

Working with the DICOM Data Standard in R

Brandon Whitcher
Pfizer Worldwide R&D

Volker J. Schmid
Ludwig-Maximilians Universität München

Andrew Thornton
Cardiff University

Abstract

The package **oro.dicom** facilitates the interaction with and manipulation of medical imaging data that conform to the DICOM standard. DICOM data, from a single file or single directory or directory tree, may be uploaded into R using basic data structures: a data frame for the header information and a matrix for the image data. A list structure is used to organize multiple DICOM files. The conversion from DICOM to ANALYZE/NIFTI is straightforward using the capabilities of **oro.dicom** and **oro.nifti**.

Keywords: export, imaging, import, medical, visualization.

1. Introduction

Medical imaging is well established in both the clinical and research areas with numerous equipment manufacturers supplying a wide variety of modalities. The DICOM (Digital Imaging and Communications in Medicine; <http://medical.nema.org>) standard was developed from earlier standards and released in 1993. It is the data format for clinical imaging equipment and a variety of other devices whose complete specification is beyond the scope of this paper. All major manufacturers of medical imaging equipment (e.g., GE, Siemens, Philips) have so-called DICOM conformance statements that explicitly state how their hardware implements DICOM. The DICOM standard provides interoperability across hardware, but was not designed to facilitate efficient data manipulation and image processing. Hence, additional data formats have been developed over the years to accommodate data analysis and image processing.

The material presented here provides users with a method of interacting with DICOM files in R ([R Development Core Team 2010](#)). Real-world data sets, that are publicly available, are used to illustrate the basic functionality of **oro.dicom** ([Whitcher et al. 2011](#)). It should be noted that the package focuses on functions for data input/output and visualization. Images in the metadata-rich DICOM format may be converted to NIFTI semi-automatically using **oro.nifti** by utilizing as much information from the DICOM files as possible. Basic visualization functions, similar to those commonly used in the medical imaging community, are provided for **nifti**. Additionally, the **oro.nifti** package allows one to track every operation on a **nifti** object in an XML-based audit trail.

The **oro.dicom** package should appeal not only to R package developers, but also to scientists

oro.dicom	
<code>create3D, create4D</code>	Create multi-dimensional arrays from DICOM header/image lists.
<code>dicom2analyze, dicom2nifti</code>	Convert DICOM objects to ANALYZE or Nifti objects.
<code>readDICOMFile, readDICOM</code>	Read single or multiple DICOM files into R.
<code>dicomTable, writeHeader</code>	Construct data frame from DICOM header list and write to a CSV file.
<code>extractHeader, header2matrix, matchHeader</code>	Extract information from DICOM headers.
<code>str2date, str2time</code>	Convert DICOM date or time entry into an R object.

Table 1: List of functions available in **oro.dicom**.

and researchers who want to interrogate medical imaging data using the statistical capabilities of R without writing and validating their own basic data input/output functionality. Table 1 lists the key functions for **oro.dicom** and groups them according to common functionality.

2. oro.dicom: DICOM data input/output in R

The DICOM “standard” for data acquired using a clinical imaging device is very broad and complex. Roughly speaking each DICOM-compliant file is a collection of fields organized into two two-byte sequences (group,element) that are represented as hexadecimal numbers and form a tag. The (group,element) combination establishes what type of information is forthcoming in the file. There is no fixed number of bytes for a DICOM header. The final (group,element) tag should be the “pixel data” tag (7FE0,0010), such that all subsequent information is related to the image(s).

All attributes in the DICOM standard require different data types for correct representation. These are known as value representations (VRs) in DICOM, which may be encoded explicitly or implicitly. There are 27 explicit VRs defined in the DICOM standard. Detailed explanations of these data types are provided in the Section 6.2 (part 5) of the DICOM standard (<http://medical.nema.org>). Internal functions have been written to manipulate each of the value representations and are beyond the scope of this article. The functions `str2date` and `str2time` are useful for converting from the DICOM `Datetime` and `Time` value representations to R date and time objects, respectively.

2.1. The DICOM header

Accessing the information stored in a single DICOM file is provided using the `readDICOMFile` function. The basic structure of a DICOM file is summarized in Figure 1, for both explicit and implicit value representations. The first two bytes represent the `group` tag and the second two bytes represent the `element` tag, regardless of the type of VR. The third set of two bytes contains the characters of the VR on which a decision about being implicit or explicit is made.

Data element with explicit VR of OB, OF, OW, SQ, UT or UN:

```

+-----+
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 |
+-----+
|<Group-->|<Element>|<VR----->|<0x0000->|<Length----->|<Value->

```

Data element with explicit VR other than as shown above:

```

+-----+
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
+-----+
|<Group-->|<Element>|<VR----->|<Length->|<Value->

```

Data element with implicit VR:

```

+-----+
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
+-----+
|<Group-->|<Element>|<Length----->|<Value->

```

Figure 1: Byte ordering for a single (group,element) tag in the DICOM standard. Explicit VRs store the VR as text characters in two bytes. More information is provided in Section 7, Part 3.5-2009 of the DICOM standard (<http://medical.nema.org>).

Explicit VRs of type (OB, OF, OW, SQ, UT, UN) skip bytes six and seven (counting from zero), convert the next four bytes into an integer `length` and read `length` number of objects from the DICOM file. All other explicit VRs follow a slightly different path where bytes six and seven (counting from zero) provide an integer `length` and all remaining bytes are read in as the `value`. If the character string in bytes four and five do not correspond to a known VR (Figure 1), then the (group,element) tag is declared to be implicit, the `length` is taken from bytes four through seven and all remaining bytes contribute to the `value`.

The basic structure of the resulting object is a list with two elements: the DICOM header (`hdr`) and the DICOM image (`img`). The header information is organized in a data frame with six columns and an unknown number of rows depending on the input parameters.

```
R> fname <- system.file(file.path("dcm", "Abdo.dcm"), package="oro.dicom")
R> abdo <- readDICOMFile(fname)
R> names(abdo)
```

```
[1] "hdr" "img"
```

```
R> head(abdo$hdr)
```

	group	element	name	code	length
1	0002	0000	GroupLength	UL	4
2	0002	0001	FileMetaInformationVersion	OB	2
3	0002	0002	MediaStorageSOPClassUID	UI	26
4	0002	0003	MediaStorageSOPInstanceUID	UI	38
5	0002	0010	TransferSyntaxUID	UI	20
6	0002	0012	ImplementationClassUID	UI	16
			value	sequence	
1			166		
2			\001		
3			1.2.840.10008.5.1.4.1.1.4		
4	1.3.46.670589.11.0.4.1996082307380007				
5			1.2.840.10008.1.2.1		
6			1.3.46.670589.17		

```
R> tail(abdo$hdr)
```

	group	element	name	code	length	value	sequence
79	0028	0101	BitsStored	US	2	12	
80	0028	0102	HighBit	US	2	11	
81	0028	0103	PixelRepresentation	US	2	0	
82	0028	1050	WindowCenter	DS	4	530	
83	0028	1051	WindowWidth	DS	4	1052	
84	7FE0	0010	PixelData	OB	131072	PixelData	

The ordering of the rows is identical to the ordering in the original DICOM file. Hence, the first five tags in the DICOM header of `Abdo.dcm` are: `GroupLength`, `FileMetaInformationVersion`,

MediaStorageSOPClassUID, MediaStorageSOPInstanceUID and TransferSyntaxUID. The last five tags in the DICOM header are also shown, with the very last tag indicating the start of the image data for that file and the number of bytes (131072) involved. When additional tags in the DICOM header information are queried (via `extractHeader`)

```
R> extractHeader(abdo$hdr, "BitsAllocated")
```

```
[1] 16
```

```
R> extractHeader(abdo$hdr, "Rows")
```

```
[1] 256
```

```
R> extractHeader(abdo$hdr, "Columns")
```

```
[1] 256
```

it is clear that the data are consistent with the header information in terms of the number of bytes ($256 \times 256 \times (16/8) = 131072$).

The first five columns are taken directly from the DICOM header information (`group`, `element`, `code`, `length` and `value`) or inferred from that information (`name`). Note, the (`group`, `element`) values are stored as character strings even though they are hexadecimal numbers. All aspects of the data frame may be interrogated in R in order to extract relevant information from the DICOM header; e.g., "BitsAllocated" as above. The `sequence` column is used to keep track of tags that are embedded in a fixed-length `SequenceItems` tag or between a `SequenceItem-SequenceDelimitationItem` pair.

When multiple DICOM files are located in a single directory, or spread across multiple directories, one may use the function `readDICOM` (applied here to the directory `hk-40`).

```
R> fname <- system.file("hk-40", package="oro.dicom")
```

```
R> data(dicom.dic)
```

```
R> hk40 <- readDICOM(fname)
```

```
R> unlist(lapply(hk40, length))
```

```
hdr img
```

```
40 40
```

The object associated with `readDICOM` is now a nested set of lists, where the `hdr` element is a list of data frames and the `img` element is a list of matrices. These two lists are associated in a pairwise sense; i.e., `hdr[[1]]` is the header information for the image `img[[1]]`. Default parameters `recursive = TRUE` and `pixelData = TRUE` (which is actually an input parameter for `readDICOMFile`) allow the user to search down all possible sub-directories and upload the image in addition to the header information, respectively. Also, by default all files are treated as DICOM files unless the `exclude` parameter is set to the unwanted file extension; e.g., `exclude = "xml"`.

The list of DICOM header information across multiple files may be converted to a single data frame using `dicomTable`, and written to disc for further analysis; e.g., using `write.csv`.

```
R> hk40.info <- dicomTable(hk40$hdr)
R> write.csv(hk40.info, file="hk40_header.csv")
R> sliceloc.col <- which(hk40$hdr[[1]]$name == "SliceLocation")
R> sliceLocation <- as.numeric(hk40.info[, sliceloc.col])
R> head(sliceLocation)
```

```
[1] 160.9315 157.8315 154.7315 151.6315 148.5315 145.4315
```

```
R> head(diff(sliceLocation))
```

```
[1] -3.1 -3.1 -3.1 -3.1 -3.1 -3.1
```

```
R> unique(extractHeader(hk40$hdr, "SliceThickness"))
```

```
[1] 3.125
```

The tag `SliceLocation` is extracted from the DICOM header information (at the first element in the list) and processed using the `diff` function, and should agree with the `SliceThickness` tag. Single DICOM fields may also be extracted from the list of DICOM header information that contain attributes that are crucial for further image processing; e.g., extracting relevant MR sequences or acquisition timings.

```
R> head(extractHeader(hk40$hdr, "SliceLocation"))
```

```
[1] 160.9315 157.8315 154.7315 151.6315 148.5315 145.4315
```

```
R> modality <- extractHeader(hk40$hdr, "Modality", numeric=FALSE)
```

```
R> head(matchHeader(modality, "mr"))
```

```
[1] TRUE TRUE TRUE TRUE TRUE TRUE
```

```
R> (seriesTime <- extractHeader(hk40$hdr, "SeriesTime", numeric=FALSE))
```

```
[1] "113751.966000" "113751.966000" "113751.966000" "113751.966000"
[5] "113751.966000" "113751.966000" "113751.966000" "113751.966000"
[9] "113751.966000" "113751.966000" "113751.966000" "113751.966000"
[13] "113751.966000" "113751.966000" "113751.966000" "113751.966000"
[17] "113751.966000" "113751.966000" "113751.966000" "113751.966000"
[21] "113751.966000" "113751.966000" "113751.966000" "113751.966000"
[25] "113751.966000" "113751.966000" "113751.966000" "113751.966000"
[29] "113751.966000" "113751.966000" "113751.966000" "113751.966000"
[33] "113751.966000" "113751.966000" "113751.966000" "113751.966000"
[37] "113751.966000" "113751.966000" "113751.966000" "113751.966000"
```

```
R> str2time(seriesTime)
```

```
$txt
[1] "11:37:51.96600" "11:37:51.96600" "11:37:51.96600"
[4] "11:37:51.96600" "11:37:51.96600" "11:37:51.96600"
[7] "11:37:51.96600" "11:37:51.96600" "11:37:51.96600"
[10] "11:37:51.96600" "11:37:51.96600" "11:37:51.96600"
[13] "11:37:51.96600" "11:37:51.96600" "11:37:51.96600"
[16] "11:37:51.96600" "11:37:51.96600" "11:37:51.96600"
[19] "11:37:51.96600" "11:37:51.96600" "11:37:51.96600"
[22] "11:37:51.96600" "11:37:51.96600" "11:37:51.96600"
[25] "11:37:51.96600" "11:37:51.96600" "11:37:51.96600"
[28] "11:37:51.96600" "11:37:51.96600" "11:37:51.96600"
[31] "11:37:51.96600" "11:37:51.96600" "11:37:51.96600"
[34] "11:37:51.96600" "11:37:51.96600" "11:37:51.96600"
[37] "11:37:51.96600" "11:37:51.96600" "11:37:51.96600"
[40] "11:37:51.96600"
```

```
$time
[1] 41871.97 41871.97 41871.97 41871.97 41871.97 41871.97 41871.97
[8] 41871.97 41871.97 41871.97 41871.97 41871.97 41871.97 41871.97
[15] 41871.97 41871.97 41871.97 41871.97 41871.97 41871.97 41871.97
[22] 41871.97 41871.97 41871.97 41871.97 41871.97 41871.97 41871.97
[29] 41871.97 41871.97 41871.97 41871.97 41871.97 41871.97 41871.97
[36] 41871.97 41871.97 41871.97 41871.97 41871.97
```

2.2. The DICOM image

Most DICOM files involve a single slice from an acquisition – the image. A notable exception is the Siemens MOSAIC format (addressed in Section 2.2.1). The **oro.dicom** package assumes the image is stored as a flat file of two-byte integers without compression. A variety of additional image formats are possible within the DICOM standard; e.g., RGB-colored, JPEG, JPEG Lossless, JPEG 2000 and run-length encoding (RLE). None of these formats are currently available in **oro.dicom**. Going back to the `Abdo.dcm` example, the image is accessed via

```
R> image(t(abdo$img), col=grey(0:64/64), axes=FALSE, xlab="", ylab="")
```

Figure 2 displays a coronal slice through the abdomen from an MRI acquisition. All information from the original data acquisition should accompany the image through the DICOM header, and this information is utilized as much as possible by **oro.dicom** to simplify the manipulation of DICOM data. As previously shown, this information is easily available to the user by matching DICOM header fields with valid strings. Note, the function `extractHeader` assumes the output should be coerced via `as.numeric` but this may be disabled setting the input parameter `numeric=FALSE`.

```
R> extractHeader(abdo$hdr, "Manufacturer", numeric=FALSE)
```

```
[1] "Philips"
```

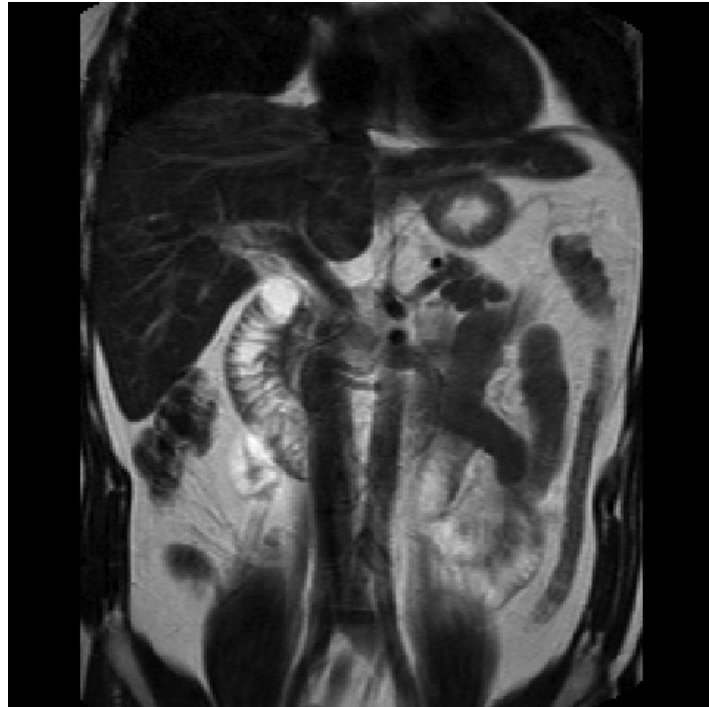


Figure 2: Coronal slice of the abdomen viewed in *neurological* convention (left is right and right is left).

```
R> extractHeader(abdo$hdr, "RepetitionTime")
```

```
[1] 2000
```

```
R> extractHeader(abdo$hdr, "EchoTime")
```

```
[1] 100
```

The basic DICOM file structure does not encourage the analysis of multi-dimensional imaging data (e.g., 3D or 4D) commonly acquired on clinical scanners. Hence, the **oro.dicom** package has been developed to access DICOM files and facilitate their conversion to the NIfTI or ANALYZE formats in R. The conversion process requires the **oro.nifti** package and will be outlined in Section 3.

Siemens MOSAIC format

Siemens multi-slice EPI (echo planar imaging) data may be collected as a “mosaic” image; i.e., all slices acquired in a single TR (repetition time) frame of a dynamic run are stored in a single DICOM file. The images are stored in an $M \times N$ array of images. The function `create3D` will try to guess the number of images embedded within the single DICOM file using the `AcquisitionMatrix` field. If this doesn’t work, one may enter the (M, N) doublet explicitly.

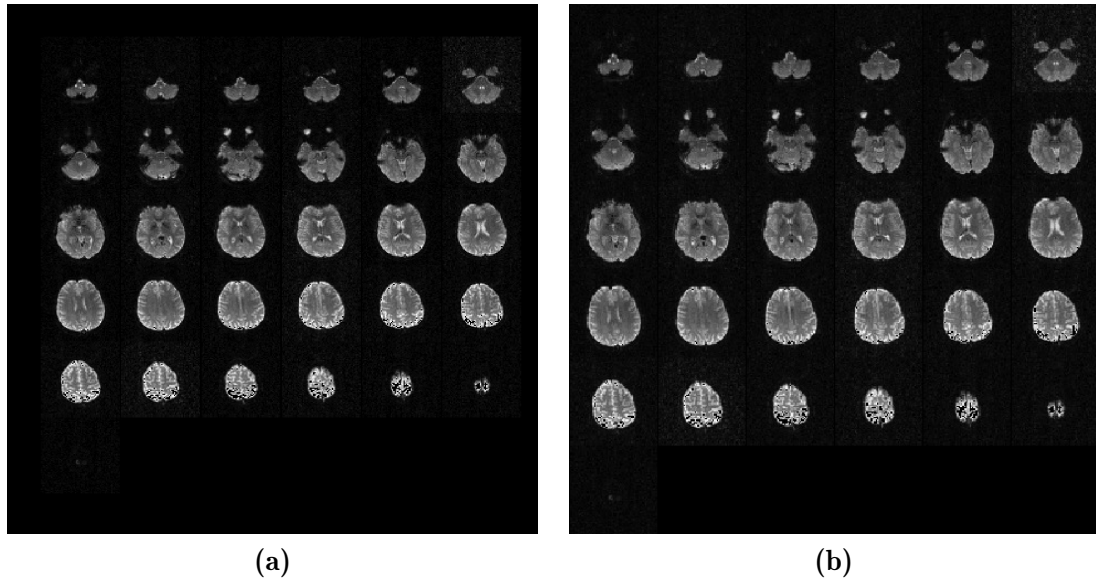


Figure 3: (a) Single MOSAIC image as read in from `readDICOMFile`. (b) Lightbox display of three-dimensional array of images after processing via `create3D`.

```
R> fname <- system.file(file.path("dcm", "MR-sonata-3D-as-Tile.dcm"),
+                        package="oro.dicom")
R> dcm <- readDICOMFile(fname)
R> dim(dcm$img)

[1] 384 384

R> dcmImage <- create3D(dcm, mosaic=TRUE)
R> dim(dcmImage)

[1] 64 64 36
```

Figure 3a is taken from the raw DICOM file, in mosaic format, and displayed with the default margins in R. Figure 3b is displayed after re-organizing the original DICOM file into a three-dimensional array (it was also converted to the NIfTI format for ease of visualization using the overloaded `image` function in `oro.nifti`).

3. Converting DICOM to NIfTI

The `oro.dicom` and `oro.nifti` packages have been specifically designed to use as much information as possible from the metadata-rich DICOM format and apply that information in the construction of the NIfTI data volume. The function `dicom2nifti` converts a list of DICOM images into an `nifti` object, and likewise `dicom2analyze` converts such a list into an `anlz` object.

Historically, data conversion from DICOM to NIfTI (or ANALYZE) has been provided outside of R using one of several standalone software packages:

- Xmedcon (Nolf 2003),
- FreeSurfer (<http://surfer.nmr.mgh.harvard.edu>),
- MRIConvert (<http://lnci.oregon.edu/~jolinda/MRIConvert>).

This is by no means an exhaustive list of software packages available for DICOM conversion. In addition there are several other R packages with the ability to process DICOM data

- **fmri** (Polzehl and Tabelow 2007),
- **tractor.base** (Clayden 2010) (part of the tractor project <http://code.google.com/p/tractor>).

3.1. An example using a single-series data set

Using the 40 images from the `hk40` object (previously defined in Section 2.1) it is straightforward to perform DICOM-to-NIfTI conversion using only default settings and plot the results in either lightbox or orthographic displays.

```
R> dput(formals(dicom2nifti))
```

```
as.pairlist(alist(dcm = , datatype = 4, units = c("mm", "sec"), rescale = FALSE, reslice =
```

```
R> (hk40n <- dicom2nifti(hk40))
```

NIfTI-1 format

```

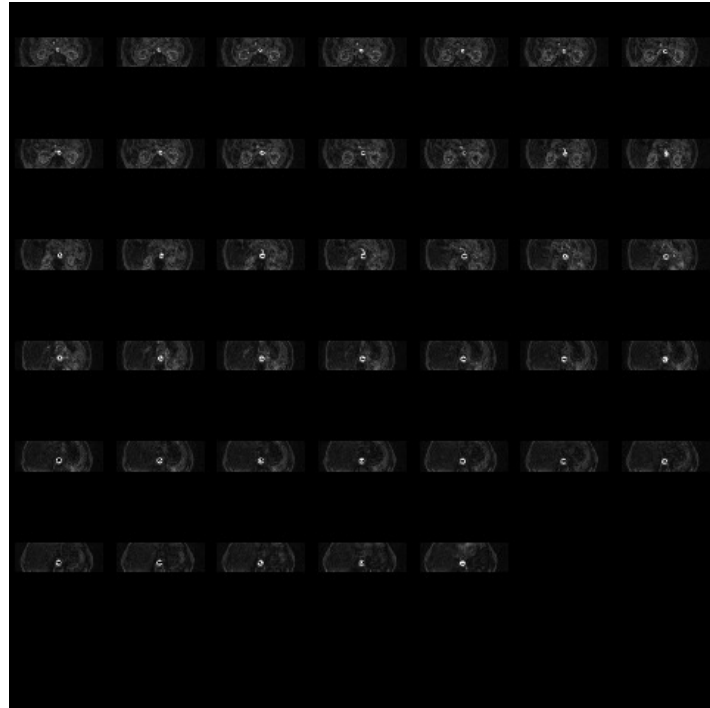
Type           : nifti
Data Type      : 4 (INT16)
Bits per Pixel : 16
Slice Code     : 0 (Unknown)
Intent Code    : 0 (None)
Qform Code     : 2 (Aligned_Anat)
Sform Code     : 2 (Aligned_Anat)
Dimension      : 256 x 256 x 40
Pixel Dimension : 1.56 x 1.56 x 3.12
Voxel Units    : mm
Time Units     : sec
```

```
R> image(hk40n)
```

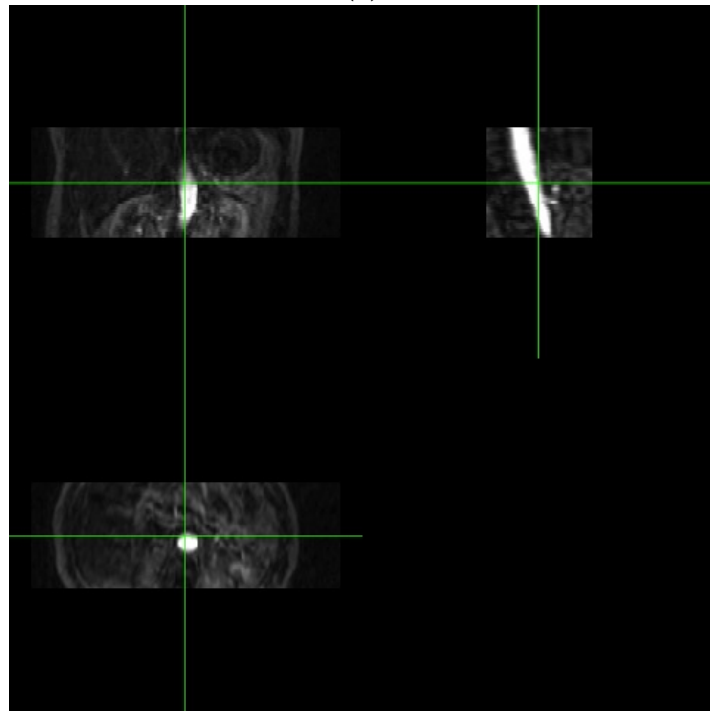
```
R> orthographic(hk40n, col.crosshairs="green")
```

By default `dicom2nifti` takes all image data from the DICOM list and creates a 3D image. Four-dimensional image volumes (three in space plus one in time) are also converted automatically by specifying `DIM=4`, where slice positions are taken from the `ImagePositionPatient` DICOM header field. For example, using `DIM=4` on the `hk40` DICOM data,

```
R> (hk40n <- dicom2nifti(hk40, DIM=4))
```



(a)



(b)

Figure 4: (a) Lightbox display of three-dimensional array of images. (b) Orthographic display of the same three-dimensional array (using the default settings for `orthographic`).

NIfTI-1 format

```

Type           : nifti
Data Type      : 4 (INT16)
Bits per Pixel : 16
Slice Code     : 0 (Unknown)
Intent Code    : 0 (None)
Qform Code     : 2 (Aligned_Anat)
Sform Code     : 2 (Aligned_Anat)
Dimension      : 256 x 256 x 40
Pixel Dimension : 1.56 x 1.56 x 3.12
Voxel Units    : mm
Time Units     : sec

```

will also produce a three-dimensional volume of images, since the `ImagePositionPatient` field is unique for each single slice of the volume.

The functions `dicom2nifti` and `dicom2analyze` will fail when the dimensions of the individual images in the DICOM list do not match. However, they do not check for different series numbers or patient IDs so caution should be exercised when scripting automated work flows for DICOM-to-NIfTI conversion. In cases where a DICOM file includes images from more than one series, the corresponding slices have to be chosen before conversion, using `dicomTable`, `extractHeader`, and `matchHeader`.

3.2. An example using a multiple-volume data set

The National Biomedical Imaging Archive (NBIA; <http://cabig.nci.nih.gov/tools/NCIA>) is a searchable, national repository integrating *in vivo* cancer images with clinical and genomic data. The NBIA provides the scientific community with public access to DICOM images, image markup, annotations, and rich metadata. The multiple MRI sequences processed here were downloaded from the “RIDER Neuro MRI” collection at <http://wiki.nci.nih.gov/display/CIP/RIDER>. A small `for` loop has been written to operate on a subset of the DICOM directory structure, where the `SeriesInstanceUID` DICOM header field is assumed to be 100% accurate in series differentiation.

```

R> subject <- "1086100996"
R> DCM <- readDICOM(subject, verbose=TRUE)
R> seriesInstanceUID <- extractHeader(DCM$hdr, "SeriesInstanceUID", FALSE)
R> for (uid in unique(seriesInstanceUID)) {
+   index <- which(unlist(lapply(DCM$hdr, function(x) uid %in% x$value)))
+   uid.dcm <- list(hdr=DCM$hdr[index], img=DCM$img[index])
+   patientsName <- extractHeader(uid.dcm$hdr, "PatientsName", FALSE)
+   studyDate <- extractHeader(uid.dcm$hdr, "StudyDate", FALSE)
+   seriesDescription <- extractHeader(uid.dcm$hdr, "SeriesDescription", FALSE)
+   fname <- paste(gsub("[^0-9A-Za-z]", "",
+                       unique(c(patientsName, studyDate, seriesDescription))),
+                 collapse="_")
+   cat("## ", fname, fill=TRUE)
+   if (gsub("[^0-9A-Za-z]", "", unique(seriesDescription)) == "axtensor") {

```

```

+     D <- 4
+     reslice <- FALSE
+   } else {
+     D <- 3
+     reslice <- TRUE
+   }
+   uid.nifti <- dicom2nifti(uid.dcm, DIM=D, reslice=reslice,
+                           descrip=c("PatientID", "SeriesDescription"))
+   writeNifti(uid.nifti, fname)
+ }

```

Note, the diffusion tensor imaging (DTI) data **axtensor** is assumed to be four dimensional and all other series (the multiple flip-angle acquisitions) are assumed to be three dimensional. There is always a balance between what information should be pre-specified versus what can easily be extracted from the DICOM headers or images.

4. Conclusion

Medical image analysis depends on the efficient manipulation and conversion of DICOM data. The **oro.dicom** package has been developed to provide the user with a set of functions that mask as many of the background details as possible while still providing flexible and robust performance.

The future of medical image analysis in R will benefit from a unified view of the imaging data standards: DICOM, Nifti, ANALYZE, AFNI, MINC, etc. The existence of a single package for handling imaging data formats would facilitate interoperability between the ever increasing number of R packages devoted to medical image analysis. We do not assume that the data structures in **oro.dicom** or **oro.nifti** are best-suited for this purpose and we welcome an open discussion around how best to provide this standardization to the end user.

Acknowledgments

The authors would like to thank the National Biomedical Imaging Archive (NBIA), the National Cancer Institute (NCI), the National Institute of Health (NIH) and all institutions that have contributed medical imaging data to the public domain. VS is supported by the German Research Council (DFG SCHM 2747/1-1).

References

- Clayden J (2010). *tractor.base: A Package for Reading, Manipulating and Visualising Magnetic Resonance Images*. R package version 1.5.0, URL <http://CRAN.R-project.org/package=tractor.base>.
- Nolf E (2003). "XMedCon - An Open-source Medical Image Conversion Toolkit." *European Journal of Nuclear Medicine*, **30**(Suppl. 2), S246. URL <http://xmedcon.sourceforge.net>.

Polzehl J, Tabelow K (2007). “**fmri**: A Package for Analyzing fMRI Data.” *RNews*, **7**(2), 13–17. URL http://www.r-project.org/doc/Rnews/Rnews_2007-2.pdf.

R Development Core Team (2010). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria. ISBN 3-900051-07-0, URL <http://www.R-project.org>.

Whitcher B, Schmid VJ, Thornton A (2011). “Working with the DICOM and NIfTI Data Standards in R.” *Journal of Statistical Software*, **44**(6), 1–28. URL <http://www.jstatsoft.org/v44/i06/>.

Affiliation:

Brandon Whitcher
Pfizer Worldwide Research & Development
610 Main Street
Cambridge, MA 02139, United States
E-mail: bwhitcher@gmail.com
URL: <http://www.imperial.ac.uk/people/b.whitcher>, <http://rigorousanalytics.blogspot.com>

Volker J. Schmid
Bioimaging group
Department of Statistics
Ludwig-Maximilians-Universität München
80539 München, Germany
E-mail: volker.schmid@lmu.de
URL: <http://volkerschmid.de>

Andrew Thornton
Cardiff University School of Medicine
Heath Park
Cardiff CF14 4XN, United Kingdom
E-mail: art27@cantab.net