Package: dupree (via r-universe)

September 30, 2024

Type Package

Title Identify Duplicated R Code in a Project

Version 0.3.0.9000

Author Russ Hyde

Maintainer Russ Hyde <russ.hyde.data@gmail.com>

Description Identifies code blocks that have a high level of similarity within a set of R files.

URL https://russhyde.github.io/dupree/, https://github.com/russHyde/dupree

BugReports https://github.com/russHyde/dupree/issues

License MIT + file LICENSE

Encoding UTF-8

Language en-GB

LazyData true

Suggests testthat (>= 2.1.0), knitr, rmarkdown, covr

Imports dplyr (>= 1.1.0), purrr, tibble, magrittr, methods, stringdist (>= 0.9.5.5), lintr (>= 3.0.0), rlang

RoxygenNote 7.2.2

Collate 'utils.R' 'dupree.R' 'dupree_classes.R' 'dupree_data_validity.R' 'dupree_code_enumeration.R' 'dups-class.R'

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

RemoteUrl https://github.com/russhyde/dupree

RemoteRef HEAD

RemoteSha 6be55893a839717d36e6b3e21bdd662951a59bf8

Contents

`as.data.frame.dups` *as.data.frame method for ‘dups‘ class*

Description

as.data.frame method for ‘dups‘ class

Usage

```
## S3 method for class 'dups'
as.data.frame(x, ...)
```

Arguments

`x`	any R object.
`...`	additional arguments to be passed to or from methods.

`as_tibble.dups` *convert a ‘dups‘ object to a ‘tibble‘*

Description

convert a ‘dups‘ object to a ‘tibble‘

Usage

```
## S3 method for class 'dups'
as_tibble(x, ...)
```

Arguments

`x`	A data frame, list, matrix, or other object that could reasonably be coerced to a tibble.
`...`	Unused, for extensibility.

dupree *Detect code duplication between the code-blocks in a set of files*

Description

This function identifies all code-blocks in a set of files and then computes a similarity score between those code-blocks to help identify functions / classes that have a high level of duplication, and could possibly be refactored.

Usage

```
dupree(files, min_block_size = 40, ...)
```

Arguments

`files`	A set of files over which code-duplication should be measured.
`min_block_size`	`dupree` uses a notion of non-trivial symbols. These are the symbols / code-words that remain after filtering out really common symbols like `<-`, `,`, etc. After filtering out these symbols from each code-block, only those blocks containing at least `min_block_size` symbols are used in the inter-block code-duplication measurement.
`...`	Unused at present.

Details

Code-blocks under a size threshold are disregarded before analysis (the size threshold is controlled by `min_block_size`); and only top-level code blocks are considered.

Every sufficiently large code-block in the input files will be present in the results at least once. If code-block X and code-block Y are present in a row of the resulting data-frame, then either X is the closest match to Y, or Y is the closest match to X (or possibly both) according to the similarity score; as such, some code-blocks may be present multiple times in the results.

Similarity between code-blocks is calculated using the longest-common-subsequence (`lcs`) measure from the package `stringdist`. This measure is applied to a tokenised version of the code-blocks. That is, each function name / operator / variable in the code blocks is converted to a unique integer so that a code-block can be represented as a vector of integers and the `lcs` measure is applied to each pair of these vectors.

Value

A `tibble`. Each row in the table summarises the comparison between two code-blocks (block 'a' and block 'b') in the input files. Each code-block in the pair is indicated by: i) the file (`file_a` / `file_b`) that contains it; ii) its position within that file (`block_a` / `block_b`; 1 being the first code-block in a given file); and iii) the line where that code-block starts in that file (`line_a` / `line_b`). The pairs of code-blocks are ordered by decreasing similarity. Any match that is returned is either the top hit for block 'a' or for block 'b' (or both).

Examples

```
# To quantify duplication between the top-level code-blocks in a file
example_file <- system.file("extdata", "duplicated.R", package = "dupree")
dup <- dupree(example_file, min_block_size = 10)
dup

# For the block-pair with the highest duplication, we print the first four
# lines:
readLines(example_file)[dup$line_a[1] + c(0:3)]
readLines(example_file)[dup$line_b[1] + c(0:3)]

# The code-blocks in the example file are rather small, so if
# `min_block_size` is too large, none of the code-blocks will be analysed
# and the results will be empty:
dupree(example_file, min_block_size = 40)
```

`dupree_dir` *Run duplicate-code detection over all R-files in a directory*

Description

Run duplicate-code detection over all R-files in a directory

Usage

```
dupree_dir(
  path = ".",
  min_block_size = 40,
  filter = NULL,
  ...,
  recursive = TRUE
)
```

Arguments

`path` A directory (By default the current working directory). All files in this directory that have a ".R", ".r" or ".Rmd" extension will be checked for code duplication.

`min_block_size` `dupree` uses a notion of non-trivial symbols. These are the symbols / code-words that remain after filtering out really common symbols like `<-`, `,`, etc. After filtering out these symbols from each code-block, only those blocks containing at least `min_block_size` symbols are used in the inter-block code-duplication measurement.

`filter` A pattern for use in grep - this is used to keep only particular files: eg, filter = "classes" would compare files with ‘classes‘ in the filename

`...` Further arguments for grep. For example, ‘filter = "test", invert = TRUE‘ would disregard all files with ‘test‘ in the file-path.

`recursive` Should we consider files in subdirectories as well?

See Also

dupree

---

dupree_package *Run duplicate-code detection over all files in the ‘R‘ directory of a package*

Description

The function fails if the path does not look like a typical R package (it should have both an R/ subdirectory and a DESCRIPTION file present).

Usage

```
dupree_package(package = ".", min_block_size = 40)
```

Arguments

`package` The name or path to the package that is to be checked (By default the current working directory).

`min_block_size` `dupree` uses a notion of non-trivial symbols. These are the symbols / code-words that remain after filtering out really common symbols like `<-`, `,`, etc. After filtering out these symbols from each code-block, only those blocks containing at least `min_block_size` symbols are used in the inter-block code-duplication measurement.

See Also

dupree

EnumeratedCodeTable-class *An S4 class to represent the code blocks as strings of integers*

Description

An S4 class to represent the code blocks as strings of integers

Slots

`blocks` A tbl_df with columns ‘file‘, ‘block‘, ‘start_line‘ and ‘enumerated_code‘

print.dups *print method for ‘dups‘ class*

Description

print method for ‘dups‘ class

Usage

```
## S3 method for class 'dups'
print(x, ...)
```

Arguments

`x`	an object used to select a method.
`...`	further arguments passed to or from other methods.

Index