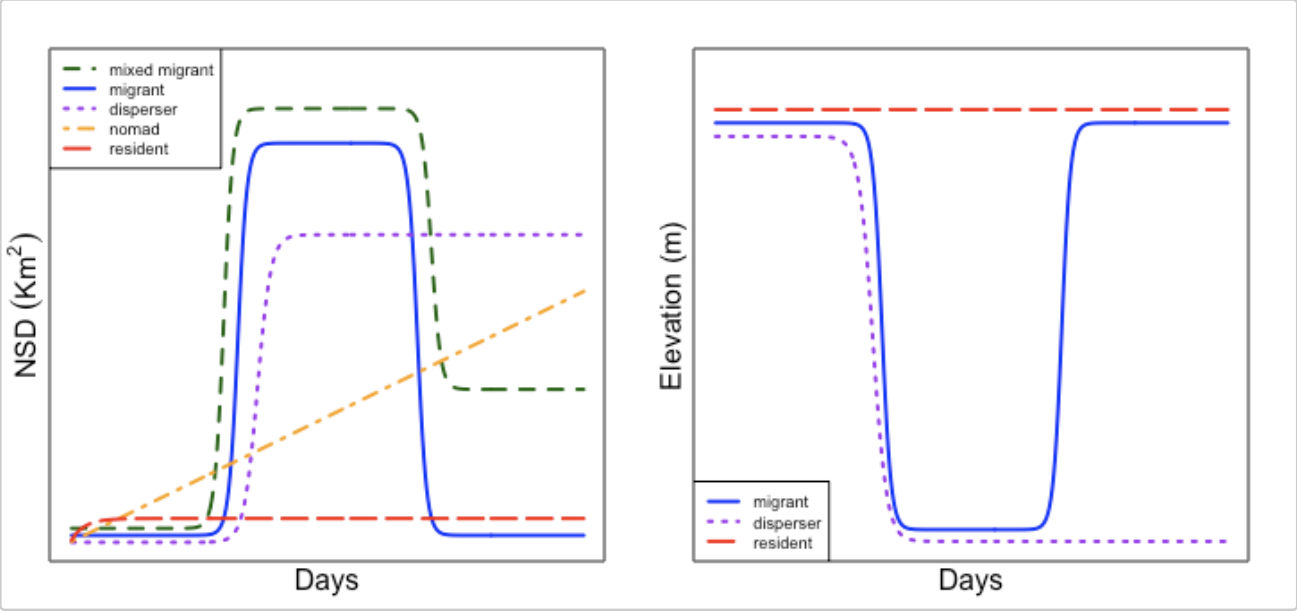
Analysis of Animal Migration in R: the migrateR Package

Derek Spitz

2016-07-06



Conceptual illustration of two families of movement models fit to Net Squared Displacement (NSD) and elevation, respectively. Elevation models can represent upward or downward movement and so can be seen as above or rotated 180^o around the x-axis. The structure of each model is described fully under "Models of Animal Movement", below.

Contents

---

1. Introduction

The following vignette is structured to illustrate the functions and workflow of the `migrateR` package. This package is designed to improve model-driven classification and quantification of animal movement by:

- Refining existing movement models
- Adapting these models to allow treatment of vertical (i.e. altitudinal and diel) migration
- Simplifying the application and evaluation of these methods

This vignette does not, however, include a full explication of model-driven movement classification, instead assuming that this is a topic with which you are already familiar. If this is your first time encountering this approach or if the underlying concepts remain unclear, we strongly recommend you start by reading (or reviewing) Bunnefeld et al. (2011; hereafter "Bunnefeld et al.") before continuing to the rest of this vignette. A brief description of these models is also included at the end of this section (Models of Animal Movement).

a. Data Formatting

The first challenge in using this package is appropriately formatting your data. `migrateR` relies on the `ltraj` format developed by Clement Calenge in the R package adehabitatLT. For help with organizing your movement data into the `ltraj` format, see the very thorough adehabitatLT vignette. For correct performance, `ltraj` coordinates must be recorded using a Universal Transverse Mercator (UTM) system. To fit elevation models, an `ltraj` must also include an `infolocs` field named `elev` containing measurements of vertical distance in meters.

There are two example `ltraj` data sets included in `migrateR` for your reference and use in worked examples. For those interested in long-distance migration, we recommend starting with the data set `elk`. The `bighorn` data set is better suited to examining altitudinal migration. Throughout this vignette we will largely rely on `bighorn`, because its small size renders it more convenient for examples.

Let's start by taking a look at the `bighorn` data set:

```
require(migrateR)
data(bighorn)
bighorn
```

```
##
## *********** List of class ltraj ***********
##
## Type of the traject: Type II (time recorded)
## Irregular traject. Variable time lag between two locs
##
## Characteristics of the bursts:
##     id     burst nb.reloc NAs          date.begin            date.end
## 1 s110 s110 2007      364   0 2007-10-31 16:00:00 2008-10-29 16:00:00
## 2 s110 s110 2008      346   0 2008-10-30 16:01:00 2009-10-12 16:01:00
## 3 s110 s110 2009      364   0 2009-11-01 16:00:00 2010-10-30 16:00:00
```

```
## 4 s110 s110 2010      210   0 2010-10-31 16:00:00 2011-05-28 12:00:00
##
##
##  infolocs provided. The following variables are available:
## [1] "elev"
```

Each `ltraj` object consists of a number of trajectories (identified by `id`, here the individual) organized into discrete groups called `bursts`. The `bighorn` data set, above, consists of four such bursts, each representing a different year (in this case, all four bursts happen to consist of data collected from a single ewe– "s110"). This data set also contains a single `infolocs` field, "elev". We recommend dividing movement data into bursts by study year. Bursts longer than one year are more likely to include more than one movement event (e.g. multiple migrations), making movement models challenging to fit. Bursts much shorter than a year (or with gaps of missing data) can cause similar difficulties and should be checked carefully. Care in checking and organizing movement data pays major dividends in saved frustration with model fitting.

Following Bunnefeld et al., we also recommend subsetting location data to one location per day before attempting to fit movement models. Both of `migrateR`'s example data sets follow this suggestion (i.e. were sub-sampled from larger initial data sets). While subsetting location data is not strictly necessary, reducing the number of points included in each burst will reduce run time and often has the ancillary benefit of improving model convergence.

b. Models of Animal Movement

MigrateR currently implements model fitting for two families of movement models, based on Net Squared Displacement (NSD; the squared distance from a trajectory's first point to each subsequent location) and elevation, respectively. We provide the specification of these models below. NSD models are adapted from those introduced by Bunnefeld et al.. Where possible elevation models follow a parallel structure to allow direct comparison and equivalent interpretation. For further details see Spitz et al. (2015) MigrateR is structured to allow easy expansion to include other model families, but these have not yet been implemented.

NSD Models

Mixed Migrant:

$$NSD = \frac{\delta}{1 + e^{(\frac{\theta - t}{\phi})}} - \frac{\delta * \zeta}{1 + e^{(\frac{2(\phi + \phi_2) + \theta + \rho - t}{\phi})}}$$

Migrant:

$$NSD = \frac{\delta}{1 + e^{(\frac{\theta - t}{\phi})}} - \frac{\delta}{1 + e^{(\frac{\theta + 2*\phi + 2*\phi_2 + \rho - t}{\phi})}}$$

Disperser:

$$NSD = \frac{\delta}{1 + e^{(\frac{\theta - t}{\phi})}}$$

Nomad:

$$NSD = \beta * t$$

Resident:

$$NSD = \gamma * (1 - e^{(\kappa * t)})$$

Elevation Models

Migrant:

$$elevation = \gamma - \frac{\delta}{1 + e^{(\frac{\theta - t}{\phi})}} + \frac{\delta}{1 + e^{(\frac{\theta + 2*\phi + 2*\phi_2 + \rho - t}{\phi})}}$$

One way (Disperser):

$$elevation = \gamma - \frac{\delta}{1 + e^{(\frac{\theta - t}{\phi})}}$$

Resident:

$$elevation = \gamma$$

Explanation of Terms

Parameter	Units	Interpretation
t	days	time since first location (or specified "`stdt`")
θ	days	midpoint of departing movement
θ_2	days	midpoint of returning movement (derived)
ϕ	days	time to complete 1/2 to 3/4 of departing movement
ϕ_2	days	time to complete 1/2 to 3/4 of returning movement
ρ^*	days	duration of occupancy on second range
γ	km^2 / m	mean NSD/elevation of resident range
δ	km^2 / m	distance separating the first and second range
ζ^*	%	difference in distance separating second and third range
δ_2	km^2	distance separating second and third range (derived)
β	km^2 per day	rate of dispersion
κ	log(km^2 per day)	logarithm of the rate constant

Asterisked terms represent revisions to the initial models, which we have implemented to improve model convergence and/or interpretation. We made these revisions such that the interpretation of all remaining terms remain unchanged. Estimates for the two terms we replaced (δ_2 & θ_2) can still be derived, (see Derived Estimates, below).

---

2. Fitting Movement Models

The workhorse of `migrateR` is the function `mvmtClass`, which fits a family of movement models to each burst in an `ltraj` object. This output from `mvmtClass` is organized in a list of class `mvmts`, which, like `ltraj` objects, contains a named element for each `burst`. Each of these element in a `mvmts` object is of class `mvmt` and contains the movement models fit to the burst, the parameter constraints used in fitting these models and the data to which the models were fit (for more on `mvmt` class objects, see Class "mvmt",below). The other functions in `migrateR` are designed to help visualize, organize and interpret the information stored in `mvmt(s)` objects.

Bursts can be fit singly, e.g. with `mvmtClass(bighorn[1])`, but can more simply be fit as a group:

```
bhs.nsd <- mvmtClass(bighorn, stdt = "10-31")
```

The (optional) `stdt` argument provides a common reference date (here October 31) from which timing parameters are measured (for more details see Quantifying Movement, below). By default `mvmtClass` fits the family of NSD models. Bursts may instead be fit to a model family based on vertical distance by setting the argument `fam = "elev"`.

```
bhs.elev <- mvmtClass(bighorn, fam = "elev", stdt = "10-31")
```

NSD models can be fit to any trajectory, but vertical-distance models can only be fit if the `ltraj` includes an `infolocs` field named `elev` containing measurements of vertical distance.

a. Checking Convergence

The `mvmtClass` function prints warnings when it encounters a problem in fitting one or more movement models. For example, running the command `mvmtClass(bighorn, stdt = "10-31")` returns the following warnings:

```
## Warning: convergence problem(s) for "s110 2008"
##  resident  :     false convergence (8)


## Warning: convergence problem(s) for "s110 2010"
##  resident  :     false convergence (8)
```

indicating problems fitting resident models to the second and fourth bursts. Any model that trigger a warning will be omitted from that burst's output. Consequently, each burst in a `mvmts` object may contain a different number of models, depending on how many were successfully fit. Omitted models will neither be included in plots nor be eligible for consideration as a possible classification (see Classifying Movement, below). Before moving on from model fitting, a first goal may be to confirm that all models were successfully fit to each burst.

We can use the function 'fullmvmt' to identify which bursts were successfully fit and which may be missing models. This function takes `mvmt(s)` objects as input and can provide logical, numeric or named output as determined by the optional argument `out`. So if we wanted to know which bursts in `bhs.nsd` are missing models, we could use:

```
!fullmvmt(bhs.nsd)
```

```
## s110 2007 s110 2008 s110 2009 s110 2010
##     FALSE      TRUE     FALSE      TRUE
```

showing that the second and fourth bursts are incomplete. Alternately, we could determine the number of models successfully fit to each burst using:

```
fullmvmt(bhs.nsd, out = "numer")
```

```
## s110 2007 s110 2008 s110 2009 s110 2010
##         5         4         5         4
```

(showing that only four out of 5 possible models were fit to the second and forth bursts) or could return a character vector containing the names of the models successfully fit to each burst using:

```
fullmvmt(bhs.nsd, out = "name")
```

```
## $`s110 2007`
## [1] "disperser" "migrant"   "mixmig"    "nomad"     "resident"
##
## $`s110 2008`
## [1] "disperser" "migrant"   "mixmig"    "nomad"
##
## $`s110 2009`
## [1] "disperser" "migrant"   "mixmig"    "nomad"     "resident"
##
## $`s110 2010`
## [1] "disperser" "migrant"   "mixmig"    "nomad"
```

which shows that the "resident" model is indeed the only model missing. This function can thus be used to identify the bursts that have had some sort of obstacle to model fit, creating opportunity to follow up on these cases in more detail.

b. Starting Estimates and Constraints

The most common obstacle to model fit (and the most common causes of warnings from the `mvmtClass` function) is parameter starting values or constraints that are a poor match to the data. In the absence of additional input, `mvmtClass` calls the `pEst()` function to get default starting values and parameter constraints for timing parameters (θ,ρ,ϕ,ϕ_2, and κ) and dynamically fills in starting estimates and constraints for distance parameters (γ, δ, and δ_2) based on the data available from each particular burst. We relied on these defaults in the examples above, but can also use `mvmtClass`'s optional `p.est` argument to manually specify these constraints and starting values. The `pEst` function organizes these values into the structure required by `mvmtClass`. Thus we can refit NSD models to `bighorn` using, e.g., a starting estimate of δ = 15:

```
pest.n2 <- pEst(s.d = 15)
bhs.nsd2 <- mvmtClass(bighorn, stdt = "10-31", p.est = pest.n2)
all(fullmvmt(bhs.nsd2))
```

```
## [1] TRUE
```

In contrast to our first call to `mvmtclass`, which returned two convergence warnings, this slight change in parameter starting values allows for all models to be successfully fit.

For some bursts or `ltraj` objects it may not be possible to successfully fit all models with a single set of parameter estimates. In these cases, the `refine` function allows additional attempts to fit models using new sets of starting parameter values and constraints. This function takes two arguments: a `mvmts` object resulting from a prior attempt to fit movement models and a new set of parameter starting values and constraints as organized by `pEst`. When the `refine` function successfully fits a model, it will be added to the output if that model was missing. If the model was already present, the new model will only replace its antecedent if the new model has lower AIC. Thus, to refine elevation model fit for `bhs.nsd` we might use:

```
fullmvmt(bhs.elev)
```

```
## s110 2007 s110 2008 s110 2009 s110 2010
##      TRUE     FALSE      TRUE      TRUE
```

```
pest.e2 <- pEst(s.d = -500)
bhs.elev2 <- refine(bhs.elev, p.est = pest.e2)
all(fullmvmt(bhs.elev2))
```

```
## [1] TRUE
```

Like `mvmtClass`, `refine` returns an object of class `mvmts`, (or from `refine` itself) allowing `refine` to be used recursively. The number of different starting values required to successfully fit models to the different bursts in your data set will depend on the range of movement behaviors your data contain.

b. Visual Checks

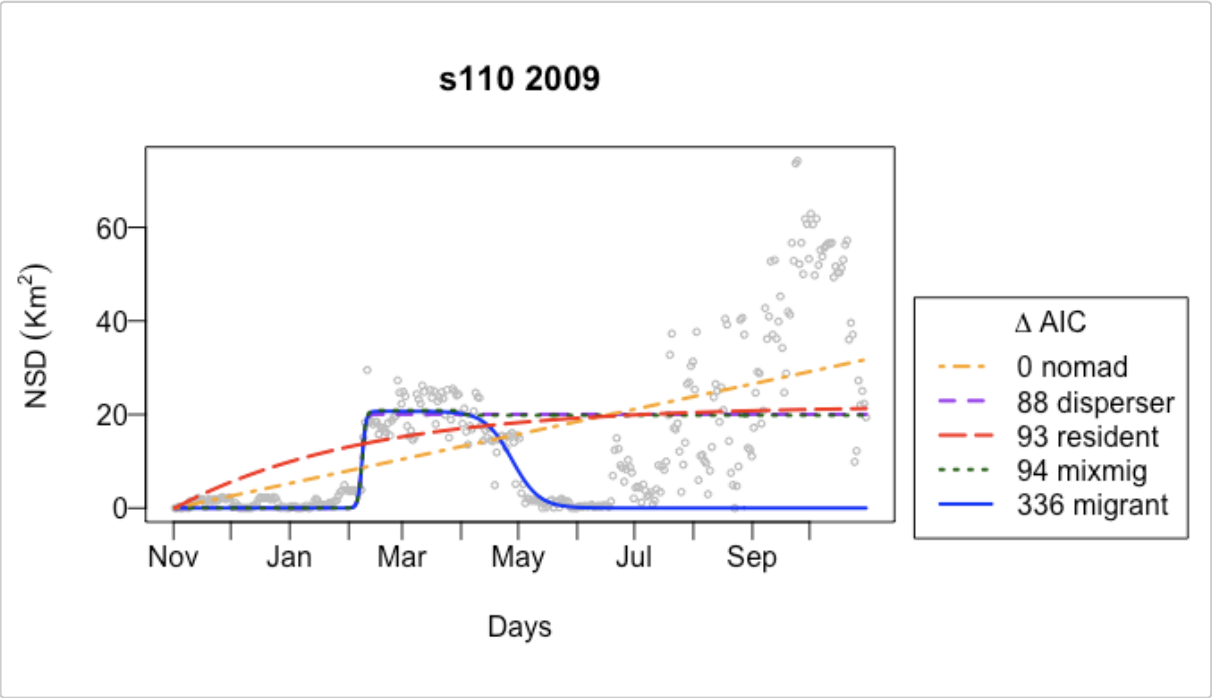
Plotting Models

Unfortunately, models that don't trigger warnings may also represent a poor fit to you data. Again, this is usually a result of poor starting parameter values or constraints, although it may also result from poor correspondence between models and data (e.g. applying elevation models to a long-distance migrant or applying NSD models to an altitudinal migrant). You should always visually inspect all models (even those that don't trigger a warning!) before moving on to classifying animal movement.

The output from `mvmtClass` can be visualized using the familiar `plot` function. Individual `mvmt` objects can be plotted individually, e.g. `plot(bhs.nsd[[3]])`. The `plot` function can also be used to serially plot each element from a `mvmts` object with

```
plot(bhs.nsd)
```

In this case, before each trajectory is plotted, a prompt will appear in the `R` console, identifying the burst and asking whether it should be skipped (`0`) or plotted (any other key; the `Esc` key may also be used from the 'R' console to cancel the command). Unless the argument `new = T` each plot will overwrite its predecessor. If the argument `new = T` R will create a new device for plotting each window (this will not work in R Studio, which only permits a single graphics device).

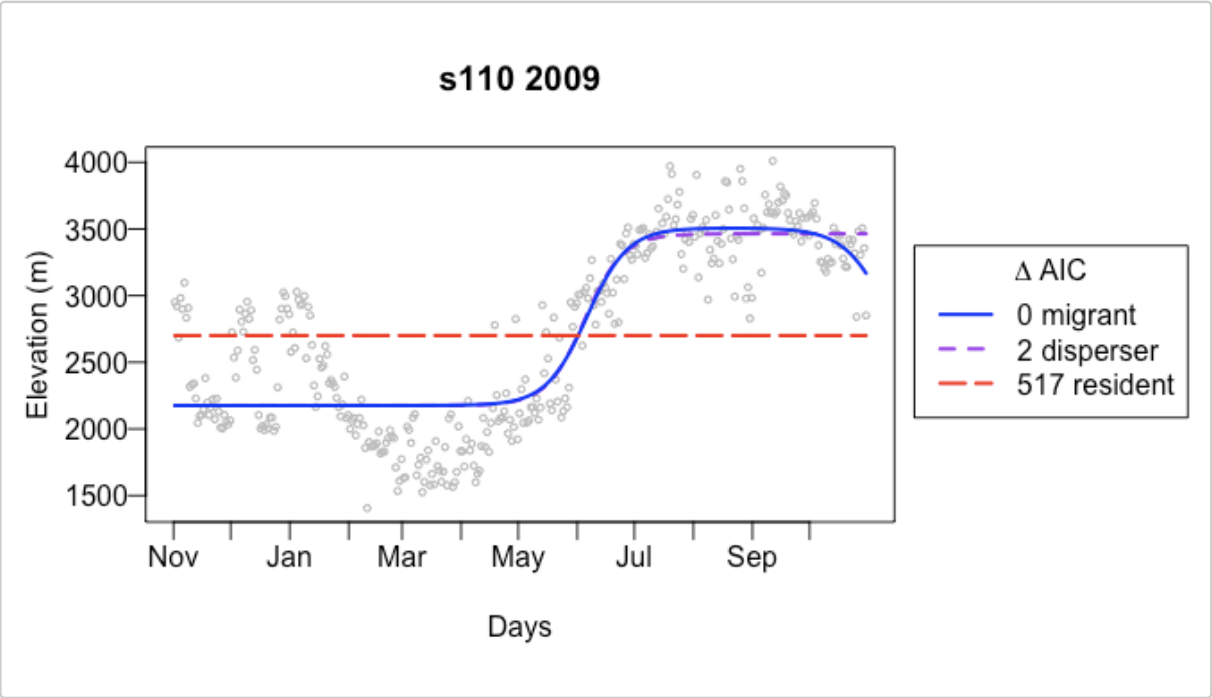


Plots are named by `burst` and each plot also includes a legend ranking the fit of each model from lowest to highest ΔAIC (models can instead be listed alphabetically by setting the optional plot argument `ranked = F`). Each model is represented by a unique color and line type and all model names are spelled out in the legend, except for the mixed-migrant model, which is abbreviated as "mixmig". The x-axis is always time and the y-axis will depend on which family of models is used. The range of the x-axis is constant (a calendar year, plotted from "stdt" or from the first location if "stdt" was not specified), but the range of the y-axis will vary by burst based on the data included (unless this range is specified directly using the `ylim` argument). Location data is shown behind the models as open grey circles.

Plots of your data will often provides helpful cues as to how you can improve starting parameter values and constraints. Sometimes it may also be helpful to visualize models that triggered warnings. By setting `mvmtClass`'s optional argument `warnOnly = T`, all models will be included in the output (even those that triggered convergence-related warnings). This allows the use of `plot`, as above, to visualize problematic bursts.

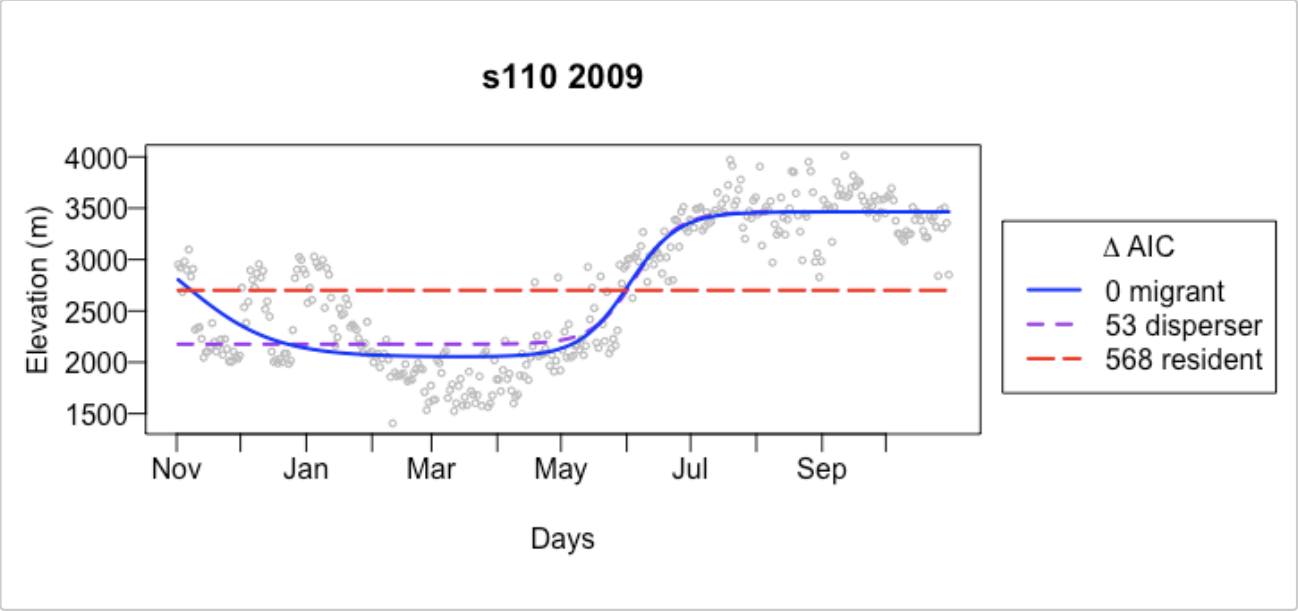
If plotted models appear to be a poor visual match to the location data, it may be worth attempting to `refine` model fit based on new starting parameter estimates and/or restrictions. For example, if we look at

```
plot(bhs.elev[[3]])
```



we see a model depicting upward movement, where downward movement may provide a better fit to the higher elevation values in November-January. To test this, we return to `refine`, setting the upper limit of $\delta = 0$, forcing the fit of a model for downward movement.

```
pest.e3 <- pEst(u.d = 0)
bhs.elev3<- refine(bhs.elev, p.est = pest.e3)
plot(bhs.elev3[[3]])
```



This confirms our suspicion that a model showing downward movement represents a better fit to the data from this burst. Restricting the value of δ in elevation plots may be especially useful, because in most species there is a fixed relationship between migratory direction and season. Defining the direction of migration can thus be used to ensure that all migrant models represent the same seasonal movement. The plot shown above, for example, is now more directly comparable to its neighboring burst, because now all models depict migration to and from the winter range (in this case, downward movement), whereas before they were offset with the third burst depicting (upward) movement to and from the summer range. Unfortunately, there is no comparable

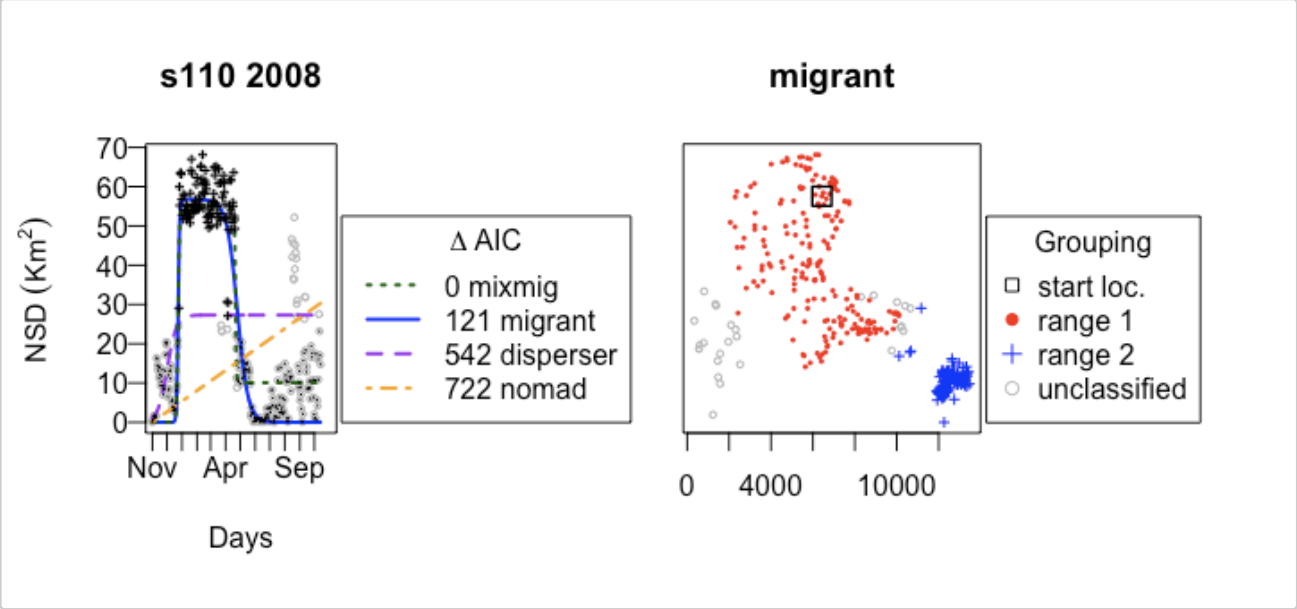
constraint for NSD models and care must be taken to ensure that offset models are not analyzed jointly.

Spatial Comparison

The `spatmig` functions provides a further visual check in the form of a crude spatial comparison. This function produces two side-by-side plots, the first resembles the standard output from using `plot` on a `mvmt` object, the second shows the location data in coordinate space. Using model parameter estimates, `spatmig` categorizes each point on both plots as either belonging to the starting range (range 1), belonging to a secondary range (range 2) or unclassified (includes outliers and points in transit). Each point is plotted with a symbol representing its assigned classification. This allows easy comparison of groupings between the one-dimensional model metrics (NSD, elevation, etc.) and spatial grouping of the locations. Where models fit the data well, we expect, e.g. high and low NSD values to be spatially separated. This is what we see when we look at:

```
spatmig(bighorn[2], bhs.nsd[2])
```

```
## s110 2008    any key to continue (or '0' to skip)
```



Points classified by the movement models as belonging to range 1 (dots) are spatially distinct from points classified as belonging to range 2 (crosses) with minimal to no overlap. By default parameter estimates from the "migrant" model is used to classify points, but this can be changed by setting the argument `model =` "mixmig" or "disperser". The name of the selected model is included as the title of the right panel.

If spatial comparison shows little or poor grouping of points for migrants, it may be worth returning to `refine` or trying to fit the location data to a different family of movement models (e.g. "elev" instead of "nsd" or vice versa).

---

3. Classifying Movement

Now that we have fit movement models to each trajectory, our next task is determining which of these models is best supported. In addition to the legends included in our plots, you may have noticed that the printed output from `mvmtClass` also contains an initial classification (listed under "topmod").

```
bhs.nsd2
```

```
##
## *********** List of class mvmts ***********
##
## Family:           nsd
## Start date:       10-31 (shared origin of timing param)
##
## Shared parameter constraints:
##
##     gamma theta phi delta rho phi2 zeta kappa
## lwr     0     1   1     0   1    1 0.05 -1.00
## upr     -   364  21     - 364   21 0.95 -0.01
##
## Characteristics of the bursts:
##       burst topmod locs          date.begin            date.end
## 1 s110 2007 mixmig  364 2007-10-31 16:00:00 2008-10-29 16:00:00
## 2 s110 2008 mixmig  346 2008-10-30 16:01:00 2009-10-12 16:01:00
## 3 s110 2009  nomad  364 2009-11-01 16:00:00 2010-10-30 16:00:00
## 4 s110 2010 mixmig  210 2010-10-31 16:00:00 2011-05-28 12:00:00
```

This output is based on the function `topmvmt`, which takes `mvmts` or `mvmt` objects as input and returns the best supported model from each burst. These models are named (e.g. “migrant”) allowing us to easily extract information about the prevalence of each movement strategy. (Throughout `migrateR`, the full name is used for all models except for the mixed-migrant model, which is abbreviated “mixmig”.)

```
top.bhs.nsd2 <- topmvmt(bhs.nsd2)
bhs.nsd2.behavior <- names(top.bhs.nsd2)
table(bhs.nsd2.behavior)
```

```
## bhs.nsd2.behavior
## mixmig  nomad
##      3      1
```

a. Constraining Model Selection

In many cases it may helpful to further specify how `topmvmt` selects a top model, using one or more of the functions optional arguments. These arguments have no effect on the underlying models `topmvmt` compares, but can change which top model is selected. We believe that analyses of migration greatly benefit from careful a priori definition of migratory behavior (e.g. minimum travel distances) and our goal here is to allow for the already common practice of including this information in these analyses.

A common complaint is that AIC comparison often favors more complicated models (leading, e.g. to a bias against correctly classifying “resident” behavior; Bunnefeld et al.). To combat this, `topmvmt` applies Arnold’s Rule which imposes an additional penalty for model complexity. This option can be disabled by setting the optional argument `a.rule = F`. Even with this option enabled, privileging complexity can remain a problem, but can often be allayed by including a prior considerations.

In some cases, for example, it may be defensible to entirely exclude one or more models from consideration. The `omit` argument allows models to be excluded by name. For example, if previous research in your study species rules out the possibility of nomadism, you could exclude nomad models from consideration by setting `omit = "nomad"`. This option should be used judiciously; if the problem is a poor fitting model (e.g. a “nomad” that is a poor fit to the data) it’s replacement is guaranteed to be an even poorer fit. A more common

application might be to exclude “mixed migrant” models, whose flexibility can often lead to either a poor correspondence to the intended behavior or a top model that offers little minimal improvement over the simpler “migrant” model (see, e.g. plots from bhs.nsd2).

While these tools may be useful in some circumstances, we strongly recommend relying on parameter-based constraints. `topmvmt` allows minimum values to be set for the length of migratory-range occupancy (ρ) and the distance separating migratory ranges (δ) using the `mrho` and `mdelta` arguments, respectively. These values may be used to constrain models to a consistent biological definition of migration, distinguishing it, for example, from exploratory forays or small seasonal shifts in range use. Any model that fails to meet the specified threshold(s) is excluded from consideration. (Alternatively, minimum values can be set for these parameters using the `p.est` argument in `mvmtClass`, but using this argument to constrain parameter estimates can interfere with model convergence. We therefore recommend applying these thresholds after models have been fit using minimal `p.est` constraints.) For example, specifying a minimum residency period of 21 days, and a minimum distance of 500 meters:

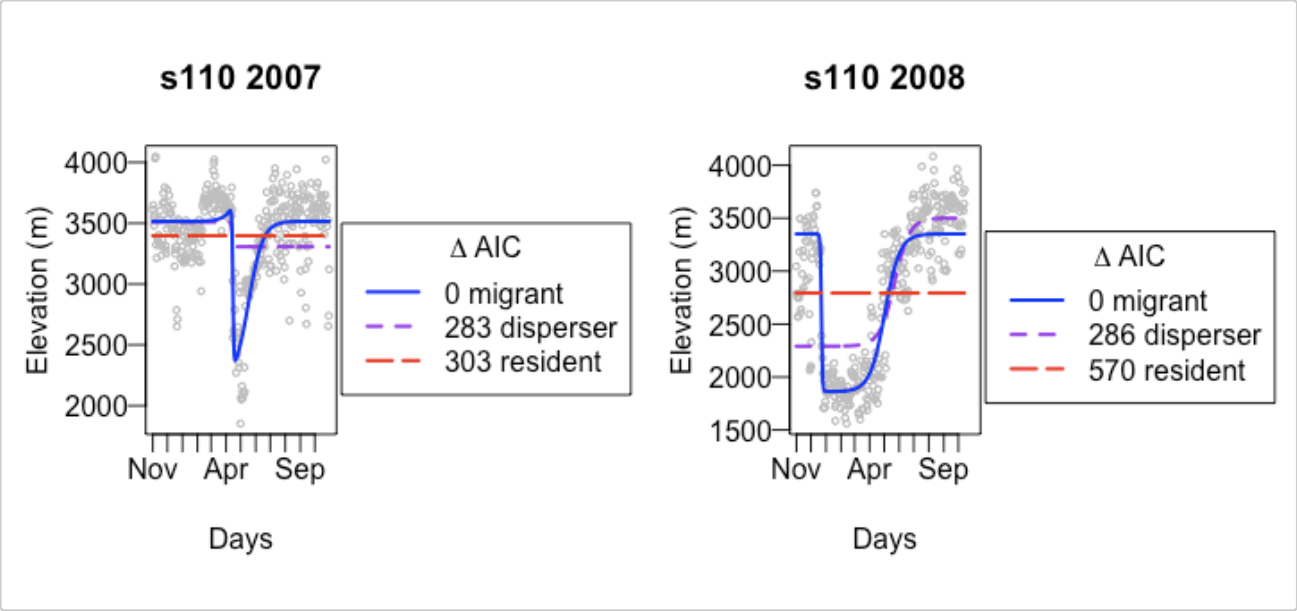
```
summary(bhs.elev3)  # default classification shown under "topmod"
```

```
##       burst  topmod locs          date.begin            date.end
## 1 s110 2007 migrant  364 2007-10-31 16:00:00 2008-10-29 16:00:00
## 2 s110 2008 migrant  346 2008-10-30 16:01:00 2009-10-12 16:01:00
## 3 s110 2009 migrant  364 2009-11-01 16:00:00 2010-10-30 16:00:00
## 4 s110 2010 migrant  210 2010-10-31 16:00:00 2011-05-28 12:00:00
```

```
top.bhs.elev3 <- topmvmt(bhs.elev3, mrho = 21, mdelta = 500)
table(names(top.bhs.elev3))
```

```
##
##  migrant resident
##        3        1
```

shows that one of the `bursts` initially classified as a migrant using the default arguments for `topmvmt` is now instead classified as resident. Comparing the plots from these bursts, we see that the migrant model from the reclassified burst (left) shows a different pattern (starting later, immediately beginning a gradual return) that contrasts with the remaining three bursts, which appear much more similar to one another (for example, the second burst, right).



The units of `mdelta` are always the same as δ, so for NSD models, the units for `mdelta would be km^2, for elevation models, meters.

---

4. Quantifying Movement

For many analyses, the parameters estimated by movement models may be of greater interest than simply identifying which model received the greatest support. Once a list of top models has been found, we can extract and organize parameter estimates from these models using the `mvmt2df` function, which takes models as its argument and outputs a `data.frame` of parameter estimates for each movement behavior present.

```
p.bhs.nsd2 <- mvmt2df(top.bhs.nsd2)
p.bhs.nsd2
```

```
## $mixmig
##               theta       phi    delta       rho    phi2      zeta
## s110 2007 17.03095 21.000000 17.05949 173.49779 1.00000 0.3525712
## s110 2008 53.16007  1.417307 54.90744 113.37707 1.00000 0.8165360
## s110 2010 46.65498 16.390587  8.98141  79.06191 2.81194 0.7227471
##
## $nomad
##                 gamma
## s110 2009 0.08750134
```

In this example nomads are only represented by one example, but parameter estimates for mixed migrants (`mixmig`) are grouped.

a. Derived Estimates

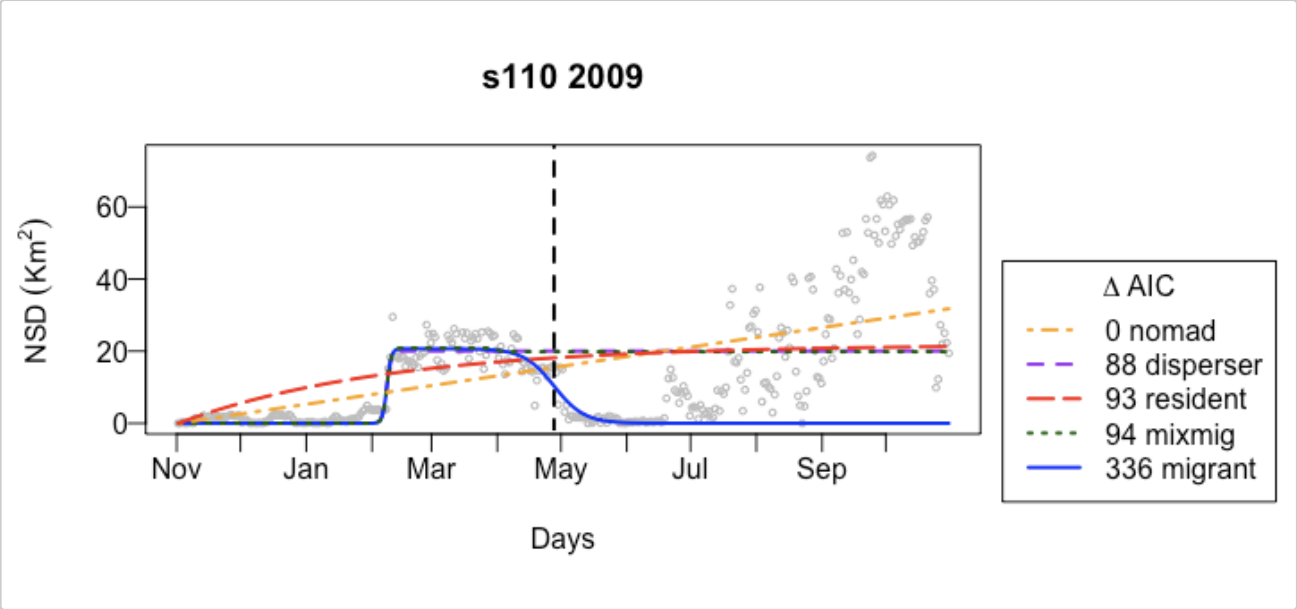
Our models have 2 slight algebraic differences from those of Bunnefeld et al. (see Models of Animal Movement, above). First, rather than estimate the timing (midpoint) of return movements (θ_2 in Bunnefeld et al.), our models directly estimate the duration of migratory-range occupancy (ρ). The timing of return movements can easily be calculated as a derived parameter, (i.e. $\theta_2 = \theta + 2 * \phi + 2 * \phi_2 + \rho$) using the function `theta2`.

```
t2 <- theta2(bhs.nsd2)
t2
```

```
##             theta2       SE
## s110 2007 358.6628 1.557175
## s110 2008 175.3284 1.926233
## s110 2009 178.7189 7.254873
## s110 2010 164.7934 8.189456
```

You can confirm this visually by adding a line for θ_2 to your plot for this burst, i.e.

```
plot(bhs.nsd[[3]])
abline(v = t2[3,1], lty = 2, lw = 2)
```



By default `theta2` calculates estimates for the "migrant" model, but estimates can instead be calculated for "mixed migrant" models by setting the optional argument `mod = "mixmig"`. Finally, unless care is taken in organizing the `ltraj bursts`, values of θ_2 may not be comparable across models (see Comparing Timing Parameters Arcoss Bursts in the next section).

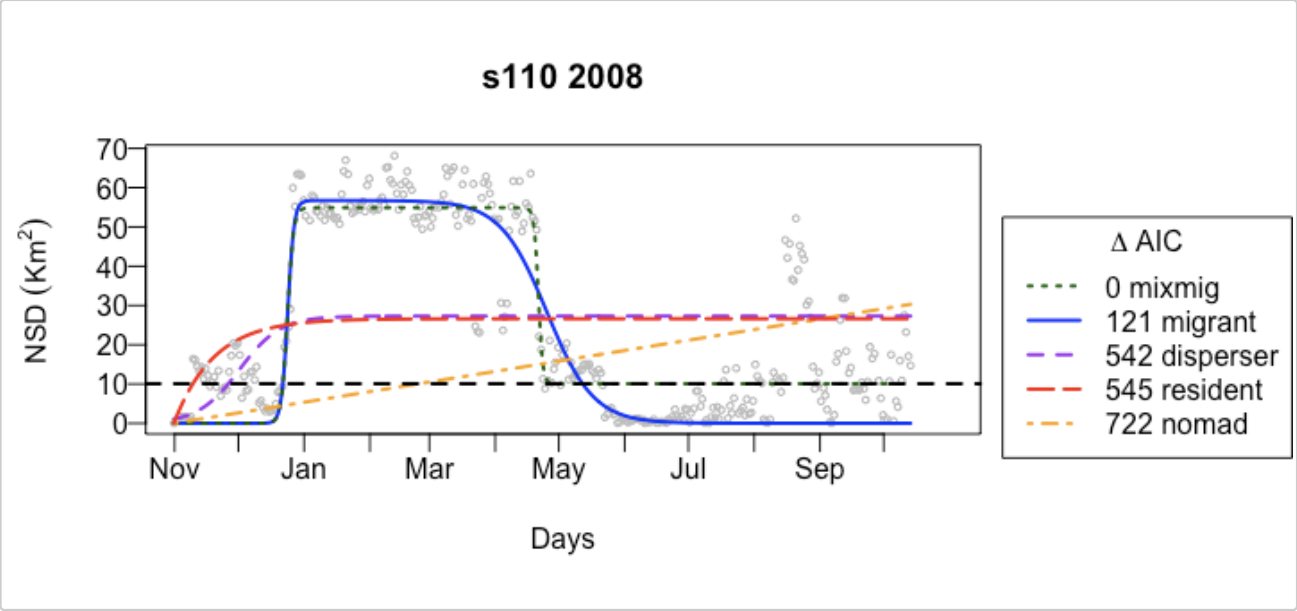
Similarly, our models omit a direct estimate of the return distance traveled by mixed migrants (δ_2 in Bunnefeld et al.), in favor of estimating the percent of their outward movement the return journey represents (ζ). As with θ_2, δ_2 can easily be derived ($\delta_2 = \zeta * \delta$), this time using the function `delta2`.

```
d2 <- delta2(bhs.nsd2)
d2
```

```
##              delta2        SE
## s110 2007  6.014686 1.1227267
## s110 2008 44.833901 1.2573225
## s110 2009  1.047703 2.4579788
## s110 2010  6.491289 0.5962059
```

Again, we can confirm these results visually using, e.g.,

```
plot(bhs.nsd2[[2]])
d <- coef(bhs.nsd2[[2]]@models$mixmig)["delta"]
abline(h = d-d2[2,1], lty = 2, lw = 2)
```



In rare cases where "mixed migrant" model fit is exceptionally poor (e.g. if ζ is allowed to equal zero) this function may return an error, but this can usually be avoided through more careful consideration of parameter constraints.

b. Start and End of Movement

In addition to knowing the midpoints of departing and returning movements (θ and θ_2), it may often be useful to define the time at which these movements begin and end. The `mvmt2dt` function can be used to perform these calculations. This function takes a `mvmt(s)` object as input and returns the start and end date of each movement The start and end of movements are defined by the argument "p", the percent of the migratory distance traveled. By default "p" = 0.05, such that the start of migration is estimated as the time at which model predictions = $0.05 * \delta$ and end time of movement is defined as the time at which model predictions = $(1 - 0.05) * \delta$. By default dates are calculated for the "migrant" model, but the "mixed migrant" ("mixmig") or "disperser" models can instead be selected using the optional "mod" argument. For each burst `mvmt2dt` returns a `data.frame` with each date in two formats: decimal days from "stdt" ("dday"; calculated from the first location if "stdt" was not specified) and POSIXct ("date"). The function will issue a warning if the intervals of departing and returning movement overlap.

```
mvmt2dt(bhs.nsd, mod = "mixmig")
```

```
## $`s110 2007`
##           dday                date
## str1 -43.98422 2007-09-17 00:22:43
## end1  79.68222 2008-01-18 15:22:23
## str2 231.58663 2008-06-18 14:04:44
## end2 237.47550 2008-06-24 11:24:43
##
## $`s110 2008`
##           dday                date
## str1  48.98687 2008-12-18 22:41:05
## end1  57.33327 2008-12-27 06:59:54
## str2 168.42731 2009-04-17 10:15:19
## end2 174.31619 2009-04-23 07:35:18
##
```

```
## $`s110 2009`
##          dday                date
## str1  97.0989 2010-02-05 01:22:24
## end1 102.9878 2010-02-10 22:42:23
## str2 149.2762 2010-03-29 06:37:46
## end2 155.1651 2010-04-04 03:57:45
##
## $`s110 2010`
##            dday                date
## str1  -1.606082 2010-10-29 09:27:14
## end1  94.916012 2011-02-02 20:59:03
## str2 155.842458 2011-04-04 20:13:08
## end2 172.401419 2011-04-21 09:38:02
```

---

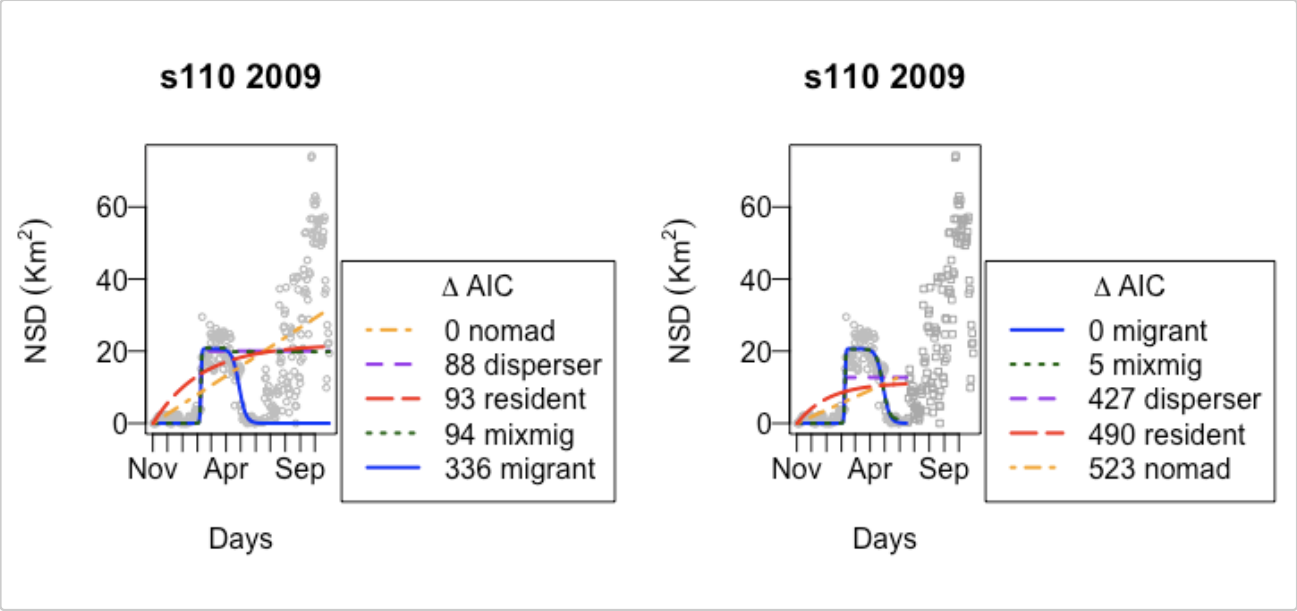
5. Advanced Options

a. Omitting Points

Many trajectories begin or end with problematic points that may compromise model fit. While we generally recommend that these problems be handled when organizing data into an `ltraj` object, we have included two optional arguments to `mvmtClass` which allow for ad hoc adjustments. For example, the third burst from bighorn ends with a period of increased variance in NSD, which compromises the interpretation of these models, leading to a classification as "nomad" even though the "migrant" model appears visually to be a superior match. To address this problem, we can use the "ecut" argument, which excludes points from the end of a trajectory. Like "stdt", this argument requires a character string formatted as "%m-%d". Looking at the plot for s110 2009, above, we might choose June 15 ("6-15") as a reasonable cutoff. We might then try:

```
bhs3.nsd <- mvmtClass(bighorn[3], stdt = "10-31", ecut = "6-15")
```

We show the original models on the left, below, and the revised models (calculated after points were excluded) below on the right.



With these later points excluded, the classification shifts from "nomad" to a clear-cut case of migration. The

excluded points are plotted as open squares, while retained points remain open circles.

The "scut" argument functions analogously to "ecut", instead excluding points from the start of a trajectory rather than its end. Both "cut" arguments can only be changed in `mvmtClass` and will be ignored if included in `refine`. Bounded points can only be omitted by removing all locations before or after or by manually altering a burst. Finally, although these arguments provide a convenient tool for data exploration, we caution against reliance on ad hoc adjustments for model fit. Variation in individual behavior can make it difficult to find a single "ecut/scut" value appropriate to all bursts. Plotting NSD models fit to the four trajectories in "bighorn", for example, could arguably suggest at least three different "ecut" values. When included, choice of "cut" values should be acknowledged as ad hoc, or, preferably, justified by other quantitative means.

b. Sensitivity to Starting Coordinates

As mentioned previously, NSD is calculated as the squared distance from a trajectory's first point to each subsequent location. NSD values can therefore be sensitive to the location of the first point. These methods implicitly assume that the first point is representative of the starting seasonal range (Bunnefeld et al.). If, instead, the first point happens to be an outlier, e.g. representing an exploratory foray, this can lead to poor model fit and, consequently, erroneous inference. Instead of fitting models to NSD, `mvmtClass` allows models to be fit to relative NSD (rNSD) the net squared displacement calculated from a reference point other than the first location. To specify a reference location set the argument `rloc = x` where `x` is the number of the location you wish to use as a reference. For example, `mvmtClass(bighorn[1], rdt = 2)` fits models to the first trajectory in bighorn using rNSD values calculated relative to the trajectory's second location. The function `findrloc` can be used to compare different choices of reference date using AIC. This function returns a number for the reference location that results in a top model with the lowest overall AIC. Thus, `findrloc` can be used to determine an optimum reference date for input into `mvmtClass`.

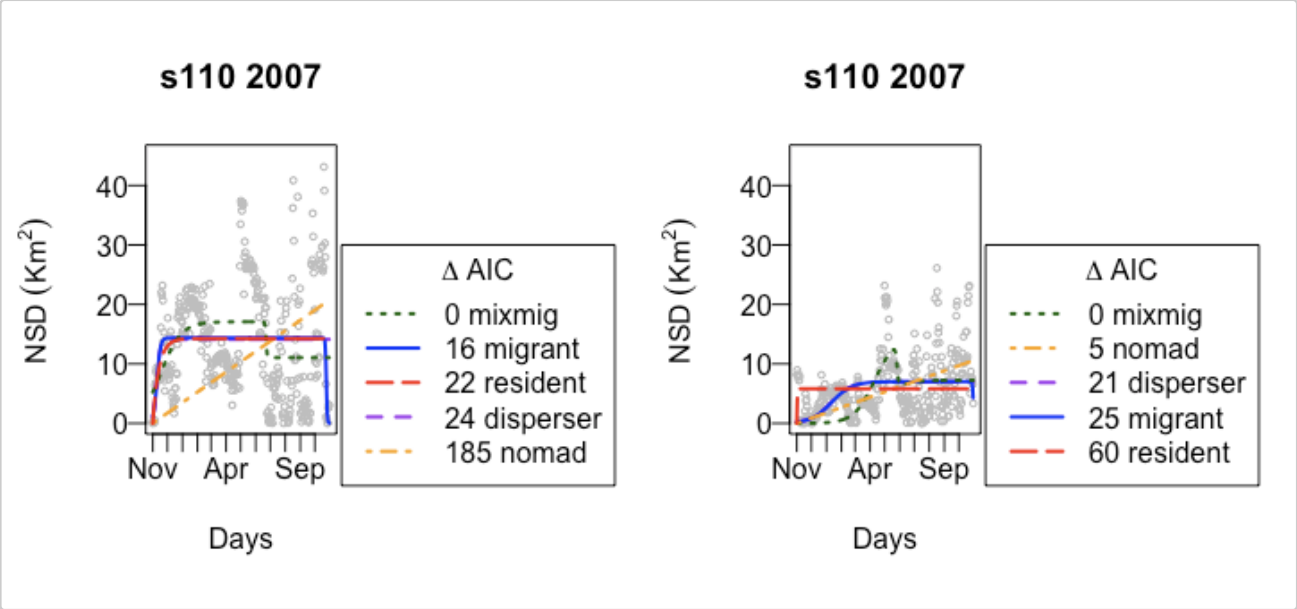
```
rlocs <- findrloc(bighorn)
```

```
## Finding best supported rNSD for 4 trajectories. This may take a moment.
##        burst            location rloc     model
## 1  s110 2007 2007-11-10 16:01:00   11    mixmig
## 2  s110 2008 2008-11-07 16:01:00    9    mixmig
## 3  s110 2009 2009-11-09 15:00:00    9    mixmig
## 4  s110 2010 2010-11-12 16:00:00   13    mixmig
```

```
bhs.rnsd <- mvmtClass(bighorn, rloc = rlocs$rloc, stdt = "10-31", p.est = pest.n2)
bhs.rnsd2 <- refine(bhs.rnsd, pEst(s.t = 220))
fullmvmt(bhs.rnsd2,"name")
```

```
## $`s110 2007`
## [1] "disperser" "migrant"   "mixmig"    "nomad"     "resident"
##
## $`s110 2008`
## [1] "disperser" "migrant"   "mixmig"    "nomad"     "resident"
##
## $`s110 2009`
## [1] "disperser" "migrant"   "mixmig"    "nomad"     "resident"
##
## $`s110 2010`
## [1] "disperser" "migrant"   "mixmig"    "nomad"     "resident"
```

To see how this changes model fit, we can compare two plots.



The left plot is the original NSD and the revised rNSD plot is on the right. Note that the y-axes here are held constant and that the rNSD plot provides lower displacement values. While the top model remains the same for both of these plots, the rNSD plot shows a more regular pattern that more closely matches the movement models. These rNSD model could be followed up either by omitting points following increase in variance in rNSD (starting ~June 25; Omitting Points) or by constraining model selection.

There is no need to specify a reference date for models of vertical movement (`fam = elev`), because one of the advantages of these models is that they are insensitive to starting location. (For more details on rNSD, see Spitz et al. 2016.)

c. Comparing Timing Parameters Across Bursts

Returning to “bighorn” we can see that several of the bursts begin on different calendar days(“date.begin”).

```
##
## *********** List of class ltraj ***********
##
## Type of the traject: Type II (time recorded)
## Irregular traject. Variable time lag between two locs
##
## Characteristics of the bursts:
##     id     burst nb.reloc NAs          date.begin            date.end
## 1 s110 s110 2007      364   0 2007-10-31 16:00:00 2008-10-29 16:00:00
## 2 s110 s110 2008      346   0 2008-10-30 16:01:00 2009-10-12 16:01:00
## 3 s110 s110 2009      364   0 2009-11-01 16:00:00 2010-10-30 16:00:00
## 4 s110 s110 2010      210   0 2010-10-31 16:00:00 2011-05-28 12:00:00
##
##
##  infolocs provided. The following variables are available:
## [1] "elev"
```

The parameters estimates from `mvmtClass` models are all made relative to the first location included in a burst. To facilitate direct comparison of parameter estimates across bursts, `mvmtClass` includes the `stdt` argument, which you can use to standardize the calendar date from which all timing parameters are estimated. The `stdt` argument requires character input formatted as “%m-%d”. For example,
`mvmtClass(bighorn, stdt = "10-31")`, which we used above, fits models to bighorn estimating timing

parameters as days from October 31. Comparing the parameter estimates to our default fit, we can see that the only changes are to the estimates for θ, the time at which the migratory movement is half completed.

```
bhs.elev0 <- mvmtClass(bighorn, fam = "elev", p.est = pest.e2)


## Warning: convergence problem(s) for "s110 2008"
##  migrant   :     singular convergence (7)


bhs.elev0r1 <- refine(bhs.elev0, p.est = pest.e3)
fullmvmt(bhs.elev0r1, out = "name")


## $`s110 2007`
## [1] "disperser" "migrant"   "resident"
##
## $`s110 2008`
## [1] "disperser" "migrant"   "resident"
##
## $`s110 2009`
## [1] "disperser" "migrant"   "resident"
##
## $`s110 2010`
## [1] "disperser" "migrant"   "resident"


bhs.elev.stdt.mig <- topmvmt(bhs.elev3, omit = c("resident", "disperser"))
bhs.elev.mig <- topmvmt(bhs.elev0r1, omit = c("resident", "disperser"))

p.elev.stdt <- mvmt2df(bhs.elev.stdt.mig)
p.elev <- mvmt2df(bhs.elev.mig)

round(p.elev.stdt[[1]] - p.elev[[1]] ,2)


##           gamma theta phi delta rho phi2
## s110 2007     0  0.67   0     0   0    0
## s110 2008     0 -0.33   0     0   0    0
## s110 2009     0  1.71   0     0   0    0
## s110 2010     0  0.67   0     0   0    0
```

We see here that for the first four bursts, theta is the only parameter estimate that changes. Occasionally, though, setting a “stdt” can have larger consequences for model fit. Because values of θ are calculated relative to either the first location or the “stdt”, setting a value for “stdt” that differs from the first location effectively changes the starting value and range limits for θ. This is most likely to be a concern when a “stdt” is chosen that differs greatly from a trajectory’s first location. This problem is most easily addressed by building `ltraj` data sets where each burst begins on or near your preferred “stdt”, but can also be resolved by making manual changes to starting values and constraints for θ through `mvmtClass`’s “p.est” argument. As always, if you have any uncertainty about how changing inputs affect model fit, visually checking the models is a good idea.

d. Class “mvmt”

Objects of class `mvmt` include three slots: “models”, “param” and “data”; “Models” is a `list` containing a named element for every model that was successfully fit, “param” is a `data.frame` containing minimum, starting and maximum parameter values, and “data” is a `data.frame` containing the data used to fit the models (three fields,

"decday" = decimal day, either "nsd" or "elev" depending on the model family fit, and "cut" indicating whether any points were excluded before fitting the models). The familiar `str` function can be used to navigate the contents of `mvmt` class objects, e.g.,

```
str(bhs.nsd[["s110 2010"]])
```

```
## Formal class 'mvmt' [package 'migrate'] with 3 slots
##  @models
##      $ disperser (AIC = 1125.267)
##      $ migrant   (AIC = 1031.667)
##      $ mixmig    (AIC = 1010.786)
##      $ nomad     (AIC = 1238.085)
##  @param        [lwr, strt, upr]
##      $ gamma     0     6   20
##      $ theta     1     90      364
##      $ phi       1     7   21
##      $ delta     0     10      20
##      $ rho       1     90      364
##      $ phi2      1     7   21
##      $ zeta      0     0   1
##      $ kappa     -1    0   0
##  @data   'data.frame':   (210 locations)
##      $ decday     0.67 1.62 2.67 3.67 4.67 5.67 ...
##      $ nsd        0 0.14 0.44 0.28 1.65 1.76 ...
```

shows us that only four of five possible models were successfully fit to this burst, with the "resident" model missing. Additional information about `mvmt` objects, including a record of any optional arguments included in the call to `mvmtClass` are stored as attributes.

References

Arnold, T. W. (2010). Uninformative Parameters and Model Selection Using Akaike's Information Criterion. *The Journal of Wildlife Management*, **74**, 1175–1178.

Bunnefeld, N., van Moorter, B., Rolandsenm C.M., Dettki, H., Solberg, E.J. & Ericsson, G. (2011). A model-driven approach to quantify migration patterns: individual, regional and yearly differences. *Journal of Animal Ecology*, **80**, 466-476.

Spitz, D. (2015). Does Migration Matter? Causes and Consequences of Migratory Behavior in Sierra Nevada Bighorn Sheep. Dissertation thesis, University of Montana.