Package ‘fishRman’

March 27, 2024

Title The Fisheries Scientist's Toolbox

Version 1.2.3

Description A bundle of analytics tools for fisheries scientists. A 'shiny' R App is included for a 'no-code' solution for retrieval, analysis, and visualization.

License AGPL (>= 3)

Depends R (>= 2.10)

Imports config (>= 0.3.1), countrycode, dplyr, ggplot2, golem (>= 0.3.3), httr, jsonlite, maps, sf, shiny (>= 1.7.1), shinyBS, shinyjs, shinyWidgets, stats, utils, viridis

Suggests testthat (>= 3.0.0)

Config/testthat/edition 3

Encoding UTF-8

RoxygenNote 7.2.3

Date 2024-03-26

URL `https://github.com/Shyentist/fish-r-man`

BugReports `https://github.com/Shyentist/fish-r-man/issues`

NeedsCompilation no

Author Pasquale Buonomo [cre, aut] (<`https://orcid.org/0000-0002-1848-9313`>)

Maintainer Pasquale Buonomo <`pasqualebuonomo@hotmail.it`>

Repository CRAN

Date/Publication 2024-03-26 23:00:06 UTC

R topics documented:

df.type *Check the type of dataframe in relation to fishRman*

Description

Function to check the type of dataframe in relation to fishRman. Most of the times the user will not call this function directly, but it can be useful for debug.

Usage

```
df.type(df)
```

Arguments

df	The dataframe to check.

Value

A string explaining how the input dataframe is treated by fishRman.

Examples

```
dated <- c("2020-01-01", "2020-01-02")
lat <- c(40, 41)
lon <- c(12,13)
mmsi <- c("34534555", "25634555")
hours <- c(0, 5)
fishing_hours <- c(1,9)

df <- data.frame(dated, lat, lon, mmsi, hours, fishing_hours)

what.type <- df.type(df)

print(what.type)

# "GFW Fishing Effort By Vessel"
```

gfw.summarize *Summarize GFW data on fishing effort*

Description

Wrapper function for ‘dplyr::summarise()‘ that summarizes GFW data into the most important measures of central tendency for fishing_hours and hours, creating a new dataframe. It will have one (or more) rows for each combination of grouping variables; if there are no grouping variables, the output will have a single row summarising all observations in the input.

Usage

```
gfw.summarize(df)
```

Arguments

`df`	A dataframe object as downloaded from GFW's Google Big Data Query.

Value

A dataframe.

See Also

[dplyr::summarise()] [dplyr::group_by()]

Examples

```
dated <- c("2020-01-01", "2020-01-02")
lat <- c(40, 41)
lon <- c(12,13)
mmsi <- c("34534555", "25634555")
hours <- c(0, 5)
fishing_hours <- c(1,9)

df <- data.frame(dated, lat, lon, mmsi, hours, fishing_hours)

summary <- gfw.summarize(df)

print(summary)
```

`run_app` *Run the Shiny Application*

Description

Run the Shiny Application

Usage

```
run_app(
  onStart = NULL,
  options = list(),
  enableBookmarking = NULL,
  uiPattern = "/",
  ...
)
```

Arguments

`onStart` A function that will be called before the app is actually run. This is only needed for `shinyAppObj`, since in the `shinyAppDir` case, a `global.R` file can be used for this purpose.

`options` Named options that should be passed to the `runApp` call (these can be any of the following: "port", "launch.browser", "host", "quiet", "display.mode" and "test.mode"). You can also specify `width` and `height` parameters which provide a hint to the embedding environment about the ideal height/width for the app.

`enableBookmarking`
Can be one of `"url"`, `"server"`, or `"disable"`. The default value, `NULL`, will respect the setting from any previous calls to `enableBookmarking()`. See `enableBookmarking()` for more information on bookmarking your app.

`uiPattern` A regular expression that will be applied to each `GET` request to determine whether the `ui` should be used to handle the request. Note that the entire request path must match the regular expression in order for the match to be considered successful.

`...` arguments to pass to golem_opts. See '?golem::get_golem_options' for more details.

Value

Does not return a value. This function launches the embedded Shiny application, making it accessible through a web browser. The user interacts with the Shiny app through the browser interface.

top.percent.by *Subset the top percent of a dataframe by a specific column*

Description

Function that sorts a dataframe in descending order for a specific column, calculates the sum of all rows for that column, applies the chosen percentage to said sum, and subsets the minimum number of consecutive rows needed to reach this value.

Usage

```
top.percent.by(df, percentage, by)
```

Arguments

`df` A dataframe object as downloaded from GFW's Google Big Data Query.

`percentage` Number. The 'x' in 'the top x percent of the dataframe'.

`by` Character. The name of the column for which the percentage will be calculated.

Value

A dataframe.

Examples

```
dated <- c("2020-01-01", "2020-01-02")
lat <- c(40, 41)
lon <- c(12,13)
mmsi <- c("34534555", "25634555")
hours <- c(0, 5)
fishing_hours <- c(1,9)

df <- data.frame(dated, lat, lon, mmsi, hours, fishing_hours)

who.fishs.the.most <- top.percent.by(df, 90, "fishing_hours")

print(who.fishs.the.most)
```

Index