Package ‘rchallenge’

January 19, 2020

Title A Simple Data Science Challenge System

Version 1.3.2

Description A simple data science challenge system using R Markdown and 'Dropbox' <https://www.dropbox.com/>.
It requires no network configuration, does not depend on external platforms
like e.g. 'Kaggle' <https://www.kaggle.com/> and can be easily installed on a personal computer.

URL https://adrtod.github.io/rchallenge

BugReports https://github.com/adrtod/rchallenge/issues

Depends R (>= 3.2.0)

Imports rmarkdown (>= 0.5.1), knitr (>= 1.6)

SystemRequirements pandoc (>= 1.12.3) -
http://johnmacfarlane.net/pandoc

License GPL-2

LazyData true

RoxygenNote 7.0.2

Encoding UTF-8

Suggests testthat (>= 2.1.0)

NeedsCompilation no

Author Adrien Todeschini [aut, cre],
Robin Genuer [ctb]

Maintainer Adrien Todeschini <adrien.todeschini@gmail.com>

Repository CRAN

Date/Publication 2020-01-19 18:30:03 UTC

R topics documented:

rchallenge-package — *rchallenge: A Simple Data Science Challenge System*

Description

A simple data science challenge system using R Markdown and 'Dropbox' <https://www.dropbox.com/>. It requires no network configuration, does not depend on external platforms like e.g. 'Kaggle' <https://www.kaggle.com/> and can be easily installed on a personal computer.

Installation

Install the R package from CRAN repositories

```
install.packages("rchallenge")
```

or install the latest development version from GitHub

```
# install.packages("devtools")
devtools::install_github("adrtod/rchallenge")
```

A recent version of pandoc (>= 1.12.3) is also required. See the pandoc installation instructions for details on installing pandoc for your platform.

Getting started

Install a new challenge in `Dropbox/mychallenge`:

```
setwd("~/Dropbox/mychallenge")
library(rchallenge)
new_challenge()
```

or for a french version:

```
new_challenge(template = "fr")
```

You will obtain a ready-to-use challenge in the folder `Dropbox/mychallenge` containing:

- `challenge.rmd`: Template R Markdown script for the webpage.
- `data`: Directory of the data containing `data_train` and `data_test` datasets.
- `submissions`: Directory of the submissions. It will contain one subdirectory per team where they can submit their submissions. The subdirectories are shared with Dropbox.
- `history`: Directory where the submissions history is stored.

The default challenge provided is a binary classification problem on the German Credit data set.

You can easily customize the challenge in two ways:

- *During the creation of the challenge*: by using the options of the `new_challenge` function.
- *After the creation of the challenge*: by manually replacing the data files in the `data` subdirectory and the baseline predictions in `submissions/baseline` and by customizing the template `challenge.rmd` as needed.

Next steps

To complete the installation:

1. Create and share subdirectories in `submissions` for each team:

   ```
   new_team("team_foo","team_bar")
   ```

2. Render the HTML page: `publish()` Use the `output_dir` argument to change the output directory. Make sure the output HTML file is rendered, e.g. using GitHub Pages.
3. Give the URL to your HTML file to the participants.
4. Refresh the webpage by repeating step 2 on a regular basis. See below for automating this step.

From now on, a fully autonomous challenge system is set up requiring no further administration. With each update, the program automatically performs the following tasks using the functions available in our package:

- `store_new_submissions`: Reads submitted files and save new files in the history.
- `print_readerr`: Displays any read errors.
- `compute_metrics`: Calculates the scores for each submission in the history.
- `get_best`: Gets the highest score per team.
- `print_leaderboard`: Displays the leaderboard.
- `plot_history`: Plots a chart of score evolution per team.
- `plot_activity`: Plots a chart of activity per team.

Automating the updates on Unix/OSX

For the step 4, you can setup the following line to your crontab using `crontab -e` (mind the quotes):

```
0 * * * * Rscript -e 'rchallenge::publish("~/Dropbox/mychallenge/challenge.rmd")'
```

This will render a HTML webpage every hour. Use the `output_dir` argument to change the output directory.

If your challenge is hosted on a Github repository you can automate the push:

```
0 * * * * cd ~/Dropbox/mychallenge && Rscript -e 'rchallenge::publish()' && git commit
-m "update html" index.html && git push
```

You might have to add the path to Rscript and pandoc at the beginning of your crontab:

```
PATH=/usr/local/sbin:/usr/local/bin:/sbin:/bin:/usr/sbin:/usr/bin
```

Depending on your system or pandoc version you might also have to explicitly add the encoding option to the command:

```
0 * * * * Rscript -e 'rchallenge::publish("~/Dropbox/mychallenge/challenge.rmd",encoding
= "utf8")'
```

Automating the updates on Windows

You can use the Task Scheduler to create a new task with a *Start a program* action with the settings (mind the quotes):

- *Program/script*: `Rscript.exe`
- *options*: `-e rchallenge::publish('~/Dropbox/mychallenge/challenge.rmd')`

Examples

- Credit approval (in french) by Adrien Todeschini (Bordeaux).
- Spam filter (in french) by Marie Chavent (Bordeaux).

Please contact me to add yours.

Note

The rendering of HTML content provided by Dropbox will be discontinued from the 3rd October 2016 for Basic users and the 1st September 2017 for Pro and Business users. See `https://www.dropbox.com/help/16`. Alternatively, GitHub Pages provide an easy HTML publishing solution via a simple GitHub repository.

version 1.16 of pandoc fails to fetch font awesome css, see `https://github.com/jgm/pandoc/issues/2737`.

Author(s)

Maintainer: Adrien Todeschini `<adrien.todeschini@gmail.com>`

Other contributors:

- Robin Genuer `<robin.genuer@isped.u-bordeaux2.fr>` [contributor]

See Also

Useful links:

- https://adrtod.github.io/rchallenge
- Report bugs at https://github.com/adrtod/rchallenge/issues

`compute_metrics` *Compute metrics of the submissions in the history.*

Description

Compute metrics of the submissions in the history.

Usage

```
compute_metrics(hist_dir = "history", metrics, y_test, ind_quiz, read_fun)
```

Arguments

`hist_dir` string. directory where the history of the submissions are stored. contains one subdirectory per team.

`metrics` named list of functions. Each function in the list computes a performance criterion and is defined as: `function(y_pred,y_test)`

`y_test` character or numeric vector. the test set output.

`ind_quiz` indices of `y_test` in the quiz subset.

`read_fun` function that reads a submission file and returns a vector of predictions.

Value

`compute_metrics` returns a named list with one named member per team. Each member is a `data.frame` where the rows are the submission files sorted by date and the columns are:

`date` the date of the submission

`file` the file name of the submission

`<metric name>.quiz`
the score obtained on the quiz subset

`<metric name>.test`
the score obtained on the test set

`countdown` *Countdown before deadline.*

Description

Countdown before deadline.

Usage

```
countdown(deadline, complete_str = intToUtf8(10004))
```

Arguments

`deadline`	POSIXct. deadline
`complete_str`	string. displayed when deadline is passed

`data_split` *Split a data.frame into training and test sets.*

Description

Split a data.frame into training and test sets.

Usage

```
data_split(
  data = get_data(”german”),
  varname = ”credit_risk”,
  p_test = 0.2,
  p_quiz = 0.5
)
```

Arguments

`data`	data.frame
`varname`	string. output variable name
`p_test`	real. proportion of samples in the test set
`p_quiz`	real. proportion of samples from the test set in the quiz set

Value

list with members

`train`	training set with output variable
`test`	test set without output variable
`y_test`	test set output variable
`ind_quiz`	indices of quiz samples in the test set

`german` *German Credit Data.*

Description

Data from Dr. Hans Hofmann of the University of Hamburg.

Usage

```
data(german)
```

Format

A `data.frame` with 1000 rows and 21 variables

Details

These data have two classes for the credit worthiness: Good or Bad. There are predictors related to attributes, such as: checking account status, duration, credit history, purpose of the loan, amount of the loan, savings accounts or bonds, employment duration, Installment rate in percentage of disposable income, personal information, other debtors/guarantors, residence duration, property, age, other installment plans, housing, number of existing credits, job information, Number of people being liable to provide maintenance for, telephone, and foreign worker status.

This is a transformed version of the German Credit data set with factors instead of dummy variables, and corrected as proposed by Groemping, U. (2019).

Source

UCI Machine Learning Repository `https://archive.ics.uci.edu/ml/datasets/Statlog+(German+Credit+Data)` `http://www1.beuth-hochschule.de/FB_II/reports/Report-2019-004.pdf`

References

Groemping, U. (2019). South German Credit Data: Correcting a Widely Used Data Set. Report 4/2019, Reports in Mathematics, Physics and Chemistry, Department II, Beuth University of Applied Sciences Berlin.

get_best *Get the best submissions per team.*

Description

Get the best submissions per team.

Usage

```
get_best(
  history,
  metrics = names(metrics),
  test_name = "quiz",
  decreasing = FALSE
)
```

Arguments

`history`	list of the submissions history per team as returned by `compute_metrics`
`metrics`	character vector. names of the metrics
`test_name`	string. name of the test set used: `"quiz"` or `"test"`
`decreasing`	logical. Should the sort order be increasing or decreasing? Must be of length 1 or with the same length as `metrics`.

Value

`get_best` returns a `data.frame` where the rows are teams in sorted order of performance. The best submission per team is retained. The sort is based on possibly several metrics in the order given by the `metrics` argument. In case of ties on the first metric, the second metric is used to break the ties, and so on. Lastly, the date is used in case of ties. The columns are:

`team`	name of the team
`n_submissions`	total number of submissions
`date`	the date of the best submission
`file`	the file name of the best submission
`<metric name>.quiz`	the score obtained on the quiz subset
`<metric name>.test`	the score obtained on the test set
`rank`	the rank of the team
`rank_diff`	the rank difference is set to 0 temporarily.

`get_data` *Get dataset value.*

Description

Get dataset value.

Usage

```
get_data(name = "german", package = "rchallenge", envir = environment(), ...)
```

Arguments

`name` string. name of the dataset.

`package` string. name of the package to look in for dataset.

`envir` the environment where the data should be loaded.

`...` additional arguments to be passed to `data`.

Value

The value of the dataset

See Also

`data`, `base`

`html_img` *HTML code for an image.*

Description

HTML code for an image.

Usage

```
html_img(file, width = "10px")
```

Arguments

`file` string. image file.

`width` string. width of display.

icon *HTML code for icons.*

Description

Currently only supports Font Awesome icons.

Usage

```
icon(name)
```

Arguments

`name`	string. name of the icon. You can see a full list of options at http://fontawesome.io/icons/.

Value

string containing the HTML code.

Note

Requires the Font Awesome HTML code: `<link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/font-a`

Examples

~~~
rmd <- '
```{r}
library(rchallenge)
```
<link rel="stylesheet"
 href="https://maxcdn.bootstrapcdn.com/font-awesome/4.6.3/css/font-awesome.min.css">
`r icon("fa-user")`
`r icon("fa-user fa-lg")`
`r icon("fa-user fa-2x")`
`r icon("fa-user fa-3x")`
`r icon("fa-user fa-3x fa-border")`
'
file <- tempfile()
cat(rmd, file=file)
writeLines(readLines(file))
rmarkdown::render(file)
~~~

last_update *Formatted last update date before deadline.*

Description

Formatted last update date before deadline.

Usage

```
last_update(deadline, format = "%d %b %Y %H:%M")
```

Arguments

deadline	POSIXct. deadline
format	string. see format.POSIXct

new_challenge *Install a new challenge.*

Description

Install a new challenge.

Usage

```
new_challenge(
  path = ".",
  out_rmdfile = "challenge.rmd",
  recursive = FALSE,
  overwrite = recursive,
  quiet = FALSE,
  showWarnings = FALSE,
  template = c("en", "fr"),
  data_dir = "data",
  submissions_dir = "submissions",
  hist_dir = "history",
  install_data = TRUE,
  baseline = "baseline",
  add_baseline = install_data,
  clear_history = overwrite,
  title = "Challenge",
  author = "",
  date = "",
  email = "EDIT_EMAIL@DOMAIN.com",
  date_start = format(Sys.Date(), "%d %b %Y"),
  deadline = paste(Sys.Date() + 90, "23:59:59"),
  data_list = data_split(get_data("german"))
)
```

Arguments

path string. install path of the challenge (should be somewhere in your Dropbox).

out_rmdfile string. name of the output R Markdown file.

recursive logical. should elements of the path other than the last be created? see dir.create.

overwrite logical. should existing destination files be overwritten? see file.copy.

quiet logical. deactivate text output.

showWarnings logical. should the warnings on failure be shown? see dir.create.

template string. name of the template R Markdown script to be installed. Two choices are available: "en" (english) and "fr" (french).

data_dir string. subdirectory of the data.

submissions_dir string. subdirectory of the submissions. see store_new_submissions.

hist_dir string. subdirectory of the history. see store_new_submissions.

install_data logical. activate installation of the data files of the template challenge.

baseline string. name of the team considered as the baseline.

add_baseline logical. activate installation of baseline submission files of the template challenge.

clear_history logical. activate deletion of the existing history folder.

title string. title displayed on the webpage.

author string. author displayed on the webpage.

date string. date displayed on the webpage.

email string. email of the challenge administrator.

date_start string. start date of the challenge.

deadline string. deadline of the challenge.

data_list list with members train, test, y_test and ind_quiz such as returned by the data_split function.

Value

The path of the created challenge is returned.

Examples

```
path <- tempdir()
wd <- setwd(path)
# english version
new_challenge()
# french version
new_challenge(template = "fr")
setwd(wd)
unlink(path)
```

new_team *Create new teams submission folders in your challenge.*

Description

Create new teams submission folders in your challenge.

Usage

```
new_team(
  ...,
  path = ".",
  submissions_dir = "submissions",
  quiet = FALSE,
  showWarnings = FALSE
)
```

Arguments

`...`	strings. names of the team subdirectories.
`path`	string. root path of the challenge. see `new_challenge`.
`submissions_dir`	string. subdirectory of the submissions. see `new_challenge`.
`quiet`	logical. deactivate text output.
`showWarnings`	logical. should the warnings on failure be shown? see `dir.create`.

Value

The paths of the created teams are returned.

Examples

```
path <- tempdir()
wd <- setwd(path)
new_challenge()
new_team("team_foo", "team_bar")
setwd(wd)
unlink(path)
```

plot_activity *Plot the density of submissions over time.*

Description

Plot the density of submissions over time.

Usage

```
plot_activity(
  history,
  baseline = "baseline",
  col = 1:length(history),
  alpha.f = 0.7,
  bw = 3600 * 24,
  by = 4,
  xlab = "Date",
  ylab = "Submissions intensity",
  bty = "l",
  fg = "darkslategray",
  col.axis = fg,
  col.lab = fg,
  text.col = fg,
  ...
)
```

Arguments

`history`	list of the submissions history per team as returned by `compute_metrics`
`baseline`	string. name of the team considered as the baseline that will not be plotted.
`col`	colors of the teams.
`alpha.f`	factor modifying the opacity alpha of colors; typically in [0,1].
`bw`	real. the smoothing bandwidth to be used by `density` in seconds.
`by`	real. height of the interval between two teams in nb of submissions.
`xlab, ylab`	axis labels. see `title`.
`bty, fg, col.axis, col.lab`	graphical parameters. see `par`.
`text.col`	the color used for the legend text. see `legend`.
`...`	further parameters passed to `plot` function.

Value

`NULL`

See Also

density

`plot_history` *Plot the history of the scores of each team over time.*

Description

The best score of each team has a bold symbol.

Usage

```
plot_history(
  history,
  metric,
  test_name = "quiz",
  baseline = "baseline",
  col = 1:length(history),
  pch = rep(21:25, 100),
  by = 0.05,
  xlab = "Date",
  ylab = "Score",
  bty = "l",
  fg = "darkslategray",
  col.axis = fg,
  col.lab = fg,
  text.col = fg,
  ...
)
```

Arguments

`history` list of the submissions history per team as returned by `compute_metrics`

`metric` string. name of the metric considered

`test_name` string. name of the test set used: `"quiz"` or `"test"`

`baseline` string. name of the team considered as the baseline. Its best score will be plotted as a constant and will not appear in the legend.

`col` colors of the teams

`pch` symbols of the teams

`by` real. interval width of grid lines

`xlab, ylab` axis labels. see `title`.

`bty, fg, col.axis, col.lab`
graphical parameters. see `par`.

`text.col` the color used for the legend text. see `legend`.

`...` further parameters passed to `plot` function.

Value

NULL

print_leaderboard *Format the leaderboard in Markdown.*

Description

Format the leaderboard in Markdown.

Usage

```
print_leaderboard(
  best,
  metrics = names(metrics),
  test_name = "quiz",
  digits = 3,
  ...
)
```

Arguments

best	list of the best submissions per team and per metric as returned by get_best.
metrics	character vector. names of the metrics to be displayed
test_name	string. name of the test set used: "quiz" or "test"
digits	integer. how many significant digits are to be used for metrics.
...	further parameters to pass to kable

Value

print_leaderboard returns a character vector of the table source code to be used in a Markdown document.

Note

Chunk option results='asis' has to be used

See Also

kable

`print_readerr` *Format read errors in Markdown.*

Description

Format read errors in Markdown.

Usage

```
print_readerr(read_err = list(), ...)
```

Arguments

`read_err`	list of read errors returned by `store_new_submissions`
`...`	further parameters to pass to `kable`

Value

`print_readerr` returns a character vector of the table source code to be used in a Markdown document.

`publish` *Render your challenge R Markdown script to a HTML page.*

Description

Render your challenge R Markdown script to a HTML page.

Usage

```
publish(
  input = "challenge.rmd",
  output_file = "index.html",
  output_dir = dirname(input),
  quiet = FALSE,
  ...
)
```

Arguments

`input`	string. name of the R Markdown input file
`output_file`	string. output file. If `NULL` then a default based on the name of the input file is chosen.
`output_dir`	string. output directory. Defaults to the directory of the input file. make sure that the output HTML file will be published online.
`quiet`	logical. deactivate text output.
`...`	further arguments to pass to `render`.

Value

The compiled document is written into the output file, and the path of the output file is returned.

Note

The rendering of HTML content provided by Dropbox will be discontinued from the 3rd October 2016 for Basic users and the 1st September 2017 for Pro and Business users. See https://www.dropbox.com/help/16. Alternatively, GitHub Pages provide an easy HTML web publishing solution via a simple GitHub repository.

See Also

render

Examples

```
path <- tempdir()
wd <- setwd(path)
new_challenge()
outdir = tempdir()
publish(output_dir = outdir, output_options = list(self_contained = FALSE))
unlink(outdir)
setwd(wd)
unlink(path)
```

---

rchallenge-defunct *Defunct functions in package ‘rchallenge’*

Description

These functions are defunct and no longer available.

Usage

```
glyphicon(...)
```

Arguments

...	parameters

Details

Defunct functions are: glyphicon

store_new_submissions *Store new submission files.*

Description

store_new_submissions copies new files from the subdirectories of submissions_dir to the respective subdirectories of hist_dir. Each team has a subdirectory. The copied files in hist_dir are prefixed with the last modification date for uniqueness. A file is considered new if its name and last modification time is new, i.e not present in hist_dir. The files must match pattern regular expression and must not throw errors or warnings when given to the valid_fun function.

Usage

```
store_new_submissions(
  submissions_dir = "submissions",
  hist_dir = "history",
  deadline,
  pattern = ".*\\.csv$",
  valid_fun
)
```

Arguments

submissions_dir	string. directory of the submissions. contains one subdirectory per team
hist_dir	string. directory where to store the history of the submissions. contains one subdirectory per team
deadline	POSIXct. deadline time for submissions. The files with last modification date after the deadline are skipped.
pattern	string. regular expression that new submission files must match (with ignore.case=TRUE)
valid_fun	function that reads a submission file and throws errors or warnings if it is not valid.

Value

store_new_submissions returns a named list of errors or warnings catched during the process. Members named after the team names are lists with members named after the file that throws an error which contain the error object.

`update_rank_diff` *Update the rank differences of the teams.*

Description

Update the rank differences of the teams.

Usage

```
update_rank_diff(best_new, best_old)
```

Arguments

`best_new`	`data.frame` of the best submissions per team as returned by `get_best`.
`best_old`	old `data.frame` of the best submissions per team and per metric.

Value

`update_rank_diff` returns the input `data.frame` `best_new` with an updated column `rank_diff`

Index