The runcode package[*]

Haim Bar and HaiYing Wang
haim.bar@uconn.edu, haiying.wang@uconn.edu

October 4, 2020

Abstract

runcode is a LaTeXpackage that executes programming source codes (including all command line tools) from LaTeX, and embeds the results in the resulting pdf file. Many programming languages can be easily used and any command-line executable can be invoked when preparing the pdf file from a tex file.

It is recommended to use this package in the server-mode together with the Python talk2stat package. Currently, the server-mode supports Julia, MatLab, Python, and R. More languages will be added.

For more details and usage examples, refer to the package's github repository, at https://github.com/Ossifragus/runcode.

1 Installation

The package on CTAN can be installed automatically by your TEX software (e.g., MikTeX Update Wizard). You can also simply put the runcode.sty file in the LaTeX project folder. To use the package you have to enable the 'shell-escape' option when compiling a LaTeX document.

The server mode requires the *talk2stat* Python package. To install it from the command line, use: `pip3 install talk2stat`
The *talk2stat* source is available from https://pypi.org/project/talk2stat/. Note that Python version 3.8.* and up is required.

2 Usage

2.1 Load the package

```
\usepackage[options]{runcode}
```

Available options are:

- `julia`: start a *talk2stat* server* for Julia [https://julialang.org/].

[*] This document corresponds to runcode v1.0, dated 2020/10/04.

- `matlab`: start a *talk2stat* server* for MatLab [`https://www.mathworks.com/products/matlab.html`].
- `R`: start a *talk2stat* server* for R [`https://www.r-project.org/`].
- `run`: run source code, and store results in cache files.
- `cache`: use cached results.
- `stopserver`: stop the *talk2stat* server(s) when the pdf compilation is done.
- `nominted`: use the *fvextra* package [`https://ctan.org/pkg/fvextra`] instead of the *minted* package [`https://ctan.org/pkg/minted`] to show code (*fvextra* does not require Python's pygments package [`https://pygments.org/`], but it does not provide syntax highlights).

* Requires the Python package *talk2stat* to be installed.

2.2 Basic commands

`\runExtCode{Arg1}{Arg2}{Arg3}[Arg4]` runs an external code. The arguments are:

- `Arg1` is the executable program.
- `Arg2` is the source file name.
- `Arg3` is the output file name (with an empty value, the counter 'codeOutput' is used).
- `Arg4` controls whether to run the code. `Arg4` is optional with three possible values: if skipped or with empty value, the value of the global Boolean variable `runcode` as determined by the `run` option when loading the package, is used; if the value is set to 'run', the code will be executed; if set to 'cache' (or anything else), use cached results (see more about the cache below).

`\showCode{Arg1}{Arg2}[Arg3][Arg4]` shows the source code, using minted for a pretty layout or fvextra (if pygments is not installed).

- `Arg1` is the programming language.
- `Arg2` is the source file name.
- `Arg3` is the first line to show (optional with a default value 1).
- `Arg4` is the last line to show (optional with a default value of the last line).

`\includeOutput{Arg1}[Arg2]` is used to embed the output from executed code.

- `Arg1` is the output file name, and it needs to have the same value as that of `Arg3` in `\runExtCode`. If an empty value is given to `Arg1`, the counter 'codeOutput' is used.
- `Arg2` is optional and it controls the type of output with a default value 'vbox'

  - `vbox` (or skipped) = verbatim in a box.
  - `tex` = pure latex.
  - `inline` = embed result in text.

`\inln{Arg1}{Arg2}[Arg3]` is designed for simple calculations; it runs one command (or a short batch) and displays the output within the text.

- `Arg1` is the executable program or programming language.
- `Arg2` is the source code.
- `Arg3` is the output type.
  - `inline` (or skipped or with empty value) = embed result in text.
  - `vbox` = verbatim in a box.

2.3 Language specific shortcuts

`\runJulia[Arg1]{Arg2}{Arg3}[Arg4]` runs an external Julia code file.

- `Arg1` is optional and uses *talk2stat*'s Julia server by default.
- `Arg2`, `Arg3`, and `Arg4` have the same effects as those of the basic command `\runExtCode`.

`\inlnJulia[Arg1]{Arg2}[Arg3]` runs Julia source code (`Arg2`) and displays the output in line.

- `Arg1` is optional and uses the Julia server by default.
- `Arg2` is the Julia source code to run. If the Julia source code is wrapped between ‘‘‘ on both sides (as in the markdown grammar), then it will be implemented directly; otherwise the code will be written to a file on the disk and then be called.
- `Arg3` has the same effect as that of the basic command `\inln`.

`\runMatLab[Arg1]{Arg2}{Arg3}[Arg4]` runs an external MatLab code file.

- `Arg1` is optional and uses *talk2stat*'s MatLab server by default.
- `Arg2`, `Arg3`, and `Arg4` have the same effects as those of the basic command `\runExtCode`.

`\inlnMatLab[Arg1]{Arg2}[Arg3]` runs MatLab source code (`Arg2`) and displays the output in line.

- `Arg1` is optional and uses the MatLab server by default.
- `Arg2` is the MatLab source code to run. If the MatLab source code is wrapped between ‘‘‘ on both sides (as in the markdown grammar), then it will be implemented directly; otherwise the code will be written to a file on the disk and then be called.
- `Arg3` has the same effect as that of the basic command `\inln`.

`\runR[Arg1]{Arg2}{Arg3}[Arg4]` runs an external R code file.

- `Arg1` is optional and uses *talk2stat]*'s R server by default.

- `Arg2`, `Arg3`, and `Arg4` have the same effects as those of the basic command `\runExtCode`.

`\inlnR[Arg1]{Arg2}[Arg3]` runs R source code (`Arg2`) and displays the output in line.

- `Arg1` is optional and uses the R server by default.
- `Arg2` is the R source code to run. If the R source code is wrapped between ‘‘‘ on both sides (as in the markdown grammar), then it will be implemented directly; otherwise the code will be written to a file on the disk and then be called.
- `Arg3` has the same effect as that of the basic command `\inln`.

3 Contributing

We welcome your contributions to this package by opening issues on GitHub and/or making a pull request. We also appreciate more example documents written using runcode.