INTRODUCTION

Animal movement is studied in fields as diverse as population ecology (Turchin, 1998), foraging theory and resource exploitation (Mouillot & Viale, 2001; Pyke, Pulliam, & Charnov, 1977), and conservation biology (Guessersian, Slip, Heller, & Harcourt, 2011). The analysis of trajectories, which are recorded tracks of individual animals moving through space and time, has been applied to answer questions about animal foraging, navigation, dispersal ecology, migration and behavioural mimicry (Codling, Plank, & Benhamou, 2008; Shamble, Hoy, Cohen, & Beatus, 2017). Animal trajectories are analysed as a set of time specific discrete locations, generally restricted to two spatial dimensions. As animal motion is continuous, trajectories are obtained by repeatedly sampling the location of an animal over time.

A variety of techniques has been used to sample locations, including GPS tracking coupled with satellite or radio telemetry (Mouillot & Viale, 2001), manual tracking (Angilletta, Roth II, Wilson, Niehaus, & Ribeiro, 2008), digitising from video of a moving animal (Shamble et al., 2017), harmonic radar (Mascanzoni & Wallin, 1986) and following tracks in snow (Fortin, Morales, & Boyce, 2005). Location sampling may occur at constant time intervals, or at biologically meaningful step "endpoints," such as landing sites of a foraging butterfly (Kareiva & Shigesada, 1983). Trajectories can be conveniently represented and analysed as a series of steps, each of which has a length and a direction. The steps correspond to vectors between consecutive locations in the sampled trajectory.

TRAJECTORY INDICES

Depending on the hypothesis being verified, different indices characterising a trajectory can be calculated. The concept of tortuosity (or, conversely, straightness) is intuitively an important characteristic of trajectories that is inversely related to the efficiency of navigation towards a destination and controls local searching intensity (Benhamou, 2004). This characteristic is also valuable in locomotor mimicry studies, allowing...
verification of whether a mimic displays a tortuous path to resemble its model, even though moving in a straight line would have allowed it to reach its destination faster. Tortuosity and straightness have been quantified in various ways, with the simplest straightness index being:

\[ d = \frac{D}{L} \quad (1) \]

where \( D \) is the straight-line distance between the start and end of the path, and \( L \) is the length of the path (Batschelet, 1981). \( d \) may be a reliable estimate of the efficiency of a directed path, but unsuitable for random paths (Benhamou, 2004). Batschelet (1981) considers \( d \) to be an easily calculable approximation to an alternative straightness index, \( r \), which is the mean vector of turning angles of a trajectory with constant step length.

Sinuosity is a measure originally defined by Bovet and Benhamou (1988) and later modified by Benhamou (2004), and may be used to reliably estimate the tortuosity of a random search path (Benhamou, 2004). Sinuosity is a function of both the mean cosine of turning angles and step length and varies between 0 (straight) and 1 (highly curved). Sinuosity is closely related to the maximum expected displacement, or \( E^a_{\text{max}} \), which is a dimensionless value that denotes the maximum expected displacement of a random path as a function of the number of steps (Cheung, Zhang, Stricker, & Srinivasan, 2007). Cheung et al. (2007) consider \( E^a_{\text{max}} \) to be superior to sinuosity as it depends on the distribution of turning angles rather than just the standard deviation. An alternative measure, \( E^p_{\text{max}} \), gives the maximum expected displacement in physical dimensions (Cheung et al., 2007):

\[ E^p_{\text{max}} = \mu_L E^a_{\text{max}} \quad (2) \]

where \( \mu_L \) is the mean step length of the trajectory.

A qualitatively different approach for describing sinuosity is the use of fractal dimension (or fractal D), which may identify scale-independent aspects of animal movement (Turchin, 1996). Fractal dimension relies on the fact that the measured length of a line (such as a coastline or an animal trajectory) depends on the length of the measuring stick used to measure it (Mandelbrot, 1967). One technique to measure the fractal dimension \( D \) of a line—the dividers method—is to repeatedly resample to different step lengths, then plot measured path length against step length. The slope of the log-log regression is then \( 1 - D \). Fractal \( D \) varies between 1 for a straight line and 2 for an infinitely variable curve. Nams (2006) has developed corrected methods to reduce bias in the original dividers method. The application of fractal \( D \) to animal trajectories may be inappropriate as animal trajectories may not be fractal curves (Benhamou, 2004; Turchin, 1996), (but see Mouillot & Viale, 2001). However, Nams (2005) considers it useful for assessing animals’ responses to environmental variation at different spatial scales.

Other measures of the shape of trajectories include directional change and directional autocorrelation. Directional change is measured as change in direction over time (Kitamura & Imafuku, 2015), hence differs from previous tortuosity/straightness measures by incorporating the speed of change, indicating how frequently and how fast an animal changes its direction of movement. The mean and standard deviation of directional change have been used to quantify nonlinearity and irregularity in the trajectories of butterflies (Kitamura & Imafuku, 2015). Directional autocorrelation is a means to measure regular changes in direction, or sinuoidal components in paths devised to detect similarities in the paths of ants and ant-mimicking spiders (Shamble et al., 2017) whose trajectories were characterised by regular, repeated curves with similar wavelengths. Applied to constant step path lengths, it is the function:

\[ C(\Delta_s) = \text{mean}(\cos(\theta_s - \theta_s + \Delta_s)) \quad (3) \]

where \( \theta_s \) is the angle of step \( s \), and \( \Delta_s \) varies from 1 to some upper limit. It is mainly applicable to relatively straight paths, as large bends may distort the analysis (Shamble et al., 2017). The values of \( C(\Delta_s) \) and \( \Delta_s \), at the first local minimum in \( C(\Delta_s) \) may be used to quantify the amplitude and wavelength of the shortest wavelength repeated sine-like shape in the trajectory (Shamble et al., 2017).

Speed, that is displacement per unit time, provides a measure of space usage and can be related to the distribution of resources (Laube & Purves, 2011). Speed—and changes in speed—can be used to identify different types of behaviour, for example resting versus foraging (Shamoun-Baranes, Bouten, Camphuysen, & Baaij, 2011). It can be derived from a trajectory change in location divided by time. Linear acceleration is change in speed divided by time, however noise in trajectory locations may result in a high level of noise in the derived speed, and an even higher level of noise in the derived linear acceleration (Laube & Purves, 2011). Consequently, it may be necessary to first smooth the trajectory, reducing high-frequency noise before deriving speed or linear acceleration. The Savitzky-Golay filter is designed to improve the signal-to-noise ratio, and maybe an appropriate filter to apply to a trajectory before calculating derivatives (Savitzky & Golay, 1964; Shamble et al., 2017). Many trajectory recording methods suffer from high-frequency noise, and for such trajectories, smoothing is applicable prior to any analysis.

While there is clearly no shortage of methods and indices for analysing trajectories, implementing them is sometimes not straightforward and may require advanced programming knowledge. Here, we describe the \texttt{trajr} R package that implements a collection of trajectory analysis methods and algorithms. \texttt{R} is an open-source and multi-platform programming language and environment which, together with a wealth of user-contributed packages, provides a wide variety of statistical, graphical and analytic techniques, and is highly extensible (R Core Team, 2016). \texttt{trajr} is also open-source, and builds on the strengths of \texttt{R} to provide a powerful, open, accessible, verifiable framework for trajectory analysis.

### 3 | THE TRAJR PACKAGE

\texttt{trajr} aims to simplify the analysis, manipulation and visualisation of 2-dimensional trajectories. It is written entirely in R, so does not require compilation, is easy to install, and highly portable. The stable release of \texttt{trajr} (version 1.0.0 at the time of writing) is available for direct installation from CRAN (http://cran.r-project.org/...
The trajr workflow is simple: read in one or more files containing trajectory data, creating objects of the R class Trajectory; optionally modify them (scale, rotate, resample to fixed step lengths); call functions to calculate characteristics of the trajectories; then visualise, evaluate or compare the characteristics, or undertake further statistical analyses using the capabilities of the R environment. trajr does not aid in the capture, creation or translation to discrete representations of trajectories. It also does not provide any general statistical methods, as they are already available within the R environment.

Objects of the class “Trajectory” are created by calling the function TrajFromCoords with a series of spatial locations, specified as (x, y) coordinates. The coordinates may have associated times, or times can be inferred on the assumption that the locations were sampled at constant time intervals. The Trajectory class inherits methods from data.frame, and is documented in the help page for TrajFromCoords. Trajectory objects should not be directly modified—to change the locations or times in a trajectory, either call the methods that exist for that purpose (e.g., TrajScale, TrajRotate, TrajReverse, TrajSmoothSG, TrajRediscretize), or obtain a set of modified (x, y, time) coordinates and call TrajFromCoords to create a new Trajectory. A trajectory trj can be visualised by calling plot(trj). With no additional arguments, the trajectory is plotted as a series of straight-line segments joining the locations, and a small filled circle is drawn at the start point (Figure 1).

trajr defines multiple functions to characterise trajectories, including several measures of tortuosity and speed (Table 1). Some trajectories may require additional processing before these functions are applied. For example, the straightness index $r$ (i.e.,
mean vector of turning angles (Batschelet, 1981) assumes a constant step length. To calculate \( r \) for a trajectory with a variable step length, it must first be resampled to a constant step length (a process called rediscretisation (Bovet & Benhamou, 1988), which can be achieved by calling the function `TrajRediscretize`. Similarly, before analysing a noisy trajectory, the function `TrajSmoothSG` can be called to create a smoothed trajectory by applying a Savitzky-Golay smoothing filter (Savitzky & Golay, 1964). Consult the help page for each function for any such requirements.

`trajr` provides a number of functions that return lower-level information about trajectories, which may be useful when calculating measures that have not been implemented within `trajr` (Table 2). As an example, consider an alternative measure of straightness, \( r \), which is the length of the mean vector of turning angles (assuming a constant step length; Batschelet, 1981). The following expression will calculate \( r \) for the trajectory `trj`, after resampling to a step length of 2:

\[
\text{Mod(TrajMeanVectorOfTurningAngles(TrajRediscretize(trj, 2)))}
\]

As trajectory analyses typically involve multiple trajectories, `trajr` provides several functions to simplify this situation. `TrajsBuild` assumes that paths are defined in multiple files. It reads the paths from the files, optionally smooths and scales them, then returns a list of `Trajectory` objects. `TrajsMergeStats` assists with constructing a data frame that characterises a list of trajectories, and simplifies the handling of some idiosyncratic R behaviour such as the treatment of NULL values. The R language has two subtly different ways to represent missing or unknown values (such as missing values in a spreadsheet), `NULL` and `NA`. `NULL` values are not allowed in vectors or data frames, so `TrajsMergeStats` converts `NULL` values to `NA`. Functions are also provided to aid in dealing with `NA` values prior to principal components analysis (`TrajsStatsReplaceNAs`) and for listing all step lengths in a list of `Trajectory` objects (`TrajStepLengths`).

`trajr` provides the facility to generate random trajectories to assist with theoretical studies, by calling the function `TrajGenerate`. Generation is based on the concept of discrete steps with random lengths and turning angles, both of which are generated using user-controllable distribution functions. This allows creation of trajectories with highly variable statistical characteristics, including correlated random walks, directed walks, Levy walks and Brownian motion (Figure 2).

### Table 2 `trajr` functions to provide low-level information about trajectories

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>TrajStepLengths</code></td>
<td>Returns the lengths of each step within the trajectory</td>
</tr>
<tr>
<td><code>TrajLength</code></td>
<td>Returns the total length of the trajectory (or a portion)</td>
</tr>
<tr>
<td><code>TrajDistance</code></td>
<td>Returns the straight-line distance from the start to the end of the trajectory (or a portion)</td>
</tr>
<tr>
<td><code>TrajAngles</code></td>
<td>Returns the turning angles (radians) of a trajectory</td>
</tr>
<tr>
<td><code>TrajMeanVectorOfTurningAngles</code></td>
<td>Returns the mean vector of the turning angles (Batschelet, 1981)</td>
</tr>
<tr>
<td><code>TrajExpectedSquareDisplacement</code></td>
<td>Returns the expected square displacement of a correlated random walk (Kareiva &amp; Shigesada, 1983)</td>
</tr>
</tbody>
</table>

### Figure 2

Three randomly generated trajectories, (a) correlated random walk, (b) directed walk, (c) Levy walk.

4 | **ALTERNATIVES TO TRAJR**

EthoVision is a well-established proprietary commercial software product that automates behavioural observations (L. P. J. J. Noldus, Spink, & Tegelenbosch, 2001). Primarily a video tracking
application, it also allows calculation of a range of parameters describing 2- or 3-dimensional trajectories extracted from video. Parameters include “tortuosity” (the inverse of the straightness index $d$ (Batschelet, 1981)), instantaneous (or “sample” velocity) or speed with built-in smoothing, instantaneous angular change and angular velocity, as well as various measures of behaviour (Noldus, 2018). Raw data can be exported for further analysis within other systems such as Microsoft Excel, SPSS or R (L. P. J. J. Noldus et al., 2001). In summary, EthoVision is intended for analysing behaviour from videos, with only limited capability for characterising trajectories, whereas trajr is focused on characterising trajectories obtained using any method. EthoVision is not open-source, it is not integrated into a general purpose statistical/analytic environment, and it is mainly intended for interactive use (although it does have a batch analysis mode), so is difficult to include as a component within a reproducible workflow.

adehabitatLT is an open-source R package for the analysis of animal movements (Calenge, Dray, & Royer-Carenzi, 2009). It provides a variety of functions to analyse movements, and is particularly concerned with habitat selection by animals as revealed by their movements. adehabitatLT does not directly provide high-level characterisations of trajectories, but rather provides a set of lower-level components, which could be used as a basis for such calculations.

5 | WORKED EXAMPLE: BEHAVIOURAL RESPONSES OF HUMPBACK WHALES TO WHALE-WATCHING VESSELS

Humpback Whales (*Megaptera novaeangliae* Borowski, 1781) change their behaviour in response to whale-watching traffic, with potential long-term negative impacts (Parsons, 2012). Behaviours affected by these disruptions include surfacing and diving, tail swiping and breaching, singing behaviour, swimming speed, swimming direction, and feeding or resting behaviours (Parsons, 2012).

Here, to illustrate some capabilities of trajr, we present part of an analysis of whale trajectories that were recorded during research into the effects of whale-watching vessels on whales (Kessler & Harcourt, 2013).

The trajectories were created by combining a vessel GPS track with manual observations of distance and direction of whales from the vessel. The result was multiple CSV files containing the time, latitude, longitude and behaviour (e.g., blow, breach) at each observation, with one file per trajectory (Figure 3). To begin the analysis, we defined a function to read a CSV trajectory file and convert the content to a form suitable for use with trajr.

```
ReadWhaleCSV <- function(filename) {
  # Read in the file
  coords <- read.csv(filename, current.char = "#", stringsAsFactors = FALSE)

  # As trajr requires dates and times to be expressed as a single numeric value, we utilised standard R functionality to convert the Time column from text such as "13-06-10 14:30" to the number of seconds since a standard reference time (referred to as the epoch).
  coords$Date <- strptime(coords$Time, "%Y-%m-%d %H:%M:%S")
  coords$Time <- as.numeric(coords>Date)

  # Whale locations are specified by their latitude and longitude, so they must be converted to a square coordinate system—here we use the sp R package (Bivand, Pebesma, Gomez-Rubio, & Pebesma, 2008) to project the data onto an Albers equal-area map projection
  library(sp)
  library(maptools)
  library(mapproj)
  library(raster)
}
```

FIGURE 3 Whale trajectories on a map of the Sydney region, centred on latitude -33.872241, longitude 151.353303. Whales travelling north average 6.5 km/hr along bearing ~28°, while whales travelling south average 3.2 km/hr along bearing ~190°. Trajectory data (Kessler & Harcourt, 2013), map data (Google Maps, 2018). [Colour figure can be viewed at wileyonlinelibrary.com]
and obtain x and y coordinates expressed as metres relative to a reference location (while the Albers projection is not strictly a square coordinate system, distortion is minimal in the region covered).

```r
# Create a SpatialPointsDataFrame with just longitude and latitude
pts <- SpatialPoints(coords[, c("Longitude", "Latitude")], proj4string = CRS("+proj=longlat")]
# Project to Albers equal-area
pts <- sp::spTransform(pts, CRS(albers.proj))
# Append transformed x & y columns onto the original coordinates
coords$x <- coordinates(pts)[, 1]
coords$y <- coordinates(pts)[, 2]
# Return the result
coords
```

Having defined the function, we then called `TrajsBuild`, specifying the names of the x, y, and time columns, to build a list of trajectories and apply smoothing.

```r
trjs <- TrajsBuild(files, spatialUnits = "m",
    csvReadFn = ReadWhaleCSV,
    csvReadDat = list(x = "x", y = "y", time = "Time"),
    smoothP = 3, smoothN = 2)
```

To quantify whale behaviour, we characterised the trajectories using several indices. For each trajectory, we calculated: frequency of breaching; frequency of tail swiping; mean speed; mean duration of "downtime" events that is time interval between surfacing events more than 120 s apart (Gulesserian et al., 2011); and two measures of tortuosity: the straightness index $d$ (trajr function `TrajStraightness`) (Batschelet, 1981); and corrected sinuosity (trajr function `TrajSinuosity2`) (Benhamou, 2004). `trajr` does not provide functions to calculate whale-specific indices (frequency of breaching and tail swiping, downtime), however, they can be calculated using R and, once calculated, are treated in the same way as all other indices. We then classified trajectories according to whether more than three boats were within 300 m of the whale at any point during the recording. We defined the R function `indicesForTrj` to calculate all indices for a single trajectory, then called the function `TrajsMergeStats`, specifying the list of trajectories and the name of the function (`indicesForTrj`) as arguments.

```r
indicesForTrj <- function(trj) {
    duration <- trj$Duration(trj)
    meanSpeed <- TrajLength(trj) / duration
    
    # A trajectory consists of possibly many observations during each up-time,
    # and no observations during a downtime. We assume a dive is any down time
    # longer than 120 seconds, following Gulesserian et al. (2011)
    diveUOTrj <- 120
    stepTimes <- diff(trj$DisplacementTime)
    downTimes <- stepTimes[stepTimes > diveUOTrj]
    
    # Index values:
    list(
        # Northern or southern migration
        isNorthern = trj$NorthernMigration(trj),
        # Were there ever more than 3 boats within 300 m 
        # Too many boats = max(trj$Boats...300 + trj$Boats...100) > 3,
        # Number of breaches / sec
        BreachFreq = length(grep(".\^breach\.", trj$Behaviour, ignore.case = TRUE)) / duration,
        # Number of tail swipes / sec
        TailSwipeFreq = length(grep(".\^swipe\.", trj$Behaviour, ignore.case = TRUE)) / duration,
        # Mean speed (km/hr) along entire trajectory
        MeanSpeed = sum(xN/m(xN*xN) / meanDownTime),
        # Mean downtime
        MeanDowntime = ifelse(length(downTimes) == 0, 0, mean(downTimes)),
        # Straightness
        Straightness = TrajStraightness(trj),
        # Sinuosity
        Sinuosity = TrajSinuosity2(trj)
    )
}
```

The result (in the variable named `indices`) is a data frame with a row for each trajectory, and a column for each index (Table 3). The analytic capabilities of R can now be applied to calculate appropriate statistics on this data frame.

### 6 OTHER EXAMPLES OF USE

Another example of `trajr` usage is the comparison of trajectories between models and mimics. To increase its chances of avoiding predator attack, a mimic may imitate not only the morphology of a model, but also its behaviour, including movement. `trajr` has been used to calculate and compare trajectory indices of mimetic clearwing moths (family Sesiidae) and their models: bees and wasps (Skowron Volponi, McLean, Volponi, & Dudley, 2018) providing...
evidence for flight path mimicry. This method has also been used to investigate imperfect locomotor mimicry in ant-mimicking spiders (McLean, 2017). Here, we present part of the clearwing moth analysis to demonstrate the use of trajr. The trajectories were manually digitised from high-speed video, producing CSV files with x, y coordinates (in pixels) for each frame. The scale of each video was calculated using a tape measure which was included within the picture, and the frame rate of each video is known. The first step is to read the CSV files, derive times from the frame rate, scale coordinates to convert from pixels to metres, and smooth the resultant trajectories.

```r
# file_list.csv: file name - list of CSV files with trajectory coordinates
# file_list.csv: frame rate of each trajectory video [frames/sec]
# file_list.csv: scale of each trajectory video [pixels/metre]
trajs <- TrajBuild(file_list.csv, file.name, file_list.csv, file_list.csv, "m",
                    rootDir = "data/clearwing-moths", smoothP = 3, smoothNH = 101)
```

The next step is to define the indices of interest, by creating a function which calculates the indices for a single trajectory (named clearwing_traj_indices in this example). Most indices can be calculated using trajr functionality, however longest hover time (the longest time of a single hovering event, that is flight slower than a determined speed, displayed by an animal) within a trajectory required some additional R code to calculate (the function longest_hover_time in the example).

```r
# Any slower than this speed is considered to be hovering
MAX_HOVER_SPEED <- 0.1

# Returns the longest single hovering time for a trajectory, or 0 if there is no hovering
longest_hover_time <- function(traj) {
  intervals <- TrajSpeedIntervals(traj, slowerThan = MAX_HOVER_SPEED)
  # Return time of longest interval, or 0 if there are no intervals
  max(interval, intervals$duration)
}

clearwing_traj_indices <- function(traj) {
  derive <- TrajDerivative(traj)
  resampled <- TrajResample(traj, .001)
  corr <- TrajAutoCorrelation(resampled, roundthrow(resampled) / 4)
  first_min <- TrajMin(firstMinimum(corr))
  list(
    longest_hover_time = longest_hover_time(traj),
    mean_speed = mean(derive$speed),
    sd_speed = sd(derive$speed),
    min_speed = min(derive$speed),
    max_speed = max(derive$speed),
    skewness = TrajSkewness(traj),
    kurtosis = TrajKurtosis(traj),
    maxpos = max(traj$pos),
    first_min_C = first_min("C"),
    Enmax <- TrajMax(traj),
    directional_change_mean = mean(traj$directionalChange(traj)),
    directional_change_sd = sd(traj$directionalChange(traj))
  )
}
```

Finally, call TrajMergeStats, which returns a data frame that contains a row for each trajectory and a column for each index.

```r
indices <- TrajMergeStats(trajs, clearwing_traj_indices)
```

To visualise the indices, the first two components from a principal component analysis were plotted, which reveals two clusters, one containing the trajectories of bee mimics and bees, and the other containing wasp mimics and wasps (Figure 4), in agreement with the conclusions of Skowron Volponi et al. (2018).

7 | CONCLUSIONS

trajr makes trajectory analysis simple and available to a wide range of researchers, including biologists without a programming background. As open-source, the implementation can be independently validated, corrected, and extended. Rather than attempt to provide an exhaustive end-to-end analysis solution, it provides building blocks that can be used to more easily create additional functionality, and implement additional or new algorithms. It allows for direct integration with the extensive statistical procedures provided by R and the user-supplied packages. As a scripted analysis, it is ideally suited to be part of a reproducible research workflow, which involves automating analyses from reading raw data through to producing publishable outputs such as plots and tables, or even entire manuscripts, without manual intervention (Stodden, Leisch, & Peng, 2014). In line with this approach, the R source code and raw data used to create all figures and analyses in this document, as well as the complete source code for both examples, is available online at GitHub (https://github.com/JimMcL/trajr-paper).

Further details on using the trajr package can be found in the package vignette that is available online from CRAN, or can be accessed from R (after the package has been installed) by running the command `vignette("trajr-vignette")`.

8 | CITATION OF METHODS IMPLEMENTED IN TRAJR

Many of the functions in trajr implement methods that are described in their original publications. Whenever this is the case, the
help pages for the functions identify the original publications. Users of the functions should cite the original sources as well as `trajr`.

ACKNOWLEDGEMENTS

DJM received funding from the Australian Research Council, Project ID: DP170101617. MASEV received funding through a doctoral scholarship ID 2016/20/T/NZB/00541 from National Science Centre (Poland). The whale research was funded by the Australian Research Council, LP07756840 to R. Harcourt and Bass & Flinders Cruises (Whale Watching Sydney), and was conducted under Macquarie Animal Ethics Approvals 2007/011 and 2010/025 and NSW Office of Environment and Heritage Scientific Permits S12600 and S12735. We would like to thank Tom Tregenza and one anonymous reviewer whose constructive comments allowed us to significantly improve the manuscript.

CONFLICT OF INTEREST

We declare that the authors have no conflict of interest.

ORCID

Donald James McLean [http://orcid.org/0000-0001-6229-7063]

REFERENCES


