Package 'seekr'

May 5, 2025

Title Extract Matching Lines from Matching Files

Version 0.1.2

Description Provides a simple interface to recursively list files from a directory, filter them using a regular expression, read their contents, and extract lines that match a user-defined pattern. The package returns a dataframe containing the matched lines, their line numbers, file paths, and the corresponding matched substrings. Designed for quick code base exploration, log inspection, or any use case involving pattern-based file and line filtering.

License MIT + file LICENSE

Encoding UTF-8

RoxygenNote 7.3.2

Suggests testthat (>= 3.0.0), withr

Config/testthat/edition 3

Imports checkmate, cli, fs, lifecycle, purrr, readr, stringr, tibble, tidyr

URL https://github.com/smartiing/seekr, https://smartiing.github.io/seekr/

BugReports https://github.com/smartiing/seekr/issues

NeedsCompilation no

Author Sacha Martingay [aut, cre, cph]

Maintainer Sacha Martingay <martingay.sacha@hotmail.com>

Repository CRAN

Date/Publication 2025-05-05 09:10:02 UTC

Contents

seek *Extract Matching Lines from Files*

Description

These functions search through one or more text files, extract lines matching a regular expression pattern, and return a tibble containing the results.

- `seek()`: Discovers files inside one or more directories (recursively or not), applies optional file name and text file filtering, and searches lines.
- `seek_in()`: Searches inside a user-provided character vector of files.

Usage

```
seek(
  pattern,
  path = ".",
  ...,
  filter = NULL,
  negate = FALSE,
  recurse = FALSE,
  all = FALSE,
  relative_path = TRUE,
  matches = FALSE
)

seek_in(files, pattern, ..., matches = FALSE)
```

Arguments

`pattern`	A regular expression pattern used to match lines.
`path`	A character vector of one or more directories where files should be discovered (only for `seek()`).
`...`	Additional arguments passed to `readr::read_lines()`, such as `skip`, `n_max`, or `locale`.
`filter`	Optional. A regular expression pattern used to filter file paths before reading. If `NULL`, all text files are considered.
`negate`	Logical. If `TRUE`, files matching the `filter` pattern are excluded instead of included. Useful to skip files based on name or extension.
`recurse`	If `TRUE` recurse fully, if a positive number the number of levels to recurse.
`all`	If `TRUE` hidden files are also returned.
`relative_path`	Logical. If TRUE, file paths are made relative to the path argument. If multiple root paths are provided, relative_path is automatically ignored and absolute paths are kept to avoid ambiguity.
`matches`	Logical. If `TRUE`, all matches per line are also returned in a `matches` list-column.
`files`	A character vector of files to search (only for `seek_in()`).

Details

[Experimental]

The overall process involves the following steps:

- **File Selection**
  - `seek()`: Files are discovered using `fs::dir_ls()`, starting from one or more directories.
  - `seek_in()`: Files are directly supplied by the user (no discovery phase).
- **File Filtering**
  - Files located inside `.git/` folders are automatically excluded.
  - Files with known non-text extensions (e.g., `.png`, `.exe`, `.rds`) are excluded.
  - If a file's extension is unknown, a check is performed to detect embedded null bytes (binary indicator).
  - Optionally, an additional regex-based path filter (`filter`) can be applied.
- **Line Reading**
  - Files are read line-by-line using `readr::read_lines()`.
  - Only lines matching the provided regular expression `pattern` are retained.
  - If a file cannot be read, it is skipped gracefully without failing the process.
- **Data Frame Construction**
  - A tibble is constructed with one row per matched line.

These functions are particularly useful for analyzing source code, configuration files, logs, and other structured text data.

Value

A tibble with one row per matched line, containing:

- `path`: File path (relative or absolute).
- `line_number`: Line number in the file.
- `match`: The first matched substring.
- `matches`: All matched substrings (if `matches = TRUE`).
- `line`: Full content of the matching line.

See Also

`fs::dir_ls()`, `readr::read_lines()`, `stringr::str_detect()`

Examples

```
path = system.file("extdata", package = "seekr")

# Search all function definitions in R files
seek("[^\\s]+(?= (=|<-) function\\()", path, filter = "\\.R$")

# Search for usage of "TODO" comments in source code in a case insensitive way
seek("(?i)TODO", path, filter = "\\.R$")
```

```
# Search for error/warning in log files
seek("(?i)error", path, filter = "\\.log$")

# Search for config keys in YAML
seek("database:", path, filter = "\\.ya?ml$")

# Looking for "length" in all types of text files
seek("(?i)length", path)

# Search for specific CSV headers using seek_in() and reading only the first line
csv_files <- list.files(path, "\\.csv$", full.names = TRUE)
seek_in(csv_files, "(?i)specie", n_max = 1)
```

Index