Unit test examples

Zoe Vernon

9/28/2020

Tests that output is as expected given correct input:

```
test_that("getCitationAndID returns a researcher with a matching name", {
  SA_html_file <- getCitationAndID('Susan Athey')
  SA_url <- paste("https://scholar.google.com/citations?user=",
                  SA_html_file$Google_ID,
                  sep = "")

  expect_true(grepl('Susan Athey', readLines(url(SA_url), n = 1)))
})

test_that("getPaperCitations returns a table with expected properties", {
  SA_html_file <- getCitationAndID('Susan Athey')
  SA_table <- getPaperCitations(SA_html_file)

  expect_length(SA_table[1,], 5)
  expect_s3_class(SA_table, 'data.frame')
  expect_s3_class(SA_table$art_title, 'factor')
  expect_s3_class(SA_table$authors, 'factor')
  expect_s3_class(SA_table$journal, 'factor')
  expect_type(SA_table$year_pub, 'integer')
  expect_type(SA_table$num_cites, 'double')
})

test_that("citationTable input should be fine", {
  scholar = "Jim Pitman"
  expect_type(scholarHTML(scholar), 'list')
  expect_length(scholarHTML(scholar), 2)
  expect_equal(names(scholarHTML(scholar)), c('id', ' citationHTML'))

  scholarJen = "Jennifer Chayes"
  expect_type(scholarHTML(scholarJen), 'list')
  expect_length(scholarHTML(scholarJen), 2)
  expect_equal(names(scholarHTML(scholar)), c('id', 'citationHTML'))
})

# tests that the citationTable function works
test_that("citationTable should work as intended", {
  scholar = "Jim Pitman"
  expect_type(citationTable(scholarHTML(scholar)), 'list')
  expect_length(head(citationTable(scholarHTML(scholar))), 5)

  scholarJen = "Jennifer Chayes"
```

```
  expect_type(citationTable(scholarHTML(scholarJen)), 'list')
  expect_length(head(citationTable(scholarHTML(scholarJen))), 5)
})

test_that("Search works with name input ", {
  name = "Jennifer Chayes"
  link = citation_page_f(name)
  citation = citation_data_base_all_f(name)
  # test that we have a real link to the citation page
  expect_type(link, "character")
  # test that we return more than 20 citations
  expect_gt(length(citation[[1]]), 20)
})

test_that("citationpage works with correct first and last name", {
  ## The HTML of citation page after entering "Jennifer Chayes" as researcher name
  researchername <- "Jennifer Chayes"
  url_citation <-
    "https://scholar.google.com/citations?user=YAHWbtkAAAAJ&hl=en&oi=ao"
  html_citation <- read_html(url_citation)

  ## Check if the function citationpage returns what we expect
  expect_equal(citationpage(researchername), html_citation)
  expect_length(citationpage(researchername), length(html_citation))
  expect_type(citationpage(researchername), 'list')
})

test_that("get_publications works with normal input", {
  # Normal input
  name <- "Eduardo Miranda"

  # Get output
  df <- get_publications(name)

  # The dataframe should have 5 columns
  expect_length(df, 5)

  # The dataframe should have 100 rows for this author
  expect_true(nrow(df) == 100)

  # The columns should have the appropriate data types
  dtypes <- c("character", "character", "character",
              "numeric",   "numeric")
  expect_true(prod(sapply(df, class) ==  dtypes) == 1)
})
```

Tests that assertions are working correctly:

```
# tests if there is an error from an incomplete name
test_that("scholarHTML returns error because of incomplete name", {
  expect_error(scholarHTML("Jim"), "needs to only have a First and Last Name")
})

# tests if there is an error from an erraneous name
```

```r
test_that("scholarHTML returns error because there is no results in Google", {
  expect_error(scholarHTML("asdasd pwefwe"), "did not return any results")
})


test_that("search fail if name is not standard character", {
  name = "jç!(;"
  expect_error(citation_data_base_all_f(name), "name not standard")
})

test_that("searcher doesn't exist" , {
  name = "Jason Nyam"
  expect_error(citation_page_f(name), "researcher don't have any citation")
})

## This is a test for checking whether input is a character string
## and return the error message if not
test_that("citationpage does not work if input is not a character string", {
  researchername <- 3.14
  expect_error(citationpage(researchername), "Please provide a valid name")
})

## This is a test for detecting numbers in the input character string
## and return the error message if not
test_that("citationpage does not work if name has numbers", {
  researchername <- "Jennifer Chayes1"
  expect_error(citationpage(researchername),
               "Enter a valid name with no digits")
})

## This is a test for detecting special characters in the input character string
## and return the error message if not
test_that("citationpage does not work if name has special characters", {
  researchername <- "/Jennifer Chayes"
  expect_error(citationpage(researchername),
               "Enter a valid name with no special characters")
})

## This is a test for checking if input has both first and last name
## and return the error message if not
test_that("citationpage does not work if name only has first name", {
  researchername <- "Jennifer"
  expect_error(citationpage(researchername),
               "Enter the first name and last name")
})

## Check if citationinfo can return an error message when input is character
test_that("citationinfo does not work if input is character", {
  article_html <- "this is html"
  expect_error(citationinfo(article_html),
               "Please provide a valid HTML with type 'list'")
})
```

```r
## Check if citationinfo can return an error message when input is numeric
test_that("citationinfo does not work if input is numeric", {
  article_html <- 3.1415
  expect_error(citationinfo(article_html),
               "Please provide a valid HTML with type 'list'")
})

test_that("get_publications fails with bad input", {
  # Bad input
  name <- "Bad Input"

  # Get output
  expect_error(get_publications(name))
})
```