Working with the NIfTI 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.nifti** facilitates the interaction with and manipulation of medical imaging data that conform to the ANALYZE, NIfTI and AFNI formats. The S4 class framework is used to develop basic ANALYZE and NIfTI classes, where NIfTI extensions may be used to extend the fixed-byte NIfTI header. One example of this, that has been implemented, is an XML-based "audit trail" tracking the history of operations applied to a data set. The conversion from DICOM to ANALYZE/NIfTI is straightforward using the capabilities of **oro.dicom**. The S4 classes have been developed to provide a user-friendly interface to the ANALYZE/NIfTI data formats; allowing easy data input, data output, image processing and visualization.

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 ANALYZE format was developed at the Mayo Clinic (in the 1990s) to store multidimensional biomedical images. It is fundamentally different from the DICOM standard since it groups all images from a single acquisition (typically three- or four-dimensional) into a pair of binary files, one containing header information and one containing the image information. The DICOM standard groups the header and image information, typically a single two-dimensional image, into a single file. Hence, a single acquisition will contain multiple DICOM files but only a pair of ANALYZE files.

The NIfTI format was developed in the early 2000s by the DFWG (Data Format Working Group) in an effort to improve upon the ANALYZE format. The resulting NIfTI-1 format adheres to the basic header/image combination from the ANALYZE format, but allows the pair of files to be combined into a single file and re-defines the header fields. In addition, NIfTI extensions allow one to store additional information (e.g., key acquisition parameters, experimental design) inside a NIfTI file.

The material presented here provides users with a method of interacting with ANALYZE and NIfTI 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.nifti** (Whitcher *et al.* 2011). It

oro.nifti	
`afni`, `anlz`, `nifti`	Class constructors for AFNI, ANALYZE and NIfTI objects.
`as(<obj>, "nifti")`	Coerce object into class `nifti`.
`audit.trail`, `aux.file`, `descrip`	Extract or replace slots in specific header fields.
`fmri2oro`, `oro2fmri`	Convert between `fmridata` (**fmri**) and `nifti` objects.
`hotmetal`, `tim.colors`	Useful color tables for visualization.
`image`, `orthographic`, `overlay`	Two-dimensional visualization methods.
`is.afni`, `is.anlz`, `is.nifti`	Logical checks.
`readAFNI`, `readANALYZE`, `readNIfTI`	Data input.
`writeAFNI`, `writeANALYZE`, `writeNIfTI`	Data otuput.

Table 1: List of functions available in **oro.nifti**. Functionality around the AFNI data format was recently added to the **oro.nifti** package. Please visit http://afni.nimh.nih.gov/afni for more information about the AFNI data format.

should be noted that **oro.nifti** focuses on functions for data input/output and visualization. S4 classes `nifti` and `anlz` are provided for further statistical analysis in R without losing contextual information from the original ANALYZE or NIfTI files. Images in the metadata-rich DICOM format may be converted to NIfTI semi-automatically using **oro.dicom** 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` and `anlz` objects. Additionally, the **oro.nifti** package allows one to track every operation on a `nifti` object in an XML-based audit trail.

The **oro.nifti** package should appeal not only to R package developers, but also to scientists 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.nifti** and groups them according to common functionality. An example of using statistical methodology in R for the analysis of functional magnetic resonance imaging (fMRI) data is given in section 2.7. Packages already available on CRAN that utilize **oro.nifti** include: **cudaBayesreg** (Ferreira da Silva 2011), **dcemriS4** (Whitcher and Schmid 2011), **dpmixsim** (Ferreira da Silva 2010), and **RNiftyReg** (Clayden 2011).

2. oro.nifti: NIfTI-1 data input/output in R

Although the industry standard for medical imaging data is DICOM, another format has come to be heavily used in the image analysis community. The ANALYZE format was originally developed in conjunction with an image processing system (of the same name) at the Mayo Foundation. A common version of the format, although not the most recent, is called ANALYZE 7.5. A copy of the file ANALYZE75.pdf has been included in **oro.nifti** (accessed via `system.file("doc/ANALYZE75.pdf", package="oro.dicom")`) since it does not appear to be available from www.mayo.edu any longer. An ANALYZE 7.5 format image is comprised

of two files, the ".hdr" and ".img" files, that contain information about the acquisition and the acquisition itself, respectively. A more recent adaption of this format is known as NIfTI-1 and is a product of the Data Format Working Group (DFWG) from the Neuroimaging Informatics Technology Initiative (NIfTI; http://nifti.nimh.nih.gov). The NIfTI-1 data format is almost identical to the ANALYZE format, but offers a few improvements

- merging of the header and image information into one file (.nii)
- re-organization of the 348-byte fixed header into more relevant categories
- possibility of extending the header information.

There are several R packages that also offer input/output functionality for the NIfTI and ANALYZE data formats in addition to image analysis capabilities for specific MRI acquisition sequences; e.g., **AnalyzeFMRI** (Bordier *et al.* 2009), **fmri** (Polzehl and Tabelow 2007) and **tractor.base** (Clayden 2010). The **Rniftilib** package provides access to NIfTI data via the nifticlib library (Granert 2010).

2.1. The NIfTI header

The NIfTI header inherits its structure (348 bytes in length) from the ANALYZE data format. The last four bytes in the NIfTI header correspond to the "magic" field and denote whether or not the header and image are contained in a single file (`magic = "n+1\0"`) or two separate files (`magic = "ni1\0"`), the latter being identical to the structure of the ANALYZE data format. The NIfTI data format added an additional four bytes to allow for "extensions" to the header. By default these four bytes are set to zero.

The first example of reading in, and displaying, medical imaging data in NIfTI format `avg152T1_LR_nifti.nii.gz` was obtained from the NIfTI website (http://nifti.nimh.nih.gov/nifti-1/). Successful execution of the commands

```
R> fname <- system.file(file.path("nifti", "mniLR.nii.gz"), package="oro.nifti")
R> (mniLR <- readNIfTI(fname))

NIfTI-1 format
  Type            : niftiAuditTrail
  Data Type       : 2 (UINT8)
  Bits per Pixel  : 8
  Slice Code      : 0 (Unknown)
  Intent Code     : 0 (None)
  Qform Code      : 0 (Unknown)
  Sform Code      : 4 (MNI_152)
  Dimension       : 91 x 109 x 91
  Pixel Dimension : 2 x 2 x 2
  Voxel Units     : mm
  Time Units      : sec

R> pixdim(mniLR)
```

```
[1] 0 2 2 2 1 1 1 1
```

```
R> descrip(mniLR)
```

```
[1] "FSL3.2beta"
```

```
R> aux.file(mniLR)
```

```
[1] "none                    "
```

produces an S4 `"nifti"` object (or `"niftiAuditTrail"` if the audit trail option is set). Some accessor functions are also provided; e.g., `aux.file` and `descrip`. The former is used to access the original name of the file (if it has been provided) and the latter is the name of a valid NIfTI header field used to hold a "description" (up to 80 characters in length).

2.2. The NIfTI image

Image information begins at the byte position determined by the `voxoffset` slot. In a single NIfTI file (`magic = "n+1\0"`), this is by default after the first 352 bytes. Header extensions extend the size of the header and come before the image information leading to a consequent increase of `voxoffset` for single NIfTI files. The split NIfTI (`magic = "ni1\0"`) and ANALYZE formats contain pairs of files, where the header and image information are separated, and do not have this problem. In this case `voxoffset` is set to 0.

The `image` function has been overloaded so that it behaves differently when dealing with medical image objects (`nifti` and `anlz`). The command

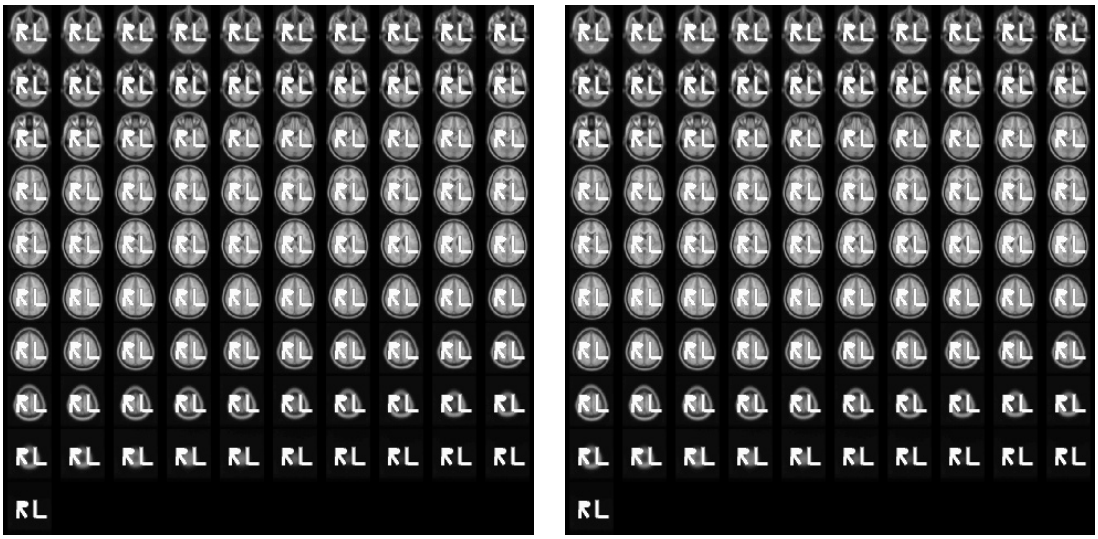
```
R> image(mniLR)
```

produces a three-dimensional array of the MNI brain, with the default NIfTI axes, and is displayed on a 10×10 grid of images (Figure 1a). The `image` function for medical image S4 objects is an attempt to balance minimal user input with enough flexibility to customize the display when necessary. For example, single slices may be viewed by using the option `plot.type="single"` in conjunction with the option `z=` to specify the slice.

The second example of reading in and displaying medical imaging data in the NIfTI format `avg152T1_RL_nifti.nii.gz` was also obtained from the NIfTI website (http://nifti.nimh.nih.gov/nifti-1/). Successful execution of the commands

```
R> fname <- system.file(file.path("nifti", "mniRL.nii.gz"), package="oro.nifti")
R> (mniRL <- readNIfTI(fname))
```

```
NIfTI-1 format
  Type            : niftiAuditTrail
  Data Type       : 2 (UINT8)
  Bits per Pixel  : 8
  Slice Code      : 0 (Unknown)
  Intent Code     : 0 (None)
```



(a) **(b)**

Figure 1: **(a)** Axial slices of MNI volume `mniLR_nifti` stored in the *neurological* convention (right-is-right), but displayed in the *radiological* convention (right-is-left). **(b)** Axial slices of MNI volume `mniRL_nifti` stored and displayed in the *radiological* convention.

```
Qform Code       : 0 (Unknown)
Sform Code       : 4 (MNI_152)
Dimension        : 91 x 109 x 91
Pixel Dimension : 2 x 2 x 2
Voxel Units      : mm
Time Units       : sec
```

```
R> image(mniRL)
```

produces a three-dimensional array of the MNI brain that is displayed in a 10×10 grid of images (Figure 1b). The two sets of data in Figure 1 are stored in two different orientations, commonly referred to as the *radiological* and *neurological* conventions. The neurological convention is where "right is right" and one is essentially looking through the subject. The radiological convention is where "right is left" and one is looking at the subject.

An additional graphical display function has been added for `nifti` and `anlz` objects that allows a so-called orthographic visualization of the data.

```
R> orthographic(mniRL)
```

As seen in Figure 2 the mid-axial, mid-sagittal and mid-coronal planes are displayed by default. The slices used may be set using `xyz = c(I,J,K)`, where (I, J, K) are appropriate indices, and the crosshairs will provide a spatial reference in each plane relative to the other two.

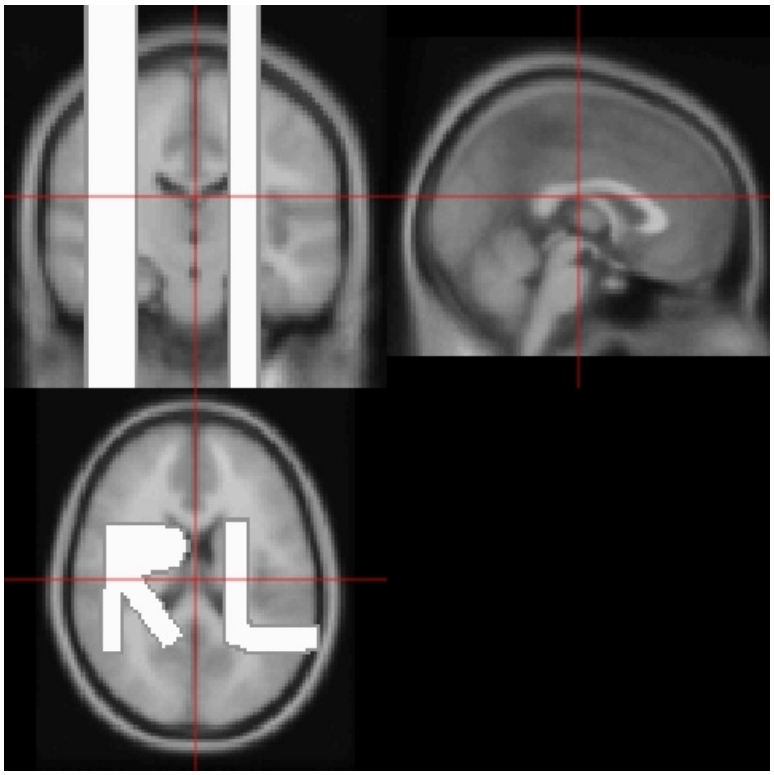


Figure 2: Orthographic display of the MNI volume `mniRL_nifti`. By default the mid-axial, mid-sagittal and mid-coronal planes are chosen.

2.3. A note on axes and orientation

The NIfTI format contains an implicit generalized spatial transformation from the data co-ordinate system (i, j, k) into a real-space "right-handed" co-ordinate system. In this real-space system, the (x, y, z) axes are *usually* set such that x increases from left to right, y increases from posterior to anterior and z increases from inferior to superior.

At this point in time the **oro.nifti** package cannot apply an arbitrary transform to the imaging data into (x, y, z) space – such a transform may require non-integral indices and interpolation steps. The package does accommodate straightforward transformations of imaging data; e.g., setting the i-axis to increase from right to left (the neurological convention). Future versions of **oro.nifti** will attempt to address more complicated spatial transformations and provide functionality to display the (x, y, z) axes on orthographic plots.

2.4. NIfTI and ANALYZE data in S4

A major improvement in the **oro.nifti** package is the fact that standard medical imaging formats are stored in unique classes under the S4 system (Chambers 2008). Essentially, NIfTI and ANALYZE data are stored as multi-dimensional arrays with extra slots created that capture the format-specific header information; e.g., for a `nifti` object

```
R> slotNames(mniRL)
```

```
 [1] ".Data"          "trail"          "extensions"
 [4] "sizeof_hdr"     "data_type"      "db_name"
 [7] "extents"        "session_error"  "regular"
[10] "dim_info"       "dim_"           "intent_p1"
[13] "intent_p2"      "intent_p3"      "intent_code"
[16] "datatype"       "bitpix"         "slice_start"
[19] "pixdim"         "vox_offset"     "scl_slope"
[22] "scl_inter"      "slice_end"      "slice_code"
[25] "xyzt_units"     "cal_max"        "cal_min"
[28] "slice_duration" "toffset"        "glmax"
[31] "glmin"          "descrip"        "aux_file"
[34] "qform_code"     "sform_code"     "quatern_b"
[37] "quatern_c"      "quatern_d"      "qoffset_x"
[40] "qoffset_y"      "qoffset_z"      "srow_x"
[43] "srow_y"         "srow_z"         "intent_name"
[46] "magic"          "extender"       "reoriented"
```

```
R> c(cal.min(mniRL), cal.max(mniRL))
```

```
[1]   0 255
```

```
R> range(mniRL)
```

```
[1]   0 255
```

```
R> mniRL@"datatype"

[1] 2

R> convert.datatype(mniRL@"datatype")

[1] "UINT8"
```

Note, an ANALYZE object has a slightly different set of slots. Slots 4–47 are taken verbatim from the definition of the NIfTI format and are read directly from a file. The slot `.Data` is the multidimensional array (since class `nifti` inherits from class `array`) and the slots `trail`, `extensions` and `reoriented` are used for internal bookkeeping. In the code above we have accessed the min/max values of the imaging data using the `cal.min` and `cal.max` accessor functions which matches a direct interrogation of the `.Data` slot using the `range` function. Looking at the `datatype` slot provides a numeric code that may be converted into a value that indicates the type of byte structure used (in this case an 8-bit or 1-byte unsigned integer).

As introduced in Section 2.1 there are currently only two accessor functions to slots in the NIfTI header (`aux.file` and `descrip`) – all other slots are either ignored or used inside of functions that operate on ANALYZE/NIfTI objects. The NIfTI class also has the ability to read and write extensions that conform to the NIfTI data format. Customized printing and validity-checking functions are available to the user and every attempt has been made to ensure that the information from the multi-dimensional array is in agreement with the header values.

The constructor function `nifti` produces valid NIfTI objects, including a consistent header, from an arbitrary array.

```
R> n <- 100
R> (random.image <- nifti(array(runif(n*n), c(n,n,1))))

NIfTI-1 format
  Type            : niftiAuditTrail
  Data Type       : 2 (UINT8)
  Bits per Pixel  : 8
  Slice Code      : 0 (Unknown)
  Intent Code     : 0 (None)
  Qform Code      : 0 (Unknown)
  Sform Code      : 0 (Unknown)
  Dimension       : 100 x 100 x 1
  Pixel Dimension : 1 x 1 x 1
  Voxel Units     : Unknown
  Time Units      : Unknown

R> random.image@"dim_"

[1]   3 100 100   1   1   1   1   1
```

```
R> dim(random.image)

[1] 100 100   1
```

The function `writeNIfTI` outputs valid NIfTI class files, which can be opened in other medical imaging software. Files can either be stored as standard `.nii` files or compressed with gnuzip (default).

```
R> writeNIfTI(random.image, "random")
R> list.files(pattern="random")

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

2.5. The audit trail

Following on from the S4 implementation of both the NIfTI and ANALYZE data formats, the ability to extend the NIfTI data format header is utilized in the **oro.nifti** package. Please use the command

```
R> options(niftiAuditTrail=TRUE)
```

to turn on the "audit trail" option in **oro.nifti** and then execute the function `enableAuditTrail()`. With the option enabled extensions are properly handled when reading and writing NIfTI data, users are allowed to add extensions to newly-created NIfTI objects by casting them as `niftiExtension` objects and adding `niftiExtensionSection` objects to the `extensions` slot, and all operations that are performed on a NIfTI object will generate what we call an *audit trail* that consists of an XML-based log (Temple Lang 2010).

Figure 3 displays output from the accessor function `audit.trail(mniLR)`, the XML-based audit trail that is stored as a NIfTI header extension.

Each log entry contains information not only about the function applied to the NIfTI object, but also various system-level information; e.g., version of R, user name, date, time, etc. When writing NIfTI-class objects to disk, the XML-based NIfTI extension is converted into plain text and saved using `ecode=6` to denote plain ASCII text. The user may control the tracking of data manipulation via the audit trail using the global option `niftiAuditTrail`.

2.6. Interactive visualization

Basic visualization of `nifti` and `anlz` class images can be achieved with any visualization for arrays in R. For example, the **EBImage** package provides functions `display` and `animate` for visualization (Sklyar *et al.* 2010). Please note that functions in **EBImage** expect grey-scale values in the range $[0, 1]$, hence the display of `nifti` data may be performed using

```
R> mniLR.range <- range(mniLR)
R> display((mniLR - min(mniLR)) / diff(mniLR.range))
```

```
R> audit.trail(mniLR)

<audit-trail xmlns="http://www.dcemri.org/namespaces/audit-trail/1.0">
  <created>
    <workingDirectory>/tmp/RtmpOk3Jv2/Rbuild1d4c4a385d96/oro.nifti/vignettes</workingDirectory>
    <filename>/tmp/RtmpOk3Jv2/Rinst1d4c570fc048/oro.nifti/nifti/mniLR.nii.gz</filename>
    <call>readNIfTI(fname = fname)</call>
    <system>
      <r-version>R version 3.1.2 Patched (2015-01-12 r67453)</r-version>
      <date>Sun Jan 18 03:06:18 PM 2015 CET</date>
      <user>rforge</user>
      <package-version>0.4.3</package-version>
    </system>
  </created>
</audit-trail>
```

Figure 3: XML-based audit trail obtained via `audit.trail(mniLR)`. Note, this function will return `NULL` if the **XML** package is not available.

Interactive visualization of multi-dimensional arrays, stored in NIfTI or ANALYZE format, is however best performed outside of R at this point in time. Popular viewers, especially for neuroimaging data, include

- FSLView (http://www.fmrib.ox.ac.uk/fsl/fslview/),
- MRIcron (http://cabiatl.com/mricron/),
- ITKSnap (http://www.itksnap.org), and
- VolView (http://www.kitware.com/products/volview.html).

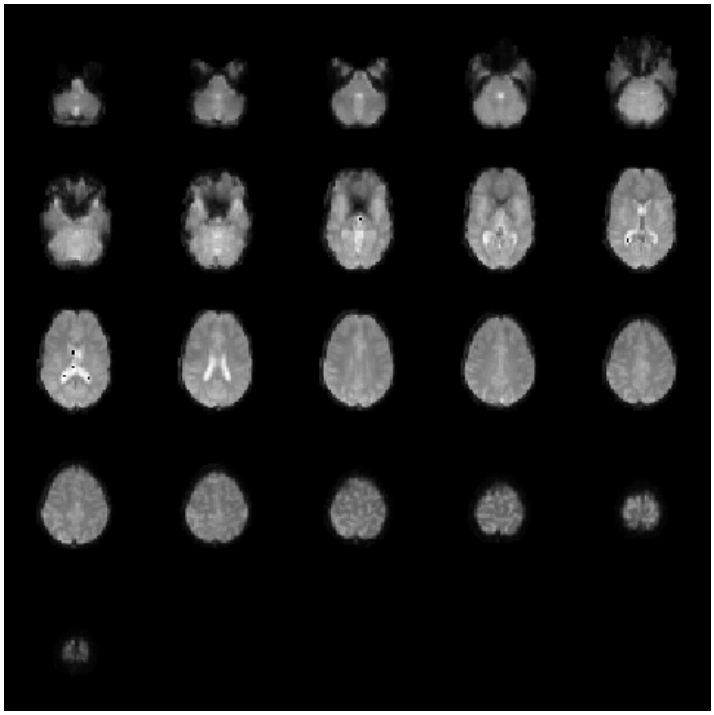
The **mritc** package provides basic interactive visualization of ANALYZE/NIfTI data using a Tcl/Tk interface (Feng and Tierney 2010).

2.7. An example using functional MRI data

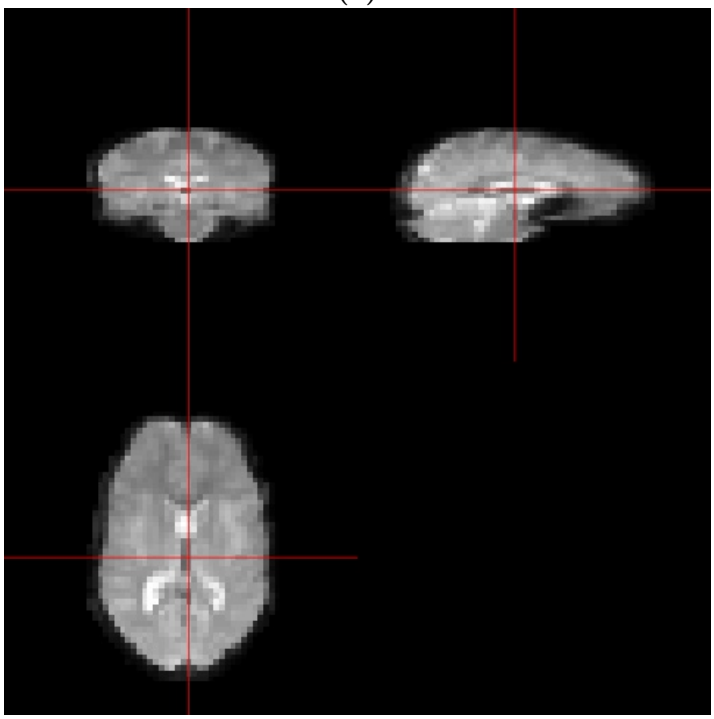
This is an example of reading in, and displaying, a four-dimensional medical imaging data set in NIfTI format `filtered_func_data` obtained from the **FSL** evaluation and example data suite (http://www.fmrib.ox.ac.uk/fsl/fsl/feeds.html). Successful execution of the commands

```
R> filtered.func.data <-
+    system.file(file.path("nifti", "filtered_func_data.nii.gz"),
+                package="oro.nifti")
R> (ffd <- readNIfTI(filtered.func.data))

NIfTI-1 format
  Type            : niftiAuditTrail
  Data Type       : 4 (INT16)
  Bits per Pixel  : 16
  Slice Code      : 0 (Unknown)
  Intent Code     : 0 (None)
  Qform Code      : 0 (Unknown)
  Sform Code      : 0 (Unknown)
  Dimension       : 64 x 64 x 21 x 64
  Pixel Dimension : 1 x 1 x 1 x 1
  Voxel Units     : Unknown
  Time Units      : Unknown

R> image(ffd, zlim=range(ffd)*0.95)
```

produces a four-dimensional (4D) array of imaging data that may be displayed in a 5×5 grid of images (Figure 4a). The first three dimensions are spatial locations of the voxel (volume element) and the fourth dimension is time for this functional MRI (fMRI) acquisition. As seen from the summary of object, there are 21 axial slices of fairly coarse resolution (4×4×6 mm) and reasonable temporal resolution (3 s). Figure 4b depicts the orthographic display of the `filtered_func_data` using the axial plane containing the left-and-right thalamus to approximately center the crosshair vertically.



(a)



(b)

Figure 4: **(a)** Axial slices of the functional MRI data set `filtered_func_data` from the first acquisition. **(b)** Orthographic display of the first volume from the functional MRI data set `filtered_func_data`.

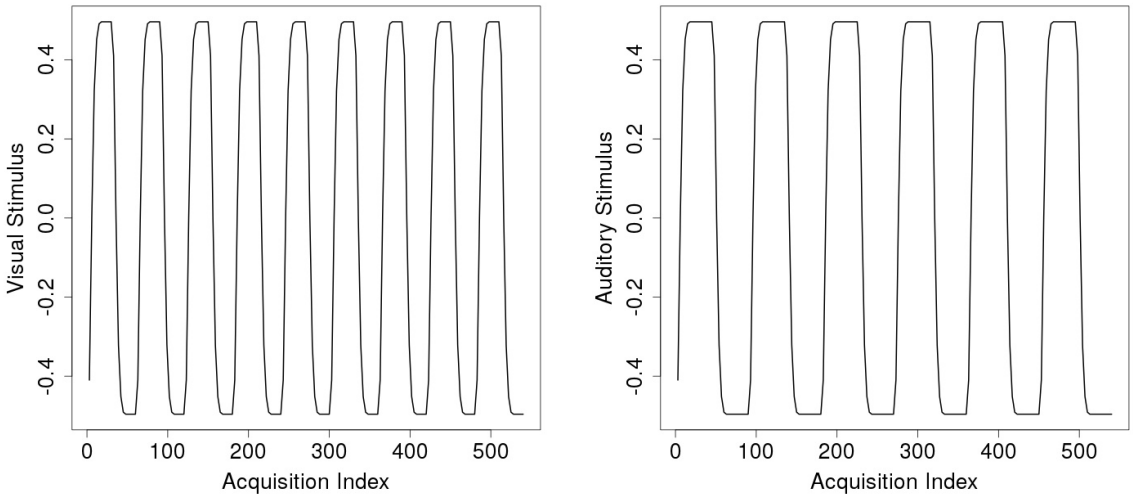


Figure 5: Visual (30 seconds on/off) and auditory (45 seconds on/off) stimuli, convolved with a parametric haemodynamic response function, used in the GLM-based fMRI analysis.

```
R> orthographic(ffd, xyz=c(34,29,10), zlim=range(ffd)*0.9)
```

Statistical analysis

The R programming environment provides a wide variety of statistical methodology for the quantitative analysis of medical imaging data. For example, functional MRI (fMRI) data are typically analyzed by applying a multiple linear regression model, commonly referred to in the literature as a general linear model (GLM), that utilizes the stimulus experiment to construct the design matrix. Estimation of the regression coefficients in the GLM produces a statistical image; e.g., Z-statistics for a voxel-wise hypothesis test on activation in fMRI experiments (Friston *et al.* 1994, 1995).

The 4D volume of imaging data in `filtered_func_data` was acquired in an experiment with a repetition time TR = 3 s, using both visual and auditory stimuli. The visual stimulus was applied using an on/off pattern for a duration of 60 seconds and the auditory stimulus was applied using an on/off pattern for a duration of 90 seconds. A parametric haemodynamic response function (HRF), with mean $\mu = 6$ and standard deviation $\sigma = 3$, is utilized here which is similar to the default values in **FSL** (Smith *et al.* 2004). We construct the experimental design and HRF in seconds, perform the convolution and then downsample by a factor of three in order to obtain columns of the design matrix that match the acquisition of the MRI data.

```
R> visual <- rep(c(-0.5,0.5), each=30, times=9)
R> auditory <- rep(c(-0.5,0.5), each=45, times=6)
R> hrf <- c(dgamma(1:15, 4, scale=1.5))
R> hrf0 <- c(hrf, rep(0, length(visual)-length(hrf)))
```

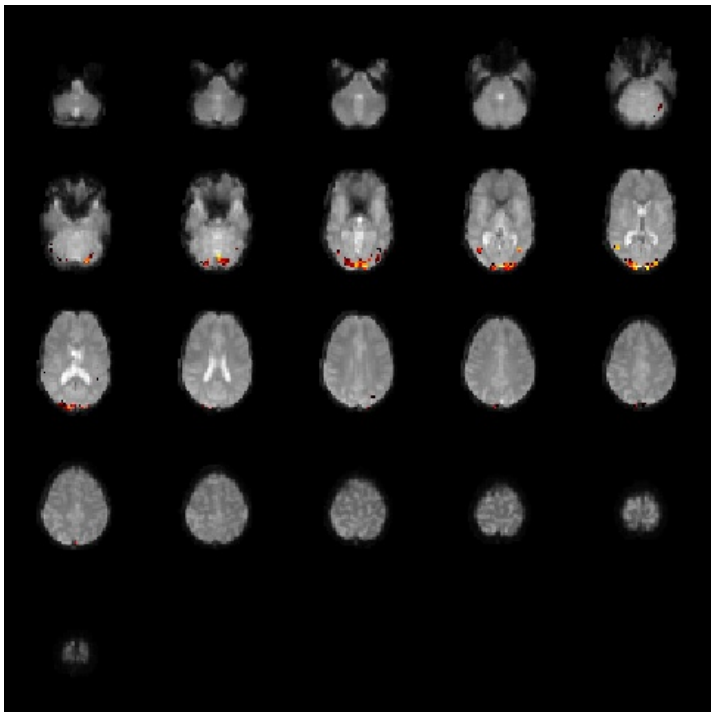
```
R> visual.hrf <- convolve(hrf0, visual)
R> hrf0 <- c(hrf, rep(0, length(auditory)-length(hrf)))
R> auditory.hrf <- convolve(hrf0, auditory)
R> index <- seq(3, 540, by=3)
R> visual.hrf <- visual.hrf[index]
R> auditory.hrf <- auditory.hrf[index]
```

Figure 5 depicts the visual and auditory stimuli, convolved with the HRF, in the order of acquisition. The design matrix is then used in a voxel-wise GLM, where the `lsfit` function in R estimates the parameters in the linear regression. At each voxel t-statistics and their associated p-values are computed for the hypothesis test of no effect for each individual stimulus, along with an F-statistic for the hypothesis test of no effect of any stimuli using the `ls.print` function.

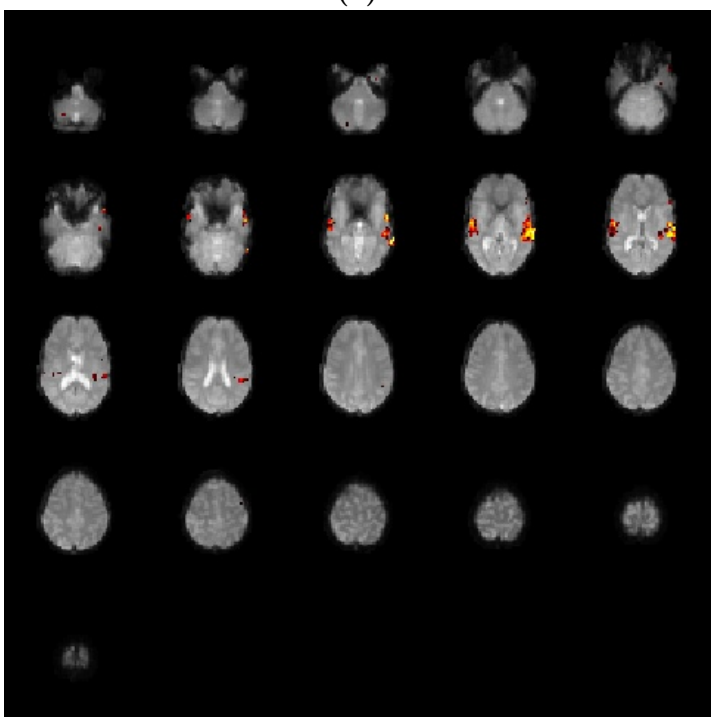
```
R> ##reduced length due to R package storage limitations
R> visual.hrf<-visual.hrf[1:64]
R> auditory.hrf<-auditory.hrf[1:64]
R> ## background threshold: 10% max intensity
R> voxel.lsfit <- function(x, thresh) { # general linear model
+    ## check against background threshold
+    if (max(x) < thresh) {
+      return(rep(NA, 5))
+    }
+    ## glm
+    output <- lsfit(cbind(visual.hrf, auditory.hrf), x)
+    ## extract t-statistic, p-values
+    output.t <- ls.print(output, print.it=FALSE)$coef.table[[1]][2:3,3:4]
+    output.f <- ls.print(output, print.it=FALSE)$summary[3]
+    c(output.t, as.numeric(output.f))
+  }
R> ## apply local glm to each voxel
R> ffd.glm <- apply(ffd, 1:3, voxel.lsfit, thresh=0.1 * max(ffd))
```

Given the multidimensional array of output from the GLM fitting procedure, the t-statistics are separated and converted into Z-statistics to follow the convention used in **FSL**. For the purposes of this example we have not applied any multiple comparisons correction procedure and, instead, use a relatively large threshold of $Z > 5$ for visualization.

```
R> dof <- ntim(ffd) - 1
R> Z.visual <- nifti(qnorm(pt(ffd.glm[1,,,], dof, log.p=TRUE), log.p=TRUE),
+                    datatype=16)
R> Z.auditory <- nifti(qnorm(pt(ffd.glm[2,,,], dof, log.p=TRUE), log.p=TRUE),
+                      datatype=16)

R> yrange <- c(5, max(Z.visual, na.rm=TRUE))
R> overlay(ffd, ifelse(Z.visual > 5, Z.visual, NA),
+          zlim.x=range(ffd)*0.95, zlim.y=yrange)
```



(a)



(b)

Figure 6: **(a)** Axial slices of the functional MRI data with the statistical image from the visual stimulus overlayed. **(b)** Axial slices of the functional MRI data with the statistical image from the auditory stimulus overlayed. Both sets of test statistics were thresholded at $Z \geq 5$ for all voxel.

```
R> yrange <- c(5, max(Z.auditory, na.rm=TRUE))
R> overlay(ffd, ifelse(Z.auditory > 5, Z.auditory, NA),
+          zlim.x=range(ffd)*0.95, zlim.y=yrange)
```

Statistical images in neuroimaging are commonly displayed as an overlay on top of a reference image (one of the dynamic acquisitions) in order to provide anatomical context. The `overlay` command in **oro.nifti** allows one to display the statistical image of voxel-wise activations overlayed on one of the original EPI (echo planar imaging) volumes acquired in the fMRI experiment. The 3D array of Z-statistics for the visual and auditory tasks are overlayed on the original data for "anatomical" reference in Figure 6. The Z-statistics that exceed the threshold appear to match know neuroanatomy, where the visual cortex in the occipital lobe shows activation under the visual stimulus (Figure 6a) and the primary auditory cortex in the temporal lobe shows activation under the auditory stimulus (Figure 6b).

3. Conclusion

Medical image analysis depends on the efficient manipulation and conversion of DICOM data. The **oro.nifti** 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.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. The authors would also like to thank K. Tabelow for providing functionality around the AFNI data format. VS is supported by the German Research Council (DFG SCHM 2747/1-1).

References

Bordier C, Dojat M, Lafaye de Micheaux P (2009). "**AnalyzeFMRI**: an R Package to Perform Statistical Analysis on FMRI Datasets." Software: R Package, **AnalyzeFMRI**, version 1.1-12, URL http://www.biostatisticien.eu/AnalyzeFMRI/.

Chambers JM (2008). *Software for Data Analysis: Programming in R*. Springer, New York.

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.

Clayden J (2011). ***RNiftyReg**: Medical Image Registration Using the NiftyReg Library*. R package version 0.3.1, based on original code by Marc Modat and Pankaj Daga, URL http://CRAN.R-project.org/package=RNiftyReg.

Feng D, Tierney L (2010). ***mritc**: MRI Tissue Classification*. R package version 0.3-1, URL http://CRAN.R-project.org/package=mritc.

Ferreira da Silva A (2010). ***dpmixsim**: Dirichlet Process Mixture Model Simulation for Clustering and Image Segmentation*. R package version 0.0-5, URL http://CRAN.R-project.org/package=dpmixsim.

Ferreira da Silva A (2011). "**cudaBayesreg**: Parallel Implementation of a Bayesian Multilevel Model for fMRI Data Analysis." *Journal of Statistical Software*, **44**(4), 1–24. URL http://www.jstatsoft.org/v44/i04.

Friston KJ, Holmes AP, Poline JB, Grasby PM, Williams SCR, Frackowiak RSJ, Turner R (1995). "Analysis of fMRI Time Series Revisited." *NeuroImage*, **2**, 45–53.

Friston KJ, Holmes AP, Worsley KJ, Poline JP, Frith CD, Frackowiak RSJ (1994). "Statistical Parametric Maps in Functional Imaging: A General Linear Approach." *Human Brain Mapping*, **2**, 189–210.

Granert O (2010). ***Rniftilib**: R Interface to NIFTICLIB (V1.1.0)*. R package version 0.0-29, URL http://CRAN.R-project.org/package=Rniftilib.

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.

Sklyar O, Pau G, Smith M, Huber W (2010). ***EBImage**: Image Processing Toolbox for R*. R package version 3.5.5.

Smith SM, Jenkinson M, Woolrich MW, Beckmann CF, Behrens TEJ, Johansen-Berg H, Bannister PR, Luca MD, Drobnjak I, Flitney DE, Niazy R, Saunders J, Vickers J, Zhang Y, De Stefano N, Brady JM, Matthews PM (2004). "Advances in Functional and Structural MR Image Analysis and Implementation as FSL." *NeuroImage*, **23**(Supplement 1), 208–219.

Temple Lang D (2010). ***XML**: Tools for Parsing and Generating XML within R and S-Plus*. R package version 3.1-0, URL http://CRAN.R-project.org/package=XML.

Whitcher B, Schmid VJ (2011). "Quantitative Analysis of Dynamic Contrast-Enhanced and Diffusion-Weighted Magnetic Resonance Imaging for Oncology in R." **44**(5), 1–29. URL http://www.jstatsoft.org/v44/i05/.

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