Package ‘isoread’

May 28, 2014

Type Package

Title Read IRMS data from isodat files.

Version 0.1

Date 2014-03-17

Author Sebastian Kopf

Maintainer Sebastian Kopf <seb.kopf@gmail.com>

Description R interface for accessing isotope ratio mass spectrometry data stored in isodat files.

License GPL-2

LazyLoad yes

Depends plyr,reshape2

Suggests testthat,isotopia,ggplot2,gridExtra

Roxygen list(wrap = FALSE)

Collate 'BinaryFileClass.R' 'IrmsDataClass.R' 'IrmsContinuousFlowDataClass.R' 'IrmsDualInletDataClass.R' 'IsodatFileClass.R' 'IsodatHydrogenContinuousFlowFileClass.R' 'export.R' 'isoread.R' 'utilities.R' 'zzz.R'

R topics documented:

`isoread-package` *isoread package*

Description

R interface to IRMS (isotope ratio mass spectrometry) file formats typically used in stable isotope geochemistry.

Details

See `isoread` for details on how to use.

Author(s)

Sebastian Kopf

`BinaryFile` *Binary File reference class*

Description

Binary File reference class

Fields

`filepath` stores the path to the binart file

`filename` stores the filename

`creation_date` stores the date the file was created (if it could be retrieved, which is not always the case when running on linux but no problem on OS X and windows)

`rawdata` this is the binary raw data from the file (typically removed during cleanup unless clean_raw = FALSE)

`keys` these are the Unicode and ASCII text fragments found in the binary file, they are used for navigating in the file when pulling out the relevant data (typically removed during cleanup unless clean_keys = FALSE)

`data` a list that contains all the actual data pulled from the file

Methods

`clean_keys(removeText = NULL, removePattern = NULL, unlessByteLength = 0, unlessText = NULL)` clean up keys by removing randomly found strings that are clearly not proper targets

`cleanup(clean_raw = TRUE, clean_keys = TRUE, ...)` clean up the object by removing the raw data and keys (and other large but only transiently important information) from memory

`find_key(pattern, occurence = 1)` find a key by a regexp pattern

`find_keys(asciiL = 10, unicodeL = 5)` finds all unicode and ascii strings and stores them for navigation around the file

`get_info(show = c())` Get basic information about the object `initialize(file, ...)` initialize BinaryFile object, requires a file path

`load(...)` load the data from the file and generate key lookup

`move_to_key(key, occurence = 1)` moves position to the end of a specific occurence of a key (use -1 for last occurence)

`parse(type, length = 1, id = NA, skip_first = 0)` parse binary data at current position in the data stream advances pointer by the size of the read data

#' @param type see `map_binary_data_type` #' @param length see `parse_binary_data` #' @param id if provided, will store the parsed data with this key in the `$data` field #' @param skip_first how many bytes to skip before reading this

`parse_array(types, n, id = NA, skip_first = 0)` repeatedly read the same set of information into a data frame

#' @param types a named vector of data types (for data types see `parse_binary_data`), #' the names are used for the columns of the resulting data frame #' @param id if provided, will store the parsed data with this key in the $data field #' @param n length of array #' @param skip_first how many bytes to skip before reading this

`process(...)` process the raw data to fill the data list

`read_file()` read the binary file

#' @note this does not work for very large files probably because of the 2^31-1 #' limit on vector size! think about ways to fix this... #' –> might have to acually read directly from the conection instead of the raw data buffer!

`skip(nbyte)` skip nbyte number of bytes in the raw data stream

`export_data` *Convenience function to export data from multiple IrmsData objects of the same class into a comma-separated value file.*

Description

Convenience function to export data from multiple IrmsData objects of the same class into a comma-separated value file.

Usage

```
export_data(data, file = "irms_data_export.csv", ...)
```

`IrmsContinuousFlowData` *IrmsContinuousFlowData reference class*

Description

IrmsContinuousFlowData reference class

Fields

`chromData` stores the chromatographic data (the actual mass and ratio data traces),

`peakTable` stores the peak table (detected peaks and all their information)

`peakTableColumns` stores the definition of which columns exist in the peak table and what their proper data types are

`peakTableKeys` stores information about which columns correspond to key elements of the peakTable (e.g. the peak number, retention time and compound name)

Methods

`check_chrom_data(masses = names(.self$plotOptions$masses), ratios = names(.self$plotOptions$rati`
checks the consistency of the chromatographic data, by default checks for all masses and ratios

`check_data(...)` check the data consistency, calls `check_crom_data` and `check_peak_table`

`check_peak_table(..., warn = TRUE)` checks the consistency of the peak table and converts data types if necessary

`export_data(file, ...)` export the data stored in this object to file

`get_mass_data(masses = names(.self$plotOptions$masses), melt = FALSE)` get the mass trace data for specific masses, can be provided in `melt = TRUE` format for easy use in ggplot style plotting

`get_peak(peak_nr, select = names(peakTable))` retrieve information for a peak in the peak table (identified by peak_nr), can specify which columns to retrieve with `selec`, retrieves all columns by default

`get_peak_by_name(names, select = names(peakTable))` retrieve information for peak(s) in the peak table (identified by names)

`get_peak_by_rt(rts, select = names(peakTable))` retrieve information for peak(s) in the peak table (identified by retention times)

`get_peak_nr_by_name(names)` find peak numbers (i.e. ids) by name(s), returns a vector of found peak numbers (integer(0) if none found)

`get_peak_nr_by_rt(rts)` find peak numbers (i.e. ids) by retention time(s), returns a vector of found peak numbers (integer(0) if none found)

`get_peak_table(type = c("ref", "data", "both"))` retrieve the peak table

`get_ratio_data(ratios = names(.self$plotOptions$ratios), melt = FALSE)` get the ratio trace data for specific ratios, can be provided in `melt = TRUE` format for easy use in ggplot style plotting

`ggplot(tlim = NULL, tunits = .self$plotOptions$tunits$labels[[.self$plotOptions$tunits$value]],`
ggplot the data

#' @param tlim time range (in tunits units)

#' @param tunits units (currently 's' or 'min')

#' @param masses vector of the masses to plot (if NULL, panel excluded)

#' @param ratios vector of the ratios to plot (if NULL, panel excluded)

`ggplot(...)` generate a ggplot object for the data in this IrmsData object

`identify_peaks(rts, compounds)` Identify peaks by mapping compound names to retention times

`init_irms_data()` initialize irms data container `map_peaks(map)` Add information to peaks by mapping properties from a data frame that contains at least the defined peak number (e.g. 'Peak Nr.') or retention time (Rt) as a column. Additional columns (other than peak nr and retention time) are mapped to the relevant peaks if they correspond to existing columns, otherwise they are disregarded with a warning.

Note: make sure to have the data.frame that is passed in set with `stringsAsFactors = F` (usually the desired setting for the mapping)

`plot(tlim = NULL, mass_ylim = NULL, ratio_ylim = NULL, masses = names(.self$plotOptions$masses),`
Plot the data (both masses and ratios) - much faster than ggplot but not as versatile

#' @param tlim time range, should be in the same tunits

#' @param masses which masses to plot (all defined in plot optinos by default)

#' @param ratios which ratios to plot (all defined in plot options by default)

#' @param tunits time units, as defined in tunits (currently either 's' or 'min'), takes the one set in plotOptions as default

`plot_data(y, ylab = "", title = "data peaks")` plot the data of the actual sample peaks, see `plot_peak_table` for details on syntax

`plot_masses(tlim = NULL, ylim = NULL, masses = names(.self$plotOptions$masses), tunits = .self$p`
Plot the masses (this if much faster than ggplot but not as versatile)

`plot_peak_table(y = NULL, ylab = "", title = "", data = get_peak_table())` Plot the data points in the peak table

#' @param y = expression which data to plot (will be evaluated in context of the data frame)

#' @param ylab = y axis label

#' @param title = title of the plot

#' @param data = peak table data (by default the whole peak table)

`plot_ratios(tlim = NULL, ylim = NULL, ratios = names(.self$plotOptions$ratios), tunits = .self$p`
Plot the ratios (this if much faster than ggplot but not as versatile)

`plot_refs(y, ylab = "", title = "references")` plot the data of the reference peaks, see `plot_peak_table` for details on syntax

`reevaluate_peak_table()` reevalutes the peak table (not currently implemented)

`set_plot_options(...)` set plot options

`set_ref_peaks(rts, set = TRUE, reevaluate = FALSE)` Identify peaks (by their retention times) as reference peaks (or remove their status as a reference peak)

`summarize(file, ....)` summarize the data stored in this object and save it to file

See Also

IrmsData, IrmsDualInletData

`IrmsData` *IrmsData reference class*

Description

IrmsData reference class

Fields

`plotOptions` holds information about default plotting options

Methods

`export_data(file, ...)` export the data stored in this object to file

`ggplot(...)` generate a ggplot object for the data in this IrmsData object

`init_irms_data()` initialize irms data container

`set_plot_options(...)` set plot options

`summarize(file, ....)` summarize the data stored in this object and save it to file

---

`IrmsDualInletData` *IrmsDualInletData reference class*

Description

IrmsDualInletData reference class

Methods

`export_data(file, ...)` export the data stored in this object to file

`ggplot(...)` generate a ggplot object for the data in this IrmsData object

`init_irms_data()` initialize irms data container

`set_plot_options(...)` set plot options

`summarize(file, ....)` summarize the data stored in this object and save it to file

Note

not implemented yet for any actual data reading

See Also

IrmsData, IrmsContinuousFlowData

`IsodatFile` *Isodat file class*

Description

Class representing an isodat binary file.

Methods

`cleanup(clean_raw = TRUE, clean_keys = TRUE, ...)` clean up the object by removing the raw data and keys (and other large but only transiently important information) from memory

`find_key(pattern, occurence = 1)` find a key by a regexp pattern

`get_info(show = c())` Get basic information about the object

See Also

BinaryFile

`IsodatHydrogenContinuousFlowFile` *H-CSIA DataClass*

Description

Objects of this class hold the isotopic data from compound specific hydrogen isotope analysis recorded in Isodat file formats (currently supported isodat version is 2.0 for chromatographic and peak table data and isodat version 2.5 and 3.0 for chromatographic data only).

Details

This class is derived from IrmsContinuousFlowData which defines a number of useful plotting, export and data access methods. This class also derived BinaryFile which provides functionality for interacting with the underlying IsodatFile.

Methods

`cleanup(clean_raw = TRUE, clean_keys = TRUE, ...)` clean up the object by removing the raw data and keys (and other large but only transiently important information) from memory

`find_key(pattern, occurence = 1)` find a key by a regexp pattern

`get_info(show = c())` Get basic information about the object

`initialize(file, ...)` initialize BinaryFile object, requires a file path

`plot_data(y, ylab = "", title = "data peaks")` plot the data of the actual sample peaks, see `plot_peak_table` for details on syntax

`plot_refs(y, ylab = "", title = "references")` plot the data of the reference peaks, see `plot_peak_table` for details on syntax

`process(...)` process the raw data to fill the data list

`reevaluate_peak_table()` reevalutes the peak table (not currently implemented)

See Also

BinaryFile, IsodatFile, IrmsContinuousFlowData, IrmsData

`isoread` *Read isotope data files*

Description

Reads isodat file(s) and returns the contents as file type specific instances of `BinaryFile` / `IrmsDataClass` (extends both).

Usage

```
isoread(files, type, load_chroms = T, ...)
```

Arguments

`file`	path to the file(s) to read
`type`	type of the files to be read • 'H_CSIA' = compound specific IRMS data for hydrogen isotopes
`load_chroms`	whether to keep the chromatograms in the objects (otherwise only peak tables are kept)
`...`	parameters passed to the `load` and `process` functions of the IsodatFile objects

Value

List of file `type` specific objects.

- 'H_CSIA' = instance(s) of `IsodatHydrogenContinuousFlowFile` which implements `IrmsContinuousFlowData`.

If file names start with a number, then the number is used as key in the list, otherwise the whole filename is the key. If there is only one file, the object is returned directly.

`isoread_folder` *Reads all isodat files in a folder.*

Description

See `isoread` for paramter and return value details.

Usage

```
isoread_folder(folder, type, extension = ".cf", ...)
```

`map_binary_data_type` *Binary data type mapping*

Description

Maps binary C data types to proper R data types and byte lengths

Usage

```
map_binary_data_type(type = c("binary", "UTF8", "UTF16", "UTF32", "short",
  "long", "long long", "float", "double"))
```

Arguments

`type`	• 'binary' = raw with 1 byte (raw data) • 'UTF8' = character with 1 byte (ascii) • 'UTF16' = character with 2 bytes (unicode) • 'UTF32' = character with 4 bytes (unicode) • 'short' = integer with 2 bytes (16bit) • 'long' = integer with 4 bytes (32bit) • 'longlong' = integer with 8 bytes (64bit) • 'float' = numeric with 4 bytes (32bit) • 'double' = numeric with 8 bytes (64bit)

Note

implemented signed int and complex if needed

`map_peaks` *Map peak table*

Description

Map peak table data of IrmsContinuousFlowData object(s) based on a data frame or input excel file.

Usage

```
map_peaks(iso, map, startRow = 3, libfile = NULL,
  colClasses = c("numeric", "character", "character"), ...)
```

Arguments

`iso`	IrmsContinuousFlowData object(s)
`map`	either a data frame with a map (containing column Rt and Component) or the file path to a mapping file, extension determines how it will be processed currently only xlsx and xls are supported. Excel file must include column headers on the indicated row (default startRow = 3) with columns Rt and Component Component
`libfile`	name of the library file, also only xlsx and xls currently supported if provided, will attempt to merge the components in the mapping file with the library information for additional details on Formula and other compound properties

---

`parse_binary_data` *Wrapper for parsing binary data.*

Description

Convenience wrapper for parsing binary data. For more details on reading binary data, check ?readBin

Usage

```
parse_binary_data(data, type, length = 1)
```

Arguments

`type`	data type see `map_binary_data_type` for details
`length`	how many instances of this object (for characters and raw this means length of string, all others a vector)

Value

read data

`quickview` *File quickview*

Description

This functions serves to gain a quick view of a loaded isodat file. It shows the masses plot and prints a minimal subset of the peak table. Optionally reloads the file (tries to keep the peak definitions).

Usage

```
quickview(iso, reload = FALSE, show = c("Peak Nr.", "Status", "Ref. Peak",
  "Component", "Rt", "Start", "End", "Ampl. 2", "d 2H/1H"))
```

Arguments

`iso`	a single isodat file obj
`reload`	whether to reload the file (this is forced if there is no chromatographic data)
`show`	list of peak table columns to show

`reload` *Reload an isodat file object.*

Description

Reload an existing isodat object with all the chromatographic data and resets the peak table if keep_peaks = FALSE. Good for interrogation of an individual file. Requires the original file to still be in the same location.

Usage

```
reload(iso, remap_peaks = TRUE, load_chroms = TRUE)
```

Arguments

`iso`	the object to reload (can be a list)
`remap_peaks`	whether to keep the peak identification or not
`load_chroms`	whether to load the chroms (much smaller object without)

Value

the reloaded obj (or list of objs)

Note

currently only for type = "H_CSIA"

`summarize_all` *Summarize a collection of IrmsData objects*

Description

Summarize a collection of IrmsData objects

Usage

```
summarize_all(iso, ...)
```

Arguments

`iso`	IrmsData object(s)
`...`	all passed to summarize

Index