THE TAG LOCATION PROBLEM

Michael D. Sumner

April 3, 2019

This chapter discusses problems faced with tracking data that concern the estimation of location and provides a flexible software environment for exploring data and applying solutions. Examples are used to illustrate the variety of problems and some of the limitations of the traditional techniques by which tracking data are analysed. The **trip** package is a dedicated suite of tools developed by the author in the software language R. This package integrates data handling and analytical techniques for track data with broader GIS and spatial data tools. This makes track data handling tools easily available, even for those without strong programming skills. The chapter concludes by extending the concerns regarding the limitations of traditional techniques to methods for deriving locations from raw data.

This chapter is not intended to be a critique of modern methods of dealing with tracking data, but introduces the variety of issues encountered and tools for applying them. Simple-to-use tools for handling spatial and temporal data are still rare and some of the problems encountered cause difficulties for researchers before they have an opportunity to explore sophisticated methods. The aim here is to illustrate some classical techniques within a software toolkit that provides better control over the details of analysis tools for those without advanced programming skills. Later chapters present solutions for the remaining problems. Work by **?** and **?** provide a more critical review of recent methods.

Aims of this chapter:

1. To introduce existing problems in tracking analyses presented with examples of classical techniques.
2. To illustrate the complexity of problems and areas that require more sophisticated solutions than traditional techniques. The problems presented here illustrate the need for solutions that come later in the thesis.
3. To present a flexible and readily customized software package as a framework for classical analyses and starting point for more sophisticated analyses.
4. To explain the compromises that are often made with regard to data representation and storage, as dictated by traditional systems, rather than an expressive model of the problem represented by the spatial and temporal data.
5. To encourage the use of techniques for automatic data validation, spatial and temporal data storage and integration with database and GIS technologies.

1 R and the trip package

The software package **trip** developed with this chapter provides an integrated system for data validation and a development framework for track analyses. This can be used as a launching point for further analysis such as validating input to Bayesian methods, or filtering for state-space models (**?**). As an extension of the R environment, **trip** also provides integration with tools for data access and output, integration with GIS and other data systems, and metadata for projections and coordinate systems. The **trip** package ties together numerous tracking analysis techniques, which previously were only available though a wide variety of disparate tools, each having various requirements and limitations.

The **trip** package was developed within the freely available software platform R, a statistical programming environment consisting of a vast community of contributing developers and users (**?**). R is organized into modules known as *packages* which provide the functionality of the language, and also the mechanism by which it is extended[1]. New packages are created using the same tools by which R itself is built and can be contributed to a public repository such as the Comprehensive R Archive Network (CRAN[2]). The repository system for contributed packages is one of the great strengths of R and is part of the reason for its ease of use and subsequent popularity. The spatial and temporal capabilities of R are advanced, including strong integration with other technologies such as databases, GIS software and the wide variety of spatial and other complex data formats.

The data coercion paradigm in R is very powerful, allowing rather different packages to share data with tight integration. There are some fundamental data representations required for spatial analysis and careful organization is needed to provide coercions between different types to get all the tools to work together. The spatial infrastructure used to create the **trip** package is described with examples of using the software in Section 4.

2 Problems with location estimation

This section presents actual tag location estimates to illustrate common problems associated with track data. The location data were provided by System Argos, which is a worldwide satellite service that provides location estimates from small mobile transmitters.

The first example is a sequence of Argos position estimates for a single elephant seal in Figure 1. All raw estimates provided are shown, with the longitude and latitude values transformed to an equal-area projection and drawn connected as a line. While there is obvious noise in the track, the general sequence of events is clear: the seal leaves Macquarie Island, swimming predominantly south-east to the Ross Sea where it spends several weeks, and then returns via a similar path in reverse.

There are a number of problems with the location estimates, some that are very obvious, but others that are more subtle. First, some of the dog-legs in the path seem very unlikely. On the outward journey the blue path shows some lateral movement to the west and east, and just before the return to Macquarie Island there is a similar movement to the west, then east. These are obvious problems seen as noise in the track, with positions that cannot be believed that do not otherwise obscure the general pattern of movement. Other dog-legs in the path are less extreme, so are more plausible.

Another problem is that there are locations that are well within the land mass of Antarctica. For a marine animal, these locations are clearly incorrect but as with the track dog-legs there are similar issues with levels of plausibility that are difficult to evaluate. A location on land may be plausible if it is near enough to the coast, though this can interact with further issues requiring interpretation. These include that the start and end location are not exactly at the known "release" and "re-capture" sight, which was the isthmus at Macquarie Island. This isthmus is quite narrow and is readily crossed by elephant seals, though regions of the island to either side of the isthmus can only be traversed by sea. Another section of the track at the beginning has the path of the animal crossing Macquarie Island itself. At the scale of this image this inaccuracy seems unimportant since the island is such a small area within the study region. However, if the region of land concerned was a much larger peninsula then the problem of choosing which region was actually visited remains.

A scheme that proposes to remove or correct extreme positions faces the problem of defining appropriate thresholds. "Extreme" dog-legs cannot simply be discarded as the question of what is "too-extreme" does not have a simple answer. A simple rule to discard any location on land will not work since these animals do actually visit coastal regions. The distances that a seal might travel inland is not very far but depending on the species studied and the environment the situation may not be so clear-cut.

There are other issues of plausibility. For example, the coastline in the figure is quite coarse and while it may be sufficient for the scale of the image it does not represent the actual coastal boundary available to the seal. The real coastline is far more tortuous, detailed and dynamic—and may be significantly different from a particular data set due to the current fast- or sea-ice cover. This is a general issue with any data set available for informing models of animal location—the assumptions and limitations must be understood and used appropriately.

[1]See `http://en.wikipedia.org/wiki/Package_Development_Process`.
[2]See `http://en.wikipedia.org/wiki/Software_repository`.

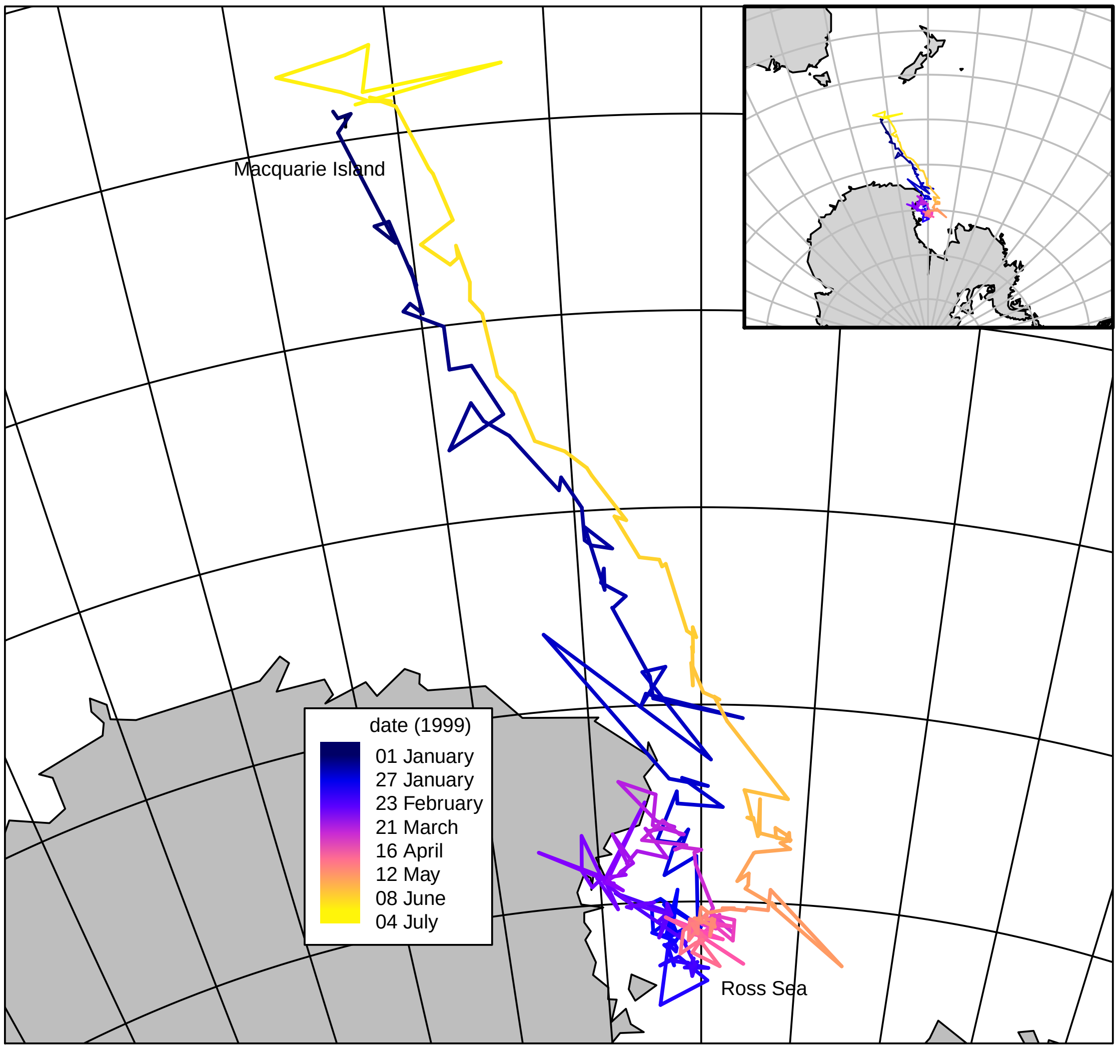


Figure 1: Raw Argos estimates for a Macquarie Island elephant seal. The line connecting the points is coloured from blue through purple to yellow in order relative to the time of each position. (Macquarie Island can just be seen in black beneath the third dark blue line segment). The outward and inward journeys are much faster than the journey throughout the Ross Sea, as shown by the colour scale change. A graticule is shown for scale, 5 degrees on the main plot, and 10 degrees on the inset.

In terms of the incorrect first and last positions in Figure 1, these could be updated to be the actual release and recapture sites, but it might be more correct to actually add those locations and times to the start and end of the sequence of records. This is a data consistency issue that leads to the next family of problems in track data.

2.1 What is a trip?

There are a number of practical issues associate with the organization of track data that can present more prosaic problems. This section discusses some terminology and suggest the use of a "trip" as the unit of interest and that can be defined with database-like validation restrictions. The idealization of an animal's trip is a continuous line in four dimensional space-time that is perfectly accurate as a representation of the seal's position. For practical reasons, this ideal case can only be represented as a series of point samples of uncertain accuracy, in two or three spatial dimensions parameterized by time.

Animal tracking can be carried out in a variety of ways, here restricted to the broad class of data derived from "tagging". A "tag" is a device attached to an animal that is used to directly sense and record data or that is indirectly detected by a remote sensing system. For the purpose of the current discussion, refer to the first type of tag as "archival" and the second type as "remotely sensed". Reviews of the practical methods for tagging in the broader context of biotelemetry for marine and terrestrial species are provided by **?**, **?** and **?**.

Archival tags record and store data that is later retreived from the tag, while remotely sensed tags emit an electronic or acoustic signal that is detected by an installed system of sensors, such as a satellite or acoustic array. (This categorization is not always maintained in practice, as archival tags may be satellite-linked in order to upload data in near real-time, but for the purpose of location estimation the distinction holds for the types of available data).

A loose set of definitions then is:

tag the device put on the animal.

track data any temporally referenced location data resulting from a device attached to an animal.

trip a specific tracking "interval" where a tag was attached, the animal is released for a time, and the tag (or just its data) is eventually retrieved. A trip may be represented in some way with track data that has some quality control or modelling applied.

Data resulting from the tagging process are identified by the device ID, as well as by the identity of the individual animal the tag is attached to. The same tag might be put on different animals, or the same animal might have been tracked on two separate occasions. For central-place foragers, a single tagging event may involve the same animal leaving the tagging site and returning multiple times. Once the interval of interest is defined it can be regarded as a trip. A single leave / return or a tag deployment / retrieval may be referred to as a trip. Whether multiple leave / return events are considered within a single trip depends on the research question—migratory animals usually won't have such clear trip boundaries as central-place foragers, for example. The difference is important as the behaviour of interest for central-place foragers is primarily between leave / return events, and the return event is usually the only opportunity to retrieve a tag. Finally, there may not be data for the entirety of the trip of interest due to tag loss or memory restrictions, and so require the inclusion of trip sections where location is uncertain or completely unknown.

For the current discussion, define a trip to coincide with the interval of interest for which there is useable data from the tagging process. "Tracks" or "track data" then are just a set of location and other data, with no particular organization or quality control.

2.2 Practical data issues

The minimum organization and quality control for trip data involves the ordering and relation between data records. The ordering of records is perhaps inconsequential, as there is the inherent order of the date-time value stored for each location, but this may reveal more basic errors in the data. There must not be occurrences of duplicated date-time records within a single trip, although duplicated locations in subsequent records are acceptable. Duplicates in time do not make sense since either they are redundant copies of a previous record, or there is an implied infinite speed. These are common from the Argos Service, an example is found in the `seal` data example given by **?** which is used in Section 4.2.[3] Analytical methods sometimes apply a non-zero time difference arbitrarily to avoid divide-by-zero errors. Less serious is the issue of successive duplicate locations, but care must be taken when calculating metrics such as speed based on inter-point distances. Each of these cases should be investigated carefully in case they hide errors from other causes such as mistaken data handling.

Missing values must also be handled carefully. Location and time coordinates cannot be used if they are missing or non-finite, even if their record appears in the correct order. Missing values can arise in a number of ways—infinities or undefined numeric values from numeric errors, or out of bounds coordinates, transformation errors, data missing from a regular sequence—and the exact reasons need to be carefully understood.[4] This is a different approach taken to that of **?** who explicitly allow missing coordinates as part of "trajectories". This is most pertinent in the context of tracks of regular time intervals where a missing point can be significant in terms of interpretation. The definitions here are not intended to determine which approach is more appropriate and there is no reason the two rationales cannot co-exist, but the current implementation in the **trip** package disallows missing coordinates.

From a programming perspective, the use of rigid classes (definitions) with validity checking can significantly reduce the time wasted solving these problems (**?**). Based on the above, the minimal data-consistency preparation required can be achieved in the following way. Read all records, sort by trip ID then date-time, remove duplicated records or records with missing or non-numeric spatial or temporal coordinates. (The definition of "invalid" for a coordinate may involve out of bounds values such as those for longitude and latitude, but this step only refers to the data values, not their interpretation). Remove or adjust any records with duplicate date-times within a single trip ID. Up to this point no interpretation has been applied to the data—this will provide a useable set of records that can pass minimal validation but each step should be carefully investigated to ensure that automated decisions are not introducing new errors.

One way to adjust duplicate times is to simply modify the values forward or back by a small amount, but this can be problematic depending on the time differences involved. The reason for duplicated times is more likely to be a problem with the data itself and should be investigated.

Other problems in this regard deal with the sensibility of movements in a particular coordinate system. The most commonly used coordinate system for tracking data is longitude and latitude on the WGS84 datum. For animals that traverse hemispheres and cross critical meridians such as the Pacific Ocean dateline (longitude 180 W / 180 E) or the prime meridian (longitude 0) a continuous path must be represented appropriately, such as longitudes in [-180, 180] or [0, 360] respectively. Many species will cross both these critical boundaries and so representing simple lines requires a smarter choice of map projection. All map projections have these regions of non-optimal usage and so the focus should be on intelligent choice of projection using tools that provide easily applied transformations.

2.3 Joining the dots

A further problem is the common practice of joining points with "straight lines". Usually the only available data are temporally referenced point locations, and lines are artefacts introduced for visual purposes. However, using these lines is quite artificial, and can become error prone when used quantitatively. Joining the points imposes a specific model of behaviour, namely that the path is a straight line between points.

This is not correct on several levels. First, the animal is moving in three spatial dimensions not two, and the movement in this third dimension is quite significant for diving animals, though it may be largely ignored for many flying or surface dwelling species. Second, even if the points represent accurate positions for the animal the line joining them most certainly does not represent the intermediate path correctly. The

[3]Another recent example of duplicated times in a GPS data set is discussed here: `http://lists.faunalia.it/pipermail/animov/2010-August/000635.html`

[4]A natural assumption is that recorded values of date-time are correct beyond question: so there is some information even if one of the spatial coordinate values is missing. This issue is a corollary to the use of filtering techniques that remove locations from track data or otherwise correct spatial positions. If there is a date-time why not interpolate or otherwise estimate missing spatial coordinates?

animal could be traversing either side of the line, or taking a far longer, more convoluted path. Thirdly, the coordinate system used to interpolate the intermediate positions can have a large effect on the outcome. "Straight-line" movement is usually assumed, but what is drawn as a straight line on a map has a very different meaning depending on the coordinate system or map projection used. For most coordinate systems shorter step lengths will be closer to the "great circle" path, but the nature of the deviation will also depend on the region traversed.

Joining points with a line guides the eye helpfully to show the sequence of points, and the mind can often overlook problems of inaccuracy to see roughly what actually happened. It is this mental capacity for reducing noise and seeing the overall picture of events that sophisticated models of track analysis and movement aim to replicate in an objective way. When our minds provide these ideas they do so by applying knowledge of the physical system: an animal swimming through water over great distances, an animal that will tend to travel quickly to an area of interest, then spend time in localized regions, an animal that will not venture far from the coastline to areas inland, etc. "An effective EDA [Exploratory Data Analysis] display presents data in a way that will make effective use of the human brain's abilities as a pattern recognition device" (**?**).

There is no end to this problem when dealing with points or lines segments themselves as the entities of interest. If a particular position is just slightly wrong, and its neighbouring points also a little inaccurate then any assessment of the distance from one point to another or the intermediate path taken between points is thrown into question.

Treatment of spatial and temporal data in modern software

The temporal nature of track data stems from the fact that the physical process of animal movement is a continuous path. This continuous process is only measured by discrete samples and so the data are inherently discontinuous. However, treatment of time in software is rarely truly continuous but rather applied as a sequence of "time slices". This is a legacy limitation that unfortunately matches the way in which track records are usually measured and stored. To choose a high-profile example, animations of tracks in Google Earth (**?**) show sequences of discrete line segments that are progressively revealed or hidden as the slider intersects the time spans of the segments. Why is the line not represented as a continuously changing entity, with extents that match the slider's extent exactly? Partial line segments could be shown, and the line shown continuously without being so bound to its input points. This is a problem not only for Google Earth but a true limitation in the representation of most spatial data in GIS and GIS-like visualizations.

This must be part of the reason why tracking analysis is rarely tightly coupled with GIS—analytically (if not visually) track data is treated as continuous or near-continuous, with more information than the static line segments joining subsequent points. Also track data is routinely processed based on great circle travel (assuming that's how the animal would choose to move) but then presented visually in a simple 2D longitude by latitude plot. Map projections provide visualizations that help prevent our brains from giving us the wrong message about distance and area on a simple plot. Ultimately a 4D visualization on a globe may be a "truer" way to visualize track data, but though current tools such as WorldWind and Google Earth will draw lines along great circles they are not well suited to track data that varies continuously in time.

GIS traditionally provides complex shapes such as multi-segment lines with multiple branches, or polygons with branched holes and islands but support for a third coordinate value for elevation is rare and time is usually non-existent.[5] Though routine computer graphics in games provides complex surfaces and lines composed of primitive elements with incredibly complex representations and interactions, it is rare to find treatment of track data as a multi-part line object, let alone with fine control over the continuous span of a single line. Modern GIS in its most commonly understood terms is not easily extended for temporal data, but provides an excellent platform for dealing with data sources, geometry and gridded data, and working with projections.

The availability of data manipulation and analysis tools is a major factor in the effective use of animal tracking data for ecological studies. While there are many analytical techniques and a wide array of

[5]Polygons are literally incapable of representing continuous 2D topological surfaces in 3(+)D geometric space and the special status of planar polygons that imposes this limitation surely will eventually be transcended by future GIS technology.

software applications, some lack the flexibility required or are restricted by cost or the required expertise to use them effectively. For some purposes track data needs to be represented as points, and for others as lines, or even better as probability density functions. Tools for seamless conversion between these data structures are practically non-existent for everyday research.

An illustrative example of the limitations of GIS data structures is seen when attempting to represent track data. As points, the geometry is stored as X and Y coordinates (and, rarely, with Z coordinates). Time is relegated to the attribute table and even then is not always supported completely by common GIS interchange formats.[6] GIS supports more complex geometry than simple points requiring more than one vertex: lines, polygons and "multipoints". It should be simple to store a track in either a "point" or "line" version, but for lines each line segment is composed of two vertices so there is no longer a simple match between a point's date-time (or other) coordinate and those of the line. The line is represented as a single object with multiple X and Y vertices with only a single attribute record, or as a series of line segments composed of two X and Y vertices each. Neither version provides a clean translation of even very simple track data to a GIS construct.

Access to the true continuous nature of track data is simply not provided by common software tools. This is a problem that extends to a general definition of topology versus geometry, for representing objects flexibly in a chosen space but discussion of that is out of scope here. **?** highlight the need for ecologists to become more adept at matching temporally varying environmental data to animal movement data. There are emerging technologies that allow for a separation of geometry and topology, unlimited coordinate attributes on GIS features, and generalizations of this are within the scope of general database theory (**?????**).

2.4 Summary of problems

The main problems can be described as a set of overlapping issues:

Inaccurate sampling Position estimates are inaccurate, with some unknown relation to the true position.

Irregular and incomplete sampling Position estimates represent discrete samples from an animal's continuous path. These may be at irregular time intervals with large gaps in the time series, and no ability to control this because of practical limitations.

Incomplete paths Paths composed of too few positions, inconsistent motion and assumptions about straight line movement.

Unlikely dog-legs There is no sense in the animal being so erratic.

Simplistic models of movement and residency Intermediate locations are shown by joining the dots, using an assumption of direct linear motion between estimates.

Many traditional analyses of modern track data deal with these problems by chaining a series of improvements in an *ad hoc* way, and the need for better approaches is well understood (**??**). Incorrect positions are removed by filtering, based on speed, distance, angle, attributes on the location data or spatial masking. Positions are updated by correction with an independent data source, such as sea surface temperature (SST) or the need to traverse a coastal boundary. Unlikely dog-legs are removed by filtering, or "corrected" by smoothing the line. Smoothing is also used to augment small samples, by interpolating along a smooth line, or smoothing positions into a 3D grid partioned by time. There are further requirements for smoothing to estimate latent locations or to match disparate spatial and temporal scales.

Many of these techniques have their own problems, compounded when these operations are chained one after the other. Models of the process may be overly simplistic (linear movement between estimates), or applied inconsistently—positions are removed, then estimates are smoothed, or compared with other data to correct or update them. Later chapters present new methods for incorporating these issues in a more integrated way.

[6]The obscure "measure" value for a fourth coordinate in shapefiles is sometimes used for time, but was not designed for it and is rarely supported by software packages.

3 Summarizing animal behaviour from point-based track data

This section revisits some of the problems presented previously and looks at the details of algorithms used. The techniques are useful for first-pass summaries, or exploring ideas, but they rely on simplistic models and are difficult to integrate sensibly.

Putting aside the limitations mentioned earlier and the fact that there is no clear basis for deciding which combination of tests should apply, some of the issues can be illustrated further by proceeding with these simple to more complex filters.

3.1 Filtering

Filtering is used to remove or modify data in some way based on metrics available or calculated from the track data. Destructive filters categorize locations for removal from the trip. Non-destructive filters update the location data for some positions. Again there is no clear distinction between these two types as a filter can be used to discard some locations entirely, update others and interpolate new locations for various purposes.

At the simplest level, destructive filtering involves categorizing each point for removal or retention. An example is a "land mask" that would deal with the issue of the points on the Antarctic continent as discussed in Section 2. A land mask filter would determine any points that fall on land, mark them for removal and discard them. The filter is very simple as it is a basic check for each point individually, with no interaction of its relationship to other points or other data sources. All points that fall on land can be determined and removed in one step, or they could be checked one after another in any order. The way the filter is applied will have no impact on the filtered result.

A more complex case applies recursion, where once some points are removed the status of the test for remaining points may have changed and so must be determined again. Metrics on sucessive locations fundamentally rely on the order and relation betweeen points, and so once points are removed the calculations must be repeated until the desired metric is reached for all retained points. Existing filters apply measures such as Argos location quality, distance between successive points, speed of movement, turning angle and land masks. A classic speed filter in marine animal tracking is a recursive rolling root-mean-square speed filter by **?**. This filter is widely used and widely cited especially in marine applications.

There is a practically endless number of metrics that can be derived from the location data that range from the very simple to complex. However, no matter what combination of decisions are applied, the main limitation of these methods is their arbitrary nature. They are applied to a purely geometric interpretation of the data that laregely ignores the physical system being modelled. Much information goes unused, and what data is used is applied independently of other sources.

The use of destructive filters is also problematic because data is discarded and the filter decision is based on the location itself, rather than the process used to estimate the location. It is hardly ever mentioned, but the Argos Service estimation process is not published and therefore not amenable to modelling.

Recursive filters are relatively complicated, but still result in a categorical decision as much simpler filters like a land mask—there is no single number that can be calculated for a given point, and the implications of minor decisions for a given filter can greatly affect the result.

Destructive filtering

Here the use of a two types of destructive filter are demonstrated to remove points based on a land mask, Argos quality and speed calculations. In Section 4 the **trip** package is used to create a version of a speed-distance-angle filter.

The Argos Service is a global satellite data collection system that provides an estimate of position based on doppler-shift signals transmitted from tags (**?**). The basic location product is a time series of longitude and latitude values with a categorical measure of location quality that is provided as a label. There is more information available with the service and guidelines for its use, but the scope of the following discussion is restricted to the widely used quality class measure. Location classes take the values "Z", "B", "A", "0", "1", "2", or "3" in order of increasing accuracy. The reported accuracies are 1000 m for "0", 350-1000 m for

"1", 150-350 m for "2", and better than 150 m for "3" (**?**). No estimate is given for "Z", "B" or "A" classes although studies have shown that these can have an acceptable accuracy (**?**).

The first filter removes any points that fall on land, and then any points that have an Argos class of "B" or worse. In Figure 2 the two panels show the Argos track plotted in longitude and latitude coordinates.

In the first panel the original track has been filtered for points on the land area, and the second for points that have a location quality class of "B" or lower. The land filter only applies to six points in the track, but the class filter has removed 214, which is a majority of the available 351 points. The effects that these filters have are independent of one another and it would not matter which were performed before the other, although the result may be quite different in combination that in the use of either one alone.

In terms of the land mask, the filter succeeds in removing points from land but there is still a line segment that crosses a portion of the Antarctic continent. A filter that applies to points is quite different to one that applies to a continuous line segment. The class filter provides a very cleaned-up track but at the expense of discarding more than half of the available data.

The next example demonstrates a recursive speed filter applying the method of **?** to the Argos track. This technique specifies a running root-mean-square (RMS) summary of speed calculated by determining the instantaneous speeds for each point to each of its two previous and next locations. The RMS is defined as the square root of the mean of the speeds squared. Any sequence of locations with an RMS exceeding the maximum speed value have the peak RMS location discarded and the RMS is recalculated. This continues until all locations have an RMS less than the maximum speed. The threshold applied in this example is 12 km/h. Again the track is presented as longitude/latitude coordinates with distance measurements calculated relative to the WGS84 ellipsoid.

In the left panels of Figure 3 are two plots of the RMS values, for the second iteration after some positions are removed and the sixth iteration when only a few nearby positions remain above the threshold. The unfiltered RMS values are shown as a grey line representing the original points, the current points that have RMS values above the maximum are shown as crosses and the points below the maximum are shown in faded black. The threshold speed is shown as a horizontal line. As successive peaks of RMS values above the maximum are removed the categorization for the remaining points changes. This filter took ten iterations to remove all of the 73 offending locations, and the resulting track is shown in the right panel.

This result seems reasonable, though there is no end to the complexity and arbitrary nature of the decisions to be made. There are practical uses for techniques like these however. Speed filtering can give a reasonable result and there is a need to quantify the limits here with comparative studies of various algorithms on known data sets.

There are more sophisticated filtering algorithms that apply a suite of tests. A published speed-distance-angle filter first removes a minimum class (such as Z), filters for speed with the McConnell algorithm and then recursively removes further offending points that have a combination of acute angles and long implied distances travelled (**?**). However, the accompanying software is quite specific and unrelated to other packages. In Section 4 a speed-distance-angle filter is created from scratch using tools available in the **trip** package.

Related filters applying a combination of metrics have been published by **?**, **?** and **?**. The algorithm published by Austin is available in the R package **DiveMove** (**?**) and those by **?** and **?** in **argosfilter**. **?** gives an overview of the relative merits of various techniques, and the need for hierarchical techniques to avoid discarding good quality locations.

Another decision process is to choose between two possible locations for each time step provided by the Argos Service in the DIAG format **?**. The Douglas Argos-Filter Algorithm applies this as part of the filter (**?**).

3.2 Non-destructive filtering

Non-destructive filters aim to update track points by some mechanism without removing corresponding data records. There are many examples such as algorithms to modify light level geo-locations for latitude with SST (**?**) and interpolative techniques to smooth tracks (**?**). Many destructive filters can be recast in a non-destructive form using a penalty smoothing approach in the style of **?**. For example, rather than filter the track to exclude locations that would imply an unrealistic speed of travel, the track can be smoothed

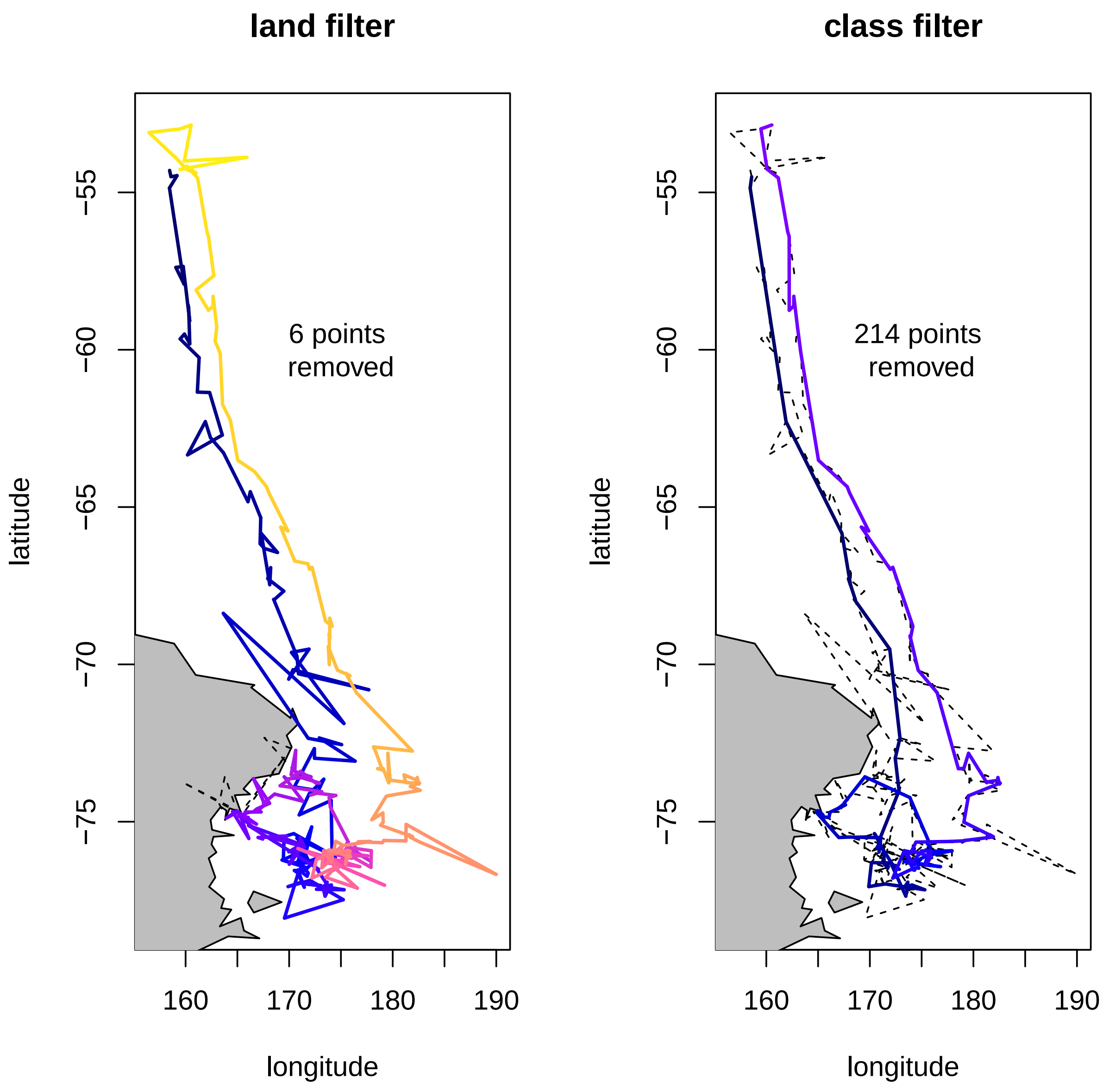


Figure 2: Land filter and Argos quality class filter (> B). In each panel the state of the track prior to filtering is shown as a dotted line.

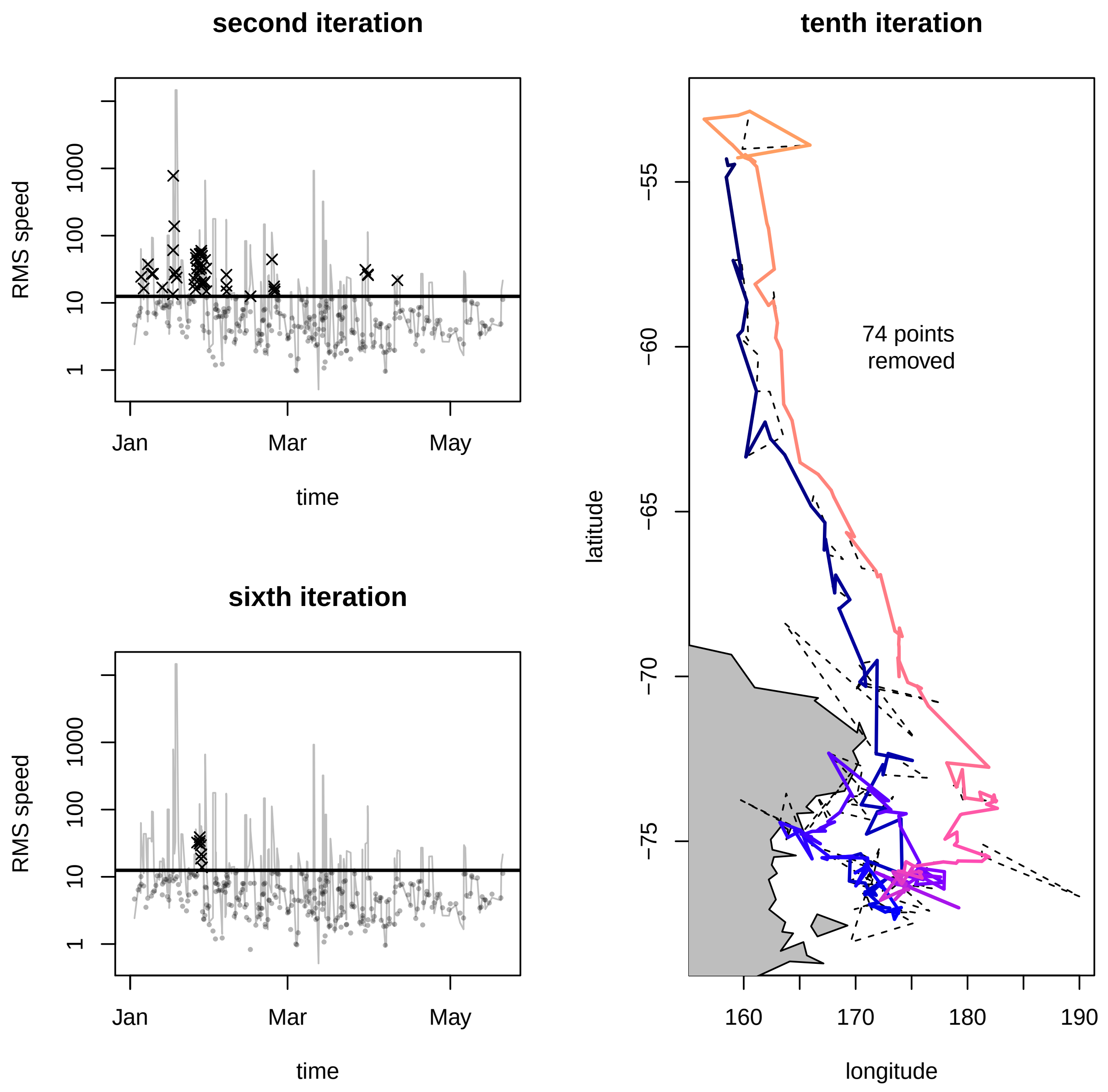


Figure 3: Recursive speed filter applied using the McConnell method. The left panels show the classifications provided for each point (RMS above the required maximum speed). The RMS speed axis is logarithmic.

using a speed of travel penalty. The smoothed locations are determined by minimizing the functional

$$J_\lambda(\hat{x}) = \sum_{i=1}^{n} d(x_i, \hat{x}_i)^2 + \lambda \sum_{i=1}^{n-1} v(\hat{x}_i, t_i, \hat{x}_{i+1}, t_{i+1})^2$$

where $\{x_1, \ldots, x_n\}$ represent the raw (unsmoothed) locations, $\{\hat{x}_1, \ldots, \hat{x}_n\}$ their smoothed counterparts, $d(x, y)$ represents the distance from x to y, and $v(x, t_x, y, t_y)$ the speed required to travel from x at time t_x to y at time t_y, and λ is the smoothing parameter. The first term is a measure of goodness of fit of the smoothed locations $\hat{x}_i$ to the raw locations x_i, while the second is a speed penalty. Minimizing J_λ trades off goodness of fit against speed of travel. When $\lambda = 0$ the smoothed track reproduces the raw track exactly. Increasing λ favours tracks requiring less extreme speeds, at the expense of reproducing the x_i.

As for the more traditional application described by **?** this process can be interpreted in a Bayesian context. Adopting a penalty based on squared speeds is equivalent to adopting a Gaussian prior on speeds, adopting a penalty based on the absolute values of speed is equivalent to an exponential prior on speed.

An application of this non-destructive filter was applied to the example Argos data set discussed earlier. Figure 4 shows the filtered result on a map and Figure 5 shows the same result with longitude and latitude plotted against time. The result seems reasonable with a plausibility comparable to that of the recursive speed filter with the advantage of not having removed any data from the trip.

3.3 Spatial smoothing—surfaces from track data

There are a number of ways of creating bivariate (surface) estimates from track data, the most common involve a surface defined by a grid of cells or connected points. (Regular grids are historically easy to store and to compute, and so are applied most commonly—irregular grids and meshes are not considered here). The most direct methods are variously called "rasterization", "gridding" or "pixellation"—these effectively generate a bivariate histogram from the input point or line data. Kernel methods apply a bivariate distribution (such as a Gaussian) to the input points or lines. Point methods are not discussed here, but can be achieved in a similar way to the line-based examples shown in Section 4.

The following is a very simplified folk history of tracking techniques, but does help explain some of the existing practices that differentiate terrestrial and marine applications. There is an apparent difference between terrestrial and marine applications in the way that surface creation, or gridding methods are applied. Terrestrial applications tend to ignore time or adapt data to avoid temporal auto-correlation. VHF techniques were the original primary tool for wildlife tracking, and the metric of interest was pure residency—the minimal region in which the animal is present (home range) and the animal's core region. In this context actual "tracks" are not the main interest. Marine applications have traditionally been explicitly interested in tracks and the temporal relationships, perhaps because of the real and perceived differences in the dynamics of marine environments. Modern techniques are seeing a far greater cross-over in these originally different fields and the differences are now out-weighed by the common goal of reliable location estimation.

Cell binning and kernel density estimation (KDE) can unproblematically convert a linear geometric track representation into a 2D histogram-like smoothing of residency, time spent map, or other utilization distribution but both must grapple with serious problems of interpretation. Points and lines simply do not represent the movement of an animal completely. Both methods must deal with issues of independence, point or line interpretations and complex boundaries and environmental relationships. If these issues can be dealt with or ignored, both cell gridding or KDE can be used as a convenient smoother for track data.

The following combinations of methods are applied in various ways in many existing publications. The grid surface is generated by operating on points or on line segments. Line segments provide a continuity through space and so they provide a natural way to connect regions visited by the animal. Lines present a harder problem to convert to a grid than points and so this is often approximated by providing interpolated points in place of line segments, assuming constant travel speed.

The influence of the points or lines on the surface is calculated by binning into the overlapping grid cells directly, or by calculating the contribution to neighbour cells via a "kernel". In the case of binning, the coverage provided by overlay with cells is not great, leading to a requirement for larger bin sizes, compromises

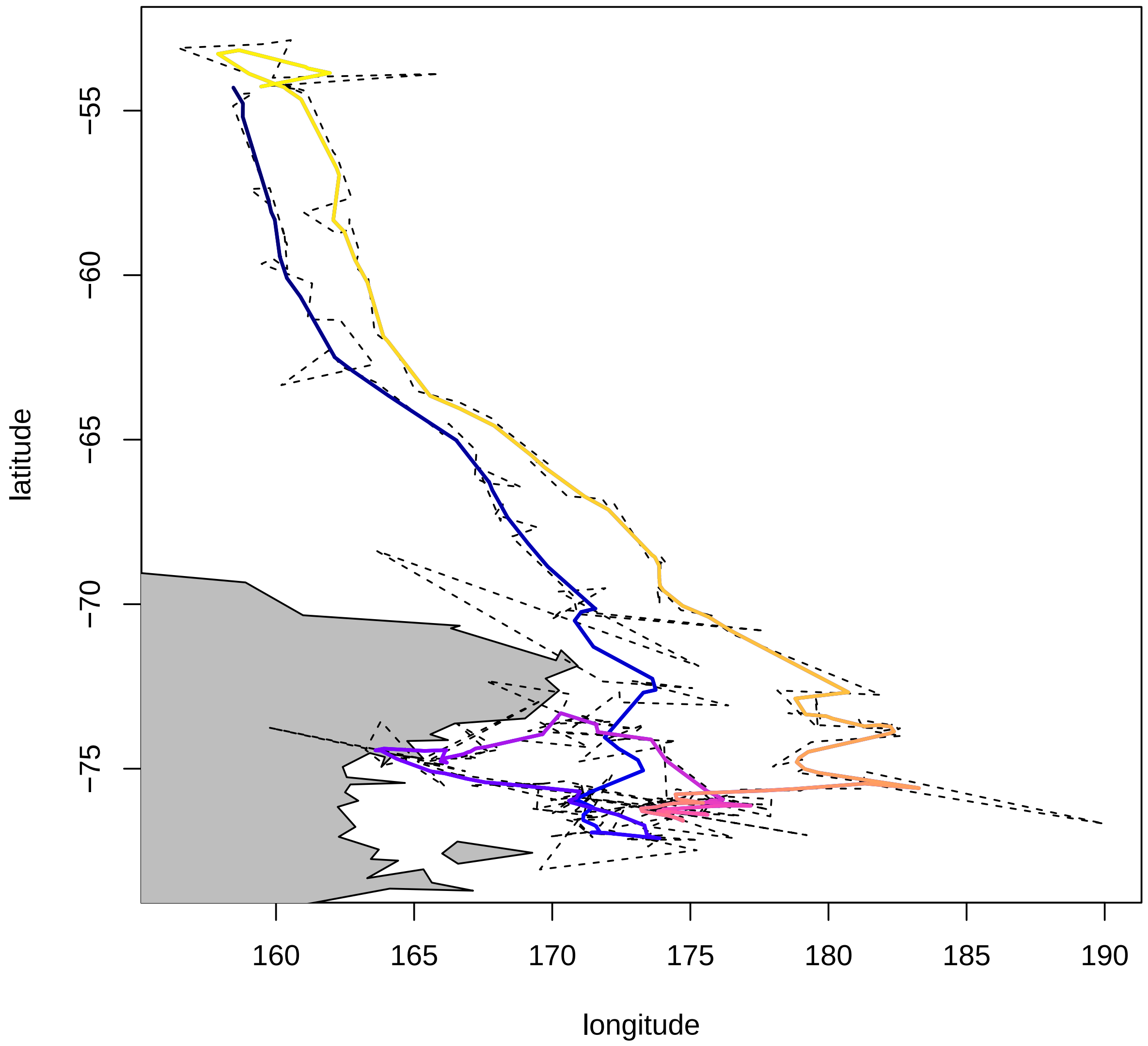
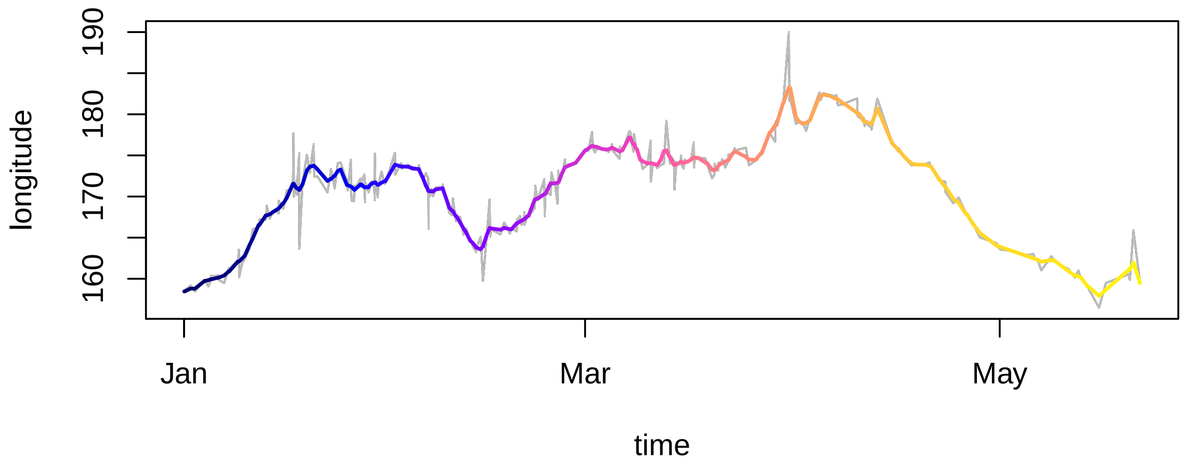


Figure 4: Argos track filtered by penalizing by sum of squares speed. The original data was used without resampling



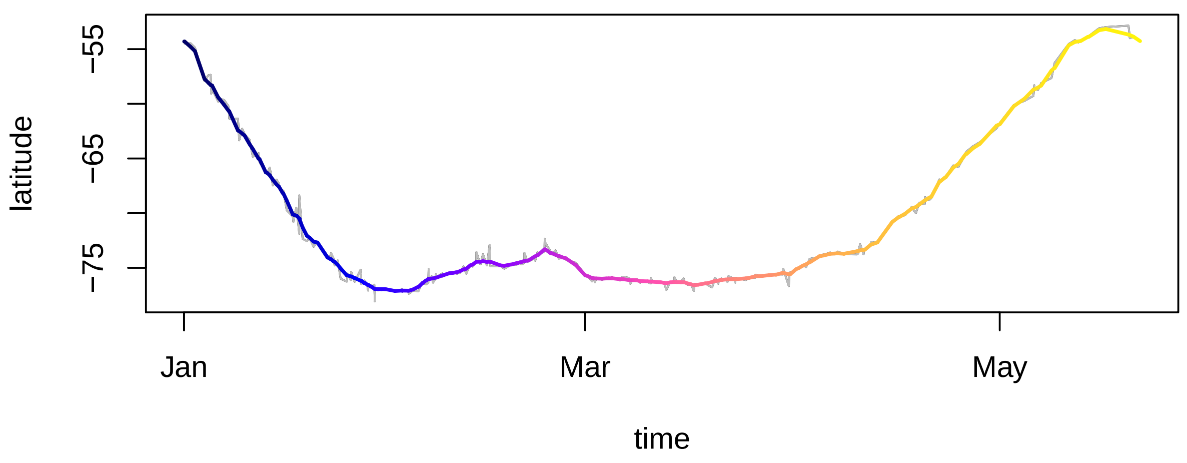


Figure 5: Penalized sum of squares speed—track longitude and latitude.

balanced against positional accuracy and the need to match with covariate data **?**. The limitations here are the same as those for histograms in general—the discontinuous histogram presents analytical difficulties as it quite senstive to the chosen origin, bin size and orientation (**??**). Kernel density has the advantages that the result is smooth without the blocky, discontinuous nature of a bivariate histogram. Regular or irregular "wireframe" representations give a smoother result and have continuous analytical and visualization counterparts via interpolation, but these data structures are more complicated to calculate and are much less widely supported.

Finally, when the cell value is determined there are various ways in which the contribution of a point or line can be calculated. The point or line can simply be summed into the cell—a point is counted as present in the cell whereas the proportional length of an overlapping line segment is added to the bin. Each bin contribution can be multiplied by a factor, such as a point value or the time interval available to a line segment. This is one of the natural cases for using line segments from track data rather than points, as when the duration of time is of interest the input points cannot represent the duration of time. Many studies approximate this by creating a regular time series by interpolation.

When kernel density methods are applied, a weighting factor can be given for either dimension of the kernel, or more generally a two-dimensional correlated weighting is used (**?**).

The following examples show the gridding of a very simple track by some different methods.

Exact gridding

Gridding (pixellation or rasterization) methods generate a grid of cells that extend completely over the region of input track data.

A very simple track is shown in Figure 6 with an underlying grid. Each line segment has a value assigned to it, the time duration between each track point. These values are [2, 3, 1, 7, 1, 1] and the first and last line segments are quite long, so their contribution to the cells is small relative to the middle segments. This is reflected in the grid, which represents the implied "time spent" in each cell. In order to determine the contribution of each line segment's value to a cell, the segment is split on the boundaries of the cells that it crosses and the value shared proportionally based on the length of the resulting segment-portion. This computation is not simple—to describe it in GIS terms this is an overlay of the line and the cells ("topology overlay") with a rule to transfer values from the lines to the cell, in this case a proportional sum. GIS can be used to perform these calculations, but as discussed in Section 2.3 working with time in GIS is not well supported and this must be done as an attribute on line objects, rather than on inherently continuous lines that vary through time or other dimensions as well as space.

The dependence of the gridding method on bin size is easy to see. Figure 7 shows the same gridding process applied to a finer grid, with the same origin. The grid again completely summarizes the contribution of the track, but the actual regions influenced by the grid are quite different—the small grid cells snugly trace the track and cover a much smaller overall area than in the case of the coarse grid. The total time duration represented by the coarse and fine grid is exactly the same, but the results are quite different.

This dependence on scale has been used explicitly to provide a compromise between positional accuracy and spatial coverage with the need to match with environmental data by **?** and **?**. **?** also explicitly used grid size to help acccount for spatial uncertainty. As discussed previously the assumptions made by these analyses become substantial, involving issues such as uniform location accuracy, uniform grid scale and constant straight line travel.

Kernel density methods

Kernel density methods are applied to overcome the dependence of the probability density function on the origin and bin size, as it is the properties of the kernel that dictate the contribution to each cell. (In theory every input point or line contributes to every cell to some degree). The result is thus less related to the data structure choices, given a sufficiently fine resolution. Figure 8 shows the same coarse simple grid with an equivalent KDE grid in a perspective plot. The KDE result is still blocky, but the overall value of the surface is continuous, and not restricted to regions near the line. By increasing the resolution, the continuous nature of the KDE approach becomes more apparent—this is shown in Figure 9, again equivalent to the vertical view of Figure 7.

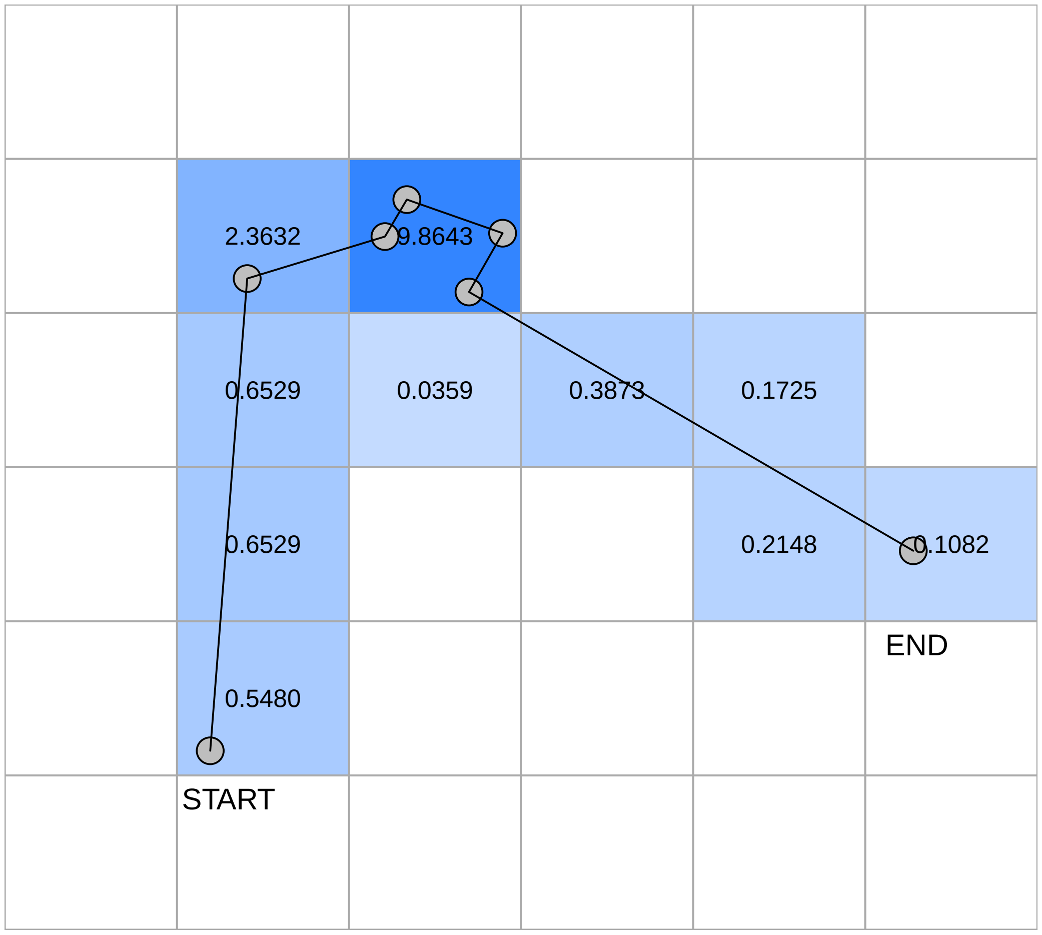


Figure 6: Track lines binned into a coarse grid, using a line segment value to sum into each cell. Cells are coloured from light to dark blue for increasing values. The sum of values resulting from each smaller line segment is shown for each cell.

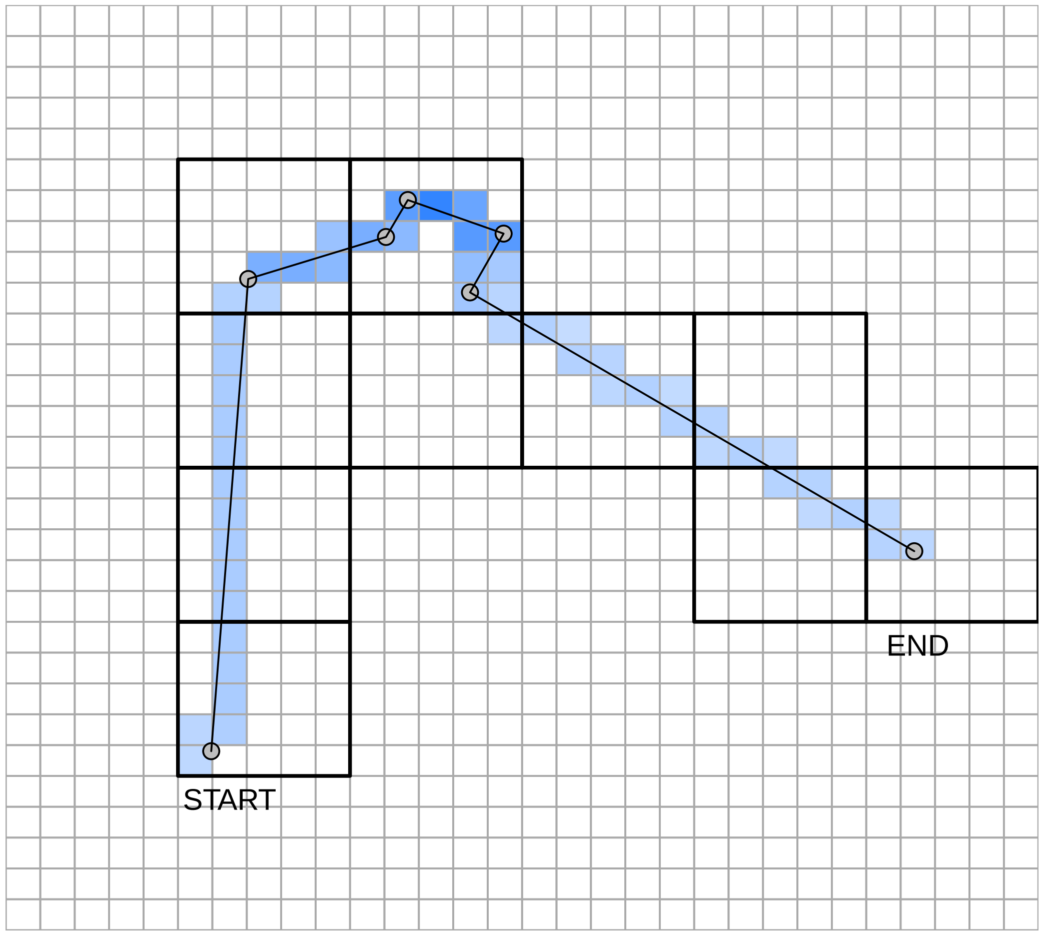


Figure 7: Track lines binned into a fine grid. The track and grid origin is the same as in Figure 6. The coarser grid outline is shown for occupied cells as thick black lines.

Kernel methods are more commonly used in terrestrial home-range applications than marine applications, perhaps as the measure of interest is a spatial region, rather than the actual trajectory of the track. Explicit line-to-cell methods have been used in marine applications because of the focus on time spent based utilization distributions. There is a convergence of aims here that is still developing, and is seeing resolution in modern statistical methods.

Kernel density methods are commonly applied to track data because of these advantages, but perhaps due to implementation difficulties with using line segments published applications use an equal-time interpolation between track points (**?**).

The **trip** package provides tools to produce simple and KDE grids with line or point interpretations. The distinction here is one between discrete and continuous representations which can be represented in the data itself, or in the visualization technique or analysis used. The lack of clarity for these distictions in spatial software is one of the problems faced in tracking research.

Other approaches have tended to use the locations more directly, by categorizing them as belonging to "focal foraging areas" (**?**) or with "first passage time" analysis (**?**).

Attribution of point or line values

Time duration between measured locations is a common value of interest, but many others may be used such as drift dives (e.g. **?**), maximum depth, temperatures encountered, etc. The **trip** package provides tools to apply arbitrary values to gridding methods by providing conversions between point and line based interpretations and the ability to store multiple attribute values for points and lines.

3.4 Partitioning tracks and grids into time periods

Gridding methods are easily applied to entire tracks by binning into a single grid. In order to partition track data into time periods the lines (or other approximation to continuous path) must be cut into time durations. This requires that the path be cut at a point intermediate to the input points as these will rarely coincide with the period boundaries.

Once the trips are cut into exact time boundaries summaries derived from them represent some form of 3D grid in order to represent each time period separately. The track line must be cut exactly at the time boundaries, assuming constant travel speed and the contribution of each part summed separately into two grids. The **trip** package provides tools to partition sets of tracks in this way and generate collections of spatial grids. Using the simple methods above the trip data can be partioned in time to give temporal-based utilization distributions (**?**).

A more general method of representing arbitrary time periods from modelled tracks is discussed in Chapter **??**.

3.5 Map projections

A paper published by **?** details the importance of the choice of map projection for analysing migration routes. Despite the wide availabity of software tools for working with map projections[7] their use is still virtually non-existent in modern tracking studies.

Distance and direction are the most obvious metrics that are calculated and many studies only use spherical approximations which are accurate except in polar regions (**?**). These approximations will be accurate for most purposes but visualization is still usually done using an equal-angle projection—plotting longitude and latitide directly on the x and y axes. Other studies provide ample justification and explanation for working with map projections, so only the importance of their use for the line-interpretation of a track is presented here. The **trip** package provides access to a full suite of projection transformations via the spatial support of the **rgdal** in R (**?**). A widely used transformation library with string codes for map projections, PROJ.4, is used to specify the required metadata and calculations (**?**).

Figure 10 shows that the perceived error in the Argos positions in the east/west direction is far less and it is clear that intuitive ideas about straight line travel are not easily conveyed in longitude and latitude

[7]See `http://trac.osgeo.org/proj/`, **?**, `http://www.eos.ubc.ca/~rich/map.html`, `http://www.manifold.net`, `http://gmt.soest.hawaii.edu/` for general projection transformations, and **?** for working with orthodromic and loxodromic paths.

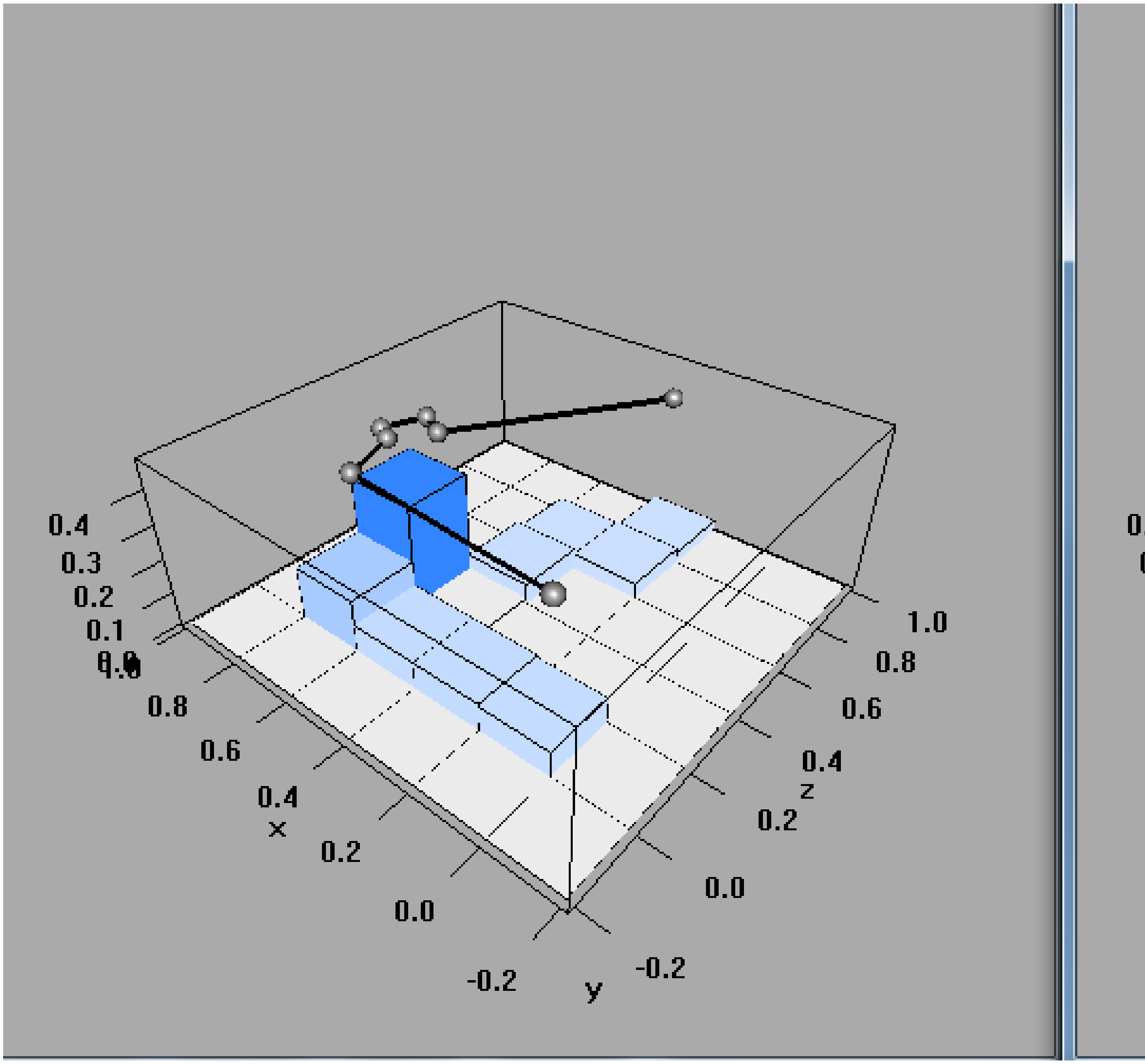


Figure 8: Side by side plots of line gridding and KDE gridding to a coarse grid. The input line is shown at an arbitrary height above the cells.

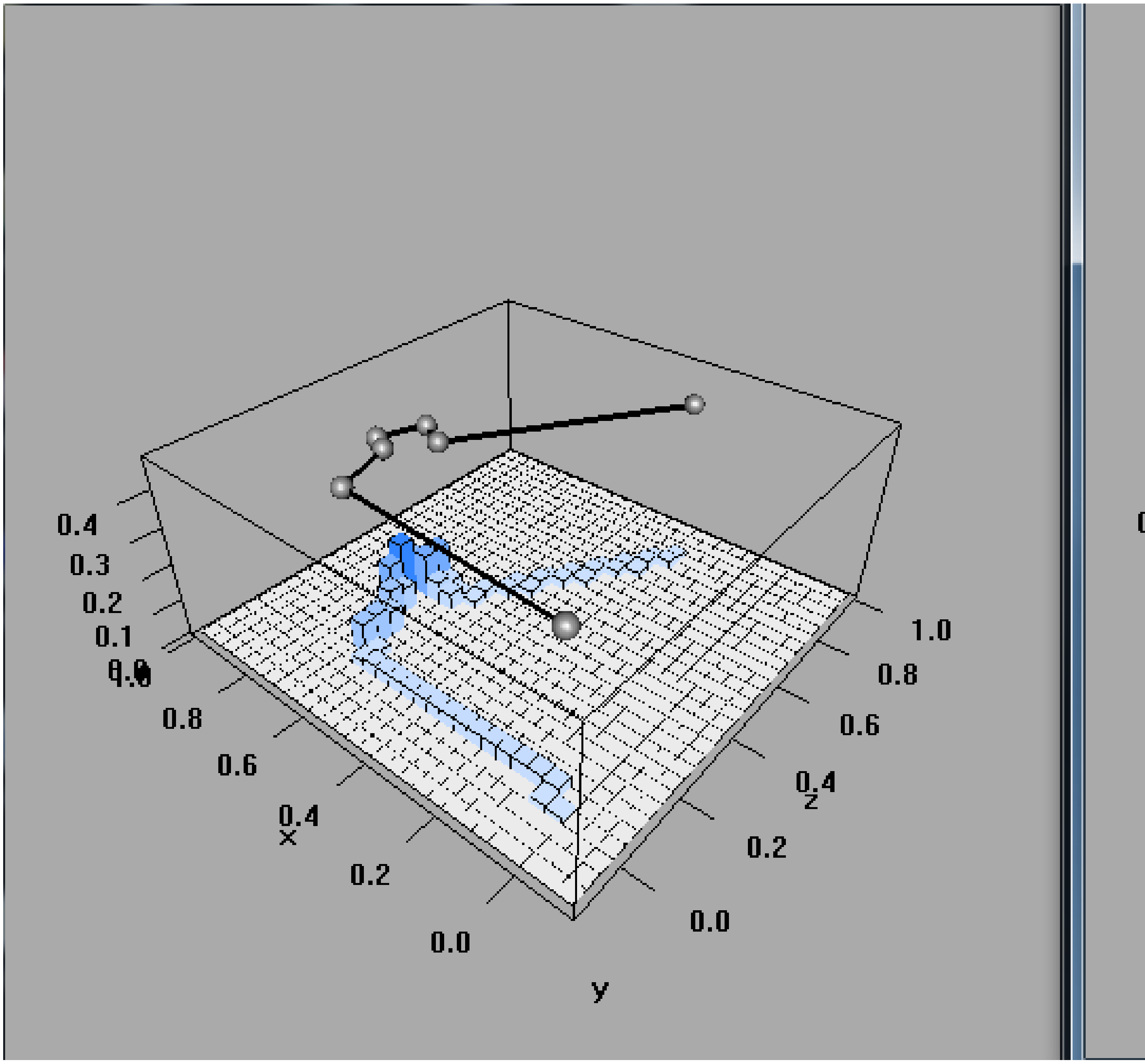


Figure 9: Side by side plots of line gridding and KDE gridding to a fine grid. The input line is shown at an arbitrary height above the cells.

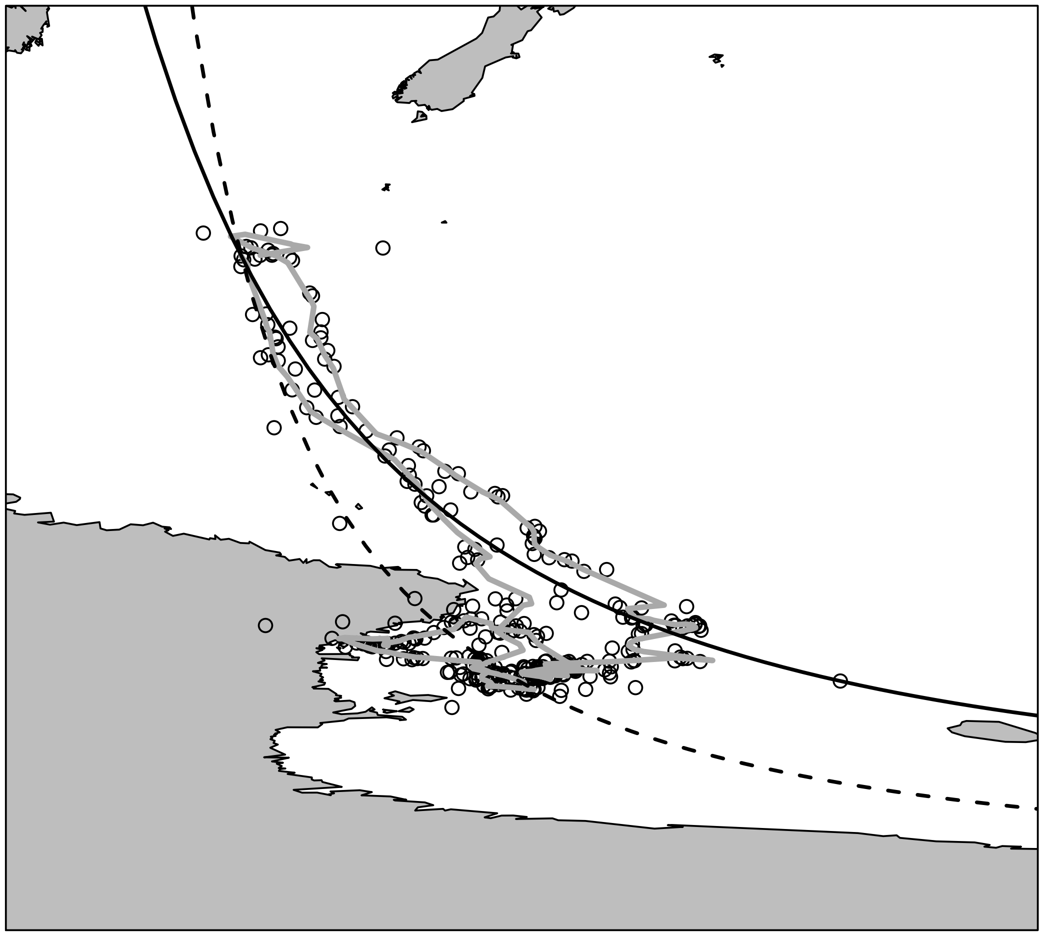


Figure 10: Raw Argos estimates and penalized smoothed track plotted in raw longitude and latitude. Great circles are drawn as curves, from Macquarie Island through the most distant raw Argos position (solid) and the most distant smooth position (dashed).

plots. The same data in Figure 11 uses a projection that more accurately represent great circles as straight-lines.

3.6 Tools for traditional methods

Given these limitations there is still a use for these filtering and gridding methods. They are also used as part of more sophisticated modelling approaches (**??**). The **trip** package provides a framework for running these algorithms on sets of animal tracks within the validation requirements discussed above. As it builds on the existing spatial infrastructure the package provides simple access to arbitrary map projections and to input and output for commonly used GIS vector and raster formats. Extra capability is provided by linkages to the **raster** package to simplify access to imagery, raster data and array formats like NetCDF and to the **spatstat** package

4 The trip package for animal track data

The **trip** package developed by the author provides programming definitions and functions for working with animal track data in the R programming environment. The package provides convenient access to commonly used methods discussed in this chapter. The **trip** package is available on `CRAN`: `http://cran.r-project.org`[8].

The **trip** package builds on the spatial infrastructure of the **sp** package. The **sp** package uses the S4 programming classes and methods of **?** to provide a coherent system for the usual spatial data types: points, lines, polygons and grids. This system provides visualization and manipulation of spatial data, a consistent conversion for data between the variety of spatial statistics packages already in R, and interfaces to GIS and map projections (**?**). The following description of **sp** objects given here is derived from **?** and **?**, which should be consulted for a more complete description.

All **sp** (and therefore `trip`) objects share the basic property `Spatial` which stores only the bounding box and map projection metadata. Each new data type then extends this to the variety of spatial types with increasing specialization: coordinates and levels of organization are added to provide `SpatialPoints`, `SpatialLines`, `SpatialPolygons` and `SpatialGrids`. `Trip` objects are specializations of the `SpatialPoints` class and so the other types are not considered further here.

The class `SpatialPoints` consists of a matrix of coordinates and the `Spatial` metadata for their bounding box and map projection—this is sufficient to locate each point on a map, but applying more information for each point requires a matching attribute table. For example data such as the name, size, colour, direction and quality for each point may be stored as well as its location. The basic table component in R is a `data.frame`, and this is extended by the class `SpatialPointsDataFrame`. A `SpatialPointsDataFrame` can be considered as a table of records (a `data.frame`) containing X and Y coordinates and other attributes. The coordinates are treated specially so that they may be used to visualize or manipulate the object in a spatial context[9]. Objects of class `trip` extend this class directly, by applying another level of information about the identity of each trip, the temporal coordinate of each point and ensuring that the records can be validated by the rules laid out in Section 2.2. This is done with a new class `TimeOrderedRecords` that stores the names of the date-time and ID attribute in the records. Checks are applied for any object aspiring to be of class `Spatial` and of class `TimeOrderedRecords` by ensuring that time values are in order within ID and that there are no duplicate times, no missing coordinates etc. If any of these constraints is broken then validation cannot occur and the object creation will fail. Further discussion and illustration of the `trip` class definition in the context of **sp** is given in **?**.

By using the **sp** classes **trip** automatically gains all the available methods for `Spatial` objects including plot, subset, coordinate system metadata and reprojection, summary and print. A `trip` object is considered as sets of "TimeOrdered" points, rather than multi-segment lines. Some functions do assume line-interpretion of trips, but in general the storage of attribute data on ordered points provides greater flexibility than storage of line objects for reasons discussed in Section 2.3.

[8]**Trip** replaced the experimental package **timeTrack** version 1.1-6, which provided similar functionality but was not integrated with **sp**.

[9]This is rather simple for points since there is only one coordinate for each record: the separation of coordinates and attribute data becomes important for more complicated spatial objects.

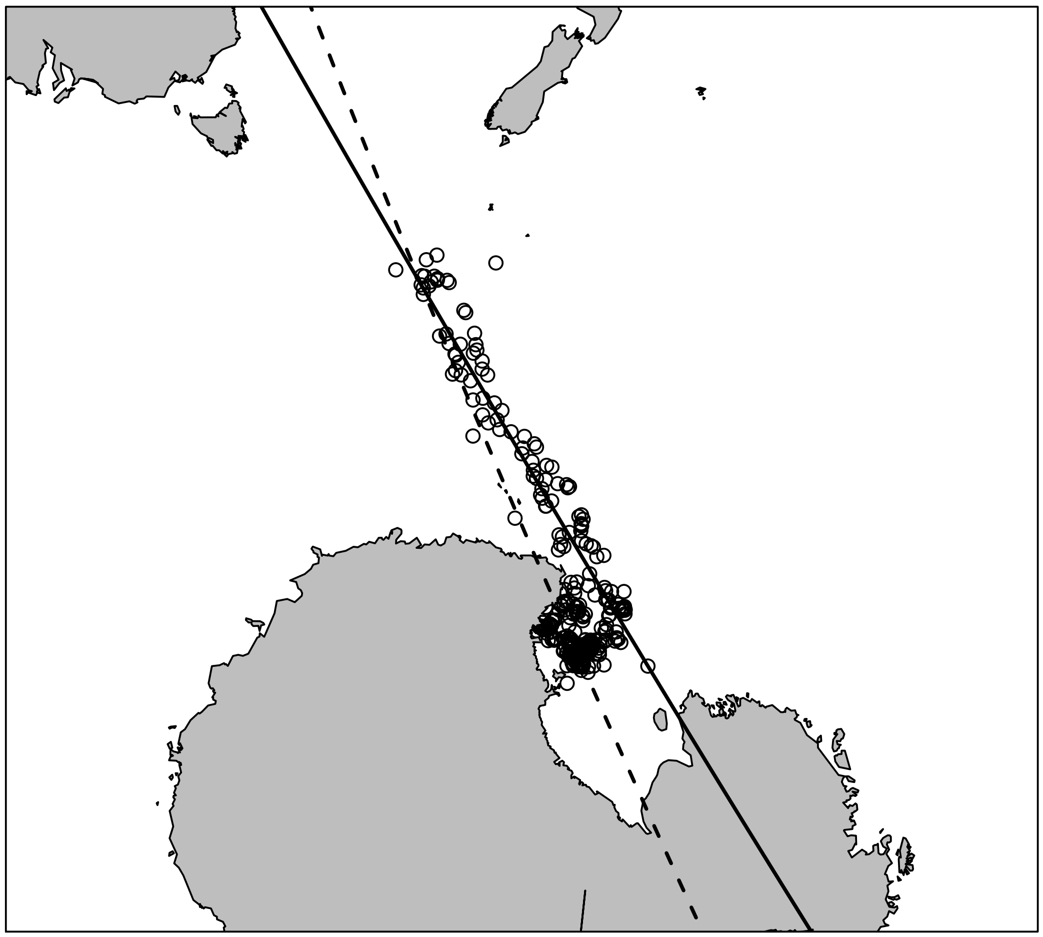


Figure 11: Raw Argos estimates and penalized smoothed track plotted in an equal-area projection. Great circles (as-the-crow-flies) are shown as lines. These pass through the Macquarie Island site to the most distant raw Argos position (solid) and the most distant smooth position (dashed).

The **trip** package depends entirely on the **methods** and **sp** packages, and in part on the **spatstat** and **maptools** packages (**??**).

The next section demonstrates some examples using the **trip** package for importing data, dealing with common problems and "filtering" and "gridding" track data.

4.1 Generating trip objects

The following examples show the simplest means of generating a trip object from a table of data. The function `readArgos` does all of this internally, as shown below. The `trip` class is derived from classes in the package **sp** and so the first step is to create a `SpatialPointsDataFrame`.

The following R code reads data from a text file, promotes resulting table to a Spatial object and converts dates and times from text.

```
## originally this was the readArgos data, but now is this:
## load("E:\\mike\\working\\doc\\PhD\\Thesis-MDSUMNER\\figuredata\\RawArgos.Rdata")

dat <- read.csv("trackfile.csv")
names(dat)  #[1] "long" "lat"  "seal"  "date"    "local"     "lq"

## [1] "long"  "lat"   "seal"  "date"  "local" "lq"

library(sp)
coordinates(dat) <- c("long", "lat")
dat$gmt <- as.POSIXct(strptime(paste(dat$date, dat$local),
                      "%d-%b-%y %H:%M:%S"), tz = "GMT") - 10 * 3600
```

A data frame in R is read in from text file using `read.csv`. The column names are printed by the `names` function. The **sp** function `coordinates` is used to promote a data frame to a `Spatial` object by specifying which columns contain the spatial coordinates.

The dates and times in this object are still just text from the file, so these are converted to the `DateTimeClasses` provided by R. An offset is applied to ensure the correct timezone interpretation. The `strptime` function provides all the required templates for parsing date-times from text—here the date stamp is a parochial format with day, short month and short year.

The examples here use longitude and latitude data, but `trip` objects can be generated using any valid coordinates. The projection metadata may be stored as per the definition of the `Spatial` classes in **sp**.

The following code attempts to generate a `trip` object by assigning the time and ID attributes. This step often presents some problems that need to be dealt with.

```
library(trip)
```

```
tr <- trip(dat, c("gmt", "seal"))
```

```
## Error in cat(try_out):  argument 1 (type 'S4') cannot be handled by 'cat'
```

The `trip` function takes the `SpatialPointsDataFrame` and the names of the date-time and ID columns. In this case the attempt fails as the data will not validate in its current form. This is important as it ensures that these simple problems are dealt with upfront so they are not propagated further to cause problems in subsequent analyses.

The next section shows an alternative method for reading **trip** data from an Argos format.

Reading data from Argos records

Argos (PRV/DAT) files can be read directly using the function `readArgos` and if the defaults for basic quality control are successful this will return a `trip` object.[10]

```
library(trip)
## read in some Argos data
argosfiles <- list.files(path = "G:/DATA/tracks/blackBrowed/", pattern = ".dat", full.names = TRUE, ignc
argosdata <- readArgos(argosfiles[1:3])

## Error in file(con, "r"):  cannot open the connection

summary(argosdata)

## Error in summary(argosdata):  object 'argosdata' not found
```

In Argos PRV (DAT) files the fields `longitude` and `latitude` contain the spatial coordinates (these have been extracted from the other data in the `SpatialPointsDataFrame` in the usual way), `date` and `time` the temporal information (these have been combined into an R date-time column called `gmt`), and `ptt` is the ID for individual instruments that is used as the trip ID. The function `readArgos` will perform some sensible quality control corrections by default. The summary command returns a listing of the individual trips, their ID, start and end times, and number of locations. The remaining data are summarized in the usual way for a **SpatialPointsDataFrame**.

When reading data from PRV files the following coordinate system is assumed:

longitude / latitude on the WGS84 datum.

This is specified using the following PROJ.4 string:

`+proj=longlat +ellps=WGS84`.

If longitude values greater than 180 are present the "+over" element is applied for the "Pacific view [0,360]" longitude convention. No further checking is done.

The function `readDiag` reads Argos DIAG (diagnostic) format that provides two sets of location coordinates. This function returns a data frame with the attributes from the files.

Dealing with common problems in track data

This section begins with the raw data frame of track data from 4.1.

```
dat <- as.data.frame(dat)
```

There are a number of simple problems at this stage that the `trip` class automatically provides validation for. Duplicated rows, such as those from overlapping Argos files, can be safely dropped.

```
dat <- dat[!duplicated(dat), ]
head(dat)

##        long       lat seal      date    local lq                 gmt
## 1 158.9467 -54.49333 b284 21-Jan-99 13:43:51  A 1999-01-21 03:43:51
## 2 155.2533 -54.42833 b284 21-Jan-99 15:18:53  B 1999-01-21 05:18:53
## 3 155.8400 -56.30833 b284 23-Jan-99 16:45:51  B 1999-01-23 06:45:51
## 4 159.3850 -56.10167 b284 23-Jan-99 13:03:35  B 1999-01-23 03:03:35
## 5 159.9650 -58.35333 b284 25-Jan-99 16:19:17  B 1999-01-25 06:19:17
## 6 158.8917 -57.99000 b284 25-Jan-99  4:45:15  B 1999-01-24 18:45:15
```

[10]These data were provided by the DPIWE Macquarie Island Albatross Project (?). Only three of the available Argos files are imported for this example.

The discarded rows removed are exact duplicate rows, matched on every data value for each column.

The ordering of rows in the data is assumed to follow the order of the date-time values with a trip ID. The date-time values are not used automatically to order the data, as this could hide deeper problems in a data set.

```r
dat <- dat[order(dat$seal, dat$gmt), ]
```

A final problem is that subsequent date-times within a single trip ID can be duplicates. Removing duplicate rows and ordering the rows still leaves the problem of what that can mean. If the location coordinates are different, the implication is that the animal moved a certain distance in no time at all. If the locations are the same then it begs the question of why the tracking system distinguishes the records at all. A zero duration time difference results in meaningless metrics of speed of movement, for example. There is no obvious solution for this and the issue must be investigated in the context of the actual data used. A simplistic solution that allows us to move on is to adjust these duplicate times by a very small amount, so that the time difference is not zero.

```r
dat$gmt <- adjust.duplicateTimes(dat$gmt, dat$seal)
```

Further data problems are more fundamental and do not have easy fixes. The `trip` class will fail to validate in the following cases and these must be dealt with individually as appropriate.

Insufficient records for a given trip ID Each set of records must have three or more locations to qualify as a `trip`. Sometimes the only available data is a start and end location but this case is deemed inappropriate for **trip**.

Invalid values for critical data Missing values or otherwise non-numeric values for locations and date-times cannot be included. The **sp** classes ensure this for location coordinates, and `trip` adds the limitation for date-times and IDs.

Non-existent date and ID data This is somewhat obvious, but **trip** will not assume a simple date-time value from the order of records or provide a default ID for single-trip data. These must be explicitly provided.

Once all of the fixes required have been applied, `trip` validation is possible.

```r
coordinates(dat) <- c("long", "lat")
tr <- trip(dat, c("gmt", "seal"))
```

The location quality class is converted to an ordered factor, and the appropriate PROJ.4 string is applied.

```r
tr$class <- ordered(tr$lq, c("Z", "B", "A", "0", "1", "2", "3"))
proj4string(tr) <- CRS("+proj=longlat +ellps=WGS84 +over")
```

The location quality class provided for Argos data is automatically converted to an "ordered factor" by `readArgos`, here shown manually. This can be used for simple selection of a range of records from a data set, even though the tokens used as text have no inherent order. This allows the data to be used directly without creating a numeric proxy for the class values.

The resulting `trip` object read from CSV text is now validated and equivalent to that returned by `readArgos`.

4.2 Filtering for unlikely movement

The infrastructure provided by the **trip** classes allows efficient implementation of custom filters based on a variety of metrics.

The data are from southern elephant seals from Macquarie Island. These animals can swim up to 12 km/hr (**?**) so that value is used to calculate a filter, which is added as a column in the data frame. The filtering algorithm is that of **?**.

```
tr$ok <- speedfilter(tr, max.speed = 12)
summary(tr)

##
## Object of class trip
##   tripID ("seal") No.Records   startTime ("gmt")     endTime ("gmt")  tripDuration
## 1            b284        281 1999-01-21 03:43:51 1999-06-09 04:34:22 139.0351 days
## 2            b290        255 1999-01-29 19:30:59 1999-07-02 22:52:25 154.1399 days
## 3            c026        351 1999-01-01 02:27:28 1999-05-20 18:51:27 139.6833 days
## 4            c993        258 1999-01-07 02:59:53 1999-04-24 03:04:44 107.0034 days
##   tripDistance  meanSpeed    maxSpeed meanRMSspeed maxRMSspeed
## 1     24099.26 488.307626 128448.7448   485.784299  64224.3759
## 2     12405.70   9.039951    209.6750     8.338208    106.7682
## 3     21136.85  25.904843    736.3698    24.210892    417.9929
## 4     28673.24  52.616908   1449.5900    49.126440    725.0411
##
## Total trip duration: 46644047 seconds (12956 hours, 2447 seconds)
##
## Derived from Spatial data:
##
## Object of class SpatialPointsDataFrame
## Coordinates:
##            min       max
## long 128.27167 198.99167
## lat  -78.06333 -52.85667
## Is projected: FALSE
## proj4string : [+proj=longlat +ellps=WGS84 +over]
## Number of points: 1145
## Data attributes:
##    seal             date            local       lq           gmt
##  b284:281   5-Feb-99 :  20   15:29:50:   3   0:114   Min.   :1999-01-01 02:27:28
##  b290:255   10-Mar-99:  17   14:42:21:   2   1: 31   1st Qu.:1999-02-11 06:03:36
##  c026:351   11-Feb-99:  17   14:48:07:   2   2: 18   Median :1999-03-10 19:43:05
##  c993:258   21-Mar-99:  17   16:08:14:   2   3:  1   Mean   :1999-03-15 13:10:03
##             23-Feb-99:  16   16:18:30:   2   A:281   3rd Qu.:1999-04-10 06:18:01
##             10-Apr-99:  15   16:39:55:   2   B:700   Max.   :1999-07-02 22:52:25
##             (Other)  :1043   (Other) :1132
##  class        ok
##  Z:  0   Mode :logical
##  B:700   FALSE:230
##  A:281   TRUE :915
##  0:114
##  1: 31
##  2: 18
##  3:  1
```

The `speedfilter` function assumes that speed is specified in km/hr and distance is calculated based on the projection metadata. If the projection is specified and not longitude and latitude then Euclidean methods are applied. If the projection is not specified longitude and latitude is assumed and ellipsoid methods for WGS84 are used.

The summary shows that a number of locations are now classified by a boolean value in a new "ok" column. Although the speed filter has not removed many locations, a customized subset can be defined based on other data. Using a minimum Argos location quality class of "A" the raw data is plotted with a default point symbol and reduced size, with lines connecting the remaining filtered points. The plot is shown in figure 12.

If this were calculated with coordinates in a different unit or projection the `speedfilter` function takes this into account and calculates distance accordingly. This integration of spatial metadata and calculation provides for a very flexible working environment for exploratory analysis.

Extending the trip package

One of the advantages of the infrastructure provided by the trip classes is that simple tasks can be strung together efficiently. This example shows how the speed-distance-angle filter of **?** can be built up from the basic tool kit.

The speed distance angle filter is distributed with an easily run example, as follows. This is reproduced almost exactly from the documentation for `argosfilter` in **?**.

```
library(argosfilter)
data(seal)
lat <- seal$lat
lon <- seal$lon
dtime <- seal$dtime
lc <- seal$lc
cfilter <- sdafilter(lat, lon, dtime, lc)

seal$sda <- !(cfilter == "removed")
```

This filter results in a column of values that specify whether the filter retains or discards the location. The following steps create a new version of this filter using the **trip** package. Load the example data and validate as a single event trip object.

```
library(argosfilter)
library(sp)
library(trip)
library(maptools)

trackAngle <- function(xy) {
    angles <- abs(c(trackAzimuth(xy), 0) -
                  c(0, rev(trackAzimuth(xy[nrow(xy):1, ]))))
    angles <- ifelse(angles > 180, 360 - angles, angles)
    angles[is.na(angles)] <- 180
    angles
}


## default values used by sdafilter
vmax <-  2
ang <- c(15, 25)
distlim <- c(2500,5000)


coordinates(seal) <- ~lon+lat
proj4string(seal) <- CRS("+proj=longlat +ellps=WGS84")
seal$id <- "seal"
```

Perform some simple sanity checks before proceeding.

```
plot(tr, axes = TRUE, cex = 0.4)
plot(world, add = TRUE, col = "grey")
lines(tr[tr$ok & tr$class > "B", ], lwd = 2,
      col = bpy.colors()[seq(10, 90, length = 4)])
```

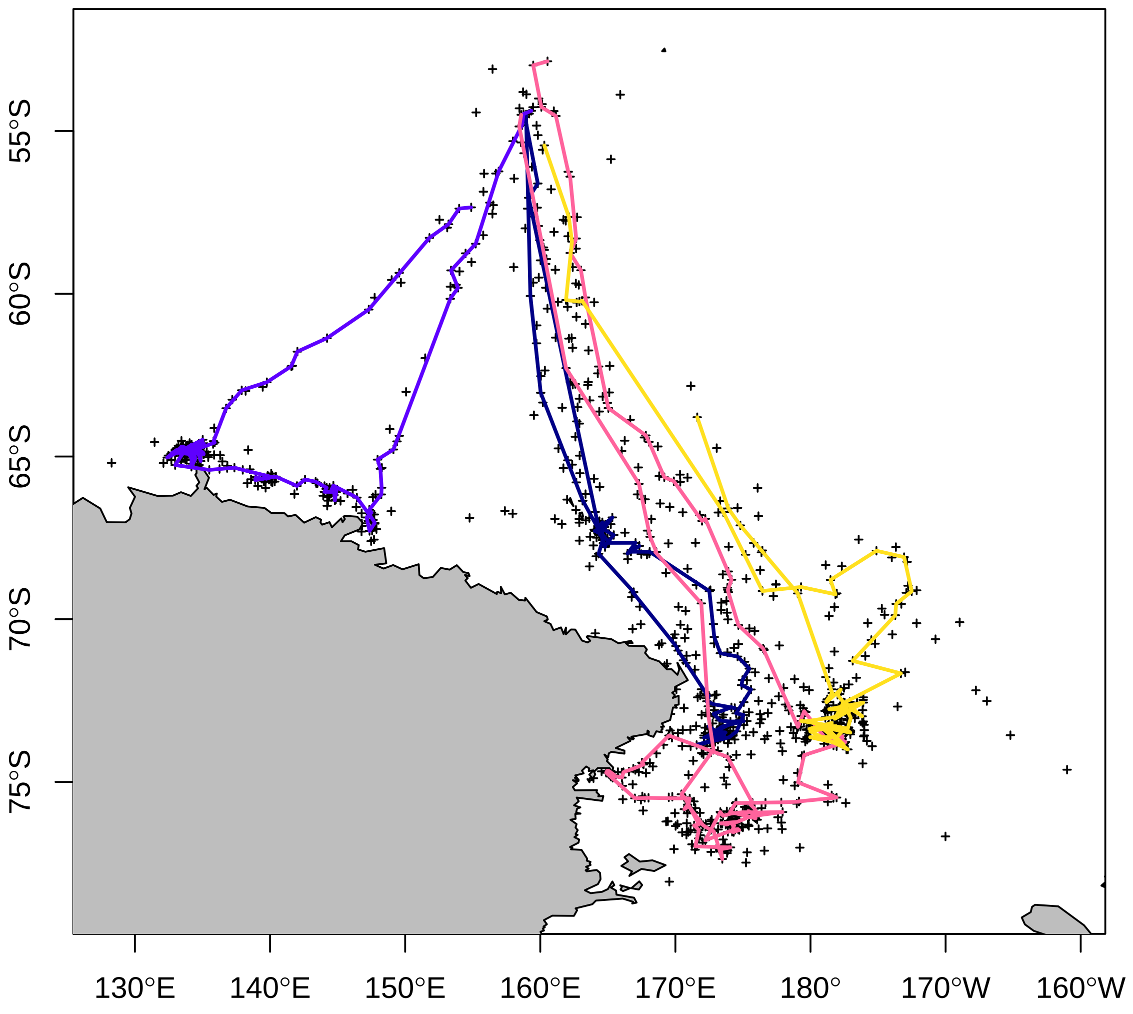


Figure 12: Plot of trips as points, with lines coloured for each separate trip.

```
range(diff(seal$dtime))

## Time differences in secs
## [1]      0 362737

which(duplicated(seal$dtime))

## [1]   18  117  123 1009 1159 1232 1294 1301
```

There are some preliminary requirements to load packages and functions for earth distances and track turning angles. The `seal` object is a data frame with columns "lon", "lat", "dtime" and "lc". The date-time values are already in `POSIXct` format, and lc uses a numeric code for the Argos class.

The records are in order of date-time, but some of the date-time values are duplicated, so for illustration these are dropped and then a new `trip` object is created.

```
seal <- seal[!duplicated(seal$dtime), ]

seal.tr <- trip(seal, c("dtime", "id"))
```

First, add the filter from the `sdafilter` function to the trip object, and also the basic speed filter available in the **trip** package (using km/hr rather than m/s).

```
seal.tr$speed.ok <- speedfilter(seal.tr, max.speed = vmax * 3.6)
```

Perform the initial simple processing for the speed distance angle filter.

```
## distances in metres
dsts <- trackDistance(coordinates(seal.tr)) * 1000
angs <- trackAngle(coordinates(seal.tr))

dprev <- c(0, dsts)
dnext <- c(dsts, 0)

## speed, lc, max distance
ok <- (seal.tr$speed.ok | dprev <= 5000) &  (seal.tr$lc > -9)
```

This creates a logical vector where the speedfilter and distance-previous pass, and discard any location class of "Z" (coded as "-9" in this example). This is effectively the first pass of the filter, distinguishing distance previous and distance next for each point.

Now the remaining parts of the filter can be run—testing for angle and distance combinations over the specified limit. Some housekeeping—create an index to keep track while running the filter on the two distance/angle limits. A temporary copy of the `trip` is used to make matching the filter easy as points are removed.

```
seal.tr$filt.row <- 1:nrow(seal.tr)
seal.tr$ok <- rep(FALSE, nrow(seal.tr))

df <- seal.tr

## first subset
df <- df[ok, ]
## distlim and angles, progressively
for (i in 1:length(distlim)) {
    dsts <- trackDistance(coordinates(df)) * 1000
    angs <- trackAngle(coordinates(df))
    dprev <- c(0, dsts)
    dnext <- c(dsts, 0)
    ok <- (dprev <= distlim[i] | dnext <= distlim[i])  | angs > ang[i]
    ok[c(1:2, (length(ok)-1):length(ok))] <- TRUE
    df <- df[ok, ]
    ok <- rep(TRUE, nrow(df))
}
```

The result is now a reduced `trip` with missing rows discarded by the filter. Using the row index created earlier, match the filter result to the original rows and tabulate the filter values for the original sdafilter and the trip version. The number of accepted points is nearly the same.

```
seal.tr$ok[ match(df$filt.row, seal.tr$filt.row)] <- ok

sum(seal.tr$sda)

## [1] 1135

table(seal.tr$sda, seal.tr$lc)

##
##          -9  -2  -1   0   1   2   3
##   FALSE  26 302  48  17  17   6   2
##   TRUE    0 374 374  61 150 116  60

sum(seal.tr$ok)

## [1] 694

table(seal.tr$ok, seal.tr$lc)

##
##          -9  -2  -1   0   1   2   3
##   FALSE  26 482 185  38  65  46  17
##   TRUE    0 194 237  40 102  76  45
```

Plot the result to see that the new **trip** version is comparable. There are differences since the distance and angle calculations are ellipsoid based, rather than spherical as used by **argosfilter** and some locations were first discarded due to impossible duplicate times in the original data. There are probably also differences in the detail of the recursive speed filter and the way that peaks are assessed.

The examples above consist of a working prototype that can be wrapped as a function for the **trip** package to efficiently apply this filter to sets of tracks within a `trip` object. The **argosfilter** example above takes at least three times as long to complete as the code developed above. The ability to extend the functionality in this way mirrors the development of the **trip** package itself, from the **sp** package, and in turn the R platform.

4.3 Creating maps of time spent

Assuming that the filtered locations give realistic information about position for the animal and that motion between these positions is constant and straight, a map of time spent, or residency can be created. The choice of grid cell size might reflect the confidence in the accuracy of the location data, or require a specific cell size for comparison with another data set.

Using the trip locations accepted by the speed filter attribute generate a grid of time spent, shown in Figure 14.

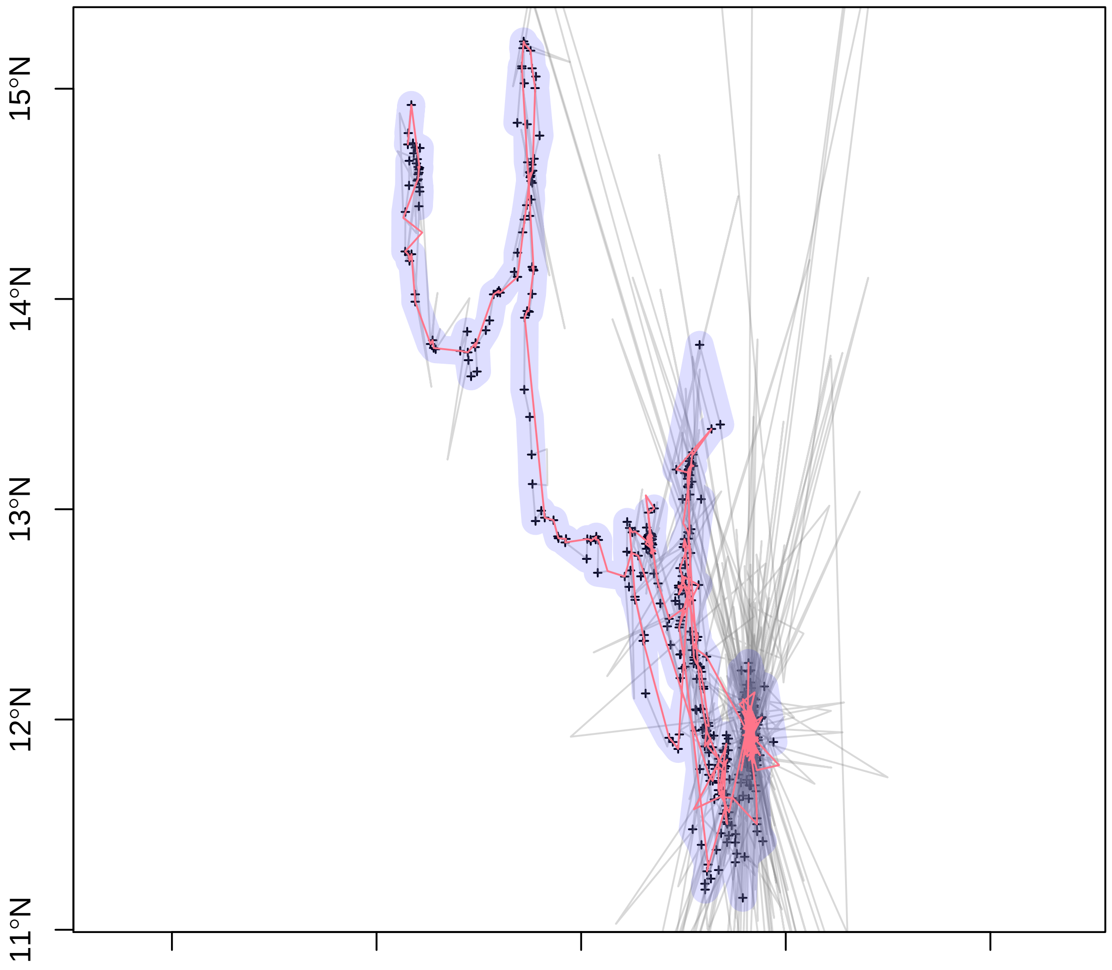


Figure 13: Original `sdafilter` (red) and custom **trip** speed distance angle filter (thick grey line). The original raw track is shown as a thin grey line.

```r
trg  <- tripGrid(tr[tr$ok, ])
image(trg, col = oc.colors(100), axes = TRUE)
```

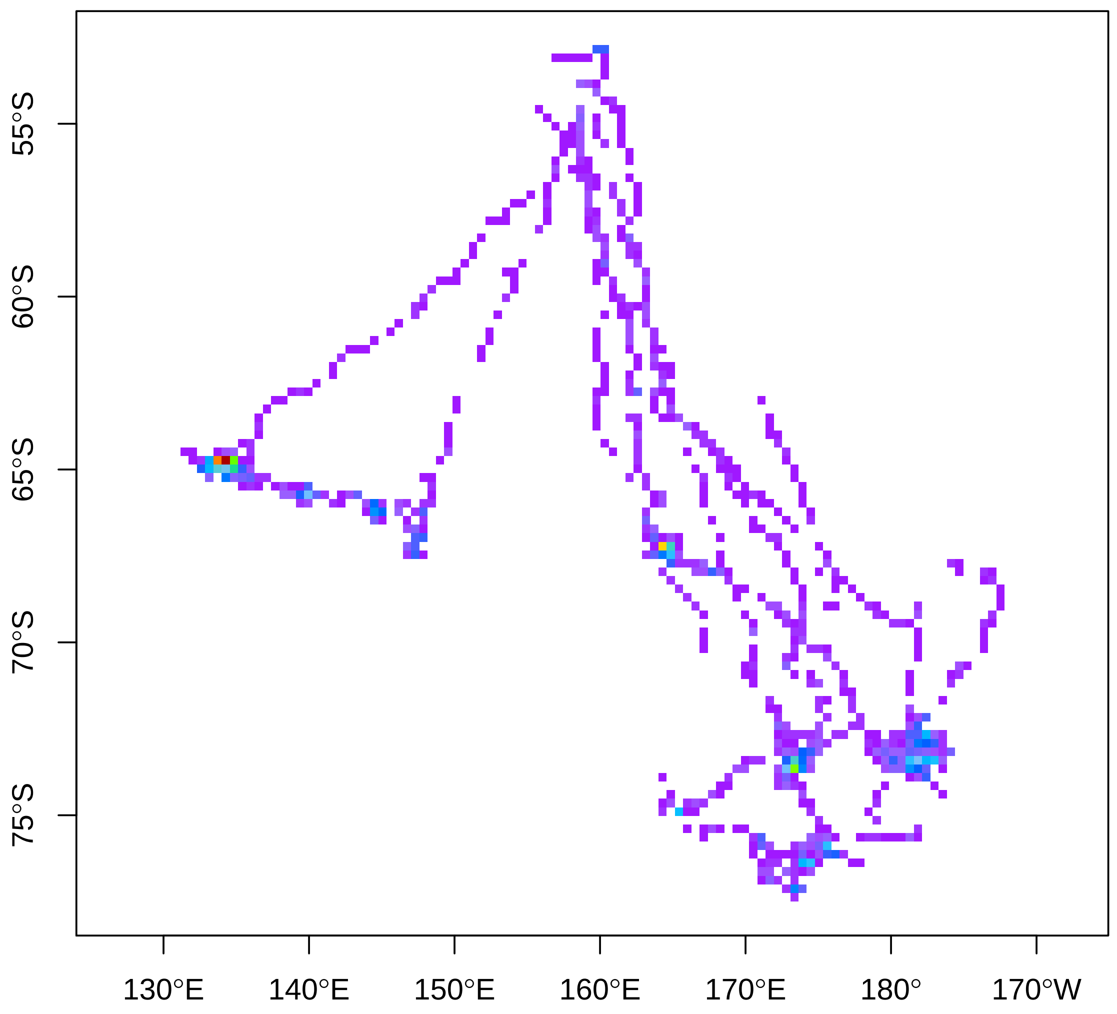


Figure 14: Simple image of a time spent grid for the trip object.

The function `tripGrid` is used with the subset of the trip object accepted by the speed filter to create a grid of time spent. The algorithm used cuts the line segments exactly as described in section 3.3 by the **spatstat** function `pixellate`. If any zero-length line segments are present a warning is issued and their time contribution is included as points. This is because the pixellate algorithm relies on weightings that are relative to the line length, which is another example of the subtle implications of point versus line interpretations.

The gridded object `trg` is a `SpatialGridDataFrame`, whose class definition is provided by the `sp` package. These grids support multiple attributes and so, conveniently, variations on the map created can be stored in a single object.

Add three new attributes to the grid object by isolating a single animal's trip, and using the kernel density method with two different sigma values.

```r
names(trg) <- 'grid.filtered'
gt <- getGridTopology(trg)
trg$grid.c026  <- tripGrid(tr[tr$ok & tr$seal == "c026", ], grid = gt)$z
trg$kde1.filtered <- tripGrid(tr[tr$ok, ], grid = gt, method = "density", sigma = 0.1)$z
trg$kde3.filtered <- tripGrid(tr[tr$ok, ], grid = gt, method = "density", sigma = 0.3)$z

for (col.name in names(trg)) trg[[col.name]] <- trg[[col.name]]/3600
```

The default name for an attribute from `tripGrid` is "z" so first this is renamed to "grid.filtered". Calculation for the extra attributes requires that they share the same origin and scale so this is stored in the `GridTopology` object `gt` and used again. The `trip` object is subset on the speed filter attribute and the seal ID, and the resulting grid attribute "z" is extracted and assigned to the grid object in one step. For the kernel density versions two values of sigma are passed onto the `density` function for each grid. The default output is in seconds, so this is converted to hours for each grid column by division. Finally, the multi-panel `spplot` function in package `sp` provides a conveniently scaled image for each of the four grids shown in Figure 15.

By default, `tripGrid` will provide a grid with dimensions 100x100 cells. This can be controlled exactly using a `GridTopology` passed in as the grid argument to the function. The convenience function `makeGridTopology` allows the user to define a specific grid from the trip object itself.

The next examples use `trip` object to create grids with a different scale.

```r
require(lattice)
library(trip)
trellis.par.set("regions", list(col=oc.colors(256)))
print( spplot(trg))
```

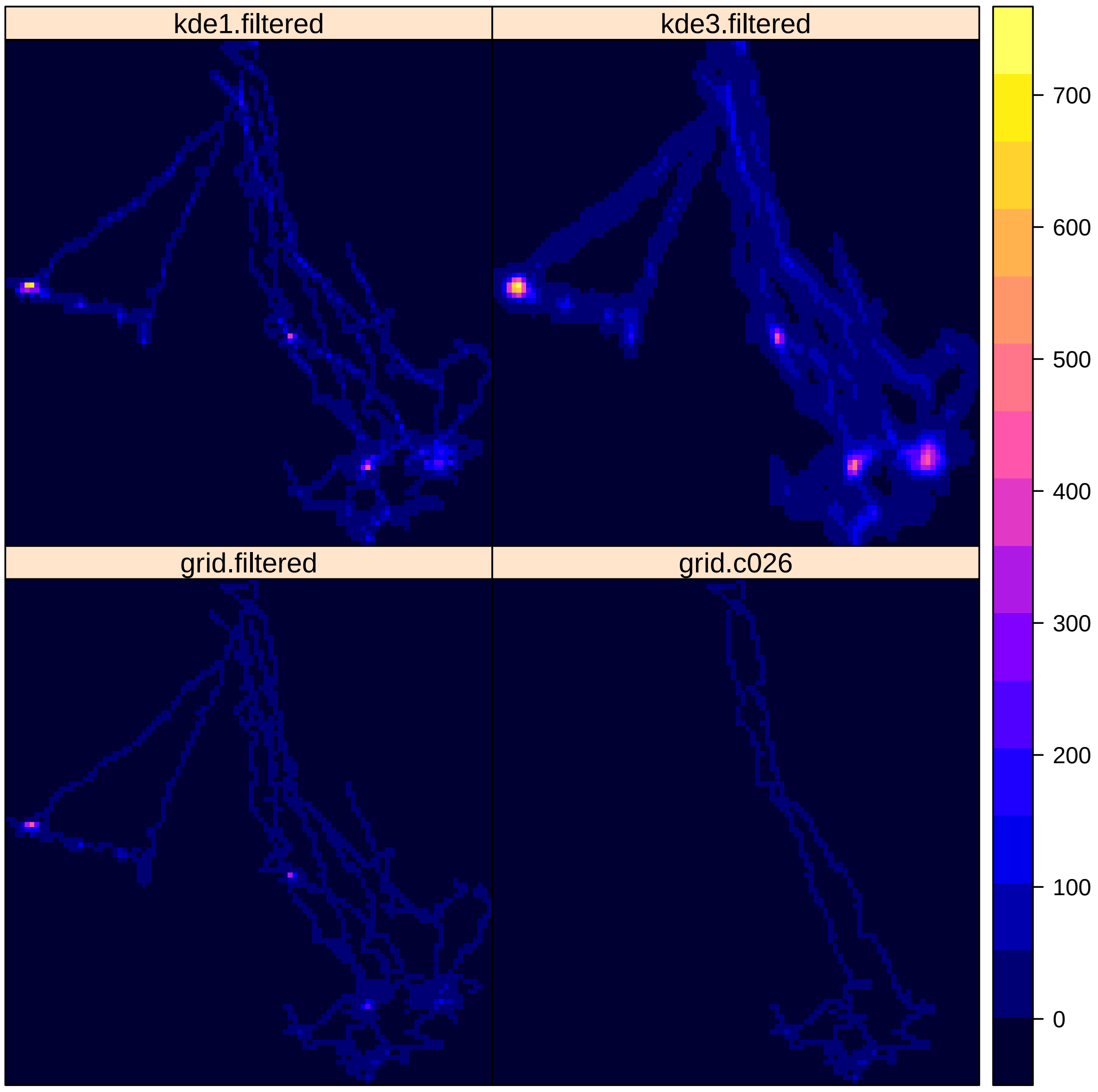
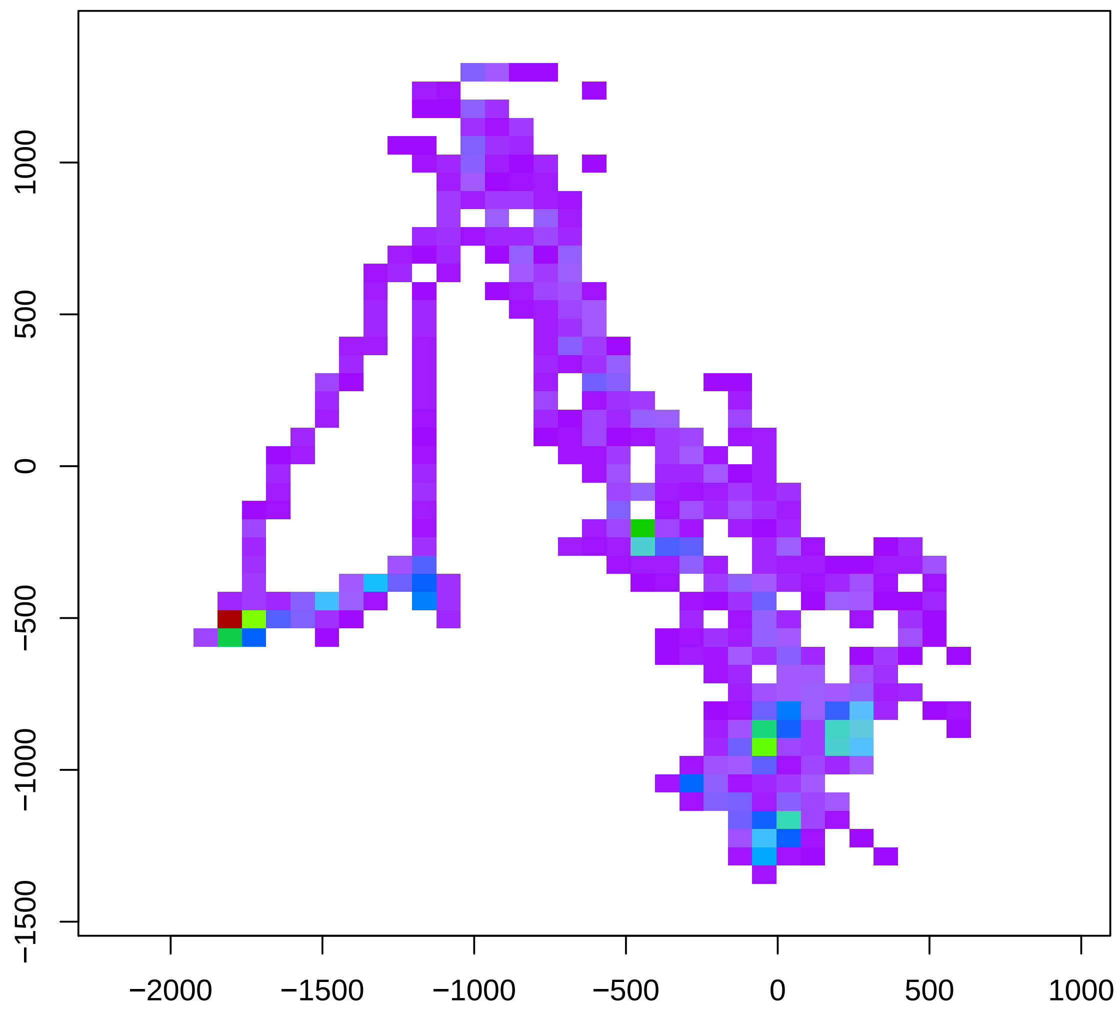


Figure 15: Four variations on a tripGrid. `grid.filtered` and `grid.c026` use the simple line-in-cell method for all four seals and for seal c026 alone. `kde3.filtered` and `kde1.filtered` use the kernel density method for line segments with a sigma of 0.3 and 0.1 respectively. Each grid has dimensions 100x100 and time spent is presented in hours.



The first example shows the creation of a grid topology with dimensions of 50x50, then another is created using cell size. The resulting plot from this coarser version in an equal area map projection is shown in Figure 16. The grid generation will assume kilometres for cell size as at the centre of the grid for longitude and latitude coordinates.

For approximate methods `tripGrid.interp` will interpolate between positions based on a specified time duration. A shorter period will result in a closer approximation to the total time spent, but will take longer to complete. The approximate method is similar to that published by **?** and **?**.

5 The need for a more general framework

Previous sections presented tools for a more systematic handling of animal track data and a variety of methods for improving track estimates and deriving spatial summaries such as time spent. Some of the problems with animal tracking are relatively simple and have reasonable solutions. The traditional methods above illustrate ways of dealing with them in an extensible software toolkit. Most of these

```
proj4string(tr) <- CRS("+proj=longlat +ellps=WGS84")
p4 <- CRS("+proj=laea +lon_0=174.5 +lat_0=-65.5 +units=km")

ptr <- tripTransform(tr, p4)

gt <- makeGridTopology(ptr, c(50, 50))
gt1 <- makeGridTopology(ptr, cellsize = c(80, 60))

grd2 <- tripGrid(ptr, grid = gt1)
image(grd2, col = oc.colors(256), axes = TRUE)


library(maptools)
data(wrld_simpl)

plot(spTransform(wrld_simpl, p4), add = TRUE, col = "grey")
```

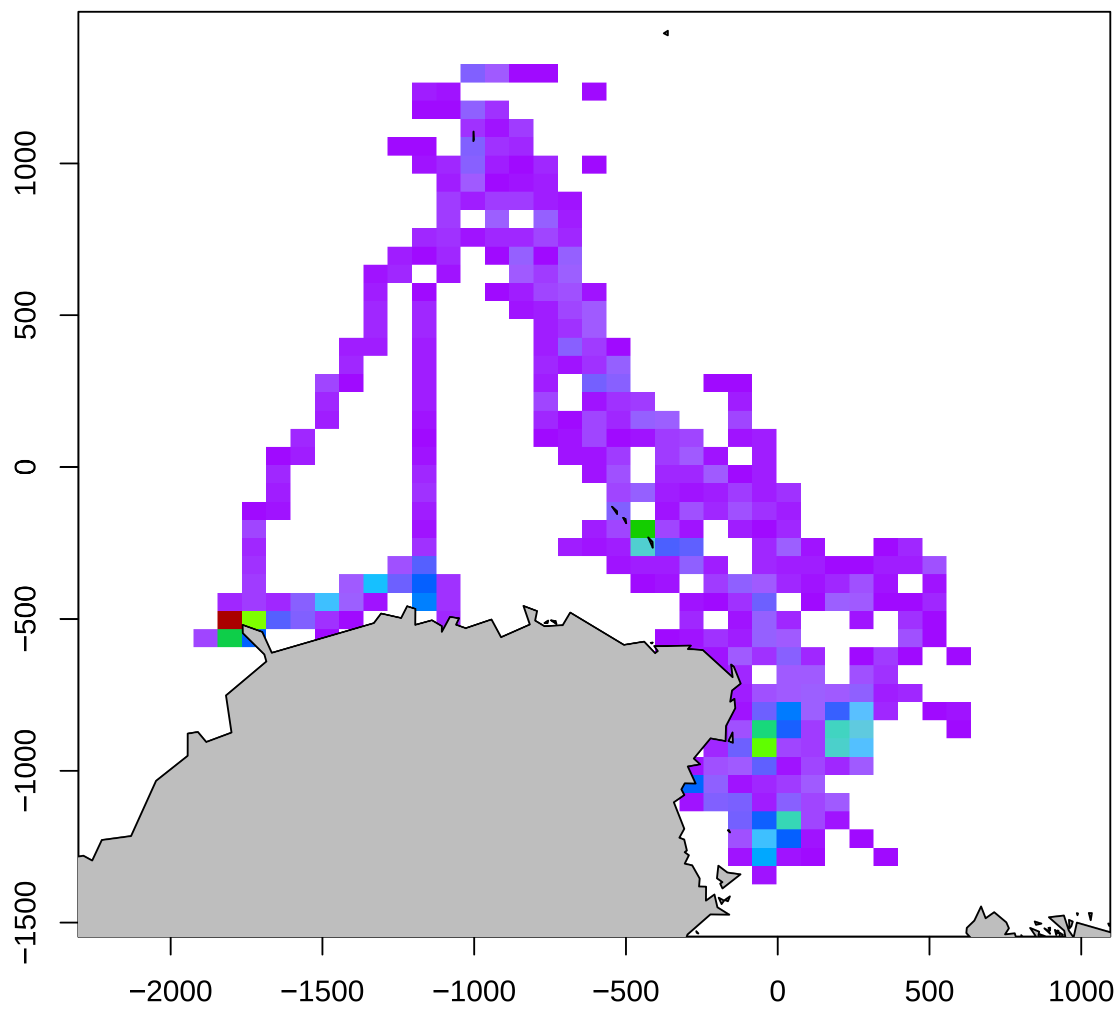


Figure 16: A relatively coarse version of `fourgrids` with a specific kilometre-based cell size (80x60km), specified within an equal area map projection.

solutions however are just "single-targets"—the easiest aspects are cherry-picked for a first-pass answer that solves a small aspect of the larger problem.

Location estimates from methods such as archival tagging and acoustic tagging can also be dealt with in this way, and there are many studies that apply these techniques as well as more sophisticated models. There is an important opportunity here though since the normal "data product" for archival tags is not location estimates over time but dense temporal records of environmental values, such as light, temperature and depth.

Importantly, different satellite methods such as the Argos Service and GPS will be handled uniformly by a general approach. These methods ultimately rely on data as raw as archival tags and acoustic arrays, with doppler shifts or ephemeris data used to determine location. The practical difference of these methods from those of archival tags is that the raw data are simply not available for research purposes, for a variety of reasons.

Another great opportunity presented by raw data is that the concept of integration of all data sources comes very naturally. For example, determining position by light level is plagued by environmental conditions that attenuate the ambient light, and the movement of animals complicates the relation of environmental measurements to independent data sets. The scale of measurement is another issue when relating values such as dive depth or water temperature to synoptic data sets.

To turn attention to "raw data" methods such as those required for archival tags, the aim is to provide a more complete solution that integrates solar information, environmental values such as temperature and depth and applies constraints on movement. This approach contrasts with the filters in this chapter by applying as much information to each location estimate as possible. Also it aims to prevent the practice of discarding "bad" location estimates data—the influence of unreliable data should be downplayed but not ignored completely.

There are of course many existing location estimates for which there is no rawer data. From this perspective an Argos location can be treated as a mere data point—not an absolute position to be retained or discarded and then smoothed by some mechanism, but simply a piece of data to help inform our models. Even completely invalid positions have a date-time value and so at the very least it can be inferred that the animal was "visible" to the remote sensing system. In conjunction with other data about the environment this tiny piece of data can be valuable.

The following describes the problems specific to the two types of tags that originally motivated this work. By considering the location estimates that come from archival tag methods we see that dealing with the points without reference to the other available information is not going to be good enough.

5.1 Location estimation from raw archival data

Earlier methods of light level geo-location rely on the determination of critical times during the day from the sequence of raw light levels. Longitude is determined directly from the time of local noon which, based on the symmetry between dawn and dusk, can be easily measured. Latitude is determined by the length of the day, measured by choosing representative times such as dawn or dusk that are distinctive in the light record. While this is a very simplified explanation given the range of traditional methods the thrust of the argument is basically correct, see Chapter **??** and **?** for more detail.

In effect, this approach reduces the data set of light values to a more abstract summary with local peaks at noon and inflection points at dawn and dusk. The majority of the light data is not used directly and only one location can be determined per day. This approach to estimation is susceptible to the movement of the animal between twilights, to poor choices for the critical times, and to day length at equinox periods. **?**, **?** and **?** review these methods and provide quantifications for their accuracy.

There is an opportunity for delving more deeply into the raw data available for location estimation provided by archival tags. Considering the problems for quality testing of derived locations, studies can utilize the raw data from the archival tag and provide an estimation that takes into account more information such as the conditions at twilight, diving depth, water temperature, and movement in an integrated way. The raw data is not just a single point estimate to be discarded or corrected, but a sequence of light, a sequence of temperature and a sequence of depths. These are all linked temporally within the main data set. In theory, this is no different for satellite tags except in that case the raw data are not available. Raw data in this case would be the doppler shift signals and satellite orbital details for Argos, and satellite ephemeris and signal timings for GPS. Acoustic tagging applications similarly have a wealth of raw data for informing locations, and potential for improving on existing techniques.

Figure 17 presents another example of the "obvious" problem, with archival tracks that are very erratic and have some estimates well inland. The sorts of inaccuracies shown tend to be easily understood by researchers, but the public perception of research, so important to biological programs, can be easily undermined by figures like this. This figure also highlights the need for uncertainty in estimates to be integrated and represented as part of track visualizations and other summaries.

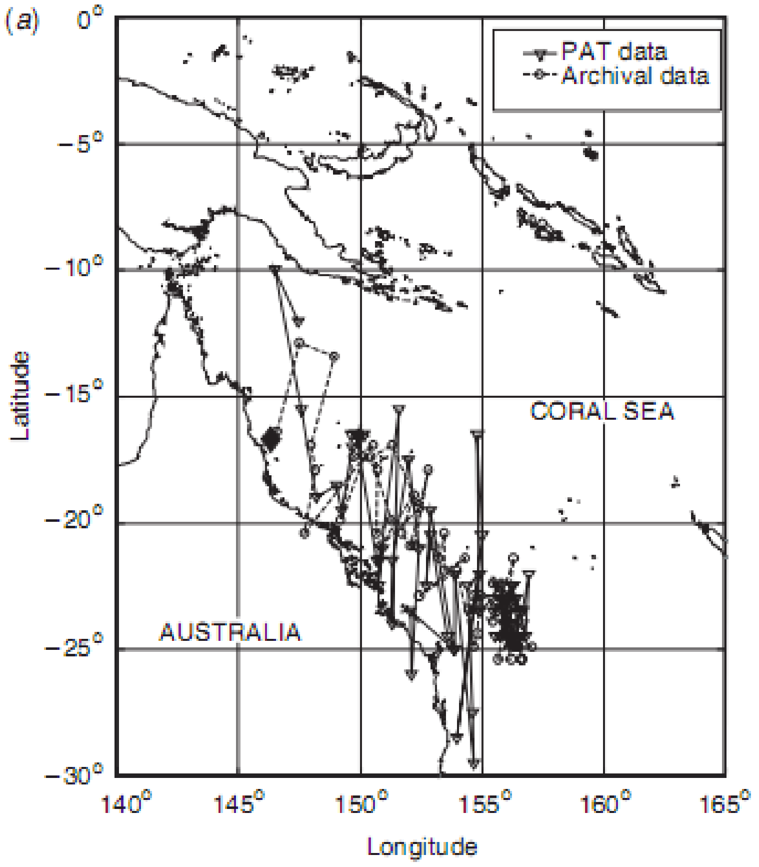


Figure 17: Light level geo-locations for black marlin in the Coral Sea. This image is taken from Figure 3a in **?** and is reproduced with the permission of the authors.

5.2 Filtering locations

Destructive filters that remove "erroneous" locations are susceptible to the following problems. Data is lost, and this may be significant for a diving animal or other situation where the opportunity for obtaining fixes is rare as was shown earlier in Section 3.1. If nothing else, an "erroneous" location at least has useful information about the time at which a fix was available. In practice techniques tend to smooth out some filter metric across multiple locations, with no clear reason to choose one technique over another. Speed, distance and direction all require at least two locations for their calculation, and so this forces the consideration of a point or line interpretation, or perhaps a hybrid of the two. The removal of a data point changes the relevant filter decision for its neighbours and so begs the question of whether the first location removed is not more valid, and perhaps better than others that should be removed in preference.

An example of this can be seen with Argos diagnostic (DIAG) data in which there are paired solutions for each locations (**?**). Usually the choice is obvious, but for many positions it depends on the choice made for neighbouring locations. Updating a location to an estimate nearer its neighbours based on some constraint may be more sensible as in Section 3.2, but there is still only a point estimate. Preferably there should be a probability distribution for the location, something that gives an indication of "how correct".

5.3 Intermediate locations

There is a need for modelling the location of the animal at times that are not accompanied by data. As discussed in Section 3.3 the goals of cell gridding and density analysis attempt to integrate this with location uncertainty, but there are limits to the use of tracks represented by simple points and lines. In Chapter **??** an approach that distinguishes "primary" locations from "intermediate" locations is presented. The importance of this for track representations is illustrated in greater detail in Chapter **??**.

6 Conclusion

This chapter has presented traditional solutions to many of the issues faced by tracking analyses with a readily available and extensible software toolkit. The **trip** package developed by the author provides an integrated environment for applying traditional algorithms in the context of widely used spatial data tools. Seemingly isolated problems with track data were shown to be inter-related in complicated ways that preclude an otherwise simple chaining together of individual solutions. The scope of these problems can be extended to include archival tag data and raw methods like light level geo-location. This highlights the need for treating any data, even ostensibly accurate location estimates, as only part of a larger suite of available information. The next chapter provides an integrated statistical approach to modelling location using these disparate types of tag data.