

Package: chromConverter (via r-universe)

October 29, 2026

Title Chromatographic File Converter

Version 0.11.2.9000

Maintainer Ethan Bass <ethanbass@gmail.com>

Description Reads chromatograms from binary formats into R objects.

Currently supports conversion of 'Agilent ChemStation', 'Agilent MassHunter', 'Agilent OpenLab', 'Shimadzu LabSolutions', 'ThermoRaw', 'Varian Workstation', and 'Waters Empower' files as well as various other formats. In addition to its internal parsers, chromConverter contains bindings to parsers in external libraries, such as 'Aston' <<https://github.com/bovee/aston>>, 'Entab' <<https://github.com/bovee/entab>>, 'rainbow' <<https://rainbow-api.readthedocs.io/>>, and 'ThermoRawFileParser' <<https://github.com/compomics/ThermoRawFileParser>>. The suggested packages 'entab' and 'chromConverterExtraTests' are not on CRAN: both are available from the repository given in 'Additional_repositories', and the latest test fixtures can also be installed from <<https://github.com/ethanbass/chromConverterExtraTests>>.

License GPL (>= 3)

URL <https://ethanbass.github.io/chromConverter/>,
<https://github.com/ethanbass/chromConverter/>

BugReports <https://github.com/ethanbass/chromConverter/issues/>

Depends R (>= 4.1.0)

Imports bitops, fs, purrr, readxl, reticulate (>= 1.41.0), stringr, tidy, utils, RaMS, tibble, xml2, bit64, data.table, base64enc, jsonlite, digest

Suggests entab, ncd4, pbapply, testthat (>= 3.1.7), mZR, chromConverterExtraTests, withr, knitr, rmarkdown, zip

VignetteBuilder knitr

Encoding UTF-8

Language en-US

Roxygen list(markdown = TRUE)

Additional_repositories <https://ethanbass.github.io/drat/>

Config/testthat/edition 3

Config/Needs/website rmarkdown, ggplot2, dplyr

Config/roxygen2/version 8.1.1

Config/pak/sysreqs cmake make libicu-dev libpng-dev libuv1-dev libxml2-dev python3

Repository <https://ethanbass.r-universe.dev>

Date/Publication 2026-10-29 01:45:45 UTC

RemoteUrl <https://github.com/ethanbass/chromConverter>

RemoteRef HEAD

RemoteSha cccce1b7edcd17bc9d854330c670beccc86f8199

Contents

add_metadata	3
alkane_ladder	4
benzoxazinoid_standards	5
call_entab	5
call_openchrom	7
call_rainbow	9
configure_openchrom	11
extract_metadata	12
get_element	13
print.chrom_list	14
read_acaml	15
read_agilent_amx	16
read_agilent_d	18
read_agilent_dx	20
read_agilent_rslt	22
read_asm	24
read_cdf	25
read_chemstation_ch	27
read_chemstation_csv	28
read_chemstation_logs	29
read_chemstation_method	31
read_chemstation_ms	33
read_chemstation_reports	34
read_chemstation_uv	36
read_chromatotec	37
read_chromeleon	38
read_chroms	39
read_mdf	42
read_metadata	43

read_mzml	46
read_peaklist	47
read_shimadzu	49
read_shimadzu_gcd	50
read_shimadzu_lcd	52
read_shimadzu_method	56
read_shimadzu_qgd	59
read_thermoraw	61
read_varian_peaklist	62
read_varian_sms	63
read_waters_arw	65
read_waters_raw	66
subset.chrom_list	67
summary.chrom_list	68
write_agilent_amx	69
write_andi_chrom	71
write_andi_ms	72
write_chroms	73
write_mzml	75
Index	78

add_metadata	<i>Add metadata to a list of chromatograms</i>
--------------	--

Description

Attaches the columns of a table to the chromatograms of a `chrom_list` as metadata fields, matching each row to a sample by name. The new fields can then be used by `subset.chrom_list` and requested from `extract_metadata`. Where an element holds several chromatograms, such as the traces `read_agilent_d` returns for each .D directory, every one of them gets the sample's values.

Usage

```
add_metadata(chrom_list, metadata, by = "name", overwrite = FALSE)
```

Arguments

<code>chrom_list</code>	A <code>chrom_list</code> object.
<code>metadata</code>	A <code>data.frame</code> , <code>tibble</code> or <code>data.table</code> with one row per sample.
<code>by</code>	The column of <code>metadata</code> holding the sample names, matched to <code>names(chrom_list)</code> . Defaults to <code>name</code> , the column <code>extract_metadata</code> identifies samples by.
<code>overwrite</code>	Whether a column may replace a field that <code>chromConverter</code> reads from the file, such as <code>sample_name</code> . Defaults to <code>FALSE</code> , in which case such a column is an error.

Value

chrom_list with the columns of metadata attached to its chromatograms, and their names recorded in an added_metadata attribute, from which chromatographR's get_peaktable fills its sample_meta. A sample without a row in metadata is left unchanged, with a warning.

See Also

[extract_metadata](#), [subset.chrom_list](#)

Examples

```
path <- system.file("extdata", "benzoxazinoid_standards",
  package = "chromConverter")
chroms <- read_chroms(path, format_in = "chemstation_ch", pattern = "dad1A",
  parser = "chromconverter", progress_bar = FALSE)
meta <- data.frame(name = c("MEOH", "BENZOS_1000PPM", "BENZOS_500PPM",
  "BENZOS_250PPM", "BENZOS_125PPM",
  "BENZOS_62,5PPM", "BENZOS_31,25PPM",
  "BENZOS_16PPM", "BENZOS_8PPM", "BENZOS_4PPM"),
  ppm = c(0, 1000 / 2^(0:8)))
chroms <- add_metadata(chroms, meta)
extract_metadata(chroms, what = "ppm")
names(subset(chroms, ppm >= 250))
```

 alkane_ladder

Shimadzu GC-FID alkane ladder

Description

An ASCII export from 'Shimadzu LabSolutions' of a GC-FID run (GC-2014) of an n-alkane ladder with nonadecane (C19) added, sampled by solid-phase microextraction, holding its peak table of 83 peaks and the chromatogram, recorded every 40 ms for 44 minutes. Find it with `system.file("extdata", "alkane_ladder.txt", package = "chromConverter")`.

License

Released under CC0 1.0 (<https://creativecommons.org/publicdomain/zero/1.0/>).

Source

Andrew W. Legan (<https://orcid.org/0000-0001-7049-9837>). The binary file of the same run is FS19_214.gcd in 'chromConverterExtraTests'.

`benzoxazinoid_standards`*Benzoxazinoid standards*

Description

A calibration series of a mixed benzoxazinoid standard, run by HPLC with an 'Agilent' diode array detector (G1315A), as ten 'ChemStation' .D directories in `extdata`. Find them with `system.file("extdata", "benzoxazinoid_standards", package = "chromConverter")`.

Details

- BENZOS_1000PPM.D to BENZOS_4PPM.D: nine twofold dilutions from 1000 ppm, injected between 20 and 22 June 2023. The names round the last three, which are 15.625, 7.8125 and 3.90625 ppm.
- ME0H.D: a methanol blank, run with the same method on 15 June 2023.

Each directory holds the 254 nm trace (`dad1A.ch`) and the 'ChemStation' report (`Report.TXT`). BENZOS_250PPM.D also holds the traces at 230, 320, 360 and 210 nm (`dad1B.ch` to `dad1E.ch`), and the pump's pressure, flow and solvent composition through the run (`LCDIAG.REG`), which `read_agilent_d` returns with `what = "instrument"`.

The standard gives four peaks at 254 nm, eluting in the 250 ppm run at 12.5 (DIBOA), 19.1 (DIMBOA), 21.0 (BOA) and 28.2 minutes (MBOA).

License

These files are released under CC0 1.0 (<https://creativecommons.org/publicdomain/zero/1.0/>).

See Also

`vignette("chromConverter")`, which works through these files.

`call_entab`*Call Entab*

Description

Converts chromatography data files using `entab` parsers. 'entab' detects the file format itself. Mass spectra are always returned in long format.

Usage

```
call_entab(  
  path,  
  data_format = c("wide", "long"),  
  format_out = c("matrix", "data.frame", "data.table"),  
  format_in = "",  
  read_metadata = TRUE,  
  metadata_format = c("chromconverter", "raw")  
)
```

Arguments

path	Path to file.
data_format	Whether to return data in wide (default) or long format.
format_out	Class of output. Either matrix, data.frame, or data.table.
format_in	Format of the file, used only to read the metadata that 'entab' does not: those of 'ChemStation' formats (any value containing chemstation) and of masshunter_dad. Defaults to "".
read_metadata	Logical. Whether to attach metadata. Defaults to TRUE.
metadata_format	Format to output metadata. Either chromconverter (standardized field names) or raw (vendor field names, unmapped).

Value

A chromatogram in the format specified by the format_out and data_format arguments.

See Also

Other external parsers: [call_openchrom\(\)](#), [call_rainbow\(\)](#), [read_thermoraw\(\)](#), [sp_converter\(\)](#), [uv_converter\(\)](#)

Examples

```
## Not run:  
call_entab("path/to/file.uv", format_in = "chemstation_uv")  
  
## End(Not run)
```

call_openchrom	<i>Parse files with OpenChrom</i>
----------------	-----------------------------------

Description

Converts files with the OpenChrom command-line interface, then reads the converted files back into R. OpenChrom removed the command-line interface in version 1.5.0, and Lablicate no longer distributes older versions, so this function works only with an existing installation of OpenChrom 1.4 or earlier. It is deprecated for that reason.

Usage

```
call_openchrom(  
  files,  
  path_out = NULL,  
  format_in,  
  format_out = c("matrix", "data.frame", "data.table"),  
  data_format = c("wide", "long"),  
  export_format = c("mzml", "csv", "cdf", "animl"),  
  return_paths = FALSE,  
  verbose = getOption("verbose")  
)
```

Arguments

files	Path to files.
path_out	Directory to export converted files. Defaults to NULL, in which case the files go to a temporary directory that is deleted when the function returns.
format_in	Either msd for mass spectrometry data, csd for FID, ECD or NPD data, or wsd for DAD/UV data.
format_out	Class of output. Either matrix, data.frame, or data.table.
data_format	Whether to return data in wide (default) or long format.
export_format	Either mzml, csv, cdf, or animl. Defaults to mzml. mzml and cdf are available only for msd. OpenChrom versions before 1.4 can export only msd files, and only to csv, cdf or animl.
return_paths	Logical. If TRUE, the function will return a character vector of paths to the newly created files.
verbose	Logical. Whether to print output from OpenChrom to the console.

Details

The call_openchrom function works by creating an xml batchfile and feeding it to the OpenChrom command-line interface. OpenChrom batchfiles consist of InputEntries (specifying the files you want to convert) and ProcessEntries (specifying what you want to do to the files). The parsers are organized into broad categories by detector-type and output format. The detector-types are msd

(mass selective detectors), csd (current selective detectors, e.g., FID, ECD, NPD), and wsd (wavelength selective detectors, e.g., DAD, and UV/VIS). Thus, when calling the OpenChrom parsers, one of these three options must be specified using the `format_in` argument.

Value

If `return_paths` is FALSE, a list with one chromatogram per file, read back with [read_mzml](#), [read_cdf](#) or [read_csv](#) according to `export_format`. `format_out` and `data_format` apply only to cdf files, except that `format_out = "matrix"` also converts csv files. animl files cannot be read back, so use `return_paths = TRUE` for them. If `return_paths` is TRUE, a character vector of paths to the newly created files.

Side effects

Chromatograms are exported in the format specified by `export_format` to the folder specified by `path_out`.

Note

Activating the OpenChrom command-line deactivates the graphical user interface (GUI). To keep using the GUI, install a second copy of OpenChrom and call that one from R.

Author(s)

Ethan Bass

References

Wenig, Philip and Odermatt, Juergen. OpenChrom: A Cross-Platform Open Source Software for the Mass Spectrometric Analysis of Chromatographic Data. *BMC Bioinformatics* **11**, no. 1 (July 30, 2010): 405. doi:10.1186/1471210511405.

See Also

Other external parsers: [call_entab\(\)](#), [call_rainbow\(\)](#), [read_thermoraw\(\)](#), [sp_converter\(\)](#), [uv_converter\(\)](#)

Examples

```
## Not run:
call_openchrom("path/to/file.RAW", format_in = "msd", export_format = "mzml")

## End(Not run)
```

call_rainbow	<i>Call 'rainbow' parsers</i>
--------------	-------------------------------

Description

Parse 'Agilent' or 'Waters' files with rainbow parsers.

Usage

```
call_rainbow(
  path,
  format_in = c("agilent_d", "waters_raw", "masshunter", "chemstation", "chemstation_uv",
    "chemstation_fid", "chemstation_ms"),
  format_out = c("matrix", "data.frame", "data.table"),
  data_format = c("wide", "long"),
  by = c("detector", "name"),
  what = NULL,
  read_metadata = TRUE,
  metadata_format = c("chromconverter", "raw"),
  collapse = TRUE,
  precision = 1,
  sparse = TRUE,
  bin_width = NULL,
  centroid = FALSE,
  hrms = FALSE
)
```

Arguments

path	Path to file.
format_in	Format of the supplied files. Either agilent_d, waters_raw, masshunter, chemstation, chemstation_uv, chemstation_fid, or chemstation_ms.
format_out	Class of output. Either matrix, data.frame, or data.table.
data_format	Whether to return data in wide (default) or long format.
by	How to group the returned list: by detector (default), or by name, one element per data file in the directory.
what	Which detectors to return (e.g. MS, UV, CAD, ELSD). Applies only when by = "detector". Defaults to NULL, which returns all of them.
read_metadata	Logical. Whether to attach metadata. Defaults to TRUE.
metadata_format	Format to output metadata. Either chromconverter (standardized field names) or raw (vendor field names, unmapped).
collapse	Logical. Whether to collapse lists that only contain a single element. Defaults to TRUE.

precision	Number of decimals of the m/z grid, which is spaced $10^{\text{precision}}$ apart. Defaults to 1. Ignored if bin_width is supplied.
sparse	Logical. Whether to drop zero intensities from MS data. Applies only to long format. Defaults to TRUE.
bin_width	Width of the m/z grid, in daltons. An alternative to precision for grids that are not a power of ten (e.g. 0.5). Defaults to NULL, in which case the grid is derived from precision as $10^{\text{precision}}$.
centroid, hrms	Logical. Whether to read the centroided (MSPeak.bin) and/or profile (MSProfile.bin) mass spectra of an 'Agilent MassHunter' .d directory. Both default to FALSE, as in 'rainbow', so no mass spectra are read unless one is TRUE. Like other mass spectra, they are binned onto the m/z grid set by precision or bin_width, which coarsens high-resolution data. Ignored for the chemstation formats.

Details

Uses **rainbow** parsers to read in Agilent (.D) and Waters (.raw) files. For agilent_d, waters_raw and masshunter, path is the data directory (.D or .raw), and the result is a list grouped according to by. For the chemstation formats, path is a single file (e.g. .uv), the result is a single chromatogram, and by and what are ignored. Otherwise, data can be filtered by detector type using the what argument.

Value

A (nested) list of chromatograms, or a single chromatogram for the chemstation formats, in the class given by format_out. The list is grouped according to the value of by.

Author(s)

Ethan Bass

See Also

Other external parsers: [call_entab\(\)](#), [call_openchrom\(\)](#), [read_thermoraw\(\)](#), [sp_converter\(\)](#), [uv_converter\(\)](#)

Examples

```
## Not run:
call_rainbow("path/to/file.D", format_in = "agilent_d")

## End(Not run)
```

configure_openchrom	<i>Configure 'OpenChrom' parser</i>
---------------------	-------------------------------------

Description

Configures **OpenChrom** to use command-line interface. Requires OpenChrom version prior to 1.5.0.

Usage

```
configure_openchrom(cli = c("null", "true", "false", "status"), path = NULL)
```

Arguments

cli	One of "null" (default), "true", "false" or "status". "true" and "false" rewrite the OpenChrom ini file to enable or disable the command-line interface. "null" leaves the ini file alone unless the interface is disabled, in which case it asks whether to enable it. "status" reports the current setting without changing anything.
path	Path to the 'OpenChrom' executable (optional). The supplied path is saved and used in later calls.

Value

If cli is "status", the string "true" or "false", saying whether the command-line interface is enabled. Otherwise, the path to the OpenChrom command-line application.

Author(s)

Ethan Bass

See Also

[call_openchrom](#)

Examples

```
## Not run:  
configure_openchrom(cli = "status")  
  
## End(Not run)
```

extract_metadata	<i>Extract metadata</i>
------------------	-------------------------

Description

Extract metadata as a `data.frame`, `data.table` or `tibble` from a list of chromatograms.

Usage

```
extract_metadata(
  chrom_list,
  what = chrom_metadata_fields(),
  detector = NULL,
  format_out = c("data.frame", "data.table", "tibble"),
  collapse = FALSE,
  expand = FALSE,
  by = c("sample", "chromatogram"),
  digits = NULL
)
```

Arguments

<code>chrom_list</code>	A list of chromatograms with attached metadata (as returned by <code>read_chroms</code> with <code>read_metadata = TRUE</code>), or a single chromatogram.
<code>what</code>	A character vector specifying the metadata elements to extract. Defaults to every field <code>chromConverter</code> attaches; no format records all of them, so the elements a format does not provide are absent from the result. Superseded names (e.g. <code>injection_volume</code> , <code>software_name</code> , <code>time_start</code>) are accepted and mapped to the names that replaced them. An element of a nested field (see <code>expand</code>) may be named too, either by the column it is reported under (<code>SampleLabel</code>) or in full (<code>acam1_metadata.SampleLabel</code>), to report it without the rest of its field. A field requested by name that no chromatogram carries produces a warning.
<code>detector</code>	A character vector of detectors to include (e.g. <code>"UV"</code> or <code>c("UV", "MS")</code>), matched case-insensitively against each chromatogram's detector attribute. Defaults to <code>NULL</code> , in which case all chromatograms are included. Useful for lists containing more than one detector per sample. It is an error if no chromatogram matches.
<code>format_out</code>	Format of object. Either <code>data.frame</code> , <code>data.table</code> or <code>tibble</code> .
<code>collapse</code>	Logical. Whether to collapse a field holding more than one value (<code>time_range</code> , or the <code>product_mz</code> of an MRM event monitoring several transitions) into a single comma-separated string. Defaults to <code>FALSE</code> , in which case such a field is spread across numbered columns (<code>time_range1</code> , <code>time_range2</code>).
<code>expand</code>	Whether to include the nested metadata fields, whose value is itself a list or table rather than a single value per chromatogram: the <code>ms_params</code> and <code>method_params</code> instrument settings, the <code>acam1_metadata</code> injection record that <code>read_agilent_rslt</code> reads from the <code>.acam1</code> file, or the whole vendor list that <code>metadata_format</code>

	= "raw" passes through. Either TRUE, to include every nested field the chromatograms carry, a character vector naming the ones to include, or FALSE (the default) to include none. Each element becomes a column of its own, named for itself (SampleName) unless that name is already taken, in which case it carries the field it came from (ms_params.polarity, since polarity is a metadata field in its own right).
by	Whether to return one row per sample (the default), that is per element of chrom_list, or one row per chromatogram. The two differ only for nested lists, such as the traces read_agilent_d returns for each .D directory. With by = "sample", each field holds the value that a sample's chromatograms agree on, ignoring those that leave it empty, and is NA where they disagree; with the default what, fields that are then empty for every sample are left out. Fields that belong to each trace, such as detector or source_file, need by = "chromatogram".
digits	Number of significant digits for the numbers in a field collapsed into a string (see collapse), or NULL (the default) to keep them in full. Numeric fields spread across columns are returned as numbers either way.

Value

A data.frame, tibble, or data.table (according to the value of format_out), with one row per sample or chromatogram (see by) and the specified metadata elements as columns, or NA if none of the specified elements could be found. For a list, the first column, name, identifies each row: by the sample's name, or with by = "chromatogram" by the chromatogram's path through the list (e.g. blue.UV).

Examples

```
path <- system.file("extdata", "benzoxazinoid_standards",
                    package = "chromConverter")
chroms <- read_chroms(path, format_in = "chemstation_ch", pattern = "dad1A",
                     parser = "chromconverter", progress_bar = FALSE)
extract_metadata(chroms, what = c("sample_name", "run_datetime", "time_range"))
```

get_element

Get one element of each sample

Description

Extracts the same element from every sample of a chrom_list whose samples are themselves lists, such as the output of [read_chroms](#) with several what values (e.g. c("dad", "instrument")). The result has one chromatogram per sample, which packages that expect a list of chromatograms, such as chromatographR, accept as is.

Usage

```
get_element(x, what)
```

Arguments

<code>x</code>	A <code>chrom_list</code> whose samples are lists of elements.
<code>what</code>	The element to extract: a single name (e.g. "dad"), or a vector of names for a nested element (e.g. <code>c("instrument", "PMP1, Pressure")</code>). A nested element can also be written as one string split at the first colon, as in "instrument:PMP1, Pressure", unless that string is itself an element name.

Value

A list of the extracted chromatograms, with the names and class of `x`.

See Also

[read_chroms](#), [extract_metadata](#)

Examples

```
path <- system.file("extdata", "benzoxazinoid_standards", "BENZOS_250PPM.D",
                    package = "chromConverter")
run <- read_chroms(path, format_in = "agilent_d",
                  what = c("chroms", "instrument"), progress_bar = FALSE)
pressure <- get_element(run, "instrument:PMP1, Pressure")
head(pressure[[1]])
```

<code>print.chrom_list</code>	<i>Print a chrom_list object</i>
-------------------------------	----------------------------------

Description

Prints a summary of a `chrom_list` without displaying the underlying chromatographic data. Attributes that are constant across all chromatograms are collapsed into a single header line, while varying attributes are shown as a table truncated to the first `n` rows. When a sample holds more than one chromatogram, its traces are printed as a block headed by the sample, and any attribute they all share is shown in that block's header rather than repeated down it.

Usage

```
## S3 method for class 'chrom_list'
print(
  x,
  n = 10,
  cols = chrom_summary_cols(),
  digits = getOption("digits"),
  ...
)
```

Arguments

x	A chrom_list object.
n	Integer. Maximum number of chromatograms to show in the table. Defaults to 10.
cols	Character vector of attribute names to report. Defaults to: sample_name, run_datetime, method, detector, wavelength, detector_range, scan_type, polarity, precursor_mz, product_mz, mz_range.
digits	Number of significant digits for numeric metadata. Defaults to getOption("digits"), as for print.data.frame .
...	Additional arguments (currently ignored).

Value

Invisibly returns x.

See Also

[extract_metadata](#)

Examples

```
path <- system.file("extdata", "benzoxazinoid_standards",
                    package = "chromConverter")
chroms <- read_chroms(path, format_in = "chemstation_ch", pattern = "dad1A",
                     parser = "chromconverter", progress_bar = FALSE)
print(chroms)
```

read_acaml

Read 'Agilent' ACAML files

Description

Extracts injection metadata from 'Agilent Common Analytical Markup Language' (ACAML) files into a table.

Usage

```
read_acaml(
  path,
  find_files,
  format_out = c("data.frame", "data.table", "tibble"),
  progress_bar = FALSE,
  cl = 1
)
```

Arguments

path	Path(s) to ACAML files or to folders that contain the files.
find_files	Logical. Whether to treat the supplied paths as directories to search for files. Inferred if not supplied, by testing whether every path is a file.
format_out	Class of output. Either matrix, data.frame, or data.table.
progress_bar	Logical. Whether to show a progress bar. Defaults to FALSE, unlike read_chroms , because an ACAML file usually accompanies a sequence rather than arriving in bulk: read_agilent_rslt calls this function on the single .acaml file in a .rslt directory, where a progress bar over one element is just noise.
cl	Argument to pbapply specifying the number of parallel workers to use or a cluster object created by makeCluster (a set of parallel R worker processes). Defaults to 1.

Details

ACAML is an XML-based format used by Agilent OpenLab to store sequence and sample meta-data. This function extracts information from the InjectionMetaData nodes embedded in the InjectionMetaDataItems custom field files, which do not seem to be readily accessible through other means.

Value

A data.frame, data.table or tibble (according to the value of format_out) with one row per injection, and a SourceFile column naming the ACAML file it came from.

Examples

```
## Not run:
read_acaml(path)

## End(Not run)
```

read_agilent_amx	<i>Read Agilent AMX method file</i>
------------------	-------------------------------------

Description

Parses an Agilent .amx method archive, extracting instrument parameters from one or more of its driver sub-files.

Usage

```
read_agilent_amx(
  path,
  what = c("dad", "pump", "comp", "sampler"),
  path_out = NULL,
  format_out = c("data.frame", "tibble", "data.table"),
  gradient_format = c("wide", "long")
)
```

Arguments

<code>path</code>	Path to the .amx file.
<code>what</code>	One or more instrument modules to parse. Any combination of "dad", "pump", "comp", and "sampler". Defaults to all four.
<code>path_out</code>	Directory into which the archive is extracted. If NULL (default), a temporary directory is used and cleaned up on exit.
<code>format_out</code>	Class of the gradient, solvents, signals and temp_controls tables. Either "data.frame", "tibble" or "data.table".
<code>gradient_format</code>	Whether to return the gradient in "wide" (default) or "long" format.

Value

A named list with one element per parsed module, plus "metadata". A module the archive has no driver file for is left out with a warning, and it is an error if none of the requested modules has one. The comp module is returned as column and sampler as autosampler. See below for the structure of each.

`metadata` — a list with scalar elements:

`method_name` Original method name.

`version` Method version string.

`status` Approval state.

`created` Creation timestamp (POSIXct, UTC).

`created_by` Username of creator.

`modified` Last-modified timestamp (POSIXct, UTC).

`modified_by` Username of last modifier.

`pump` — a list with scalar elements `flow_mL_min`, `stop_time_min`, `post_time_min`, `pressure_low_bar`, `pressure_high_bar`, plus:

`solvents` A table of active solvent channels: `channel` (e.g. "A"), `percentage` and `solvent` (its name, or NA).

`gradient` A table of timetable entries. Wide format (default): `time_min` plus one `pct_<channel>` column per active channel, and a `flow_mL_min` column if the timetable changes the flow. Long format: `time_min`, `channel`, `percent`, where a flow channel holds the flow in mL/min.

dad — a list with scalar elements `peakwidth_code`, `slitwidth_nm`, `uv_lamp_required`, `vis_lamp_required`, `spectra_from_nm`, `spectra_to_nm`, `spectra_step_nm`, plus:

signals A table of active signals: `id` (e.g. "A"), `wavelength_nm`, `bandwidth_nm`, `reference_nm` and `reference_bandwidth_nm` (NA without a reference).

peakwidth_code is the stored code for the detector's peak width setting, which sets its data rate and response time; for example, 4 is "> 0.1 min". It is not decoded, since detector models offer different settings.

column — a list with scalar element `post_time_min`, plus:

temp_controls Two-row table (Left/Right): `side`, `temperature_C`, `not_ready_limit_C`, `equilibration_time_min`.

autosampler — a list with scalar elements: `thermostat_installed`, `draw_speed_uL_min`, `eject_speed_uL_min`, `wait_after_draw_min`, `injection_volume_uL`, `wash_time_s`.

Examples

```
## Not run:
read_agilent_amx(path)

## End(Not run)
```

<code>read_agilent_d</code>	<i>Read files from 'Agilent ChemStation' .D directories</i>
-----------------------------	---

Description

Reads the `.ch`, `.uv`, `Report.TXT` (or `Result.xml`) and `LCDIAG.REG` files in an 'Agilent' `.D` directory. Other files in the directory are ignored.

Usage

```
read_agilent_d(
  path,
  what = c("dad", "chroms", "peak_table", "MS1", "MS2", "TIC", "BPC"),
  format_out = c("matrix", "data.frame", "data.table"),
  data_format = c("wide", "long"),
  read_metadata = TRUE,
  metadata_format = c("chromconverter", "raw"),
  collapse = TRUE
)
```

Arguments

path	Path to 'Agilent' .D directory.
what	Whether to extract chromatograms (chroms), DAD data (dad), peak tables (peak_table) and/or instrument traces (instrument), such as pump pressure, flow, solvent composition and temperature, read from LCDIAG.REG, and/or mass spectra (MS1, MS2) with the total ion chromatogram (TIC) and base peak chromatogram (BPC). Accepts multiple arguments, and defaults to all of these but instrument. Types the directory does not contain are left out, and it is an error if it contains none of them. With the default, mass spectrometry data that cannot be read are also left out, with a warning, if the directory holds other data. The TIC and BPC are read from the scan records, so they are returned even when the spectra cannot be.
format_out	Class of output. Either matrix, data.frame, or data.table.
data_format	Whether to return data in wide (default) or long format.
read_metadata	Logical. Whether to attach metadata. Defaults to TRUE.
metadata_format	Format to output metadata. Either chromconverter (standardized field names) or raw (vendor field names, unmapped).
collapse	Logical. Whether to collapse lists that only contain a single element. Defaults to TRUE.

Details

Instrument traces are named from LCDIAG.REG, so the names differ between 'ChemStation' revisions and/or instruments (e.g. "PMP1, Pressure" and "PMP1, PMP1A, Pressure"). Parts of the file that cannot be read are skipped with a warning.

Mass spectra are read from the AcqData folder of a 'MassHunter' .d directory. Its MSScan.bin and MSScan.xsd describe every scan, as in [read_agilent_dx](#), and the spectra are in MSPeak.bin, or in MSPProfile.bin for profiles. Each profile starts with its first m/z and the step between points, as two doubles or two 32-bit floats, followed by the intensities as 32-bit floats or integers respectively. The points are spaced by the scan's SamplingPeriod rather than the stored step, which can be slightly smaller. Spectra can be read only from unit-resolution instruments, such as single quadrupole GC-MS and LC-MS, and triple quadrupole LC-MS. MS1 holds the scans at MS level 1 and MS2 the product ion and MRM scans, each with the m/z it selected in the first quadrupole as precursor_mz. Neutral loss and precursor ion scans are skipped with a warning. The TIC and BPC come from the record of every scan, so they are also returned for runs whose spectra cannot be read. For high-resolution spectra, try parser = "rainbow" with centroid or hrms.

Value

A list with one element per type found, each a chromatogram or a list of them named by file, in the format specified by format_out and data_format. If data_format is wide, the chromatograms will be returned with retention times as rows and columns containing signal intensity for each signal. If long format is requested, retention times will be in the first column. The format_out argument determines whether the chromatogram is returned as a matrix, data.frame or data.table. Peak tables are returned in the chromatographr layout described in [read_peaklist](#), named by wavelength (e.g. "254"), and keep the report's name for the signal as signal_descriptor. They are read from

Result.xml where the directory has one, since unlike Report.TXT it records where each peak starts and ends. Mass spectra and the BPC are always in long format, with rt, mz and intensity columns, and a precursor_mz column before mz for MS2. They are returned as a data.table when format_out is matrix. Each table of spectra has a scan_info attribute with one row per scan, giving its scan, rt, ms_level, polarity and scan_type ("scan", "SIM", "MRM" or "product ion scan", and NA for any other type), and for MS2 its precursor_mz and collision_energy. Metadata are attached as [attributes](#) if read_metadata is TRUE. With collapse = TRUE, a list of one element is replaced by that element.

Author(s)

Ethan Bass

See Also

Other 'Agilent' parsers: [read_agilent_dx\(\)](#), [read_agilent_rslt\(\)](#), [read_chemstation_ch\(\)](#), [read_chemstation_csv\(\)](#), [read_chemstation_logs\(\)](#), [read_chemstation_method\(\)](#), [read_chemstation_ms\(\)](#), [read_chemstation_reports\(\)](#), [read_chemstation_uv\(\)](#)

Examples

```
path <- system.file("extdata", "benzoxazinoid_standards", "BENZOS_250PPM.D",
                    package = "chromConverter")
run <- read_agilent_d(path)
names(run$chroms)
pump <- read_agilent_d(path, what = "instrument")
names(pump)
```

read_agilent_dx	<i>Read 'Agilent' DX files</i>
-----------------	--------------------------------

Description

Reads 'Agilent' .dx files.

Usage

```
read_agilent_dx(
  path,
  what = c("chroms", "dad"),
  path_out = NULL,
  format_out = c("matrix", "data.frame", "data.table"),
  data_format = c("wide", "long"),
  read_metadata = TRUE,
  metadata_format = c("chromconverter", "raw"),
  collapse = TRUE
)
```

Arguments

path	Path to Agilent .dx file.
what	Whether to extract chromatograms (chroms), DAD data (dad), auxiliary instrumental data (instrument), such as temperature, pressure or solvent composition, and/or mass spectra (MS1) with the total ion chromatogram (TIC) and base peak chromatogram (BPC). Accepts multiple arguments, and defaults to chroms and dad. When more than one is requested, any the archive does not contain are left out; a single one that is missing is an error.
path_out	A directory to export unzipped files. If a path is not specified, a temporary directory is used. The files are extracted into a folder named for path, overwriting any files of the same name already there.
format_out	Class of output. Either matrix, data.frame, or data.table.
data_format	Whether to return data in wide (default) or long format.
read_metadata	Logical. Whether to attach metadata. Defaults to TRUE.
metadata_format	Format to output metadata. Either chromconverter (standardized field names) or raw (vendor field names, unmapped).
collapse	Logical. Whether to collapse lists that only contain a single element. Defaults to TRUE.

Details

The archive is extracted to path_out, or to a temporary directory that is deleted afterwards, and its .ch, .uv and .it files are read with [read_chemstation_ch](#), [read_chemstation_uv](#) and an internal reader for .it files.

The mass spectra of an MS injection are not in the archive. They are in the .MSScan.bin and .MSPeak.bin files of the same name beside it. The first is needed for MS1, TIC and BPC, the second for MS1. MSScan.bin holds one record per scan, laid out as the archive's MSData/MSScan.xsd describes, which gives the retention time, TIC and base peak of the scan and the location of its spectrum in MSPeak.bin. Each spectrum there is the scan's m/z values followed by as many intensities, as 32- or 64-bit floats or 32-bit integers depending on the scan's spectrum format and size. Only scans at MS level 1 are read into MS1, while the TIC and BPC cover every scan.

Where the archive holds an injection.acmd file, the run_datetime and sample_injection_volume attributes are taken from it. Its run time records the offset from UTC, while the .ch and .uv headers give only the local time.

Value

A list with one element per type in what, each a chromatogram or a list of chromatograms named by signal, in the format specified by format_out and data_format. Mass spectra and the BPC are always in long format, with rt, mz and intensity columns, and are returned as a data.table when format_out is matrix. With collapse = TRUE, a list of one element is replaced by that element.

Author(s)

Ethan Bass

See Also

Other 'Agilent' parsers: `read_agilent_d()`, `read_agilent_rslt()`, `read_chemstation_ch()`, `read_chemstation_csv()`, `read_chemstation_logs()`, `read_chemstation_method()`, `read_chemstation_ms()`, `read_chemstation_reports()`, `read_chemstation_uv()`

Examples

```
## Not run:
read_agilent_dx(path)

## End(Not run)
```

<code>read_agilent_rslt</code>	<i>Read files from 'Agilent' .rslt directories</i>
--------------------------------	--

Description

Reads a sequence of injections from an 'Agilent' .rslt directory. Combines raw chromatogram data from .dx files with injection-level metadata parsed from the accompanying .acaml file.

Usage

```
read_agilent_rslt(
  path,
  what = c("chroms", "dad"),
  path_out = NULL,
  format_out = c("matrix", "data.frame", "data.table"),
  data_format = c("wide", "long"),
  read_metadata = TRUE,
  metadata_format = c("chromconverter", "raw"),
  collapse = TRUE,
  sample_names = c("basename", "sample_name"),
  progress_bar = FALSE,
  cl = 1
)
```

Arguments

<code>path</code>	Path to 'Agilent' .rslt directory.
<code>what</code>	Whether to extract chromatograms (chroms), DAD data (dad), auxiliary instrumental data (instrument), such as temperature, pressure or solvent composition, and/or mass spectra (MS1) with the total ion chromatogram (TIC) and base peak chromatogram (BPC). Accepts multiple arguments, and defaults to chroms and dad. When more than one is requested, any the archive does not contain are left out; a single one that is missing is an error.

path_out	A directory to export unzipped files. If a path is not specified, a temporary directory is used. The files are extracted into a folder named for path, overwriting any files of the same name already there.
format_out	Class of output. Either <code>matrix</code> , <code>data.frame</code> , or <code>data.table</code> .
data_format	Whether to return data in wide (default) or long format.
read_metadata	Logical. Whether to attach metadata. Defaults to <code>TRUE</code> .
metadata_format	Format to output metadata. Either <code>chromconverter</code> (standardized field names) or <code>raw</code> (vendor field names, unmapped).
collapse	Logical. Whether to collapse lists that only contain a single element. Defaults to <code>TRUE</code> .
sample_names	How to name the chromatograms that are returned. Either <code>basename</code> (default), to use the name of the source <code>.dx</code> file, or <code>sample_name</code> , to use the sample name field from the metadata.
progress_bar	Logical. Whether to show a progress bar while reading the sequence's <code>.dx</code> files. Defaults to <code>FALSE</code> , since read_chroms already reports progress over the directories it found and a second bar inside each one would be redrawn per directory. Set to <code>TRUE</code> when calling this function directly on a sequence with many injections.
cl	Argument to pbapply specifying the number of clusters to use or a cluster object created by makeCluster . Defaults to 1.

Details

Currently this function only reads `.dx` chromatogram files. To read the peak tables stored in `.rx` files, use [read_peaklist](#) with `format_in = "openlab"`.

Value

A list of chromatograms (one `read_agilent_dx`-style result per injection in the sequence), in the format specified by `data_format` and `format_out`. If `read_metadata` is `TRUE`, injection-level metadata parsed from the `.acaml` file is attached to each chromatogram as attributes.

Author(s)

Ethan Bass

See Also

Other 'Agilent' parsers: [read_agilent_d\(\)](#), [read_agilent_dx\(\)](#), [read_chemstation_ch\(\)](#), [read_chemstation_csv\(\)](#), [read_chemstation_logs\(\)](#), [read_chemstation_method\(\)](#), [read_chemstation_ms\(\)](#), [read_chemstation_reports\(\)](#), [read_chemstation_uv\(\)](#)

Examples

```
## Not run:
read_agilent_rslt("path/to/sequence.rslt")

## End(Not run)
```

read_asm	<i>Read 'Allotrope Simple Model' (ASM) files</i>
----------	--

Description

Reads chromatograms, mass spectra and peak lists from '**Allotrope Simple Model**' chromatography files into R.

Usage

```
read_asm(
  path,
  what = c("chroms", "MS1"),
  data_format = c("wide", "long"),
  format_out = c("matrix", "data.frame", "data.table"),
  peaktable_format = c("chromatographr", "original"),
  read_metadata = TRUE,
  metadata_format = c("chromconverter", "raw"),
  collapse = TRUE
)
```

Arguments

path	Path to ASM .json file.
what	What to read: 2D chromatograms (chroms), mass spectra (MS1), peak lists (peak_table) and/or instrument traces (instrument), such as pump pressure, flow rate or temperature. Defaults to chroms and MS1, dropping whichever the file does not contain.
data_format	Whether to return data in wide (default) or long format.
format_out	Class of output. Either matrix, data.frame, or data.table.
peaktable_format	Whether to return peak tables in the standard chromatographr layout (rt, start, end, area, height and FWHM) or original format, with every field the file records under its ASM name.
read_metadata	Logical. Whether to attach metadata. Defaults to TRUE.
metadata_format	Format to output metadata. Either chromconverter (standardized field names) or raw (vendor field names, unmapped).
collapse	Logical. Whether to collapse lists that only contain a single element. Defaults to TRUE.

Details

Retention times in seconds are converted to minutes. Other units, such as retention volumes (mL), are returned as the file declares them. The unit is recorded in the `time_unit` attribute.

A file can hold several injections, such as a sequence exported from a chromatography data system. Each injection is then returned as a separate sample, named by its sample name, and [read_chroms](#) adds each one to the list of samples it returns.

Mass spectra have so far only been tested against the example files published by the Allotrope Foundation.

Value

A 2D chromatogram in the format specified by `format_out` and `data_format`, or a list of them named by detection type if the file holds more than one (or `collapse = FALSE`). Mass spectra are returned as a `data.frame` (or `data.table`) with columns `rt`, `mz` and `intensity`, and peak tables as a `data.frame` (or `data.table`) per measurement. Instrument traces are returned as 2D chromatograms named by trace. Where more than one of these is returned, they are combined in a list named by `what`. A file with several injections returns a `chrom_list` with one such element per injection. Metadata are attached as [attributes](#) if `read_metadata` is `TRUE`.

Author(s)

Ethan Bass

Examples

```
## Not run:
read_asm("path/to/file.json")
read_asm("path/to/file.json", what = "peak_table")

## End(Not run)
```

read_cdf

Read CDF

Description

Reads 'Analytical Data Interchange' (ANDI) netCDF (.cdf) files.

Usage

```
read_cdf(
  path,
  format_out = c("matrix", "data.frame", "data.table"),
  data_format = c("wide", "long"),
  what = NULL,
  read_metadata = TRUE,
  metadata_format = c("chromconverter", "raw"),
```

```

collapse = TRUE,
peaktable_format = c("chromatographr", "original"),
...
)

```

Arguments

<code>path</code>	Path to ANDI netCDF file.
<code>format_out</code>	Class of output. Either <code>matrix</code> , <code>data.frame</code> , or <code>data.table</code> .
<code>data_format</code>	Whether to return data in wide or long format. "long" returns the retention time as the first column of the <code>data.frame</code> or <code>matrix</code> , while "wide" returns it as the rownames. The mass spectra of an ANDI MS file are always long, so there it applies only to the TIC.
<code>what</code>	For ANDI chrom files, <code>chroms</code> and/or <code>peak_table</code> . For ANDI MS files, <code>MS1</code> scans (<code>MS1</code>) and/or the total ion chromatogram (<code>TIC</code>).
<code>read_metadata</code>	Logical. Whether to attach metadata. Defaults to <code>TRUE</code> .
<code>metadata_format</code>	Format to output metadata. Either <code>chromconverter</code> (standardized field names) or <code>raw</code> (vendor field names, unmapped).
<code>collapse</code>	Logical. Whether to collapse lists that only contain a single element. Defaults to <code>TRUE</code> .
<code>peaktable_format</code>	Whether to return peak tables in the standard <code>chromatographr</code> layout (the default) or with all the columns the file provides (<code>original</code>). A <code>chromatographr</code> table has the columns <code>rt</code> , <code>start</code> , <code>end</code> , <code>area</code> , <code>height</code> and <code>FWHM</code> (the full width at half maximum), with times in the same unit as the format's chromatograms and <code>NA</code> where the file does not record a value.
<code>...</code>	Additional arguments to the parser. For ANDI MS files, pass <code>ms_format</code> to return the mass spectra as a <code>data.frame</code> (the default) or a list.

Details

Retention times are returned in minutes, converted from the unit the file declares. An ANDI chrom file declares it in its `retention_unit` attribute, which also governs the peak table. Seconds is both what the template uses and what all but one of its conformance files declare, so a file that declares no unit is read as seconds, unless `chromConverter` wrote it, in which case it is read as minutes.

An ANDI MS file has no mandatory unit attribute, since the specification never formally defined its axes units, so `scan_acquisition_time` is read as seconds, the only unit the specification suggests, unless a `raw_data_time_units` attribute says otherwise.

Either kind of file warns about a unit it does not recognize and reads it as seconds.

Some writers that do not record peak heights fill `peak_height` with `-1` or `0` for every peak. A `chromatographr` peak table returns `NA` heights in these cases.

Value

A chromatogram in the format specified by the `format_out` and `data_format` arguments.

Author(s)

Ethan Bass

Examples

```
## Not run:  
read_cdf("path/to/file.cdf")  
  
## End(Not run)
```

read_chemstation_ch	<i>Read 'Agilent ChemStation' CH files</i>
---------------------	--

Description

Reads 'Agilent ChemStation' .ch files.

Usage

```
read_chemstation_ch(  
  path,  
  format_out = c("matrix", "data.frame", "data.table"),  
  data_format = c("wide", "long"),  
  read_metadata = TRUE,  
  metadata_format = c("chromconverter", "raw"),  
  scale = TRUE,  
  source_file = NULL  
)
```

Arguments

path	Path to 'Agilent' .ch file.
format_out	Class of output. Either matrix, data.frame, or data.table.
data_format	Whether to return data in wide (default) or long format.
read_metadata	Logical. Whether to attach metadata. Defaults to TRUE.
metadata_format	Format to output metadata. Either chromconverter (standardized field names) or raw (vendor field names, unmapped).
scale	Whether to multiply the data by the file's scaling factor and add its intercept. Defaults to TRUE. 'MassHunter' seems to ignore the scaling factor in at least some types of 'ChemStation' files.
source_file	Source file from which chromatogram data was originally derived.

Details

'Agilent' .ch files come in several formats. This parser detects the version from the file and reads versions 8, 30, 130, 81, 179 and 181 from 'Agilent ChemStation' and 'Agilent OpenLab'; any other version is an error. Versions 30 and 130 are generally produced by ultraviolet detectors, and 81, 179 and 181 by flame ionization detectors (FID).

Value

A 2D chromatogram in the format specified by `data_format` and `format_out`. If `data_format` is wide, the chromatogram will be returned with retention times as row names and a single column for the intensity. If long format is requested, two columns will be returned: one for the retention time and one for the intensity. The `format_out` argument determines whether the chromatogram is returned as a matrix, `data.frame`, or `data.table`. Metadata are attached to the chromatogram as [attributes](#) if `read_metadata` is TRUE.

Note

This function was adapted from the [Chromatography Toolbox](#) (© James Dillon 2014).

Author(s)

Ethan Bass

See Also

Other 'Agilent' parsers: [read_agilent_d\(\)](#), [read_agilent_dx\(\)](#), [read_agilent_rslt\(\)](#), [read_chemstation_csv\(\)](#), [read_chemstation_logs\(\)](#), [read_chemstation_method\(\)](#), [read_chemstation_ms\(\)](#), [read_chemstation_reports\(\)](#), [read_chemstation_uv\(\)](#)

Examples

```
read_chemstation_ch("tests/testthat/testdata/chemstation_130.ch")
```

`read_chemstation_csv` *Read 'Agilent ChemStation' CSV files*

Description

Reads 'Agilent ChemStation' .csv files.

Usage

```
read_chemstation_csv(  
  path,  
  format_out = "matrix",  
  data_format = "wide",  
  read_metadata = TRUE  
)
```

Arguments

path	Path to 'Agilent' .csv file.
format_out	Class of output. Either matrix, data.frame, or data.table.
data_format	Whether to return data in wide (default) or long format.
read_metadata	Logical. Whether to attach metadata. Defaults to TRUE. The files record no instrument metadata, so only the settings chromConverter used to parse the file are attached.

Details

'Agilent ChemStation' CSV files are encoded in UTF-16.

Value

A chromatogram in the format specified by format_out and data_format.

Author(s)

Ethan Bass

See Also

Other 'Agilent' parsers: [read_agilent_d\(\)](#), [read_agilent_dx\(\)](#), [read_agilent_rslt\(\)](#), [read_chemstation_ch\(\)](#), [read_chemstation_logs\(\)](#), [read_chemstation_method\(\)](#), [read_chemstation_ms\(\)](#), [read_chemstation_reports\(\)](#), [read_chemstation_uv\(\)](#)

Examples

```
read_chemstation_csv("tests/testthat/testdata/dad1.csv")
```

read_chemstation_logs *Read 'Agilent ChemStation' log files*

Description

Reads the event log 'Agilent ChemStation' writes to the folder of each sequence, or the RUN.LOG it writes to each .D folder when that file is named in paths. These record instrument errors, aborted runs and routine readings such as pump pressure and column temperature.

Usage

```
read_chemstation_logs(  
  paths,  
  what = c("injections", "problems", "events"),  
  format_out = "data.frame"  
)
```

Arguments

paths	Paths to 'ChemStation' .LOG files, or to directories to search for sequence logs.
what	What to return: one row per injection (injections, the default), only the events marked as a problem (problems), or every event (events).
format_out	Class of the returned object: data.frame (the default) or data.table.

Details

Directories in paths are searched, including their subdirectories, for sequence logs: every .LOG file outside a .D folder. RUN.LOG files are left out, since the sequence log holds their events. A file found this way that is not a 'ChemStation' log is skipped with a warning; a file named in paths that is not one is an error.

Tested on logs from 'ChemStation' revisions A.10.02, B.01.03 and B.04.02.

Value

A data.frame or data.table. For injections, one row per injection, with columns:

- sequence: the name of the 'ChemStation' sequence, taken from the log's own sequence events or, failing those, from the name of the file. NA for a RUN.LOG, which has neither.
- folder: the folder the sequence's data are in, which can differ from sequence. This is the folder holding the log, or for a RUN.LOG the folder holding its .D folder.
- injection: numbers the injections in the order the log starts them.
- data_file: the .D folder the log names for the injection. The log records it only once the run's data are analyzed, so an aborted injection has none, and its case can differ from the folder on disk.
- sample: the sample, as the log names it.
- start: POSIXct, UTC. The run_datetime of the traces and reports in a .D folder is instead the instrument's local time labeled as UTC, so match injections to them by data_file rather than by time.
- minutes: from the injection's first event to its last.
- status: completed, aborted, stopped by user or incomplete.
- pressure_start, pressure_end: the first and last reading of pump 1, in bar.
- problems: the injection's problem messages, separated by "; ".

For events, one row per event, with sequence, folder, injection and data_file as above, and:

- time: POSIXct, UTC, as recorded in the event's header. The local time printed in the log's text is not returned.
- problem: whether the event reports an alarm from an instrument module (any event from a module with a non-zero event_code, such as a leak or a shutdown), a method that was aborted, timed out, stopped by an instrument error or stopped by the user, or a sequence that was terminated or stopped.
- event_code, module_id: hexadecimal strings, since 'Agilent' does not document them. The same module always carries the same module_id (for example, 1da6 for the pump in 'Agilent 1100' logs).

- source: e.g. "1100 PMP 1", "Method" or "Sequence".
- message: the event's text. The log splits a message longer than 45 characters over several events, which are joined, but stores a message of exactly 45 characters without its last character.

Events of the sequence itself, events between injections such as loading the next method, and method runs that acquire no data, such as a data-analysis-only re-run of the sequence, have no injection or data_file.

For problems, the events marked as a problem, without that column and with incident, which numbers the runs of problem events in the same log that are less than a minute apart, such as a leak and the shutdowns that follow it.

Author(s)

Ethan Bass

See Also

Other 'Agilent' parsers: [read_agilent_d\(\)](#), [read_agilent_dx\(\)](#), [read_agilent_rslt\(\)](#), [read_chemstation_ch\(\)](#), [read_chemstation_csv\(\)](#), [read_chemstation_method\(\)](#), [read_chemstation_ms\(\)](#), [read_chemstation_reports\(\)](#), [read_chemstation_uv\(\)](#)

Examples

```
read_chemstation_logs("tests/testthat/testdata/chemstation_sequence.LOG")
read_chemstation_logs("path/to/sequences", what = "problems")
```

read_chemstation_method

Read an 'Agilent ChemStation' method

Description

Reads the pump, diode array detector, autosampler and column thermostat settings from the module register files (LPMP1.REG, LDAD1.REG, LALS1.REG or LWLS1.REG, and LTHM1.REG) of a 'ChemStation' .M method directory. Revisions that store a module's settings as 'RapidControl' XML (e.g. AgilentPumpDriver1.RapidControl.MethodXML.xml) instead are read as 'OpenLab CDS' methods are by [read_agilent_amx](#).

Usage

```
read_chemstation_method(
  path,
  what = c("pump", "dad", "sampler", "column"),
  format_out = c("data.frame", "tibble", "data.table"),
  gradient_format = c("wide", "long")
)
```

Arguments

path	Path to a .M directory, or to a .D directory holding a copy of its method.
what	One or more instrument modules to read: any combination of "pump", "dad", "sampler" and "column". Defaults to all four.
format_out	Class of the solvents, gradient, signals and temp_controls tables: "data.frame", "tibble" or "data.table".
gradient_format	Whether to return the gradient in "wide" (default) or "long" format.

Details

Depending on how 'ChemStation' is configured, a .D directory may hold a copy of the method it was acquired with, in ACQ.M or, in older revisions, RUN.M. path can then be the .D directory itself. Its DA.M holds the data analysis method instead, whose pump settings may not be those the run was acquired with; reading it gives a warning.

The gradient follows the pump's timetable, with each parameter changing linearly between the times it is set. Channel A is the pump's primary channel and delivers whatever the other channels leave, so it is always included and its share is computed rather than read.

Value

A named list with one element per module read. A module the method has no settings for is left out with a warning, and it is an error if none of the requested modules are present. The sampler module is returned as autosampler. The elements are those of [read_agilent_amx](#), with NA for a setting the method does not record:

pump: a list with scalar elements flow_mL_min, stop_time_min, post_time_min, pressure_low_bar and pressure_high_bar, plus:

- solvents: a table of the channels in use, with columns channel, percentage (at the start of the run) and solvent (the name entered for it, or NA).
- gradient: the timetable. Wide format: time_min and a pct_<channel> column per channel in use, plus flow_mL_min if the timetable sets the flow. Long format: time_min, channel and percent, where a flow channel holds the flow in mL/min. An empty timetable gives the starting composition at time 0.

dad: a list with scalar elements peakwidth_code, slitwidth_nm, uv_lamp_required, vis_lamp_required, spectra_from_nm, spectra_to_nm and spectra_step_nm, plus signals, a table of the stored signals with columns id, wavelength_nm, bandwidth_nm, reference_nm and reference_bandwidth_nm (NA without a reference). peakwidth_code is the stored code for the detector's peak width setting, which sets its data rate and response time; for example, 4 is "> 0.1 min". It is not decoded, since detector models offer different settings.

autosampler: a list with scalar elements thermostat_installed, draw_speed_uL_min, eject_speed_uL_min, wait_after_draw_min, injection_volume_uL and wash_time_s. Register files record only the speeds and the injection volume.

column: a list with scalar element post_time_min (NA when off), plus temp_controls, a two-row table (Left, Right) with columns side, temperature_C, not_ready_limit_C and equilibration_time_min. A temperature stored below absolute zero (e.g. -274) is NA; this may mean no temperature was set.

Author(s)

Ethan Bass

See Also

[read_agilent_amx](#) for 'OpenLab CDS' methods, and [read_agilent_d](#) with `what = "instrument"` for the pump's record of what it delivered.

Other 'Agilent' parsers: [read_agilent_d\(\)](#), [read_agilent_dx\(\)](#), [read_agilent_rslt\(\)](#), [read_chemstation_ch\(\)](#), [read_chemstation_csv\(\)](#), [read_chemstation_logs\(\)](#), [read_chemstation_ms\(\)](#), [read_chemstation_reports\(\)](#), [read_chemstation_uv\(\)](#)

Examples

```
## Not run:
method <- read_chemstation_method("path/to/METHOD.M")
method$pump$gradient

## End(Not run)
```

read_chemstation_ms	<i>Read 'Agilent ChemStation' MS files</i>
---------------------	--

Description

Reads 'Agilent ChemStation MSD Spectral Files' beginning with x01/x32/x00/x00.

Usage

```
read_chemstation_ms(
  path,
  what = c("MS1", "BPC", "TIC"),
  format_out = c("matrix", "data.frame", "data.table"),
  data_format = "long",
  read_metadata = TRUE,
  metadata_format = c("chromconverter", "raw"),
  collapse = TRUE
)
```

Arguments

path	Path to 'Agilent' .ms file.
what	Which streams to return: MS1, BPC and/or TIC. Defaults to all three.
format_out	Class of output. Either <code>matrix</code> , <code>data.frame</code> , or <code>data.table</code> .
data_format	Whether to return the BPC and TIC in long (default) or wide format. Mass spectra are always returned in long format.
read_metadata	Logical. Whether to attach metadata. Defaults to TRUE.

metadata_format	Format to output metadata. Either chromconverter (standardized field names) or raw (vendor field names, unmapped).
collapse	Logical. Whether to collapse lists that only contain a single element. Defaults to TRUE.

Value

A list of the streams in what, in the format specified by format_out. In wide format, the BPC and TIC have retention times as rows and a single intensity column; in long format, a retention time column and an intensity column. MS data will always be returned in long format. The format_out argument determines whether the chromatogram is returned as a matrix, data.frame, or data.table. Metadata are attached as [attributes](#) if read_metadata is TRUE. With collapse = TRUE, a list of one stream is replaced by that stream.

Note

Many thanks to Evan Shi and Eugene Kwan for providing helpful information on the structure of these files in the [rainbow documentation](#).

Author(s)

Ethan Bass

See Also

Other 'Agilent' parsers: [read_agilent_d\(\)](#), [read_agilent_dx\(\)](#), [read_agilent_rslt\(\)](#), [read_chemstation_ch\(\)](#), [read_chemstation_csv\(\)](#), [read_chemstation_logs\(\)](#), [read_chemstation_method\(\)](#), [read_chemstation_reports\(\)](#), [read_chemstation_uv\(\)](#)

Examples

```
## Not run:  
read_chemstation_ms(path)  
  
## End(Not run)
```

read_chemstation_reports

Read 'Agilent ChemStation' report files

Description

Reads peak tables from 'Agilent ChemStation' Report.TXT files, or from the Result.xml files 'ChemStation' writes when set up to export its results. Only Result.xml records where each peak starts and ends, so it is read in preference to a Report.TXT in the same directory.

Usage

```
read_chemstation_reports(  
  paths,  
  peaktable_format = c("chromatographr", "original"),  
  metadata_format = c("chromconverter", "raw"),  
  data_format = NULL  
)
```

Arguments

paths	Paths to 'ChemStation' Report.TXT or Result.xml files.
peaktable_format	Whether to return peak tables in the standard chromatographr layout (the default) or with all the columns the file provides (original). A chromatographr table has the columns rt, start, end, area, height and FWHM (the full width at half maximum), with times in the same unit as the format's chromatograms and NA where the file does not record a value.
metadata_format	Format to output metadata. Either chromconverter or raw.
data_format	Deprecated. Use peaktable_format instead.

Value

A peak_list: a list with one element per report, named by its .D directory, each holding one peak table per signal, named by wavelength.

Author(s)

Ethan Bass

See Also

Other 'Agilent' parsers: [read_agilent_d\(\)](#), [read_agilent_dx\(\)](#), [read_agilent_rslt\(\)](#), [read_chemstation_ch\(\)](#), [read_chemstation_csv\(\)](#), [read_chemstation_logs\(\)](#), [read_chemstation_method\(\)](#), [read_chemstation_ms\(\)](#), [read_chemstation_uv\(\)](#)

Examples

```
path <- system.file("extdata", "benzoxazinoid_standards", "BENZOS_250PPM.D",  
  "Report.TXT", package = "chromConverter")  
read_chemstation_reports(path)
```

read_chemstation_uv	<i>Read 'Agilent ChemStation' DAD files</i>
---------------------	---

Description

Agilent .uv files come in several formats. This parser detects the version from the file and reads versions 31 and 131 from 'Agilent ChemStation' and 'Agilent OpenLab'; any other version is an error.

Usage

```
read_chemstation_uv(  
  path,  
  format_out = c("matrix", "data.frame", "data.table"),  
  data_format = c("wide", "long"),  
  read_metadata = TRUE,  
  metadata_format = c("chromconverter", "raw"),  
  scale = TRUE,  
  source_file = NULL  
)
```

Arguments

path	Path to 'Agilent' .uv file.
format_out	Class of output. Either <code>matrix</code> , <code>data.frame</code> , or <code>data.table</code> .
data_format	Whether to return data in wide (default) or long format.
read_metadata	Logical. Whether to attach metadata. Defaults to TRUE.
metadata_format	Format to output metadata. Either <code>chromconverter</code> (standardized field names) or <code>raw</code> (vendor field names, unmapped).
scale	Whether to scale the data by the scaling factor present in the file. Defaults to TRUE.
source_file	Source file from which UV data was originally derived.

Details

Retention times are read from each spectrum, so they are unevenly spaced when the detector did not store every spectrum. Interpolate onto a common time axis before comparing runs.

Value

A 3D chromatogram in the format specified by `data_format` and `format_out`. If `data_format` is wide, the chromatogram will be returned with retention times as row names and wavelengths as columns. If long format is requested, three columns will be returned: one for the retention time, one for the wavelength and one for the intensity. The `format_out` argument determines whether the chromatogram is returned as a `matrix`, `data.frame`, or `data.table`. Metadata are attached to the chromatogram as [attributes](#) if `read_metadata` is TRUE.

Note

This function was adapted from the parser in the rainbow project licensed under GPL 3 by Evan Shi <https://rainbow-api.readthedocs.io/en/latest/agilent/uv.html>.

Author(s)

Ethan Bass

See Also

Other 'Agilent' parsers: [read_agilent_d\(\)](#), [read_agilent_dx\(\)](#), [read_agilent_rslt\(\)](#), [read_chemstation_ch\(\)](#), [read_chemstation_csv\(\)](#), [read_chemstation_logs\(\)](#), [read_chemstation_method\(\)](#), [read_chemstation_ms\(\)](#), [read_chemstation_reports\(\)](#)

Examples

```
read_chemstation_uv("tests/testthat/testdata/dad1.uv")
```

read_chromatotec	<i>Read 'Chromatotec' file</i>
------------------	--------------------------------

Description

Reads 'Chromatotec' .Chrom files.

Usage

```
read_chromatotec(  
  path,  
  what = c("chrom", "peak_table"),  
  format_out = c("matrix", "data.frame", "data.table"),  
  data_format = c("wide", "long"),  
  read_metadata = TRUE,  
  metadata_format = c("chromconverter", "raw"),  
  collapse = TRUE,  
  peaktble_format = c("chromatographr", "original")  
)
```

Arguments

path	The path to 'Chromatotec' .Chrom file.
what	Whether to extract chromatograms (chrom) and/or peak_table data. Accepts multiple arguments, and defaults to both.
format_out	Class of output. Either matrix, data.frame, or data.table.
data_format	Whether to return data in wide (default) or long format.

read_metadata Logical. Whether to attach metadata. Defaults to TRUE.

metadata_format Format to output metadata. Either chromconverter (standardized field names) or raw (vendor field names, unmapped).

collapse Logical. Whether to collapse lists that only contain a single element. Defaults to TRUE.

peaktable_format Whether to return peak tables in the standard chromatographr layout (the default) or with all the columns the file provides (original). A chromatographr table has the columns rt, start, end, area, height and FWHM (the full width at half maximum), with times in the same unit as the format's chromatograms and NA where the file does not record a value.

Value

A chromatogram and/or peak table from the specified path, according to the value of what, with times in minutes. Chromatograms are returned in the format specified by format_out. The rows of the peak table that report a total over most of the run rather than a peak, such as NMTHC-EqC6 and THC_PAMS, are only kept with peaktable_format = "original". In the original table, Start and End are where integration of the peak starts and ends, and SubstanceIndex appears to be the position of its substance in the run's substance table (0 for an unnamed peak); U1, U3 and U4 have not been identified.

Author(s)

Ethan Bass

Examples

```
## Not run:
read_chromatotec(path)

## End(Not run)
```

read_chromeleon	<i>Read 'Chromeleon' ASCII files</i>
-----------------	--------------------------------------

Description

Reads 'Thermo Fisher Chromeleon™ CDS' ASCII (.txt) files.

Usage

```
read_chromeleon(
  path,
  format_out = c("matrix", "data.frame", "data.table"),
  data_format = c("wide", "long"),
```

```
    read_metadata = TRUE,  
    metadata_format = c("chromconverter", "raw"),  
    decimal_mark = NULL  
  )
```

Arguments

path	Path to 'Chromeleon' ASCII file.
format_out	Class of output. Either matrix, data.frame, or data.table.
data_format	Whether to return data in wide (default) or long format.
read_metadata	Logical. Whether to attach metadata. Defaults to TRUE.
metadata_format	Format to output metadata. Either chromconverter (standardized field names) or raw (vendor field names, unmapped).
decimal_mark	Which character is used as the decimal separator in the file. By default, decimal mark will be detected automatically, but it can also be manually set as "." or ",".

Value

A chromatogram in the format specified by format_out (retention time x wavelength).

Author(s)

Ethan Bass

Examples

```
## Not run:  
read_chromeleon("path/to/file.txt")  
  
## End(Not run)
```

read_chroms	<i>Read Chromatograms</i>
-------------	---------------------------

Description

Reads chromatograms from specified folders or vector of paths using either an internal parser or bindings to an external library, such as [Aston](#), [Entab](#), [ThermoRawFileParser](#), [OpenChrom](#), [rainbow](#).

Usage

```

read_chroms(
  paths,
  format_in = supported_formats(),
  find_files,
  pattern = NULL,
  parser = c("", "chromconverter", "aston", "entab", "thermoraw", "openchrom", "rainbow"),
  format_out = c("matrix", "data.frame", "data.table"),
  data_format = c("wide", "long"),
  path_out = NULL,
  export_format = c("", "csv", "chemstation_csv", "cdf", "mzml", "animl", "arw"),
  force = FALSE,
  read_metadata = TRUE,
  metadata_format = c("chromconverter", "raw"),
  progress_bar,
  cl = 1,
  verbose = getOption("verbose"),
  sample_names = c("basename", "sample_name"),
  sort_by = c("auto", "none", "acquisition_time", "file_time"),
  dat = NULL,
  ...
)

```

Arguments

paths	Paths to data files or directories containing the files.
format_in	Format of the files to be imported or converted. One of: <code>agilent_d</code> , <code>agilent_dx</code> (or <code>openlab_dx</code>), <code>agilent_rslt</code> (or <code>rslt</code> , <code>sirslt</code>), <code>asm</code> (or <code>allotrope</code>), <code>cdf</code> (or <code>andi</code>), <code>chemstation_ch</code> (or <code>chemstation_fid</code>), <code>chemstation_csv</code> , <code>chemstation_ms</code> , <code>chemstation_uv</code> , <code>chromatotec</code> , <code>chromeleon_uv</code> , <code>csd</code> , <code>csv</code> , <code>masshunter_dad</code> , <code>mdf</code> , <code>msd</code> , <code>mzml</code> , <code>mzxml</code> , <code>other</code> , <code>shimadzu_ascii</code> , <code>shimadzu_dad</code> , <code>shimadzu_fid</code> , <code>shimadzu_gcd</code> , <code>shimadzu_lcd</code> , <code>shimadzu_qgd</code> , <code>thermoraw</code> , <code>varian_sms</code> , <code>waters_arw</code> , <code>waters_raw</code> , <code>wsd</code> . A name in parentheses is an alias, which behaves exactly like the format it follows. Version-specific names for the 'Agilent ChemStation' formats (<code>chemstation_130</code> , for instance) are accepted as well, but are normally supplied by <code>chromConverter</code> 's own file-type detection rather than being provided by the user.
find_files	Logical. Whether to treat paths as directories to search for data files. Inferred from paths if not supplied: anything that is not a file is searched as a directory, except for the formats that are themselves directories (e.g. 'Agilent' .d), which are recognized by their extension.
pattern	Regular expression that file names must match (e.g. a file extension). Defaults to <code>NULL</code> , in which case the extension is deduced from <code>format_in</code> .
parser	What parser to use (optional). Current options are <code>chromconverter</code> , <code>aston</code> , <code>entab</code> , <code>thermoraw</code> , <code>openchrom</code> , <code>rainbow</code> .
format_out	Class of output. Either <code>matrix</code> , <code>data.frame</code> , or <code>data.table</code> .
data_format	Whether to output data in wide or long format. Either <code>wide</code> (default) or <code>long</code> .

path_out	Path for exporting files. If it is not specified, the user is asked whether to export to a temp directory in the working directory. A directory that does not exist is created after asking.
export_format	Export format: csv, chemstation_csv (UTF-16 encoding), cdf, mzml, arw, or animl, which requires an openchrom parser.
force	Logical. Whether to overwrite files when exporting. Defaults to FALSE.
read_metadata	Logical, whether to attach metadata (if it's available). Defaults to TRUE.
metadata_format	Format to output metadata. Either chromconverter or raw.
progress_bar	Logical. Whether to show progress bar. Defaults to TRUE if pbapply is installed.
cl	Argument to pbapply specifying the number of parallel workers to use or a cluster object created by makeCluster . Defaults to 1.
verbose	Logical. Whether to print status messages, and the output of external parsers, to the R console.
sample_names	Which sample names to use. Options are basename to use the filename (default) or sample_name to use the sample name encoded in the file metadata. A sample with no sample_name, or with conflicting ones, is named for its file with a warning. Several files in one 'Agilent' .D directory that would share a name also get their file name, e.g. RUTIN_2.dad1A.
sort_by	How to sort the chromatograms: auto (default) sorts files by acquisition time unless paths lists the files explicitly or any acquisition time is missing; none keeps files in the order given, or in alphabetical order if find_files = TRUE; acquisition_time sorts by the acquisition time recorded in each file (run_datetime); file_time sorts by the time when each file was last modified.
dat	Deprecated. Existing list of chromatograms to append results to. Use c() on the returned chrom_list objects instead. Defaults to NULL.
...	Additional arguments to the parser. Where the parser does not take ..., arguments it does not accept are dropped with a warning.

Details

Provides a unified interface to all chromConverter parsers. The formats it recognizes are listed under the format_in argument. It also wraps the 'OpenChrom' parsers, which cover many additional formats but require 'OpenChrom' 1.4 or earlier (see [call_openchrom](#)). The 'Entab', 'ThermoRawFileParser' and 'OpenChrom' parsers must be installed separately; see the instructions in the [README](#).

If paths to individual files are provided, read_chroms infers the file format from the first file and selects a parser for it. When providing paths to directories, the file format must be specified using the format_in argument.

Value

A chrom_list of chromatograms in matrix, data.frame, or data.table format, according to the value of format_out. Chromatograms may be returned in either wide or long format according to the value of data_format. Peak tables are returned in the chromatographr layout described in [read_peaklist](#), which can also return all the columns the file provides.

Side effects

If `export_format` is provided, chromatograms are written to the folder given by `path_out` in that format. The options are `csv`, `chemstation_csv`, `cdf`, `mzml` and `arw`, as well as `animl` (AnIML) when an `openchrom` parser is selected. Files are also written to `path_out` whenever the `thermoraw` or `openchrom` parser is used, as these parsers convert the files before reading them: `thermoraw` to `mzML`, and `openchrom` to `export_format` (`mzml` by default).

Author(s)

Ethan Bass

Examples

```
path <- system.file("extdata/alkane_ladder.txt", package = "chromConverter")
chroms <- read_chroms(path, format_in = "shimadzu_ascii",
                      find_files = FALSE, progress_bar = FALSE)
```

read_mdf

Read 'Lumex' MDF

Description

Reads 'Lumex' .mdf files.

Usage

```
read_mdf(
  path,
  format_out = c("matrix", "data.frame", "data.table"),
  data_format = c("wide", "long"),
  read_metadata = TRUE
)
```

Arguments

<code>path</code>	The path to a 'Lumex' .mdf file.
<code>format_out</code>	Class of output. Either <code>matrix</code> , <code>data.frame</code> , or <code>data.table</code> .
<code>data_format</code>	Whether to return data in wide (default) or long format.
<code>read_metadata</code>	Logical. Whether to attach metadata. Defaults to <code>TRUE</code> .

Value

A chromatogram in the format specified by the `format_out` and `data_format` arguments.

Author(s)

Ethan Bass

Examples

```
## Not run:
read_mdf("path/to/file.mdf")

## End(Not run)
```

read_metadata	<i>Read the metadata of chromatograms</i>
---------------	---

Description

Reads the metadata of chromatogram files into a table, without returning the chromatograms. It takes the same paths and formats as [read_chroms](#) and gives the table that [extract_metadata](#) would give for the chromatograms [read_chroms](#) returns.

Usage

```
read_metadata(
  paths,
  format_in = supported_formats(),
  find_files,
  pattern = NULL,
  parser = c("", "chromconverter", "aston", "entab", "thermoraw", "openchrom", "rainbow"),
  what = chrom_metadata_fields(),
  format_out = c("data.frame", "data.table", "tibble"),
  metadata_format = c("chromconverter", "raw"),
  collapse = FALSE,
  expand = FALSE,
  digits = NULL,
  progress_bar,
  cl = 1,
  verbose = getOption("verbose"),
  sample_names = c("basename", "sample_name"),
  sort_by = c("auto", "none", "acquisition_time", "file_time"),
  ...
)
```

Arguments

paths	Paths to data files or directories containing the files.
format_in	Format of the files to be imported or converted. One of: <code>agilent_d</code> , <code>agilent_dx</code> (or <code>openlab_dx</code>), <code>agilent_rslt</code> (or <code>rslt</code> , <code>sirslt</code>), <code>asm</code> (or <code>allotrope</code>), <code>cdf</code> (or <code>andi</code>), <code>chemstation_ch</code> (or <code>chemstation_fid</code>), <code>chemstation_csv</code> , <code>chemstation_ms</code> , <code>chemstation_uv</code> , <code>chromatotec</code> , <code>chromeleon_uv</code> , <code>csd</code> , <code>csv</code> , <code>masshunter_dad</code> , <code>mdf</code> , <code>msd</code> , <code>mzml</code> , <code>mzxml</code> , <code>other</code> , <code>shimadzu_ascii</code> , <code>shimadzu_dad</code> , <code>shimadzu_fid</code> , <code>shimadzu_gcd</code> , <code>shimadzu_lcd</code> , <code>shimadzu_qgd</code> , <code>thermoraw</code> , <code>varian_sms</code> , <code>waters_arw</code> , <code>waters_raw</code> , <code>wsd</code> . A name in parentheses is an alias, which behaves exactly like

the format it follows. Version-specific names for the 'Agilent ChemStation' formats (chemstation_130, for instance) are accepted as well, but are normally supplied by chromConverter's own file-type detection rather than being provided by the user.

find_files	Logical. Whether to treat paths as directories to search for data files. Inferred from paths if not supplied: anything that is not a file is searched as a directory, except for the formats that are themselves directories (e.g. 'Agilent' .d), which are recognized by their extension.
pattern	Regular expression that file names must match (e.g. a file extension). Defaults to NULL, in which case the extension is deduced from format_in.
parser	What parser to use (optional). Current options are chromconverter, aston, entab, thermoraw, openchrom, rainbow.
what	A character vector specifying the metadata elements to extract. Defaults to every field chromConverter attaches; no format records all of them, so the elements a format does not provide are absent from the result. Superseded names (e.g. injection_volume, software_name, time_start) are accepted and mapped to the names that replaced them. An element of a nested field (see expand) may be named too, either by the column it is reported under (SampleLabel) or in full (acaml_metadata.SampleLabel), to report it without the rest of its field. A field requested by name that no chromatogram carries produces a warning.
format_out	Class of the table. Either data.frame, data.table or tibble.
metadata_format	Format to output metadata. Either chromconverter or raw.
collapse	Logical. Whether to collapse a field holding more than one value (time_range, or the product_mz of an MRM event monitoring several transitions) into a single comma-separated string. Defaults to FALSE, in which case such a field is spread across numbered columns (time_range1, time_range2).
expand	Whether to include the nested metadata fields, whose value is itself a list or table rather than a single value per chromatogram: the ms_params and method_params instrument settings, the acaml_metadata injection record that read_agilent_rslt reads from the .acaml file, or the whole vendor list that metadata_format = "raw" passes through. Either TRUE, to include every nested field the chromatograms carry, a character vector naming the ones to include, or FALSE (the default) to include none. Each element becomes a column of its own, named for itself (SampleName) unless that name is already taken, in which case it carries the field it came from (ms_params.polarity, since polarity is a metadata field in its own right).
digits	Number of significant digits for the numbers in a field collapsed into a string (see collapse), or NULL (the default) to keep them in full. Numeric fields spread across columns are returned as numbers either way.
progress_bar	Logical. Whether to show progress bar. Defaults to TRUE if pbapply is installed.
cl	Argument to pbapply specifying the number of parallel workers to use or a cluster object created by makeCluster . Defaults to 1.
verbose	Logical. Whether to print status messages, and the output of external parsers, to the R console.

sample_names	Which sample names to use. Options are basename to use the filename (default) or sample_name to use the sample name encoded in the file metadata. A sample with no sample_name, or with conflicting ones, is named for its file with a warning. Several files in one 'Agilent' .D directory that would share a name also get their file name, e.g. RUTIN_2.dad1A.
sort_by	How to sort the chromatograms: auto (default) sorts files by acquisition time unless paths lists the files explicitly or any acquisition time is missing; none keeps files in the order given, or in alphabetical order if find_files = TRUE; acquisition_time sorts by the acquisition time recorded in each file (run_datetime); file_time sorts by the time when each file was last modified.
...	Additional arguments to the parser. Arguments that a format's metadata reader does not take are ignored.

Details

The formats that are slow to read in full are read without decoding the data: 'Agilent' .uv, 'Chromeleon', 'OpenLab' .dx, 'Shimadzu' .lcd, .gcd, .qgd and ASCII, 'Thermo' RAW, 'Varian' SMS and 'Waters' .arw files, and ANDI (.cdf), mzML and mzXML. 'OpenLab' 131 .uv files are the exception: their scaling is worked out from the data, so they are decoded in full. Other formats are read in full and their metadata extracted.

The 'OpenLab' .dx and 'Shimadzu' formats report only the fields that describe the whole file, leaving out those of its traces, such as detector or wavelength. 'Thermo' RAW files leave out the fields that come from the mzML they are converted to, such as detector_range.

If parser is not given, each format is read with the parser that can skip the data, which for .uv files is chromconverter even where 'Entab' is installed.

Value

A data.frame, data.table or tibble with one row per sample and one column per metadata field, as described in [extract_metadata](#). A table with no rows if none of the files could be read.

Author(s)

Ethan Bass

Examples

```
path <- system.file("extdata", "benzoxazinoid_standards",
                    package = "chromConverter")
read_metadata(path, format_in = "chemstation_ch", pattern = "dad1A",
              parser = "chromconverter", progress_bar = FALSE,
              what = c("sample_name", "run_datetime", "method"))
```

read_mzml

*Read mzML files***Description**

Extracts data from mzML files using parsers from either RaMS or mzR.

Usage

```
read_mzml(
  path,
  format_out = c("matrix", "data.frame", "data.table"),
  data_format = c("wide", "long"),
  parser = c("RaMS", "mzR"),
  what = c("MS1", "MS2", "BPC", "TIC", "DAD", "chroms", "metadata", "everything"),
  verbose = FALSE,
  ...
)
```

Arguments

path	Path to .mzml file.
format_out	Class of output. Either matrix, data.frame, or data.table. Applies only to the TIC, BPC, DAD and chroms in wide format.
data_format	Whether to return data in wide (default) or long format.
parser	What parser to use. Either RaMS or mzR.
what	What types of data to return (argument to RaMS::grabMSdata). Options include MS1, MS2, BPC, TIC, DAD, chroms, metadata, or everything. Defaults to all of them. EIC and EIC_MS2 return extracted ion chromatograms from the MS1 or MS2 data for the masses given by the mz and ppm arguments of RaMS::grabMSdata , passed through ...
verbose	Argument to grabMSdata controlling verbosity. Ignored by mzR.
...	Additional arguments to grabMSdata. Ignored by mzR.

Details

Both parsers return a list with one element per stream in what. Mass spectra and EICs are always long. With data_format = "wide" (the default), TIC and BPC are returned as 2D chromatograms of class format_out, and DAD as a wide chromatogram of class format_out; with "long", every stream is a long data.table.

The chroms stream holds the chromatograms stored in the file, such as the SIM and SRM traces of a triple quadrupole run. In wide format it is a list of 2D chromatograms named by their ID in the file (or Chromatogram 1, Chromatogram 2 and so on where it has none), each with the precursor_mz and product_mz it monitors as attributes. If any trace repeats a retention time, as in some malformed files, the whole stream is returned as the long data.table instead, with a warning.

The mzR parser reads files that RaMS cannot, such as those compressed with MS-Numpress. It returns only MS1, MS2, BPC, TIC and DAD, and no file metadata. Its TIC and BPC are the per-scan values recorded in the MS1 spectra.

Value

A named list of the streams in what.

Author(s)

Ethan Bass

Examples

```
## Not run:
read_mzml("path/to/file.mzML")

## End(Not run)
```

read_peaklist	<i>Read peak lists</i>
---------------	------------------------

Description

Reads peak lists from specified folders or vector of paths.

Usage

```
read_peaklist(
  paths,
  find_files,
  format_in = c("chemstation", "shimadzu_fid", "shimadzu_dad", "shimadzu_lcd",
    "shimadzu_gcd", "chromatotec", "asm", "cdf", "openlab"),
  pattern = NULL,
  peaktable_format = c("chromatographr", "original"),
  metadata_format = c("chromconverter", "raw"),
  read_metadata = TRUE,
  progress_bar,
  cl = 1,
  sort_by = c("auto", "none", "acquisition_time", "file_time"),
  data_format = NULL
)
```

Arguments

paths	Paths to files or folders containing peak list files.
find_files	Logical. Whether to treat the supplied paths as directories to search for files. Inferred if not supplied, by testing whether every path is a file.
format_in	Format of files to be imported/converted. One of chemstation (the default), shimadzu_fid, shimadzu_dad, shimadzu_lcd, shimadzu_gcd, chromatotec, asm, cdf (ANDI chrom) or openlab ('Agilent OpenLab CDS' .rx files, named by signal from the .acaml file beside them).
pattern	A pattern (e.g. a file extension). Defaults to NULL, in which case the file extension will be deduced from format_in.
peaktable_format	Whether to return peak tables in the standard chromatographr layout (the default) or with all the columns the file provides (original). A chromatographr table has the columns rt, start, end, area, height and FWHM (the full width at half maximum), with times in the same unit as the format's chromatograms and NA where the file does not record a value.
metadata_format	Format to output metadata. Either chromconverter (standardized field names) or raw (vendor field names, unmapped).
read_metadata	Logical. Whether to attach metadata. Defaults to TRUE.
progress_bar	Logical. Whether to show progress bar. Defaults to TRUE if pbapply is installed.
cl	Argument to pbapply specifying the number of parallel workers to use or a cluster object created by makeCluster (a set of parallel R worker processes). Defaults to 1.
sort_by	How to sort the samples: auto (default) sorts files by acquisition time unless paths lists the files explicitly or any acquisition time is missing; none keeps files in the order given, or in alphabetical order if find_files = TRUE; acquisition_time sorts by the acquisition time recorded in each file (run_datetime); file_time sorts by the time when each file was last modified.
data_format	Deprecated. Use peaktable_format instead.

Value

A peak_list: a list with one element per sample, holding its peak table, or a list of peak tables where the file records more than one. The tables are named by wavelength (e.g. "254") where the file records one, and otherwise by the file's own name for the signal; a table identical to another at the same wavelength is a copy and is left out. Each row is a peak. Every table starts with a sample column and a lambda column giving the signal's wavelength, which is NA for a detector without one or a file that does not record it, followed by the columns of peaktable_format.

Author(s)

Ethan Bass

Examples

```
path <- system.file("extdata", "benzoxazinoid_standards",
                    package = "chromConverter")
peak_list <- read_peaklist(path, progress_bar = FALSE)
names(peak_list)
peak_list[["BENZOS_250PPM"]][["254"]]
```

read_shimadzu

Read 'Shimadzu' ASCII

Description

Reads 'Shimadzu' ASCII (.txt) files. These files can be exported from 'Shimadzu LabSolutions' by right clicking on samples in the sample list and selecting File Conversion:Convert to ASCII.

Usage

```
read_shimadzu(
  path,
  what = "chroms",
  format_in = NULL,
  include = c("fid", "lc", "dad", "uv", "tic"),
  format_out = c("matrix", "data.frame", "data.table"),
  data_format = c("wide", "long"),
  peaktable_format = c("chromatographr", "original"),
  read_metadata = TRUE,
  metadata_format = c("chromconverter", "raw"),
  ms_format = c("data.frame", "list"),
  collapse = TRUE,
  scale = TRUE
)
```

Arguments

path	Path to Shimadzu .txt ASCII file.
what	Whether to extract chromatograms (chroms), peak_table, and/or ms_spectra. Accepts multiple arguments.
format_in	This argument is deprecated and is no longer required.
include	Which chromatograms to include: any of fid, lc, dad, uv, tic, and status. Defaults to all but status.
format_out	Class of output. Either matrix, data.frame, or data.table.
data_format	Whether to return data in wide (default) or long format.

peaktable_format	Whether to return peak tables in the standard chromatographr layout (the default) or with all the columns the file provides (original). A chromatographr table has the columns rt, start, end, area, height and FWHM (the full width at half maximum), with times in the same unit as the format's chromatograms and NA where the file does not record a value.
read_metadata	Logical. Whether to attach metadata. Defaults to TRUE.
metadata_format	Format to output metadata. Either chromconverter (standardized field names) or raw (vendor field names, unmapped).
ms_format	Whether to return mass spectral data as a (long) data.frame or a list.
collapse	Logical. Whether to collapse lists that only contain a single element. Defaults to TRUE.
scale	Whether to scale the data by the scaling factor present in the file. Defaults to TRUE.

Value

A nested list of elements from the specified file, where the top levels are chromatograms, peak tables, and/or mass spectra according to the value of what, and the chromatograms are grouped by type (e.g. fid). Where collapse is TRUE, a list of one is unwrapped at the top level and within each type of chromatogram. Chromatograms are returned in the format specified by format_out and data_format.

Author(s)

Ethan Bass

See Also

Other 'Shimadzu' parsers: [read_shimadzu_gcd\(\)](#), [read_shimadzu_lcd\(\)](#), [read_shimadzu_method\(\)](#), [read_shimadzu_qgd\(\)](#), [read_sz_lcd_2d\(\)](#), [read_sz_lcd_3d\(\)](#), [read_sz_tables\(\)](#)

Examples

```
path <- system.file("extdata/alkane_ladder.txt", package = "chromConverter")
read_shimadzu(path)
```

read_shimadzu_gcd	<i>Read 'Shimadzu' GCD</i>
-------------------	----------------------------

Description

Read chromatogram data streams from 'Shimadzu' .gcd files.

Usage

```
read_shimadzu_gcd(
  path,
  what = "chroms",
  format_out = c("matrix", "data.frame", "data.table"),
  data_format = c("wide", "long"),
  read_metadata = TRUE,
  metadata_format = c("chromconverter", "raw"),
  collapse = TRUE,
  peaktable_format = c("chromatographr", "original")
)
```

Arguments

<code>path</code>	Path to 'Shimadzu' .gcd file.
<code>what</code>	What stream to get: current options are chromatograms (chroms) and/or peak lists (peak_table). If a stream is not specified, the function will default to chroms.
<code>format_out</code>	Class of output. Either matrix, data.frame, or data.table.
<code>data_format</code>	Whether to return data in wide (default) or long format.
<code>read_metadata</code>	Logical. Whether to attach metadata. Defaults to TRUE.
<code>metadata_format</code>	Format to output metadata. Either chromconverter (standardized field names) or raw (vendor field names, unmapped).
<code>collapse</code>	Logical. Whether to collapse lists that only contain a single element. Defaults to TRUE.
<code>peaktable_format</code>	Whether to return peak tables in the standard chromatographr layout (the default) or with all the columns the file provides (original). A chromatographr table has the columns rt, start, end, area, height and FWHM (the full width at half maximum), with times in the same unit as the format's chromatograms and NA where the file does not record a value.

Details

A parser to read chromatogram data streams from 'Shimadzu' .gcd files. GCD files are encoded as 'Microsoft' OLE documents. The parser relies on the [olefile](#) package in Python to unpack the files. The chromatogram data is encoded in streams titled LSS Raw Data:Chromatogram Ch<#>. Each stream begins with a 24-byte header:

- 4 bytes: segment label (17234).
- 4 bytes: Little-endian integer specifying the sampling interval in milliseconds.
- 4 bytes: Little-endian integer specifying the number of values in the stream.
- 4 bytes: Little-endian integer specifying a byte count, which does not match the size of the stream exactly.
- 8 bytes of 00s

After the header, the data are encoded as 64-bit (little-endian) floating-point numbers. Retention times are derived from the number of values and the sampling interval encoded in the header, rather than read from the file: the *n*th value is placed at *n* times the sampling interval.

Value

A 2D chromatogram in the format specified by `data_format` and `format_out`. If `data_format` is `wide`, the chromatogram will be returned with retention times as row names and a single column for the intensity. If `long` format is requested, two columns will be returned: one for the retention time and one for the intensity. The `format_out` argument determines whether the chromatogram is returned as a `matrix`, `data.frame`, or `data.table`. Metadata are attached to the chromatogram as `attributes` if `read_metadata` is `TRUE`.

Author(s)

Ethan Bass

See Also

Other 'Shimadzu' parsers: `read_shimadzu()`, `read_shimadzu_lcd()`, `read_shimadzu_method()`, `read_shimadzu_qgd()`, `read_sz_lcd_2d()`, `read_sz_lcd_3d()`, `read_sz_tables()`

Examples

```
## Not run:
read_shimadzu_gcd("path/to/file.gcd")

## End(Not run)
```

read_shimadzu_lcd	<i>Read 'Shimadzu' LCD</i>
-------------------	----------------------------

Description

Read PDA, chromatogram, mass spectrometry and peak table streams from 'Shimadzu' .lcd files.

Usage

```
read_shimadzu_lcd(
  path,
  what,
  format_out = c("matrix", "data.frame", "data.table"),
  data_format = c("wide", "long"),
  read_metadata = TRUE,
  metadata_format = c("chromconverter", "raw"),
  scale = TRUE,
  sparse = TRUE,
  centroided = TRUE,
```

```
collapse = TRUE,
peaktable_format = c("chromatographr", "original")
)
```

Arguments

path	Path to 'Shimadzu' .lcd file.
what	<p>What stream to get: current options are DAD (for which PDA is accepted as a synonym, since that is what 'Shimadzu' calls the same detector), chromatograms (chroms), TIC, mass spectra (MS1, MS2, or MS for both), and/or peak lists (peak_table). MS1 and MS2 count stages of mass selection rather than name a scan type, so a triple quadrupole acquisition can land on either: an MRM scan selects a precursor in Q1 (the first quadrupole) and a product in Q3 (the third), so it is MS2, while a SIM scan sets Q1 and Q3 to the same mass, selecting nothing after the collision cell, and so is MS1.</p> <p>If a stream is not specified, the richest one the file contains is returned: DAD if there is a PDA stream, otherwise chroms, and otherwise MS for a file whose only detector is the mass spectrometer. The mass spectrometry streams are read from whichever container the file uses: QTFL RawData (centroided quadrupole time-of-flight data), TLM Raw Data (triple quadrupole full scan, product-ion scan, MRM and SIM data), TTFL Raw Data (ion trap time-of-flight profile data) or Mass Raw Data (centroided or profile single quadrupole data).</p>
format_out	Class of output. Either matrix, data.frame, or data.table.
data_format	Whether to return data in wide (default) or long format.
read_metadata	Logical. Whether to attach metadata. Defaults to TRUE.
metadata_format	Format to output metadata. Either chromconverter (standardized field names) or raw (vendor field names, unmapped).
scale	Whether to scale the data by the scaling factor present in the file. Defaults to TRUE.
sparse	Logical. Whether to return mass spectra in sparse format (excluding zeros), as <code>call_rainbow()</code> does. Defaults to TRUE. Applies only to triple quadrupole profile spectra, whose m/z grid is largely empty; ignored for every other stream.
centroided	Logical. Whether to return the centroided single quadrupole scans (TRUE, the default) or the profile scans (FALSE), which some methods record as a separate event. Ignored for every other stream.
collapse	Logical. Whether to collapse lists that only contain a single element. Defaults to TRUE.
peaktable_format	Whether to return peak tables in the standard chromatographr layout (the default) or with all the columns the file provides (original). A chromatographr table has the columns rt, start, end, area, height and FWHM (the full width at half maximum), with times in the same unit as the format's chromatograms and NA where the file does not record a value.

Details

A parser to read data from 'Shimadzu' .lcd files. LCD files are encoded as 'Microsoft' OLE documents. The parser relies on the [olefile](#) package in Python to unpack the files. Each detector writes its own storage, and this function dispatches over them:

- **PDA** (PDA 3D Raw Data:3D Raw Data), requested as DAD: one delta-encoded segment per retention time, each holding a full spectrum. Read by [read_sz_lcd_3d\(\)](#), which documents the segment header and the delta encoding.
- **Chromatograms** (LSS Raw Data:Chromatogram Ch<#>): one stream per channel, delta-encoded in the same way. Read by [read_sz_lcd_2d\(\)](#).
- **Quadrupole time-of-flight mass spectra** (QTFL RawData): centroided scans stored as a scan header, a block of flight times and a block of intensities, with the mass axis reconstructed from the calibration table in the file. Read by [read_sz_qtof\(\)](#), which documents the scan header, the flight-time conversion and the intensity scaling. The total ion current is held separately, in Centroid SumTIC ([read_sz_qtof_tic\(\)](#)).
- **Triple quadrupole mass spectra** (TLM Raw Data): zlib-compressed scan records located through a spectrum index, covering full scan, product-ion scan, MRM and SIM acquisitions. Read by [read_sz_tlm\(\)](#), which documents the index, the scan header and the layout of each scan type. The total ion current is again held separately, in TIC Data.
- **Ion trap time-of-flight mass spectra** (TTFL Raw Data): run-length encoded profile scans from the LCMS-IT-TOF, with the mass axis interpolated between the calibrant ions stored in the file. Read by [read_sz_ittof\(\)](#), which documents the scan header, the encoding and the calibration. The total ion current is read from the scan headers ([read_sz_ittof_tic\(\)](#)).
- **Single quadrupole mass spectra** (Mass Raw Data): centroided and profile scans from instruments such as the LCMS-2020, located through a spectrum index. Read by [read_sz_sq\(\)](#), which documents the scan header, the records and the approximate profile m/z axis. The total ion current is held separately, in TIC Data ([read_sz_sq_tic\(\)](#)).
- **Peak tables** (Peak Table): integration results as reported by 'LabSolutions', one stream per channel. Read by [read_sz_tables\(\)](#), which documents the two record layouts.

The mass spectrometry containers are mutually exclusive: a file holds one of them, according to the instrument that wrote it.

A mass spectrometry run is divided into **acquisition events**: the scan functions defined by the method, each with its own polarity, MS level and mass range, which the instrument cycles through as the run proceeds. The mass spectrometry streams are reported per event — TIC returns one chromatogram for each, and the scan_info attribute of a table of spectra names the event every scan came from.

Value

A named list with one element per stream requested, or the element itself where there is only one and collapse is TRUE. Chromatograms and the PDA data are returned in the format specified by data_format and format_out. In wide format, retention times are the rows, and each 2D chromatogram has a single intensity column while the PDA data have one column per wavelength. If long format is requested, the retention time and the intensity are returned as columns, alongside the detector, channel, wavelength and unit for a 2D chromatogram. The format_out argument determines whether they are returned in matrix, data.frame, or data.table format. Mass spectra and peak tables are always long. Metadata are attached as [attributes](#) when read_metadata is TRUE.

Note

Times are stored as a 'Windows' FILETIME, which is always UTC, so `run_datetime` is reported in UTC. The files also record the offset of the local time zone (e.g. `+01'00'`), but this is the standard offset of the zone rather than the offset that was in force, and it seems that no daylight saving information is stored anywhere in the file. The local times displayed by 'LabSolutions' therefore cannot be reconstructed from the recorded offset alone: where daylight saving time applied, they are an hour ahead of it. Rendering `run_datetime` in the zone where the data were acquired, e.g. `format(attr(x, "run_datetime"), tz = "Europe/Paris")`, recovers them exactly.

As of `v0.10.0`, 2D chromatograms are scaled by the calibration factor and the value factor recorded for each channel, so their intensities match those reported by 'LabSolutions'. An absorbance axis can be reported in uAU, mAU or AU, and the file records the size of each as a **value factor**: 1, 1000 and 1e6, since one mAU is a thousand uAU and one AU a million. The smallest of them is the **base unit**, while 'LabSolutions' displays the data in whichever unit the method selected, usually mAU. Other detectors work the same way: a refractive index axis measures in nRI and displays uRI.

Two factors separate the stored integers from the displayed value. The calibration factor converts an integer into base units, and the value factor converts base units into the displayed unit; an intensity as 'LabSolutions' reports it is the integer times the one divided by the other. The calibration factor is 1 on some channels and not on others (e.g. about 0.024 for an SPD-20A and 0.0032 for an RID-10A), which suggests the integers are detector counts whose size varies by module, though the file does not say so.

For a 2D chromatogram, `scale = TRUE` applies both factors. `scale = FALSE` returns the stored integers. Where the calibration factor is 1, their `detector_y_unit` is the base unit (e.g. uAU), and the values and the attribute agree. Where it is not, the integers are in no unit we can name, so `detector_y_unit` is NA; the scaled attribute records which of the two was returned.

PDA data is instead returned as it is encoded in the file, matching the [PDA 3D] section of a 'LabSolutions' ASCII export, which declares no intensity unit or multiplier. The 3D Data Item describes the absorbance axis in mAU with a value factor of 1000, which would imply scaling the values by 0.001, but the [PDA Multi Chromatogram] traces in the ASCII export, which are extracted from the same data, report values on the same scale as the raw data with a multiplier of 1. Until this can be resolved, PDA data is left unscaled and the `scale` argument is ignored.

Max Plot is the maximum absorbance over the wavelength range at each point in time, so the value at one time can come from a different wavelength than the value at the next, and no single wavelength describes the trace. It is derived from the PDA data but is read as a 2D chromatogram, so it is scaled and currently differs from the PDA data by a factor of 1000. The wavelength range it was taken over is not recorded in the 2D Data Item, whose nanometer axis spans 0 to 0, so its wavelength attribute is NA rather than the acquisition range of the PDA stream.

Author(s)

Ethan Bass

See Also

Other 'Shimadzu' parsers: `read_shimadzu()`, `read_shimadzu_gcd()`, `read_shimadzu_method()`, `read_shimadzu_qgd()`, `read_sz_lcd_2d()`, `read_sz_lcd_3d()`, `read_sz_tables()`

Examples

```
## Not run:
read_shimadzu_lcd(path)

## End(Not run)
```

read_shimadzu_method *Read a 'Shimadzu' method*

Description

Reads the instrument settings stored with a 'Shimadzu' GC run (.gcd), LC run (.lcd) or GC-MS run (.qgd), such as the oven program, gradient, flows and column.

Usage

```
read_shimadzu_method(
  path,
  what = NULL,
  format_out = c("data.frame", "tibble", "data.table")
)
```

Arguments

path	Path to a .gcd, .lcd or .qgd file.
what	One or more modules to read. For GC runs, any of "oven", "injector", "detector", "autosampler" and "column"; for GC-MS runs, "oven", "injector", "autosampler", "column" and "ms"; for LC runs, "pump", "column" and, with a PDA detector, "dad", and with a single or triple quadrupole mass spectrometer, "ms". Defaults to all the modules the run has.
format_out	Class of the tables returned: "data.frame", "tibble" or "data.table".

Details

Method parameters are attached automatically to every chromatogram and peak table read from these files, in the method_params metadata field. This includes all scalar values but leaves out the oven program, gradient and other tables, which are only returned by this function. The ms module of a GC-MS or LC-MS run is attached as ms_params.

Value

A named list with one element per module.

For GC runs:

oven: a list with scalar elements equilibration_time_min, initial_temperature_C and initial_hold_min, plus program, a table of the temperature ramps with columns rate_C_min, temperature_C and hold_min.

injector: a list with scalar elements name, temperature_C, split_ratio, column_flow_mL_min, linear_velocity_cm_s, pressure_kPa, total_flow_mL_min and purge_flow_mL_min.

detector: a list with scalar elements name, temperature_C, h2_flow_mL_min, air_flow_mL_min, makeup_flow_mL_min and sampling_rate_ms.

autosampler: a list with scalar element injection_volume_uL.

column: a list with scalar elements name, length_m, diameter_mm, film_thickness_um, max_temperature_C and serial_number.

For GC-MS runs, oven, injector, autosampler and column as for GC runs, and ms, a list with scalar elements start_time_min and end_time_min (the window in which the mass spectrometer acquires), ion_source_temperature_C, interface_temperature_C and event_time_s. Files without the labeled method (e.g. from a GC-2010) record no equilibration_time_min, injector name or pressure_kPa, injection_volume_uL, column diameter_mm or film_thickness_um, or interface_temperature_C, which are then NA. A split ratio of -1 appears to mean the split is off. The injector settings are the values at the start of the run; a method may program the pressure or purge flow to change later, so a purge flow of 0 can be the start of a purge program.

For LC runs:

pump: a list with scalar elements mode, the abbreviation 'LabSolutions' stores for the pump mode (e.g. ISO, BGE or LPGE), flow_mL_min and stop_time_min, plus gradient, a table with columns time_min and pct_B, and solvents, a table of the mobile phase names entered in the method, with columns pump, line and solvent, or NULL if none were entered. The gradient starts from the pump's initial concentration of B at time 0 and changes linearly between the times listed. It is NULL for an isocratic run whose time program sets no concentration. flow_mL_min is NA if the method sets no flow.

column: a list with scalar element temperature_C, the temperature of the column oven, which is NA if the oven is not in use.

ms: the ion source interface of a single or triple quadrupole mass spectrometer, a list with scalar elements dl_temperature_C, the temperature of the desolvation line (CDL on single quadrupole instruments), and heat_block_temperature_C. It is attached to the chromatograms as ms_params.

dad: the PDA detector, a list with scalar elements start_wavelength_nm, end_wavelength_nm, sampling_interval_ms, end_time_min and cell_temperature_C (NA if the cell is not thermostatted), plus channels, a table of the channels the acquisition method extracts, with columns channel, wavelength_nm and bandwidth_nm. Data processing can change these channels; the wavelengths of the peak tables are those of the processed channels.

File format

.gcd and .lcd files store the method and instrument configuration as labeled XML, in the GUC.1.METHOD and GUC.1.CONFIG streams, and the PDA settings in PDA.1.METHOD. The settings of a single or triple quadrupole mass spectrometer are records stored as hex in the same XML, in the LCMS unit of its own GUC.1.METHOD or GUM.METHOD stream; of these, only the interface record (Ifs) is read, whose DL and heat block temperatures are uint16 in hundredths of a degree at bytes 70 and 74. They were checked against a method printout for an 'LCMS-2020' and the published method of an 'LCMS-8040'.

.qgd files from a GC-2030 (e.g. with a 'GCMS-QP2020 NX') also hold the GC method and configuration as labeled XML, in GCMS Configuration/GUC_GC2, and their oven, injector, autosampler and column settings are read from it as for .gcd files. Every .qgd file stores the method as unlabeled

binary records, one stream per module, at fixed byte offsets, which are read for the other settings and for files without the XML. Values are little-endian float32 unless marked otherwise, and bytes not listed are not read. The same streams are found in files from 'GCMS-QP2010' and 'QP2020 NX' instruments:

- GC-2010 Instrument Parameters/Column Oven Parameter, 260 bytes:
 - 8: the number of ramps, n (int32).
 - 12, 16: the initial temperature and hold.
 - 20, 100, 180: the rates, target temperatures and holds, each an array of 20 slots, of which those past n are zero.
- GC-2010 Instrument Parameters/Injection Parameter-1, 512 bytes:
 - 12, 16, 20: the column flow, linear velocity and split ratio.
 - 24, 120, 216, 312: the temperature, total flow, pressure and purge flow programs, 96 bytes each: the number of ramps (int32), the initial value and hold, then arrays of 7 rates, targets and holds.
- GCMS Configuration/Column-1, 832 bytes:
 - 0, 64: the column name and serial number, null-terminated strings of up to 64 bytes.
 - 132: the length in tenths of a metre (int32).
 - 140: the maximum temperature (int32).
- QP5K Instrument Parameters/MS Parameter, 600 or 664 bytes:
 - 20: the ion source temperature in tenths of a degree (int32).
 - 44, 48: the start and end of acquisition in milliseconds (int32).
 - 52: the event time in milliseconds (int32).

The interface temperature is only read from the XML, as the additional heater the configuration names MS. Method heaters are numbered by their position in the configuration's list of heaters, which starts at 2, so the MS heater is method heater 4 in every file seen.

The settings were checked against the 'GCMSsolution' method printouts published with three 'QP2020 NX' runs.

Author(s)

Ethan Bass

See Also

[read_chemstation_method](#) and [read_agilent_amx](#) for 'Agilent' methods.

Other 'Shimadzu' parsers: [read_shimadzu\(\)](#), [read_shimadzu_gcd\(\)](#), [read_shimadzu_lcd\(\)](#), [read_shimadzu_qgd\(\)](#), [read_sz_lcd_2d\(\)](#), [read_sz_lcd_3d\(\)](#), [read_sz_tables\(\)](#)

Examples

```
## Not run:
method <- read_shimadzu_method("path/to/file.gcd")
method$oven$program

## End(Not run)
```

read_shimadzu_qgd	<i>Read 'Shimadzu' QGD files</i>
-------------------	----------------------------------

Description

Reads 'Shimadzu GCMSsolution' .qgd GC-MS data files.

Usage

```
read_shimadzu_qgd(
  path,
  what = c("MS1", "MS2", "TIC"),
  format_out = c("matrix", "data.frame", "data.table"),
  data_format = c("wide", "long"),
  read_metadata = TRUE,
  metadata_format = c("chromconverter", "raw"),
  collapse = TRUE
)
```

Arguments

path	Path to 'Shimadzu' .qgd file.
what	What stream to get: current options are full scans (MS1), MRM scans (MS2) and the TIC. If a stream is not specified, the function returns all three. A level asked for by name that the file doesn't hold is left out with a warning.
format_out	Class of output. Either matrix, data.frame, or data.table.
data_format	Whether to return data in wide (default) or long format.
read_metadata	Logical. Whether to attach metadata. Defaults to TRUE.
metadata_format	Format to output metadata. Either chromconverter (standardized field names) or raw (vendor field names, unmapped).
collapse	Logical. Whether to collapse lists that only contain a single element. Defaults to TRUE.

Details

The MS data is stored in the GCMS Raw Data storage, which contains an MS Raw Data stream with MS scans, a TIC Data stream containing the total ion chromatogram, and a Retention Time stream containing the retention times. All known values are little-endian. The retention time stream is an array of 4-byte integers giving the time of each scan in milliseconds, and the TIC stream an array of 8-byte integers, one per retention time.

The MS Raw Data stream holds one block per scan, located through the Spectrum Index stream. The index is an array of byte offsets, held as 4-byte integers in some files and as 8-byte integers after a 2-byte header in others, such as the MRM files seen so far. Each block begins with a 32-byte header:

- scan number (4-byte integer)
- retention time in milliseconds (4-byte integer)
- unknown (12 bytes)
- number of bytes in each intensity value (2-byte integer)
- number of m/z values in the block (2-byte integer)
- MRM event number (2-byte integer)
- number of MRM transitions in the block (2-byte integer)
- unknown (4 bytes)

In a full scan, each m/z value is followed by its intensity. The m/z values are 2-byte integers holding m/z times 20. Intensities are unsigned integers of the width given in the header, except that the top bit of a 4-byte intensity is dropped, since it appears to flag a saturated value, as it does in 'Shimadzu' triple quadrupole .lcd files.

An MRM scan leaves the width and the number of m/z values at zero and instead holds one record per transition: the precursor and product m/z, each stored like the m/z of a full scan, followed by an intensity, an unsigned integer whose top bit is dropped when it is 4 bytes wide. No MRM file with 4-byte intensities has been seen. The header doesn't give the intensity width, so the reader works it out from the size of the block.

Value

A list holding whichever of the requested spectra the file has (MS1, MS2) and the TIC, or the one requested where collapse is TRUE. The TIC is returned in the format specified by data_format and format_out. If data_format is wide, it is returned with retention times as rows and a single column for the intensity. If long format is requested, two columns are returned: one for the retention time and one for the intensity. The format_out argument determines whether it is returned as a matrix, data.frame, or data.table. The spectra are always long, with columns scan, rt, mz and intensity, plus precursor_mz before mz in the MRM spectra, and a matrix resolves to a data.table. Each table of spectra has a scan_info attribute with one row per scan, giving its scan, rt, ms_level and scan_type ("scan" or "MRM"). Metadata are attached as attributes if read_metadata is TRUE.

Author(s)

Ethan Bass

See Also

Other 'Shimadzu' parsers: [read_shimadzu\(\)](#), [read_shimadzu_gcd\(\)](#), [read_shimadzu_lcd\(\)](#), [read_shimadzu_method\(\)](#), [read_sz_lcd_2d\(\)](#), [read_sz_lcd_3d\(\)](#), [read_sz_tables\(\)](#)

Examples

```
## Not run:
read_shimadzu_qgd("path/to/file.qgd")

## End(Not run)
```

read_thermoraw	<i>Read ThermoRaw</i>
----------------	-----------------------

Description

Converts a 'Thermo' .raw file to mzML with the [ThermoRawFileParser](#) and reads the result with [read_mzml](#).

Usage

```
read_thermoraw(
  path,
  path_out = NULL,
  format_out = c("matrix", "data.frame", "data.table"),
  read_metadata = TRUE,
  metadata_format = c("chromconverter", "raw"),
  centroid = TRUE,
  verbose = getOption("verbose")
)
```

Arguments

path	Path to 'Thermo' .raw file.
path_out	Path to directory to export mzML files. If path_out isn't specified, a temporary directory is used and deleted afterwards.
format_out	Class of output. Either matrix, data.frame, or data.table.
read_metadata	Logical. Whether to attach metadata. Defaults to TRUE.
metadata_format	Format to output metadata. Either chromconverter (standardized field names) or raw (vendor field names, unmapped).
centroid	Logical. Whether to centroid profile spectra with the vendor's peak picking. Defaults to TRUE. Set it to FALSE to keep profile spectra as acquired, which are about ten times larger. Spectra acquired in centroid mode stay centroided either way.
verbose	Logical. Whether to print output from ThermoRawFileParser to the console. Ignored on Windows, where the output is always printed.

Details

The ThermoRawFileParser must be installed separately.

Value

A list of chromatograms, as returned by [read_mzml](#), in the format specified by format_out.

Side effects

If path_out is specified, the mzML file (and, if read_metadata is TRUE, a metadata .txt file) is left there.

Author(s)

Ethan Bass

References

Hulstaert Niels, Jim Shofstahl, Timo Sachsenberg, Mathias Walzer, Harald Barsnes, Lennart Martens, and Yasset Perez-Riverol. ThermoRawFileParser: Modular, Scalable, and Cross-Platform RAW File Conversion. *Journal of Proteome Research* **19**, no. 1 (January 3, 2020): 537–42. doi:[10.1021/acs.jproteome.9b00328](https://doi.org/10.1021/acs.jproteome.9b00328).

See Also

Other external parsers: [call_entab\(\)](#), [call_openchrom\(\)](#), [call_rainbow\(\)](#), [sp_converter\(\)](#), [uv_converter\(\)](#)

Examples

```
## Not run:  
read_thermoraw(path)  
  
## End(Not run)
```

read_varian_peaklist	<i>Read 'Varian' peak list</i>
----------------------	--------------------------------

Description

Read peak list(s) from 'Varian MS Workstation'.

Usage

```
read_varian_peaklist(path)
```

Arguments

path	Path to 'Varian' peak list file.
------	----------------------------------

Value

A data.frame containing the information from the specified report.

Author(s)

Ethan Bass

See Also

Other 'Varian' parsers: [read_varian_sms\(\)](#)

Examples

```
## Not run:
read_varian_peaklist(path)

## End(Not run)
```

read_varian_sms	<i>Read 'Varian' SMS</i>
-----------------	--------------------------

Description

Reads 'Varian Workstation' SMS files.

Usage

```
read_varian_sms(
  path,
  what = c("MS1", "TIC", "BPC"),
  format_out = c("matrix", "data.frame", "data.table"),
  data_format = "long",
  read_metadata = TRUE,
  collapse = TRUE
)
```

Arguments

path	Path to a 'Varian' .SMS file.
what	Which streams to get: mass spectra (MS1), the total ion chromatogram (TIC) and/or the base peak chromatogram (BPC). Accepts multiple arguments. Defaults to all three.
format_out	Class of output. Either <code>matrix</code> , <code>data.frame</code> , or <code>data.table</code> .
data_format	Whether to return data in long (default) or wide format. Mass spectra are always returned in long format.
read_metadata	Logical. Whether to attach metadata. Defaults to TRUE.
collapse	Logical. Whether to collapse lists that only contain a single element. Defaults to TRUE.

Details

Varian SMS files begin with a DIRECTORY holding the offsets of each section. The first section is MSData, which begins at byte 3238 in every file seen so far, and is itself divided in two. The first part, after a short header, holds chromatogram data, one record per scan: the scan number, the retention time (as a 64-bit float), the ion time (μ sec, as a 2-byte unsigned integer), the total ion chromatogram (TIC), the base peak chromatogram (BPC), and further unidentified fields. The scan numbers and the TIC and BPC intensities are stored as 4-byte little-endian integers. A run of null bytes then separates this part from the segments holding the mass spectra.

The mass spectra are encoded differently. Each scan is a series of variable-length values, separated from the next scan by two null bytes. Within a scan the values are paired: the first of each pair is the delta-encoded mass-to-charge ratio and the second is the intensity. Each value is a big-endian integer whose length, $1 + (d \% 4)$ bytes, and bit width are set by its leading hexadecimal digit (d). Values beginning with 0-3 are single bytes. For $d \geq 4$, the lowest n bits are preserved according to the following scheme:

- d = 4-5 -> preserve lowest 13 bits
- d = 6-7 -> preserve lowest 14 bits
- d = 8-9 -> preserve lowest 21 bits
- d = 10-11 (A-B) -> preserve lowest 22 bits
- d = 12-13 (C-D) -> preserve lowest 27 bits
- d = 14-15 (E-F) -> preserve lowest 28 bits

No file seen so far carries a leading digit above C, so the rules for D and for E-F are extrapolated from the others rather than observed.

Value

A chromatogram or list of chromatograms from the specified file, according to the value of what. Chromatograms are returned in the format specified by format_out, except that mass spectra are returned as a data.table when format_out is matrix.

Note

There is still only limited support for the extraction of metadata from this file format.

Author(s)

Ethan Bass

See Also

Other 'Varian' parsers: [read_varian_peaklist\(\)](#)

Examples

```
## Not run:
read_varian_sms(path)

## End(Not run)
```

read_waters_arw	<i>Read 'Waters' ASCII (.arw)</i>
-----------------	-----------------------------------

Description

Reads 'Waters' ASCII .arw files.

Usage

```
read_waters_arw(  
  path,  
  format_out = c("matrix", "data.frame", "data.table"),  
  data_format = c("wide", "long"),  
  read_metadata = TRUE,  
  metadata_format = c("chromconverter", "raw")  
)
```

Arguments

path	Path to Waters .arw file.
format_out	Class of output. Either matrix, data.frame, or data.table.
data_format	Whether to return data in wide (default) or long format.
read_metadata	Logical. Whether to attach metadata. Defaults to TRUE.
metadata_format	Format to output metadata. Either chromconverter (standardized field names) or raw (vendor field names, unmapped).

Details

For help exporting files from Empower, you can consult the official documentation: [How_to_export_3D_raw_data_from_Empower](#)

Value

A chromatogram in the format specified by the format_out and data_format arguments.

Author(s)

Ethan Bass

See Also

Other 'Waters' parsers: [read_waters_raw\(\)](#)

Examples

```
## Not run:
read_waters_arw("path/to/file.arw")

## End(Not run)
```

read_waters_raw	<i>Read 'Waters' RAW</i>
-----------------	--------------------------

Description

Reads 'Waters MassLynx' (.raw) files into R.

Usage

```
read_waters_raw(
  path,
  format_out = c("matrix", "data.frame", "data.table"),
  data_format = c("wide", "long"),
  read_metadata = TRUE,
  metadata_format = c("chromconverter", "raw")
)
```

Arguments

path	Path to Waters .raw file.
format_out	Class of output. Either matrix, data.frame, or data.table.
data_format	Whether to return data in wide (default) or long format.
read_metadata	Logical. Whether to attach metadata. Defaults to TRUE.
metadata_format	Format to output metadata. Either chromconverter (standardized field names) or raw (vendor field names, unmapped).

Value

A list of 2D chromatograms, one per trace and named by it, in the format specified by format_out and data_format.

Note

This parser reads only 2D chromatograms (retention time and intensity), not mass spectra or DAD data.

Author(s)

Ethan Bass

See Also

Other 'Waters' parsers: [read_waters_arw\(\)](#)

Examples

```
## Not run:
read_waters_raw("path/to/file.raw")

## End(Not run)
```

subset.chrom_list	<i>Select chromatograms by their metadata</i>
-------------------	---

Description

Retains the chromatograms of a `chrom_list` whose metadata meet the specified condition, such as `sample_name == "blank"` or `run_datetime > as.POSIXct("2024-01-01", tz = "UTC")`. The condition can refer to any field [extract_metadata](#) reports, such as `sample_name` or `method`, as well as the chromatogram's name in the list.

Usage

```
## S3 method for class 'chrom_list'
subset(x, subset, ...)
```

Arguments

<code>x</code>	A <code>chrom_list</code> object.
<code>subset</code>	An expression giving a single TRUE or FALSE for each chromatogram.
<code>...</code>	Ignored.

Details

Numeric fields such as `time_range` and `sample_injection_volume` compare as numbers wherever the file's value parses as one, and a field with several values can be indexed (e.g. `time_range[2]`). A field a chromatogram does not carry is NA, and a chromatogram for which `subset` is NA is dropped, as in [subset\(\)](#) for data frames. Where an element holds several chromatograms, such as the traces [read_agilent_d](#) returns for each `.D` directory, it is kept or dropped as a whole, and `subset` sees the fields its chromatograms agree on, ignoring those that leave a field empty; a field they disagree on is NA.

Value

A `chrom_list` containing the selected chromatograms.

See Also

[extract_metadata](#)

Examples

```
path <- system.file("extdata", "benzoxazinoid_standards",
                    package = "chromConverter")
chroms <- read_chroms(path, format_in = "chemstation_ch", pattern = "dad1A",
                     parser = "chromconverter", progress_bar = FALSE)
subset(chroms, sample_name != "MEOH")
subset(chroms, run_datetime < as.POSIXct("2023-06-21", tz = "UTC"))
```

summary.chrom_list	<i>Summarize a chrom_list object</i>
--------------------	--------------------------------------

Description

Returns what [print.chrom_list](#) displays as a table, with one row per chromatogram: the sample it belongs to, its size, and the metadata fields in cols. Unlike print, nothing is collapsed into a header, abbreviated or truncated to the first few rows, so the result can be filtered and joined against.

Usage

```
## S3 method for class 'chrom_list'
summary(
  object,
  cols = chrom_summary_cols(),
  format_out = c("data.frame", "data.table", "tibble"),
  digits = NULL,
  ...
)
```

Arguments

object	A chrom_list object.
cols	Character vector of attribute names to report. Defaults to: sample_name, run_datetime, method, detector, wavelength, detector_range, scan_type, polarity, precursor_mz, product_mz, mz_range. A field that no chromatogram carries, or that all of them leave empty, is omitted rather than filled with NA.
format_out	Format of object. Either data.frame, data.table or tibble.
digits	Number of significant digits for the numbers in a field collapsed into a string, or NULL (the default) to keep them in full.
...	Additional arguments (currently ignored).

Value

A data.frame, data.table or tibble (according to the value of format_out) with one row per chromatogram. The first columns describe where the chromatogram sits and how large it is — sample, trace (only when a sample holds more than one), n_rows and n_cols — followed by one column per metadata field found. A field no chromatogram records, or that every one of them

leaves empty, is dropped rather than filled with NA. A field holding more than one value, such as the product_mz of an MRM event monitoring several transitions, is collapsed to a comma-separated string so that it occupies one column.

See Also

[extract_metadata](#), [print.chrom_list](#)

Examples

```
path <- system.file("extdata", "benzoxazinoid_standards",  
                    package = "chromConverter")  
chroms <- read_chroms(path, format_in = "chemstation_ch", pattern = "dad1A",  
                      parser = "chromconverter", progress_bar = FALSE)  
summary(chroms)
```

write_agilent_amx	<i>Write Agilent AMX method file</i>
-------------------	--------------------------------------

Description

Writes a copy of an 'OpenLab CDS' acquisition method (.amx) with a new pump gradient, flow, stop time, post time or column temperature. All other settings are copied unchanged from the template.

Usage

```
write_agilent_amx(  
  path,  
  path_out,  
  gradient = NULL,  
  flow = NULL,  
  stop_time = NULL,  
  post_time = NULL,  
  temperature = NULL,  
  force = FALSE  
)
```

Arguments

path	Path to the template .amx file.
path_out	Path to write the new method.
gradient	A table of solvent composition over time (see Details). NULL (default) keeps the template's gradient.
flow	Flow rate in mL/min. NULL (default) keeps the template's value.
stop_time	Pump stop time in minutes. NULL (default) keeps the template's value.

post_time	Pump post time in minutes, at least 0.01. 0 turns the post time off. NULL (default) keeps the template's value.
temperature	Column temperature in degrees Celsius. NULL (default) keeps the template's value.
force	Whether to overwrite path_out if it exists. Defaults to FALSE.

Details

The writer has only been tested on methods for a G7104C quaternary pump and a G7116A column compartment. Other modules may lay out their methods differently. If the writer fails on one of your methods, or writes one that OpenLab will not load, please [open an issue](#) and attach the template. Open each written method in OpenLab and check its settings before you run it.

gradient takes the table that `read_agilent_amx()` returns as `pump$gradient`, in wide or long format. It needs a percentage for every solvent channel the template uses, and each row must add up to 100. It must have a row at time zero, which sets the starting composition. The other rows replace the composition steps in the pump timetable. Flow steps in the timetable are kept, and gradient cannot change them. If the template has flow steps, drop the `flow_mL_min` column (or the "flow" rows in long format) from the table `read_agilent_amx()` returns.

Setting flow gives a constant flow rate and removes the template's flow steps.

temperature sets both sides of the column compartment, and the template must already control the temperature. A template whose compartment is set to "Not Controlled" gives an error.

Setting stop_time also sets the run time of the method. The checksum of each edited part is recalculated. The Agilent/ReportableInformation part, which repeats the settings as formatted text, is not rewritten and still describes the template.

The writer needs the 'zip' package.

Value

Invisibly returns `path_out`.

See Also

`read_agilent_amx()`

Other write functions: `write_andi_chrom()`, `write_andi_ms()`, `write_chroms()`, `write_mzml()`

Examples

```
## Not run:
gradient <- data.frame(time_min = c(0, 2, 8),
                      pct_A = c(70, 70, 5),
                      pct_B = c(30, 30, 95))
write_agilent_amx("template.amx", "new.amx", gradient = gradient,
                 stop_time = 10)

## End(Not run)
```

write_andi_chrom	<i>Write ANDI chrom CDF file from chromatogram</i>
------------------	--

Description

Exports a chromatogram in ANDI (Analytical Data Interchange) chromatography format (ASTM E1947-98). The format holds a single trace, so a 3D chromatogram must be reduced to one: name the column to export with the `lambda` argument, or the first column is exported with a warning.

Usage

```
write_andi_chrom(x, path_out, sample_name = NULL, lambda = NULL, force = FALSE)
```

Arguments

<code>x</code>	A chromatogram in (wide) format.
<code>path_out</code>	The path to write the file.
<code>sample_name</code>	The name of the file. If a name is not provided, the name will be derived from the <code>sample_name</code> attribute.
<code>lambda</code>	The wavelength to export, for a 3D chromatogram. Either a string matching one of the columns of <code>x</code> or the index of the column to export.
<code>force</code>	Whether to overwrite existing files at the specified path. Defaults to <code>FALSE</code> .

Details

Retention times are written in the unit the chromatogram reports, declared in the file's mandatory `retention_unit` attribute as `Minutes` or `Seconds`. A chromatogram whose `time_unit` is missing or unrecognized is taken to be in minutes, since the attribute cannot be left unset. The run length, delay time and sampling interval are derived from the retention times, and `detector_maximum_value` and `detector_minimum_value` report the range of the exported trace. `actual_sampling_interval` is the mean of the intervals, and `uniform_sampling_flag` reports whether every interval matches that mean. The per-point times are written to `raw_data_retention` either way, which is what a reader needs where the flag is `N`.

Value

Invisibly returns the path to the written CDF file.

Side effects

Exports a chromatogram in ANDI chromatography format (netCDF) in the directory specified by `path_out`. The file will be named according to the value of `sample_name`. If no `sample_name` is provided, the `sample_name` attribute will be used if it exists.

Author(s)

Ethan Bass

See Also

Other write functions: `write_agilent_amx()`, `write_andi_ms()`, `write_chroms()`, `write_mzml()`

Examples

```
path <- system.file("extdata/alkane_ladder.txt", package = "chromConverter")
chrom <- read_shimadzu(path, what = "chroms")
# the file is named for the `sample_name` attribute unless one is supplied
write_andi_chrom(chrom, path_out = tempdir(), force = TRUE)
```

write_andi_ms

Write ANDI MS CDF file from chromatogram

Description

Exports mass spectrometry data in ANDI (Analytical Data Interchange) MS format (ASTM E1947-98). A total ion chromatogram is derived from the MS1 scans if the supplied object does not already carry one.

Usage

```
write_andi_ms(
  x,
  path_out,
  sample_name = NULL,
  force = FALSE,
  ms_params = list()
)
```

Arguments

x	A list of chromatograms containing an element whose name includes MS (e.g. MS1), or a single chromatogram of MS1 scans, in wide or long format.
path_out	The path to write the file.
sample_name	The name of the file. If a name is not provided, the name will be derived from the sample_name attribute.
force	Whether to overwrite existing files at the specified path. Defaults to FALSE.
ms_params	A list of settings recorded in the file: ionization_mode, ionization_polarity, detector_type, scan_function and experiment_type. A setting that is not given is taken from the chromatogram's metadata where it records one: the polarity from its polarity attribute, the scan function from scan_type, and the ionization mode and detector type from its ms_params attribute. Failing that, the defaults are "Electron Impact", "Positive Polarity" and "Electron Multiplier", and a scan function is left out of the file. experiment_type is not read from metadata and defaults to "Centroided Mass Spectrum".

Details

Retention times are written in seconds, the only time unit the specification suggests, and declared in `raw_data_time_units`. A chromatogram is converted unless its `time_unit` already reports seconds, so one whose unit is missing or unrecognized is taken to be in minutes.

Value

Invisibly returns the path to the written CDF file.

Side effects

Exports mass spectrometry data in ANDI MS format (netCDF) in the directory specified by `path_out`. The file will be named according to the value of `sample_name`. If no `sample_name` is provided, the `sample_name` attribute will be used if it exists.

Author(s)

Ethan Bass

See Also

Other write functions: `write_agilent_amx()`, `write_andi_chrom()`, `write_chroms()`, `write_mzml()`

Examples

```
## Not run:
write_andi_ms(chrom, path_out = "path/to/directory")

## End(Not run)
```

write_chroms	<i>Write chromatograms</i>
--------------	----------------------------

Description

Writes chromatograms to disk in the format specified by `export_format`: either `mzml`, `cdf`, `csv`, or `arw`.

Usage

```
write_chroms(
  chrom_list,
  path_out,
  export_format = c("mzml", "cdf", "csv", "arw"),
  what = "",
  force = FALSE,
  show_progress = TRUE,
  verbose = getOption("verbose"),
```

```
    ...  
  )
```

Arguments

chrom_list	A list of chromatograms.
path_out	Path to directory for writing files.
export_format	Format to export files: either "mzml", "cdf", "csv", "arw".
what	What to write. Used by the cdf and mzml exporters only. For export_format = "cdf", either "MS1" (written by write_andi_ms) or "chrom" (written by write_andi_chrom). For export_format = "mzml", any of "MS1", "MS2", "TIC", "BPC" and/or "DAD". If it is not specified, the streams to write are inferred from the supplied data.
force	Logical. Whether to overwrite existing files. Defaults to FALSE.
show_progress	Logical. Whether to show progress bar. Defaults to TRUE.
verbose	Logical. Whether to print verbose output.
...	Additional arguments to write function.

Value

No return value. The function is called for its side effects.

Side effects

Exports a chromatogram in the file format specified by export_format in the directory specified by path_out.

Author(s)

Ethan Bass

See Also

Other write functions: [write_agilent_amx\(\)](#), [write_andi_chrom\(\)](#), [write_andi_ms\(\)](#), [write_mzml\(\)](#)

Examples

```
path <- system.file("extdata/alkane_ladder.txt", package = "chromConverter")  
chroms <- read_chroms(path, format_in = "shimadzu_ascii",  
                      find_files = FALSE, progress_bar = FALSE)  
write_chroms(chroms, path_out = tempdir(), export_format = "csv",  
             show_progress = FALSE)
```

write_mzml

*Write mzML***Description**

Writes spectra and chromatograms to an mzML file.

Usage

```
write_mzml(
  data,
  path_out,
  sample_name = NULL,
  what = NULL,
  instrument_info = NULL,
  centroided = TRUE,
  compress = TRUE,
  indexed = TRUE,
  force = FALSE,
  show_progress = TRUE,
  verbose = getOption("verbose")
)
```

Arguments

data	A named list of data.frames or data.tables, keyed by stream (MS1, MS2, TIC, BPC, DAD), or a single chromatogram carrying a detector attribute that says which stream it is.
path_out	The path to write the file.
sample_name	The name of the file. If a name is not provided, the name will be derived from the sample_name attribute, and it is an error if there is no such attribute.
what	Which streams to write to mzML: "MS1", "MS2", "TIC", "BPC", and/or "DAD". Defaults to every element of data that holds any rows.
instrument_info	Controlled-vocabulary terms describing the instrument, as a list of lists with elements cvRef, accession, name and value, each written as one cvParam of the instrumentConfiguration. Defaults to NULL, in which case MS:1000031 ("instrument model") is written with the chromatogram's detector_model or instrument as its value, or bare where it records neither.
centroided	Logical. Whether the spectra are centroided, written as MS:1000127 or, when FALSE, MS:1000128 ("profile spectrum"). Defaults to TRUE. Set it to FALSE for the profile scan types of a triple quadrupole (a full scan or a product-ion scan, as opposed to SIM or MRM).
compress	Logical. Whether to use zlib compression. Defaults to TRUE.
indexed	Logical. Whether to write indexed mzML. Defaults to TRUE.

<code>force</code>	Logical. Whether to overwrite existing files at <code>path_out</code> . Defaults to <code>FALSE</code> .
<code>show_progress</code>	Logical. Whether to show progress bar. Defaults to <code>TRUE</code> .
<code>verbose</code>	Logical. Whether or not to print status messages.

Details

Mass spectra and DAD spectra are written to the `spectrumList`, while the total ion current (TIC) and the base peak chromatogram (BPC) go to the `chromatogramList`, since the controlled vocabulary has terms for those two summaries. DAD spectra are written as electromagnetic radiation spectra (MS:1000804) using Thermo's naming convention, with `controllerType=4` in the spectrum ID for compatibility with existing tools.

Asking for both MS1 and MS2 writes them into one `spectrumList`, interleaved in acquisition order: on the scan column they share, or on retention time where neither has one. Each spectrum is then named for its scan (`scan=417`) rather than for its position in the list, which keeps the names unique across the levels. An MS2 spectrum carries the precursor it came from as MS:1000744 ("selected ion m/z"), and a `spectrumRef` to the MS1 spectrum that precedes it, and its collision energy as MS:1000045 where it is known, from the `scan_info` attribute or a `collision_energy` (or voltage) column. The isolation window and precursor charge are not written, as no parser in the package reads them.

MRM scans are written to the `chromatogramList` instead, as one selected reaction monitoring chromatogram (MS:1001473) per transition, with its precursor and product m/z as the targets of their isolation windows, as 'msconvert' writes them. A scan is taken to be MRM where the `scan_type` column of the `scan_info` attribute says so, or, where it records no type, where the scan holds more than one precursor_mz. Precursor or product m/z values that lie within 0.005 of the next value up are taken to be the same transition. Read them back with `read_mzml(what = "chroms")`.

Retention times are written in minutes (U0:0000031), as `chromConverter` reports them, rather than converted to seconds as [write_andi_ms](#) does.

The streams to write come from the names of data, so a bare chromatogram has to say what it holds through its detector attribute: UV and DAD are written as a DAD stream, and MS as MS1, or as MS2 where the table also carries an `ms_level` attribute above 1, or a `scan_info` attribute whose scans are all above level 1. Any other detector, including a missing or NA one (which is how several parsers report an unknown detector), is an error: the function stops rather than guess, and asks for a named list instead.

A one-dimensional DAD stream (a single wavelength) is refused: mzML has no axis to write it along, so it would become one single-point spectrum per retention time. Use [write_andi_chrom](#) for a single trace. If it is the only stream requested this is an error; otherwise it is dropped with a warning and the rest is written.

The file's metadata are taken from the MS1 stream if it is written, and otherwise from the first stream requested. That stream's `sample_name` attribute names the file unless `sample_name` is supplied.

If `indexed = TRUE`, the function will generate an indexed mzML file, which allows faster random access to spectra. The file is assembled by writing XML strings straight to a connection rather than by building a document in memory.

Value

Invisibly returns the path to the written mzML file.

Author(s)

Ethan Bass

See Also

Other write functions: [write_agilent_amx\(\)](#), [write_andi_chrom\(\)](#), [write_andi_ms\(\)](#), [write_chroms\(\)](#)

Examples

```
## Not run:  
chrom <- read_chroms("path/to/file.qgd", progress_bar = FALSE)  
write_mzml(chrom[[1]], path_out = "path/to/directory")  
  
## End(Not run)
```

Index

- * **'Agilent' parsers**
 - read_agilent_d, [18](#)
 - read_agilent_dx, [20](#)
 - read_agilent_rslt, [22](#)
 - read_chemstation_ch, [27](#)
 - read_chemstation_csv, [28](#)
 - read_chemstation_logs, [29](#)
 - read_chemstation_method, [31](#)
 - read_chemstation_ms, [33](#)
 - read_chemstation_reports, [34](#)
 - read_chemstation_uv, [36](#)
- * **'Shimadzu' parsers**
 - read_shimadzu, [49](#)
 - read_shimadzu_gcd, [50](#)
 - read_shimadzu_lcd, [52](#)
 - read_shimadzu_method, [56](#)
 - read_shimadzu_qgd, [59](#)
- * **'Varian' parsers**
 - read_varian_peaklist, [62](#)
 - read_varian_sms, [63](#)
- * **'Waters' parsers**
 - read_waters_arw, [65](#)
 - read_waters_raw, [66](#)
- * **datasets**
 - alkane_ladder, [4](#)
 - benzoxazinoid_standards, [5](#)
- * **external parsers**
 - call_entab, [5](#)
 - call_openchrom, [7](#)
 - call_rainbow, [9](#)
 - read_thermoraw, [61](#)
- * **write functions**
 - write_agilent_amx, [69](#)
 - write_andi_chrom, [71](#)
 - write_andi_ms, [72](#)
 - write_chroms, [73](#)
 - write_mzml, [75](#)
- add_metadata, [3](#)
- alkane_ladder, [4](#)
- attributes, [20, 25, 28, 34, 36, 52, 54, 60](#)
- benzoxazinoid_standards, [5](#)
- call_entab, [5](#)
- call_entab(), [8, 10, 62](#)
- call_openchrom, [7, 11, 41](#)
- call_openchrom(), [6, 10, 62](#)
- call_rainbow, [9](#)
- call_rainbow(), [6, 8, 53, 62](#)
- configure_openchrom, [11](#)
- extract_metadata, [3, 4, 12, 14, 15, 43, 45, 67, 69](#)
- get_element, [13](#)
- makeCluster, [16, 23, 41, 44, 48](#)
- pbapply, [16, 23, 41, 44, 48](#)
- print.chrom_list, [14, 68, 69](#)
- print.data.frame, [15](#)
- RaMS: :grabMSdata, [46](#)
- read.csv, [8](#)
- read_acaml, [15](#)
- read_agilent_amx, [16, 31–33, 58](#)
- read_agilent_amx(), [70](#)
- read_agilent_d, [3, 5, 13, 18, 33, 67](#)
- read_agilent_d(), [22, 23, 28, 29, 31, 33–35, 37](#)
- read_agilent_dx, [19, 20](#)
- read_agilent_dx(), [20, 23, 28, 29, 31, 33–35, 37](#)
- read_agilent_rslt, [16, 22](#)
- read_agilent_rslt(), [20, 22, 28, 29, 31, 33–35, 37](#)
- read_asm, [24](#)
- read_cdf, [8, 25](#)
- read_chemstation_ch, [21, 27](#)

`read_chemstation_ch()`, 20, 22, 23, 29, 31, 33–35, 37
`read_chemstation_csv`, 28
`read_chemstation_csv()`, 20, 22, 23, 28, 31, 33–35, 37
`read_chemstation_logs`, 29
`read_chemstation_logs()`, 20, 22, 23, 28, 29, 33–35, 37
`read_chemstation_method`, 31, 58
`read_chemstation_method()`, 20, 22, 23, 28, 29, 31, 34, 35, 37
`read_chemstation_ms`, 33
`read_chemstation_ms()`, 20, 22, 23, 28, 29, 31, 33, 35, 37
`read_chemstation_reports`, 34
`read_chemstation_reports()`, 20, 22, 23, 28, 29, 31, 33, 34, 37
`read_chemstation_uv`, 21, 36
`read_chemstation_uv()`, 20, 22, 23, 28, 29, 31, 33–35
`read_chromatotec`, 37
`read_chromeleon`, 38
`read_chroms`, 13, 14, 16, 23, 25, 39, 43
`read_mdf`, 42
`read_metadata`, 43
`read_mzml`, 8, 46, 61
`read_peaklist`, 19, 23, 41, 47
`read_shimadzu`, 49
`read_shimadzu()`, 52, 55, 58, 60
`read_shimadzu_gcd`, 50
`read_shimadzu_gcd()`, 50, 55, 58, 60
`read_shimadzu_lcd`, 52
`read_shimadzu_lcd()`, 50, 52, 58, 60
`read_shimadzu_method`, 56
`read_shimadzu_method()`, 50, 52, 55, 60
`read_shimadzu_qgd`, 59
`read_shimadzu_qgd()`, 50, 52, 55, 58
`read_sz_ittof()`, 54
`read_sz_ittof_tic()`, 54
`read_sz_lcd_2d()`, 50, 52, 54, 55, 58, 60
`read_sz_lcd_3d()`, 50, 52, 54, 55, 58, 60
`read_sz_qtof()`, 54
`read_sz_qtof_tic()`, 54
`read_sz_sq()`, 54
`read_sz_sq_tic()`, 54
`read_sz_tables()`, 50, 52, 54, 55, 58, 60
`read_sz_tlm()`, 54
`read_thermoraw`, 61
`read_thermoraw()`, 6, 8, 10
`read_varian_peaklist`, 62
`read_varian_peaklist()`, 64
`read_varian_sms`, 63
`read_varian_sms()`, 63
`read_waters_arw`, 65
`read_waters_arw()`, 67
`read_waters_raw`, 66
`read_waters_raw()`, 65

`sp_converter()`, 6, 8, 10, 62
`subset()`, 67
`subset.chrom_list`, 3, 4, 67
`summary.chrom_list`, 68

`uv_converter()`, 6, 8, 10, 62

`write_agilent_amx`, 69
`write_agilent_amx()`, 72–74, 77
`write_andi_chrom`, 71, 76
`write_andi_chrom()`, 70, 73, 74, 77
`write_andi_ms`, 72, 76
`write_andi_ms()`, 70, 72, 74, 77
`write_chroms`, 73
`write_chroms()`, 70, 72, 73, 77
`write_mzml`, 75
`write_mzml()`, 70, 72–74