Structural MRI statistics using RMINC v1.4

Author:
Jason Lerch

May 4, 2016

Preface

This mini-book attempts to provide a general introduction to the statistics side of structural brain imaging, with a heavy emphasis on practical worked examples. It also introduces a particular toolkit, RMINC, designed to make running these types of statistical analyses easier. It is targeted at the general user, who may or may not have some statistical background, but does have some data they want analysed in a straightforward way. It is not meant to be a complete handbook on statistics, but hopefully will provide enough of a primer to get by, at least for a little while.

This mini-book exists for a number of reasons. I have over the years been asked multiple questions relating to structural brain imaging and statistics, and have had the chance to learn answers to those questions from countless people. This book thus exists as an attempt to put some of those answers down on paper. Secondly, writing this book is part and parcel of the development of RMINC; it is easier to write code useable by others if one documents it first, and then writes code to fit the documentation.

The book will likely be an incomplete work in progress for a long while yet. The LaTeXsource for this book are packaged along with RMINC itself, and contributors are most welcome!

Contents

Chapter 1

Introduction

The process of analysing brain imaging data is typically comprised of a series of stages. The study is designed with various choices made about the biology question that is to be addressed and the data necessary to answer the questions thus posed. Then the data is then acquired, and once that is completed, the images are processed in various automatic, semi-automatic, or manual ways and then analysed.

This book deals mainly with the final part, the data analysis, though there will be several side-tracks into the other topics. A single example will be used throughout: a mouse brain imaging study comparing male to female mouse brains. The methods described herein should be easily transferable to any other structural imaging study which looks at brain shape, tissue classification, or signal intensities.

1.1 Installing the tools

All the analyses will be performed using RMINC, which is a library designed to handle MINC volumes inside the R statistical environment. All the tools needed are freely available, and are designed to run on Debian/Ubuntu Linux and Mac OSX. Installation and setup is described in some more detail below.

1.1.1 R

R is a statistical programming languange that originated in the late 90's as an open source alternative to the S language of Bell Labs. Since then R has attracted a large community of open source developers and statisticians who continually work to improve R either by improving R itself or extending it with packages that provide extra functionality. R can be installed on almost all platforms, detailed instructions can be found on the R project's web page `https://www.r-project.org/`

To learn more about R there are many resources available on the internet. In print one could try Intoductory Statistics with R by Peter Dalgaard.

1.1.2 MINC

The Medical Imaging NetCDF file format is an effort to provide a unified imaging format for a variety of medical imaging needs at the Montreal Neurological Institute. Since, it has grown a robust suite of tools for nearly all computing tasks performed with medical imaging data.

The most ideal approach to begin working with MINC files is to acquire the minc toolkit (v2). This provides all the necessary infrastructure to manipulate minc files both for R and from the command line.

The toolkit can be installed according to the instructions `http://bic-mni.github.io/` in the section V2. Or built from source with code from `https://github.com/BIC-MNI/minc-toolkit-v2`.

1.1.3 RMINC

RMINC was created as an attempt to make performing statistical modelling with medical imaging data easy, specifically for Voxel-Based Morphometry discussed in more detail below. It was created by Jason Lerch, it merged with Jim Nikelski's MincIO package, and has been refined ever since by members of Montreal Neurological Institute and the Mouse Imaging Centre.

Installation of RMINC has been streamlined, in most cases it should suffice to run `install.packages("RMINC")`. If the minc-toolkit has been installed as recommended above RMINC should be able to find it automatically. If not RMINC will attempt to acquire the bare-bones C library libminc for you. More details can be found in the package's INSTALL file.

1.2 Overview of the analysis process

The data analysis process usually proceeds in the following way. First the input images are assessed for correctness; any obvious processing errors are removed from any subsequent analyses. The question of what constitutes an outlier is often a tricky one. In order to avoid the temptation to manipulate the data in a biased way it is best if the person who reviews the input data is blind about the categorization of each particular dataset.

Once all the acceptable datasets are in place a series of descriptive statistics can be generated, usually consisting of means and standard deviations of all images in the study as well as of all the subgroupings. This is followed by generating statistical maps of the main variables of interest. These are then thresholded for significance while taking multiple comparisons into account. There is then often a series of steps in which new statistical models are analyzed and thresholded until the results become more understandable. This usually involves lots of plotting of individual datapoints.

1.3 Data used throughout this book

This book will consistently work with one dataset consisting of 5 male and 5 female C57Bl/6 mice, taken from a larger dataset published in a 2007 NeuroImage paper by Spring et al. The mice, all 12 weeks old, were scanned using an overnight T2-weighted FSE sequence, then all aligned into a common space using an automated image registration algorithm (i.e. deformation based morphometry). The final metric of interest was then the Jacobian determinant of the deformations needed to align each mouse to the final common atlas. These details are relatively unimportant for this book - the input might as well be voxel density maps from VBM - but at least it gives some background for those who care. If you want to follow along with the examples used in this book you can download the data at `http://launchpad.net/rminc`. Note that the data has been downsampled to 120 micron voxels (from the original 32 micron voxels) to keep the download within reasonable limits.

Chapter 2

Preparing the data

This chapter will briefly discuss how to generate structural imaging data useable for the statistical analyses described in the rest of the book.

2.1 Types of datasets

Describe what can be done.

2.1.1 Voxel based morphometry

Some more detail on VBM.

2.1.2 Deformation based morphometry

Some more detail on DBM.

2.1.3 Other

Mention cortical thickness, manual segmentation, etc.

2.2 Input data

Once the files have been processed, the easiest way to proceed is by settting up a text file containing all the necessary information about each scan. This file should be comma or space separated, have one row per scan, with each column containing info about each scan. One of the columns should contain the filename pointing to the MINC volumes to be processed. The example from the five male and five female mice is the following:

```
Filename, Gender, coil, weight
volumes/img_08nov05.0-fwhm1.0.mnc,Female,1,22.6
```

```
volumes/img_08nov05.1-fwhm1.0.mnc,Female,2,19.6
volumes/img_08nov05.2-fwhm1.0.mnc,Female,3,21.8
volumes/img_29sept05.0-fwhm1.0.mnc,Male,1,24.0
volumes/img_29sept05.2-fwhm1.0.mnc,Male,3,27.0
volumes/img_30sept05.0-fwhm1.0.mnc,Male,1,28.3
volumes/img_30sept05.1-fwhm1.0.mnc,Male,2,26.5
volumes/img_30sept05.2-fwhm1.0.mnc,Male,3,28.1
volumes/img_31oct05.0-fwhm1.0.mnc,Female,1,20.5
volumes/img_31oct05.2-fwhm1.0.mnc,Female,3,20.0
```

Notice how the first row contains a header. This is optional, but makes later access to the data easier and is therefore recommended.

The next step is to actually load this file into R. The steps are given below:

```
options(warn=0)
library(RMINC)

## Loading required package: BatchJobs
## Loading required package: BBmisc
## Sourcing configuration file: '/micehome/chammill/R/x86_64-pc-linux-gnu-library/3.2/Batch
## BatchJobs configuration:
##   cluster functions: Interactive
##   mail.from:
##   mail.to:
##   mail.start: none
##   mail.done: none
##   mail.error: none
##   default.resources:
##   debug: FALSE
##   raise.warnings: FALSE
##   staged.queries: TRUE
##   max.concurrent.jobs: Inf
##   fs.timeout: NA

inputFile <- file.path(dataDir, "control-file.csv")
gf <- read.csv(inputFile)
```

The library commands load the RMINC library into R. The next line then reads the information describing this dataset from a comma-separated text file. The basic syntax of an R command is a variable name - which can be whatever you chose, within only a few limits - on the left hand side, the arrow (less than followed by a dash) indicating an assignment, and then the function call (in this case read.csv to read a comma separated value) with any arguments (in this case the filename) in parentheses. Strings - such as the filename in this case - are placed inside quotes.

```
xtable(gf, caption="GLIM File")
```

	Filename	Gender	coil	weight
1	volumes/img_08nov05.0-fwhm1.0.mnc	Female	1	22.60
2	volumes/img_08nov05.1-fwhm1.0.mnc	Female	2	19.60
3	volumes/img_08nov05.2-fwhm1.0.mnc	Female	3	21.80
4	volumes/img_29sept05.0-fwhm1.0.mnc	Male	1	24.00
5	volumes/img_29sept05.2-fwhm1.0.mnc	Male	3	27.00
6	volumes/img_30sept05.0-fwhm1.0.mnc	Male	1	28.30
7	volumes/img_30sept05.1-fwhm1.0.mnc	Male	2	26.50
8	volumes/img_30sept05.2-fwhm1.0.mnc	Male	3	28.10
9	volumes/img_31oct05.0-fwhm1.0.mnc	Female	1	20.50
10	volumes/img_31oct05.2-fwhm1.0.mnc	Female	3	20.00

Table 2.1: GLIM File

```
#Make filenames absolute with the data-dir (generally unnecessary)
gf$Filename <- file.path(dataDir, gf$Filename)
```

The little code fragment and table above just shows what the gf variable looks like after being read into R[1]. The last line adjusts the filenames so they can be found reproducibly on your system.

[1] xtable is only necessary for display purposes in this manual (which, by the way, is being written using knitr, a tool for combining R with latex).

Chapter 3

Descriptive statistics

Before going into explicit hypothesis tests it is often useful to get a general feel for what the data looks like - this is where descriptive statistics come in. The most common functions include computing the mean and variance or standard deviation at every voxel. If the data is inherently divided into groups, such as patients and control, or, in our example dataset, males versus females, then the descriptive stats can also be grouped by those variables.

To start we can look at the mean Jacobian determinant at every voxel of all the data combined:

```
setwd(dataDir)
overall.mean <- mincMean(gf$Filename)

## Method: mean
## Number of volumes: 10
## Volume sizes: 71 137 105
## N GROUPS: 1.000000
## In slice
##  0  1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24
## Done

overall.mean

## Multidimensional MINC volume
## Columns:
## [1] "/tmp/Rtmp4swY1O/RMINC_doc_data/volumes/img_08nov05.0-fwhm1.0.mnc"
```

The `mincMean` function computes the mean at every voxel of a set of filenames specified as an argument. The output is in this case assigned to the `overall.mean` variable. Repeating the variable in the R session, as done above, causes a summary to be printed.

One thing to note about the R syntax above: the dollar symbol is used to access a specific column inside a data frame. What this means is that inside

the `gf` variable - which, remember, is the variable that was read in from the comma-separated values file which describes the dataset - each column has a name which can be accessed by that dollar variable. Here are some examples, first showing the entire contents of `gf` and then two separate columns alone:

```
gf
```

```
##                                                                 Filename
## 1   /tmp/Rtmp4swY1O/RMINC_doc_data/volumes/img_08nov05.0-fwhm1.0.mnc
## 2   /tmp/Rtmp4swY1O/RMINC_doc_data/volumes/img_08nov05.1-fwhm1.0.mnc
## 3   /tmp/Rtmp4swY1O/RMINC_doc_data/volumes/img_08nov05.2-fwhm1.0.mnc
## 4  /tmp/Rtmp4swY1O/RMINC_doc_data/volumes/img_29sept05.0-fwhm1.0.mnc
## 5  /tmp/Rtmp4swY1O/RMINC_doc_data/volumes/img_29sept05.2-fwhm1.0.mnc
## 6  /tmp/Rtmp4swY1O/RMINC_doc_data/volumes/img_30sept05.0-fwhm1.0.mnc
## 7  /tmp/Rtmp4swY1O/RMINC_doc_data/volumes/img_30sept05.1-fwhm1.0.mnc
## 8  /tmp/Rtmp4swY1O/RMINC_doc_data/volumes/img_30sept05.2-fwhm1.0.mnc
## 9   /tmp/Rtmp4swY1O/RMINC_doc_data/volumes/img_31oct05.0-fwhm1.0.mnc
## 10  /tmp/Rtmp4swY1O/RMINC_doc_data/volumes/img_31oct05.2-fwhm1.0.mnc
##    Gender coil weight
## 1  Female    1   22.6
## 2  Female    2   19.6
## 3  Female    3   21.8
## 4    Male    1   24.0
## 5    Male    3   27.0
## 6    Male    1   28.3
## 7    Male    2   26.5
## 8    Male    3   28.1
## 9  Female    1   20.5
## 10 Female    3   20.0
```

```
gf$Filename
```

```
##  [1] "/tmp/Rtmp4swY1O/RMINC_doc_data/volumes/img_08nov05.0-fwhm1.0.mnc"
##  [2] "/tmp/Rtmp4swY1O/RMINC_doc_data/volumes/img_08nov05.1-fwhm1.0.mnc"
##  [3] "/tmp/Rtmp4swY1O/RMINC_doc_data/volumes/img_08nov05.2-fwhm1.0.mnc"
##  [4] "/tmp/Rtmp4swY1O/RMINC_doc_data/volumes/img_29sept05.0-fwhm1.0.mnc"
##  [5] "/tmp/Rtmp4swY1O/RMINC_doc_data/volumes/img_29sept05.2-fwhm1.0.mnc"
##  [6] "/tmp/Rtmp4swY1O/RMINC_doc_data/volumes/img_30sept05.0-fwhm1.0.mnc"
##  [7] "/tmp/Rtmp4swY1O/RMINC_doc_data/volumes/img_30sept05.1-fwhm1.0.mnc"
##  [8] "/tmp/Rtmp4swY1O/RMINC_doc_data/volumes/img_30sept05.2-fwhm1.0.mnc"
##  [9] "/tmp/Rtmp4swY1O/RMINC_doc_data/volumes/img_31oct05.0-fwhm1.0.mnc"
## [10] "/tmp/Rtmp4swY1O/RMINC_doc_data/volumes/img_31oct05.2-fwhm1.0.mnc"
```

```
gf$Gender
```

```
##  [1] Female Female Female Male   Male   Male   Male   Male
##  [9] Female Female
## Levels: Female Male
```

So if, in the text file that describes the dataset, the column containing all the filenames was called "jacobians", then the `mincMean` command would have been `mincMean(gf$jacobians)`. If an incorrect column is specified - i.e. something which does not contain filenames - then you should receive an error.

3.1 Writing results to file

Once the means at every voxel have been computed, they can be written to file. This is done with command below:

```
outDir <- tempdir()
outFilename <- "overall-mean.mnc"
outFullFilename <- file.path(outDir, outFilename)
mincWriteVolume(overall.mean, outFullFilename)

## Writing column 1 to file /tmp/Rtmp4swY1O/overall-mean.mnc
## Range: 0.084176 -0.108390
```

The `mincWriteVolume` command takes two arguments in the above example - the variable containing the data, and a string giving the filename to which the data should be written to. This MINC file can then be read and viewed with the standard MINC tools such as mincinfo, register, Display, etc.

3.2 Creating summaries by group

Most often we are more interested in how the means break down by the grouping in this dataset. This can be done by adding another variable to the mincMean call:

```
setwd(dataDir)
group.means <- mincMean(gf$Filename, gf$Gender)

## Method: mean
## Number of volumes: 10
## Volume sizes: 71 137 105
## N GROUPS: 2.000000
## In slice
##  0  1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24
## Done

group.means

## Multidimensional MINC volume
## Columns:       Female Male
## [1] "/tmp/Rtmp4swY1O/RMINC_doc_data/volumes/img_08nov05.0-fwhm1.0.mnc"
```

The *Gender* variable has two levels in it: *Male* and *Female*. So it will take the mean for all subjects in each group. These can then be written to file by specifiying the column.

```
fileOut <- file.path(tempdir(), "male-mean.mnc")
mincWriteVolume(group.means, fileOut, "Male")

## Writing column Male to file /tmp/Rtmp4swY1O/male-mean.mnc
## Range: 0.109804 -0.148817

#
fileOut <- file.path(tempdir(), "female-mean.mnc")
mincWriteVolume(group.means, fileOut, "Female")

## Writing column Female to file /tmp/Rtmp4swY1O/female-mean.mnc
## Range: 0.193389 -0.174766
```

If the difference between the two columns is of interest, one can just subtract the two data columns and then write out the result as a new minc volume.

```
setwd(dataDir)
difference <- group.means[,"Male"] - group.means[,"Female"]
mean(difference)
#
fileOut <- file.path(tempdir(), "diff.mnc")
likeFile <- gf$Filename[1]
mincWriteVolume(difference, fileOut, likeFile)
```

Notice how *mincWriteVolume* now needs a third argument: the name of a minc-file which has the same dimensions as the data. By default commands such as *mincMean* will store that information; after the subtraction above, however, the result is just a series of numbers with all metadata removed, so it has to be specified when writing the data to file.

Of course means are not the only items of interest. Also computable are the standard-deviations, variances, and sums, as illustrated below. Just like *mincMean* a column of filenames is required and a grouping variable is optional.

```
setwd(dataDir)
volVariance <- mincVar(gf$Filename, gf$Gender)

## Method: var
## Number of volumes: 10
## Volume sizes: 71 137 105
## N GROUPS: 2.000000
## In slice
##  0  1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24
## Done
```

```
volStdDev <- mincSd(gf$Filename)

## Method: var
## Number of volumes: 10
## Volume sizes: 71 137 105
## N GROUPS: 1.000000
## In slice
##  0  1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24
## Done

volSum <- mincSum(gf$Filename, gf$Gender)

## Method: sum
## Number of volumes: 10
## Volume sizes: 71 137 105
## N GROUPS: 2.000000
## In slice
##  0  1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24
## Done
```

Chapter 4

Linear Models

Linear models represent the mainstay of structural brain imaging. Their essence is quite simple: the data at every voxel is modelled by a set of terms corresponding to extra information about each scan. One can then perform hypothesis tests on each linear model to calculate the significance of either the entire model or even the marginal significance of each term in the model. This can be used to ask the question of, for example, in which voxels the gender of the subject predicts the values at that voxel.

This approach is also known as *massively univariate statistics* - i.e. a separate linear model is calculated at every voxel, resulting in thousands or even millions of separate models for every statistical test applied to the images. The last step in analysing such data is thus often to account for these thousands of comparisons so that the results do not occur just by random chance.

4.1 First linear model

Let's start with a simple linear model - lets see where the Jacobian determinants contained in the files used in the male-female dataset are best modelled by the gender of the mouse.

```
setwd(dataDir)
vs <- mincLm(Filename ~ Gender, gf)

## Method: lm
## Number of volumes: 10
## Volume sizes: 71 137 105
## N: 10 P: 2
## In slice
##  0  1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24
## Done

print(vs)
```

```
## Multidimensional MINC volume
## Columns:       F-statistic R-squared beta-(Intercept) beta-GenderMale tvalue-(Intercept)
## [1] "/tmp/Rtmp4swY1O/RMINC_doc_data/volumes/img_08nov05.0-fwhm1.0.mnc"

#
fileOut <- file.path(tempdir(), "simple-lm.mnc")
mincWriteVolume(vs, fileOut, "tvalue-GenderMale")

## Writing column tvalue-GenderMale to file /tmp/Rtmp4swY1O/simple-lm.mnc
## Range: 9.581574 -8.354640
```

mincLm is the command to run linear models in RMINC. Its basic use is to provide a formula (same syntax as the R `lm` command) with the left side containing the filenames, the right side the variables to be regressed. The output of mincLm depends on the formula. There will always be a column of F-statistics, representing the significance of the entire model. Then there is one column for each of the terms in the model. The above linear model, relating the Jacobian determinant to gender, will thus have three columns:

F-statistic representing the significance of the entire model.

(Intercept) the intercept term - this term is rarely interesting, as it tests for whether the intercept is 0. There's no reason to believe it should be in most cases, so this value will be highly significant but meaningless.

GenderMale the term testing whether the "Male" level of the Gender factor is significant. In this case this term is the most interesting and therefore the one written to file.

The output is placed into a variable that can be written to file in the same way as described in the descriptive statistics section.

4.2 Plotting voxels

```
setwd(dataDir)
options(show.signif.stars=FALSE)
voxel <- mincGetVoxel(gf$Filename, 44, 20, 52)
summary(lm(voxel ~ Gender, gf))

##
## Call:
## lm(formula = voxel ~ Gender, data = gf)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
```

```
## -0.13849 -0.04947  0.01239  0.04120  0.11775
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.01900    0.03443   0.552    0.596
## GenderMale  -0.08765    0.04869  -1.800    0.110
##
## Residual standard error: 0.07699 on 8 degrees of freedom
## Multiple R-squared:  0.2883,Adjusted R-squared:  0.1993
## F-statistic: 3.241 on 1 and 8 DF,  p-value: 0.1095

vs[635093,]

##        F-statistic          R-squared   beta-(Intercept)
##         3.24086174         0.28831079         0.01899564
##    beta-GenderMale tvalue-(Intercept)  tvalue-GenderMale
##        -0.08765460         0.55172722        -1.80023936
```

The code above does the following: it gets the voxel from coordinates 44, 20, 52 for all subjects, then computes a linear model relating that voxel to Genotype using standard R functions. Lastly it prints the results from that same voxel as computed by `mincLm`[1]. This helps illustrate what the output of `mincLm` stores: the F-statistic is the same as can be found in the last line of the summary command, and the t-statistics for the Intercept and Genotype column can be found under "t-value" when using standard R functions.

`mincGetVoxel` needs three coordinates, given in voxel space in the same order as stored in the file. Just printing the voxel will show the corresponding world coordinates:

```
print(voxel)

##  [1]  0.13674168  0.04150733  0.07639956 -0.03271777
##  [5] -0.12701763 -0.06638037 -0.09147905 -0.02570000
##  [9] -0.04017891 -0.11949146
## attr(,"class")
## [1] "mincVoxel" "vector"
## attr(,"filenames")
##  [1] "/tmp/Rtmp4swY1O/RMINC_doc_data/volumes/img_08nov05.0-fwhm1.0.mnc"
##  [2] "/tmp/Rtmp4swY1O/RMINC_doc_data/volumes/img_08nov05.1-fwhm1.0.mnc"
##  [3] "/tmp/Rtmp4swY1O/RMINC_doc_data/volumes/img_08nov05.2-fwhm1.0.mnc"
##  [4] "/tmp/Rtmp4swY1O/RMINC_doc_data/volumes/img_29sept05.0-fwhm1.0.mnc"
##  [5] "/tmp/Rtmp4swY1O/RMINC_doc_data/volumes/img_29sept05.2-fwhm1.0.mnc"
##  [6] "/tmp/Rtmp4swY1O/RMINC_doc_data/volumes/img_30sept05.0-fwhm1.0.mnc"
```

[1]The actual number indexed here - 635093 - might appear odd. RMINC treats all MINC volumes as 1-dimensional arrays, so the actual index has to be computed by the following formula: $(index_1 * size_2 + index_2) + size_3 + index_3$

```
##  [7] "/tmp/Rtmp4swY1O/RMINC_doc_data/volumes/img_30sept05.1-fwhm1.0.mnc"
##  [8] "/tmp/Rtmp4swY1O/RMINC_doc_data/volumes/img_30sept05.2-fwhm1.0.mnc"
##  [9] "/tmp/Rtmp4swY1O/RMINC_doc_data/volumes/img_31oct05.0-fwhm1.0.mnc"
## [10] "/tmp/Rtmp4swY1O/RMINC_doc_data/volumes/img_31oct05.2-fwhm1.0.mnc"
## attr(,"voxelCoord")
## [1] 44 20 52
## attr(,"worldCoord")
## [1] -0.03 -5.79  1.08
```

If the coordinates are specified in world coordinates then `mincGetWorldVoxel` is what you want - it also takes three coordinates, this time in world space in xspace,yspace,zspace order:

```
world.voxel <- mincGetWorldVoxel(gf$Filename, -3.6, -3.9, -1.5)
world.voxel
```

```
##  [1] -0.089431985 -0.134077317 -0.284454840  0.027110817
##  [5]  0.119694881  0.010154088  0.003454434  0.130681203
##  [9] -0.149013026 -0.118182420
## attr(,"class")
## [1] "mincVoxel" "vector"
## attr(,"filenames")
##  [1] "/tmp/Rtmp4swY1O/RMINC_doc_data/volumes/img_08nov05.0-fwhm1.0.mnc"
##  [2] "/tmp/Rtmp4swY1O/RMINC_doc_data/volumes/img_08nov05.1-fwhm1.0.mnc"
##  [3] "/tmp/Rtmp4swY1O/RMINC_doc_data/volumes/img_08nov05.2-fwhm1.0.mnc"
##  [4] "/tmp/Rtmp4swY1O/RMINC_doc_data/volumes/img_29sept05.0-fwhm1.0.mnc"
##  [5] "/tmp/Rtmp4swY1O/RMINC_doc_data/volumes/img_29sept05.2-fwhm1.0.mnc"
##  [6] "/tmp/Rtmp4swY1O/RMINC_doc_data/volumes/img_30sept05.0-fwhm1.0.mnc"
##  [7] "/tmp/Rtmp4swY1O/RMINC_doc_data/volumes/img_30sept05.1-fwhm1.0.mnc"
##  [8] "/tmp/Rtmp4swY1O/RMINC_doc_data/volumes/img_30sept05.2-fwhm1.0.mnc"
##  [9] "/tmp/Rtmp4swY1O/RMINC_doc_data/volumes/img_31oct05.0-fwhm1.0.mnc"
## [10] "/tmp/Rtmp4swY1O/RMINC_doc_data/volumes/img_31oct05.2-fwhm1.0.mnc"
## attr(,"worldCoord")
## [1] -3.6 -3.9 -1.5
## attr(,"voxelCoord")
## [1] 22 36 22
```

4.3 Creating images

RMINC has the ability to call ray_trace to create images of individual slices corresponding to a specific voxel. For this to work ray_trace[2] has to be installed and present in the path, as does MICe-minc-tools[3].

[2] `http://packages.bic.mni.mcgill.ca`
[3] `http://wiki.phenogenomics.ca:8080/display/MICePub/MICe-minc-tools`

An example is given below - this will create an image of the slice corresponding to the voxel location along with a cross-hair over that voxel. The output can be seen in figure *ray-trace-img*, along with a box-and-whiskers plot of the data at that voxel.

```
mincRayTraceStats(voxel, file.path(volDir, "anatomy.mnc"),
                  vs, "GenderMale", image.min=350000, image.max=1.0e+06,
                  display=F)
```

The `mincRayTraceStats` function needs the following arguments: a voxel, obtained by mincGetVoxel or mincGetWorldVoxel, the path towards a MINC image containing some background anatomy, the output of mincLm (vs in this case), and minimum and maximum values of the background anatomy.

4.4 Using subsets

It is quite common to want to run a linear model on only a subset of the data. This can be quite easily accomplished in `mincLm` using an extra subsetting specification:

```
setwd(dataDir)
vs <- mincLm(Filename ~ Gender, gf, coil==1)

## Method: lm
## Number of volumes: 4
## Volume sizes: 71 137 105
## N: 4 P: 2
## In slice
##  0  1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24
## Done

vs

## Multidimensional MINC volume
## Columns:       F-statistic R-squared beta-(Intercept) beta-GenderMale tvalue-(Intercept)
## [1] "/tmp/Rtmp4swY1O/RMINC_doc_data/volumes/img_08nov05.0-fwhm1.0.mnc"
```

This is the same linear model command as executed above, but this time using only use mice scanned on RF coil number 1. The subset command works exactly the same way as for the standard `lm` command from R.

4.5 Multiple Comparisons

The example below illustrates the entire process involved in running a linear model and correcting for multiple comparisons using the False Discovery Rate.

```
setwd(dataDir)
vs <- mincLm(Filename ~ Gender, gf)

## Method: lm
## Number of volumes: 10
## Volume sizes: 71 137 105
## N: 10 P: 2
## In slice
##  0  1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24
## Done

print(vs)

## Multidimensional MINC volume
## Columns:       F-statistic R-squared beta-(Intercept) beta-GenderMale tvalue-(Intercept)
## [1] "/tmp/Rtmp4swY1O/RMINC_doc_data/volumes/img_08nov05.0-fwhm1.0.mnc"

#
qvals <- mincFDR(vs, mask = file.path(volDir, "mask.mnc"), method="pFDR")

##
## Computing FDR threshold for all columns
## Start: 0 0 0
## Count: 71 137 105
##   Computing threshold for  F-statistic
##   Computing threshold for  tvalue-(Intercept)
##   Computing threshold for  tvalue-GenderMale

print(qvals)

## Multidimensional MINC volume
## Columns:       F-statistic tvalue-(Intercept) tvalue-GenderMale
## [1] "/tmp/Rtmp4swY1O/RMINC_doc_data/volumes/img_08nov05.0-fwhm1.0.mnc"
## Degrees of Freedom: c(1, 8) 8 8
## FDR Thresholds:
##      F-statistic tvalue-(Intercept) tvalue-GenderMale
## 0.01          NA                NA                NA
## 0.05          NA                NA                NA
## 0.1           NA                NA                NA
## 0.15    8.775918                NA          2.962418
## 0.2     5.352080                NA          2.313456

#
fileOut <- file.path(tempdir(), "Gender-FDR.mnc")
mincWriteVolume(qvals, fileOut, "tvalue-GenderMale")

## Writing column tvalue-GenderMale to file /tmp/Rtmp4swY1O/Gender-FDR.mnc
## Range: 1.000000 0.119666
```

The first command computes a linear model using `mincLm`. The results are then passed on to `mincFDR`, which computes the False Discovery Rate threshold separately for each of the terms in the linear model. Only results from within the mask specified as an optional argument to `mincFDR` are considered. The thresholds detected at different levels (0.01, 0.05, 0.10, 0.15, and 0.20) are then printed out. In this example seen above there is no data at a FDR level of 0.01, 0.05, or 0.10, but any t-statistic greater than 2.96 (or less than -2.96) would be significant at a 15% false positive level - i.e. 15% of the voxels above that threshold would be, on average, false positives. The "GenderMale" column is then written to file. Note that we use the positive false discovery rate above - the more standard false discovery rate is the default (i.e. if no method argument is specified for `mincFDR`).

You can also compute the FDR from a volume that has already been written to file:

```
setwd(dataDir)
volname <-file.path(tempdir(), "simple-lm.mnc")
volume <- mincGetVolume(volname)

## Start: 0 0 0
## Count: 71 137 105

#
qvals <- mincFDR(volume, df=21, statType="t")

##   Computing threshold for  1

## Warning:  replacing previous import by 'grid::arrow' when loading
'qvalue'
## Warning:  replacing previous import by 'grid::unit' when loading
'qvalue'

print(qvals)

## Multidimensional MINC volume
## Columns:       1
## [1] "/tmp/Rtmp4swY1O/simple-lm.mnc"
## Degrees of Freedom: 21
## FDR Thresholds:
##             1
## 0.01 4.189899
## 0.05 3.003183
## 0.1  2.407636
## 0.15 2.018176
## 0.2  1.746950
```

Notice how in this case you have to specify the degrees of freedom, since the

information associated with the linear model is lost after writing to file. Also note that the `statType` argument is required to inform the FDR functions that the FDR is to be computed on t-statistics (options are "t", and "F").

Chapter 5

Bits and Pieces

5.1 Correlations

5.2 non-parametric statistics

5.3 mixed effects models

5.4 Cleaning up

Chapter 6

Advanced Topics

6.1 Running arbitrary R functions

RMINC has the capacity to run arbitrary R functions at ever voxel of a set of files. This comes in quite handy when there are no easily wrapped functions that exist in RMINC but there is some existing R module you would like to try out. Some words of caution are in order, however:

- Running arbitrary functions may involve writing your own small R function to wrap the code you want to use, which is a bit ugly.
- It is slow. Slower than molasses on a cold Georgia winter morning.

Here's an example of how it works:

```
f <- function() {
  return(tapply(x, gf$Genotype, mean))
}
vs <- mincApply(gf$jacobians, quote(f()), mask="small-mask.mnc")
```

The code above has two parts. The first is creating the function, which has the following properties:

- It takes no arguments.
- It works on the variable x. This function will be evaluated at every voxel, where x will be a vector containing the values at that voxel for all the files.
- It returns a vector.

An updated alternative that does not require that you wrap your function as a single argument function is mincApplyRCPP

```
vs <- mincApplyRCPP(gf$jacobians, tapply, INDEX = gf$Genotype, FUN = mean)
```

`mincApplyRCPP` allows you to pass additional arguments to your function of interest, in this case INDEX and FUN, via the `...` interface. For safety it is highly recommended you supply arguments by name. It is worth reading `?mincApplyRCPP` for additional warnings regarding the destructive binding of `...` arguments.

An additional convenience provided by `mincApplyRCPP` is the type conversion argument. To provide added flexibility, `mincApplyRCPP` gives you the option to caste your results into a format useful to you. By default the argument `collate` is set to `simplify2minc` which attempts to intelligently caste your results as a known RMINC object. This argument can take any function, it is applied to the list of voxel results produced internally. Example casting functions are `identity` which returns the result list unaltered, and `dplyr::bind_rows` which will try to collapse the results into a data.frame.

Index