Package 'tutorial.helpers'

May 8, 2026

Title Helper Functions for Creating Tutorials

Version 0.6.1

Description Helper functions for creating, editing, and testing tutorials created with the 'learnr' package. Provides a simple method for allowing students to download their answers to tutorial questions. For examples of its use, see the 'r4ds.tutorials' package.

Depends R (>= 4.1.0)

License MIT + file LICENSE

Encoding UTF-8

RoxygenNote 7.3.2

VignetteBuilder quarto

Suggests fs, googledrive, httr, knitr, quarto, roxygen2, testthat (>= 3.0.0), tidyverse

Config/testthat/edition 3

Imports dplyr, jsonlite, learnr, mime, purrr, rmarkdown, rstudioapi, rvest, shiny, tibble

BugReports https://github.com/PPBDS/tutorial.helpers/issues

URL https://ppbds.github.io/tutorial.helpers/, https://github.com/PPBDS/tutorial.helpers

NeedsCompilation no

Author David Kane [aut, cre, cph] (ORCID: <https://orcid.org/0000-0002-6660-3934>)

Maintainer David Kane <dave.kane@gmail.com>

Repository CRAN

Date/Publication 2025-12-23 06:10:11 UTC

Contents

`check_current_tutorial` *Check current tutorial*

Description

A function which formats the exercise numbers and code chunk labels correctly. `check_current_tutorial()` reads in the tutorial which is in the active editor window in Positron. It determines what the number of each exercise should be and fixes any mistakes. It ensures that all code chunk labels are the correct function of the section title and exercise number.

Usage

```
check_current_tutorial()
```

check_key_vars *Check Key Variables in List of Tibbles*

Description

This function checks if specified key variables are present in each tibble's "id" column and returns only tibbles that contain all required key variables.

Usage

```
check_key_vars(tibble_list, key_vars, verbose = FALSE)
```

Arguments

tibble_list	A named list of tibbles, each containing an "id" column with question identifiers
key_vars	A character vector of key variables to check for
verbose	A logical value (TRUE or FALSE) specifying verbosity level. If TRUE, reports tibbles that are removed and why.

Value

A list of tibbles that contain all required key variables

Examples

```
## Not run:
# Create sample data
path <- file.path(find.package("tutorial.helpers"), "tests/testthat/fixtures/answers_html")

tibble_list <- gather_submissions(path, "stop")

result <- check_key_vars(tibble_list,
                         key_vars = c("name", "email"))


## End(Not run)
```

check_membership *Check Membership in List of Tibbles*

Description

This function filters a list of tibbles based on whether a key variable's value is among the membership values. It first uses check_key_vars() to ensure the key variable exists, then checks membership. Useful for keeping only specific students or participants.

Usage

```
check_membership(tibble_list, key_var, membership, verbose = FALSE)
```

Arguments

tibble_list	A named list of tibbles, each containing an "id" column and an "answer" column
key_var	A character string specifying the key variable to check
membership	A character vector of allowed values for the key variable
verbose	Logical indicating whether to report removed items (default: FALSE)

Value

A list of tibbles where the key variable exists and its value is in the membership list

Examples

```
## Not run:
# Create sample data with student emails
path <- file.path(find.package("tutorial.helpers"), "tests/testthat/fixtures/answers_html")

tibble_list <- gather_submissions(path, "stop")

result <- check_membership(tibble_list,
                           key_var = "email",
                           membership = c("bluebird.jack.xu@gmail.com",
                                          "hassan.alisoni007@gmail.com"))


## End(Not run)
```

---

check_tutorial_defaults

Confirm that a tutorial has the recommended components

Description

Checks that tutorials contain required libraries and child documents. The function looks for library() calls and child document inclusions in the tutorial files.

Usage

```
check_tutorial_defaults(
  tutorial_paths,
  libraries = c("learnr", "tutorial.helpers"),
  children = c("info_section", "download_answers")
)
```

Arguments

tutorial_paths	Character vector of the paths to the tutorials to be examined.
libraries	Character vector of library names that should be loaded in the tutorial. The function looks for library(name) calls. Default is c("learnr", "tutorial.helpers").
children	Character vector of child document names (without the .Rmd extension) that should be included in the tutorial. The function looks for these in child document inclusion chunks. Default is c("info_section", "download_answers").

Value

No return value, called for side effects.

Examples

```
# Check with default requirements
check_tutorial_defaults(
  tutorial_paths = return_tutorial_paths("tutorial.helpers")
)

# Check for specific libraries only
check_tutorial_defaults(
  tutorial_paths = return_tutorial_paths("tutorial.helpers"),
  libraries = c("learnr", "knitr"),
  children = c("copy_button")
)
```

---

determine_code_chunk_name

Determine the code chunk name of a new exercise in a tutorial.

Description

Determine the code chunk name of a new exercise in a tutorial.

Usage

```
determine_code_chunk_name(file_path = NULL)
```

Arguments

file_path	Character string of the file path to the tutorial

Value

The section id of the exercise based on its section

determine_exercise_number *Finds the number of the next exercise in a tutorial*

Description

Finds the number of the next exercise in a tutorial

Usage

```
determine_exercise_number(file_path = NULL)
```

Arguments

file_path	Character string of the file path to the tutorial

Value

The next exercise number based on the file argument or the active document.

download_google_drive *Download Files from a Google Drive Folder*

Description

Downloads all files or filtered files from a public Google Drive folder to a local directory.

Usage

```
download_google_drive(
  url,
  path = NULL,
  title = NULL,
  overwrite = TRUE,
  verbose = TRUE
)
```

Arguments

url	Character string. The Google Drive folder URL or ID. Supports standard folder URLs (e.g., "https://drive.google.com/drive/folders/FOLDER_ID") or direct folder IDs.
path	Character string or NULL. The local directory path for downloads. If NULL (default), uses the current working directory. If the directory doesn't exist, it will be created.

title	Character vector or NULL. Patterns to match against file names for filtering. If provided, only files whose names contain any of these patterns (case-insensitive) are downloaded. If NULL (default), all files are downloaded.
overwrite	Logical. If TRUE (default), overwrites existing files. If FALSE, skips files that already exist.
verbose	Logical. If TRUE (default), prints detailed progress messages.

Value

Character string. The path to the directory where files were downloaded.

Examples

```
## Not run:
# Download all files to current directory
download_google_drive("https://drive.google.com/drive/folders/1Rgxfiw")

# Download to a specific directory with filtering
download_google_drive(
  url = "https://drive.google.com/drive/folders/1Rgxfiw",
  path = "./my_data",
  title = c("report", ".csv"),
  overwrite = FALSE
)

## End(Not run)
```

---

format_tutorial *Format RMarkdown tutorial code chunks*

Description

This function processes an R Markdown tutorial file to standardize code chunk labels based on section names and exercise numbers. It also renumbers exercises sequentially within each section and fixes spacing in topic headers.

Usage

```
format_tutorial(file_path)
```

Arguments

file_path	Character string. Path to the R Markdown file to process.

Details

The function applies the following formatting rules:

- Topic headers (# headers) have their spacing standardized
- Exercises are renumbered sequentially within each section
- Code chunks are relabeled according to the pattern: section-name-exercise-number
- Chunks with `eval = FALSE` receive a `-hint-N` suffix
- Chunks with `include = FALSE` receive a `-test` suffix
- Chunks with label "setup" are not modified
- Chunks with the "file" option are not modified
- Unlabeled chunks without key options are not modified
- All formatted chunks preserve their original options
- Content between quadruple backticks ( ) is preserved untouched

Value

Character string containing the formatted R Markdown content.

`gather_submissions` *Gather Submissions*

Description

This function finds and reads HTML/XML files from a local directory or Google Drive folder that match specified patterns. It extracts tables from the files and returns a list of tibbles containing the submission data.

Usage

```
gather_submissions(path, title, keep_loc = NULL, verbose = FALSE)
```

Arguments

`path`	The path to the local directory containing the HTML/XML files, or a Google Drive folder URL. If it's a Google Drive URL, the function will download the entire folder to a temporary directory.
`title`	A character vector of patterns to match against the file names. Each pattern is processed separately and results are combined.
`keep_loc`	A character string specifying where to save downloaded files (only for Google Drive URLs). If NULL (default), files are downloaded to a temporary directory and deleted after processing. If specified, files are downloaded to this location and kept.
`verbose`	A logical value (TRUE or FALSE) specifying verbosity level. If TRUE, reports files that are removed during processing.

Details

Google Drive allows for more than one file with the exact same name. If you download files manually ("by hand"), you will get both files but with one of them automatically renamed by your browser. However, if you use the Google Drive functionality in this function, the second file will overwrite the first, potentially resulting in data loss.

Value

A named list of tibbles, where each tibble contains the data from one HTML/XML file that matches any of the specified patterns and has valid table structure.

Examples

```
## Not run:
# Find submissions from local directory

path <- file.path(find.package("tutorial.helpers"), "tests/testthat/fixtures/answers_html")

tibble_list <- gather_submissions(path = path, title = "stop", verbose = TRUE)

# Find submissions from Google Drive folder (temporary download)
drive_url <- "https://drive.google.com/drive/folders/10do12t0fZsfrIrKePxwjpH8IqBNVO86N"
tibble_list <- gather_submissions(
  path = drive_url,
  title = c("positron")
)

# Find submissions from Google Drive folder (keep files)
tibble_list <- gather_submissions(
  path = drive_url,
  title = c("introduction"),
  keep_loc = "temp_file"
)

## End(Not run)
```

---

get_submissions_from_learnr_session

Return a list of tutorial answers

Description

Grabs information from the `learnr` session environment, not directly from the session object itself. Since we are using the session environment, we currently don't (?) have a way to save the environment and hence can't test this function.

Usage

```
get_submissions_from_learnr_session(sess)
```

Arguments

sess	session object from shiny with learnr

Value

a list which includes the exercise submissions of tutorial

---

knit_tutorials *Knit a set of tutorials*

Description

We define "testing" a tutorial as (successfully) running `render()` on it. This function renders all the tutorials provided in `tutorial_paths`. There is no check to see if the rendered file looks OK. If a tutorial fails to render, then an error will be generated which will propagate to the caller.

Usage

```
knit_tutorials(tutorial_paths)
```

Arguments

tutorial_paths	Character vector of the paths to the tutorials to be knitted.

Value

No return value, called for side effects.

Examples

```
## Not run:
  knit_tutorials(tutorial_paths = return_tutorial_paths("tutorial.helpers"))

## End(Not run)
```

make_exercise *Add a tutorial code exercise or question to the active document*

Description

When writing tutorials, it is handy to be able to insert the skeleton for a new code exercise or question. Note that the function determines the correct exercise number to use and also adds appropriate code chunk labels, based on the exercise number and section title.

Usage

```
make_exercise(type = "no-answer", file_path = NULL)
```

Arguments

type	Character of question type. Must be one of "code", "no-answer", or "yes-answer". Abbreviations such as "no", "yes", and "co" are allowed.
file_path	Character path to a file. If NULL, the RStudio active document is used, which is the default behavior. An actual file path is used for testing.

Value

Exercise skeleton corresponding to the type argument.

match_questions *Match Questions by Pattern*

Description

This function takes a single HTML file or tibble and finds all questions/answers that contain a specified pattern. It returns the question IDs (from the 'id' column) for rows where the answer contains the pattern.

Usage

```
match_questions(x, pattern, ignore.case = TRUE)
```

Arguments

x	Either a file path to an HTML file or a tibble with 'id' and 'answer'/'data' columns
pattern	A character string to search for in the answers
ignore.case	Logical; should the search be case-insensitive? (default: TRUE)

Value

A character vector of question IDs where the answer contains the pattern

Examples

```
## Not run:
# Search in an HTML file
question_ids <- match_questions("path/to/submission.html", "temperance")
# Returns: c("temperance-16", "temperance-19")

# Search in a tibble

path <- file.path(find.package("tutorial.helpers"), "tests/testthat/fixtures/answers_html")

tibble <- gather_submissions(path, title = "stop")[[1]]

result <- match_questions(tibble, "http")

## End(Not run)
```

---

open_github_pages *Open Multiple GitHub Pages in Browser Tabs*

Description

This function opens multiple GitHub.io pages in browser tabs, making it easy for teaching fellows to quickly review student webpages. It can accept either a vector of URLs or a tibble/data.frame containing URLs.

Usage

```
open_github_pages(
  urls,
  url_var = NULL,
  label_var = NULL,
  delay_seconds = 0.5,
  browser = "default",
  verbose = FALSE
)
```

Arguments

urls	Either a character vector of URLs to open, OR a tibble/data.frame containing submission data with a URL column
url_var	Character string specifying the column name containing URLs. Required when urls is a tibble/data.frame. Ignored when urls is a character vector.

label_var	Character string specifying the column name to use for identifying each submission in verbose output (e.g., "name", "email"). Only used when urls is a tibble/data.frame.
delay_seconds	Numeric value specifying delay between opening each URL (default is 0.5 seconds to allow browser to process each request)
browser	Character string specifying which browser to use. Options are "default" (system default), "chrome", "firefox", "safari", or "edge". On Windows, also supports "msedge". Default is "default".
verbose	Logical value (TRUE or FALSE) specifying verbosity level. If TRUE, reports each URL as it's being opened.

Details

The function uses the system's default method to open URLs, which typically opens them in the default browser. Most modern browsers will open multiple URLs as tabs in the same window when called in quick succession.

The delay between opening URLs helps ensure the browser has time to process each request properly. You may need to adjust this delay based on your system performance and browser behavior.

Value

Invisible NULL. Function is called for its side effect of opening browser tabs.

Examples

```
## Not run:
# Open multiple GitHub Pages from vector
student_sites <- c("https://github.com/Abdul-Hannan96/stops.git")

open_github_pages(student_sites, verbose = TRUE)

# Open from tibble/data.frame
path <- file.path(find.package("tutorial.helpers"), "tests/testthat/fixtures/answers_html")

result <- submissions_answers(
  path = path,
  title = c("stop"),
  key_var = "email",
  membership = c("bluebird.jack.xu@gmail.com", "abdul.hannan20008@gmail.com"),
  vars = c("name","email","temperance-15"),
  verbose = TRUE
)

open_github_pages(result,
                  url_var = "temperance-15",
                  verbose = TRUE)

## End(Not run)
```

`return_tutorial_paths` *Return all the paths to the tutorials in a package*

Description

Takes a package name and returns a character vector of all the paths to tutorials in the installed package. This function looks for all R Markdown files (.Rmd) in the inst/tutorials/ subdirectories of the specified package. It uses learnr::available_tutorials() to identify tutorial directories, with a fallback to directory scanning if that fails.

Usage

```
return_tutorial_paths(package)
```

Arguments

`package`	Character string of the package name to be tested.

Details

The function first checks if the package is installed and has a tutorials directory. It then attempts to use learnr::available_tutorials() to get the official list of tutorial directories. If that fails (e.g., if the package doesn't properly register its tutorials with learnr), it falls back to scanning all subdirectories under inst/tutorials/. Finally, it collects all .Rmd files from these directories.

Returns an empty character vector if the package has no tutorials or doesn't exist, rather than throwing an error.

Value

Character vector of the full paths to all installed tutorials in `package`. Returns character(0) if no tutorials are found or if the package doesn't exist.

Examples

```
## Not run:
  # Get all learnr tutorial paths
  return_tutorial_paths('learnr')

  # Get tutorial paths from your own package
  return_tutorial_paths('tutorial.helpers')

  # Returns empty vector for packages without tutorials
  return_tutorial_paths('base')

## End(Not run)
```

`set_positron_settings` *Configure Positron Settings*

Description

Locates or creates the Positron `settings.json` file on Windows or macOS, then updates those settings based on the provided configuration list. Users can specify settings like RStudio keyboard shortcuts. The function can also optionally configure R profile settings including binary package preferences and download timeout. Positron is an R-focused Integrated Development Environment (IDE) based on Visual Studio Code, designed to enhance the R programming experience with a modern interface and features.

Usage

```
set_positron_settings(
  home_dir = path.expand("~"),
  set.rprofile = TRUE,
 positron_settings = list(editor.wordWrap = "on", workbench.startupEditor = "none",
    files.defaultLanguage = "r", workbench.editor.enablePreview = FALSE,
   terminal.integrated.defaultProfile.windows = "Git Bash", git.enableSmartCommit =
    TRUE, git.confirmSync = FALSE)
)
```

Arguments

`home_dir`	Optional character string specifying the base directory to use as the user's home directory. Defaults to `path.expand("~")`. Useful for testing or custom setups.
`set.rprofile`	Logical, defaults to `TRUE`. If `TRUE`, runs `set_rprofile_settings()` after applying settings to configure binary package installation and download timeout in the R profile.
`positron_settings`	List of settings to apply. Can be structured as a list of lists where each sub-list contains a setting name and value (e.g., `list(list("rstudio.keymap.enable", TRUE))`), or as a named list (e.g., `list("rstudio.keymap.enable" = TRUE)`). Defaults to a named list with seven settings: `list("editor.wordWrap" = "on", "workbench.startupEditor" = "none", "files.defaultLanguage" = "r", "workbench.editor.er = FALSE, "terminal.integrated.defaultProfile.windows" = "Git Bash", "git.enableSmartCommit" = TRUE, "git.confirmSync" = FALSE)`, which enables word wrap for improved code readability, disables startup editor for a clean interface, sets R as the default language for new files, disables preview mode for predictable tab behavior, sets Git Bash as the default terminal on Windows, enables auto-staging of Git changes, and disables confirmation dialogs for Git push/pull, respectively.

Details

This function uses the `jsonlite` package to handle JSON operations and creates the necessary directory structure if it doesn't exist. It is designed to work cross-platform by detecting the operating system and constructing the appropriate file path to Positron's user settings. The function applies the settings provided in the `positron_settings` parameter. If the `positron_settings` list is empty, no changes are made to the `settings.json` file. By default, seven settings are applied unless overridden: enabling word wrap for better code readability, disabling startup editor for a clean interface, setting R as the default language for new files, disabling preview mode for predictable tab behavior, setting Git Bash as the default terminal on Windows, enabling Git smart commit, and disabling Git sync confirmation dialogs.

Note: Windows file paths in settings should use forward slashes (/) or escaped backslashes (\\). The function will automatically handle path normalization for Windows.

Value

Invisible `NULL`. The function's purpose is its side effect: modifying or creating the `settings.json` file. It also prints messages to the console indicating actions taken.

Examples

```
## Not run:
  # Apply default settings (word wrap, clean startup, R language,
  # no preview, Git Bash terminal, smart commit, no sync confirmation)
  set_positron_settings()

  # Enable RStudio keyboard shortcuts using list of lists structure
  set_positron_settings(
    positron_settings = list(list("rstudio.keymap.enable", TRUE))
  )

  # Enable RStudio keyboard shortcuts using named list structure
  set_positron_settings(
    positron_settings = list("rstudio.keymap.enable" = TRUE)
  )

  # Apply multiple settings using named list
  set_positron_settings(
    positron_settings = list(
      "rstudio.keymap.enable" = TRUE,
      "editor.wordWrap" = "on"
    )
  )

  # Set a Windows file path (use forward slashes)
  set_positron_settings(
    positron_settings = list(
      "files.dialog.defaultPath" = "C:/Users/username/projects"
    )
  )

  # Apply settings without modifying .Rprofile
```

```
  set_positron_settings(
    set.rprofile = FALSE,
    positron_settings = list("rstudio.keymap.enable" = TRUE)
  )

  # Handle case where settings directory does not exist
  set_positron_settings(
    home_dir = tempfile(),  # Simulate a non-existent directory
    positron_settings = list("rstudio.keymap.enable" = TRUE)
  )

  # Handle case with invalid JSON file
  # Create an invalid JSON file for testing
  dir.create(file.path(tempdir(), "Positron", "User"), recursive = TRUE)
  writeLines("invalid json", file.path(tempdir(), "Positron", "User", "settings.json"))
  set_positron_settings(
    home_dir = tempdir(),
    positron_settings = list("rstudio.keymap.enable" = TRUE)
  )

## End(Not run)
```

`set_rprofile_settings` *Configure R Profile Settings for Better User Experience*

Description

Configures important settings in your `.Rprofile` to improve the R experience, especially for new users. This includes:

- Setting package installation to use binaries (non-Linux systems)
- Increasing the timeout for downloads to prevent installation failures

These settings help avoid common issues like source compilation failures on Windows and timeout errors when downloading large packages.

You can examine your `.Rprofile` to confirm changes with `usethis::edit_r_profile()`

Usage

```
set_rprofile_settings(set_for_session = TRUE, backup = TRUE)
```

Arguments

`set_for_session`	Logical, defaults to `TRUE`. If `TRUE`, also applies these settings to the current R session.
`backup`	Logical, defaults to `TRUE`. If `TRUE`, creates a backup of existing `.Rprofile` before modifying it.

Value

Invisible NULL. Called for side effects.

Examples

```
## Not run:
  # Apply settings to .Rprofile and current session
  set_rprofile_settings()

  # Only modify .Rprofile, don't change current session
  set_rprofile_settings(set_for_session = FALSE)

  # Modify without creating backup
  set_rprofile_settings(backup = FALSE)

## End(Not run)
```

---

show_file *Display the contents of a text file that match a pattern*

Description

This function reads the contents of a text file and either prints the specified range of rows that match a given regular expression pattern, prints the code lines within R code chunks, or extracts the YAML header. If start is a negative number, it prints the last abs(start) lines, ignoring missing lines at the end of the file. If start is 0, it prints the entire file.

Usage

```
show_file(path, start = 1, end = NULL, pattern = NULL, chunk = "None")
```

Arguments

path	A character vector representing the path to the text file.
start	An integer specifying the starting row number (inclusive) to consider. Default is 1. If negative, it represents the number of lines to print from the end of the file. If 0, prints the entire file.
end	An integer specifying the ending row number (inclusive) to consider. Default is the last row.
pattern	A regular expression pattern to match against each row. Default is NULL (no pattern matching).
chunk	A character string indicating what content to extract. Possible values are "None" (default - no chunk processing), "All" (print all R code chunks), "Last" (print only the last R code chunk), or "YAML" (extract the YAML header without delimiters).

Value

The function prints the contents of the specified range of rows that match the pattern (if provided), the code lines within R code chunks (if chunk is "All" or "Last"), or the YAML header content (if chunk is "YAML") to the console. If no rows match the pattern, nothing is printed. If start is negative, the function prints the last abs(start) lines, ignoring missing lines at the end of the file. If start is 0, the function prints the entire file.

Examples

```
## Not run:
# Display all rows of a text file
show_file("path/to/your/file.txt")

# Display the entire file
show_file("path/to/your/file.txt", start = 0)

# Display rows 5 to 10 of a text file
show_file("path/to/your/file.txt", start = 5, end = 10)

# Display all rows of a text file that contain the word "example"
show_file("path/to/your/file.txt", pattern = "example")

# Print all code lines within R code chunks
show_file("path/to/your/file.txt", chunk = "All")

# Print only the last R code chunk
show_file("path/to/your/file.txt", chunk = "Last")

# Extract the YAML header
show_file("path/to/your/file.Rmd", chunk = "YAML")

# Display the last 5 lines of a text file, ignoring missing lines at the end
show_file("path/to/your/file.txt", start = -5)

## End(Not run)
```

---

submissions_answers *Extract Answers from Submissions with Filtering*

Description

This function gathers submissions matching a title pattern, filters them by membership, and extracts specified variables, returning a tibble with one row per valid submission and one column per variable.

Usage

```
submissions_answers(
  path,
  title,
  key_var = NULL,
  membership = NULL,
  vars,
  keep_file_name = NULL,
  verbose = FALSE
)
```

Arguments

path	The path to the local directory or Google Drive folder URL containing submissions
title	A character vector of patterns to match against file names (passed to gather_submissions)
key_var	A character string specifying the key variable to check for membership (e.g., "email"). If NULL (default), no membership filtering is applied.
membership	A character vector of allowed values for the key variable, or "*" to include all submissions. If NULL (default), no membership filtering is applied. Ignored if key_var is NULL.
vars	A character vector of variables/questions to extract, or "*" to extract all available variables
keep_file_name	How to handle file names: NULL (don't include), "All" (full name), "Space" (up to first space), "Underscore" (up to first underscore)
verbose	A logical value (TRUE or FALSE) specifying verbosity level. If TRUE, reports files that are removed during processing.

Value

A tibble with one row per valid submission, columns for each variable, and optionally a "source" column

Examples

```
## Not run:
# Extract specific variables from submissions matching title pattern
path <- file.path(find.package("tutorial.helpers"), "tests/testthat/fixtures/answers_html")

result <- submissions_answers(
  path = path,
  title = c("stop"),
  key_var = "email",
  membership = c("bluebird.jack.xu@gmail.com", "abdul.hannan20008@gmail.com"),
  vars = c("name", "email", "introduction-1"),
  verbose = TRUE
)
```

```
# Extract all variables from submissions
result_all <- submissions_answers(
  path = path,
  title = c("stop"),
  key_var = "email",
  membership = c("bluebird.jack.xu@gmail.com", "abdul.hannan20008@gmail.com"),
  vars = "*",
  verbose = TRUE
)

drive_url <- "https://drive.google.com/drive/folders/10do12t0fZsfrIrKePxwjpH8IqBNVO86N"
x <- submissions_answers(
  path = drive_url,
  title = c("introduction"),
  key_var = "email",
  membership = c("fmehmud325@gmail.com"),
  vars = c("email", "name", "what-you-will-learn-15")
)

## End(Not run)
```

`submissions_summary` *Process Submissions Summary*

Description

This function processes submissions from a local directory or Google Drive folder containing HTML/XML files. It extracts tables from the files, filters them based on a pattern and key variables, and returns either a summary tibble or a combined tibble with all the data.

Usage

```
submissions_summary(
  path,
  title = ".",
  return_value = "Summary",
  vars = NULL,
  verbose = FALSE,
  keep_file_name = NULL,
  emails = NULL
)
```

Arguments

`path`	The path to the local directory containing the HTML/XML files, or a Google Drive folder URL. If it's a Google Drive URL, the function will download individual files to a temporary directory.
`title`	A character vector of patterns to match against the file names (default: "."). Each pattern is processed separately and results are combined.

return_value	The type of value to return. Allowed values are "Summary" (default) or "All".
vars	A character vector of key variables to extract from the "id" column (default: NULL).
verbose	A logical value (TRUE or FALSE) specifying verbosity level. If TRUE, reports files that are removed during processing.
keep_file_name	Specifies whether to keep the file name in the summary tibble. Allowed values are NULL (default), "All" (keep entire file name), "Space" (keep up to first space), or "Underscore" (keep up to first underscore). Only used when return_value is "Summary".
emails	A character vector of email addresses to filter results by, "*" to include all emails, or NULL to skip email filtering (default: NULL).

Value

If return_value is "Summary", returns a tibble with one row for each file, columns corresponding to the vars, and an additional "answers" column indicating the number of rows in each tibble. If return_value is "All", returns a tibble with all the data combined from all the files.

Examples

```
## Not run:
# Process submissions from local directory
path <- file.path(find.package("tutorial.helpers"), "tests/testthat/fixtures/answers_html")

result <- submissions_summary(path = path,
                              vars = "email",
                              title = "stop")

drive_url <- "https://drive.google.com/drive/folders/10do12t0fZsfrIrKePxwjpH8IqBNVO86N"
x <- submissions_summary(
  path = drive_url,
  title = c("positron"),
  vars = c("email", "name")
)



## End(Not run)
```

---

submission_server *Tutorial submission functions*

Description

Provides the core Shiny server and UI hooks for collecting and downloading student answers from a learnr tutorial. submission_server() should be called in an Rmd code chunk with context="server".

This function was modified from Colin Rundel's learnrhash package (https://github.com/rundel/learnrhash).

UI block to include a download button and simple instructions for students.

Usage

```
submission_server()

submission_ui
```

Format

An object of class `shiny.tag` of length 3.

Details

The server function uses a Shiny downloadHandler to let students download their answers. All main logic must be wrapped in `local()` with `parent.frame()` to ensure access to the live learnr session and objects created in the parent environment.

The `session` object (created by Shiny) is only available inside the `downloadHandler$content` function, so any test-case extraction or answer writing must happen there.

For reference: the `file` argument in `content` is a temporary file path created by Shiny, and your handler's job is to write the downloadable file there.

If you want to generate test fixtures, insert `browser()` inside the `content` function, then use functions like `get_submissions_from_learnr_session(session)` at the prompt.

See also: https://mastering-shiny.org/action-transfer.html#downloading-reports

Value

No return value; called for side effects in a Shiny/learnr session.

An object of class shiny.tag

Examples

```
if(interactive()){
  submission_server()
}

if(interactive()){
  submission_ui
}
```

`write_answers` *Write tutorial answers to file*

Description

Take a tutorial session (or a submission list), extract all submitted answers, and write out an HTML file with those answers.

Usage

```
write_answers(file, obj)
```

Arguments

file	Output file path (should end in .html).
obj	Either a Shiny session object (from learnr) or a list of submissions (as returned by get_submissions_from_learnr_session()).

Index