rotations: An R Package for $SO(3)$ Data

by Bryan Stanfill, Heike Hofmann, Ulrike Genschel

Abstract In this article we introduce the **rotations** package which provides users with the ability to simulate, analyze and visualize three-dimensional rotation data. More specifically it includes four commonly used distributions from which to simulate data, five estimators of the central orientation, seven confidence region estimation procedures and a novel approach to visualizing these data. All of the above features are available to two different parameterizations of rotations: three-by-three matrices and quaternions. In addition two datasets are included that illustrate the use of rotation data in practice.

Introduction

Data in the form of three-dimensional rotations find application in many scientific areas, such as biomedical engineering, computer vision, and geological and materials sciences where such data represent the positions of objects within some three-dimensional reference frame. For example, Humbert et al. (1996); Bingham et al. (2009a); Bachmann et al. (2010) apply rotation data to study the orientation of cubic crystals on the surfaces of metal. Rancourt et al. (2000) use rotations to represent variations in human movement while performing a task.

A common goal shared in the analysis of this kind of data across all these fields is to estimate the main or central orientation for a sample of rotations. More formally, let $SO(3)$ denote the rotation group, which consists of all real-valued 3×3 matrices R with determinant equal to +1. Then observations $\boldsymbol{R}_1, \ldots, \boldsymbol{R}_n \in SO(3)$ can be conceptualized as a random sample from a *location model*

$$\mathbf{R}_i = \boldsymbol{S}\boldsymbol{E}_i, \quad i = 1, \ldots, n, \tag{1}$$

where $\boldsymbol{S} \in SO(3)$ is the *fixed* parameter of interest indicating the central orientation, and $\boldsymbol{E}_1, \ldots, \boldsymbol{E}_n \in SO(3)$ denote i.i.d. *random* rotations which symmetrically perturb $\boldsymbol{S}$. Model (1) is a rotation-matrix analog of a location model for scalar data $Y_i = \mu + e_i$, where $\mu \in \mathbb{R}$ denotes a mean and $e_i \in \mathbb{R}$ denotes an additive error symmetrically distributed around zero.

In the context of a location model in $SO(3)$, the assumption of symmetric perturbation $\boldsymbol{E}_i$ implies that the observations $\boldsymbol{R}_i$ have no preferred direction relative to $\boldsymbol{S}$. Further, the symmetry assumption implies that $E(\boldsymbol{R}_1) = c\boldsymbol{S}$ for some $c \in \mathbb{R}^+$. Also note that under the symmetry assumption, (1) could be equivalently specified as $\boldsymbol{R}_i = \boldsymbol{E}_i\boldsymbol{S}$, though the specification in (1) is the most common form in the literature, see Bingham et al. (2009a) for details.

While there is a multitude of packages and functions available in R to estimate the mean in a location model; the toolbox for rotational data is limited. The **orientlib** (Murdoch, 2003) package includes the definition of an orientation class along with a few methods to summarize and visualize rotation data. A strength of the **orientlib** package is its thorough exploration of rotation representations, but the estimation and visualization techniques are lacking and no methods for inference are available. The **onion** package includes functions for rotation algebra but only the quaternion form is available and data analysis is not possible (Hankin, 2011). The **uarsbayes** (Qiu, 2013) package includes functions for data generation and Bayes inference but this package is currently not publicly available. Packages for circular and spherical data, e.g. **circular** (Agostinelli and Lund, 2011) and **SpherWave** (Oh and Kim, 2013), can possibly be used but their extension to rotation data is not trivial or straightforward.

The **rotations** package fills this void by providing users with the tools necessary to simulate rotations from (1) allowing for several distributions for the perturbation matrices $\boldsymbol{E}_i$. Estimation and inference for $\boldsymbol{S}$ in (1) from both frequentist and Bayesian perspectives are available along with novel visualization techniques. The remainder of this manuscript introduces rotation data more fully and discusses the ways they are handled by Version 1.0 of the **rotations** package.

Rotation parameterizations

Several parameterizations of rotations exist. We consider two of the most commonly used: orthogonal 3×3 matrices with determinant one and four-dimensional unit vectors called *quaternions*. The **rotations** package allows for both parameterizations as input as well as transforming one into the other. We will briefly discuss each:

Matrix form

Rotations in three-dimensions can be represented by 3×3 orthogonal matrices with determinant one. Matrices with these characteristics form a group called the *special orthogonal group*, or *rotation group*, denoted $SO(3)$. Every element in $SO(3)$ is associated with a skew-symmetric matrix $\Phi(\boldsymbol{W})$ where

$$\boldsymbol{\Phi}(\boldsymbol{W}) = \begin{pmatrix} 0 & -w_3 & w_2 \\ w_3 & 0 & -w_1 \\ -w_2 & w_1 & 0 \end{pmatrix}$$

and $\boldsymbol{W} \in \mathbb{R}^3$. Applying the exponential operator to the matrix $\Phi(\boldsymbol{W})$ results in the rotation $\boldsymbol{R}$

$$\boldsymbol{R} = \exp[\boldsymbol{\Phi}(\boldsymbol{W})] = \sum_{k=0}^{\infty} \frac{[\boldsymbol{\Phi}(\boldsymbol{W})]^k}{k!}. \tag{2}$$

Since $\boldsymbol{\Phi}(\boldsymbol{W})$ is skew-symmetric, it can be show that (2) reduces to

$$\boldsymbol{R} = \cos(r)\boldsymbol{I}_{3\times 3} + \sin(r)\boldsymbol{\Phi}(\boldsymbol{U}) + [1 - \cos(r)]\boldsymbol{U}\boldsymbol{U}^\top \tag{3}$$

where $r = \|\boldsymbol{W}\|$, $\boldsymbol{U} = \boldsymbol{W}/\|\boldsymbol{W}\|$. In the material sciences literature the values r and $\boldsymbol{U} \in \mathbb{R}^3$ are termed the *misorientation angle* and *misorientation axis*, respectively.

Given a rotation matrix $\boldsymbol{R}$ one can find the associated skew-symmetric matrix $\boldsymbol{\Phi}(\boldsymbol{W})$ by applying the logarithm operator defined by

$$\text{Log}(\boldsymbol{R}) = \begin{cases} \boldsymbol{0} & \text{if } \theta = 0 \\ \frac{r}{2\sin r}(\boldsymbol{R} - \boldsymbol{R}^\top) & \text{otherwise} \end{cases} \tag{4}$$

where $r \in [-\pi, \pi)$ satisfies $\text{tr}(\boldsymbol{R}) = 1 + 2\cos r$. For more on the correspondence between $SO(3)$ and skew-symmetric matrices see Stanfill et al. (2013).

The **rotations** package defines the `S3` class `"SO3"`, which internally stores a sample of n rotations as a $n \times 9$ matrix. If $n = 1$ then an object of class `"SO3"` is printed as a 3×3 matrix but for $n > 1$ the $n \times 9$ matrix is printed. Objects can be coerced into, or tested for the class `"SO3"` with the `as.SO3` and `is.SO3` functions, respectively. Any object passed to `is.SO3` is tested for three characteristics: dimensionality, i.e. is $n \times 9$ for some integer n, orthogonality and determinant one.

The `as.SO3` function coerces the input into the class `"SO3"`. Given an angle r and axis $\boldsymbol{U}$, `as.SO3` will form a matrix according to (3); given a three-dimensional vector $\boldsymbol{W}$ the length of that vector will be taken to be the angle of rotation $r = \|\boldsymbol{W}\|$ and the axis is taken to be the unit-vector in the direction of $\boldsymbol{W}$, $\boldsymbol{U} = \boldsymbol{W}/\|\boldsymbol{W}\|$. Alternatively, one can also supply a rotation $\boldsymbol{Q}$ in the quaternion representation then the `as.SO3` function returns the matrix equivalent of $\boldsymbol{Q}$. For all input types the function `as.SO3` returns an $n \times 9$ matrix of class `"SO3"` where each row corresponds to a rotation matrix. Below we illustrate the use of the `as.SO3` function by constructing first the 3×3 matrix associated with a $90°$ rotation about the y-axis, i.e. $r = \pi/2$ and $\boldsymbol{U} = (0, 1, 0)$, rounded to three digits.

```
r <- pi/2
U <- c(0, 1, 0)
W <- U * r
R <- as.SO3(W)
R

##            [,1] [,2]     [,3]
## [1,]  6.12e-17    0 1.00e+00
## [2,]  0.00e+00    1 0.00e+00
## [3,] -1.00e+00    0 6.12e-17

identical(R, as.SO3(U, r))

## [1] TRUE
```

Given a rotation matrix $\boldsymbol{R}$, the functions `mis.angle` and `mis.axis` will determine the misorientation angle and axis of an object with class `"SO3"` as illustrated in the next example.

```
mis.angle(R) * 2/pi

## [1] 1
```

```
mis.axis(R)

##      [,1] [,2] [,3]
## [1,]    0    1    0
```

Quaternion form

Quaternions are unit vectors in $\mathbb{R}^4$ that can be expressed as the sum of one real entry and three imaginary parts

$$\boldsymbol{Q} = x_1 + x_2 i + x_3 j + x_4 k \tag{5}$$

where $i^2 = j^2 = k^2 = ijk = -1$. We can write $\boldsymbol{Q} = (s, \boldsymbol{V})$ as tuple of the scalar s for coefficient $\mathbf{1}$ and vector $\boldsymbol{V}$ for the imaginary coefficients, i.e. $s = x_1$ and $\boldsymbol{V} = (x_2, x_3, x_4)$.

A rotation around axis $\boldsymbol{U}$ by angle r translates to $\boldsymbol{Q} = (s, \boldsymbol{V})$ with

$$s = \cos(r/2), \;\; \boldsymbol{V} = \boldsymbol{U} \sin(r/2).$$

Note that rotations in quaternion form are over-parametrized: $\boldsymbol{Q}$ and $-\boldsymbol{Q}$ represent equivalent rotations. This ambiguity has no impact on the distributional models, parameter estimation or inference methods to follow. Hence, for consistency, the **rotations** package only generates quaternions satisfying $x_1 \geq 0$. Data provided by the user does not need to satisfy this condition, however.

The `S3` class `"Q4"` is defined to manipulate rotations of this parameterization with all the same functionality available for the class `"SO3"`, e.g. `is.Q4` and `as.Q4` will test for, and if necessary, coerce to class `"Q4"`. Internally, a sample of n quaternions is stored in form of a $n \times 4$ matrix with each row a unit vector. Single quaternions are printed according to the representation in (5) (see example below) while a sample of size n is printed as a $n \times 4$ matrix with column names `Real`, `i`, `j` and `k` to distinguish between the real and complex components.

The following code creates the same rotation from the previous section in the form of a quaternion with the `as.Q4` function. This function works much the same way as the `as.SO3` function in terms of possible inputs but returns a vector of length four of the class `"Q4"`. We round the output to three digits for compactness.

```
as.Q4(U, r)

## 0.707106781186548 + 0 * i + 0.707106781186547 * j + 0 * k

as.Q4(as.SO3(U, r))

## 0.707106781186548 + 0 * i + 0.707106781186547 * j + 0 * k
```

Data generation

If the rotation $\boldsymbol{E}_i \in SO(3)$ from (1) has an axis $\boldsymbol{U}$ that is uniformly distributed on the unit sphere and an angle r that is independently distributed about 0 according to some symmetric distribution function then $\boldsymbol{E}_i$ is said to belong to the *uniform-axis random spin*, or *UARS*, class of distributions. From Bingham et al. (2009a) the density for $\boldsymbol{E}_i$ is given by

$$f(\boldsymbol{E}_i|\kappa) = \frac{4\pi}{3 - \operatorname{tr}(\boldsymbol{E}_i)} C\left(\operatorname{acos}\left\{\frac{\operatorname{tr}(\boldsymbol{E}_i) - 1}{2}\right\}\middle|\,\kappa\right) \tag{6}$$

where $C(\cdot|\kappa)$ is the distribution function connected to the angle of rotation r with concentration parameter κ. Members of the UARS family of distributions are differentiated based on the angular distribution.

The **rotations** package allows the user access to four members of the UARS class. Each member is differentiated by the distribution function for r: the uniform distribution on the circle, the matrix Fisher (Langevin, 1905; Downs, 1972; Khatri and Mardia, 1977; Jupp and Mardia, 1979), the Cayley (Schaeben, 1997; León et al., 2006) and a circular-von Mises-based distribution (Bingham et al., 2009a). Note: probability distribution functions on $SO(3)$ such as (6) are defined with respect to the Haar measure μ. That is, to compute the expectation of a random rotation $\boldsymbol{E}$ with corresponding misorientation angle r, one would differentiate $f(\boldsymbol{E}|\kappa)$ with respect to $\partial\mu$ over $SO(3)$ where $\partial\mu = [1 - \cos(r)]\partial r/2\pi$ and

∂r is the Lebesgue measure. Because the Haar measure acts as the uniform distribution on $SO(3)$, i.e. $\mu(\boldsymbol{E}) = 1$ for all $\boldsymbol{E} \in SO(3)$, the angular distribution $C(r|\kappa) = [1 - \cos(r)]/(2\pi)$ is referred to as the uniform distribution for misorientation angles and has been included in the **rotations** package under the name `-haar`.

The spread of the Cayley, matrix Fisher and circular-von Mises distributions is controlled by the concentration parameter κ. Note that concentration is a distribution specific parameter and not compatible across different distributions. To make comparisons across distribution possible we also allow for specification of the circular variance, which is defined as $\nu = 1 - E[\cos(r)]$ where $E[\cos(r)]$ is often referred to as the *mean resultant length* (Fisher, 1996). The form of each angular distribution along with the circular variance as a function of the concentration parameter is given in Table 1.

Name	Density $C(r\|\kappa)$	Circular variance ν	Function
Uniform	$\frac{1-\cos(r)}{2\pi}$	$\frac{3}{2}$	`-haar`
Cayley	$\frac{\Gamma(\kappa+2)(1+\cos r)^{\kappa}(1-\cos r)}{2^{(\kappa+1)}\sqrt{\pi}\Gamma(\kappa+1/2)}$	$\frac{3}{\kappa+2}$	`-cayley`
matrix Fisher	$\frac{[1-\cos(r)]\exp[2\kappa\cos(r)]}{2\pi[\mathrm{I}_0(2\kappa)-\mathrm{I}_1(2\kappa)]}$	$\frac{3\mathrm{I}_0(2\kappa)-4\mathrm{I}_1(2\kappa)+\mathrm{I}_2(2\kappa)}{2[\mathrm{I}_0(2\kappa)-\mathrm{I}_1(2\kappa)]}$	`-fisher`
circular-von Mises	$\frac{\exp[\kappa\cos(r)]}{2\pi\mathrm{I}_0(\kappa)}$	$\frac{\mathrm{I}_0(\kappa)-\mathrm{I}_1(\kappa)}{\mathrm{I}_0(\kappa)}$	`-vmises`

Table 1: Circular densities and circular variance ν; $I_i(\cdot)$ represents the modified Bessel function of order i and $\Gamma(\cdot)$ is the gamma function.

For a given concentration `d`, `p` and `r` take the same meaning as for the more familiar distributions such as `dnorm`. To simulate a sample of $SO(3)$ data, the `ruars` function takes arguments `n`, `rangle`, and `kappa` to specify the sample size, angular distribution and concentration as shown below. Alternatively, one can specify the circular variance ν. Note: circular variance takes precedence over concentration in case both are provided. The `space` argument determines the parameterization to form. Note that if a single rotation is printed, then the rotation is printed as a 3×3 matrix as demonstrated in Section 2.2.1. If a sample of rotations is printed then the $n \times 9$ matrix is printed along with column titles that specify which element of the matrix each row corresponds to. For example, the $\boldsymbol{R}_{\{1,1\}}$ element of a rotation matrix is printed under the column heading `R11` as illustrated below. Note that we rounded the output to three digits for compactness.

```
Rs <- ruars(n = 20, rangle = rcayley, kappa = 1, space = "SO3")
Qs <- ruars(n = 20, rangle = rcayley, kappa = 1, space = "Q4")
Rs <- ruars(n = 20, rangle = rcayley, nu = 1, space = "SO3")
Qs <- ruars(n = 20, rangle = rcayley, nu = 1, space = "Q4")
head(Rs, 3)

##          R11    R21    R31    R12    R22    R32    R13    R23    R33
## [1,] -0.3229 -0.565 -0.759  0.934 -0.320 -0.159 -0.153 -0.760  0.631
## [2,]  0.5825  0.195  0.789  0.557 -0.802 -0.214  0.592  0.564 -0.576
## [3,]  0.0683  0.940 -0.334 -0.629  0.300  0.717  0.774  0.161  0.612
```

Data analysis

In this section we present functions in the **rotations** package to compute point estimates and confidence regions for the central orientation $\boldsymbol{S}$.

Estimation of central orientation

Given a sample of n observations $\boldsymbol{R}_1, \ldots, \boldsymbol{R}_n$ generated according to (1) the **rotations** package offers four built-in ways to estimate the central orientation $\boldsymbol{S}$ from a frequentest perspective. These estimators are either Riemannian- or Euclidean-based in geometry and either using the L_2 or L_1 norm, i.e. mean- or median-type. We briefly discuss how the choice of geometry affects estimation of $\boldsymbol{S}$.

The choice of geometry results in two different metrics to measure the distance between rotation matrices $\boldsymbol{R}_1$ and $\boldsymbol{R}_2 \in SO(3)$. The Euclidean distance, d_E, between two rotations is defined by

$$d_E(\boldsymbol{R}_1, \boldsymbol{R}_2) = \|\boldsymbol{R}_1 - \boldsymbol{R}_2\|_F$$

where $\|\boldsymbol{A}\|_F = \sqrt{\mathbf{tr}(\boldsymbol{A}^\top \boldsymbol{A})}$ denotes the Frobenius norm and $\mathbf{tr}(\cdot)$ denotes the trace of a matrix. It follows $d_E(\cdot,\cdot)$ is an extrinsic distance measure. The Euclidean distance between two rotation matrices corresponds to the length of the shortest path in $\mathbb{R}^{3\times 3}$ that connects them.

Estimators based on the Euclidean distance form the class of *projected* estimators. The name is derived from the algorithms used to compute these estimators, which find the generic 3×3 matrix that minimizes the loss function then projects that matrix into $SO(3)$. For an object with class `"SO3"` the `median` or `mean` function with argument `type="projected"` will return a 3×3 matrix in $SO(3)$ that minimizes the first- or second-order loss function, respectively.

By staying in the Riemannian space $SO(3)$, called the intrinsic approach, the natural distance metric becomes the Riemannian (or geodesic) distance, d_R, which for two rotations $\boldsymbol{R}_1, \boldsymbol{R}_2 \in SO(3)$ is defined as

$$d_R(\boldsymbol{R}_1, \boldsymbol{R}_2) = \frac{1}{\sqrt{2}}\|\mathrm{Log}(\boldsymbol{R}_1^\top \boldsymbol{R}_2)\|_F = |r|,$$

where $\mathrm{Log}(\boldsymbol{R})$ denotes the logarithm of $\boldsymbol{R}$ defined in (4) and $r \in [-\pi, \pi)$ is the misorientation angle of $\boldsymbol{R}_1^\top \boldsymbol{R}_2$. The Riemannian distance corresponds to the length of the shortest path that connects $\boldsymbol{R}_1$ and $\boldsymbol{R}_2$ *within* the space $SO(3)$. For this reason, the Riemannian distance is often considered the more natural metric on $SO(3)$. As demonstrated in Stanfill et al. (2013), the Euclidean and Riemannian distances are related by $2\sqrt{2}\sin[d_R(\boldsymbol{R}_1, \boldsymbol{R}_2)/2] = d_E(\boldsymbol{R}_1, \boldsymbol{R}_2)$.

The distance between two objects of class `"SO3"`, e.g. `R1` and `R2`, is computed by the function `rot.dist`. The argument `method` specifies which type of distance to compute: the `"extrinsic"` option (default) will return the Euclidean distance and the `"intrinsic"` option will return the Riemannian distance. If only one object `R1` is provided then the second object `R2` is set to the 3×3 identity matrix. If `R1` is an $n \times 9$ matrix representing a sample of rotations, then `rot.dist` will return a vector of length n where the ith element represents the specified rotational distance between `R2` and the ith row of `R1`.

Estimators based on the Riemannian distance metric are called `geometric` estimators because they preserve the geometry of $SO(3)$. These can be computed using the `mean` and `median` functions with the argument `type="geometric."` Table 2 summarizes the four estimators including their formal definition and how they can be computed.

The estimators in Table 2 find estimates based on minimization of L_1- and L_2-norms in the chosen geometry. The function `gradient.search` provides the option to optimize for any other arbitrary minimization criterion. As the name suggests, the minimization is done along the gradient of the minimization function in the rotation space. Starting from an initial, user-specified rotation, the algorithm finds a (local) minimum by stepping iteratively in the direction of the steepest descent. Step size is regulated internally by adjusting for curvature of the minimization function.

We highlight this process in the example below. The function `L1.error` is defined to minimize the intrinsic L_1-norm, the result from the optimization should therefore agree with the geometric median of the sample. In fact, the difference between the two results is at the same level as the minimal difference (`minerr`) used for convergence of the gradient search. What is gained in flexibility of the optimization is, of course, paid for in terms of speed: the built-in median function is faster by far than the gradient search.

```r
# error function definition
L1.error <- function(sample, Shat) {
    sum(rot.dist(sample, Shat, method = "intrinsic", p = 1))
}
cayley.sample <- ruars(n = 50, rangle = rcayley, nu = 1, space = "SO3")
# gradient based optimization
system.time(SL1 <- gradient.search(cayley.sample, L1.error))

##    user  system elapsed
##   3.765   0.004   3.771

# in-built function
system.time(S <- median(cayley.sample, type = "geometric"))

##    user  system elapsed
##   0.003   0.000   0.003
```

```
rot.dist(S, SL1$Shat)

## [1] 1.08e-05
```

Estimator name	Definition	Code
Projected Mean	$\widehat{\boldsymbol{S}}_E = \underset{\boldsymbol{S}\in SO(3)}{\operatorname{argmin}} \sum_{i=1}^{n} d_E^2(\boldsymbol{S}, \boldsymbol{R}_i)$	`mean(Rs,type = "projected")`
Projected Median	$\widetilde{\boldsymbol{S}}_E = \underset{\boldsymbol{S}\in SO(3)}{\operatorname{argmin}} \sum_{i=1}^{n} d_E(\boldsymbol{S}, \boldsymbol{R}_i)$	`median(Rs,type = "projected")`
Geometric Mean	$\widehat{\boldsymbol{S}}_R = \underset{\boldsymbol{S}\in SO(3)}{\operatorname{argmin}} \sum_{i=1}^{n} d_R^2(\boldsymbol{S}, \boldsymbol{R}_i)$	`mean(Rs,type = "geometric")`
Geometric Median	$\widetilde{\boldsymbol{S}}_R = \underset{\boldsymbol{S}\in SO(3)}{\operatorname{argmin}} \sum_{i=1}^{n} d_R(\boldsymbol{S}, \boldsymbol{R}_i)$	`median(Rs,type = "geometric")`

Table 2: A summary of the estimators included in **rotations** package. `Rs` is a sample of n rotations with class `"SO3"` or `"Q4"` as generated in the Section 2.3.

Confidence regions

Asymptotic results for the distribution of the projected mean $\widehat{\boldsymbol{S}}_E$ and median $\widetilde{\boldsymbol{S}}_E$, see Table 2, can be used to construct confidence regions for the central orientation $\boldsymbol{S}$. In the literature two approaches are taken to derive the limiting distribution of the vector that generates to the centered estimator. More specifically, the vector $\widehat{\boldsymbol{h}}$ satisfying

$$\exp[\boldsymbol{\Phi}(\widehat{\boldsymbol{h}})] = \boldsymbol{S}^\top \widehat{\boldsymbol{S}}_E$$

has been shown to have a multivariate normal distribution. The first employs results about matrix decompositions while the other takes a moment-based approach. A brief summary of these methods is given next.

Prentice (1986) used results found in Tyler (1981) and the fact that $\widehat{\boldsymbol{S}}_E$ is a function of the spectral decomposition of $\overline{\boldsymbol{R}} = \sum_{i=1}^{n} \boldsymbol{R}_i/n$ in order to justify a multivariate normal limiting distribution for the scaled vector $\sqrt{n}\,\widehat{\boldsymbol{h}}$. Unsatisfied with the coverage rate achieved by Prentice (1986), Fisher et al. (1996) proposed a pivotal bootstrap procedure that results in coverage rates closer to the nominal in even small samples. Note that $\widetilde{\boldsymbol{S}}_E$ cannot be expressed as a function of the sample spectral decomposition, therefore this approach is not applicable.

It has also been shown that both estimators $\widehat{\boldsymbol{S}}_E$ and $\widetilde{\boldsymbol{S}}_E$ are M-estimators allowing for a moment-based approach to confidence region estimation. Chang and Rivest (2001) used this moment-based approach to estimate a confidence region for the central orientation based on the normal limiting distribution. In an unpublished report by Drs. Zhang and Nordman a pivotal bootstrap approach was implemented to improve coverage rates in small samples.

These six methods are available through the wrapper function `region`. They are differentiated based on the `method`, `type` and `estimator` arguments. Set `estimator="mean"` or `estimator="median"` to estimate a region based on $\widehat{\boldsymbol{S}}_E$ or $\widetilde{\boldsymbol{S}}_E$, respectively. For $\widehat{\boldsymbol{S}}_E$ one can choose `method="eigen"` for the eigenvector-based methods or `method="moment"` for the moment-based approach. Since the eigenvector-based methods cannot be applied to $\widetilde{\boldsymbol{S}}_E$ an error is returned if `method="eigen"` and `estimator="median"` are combined. A bootstrap version of the specified method is implemented if `type="bootstrap"` or the normal limiting distribution can be chosen with `type="theory"`. If a bootstrap type region is specified one can additionally specify the bootstrap sample size with the `m` argument, which is set to 300 by default. Regardless of the method and type chosen a single value is returned on the interval $(0, \pi]$. This value corresponds to the radius of the confidence region centered at each of the axes of the respective estimators.

In the example code below a sample of $n = 50$ rotations are drawn from the Cayley-UARS($\boldsymbol{I}_{3\times3}, \kappa = 10$) distribution then the four types of moment-based confidence regions are constructed. For a graphical representation of this dataset along with an interpretation of the confidence regions see Figure **??**.

```
Rs <- ruars(50, rcayley, kappa = 10)
region(Rs, method = "moment", type = "theory", estimator = "mean", alp = 0.05)

## [1] 0.157

region(Rs, method = "moment", type = "bootstrap", estimator = "mean", alp = 0.05,
    m = 300)

## [1] 0.155

region(Rs, method = "moment", type = "theory", estimator = "median", alp = 0.05)

## [1] 0.182

region(Rs, method = "moment", type = "bootstrap", estimator = "median", alp = 0.05,
    m = 300)

## [1] 0.225
```

Bayesian Methods

The Bayesian point and credible region estimation functions in the **rotations** package are based on the Metropolis-Hastings within Gibbs algorithm described in Bingham et al. (2009b) and Bingham et al. (2010b). With observations $\boldsymbol{R}_1, \ldots, \boldsymbol{R}_n \in SO(3)$, tuning parameters ϕ and σ^2, and starting values $\boldsymbol{S}^0 \in SO(3)$ and κ^0, a draw from the posterior distribution for $(\boldsymbol{S}^j, \kappa^j)$, $j = 1, \ldots, m$ is generated by the following.

1. Generate $\boldsymbol{S}^{j*} \sim F(\boldsymbol{S}^{j-1}, \phi)$ as a proposal for $\boldsymbol{S}^j$.
2. Compute $r_j^1 = g_n(\boldsymbol{S}^{j*}, \kappa^{j-1})/g_n(\boldsymbol{S}^{j-1}, \kappa^{j-1})$.
3. Generate $W_j^1 \sim \text{Bernoulli}(\min(1, r_j^1))$ and let $\boldsymbol{S}^j = W_j^1 \boldsymbol{S}^{j*} + (1 - W_j^1)\boldsymbol{S}^{j-1}$.
4. Generate $\log(\kappa^{j*}) \sim N(\log(\kappa^{j-1}), \sigma^2)$, with κ^{j*} as a candidate for κ^j.
5. Compute $r_j^2 = g_n(\boldsymbol{S}^j, \kappa^{j*})\kappa^{j*}/g_n(\boldsymbol{S}^j, \kappa^{j-1})\kappa^{j-1}$.
6. Generate $W_j^2 \sim \text{Bernoulli}(\min(1, r_j^2))$ and let $\kappa^j = W_j^2 \kappa^{j*} + (1 - W_j^2)\kappa^{j-1}$.

The forms of the functions $F(\boldsymbol{S}, \phi)$ and $g_n(\boldsymbol{S}, \kappa)$ vary with the distributional model assumed for the data as well as the priors chosen for the model parameters. The function $F(\boldsymbol{S}, \phi)$ is the distribution from which proposal values for the central orientation parameter are drawn and is usually set to be the same distribution function used in the data likelihood. The $g_n(\boldsymbol{S}, \kappa)$ function is the product of the posterior distributions and the likelihood; for its full form assuming non-informative priors and a circular-von Mises or matrix Fisher likelihood see Bingham et al. (2009b) and Bingham et al. (2010a), respectively. As these results have not appeared for the Cayley distributions then we describe it here.

For the Cayley distribution it can be show the Jeffreys prior for κ is given by

$$\pi(\kappa) = \sqrt{\psi'(\kappa + 0.5) - \psi'(\kappa + 2)}$$

where $\psi'(x) = \partial^2 \log[\Gamma(x)]/\partial x^2 = \sum_{k=0}^{\infty}(x+k)^{-2}$ is the tri-gamma function. A uniform prior on $SO(3)$ is placed on $\boldsymbol{S}$, that is $\pi(\boldsymbol{S}) = 1$ with respect to the Haar measure for all $\boldsymbol{S} \in SO(3)$. It follows the posterior distribution for $\boldsymbol{S}$ and κ is proportional to

$$g_n(\boldsymbol{S}, \kappa) = \sqrt{\psi'(\kappa + 0.5) - \psi'(\kappa + 2)} \left[\frac{\pi^{1/2}\Gamma(\kappa + 2)}{\Gamma(\kappa + 0.5)}\right]^n \prod_{i=1}^{n} \left\{\frac{1}{2} + \frac{1}{4}\left[\text{tr}(\boldsymbol{S}^\top \boldsymbol{R}_i) - 1\right]\right\}^{\kappa}.$$

The function `MCMCSO3` implements the algorithm described above using non-informative priors for the parameters. In particular a uniform prior on the space of rotations $SO(3)$ for $\boldsymbol{S}$ and Jeffreys prior for the concentration κ are chosen. The function returns a list consisting of draws from the posterior distributions for $\boldsymbol{S}$ and κ along with acceptance rates for both parameters. The likelihood is differentiated by the `type` argument with possible options `"Cayley"`, `"Fisher"` or `"Mises"`. Initial values for the central orientation and concentration parameters are set with the `S0` and `kappa0` arguments. The proposal distributions from which new proposal parameter values are drawn are controlled by the tuning parameters `tuneS` and `tuneK`. These values correspond to the parameters ϕ and σ in steps 1 and 4 of the algorithm, respectively. Finally, `burn_in` determines the burn in period and `m` the final number of draws for each parameter.

Bayesian point estimates for $\boldsymbol{S}$ and κ are computed by the function `bayes.mean`. In particular the posterior mode of the central orientation distribution and the posterior mean of the concentration distribution are returned. Bayesian credible regions are available with the function `bayesCR`, which returns a list consisting of the mode of the posterior distribution for the central orientation as well as the radius of the $100(1-\alpha)\%$ credible region centered at the posterior mode, where α is user-specified. Both `bayes.mean` and `bayesCR` require the same arguments as required by `MCMCSO3` with an additional argument for the α level of the region in `bayesCR`. Alternatively, in the `region` function one can set `moment="bayes"` and `estimator="mean"` to produce Bayesian credible regions where the likelihood is specified with the `type` argument.

Visualizations

The **rotations** package offers a novel method to visualize $SO(3)$ data within the framework of the **ggplot2** package (Wickham, 2009). Recall that rotations can be represented by orthogonal matrices meaning each axis, represented by each column, has length one and is perpendicular to the other axes. This allows us to illustrate rotation axes separately as a point on the surface of a unit sphere.

Calling the `plot` function with a `"SO3"` object will result in a sphere that represents a column of the supplied data. The `center` argument defines the center of the plot and is usually set to the identity rotation `id.SO3` or an estimate of the central orientation, e.g. `mean(Rs,method="projected")`. The user can specify which columns to visualize with the `col` argument with options 1, 2, and 3 representing the x-, y- and z- axes, respectively. If the data are concentrated in one part of the sphere then the `to_range` argument can be set to `TRUE` and the range of the plot is set to the area of the sphere that contains the points.

All of the four estimates of the central orientation can be plotted along with a sample of rotations. Setting the argument `estimates_show="all"` will display all four simultaneously. If only a few estimates are of interest then any combination of `"proj.mean"`, `"proj.median"`, `"geom.mean"` or `"geom.median"` are valid inputs. The estimators are indicated by color and a legend is provided, see Figure 1. Finally, the `mean_regions` and `median_regions` options allow the user to draw a circle on the surface of the sphere representing the confidence region for that axis, centered at $\widehat{\boldsymbol{S}}_E$ and $\widetilde{\boldsymbol{S}}_E$ respectively. If estimators are plotted along with the different regions then shapes represent the estimators and colors represent the region methods, see Figure 1. Given the sample of rotations generated in Section 2.4.2 the example code below illustrates the `plot` function for objects of class `"SO3"`. Figure 1 illustrates the results of these commands.

```r
plot(Rs, center = mean(Rs), show_estimates = "all")
plot(Rs, center = mean(Rs), show_estimates = "proj.mean", mean_regions = "all",
    alp = 0.05)
```

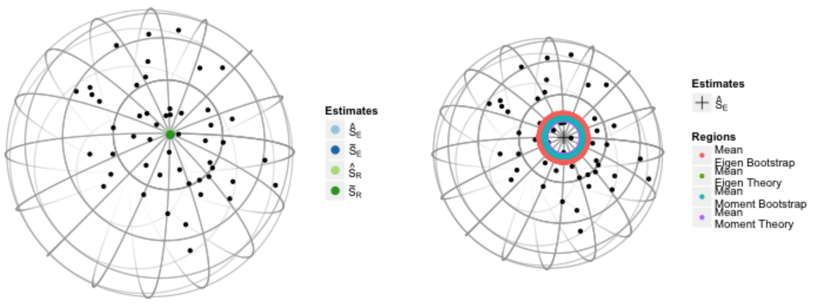

Figure 1: The x-axis of a random sample from the Cayley-UARS distribution with $\kappa = 1$, $n = 50$. All for point estimates are displayed in the left sub-figure and all three region methods along with the projected mean are on the right.

Datasets

Datasets drill and nickel are included in the **rotations** package to illustrate how the two representations of orientation data discussed here are used in practice. The drill dataset was collected to assess variation in human movement while performing a task (Rancourt, 1995). Eight subjects drilled into a metal plate while being monitored by infared cameras. Quaternions are used to represent the orientation of each subjects' wrist, elbow and shoulder in one of six positions. For some subjects several replicates are available. See Rancourt et al. (2000) for one approach to analyzing these data.

In the nickel dataset rotation matrices are used to represent the orientation of cubic crystals on the surface of a nickel sample measured with Electron Backscatter Diffraction. Each location on the surface of the nickel is identified by the xpos and ypos columns while the rep column identifies which of the fourteen replicate scans that measurement corresponds to. The last nine columns, denoted v1-v9, represent the elements of the rotation matrix at that location in vector form. See Bingham et al. (2009a, 2010a); Stanfill et al. (2013) for more details. In the example code below we illustrate how to load the datasets into an R session and some basic analysis techniques. For the drill dataset we estimate the central orientation of the measurements for Subject 1's wrist and for the nickel data we estimate the central orientation of the fourteen scans at location 1. For simplicity we round to three digits.

```
data(drill)
head(drill)

##   Subject Joint Position Replicate    Q1     Q2      Q3    Q4
## 1       1 Wrist        1         1 0.944 -0.192 -0.1558 0.217
## 2       1 Wrist        1         2 0.974 -0.120 -0.1111 0.158
## 3       1 Wrist        1         3 0.965 -0.133 -0.1406 0.177
## 4       1 Wrist        1         4 0.956 -0.134 -0.1152 0.233
## 5       1 Wrist        1         5 0.953 -0.199 -0.0611 0.222
## 6       1 Wrist        2         1 0.963 -0.159 -0.1270 0.177

Subj1Wrist <- subset(drill, Subject == "1" & Joint == "Wrist")
Subj1Wdata <- as.Q4(Subj1Wrist[, 5:8])
mean(Subj1Wdata)

## 0.987285847145712 - 0.0698275323542443 * i - 0.134149482591381 *
##     j + 0.0489355501813865 * k

data(nickel)
head(nickel[, 1:6])

##   xpos  ypos location rep     V1    V2
## 1    0 0.346        1   1 -0.648 0.686
## 2    0 0.346        1   2 -0.645 0.688
## 3    0 0.346        1   3 -0.645 0.688
## 4    0 0.346        1   4 -0.646 0.688
## 5    0 0.346        1   5 -0.646 0.686
## 6    0 0.346        1   6 -0.644 0.690

Location1 <- subset(nickel, location == 1)
Loc1data <- as.SO3(Location1[, 5:13])
mean(Loc1data)

##        [,1]   [,2]   [,3]
## [1,] -0.645 -0.286 -0.708
## [2,]  0.687 -0.623 -0.374
## [3,] -0.334 -0.728  0.599
```

Summary

In this manuscript we introduced the **rotations** package and demonstrated how it can be used to generate, analyze and visualize rotation data. The **rotations** package is compatible with the quaternion specific **onion** package by applying its `as.quaternion` function to a transposed `"Q4"` object. This gives the user access to a wide range of algebraic functions unique to quaternions. Also compatible with the **rotations** package is the **orientlib** package, which includes additional parameterizations of rotations. To translate rotation matrices generated by the **rotations** package into a form usable by the **orientlib** package, first coerce a `"SO3"` object into a matrix of the same dimension, i.e.$n \times 9$, then apply the `rotvector` function provided by the **orientlib** package. Quaternions, however, in the **orientlib** package are of the form $Q = x_1 i + x_2 j + x_3 k + x_4$, cf. (5), which may lead to confusion when translating quaternions between the **orintlib** package and either of the **onion** or **rotations** packages. Below is a demonstration of how quaternions and rotation matrices generated by the **rotations** package can be translated into a form usable by the **onion** and **orientlib** packages, respectively.

```
Qs <- ruars(20, rcayley, space = "Q4")
Rs <- as.SO3(Qs)
suppressMessages(require(onion))
onionQs <- as.quaternion(t(Qs))
suppressMessages(require(orientlib))
orientRs <- rotvector(matrix(Rs, ncol = 9))
```

Computational speed of the **rotations** package has been enhanced through use of the **Rcpp** and **RcppArmadillo** packages (Eddelbuettel, 2013; Eddelbuettel and Sanderson, 2013). In future version of the package we plan to extend the parameterization and estimator sections to include robust estimators currently being developed by the authors.

Bibliography

C. Agostinelli and U. Lund. *R package* `circular`*: Circular Statistics (version 0.4-3)*. CA: Department of Environmental Sciences, Informatics and Statistics, Ca' Foscari University, Venice, Italy. UL: Department of Statistics, California Polytechnic State University, San Luis Obispo, California, USA, 2011. URL `https://r-forge.r-project.org/projects/circular/`. [p1]

F. Bachmann, R. Hielscher, P. Jupp, W. Pantleon, H. Schaeben, and E. Wegert. Inferential statistics of electron backscatter diffraction data from within individual crystalline grains. *Journal of Applied Crystallography*, 43(6):1338–1355, 2010. [p1]

M. A. Bingham, D. J. Nordman, and S. B. Vardeman. Modeling and inference for measured crystal orientations and a tractable class of symmetric distributions for rotations in three dimensions. *Journal of the American Statistical Association*, 104(488):1385–1397, 2009a. [p1, 3, 9]

M. A. Bingham, S. B. Vardeman, and D. J. Nordman. Bayes one-sample and one-way random effects analyses for 3-d orientations with application to materials science. *Bayesian Analysis*, 4(3):607–629, 2009b. [p7]

M. A. Bingham, B. K. Lograsso, and F. C. Laabs. A statistical analysis of the variation in measured crystal orientations obtained through electron backscatter diffraction. *Ultramicroscopy*, 110(10): 1312–1319, 2010a. [p7, 9]

M. A. Bingham, D. J. Nordman, and S. B. Vardeman. Finite-sample investigation of likelihood and bayes inference for the symmetric von mises–fisher distribution. *Computational Statistics & Data Analysis*, 54(5):1317–1327, 2010b. [p7]

T. Chang and L.-P. Rivest. M-estimation for location and regression parameters in group models: A case study using stiefel manifolds. *Annals of statistics*, pages 784–814, 2001. [p6]

T. Downs. Orientation statistics. *Biometrika*, 59(3):665–676, 1972. [p3]

D. Eddelbuettel. *Seamless R and C++ Integration with Rcpp*. Springer, New York, 2013. ISBN 978-1-4614-6867-7. [p10]

D. Eddelbuettel and C. Sanderson. RcppArmadillo: Accelerating R with high-performance C++ linear algebra. *Computational Statistics and Data Analysis*, in press, 2013. URL `http://dx.doi.org/10.1016/j.csda.2013.02.005`. [p10]

N. I. Fisher. *Statistical Analysis of Circular Data*. Cambridge University Press, 1996. ISBN 0521568900. [p4]

N. I. Fisher, P. Hall, B.-Y. Jing, and A. T. Wood. Improved pivotal methods for constructing confidence regions with directional data. *Journal of the American Statistical Association*, 91(435):1062–1070, 1996. [p6]

R. K. S. Hankin. *onion: octonions and quaternions*, 2011. URL http://CRAN.R-project.org/package=onion. R package version 1.2-4. [p1]

M. Humbert, N. Gey, J. Muller, and C. Esling. Determination of a mean orientation from a cloud of orientations. application to electron back-scattering pattern measurements. *Journal of applied crystallography*, 29(6):662–666, 1996. [p1]

P. Jupp and K. Mardia. Maximum likelihood estimators for the matrix von Mises-Fisher and Bingham distributions. *The Annals of Statistics*, 7(3):599–606, 1979. ISSN 0090-5364. [p3]

C. Khatri and K. Mardia. The von Mises-Fisher matrix distribution in orientation statistics. *Journal of the Royal Statistical Society. Series B (Methodological)*, 39(1):95–106, 1977. ISSN 0035-9246. [p3]

P. Langevin. Magnetism and the theory of the electron. *Annales de chimie et de physique*, 5:70, 1905. [p3]

C. León, J. Massé, and L. Rivest. A statistical model for random rotations. *Journal of Multivariate Analysis*, 97(2):412–430, 2006. [p3]

D. Murdoch. Orientlib: An R package for orientation data. *Journal of Statistical Software*, 8(19), 2003. [p1]

H. Oh and D. Kim. *SpherWave: Spherical Wavelets and SW-based Spatially Adaptive Methods*, 2013. URL http://CRAN.R-project.org/package=SpherWave. R package version 1.2.2. [p1]

M. Prentice. Orientation statistics without parametric assumptions. *Journal of the Royal Statistical Society. Series B (Methodological)*, 48(2):214–222, 1986. ISSN 0035-9246. [p6]

Y. Qiu. *Isotropic Distributions for 3-Dimensional Rotations and One-sample Bayes Inference*. PhD thesis, Iowa State University, 2013. [p1]

D. Rancourt. *Arm Posture and Hand Mechanical Impedance in the Control of a Hand-held Power Drill,*. dissertation, MIT, 1995. [p9]

D. Rancourt, L. Rivest, and J. Asselin. Using orientation statistics to investigate variations in human kinematics. *Journal of the Royal Statistical Society: Series C (Applied Statistics)*, 49(1):81–94, 2000. ISSN 1467-9876. [p1, 9]

H. Schaeben. A simple standard orientation density function: The hyperspherical de la Vallée Poussin kernel. *Phys. Stat. Sol. (B)*, 200:367–376, 1997. [p3]

B. Stanfill, U. Genschel, and H. Hofmann. Point estimation of the central orientation of random rotations. *Technometrics*, 2013. doi: 10.1080/00401706.2013.826145. [p2, 5, 9]

D. E. Tyler. Asymptotic inference for eigenvectors. *The Annals of Statistics*, pages 725–736, 1981. [p6]

H. Wickham. *ggplot2: Elegant Graphics for Data Analysis*. Springer New York, 2009. ISBN 978-0-387-98140-6. URL http://had.co.nz/ggplot2/book. [p8]

Bryan Stanfill
Department of Statistics
Iowa State University
Ames, IA 50011
stanfill@iastate.edu

Heike Hofmann
Department of Statistics
Iowa State University
Ames, IA 50011
hofmann@mail.iastate.edu

Ulrike Genschel
Department of Statistics
Iowa State University
Ames, IA 50011
ulrike@mail.iastate.edu