Sparse Canononical Correlation Analysis for Neuroimaging (SCCAN): A Tutorial

Introduction

Overview

Sparse canonical correlation analysis for neuroimaging (SCCAN) is a general purpose tool for "two-sided" multiple regression. It is an extension of Hotelling's seminal canonical correlation analysis [1, 2] which itself is a multi-modal extension of principal component analysis (PCA). This technique allows one to symmetrically compare one matrix of data to another and find linear relationships between them in a low-dimensional space. SCCAN derives from classic canonical correlation analysis and also relates to singular value decomposition. To handle data with $p >> n$ (common in medical imaging scenarios) SCCAN uses high-dimensional regularization methods common in ℓ_1 regression and spatial regularization to help ensure the biological plausibility of statistical maps in medical imaging (often referred to as *eigenanatomy*). This problem is a difficult optimization (NP-hard) and, to improve solution interpretability and stability, SCCAN allows one to to use prior knowledge to constrain the solution space.

Another view (perhaps simpler and broader) is that eigenanatomy is a general framework for reducing the dimensionality of imaging data into interpretable pieces. With mass-univariate testing Eigenanatomy is motivated by two ideas:

Voxels that change together should hang together.

and

Clustering before hypothesis testing, not hypothesis testing then clustering.

These two strategies conserve statistical power in a controllable manner (contrasting with mass univariate techniques) by reducing the number of statistical tests performed.

Comparison with other techniques

Common mass univariate techniques (e.g., voxel-based morphometry [3]) require adjustment for multiple comparisons and subsequent ad hoc clustering of significant voxels into potentially anatomically meaningful clusters [4]. Decomposition techniques, such as SCCAN, invert this statistical directionality by first "decomposing", or "clustering", voxels in an anatomically constrained, yet data-driven, manner followed by significance testing. Repeating from the previous section, this prioritizing of the dimensionality reduction step mitigates the multiple comparison issue.

Traditional decomposition methods, such as the orthogonality-constrained PCA and independent components analysis (ICA), have found widespread utiltity in neuroimaging (e.g., PCA: [5, 6] and ICA: [7–9]). However, without additional constraints, such solutions are not "sparse" in the sense that the solution space is non-zero over the entire problem domain (e.g., PCA- and ICA-derived eigenvectors) and can produce negative weights which limits biological interpretability [10].

One potentially problematic issue with sparse matrix decompositions, such as SCCAN, is the potential collinearity of the "eigenvectors" (or, more accurately, the "pseudo-eigenvectors" since they do not necessarily satisfy orthogonality) resulting from optimization of the eigenvalue problem modified by constraints (e.g., positivity and sparsity) [10, 11].

Simple example

```
# Disclaimer:  This is an example I (Nick) put together to conceptually illustrate
# a subset of the motivating principles behind SCCAN as I understand them (which,
# admittedly, might not be very well).  I need verify with Brian so take it with
# a grain of salt for now.
#
# With mass univariate testing, there are certain obvious things that one does to
# reduce the number of statistical tests performed.  For example, we limit testing
# to masked regions of interest.  Obviously, we completely ignore image values
# in the skull if we are only interested in white matter differences and one of the
# reasons why we develop processing tools to get good regional masks is precisely
# to reduce the number of tests performed.  Decomposition methods, such as SCCAN,
# while also using regional masks, try to reduce the number of tests even further
# by finding those regions which actually demonstrate variance.  And we do this before
# statistical testing for purposes of increased statistical power.  If an area
# (e.g., fornix) doesn't actually demonstrate (e.g., FA) intensity variation,
# why should we waste statistical power by testing that area?
#
# In addition, we can even further reduce statistical testing by combining regions
# which happen to be correlated in their variation which can be done with decomposition
# techniques such as PCA.  And this is what is demonstrated in the example below.
# Suppose we have two regions, which could be voxels or anatomical regions (e.g.,
# Fornix and Splenium), and the intensity (e.g., FA) differences between two groups
# ("experimental" and "control") are correlated.  With a univariate paradigm, we
# would perform two statistical t-tests and then correct for multiple comparisons.
# This could lead to non-significance for one or both regions which would mask
# what is actually occurring.  Instead, we perform PCA which reveals that the two
# regions are correlated.  We then perform testing in this decomposed space which
# shows significance and we don't have to perform multiple comparisons correction!
#
# Because of the simplicity of this example, we omit some key concepts associated
# with SCCAN such as sparsity, clustering, and the multi-modal CCA aspects of SCCAN.
# However, those concepts will be explored in the actual imaging examples below.

library( ggplot2 )
library( MASS )
library( pander )

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/" );

# We first generate sample data for two fictional regions where we're looking
# at tissue differences between a control group and an experimental group.
# These differences could be something like gray matter desnity or fractional
```

```
# anisotropy.  We design the data so that it is correlated (as many real data
# are) between the two regions.

set.seed( 123 )

numberOfControls <- 30
numberOfExperimentals <- 35

rotationMatrix <- matrix( c( 0.707, 0.4, 0.707, 0.707 ), nrow = 2 )

controlSigma <- matrix( 0.001 * c( 1, 0.05, 0.05, 1 ), nrow = 2 ) %*% rotationMatrix
controlValues <- mvrnorm( numberOfControls, c( 0.6, 0.6 ), controlSigma )

experimentalSigma <- matrix( 0.001 * c( 1, 0.05, 0.05, 1 ), nrow = 2 ) %*% rotationMatrix
experimentalValues <- mvrnorm( numberOfExperimentals, c( 0.585, 0.585 ), experimentalSigma )

studyDataFrame <- data.frame(
  Group = factor( c( rep( 'Control', numberOfControls ),
                     rep( 'Experimental', numberOfExperimentals ) ) ),
  Region1 = c( controlValues[, 2], experimentalValues[, 2] ),
  Region2 = c( controlValues[, 1], experimentalValues[, 1] ) )

# Perform PCA on the study data using Region1 and Region2 columns and then
# rotate the canonical coordinates based on the PCA decomposition.  We want
# to use this to display and show the directions of variance in the sample data.

studyPca <- prcomp( studyDataFrame[, 2:3], center = TRUE, scale = TRUE )
rotatedCoordinates <- studyPca$rotation %*% matrix( c( 1, 0, 0, 1 ), nrow = 2 )

xlimits <- c( 0.55, 0.66 )
ylimits <- c( 0.55, 0.66 )

arrowPosition <- data.frame( Group = c( "Control", "Control" ),
                             x = rep( mean( xlimits ), 2 ),
                             y = rep( mean( ylimits ), 2 ),
                             xrot = rotatedCoordinates[1, ],
                             yrot = rotatedCoordinates[2, ]
                           )
arrowPosition$xend <- arrowPosition$x +
  0.4 * arrowPosition$xrot * ( xlimits[2] - xlimits[1] ) * ( studyPca$sdev )^2
arrowPosition$yend <- arrowPosition$y +
  0.4 * arrowPosition$yrot * ( ylimits[2] - ylimits[1] ) * ( studyPca$sdev )^2

# Plot the fictitious data

studyPlot <- ggplot( studyDataFrame,
   aes( x = Region1, y = Region2, colour = Group ) ) +
   geom_segment( data = arrowPosition, aes( x = x, y = y, xend = xend, yend = yend ),
                 colour = "black", arrow = arrow(), size = 0.5, linetype = "dashed" ) +
   geom_point( size = 5, alpha = 0.8 ) +
   xlab( "Region 1" ) + ylab( "Region 2" ) +
   xlim( xlimits ) + ylim( ylimits )
```

```
studyPlotFile <- paste0( figuresDirectory, 'simpleExamplePlot.pdf' )
ggsave( filename = studyPlotFile, plot = studyPlot, width = 5.75, height = 4 )
```

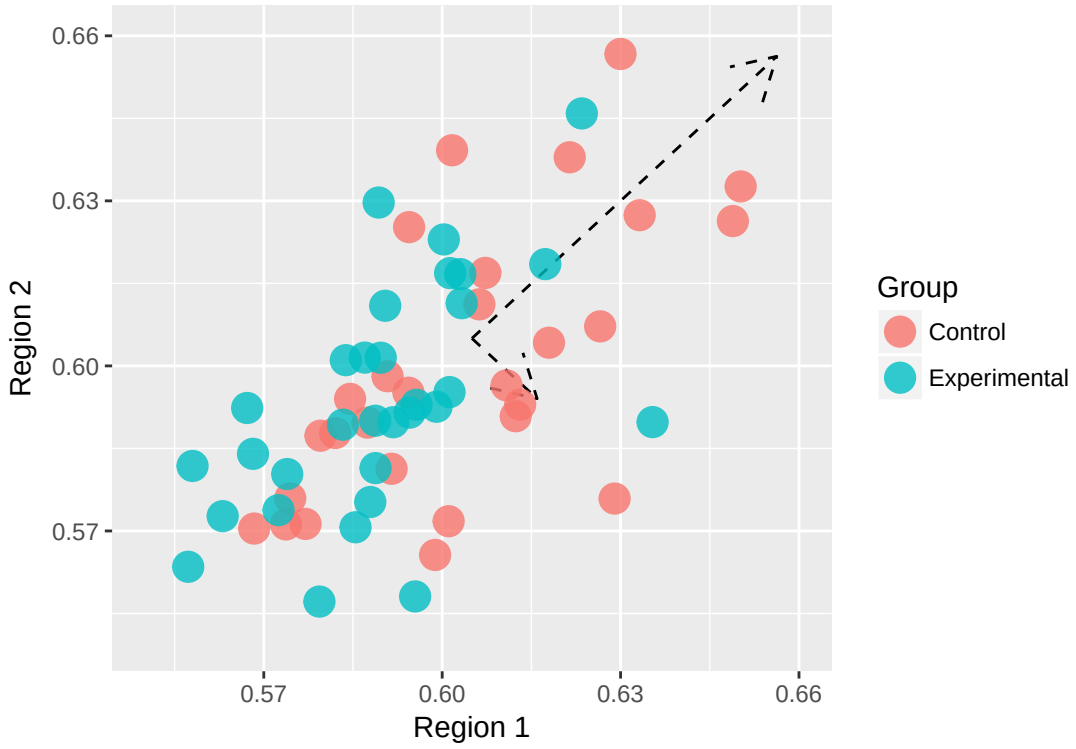


Figure 1: Fictitious data showing regional brain values (e.g., gray matter or FA) between two groups. Note that the data is correlated between the two regions and we show the principal components of variance using PCA decomposition and illustrate this difference with the plotted arrows.

```
# Since PCA tells us that we can decompose the data such that a single eigenvector
# explains ~80% of the variance in the data (cf command:  summary( studyPca ) ), we
# transform that data and do a single test (instead of two) along the principal
# variance direction.

studyDataFrame$TransformedData <-
  data.matrix( studyDataFrame[, 2:3] ) %*% studyPca$rotation[, 1]
studyTtest <- t.test( TransformedData ~ Group, data = studyDataFrame )

# We also perform independent testing of the two regions and adjust accordingly.

studyTtest1 <- t.test( Region1 ~ Group, data = studyDataFrame )
studyTtest2 <- t.test( Region2 ~ Group, data = studyDataFrame )

qvalues <- p.adjust( c( studyTtest1$p.value, studyTtest2$p.value ), method = "holm" )

studyReportDataFrame <- data.frame(
  TestNames = c( "Region 1", "Region 2", "PCA Transformed" ),
  Pvalues = c( studyTtest1$p.value, studyTtest2$p.value, studyTtest$p.value ),
  AdjustedPvalues = c( qvalues, NA ) )
```

```
colnames( studyReportDataFrame ) <- c( "Regions", "p-values", "Adjusted p-values" )

pander( studyReportDataFrame, style = "rmarkdown", caption = "Statistical testing of
the two regions as well as the PCA transformation of the data which takes into account
the correlated nature of both regions.  Note that the traditional univariate paradigm
results in the non-significance of Region 2 whereas testing the PCA transformed
data maintains the signicance of both regions." )
```

Table 1: Statistical testing of the two regions as well as the PCA transformation of the data which takes into account the correlated nature of both regions. Note that the traditional univariate paradigm results in the non-significance of Region 2 whereas testing the PCA transformed data maintains the signicance of both regions.

Regions	p-values	Adjusted p-values
Region 1	0.00946	0.01892
Region 2	0.262	0.262
PCA Transformed	0.04262	NA

Applications

SCCAN-related methods have been applied in the following papers:

- *Dementia induces correlated reductions in white matter integrity and cortical thickness: a multivariate neuroimaging study with sparse canonical correlation analysis* [12].
- *Methodological considerations in longitudinal morphometry of traumatic brain injury* [13].
- *Sparse canonical correlation analysis relates network-level atrophy to multivariate cognitive measures in a neurodegenerative population* [11]. Repeatable cortical structural networks associated with specific pyschometric testing are determined from SCCAN by determining maximal correlations between cognitive measurements from the Philadelphia Brief Assessment of Cognition (e.g., apathy, agitation, and social comportment) and gray matter density determined from T1-weighted MRI. *The tutorial in the next section is based on this work.*
- *Eigenanatomy: Sparse dimensionality reduction for multi-modal medical image analysis* [10]. A comparison is performed using decomposition techniques (ICA, PCA, and SCCAN) on multi-modality neuroimaging data (cortical thickness from T1, cerebral blood flow from ASL, and fractional anisotropy from diffusion-weighted MRI) from a publicly available cohort. SCCAN (i.e., *eigenanatomy*) outperforms the other methods in predicting subject age. Data and analysis scripts for this work are publicly available.
- *Subject-specific functional connectivity parcellation via Prior Based Eigenanatomy* [14]. Sparse decomposition is performed on individual subject fMRI data (the corresponding image data matrix is $t \times p$ where t is the number of time points and p is the number of voxels in the user-specified mask. A spatial (i.e., anatomical) prior is accommodated by the eigenanatomy framework and is used to describe the default mode network–hippocampus functional connectivity. These connectivity patterns are then used to classify mild cognitive impairment subjects from cognitively normal subjects.
- *White matter imaging helps dissociate tau from TDP-43 in frontotemporal lobar degeneration* [15].
- *The power of neuroimaging biomarkers for screening frontotemporal dementia* [16].
- *Genetic and neuroanatomic associations in sporadic frontotemporal lobar degeneration* [17].

ANTsR implementation

The various SCCAN-related methods and associated functionality are all available in the ANTsR package[1]. The two main R functions are:

- `sparseDecom` — Decomposes a matrix into sparse eigenevectors to maximize explained variance.
- `sparseDecom2` — Decomposes two matrices into paired sparse eigenevectors to maximize canonical correlation.

Auxiliary functions include:

- `initializeEigenanatomy`
- `eigSegs`
- `joinEigenanatomy`

[1] https://github.com/stnava/ANTsR

Tutorial

Below is a series of tutorials based on the work presented in [11] described above where structural networks are determined based on gray matter density computed from T1-weighted MRI and neurocognitive testing using the Philadelphia Brief Assessment of Cognition (PBAC).

Initialization

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

library( knitr )
library( visreg )
library( pheatmap )
library( pander )
library( png )
library( misc3d )
library( rgl )
library( pixmap )
library( randomForest )
library( ggplot2 )
library( stargazer )
invisible( suppressMessages( library( ANTsR ) ) )
```

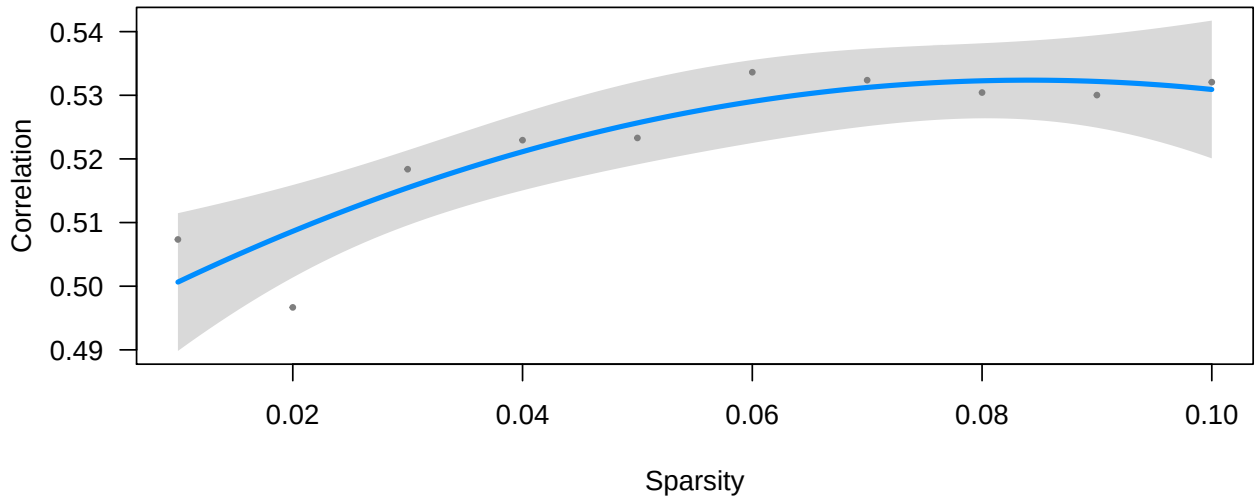
Read input data

```
# Load the AAL (Automated Anatomical Labeling) data table from ANTsR as well as
# the AAL label image.

data( aal, package = 'ANTsR' )
aalLabelTable <- aal
aalFileName <- paste0( dataDirectory, "aal.nii.gz" )
aalImage <- antsImageRead( filename = aalFileName, dimension = 3 )

# For convenience, the data set for test/train subjects are stored in 2-D images where
# the rows correspond to different subjects and the columns are the voxels within
# the 3-D template gray matter mask.

trainingFile <- paste0( dataDirectory, "pbac_train_img.mha" )
trainingImageData <- as.matrix( antsImageRead( filename = trainingFile, dimension = 2 ) )

testingFile <- paste0( dataDirectory, "pbac_test_img.mha" )
testingImageData <- as.matrix( antsImageRead( filename = testingFile, dimension = 2 ) )

grayMatterMaskFile <- paste0( dataDirectory, "mask.nii.gz" )
grayMatterMask <- antsImageRead( filename = grayMatterMaskFile, dimension = 3 )

# Read in the corresponding cognitive data for the test/train subjects.
```

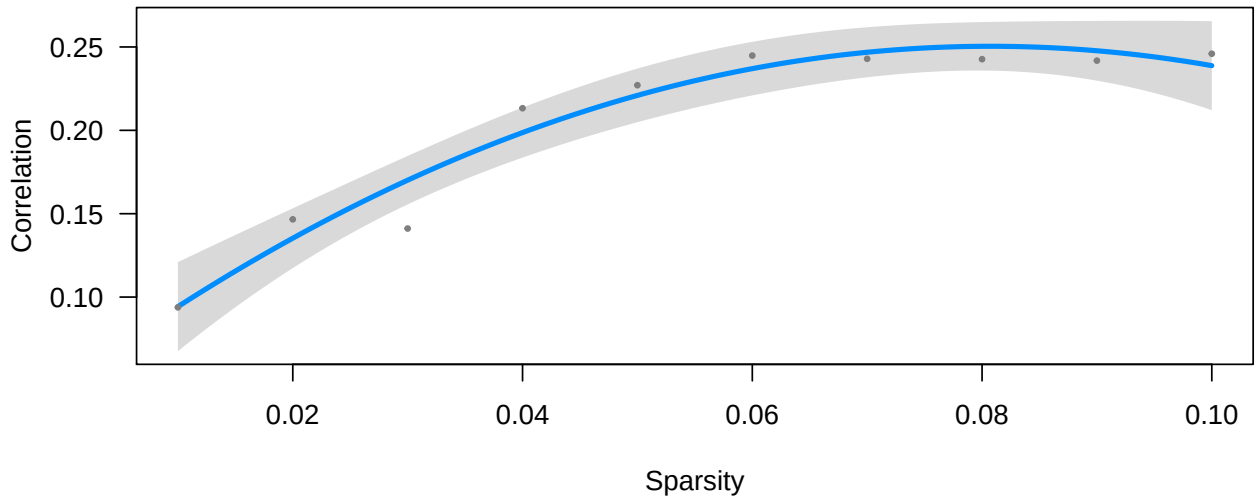
```
trainingCognitiveData <- read.csv( paste0( dataDirectory, "pbac_train_cog.csv" ) )
testingCognitiveData <- read.csv( paste0( dataDirectory, "pbac_test_cog.csv" ) )
```

SCCAN for sparse regression

```
# Here we use SCCAN to find brain regions relating to age.  In this case, SCCAN
# is used as a sparse regression utility.  We impose a "cluster threshold"
# regularization to prevent isolated voxels from appearing in the solution.
# Evaluation as a function of sparseness is performed using the testing data.
# This type of approach can be useful in parameter selection i.e., choosing the
# optimization criterion based on the training data.

trainingAgeMatrix <- matrix( trainingCognitiveData$age, ncol = 1 )
sparsityValues <- seq( from = 0.01, to = 0.1, length = 10 )
trainingAgeCorrelations <- rep( 0, length( sparsityValues ) )
testingAgeCorrelations <- rep( 0, length( sparsityValues ) )

for( i in 1:length( sparsityValues ) )
  {
  ageSccanResults <- sparseDecom2(
    inmatrix = list( scale( trainingImageData ), scale( trainingAgeMatrix ) ),
    sparseness = c( sparsityValues[i], 0.9 ), inmask = c( grayMatterMask, NA ),
    nvecs = 2, mycoption = 0, cthresh = c( 1000, 0 ), ell1 = 10, smooth = 0.5,
    verbose = 0 )

  # ageSccanResults$eig1 contain the eigenvectors for "view 1" (in CCA terminology)
  # whereas ageSccanResults$eig2 contain the eigenvectors for "view 2".  In this
  # scenario only the first view (imaging data) is relevant.  One can view the
  # eigenvectors as images using the ANTsR::matrixToImages() function, e.g.,
  #    sccanFirstEigenImage <- matrixToImages( t( ageSccanResults$eig1 ),
  #          grayMatterMask )[[1]]

  # determine correlation with training data

  trainingAgePredictors <- trainingImageData %*% ageSccanResults$eig1
  trainingAgeCorrelations[i] <- cor( trainingAgePredictors, trainingCognitiveData$age )

  # validate using testing data

  testingAgePredictors <- testingImageData %*% ageSccanResults$eig1
  testingAgeCorrelations[i] <- cor( testingAgePredictors, testingCognitiveData$age )
  }
sparseRegressionDataFrame <- data.frame(
                  sparsityValues = sparsityValues,
                  trainingAgeCorrelations = trainingAgeCorrelations,
                  testingAgeCorrelations = testingAgeCorrelations )

trainingLm <- lm( trainingAgeCorrelations ~ sparsityValues + I( sparsityValues^2 ) )
visreg( trainingLm, xlab = "Sparsity", ylab = "Correlation" )
```




```
testingLm <- lm( testingAgeCorrelations ~ sparsityValues + I( sparsityValues^2 ) )
visreg( testingLm, xlab = "Sparsity", ylab = "Correlation" )
```



SCCAN with prior initialization

```
# Here we use the same data as in the previous example but use SCCAN to find brain
# regions relating to the battery of tests that measure language-related cognitive
# function.  Due to the language association, we initialize SCCAN with left
# hemispheric regions.  In this case, the initialization controls the sparseness
# parameters for each eigenvector.  Thus, the sparsity parameter will be overridden
# by the priors, thereby enabling a "per eigenvector" sparsity value.

languageBatteryTypes <- c( "Speech", "Writing", "Semantic", "Reading", "Naming" )

trainingCognitiveMatrix <- cbind( trainingCognitiveData$speech_adj,
                                  trainingCognitiveData$writing_adj,
                                  trainingCognitiveData$semantic_adj,
                                  trainingCognitiveData$reading_adj,
                                  trainingCognitiveData$naming_adj )
colnames( trainingCognitiveMatrix ) <- languageBatteryTypes

testingCognitiveMatrix <- cbind( testingCognitiveData$speech_adj,
                                 testingCognitiveData$writing_adj,
                                 testingCognitiveData$semantic_adj,
                                 testingCognitiveData$reading_adj,
                                 testingCognitiveData$naming_adj )
colnames( testingCognitiveMatrix ) <- languageBatteryTypes

# Initialize the left hemispheric AAL regional priors associated with the language
# centers.  More information re. AAL can be found at various neuroimaging sites, e.g.,
#    http://neuro.imm.dtu.dk/wiki/Automated_Anatomical_Labeling
# The specific regions of interest for the testing below are:
#      Label                    Region
#       13               Left area triangularis
#       81               Left superior temporal gyrus
#       39               Left parahippocampal gyrus
#       79               Left transverse temporal gyri

aalLanguageRegionalLabels <- c( 13, 81, 39, 79 )

# Create a label matrix ( size = numberOfCognitiveLabels x numberOfVoxelsInMask )
# to be used as a spatial prior

numberOfCognitiveLabels <- length( aalLanguageRegionalLabels )
numberOfVoxelsInMask <- sum( grayMatterMask > 0.5 )

aalLanguageRegionalLabelMatrix <- matrix(
  rep( 0, numberOfVoxelsInMask * numberOfCognitiveLabels ),
  nrow = numberOfCognitiveLabels )

for( i in 1:numberOfCognitiveLabels )
  {
  perLabelVector <- aalImage[grayMatterMask == 1] == aalLanguageRegionalLabels[i]
  aalLanguageRegionalLabelMatrix[i,] <- as.numeric( perLabelVector )
  }
eigenInitialization <- initializeEigenanatomy( aalLanguageRegionalLabelMatrix,
```

```
                        grayMatterMask )

priorWeights <- c( 0.9, 0.5, 0.05 )

# Create 3-D brain volumetric rendering of the initial AAL language labels
#   and create some variables for rendering the results.

languagePriorWeightsSccanPlotFiles <- rep( '', length( priorWeights ) )
languagePriorWeightsSccanPlot3DFiles <- rep( '', length( priorWeights ) )
cognitiveSccanColors <- c( "red", "green", "blue", "yellow" )

brain <- renderSurfaceFunction( surfimg = list( aalImage ), alphasurf = 0.1,
    funcimg = eigenInitialization$initlist, smoothsval = 1.5, smoothfval = 0,
    mycol = cognitiveSccanColors )

id <- par3d( "userMatrix" )
rid <- rotate3d( id, -pi / 2, 1, 0, 0 )
rid2 <- rotate3d( id, pi / 2, 0, 0, 1 )
par3d( userMatrix = id )
languageInitializationSccanPlot3DFile <- paste0( figuresDirectory,
  "cognitiveSccanEigenImages3D_Initialization"  )
dd <- make3ViewPNG( rid, id, rid2, paste0( figuresDirectory,
  "cognitiveSccanEigenImages3D_Initialization" ) )
languageInitializationSccanPlot3DFile <- paste0( figuresDirectory,
  "cognitiveSccanEigenImages3D_Initialization", ".png"  )
par3d( userMatrix = id )

for( i in 1:length( priorWeights ) )
  {
  cognitiveSccanResult <- sparseDecom2(
       inmatrix = list( scale( trainingImageData ), scale( trainingCognitiveMatrix ) ),
       its = 20, mycoption = 0, sparseness = c( 0, -0.5 ), nvecs = numberOfCognitiveLabels,
       inmask = c( grayMatterMask, NA ), cthresh = c( 1000, 0 ), verbose = 0, ell1 = 10,
       initializationList = eigenInitialization$initlist, priorWeight = priorWeights[i],
       smooth = 0, perms = 0 )

  # calculate the predictors for both (imaging and cognitive testing) views

  trainingImagePredictors <- trainingImageData %*% cognitiveSccanResult$eig1
  colnames( trainingImagePredictors ) <- paste0( "GM", c( 1:numberOfCognitiveLabels ) )
  trainingCognitivePredictors <- trainingCognitiveMatrix %*% cognitiveSccanResult$eig2
  bestPredictor <- which.max(
    abs( diag( cor( trainingImagePredictors, trainingCognitivePredictors ) ) ) )

  trainingDataFrame <- data.frame( trainingImagePredictors, trainingCognitivePredictors )

  # Set up a linear model with the training data to calculate cognitive predictors
  # with the testing data using the best predictor.

  lmFormula <- as.formula( paste0( "Variate00" ,bestPredictor-1 , "~ GM1+GM2+GM3+GM4" ) )
  trainingLm <- lm( lmFormula, data = trainingDataFrame )

  testingImagePredictors <- testingImageData %*% cognitiveSccanResult$eig1
```

```
  colnames( testingImagePredictors ) <- paste0( "GM", c( 1:numberOfCognitiveLabels ) )
  testingCognitivePredictors <- testingCognitiveMatrix %*% cognitiveSccanResult$eig2
  testingDataFrame <- data.frame( testingImagePredictors, testingCognitivePredictors )

  # How well does the gray matter image view predict language-cognitive ability?
  # We don't print this test but report the correlation and p-values directly in
  # caption for the corresponding 3-D brain renderings produced below.

  testingCorrelationTest <- cor.test( testingDataFrame[, bestPredictor],
          predict( trainingLm, newdata = testingDataFrame ) )

  cognitiveSccanImages <- matrixToImages( t( cognitiveSccanResult$eig1 ), grayMatterMask )
  for( image in cognitiveSccanImages )
    {
    image[grayMatterMask == 1] <- abs( image[grayMatterMask == 1] )
    image[grayMatterMask == 1] <- image[grayMatterMask == 1] /
       max( image[grayMatterMask == 1] )
    }

  # Create 2-D slice mosaics in sagittal view.  We don't render them in the
  # pdf document but write them to disk for perusal and to illustrate use
  # case.

  languagePriorWeightsSccanPlotFiles[i] <- paste0( figuresDirectory,
    "cognitiveSccanEigenImages_PriorWeight", priorWeights[i], ".jpg" )
  plot( grayMatterMask, cognitiveSccanImages, color.overlay = cognitiveSccanColors,
        axis = 1, nslices = 20, outname = languagePriorWeightsSccanPlotFiles[i] )

  # Create 3-D brain volumetric rendering.  These are rendered in Figures 1-4.

  brain <- renderSurfaceFunction( surfimg = list( aalImage ), alphasurf = 0.1,
      funcimg = cognitiveSccanImages, smoothsval = 1.5, smoothfval = 0,
      mycol = cognitiveSccanColors )

  id <- par3d( "userMatrix" )
  rid <- rotate3d( id, -pi / 2, 1, 0, 0 )
  rid2 <- rotate3d( id, pi / 2, 0, 0, 1 )
  par3d( userMatrix = id )
  languagePriorWeightsSccanPlot3DFiles[i] <- paste0( figuresDirectory,
    "cognitiveSccanEigenImages3D_PriorWeight", priorWeights[i]  )
  dd <- make3ViewPNG( rid, id, rid2, languagePriorWeightsSccanPlot3DFiles[i] )
  languagePriorWeightsSccanPlot3DFiles[i] <- paste0( figuresDirectory,
    "cognitiveSccanEigenImages3D_PriorWeight", priorWeights[i], ".png"  )
  par3d( userMatrix = id )
  }

# Figures 1 through 4 show that the best results initialized by the prior can
# drift away from that initialization.  A fundamental question is---Where in the
# brain do the solution vectors end up?  We write a quick function to answer this
# question for the weak prior case (priorWeighting = 0.1).

reportAnatomy <- function( eigenImageList, mask, weight = 0.3 )
  {
```

```
  sccanAalLabels <- c()
  for( eigenImage in eigenImageList )
    {
    nonZeroIndices<- abs( eigenImage[mask == 1] ) > 0
    sccanAalLabels <- append( sccanAalLabels, aalImage[mask == 1][nonZeroIndices] )
    }
  anatomicalCount <- hist( sccanAalLabels, breaks = 0:100, plot = FALSE )$count
  anatomicalCount[anatomicalCount < weight * max( anatomicalCount )] <- 0
  aalIndices <- which( anatomicalCount > 0 )

  return( aalLabelTable$label_name[aalIndices] )
  }

reportedAnatomy <- reportAnatomy( cognitiveSccanImages, grayMatterMask )
```

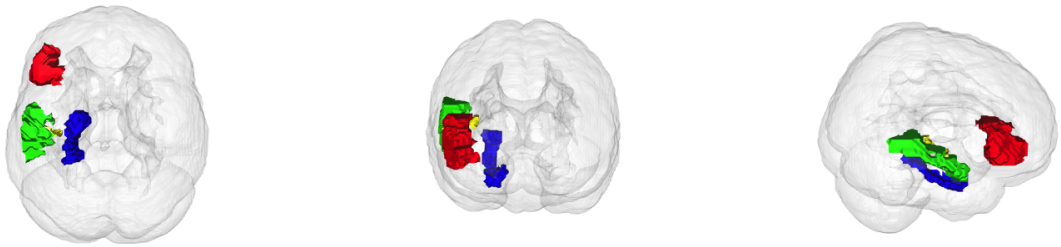


Figure 2: 3-D renderings of the initial language regions from the AAL image. Regional labels are left area triangularis (red), left superior temporal gyrus (green), left parahippocampal gyrus (yellow), left transverse temporal gyri (blue).

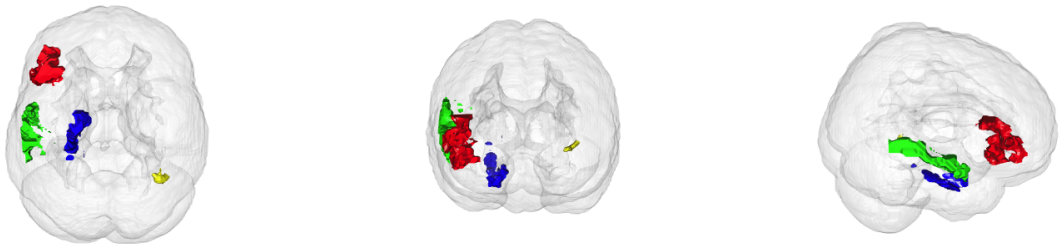


Figure 3: 3-D renderings of the eigenvectors for the imaging (gray matter) view constructed from a strong prior (= 0.9). Pearson's product-moment correlation results in a correlation value of 0.4 (p-value = 0.0002)

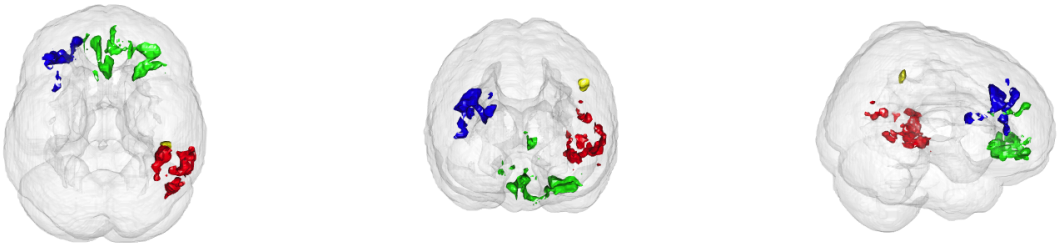


Figure 4: 3-D renderings of the eigenvectors for the imaging (gray matter) view constructed from a medium prior (= 0.5). Pearson's product-moment correlation results in a correlation value of 0.65 (p-value = 2e-11)

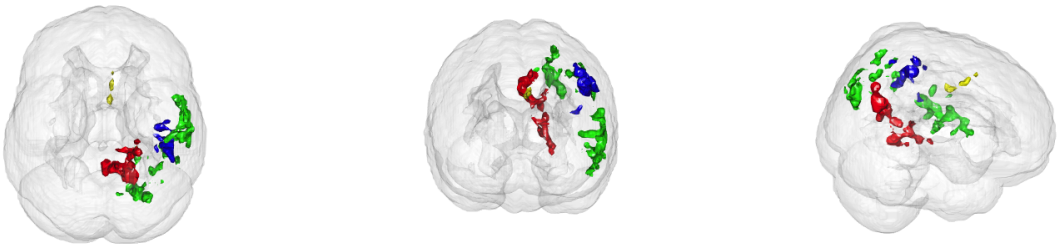


Figure 5: 3-D renderings of the eigenvectors for the imaging (gray matter) view constructed from a weak prior (= 0.1). Pearson's product-moment correlation results in a correlation value of 0.4 (p-value = 0.0001)

```
reportedAnatomyDataFrame <- data.frame( newRegions = reportedAnatomy )
colnames( reportedAnatomyDataFrame ) <- c( 'Predictor regions' )
pander( reportedAnatomyDataFrame, style = "rmarkdown", caption = "Using a weak prior
  (= 0.1), the solution migrates from the initialized regions.  The new regions
  associated with the best \"language\" predictor are provided below." )
```

Table 2: Using a weak prior (= 0.1), the solution migrates from the initialized regions. The new regions associated with the best "language" predictor are provided below.

Predictor regions
Calcarine_R
Occipital_Sup_R
Occipital_Mid_R
Parietal_Inf_R
SupraMarginal_R
Angular_R
Precuneus_R
Temporal_Sup_R
Temporal_Mid_R

```
# How good were our original hypothetical regions as predictors? -- Good question.
```

A closer look at the SCCAN eigenvectors

```
# Recalling that CCA maximizes PearsonCorrelation$(XW^T$,ZY^T)$, where $X$ and $Z$ are
# data matrices, we can study the eigenvector matrix $Y$ (or $W$) which contrasts or
# combines columns of the associated data matrix. In this example, we look at $Y$
# (prior weighting = 0.1) which operates on the language-related cognition/design matrix.

# Technical note:  In order to break this tutorial into reasonably sized chunks and take
# advantage of caching, we need to re-run SCCAN with the final parameters (prior weighting
# = 0.1) from the previous chunk.  ANTsR stores an external pointer to the image which is
# not preserved between document compilations ("stitching").

eigenInitialization <- initializeEigenanatomy( aalLanguageRegionalLabelMatrix,
                        grayMatterMask )

cognitiveSccanResult <- sparseDecom2(
     inmatrix = list( scale( trainingImageData ), scale( trainingCognitiveMatrix ) ),
     its = 20, mycoption = 0, sparseness = c( 0, -0.5 ), nvecs = numberOfCognitiveLabels,
     inmask = c( grayMatterMask, NA ), cthresh = c( 1000, 0 ), verbose = 0, ell1 = 10,
     initializationList = eigenInitialization$initlist, priorWeight = 0.1,
     smooth = 0, perms = 0 )

cognitiveSccanHeatMapFile <- paste0( figuresDirectory,
  "cognitiveSccanHeatMap_PriorWeight0.1.png"  )

rownames( cognitiveSccanResult$eig2 ) <- languageBatteryTypes
```

```
pheatmap( cognitiveSccanResult$eig2, cluster_rows = TRUE, cluster_cols = TRUE,
  filename = cognitiveSccanHeatMapFile )
```

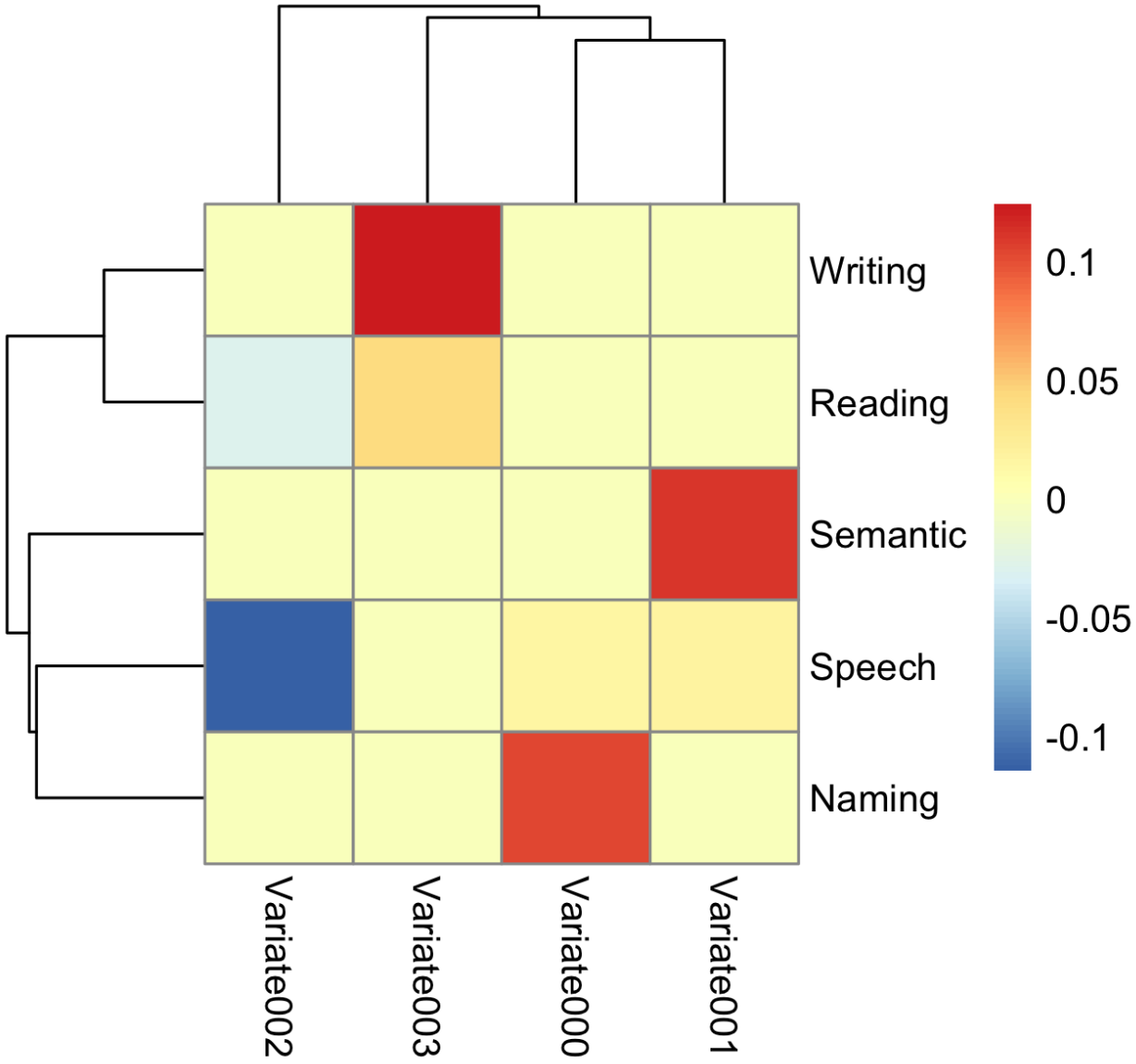


Figure 6: Heat map of the eigenvectors for the language/cognitive view constructed from a weak prior (= 0.1).

SCCAN regression with nuisance variables

```
# One often wants to control for the presence of nuisance variables in conjunction
# with SCCAN results.  There are several options including:
#     (1) control after you do dimensionality reduction,
#     (2) orthogonalize the predictors before decomposition, and
```

```
#     (3) use alternative SCCAN formulations (e.g. set ``mycoption`` to 0 or 2).
# We first try the options (1) and (2) as they are more traditional.

# Option (1)---control for nuisance variables (e.g., age and mmse) after dimensionality
# reduction.

eigenInitialization <- initializeEigenanatomy( aalLanguageRegionalLabelMatrix,
                        grayMatterMask )

cognitiveSccanResult <- sparseDecom2(
     inmatrix = list( scale( trainingImageData ), scale( trainingCognitiveMatrix ) ),
     its = 20, mycoption = 0, sparseness = c( 0, -0.9 ), nvecs = numberOfCognitiveLabels,
     inmask = c( grayMatterMask, NA ), cthresh = c( 1000, 0 ), verbose = 0, ell1 = 10,
     initializationList = eigenInitialization$initlist, priorWeight = 0.1,
     smooth = 0, perms = 0 )

trainingImagePredictors <- trainingImageData %*% cognitiveSccanResult$eig1
colnames( trainingImagePredictors ) <- paste0( "GM", c( 1:numberOfCognitiveLabels ) )
trainingCognitivePredictors <- trainingCognitiveMatrix %*% cognitiveSccanResult$eig2
bestPredictor <- which.max(
  abs( diag( cor( trainingImagePredictors, trainingCognitivePredictors ) ) ) )

# Set up a linear model with the training data to get the cognitive predictors
# with the testing data using the best predictor (as before) but this time, add
# mmse and age as covariates.

trainingDataFrame <- data.frame( trainingImagePredictors, trainingCognitivePredictors,
   MMSE = trainingCognitiveData$mmse, Age = trainingCognitiveData$age )
lmFormula <- as.formula( paste0( "Variate00" ,bestPredictor-1 ,
   "~ GM1+GM2+GM3+GM4+Age+MMSE" ) )
trainingLm <- lm( lmFormula, data = trainingDataFrame )

testingImagePredictors <- testingImageData %*% cognitiveSccanResult$eig1
colnames( testingImagePredictors ) <- paste0( "GM", c( 1:numberOfCognitiveLabels ) )
testingCognitivePredictors <- testingCognitiveMatrix %*% cognitiveSccanResult$eig2
testingDataFrame <- data.frame( testingImagePredictors, testingCognitivePredictors,
  MMSE = testingCognitiveData$mmse, Age = testingCognitiveData$age )
testingLm <- lm( lmFormula, data = testingDataFrame )
```

```
# We output the two linear models in Table 2 using the stargazer package.

stargazer( trainingLm, testingLm, model.names = FALSE, header = FALSE,
  no.space = TRUE, ci = TRUE, ci.level = 0.90, single.row = TRUE, type = 'latex',
  table.placement = 'h', dep.var.caption = "", dep.var.labels.include = FALSE,
  column.labels = c( "{\\bf Training}", "{\\bf Testing}" ), model.numbers = FALSE,
  title = "Summary of training and testing linear models with nuisance variables
   included after decomposition (option 1).  90\\% confidence intervals for the
   covariates are given in parentheses." )
```

```
# Now we try option (2)---orthogonalize the predictors against MMSE and age.  In
# other words, we regress out the effects of MMSE on both the imaging and
# language/cognitive views prior to decomposition.
```

Table 3: Summary of training and testing linear models with nuisance variables included after decomposition (option 1). 90% confidence intervals for the covariates are given in parentheses.

	Training	**Testing**
GM1	22.123*** (16.405, 27.842)	20.017*** (14.187, 25.846)
GM2	3.099** (0.855, 5.342)	6.237*** (2.866, 9.609)
GM3	−0.609 (−3.324, 2.105)	0.812 (−2.173, 3.797)
GM4	5.636*** (3.064, 8.207)	−0.999 (−4.104, 2.107)
Age	−0.001 (−0.003, 0.001)	0.0003 (−0.002, 0.002)
MMSE	−0.009*** (−0.012, −0.006)	−0.008*** (−0.012, −0.004)
Constant	0.757*** (0.337, 1.177)	0.889*** (0.452, 1.326)
Observations	89	83
R^2	0.683	0.532
Adjusted R^2	0.660	0.495
Residual Std. Error	0.093 (df = 82)	0.097 (df = 76)
F Statistic	29.440*** (df = 6; 82)	14.393*** (df = 6; 76)
Note:		*p<0.1; **p<0.05; ***p<0.01

```
trainingCognitiveResiduals <- residuals( lm(
  trainingCognitiveMatrix ~ trainingCognitiveData$mmse + trainingCognitiveData$age ) )
trainingImageResiduals <- residuals( lm(
  trainingImageData ~ trainingCognitiveData$mmse + trainingCognitiveData$age ) )

eigenInitialization <- initializeEigenanatomy( aalLanguageRegionalLabelMatrix,
                        grayMatterMask )

cognitiveSccanResult <- sparseDecom2(
     inmatrix = list( scale( trainingImageResiduals ), scale( trainingCognitiveResiduals ) ),
     its = 20, mycoption = 0, sparseness = c( 0, -0.9 ), nvecs = numberOfCognitiveLabels,
     inmask = c( grayMatterMask, NA ), cthresh = c( 1000, 0 ), verbose = 0, ell1 = 10,
     initializationList = eigenInitialization$initlist, priorWeight = 0.1,
     smooth = 0, perms = 0 )

# Decomposition of the residualized data produces eigenvectors which we can apply
# to the original data.  We then construct the linear models with the original
# testing and training data but including the nuisance variables in the model formula.

trainingImagePredictors <- trainingImageData %*% cognitiveSccanResult$eig1
colnames( trainingImagePredictors ) <- paste0( "GM", c( 1:numberOfCognitiveLabels ) )
trainingCognitivePredictors <- trainingCognitiveMatrix %*% cognitiveSccanResult$eig2
bestPredictor <- which.max(
  abs( diag( cor( trainingImagePredictors, trainingCognitivePredictors ) ) ) )

trainingDataFrame <- data.frame( trainingImagePredictors, trainingCognitivePredictors,
   MMSE = trainingCognitiveData$mmse, Age = trainingCognitiveData$age )
lmFormula <- as.formula( paste0( "Variate00" ,bestPredictor-1 ,
   "~ GM1+GM2+GM3+GM4+Age+MMSE" ) )
trainingLm <- lm( lmFormula, data = trainingDataFrame )

testingImagePredictors <- testingImageData %*% cognitiveSccanResult$eig1
colnames( testingImagePredictors ) <- paste0( "GM", c( 1:numberOfCognitiveLabels ) )
```

```
testingCognitivePredictors <- testingCognitiveMatrix %*% cognitiveSccanResult$eig2
testingDataFrame <- data.frame( testingImagePredictors, testingCognitivePredictors,
  MMSE = testingCognitiveData$mmse, Age = testingCognitiveData$age )
testingLm <- lm( lmFormula, data = testingDataFrame )
```

```
# We output the two linear models in Table 3 using the stargazer package.

stargazer( trainingLm, testingLm, model.names = FALSE, header = FALSE,
  no.space = TRUE, ci = TRUE, ci.level = 0.90, single.row = TRUE, type = 'latex',
  table.placement = 'h', dep.var.caption = "", dep.var.labels.include = FALSE,
  column.labels = c( "{\\bf Training}", "{\\bf Testing}" ), model.numbers = FALSE,
  title = "Summary of training and testing linear models with nuisance variables
   included after decomposition on the residualized data matrices (option 2).
   90\\% confidence intervals for the covariates are given in parentheses." )
```

Table 4: Summary of training and testing linear models with nuisance variables included after decomposition on the residualized data matrices (option 2). 90% confidence intervals for the covariates are given in parentheses.

	Training	**Testing**
GM1	-5.166^{***} (-8.318, -2.014)	-3.388^{*} (-6.658, -0.118)
GM2	18.949^{***} (14.521, 23.377)	17.008^{***} (12.492, 21.525)
GM3	-0.561 (-4.049, 2.928)	-1.589 (-5.498, 2.319)
GM4	2.007 (-1.462, 5.477)	3.489 (-0.361, 7.339)
Age	0.001 (-0.002, 0.003)	-0.001 (-0.003, 0.001)
MMSE	0.007^{***} (0.004, 0.010)	0.006^{**} (0.001, 0.010)
Constant	-1.472^{***} (-2.028, -0.915)	-1.272^{***} (-1.853, -0.691)
Observations	89	83
R^2	0.607	0.502
Adjusted R^2	0.578	0.463
Residual Std. Error	0.111 (df = 82)	0.118 (df = 76)
F Statistic	21.104^{***} (df = 6; 82)	12.779^{***} (df = 6; 76)
Note:		*p<0.1; **p<0.05; ***p<0.01

Predicting the full cognitive battery from the neuroimaging data

```
# Try to predict all the demographic variability from the imaging data. We use
# `mycoption 0` to try to reduce correlation in low-dimensional space.  This
# enforces a new SCCAN constraint (not previously reported).  (Nick:  We've been
# using mycoption = 0 this whole time.)

trainingCognitiveMatrix <- as.matrix( trainingCognitiveData )

trainingCognitiveResiduals <- residuals( lm(
  trainingCognitiveMatrix ~ trainingCognitiveData$mmse + trainingCognitiveData$age ) )
trainingImageResiduals <- residuals( lm( trainingImageData ~ trainingCognitiveData$mmse ) )

cognitiveSccanResult <- sparseDecom2(
     inmatrix = list( scale( trainingImageResiduals ),
```

```
      scale( trainingCognitiveResiduals ) ),
    its = 20, mycoption = 0, sparseness = c( 0.02, -0.05 ), nvecs = 11,
    inmask = c( grayMatterMask, NA ), cthresh = c( 1000, 0 ), smooth = 0.5 )

trainingImagePredictors <- trainingImageData %*% cognitiveSccanResult$eig1
colnames( trainingImagePredictors ) <-
  paste0( "GM", c( 1:ncol( trainingImagePredictors ) ) )
trainingCognitivePredictors <- trainingCognitiveResiduals %*% cognitiveSccanResult$eig2

testingImagePredictors <- testingImageData %*% cognitiveSccanResult$eig1
colnames( testingImagePredictors ) <-
  paste0( "GM", c( 1:ncol( testingImagePredictors ) ) )

testingCognitiveMatrix <- as.matrix( testingCognitiveData )
testingCognitivePredictors <- testingCognitiveMatrix %*% cognitiveSccanResult$eig2

eigenImageList <- matrixToImages( t( cognitiveSccanResult$eig1 ), grayMatterMask )

cognitivePredictorNames <- rep( 'NA', ncol( cognitiveSccanResult$eig2 ) )
weights <- rep( 'NA', ncol( cognitiveSccanResult$eig2 ) )
correlations <- rep( 'NA', ncol( cognitiveSccanResult$eig2 ) )
predictorRegions <- rep( 'NA', ncol( cognitiveSccanResult$eig2 ) )
predictorRegionPlot3DFiles <- rep( 'NA', ncol( cognitiveSccanResult$eig2 ) )

for( i in 1:ncol( cognitiveSccanResult$eig2 ) )
  {
  trainingCognitiveDataFrame <- data.frame( Cognitive = trainingCognitivePredictors[, i],
    trainingImagePredictors, Age = trainingCognitiveData$age,
    MMSE = trainingCognitiveData$mmse )
  lmFormula <- formula( paste( "Cognitive ~ Age + MMSE + ",
    paste0( "GM", c( 1:ncol( trainingImagePredictors ) ), collapse = '+' ),
    collapse = '+' ) )
  trainingLm <- lm( lmFormula, data = trainingCognitiveDataFrame )
  trainingLmStats <- bigLMStats( trainingLm )

  testingCognitiveDataFrame <- data.frame( Cognitive = testingCognitivePredictors[, i],
    testingImagePredictors, Age = testingCognitiveData$age,
    MMSE = testingCognitiveData$mmse )

  testingCorrelation <- cor( testingCognitivePredictors[, i], predict( trainingLm,
    newdata = testingCognitiveDataFrame ) )
  correlations[i] <- format( testingCorrelation, digits = 3 )

  nonZeroIndices <- which( abs( cognitiveSccanResult$eig2[, i] ) > 0 )
  nonZeroNames <- colnames( trainingCognitiveData )[nonZeroIndices]
  nonZeroWeights <- abs( cognitiveSccanResult$eig2[nonZeroIndices, i] )

  cognitivePredictorNames[i] <- paste( nonZeroNames, collapse = ', ' )
  weights[i] <- paste( format( nonZeroWeights, digits = 3 ), collapse = ', ' )

  regions <- reportAnatomy( list( eigenImageList[[i]] ), grayMatterMask, 0.5 )
  predictorRegions[i] <- paste( regions, collapse = ", " )
```

```
  significantIndices <- which( p.adjust( trainingLmStats$beta.pval ) < 0.1 ) - 2
  visualizationImages <- list()
  for( j in 1:length( significantIndices ) )
    {
    visualizationImages[[j]] <-
      abs( antsImageClone( eigenImageList[[significantIndices[j]]] ) )
    }

  brain <- renderSurfaceFunction( surfimg = list( aalImage ), alphasurf = 0.1,
      funcimg = visualizationImages, smoothsval = 1.5, smoothfval = 0,
      mycol = rainbow( length( visualizationImages ) ) )

  id <- par3d( "userMatrix" )
  rid <- rotate3d( id, -pi / 2, 1, 0, 0 )
  rid2 <- rotate3d( id, pi / 2, 0, 0, 1 )
  par3d( userMatrix = id )

  predictorRegionPlot3DFiles[i] <- paste0( figuresDirectory,
    "predictorSccanEigenImages3D_Vector", i )
  dd <- make3ViewPNG( rid, id, rid2, predictorRegionPlot3DFiles[i] )
  predictorRegionPlot3DFiles[i] <- paste0( figuresDirectory,
    "predictorSccanEigenImages3D_Vector", i, ".png" )
  par3d( userMatrix = id )
  }

cognitiveSccanDataFrame <- data.frame( 1:ncol( cognitiveSccanResult$eig2 ),
  cognitivePredictorNames, weights, correlations )
colnames( cognitiveSccanDataFrame ) <- c( 'Vector', 'Cognitive predictors',
  'Eigenvector weights', 'Correlation' )
pander( cognitiveSccanDataFrame, style = "rmarkdown", caption = "The set of
  cognitive predictors (with non-zero weights) for each eigenvector." )
```

Table 5: The set of cognitive predictors (with non-zero weights) for each eigenvector.

Vector	Cognitive predictors	Eigenvector weights	Correlation
1	socialcomportment	0.107	0.108
2	naming_adj, JOLO_adj	1.28e-01, 6.52e-07	0.41
3	fluency_adj	0.123	0.228
4	apathy	0.122	0.463
5	fluency_adj, recog_adj	3.20e-07, 1.11e-01	0.000591
6	speech_adj	0.109	-0.108
7	reading_adj	0.108	0.185
8	semantic_adj	0.111	-0.0848
9	age, JOLO_adj	1.07e-01, 7.83e-08	-0.319
10	dig_bwd_adj	0.107	0.198
11	agitation	0.136	0.176

```
cognitiveSccanDataFrame <- data.frame( 1:ncol( cognitiveSccanResult$eig2 ),
  predictorRegions )
colnames( cognitiveSccanDataFrame ) <- c( 'Eigenvector', 'Predictor regions' )
```

```
pander( cognitiveSccanDataFrame, style = "rmarkdown", caption = "The set of
  predictor regions for each eigenvector." )
```

Table 6: The set of predictor regions for each eigenvector. (continued below)

Eigenvector	Predictor regions
1	Frontal_Inf_Oper_R, Frontal_Inf_Tri_R, Frontal_Inf_Orb_R, Insula_R Temporal_Mid_R, Temporal_Inf_R
2	Precuneus_R
3	Frontal_Sup_Medial_L, Frontal_Med_Orb_L, Cingulum_Ant_L Temporal_Mid_L, Temporal_Inf_L
4	Fusiform_L, Temporal_Inf_L
5	ParaHippocampal_R, Fusiform_R
6	SupraMarginal_R, Temporal_Sup_R
7	Frontal_Sup_R
8	SupraMarginal_L, Temporal_Sup_L, Temporal_Mid_L
9	Frontal_Mid_L
10	
11	

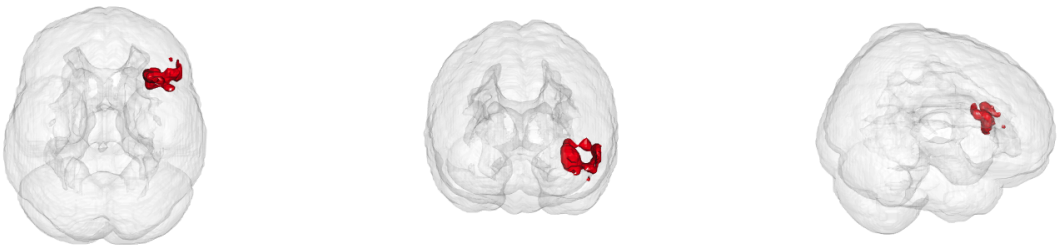


Figure 7: Eigenvector 1 of the gray matter imaging view (cf Table 5).

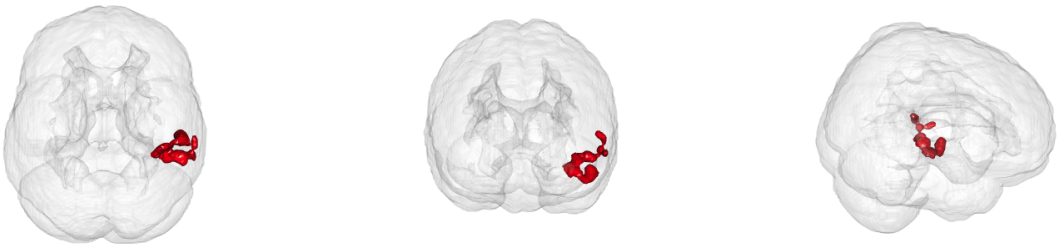


Figure 8: Eigenvector 2 of the gray matter imaging view (cf Table 5).

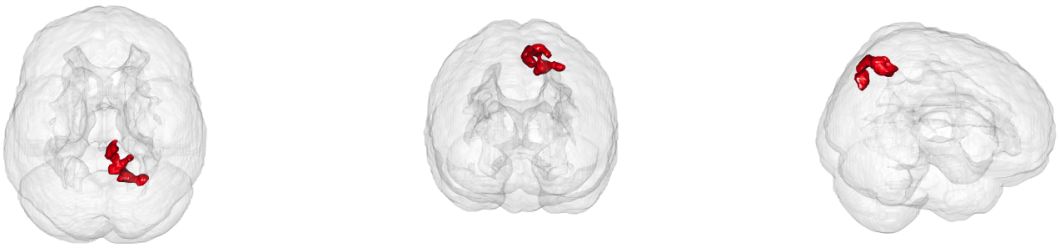


Figure 9: Eigenvector 3 of the gray matter imaging view (cf Table 5).

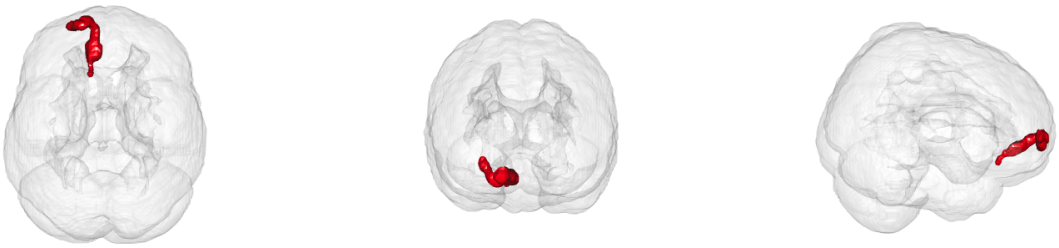


Figure 10: Eigenvector 4 of the gray matter imaging view (cf Table 5).

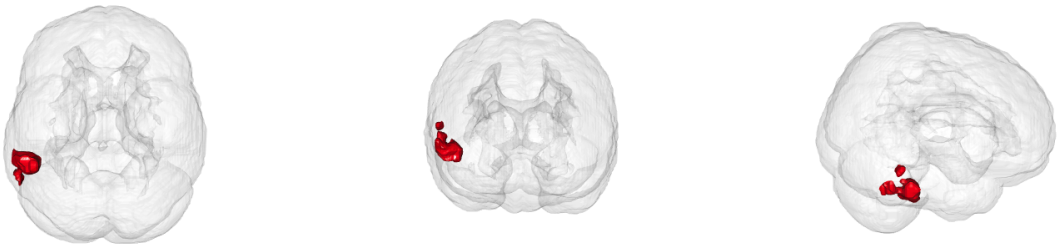


Figure 11: Eigenvector 5 of the gray matter imaging view (cf Table 5).

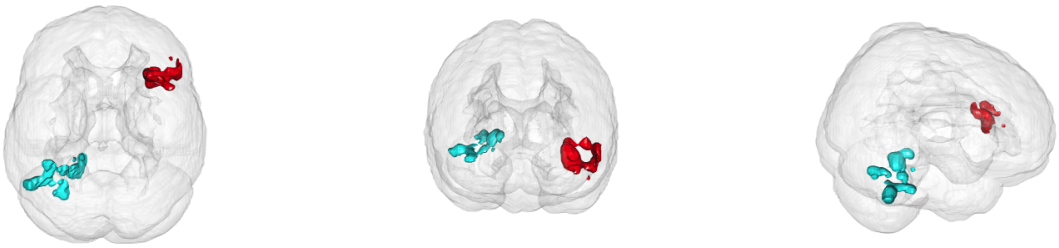


Figure 12: Eigenvector 6 of the gray matter imaging view (cf Table 5).

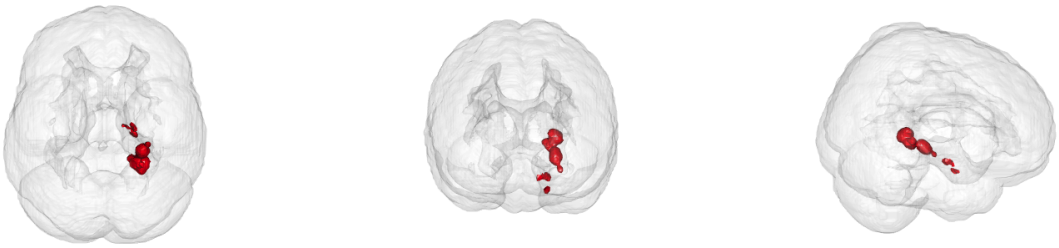


Figure 13: Eigenvector 7 of the gray matter imaging view (cf Table 5).

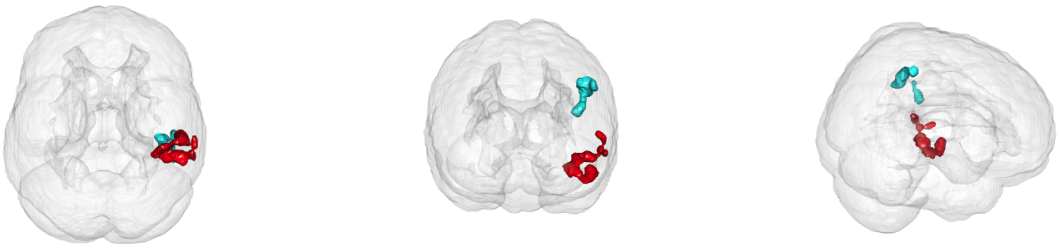


Figure 14: Eigenvector 8 of the gray matter imaging view (cf Table 5).

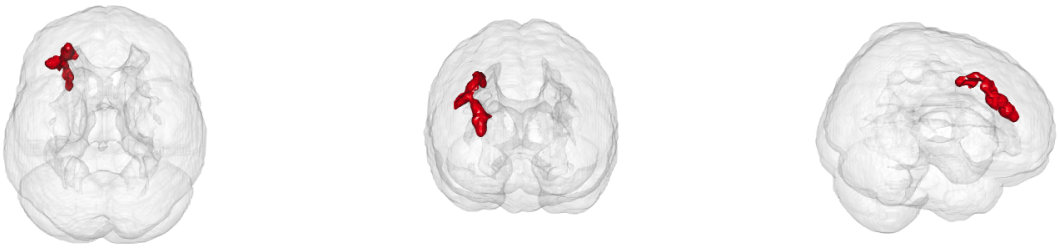


Figure 15: Eigenvector 9 of the gray matter imaging view (cf Table 5).

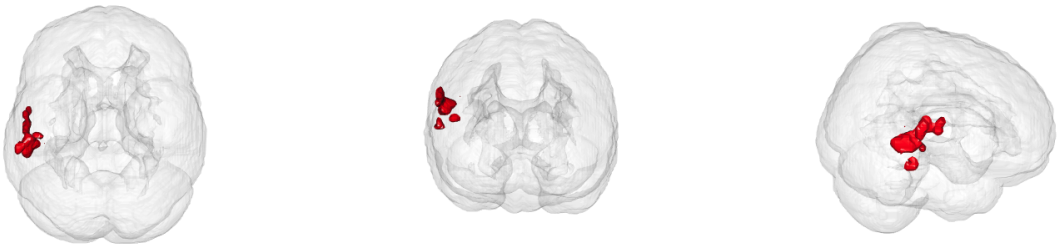


Figure 16: Eigenvector 10 of the gray matter imaging view (cf Table 5).

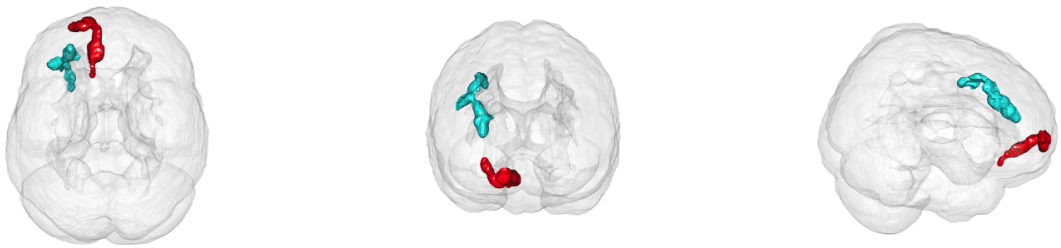
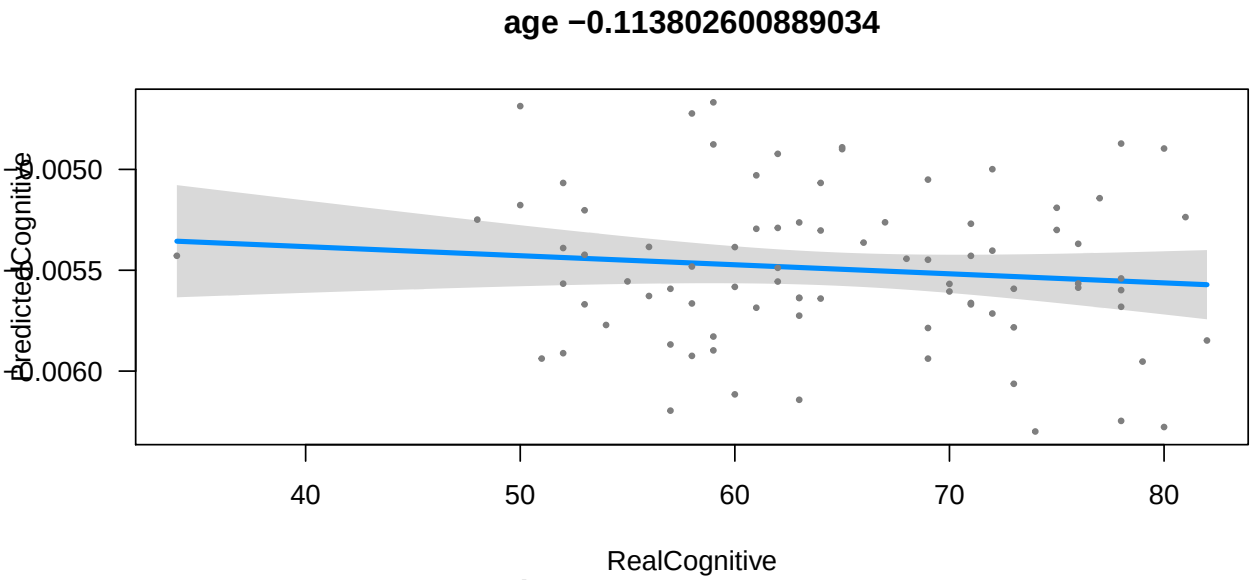


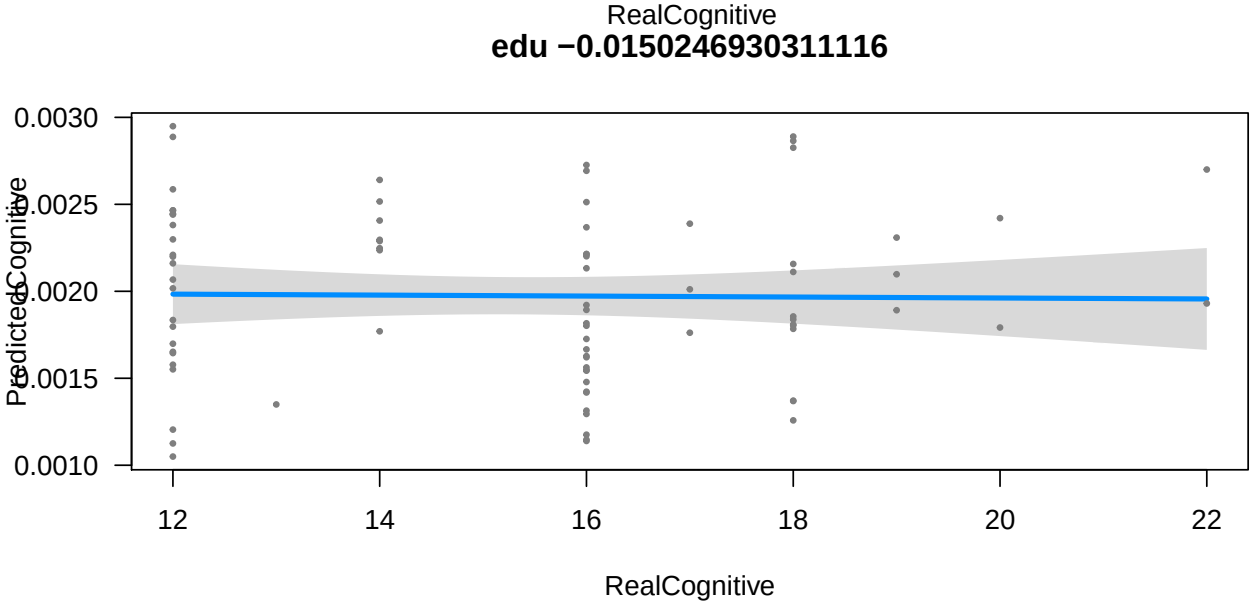
Figure 17: Eigenvector 11 of the gray matter imaging view (cf Table 5).

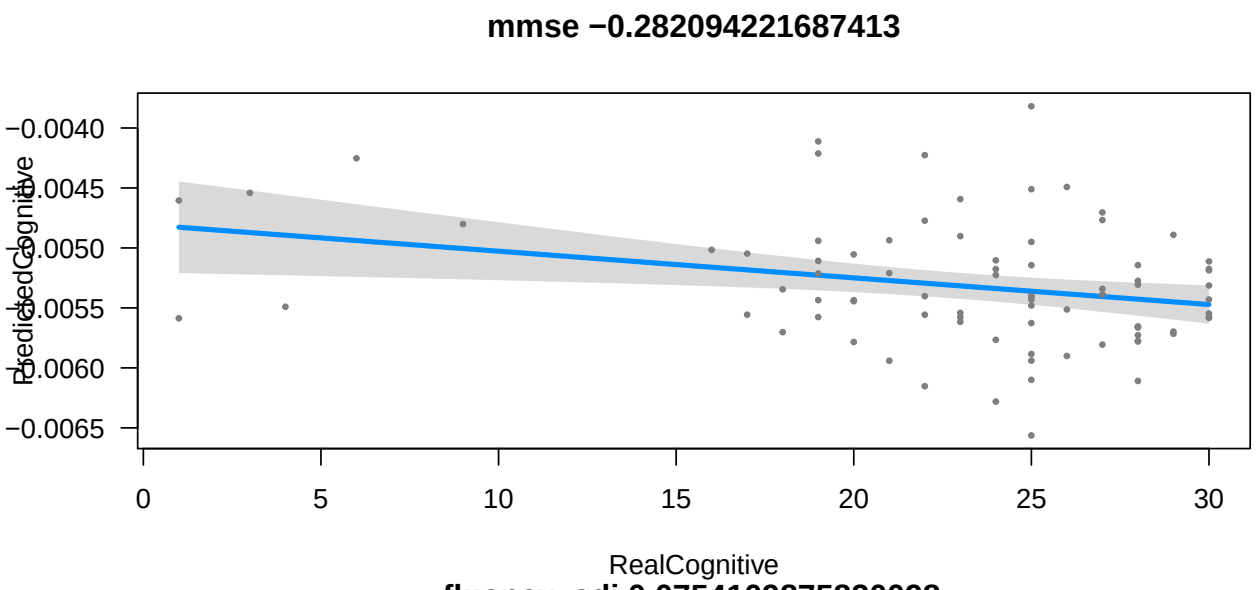
Predicting the full cognitive battery

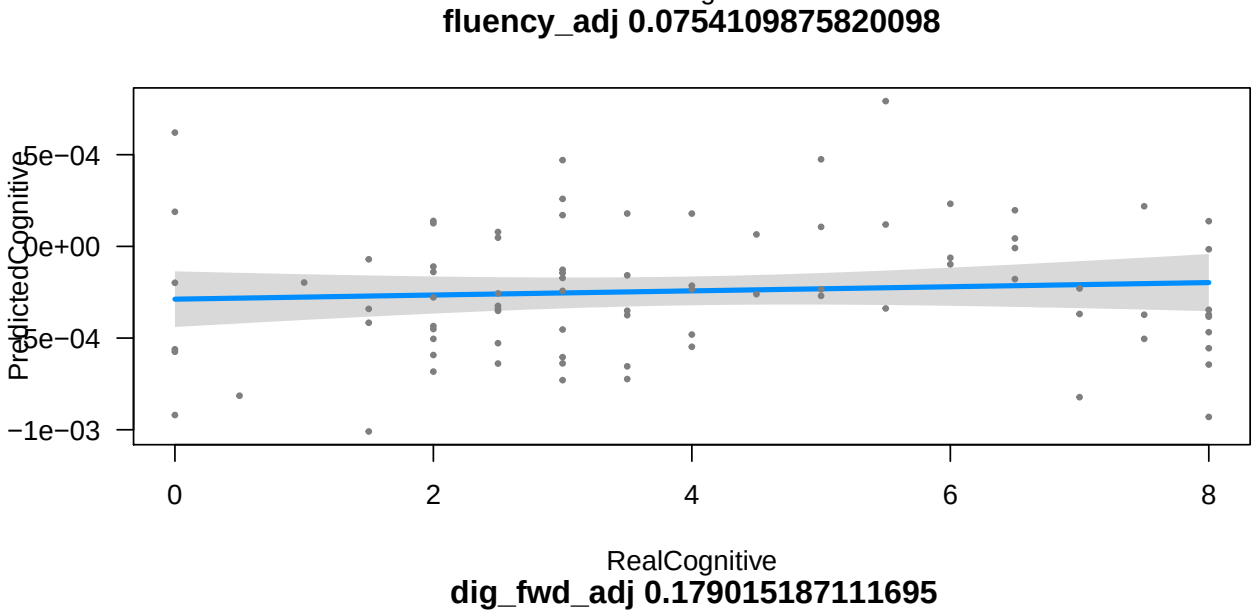
```
# We use SCCAN to transform the gray matter signal for predicting the full cognitive
# battery.

# We recalculate the residualized because of r chunk caching.

trainingCognitiveResiduals <- residuals( lm(
  trainingCognitiveMatrix ~ trainingCognitiveData$mmse + trainingCognitiveData$age ) )
trainingImageResiduals <- residuals( lm( trainingImageData ~ trainingCognitiveData$mmse ) )

cognitiveSccanResult <- sparseDecom2(
     inmatrix = list( scale( trainingImageResiduals ),
       scale( trainingCognitiveResiduals ) ), verbose = TRUE,
     its = 20, mycoption = 0, sparseness = c( 0.02, -0.9 ), nvecs = 11,
     inmask = c( grayMatterMask, NA ), cthresh = c( 1000, 0 ), smooth = 0.5 )

predictedTrainingCognitiveBattery <- data.frame( Voxels = trainingImageData %*%
   cognitiveSccanResult$eig1 %*% t( cognitiveSccanResult$eig2 ) )

predictedTestingCognitiveBattery <- data.frame( Voxels = testingImageData %*%
   cognitiveSccanResult$eig1 %*% t( cognitiveSccanResult$eig2 ) )

testingCorrelationPvalues <- rep( NA, ncol( testingCognitiveData ) )

for( i in 1:ncol( testingCognitiveData ) )
  {
  testingCorrelationPvalues[i] <- cor.test( data.matrix( testingCognitiveData )[, i],
    predictedTestingCognitiveBattery[, i] )$p.value
  cognitiveDataFrame <- data.frame( PredictedCognitive =
    predictedTestingCognitiveBattery[, i], RealCognitive = testingCognitiveData[, i] )
  testingLm <- lm( PredictedCognitive ~ RealCognitive, data = cognitiveDataFrame )
  title <- paste( colnames( testingCognitiveData )[i],
    cor( data.matrix( testingCognitiveData )[, i],
    predictedTestingCognitiveBattery[, i] ) )
  visreg( testingLm , main = title )
```

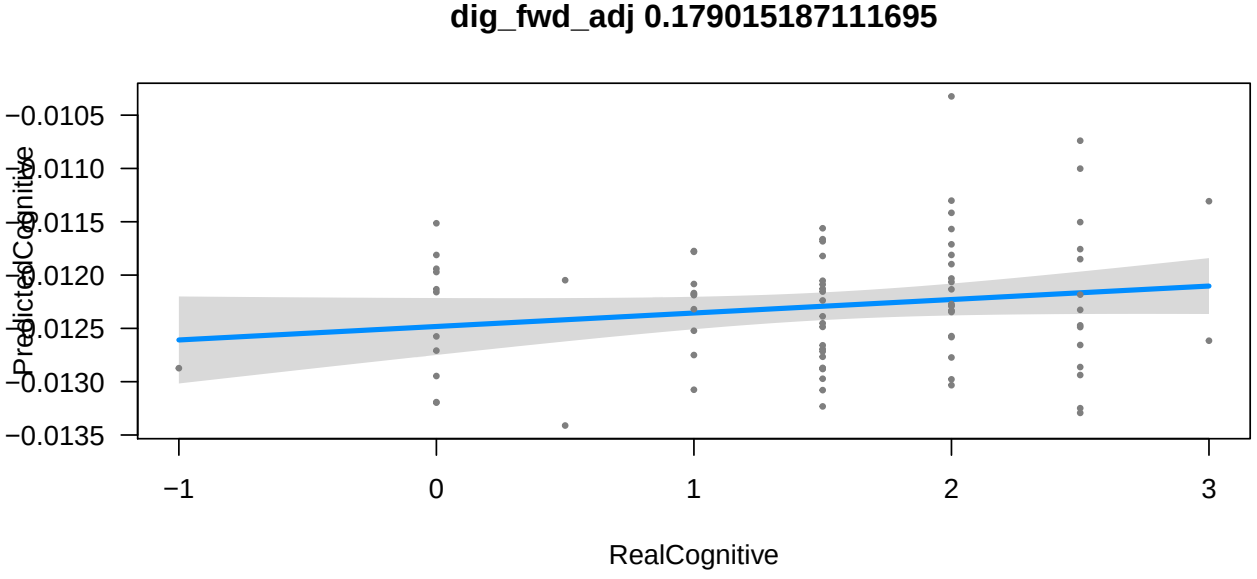
```
}
```

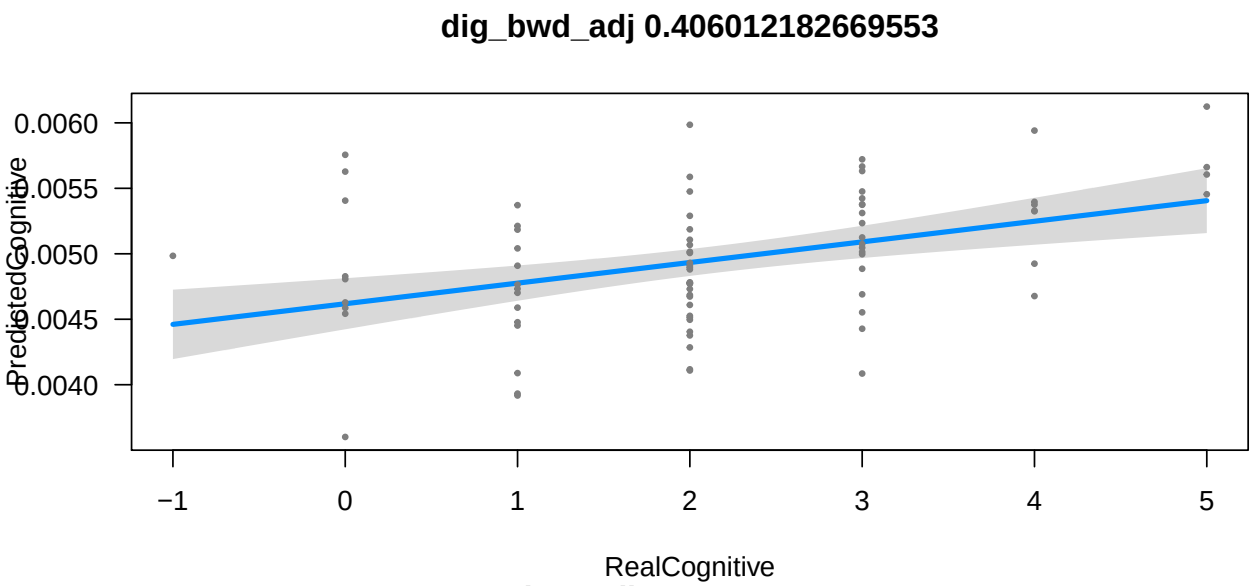

age −0.113802600889034
PredictedCognitive
−0.0050
−0.0055
−0.0060
40
50
60
70
80
RealCognitive

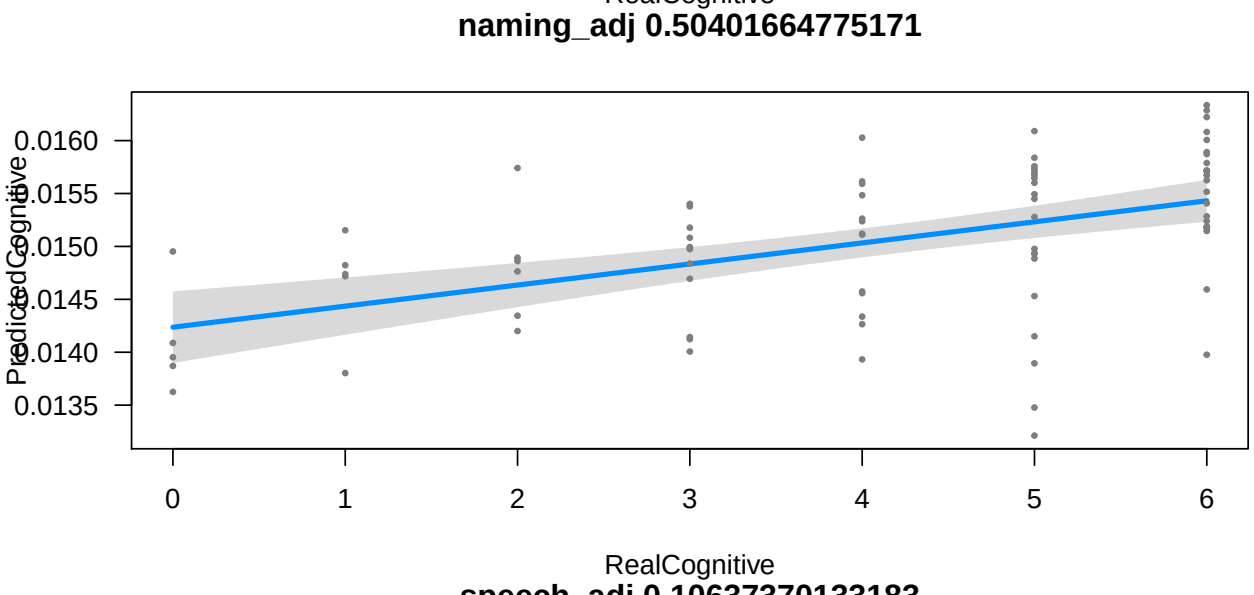

edu −0.0150246930311116
PredictedCognitive
0.0030
0.0025
0.0020
0.0015
0.0010
12
14
16
18
20
22
RealCognitive

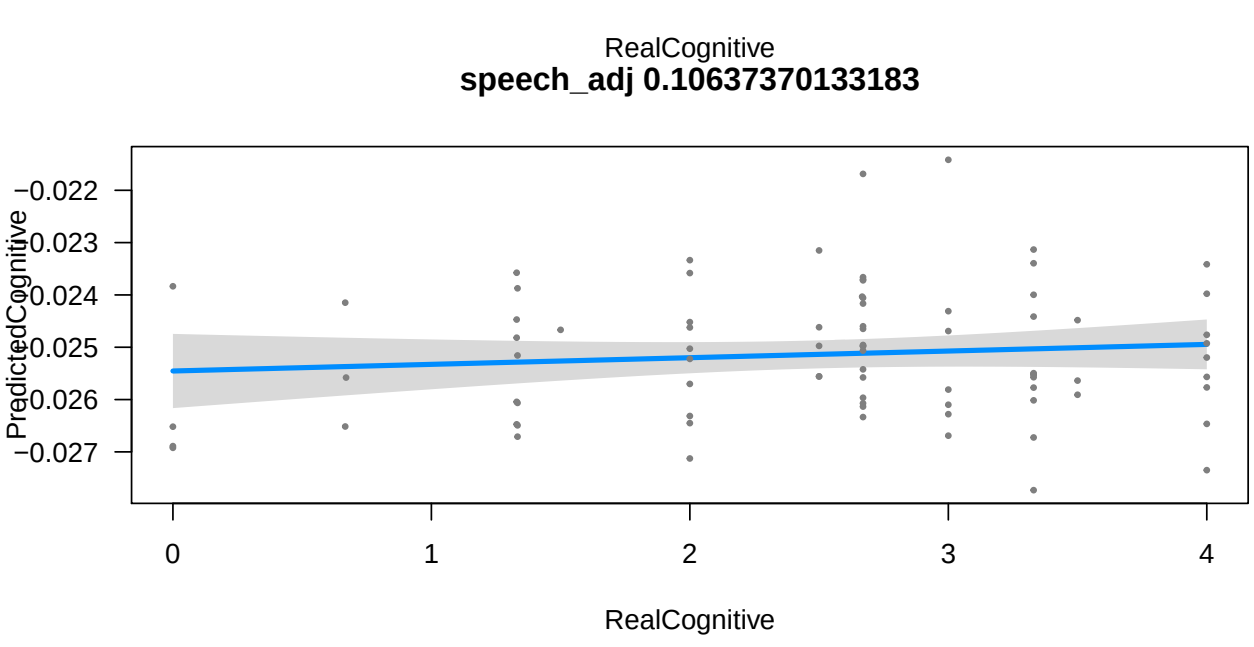

mmse −0.282094221687413
−0.0040
−0.0045
−0.0050
−0.0055
−0.0060
−0.0065
PredictedCognitive
0
5
10
15
20
25
30
RealCognitive

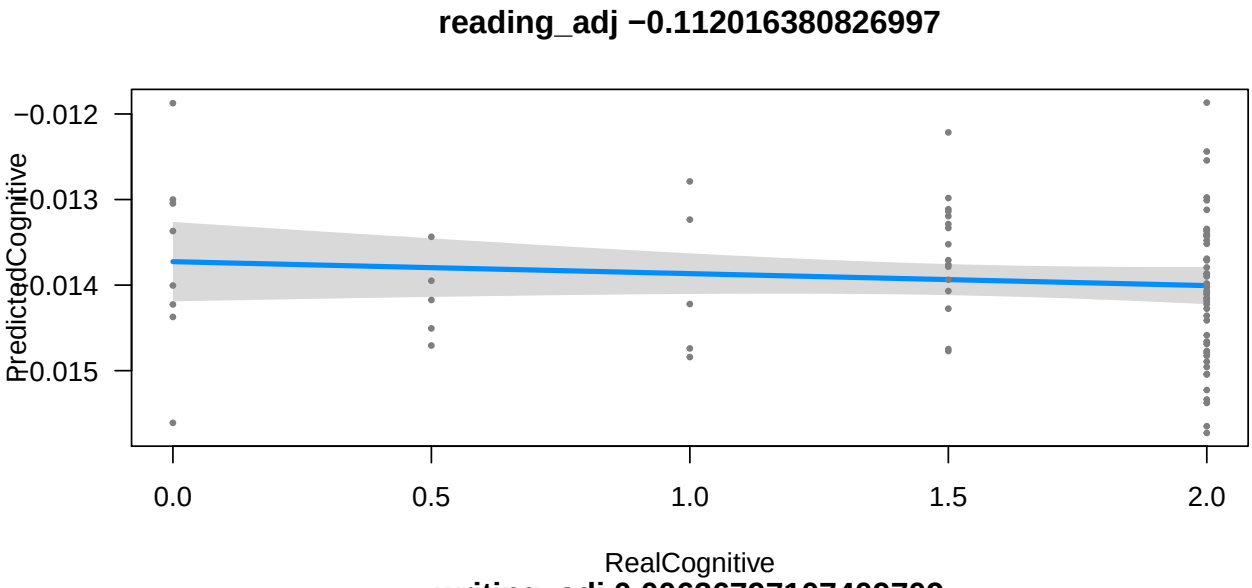

fluency_adj 0.0754109875820098
5e−04
0e+00
−5e−04
−1e−03
PredictedCognitive
0
2
4
6
8
RealCognitive

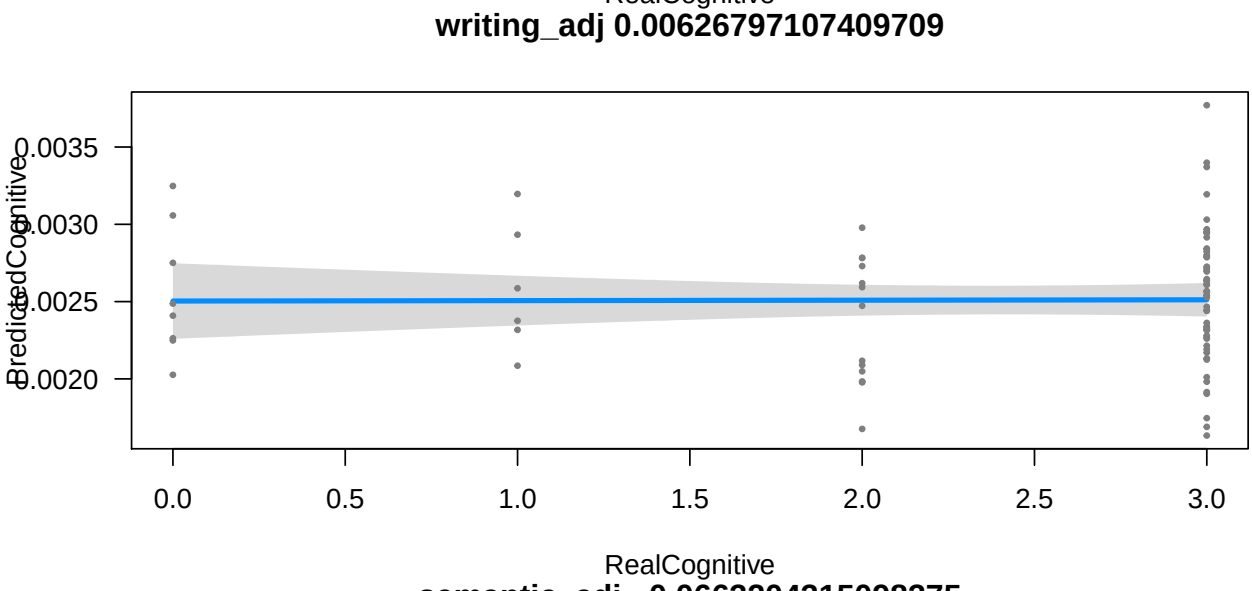

dig_fwd_adj 0.179015187111695
−0.0105
−0.0110
−0.0115
−0.0120
−0.0125
−0.0130
−0.0135
PredictedCognitive
−1
0
1
2
3
RealCognitive

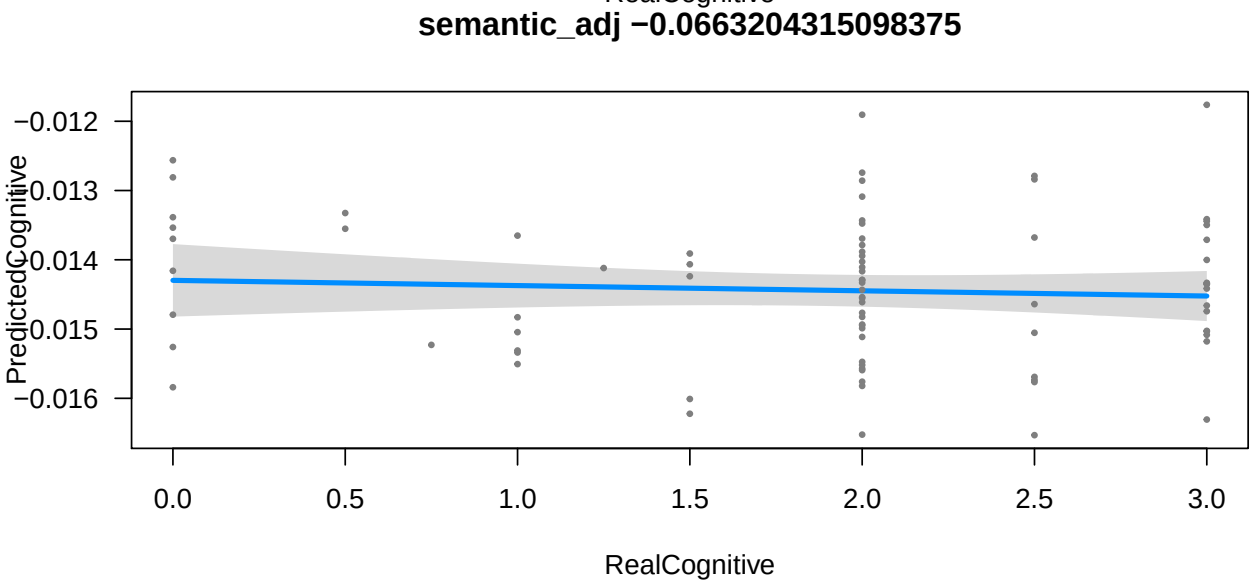

dig_bwd_adj 0.406012182669553
PredictedCognitive
0.0060
0.0055
0.0050
0.0045
0.0040
−1
0
1
2
3
4
5
RealCognitive

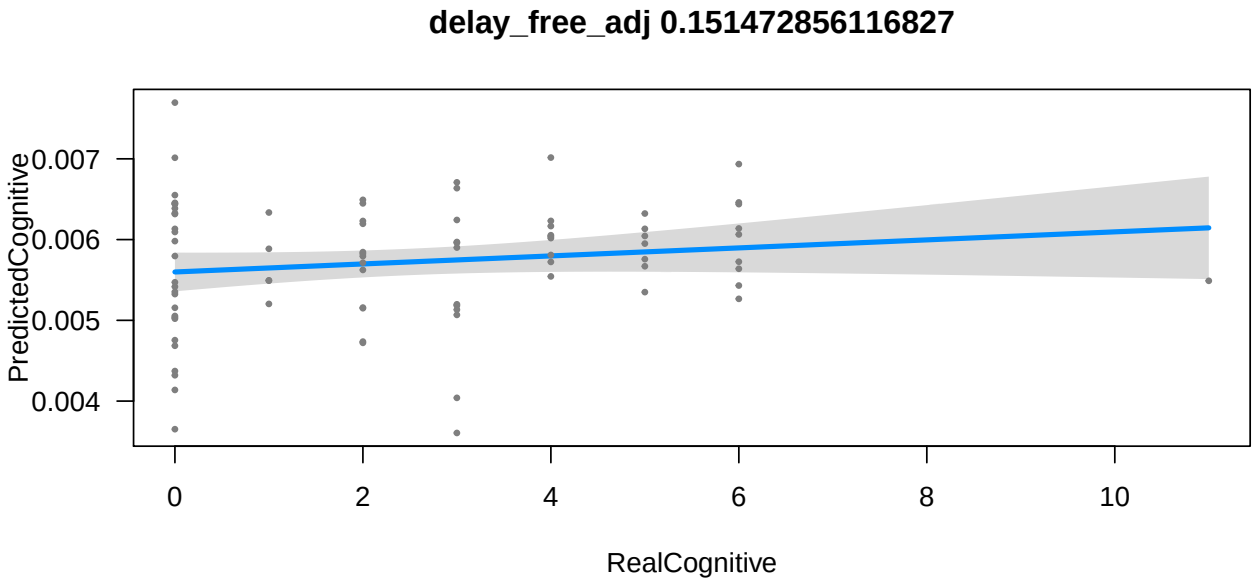

naming_adj 0.50401664775171
PredictedCognitive
0.0160
0.0155
0.0150
0.0145
0.0140
0.0135
0
1
2
3
4
5
6
RealCognitive

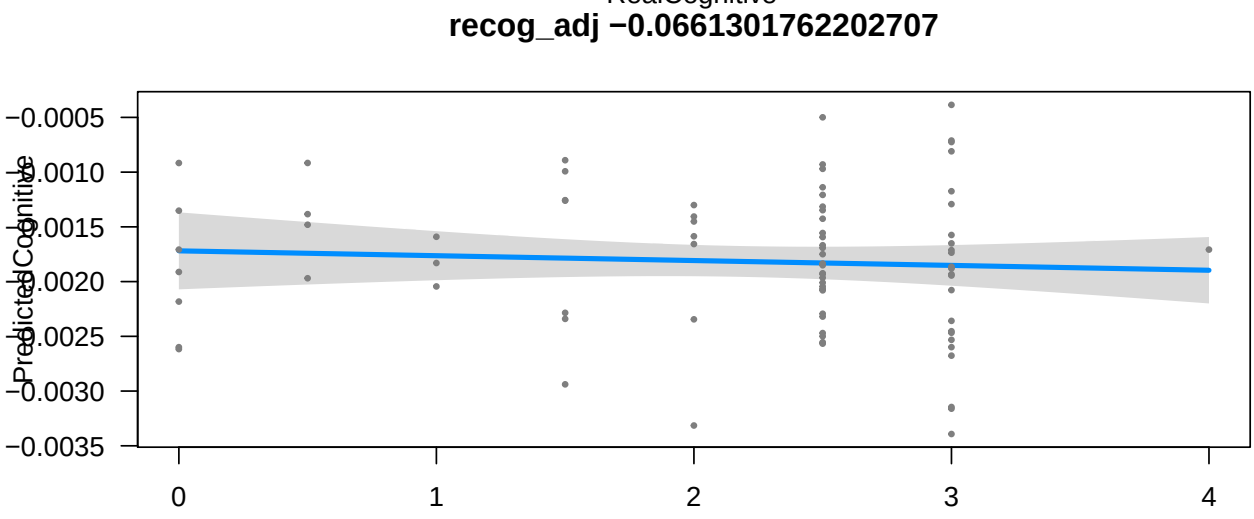

speech_adj 0.10637370133183
PredictedCognitive
−0.022
−0.023
−0.024
−0.025
−0.026
−0.027
0
1
2
3
4
RealCognitive

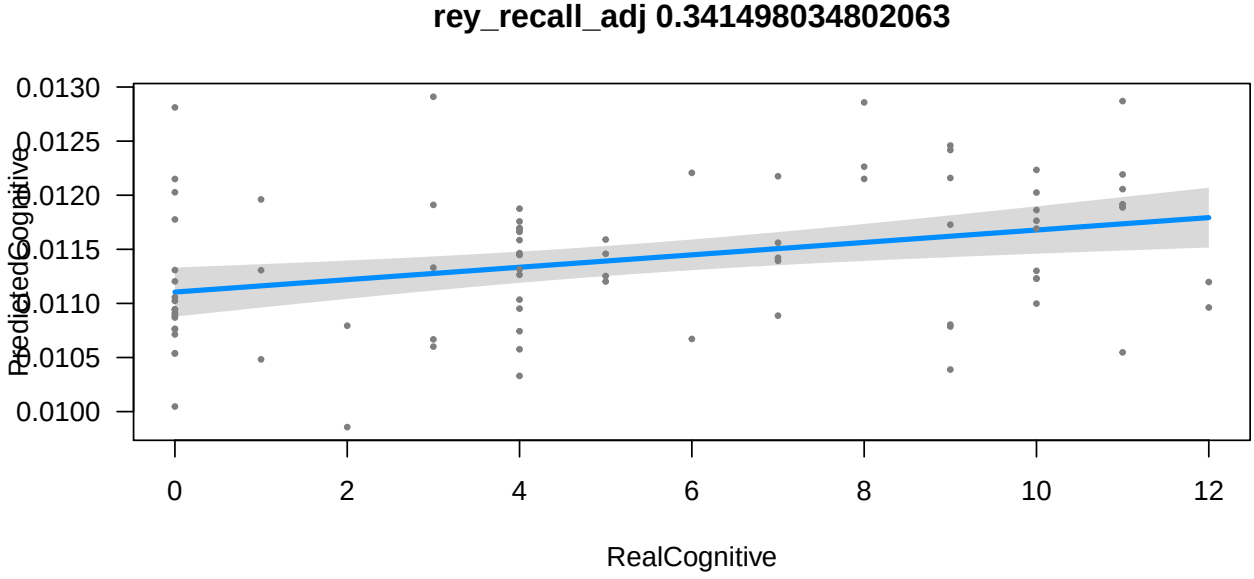

reading_adj −0.112016380826997
PredictedCognitive
−0.012
−0.013
−0.014
−0.015
0.0
0.5
1.0
1.5
2.0
RealCognitive

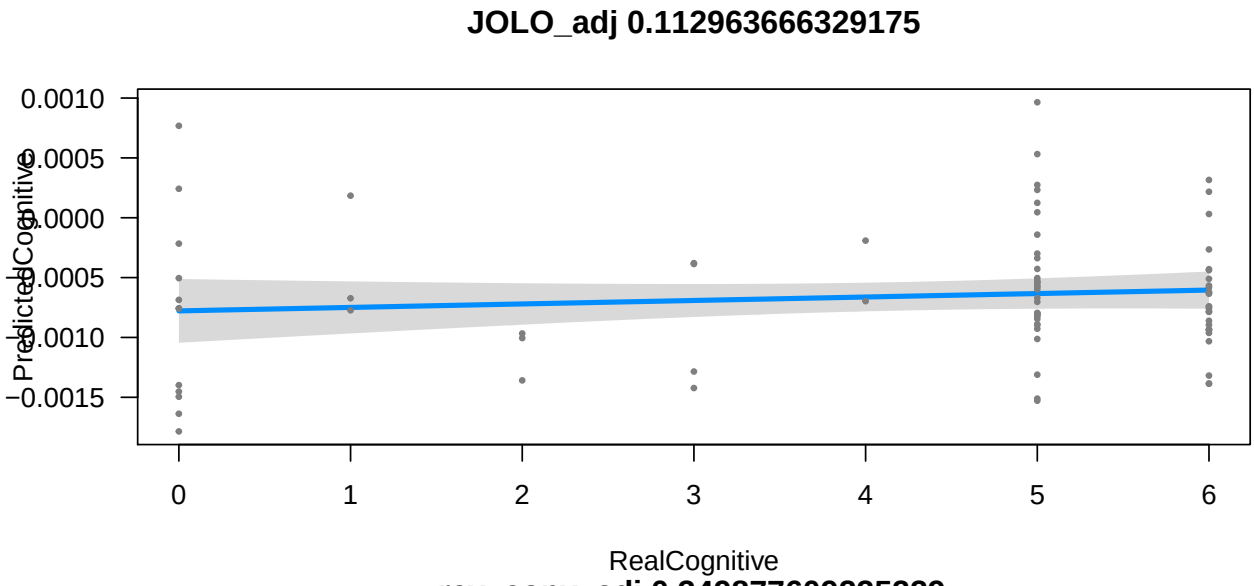

writing_adj 0.00626797107409709
PredictedCognitive
0.0035
0.0030
0.0025
0.0020
0.0
0.5
1.0
1.5
2.0
2.5
3.0
RealCognitive

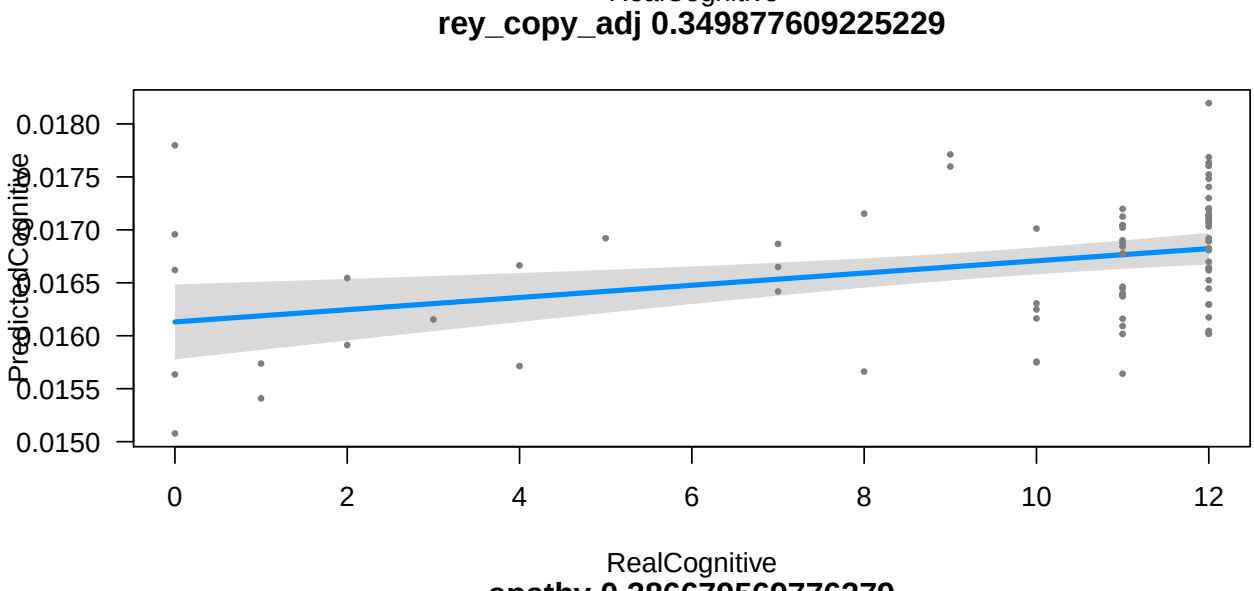

semantic_adj −0.0663204315098375
PredictedCognitive
−0.012
−0.013
−0.014
−0.015
−0.016
0.0
0.5
1.0
1.5
2.0
2.5
3.0
RealCognitive

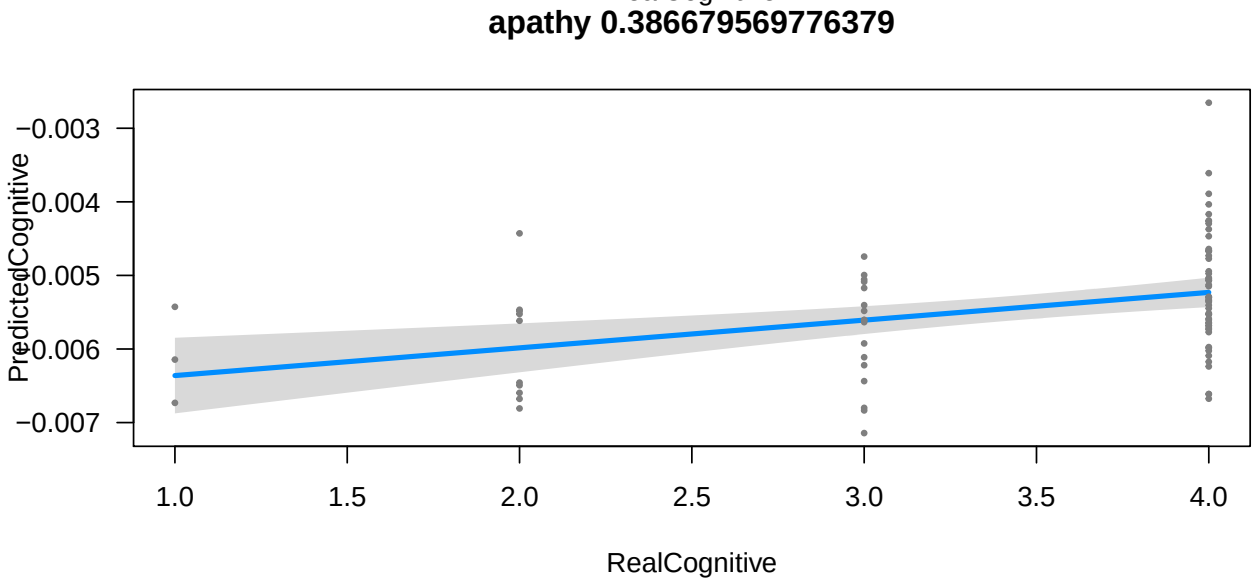

delay_free_adj 0.151472856116827
PredictedCognitive
0.007
0.006
0.005
0.004
0
2
4
6
8
10
RealCognitive

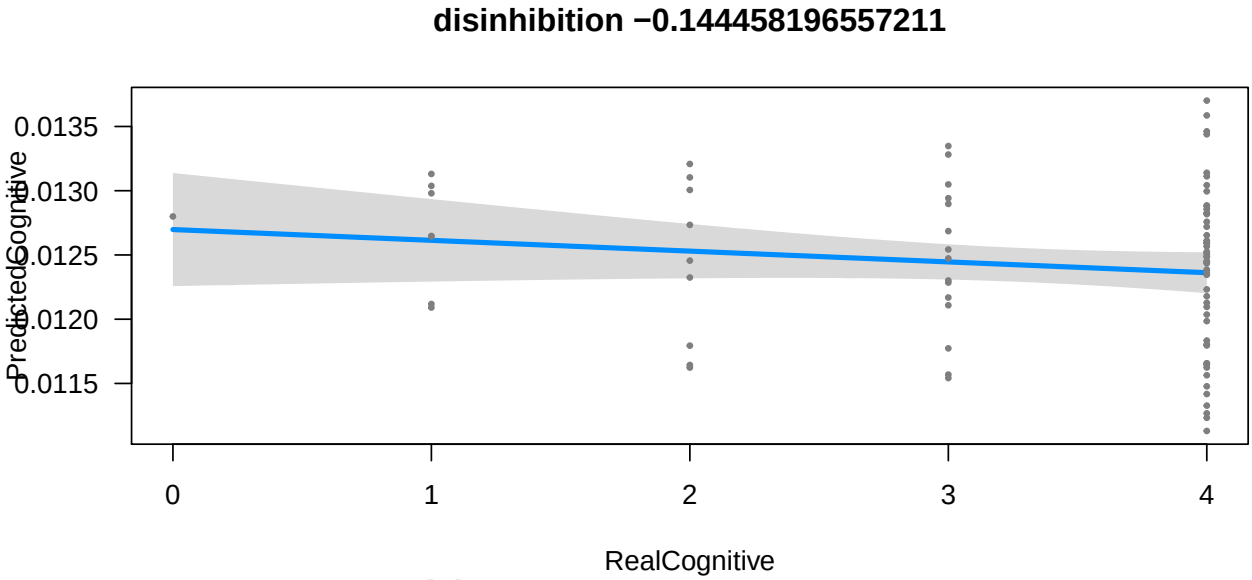

recog_adj −0.0661301762202707
PredictedCognitive
−0.0005
−0.0010
−0.0015
−0.0020
−0.0025
−0.0030
−0.0035
0
1
2
3
4
RealCognitive

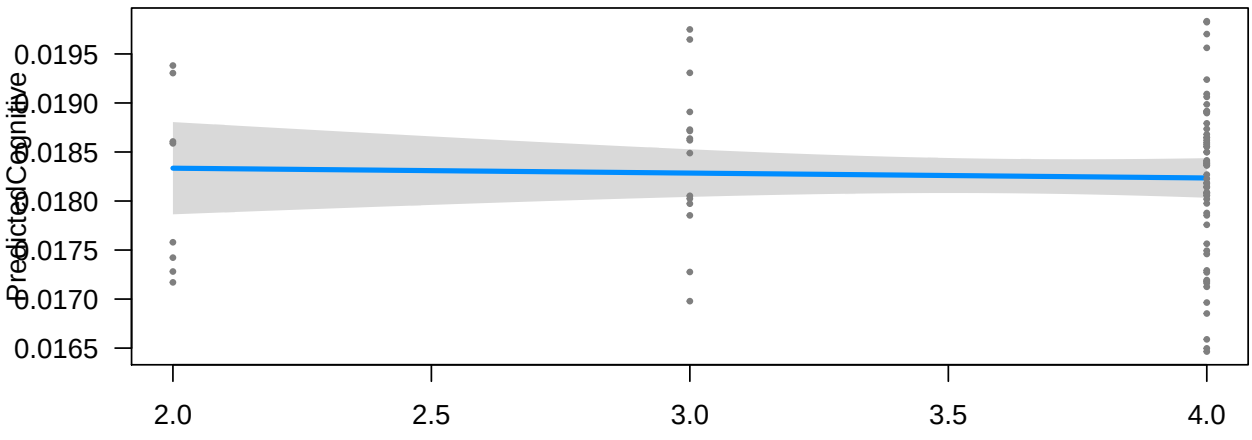

rey_recall_adj 0.341498034802063
PredictedCognitive
0.0130
0.0125
0.0120
0.0115
0.0110
0.0105
0.0100
0
2
4
6
8
10
12
RealCognitive

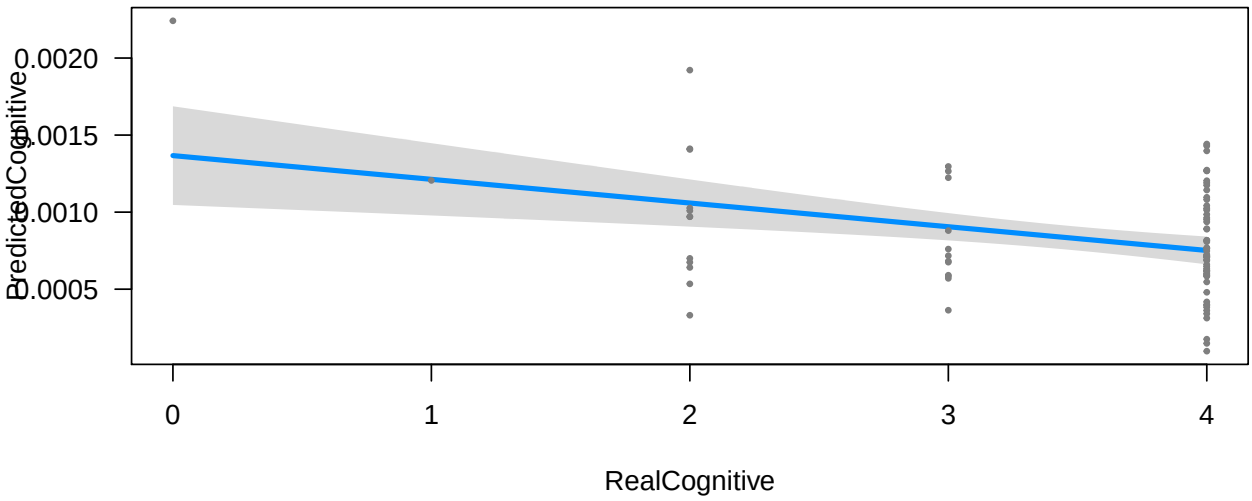

JOLO_adj 0.112963666329175
PredictedCognitive
0.0010
0.0005
0.0000
−0.0005
−0.0010
−0.0015
0
1
2
3
4
5
6
RealCognitive

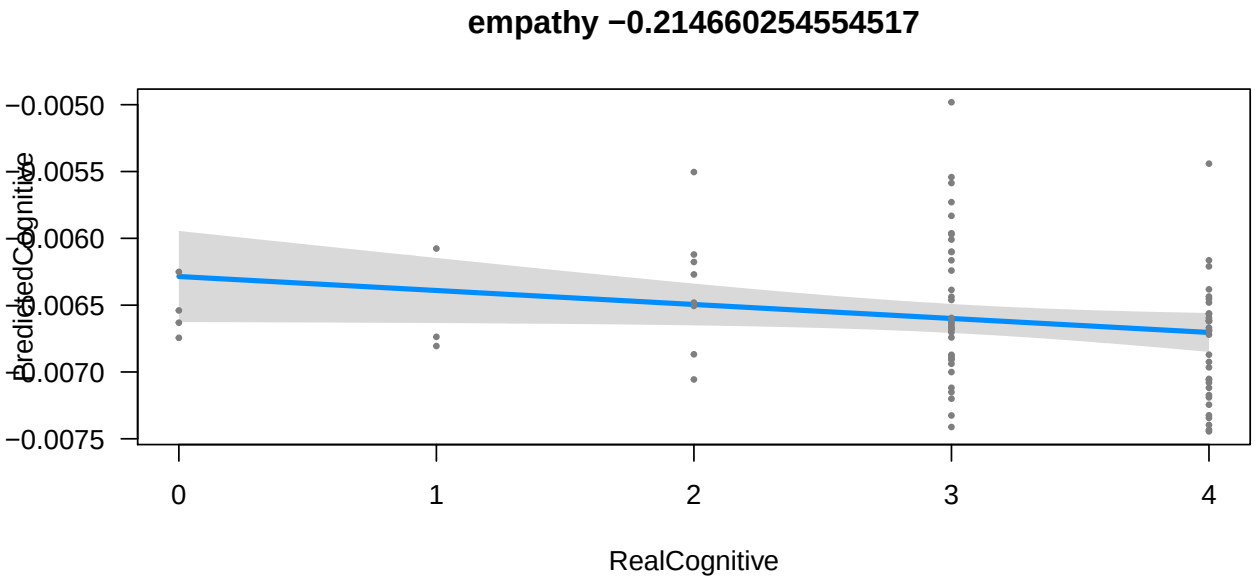

rey_copy_adj 0.349877609225229
PredictedCognitive
0.0180
0.0175
0.0170
0.0165
0.0160
0.0155
0.0150
0
2
4
6
8
10
12
RealCognitive


apathy 0.386679569776379
PredictedCognitive
−0.003
−0.004
−0.005
−0.006
−0.007
1.0
1.5
2.0
2.5
3.0
3.5
4.0
RealCognitive


disinhibition −0.144458196557211
PredictedCognitive
0.0135
0.0130
0.0125
0.0120
0.0115
0
1
2
3
4
RealCognitive


socialcomportment −0.0409747236782499
PredictedCognitive
0.0195
0.0190
0.0185
0.0180
0.0175
0.0170
0.0165
2.0
2.5
3.0
3.5
4.0
RealCognitive


agitation −0.356389564579693
PredictedCognitive
0.0020
0.0015
0.0010
0.0005
0
1
2
3
4
RealCognitive




```
# Correct for multiple comparisons testing and print those cognitive batteries that
# can be predicted using the decomposed imaging data.

testingCorrelationPvalues[is.na( testingCorrelationPvalues )] <- 1
testingCorrelationPvalues <- p.adjust( testingCorrelationPvalues, method = 'BH' )

significantIndices <- which( testingCorrelationPvalues <= 0.05 )

testingCorrelationPvaluesDataFrame <- data.frame( Battery =
  colnames( testingCognitiveData )[significantIndices],
  Qvalue = testingCorrelationPvalues[significantIndices] )

pander( testingCorrelationPvaluesDataFrame, style = "rmarkdown", caption =
  "Cognitive batteries capable of being predicted and significance value after
  multiple comparisons correction." )
```

Table 8: Cognitive batteries capable of being predicted and significance value after multiple comparisons correction.

Battery	Qvalue
mmse	0.02932
dig_bwd_adj	0.001469
naming_adj	2.495e-05
rey_recall_adj	0.00553
rey_copy_adj	0.004979
apathy	0.002136
agitation	0.004952

Questions for Brian:

- Are the pseudo-eigenvectors ordered in terms of the variance explained? It would seem not. In the R code chunk `priorinitialization`, I can do the following

```
> i <- 1
> cognitiveSccanResult <- sparseDecom2(
inmatrix = list( scale( trainingImageData ), scale( trainingCognitiveMatrix ) ),
its = 20, mycoption = 0, sparseness = c( 0, -0.5 ), nvecs = numberOfCognitiveLabels,
inmask = c( grayMatterMask, NA ), cthresh = c( 1000, 0 ), verbose = 0, ell1 = 10,
initializationList = eigenInitialization$initlist, priorWeight = priorWeightings[i]
smooth = 0, perms = 0 )
> norm( as.matrix( cognitiveSccanResult$eig1[,1] ), type = "f" )
[1] 0.003823065
> norm( as.matrix( cognitiveSccanResult$eig1[,2] ), type = "f" )
[1] 0.005176968
> norm( as.matrix( cognitiveSccanResult$eig1[,3] ), type = "f" )
[1] 0.008712231
> norm( as.matrix( cognitiveSccanResult$eig1[,4] ), type = "f" )
[1] 0.01931158
> norm( as.matrix( cognitiveSccanResult$eig2[,1] ), type = "f" )
[1] 0.1086147
> norm( as.matrix( cognitiveSccanResult$eig2[,2] ), type = "f" )
[1] 0.08315364
> norm( as.matrix( cognitiveSccanResult$eig2[,3] ), type = "f" )
[1] 0.1380107
> norm( as.matrix( cognitiveSccanResult$eig2[,4] ), type = "f" )
[1] 0.1814885
```

  It would seem that they're coordinated between the two views. Also, it would seem that this accounts for the colors not seeming to coordinate across Figures 1, 2, and 3.

- What does a negative sparseness value imply?

- (priorinitialization, line 350) Is the `abs()` call necessary? I thought the elements of the eigenvector are constrained to be positive.

- In the tutorial, you set `copt <- 0` and write `# 0 for most applications, 1 for priors`. However, in the rcode chunk `priorinitialization` you use `mycoption = 0`. Was there a reason for this? It seems like `0` represents a compromise between spatial orthogonality and low-dimensional orthogonality as these might be at odds, i.e., the notion of eigenvector vs. pseudo-eigenvector.

- For the various options of dealing with nuisance variables, could one residualize out the effects of the nuisance variables and then just work on the "clean" data?

References

1. Hotelling, H. "**Canonical Correlation Analysis (CCA)**" *J. Educ. Psychol.* (1935):

2. Hotelling, H. "**Relations Between Two Sets of Variants**" *Biometrika* (1936): 321–377.

3. Maguire, E. A., Gadian, D. G., Johnsrude, I. S., Good, C. D., Ashburner, J., Frackowiak, R. S., and Frith, C. D. "**Navigation-Related Structural Change in the Hippocampi of Taxi Drivers**" *Proc Natl Acad Sci U S A* 97, no. 8 (2000): 4398–403. doi:10.1073/pnas.070039597

4. Nichols, T. and Hayasaka, S. "**Controlling the Familywise Error Rate in Functional Neuroimaging: A Comparative Review**" *Stat Methods Med Res* 12, no. 5 (2003): 419–46. doi:10.1191/0962280203sm341ra

5. Habeck, C., Stern, Y., and Alzheimer's Disease Neuroimaging Initiative. "**Multivariate Data Analysis for Neuroimaging Data: Overview and Application to Alzheimer's Disease**" *Cell Biochem Biophys* 58, no. 2 (2010): 53–67. doi:10.1007/s12013-010-9093-0

6. Shamy, J. L., Habeck, C., Hof, P. R., Amaral, D. G., Fong, S. G., Buonocore, M. H., Stern, Y., Barnes, C. A., and Rapp, P. R. "**Volumetric Correlates of Spatiotemporal Working and Recognition Memory Impairment in Aged Rhesus Monkeys**" *Cereb Cortex* 21, no. 7 (2011): 1559–73. doi:10.1093/cercor/bhq210

7. McKeown, M. J., Makeig, S., Brown, G. G., Jung, T. P., Kindermann, S. S., Bell, A. J., and Sejnowski, T. J. "**Analysis of FMRI Data by Blind Separation into Independent Spatial Components**" *Hum Brain Mapp* 6, no. 3 (1998): 160–88.

8. Calhoun, V. D., Adali, T., Pearlson, G. D., and Pekar, J. J. "**A Method for Making Group Inferences from Functional MRI Data Using Independent Component Analysis**" *Hum Brain Mapp* 14, no. 3 (2001): 140–51.

9. Calhoun, V. D., Liu, J., and Adali, T. "**A Review of Group ICA for FMRI Data and ICA for Joint Inference of Imaging, Genetic, and ERP Data**" *Neuroimage* 45, no. 1 Suppl (2009): S163–72. doi:10.1016/j.neuroimage.2008.10.057

10. Kandel, B. M., Wang, D. J. J., Gee, J. C., and Avants, B. B. "**Eigenanatomy: Sparse Dimensionality Reduction for Multi-Modal Medical Image Analysis**" *Methods* 73, (2015): 43–53. doi:10.1016/j.ymeth.2014.10.016

11. Avants, B. B., Libon, D. J., Rascovsky, K., Boller, A., McMillan, C. T., Massimo, L., Coslett, H. B., Chatterjee, A., Gross, R. G., and Grossman, M. "**Sparse Canonical Correlation Analysis Relates Network-Level Atrophy to Multivariate Cognitive Measures in a Neurodegenerative Population.**" *Neuroimage* 84, (2014): 698–711. doi:10.1016/j.neuroimage.2013.09.048, Available at http://dx.doi.org/10.1016/j.neuroimage.2013.09.048

12. Avants, B. B., Cook, P. A., Ungar, L., Gee, J. C., and Grossman, M. "**Dementia Induces Correlated Reductions in White Matter Integrity and Cortical Thickness: A Multivariate Neuroimaging Study with Sparse Canonical Correlation Analysis**" *Neuroimage* 50, no. 3 (2010): 1004–16. doi:10.1016/j.neuroimage.2010.01.041

13. Kim, J., Avants, B., Whyte, J., and Gee, J. C. "**Methodological Considerations in Longitudinal Morphometry of Traumatic Brain Injury**" *Front Hum Neurosci* 7, (2013): 52. doi:10.3389/fnhum.2013.00052

14. Dhillon, P. S., Wolk, D. A., Das, S. R., Ungar, L. H., Gee, J. C., and Avants, B. B. "**Subject-Specific Functional Parcellation via Prior Based Eigenanatomy.**" *Neuroimage* (2014): doi:10.1016/j.neuroimage.2014.05.026, Available at http://dx.doi.org/10.1016/j.neuroimage.2014.05.026

15. McMillan, C. T., Irwin, D. J., Avants, B. B., Powers, J., Cook, P. A., Toledo, J. B., McCarty Wood, E., Van Deerlin, V. M., Lee, V. M.-Y., Trojanowski, J. Q., and Grossman, M. "**White Matter Imaging Helps Dissociate Tau from TDP-43 in Frontotemporal Lobar Degeneration.**" *J Neurol Neurosurg*

Psychiatry 84, no. 9 (2013): 949–955. doi:10.1136/jnnp-2012-304418, Available at http://dx.doi.org/10.1136/jnnp-2012-304418

16. McMillan, C. T., Avants, B. B., Cook, P., Ungar, L., Trojanowski, J. Q., and Grossman, M. "**The Power of Neuroimaging Biomarkers for Screening Frontotemporal Dementia.**" *Hum Brain Mapp* (2014): doi:10.1002/hbm.22515, Available at http://dx.doi.org/10.1002/hbm.22515

17. McMillan, C. T., Toledo, J. B., Avants, B. B., Cook, P. A., Wood, E. M., Suh, E., Irwin, D. J., Powers, J., Olm, C., Elman, L., McCluskey, L., Schellenberg, G. D., Lee, V. M.-Y., Trojanowski, J. Q., Van Deerlin, V. M., and Grossman, M. "**Genetic and Neuroanatomic Associations in Sporadic Frontotemporal Lobar Degeneration.**" *Neurobiol Aging* 35, no. 6 (2014): 1473–1482. doi:10.1016/j.neurobiolaging.2013.11.029, Available at http://dx.doi.org/10.1016/j.neurobiolaging.2013.11.029