Orientlib: An R Package for Orientation Data

D.J. Murdoch

December 12, 2003

Abstract

Orientlib is an R package to facilitate working with orientation data, i.e. elements of $SO(3)$. It provides automatic translation between different representations of orientations, including rotation matrices, quaternions, Euler angles and skew-symmetric matrices; it also has functions for fitting regression models and displaying orientations. This paper reviews simple properties of orientation data and describes Orientlib.

Keywords: SO(3), orientation data, Euler angles, quaternions, rotation matrices

1 Introduction

This paper describes Orientlib, an R package that provides routines for working with data describing the orientation of solid objects in 3-space. R (Ihaka and Gentleman, 1996) is an open source implementation of the S language (Chambers and Hastie, 1992; Chambers, 1998). The aim of Orientlib is to facilitate research into statistical methods using orientation data, by providing general support routines, including conversions between commonly used representations, simple transformations and statistical summaries, and one form of modelling function.

Orientlib was motivated by work done several years ago in calibrating the Polhemus FasTrak system used by the Ergonomics Research Group (ERG) at Queen's University. The FasTrak system is composed of a radio frequency transmitter that broadcasts a polarized signal, and a number of detectors that measure received signal strength in three orthogonal planes. A central unit collects all of the

information and outputs an estimate of the spatial position and orientation of each of the detectors.

The ERG was using the FasTrak on site at an industrial plant, and found that structural steel in the building distorted the measurements unacceptably. We were involved in developing calibration procedures for it; these have been described elsewhere (Day, Murdoch and Dumas, 2000). Other recent work in orientation data includes that of Rancourt *et al.* (2000) on modelling and Shin *et al.* (2001; 2003) on design of experiments. ORIENTLIB is designed to provide the support necessary to test and implement such work.

2 Orientation data

Orientation data describes the spatial orientation of a rigid object. An orientation may be represented as the transformation necessary to rotate standard vectors to the desired orientation, or as the inverse transformation, which rotates the desired orientation into a standard orientation. Orientations form the non-commutative compact group $SO(3)$.

There are a number of different ways in which elements of $SO(3)$ may be represented.

2.1 3×3 matrices

Rotations are linear transformations that preserve length and handedness; they may be represented by 3×3 orthogonal matrices (i.e. matrices A satisfying $AA^t = A^tA = I$), with determinant 1. Each such matrix uniquely identifies an orientation, but uses 9 numerical components to do it, with 6 nonlinear constraints. Though $SO(3)$ is a 3 dimensional manifold, there is no subset of 3 of the 9 matrix entries that uniquely identifies the orientation.

2.2 Quaternions

Any SO(3) matrix may be represented as a rotation through angle θ about a single axis parallel to (x, y, z). A four component unit vector, whose first three components are parallel to (x, y, z) and whose fourth component is $\cos(\theta/2)$, corresponds to Hamilton's definition of quaternions: the cosine forms the real part and the components of the

vector parallel to (x, y, z) form the three imaginary parts. Quaternion multiplication corresponds to composition of rotations. Each quaternion q and its negative $-q$ represent the same orientation; other than this, the representation is unique.

2.3 Euler angles

Any orientation may be written as the composition of 3 rotations about standard axes. There are in common use a large number of conventions specifying the order of rotations. Two are the "x-convention" in which rotations are specified as counter-clockwise rotations about the Z axis, then about the X axis, then Z axis again, and the pitch-roll-yaw convention, in which the rotations are about the Z, then Y, then X axes. All Euler schemes give a number of different representations for each orientation: multiples of 2π may be added to any rotation angle, and certain combinations of angles lead to other non-uniqueness.

2.4 Skew-symmetric matrices

A skew symmetric matrix is a matrix S satisfying $S + S^t = 0$. It is easy to see that a 3×3 skew symmetric matrix can be written as

$$S = \begin{pmatrix} 0 & c & -b \\ -c & 0 & a \\ b & -a & 0 \end{pmatrix}$$

and it is not hard to verify that $\mathrm{Exp}(S) = \sum_{i=0}^{\infty} S^i/i!$ is a rotation matrix with rotation axis (a, b, c). Less obvious is that the rotation angle is the Euclidean norm $\sqrt{a^2 + b^2 + c^2}$; thus each orientation matrix has multiple representations as "component vectors" (a, b, c) differing in length by multiples of 2π.

2.5 Properties of Orientations

Given any pair of orientations U and V, there is a unique orientation $A_1 = VU^t$ such that $V = A_1U$, where the product indicates composition of rotations. When the orientations are represented as 3×3 matrices, the product is the usual matrix product; using quaternions, it is the usual quaternion product. In the other representations composition of rotations does not have a simple form.

Because $SO(3)$ is non-commutative, the matrix $A_2 = U^t V$ solving $V = UA_2$ is generally different from A_1 as defined above. If U and V share an axis of rotation, or if both are rotations of π radians (i.e. are symmetric) with orthogonal axes of rotation, then $A_1 = A_2$.

The rotation angle of A_1 will always match that of A_2; this gives a natural distance measure between U and V. It turns out that this is a monotone transformation of coordinatewise Euclidean distance. Stephens (1979) described an algorithm for finding the nearest $SO(3)$ matrix to a given matrix.

Prentice (1989) defined a model for matched pairs (U_i, V_i) of orientations, in which $V_i = A_1^t U_i A_2 E_i$, where U_i is treated as fixed, and E_i is a random orientation having a distribution depending only on the magnitude of the rotation angle, but not on the direction of the rotation axis. Shin (1999) derived conditions on the choice of the U_i orientations under which both A_1 and A_2 were identifiable.

A simple generalization of Prentice's model to incorporate covariates is to allow A_1 and A_2 to depend on the covariates. One way to do this is to write each as the nearest $SO(3)$ matrix to a matrix whose entries are modelled as linear combinations of covariates, i.e.

$$A_k(x_1, x_2, \ldots) = \text{argmin}_{A \in SO(3)} ||A - \sum_j x_j B_{kj}||$$

where B_{kj}, $k = 1, 2$, $j = 1, \ldots, p_k$ are arbitrary 3×3 matrices regarded as regression coefficients in the model, and the x_j values are scalar covariates.

If for either $k = 1$ or $k = 2$ the B_{kj} matrices are all known, then the rest may be estimated by least squares. For example, we may know the coefficients for $k = 1$, so that A_1 is known. A common example would be to know that A_1 is the identity matrix. Then the B_{2j} coefficients involved in calculating A_2 are estimated by coordinatewise least squares fitting of the linear model to coordinates of the matrices $U_i^t A_1 V_i$. Similarly, if A_2 is known, the B_{1j} coefficients are obtained by least squares on the entries of $U_i A_2 V_i^t$.

In the case where both sets of coefficients are unknown, nonlinear least squares is necessary. By analogy with Prentice (1989), one approach is to alternate between treating A_2 as fixed and using linear least squares to estimate B_{1j}, and treating A_1 as fixed and estimating B_{2j}. It is known (Shin, 1999) that Prentice's algorithm sometimes fails to find the global minimum, so care must be taken to try multiple starting values when good initial estimates are not known.

The ORIENTLIB package includes an example fitting this model to artificial data; see `?orientlm`. Different regression models using other representations of orientations are possible, but the non-uniqueness of those representations complicates fitting.

2.6 Graphical Displays of Orientations

Displaying orientations is challenging. One approach is to display various one-dimensional statistics in scatterplots; for example, the rotation angle or components of the rotation axis. These plots are straightforward to produce, but fail to give the whole picture about the orientation.

Murdoch (1996) introduced the use of stylized sailboats to display orientations. These are drawn with polygons, and have enough asymmetry to allow visual identification of the amount and direction of rotation (Figures 1 and 2). Sailboats can be rendered using the `djmrgl` package (Murdoch, 2001), available from `http://www.stats.uwo.ca/faculty/murdoch/software`, the `rgl` package (Adler and Nenadic, 2003), available from `http://wsopuppenkiste.wiso.uni-goettingen.de/~dadler/rgl/` or wireframe versions can be rendered using `scatterplot3d` (Ligges and Mächler, 2003). Other 3D rendering packages could also be used.

3 Design of the package

R currently has two systems for object-oriented programming: one based on version 3 of the S language (Chambers and Hastie, 1992) and another based on version 4 (Chambers, 1998), which we call the "Sv4 classes". The former uses conventions in function names to attach them to objects of different classes, whereas the latter uses a registration system to connect methods to classes.

The ORIENTLIB package is designed around an Sv4 class called `orientation`. This is an abstract class: users cannot create instances of `orientation`. Instead, they create instances of concrete descendant classes `rotmatrix` (3×3 rotation matrices), `rotvector` (the same matrices written out as 9 component vectors), `eulerzxz` or `eulerzyx` (Euler angles), `quaternion` (quaternions as 4 component vectors), `skewmatrix` or `skewvector` (skew-symmetric matrices stored whole or as their component vectors respectively). The `orientation` class is

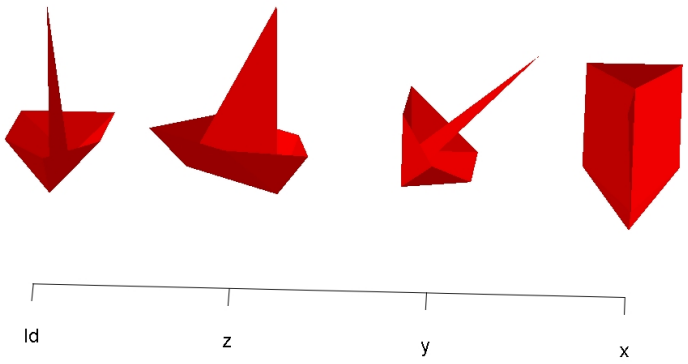

Figure 1: Four orientations displayed using sailboats and rendered using the `djmrgl` package. From left to write, they are the identity orientation, then rotations of $\pi/4$ about the z, y and x axes.

declared to be a descendant of the `vector` class, so methods are defined in each descendant type for finding the length of an `orientation`, setting and extracting sub-vectors (e.g. `x[1:10] <- x[11:20]`), and extracting and setting elements (e.g. `x[[3]] <- c(1,2,3)`).

Conversion methods are also defined to allow conversion between all of the orientations. These may be called with the same names as the constructor functions, so that the following code creates an orientation using the `eulerzyx` class, then coerces it to a `rotmatrix`, and back to an `eulerzyx`:

```
> x <- eulerzyx(1,2,3)
> y <- rotmatrix(x)
> z <- eulerzyx(y)
```

Most other functions are designed to work on the `orientation` class. They coerce their arguments to whichever storage form is most convenient. For example, the `mean` method is defined as

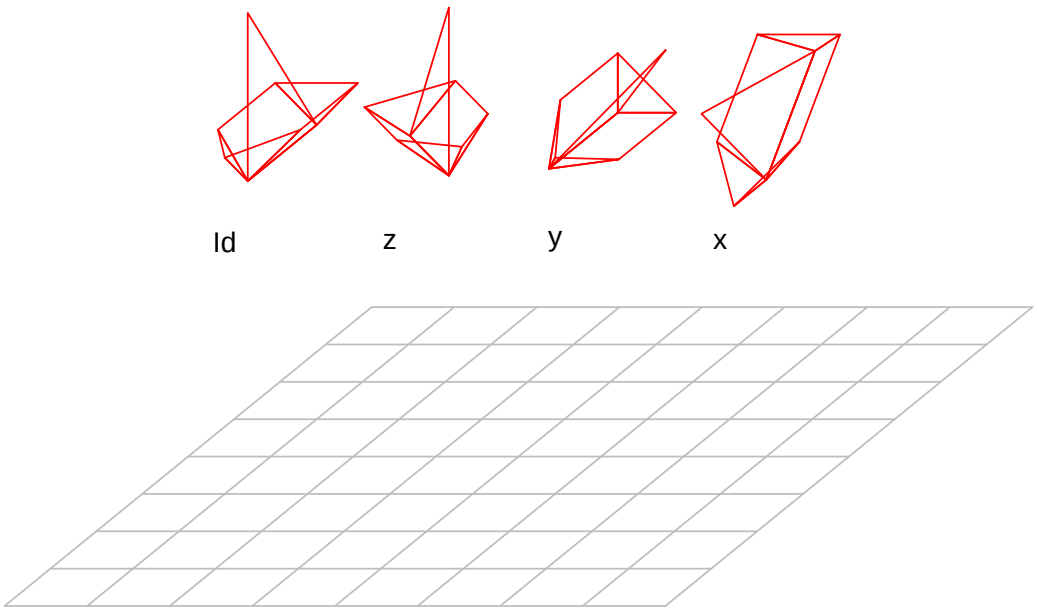

Figure 2: The same orientations as in Figure 1 rendered using the `scatterplot3d` package.

```
setMethod('mean', 'orientation',
    function(x) {
        x <- rotmatrix(x)@x
        nearest.SO3(apply(x,c(1,2),mean))
    }
)
```

The first line of the method definition converts the orientation to `rotmatrix` form and extracts the array of 3×3 matrices; the second line takes the usual mean of those matrices, and then finds the nearest $SO(3)$ matrix to the result. Implicit in this style of computation is the assumption that the user usually doesn't care which method of storage is being used. If they do, they should coerce the result to the desired storage class when necessary.

The package provides a single regression function, `orientlm`. Because the model discussed in Section 2.5 allows covariates to appear in either A_1 or A_2, and the standard R modelling functions don't support Sv4 vectors in formulas, this function needs an unusual design. We give it four main arguments, as well as several optional ones. The

main arguments are the response `orientation` V_i, a formula for the model for A_1, the "true" orientation U_i, and a formula for the model for A_2. If any of the last three arguments above are missing, they are treated as identity orientations. With this design, the standard R functions `model.frame` and `model.matrix` may be used to construct the design matrices for the linear regressions.

4 List of classes, methods and functions

In this section we present an overview of the classes, methods and functions in the ORIENTLIB package. For full documentation on each, see the help files in the package itself.

4.1 Class `orientation`

This is the abstract base class, descending from `vector`. Most methods are designed to work on `orientation` objects.

4.2 Descendant classes

Each of the descendant classes has one slot named `x` which holds the data.

The `rotmatrix` class uses a $3 \times 3 \times n$ array to hold 3×3 matrix representations. The `skewmatrix` class stores skew-symmetric matrices in the same shape of array.

The other descendant classes all use $n \times m$ matrices to hold their data, where m is 3, 4 or 9. The `rotvector` class holds vectorized 3×3 matrices, stored in column-major order. The `eulerzyx` class holds 3 Euler angles using the roll-pitch-yaw scheme. The `eulerzxz` class holds 3 Euler angles using the x-convention. The `quaternion` class holds the 4 components of the quaternion representation. The `skewvector` class holds the 3 element component vector of the skew-symmetric matrix representation.

4.3 Methods

The following methods are defined for the `orientation` class:

`coerce`: Methods are defined to coerce `orientation` objects to any concrete descendant class.

`%*%`: Matrix multiplication acts on `orientation` objects component by component. The product of two orientations is the rotation formed by applying the right term followed by the left term.

`^`: An orientation is raised to a power by multiplying its rotation angle by that power.

`t`: The transpose of an orientation is its inverse.

`mean`: The mean of an `orientation` object is the nearest SO(3) matrix to the element-by-element mean of its 3×3 matrix representation.

`weighted.mean`: The weighted mean of an `orientation` object is defined analogously to the mean.

The following methods are defined separately for each descendant class:

`coerce`: Coercion methods are defined to coerce between all pairs of concrete classes of orientations.

`[` and `[<-`: Extract or assign to a subvector of an `orientation`.

`[[` and `[[<-`: Extract or assign to an entry of an `orientation`.

`length`: Find the length of an `orientation`.

4.4 Functions

The following functions are defined. Some of them are defined as methods to take advantage of the method dispatch mechanism in R, but it is simpler to describe them as functions.

`rotmatrix`, `rotvector`, `eulerzyx`, `eulerzxz`, `quaternion`, `skewmatrix`, `skewvector`: These functions create objects of the corresponding class, or coerce one class to another.

`rotation.angle`: The rotation angle of an orientation is the angle (in radians) by which it is rotated from the standard (identity) orientation.

`rotation.distance`: The rotation distance between two orientations is the angle by which one may be rotated to produce the other.

`nearest.SO3`: The nearest $SO(3)$ matrix to a given 3×3 matrix is the one which is nearest in componentwise Euclidean distance. This function works on a $3 \times 3 \times n$ array of matrices using Stephens' (1979) algorithm, producing an `orientation` object.

`nearest.orthog`: The nearest orthogonal matrix to a given 3×3 matrix is the one which is nearest in componentwise Euclidean distance. This function produces an array of the same shape as the input.

`boat3d`: Draw stylized sailboats.

`orientlm`: Fit the generalized Prentice linear model.

References

Adler, D. and Nenadic, O. (2003). A framework for an R to OpenGL interface for interactive 3D graphics. Submitted.

Chambers, J. M. (1998). *Programming with Data: A Guide to the S Language*. Springer-Verlag.

Chambers, J. M. and Hastie, T. J. (1992). *Statistical Models in S*. Wadsworth and Brooks/Cole.

Day, J. S., Murdoch, D. J., and Dumas, G. S. (2000). Calibration of position and angular data from a magnetic tracking device. *Journal of Biomechanics*, 33:1039–1045.

Ihaka, R. and Gentleman, R. (1996). R: A language for data analysis and graphics. *Journal of Computational and Graphical Statistics*, 5(3):299–314.

Ligges, U. and Mächler, M. (2003). Scatterplot3d–an R package for visualizing multivariate data. *Journal of Statistical Software*, 8(11):1–20.

Murdoch, D. J. (1996). *User's Manual for CALIBRATE: A Program to Calibrate FasTrak Data*. Unpublished manual for computer program.

Murdoch, D. J. (2001). RGL: An R interface to OpenGL. In Hornik, K. and Leisch, F., editors, *Proceedings of the 2nd International Workshop on Distributed Statistical Computing, Vienna, March 2001*. ISSN 1609-395X, http://www.ci.tuwien.ac.at/Conferences/DSC-2001/Proceedings.

Prentice, M. J. (1989). Spherical regression on matched pairs of orientation statistics. *Journal of the Royal Statistical Society, Series B*, 51:241–248.

Rancourt, D., Rivest, L. P., and Asselin, J. (2000). Using orientation statistics to investigate variations in human kinematics. *Applied Statistics*, 15:39–50.

Shin, H. H. (1999). *Experimental Designs for Orientation Models*. PhD thesis, Queen's University.

Shin, H. H., Takahara, G. K., and Murdoch, D. J. (2001). Uniqueness, consistency and optimality in spherical regression experiments. *Statistics and Probability Letters*, 54:61–65.

Shin, H. H., Takahara, G. K., and Murdoch, D. J. (2003). Minimal optimal designs for orientation experiments. In preparation.

Stephens, M. (1979). Vector correlation. *Biometrika*, 66:41–48.