Package 'ShinyImage'

September 3, 2017

Type Package

Title Image Manipulation, with an Emphasis on Journaling

Version 0.1.0

Author Alexander Fu [aut],
Ariel Shin [aut],
Norm Matloff [aut, cre]

Maintainer Norm Matloff <nsmatloff@ucdavis.edu>

Description Standard imaging operations, e.g. crop and contrast adjustment, but with ability to go back and forth through sequence of changes, with records being persistent. Optional Shiny interface. Useful to help with the research reproducibility problem, and as a teaching tool.

Depends EBImage, R6, shiny, shinyjs

License MIT + file LICENSE

LazyData TRUE

Suggests knitr, rmarkdown

VignetteBuilder knitr

NeedsCompilation no

Repository CRAN

Date/Publication 2017-09-03 17:06:44 UTC

R topics documented:

runShiny-function *runShiny function to start shiny app*

Description

A function to start shiny app

Usage

```
runShiny(current_shinyimg)
```

Arguments

current_shinyimg	takes in an optional parameter of a image user is currently editing in the command line

Examples

```
## Not run:
   runShiny()

## End(Not run)
```

shinyimg *An EBImage wrapper with integrated history tracking.*

Description

An EBImage wrapper with integrated history tracking.

Usage

```
shinyimg
```

Format

R6Class object.

Value

Object of R6Class with manipulation functions.

Methods

Documentation The user should not need to create an action object. This is a class used exclusively by a shinyimg to keep track of a set of changes.

`set_autodisplay()` Turns on automatic rendering of image. User must still initially call render() to display image

`set_autodisplay_OFF()` Turns off automatic image rendering

`new(img)` Default constructor. `img` can be either a URL or a location of a local image.

`undo()` Undoes the last change done to this image. When the original image state is reached, no more undos are possible.

`redo()` Redos the next action after an undo has been performed. Will no longer redo if there are no more undos to redo.

`shinyUndo()` Undoes the last change done to this image without autorendering; used by Shiny. When the original image state is reached, no more undos are possible.

`redo()` Redos the next action after an undo has been performed without autorendering; used by Shiny. Will no longer redo if there are no more undos to redo.

`toggle_ll()` Returns status of lazy loading.

`copy()` Returns a copy of the image.

`add_brightness()` Adds brightness to the image.

`remove_brightness()` Removes brightness (darkens) to the image.

`add_contrast()` Adds contrast to the image.

`remove_contrast()` Removes contrast from the image.

`add_gamma()` Adds gamma correction to the image.

`remove_gamma()` Remoevs gamma correction from the image.

`add_blur()` Adds blur to the entire photo.

`remove_blur()` Removes blur from the entire photo.

`add_rotate()` Rotates image to the right.

`remove_rotate()` Rotates image to the left.

`set_brightness()` Sets the brightness of the image by number inputted.

`set_contrast()` Sets the contrast of the image by number inputted.

`set_gamma()` Sets the gamma correction of the image by number inputted.

`set_blur()` Sets the blur of the image by number inputted.

`set_rotate()` Sets the degree of rotation of the image by number inputted.

`set_grayscale((num))` Sets the image to grayscale if 1 is inputted; Reverts the image back to colormode if 0 is inputted

`crop()` Uses locator to get corners of an image. Automatically finds min and max coordinates. After two points are selected, a cropping selection can be create in order to crop the image to the desired size.

`cropxy()` Performs same action as crop but it is used by Shiny and the parameters are different.

`get_raw()` Gets the raw matrix slices of the current image.

`gethistory()` Returns a copy of the members of the shinyimg object stored in myhistory.

`get_brightness()` Returns a copy of the value stored for brightness.

`get_contrast()` Returns a copy of the value stored for contrast.

`get_gamma()` Returns a copy of the value stored for gamma correction.

`get_blur()` Returns a copy of the value stored for blur.

`get_rotate()` Returns a copy of the value stored for rotation.

`get_color()` Returns a copy of the value stored for grayscale/colormode.

`get_imghistory()` Returns a copy of the list of image histories.

`get_actions()` Returns a copy of the list of the input parameters.

`checkRedo()` Returns a bool value to check the status of available Redoes; used by Shiny.

`save(filepath)` Saves the current state to be resumed later. `filepath` has a default value of 'workspace.si'

`saveImage(filepath)` Saves a jpg of the image.

`load(filepath)` Loads a previously saved state. `filepath` has a default value of 'workspace.si'

`size()` Returns the current image dimentions.

`render()` Renders the current image.

`toggle_render()` Toggles the automatic rendering after making a change. By default, this option is off.

Examples

```
small_tiger = shinyimg$new(system.file("images","tiger_small.jpg",package="ShinyImage"))

small_tiger$add_brightness() # Adds brightness to image

small_tiger$add_contrast() #Adds contrast to image

small_tiger$undo() # Undoes the brightness addition

small_tiger$redo() # Redoes the brightness addition

## Not run:

small_tiger$add_gamma() #Adds Gamma correction

small_tiger$add_blur() #Adds blur to image

small_tiger$add_rotate() #Adds rotation by 1 degree

small_tiger$save('save.si') # Saves the current state. The filename is optional.

small_tiger$load('save.si') # Loads from a previously saved state. The filename is optional.
#Requires a previously instantiated shinyimg instance (argument provided to new can be null).


## End(Not run)
```

shinyload *Wrapper to load an image from a cold boot.*

Description

Wrapper to load an image from a cold boot.

Usage

```
shinyload(filename)
```

Arguments

filename	The filename of a file previously generated by shinyimg's $save function.

Examples

```
## Not run:

local_tiger = shinyimg$new(system.file("images","sample.jpg",package="ShinyImage"))
# sample.jpg is titled A tiger in the water
#  By Bob Jagendorf
#  [CC BY 2.0 (http://creativecommons.org/licenses/by/2.0)],
#  via Wikimedia Commons
local_tiger$save("tiger.si")
# Restart R
reloaded_tiger = shinyload("tiger.si")


## End(Not run)
```

siaction *Class providing object describing one action.*

Description

Class providing object describing one action.

Usage

```
siaction
```

Format

R6Class object.

Value

Object of R6Class representing a single ShinyImage action.

Fields

`brightness` Stores address of your lightning server.

`contrast` Stores id of your current session on the server.

`gamma` Stores url of the last visualization created by this object.

`crop` A double nested sequence of crops c(c(x1, y1), c(x2, y2)).

`blur` stores value of blur

`rotate` stores value of rotate

`grayscale` stores value of colormode (1 if grayscale, 0 if color) #' @section Methods:

Documentation The user should not need to create an action object. This is a class used exclusively by a shinyimg to keep track of a set of changes.

`new(brightness, contrast, gamma, crop)` This method is used to create object of this class with the appropriate parameters.

`get_action()` This method returns a c() list of the input parameters.

Examples

```
crop = c(c(0, 0), c(1200, 1400))
siaction$new(0.1, 1, 0, crop, 1, 0, 0)
```

Index