

# Reference manual for the KFSST package

August 9, 2006

**Title** Kalman Filter tracking including Sea Surface Temperature

**Version** 0.1

**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>

**License** BSD

## R topics documented:

blue.shark . . . . .	1
get.sst.from.server . . . . .	2
kfsst-internal . . . . .	3
kfsst-package . . . . .	4
kfsst . . . . .	4
plot.kfsst . . . . .	7
plotmap . . . . .	8
print.kfsst . . . . .	9
road.map . . . . .	9
write.sst.field . . . . .	10
<b>Index</b>	<b>12</b>

---

blue.shark	<i>A track of a blue shark released near Hawai'i</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.

**Examples**

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

---

```
get.sst.from.server
```

*Get SST-field from server*

---

**Description**

This function allows easy access to a fairly coarse SST database that has been setup for this purpose. Data is downloaded from within R and stored in a format ready to be post-processed.

**Usage**

```
get.sst.from.server(track, folder = tempdir(), server = "http://atlas.nmfs.hawaii.edu/cgi-bin/gac3day_extract.py")
```

**Arguments**

<code>track</code>	A single <code>data.frame</code> containing a track or a list of <code>data.frames</code> 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.
<code>folder</code>	Is where the downloaded raw data files are stored. This defaults to a temporary directory.
<code>server</code>	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: <a href="http://atlas.nmfs.hawaii.edu/cgi-bin/gac3day_extract.py">http://atlas.nmfs.hawaii.edu/cgi-bin/gac3day_extract.py</a> and <a href="http://atlas.nmfs.hawaii.edu/cgi-bin/gac8day_extract.py">http://atlas.nmfs.hawaii.edu/cgi-bin/gac8day_extract.py</a>

**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.  
bin/gac3day_extract.py')
```

or

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

To use a user supplied SST-source please see documentation for the function `write.sst.field`.

**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**

`write.sst.field`, `blue.shark`

**Examples**

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

---

kfsst-internal      *Internal kfsst objects*

---

**Description**

Internal kfsst objects.

**Details**

These are not to be called by the user.

---

kfsst-package

*Kalman Filter tracking including Sea Surface Temperature*


---

## Description

Kalman Filter tracking including Sea Surface Temperature

## Details

Package: kfsst  
 Type: Package  
 Version: 0.1  
 Date: 2006-06-19  
 License: BSD

Please read the [kfsst](#) helpfile.

## Author(s)

Anders Nielsen [⟨anielsen@dina.kvl.dk⟩](mailto:anielsen@dina.kvl.dk), John Sibert [⟨sibert@hawaii.edu⟩](mailto:sibert@hawaii.edu)

## References

Nielsen, A., Bigelow, K. A., Musyl, M. K., and Sibert, J. R. (2006). Improving light-based geolocation by including sea surface temperature. *Fisheries Oceanography*, 15(4), 314-325.

## See Also

[road.map](#), [link{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 in and the SST-field has been constructed (see the `road.map` function for advice) this function does the actual optimization of the model and reconstruction of the track.

Basically this function only needs the raw track and the SST file, but 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, sst.file = "sst.dat", fix.first = TRUE, fix.last = TRUE,
      theta.active = c(u.active, v.active, D.active, bx.active, by.active, bsst.active),
      theta.init = c(u.init, v.init, D.init, bx.init, by.init, bsst.init, sx.init, sy.init),
      u.active = TRUE, v.active = TRUE, D.active = TRUE, bx.active = TRUE, by.active = TRUE,
      sx.active = TRUE, sy.active = TRUE, ssst.active = TRUE, a0.active = TRUE,
      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,
      var.struct = "solstice", dev.pen = 0, save.dir = NULL, admb.string = "")
```

**Arguments**

<code>data</code>	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.
<code>sst.file</code>	A string containing the filename (including path) to a file returned from the function <code>write.sst.field</code> .
<code>fix.first</code>	TRUE (default) if the first position in the data set is the true release position (known without error), FALSE otherwise.
<code>fix.last</code>	TRUE (default) if the last position in the data set is the true recapture/popoff position (known without error), FALSE otherwise.
<code>theta.active</code>	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 <code>c(u.active, v.active, D.active, bx.active, by.active, bsst.active, sx.active, sy.active, ssst.active, a0.active, b0.active)</code> , hence a value of <code>c(0,0,1,1,1,1,1,1,1,1,1)</code> would result in a model where $u$ and $v$ were fixed at their initial values.
<code>theta.init</code>	A numeric vector of eleven elements, each corresponding to a model parameter. The order of the elements in this vector is <code>c(u.init, v.init, D.init, bx.init, by.init, bsst.init, sx.init, sy.init, ssst.init, a0.init, b0.init)</code> and the default value is <code>c(0, 0, 100, 0, 0, 0, 0.1, 1.0, 0.1, 0.001, 0)</code> . It is unwise to initialize elements <code>D.init</code> , <code>sx.init</code> , <code>sy.init</code> , and <code>ssst.init</code> below zero, as they correspond to standard deviations.
<code>u.active</code>	TRUE (default) if $u$ should be optimized, FALSE if it should be fixed at its initial value.
<code>v.active</code>	TRUE (default) if $v$ should be optimized, FALSE if it should be fixed at its initial value.
<code>D.active</code>	TRUE (default) if $D$ should be optimized, FALSE if it should be fixed at its initial value.
<code>bx.active</code>	TRUE (default) if $b_x$ should be optimized, FALSE if it should be fixed at its initial value.
<code>by.active</code>	TRUE (default) if $b_y$ should be optimized, FALSE if it should be fixed at its initial value.

<code>bsst.active</code>	TRUE (default) if $b_{sst}$ should be optimized, FALSE if it should be fixed at its initial value.
<code>sx.active</code>	TRUE (default) if $\sigma_x$ should be optimized, FALSE if it should be fixed at its initial value.
<code>sy.active</code>	TRUE (default) if $\sigma_y$ should be optimized, FALSE if it should be fixed at its initial value.
<code>ssst.active</code>	TRUE (default) if $\sigma_{sst}$ should be optimized, FALSE if it should be fixed at its initial value.
<code>a0.active</code>	If the variance structure <code>var.struct="solstice"</code> 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.
<code>b0.active</code>	If the variance structure <code>var.struct="solstice"</code> 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.
<code>u.init</code>	The initial value of $u$ . Default is 0.
<code>v.init</code>	The initial value of $v$ . Default is 0.
<code>D.init</code>	The initial value of $D$ . Default is 100.
<code>bx.init</code>	The initial value of $b_x$ . Default is 0.
<code>by.init</code>	The initial value of $b_y$ . Default is 0.
<code>bsst.init</code>	The initial value of $b_{sst}$ . Default is 0.
<code>sx.init</code>	The initial value of $\sigma_x$ . Default is 0.1.
<code>sy.init</code>	The initial value of $\sigma_y$ . Default is 1.0.
<code>ssst.init</code>	The initial value of $\sigma_{sst}$ . Default is 0.1.
<code>a0.init</code>	If the variance structure <code>var.struct="solstice"</code> is chosen this sets the initial value of $a_0$ . Default is 0.001. If a different variance structure is selected this is ignored.
<code>b0.init</code>	If the variance structure <code>var.struct="solstice"</code> is chosen this sets the initial value of $b_0$ . Default is 0. If a different variance structure is selected this is ignored.
<code>var.struct</code>	Three options are available: "uniform", "solstice"(default), and "daily".
<code>dev.pen</code>	If <code>var.struct="daily"</code> is set, this parameter sets the derivative penalty.
<code>save.dir</code>	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.
<code>admb.string</code>	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 <a href="http://otter-rsch.com">http://otter-rsch.com</a> )

## 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 `kftrack` 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>

**References**

<http://www.tracking.nielsensweb.org>

**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**

<code>x</code>	is a <code>kfsst</code> object typically generated with the <code>kfsst</code> function
<code>ci</code>	If TRUE adds confidence regions for the most probable track to the plot
<code>points</code>	If FALSE the raw geo-locations are omitted
<code>pred</code>	If FALSE the predicted plot is omitted
<code>most</code>	If FALSE the most probable track is omitted
<code>gmt</code>	If TRUE (and GMT is correctly installed) a GMT-based postscript file with the track saved in the working directory
<code>...</code>	additional graphics parameters

**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 NAs 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	<i>Print kfsst object</i>
-------------	---------------------------

---

**Description**

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

**Usage**

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

**Arguments**

<code>x</code>	an object of class <code>kfsst</code> typically generated with the <a href="#">kfsst</a> function.
<code>...</code>	additional arguments

**Author(s)**

Anders Nielsen (anders.nielsen@hawaii.edu), John Sibert (sibert@hawaii.edu)

**See Also**

[kfsst](#)

---

road.map	<i>Prints the road map of running kfsst for a track</i>
----------	---

---

**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](#)

---

`write.sst.field`      *Writes a smoothed version of the SST-field to a file*

---

**Description**

Starting from a directory (`datadir`) of data files each containing a grid of latitude, longitude, and corresponding SST measurements, this function reads the files one by one and spatially smooths the SST-field, computes the gradient of the smooth field, and write these informations to a combined file (`filename`). The original data files each represent the SST-field at a specific time interval (say a week). The time information is contained in the name of each file.

**Usage**

```
write.sst.field(datadir, nlon = 100, nlat = 150, filename = "sst.dat", alpha = 0
  from.ystr = c(3, 6), from.dstr = c(7, 9), to.ystr = c(11, 14), to.dstr = c(15,
  peak = FALSE)
```

**Arguments**

<code>datadir</code>	Is the folder where the raw data files are located
<code>nlon</code>	The smoothed SST-fields and their gradient-fields are represented on a <code>nlon</code> -by- <code>nlat</code> grid
<code>nlat</code>	The smoothed SST-fields and their gradient-fields are represented on a <code>nlon</code> -by- <code>nlat</code> grid
<code>filename</code>	Is the name of the file where the smoothed fields are saved
<code>alpha</code>	Is a scalar between 0 and 1 determining the degree of smoothing used. The default is 5%, which means that the smoothed field at any point is calculated from the 5% nearest points in the observed data.
<code>from.ystr</code>	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 <code>RSyyyyddd_YYYYDDD.dat</code> , where <code>yyyy</code> is the year of the first date the argument should be <code>c(3, 6)</code> .
<code>from.dstr</code>	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.
<code>to.ystr</code>	Is similar to <code>from.ystr</code> , but here for the year of the last date the data file represents.
<code>to.dstr</code>	Is similar to <code>from.dstr</code> , but here for the 'number of days into the year' of the last date the data file represents.
<code>peak</code>	If TRUE allows to visually compare the raw and the smoothed field.

**Details**

The grid size of the internal representation can be finer or coarser than the actual data set, and should be chosen based on size of the area. This way of representing the SST-field is clearly sub-optimal, and will hopefully be replaced in later versions.

The default smoothing scale of 5% is probably coarse in many cases, especially if the area is large.

It is recommended to carry out sensitive analysis with respect to the degree of smoothing and the grid size of the internal representation.

The smoothing is presently done via the `locfit` R-package.

**Value**

The filename returned from the function is where the internal representation is saved.

**Author(s)**

Anders Nielsen <anielsen@dina.kvl.dk>, 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
```

# Index

## \*Topic **datasets**

blue.shark, 1

## \*Topic **internal**

kfsst-internal, 3

## \*Topic **models**

get.sst.from.server, 2

kfsst, 4

kfsst-package, 3

plot.kfsst, 7

plotmap, 8

print.kfsst, 9

road.map, 9

write.sst.field, 10

## \*Topic **package**

kfsst-package, 3

.First(*kfsst-internal*), 3

.generate.dat.file.kfsst  
(*kfsst-internal*), 3

.plotsst(*kfsst-internal*), 3

.read.output.kfsst  
(*kfsst-internal*), 3

blue.shark, 1, 3, 7, 9, 11

get.sst.from.server, 2

kfsst, 4, 4, 7-9, 11

kfsst-internal, 3

kfsst-package, 3

plot.kfsst, 7

plotmap, 8

print.kfsst, 9

road.map, 4, 6, 9

write.sst.field, 3, 4, 10