

A tutorial for resting state fMRI analysis using ANTsR

Introduction

Overview

fMRI issues

- Nuisance signal from CSF and WM [1]
- Bandpass filtering
- Motion correction [2, 3]
- Global signal[4]

ANTsR implementation

The main fMRI-specific functions are:

- `fMRINormalization`
- `preprocessRestingBOLD` (supplants `preprocessfMRI?`)
- `antsBold`
- `antsMotionCalculation` (supplants `antsMotionCorr` and `antsMotionCorrStats?`)
- `antsSpatialICAfMRI`
- `filterfMRIforNetworkAnalysis`
- `frequencyFilterfMRI`
- `getfMRIin nuisanceVariables`

Helper functions include:

- `timeseries2matrix`
- `matrixToImages`
- `icaWhiten`

Tutorial

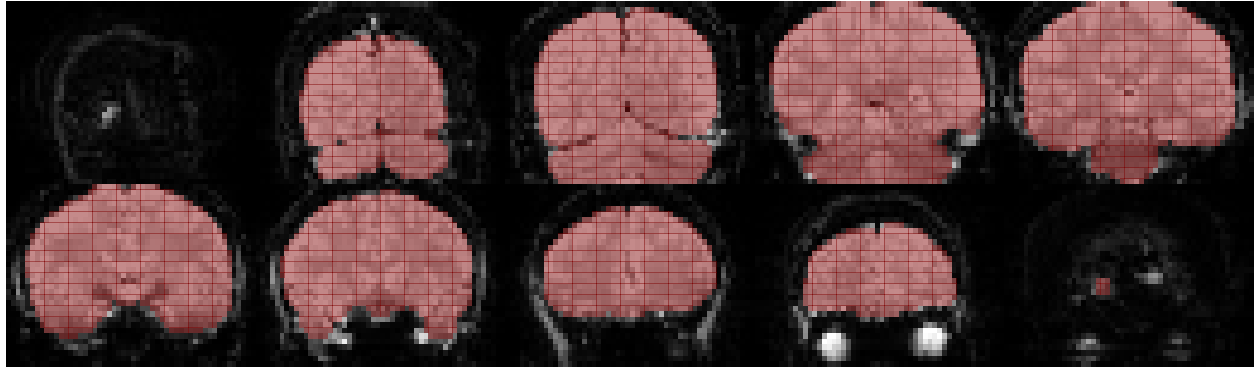
Initialization

```
# We include all the necessary R package dependencies. We assume that the user  
# is running this script (stitchTutorialDocument.R) in the repo directory.
```

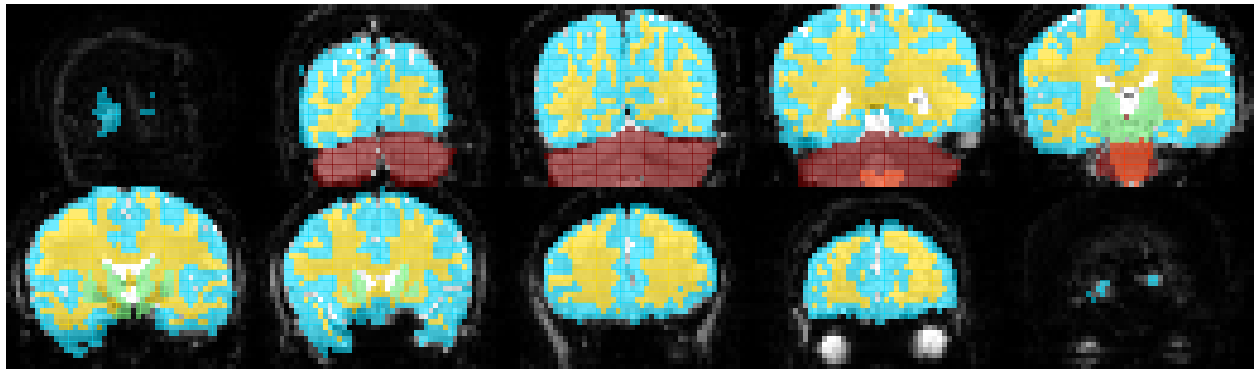
```
invisible( suppressMessages( library( ANTsR ) ) )  
library( pander )  
library( ggplot2 )  
  
rootDirectory <- "./"  
knitr::opts_knit$set( root.dir = rootDirectory )  
knitr::opts_chunk$set( comment = "" )  
  
figuresDirectory <- paste0( rootDirectory, "Figures/" )  
if( ! dir.exists( figuresDirectory ) )  
{  
  dir.create( paste0( rootDirectory, "Figures/" ) )  
}  
dataDirectory <- paste0( rootDirectory, "Data/" );
```

Read in input data

```
# Load the AAL (Automated Anatomical Labeling) data table and the AAL label image.  
# Also load the individual subject resting state BOLD images: 4-D bold, 3-D bold  
# mask image, and 3-D segmentation (csf, gm, wm, etc.) image.  
  
data( aal, package = 'ANTsR' )  
aalLabelTable <- aal  
aalFileName <- paste0( dataDirectory, "aal.nii.gz" )  
aalImage <- antsImageRead( filename = aalFileName, dimension = 3 )  
  
restingStateBoldFile <- paste0( dataDirectory, "rsbold.nii.gz" )  
restingStateBoldImage <- antsImageRead( restingStateBoldFile, dimension = 4 )  
  
restingStateBoldMaskFile <- paste0( dataDirectory, "rsboldmask.nii.gz" )  
restingStateBoldMaskImage <- antsImageRead( restingStateBoldMaskFile, dimension = 3 )  
  
restingStateBoldSegFile <- paste0( dataDirectory, "rsboldseg.nii.gz" )  
restingStateBoldSegImage <- antsImageRead( restingStateBoldSegFile, dimension = 3 )  
  
# Let's look at the images to make sure things make sense, e.g. masks are aligned.  
# Average of 4-D bold with mask superimposed  
  
restingStateBoldAverage <- getAverageOfTimeSeries( restingStateBoldImage )  
invisible( plot.antsImage( restingStateBoldAverage, restingStateBoldMaskImage,  
  alpha = 0.75, ncolumns = 5 ) )
```



```
# Average of 4-D bold with segmentation mask superimposed
invisible( plot.antsImage( restingStateBoldAverage, restingStateBoldSegImage,
  alpha = 0.9, ncolumns = 5 ) )
```



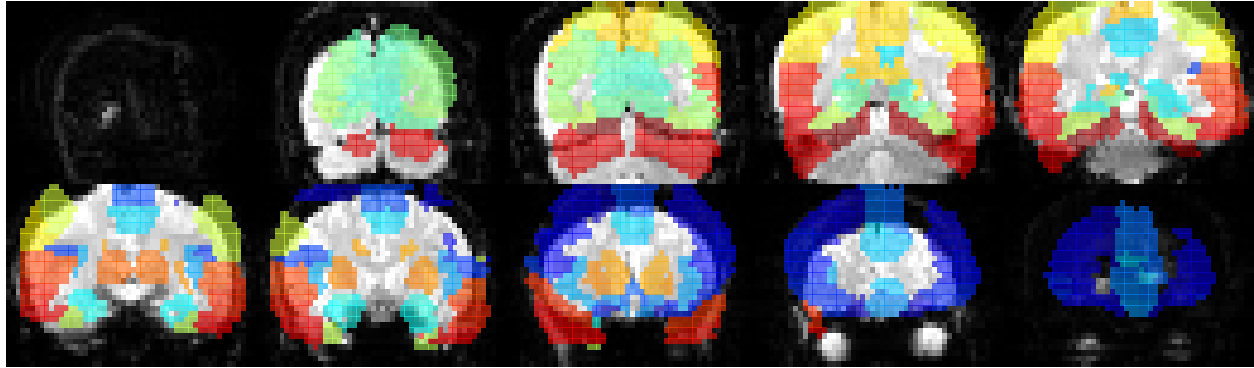
Spatially normalize AAL image

```
# The AAL image (which will be used later in the tutorial) is not in the space of the
# BOLD image so we do a quick registration of the AAL labels to the bold mask. We
# first do an "AffineFast" transform to see if that has sufficient degrees of freedom.
```

```
aalRegistration <- antsRegistration( fixed = restingStateBoldMaskImage,
  moving = aalImage, typeofTransform = "AffineFast",
  outprefix = paste0( dataDirectory, "rsboldxaal" ) )
```

```
aalWarpedImage <- antsApplyTransforms( fixed = restingStateBoldMaskImage,
  moving = aalImage, interpolator = 'genericLabel',
  transformlist = aalRegistration$fwdtransforms )
```

```
invisible( plot.antsImage( restingStateBoldAverage, aalWarpedImage,
  alpha = 0.9, ncolumns = 5 ) )
```

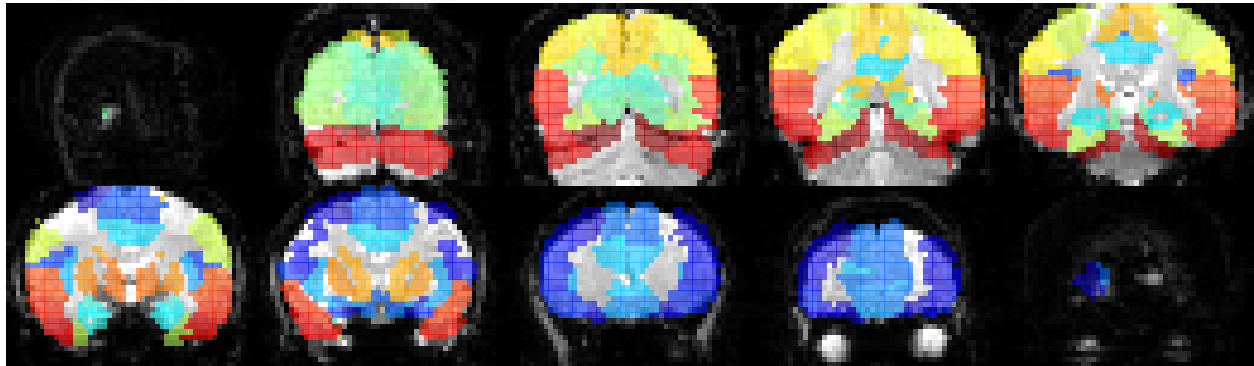


*# Clearly the alignment is not as good as we would like so we redo the registration
using an "ElasticSyN" transform which looks much better.*

```
aalRegistration <- antsRegistration( fixed = restingStateBoldMaskImage,
                                     moving = aalImage, typeofTransform = "ElasticSyN",
                                     outprefix = paste0( dataDirectory, "rsboldxaal" ) )

aalWarpedImage <- antsApplyTransforms( fixed = restingStateBoldMaskImage,
                                       moving = aalImage, interpolator = 'genericLabel',
                                       transformlist = aalRegistration$fwdtransforms )

invisible( plot.antsImage( restingStateBoldAverage, aalWarpedImage,
                           alpha = 0.9, ncolumns = 5 ) )
```



Preprocessing the resting state fMRI data

*# The evolution of fMRI functionality in ANTsR is still ongoing. It began with
various utility functions to perform different aspects of fMRI preprocessing
(e.g., motion correction, band-pass filtering). The function ``preprocessfMRI``
was created to join all these components into a single function with slight
enhancements made to create the function ``preprocessRestingBOLD``. We should
probably deprecate the former but haven't yet. Although this basic functionality
should suffice for most users, Brian has recently created the function
``fmriNormalization`` to take advantage of fMRI with simultaneous structural T1-
weighted acquisitions that have been processed through the ``antsCorticalThickness.sh``
script.*

```

# We process our current subject with ``preprocessRestingBOLD`` and plot the average
# of the resulting processed fMRI.

preprocessedRestingState <-
  preprocessRestingBOLD( restingStateBoldImage,
    maskImage = restingStateBoldMaskImage,
    denseFramewise = FALSE, numberOfCompCorComponents = 6,
    doMotionCorrection = TRUE, motionCorrectionAccuracyLevel = 0,
    motionCorrectionIterations = 1, frequencyLowThreshold = 0.01,
    frequencyHighThreshold = 0.1,
    spatialSmoothingType = "gaussian",
    spatialSmoothingParameters = 2 )

pander( summary( preprocessedRestingState ), style = "rmarkdown",
  caption = "Returned values from the function preprocessRestingBOLD." )

```

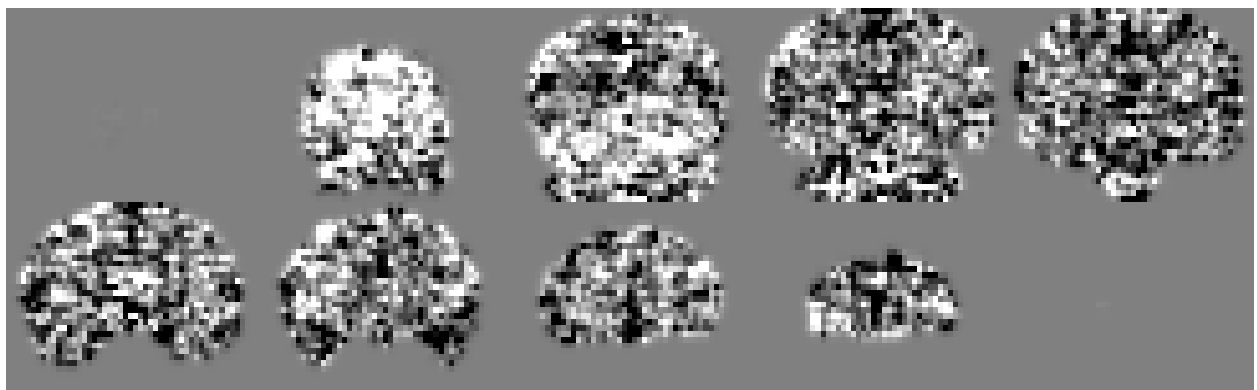
Table 1: Returned values from the function preprocessRestingBOLD.

	Length	Class	Mode
cleanBoldImage	1	antsImage	S4
maskImage	1	antsImage	S4
DVARS	225	-none-	numeric
DVARSpotCleaning	225	-none-	numeric
FD	225	-none-	numeric
globalSignal	225	-none-	numeric
nuisanceVariables	1350	-none-	numeric

```

invisible( plot.antsImage(
  getAverageOfTimeSeries( preprocessedRestingState$cleanBoldImage ), ncolumns = 5 ) )

```



```

# We continue to check the preprocessing by plotting:
# 1. the framewise displacement (FD)
# 2. the global signal before and after regression (globalSignal)
# 3. comparing the DVARS of the original data (DVARS) and the processed
#    data (DVARSpotCleaning)

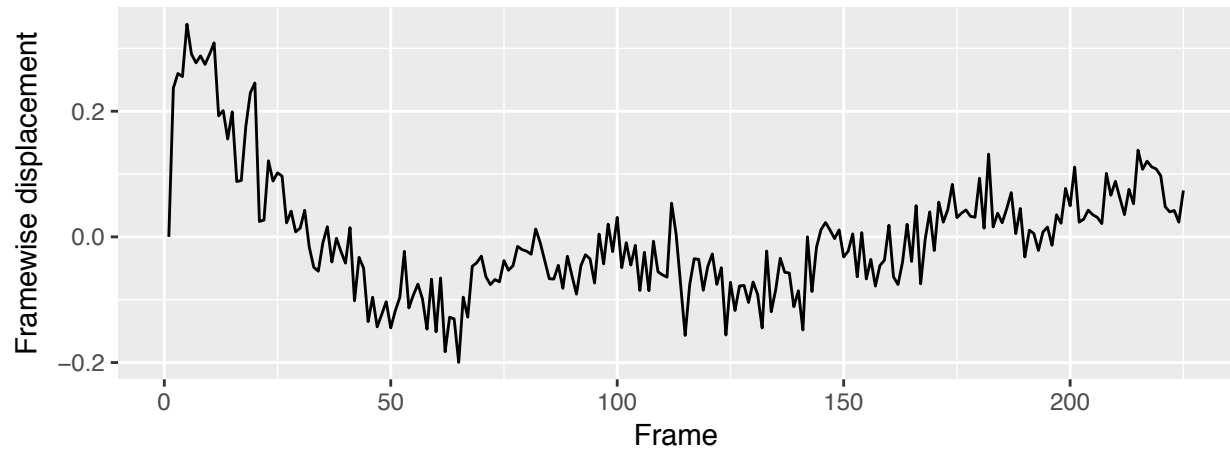
numberOfTimeFrames <- dim( restingStateBoldImage )[4]

```

```
# Plot the framewise displacement.
```

```
fdDataFrame <- data.frame( Frame = 1:numberOfTimeFrames,
  FD = preprocessedRestingState$FD - mean( preprocessedRestingState$FD ) )
```

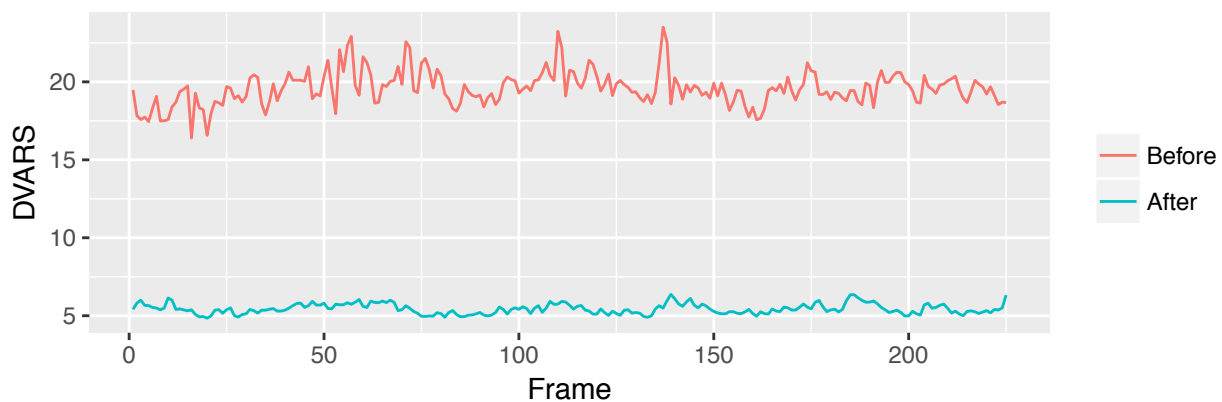
```
ggplot( fdDataFrame ) +
  geom_line( aes( x = Frame, y = FD ), size = 0.5 ) +
  xlab( "Frame" ) + ylab( "Framewise displacement" ) +
  theme( legend.title = element_blank() ) + theme( aspect.ratio=1/3 )
```



```
# Plot the DVARs. Defined as the framewise backwards RMS voxelwise difference averaged  
# over each time frame.
```

```
dvarsDataFrame <- data.frame( Frame = rep( 1:numberOfTimeFrames, 2 ),
  DVARs = c( preprocessedRestingState$DVARs,
    preprocessedRestingState$DVARs$postCleaning ),
  Type = factor( c( rep( "Before", numberOfTimeFrames ),
    rep( "After", numberOfTimeFrames ) ), levels = c( "Before", "After" ) ) )
```

```
ggplot( dvarsDataFrame ) +
  geom_line( aes( x = Frame, y = DVARs, colour = Type ), size = 0.5 ) +
  xlab( "Frame" ) + ylab( "DVARs" ) +
  theme( legend.title = element_blank() ) + theme( aspect.ratio=1/3 )
```



```

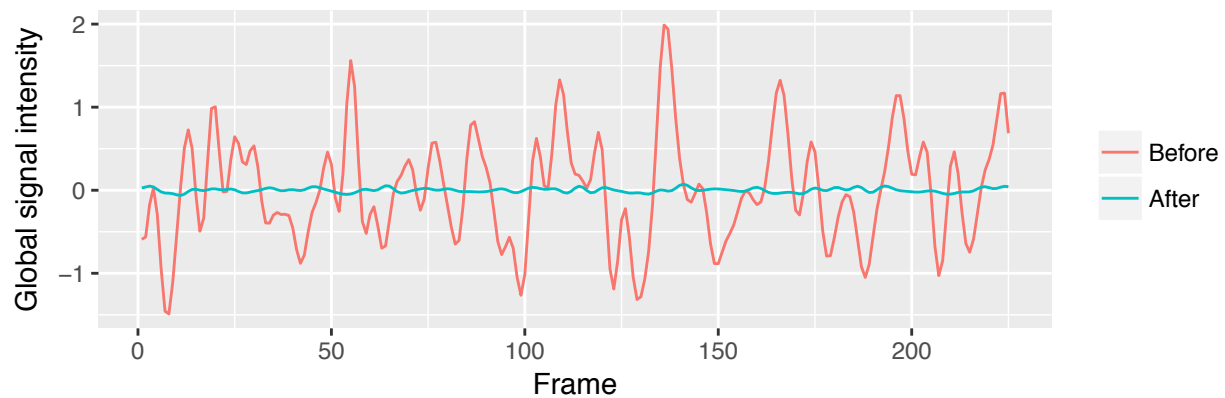
# Plot the global signal. Do we regress out the global signal? Still an open issue.
# Let's just explore the approach to regressing it out afterwards. A better way would
# be to include it as an ``initialNuisanceVariable`` in ``preprocessRestingBOLD()``.

boldMatrix <- timeseries2matrix(
  preprocessedRestingState$cleanBoldImage, restingStateBoldMaskImage )
boldMatrixGlobalSignalRegressedOut <-
  residuals( lm( boldMatrix ~ scale( preprocessedRestingState$globalSignal ) ) )

globalSignalDataFrame <- data.frame( Frame = rep( 1:numberOfTimeFrames, 2 ),
  GlobalSignal = c( preprocessedRestingState$globalSignal,
    apply( boldMatrixGlobalSignalRegressedOut, mean, MARGIN = 1 ) ),
  Type = factor( c( rep( "Before", numberOfTimeFrames ),
    rep( "After", numberOfTimeFrames ) ), levels = c( "Before", "After" ) ) )

ggplot( globalSignalDataFrame ) +
  geom_line( aes( x = Frame, y = GlobalSignal, colour = Type ), size = 0.5 ) +
  xlab( "Frame" ) + ylab( "Global signal intensity" ) +
  theme( legend.title = element_blank() ) + theme( aspect.ratio=1/3 )

```



References

1. Behzadi, Y., Restom, K., Liau, J., and Liu, T. T. “**A Component Based Noise Correction Method (CompCor) for BOLD and Perfusion Based FMRI**” *Neuroimage* 37, no. 1 (2007): 90–101. doi:10.1016/j.neuroimage.2007.04.042
2. Power, J. D., Barnes, K. A., Snyder, A. Z., Schlaggar, B. L., and Petersen, S. E. “**Spurious but Systematic Correlations in Functional Connectivity MRI Networks Arise from Subject Motion**” *Neuroimage* 59, no. 3 (2012): 2142–54. doi:10.1016/j.neuroimage.2011.10.018
3. Power, J. D., Mitra, A., Laumann, T. O., Snyder, A. Z., Schlaggar, B. L., and Petersen, S. E. “**Methods to Detect, Characterize, and Remove Motion Artifact in Resting State FMRI**” *Neuroimage* 84, (2014): 320–41. doi:10.1016/j.neuroimage.2013.08.048
4. Liu, T. T., Nalci, A., and Falahpour, M. “**The Global Signal in FMRI: Nuisance or Information?**” *Neuroimage* 150, (2017): 213–229. doi:10.1016/j.neuroimage.2017.02.036