Package 'dbcturbo'

November 21, 2026

Type Package

Title High-Performance Streaming Reader for 'DATASUS' DBC Files

Version 0.3.0

Date 2026-11-20

Description A modern, memory-efficient R package for reading and converting DBC files produced by the Brazilian Ministry of Health's 'DATASUS' system. DBC files are DBF databases compressed with the 'PKWare' implode algorithm. Unlike 'read.dbc', this package processes files in streaming mode, never loading the full dataset into RAM. It can export directly to CSV or DBF and provides R helpers for in-memory reading and Parquet conversion.

License AGPL-3

URL https://github.com/GPimentel14/dbcturbo

BugReports https://github.com/GPimentel14/dbcturbo/issues

Encoding UTF-8

Depends R (>= 4.0.0)

NeedsCompilation yes

SystemRequirements C99 compiler

Suggests data.table (>= 1.14.0), arrow (>= 12.0.0), testthat (>= 3.0.0), knitr, rmarkdown

VignetteBuilder knitr

Config/roxygen2/version 8.1.0

Author Gumercindo Pimentel Peralta [aut, cre] (ORCID: <https://orcid.org/0000-0002-5896-3171>), Juliana da Silva [ths] (ORCID: <https://orcid.org/0000-0002-1089-6766>), Mark Adler [ctb] (Author of blast.c (PKWare implode decompressor)), Daniela Petruzalek [ctb] (Author of read.dbc, dbc2dbf.c), Pablo Fonseca [ctb] (Author of blast-dbf)

Maintainer Gumercindo Pimentel Peralta <gumercindopimentel@gmail.com>

Repository CRAN

Date/Publication 2026-11-21 03:30:02 UTC

Contents

dbc2dbf 2
dbc_batch_to_csv 3
dbc_inspect 4
dbc_to_csv 5
dbc_to_parquet 6
dbc_validate 7
read_dbc 8
read_dbc_chunks 10

Index **11**

`dbc2dbf` *Decompress a DATASUS DBC file to DBF format*

Description

Calls the C streaming engine to decompress a `.dbc` file into a standard `.dbf` file without loading the full payload into RAM. The compressed payload is processed in 64 KB chunks using the PKWare Implode algorithm (`blast.c`, Mark Adler).

Usage

```
dbc2dbf(input_file, output_file)
```

Arguments

`input_file`	Character string. Path to the source `.dbc` file. Must exist and be readable.
`output_file`	Character string. Path for the output `.dbf` file. Created or overwritten.

Value

`TRUE` invisibly on success. On failure, stops with a descriptive error message.

See Also

`dbc_to_csv`, `dbc_inspect`

Examples

```
dbc <- system.file("extdata", "sids.dbc", package = "dbcturbo")
dbf <- tempfile(fileext = ".dbf")
dbc2dbf(dbc, dbf)
file.exists(dbf)
unlink(dbf)
```

dbc_batch_to_csv *Convert all DBC files in a directory to CSV files*

Description

Finds DBC files in a directory and converts each one with dbc_to_csv. When recursive = TRUE, the relative directory layout beneath input_dir is preserved in output_dir. Existing output files are never replaced unless overwrite = TRUE.

Usage

```
dbc_batch_to_csv(
  input_dir,
  output_dir,
  pattern = "\\.dbc$",
  recursive = FALSE,
  batch_size = 4096L,
  encoding = NULL,
  cols = NULL,
  overwrite = FALSE,
  verbose = FALSE,
  workers = 1L
)
```

Arguments

input_dir	Character string. Directory containing DBC files.
output_dir	Character string. Destination directory for CSV files. Created when it does not exist.
pattern	Regular expression used to select input files. Defaults to "\\.dbc$", case-insensitively.
recursive	Logical. Search subdirectories? Default FALSE.
batch_size	Integer. Number of records per CSV write batch. Passed to dbc_to_csv.
encoding	Character string or NULL. Passed to dbc_to_csv.
cols	Character vector or NULL. Optional fields to write for every input file. Passed to dbc_to_csv.
overwrite	Logical. Replace existing CSV files? Default FALSE.
verbose	Logical. Print per-file conversion progress? Default FALSE.
workers	Positive integer. Number of files to convert concurrently on macOS and Linux. Defaults to 1L. Windows uses sequential execution because parallel::mclapply() is not available there.

Value

A data frame with one row per converted file and columns input_file and output_file. For an empty input directory, returns a zero-row data frame with those columns.

Examples

```
input_dir <- tempfile("dbc-input-")
output_dir <- tempfile("dbc-output-")
dir.create(input_dir)
file.copy(system.file("extdata", "sids.dbc", package = "dbcturbo"),
          file.path(input_dir, "sids.dbc"))
result <- dbc_batch_to_csv(input_dir, output_dir)
result
unlink(c(input_dir, output_dir), recursive = TRUE)
```

`dbc_inspect` *Inspect the metadata of a DATASUS DBC file without decompressing data*

Description

Reads only the DBF header embedded in the `.dbc` file and returns field metadata and the total record count. The compressed data payload is never touched, making this function very fast even for large files.

Usage

```
dbc_inspect(input_file)
```

Arguments

`input_file`	Character string. Path to the `.dbc` file.

Value

A named list with:

`fields` A `data.frame` with columns `name` (character), `type` (one-character string: `C`=text, `N`=numeric, `D`=date, `L`=logical), `width` (integer), `decimals` (integer).

`nrecords` Integer. Total number of records.

See Also

`dbc_to_csv`, `dbc2dbf`

Examples

```
dbc <- system.file("extdata", "sids.dbc", package = "dbcturbo")
meta <- dbc_inspect(dbc)
meta$nrecords
head(meta$fields)
```

dbc_to_csv *Convert a DATASUS DBC file directly to CSV (streaming, low-RAM)*

Description

Decompresses a `.dbc` file and writes a UTF-8 CSV to disk. Records are processed in batches, so peak RAM usage is `O(batch_size * record_width)` regardless of file size.

Usage

```
dbc_to_csv(
  input_file,
  output_file,
  batch_size = 4096L,
  encoding = NULL,
  cols = NULL,
  verbose = FALSE,
  progress = NULL,
  ...
)
```

Arguments

`input_file`	Character string. Path to the source `.dbc` file.
`output_file`	Character string. Path for the output `.csv` file.
`batch_size`	Integer `>= 1`. Number of records per write batch. Default `4096L`.
`encoding`	Character string or `NULL`. Source encoding of character fields (e.g. `"CP850"`, `"latin1"`). `NULL` (default) assumes ASCII / UTF-8.
`cols`	Character vector or `NULL`. Names of fields to write. `NULL` (default) writes every field. Selection is applied by the C writer, so unselected fields are not written to the CSV.
`verbose`	Logical. If `TRUE`, prints file metrics, a progress bar, and elapsed time upon completion. Default `FALSE`.
`progress`	Function or `NULL`. Custom callback called after each batch with two numeric arguments: `done` and `total` (record counts). Ctrl+C is honoured between batches.
`...`	Optional internal arguments.

Details

The output CSV is prefixed with a UTF-8 BOM so that Microsoft Excel opens it with correct encoding without additional configuration.

Value

`TRUE` invisibly on success. On failure, stops with a descriptive error message.

See Also

dbc2dbf, dbc_inspect

Examples

```
dbc <- system.file("extdata", "sids.dbc", package = "dbcturbo")
csv <- tempfile(fileext = ".csv")
dbc_to_csv(dbc, csv)
head(utils::read.csv(csv, fileEncoding = "UTF-8"))
unlink(csv)
```

---

dbc_to_parquet *Convert a DATASUS DBC file to Parquet format*

Description

Streams typed chunks directly from a `.dbc` file into a `.parquet` file using `read_dbc_chunks` and **arrow**'s `ParquetFileWriter`, without creating an intermediate temporary CSV file.

Usage

```
dbc_to_parquet(
  input_file,
  output_file,
  batch_size = 100000L,
  encoding = "latin1",
  verbose = FALSE,
  progress = NULL,
  cols = NULL
)
```

Arguments

`input_file`	Character string. Path to the source `.dbc` file.
`output_file`	Character string. Path for the output `.parquet` file.
`batch_size`	Integer. Number of rows per streamed chunk / row group. Default `100000L`.
`encoding`	Character string. Source encoding of character fields. Default `"latin1"`.
`verbose`	Logical. If `TRUE`, prints file metrics, a progress bar, and elapsed time upon completion. Default `FALSE`.
`progress`	Function or `NULL`. Optional callback for progress reporting.
`cols`	Character vector or `NULL`. Names of columns to include in the Parquet file. `NULL` (default) writes all columns.

Details

This is the recommended workflow for Big Data and epidemiological research, as Parquet files are heavily compressed, columnar, and preserve types.

Value

TRUE invisibly on success. Stops if the **arrow** package is not installed.

Examples

```
if (requireNamespace("arrow", quietly = TRUE)) {
  dbc <- system.file("extdata", "sids.dbc", package = "dbcturbo")
  parquet <- tempfile(fileext = ".parquet")
  dbc_to_parquet(dbc, parquet)
  arrow::read_parquet(parquet)
  unlink(parquet)
}
```

---

dbc_validate *Validate the structural integrity of a DATASUS DBC file*

Description

Checks the embedded DBF header, field descriptors, record width, and file boundaries. With deep = TRUE, also decompresses the complete payload and verifies the declared record count, record markers, stream termination, and absence of unexpected trailing data. No data frame or temporary DBF file is created.

Usage

```
dbc_validate(file, deep = TRUE)
```

Arguments

file	Character string. Path to the .dbc file.
deep	Logical. If TRUE (default), decompress and validate every record. If FALSE, validate only the embedded DBF header.

Details

This is a structural and decompression check. It does not compare the contents with an external reference dataset.

Value

TRUE invisibly when the file passes validation. Invalid, truncated, or corrupt files raise an error.

See Also

dbc_inspect, read_dbc

Examples

```
dbc <- system.file("extdata", "sids.dbc", package = "dbcturbo")
dbc_validate(dbc)
```

---

read_dbc *Read a DATASUS DBC file into an R data frame*

Description

High-level convenience wrapper. By default the native engine decodes the whole file directly into R columns without temporary files, which is the fastest option at every file size. Set engine to "data.table" or "base", or pass CSV-reader arguments through ..., to use the temporary-CSV route instead.

Usage

```
read_dbc(
  file,
  batch_size = 4096L,
  encoding = "latin1",
  verbose = FALSE,
  cols = NULL,
  coerce_types = TRUE,
  engine = c("auto", "native", "data.table", "base"),
  native_threshold = Inf,
  n_max = Inf,
  ...
)
```

Arguments

file	Character string. Path to the .dbc file.
batch_size	Integer. Passed to dbc_to_csv. Default 4096L.
encoding	Character string. Source encoding of character fields. Default "latin1" (standard for modern DATASUS files; use "CP850" for older DOS-era files).
verbose	Logical. Passed to dbc_to_csv. Default FALSE.
cols	Character vector or NULL. Names of the columns to keep in the returned object. NULL (default) returns all columns. Column names are validated against dbc_inspect metadata and selection is applied before the temporary CSV is written.

`coerce_types` Logical. If `TRUE` (default), columns are automatically converted to their native R types based on the DBF field metadata:

- `D` (date) → `Date` (DBF stores dates as `YYYYMMDD` strings; blank or `"00000000"` become `NA`).
- `N` with decimals > 0 → `numeric`.
- `N` with decimals == 0 → `integer` when values fit in R integers, otherwise `numeric`.
- `L` (logical) → `logical` (`"T"`/`"Y"`/`"S"`/`"1"` → `TRUE`; `"F"`/`"N"`/`"0"` → `FALSE`; others → `NA`).
- `C` (character) → unchanged.

`engine` Character string selecting the reader. `"auto"` (default) uses the native engine whenever no extra CSV-reader arguments are supplied; otherwise it falls back to **data.table** when installed and R base otherwise. `"native"` always uses direct native decoding; `"data.table"` requires **data.table**; `"base"` always uses `utils::read.csv()`.

`native_threshold` Non-negative number of bytes. In `"auto"` mode, files at or below this size use the native engine. Defaults to `Inf` (native for every size). Set to `0` to always use the CSV route.

`n_max` Non-negative number. Maximum number of active (non-deleted) records to read. Defaults to `Inf` (read all records). In the native engine, decompression stops immediately once `n_max` rows are read.

`...` Additional arguments forwarded to the CSV reader.

Details

The native engine transcodes character fields to UTF-8 in C as it decodes, so its result is identical to the CSV engines without an extra R-side `iconv()` pass.

Value

A `data.frame` or `data.table` (if **data.table** is installed).

Examples

```
dbc <- system.file("extdata", "sids.dbc", package = "dbcturbo")
df <- read_dbc(dbc)
head(df)

# Read a subset of fields.
fields <- dbc_inspect(dbc)$fields$name[1:3]
selected <- read_dbc(dbc, cols = fields)
names(selected)
```

read_dbc_chunks *Read a DATASUS DBC file in memory-bounded chunks*

Description

Streams a DBC file once and calls `callback` with each data-frame chunk. A chunk is no longer retained by dbcturbo after the callback returns, so memory use is approximately proportional to `chunk_size` unless the callback stores the chunks.

Usage

```
read_dbc_chunks(
  file,
  callback,
  chunk_size = 100000L,
  encoding = "latin1",
  cols = NULL,
  coerce_types = TRUE
)
```

Arguments

`file`	Character string. Path to the DBC file.
`callback`	Function called with one data frame per chunk.
`chunk_size`	Positive integer. Maximum number of non-deleted records per chunk. Defaults to `100000L`.
`encoding`	Source encoding for character fields. Defaults to `"latin1"`.
`cols`	Character vector or `NULL`. Fields to include.
`coerce_types`	Logical. Convert DBF numeric, logical, and date fields to native R types. Defaults to `TRUE`.

Value

Total number of delivered rows, invisibly.

See Also

`read_dbc`, `dbc_validate`

Examples

```
dbc <- system.file("extdata", "sids.dbc", package = "dbcturbo")
rows <- 0L
read_dbc_chunks(dbc, chunk_size = 25L, callback = function(x) {
  rows <<- rows + nrow(x)
})
rows
```

Index

dbc2dbf, 2, *4*, *6*
dbc_batch_to_csv, 3
dbc_inspect, *2*, 4, *6*, *8*
dbc_to_csv, *2–4*, 5, *8*
dbc_to_parquet, 6
dbc_validate, 7, *10*

read_dbc, *8*, 8, *10*
read_dbc_chunks, *6*, 10