Package: RTutor (via r-universe)

August 26, 2024

Type Package

Title Interactive R problem sets with automatic testing of solutions and automatic hints

Version 2020.11.25

Date 2020-11-25

Author Sebastian Kranz

Maintainer Sebastian Kranz <sebastian.kranz@uni-ulm.de>

Description Interactive R problem sets with automatic testing of solutions and automatic hints

License Programm code: GPL >= 2.0 Contributed problem sets: Creative Commons (CY)

Depends shinyEvents,markdown,whisker, stringr, stringtools, data.table, tibble, dplyr, shiny (>= 0.11.1), dplyrExtras, shinyAce, shinyBS (>= 0.61), hwriter, restorepoint, RCurl, knitr, jsonlite, DT, memoise, yaml, rmarkdown

Suggests digest

RoxygenNote 7.1.1

Repository https://skranz.r-universe.dev

RemoteUrl https://github.com/skranz/RTutor

RemoteRef master

RemoteSha 4502cab94cb9233e4b11aab42b29f6d2b0576113

Contents

add.failure *Used inside tests: adds a failure to an exercise*

Description

Used inside tests: adds a failure to an exercise

Usage

```
add.failure(message, ..., add.new.line = TRUE, ps = get.ps())
```

Arguments

message	a longer description shown to the user
...	variables that will be rendered into messages that have whiskers

add.success *Used inside tests: adds a sucess message*

Description

Used inside tests: adds a sucess message

Usage

```
add.success(message, ..., ps = get.ps())
```

Arguments

message	a longer description shown to the user
...	variables that will be rendered into messages that have whiskers

add.warning *Used inside tests: adds a warning*

Description

Used inside tests: adds a warning

Usage

```
add.warning(message, ..., ps = get.ps())
```

Arguments

message	a longer description shown to the user
...	variables that will be rendered into messages that have whiskers

auto.hint *This is just a place holder in a hint block*

Description

Only used inside a hint block.

Usage

```
auto.hint()
```

Details

It says that the automatic hint shall be shown. This makes sense if you want to show the automatic hint in addition to a custom hint. Also see auto_hint_else()

auto.hint.else *This is just a place holder in a hint block*

Description

Only used inside a hint block.

Usage

```
auto.hint.else()
```

Details

It says that the automatic hint should be shown unless some hint with hint.stud.call has been shown (or ps$shown.custom.hints has been manually assigned a value above 0.)

awards *Show all your awards*

Description

Show all your awards

Usage

```
awards(ups = get.ups(), as.html = FALSE, details = TRUE, ps = get.ps())
```

check.assign *Checks an assignment to a variable*

Description

By default a solution is considered correct if the assignment yields the same value than the sample solution, or has the same rhs (e.g. a call runif(1,0,1)), even if the value differs.

Usage

```
check.assign(
  call,
  check.arg.by.value = TRUE,
  allow.extra.arg = FALSE,
  ignore.arg = NULL,
  success.message = NULL,
  failure.message = NULL,

   no.command.failure.message = "You have not yet included correctly, all required R commands in your co
  ok.if.same.val = TRUE,
  call.object = NULL,
  s3.method = NULL,
  ps = get.ps(),
  stud.env = ps$stud.env,
  part = ps$part,
  stud.expr.li = ps$stud.expr.li,
  verbose = FALSE,
  only.check.assign.exists = FALSE,
  noeval = isTRUE(ps$noeval),
  other.sols = NULL,
  check.cols = NULL,
  sort.cols = NULL,
  ...
)
```

Arguments

call	the correct assignment that shall be checked (not a quoted call)
allow.extra.arg	if TRUE (not default) the student is allowed to supply additional arguments to the call that were not in the solution. Useful, e.g. if the student shall plot something and is allowed to customize her plot with additional arguments.
ignore.arg	a vector of argument names that will be ignored when checking correctness
ok.if.same.val	if TRUE (default) the call will be considered as correct, if it yields the same resulting value as the solution, even if its arguments differ.
call.object	alternatively to call a quoted call (call object)

`only.check.assign.exists`
: if TRUE (default = FALSE) only check if an assignemnt to the lhs variable exists no matter whether the assignment is correct. May be sensible if there are additional tests specified afterwards that check some characteristics of the assigned variable.

`other.sols`
: a list of quoted assignments, e.g. `list(quote(x<-5), quote(x<-10))` of other solutions that are also correct.

`check.cols`
: only relevant if a data frame (or tibble) is computed. An optional character vector of column names. If provided only check whether those columns are correctly computed but ignore other columns. Only works if `compare.vals = TRUE` (default).

`sort.cols`
: only relevant if a data frame (or tibble) is computed. An optional character vector of column names. If provided sort the sample solution and student's solution by these columns before comparing. This means that also solutions that are originally sorted in a different fashion are accepted. Useful in combination with `check.cols`.

`check.assign.with.multiple.sol`
: *Checks an assignment to a variable with up to 5 possibly correct solutions*

Description

Can be called in a #< test block for a custom test.

Usage

```
check.assign.with.multiple.sol(
  sol1,
  sol2,
  sol3,
  sol4,
  sol5,
  ...,
  sol.list = list()
)
```

Arguments

`sol1`
: An assignment that needs to be checked, e.g. x<-5. Similar for sol2, sol3, sol4, sol5.

Examples

```
# Assume the task is that x shall be a number
# below 11 and divisible by 5

check.assign.with.multiple.sol(x<-5, x<-10)
```

`check.call` *Checks whether the user makes a particular function call in his code or call a particular R statement*

Description

Checks whether the user makes a particular function call in his code or call a particular R statement

Usage

```
check.call(
  call,
  check.arg.by.value = TRUE,
  allow.extra.arg = FALSE,
  ignore.arg = NULL,
  success.message = NULL,
  failure.message = NULL,
  no.command.failure.message = NULL,
  ok.if.same.val = NA,
  s3.method = NULL,
  ps = get.ps(),
  stud.env = ps$stud.env,
  part = ps$part,
  stud.expr.li = ps$stud.expr.li,
  verbose = FALSE,
  noeval = isTRUE(ps$noeval),
  hint.on.fail = isTRUE(ps$rps$hint.on.fail),
  check.cols = NULL,
  sort.cols = NULL,
  is.ggplot = FALSE,
  ...
)
```

Arguments

`call`	the correct function call that shall be checked (not a quoted call)
`check.arg.by.value`	if TRUE (default) check whether students arguments have the same value than in given call, even if their unevaluted representation looks different

allow.extra.arg
: if TRUE (not default) the student is allowed to supply additional arguments to the call that were not in the solution. Useful, e.g. if the student shall plot something and is allowed to customize her plot with additional arguments.

ignore.arg
: a vector of argument names that will be ignored when checking correctness

ok.if.same.val
: if TRUE (default) the call will be considered as correct, if it yields the same resulting value as the solution, even if its arguments differ.

hint.on.fail
: Shall automatically be a hint shown if a test fails. By default FALSE, i.e. student has to type `hint()`. Yet, default can be overwritten in call to `create.ps`.

check.cols
: only relevant if a data frame (or tibble) is computed. An optional character vector of column names. If provided only check whether those columns are correctly computed but ignore other columns. Only works if `compare.vals = TRUE` (default).

sort.cols
: only relevant if a data frame (or tibble) is computed. An optional character vector of column names. If provided sort the sample solution and student's solution by these columns before comparing. This means that also solutions that are originally sorted in a different fashion are accepted. Useful in combination with `check.cols`.

`check.col` *Test: Compare the column col of the matrix or data.frame df with either the values from the given solutions or with the result of an expression that is evaluated in the students solution*

Description

Test: Compare the column col of the matrix or data.frame df with either the values from the given solutions or with the result of an expression that is evaluated in the students solution

Usage

```
check.col(
  df,
  col,
  expr = NULL,
  class.df = c("data.frame", "data.table", "matrix"),
  check.all = FALSE,
  exists = check.all,
  length = check.all,
  class = check.all,
  values = check.all,
  tol = .Machine$double.eps^0.5,
  failure.exists = "{{df}} does not have a column {{col}}.",

   failure.length = "{{df}} has {{length_stud}} rows but it shall have {{length_sol}} rows.",
```

```
  failure.class = "Column {{col}} of {{df}} has a wrong class. It should be {{class_sol}} but it is {{cl
  failure.values = "Column {{col}} of {{df}} has wrong values.",
  failure.message.add = NULL,
  success.message = "Great, column {{col}} of {{df}} has correct {{tests}}.",
  part = NULL,
  ps = get.ps(),
  stud.env = ps$stud.env,
  verbose = FALSE,
  unsubst.expr = NULL,
  str.expr = NULL
)
```

Arguments

df	name of the data frame or matrix
col	name of the column
expr	the test expression that will be evaluated
exists	shall existence be checked (similar length, class, values)
failure.exists	a message that is shown if the variable does not exists (similar the other failure.??? variables)
failure.message.add	a text that will be added to all failure messages

---

check.expr *Test: Compare the expression check.expr evaluated in the student's environment with the solution correct.expr*

Description

Test: Compare the expression check.expr evaluated in the student's environment with the solution correct.expr

Usage

```
check.expr(
  check.expr,
  correct.expr,
  failure.message = "{{check_expr}} has the wrong values!",
  success.message = "Great, {{check_expr}} seems correct.",
  part = NULL,
  ps = get.ps(),
  stud.env = ps$stud.env,
  verbose = FALSE,
  unsubst.check.expr = NULL,
  unsubst.correct.expr = NULL,
  str.check.expr = NULL,
```

```
  str.correct.expr = NULL,
  tol = .Machine$double.eps^0.5
)
```

Arguments

check.expr	the expression to be checked
correct.expr	the correct expression
vars	a variable name or vector of variable names
exists	shall existence be checked (similar length, class, values)
failure.exists	a message that is shown if the variable does not exists (similar the other failure.??? variables)
failure.message.add	a text that will be added to all failure messages

---

check.file.exists *Check whether a given file exists*

Description

Check whether a given file exists

Usage

```
check.file.exists(
  file,
  failure.message = paste0("Sorry, but I cannot find the file \"", file,
    "\" in your current working directory."),
  success.message = paste0("Great, I have found the file \"", file, "\"!"),
  ps = get.ps(),
  part = NULL,
  ...
)
```

---

check.function *Checks a function written by the student*

Description

Checks a function written by the student

Usage

```
check.function(
  code,
  ...,
  check.args = TRUE,
  check.defaults = FALSE,
  check.args.order = TRUE,
  allow.extra.arg = TRUE,
  ps = get.ps(),
  stud.env = ps$stud.env,
  verbose = FALSE,
  part = NULL
)
```

Arguments

code	code of the form fun_name = function(x,y) #body of function. It is important to wrap the code in and to assign the function name with = (don't use <-). See example below.
...	you can add several test calls to the function. It will be checked whether the users' function returns the same values in those calls than the function in the solution. You can also have a code block wrapped in that ends with a call to the function. In this way you can e.g. specify a random seeds before calling the function.
check.args	if TRUE check the arguments of the user function. If a character vector only check the given arguments.
check.defaults	TRUE = check the default values of the arguments of the user function. If a character vector only check the default values of the given arguments.
check.args.order	if TRUE make sure that the checked arguments appear in the same order in the user function than in the solution
allow.extra.arg	if TRUE the user function can have additional arguments (at the end) that are not in the solution

---

check.problem.set *Checks a student problem set*

Description

The command will be put at the top of a student's problem set. It checks all exercises when the problem set is sourced. If something is wrong, an error is thrown and no more commands will be sourced.

Usage

```
check.problem.set(
  ps.name,
  stud.path,
  stud.short.file,
  reset = FALSE,
  set.warning.1 = TRUE,
  user.name = "GUEST",
  do.check = interactive(),
  verbose = FALSE,
  catch.errors = TRUE,
  from.knitr = isTRUE(getOption("knitr.in.progress")) | !interactive(),
  use.null.device = TRUE,
  just.init = FALSE,
  stud.code = NULL
)
```

---

check.regression — *Check whether an object from a call to lm, glm or some other regression function is correct*

Description

Check whether an object from a call to lm, glm or some other regression function is correct

Usage

```
check.regression(
  var,
  str.expr,
  part = NULL,
  ps = get.ps(),
  stud.env = ps$stud.env,
  verbose = FALSE,
  failure.message = paste0("Hmm... your regression ", var, " seems incorrect."),
  success.message = paste0("Great, your regression ", var, " looks correct."),
  tol = 1e-10
)
```

check.variable *Test: Check whether a variable is equal to a specified expression*

Description

Test: Check whether a variable is equal to a specified expression

Usage

```
check.variable(
  var,
  expr,
  length = check.all,
  dim = check.all,
  class = check.all,
  values = check.all,
  check.all = TRUE,
  ...,
  tol = .Machine$double.eps^0.5,
  failure.exists = "You have not yet generated the variable {{var}}.",

   failure.length = "Your variable {{var}} has length {{length_stud}} but it shall have length {{length_
  failure.dim = "Your variable {{var}} has the wrong dimensions (rows x columns).",

   failure.class = "Your variable {{var}} has a wrong class. It should be {{class_sol}} but it is {{class
  failure.values = "Your variable {{var}} has wrong values.",
  success.message = "Great, {{var}} has correct {{tests}}.",
  ps = get.ps(),
  stud.env = ps$stud.env,
  verbose = FALSE,
  part = NULL
)
```

Arguments

var	a the variable name as string
expr	an expression that will be evaluated in the student environment and returns the variable
length	shall length be checked (similar dim, class, values)
failure.length	a message that is shown if the variable does not exists (similar the other failure.??? variables)
failure.message.add	a text that will be added to all failure messages

create.ps *Generate a problem set from a solution file*

Description

Generates .rps file, and .rmd files for empty ps , sample solution and output solution

Usage

```
create.ps(
  sol.file,
  ps.name = NULL,
  user.name = "ENTER A USER NAME HERE",
  sol.user.name = "Jane Doe",
  dir = getwd(),
  header = "",
  footer = "",
  libs = NULL,
  stop.when.finished = FALSE,
  extra.code.file = NULL,
  var.txt.file = NULL,
  rps.has.sol = TRUE,
  fragment.only = TRUE,
  add.enter.code.here = FALSE,
  add.shiny = TRUE,
  make.rmd = TRUE,
  addons = NULL,
  whitelist.report = FALSE,
  wl = rtutor.default.whitelist(),
  use.memoise = FALSE,
  memoise.funs = rtutor.default.memoise.funs(),
  precomp = FALSE,
  preknit = FALSE,
  force.noeval = FALSE,
  html.data.frame = TRUE,
  table.max.rows = 25,
  round.digits = 8,
  signif.digits = 8,
  knit.print.opts = make.knit.print.opts(html.data.frame = html.data.frame,
   table.max.rows = table.max.rows, round.digits = round.digits, signif.digits =
    signif.digits),
  knitr.opts.chunk = list(dev = "svg"),
  e.points = 1,
  min.chunk.points = 0,
  chunk.points = 0,
  keep.fill.in.output.sol = TRUE,
  hint.on.fail = FALSE,
```

```
  empty.task.txt = "# Enter your code here.",
  placeholder = "___",
  short.first.chunk = TRUE,
  stop.if.visual.mode.garbling = TRUE,
  bolden.part.counters = FALSE
)
```

Arguments

sol.file	file name of the _sol.rmd file that specifies the problem set
ps.name	the name of the problem set
user.name	can pick a default user.name (will typically not be set)
sol.user.name	the user.name set in the sample solution
dir	the directory in which all files are found and wil be saved to
libs	character vector with names of libraries that will be used by the problem set
extra.code.file	the name of an r file that contains own functions that will be accessible in the problme set
var.txt.file	name of the file that contains variable descriptions (see thee vignette for an explanation of the file format)
rps.has.sol	shall the sample solution be stored in the .rps file. Set this option to FALSE if you use problem sets in courses and don't want to assess students the sample solution easily
add.shiny	shall we compile the ps so that it can be shown as a web-based shiny app. Default is TRUE. Set FALSE if not needed to speed up compilation
make.rmd	Shall a Rmd problem set file and sample solution file be generated. Default is TRUE You can set to FALSE if you only want a shiny version to slightly speed up compilation and avoid file clutter.
use.memoise	shall functions like read.csv be memoised? Data sets then only have to be loaded once. This can make problem sets run faster. Debugging may be more complicated, however.
memoise.funs	character vector of function names that will be memoised when use.memoise = TRUE. By default a list of functions that load data from a file.
precomp	shall chunk environments be computed from sample solution when problem set is generated? Default = FALSE
preknit	shall sample solution of chunks be knitted when problem set is generated. Default = FALSE
force.noeval	shall problem set only be shown in noeval mode? (Used as a security against accidentially forgetting to set noeval=TRUE in show.ps, when showing the problem set in a web app.)
html.data.frame	shall data frames in shiny-based problem set be shown as html? Default is TRUE.
table.max.rows	How many rows of a printed data frame shall be shown? Default is 25.

`round.digits` Digits for rounding of shown data frames.

`signif.digits` Significant digits for shown data frames.

`knitr.opts.chunk`
A list of global knitr chunk options for shiny problem set, see https://yihui.org/knitr/options/. By default `list(dev="svg")`. Has the same effect as if you would call `knitr::opts_chunk` with those options before you call `show.ps`.

`e.points` how many points does the user get per required expression in a chunk (expressions in a task do not count). Default=1

`min.chunk.points`
minimal points for checking a chunk even if no none-task expression has to be entered. By default=0.5. I feel there may be a higher motivation to continue a problem set if there are may be some free point chunks farther below. Also it feels nice to get points, even if it is just for pressing the check button.

`chunk.points` you may also specify fixed points given for solving a chunk that will be added to the points per expression. Default=0

`keep.fill.in.output.sol`
if TRUE (default) the original code with placeholders of a fill in block will be shown in the output solution Rmd file as a comment before the solution. If FALSE only the solution will be shown.

`hint.on.fail` shall by default a hint be shown already if the correctness test fails. If FALSE (default) hints are only shown if the user types hint() or, in the shiny version, presses the hint button.

`empty.task.txt` A text that will be shown in chunks without any task block. Default is `empty.task.txt = "# Enter your code here."`

`placeholder` The string you use as placeholder in fill_in blocks. By default `"___"`. This should be a pattern that you don't use in your normal code. If a user's input cannot be parsed, we replace this pattern by an internal represenataion that is valid R syntax.

`short.first.chunk`
If `TRUE` (default) the first chunk is more compact and only contains the user name line. Otherwise it also contains the calls to check.problem.set which would allow to check the problem set also without the RStudio Addin.

`stop.if.visual.mode.garbling`
If `TRUE` (default) stops the creation of the problem set and shows an informative error message if it looks as if the solution file was shown in the new visual mode for markdown files. The new visual mode markdown feature of RStudio is cool for solving RTutor problem sets. But you should never view the solution file from which you generate the problem set in visual mode, since it rewrites the code in an unparseable way.

`bolden.part.counters`
if TRUE change lines that start with a) or b) etc to **a)** and **b)**. This turns-off auto enumeration and makes problem sets look nicer in visual markdown mode.

display *Displays the given text*

Description

Displays the given text

Usage

```
display(..., collapse = "\n", sep = "", start.char = "\n", end.char = "\n")
```

get.ps *Get the current problem set*

Description

Either a globally stored problem set or if RTutor runs as a web-app the associated problem set with the current shiny session

Usage

```
get.ps(force.global = FALSE)
```

hint *Shows a hint for the current problem.*

Description

Shows a hint for the current problem.

Usage

```
hint(..., ps = get.ps())
```

hint.else *Show a hint only if no hint.stud.call or hint.stud.assign was triggered.*

Description

It says that the automatic hint should be shown unless some hint with hint.stud.call has been shown (or ps$shown.custom.hints has been manually assigned a value above 0.)

Usage

```
hint.else(msg, add.line.breaks = TRUE, ps = get.ps())
```

hint.else.active *Get or set whether hint.else or auto.hint.else would be triggered.*

Description

If a hint.stud.call or hint.stud.assign is shown then a hint.else or auto.hint.else would not be triggered. This function returns TRUE if hint.else would still be triggered or otherwise FALSE.

Usage

```
hint.else.active(activate = NULL, ps = get.ps())
```

Details

If you set the argument activate you can change this status.

hint.for.assign *Default hint for an assignment*

Description

Default hint for an assignment

Usage

```
hint.for.assign(
  expr,
  ps = get.ps(),
  env = ps$stud.env,
  stud.expr.li = ps$stud.expr.li,
  part = ps$part,
  s3.method = NULL,
  expr.object = NULL,
  start.char = "\n",
  end.char = "\n",
  ...
)
```

hint.for.call *Default hint for a call*

Description

Default hint for a call

Usage

```
hint.for.call(
  call,
  ps = get.ps(),
  env = ps$stud.env,
  stud.expr.li = ps$stud.expr.li,
  part = ps$part,
  from.assign = !is.null(lhs),
  lhs = NULL,
  call.obj = NULL,
  s3.method = NULL,
  start.char = "\n",
  end.char = "\n"
)
```

hint.for.compute *Default hint for a compute block*

Description

Default hint for a compute block

Usage

```
hint.for.compute(
  expr,
  hints.txt = NULL,
  var = "",
  ps = get.ps(),
  env = ps$stud.env,
  stud.expr.li = ps$stud.expr.li,
  part = ps$part,
  start.char = "\n",
  end.char = "\n",
  ...
)
```

hint.for.function *Default hint for a function*

Description

Default hint for a function

Usage

```
hint.for.function(code, ..., ps = get.ps())
```

hint.stud.assign *Show the hint if the student made the specified wrong assignment*

Description

Show the hint if the student made the specified wrong assignment

Usage

```
hint.stud.assign(var, call, msg, ps = get.ps(), env = parent.frame())
```

Arguments

var	name of the to be assigned variable as character
call	an unquoted call that we check whether the student makes it
msg	a string that shall be shown as hint if the student made the call in his code

hint.stud.call *Show the hint if the student made the specified wrong call*

Description

Show the hint if the student made the specified wrong call

Usage

```
hint.stud.call(
  call,
  msg = "",
  ps = get.ps(),
  env = parent.frame(),
  qcall,
  var = NULL
)
```

Arguments

call	an unquoted call that we check whether the student makes it
msg	a string that shall be shown as hint if the student made the call in his code

---

hint.stud.fun *Show the hint if the student calls a specific function*

Description

Show the hint message if the student has called a certain function (not nested in another function) somewhere in the chunk. If you also want to consider the call arguments use hint.stud.call or hint.stud.assign instead.

Usage

```
hint.stud.fun(fun.name, msg, ps = get.ps(), env = parent.frame())
```

Arguments

fun.name	the function name as string.
msg	a string that shall be shown as hint if the student made the call in his code

---

holds.true *To be used in a test block*

Description

Checks whether a certain condition on the stud's generated variables hold true

Usage

```
holds.true(
  cond,
  short.message = failure.message,
  failure.message = "Failure in holds.true",
  success.message = "Great, the condition {{cond}} holds true in your solution!",
  part = NULL,
  ps = get.ps(),
  stud.env = ps$stud.env,
  cond.str = NULL,
  ...
)
```

Arguments

cond	The condition to be checked
failure.message	The failure message to be shown if the text fails.
success.message	The success message

---

make.hint.report *Helper function when developing problem sets*

Description

Tries to check all chunks with given solution and shows the corresponding hint.

Usage

```
make.hint.report(ps.name, out.file = paste0(ps.name, "_hint_report.Rmd"))
```

---

make.submission *Grade your problem set and make submission file*

Description

The command will rerun and check all chunks of your problem set and grade it, i.e. it determines which tests are passed or not. The results are stored in a submission file: psname___username.sub, which will be part of the submitted solution. The function works similarly than check.problem.set, but makes sure that all exercies are checked.

Usage

```
make.submission(
  ps = get.ps(),
  user.name = get.user.name(),
  ps.name = ps$name,
  stud.path = ps$stud.path,
  stud.short.file = ps$stud.short.file,
  add.log = TRUE,
  reset = TRUE,
  set.warning.1 = TRUE,
  verbose = FALSE,
  catch.errors = TRUE,
  from.knitr = !interactive(),
  use.null.device = TRUE,
  ups.dir = ps$ups.dir
)
```

name.rmd.chunks *Set default names for the chunks of problem set rmd files*

Description

Set default names for the chunks of problem set rmd files

Usage

```
name.rmd.chunks(
  rmd.file = NULL,
  txt = readLines(rmd.file, warn = FALSE),
  only.empty.chunks = FALSE,
  keep.options = TRUE,
  valid.file.name = FALSE
)
```

Arguments

rmd.file	file name
txt	alternative the code as txt file
only.empy.chunks	if FALSE (default) name all chunks. Otherwise only empty chunks are overwritten
keep.option	if TRUE (default) don't change chunk options; otherwise clear all chunk options (dangerous)

read.yaml *Reads a yaml file and returns as a list*

Description

Reads a yaml file and returns as a list

Usage

```
read.yaml(
  file = NULL,
  verbose = FALSE,
  keep.quotes = TRUE,
  quote.char = "__QUOTE__",
  text = NULL,
  catch.error = TRUE,
  check.by.row = FALSE,
  space.after.colon = FALSE,
  utf8 = TRUE
)
```

`rtutor.app.skel` *Generate a skeleton for a shinyapps.io app of a problem set*

Description

Generate a skeleton for a shinyapps.io app of a problem set

Usage

```
rtutor.app.skel(
  ps.name,
  app.name = ps.name,
  app.dir,
  rps.app = !is.null(rps.dir),
  pkg.name = NULL,
  rps.file = paste0(ps.name, ".rps"),
  rps.dir = NULL,
  overwrite = FALSE,
  github.user = "GITHUB_USERNAME",
  rtutor.fork = "skranz",
  libs = NULL,
  ps.show.opts = NULL,
  direct.execution = FALSE,
  shinyapps.account.info = list(name = "<SHINYAPPS_USERNAME>", token = "<TOKEN>",
    secret = "<SECRET>"),
  ...
)
```

Arguments

`ps.name`	Name of the problem set
`app.name`	Name of your app. Should have no white spaces or special characters
`app.dir`	Your local directory to which you want to deploy your app files
`rps.app`	logical. If 'TRUE' create an app based on an .rps file. Otherwise create the app based on a problem set package that is hosted on Github.
`pkg.name`	If you create the app from a package this is the name of your package.
`rps.file`	The name of your rps file without directory if you create the app from a .rps file
`rps.dir`	the folder of your rps.file
`github.user`	If you create the app from a package this is the name of your Github user name.
`rtutor.fork`	Note that shinyapps.io only works with R packages directly installed from Github or CRAN. It is therefore not possible to locally change RTutor and use the adapted version for your own problemsets. This option however allows you to refer to your fork on github. Default is the main package under skranz.

ps.show.opts
: ps.show() arguments which are added to the generated ps.show. Has to be given as a named list, e.g. 'ps.show.opts=list(show.solution.btn=FALSE)' if one wants to create an app which does not show the solution button. By default only the necessary options are set. If those are provided, they are overwritten. This way, one can for example set the user.name to something different than Guest.

direct.execution
: If TRUE the generated file deployapp.R is directly executably in the sense that the safety checks within the file are off (i.e. the saving 'if' clauses are set to TRUE). Use with care! Default is 'FALSE'.

shinyapps.account.info
: Expects a List with the account info according to http://shiny.rstudio.com/articles/shinyapps.html . Default is list(name='<SHINYAPPS_USERNAME>', token='<TOKEN>',secret='<SECRET>'), i.e. the example from that site.

---

rtutor.package.skel *Generate a package skeleton for a shiny based RTutor problem set that shall be deployed as a package*

Description

Generate a package skeleton for a shiny based RTutor problem set that shall be deployed as a package

Usage

```
rtutor.package.skel(
  sol.file,
  ps.name,
  pkg.name,
  pkg.parent.dir,
  author = "AUTHOR_NAME",
  github.user = "GITHUB_USERNAME",
  date = format(Sys.time(), "%Y-%d-%m"),
  source.dir = getwd(),
  rps.file = paste0(ps.name, ".rps"),
  libs = NULL,
  extra.code.file = NULL,
  var.txt.file = NULL,
  ps.file = paste0(ps.name, ".Rmd"),
  overwrite = FALSE,
  overwrite.ps = TRUE,
  ...
)
```

`rtutor.skel.show.opts.string` *Intermediary Function helping to build the ps.show() Options string*

Description

Expects two lists with arguments.

Usage

```
rtutor.skel.show.opts.string(mandatory, optional)
```

Arguments

`mandatory`	Are always set but may be overwritten by optional
`optional`	Are intended to be set by the user. May overwrite mandatory ones if set explicitely.

`run.ps` *Run a problem set from a package in the browser*

Description

Only works if a package for the problem set is loaded. For problem sets stored in a local .rps file, use show.ps() instead

Usage

```
run.ps(
  user.name,
  ps.name = info$ps[1],
  dir = getwd(),
  package = NULL,
  auto.save.code = FALSE,
  clear.user = FALSE,
  run.solved = FALSE,
  sample.solution = FALSE,
  show.solution.btn = NA,
  launch.browser = TRUE,
  info = get.package.info(package),
  deploy.local = !make.web.app,
  make.web.app = FALSE,
  pkg.dir = path.package(info$package),
  rps.dir = find.pkg.rps.dir(ps.name, pkg.dir),
  material.dir = find.pkg.material.dir(ps.name, pkg.dir),
  ...
)
```

Arguments

user.name	Your user name
ps.name	Name of the problem set. By default the first problem set name of the loaded RTutor problem set package.
dir	your working directory for the problem set
package	name of the package that contains your problem set. Is automatically chosen a (single) package with an RTutor problem set is loaded.
auto.save.code	If TRUE all entered code will be automatically saved for the user. If FALSE (default) no entered code is saved. The user statistics (how many chunks are solved) are still saved, however.
clear.user	If TRUE all previously saved data for the user is removed if the problem set starts. Can be useful for developlmen or for resetting things.
run.solved	If TRUE and also sample.soulution=TRUE all previously solved chunks will be run when the problem set is newly shown. This can be very time consuming. I would suggest in most cases to keep the default run.solved=FALSE.
sample.solution	If TRUE the sample solution is shown in all chunks. Can be useful when developing a problem set. Note that one can create a problem set such that the sample solution is not available, e.g. if one wants to avoid that students just look at the sample solution.
show.solution.btn	If TRUE add a button to each chunk to show the sample solution. Note that one can create a problem set such that the sample solution is never available.
launch.browser	if TRUE (default) show the problem set in the browser. Otherwise it is shown in the RStudio viewer pane
pkg.dir	the package directory under which problem set files are searched under pkg.dir/ps/ps.name/. Will be set by default to currently loaded RTutorProblemSet package
rps.dir	directory of rps.files. Will be set to default for current package
material.dir	directory of additional problem set files. Will be set to default for current package
...	additional arguments of show.ps

---

sc *Like paste0 but returns an empty vector if some string is empty*

Description

Like paste0 but returns an empty vector if some string is empty

Usage

```
sc(..., sep = "", collapse = NULL)
```

show.ps *Run a shiny based problem set in the browser*

Description

Main function to locally run a shiny based problem set in your browser. There are a lot of parameters that control the behavior of the problem set. Only the main parameters are explained below.

Usage

```
show.ps(
  ps.name,
  user.name = "default_user",
  auto.save.code = FALSE,
  clear.user = FALSE,
  run.solved = FALSE,
  sample.solution = FALSE,
  prev.chunks.sample.solution = show.solution.btn,
  launch.browser = TRUE,
  catch.errors = TRUE,
  dir = getwd(),
  rps.dir = dir,
  offline = !can.connect.to.MathJax(),
  left.margin = 2,
  right.margin = 2,
  is.solved,
  make.web.app = FALSE,
  make.session.ps = make.web.app,
  save.nothing = FALSE,
  show.revert.btn = TRUE,
  show.solution.btn = NA,
  show.data.exp = TRUE,
  show.download.rmarkdown = TRUE,
  disable.graphics.dev = TRUE,
  check.whitelist = !is.null(wl),
  wl = NULL,
  verbose = FALSE,
  html.data.frame = TRUE,
  table.max.rows = 25,
  round.digits = 8,
  signif.digits = 8,
  knit.print.opts = make.knit.print.opts(html.data.frame = html.data.frame,
   table.max.rows = table.max.rows, round.digits = round.digits, signif.digits =
  signif.digits, print.data.frame.fun = print.data.frame.fun, print.matrix.fun =
    print.matrix.fun),
  print.data.frame.fun = NULL,
  print.matrix.fun = NULL,
```

```
  precomp = FALSE,
  noeval = FALSE,
  need.login = FALSE,
  login.dir = paste0(dir, "/login"),
  show.points = TRUE,
  stop.app.if.window.closes = !make.session.ps,
  sav.file = paste0(user.name, "_", ps.name, ".sav"),
  load.sav = FALSE,
  show.save.btn = FALSE,
  import.rmd = FALSE,
  rmd.file = paste0(ps.name, "_", user.name, "_export.rmd"),
  ...
)
```

Arguments

ps.name	Name of the problem set
user.name	A user name. Should be a valid variable name
auto.save.code	If TRUE all entered code will be automatically saved for the user. If FALSE (default) no entered code is saved. The user statistics (how many chunks are solved) are still saved, however.
clear.user	If TRUE all previously saved data for the user is removed if the problem set starts. Can be useful for development or for resetting things.
run.solved	If TRUE and also sample.solution=TRUE all previously solved chunks will be run when the problem set is newly shown. This can be very time consuming. I would suggest in most cases to keep the default run.solved=FALSE.
sample.solution	If TRUE the sample solution is shown in all chunks. Can be useful when developing a problem set. Note that one can create a problem set such that the sample solution is not available, e.g. if one wants to avoid that students just look at the sample solution.
prev.chunks.sample.solution	If TRUE and a user edits a chunk without having checked all previous chunks then previous chunks will be automatically be checked with the sample solution. If FALSE previous chunks will be checked with the user's entered solutions. Has by default the same value as show.solution.btn.
launch.browser	if TRUE (default) show the problem set in the browser. Otherwise it is shown in the RStudio viewer pane
catch.errors	by default TRUE only set FALSE for debugging purposes in order to get a more informative traceback()
dir	your working directory for the problem set, by default getwd()
rps.dir	directory of rps.files by default equal to dir
offline	(FALSE or TRUE) Do you have no internet connection. By default it is checked whether RTutor can connect to the MathJax server. If you have no internet connection, you cannot render mathematic formulas. If RTutor wrongly thinks you have an internet connection, while you don't, your chunks may not show at all. If you encounter this problem, set manually offline=TRUE.

show.solution.btn
: If TRUE add a button to each chunk to show the sample solution. Note that one can create a problem set such that the sample solution is never available. By default TRUE if a sample solution is available in the problem set.

show.download.rmarkdown
: If TRUE the user is able to download the R-Markdown file of their solution in the submissions-tab. If FALSE the corresponding button is not rendered. By default set to TRUE.

html.data.frame
: shall data.frames and matrices be printed as html table if a chunk is checked? (Default=TRUE)

table.max.rows
: the maximum number of rows that is shown if a data.frame is printed as html.table

round.digits
: the number of digits that printed data.frames shall be rounded to

---

stats *Shows your progress*

Description

Shows your progress

Usage

```
stats(
  do.display = TRUE,
  use.old.stats = FALSE,
  ups = get.ups(),
  ps = get.ps(),
  rps = ps$rps
)
```

---

test.H0 *Helper function for custom test blocks. Check whether a certain null hypothesis is not significantly rejected*

Description

Helper function for custom test blocks. Check whether a certain null hypothesis is not significantly rejected

Usage

```
test.H0(
  test.expr,
  p.value,
  test.name = "",
  alpha.warning = 0.05,
  alpha.failure = 0.001,
  short.message,
  warning.message,
  failure.message,

   success.message = "Great, I could not significantly reject the null hypothesis from the test '{{test_
  check.warning = TRUE,
  part = NULL,
  ps = get.ps(),
  stud.env = ps$stud.env,
  ...
)
```

Arguments

test.expr	an expression that calls a test which will be evaluated in stud.env. The test must return a list that contains a field "p.value"
p.value	Instead of providing test.expr, one can directly provide a p.value from a previously run test
test.name	an optional test.name that can be used to fill the test_name whiskers in warning or failure messages.
alpha.warning	default=0.05 a p.value below a warning is printed that the code may be wrong
alpha.failure	default=0.001 the critical p.value below which the stud code is considered wrong
short.message, failure.messages	warning.messages Messages in case of a failure and warning and short message for the log.file
check.warning	if FALSE don't check for a warning

Value

TRUE if H0 cannot be rejected, FALSE if not and "warning" if it can be weakly rejected

---

test.H0.rejected *Can be used in a custom test block. Checks whether a certain H0 can be significantly rejected*

Description

Can be used in a custom test block. Checks whether a certain H0 can be significantly rejected

Usage

```
test.H0.rejected(
  test.expr,
  p.value,
  test.name = "",
  alpha.warning = 0.01,
  alpha.failure = 0.05,
  short.message = "Fail to reject '{{test_name}}', p.value = {{p_value}}",

   warning.message = "The null hypothesis from the test '{{test_name}}', should not be rejcected, but I g

   failure.message = "I couldn't significantly reject the null hypothesis from the test '{{test_name}}'

   success.message = "Great, I could significantly reject the null hypothesis from the test '{{test_name
  check.warning = TRUE,
  ps = get.ps(),
  stud.env = ps$stud.env,
  part = NULL,
  ...
)
```

Arguments

test.expr	an expression that calls a test which will be evaluated in stud.env. The test must return a list that contains a field "p.value"
p.value	Instead of providing test.expr, one can directly provide a p.value from a previously run test
test.name	an optional test.name that can be used to fill the test_name whiskers in warning or failure messages.
alpha.warning	default=0.05 a p.value below a warning is printed that the code may be wrong
alpha.failure	default=0.001 the critical p.value below which the stud code is considered wrong
short.message, failure.messages	warning.messages Messages in case of a failure and warning and short message for the log.file
check.warning	if FALSE don't check for a warning

Value

TRUE if H0 can be rejected, FALSE if not and "warning" if it can be weakly rejected

true *A robust implementation of isTRUE*

Description

Returns FALSE if evaluation expr yields an error or is not TRUE.

Usage

```
true(expr, envir = parent.frame())
```

Details

Useful for customized hints were evaluating an expression may often cause errors, e.g. if a user did not define a variable.

with.random.seed *Calls a function with a specified random.seed*

Description

Calls a function with a specified random.seed

Usage

```
## S3 method for class 'random.seed'
with(expr, seed = 1234567890)
```

Arguments

expr	the expression to be evaluated
seed	the seed as integer

Index