# Package 'ahocorasick'

June 2, 2026

**Title** Fast Multi-Pattern String Matching with the 'Aho-Corasick' Algorithm

**Version** 0.2.0

**Description** Provide fast multi-pattern string matching for 'R' using the 'Aho-Corasick' algorithm, powered by the 'Rust' 'aho-corasick' crate. It builds reusable automatons for detecting matches, counting matches, locating character, extracting matched text, and replacing matches in character vectors. For more details on the 'Aho-Corasick' algorithm, please see Aho and Corasick (1975) <doi:10.1145/360825.360855>.

**License** MIT + file LICENSE

**URL** https://yousa-mirage.github.io/r-ahocorasick/, https://github.com/Yousa-Mirage/r-ahocorasick

**BugReports** https://github.com/Yousa-Mirage/r-ahocorasick/issues

**Encoding** UTF-8

**Config/testthat/edition** 3

**Config/testthat/parallel** true

**Config/rextendr/version** 0.5.0

**SystemRequirements** Cargo (Rust's package manager), rustc >= 1.65.0, xz

**Depends** R (>= 4.2)

**Imports** checkmate, cli, fs, rlang

**Suggests** dplyr, knitr, pkgdown, rmarkdown, tibble, tidyr, testthat (>= 3.0.0)

**VignetteBuilder** knitr

**Language** en-US

**Config/roxygen2/version** 8.0.0

**NeedsCompilation** yes

**Author** Hao Cheng [aut, cre, cph]

**Maintainer** Hao Cheng <Yousa-Mirage@foxmail.com>

**Repository** CRAN

**Date/Publication** 2026-06-02 11:00:07 UTC

# Contents



---

`ac_build` *Build an Aho-Corasick automaton*

---

## Description

`ac_build()` compiles a character vector of patterns into a reusable automaton backed by the Rust `aho-corasick` crate.

## Usage

```
ac_build(
  patterns,
  match_kind = c("standard", "leftmost_first", "leftmost_longest"),
  implementation = c("auto", "noncontiguous_nfa", "contiguous_nfa", "dfa"),
  ascii_case_insensitive = FALSE,
  duplicate = c("keep", "error", "deduplicate")
)
```

## Arguments

| | |
|---|---|
| `patterns` | A character vector of non-empty patterns. |
| `match_kind` | Matching semantics:<br>• `"standard"` supports overlapping search (Default).<br>• `"leftmost_first"` returns leftmost non-overlapping matches, breaking ties by pattern order.<br>• `"leftmost_longest"` returns leftmost non-overlapping matches, breaking ties by longest match. |

`implementation` Rust automaton implementation. `"auto"` lets the crate choose.

`ascii_case_insensitive`
: Use ASCII-only case-insensitive matching. Default is `FALSE`.

`duplicate` How duplicate patterns are handled:

- `"keep"` preserves duplicates in their original order.
- `"error"` fails if `patterns` contains duplicates.
- `"deduplicate"` keeps the first occurrence of each pattern and drops later duplicates.

## Value

An immutable `<ac_automaton>` object.

## See Also

`ac_locate()`, `ac_locate_df()`, `ac_detect()`, `ac_count()`, `ac_extract()`, `ac_extract_df()`, `ac_replace()`, `ac_patterns()`.

## Examples

```
ac <- ac_build(c("hello", "world"))
length(ac)
ac_info(ac)
```

---

# ac_count *Count pattern matches in documents*

---

## Description

`ac_count()` returns the number of pattern matches in each document.

## Usage

```
ac_count(ac, doc, overlapping = FALSE, na = c("keep", "zero", "error"))
```

## Arguments

`ac` An `<ac_automaton>` object created by `ac_build()`.

`doc` A character vector of documents to search.

`overlapping` Default is `FALSE`. If `TRUE`, count overlapping matches. This is only supported when `ac` was built with `match_kind = "standard"`.

`na` How to handle `NA` documents. `"keep"` returns `NA_integer_` (default); `"zero"` treats missing documents as zero matches; `"error"` fails.

## Value

An integer vector with the same length as `doc`.

## See Also

ac_count_file(), ac_detect(), ac_locate(), ac_extract().

## Examples

```
if (requireNamespace("dplyr", quietly = TRUE)) {
  ac <- ac_build(c("hello", "world"))
  docs <- data.frame(doc = c("hello world", "nothing", "world"))
  dplyr::mutate(docs, n_matches = ac_count(ac, doc))
}
```

---

# ac_count_file *Count pattern matches in files*

---

## Description

ac_count_file() returns the number of pattern matches in each file.

## Usage

```
ac_count_file(ac, path, stream = FALSE, overlapping = FALSE)
```

## Arguments

| | |
|---|---|
| ac | An <ac_automaton> object created by ac_build(). |
| path | A vector of file paths to search. |
| stream | If FALSE (default), each file is read into memory before searching. If TRUE, files are searched as streams. Stream search requires an automaton built with match_kind = "standard". |
| overlapping | Default is FALSE. If TRUE, count overlapping matches. This is only supported when stream = FALSE and ac was built with match_kind = "standard". |

## Value

An integer vector with the same length as path.

## See Also

ac_count(), ac_detect_file(), ac_locate_bytes().

## Examples

```
ac <- ac_build(c("hello", "world"))
path <- tempfile()
writeLines("hello hello world", path)
ac_count_file(ac, path)
```

## ac_detect *Detect pattern matches in documents*

### Description

`ac_detect()` returns whether each document has at least one pattern match.

### Usage

```
ac_detect(ac, doc, na = c("keep", "false", "error"))
```

### Arguments

| | |
|---|---|
| ac | An <ac_automaton> object created by `ac_build()`. |
| doc | A character vector of documents to search. |
| na | How to handle NA documents. `"keep"` returns NA (default); `"false"` treats missing documents as not matched; `"error"` fails. |

### Value

A logical vector with the same length as doc.

### See Also

`ac_detect_file()`, `ac_count()`, `ac_locate()`, `ac_extract()`.

### Examples

```
if (requireNamespace("dplyr", quietly = TRUE)) {
  ac <- ac_build(c("hello", "world"))
  docs <- data.frame(doc = c("hello world", "nothing", "world"))
  dplyr::mutate(docs, matched = ac_detect(ac, doc))
}
```

## ac_detect_file *Detect pattern matches in files*

### Description

`ac_detect_file()` returns whether each file has at least one pattern match.

### Usage

```
ac_detect_file(ac, path, stream = FALSE)
```

### Arguments

| | |
|---|---|
| ac | An <ac_automaton> object created by ac_build(). |
| path | A vector of file paths to search. |
| stream | If FALSE (default), each file is read into memory before searching. If TRUE, files are searched as streams. Stream search requires an automaton built with match_kind = "standard". |

### Value

A logical vector with the same length as path.

### See Also

ac_detect(), ac_count_file(), ac_locate_bytes().

### Examples

```
ac <- ac_build(c("hello", "world"))
path <- tempfile()
writeLines("hello world", path)
ac_detect_file(ac, path)
```

---

## ac_extract *Extract pattern matches from documents*

---

### Description

ac_extract() returns one list element per document. Each element contains the matched text and the corresponding pattern values.

### Usage

```
ac_extract(ac, doc, overlapping = FALSE, na = c("keep", "empty", "error"))
```

### Arguments

| | |
|---|---|
| ac | An <ac_automaton> object created by ac_build(). |
| doc | A character vector of documents to search. |
| overlapping | Default is FALSE. If TRUE, extract overlapping matches. This is only supported when ac was built with match_kind = "standard". |
| na | How to handle NA documents. "keep" returns one row with missing matches and patterns values (default); "empty" treats missing documents as no matches; "error" fails. |

## Value

A list with the same length as doc. Each element is a data frame with one row per match and two columns:

- matches: Text matched in the document.
- patterns: Pattern values corresponding to each match.

## See Also

ac_extract_df(), ac_locate(), ac_detect(), ac_count().

## Examples

```
if (
  requireNamespace("dplyr", quietly = TRUE) &&
    requireNamespace("tibble", quietly = TRUE) &&
    requireNamespace("tidyr", quietly = TRUE)
) {
  ac <- ac_build(c("hello", "world"))
  tibble::tibble(doc = c("hello world", "nothing", "world")) |>
    dplyr::mutate(extracted = ac_extract(ac, doc)) |>
    tidyr::unnest(extracted)
}
```

---

# ac_extract_df *Extract pattern matches as a data frame*

---

## Description

ac_extract_df() is the data-frame form of ac_extract(). It is useful when you want one row per match instead of one list element per document.

## Usage

```
ac_extract_df(ac, doc, overlapping = FALSE, na = c("omit", "keep", "error"))
```

## Arguments

| | |
|---|---|
| ac | An <ac_automaton> object created by ac_build(). |
| doc | A character vector of documents to search. |
| overlapping | Default is FALSE. If TRUE, extract overlapping matches. This is only supported when ac was built with match_kind = "standard". |
| na | How to handle NA documents. "omit" drops missing documents (default); "keep" returns one row with missing result columns for each missing document; "error" fails. |

## Value

A data frame with one row per match and three columns: `doc_id`, `matches`, and `patterns`.

## See Also

`ac_extract()`, `ac_locate_df()`.

## Examples

```
ac <- ac_build(c("hello", "world"))
doc <- c("hello world", "nothing", "world hello")
ac_extract_df(ac, doc)
```

---

# ac_extract_file *Extract pattern matches from files*

---

## Description

`ac_extract_file()` returns one list element per file. Each element contains the matched text and the corresponding pattern values.

## Usage

```
ac_extract_file(ac, path, stream = FALSE, overlapping = FALSE)
```

## Arguments

| | |
|---|---|
| `ac` | An <ac_automaton> object created by `ac_build()`. |
| `path` | A vector of file paths to search. |
| `stream` | If `FALSE` (default), each file is read into memory before searching. If `TRUE`, files are searched as streams. Stream search requires an automaton built with `match_kind = "standard"`. |
| `overlapping` | Default is `FALSE`. If `TRUE`, extract overlapping matches. This is only supported when `stream = FALSE` and `ac` was built with `match_kind = "standard"`. |

## Value

A list with the same length as `path`. Each element is a data frame with one row per match and two columns:

- `matches`: Text matched in the file.
- `patterns`: Pattern values corresponding to each match.

## See Also

`ac_extract()`, `ac_detect_file()`, `ac_count_file()`.

### Examples

```
ac <- ac_build(c("hello", "world"))
path <- tempfile()
writeLines("hello world", path)
ac_extract_file(ac, path)
```

---

## ac_info *Return automaton metadata*

---

### Description

Return automaton metadata

### Usage

```
ac_info(ac)
```

### Arguments

| | |
|---|---|
| ac | An <ac_automaton> object created by ac_build(). |

### Value

A list of automaton metadata.

### See Also

ac_build(), ac_patterns().

### Examples

```
ac <- ac_build(c("hello", "world"))
ac_info(ac)
```

---

## ac_locate *Locate pattern matches in strings*

---

### Description

ac_locate() searches a character vector with a compiled automaton and returns one list element per document. Character offsets are 1-based and inclusive, so they can be used directly with substr().

### Usage

```
ac_locate(ac, doc, overlapping = FALSE, na = c("keep", "empty", "error"))
```

### Arguments

| | |
|---|---|
| ac | An <ac_automaton> object created by ac_build(). |
| doc | A character vector of documents to search. |
| overlapping | Default is FALSE. If TRUE, report overlapping matches. This is only supported when ac was built with match_kind = "standard". |
| na | How to handle NA documents. "keep" returns one row with missing pattern_id, start, and end values (default); "empty" treats missing documents as no matches; "error" fails. |

### Value

A list with the same length as doc. Each element is a data frame with one row per match and three columns:

- pattern_id: Index of the matched pattern in ac_patterns(ac).
- start: 1-based index of the first character in each match.
- end: 1-based index of the last character in each match.

### See Also

ac_locate_df(), ac_locate_bytes(), ac_extract(), ac_detect(), ac_count().

### Examples

```
if (
  requireNamespace("dplyr", quietly = TRUE) &&
    requireNamespace("tibble", quietly = TRUE) &&
    requireNamespace("tidyr", quietly = TRUE)
) {
  ac <- ac_build(c("hello", "world"))
  tibble::tibble(doc = c("hello world", "nothing", "world")) |>
    dplyr::mutate(hits = ac_locate(ac, doc)) |>
    tidyr::unnest(hits)
}
```

---

## ac_locate_bytes *Locate pattern matches with byte offsets*

---

### Description

ac_locate_bytes() searches a character vector with a compiled automaton and returns byte offsets from the Rust aho-corasick crate. Byte offsets are 0-based, and byte_end is end-exclusive.

### Usage

```
ac_locate_bytes(ac, doc, overlapping = FALSE, na = c("omit", "keep", "error"))
```

### Arguments

| | |
|---|---|
| ac | An <ac_automaton> object created by ac_build(). |
| doc | A character vector of documents to search. |
| overlapping | Default is FALSE. If TRUE, report overlapping matches. This is only supported when ac was built with match_kind = "standard". |
| na | How to handle NA documents. "omit" drops missing documents (default); "keep" returns one row with missing result columns for each missing document; "error" fails. |

### Value

A data frame with one row per match and four columns: doc_id, pattern_id, byte_start, and byte_end.

### See Also

ac_locate(), ac_locate_df().

### Examples

```
ac <- ac_build(c("hello", "world"))
doc <- c("hello world", "nothing", "world hello")
ac_locate_bytes(ac, doc)
```

---

## ac_locate_df *Locate pattern matches as a data frame*

---

### Description

ac_locate_df() is the data-frame form of ac_locate(). It is useful when you want one row per match instead of one list element per document.

### Usage

```
ac_locate_df(ac, doc, overlapping = FALSE, na = c("omit", "keep", "error"))
```

### Arguments

| | |
|---|---|
| ac | An <ac_automaton> object created by ac_build(). |
| doc | A character vector of documents to search. |
| overlapping | Default is FALSE. If TRUE, report overlapping matches. This is only supported when ac was built with match_kind = "standard". |
| na | How to handle NA documents. "omit" drops missing documents (default); "keep" returns one row with missing result columns for each missing document; "error" fails. |

### Value

A data frame with one row per match and four columns: `doc_id`, `pattern_id`, `start`, and `end`.

### See Also

`ac_locate()`, `ac_locate_bytes()`, `ac_extract_df()`.

### Examples

```
ac <- ac_build(c("hello", "world"))
doc <- c("hello world", "nothing", "world hello")
ac_locate_df(ac, doc)
```

---

## ac_locate_file *Locate pattern matches in files*

---

### Description

`ac_locate_file()` searches files with a compiled automaton and returns one list element per file. Character offsets are 1-based and inclusive, so they can be used directly with `substr()`.

### Usage

```
ac_locate_file(ac, path, overlapping = FALSE)
```

### Arguments

| | |
|---|---|
| `ac` | An `<ac_automaton>` object created by `ac_build()`. |
| `path` | A vector of file paths to search. |
| `overlapping` | Default is `FALSE`. If `TRUE`, report overlapping matches. This is only supported when `ac` was built with `match_kind = "standard"`. |

### Details

File location search is always non-streaming. Converting byte offsets from a streaming search into R-facing character offsets would require a second pass over the same file to reconstruct UTF-8 character boundaries. Keeping `ac_locate_file()` as a simple in-memory search is the clearest implementation.

### Value

A list with the same length as `path`. Each element is a data frame with one row per match and three columns:

- `pattern_id`: Index of the matched pattern in `ac_patterns(ac)`.
- `start`: 1-based index of the first character in each match.
- `end`: 1-based index of the last character in each match.

## See Also

ac_locate(), ac_detect_file(), ac_count_file(), ac_extract_file().

## Examples

```
ac <- ac_build(c("hello", "world"))
path <- tempfile()
writeLines("hello world", path)
ac_locate_file(ac, path)
```

---

`ac_patterns` *Return patterns stored in an automaton*

---

## Description

Return patterns stored in an automaton

## Usage

```
ac_patterns(ac)
```

## Arguments

| | |
|---|---|
| ac | An <ac_automaton> object created by ac_build(). |

## Value

A character vector of stored patterns.

## See Also

ac_build(), ac_info().

## Examples

```
ac <- ac_build(c("hello", "world"))
ac_patterns(ac)
```

ac_replace *Replace pattern matches in documents*

## Description

ac_replace() replaces all non-overlapping matches in each document with the corresponding replacement string.

## Usage

```
ac_replace(ac, doc, replace_with, na = c("keep", "empty", "error"))
```

## Arguments

| | |
|---|---|
| ac | An <ac_automaton> object created by ac_build(). |
| doc | A character vector of documents to search and replace. |
| replace_with | A character vector of replacements. If length 1, the same replacement is used for every pattern. Otherwise, it **MUST** have the same length as ac_patterns(ac), and replacements are matched to patterns by position. |
| na | How to handle NA documents. "keep" returns NA_character_ (default); "empty" treats missing documents as empty strings; "error" fails. |

## Value

A character vector with the same length and names as doc.

## See Also

ac_build(), ac_detect(), ac_count(), ac_extract(), ac_locate().

## Examples

```
ac <- ac_build(c("fox", "brown", "quick"))
ac_replace(
  ac,
  "The quick brown fox.",
  c("sloth", "grey", "slow")
)

ac <- ac_build(c("append", "appendage", "app"), match_kind = "leftmost_first")
ac_replace(ac, "append the app to the appendage", c("x", "y", "z"))
```

ac_replace_file *Replace pattern matches in files*

## Description

ac_replace_file() replaces all non-overlapping matches in input files and writes the result to output files.

## Usage

```
ac_replace_file(ac, path, replace_with, output = NULL, stream = FALSE)
```

## Arguments

| | |
|---|---|
| ac | An <ac_automaton> object created by ac_build(). |
| path | A vector of input file paths to search and replace. |
| replace_with | A character vector of replacements. If length 1, the same replacement is used for every pattern. Otherwise, it **MUST** have the same length as ac_patterns(ac), and replacements are matched to patterns by position. |
| output | A vector of output file paths. It must have the same length as path. If NULL, output paths are created by adding "_replaced" suffix. Existing output files are overwritten. |
| stream | If FALSE (default), each file is read into memory before replacement. If TRUE, files are searched and replaced as streams. Stream replacement requires an automaton built with match_kind = "standard". |

## Value

A character vector of output file paths with the same length as path.

## See Also

ac_replace(), ac_detect_file(), ac_count_file().

## Examples

```
ac <- ac_build(c("fox", "brown", "quick"))
path <- tempfile(fileext = ".txt")
writeLines("The quick brown fox.", path)
ac_replace_file(path = path, ac = ac, replace_with = c("sloth", "grey", "slow"))
```

# Index