Package: kwb.code (via r-universe)

September 7, 2024

Title Analyse Your R Code!

Version 0.3.0

Description This package allows you to parse your R scripts and to calculate some staticstics on your code.

License MIT + file LICENSE

URL https://github.com/KWB-R/kwb.code

BugReports https://github.com/KWB-R/kwb.code/issues

Imports dplyr, kwb.file, kwb.utils, stringr

Suggests covr, knitr, rmarkdown, testthat (>= 3.0.0)

VignetteBuilder knitr

Remotes github::kwb-r/kwb.file, github::kwb-r/kwb.utils

ByteCompile true

Config/testthat/edition 3

Encoding UTF-8

LazyData true

RoxygenNote 7.2.3

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

RemoteUrl https://github.com/KWB-R/kwb.code

RemoteRef HEAD

RemoteSha bc81324403e3881124fa2230c023807eba26e32d

Contents

analyse *Analyse the Parse Tree of an R Script*

Description

Analyse the Parse Tree of an R Script

Usage

```
analyse(x, path = "")
```

Arguments

x	parse tree as returned by parse
path	for internal use only (when this function is called recursively)

Value

list representing type information on the nodes in the parse tree

Examples

```
# Parse an R script file (here, a file from kwb.utils)
x <- parse("https://raw.githubusercontent.com/KWB-R/kwb.utils/master/R/log.R")

# Analyse the parse tree (This may take some time!)
result <- kwb.code::analyse(x)

# Show the structure of the result list (only 3 levels!)
str(result, 3)
```

`arg_names` *Get Argument Names of a Function*

Description

Get Argument Names of a Function

Usage

```
arg_names(x)
```

Arguments

`x`	function name or function

Value

vector of character

Examples

```
arg_names("sum")
arg_names(mean)
```

`extract_from_parse_tree` *Extract Elements from Parse Tree of R Script*

Description

The idea of this function is to collect objects of interest from the parse tree, e.g. the names of functions that are called by a script. Therefore, set the function `matches` so that it returns `TRUE` for the nodes in the tree that are of interest.

Usage

```
extract_from_parse_tree(
  x,
  matches = matches_function,
  dbg = FALSE,
  path = integer(),
  parent = NULL,
  index = -1
)
```

Arguments

`x`	parse tree as returned by `parse`
`matches`	function that is called for each node of the tree. Give a function here that returns `TRUE` if the object is to be selected and `FALSE` else. The value of `TRUE` must be given an attribute `name` that is expected to be a character of length one.
`dbg`	if `TRUE` each node in printed during the climing of the tree
`path`	for internal use
`parent`	for internal use
`index`	for internal use

Value

vector of character or `NULL`

`find_string_constants` *Show String Constants Used in R Scripts*

Description

Show String Constants Used in R Scripts

Usage

```
find_string_constants(root = "./R")
```

Arguments

`root`	path from which to look recursively for R scripts

`find_weaknesses_in_scripts` *Find weaknesses in R scripts*

Description

Find weaknesses in R scripts

Usage

```
find_weaknesses_in_scripts(
  x = parse_scripts(root),
  root = NULL,
  min_duplicate_string_length = 6L,
  min_duplicate_frequency = 3L
)
```

Arguments

x	list of named parse trees as returned by parse_scripts. Not required if root is given.
root	path to folder containing R scripts
min_duplicate_string_length	minimum number of characters that a string constant must have to be considered as a duplicate
min_duplicate_frequency	minimum frequency of a string constant to be considered as a duplicate

Value

data frame with columns file, expression, frequency, recommendation

---

get_elements_by_type *Extract Sections of Same "Type" from Parse Tree*

Description

Extract Sections of Same "Type" from Parse Tree

Usage

```
get_elements_by_type(x, result = NULL, dbg = TRUE)
```

Arguments

x	parse tree as returned by parse
result	optional. Result as returned by analyse
dbg	if TRUE, debug messages are shown

Examples

```
# Parse an R script file (here, a file from kwb.utils)
x <- parse("https://raw.githubusercontent.com/KWB-R/kwb.utils/master/R/log.R")

# For each "type" of code segment, extract all occurrences
elements <- get_elements_by_type(x)

# Show all for-loops
elements$`language|call|for|4|`

# Show all if-statements
elements$`language|call|if|3|`

# Show all if-else-statements
elements$`language|call|if|4|`
```

`get_full_function_info` *Get information on function definitions in parsed R scripts*

Description

Get information on function definitions in parsed R scripts

Usage

```
get_full_function_info(trees)
```

Arguments

`trees`	list of R script parse trees as provided by `parse_scripts`

See Also

`parse_scripts`

`get_function_call_frequency` *Which Function is Called How Often?*

Description

Which Function is Called How Often?

Usage

```
get_function_call_frequency(tree, simple = FALSE, dbg = TRUE)
```

Arguments

`tree`	parse tree as returned by `parse_scripts`
`simple`	if `TRUE`, a simple approach using a simple regular expression is used. This approach is fast but not correct as it e.g. counts function calls that are commented out or even string expressions that just look like function calls. Leaving this argument to its default, `FALSE`, will return only real function calls by evaluating the full structure of parse tree.
`dbg`	if `TRUE`, debug messages are shown

Value

data frame with columns `name` (name of function), `count` (number of times the function is called)

`get_names_of_used_packages` *Get Names of Packages Used in R-Scripts*

Description

Get Names of Packages Used in R-Scripts

Usage

```
get_names_of_used_packages(root_dir, pattern = "[.][rR](md)?$")
```

Arguments

`root_dir`	directory in which to look recursively for R-scripts
`pattern`	regular expression matching the names of the files to be considered

`get_package_function_usage` *How Often Are the Functions of a Package Used?*

Description

How Often Are the Functions of a Package Used?

Usage

```
get_package_function_usage(tree, package, simple = FALSE, by_script = FALSE)
```

Arguments

`tree`	parse tree as returned by `parse_scripts`
`package`	name of the package (must be installed)
`simple`	if `TRUE`, a simple approach using a simple regular expression is used. This approach is fast but not correct as it e.g. counts function calls that are commented out or even string expressions that just look like function calls. Leaving this argument to its default, `FALSE`, will return only real function calls by evaluating the full structure of parse tree.
`by_script`	if `TRUE` the functions are counted and returned by script, otherwise they are counted over all scripts

Value

data frame with columns `name` (name of the function), `prefixed` (number of function calls prefixed with <package>`::` or <package>`:::`), `non_prefixed` (number of function calls that are not prefixed with the package name) and `total` (total number of function calls)

Examples

```
# Read all scripts that are provided in the kwb.fakin package
tree <- kwb.code::parse_scripts(root = system.file(package = "kwb.fakin"))

# Check which functions from kwb.utils are used and how often
get_package_function_usage(tree, package = "kwb.utils")

# Hm, this does not seem to be the whole truth...
```

`get_package_usage_per_script` *Get Package Usage per Script*

Description

Get Package Usage per Script

Usage

```
get_package_usage_per_script(root, packages, pattern = "\\.R$", ...)
```

Arguments

`root`	root directory with R scripts
`packages`	vector with package names to be checked
`pattern`	default: "\.R$"
`...`	additional arguments passed to `get_package_function_usage`

Value

tibble with information on used packages

`get_string_constants_in_scripts` *Get Frequency of String Constant Usage in R Scripts*

Description

Get Frequency of String Constant Usage in R Scripts

Usage

```
get_string_constants_in_scripts(
  root,
  scripts = dir(root, "\\.[Rr]$", recursive = TRUE),
  two_version_check = TRUE,
  FUN = NULL
)
```

Arguments

`root`	path to folder in which to look for R scripts
`scripts`	optional. Paths to R scripts in which to search for string constants, relative to `root`
`two_version_check`	if `TRUE` (default), two different implementations of this function are used and the results are compared internally. Set this argument to `FALSE` to get the result as fast as possible.
`FUN`	optional. Function used to browse the code tree for string constants. If `NULL` (the default), `kwb.code:::fetch_string_constants_1` is used.

Value

data frame with columns `file_id` (file identifier), `string` (string constant found in the file) and `count` (number of occurrences of the string counted in the file). The file identifier can be resolved to a full file name using the "file database" that is stored in the attribute "file_db".

Examples

```
root <- system.file(package = "kwb.code")
constants <- get_string_constants_in_scripts(root)

# Get paths to files from "file database" stored in attribute "file_db"
kwb.utils::getAttribute(constants, "file_db")
```

`parse_scripts` *Parse all given R scripts into a tree structure*

Description

Parse all given R scripts into a tree structure

Usage

```
parse_scripts(
  root,
  scripts = dir(root, "\\.R$", ignore.case = TRUE, recursive = TRUE),
  dbg = TRUE
)
```

Arguments

`root`	root directory to which the relative paths given in `scripts` relate
`scripts`	relative file paths to R scripts. By default all files ending with ".R" or ".r" below the `root` folder (recursively) are parsed.
`dbg`	if `TRUE` debug messages are shown

See Also

`to_full_script_info`

Examples

```
## Not run:
# Download some example code files from github...
url.base <- "https://raw.githubusercontent.com/hsonne/blockrand2/master/R/"
urls <- paste0(url.base, c("blockrand2_create.R", "blockrand2_main.R"))

targetdir <- file.path(tempdir(), "blockrand2")
targetdir <- kwb.utils::createDirectory(targetdir)

for (url in urls) {
  download.file(url, file.path(targetdir, basename(url)))
}

# By default, all R scripts below the root are parse
trees <- parse_scripts(root = targetdir)

# All elements of trees are expressions
sapply(trees, is.expression)

# Analyse the scripts on the script level
scriptInfo <- to_full_script_info(trees)

scriptInfo

# Analyse the scripts on the function level
functionInfo <- get_full_function_info(trees)

functionInfo

## End(Not run)
```

`to_full_script_info` *Get script statistics from a list of R script trees*

Description

Get script statistics from a list of R script trees

Usage

```
to_full_script_info(trees)
```

Arguments

`trees`	list of R script parse trees as provided by `parse_scripts`

See Also

`parse_scripts`

`walk_tree` *Walk Along a Parse Tree*

Description

Walk Along a Parse Tree

Usage

```
walk_tree(
  x,
  path = "",
  depth = 0L,
  max_depth = 20L,
  dbg = TRUE,
  config = list(),
  context = NULL
)
```

Arguments

`x`	parse tree as returned by `parse` or a sub-tree of the parse tree
`path`	for internal use only. Path to the element in the parse tree.
`depth`	for internal use only. Recursion depth.
`max_depth`	maximum recursion level. Default: 20L
`dbg`	whether or not to show debug messages
`config`	list defining modifications of nodes in the node tree. TODO: describe further
`context`	if not `NULL` (the default) this is expected to be a list containing additional data. Currently list element "file" is used to pass the name of the script that the current tree was read from.

Examples

```
walk_tree(parse(text = "x <- 1:n"))
```

Index