RA: A simple, fast, and portable multidimensional array storage format

David S. Smith
Institute of Imaging Science, Vanderbilt University, Nashville, TN, USA

Purpose. The offline use of raw data is proliferating in MRI, driven by advanced acquisitions and reconstructions. Unfortunately the variety of formats has made it harder to both provide data for reproducible research and to test new reconstruction packages on existing data. This Tower of Babel is exacerbated by the numerous proprietary formats used by vendors.

For our reconstruction development, we wanted raw, anonymized data with minimal metadata—essentially a multidimensional array on disk—stored in a single file that can be read and written at full disk speed with one line of code. Surprisingly, none of the existing data formats fit our simple requirements.

Hierarchical Data Format v5 (HDF5) is burdensome for the casual user to install and to manipulate, raising the barrier to reproducible research, and doesn't support native complex floats. The ISMRM Raw Data[2] (ISMRMRD) format is based on HDF5, so it suffers from the same problems, with the added problem of encoding the data as a binary blob that can only be decoded by parsing a complex derived header type, making memory mapping impossible.

The popular MAT format from MATLAB is widespread and relatively easy to manipulate, but the older version 5 doesn't support arrays over 2 GB, and later versions are based on HDF5.

The CFL format from the Berkeley Advanced Reconstruction Toolbox[3] came the closest to meeting our requirements, but it uses a separate text header file to store the array dimensions (not ideal) and supports only single-precision complex floats.

Here we present our solution: the **RA (raw array)** format, a simple, fast, and portable format for storing multidimensional arrays on disk.

Methods. The proposed RA file format is a simple concatenation of a header array and a data array. The header comprises at least seven 64-bit unsigned integers. The array data can be anything. User metadata can be appended to an RA file with no harmful effects, for example to store acquisition details. Table 1 shows the file internals.

Table 1–RA File Structure

Offset (bytes)	Type	Field	Description
			Header
0	`uint64`	`magic`	magic number
8	`uint64`	`flags`	endianness, etc.
16	`uint64`	`etype`	element type code
24	`uint64`	`ebyte`	element size in bytes
32	`uint64`	`size`	data length in bytes
40	`uint64`	`ndims`	number of dimensions
48	`uint64[]`	`dims`	array dimensions
$48 + 8\times$ndims	`uint8[]`	`data`	**Array Data**
$48 + 8\times$ndims $+$ size	-	-	**User Metadata**

Most of the fields in Table 1 are self explanatory. The magic number was chosen as the ASCII representation of "rawarray" (`0x7961727261776172`) and uniquely identifies an RA file.

The element type codes supported so far are 0: user-defined struct, 1: signed integer, 2: unsigned integer, 3: floating point (IEEE-754), and 4: complex float (float tuples). Codes 5 and up are available for future use.

The array data is stored in the native binary format, with flags to signal endianness and whether it is row or column major. Last is optional user-inserted metadata. The user metadata is ignored on read, which increases speed and decreases code complexity.

A checksum was deliberately omitted because it is almost impossible to checksum a file containing its checksum. Some methods (e.g. tar) zero out the checksum field and then checksum the rest of the file, but this requires special software that understands the format, so standard command-line checksum tools won't work.

Time stamping was omitted because file systems already provide that. Adding a time stamp that changes upon rewrite or access also foils checksums. HDF5 files are problematic to checksum for this reason. Two RA files are defined as identical if and only if they contain identical contents, no matter when they were created or accessed last.

Results. Figure 1 shows example timings for writing a large array, typical of MRI data sizes, in MATLAB R2015b. The MAT and HDF5 files are written using the provided MATLAB functions, which are calling low-level, compiled code, while the RA write function was written in about 35 lines of pure MATLAB code and called with one line ("`rawrite(x,'x.ra')`"). Nevertheless, RA edged out HDF5.

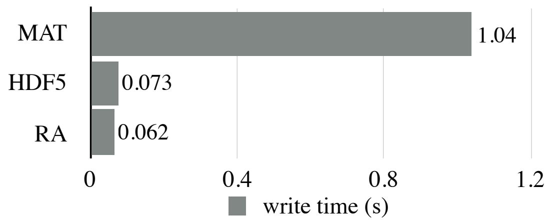

Figure 1—Time to write a 256 x 256 x 64 array of double-precision floats in three formats from MATLAB R2015b. The proposed format was slightly faster than HDF5.

Discussion. RA already accomplishes our original goals, but if necessary compression or encryption could be added in the future. While RA is optimized for storing a single array, it could serve as a file container by using composite types. We think containing files is better left to the file system, however, which is already designed to hierarchically organize heterogeneous objects.

In contrast to the all-encompassing idea of HDF5, our vision for archiving raw MRI data is a directory structure on an open-sourced file system (e.g. ext4), in which subdirectories are used for grouping, metadata is stored as human-readable markup (e.g. YAML), and data is stored in individual RA files. Packaging of the directory structure could be accomplished by existing battle-proven formats, such as tar, zip, or iso. In other words, keep it simple and use what works.

Conclusions. RA was as fast as HDF5 and far simpler. The reference RA implementation can be downloaded from `https://github.com/davidssmith/ra`.

References. [1] https://www.hdfgroup.org/HDF5/, [2] http://ismrmrd.github.io/, [3] Uecker et al. 2015, Proc ISMRM 23:2486

Acknowledgements. NCI K25 CA176219