A Guide to Pre-Processing High-Throughput Animal Tracking Data

Pratik Rajan Gupte[1, 2], Christine E. Beardsworth[2], Orr Spiegel[3], Emmanuel Lourie[4], Sivan Toledo[5], Ran Nathan[4], Allert I. Bijleveld[2]

1 Groningen Institute for Evolutionary Life Sciences, University of Groningen, The Netherlands.

2 Deptartment of Coastal Systems, NIOZ Royal Netherlands Institute for Sea Research, The Netherlands.

3 School of Zoology, Faculty of Life Sciences, Tel Aviv University, Tel Aviv 69978, Israel.

4 Movement Ecology Lab, Department of Ecology, Evolution, and Behavior, Alexander Silberman Institute of Life Sciences, The Hebrew University of Jerusalem, 91904 Jerusalem, Israel.

5 Blavatnik School of Computer Science, Tel Aviv University, Israel.

Correspondence Author

Pratik R. Gupte

Address

Groningen Institute for Evolutionary Life Sciences,

Nijenborgh 7/5172.0581 9747 AG Groningen

The Netherlands.

Email

pratikgupte16@gmail.com OR p.r.gupte@rug.nl

ORCID

https://orcid.org/0000-0001-5294-7819

Running Headline

Cleaning high-throughput animal tracking data

Abstract

1. Modern, high-throughput animal tracking studies collect increasingly large volumes of data at very fine temporal scales. At these scales, location error can exceed the animal's step size, confounding inferences from tracking data. 'Cleaning' the data to exclude positions with large location errors prior to analyses is one of the main ways movement ecologists deal with location errors. Cleaning data to reduce location error before making biological inferences is widely recommended, and ecologists routinely consider cleaned data to be the ground-truth. Nonetheless, uniform guidance on this crucial step is scarce.

2. Cleaning high-throughput data must strike a balance between rejecting location errors without discarding valid animal movements. Additionally, users of high-throughput systems face challenges resulting from the high volume of data itself, since processing large data volumes is computationally intensive and difficult without a common set of efficient tools. Furthermore, many methods that cluster movement tracks for ecological inference are based on statistical phenomena, and may not be intuitive to understand in terms of the tracked animal's biology.

3. In this article we introduce a pipeline to pre-process high-throughput animal tracking data in order to prepare it for subsequent analysis. We demonstrate this pipeline on simulated movement data to which we have randomly added location errors. We further suggest how large volumes of cleaned data may be synthesized into biologically meaningful 'residence patches'. We then use calibration data to show how the pipeline improves its quality, and to verify that the residence patch synthesis accurately captures animal space-use. Finally, turning to real tracking data from Egyptian fruit bats (*Rousettus aegyptiacus*), we demonstrate the pre-processing pipeline and residence patch method in a fully worked out example.

4. To help with fast implementations of our pipeline, and to help standardise methods, we developed the `R` package `atlastools`, which we introduce here. Our pre-processing pipeline and `atlastools` can be used with any high-throughput animal movement data in which the high data volume combined with knowledge of the tracked individuals' biology can be used to reduce location errors. The use of common pre-processing steps that are simple yet robust promotes standardised methods in the field of movement ecology and better inferences from data.

Keywords ATLAS, data cleaning, movement ecology, high-throughput tracking, *R* package *atlastools*, residence patch

1 Introduction

Tracking individual animals is the methodological mainstay of movement ecology, which seeks to link animal movement with internal causes, environmental factors, and resulting consequences (Holyoak et al., 2008; Nathan et al., 2008). Investigating fine-scale environmental and social drivers and the consequences of movement requires position data from many individuals at high temporal and spatial resolution. Such high-throughput tracking is possible using GPS tags (see recent examples in Harel et al., 2016; Papageorgiou et al., 2019; Strandburg-Peshkin et al., 2015), yet 'reverse-GPS' systems developed to track animals over land (MacCurdy et al., 2009, 2019; Toledo et al., 2014, 2016, 2020; Weiser et al., 2016) and in aquatic systems (Baktoft et al., 2019, 2017; Hussey et al., 2015; Jung et al., 2015) routinely produce high-throughput tracking data at much lower costs. Although high-resolution tracking inherently provides more detailed information about the true path of the tracked animal, high-throughput data presents a two-fold challenge to ecologists. First, the location error of each position may approach or exceed the true step size of the animal compared to low-resolution tracking with the same measurement error. This biases derived metrics such as speed and tortuosity (see Calenge et al., 2009; Hurford, 2009; Noonan et al., 2019; Ranacher et al., 2016). Additionally, the absolute location errors around a position introduces uncertainty when studying the location of an animal relative to fixed landscape features (e.g. roads) or mobile elements (e.g. other tracked individuals). Users have two main options to improve data quality: making inferences after modelling the system-specific location error (Fleming et al., 2014, 2020; Johnson et al., 2008; Jonsen et al., 2005, 2003; Patterson et al., 2008), or pre-processing data to clean it of positions with large location errors (Bjørneraas et al., 2010). The first approach may be limited by the animal movement models it can fit to the data (Fleming et al., 2014, 2020; Noonan et al., 2019), and may be beyond the computational capacity of common hardware, leading users to prefer data cleaning instead. When attempting data cleaning, the second challenge of high-throughput tracking reveals itself: the large number of observations themselves (Toledo et al., 2020; Weiser et al., 2016). Having large volumes of data to clean makes manual identification and removal of errors from individual tracks prohibitively time consuming, incentivising automation based on a protocol.

Pre-processing steps must justifiably discard large location errors, also called outliers, from tracks (analogous to reducing false positives) while avoiding the overzealous rejection of valid animal movements (analogous to reducing false negatives). How well researchers balance these imperatives has consequences for downstream analyses (Stine and Hunsaker, 2001). For instance, small-scale resource selection functions can easily infer spurious preference and avoidance effects when there is uncertainty about an animal's true position and movement (Visscher, 2006). Ecologists recognise that tracking data are imperfect observations of the underlying process of animal movement, yet they implicitly consider cleaned data equivalent to be the ground-truth. This assumption is reflected by popular statistical methods in movement ecology such as Hidden Markov Models (HMMs) (Langrock et al., 2012), stationary-phase identification methods (Patin et al., 2020), or step-selection functions (SSFs) (Avgar et al., 2016; Barnett and Moorcroft, 2008; Signer et al., 2017), which expect low location errors relative to real animal movement (i.e., a high signal-to-noise ratio). This makes the reproducible, standardised removal of location errors crucial to any animal tracking study. While gross errors are often removed by positioning-system algorithms, 'reasonable' errors often remain to confront end users (Fischler and Bolles, 1981; Ranacher et al., 2016; Weiser et al., 2016). Location errors may be exacerbated by the conditions of deployment, with systematic differences in location error among different landscape types (see D'Eon et al., 2002; Frair et al., 2004; Lewis et al., 2007). Standardised pre-processing steps should thus be general enough to tackle animal movement data recovered from a range of environments.

Despite the importance and ubiquity of reducing location errors in tracking data, movement ecologists lack formal guidance on this crucial step. Pre-processing protocols are often system-specific, or have limited availability, or may not be easily tractable for mainstream computing hardware and software. Furthermore, filtering out positions on their location error estimates may not be sufficient to recover a good track estimate when location error does not correspond to biological realism in movement (Ranacher et al., 2016; Weiser et al., 2016). This makes identifying and removing implausible movements from a track an important component of recovering true animal movement (Bjørneraas et al., 2010). Even after removing large location errors and evidently unrealistic movement, a track may comprise of positions that are distributed around the true animal location. Discarding these positions as 'untrue' would lead to excessive data loss, but treating them as 'real' locations would lead to unrealistic estimates of metrics such as speed (Noonan et al., 2019). The large data-volumes of high-throughput tracking allow for a neat solution: tracks can be 'smoothed' to reduce small location errors that have remained undetected. This large data volume becomes a challenge when users seek to examine animal space-use in relation to relevant environmental covariates, such as the predictors of prolonged residence in an area (see Bracis et al., 2018). First, statistical modelling using high-throughput data may suffer from pseudo-replication since both the response (e.g. step-length) and the predictors (e.g. resource landscape values) are spatio-temporally auto-correlated and hence non-independent (Aarts et al., 2008; Bijleveld et al., 2016; Fleming et al., 2014; Harel et al., 2016; Oudman et al., 2018). Second, fitting statistical models to datasets with many millions of observations may require significant innovation in mainstream statistical software (e.g. Wood et al., 2015), and users may instead want to reduce data volumes by thinning or clustering.

Here, we present a hands-on guide to pre-processing high-throughput tracking data, and demonstrate a relatively simple data cleaning pipeline to prepare these data for subsequent analyses (see Fig. 1). We take two important considerations into account, (1) that methods must be computationally efficient, and (2) that the pre-processing steps should be easily understood and reproduced. With these in mind we have developed the `R` package `atlastools` (Gupte, 2020; R Core Team, 2020), which implements important steps of the pre-processing pipeline. `R` is the computational environment of choice in movement ecology (Joo et al., 2020) and formalising tools as an `R` package improves portability and reproducibility (Marwick et al., 2018). `atlastools` functions are easy to use and understand, and rely on the efficiency of the `R` package `data.table` (Dowle and Srinivasan, 2020). We begin by showing how the pipeline can clean simulated tracks to which we have artificially introduced location error (Gurarie et al., 2017). Beginning with basic spatio-temporal filtering of positions, we cover filtering movement tracks, and reducing the effect of location error with a median smooth. Then, we suggest one solution to issues of computational tractability, which is to synthesise residence patches (*sensu* Barraquand and Benhamou, 2008; Bijleveld et al., 2016; Oudman et al., 2018) from clusters of spatio-temporally proximate positions, and so reveal important aspects of animal space use. Using calibration data from a manually transported ATLAS tag, we demonstrate how the residence patch method in `atlastools` accurately identifies areas of prolonged residence under real field conditions. Finally, we turn to data from Egyptian fruit bats (*Rousettus aegyptiacus*) tracked in the Hula Valley, Israel, to show a fully worked out example of the pre-processing pipeline and the residence patch method.

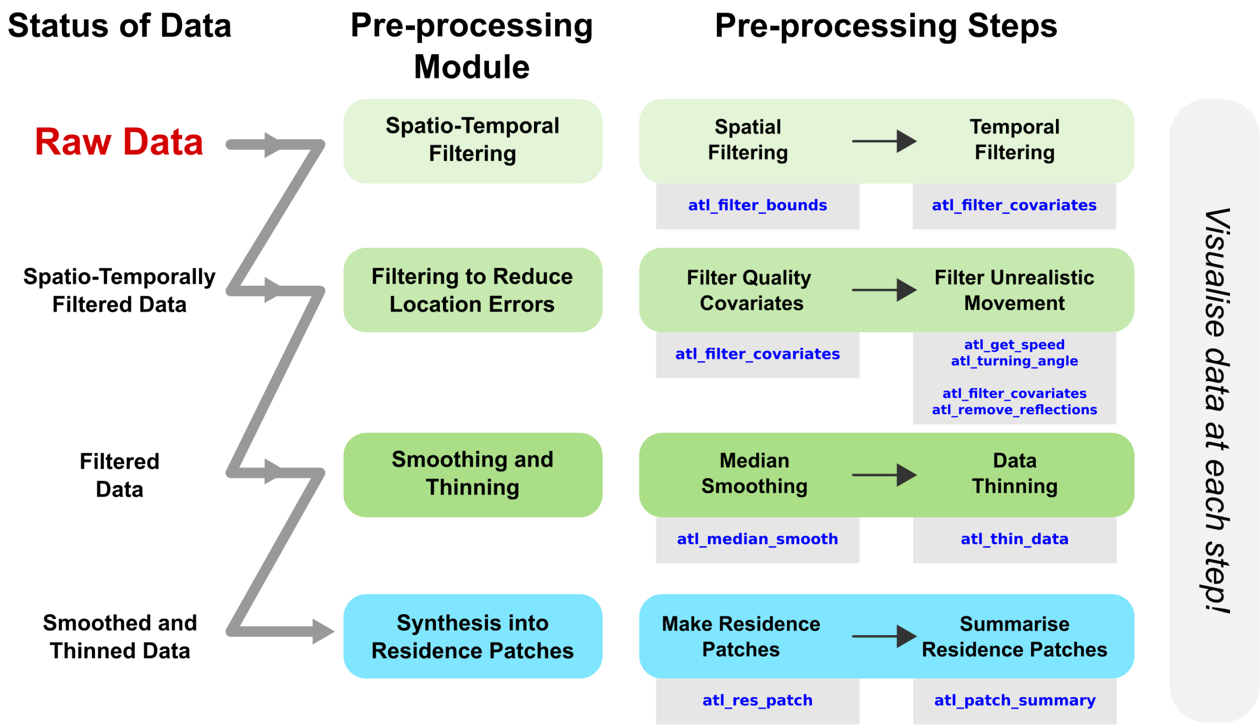

Figure 1. A general, modular pipeline for pre-processing high-throughput tracking data from raw localisations to cleaned data, and optionally into residence patches. Users should begin by plotting their data to determine its status, for instance whether it needs to be susbet within a certain time interval, or whether there are obvious location errors. Users can then pre-process their data using the appropriate module, following the pre-processing steps corresponding to each module. The `atlastools` function that may be used to implement each pre-processing step is shown in the grey boxes underneath each step. Users may use the pipeline in full or just the module most appropriate for their data. In all cases, users are strongly encouraged to visualise their data and scan it for location errors as a first step, and to visualise it along the way.

2 Pipeline Overview, Usage, and Simulating Data

2.1 The Pipeline and `atlastools`

We lay out a modular pipeline for pre-processing raw high-throughput tracking data using the `R` package `atlastools` (Fig. 1). While the pipeline and package were designed with ATLAS systems in mind, the principles and functions can be used with any high-throughput tracking data. Users may follow the pipeline in full, or implement the module most suitable for their data.

1. After data access, we first encourage users to visualise their data to check for evident location errors (Slingsby and van Loon, 2016).

2. Users can install `atlastools` using the `install_github` function from the `R` package `remotes`. The package can be installed using the `R` command

   `remotes::install_github("pratikunterwegs/atlastools")`.

   Package functions are prefixed '`atl_`', and rely heavily on the `data.table` package (Dowle and Srinivasan 2020). Certain functions modify the input data "in place", and users must work on a copy of their data to preserve both the original and cleaned data (details in the Supplementary Material). Since ATLAS systems typically cover an area of a few hundred square kilometres, `atlastools` assumes a coordinate system in metres, and users must convert their geographic coordinates to metres.

3. Users with a specific area or time of interest can use simple spatio-temporal range filters to select positions (see SPATIO-TEMPORAL FILTERING).

4. Next, users should reduce gross location errors by removing unreliable positions which may be identified by an error measure, or by the plausibility of associated movement metrics (see FILTERING TO REDUCE LOCATION ERRORS).

5. Users should then reduce small-scale location errors by applying a median smooth (see SMOOTHING AND THINNING DATA).

6. Users who need uniformly thinned data can then thin by either aggregating or resampling (see SMOOTHING AND THINNING DATA). At this stage, the data are ready for a number of popular statistical treatments such as Hidden Markov Model-based classification (Michelot et al., 2016).

7. Finally, users wishing to study prolonged animal residence in an area can classify their data into residence patches based on the movement ecology of their study system, after filtering out non-stationary positions (see SYNTHESISING MOVEMENT TRACKS INTO RESIDENCE PATCHES).

2.2 Simulating Movement Tracks

To demonstrate the pipeline, we simulated a realistic movement track of 5,000 positions (unbiased correlated velocity model; UCVM) using the `R` package `smoove` (Gurarie et al., 2017, see Fig. 2.a). We added three kinds of error to the simulated track: (1) normally distributed small-scale offsets to the X and Y coordinates independently, (2) normally distributed large-scale offsets to a random subset (0.5 %) of the positions, and (3) large-scale displacement of a continuous sequence of 300 of the 5,000 positions (indices 500 – 800) (Fig. 2.a). To demonstrate the residence patch method, we chose to simulate three independent rotational/advective correlated velocity movement (RACVM) tracks of 500 positions each ($\omega = 7$, initial velocity = 0.1, $\mu = 0$; see Gurarie et al. 2017), and connected them together with a roughly linear path (see Fig. 6.a). RACVM models approximate the tracks of soaring birds which circle on thermals over a relatively small area, and move between thermals ('thermalling'; Gurarie et al., 2017; Harel et al., 2016). This complex track structure provides a suitable challenge for the residence patch method and helps to demonstrate its generality.

3 Spatio-Temporal Filtering

3.1 Spatial Filtering Using Bounding Boxes and Polygons

First, data can be filtered to exclude positions outside the spatial bounds of a study area. This simply involves comparing position coordinates with the range of acceptable coordinates (the bounding box), and removing those positions outside them (Fig. 2.b; Listing 1). A bounding box filter retains data from within a subset of the tracking range without the need for a geospatial representation such as a shapefile. This can help remove unreliable data from a tracking system that is less accurate beyond a certain range (e.g. ATLAS; Beardsworth et al. *in prep.*). In some special cases, users may wish to *remove* positions inside a bounding box, either because movement behaviour within an area is not the focus of a study, or because positions recorded within an area are known to be erroneous. An example of the former is studies seeking to study transit behaviour between features which can be approximated by their bounding boxes. Instances of the latter are likely to be system specific, but are known from ATLAS systems (Bijleveld et al. *in prep.*). One drawback to bounding box filters is that they are restricted to rectangular areas. Users seeking to filter for areas with other geometries, such as a circular or irregular study area, need a geometric intersection between their data and a spatial representation of the area of interest (e.g. shapefile, geopackage, or `sf`-object in `R`). The `atlastools` function `atl_filter_bounds` implements bounding box and explicit spatial filters and accepts X and Y coordinate ranges, an `sf`-polygon object (Pebesma, 2018), or any combination of the three to filter the data (Listing 1). Multipolygon objects are supported, allowing data from different areas to be selected. When both coordinate ranges and a polygon are provided, the data is first filtered by the ranges and then the polygon. The boolean function argument `remove_inside` determines whether positions inside the bounds are retained (`FALSE`) or removed (`TRUE`).

```
filtered_data <- atl_filter_bounds(data = data,
                         x = "X", y = "Y",
                         x_range = c(x_min, x_max),
                         y_range = c(y_min, y_max),
                         sf_polygon = your_polygon,
                         remove_inside = FALSE)
```

Listing 1. The `atl_filter_bounds` function removes positions outside an area defined by coordinate ranges, a polygon, or all three (`remove_inside = FALSE`), or positions inside the area (`remove_inside = TRUE`). The arguments `x` and `y` determine which columns are considered the X and Y coordinates, the arguments `x_range` and `y_range` determine the acceptable range of coordinates in a reference system based in metres, while the `sf_polygon` argument allows the data to be filtered by the user specified `sf-(MULTI)POLYGON` object. `atl_filter_bounds` returns a filtered `data.table`, which must be saved as an object (here, `filtered_data`).

3.2 Temporal and Spatio-temporal Filters

Temporal filtering can exclude positions from periods during tracking when data are expected to be unreliable, either due to abnormal movement behaviour or poor tracking quality. An animal's movement may be non-representative when it has been fitted with a tracker but has not been released, leading to artificial stationary positions, or in the time shortly after release when its movement may be influenced by the stress of capture and handling (see example implementation in `amt`; Signer et al. (2019)). In these cases, all positions from a short time after release (e.g. 24 hours) can be excluded. Periods of poor tracking quality may result from system malfunctions, and temporal filters can be applied to exclude such chunks from the datastream. Temporal filters can be combined with spatial filters to exclude time-location combinations which would be unrealistic for the tracked animal. For example, red knots in the Dutch Wadden Sea congregate at communal roosts during high-tide, when their foraging grounds on the inter-tidal mudflats are inaccessible (van Gils et al., 2006). A spatio-temporal filter that retains positions inside the bounds of roosts during high-tide can quickly return data that reveals individual presence at high-tide roosts, while excluding positions that erroneously place individuals on the inundated mudflats. Users should apply filters in sequence rather than all at once, and visualise the output after each filtering step ('sanity checks').

The atlastools function `atl_filter_covariates` allows convenient filtering of a dataset by any number of logical statements (Listing 2). This function can be used to easily filter timestamps in a range, as well as combine simple spatial and temporal filters. It accepts a character vector of `R` expressions that each return a logical vector (i.e., `TRUE` or `FALSE`; Listing 2). The function returns only those data which satisfy each of the filter conditions. Users must ensure that the filtering variables exist in their dataset in order to avoid errors.

4 Filtering to Reduce Location Errors

4.1 Filtering on Quality Covariates

Tracking data covariates can be good indicators of the reliability of calculated positions (Beardsworth et al. *in prep.*). For intance, in ATLAS systems the number of base stations involved in each localisation is an indirect indicator of data quality,

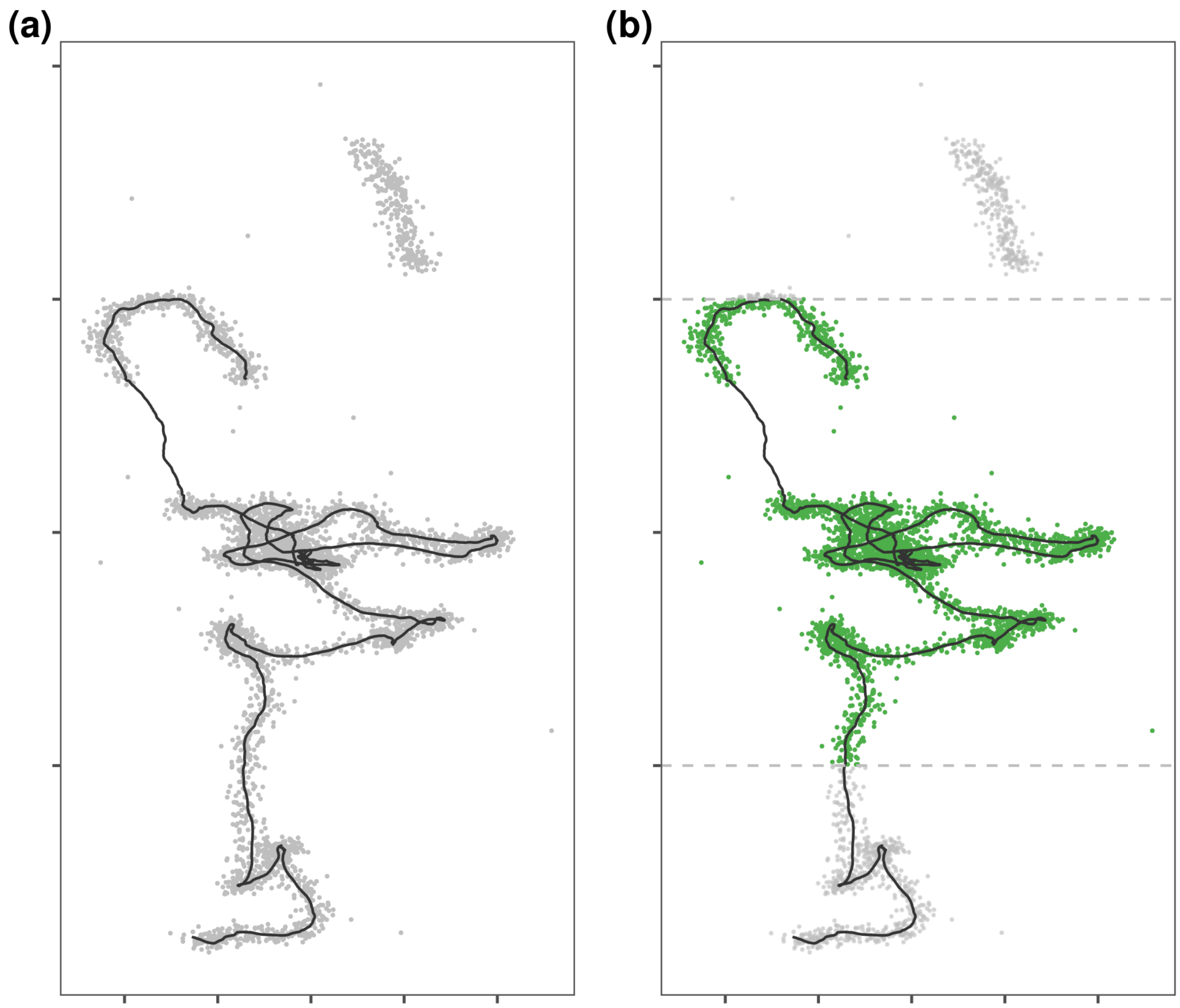

Figure 2. (a) A velocity-autocorrelated movement track simulated for 5,000 positions (black line) using the `smoove` package (Gurarie et al. 2017). Three kinds of errors have been artificially added: (1) each position (grey points) is offset from the canonical track with the addition of normally distributed small-scale error, (2) large-scale error has been added to 0.5% of positions, and (3) 300 positions (indices 500 – 800) have been displaced to the top-right of the track to simulate a gross distortion that affects a continuous subset of the track. The goal of pre-processing such datasets is to get the estimated positions (grey points) to match the canonical track (black line) as closely as possible. **(b)** Tracks can be quickly filtered by spatial bounds ($0.5 \leq Y \leq 1.0$; dashed grey lines) using the `atlastools` function `atl_filter_bounds`. Setting the function argument `remove_inside = FALSE` retains positions within user supplied bounds (dashed grey lines; green points), and excludes those outside (grey points).

and positions localised using more receivers are usually more reliable (the minimum required for an ATLAS localisation is 3; see Weiser et al., 2016). GPS systems' location error may be similarly indicated indirectly by the number of satellites involved in the localisation, or directly by an error measure such as the Horizontal Dilution of Precision (HDOP). ATLAS and other TOA systems also calculate direct measures of location error during localisation: `VARX`, `VARY`, and `COVXY`, which are the variance of the X and Y coordinates, and the covariance of the X and Y coordinates, respectively (MacCurdy et al.,

```r
night_data <- atl_filter_covariates(data = dataset,
                  filters = c("!inrange(hour, 6, 18)"))

data_in_area <- atl_filter_covariates(data = dataset,
                    filters = c("between(time, t_min, t_max)",
                                "between(x, x_min, x_max)"))
```

Listing 2. The atl_filter_covariates function can be used both as a simple temporal filter, and also as a combined spatio-temporal filter. Filter predicates are passed to the `filters` argument as a character vector, each of which is then evaluated as an `R` expression in the context of the data supplied. Only rows in the data satisfying all the conditions passed as filters are retained. Users must make sure the filter variables exist in their dataset. Here, the first example shows how nighttime data can be retained using a predicate (`inrange` from `data.table`) that determine whether the value of 'hour' is between 6 and 18. The `!` sign indicates that the negative of the predicate should be returned, i.e., `TRUE` when the hour is between 6 PM and 6 AM, but `FALSE` when the hour is between 6 AM and 6 PM. The second example shows the use of multiple filter statements; this data will be filtered to be within the range of times bounded by `t_min` and `t_max`, and with X coordinates between `x_min` and `x_max`. The `between` function is from `data.table`.

```r
filtered_data <- atl_filter_covariates(data = data,
                    filters = c("NBS > 3",
                                "SD < 100",
                                "between(day, 5, 8)"))
```

Listing 3. Filtering ATLAS data on position covariates. The `filters` argument accepts a character vector with the logical statements. The function only retains data for which *all* the conditions are satisfied; here that is positions calculated using $>$ 3 base stations (`NBS`), with location error (`SD`) $<$ 100, and data between an arbitrary day 5 and day 8.

2009, 2019; Weiser et al., 2016). A location error measure associated with each coordinate pair (similar to GPS HDOP) can be calculated and assigned to a new column `SD` using the formula for the sum of correlated random variables

$$SD = \sqrt{VARX + VARY + 2 \times COVXY}$$

Filtering on the position-specific standard deviation allows removing unreliable positions, and the filter can be applied using `atl_filter_covariates`.

4.2 Filtering Unrealistic Movement

Filtering on system-generated measures of error may not result in the removal of all erroneous positions, and data may remain which would require biologically implausible movement. Users are encouraged to visualise their tracks before and after filtering point locations, and especially to 'join the dots' and connect consecutive positions with lines. Whether the resulting track looks realistic is ultimately a subjective human judgement, but one which implicitly integrates prior knowledge of the movement ecology of the study species to ask, 'Does the animal move this way?'. Segments which appear to represent unrealistic animal movement are often obvious to researchers with extensive experience of the study system (the non-movement approach; see Bjørneraas et al., 2010). Since it is both difficult and prohibitively time consuming to exactly reproduce expert judgement when dealing with large volumes of tracking data from multiple individuals, some automation is necessary. Users should first manually examine a representative subset of tracks and attempt to visually identify problems — either with individual positions, or with subsets of the track — that persist after basic filtering. Once such problems are identified, users can conceptualise algorithms that can be applied to their data to resolve them.

An example of a problem with individual positions is that of point outliers or 'spikes' (Bjørneraas et al., 2010), where a single position is displaced far from the track (see Fig. 3.a). Point outliers are characterised by artificially high speeds between the outlier and the positions before and after (called incoming and outgoing speed, respectively Bjørneraas et al., 2010), lending a 'spiky' appearance to the track. Removing spikes is simple: remove positions with extreme incoming and outgoing speeds. Users must first define plausible upper limits for speed and turning angle for the study species (Calenge et al., 2009; Seidel et al., 2018). Here, it is important to remember that speed estimates are scale-dependent; high-throughput tracking typically overestimates the speed between positions where the animal is stationary or moving slowly due to small-scale location errors (Noonan et al., 2019; Ranacher et al., 2016). Even after data with large location errors have been removed by filters, it is advisable to begin with a liberal (high) speed threshold that excludes only the most unlikely of speeds. Estimates of maximum speed may not always be readily obtained for all species, and an alternative is to use a data-driven threshold such as the 95th percentile of speeds from the track. Once a speed threshold S has been chosen, positions with incoming *and* outgoing speeds $\geq S$ may be identified as spikes and removed. Some species can realistically achieve speeds $\geq S$ in fast transit segments when assisted by environmental factors, such as birds with tailwinds, and a simple filter on incoming and outgoing speeds would exclude this valid data. To avoid removing real, fast transit segments while still excluding spikes we suggest combining the speed filter with a filter on the turning angles of each position (see Calenge et al., 2009). This combined filter assumes that positions in high-throughput tracking with both high speeds and large turning angles are likely to be due to location errors, since most species are unable to turn sharply at high speed. Users can then retain only those positions whose incoming and outgoing speeds are both $< S$ which or satisfy the condition $\theta < A$, where θ is the turning angle, and A is the turning angle threshold. The removal of spikes is implemented using the `atl_filter_covariates` function (Listing 4). Many other track metrics may be used to identify implausible movement and on which data may be filtered (Seidel et al., 2018).

Sometimes entire susbets of the track may be affected by the same large-scale location error. For instance, multiple consecutive positions may be roughly translated (geometrically) away from the real track and form 'prolonged spikes', or 'reflections' (see Fig. 3.a, b). These cannot be corrected by targeted removal of individual positions, as in Bjorneraas and colleagues' approach (2010), since there are no positions with both high incoming and outgoing speeds. Since filtering individual positions will not suffice, algorithms to correct such errors must take a track-level view, and target the displaced sequence overall. Track-subset algorithms are likely to be system-specific, and may be challenging to conceptualise or implement. In the case of prolonged spikes, one relatively simple solution is identifying the bounds and removing positions between them. We show an illustrative example of such an algorithm in the form of track-subset filtering for prolonged spikes using the `atlastools` function `atl_remove_reflections` (Listing 5). Users are strongly encouraged to visualise

```
data$speed_in <- atl_get_speed(data,
                x = "x", y = "y",
                time = "time", type = c("in"))

data$angle <- atl_turning_angle(data,
                  x = "x", y = "y", time = "time")

filtered_data <- atl_filter_covariates(data = data,
        filters = c("(speed_in < S & speed_out < S) | angle < A"))
```

Listing 4. Filtering a movement track on incoming and outgoing speeds, and on turning angle to remove unrealistic movement. The functions `atl_get_speed` and `atl_turning_angle` are used to get the speeds and turning angles before filtering, and assigned to a column in the data (assignment of `speed_out` is not shown). The filter step only retains positions with speeds below the speed threshold S *or* angles above the turning angle threshold θ, i.e., positions where the animal is slow but makes sharp turns, and data where the animal moves quickly in a relatively straight line.

```
filtered_data <- atl_remove_reflections(data = track_data,
                       x = "x", y = "y", time = "time",
                       point_angle_cutoff = A,
                       reflection_speed_cutoff = S,
                       est_ref_len = N)
```

Listing 5. Removing prolonged spikes from a movement track. The important function arguments here are `point_angle_cutoff` (A), `reflection_speed_cutoff` (S), and `est_ref_len`, the maximum number of positions after the inner bound that are candidates for the end of the prolonged spike, i.e., the outer bound. If the prolonged spike ends after less than N positions, the true end point is used as the outer bound of the spike. However, the algorithm behind this function fails when the prolonged spike ends after more than N positions. Users are advised to use a liberally large value of N in the `est_ref_len` argument; 1,000 may be appropriate for 3s interval data. Further, users are cautioned against relying on such algorithms for severely distorted data.

their data before and after applying this method, and we caution against relying on this method if data are heavily distorted by errors affecting entire track-subsets.

5 Smoothing and Thinning Data

5.1 Median Smoothing

After filtering out large location errors may, the track may still look 'spiky' at small scales, and this is due to smaller location errors. These smaller errors are challenging to remove since their covariates (such as speed and turning angles) are within the expected range of movement behaviour for the study species. The large data volumes of high-throughput tracking allow users to resolve this problem by smoothing the positions. A 'smooth' works by approximating the value of an observation based on neighbouring values. For a one-dimensional series of observations, the neighbouring values are the K observations centred on each index value i. The range $i - (K-1)/2 \ldots i + (K-1)/2$ is referred to as the moving window as it shifts with i, and K is called the moving window size. A common smooth is nearest neighbour averaging, in which the value of an observation x_i is the average of the moving window K. The median smooth is a variant of nearest

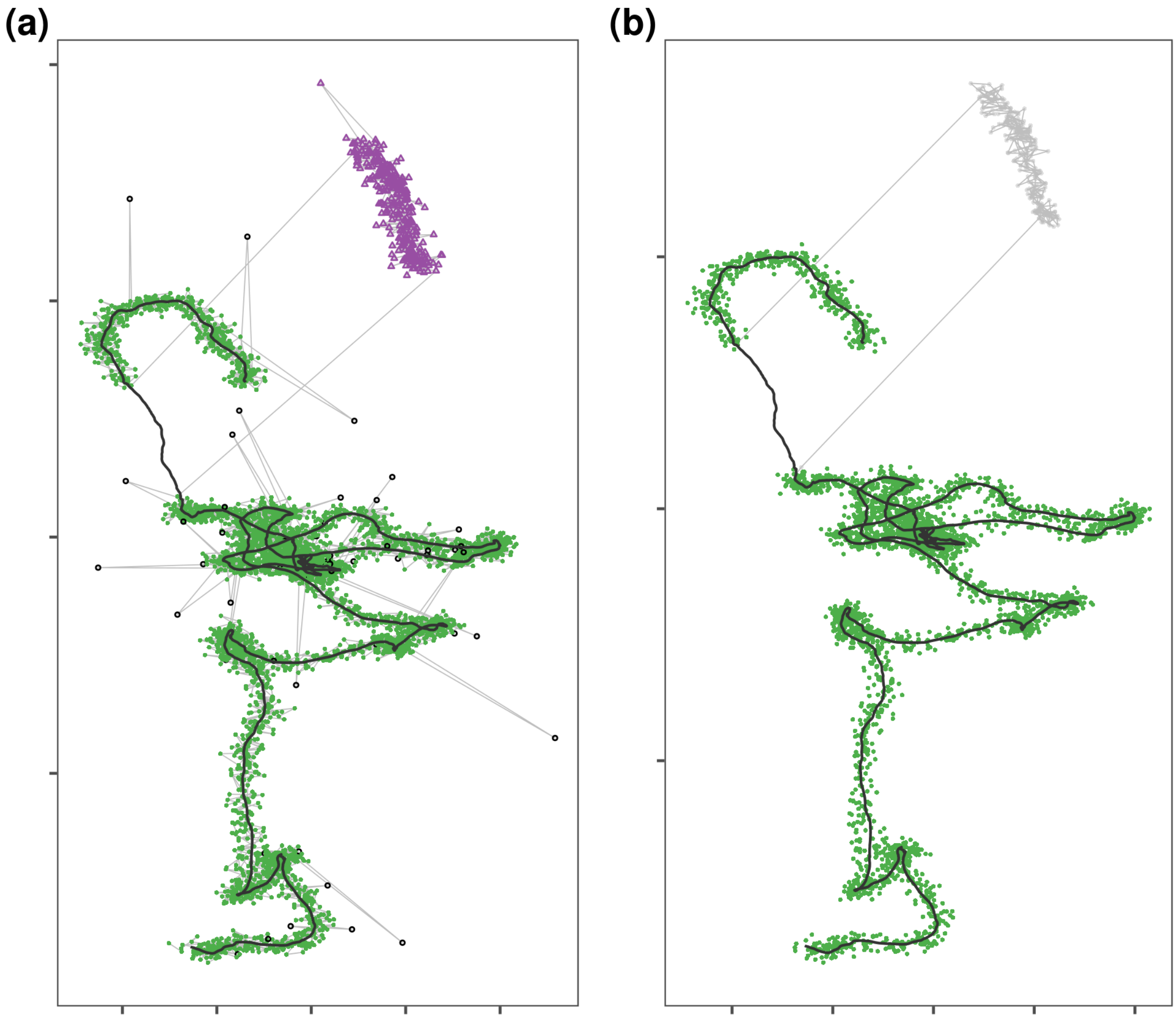

Figure 3. Reducing large-scale outliers in a movement track. The goal is to remove positions far away from the canonical track (black line), and retain those with only reasonable displacements from the canonical track (green points). **(a)** location error may affect single observations resulting in point outliers or 'spikes' (grey circles), but it may also affect continuous subsets of a track, called a 'prolonged spike' (purple triangles). The former may be targeted by filtering on appropriate covariates such as speed and turning angle using the `atlastools` function `atl_filter_covariates`. Here, we filter out positions with incoming and outgoing speeds $\geq$ the 90$^{\text{th}}$ percentile (0.025) and turning angle $\geq$ the 10$^{\text{th}}$ percentile (35°). Howevever, prolonged spikes cannot be effectively corrected by targeting each coordinate pair in isolation. **(b)** The `atlastools` function `atl_remove_reflections` to remove prolonged spikes (grey circles) in tracking data is an example of targeting track subsets with a common distortion (here, a geometric translation). While this method filters out the prolonged spike to return only positions with reasonable displacement from the canonical track (green points) in this example, conceptualising and implementing such algorithms is difficult. Users are cautioned to frequently check this and similar methods' results.

neighbour averaging which uses the median rather than the mean, and it is robust to outliers (Tukey 1977). The median 241

```r
atl_median_smooth(data = track_data,
                  x = "x", y = "y",
                  time = "time",
                  moving_window = 5)
```

Listing 6. Median smoothing a movement track using the function `atl_median_smooth` function with a moving window $K = 5$. Larger values of K yield smoother tracks, but K should always be some orders of magnitude lower than the number of observations.

smoothed value of the X coordinate, for instance, is

$$X_i = Median(X_{i-(K-1)/2} \dots X_{i+(K-1)/2})$$

Users can apply a median smooth with an appropriate K independently to the X and Y coordinates of a movement track to smooth it (see Fig. 4.a – e). The median smooth is robust to even very large temporal and spatial gaps, and does not interpolate between positions when data are missing. Thus it is not necessary to split the data into segments separated by periods of missing observations when applying the filter (see Fig. 4).

Smoothing does not change the number of observations, but does decouple the coordinates from some of their covariates. For instance, smoothing breaks the relationship between a coordinate and the location error estimate around it (`VARX`, `VARY`, and `SD` in ATLAS systems, or HDOP in GPS tracking). This makes subsequent filtering on covariates of data quality unreliable, and smoothed data are unsuitable for use with methods that model location uncertainty (Calabrese et al., 2016; Fleming et al., 2014, 2020; Noonan et al., 2019). Furthermore, while larger K may result in smoother tracks (Fig. 4.b – e), one drawback of using a large K is that short, quick forays away from the main track are likely to be smoothed away, leading to a loss in detail of the individual's small-scale movement. Users must themselves judge how best to trade large-scale and small-scale accuracy, and choose K accordingly. One empirical way to compare K is by calculating the root mean squared error (RMSE) for different K on the same data. Median smoothing is provided by the `atlastools` function `atl_median_smooth`, with the only option being the moving window size, which must be an odd integer (Listing 6).

5.2 Thinning Movement Tracks

Most data at this stage is technically 'clean', yet its volume alone may pose challenges for lower-specification or older hardware and software if these are not optimised for efficient computation. Thinning data need not compromise researchers' ability to answer scientific questions with them; for instance, social interactions lasting 1 – 2 minutes would still be detected on thinning from a sampling interval of 1 second to 1 minute. Indeed, temporal auto-correlation may hinder some methods such as the estimation of home-ranges or step-selection functions (Dupke et al., 2017; Fleming et al., 2014). Added to the requirement of uniform sampling intervals by many methods in the field, evenly reducing data volumes is a worthwhile endeavour (e.g. Avgar et al., 2016; Fleming et al., 2014; Michelot et al., 2016). Two plausible approaches here are resampling and aggregation, and both approaches begin with identifying time-interval groups (e.g. of 1 minute). Resampling picks

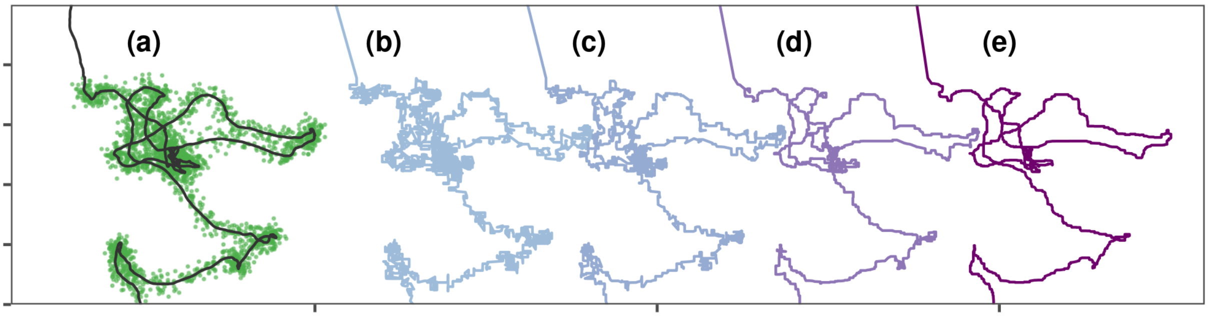

Figure 4. Median smoothing position coordinates reduces small-scale location error in tracking data. **(a)** The goal of this step is to approximate the simulated canonical track (black line), given positions with small-scale error remaining from previous steps (green points). The canonical track is shown 'zoomed in' for better representation, and the missing data at the top of the track (line with no points) is from where the prolonged spike was earlier removed. Median smoothing the given position coordinates ((a) green points) over a moving window (K) of **(b)** 3, **(c)** 5, **(d)** 11, and **(e)** 21 positions respectively, yields approximations (purple lines) of the canonical track. Users are cautioned that there is no correct K, and they must subjectively choose a K which most usefully trades small-scale details of the track for large-scale accuracy.

one position from each time-interval group. Aggregation involves computing the mean values of all covariates for positions within a time-interval group. Both approaches yield one position per time-interval group. Categorical variables, such as the habitat type associated with each position can be aggregated using a suitable measure such as the mode.

The aggregation method is less sensitive to selecting point outliers by chance than resampling. When users want to account for location error with methods such as state-space models (Johnson et al., 2008; Jonsen et al., 2005, 2003), or continuous time movement models (Calabrese et al., 2016; Fleming et al., 2014, 2020; Gurarie et al., 2017; Noonan et al., 2019), correctly propagating the location error when thinning is important. In ATLAS systems the location error (`SD`; see Filtering on Position Covariates) is calculated from the variance-covariance matrix of the coordinates of candidate positions considered by the location solver (Weiser et al., 2016); this is equivalent to GPS systems' HDOP (Ranacher et al., 2016). The error around each coordinate (`VARX` or `VARY` in ATLAS systems) can be propagated to the averaged position as the sum of errors divided by the square of the number of observations contributing to each average (N):

$$Var(X)_{agg} = \left(\sum_{i=1}^{i=N} Var(X)_i \right) / N^2$$

Similarly, the overall location error estimate for the average of N positions in a time-interval can be calculated by treating it as a variance (SD^2):

$$SD_{agg} = \sqrt{\left(\sum_{i=1}^{i=N} SD_i^2 \right) / N^2}$$

Users may question why thinning should be implemented after cleaning steps, when aggregation can obtain consensus positions over an interval, correctly propagate the location error, and also reduce data volumes. We caution users that thinning causes an extensive loss of small-scale detail in the data. Additionally, thinning prior to excluding unrealistic

```r
thinned_data <- atl_thin_data(data,
                              interval = 60,
                              id_columns = c("animal_id"),
                              method = "aggregate")
```

Listing 7. Code to thin data by aggregation in `atlastools`. The method can be either "aggregate" or "resample". The time interval is specified in seconds, while the `id_columns` allows a character vector of column names to be passed to the function, with these columns used as identity variables. Both methods return a dataset with as many rows as there are time-intervals. While the resampling method retains all columns, the aggregation method drops the ATLAS specific columns specifying covariance in X and Y coordinates (COVXY), and location error (SD).

movement and smoothing can lead to estimates of essential metrics — such as speed — that are substantially different from the true value (Noonan et al. 2019; see Fig. 5.c). The mis-estimation of track metrics could have knock-on consequences for the implementation of subsequent filters based on detecting unrealistic movement. However, thinning before data-cleaning may have its place as a useful step before exploratory visualisation of the movement track, since reduced data volumes are easier to handle for plotting software.

Thinning is implemented in `atlastools` using the `atl_thin_data` function, with either aggregation or resampling (specified by the `method` argument) over an interval using the `interval` argument. The column of timestamps must be named 'time' and column classes except the identity columns (*see below*) should allow averaging. The 'resample' option returns a thinned dataset with all columns from the input data, but 'aggregate' drops `COVXY`, as this cannot be propagated. Using 'resample' returns the actual timestamp (in UNIX time) of each sample, while 'aggregate' returns the mean timestamp (also in UNIX time). In both cases, an extra column `time_agg` is added which has a uniform difference between each element corresponding to the user-defined thinning interval. Due to its development for ATLAS systems, the 'aggregate' assumes the variance around coordinates is named `VARX` and `VARY`, and standard deviation around each position is named `SD`. These columns must be present together for the function to correctly handle the error. If there is no measure of error, the function simply returns the averaged position and covariates in each time interval. Grouping variables' names (such as animal identity) may be passed as a character vector to the `id_columns` argument (Listing 7).

6 Synthesising Movement Tracks into Residence Patches

6.1 The Residence Patch Algorithm

Tracking data that are still too large for statistical packages, or have strong autocorrelation, can benefit from segmentation-clustering into 'residence patches' (Barraquand and Benhamou, 2008; Bijleveld et al., 2016; Oudman et al., 2018). Making the patch the unit of observation conveniently sidesteps pseudo-replication while and reduces computational requirements. Furthermore, metrics such as the distance travelled within and between patches can help compare broad-scale individual movement strategies.

First, users should identify positions representing bouts of stationary behaviour, for instance on their speed or residence

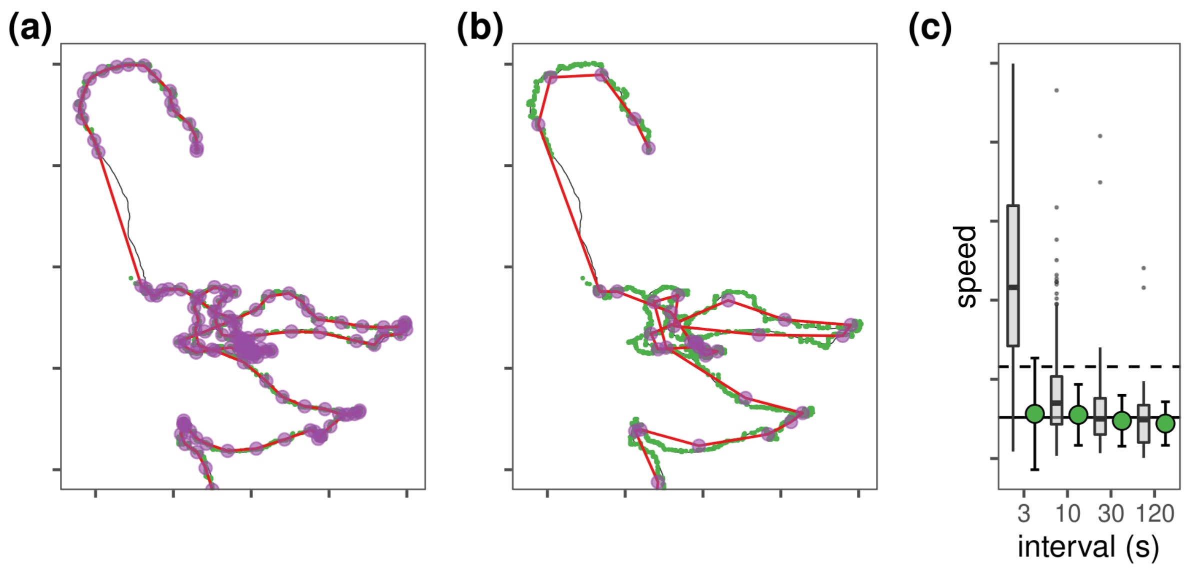

Figure 5. Thinning a movement track using aggregation preserves track structure while reducing data volume, but thinning without removing gross location errors and unrealistic movement affects track metrics such as speed. (a, b) Movement track positions with an interval of 1 second (green points) aggregated over intervals of (a) 10 and (b) 30 seconds (purple circles), after removing large location errors or filtering unrealistic movement. The canonical track is shown as a black line in both panels, while the red line shows the track connecting thinned positions. (c) Boxplots show the median and interquartile ranges for speed estimates of tracks aggregated over intervals of 3, 10, 30, and 120 seconds, but without the removal of location errors. For comparison, the median and 95$^{\text{th}}$ percentile of speed of the canonical track are shown as solid and dashed horizontal lines, respectively. Green circles and associated error bars show, in contrast, the median ($\pm$ standard deviation) speed for the same aggregation interval, but after removing large- and small-scale location errors (median smooth $K = 11$). Removing location errors and unrealistic movement before aggregation substantially improves speed estimates, especially at smaller aggregation intervals, as it removes the apparent spikes that contribute to over estimation of speed if not filtered.

time (Bracis et al., 2018). Patch identification assesses whether consecutive positions and the bouts they comprise are spatio-temporally independent, and clusters them together if they are not. The splitting and lumping of positions into bouts, and bouts into patches is based on simple user specified thresholds — the distance and the time interval between positions (and bouts) beyond which they should be considered independent. Users are encouraged to base these thresholds in the movement habits of their study species. For example, residence patch classification of red knot movement tracks considers consecutive stationary positions independent if they are 20m apart and considers consecutive bouts independent if the distance between them $\geq$ 100m, or the time difference between them $\geq$ 30 minutes (Listing 8). The 20m distance represents a maximum speed of 6.667 m/s between positions, above which the individual is more likely in transit. The 100m and 30 minute thresholds are chosen to account for potentially missing data between bouts; if a track ends abruptly and then reappears $\geq$ 100m away or $\geq$ 30 minutes later, this is more safely considered a new residence patch.

A cleaned movement track can be classified into residence patches using the function `atl_res_patch` (see Fig. 6.c). `atl_res_patch` requires three parameters: (1) the distance threshold between positions (called `buffer_size`), (2) the distance threshold between clusters of positions (called `lim_spat_indep`), and (3) the time interval between clusters (called `lim_time_indep`). Clusters formed of fewer than a minimum number of positions can be excluded. Our residence patch

```r
patches <- atl_res_patch(data = track_data,
                buffer_radius = 10,
                lim_spat_indep = 100,
                lim_time_indep = 30,
                min_fixes = 3,
                summary_variables = c("speed"),
                summary_functions = c("mean", "sd"))

patch_summary <- atl_patch_summary(patch_data = patches,
                    which_data = "summary",
                    buffer_radius = 10)
```

Listing 8. The `atl_res_patch` function can be used to classify a track into residence patches. The arguments `buffer_radius` and `lim_spat_indep` are specified in metres, while the `lim_time_indep` is provided in minutes. In this example, specifying `summary_variables = c("speed")`, and `summary_functions = c("mean", "sd")` will provide the mean and standard deviation of instantaneous speed in each residence patch. The `atl_patch_summary` function is used to access the classified patch in one of three ways, here using the `summary` option which returns a table of patch-wise summary statistics.

algorithm is capable of correctly identifying clusters of related residence points from a movement track (Fig. 7.a, 7.b). This includes clusters where the animal is relatively stationary (orange and green patches, Fig. 7.c), as well as clusters where the animal is moving slowly (blue patch, Fig. 7.c). This flexibility is especially useful when studying movements that may represent two different modes of the same behaviour, for instance, area-restricted search, as well as slow, searching behaviour with a directional component.

The function `atl_patch_summary` can be used to extract patch-specific summary data such as the median coordinates, the patch duration, the distance travelled within the patch, and the patch area. Position covariates such as speed may also be summarised patch-wise by passing covariate names and summary functions as character vectors to the `summary_variables` and `summary_functions` arguments, respectively. Setting the `which_data` argument to `"spatial"`, returns `sf MULTIPOLYGON` objects, and setting `which_data = "points"` returns the positions in each patch, with patch-specific covariates.

6.2 Validating the Residence Patch Method

We applied the pre-processing pipeline using `atlastools` functions described above to a calibration dataset to verify that the residence patch method could correctly identify known stopping points (see Fig. 7). We collected the calibration data (n = 50,816) by walking and boating with a hand-held WATLAS tag (sampling frequency = 1 Hz) around the island of Griend (53.25°N, 5.25°E) in August 2020 (Beardsworth et al. *in prep.*; Bijleveld et al. *in prep.*). Stops in the calibration track were recorded as waypoints using a handheld GPS device (Garmin Dakota 10) at each stop. We estimated the real duration of each stop as the time difference between the first and last position recorded within 50m of each waypoint, within a 10 minute window before and after the waypoint timestamp (to avoid biased durations from revisits). Stops had a median duration of 10.28 minutes (range: 1.75 minutes – 20 minutes; see Supplementary Material). We cleaned the data before constructing residence patches by (1) removing a single outlier ($>$ 15 km away), removing unrealistic movement ($\geq$ 15 m/s), smoothing the data (K = 5), and (4) thinning the data by resampling over a 30 second interval. The cleaning steps retained

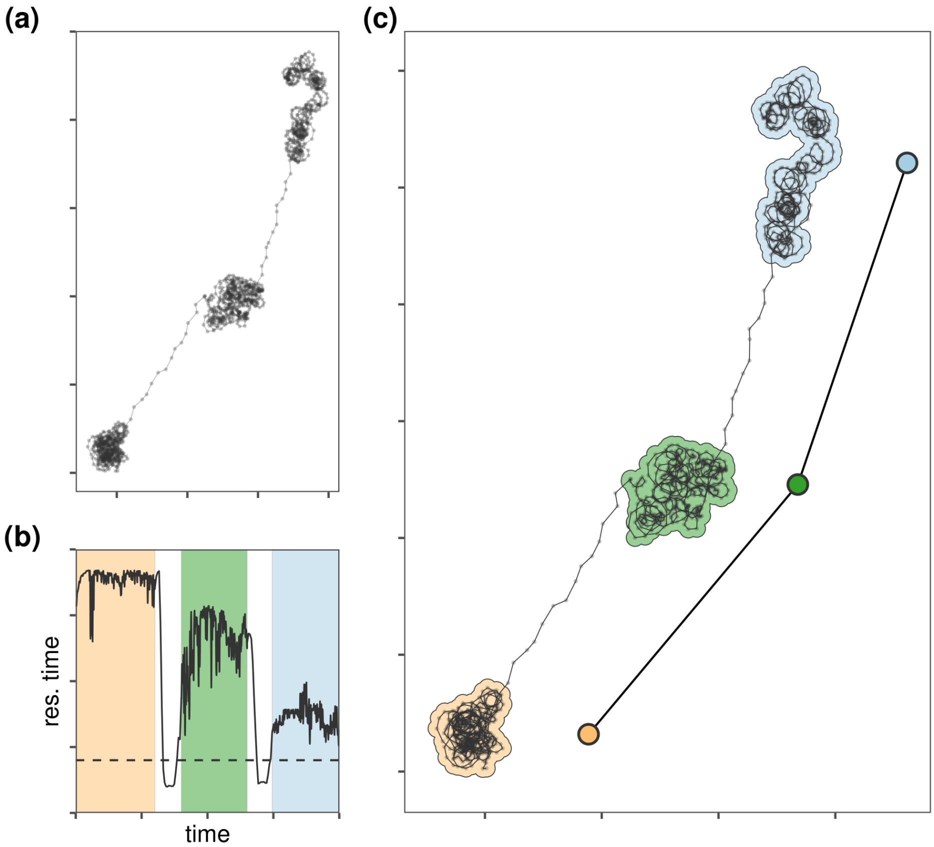

Figure 6. A rotation-advection movement track, similar to the movement of a soaring bird (Gurarie et al. 2017) classified into residence patches with the `atlastools` function. (a) The goal is to segment-cluster the track into areas of prolonged residence (clusters of points) while leaving out the transit between them. (b) A plot of residence time against time (solid line; Bracis et al. 2018) shows how the residence patch algorithm segments and clusters positions of prolonged residence. Regions are shaded by the temporal bounds of each residence patch. The arguments passed to `atl_res_patch` determine the clustering, and can be adjusted to get a result that fits the study system. Users are cautioned that there are no 'correct' arguments. (c) The residence patch method correctly identifies clusters of positions where the individual is relatively stationary (orange and green patches), as well as positions where it is slowly moving (blue patch). This is especially useful when studying more complex behaviour such as area-restricted search, which may have a directional component. The function `atl_patch_summary` conveniently presents the classified data for further use, either as an `sf` object (left: coloured polygons around clustered points), or as summary data of each patch (right: coloured circles). While the `sf` object retains the patch geometry as well as the patch attributes (such as duration), the summary data discards the patch geometry. Both spatial and summary objects lose the details of movement between patches (compare black lines of the canonical track [left], and lines connecting patch centroids [right]), but are helpul for questions where the general structure is more relevant.

37,324 positions (74.45%), while thinning reduced these to 1,803 positions (4.8% positions of the smoothed track). Details and code are provided in the Supplementary Material (see VALIDATING THE RESIDENCE PATCH METHOD WITH CALIBRATION DATA).

We identified stationary positions (residence time $\geq$ 5 minutes) using the `recurse` package (n = 837, 46.42 %; radius = 50m Bracis et al., 2018). We clustered these positions into residence patches with a buffer radius of 5m, spatial independence limit of 50m, temporal independence limit of 5 minutes, and a minimum of 3 positions per patch. Inferred residence patches corresponded well to the locations of stops (see Fig. 7.c). However, the residence patch algorithm detected more stops than were logged as waypoints (n = 28, n waypoints = 21). One of these was the field station on Griend where the tag was stored between trips (red crossed-square, Fig. 7.c). The method also did not detect two stops of 105 and 563 seconds (1.75 and 9.4 minutes) since they were data poor and aggregated away in the thinning step (n positions = 6, 15). To determine whether the residence patch method correctly identified the duration of stops in the calibration track, we first extracted the patch attributes using the function `atl_patch_summary`. We then matched the patches to the waypoints by their median coordinates (rounded to 100 metres). We assigned the inferred duration of the stop as the duration of the spatially matched residence patch. We compared the inferred duration with the real duration using a linear model with the inferred duration as the only predictor of the real duration. Inferred duration was a good predictor of the real duration of a stop (linear model estimate = 1.021, t-value = 12.965, $p < 0.0001$, R^2 = 0.908; see Supplementary Material Fig. 1.7). This translates to a 2% underestimation of the stop duration at a tracking interval of 30 seconds.

7 Worked-Out Example on Animal Tracking Data

We present a fully worked-out example of our pre-processing pipeline and residence patch method using movement data from three Egyptian fruit bats tracked using the ATLAS system (*Rousettus aegyptiacus*; Toledo et al. (2020)). Code can be found in the Supplementary Material (see PROCESSING EGYPTIAN FRUIT BAT TRACKS). Bats were tracked over three nights (5th, 6th, and 7th May, 2018) in the Hula Valley, Israel (33.1°N, 35.6°E), with an average of 13,370 positions (SD = 2,173; range = 11,195 – 15,542; interval = 8 seconds) per individual. Plotting the tracks showed severe distortions (see Supplementary Material Fig. 2.1). We first reduced location errors by removing observations with ATLAS SD > 20, and observations calculated using fewer than four base stations (mean positions remaining = 10,447 / individual; 78% of the raw data on average). We removed unrealistic movement represented by positions with incoming and outgoing speeds > 20 m/s leaving 10,337 positions per individual on average (98% of previous step). We median smoothed the data with a moving window K size = 5, and no observations were lost.

We began the construction of residence patches by finding the residence time within 50 metres of each position (Bracis et al., 2018). Bats may repeatedly traverse the same routes, and this could artificially inflate the residence time of positions along these routes. To avoid confusing revisits with residence, we limited the summation of residence times at each position to the period until the first departure of 60 minutes or more. Thus, two nearby locations ($\leq$ 50m apart) each visited for one minute at a time, but separated by an interval of some hours would not have a residence time of two minutes each, but only one minute each. Bats had a mean residence time at locations of 100.54 minutes (SD = 114.7); this measure was strongly biased by time spent at the roost. We opted for a first-principles approach and selected as residence positions any locations with a residence time > 5 minutes, reasoning that a flying animal stopping for > 5 minutes at a location should plausibly indicate resource use or another interesting behaviour. This step retained 7,819 positions per bat on average (75.6%) of the smoothed data; suggesting the extent to which roosting biases the data.

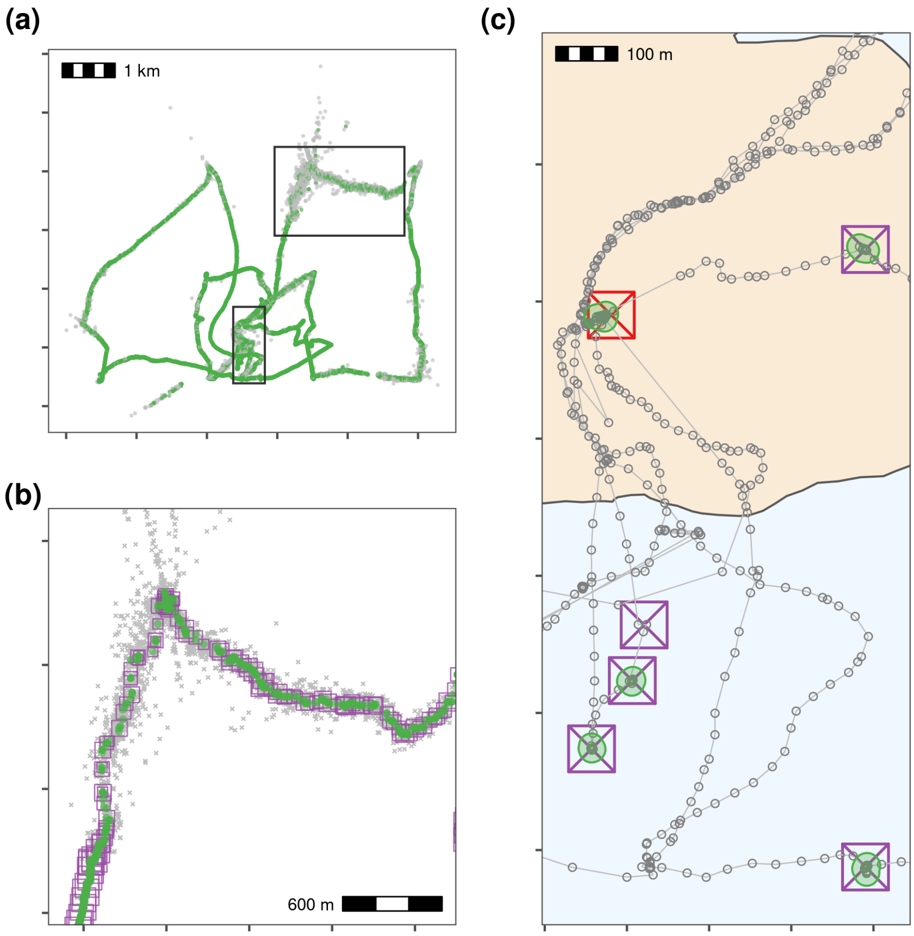

Figure 7. Pre-processing steps for WATLAS calibration data showing filtering filtering on speed, median smoothing and thinning by aggregation, and making residence patches. **(a)** Positions with incoming and outgoing speed $\geq$ the 15 m/s are removed (N removed = 13,491, 26.6%; grey crosses = removed, green points = retained). Rectangles show the region expanded in subfigures (b) and (c). **(b)** Expanded view of upper rectangle in (a) showing the raw data (grey crosses), the median smoothed positions (green circles; moving window K = 5), and the smoothed track thinned by aggregation to a 30 second interval (purple squares). Square size corresponds to the number of positions used to calculate the averaged position during thinning, with no significant differences. Median smoothing retains all 37,324 observations (though changing their coordinates). Thinning aggregates these positions into 1,803 aggregates (4.8% of the smoothed data volume). Median smoothing recovers a good estimate of the true track from significantly over-dispersed raw data, while thinning reduces the data volume but preserves track structure. **(c)** Classifying thinned data into residence patches yields robust estimates of the duration of known stops. The island of Griend (53.25°N, 5.25°E) is shown in beige. Residence patches (green polygons; function parameters in text) correspond well to the locations of known stops (purple crossed-squares). However, the algorithm identified all areas with prolonged residence, including those which were not intended stops, such as stops at the field station (n = 12; green polygon over red crossed-squares). The algorithm also failed to find two stops of 6 and 15 seconds duration, since these were lost in the data thinning step (crossed-square without green polygon shows one of these).

We constructed residence patches with a buffer distance of 25m, a spatial independence limit of 100m, a temporal independence limit of 30 minutes, and rejected patches with fewer than three positions. We extracted summary data and spatial polygons from the constructed residence patches. Plotting the bats' residence patches and the linear paths between them showed that though all three bats roosted at the same site, they used distinct areas of the study site (Fig. 9.a). Bats tended to show prolonged residence near known food sources (fruit trees), travelling repeatedly between previously visited areas (Fig. 9.b). However, bats also appeared to spend some time at locations where no fruit trees were recorded, prompting questions about their use of other food sources, or another behaviour entirely (Fig. 9.b, 9.c). Bats occurring close together did not have strongly overlapping residence patches, and their paths to and from area of co-occurrence were different (Fig. 9.a, 9.c). Constructing residence patches for multiple individuals over multiple nights suggests interesting dynamics of within- and between-individual overlap. 387 388 389 390 391

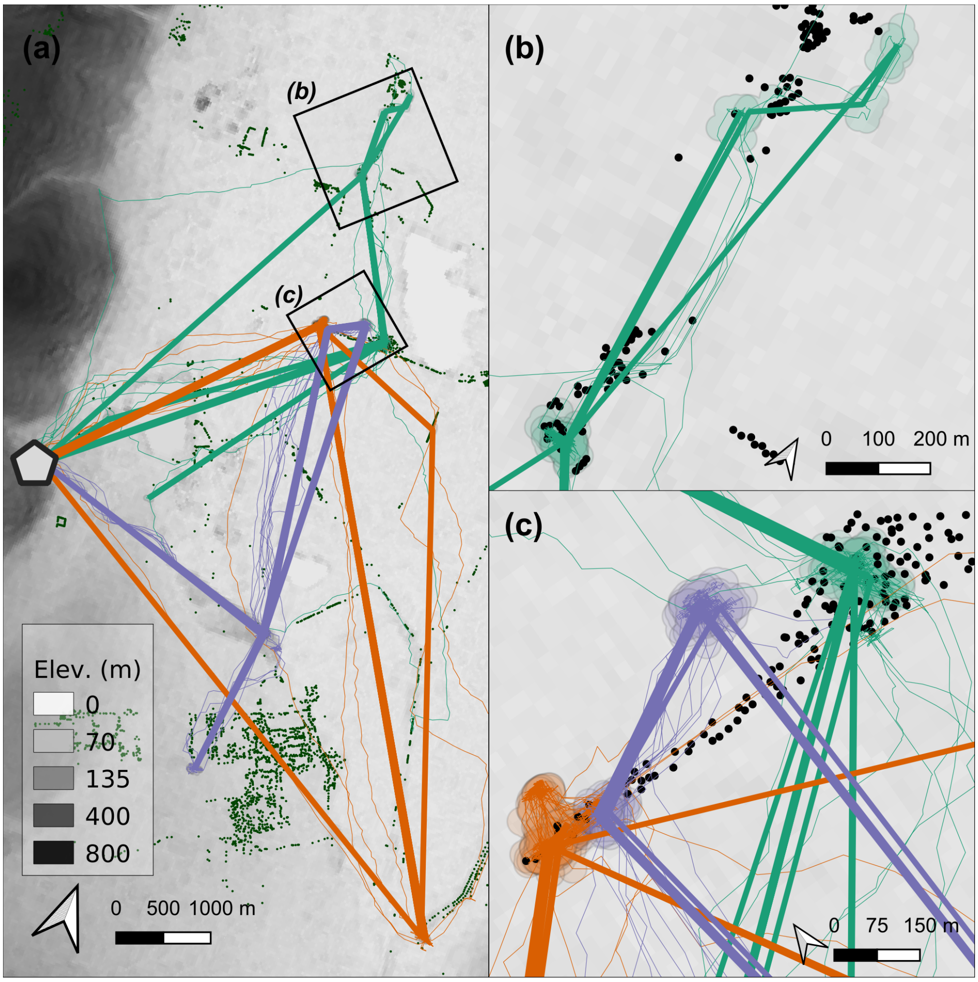

Figure 8. Synthesising animal tracks into residence patches can reveal how individuals in a population move in relation to landscape features, prior exploration, and other individuals. **(a)** Linear approximations of the paths (coloured lines) between residence patches (circles) show the movement of three Egyptian fruit bats (*Rousettus aegyptiacus*), tracked over three nights in the Hula Valley, Israel. Bat tracks are are shown as thin lines below the linear approximations. Colours show bat identity. The grey hexagon represents a roost site. Dark green points represent known fruit trees. Background is shaded by elevation at 30 metre resolution (SRTM). **(b)** Spatial representations of an individual bat's residence patches (green polygons) can be used to study site-fidelity by examining overlaps between patches, or to study resource selection by inspecting overlaps with known resources (here, black circles show the location of clusters of fruit trees). In addition, the linear approximation of movement between patches (blue paths in panel (a)) can be contrasted with the estimated real path between patches (green lines). **(c)** Residence patch polygons can also show how individuals partition space and resources (black circles are fruit trees), and the combined influence of resource and potential social cues on movement between patches (coloured paths). Patches and paths are coloured by bat identity.

8 Discussion

Our guide anticipates high-throughput animal tracking data becoming increasingly common. A uniform pipeline and toolset for data cleaning promotes reproducibility and standardisation across studies, making comparative inferences more robust. The open-source `R` package `atlastools` serves as a starting point for methodological collaboration among movement ecologists. Efficient location error modelling approaches (Fleming et al., 2020) may eventually make data-cleaning optional. Yet cleaning tracking data even partially before modelling location error will be faster than error-modelling on the full data, and the removal of large location errors may improve model fits. Thus we see our pipeline as complementary to these approaches (Fleming et al., 2014, 2020). Finally, we recognise that the diversity and complexity of animal movement and data collection techniques often requires bespoke pre-processing solutions. Though the principles outlined here are readily generalised, users' requirements will eventually exceed the particular tools we provide. We see this as an incentive for more users to be involved in developing methods for their systems. We offer our pipeline and package as a foundation for system specific tools in the belief that simple, robust concepts are key to methods development that balances system-specificity and broad applicability.

9 Backmatter

9.1 Competing Interests

The authors declare that they have no competing interests.

9.2 Acknowledgements

PRG would like to thank Pedro M. Santos Neves for introducing PRG to `R` package development, for help with setting up `atlastools`, and for help with archiving it on Zenodo; A. Ramesh and F.J. Weissing for discussions on error propagation; J.R.L. Gismann and E. Gobbens for feedback that improved the manuscript; members of the Modelling Adaptive Response Mechanisms Group (Weissing Lab), and the Theoretical Biology department at the University of Groningen for helpful discussions on `atlastools` and the manuscript. All authors thank the attendees of ATLAS workshops held in May and June 2020 at the Hebrew University of Jerusalem for helpful comments on the pipeline and `atlastools`.

9.3 Authors' Contributions

PRG wrote the manuscript and inline code snippets, performed the analyses, prepared the figures, and developed the `R` package `atlastools`. CEB and AIB collected the calibration track, and EL collected the bat movement data and fruit tree locations. RN conceived the idea of writing this manuscript, and PRG, AIB, OS, CEB, ST, and EL contributed to its design, and the design of `atlastools`. All authors contributed to the writing of the manuscript, and the design of figures.

9.4 Data Availability

The data and source code to reproduce the figures and analyses in this article and in the Supplementary Material can be found in the Zenodo repository at https://doi.org/10.5281/zenodo.4287462.

9.5 Supplementary Material

1. Supplementary Material 1: Code for worked out examples on calibration data from the Dutch Wadden Sea, and bat tracking data from the Hula Valley, Israel.

2. Supplementary Material 2: Manual for the `R` package `atlastools`.

References

Aarts, G., M. MacKenzie, B. McConnell, M. Fedak, and J. Matthiopoulos. 2008. Estimating space-use and habitat preference from wildlife telemetry data. Ecography 31:140–160.

Avgar, T., J. R. Potts, M. A. Lewis, and M. S. Boyce. 2016. Integrated step selection analysis: Bridging the gap between resource selection and animal movement. Methods in Ecology and Evolution 7:619–630.

Baktoft, H., K. Ø. Gjelland, F. Økland, J. S. Rehage, J. R. Rodemann, R. S. Corujo, N. Viadero, and U. H. Thygesen. 2019. Opening the black box of high resolution fish tracking using yaps. bioRxiv page 2019.12.16.877688.

Baktoft, H., K. Ø. Gjelland, F. Økland, and U. H. Thygesen. 2017. Positioning of aquatic animals based on time-of-arrival and random walk models using YAPS (Yet Another Positioning Solver). Scientific Reports 7:14294.

Barnett, A. H., and P. R. Moorcroft. 2008. Analytic steady-state space use patterns and rapid computations in mechanistic home range analysis. Journal of Mathematical Biology 57:139–159.

Barraquand, F., and S. Benhamou. 2008. Animal movements in heterogeneous landscapes: Identifying profitable places and homogeneous movement bouts. Ecology 89:3336–3348.

Bijleveld, A. I., R. B. MacCurdy, Y.-C. Chan, E. Penning, R. M. Gabrielson, J. Cluderay, E. L. Spaulding, A. Dekinga, S. Holthuijsen, J. ten Horn, M. Brugge, J. A. van Gils, D. W. Winkler, and T. Piersma. 2016. Understanding spatial distributions: Negative density-dependence in prey causes predators to trade-off prey quantity with quality. Proceedings of the Royal Society B: Biological Sciences 283:20151557.

Bjørneraas, K., B. V. Moorter, C. M. Rolandsen, and I. Herfindal. 2010. Screening Global Positioning System Location Data for Errors Using Animal Movement Characteristics. The Journal of Wildlife Management 74:1361–1366.

Bracis, C., K. L. Bildstein, and T. Mueller. 2018. Revisitation analysis uncovers spatio-temporal patterns in animal movement data. Ecography 41:1801–1811.

Calabrese, J. M., C. H. Fleming, and E. Gurarie. 2016. Ctmm: An r package for analyzing animal relocation data as a continuous-time stochastic process. Methods in Ecology and Evolution 7:1124–1132.

Calenge, C., S. Dray, and M. Royer-Carenzi. 2009. The concept of animals' trajectories from a data analysis perspective. Ecological Informatics 4:34–41.

D'Eon, R. G., R. Serrouya, G. Smith, and C. O. Kochanny. 2002. GPS Radiotelemetry Error and Bias in Mountainous Terrain. Wildlife Society Bulletin (1973-2006) 30:430–439.

Dowle, M., and A. Srinivasan. 2020. Data.Table: Extension of 'data.Frame'.

Dupke, C., C. Bonenfant, B. Reineking, R. Hable, T. Zeppenfeld, M. Ewald, and M. Heurich. 2017. Habitat selection by a large herbivore at multiple spatial and temporal scales is primarily governed by food resources. Ecography 40:1014–1027.

Fischler, M. A., and R. C. Bolles. 1981. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. Communications of the ACM 24:381–395.

Fleming, C. H., J. M. Calabrese, T. Mueller, K. A. Olson, P. Leimgruber, and W. F. Fagan. 2014. From Fine-Scale Foraging to Home Ranges: A Semivariance Approach to Identifying Movement Modes across Spatiotemporal Scales. The American Naturalist 183:E154–E167.

Fleming, C. H., J. Drescher-Lehman, M. J. Noonan, T. S. B. Akre, D. J. Brown, M. M. Cochrane, N. Dejid, V. DeNicola, C. S. DePerno, J. N. Dunlop, N. P. Gould, J. Hollins, H. Ishii, Y. Kaneko, R. Kays, S. S. Killen, B. Koeck, S. A. Lambertucci, S. D. LaPoint, E. P. Medici, B.-U. Meyburg, T. A. Miller, R. A. Moen, T. Mueller, T. Pfeiffer, K. N. Pike, A. Roulin, K. Safi, R. Séchaud, A. K. Scharf, J. M. Shephard, J. A. Stabach, K. Stein, C. M. Tonra, K. Yamazaki, W. F. Fagan, and J. M. Calabrese. 2020. A comprehensive framework for handling location error in animal tracking data*. bioRxiv page 2020.06.12.130195.

Frair, J. L., S. E. Nielsen, E. H. Merrill, S. R. Lele, M. S. Boyce, R. H. M. Munro, G. B. Stenhouse, and H. L. Beyer. 2004. Removing GPS collar bias in habitat selection studies. Journal of Applied Ecology 41:201–212.

Gupte, P. R. 2020. Atlastools: Pre-processing Tools for High Frequency Tracking Data. Zenodo.

Gurarie, E., C. H. Fleming, W. F. Fagan, K. L. Laidre, J. Hernández-Pliego, and O. Ovaskainen. 2017. Correlated velocity models as a fundamental unit of animal movement: Synthesis and applications. Movement Ecology 5:13.

Harel, R., N. Horvitz, and R. Nathan. 2016. Adult vultures outperform juveniles in challenging thermal soaring conditions. Scientific Reports 6:27865.

Holyoak, M., R. Casagrandi, R. Nathan, E. Revilla, and O. Spiegel. 2008. Trends and missing parts in the study of movement ecology. Proceedings of the National Academy of Sciences of the United States of America 105:19060–5.

Hurford, A. 2009. GPS Measurement Error Gives Rise to Spurious 180° Turning Angles and Strong Directional Biases in Animal Movement Data. PLOS ONE 4:e5632.

Hussey, N. E., S. T. Kessel, K. Aarestrup, S. J. Cooke, P. D. Cowley, A. T. Fisk, R. G. Harcourt, K. N. Holland, S. J. Iverson, J. F. Kocik, J. E. Mills Flemming, and F. G. Whoriskey. 2015. Aquatic animal telemetry: A panoramic window into the underwater world. Science 348:1255642–1255642.

Johnson, D. S., J. M. London, M.-A. Lea, and J. W. Durban. 2008. Continuous-Time Correlated Random Walk Model for Animal telemetry Data. Ecology 89:1208–1215.

Jonsen, I. D., J. M. Flemming, and R. A. Myers. 2005. Robust State–Space Modeling of Animal Movement Data. Ecology 86:2874–2880.

Jonsen, I. D., R. A. Myers, and J. M. Flemming. 2003. Meta-Analysis of Animal Movement Using State-Space Models. Ecology 84:3055–3063.

Joo, R., S. Picardi, M. E. Boone, T. A. Clay, S. C. Patrick, V. S. Romero-Romero, and M. Basille. 2020. A decade of movement ecology. arXiv:2006.00110 [q-bio] .

Jung, K. W., Z. D. Deng, J. J. Martinez, D. R. Geist, G. A. McMichael, J. R. Stephenson, and P. J. Graf. 2015. Performance of an acoustic telemetry system in a large fishway. Animal Biotelemetry 3:17.

Langrock, R., R. King, J. Matthiopoulos, L. Thomas, D. Fortin, and J. M. Morales. 2012. Flexible and practical modeling of animal telemetry data: Hidden Markov models and extensions. Ecology 93:2336–2342.

Lewis, J. S., J. L. Rachlow, E. O. Garton, and L. A. Vierling. 2007. Effects of habitat on GPS collar performance: Using data screening to reduce location error. Journal of Applied Ecology 44:663–671.

MacCurdy, R., R. Gabrielson, E. Spaulding, A. Purgue, K. Cortopassi, and K. Fristrup. 2009. Automatic Animal Tracking Using Matched Filters and Time Difference of Arrival. JCM 4:487–495.

MacCurdy, R. B., A. I. Bijleveld, R. M. Gabrielson, and K. A. Cortopassi. 2019. Automated Wildlife Radio Tracking. Chap. 33, pages 1219–1261 in Handbook of Position Location. John Wiley & Sons, Ltd.

Marwick, B., C. Boettiger, and L. Mullen. 2018. Packaging Data Analytical Work Reproducibly Using R (and Friends). The American Statistician 72:80–88.

Michelot, T., R. Langrock, and T. A. Patterson. 2016. moveHMM: An R package for the statistical modelling of animal movement data using hidden Markov models. Methods in Ecology and Evolution 7:1308–1315.

Nathan, R., W. M. Getz, E. Revilla, M. Holyoak, R. Kadmon, D. Saltz, and P. E. Smouse. 2008. A movement ecology paradigm for unifying organismal movement research. Proceedings of the National Academy of Sciences 105:19052–19059.

Noonan, M. J., C. H. Fleming, T. S. Akre, J. Drescher-Lehman, E. Gurarie, A.-L. Harrison, R. Kays, and J. M. Calabrese. 2019. Scale-insensitive estimation of speed and distance traveled from animal tracking data. Movement Ecology 7:35.

Oudman, T., T. Piersma, M. V. Ahmedou Salem, M. E. Feis, A. Dekinga, S. Holthuijsen, J. ten Horn, J. A. van Gils, and A. I. Bijleveld. 2018. Resource landscapes explain contrasting patterns of aggregation and site fidelity by red knots at two wintering sites. Movement Ecology 6:24–24.

Papageorgiou, D., C. Christensen, G. E. C. Gall, J. A. Klarevas-Irby, B. Nyaguthii, I. D. Couzin, and D. R. Farine. 2019. The multilevel society of a small-brained bird. Current Biology 29:R1120–R1121.

Patin, R., M.-P. Etienne, E. Lebarbier, S. Chamaillé-Jammes, and S. Benhamou. 2020. Identifying stationary phases in multivariate time series for highlighting behavioural modes and home range settlements. Journal of Animal Ecology 89:44–56.

Patterson, T. A., L. Thomas, C. Wilcox, O. Ovaskainen, and J. Matthiopoulos. 2008. State–space models of individual animal movement. Trends in Ecology & Evolution 23:87–94.

Pebesma, E. 2018. Simple Features for R: Standardized Support for Spatial Vector Data. The R Journal 10:439–446.

R Core Team. 2020. R: A Language and Environment for Statistical Computing. R Foundation for Statistical Computing, Vienna, Austria.

Ranacher, P., R. Brunauer, W. Trutschnig, S. V. der Spek, and S. Reich. 2016. Why GPS makes distances bigger than they are. International Journal of Geographical Information Science 30:316–333.

Seidel, D. P., E. Dougherty, C. Carlson, and W. M. Getz. 2018. Ecological metrics and methods for GPS movement data. International Journal of Geographical Information Science 32:2272–2293.

Signer, J., J. Fieberg, and T. Avgar. 2017. Estimating utilization distributions from fitted step-selection functions. Ecosphere 8:e01771.

———. 2019. Animal movement tools (amt): R package for managing tracking data and conducting habitat selection analyses. Ecology and Evolution 9:880–890.

Slingsby, A., and E. van Loon. 2016. Exploratory Visual Analysis for Animal Movement Ecology. Computer Graphics Forum 35:471–480.

Stine, P. A., and C. T. Hunsaker. 2001. An Introduction to Uncertainty Issues for Spatial Data Used in Ecological Applications. Pages 91–107 in C. T. Hunsaker, M. F. Goodchild, M. A. Friedl, and T. J. Case, eds. Spatial Uncertainty in Ecology: Implications for Remote Sensing and GIS Applications. Springer, New York, NY.

Strandburg-Peshkin, A., D. R. Farine, I. D. Couzin, and M. C. Crofoot. 2015. Shared decision-making drives collective movement in wild baboons. Science 348:1358–1361.

Toledo, S., O. Kishon, Y. Orchan, Y. Bartan, N. Sapir, Y. Vortman, and R. Nathan. 2014. Lightweight low-cost wildlife tracking tags using integrated transceivers. Pages 287–291 in 2014 6th European Embedded Design in Education and Research Conference (EDERC).

Toledo, S., O. Kishon, Y. Orchan, A. Shohat, and R. Nathan. 2016. Lessons and Experiences from the Design, Implementation, and Deployment of a Wildlife Tracking System. Pages 51–60 in 2016 IEEE International Conference on Software Science, Technology and Engineering (SWSTE).

Toledo, S., D. Shohami, I. Schiffner, E. Lourie, Y. Orchan, Y. Bartan, and R. Nathan. 2020. Cognitive map–based navigation in wild bats revealed by a new high-throughput tracking system. Science 369:188–193.

Tukey, J. W. 1977. Exploratory Data Analysis, vol. 2. Reading, MA.

van Gils, J. A., B. Spaans, A. Dekinga, and T. Piersma. 2006. Foraging in a tidally structured environment by red knots (Calidris canutus): Ideal, but not free. Ecology 87:1189–1202.

Visscher, D. R. 2006. GPS measurement error and resource selection functions in a fragmented landscape. Ecography 29:458–464.

Weiser, A. W., Y. Orchan, R. Nathan, M. Charter, A. J. Weiss, and S. Toledo. 2016. Characterizing the Accuracy of a Self-Synchronized Reverse-GPS Wildlife Localization System. Pages 1–12 in 2016 15th ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN).

Wood, S. N., Y. Goude, and S. Shaw. 2015. Generalized additive models for large data sets. Journal of the Royal Statistical Society: Series C (Applied Statistics) 64:139–155.