

Package ‘marcxmlr’

September 20, 2026

Title Faithful and Scalable MARCXML Parsing

Version 0.2.1

Description Parses Machine-Readable Cataloging ('MARC 21') XML
<<https://www.loc.gov/standards/marcxml/>> into a canonical tidy long
representation while preserving leaders, control fields, data fields,
indicators, repeated fields, repeated subfields, and source order.
Provides an in-memory reader for manageable catalogues and a
bounded-memory converter that writes larger collections as 'Parquet'
datasets, with optional local parallel processing.

License MIT + file LICENSE

URL <https://github.com/larry77/marcxmlr>

BugReports <https://github.com/larry77/marcxmlr/issues>

Encoding UTF-8

Language en-US

RoxygenNote 7.3.2

Depends R (>= 4.1.0)

Imports purrr (>= 1.0.0), rlang, stats, tibble (>= 3.0.0), xml2 (>= 1.3.0)

Suggests arrow, dplyr, future (>= 1.69.0), future.mirai, futurize,
furry, mori, testthat (>= 3.0.0), XML

Config/testthat/edition 3

SystemRequirements libxml2 (>= 2.9.0)

NeedsCompilation yes

Author Lorenzo Isella [aut, cre]

Maintainer Lorenzo Isella <lorenzo.isella@gmail.com>

Repository CRAN

Date/Publication 2026-09-20 00:40:02 UTC

Contents

marcxmlr-package	2
marcxml_to_parquet	2
read_marcxml	4
Index	6

marcxmlr-package	<i>marcxmlr: Faithful and Scalable MARCXML Parsing</i>
------------------	--

Description

Parse MARC21 XML into a canonical tidy long representation while preserving repeated structures and source order. Use `read_marcxml()` for in-memory work and `marcxml_to_parquet()` for bounded-memory conversion to a disk-backed Parquet dataset.

See Also

- Useful links:
- <https://www.loc.gov/standards/marcxml/>
 - <https://www.loc.gov/marc/marcdocz.html>

marcxml_to_parquet	<i>Convert a MARCXML collection to a Parquet dataset</i>
--------------------	--

Description

`marcxml_to_parquet()` streams complete MARCXML records from a collection, parses them in bounded batches, and writes the canonical long representation as a directory of Parquet files. It does not construct a DOM for the complete XML document and does not materialize the complete parsed result in R.

Usage

```
marcxml_to_parquet(  
  file,  
  output_dir,  
  batch_records = 5000L,  
  workers = 1L,  
  chunk_records = NULL,  
  compression = "snappy",  
  verbose = TRUE  
)
```

Arguments

file	One or more MARCXML collection paths, or glob patterns such as "catalogue/*.xml". Glob matches are processed in sorted order.
output_dir	Path for the new Parquet dataset directory. It must not already exist. The directory is published only after successful conversion.
batch_records	Maximum number of records converted into one bounded canonical batch before writing. This bounds normal working memory, though an unusually large individual record can itself require substantial memory.
workers	Number of local worker processes. The default, 1, is sequential. Values greater than one require the optional parallel packages listed in Suggests and, with current dependency versions, R 4.3 or later.
chunk_records	Number of records assigned to each parsing and writing task. NULL targets approximately two tasks per worker in each batch.
compression	Parquet compression codec passed to <code>arrow::write_parquet()</code> .
verbose	Whether to report cumulative records and files after each completed batch.

Details

The input must have a collection root in the official MARCXML namespace (<http://www.loc.gov/MARC21/slim>) or no namespace. A standalone record can be read with `read_marcxml()` but is not accepted by this collection converter.

With `workers = 1` and default `chunk_records = NULL`, supported ordinary collections use a two-pass native libxml2 engine. The first pass validates and counts the collection; the second fills bounded canonical batches directly from `xmlTextReaderExpand()` nodes and writes them with `arrow`. No record XML is serialized or reparsed on this path.

For one input file, unsupported input, explicit `chunk_records`, and parallel calls retain the established serialized-record/native or XML event-stream implementations.

When `file` resolves to multiple files, complete files are the unit of parallel work. Each file is parsed by the existing sequential engine and writes independent Parquet fragments. Global `record_id` values are assigned deterministically in resolved file order, regardless of worker completion order. XML/libxml2 external pointers are never sent to workers. Explicit `chunk_records` is not supported for multi-file input.

Parallel work is dispatched through a temporary `future.mirai` plan using `futurize`; the caller's previous future plan is restored on exit.

Each task writes a uniquely named temporary file and renames it only after a successful Parquet write. All files are first written under a staging directory beside `output_dir`; the completed directory is renamed into place only after the XML input has been fully processed. Existing output is never overwritten.

Open the result with `arrow::open_dataset(output_dir)`. Opening a dataset is lazy; calling `collect()` on the entire dataset will nevertheless materialize every row in R memory.

Value

Invisibly, a tibble with one row per resolved input file containing the normalized input and output paths, record and row counts, number of batches, and number of Parquet files. Single-file input therefore retains the existing one-row return value. Parsed rows remain in the dataset directory.

Examples

```
if (requireNamespace("XML", quietly = TRUE) &&
    requireNamespace("arrow", quietly = TRUE)) {
  example_file <- system.file(
    "extdata", "example-marxml.xml", package = "marxmlr"
  )
  output <- tempfile("marxml-parquet-")

  conversion <- marxml_to_parquet(
    example_file,
    output_dir = output,
    workers = 1L,
    verbose = FALSE
  )

  dataset <- arrow::open_dataset(output)
  conversion
  dataset

  unlink(output, recursive = TRUE)
}
```

read_marxml

Read MARXML into a canonical long tibble

Description

`read_marxml()` reads a MARC21 XML collection or a standalone record and returns one row for each leader, control field, or data-field subfield. It preserves repeated fields, repeated subfields, indicators, and source order.

Usage

```
read_marxml(file, n_max = Inf, workers = 1L, chunk_records = NULL)
```

Arguments

<code>file</code>	Path to a MARXML file.
<code>n_max</code>	Maximum number of records to parse. Use <code>Inf</code> for every record or <code>0</code> to return an empty result with the canonical schema.
<code>workers</code>	Number of local worker processes. The default, <code>1</code> , is sequential. Values greater than one require the optional parallel packages listed in Suggests and, with current dependency versions, R 4.3 or later.
<code>chunk_records</code>	Number of records assigned to each parsing task. <code>NULL</code> creates one task in sequential mode and approximately two tasks per worker in parallel mode.

Details

This function materializes the parsed result in memory. On the supported sequential native path it does not build a DOM for the complete XML input; compatibility fallbacks may do so. Use `marxml_to_parquet()` for catalogues whose canonical result may not fit in memory.

`record_id` is positional identity in the selected input and is not derived from control field 001. `field_order` is zero for the leader and then counts variable fields from one. `field_occurrence` counts occurrences of a field type and tag within a record.

Data-field rows carry `subfield_order`, the position within the containing field, and `subfield_occurrence`, the occurrence of that code within the same field. Structural columns that do not apply are NA.

With `workers = 1` and default `chunk_records = NULL`, supported ordinary input uses a two-pass native libxml2 engine: the first pass validates and counts selected records and the second fills the canonical columns directly from expanded record nodes. No record XML is serialized or reparsed on this path. Unsupported input falls back to the reference xml2 implementation.

Parallel parsing retains the established serialized-record implementation. XML/libxml2 external pointers are never sent to workers. The caller's previous future plan is restored on exit.

Value

A tibble with columns `record_id`, `field_type`, `tag`, `subfield_code`, `value`, `field_order`, `field_occurrence`, `ind1`, `ind2`, `subfield_order`, and `subfield_occurrence`, in that order.

Examples

```
example_file <- system.file(
  "extdata", "example-marxml.xml", package = "marxmlr"
)

records <- read_marxml(example_file)
records

records[
  records$record_id == 1L & records$tag == "856",
  c("subfield_code", "value", "subfield_order", "subfield_occurrence")
]
```

Index

`arrow::write_parquet()`, [3](#)

`marxml_to_parquet`, [2](#)

`marxmlr` (`marxmlr`-package), [2](#)

`marxmlr`-package, [2](#)

`read_marxml`, [4](#)

`read_marxml()`, [3](#)