# Package 'teachingApps'

June 10, 2018

**Type** Package

**Title** Apps for Teaching Statistics, R Programming, and Shiny App Development

**Version** 1.0.4

**Date** 2018-06-08

**Maintainer** Jason Freels <auburngrads@live.com>

**Description** Contains apps and gadgets for teaching data analysis and statistics concepts along with how to implement them in R. Includes tools to make app development easier and faster by nesting apps together.

**URL** https://github.com/Auburngrads/teachingApps

**BugReports** https://github.com/Auburngrads/teachingApps/issues

**Encoding** UTF-8

**License** GPL (>= 2)

**LazyData** true

**Depends** R (>= 3.1.0), shiny, pacman

**Imports** shinyAce,shinythemes,shinydashboard,miniUI,radarchart,dplyr, DT,metricsgraphics,leaflet,threejs,d3heatmap,tidyr,datasets, plotly,dygraphs,networkD3,diagram,DiagrammeR,visNetwork, actuar,markdown,stats,knitr,data.table,Rcpp (>= 0.12.14), rprojroot,utils,devtools,graphics,scales,yaml,magrittr,ggplot2, RcppNumerical

**NeedsCompilation** yes

**Archs** i386, x64

**Suggests** testthat, rmarkdown

**RoxygenNote** 6.0.1

**LinkingTo** Rcpp (>= 0.12.14), BH (>= 1.58.0-1), RcppEigen, RcppNumerical

**Author** Jason Freels [aut, cre], Bradley Boehmke [ctb]

**Repository** CRAN

**Date/Publication** 2018-06-10 04:22:55 UTC

# R topics documented:



---

`add_css` *Add teachingApps CSS style rules to an app*

---

## Description

Add teachingApps CSS style rules to an app

## Usage

```
add_css()
```

## Details

This function should not be called directly but must be be included within the body of an app's `ui`

## Value

teachingApps style rules defined to an app

## See Also

`add_theme`

## add_logo *Adds a branding logo to the footer of a navbarPage app*

### Description

Adds a branding logo to the footer of a navbarPage app

### Usage

```
add_logo(app_dir = getShinyOption("appDir"),
  git_user = getShinyOption("gitUser"), icon = getShinyOption("icon"),
  img = getShinyOption("img"))
```

### Arguments

| | |
|---|---|
| app_dir | character Directory in which the app files are located |
| git_user | character GitHub account username (see details) |
| icon | character Name of a fontAwesome icon printed in the app footer |
| img | character Path to an image printed in the app footer |

### Value

A fontAwesome icon or an image printed in the footer of a navbarPage app

### See Also

create_logo

## add_options *Pass objects and customization options to a shiny app*

### Description

Provides a general method for passing arguments to shiny apps allowing for dynamic customization.

### Usage

```
add_options(opts, dir, theme = "flatly", icon = NULL, img = NULL,
  git_user = NULL)
```

### Arguments

| | |
|---|---|
| opts | A list of additional options or objects to pass to a shiny app |
| dir | A character string indicating the path to the directory containing ui.R and server.R |
| theme | A character string naming a Bootswatch color theme (used by shinythemes::shinytheme) |
| icon | A character string naming a fontAwesome icon to be placed in the footer of a navbarPage app |
| img | A character string for the path/url of an image to be placed in the footer of a navbarPage app |
| git_user | A character string for github username used in the branding link |

### Details

Shiny apps are not functions. Thus, customization options cannot be passed to a shiny app as simply as arguments are passed between functions. Further, the manner in which objects are loaded prior to deploying an app differ if the app will be published as a stand-alone or embedded within an rmarkdown docmuent. Assigning objects as shiny::shinyOptions ensures that these values are passed to a shiny app and can be deployed.

### Value

A list of shiny options set with shinyOptions

### See Also

add_css

add_logo

---

## add_packages *Install and load an R package*

---

### Description

Install and load an R package

### Usage

```
add_packages(pkg = NULL, repo = NULL, pub = FALSE)
```

### Arguments

| | |
|---|---|
| pkg | character Name of a package to be installed/loaded |
| repo | character Name of the repository from which the package should be installed. |
| pub | logical variable indicating whether the app be published (see details) |

## Details

If `repo = NULL` the package will be installed from the CRAN. Otherwise, `repo` is a character string that referring to the GitHub account in which the package is located.

When publishing apps on shinyapps.io or shinyServer, attempting to `install.packages` will result in an error. Calls to `install.packages` should not be included within an app.

## Value

A printed shiny app

---

`add_rmd` *Add an rmarkdown file to an app*

---

## Description

Run inline and stand-alone code chunks and include results as part of a shiny app. Include LaTeX-typeset equations with MathJax

## Usage

```
add_rmd(rmd, path)
```

## Arguments

| | |
|---|---|
| `rmd` | `character` Name of an rmarkdown file saved in the app directory |
| `path` | Path to a file outside of the app directory |

## See Also

`add_server` `add_ui`

## Examples

```
## Not run:
# see examples in add_server and add_ui documentation

## End(Not run)
```

add_server *Add the server of one app to the server of another app*

## Description

Sources a server.R file before parsing and evaluating its contents in a specified environment

## Usage

```
add_server(app, path, env = NULL)
```

## Arguments

| | |
|---|---|
| app | Name of the teachingApp from which the content of the server.R will be pulled |
| path | Path to a directory containing the app from which the content of the server.R will be pulled |
| env | Environment in which the call is made, typically environment() |

## Details

Currently, this function can be used to insert an server into a navbarPage app. The types of apps that can be inserted are:

- fluidPage
- bootstrapPage
- pageWithSidebar
- basicPage
- fixedPage

A server can be added as an entire tabPanel or as a row within within a tabPanel portion of a shiny app.

May be used with apps stored in packages other than teachingApps. However, apps are assumed be stored in the apps/ directory located at top level of the package.

## Value

An Observer-class object resulting from evaluating a server.R file

## See Also

add_ui, add_rmd

## Examples

```
## Not run:

## server.R from app: 'maximum_likelihood'

server_ml <- system.file('apps',
                         'maximum_likelihood',
                         'server.R',
                         package = 'teachingApps')
browseURL(server_ml)

## server.R from app: 'distribution_weibull'

server_dw <- system.file('apps',
                         'distribution_weibull',
                         'server.R',
                         package = 'teachingApps')
browseURL(server_dw)


## End(Not run)
```

---

add_theme *Add a bootswatch color theme to an app*

---

## Description

Add a bootswatch color theme to an app

## Usage

```
add_theme(theme = NULL)
```

## Arguments

| | |
|---|---|
| theme | character A bootswatch theme name (see details) |

## Details

This function should not be called directly but is invoked when an app is rendered.

Themes are provided by calling `shinythemes::shinytheme`, therefore available theme names are those provided by the `shinythemes` package. By default, `theme = 'flatly'`

add_ui *Add the UI of one app within the UI of another app*

## Description

Sources a `ui.R` file before parsing and evaluating its contents in a specified environment

## Usage

```
add_ui(app, path)
```

## Arguments

| | |
|---|---|
| `app` | Name of the app from which the content of the `ui.R` will be pulled |
| `path` | Path to a directory containing the app from which the content of the `ui.R` will be pulled |

## Details

Currently, this function can be used to insert an `server` into a `navbarPage` app. The types of apps that can be inserted are:

- `fluidPage`
- `bootstrapPage`
- `pageWithSidebar`
- `basicPage`
- `fixedPage`

A `server` can be added as an entire `tabPanel` or as a row within within a `tabPanel` portion of a shiny app.

May be used with apps stored in packages other than teachingApps. However, apps are assumed be stored in the `apps/` directory located at top level of the package.

## Value

A list of length 2

| | |
|---|---|
| `head` | A sub list containing the HTML content within the <head> tag |
| `body` | A sub list containing the HTML content within the <body> tag |

## See Also

`add_server` `add_rmd`

## Examples

```
## Not run:

## ui.R from app: 'maximum_likelihood'

ui_ml <- system.file('apps',
                     'maximum_likelihood',
                     'server.R',
                     package = 'teachingApps')
browseURL(ui_ml)

## ui.R from app: 'distribution_weibull'

ui_dw <- system.file('apps',
                     'distribution_weibull',
                     'server.R',
                     package = 'teachingApps')
browseURL(ui_dw)


## End(Not run)
```

---

`add_update` *Add an update to a shiny app*

---

## Description

Pass app updates from a local inst directory to an app in an installed package

## Usage

```
add_update(local_pkg, ..., app_name, open_dir = FALSE, update_css = FALSE)
```

## Arguments

| | |
|---|---|
| `local_pkg` | `character` Path to the local version of the package from which updates will be passed |
| `...` | Additional directory names passed to `file.path()` (see details) |
| `app_name` | `character` Name of the app to be updated |
| `open_dir` | `logical` If `TRUE`, `browseURL()` is called to view the files in the app directory |
| `update_css` | `logical` If `TRUE` the css file is updated |

## Details

This function enables ultra-fast updates to shiny apps without needing to rebuild the package. It is assumed that two versions of a package exist on the user's machine. The first version is an installed package stored in the user's library, while the second version is a pre-compiled (in-work) version of the package. This function allows users to pass updates to an app from the in-work version of the package to the installed version while ensuring that the app can be deployed/published. Because files in the `inst/` directory aren't compiled when packages are built, updates can be passed to an installed package. This is useful for testing changes made to an app without re-building the package each time.

The `local_pkg` argument can be specified by providing a full file path to any file in the uncompiled version of the package. The root directory of the in-work package is located using `rprojroot::find_root`. The root directory of the installed version of the package is located using `devtools::inst()`. Any changes made to an app in the in-work package are passed to the app within the installed version of the package stored in the user's package library.

The `...` arguments are passed `file.path()` and name the directories between the package root directory and the `app_name/` directory. Note: the `inst` has already been provided and should not be included. For `update_css=TRUE` the `...` argument specifies the directories between the package root and the directory in which the css files are stored.

## Examples

```
# In the \code{teachingApps} package, apps are stored in the
# \code{inst/apps/} directory.
## Not run:
teachingApps::add_update(local_pkg = file.choose(),
                         'apps',
                         app_name = 'maximum_likelihood')

## End(Not run)

# Open an app directory to make and push updates

## Not run:
teachingApps::add_update(local_pkg = file.choose(),
                         'apps',
                         app_name = 'maximum_likelihood',
                         open_dir = TRUE)

## End(Not run)
```

---

# Birmbaum-Saunders *The Birmbaum-Saunders Distribution*

---

## Description

Density, distribution function, quantile function and random generation for the BISA distribution with location `loc` and scale `scale`.

## Usage

```
qbisa(p, shape, scale = 1)

pbisa(q, shape, scale = 1)

dbisa(x, shape, scale = 1)

rbisa(n, shape, scale = 1)
```

## Arguments

| | |
|---|---|
| p | Vector of probabilities |
| shape | Shape parameter |
| scale | Scale parameter |
| q | Vector of quantiles |
| x | Vector of quantiles |
| n | Number of observations |

## Details

If shape is not specified, a default value of 1 is used.

The Birmbaum-Saunders distribution with shape $\beta$ and scale $\theta$ has density

$$f(x;\theta,\beta) = \frac{\sqrt{\frac{x}{\theta}} + \sqrt{\frac{\theta}{x}}}{2\beta x}\phi_{NOR(z)}, \quad x \geq 0$$

where $\phi_{NOR}(z)$ is the density of the standard normal distribution and

$$z = \frac{1}{\beta}\left(\sqrt{\frac{x}{\theta}} - \sqrt{\frac{\theta}{x}}\right)$$

.

## Value

dbisa gives the density, pbisa gives the distribution function, qbisa gives the quantile function, and rbisa generates random observations.

The length of the result is determined by n for rbisa, and is the maximum of the lengths of the numerical arguments for the other functions.

The numerical arguments other than n are recycled to the length of the result.

## Source

Birnbaum, Z. W.; Saunders, S. C. (1969), "A new family of life distributions", Journal of Applied Probability, 6 (2): 319–327, JSTOR 3212003, doi:10.2307/3212003

`create_logo` *Create a branding logo for a teachingApp*

## Description

Create a branding logo for a teachingApp

## Usage

```
create_logo(app_dir = NULL, git_user = NULL, icon = NULL, img = NULL)
```

## Arguments

| | |
|---|---|
| `app_dir` | `character` Directory in which the app files are located |
| `git_user` | `character` GitHub account username (see details) |
| `icon` | `character` Name of a fontAwesome icon printed in the app footer |
| `img` | `character` Path to an image printed in the app footer |

## Details

This function should not be called directly, but is invoked by `add_optionss` when an app is rendered.

By default, the branding logo is the GitHub fontAwesome icon ('fa fa-github'). If `img` is specified, it takes precedence oven `icon`.

Hovering over the logo will reveal a link to view the code used to create the app. This is helpful in a teaching environment, where students often are interested in understanding how an app functions. The URL for the link is of the form (https://github.com`git_user`app_pkgblob/master/inst/apps) where `app_pkg` is created dynamically.

## Value

HTML code for inserting a logo (icon or image) in the footer of a navbarPage app

## See Also

`add_options`

`add_logo`

`Four Parameter Beta` *The Four Parameter Beta Distribution*

## Description

Density, distribution function, quantile function and random generation for the four parameter Beta distribution with minimum value `min` and scale `scale`.

## Usage

```
dbeta4(x, min, max, shape1, shape2, gap = 0)

pbeta4(q, min, max, shape1, shape2, gap = 0)

qbeta4(p, min, max, shape1, shape2)

rbeta4(n, min, max, shape1, shape2)
```

## Arguments

| | |
|---|---|
| `x` | Vector of quantiles |
| `min` | The minumum value on which the distribution is defined |
| `max` | The maximum value on which the distribution is defined |
| `shape1` | Shape parameter |
| `shape2` | Shape parameter |
| `gap` | Spacing from `min` and `max` |
| `q` | Vector of quantiles |
| `p` | Vector of probabilities |
| `n` | Number of observations |

## Details

If `shape` is not specified, a default value of 1 is used.

The Birmbaum-Saunders distribution with shape $\beta$ and scale $\theta$ has density

$$f(x;\theta,\beta) = \frac{\sqrt{\frac{x}{\theta}} + \sqrt{\frac{\theta}{x}}}{2\beta x}\phi_{NOR(z)}, \quad x \geq 0$$

where $\phi_{NOR}(z)$ is the density of the standard normal distribution and

$$z = \frac{1}{\beta}\left(\sqrt{\frac{x}{\theta}} - \sqrt{\frac{\theta}{x}}\right)$$

.

## Value

dbeta4 gives the density, pbeta4 gives the distribution function, qbeta4 gives the quantile function, and rbeta4 generates random observations.

The length of the result is determined by n for rbeta4, and is the maximum of the lengths of the numerical arguments for the other functions.

The numerical arguments other than n are recycled to the length of the result.

## Source

Birnbaum, Z. W.; Saunders, S. C. (1969), "A new family of life distributions", Journal of Applied Probability, 6 (2): 319–327, JSTOR 3212003, doi:10.2307/3212003

---

# gadget_clean_columns *Visually remove columns from data set*

---

## Description

Shiny gadget used to visually inspect columns in a data set and select columns to remove

## Usage

```
gadget_clean_columns(data, rownames = TRUE, theme = "flatly",
  width = "100%", height = "600px", css = NULL)
```

## Arguments

| | |
|---|---|
| data | A data set |
| rownames | logical Should rownames be included? |
| theme | character A bootswatch theme provided to shinythemes::shinytheme |
| width | character Width of the gadget (in valid css units) |
| height | character Height of the gadget (in valid css units) |
| css | character Path to a custom css file |

## Value

A list of length 2

| | |
|---|---|
| data | A data.frame containing the columns that were not removed |
| script | A line of code that can be used to replicate cleaning performed in the gadget |

A printed shiny app

## Examples

```
## Not run: clean_columns(mtcars)
```

## gadget_lm *Function Title*

### Description

Description

### Usage

```
gadget_lm(data, xvar, yvar, theme = "flatly", width = "100%", css = NULL,
  height = "600px", ...)
```

### Arguments

| | |
|---|---|
| data | A data.frame object |
| xvar | Column title (as a character-string) from data to display on the x-axis |
| yvar | Column title (as a character-string) from data to display on the y-axis |
| theme | character string naming a color theme bootswatch color theme. Must be one of the themes that can be used in shinythemes::shinytheme() |
| width | Width of the printed app. |
| css | Path to a custom css file. If NULL the default css file is used |
| height | Height of the printed app. |
| ... | Additional options passed to shiny::shinyAppDir() |

## inst *Get the installation path of a package*

### Description

Given the name of a package, this returns a path to the installed copy of the package, which can be passed to other functions.

### Usage

```
inst(name)
```

### Arguments

| | |
|---|---|
| name | the name of a package. |

### Details

It searches for the package in .libPaths(). If multiple dirs are found, it will return the first one.

### Source

Deprecated function from the devtools package

### Examples

```
inst("devtools")
inst("grid")
## Not run:
# Can be passed to other devtools functions
unload(inst("ggplot2"))

## End(Not run)
```

---

## jkf.par *Custom* par *function*

---

### Description

Custom par function

### Usage

```
jkf.par(...)
```

### Arguments

| | |
|---|---|
| ... | Parameter passed to par in addition to those defined |

---

## Largest Extreme Value *The Largest Extreme Value Distribution*

---

### Description

Density, distribution function, quantile function and random generation for the LEV distribution with location loc and scale scale.

### Usage

```
qlev(p, loc = 0, scale = 1)

plev(q, loc = 0, scale = 1)

dlev(x, loc = 0, scale = 1)

rlev(n, loc = 0, scale = 1)
```

### Arguments

| | |
|---|---|
| `p` | Vector of probabilities |
| `loc` | Location parameter |
| `scale` | Scale parameter |
| `q` | Vector of quantiles |
| `x` | Vector of quantiles |
| `n` | Number of observations |

### Details

If `loc` is not specified, a default value of 0 is used. If `scale` is not specified, a default value of 1 is used.

The largest extreme value distribution with location parameter $\mu$ and scale $\sigma$ has density

$$f(x;\mu,\sigma) = \frac{1}{\sigma}\phi_{LEV}\left(\frac{x-\mu}{\sigma}\right), \quad -\infty < x < \infty$$

where $\phi_{LEV}(z)$ exp[-z - exp(-z)] is the density of the standard LEV distribution.

### Value

`dlev` gives the density, `plev` gives the distribution function, `qlev` gives the quantile function, and `rlev` generates random observations.

The length of the result is determined by `n` for `rlev`, and is the maximum of the lengths of the numerical arguments for the other functions.

The numerical arguments other than `n` are recycled to the length of the result.

---

## `Smallest Extreme Value`

*The Smallest Extreme Value Distribution*

---

### Description

Density, distribution function, quantile function and random generation for the SEV distribution with location `loc` and scale `scale`.

### Usage

```
qsev(p, loc = 0, scale = 1)

psev(q, loc = 0, scale = 1)

dsev(x, loc = 0, scale = 1)

rsev(n, loc = 0, scale = 1)
```

### Arguments

| | |
|---|---|
| p | Vector of probabilities |
| loc | Location parameter |
| scale | Scale parameter |
| q | Vector of quantiles |
| x | Vector of quantiles |
| n | Number of observations |

### Details

If loc is not specified, a default value of 0 is used. If scale is not specified, a default value of 1 is used.

The smallest extreme value distribution with location parameter $\mu$ and scale $\sigma$ has density

$$f(x;\mu,\sigma) = \frac{1}{\sigma}\phi_{SEV}\left(\frac{x-\mu}{\sigma}\right), \quad -\infty < x < \infty$$

where $\phi_{SEV}(z)$ exp[z - exp(z)] is the density of the standard LEV distribution.

### Value

dsev gives the density, psev gives the distribution function, qsev gives the quantile function, and rsev generates random observations.

The length of the result is determined by n for rsev, and is the maximum of the lengths of the numerical arguments for the other functions.

The numerical arguments other than n are recycled to the length of the result.

---

## teachingApp *Render a teachingApp With Options*

---

### Description

Renders a teachingApp as a stand-alone shiny app or as an element within an rmarkdown document.

### Usage

```
teachingApp(app_name = NULL, theme = "flatly", width = "100%",
  height = "800px", icon = "fa fa-github", img = NULL,
  git_user = "Auburngrads", more_opts = list(NA), launch.browser = TRUE,
  ...)
```

## Arguments

| | |
|---|---|
| app_name | character Name of the app to be rendered |
| theme | character Name of a bootswatch color theme (provided by shinythemes::shinytheme) |
| width | character The width of the printed app (in pixels) |
| height | character The height of the printed app (in pixels) |
| icon | character A fontAwesome icon to be placed in the footer of a navbarPage app |
| img | character A path (or URL) to an image to be placed in the footer of a navbarPage app |
| git_user | character GitHub username used in the branding logo |
| more_opts | A list of additional options/objects that can be passed to the app (see Details) |
| launch.browser | logical If TRUE The app launches in the user's default browser |
| ... | A list of additional options passed to shiny::shinyAppDir() |

## Details

The teachingApps package provides an infrastructure that allows users to dynamically change the appearance and function of shiny apps. R users a familiar with writing functions to dynamically alter some output - in this case the output is a app. Normally,

## Value

A printed shiny app

## See Also

codelinkcreate_logo

codelinkadd_logo

## Examples

```
## Not run:
teachingApps(app_name = 'distribution_weibull',
             theme = 'spacelab',
             height = '800px')

teachingApps(app_name = 'maximum_likelihood_simulation',
             theme = 'flatly',
             height = '600px')

## End(Not run)
```

# Index