Package 'memery'

May 18, 2018

Version 0.5.0

Title Internet Memes for Data Analysts

Description Generates internet memes that optionally include a superimposed inset plot and other atypical features,
combining the visual impact of an attention-grabbing meme with graphic results of data analysis.
The package differs from related packages that focus on imitating and reproducing standard memes.
Some packages do this by interfacing with online meme generators whereas others achieve this natively.
This package takes the latter approach. It does not interface with online meme generators or require any authentication with external websites.
It reads images directly from local files or via URL and meme generation is done by the package.
While this is similar to the 'meme' package available on CRAN, it differs in that the focus is on allowing for non-standard meme layouts and hybrids of memes mixed with graphs.
While this package can be used to make basic memes like an online meme generator would produce,
it caters primarily to hybrid graph-meme plots where the meme presentation can be seen as a backdrop highlighting foreground graphs of data analysis results.
The package also provides support for an arbitrary number of meme text labels with arbitrary size, position and other attributes
rather than restricting to the standard top and/or bottom text placement.
This is useful for proper aesthetic interleaving of plots of data between meme image backgrounds and overlain text labels.
The package offers a selection of templates for graph placement and appearance with respect to the underlying meme.
Graph templates also permit additional template-specific customization.
Animated gif support is provided but this is optional and functional only if the 'magick' package is installed.
'magick' is not required unless gif functionality is desired.

License MIT + file LICENSE

Encoding UTF-8

LazyData true

ByteCompile true

URL https://github.com/leonawicz/memery

BugReports https://github.com/leonawicz/memery/issues

Suggests testthat, knitr, rmarkdown, covr, magick, lintr

VignetteBuilder knitr

Depends showtext

Imports sysfonts, grid, png, jpeg, Cairo, ggplot2, cowplot, magrittr, purrr, shiny, shinycssloaders, shinyBS, colourpicker

RoxygenNote 6.0.1

NeedsCompilation no

Author Matthew Leonawicz [aut, cre]

Maintainer Matthew Leonawicz <mfleonawicz@alaska.edu>

Repository CRAN

Date/Publication 2018-05-17 22:45:52 UTC

R topics documented:

`car_shiny` *The ca(R) Shiny promotional meme*

Description

Recreate the ca(R) Shiny promotional meme using different plots, frame rate or output size.

Usage

```
car_shiny(file, p1 = NULL, p2 = NULL, fps = 10, test_frame = FALSE,
  mult = 1)
```

Arguments

file	character, output filename.
p1	ggplot object for top half of (ca)R Shiny meme.
p2	ggplot object for bottom half of (ca)R Shiny meme.
fps	integer, frames per second.
test_frame	logical, keep only the first frame. Ideal for saving time during testing.
mult	numeric, factor by which to multiply the output meme dimensions. Use conservatively.

Details

This function offers limited control over customization; it is intended as a canned example. For additional customization, use this function's source code in an R script where you can easily alter other values.

After writing the first layer of output to `file`, `file` is recursively read and written again two more times in order to add all layers to the animated gif, since the underlying `meme` function does not accept vectorized inset plots.

The Shiny logo is by RStudio.

Value

nothing is returned, but a file is saved to disk.

Examples

```
library(ggplot2)
file <- "memery-car-shiny.gif" # outfile
set.seed(1)
p1 <- ggplot(data.frame(x = rbeta(100000, 10, 3)), aes(x)) +
  geom_histogram(colour = "white", fill = "#88888880", size = 1, bins = 30)

means <- (8:1)^3
sds <- 10*(8:1)
d <- data.frame(
  x = rep(factor(1:8), each = 100),
  y = unlist(purrr::map2(means, sds, ~rnorm(100, .x, .y) / 200))
)

p2 <- ggplot(d, aes(x, y)) +
  geom_boxplot(colour = "white", fill = "#5495CF80", outlier.colour = NA) +
  geom_point(shape = 21, colour = "white", fill = "#5495CF80", size = 1,
             position = position_jitter(0.15)) +
  scale_x_discrete(expand = c(0, 0.02)) + scale_y_continuous(expand = c(0.02, 0)) +
  theme_void() +
  theme(plot.margin = unit(rep(5, 4), "mm"),
        panel.grid.major = element_line(colour = "#FFFFFF50", linetype = 2),
        panel.grid.minor = element_line(colour = "#FFFFFF50", linetype = 2))

## Not run: car_shiny(file, p1, p2, test_frame = TRUE)
```

inset *Meme plot inset templates*

Description

Templates for the position and background of a ggplot inset.

Usage

```
inset_position(type = "default", size = 0.2, margin = 0.025)

inset_background(type = "default")

inset_templates(type)
```

Arguments

type	character, name of template.
size	numeric, width (length-1) or width and height (length-2) of inset. See details.
margin	numeric, x-axis margin (length-1) or x- and y-axis margins (length-2) around corner insets. See details.

Details

inset_position and inset_background assist with some basic options for position and background of the optional ggplot inset graphic. inset_templates can be used to view the available templates. See examples. If a template is not available to suit your needs, provide your own argument list to meme in the form of, e.g., inset_pos = list(w = 0.95, h = 0.6, x = 0.5, y = 0.325).

Value

a list of arguments passed to either inset_pos or inset_bg in meme.

Size and position

The coordinate system for the meme plot ranges from zero to one in x and y. The width, height and (x,y) center defined by inset_position arguments therefore take values between zero and one.

The default position is for an inset plot that takes up 95% of the width and 60% of the height of the meme plot, with 2.5% margins on the sides and bottom. Other templates include four corner thumbnails. To use these, set type equal to "tl", "tr", "br" or "bl", for top right, top left, bottom right and bottom left, respectively. There is also a "center" type.

When specifying the corner or center inset types, the inset is a square thumbnail with width and height of 0.2 units (20%) and 0.025 (2.5%) margins from the edges of the plot. However, these templates are not absolute. You can further adjust the size and distance from the edges using size and margin. These arguments can be scalar, in which case the inset remains square and the margins are equal. If a length-2 vector, size can provide unique width and height for a rectangular inset.

Similarly, `margin` can provide different margins for the distance to a side vs. the top or bottom edge of the meme plot.

For `type = "center"`, `size` is used but `margin` is ignored, giving you control over the thumbnail size. Appending the letter `q` to a corner thumbnail template ID, e.g. `type = "blq"`, yields a quadrant plot. In contrast to `type = "center"`, these types allow for user control over margins for plots of fixed coverage area. Specifying the right combination of `size` and `margin` with a corner thumbnail template can be used to create a quadrant plot, but using a quadrant template simplifies this.

`size` and `margin` are provided for convenience, adding more control over the position templates. If you require more specific size and position control, simply pass your own 4-argument list as described above. This is the structure generated by `inset_position` for any `type` and expected by `meme`.

Background

For `inset_background`, the few templates all revolve around the `type = "default"` template. `type = "sq"` simply removes the rounded corners. `"op"` provides a fully opaque white background instead of the default 50% transparency. `"opsq"` does both. `"blank"` hides the background panel. There is no substantial need to provide many templates because, as with `inset_position`, `inset_background` generates a simple list of a few arguments that can be easily provided to `meme` explicitly without the use of `inset_background`.

Examples

```
inset_templates("position")
inset_templates("background")
inset_position()
inset_position("br")
inset_position("brq", margin = 0.05)
inset_position("br", size = 0.4, margin = 0)
inset_background()
inset_background("opsq")
inset_background("blank")
```

`meme` *Generate a meme*

Description

Generate a meme with a background image, text label and optional inset graphic.

Usage

```
meme(img, label, file, size = 1, family = "Impact", col = "white",
  shadow = "black", label_pos = text_position(length(label)),
  inset = NULL, ggtheme = memetheme(), inset_bg = inset_background(),
  inset_pos = inset_position(), width, height, mult = 1)
```

```
meme_gif(img, label, file, size = 1, family = "Impact", col = "white",
  shadow = "black", label_pos = text_position(length(label)),
  inset = NULL, ggtheme = memetheme(), inset_bg = inset_background(),
  inset_pos = inset_position(), width, height, mult = 1, fps = 10,
  frame = 0, ...)
```

Arguments

`img`	path to image file, png or jpg.
`label`	character, meme text. May be a vector, matched to `label_pos`.
`file`	output file, png or jpg.
`size`	label size. Actual size affected by image size (i.e., `cex`).
`family`	character, defaults to `"Impact"`, the classic meme font. See details.
`col`	label color.
`shadow`	label shadow/outline color.
`label_pos`	named list of position elements for the meme text `w`, `h`, `x` and `y`. Each element may be a vector. See details.
`inset`	a ggplot object. This is an optional inset plot and may be excluded.
`ggtheme`	optional ggplot2 theme. If ignored, the default `memery` ggplot2 theme is used.
`inset_bg`	a list of background settings for the plotting region containing `inset`. See details.
`inset_pos`	named list of position elements for the `inset` inset plot: `w`, `h`, `x` and `y`.
`width`	numeric, width of overall meme plot in pixels. If missing, taken from `img` size.
`height`	numeric, height of overall meme plot in pixels. If missing, taken from `img` size.
`mult`	numeric, a multiplier. Used to adjust width and height. See details.
`fps`	frames per second, only applicable to `meme_gif`. See details.
`frame`	integer, frame numbers to include. The default `frame = 0` retains all frames. Only applicable to `meme_gif`. See details.
`...`	additional arguments passed to `meme_gif`.

Details

This function generates and saves a meme as a jpg or png file.

Fonts

Memes use the Impact font by default. This is a Windows font. If using `memery` on Linux for example, you would have to first install the font if not already installed on the system. If Impact or any other font family passed to `meme`, e.g. `family = "Consolas"`, is not installed on an operating system, `meme` will ignore it and fall back on `family = "serif"` internally. If unfamiliar, explore the documentation and examples available for the `showtext` and `sysfonts` packages, which `merery` leverages.

Text labels

List elements in `label_pos` must all be the same length and must match the length of `label`. This is provided for generality but is most suited to length-2 cases; the use of meme title/subtitle or top/bottom text pairs. Similarly, `size`, `family`, `col` and `shadow` may be vectorized. For example, top and bottom text can have different font size and family and the font text and shadow can be different colors.

Inset graphic

The meme plot may optionally include an inset plot by passing a ggplot object to `inset`. This makes the memes more fun for data analysts. See examples.

The plotting region containing `inset` is a specific viewport in the `grid` layout. `inset_bg` is a list of arguments that affect the background of this part of the meme plot. They define a rectangle that by default is semi-transparent with rounded corners and no border color. This can be changed via the list arguments `fill`, `col` and `r`.

The inset plot `inset` is placed above this layer and also fills the region. The default ggplot2 theme used my `meme`, `memetheme`, uses transparent ggplot plot and panel background fill and plot border color that allow the background viewport rectangle and its rounded corners to show through. The default theme also has no plot margins.

If you supply a different theme via `ggtheme`, you can provide different plot and panel background fill and plot border color as part of the theme. For similar no-margin themes, if the plot background fill or border color are not fully transparent, set the viewport rectangle corner radius to zero so that rounded corners do not show inside the panel background. For opaque plot background fill this will not matter.

Of course, the plot and panel background fill should still be transparent or semi-transparent if occupying a large amount of the total meme plot area or it will obscure the meme image itself. An alternative is to use `inset` as, for example, a tiny thumbnail in the corner of the meme plot, in which case full opacity is not necessarily an issue. If you do not want to override the theme of your plot and do not wish to pass a theme explicitly by `ggtheme`, you can set `ggtheme = NULL`

Dimensions and image processing

Specifying `width` and `height` is not required. By default, output file dimensions are taken from the input file, `img`. However, these arguments can be used to override the default dimension matching. The aspect ratio is fixed so if you change the two disproportionately, you will increasing the canvas, adding bars on two sides; it will not stretch the image.

`mult` can be set less than one if relying on `img` dimension for meme plot width and height and `img` is large. It is equivalent to scaling proportionately with `width` and `height` maintaining the original aspect ratio. Whether or not `width` and/or `height` are provided, `mult` is always applied (defaults to `mult = 1`).

Beyond this basic resizing to help control output file size, it is not the intent of `memery` to offer general image processing capabilities. If adjustments to source images are desired, you should use a dedicated package for image processing. `magick` is recommended.

Reading and writing gifs

Reading and writing gifs requires the `magick` package, which in turn requires that you have ImageMagick installed on your system. Since this is not required for any other part of `memery` and it represents a minor use case, the package does not have these dependencies. `magick` is listed as a suggested package for memery; it is not imported as a dependency. `meme_gif` is an optional extra function. In order to use it, install ImageMagick on your system and install the `magick` package.

See the example below if your system meets these requirements. As with jpg or png image inputs, if additional control is required for making custom adjustments to gif image frames, use the `magick` package for image pre-processing. `meme` only provides basic control over output size and `meme_gif` only adds control over gif frame selection and rate.

Examples

```
# Prepare data and make a graph
library(ggplot2)
x <- seq(0, 2*pi, length.out = 50)
panels <- rep(c("Plot A", "Plot B"), each = 50)
d <- data.frame(x = x, y = sin(x), grp = panels)
txt <- c("Philosoraptor's plots", "I like to make plots",
  "Figure 1. (A) shows a plot and (B) shows another plot.")

p <- ggplot(d, aes(x, y)) + geom_line(colour = "cornflowerblue", size = 2) +
  geom_point(colour = "orange", size = 4) + facet_wrap(~grp) +
  labs(title = txt[1], subtitle = txt[2], caption = txt[3])

# meme image background and text labels
img <- system.file("philosoraptor.jpg", package = "memery")
lab <- c("Title meme text", "Subtitle text")

## Not run:

# Not run due to file size
# basic meme
meme(img, lab[1:2], "meme_basic.jpg")
# data analyst's meme
meme(img, lab[1:2], "meme_data.jpg", size = 2, inset = p, mult = 2)

## End(Not run)

# data meme with additional content control
vp_bg <- list(fill = "#FF00FF50", col = "#FFFFFF80") # graph background
# arbitrary number of labels, placement, and other vectorized attributes
lab <- c(lab, "Middle plot text")
pos <- list(w = rep(0.9, 3), h = rep(0.3, 3), x = c(0.35, 0.65, 0.5),
  y = c(0.95, 0.85, 0.3))
fam <- c("Impact", "serif", "Impact")
clrs1 <- c("black", "orange", "white")
clrs2 <- clrs1[c(2, 1, 1)]
meme(img, lab, "meme_data2.jpg", size = c(2, 1.5, 1), family = fam, col = clrs1,
  shadow = clrs2, label_pos = pos, inset = p, inset_bg = vp_bg, mult = 2)
```

```
## Not run:

# Not run due to file size and software requirements
# GIF meme. Requires Imagemagick and magick package. See details.
p <- ggplot(d, aes(x, y)) + geom_line(colour = "white", size = 2) +
  geom_point(colour = "orange", size = 1) + facet_wrap(~grp) +
  labs(title = "The wiggles", subtitle = "Plots for cats",
       caption = "Figure 1. Gimme sine waves.")
lab <- c("R plots for cats", "Sine wave sine wave sine wave sine wave...")
pos <- list(w = rep(0.9, 2), h = rep(0.3, 2), x = rep(0.5, 2), y = c(0.9, 0.75))
img <- "http://forgifs.com/gallery/d/228621-4/Cat-wiggles.gif"
meme_gif(img, lab, "meme_data3.gif", size = c(1.5, 0.75), label_pos = pos,
         inset = p, inset_bg = list(fill = "#00BFFF80"), mult = 1.5, fps = 20)

## End(Not run)
```

`memeApp` *Run memery example app*

Description

Launch the memery example app in your browser.

Usage

```
memeApp()
```

Details

This example app launches with animated gif support and relevant initial example gif if the magick package and ImageMagick software are installed on the user's system. Otherwise, a simplified version of the app that does not provide gif support launches and starting images must be jpg or png.

Due to how `meme_gif` works, gifs will not display in a hosted app, i.e., on `shinyapps.io`. Memes based on gifs with many frames also take longer to render. By default, the app is set to render any animated gif input into a static meme using only the first frame in the animated gif. If it takes one second to render a single frame, expect it to take about 43 seconds to render the 43-frame example animated gif that comes preloaded in the app. If the user wishes to wait, the input control menu for output frames can be switched from first frame to all frames.

While jpg and png memes will display in a hosted app like on `shinyapps.io`, the impact font will also not likely be available on a given server. For all these reasons this packaged app is not hosted elsewhere. The best and intended experience is to use the app locally via the `memery` package.

All ggplot objects that exist in the global environment when the app is launched propagate a selection menu in app for use as inset plots to overlay meme image backgrounds. If there are no ggplot objects in the global environment, one named `memery_testplot` will be created within the app environment and will appear in the selection menu instead.

A meme can be saved from an app by right-clicking on the image in your web browser and selecting the save option just like with any other web images.

Examples

```
## Not run: memeApp()
```

`memery` *memery: Internet memes for data analysts.*

Description

The memery package generates internet memes that optionally include a superimposed inset plot and other atypical features, combining the visual impact of an attention-grabbing meme with graphic results of data analysis.

Details

The package differs from related packages that focus on imitating and reproducing standard memes. Some packages do this by interfacing with online meme generators whereas others achieve this natively. This package takes the latter approach. It does not interface with online meme generators or require any authentication with external websites. It reads images directly from local files or via URL and meme generation is done by the package.

While this is similar to the 'meme' package available on CRAN, it differs in that the focus is on allowing for non-standard meme layouts and hybrids of memes mixed with graphs. While this package can be used to make basic memes like an online meme generator would produce, it caters primarily to hybrid graph-meme plots where the meme presentation can be seen as a backdrop highlighting foreground graphs of data analysis results.

The package also provides support for an arbitrary number of meme text labels with arbitrary size, position and other attributes rather than restricting to the standard top and/or bottom text placement. This is useful for proper aesthetic interleaving of plots of data between meme image backgrounds and overlain text labels. The package offers a selection of templates for graph placement and appearance with respect to the underlying meme. Graph templates also permit additional template-specific customization.

Animated gif support is provided but this is optional and functional only if the `magick` package is installed.

`memetheme` *Default meme theme*

Description

The default ggplot2 theme for meme plots.

Usage

```
memetheme(base_size = 14, base_family = "", base_col = "white")
```

Arguments

base_size	numeric, the base size.
base_family	character, the base font family.
base_col	character, the base color for all title text and axis lines and ticks.

Value

a ggplot2 theme.

---

text_position *Default meme text position*

Description

Convenience function for meme text position in a meme plot.

Usage

```
text_position(n)
```

Arguments

n	integer, number of meme text labels.

Details

This function takes an integer 1 or 2 and returns, respectively, top or symmetrical top and bottom meme text position arguments. This function is used as the default for the label_pos argument in meme. It is provided if you do not want to bother with specifying coordinates and exact placement does not matter. It is not intended for placement of more than two meme text labels and any value other than 1 or 2 returns an error.

Value

a list of meme text label position arguments: w (width), h (height), and x and y coordinates.

Examples

```
text_position(1)
text_position(2)
```

Index