Package: nirs4allformats (via r-universe)

June 12, 2026

Title Read Near-Infrared Spectroscopy and Spectral File Formats

Version 0.1.0

Description Thin R binding for the Rust-first 'nirs4all-formats' near-infrared spectroscopy (NIRS) file-loading core. When installed via R CMD INSTALL with Cargo available, the package compiles a native 'extendr' static library from 'src/rust/' and dispatches probe, read, and walk calls directly through Rust. Without Cargo it falls back to invoking the 'nirs4all-formats' command-line interface. This is the complete build: it ships every reader, including the optional large ones (HDF5/netCDF, Parquet/Arrow, MATLAB) on top of the core readers (JCAMP-DX, Galactic SPC, Bruker OPUS, ASD, ENVI, CSV, Excel, and many vendor ASCII/binary formats). A smaller sibling package 'nirs4allformats.lite' drops only the Parquet/Arrow reader for size-sensitive installs.

SystemRequirements Cargo (Rust's package manager), rustc

URL https://github.com/GBeurier/nirs4all-formats

BugReports https://github.com/GBeurier/nirs4all-formats/issues

License MIT + file LICENSE

Encoding UTF-8

Imports jsonlite

Suggests testthat (>= 3.0.0), tibble

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.2

Config/testthat/edition 3

Config/pak/sysreqs libclang-dev

Repository https://gbeurier.r-universe.dev

Date/Publication 2026-06-12 08:00:47 UTC

RemoteUrl https://github.com/GBeurier/nirs4all-formats

RemoteRef HEAD

RemoteSha 7ebcbc1d5a709c97ce2247fa309065078b9ba517

RemoteSubdir bindings/r/nirs4allformats

Contents

`as.data.frame.nirs4allformats_dataset`

Build a wide data frame from a dataset

Description

`as.data.frame()` method for nirs4allformats_dataset objects. Returns a wide table whose first column is `sample_id`, followed by any target columns, followed by one spectral column per wavelength. Spectral columns are named `x_<wavelength>` (the axis value formatted without scientific notation).

Usage

```
## S3 method for class 'nirs4allformats_dataset'
as.data.frame(x, row.names = NULL, optional = FALSE, ...)
```

Arguments

`x`	An `nirs4allformats_dataset` from `nirs4allformats_open_dataset()`.
`row.names`	Ignored; present for S3 method signature compatibility.
`optional`	Ignored; present for S3 method signature compatibility.
`...`	Ignored; present for S3 method consistency.

Value

A `data.frame` with columns `sample_id`, the target columns (if any), and `x_<wavelength>` spectral columns.

See Also

nirs4allformats_open_dataset(), as.matrix.nirs4allformats_dataset(), nirs4allformats_as_tibble().

Examples

```
## Not run:
ds <- nirs4allformats_open_dataset("samples/csv_tsv/synthetic_nirs.csv")
df <- as.data.frame(ds)
names(df)[1:5]

## End(Not run)
```

---

as.matrix.nirs4allformats_dataset

Extract the spectral matrix from a dataset

Description

as.matrix() method for nirs4allformats_dataset objects. Returns the stored n_samples x n_wavelengths numeric matrix of spectra. Rows correspond to x$sample_ids and columns to x$wavelengths.

Usage

```
## S3 method for class 'nirs4allformats_dataset'
as.matrix(x, ...)
```

Arguments

x	An nirs4allformats_dataset from nirs4allformats_open_dataset().
...	Ignored; present for S3 method consistency.

Value

A numeric matrix with one row per sample and one column per wavelength.

See Also

nirs4allformats_open_dataset(), as.data.frame.nirs4allformats_dataset().

Examples

```
## Not run:
ds <- nirs4allformats_open_dataset("samples/csv_tsv/synthetic_nirs.csv")
m <- as.matrix(ds)
dim(m)

## End(Not run)
```

nirs4allformats_as_tibble

Convert a dataset to a tibble

Description

Convenience wrapper that converts an nirs4allformats_dataset to a tibble via `as.data.frame.nirs4allformats_dataset()`. The optional `tibble` package must be installed.

Usage

```
nirs4allformats_as_tibble(dataset)
```

Arguments

`dataset`	An `nirs4allformats_dataset` from `nirs4allformats_open_dataset()`.

Value

A `tibble::tibble` with the same columns as `as.data.frame.nirs4allformats_dataset()` (`sample_id`, target columns, `x_<wavelength>` spectral columns).

See Also

`nirs4allformats_open_dataset()`, `as.data.frame.nirs4allformats_dataset()`.

Examples

```
## Not run:
ds <- nirs4allformats_open_dataset("samples/csv_tsv/synthetic_nirs.csv")
nirs4allformats_as_tibble(ds)

## End(Not run)
```

nirs4allformats_native_available

Report whether the native Rust backend is loaded

Description

Returns `TRUE` when the compiled extendr static library is registered in the running R session, i.e. the package was installed with Cargo on `PATH` and the Rust core is callable directly. When `FALSE`, filesystem reads fall back to the `nirs4all-formats` CLI, and the in-memory paths (`nirs4allformats_open_bytes()`, `nirs4allformats_open_with_sidecars()`) are unavailable.

Usage

```
nirs4allformats_native_available()
```

Value

A length-one logical: TRUE if the native backend is loaded, otherwise FALSE.

See Also

nirs4allformats_open_bytes(), nirs4allformats_open_with_sidecars().

Examples

```
## Not run:
if (nirs4allformats_native_available()) {
  message("native extendr backend active")
} else {
  message("using nirs4all-formats CLI fallback")
}

## End(Not run)
```

---

nirs4allformats_open_bytes

Decode raw bytes through the native registry

Description

Decodes an in-memory byte buffer through the Rust registry and returns the normalized records as nested R lists, without touching the filesystem. The name drives extension-based sniffing and provenance. This path requires the native extendr static library; it is unavailable through the CLI fallback and raises an error when the native library is absent.

Formats that need companion files (sidecars) are rejected here; use nirs4allformats_open_with_sidecars() for those.

Usage

```
nirs4allformats_open_bytes(name, bytes)
```

Arguments

name	Character scalar. Logical file name (with extension) used for format sniffing and recorded in provenance, e.g. "spectrum.dx".
bytes	A raw vector containing the file contents.

Value

A list of records, identical in shape to nirs4allformats_open_records().

See Also

nirs4allformats_open_with_sidecars(), nirs4allformats_open_records(), nirs4allformats_native_available

Examples

```
## Not run:
bytes <- readBin("spectrum.dx", what = "raw",
                 n = file.info("spectrum.dx")$size)
records <- nirs4allformats_open_bytes("spectrum.dx", bytes)

## End(Not run)
```

---

nirs4allformats_open_dataset

Read a spectroscopy file into a flat spectral dataset

Description

Loads a file with nirs4allformats_open_records() and projects one signal per record into a rectangular, R-friendly dataset: a samples-by-wavelengths matrix plus sample IDs, targets and metadata. All records must share the same spectral axis (an error is raised otherwise), so this is intended for a homogeneous set of spectra. For heterogeneous or N-dimensional data, work from nirs4allformats_open_records() directly.

Parsing happens only in Rust; the R layer just selects a signal and reshapes the JSON the core returns.

Usage

```
nirs4allformats_open_dataset(path, signal = NULL)
```

Arguments

path	Character scalar. Path to the input file (resolved with normalizePath(), mustWork = TRUE).
signal	Optional character scalar naming the signal channel to project. When NULL (default) the channel is auto-selected (see *Signal selection*).

Value

An object of class `nirs4allformats_dataset`: a named list with

`x` Numeric matrix of spectra, `n_samples` x `n_wavelengths`.

`wavelengths` Numeric vector of axis coordinates (length `n_wavelengths`).

`targets` `data.frame` of reference values, one column per target key (zero columns when none were parsed).

`sample_ids` Character vector of per-row identifiers.

`metadata` List of per-record metadata lists.

`signal_type` Signal type of the selected channel.

`axis_unit` Unit string of the spectral axis (e.g. `"nm"`).

`formats` Character vector of the source format per row.

Use `as.matrix()` / `as.data.frame()` / `nirs4allformats_as_tibble()` to project it into common R shapes.

Signal selection

When `signal` is `NULL` the channel is chosen per record in this order:

1. the first signal whose `signal_type` equals the record-level `signal_type`;
2. otherwise the first present of `"reflectance"`, `"absorbance"`, `"transmittance"`, `"signal"`;
3. otherwise the alphabetically first signal name.

Passing an explicit `signal` name selects that channel and errors if a record lacks it.

Sample IDs

Each row's identifier is taken from `metadata$sample_id` when present; otherwise it is derived from the source file basename and 0-based row index (`"<basename>:<i>"`), falling back to `"record:<i>"` when no source path is known.

Targets and metadata

Reference values found under each record's `targets` are gathered into a `data.frame` (missing values become `NA`). The full per-record metadata lists are preserved verbatim in the `metadata` field of the returned object.

Transport

The call is served by the native extendr static library when it is present (built by `R CMD INSTALL` with Cargo on `PATH`). Otherwise it shells out to the `nirs4all-formats` CLI: the `NIRS4ALL_FORMATS_CLI` environment variable may point to a prebuilt binary (it is whitespace-split into command + arguments), a `nirs4all-formats` binary on `PATH` is used if found, and in a source checkout it falls back to `cargo run -p nirs4all-formats-cli`.

See Also

nirs4allformats_open_records() for the lossless record view, as.matrix.nirs4allformats_dataset(), as.data.frame.nirs4allformats_dataset(), nirs4allformats_as_tibble().

Examples

```
## Not run:
ds <- nirs4allformats_open_dataset("samples/csv_tsv/synthetic_nirs.csv")
dim(as.matrix(ds))
head(as.data.frame(ds))
ds$wavelengths[1:5]

# Select a specific channel by name
ds_abs <- nirs4allformats_open_dataset("spectrum.dx", signal = "absorbance")

## End(Not run)
```

---

nirs4allformats_open_records

Read a spectroscopy file into normalized records

Description

Decodes a single file through the Rust nirs4all-formats registry and returns the normalized records exactly as the core emits them, as nested R lists. Format detection is content-based: the file is sniffed and dispatched to the highest-confidence reader. No reshaping, alignment or column-building is done here – this is the faithful, lossless view of the Rust SpectralRecord model. For a flat spectral matrix use nirs4allformats_open_dataset().

Parser logic lives only in Rust; this function never parses bytes itself. It dispatches through the native extendr library when available and otherwise through the nirs4all-formats CLI (see *Transport* in nirs4allformats_open_dataset()).

Usage

```
nirs4allformats_open_records(path)
```

Arguments

path	Character scalar. Path to the input file. It is resolved with normalizePath() (mustWork = TRUE), so the file must exist.

Value

A list of records. Each record is a named list mirroring the Rust SpectralRecord:

signals Named list of signal channels. Each channel carries values (flat C-order buffer), shape, dims (exactly one is "x"), optional coords, signal_type, unit, role, source and an axis (values, unit, kind, order).

`signal_type` Record-level signal type (e.g. `"absorbance"`, `"reflectance"`, `"unknown"`).

`targets` Named list of reference values parsed from the file.

`metadata` Named list of typed metadata key/value pairs.

`provenance` Reader name/version, per-source SHA-256 (`sources`), `format`, `record_schema_version` and `warnings`.

`quality_flags` Character vector of quality annotations.

See Also

`nirs4allformats_open_dataset()` for a flat matrix view, `nirs4allformats_probe_path()` to inspect candidate readers, `nirs4allformats_walk_path()` to scan a directory.

Examples

```
## Not run:
records <- nirs4allformats_open_records("samples/csv_tsv/synthetic_nirs.csv")
length(records)
records[[1]]$provenance$format
names(records[[1]]$signals)

## End(Not run)
```

---

nirs4allformats_open_with_sidecars

Decode raw bytes with companion sidecar files

Description

Decodes an in-memory primary buffer together with a named map of companion files (sidecars) through the Rust registry, returning normalized records as nested R lists. This serves formats that split a measurement across multiple files, such as ENVI Standard cubes (`.img` + `.hdr`) or ERDAS LAN. Sidecar names are interpreted as paths relative to the primary file.

This path requires the native extendr static library and raises an error when it is absent; it has no CLI fallback.

Usage

```
nirs4allformats_open_with_sidecars(name, bytes, sidecars = list())
```

Arguments

`name`	Character scalar. Logical file name of the primary file (with extension), used for sniffing and provenance, e.g. `"cube.img"`.
`bytes`	A `raw` vector with the primary file contents.
`sidecars`	A named list of `raw` vectors. Each name is a companion file path relative to the primary file (e.g. `"cube.hdr"`); each value is that file's bytes. Defaults to an empty list.

Value

A list of records, identical in shape to `nirs4allformats_open_records()`.

See Also

`nirs4allformats_open_bytes()`, `nirs4allformats_open_records()`, `nirs4allformats_native_available()`.

Examples

```
## Not run:
read_raw <- function(p) readBin(p, "raw", n = file.info(p)$size)
records <- nirs4allformats_open_with_sidecars(
  "cube.img",
  read_raw("cube.img"),
  list("cube.hdr" = read_raw("cube.hdr"))
)

## End(Not run)
```

`nirs4allformats_probe_path` *List candidate readers for a file*

Description

Sniffs a file (reading only its head, not a full parse) and returns the ordered list of readers that recognize it, highest confidence first. Useful for diagnosing format detection without decoding the whole file.

Sniffing is performed entirely in Rust. The native extendr library is used when present; otherwise the `nirs4all-formats probe` CLI command is invoked (see *Transport* in `nirs4allformats_open_dataset()`).

Usage

```
nirs4allformats_probe_path(path)
```

Arguments

`path`	Character scalar. Path to the file to probe (resolved with `normalizePath()`, `mustWork = TRUE`).

Value

A list of candidate descriptors. Each entry includes at least a `format` name and a confidence indication, ordered from most to least confident. The list is empty when no reader recognizes the file.

See Also

nirs4allformats_open_records(), nirs4allformats_walk_path().

Examples

```
## Not run:
probes <- nirs4allformats_probe_path("samples/csv_tsv/synthetic_nirs.csv")
probes[[1]]$format

## End(Not run)
```

---

nirs4allformats_version

Report the nirs4allformats binding version

Description

Returns the version of the nirs4allformats R binding as a character scalar. This is the binding's own version and is independent of the underlying Rust nirs4all-formats core version.

Usage

```
nirs4allformats_version()
```

Value

A length-one character vector with the binding version.

See Also

nirs4allformats_native_available().

Examples

```
nirs4allformats_version()
```

`nirs4allformats_walk_path` *Walk a directory and report per-file outcomes*

Description

Recursively visits a directory (or a single file) and reports the detection outcome for each visited file: whether it parsed, errored, or is unsupported, together with its detected format. Only sniffing and walking happen here; no file is fully decoded.

The walk runs in Rust. The native extendr library is used when present; otherwise the `nirs4all-formats scan --json` CLI command is invoked and its `entries` are returned (see *Transport* in `nirs4allformats_open_dataset()`).

Usage

```
nirs4allformats_walk_path(
  path,
  max_depth = NULL,
  include_hidden = FALSE,
  follow_symlinks = FALSE,
  include_unsupported = FALSE
)
```

Arguments

`path`	Character scalar. Directory or file to scan (resolved with `normalizePath()`, `mustWork = TRUE`).
`max_depth`	Optional integer. Maximum recursion depth; `NULL` (default) means unlimited.
`include_hidden`	Logical. Include hidden files/directories. Defaults to `FALSE`.
`follow_symlinks`	Logical. Follow symbolic links during the walk. Defaults to `FALSE`.
`include_unsupported`	Logical. Include entries for files no reader recognizes. Defaults to `FALSE`.

Value

A list of per-file outcome entries. Each entry includes at least a `status` (e.g. `"parsed"`, `"error"`, `"unsupported"`) and, when detected, a `format`.

See Also

`nirs4allformats_probe_path()`, `nirs4allformats_open_records()`.

Examples

```
## Not run:
entries <- nirs4allformats_walk_path("samples/asd")
length(entries)
entries[[1]]$status
entries[[1]]$format

# Limit recursion depth and include unsupported files
nirs4allformats_walk_path("samples", max_depth = 1, include_unsupported = TRUE)

## End(Not run)
```

Index