Package 'proffer'

July 26, 2021

Title Profile R Code and Visualize with 'Pprof'

Version 0.1.5

Encoding UTF-8

Language en-US

License MIT + file LICENSE

URL https://github.com/r-prof/proffer,
https://r-prof.github.io/proffer/

BugReports https://github.com/r-prof/proffer/issues

Description Like similar profiling tools,
the 'proffer' package automatically detects
sources of slowness in R code.
The distinguishing feature of 'proffer' is its utilization of
'pprof', which supplies interactive visualizations
that are efficient and easy to interpret.
Behind the scenes, the 'profile' package converts
native Rprof() data to a protocol buffer
that 'pprof' understands.
For the documentation of 'proffer',
visit <https://r-prof.github.io/proffer/>.
To learn about the implementations and methodologies of
'pprof', 'profile', and protocol buffers,
visit <https://github.com/google/pprof>.
<https://developers.google.com/protocol-buffers>,
and <https://github.com/r-prof/profile>, respectively.

Depends R (>= 3.3.0)

Imports cli (>= 2.0.0), pingr (>= 2.0.1), processx (>= 3.4.0), profile (>= 1.0), RProtoBuf (>= 0.4.14), utils, withr (>= 2.1.2)

Suggests testthat (>= 2.1.0)

SystemRequirements Graphviz (https://www.graphviz.org/), pprof (https://github.com/google/pprof)

RoxygenNote 7.1.1

NeedsCompilation no

Author William Michael Landau [aut, cre]
(<https://orcid.org/0000-0003-1878-3253>),
Eli Lilly and Company [cph]

Maintainer William Michael Landau <will.landau@gmail.com>

Repository CRAN

Date/Publication 2021-07-26 16:30:02 UTC

R topics documented:

`proffer-package` *proffer: profile R code with pprof*

Description

It can be challenging to find sources of slowness in large workflows, and the proffer package can help. Proffer runs R code and displays summaries to show where the code is slowest. Proffer leverages the pprof utility to create highly efficient, clear, easy-to-read interactive displays that help users find ways to reduce runtime. The package also contains helpers to convert profiling data to and from pprof format and visualize existing profiling data files. For documentation, visit https://r-prof.github.io/proffer/.

Author(s)

William Michael Landau <will.landau@gmail.com>

References

https://github.com/r-prof/proffer

Examples

```
# TBD
if (identical(Sys.getenv("PROFFER_EXAMPLES"), "true")) {
# Start a pprof virtual server in the background.
px <- pprof(replicate(1e2, sample.int(1e4)))
# Terminate the server.
px$kill()
}
```

`install_go` *Install pprof and Go on Linux*

Description

On Linux, this function actually installs Go, which comes with its own installation of `pprof`. On Mac and Windows, the function simply points the user to a link to download the installer. Assumes amd64 architecture.

Usage

```
install_go(destination = Sys.getenv("HOME"), version = "1.14", quiet = FALSE)
```

Arguments

`destination`	Only relevant to Linux, full path to the Go installation with the `pprof` and Go executables. Defaults to `Sys.getenv("HOME")`, which means the default Go installation path is `file.path(Sys.getenv("HOME"),"go")`. That means the Go binary will be at `file.path(Sys.getenv("HOME"),"go/bin/go")` and `pprof` will be at `file.path(Sys.getenv("HOME"),"go/pkg/tool/linux_amd64/pprof")`. You will need to set environment variables in your `.Renviron` file, e.g. PROFFER_PPROF_BIN=/home/you/go/pkg/tool/linux_amd64/pprof and PROFFER_GO_BIN=/home/you/go/ `usethis::edit_r_environ()` is helpful for this.
`version`	Character, a version string such as `"1.14"`.
`quiet`	Logical, whether to suppress console messages.

Details

On Linux, users will need to set the environment variables `PROFFER_PPROF_BIN` and `PROFFER_GO_BIN` using `usethis::edit_r_environ()`. Typically, if `destination` is /home/you, then typically those lines look like PROFFER_GO_BIN=/home/you/go/pkg/tool/linux_amd64/pprof PROFFER_PPROF_BIN=/home/you/go/bi

pprof *Profile R code and visualize with pprof.*

Description

Run R code and display profiling results in a local interactive pprof server. Results are collected with `record_pprof()`.

Usage

```
pprof(
  expr,
  host = "localhost",
  port = proffer::random_port(),
  browse = interactive(),
  verbose = TRUE,
  ...
)
```

Arguments

`expr`	R code to run and profile.
`host`	Host name. Set to `"localhost"` to view locally or `"0.0.0.0"` to view from another machine. If you view from another machine, the printed out URL will not be valid. For example, if `pprof()` or `serve_pprof()` prints "http://0.0.0.0:8080", then you need to replace 0.0.0.0 with your computer's name or IP address, e.g. "http://my_computer.com:8080".
`port`	Port number for hosting the local pprof server. Chosen randomly by default.
`browse`	Logical, whether to open a browser to view the pprof server.
`verbose`	Logical, whether to print console messages such as the URL of the local `pprof` server.
`...`	Additional arguments passed on to `Rprof()` via `record_pprof()`.

Value

A `processx::process$new()` handle. Use this handle to take down the server with $kill().

Examples

```
if (identical(Sys.getenv("PROFFER_EXAMPLES"), "true")) {
# Start a pprof virtual server in the background.
px <- pprof(replicate(1e2, sample.int(1e4)))
# Terminate the server.
px$kill()
}
```

pprof_path *Show the path to the pprof executable.*

Description

Defaults to the PROFFER_PPROF_BIN environment variable. Otherwise, it searches your Go lang installation for pprof.

Usage

```
pprof_path()
```

Details

See https://github.com/r-prof/proffer#installation for setup instructions.

Value

Character, path to pprof it exists and "" otherwise.

Examples

```
if (identical(Sys.getenv("PROFFER_EXAMPLES"), "true")) {
pprof_path()
}
```

pprof_sitrep *Verify pprof installation*

Description

Check if pprof and its dependencies are installed.

Usage

```
pprof_sitrep()
```

Examples

```
pprof_sitrep()
```

random_port *Choose a random TCP port.*

Description

Choose a random TCP port that is unlikely to be occupied by another process.

Usage

```
random_port(lower = 49152L, upper = 65355L)
```

Arguments

lower	Integer of length 1, lower bound of the port number.
upper	Integer of length 1, upper bound of the port number.

Value

Port number, positive integer of length 1.

Examples

```
random_port()
```

record_pprof *Profile R code and record pprof samples.*

Description

Run R code and record pprof samples. Profiles are recorded with `record_rprof()` and then converted with `to_pprof()`.

Usage

```
record_pprof(expr, pprof = tempfile(), ...)
```

Arguments

expr	An R expression to profile.
pprof	Path to a file with pprof samples. Also returned from the function.
...	Additional arguments passed on to `Rprof()` via `record_rprof()`.

Value

Path to a file with pprof samples.

Examples

```
if (identical(Sys.getenv("PROFFER_EXAMPLES"), "true")) {
# Returns a path to pprof samples.
record_pprof(replicate(1e2, sample.int(1e4)))
}
```

`record_rprof` *Profile R code and record Rprof samples.*

Description

Run R code and record Rprof samples.

Usage

```
record_rprof(expr, rprof = tempfile(), ...)
```

Arguments

`expr`	An R expression to profile.
`rprof`	Path to a file with Rprof samples. Also returned from the function.
`...`	Additional arguments passed on to `Rprof()`.

Value

Path to a file with Rprof samples.

Examples

```
if (identical(Sys.getenv("PROFFER_EXAMPLES"), "true")) {
# Returns a path to Rprof samples.
record_rprof(replicate(1e2, sample.int(1e4)))
}
```

`serve_pprof` *Visualize profiling data with pprof.*

Description

Visualize profiling data with pprof.

Usage

```
serve_pprof(
  pprof,
  host = "localhost",
  port = proffer::random_port(),
  browse = interactive(),
  verbose = TRUE
)
```

Arguments

pprof	Path to pprof samples.
host	Host name. Set to "localhost" to view locally or "0.0.0.0" to view from another machine. If you view from another machine, the printed out URL will not be valid. For example, if pprof() or serve_pprof() prints "http://0.0.0.0:8080", then you need to replace 0.0.0.0 with your computer's name or IP address, e.g. "http://my_computer.com:8080".
port	Port number for hosting the local pprof server. Chosen randomly by default.
browse	Logical, whether to open a browser to view the pprof server.
verbose	Logical, whether to print console messages such as the URL of the local pprof server.

Details

Uses a local interactive server. Navigate a browser to a URL in the message. The server starts in a background process

Value

A processx::process$new() handle. Use this handle to take down the server with $kill().

Examples

```
if (identical(Sys.getenv("PROFFER_EXAMPLES"), "true")) {
pprof <- record_pprof(replicate(1e2, sample.int(1e4)))
# Start a pprof virtual server in the background.
px <- serve_pprof(pprof)
# Terminate the server.
px$kill()
}
```

serve_rprof *Visualize Rprof() output with pprof.*

Description

Use pprof to visualize profiling data produced by `Rprof()` or `record_rprof()`.

Usage

```
serve_rprof(
  rprof,
  host = "localhost",
  port = proffer::random_port(),
  browse = interactive(),
  verbose = TRUE
)
```

Arguments

rprof	Path to profiling samples generated by `Rprof()` or `record_rprof()`.
host	Host name. Set to `"localhost"` to view locally or `"0.0.0.0"` to view from another machine. If you view from another machine, the printed out URL will not be valid. For example, if `pprof()` or `serve_pprof()` prints "http://0.0.0.0:8080", then you need to replace 0.0.0.0 with your computer's name or IP address, e.g. "http://my_computer.com:8080".
port	Port number for hosting the local pprof server. Chosen randomly by default.
browse	Logical, whether to open a browser to view the pprof server.
verbose	Logical, whether to print console messages such as the URL of the local `pprof` server.

Details

Uses a local interactive server. Navigate a browser to a URL in the message. The server starts in a background process

Value

A `processx::process$new()` handle. Use this handle to take down the server with $kill().

Examples

```
if (identical(Sys.getenv("PROFFER_EXAMPLES"), "true")) {
rprof <- record_rprof(replicate(1e2, sample.int(1e4)))
# Start a pprof virtual server in the background.
px <- serve_rprof(rprof)
# Terminate the server.
px$kill()
}
```

test_pprof *Test* pprof()

Description

Do a test run of pprof() to verify that the system dependencies like pprof work as expected.

Usage

```
test_pprof(
  host = "localhost",
  port = proffer::random_port(),
  browse = interactive(),
  verbose = TRUE
)
```

Arguments

host	Host name. Set to "localhost" to view locally or "0.0.0.0" to view from another machine. If you view from another machine, the printed out URL will not be valid. For example, if pprof() or serve_pprof() prints "http://0.0.0.0:8080", then you need to replace 0.0.0.0 with your computer's name or IP address, e.g. "http://my_computer.com:8080".
port	Port number for hosting the local pprof server. Chosen randomly by default.
browse	Logical, whether to open a browser to view the pprof server.
verbose	Logical, whether to print console messages such as the URL of the local pprof server.

Details

See https://github.com/r-prof/proffer#installation for setup instructions.

See Also

pprof()

Examples

```
if (identical(Sys.getenv("PROFFER_EXAMPLES"), "true")) {
test_pprof()
}
```

to_pprof *Convert Rprof samples to pprof format.*

Description

Convert Rprof samples to pprof format.

Usage

```
to_pprof(rprof, pprof = tempfile())
```

Arguments

rprof	Path to Rprof samples.
pprof	Path to pprof samples.

Value

Path to pprof samples.

Examples

```
if (identical(Sys.getenv("PROFFER_EXAMPLES"), "true")) {
rprof <- record_rprof(replicate(1e2, sample.int(1e4)))
to_pprof(rprof)
}
```

to_rprof *Convert pprof samples to Rprof format.*

Description

Convert pprof samples to Rprof format.

Usage

```
to_rprof(pprof, rprof = tempfile())
```

Arguments

pprof	Path to pprof samples.
rprof	Path to Rprof samples.

Value

Path to pprof samples.

Examples

```
if (identical(Sys.getenv("PROFFER_EXAMPLES"), "true")) {
pprof <- record_pprof(replicate(1e2, sample.int(1e4)))
to_rprof(pprof)
}
```

Index