

# Package ‘swirl’

October 14, 2022

**Title** Learn R, in R

**Description** Use the R console as 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** 2.4.5

**License** MIT + file LICENSE

**Depends** R (>= 3.1.0)

**Imports** stringr, testthat (>= 1.0.2), httr (>= 1.1.0), yaml, RCurl, digest, tools, methods

**Suggests** stringi

**Encoding** UTF-8

**LazyData** true

**RoxygenNote** 7.0.2

**NeedsCompilation** no

**Author** Sean Kross [aut, cre],  
Nick Carchedi [aut],  
Bill Bauer [aut],  
Gina Grdina [aut],  
Filip Schouwenaars [ctb],  
Wush Wu [ctb]

**Maintainer** Sean Kross <sean@seankross.com>

**Repository** CRAN

**Date/Publication** 2020-01-16 15:20:02 UTC

## R topics documented:

AnswerTests . . . . .	2
any_of_exprs . . . . .	5
bye . . . . .	5

calculates_same_value . . . . .	6
delete_progress . . . . .	7
email_admin . . . . .	7
expr_creates_var . . . . .	8
expr_identical_to . . . . .	9
expr_is_a . . . . .	9
expr_uses_func . . . . .	10
func_of_newvar_equals . . . . .	11
info . . . . .	12
InstallCourses . . . . .	12
install_course . . . . .	13
install_course_directory . . . . .	14
install_course_dropbox . . . . .	14
install_course_github . . . . .	15
install_course_google_drive . . . . .	16
install_course_url . . . . .	17
install_course_zip . . . . .	17
install_from_swirl . . . . .	18
is_robust_match . . . . .	19
main . . . . .	20
nxt . . . . .	20
omnitest . . . . .	21
play . . . . .	22
reset . . . . .	23
restart . . . . .	23
rmatch_calls . . . . .	24
select_language . . . . .	25
skip . . . . .	25
submit . . . . .	26
swirl . . . . .	26
swirl_options . . . . .	27
uninstall_all_courses . . . . .	28
uninstall_course . . . . .	29
val_has_length . . . . .	29
val_matches . . . . .	30
var_is_a . . . . .	31
zip_course . . . . .	31

**Index****33**

## 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 [skips](#) 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 `omni test(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 satisfy 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.

`calculates_same_value`: Test that the user's expression evaluates to a certain value.

`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 it 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()`, `calculates_same_value()`, `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	<i>Test that the user has entered one of several possible expressions.</i>
--------------	--

---

**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](#), [calculates\\_same\\_value\(\)](#), [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	<i>Exit swirl.</i>
-----	--------------------

---

**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)
```

---

calculates\_same\_value *Test that the user's expression evaluates to a certain value.*

---

**Description**

Test that the value calculated by the user's expression is the same as the value calculated by the given expression.

**Usage**

```
calculates_same_value(expression)
```

**Arguments**

expression      An expression whose value will be compared to the value of the user's expression.

**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\(\)](#), [var\\_is\\_a\(\)](#)

**Examples**

```
## Not run:
# Test that a user's expression evaluates to a certain value
#
calculates_same_value('matrix(1:20, nrow=4, ncol=5)')

## End(Not run)
```

---

delete_progress	<i>Delete a user's progress</i>
-----------------	---------------------------------

---

**Description**

Delete a user's progress

**Usage**

```
delete_progress(user, path = NULL)
```

**Arguments**

user	The user name whose progress will be deleted.
path	If specified, the directory where the user_data can be found

**Examples**

```
## Not run:  
  
delete_progress("bill")  
  
## End(Not run)
```

---

email_admin	<i>Send diagnostic email to swirl admin</i>
-------------	---

---

**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	<i>Test that a new variable has been created.</i>
------------------	---

---

## 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\(\)](#), [calculates\\_same\\_value\(\)](#), [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\(\)](#), [calculates\\_same\\_value\(\)](#), [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\(\)](#), [calculates\\_same\\_value\(\)](#), [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	<i>Test that a particular function has been used.</i>
----------------	---

---

**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\(\)](#), [calculates\\_same\\_value\(\)](#), [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\(\)](#), [calculates\\_same\\_value\(\)](#), [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	<i>Display a list of special commands.</i>
------	--

---

### Description

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

### Usage

`info()`

---

InstallCourses	<i>Installing Courses</i>
----------------	---------------------------

---

### 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\\_course](#) 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](#).

Uninstall all courses with [uninstall\\_all\\_courses](#).

### See Also

Other InstallCourses: [install\\_course\\_directory\(\)](#), [install\\_course\\_dropbox\(\)](#), [install\\_course\\_github\(\)](#), [install\\_course\\_google\\_drive\(\)](#), [install\\_course\\_url\(\)](#), [install\\_course\\_zip\(\)](#), [install\\_course\(\)](#), [install\\_from\\_swirl\(\)](#), [uninstall\\_all\\_courses\(\)](#), [uninstall\\_course\(\)](#), [zip\\_course\(\)](#)

---

install_course	<i>Install a course from The swirl Course Network or install a course from a local .swc file.</i>
----------------	---

---

### Description

Version 2.4 of swirl introduces a new, simple, and fast way of installing courses in the form of .swc files. This function allows a user to grab a .swc file from The swirl Course Network which is maintained by Team swirl, or the user can use this function to install a local .swc file. When using this function please only provide an argument for either course\_name or swc\_path, never both.

### Usage

```
install_course(course_name = NULL, swc_path = NULL, force = FALSE)
```

### Arguments

course_name	The name of the course you wish to install.
swc_path	The path to a local .swc file. By default this argument defaults to file.choose() so the user can select the file using their mouse.
force	Should course installation be forced? 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\\_url\(\)](#), [install\\_course\\_zip\(\)](#), [install\\_from\\_swirl\(\)](#), [uninstall\\_all\\_courses\(\)](#), [uninstall\\_course\(\)](#), [zip\\_course\(\)](#)

### Examples

```
## Not run:

# Install the latest version of Team swirl's R Programming course.
install_course("R Programming")

# Install a local .swc file by using your mouse and keyboard to select the
# file.
install_course()

# Install a .swc file from a specific path.
install_course(swc_path = file.path("~", "Downloads", "R_Programming.swc"))

## End(Not run)
```

---

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\\_course\(\)](#), [install\\_from\\_swirl\(\)](#), [uninstall\\_all\\_courses\(\)](#), [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\\_course\(\)](#), [install\\_from\\_swirl\(\)](#), [uninstall\\_all\\_courses\(\)](#), [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

<code>github_username</code>	The username that owns the course repository.
<code>course_name</code>	The name of the repository which should be the name of the course.
<code>branch</code>	The branch of the repository containing the course. The default branch is "master".
<code>multi</code>	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\\_course\(\)](#), [install\\_from\\_swirl\(\)](#), [uninstall\\_all\\_courses\(\)](#), [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\\_course\(\)](#), [install\\_from\\_swirl\(\)](#), [uninstall\\_all\\_courses\(\)](#), [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\\_course\(\)](#), [install\\_from\\_swirl\(\)](#), [uninstall\\_all\\_courses\(\)](#), [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\\_course\(\)](#), [install\\_from\\_swirl\(\)](#), [uninstall\\_all\\_courses\(\)](#), [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](https://github.com/swirldev/swirl_courses). This function provides the easiest method of installing a course from the repository.

We have another repository at [https://github.com/swirldev/swirl\\_misc](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.

The central repository of swirl courses is mirrored at [https://bitbucket.org/swirldevmirror/swirl\\_courses](https://bitbucket.org/swirldevmirror/swirl_courses). If you cannot access GitHub you can download swirl courses from bitbucket by using the mirror = "bitbucket" option (see below).

**Usage**

```
install_from_swirl(course_name, dev = FALSE, mirror = "github")
```

**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.
mirror	Select swirl course repository mirror. Valid arguments are "github" and "bitbucket".

**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\\_course\(\)](#), [uninstall\\_all\\_courses\(\)](#), [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)

# To install a course from the Bitbucket mirror
install_from_swirl("R Programming", mirror = "bitbucket")

## End(Not run)
```

---

is_robust_match	<i>Recursively expand both the correct expression and the user's expression and test for a match. CAUTION: May raise errors, as in rmatch_calls.</i>
-----------------	--

---

**Description**

Recursively expand both the correct expression and the user's expression and test for a match. CAUTION: May raise errors, as in rmatch\_calls.

**Usage**

```
is_robust_match(expr1, expr2, eval_for_class, eval_env = NULL)
```

**Arguments**

expr1	expression
expr2	expression
eval_for_class	TRUE or FALSE. If TRUE, evaluate the first argument of an S3 method to determine its class. Default=FALSE.
eval_env	parent environment for evaluations to determine class. Ignored if eval_for_class=FALSE

**Value**

TRUE or FALSE according to whether expanded expressions match.

**Examples**

```
## Not run:

  expr1 <- quote(print(paste("my_name_is", "mud")))
  expr2 <- quote(print(paste("my_name_is", "mud", sep=" ")))
  err <- try(ans <- is_robust_match(expr1, expr2, eval_for_class=TRUE), silent=TRUE)
  if(is(ans, "try-error")){
    ans <- isTRUE(all.equal())
  }

## 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,
  eval_for_class = as.logical(NA)
)
```

**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.
eval_for_class	a logical value. If TRUE, evaluate the first argument of an S3 method to determine its class. Default=TRUE. Global value may also be set as customTests\$EVAL_FOR_CLASS.

**See Also**

Other AnswerTests: [AnswerTests](#), [any\\_of\\_exprs\(\)](#), [calculates\\_same\\_value\(\)](#), [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 `next()` 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()
```

---

```
restart
```

*Restart the current swirl lesson.*

---

**Description**

Restart the current swirl lesson.

**Usage**

```
restart()
```

---

rmatch\_calls

*Recursively expand match calls in an expression from the bottom up.*


---

## Description

Given an expression, `expr`, traverse the syntax tree from the bottom up, expanding the call to include default values of named formals as appropriate, and applying `match.call` to the result. Functionality is limited to expressions containing ordinary functions or S3 methods. If parameter `eval_for_class` has its default value of `FALSE`, an error will be raised for any S3 method whose first argument (as an expression) is not atomic. If `eval_for_class` is `TRUE`, the first argument will be evaluated to determine its class. Evaluation will take place in the environment given by parameter `eval_env`. CAUTION: `eval_for_class=TRUE` is likely to result in multiple evaluations of the same code. Expressions containing S4 or reference class methods will also raise errors.

## Usage

```
rmatch_calls(expr, eval_for_class = FALSE, eval_env = NULL)
```

## Arguments

<code>expr</code>	an R expression (a.k.a. abstract syntax tree)
<code>eval_for_class</code>	<code>TRUE</code> or <code>FALSE</code> . If <code>TRUE</code> , evaluate the first argument of an S3 method to determine its class. Default= <code>FALSE</code> .
<code>eval_env</code>	environment in which to evaluate for class. Ignored if <code>eval_for_class=FALSE</code>

## Value

an equivalent R expression with function or method calls in canonical form.

## Examples

```
## Not run:

# Function
rmatch_calls(quote(help("print")))
help(topic = "print", package = NULL, lib.loc = NULL, verbose = getOption("verbose"),
try.all.packages = getOption("help.try.all.packages"), help_type = getOption("help_type"))

# S3 method with atomic first argument
rmatch_calls(quote(seq(0, 1, by=.5)))
seq(from = 0, to = 1, by = 0.5, length.out = NULL, along.with = NULL)

# S3 method with non-atomic first argument, eval_for_class = FALSE (default)
rmatch_calls(quote(seq(as.Date("2014-02-01"), as.Date("2014-03-01"))))
#Error in rmatch_calls(quote(seq(as.Date("2014-02-01"), as.Date("2014-03-01")))) :
# Illegal expression, seq(as.Date(x = "2014-02-01"), as.Date(x = "2014-03-01")):
# The first argument, as.Date(x = "2014-02-01"), to S3 method 'seq', is a call,
# which (as an expression) is not atomic, hence its class can't be determined in an
```



```
# abstract syntax tree without additional information.

# S3 method with non-atomic first argument, eval_for_class = TRUE
rmatch_calls(quote(seq(as.Date("2014-02-01"), as.Date("2014-03-01"))), eval_for_class=TRUE)
seq(from = as.Date(x = "2014-02-01"), to = as.Date(x = "2014-03-01"),
     length.out = NULL, along.with = NULL)

## End(Not run)
```

---

select_language	<i>Select a language</i>
-----------------	--------------------------

---

### Description

Select a language for swirl's menus.

### Usage

```
select_language(language = NULL, append_rprofile = FALSE)
```

### Arguments

**language** The language that swirl's menus will use. This must be one of the following values: "chinese\_simplified", "english", "french", "german", "korean", "spanish", or "turkish". If NULL the user will be asked to choose a language interactively. The default value is NULL.

**append\_rprofile**

If TRUE this command will append options(swirl\_language = [selected language]) to the end of your Rprofile. The default value is FALSE.

---

skip	<i>Skip the current unit of instruction.</i>
------	--

---

### 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)
```

---

swirl\_options

*Get swirl options*

---

**Description**

This function is a wrapper for `options()` that allows the user to see the state of how certain options for swirl are set up.

**Usage**

```
swirl_options(...)
```

## Arguments

... any options can be defined, using name = value.

## Examples

```
## Not run:  
# See current current swirl options  
swirl_options()  
  
# Set an option  
swirl_options(swirl_logging = TRUE)  
  
## End(Not run)
```

---

uninstall\_all\_courses *Uninstall all courses*

---

## Description

Uninstall all courses

## Usage

```
uninstall_all_courses(force = FALSE)
```

## Arguments

force If TRUE the user will not be asked if they're sure they want to delete the contents of the directory where courses are stored. 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\\_url\(\)](#), [install\\_course\\_zip\(\)](#), [install\\_course\(\)](#), [install\\_from\\_swirl\(\)](#), [uninstall\\_course\(\)](#), [zip\\_course\(\)](#)

## Examples

```
## Not run:  
  
uninstall_all_courses()  
  
## End(Not run)
```

---

uninstall_course	<i>Uninstall a course</i>
------------------	---------------------------

---

**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\\_course\(\)](#), [install\\_from\\_swirl\(\)](#), [uninstall\\_all\\_courses\(\)](#), [zip\\_course\(\)](#)

**Examples**

```
## Not run:  
  
uninstall_course("Linear Regression")  
  
## End(Not run)
```

---

val_has_length	<i>Test that the value of the expression has a particular length.</i>
----------------	---

---

**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\(\)](#), [calculates\\_same\\_value\(\)](#), [expr\\_creates\\_var\(\)](#), [expr\\_identical\\_to\(\)](#), [expr\\_is\\_a\(\)](#), [expr\\_uses\\_func\(\)](#), [func\\_of\\_newvar\\_equals\(\)](#), [ommitest\(\)](#), [val\\_matches\(\)](#), [var\\_is\\_a\(\)](#)

**Examples**

```
## Not run:
# Test that the user has created a variable 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\(\)](#), [calculates\\_same\\_value\(\)](#), [expr\\_creates\\_var\(\)](#), [expr\\_identical\\_to\(\)](#), [expr\\_is\\_a\(\)](#), [expr\\_uses\\_func\(\)](#), [func\\_of\\_newvar\\_equals\(\)](#), [ommitest\(\)](#), [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	<i>Test that the value of the expression is of a specific class.</i>
----------	--

---

**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\(\)](#), [calculates\\_same\\_value\(\)](#), [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 environment is numeric.
var_is_a('numeric', 'x')

## End(Not run)
```

---

zip_course	<i>Zip a course directory</i>
------------	-------------------------------

---

**Description**

**Warning:** This function will be deprecated after swirl version 2.4.

**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\\_course\(\)](#), [install\\_from\\_swirl\(\)](#), [uninstall\\_all\\_courses\(\)](#), [uninstall\\_course\(\)](#)

**Examples**

```
## Not run:  
  
zip_course("~/Desktop/LOESS_Modeling")  
zip_course("~/Desktop/SNA_Tutorial", "~/tutorials")  
  
## End(Not run)
```



# Index

## \* AnswerTests

AnswerTests, 2

any\_of\_exprs, 5

calculates\_same\_value, 6

expr\_creates\_var, 8

expr\_identical\_to, 9

expr\_is\_a, 9

expr\_uses\_func, 10

func\_of\_newvar\_equals, 11

omnittest, 21

val\_has\_length, 29

val\_matches, 30

var\_is\_a, 31

## \* InstallCourses

install\_course, 13

install\_course\_directory, 14

install\_course\_dropbox, 14

install\_course\_github, 15

install\_course\_google\_drive, 16

install\_course\_url, 17

install\_course\_zip, 17

install\_from\_swirl, 18

InstallCourses, 12

uninstall\_all\_courses, 28

uninstall\_course, 29

zip\_course, 31

AnswerTests, 2, 5, 6, 8–11, 22, 30, 31

any\_of\_exprs, 3, 4, 5, 6, 8–11, 22, 30, 31

bye, 5

calculates\_same\_value, 4, 5, 6, 8–11, 22, 30, 31

class, 4, 9

delete\_progress, 7

email\_admin, 7

expr\_creates\_var, 4–6, 8, 9–11, 22, 30, 31

expr\_identical\_to, 4–6, 8, 9, 10, 11, 22, 30, 31

expr\_is\_a, 4–6, 8, 9, 9, 10, 11, 22, 30, 31

expr\_uses\_func, 4–6, 8–10, 10, 11, 22, 30, 31

expression, 4

func\_of\_newvar\_equals, 4–6, 8–10, 11, 22, 30, 31

info, 12

install\_course, 12, 13, 14–19, 28, 29, 32

install\_course\_directory, 12, 13, 14, 15–19, 28, 29, 32

install\_course\_dropbox, 12–14, 14, 15–19, 28, 29, 32

install\_course\_github, 12–15, 15, 16–19, 28, 29, 32

install\_course\_google\_drive, 12–15, 16, 17–19, 28, 29, 32

install\_course\_url, 12–16, 17, 18, 19, 28, 29, 32

install\_course\_zip, 12–17, 17, 19, 28, 29, 32

install\_from\_swirl, 12–18, 18, 28, 29, 32

InstallCourses, 12, 13–19, 28, 29, 32

is\_robust\_match, 19

length, 4, 29

main, 20

nxt, 20

omnittest, 4–6, 8–11, 21, 30, 31

play, 22

regex, 4

reset, 23

restart, 23

rmatch\_calls, 24

select\_language, [25](#)  
skip, [3](#), [25](#)  
submit, [26](#)  
swirl, [26](#)  
swirl\_options, [27](#)

uninstall\_all\_courses, [12–19](#), [28](#), [29](#), [32](#)  
uninstall\_course, [12–19](#), [28](#), [29](#), [32](#)

val\_has\_length, [4–6](#), [8–11](#), [22](#), [29](#), [30](#), [31](#)  
val\_matches, [4–6](#), [8–11](#), [22](#), [30](#), [30](#), [31](#)  
var\_is\_a, [4–6](#), [8–11](#), [22](#), [30](#), [31](#)

zip\_course, [12–19](#), [28](#), [29](#), [31](#)