Package: icesUtils (via r-universe)

May 28, 2026

Version 0.1.3

Title Functions for processing ICES data and producing ICES shiny applications

Depends R (>= 4.3)

Imports htmltools, icesSAG, icesSD, icesVocab, readxl, jsonlite

Suggests shiny

Description A collection of functions useful for processing ICES data and making ICES shiny apps.

License GPL (>= 2)

URL https://github.com/ices-tools-dev/icesUtils, https://ices-tools-prod.r-universe.dev/icesUtils

BugReports https://github.com/ices-tools-dev/icesUtils/issues

RoxygenNote 7.3.3

Encoding UTF-8

Language en-GB

Config/pak/sysreqs libpng-dev libssl-dev

Repository https://ices-tools-prod.r-universe.dev

Date/Publication 2026-04-28 13:36:54 UTC

RemoteUrl https://github.com/ices-tools-dev/icesUtils

RemoteRef HEAD

RemoteSha cc1b41038e39e5f449c85b1802f6bf47156c4d84

Contents

`icesUtils-package` *Functions for processing ICES data and producing ICES shiny applications*

Description

A collection of functions useful for processing ICES data and making ICES shiny apps.

Details

Group of functions:

`first_function` example function

Author(s)

Maintainer: Colin Millar <colin.millar@ices.dk>

Authors:

- Luca Lamoni
- Neil Maginnis

References

ICES Stock Assessment Graphs database: http://sg.ices.dk.

ICES Stock Assessment Graphs web services: http://sg.ices.dk/webservices.aspx.

See Also

Useful links:

- https://github.com/ices-tools-dev/icesUtils
- https://ices-tools-prod.r-universe.dev/icesUtils
- Report bugs at https://github.com/ices-tools-dev/icesUtils/issues

`first_function` *Get summary data for a stock*

Description

Find the summary data for a specific stock in a given assessment year.

Usage

```
first_function(stock, year)
```

Arguments

`stock`	a stock name, e.g. cod-347d, or cod to find all cod stocks, or NULL (default) to process all stocks.
`year`	the assessment year, e.g. 2015, or 0 to process all years.

Details

This function retrieves summary data for a specified stock and year from the SAG and SD Databases. It first validates the stock code against the ICES vocabulary database, then fetches stock information from the Stock Data (SD) database, and the summary table data from the Stock Assessment Graphs (SAG) database.

The results from both sources are combined into a single dataframe which is returned to the user.

Value

A dataframe.

Author(s)

Colin Millar.

See Also

`getSAG` gets summary table from SAG DB. `icesSAG-package` gives an overview of the icesSAG package.

`icesUtils-package` gives an overview of the icesUtils package.

Examples

```
assessment_summary <- first_function("had.27.46a20", 2023)
head(assessment_summary)
```

get_bycatch_ecoregion *Retrieve bycatch advice results for a given ecoregion*

Description

Queries the ICES bycatch API and returns the bycatch advice results for a specified ecoregion.

Usage

```
get_bycatch_ecoregion(Ecoregion)
```

Arguments

Ecoregion	A character string giving the ecoregion name to be passed to the ICES bycatch API.

Details

This function builds the API request URL using the supplied ecoregion name, URL-encodes it, and parses the JSON response with ‘jsonlite::fromJSON()‘.

It does not perform validation of the API response, so downstream cleaning is usually needed before plotting or analysis.

Value

A data frame or list, depending on the API response structure, containing bycatch advice results for the requested ecoregion.

Examples

```
## Not run:
bycatch_raw <- get_bycatch_ecoregion("Greater North Sea")

## End(Not run)
```

image_fullscreen_on_click_js *JavaScript to view image fullscreen on click*

Description

Adds a JavaScript handler that lets users click on an image to enter/exit fullscreen

Usage

```
image_fullscreen_on_click_js()
```

Value

A script tag containing JavaScript code.

Examples

```
if (requireNamespace("shiny", quietly = TRUE) && interactive()) {
  library(shiny)

  addResourcePath(
    "mypkg",
    system.file("www", package = "icesUtils")
  )

  ui <- navbarPage(
  header = image_fullscreen_on_click_js(),
  title = "Example",
  "Click for full screen",
 tags$img(src = "mypkg/example-image.png", id = "demo-img", width = "300px", height = "300px")
  )

server <- function(input, output, session) {}

shinyApp(ui, server)
}
```

---

navbar_dropdown_autoclose_js

JavaScript to close navbar dropdown after navigation via actionbuttons or other non-navbar input

---

Description

Adds a JavaScript handler that closes open navbar dropdown menus when clicking outside the navbar.

Usage

```
navbar_dropdown_autoclose_js()
```

Value

A script tag containing JavaScript code.

Examples

```
if (requireNamespace("shiny", quietly = TRUE) && interactive()) {
  library(shiny)
  ui <-
navbarPage(header = navbar_dropdown_autoclose_js(),
```

```
            id = "main-navbar",
            title = "Example",
            navbarMenu(
              "Menu",
              tabPanel("A", value = "tab_a", actionButton("navigate2B", "Go to Tab B")),
              tabPanel("B", value = "tab_b", actionButton("navigate2A", "Go to Tab A"))
            )
  )


  server <- function(input, output, session) {
   observeEvent(input$navigate2B, {
     updateNavbarPage(session, inputId = "main-navbar", selected = "tab_b")
   })

   observeEvent(input$navigate2A, {
     updateNavbarPage(session, inputId = "main-navbar", selected = "tab_a")
   })
  }

  shinyApp(ui, server)
  }
```

`prepare_nested_text_lists_from_excel` *Create a named nested list of dataframes containing text content, from excel files and a vector of names.*

Description

Uses `prepare_text_from_excel` to make a nested list containing text content extractable by *section*. Requires that *paths* and *names* are vectors of equal length. Output works with `select_text` where *names* gives the *top_level* parameter to `select_text`. vector of names an excel file into a list of dataframes. Works with `usethis::use_data` and `select_text` to provide a simple means of incorporating text in shiny applications. To work with `usethis::use_data` the excel file should have named tabs,

Usage

```
prepare_nested_text_lists_from_excel(paths, names)
```

Arguments

`paths`	valid paths provided either as vector or list
`names`	list or vector of characters

See Also

prepare_text_from_excel converts texts provided via a single excel file into a structure that select_text can work with. Used by this function on each file provided.

select_text extracts individual texts by tab, section and, optionally, top-level names.

---

prepare_text_from_excel *Generate a list of dataframes containing text content, from excel*

Description

Converts an excel file into a list of dataframes. Works with usethis::use_data and select_text to provide a simple means of incorporating text in shiny applications. To work with usethis::use_data the excel file should have named tabs, each tab having 2 columns, **section** and **text**. *text* is the content, *section* is used by select_text to extract the relevant section. Texts provided in *text* are wrapped in the HTML paragraph tag <p>.

Usage

```
prepare_text_from_excel(path_to_file)
```

Arguments

path_to_file	path to the file containing text content

Value

a list of dataframes

See Also

prepare_nested_text_lists_from_excel converts texts in multiple files into a hierarchical structure that select_text can work with.

select_text extracts individual texts by tab, section and, optionally, top-level names.

Examples

```
example_path <- system.file("extdata", "text_function_example.xlsx", package = "icesUtils")
prepare_text_from_excel(path_to_file = example_path)
```

select_text *Get text to display from list of dfs.*

Description

display_text´ subsets a list of dataframes and extracts the relevant section of text for tab and section If top_level is supplied, accepts a named list of lists, and filters the top level list by top_level name.

Usage

```
select_text(list_of_texts, tab, section = NULL)
```

Arguments

list_of_texts	a list of dataframes each containing 'section' and 'text' columns
tab	a character vector - the name of a list entry
section	a character vector indicating which row to extract

Value

A character string

Author(s)

Neil Maginnis

See Also

prepare_text_from_excel is a function that converts texts provided via an excel file into a structure that select_text can work with.

prepare_nested_text_lists_from_excel converts texts in multiple files into a hierarchical structure that select_text can work with.

Examples

```
greetings <- data.frame(section = c("welcome", "goodbye"),
text = c("Hello world", "Thanks for stopping by"))
content <- data.frame(section = "help", text = "Use the function as shown here")
texts <- list(greetings=greetings, content = content)
select_text(list_of_texts = texts, "greetings", "welcome")
select_text(list_of_texts = texts, "content", "help")
select_text(list_of_texts = texts, "greetings", "goodbye")
```

Index