Journal of Avian Biology

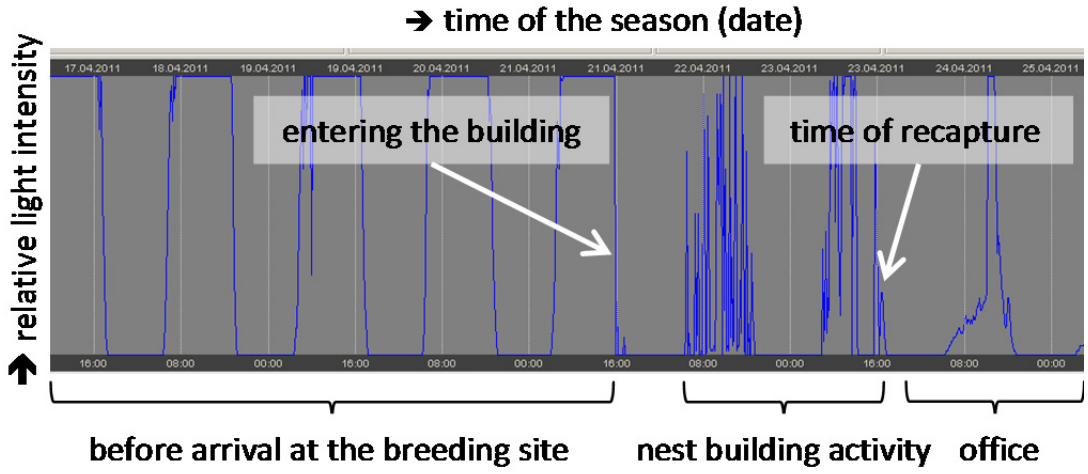
JAV-00485

Liechti, F., Scandolara, C., Rubolini, D., Ambrosini, F., Korner-Nievergelt, F., Hahn, S., Lardelli, R., Romano, M., Caprioli, M., Romano, A., Sicurella, B. and Saino, N. 2014. Timing of migration and residence areas during the non-breeding period of barn swallows *Hirundo rustica* in relation to sex and population. – J. Avian Biol. doi: 10.1111/jav.00485

Supplementary material

Appendix 1. Detection of departure and arrival at the breeding site

Figure A1. Example of a light curve recorded by the geolocator (SOI GDL2.11). The blue lines represent the light intensity recorded. Natural day- and nighttime can be clearly recognized during the period before arrival (to the right). We assumed that the abrupt drop in light intensity on the 21.4.2011 marks the first visit to the nest site. The next two days are characterized by many artificial shading most probably due to visits to the nest. In the afternoon of the 23.4.2011 the bird was recaptured. On the 24.4.2011 the logger was laying in the office.



Appendix 2. Sex differences in the effects of the position of the sub-Saharan residence area (SRP) on phenology

We investigated whether the SRP latitude differentially affected the phenology of non-breeding events of either sex. To this end, we added to the models shown in Table 3 the statistical interaction between sex and SRP latitude, respectively. These analyses were ran only on the 'reduced' dataset, because the inclusion of the five southern wintering males strongly increased model multicollinearity (VIF > 25, details not shown) when the interaction terms were tested. Moreover, even in this reduced dataset, we could not test the interaction term between sex and longitude, again because its inclusion (either separately or simultaneously with the sex × latitude effect) raised model multicollinearity beyond acceptable levels (VIF > 10). SRP latitude did not differentially predict phenology of non-breeding events of either sex (sex × latitude effect; duration of stay in the wintering range, $F_{1,58} = 0.73$, $p = 0.39$; departure from the wintering range, $F_{1,60} = 0.33$, $p = 0.57$; duration of spring migration, $F_{1,55} = 0.58$, $p = 0.45$; arrival to the breeding colony, $F_{1,55} = 0.05$, $p = 0.82$).

Appendix 3. Comparison of the positions of the sub-Saharan residence areas (SRPs) between years, sexes and populations using a randomisation test

The median SRP of the 59 individuals tracked in 2010-2011 was 7.0° N - 12.8° E, while for the 33 individuals tracked in 2011-2012 it was 3.2° N - 15.3° E. The median SRP in 2011-2012 was thus 506 km to the SE compared to the previous year.

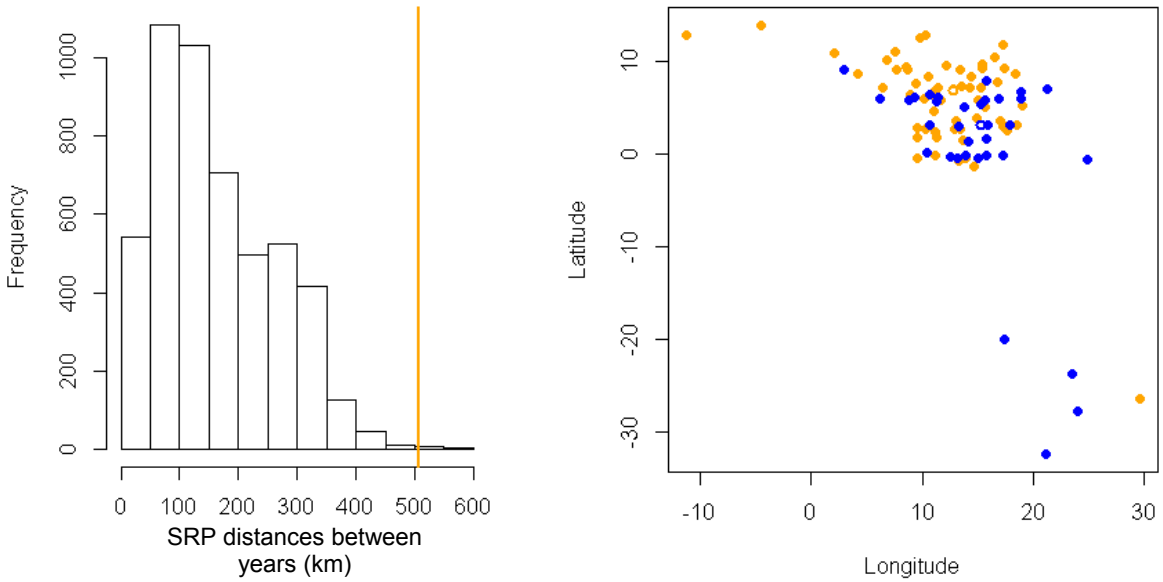
When assigning the year randomly to the individuals repeatedly (4999 iterations), the median distance between the SRP of the two years was 139 km. The observed distance of the median SRP between the two years (506 km) was significantly higher than expected by chance ($p = 0.002$, Fig. A2).

For comparing the SRP between the sexes and populations using the same randomisation procedure (i.e. assigning sexes and populations randomly to individuals, respectively; see Table A1), we therefore corrected for the between-year difference by shifting the locations of 2011 by 506 km to the SE, so that the median SRP of the two years coincided.

Table A1. Comparison of SRP median distances between sexes and populations using a randomisation test (see details above). For the six groups (3 populations x 2 sexes) differences between the median SRPs (km) of each group and p-values for differences from the randomisation test are given (M = males, F = females). There is no statistically significant difference in the median SRP between any group (all $p > 0.06$).

Group	SE-pop F		N-pop M		N-pop F		SW-pop M		SW-pop F	
	km	p	km	p	km	p	km	p	km	p
SE-pop M	535	0.291	429	0.156	413	0.240	456	0.135	632	0.065
SE-pop F			360	0.532	732	0.086	377	0.479	564	0.273
N-pop M					410	0.128	30	0.980	244	0.613
N-pop F							413	0.133	435	0.239
SW-pop M									215	0.700

Figure A2. Left: distribution of randomised distances between median SRPs when years were assigned at random (n = 4999); orange line: observed distance between median SRPs of the two years (506 km). Right: SRP of the individuals in 2010-2011 (orange) and 2011-2012 (blue); open circles = median location.



Appendix 4: Comparison of sex-specific variances between the phenological events during the non-breeding period

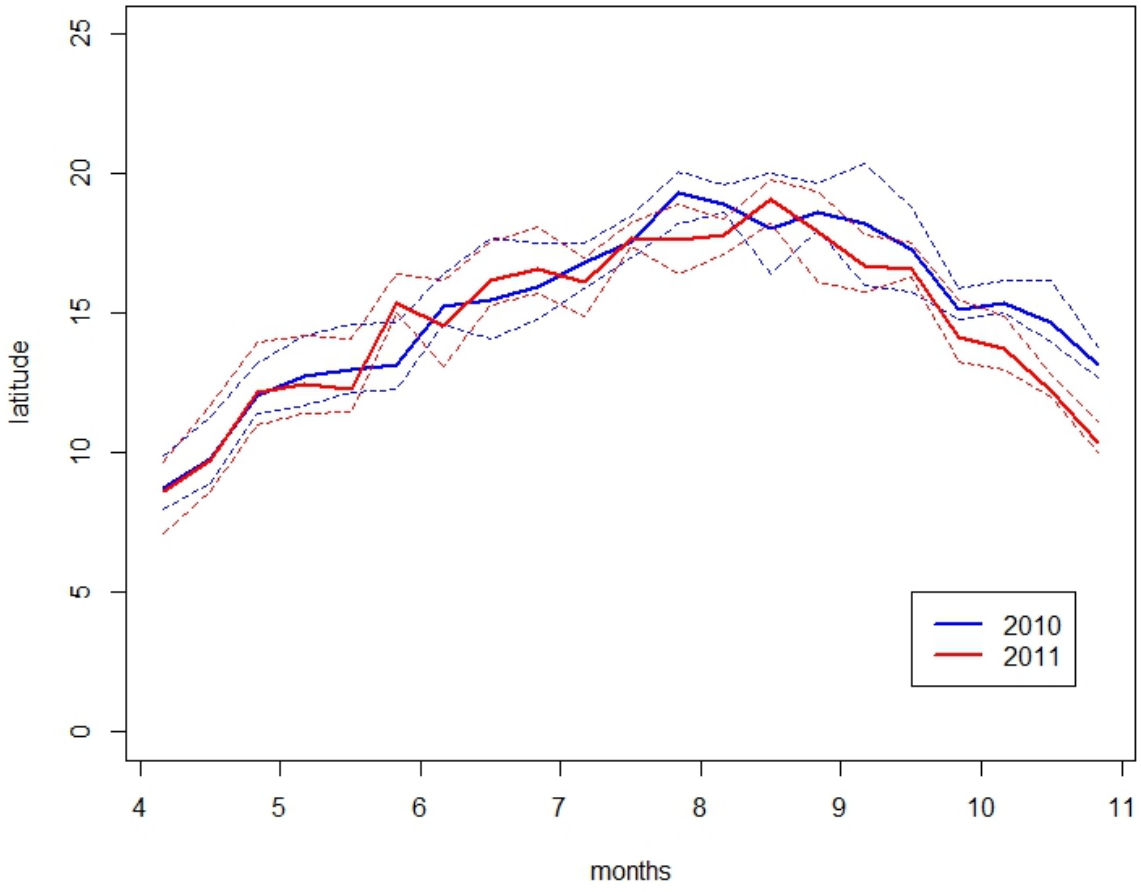
For each phenological event table A2 shows the variance and results of Levene's test for differences between the sexes. Below the results (and R-code) of ANOVA for differences in variance in between sexes and between sexes for each phenological event are given.

Table A2. Sex-specific variances in the departure times from and the arrival times at the breeding grounds and sub-Saharan residence areas. The table shows the variance of the residuals from ANOVAs fitted to sex-specific data using year, population and their interaction as explanatory variables. The Levene's test for pairwise comparisons of variances between the sexes is shown.

Event	Males variance	Females variance	Levene's test
Departure from the breeding colony	19.8	58.6	$F_{1,99} = 6.11$, $p = 0.015$
Arrival to the sub-Saharan residence area	73.5	61.7	$F_{1,90} = 0.03$, $p = 0.87$
Departure from the the sub-Saharan residence area	132.4	198.3	$F_{1,69} = 0.72$, $p = 0.40$
Arrival to the breeding colony	128.7	123.7	$F_{1,64} = 0.02$, $p = 0.89$

Appendix 5. Variation in weather conditions between the two study years

Figure A3. Movement of the intertropical convergence zone (ITCZ) in sub-Saharan Africa from April to October in the two study years (2010, 2011). Shown are the course of the mean (solid line) and the range (dashed lines) of the ITCZ latitude for the longitudinal range 2.5° – 22.5° E. Data from CPC/NOAA (http://cpc.ncep.noaa.gov/products/fews/ITCZ/itcz.shtml).



Appendix 6. Distribution of stationary sites of birds with multiple non-breeding ranges

Figure A4. Distribution of stationary sites south of the Sahara (<23.5°N) of individuals with multiple non-breeding range. For each individual (n = 22) the position of the mode of the kernel densities (see Methods) (dots) and the 90% range in longitude and latitude (crossing lines) are given. Only stopover period of at least 2 weeks are shown. Stationary sites from the same individual are connected by a line. The colours refer to the breeding area(see legend). The four graphs are grouped according to year (top to bottom) and sex (left – right).

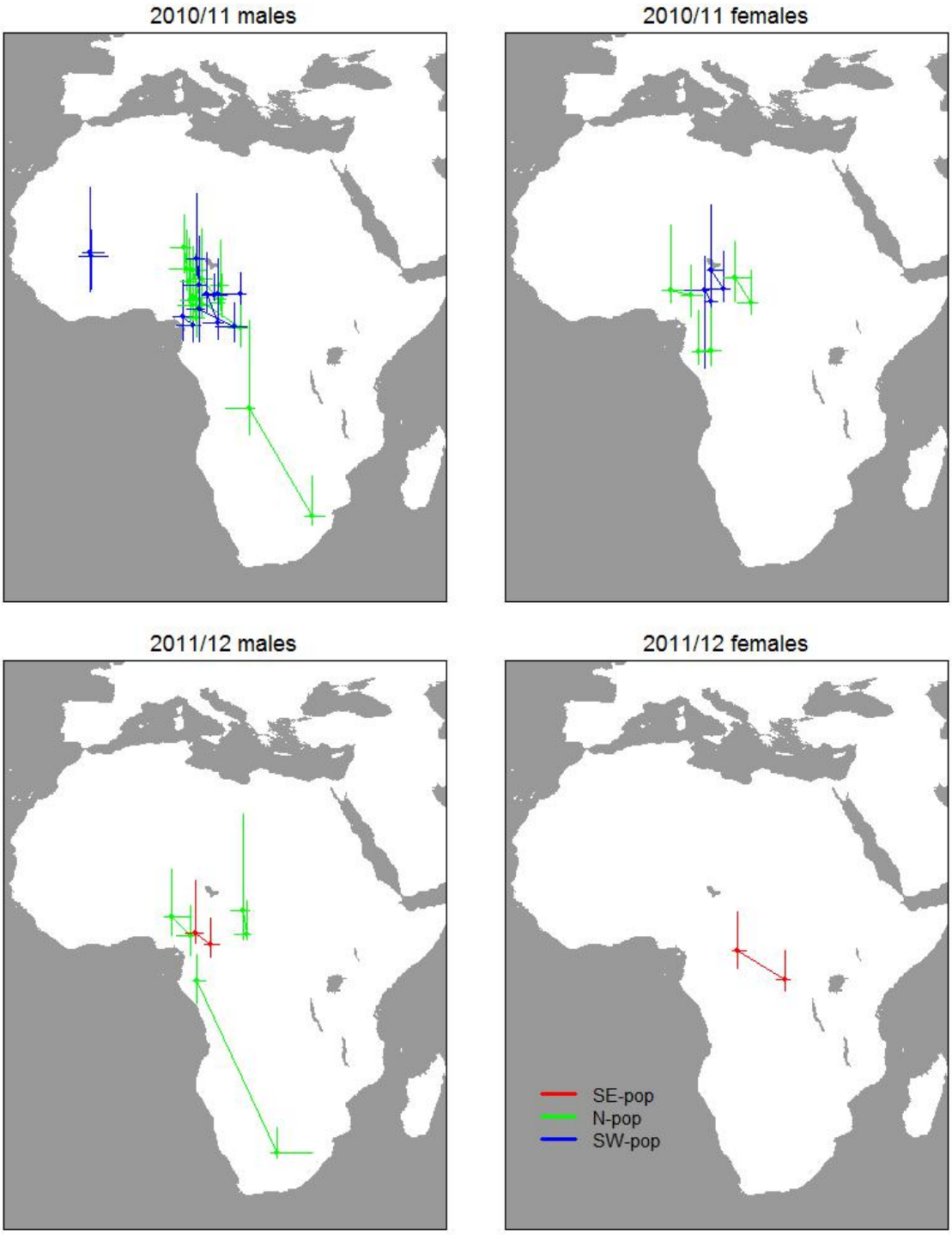
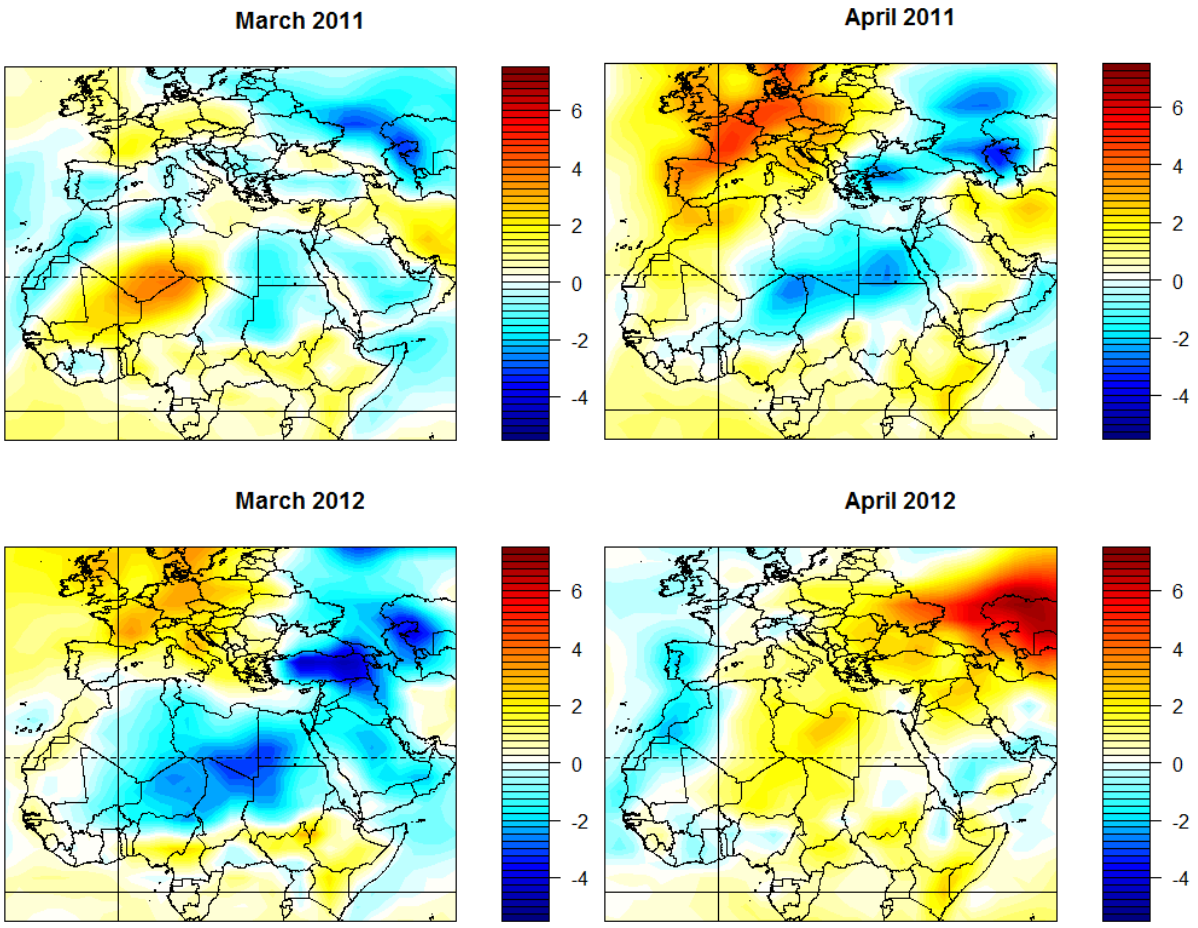


Figure A5. March-April temperature anomalies across the barn swallow wintering and migration range in the two years of study. Anomalies are expressed as the deviation (in °C) from the long-term (1970-2000) monthly mean values. Original data were gridded on a 2.5 x 2.5° grid and were downloaded from the NCEP/NCAR Reanalysis dataset provided by the NOAA/OAR/ESRL PSD, Boulder, Colorado, USA, from their Web site (http://www.esrl.noaa.gov/psd/).



Appendix 7. R-code for the data analysis of light-level geolocators

```
#-------------------------------------------------------------------------------------------
# Analysis of light-level logger data for SOI-GDL2.10 & SOI-GDL2.11
# by F. Korner-Nievergelt & F. Liechti
# August 2013
# R 3.0.0 (2013-04-03)
#-------------------------------------------------------------------------------------------
#-------------------------------------------------------------------------------------------
# contents (every part can be run in its own)
#-------------------------------------------------------------------------------------------
# 0) libraries and functions
# 1) data input and definition of weights for each sun-event according to the running-maximum-method
#     point (3 and 4) in description of methods
# 1a) identify stationary periods using change-point for events with weight > 0
#     point (5 and 6) in description of methods
# 2) merging of stationary periods according to overlap of the kernels
#     point (7) in description of methods
# -------------------------------------------------------------------------------------------
# 0) load packages and define functions
#  -------------------------------------------------------------------------------------------
library(ks)
library(birdring)
library(geosphere)
#-------------------------------------------------------------------------------------------
# own functions
ownmode <- function(x, w=NULL) {
# mode of x
# x: vector of numerics
# w: vector of plain (!) numbers of weights   -> does not work yet...
if(is.null(w)){
  if(sum(!is.na(x))>1){
  x <- x[!is.na(x)]
  xdens <- density(x)
  xmax <- xdens$x[xdens$y==max(xdens$y, na.rm=TRUE)]
  }
  if(sum(!is.na(x))<2) xmax <- NA
  }
if(!is.null(w)){
  x <- rep(x,w)
  if(sum(!is.na(x))>1){
  x <- x[!is.na(x)]
  xdens <- density(x)
  xmax <- xdens$x[xdens$y==max(xdens$y, na.rm=TRUE)]
  }
  if(sum(!is.na(x))<2) xmax <- NA
  }
  return(mean(xmax))
  }

insidekernel <- function(x,y, kernelobj, percentage=95){
# looks up whether a point (x,y) is inside a specific kernel density isoline
            kritdensity <- contourLevels(kernelobj, prob=1-percentage/100)
            densitymat <- matrix(kernelobj$estimate, ncol=length(kernelobj$eval.points[[1]]),
nrow=length(kernelobj$eval.points[[2]]), byrow=FALSE)
            nearestx <- kernelobj$eval.points[[1]][abs(kernelobj$eval.points[[1]]-
x)==min(abs(kernelobj$eval.points[[1]]-x))][1]
            nearesty <- kernelobj$eval.points[[2]][abs(kernelobj$eval.points[[2]]-
y)==min(abs(kernelobj$eval.points[[2]]-y))][1]
```

```
            insidek <- densitymat[kernelobj$eval.points[[1]]==nearestx,
kernelobj$eval.points[[2]]==nearesty]>=kritdensity
            return(insidek)
            }
#-------------------------------------------------------------------------------------------
# set working directory
setwd("path/to/folder/") # insert path to folder with geolocator light data files
#-------------------------------------------------------------------------------------------
# 1) calculating the weights
#-------------------------------------------------------------------------------------------
# read sun event data
dat <- read.table("filename.csv", sep=";", header=TRUE)
str(dat) # should look similar to this one:

# 'data.frame':  12324 obs. of  9 variables:
# $ tFirst : Factor w/ 10253 levels "2010-08-20 04:41:00",..: 1 2 3 4 5 6 7 8 9 10 ...
# $ tSecond: Factor w/ 10256 levels "2010-08-20 18:42:00",..: 1 2 3 4 5 6 7 8 9 10 ...
# $ type   : int  1 2 1 2 1 2 1 2 1 2 ...
# $ site   : int  0 0 0 0 0 0 0 0 0 0 ...
# $ prob   : num  0.0206 0.0597 0.0146 0.0597 0.0146 ...
# $ lon    : num  5.61 7.08 8.3 5.52 6.74 ...
# $ lat    : num  41 44.9 42.8 36.7 33 ...
# $ logCode: Factor w/ 37 levels "1RZ","1SN","1ST",..: 12 12 12 12 12 12 12 12 12 12 ...

# explanations of variables:
# tFirst   : date and time of first sun event
# tSecond  : date and time of second sun event
# type     : type of sunevent 1 = sunrise, 2 = sunset
# site     : not used below
# prob     : not used below
# lon      : longitude
# lat      : latitude
# logCode  : id of bird/logger

# define date as a date variable
dat$tFirst.date <- strptime(as.character(dat$tFirst), format="%Y-%m-%d %H:%M")
dat$tSecond.date <- strptime(as.character(dat$tSecond), format="%Y-%m-%d %H:%M")

# extract specific date variables
dat$month <- as.numeric(substring(as.character(dat$tFirst.date), 6, 7))
dat$dayofyear <- dat$tFirst.date$yday+1
dat$dayofwinter <- dat$dayofyear
dat$year <- dat$tFirst.date$year+1900
dat$dayofwinter[dat$dayofyear<180] <-dat$dayofyear[dat$dayofyear<180]+
ifelse(dat$year[dat$dayofyear<180]==2012, 366, 365)

# order data according to logCode (bird id) and date
dat <- dat[order(dat$logCode, dat$tFirst.date),]
dat <- dat[dat[,2]!="",]  # delete observations with missing times of sunevents

#-------------------------------------------------------------------------------------------
### calculation of the weight factor based on day/night length
### weight accounts for how many times a specific day/night length is the least shaded one within the
moving window of size m

### max daylength and min nightlength for a 2m sized window
daylength <- ifelse (dat$type == 1, difftime(dat$tSecond.date,dat$tFirst.date,units="mins"), 0)
nightlength <- ifelse (dat$type == 2, difftime(dat$tSecond.date,dat$tFirst.date,units="mins"), 10000)

dt <- vector(mode="numeric",length(daylength))      # difference of absolute day/night length to the
maximum/minimum in the moving window
w1 <- vector(mode="numeric",length(daylength))      # weight factor
```

```
dtdet <- vector(mode="numeric",length(daylength))   # difference of the residual (detrended) day/night
length to the maximum/minimum in the moving window
w1det <- vector(mode="numeric",length(daylength))   # weight factor
n_w1 <- vector(mode="numeric",length(daylength))    # number of observations used

# parameter settings
m <- 3                                  # size of moving window "+-m"
i <- 2*m+1
j <-  -2*m
for (i in (2*m+1):(length(daylength)-2*m)) {
  for (j in seq(-2*m,2*m,by=2))  {
    if (dat$logCode[i-2*m]==dat$logCode[i+2*m])  {
      if (dat$type[i+j] == 1) {
        mod <- lm(daylength[seq((i-2*m),(i+2*m), by=2)]~dat$dayofwinter[seq((i-2*m),(i+2*m), by=2)])
        residdaylength <- resid(mod)
        dtdet[i+j] <-  max(residdaylength)-residdaylength[j/2+m+1]
        w1det[i+j] <- w1det[i+j] + dtdet[i+j]

        dt[i+j] <-  max(daylength[(i-2*m):(i+2*m)])-daylength[i+j]
        w1[i+j] <- w1[i+j] +dt[i+j]
        n_w1[i+j] <- n_w1[i+j] + 1
      }
      if (dat$type[i+j] ==2) {
        mod <- lm(nightlength[seq((i-2*m),(i+2*m), by=2)]~dat$dayofwinter[seq((i-2*m),(i+2*m), by=2)])
        residnightlength <- resid(mod)
        dtdet[i+j] <- residnightlength[j/2+m+1] - min(residnightlength)
        w1det[i+j] <- w1det[i+j] + dtdet[i+j]

        dt[i+j] <- nightlength[i+j] - min(nightlength[(i-2*m):(i+2*m)])
        w1[i+j] <- w1[i+j] + dt[i+j]
        n_w1[i+j] <- n_w1[i+j] + 1
      }
    }}
}

# calculating weights: 4 different weights: 0 for differences larger than 20 min,
#                                  1 for differences between 10 and 20
#                                  2 for differences between 5 and 10
#                                  3 for differences between 0 and 5 min to the min/max night/daylength

w1 <- w1/(n_w1)       # average difference to the min/max night/day length in minutes
w1det <- w1det/(n_w1) # average difference to the min/max detrended night/day length in minutes

c1 <- 5      # level1
c2 <- 10     # level2
c3 <- 20     # level3

w2 <- ifelse (w1 < c3, 1, 0)
w2 <- ifelse (w1 < c2, 2, w2)
w2 <- ifelse (w1 < c1, 3, w2)

w2det <- ifelse (w1det < c3, 1, 0)
w2det <- ifelse (w1det < c2, 2, w2det)
w2det <- ifelse (w1det < c1, 3, w2det)

# include daylength, nightlength, mean difference to min/max, weight and detrended difference and
weight in the data frame
dat$daylength <- ifelse(dat$type==1,daylength,nightlength)
dat$mean_td <- w1
dat$posweight <- w2
dat$mean_tddet <- w1det
dat$posweightdet <- w2det
```

```
# delete dates from 13. 9. to 3. 10. and from 10.3. to 30.3. due to equinox
dat$equinox <- (dat$dayofyear >= 256 & dat$dayofyear <= 276)| (dat$dayofyear>=69 &
dat$dayofyear<= 89)

# optional saving of results
# write.table(dat, file="/data/positions_and_weights.txt", row.names=FALSE, sep="\t")
#-----------------------------------------------------------------------------------------------
# 1a)  identify stationary periods using change-point for events with weight > 0
# -----------------------------------------------------------------------------------------------
# optional reading the data (results from first step), and re-defining tFirst.date and tSecond.date
# dat <- read.table("/positions_and_weights.txt", header=TRUE, sep="\t")
# dat$tFirst.date <- strptime(as.character(dat$tFirst), format="%Y-%m-%d %H:%M")
# dat$tSecond.date <- strptime(as.character(dat$tSecond), format="%Y-%m-%d %H:%M")

datw <- dat[dat$posweightdet>0,]    # select only events with detrended weight >0
# datw <- dat[dat$posweight>0,]     # select only events with weight >0, if not-detrended weights would
like to be used

#parameters for regression method
ndays <- 7     # length of data used to fit the regressions for the past ndays and the following ndays
# the slopes if these regressions are used to determine stationary periods, moving periods and
changepoints
tsds <- 0.1    # threshold difference in slope (to identify diffbetas close to zero = stationary),
comparison between regression of the past ndays with the ndays in future
tsds2 <- 0.04  # threshold difference in slope (for the sum of two consecutive diffbetas)
trsh2 <-  0.05 # threshold slope for sun events~date, higher=movement
tslon <- 0.2   # threshold slope for longitude within a phase, higher=movement

# going through all loggers
for(i in 1:nlevels(dat$logCode)){
  log_code <-  levels(dat$logCode)[i]
  tab <- datw[datw$logCode==log_code,]

  datrise <- tab[tab$type==1,]
  datset <- tab[tab$type==2,]

  datrise$hour <- as.numeric(substring(as.character(datrise$tFirst.date), 12, 13))
  datrise$min <- as.numeric(substring(as.character(datrise$tFirst.date), 15, 16))
  datrise$hour.dec <- datrise$hour + datrise$min/60
  datrise$date.num <- as.numeric(datrise$tFirst.date)/60/60/24

  datset$hour <- as.numeric(substring(as.character(datset$tFirst.date), 12, 13))
  datset$min <- as.numeric(substring(as.character(datset$tFirst.date), 15, 16))
  datset$hour.dec <- datset$hour + datset$min/60
  datset$date.num <- as.numeric(datset$tFirst.date)/60/60/24

  # regression event-time ~ datum for ndays before (past) und ndays after (future)
  # the actual observation
  # separately for sunset and sunrise
  tab$beta.past <- NA
  tab$beta.future <- NA
  tab$R2.past <- NA
  tab$R2.future <- NA
  tab$density.past <- NA
  tab$density.future <- NA
  tab$resid.past <- NA
  tab$resid.future <- NA
  tab$p.past <- NA
  tab$p.future <- NA
  tab$date.num <- as.numeric(tab$tFirst.date)/60/60/24
```

```
  for(k in 1:nrow(tab)){
    if(tab$type[k]==1) tempdat <- datrise else tempdat <- datset
    index <- tempdat$tFirst.date == tab$tFirst.date[k]
    tempdat$rownumber <- 1:nrow(tempdat)

    # regressions in past
    indexpast <- tempdat$rownumber >= (tempdat$rownumber[index]-ndays) & tempdat$rownumber <
tempdat$rownumber[index]
    if(sum(indexpast)>2){
      mod <- lm(hour.dec~date.num, data=tempdat[indexpast,])
      tab$beta.past[k] <-  coef(mod)[2]
      tab$R2.past[k] <- summary(mod)$r.squared
      tab$density.past[k] <- dnorm(tempdat$hour.dec[index], mean=predict(mod,
newdata=tempdat[index,]), sd=summary(mod)$sigma)
      plower <- pnorm(tempdat$hour.dec[index], mean=predict(mod, newdata=tempdat[index,]),
sd=summary(mod)$sigma)
      if(plower>0.5) plower <- 1-plower
      tab$p.past[k] <- 2*plower
      tab$resid.past[k] <- tempdat$hour.dec[index]- predict(mod, newdata=tempdat[index,])
    }
    # regressions in future
    indexfuture <- tempdat$rownumber > tempdat$rownumber[index] & tempdat$rownumber <=
(tempdat$rownumber[index]+ndays)
    if(sum(indexfuture)>2){
      mod <- lm(hour.dec~date.num, data=tempdat[indexfuture,])
      tab$beta.future[k] <-  coef(mod)[2]
      tab$R2.future[k] <- summary(mod)$r.squared
      tab$density.future[k] <- dnorm(tempdat$hour.dec[index], mean=predict(mod,
newdata=tempdat[index,]), sd=summary(mod)$sigma)
      plower <- pnorm(tempdat$hour.dec[index], mean=predict(mod, newdata=tempdat[index,]),
sd=summary(mod)$sigma)
      if(plower>0.5) plower <- 1-plower
      tab$p.future[k] <- 2*plower
      tab$resid.future[k] <- tempdat$hour.dec[index]- predict(mod, newdata=tempdat[index,])
    }
  }  # close k

  matchindex <- match(dat$tFirst.date[dat$logCode==log_code], tab$tFirst.date)

  dat$p.past[dat$logCode==log_code] <- tab$p.past[matchindex]
  dat$resid.past[dat$logCode==log_code] <- tab$resid.past[matchindex]
  dat$beta.past[dat$logCode==log_code] <- tab$beta.past[matchindex]
  dat$R2.past[dat$logCode==log_code] <- tab$R2.past[matchindex]
  dat$density.past[dat$logCode==log_code] <- tab$density.past[matchindex]
  dat$p.future[dat$logCode==log_code] <- tab$p.future[matchindex]
  dat$resid.future[dat$logCode==log_code] <- tab$resid.future[matchindex]
  dat$beta.future[dat$logCode==log_code] <- tab$beta.future[matchindex]
  dat$R2.future[dat$logCode==log_code] <- tab$R2.future[matchindex]
  dat$density.future[dat$logCode==log_code] <- tab$density.future[matchindex]

  tab$diffbeta <- tab$beta.future-tab$beta.past
  tab$cpboth <- NA

  for(k in 2:(nrow(tab)-2)){
    if((tab$dayofyear[k]- tab$dayofyear[k-1])<2){
          tab$cpboth[k] <- abs(tab$diffbeta[k]+tab$diffbeta[k-1])>2*tsds2
    }
  }
  tab$cpboth[is.na(tab$cpboth)] <- FALSE

  #  find changepoints and decide whether phases are stationary or not
```

```
  datrise <- tab[tab$type==1,]    # redefine these data sets, now including diffbetas
  datset <- tab[tab$type==2,]
  datrise$hour <- as.numeric(substring(as.character(datrise$tFirst.date), 12, 13))
  datrise$min <- as.numeric(substring(as.character(datrise$tFirst.date), 15, 16))
  datrise$hour.dec <- datrise$hour + datrise$min/60
  datrise$date.num <- as.numeric(datrise$tFirst.date)/60/60/24

  datset$hour <- as.numeric(substring(as.character(datset$tFirst.date), 12, 13))
  datset$min <- as.numeric(substring(as.character(datset$tFirst.date), 15, 16))
  datset$hour.dec <- datset$hour + datset$min/60
  datset$date.num <- as.numeric(datset$tFirst.date)/60/60/24

  datfulli <- data.frame(dayofyear=c(200:365, 1:180))
  datfulli$phase <- 1
  datfulli$nevents <- 0
  datfulli$nevens[1] <- NA
  datfulli$stationary <- NA
  datfulli$cp <- FALSE

  for(k in 2:nrow(datfulli)){
    datfulli$nevents[k] <- sum(is.element(tab$dayofyear, datfulli$dayofyear[k]))
    if(datfulli$nevents[k] ==0) datfulli$phase[k] <- datfulli$phase[k-1]
    if(datfulli$nevents[k] ==0) next

    # for rise
    askifstationaryrise <- NA
    askifcprise <- FALSE
    if(is.element(datfulli$dayofyear[k], datrise$dayofyear)){
      krise <- c(1:nrow(datrise))[datrise$dayofyear==datfulli$dayofyear[k]]
      askifstationaryrise <- abs(datrise$diffbeta[krise]) < tsds  & mean(c(abs(datrise$beta.past[krise]),
abs(datrise$beta.future[krise])), na.rm=TRUE) < trsh2 & !datrise$cpboth[krise]
      if(!is.na(askifstationaryrise)){
        if(!askifstationaryrise){
          diffdiffbeta1 <-  datrise$diffbeta[krise] -datrise$diffbeta[krise-1]
          diffdiffbeta2 <-  datrise$diffbeta[krise+1] -datrise$diffbeta[krise]
          if(length(diffdiffbeta1)>0&length(diffdiffbeta2)>0) askifcprise <- sign(diffdiffbeta1) !=
sign(diffdiffbeta2)
        }}
    }# for rise

    # for set
    askifstationaryset <- NA
    askifcpset <- FALSE
    if(is.element(datfulli$dayofyear[k], datset$dayofyear)){
      kset <- c(1:nrow(datset))[datset$dayofyear==datfulli$dayofyear[k]]
      askifstationaryset <- abs(datset$diffbeta[kset]) < tsds  & mean(c(abs(datset$beta.past[kset]),
abs(datset$beta.future[kset])), na.rm=TRUE) < trsh2 & !datset$cpboth[kset]
      if(!is.na(askifstationaryset)){
        if(!askifstationaryset){
          diffdiffbeta1 <-  datset$diffbeta[kset] - datset$diffbeta[kset-1]
          diffdiffbeta2 <-  datset$diffbeta[kset+1] -datset$diffbeta[kset]
          if(length(diffdiffbeta1)>0&length(diffdiffbeta2)>0) askifcpset <- sign(diffdiffbeta1) !=
sign(diffdiffbeta2)
        }}
    }# for set

    # decide both on rise and set  (both need to be stationary but NAs are ignored)
    if(sum(c(askifstationaryrise,askifstationaryset),
na.rm=TRUE)==sum(!is.na(c(askifstationaryrise,askifstationaryset))))  datfulli$stationary[k] <- TRUE
    # non stationary if at least one is FALSE
    if(!is.na(askifstationaryrise))  if(!askifstationaryrise)   datfulli$stationary[k] <- FALSE
    if(!is.na(askifstationaryset))  if(!askifstationaryset)   datfulli$stationary[k] <- FALSE
```

```
    if(is.na(askifstationaryrise) &  is.na(askifstationaryset))  datfulli$stationary[k] <- NA

    if(!is.na(askifcpset)&!is.na(askifcprise)){
      if(askifcpset|askifcprise) datfulli$cp[k] <- TRUE
      if(askifcpset|askifcprise) datfulli$phase[k] <- datfulli$phase[k-1] +1
      else  datfulli$phase[k] <- datfulli$phase[k-1]
    }

    if(!is.na(askifcpset)&is.na(askifcprise)){
      if(askifcpset) datfulli$cp[k] <- TRUE
      if(askifcpset) datfulli$phase[k] <- datfulli$phase[k-1] +1
      else  datfulli$phase[k] <- datfulli$phase[k-1]
    }

    if(is.na(askifcpset)&!is.na(askifcprise)){
      if(askifcprise) datfulli$cp[k] <- TRUE
      if(askifcprise) datfulli$phase[k] <- datfulli$phase[k-1] +1
      else  datfulli$phase[k] <- datfulli$phase[k-1]
    }
    if(is.na(askifcpset)&is.na(askifcprise)){
      datfulli$phase[k] <- datfulli$phase[k-1]
    }
  } # close k  (rows of datfulli)

  tab$phase <- datfulli$phase[match(tab$dayofyear, datfulli$dayofyear)]
  tab$stationary <- datfulli$stationary[match(tab$dayofyear, datfulli$dayofyear)]
  tab$cp <- datfulli$cp[match(tab$dayofyear, datfulli$dayofyear)]

  dat$phase[dat$logCode==log_code] <- tab$phase[matchindex]
  dat$stationary[dat$logCode==log_code] <- tab$stationary[matchindex]
  dat$cp[dat$logCode==log_code] <- tab$cp[matchindex]
  dat$cpboth[dat$logCode==log_code] <- tab$cpboth[matchindex]

  # fill up the phases and decide for stationary or not in the non-weighted data set
  phasen <- unique(dat$phase[dat$logCode==log_code])
  phasen <- phasen[!is.na(phasen)] # delete NAs
  dat$slopelon <- NA
  dat$slopelat <- NA
  datw$phase <- NA

  for(p in phasen){
    index <- dat$logCode==log_code & dat$phase==p
    index[is.na(index)] <- FALSE
    firstday <- min(dat$dayofwinter[index])
    lastday <-  max(dat$dayofwinter[index])
    index2 <-  dat$logCode==log_code & dat$dayofwinter>=firstday & dat$dayofwinter<=lastday
    dat$phase[index2] <- p
    statav <- ifelse(mean(dat$stationary[index2], na.rm=TRUE)<0.5, FALSE, TRUE)
    dat$stationary[index2] <- statav

    # add slope for longitude and latitude based on weighted positions
    datw$phase[datw$dayofwinter>=firstday & datw$dayofwinter<=lastday &
datw$logCode==log_code] <- p
    indexw <- datw$logCode==log_code & datw$phase==p
    indexw[is.na(indexw)] <- FALSE
    if(sum(indexw)>1){
      slopelon <- coef(lm(lon~dayofwinter, datw[indexw,]))[2]
      if(sum(indexw)-sum(is.na(datw$lat[indexw]))>1)
        slopelat <- coef(lm(lat~dayofwinter, datw[indexw,]))[2]
      dat$slopelon[index2] <- slopelon
      dat$slopelat[index2] <- slopelat
    }
```

```
  }# close p

  dat$stationary[dat$logCode==log_code]<- abs(dat$slopelon[dat$logCode==log_code])< tslon

  tab <- dat[dat$logCode==log_code,]
  tab$diffbeta <- tab$beta.future-tab$beta.past

  # draw cp_regression graph
  jpeg(file.path("temp", paste0("regr", log_code, "tsdiff", tsds, "tsboth", tsds2, "tsslope", trsh2, "days",
ndays, "tslon", tslon ,"det.jpg")), width = 480, height = 550)

  par(mfrow=c(4,1), mar=c(2,4,1,1))
  plot(datrise$tFirst.date, datrise$hour.dec, type="l", main=log_code, col="orange",
xlim=as.numeric(range(tab$tFirst.date)), ylab="time")
  abline(v=as.numeric(tab$tFirst.date[tab$cp]))
  ywerte <- rep(max(datrise$hour.dec), sum(!is.na(tab$stationary)))
  xwerte <- tab$tFirst.date[!is.na(tab$stationary)]
  stationary <- tab$stationary[!is.na(tab$stationary)]
  points(xwerte,ywerte , pch=15, col=c("orange", "blue")[as.numeric(stationary)+1])
  #text(tab$tFirst.date, rep(max(datrise$hour.dec), nrow(tab))-0.05, tab$phase, cex=0.7)
  points(tab$tFirst.date, rep(max(datrise$hour.dec), nrow(tab))-0.05, pch=15, col=tab$phase, cex=0.5)

  par(new=TRUE)
  plot(datset$tFirst.date, datset$hour.dec, type="l", main=log_code, col="blue",
xlim=as.numeric(range(tab$tFirst.date)), ylab=NA)
  par(new=FALSE)

  plot(tab$tFirst.date, tab$diffbeta, xlim=as.numeric(range(tab$tFirst.date)), pch=16, cex=0.7,
ylab="diff slope", col=c("orange", "blue")[tab$type])
  abline(h=0)
  abline(v=as.numeric(tab$tFirst.date[tab$cp]))

  plot(tab$tFirst.date, tab$slopelon, xlim=as.numeric(range(tab$tFirst.date)), pch=16, cex=0.7,
ylab="slope", col=c("orange", "blue")[tab$type])
  #lines(tab$tFirst.date, tab$beta.future)
  abline(v=as.numeric(tab$tFirst.date[tab$cp]))

  plot(tab$tFirst.date, tab$lon, xlim=as.numeric(range(tab$tFirst.date)), type="l", ylab="latitude (or),
longitude (bl)", col="blue")
  par(new=TRUE)
  plot(tab$tFirst.date, tab$lat, xlim=as.numeric(range(tab$tFirst.date)), type="l", ylab="latitude (or),
longitude (bl)", col="orange")
  dev.off()
} # close i

# The following warning can be ignored: "In summary.lm(mod) : essentially perfect fit: summary may
be unreliable"

# optional save results
# write.table(dat, "positions_weight_phase.txt", row.names=FALSE, sep="\t")
#-------------------------------------------------------------------------------
# 2) merge stationary periods according to the loxodromic distance between them
#-------------------------------------------------------------------------------
# optional read in data from step 1
# dat <- read.table("positions_weight_phase.txt", header=TRUE, sep="\t")
# dat <- dat[order(dat$logCode, dat$tFirst.date),]

# increase the equinox-period to plusminus 21 days
dat$equinox <-(dat$dayofyear >= 245 & dat$dayofyear <= 287)| (dat$dayofyear>=58 &
dat$dayofyear<= 100) # plusminus 21 days

moddist <- 200 # if two modus are closer than moddist, the two consecutive sites are merged
```

```
dat$rownumber <- 1:nrow(dat)
datw <- dat[dat$posweightdet>0 & !dat$equinox,]   # filtered dat

datpsite <- aggregate(datw$lon, list(site=datw$phase, logCode=datw$logCode), ownmode)
names(datpsite)[names(datpsite)=="x"] <- "lon"
datpsite$lat <- aggregate(datw$lat, list(site=datw$phase, logCode=datw$logCode), ownmode)$x
datpsite$n <- aggregate(datw$lat, list(site=datw$phase, logCode=datw$logCode), length)$x
datpsite$stationary <- aggregate(as.numeric(datw$stationary), list(site=datw$phase,
logCode=datw$logCode), median)$x
datpsite$n_lat <- aggregate(datw$lat, list(site=datw$phase, logCode=datw$logCode), function(x)
sum(!is.na(x)))$x

# merge sites
datpsite <-datpsite[datpsite$n>4,]  # use only sites with more than 4 locations

datpsite$sitemerged <- datpsite$site
datpsite$distcentr <- NA
for(ind in levels(factor(datpsite$logCode))){
  index <- datpsite$logCode==ind
  if(sum(index)<2) next
  for(i in 1:(sum(index)-1)){
    if(is.na(datpsite$stationary[index][i])|is.na(datpsite$stationary[index][i+1])) next  # unknown
    if(datpsite$stationary[index][i]<0.5|datpsite$stationary[index][i+1]<0.5) next  # migration
    if(is.na(datpsite$lat[index][i])|is.na(datpsite$lat[index][i+1])) next  # unknown
    distbetweencentroid <- distCosine(c(datpsite$lon[index][i], datpsite$lat[index][i]),
c(datpsite$lon[index][i+1], datpsite$lat[index][i+1]))/1000
    if(distbetweencentroid< moddist) datpsite$sitemerged[index][i+1] <- datpsite$sitemerged[index][i]
    if(distbetweencentroid>=moddist) datpsite$sitemerged[index][i+1] <- datpsite$sitemerged[index][i]+1
    datpsite$distcentr[index][i] <-  distbetweencentroid
    # write.table(datpsite, file="datpsite_withsitemerged.txt", row.names=FALSE)    # optional: save a
copy (just in case)
  }
}

# include in dat and in datw
dat$sitemerged <- datpsite$sitemerged[match(paste(dat$logCode, dat$phase),
paste(datpsite$logCode, datpsite$site))]
datw$sitemerged <- datpsite$sitemerged[match(paste(datw$logCode, datw$phase),
paste(datpsite$logCode, datpsite$site))]

# fill the movement periods between two equal stationary sites with this site
dat$sitemerged_filled <- dat$sitemerged
for(i in levels(dat$logCode)){
  for(j in unique(dat$sitemerged[dat$logCode==i])){
    if(is.na(j)) next
    erstezeile <- min(dat$rownumber[dat$logCode==i & dat$sitemerged == j], na.rm=TRUE)
    letztezeile <- max(dat$rownumber[dat$logCode==i & dat$sitemerged == j], na.rm=TRUE)
    dat$sitemerged_filled[dat$rownumber>=erstezeile & dat$rownumber<=letztezeile] <- j
  }
}

# optional save results
# write.table(dat, file="positions_weight_phase_sitesmerged.txt", row.names=FALSE, sep=";")
# end code     ----------------------------------------------------------------------------------------------------------------------
```