

Starting with Raw Data

John Muschelli

Image formats: DICOM

DICOM (Digital Imaging and Communications in Medicine) format

- ▶ Standardized way of representing images
- ▶ Usually how data is given either from scanner or hospital PACS (picture archiving and communication system) system
- ▶ 2 integral pieces: Image data (in pixels) and header (meta-data (data about data))
 - ▶ Think of a JPEG/bitmap and a text file

DICOM pixel data

The pixel data in a DICOM file is a matrix (fixed number of rows and columns).

One DICOM file represents one “slice” of the brain.

The `oro.dicom` package is good for reading in this data.

```
library(oro.dicom)
library(divest)
dcm_file = system.file("extdata", "testdata",
                        "01.dcm", package = "divest")
slice = readDICOM(dcm_file)
class(slice)
```

```
[1] "list"
```

readDICOM output

The output is a list with 2 elements: the DICOM header (`hdr`) and image (`img`) information, both of which are lists.

Each element of `hdr` has a `data.frame`, and the elements of `img` are matrices:

```
names(slice)
```

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

```
class(slice$hdr)
```

```
[1] "list"
```

```
class(slice$hdr[[1]])
```

```
[1] "data.frame"
```

```
class(slice$img)
```

Display DICOM Image using image command

(We transpose the data using `t()` so the image faces “up” instead of “right”).

```
image(t(slice$img[[1]]), col=gray(0:64/64))
```



DICOM Header Information

What about the header?

There are many fields, for example pixelSpacing, which is the dimensions (x and y) of a pixel in millimeters (mm):

```
hdr = slice$hdr[[1]]  
hdr[hdr$name == 'PixelSpacing', "value"]
```

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

DICOM Header Information

Many fields - most important are tag, name and value:

```
head(hdr)
```

	group	element	name	code	length	value
1	0002	0000	GroupLength	UL	4	188
2	0002	0001	FileMetaInformationVersion	OB	2	\001
3	0002	0002	MediaStorageSOPClassUID	UI	26	1.2.840.10008.5.1.4.1.1.4
4	0002	0003	MediaStorageSOPInstanceUID	UI	56	1.3.12.2.1107.5.2.30.25471.30000008091010241767100006233
5	0002	0010	TransferSyntaxUID	UI	20	1.2.840.10008.1.2.1
6	0002	0012	ImplementationClassUID	UI	18	1.3.12.2.1107.5.2

DICOM Header Information

You can reshape this into a “wide” format using `dicomTable`

```
wide = oro.dicom::dicomTable(slice$hdr)
wide[, 1:5]
```

```
0002-0000-GroupLength 0002-0001-FileMetaInformationVersion
1 188 \000
0002-0002-MediaStorageSOPClassUID
1 1.2.840.10008.5.1.4.1.1.4
0002-0003-MediaStorageSOPInstanceUID
1 1.3.12.2.1107.5.2.30.25471.30000008091010241767100006233
0002-0010-TransferSyntaxUID
1 1.2.840.10008.1.2.1
```


Multiple DICOM files

We have discussed only one slice of the brain. What about multiple slices? If you pass a directory into `readDICOM`, it will read in all DICOM files in that directory.

```
dcm_fol = system.file("extdata", "testdata", package = "diveR")
all_slices = readDICOM(dcm_fol)
n = names(all_slices$hdr)
keep = grep("0(1|2).dcm", n)
all_slices$hdr = all_slices$hdr[keep]
all_slices$img = all_slices$img[keep]
```

Multiple DICOM files: header reshaping

When you pass in multiple headers, you have a row for each file:

```
wide = oro.dicom::dicomTable(all_slices$hdr)
wide[, 1:5]
```

	0002-0000-GroupLength	0002-0001-FileMetaInformationV
01.dcm	188	
02.dcm	188	
	0002-0002-MediaStorageSOPClassUID	
01.dcm	1.2.840.10008.5.1.4.1.1.4	
02.dcm	1.2.840.10008.5.1.4.1.1.4	
		0002-0003-MediaStorageSOPInstanceUID
01.dcm	1.3.12.2.1107.5.2.30.25471.3000000809101024176710000	
02.dcm	1.3.12.2.1107.5.2.30.25471.3000000809101024176710000	
	0002-0010-TransferSyntaxUID	
01.dcm	1.2.840.10008.1.2.1	
02.dcm	1.2.840.10008.1.2.1	

NIfTI

Now that we have multiple slices read in, we can convert it to a 3-dimensional (3D) array, where you can think of the array as stacking each slice (which is a matrix) on top of each other. If each DICOM is a piece of paper, the 3D array is a stack of paper.

The way we store this 3D array is in the NIfTI (Neuroimaging Informatics Technology Initiative) format.

DICOM vs. NIfTI

	DICOM	NIfTI
File extension:	.dcm	.nii or .nii.gz (compressed)
Each file is a:	slice of the brain	3D image of brain
Header information:	Many fields, protected health information, hospital-related meta-data	Image meta-data, no patient information
Different Images	Different Folders	Different Files (can be same directory)

NIfTI

We can convert this list of header information and image information to a `nifti` object (an R object) with the `dicom2nifti` command:

```
nii = dicom2nifti(all_slices)
dim(nii); class(nii)
```

```
[1] 2 224 256
```

```
[1] "nifti"
```

```
attr(,"package")
```

```
[1] "oro.nifti"
```

We can see that this `nii` object is indeed a `nifti` object and has 3 dimensions.

Writing out NIfTI file

The `writenii` command from the `neurobase` package can write out this `nifti` object to a NIfTI file (based on `writeNIfTI` from `oro.nifti`):

```
library(neurobase)
writenii(nim = nii, filename = "Output_3D_File")
list.files(pattern = "Output_3D_File")
```

```
[1] "Output_3D_File.nii.gz"
```

Note that the extension is `.nii.gz`, which is a compressed NIfTI file, which saves disk space for storage.

You can output a non-compressed file using the argument `gzipped=FALSE`.

(NB: The filename argument in `writeNIfTI` should NOT have an extension, but can in `writenii`)

divest, dcm2niir and dcm2niix

dcm2niix is a piece of software from Dr. Chris Rorden that can convert nearly any DICOM format to a NIfTI. It is our recommended way of converting DICOM to NIfTI files.

- ▶ `divest` - wraps dcm2niix code in Rcpp (newer, still being tested)
- ▶ `dcm2niir` - wraps the binary dcm2niix program. This is not as cross-platform (Linux and Mac OSX only), but may handle some cases a bit better than `divest` (until that package matures).

In each, you specify a path argument:

```
library(dcm2niir)
d = dcm2nii(path)
library(divest)
res = readDicom(path, interactive = FALSE)
```

Other DICOM converters

Although our main goal is to analyze neuro data with the fewest pieces of software, we are pragmatic and use existing software if it works well.

The dcm2nii software converts DICOM files to NIfTI files, and can handle many different formats and file types.

In general, the format we will be using will be NIfTI, and we will store out data in compressed format, so extensions of our images will be `.nii.gz`.

Other formats

- ▶ For Philips scanners, files from the scanner are PAR/REC and not DICOM, but can still be converted using `dcm2nii`. `r2a` can also convert these to NIfTI.
- ▶ NIfTI format was based on ANALYZE format where the header and image were in separate `.hdr` and `.img` files. This is an older format and we will not use this way of storing data because 1) NIfTI can have one file with both header and image information, and 2) Can be stored as compressed `.nii.gz` files.
- ▶ NRRD (Nearly Raw Raster Data) is another format similar to NIfTI. Much of the neuroimaging software can read in both NRRD and NIfTI files, but NIfTI is much more common.

Conclusions

In this course, we will be using NIfTI images, and will be using `readnii` to read them in, and `writenii` to write them out.

- ▶ Images will be in gzipped format (extension `.nii.gz`)
- ▶ Additional software (e.g. FSL) will be useful to reorient the data to a specified orientation before reading
- ▶ DICOM images can be converted robustly using `divest` or `dcm2nii`