as in section 5 (e.g. Q35; Q39) where a comparison of the overall results across related questions is required

Usage

```
fps_make_allfarms_table(table_list, allfarms_lookup_list, ratio = FALSE)
```

Arguments

table_list	A named list of data frames containing the tables to be processed.
allfarms_lookup_list	A named list where names are the names of the resulting "All farms" table, and values are named vectors mapping sub-questions to their descriptive names (response options).
ratio	Logical, defaults to FALSE. If TRUE, columns are referenced assuming the analysis method was "ratio", otherwise "mean" is used.

Value

A modified list of tables, including aggregated rows for the "All Farms" category based on the lookup list.

Author(s)

Tom Pearson

Examples

```
#table_list <- example_table_list
#allfarms_lookup_list <-
#list("Q35_allfarms" = c("Q35_1" = "Stored under cover",
#                   "Q35_2" = "Stored in the open on a concret (impereable) base: covered",
#                  "Q35_3" = "Stored in the open on a concret (impereable) base: uncovered",
#                 "Q35_4" = "Stored in the open on a field site (no construction base): covered",
#                 "Q35_5" = "Stored in the open on a field site (no construction base): uncovered",
#                         "Q35_6" = "Store in any other type of facility"))
#result <- fps_make_allfarms_table(table_list, allfarms_lookup_list, ratio = FALSE)
```

`fps_name_responses` *FPS: Name responses: Label the coded responses in the analysis results*

Description

Joins coded variable names from analysis outputs to human-readable names manually specified in a lookup dataframe, optionally renames the tables, and processes them for narrative automation later on.

Usage

```
fps_name_responses(
  table_list,
  analysis_questions,
  response_lookup_df,
  survey_year = as.numeric(format(Sys.Date(), "%Y")),
  ratio = FALSE,
  category = "All farms",
  factor = 1,
  rename = FALSE
)
```

Arguments

`table_list`	A list of data frames containing analysis outputs.
`analysis_questions`	A list or vector of questions analysed (must match table names in `table_list`).
`response_lookup_df`	A data frame mapping response codes to human-readable names, including a "question" column.
`survey_year`	Numeric. Defaults to the current system year (e.g. 2025). This will be appended to the output table so that data can be distinguished when adding to the time series later on.
`ratio`	Logical. If `TRUE`, use "ratio" instead of "mean" as the aggregation method assumed to be used in the analysis. Default is `FALSE`.
`category`	Character. The category to filter by (e.g. "All farms"). Default is `"All farms"`.
`factor`	The factor or grouping variable to extract from `table_list`. Default is `1` which returns the value associated with the 1st factor in the table (values are the same across all factors of "All farms").
`rename`	Logical or character. If `TRUE`, attempts to rename the tables based on extracted response patterns. If a character string, uses it as the new name. Default is `FALSE`.

Value

A list of processed data frames with human-readable response names.

Author(s)

Tom Pearson

Examples

```
#result <- fps_name_responses(
#  table_list = s8_livestock_table_list,
#  analysis_questions = s8_livestock_questions_list,
#  response_lookup_df = s8_livestock_response_lookup_df,
#  survey_year = 2025,
#  ratio = TRUE,
#  category = "dairy",
#  factor = "fps_robust",
#  rename = TRUE)
```

---

fps_prepare_results *FPS: Analysis: Prepare results*

Description

This function calculates means or ratios for survey data in binary format (using `fps_analyse_by_factor`), formats the results, and saves them to an Excel file. The function iterates over multiple factors and variables, performing the analysis and saving the results into separate sheets (1 for each factor) in the output file.

Usage

```
fps_prepare_results(
  design,
  factors,
  variables,
  denominator = NULL,
  ratio = FALSE,
  factor_col = "cat",
  excel_file_path
)
```

Arguments

`design`	A survey design object created using the `survey` or `srvyr` packages.
`factors`	A character vector specifying the grouping variables (factors) for the analysis.
`variables`	A character vector specifying the binary variables to analyze.

denominator	(Optional) A character string specifying the denominator variable for ratio calculations. Required if ratio = TRUE.
ratio	A logical value indicating whether to calculate ratios (TRUE) or means (FALSE). Defaults to FALSE.
factor_col	A character string specifying the column name for factors in the output. Defaults to "cat".
excel_file_path	A character string specifying the file path where the Excel workbook will be saved.

Details

- Means are calculated when ratio = FALSE (default). Ratios are calculated when ratio = TRUE, and a denominator variable must be provided.
- Results are saved in separate sheets named after each factor in the factors parameter.
- Binary variables in variables should contain 1 for presence and 0 for absence.
- For variables with "answered" in their names, warnings from convergence issues are suppressed to avoid irrelevant messages.

Value

Saves the analysis results into an Excel file at the specified excel_file_path.

Author(s)

Tom Pearson

Examples

```
#fps_prepare_results(
#  design = fpsdesign,
#  factors = c("region", "farm_type"),
#  variables = c("use_cover_crops", "answered"),
#  denominator = "total_area",
#  ratio = TRUE,
#  factor_col = "group",
#  excel_file_path = "results.xlsx")
```

---

fps_process_factors *FPS: Analysis: Process factors*

Description

This function processes factors in the FPS dataset, including standard factors (i.e. region, farm type, and farm size) and ad-hoc factors provided by the user depending on the question. Ad-hoc factors are added to the dataset, their levels are mapped and ordered, and both standard and ad-hoc factors are prepared for analysis.

Usage

```
fps_process_factors(
  .data,
  question,
  factors,
  factors_list,
  adhoc_factors_list = NULL,
  adhoc_factors_levels_list = NULL
)
```

Arguments

.data	A data frame containing the FPS data.
question	A character string specifying the question of which to process the factors for.
factors	A character vector of existing factor column names to include in the analysis.
factors_list	A named list of factors with their levels to filter and order the factors.
adhoc_factors_list	A named list where each question maps to additional ad-hoc factor column names. Defaults to NULL.
adhoc_factors_levels_list	A named list mapping ad-hoc factor column names to their levels. Defaults to NULL.

Value

A list with four components:

data The processed data frame with factors formatted.

factors An updated character vector of all factors included in the analysis.

factors_list A filtered and ordered named list of factors with levels used in the analysis.

factors_list_full A complete list of factors (including ad-hoc factors) with their levels.

Author(s)

Tom Pearson

Examples

```
#processed <-
#fps_process_factors(
#  .data = df,
#  question = "Q1",
#  factors = c("fps_gor", "fps_slr_name"),
#  factors_list = factors_list,
#  adhoc_factors_list = adhoc_factors_list,
#  adhoc_factors_levels_list = adhoc_factors_levels_list)
```

`fps_read_excel_allsheets` *FPS: Analysis: Read all Excel sheets*

Description

This function reads all sheets from an Excel workbook and returns them as a list of data frames.

Usage

```
fps_read_excel_allsheets(filename)
```

Arguments

`filename`	A string specifying the path to the Excel file.

Details

Each sheet in the Excel workbook is read into a separate data frame. The sheets are converted to data frames regardless of their original format.

Value

A named list of data frames, where each element corresponds to a sheet in the workbook. The names of the list elements are the sheet names.

Author(s)

Tom Pearson

Examples

```
#sheets <- fps_read_excel_allsheets("data.xlsx")
```

`fps_remove_no_column` *FPS: Dataset: Remove "No" column*

Description

This function processes a list of tables (`table_list`) by removing columns corresponding to "No" responses from specified questions, ensuring data consistency for further analysis. Optionally, data for questions with "No" responses removed can be combined with similarly formatted questions

Usage

```
fps_remove_no_column(table_list, questions, questions_to_combine_list = NULL)
```

Arguments

table_list	A named list of lists, where each list element corresponds to a question and each list element within each question corresponds to a table of factor-based results.
questions	A character vector specifying question names to process which should match the names of at least one of the tables in table_list.
questions_to_combine_list	An optional named list. The names represent the parent question to which a related question will be combined to. The value is a single character vector containing the related question to combine to the parent question.

Value

A modified version of table_list with processed columns and optionally combined questions.

Author(s)

Tom Pearson

Examples

```
#table_list <- list(Q9_exclNA = ..., Q18_exclNA = ...)
#questions <- c("Q9_exclNA", "Q18_exclNA")
#questions_to_combine_list <- list("Q9_exclNA" = "Q13_exclNA")
#result <- fps_remove_no_column(table_list, questions, questions_to_combine_list)
```

---

fps_response_lookup_as_df

FPS: Name responses: Response lookup as data frame

Description

Converts a list of questions containing responses (coded and English-readable) into a structured dataframe. The dataframe is designed for later use in operations such as left_join to match response codes with their corresponding English-readable response names.

Usage

```
fps_response_lookup_as_df(.data, ratio = FALSE)
```

Arguments

.data	A named list or similar object where each element represents a question containing named vectors pairing a response option code to its English-readable response.
ratio	A logical value indicating whether to use "mean" or "ratio" as a suffix for the response column. Defaults to FALSE, which appends "_mean".

Value

A dataframe with columns:

response_names The original response values extracted from `.data`.

question The question names derived from `.data`.

response The response codes with a suffix ("_mean" or "_ratio").

Author(s)

Tom Pearson

Examples

```
#s1_response_lookup_list <- list("Q1" = c("Q1_1" = "Holdings with a nutrient management plan",
#                                  "Q1_2" = "Holdings without a nutrient management plan",
#                                         "Q1_3" = "Not applicable"))
#fps_response_lookup_as_df(s1_response_lookup_list, ratio = TRUE)
```

`fps_stacked` *FPS: Charts: Stacked auto-barchart for GOV.UK*

Description

This function processes survey data to create a table for use on GOV.UK as a stacked percentage chart using the `{barchart stacked}` tag. The output shows the proportions of different responses over the specified years. It supports multiple questions and allows for comparison across different survey years.

Usage

```
fps_stacked(.data, questions, svy_years)
```

Arguments

`.data`	A `tbl_list` for a specific section containing survey data for different questions.
`questions`	A character vector of questions (e.g. "Q1", "Q2", etc.) for which the data should be processed. The function supports multiple questions but assumes the data is comparable across them.
`svy_years`	A vector of survey years to include in the chart (e.g. `c(2020, 2021, 2022)`).

Details

- The function works with survey data stored in `tbl_list` for each section. Each table should represent a survey question.
- It filters the data to include only the specified survey years, formats the response proportions as percentages, and arranges the data in a stacked format.
- The input questions should be valid and exist in the `.data` list. If a question is not collected in the given years, the function will notify the user.
- The function will process data for multiple questions by combining their data into one, ensuring the columns are compatible.

Value

A table formatted as a knitr object showing the proportion of different responses in a stacked format, for each year in `svy_years`. Each response is shown as a percentage of the total, with a "100%" total column added.

Author(s)

Tom Pearson

Examples

```
#fps_stacked(survey_data, questions = c("Q1", "Q2"), svy_years = c(2020, 2021, 2022))
```

`fps_update_dataset` *FPS: Dataset: Update dataset*

Description

This function processes survey results into a workbook, formatting data and performing validation checks before inserting data into a specific worksheet at specific row positions in the workbook. Ad-hoc factors can be added to customize the processing for specific questions.

Usage

```
fps_update_dataset(
  table_list,
  questions,
  standard_factors_list,
  workbook,
  sheet,
  special_qs,
  adhoc_factors_list = NULL,
  adhoc_factors_levels_list = NULL,
  ratio = FALSE
)
```

Arguments

table_list	A named list of data tables. Each table contains the results for a specific question and factor.
questions	A named list of row number vectors. Each element's name (e.g. "Q1") should correspond to a question in table_list. Each row number value corresponds to the first row of the table in the dataset template that each question's data (for each factor) should be inserted.
standard_factors_list	A named list of factors and their corresponding levels for standard questions.
workbook	An openxlsx workbook object where the data will be written.
sheet	A character string specifying the name of the worksheet where the data will be written.
special_qs	A character vector of question names that require special validation handling.
adhoc_factors_list	An optional named list of ad-hoc factors to be included for specific questions.
adhoc_factors_levels_list	An optional named list of factor levels corresponding to adhoc_factors_list.
ratio	Logical. If TRUE, calculates ratios; if FALSE, calculates means (default is FALSE).

Value

None. Updates the workbook in memory. The workbook must be saved separately using saveWorkbook.

Author(s)

Tom Pearson

See Also

saveWorkbook

Examples

```
#fps_update_dataset(table_list, questions, standard_factors_list, workbook, "Sheet1", special_qs)
```

`fps_update_timeseries` *FPS: Time-series: Update time-series*

Description

Updates the time-series data for each table or chart displayed in the statistical notice. The function reads in existing time-series data from CSV files and appends the latest survey data taken from the analysis so the full time-series is updated and ready to be used with the narrative functions. The updated data are also written back to disk.

Usage

```
fps_update_timeseries(
  tbl_list,
  questions,
  survey_year = as.numeric(format(Sys.Date(), "%Y")),
  timeseries_directory,
  output_directory = paste0(here::here("outputs", "timeseries"), "/"),
  write_to_timeseries_directory = FALSE
)
```

Arguments

`tbl_list`	A named list of data frames, where each data frame corresponds to a table's latest survey results.
`questions`	A character vector of question identifiers to update (e.g. "Q17").
`survey_year`	Numeric. The year of the latest survey data being added (defaults to the current year).
`timeseries_directory`	A string specifying the directory path of your pipeline's outputs folder. Defaults to `"example_project/outputs/timeseries/"`
`write_to_timeseries_directory`	Logical. If `TRUE`, updated time-series data is written back to the original directory to overwrite any existing time-series files there. Defaults to `FALSE`.

Details

- Ensure response option names in the `name_responses` script match the column names in the time-series CSV files.
- Time-series CSV filenames should correspond to the question identifiers (e.g., "Q17_exclNA.csv").
- If write_to_timeseries_directory is `TRUE`, then any results output will overwrite any files in the time-series directory supplied for the given question.

Value

A named list of updated time-series data frames, where each data frame includes both the time-series data and the latest survey data.

Author(s)

Tom Pearson

Examples

```
#fps_update_timeseries(
#  tbl_list = s8_livestock_tbl_list,
#  questions = s8_livestock_ts_qs, timeseries_directory = "path/to/timeseries/",
#  survey_year = 2023, write_to_timeseries_directory = FALSE)
```

---

fps_write_dataset *FPS: Render: Write dataset (XLSX)*

Description

This function saves the given dataset (workbook) to a file in the specified output directory. The dataset is only saved once per run regardless of how many sections are being knit. Optionally, old copies of the dataset can be deleted based on a specified time period.

Usage

```
fps_write_dataset(
  out_dir = ".",
  workbook,
  delete = FALSE,
  delete_unit = "hours",
  delete_number = 12,
  date = tolower(format(Sys.Date(), "%d%b%y"))
)
```

Arguments

out_dir	Character. The directory where the dataset should be saved. Defaults to the current directory (". ").
workbook	A Workbook object created with the openxlsx package. This is the dataset to save.
delete	Logical. If TRUE, old copies of the dataset in the output directory will be deleted. Defaults to FALSE.
delete_unit	Character. The time unit for deleting old files (e.g., "hours", "days"). Defaults to "hours".
delete_number	Numeric. The number of time units to determine file age for deletion. Files older than this value will be deleted. Defaults to 12.
date	Character. The date string used to name the output file. Defaults to today's date in the format "%d%b%y" (e.g., "14jan25").

Details

- The Excel file is saved with the name `fps-ghg-dataset-{date}.xlsx`.
- Old files with `.xlsx` extensions in the output directory are deleted if `delete` is `TRUE` and they meet the age criteria.
- File overwriting is enabled for the saved dataset.

Author(s)

Tom Pearson

See Also

saveWorkbook

Examples

```
#fps_write_dataset(workbook = my_workbook, delete = TRUE, delete_unit = "hours", delete_number = 24)
```

---

get_doc *FPS: Narrative: Print the direction of change (doc) associated with two values generated from the analysis results*

Description

This function determines the direction of change (increase, decrease, or no change) for a specified question and response in Farm Practices Survey data, when comparing values between two survey years.

Usage

```
get_doc(
  .data,
  question,
  response_key,
  svy_years,
  past = FALSE,
  abbr = FALSE,
  get_pc = FALSE
)
```

Arguments

.data	A list of data frames, each representing survey question data.
question	A character or numeric value specifying the question to process. Numeric input is automatically converted to "QX" format (e.g. "Q4").
response_key	A character vector of keywords for the desired response(s). Supports regex patterns for filtering responses.
svy_years	A numeric or character vector of length 2 specifying the survey years for comparison (e.g. c(2024, 2023)).
past	Logical. If TRUE, returns past-tense narrative (e.g. "increased").
abbr	Logical. If TRUE, returns abbreviated narrative (e.g. "an increase" instead of "an increase from").
get_pc	Logical. If TRUE, returns the percentage change instead of narrative.

Details

- The comparison is always done from an older date to a newer date.
- Values are rounded using round_half_up before comparison.
- Handles multiple response keys by summing their values unless the question is irregular.

Value

A character string describing the direction of change or the percentage change if get_pc = TRUE.

Author(s)

Tom Pearson

See Also

round_half_up

Examples

```
#data_list <- list(Q4 = data.frame(year = c("2024", "2023"),
#                                  value = c(60, 50),
#                                  response = c("yes", "yes")))
#get_doc(data_list, question = 4, response_key = "yes",
#        svy_years = c(2024, 2023), past = FALSE, abbr = FALSE, get_pc = TRUE)
```

get_name *FPS: Narrative: Print the name/category associated with a value based on its ordered position in the analysis' results.*

Description

This function retrieves the response or percentage corresponding to the specified ordinal position (e.g. the most common response) for a question from a list of survey data, ordering by the proportion of holdings in descending order. You can also exclude specific responses and request the result as a percentage rather than its name.

Usage

```
get_name(
  .data,
  questions,
  svy_year,
  ordinal,
  get_percent = FALSE,
  responses_to_exclude = NULL
)
```

Arguments

`.data`	A list of data frames, each representing survey question data.
`questions`	A character or numeric vector specifying the question(s) to process. Numeric input is automatically converted to "QX" format (e.g. `"Q4"`).
`svy_year`	A character or numeric value specifying the survey year (e.g. `2024`).
`ordinal`	A numeric value specifying the ordinal position (e.g. 1 for the most frequent response).
`get_percent`	Logical. If `TRUE`, returns the percentage of the selected response instead of the response itself.
`responses_to_exclude`	A character vector of responses to exclude from the result. Responses are matched using regular expressions.

Details

- The data are ordered by the proportion of holdings in descending order (pre-rounding).
- The function allows the exclusion of certain responses by their keywords.
- The `ordinal` parameter specifies which position to retrieve (e.g., the most common response or the 2nd most common).
- If `get_percent` is `TRUE`, the percentage of the selected response is returned.

Value

A character string representing either the response or the percentage, depending on `get_percent`.

Author(s)

Tom Pearson

See Also

round_half_up

Examples

```
#data_list <- list(Q4 = data.frame(year = c("2024", "2024", "2024"),
#                                  value = c(50, 30, 20),
#                                  response = c("yes", "no", "maybe")))
#get_name(data_list, questions = 4, svy_year = 2024, ordinal = 1)
#get_name(data_list, questions = 4, svy_year = 2024, ordinal = 2, get_percent = TRUE)
```

`get_percent` *FPS: Narrative: Extract a single percentage from the analysis results*

Description

This function extracts and calculates the percentage value for a given set of questions and response keys from the analysed survey data.

Usage

```
get_percent(.data, questions, response_key, svy_year)
```

Arguments

`.data`	A list of data frames, each representing survey question data.
`questions`	A character or numeric vector specifying the question(s) to process. Numeric input is automatically converted to "QX" format (e.g. `"Q4"`).
`response_key`	A character vector of keywords for the desired response(s). Supports regex patterns for filtering responses.
`svy_year`	A numeric or character string representing the survey year (e.g. `2024`).

Details

- Rounds values using round_half_up before summing to ensure consistency.
- Handles filtering of irregular or multiple-choice data with validation.
- Provides warnings and errors when input parameters are missing or invalid.

Value

A string representing the percentage value(s) for the specified question(s) and response(s). If multiple responses are provided, their values are summed.

Author(s)

Tom Pearson

See Also

round_half_up

Examples

```
#data_list <- list(Q4 = data.frame(year = "2024", value = c(50, 30),
#                                  response = c("yes", "no")))
#get_percent(data_list, questions = 4, response_key = "yes", svy_year = "2024")
```

---

round_number *FPS: Narrative: Round numbers (0 dp) and optionally format*

Description

This function rounds numeric values to the nearest whole number and formats them with optional prefix, suffix, and thousands separator. It uses the `scales::label_number()` function for formatting.

Usage

```
round_number(val, prefix = NULL, suffix = NULL, big.mark = "")
```

Arguments

`val`	A numeric vector of values to be rounded.
`prefix`	A character string to be added before the number (optional).
`suffix`	A character string to be added after the number (optional).
`big.mark`	A character string to separate thousands (optional, default is no mark, "").

Details

- The `val` input must be numeric.
- `prefix`, `suffix`, and `big.mark` are optional formatting parameters to control how the numbers are displayed.
- The function uses label_number to format the output, rounding to the nearest whole number.
- The function uses round_half_up to ensure rounding of halves always rounds up, rather than R's default of down when even.

Value

A character vector with the values rounded to whole numbers and formatted according to the specified options.

Author(s)

Tom Pearson

See Also

label_number, round_half_up

Examples

```
#round_number(12345.67)  # Returns "12346"
#round_number(12345.67, prefix = "£")  # Returns "£12346"
#round_number(12345.67, suffix = " units")  # Returns "12346 units"
#round_number(12345.67, big.mark = ",")  # Returns "12,346"
#round_number(12345.67, prefix = "£", suffix = " units", big.mark = ",")  # Returns "£12,346 units"
```

---

round_percent *FPS: Narrative: Convert proportions to rounded percentages*

Description

This function converts a decimal value (e.g. 0.25) into a percentage (e.g., 25%) by multiplying by 100 and rounding to the nearest whole number. The result is formatted as a percentage.

Usage

```
round_percent(val)
```

Arguments

`val`	A numeric vector of decimal values to be rounded to percentages.

Details

- The input `val` should be numeric and represent decimal values between 0 and 1.
- The function uses label_percent to format the output as percentages, rounding to the nearest whole number.
- The function uses round_half_up to ensure rounding of halves always rounds up, rather than R's default of down when even.

Value

A character vector with the values converted to percentages (e.g. "25%" for 0.25).

Author(s)

Tom Pearson

See Also

label_percent, round_half_up

Examples

```
#round_percent(0.25)  # Returns "25%"
#round_percent(c(0.1345, 0.155, 0.245))  # Returns "13%", "16%", "25%"
```

Index