`oro.nifti`: Rigorous - NIfTI Input / Output

Brandon Whitcher
bjw34032@users.sourceforge.net
Volker J. Schmid
volkerschmid@users.sourceforge.net
Andrew Thornton
zeripath@users.sourceforge.net

June 5, 2011

1 Introduction

The **oro.nifti** package requires incoming data to be in the ANALYZE˜7.5 or NIfTI formats. Data acquisition and input (e.g., from DICOM using **oro.dicom**) must be performed by the user before **oro.nifti** may be used to summarize the data. Several software packages allow DICOM-to-NIfTI (or ANALYZE) conversion; e.g.,

- oro.dicom (rigorous.r-forge.r-project.org)
- FreeSurfer (surfer.nmr.mgh.harvard.edu)
- Xmedcon (xmedcon.sourceforge.net)
- MRIConvert (lnci.oregon.edu/˜jolinda/MRIConvert)

This is by no means an exhaustive list of software available for DICOM conversion.

1.1 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 x-axis to increase from right to left (neurological). Future versions of **oro.nifti** will attempt to address more complicated transformations.

1.2 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. Essentially, NIfTI and ANALYZE data are stored as multi-dimensional arrays with extra slots created that capture the format-specific header information. The NIfTI class also has the ability to read and write extensions that conform to the data format standard. Customized printing and validity-checking functions are available to the user and every attempt is made to ensure that the information from the multi-dimensional array is in agreement with the header values.

1.3 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. First, extensions are properly handled when reading and writing NIfTI data. Second, users are allowed to add extensions to newly-created NIfTI objects using various functions and the **XML** package. Third, by default all operations that are performed on a NIfTI object will generate what we call an *audit trail* that consists of an XML-based log. 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 appropriately (ecode = 6). The user may control the tracking of data manipulation via the audit trail using a global option. For example please use the command

```
> options(niftiAuditTrail = FALSE)
```

to turn off the "audit trail" option in **oro.nifti**.

Interactive visualization of multidimensional arrays, stored in NIfTI or Analyze format, is best performed outside of R at this point in time. Popular viewers, especially for brain imaging, are

- FSLView (http://www.fmrib.ox.ac.uk/fsl/fslview/)
- MRIcroN (http://www.sph.sc.edu/comd/rorden/MRicron/)

1.4 Examples

1.4.1 Labelled LR Standard (MNI152) Images in NIfTI Format

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 (nifti.nimh.nih.gov/nifti-1/). Successful execution of the command:

```
> (mni.LR <- readNIfTI(system.file("nifti/mniLR.nii.gz", package = "oro.nifti")))

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
```

```
> audit.trail(mni.LR)
```

```
<audit-trail xmlns="http://www.dcemri.org/namespaces/audit-trail/1.0">
  <created>
    <workingDirectory>/tmp/RtmpwS9edk/Rbuilde3edb5d/oro.nifti/inst/doc</workingDirectory>
    <filename>/tmp/RtmpwS9edk/Rinst4cf4170/oro.nifti/nifti/mniLR.nii.gz</filename>
    <call>readNIfTI(fname = system.file("nifti/mniLR.nii.gz", package = "oro.nifti"))</cal
    <system>
      <r-version.version.string>R version 2.13.0 Patched (2011-06-04 r56046)</r-version.ve
      <date>Sun Jun 05 11:18:47 PM 2011 CEST</date>
      <user>rforge</user>
      <package-version.Version>0.2.6</package-version.Version>
    </system>
  </created>
</audit-trail>
```

```
> descrip(mni.LR)
```

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

```
> image(mni.LR)
```

produces a 4D array of the image data, with the default NIfTI axes, and is displayed on a 10×10 grid of images (Figure˜1). Note, the `image` function has been modified to accept `nifti` and `anlz` objects and display them with minimal user input. Two accessor functions are also shown here: `audit.trail` and `descrip`. The former is used to access the XML-based audit trail that is stored as a NIfTI header extension and the latter is the name of a valid NIfTI header field (allowed to store up to 80 characters).

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

```
> (mni.RL <- readNIfTI(system.file("nifti/mniRL.nii.gz", package = "oro.nifti")))
```

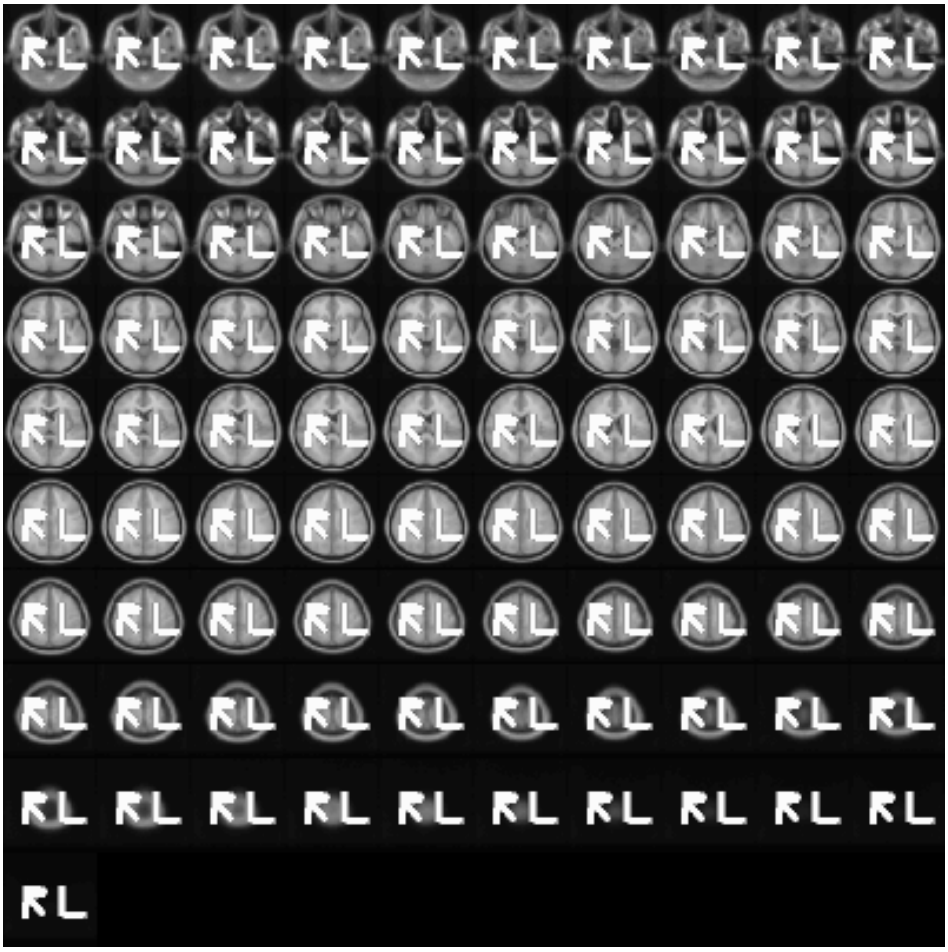

Figure 1: Axial slices of MNI volume `mniLR_nifti` stored in radiological convention.

```
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
```

```
> image(mni.RL)
```

produces a 4D array of the image data that may be displayed in a 10×10 grid of images (Figure˜2).

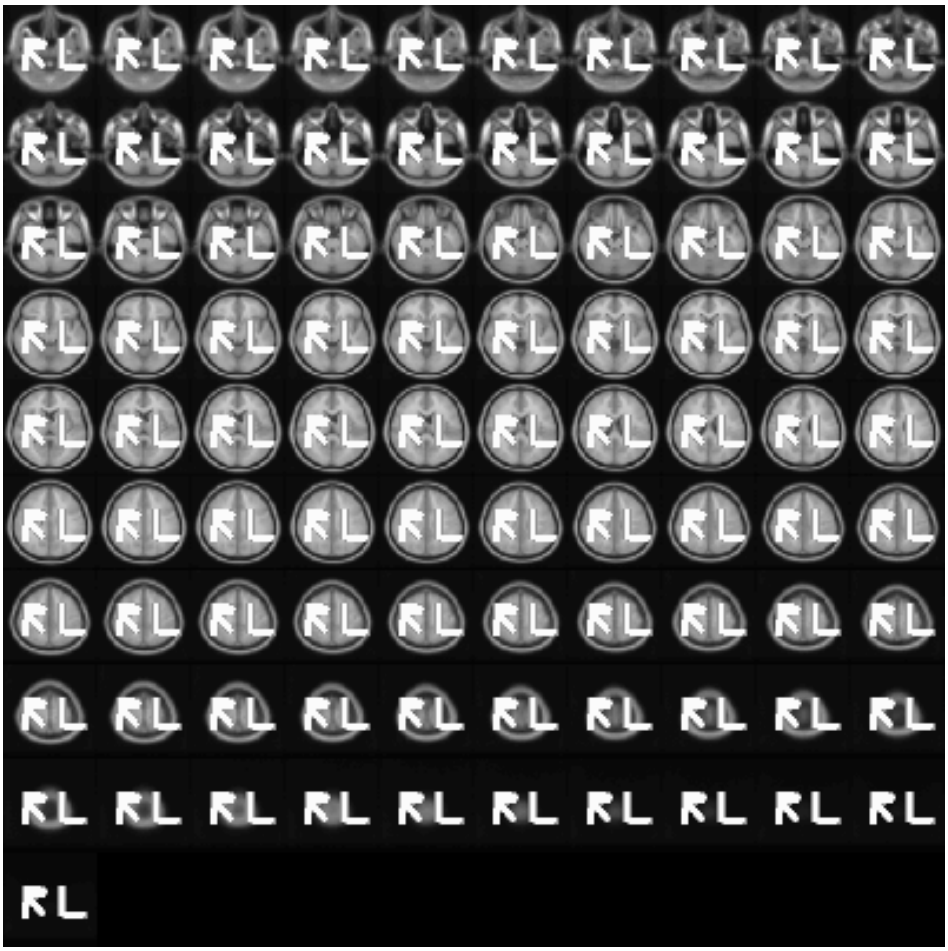

Figure 2: Axial slices of MNI volume `avg152T1_RL_nifti` stored in neurological convention.

The first image (LR) is stored in radiological convention. The second image (RL) is stored in neurological convention. Any NIfTI-1 compliant viewing software should display these images identically.

1.4.2 Simple Time-series or Multi-volume Image

This is an example of reading in, and displaying, a four-dimensional medical imaging data set in NIfTI format (`filtered_func_data.nii`) obtained from the NIfTI website (nifti.nimh.nih.gov/nifti-1/). Successful execution of the command

```
> (ffd <- readNIfTI(system.file("nifti/ffd.nii.gz", package = "oro.nifti")))

NIfTI-1 format
  Type            : niftiAuditTrail
  Data Type       : 4 (INT16)
  Bits per Pixel  : 16
  Slice Code      : 0 (Unknown)
  Intent Code     : 0 (None)
  Qform Code      : 1 (Scanner_Anat)
  Sform Code      : 1 (Scanner_Anat)
  Dimension       : 64 x 64 x 21 x 18
  Pixel Dimension : 4 x 4 x 6 x 3
  Voxel Units     : mm
  Time Units      : sec

> image(ffd)
```

produces a four-dimensional (4D) array of imaging data that may be displayed in a 5×5 grid of images (Figure˜3). The first three dimensions are spatial locations of the voxel (volume element) and the fourth dimension is time.

An additional graphical display function has been added for `nifti` and `anlz` objects that allows orthographic displays.

```
> orthographic(ffd)
```

1.4.3 Statistic Image

This is an example of reading in and displaying a statistical image so that it may be overlayed on the EPI (echo planar imaging) data taken from the functional MRI experiment. The original NIfTI files (`filtered_func_data.nii` and `zstat1.nii`) were obtained from the NIfTI website (nifti.nimh.nih.gov/nifti-1/). Successful execution of the command

```
> (zstat1 <- readNIfTI(system.file("nifti/zstat1.nii.gz", package = "oro.nifti")))
```

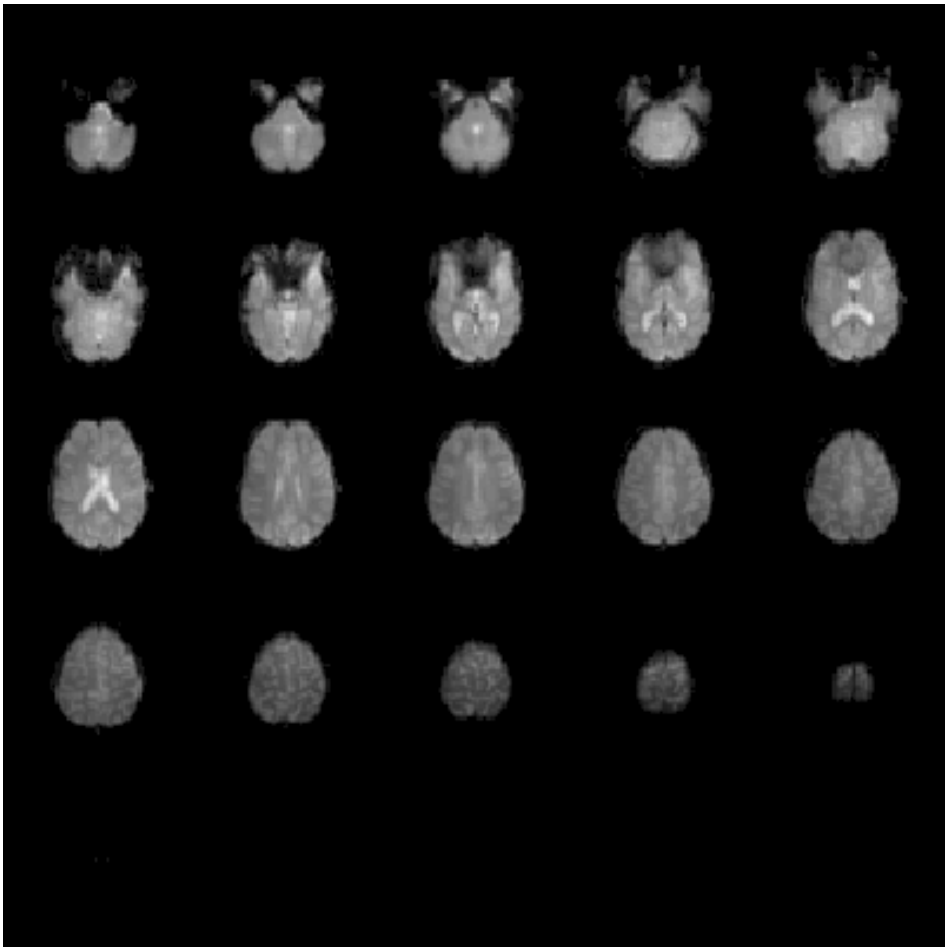

Figure 3: Axial slices of the functional MRI "volume" `filtered_func_data` from the first acquisition.

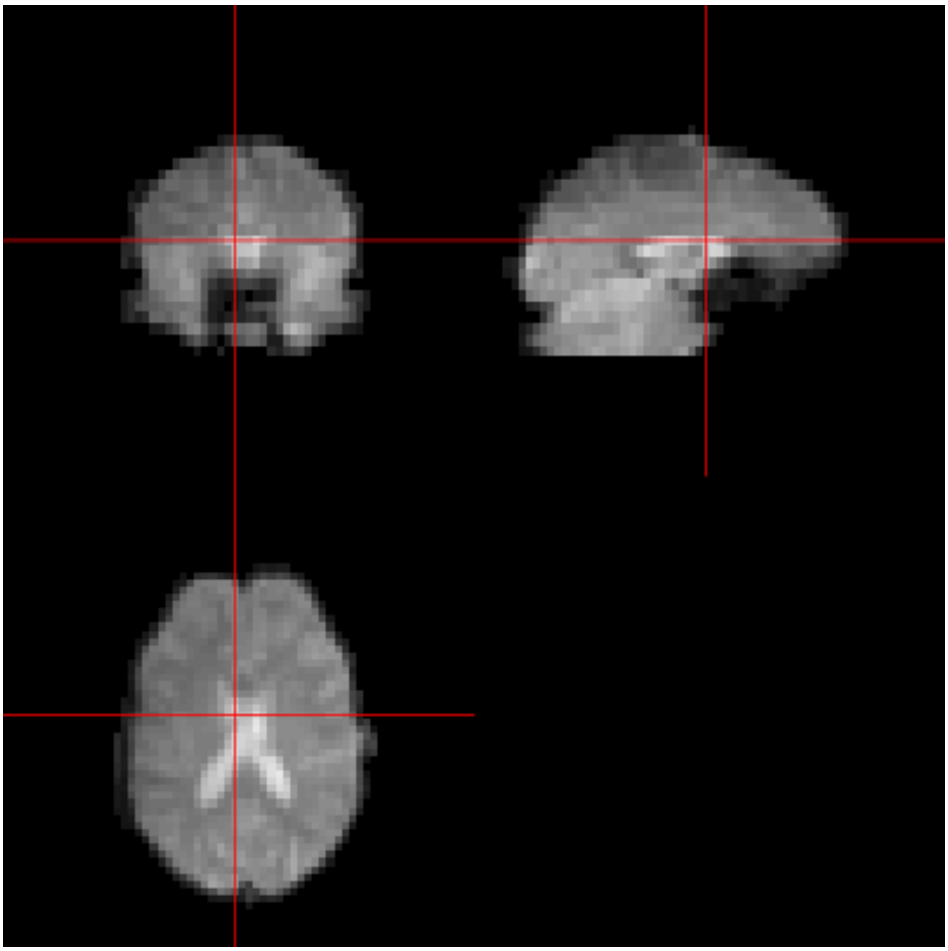

Figure 4: Orthographic display of the first volume from the functional MRI dataset `filtered_func_data`. By default the mid-axial, mid-saggittal and mid-coronal planes are chosen.

```
NIfTI-1 format
  Type            : niftiAuditTrail
  Data Type       : 16 (FLOAT32)
  Bits per Pixel  : 32
  Slice Code      : 0 (Unknown)
  Intent Code     : 5 (Zscore)
  Qform Code      : 1 (Scanner_Anat)
  Sform Code      : 0 (Unknown)
  Dimension       : 64 x 64 x 21
  Pixel Dimension : 4 x 4 x 6
  Voxel Units     : mm
  Time Units      : sec
```

```
> overlay(ffd, ifelse(abs(zstat1) > 5, zstat1, NA), zlim.y = range(zstat1))
```

produces a 4D array of parameter estimates (essentially coefficients from a linear regression performed at each voxel) that may be overlayed on the original data for anatomical reference (Figure˜5). The function `overlay` extends the capabilities of displaying "images" by allowing one to add a statistical image to an underlying structural image.

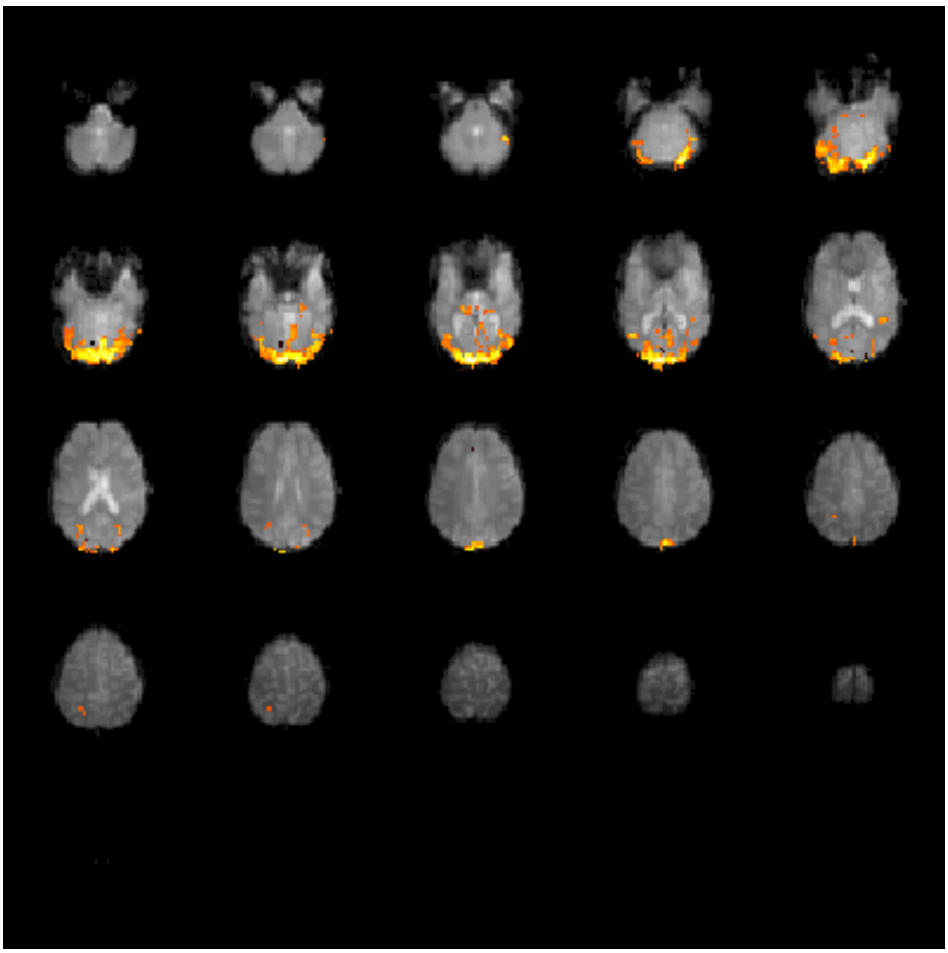

Figure 5: Axial slices of the functional MRI data with with the statistical image overlayed. The test statistics were thresholded at $|Z| \geq 5$.