Package: shinyExprPortal (via r-universe)

September 13, 2024

Title A Configurable 'shiny' Portal for Sharing Analysis of Molecular Expression Data

Version 1.2.1.9000

Maintainer Rafael Henkin <`r.henkin@qmul.ac.uk`>

Description Enables deploying configuration file-based 'shiny' apps with minimal programming for interactive exploration and analysis showcase of molecular expression data. For exploration, supports visualization of correlations between rows of an expression matrix and a table of observations, such as clinical measures, and comparison of changes in expression over time. For showcase, enables visualizing the results of differential expression from package such as 'limma', co-expression modules from 'WGCNA' and lower dimensional projections.

Depends R (>= 3.5.0)

License MIT + file LICENSE

Imports config, stats, utils, shiny, htmltools, markdown, cli, dplyr, tidyr, yaml, data.table, bslib, iheatmapr, vegawidget, DT, qvalue, parallel, Rfast, rlang, shinyhelper

Encoding UTF-8

RoxygenNote 7.3.2

Roxygen list(markdown = TRUE)

Suggests testthat (>= 3.0.0), whisker, knitr, rmarkdown, r2d3, kableExtra, RColorBrewer

VignetteBuilder knitr

URL `https://c4tb.github.io/shinyExprPortal/`, `https://github.com/C4TB/shinyExprPortal`

Config/testthat/edition 3

BugReports `https://github.com/C4TB/shinyExprPortal/issues`

Repository https://c4tb.r-universe.dev

RemoteUrl https://github.com/c4tb/shinyexprportal

RemoteRef HEAD

RemoteSha b8a1bf982d3868be6d027a8dac820058edbac87c

Contents

`create_config_template` *Create a bare-bones configuration file template*

Description

The resulting file contain placeholder text in upper case for you to edit according to your needs. It also includes the three correlation modules by default.

Usage

```
create_config_template(target_dir, filename = "config.yaml")
```

Arguments

`target_dir`	location to create the configuration file
`filename`	optional file name, default is config.yaml

Value

Creates configuration file in `target_dir`

Examples

```
if (interactive()) {
    dir.create("newapp")
    create_config_template("newapp")
}
```

`create_config_wizard` *Create configuration and app.R files*

Description

This function runs an interactive wizard that guides the user through the creation of a basic configuration file. The wizard will work with the simple case of expression data where one sample matches exactly to one subject.

Usage

```
create_config_wizard(target_dir)
```

Arguments

`target_dir`	location where the configuration will be saved

Details

Before you run the wizard, you should ensure that the target folder contains at least the expression matrix and measures data files. The expression matrix should follow the format of sample IDs in columns and genes in rows, with gene names in the first column of the table. The measures file should follow the format of subjects in rows and measures in columns, and you should ensure that all subjects have one sample and vice-versa.

Value

Creates configuration file in `target_dir`

Examples

```
if (interactive()) {
    dir.create("newapp")
    create_config_wizard("newapp")
}
```

`create_example` *Create example files*

Description

Create example files for measures, expression matrix and lookup table

Usage

```
create_example(target_dir)
```

Arguments

`target_dir`	location where to create the files

Value

Create examples files in target_dir

Examples

```
if (interactive()) {
    dir.create("newapp")
    create_example("newapp")
}
```

`create_module_template` *Creates a module code template in current working directory*

Description

Creates a module code template in current working directory

Usage

```
create_module_template(module_name, target_dir = "")
```

Arguments

`module_name`	module name in camelCase
`target_dir`	Optional folder where to save the file. Saves in current folder otherwise.

Value

Create file for `module_name` in current working directory

Examples

```
if (interactive()) {
    create_module_template("newModule")
}
```

run_app *Run the Shiny Application*

Description

This function should be run only after you have created the configuration file and placed all required files in the app folder. See `vignette("quickstart", package = "shinyExprPortal")` for help with setup or `vignette("fullguide", package = "shinyExprPortal")` for a complete configuration guide.

Usage

```
run_app(
  config_file,
  data_folder = "",
  custom_modules = NULL,
  nthreads = 1L,
  ...
)
```

Arguments

`config_file`	The name of the yaml configuration file
`data_folder`	Optional directory prefix for data files. Use this argument if you want to version your files across different folders
`custom_modules`	Optional list of available custom modules. See the 'Details' section.
`nthreads`	Optional number of threads/cores to speed up loading files and computing correlations on UNIX-based systems. Default is 1
`...`	Further optional arguments.

Details

`custom_modules` should contain a list of names for user-defined modules that are loaded in the environment before calling run_app. Each module should be accompanied by the corresponding mod_moduleName_ui, mod_moduleName_server moduleName_config functions. These functions could be placed in a custom_modules.R file, for example, and loaded using `source`. The package will then parse the configuration file, and if it contains one of the custom module names, it will call the module configuration parsing function and add it to the interface. See `vignette("customization")` for a complete example.

Please note that if running on Windows, nthreads will be always set to 1 due to limitations on the current implementation.

Value

Runs the app

See Also

`create_config_wizard()` to create a configuration using a wizard, `create_config_template()` to create a configuration file template.

Examples

```
if (interactive()) {
run_app("config.yaml", nthreads = 4)
}
```

`show_available_modules` *Print list of currently supported modules*

Description

See `vignette("config", package = "shinyExprPortal")` for details on how to configure each module.

Usage

```
show_available_modules()
```

Value

list of available modules

Examples

```
show_available_modules()
```

Index