Package: dicomr (via r-universe)

November 26, 2026

Type Package

Title Read, Write and Decode DICOM Medical Image Files

Version 0.1.0

Description A native reader and writer for DICOM (Digital Imaging and Communications in Medicine) files, the standard format for medical images, published by the National Electrical Manufacturers Association <https://www.dicomstandard.org/current>. Modeled on the 'Python' package 'pydicom' and checked against it. Parses the file format in C++ (implicit and explicit value representations, both byte orders, deflated data sets, nested sequences and encapsulated pixel data), gives keyword and tag based access to data elements, writes data sets back to disk, and decodes pixel data into R arrays: uncompressed, RLE Lossless and JPEG Baseline (through 'libjpeg'), with YBR to RGB color conversion. Includes helpers for cardiac and other ultrasound, such as calibration from the Sequence of Ultrasound Regions and frame timing for cine loops.

License MIT + file LICENSE

Encoding UTF-8

LinkingTo Rcpp

Imports graphics, grDevices, Rcpp (>= 1.0.0), tools

Suggests jsonlite, testthat (>= 3.0.0)

SystemRequirements libjpeg (deb: libjpeg-dev, rpm: libjpeg-turbo-devel, brew: jpeg-turbo), zlib (deb: zlib1g-dev, rpm: zlib-devel)

Config/testthat/edition 3

RoxygenNote 7.3.1

NeedsCompilation yes

Author Matt Segar [aut, cre, cph], Darcy Mason [ctb, cph] (author of 'pydicom', from which the DICOM data dictionary is derived), pydicom contributors [ctb, cph] (DICOM data dictionary derived from 'pydicom')

Maintainer Matt Segar <cscswimmer227@gmail.com>

Config/pak/sysreqs libjpeg-dev zlib1g-dev

Repository https://cran.r-universe.dev

Date/Publication 2026-11-26 20:18:13 UTC

RemoteUrl https://github.com/cran/dicomr

RemoteRef HEAD

RemoteSha ecd87bb2ff364b0154f3e088f82bd078f591bf20

Contents

as.data.frame.dicom_dataset . . . 2
dcm_apply_rescale . . . 3
dcm_compare_pydicom . . . 4
dcm_file_meta . . . 5
dcm_frame_times . . . 5
dcm_get . . . 6
dcm_keywords . . . 7
dcm_pixel_array . . . 7
dcm_pixel_info . . . 9
dcm_set_pixel_data . . . 9
dcm_show . . . 10
dcm_tag . . . 11
dcm_tag_string . . . 11
dcm_us_distance . . . 12
dcm_us_pixel_to_physical . . . 13
dcm_us_regions . . . 13
dicom_dataset . . . 14
dicom_dictionary . . . 15
read_dicom . . . 15
write_dicom . . . 16

Index **18**

`as.data.frame.dicom_dataset`

Flatten a data set into a data frame

Description

One row per element, with sequence items flattened and marked by ‘depth‘ and ‘path‘. Values are formatted as text in the ‘value‘ column.

Usage

```
## S3 method for class 'dicom_dataset'
as.data.frame(x, row.names = NULL, optional = FALSE, ...)
```

Arguments

x	A ‘dicom_dataset‘.
row.names, optional	Unused, for compatibility with the generic.
...	Unused.

Value

A data frame with columns ‘tag‘, ‘keyword‘, ‘name‘, ‘vr‘, ‘vm‘, ‘value‘, ‘depth‘ and ‘path‘.

Examples

```
echo <- read_dicom(system.file("extdata", "echo_synthetic.dcm", package = "dicomr"))
df <- as.data.frame(echo)
head(df[, c("tag", "keyword", "vr", "value")])
```

---

dcm_apply_rescale *Apply the modality rescale (slope and intercept)*

Description

Converts stored values to output units such as Hounsfield units, using Rescale Slope (0028,1053) and Rescale Intercept (0028,1052).

Usage

```
dcm_apply_rescale(arr, ds)
```

Arguments

arr	An array from [dcm_pixel_array()].
ds	The data set it came from.

Value

A double array.

Examples

```
ct <- read_dicom(system.file("extdata", "ct_synthetic.dcm", package = "dicomr"))
hu <- dcm_apply_rescale(dcm_pixel_array(ct), ct)
range(hu)
```

`dcm_compare_pydicom` *Compare dicomr's reading of a file with pydicom's*

Description

Runs pydicom (through Python) on the same file and compares every data element, including nested sequence items, and the decoded pixel array. Useful for checking your own files. Requires Python 3 with 'pydicom' and 'numpy' installed, plus the 'jsonlite' package.

Usage

```
dcm_compare_pydicom(
  file,
  python = "python3",
  pixels = TRUE,
  jpeg_plugin = "pillow",
  ...
)
```

Arguments

`file`	Path to a DICOM file.
`python`	The Python executable.
`pixels`	Also compare the decoded pixel arrays.
`jpeg_plugin`	pydicom decoding plugin for JPEG Baseline/Extended data: '"pillow"' (the default, which 'jpeg_mode = "pydicom"' reproduces) or '"pylibjpeg"', or '"default"' to let pydicom choose.
`...`	Passed to [dcm_pixel_array()], for example 'jpeg_mode'.

Value

A list with 'elements' (number compared), 'issues' (a data frame of differences), and 'pixels': a list with 'status' ("equal", "close", "different", "pydicom_failed", "dicomr_failed", "both_failed" or "none"), 'max_abs_diff', 'fraction_equal' and any error messages.

Examples

```
# Needs Python 3 with pydicom and numpy, so it is not run here
## Not run:
r <- dcm_compare_pydicom(system.file("extdata", "ct_synthetic.dcm", package = "dicomr"))
r$issues
r$pixels$status

## End(Not run)
```

dcm_file_meta *File level information of a data set read from disk*

Description

File level information of a data set read from disk

Usage

```
dcm_file_meta(ds)

dcm_transfer_syntax(ds)
```

Arguments

ds	A ‘dicom_dataset‘ from [read_dicom()].

Value

‘dcm_file_meta()‘ returns the File Meta Information (group 0002) as a ‘dicom_dataset‘. ‘dcm_transfer_syntax()‘ returns the transfer syntax UID with its name as the ‘name‘ attribute.

Examples

```
ct <- read_dicom(system.file("extdata", "ct_synthetic.dcm", package = "dicomr"))
dcm_transfer_syntax(ct)
dcm_file_meta(ct)
```

dcm_frame_times *Frame times for a cine loop*

Description

Returns the time of each frame in milliseconds from the first frame, from Frame Time Vector (0018,1065) if present, otherwise Frame Time (0018,1063), otherwise Cine Rate (0018,0040) or Recommended Display Frame Rate.

Usage

```
dcm_frame_times(ds)
```

Arguments

ds	A ‘dicom_dataset‘.

Value

A numeric vector with one time per frame (ms), or ‘NULL‘ if the file has no timing information. The frame rate (frames per second) is in the ‘frame_rate‘ attribute.

Examples

```
echo <- read_dicom(system.file("extdata", "echo_synthetic.dcm", package = "dicomr"))
dcm_frame_times(echo)
```

`dcm_get` *Get or set data elements*

Description

‘dcm_get()‘ returns an element’s value, ‘dcm_element()‘ the whole element (group, element, VR and value), ‘dcm_has()‘ tests for presence, and ‘dcm_set()‘ adds or replaces an element. The usual R operators also work: ‘ds$PatientName‘, ‘ds[["PatientName"]]‘, ‘ds[["0010,0010"]]‘, ‘ds[[c(0x0010, 0x0010)]]‘, and the same forms on the left of ‘<-‘. Assigning ‘NULL‘ removes the element.

Usage

```
dcm_get(ds, tag, default = NULL)

dcm_element(ds, tag)

dcm_has(ds, tag)

dcm_set(ds, tag, value, vr = NULL)
```

Arguments

`ds`	A ‘dicom_dataset‘.
`tag`	A keyword or tag (see [dcm_tag()]).
`default`	Returned when the element is absent.
`value`	The new value.
`vr`	The value representation. Taken from the existing element or the dictionary when missing; required for private tags.

Value

‘dcm_get()‘ the value; ‘dcm_element()‘ a ‘dicom_element‘ or ‘NULL‘; ‘dcm_has()‘ a logical; ‘dcm_set()‘ the modified data set.

Examples

```
ct <- read_dicom(system.file("extdata", "ct_synthetic.dcm", package = "dicomr"))
dcm_get(ct, "Modality")
dcm_get(ct, "(0028,1050)")   # Window Center
dcm_get(ct, "BodyPartExamined", default = NA)
dcm_has(ct, "PixelData")
dcm_element(ct, "PixelSpacing")
ct <- dcm_set(ct, "BodyPartExamined", "HEAD")
ct$PatientName <- "Anonymous"
ct$PatientName
```

---

dcm_keywords *Keywords of the elements in a data set*

Description

Keywords of the elements in a data set

Usage

```
dcm_keywords(ds)
```

Arguments

ds	A ‘dicom_dataset‘.

Value

A character vector (empty strings for private or unknown tags).

Examples

```
ct <- read_dicom(system.file("extdata", "ct_synthetic.dcm", package = "dicomr"))
head(dcm_keywords(ct))
```

---

dcm_pixel_array *Decode pixel data into an R array*

Description

Returns pixel data with the same axis order as pydicom’s ‘pixel_array‘: ‘frames x rows x columns x samples‘. With ‘drop = TRUE‘ (the default) the frame axis is dropped for single-frame images and the sample axis is dropped for monochrome images, so a single-frame greyscale image is a ‘rows x columns‘ matrix and a 30-frame colour echo clip is a ‘30 x rows x columns x 3‘ array. ‘arr[f, r, c, s]‘ in R is ‘arr[f-1, r-1, c-1, s-1]‘ in Python.

Usage

```
dcm_pixel_array(
  ds,
  frames = NULL,
  as_rgb = TRUE,
  drop = TRUE,
  correct_unused_bits = FALSE,
  jpeg_mode = c("pydicom", "accurate")
)
```

Arguments

ds	A ‘dicom_dataset‘ from [read_dicom()].
frames	Optional 1-based frame numbers to decode, for example ‘1:10‘. Only these frames are decoded, which saves time and memory on long cine loops.
as_rgb	Convert YBR colour data to RGB.
drop	Drop length-one frame and sample axes as pydicom does.
correct_unused_bits	Clear (or sign-extend) the bits above Bits Stored. pydicom’s default is ‘FALSE‘.
jpeg_mode	How to decode colour JPEG. ‘"pydicom"‘ (the default) reproduces pydicom with its Pillow plugin exactly: libjpeg-turbo decodes without a colour transform, then pydicom’s YBR to RGB formula is applied. Use it when results must match a Python pipeline, such as a model trained on pydicom output. ‘"accurate"‘ lets libjpeg-turbo do the colour conversion in integer arithmetic; values can differ from ‘"pydicom"‘ by a grey level or two.

Details

Supported encodings: uncompressed (implicit/explicit VR, little/big endian, deflated), RLE Lossless, and 8-bit JPEG Baseline / Extended through libjpeg-turbo. YBR_FULL and YBR_FULL_422 data are converted to RGB when ‘as_rgb = TRUE‘, using the same float formula as pydicom.

The values are the stored values. Apply [dcm_apply_rescale()] for Hounsfield units and similar.

Value

An integer array (double for 32-bit unsigned or float data), with attribute ‘photometric_interpretation‘ giving the colour space of the returned values.

Examples

```
echo <- read_dicom(system.file("extdata", "echo_synthetic.dcm", package = "dicomr"))
clip <- dcm_pixel_array(echo)               # frames x rows x cols x 3
dim(clip)
first <- dcm_pixel_array(echo, frames = 1)  # rows x cols x 3
dim(first)
ct <- read_dicom(system.file("extdata", "ct_synthetic.dcm", package = "dicomr"))
dim(dcm_pixel_array(ct))                    # rows x cols
```

dcm_pixel_info *Image geometry and pixel description*

Description

Image geometry and pixel description

Usage

```
dcm_pixel_info(ds)
```

Arguments

ds	A ‘dicom_dataset‘.

Value

A list with ‘rows‘, ‘columns‘, ‘samples_per_pixel‘, ‘frames‘, ‘bits_allocated‘, ‘bits_stored‘, ‘pixel_representation‘, ‘planar_configuration‘, ‘photometric_interpretation‘ and ‘transfer_syntax‘.

Examples

```
echo <- read_dicom(system.file("extdata", "echo_synthetic.dcm", package = "dicomr"))
str(dcm_pixel_info(echo))
```

dcm_set_pixel_data *Store an R array as uncompressed pixel data*

Description

The inverse of [dcm_pixel_array()]: writes ‘arr‘ into Pixel Data and sets Rows, Columns, Samples per Pixel, Number of Frames, Bits Allocated/Stored, High Bit, Pixel Representation, Photometric Interpretation and Planar Configuration to match. The data set’s transfer syntax becomes Explicit VR Little Endian if it was compressed.

Usage

```
dcm_set_pixel_data(
  ds,
  arr,
  photometric_interpretation = NULL,
  bits_allocated = NULL
)
```

Arguments

ds	A ‘dicom_dataset‘.
arr	An integer array: ‘rows x cols‘ (one greyscale frame), ‘frames x rows x cols‘ (greyscale cine), ‘rows x cols x 3‘ (one colour frame) or ‘frames x rows x cols x 3‘.
photometric_interpretation	‘"MONOCHROME2"‘, ‘"MONOCHROME1"‘ or ‘"RGB"‘. Guessed from the array when missing.
bits_allocated	8, 16 or 32. Chosen from the value range when missing.

Value

The modified data set.

Examples

```
img <- matrix(as.integer(seq(0, 4095, length.out = 64 * 64)), 64, 64)
ds <- dicom_dataset(Modality = "OT")
ds <- dcm_set_pixel_data(ds, img)
ds$BitsAllocated
identical(as.vector(dcm_pixel_array(ds)), as.vector(img))
```

---

dcm_show *Show a frame*

Description

A quick look at one frame with base graphics.

Usage

```
dcm_show(ds, frame = 1, ...)
```

Arguments

ds	A ‘dicom_dataset‘.
frame	Frame number (1-based).
...	Passed to [graphics::plot()] (colour) or [graphics::image()] (greyscale).

Value

‘NULL‘, invisibly.

Examples

```
echo <- read_dicom(system.file("extdata", "echo_synthetic.dcm", package = "dicomr"))
dcm_show(echo, frame = 2)
ct <- read_dicom(system.file("extdata", "ct_synthetic.dcm", package = "dicomr"))
dcm_show(ct)
```

dcm_tag *Convert a tag specification to a DICOM tag*

Description

Accepts a keyword (‘"PatientName"‘), a tag string (‘"(0010,0010)"‘, ‘"0010,0010"‘, ‘"00100010"‘, ‘"0x00100010"‘), a single number (‘0x00100010‘) or a group/element pair (‘c(0x0010, 0x0010)‘).

Usage

```
dcm_tag(x)

dcm_tag_number(x)
```

Arguments

x	A tag specification.

Value

‘dcm_tag()‘ returns the tag as an 8-character hex string such as ‘"00100010"‘, which is how elements are named inside a data set. ‘dcm_tag_number()‘ returns the tag as a number.

Examples

```
dcm_tag("PatientName")
dcm_tag(c(0x0028, 0x0010))
dcm_tag("(7FE0,0010)")
dcm_tag_number("Rows")
```

dcm_tag_string *Format a tag as "(gggg,eeee)"*

Description

Format a tag as "(gggg,eeee)"

Usage

```
dcm_tag_string(x)
```

Arguments

x	A tag specification (see [dcm_tag()]).

Value

A character string.

Examples

```
dcm_tag_string("PatientName")
dcm_tag_string(0x7FE00010)
```

`dcm_us_distance` *Distance between two points in physical units*

Description

For 2D regions where both axes share units (cm in echo), returns the straight-line distance, for example a caliper measurement of an LV dimension. Both points must fall in the same region.

Usage

```
dcm_us_distance(ds, p1, p2, region = NULL)
```

Arguments

`ds`	A ‘dicom_dataset‘ or a data frame from [dcm_us_regions()].
`p1, p2`	Points as ‘c(row, col)‘, 1-based array indices.
`region`	Optional region number.

Value

A number with a ‘units‘ attribute.

Examples

```
echo <- read_dicom(system.file("extdata", "echo_synthetic.dcm", package = "dicomr"))
dcm_us_distance(echo, p1 = c(row = 5, col = 10), p2 = c(row = 25, col = 40))
```

dcm_us_pixel_to_physical *Convert pixel positions to physical coordinates*

Description

Uses the ultrasound region that contains each point (or the one you name) to turn pixel positions into physical values: ‘physical = reference_physical + (pixel - (region_origin + reference_pixel)) * delta‘.

Usage

```
dcm_us_pixel_to_physical(ds, row, col, region = NULL)
```

Arguments

ds	A ‘dicom_dataset‘, or a data frame from [dcm_us_regions()].
row, col	Pixel positions as R array indices (1-based), matching ‘arr[row, col]‘ on a frame from [dcm_pixel_array()].
region	Optional region number to use for every point.

Value

A data frame with ‘row‘, ‘col‘, ‘region‘, ‘x‘, ‘y‘, ‘units_x‘ and ‘units_y‘; ‘x‘ runs across the image and ‘y‘ down it.

Examples

```
echo <- read_dicom(system.file("extdata", "echo_synthetic.dcm", package = "dicomr"))
dcm_us_pixel_to_physical(echo, row = c(1, 11), col = c(33, 33))
```

dcm_us_regions *Ultrasound region calibration*

Description

Parses the Sequence of Ultrasound Regions (0018,6011) into a data frame, one row per region. Each region says where it sits in the image (in 0-based pixel coordinates, as in the standard), what it shows (2D tissue, colour flow, spectral Doppler, M-mode, ECG trace...), and how pixels map to physical units: ‘delta_x‘ and ‘delta_y‘ are the physical size of one pixel in ‘units_x‘ and ‘units_y‘ (for 2D echo usually cm per pixel; for spectral Doppler usually seconds per pixel across and cm/sec per pixel up).

Usage

```
dcm_us_regions(ds)
```

Arguments

ds	A ‘dicom_dataset‘.

Value

A data frame (zero rows if the file has no regions).

Examples

```
echo <- read_dicom(system.file("extdata", "echo_synthetic.dcm", package = "dicomr"))
reg <- dcm_us_regions(echo)
reg[, c("region", "data_type_label", "delta_x", "units_x_label")]
```

---

dicom_dataset *Create an empty data set or a sequence item*

Description

Create an empty data set or a sequence item

Usage

```
dicom_dataset(...)
```

Arguments

...	Named values, using DICOM keywords as names.

Value

A ‘dicom_dataset‘.

Examples

```
item <- dicom_dataset(PhysicalDeltaX = 0.03, PhysicalUnitsXDirection = 3L)
ds <- dicom_dataset(Modality = "US", SequenceOfUltrasoundRegions = list(item))
ds
```

dicom_dictionary *Look up tags or keywords in the DICOM data dictionary*

Description

Look up tags or keywords in the DICOM data dictionary

Usage

```
dicom_dictionary(x)
```

Arguments

x	Character keywords or tag specifications. If missing, the whole dictionary is returned.

Value

A data frame with columns ‘tag‘, ‘vr‘, ‘vm‘, ‘name‘, ‘keyword‘ and ‘retired‘.

Examples

```
dicom_dictionary(c("PatientName", "PhysicalDeltaX", "(7FE0,0010)"))
nrow(dicom_dictionary())
```

read_dicom *Read a DICOM file*

Description

Parses a DICOM Part 10 file (or a raw data set without the 128-byte preamble when ‘force = TRUE‘) into a ‘dicom_dataset‘.

Usage

```
read_dicom(file, stop_before_pixels = FALSE, force = FALSE)
```

Arguments

file	Path to the file, or a raw vector holding the file's bytes.
stop_before_pixels	If ‘TRUE‘, stop reading at the Pixel Data element, which is much faster when you only need the header.
force	Read files that lack the "DICM" prefix.

Details

Values are converted to R types following pydicom: text VRs become character vectors (split on backslash), ‘IS‘ becomes integer, ‘DS‘, ‘FL‘, ‘FD‘, ‘UL‘ become double, ‘US‘, ‘SS‘, ‘SL‘ become integer, ‘AT‘ becomes a hex tag string, ‘SQ‘ becomes a list of data sets, and ‘OB‘/‘OW‘/‘UN‘ and other binary VRs stay as raw vectors. Encapsulated (compressed) pixel data is kept as its fragments; decode it with [dcm_pixel_array()].

Value

A ‘dicom_dataset‘. File level information is available with [dcm_file_meta()] and [dcm_transfer_syntax()].

Examples

```
f <- system.file("extdata", "echo_synthetic.dcm", package = "dicomr")
ds <- read_dicom(f)
ds$Modality
ds[["(0028,0008)"]]          # Number of Frames, by tag
hdr <- read_dicom(f, stop_before_pixels = TRUE)
dcm_has(hdr, "PixelData")
```

---

write_dicom *Write a data set to a DICOM file*

Description

Writes a DICOM Part 10 file: 128-byte preamble, "DICM", the File Meta Information group (rebuilt with a correct group length), then the data set. Sequences are written with undefined lengths. Compressed pixel data is written back as it was read; changing between compressed and uncompressed transfer syntaxes is not supported.

Usage

```
write_dicom(ds, file, transfer_syntax = NULL)
```

Arguments

ds	A ‘dicom_dataset‘.
file	Output path.
transfer_syntax	Transfer syntax UID. Defaults to the one the data set was read with, or Explicit VR Little Endian. Uncompressed data sets can be written as Implicit VR Little Endian (‘"1.2.840.10008.1.2"‘), Explicit VR Little Endian (‘"1.2.840.10008.1.2.1"‘), Explicit VR Big Endian (‘"1.2.840.10008.1.2.2"‘) or Deflated (‘"1.2.840.10008.1.2.1.99"‘).

Value

The path, invisibly.

Examples

```
ct <- read_dicom(system.file("extdata", "ct_synthetic.dcm", package = "dicomr"))
out <- tempfile(fileext = ".dcm")
write_dicom(ct, out, transfer_syntax = "1.2.840.10008.1.2.1")
dcm_transfer_syntax(read_dicom(out))
unlink(out)
```

Index

as.data.frame.dicom_dataset, 2

dcm_apply_rescale, 3
dcm_compare_pydicom, 4
dcm_element *(dcm_get)*, 6
dcm_file_meta, 5
dcm_frame_times, 5
dcm_get, 6
dcm_has *(dcm_get)*, 6
dcm_keywords, 7
dcm_pixel_array, 7
dcm_pixel_info, 9
dcm_set *(dcm_get)*, 6
dcm_set_pixel_data, 9
dcm_show, 10
dcm_tag, 11
dcm_tag_number *(dcm_tag)*, 11
dcm_tag_string, 11
dcm_transfer_syntax *(dcm_file_meta)*, 5
dcm_us_distance, 12
dcm_us_pixel_to_physical, 13
dcm_us_regions, 13
dicom_dataset, 14
dicom_dictionary, 15

read_dicom, 15

write_dicom, 16