Journal of Statistical Software

October 2011, Volume 44, Issue 8. *http://www.jstatsoft.org/*

TractoR: Magnetic Resonance Imaging and Tractography with R

Jonathan D. Clayden
University College London

Susana Muñoz Maniega
University of Edinburgh

Amos J. Storkey
University of Edinburgh

Martin D. King
University College London

Mark E. Bastin
University of Edinburgh

Chris A. Clark
University College London

Abstract

Statistical techniques play a major role in contemporary methods for analyzing magnetic resonance imaging (MRI) data. In addition to the central role that classical statistical methods play in research using MRI, statistical modeling and machine learning techniques are key to many modern data analysis pipelines. Applications for these techniques cover a broad spectrum of research, including many preclinical and clinical studies, and in some cases these methods are working their way into widespread routine use.

In this manuscript we describe a software tool called **TractoR** (for "Tractography with R"), a collection of packages for the R language and environment, along with additional infrastructure for straightforwardly performing common image processing tasks. **TractoR** provides general purpose functions for reading, writing and manipulating MR images, as well as more specific code for fitting signal models to diffusion MRI data and performing tractography, a technique for visualizing neural connectivity.

Keywords: magnetic resonance imaging, diffusion, tractography, image processing, machine learning, R.

1. Introduction

Magnetic resonance imaging (MRI) is a noninvasive medical imaging technique which is used routinely in hospitals worldwide. By exploiting fundamentally quantum-mechanical characteristics of water molecules in the presence of a strong magnetic field, MRI can recover detailed images of body tissues at a resolution of around 1 mm. Unlike many other medical imaging methods, MRI uses no ionizing radiation, and therefore permits repeated scanning.

`tractor`	Top-level managed directory
`tractor/fdt`	Data, brain mask and DTI maps (Section 4.1)
`tractor/fdt.bedpostX`	Data processed by **FSL** BEDPOST (Section 4.2)
`tractor/fdt.track`	Tractography output (Section 5)
`tractor/camino`	Files for interoperability with Camino
`tractor/objects`	Binary R objects

Table 1: The layout of a session directory in **TractoR** version 1.8.0.

MRI is also a very flexible technique. By manipulating the nuclear magnetic resonance signal from water molecules, a broad variety of tissue contrasts can be obtained. Of particular interest in this paper is diffusion MRI (dMRI), wherein image intensity at each pixel (or voxel for 3D images) depends on the local pattern of self-diffusion of water (Le Bihan 2003). Since elements of body tissue such as cell membranes present obstacles to diffusion, these images can provide information about microscopic tissue structure; and the orientation of highly linear structures such as the brain's white matter can also be inferred.

The development of new MRI analysis techniques is a major area of research, and some of the ideas generated by this research are filtering through to clinical applications. Statistical techniques play a very important part in many of these new developments. Currently, MATLAB is probably the most widely-used programming language for the development of new MRI analysis tools, with the **SPM** (statistical parametric mapping) package being a notable example (Friston *et al.* 2007); but C++, Java and Python are also common choices. Due to its statistical pedigree and vectorized programming model, R (R Development Core Team 2011) is a strong candidate for software development in this field.

In this paper we introduce a software package called **TractoR** (for "Tractography with R"), which consists of several R packages, along with a simple command line based front-end and some associated infrastructure for performing common MRI analysis tasks. **TractoR** is free software, available from `http://code.google.com/p/tractor` under the terms of the GNU General Public License, version 2.

2. Package overview and conventions

The core functionality of **TractoR** is provided in four R packages. The **tractor.base** package provides general functions for reading, writing and manipulating magnetic resonance images; **tractor.session** maintains the various image files associated with a particular scan session or subject, and provides interfaces to third-party software; **tractor.nt** provides functions for performing "neighbourhood tractography"; and **tractor.utils** provides various utility functions, primarily used by the **TractoR** shell interface.

The session abstraction obviates the need for the user to keep track of the locations of large numbers of files, instead following a convention for simplicity. The general layout in the current version of **TractoR** is shown in Table 1, and the **tractor.session** package provides accessor functions for obtaining the true path to a particular image, such as the fractional anisotropy map (located at `tractor/fdt/dti_FA`). Note that since this is an abstraction, the layout of session directories may change in a future version of the software.

Classes in **TractoR** are implemented as lists of functions, defined in a scope in which a set

of member variables are defined. These variables are not directly accessible externally, but accessor functions provide access to them. This will be demonstrated explicitly in the next section.

3. The MriImage class

The core data type used in **TractoR** is the `MriImage`. This class is defined in the **tractor.base** R package, along with methods for reading and writing objects from and to standard medical image file types; visualization; and simple image manipulation. This data type consists of an array of voxel intensity values along with a set of metadata describing the voxel dimensions, coordinate origin and storage format of the image.

The standard file format used by **TractoR** to store `MriImage` objects is the NIfTI-1 format, a widely-supported standard in medical imaging (http://nifti.nimh.nih.gov/nifti-1)[1]. Reading an image from such a file is simple.

```
R> library("tractor.base")
R> i <- newMriImageFromFile(file.path(Sys.getenv("TRACTOR_HOME"), "share",
+    "mni", "brain.nii.gz"))
```

(Note that this example requires that the full **TractoR** distribution has been installed, and that the `TRACTOR_HOME` environment variable has been correctly set to point to its location on disk.) This particular image is a standard representation of the brain in Montréal Neurological Institute space (MNI space; see Evans *et al.* 1994), provided by the International Consortium for Brain Mapping. Information about the image can be obtained through its `print()` method.

```
R> print(i)
```

```
* * * * INFO: Image source     : /usr/local/tractor/share/mni/brain
* * * * INFO: Image dimensions : 91 x 109 x 91 voxels
* * * * INFO: Coordinate origin: (46,64,37)
* * * * INFO: Voxel dimensions : 2 x 2 x 2 mm
* * * * INFO: Data type        : unsigned integer, 8 bits/voxel
* * * * INFO: Endianness       : little
```

Access to the array of voxel values stored with the image, or specific elements of metadata, is through a series of accessor functions.

```
R> d <- i$getData()
R> class(d)
```

```
[1] "array"
```

```
R> dim(d)
```

```
[1]  91 109  91
```

[1] **TractoR** will also support the upcoming NIfTI-2 file format.

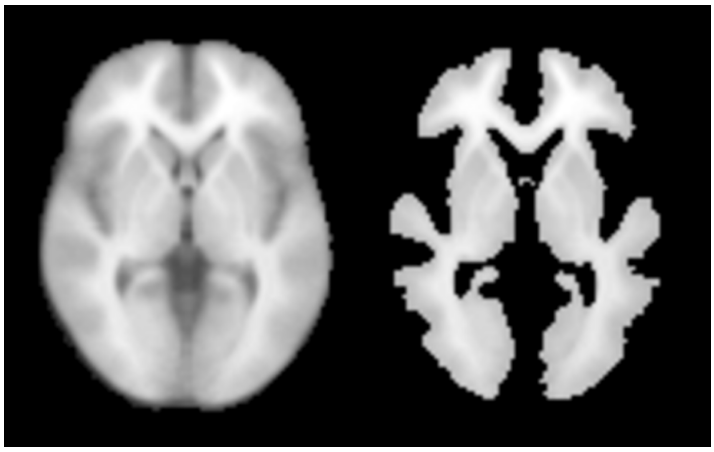

Figure 1: One slice of the standard MNI space brain image, before (left) and after (right) thresholding at a voxel intensity of 150.

```
R> i$getOrigin()

[1] 46 64 37
```

Additional accessor functions for the `MriImage` class – or, analogously, any other class – may be obtained by calling the R `names()` function with an object of the relevant class as a parameter.

```
R> names(i)

 [1] "getData"            "getDataAtPoint"     "getMetadata"
 [4] "getDataType"        "getDimensionality"  "getDimensions"
 [7] "getEndianness"      "getFieldOfView"     "getOrigin"
[10] "getSource"          "getVoxelDimensions" "getVoxelUnit"
[13] "isInternal"         "setEndianness"      "setSource"
[16] "summarise"          "summarize"
```

Slices of an image, maximum intensity projections and "contact sheet" style visualizations of all image slices can be easily created, with the aspect ratio of the image set according to the voxel dimensions. Simple image processing operations such as applying a lower intensity threshold can also be applied, creating new `MriImage` objects.

```
R> createSliceGraphic(i, z = 37)
R> j <- newMriImageByThresholding(i, 150)
R> createSliceGraphic(j, z = 37)
```

The plane of the graphic is set by the option `z = 37`, indicating the slice perpendicular to the z-axis (bottom-to-top), 37 voxels from the bottom of the brain volume. The two graphics resulting from the code above may be seen in Figure 1.

New images may also be created by applying an arbitrary function to an existing image, or a pair of images, as in the examples below.

```
R> k <- newMriImageWithSimpleFunction(i, function (x) x^2)
R> l <- newMriImageWithBinaryFunction(i, j, "*")
```

Multiplying images, as in the latter example, may be achieved more succinctly using the standard operator, as in `i * j`.

DICOM (Digital Imaging and Communications in Medicine; see http://dicom.nema.org/) is an extremely complex standard for storing and transferring medical imaging information, and is the format in which image data is usually obtained from an MRI scanner. An `MriImage` object may be created from a DICOM file – or a directory of related DICOM files where each file represents a slice of a larger image – and subsequently converted to NIfTI-1 format.

```
R> i <- newMriImageFromDicom(file.path(Sys.getenv("TRACTOR_HOME"), "tests",
+    "data", "dicom", "01.dcm"))
R> print(i)

* * * * INFO: Image source     : /usr/local/tractor/tests/data/dicom/01.dcm
* * * * INFO: Image dimensions : 224 x 256 voxels
* * * * INFO: Coordinate origin: (1,1)
* * * * INFO: Voxel dimensions : 1 x 1 mm
* * * * INFO: Data type        : unsigned integer, 16 bits/voxel
* * * * INFO: Endianness       : little

R> writeMriImageToFile(i, "image", "NIFTI")
```

The output file will be called `image.nii`.[2]

Alternatively, a DICOM file may be read into a `DicomMetadata` object, which retains all of the information stored in the file. Individual DICOM "tags", containing information about the scan acquisition, may then be extracted.

```
R> m <- newDicomMetadataFromFile(file.path(Sys.getenv("TRACTOR_HOME"),
+    "tests", "data", "dicom", "01.dcm"))
R> m$getTagValue(0x0018, 0x0087)

[1] 1.494

R> getDescriptionForDicomTag(0x0018, 0x0087)

[1] "Magnetic Field Strength"
```

Finally, an `MriImage` may be created from a standard R array object, by associating it with an `MriImageMetadata` object. The latter contains all of the information about an `MriImage`, but not the actual voxel values. In the following example we create a mask image whose value is 1 wherever the original image is positive and 0 everywhere else. This could be more simply achieved using `newMriImageWithSimpleFunction`, but the longer version is instructive.

[2] A gzipped NIfTI file, or image/header pair, may be obtained by substituting `"NIFTI_GZ"` or `"NIFTI_PAIR"` as the last argument to `writeMriImageToFile`.

```
R> i <- newMriImageFromFile(file.path(Sys.getenv("TRACTOR_HOME"), "share",
+    "mni", "brain.nii.gz"))
R> d <- array(as.integer(i$getData() > 0), dim = i$getDimensions())
R> m <- newMriImageMetadataFromTemplate(i$getMetadata(),
+    datatype = getDataTypeByNiftiCode(2))
R> j <- newMriImageWithData(d, m)
```

The first line here reads the original image, and the second binarises it. The third line creates an `MriImageMetadata` object which is identical to that associated with the original image, except that its data type is changed to save disk space[3]. The final line creates the new image object.

4. Modeling the diffusion-weighted signal

The **tractor.session** package for R defines the `MriSession` class, which encapsulates a single scanning session with a single subject, and manages a hierarchy of image files and other information which relate to that scan. Since several of these files are involved in performing most data processing tasks, the abstraction obviates the necessity to specify each of these files individually – rather, only the top-level directory containing the session data need be given explicitly.

Some initial preprocessing is required to move from a set of raw data files acquired from an MRI scanner to a data set ready for fiber tracking as laid out below, and **tractor.session** provides tools to perform that preprocessing, but those steps are omitted here for brevity.[4]

4.1. The diffusion tensor

Diffusion MRI data are usually acquired as a series of 3D volumes, each of which has an associated diffusion sensitizing magnetic gradient applied along a particular direction relative to the scanner's native coordinate system, represented as a normalized column vector, G. Under the assumption that the spatial distribution of diffusing water molecules after a particular time, t, is 3D Gaussian, the relationship between the signal amplitude in the presence of diffusion sensitization, S, and the signal without it, S_0, is

$$\frac{S(b,\mathsf{G})}{S_0} = \exp\left(-b\mathsf{G}^\mathsf{T}\mathsf{D}\mathsf{G}\right) \ . \tag{1}$$

Here, b is the diffusion "weighting factor", which depends on the diffusion time and the strength of the magnetic gradient applied. D is known as the "effective diffusion tensor", represented relative to the scanner's coordinate system, and is related to the covariance matrix of the 3D Gaussian molecular displacement distribution by $\Sigma = 2\mathsf{D}t$. Using Equation 1, the elements of the diffusion tensor matrix may be estimated in each voxel of the brain from the log-transformed signal intensities with and without diffusion weighting using ordinary or weighted least-squares estimation, or more sophisticated methods where required. The combination of

[3] The meanings of the various NIfTI data type codes can be found from the variable `.Nifti$datatypes`.

[4] The `preproc` script, inside the full **TractoR** distribution's `share/experiments` subdirectory, demonstrates how each of the standard preprocessing steps can be achieved. They are: conversion of DICOM files to NIfTI-1 format if required (as described in Section 3); coregistration of diffusion-weighted volumes; and brain extraction.

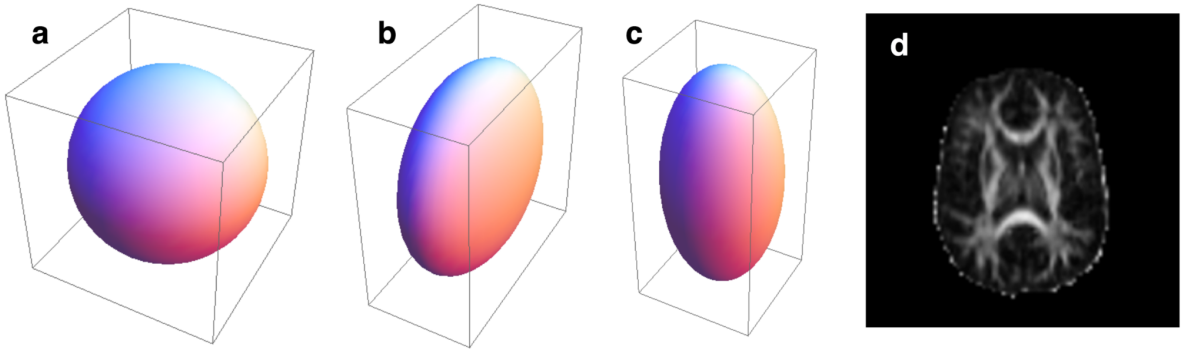

Figure 2: Ellipsoids representing isotropic (a), oblate (b) and prolate (c) diffusion profiles, along with a map of fractional anisotropy in a slice of the human brain (d).

dMRI acquisition with diffusion tensor estimation is referred to as diffusion tensor imaging (Basser *et al.* 1994).

With the diffusion tensor estimated at each voxel in the brain, it is common for many applications to calculate the eigenvalues, λ_1, λ_2 and λ_3, of each tensor, along with various derived quantities. The eigenvalues represent effective diffusivities, generally specified in $\mathrm{mm}^2\ \mathrm{s}^{-1}$, along the principal axes of the displacement distribution. From these are typically calculated measures such as mean diffusivity (MD), the mean of the eigenvalues, $\bar{\lambda}$, and fractional anisotropy (FA), given by

$$\mathrm{FA} = \sqrt{\frac{3}{2}} \sqrt{\frac{(\lambda_1 - \bar{\lambda})^2 + (\lambda_2 - \bar{\lambda})^2 + (\lambda_3 - \bar{\lambda})^2}{\lambda_1^2 + \lambda_2^2 + \lambda_3^2}} \,. \tag{2}$$

Considering the isosurface of probability density after a given diffusion time as an ellipsoid, MD describes the volume of the ellipsoid while FA describes its shape. FA is zero for a spherical ellipsoid, representing isotropic diffusion, while it is close to unity when one eigenvalue is substantially larger than the others, corresponding to a prolate ellipsoid (Figure 2a–c). The white matter of the brain is highly linearized, and FA is consequently higher in these regions (Figure 2d).

Diffusion tensor fitting may be performed with **TractoR** using either ordinary least-squares or iterative weighted least-squares approaches. The latter is recommended due to heteroskedasticity in the log-transformed signal intensities (Salvador *et al.* 2005). It may be performed as follows.

```
R> library("tractor.session")
R> s <- newSessionFromDirectory(file.path(Sys.getenv("TRACTOR_HOME"),
+    "tests", "data", "session-12dir"))
R> createDiffusionTensorImagesForSession(s, method = "iwls")
```

This command will use preprocessed data files already stored in the specified session directory to estimate diffusion tensors at each image voxel, and write out a series of images representing various quantities derived from it or its diagonalization, including MD and FA.[5] Ordinary

[5]Be aware that running this command on the test data supplied with **TractoR** exactly as given will overwrite some standard test files, and may lead to some of **TractoR**'s self tests failing afterwards. It is therefore preferable to use a copy of the test data, or a different data set, when experimenting.

least-squares tensor fitting may be performed by substituting `method="ls"` into the last command. The full path to any particular image created within the session directory in this way may be obtained as follows.

```
R> s$getImageFileNameByType("FA")

[1] "/usr/local/tractor/tests/data/session-12dir/tractor/fdt/dti_FA"
```

An interface to the tensor estimation tool in the FMRIB Software Library (**FSL**; see `http://www.fmrib.ox.ac.uk/fsl/` and Smith *et al.* 2004) may be used as an alternative, for its somewhat faster speed, as follows.

```
R> runDtifitWithSession(s)
```

In this case only the ordinary least-squares method is used, and **FSL** must be installed on the user's system as well as **TractoR**.

We note that other diffusion signal models, such as a Gaussian mixture model or the so-called "q-ball" model, can be fitted to dMRI data using the **dti** R package (Polzehl and Tabelow 2011).

4.2. A sampling-based approach

The assumption of a 3D Gaussian molecular displacement distribution for self-diffusion of water in living tissue is oversimplistic. Even in highly linear white matter, more complex patterns of diffusion occur regularly at the crossings of multiple pathways (Jones 2008). For the purpose of fibre tracking, discussed in the next section, it is therefore usually desirable to allow for contributions from multiple fibre populations with different orientations. One such generalisation, described by Behrens *et al.* (2007) and often referred to as the "ball and sticks" model, treats the displacement distribution as a mixture of pure isotropic and anisotropic components. Assuming that the orientation of the ith fibre population is given by the column vector N_i, the signal model then becomes

$$\frac{S(b,\mathsf{G})}{S_0} = \left(1 - \sum_i f_i\right) \exp(-bD) + \sum_i f_i \exp\left(-bD\left(\mathsf{G}^\mathsf{T}\mathsf{N}_i\right)^2\right) \ , \tag{3}$$

where $f_i \in [0,1]$ is the volume fraction of the ith anisotropic component, and D is the effective diffusivity.

In a standard dMRI acquisition, the known values in Equation 3 are b and G, while S and S_0 are observed, and D, (f_i) and (N_i) are to be estimated. Rather than calculate point estimates of these parameters, Behrens *et al.* (2007) put forward a Markov chain Monte Carlo (MCMC) approach to estimating their posterior distributions.[6] They refer to their algorithm as BEDPOST (Bayesian estimation of diffusion parameters obtained using sampling techniques). The algorithm is provided as part of **FSL**, and **TractoR** provides a simple interface to it for R, viz.

[6]The priors used by the algorithm as published are uninformative. Noise in each voxel is modelled to be independent and identically distributed Gaussian, with a Gamma distribution prior on the precision parameter.

```
R> runBedpostWithSession(s, nFibres = 2)
```

Here, the `nFibres` argument determines the maximum number of anisotropic components allowed per voxel; i.e., the upper limit on i in the model in Equation 3. BEDPOST uses the automatic relevance determination framework on the volume fraction parameters, (f_i), to set priors which allow the weights of components to be forced to zero where they are irrelevant for predicting the signal (see MacKay 1995). In this way, only as many components are maintained in each voxel as are supported by the data. BEDPOST typically takes several hours to run, at the end of which a number of new files are created within the session directory, representing the estimated distributions for each parameter of interest – notably the orientation vectors, (N_i).

5. Fibre tracking

Diffusion fibre tracking, or "tractography", is the process of reconstructing white matter tract trajectories by following the local orientation information estimated as described in Section 4. It has a wide spectrum of applications in clinical imaging and neuroscience.

5.1. Tracking from single seed points

The standard algorithm for performing streamline tractography from a single seed point may be described as follows.

1. Start with the current "front" of the streamline set to the seed point.
2. Sample a local fibre orientation at the streamline front.
3. Move the front some small distance in the direction of the sampled direction.
4. Return to step 2, and repeat until a stopping criterion is met.
5. Return to step 1 and repeat once, travelling in the opposite direction away from the seed point.

Step 2 in the process described above involves some subtlety. Firstly, if a point estimate of the local fibre orientation is used – such as the principal eigenvector of a single diffusion tensor – then the sampling is deterministic. If MCMC is used to estimate a distribution over orientations, on the other hand, then the sampling may be repeated multiple times with different results in general. Thus, multiple streamlines may be generated from a single seed point. Local uncertainty will tend to accumulate in the streamlines' trajectories as one moves away from the seed point, and a set of streamlines generated in this way give an indication of the precision available for tracking pathways from the seed. Secondly, in models of diffusion allowing for multiple anisotropic components, multiple fibre directions may be present within each voxel, and one must therefore decide which component to sample from. The convention is usually to sample from all components, and then choose the sample which is most similar to the orientation of the previous step; but alternative strategies are possible, and may be more robust (e.g., Clayden and Clark 2010).

White matter pathways terminate in grey matter, which does not have the orientational coherence of white matter, and orientation uncertainty is therefore very high in these areas.

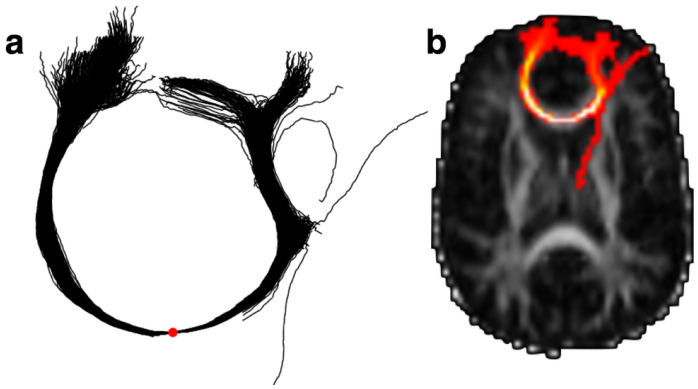

Figure 3: Example of fibre tracking in the brain. A set of 1000 sampled streamlines from a single seed point (the red dot) are shown individually (a), and summarised in a visitation map (b).

Stopping criteria usually ensure that the streamline does not leave the brain or turn back on itself, but entering grey matter may also be a reason to stop tracking.

Assuming BEDPOST has already been run on the session directory, and **FSL** is installed on the user's system, single seed tractography can be run as follows.

```
R> library("tractor.nt")
R> s <- newSessionFromDirectory(file.path(Sys.getenv("TRACTOR_HOME"),
+    "tests", "data", "session-12dir"))
R> l <- newStreamlineSetTractFromProbtrack(s, c(49, 58, 14), nSamples = 1000)
```

The second argument to `newStreamlineSetTractFromProbtrack` here provides the 3D coordinates of the seed point – using the R convention of indexing from one – and `nSamples` is the number of streamlines to generate. (If `nSamples` is set to 1, then only a single streamline is generated, and the result is analogous to "deterministic" tractography.) The actual tracking is performed by **FSL**'s "Probtrack" program. The resulting set of streamlines, which has class `StreamlineSetTract`, may be visualised using the `plot` method, viz. `plot(l)`. The result is shown in Figure 3a.

A common method of presentation for tractography results is as a visitation map or spatial histogram, an image with the same resolution as the original dMRI images wherein each voxel's value is the number of streamlines which pass through it. An example, corresponding to the tracking result obtained above, is shown in Figure 3b, as a maximum intensity projection overlaid on a slice of the FA image. This visualisation may be created using the following **TractoR** code.

```
R> f <- s$getImageByType("FA")
R> i <- newMriImageAsVisitationMap(l)
R> createSliceGraphic(f, z = 14)
R> createProjectionGraphic(i, axis = 3, colourScale = 2, add = TRUE)
```

The `axis` option in the last line controls the axis of the projection, while the `colourScale` option is used to select a heatmap, rather than grayscale, color scale for the overlay. Since

`add = TRUE` is given, the projection is overlaid on the FA slice graphic created by the previous command.

If only a visitation map is required, without the intermediate set of streamlines, then the same effect may be achieved using the command

```
R> r <- runProbtrackWithSession(s, c(49, 58, 14), mode = "simple",
+    requireImage = TRUE, nSamples = 1000)
```

The result of this command, `r`, will be an R list including an element, `r$image`, which contains the visitation map as an `MriImage` object.

5.2. Tracking between regions

An alternative to single seed tractography which is relevant to many applications is tracking between regions of the brain. Under this arrangement, a seed region is used to generate streamlines, and one or more "waypoint" regions are used to constrain their paths. In most implementations, streamlines which do not pass through all of the waypoint regions are simply discarded.

TractoR provides facilities for generating images representing simple cuboidal regions, and performing tracking using them as seed and target masks. For example,

```
R> sm <- newMriImageAsShapeOverlay(type = "block", baseImage = f,
+    centre = c(41, 70, 15), width = 3)
R> wm <- newMriImageAsShapeOverlay(type = "block", baseImage = f,
+    centre = c(55, 69, 16), width = 3)
R> r <- runProbtrackWithSession(s, mode = "seedmask", seedMask = sm,
+    waypointMasks = list(wm), requireImage = TRUE, nSamples = 50)
R> createSliceGraphic(f, z = 15)
R> createProjectionGraphic(r$image, axis = 3, colourScale = 2, add = TRUE)
R> createSliceGraphic(sm, z = 15, colourScale = "green", add = TRUE)
R> createSliceGraphic(wm, z = 15, colourScale = "blue", add = TRUE)
```

Figure 4: A visitation map showing the results of fibre tracking between regions. 50 streamlines have been generated from each voxel location within the seed region (green), and only those streamlines which passed through the waypoint region (blue) have been retained.

In this case, the `nSamples` option controls the number of streamlines generated *per seed point* within the mask. These commands create a seed mask and waypoint mask (each as a $3 \times 3 \times 3$ voxel block: the first two lines), run tractography using them (line 3), and visualise the results. The visualisation is built up with an FA slice as the base layer, then a projection of the tract, and finally the two regions of interest. It can be seen in Figure 4.

5.3. Neighbourhood tractography

"Neighbourhood tractography" is an alternative to waypoint methods, which uses a reference tract and machine learning methods to find consistent representations of a particular tract of interest in a group of dMRI data sets (Clayden *et al.* 2007, 2009b). The principle is to model the variation in shape of the tract across individuals, and then evaluate the plausibility of a given "candidate tract" as a match to the reference tract. The **tractor.nt** package provides functions and supporting data structures for performing neighbourhood tractography, and a standard set of reference tracts are provided with the main **TractoR** distribution (Muñoz Maniega *et al.* 2008). New reference tracts may also be created by the user.

Reference and candidate tracts are represented for these purposes as B-splines with a fixed distance between knot points. Where multiple streamlines are generated from a single seed point, the spline is fitted to the spatial median of the set. One knot is arranged to coincide with the seed point, and for the ith candidate tract in a data set there are then L_1^i knots to one side of the seed, and L_2^i to the other. Working away from the seed point, we denote the angle between the straight line connecting knot $u-1$ to knot u, and its equivalent in the reference tract, with ϕ_u^i. The index, u, is taken as being negative to one side of the seed, and positive to the other side.

A set of indicator variables, (z^i), describe which candidate tract is the best match to the reference tract within the data set, such that $z^i = 1$ if tract i is the best match and $z^i = 0$ otherwise. The special case $z^0 = 1$ is also allowed, to indicate no match. The likelihood of the model given the observed data then depends on the value of the relevant indicator variable. Our aim is to establish the posterior distribution $P(z^i \,|\, D)$, where D represents all data[7], for all candidate tracts, subject to the constraint that

$$\sum_{i=0}^{N} P(z^i = 1) = 1\,;$$

i.e., there is exactly one best match, or none.

Given the shape and length data for the best matching tract and an appropriate reference tract, the likelihood is given by

$$\begin{aligned} P(\mathbf{d}^i \,|\, \mathbf{A}, \mathbf{p}, z^i = 1) = {} & P(L_1^i \,|\, L_1^*, \mathbf{p}_1, z^i = 1)\, P(L_2^i \,|\, L_2^*, \mathbf{p}_2, z^i = 1) \\ & \times \prod_{u=1}^{\check{L}_1^i} P(\phi_{-u}^i \,|\, \alpha_u, z^i = 1) \prod_{u=1}^{\check{L}_2^i} P(\phi_u^i \,|\, \alpha_u, z^i = 1)\,, \end{aligned} \tag{4}$$

where $\mathbf{d}^i = (L_1^i, L_2^i, (\phi_u^i))$, $\mathbf{A} = (\alpha_u)$ and $\mathbf{p} = (\mathbf{p}_1, \mathbf{p}_2)$ – the latter two being parameters of the model. L_1^* and L_2^* are the lengths of the reference tract corresponding to L_1^i and L_2^i

[7]Note that the data here are not the raw images acquired from the scanner, but rather the variables describing tract shape which are derived from them.

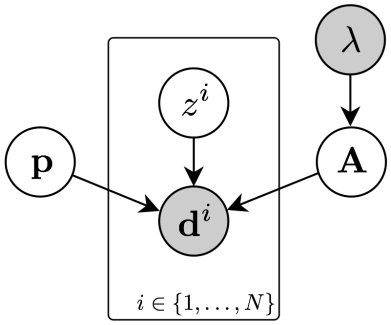

Figure 5: Graphical representation of the tract shape model for tracts matched to a reference tract, including priors on model parameters. The shaded nodes represent observed variables.

respectively; $\check{L}_1^i = \min\{L_1^i, L_1^*\}$, and equivalently for $\check{L}_2^i$. The corresponding forward model for tracts which do not match the reference (i.e., with $z^i = 0$) is uninformative, since the reference tract is not a good predictor of their topologies.

A graphical representation of this model is shown in Figure 5, and the distributional details are as follows.

$$\begin{aligned} L_1^i \,|\, L_1^* &\sim \text{Multinomial}(n_1, \mathbf{p}_1) \\ L_2^i \,|\, L_2^* &\sim \text{Multinomial}(n_2, \mathbf{p}_2) \\ \frac{\cos \phi_u^i + 1}{2} &\sim \text{Beta}(\alpha_u, 1) \end{aligned} \tag{5}$$

where $n_1 = |\mathbf{p}_1|$, and equivalently for n_2. Multinomial distributions are appropriate for the tract length variables because they reflect the number of knots either side of the seed point, which must be integral. The true length of the tract is therefore approximated by the sum of the fixed-length gaps between knots. We apply the prior

$$\alpha_u - 1 \sim \text{Exponential}(\lambda) \,, \tag{6}$$

which constrains each α_u to ensure that smaller deviations from the reference tract are always more likely (i.e., $\alpha_u \geq 1$), and also regularises their distributions to avoid model overfitting for small data sets.

The model may be fitted in a supervised fashion by choosing a set of training tracts representing good matches to the reference (Clayden *et al.* 2007), or taking an unsupervised approach using an expectation-maximisation (EM) algorithm (Clayden *et al.* 2009b). The unsupervised approach is generally preferable in applications, unless test data are very scarce, in which case it may be helpful to train from another population. The same framework may also be used to remove individual streamlines which are inconsistent with the trajectory of the reference tract (Clayden *et al.* 2009a).

A set of B-spline tract representations suitable for training or evaluating against a model may be created using the following commands.

```
R> library("tractor.nt")
R> s <- newSessionFromDirectory(file.path(Sys.getenv("TRACTOR_HOME"),
+    "tests", "data", "session-12dir"))
```

```
R> r <- getNTResource("reference", "pnt", list(tractName = "genu"))
R> n <- createNeighbourhoodInfo(centre = c(49, 58, 14), width = 3)
R> l <- calculateSplinesForNeighbourhood(s, n, r, nSamples = 100)
```

Here, we are using the standard reference for the "genu" tract as the reference[8], and creating B-spline candidate tracts for every point within a $3 \times 3 \times 3$ voxel neighbourhood centred at (49,58,14). Each B-spline is fitted to the median of 100 sampled streamlines. We must then calculate the components of $\mathbf{d}^i$ for each seed point, and this is achieved with

```
R> d <- createDataTableForSplines(l, r$getTract(), "knot", subjectId = 1,
+    neighbourhood = n)
```

The reference tract needs to be provided here to calculate the angles, (ϕ_u^i), between segments of the reference and candidate tracts.

We can now fit a model using these B-spline tracts as training data. (Of course, this is not wise in practice without manually removing aberrant tracts, and in any case tracts from several data sets should be used for a generalizable model, but this serves to illustrate the method.)

```
R> m <- newMatchingTractModelFromDataTable(d, r$getTract(), lambda = 1,
+    alphaOffset = 1, weights = NULL, asymmetric = TRUE)
R> print(m)

* * * * INFO: Asymmetric model  : TRUE
* * * * INFO: Alphas (left)     : 1.98, 1.96, 1.95, 1.97, 1.98
* * * * INFO: Alphas (right)    : 1.99, 1.95, 1.97, 1.96, 1.91
* * * * INFO: Ref tract lengths : 6 (left), 6 (right)
* * * * INFO: Length cutoff     : 13
* * * * INFO: Point type        : knot
```

The model object has class `MatchingTractModel`. The values of `lambda` (λ), `alphaOffset` and `weights` specified here will have a significant effect on the outcome – indeed, with only one subject in the data set, the prior dominates in this example, limiting the values of α_u severely. The larger the value of `lambda`, the greater the extent to which the prior will favour small values of α_u, since the exponential prior has mean $1/\lambda$. A value of `alphaOffset=1` corresponds to the prior in Equation 6. If the specified weights are `NULL`, then each tract is given the same weight. In the EM context, this corresponds to an assumption that each tract is *a priori* equiprobable. The `asymmetric` parameter determines whether a constraint that $\alpha_u = \alpha_{-u}$ should be applied (the symmetric case) or not (the asymmetric case). It is generally wise to set this to `TRUE` unless very few data are available.

The trained model generated above may be used directly to calculate the posterior matching probabilities of a new set of candidate tracts. However, in applications it is often helpful to work in an unsupervised fashion, learning the model and calculating posteriors in one go, since this removes the need for separate training data. This can be achieved as follows.

[8]The `getNTResource()` function is used to obtain various standard resources provided with **TractoR** for performing neighbourhood tractography. In this case, we are requesting a reference tract (`"reference"`), associated with tract name `"genu"`, appropriate for probabilistic neighbourhood tractography (`"pnt"`).

```
R> m <- runMatchingEMForDataTable(d, r$getTract(), lambda = 1,
+    alphaOffset = 1, asymmetricModel = TRUE)
R> print(m$mm)

* * * * INFO: Asymmetric model  : TRUE
* * * * INFO: Alphas (left)     : 1.98, 1.96, 1.94, 1.97, 1.98
* * * * INFO: Alphas (right)    : 1.98, 1.95, 1.97, 1.96, 1.92
* * * * INFO: Ref tract lengths : 6 (left), 6 (right)
* * * * INFO: Length cutoff     : 13
* * * * INFO: Point type        : knot

R> which.max(m$tp[[1]])

[1] 14
```

The list `m$tp` gives the tract matching posteriors for each subject in the data set. Here, under the final model, the 14th (of 27) candidate tracts has the highest posterior. The mean FA within this tract may finally be calculated as follows.

```
R> t <- runProbtrackWithSession(s, n$vectors[ , 14], mode = "simple",
+    requireImage = TRUE, nSamples = 1000)
R> calculateMetricForResult(t, type = "fa", mode = "binary",
+    threshold = 0.01)

[1] 0.4503506
```

This represents the mean FA within the region visited by at least 1% (`threshold=0.01`) of the streamlines initiated from the final seed point. Mean FA values from a full data set may then be analysed using standard R hypothesis testing functions such as `t.test()` or `cor.test()`.

It should be noted that there is considerable scope in the code provided with the **tractor.nt** package for adjusting the exact processes applied, or for using similar models for other purposes, but we have focussed on a standard usage here for brevity.

6. The TractoR shell interface

In addition to the three main R packages whose major functionality has been laid out above, **TractoR** provides a shell interface and set of related R "experiment scripts" for performing various common tasks quickly and directly. In addition, it allows those without experience of working with R to use some of **TractoR**'s core functionality immediately. An additional R package, **tractor.utils**, exists mainly to support this interface. It should be noted that the interface uses a Unix shell script and is therefore not directly usable in Windows.

The interface works through a single shell script program called `tractor`, which may be executed by typing just its name if the `bin` subdirectory of the **TractoR** distribution is on the user's `PATH`. Specific tasks are chosen by giving the name of a particular experiment script: for example, the `preproc` script may be used for preprocessing dMRI data, the `track` script can be used for performing tractography, and so on. All available experiment scripts may be listed by using the `list` script, viz.

```
sh> tractor list
Starting TractoR environment...
Experiment scripts found in /usr/local/tractor/share/experiments:
 [1] age             binarise        camino2fsl      caminofiles
 [5] chfiletype      contact         dicomread       dicomsort
 [9] dicomtags       dirviz          extract         fsl2camino
[13] gmap            gmean           hnt-eval        hnt-interpret
[17] hnt-ref         hnt-viz         identify        imageinfo
[21] list            mean            mkbvecs         mkroi
[25] mtrack          peek            platform        plotcorrections
[29] pnt-collate     pnt-data-sge    pnt-data        pnt-em
[33] pnt-eval        pnt-interpret   pnt-prune       pnt-ref
[37] pnt-train       pnt-viz         preproc         proj
[41] rtrack          slice           status          track

Experiment completed with 0 warning(s) and 0 error(s)
```

In addition, a description and list of supported arguments and options for a particular script may be obtained by using the `-o` flag to the `tractor` program.

```
sh> tractor -o mean
OPTIONS for script /usr/local/tractor/share/experiments/mean.R (* required)
   Metric: NULL [weight,AVF,FA,MD,Lax,Lrad]
   AveragingMode: binary [weighted]
   WeightThreshold: 0.01
   ThresholdRelativeTo: nothing [maximum,minimum]
ARGUMENTS: image file, [session directory]

Calculate the mean or weighted mean value of a metric within the nonzero
region of a brain volume (usually tractography output). The metric can be
FA, MD, Lax, Lrad or AVF, and the specified image can be used as a binary
mask (the default) or as a set of weights (with AveragingMode:weighted).
In the latter case any weight threshold given is ignored.
```

The `tractor` program has a full Unix `man` page, and that may be consulted for further details on how to use and configure this interface.

7. Conclusion

In this paper we have described **TractoR**, a set of R packages along with a separate shell interface and scripting platform for processing magnetic resonance images in general, and dMRI data in particular. The package provides facilities for general-purpose reading, writing and manipulation of 2D, 3D and 4D images, along with signal modelling, tractography and tract shape modelling functions specific to dMRI. It is intended that this platform will continue to broaden in the future, making further use of the comprehensive statistical capabilities of the R language and package ecosystem. Reducing **TractoR**'s reliance on **FSL** for key functionality

such as tractography and registration is also planned for the future, for example by making use of the recently released **RNiftyReg** image registration package (Clayden 2011).

References

Basser PJ, Mattiello J, Le Bihan D (1994). "Estimation of the Effective Self-Diffusion Tensor from the NMR Spin Echo." *Journal of Magnetic Resonance, Series B*, **103**(3), 247–254.

Behrens TEJ, Johansen-Berg H, Jbabdi S, Rushworth MFS, Woolrich MW (2007). "Probabilistic Diffusion Tractography with Multiple Fibre Orientations: What Can We Gain?" *NeuroImage*, **34**(1), 144–155.

Clayden JD (2011). ***RNiftyReg**: Medical Image Registration Using the **NiftyReg** Library*. R package version 0.3.1, URL http://CRAN.R-project.org/package=RNiftyReg.

Clayden JD, Clark CA (2010). "On the Importance of Appropriate Fibre Population Selection in Diffusion Tractography." In *Proceedings of the ISMRM-ESMRMB Joint Annual Meeting*, p. 1679. International Society for Magnetic Resonance in Medicine.

Clayden JD, King MD, Clark CA (2009a). "Shape Modelling for Tract Selection." In GZ Yang, D Hawkes, D Rueckert, A Noble, C Taylor (eds.), *Medical Image Computing and Computer-Assisted Intervention*, volume 5762 of *Lecture Notes in Computer Science*, pp. 150–157. Springer-Verlag.

Clayden JD, Storkey AJ, Bastin ME (2007). "A Probabilistic Model-Based Approach to Consistent White Matter Tract Segmentation." *IEEE Transactions on Medical Imaging*, **26**(11), 1555–1561.

Clayden JD, Storkey AJ, Muñoz Maniega S, Bastin ME (2009b). "Reproducibility of Tract Segmentation Between Sessions Using an Unsupervised Modelling-Based Approach." *NeuroImage*, **45**(2), 377–385.

Evans A, Kamber M, Collins DL, MacDonald D (1994). "An MRI-Based Probabilistic Atlas of Neuroanatomy." In S Shorvon, D Fish, F Andermann, GM Bydder, H Stefan (eds.), *Magnetic Resonance Scanning and Epilepsy*, volume 264 of *NATO ASI Series A, Life Sciences*, pp. 263–274. Plenum Press.

Friston KJ, Ashburner J, Kiebel SJ, Nichols TE, Penny WD (eds.) (2007). *Statistical Parametric Mapping: The Analysis of Functional Brain Images*. Academic Press, New York.

Jones DK (2008). "Studying Connections in the Living Human Brain with Diffusion MRI." *Cortex*, **44**(8), 936–952.

Le Bihan D (2003). "Looking into the Functional Architecture of the Brain with Diffusion MRI." *Nature Reviews Neuroscience*, **4**(6), 469–480.

MacKay DJC (1995). "Probable Networks and Plausible Predictions – A Review of Practical Bayesian Methods for Supervised Neural Networks." *Network: Computation in Neural Systems*, **6**(3), 469–505.

Muñoz Maniega S, Bastin ME, McIntosh AM, Lawrie SM, Clayden JD (2008). "Atlas-Based Reference Tracts Improve Automatic White Matter Segmentation with Neighbourhood Tractography." In *Proceedings of the ISMRM 16th Scientific Meeting & Exhibition*, p. 3318. International Society for Magnetic Resonance in Medicine.

Polzehl J, Tabelow K (2011). "Beyond the Gaussian Model in Diffusion-Weighted Imaging: The Package **dti**." *Journal of Statistical Software*, **44**(12), 1–25. URL http://www.jstatsoft.org/v44/i12/.

R Development Core Team (2011). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria. ISBN 3-900051-07-0, URL http://www.R-project.org/.

Salvador R, Peña A, Menon DK, Carpenter TA, Pickard JD, Bullmore ET (2005). "Formal Characterization and Extension of the Linearized Diffusion Tensor Model." *Human Brain Mapping*, **24**(2), 144–155.

Smith SM, Jenkinson M, Woolrich MW, Beckmann CF, Behrens TEJ, Johansen-Berg H, Bannister PR, De Luca M, Drobnjak I, Flitney DE, Niazy RK, Saunders J, Vickers J, Zhang Y, De Stefano N, Brady JM, Matthews PM (2004). "Advances in Functional and Structural MR Image Analysis and Implementation as **FSL**." *NeuroImage*, **23**(Supplement 1), S208–S219.

Affiliation:

Jonathan D. Clayden
Imaging & Biophysics Unit
UCL Institute of Child Health
30 Guilford Street
London WC1N 1EH, United Kingdom
E-mail: j.clayden@ucl.ac.uk
URL: http://www.homepages.ucl.ac.uk/~sejjjd2/

Journal of Statistical Software http://www.jstatsoft.org/
published by the American Statistical Association http://www.amstat.org/
Volume 44, Issue 8 *Submitted:* 2010-10-28
October 2011 *Accepted:* 2011-06-02