Package 'rbranding'

November 21, 2025

Title Manage Branding and Accessibility of R Projects

Version 0.1.0

Description A tool for building projects that are visually consistent, accessible, and easy to maintain. It provides functions for managing branding assets, applying organization-wide themes using 'brand.yml', and setting up new projects with accessibility features and correct branding. It supports 'quarto', 'shiny', and 'rmarkdown' projects, and integrates with 'ggplot2'. The accessibility features are based on the Web Content Accessibility Guidelines <https://www.w3.org/WAI/WCAG22/quickref/?versions=2.1> and Accessible Rich Internet Applications (ARIA) specifications <https://www.w3.org/WAI/ARIA/apg/>. The branding framework implements the 'brand.yml' specification <https://posit-dev.github.io/brand-yml/>.

License MIT + file LICENSE

URL https://epiforesite.github.io/rbranding/,
https://github.com/EpiForeSITE/rbranding

BugReports https://github.com/EpiForeSITE/rbranding/issues

Imports credentials, ggplot2, utils, yaml

Suggests DT, grid, here, htmltools, knitr, leaflet, pkgdown, plotly, png, rmarkdown, shiny, showtext, sysfonts, thematic, tinytest

VignetteBuilder knitr

Encoding UTF-8

RoxygenNote 7.3.2

NeedsCompilation no

Author Willy Ray [aut, cre],
Andrew Pulsipher [aut, ctb] (ORCID: <https://orcid.org/0000-0002-0773-3210>),
Centers for Disease Control and Prevention's Center for Forecasting and Outbreak Analytics [fnd] (Cooperative agreement CDC-RFA-FT-23-0069)

Maintainer Willy Ray <william.ray@hsc.utah.edu>

Repository CRAN

Date/Publication 2025-11-21 15:00:02 UTC

Contents

`brand_add_logo` *Add Brand Logo to ggplot2 Plot*

Description

`brand_add_logo` adds a logo from the brand configuration as an annotation to a ggplot2 plot.

Usage

```
brand_add_logo(x = 0.9, y = 0.1, size = 0.05, logo_type = "icon")
```

Arguments

`x`	Numeric. Horizontal position of the logo (0-1 scale). Default is 0.9.
`y`	Numeric. Vertical position of the logo (0-1 scale). Default is 0.1.
`size`	Numeric. Size of the logo as a fraction of the plot (0-1 scale). Default is 0.05.
`logo_type`	Character. Which logo to use: "icon" (default) or "full".

Details

This function reads the logo path from the stored brand configuration and creates a ggplot2 annotation layer. The brand configuration must be loaded first using brand_set_ggplot().

The function supports PNG images and requires the 'png' and 'grid' packages.

Value

A ggplot2 annotation_custom layer that can be added to a plot with `+`.

Examples

```
{
# First set the brand theme to load configuration
old_wd <- getwd()
setwd(tempdir()) # Change to temp directory for example
brand_init()
get_brand_public()
get_template("blank")
brand_set_ggplot()

# Create a plot and add logo
library(ggplot2)
ggplot(mtcars, aes(x = mpg, y = wt)) +
  geom_point() +
  labs(title = "Example Plot") +
  brand_add_logo()

# Customize logo position and size
ggplot(mtcars, aes(x = mpg, y = wt)) +
  geom_point() +
  labs(title = "Example Plot") +
  brand_add_logo(x = 0.1, y = 0.9, size = 0.08)

setwd(old_wd) # Restore original working directory
}
```

`brand_init` *Initialize branding configuration*

Description

`brand_init` initializes the branding configuration by creating two files:

- `rbranding_config.yml`: contains remote and local file paths to brand files
- `_brand.yml`: a placeholder branding file It is intended to be run once. Use a `get_brand_*()` function to download/update the brand file.

Usage

```
brand_init(brand_url = NULL, install_path = ".")
```

Arguments

`brand_url`	Optional URL. Points to the remote brand file. If `NULL`, defaults to rbranding's brand file on GitHub.
`install_path`	Optional string. Directory where the files should be created. Defaults to the current working directory.

Value

NULL. Called for its side effects: downloading and creating `rbranding_config.yml` and `_brand.yml` files.

Examples

```
tmpdir <- file.path(tempdir(), "brand_files")

brand_init(install_path = tmpdir)

# Clean up
unlink(tmpdir, recursive = TRUE)
```

`brand_reset_ggplot` *Reset ggplot2 Theme to Previous State*

Description

`brand_reset_ggplot` resets the ggplot2 theme to the state it was in before brand_set_ggplot() was called.

Usage

```
brand_reset_ggplot()
```

Value

Invisibly returns TRUE if reset was successful, FALSE if no previous theme was stored.

Examples

```
{
# Set brand theme
old_wd <- getwd()
setwd(tempdir()) # Change to temp directory for example
brand_init()
get_brand_public()
brand_set_ggplot()

# Create some plots with brand theme...

# Reset to original theme
brand_reset_ggplot()
setwd(old_wd) # Restore original working directory
}
```

`brand_set_ggplot` *Set ggplot2 Theme from Brand Configuration*

Description

`brand_set_ggplot` sets the ggplot2 theme based on colors and typography defined in a _brand.yml file. This function reads the brand configuration and applies it as the default ggplot2 theme.

Usage

```
brand_set_ggplot(brand_file = NULL, use_fonts = TRUE)
```

Arguments

`brand_file`	Path to the _brand.yml file. If NULL, looks for _brand.yml in the current directory.
`use_fonts`	Logical. Whether to attempt to load and use custom fonts from the brand file. Default is TRUE.

Details

This function reads a brand.yml file and extracts color and = typography information to create a custom ggplot2 theme. The function:

- Maps brand colors to ggplot2 theme elements
- Attempts to load Google Fonts specified in the brand file
- Stores the previous theme for later restoration
- Sets the new theme as the default for all subsequent ggplot2 plots

The brand.yml file should follow the schema defined at: https://github.com/posit-dev/brand-yml/

Value

Invisibly returns the previous ggplot2 theme (for potential restoration).

Examples

```
{
# Set theme from default _brand.yml file
old_wd <- getwd()
setwd(tempdir()) # Change to temp directory for example
brand_init()
get_brand_public()
brand_set_ggplot()

# Create a plot - will use the brand theme
library(ggplot2)
ggplot(mtcars, aes(x = mpg, y = wt)) +
  geom_point() +
  labs(title = "Example Plot with Brand Theme")

# Reset to original theme
brand_reset_ggplot()

setwd(old_wd) # Restore original working directory
}
```

---

get_brand_private_github

Download the latest branding file from a private GitHub repository

---

Description

`get_brand_private_github` downloads the latest `_brand.yml` file from the remote URL specified in `rbranding_config.yml` or provided as function arguments. The remote file is assumed to be in a private GitHub repository and requires authentication. If the local `_brand.yml` file does not exist, it will be created. If the local file is different from the remote file, the function will save the contents to `bak_brand.yml` (as backup) and overwrite the local file with the contents of the remote file. When the function is run interactively (e.g., in RStudio console), the user is instead prompted to choose whether to overwrite the file and whether or not to create the backup.

Usage

```
get_brand_private_github(
  remote_file = NULL,
  local_file = NULL,
  auth_token = NULL,
  config_file = "rbranding_config.yml",
  run_interactive = TRUE,
  backup = FALSE,
  backup_folder = "."
)
```

Arguments

`remote_file`	Optional URL. Points to the remote brand file. If `NULL`, the value in the configuration file will be used.
`local_file`	Optional string. Path to the local branding file. If `NULL`, the value in the configuration file will be used.
`auth_token`	Optional authentication token for accessing the private GitHub repository. If `NULL`, the function will attempt to retrieve the token from the `GITHUB_TOKEN` environment variable or the git credential store.
`config_file`	Path to the configuration file. Default is `rbranding_config.yml`.

`run_interactive`	Logical indicating whether to run interactively. Defaults to TRUE.
`backup`	Logical indicating whether to create a backup of the local file if it is different from the remote file. Ignored if run interactively. Defaults to FALSE.
`backup_folder`	Folder where the backup file should be saved, if needed. Defaults to current working directory.

Value

NULL. Called for its side effects: updating `_brand.yml` and possibly creating `bak_brand.yml`

Examples

```
# Interactive example
if (interactive()) {
  tmpdir <- file.path(tempdir(), "brand_files")

  # Initialize config and local brand file
  brand_init(install_path = tmpdir)

  # Update local brand file if needed
  get_brand_private_github(
    config_file = file.path(tmpdir, "rbranding_config.yml")
  )

  # Cleanup
  unlink(tmpdir, recursive = TRUE)
}

## Not run:
  # Example not run because it requires a GitHub
  # personal access token with repo access

  tmpdir <- file.path(tempdir(), "brand_files")
  brand_init(install_path = tmpdir)

  get_brand_private_github(
   config_file = file.path(tmpdir, "rbranding_config.yml"),
   auth_token = "your_github_token_here",
   run_interactive = FALSE,
   backup = TRUE,
   backup_folder = tmpdir
  )

  # Cleanup
  unlink(tmpdir, recursive = TRUE)

## End(Not run)
```

get_brand_public — *Download the latest branding file from a public source*

Description

`get_brand_public` downloads the latest `_brand.yml` file from the remote URL specified in `rbranding_config.yml` or provided as function arguments. The remote file is assumed to be publicly accessible (no authentication), such as a website or public GitHub repository. If the local `_brand.yml` file does not exist, it will be created. If the local file is different from the remote file, the function will save the contents to `bak_brand.yml` (as backup) and overwrite the local file with the contents of the remote file. When the function is run interactively (e.g.,in RStudio console), the user is instead prompted to choose whether to overwrite the file and whether or not to create the backup.

Usage

```
get_brand_public(
  remote_file = NULL,
  local_file = NULL,
  config_file = "rbranding_config.yml",
  run_interactive = TRUE,
  backup = FALSE,
  backup_folder = "."
)
```

Arguments

`remote_file`	Optional URL. Points to the remote brand file. If `NULL`, the value in the configuration file will be used.
`local_file`	Optional string. Path to the local branding file. If `NULL`, the value in the configuration file will be used.
`config_file`	Path to the configuration file. Default is `rbranding_config.yml`.
`run_interactive`	Logical indicating whether to run interactively. Defaults to TRUE.
`backup`	Logical indicating whether to create a backup of the local file if it is different from the remote file. Ignored if run interactively. Defaults to FALSE.
`backup_folder`	Folder where the backup file should be saved, if needed. Defaults to current working directory.

Value

NULL. Called for its side effects: updating `_brand.yml` and possibly creating `bak_brand.yml`

Examples

```
# Interactive example
if (interactive()) {
  tmpdir <- file.path(tempdir(), "brand_files")

  # Initialize config and local brand file
  brand_init(install_path = tmpdir)

  # Update local brand file if needed
  get_brand_public(
    config_file = file.path(tmpdir, "rbranding_config.yml")
  )

  # Cleanup
  unlink(tmpdir, recursive = TRUE)
}

# Non-interactive example
tmpdir <- file.path(tempdir(), "brand_files")
brand_init(install_path = tmpdir)

get_brand_public(
 config_file = file.path(tmpdir, "rbranding_config.yml"),
 run_interactive = FALSE,
 backup = TRUE,
 backup_folder = tmpdir
)

# Cleanup
unlink(tmpdir, recursive = TRUE)
```

`get_template` *Copy template files into project*

Description

`get_template` copies example files from the package's `templates` directory into the user's current working directory or a specified subdirectory.

Usage

```
get_template(template_name = NULL, install_to = NULL)
```

Arguments

`template_name`	Optional string. Name of the template to use. Corresponds to a folder name `templates/`. If NULL (default) within an interactive session, the function will list available templates and prompt the user to select one.
`install_to`	Optional string. Directory where the example files should be copied. If NULL (default), the current working directory will be used.

Value

NULL. Called for its side effects: copying template files into the user's project directory.

Examples

```
if (interactive()) {
  get_template() # prompts user to select an example
}

tmpdir <- file.path(tempdir(), "wastewater_test")
get_template(template_name = "shiny_wastewater", install_to = tmpdir)

# Cleanup
unlink(tmpdir, recursive = TRUE)
```

Index