Package: ProTrackR2 (via r-universe)

October 25, 2024

Title Manipulate and Play 'ProTracker' Modules

Version 0.0.4

Maintainer Pepijn de Vries <pepijn.devries@outlook.com>

Description 'ProTracker' is a popular music tracker to sequence music
on a Commodore Amiga machine. This package offers the
opportunity to import, export, manipulate and play 'ProTracker'
module files. Even though the file format could be considered
archaic, it still remains popular to this date. This package
intends to contribute to this popularity and therewith keeping
the legacy of 'ProTracker' and the Commodore Amiga alive. This
package is the successor of 'ProTrackR' providing better
performance.

License GPL (>= 3)

Encoding UTF-8

Roxygen list(markdown = TRUE)

RoxygenNote 7.2.3

LinkingTo cpp11

Collate 'ProTrackR2-package.R' 'cell.R' 'cpp11.R' 'helpers.R' 'io.R'
'mod_info.R' 'modplug.R' 'pattern.R' 'render.R' 's3.R'
'samples.R' 'select_ops.R' 'validate.R'

Imports audio

Suggests cli, htmltools, kableExtra, knitr, rmarkdown

VignetteBuilder knitr

URL https://pepijn-devries.github.io/ProTrackR2/

Repository https://pepijn-devries.r-universe.dev

RemoteUrl https://github.com/pepijn-devries/ProTrackR2

RemoteRef master

RemoteSha e58cc68ed53d2f60226afc5af172983df6028fc8

Contents

---

as_modplug_pattern	*Format a ProTracker pattern conform OpenMPT specs*

Description

OpenMpt is a popular modern music tracker. This function allows you to format a pattern such that it can be pasted directly into OpenMPT. On Windows you can use `writeClipboard()` for this purpose.

Usage

```
as_modplug_pattern(pattern, ...)
```

Arguments

pattern	An object of class `pt2pat` to be formatted
...	Ignored

Value

Returns a `character` object formatted such that it can be copied into OpenMPT

Author(s)

Pepijn de Vries

Examples

```
mod    <- pt2_read_mod(system.file("mod.intro", package = "ProTrackR2"))
mp_pat <- as_modplug_pattern(pt2_pattern(mod, 0L))
```

---

format.pt2mod   *Implementation of basic S3 methods*

Description

Implementation of basic S3 methods, such as, format, print, as.raw and as.character (see usage section for a complete overview). See vignette('s3class') for an overview of ProTrackR2 S3 class objects. See usage section for an overview of available methods.

Usage

```
## S3 method for class 'pt2mod'
format(x, ...)

## S3 method for class 'pt2mod'
print(x, ...)

## S3 method for class 'pt2pat'
format(
  x,
  padding = " ",
  empty_char = "-",
  fmt = getOption("pt2_cell_format"),
  ...
)

## S3 method for class 'pt2pat'
print(x, sep = "  ", show_header = TRUE, show_row = TRUE, ...)

## S3 method for class 'pt2pat'
as.character(x, ...)

## S3 method for class 'pt2celllist'
as.raw(x, ...)

## S3 method for class 'logical'
as.raw.pt2celllist(x, compact = TRUE, ...)

## S3 method for class 'pt2pat'
as.raw(x, ...)

## S3 method for class 'logical'
as.raw.pt2pat(x, compact = TRUE, ...)
```

```
## S3 method for class 'pt2cell'
format(
  x,
  padding = " ",
  empty_char = "-",
  fmt = getOption("pt2_cell_format"),
  ...
)

## S3 method for class 'pt2cell'
print(x, ...)

## S3 method for class 'pt2cell'
as.character(x, ...)

## S3 method for class 'pt2cell'
as.raw(x, ...)

## S3 method for class 'logical'
as.raw.pt2cell(x, compact = TRUE, ...)

## S3 method for class 'pt2samp'
format(x, ...)

## S3 method for class 'pt2samp'
print(x, ...)

## S3 method for class 'pt2patlist'
format(x, ...)

## S3 method for class 'pt2patlist'
print(x, ...)

## S3 method for class 'pt2celllist'
format(x, ...)

## S3 method for class 'pt2celllist'
print(x, ...)

## S3 method for class 'pt2samplist'
format(x, ...)

## S3 method for class 'pt2samplist'
print(x, ...)

## S3 method for class 'pt2mod'
as.raw(x, ...)
```

```
## S3 method for class 'pt2samp'
as.raw(x, ...)

## S3 method for class 'pt2samp'
as.integer(x, ...)
```

Arguments

x	Object to apply S3 method to. See 'usage' section for allowed object types.
...	Passed on to other methods.
padding	A `vector` of `character` strings used to pad between note and instrument number (element 1) and between instrument number and effect command (element 2). Values are recycled.
empty_char	A `vector` of single `character` values used to represent empty pattern elements. First element is used for notes, second for instrument number, the third for effect commands. Values are recycled.
fmt	Experimental feature to format a `pt2cell`. It should be a named `list` containing formatting strings for elements in the cell. It should contain the elements `"note"`, `"padding"`, `"instrument"` and `"effect"`. Its implementation may change in future releases.
sep	A separator `character` string for concatenating pattern table columns (i.e. channels).
show_header	A `logical` value indicating if a header should be shown for the pattern table.
show_row	A `logical` value indicating if the row of a pattern table should be labelled with its index.
compact	Should the pattern be formatted using a compact notation (as used for file storage), or a none-compact format as used by the player? This can be set with the `compact` argument.

Value

The following is returned by the different methods:

- `format`: a formatted `character` representation of the object
- `print`: same as `format`
- `as.character`: same as `format`
- `as.raw`: a `raw` representation of the object. In many cases it inherits the same class as `x`
- `as.integer`: converted `raw` 8 bit sample data to signed pulse code modulation `integer` values between -128 and +127.

play   *Play a ProTracker module*

Description

Renders a ProTracker module as `audio::audioSample()` and plays it.

Usage

```
## S3 method for class 'pt2mod'
play(x, duration = NA, options = pt2_render_options(), position = 0L, ...)
```

Arguments

x	Object to be played.
duration	Duration of the rendered output in seconds. When set to NA the duration of the module is calculated and used for rendering.
options	A list of options used for rendering the audio. Use `pt2_render_options()` to obtain default options, or modify them.
position	Starting position in the pattern sequence table (`pt2_pattern_table()`). Should be a non negative value smaller than the mule length (`pt2_length()`).
...	Arguments passed to `pt2_render()`.

Value

Returns an [audio::$.audioInstance] object which allows you to control the playback (pause, resume, rewind).

Author(s)

Pepijn de Vries

Examples

```
## Not run:
mod <- pt2_read_mod(system.file("mod.intro", package = "ProTrackR2"))

## ctrl will contain the audioInstance that will let you control the audio playback:
ctrl <- play(mod)

## End(Not run)
```

---

pt2_cell *Select a cell from a ProTracker pattern table*

Description

A cell is an element at a specific row and column (channel). It holds information aboute the note to be played, the instrument (sample) number and the effect to be applied. For mor information about cells (class `pt2cell`) consult `vignette("s3class")`. For more information about selecting elements from ProTrackR2 class objects check out `vignette("select_opts")`.

Usage

```
pt2_cell(pattern, i, j, ...)
```

Arguments

pattern	A `pt2pat` class object to extract a cell (`pt2cell`) from.
i, j	Indices for extracting or replacing ProTrackR2 object elements. The indices starts at 0, for consistency with ProTracker!
...	Ignored

Value

Returns a cell object from the table as class `pt2cell`.

Author(s)

Pepijn de Vries

Examples

```
mod <- pt2_read_mod(system.file("mod.intro", package = "ProTrackR2"))

pt2_cell(mod$patterns[[1]], 0L, 0L)
```

---

pt2_duration *Calculate the duration of the module*

Description

How long a module will play depends on several aspects such as the length of the pattern sequence table (`pt2_pattern_table()`, `pt2_length()`), the speed and tempo at which the patterns are defined, loops, pattern breaks and delay effects. The duration in seconds of the module is calculated by this function.

Usage

```
pt2_duration(x, options = pt2_render_options(), position = 0L, ...)
```

Arguments

x	Object for which to determine the duration. Should be of class pt2mod.
options	A list of options used for rendering the audio. Use pt2_render_options() to obtain default options, or modify them.
position	Starting position in the pattern sequence table (pt2_pattern_table()). Should be a non negative value smaller than the mule length (pt2_length()).
...	Ignored

Value

The duration in seconds (as a difftime class object)

Author(s)

Pepijn de Vries

Examples

```
mod <- pt2_read_mod(system.file("mod.intro", package = "ProTrackR2"))

pt2_duration(mod)
```

---

pt2_length   *Obtain ProTracker module information*

Description

Obtain information about a protracker module or embedded samples.

Usage

```
pt2_length(mod, ...)

pt2_length(mod, ...) <- value

pt2_n_pattern(mod, ...)

pt2_pattern_table(mod, ...)

pt2_name(x, ...)

## S3 method for class 'pt2mod'
pt2_name(x, ...)
```

```
## S3 method for class 'pt2samp'
pt2_name(x, ...)

pt2_n_sample(mod, ...)
```

Arguments

`...`	Ignored
`value`	New length of a module in number of patterns in the pattern sequence table.
`x, mod`	A `pt2mod` class object for which to obtain information. For `x` also samples of class `pt2samp` are allowed as input.

Value

Returns information about the specified ProTracker module

Author(s)

Pepijn de Vries

Examples

```
mod <- pt2_read_mod(system.file("mod.intro", package = "ProTrackR2"))

pt2_length(mod)
pt2_n_pattern(mod)
pt2_n_sample(mod)
pt2_pattern_table(mod)
pt2_name(mod)
```

`pt2_new_mod`	*Create a new (empty) ProTracker module*

Description

Creates an empty ProTracker module, it is returned as a `pt2mod` class object.

Usage

```
pt2_new_mod(name, ...)
```

Arguments

`name`	Name for the new module. It will be truncated if longer than 20 characters.
`...`	Ignored

Value

A `pt2mod` class module, with no samples and one empty pattern.

Author(s)

Pepijn de Vries

Examples

```
mod <- pt2_new_mod("my_song")
```

---

pt2_new_pattern   *Create a new ProTracker pattern*

Description

Creates a new ProTracker pattern, consisting of four channels and 64 rows.

Usage

```
pt2_new_pattern(..., compact = TRUE)
```

Arguments

`...`	Currently ignored
`compact`	Should the pattern be formatted using a compact notation (as used for file storage), or a none-compact format as used by the player? This can be set with the `compact` argument.

Value

Returns a new clean `pt2pat` object.

Author(s)

Pepijn de Vries

Examples

```
pt2_new_pattern()
```

---

pt2_pattern *Retrieve a pattern from a ProTracker module*

Description

Get a pattern table (sequence of notes and effects on each of the 4 channels) at a specific index from a ProTracker module.

Usage

```
pt2_pattern(mod, i, ...)
```

Arguments

mod	A pt2mod class objects from which to retrieve a pattern table
i	The index (integer) of the pattern. Note that the index starts at 0.
...	Ignored

Value

A pt2pat object representing the pattern.

Author(s)

Pepijn de Vries

Examples

```
mod <- pt2_read_mod(system.file("mod.intro", package = "ProTrackR2"))

pt2_pattern(mod, 0L)
```

---

pt2_read_mod *Read and write ProTracker modules*

Description

Functions to read and write ProTracker module. The read function will read a number of mod files that are compatible with ProTracker, this includes files compressed with PowerPacker (PP). The write function will only write modules conform ProTracker specifications.

Usage

```
pt2_read_mod(file, ...)

pt2_write_mod(mod, file, ...)
```

Arguments

`file`	Filename of the file to read from or write to.
`...`	Ignored
`mod`	An object of class `pt2mod`.

Value

`pt2_read_mod()` returns a `pt2mod` class object when successful. `pt_write_mod()` returns `NULL` invisibly.

Author(s)

Pepijn de Vries

Examples

```
mod <- pt2_read_mod(system.file("mod.intro", package = "ProTrackR2"))
```

---

pt2_read_sample   *Read and write ProTracker audio samples*

Description

Functions to read and write ProTracker audio samples. Reading is supported for common types of WAV, IFF and AIFF files. Writing is supported for WAV and IFF files.

Usage

```
pt2_read_sample(file, ...)

pt2_write_sample(sample, file, ...)
```

Arguments

`file`	Filename of the file to read from or write to. For `pt2_write_sample()` the file extension will be used to determine which file format to write.
`...`	Ignored
`sample`	An object of class `pt2samp`.

Value

`pt2_read_sample()` returns a `pt2samp` class object when successful. `pt_write_sample()` returns `NULL` invisibly.

Author(s)

Pepijn de Vries

Examples

```
mod <- pt2_read_mod(system.file("mod.intro", package = "ProTrackR2"))
my_sample <- pt2_sample(mod, 1L)
my_sample_file <- tempfile(fileext = ".iff")
pt2_write_sample(my_sample, my_sample_file)
```

---

pt2_render   *Render ProTracker modules to a playable format*

Description

Renders a 16bit pulse-code modulation waveform from a ProTracker module. The rendered format can be played on a modern machine.

Usage

```
pt2_render(x, duration = NA, options = pt2_render_options(), ...)

## S3 method for class 'pt2mod'
pt2_render(
  x,
  duration = NA,
  options = pt2_render_options(),
  position = 0L,
  ...
)
```

Arguments

x	The object to be rendered
duration	Duration of the rendered output in seconds. When set to NA the duration of the module is calculated and used for rendering.
options	A list of options used for rendering the audio. Use pt2_render_options() to obtain default options, or modify them.
...	Ignored
position	Starting position in the pattern sequence table (pt2_pattern_table()). Should be a non negative value smaller than the mule length (pt2_length()).

Value

Rendered audio inheriting the audio::audioSample() class.

Author(s)

Pepijn de Vries

Examples

```
mod <- pt2_read_mod(system.file("mod.intro", package = "ProTrackR2"))
aud <- pt2_render(mod)
```

---

pt2_render_options   *Retrieve options for rendering*

Description

Retrieve options for rendering ProTracker modules. See also `pt2_render()`.

Usage

```
pt2_render_options(...)
```

Arguments

...	Specify custom options.

Value

Returns a named `list` of options that can be used for rendering ProTracker modules (see `pt2_render()` and `play()`). It contains the following elements:

- sample_rate: an integer value specifying the sample rate of the output in Hz.
- stereo_separation: an integer percentage determining how much the tracker channels will be separated to the left and right stereo output channels.
- amiga_filter: a `character` string specifying the hardware filter to be emulated. Can be `"A500"` for emulating Amiga 500 hardware filters, or `"A1200"` for emulating Amiga 1200 hardware filters.
- speed: An integer value specifying the initial speed of the module measured in 'ticks' per row. Should be in range of 1 and 31.
- tempo: An integer value specifying the initial tempo of the module. When speed is set to 6, it measures the tempo as beats per minute. Should be in the range of 32 and 255
- led_filter: A `logical` value specifying the state of the hardware LED filter to be emultated.

Author(s)

Pepijn de Vries

Examples

```
pt2_render_options(stereo_separation = 100)
```

---

pt2_sample   *Obtain sample data and info from a ProTracker module*

Description

Obtain sample data and info from a ProTracker module at a specific index. ProTracker modules can hold up to 31 samples. The index should range from 0 to 30.

Usage

```
pt2_sample(mod, i, ...)
```

Arguments

mod	An object of class pt2mod from which to obtain sample data and information
i	The index of the requested sample (between 0 and 30).
...	Ignored.

Value

Returns a sample object of class pt2samp.

Author(s)

Pepijn de Vries

Examples

```
mod <- pt2_read_mod(system.file("mod.intro", package = "ProTrackR2"))

smp <- pt2_sample(mod, 0L)
```

---

pt2_validate   *Validate ProTrackR2 S3 class objects*

Description

Check aspects of S3 class objects for validity. For samples for instance it is checked if all parameters (volume, finetune, etc.) are within ProTracker specifications.

Usage

```
pt2_validate(x, ...)

## S3 method for class 'pt2samp'
pt2_validate(x, ...)
```

Arguments

x	object to be validated
...	Ignored

Value

A `logical` value indicating whether the object is valid or not

Author(s)

Pepijn de Vries

Examples

```
mod <- pt2_read_mod(system.file("mod.intro", package = "ProTrackR2"))

pt2_validate(mod$samples[[1]])
```

$.pt2mod   *Select and assign operators for ProTrackR2 S3 class objects*

Description

Functions to select and assign elements to ProTracker modules. See `vignette('s3class')` for an overview of ProTrackR2 S3 class objects. See `vignette('sel_assign')` for practical guidance on selecting and assigning elements of ProTrackR2 class objects.

Usage

```
## S3 method for class 'pt2mod'
x$i, ...

## S3 replacement method for class 'pt2mod'
x$i <- value

## S3 method for class 'pt2mod'
x[[i, ...]]

## S3 method for class 'pt2patlist'
x[i, ...]

## S3 method for class 'pt2patlist'
x[[i, ...]]

## S3 replacement method for class 'pt2patlist'
x[[i]] <- value
```

```
## S3 method for class 'pt2samplist'
x[i, ...]

## S3 method for class 'pt2samplist'
x[[i, ...]]

## S3 method for class 'pt2pat'
x[i, j, ...]

## S3 replacement method for class 'pt2pat'
x[i, j, ...] <- value
```

Arguments

x	Object to apply S3 method to. See 'usage' section for allowed object types.
i, j	Indices for extracting or replacing ProTrackR2 object elements
...	Passed on to other methods.
value	Replacement value for the selected element(s).

Value

Returns the selected object in case of a selection ([, [[, or $) operator. Returns the updated object x in case of an assignment (<-) operator.

Index