Reference manual for the ukfsst package

August 26, 2011

Title Kalman Filter tracking including Sea Surface Temperature

Version 0.3

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

Description Archival tagged marine creatures are typically geolocated based on observed light-levels. This package uses these raw geolocations and observed sea surface temperatures in a coherent state-space model to reconstruct the track via the extended Kalman filter.

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

Depends R (>= 1.8.0), date, kftrack

License BSD

R topics documented:

ukfsst-package . . . 2
blue.shark . . . 2
get.avhrr.sst . . . 3
get.blended.sst . . . 4
get.sst.from.server . . . 5
kfsst . . . 7
plot.kfsst . . . 10
plotmap . . . 11
print.kfsst . . . 12
road.map . . . 13

Index **14**

ukfsst-package *Unscented Kalman filter tracking of marine animals*

Description

Given a dataset of date, lon, lat, and tag-measured sea surface temperature this uses the unscented Kalman filter to estimate the most probable track of the tagged animal.

Details

Package:	ukfsst
Type:	Package
Version:	0.2
Date:	2006-12-11
License:	BSD

See the blue.shark example

Author(s)

Anders Nielsen <anielsen@dina.kvl.dk>, John Sibert <sibert@hawaii.edu>, and Chi Lam <chihinl@usc.edu>

See Also

road.map, link{blue.shark}

blue.shark *A track of a blue shark released near Hawai'i*

Description

This blue shark was released from longline fishing gear in April 2001. The PSAT tag from Microwave Telemetry (PTT-100) was attached and reported back after around 100 days at liberty.

Usage

```
data(blue.shark)
```

Format

A data frame with 45 observations on the following 6 variables.

`day` Integer giving the day of month

`month` Integer giving the month number

`year` Four digit integer giving the year

`Long` Raw light-based longitude in degrees east

`Lat` Raw light-based latitude in degrees north

`sst` Sea surface temperature (SST) derived from the tag in degrees Celsius.

Author(s)

Mike Musyl <Michael.Musyl@noaa.gov>, Anders Nielsen <anielsen@dina.kvl.dk>

Examples

```
data(blue.shark)
sst.path<-get.sst.from.server(blue.shark)
fit<-kfsst(blue.shark, bx.active=FALSE, bsst.active=FALSE)
plot(fit, ci=TRUE, pred=FALSE)
```

---

get.avhrr.sst *Get SST-field from avhrr source*

Description

This function allows easy access to a avhrr SST database.

Usage

```
get.avhrr.sst (track, folder = tempdir(),
               server = "http://coastwatch.pfeg.noaa.gov/coastwatch/CWBrowserWW360.
               product = "TAGssta", nday = "8day", centertime="12")
```

Arguments

`track`	A single `data.frame` containing a track or a `list` of `data.frames` each containing a track. The idea is that the function should only download the SST-data spanning the region and period of the tracks that needs to be analyzed.
`folder`	Is where the downloaded raw data files are stored. This defaults to a temporary directory.
`server`	The url of the server
`product`	The 7-character name of the imagery product. Refer to the Coastwatch site for the complete list of relevant SST products. http://coastwatch.pfeg.noaa.gov/coastwatch/CWBrowserWW360.jsp?get=gridData&dataSet=
`nday`	Time resolution should be either '5day' or '8day'
`centertime`	The time stamp of the image '00' is from midnight to midnight, and '12' is from noon to noon.

Value

The path returned from the function is where all the raw SST files are saved.

Author(s)

Anders Nielsen <anders.nielsen@hawaii.edu>, Chi Lam <chihinl@usc.edu>, Dave Foley <Dave.Foley@noaa.gov>

References

TALK TO DAVE FOLEY

See Also

get.sst.from.server, get.avhrr.sst

Examples

```
# No example supplied here
```

---

get.blended.sst *Get SST-field from blended source*

Description

This function allows easy access to a blended SST database.

Usage

```
get.blended.sst(track, folder = tempdir(),
  server = "http://coastwatch.pfeg.noaa.gov/coastwatch/CWBrowserWW360.jsp?get=gridD
  nday = "5day")
```

Arguments

track	A single data.frame containing a track or a list of data.frames each containing a track. The idea is that the function should only download the SST-data spanning the region and period of the tracks that needs to be analyzed.
folder	Is where the downloaded raw data files are stored. This defaults to a temporary directory.
server	the url of the server
nday	Time resolution should be either '5day' or '8day'

Value

The path returned from the function is where all the raw SST files are saved.

Author(s)

Anders Nielsen <anders.nielsen@hawaii.edu>, Chi Hin Lam <chihinl@usc.edu>, Dave Foley <Dave.Foley@noaa.gov>

References

TALK TO DAVE FOLEY

See Also

blue.shark, get.sst.from.server, get.avhrr.sst

Examples

```
# No example supplied here, but check out the example
# in the blue.shark dataset documentation
```

---

get.sst.from.server

Get SST-field from server

Description

This function allows easy access to three different SST sources that has been setup for this purpose. Data is downloaded from within R and stored in a format ready to be used.

Usage

```
get.sst.from.server(track, folder = tempdir(),
       server = "http://atlas.nmfs.hawaii.edu/cgi-bin/reynolds_extract.py")
```

Arguments

track	A single data.frame containing a track or a list of data.frames each containing a track. The idea is that the function should only download the SST-data spanning the region and period of the tracks that needs to be analyzed.
folder	Is where the downloaded raw data files are stored. This defaults to a temporary directory.
server	Presently three servers are available. The default is the fairly coarse Reynold's one 1x1-degree 8-day composites. This source is fast to download, but may be too coarse in some areas. The two other servers are the AVHRR-GAC 3-day and AVHRR-GAC 8-day composites these have a 0.1x0.1-degree resolution. The server names are: http://atlas.nmfs.hawaii.edu/cgi-bin/gac3day_extract.py and http://atlas.nmfs.hawaii.edu/cgi-bin/gac8day_extract.py

Details

The servers has been set up to extract SST-fields that covers a track (or a set of tracks) in simple way from within R. To use the default source type a command similar to:

sst.path <- get.sst.from.server(track1)

Notice 'track1' can be replaced by a list of tracks like:

sst.path <- get.sst.from.server(list(track1, track2))

to obtain an SST-field covering a set of tracks.

To use one of the two other servers simply supply the server name as in:

```
sst.path <- get.sst.from.server(list(track1, track2), server='http://atlas.nmfs.haw
bin/gac3day_extract.py')
```

or

```
sst.path <- get.sst.from.server(list(track1, track2), server='http://atlas.nmfs.haw
bin/gac8day_extract.py')
```

To use a user supplied SST-source please see documentation for the function ....

Value

The path returned from the function is where all the raw SST files are saved.

Author(s)

Anders Nielsen <anielsen@dina.kvl.dk>, Russell Moffitt <Russell.Moffitt@noaa.gov>

References

TALK TO RUSS

See Also

blue.shark

Examples

```
# No example supplied here, but check out the example
# in the blue.shark dataset documentation
```

`kfsst` *Kalman Filter based optimization of the tracking model including Sea Surface Temperature*

Description

After the track has been read, and SST data retrieved via the function `get.sst.from.server`, this function does the actual optimization of the model and reconstruction of the track.

Basically this function only needs the raw track as the reference to the SST-source is stored elsewhere. It has a lot of options to modify the model, which are presented here. Quite often it is necessary to simplify the model to get a converging fit - especially for short tracks.

Usage

```
kfsst(data, fix.first = TRUE, fix.last = TRUE,
      theta.active = c(u.active, v.active, D.active, bx.active,
      by.active, bsst.active, sx.active, sy.active, ssst.active,
      a0.active, b0.active, r.active),
      theta.init = c(u.init, v.init, D.init, bx.init, by.init,
      bsst.init, sx.init, sy.init, ssst.init, a0.init, b0.init,
      r.init),
      u.active = TRUE, v.active = TRUE, D.active = TRUE,
      bx.active = TRUE, by.active = TRUE, bsst.active = TRUE,
      sx.active = TRUE, sy.active = TRUE, ssst.active = TRUE,
      a0.active = TRUE, b0.active = TRUE, r.active=FALSE,
      u.init = 0, v.init = 0, D.init = 100, bx.init = 0,
      by.init = 0, bsst.init = 0, sx.init = 0.1, sy.init = 1,
      ssst.init = 0.1, a0.init = 0.001, b0.init = 0, r.init=200,
      var.struct = "solstice", save.dir = NULL, admb.string = "",
      from.ystr=c(3,6), from.dstr=c(7,9), to.ystr=c(11,14),
      to.dstr=c(15,17), localsstfolder=NULL)
```

Arguments

`data`	A data.frame consisting of six 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. The final column should contain the SST measurement derived from the tag on the fish.
`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 of eleven 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, bsst.active, sx.active, sy.active, ssst.active, a0.active, b0.active)`, hence a value of `c(0,0,1,1,1,1,1,1,1,1,1,` would result in a model where u and v were fixed at there initial values.

`theta.init` A numeric vector of eleven 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, bsst.init, sx.init, sy.init, ssst.init, a0.init, b0.init)` and the default value is `c(0, 0, 100, 0, 0, 0, 0.1, 1.0, 0.1, 0.001, 0, 2)`. It is unwise to initialize elements `D.init`, `sx.init`, `sy.init`, and `ssst.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.

`bsst.active` `TRUE` (default) if b_{sst} 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.

`ssst.active` `TRUE` (default) if σ_{sst} 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.

`r.active` If the radius is to be estimated from data. The flag should be set to `TRUE` if the radius should be optimized and `FALSE` (default) if it should be fixed at its initial value.

`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.
bsst.init	The initial value of b_{sst}. Default is 0.
sx.init	The initial value of σ_x. Default is 0.1.
sy.init	The initial value of σ_y. Default is 1.0.
ssst.init	The initial value of σ_{sst}. Default is 0.1.
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.
r.init	The initial value for the radius (in nautical miles) around each track point where the SST is to be used (the default is 200).
var.struct	Two options are available: `"uniform"`, `"solstice"`(default).
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)
from.ystr	Is an integer vector with two elements describing what part of the file name describe the year of the first date the data file represents. For instance if the names of the data files all have the format `RSyyyyddd_YYYYDDD.dat`, where `yyyy` is the year of the first date the argument should be `c(3,6)`.
from.dstr	Is an integer vector with two elements describing what part of the file name describe the 'number of days into the year' of the first date the data file represents.
to.ystr	Is similar to `from.ystr`, but here for the year of the last date the data file represents.
to.dstr	Is similar to `from.dstr`, but here for the 'number of days into the year' of the last date the data file represents.
localsstfolder	If the SST source is a bunch of files in a local folder this is where the folder name is given as a string

Details

Here should the model briefly be described, but for now read all about it in the reference given below.

Value

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

Author(s)

Anders Nielsen <anielsen@dina.kvl.dk>, John Sibert <sibert@hawaii.edu>, and Chi Lam <chihinl@usc.edu>

See Also

road.map, link{blue.shark}

Examples

```
# No example supplied here, but check out the example
# in the blue.shark dataset documentation
```

---

plot.kfsst *Plot a fit from kfsst.*

Description

Plots four figures based on the reconstructed track. One plot of the reconstructed track (possibly on top of a map, but it requires GMT to be installed also see: http://gmt.soest.hawaii.edu/. Three additional plots illustrating how well each of the observed coordinates matches the fitted track.)

Usage

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

Arguments

x	is a kfsst object typically generated with the kfsst 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 GMT is correctly installed) a GMT-based postscript file with the track saved in the working directory
...	additional graphics parameters

Details

Besides the arguments described above there is one other option if GMT is installed on the system. If map=TRUE is specified the drawing of the track is overlaid on a simple land/water map.

Value

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

Author(s)

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

See Also

kfsst, blue.shark

Examples

```
# No example supplied here, but check out the example
# in the blue.shark dataset documentation
```

---

plotmap *Plot land area on a map with colored polygons*

Description

Plots a map within given rectangular region showing land areas as colored polygons. Requires the mapping utility GMT.

Usage

```
plotmap(x1, x2, y1, y2, resolution=3,
        grid=FALSE, add=FALSE, save=FALSE,
        landcolor="darkgreen", seacolor="lightblue",
        zoom=FALSE)
```

Arguments

x1	Longitude of lower left corner of rectangle
x2	Longitude of upper right corner of rectangle
y1	Latitude of lower left corner of rectangle
y2	Latitude of upper right corner of rectangle
resolution	Map resolution, integer: 1 (highly detailed) to 5 (crude)
grid	Whether to plot grid lines on map
add	Whether to add polygons to an existing plot
save	Whether to return matrix of polygons
landcolor	Color of polygons
seacolor	Color of ocean
zoom	Whether to start in interactive zoom mode

Details

A map is plotted with polygons clipped at borders of map region.

If the function is started in zoom mode two left-clicks on the map will zoom it to the rectangle spanned by the two points. This zooming is repeated until a right-click on the map is done.

Value

Value is `NULL` unless `save` is `TRUE`, in which case a 2-column matrix is returned containing latitude and longitude coordinates of the polygon vertices. Polygons are separated by *NA*s in both columns.

Author(s)

Pierre Kleiber, and Anders Nielsen `<anders.nielsen@hawaii.edu>`

See Also

`kfsst`

Examples

```
#This function requires GMT to be installed. If you have it try typing:
#  plotmap(8,13,53,58,res=1,zoom=TRUE)
```

`print.kfsst` *Print kfsst object*

Description

Prints a pretty summary of an object of class `kfsst`

Usage

```
print.kfsst(x, ...)
```

Arguments

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

Details

Notice also that below the printing of estimates and standard deviations are a list of names. Each of these refer to an object that is also stored in the fitted object. For instance to get the negative log likelihood value type `fit$nlogL`, and to get the number of parameters type `fit$npar`, where `fit` is the name of the object returned from the function `kfsst`.

Author(s)

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

See Also

kfsst

---

road.map *Prints the road map of running kfsst for a track*

Description

This is a pure documentation function. Once invoked it outlines the steps needed to run the SST aided tracking model.

Usage

```
road.map()
```

Value

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

Author(s)

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

See Also

kfsst, blue.shark

Index

∗Topic **datasets**
blue.shark, 2
∗Topic **models**
get.avhrr.sst, 3
get.blended.sst, 4
get.sst.from.server, 5
kfsst, 7
plot.kfsst, 10
plotmap, 11
print.kfsst, 12
road.map, 13
∗Topic **package**
ukfsst-package, 2

blue.shark, 2, *5, 6*, *11*, *13*

get.avhrr.sst, 3, *4, 5*
get.blended.sst, 4
get.sst.from.server, *4*, 5, *5*, *7*

kfsst, 7, *10–13*

plot.kfsst, 10
plotmap, 11
print.kfsst, 12

road.map, *2*, *10*, 13

ukfsst *(ukfsst-package)*, 2
ukfsst-package, 2