R code to implement major results in the manuscript:
'Hidden Markov Models for Monitoring Circadian Rhythmicity in Telemetric Activity Data'

Maintainer: Qi Huang <Q.Huang.6@warwick.ac.uk>

Required R package: *depmixS4*

Data:
Three healthy example subjects' data sets

Functions:
(see comments in code for detailed explanation)
Homogeneous_HMMs
Harmonic_HMMs
figure_HMM_decoding_SP
figure_day_profile

Example:
By using on the above four functions, example_HMMs shows how to implement major HMMs results namely on homogeneous HMMs (with plots and parameters as discussed in the section "Application to activity data from healthy subjects") and harmonic HMMs (with plots and parameters as discussed in the section "Circadian Harmonic Markov Model and Circadian Parameters").

example_HMMs

```
#This R code gives example to implement major results discussed in the
manuscript:
#'Hidden Markov Models for Monitoring Circadian Rhythmicity in Telemetric
Activity Data'


# Major assumption:
#1) Gaussian distributed data  (after a suitable transformation)
#2) 3 states:
#   1: inactive state; 2: moderately active state; 3: highly active state

#R packages depmixS4 is applied to implement HMMs.

library("depmixS4")

#Four functions that used in example_HMMs are in the last



##############################################################
####Example subjects ids in the main manuscript
##############################################################
id_all<-c(9,16,20)
id<-9
info<-read.csv(paste0('S',id,'.csv'))

# info contains:
#time: time of observations.  5-min resolution is used.

#activity:  activity observations (5-min aggregates, may have missing values).


info$sqValue<-sqrt(info$activity) # square root is applied to improve normality

#NOTE: square root of 5-min average is sufficient for our data but maybe not for
others. Try 10-min (or more) if you keep meeting convergence problem.

##############################################################
##Homogeneous HMMs, where homogeneous transition probabilities are assumed.
##############################################################

y<-depmix(sqValue~1,data=info,nstates=3,family=gaussian(),ntimes=nrow(info))
```

```
HMMs<-fit(y,verbose=F)
#NOTE: the likelihood may have local maxima and it is advised to test different
starting values using set.seed() and choose the results via AIC/BIC.
#By experience, around 5% subjects are senstive to the starting value.

# Summarise some useful parameters for further analysis and plotting.
# More parameters are avaiable, see help document of R package depmixS4.
# source('Homogeneous_HMMs.R')
HMM_results<-Homogeneous_HMMs(HMM=HMMs)


###########################################################
####Harmonic HMMs (Circadian Harmonic)
###########################################################
# 24-hour circadian harmonic is applied to make the transition probability oscillates

sf<-1/12 #sampling frequency in hour unit, i.e. here sf=5min/60min=1/12
lag<-seq(0,(nrow(info)-1))

info$sin_part<-sin(2*pi*lag*sf/24)
info$cos_part<-cos(2*pi*lag*sf/24)


set.seed(6)

y_hmc<-
depmix(sqValue~1,transition=~sin_part+cos_part,data=info,nstates=3,family=gauss
ian(),ntimes=nrow(info))
HMM_hmc<-fit(y_hmc,verbose=F)
#NOTE: the likelihood may have local maxima and it's a more common case in
harmnonic HMMs as more parameters are involved
#It is advised to test different starting values using set.seed() and choose the results
via logLik/AIC/BIC.
#for example, for subject 9, set.seed(6) has better logLik/AIC/BIC results than
set.seed(4)


# Summarise some useful parameters for further analysis and plotting.
# More parameters are avaiable, see help document of R package depmixS4.
source('Harmonic_HMMs.R')
HMM_results_hmc<-
Harmonic_HMMs(HMM=HMM_hmc,sin_part=info$sin_part,cos_part=info$cos_part)


###########################################################
```

```
#The function figure_HMM_decoding_SP generates plots like Figure 5 & 6 in the
manuscript.
###########################################################
# source('figure_HMM_decoding_SP.R')

#plot Homogeneous HMMs results

figure_HMM_decoding_SP(HMM_results,info, id)

#plot harmonic HMMs results
#figure_HMM_decoding_SP(HMM_results_hmc,info, id)


###########################################################
####Based on HMM_results_hmc, compute day profile and circadian parameters
###########################################################

#extract the oscillated state probability
circadian_states_prob<-HMM_results_hmc$circadian_states_prob
time<-as.POSIXct(info$time, format="%Y-%m-%d %H:%M:%S")

#circadian_states_prob is 24-h periodic.
#Most healthy people sleep/rest at night so generally we choose 12pm-11:59am to
present a day (with one-phase sleep/rest):

hour_day_start<-12

# The selection of hour_day_start will affect center rest time and Rhythm Index (RI).
I suggest one should choose hour_day_start which leads to highest RI and by
experience, hour_day_start =12 is suitable for most healthy subjects.

#For late-type person, one should select another suitable time segment, i.e. 0am-
23:59pm

library('lubridate')

# this function is used to find an appropriate time index of a day start
find_day_start<-function(hour_day_start,time){
  hour_time<-hour(time)
  min_time<-minute(time)
  A<-which(hour_time==hour_day_start)
  B<-which(min_time<60*sf)
  one_day_start<-A[min(which(A%in%B, arr.ind = TRUE))]
  return(one_day_start)
}
```

```
one_day_start<-find_day_start(hour_day_start=hour_day_start,time=time)
one_day_end<-one_day_start+(24/sf-1) #we use 5-mins data (sf=1/12) so that one
day has 24*12 points

# one day profile
one_day_prob<-circadian_states_prob[one_day_start:one_day_end,]
#three states probability
p1<-one_day_prob$state_1 #inactive state probability
p2<-one_day_prob$state_2 #moderately active state probability
p3<-one_day_prob$state_3 #highly active state probability

############################################################
#Following circadian parameters are calculated based on one_day_prob
#see details in Circadian Parameters Section of the main manuscript
############################################################

#amount of rest: duration of rest per day (hours)
rest_amount<-24*sum(p1)/nrow(one_day_prob)


index<-seq(0,24-1/12,1/12) #absolute index to compute the gravity centre of p1
#center of rest which corresponds to the gravity centre of p1
#Note: only valid with suitable hour_day_start where the major sleep/rest is of one-
phase.

center_rest<-sum(p1*index/sum(p1))

#center_rest is the position in index not the clock time.
#One can easliy convert it to clock time via:

index_clocktime<-seq(hour_day_start,hour_day_start+24-sf,sf)
clocktime<-index_clocktime
clocktime[index_clocktime>24]<-(index_clocktime-24)[index_clocktime>24]
center_rest_clock<-clocktime[which.min(abs(index-center_rest))]


#worst clock : complete lack of circadian rhythm where the probability of rest is
constant and equal to rest_amount/24
worst_p1<-rep(mean(p1),24/sf)

#perfect clock: rest period with no interruptions
find_perfect_p1<-function(center_rest,rest_amount,index,sf){
  perfect_p1<-rep(0,24/sf)

  t1<-center_rest-rest_amount/2
  t2<-center_rest+rest_amount/2
  perfect_t2<-which.min(abs(index-center_rest))
  perfect_t1<-which.min(abs(index-center_rest+rest_amount/2))
  perfect_t3<-which.min(abs(index-center_rest-rest_amount/2))
  perfect_p1[perfect_t1:perfect_t3]<-1
  if(t2>max(index)){offset<-round((t2-max(index))/sf);perfect_p1[1:offset]<-1}
  if(t1<min(index)){offset<-round((min(index)-t1)/sf);perfect_p1[-offset+L,L]<-1}
  return(perfect_p1)
}
perfect_p1<-find_perfect_p1(center_rest,rest_amount,index,sf)

#rhythm index RI, see in eq(4) of the main manuscript.

RI<-(sum(p1[which(perfect_p1>0)])*sf/rest_amount-rest_amount/24)*24/(24-
rest_amount)

############################################################
#the function figure_day_profile generates day profile plots like Figure 7 in the
manuscript.
############################################################

#We choose 12pm-11:59am to present one day  so that use the following setting in
the plot. For other hour_day_start, the following parameters need to be adjusted.
index_position<-seq(0,24,2)
index_lable<-c('12','14','16','18','20','22',
            '0/24','2','4','6','8','10','12') #corresponding clock time
#Note: one need to change index_lable if the day segment changes.

#source('figure_day_profile.R')
par(fig=c(0,1,0,1), new=F)

figure_day_profile(id,one_day_prob,index,index_position,index_lable)
```

function: Homogeneous_HMMs

```
#' Given the output of depmix (Homogeneous HMMs), this function summarises
some useful results for further analysis and plotting

#' @param HMM: HMM fitting results  (the return of depmix function). Three
activity states are assumed:
# 1: inactive state; 2: moderately active state; 3: highly active state
#' @return ML_states: maximal likelihood states at each time point (results of local
decoding)
#' @return prob_ML_states: probability of ML_states at each time point (results of
local decoding)
#' @return trans_matrix: transition probability matrix
#' @return AIC and BIC
#' @return obs_density_sq: mean and standard  deviation of  observation densities
(square root), conditioned on 3 states
#' @return obs_density: 5%, 50%, 95% quantile of observation densities (original
scale), conditioned on 3 states

Homogeneous_HMMs<-function(HMM){

  ##in HMM, the order of 3 states are random. we need to re-arrange them according
to their mean values
  #note: all the following results are based on 3 states!
  #for other number of states, the parameters order in HMM will change and one
should carefully extract them form getpars(HMM)


  obs_params<-data.frame(mean=summary(HMM)[1:3],sd=summary(HMM)[4:6])

  state1_lable<-which.min(obs_params$mean)
  state3_lable<-which.max(obs_params$mean)
  state2_lable<-which(obs_params$mean==median(obs_params$mean))

  ##################################
  ###Local decoding
  ##################################
  e1<-forwardbackward(HMM)$gamma
  L<-nrow(e1)
  state1<-e1[,state1_lable]
  state2<-e1[,state2_lable]
  state3<-e1[,state3_lable]
  states<-rep(NA,L)
```

```
 for(i in 1:L){
   states[i]<-ifelse((state1[i]>state2[i] &
state1[i]>state3[i]),0,ifelse(state2[i]>state3[i],1,2))
 }

 #ML_states: maximal likelihood of current states
 ML_states<-states+1
 #ML_states has 3 numbers where
 # 1: inactive; 2: moderately active; 3: highly active

 #probability of each states
 prob_ML_states<-cbind(state1,state2,state3)



 ##################################
 ##Summarise transition matrix
 ##################################
 A<-getpars(HMM)

 tmat<-matrix(NA,nrow=3,ncol=3)
 for (i in 1:3){
   start<-3*i+1
   tmat[i,]<-A[start:(start+3-1)]
 }

 trans_matrix<-matrix(NA,nrow=3,ncol=3)

 ##################################
 ####summerise observation densities, conditioned on 3 states
 ##################################


 mean_obs<-obs_params$mean

 state_lable<-c(state1_lable,state2_lable,state3_lable)

 for (i in 1:3){
   for (j in 1:3){
     trans_matrix[i,j]<-tmat[state_lable[i],state_lable[j]]
   }}

 mean_sq<-
c(mean_obs[state1_lable],mean_obs[state2_lable],mean_obs[state3_lable])
```

```
  sd_sq<-
c(obs_params$sd[state1_lable],obs_params$sd[state2_lable],obs_params$sd[state3_
lable])
  nc_prams<-(mean_sq/sd_sq)^2

  var<-sd_sq^2
  obs_density<-matrix(NA,3,3)
  for (i in 1:3){
    obs_density[i,]<- qchisq(p=c(0.05,0.5,0.95), df=1, ncp = nc_prams[i])*var[i]
  }

  obs_density_sq<-data.frame(mean=mean_sq,sd=sd_sq)


  return(list(ML_states=ML_states,
prob_ML_states=prob_ML_states,trans_matrix=trans_matrix,
              AIC=AIC(HMM),BIC=BIC(HMM),
              obs_density=obs_density,obs_density_sq=obs_density_sq))

}
```

function: Harmonic_HMMs

```
#' Harmonic_HMMs
#' Given the output of depmix (Harmonic HMMs), this function summarises some
useful results for further analysis and plotting

#' @param HMM: 24-hour circadian harmonic HMM fitting results  (the return of
depmix function). Three activity states are assumed:
# if 3 states: 1: inactive state; 2: moderately active state; 3: highly active state
# if 2 states: 1: inactive state; 2:  active state;

#' @return ML_states: maximal likelihood states at each time point (results of local
decoding)
#' @return prob_ML_states: probability of ML_states at each time point (results of
local decoding)
#' @return transition_prob (oscillates with 24-h period):  transition probability
#' @return circadian_states_prob  (oscillates with 24-h period):   state probability
#' @return AIC and BIC
#' @return obs_density_sq: mean and standard  deviation of  observation densities
(square root), conditioned on 3 states
#' @return obs_density: 5%, 50%, 95% quantile of observation densities (original
scale), conditioned on 3 states

Harmonic_HMMs<-function(HMM,sin_part,cos_part){

  ##in HMM, the order of m states are random. we need to re-arrange them
according to their mean values
  #note: all the following results are based on m=2 or m=3 states!
  #for other number of states, the parameters order in HMM will change and one
should carefully extract them from getpars(HMM)
  A<-getpars(HMM)

###########################################################
######
  ###local decoding
###########################################################
######
  e1<-forwardbackward(HMM)$gamma
  m<-ncol(e1) #num of state
  L<-nrow(e1)

  ## in HMM, the order of m states are random. we need to re-arrange them
  raw.mean=A[which(names(A) %in% c("(Intercept)"))]
  label.order=order(raw.mean)
  #re-order
  mean=unname(raw.mean[label.order])
  sd=unname(A[which(names(A) %in% c("sd"))][label.order])
  obs_density_sq<-data.frame(mean=mean,sd=sd)

  #decoding state probs
  prob_ML_states<-matrix(NA,nrow=L,ncol=m)
  for (i in 1:m){
    prob_ML_states[,i]<-e1[,label.order[i]]
  }

  #ML state; local decoding
  ML_states<-rep(NA,L)

  for(i in 1:L){
    ML_states[i]<-which.max(prob_ML_states[i,])
  }


##############################################################
######
  ###computue the time-varing transition probabilities: transition_prob

##############################################################
######

  trans.all<-matrix(NA,nrow=L,ncol=m*m)#order: 1 to 1, 1 to 2, 1 to 3, 2 to 1, 2 to 2,
.....
  # parameters of the time-varing transition matrix. see eq(3) of the paper.
  tmat_all<-array(NA,c(m,3,m))

  if(m==3){
    tmat_all[1,1,]<- A[which(names(A) %in%
c("(Intercept).St1","(Intercept).St2","(Intercept).St3"))][1:3]
    tmat_all[2,1,]<- A[which(names(A) %in%
c("(Intercept).St1","(Intercept).St2","(Intercept).St3"))][4:6]
    tmat_all[3,1,]<- A[which(names(A) %in%
c("(Intercept).St1","(Intercept).St2","(Intercept).St3"))][7:9]

    tmat_all[1,2,]<- A[which(names(A) %in%
c("sin_part.St1","sin_part.St2","sin_part.St3"))][1:3]
    tmat_all[2,2,]<- A[which(names(A) %in%
c("sin_part.St1","sin_part.St2","sin_part.St3"))][4:6]
    tmat_all[3,2,]<- A[which(names(A) %in%
c("sin_part.St1","sin_part.St2","sin_part.St3"))][7:9]
```

```
   tmat_all[1,3,]<- A[which(names(A) %in%
c("cos_part.St1","cos_part.St2","cos_part.St3"))][1:3]
   tmat_all[2,3,]<- A[which(names(A) %in%
c("cos_part.St1","cos_part.St2","cos_part.St3"))][4:6]
   tmat_all[3,3,]<- A[which(names(A) %in%
c("cos_part.St1","cos_part.St2","cos_part.St3"))][7:9]

   tmat1<-tmat_all[label.order[1],,]
   tmat2<-tmat_all[label.order[2],,]
   tmat3<-tmat_all[label.order[3],,]
  }

  if(m==2){
   tmat_all[1,1,]<- A[which(names(A) %in%
c("(Intercept).St1","(Intercept).St2"))][1:2]
   tmat_all[2,1,]<- A[which(names(A) %in%
c("(Intercept).St1","(Intercept).St2"))][3:4]

   tmat_all[1,2,]<- A[which(names(A) %in% c("sin_part.St1","sin_part.St2"))][1:2]
   tmat_all[2,2,]<- A[which(names(A) %in% c("sin_part.St1","sin_part.St2"))][3:4]

   tmat_all[1,3,]<- A[which(names(A) %in% c("cos_part.St1","cos_part.St2"))][1:2]
   tmat_all[2,3,]<- A[which(names(A) %in% c("cos_part.St1","cos_part.St2"))][3:4]

   tmat1<-tmat_all[label.order[1],,]
   tmat2<-tmat_all[label.order[2],,]
  }

  transition_from_one_state_m3<-function(a11,a22,a33){
   a123<-c(a11,a22,a33)
   a1<-a123[label.order[1]]
   a2<-a123[label.order[2]]
   a3<-a123[label.order[3]]
   trans<-c(a1,a2,a3)/sum(c(a1,a2,a3))
   trans1<-replace(trans, is.na(trans), (1-sum(trans[!is.na(trans)]))/
sum(is.na(trans)) )
   return(trans1)  }

  transition_from_one_state_m2<-function(a11,a22){
   a12<-c(a11,a22)
   a1<-a12[label.order[1]]
   a2<-a12[label.order[2]]
   trans<-c(a1,a2)/sum(c(a1,a2))
   trans1<-replace(trans, is.na(trans), (1-sum(trans[!is.na(trans)]))/
sum(is.na(trans)) )
   return(trans1)  }
```

```
for (i in 1:L){
 sin_i<-sin_part[i]
 cos_i<-cos_part[i]

 if(m==3){
 # transition from state 1
 a11<-exp(c(1,sin_i,cos_i)%*% tmat1[,1])
 a22<-exp(c(1,sin_i,cos_i)%*% tmat1[,2])
 a33<-exp(c(1,sin_i,cos_i)%*% tmat1[,3])
 cons<-a11+a22+a33
 y1<-transition_from_one_state_m3(a11,a22,a33)

 trans.all[i,1:3]<-y1
 # transition from state 2
 a11<-exp(c(1,sin_i,cos_i)%*% tmat2[,1])
 a22<-exp(c(1,sin_i,cos_i)%*% tmat2[,2])
 a33<-exp(c(1,sin_i,cos_i)%*% tmat2[,3])
 cons<-a11+a22+a33
 y2<-transition_from_one_state_m3(a11,a22,a33)
 trans.all[i,4:6]<-y2

 # transition from state 3
 a11<-exp(c(1,sin_i,cos_i)%*% tmat3[,1])
 a22<-exp(c(1,sin_i,cos_i)%*% tmat3[,2])
 a33<-exp(c(1,sin_i,cos_i)%*% tmat3[,3])
 cons<-a11+a22+a33
 y3<-transition_from_one_state_m3(a11,a22,a33)
 trans.all[i,7:9]<-y3
 }

 if(m==2){
  # transition from state 1
  a11<-exp(c(1,sin_i,cos_i)%*% tmat1[,1])
  a22<-exp(c(1,sin_i,cos_i)%*% tmat1[,2])
  cons<-a11+a22
  y1<-transition_from_one_state_m2(a11,a22)
  trans.all[i,1:2]<-y1
  # transition from state 2
  a11<-exp(c(1,sin_i,cos_i)%*% tmat2[,1])
  a22<-exp(c(1,sin_i,cos_i)%*% tmat2[,2])
  cons<-a11+a22
  y2<-transition_from_one_state_m2(a11,a22)
  trans.all[i,3:4]<-y2
 }
 }



##############################################################
######
 ###computue the probability of each states (time-varing): prob_states

##############################################################
######
 prob.states<-matrix(NA,nrow=L,ncol=m)
 initial_state<-A[1:m][label.order]
 prob.states[1,]<-unname(initial_state)
 ######

 for (i in 2:L){
  trans_matrix_j<-matrix(NA,nrow=m,ncol=m)
  for (j in 1:m){
   m.start=m*(j-1)+1
   m.end=m.start+m-1
   trans_matrix_j[j,]<-trans.all[i,m.start:m.end]
  }
  for (j in 1:m){
   prob.states[i,j]<-prob.states[i-1,]%*% trans_matrix_j[,j]
  }
 }

 ###############observation densities################

 # convert to orig scale
 nc_prams<-(obs_density_sq$mean/obs_density_sq$sd)^2
 var<-obs_density_sq$sd^2

 obs_density<-matrix(NA,m,3)
 # 5th, 50th and 95th of obs density in orig scale
 for (i in 1:m){
  obs_density[i,]<- qchisq(p=c(0.05,0.5,0.95), df=1, ncp = nc_prams[i])*var[i]
 }

 return(list(ML_states=ML_states, prob_ML_states=prob_ML_states,
        transition_prob=trans.all,
        circadian_states_prob=data.frame(prob.states),
        AIC=AIC(HMM),BIC=BIC(HMM),
        obs_density=obs_density,obs_density_sq=obs_density_sq))
}
```

function: figure_HMM_decoding_SP

```r
#' This function generates plots like Figure 5&6 in the main manuscript
#' @param HMM_results: the results of functions Harmonic_HMMs or
Homogeneous_HMMs
#' @param info: time series of activity
#' @param id: subject id used in caption
#' @param n_states: we assume 3 states

figure_HMM_decoding_SP<-function(HMM_results,info, id,n_states=3){

  ##############################################
  #Parameters setting
  ##############################################

time<- as.POSIXct(info$time, format="%Y-%m-%d %H:%M:%S") #get appropriate
format for plotting
  obs<-info$activity
  prob_ML_states<-HMM_results$prob_ML_states
  ML_states<-HMM_results$ML_states

  L<-length(time)
  obs_max<-max(obs,na.rm=T)
  level_states<-c(0,obs_max/3,obs_max*2/3)



  ##############################################
  #local decoding
  ##############################################


  par(mar=c(2,4,3,0.7), mgp=c(1,.4,0), tck=-.01)
  par(fig=c(0,1,0.4,1), new=F)

  step_obs<-10
  if (obs_max>100){step_obs<-20}
  if (obs_max>200){step_obs<-40}


plot(time,obs,xaxt='n',yaxt="n",ylab="",xlab="",ylim=range(0,obs_max),pch=19,cex=
0.5, col=gray(0.5))

  ML_states_line<-rep(NA,L)
  for (i in 1:L){
    ML_states_line[i]<-level_states[ML_states[i]]
  }
  lines(time, ML_states_line,type='l',col='gold', lwd=2)

  legend('topright', c("5 min aggregated accelerometer data") , col=gray(0.5),
pch=19, cex=0.85)
  legend('topleft', c("states estimation") , col='gold', lty=1, lwd=2 , cex=0.85)

  axis(side=2, at=seq(0,obs_max,step_obs), labels=T, cex.axis      =0.85, las=2)
  axis.POSIXct(side=1, at=cut(time, "days"), format="%d/%m",cex.axis =0.85)
  mtext('Accelerations/minute', side=2, line=2,cex=1)
  mtext(paste0('Local Decoding of Subject ',id), side=3, line=1,cex=1,font=2)

  ##############################################
  #legend of 3 states
  ##############################################
  par(fig = c(0.05, 0.95, 0, 0.41), oma = c(0, 0, 0, 0), mar = c(0, 1, 0, 0), new = TRUE)
  plot(0, 0, type = "n", bty = "n", xaxt = "n", yaxt = "n")
  legend("topleft",  c(" inactive state"),
       xpd = TRUE, bty = "n", lty=c(1),col = c(rgb(0,0,1,0.8)), cex = 0.85,lwd=2)
  legend("topright",  c( "moderately active state", "highly active state"),
       xpd = TRUE,
       bty = "n", lty=c(1,1),col = c(rgb(1,0,0,0.4),rgb(1,0,0,0.8)), cex = 0.85,lwd=2)

  ##############################################
  #state probability (SP) of 3 states probabilities
  ##############################################

  par(fig=c(0,1,0,0.37), new=TRUE)

  par(mar=c(4,4,1,0.7), mgp=c(.5,1,0), tck=-.01)


  plot(time, rep(-2,L), ylim=c(0,1),xlab='', ylab='', xaxt='n',yaxt='n', cex=0.5,  las=1)

  plot_p<-matrix(NA,nrow=L-1,ncol=2*n_states)
  a<-0
  for(i in 2:L){
    for(j in 1:n_states){
      plot_p[i-1,(j*2-1)]<-prob_ML_states[i-1,j]
      plot_p[i-1,j*2]<-prob_ML_states[i,j]


      if (j==1){col_states<-rgb(0,0,1,0.8)}
      if (j==2){col_states<-rgb(1,0,0,0.4)}
      if (j==3){col_states<-rgb(1,0,0,0.8)}
      if (j==1){
        point_1<-a
        point_2<-point_1+plot_p[i-1,(j*2-1)]
        point_4<-a
        point_3<-point_4+plot_p[i-1,(j*2)] }

      if (j==2){
        point_1<-a+plot_p[i-1,(j-1)*2-1]
        point_2<-point_1+plot_p[i-1,(j*2-1)]
        point_4<-a+plot_p[i-1,(j-1)*2]
        point_3<-point_4+plot_p[i-1,(j*2)] }

      if (j==3){
        point_1<-a+plot_p[i-1,(j-2)*2-1]+plot_p[i-1,(j-1)*2-1]
        point_2<-point_1+plot_p[i-1,(j*2-1)]
        point_4<-a+plot_p[i-1,(j-2)*2]+plot_p[i-1,(j-1)*2]
        point_3<-point_4+plot_p[i-1,(j*2)]}

      polygon(c(time[i-1],time[i-
1],time[i],time[i]),c(point_1,point_2,point_3,point_4),col=col_states,border=NA)
      lines(c(time[i-1],time[i]),c(point_2,point_3),  col=col_states)
    }
  }

  axis(side=2, at=seq(0,1,0.2), labels=T, cex.axis     =0.85, las=2)
  axis.POSIXct(side=1, at=cut(time, "days"), format="%d/%m",cex.axis =0.85)
  mtext('Probability', side=2, line=2,cex=1)
  mtext('Date', side=1, line=2,cex=1)
}
```

function: figure_day_profile

```
#' This function generates day profile plots like Figure 7 in the main manuscript
#' @param one_day_prob: selected one day circadian state probabilities
#' @param id: subject id used in caption
#' @param index: x coordinates of points in the plot
#' @param index_position/index_lable (X-axis parameters):  position of tic marks
and labels(clock time used here)
figure_day_profile<-function(id,one_day_prob,index,index_position,index_lable){

 #######24h oscillated state probability plot ################

 SP<-function(one_day_prob,n_states=3,L=288){

  L<-length(index)
  plot_p<-matrix(NA,nrow=L-1,ncol=2*n_states)

  a<-0
  for(i in 2:L){
   for(j in 1:n_states){
    plot_p[i-1,(j*2-1)]<-one_day_prob[i-1,j]
    plot_p[i-1,j*2]<-one_day_prob[i,j]


    if (j==1){col_states<-rgb(0,0,1,0.8)}
    if (j==2){col_states<-rgb(1,0,0,0.4)}
    if (j==3){col_states<-rgb(1,0,0,0.8)}

    if (j==1){
     point_1<-a
     point_2<-point_1+plot_p[i-1,(j*2-1)]
     point_4<-a
     point_3<-point_4+plot_p[i-1,(j*2)] }

    if (j==2){
     point_1<-a+plot_p[i-1,(j-1)*2-1]
     point_2<-point_1+plot_p[i-1,(j*2-1)]
     point_4<-a+plot_p[i-1,(j-1)*2]
     point_3<-point_4+plot_p[i-1,(j*2)] }

    if (j==3){
     point_1<-a+plot_p[i-1,(j-2)*2-1]+plot_p[i-1,(j-1)*2-1]
     point_2<-point_1+plot_p[i-1,(j*2-1)]
     point_4<-a+plot_p[i-1,(j-2)*2]+plot_p[i-1,(j-1)*2]
     point_3<-point_4+plot_p[i-1,(j*2)]}
```

```
      polygon(c(index[i-1],index[i-
1],index[i],index[i]),c(point_1,point_2,point_3,point_4),col=col_states,border=NA)
      lines(c(index[i-1],index[i]),c(point_2,point_3),  col=col_states)
     }
    }
    axis(side=2, at=seq(0,1,0.2), labels=T, cex.axis   =1, las=2)
    axis(side=1, at=index_position, labels=index_lable, cex.axis     =1, las=1)

    mtext('Probability', side=2, line=2)
  }

  plot(index, rep(-2,length(index)), ylim=c(0,1),xlab='', ylab='', xaxt='n',yaxt='n',
cex=0.5,  las=1)
  SP(one_day_prob)
  mtext(paste(' Day Profile of Subject',id), side=3,font = 2)
  mtext('Clock Time (One Day)', side=1,cex=1, line=2)

}
```