Package 'tutorial.helpers'

May 21, 2023

Title Helper Functions for Creating Tutorials

Version 0.2.5

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' and 'all.primer.tutorials' packages.

Depends R (>= 4.1.0)

License MIT + file LICENSE

Encoding UTF-8

RoxygenNote 7.2.3

VignetteBuilder knitr

Suggests checkmate, knitr, roxygen2, rsconnect, rvest, testthat (>= 3.0.0), tidyverse

Config/testthat/edition 3

Imports dplyr, grDevices, gridExtra, learnr, parsermd, purrr, readr, rmarkdown, rstudioapi, shiny, stringr, tibble

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

URL https://ppbds.github.io/tutorial.helpers/

NeedsCompilation no

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

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

Repository CRAN

Date/Publication 2023-05-21 07:10:02 UTC

R topics documented:

`check_current_tutorial` *Check current tutorial*

Description

An add-in for formatting tutorials.

Uses format_tutorial() to format the tutorial Rmd open in the current editor

Usage

```
check_current_tutorial()
```

`check_tutorial_defaults` *Confirm that a tutorial has the recommended components*

Description

There are three code components: the use of a copy-code button, an information request, and a download page. It is tricky to know where to store the "truth" of what these components should look like. For now, the truth is defined as the `skeleton.Rmd` which defines the template for creating a new tutorial.

All tutorials should also have `library(learnr)` and `library(tutorial.helpers)`, both of which exist in the skeleton

Usage

```
check_tutorial_defaults(tutorial_paths)
```

Arguments

`tutorial_paths` Character vector of the paths to the tutorials to be examined.

Value

No return value, called for side effects.

Examples

```
check_tutorial_defaults(tutorial_paths = return_tutorial_paths("tutorial.helpers"))
```

`format_tutorial` *Re-format a tutorial*

Description

A function for formatting tutorial Rmd files. Used by check_current_tutorial() to re-format the currently open tutorial in RStudio. It renumbers the exercises so that they are in order. It ensures that chunk labels use this numbering, along with the section title.

Usage

```
format_tutorial(file_path)
```

Arguments

`file_path`	Character string.

Value

Formatted document with correct code and hint chunk labels.

`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 (we assume!) an error will be generated which will then filter up to our testing rig.

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

```
knit_tutorials(tutorial_paths = return_tutorial_paths("tutorial.helpers"))
```

---

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. We bind `make_exercise()` and friends as an RStudio add-in to provide this functionality. Note that the function determines the correct exercise number to use and also adds appropriate code chunk names, based on the exercise number and section title.

Usage

```
make_exercise(type = "code")

make_no_answer()

make_yes_answer()
```

Arguments

type	Character of question type. Must be one of "code", "no-answer", or "yes-answer".

Details

It appears that the RStudio addins must have function names only as the Binding value. In other words, you can't have `make_exercise(type = 'no-answer')` as the value. So, we need two extra functions — `make_no_answer()` and `make_yes_answer()` —which just call `make_exercise()` while passing in the correct argument.

Value

Exercise skeleton corresponding to the `type` argument.

`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. Assumes that every Rmd file in inst/tutorials/*/ is a tutorial, which should be true.

Usage

```
return_tutorial_paths(package)
```

Arguments

package	Character vector of the package name to be tested.

Value

Character vector of the full paths to all installed tutorials in `package`.

Examples

```
return_tutorial_paths('learnr')
```

set_binary_only_in_r_profile

Set pkgType to binary in .Rprofile

Description

This functions sets the pkgType global option to "binary" in your .Rprofile. New R users, especially those on Windows, should never install from source. Doing so fails too often, and too confusingly. It also sets the value for this R session. So, you do not need to either restart R nor source the .Rprofile by hand.

You can examine your .Rprofile to confirm this change with usethis::edit_r_profile()

Usage

```
set_binary_only_in_r_profile()
```

Value

No return value, called for side effects.

set_rstudio_settings

Select smart setting for RStudio

Description

This functions selects RStudio settings which make learning easier for new users. These settings are stored in: ~/.config/rstudio/rstudio-prefs.json. The most important changes are save_workspace to "never", load_workspace to FALSE, and "insert_native_pipe_operator" to TRUE.

Usage

```
set_rstudio_settings()
```

Value

No return value, called for side effects.

submission_server *Tutorial submission functions*

Description

The following function was modified from Colin Rundel's learnrhash package, available at https://github.com/rundel/learnrha
Note that when including these functions in a learnr Rmd document it is necessary that the server function, `submission_server()`, be included in an R chunk where `context="server"`.

Usage

```
submission_server(session)

submission_ui
```

Arguments

`session`	Session object from `Shiny` with `learnr`.

Format

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

Value

No return value, called for side effects.

An object of class shiny.tag.

Examples

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

if(interactive()){
  submision_ui
}
```

`write_answers` *Write tutorial answers to file*

Description

Take a tutorial session, extract out all the submitted answers, and write out a file — either as html, rds or pdf — with all of those answers.

Usage

```
write_answers(file, session, is_test = FALSE)
```

Arguments

`file`	Location to render answers to. Output file type determined by file suffix. Acceptable values are "html", "rds" and "pdf".
`session`	Session object from `Shiny` with `learnr`.
`is_test`	`TRUE`/`FALSE` depending on whether or not we are just testing the function. Default is `TRUE`.

Details

We only keep track of the questions/exercises that the student has completed. So, if she only answers three questions, the resulting output will only have 6 rows (the three answers plus the header row plus the first row with tutorial info plus the last row with the time taken). The other obvious approach is to keep all the questions/exercises and leave unanswered ones as NA. Not sure if that approach is better, or even possible.

Examples

```
if(interactive()){
  write_answers("outfile.pdf", sess)
}
```

Index