# Package 'swirl'

August 26, 2014

**Title** Learn R, in R.

**Description** swirl turns the R console into an interactive learning environment. Users receive immediate feedback as they are guided through self-paced lessons in data science and R programming.

**URL** http://swirlstats.com

**Version** 2.2.16

**License** GPL-3

**Depends** R (>= 3.0.2)

**Imports** stringr, testthat, httr, yaml, RCurl, digest, tools

**Author** Nick Carchedi [aut, cre],Bill Bauer [aut],Gina Grdina [aut],Sean Kross [aut]

**Maintainer** Nick Carchedi <nick.carchedi@gmail.com>

**NeedsCompilation** no

**Repository** CRAN

**Date/Publication** 2014-08-26 22:57:01

## R topics documented:


AnswerTests . . . . . 2
any_of_exprs . . . . . 4
bye . . . . . 5
email_admin . . . . . 5
expr_creates_var . . . . . 6
expr_identical_to . . . . . 6
expr_is_a . . . . . 7
expr_uses_func . . . . . 8
func_of_newvar_equals . . . . . 9
info . . . . . 9
InstallCourses . . . . . 10

install_course_directory . . . 11
install_course_dropbox . . . 11
install_course_github . . . 12
install_course_google_drive . . . 13
install_course_url . . . 13
install_course_zip . . . 14
install_from_swirl . . . 15
main . . . 16
nxt . . . 16
omnitest . . . 17
play . . . 18
reset . . . 19
skip . . . 19
submit . . . 20
swirl . . . 20
uninstall_course . . . 21
val_has_length . . . 22
val_matches . . . 23
var_is_a . . . 23
zip_course . . . 24

**Index** **26**

---

`AnswerTests` *Answer Tests*

---

## Description

Answer tests are how swirl determines whether a user has answered a question correctly or not. Each question has one or more answer tests associated with it, all of which must be satisfied in order for a user's response to be considered correct. As the instructor, you can specify any combination of our predefined answer tests or create your own custom answer tests to suit your specific needs. This document will explain your options.

## Details

For each question that you author as part of a swirl lesson, you must specify exactly one *correct answer*. This is separate and distinct from the answer tests. This does not have to be the only correct answer, but it must answer the question correctly. If a user `skip`s your question, this is the answer that will be entered on his or her behalf.

If you're using the swirlify authoring tool, the correct answer will be automatically translated into the appropriate answer test for most question types. Questions that require the user to enter a valid command at the R prompt (which we'll refer to as *command questions*) are the only exception. Since there are often many possible ways to answer a command question, you must determine how you'd like swirl to assess the correctness of a user's response. This is where answer tests come in.

You can specify any number of answer tests. If you use more than one, you must separate them with semicolons. If you do not specify any answer tests for a command question, then the default

test will be used. The default test is `omnitest(correctExpr='<correct_answer_here>')`, which will simply check that the user's expression matches the expression that you provided as a correct answer.

In many cases, the default answer test will provide sufficient vetting of a user's response to a command question. While it is somewhat restrictive in the sense that it requires an exact match of expressions (ignoring whitespace), it is liberating to the course author for two reasons.

1. It allows for fast prototyping of content. As you're developing content, you may find that determining how to test for correctness distracts you from the message you're trying to communicate.
2. You don't have to worry about what happens if the user enters an incorrect response, but is allowed to proceed because of an oversight in the answer tests. Since swirl sessions are continuous, accepting an incorrect answer early in a lesson can cause problems later on. By using the default answer test, you eliminate this burden by requiring an exact match of expressions and hence not allowing the user to advance until you are certain they've entered the correct response.

It's important to keep in mind that as your content matures, you can always go back and make your answer testing strategy more elaborate. The main benefit of using tests other than the default is that the user will not be required to enter an expression exactly the way you've specified it. He or she will have more freedom in terms of how they respond to a question, as long as they satify the conditions that you see as being most important.

## Predefined Answer Tests

Each of the predefined answer tests listed below has its own help file, where you'll find more detailed explanations and examples.

`any_of_exprs`: Test that the user's expression matches any of several possible expressions.

`expr_creates_var`: Test that a new variable has been created.

`expr_identical_to`: Test that the user has entered a particular expression.

`expr_is_a`: Test that the expression itself is of a specific `class`.

`expr_uses_func`: Test that a particular function has been used.

`func_of_newvar_equals`: Test the result of a computation such as `mean(newVar)` applied to a specific (user-named) variable created in a previous question.

`omnitest`: Test for a correct expression, a correct value, or both.

`val_has_length`: Test that the value of the expression has a particular `length`.

`val_matches`: Test that the user's expression matches a regular expression (`regex`).

`var_is_a`: Test that the *value* of the expression is of a specific `class`.

## Custom Answer Tests

Occasionally, you may want to test something that is outside the scope of our predefined answer tests. Fortunately, this is very easy to do. If you are using the swirlify authoring tool, then a file called `customTests.R` (case-sensitive) is automatically created in the lesson directory. If it's not there already, you can create the file manually.

In this file, you can write your own answer tests. These answer tests are then available to you just the same as any of the standard tests. However, the scope of a custom answer test is limited to the lesson within which you've defined it.

Each custom answer test is simply an R function that follows a few basic rules:

1. Give the function a distinct name that will help you remember what is does (e.g. `creates_matrix_with_n_rows`).
2. The first line of the function body is `e <- get("e", parent.frame())`, which gives you access to the environment `e`. Any important information, such as the expression typed by the user, will be available to you through `e`.
3. Access the expression entered by the user with `e$expr` and the value of the expression with `e$val`. Note that `e$expr` comes in the form of an unevaluated R `expression`.
4. The function returns `TRUE` if the test is passed and `FALSE` otherwise. You should be careful that no other value could be returned (e.g. `NA`, `NULL`, etc.)

## See Also

Other AnswerTests: `any_of_exprs`; `expr_creates_var`; `expr_identical_to`; `expr_is_a`; `expr_uses_func`; `func_of_newvar_equals`; `omnitest`; `val_has_length`; `val_matches`; `var_is_a`

---

`any_of_exprs` *Test that the user has entered one of several possible expressions.*

---

## Description

Returns `TRUE` if the expression the user has entered matches any of the expressions given (as character strings) in the argument.

## Usage

```
any_of_exprs(...)
```

## Arguments

| | |
|---|---|
| `...` | any number of expressions, as character strings |

## Value

`TRUE` or `FALSE`

## See Also

Other AnswerTests: `AnswerTests`; `expr_creates_var`; `expr_identical_to`; `expr_is_a`; `expr_uses_func`; `func_of_newvar_equals`; `omnitest`; `val_has_length`; `val_matches`; `var_is_a`

## Examples

```
## Not run:

# Test that a user has entered either cor(x, y) or cor(y, x)
any_of_exprs('cor(x, y)', 'cor(y, x)')

## End(Not run)
```

---

bye *Exit swirl.*

---

## Description

swirl operates by installing a callback function which responds to commands entered in the R console. This is how it captures and tests answers given by the user in the R console. swirl will remain in operation until this callback is removed, which is what `bye()` does.

## Usage

```
bye()
```

## Examples

```
## Not run:

| Create a new variable called `x` that contains the number 3.

> bye()

| Leaving swirl now. Type swirl() to resume.

## End(Not run)
```

---

email_admin *Send diagnostic email to swirl admin*

---

## Description

Typing `email_admin()` at the prompt will attempt to open a new email in your default browser or email client. The email will include space for you to describe the problem you are experiencing. It will also have the output from `sessionInfo`, which you should not alter.

## Usage

```
email_admin()
```

## expr_creates_var *Test that a new variable has been created.*

### Description

Tests if the `e$expr` creates one new variable (of correct name if given.) If so, returns `TRUE`.

### Usage

```
expr_creates_var(correctName = NULL)
```

### Arguments

| | |
|---|---|
| `correctName` | expected name of the new variable or `NULL` |

### Value

`TRUE` or `FALSE`

### See Also

Other AnswerTests: `AnswerTests`; `any_of_exprs`; `expr_identical_to`; `expr_is_a`; `expr_uses_func`; `func_of_newvar_equals`; `omnitest`; `val_has_length`; `val_matches`; `var_is_a`

### Examples

```
## Not run:
# Test if the user has entered an expression which creates
# a new variable of any name.
expr_creates_var()
#
# Test if the user has entered an expression which creates
# a variable named 'myNum'
#
expr_creates_var('myNum')

## End(Not run)
```

## expr_identical_to *Test that the user has entered a particular expression.*

### Description

Test that the user has entered an expression identical to that given as the first argument.

## Usage

```
expr_identical_to(correct_expression)
```

## Arguments

| | |
|---|---|
| correct_expression | the correct or expected expression as a string |

## Value

TRUE or FALSE

## See Also

Other AnswerTests: AnswerTests; any_of_exprs; expr_creates_var; expr_is_a; expr_uses_func; func_of_newvar_equals; omnitest; val_has_length; val_matches; var_is_a

## Examples

```
## Not run:
  # Test that a user has entered a particular command
  #
  expr_identical_to('myVar <- c(3, 5, 7)')

## End(Not run)
```

---

# expr_is_a *Test that the expression itself is of a specific* class.

---

## Description

Returns TRUE if e$expr is of the given class.

## Usage

```
expr_is_a(class)
```

## Arguments

| | |
|---|---|
| class | expected class of the given expression |

## Value

TRUE or FALSE

## See Also

Other AnswerTests: AnswerTests; any_of_exprs; expr_creates_var; expr_identical_to; expr_uses_func; func_of_newvar_equals; omnitest; val_has_length; val_matches; var_is_a

## Examples

```
## Not run:
# Test if the expression entered by a user is an assignment
#
expr_is_a('<-')

## End(Not run)
```

---

`expr_uses_func` *Test that a particular function has been used.*

---

## Description

Returns `TRUE` if the `e$expr` uses the function whose name is given as the first argument.

## Usage

```
expr_uses_func(func)
```

## Arguments

| | |
|---|---|
| `func` | name of the function expected to be used |

## Value

`TRUE` or `FALSE`

## See Also

Other AnswerTests: `AnswerTests`; `any_of_exprs`; `expr_creates_var`; `expr_identical_to`; `expr_is_a`; `func_of_newvar_equals`; `omnitest`; `val_has_length`; `val_matches`; `var_is_a`

## Examples

```
## Not run:
# Test that the user has entered an expression using sd()
#
expr_uses_func('sd')

## End(Not run)
```

## func_of_newvar_equals *Test the result of a computation applied to a specific (user-named) variable created in a previous question.*

### Description

Tests the result of a computation such as `mean(newVar)` applied to a specific variable created in a previous question and saved behind the scenes as `e$newVar`.

### Usage

```
func_of_newvar_equals(correct_expression)
```

### Arguments

`correct_expression`
: expression expected to be applied

### Value

`TRUE` or `FALSE`

### See Also

Other AnswerTests: `AnswerTests`; `any_of_exprs`; `expr_creates_var`; `expr_identical_to`; `expr_is_a`; `expr_uses_func`; `omnitest`; `val_has_length`; `val_matches`; `var_is_a`

### Examples

```
## Not run:
# Test if user has taken the mean of a variable created
# in an earlier question.
#
func_of_newvar_equals('mean(newVar)')

## End(Not run)
```

## info *Display a list of special commands.*

### Description

Display a list of the special commands, `bye()`, `play()`, `nxt()`, `skip()`, and `info()`.

### Usage

```
info()
```

## Examples

```
## Not run:

| Create a new variable called `z` that contains the number 11.

> info()

| When you are at the R prompt (>):
| -- Typing skip() allows you to skip the current question.
| -- Typing play() lets you experiment with R on your own; swirl will ignore what
| you do...
| -- UNTIL you type nxt() which will regain swirl's attention.
| -- Typing bye() causes swirl to exit. Your progress will be saved.
| -- Typing info() displays these options again.

> bye()

| Leaving swirl now. Type swirl() to resume.

## End(Not run)
```

---

`InstallCourses` *Installing Courses*

---

## Description

swirl is designed so that anyone can create interactive content and share it with the world or with just a few people. Users can install courses from a variety of sources using the functions listed here. Each of these functions has its own help file, which you can consult for more details.

## Details

If you're just getting started, we recommend using `install_from_swirl` to install courses from our official course repository. Otherwise, check out the help file for the relevant install function below.

You can uninstall a course from swirl at any time with `uninstall_course`.

## See Also

Other InstallCourses: `install_course_directory`; `install_course_dropbox`; `install_course_github`; `install_course_google_drive`; `install_course_url`; `install_course_zip`; `install_from_swirl`; `uninstall_course`; `zip_course`

## install_course_directory

*Install a course from a course directory*

### Description

Install a course from a course directory

### Usage

```
install_course_directory(path)
```

### Arguments

| | |
|---|---|
| path | The path to the course directory. |

### See Also

Other InstallCourses: InstallCourses; install_course_dropbox; install_course_github; install_course_google_drive; install_course_url; install_course_zip; install_from_swirl; uninstall_course; zip_course

### Examples

```
## Not run:

install_course_directory("~/Desktop/my_course")

## End(Not run)
```

## install_course_dropbox

*Install a course from a zipped course directory shared on Dropbox*

### Description

Install a course from a zipped course directory shared on Dropbox

### Usage

```
install_course_dropbox(url, multi = FALSE)
```

### Arguments

| | |
|---|---|
| url | URL of the shared file |
| multi | The user should set to TRUE if the zipped directory contains multiple courses. The default value is FALSE. |

## See Also

Other InstallCourses: InstallCourses; install_course_directory; install_course_github; install_course_google_drive; install_course_url; install_course_zip; install_from_swirl; uninstall_course; zip_course

## Examples

```
## Not run:

install_course_dropbox("https://www.dropbox.com/s/xttkmuvu7hh72vu/my_course.zip")

## End(Not run)
```

---

`install_course_github` *Install a course from a GitHub repository*

---

## Description

Install a course from a GitHub repository

## Usage

```
install_course_github(github_username, course_name, branch = "master",
  multi = FALSE)
```

## Arguments

| | |
|---|---|
| `github_username` | The username that owns the course repository. |
| `course_name` | The name of the repository which should be the name of the course. |
| `branch` | The branch of the repository containing the course. The default branch is `"master"`. |
| `multi` | The user should set to `TRUE` if the repository contains multiple courses. The default value is `FALSE`. |

## See Also

Other InstallCourses: InstallCourses; install_course_directory; install_course_dropbox; install_course_google_drive; install_course_url; install_course_zip; install_from_swirl; uninstall_course; zip_course

## Examples

```
## Not run:

install_course_github("bcaffo", "Linear_Regression")
install_course_github("jtleek", "Twitter_Map", "geojson")

## End(Not run)
```

## install_course_google_drive *Install a course from a zipped course directory shared on Google Drive*

### Description

Install a course from a zipped course directory shared on Google Drive

### Usage

```
install_course_google_drive(url, multi = FALSE)
```

### Arguments

| | |
|---|---|
| url | URL of the shared file |
| multi | The user should set to TRUE if the zipped directory contains multiple courses. The default value is FALSE. |

### See Also

Other InstallCourses: InstallCourses; install_course_directory; install_course_dropbox; install_course_github; install_course_url; install_course_zip; install_from_swirl; uninstall_course; zip_course

### Examples

```
## Not run:

install_course_google_drive("https://drive.google.com/file/d/F3fveiu873hfjZZj/edit?usp=sharing")

## End(Not run)
```

## install_course_url *Install a course from a url that points to a zip file*

### Description

Install a course from a url that points to a zip file

### Usage

```
install_course_url(url, multi = FALSE)
```

### Arguments

| | |
|---|---|
| url | URL that points to a zipped course directory |
| multi | The user should set to TRUE if the zipped directory contains multiple courses. The default value is FALSE. |

### See Also

Other InstallCourses: InstallCourses; install_course_directory; install_course_dropbox; install_course_github; install_course_google_drive; install_course_zip; install_from_swirl; uninstall_course; zip_course

### Examples

```
## Not run:

install_course_url("http://www.biostat.jhsph.edu/~rpeng/File_Hash_Course.zip")

## End(Not run)
```

---

## install_course_zip *Install a course from a zipped course folder*

---

### Description

Install a course from a zipped course folder

### Usage

```
install_course_zip(path, multi = FALSE, which_course = NULL)
```

### Arguments

| | |
|---|---|
| path | The path to the zipped course. |
| multi | Set to TRUE if the zipped directory contains multiple courses. The default value is FALSE. |
| which_course | A vector of course names. Only for use when zip file contains multiple courses, but you don't want to install all of them. |

### See Also

Other InstallCourses: InstallCourses; install_course_directory; install_course_dropbox; install_course_github; install_course_google_drive; install_course_url; install_from_swirl; uninstall_course; zip_course

## Examples

```
## Not run:

install_course_zip("~/Desktop/my_course.zip")

install_course_zip("~/Downloads/swirl_courses-master.zip", multi=TRUE,
                   which_course=c("R Programming", "Data Analysis"))

## End(Not run)
```

---

`install_from_swirl` *Install a course from the official course repository*

---

## Description

We are currently maintaining a central repository of contributed swirl courses at https://github.com/swirldev/swirl_courses. This function provides the easiest method of installing a course form the repository.

We have another repository at https://github.com/swirldev/swirl_misc, where we keep experimental features and content. The dev argument allows you to access this repository. Content in the swirl_misc repository is not guaranteed to work.

## Usage

```
install_from_swirl(course_name, dev = FALSE)
```

## Arguments

| | |
|---|---|
| `course_name` | The name of the course you wish to install. |
| `dev` | Set to `TRUE` to install a course in development from the swirl_misc repository. |

## See Also

Other InstallCourses: `InstallCourses`; `install_course_directory`; `install_course_dropbox`; `install_course_github`; `install_course_google_drive`; `install_course_url`; `install_course_zip`; `uninstall_course`; `zip_course`

## Examples

```
## Not run:

install_from_swirl("R_Programming") # Directory name

### OR ###

install_from_swirl("R Programming") # Course name

# To install a course in development from the swirl_misc repository
```

```
install_from_swirl("Including Data", dev = TRUE)

## End(Not run)
```

---

`main` *Return to swirl's main menu.*

---

## Description

Return to swirl's main menu from a lesson in progress.

## Usage

```
main()
```

## Examples

```
## Not run:

| The simplest way to create a sequence of numbers in R is by using
| the `:` operator. Type 1:20 to see how it works.

> main()

| Returning to the main menu...

## End(Not run)
```

---

`nxt` *Begin the upcoming question or unit of instruction.*

---

## Description

This is the way to regain swirl's attention after viewing a video or `play()`'ing around in the console.

## Usage

```
nxt()
```

## Examples

```
## Not run:

| Create a new variable called `y` that contains the number 8.

> play()

| Entering play mode. Experiment as you please, then type nxt()
| when you ready to resume the lesson.

> 10/14
> [1] 0.7142857
> zz <- 99
> zz
> [1] 99
> nxt()

| Resuming lesson...

## End(Not run)
```

---

`omnitest` *Test for a correct expression, a correct value, or both.*

---

## Description

Omnitest can test for a correct expression, a correct value, or both. In the case of values it is limited to testing for character or numeric vectors of length 1.

## Usage

```
omnitest(correctExpr = NULL, correctVal = NULL, strict = FALSE)
```

## Arguments

| | |
|---|---|
| `correctExpr` | the correct or expected expression as a string |
| `correctVal` | the correct value (numeric or character) |
| `strict` | a logical value indicating that the expression should be as expected even if the value is correct. If `FALSE` (the default) a correct value will pass the test even if the expression is not as expected, but a notification will be issued. |

## See Also

Other AnswerTests: `AnswerTests`; `any_of_exprs`; `expr_creates_var`; `expr_identical_to`; `expr_is_a`; `expr_uses_func`; `func_of_newvar_equals`; `val_has_length`; `val_matches`; `var_is_a`

## Examples

```
## Not run:

  # Test that a user has chosen a correct menu item
  #
  omnitest(correctVal='Men in a college dorm.')

  # Test that a user has entered a correct number at the
  # command line
  #
  omnitest(correctVal=19)

  # Test that a user has entered a particular command
  #
  omnitest('myVar <- c(3, 5, 7)')

  # Test that a user has entered a command which computes
  # a specific value but perhaps in a different manner
  # than anticipated
  #
  omnitest('sd(x)^2', 5.95)
  #
  # If the user enters sd(x)*sd(x), rather than sd(x)^2, a notification
  # will be issued, but the test will not fail.

  # Test that a user has entered a command which computes
  # a specific value in a particular way
  #
  omnitest('sd(x)^2', 5.95, strict=TRUE)
  #
  # In this case, if the user enters sd(x)*sd(x) the test will fail.


## End(Not run)
```

---

`play` *Tell swirl to ignore console input for a while.*

---

## Description

It is sometimes useful to play around in the R console out of curiosity or to solidify a concept. This command will cause swirl to remain idle, allowing the user to experiment at will, until the command `nxt()` is entered.

## Usage

```
play()
```

## Examples

```
## Not run:

| Create a new variable called `y` that contains the number 8.

> play()

| Entering play mode. Experiment as you please, then type nxt()
| when you ready to resume the lesson.

> 10/14
> [1] 0.7142857
> zz <- 99
> zz
> [1] 99
> nxt()

| Resuming lesson...

## End(Not run)
```

---

`reset` *Start over on the current script question.*

---

## Description

During a script question, this will reset the script back to its original state, which can be helpful if you get stuck.

## Usage

```
reset()
```

---

`skip` *Skip the current unit of instruction.*

---

## Description

swirl will enter the correct answer and notify the user of the names of any new variables which it may have created in doing so. These may be needed for subsequent questions.

## Usage

```
skip()
```

## Examples

```
## Not run:

| Create a new variable called `y` that contains the number 8.

> skip()

| I've entered the correct answer for you.

| In doing so, I've created the variable(s) y, which you may need later.

## End(Not run)
```

---

`submit` *Submit the active R script in response to a question.*

---

## Description

When a swirl question requires the user to edit an R script, the `submit()` function allows the user to submit their response.

## Usage

```
submit()
```

## Examples

```
## Not run:

| Create a function called f that takes one argument, x, and
| returns the value of x squared.

> submit()

| You are quite good my friend!

## End(Not run)
```

---

`swirl` *An interactive learning environment for R and statistics.*

---

## Description

This function presents a choice of course lessons and interactively tutors a user through them. A user may be asked to watch a video, to answer a multiple-choice or fill-in-the-blanks question, or to enter a command in the R console precisely as if he or she were using R in practice. Emphasis is on the last, interacting with the R console. User responses are tested for correctness and hints are given if appropriate. Progress is automatically saved so that a user may quit at any time and later resume without losing work.

### Usage

```
swirl(resume.class = "default", ...)
```

### Arguments

| | |
|---|---|
| resume.class | for development only; please accept the default. |
| ... | arguments for special purposes only, such as lesson testing |

### Details

There are several ways to exit swirl: by typing bye() while in the R console, by hitting the Esc key while not in the R console, or by entering 0 from the swirl course menu. swirl will print a goodbye message whenever it exits.

While swirl is in operation, it may be controlled by entering special commands in the R console. One of the special commands is bye() as discussed above. Others are play(), nxt(), skip(), and info(). The parentheses are important.

Sometimes a user will want to play around in the R console without interference or commentary from swirl. This can be accomplished by using the special command play(). swirl will remain in operation, silently, until the special command nxt() is entered.

The special command skip() can be used to skip a question if necessary. swirl will enter the correct answer and notify the user of the names of any new variables which it may have created in doing so. These may be needed for subsequent questions.

Finally, info() may be used to display a list of the special commands themselves with brief explanations of what they do.

### Examples

```
## Not run:

swirl()

## End(Not run)
```

---

## uninstall_course *Uninstall a course*

---

### Description

Uninstall a course

### Usage

```
uninstall_course(course_name)
```

### Arguments

| | |
|---|---|
| course_name | Name of course to be uninstalled |

## See Also

Other InstallCourses: InstallCourses; install_course_directory; install_course_dropbox; install_course_github; install_course_google_drive; install_course_url; install_course_zip; install_from_swirl; zip_course

## Examples

```
## Not run:

uninstall_course("Linear Regression")

## End(Not run)
```

---

# val_has_length *Test that the value of the expression has a particular* length.

---

## Description

Test the the length of e$val is that given by the first argument.

## Usage

```
val_has_length(len)
```

## Arguments

| | |
|---|---|
| len | expected length of the variable created by a user |

## Value

TRUE or FALSE

## See Also

Other AnswerTests: AnswerTests; any_of_exprs; expr_creates_var; expr_identical_to; expr_is_a; expr_uses_func; func_of_newvar_equals; omnitest; val_matches; var_is_a

## Examples

```
## Not run:
# Test that the user has created a varible of length 10
#
val_has_length(10)

## End(Not run)
```

## val_matches — *Test that the user's expression matches a regular expression.*

### Description

Returns TRUE if `as.character(e$val)` matches the regular expression given as the first argument.

### Usage

```
val_matches(regular_expression)
```

### Arguments

| | |
|---|---|
| regular_expression | a regular expression which user value should match |

### Value

TRUE or FALSE

### See Also

Other AnswerTests: AnswerTests; any_of_exprs; expr_creates_var; expr_identical_to; expr_is_a; expr_uses_func; func_of_newvar_equals; omnitest; val_has_length; var_is_a

### Examples

```
## Not run:
  # Test that a user has entered a value matching
  # '[Cc]ollege [Ss]tudents' or has selected it
  # in a multiple choice question.
  #
  val_matches('[Cc]ollege [Ss]tudents')

## End(Not run)
```

## var_is_a — *Test that the value of the expression is of a specific class.*

### Description

Returns TRUE if a variable of the given name exists in the global environment and is of the given class.

### Usage

```
var_is_a(class, var_name)
```

### Arguments

| | |
|---|---|
| class | expected class which the given variable |
| var_name | name of the variable |

### Value

TRUE or FALSE

### See Also

Other AnswerTests: AnswerTests; any_of_exprs; expr_creates_var; expr_identical_to; expr_is_a; expr_uses_func; func_of_newvar_equals; omnitest; val_has_length; val_matches

### Examples

```
## Not run:
# Test that a variable named "x" in the global environmentis numeric.
var_is_a('numeric', 'x')

## End(Not run)
```

---

## zip_course *Zip a course directory*

---

### Description

Zip a course directory

### Usage

```
zip_course(path, dest = NULL)
```

### Arguments

| | |
|---|---|
| path | Path to the course directory to be zipped. |
| dest | Path to directory in which the .zip should be saved. The default value is NULL, which will cause the .zip to be created one level above the directory specified in path. |

### See Also

Other InstallCourses: InstallCourses; install_course_directory; install_course_dropbox; install_course_github; install_course_google_drive; install_course_url; install_course_zip; install_from_swirl; uninstall_course

## Examples

```
## Not run:

zip_course("~/Desktop/LOESS_Modeling")
zip_course("~/Desktop/SNA_Tutorial", "~/tutorials")

## End(Not run)
```

# Index


AnswerTests, 2, *4*, *6–9*, *17*, *22–24*
any_of_exprs, *3, 4*, 4, *6–9*, *17*, *22–24*

bye, 5

class, *3*, *7*

email_admin, 5
expr_creates_var, *3, 4*, 6, *7–9*, *17*, *22–24*
expr_identical_to, *3, 4*, *6*, 6, *7–9*, *17*, *22–24*
expr_is_a, *3, 4*, *6, 7*, 7, *8, 9*, *17*, *22–24*
expr_uses_func, *3, 4*, *6, 7*, 8, *9*, *17*, *22–24*
expression, *4*

func_of_newvar_equals, *3, 4*, *6–8*, 9, *17*, *22–24*

info, 9
install_course_directory, *10*, 11, *12–15*, *22*, *24*
install_course_dropbox, *10, 11*, 11, *12–15*, *22*, *24*
install_course_github, *10–12*, 12, *13–15*, *22*, *24*
install_course_google_drive, *10–12*, 13, *14, 15*, *22*, *24*
install_course_url, *10–13*, 13, *14, 15*, *22*, *24*
install_course_zip, *10–14*, 14, *15*, *22*, *24*
install_from_swirl, *10–14*, 15, *22*, *24*
InstallCourses, 10, *11–15*, *22*, *24*

length, *3*, *22*

main, 16

nxt, 16

omnitest, *3, 4*, *6–9*, 17, *22–24*

play, 18

regex, *3*
reset, 19

skip, *2*, 19
submit, 20
swirl, 20

uninstall_course, *10–15*, 21, *24*

val_has_length, *3, 4*, *6–9*, *17*, 22, *23, 24*
val_matches, *3, 4*, *6–9*, *17*, *22*, 23, *24*
var_is_a, *3, 4*, *6–9*, *17*, *22, 23*, 23

zip_course, *10–15*, *22*, 24