Package ‘marcxmlr’

September 12, 2026

Title Faithful and Scalable MARCXML Parsing

Version 0.1.0

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), stats, tibble (>= 3.0.0), xml2 (>= 1.3.0)

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

Config/testthat/edition 3

NeedsCompilation no

Author Lorenzo Isella [aut, cre]

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

Repository CRAN

Date/Publication 2026-09-12 14:30:02 UTC

Contents

`marcxmlr-package` *marcxmlr: Faithful and Scalable MARCXML Parsing*

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` *Convert a MARCXML collection to a Parquet dataset*

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	Path to a MARCXML collection.
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 serialized records retained in a batch before parsing and 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 arrow::write_parquet().
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 converter.

Complete records are serialized in the main process before parallel work. This prevents XML external pointers from crossing process boundaries. With multiple workers, record strings are exposed through mori shared memory, and futurize dispatches purrr tasks through a temporary future.mirai plan. The 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 one-row tibble containing normalized input and output paths, record and row counts, number of batches, and number of Parquet files. Parsed rows remain in the dataset directory.

Examples

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

  conversion <- marcxml_to_parquet(
    example_file,
```

```
  output_dir = output,
  workers = 1L,
  verbose = FALSE
)

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

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

`read_marcxml` *Read MARCXML into a canonical long tibble*

Description

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

Arguments

`file`	Path to a MARCXML file.
`n_max`	Maximum number of records to parse. Use `Inf` for every record or `0` to return an empty result with the canonical schema.
`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 task. `NULL` creates one task in sequential mode and approximately two tasks per worker in parallel mode.

Details

This function materializes both the XML input and parsed result in memory. Use `marcxml_to_parquet()` for catalogues that 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`.

Parallel parsing serializes complete records before dispatch, so `xml2` 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-marcxml.xml", package = "marcxmlr"
)

records <- read_marcxml(example_file)
records

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

Index