Reference manual for the KFtrack package

March 21, 2012

Version 0.70

Date 2012-03-21

Title kftrack

Author John Sibert <sibert@hawaii.edu>, Anders Nielsen <anders.nielsen@hawaii.edu>

Maintainer Anders Nielsen <anders.nielsen@hawaii.edu>

Depends R (>= 1.5.0)

Description This package estimates most probable track from archival tagging data

License BSD

URL http://www.r-project.org, https://www.soest.hawaii.edu/tag-data/tracking/kftrack

R topics documented:

kftrack-package *KFtrack estimates most probable track from archival tagging data*

Description

Fit a state space model to observed geo–locations via the extended Kalman filter. A few variations of the model is available.

Author(s)

John Sibert <sibert@hawaii.edu>, Anders Nielsen <anders.nielsen@hawaii.edu>

References

Sibert, J., Musyl, M. K. and Brill, R.W. (2003) Horizontal movements of bigeye tuna near Hawaii determined by Kalman filter analysis of archival tagging data. Fish. Oceanogr. 12(3):141–151.

addcoast *Adds coastline to plotted track*

Description

Adds costline to a plotted track, if GMT is installed on the system.

Usage

```
addcoast(res = 3, ...)
```

Arguments

res	an integer resolution from 1 (full) to 5 (crude)
...	additional graphical parameters

Note

GMT must be installed in order to use this function

Author(s)

John Sibert <jsibert@soest.hawaii.edu>, Anders Nielsen <anielsen@dina.kvl.dk>

References

http://gmt.soest.hawaii.edu

See Also

addmap

Examples

```
data(big.241)
fit<-kftrack(big.241, fix.last=FALSE)
plot(fit)
#addcoast()
```

---

addmap *Adds map to plotted track*

Description

Adds blue/green areas to the map with the estimated track, if GMT is installed on the system.

Usage

```
addmap(kf.obj, res = 3, ci=FALSE, points=TRUE, pred=TRUE, most=TRUE, ...)
```

Arguments

kf.obj	an object of type kftrack
res	the resolution
ci	If TRUE adds confidence regions for the most probable track to the plot
points	If FALSE the raw geo-locations are omitted
pred	If FALSE the predicted plot is omitted
most	If FALSE the most probable track is omitted
...	additional graphical parameters

Note

GMT must be installed in order to use this function

Author(s)

John Sibert <jsibert@soest.hawaii.edu>, Anders Nielsen <anielsen@dina.kvl.dk>

References

http://gmt.soest.hawaii.edu

See Also

addcoast

Examples

```
data(big.241)
fit<-kftrack(big.241, fix.last=FALSE)
plot(fit)
#addmap(fit)
```

`big.241` *Data for bigeye tuna tag number 241*

Description

The `big.241` data frame has 76 rows and 5 columns.

Usage

```
data(big.241)
```

Format

This data frame contains the following columns:

day a numeric vector containing integer values corresponding to the day part of the observation dates

month a numeric vector containing integer values corresponding to the month part of the observation dates

year a numeric vector containing four digit integer values corresponding to the year part of the observation dates

long a numeric vector containing the longitude measurements

lati a numeric vector containing the latitude measurements

Note

The dates should be in increasing order.

Author(s)

John Sibert <jsibert@soest.hawaii.edu>, Anders Nielsen <anielsen@dina.kvl.dk>

References

Sibert, J., Musyl, M. K. and Brill, R.W. (2002) Horizontal movements of bigeye tuna near Hawaii determined by Kalman filter analysis of archival tagging data. Fish. Oceanogr. In press(?):??–??.

See Also

`kftrack`

Examples

```
data(big.241)
big.241[1:10,]
#fit<-kftrack(big.241, fix.last=FALSE)
#plot(fit)
```

`kftrack` *Kalman filter tracking (of tagged individuals)*

Description

Fit a state space model to observed geo–locations via the extended Kalman filter. A few variations of the model is available.

Usage

```
kftrack(data, fix.first=TRUE, fix.last=TRUE,
  theta.active=c(u.active, v.active, D.active, bx.active, by.active,
                 sx.active, sy.active, a0.active, b0.active, vscale.active),
  theta.init=c(u.init, v.init, D.init, bx.init, by.init, sx.init, sy.init,
               a0.init, b0.init, vscale.init),
  u.active=TRUE, v.active=TRUE, D.active=TRUE, bx.active=TRUE,
  by.active=TRUE, sx.active=TRUE, sy.active=TRUE, a0.active=TRUE,
  b0.active=TRUE, vscale.active=TRUE, u.init=0, v.init=0, D.init=100,
  bx.init=0, by.init=0, sx.init=.5, sy.init=1.5, a0.init=0.001, b0.init=0,
  vscale.init=1, var.struct="solstice", dev.pen=0.0, save.dir=NULL, admb.string="")
```

Arguments

`data`	A data.frame consisting of five columns. The first three columns should contain `day`, `month` and `year` corresponding to valid dates. The dates must be sorted in ascending order. Column four and five should contain the `longitude` and `latitude` in degrees. A valid data set example is supplied as part of the package (see `big.241`) .
`fix.first`	`TRUE` (default) if the first position in the data set is the true release position (known without error), `FALSE` otherwise.
`fix.last`	`TRUE` (default) if the last position in the data set is the true recapture/popoff position (known without error), `FALSE` otherwise.
`theta.active`	A logical vector with nine elements, each corresponding to a model parameter. If an element is set to `TRUE` the value of corresponding parameter is optimized, otherwise it is kept at its initial value. The default value is `TRUE` for all parameters. The values `1/0` can be used instead of `TRUE/FALSE`. The order of the elements in this vector is `c(u.active, v.active, D.active, bx.active, by.active, sx.active, sy.active, a0.active, b0.active)`, hence a value of `c(0,0,1,1,1,1,1,1,1)` would result in a model where u and v were fixed at there initial values.

`theta.init` A numeric vector with nine elements, each corresponding to a model parameter. The order of the elements in this vector is `c(u.init, v.init, D.init, bx.init, by.init, sx.init, sy.init, a0.init, b0.init)` and the default value is `c(0, 0, 100, 0, 0, 0.5, 1.5, 0.001, 0)`. It is unwise to initialize elements `D.init`, `sx.init` and `sy.init` below zero, as they correspond to standard deviations.

`u.active` `TRUE` (default) if u should be optimized, `FALSE` if it should be fixed at its initial value.

`v.active` `TRUE` (default) if v should be optimized, `FALSE` if it should be fixed at its initial value.

`D.active` `TRUE` (default) if D should be optimized, `FALSE` if it should be fixed at its initial value.

`bx.active` `TRUE` (default) if b_x should be optimized, `FALSE` if it should be fixed at its initial value.

`by.active` `TRUE` (default) if b_y should be optimized, `FALSE` if it should be fixed at its initial value.

`sx.active` `TRUE` (default) if σ_x should be optimized, `FALSE` if it should be fixed at its initial value.

`sy.active` `TRUE` (default) if σ_y should be optimized, `FALSE` if it should be fixed at its initial value.

`a0.active` If the variance structure `var.struct="solstice"` is chosen this flag should be set to `TRUE` (default) if a_0 should be optimized, `FALSE` if it should be fixed at its initial value. If a different variance structure is selected this flag is ignored.

`b0.active` If the variance structure `var.struct="solstice"` is chosen this flag should be set to `TRUE` (default) if b_0 should be optimized, `FALSE` if it should be fixed at its initial value. If a different variance structure is selected this flag is ignored.

`vscale.active` If the variance structure `var.struct="specified"` and this is `TRUE` a common scaling parameter is estimated for the specified covariance matrices

`u.init` The initial value of u. Default is 0.

`v.init` The initial value of v. Default is 0.

`D.init` The initial value of D. Default is 100.

`bx.init` The initial value of b_x. Default is 0.

`by.init` The initial value of b_y. Default is 0.

`sx.init` The initial value of σ_x. Default is 0.5.

`sy.init` The initial value of σ_y. Default is 1.5.

`a0.init` If the variance structure `var.struct="solstice"` is chosen this sets the initial value of a_0. Default is 0.001. If a different variance structure is selected this is ignored.

`b0.init` If the variance structure `var.struct="solstice"` is chosen this sets the initial value of b_0. Default is 0. If a different variance structure is selected this is ignored.

`vscale.init`	Initial value for the common scaling parameter for the specified covariance matrices
`var.struct`	Four options are available: `"uniform"`, `"specified"`, `"solstice"`(default) and `"daily"`. These are defined in the details section.
`dev.pen`	If `var.struct="daily"` is set, this parameter sets the derivative penalty.
`save.dir`	`NULL` (default) if the estimation should be done in a temporary directory, otherwise the quoted name of the directory where the estimation should be saved.
`admb.string`	Additional command line arguments to the underlying AD Model Builder program can be passed as a string. For instance "-est". The available command line arguments can be found in the AD Model Builder documentation (see http://otter-rsch.com)

Details

The model is a state space model, where the state equation is describing the movements of a fish in an axis–parallel plane. A random walk model is assumed:

$$\alpha_i = \alpha_{i-1} + c_i + \eta_i, \quad i = 1, \ldots, T$$

Here α_i is a two dimensional vector containing the coordinates at time t_i, c_i is the drift vector describing the deterministic part of the movement, and η_i is the noise vector a describing the random part of the movement. The deterministic part of the movement is assumed to be proportional to time:

$$c_i = (u\Delta t_i, v\Delta t_i)'$$

The random part is assumed to be serially uncorrelated and follow a two dimensional Gaussian distribution with mean vector 0 and covariance matrix Q_i, where

$$Q_i = 2D\Delta t_i < a0 > I, \quad \text{(here 'I' is the } 2 \times 2 \text{ unit matrix).}$$

The measurement equation of the state space model is a non–linear mapping of the coordinates on the axis–parallel plane on to the sphere. The original coordinates were in Nautical miles and the coordinates on the sphere are in degrees of longitude and latitude. The measurement equation describing the actual position y_i is:

$$y_i = z(\alpha_i) + d_i + \varepsilon_i, \quad i = 1, \ldots, T$$

where z is the coordinate change function given by:

$$z(\alpha_i) = \left(\frac{\alpha_{i,1}}{60\cos(\alpha_{i,2}\pi/180/60)}\ ,\ \frac{\alpha_{i,2}}{60} \right)'$$

d_i is the observed bias:

$$d_i = (b_x\ ,\ b_y)'$$

and ε_i is the measurement error which is assumed to follow a Gaussian distribution with mean vector 0 and covariance matrix H_i, where

$$H_i = \text{diag}(\sigma_x^2, \sigma_{y_i}^2)$$

The arguments `u.active`, `v.active`, `D.active`, `bx.active`, `by.active`, `sx.active` and `sy.active` offers an alternative way of specifying the argument vector `theta.active`. This is useful for two reasons. Firstly, if only a few of the elements of `theta.active` is changed from their defaults, it is convenient not having to specify the entire vector. Secondly, it is not required to remember the correct order of the arguments in `theta.active`, if they are specified individually. If `theta.active` is specified any individually specified arguments are ignored and the values of `theta.active` is used.

Similarly the arguments `u.init`, `v.init`, `D.init`, `bx.init`, `by.init`, `sx.init` and `sy.init` offers an alternative way of specifying the argument vector `theta.init`.

The argument `var.struct` sets the model for the latitude error $\sigma^2_{y_i}$. Three options are available.

If `"uniform"` the same variance is assumed for all observations.

If `"solstice"` the variance is assumed to follow the model:

$$\sigma^2_{y_i} = \sigma^2_{y_0} \Big/ \Big(\cos^2 \left(2\pi (J_i + (-1)^{s_i} b_0)/365.25 \right) + a_0 \Big)$$

where J_i is the number of days since last solstice prior to all observations, s_i is the season number since the beginning of the track (one for the first 182.625 days, then two for the next 182.625, then three and so on). a_0, b_0 and $\sigma^2_{y_0}$ are model parameters.

If `"daily"` the variance is assumed to have a different value at each time step, and ψ_i are normally distributed random variables with mean zero and variance σ^2_ψ representing transient deviations in the latitude error.

Value

An object of class `kftrack` is returned. This object contains information about the fit and estimated tracks.

Author(s)

John Sibert <jsibert@soest.hawaii.edu>, Anders Nielsen <anielsen@dina.kvl.dk>

References

Sibert, J., Musyl, M. K. and Brill, R.W. (2002) Horizontal movements of bigeye tuna near Hawaii determined by Kalman filter analysis of archival tagging data. Fish. Oceanogr. In press(?):??–??.

See Also

`plot.kftrack`

Examples

```
data(big.241)
fit<-kftrack(big.241, fix.last=FALSE)
plot(fit)
```

`plot.kftrack` *Plot a kftrack object*

Description

Plots the estimated track.

Usage

```
plot.kftrack(x, ci=FALSE, points=TRUE, pred=TRUE, most=TRUE, gmt=FALSE, ...)
```

Arguments

`x`	is a `kftrack` object typically generated with the `kftrack` function
`ci`	If TRUE adds confidence regions for the most probable track to the plot
`points`	If FALSE the raw geo-locations are omitted
`pred`	If FALSE the predicted plot is omitted
`most`	If FALSE the most probable track is omitted
`gmt`	If TRUE (and if gmt is correctly installed) a a GMT-based postscript version of the plot will be saved in the working directory
`...`	additional graphics parameters

Value

No value is returned this function is invoked for its side effects.

Author(s)

John Sibert <jsibert@soest.hawaii.edu>, Anders Nielsen <anielsen@dina.kvl.dk>

See Also

`kftrack`, `addmap`, `addcoast`

Examples

```
data(big.241)
fit<-kftrack(big.241, fix.last=FALSE)
plot(fit)
```

plot.kftrack.scan *Plot a kftrack scanning object*

Description

Plots the result of a scanning for and estimation of a premature pop–off position.

Usage

```
plot.kftrack.scan(x, ...)
```

Arguments

x	is a kftrack.scan object typically generated with the .init.scan function.
...	additional graphics parameters

Value

No value is returned this function is invoked for its side effects.

Author(s)

John Sibert <jsibert@soest.hawaii.edu>, Anders Nielsen <anielsen@dina.kvl.dk>

See Also

kftrack, addmap, addcoast

print.kfhead *Print a kfhead object*

Description

Prints the header information from the kftrack.dat file.

Usage

```
print.kfhead(x, ...)
```

Arguments

x	is an kfhead object
...	additional arguments

Author(s)

John Sibert <jsibert@soest.hawaii.edu>, Anders Nielsen <anielsen@dina.kvl.dk>

See Also

print.kftrack

---

print.kftrack *Print kftrack object*

Description

Prints a pretty summary of an object of class kftrack

Usage

```
print.kftrack(x, ...)
```

Arguments

x	an object of class kftrack typically generated with the kftrack function.
...	additional arguments

Author(s)

John Sibert <jsibert@soest.hawaii.edu>, Anders Nielsen <anielsen@dina.kvl.dk>

See Also

print.kftrack

---

upload.track *Upload a track to be mapped on the server*

Description

Uploads a track to be mapped on the server using Google maps

Usage

```
upload.track(fit)
```

Arguments

fit	A fit as returned from kftrack or kfsst

Details

This function will prepare a data file with the information in the fitted object and some information about the track and on how to contact the person who posts it. The track can then be viewed on a Google map on the server.

Value

No value is returned, but a file named 'track4upload.RData' is saved in the present working directory.

Author(s)

John Sibert <jsibert@soest.hawaii.edu>, Anders Nielsen <anielsen@dina.kvl.dk>

References

Google maps API http://www.google.com/apis/maps/

See Also

addmap,kftrack,write.html,write.kml

---

write.html *Write html page with a map of the track(s)*

Description

Write html page with a map of the track(s). This html code Google maps interface to display the estimated track(s) on a pretty map.

Usage

```
write.html(fitlist, description = rep("", length(fitlist)), file = "track.html", np
```

Arguments

fitlist	One fit or a list of fits as returned from kftrack or kfsst
description	One string (in case of only one fit), or a vector of strings, each with a description of a tag. Html tags are allowed
file	The file where the html code is written
npoints	Number of points in the polygon representation of the confidence ellipses
level	The confidence level
key	The Google Maps API key corresponding to the site where the page is to be published. If no key is supplied the page will still work, but should only be used locally

Author(s)

John Sibert <jsibert@soest.hawaii.edu>, Anders Nielsen <anielsen@dina.kvl.dk>

References

Google maps API http://www.google.com/apis/maps/

See Also

addmap and kftrack

Examples

```
data(big.241)
fit<-kftrack(big.241, fix.last=FALSE)
write.html(fit)
#browseURL(normalizePath('track.html'))
```

---

write.kml *Write fit to kml file*

Description

Writes a copy of the track, the predicted track, the most proable track, and its confidence ellipses to a kml file. These files can be used with Google Earth to see the track.

Usage

```
write.kml(fit, description = "", file = "track.kml", npoints = 20, level = 0.95)
```

Arguments

fit	The fit is either the returned object from a kftrack fit or a kfsst fit
description	A description of the track can be added here
file	A filename where the output is written
npoints	Number of points in the polygon representation of the confidence ellipses
level	The confidence level

Author(s)

John Sibert <jsibert@soest.hawaii.edu>, Anders Nielsen <anielsen@dina.kvl.dk>

References

http://earth.google.com/

See Also

addmap and kftrack

Index