

# Package: lightr (via r-universe)

July 11, 2024

**Title** Read Spectrometric Data and Metadata

**Version** 1.7.1.9000

**Description** Parse various reflectance/transmittance/absorbance spectra file formats to extract spectral data and metadata, as described in Gruson, White & Maia (2019) <doi:10.21105/joss.01857>. Among other formats, it can import files from 'Avantes' <<https://www.avantes.com/>>, 'CRAIC' <<https://www.microspectra.com/>>, and 'OceanInsight' (formerly 'OceanOptics') <<https://www.oceaninsight.com/>> brands.

**Depends** R (>= 4.0.0)

**Imports** future.apply, progressr, xml2 (>= 1.0.0)

**Suggests** digest, future, knitr, lintr, pavo, rmarkdown, spelling, testthat (>= 3.0.0), withr

**URL** <https://docs.ropensci.org/lightr/>,  
<https://github.com/ropensci/lightr>

**BugReports** <https://github.com/ropensci/lightr/issues>

**License** GPL (>=2)

**RoxygenNote** 7.3.1

**Roxygen** list(markdown = TRUE)

**Language** en-GB

**VignetteBuilder** knitr

**Config/testthat/edition** 3

**Repository** <https://ropensci.r-universe.dev>

**RemoteUrl** <https://github.com/ropensci/lightr>

**RemoteRef** main

**RemoteSha** ff286dc09c21e25997ebf1486ae514eb24eb1ca9

## Contents

lr_convert_tocsv . . . . .	2
lr_get_metadata . . . . .	3
lr_get_spec . . . . .	5
lr_parse_generic . . . . .	6
lr_parse_jaz . . . . .	7
lr_parse_jdx . . . . .	9
lr_parse_procspec . . . . .	10
lr_parse_spc . . . . .	11
lr_parse_trm . . . . .	12
lr_parse_ttt . . . . .	14

<b>Index</b>	<b>16</b>
--------------	-----------

---

lr_convert_tocsv	<i>Convert spectral data files to csv files</i>
------------------	---

---

### Description

Convert spectral data files to csv files

### Usage

```
lr_convert_tocsv(
  where = NULL,
  ext = "txt",
  decimal = ".",
  sep = NULL,
  subdir = FALSE,
  ignore.case = TRUE,
  overwrite = FALSE,
  metadata = TRUE
)
```

### Arguments

where	Folder in which files are located (defaults to current working directory).
ext	File extension to be searched for, without the "." (defaults to txt). You can also use a character vector to specify multiple file extensions.
decimal	Character to be used to identify decimal plates (defaults to .).
sep	Column delimiting characters to be considered in addition to the default (which are: tab, space, and ";")
subdir	Should subdirectories within the where folder be included in the search? (defaults to FALSE).
ignore.case	Should the extension search be case insensitive? (defaults to TRUE)

overwrite	logical. Should the function overwrite existing files with the same name? (defaults to FALSE).
metadata	logical (defaults to TRUE). Should metadata be exported as well? They will be exported in csv files with the <code>_metadata.csv</code> suffix.

### Details

You can customise the type of parallel processing used by this function with the `future::plan()` function. This works on all operating systems, as well as high performance computing (HPC) environment. Similarly, you can customise the way progress is shown with the `progressr::handlers()` functions (progress bar, acoustic feedback, nothing, etc.)

### Value

Convert input files to csv and invisibly return the list of created file paths

### Warning

When `metadata = TRUE`, if **either** the data **or** metadata export fails, nothing will be returned for this file.

---

lr_get_metadata	<i>Extract metadata from spectra files</i>
-----------------	--

---

### Description

Finds and imports metadata from spectra files in a given location.

### Usage

```
lr_get_metadata(
  where = getwd(),
  ext = "ProcSpec",
  sep = NULL,
  subdir = FALSE,
  subdir.names = FALSE,
  ignore.case = TRUE
)
```

### Arguments

where	Folder in which files are located (defaults to current working directory).
ext	File extension to be searched for, without the "." (defaults to txt). You can also use a character vector to specify multiple file extensions.
sep	Column delimiting characters to be considered in addition to the default (which are: tab, space, and ";")

subdir	Should subdirectories within the where folder be included in the search? (defaults to FALSE).
subdir.names	Should subdirectory path be included in the name of the spectra? (defaults to FALSE).
ignore.case	Should the extension search be case insensitive? (defaults to TRUE)

### Details

You can customise the type of parallel processing used by this function with the `future::plan()` function. This works on all operating systems, as well as high performance computing (HPC) environment. Similarly, you can customise the way progress is shown with the `progressr::handlers()` functions (progress bar, acoustic feedback, nothing, etc.)

### Value

A data.frame containing one file per row and the following columns:

- name: File name (without the extension)
- user: Name of the spectrometer operator
- datetime: Timestamp of the recording (ISO 8601 format)
- spec\_model: Model of the spectrometer
- spec\_ID: Unique ID of the spectrometer
- white\_inttime: Integration time of the white reference (in ms)
- dark\_inttime: Integration time of the dark reference (in ms)
- sample\_inttime: Integration time of the sample (in ms)
- white\_avgs: Number of averaged measurements for the white reference
- dark\_avgs: Number of averaged measurements for the dark reference
- sample\_avgs: Number of averaged measurements for the sample
- white\_boxcar: Boxcar width for the white reference
- dark\_boxcar: Boxcar width for the dark reference
- sample\_boxcar: Boxcar width for the sample reference

### Warning

white\_inttime, dark\_inttime and sample\_inttime should be equal. The normalised data may be inaccurate otherwise.

### References

White TE, Dalrymple RL, Noble DWA, O'Hanlon JC, Zurek DB, Umbers KDL. Reproducible research in the study of biological coloration. *Animal Behaviour*. 2015 Aug 1;106:51-7 ([doi:10.1016/j.anbehav.2015.05.007](https://doi.org/10.1016/j.anbehav.2015.05.007)).

**Examples**

```
lr_get_metadata(system.file("testdata", "procspec_files",
                           package = "lightr"),
               ext = "ProcSpec")
```

---

lr_get_spec	<i>Extract spectral data from spectra files</i>
-------------	---

---

**Description**

Finds and imports reflectance/transmittance/absorbance data from spectra files in a given location.

**Usage**

```
lr_get_spec(
  where = getwd(),
  ext = "txt",
  lim = c(300, 700),
  decimal = ".",
  sep = NULL,
  subdir = FALSE,
  subdir.names = FALSE,
  ignore.case = TRUE,
  interpolate = TRUE
)
```

**Arguments**

where	Folder in which files are located (defaults to current working directory).
ext	File extension to be searched for, without the "." (defaults to txt). You can also use a character vector to specify multiple file extensions.
lim	A vector with two numbers determining the wavelength limits to be considered (defaults to c(300, 700)).
decimal	Character to be used to identify decimal plates (defaults to .).
sep	Column delimiting characters to be considered in addition to the default (which are: tab, space, and ";")
subdir	Should subdirectories within the where folder be included in the search? (defaults to FALSE).
subdir.names	Should subdirectory path be included in the name of the spectra? (defaults to FALSE).
ignore.case	Should the extension search be case insensitive? (defaults to TRUE)
interpolate	Boolean indicated whether spectral data should be interpolated and pruned at every nanometre. Note that this option can only work if all input data samples the same wavelengths. Defaults to TRUE.

**Details**

You can customise the type of parallel processing used by this function with the `future::plan()` function. This works on all operating systems, as well as high performance computing (HPC) environment. Similarly, you can customise the way progress is shown with the `progressr::handlers()` functions (progress bar, acoustic feedback, nothing, etc.)

**Value**

A data.frame, containing the wavelengths in the first column and individual imported spectral files in the subsequent columns.

**See Also**

`pavo::getspec()`

**Examples**

```
spcs <- lr_get_spec(system.file("testdata", package = "lightr"), ext = "jdx")
head(spcs)
```

---

lr_parse_generic	<i>Generic function to parse spectra files that don't have a specific parser</i>
------------------	--

---

**Description**

Generic function to parse spectra files that don't have a specific parser

**Usage**

```
lr_parse_generic(filename, decimal = ".", sep = NULL)
```

**Arguments**

filename	Path of the file to parse
decimal	Character to be used to identify decimal plates (defaults to .).
sep	Column delimiting characters to be considered in addition to the default (which are: tab, space, and ";")

**Details**

'processed' column computed by official software and provided as is.

**Value**

A named list of two elements:

- data: a dataframe with columns "wl", "dark", "white", "scope" and "processed", in this order.
- metadata: a character vector with metadata including:
  - user: Name of the spectrometer operator
  - datetime: Timestamp of the recording in format '%Y-%m-%d %H:%M:%S' and UTC timezone. If timezone is missing in source file, UTC time will be assumed (for reproducibility purposes across computers with different localtimes).
  - spec\_model: Model of the spectrometer
  - spec\_ID: Unique ID of the spectrometer
  - white\_inttime: Integration time of the white reference (in ms)
  - dark\_inttime: Integration time of the dark reference (in ms)
  - sample\_inttime: Integration time of the sample (in ms)
  - white\_avgs: Number of averaged measurements for the white reference
  - dark\_avgs: Number of averaged measurements for the dark reference
  - sample\_avgs: Number of averaged measurements for the sample
  - white\_boxcar: Boxcar width for the white reference
  - dark\_boxcar: Boxcar width for the dark reference
  - sample\_boxcar: Boxcar width for the sample reference

**Examples**

```
res_csv <- lr_parse_generic(  
  system.file("testdata", "spec.csv", package = "lightr"),  
  sep = ","  
)  
head(res_csv$data)  
# No metadata is extracted with this parser  
res_csv$metadata  
  
res_craic <- lr_parse_generic(  
  system.file("testdata", "CRAIC_export.txt", package = "lightr")  
)  
head(res_craic$data)  
# No metadata is extracted with this parser  
res_craic$metadata
```

---

lr\_parse\_jaz

*Parse OceanInsight converted file*

---

**Description**

Parse OceanInsight (formerly OceanOptics) converted file. <https://www.oceaninsight.com/>

**Usage**

```
lr_parse_jaz(filename)
```

```
lr_parse_jazirrad(filename)
```

**Arguments**

filename            Path of the file to parse

**Details**

'processed' column computed by official software and provided as is.

**Value**

A named list of two elements:

- data: a dataframe with columns "wl", "dark", "white", "scope" and "processed", in this order.
- metadata: a character vector with metadata including:
  - user: Name of the spectrometer operator
  - datetime: Timestamp of the recording in format '%Y-%m-%d %H:%M:%S' and UTC timezone. If timezone is missing in source file, UTC time will be assumed (for reproducibility purposes across computers with different localtimes).
  - spec\_model: Model of the spectrometer
  - spec\_ID: Unique ID of the spectrometer
  - white\_inttime: Integration time of the white reference (in ms)
  - dark\_inttime: Integration time of the dark reference (in ms)
  - sample\_inttime: Integration time of the sample (in ms)
  - white\_avgs: Number of averaged measurements for the white reference
  - dark\_avgs: Number of averaged measurements for the dark reference
  - sample\_avgs: Number of averaged measurements for the sample
  - white\_boxcar: Boxcar width for the white reference
  - dark\_boxcar: Boxcar width for the dark reference
  - sample\_boxcar: Boxcar width for the sample reference

**Examples**

```
res_jaz <- lr_parse_jaz(system.file("testdata", "jazspec.jaz",
                                   package = "lightr"))
head(res_jaz$data)
res_jaz$metadata

res_jazirrad <- lr_parse_jazirrad(system.file("testdata", "irrad.JazIrrad",
                                             package = "lightr"))
head(res_jazirrad$data)
res_jazirrad$metadata

res_usb4000 <- lr_parse_jaz(system.file("testdata", "00usb4000.txt",
```



```
                                package = "lightr"))
head(res_usb4000$data)
res_usb4000$metadata

res_transmission <- lr_parse_jaz(
  system.file("testdata", "FMNH6834.00000001.Master.Transmission",
             package = "lightr")
)
head(res_transmission$data)
res_transmission$metadata
```

---

lr\_parse\_jdx

*Parse OceanInsight JCAMP-DX (.jdx) file*

---

## Description

Parse OceanInsight (formerly OceanOptics) JCAMP-DX (.jdx) file. <https://www.oceaninsight.com/>

## Usage

```
lr_parse_jdx(filename)
```

## Arguments

filename      Path of the file to parse

## Details

'processed' column computed by **lightr** with the function `lr_compute_processed()`.

## Value

A named list of two elements:

- data: a dataframe with columns "wl", "dark", "white", "scope" and "processed", in this order.
- metadata: a character vector with metadata including:
  - user: Name of the spectrometer operator
  - datetime: Timestamp of the recording in format '%Y-%m-%d %H:%M:%S' and UTC timezone. If timezone is missing in source file, UTC time will be assumed (for reproducibility purposes across computers with different localtimes).
  - spec\_model: Model of the spectrometer
  - spec\_ID: Unique ID of the spectrometer
  - white\_inttime: Integration time of the white reference (in ms)
  - dark\_inttime: Integration time of the dark reference (in ms)
  - sample\_inttime: Integration time of the sample (in ms)

- white\_avgs: Number of averaged measurements for the white reference
- dark\_avgs: Number of averaged measurements for the dark reference
- sample\_avgs: Number of averaged measurements for the sample
- white\_boxcar: Boxcar width for the white reference
- dark\_boxcar: Boxcar width for the dark reference
- sample\_boxcar: Boxcar width for the sample reference

## References

McDonald RS, Wilks PA. JCAMP-DX: A Standard Form for Exchange of Infrared Spectra in Computer Readable Form. *Applied Spectroscopy*. 1988;42(1):151-62.

## Examples

```
res_jdx <- lr_parse_jdx(system.file("testdata", "OceanOptics_period.jdx",
                                  package = "lightr"))
head(res_jdx$data)
res_jdx$metadata
```

---

lr\_parse\_procspec      *Parse OceanInsight ProcSpec file*

---

## Description

Parse OceanInsight (formerly OceanOptics) ProcSpec file. <https://www.oceaninsight.com/>

## Usage

```
lr_parse_procspec(filename)
```

## Arguments

filename      Path of the file to parse

## Details

'processed' column computed by official software and provided as is.

## Value

A named list of two elements:

- data: a dataframe with columns "wl", "dark", "white", "scope" and "processed", in this order.
- metadata: a character vector with metadata including:
  - user: Name of the spectrometer operator

- `datetime`: Timestamp of the recording in format `'%Y-%m-%d %H:%M:%S'` and UTC timezone. If timezone is missing in source file, UTC time will be assumed (for reproducibility purposes across computers with different localtimes).
- `spec_model`: Model of the spectrometer
- `spec_ID`: Unique ID of the spectrometer
- `white_inttime`: Integration time of the white reference (in ms)
- `dark_inttime`: Integration time of the dark reference (in ms)
- `sample_inttime`: Integration time of the sample (in ms)
- `white_avgs`: Number of averaged measurements for the white reference
- `dark_avgs`: Number of averaged measurements for the dark reference
- `sample_avgs`: Number of averaged measurements for the sample
- `white_boxcar`: Boxcar width for the white reference
- `dark_boxcar`: Boxcar width for the dark reference
- `sample_boxcar`: Boxcar width for the sample reference

## References

<https://www.oceaninsight.com/support/faqs/software/>

## Examples

```
res <- lr_parse_procspec(system.file("testdata", "procspec_files",
                                   "OceanOptics_Linux.ProcSpec",
                                   package = "lightr"))

head(res$data)
res$metadata
```

---

<code>lr_parse_spc</code>	<i>Parse SPC binary file</i>
---------------------------	------------------------------

---

## Description

Parse SPC binary file. (Used by CRAIC <https://www.microspectra.com/> and OceanInsight <https://www.oceaninsight.com/>)

## Usage

```
lr_parse_spc(filename)
```

## Arguments

`filename`            Path of the file to parse

## Details

'processed' column computed by official software and provided as is.

**Value**

A named list of two elements:

- data: a dataframe with columns "wl", "dark", "white", "scope" and "processed", in this order.
- metadata: a character vector with metadata including:
  - user: Name of the spectrometer operator
  - datetime: Timestamp of the recording in format '%Y-%m-%d %H:%M:%S' and UTC timezone. If timezone is missing in source file, UTC time will be assumed (for reproducibility purposes across computers with different localtimes).
  - spec\_model: Model of the spectrometer
  - spec\_ID: Unique ID of the spectrometer
  - white\_inttime: Integration time of the white reference (in ms)
  - dark\_inttime: Integration time of the dark reference (in ms)
  - sample\_inttime: Integration time of the sample (in ms)
  - white\_avgs: Number of averaged measurements for the white reference
  - dark\_avgs: Number of averaged measurements for the dark reference
  - sample\_avgs: Number of averaged measurements for the sample
  - white\_boxcar: Boxcar width for the white reference
  - dark\_boxcar: Boxcar width for the dark reference
  - sample\_boxcar: Boxcar width for the sample reference

**In development**

Metadata parsing has not yet been implemented for this file format.

**Examples**

```
res <- lr_parse_spc(system.file("testdata", "compare", "CRAIC", "CRAIC.spc",
                               package = "lightr"))
head(res$data)
res$metadata
```

---

lr\_parse\_trm

*Parse Avantes binary file*


---

**Description**

Parse Avantes binary file (TRM, ABS, ROH, DRK, REF, RAW8, RFL8 file extensions). <https://www.avantes.com/products/spectrometers/>

## Usage

```
lr_parse_trm(filename)

lr_parse_abs(filename)

lr_parse_roh(filename)

lr_parse_rfl8(filename, specnum = 1L)

lr_parse_raw8(filename, specnum = 1L)

lr_parse_irr8(filename, specnum = 1L)
```

## Arguments

filename	Path of the file to parse
specnum	Integer representing the position of the spectrum to read in the file. This option only makes sense for AvaSoft8 files and is ignored in the other cases.

## Details

'processed' column computed by **lightr** with the function `lr_compute_processed()`.

## Value

A named list of two elements:

- data: a dataframe with columns "wl", "dark", "white", "scope" and "processed", in this order.
- metadata: a character vector with metadata including:
  - user: Name of the spectrometer operator
  - datetime: Timestamp of the recording in format '%Y-%m-%d %H:%M:%S' and UTC timezone. If timezone is missing in source file, UTC time will be assumed (for reproducibility purposes across computers with different localtimes).
  - spec\_model: Model of the spectrometer
  - spec\_ID: Unique ID of the spectrometer
  - white\_inttime: Integration time of the white reference (in ms)
  - dark\_inttime: Integration time of the dark reference (in ms)
  - sample\_inttime: Integration time of the sample (in ms)
  - white\_avgs: Number of averaged measurements for the white reference
  - dark\_avgs: Number of averaged measurements for the dark reference
  - sample\_avgs: Number of averaged measurements for the sample
  - white\_boxcar: Boxcar width for the white reference
  - dark\_boxcar: Boxcar width for the dark reference
  - sample\_boxcar: Boxcar width for the sample reference

**Examples**

```

res_trm <- lr_parse_trm(
  system.file("testdata", "avantes_trans.TRM", package = "lightr")
)
head(res_trm$data)
res_trm$metadata

res_roh <- lr_parse_roh(
  system.file("testdata", "avantes_reflect.ROH", package = "lightr")
)
head(res_roh$data)
res_roh$metadata

# This parser has a unique `specnum` argument
res_rfl8_1 <- lr_parse_rfl8(
  system.file("testdata", "compare", "Avantes", "feather.RFL8", package = "lightr"),
  specnum = 1
)
head(res_rfl8_1$data)
res_rfl8_1$metadata

res_rfl8_2 <- lr_parse_rfl8(
  system.file("testdata", "compare", "Avantes", "feather.RFL8", package = "lightr"),
  specnum = 2
)
head(res_rfl8_2$data)
res_rfl8_2$metadata

```

---

lr\_parse\_ttt

*Parse Avantes converted file*


---

**Description**

Parse Avantes converted file. <https://www.avantes.com/products/spectrometers/>

**Usage**

```
lr_parse_ttt(filename)
```

```
lr_parse_trt(filename)
```

**Arguments**

filename      Path of the file to parse

**Details**

'processed' column computed by official software and provided as is.

**Value**

A named list of two elements:

- data: a dataframe with columns "wl", "dark", "white", "scope" and "processed", in this order.
- metadata: a character vector with metadata including:
  - user: Name of the spectrometer operator
  - datetime: Timestamp of the recording in format '%Y-%m-%d %H:%M:%S' and UTC timezone. If timestamp is missing in source file, UTC time will be assumed (for reproducibility purposes across computers with different localtimes).
  - spec\_model: Model of the spectrometer
  - spec\_ID: Unique ID of the spectrometer
  - white\_inttime: Integration time of the white reference (in ms)
  - dark\_inttime: Integration time of the dark reference (in ms)
  - sample\_inttime: Integration time of the sample (in ms)
  - white\_avgs: Number of averaged measurements for the white reference
  - dark\_avgs: Number of averaged measurements for the dark reference
  - sample\_avgs: Number of averaged measurements for the sample
  - white\_boxcar: Boxcar width for the white reference
  - dark\_boxcar: Boxcar width for the dark reference
  - sample\_boxcar: Boxcar width for the sample reference

**Examples**

```
res_ttt <- lr_parse_ttt(  
  system.file("testdata", "avantes_export.ttt", package = "lightr")  
)  
head(res_ttt$data)  
res_ttt$metadata  
  
res_trt <- lr_parse_trt(  
  system.file("testdata", "avantes_export2.trt", package = "lightr")  
)  
head(res_trt$data)  
res_trt$metadata
```

# Index

`future::plan()`, [3](#), [4](#), [6](#)

`lr_compute_processed()`, [9](#), [13](#)  
`lr_convert_to_csv`, [2](#)  
`lr_get_metadata`, [3](#)  
`lr_get_spec`, [5](#)  
`lr_parse_abs` (`lr_parse_trm`), [12](#)  
`lr_parse_generic`, [6](#)  
`lr_parse_irr8` (`lr_parse_trm`), [12](#)  
`lr_parse_jaz`, [7](#)  
`lr_parse_jazirrad` (`lr_parse_jaz`), [7](#)  
`lr_parse_jdx`, [9](#)  
`lr_parse_procspec`, [10](#)  
`lr_parse_raw8` (`lr_parse_trm`), [12](#)  
`lr_parse_rfl8` (`lr_parse_trm`), [12](#)  
`lr_parse_roh` (`lr_parse_trm`), [12](#)  
`lr_parse_spc`, [11](#)  
`lr_parse_trm`, [12](#)  
`lr_parse_trt` (`lr_parse_ttt`), [14](#)  
`lr_parse_ttt`, [14](#)

`pavo::getspec()`, [6](#)  
`progressr::handlers()`, [3](#), [4](#), [6](#)