Package: opusreader2 (via r-universe)

September 12, 2024

Title Read Spectroscopic Data from Bruker OPUS Binary Files

Version 0.6.3

Description Read data from OPUS binary files of Fourier-Transform infrared spectrometers of the company Bruker Optics GmbH & Co.

License MIT + file LICENSE

Encoding UTF-8

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.1

URL https://github.com/spectral-cockpit/opusreader2

BugReports https://github.com/spectral-cockpit/opusreader2/issues

Suggests knitr, rmarkdown, testthat (>= 3.0.0), future.apply, future, progressr

Config/testthat/edition 3

VignetteBuilder knitr

NeedsCompilation no

Repository https://spectral-cockpit.r-universe.dev

RemoteUrl https://github.com/spectral-cockpit/opusreader2

RemoteRef HEAD

RemoteSha e846540049149ca440f87c944251bd7c84421d0d

Contents

opus_file *Get location of a sample OPUS file*

Description

Utility function that retrieves the location of the sample OPUS binary file on disk.

Usage

```
opus_file()
```

Value

a character vector storing the location of the sample OPUS file

Examples

```
fn <- opus_file()
fn
```

read_opus *Read OPUS binary files produced by a Bruker spectrometer*

Description

This function can be used to read and parse OPUS files, to make it usable for other processing steps.

Usage

```
read_opus(dsn, data_only = FALSE, parallel = FALSE, progress_bar = FALSE)
```

Arguments

dsn	data source name. Can be a path to a specific file or a path to a directory. The listing of the files in a directory is recursive.
data_only	read data and parameters with FALSE per default, or only read data NULL, which only returns the parsed data as an in-memory R object.
parallel	read files in parallel via chunking. Default is FALSE.
progress_bar	print a progress bar. Default is FALSE.

Value

Nested list (S3 object) containing the parsed contents of the binary encoded blocks of an OPUS file. The first level names of the list correspond to the display names as shown in the Bruker OPUS viewer software. However, in snake_case and more standardized naming to allow for better output handling. Each parsed block element is a sublist containing **a)** the binary read instructions decoded/derived from the header (`$block_type`, `$channel_type`, `$text_type` and `$additional_type`, `$offset` (bytes), `$next_offset` (bytes), `$chunk_size` (bytes)); **b)** if parameter block, nested list of specific parameters under `$parameters`, which has elements named according to capitalized Bruker-internal "three-letter-string" definitions (e.g., "DPF := Data Point Format"). Possible first-level block names and information provided include:

- `refl_no_atm_comp_data_param` : class "parameter" (viewer: "Data Parameters Refl". Parameter list with metadata for `refl` data block (`refl`).
- `refl_no_atm_comp`: class "data" (spectrum; viewer: "Refl"). Unprocessed (raw; i.e, not atmospherically compensated) reflectance spectra (`:= sc_sample / sc_ref`). Note that this element is the untreated spectra before an eventual "atmospheric compensation" routine is applied.
- `refl_data_param` : class "parameter" (viewer: "Data Parameters Refl"). Parameter list with metadata for `refl` data block (metadata of reflectance spectrum; see `refl` output). Note that this element only results if "atmospheric compensation" was activated in the OPUS measurement settings.
- `refl`: class "data" (spectrum; viewer: "Refl"). Atmospherically compensated reflectance spectra (`:= sc_sample_corr / sc_ref_corr`). This result spectrum only exists if either correction of CO2 and/or water vapour bands is set in OPUS setting (proprietary algorithm; could possibly be reverse engineered). If `refl` exists, it has always a corresponding untreated `refl_no_atm_comp` spectrum (the latter present in file but not shown in the OPUS viewer, where only (final) `ab` is displayed)
- `quant_report_refl`: class "parameter" (viewer: "Quant Report Refl"). Quantification report for tools of multivariate calibration on `refl` data (i.e., PLS regression) offered in the QUANT2 OPUS package. Nested list with Bruker-internal "three-letter-string" definitions. "TIT" is the title of a nested quantification table, `"E<digit>[2]"` stands probably for entry, `"F<digit>[2]"` for field, and `"Z<digit>[2]"` we do not yet know what it maps to. There seems more information needed, which we can get by expanding the header parsing algorithm.
- `ab_no_atm_comp_data_param` : class "parameter" (viewer: "Data Parameters AB"). Parameter list with metadata for `ab` data block (spectrum; see `ab` output).
- `ab_no_atm_comp`: class "data" (spectrum; viewer: "Refl"). Unprocessed (raw; i.e, not atmospherically compensated) reflectance spectra (`:= sc_sample/ sc_ref`).
- `ab_data_param` : class "parameter" (viewer: "Data Parameters Refl"). Parameter list with metadata for `ab` data block (spectrum; see `ab`). Note that this element only results if "atmospheric compensation" was activated in the OPUS measurement settings.
- `ab`: class "data" (spectrum; viewer: "AB"). Atmospherically compensated (apparent) absorbance spectra (`:= log10(1 / (sc_sample_corr / sc_ref_corr))`). Only exists if either correction of CO2 and/or water vapour bands is set in OPUS setting (proprietary algorithm; could possibly be reverse engineered). If `AB` exists, it has always a corresponding untreated `ab_no_atm_comp` spectrum (the latter present in file but not shown in the OPUS viewer, where only final `ab` is displayed).

- quant_report_ab: class "parameter" (viewer: "Quant Report AB"). Quantification report for tools of multivariate calibration on ab data (i.e., PLS regression) offered in the QUANT2 OPUS package. Nested list with Bruker-internal "three-letter-string" definitions. "TIT" is the title of a nested quantification table, "E<digit>[2]" stands probably for entry, "F<digit>[2]" for field, and "Z<digit>[2]" we do not yet know what it maps to. There seems more information needed, which we can get by expanding the header parsing algorithm.
- sc_sample_data_param: class "parameter" (metadata; viewer: "Data Parameters ScSm"). Describes the sc_sample data block (see sc_sample).
- sc_sample: class "data" (spectrum). Single channel (sc) spectrum of the sample (y-axis: intensity).
- ig_sample_data_param: class "parameter" (metadata; viewer: "Data Parameters IgSm").
- ig_sample: class "data" (signal, viewer: "IgSm"). Interferogram of the sample measurement. Oscillatory signal (x-axis: optical path difference (OPD); y-axis: amplitude of the signal).
- sc_ref_data_param: class "parameter" (metadata; viewer: "Data Parameters ScRf"). Describes the sc_sample data block (see sc_ref).
- sc_ref: class "data" (spectrum; viewer: "Data Parameters IgSm"). Single channel (sc) spectrum of the reference (background material: e.g., gold; y-axis: intensity).
- ig_ref_data_param: class "parameter" (metadata; viewer: "Data Parameters IgRf").
- ig_ref: class "data" (spectrum; viewer: "IgRf"). Interferogram of the reference measurement. (background material: e.g., gold). Oscillatory signal (x-axis: optical path difference (OPD); y-axis: amplitude of the signal)
- optics: class "parameter (metadata; viewer: "Optic Parameters"). Optic setup and settings such as "Accessory", "Detector Setting" or "Source Setting".
- optics_ref: class "parameter (metadata; viewer: "Optic Parameters Rf"). Optic setup and settings specific to reference measurement such as "Accessory", "Detector Setting" or "Source Setting".
- acquisition_ref: class "parameter" (metadata; viewer: "Acquisition parameters Rf". Settings such as ""Additional Data Treatment", (number) of "Background Scans" or "Result Spectrum" (e.g. value "Absorbance").
- fourier_transformation_ref:
- fourier_transformation: class "parameter"
- sample:
- acquisition:
- instrument_ref:
- instrument:
- lab_and_process_param_1:
- lab_and_process_param_2:
- info_block:
- history:
- unknown: if a block-type can not be matched, no parsing is done and an empty list entry is returned. This gives you a hint that there is a block that can not yet be parsed. You can take further steps by opening an issue.

Details

`read_opus()` is the high-level interface to read multiple OPUS files at once from a data source name (`dsn`). It optionally supports parallel reads via the future framework. When reading in parallel, a progress bar can be enabled, which uses progressr under the hood for progress updates. If `parallel = TRUE`, one can specify across how many chunks the OPUS files are distributed onto the registered parallel workers. This can be done via `options(number_of_chunks = <integer>)`. The default value is `number_of_chunks = "registered workers"`, which will split the OPUS files across number of chunks corresponding to the number of registered workers.

`read_opus_single` *Read a single opus file*

Description

Read a single opus file

Usage

```
read_opus_single(dsn, data_only = FALSE)
```

Arguments

`dsn`	source path of an opus file
`data_only`	read data and parameters with `FALSE` per default, or only read data

Index