Package 'xmpdf'

February 26, 2023

Encoding UTF-8

Type Package

Title Edit 'XMP' Metadata and 'PDF' Bookmarks and Documentation Info

Version 0.1.3

Description
Edit 'XMP' metadata <https://en.wikipedia.org/wiki/Extensible_Metadata_Platform>
in a variety of media file formats as well as
edit bookmarks (aka outline aka table of contents) and documentation info entries in 'pdf' files.
Can detect and use a variety of command-line tools to perform these operations such as
'exiftool' <https://exiftool.org/>, 'ghostscript' <https://www.ghostscript.com/>,
and/or 'pdftk' <https://gitlab.com/pdftk-java/pdftk>.

License GPL (>= 3)

URL https://trevorldavis.com/R/xmpdf/dev/

BugReports https://github.com/trevorld/r-xmpdf/issues

LazyData true

LazyLoad yes

Imports datetimeoffset (>= 0.2.1), grDevices, jsonlite, purrr, R6,
rlang, tools, utils

Suggests exiftoolr, grid, knitr, qpdf, pdftools, rmarkdown, testthat
(>= 3.0.0)

VignetteBuilder knitr, rmarkdown

SystemRequirements 'ghostscript' or 'pdftk' for editing pdf bookmarks
and/or documentation info entries. 'exiftool' for editing pdf
documentation info entries and/or xmp metadata.

RoxygenNote 7.2.1

Config/testthat/edition 3

NeedsCompilation no

Author Trevor L Davis [aut, cre] (<https://orcid.org/0000-0001-6341-4639>),
W. Trevor King Some pdf code adapted from 'pdf-merge.py' [ctb],
Linux Foundation [dtc] (Uses some data from the ``SPDX License List''
<https://github.com/spdx/license-list-XML>)

Maintainer Trevor L Davis <trevor.l.davis@gmail.com>

Depends R (>= 2.10)

Repository CRAN

Date/Publication 2023-02-26 01:00:02 UTC

R topics documented:

`as_docinfo` *Coerce to docinfo objects*

Description

`as_docinfo()` coerces objects into a `docinfo()` object.

Usage

```
as_docinfo(x, ...)

## S3 method for class 'xmp'
as_docinfo(x, ...)
```

Arguments

`x`	An object that can reasonably be coerced to a `docinfo()` object.
`...`	Further arguments passed to or from other methods.

Value

A `docinfo()` object.

Examples

```
x <- xmp(`dc:Creator` = "John Doe", `dc:Title` = "A Title")
as_docinfo(x)
```

---

as_lang_alt *Coerce to XMP "language alternative" structure*

Description

as_lang_alt() coerces to an XMP "language alternative" structure suitable for use with xmp() objects.

Usage

```
as_lang_alt(x, ...)

## S3 method for class 'character'
as_lang_alt(x, ..., default_lang = getOption("xmpdf_default_lang"))

## S3 method for class 'lang_alt'
as_lang_alt(x, ...)

## S3 method for class 'list'
as_lang_alt(x, ..., default_lang = getOption("xmpdf_default_lang"))
```

Arguments

x	Object suitable for coercing
...	Ignored
default_lang	Language tag value to copy as the "x-default"

Value

A named list of class "lang_alt".

See Also

xmp(), as_xmp(), get_xmp(), and set_xmp(). For more information about the XMP "language alternative" structure see https://github.com/adobe/xmp-docs/blob/master/XMPNamespaces/XMPDataTypes/CoreProperties.md#language-alternative.

Examples

```
as_lang_alt("A single title")
as_lang_alt(c(en = "An English Title", fr = "A French Title"))
as_lang_alt(c(en = "An English Title", fr = "A French Title"), default_lang = "en")
as_lang_alt(list(en = "An English Title", fr = "A French Title"))
```

as_xmp *Coerce to xmp objects*

Description

`as_xmp()` coerces objects into an `xmp()` object.

Usage

```
as_xmp(x, ...)

## S3 method for class 'docinfo'
as_xmp(x, ...)

## S3 method for class 'list'
as_xmp(x, ...)
```

Arguments

`x`	An object that can reasonably be coerced to a `xmp()` object.
`...`	Further arguments passed to or from other methods.

Value

An `xmp()` object.

Examples

```
di <- docinfo(author = "John Doe", title = "A Title")
as_xmp(di)

l <- list(`dc:creator` = "John Doe", `dc:title` = "A Title")
as_xmp(l)
```

bookmarks *Set/get pdf bookmarks*

Description

`get_bookmarks()` gets pdf bookmarks from a file. `set_bookmarks()` sets pdf bookmarks for a file.

Usage

```
get_bookmarks(filename, use_names = TRUE)

get_bookmarks_pdftk(filename, use_names = TRUE)

set_bookmarks(bookmarks, input, output = input)

set_bookmarks_pdftk(bookmarks, input, output = input)

set_bookmarks_gs(bookmarks, input, output = input)
```

Arguments

filename	Filename(s) (pdf) to extract bookmarks from.
use_names	If TRUE (default) use filename as the names of the result.
bookmarks	A data frame with bookmark information with the following columns: **title** Title for bookmark (mandatory, character) **page** Page number for bookmark (mandatory, integer) **level** Level of bookmark e.g. 1 top level, 2 second level, etc. (optional, integer). If missing will be inferred from count column else will be assumed to be 1L. **count** Number of bookmarks immediately subordinate (optional, integer). Excludes subordinates of subordinates. Positive count indicates bookmark should start open while negative count indicates that this bookmark should start closed. If missing will be inferred from level column and (if specified) the open column else will be assumed to be 0L. Note some pdf viewers quietly ignore the initially open/closed feature. **open** Whether the bookmark starts open or closed if it has subordinate bookmarks (optional, logical). If missing will default to open. Ignored if the count column is specified (instead use a negative count if the bookmark should start closed). Note some pdf viewers quietly ignore the initially open/closed feature. **fontface** Font face of the bookmark (optional, integer). If NA_character_ or NA_integer_ will be unset (defaults to "plain"). "plain" or 1 is plain, "bold" or 2 is bold, "italic" or 3 is italic, Note many pdf viewers quietly ignore this feature. **color** Color of the bookmark (optional, character). If NA_character_ will be unset (presumably defaults to "black"). Note many pdf viewers quietly ignore this feature.
input	Input pdf filename.
output	Output pdf filename.

Details

get_bookmarks() will try to use the following helper functions in the following order:

1. get_bookmarks_pdftk() which wraps pdftk command-line tool

`set_bookmarks()` will try to use the following helper functions in the following order:

1. `set_bookmarks_gs()` which wraps `ghostscript` command-line tool
2. `set_bookmarks_pdftk()` which wraps `pdftk` command-line tool

Value

`get_bookmarks()` returns a list of data frames with bookmark info (see `bookmarks` parameter for details about columns) plus "total_pages", "filename", and "title" attributes. `NA` values in the data frame indicates that the backend doesn't report information about this pdf feature. `set_bookmarks()` returns the (output) filename invisibly.

Known limitations

- `get_bookmarks_pdftk()` doesn't report information about bookmarks color, fontface, and whether the bookmarks should start open or closed.
- `set_bookmarks_gs()` supports most bookmarks features including color and font face but only action supported is to view a particular page.
- `set_bookmarks_pdftk()` only supports setting the title, page number, and level of bookmarks.

See Also

`supports_get_bookmarks()`, `supports_set_bookmarks()`, `supports_gs()`, and `supports_pdftk()` to detect support for these features. For more info about the pdf bookmarks feature see https://opensource.adobe.com/dc-acrobat-sdk-docs/library/pdfmark/pdfmark_Basic.html#bookmarks-out.

Examples

```
# Create 2-page pdf using `pdf)` and add some bookmarks to it
if (supports_set_bookmarks() && supports_get_bookmarks() && require("grid", quietly = TRUE)) {
  f <- tempfile(fileext = ".pdf")
  pdf(f, onefile = TRUE)
  grid.text("Page 1")
  grid.newpage()
  grid.text("Page 2")
  invisible(dev.off())

  print(get_bookmarks(f)[[1]])


  bookmarks <- data.frame(title = c("Page 1", "Page 2"), page = c(1, 2))

  set_bookmarks(bookmarks, f)
  print(get_bookmarks(f)[[1]])
  unlink(f)
}
```

`cat_bookmarks` *Concatenate pdf bookmarks*

Description

`cat_bookmarks()` concatenates a list of bookmarks into a single bookmarks data frame while updating the page numbers. Useful if wanting to concatenate multiple pdf files together and would like to preserve the bookmarks information.

Usage

```
cat_bookmarks(
  l,
  method = c("flat", "filename", "title"),
  open = NA,
  color = NA_character_,
  fontface = NA_character_
)
```

Arguments

`l`	A list of bookmark data frames as returned by `get_bookmarks()`. Each data frame should have a "total_pages" attribute. If `method = "filename"` each data frame should have a "filename" attribute. If `method = "title"` each data frame should have a "title" attribute.
`method`	If "flat" simply concatenate the bookmarks while updating page numbers. If "filename" place each file's bookmarks a level under a new bookmark matching the (base)name of the filename and then concatenate the bookmarks while updating page numbers. If "title" place each file's bookmarks a level under a new bookmark matching the title of the file and then concatenate the bookmarks while updating page numbers.
`open`	If `method = "filename"` or `method = "title"` a logical for whether the new top level bookmarks should start open? If missing will default to open. Note some pdf viewers quietly ignore the initially open/closed feature.
`color`	If `method = "filename"` or `method = "title"` the color of the new top level bookmarks. If `NA_character_` will be unset (presumably defaults to "black"). Note many pdf viewers quietly ignore this feature.
`fontface`	If `method = "filename"` or `method = "title"` should the fontface of the new top level bookmarks. If `NA_character_` or `NA_integer_` will be unset (defaults to "plain"). "plain" or 1 is plain, "bold" or 2 is bold, "italic" or 3 is italic, Note many pdf viewers quietly ignore this feature.

Value

A data frame of bookmark data (as suitable for use with `set_bookmarks()`). A "total_pages" attribute will be set for the theoretical total pages of the concatenated document represented by the concatenated bookmarks.

See Also

get_bookmarks() and set_bookmarks() for setting bookmarks. cat_pages() for concatenating pdf files together.

Examples

```
if (supports_get_bookmarks() && supports_set_bookmarks() && require("grid", quietly = TRUE)) {
 # Create two different two-page pdf files
 make_pdf <- function(f, title) {
   pdf(f, onefile = TRUE, title = title)
   grid.text(paste(title, "Page 1"))
   grid.newpage()
   grid.text(paste(title, "Page 2"))
   invisible(dev.off())
 }
 f1 <- tempfile(fileext = "_doc1.pdf")
 on.exit(unlink(f1))
 make_pdf(f1, "Document 1")

 f2 <- tempfile(fileext = "_doc2.pdf")
 on.exit(unlink(f2))
 make_pdf(f2, "Document 2")

 # Add bookmarks to the two two-page pdf files
 bookmarks <- data.frame(title = c("Page 1", "Page 2"),
                         page = c(1L, 2L))
 set_bookmarks(bookmarks, f1)
 set_bookmarks(bookmarks, f2)
 l <- get_bookmarks(c(f1, f2))
 print(l)

 bm <- cat_bookmarks(l, method = "flat")
 cat('\nmethod = "flat":\n')
 print(bm)

 bm <- cat_bookmarks(l, method = "filename")
 cat('\nmethod = "filename":\n')
 print(bm)

 bm <- cat_bookmarks(l, method = "title")
 cat('\nmethod = "title":\n')
 print(bm)

 # `cat_bookmarks()` is useful for setting concatenated pdf files
 # created with `cat_pages()`
 if (supports_cat_pages()) {
    fc <- tempfile(fileext = "_cat.pdf")
    on.exit(unlink(fc))
    cat_pages(c(f1, f2), fc)
    set_bookmarks(bm, fc)
    unlink(fc)
 }
```

```
 unlink(f1)
 unlink(f2)
}
```

---

cat_pages *Concatenate pdf documents together*

Description

cat_pages() concatenates pdf documents together.

Usage

```
cat_pages(input, output)

cat_pages_gs(input, output)

cat_pages_pdftk(input, output)

cat_pages_qpdf(input, output)
```

Arguments

input	Filename(s) (pdf) to concatenate together
output	Filename (pdf) to save concatenated output to

Details

cat_pages() will try to use the following helper functions in the following order:

1. cat_pages_qpdf() which wraps qpdf::pdf_combine()
2. cat_pages_pdftk() which wraps pdftk command-line tool
3. cat_pages_gs() which wraps ghostscript command-line tool

Value

The (output) filename invisibly.

See Also

supports_cat_pages(), supports_gs(), and supports_pdftk() to detect support for these features. cat_bookmarks() for generating bookmarks for concatenated files.

Examples

```
if (supports_cat_pages() && require("grid", quietly = TRUE)) {
  # Create two different two-page pdf files
  make_pdf <- function(f, title) {
    pdf(f, onefile = TRUE, title = title)
    grid.text(paste(title, "Page 1"))
    grid.newpage()
    grid.text(paste(title, "Page 2"))
    invisible(dev.off())
  }
  f1 <- tempfile(fileext = "_doc1.pdf")
  on.exit(unlink(f1))
  make_pdf(f1, "Document 1")

  f2 <- tempfile(fileext = "_doc2.pdf")
  on.exit(unlink(f2))
  make_pdf(f2, "Document 2")

  fc <- tempfile(fileext = "_cat.pdf")
  on.exit(unlink(fc))
  cat_pages(c(f1, f2), fc)

  # Use `cat_bookmarks()` to create pdf bookmarks for concatenated output files
  if (supports_get_bookmarks() && supports_set_bookmarks()) {
     l <- get_bookmarks(c(f1, f2))
     bm <- cat_bookmarks(l, "title")
     set_bookmarks(bm, fc)
     print(get_bookmarks(fc)[[1]])
  }
  unlink(f1)
  unlink(f2)
  unlink(fc)
}
```

`docinfo` *PDF documentation info dictionary object*

Description

`docinfo()` creates a PDF documentation info dictionary object. Such objects can be used with `set_docinfo()` to edit PDF documentation info dictionary entries and such objects are returned by `get_docinfo()`.

Usage

```
docinfo(
  author = NULL,
  creation_date = NULL,
  creator = NULL,
```

```
  producer = NULL,
  title = NULL,
  subject = NULL,
  keywords = NULL,
  mod_date = NULL
)
```

Arguments

author	The document's author. Matching xmp metadata tag is dc:creator.
creation_date	The date the document was created. Will be coerced by datetimeoffset::as_datetimeoffset(). Matching xmp metadata tag is xmp:CreateDate.
creator	The name of the application that originally created the document (if converted to pdf). Matching xmp metadata tag is xmp:CreatorTool.
producer	The name of the application that converted the document to pdf. Matching xmp metadata tag is pdf:Producer.
title	The document's title. Matching xmp metadata tag is dc:title.
subject	The document's subject. Matching xmp metadata tag is dc:description.
keywords	Keywords for this document (for cross-document searching). Matching xmp metadata tag is pdf:Keywords. Will be coerced into a string by paste(keywords, collapse = ", ").
mod_date	The date the document was last modified. Will be coerced by datetimeoffset::as_datetimeoffset(). Matching xmp metadata tag is xmp:ModifyDate.

Known limitations

- Currently does not support arbitrary info dictionary entries.

docinfo R6 Class Methods

`get_item(key)` Get documentation info value for key `key`. Can also use the relevant active bindings to get documentation info values.

`set_item(key, value)` Set documentation info key `key` with value `value`. Can also use the relevant active bindings to set documentation info values.

`update(x)` Update documentation info key entries using non-NULL entries in object `x` coerced by `as_docinfo()`.

docinfo R6 Active Bindings

`author` The document's author.

`creation_date` The date the document was created.

`creator` The name of the application that originally created the document (if converted to pdf).

`producer` The name of the application that converted the document to pdf.

`title` The document's title.

`subject` The document's subject.

`keywords` Keywords for this document (for cross-document searching).

`mod_date` The date the document was last modified.

See Also

get_docinfo() and set_docinfo() for getting/setting such information from/to PDF files. as_docinfo() for coercing to this object. as_xmp() can be used to coerce docinfo() objects into xmp() objects.

Examples

```
if (supports_set_docinfo() && supports_get_docinfo() && require("grid", quietly = TRUE)) {
  f <- tempfile(fileext = ".pdf")
  pdf(f, onefile = TRUE)
  grid.text("Page 1")
  grid.newpage()
  grid.text("Page 2")
  invisible(dev.off())

  cat("\nInitial documentation info\n")
  d <- get_docinfo(f)[[1]]
  print(d)

  d <- update(d,
              author = "John Doe",
              title = "Two Boring Pages",
              keywords = "R, xmpdf")
  set_docinfo(d, f)

  cat("\nDocumentation info after setting it\n")
  print(get_docinfo(f)[[1]])

  unlink(f)
}
```

---

edit_docinfo *Set/get pdf document info dictionary*

Description

get_docinfo() gets pdf document info from a file. set_docinfo() sets pdf document info for a file.

Usage

```
get_docinfo(filename, use_names = TRUE)

get_docinfo_pdftools(filename, use_names = TRUE)

get_docinfo_exiftool(filename, use_names = TRUE)

set_docinfo_exiftool(docinfo, input, output = input)

get_docinfo_pdftk(filename, use_names = TRUE)
```

```
set_docinfo(docinfo, input, output = input)

set_docinfo_gs(docinfo, input, output = input)

set_docinfo_pdftk(docinfo, input, output = input)
```

Arguments

filename	Filename(s) (pdf) to extract info dictionary entries from.
use_names	If TRUE (default) use filename as the names of the result.
docinfo	A "docinfo" object (as returned by docinfo() or get_docinfo()).
input	Input pdf filename.
output	Output pdf filename.

Details

get_docinfo() will try to use the following helper functions in the following order:

1. get_docinfo_pdftk() which wraps pdftk command-line tool
2. get_docinfo_exiftool() which wraps exiftool command-line tool
3. get_docinfo_pdftools() which wraps pdftools::pdf_info()

set_docinfo() will try to use the following helper functions in the following order:

1. set_docinfo_exiftool() which wraps exiftool command-line tool
2. set_docinfo_gs() which wraps ghostscript command-line tool
3. set_docinfo_pdftk() which wraps pdftk command-line tool

Value

docinfo() returns a "docinfo" R6 class. get_docinfo() returns a list of "docinfo" R6 classes. set_docinfo() returns the (output) filename invisibly.

Known limitations

- Currently does not support arbitrary info dictionary entries.
- As a side effect set_docinfo_gs() seems to also update in previously set matching XPN metadata while set_docinfo_exiftool() and set_docinfo_pdftk() don't update any previously set matching XPN metadata. Some pdf viewers will preferentially use the previously set document title from XPN metadata if it exists instead of using the title set in documentation info dictionary entry. Consider also manually setting this XPN metadata using set_xmp().
- Old metadata information is usually not deleted from the pdf file by these operations. If deleting the old metadata is important one may want to try qpdf::pdf_compress(input, linearize = TRUE).
- get_docinfo_exiftool() will "widen" datetimes to second precision.
- get_docinfo_pdftools()'s datetimes may not accurately reflect the embedded datetimes.
- set_docinfo_pdftk() may not correctly handle documentation info entries with newlines in them.

See Also

docinfo() for more information about the documentation info objects. supports_get_docinfo(), supports_set_docinfo(), supports_gs(), and supports_pdftk() to detect support for these features. For more info about the pdf document info dictionary see https://opensource.adobe.com/dc-acrobat-sdk-docs/library/pdfmark/pdfmark_Basic.html#document-info-dictionary-docinfo.

Examples

```
if (supports_set_docinfo() && supports_get_docinfo() && require("grid", quietly = TRUE)) {
  f <- tempfile(fileext = ".pdf")
  pdf(f, onefile = TRUE)
  grid.text("Page 1")
  grid.newpage()
  grid.text("Page 2")
  invisible(dev.off())

  cat("\nInitial documentation info:\n\n")
  d <- get_docinfo(f)[[1]]
  print(d)

  d <- update(d,
              author = "John Doe",
              title = "Two Boring Pages",
              keywords = c("R", "xmpdf"))
  set_docinfo(d, f)

  cat("\nDocumentation info after setting it:\n\n")
  print(get_docinfo(f)[[1]])

  unlink(f)
}
```

---

edit_xmp *Set/get xmp metadata*

Description

get_xmp() gets xmp metadata from a file. set_xmp() sets xmp metadata for a file.

Usage

```
get_xmp(filename, use_names = TRUE)

get_xmp_exiftool(filename, use_names = TRUE)

set_xmp(xmp, input, output = input)

set_xmp_exiftool(xmp, input, output = input)
```

Arguments

filename	Filename(s) to extract xmp metadata from.
use_names	If TRUE (default) use filename as the names of the result.
xmp	An xmp() object.
input	Input filename.
output	Output filename.

Details

get_xmp() will try to use the following helper functions in the following order:

1. get_xmp_exiftool() which wraps exiftool command-line tool

set_xmp() will try to use the following helper functions in the following order:

1. set_xmp_exiftool() which wraps exiftool command-line tool

Value

get_xmp() returns a list of xmp() objects. set_xmp() returns the (output) filename invisibly.

See Also

xmp() for more information about xmp metadata objects. supports_get_xmp(), supports_set_xmp(), and supports_exiftool() to detect support for these features. For more info about xmp metadata see https://www.exiftool.org/TagNames/XMP.html.

Examples

```
x <- xmp(attribution_url = "https://example.com/attribution",
         creator = "John Doe",
         description = "An image caption",
         date_created = Sys.Date(),
         spdx_id = "CC-BY-4.0")
print(x)
print(x, mode = "google_images", xmp_only = TRUE)
print(x, mode = "creative_commons", xmp_only = TRUE)

if (supports_set_xmp() &&
    supports_get_xmp() &&
    capabilities("png") &&
    requireNamespace("grid", quietly = TRUE)) {

  f <- tempfile(fileext = ".png")
  png(f)
  grid::grid.text("This is an image!")
  invisible(dev.off())
  set_xmp(x, f)
  print(get_xmp(f)[[1]])
}
```

enable_feature_message *Messages for how to enable feature*

Description

`enable_feature_message()` returns a character vector with the information needed to install the requested feature. Formatted for use with `rlang::abort()`, `rlang::warn()`, or `rlang::inform()`.

Usage

```
enable_feature_message(
  feature = c("cat_pages", "get_bookmarks", "get_docinfo", "get_xmp", "n_pages",
    "set_bookmarks", "set_docinfo", "set_xmp")
)
```

Arguments

`feature`	Which xmpdf feature to give information for.

Value

A character vector formatted for use with `rlang::abort()`, `rlang::warn()`, or `rlang::inform()`.

Examples

```
rlang::inform(enable_feature_message("get_bookmarks"))
```

n_pages *Get number of pages in a document*

Description

`n_pages()` returns the number of pages in the (pdf) file(s).

Usage

```
n_pages(filename, use_names = TRUE)

n_pages_exiftool(filename, use_names = TRUE)

n_pages_qpdf(filename, use_names = TRUE)

n_pages_pdftk(filename, use_names = TRUE)

n_pages_gs(filename, use_names = TRUE)
```

Arguments

filename	Character vector of filenames.
use_names	If TRUE (default) use filename as the names of the result.

Details

n_pages() will try to use the following helper functions in the following order:

1. n_pages_qpdf() which wraps qpdf::pdf_length()
2. n_pages_exiftool() which wraps exiftool command-line tool
3. n_pages_pdftk() which wraps pdftk command-line tool
4. n_pages_gs() which wraps ghostscript command-line tool

Value

An integer vector of number of pages within each file.

See Also

supports_n_pages() detects support for this feature.

Examples

```
if (supports_n_pages() && require("grid", quietly = TRUE)) {
  f <- tempfile(fileext = ".pdf")
  pdf(f, onefile = TRUE)
  grid.text("Page 1")
  grid.newpage()
  grid.text("Page 2")
  invisible(dev.off())
  print(n_pages(f))
  unlink(f)
}
```

---

spdx_licenses *SPDX License List data*

Description

spdx_licenses is a data frame of SPDX License List data.

Usage

```
spdx_licenses
```

Format

a data frame with eight variables:

id SPDX Identifier.

name Full name of license. For Creative Commons licenses these have been tweaked from the SPDX version to more closely match the full name used by Creative Commons Foundation.

url URL for copy of license located at `spdx.org`

fsf Is this license considered Free/Libre by the FSF?

osi Is this license OSI approved?

deprecated Has this SPDFX Identifier been deprecated by SPDX?

url_alt Alternative URL for license. Manually created for a subset of Creative Commons licenses. Others taken from https://github.com/sindresorhus/spdx-license-list.

pd Is this license a "public domain" license? Manually created.

See Also

See https://spdx.org/licenses/ for more information.

`supports` *Detect support for features*

Description

`supports_get_bookmarks()`, `supports_set_bookmarks()`, `supports_get_docinfo()`, `supports_set_docinfo()`, `supports_get_xmp()`, `supports_set_xmp()`, `supports_cat_pages()`, and `supports_n_pages()` detects support for the functions `get_bookmarks()`, `set_bookmarks()`, `get_docinfo()`, `set_docinfo()`, `get_xmp()`, `set_xmp()`, `cat_pages()`, and `n_pages()` respectively. `supports_exiftool()`, `supports_gs()` and `supports_pdftk()` detects support for the command-line tools `exiftool`, `ghostscript` and `pdftk` respectively as used by various lower-level functions.

Usage

```
supports_get_bookmarks()

supports_set_bookmarks()

supports_get_docinfo()

supports_set_docinfo()

supports_get_xmp()

supports_set_xmp()

supports_cat_pages()
```

supports_n_pages()

supports_exiftool()

supports_gs()

supports_pdftk()

Details

- supports_exiftool() detects support for the command-line tool exiftool which is required for get_docinfo_exiftool(), get_xmp_exiftool(), set_xmp_exiftool(), and n_pages_exiftool().
- supports_gs() detects support for the command-line tool ghostscript which is required for set_docinfo_gs(), set_bookmarks_gs(), cat_pages_gs(), and n_pages_gs().
- supports_pdftk() detects support for the command-line tool pdftk which is required for get_bookmarks_pdftk(), set_bookmarks_pdftk(), get_docinfo_pdftk(), set_docinfo_pdftk(), cat_pages_pdftk(), and n_pages_pdftk().
- requireNamespace("qpdf", quietly = TRUE) detects support for the R packages qpdf which is required for cat_pages_qpdf() and n_pages_qpdf().

Examples

```
# Detect for higher-level features
supports_get_docinfo()
supports_set_docinfo()
supports_get_bookmarks()
supports_set_bookmarks()
supports_get_xmp()
supports_set_xmp()
supports_cat_pages()
supports_n_pages()

# Detect support for lower-level helper features
supports_exiftool()
supports_gs()
supports_pdftk()
print(requireNamespace("qpdf", quietly = TRUE))
print(requireNamespace("pdftools", quietly = TRUE))
```

---

xmp *XMP metadata object*

Description

xmp() creates an XMP metadata object. Such objects can be used with set_xmp() to edit XMP medata for a variety of media formats and such objects are returned by get_xmp().

Usage

```
xmp(
  ...,
  alt_text = NULL,
  attribution_name = NULL,
  attribution_url = NULL,
  create_date = NULL,
  creator = NULL,
  creator_tool = NULL,
  credit = NULL,
  date_created = NULL,
  description = NULL,
  ext_description = NULL,
  headline = NULL,
  keywords = NULL,
  license = NULL,
  marked = NULL,
  modify_date = NULL,
  more_permissions = NULL,
  producer = NULL,
  rights = NULL,
  subject = NULL,
  title = NULL,
  usage_terms = NULL,
  web_statement = NULL,
  auto_xmp = c("cc:attributionName", "cc:license", "dc:rights", "dc:subject",
    "photoshop:Credit", "xmpRights:Marked", "xmpRights:UsageTerms",
    "xmpRights:WebStatement"),
  spdx_id = NULL
)
```

Arguments

`...`	Entries of xmp metadata. The names are either the xmp tag names or alternatively the xmp namespace and tag names separated by ":". The values are the xmp values.
`alt_text`	Brief textual description that can be used as its "alt text" (XMP tag `Iptc4xmpCore:AltTextAccessibili` Will be coerced by `as_lang_alt()`. Core IPTC photo metadata.
`attribution_name`	The name to be used when attributing the work (XMP tag `cc:attributionName`). Recommended by Creative Commons. If missing and `"cc:attributionName"` in `auto_xmp` and and `photoshop:Credit` non-missing will use that else if `dc:creator` non-missing then will automatically use `paste(creator, collapse = " and ")`.
`attribution_url`	The URL to be used when attributing the work (XMP tag `cc:attributionURL`). Recommended by Creative Commons.
`create_date`	The date the digital document was created (XMP tag `xmp:CreateDate`). Will be

coerced by datetimeoffset::as_datetimeoffset(). Related pdf documentation info key is CreationDate. Not to be confused with photoshop:DateCreated which is the date the intellectual content was created.

creator — The document's author(s) (XMP tag dc:creator). Related pdf documentation info key is Author. Core IPTC photo metadata used by Google Photos. If credit is missing and "photoshop:Credit" in auto_xmp then we'll also use this for the photoshop:Credit XMP tag.

creator_tool — The name of the application that originally created the document (XMP tag xmp:CreatorTool). Related pdf documentation info key is Creator.

credit — Credit line field (XMP tag photoshop:Credit). Core IPTC photo metadata used by Google Photos. If missing and "photoshop:Credit" in auto_xmp and dc:creator non-missing then will automatically use paste(creator, collapse = " and ").

date_created — The date the intellectual content was created (XMP tag photoshop:DateCreated). Will be coerced by datetimeoffset::as_datetimeoffset(). Core IPTC photo metadata. Not to be confused with xmp:CreateDate for when the digital document was created.

description — The document's subject (XMP tag dc:description). Will be coerced by as_lang_alt(). Core IPTC photo metadata. Related pdf documentation info key is Subject.

ext_description — An extended description (for accessibility) if the "alt text" is insufficient (XMP tag Iptc4xmpCore:ExtDescrAccessibility). Will be coerced by as_lang_alt(). Core IPTC photo metadata.

headline — A short synopsis of the document (XMP tag photoshop:Headline). Core IPTC photo metadata.

keywords — Character vector of keywords for this document (for cross-document searching). Related pdf documentation info key is pdf:Keywords. Will be coerced into a string by paste(keywords, collapse = ", ").

license — The URL of (open source) license terms (XMP tag cc:license). Recommended by Creative Commons. Note xmpRights:WebStatement set in web_statement is a more popular XMP tag (e.g. used by Google Images) that can also hold the URL of license terms or a verifying web statement. If cc:license in auto_xmp and spdx_id is not NULL then we'll automatically use an URL from spdx_licenses corresponding to that license.

marked — Whether the document is a rights-managed resource (XMP tag xmpRights:Marked). Use TRUE if rights-managed, FALSE if public domain, and NULL if unknown. Creative Commons recommends setting this. If xmpRights:Marked in auto_xmp and spdx_id is not NULL then we can automatically set this for a subset of SPDX licenses (including all Creative Commons licenses).

modify_date — The date the document was last modified (XMP tag xmp:ModifyDate). Will be coerced by datetimeoffset::as_datetimeoffset(). Related pdf documentation info key is ModDate.

more_permissions — A URL for additional permissions beyond the license (XMP tag cc:morePermissions). Recommended by Creative Commons. Contrast with the LicensorURL property of plus:Licensor XMP tag.

`producer` The name of the application that converted the document to pdf (XMP tag `pdf:Producer`). Related pdf documentation info key is `Producer`.

`rights` (copy)right information about the document (XMP tag `dc:rights`). Will be coerced by `as_lang_alt()`. Core IPTC photo metadata used by Google Photos that Creative Commons also recommends setting. If `dc:rights` in `auto_xmp` and `creator` and `date_created` are not `NULL` then we can automatically generate a basic copyright statement with the help of `spdx_id`.

`subject` List of description phrases, keywords, classification codes (XMP tag `dc:subject`). Core IPTC photo metadata. A character vector. Similar but less popular to the XMP tag `pdf:Keywords` which is a string. If `dc:subject` in `auto_xmp` and `keywords` is not NULL then we can automatically extract the keywords from it using `strsplit(keywords, ", ")[[1]]`.

`title` The document's title (XMP tag `dc:title`). Will be coerced by `as_lang_alt()`. Related pdf documentation info key is `Title`.

`usage_terms` A string describing legal terms of use for the document (XMP tag `xmpRights:UsageTerms`). Will be coerced by `as_lang_alt()`. Core IPTC photo metadata and recommended by Creative Commons. If `xmpRights:UsageTerms` in `auto_xmp` and `spdx_id` is not `NULL` then we can automatically set this with that license's name and URL.

`web_statement` Web Statement of Rights (XMP tag `xmpRights:WebStatement`): a string of a full URL with license information about the page. If `xmpRights:WebStatement` in `auto_xmp` and `spdx_id` is not `NULL` then we'll automatically use an URL from spdx_licenses corresponding to that license. Core IPTC photo metadata used by Google Photos. Also recommended by Creative Commons (who also recommends using a "verifying" web statement).

`auto_xmp` Character vector of XMP metadata we should try to automatically determine if missing from other XMP metadata and `spdx_id`.

`spdx_id` The id of a license in the SPDX license list. See spdx_licenses.

Value

An xmp object as can be used with `set_xmp()`. Basically a named list whose names are the (optional) xmp namespace and tag names separated by ":" and the values are the xmp values. Datetimes should be a datetime object such as `POSIXlt()`.

`xmp` R6 Class Methods

`fig_process(..., auto = c("fig.alt", "fig.cap", "fig.scap"))` Returns a function to embed XMP metadata suitable for use with {knitr}'s `fig.process` chunk option. `...` are local XMP metadata changes for this function. `auto` are which chunk options should be used to further update metadata values.

`get_item(key)` Get XMP metadata value for key `key`. Can also use the relevant active bindings to get more common values.

`print(mode = c("null_omit", "google_images", "creative_commons", "all"), xmp_only = FALSE)` Print out XMP metadata values. If `mode` is "null_omit" print out which metadata would be embedded. If `mode` is "google images" print out values for the five fields Google Images uses. If

mode is creative_commons print out the values for the fields Creative Commons recommends be set when using their licenses. If mode is all print out values for all XMP metadata that we provide active bindings for (even if NULL). If xmp_only is TRUE then don't print out spdx_id and auto_xmp values.

set_item(key, value) Set XMP metadata key key with value value. Can also use the relevant active bindings to set XMP metadata values.

update(x) Update XMP metadata entries using non-NULL entries in x coerced by as_xmp().

xmp R6 Active Bindings

alt_text The image's alt text (accessibility).

attribution_name The name to attribute the document.

attribution_url The URL to attribute the document.

create_date The date the document was created.

creator The document's author.

creator_tool The name of the application that originally created the document.

credit Credit line.

date_created The date the document's intellectual content was created

description The document's description.

ext_description An extended description for accessibility.

headline A short synopsis of document.

keywords String of keywords for this document (less popular than subject)).

license URL of (open-source) license terms the document is licensed under.

marked Boolean of whether this is a rights-managed document.

modify_date The date the document was last modified.

more_permissions URL for acquiring additional permissions beyond license.

producer The name of the application that converted the document (to pdf).

rights The document's copy(right) information.

subject Vector of key phrases/words/codes for this document (more popular than keywords)).

title The document's title.

usage_terms The document's rights usage terms.

web_statement A URL string for the web statement of rights for the document.

spdx_id The id of a license in the SPDX license list. See spdx_licenses.

auto_xmp Character vector of XMP metadata we should try to automatically determine if missing from other XMP metadata and spdx_id.

XMP tag recommendations

- https://exiftool.org/TagNames/XMP.html recommends "dc", "xmp", "Iptc4xmpCore", and "Iptc4xmpExt" schemas if possible
- https://github.com/adobe/xmp-docs/tree/master/XMPNamespaces are descriptions of some common XMP tags
- https://www.iptc.org/std/photometadata/specification/IPTC-PhotoMetadata#xmp-namespaces-and-ider is popular for photos
- https://developers.google.com/search/docs/appearance/structured-data/image-license-metadata#iptc-photo-metadata are the subset of IPTC photo metadata which Google Photos uses (if no structured data on web page)
- https://wiki.creativecommons.org/wiki/XMP are Creative Commons license recommendations

See Also

get_xmp() and set_xmp() for getting/setting such information from/to a variety of media file formats. as_xmp() for coercing to this object. as_docinfo() can be used to coerce xmp() objects into docinfo() objects.

Examples

```
x <- xmp(attribution_url = "https://example.com/attribution",
         creator = "John Doe",
         description = "An image caption",
         date_created = Sys.Date(),
         spdx_id = "CC-BY-4.0")
print(x)
print(x, mode = "google_images", xmp_only = TRUE)
print(x, mode = "creative_commons", xmp_only = TRUE)

if (supports_set_xmp() &&
    supports_get_xmp() &&
    capabilities("png") &&
    requireNamespace("grid", quietly = TRUE)) {

  f <- tempfile(fileext = ".png")
  png(f)
  grid::grid.text("This is an image!")
  invisible(dev.off())
  set_xmp(x, f)
  print(get_xmp(f)[[1]])
}
```

xmpdf *xmpdf: Edit 'XMP' Metadata and 'PDF' Bookmarks and Documentation Info*

Description

Edit 'XMP' metadata https://en.wikipedia.org/wiki/Extensible_Metadata_Platform in a variety of media file formats as well as edit bookmarks (aka outline aka table of contents) and documentation info entries in 'pdf' files. Can detect and use a variety of command-line tools to perform these operations such as 'exiftool' https://exiftool.org/, 'ghostscript' https://www.ghostscript.com/, and/or 'pdftk' https://gitlab.com/pdftk-java/pdftk.

Package options

The following xmpdf option may be set globally via base::options():

xmpdf_default_lang Set new default default_lang argument value for as_lang_alt().

Author(s)

Maintainer: Trevor L Davis <trevor.l.davis@gmail.com> (ORCID)

Other contributors:

- W. Trevor King Some pdf code adapted from 'pdf-merge.py' [contributor]
- Linux Foundation (Uses some data from the "SPDX License List" <https://github.com/spdx/license-list-XML>) [data contributor]

See Also

Useful links:

- https://trevorldavis.com/R/xmpdf/dev/
- Report bugs at https://github.com/trevorld/r-xmpdf/issues

Index