Package 'awkreader'

September 3, 2026

Title File Reading with Pre-Filtering, Pattern Searching, and Distributed Files

Version 0.1.0

Description Provides high-performance tools for out-of-core text processing and data ingestion by leveraging system 'AWK' utilities. Allows users to count records, filter rows, and compute streaming aggregations—such as group-by means, streaming medians, standard deviations, and correlations—directly on disk prior to reading data into R. By delegating line-by-line filtering and summarization to system-level 'AWK' commands and streaming results back through data.table::fread(), the package significantly reduces memory footprint and execution times when working with large individual files or multi-file directory structures.

License MIT + file LICENSE

Encoding UTF-8

RoxygenNote 7.3.3

Suggests knitr, rmarkdown, testthat (>= 3.0.0)

VignetteBuilder knitr

Config/testthat/edition 3

Imports data.table

NeedsCompilation no

Author David Shilane [aut],
Akshat Maurya [aut, cre],
Jason Livingston [aut],
Chung-Woo (Caffrey) Lee [aut],
Mayur Bansal [aut],
Srivastav Budugutta [aut]

Maintainer Akshat Maurya <codingmaster902@gmail.com>

Repository CRAN

Date/Publication 2026-09-03 12:20:11 UTC

Contents

aggregated.fread 2

combined.fread 4
filtered.fread 6
pattern.fread 9
record.count 11

Index **14**

`aggregated.fread` *Fast Stream Aggregation of Delimited Files via AWK*

Description

High-performance aggregation and streaming of multiple tabular data files using an optimized AWK engine pipeline. This function parses columns, groups multi-file inputs, and calculates summarizations (like `sum()`, `mean()`, and `sd()`) natively in the shell before pulling structured aggregates back into R. It safely evaluates scalar mathematical transformations passed as strings (e.g., `"sqrt(col)"`).

Usage

```
aggregated.fread(
  the.files,
  value.code = "data",
  delim = ",",
  file.pattern = NULL,
  recursive = FALSE,
  num.batches = 1,
  num.files.per.batch = 1000,
  summarize.with = NULL,
  return.as = "result",
  group.by = NULL,
  path.to.awk = "awk",
  file.header = "file",
  include.filename = FALSE,
  skip = 0,
  show.warnings = TRUE,
  nrows = Inf,
  sample.size.median = -1,
  return.data.table = TRUE
)
```

Arguments

`the.files`	A character vector containing paths to the targeted delimited text datasets.
`value.code`	A internal character tracking string for code generation mappings. Defaults to `"data"`.
`delim`	A character string defining the field separator delimiter within files. Defaults to `","`.

`file.pattern` Optional character string to filter file names when `the.files` contains directory paths. Accepts simple extensions (e.g., `"csv"` or `".csv"`), wildcards (e.g., `"*.csv"`), or regular expressions (e.g., `"\.csv$"`). Default is `NULL` (includes all files).

`recursive` Logical. Should directory searches recurse into subdirectories when `the.files` contains directory paths? Default is `FALSE`.

`num.batches` An integer defining the processing grouping structure. Defaults to `1`.

`num.files.per.batch` An integer specifying the maximum threshold of files targeted per concurrent process execution loop. Defaults to `1000`.

`summarize.with` A named list specifying operations and target columns. Names must match supported functions: `"sum"`, `"mean"`, or `"sd"`. Elements can include raw columns or functional call strings like `"sqrt(price)"`. Defaults to `NULL`.

`return.as` A character string defining what data is returned to the environment. Options include `"result"` (aggregated dataset), `"code"` (generated AWK commands), or `"all"` (a combined list of both). Defaults to `"result"`.

`group.by` A character vector or list containing column names to act as the aggregation grouping dimensions. Defaults to `NULL`.

`path.to.awk` A character string declaring the binary location execution path. Defaults to `"awk"`.

`file.header` A character string tracking column identification titles when filenames are tracked. Defaults to `"file"`.

`include.filename` A logical value indicating whether the tracking origin file string column should append to final structures. Defaults to `FALSE`.

`skip` An integer, list, or character regex pattern. Controls metadata/line skipping configuration routines before row parsing evaluations trigger. Defaults to `0`.

`show.warnings` A logical value determining whether underlying structural data warnings should surface during operation execution cycles. Defaults to `TRUE`.

`nrows` A numeric ceiling threshold limiting output aggregation allocations. Defaults to `Inf`.

`sample.size.median` Integer. Maximum number of observations per group used to compute streaming medians via the P-Square algorithm. A positive integer caps the sample size per group for faster execution on large datasets. Set to `-1` (default) or `0` to compute across all rows without sampling.

`return.data.table` A logical value. If `TRUE`, returns a `data.table` object; otherwise, reverts the format to a standard base `data.frame`. Defaults to `TRUE`.

Value

Depending on the configuration argument passed to `return.as`, returns either a `data.table` (`data.frame` if `return.data.table = FALSE`), a character vector representing full system executable command configurations, or a combined metadata list package object.

Examples

```
# Create a sample CSV file using base R mtcars dataset
tmp_file <- tempfile(fileext = ".csv")
write.csv(mtcars, tmp_file, row.names = FALSE)

# Aggregate data over multiple columns grouped by cylinders
agg_res <- aggregated.fread(
  the.files = tmp_file,
  group.by = "cyl",
  summarize.with = list(
    mean = list("hp", "mpg"),
    sd = "wt"
  ),
  return.as = "all"
)
print(agg_res$result)

# Clean up temporary file
unlink(tmp_file)
```

`combined.fread` *Fast Batch Combining of Flat Files via AWK*

Description

A streamlined wrapper around `filtered.fread()` designed to quickly read, process, and combine multiple flat files into a single dataset without applying row filters.

Usage

```
combined.fread(
  the.files,
  path.to.awk = NULL,
  header = TRUE,
  the.variables = ".",
  file.pattern = NULL,
  recursive = FALSE,
  include.filename = TRUE,
  skip = 0,
  file.header = "file",
  num.files.per.batch = 1000,
  return.as = "result",
  envir = parent.frame(),
  show.warnings = FALSE,
  return.data.table = TRUE,
  nrows = Inf,
  drop = NULL,
```

```
  ...
)
```

Arguments

the.files	A character vector of file paths to process. Non-existent files are automatically filtered out.
path.to.awk	A character string specifying the path to the AWK binary. If NULL (default), the function attempts to invoke a global system call to "awk".
header	A logical value indicating whether the target files contain a header row. Default is TRUE.
the.variables	A character vector specifying which columns to retain. Use "." (default) to retain all columns.
file.pattern	Optional character string to filter file names when the.files contains directory paths. Accepts simple extensions (e.g., "csv" or ".csv"), wildcards (e.g., "*.csv"), or regular expressions (e.g., "\.csv$"). Default is NULL (includes all files).
recursive	Logical. Should directory searches recurse into subdirectories when the.files contains directory paths? Default is FALSE.
include.filename	A logical value indicating whether to include a source file tracking column in the returned dataset. Default is TRUE.
skip	A numeric offset, a character regex pattern, or a structured list indicating lines to bypass. If a list is used, it **must follow dot notation**: • skip.metadata.rows: An integer count or a character regex pattern used to identify where the metadata block ends. • skip.data.rows: An integer specifying the number of data rows to explicitly skip after the header. Default is 0.
file.header	A character string defining the column name for the tracked file origin. Only utilized if include.filename = TRUE. Default is "file".
num.files.per.batch	An integer specifying how many files to aggregate per AWK system pipeline call. Default is 1000.
return.as	A character string specifying the desired return object. Options are "result" (default), "code" (returns raw generated AWK scripts), or "all" (returns both).
envir	The environment context in which evaluation variables are evaluated. Default is parent.frame().
show.warnings	A logical value determining whether underlying data.table::fread() messages should be displayed or suppressed. Default is FALSE.
return.data.table	A logical value indicating whether to return a data.table object or a standard data.frame. Default is TRUE.
nrows	An integer specifying the maximum total rows to parse out from the batch pipeline. Default is Inf.

drop	A character or numeric index vector specifying columns that should be explicitly excluded from the final output.
...	Extra parameters forwarded to underlying internal setup routines inside filtered.fread().

Value

A data.table (or data.frame), or a character vector containing the raw shell commands, depending on the value passed to return.as.

Examples

```
# Create sample CSV files with metadata rows
f1 <- tempfile(fileext = ".csv")
f2 <- tempfile(fileext = ".csv")

writeLines(c("# Title: Jan Log", "# Status: Active", "id,val", "1,10", "2,20"), f1)
writeLines(c("# Title: Feb Log", "# Status: Active", "id,val", "3,30", "4,40"), f2)

# Combine multiple log files while skipping the 2 metadata rows
all_logs <- combined.fread(
  the.files = c(f1, f2),
  skip = list(skip.metadata.rows = 2, skip.data.rows = 0)
)
print(all_logs)

# Clean up temporary files
unlink(c(f1, f2))
```

---

filtered.fread *Fast, Filtered Reading of Multiple Files via AWK*

Description

Translates R-style filtering statements into highly efficient AWK commands to process, filter, and select specific columns from multiple flat files simultaneously. Batched outputs are fast-loaded and bound together into a single dataset.

Usage

```
filtered.fread(
  the.files,
  path.to.awk = NULL,
  file.pattern = NULL,
  recursive = FALSE,
  header = TRUE,
  delim = ",",
  the.filter = NULL,
```

```
  the.variables = ".",
  include.filename = TRUE,
  skip = 0,
  file.header = "file",
  num.files.per.batch = 1000,
  return.as = "result",
  envir = parent.frame(),
  and.symbol = "&",
  or.symbol = "|",
  in.symbol = "%in%",
  nin.symbol = "%nin%",
  show.warnings = FALSE,
  return.data.table = TRUE,
  nrows = Inf,
  drop = NULL,
  ...
)
```

Arguments

`the.files`	A character vector of file paths to process. Non-existent files are automatically filtered out.
`path.to.awk`	A character string specifying the path to the AWK binary. If `NULL` (default), the function attempts to invoke a global system call to `"awk"`.
`file.pattern`	Optional character string to filter file names when `the.files` contains directory paths. Accepts simple extensions (e.g., `"csv"` or `".csv"`), wildcards (e.g., `"*.csv"`), or regular expressions (e.g., `"\.csv$"`). Default is `NULL` (includes all files).
`recursive`	Logical. Should directory searches recurse into subdirectories when `the.files` contains directory paths? Default is `FALSE`.
`header`	A logical value indicating whether the target files contain a header row. Default is `TRUE`. If `FALSE`, columns are auto-assigned as `V1`, `V2`, etc.
`delim`	A character string specifying the column separator within the files. Default is `","`.
`the.filter`	A character string or unquoted expression outlining the filtering logic to pass to AWK. Supports implicit translation of basic math and symbolic calls. Default is `NULL` (no filtering).
`the.variables`	A character vector specifying which columns to retain. Use `"."` (default) to retain all columns.
`include.filename`	A logical value indicating whether to include a source file tracking column in the returned dataset. Default is `TRUE`.
`skip`	A numeric offset, a character regex pattern, or a structured list indicating lines to bypass. If a list is used, it **must follow dot notation**: • `skip.metadata.rows`: An integer count or a character regex pattern used to identify where the metadata block ends before hitting the core table.

- `skip.data.rows`: An integer specifying the number of data rows to explicitly skip after the header.

Default is `0`.

`file.header` A character string defining the column name for the tracked file origin. Only utilized if `include.filename = TRUE`. Default is `"file"`.

`num.files.per.batch`
An integer specifying how many files to aggregate per AWK system pipeline call. Default is `1000`.

`return.as` A character string specifying the desired return object. Options are:

- `"result"` (default): Returns the compiled dataset.
- `"code"`: Bypasses compilation and returns a character vector of the raw generated AWK scripts.
- `"all"`: Returns a structured list containing both the compiled dataset and the underlying AWK statements.

`envir` The environment context in which evaluation characters or external metadata strings are parsed. Default is `parent.frame()`.

`and.symbol` A character replacement flag for logical AND statements. Default is `"&"`.

`or.symbol` A character replacement flag for logical OR statements. Default is `"|"`.

`in.symbol` A character replacement flag for inclusion tests. Default is `"%in%"`.

`nin.symbol` A character replacement flag for exclusion tests. Default is `"%nin%"`.

`show.warnings` A logical value determining whether underlying `data.table::fread()` shell messages should be displayed or suppressed. Default is `FALSE`.

`return.data.table`
A logical value indicating whether to return a `data.table` object or a standard `data.frame`. Default is `TRUE`.

`nrows` An integer specifying the maximum total rows to parse out from the batch pipeline. Default is `Inf`.

`drop` A character or numeric index vector specifying columns that should be explicitly excluded from the final output.

`...` Extra parameters forwarded to underlying internal setup routines.

Value

A `data.table` (or `data.frame`), or a character vector containing the raw shell commands, depending on the value passed to `return.as`.

Examples

```
# Create a sample CSV file using base R mtcars dataset
tmp_file <- tempfile(fileext = ".csv")
write.csv(mtcars, tmp_file, row.names = FALSE)

# Standard usage with a numeric filter and dot-notation row skipping
my_data <- filtered.fread(
  the.files = tmp_file,
```

```
  the.filter = "hp > 100",
  skip = list(skip.metadata.rows = 0, skip.data.rows = 2)
)
print(my_data)

# Clean up temporary file
unlink(tmp_file)
```

`pattern.fread` *Pattern-Based Subsetting and Reading of Multiple Files via AWK*

Description

Reads and aggregates multiple flat files concurrently, filtering rows based on regular expression patterns processed directly via AWK before parsing the data into R.

Usage

```
pattern.fread(
  the.files,
  path.to.awk = NULL,
  header = TRUE,
  the.patterns = NULL,
  file.pattern = NULL,
  recursive = FALSE,
  tf = TRUE,
  delim = ",",
  connectors = "or",
  the.variables = ".",
  include.filename = TRUE,
  skip = 0,
  file.header = "file",
  num.files.per.batch = 1000,
  return.as = "result",
  envir = parent.frame(),
  show.warnings = FALSE,
  return.data.table = TRUE,
  nrows = Inf,
  drop = NULL,
  ...
)
```

Arguments

`the.files`	A character vector of file paths to process. Non-existent files are automatically filtered out.

path.to.awk
: A character string specifying the path to the AWK binary. If NULL (default), the function attempts to invoke a global system call to "awk".

header
: A logical value indicating whether the target files contain a header row. Default is TRUE.

the.patterns
: A character vector containing the regular expression patterns to match against data rows. Default is NULL.

file.pattern
: Optional character string to filter file names when the.files contains directory paths. Accepts simple extensions (e.g., "csv" or ".csv"), wildcards (e.g., "*.csv"), or regular expressions (e.g., "\.csv$"). Default is NULL (includes all files).

recursive
: Logical. Should directory searches recurse into subdirectories when the.files contains directory paths? Default is FALSE.

tf
: A logical value determining whether to include rows that match the patterns (TRUE) or exclude rows that match them (FALSE). Default is TRUE.

delim
: A character string specifying the column separator within the files. Default is ",".

connectors
: A character string defining how multiple patterns should be combined logically. Options are "or" (default) or "and".

the.variables
: A character vector specifying which columns to retain. Use "." (default) to retain all columns.

include.filename
: A logical value indicating whether to include a source file tracking column in the returned dataset. Default is TRUE.

skip
: A numeric offset, a character regex pattern, or a structured list indicating lines to bypass. If a list is used, it **must follow dot notation**:

  - skip.metadata.rows: An integer count or a character regex pattern used to identify where the metadata block ends.
  - skip.data.rows: An integer specifying the number of data rows to explicitly skip after the header.

  Default is 0.

file.header
: A character string defining the column name for the tracked file origin. Only utilized if include.filename = TRUE. Default is "file".

num.files.per.batch
: An integer specifying how many files to aggregate per AWK system pipeline call. Default is 1000.

return.as
: A character string specifying the desired return object. Options are "result" (default), "code", or "all".

envir
: The environment context in which evaluation characters are parsed. Default is parent.frame().

show.warnings
: A logical value determining whether underlying terminal messages should be shown. Default is FALSE.

return.data.table
: A logical value indicating whether to return a data.table or standard data.frame. Default is TRUE.

nrows	An integer specifying the maximum total rows to parse out. Default is Inf.
drop	A character or numeric index vector specifying columns to explicitly exclude.
...	Extra parameters forwarded to internal setup routines.

Value

A data.table (or data.frame) containing rows satisfying the pattern configurations.

Examples

```
# Create sample log files
log1 <- tempfile(fileext = ".log")
log2 <- tempfile(fileext = ".log")
writeLines(c("INFO: System started", "Error: Disk full"), log1)
writeLines(c("INFO: User login", "Critical: Database unreachable"), log2)

# Extract rows matching "Error" or "Critical" across log files
errors <- pattern.fread(
  the.files = c(log1, log2),
  the.patterns = c("Error", "Critical"),
  connectors = "or"
)
print(errors)

# Clean up temporary files
unlink(c(log1, log2))
```

---

record.count *Efficient Record and Row Counting via AWK*

Description

Computes the total number of records matching specific logical filtering criteria across multiple large files using AWK. This performs counting at the shell level, eliminating the overhead of loading whole datasets into memory.

Usage

```
record.count(
  the.files,
  path.to.awk = NULL,
  delim = ",",
  the.filter = NULL,
  file.pattern = NULL,
  recursive = FALSE,
  the.variables = ".",
  include.filename = TRUE,
```

```
  skip = 0,
  file.header = "file",
  num.files.per.batch = 1000,
  return.as = "result",
  envir = parent.frame(),
  and.symbol = "&",
  or.symbol = "|",
  in.symbol = "%in%",
  nin.symbol = "%nin%",
  show.warnings = FALSE,
  nrows = Inf,
  drop = NULL,
  ...
)
```

Arguments

`the.files`	A character vector of file paths to scan. Non-existent files are automatically filtered out.
`path.to.awk`	A character string specifying the path to the AWK binary. If `NULL` (default), the function attempts to invoke a global system call to `"awk"`.
`delim`	A character string specifying the column separator within the files. Default is `","`.
`the.filter`	A character string or unquoted expression outlining the filtering logic to pass to AWK. Default is `NULL` (counts all records matching layout rules).
`file.pattern`	Optional character string to filter file names when `the.files` contains directory paths. Accepts simple extensions (e.g., `"csv"` or `".csv"`), wildcards (e.g., `"*.csv"`), or regular expressions (e.g., `"\.csv$"`). Default is `NULL` (includes all files).
`recursive`	Logical. Should directory searches recurse into subdirectories when `the.files` contains directory paths? Default is `FALSE`.
`the.variables`	A character vector specifying active evaluation columns. Default is `"."`.
`include.filename`	A logical value indicating whether to retain tracking metrics grouped by individual files. Default is `TRUE`.
`skip`	A numeric offset, a character regex pattern, or a structured list indicating lines to bypass. If a list is used, it **must follow dot notation**: • `skip.metadata.rows`: An integer count or a character regex pattern used to identify where the metadata block ends. • `skip.data.rows`: An integer specifying the number of data rows to explicitly skip after the header. Default is `0`.
`file.header`	A character string defining the tracking header index. Default is `"file"`.
`num.files.per.batch`	An integer specifying how many files to aggregate per pipeline call. Default is `1000`.

return.as	A character string specifying the desired return format. Options are "result" (default), "code", or "all".
envir	The environment context in which variables are parsed. Default is parent.frame().
and.symbol	A character replacement flag for logical AND statements. Default is "&".
or.symbol	A character replacement flag for logical OR statements. Default is "|".
in.symbol	A character replacement flag for inclusion tests. Default is "%in%".
nin.symbol	A character replacement flag for exclusion tests. Default is "%nin%".
show.warnings	A logical value determining whether terminal messages are displayed. Default is FALSE.
nrows	An integer specifying the maximum total matching record sets to count. Default is Inf.
drop	A character or numeric index vector specifying columns to exclude from mapping.
...	Extra parameters forwarded to underlying internal setup routines.

Value

A data.table summarizing record counts per file (or overall), or raw shell string arrays.

Examples

```
# Create a sample CSV file with price data
tmp_file <- tempfile(fileext = ".csv")
write.csv(data.frame(id = 1:3, price = c(500, 15000, 20000)), tmp_file, row.names = FALSE)

# Get matching transaction counts without importing full rows
total_expensive_items <- record.count(
  the.files = tmp_file,
  the.filter = "price > 10000"
)
print(total_expensive_items)

# Clean up temporary file
unlink(tmp_file)
```

Index

aggregated.fread, 2

combined.fread, 4

filtered.fread, *4*, 6

pattern.fread, 9

record.count, 11