Package 'anotherSAS7bdat'

November 9, 2026

Title Read 'SAS' Data in Chunks and Convert to 'Parquet'

Version 0.0.9

Description Reads 'SAS' 'sas7bdat' files through a persistent native reader. Processes configurable chunks without reopening the input or restarting the parser between chunks. Supports uncompressed, run-length encoded, and Ross Data Compression data through the bundled 'ReadStat' library. Also streams native decoded batches directly to 'Parquet' through the 'Arrow' C Data Interface and the optional 'arrow' package, without constructing R data frames.

License MIT + file LICENSE

URL https://github.com/Fred-Wu/anotherSAS7bdat

BugReports https://github.com/Fred-Wu/anotherSAS7bdat/issues

Encoding UTF-8

Depends R (>= 4.1.0)

Imports Rcpp (>= 1.0.14), hms (>= 0.5.0), utils

LinkingTo Rcpp (>= 1.0.14)

Suggests testthat (>= 3.3.0), haven (>= 2.5.5), arrow (>= 18.1.0)

SystemRequirements C++17, POSIX threads, iconv

Config/testthat/edition 3

Config/roxygen2/version 8.1.0

NeedsCompilation yes

Author Jianyun Wu [aut, cre],
Evan Miller [aut, cph] (Author of included ReadStat code)

Maintainer Jianyun Wu <jianyun.fred.wu@hotmail.com>

Repository CRAN

Date/Publication 2026-11-09 11:10:02 UTC

Contents

sas7bdat_read 2
sas7bdat_to_parquet 6

Index **9**

`sas7bdat_read` *Read SAS files in continuous chunks*

Description

Read a SAS file a group of rows at a time. Use `sas7bdat_read()` to run a function on every chunk automatically, or use `sas7bdat_open()` and `sas7bdat_read_chunk()` to request data frames yourself. Inspect the reader with `sas7bdat_info()` and close it with `sas7bdat_close()` when finished.

Usage

```
sas7bdat_read(
  path,
  callback,
  chunk_rows = 100000L,
  chunk_bytes = NULL,
  columns = NULL,
  encoding = NULL,
  dates = TRUE
)

sas7bdat_open(
  path,
  chunk_rows = 100000L,
  chunk_bytes = NULL,
  columns = NULL,
  encoding = NULL,
  dates = TRUE
)

sas7bdat_read_chunk(reader, n = NULL)

sas7bdat_close(reader)

sas7bdat_info(reader)
```

Arguments

`path`	A character string specifying the path to a local `.sas7bdat` file.
`callback`	A function whose first argument receives each chunk as a data frame.
`chunk_rows`	A positive whole number specifying the maximum rows per chunk. Fewer rows are returned when the `chunk_bytes` target is reached first or when fewer rows remain in the file.

chunk_bytes	Optional approximate decoded size limit for each chunk, as a numeric byte count or a size string with units B, KB, MB, GB, KiB, MiB, or GiB (case-insensitive). The default, NULL, sizes native buffers automatically from decoded rows and uses chunk_rows to end each chunk.
columns	A character vector of column names, or NULL for all columns.
encoding	A character string specifying the source encoding, or NULL to use the encoding recorded in the file.
dates	A logical value indicating whether to convert recognized SAS date/time formats to R date/time classes. Dates become Date, datetimes become POSIXct in UTC, and times become hms values displayed as HH:MM:SS (with fractional seconds when present). Set to FALSE to retain the raw SAS numeric values.
reader	A sas7bdat_reader object created by sas7bdat_open().
n	A positive whole number specifying the maximum rows for this read, or NULL to use the reader's chunk_rows. The reader's chunk_bytes target still applies.

Details

With sas7bdat_read(), each chunk is passed to callback as its first positional argument. That argument contains the actual data frame for the current chunk. You choose its name; it does not have to be named chunk. The package calls your function again with the next data frame until reading finishes.

sas7bdat_read() returns a reading summary, not the dataset or your callback's results. Assignments inside the callback are local to that call. To use rows or calculated results after reading, your callback must save them in an object outside the callback or write them to a file. Callback return values are not collected; returning exactly FALSE stops reading.

With sas7bdat_open(), the returned reader is a bookmark attached to the open file, not a data frame. Each sas7bdat_read_chunk(reader) returns the next data frame and advances the position. Assigning it to a variable replaces that variable's previous value; it does not append rows. In a loop, stop when the result is NULL. Previously returned data frames stay valid after later reads or closing. sas7bdat_info() inspects the reader without advancing it.

Information returned by sas7bdat_info() is a snapshot at the time of the call. A saved list does not update when rows are read or the reader is closed. Call sas7bdat_info() again to get the current status and counters.

Reopening the file starts from the beginning. Readers cannot be saved with saveRDS() for later reuse or shared between R processes.

The file is opened once and row reading continues across chunk boundaries without restarting the parser. Metadata setup may seek within the file once at initialization. Internal SAS RLE and RDC compression are detected automatically. The file must be seekable; ZIP/gzip wrappers are unsupported.

After the first chunk request, a native worker decodes ahead while R converts or processes earlier chunks. Read-ahead is bounded to one queued batch and one in-progress batch, each limited by the optional chunk_bytes and the row count requested when that batch starts. Changing n takes effect on output chunks even when data has already been decoded. Closing cancels the worker and discards unread data. Returned data frames own their values and remain valid when native buffers are reused.

Column names are case-sensitive and must be unique. Output columns follow the requested order. Returned text is converted to UTF-8.

`chunk_rows` sets an upper limit on rows per chunk. Each chunk ends after a complete row when either the row limit or the approximate decoded size limit, `chunk_bytes`, is reached, when supplied. If the byte limit is reached first, the chunk contains fewer than `chunk_rows` rows, even before the end of the file. Row counts can vary between chunks as decoded row sizes vary. For `sas7bdat_read_chunk()`, supplying `n` overrides the row limit for that call; the byte target remains in effect. The final chunk may also contain fewer rows if fewer remain in the file.

With `chunk_bytes = NULL`, the requested row count controls chunk sizes. The reader starts with small native buffers and samples the first 256 decoded rows, across chunks if needed. Those rows remain in the normal output stream. The sample estimates text sizes for reserving buffers for the rest of the chunk, with 25 percent text headroom. Estimates are then updated once per batch as text sizes change. Buffers grow if needed and are reused across chunks. Files with fewer than 256 rows use the rows available. Sampling does not reopen the file or make an extra parser pass.

Automatic sizing controls buffer allocation; it does not impose a byte limit. Choose a row count that fits available memory, or supply `chunk_bytes` as an additional limit. An explicit byte limit is never increased by the estimator.

The byte target is approximate and may be exceeded by the last row added to a chunk. It is not a limit on total R memory usage; even a single row may exceed the target. Native buffers for read-ahead and the output chunk can coexist, as can earlier data frames retained by your callback. Reusable native buffers retain their allocated capacity until the reader is closed. MB and GB use powers of 1,000; MiB and GiB use powers of 1,024.

SAS numerics are doubles. Special missing values use haven-compatible tagged NAs; character padding is trimmed by ReadStat. Column labels and SAS formats are retained as `label` and `format.sas` attributes.

Date/time formats are identified once when the reader opens. With `dates = TRUE`, conversion occurs while constructing each output column: dates use R's epoch, datetimes use UTC, and times retain seconds as `hms`. Missing values and their tags are preserved. With `dates = FALSE`, numeric values remain in their original SAS units.

User-defined formats and SAS catalog files are not interpreted. Files exceeding 2,147,483,647 physical observations are rejected in this version, before counter overflow.

To retain the entire dataset, you must explicitly save the chunks and combine them, as shown below. The entire decoded dataset and temporary copies must then fit in memory. For faster binding of many chunks, use `data.table::rbindlist()` from the optional data.table package in place of `do.call(rbind, chunks)`. It returns a `data.table`, which also inherits from `data.frame`.

Value

- `sas7bdat_read()` returns an invisible reading summary: `rows` is the number of rows passed to the callback, `chunks` is the number of calls, and `completed` indicates whether reading reached the end of the file. An empty file calls the callback zero times and returns a completed summary with zero rows.
- `sas7bdat_open()` returns a `sas7bdat_reader` that remembers the reading position in the open file.
- `sas7bdat_read_chunk()` returns a data frame, or `NULL` when no rows remain.

- `sas7bdat_info()` returns a snapshot list with `rows_total`, `rows_delivered`, `columns` (names, types, formats, and labels), `compression`, and `status` (`open`, `eof`, `error`, or `closed`). `eof` means end of file. Additional diagnostic counters are `rows_decoded`, `file_opens`, `file_closes`, `bytes_read`, and `seeks`. Byte and seek counters include metadata setup; they do not measure R memory use. With read-ahead, `rows_decoded` can be greater than `rows_delivered`. `estimation_rows` counts the initial rows used for buffer sizing (up to 256); `estimated_row_bytes` is the approximate decoded bytes per row, updated after each batch. Both are zero before sampling begins.
- `sas7bdat_close()` returns `NULL`, invisibly.

Closing and stopping

`sas7bdat_read()` closes its reader automatically at the end, on an error or interruption, or when the callback returns exactly `FALSE`.

When using `sas7bdat_open()`, call `sas7bdat_close()` when finished. Inside a function, use `on.exit(sas7bdat_close(reader), add = TRUE)` for cleanup even if processing fails. Closing more than once is safe. Reading from a closed reader is an error; open a new reader to start again.

Example files

The package includes three files generated with SAS 9.4:

- `example_uncompressed.sas7bdat`: uncompressed.
- `example_rle.sas7bdat`: RLE compression (`COMPRESS=CHAR`).
- `example_rdc.sas7bdat`: RDC compression (`COMPRESS=BINARY`).

Each contains the same 120 rows with numeric values, Unicode text, dates, times, datetimes, column labels, and ordinary and special missing values. Use any of these filenames in `system.file()` in the examples below. The SAS program used to generate them is included as `examples/generate.sas`. SAS is not needed to read the bundled files.

Examples

```
# Read the entire file into sas_data using a callback.
path <- system.file("examples", "example_uncompressed.sas7bdat",
                    package = "anotherSAS7bdat")
chunks <- list()

sas7bdat_read(
  path,
  chunk_rows = 25,
  callback = function(data) {
    chunks[[length(chunks) + 1L]] <<- data
  }
)

sas_data <- do.call(rbind, chunks)
# Faster alternative with data.table installed:
# sas_data <- data.table::rbindlist(chunks, use.names = TRUE)
rm(chunks)
```

```
# Read the entire RLE-compressed file using a reader.
path <- system.file("examples", "example_rle.sas7bdat",
                    package = "anotherSAS7bdat")
reader <- sas7bdat_open(path, chunk_rows = 25)
chunks <- list()

repeat {
  data <- sas7bdat_read_chunk(reader)
  if (is.null(data)) break
  chunks[[length(chunks) + 1L]] <- data
}

sas7bdat_close(reader)

sas_data <- do.call(rbind, chunks)
# Faster alternative with data.table installed:
# sas_data <- data.table::rbindlist(chunks, use.names = TRUE)
rm(chunks)
sas7bdat_info(reader)$status  # "closed"
```

`sas7bdat_to_parquet` *Convert SAS data directly to Parquet*

Description

Stream a SAS file into one Parquet file without constructing R data frames or R character vectors. Requires the optional arrow package (>= 18.1.0) with Parquet support enabled.

Usage

```
sas7bdat_to_parquet(
  path,
  output,
  chunk_rows = 100000L,
  chunk_bytes = NULL,
  columns = NULL,
  encoding = NULL,
  dates = TRUE,
  compression = "snappy",
  missing_tags = FALSE,
  progress = interactive()
)
```

Arguments

`path`	A character string specifying the path to a local `.sas7bdat` file.
`output`	Path to a new local Parquet file. The parent directory must exist. Existing files are never overwritten.

`chunk_rows` A positive whole number specifying the maximum rows per chunk. Fewer rows are returned when the `chunk_bytes` target is reached first or when fewer rows remain in the file.

`chunk_bytes` Optional approximate decoded size limit for each chunk, as a numeric byte count or a size string with units B, KB, MB, GB, KiB, MiB, or GiB (case-insensitive). The default, `NULL`, sizes native buffers automatically from decoded rows and uses `chunk_rows` to end each chunk.

`columns` A character vector of column names, or `NULL` for all columns.

`encoding` A character string specifying the source encoding, or `NULL` to use the encoding recorded in the file.

`dates` Whether to convert recognized SAS dates and datetimes to Parquet date and UTC timestamp types. SAS times retain double seconds and are restored as `hms` when imported into R with arrow. Set to `FALSE` to retain all original SAS numeric values.

`compression` Parquet compression codec, for example `"snappy"`, `"zstd"`, or `"uncompressed"`, supported by your arrow installation.

`missing_tags` Whether to retain SAS special missing tags in additional character columns. The default, `FALSE`, keeps the original column set. Set to `TRUE` to add one `<name>__sas_missing` column for each numeric source column, with unique names when necessary. These contain `"a"` through `"z"` or `"_"` for special missing values, and null otherwise. Numeric missing values themselves become Parquet nulls. With `FALSE`, ordinary and special missing values both become nulls; their missing status is preserved but special codes are discarded.

`progress` A logical value. Set to `TRUE` to show a console progress bar, or `FALSE` to hide it. The default, `interactive()`, shows the bar in interactive R sessions and hides it when running via `Rscript`. Progress uses the SAS row count and updates after each batch is written. It reaches 100% only after the completed output file is published.

Details

Native decoded batches are exported through the Arrow C Data Interface. R coordinates the writer and holds small schema and batch handles; dataset values remain in native buffers. The persistent SAS reader, column selection, UTF-8 decoding, adaptive buffer sizing, and bounded read-ahead are shared with `sas7bdat_read()`. Arrow is needed only for this conversion function.

Each decoded batch becomes a Parquet row group, with at most `chunk_rows` rows. An explicit `chunk_bytes` can produce smaller groups; the default `NULL` lets the row limit control batch sizes. Native output batches are released after each write; the whole dataset is never collected.

With `dates = TRUE`, recognized SAS dates become Parquet dates (days since 1970-01-01), and datetimes become UTC timestamps rounded to the nearest microsecond. Fractional dates and out-of-range temporal values cause an error. SAS times remain double seconds, preserving negative times, values beyond 24 hours, and fractional seconds. R metadata restores their `hms` class and `HH:MM:SS` display when read with `arrow::read_parquet()` or converted from an Arrow table to an R data frame. With `dates = FALSE`, all numeric values retain their original SAS units and double precision, including sub-microsecond datetime values.

File metadata retains `sas.table_name`, `sas.label`, and `sas.encoding`. Each field retains `sas.label`, `sas.format`, `sas.type`, `sas.units`, and `sas.missing_tags`; tag fields identify their source in `sas.missing_for`. R metadata also restores column `label` and `format.sas` attributes and the file label on import. Other Parquet clients see times as double seconds. Inspect the schema metadata with `arrow::read_parquet(output, as_data_frame = FALSE)$schema`.

Conversion writes a temporary file in the output directory and publishes it only after reading and writing finish successfully. Errors and interrupts close the reader and remove the incomplete temporary output. Empty SAS files produce a valid empty Parquet file with the selected schema.

Value

An invisible list containing `path`, `rows`, `chunks`, `completed`, and `missing_tag_columns`, a named mapping from numeric source columns to their tag columns (empty when `missing_tags = FALSE`).

Examples

```
if (requireNamespace("arrow", quietly = TRUE) &&
    utils::packageVersion("arrow") >= "18.1.0" && arrow::arrow_with_parquet()) {
  path <- system.file("examples", "example_rle.sas7bdat",
                      package = "anotherSAS7bdat")
  output <- tempfile(fileext = ".parquet")
  sas7bdat_to_parquet(path, output)
  data <- arrow::read_parquet(output)
  unlink(output)
}
```

Index

sas7bdat_close *(sas7bdat_read)*, 2
sas7bdat_info *(sas7bdat_read)*, 2
sas7bdat_open *(sas7bdat_read)*, 2
sas7bdat_read, 2
sas7bdat_read_chunk *(sas7bdat_read)*, 2
sas7bdat_to_parquet, 6