

Package ‘dbcturbo’

October 10, 2026

Type Package

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

Version 0.2.0

Date 2026-09-30

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 Gumerindo 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 Gumerindo Pimentel Peralta <gumercindopimentel@gmail.com>

Repository CRAN

Date/Publication 2026-10-10 11:20:20 UTC

Contents

dbc2dbf	2
dbc_batch_to_csv	3
dbc_inspect	4
dbc_to_csv	5
dbc_to_parquet	6
read_dbc	7
Index	9

dbc2dbf	<i>Decompress a DATASUS DBC file to DBF format</i>
---------	--

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	<i>Convert all DBC files in a directory to CSV files</i>
------------------	--

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

<code>input_dir</code>	Character string. Directory containing DBC files.
<code>output_dir</code>	Character string. Destination directory for CSV files. Created when it does not exist.
<code>pattern</code>	Regular expression used to select input files. Defaults to <code>"*.dbc\$"</code> , case-insensitively.
<code>recursive</code>	Logical. Search subdirectories? Default <code>FALSE</code> .
<code>batch_size</code>	Integer. Number of records per CSV write batch. Passed to dbc_to_csv .
<code>encoding</code>	Character string or <code>NULL</code> . Passed to dbc_to_csv .
<code>cols</code>	Character vector or <code>NULL</code> . Optional fields to write for every input file. Passed to dbc_to_csv .
<code>overwrite</code>	Logical. Replace existing CSV files? Default <code>FALSE</code> .
<code>verbose</code>	Logical. Print per-file conversion progress? Default <code>FALSE</code> .
<code>workers</code>	Positive integer. Number of files to convert concurrently on macOS and Linux. Defaults to 1L. Windows uses sequential execution because <code>parallel::mclapply()</code> 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	<i>Inspect the metadata of a DATASUS DBC file without decompressing data</i>
-------------	--

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 $O(\text{batch_size} * \text{record_width})$ 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	<i>Convert a DATASUS DBC file to Parquet format</i>
----------------	---

Description

A high-level convenience wrapper that first writes the .dbc file to a temporary CSV using the C engine, then converts that CSV into a .parquet file using the **arrow** package. It therefore requires temporary disk space for the CSV and is not an end-to-end streaming writer.

Usage

```

dbc_to_parquet(
  input_file,
  output_file,
  batch_size = 8192L,
  encoding = "CP850",
  verbose = FALSE,
  progress = 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. Passed to dbc_to_csv . Default 8192L.
encoding	Character string. Source encoding of character fields. Default "CP850".
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.

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)
}
```

read_dbc

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

High-level convenience wrapper. The native engine decodes small files directly into R columns without temporary files. Larger files use a temporary CSV and `data.table::fread()` (if available) or `utils::read.csv()`.

Usage

```
read_dbc(
  file,
  batch_size = 4096L,
  encoding = "CP850",
  verbose = FALSE,
  cols = NULL,
  coerce_types = TRUE,
  engine = c("auto", "native", "data.table", "base"),
  native_threshold = 50 * 1024^2,
  ...
)
```

Arguments

<code>file</code>	Character string. Path to the .dbc file.
<code>batch_size</code>	Integer. Passed to <code>dbc_to_csv</code> . Default 4096L.
<code>encoding</code>	Character string. Source encoding of character fields. Default "CP850" (standard DATASUS legacy encoding).
<code>verbose</code>	Logical. Passed to <code>dbc_to_csv</code> . 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 <code>dbc_inspect</code> 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 for files no larger than <code>native_threshold</code> , then data.table when installed and R base otherwise. "native" always uses direct native decoding; "data.table" requires data.table ; "base" always uses <code>utils::read.csv()</code> .
native_threshold	Non-negative number of bytes. In "auto" mode, files at or below this size use the native engine. Defaults to 50 MiB. Set to 0 to always use the CSV route in "auto".
...	Additional arguments forwarded to the CSV reader.

Details

For files with more than one million records, use `dbc_to_csv` directly and load the resulting CSV with `arrow::read_csv_arrow()` or `data.table::fread()` for maximum performance.

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)
```

Index

`dbc2dbf`, [2](#), [4](#), [6](#)

`dbc_batch_to_csv`, [3](#)

`dbc_inspect`, [2](#), [4](#), [6](#), [8](#)

`dbc_to_csv`, [2–4](#), [5](#), [6–8](#)

`dbc_to_parquet`, [6](#)

`read_dbc`, [7](#)