Take screenshots of web pages from R

199 commits | 1 branch | 9 releases | 10 contributors

Branch: master | New pull request | Find file

Clone or download

jeroenjanssens and wch Add support to specify casper options to fix #60 (#61) ... Latest commit 311882f Mar 27, 2018

R	Add support to specify casper options to fix #60 (#61)	Mar 27, 2018
inst	Add support to specify casper options to fix #60 (#61)	Mar 27, 2018
man	Add support to specify casper options to fix #60 (#61)	Mar 27, 2018
sourcetools	Rename tools/ to sourcetools/	Mar 16, 2017
tools	Move README images to tools/	Mar 16, 2017
vignettes	Fixes so that R CMD check doesn't leave behind extra temp dir	Sep 25, 2017
.Rbuildignore	Rename tools/ to sourcetools/	Mar 16, 2017
.gitignore	Initial commit	Dec 16, 2014
.travis.yml	Check with multiple versions of R on travis	Dec 27, 2016
DESCRIPTION	Add basic `appshot.shiny.appobj` functionality (#55)	Feb 28, 2018
NAMESPACE	Add support for Rmd files	Nov 29, 2017
NEWS.md	Add basic `appshot.shiny.appobj` functionality (#55)	Feb 28, 2018
README.Rmd	Add support for Rmd files	Nov 29, 2017
README.md	Add support for Rmd files	Nov 29, 2017
appveyor.yml	Use Appveyor	Sep 22, 2016
webshot.Rproj	Update Rproj file	May 26, 2017

README.md

webshot

build passing build passing

Webshot makes it easy to take screenshots of web pages from R. It can also:

- Run Shiny applications locally and take screenshots of the application.
- Render R Markdown documents and take screenshots of the document. Webshot can handle both static Rmd documents and interactive ones (those with `runtime: shiny`).

Installation

Webshot can be installed from CRAN. Webshot also requires the external program PhantomJS. You may either download PhantomJS from its website, or use the function `webshot::install_phantomjs()` to install it automatically.

```
install.packages("webshot")
webshot::install_phantomjs()
```

Usage

By default, `webshot` will use a 992x744 pixel viewport (a virtual browser window) and take a screenshot of the entire page, even the portion outside the viewport.

```
library(webshot)
webshot("https://www.r-project.org/", "r.png")
webshot("https://www.r-project.org/", "r.pdf") # Can also output to PDF
```

You can clip it to just the viewport region:

```
webshot("https://www.r-project.org/", "r-viewport.png", cliprect = "viewport")
```

You can also get screenshots of a portion of a web page using CSS selectors. If there are multiple matches for the CSS selector, it will use the first match.

```
webshot("https://www.r-project.org/", "r-sidebar.png", selector = ".sidebar")
```

If you supply multiple CSS selectors, it will take a screenshot containing all of the selected items.

```
webshot("https://www.r-project.org/", "r-selectors.png",
        selector = c("#getting-started", "#news"))
```

The clipping rectangle can be expanded to capture some area outside the selected items:

```
webshot("https://www.r-project.org/", "r-expand.png",
        selector = "#getting-started",
        expand = c(40, 20, 40, 20))
```

You can take higher-resolution screenshots with the `zoom` option. This isn't exactly the same as taking a screenshot with a HiDPI ("Retina") device: it is like increasing the zoom to 200% in a desktop browser and doubling the height and width of the browser window. This differs from using a HiDPI device because some web pages load different, higher-resolution images when they know they will be displayed on a HiDPI device (but using zoom will not report that there is a HiDPI device).

```
webshot("https://www.r-project.org/", "r-sidebar-zoom.png",
        selector = ".sidebar", zoom = 2)
```

Vectorization

All parameters of function `webshot`. That means that multiple screenshots can be taken with a single command. When taking a lot of screenshots, vectorization can divide by 5 the execution time.

```
# Take a screenshot of different sites
webshot(c("https://www.r-project.org/", "https://github.com/wch/webshot"),
        file = c("r.png", "webshot.png"))

# Save screenshots of the same site in different formats
webshot("https://www.r-project.org/", file = c("r.png", "r.pdf"))

# Take screenshots of different sections of the same site.
# Note that unlike arguments "url" and "file", a list is required to specify
# multiple selectors. This is also the case for arguments "cliprect" and
# "expand"
webshot("http://rstudio.github.io/leaflet/",
        file = c("leaflet_features.png", "leaflet_install.png"),
        selector = list("#features", "#installation"))
```

Screenshots of Shiny applications

The `appshot()` function will run a Shiny app locally in a separate R process, and take a screenshot of it. After taking the screenshot, it will kill the R process that is running the Shiny app.

```
# Get the directory of one of the Shiny examples
appdir <- system.file("examples", "01_hello", package="shiny")
appshot(appdir, "01_hello.png")
```

Screenshots of R Markdown documents

The `rmdshot()` function takes screenshots of R Markdown documents. For static R Markdown documents, it renders them to HTML in a temporary directory (using `rmarkdown::render()`) and then takes a screenshot.

For dynamic R Markdown documents, it runs them using `rmarkdown::run()` in a separate R process and then takes a screenshot. After taking the screenshot, it will kill the R process that is running the document.

```
rmdshot("document.rmd", "document.png")
```

Manipulating images

If you have GraphicsMagick (recommended) or ImageMagick installed, you can pass the result to `resize()` to resize the image after taking the screenshot. This can take any valid ImageMagick geometry specifiction, like `"75%"`, or `"400x"` (for an image 400 pixels wide). However, you may get different (and often better) results by using the `zoom` option: the fonts and graphical elements will render more sharply. However, compared to simply resizing, zooming out may result in slightly different positioning of text and layout elements.

You can also call `shrink()`, which runs OptiPNG to shrink the PNG file losslessly.

```
webshot("https://www.r-project.org/", "r-small-resized.png") %>%
  resize("75%") %>%
  shrink()

# Using zoom instead of resize()
webshot("https://www.r-project.org/", "r-small-zoomed.png", zoom = 0.75) %>%
  shrink()

# Can specify pixel dimensions for resize()
webshot("https://www.r-project.org/", "r-small.png") %>%
  resize("400x") %>%
  shrink()
```

To illustrate the difference between `resize()` and `zoom`, here is an image with `resize("50%")`:

And here is one with `zoom = 0.5`. If you look closely, you'll see that the text and graphics are sharper. You'll also see that the bullet points and text are positioned slightly differently: