Displaying Image Information

John Muschelli

Johns Hopkins Bloomberg School of Public Health

January 25, 2015

Reading In Data

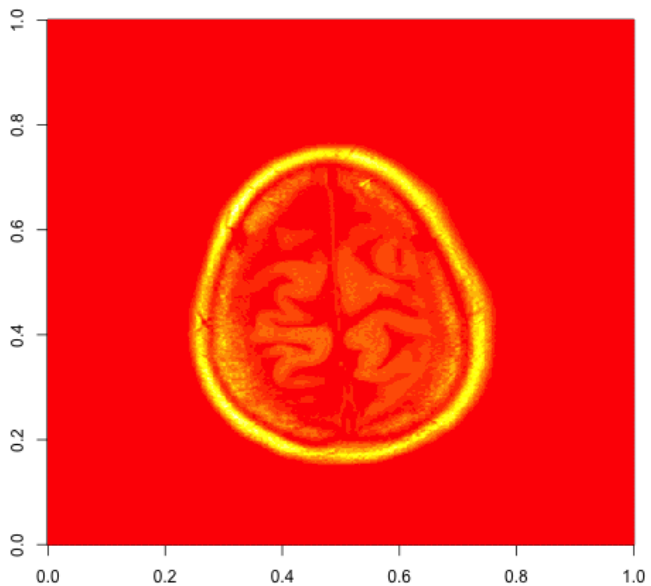
The `readNIfTI` command (`oro.nifti` package) can read in NIfTI file (compressed or not) into a `nifti` object. We will read in the NIfTI (T1-weighted) image from the previous session:

```
library(oro.nifti)
print({nii = readNIfTI(fname = "Output_3D_File")})
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        : 512 x 512 x 22
  Pixel Dimension  : 0.47 x 0.47 x 5
  Voxel Units      : mm
  Time Units       : sec
```

Visualizing a Slice

The `nifti` object is a 3D `array` (see `?array`) with header information. We can use the `image` function (`graphics` package) to visualize a slice (slice 20 in the z-direction/axial):

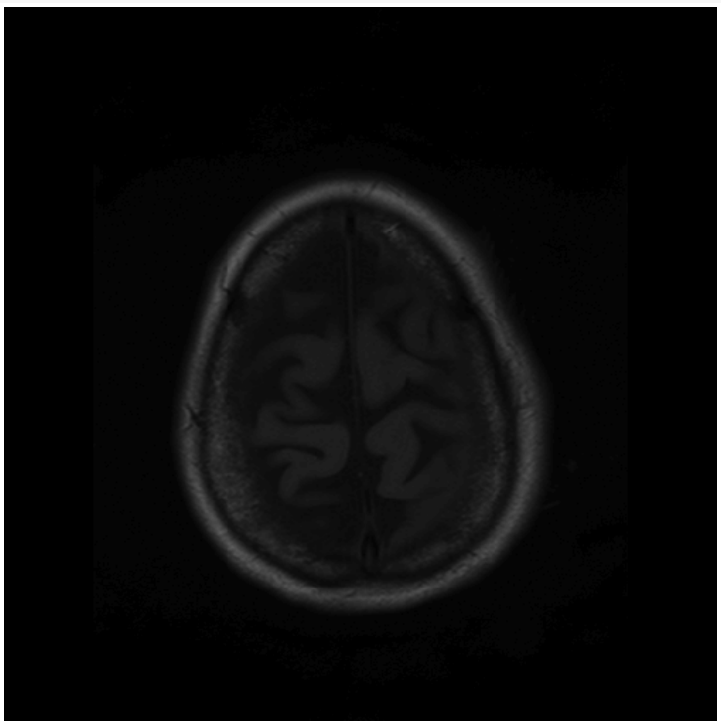
```
image(nii[,,20])
```



Visualizing a Slice

`graphics::image` uses `heat.colors(12)` for coloring, which is not useful for this task. We can either set the colors manually, or use the `oro.nifti::image` function. The function is still `image`, but we don't pass in a slice, but the `nifti` object and specify the slice z=20:

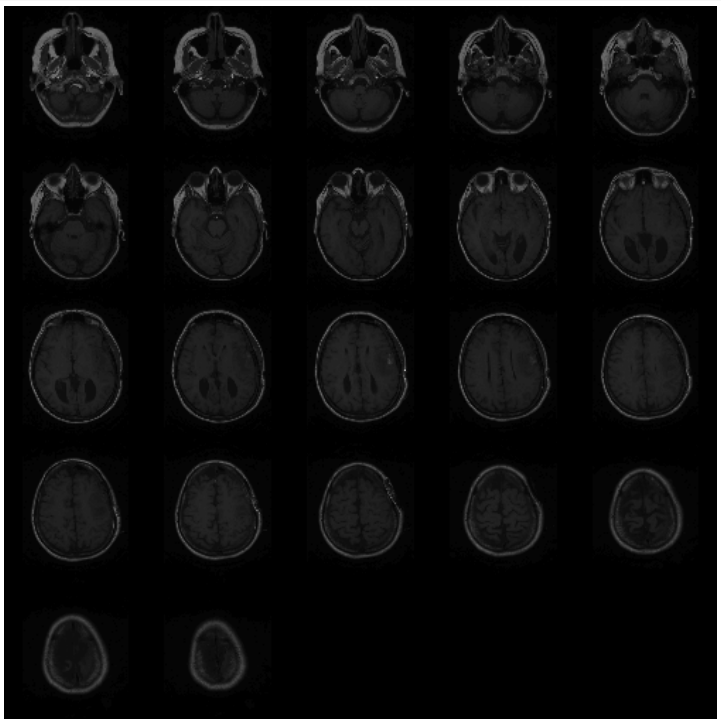
```
image(nii, z = 20, plot.type='single')
```



Visualizing a Slice

If `plot.type` is not `'single'`, `image.nifti` defaults to plotting ALL slices with data, even if `z` is specified (also called a "lightbox"):

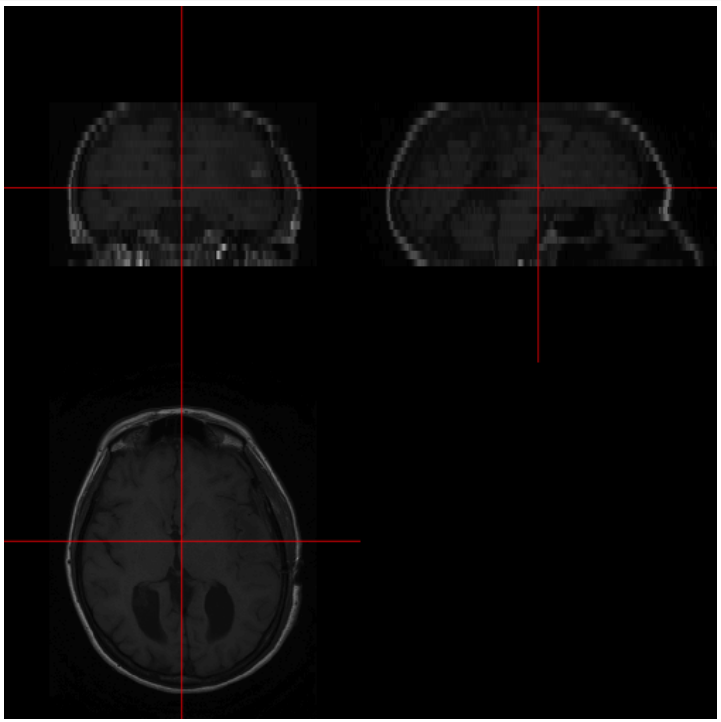
```
image(nii, z = 20)
```



Visualizing all 3 planes

To show all 3 planes (axial, sagittal, and coronal) of an image, we can use the `orthographic` function:

```
orthographic(nii)
```



Histograms

What about the **data**? We can do normal operations, such as histograms of the image intensities and intensities over 20:

```
par(mfrow=c(1,2));
hist(nii, breaks = 2000); hist(nii[nii > 20], breaks = 2000
```

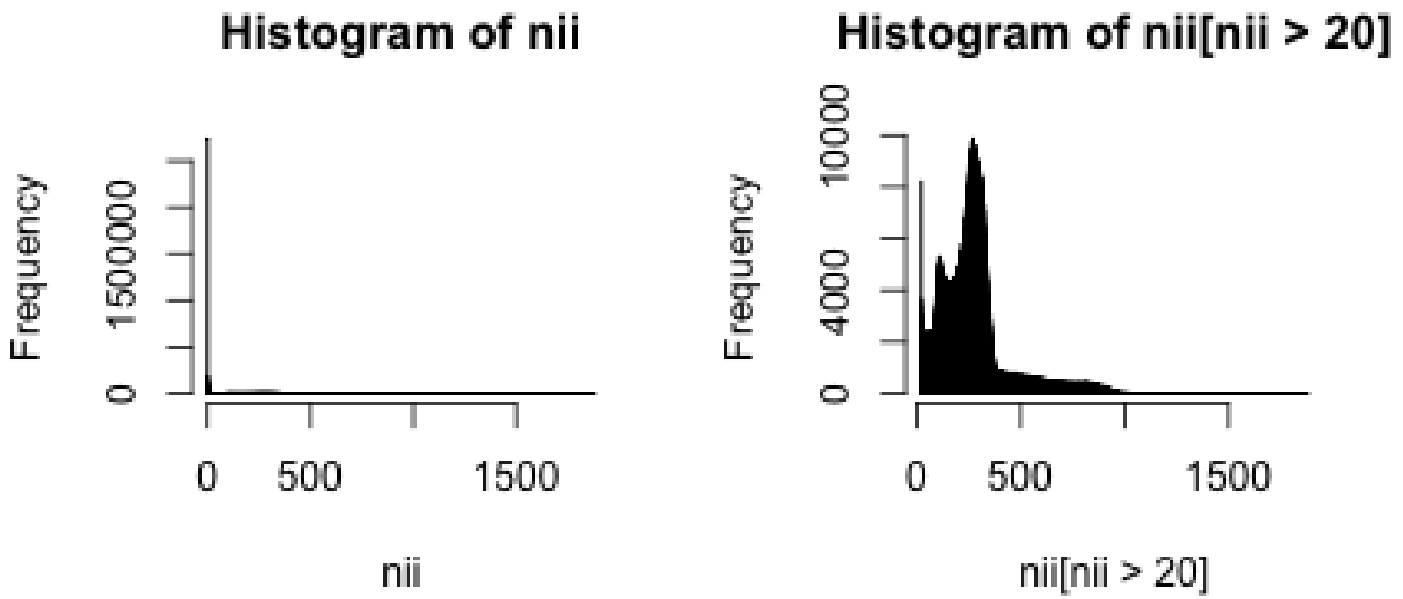


Image Overlays

We can do overlays as well, where we have one image and color it by a second. For example, we plot slice 10 and highlight values between 300 and 400 (next slide we discuss the code):

```
library(fslr) # need niftiarr
mask = fslr::niftiarr(nii, nii > 300 & nii < 400)
mask[mask == 0] = NA
overlay(nii, mask, col.y= c("red"),
        plot.type="single", z = 10)
```

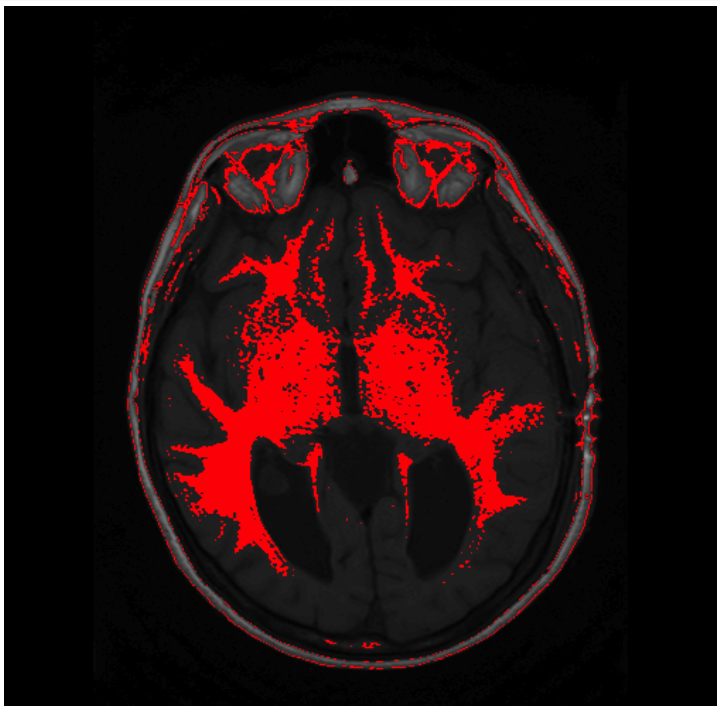


Image Overlays: Explained

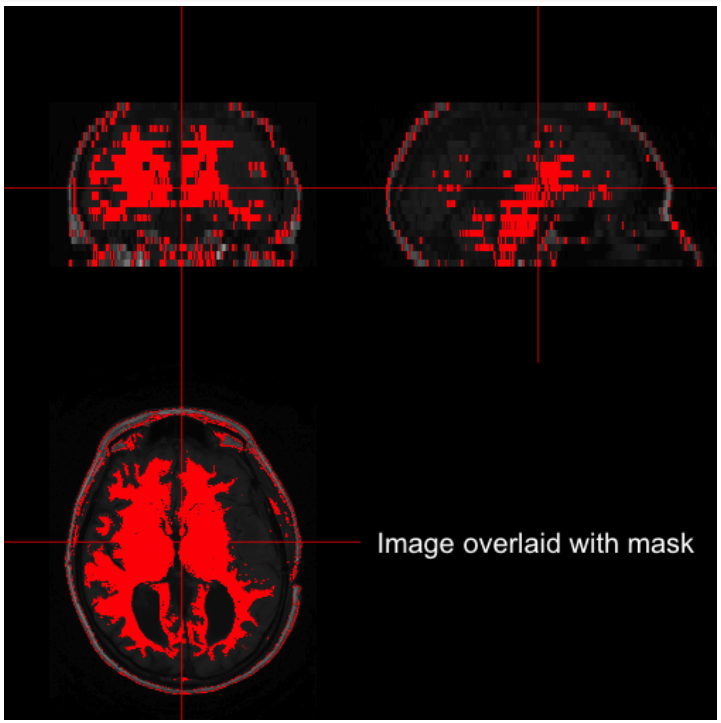
We load the `fslr` package, which has helper functions for `nifti` objects. The `nii > 300 & nii < 400` operation returns an `array`, not `nifti` object. The `niftiarr` command takes in a `nifti` object and array and returns a `nifti` object with the array in the data slot. We then set any 0 in mask to `NA` so the `overlay` (`oro.nifti` package) will not mask out data.

```
library(fslr) # need niftiarr
mask = fslr::niftiarr(nii, nii > 300 & nii < 400)
mask[mask == 0] = NA
overlay(nii, mask, col.y= c("red"),
        plot.type="single", z = 10)
```

Image Overlays: 3 Planes

We can perform the same operation of overlaying, but in all 3 planes:

```
orthographic(nii, mask, col.y= c("red"),
                text ="Image overlaid with mask")
```



Functions discussed here

- `readNIfTI`: read in data
- `graphics::image`: display matrix data
- `oro.nifti::image`: display `nifti` data
- `oro.nifti::orthographic`: display 3-planes of an image
- `oro.nifti::overlay`: display overlay of 2 images (NA are not plotting in `y` image)
- `fslr::niftiarr(x,y`: Copy `nifti` header from object `x` and put in new `array` `y` into data part