Light-level geolocation analyses

Last edited on 2018-11-13

Contents

Preface **5**

Acknowledgements 6

License 6

1 Structure of the manual **7**

The datasets 7

Reproducing the analyses 8

2 Getting started **9**

3 Loading data **11**

4 Twilight Annotation **13**

Cleaning/Filtering twilight times 21

5 GeoLight **23**

6 probGLS **25**

7 SGAT **27**

8 FLightR **29**

9 Data repositories **31**

References **33**

Preface

Note: The Manual is currently under development and content may not show up (ask Simeon if you need immediate access)!

This manual is part of the following publication and has been written by the same group of authors:

Simeon Lisovski, Silke Bauer, Martins Briedis, Kiran Danjahl-Adams, Sarah Davidson, Christoph Meier, Lykke Pedersen, Julia, Karagicheva, Benjamin Merkel, Janne Ouwehand, Michael T. Hallworth, Eldar Rakhimberdiev, Michael Sumner, Caz Taylor, Simon Wotherspoon, Eli Bridge (201X) The Nuts and Bolts of Light-Level Geolocation Analyses. Journal X:xxx-xxx.

Geolocation by light is a method of animal tracking that uses small, light-detecting data loggers (referred to as geolocators) to determine the locations of animals based on the light environment they move through.

Technological and fieldwork issues aside, effective use of light level geolocation requires translation of a time series of light levels into geographical locations. Geographical locations that are derived from light-level data are subject to error which directly arises from noise in the light-level data, i.e. unpredictable shading of the light sensor due to weather or the habitat (Lisovski et al., 2012). Although light-level geolocation has provided a wealth of new insights into the annual movements of hundreds of bird species and other taxa, researchers struggle with the analytical steps that are needed to obtain location estimates, interpret them, present their results, and document what they have done.

This manual has been written by some of the leading experts in geolocator analysis and is based on material created for several international training workshops. It offers code and experience that we have accumulated over the last decade, and we hope that this collection of analysis using different open source software tools (R packages) helps both newcomers and experienced users of light-level geolocation.

Acknowledgements

We want to acknowledge all people that have been involved in the development of geolocator tools as well as all participants of the many international geolocator workshops. Furthermore, we like to acknowledge Steffen Hahn and Felix Liechti organisewho organised a first workshop of the analysis of geolocator data from songbirds back in 2011. This workshop has been financially supported by the Swiss Ornithological Institute and the Swiss National Science Foundation. The National Centre for Ecological Analysis and Synthesis (NCEAS) has supported two meetings with experts in geolocator analysis in 2012 ans 2013 and many of the tools that are discussed in this manual were kick started at these meetings. We want to thank James Fox from Migrate Technology Ltd. as well as the US National Science Foundation for contiouing financial support to develop tools and organise workshops.

License

Chapter 1

Structure of the manual

This manual should allow users with limited knowledge in R coding to perform a state-of-the-art analysis of geolocator data. Thus, we start with the very basics of loading packages and data 3 Starting with the initial data editing steps, which we call twilight annotation 4, we provide instructions on how to use several prominent analysis packages, illustrate the general analysis workflow using example data, and provide some recommendations for how to visualize and present results. We do not cover every available analysis package but focus on what we percieve to be the most frequently used tools, which are GeoLight 5, probGLS 6, SGAT 7 and FLightR 8. The manual concludes with a section on data repositories such as Movebank that allows storing and shring geolocator tracks 9.

The datasets

To illustrate the capabilities of the different packages, discuss the potential pitfalls, and provide some recommendations, we will use raw geolocator data from four individuals of different species. All used tag data and the results as well as the code for the analyses has been uploaded onto Movebank unter study: xxxx.

TagID	Species	Folder	Tag type
M034	Red-backed Shrike	LanCol	Integio (Migrate Technology Ltd.)
14SA	European bee-eater	MerApi	PAM (Swiss Ornithological Institute)
PasCir01	Purple martin	PasCir	Custom (by Eli Bridge)
2655	Brünnich's guillemot	UriLom	Lotek

Although all of these tag types record light values over time, they differ in some key details. First, tags often differ in the frequency at which they write/log data. Many tags collect a reading every minute and store the maximal light value every 5 or 10 minutes. Other may store a maximum every 2 minutes. The tag that yielded the Purple martin data set, averaged 1min readings every 10min instead of taking a maximum. These four tags also differ in their sensitivity and how they record light levels. Some tags are sensitive only at low light levels and quickly "max out" when they experience a lot of light. As such, their light-levels do not have units and are simply an index of light intensity. The Integio tags can record unique light values for all natural light levels on earth, and they store lux values that range from 0 to ~70,000. Depending on the tag type, you may have to perform some preliminary steps such as log-transforming your data or time shifting light values for sunsets (we will provide details while working on the specific datasets).

Reproducing the analyses

This manual contains code that can be copy pasted into an R script and executed to reproduce the results. In order to do so, you need to download the raw data as well as annotated twilight files used in this manual. The data need to be in a specific structure of folders and we do recommend you have a similar structure for your own analysis. During the processing of the data we save intermediate steps that allow us to step into the next analysis step without going through all initial and often time consuming parts. Having your raw data and your results in a well structured fomr, becomes especially important if you run analyses for many tags of the same or different species. It is also recommended that you create a single R script for each analysis (e.g. for each individual and each analysis using different tools). For example, you can name the R scripts using the tag id and the tool e.g. `14SA_SGAT.R`. Since this manual is dealing with tags from different species, the following structure with sub-folders per speces (first three letters of the genus name and the species name) is setup within the main folder (called *data*):

- RawData
  - LanCol
  - MerApi
  - PasCir
- Results
  - LanCol
  - MerApi
  - PasCir
- RCode
  - LanCol
  - MerApi
  - PasCir

You can download the folders with the raw data as well as the annotaded twilight files directly via R and extract into a *data* folder.

```
url <- "https://github.com/slisovski/TheGeolocationManual/raw/master/download/data.zip"

temp <- tempfile()
download.file(url, temp)
unzip(temp, exdir = "data")
```

We also recommend using R Studio and creating a project (File -> NewProject). Alternatively, you can set the working directory using the `setwd` function. With the *data* folder in your project folder (or more in general in your working directory) you should be able to run the code provided in this manual.

We also recommend to use *R Studio* and to create a project (File -> NewProject). Save the project file into the existing *Data* folder. This makes sure that *Data* is your working directory and it will remain the working directory even if the folder moves around on your drive. Alternatively, you can set the working directory using the `setwd` function. With the suggested folder structure and the raw data and the annotaded twiligth files you should be able to run the code provided in this manual.

Chapter 2

Getting started

To analyse light-level geolocator data in R we need a couple of R packages as well as functions that allow to run our code. We created a package called *GeoLocTools* that contains functions that are not nessesarily associated to a certain package but are used in this manual. Importantly the package can also run a check on you system (function: *setupGeolocation()*), detecting packages that are already on your computer and installs the missing tools directly from CRAN or GitHub.

The package requires *devtools* (install if nessesary using the *install.packages()* function). With *devtools* on your system, you are able to download and built as well as install R packages directly from GitHub (e.g. *GeoLocTools*).

```
library(devtools)
install_github("SLisovski/GeoLocTools")
```

You should now be able to load the package and run the `setupGeolocation()` function. We recommend to include this line at the beginning of each script you create for a geolocator analysis. Also check (every now and then), if there is a new version of *GeoLocTools* available. And if that is the case, re-install the package using the same code you used for initial installation.

```
library(GeoLocTools)
setupGeolocation()
```

if you see "You are all set!" in your console, the function ran succesfully and you are able to proceed.

Amongst dependencies, the following geolocator specific packages are loaded by this function:

- twGeos
- GeoLight
- probGLS
- SGAT
- FLightR

- **What the $#@%!#!!!** Although the *GeoLocTools* should make things much easier, it is quite common for problems to arise when setting up your environment. A few frequent and frustrating issues are:
- **Outdated version of R**. If you are not running the latest (or at least a recent) version of R, then some of the packages might not be compatible. Use *sessionInfo()* to see what version of R you are running. You can ususally track down the latest version of R at the R project webpage: www.r-project.org.

Note that you may have to reinstall all of your packages when you get a new version of R. So expect to spend a few minutes on the update.)

- **Missing libraries**. Some packages require that you have specific sofware libraries installed an accessible on your system. if you get a message like "configure: error: geos-config not found or not executable," you may be missing a library. Dealing with these issues may require some use of the Bash or Unix shell to install or locate a library. You can often find instructions for intalling new libraries by searching the internet, but if you do not feel comfortable installing stuff with the command line or you do not have permission to do so, you will probably need to seek some assistance from someone with IT credentials.

- **Typos**. Probably the most common error in R arises simply from typos. Even published scripts or manuals like these may contain small typos that prevent your script from running.

Chapter 3

Loading data

The first step is to load your raw data into R. Different geolocator types (e.g. from different manufacturers or different series) provide raw data in different formats. And while there are functions available to read a whole range of formats, you may have to either write your own function, use simple read text utilities or get in touch with the package managers to write code that fits your format if it is not yet implemented.

The most frequently used geolocators provide files with the extension `.lux` (Migrate Technology Ltd), `.lig` (BAS, Biotrack) or `.glf` (Swiss Ornithological Institute). The functions `readMTlux`, `ligTrans` and `glfTrans` allows you to read these files. The documentations of the different packages may help to provide information on how to read other files (e.g. ?GeoLight). In most cases the raw data is stored in a text file that can also be read in to R using the base function `read.table()`.

A short note on ***naming and saving of data files*** (final results and intermediate steps): We have already discussed, that it makes sense to have a certain folder structure for the analysis of geolocators. It not only helps to keep track of all files and analysis, but most importantly it allows to run the same code for saving and reading of data once you defined a set of metadata information.

With the suggested data structure, we can then define metadata information on the individual, the species, the deployment location, and define the sub-folder for saving and extracting data files.

```
ID <- "14SA"
Species <- "MerApi"

lon.calib <- 11.96
lat.calib <- 51.32

wd <- "data"
```

By using the above metadata we can use the `paste0` command to include this information in reading and writing of files.

```
raw <- glfTrans(paste0(wd, "/RawData/", Species, "/", ID, ".glf"))
  names(raw) <- c("Date", "Light")
  raw$Light  <- log(raw$Light+0.0001) + abs(min(log(raw$Light+0.0001)))
head(raw)
```

```
                 Date Light
1 2015-07-10 00:00:00     0
2 2015-07-10 00:05:00     0
3 2015-07-10 00:10:00     0
4 2015-07-10 00:15:00     0
5 2015-07-10 00:20:00     0
6 2015-07-10 00:25:00     0
```

In this case it is required log transform the light data. In addition, we add a small value since the night readings are sometimes smaller than zero, values that cannot be log transformed.

Adding to the confusion of different raw data types, the read functions also provide different output. However, the most important columns are,

1. Date
2. Light

and these columns need to be in a specific format with Date being a `POSIXc` class and Light being `numeric` integers. Check if the structure of your data follows the required format with the function `str`. If not adjust Date format with `as.POSIXct(raw$Date, tz = "GMT")`.

```
str(raw)
```

```
'data.frame':   112161 obs. of  2 variables:
 $ Date : POSIXct, format: "2015-07-10 00:00:00" "2015-07-10 00:05:00" ...
 $ Light: num  0 0 0 0 0 0 0 0 0 0 ...
```

Do I need to log-transform my raw light measurements?

Log-transformation of the light intensities is helpful to visualise and inspect the data and for the twilight annotation process. It allows to focus at the low light values while seeing the whole light curve and thus makes sense for the tags that measure the full light spectrum (e.g. tags from Migrate Technology Ltd. and from the Swiss Ornithological Institute). If you proceed to analyse your data with FLightR, where you need the raw light intensities, there is no need to back-transform you light data as FLightR will do that automatically.

Chapter 4

Twilight Annotation

There are a few options for how to define and edit twilights.

All tools discussed in this manual require as one of their inputs a data frame containing the times of sunrise and sunset (henceforth twilight events) for the duration of the study period. The twilight events are estimated based on a light-level threshold, which is the light value that separates day from night - values above the threshold indicate the sun has risen and values below the threshold value indicate the sun has set. There are a few options for how to generate the twilight data. `twilightCalc` is one function that allows transitions to be defined and is part of the GeoLight package. Given the much better realisation of this process in TwGeos, we will not discuss the GeoLight version of defining twilights. `TwGeos` provides an easier to use and more interactive process that is called `preprocessLight`. An important input, besides the raw data, is a pre-defined light intensity threshold value.

How do I choose the right threshold?

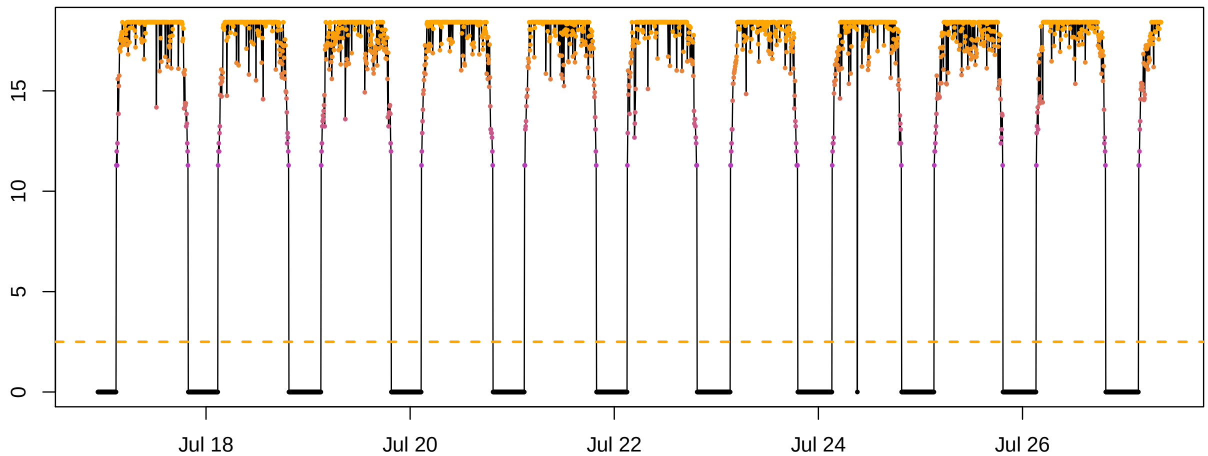
How do I know which threshold to use: You should choose the lowest value that is consistently above any noise in the nighttime light levels. Here, we use a threshold of 2.5, that is above any nighttime noise. However, this value is tag and species specific. For forest interior, ground dwelling species a lower threshold may be helpful, especially if there isn't much 'noise' during the night. A threshold of 1 may be appropriate for such species.

It is a good idea to plot (parts) of the dataset and see how the threshold fits into the light recordings:

```
threshold <- 2.5

col = colorRampPalette(c('black',"purple",'orange'))(50)[as.numeric(cut(raw[2000:5000,2],breaks = 50))]

par(mfrow = c(1, 1), mar = c(2, 2, 2, 2) )
with(raw[2000:5000,], plot(Date, Light, type = "o", pch=16,  col = col, cex = 0.5))
abline(h=threshold, col="orange", lty = 2, lwd = 2)
```

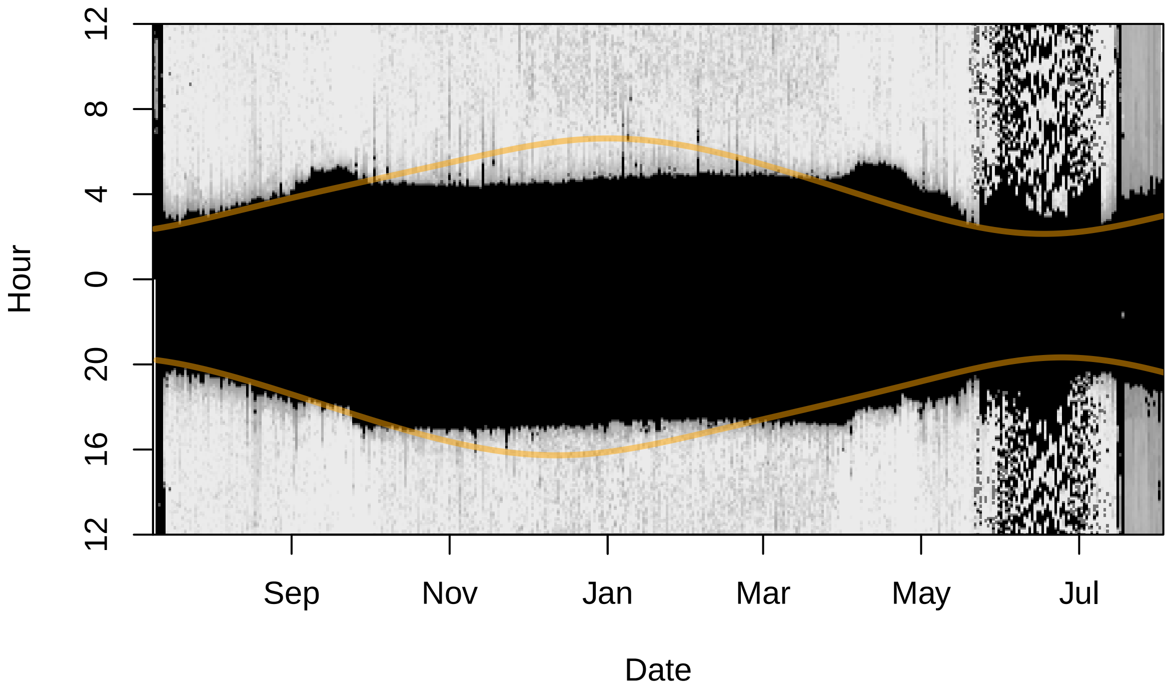


Another useful plot can be created using lightImage; In the resulting figure, each day is represented by a thin horizontal line that plots the light values as grayscale pixels (dark = low light and white = maximum light) in order from bottom to top. a A light image allows you to visualize an entire data set at once, and easily spot discrepancies in light to dark transitions. Additionally, you can add the sunrise and sunset times of the deployment or retrieval locaitons (using `addTwilightLine`). This may help to spot inconsistncies in the dataset, e.g.: *time shifts - resulting in a good overlap of twilight times at the beginning but a systematic shift between expected and recorded twilight times.* false time zone - if the predicted sunrise and sunset times are shifted up- or downwards it is highly likely that your raw data is not recorded (or has been transformed) in GMT (or UTC). Check with producer or data provider. Furthermore, the lines can help to identify the approximate timing of departure and arrival to the known deployment or retrieval site and this may help to identify calibration periods that are required in the next steps of the analysis.

```
offset <- 12 # adjusts the y-axis to put night (dark shades) in the middle

lightImage( tagdata = raw, # light data
  offset = offset,
  zlim = c(0, 20)) # y axis

tsimageDeploymentLines(raw$Date, lon = lon.calib, lat = lat.calib,
                       offset = offset, lwd = 3, col = adjustcolor("orange", alpha.f = 0.5))
```



Depending on the tag type, geolocator data are automatically adjusted for clock drift by the manufacturer, or, can be easily corrected by comparing the internal device time and real time when data is downloaded. For practical reasons, clock drift in geolocators is assumed to occur at a constant rate. If geolocator data are affected by clock drift the longitude estimates during stationary periods will drift continuously in one direction. In case the tag had stopped recording before data download or the internal time stamp is obviously incorrect, clock drift can be adjusted during the process of locations estimation. In short, an estimated clock drift is added to the twilight data and longitudinal positions are (re)calculated, e.g. using a best-guess sun elevation angle. If there are no directional changes in longitude during the stationary period anymore (the slope of a linear regression would be zero for longitude data plotted over time), clock drift is adequately corrected for. Latitude estimates are negligibly affected due to the small difference in shifting sunrise and sunset times within the same day.

In the next step, we want to define daily sunrise and sunset times. `preprocessLight` is an interactive function for editing light data and deriving these twilight times Note: if you are working on a Mac you must install Quartz first (https://www.xquartz.org) and then set gr.Device to “x11” in the function. If you are working with a virtual machine, the function may not work at all. Detailed instructions of how to complete the interactive process can be found by running the following code:

```
?preprocessLight
```

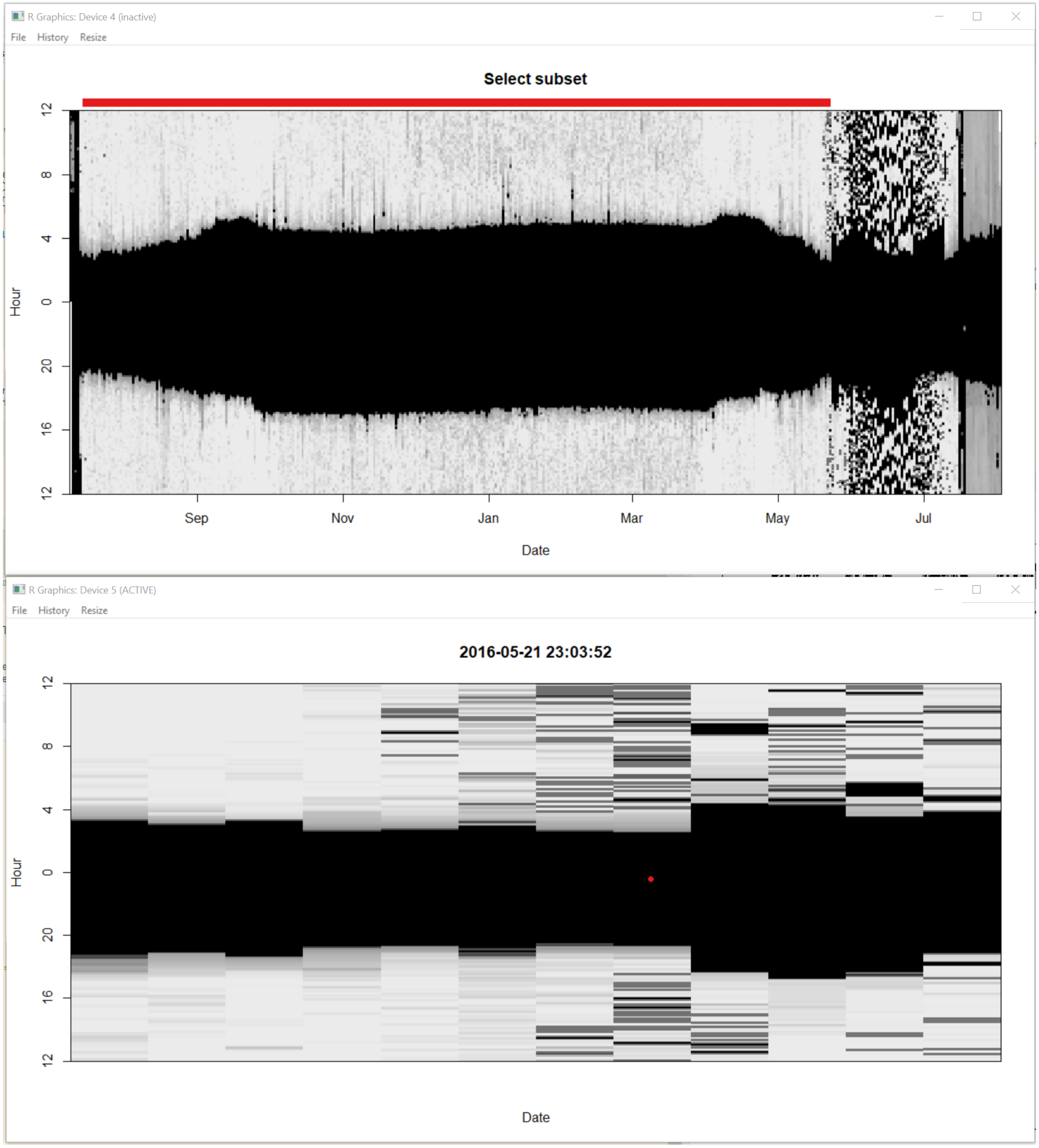
Below, we explain the major functionalities.

When you run,

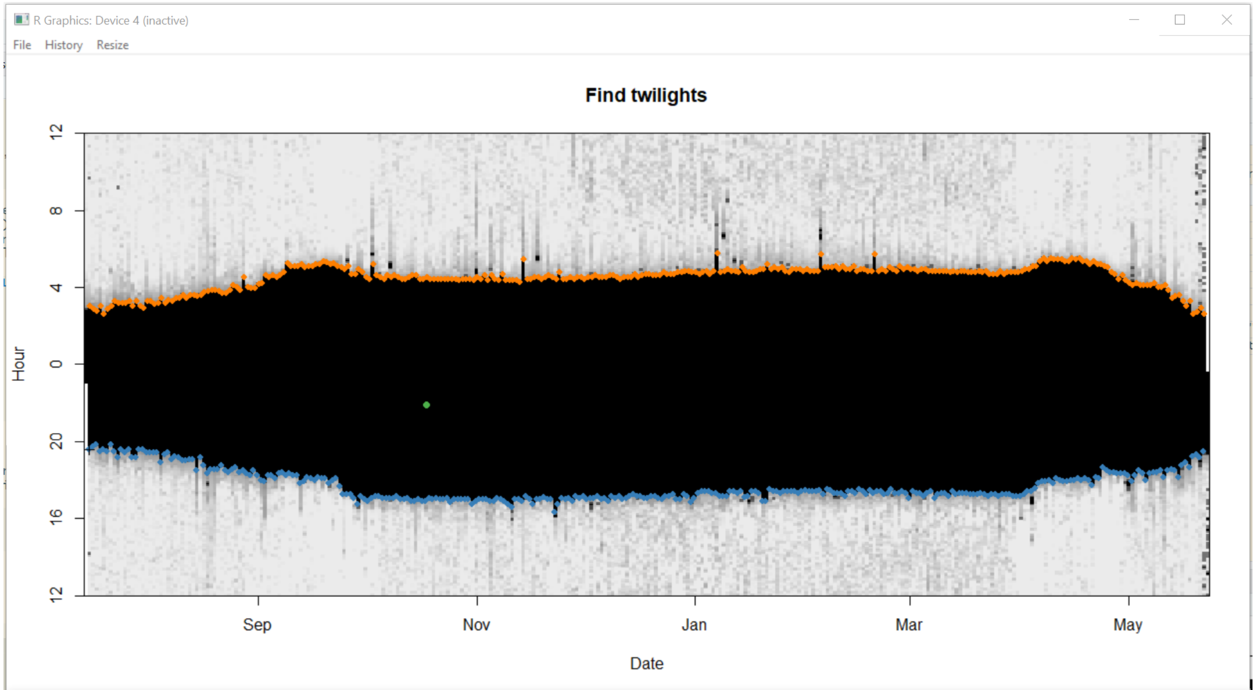
```
twl <- preprocessLight(raw,
  threshold = threshold,
  offset = offset,
  lmax = 20, # max. light value (adjust if contrast between night and day is weak)
  gr.Device = "x11") # x11 works on a mac (if Quarz has been installed and works on most Windows machin
```

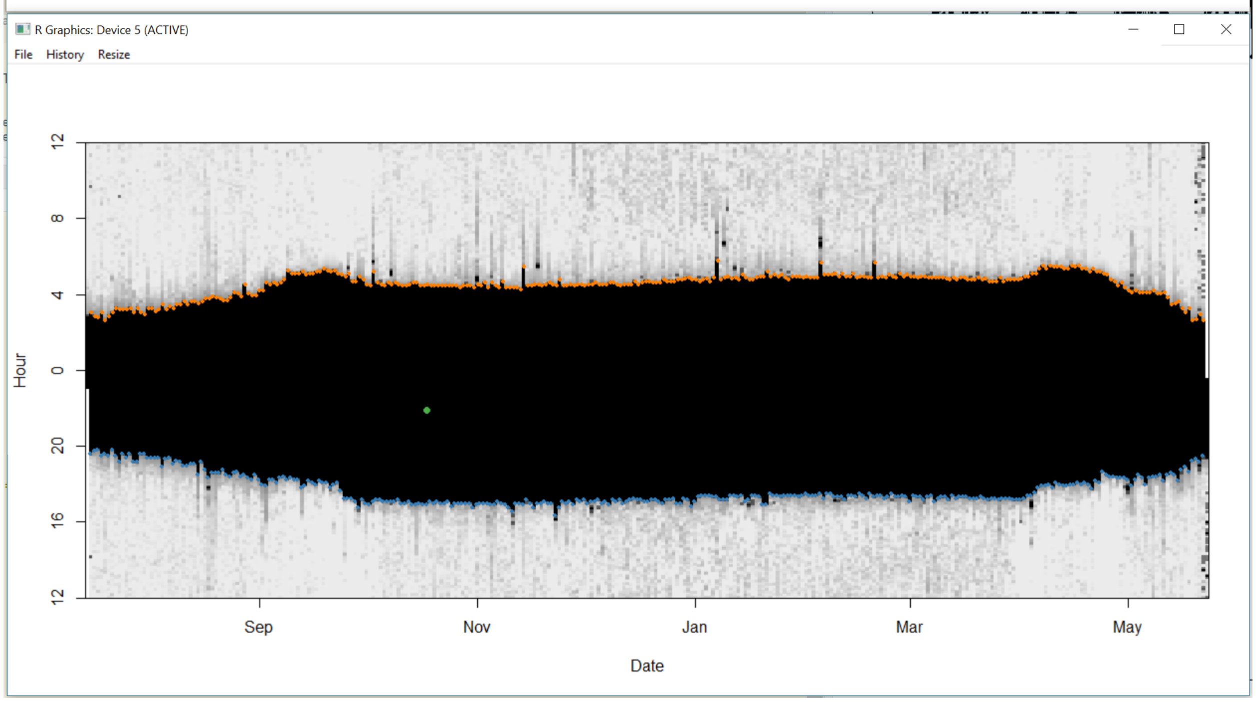
two windows will appear. Move them so they are not on top of each other and you can see both. They should look like a big black blob. This identifies the "nightime" period over time. The top of the blob shows all the sunrises and the bottom of blob shows all the sunsets. You can note for instance that the days get longer (and thus the nights shorter) at the end of the time series, because the blob gets thinner. You may even note changes in the light image that relate to changes in activity patterns or breeding behavior.

Step 1. Click on the window entitled "Select subset". With the left mouse button choose where you want the start of the dataset to be, and right mouse button to choose the end. You will notice that the red bar at the top moves and that the second window zooms into that time period. Select when you want your time series to start and end. This allows you to ignore for instance periods of nesting. Once you are happy with the start and end of the timeseries press "a" on the keyboard to accept and move to next step.



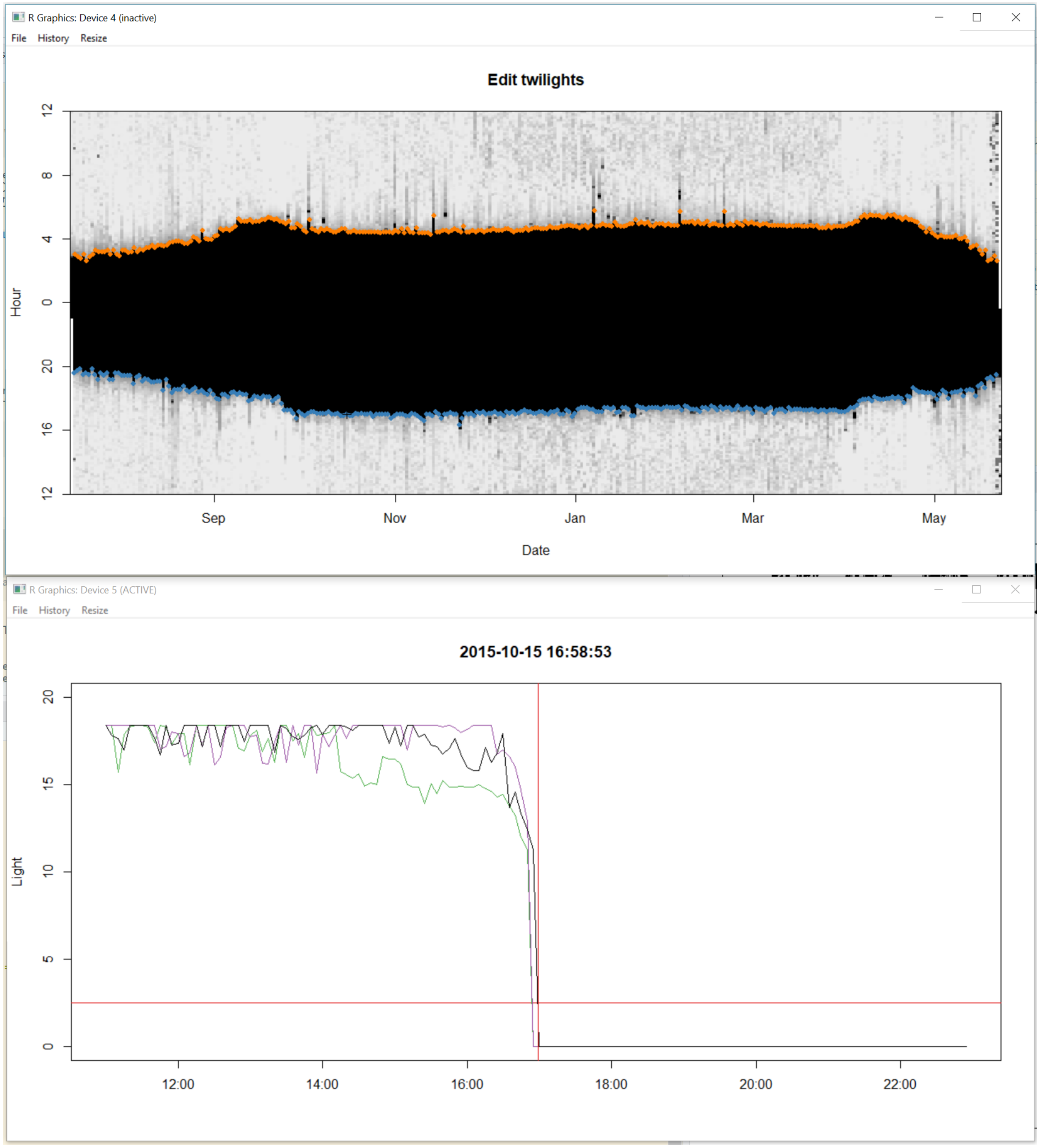
Step 2. click on the window entitled "Find twilights" and the second window will zoom in. All you need to do here is click in the dark part (in the zoomed in image i.e. the one not entitled "Find twilights") of the image and this will identify all the sunrises (orange) and sunsets (blue) based on the threshold defined in the previous section. Press "a" on the keyboard to accept and move to next step.





Step 3. This step is for adding or deleting points. If there are no missing data points, you can skip this step by pressing "a" on the keyboard. However, if you do want to add a point, you can click on the "Insert twilights" window to select a region of "the blob" that the second unintitled window will zoom into. In the zoomed window, use left mouse click to add a sunrise, and right mouse click to add a sunset. You can use "u" on the keyboard to undo any changes, and "d" to delete any points which are extra. Press "a" to move to next step.

Step 4. This step allows you to find points which have been miss-classified (often because the bird was in the shade or in a burrow) and to move the respective sunrise or sunset to where it should be. Choose a point by clicking on it in the "edit twilights" window and the other window will display the sunrise (or sunset) from the previous and next days (purple and green) relative to the current sunrise or sunset (in black). Thus if the black line shows a much earlier sunset or later sunrise than the purple and green ones, it is likely badly classified. . You can then left click at the point where you want the day to start and press “a” to accept and move the sunrise or sunset. You will notice the red line then moves. Do this for as many points as necessary.



Then close the windows with “q”.

How important is it to edit twilights?

If you have no a priori reason and criteria to strongly edit twilight events, it is generally better to be a bit conservative with editing. This prevents that data are changed into an unwanted direction, e.g. erroneously removing good data points (amidst shading events), or informative events such as strong movements. Also the criteria to edit or remove badly classified twilights will be different depending on the method you use to infer locations. For curve methods, similarity in the shape of the curve around sunrise or sunset is most important, while for threshold methods the similarity in the sunrise and sunset events itself is important.

Save the output file as a .csv file, so that you never have to do this step again.

Have a look at the output

```r
head(twl)
```

```
             Twilight  Rise Deleted Marker Inserted           Twilight3
1 2015-07-15 19:34:02 FALSE   FALSE      0    FALSE 2015-07-15 19:34:02
2 2015-07-16 03:01:00  TRUE   FALSE      0    FALSE 2015-07-16 03:01:00
3 2015-07-16 19:43:53 FALSE   FALSE      0    FALSE 2015-07-16 19:43:53
4 2015-07-17 02:51:06  TRUE   FALSE      0    FALSE 2015-07-17 02:51:06
5 2015-07-17 19:48:53 FALSE   FALSE      0    FALSE 2015-07-17 19:48:53
6 2015-07-18 02:46:06  TRUE   FALSE      0    FALSE 2015-07-18 02:46:06
  Marker3
1       0
2       0
3       0
4       0
5       0
6       0
```

The output contains the following important information:

- **Twilight**
- The date and time of the sunrise/sunset events
- **Rise**
- whether the Twilight is a sunrise (TRUE) or a sunset (FALSE)
- **Deleted**
- whether you marked this twilight with a “d”, that means it is still in the file and can/should be exluded later on.
- Marker (see detailed description in `?preprocessLight`)
- Inserted (whether this Twilight was manually inserted)
- Twilight3 (the original Twilight. Only different to Twilight if you edited the timing)

Other processes like `twilightCalc` or the software `TAGS` produce different outputs but it is preferred to get them into this format (at least with the columns `Twilight` and `Rise`), since you can go ahead with any analysis you want using these two columns (***note: do not save these two columns only, since the other information is important to reproduce your analysis***).

To save this file we use the metadata variables that were defined above:

```
write.csv(twl, paste0(wd, "/Results/", Species, "/", ID, "_twl.csv"), row.names = F)
```

This can later be loaded using the following code (note, that you have to define the class type `POSIXC` for the date):

```
twl <- read.csv(paste0(wd, "/Results/", Species, "/", ID, "_twl.csv"))
twl$Twilight <- as.POSIXct(twl$Twilight, tz = "GMT") # get the Twilight times back into the POSIX. clas
```

The result of this first part that is **independent** of which package/analysis will be used next is the twilight file that should at least look like (can have more columns):

```
head(twl[,c(1,2)])
```

```
             Twilight  Rise
1 2015-07-15 19:34:02 FALSE
2 2015-07-16 03:01:00  TRUE
3 2015-07-16 19:43:53 FALSE
4 2015-07-17 02:51:06  TRUE
5 2015-07-17 19:48:53 FALSE
6 2015-07-18 02:46:06  TRUE
```

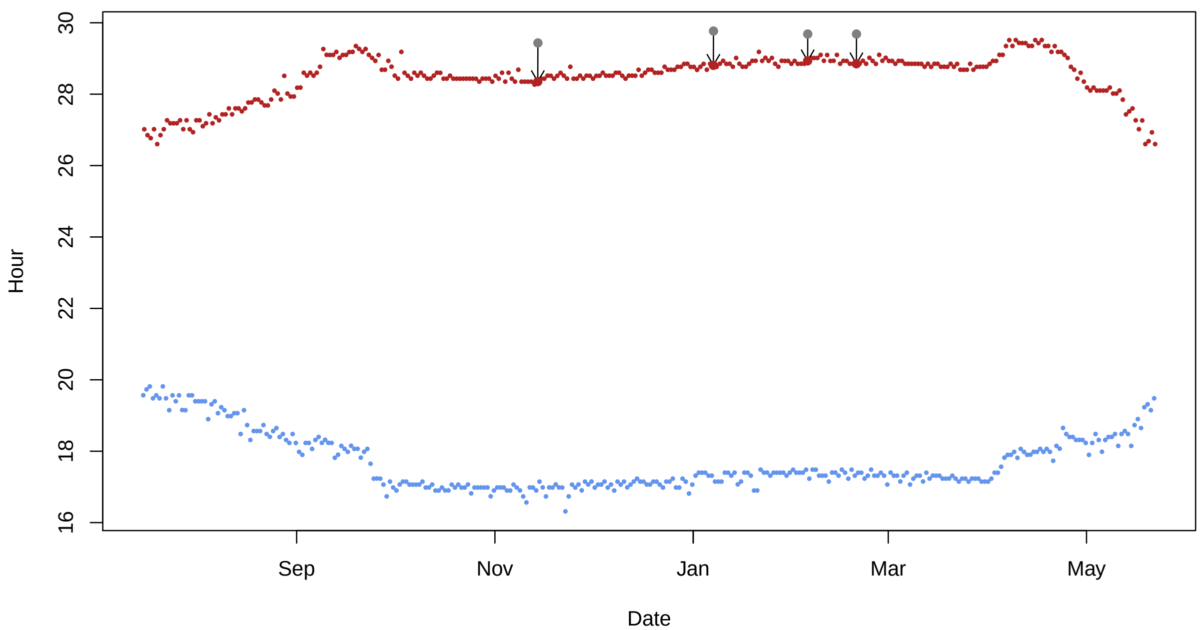
Cleaning/Filtering twilight times

Automated filtering of twilight times should be handled carefully. There is no *perfect* function that cleans your twilight file. However, `twilightEdit` can help to filter and remove (mark them as deleted) outliers (e.g. false twilights). The filtering and removing of twilight times is based on a set of rules:

1) if a twilight time is e.g. 45 minutes (**outlier.mins**) different to its surrounding twilight times, and these sourrounding twilight times are within a certain range of minutes (**stationary.mins**), then the twilight times will be adjusted to the median of the surrounding twilights.
2) if a twilight time is e.g. 45 minutes (**outlier.mins**) different to its surrounding twilight times, but the surrounding twilight times are more variable then you would expect them to be if they were recorded during stationary behavior, then the twilight time will be marked as deleted.

The argument **windows** defines the number of twilight times surrounding the twilight in focus (e.g. same as in conventional moving window methods).

```
twl <- twilightEdit(twilights = twl,
                    offset = offset,
                    window = 4,           # two days before and two days after
                    outlier.mins = 45,    # difference in mins
                    stationary.mins = 25, # are the other surrounding twilights within 25 mins of one a
                    plot = TRUE)
```



In this particular case and with the parameters, four twilight times have been corrected. Based on the output, you can also exclude them for further analysis. While you can also save the output file, we recommend archiving the twilight file from above and redo the `twilightEdit` after reading in the archived twilight file from above.

This method helps to adjust and remove twilight times that are either outliers or false twilights given a set of rules. While subjective to a certain degree as well as reproducible, the method may not be able to detect all false twilight times and may even remove correct entries during fast migration periods.

Chapter 5

GeoLight

...

Chapter 6

probGLS

...

Chapter 7

SGAT

...

Chapter 8

FLightR

...

Chapter 9

Data repositories

...

References

Ekstrom, P. (2004). An advance in geolocation by light. Memoirs of the National Insitute of Polar Research, Special Issue, 58, 210–226. Ekstrom, P. (2007). Error measures for template-fit geolocation based on light. Deep Sea Research Part II: Topical Studies in Oceanography, 54, 392–403. Lisovski, S., Hahn, S. (2012a). GeoLight - processing and analysing light-based geolocator data in R. Methods in Ecology and Evolution, 3, 1055–1059. Lisovski, S., Hewson, C.M., Klaassen, R.H.G., Korner-Nievergelt, F., Kristensen, M.W. & Hahn, S. (2012b). Geolocation by light: accuracy and precision affected by environmental factors. Methods in Ecology and Evolution, 3, 603–612. Pedersen L., et. al. Rakhimberdiev, E., Winkler, D.W., Bridge, E., Seavy, N.E., Sheldon, D., Piersma, T. & Saveliev, A. (2015). A hidden Markov model for reconstructing animal paths from solar geolocation loggers using templates for light intensity. Movement Ecology, 3, 25. Rakhimberdiev, E., Senner, N. R., Verhoeven, M. A., Winkler, D. W., Bouten, W. and Piersma T. (2016) Comparing inferences of solar geolocation data against high-precision GPS data: annual movements of a double-tagged Black-Tailed Godwit. Journal of Avian Biology 47: 589-596. Rakhimberdiev, E., Saveliev, A., Piersma, T., & Karagicheva, J. (2017). FLightR: An R package for reconstructing animal paths from solar geolocation loggers. Methods in Ecology and Evolution, 8(11), 1482-1487.

Bibliography

Lisovski, S., Hewson, C. M., Klaassen, R. H. G., Korner-Nievergelt, F., Kristensen, M. W., and Hahn, S. (2012). Geolocation by light: accuracy and precision affected by environmental factors. *Methods in Ecology and Evolution*, 3(3):603–612.