Synchrony in relationship, example with MONRADO Data

Thomas Gargot

May, 10th 2016

Contents

```
# Clean Environement
rm(list = ls(all.names = TRUE))
```

Fixed variables

```
FileExtension <- ".MTS.avi_res.csv"

# working directory
# where this report is
setwd("/Users/Ofix/Documents/Fac/internat/Recherche/projets/synchro/synchroData/Git/Monrado/Reports/")

# blue will refer to father
# red will refer to mother
# green to child
colOrderList <- c("blue","red", "green")

ParticipantsList <- c("father", "mother", "child")
```

```
## Create a csv files list with the directories
FullNameList <- list.files("../Data/CSV/raw", full.names=TRUE)
FullNameList

## Create a csv files list without the directories
filesList <- list.files("../Data/CSV/raw", full.names=FALSE)
filesList
```

Functions list

Import Data List

Function that import data from .csv files inside a CSV folder ##### Arguments: List FullNameList with the full name of the .csv

```
importdata <-function(FullnameList){
    data <- c()
    for (i in FullnameList){
        dataAlone <- read.csv(i)
        print(head(dataAlone[,c(2:5)]))
        mydata.nas <- apply(dataAlone[,c(2:5)], 1, function(x){all(is.na(x))})
        dataAlone <- dataAlone[!mydata.nas,]
        print(i)
        data <- rbind(data, dataAlone)
    }
    return (data)
}
```

MeanMotionByTime

Function that takes raw motion history data and compute the mean on a given interval. Intervals don't overlap, so the frequency of the data change (from 25 frames by seconde to 25 frames/interval by second).

Arguments:

- subject : Subject studied (patient, mother, father or therapist)
- indexOfvideos : List of videos studied (element eg 3 or list eg 1:3 or c(1,2,4))
- interval : number of frames in the studied interval
- data : data frame where there is data

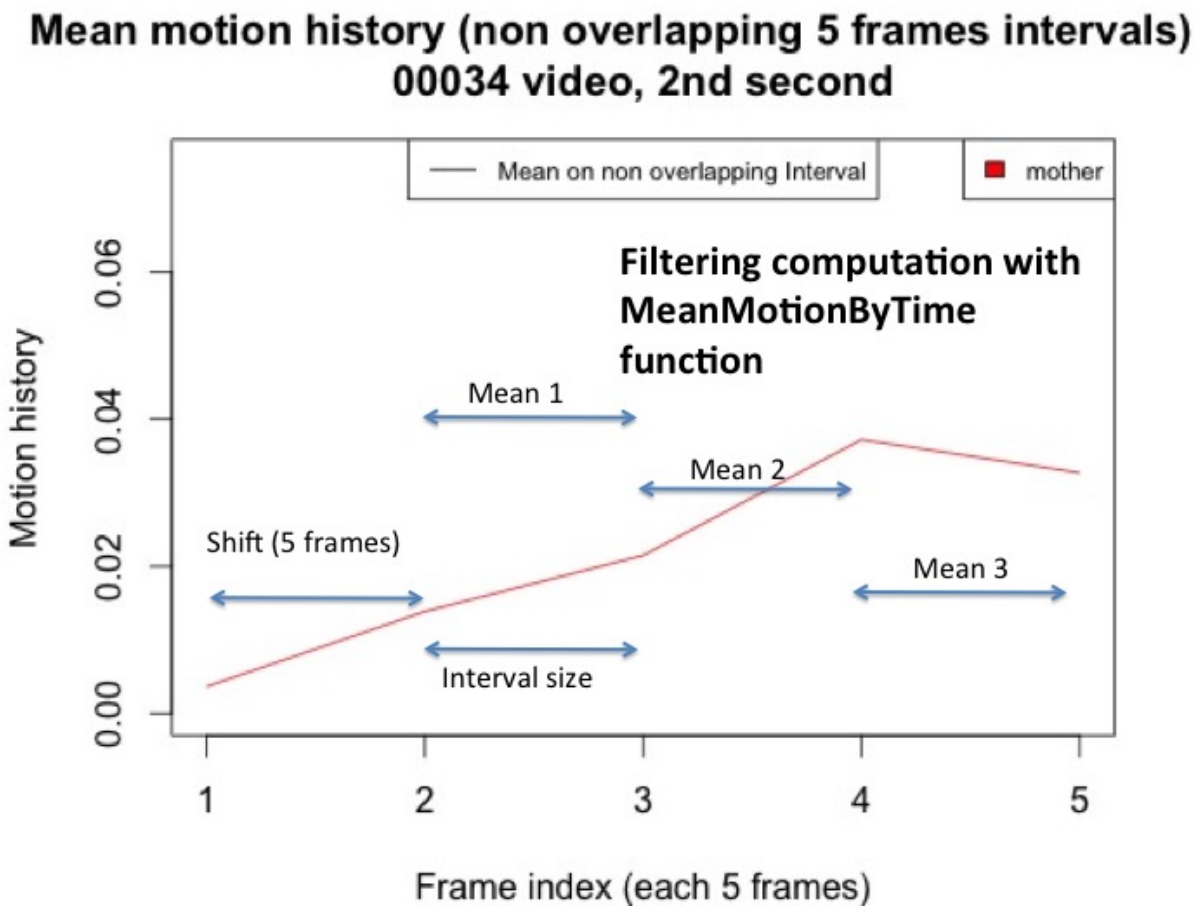


Figure 1:

```
MeanMotionByTime <- function(subject, indexOfvideos=1:NumberOfvideos, interval, data){
  x <- c()
  for (fam in families[indexOfvideos]){
        dataVector <- data[which(data$family==fam), subject]
        ## with ceiling : superior limit of the round
        IntervalNumbersVideo <- ceiling(length(dataVector)/interval)
        for (i in 1:IntervalNumbersVideo){
                borneinf<- 1+(i-1)*interval
                bornesup <-i*interval
```

```
                dataVectorInterval <- dataVector[borneinf:bornesup]
                mean <- mean(dataVectorInterval, na.rm=TRUE)
                x <- c(x, mean)}}
  return (x)}
```

Slidinginterval

Function that takes raw motion history data and compute the mean on a given interval. The interval overlap, so the frequency of the data don't change. It stays at 25 frames/s.

Arguments:

- subject : subject studied (patient, mother, father or therapist)
- indexOfvideos : list of videos studied (element eg. 3 or list eg 1:3 or c(1,2,4))
- interval : number of frames in the studied interval
- data : data frame where there is data

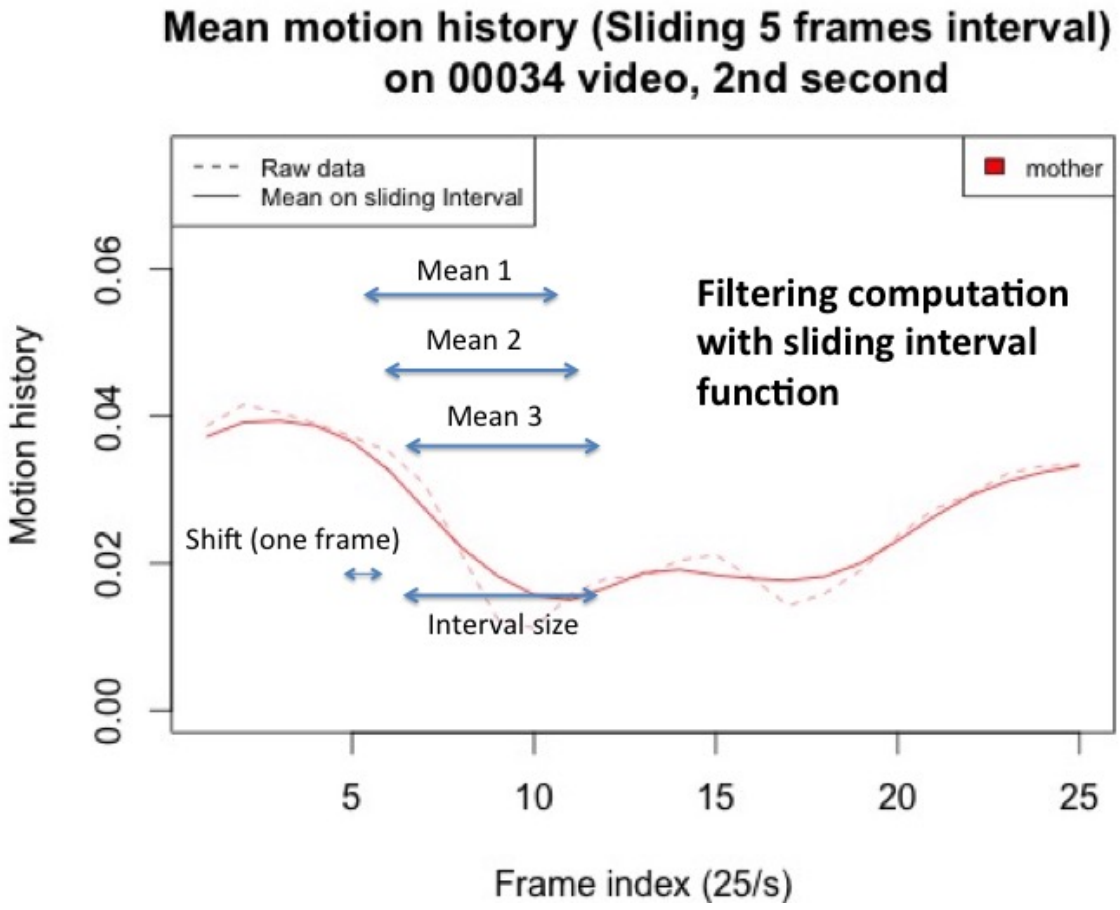


Figure 2:

```
SlidingInterval <- function(subject, indexOfvideos=1:NumberOfvideos, interval, data)
  {x <- c()
  for (file in families[indexOfvideos]){
```

```
      dataVector <- data[which(data$family==file), subject]
#      print(str(dataVector))
      NBofAnalysedFrames <- length(dataVector)-interval+1
#      print(NBofAnalysedFrames)
         for (i in 1:NBofAnalysedFrames){
             borneinf <- (i)
#             print(borneinf)
             bornesup <-(interval-1+i)
#             print(bornesup)
             dataVectorInterval <- dataVector[borneinf:bornesup]
             mean <- mean(dataVectorInterval, na.rm=TRUE)
             x <- c(x, mean)}}
  return (x)}
```

MeanSynchronyByTime (TODO)

Import data

```
data <- importdata(FullNameList)
```

Clean dataframe

Add new columns: compute minutes and log on data frame

```
# Detete No relevant subject here
data$therapist <- NULL

# compute time in minute
data$timeMin <- data$frame/(25*60)

## Create a list of files without the extention of the video
families <- c()
for (i in filesList){
    name <- sub(FileExtension, "", i)
    families <- c(families, name)
}
families
```

```
##  [1] "00034"   "00037"   "00041"   "00048"   "0206"    "1106"    "1606"
##  [8] "BAJE059" "BALE050" "BALU062" "BEAL036" "BEAM031" "BICA"    "BRLO041"
## [15] "COLO022" "DIPE004" "DOMA"    "DRNE"    "FOMA057" "GROP039" "HAJA052"
## [22] "HUMA058" "JAEM046" "JEEO040" "JOCE014" "LACL"    "MAEL048" "MAME20"
## [29] "MAPA029" "MIPH043" "MOSA065" "RAEM049" "RAKA008" "RIEM0"   "SEEM035"
## [36] "SHAN042" "SOGA061" "TIUG032" "VINO"
```

```
NumberOfvideos <- length(families)
NumberOfvideos
```

```
## [1] 39
```

```
# create a list with the simplified dname (whitout extension), make a data frmae of it  and merge 2 dat
a <- data.frame(family = families, unique(data$file))
data <- merge(data, a, by.x="file", by.y="unique.data.file.")
```

```
data$fatherShifted <- data$father + min(data$father[which (data$father >0)])/2
data$logFather <- log(data$fatherShifted)

data$motherShifted <- data$mother + min(data$mother[which (data$mother >0)])/2
data$logMother <- log(data$motherShifted)

data$childShifted <- data$child + min(data$child[which (data$child >0)])/2
data$logChild <- log(data$childShifted)

# Add date TODO
data$file <- NULL

data <- data[,c("family", "frame", "timeMin", "child", "childShifted", "logChild", "father", "fatherShi
```

Presentation of the data

```
str(data)
```

```
## 'data.frame':    914108 obs. of  12 variables:
##  $ family       : Factor w/ 39 levels "00034","00037",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ frame        : int  1 2 3 4 5 6 7 8 9 10 ...
##  $ timeMin      : num  0.000667 0.001333 0.002 0.002667 0.003333 ...
##  $ child        : num  1.62e-04 1.89e-04 7.50e-05 5.36e-05 8.04e-05 ...
##  $ childShifted : num  1.62e-04 1.89e-04 7.55e-05 5.40e-05 8.08e-05 ...
##  $ logChild     : num  -8.73 -8.57 -9.49 -9.83 -9.42 ...
##  $ father       : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ fatherShifted: num  NA NA NA NA NA NA NA NA NA NA ...
##  $ logFather    : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ mother       : num  4.00e-04 4.56e-04 2.23e-04 8.85e-05 9.58e-05 ...
##  $ motherShifted: num  4.01e-04 4.57e-04 2.24e-04 8.89e-05 9.61e-05 ...
##  $ logMother    : num  -7.82 -7.69 -8.41 -9.33 -9.25 ...
```

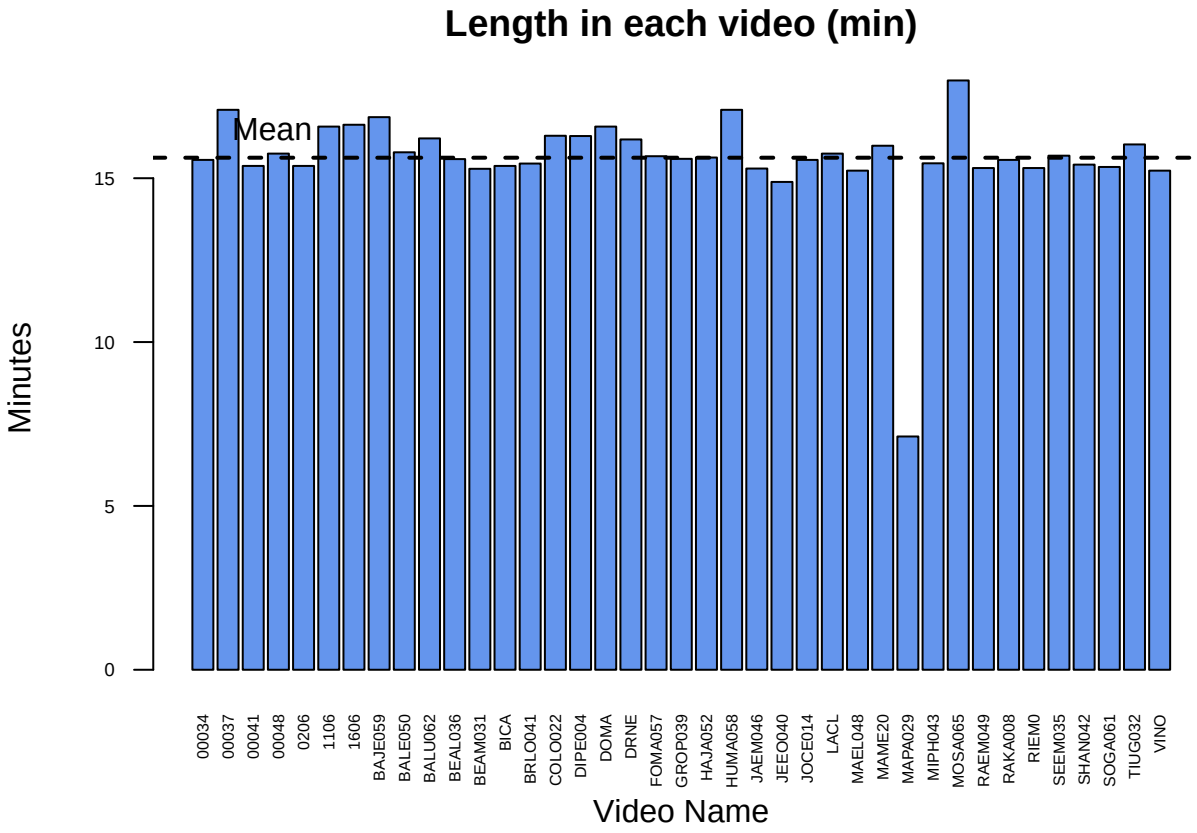
```
summary(data)
```

```
##      family           frame          timeMin              child
##  MOSA065: 26975   Min.   :    1   Min.   : 0.000667   Min.   :0.0000000
##  00037  : 25631   1st Qu.: 5860   1st Qu.: 3.906667   1st Qu.:0.0006479
##  HUMA058: 25631   Median :11747   Median : 7.831333   Median :0.0034404
##  BAJE059: 25295   Mean   :11831   Mean   : 7.887165   Mean   :0.0093006
##  1606   : 24947   3rd Qu.:17761   3rd Qu.:11.840667   3rd Qu.:0.0117136
##  1106   : 24863   Max.   :26975   Max.   :17.983333   Max.   :0.9270200
##  (Other):760766
##   childShifted          logChild             father       fatherShifted
##  Min.   :0.0000004   Min.   :-14.66620   Min.   :0.0      Min.   :0.0
##  1st Qu.:0.0006483   1st Qu.: -7.34117   1st Qu.:0.0      1st Qu.:0.0
##  Median :0.0034409   Median : -5.67204   Median :0.0      Median :0.0
##  Mean   :0.0093010   Mean   : -6.19280   Mean   :0.0      Mean   :0.0
##  3rd Qu.:0.0117140   3rd Qu.: -4.44697   3rd Qu.:0.0      3rd Qu.:0.0
##  Max.   :0.9270204   Max.   : -0.07578   Max.   :0.1      Max.   :0.1
##                                          NA's   :742791   NA's   :742791
##    logFather          mother       motherShifted      logMother
##  Min.   :-14.6    Min.   :0.00     Min.   :0.00     Min.   :-14.78
```

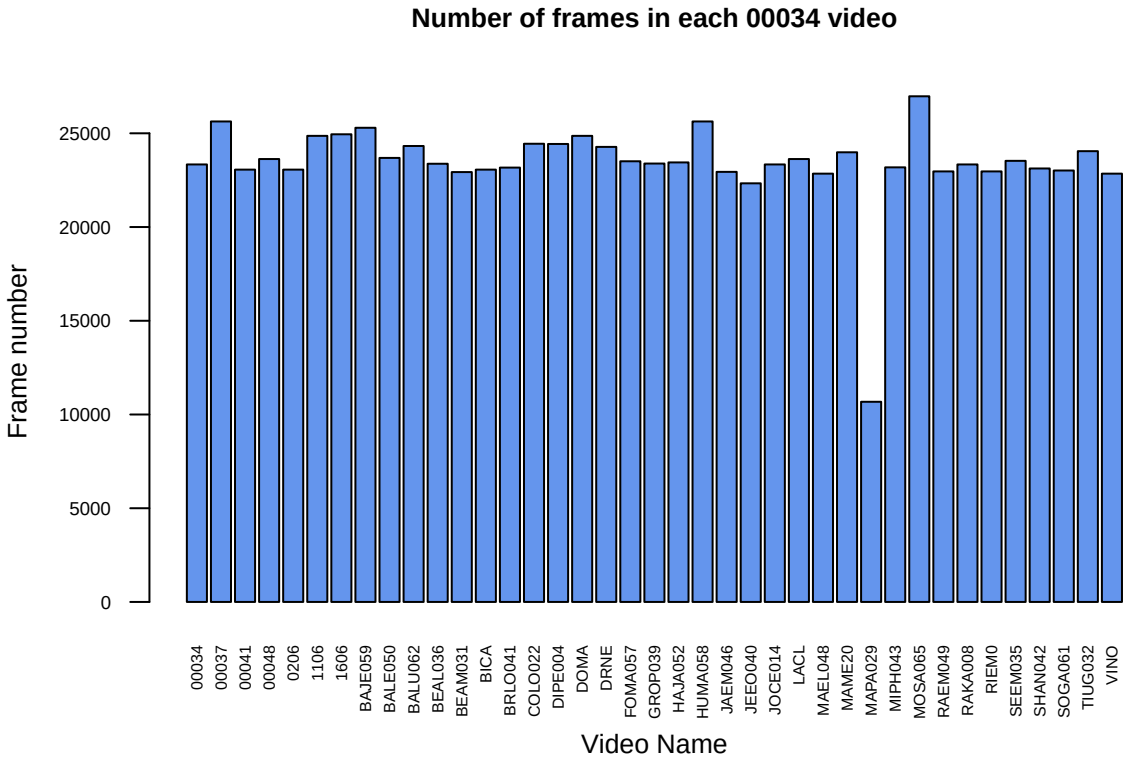
```
##  1st Qu.: -9.2    1st Qu.:0.00     1st Qu.:0.00     1st Qu.: -8.41
##  Median : -6.8    Median :0.00     Median :0.00     Median : -6.44
##  Mean   : -7.3    Mean   :0.01     Mean   :0.01     Mean   : -7.03
##  3rd Qu.: -5.1    3rd Qu.:0.01     3rd Qu.:0.01     3rd Qu.: -5.09
##  Max.   : -2.0    Max.   :0.96     Max.   :0.96     Max.   : -0.04
##  NA's   :742791   NA's   :147978   NA's   :147978   NA's   :147978
```

The timeMin is calculated with a frame rate of 25/sec.

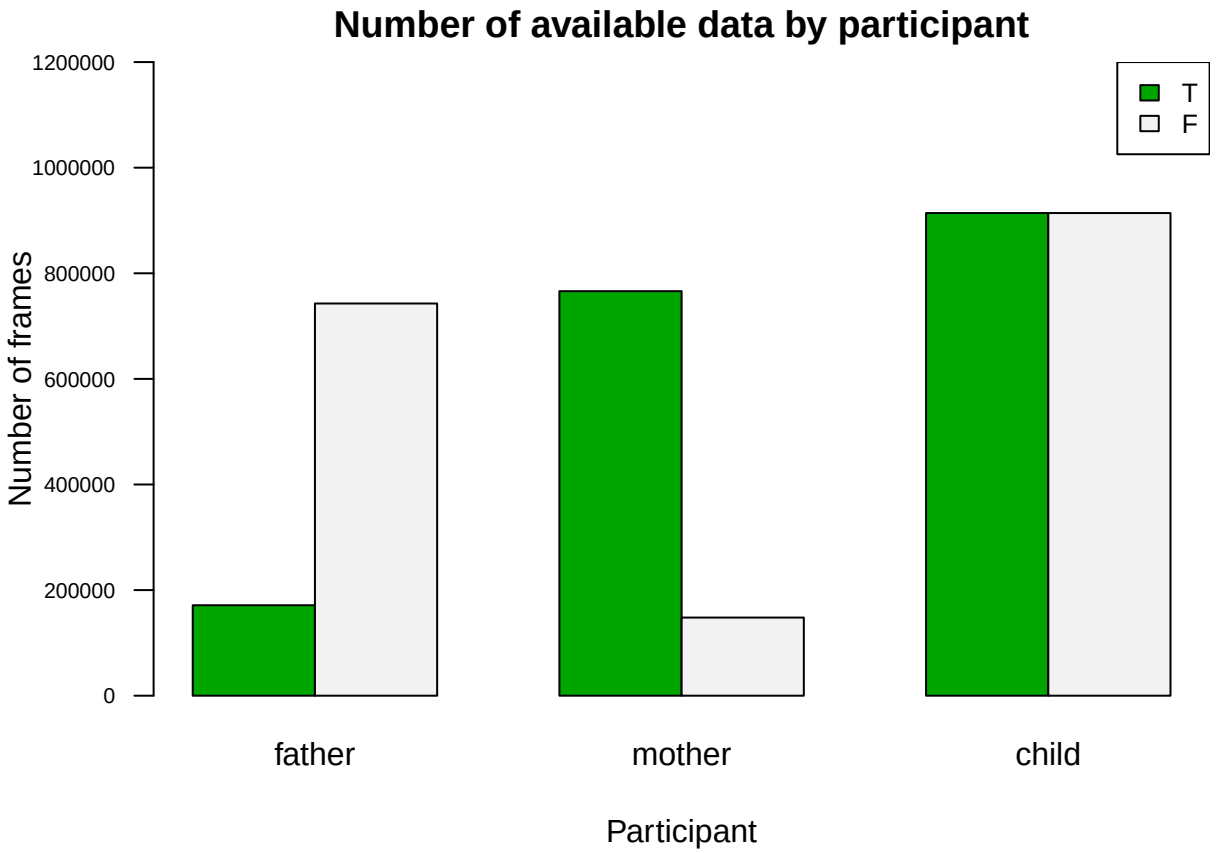
Length of the videos in minutes



Length of the videos in number of frames



Number of Available (True) and Not Available (False) data for each participant

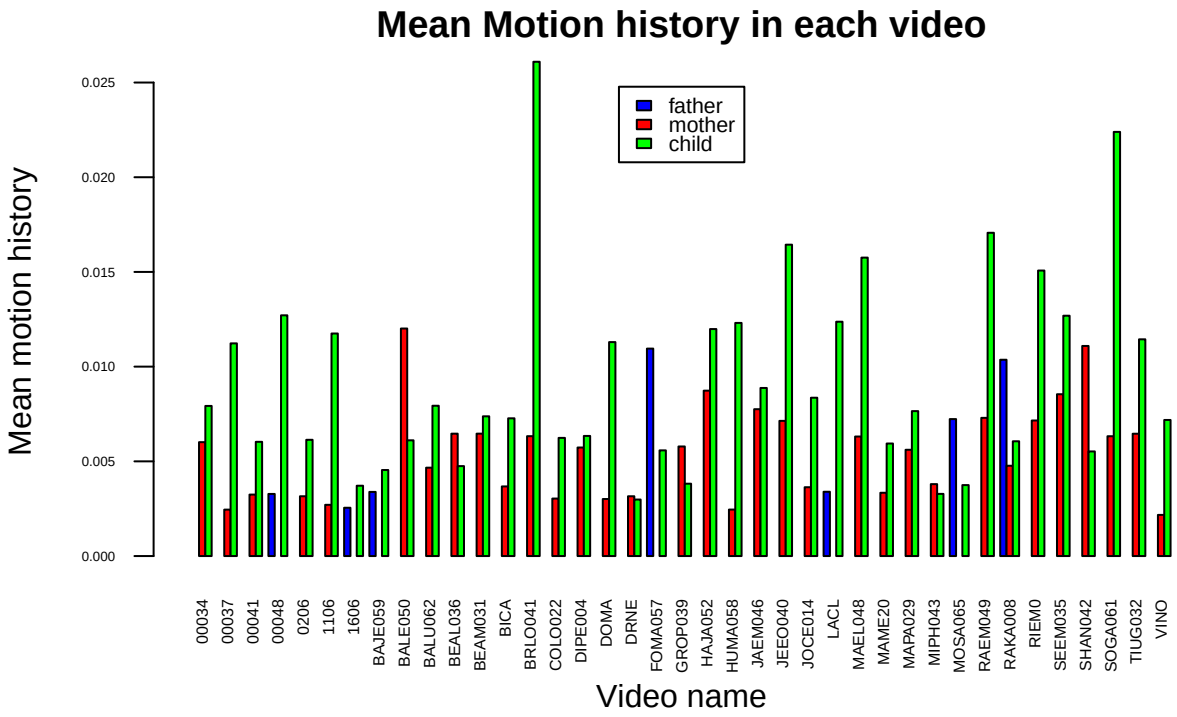


*

All the participants involved are filmed. * All the children are filmed and we have data dor each. * More often there is the other with him/her sometimes, it is the father * In some videos for instance RAKA008, there are 3 subjects

Global Motion history

Mean Motion history by video by participant



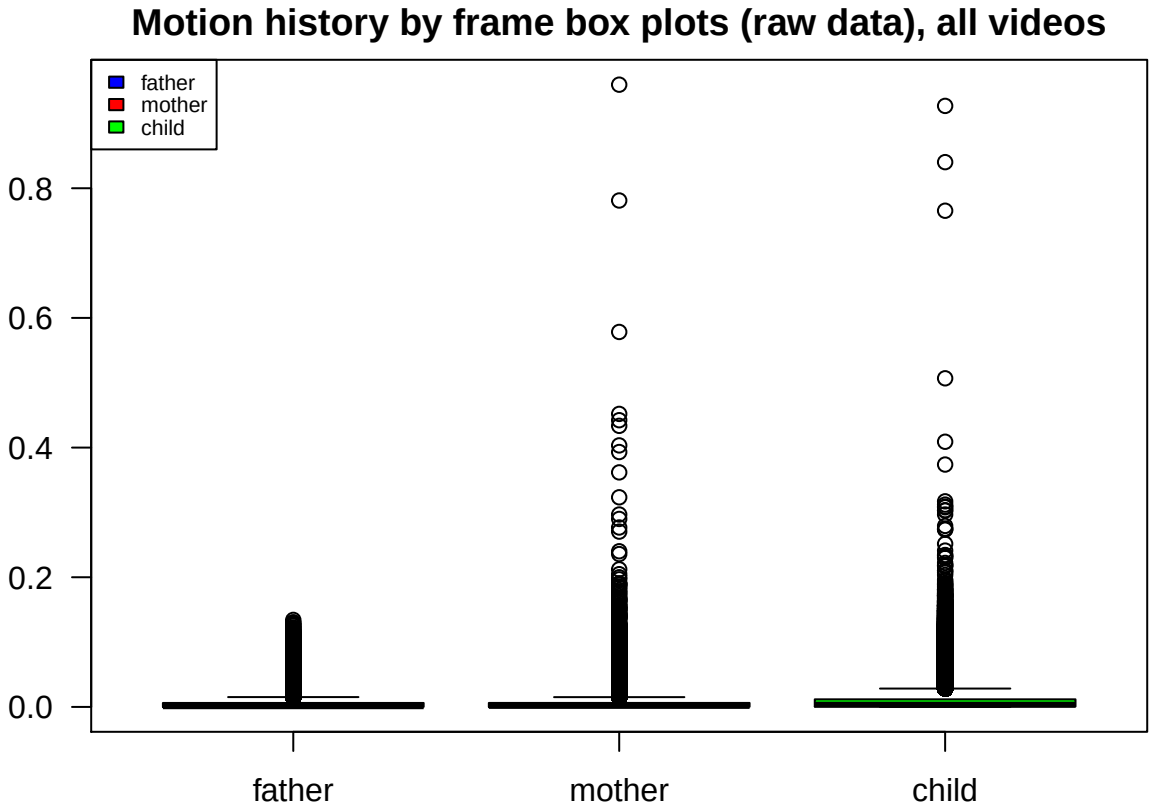
We can see that configurations of subjects are very similar (with always 2 subjects, except RAKA008 with 3 subjects). More often the child is with his mother. Consequently, it makes the comparaisons of the videos quite easy

Motion history box plots by frame (raw data), all videos

The motion history is not normalized at all. Most of motions are very small but some of them are much big (long tail). This is very usual with this algorithm extraction motion history.

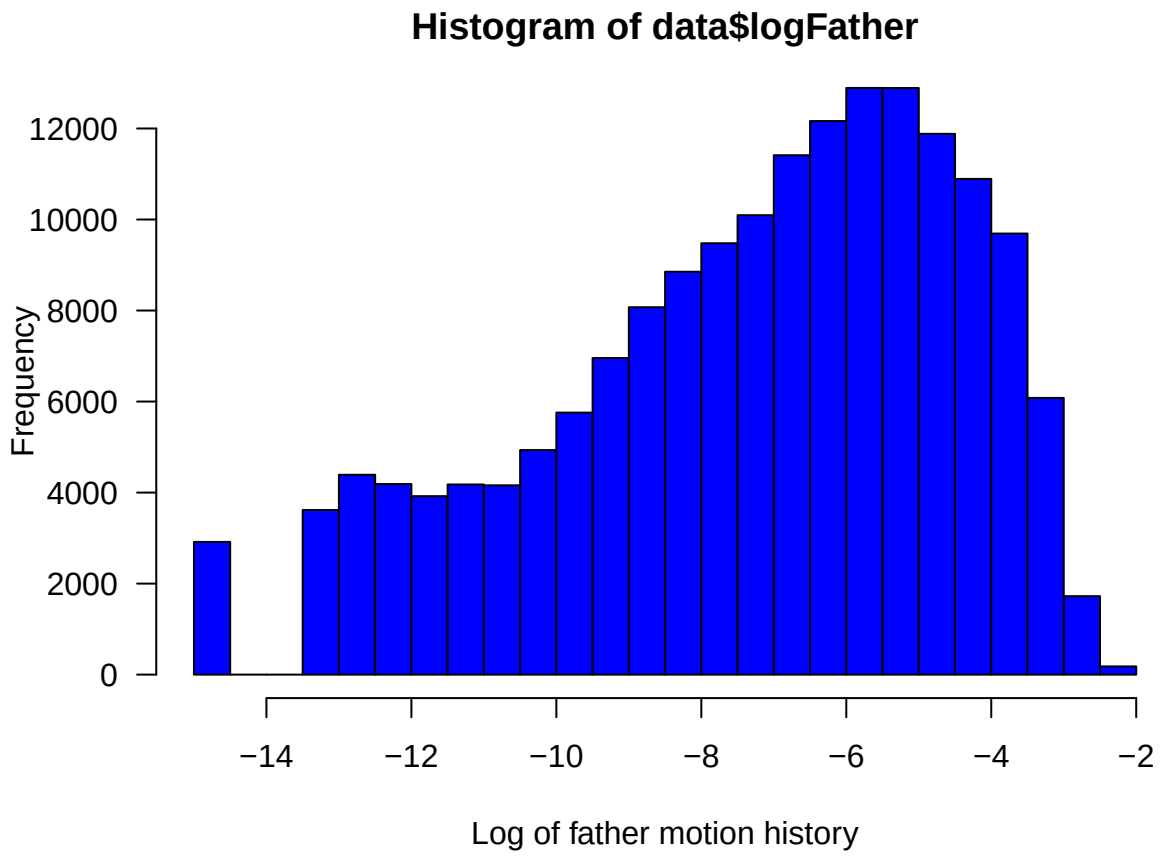
The subjects data are very similar.

```
par(mar=c(3,3,2,2))
boxplot(data$father, data$mother, data$child,
        col=colOrderList,
        names=ParticipantsList,
        main= "Motion history by frame box plots (raw data), all videos", las=1)
par(mar=c(1,0.5,0.5,1))
legend("topleft", ParticipantsList, fill=colOrderList, cex=0.7)
```

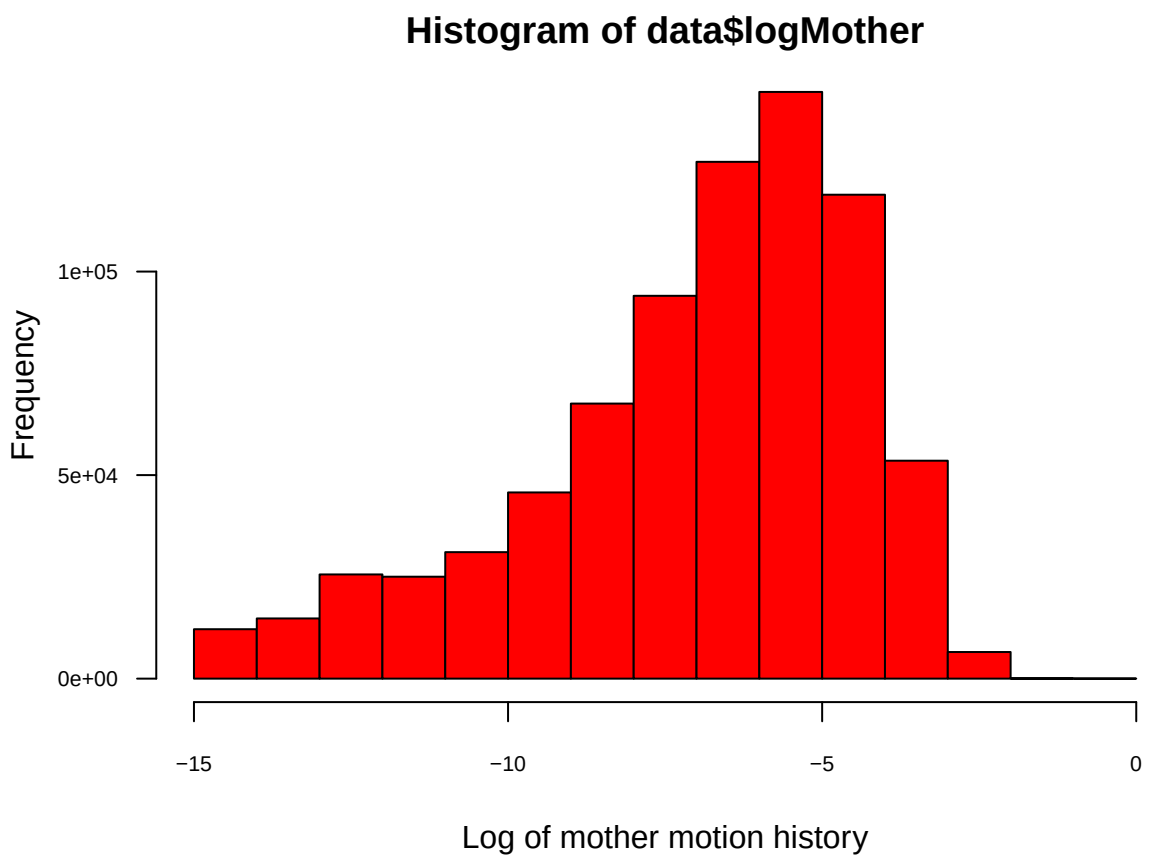


The subjects data distribution are very similar.

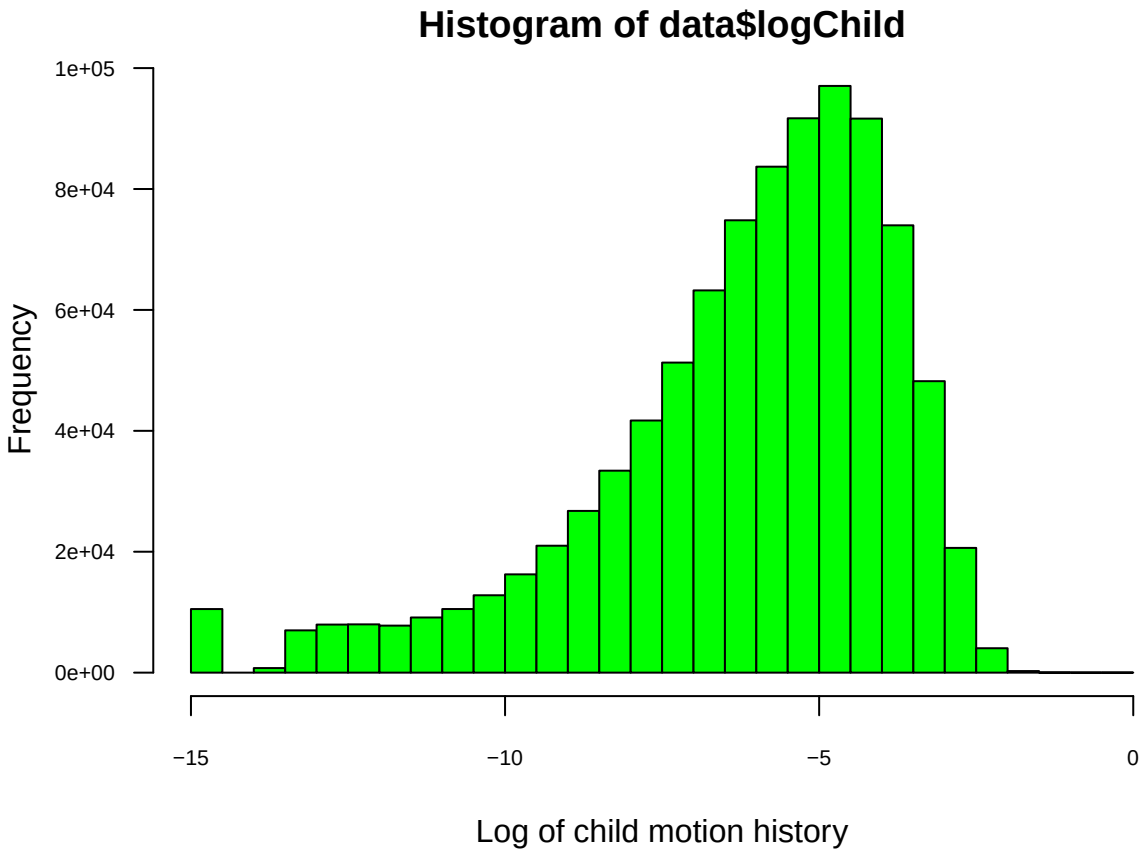
```
par(mar=c(4,4,2,2))
hist(data$logFather, col="blue", las=1, xlab="Log of father motion history")
```




```
hist(data$logMother, col="red", las=1, xlab="Log of mother motion history", cex.axis=0.7)
```



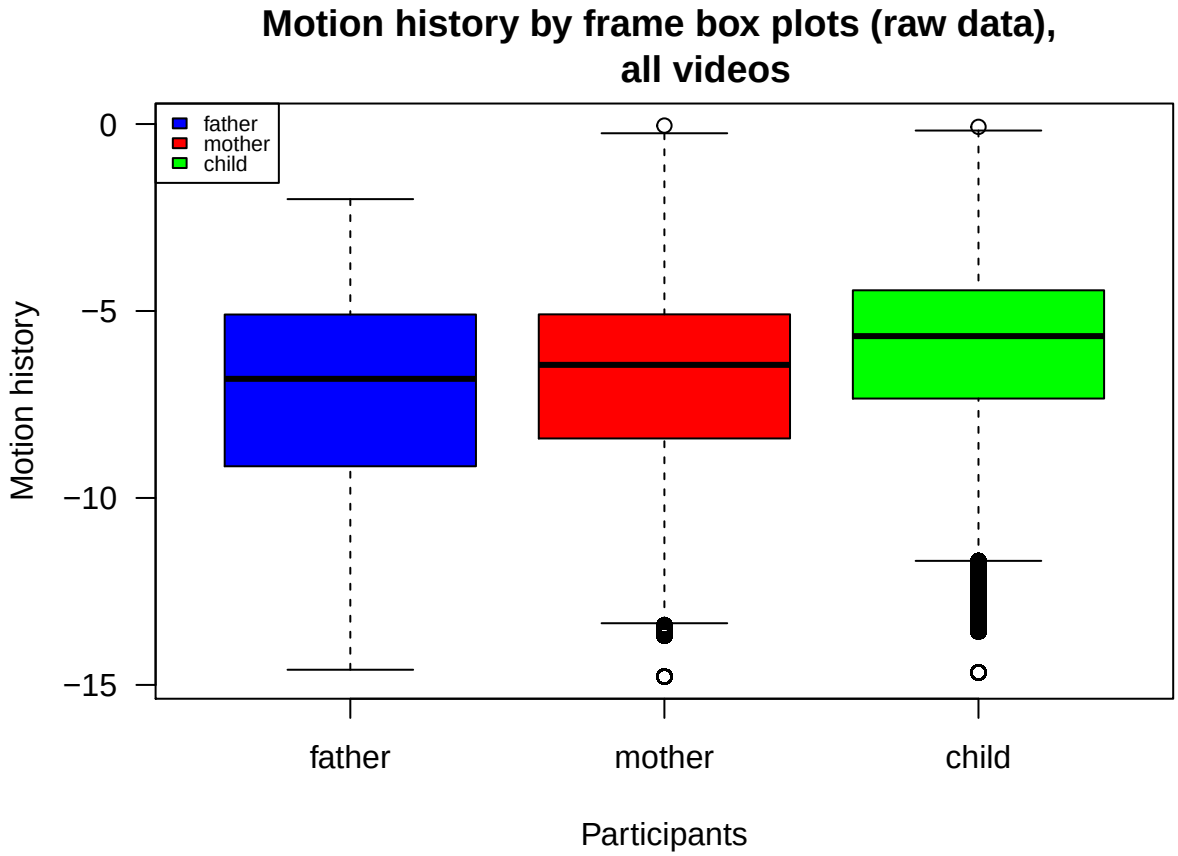

```
hist(data$logChild, col="green", las=1, xlab="Log of child motion history", cex.axis=0.7)
```



When doing the log, we almost normalized the distribution. We couldn't do the log on 0. The result would gave -Inf. We shifted all the distribution to the right by adding the half of the minimum after 0 of the distribution.

$\text{data}childShifted < -data\text{child} + \min(\text{data}child[which(data\text{child} > 0)])/2$

```
par(mar=c(4,4,3,2))
boxplot(data$logFather, data$logMother, data$logChild,
        col=colOrderList,
        names=ParticipantsList,
        main= "Motion history by frame box plots (raw data),
        all videos", las=1, xlab="Participants", ylab="Motion history")
par(mar=c(1,0.5,0.5,1))
legend("topleft", ParticipantsList, fill=colOrderList, cex=0.7)
```



Raw data and mean of Motion History on sliding and non overlapping intervals on 00034 video

It is the first video of 00034.

Raw data

```
rawdataMother <- data[which(data$family=="00034"),]$mother
rawdataChild <- data[which(data$family=="00034"),]$child

summary(rawdataMother)
```

```
##      Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
## 0.0000000 0.0004453 0.0026660 0.0060140 0.0085460 0.1735000
```

```
summary(rawdataChild)
```

```
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
## 0.000000 0.000537 0.002285 0.007926 0.008199 0.179000
```

Sliding interval

```
## REMINDER:
# SlidingInterval <- function(subject, indexOfvideos=1:NumberOfvideos, interval, data) with :
# subject : subject studied (patient, mother, father or therapist)
# indexOfvideos : list of videos studied (element eg. 3 or list eg 1:3 or c(1,2,4))
# interval : number of frames in the studied interval
# data : data frame where there is data
# repalce by 5 after
slidedMother <- SlidingInterval("mother", 1 , 5, data)
slidedChild <- SlidingInterval("child", 1 , 5, data)

summary(slidedMother)
```

```
##      Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
## 5.400e-07 5.367e-04 2.972e-03 6.015e-03 8.495e-03 1.616e-01
```

```
summary(slidedChild)
```

```
##      Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
## 0.0000000 0.0006375 0.0024400 0.0079270 0.0082810 0.1684000
```

Non overlapping interval

```
motherFive <- MeanMotionByTime("mother", indexOfvideos=1, interval=5, data)

childFive <- MeanMotionByTime("child", indexOfvideos=1, interval=5, data)

summary(childFive)
```

```
##      Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
## 2.100e-07 6.356e-04 2.417e-03 7.925e-03 8.280e-03 1.646e-01
```

```
summary(motherFive)
```

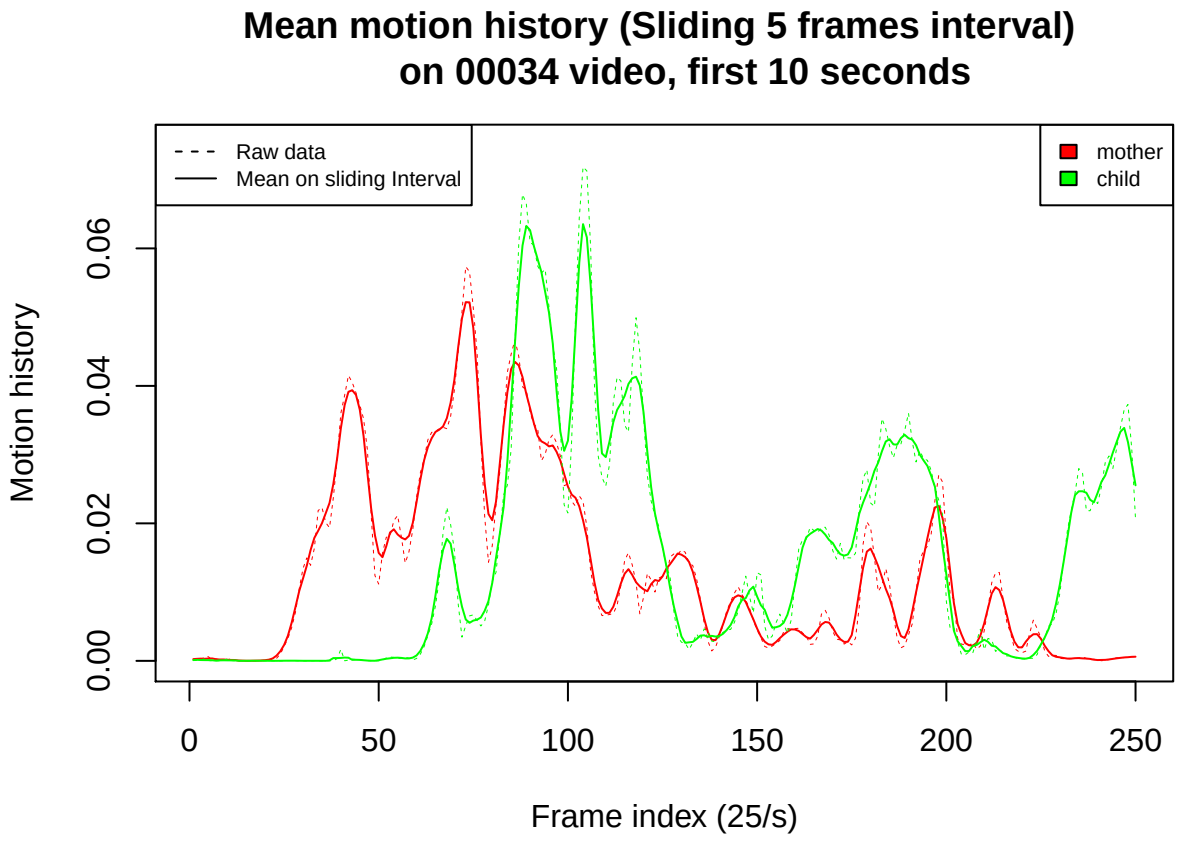
```
##      Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
## 5.400e-07 5.269e-04 3.015e-03 6.013e-03 8.504e-03 1.529e-01
```

Focus on the motion history of the first 10 seconds of the first video 00034

Sliding interval function on a 5 frames interval

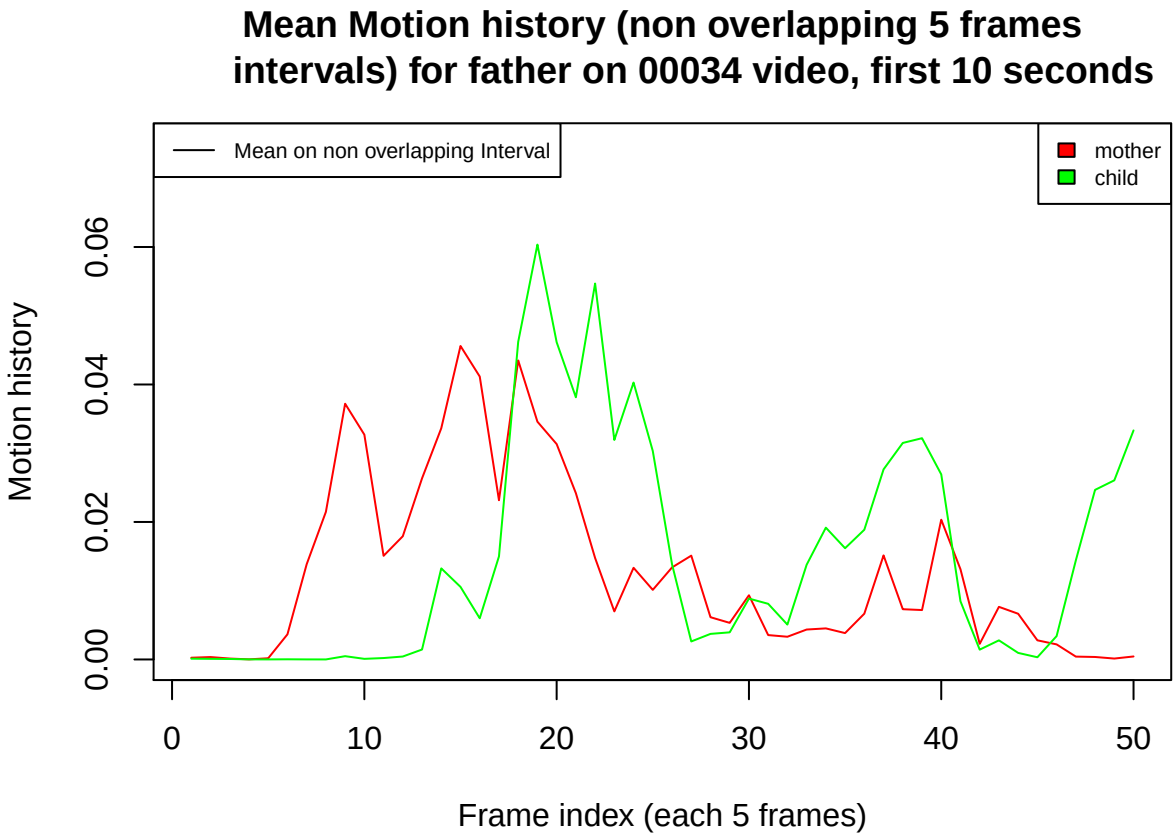
```
par(mar=c(4,4,4,2))
plot(1:250, data$mother[3:252], main="Mean motion history (Sliding 5 frames interval)
     on 00034 video, first 10 seconds ", xlab="Frame index (25/s)",
     ylab="Motion history",
     col="red", type="l", lty=2, lwd=0.5, ylim=c(0, 0.075))
lines(slidedMother[1:250],  col="red", lty=1)
lines(slidedChild[1:250],  col="green", lty=1)
lines(data$child[3:252],  col="green", lty=2, lwd=0.5)
```

```
legend("topleft", c("Raw data", "Mean on sliding Interval"), lty=c(2, 1), cex=0.7)
legend("topright", ParticipantsList[c(2,3)], fill=colOrderList[c(2,3)], cex=0.7)
```



Non overlapping interval function on a 5 frames interval

```
par(mar=c(4,4,4,2))
plot (1:50, motherFive[1:50], type="l", col="red",
main="Mean Motion history (non overlapping 5 frames
      intervals) for father on 00034 video, first 10 seconds",
ylab="Motion history", xlab="Frame index (each 5 frames)", ylim=c(0, 0.075))
lines(childFive[1:50], col="green", lty=1)
legend("topleft", "Mean on non overlapping Interval" , lty=1, cex=0.7)
legend("topright", ParticipantsList[c(2,3)], fill=colOrderList[2:3], cex=0.7)
```



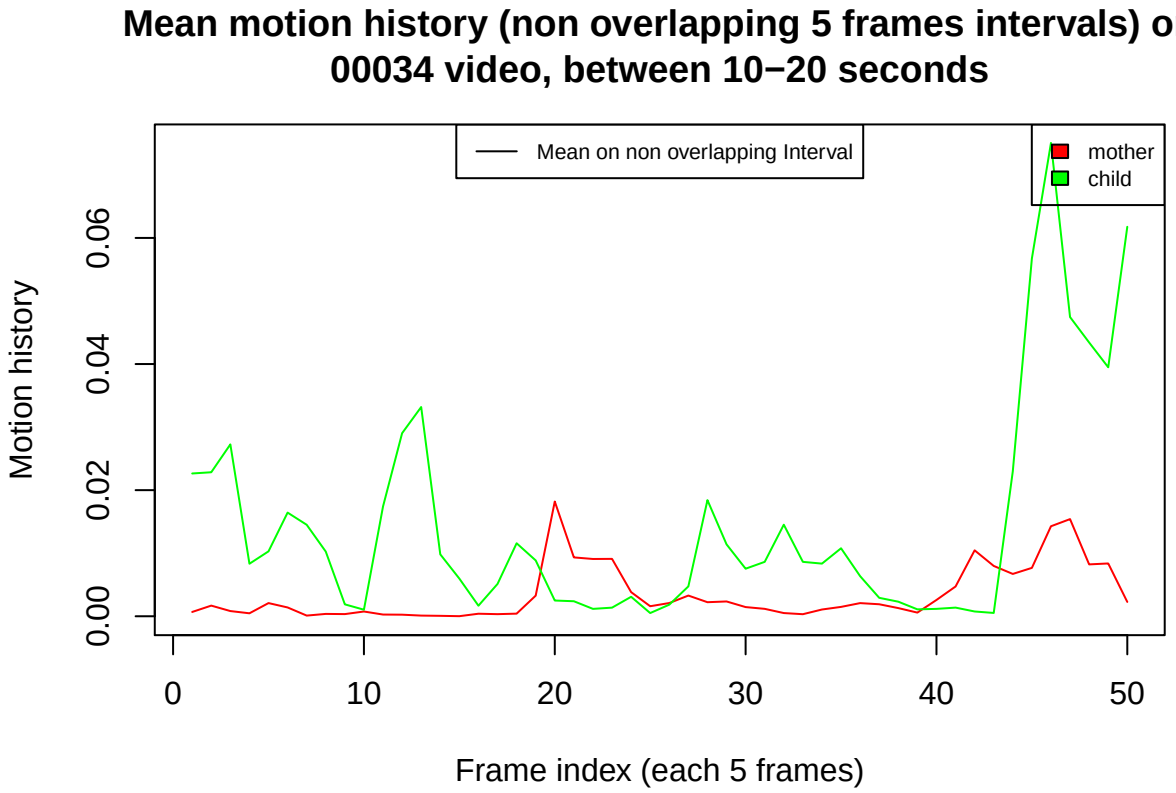
Motion history of the father during 10-20 seconds of the first video 00034

Non overlapping interval function on a 5 frames interval

```
par(mar=c(4,4,4,2))
plot(1:250, data$mother[253:502], main="Mean motion history (Sliding 5 frames
     interval) for father on 00034 video, 10-20 seconds", xlab="Frame index (25/s)",
     ylab="Motion history", col="red", type="l", lty=2, lwd=0.5, ylim=c(0, 0.075))
lines(slidedMother[251:500],  col="red", lty=1)
lines(data$child[253:502],  col="green", lty=2, lwd=0.5)
lines(slidedChild[251:500],  col="green", lty=1)
legend("topleft", c("Raw data", "Mean on sliding Interval") , lty=c(2, 1), cex=0.7)
legend("topright", ParticipantsList[c(2,3)], fill=colOrderList[c(2,3)], cex=0.7)
```

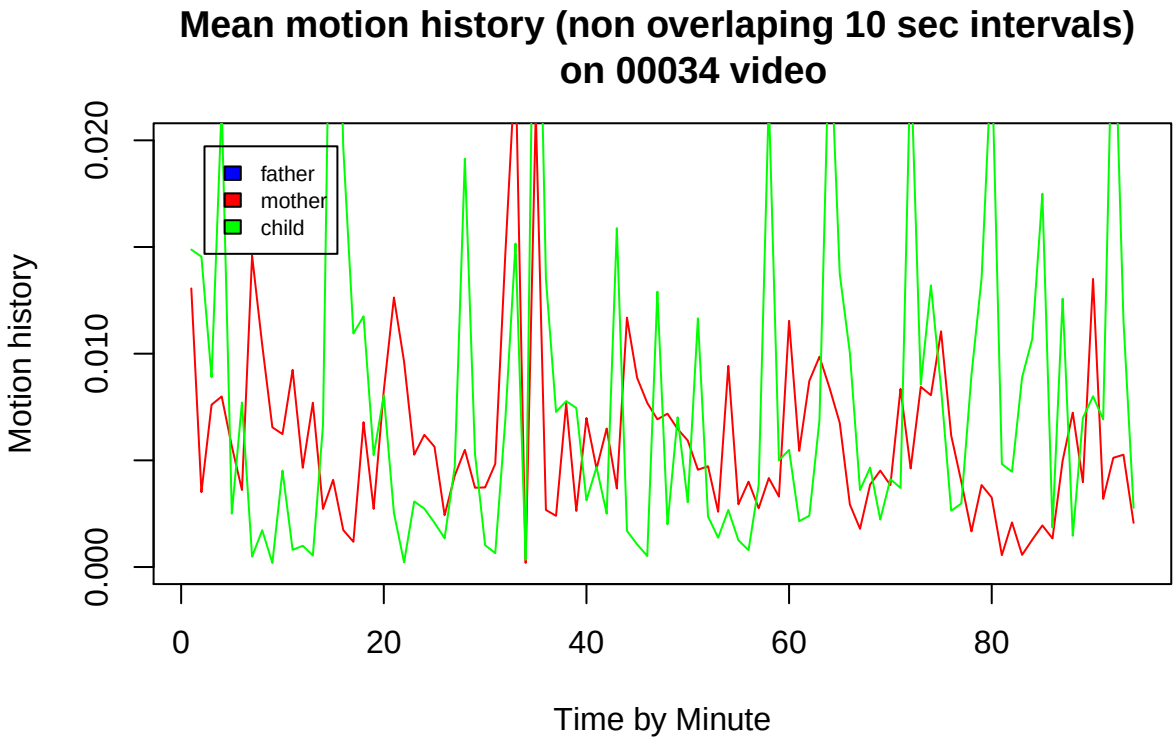
Non overlapping interval function on a 5 frames interval

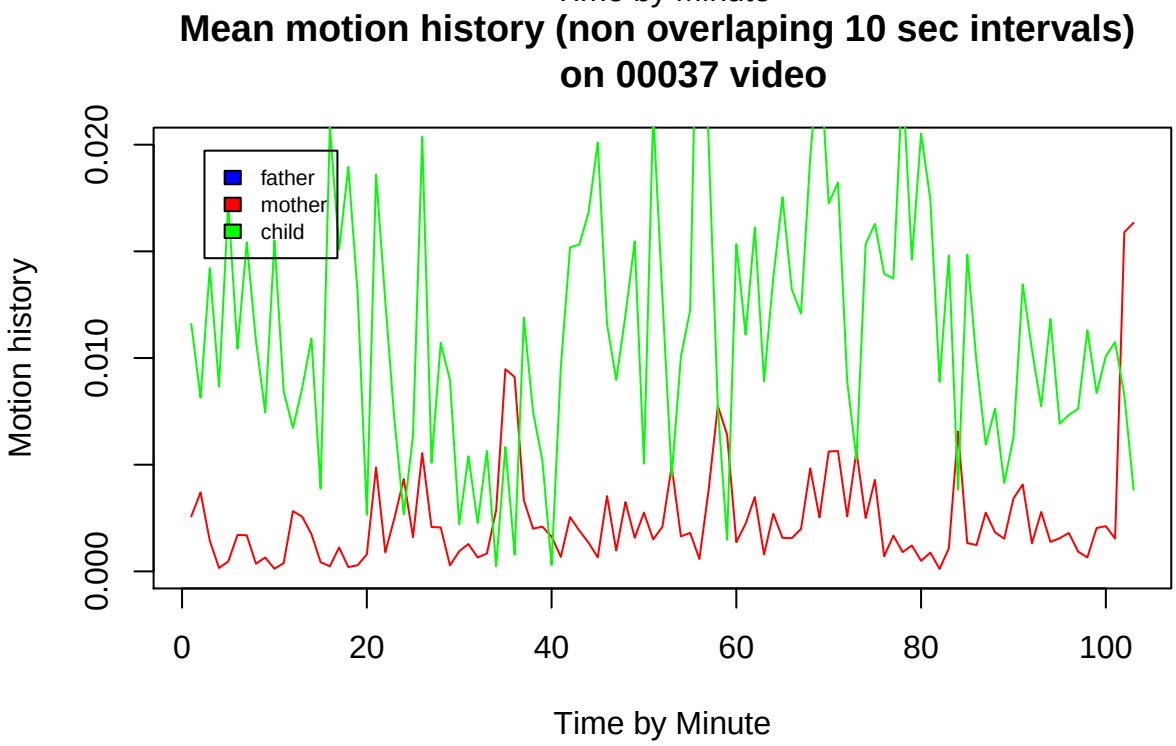
```
plot (1:50, motherFive[51:100], type="l", col="red",
main="Mean motion history (non overlapping 5 frames intervals) on
00034 video, between 10-20 seconds",
ylab="Motion history", xlab="Frame index (each 5 frames)", ylim=c(0, 0.075))
lines(childFive[51:100], col="green", lty=1)
legend("top", "Mean on non overlapping Interval" , lty=1, cex=0.7)
legend("topright", ParticipantsList[c(2,3)], fill=colOrderList[c(2,3)], cex=0.7)
```

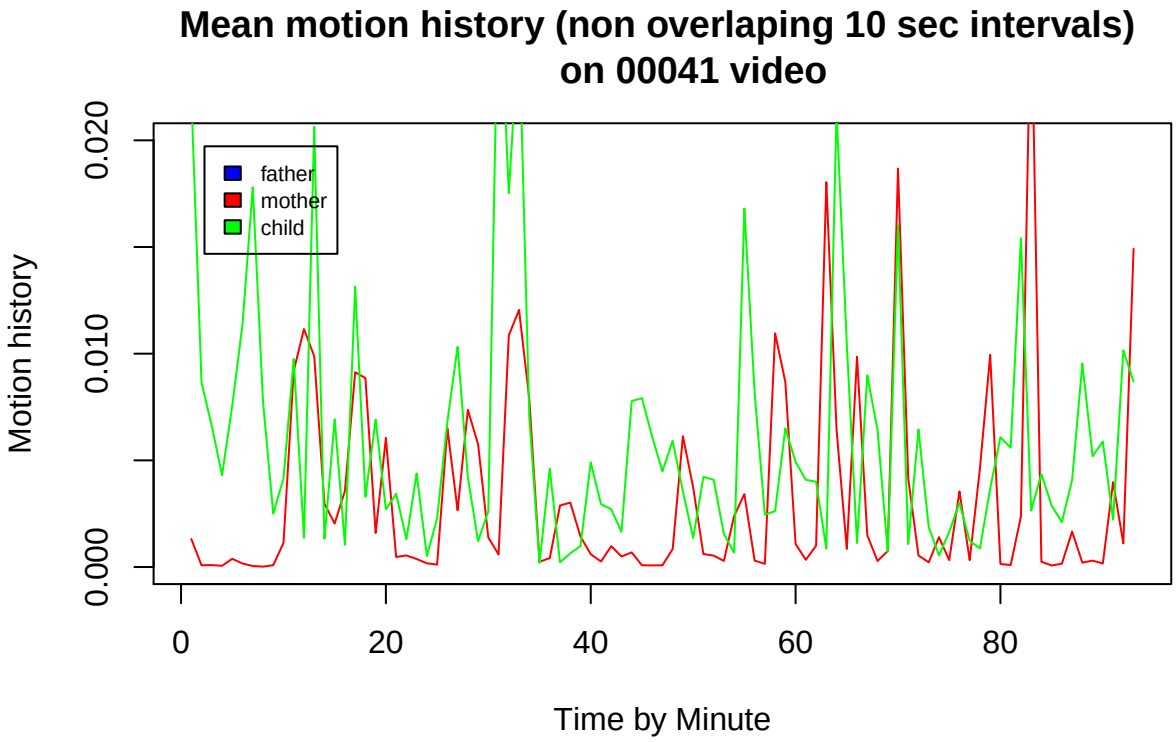


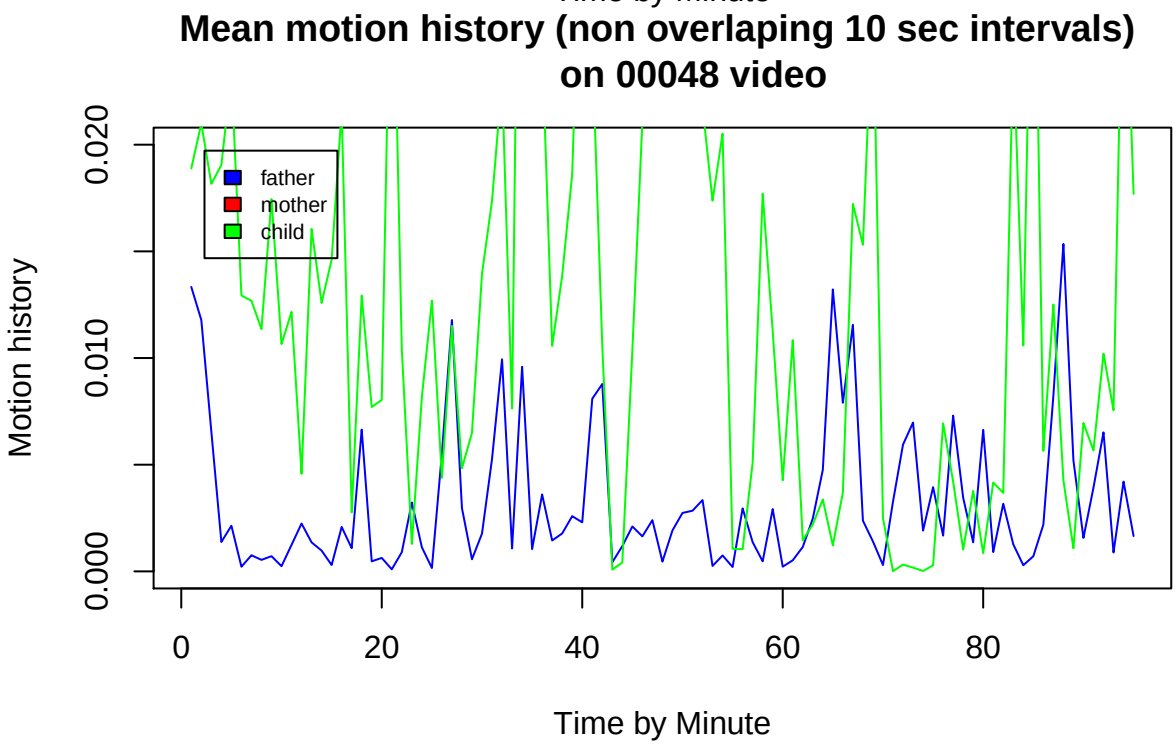
Mean motion history by 10 sec plots

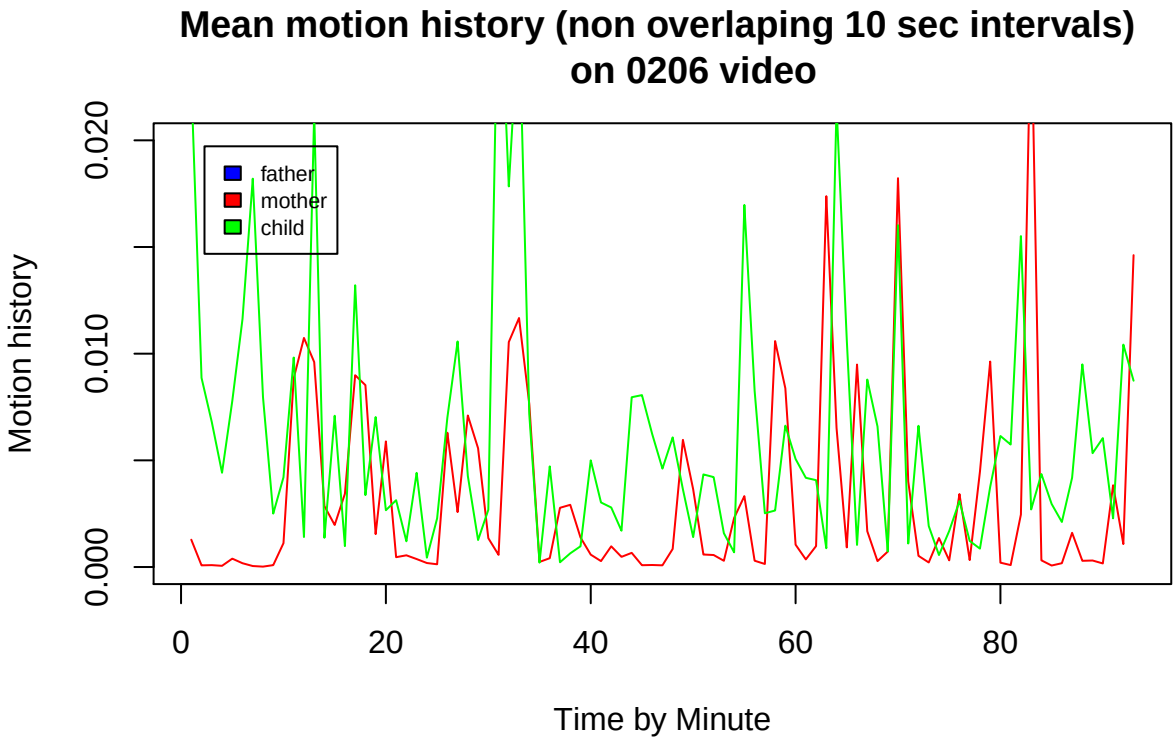
```
for (i in 1:NumberOfvideos){
  fatherMinute<- MeanMotionByTime("father", indexOfvideos=i, interval=250, data)

  motherMinute<- MeanMotionByTime("mother", indexOfvideos=i, interval=250, data)

  childMinute<- MeanMotionByTime("child", indexOfvideos=i, interval=250, data)

  par(mar=c(4,4,4,2))
      plot (1:length(fatherMinute), fatherMinute, type="l", col="blue",
      main=paste("Mean motion history (non overlaping 10 sec intervals)
      on ", families[i], " video" , sep=""),
      ylab="Motion history", xlab="Time by Minute", ylim=c(0, 20E-03))
#      xaxp=c(0, length(fatherMinute), length(fatherMinute)))
      lines(motherMinute, col="red")
      lines(childMinute, col="green")
      legend("topleft", inset=.05, ParticipantsList[1:3],
             fill=colOrderList[1:3], cex=0.7)}
```

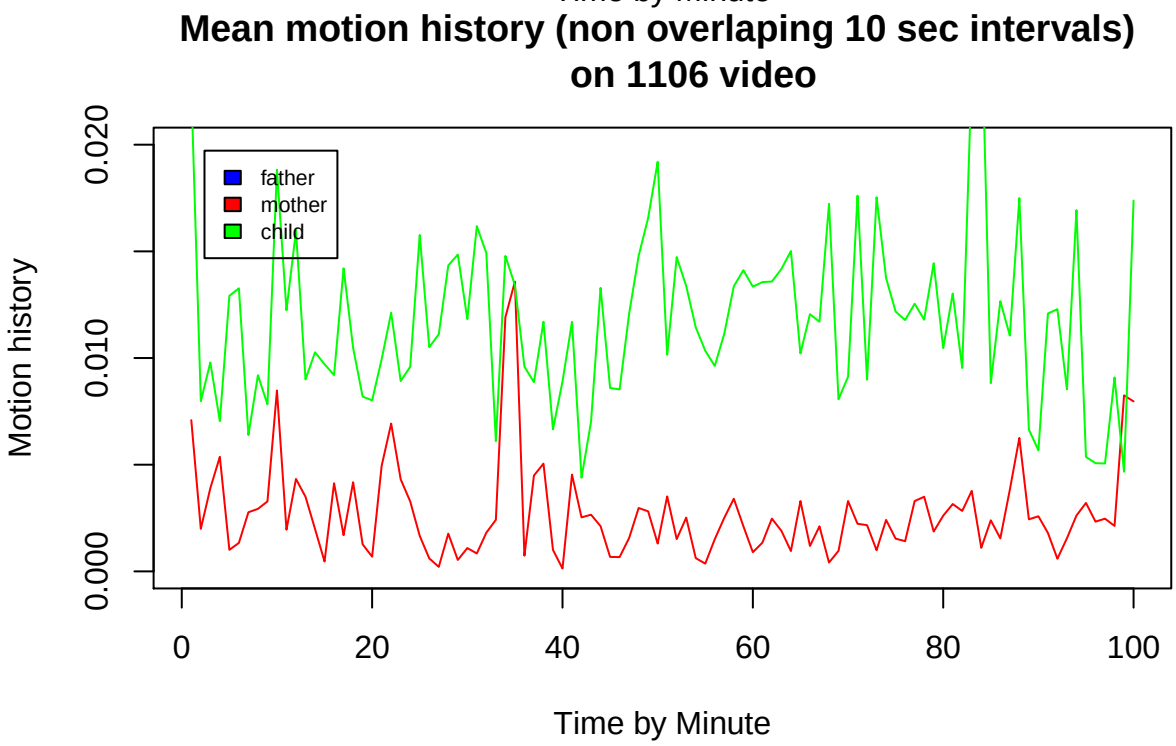

Mean motion history (non overlaping 10 sec intervals)
on 00034 video
Motion history
0.000
0.010
0.020
father
mother
child
0
20
40
60
80
Time by Minute

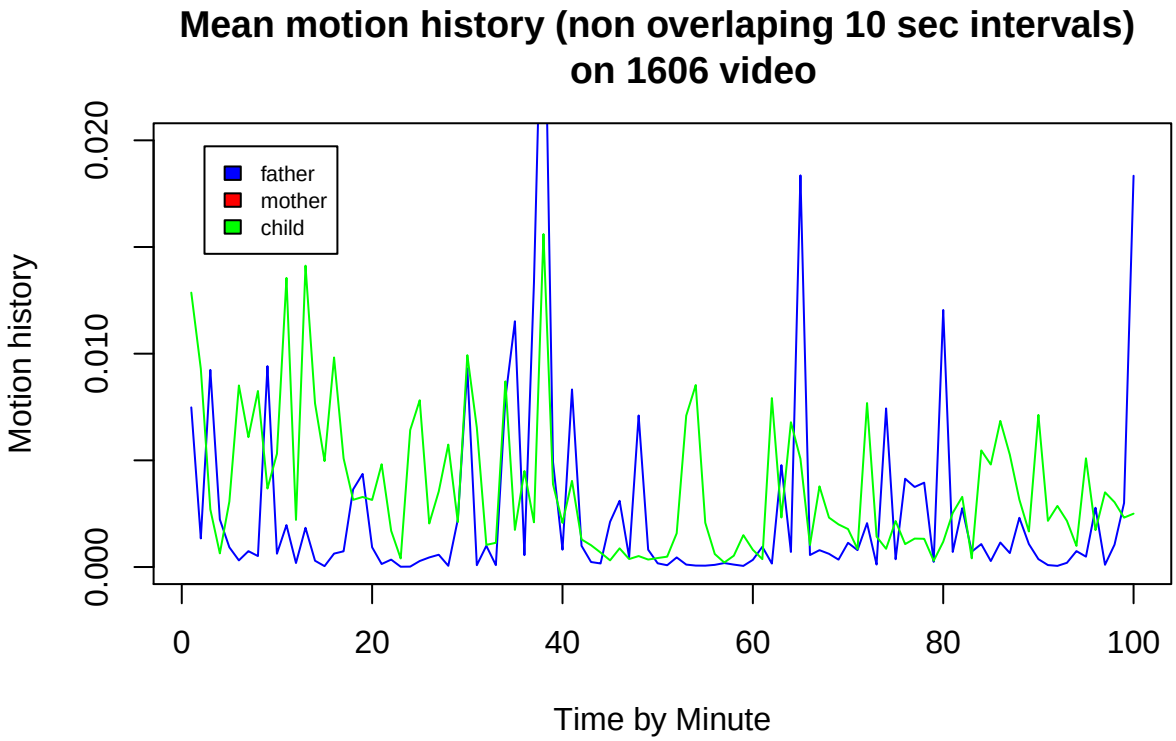

Mean motion history (non overlaping 10 sec intervals)
on 00037 video
Motion history
0.000
0.010
0.020
father
mother
child
0
20
40
60
80
100
Time by Minute

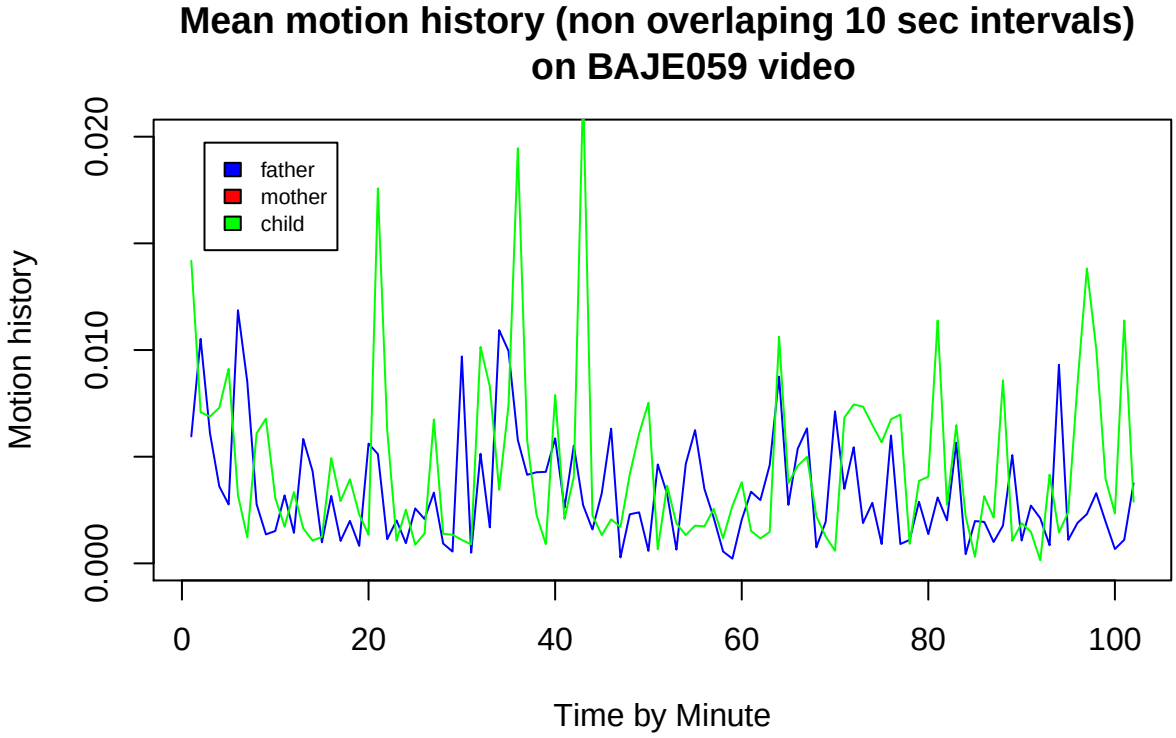

Mean motion history (non overlaping 10 sec intervals)
on 00041 video
Motion history
0.000
0.010
0.020
father
mother
child
0
20
40
60
80
Time by Minute

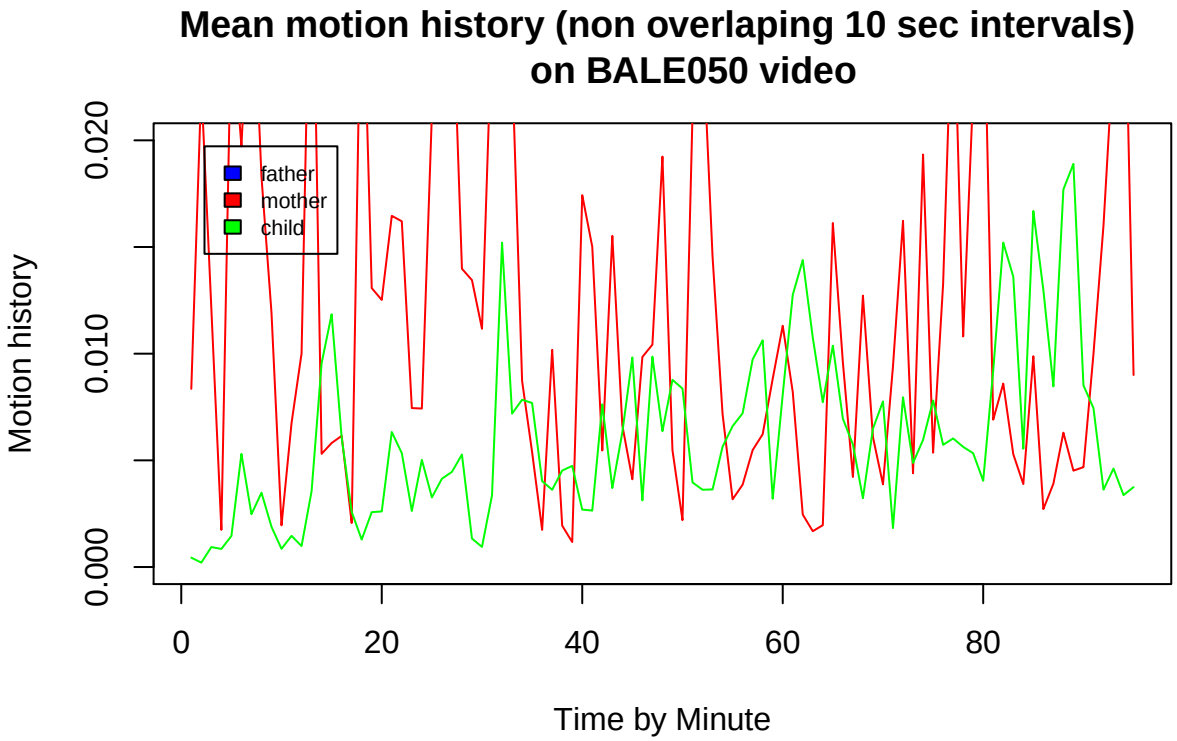

Mean motion history (non overlaping 10 sec intervals)
on 00048 video
Motion history
0.000
0.010
0.020
father
mother
child
0
20
40
60
80
Time by Minute

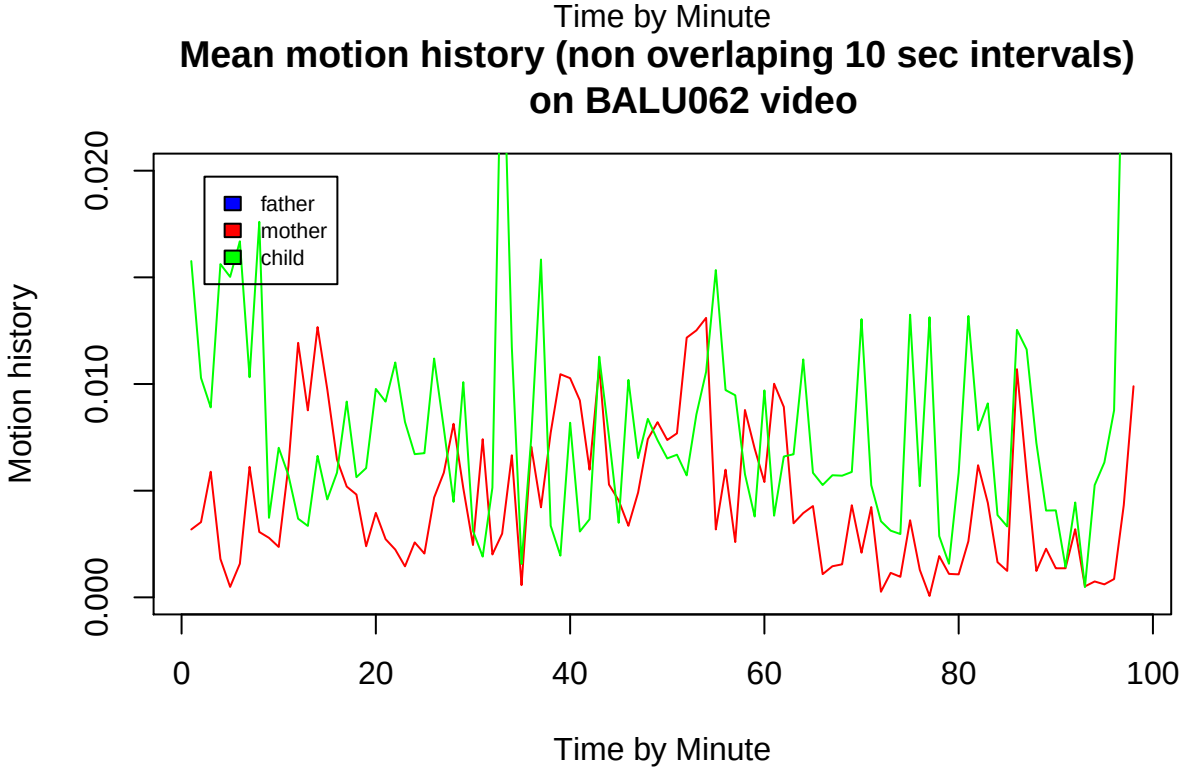

Mean motion history (non overlaping 10 sec intervals)
on 0206 video
Motion history
0.000
0.010
0.020
father
mother
child
0
20
40
60
80
Time by Minute

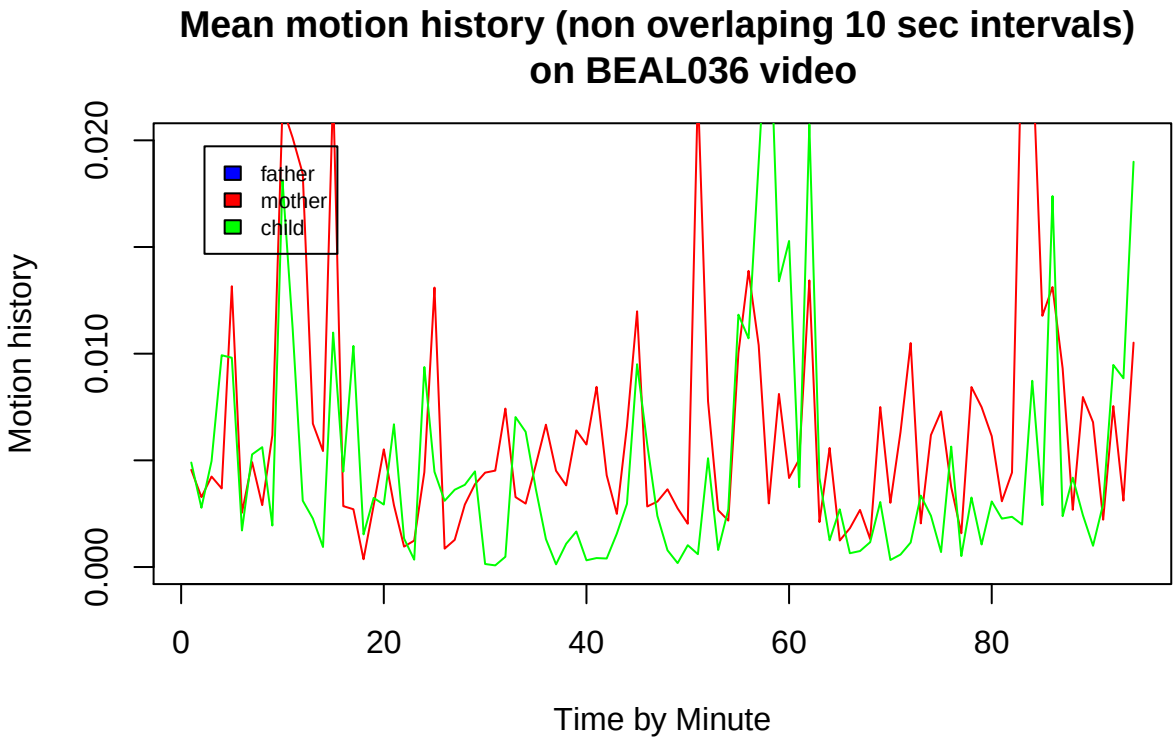

Mean motion history (non overlaping 10 sec intervals)
on 1106 video
Motion history
0.000
0.010
0.020
father
mother
child
0
20
40
60
80
100
Time by Minute

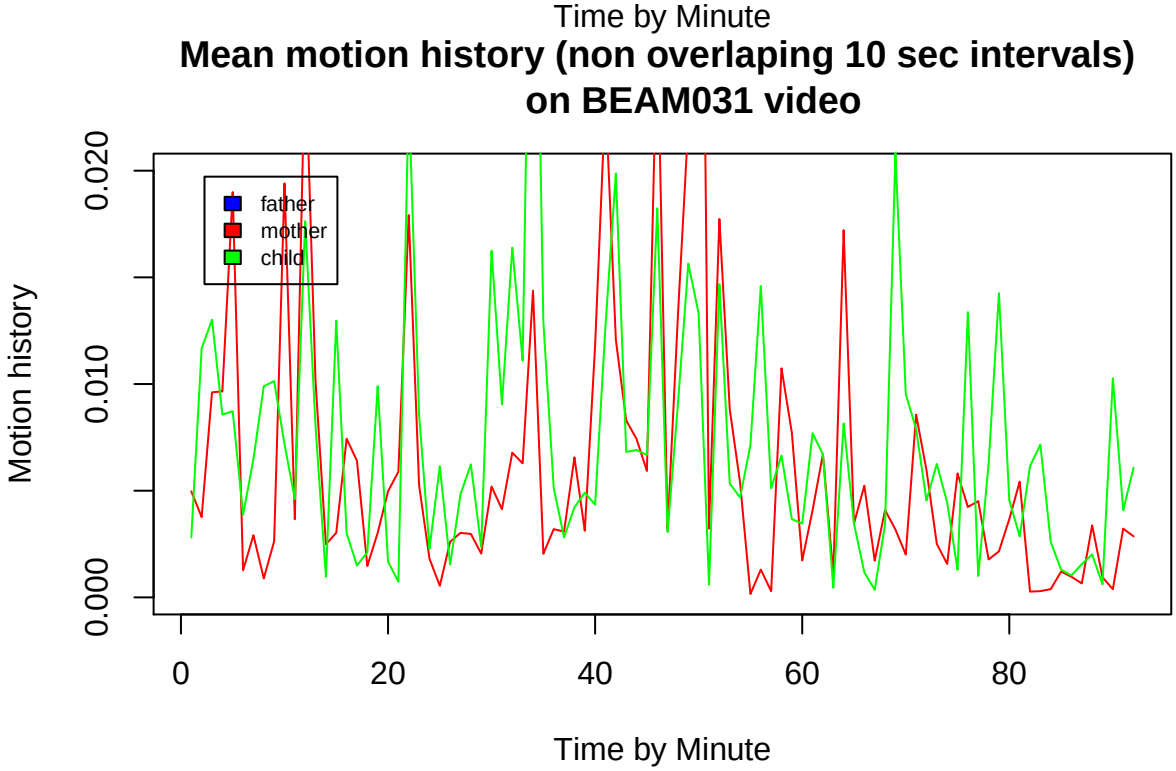

Mean motion history (non overlaping 10 sec intervals)
on 1606 video
Motion history
0.000
0.010
0.020
father
mother
child
0
20
40
60
80
100
Time by Minute

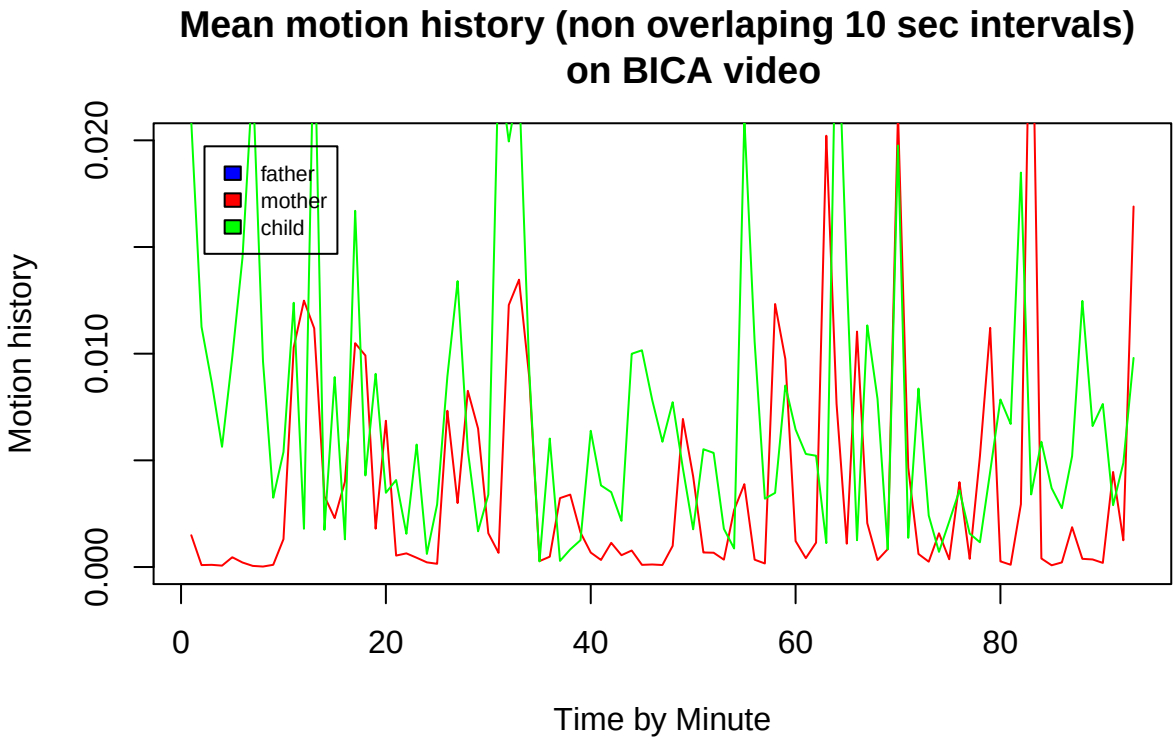

Mean motion history (non overlaping 10 sec intervals)
on BAJE059 video
Motion history
0.000
0.010
0.020
father
mother
child
0
20
40
60
80
100
Time by Minute

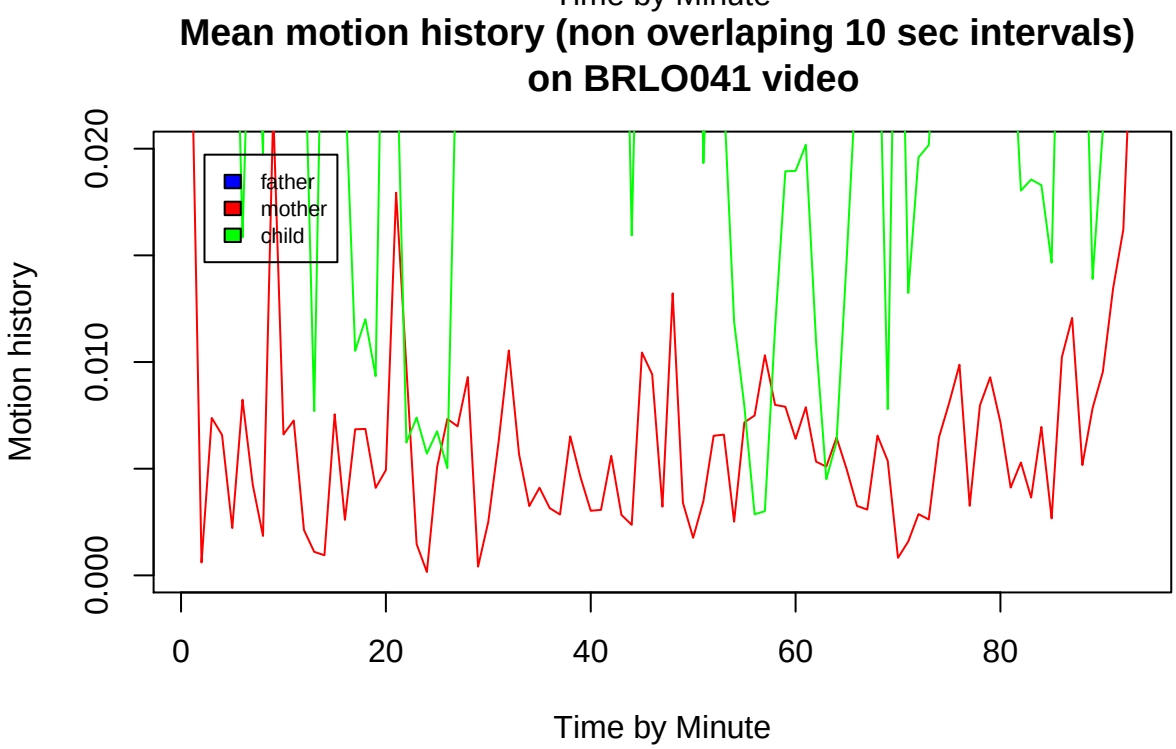

Mean motion history (non overlaping 10 sec intervals)
on BALE050 video
Motion history
0.000
0.010
0.020
father
mother
child
0
20
40
60
80
Time by Minute

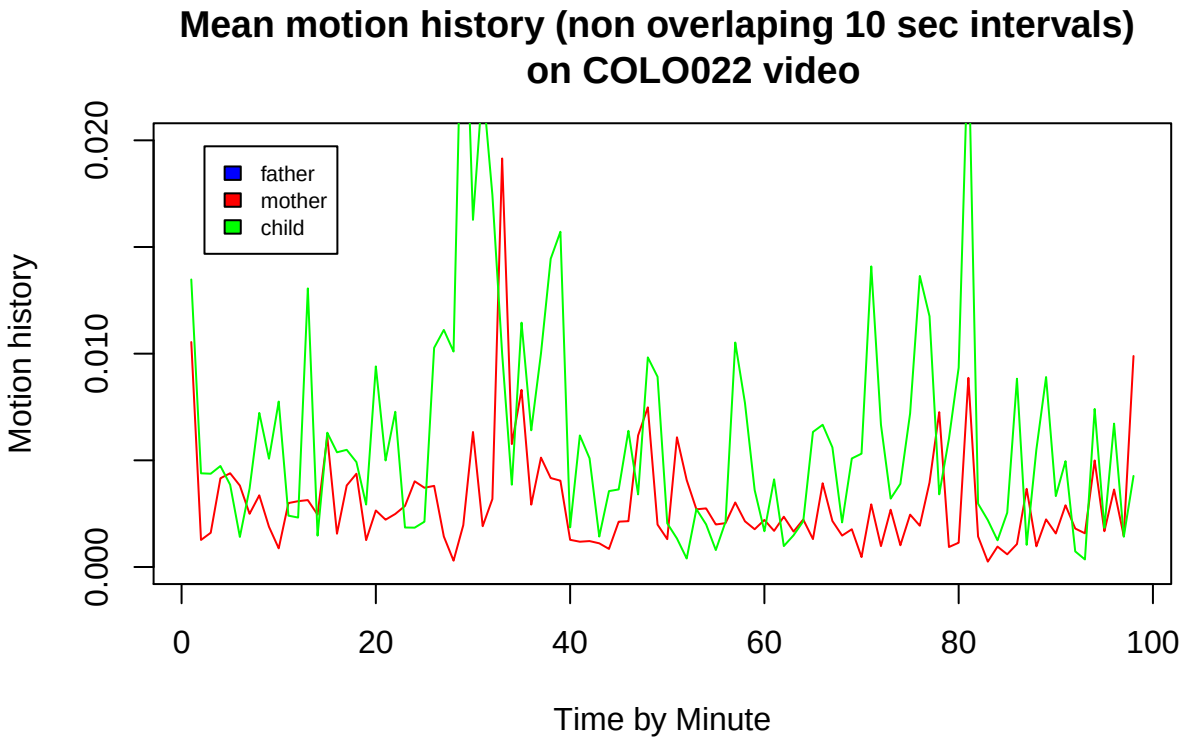

Mean motion history (non overlaping 10 sec intervals)
on BALU062 video
Motion history
0.000
0.010
0.020
father
mother
child
0
20
40
60
80
100
Time by Minute

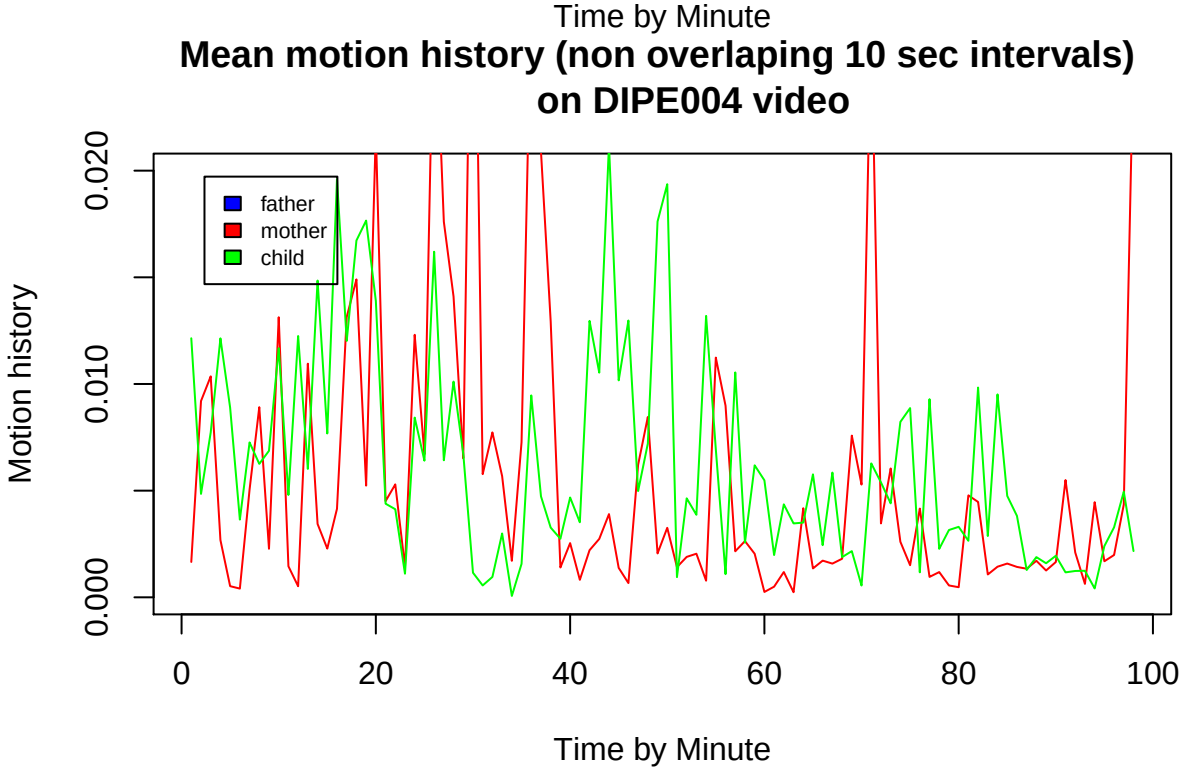

Mean motion history (non overlaping 10 sec intervals)
on BEAL036 video
Motion history
0.000
0.010
0.020
father
mother
child
0
20
40
60
80
Time by Minute

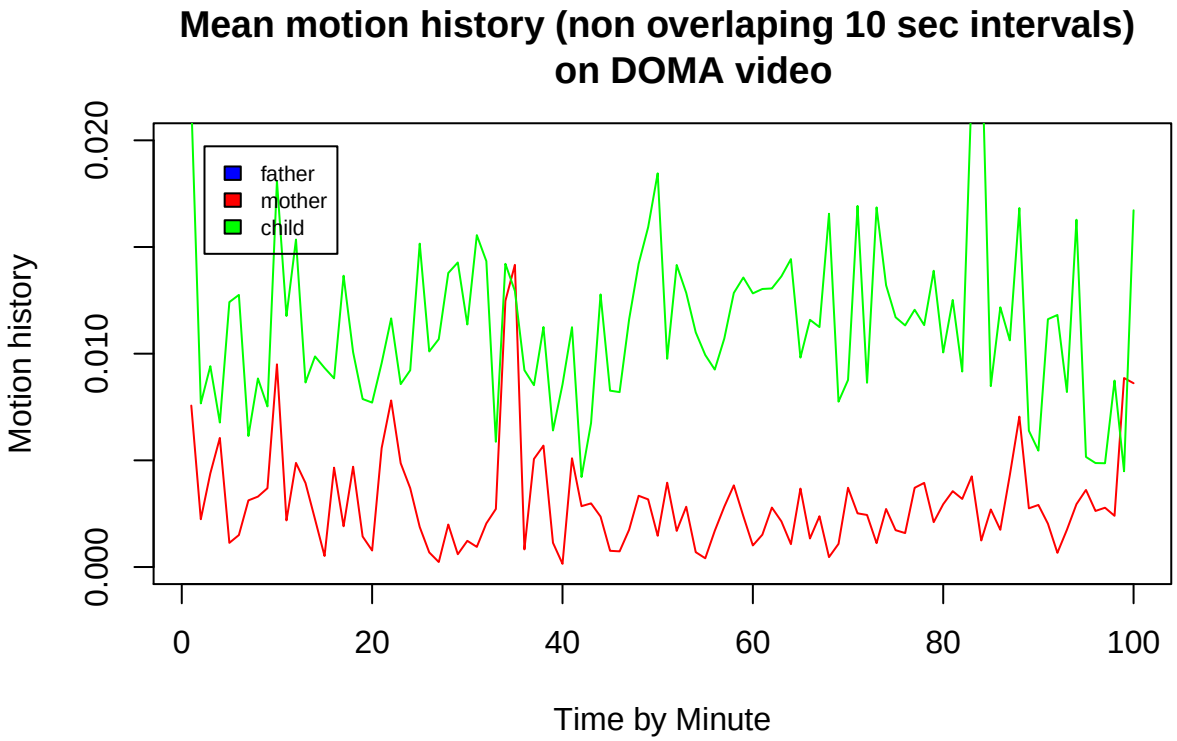

Mean motion history (non overlaping 10 sec intervals)
on BEAM031 video
Motion history
0.000
0.010
0.020
father
mother
child
0
20
40
60
80
Time by Minute

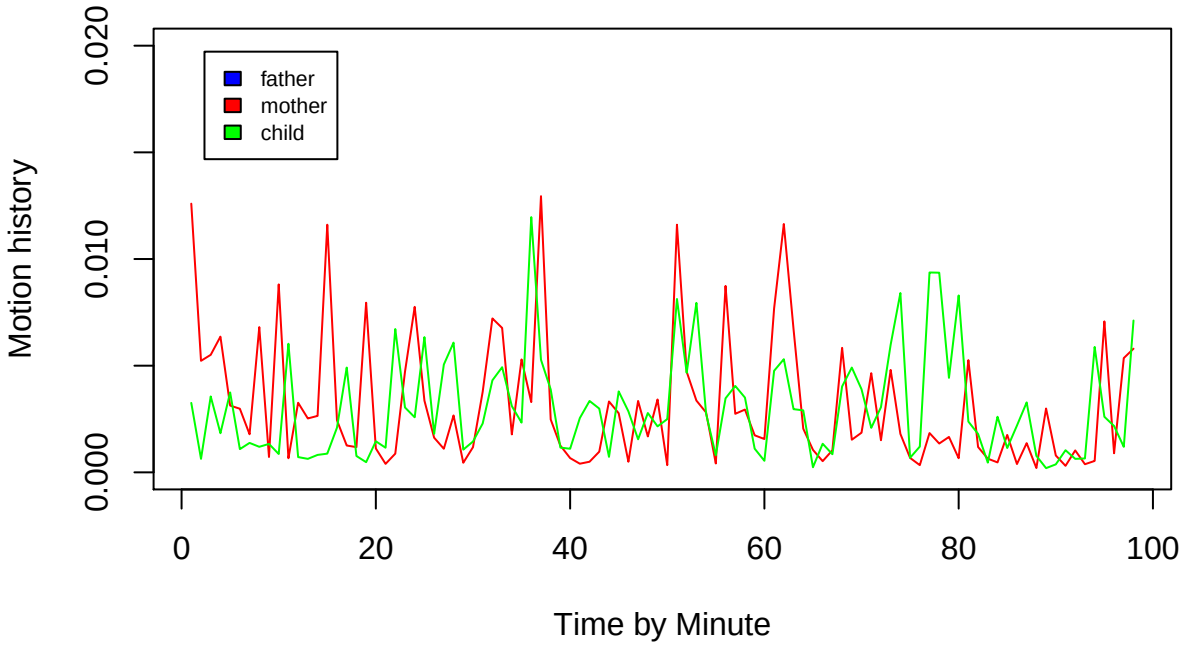

Mean motion history (non overlaping 10 sec intervals)
on BICA video
Motion history
0.000
0.010
0.020
father
mother
child
0
20
40
60
80
Time by Minute

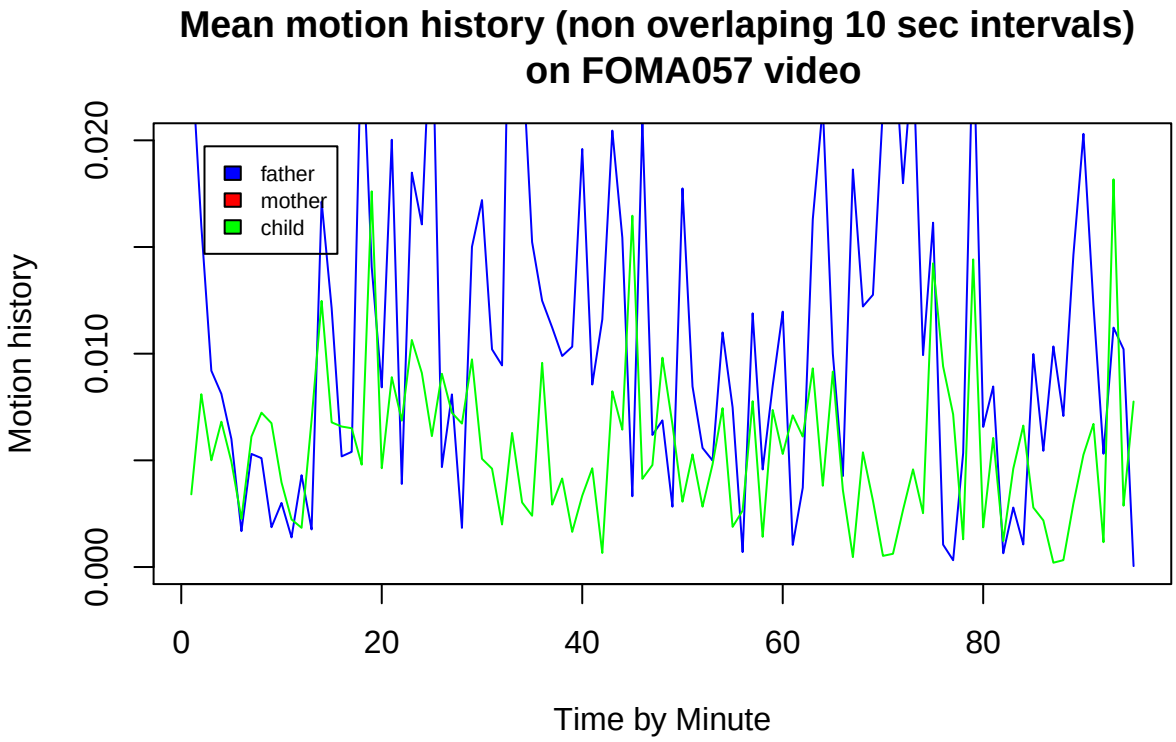

Mean motion history (non overlaping 10 sec intervals)
on BRLO041 video
Motion history
0.000
0.010
0.020
father
mother
child
0
20
40
60
80
Time by Minute

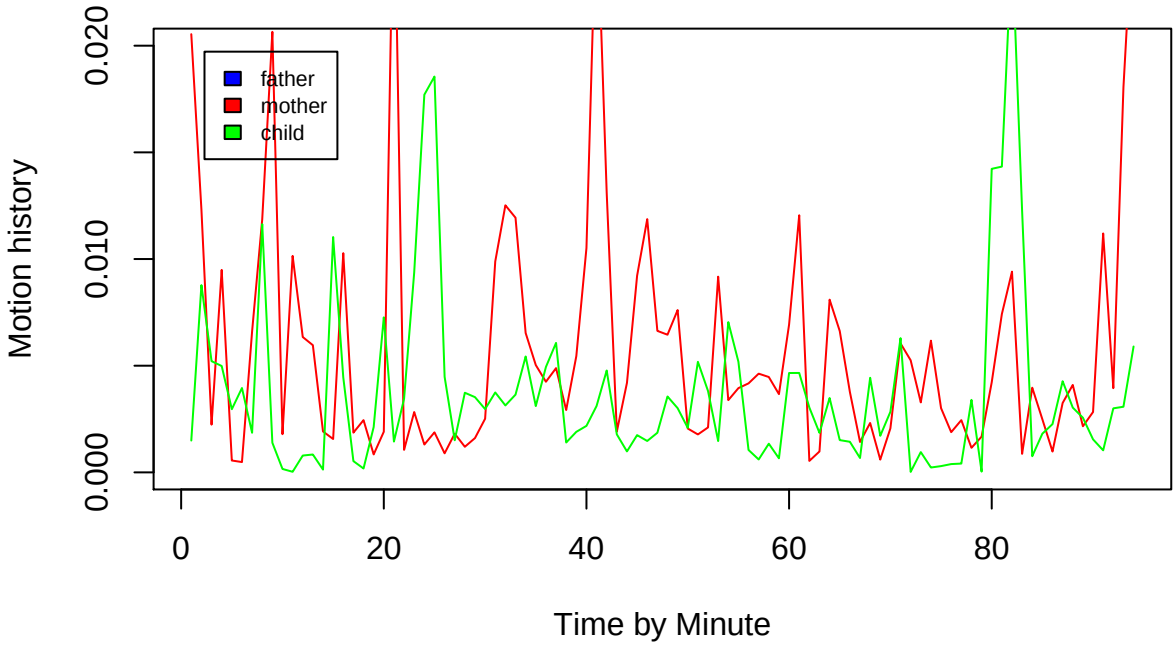

Mean motion history (non overlaping 10 sec intervals)
on COLO022 video
Motion history
0.000
0.010
0.020
father
mother
child
0
20
40
60
80
100
Time by Minute

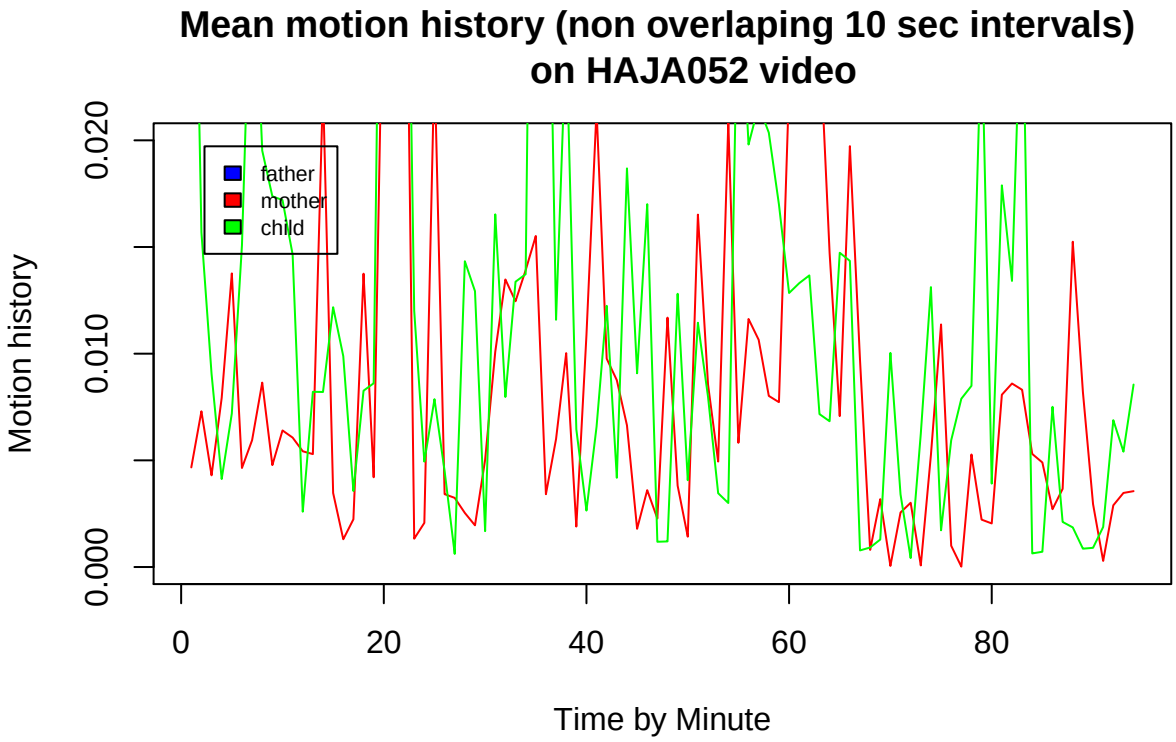

Mean motion history (non overlaping 10 sec intervals)
on DIPE004 video
Motion history
0.000
0.010
0.020
father
mother
child
0
20
40
60
80
100
Time by Minute

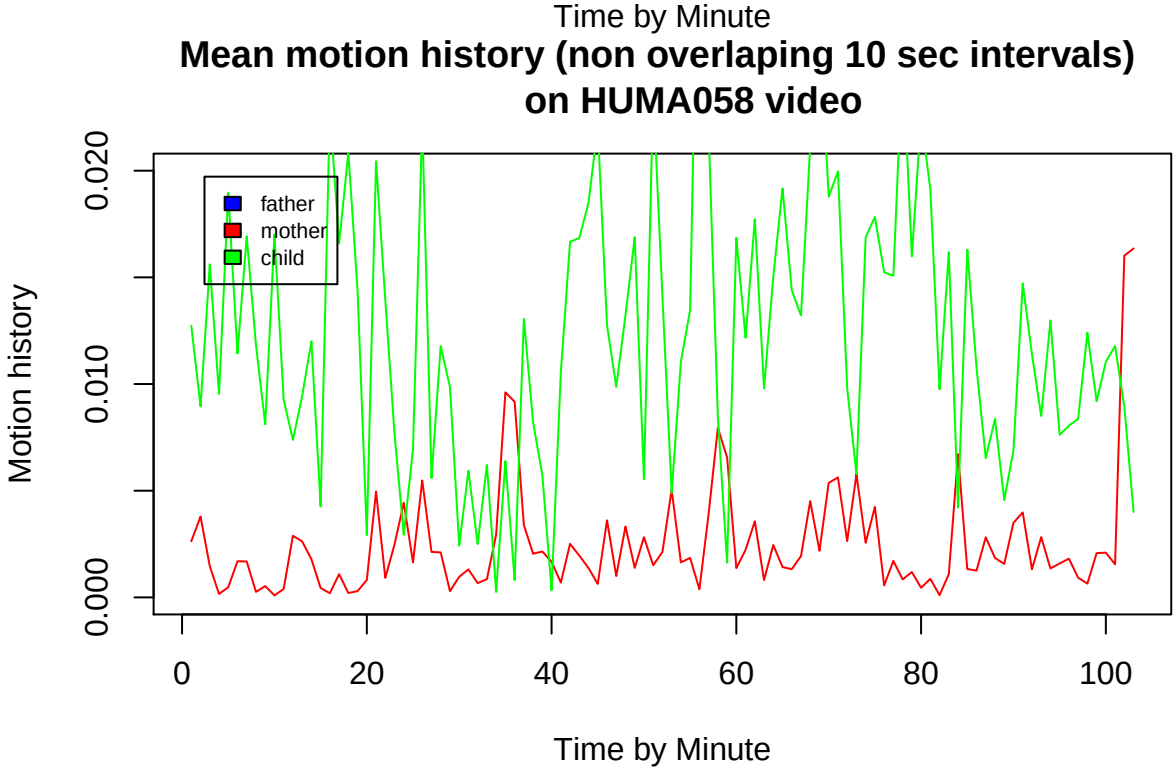

Mean motion history (non overlaping 10 sec intervals)
on DOMA video
Motion history
0.000
0.010
0.020
father
mother
child
0
20
40
60
80
100
Time by Minute

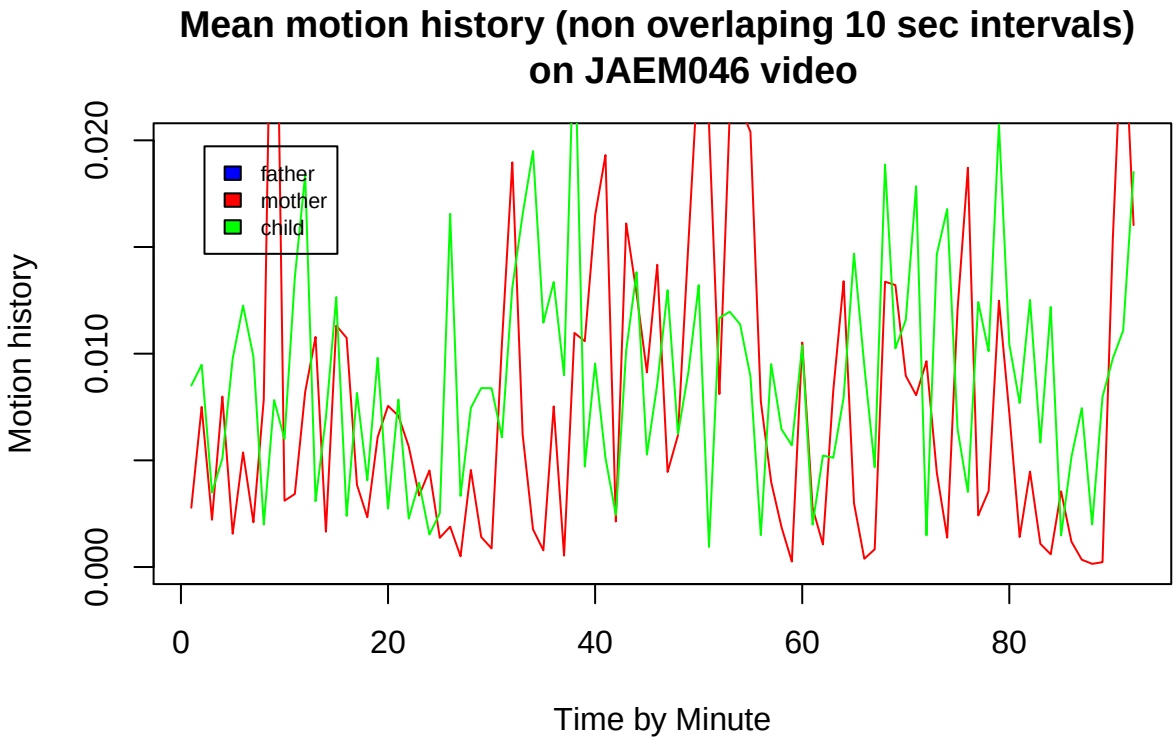

Mean motion history (non overlaping 10 sec intervals)
on DRNE video
Motion history
0.000
0.010
0.020
father
mother
child
0
20
40
60
80
100
Time by Minute

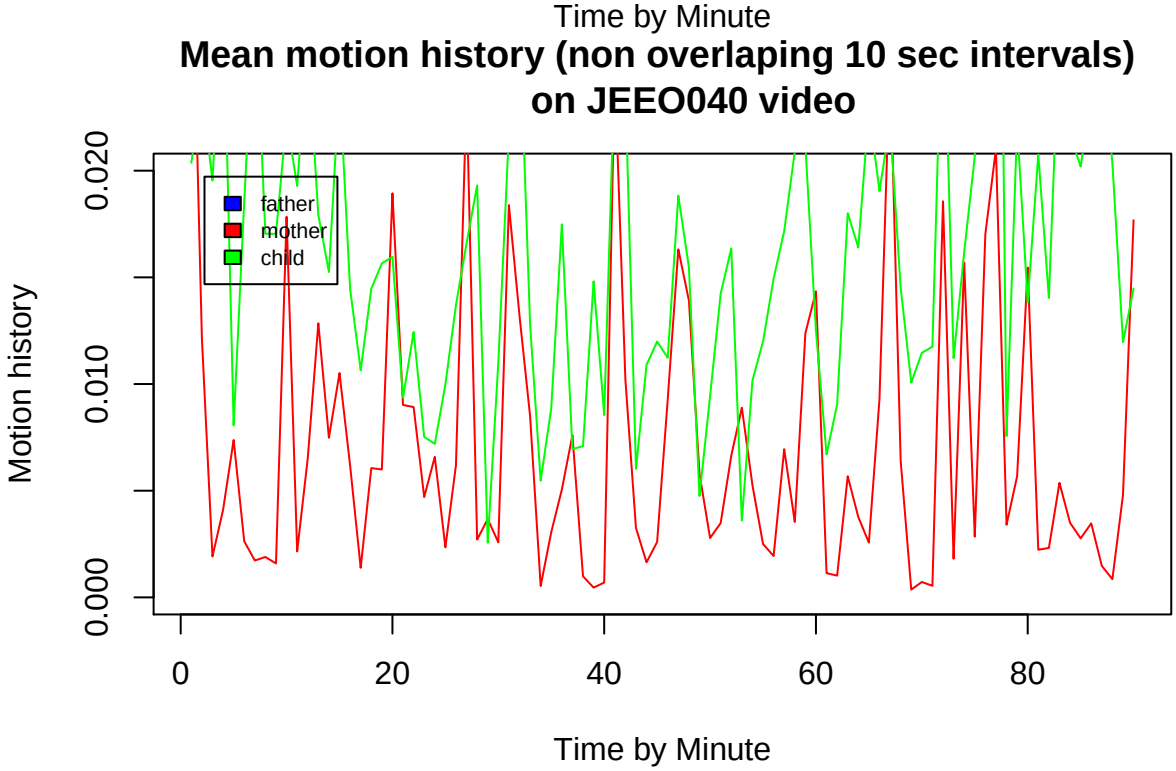

Mean motion history (non overlaping 10 sec intervals)
on FOMA057 video
Motion history
0.000
0.010
0.020
father
mother
child
0
20
40
60
80
Time by Minute

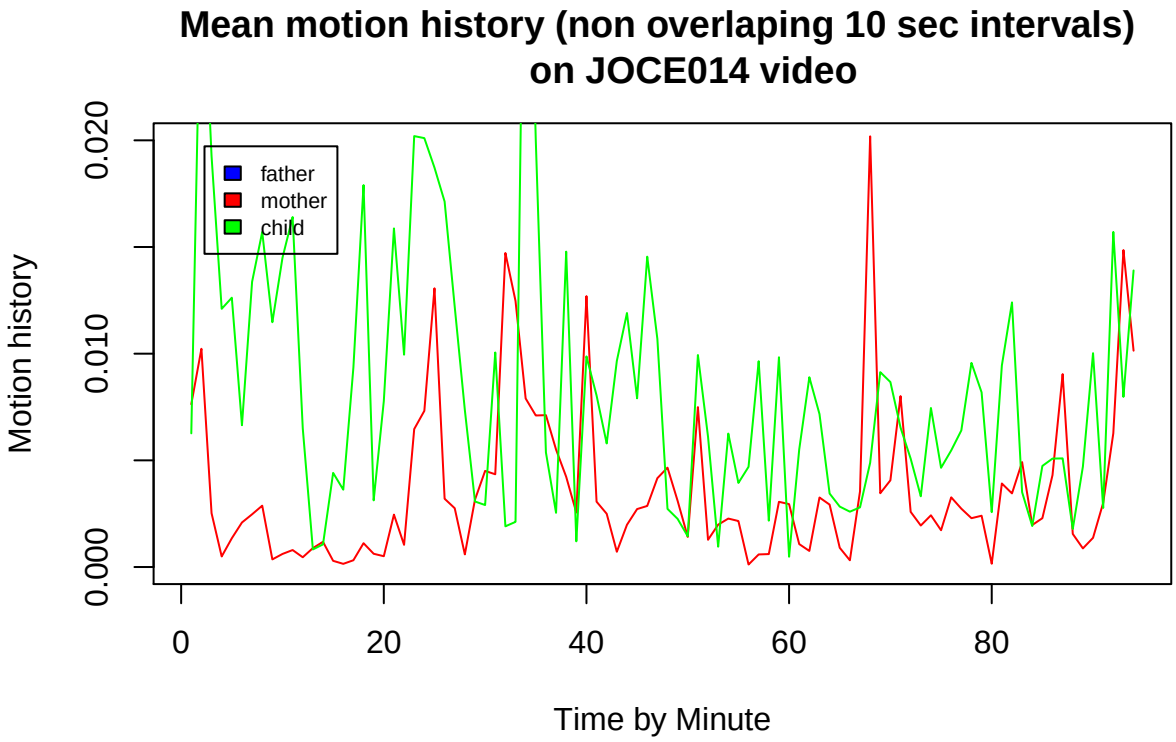

Mean motion history (non overlaping 10 sec intervals)
on GROP039 video
Motion history
0.000
0.010
0.020
father
mother
child
0
20
40
60
80
Time by Minute

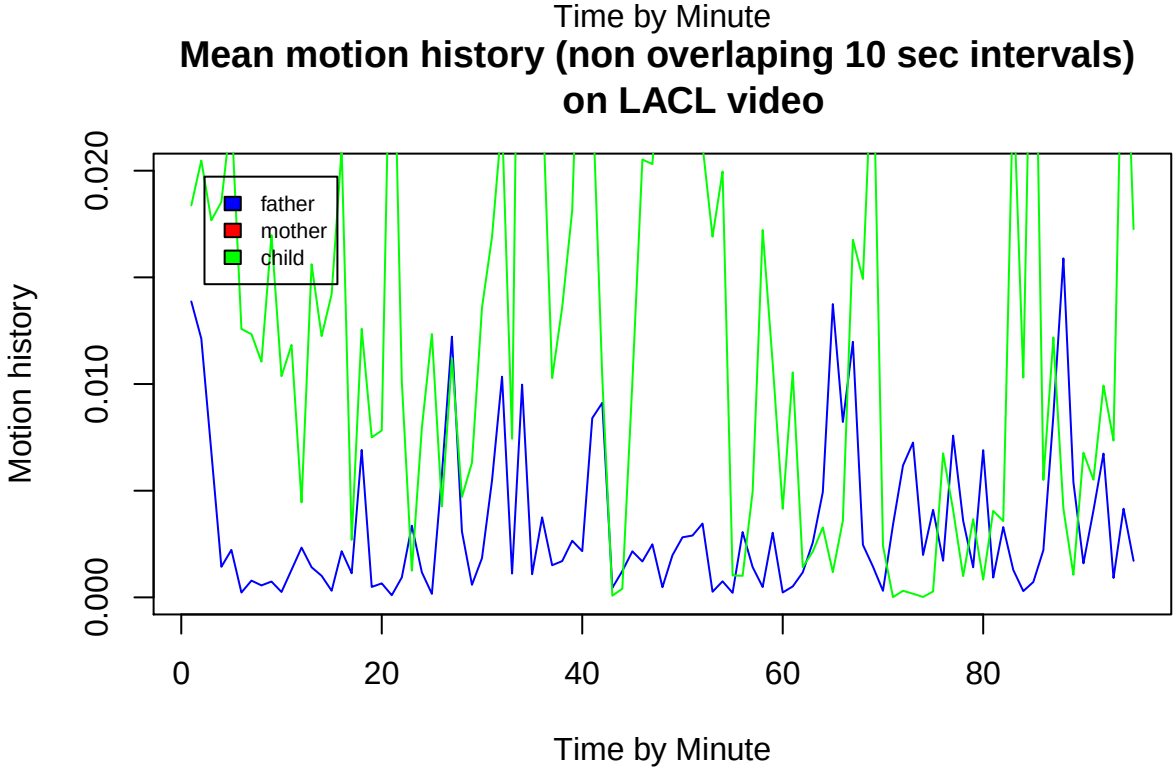

Mean motion history (non overlaping 10 sec intervals)
on HAJA052 video
Motion history
0.000
0.010
0.020
father
mother
child
0
20
40
60
80
Time by Minute

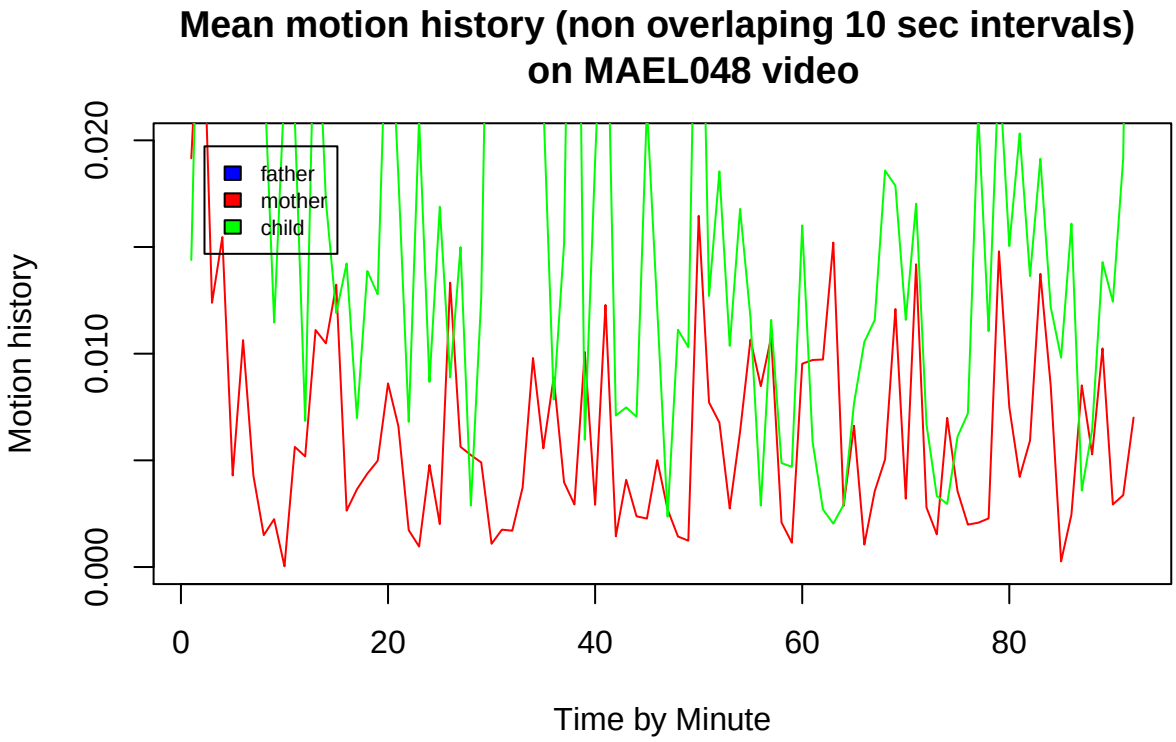

Mean motion history (non overlaping 10 sec intervals)
on HUMA058 video
Motion history
0.000
0.010
0.020
father
mother
child
0
20
40
60
80
100
Time by Minute

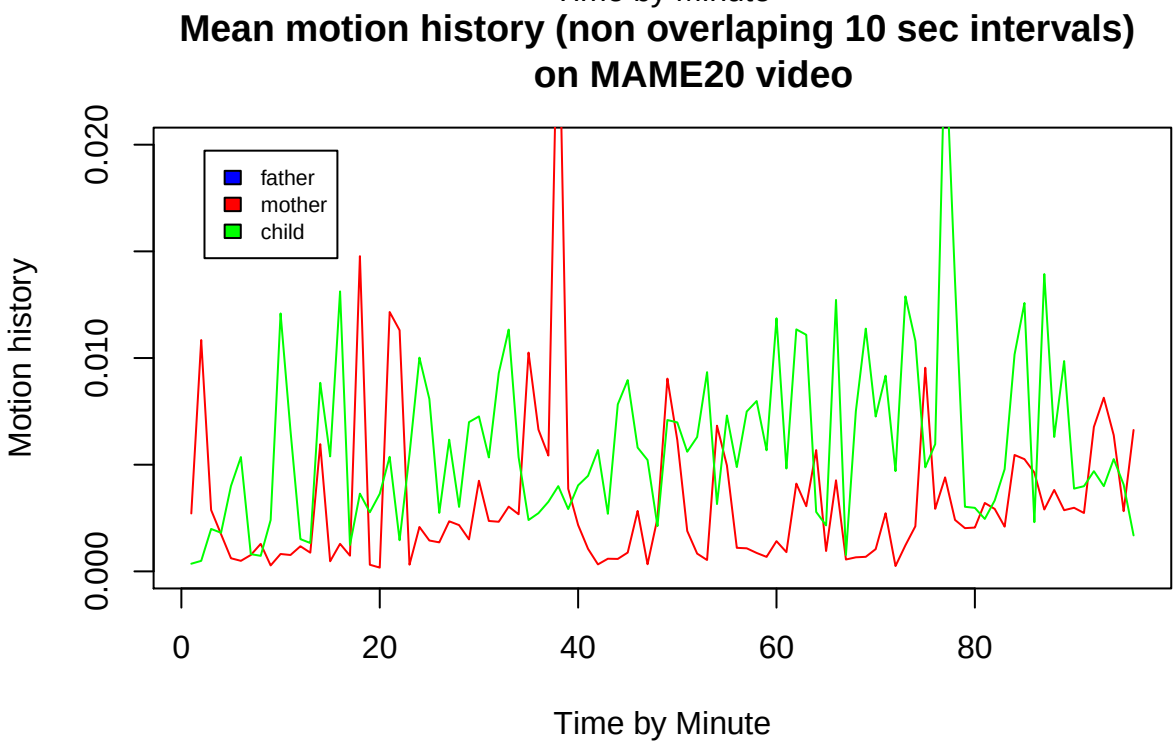

Mean motion history (non overlaping 10 sec intervals)
on JAEM046 video
Motion history
0.000
0.010
0.020
father
mother
child
0
20
40
60
80
Time by Minute

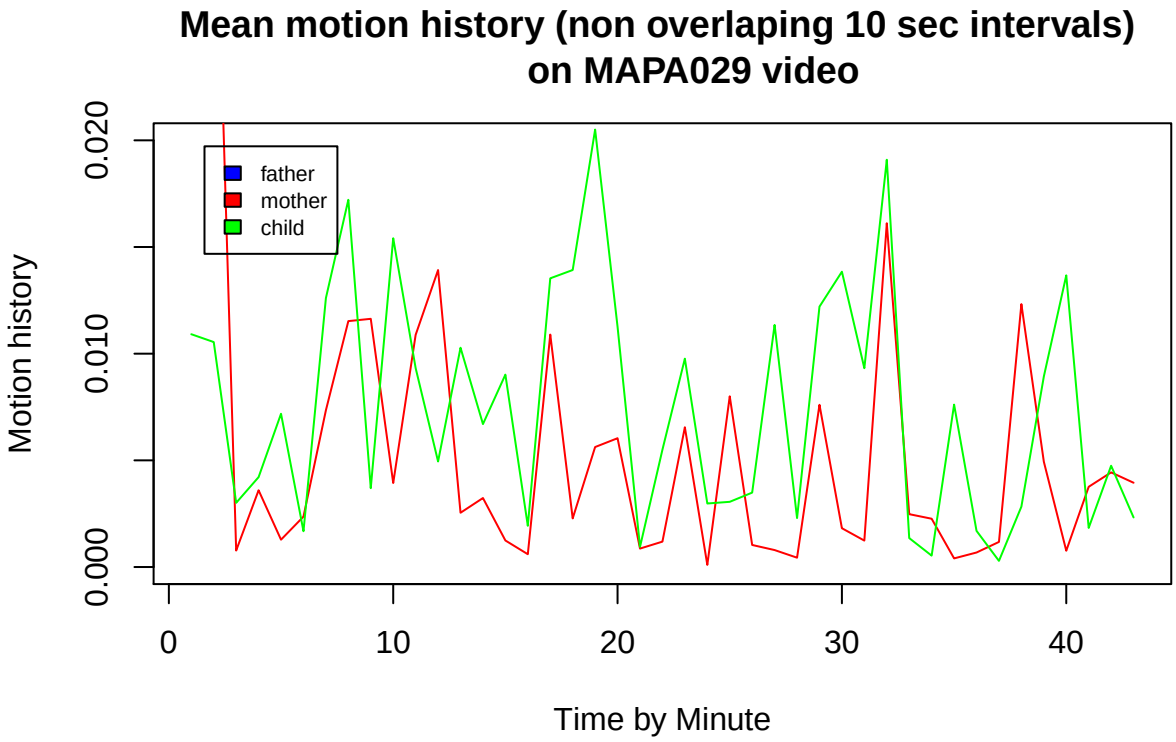

Mean motion history (non overlaping 10 sec intervals)
on JEEO040 video
Motion history
0.000
0.010
0.020
father
mother
child
0
20
40
60
80
Time by Minute

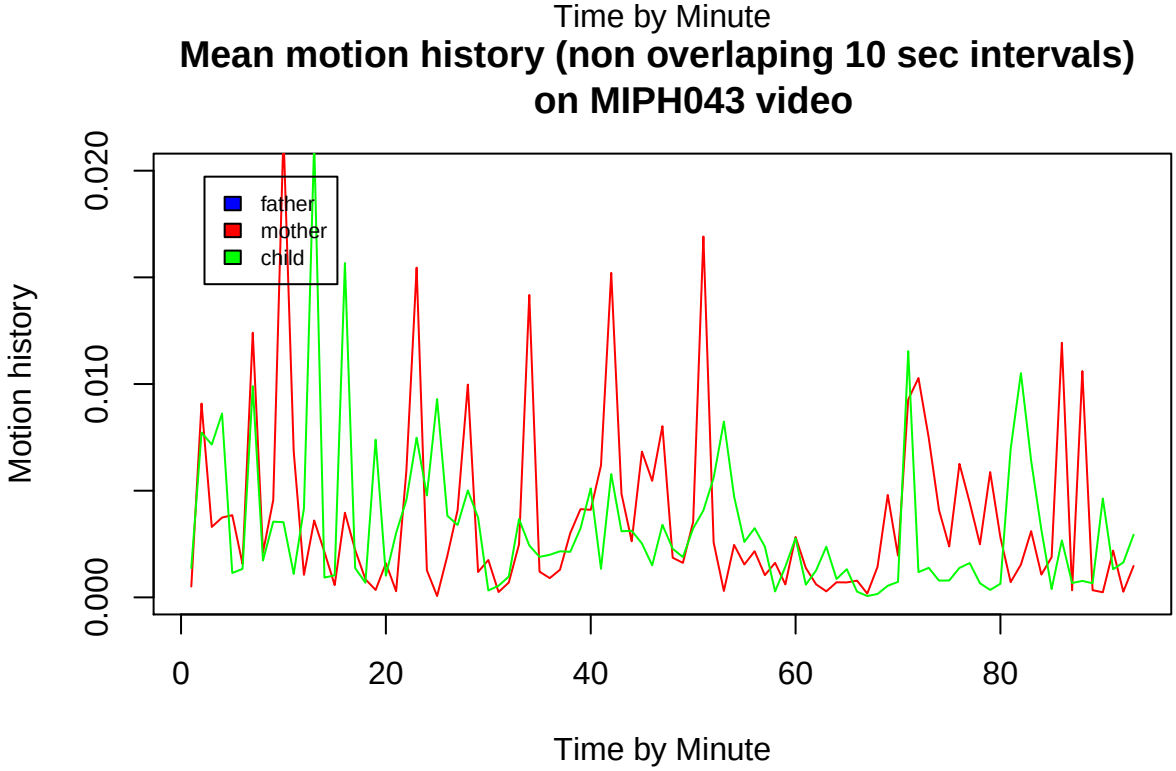

Mean motion history (non overlaping 10 sec intervals)
on JOCE014 video
Motion history
0.000
0.010
0.020
father
mother
child
0
20
40
60
80
Time by Minute

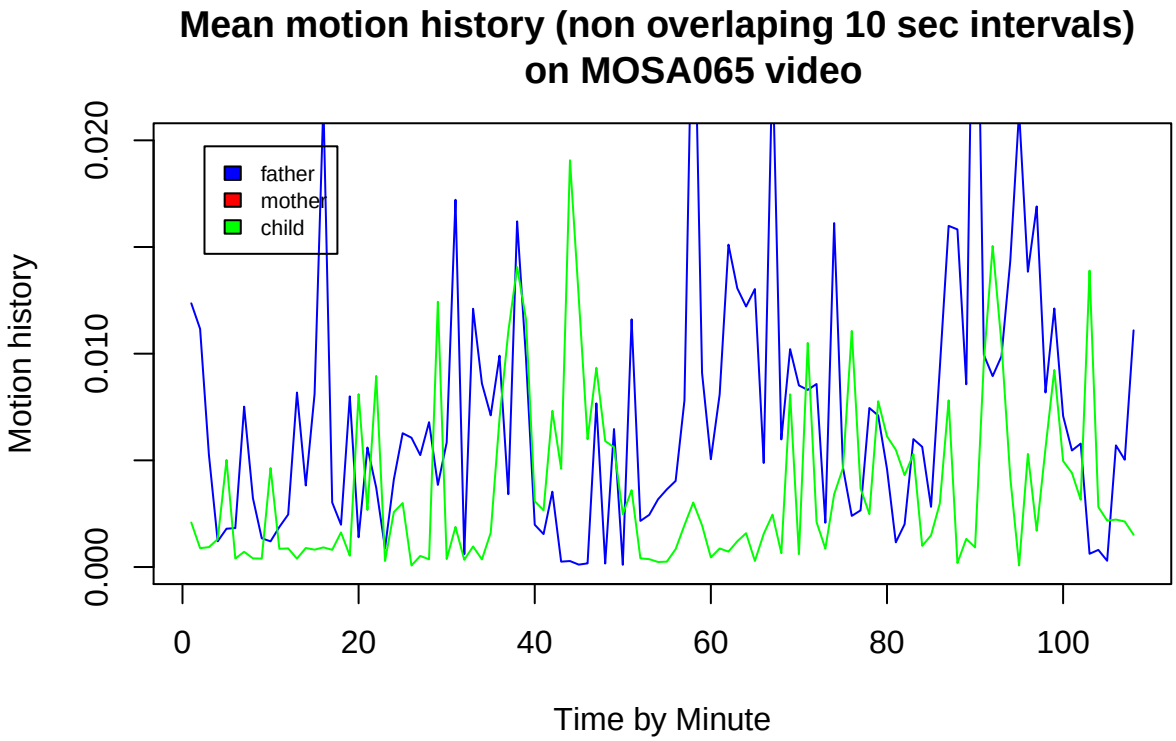

Mean motion history (non overlaping 10 sec intervals)
on LACL video
Motion history
0.000
0.010
0.020
father
mother
child
0
20
40
60
80
Time by Minute

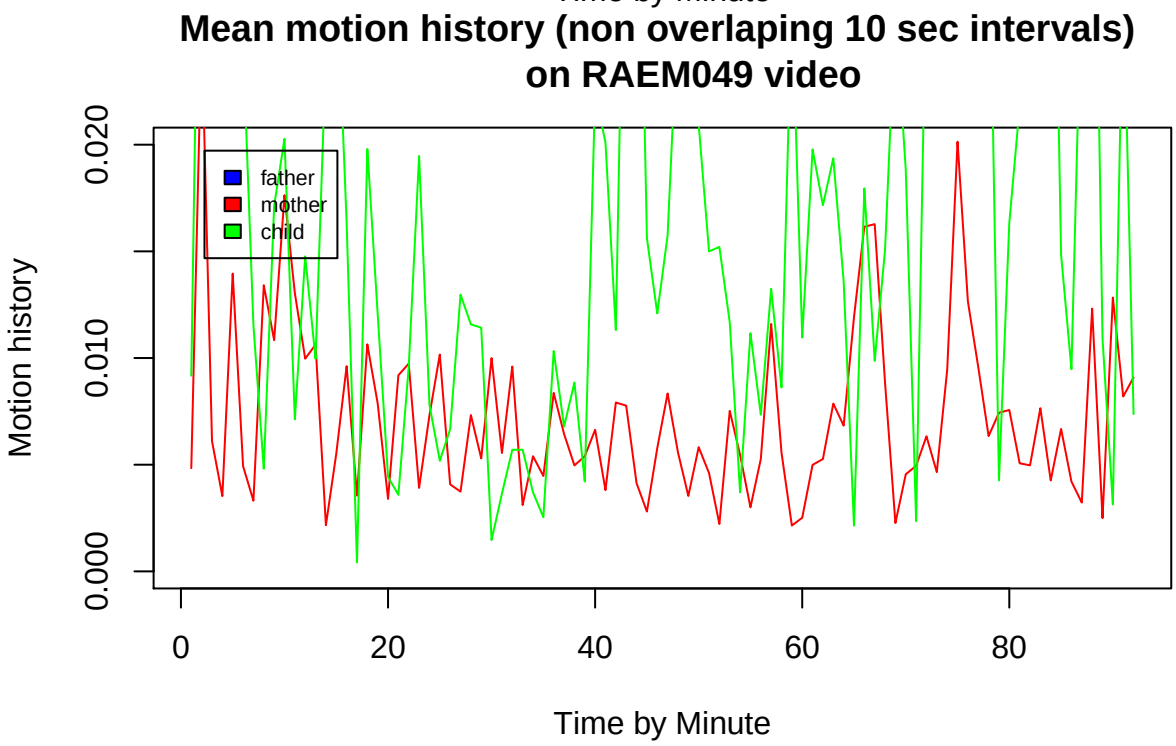

Mean motion history (non overlaping 10 sec intervals)
on MAEL048 video
Motion history
0.000
0.010
0.020
father
mother
child
0
20
40
60
80
Time by Minute

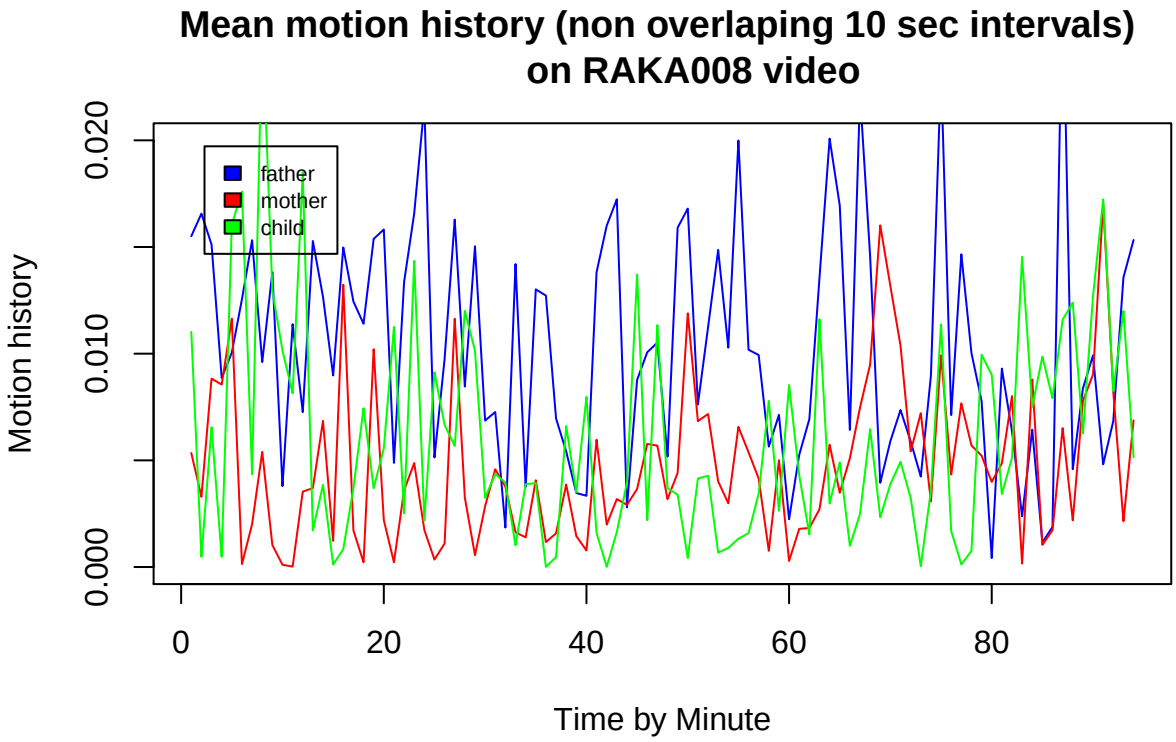

Mean motion history (non overlaping 10 sec intervals)
on MAME20 video
Motion history
0.000
0.010
0.020
father
mother
child
0
20
40
60
80
Time by Minute

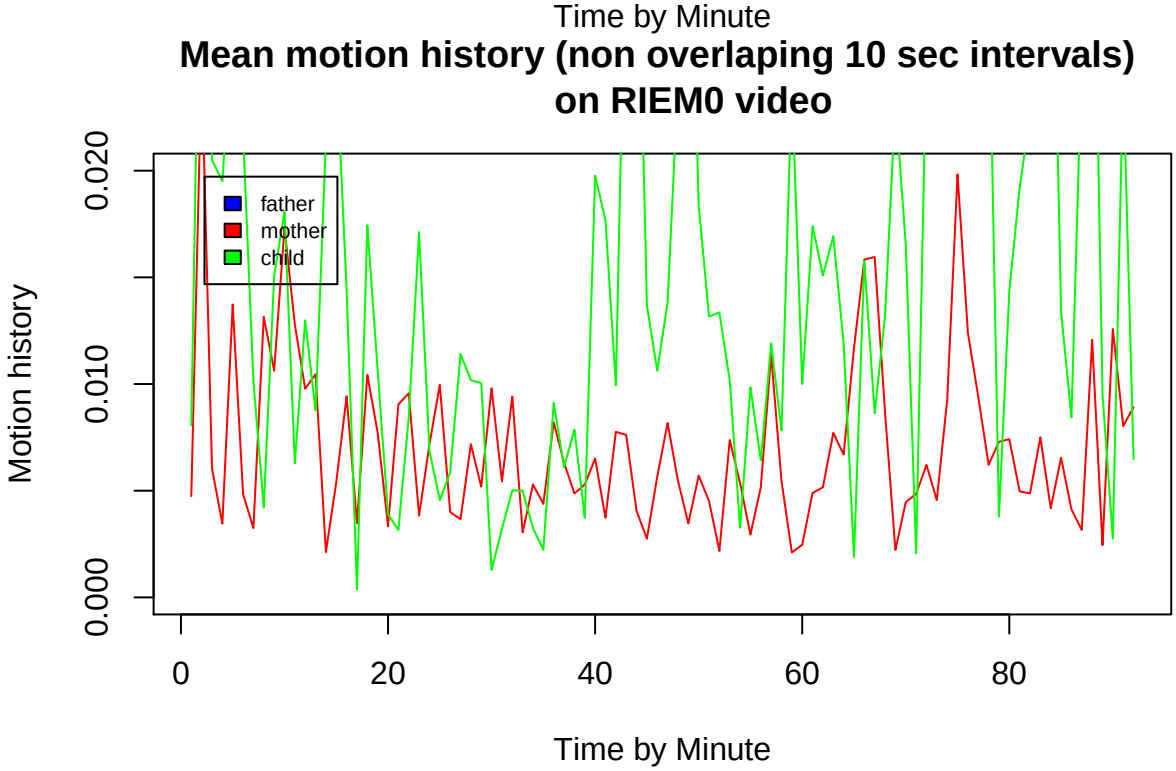

Mean motion history (non overlaping 10 sec intervals)
on MAPA029 video
Motion history
0.000
0.010
0.020
father
mother
child
0
10
20
30
40
Time by Minute

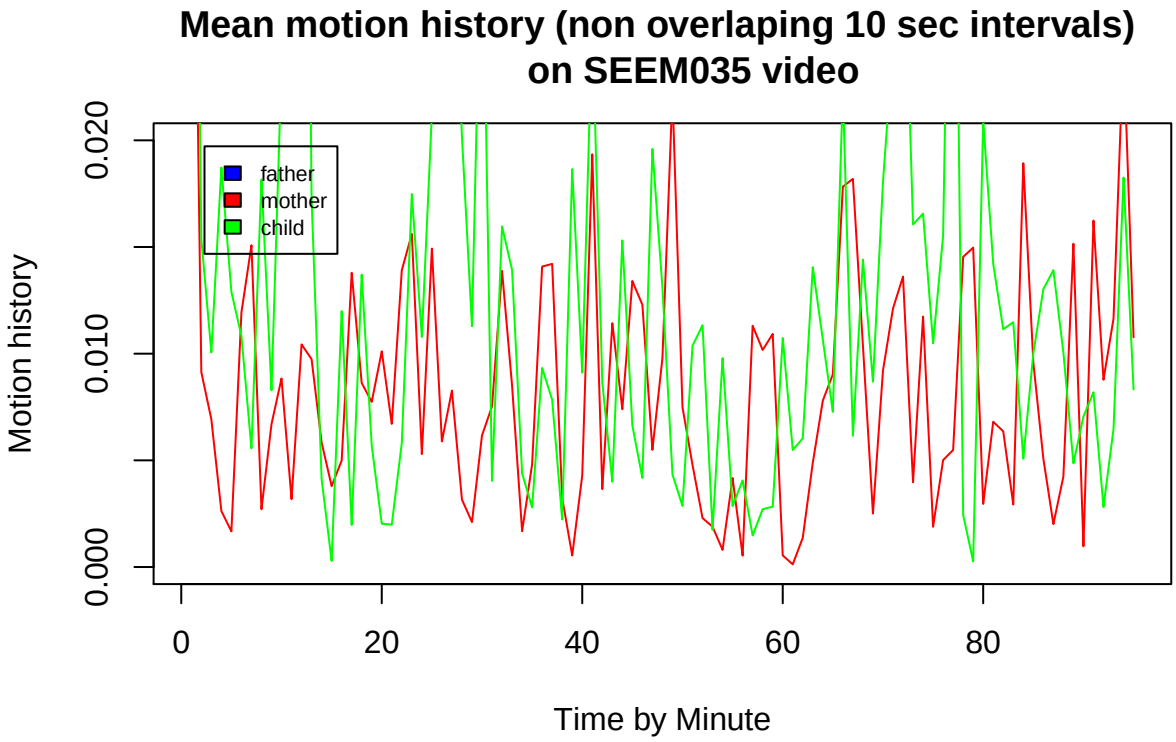

Mean motion history (non overlaping 10 sec intervals)
on MIPH043 video
Motion history
0.000
0.010
0.020
father
mother
child
0
20
40
60
80
Time by Minute

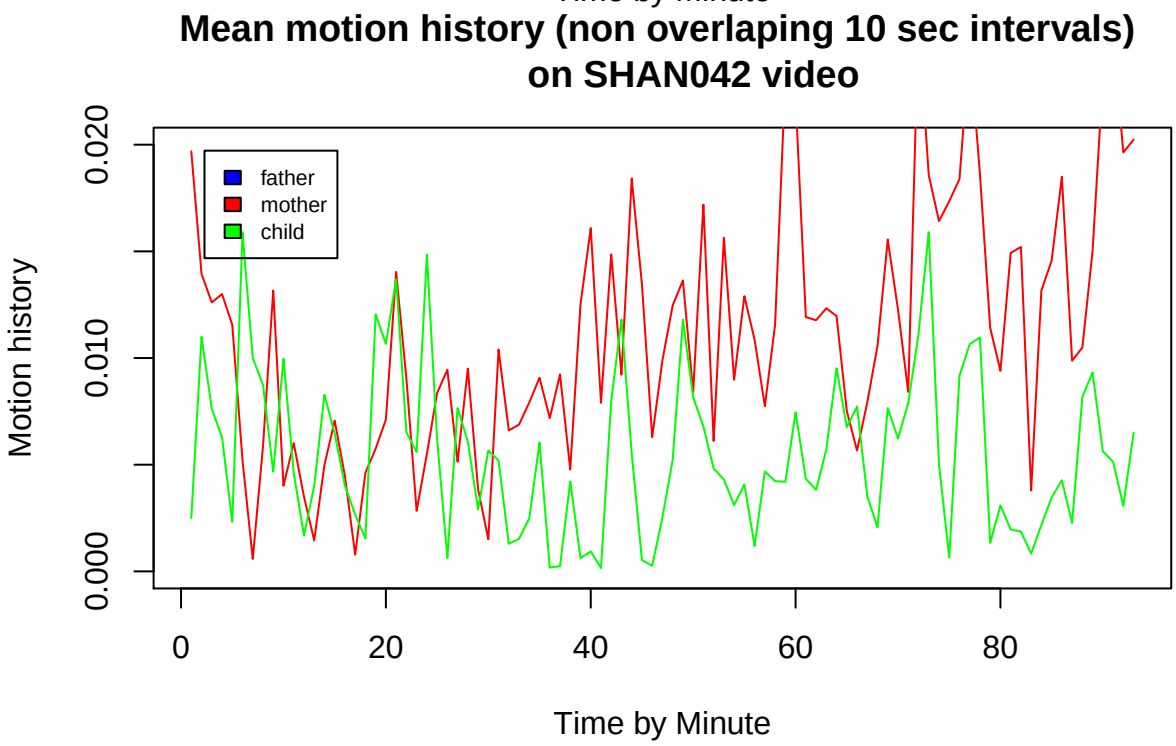

Mean motion history (non overlaping 10 sec intervals)
on MOSA065 video
Motion history
0.000
0.010
0.020
father
mother
child
0
20
40
60
80
100
Time by Minute

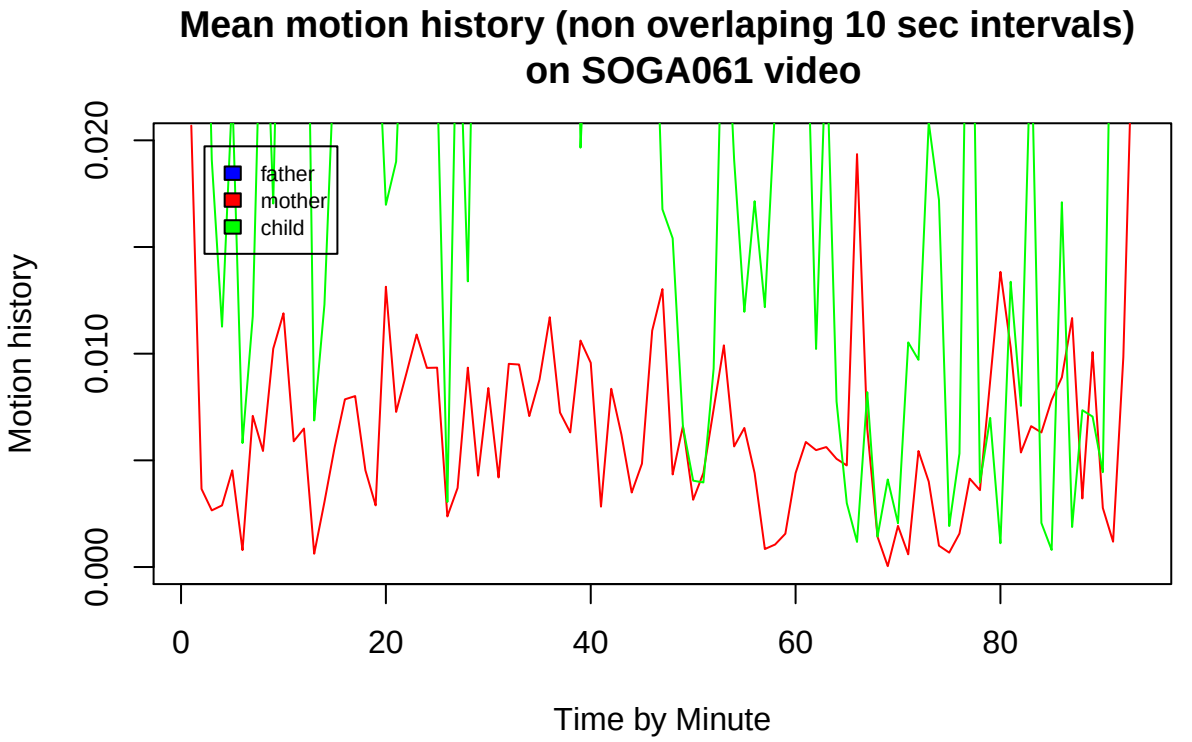

Mean motion history (non overlaping 10 sec intervals)
on RAEM049 video
Motion history
0.000
0.010
0.020
father
mother
child
0
20
40
60
80
Time by Minute

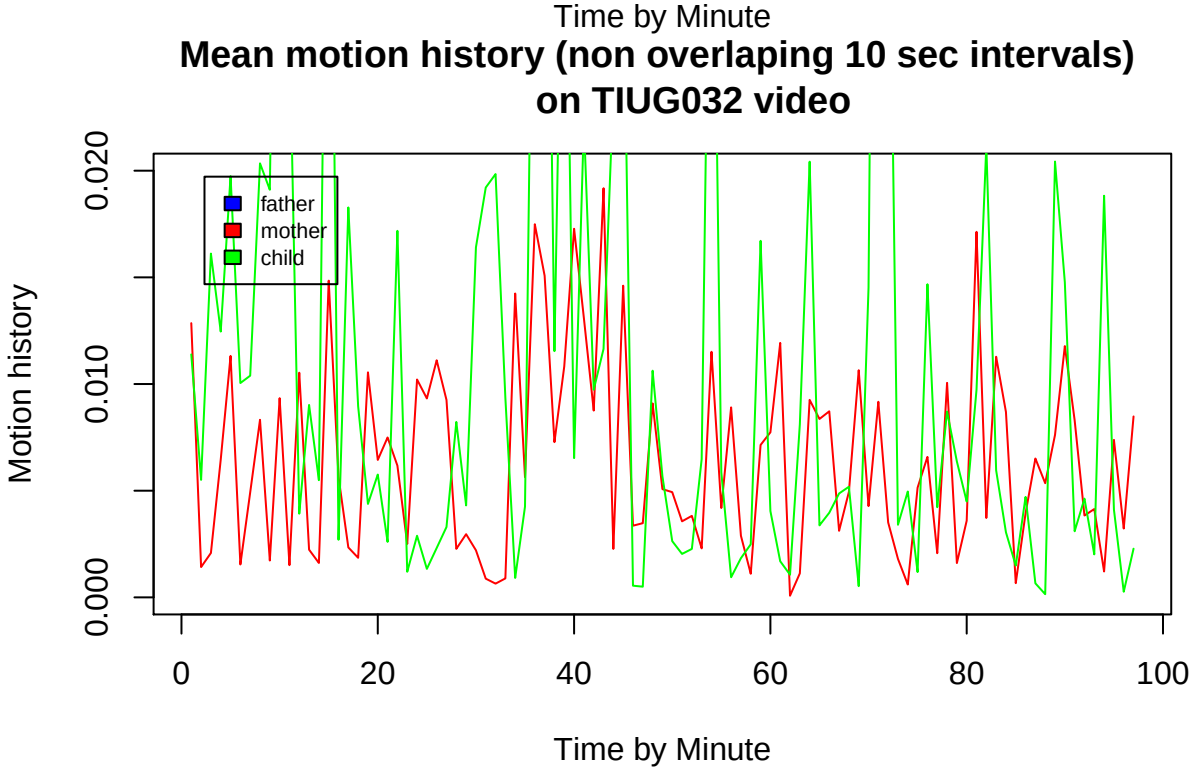

Mean motion history (non overlaping 10 sec intervals)
on RAKA008 video
Motion history
0.000
0.010
0.020
father
mother
child
0
20
40
60
80
Time by Minute

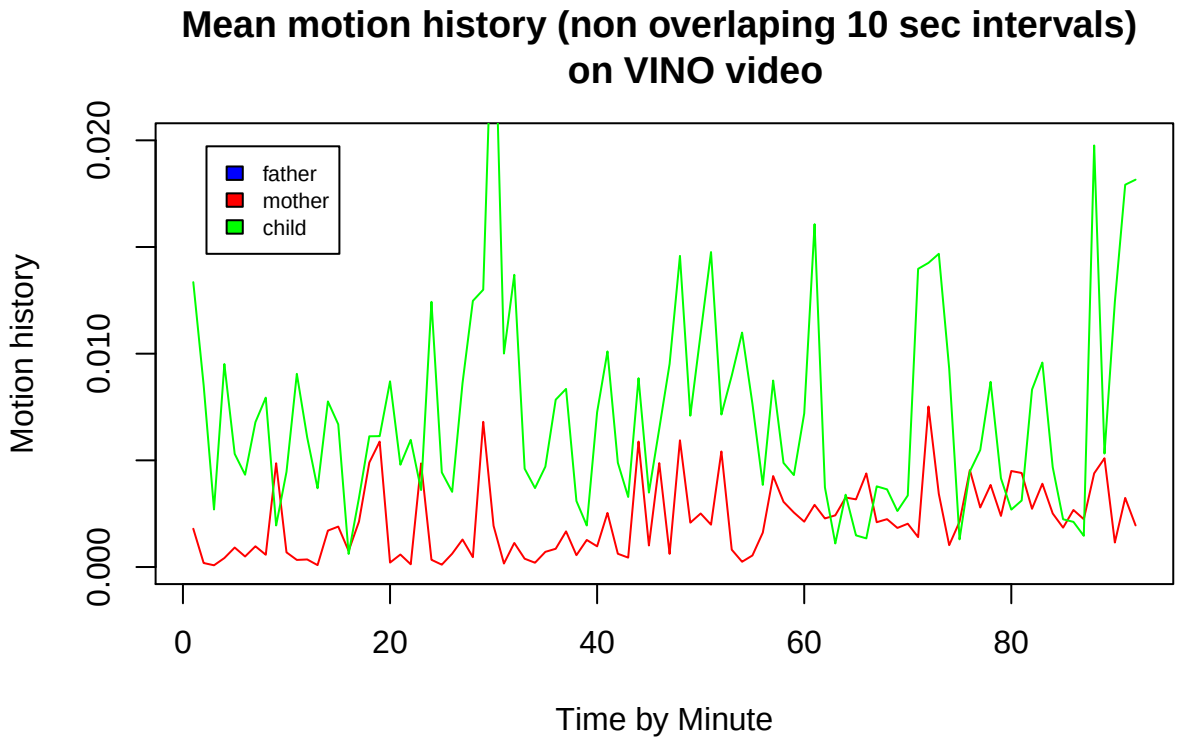

Mean motion history (non overlaping 10 sec intervals)
on RIEM0 video
Motion history
0.000
0.010
0.020
father
mother
child
0
20
40
60
80
Time by Minute


Mean motion history (non overlaping 10 sec intervals)
on SEEM035 video
Motion history
0.000
0.010
0.020
father
mother
child
0
20
40
60
80
Time by Minute


Mean motion history (non overlaping 10 sec intervals)
on SHAN042 video
Motion history
0.000
0.010
0.020
father
mother
child
0
20
40
60
80
Time by Minute


Mean motion history (non overlaping 10 sec intervals)
on SOGA061 video
Motion history
0.000
0.010
0.020
father
mother
child
0
20
40
60
80
Time by Minute


Mean motion history (non overlaping 10 sec intervals)
on TIUG032 video
Motion history
0.000
0.010
0.020
father
mother
child
0
20
40
60
80
100
Time by Minute



Export no log data in text files

```
videoIndex <- 1
# videoName is the name of current video
for (videoName in unique(data$family)){
# Compute slinding interval for each participant

      print(paste("Computing slidedFather", videoName))
      slidedFather <- SlidingInterval("father", videoIndex, 5, data)
#      print(head(data[which(data$family==videoName),]$father))
#      print(tail(data[which(data$family==videoName),]$father))
#      print(table(is.na(slidedFather)))
#      print(table(is.na(data[which(data$family==videoName),]$father)))

      print(paste("Computing slidedMother", videoName))
      slidedMother <- SlidingInterval("mother", videoIndex, 5, data)
#      print(head(data[which(data$family==videoName),]$mother))
#      print(tail(data[which(data$family==videoName),]$mother))
#      print(table(is.na(data[which(data$family==videoName),]$mother)))
#      print(table(is.na(slidedMother)))

      print(paste("Computing slidedChild", videoName))
      slidedChild <- SlidingInterval("child", videoIndex, 5, data)
#      print(head(data[which(data$family==videoName),]$child))
#      print(tail(data[which(data$family==videoName),]$child))
#      print(table(is.na(data[which(data$family==videoName),]$child)))
#      print(table(is.na(slidedChild)))
```

```
      slidedVideo <- data.frame(
        slidedFather, slidedMother, slidedChild,
        "video"=rep(families[videoIndex], length(slidedFather)),
        frame_index = 1:length(slidedFather))


#      slidedVideo.nas <- apply(slidedVideo, 1, function(x){all(is.na(x))})
#      slidedVideo <- slidedVideo[!slidedVideo.nas,]
#      indexes <-  data.frame ("video"=rep(families[videoIndex], length(slidedFather)),
#        frame_index = 1:length(slidedFather))
            write.csv(slidedVideo, paste("../Data/CSV/filtered/noLog/",videoName, ".slideddata.csv", sep
      videoIndex <-(videoIndex+1)
      }
```

Export log data in text files

```
videoIndex <- 1
# videoName is the name of current video
for (videoName in unique(data$family)){
# Compute slinding interval for each participant
      print(paste("Computing slidedFather", videoName))
      slidedFather <- SlidingInterval("logFather", videoIndex, 5, data)
      print(paste("Computing slidedMother", videoName))
      slidedMother <- SlidingInterval("logMother", videoIndex, 5, data)
      print(paste("Computing slidedChild", videoName))
      slidedChild <- SlidingInterval("logChild", videoIndex, 5, data)

      slidedVideo <- data.frame(
        slidedFather, slidedMother, slidedChild,
        "video"=rep(families[videoIndex], length(slidedFather)),
        frame_index = 1:length(slidedFather))

            write.csv(slidedVideo, paste("../Data/CSV/filtered/log/",videoName, ".log.slideddata.csv", s
      videoIndex <-(videoIndex+1)
      }
```

SyncPy utilisation for creating synchrony dataframe

After extracting filtered motion motion history with mean on sliding interval (overlapping interval) of 5 frames

And after puting this data on a CSV file slideddata.csv

We import this data on python Script with panda module Call_S_Estimator.py

This script will compute the synchrony between each dyad of the interaction and of the whole group

It will return a csv file for each video SSIXXXX.csv with XXXX the name of the video (F1044C, F1044D1, etc) that we can import with R with

this following function

```
SSIlog <- data.frame()
for (file in SSIlogFilesList){
    SSIalone <- read.csv(file)
    SSIlog<- rbind.fill(SSIlog, SSIalone)}
```

```
SSInoLog <- data.frame()
for (file in SSInoLogFilesList){
 #   print(file)
    SSIalone <- read.csv(file)
  #  print(str(SSIalone))
    SSInoLog<- rbind.fill(SSInoLog, SSIalone)}
```

Description of SSIlog data frame

```
str(SSIlog)
```

```
## 'data.frame':    3637 obs. of  8 variables:
##  $ X           : int  0 1 2 3 4 5 6 7 8 9 ...
##  $ Interval    : int  1 2 3 4 5 6 7 8 9 10 ...
##  $ Time_min    : num  0 0.167 0.333 0.5 0.667 ...
##  $ video       : int  37 37 37 37 37 37 37 37 37 37 ...
##  $ SSI_mo_ch   : num  0.04949 0.03257 0.31628 0.04023 0.00231 ...
##  $ SSI_fa_ch   : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ SSI_fa_mo   : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ SSI_fa_mo_ch: num  NA NA NA NA NA NA NA NA NA NA ...
```

```
#View(SSI)
table(SSIlog$video)
```

```
##
##    1   34   37   41   48  206 1106 1606
## 2966   93  102   92   94   92   99   99
```

```
SSIlog$video <- as.factor(SSIlog$video)
unique(SSIlog$video)
```

```
## [1] 37   34   41   48   206  1106 1606 1
## Levels: 1 34 37 41 48 206 1106 1606
```

```
#SSI <- SSI[-which(SSI$video==""),]
```

Description of noLogSSI data frame

```
str(SSInoLog)
```

```
## 'data.frame':    1807 obs. of  8 variables:
##  $ X           : int  0 1 2 3 4 5 6 7 8 9 ...
##  $ Interval    : int  1 2 3 4 5 6 7 8 9 10 ...
##  $ Time_min    : num  0 0.167 0.333 0.5 0.667 ...
##  $ video       : chr  "BALE050" "BALE050" "BALE050" "BALE050" ...
##  $ SSI_mo_pa   : num  0.01932 0.01111 0.13627 0.06911 0.00568 ...
##  $ SSI_fa_pa   : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ SSI_fa_mo   : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ SSI_fa_mo_pa: num  NA NA NA NA NA NA NA NA NA NA ...
```

```
#View(SSInoLog)
unique(SSInoLog$video)
```

```
##  [1] "BALE050" "34"      "37"      "41"      "48"      "206"     "1106"
##  [8] "1606"    "BAJE059" "BALU062" "BEAL036" "BEAM031" "BICA"    "BRLO041"
## [15] "COLO022" "DIPE004" "DOMA"    "DRNE"    "FOMA057" "GROP039" "HAJA052"
## [22] "HUMA058" "JAEM046" "JEEO040" "JOCE014" "LACL"    "MAEL048" "MAME20"
## [29] "MAPA029" "MIPH043" "MOSA065" "RAEM049" "RAKA008" "RIEM0"   "SEEM035"
## [36] "SHAN042" "SOGA061" "TIUG032" "VINO"
```

```
table(SSInoLog$video)
```

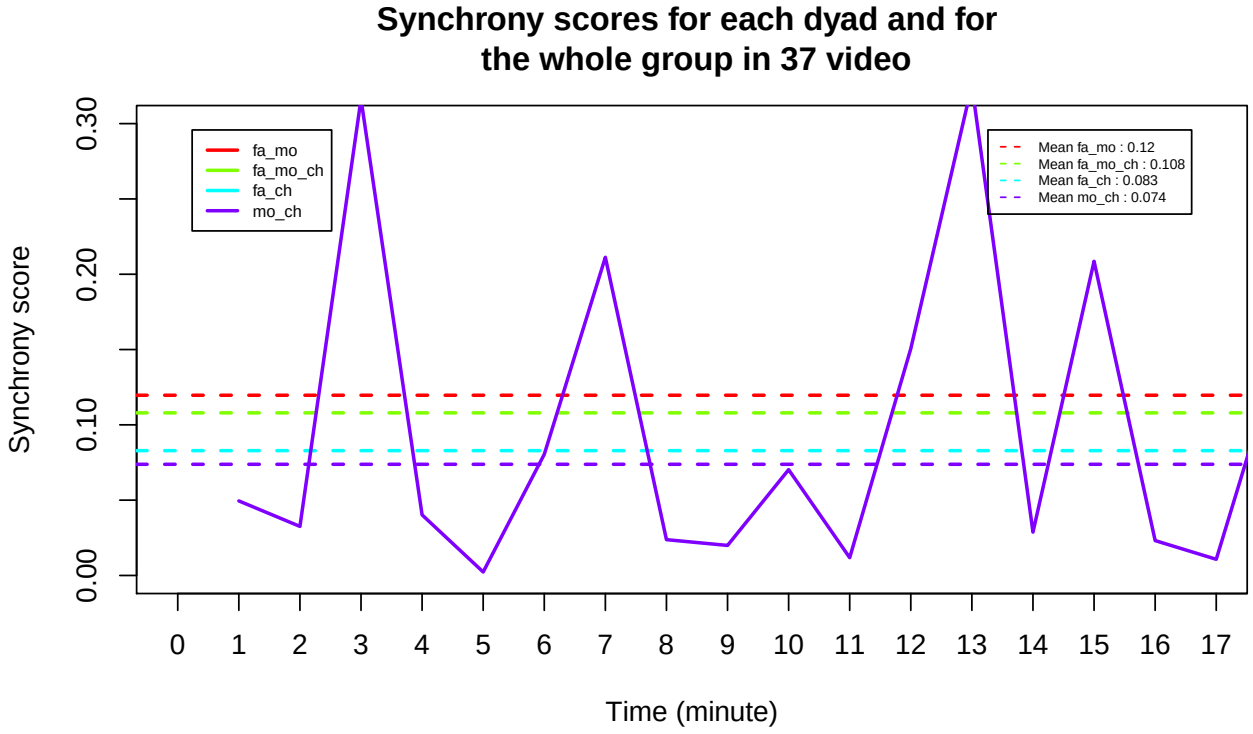
```
##
##    1106    1606     206      34      37      41      48 BAJE059 BALE050
##      49      49      46      46      51      46      47      50      47
## BALU062 BEAL036 BEAM031    BICA BRLO041 COLO022 DIPE004    DOMA    DRNE
##      48      46      45      46      46      48      48      49      48
## FOMA057 GROP039 HAJA052 HUMA058 JAEM046 JEEO040 JOCE014    LACL MAEL048
##      47      46      46      51      45      44      46      47      45
##  MAME20 MAPA029 MIPH043 MOSA065 RAEM049 RAKA008   RIEM0 SEEM035 SHAN042
##      47      21      46      53      45      46      45      47      46
## SOGA061 TIUG032    VINO
##      46      48      45
```

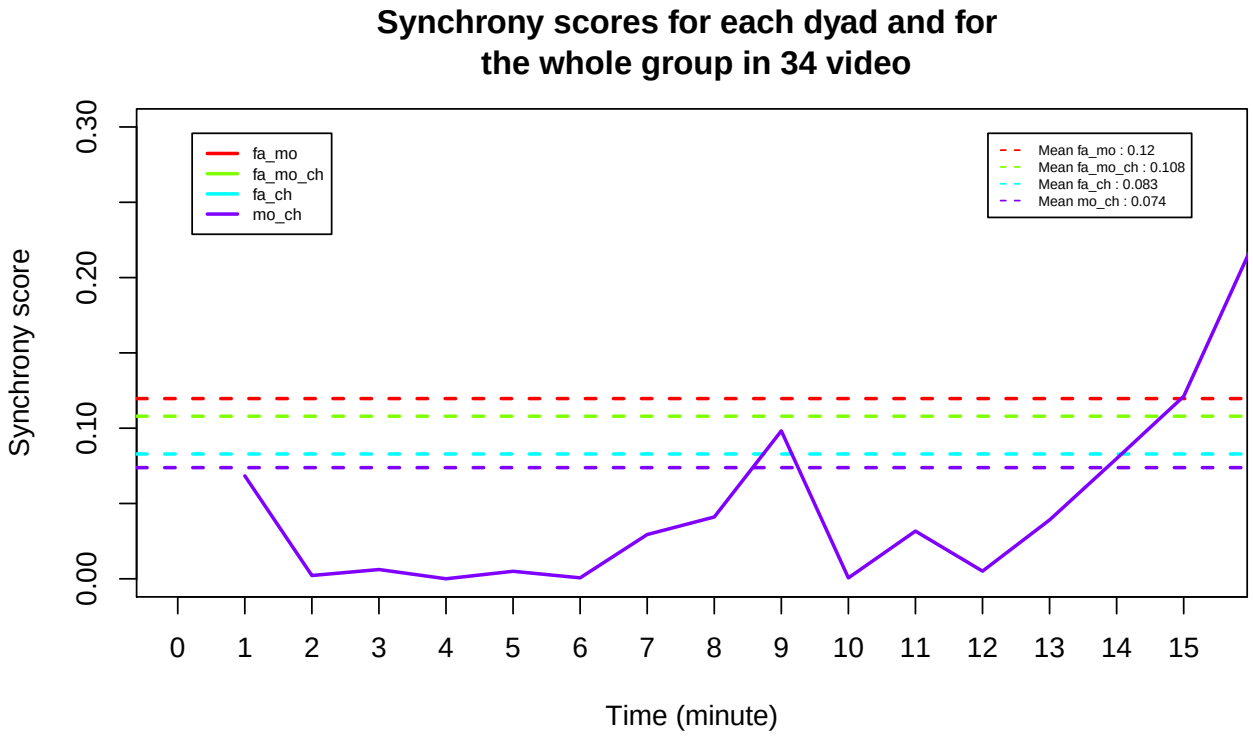
```
SSInoLog$video <- as.factor(SSInoLog$video)
unique(SSInoLog$video)
```

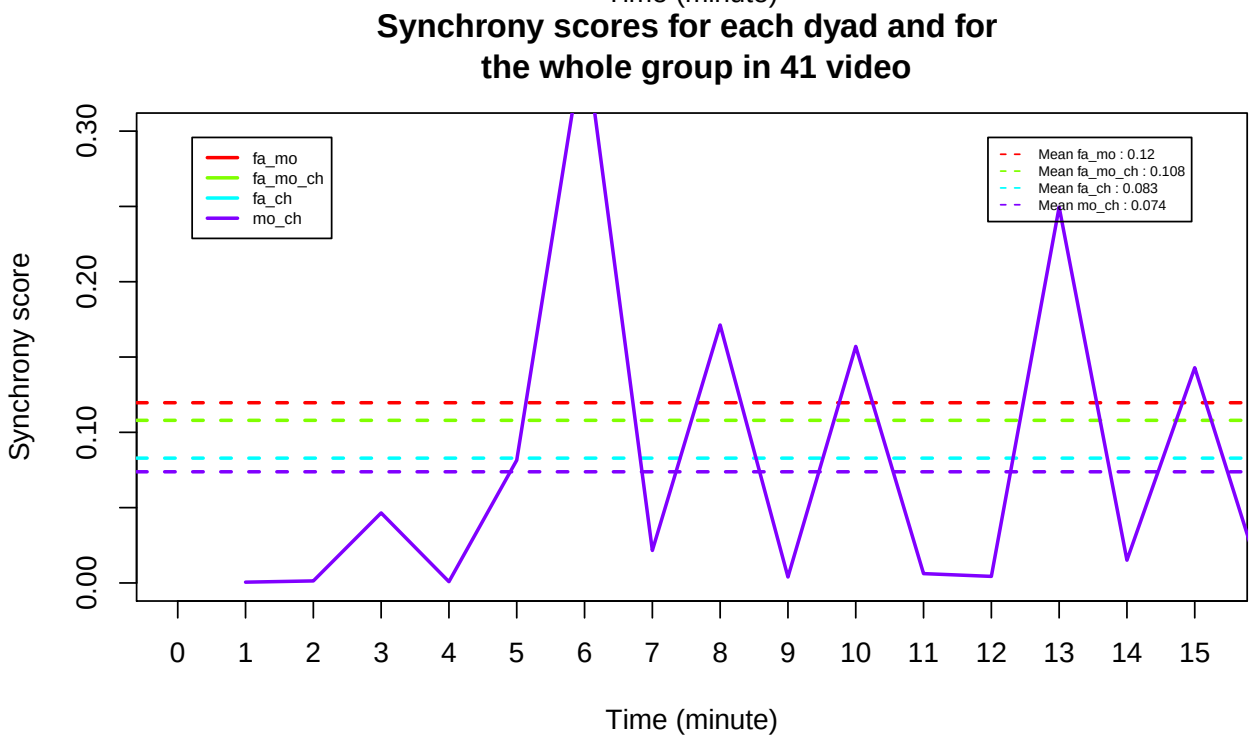
```
##  [1] BALE050 34      37      41      48      206     1106    1606
##  [9] BAJE059 BALU062 BEAL036 BEAM031 BICA    BRLO041 COLO022 DIPE004
## [17] DOMA    DRNE    FOMA057 GROP039 HAJA052 HUMA058 JAEM046 JEEO040
## [25] JOCE014 LACL    MAEL048 MAME20  MAPA029 MIPH043 MOSA065 RAEM049
## [33] RAKA008 RIEM0   SEEM035 SHAN042 SOGA061 TIUG032 VINO
## 39 Levels: 1106 1606 206 34 37 41 48 BAJE059 BALE050 BALU062 ... VINO
```

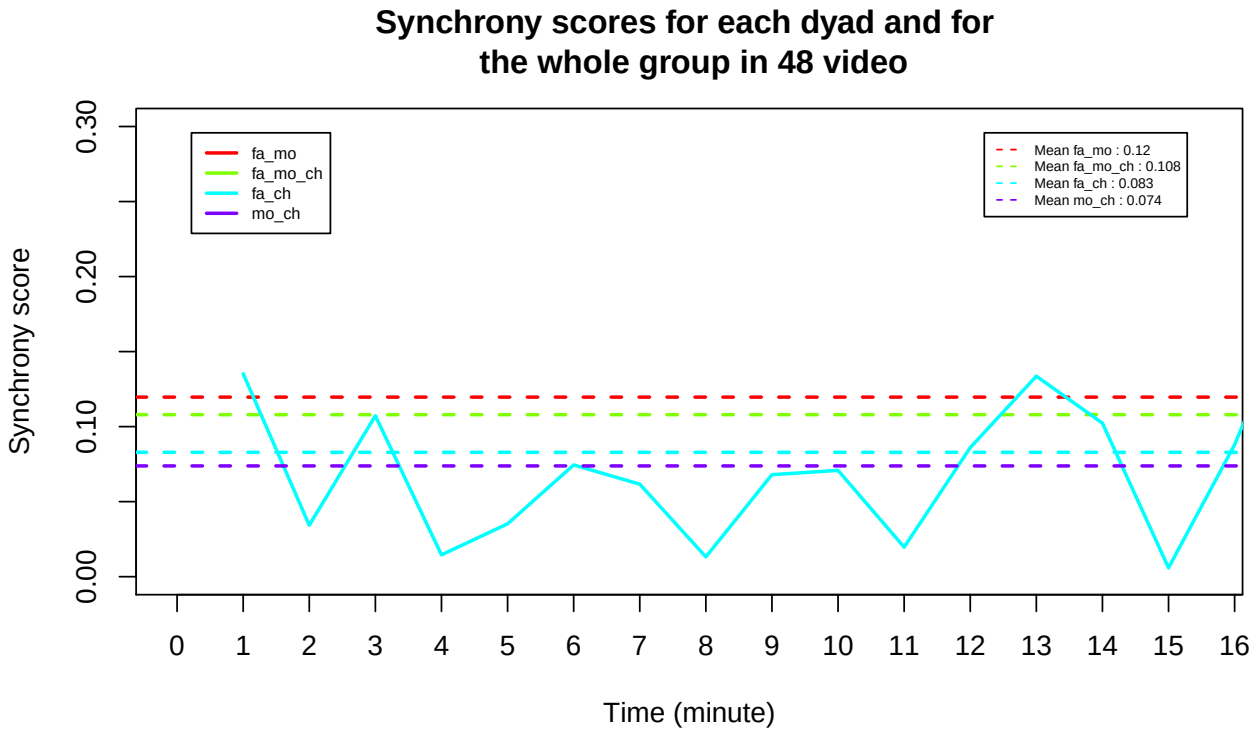
Synchrony scores log for each dyad, triad and for the whole group

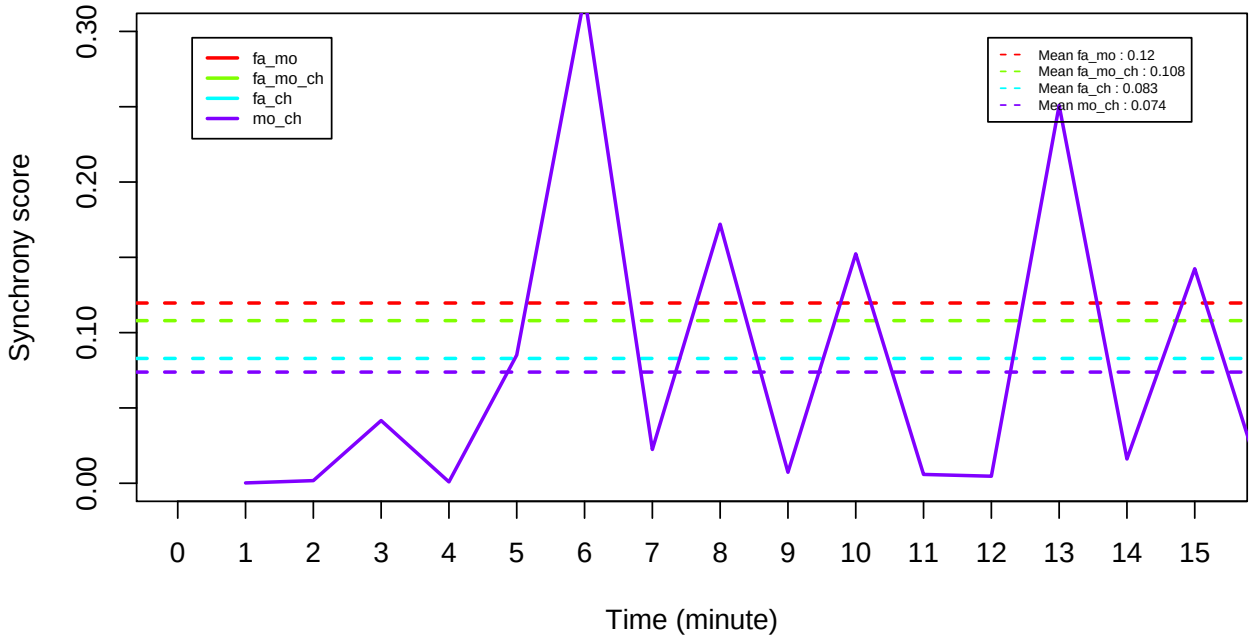
```
for (i in unique(SSIlog$video))
      {par(mar=c(4,4,4,3), mfrow=c(1,1))
      plot(SSIlog[which(SSIlog$video==i),]$Time_min,
           SSIlog[which(SSIlog$video==i),]$SSI_fa_mo,
           type="l", ylim=c(0, 0.3), col=rainbow(4)[1],
           main=paste("Synchrony scores for each dyad and for \n the whole group in", i, "video"),
           xlab = "Time (minute)", ylab="Synchrony score", lwd=2,
xaxp=c(0,length(SSIlog$Time_min), length(SSIlog$Time_min)))
      abline(h=mean(SSIlog$SSI_fa_mo, na.rm=TRUE), col=rainbow(4)[1], lwd=2, lty=2)
      lines(SSIlog[which(SSIlog$video==i),]$SSI_fa_mo_ch, col=rainbow(4)[2], lwd=2)
      abline(h= mean(SSIlog$SSI_fa_mo_ch, na.rm=TRUE), col=rainbow(4)[2], lwd=2, lty=2)
      lines(SSIlog[which(SSIlog$video==i),]$SSI_fa_ch, col=rainbow(4)[3], lwd=2)
      abline(h= mean(SSIlog$SSI_fa_ch, na.rm=TRUE), col=rainbow(4)[3], lwd=2, lty=2)
lines(SSIlog[which(SSIlog$video==i),]$SSI_mo_ch, col=rainbow(4)[4], lwd=2)
abline(h= mean(SSIlog$SSI_mo_ch, na.rm=TRUE), col=rainbow(4)[4], lwd=2, lty=2)

legend("topleft", inset=.05, c("fa_mo", "fa_mo_ch", "fa_ch",
"mo_ch"),
col=rainbow(4), cex=0.6, lwd=2)

legend ("topright", inset=.05, c(
      paste ("Mean fa_mo :", round(mean(SSIlog$SSI_fa_mo, na.rm=TRUE),3)),
      paste ("Mean fa_mo_ch :", round(mean(SSIlog$SSI_fa_mo_ch,na.rm=TRUE),3)),
      paste ("Mean fa_ch :", round(mean(SSIlog$SSI_fa_ch, na.rm=TRUE),3)),
      paste ("Mean mo_ch :", round(mean(SSIlog$SSI_mo_ch,na.rm=TRUE),3))),
col=rainbow(4), cex=0.5, lty=2, lwd=1)}
```

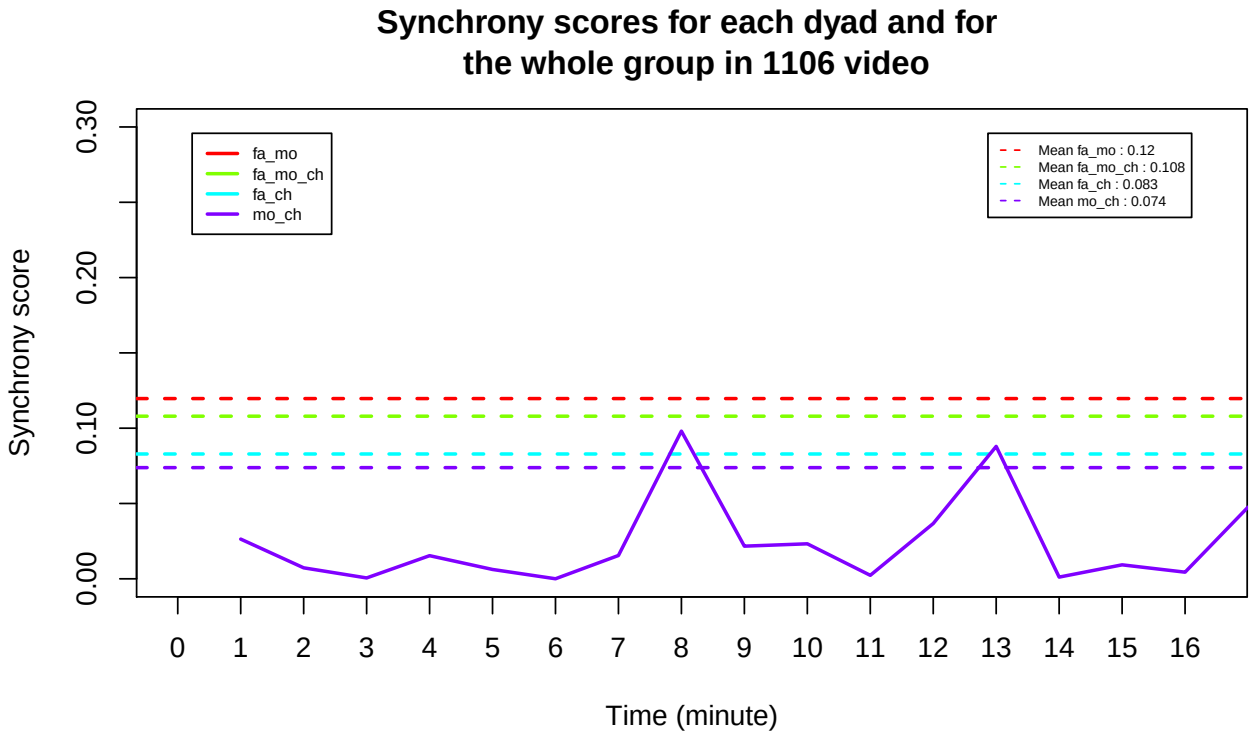


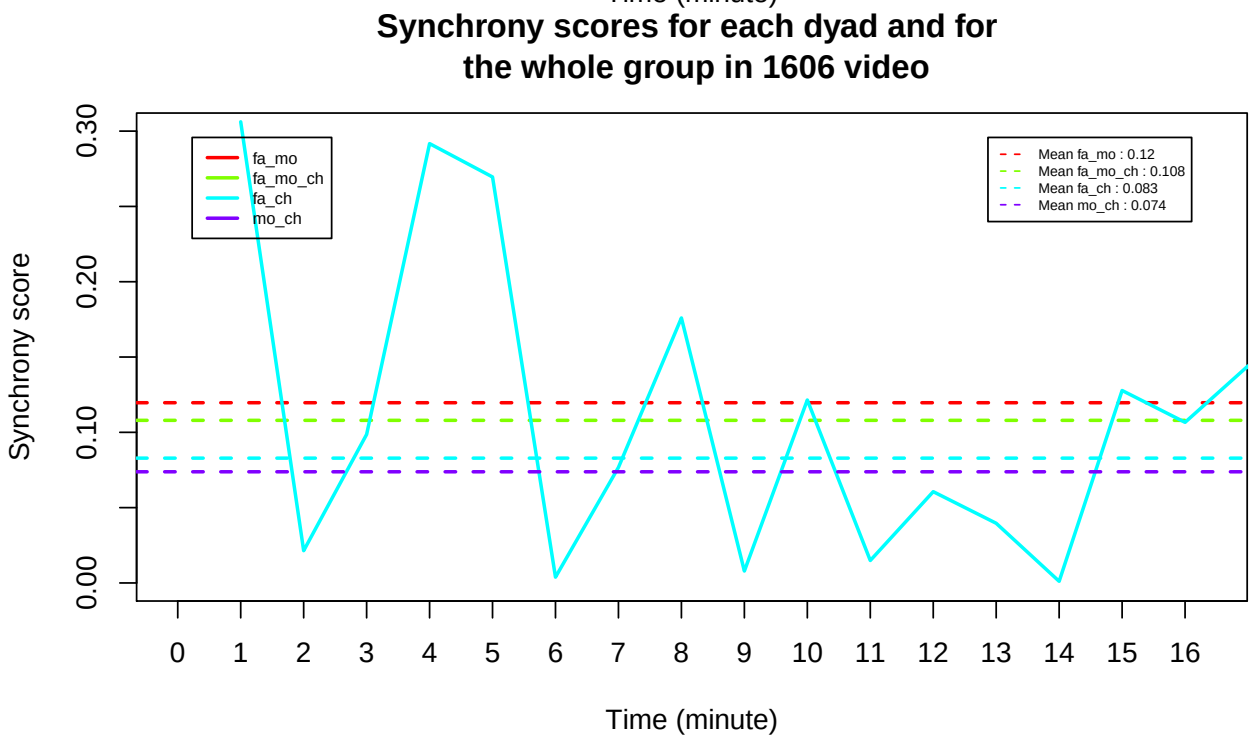

Synchrony scores for each dyad and for the whole group in 34 video
Synchrony score
0.00
0.10
0.20
0.30
fa_mo
fa_mo_ch
fa_ch
mo_ch
Mean fa_mo : 0.12
Mean fa_mo_ch : 0.108
Mean fa_ch : 0.083
Mean mo_ch : 0.074
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
Time (minute)

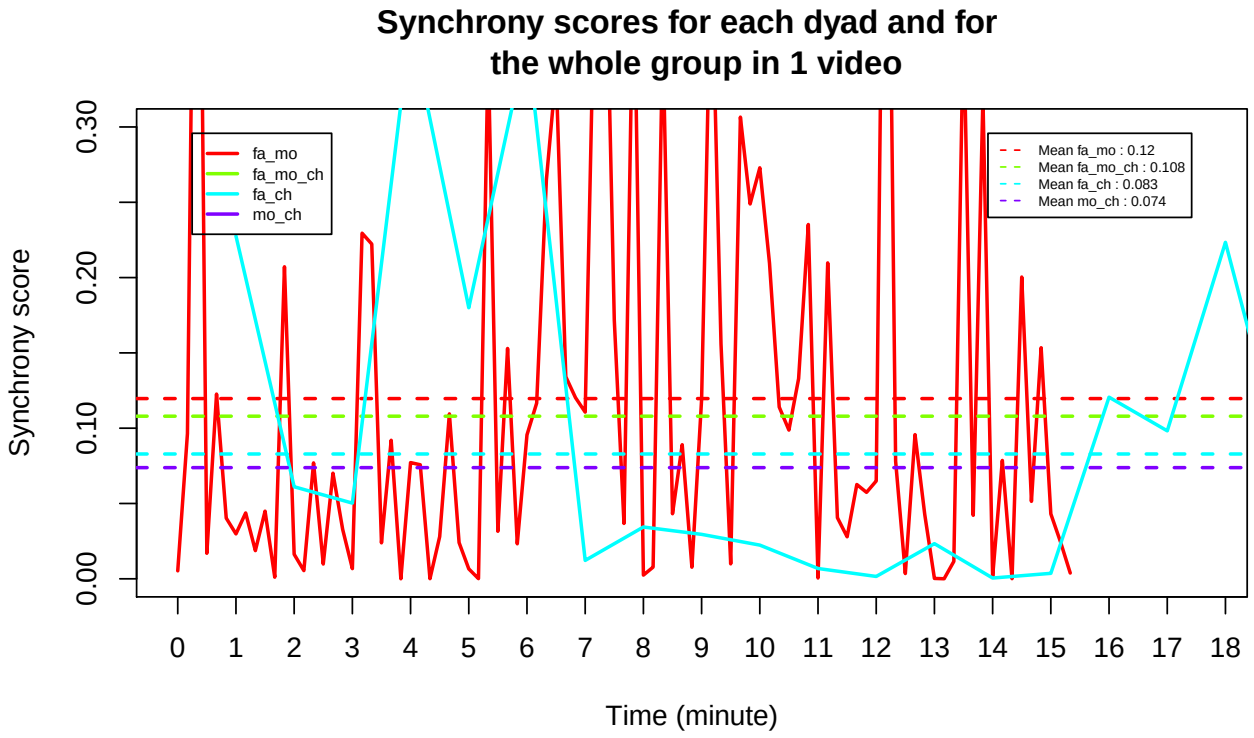

Synchrony scores for each dyad and for the whole group in 41 video
Synchrony score
0.00
0.10
0.20
0.30
fa_mo
fa_mo_ch
fa_ch
mo_ch
Mean fa_mo : 0.12
Mean fa_mo_ch : 0.108
Mean fa_ch : 0.083
Mean mo_ch : 0.074
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
Time (minute)


Synchrony scores for each dyad and for the whole group in 48 video
Synchrony score
0.00
0.10
0.20
0.30
fa_mo
fa_mo_ch
fa_ch
mo_ch
Mean fa_mo : 0.12
Mean fa_mo_ch : 0.108
Mean fa_ch : 0.083
Mean mo_ch : 0.074
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16
Time (minute)


Synchrony scores for each dyad and for the whole group in 206 video
Synchrony score
0.00
0.10
0.20
0.30
fa_mo
fa_mo_ch
fa_ch
mo_ch
Mean fa_mo : 0.12
Mean fa_mo_ch : 0.108
Mean fa_ch : 0.083
Mean mo_ch : 0.074
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
Time (minute)


Synchrony scores for each dyad and for the whole group in 1106 video
Synchrony score
0.00
0.10
0.20
0.30
fa_mo
fa_mo_ch
fa_ch
mo_ch
Mean fa_mo : 0.12
Mean fa_mo_ch : 0.108
Mean fa_ch : 0.083
Mean mo_ch : 0.074
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16
Time (minute)


Synchrony scores for each dyad and for the whole group in 1606 video
Synchrony score
0.00
0.10
0.20
0.30
fa_mo
fa_mo_ch
fa_ch
mo_ch
Mean fa_mo : 0.12
Mean fa_mo_ch : 0.108
Mean fa_ch : 0.083
Mean mo_ch : 0.074
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16
Time (minute)



Synchrony scores noLog for each dyad, triad and for the whole group

```
for (i in unique(SSInoLog$video))
      {par(mar=c(4,4,4,3), mfrow=c(1,1))
      plot(SSInoLog[which(SSInoLog$video==i),]$Time_min,
           SSInoLog[which(SSInoLog$video==i),]$SSI_fa_mo,
           type="l", ylim=c(0, 0.3), col=rainbow(4)[1],
           main=paste("Synchrony scores for each dyad and for \n the whole group in", i, "video"),
           xlab = "Time (minute)", ylab="Synchrony score", lwd=2,
xaxp=c(0,length(SSInoLog$Time_min), length(SSInoLog$Time_min)))
      abline(h=mean(SSInoLog$SSI_fa_mo, na.rm=TRUE), col=rainbow(4)[1], lwd=2, lty=2)
      lines(SSInoLog[which(SSInoLog$video==i),]$SSI_fa_mo_ch, col=rainbow(4)[2], lwd=2)
      abline(h= mean(SSInoLog$SSI_fa_mo_ch, na.rm=TRUE), col=rainbow(4)[2], lwd=2, lty=2)
      lines(SSInoLog[which(SSInoLog$video==i),]$SSI_fa_ch, col=rainbow(4)[3], lwd=2)
      abline(h= mean(SSInoLog$SSI_fa_ch, na.rm=TRUE), col=rainbow(4)[3], lwd=2, lty=2)
lines(SSInoLog[which(SSInoLog$video==i),]$SSI_mo_ch, col=rainbow(4)[4], lwd=2)
abline(h= mean(SSInoLog$SSI_mo_ch, na.rm=TRUE), col=rainbow(4)[4], lwd=2, lty=2)

legend("topleft", inset=.05, c("fa_mo", "fa_mo_ch", "fa_ch",
"mo_ch"),
col=rainbow(4), cex=0.6, lwd=2)

legend ("topright", inset=.05, c(
      paste ("Mean fa_mo :", round(mean(SSInoLog$SSI_fa_mo, na.rm=TRUE),3)),
      paste ("Mean fa_mo_ch :", round(mean(SSInoLog$SSI_fa_mo_ch,na.rm=TRUE),3)),
      paste ("Mean fa_ch :", round(mean(SSInoLog$SSI_fa_ch, na.rm=TRUE),3)),
```

```
      paste ("Mean mo_ch :", round(mean(SSInoLog$SSI_mo_ch,na.rm=TRUE),3))),
col=rainbow(4), cex=0.5, lty=2, lwd=1)}
```

```
## Warning in mean.default(SSInoLog$SSI_fa_mo_ch, na.rm = TRUE): argument is
## not numeric or logical: returning NA
```

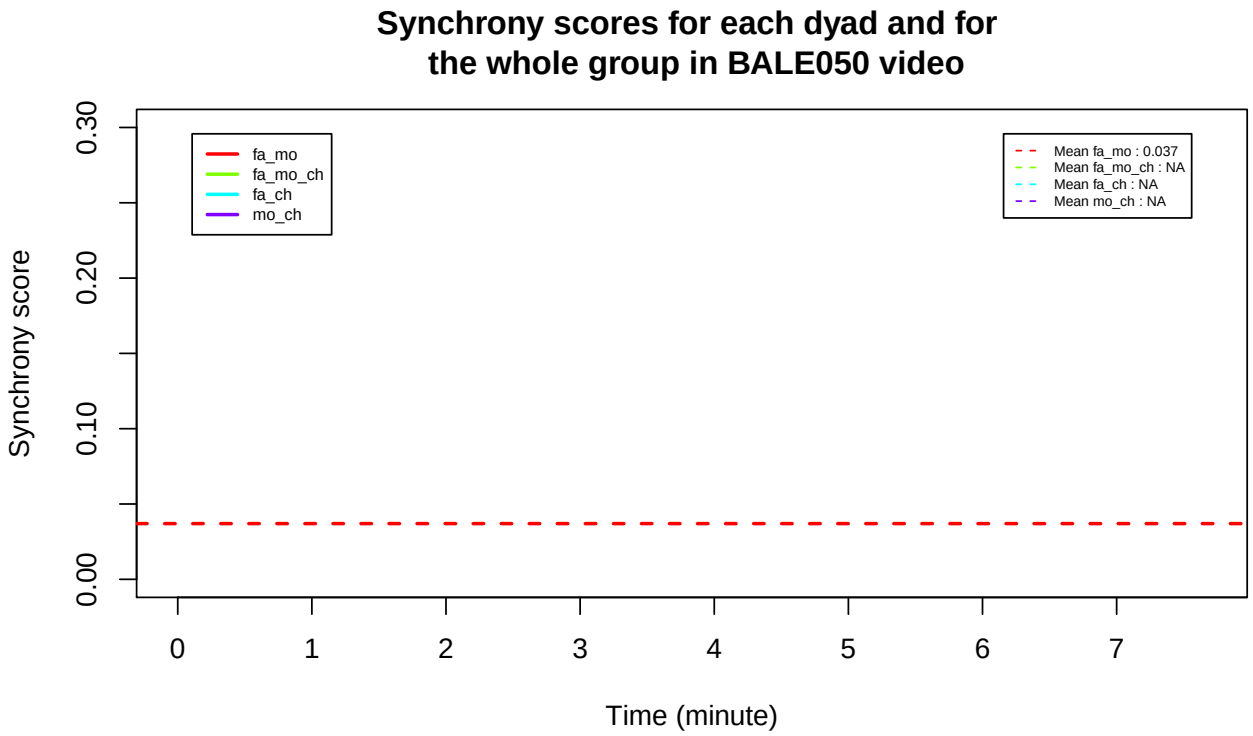
```
## Warning in mean.default(SSInoLog$SSI_fa_ch, na.rm = TRUE): argument is not
## numeric or logical: returning NA
```

```
## Warning in mean.default(SSInoLog$SSI_mo_ch, na.rm = TRUE): argument is not
## numeric or logical: returning NA
```

```
## Warning in mean.default(SSInoLog$SSI_fa_mo_ch, na.rm = TRUE): argument is
## not numeric or logical: returning NA
```

```
## Warning in mean.default(SSInoLog$SSI_fa_ch, na.rm = TRUE): argument is not
## numeric or logical: returning NA
```

```
## Warning in mean.default(SSInoLog$SSI_mo_ch, na.rm = TRUE): argument is not
## numeric or logical: returning NA
```




```
## Warning in mean.default(SSInoLog$SSI_fa_mo_ch, na.rm = TRUE): argument is
## not numeric or logical: returning NA
```

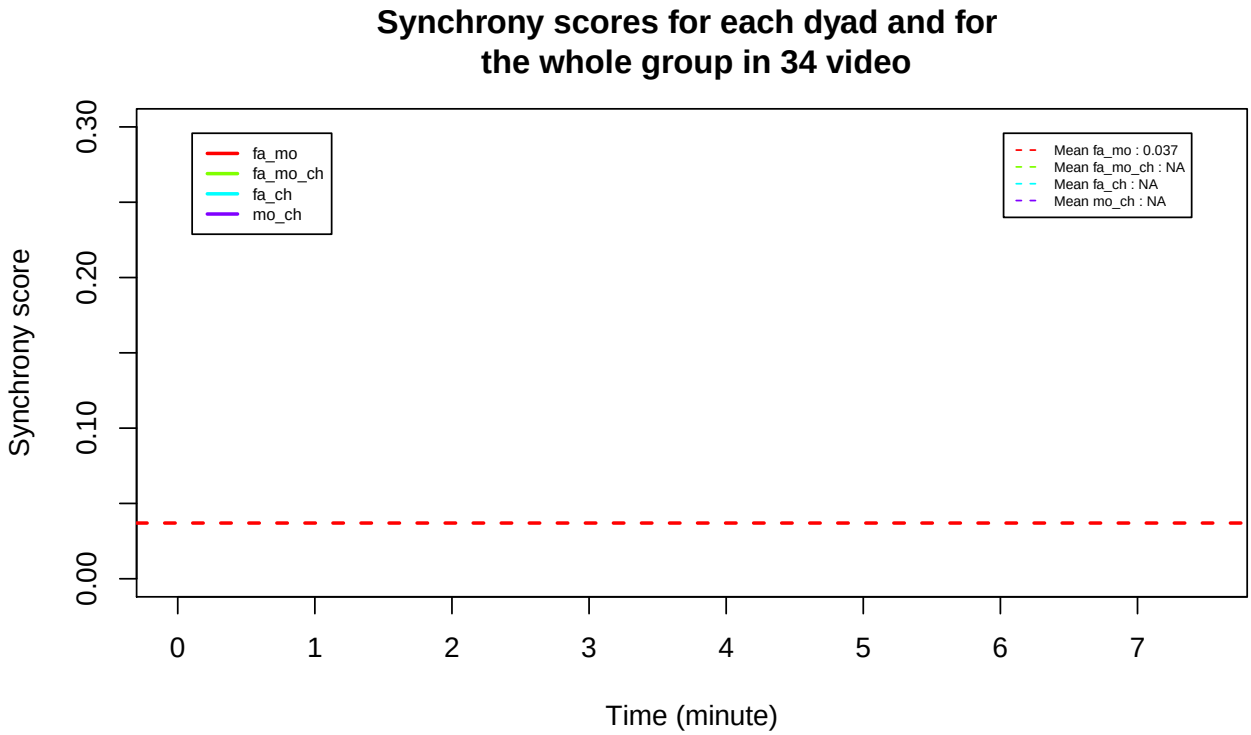
```
## Warning in mean.default(SSInoLog$SSI_fa_ch, na.rm = TRUE): argument is not
## numeric or logical: returning NA
```

```
## Warning in mean.default(SSInoLog$SSI_mo_ch, na.rm = TRUE): argument is not
## numeric or logical: returning NA
```

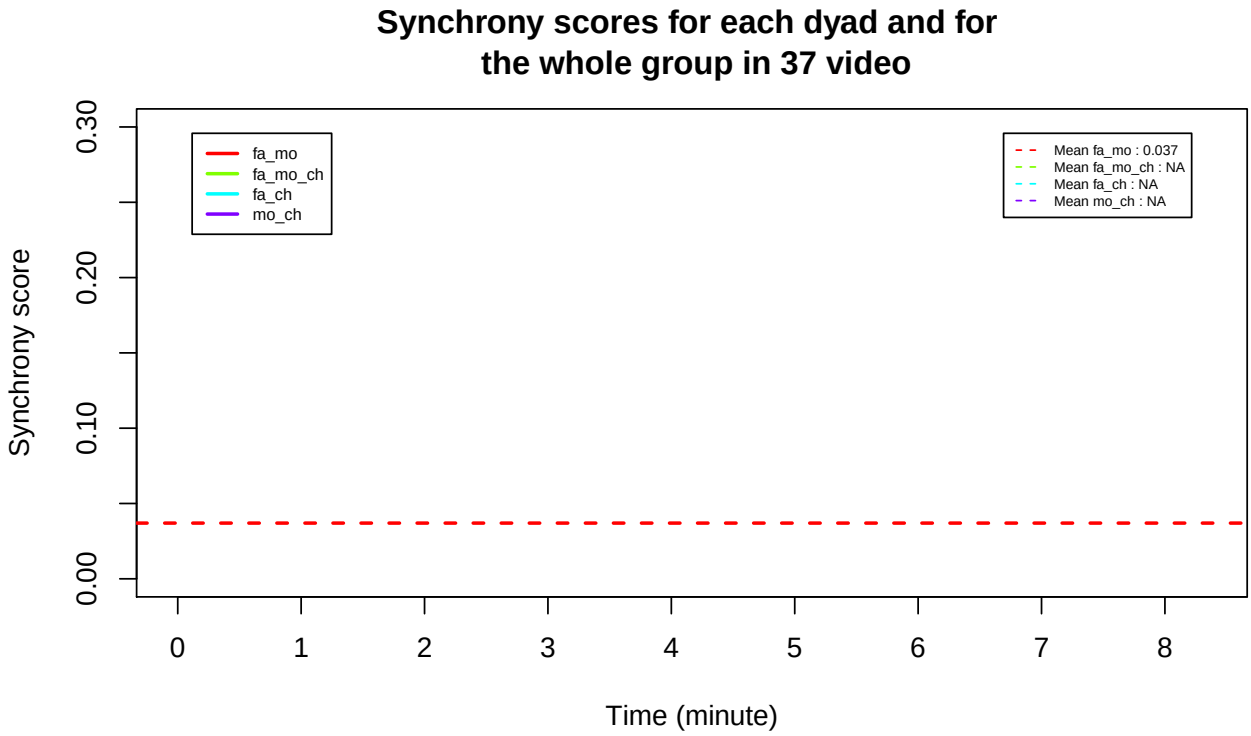
```
## Warning in mean.default(SSInoLog$SSI_fa_mo_ch, na.rm = TRUE): argument is
## not numeric or logical: returning NA
```

```
## Warning in mean.default(SSInoLog$SSI_fa_ch, na.rm = TRUE): argument is not
## numeric or logical: returning NA
```

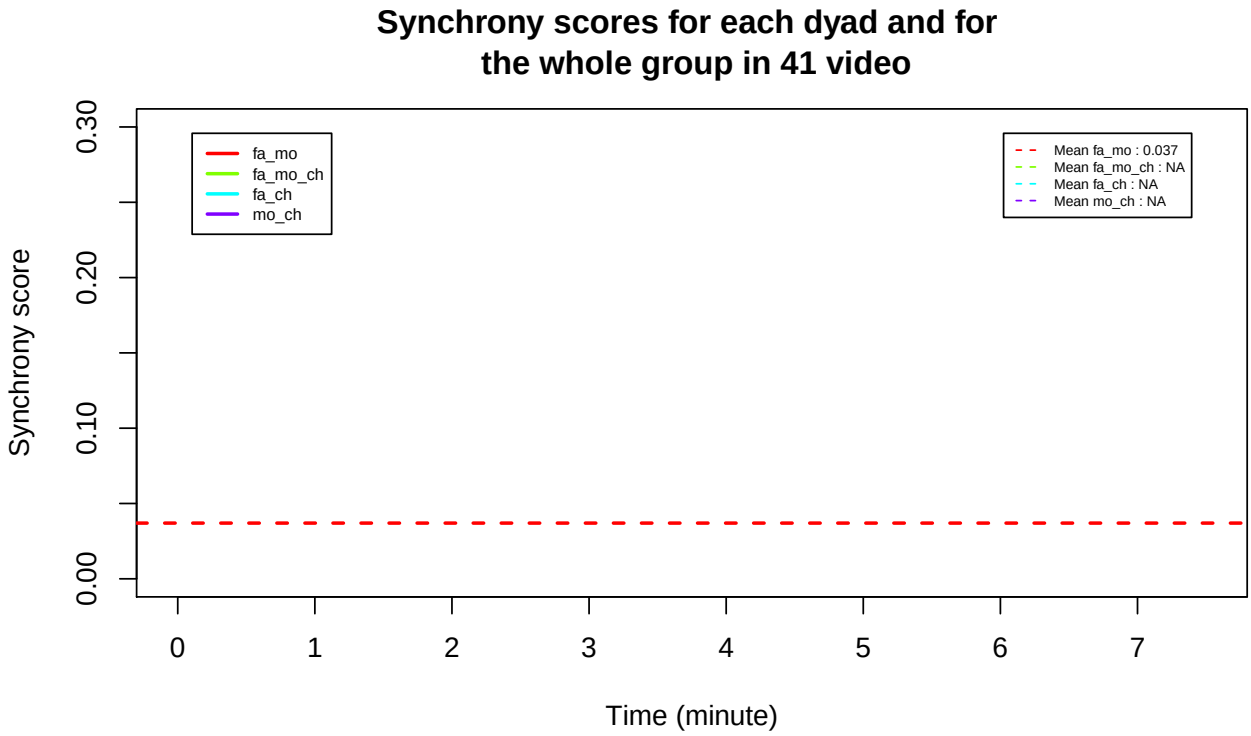
```
## Warning in mean.default(SSInoLog$SSI_mo_ch, na.rm = TRUE): argument is not
## numeric or logical: returning NA
```



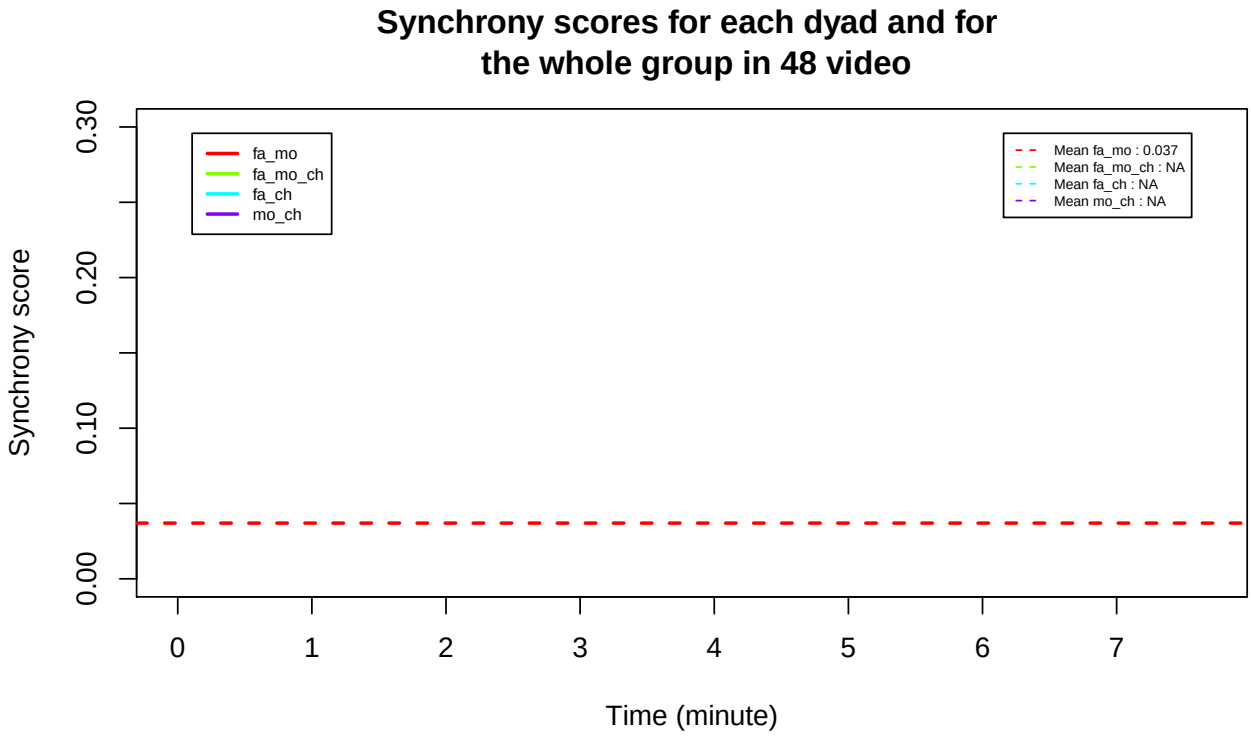

```
## Warning in mean.default(SSInoLog$SSI_fa_mo_ch, na.rm = TRUE): argument is
## not numeric or logical: returning NA

## Warning in mean.default(SSInoLog$SSI_fa_ch, na.rm = TRUE): argument is not
## numeric or logical: returning NA

## Warning in mean.default(SSInoLog$SSI_mo_ch, na.rm = TRUE): argument is not
## numeric or logical: returning NA

## Warning in mean.default(SSInoLog$SSI_fa_mo_ch, na.rm = TRUE): argument is
## not numeric or logical: returning NA

## Warning in mean.default(SSInoLog$SSI_fa_ch, na.rm = TRUE): argument is not
## numeric or logical: returning NA

## Warning in mean.default(SSInoLog$SSI_mo_ch, na.rm = TRUE): argument is not
## numeric or logical: returning NA
```



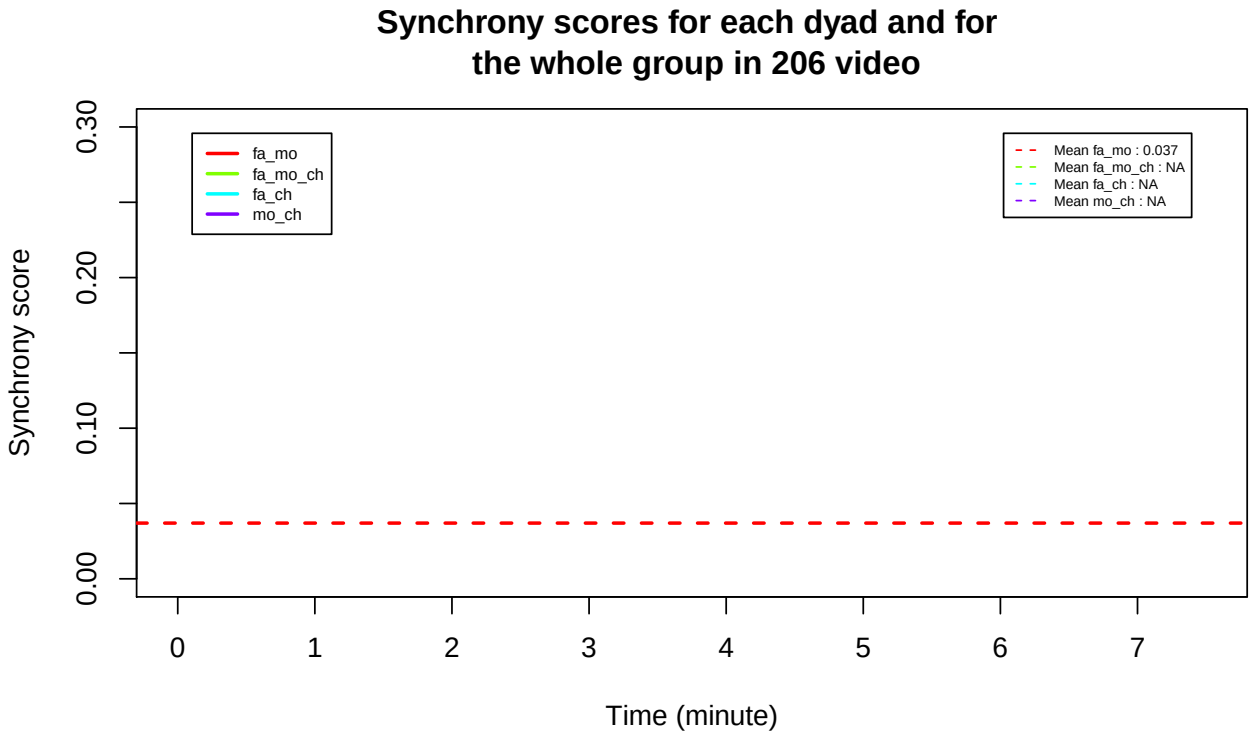

```
## Warning in mean.default(SSInoLog$SSI_fa_mo_ch, na.rm = TRUE): argument is
## not numeric or logical: returning NA

## Warning in mean.default(SSInoLog$SSI_fa_ch, na.rm = TRUE): argument is not
## numeric or logical: returning NA

## Warning in mean.default(SSInoLog$SSI_mo_ch, na.rm = TRUE): argument is not
## numeric or logical: returning NA

## Warning in mean.default(SSInoLog$SSI_fa_mo_ch, na.rm = TRUE): argument is
## not numeric or logical: returning NA

## Warning in mean.default(SSInoLog$SSI_fa_ch, na.rm = TRUE): argument is not
## numeric or logical: returning NA

## Warning in mean.default(SSInoLog$SSI_mo_ch, na.rm = TRUE): argument is not
## numeric or logical: returning NA
```



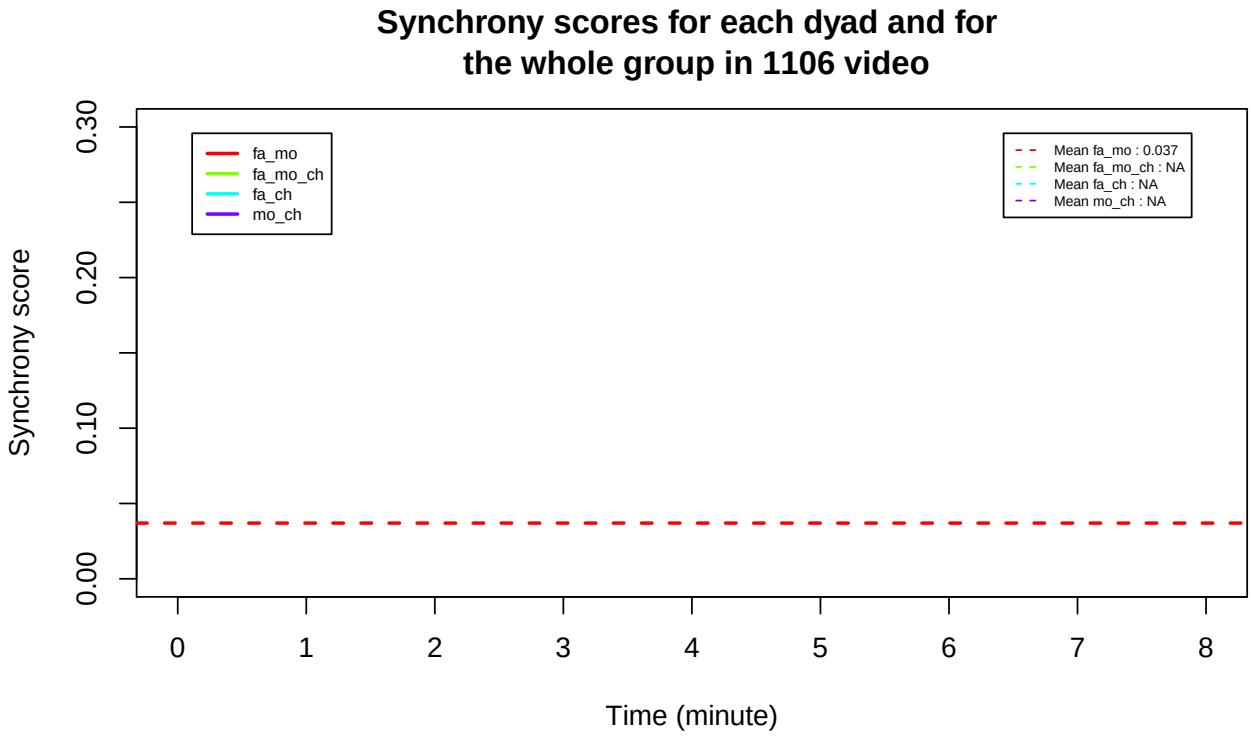

```
## Warning in mean.default(SSInoLog$SSI_fa_mo_ch, na.rm = TRUE): argument is
## not numeric or logical: returning NA

## Warning in mean.default(SSInoLog$SSI_fa_ch, na.rm = TRUE): argument is not
## numeric or logical: returning NA

## Warning in mean.default(SSInoLog$SSI_mo_ch, na.rm = TRUE): argument is not
## numeric or logical: returning NA

## Warning in mean.default(SSInoLog$SSI_fa_mo_ch, na.rm = TRUE): argument is
## not numeric or logical: returning NA

## Warning in mean.default(SSInoLog$SSI_fa_ch, na.rm = TRUE): argument is not
## numeric or logical: returning NA

## Warning in mean.default(SSInoLog$SSI_mo_ch, na.rm = TRUE): argument is not
## numeric or logical: returning NA
```



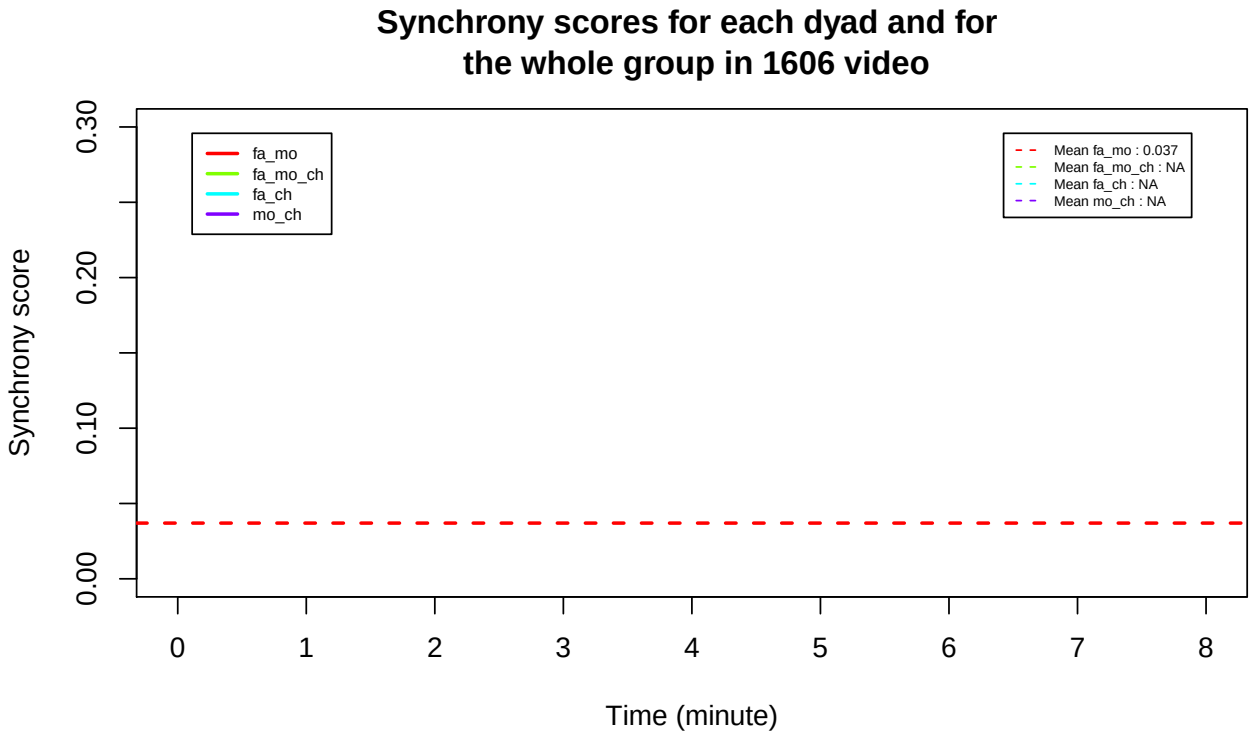

```
## Warning in mean.default(SSInoLog$SSI_fa_mo_ch, na.rm = TRUE): argument is
## not numeric or logical: returning NA

## Warning in mean.default(SSInoLog$SSI_fa_ch, na.rm = TRUE): argument is not
## numeric or logical: returning NA

## Warning in mean.default(SSInoLog$SSI_mo_ch, na.rm = TRUE): argument is not
## numeric or logical: returning NA

## Warning in mean.default(SSInoLog$SSI_fa_mo_ch, na.rm = TRUE): argument is
## not numeric or logical: returning NA

## Warning in mean.default(SSInoLog$SSI_fa_ch, na.rm = TRUE): argument is not
## numeric or logical: returning NA

## Warning in mean.default(SSInoLog$SSI_mo_ch, na.rm = TRUE): argument is not
## numeric or logical: returning NA
```



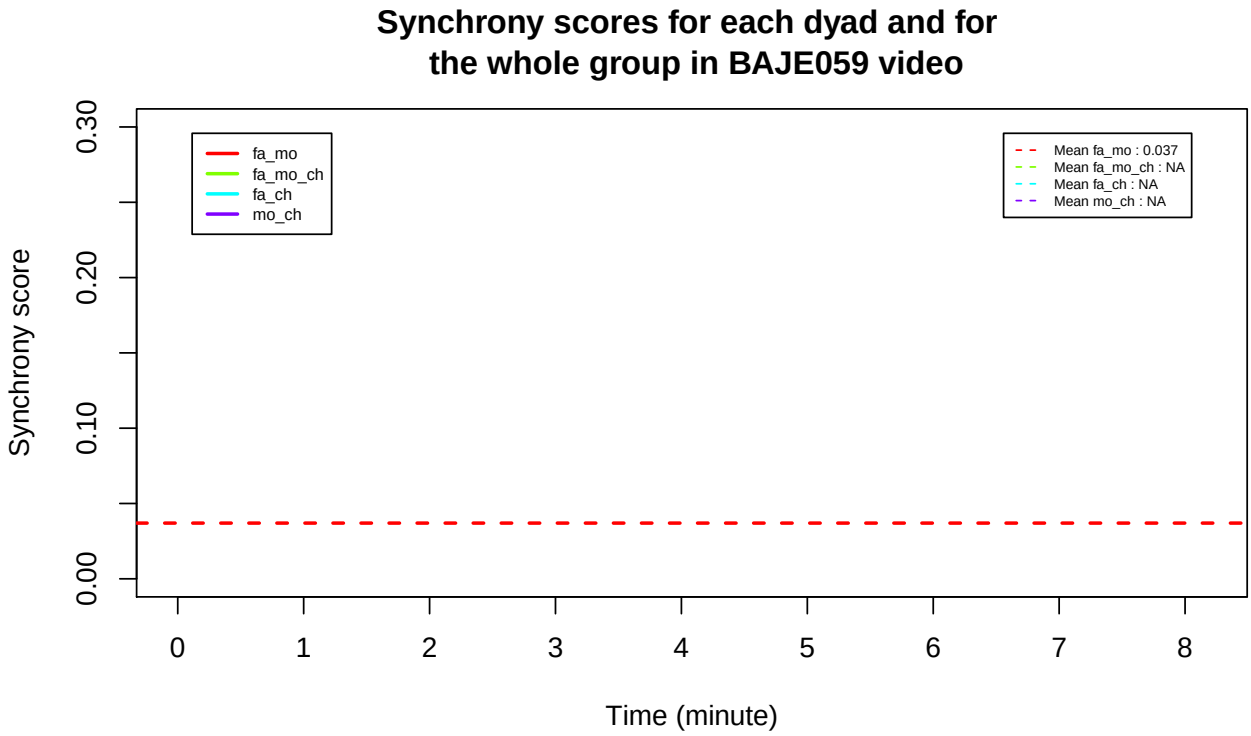

```
## Warning in mean.default(SSInoLog$SSI_fa_mo_ch, na.rm = TRUE): argument is
## not numeric or logical: returning NA

## Warning in mean.default(SSInoLog$SSI_fa_ch, na.rm = TRUE): argument is not
## numeric or logical: returning NA

## Warning in mean.default(SSInoLog$SSI_mo_ch, na.rm = TRUE): argument is not
## numeric or logical: returning NA

## Warning in mean.default(SSInoLog$SSI_fa_mo_ch, na.rm = TRUE): argument is
## not numeric or logical: returning NA

## Warning in mean.default(SSInoLog$SSI_fa_ch, na.rm = TRUE): argument is not
## numeric or logical: returning NA

## Warning in mean.default(SSInoLog$SSI_mo_ch, na.rm = TRUE): argument is not
## numeric or logical: returning NA
```



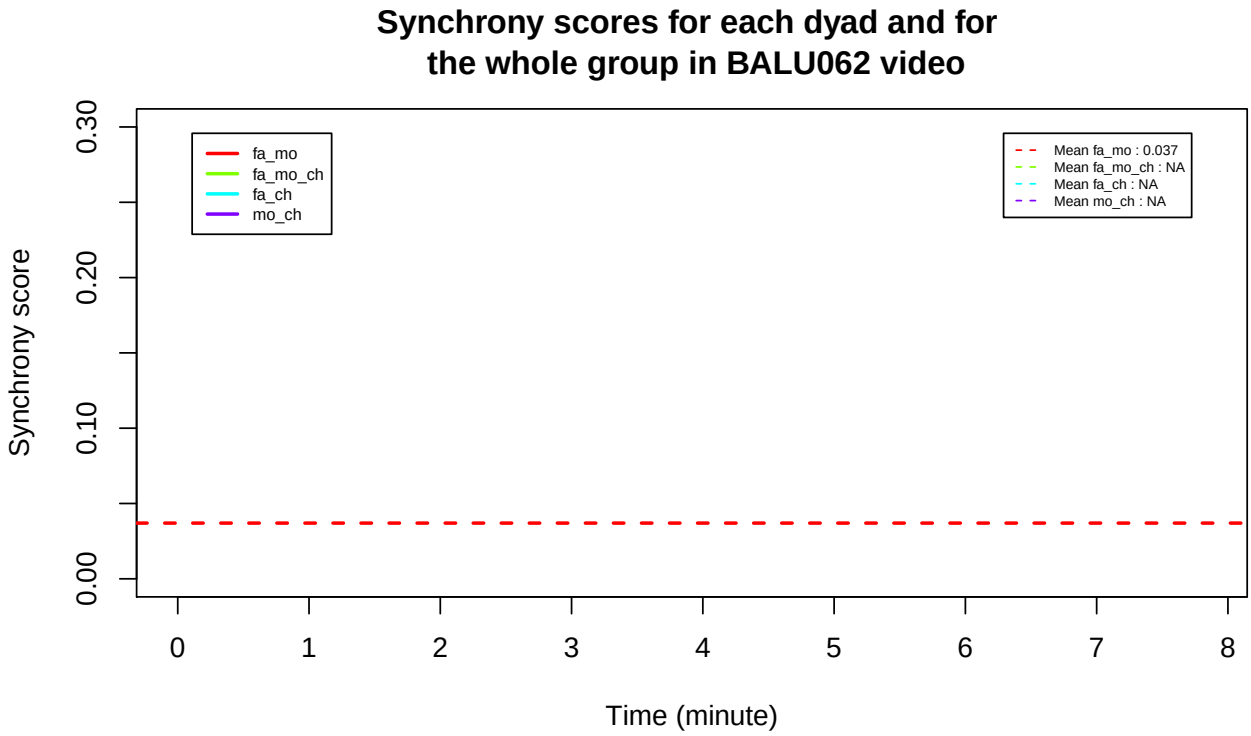

```
## Warning in mean.default(SSInoLog$SSI_fa_mo_ch, na.rm = TRUE): argument is
## not numeric or logical: returning NA

## Warning in mean.default(SSInoLog$SSI_fa_ch, na.rm = TRUE): argument is not
## numeric or logical: returning NA

## Warning in mean.default(SSInoLog$SSI_mo_ch, na.rm = TRUE): argument is not
## numeric or logical: returning NA

## Warning in mean.default(SSInoLog$SSI_fa_mo_ch, na.rm = TRUE): argument is
## not numeric or logical: returning NA

## Warning in mean.default(SSInoLog$SSI_fa_ch, na.rm = TRUE): argument is not
## numeric or logical: returning NA

## Warning in mean.default(SSInoLog$SSI_mo_ch, na.rm = TRUE): argument is not
## numeric or logical: returning NA
```



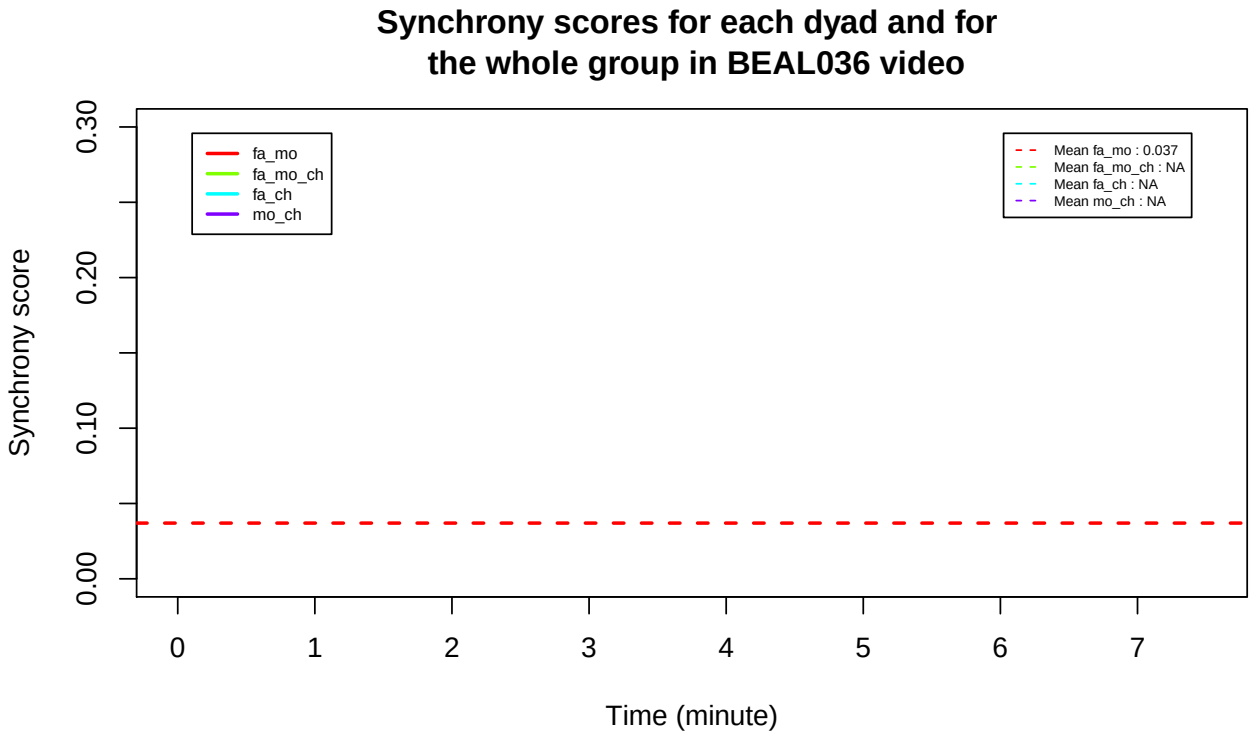

```
## Warning in mean.default(SSInoLog$SSI_fa_mo_ch, na.rm = TRUE): argument is
## not numeric or logical: returning NA

## Warning in mean.default(SSInoLog$SSI_fa_ch, na.rm = TRUE): argument is not
## numeric or logical: returning NA

## Warning in mean.default(SSInoLog$SSI_mo_ch, na.rm = TRUE): argument is not
## numeric or logical: returning NA

## Warning in mean.default(SSInoLog$SSI_fa_mo_ch, na.rm = TRUE): argument is
## not numeric or logical: returning NA

## Warning in mean.default(SSInoLog$SSI_fa_ch, na.rm = TRUE): argument is not
## numeric or logical: returning NA

## Warning in mean.default(SSInoLog$SSI_mo_ch, na.rm = TRUE): argument is not
## numeric or logical: returning NA
```



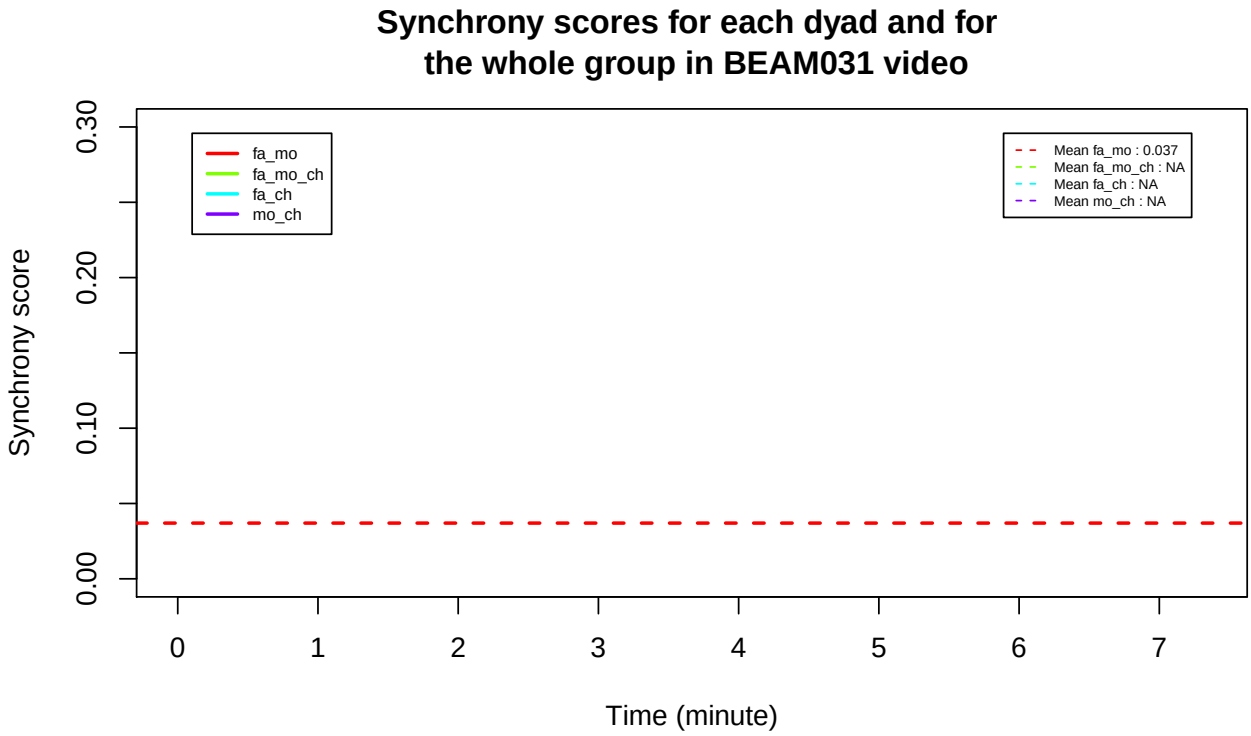

```
## Warning in mean.default(SSInoLog$SSI_fa_mo_ch, na.rm = TRUE): argument is
## not numeric or logical: returning NA

## Warning in mean.default(SSInoLog$SSI_fa_ch, na.rm = TRUE): argument is not
## numeric or logical: returning NA

## Warning in mean.default(SSInoLog$SSI_mo_ch, na.rm = TRUE): argument is not
## numeric or logical: returning NA

## Warning in mean.default(SSInoLog$SSI_fa_mo_ch, na.rm = TRUE): argument is
## not numeric or logical: returning NA

## Warning in mean.default(SSInoLog$SSI_fa_ch, na.rm = TRUE): argument is not
## numeric or logical: returning NA

## Warning in mean.default(SSInoLog$SSI_mo_ch, na.rm = TRUE): argument is not
## numeric or logical: returning NA
```



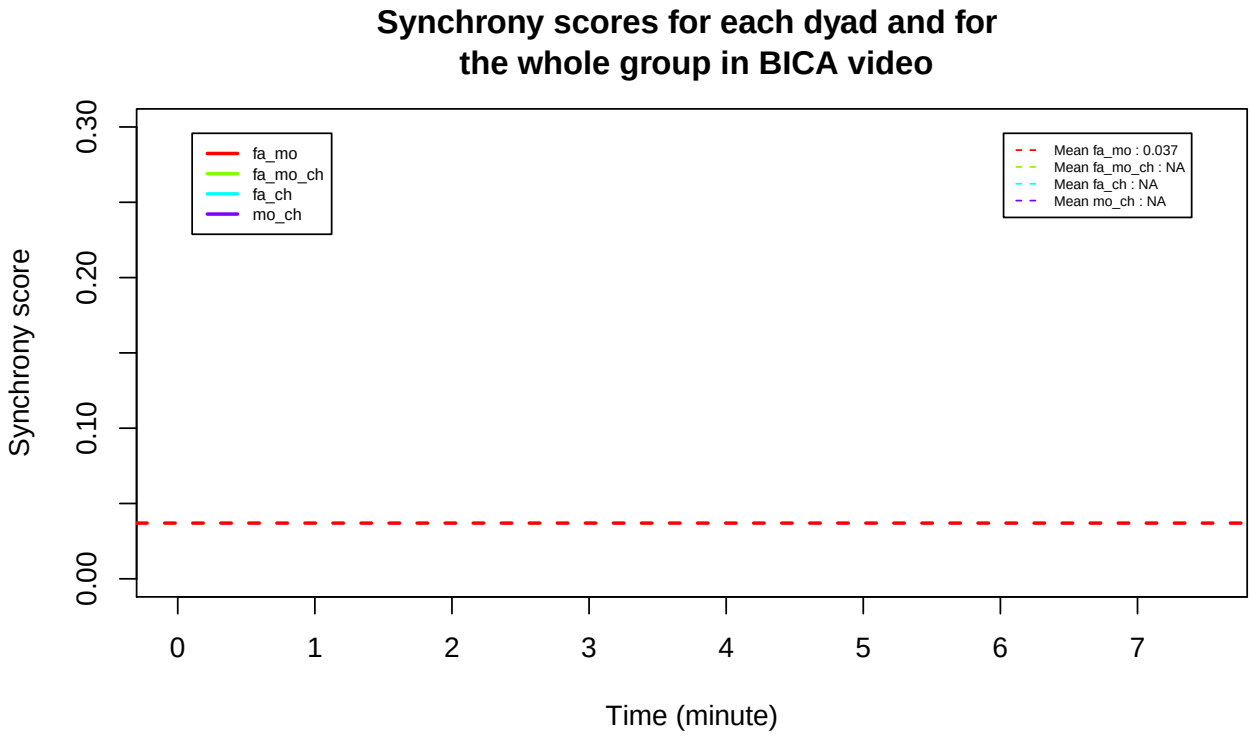

```
## Warning in mean.default(SSInoLog$SSI_fa_mo_ch, na.rm = TRUE): argument is
## not numeric or logical: returning NA

## Warning in mean.default(SSInoLog$SSI_fa_ch, na.rm = TRUE): argument is not
## numeric or logical: returning NA

## Warning in mean.default(SSInoLog$SSI_mo_ch, na.rm = TRUE): argument is not
## numeric or logical: returning NA

## Warning in mean.default(SSInoLog$SSI_fa_mo_ch, na.rm = TRUE): argument is
## not numeric or logical: returning NA

## Warning in mean.default(SSInoLog$SSI_fa_ch, na.rm = TRUE): argument is not
## numeric or logical: returning NA

## Warning in mean.default(SSInoLog$SSI_mo_ch, na.rm = TRUE): argument is not
## numeric or logical: returning NA
```



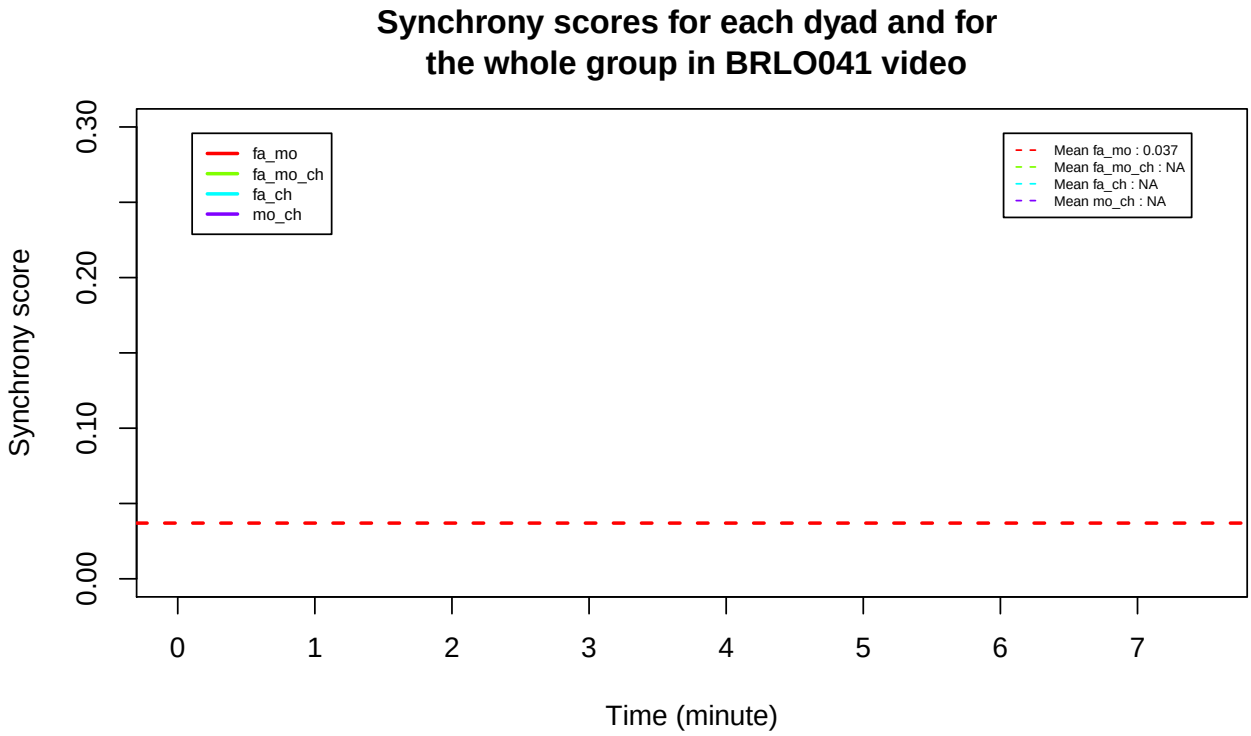

```
## Warning in mean.default(SSInoLog$SSI_fa_mo_ch, na.rm = TRUE): argument is
## not numeric or logical: returning NA

## Warning in mean.default(SSInoLog$SSI_fa_ch, na.rm = TRUE): argument is not
## numeric or logical: returning NA

## Warning in mean.default(SSInoLog$SSI_mo_ch, na.rm = TRUE): argument is not
## numeric or logical: returning NA

## Warning in mean.default(SSInoLog$SSI_fa_mo_ch, na.rm = TRUE): argument is
## not numeric or logical: returning NA

## Warning in mean.default(SSInoLog$SSI_fa_ch, na.rm = TRUE): argument is not
## numeric or logical: returning NA

## Warning in mean.default(SSInoLog$SSI_mo_ch, na.rm = TRUE): argument is not
## numeric or logical: returning NA
```



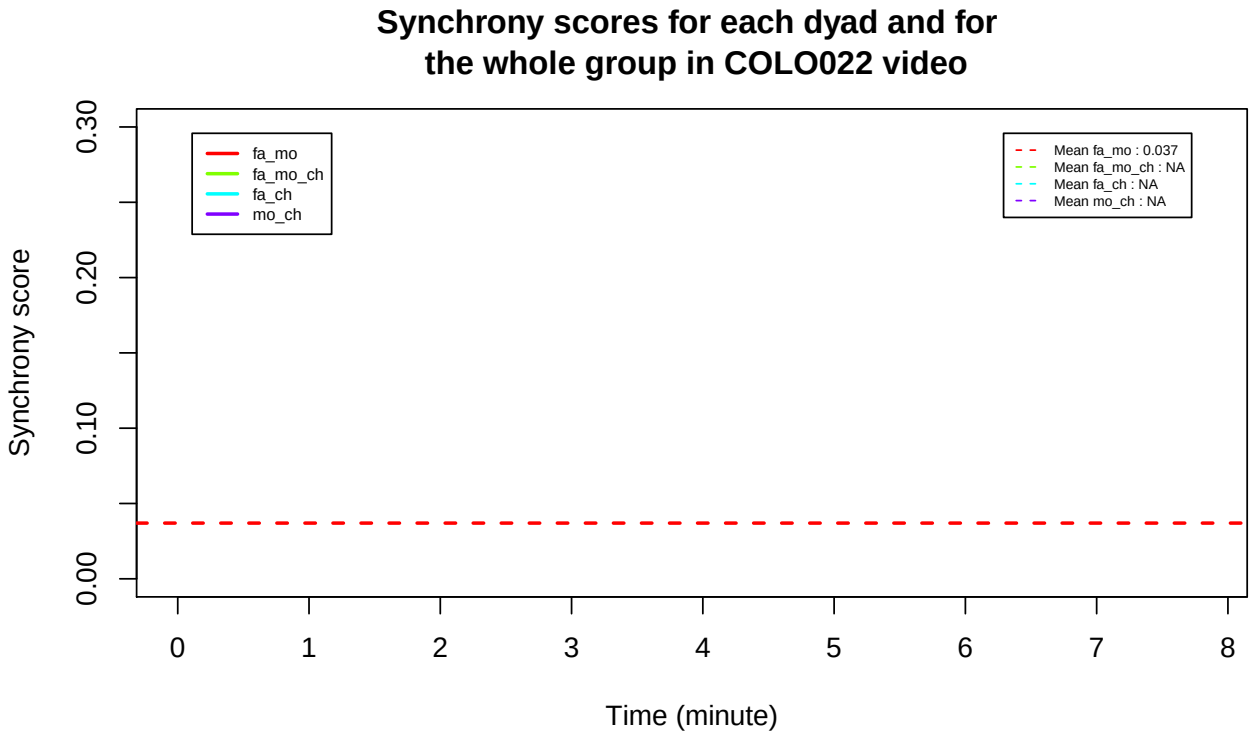

```
## Warning in mean.default(SSInoLog$SSI_fa_mo_ch, na.rm = TRUE): argument is
## not numeric or logical: returning NA

## Warning in mean.default(SSInoLog$SSI_fa_ch, na.rm = TRUE): argument is not
## numeric or logical: returning NA

## Warning in mean.default(SSInoLog$SSI_mo_ch, na.rm = TRUE): argument is not
## numeric or logical: returning NA

## Warning in mean.default(SSInoLog$SSI_fa_mo_ch, na.rm = TRUE): argument is
## not numeric or logical: returning NA

## Warning in mean.default(SSInoLog$SSI_fa_ch, na.rm = TRUE): argument is not
## numeric or logical: returning NA

## Warning in mean.default(SSInoLog$SSI_mo_ch, na.rm = TRUE): argument is not
## numeric or logical: returning NA
```



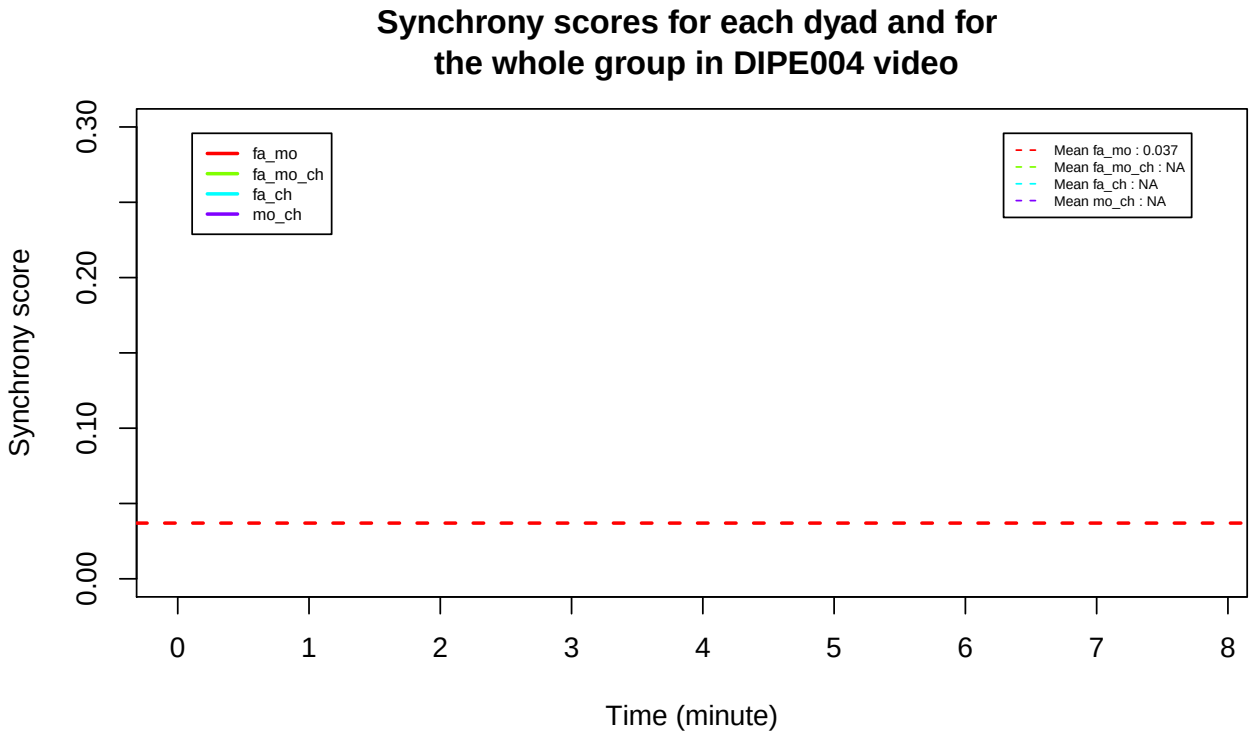

```
## Warning in mean.default(SSInoLog$SSI_fa_mo_ch, na.rm = TRUE): argument is
## not numeric or logical: returning NA

## Warning in mean.default(SSInoLog$SSI_fa_ch, na.rm = TRUE): argument is not
## numeric or logical: returning NA

## Warning in mean.default(SSInoLog$SSI_mo_ch, na.rm = TRUE): argument is not
## numeric or logical: returning NA

## Warning in mean.default(SSInoLog$SSI_fa_mo_ch, na.rm = TRUE): argument is
## not numeric or logical: returning NA

## Warning in mean.default(SSInoLog$SSI_fa_ch, na.rm = TRUE): argument is not
## numeric or logical: returning NA

## Warning in mean.default(SSInoLog$SSI_mo_ch, na.rm = TRUE): argument is not
## numeric or logical: returning NA
```



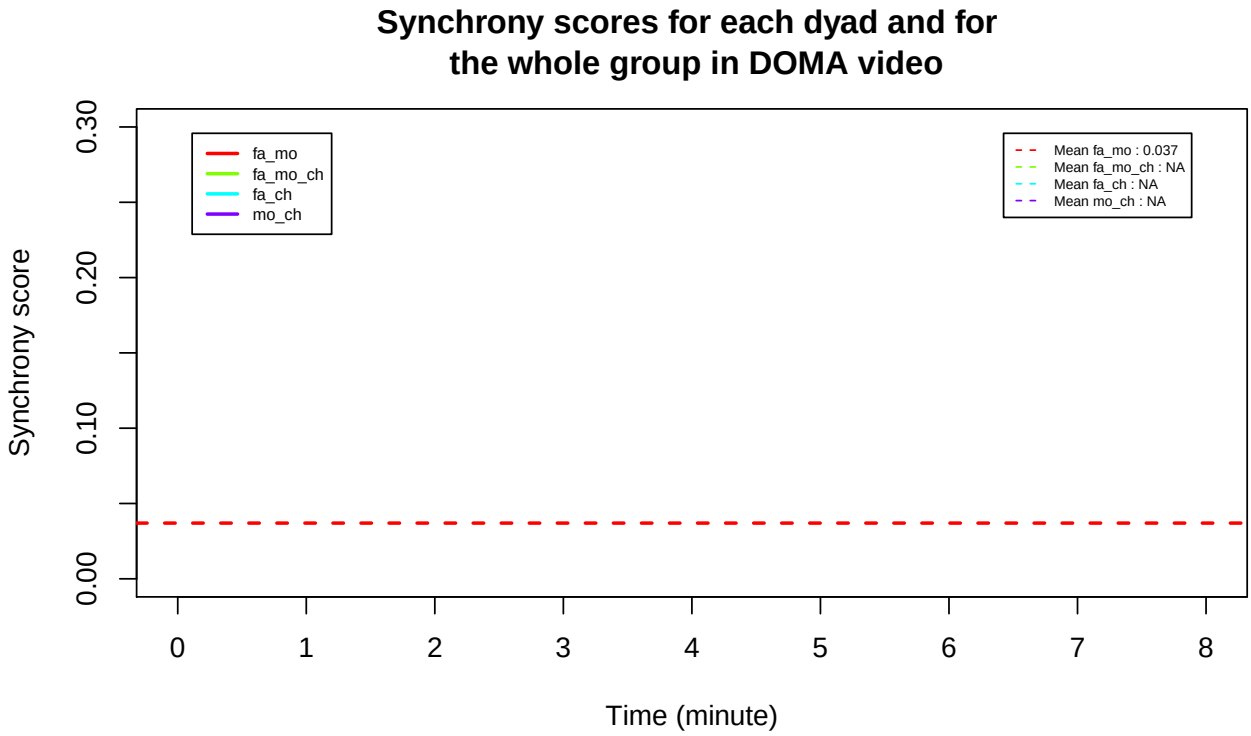

```
## Warning in mean.default(SSInoLog$SSI_fa_mo_ch, na.rm = TRUE): argument is
## not numeric or logical: returning NA

## Warning in mean.default(SSInoLog$SSI_fa_ch, na.rm = TRUE): argument is not
## numeric or logical: returning NA

## Warning in mean.default(SSInoLog$SSI_mo_ch, na.rm = TRUE): argument is not
## numeric or logical: returning NA

## Warning in mean.default(SSInoLog$SSI_fa_mo_ch, na.rm = TRUE): argument is
## not numeric or logical: returning NA

## Warning in mean.default(SSInoLog$SSI_fa_ch, na.rm = TRUE): argument is not
## numeric or logical: returning NA

## Warning in mean.default(SSInoLog$SSI_mo_ch, na.rm = TRUE): argument is not
## numeric or logical: returning NA
```



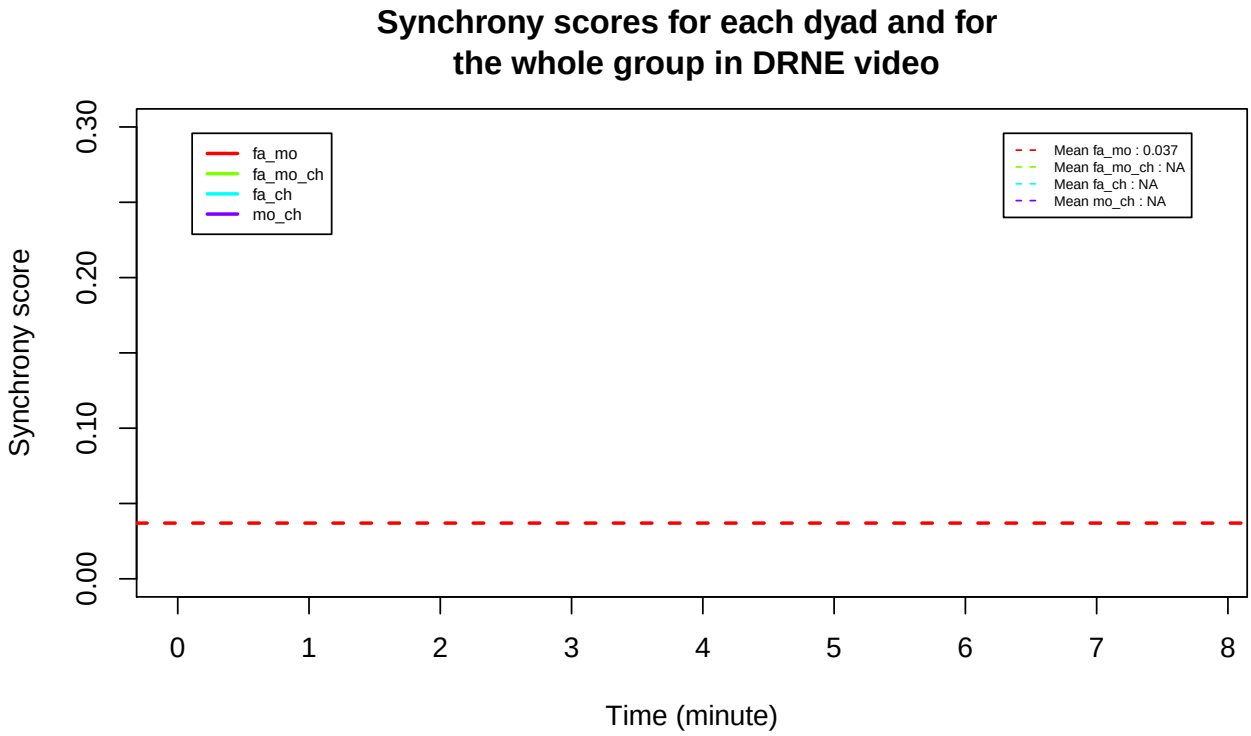

```
## Warning in mean.default(SSInoLog$SSI_fa_mo_ch, na.rm = TRUE): argument is
## not numeric or logical: returning NA

## Warning in mean.default(SSInoLog$SSI_fa_ch, na.rm = TRUE): argument is not
## numeric or logical: returning NA

## Warning in mean.default(SSInoLog$SSI_mo_ch, na.rm = TRUE): argument is not
## numeric or logical: returning NA

## Warning in mean.default(SSInoLog$SSI_fa_mo_ch, na.rm = TRUE): argument is
## not numeric or logical: returning NA

## Warning in mean.default(SSInoLog$SSI_fa_ch, na.rm = TRUE): argument is not
## numeric or logical: returning NA

## Warning in mean.default(SSInoLog$SSI_mo_ch, na.rm = TRUE): argument is not
## numeric or logical: returning NA
```



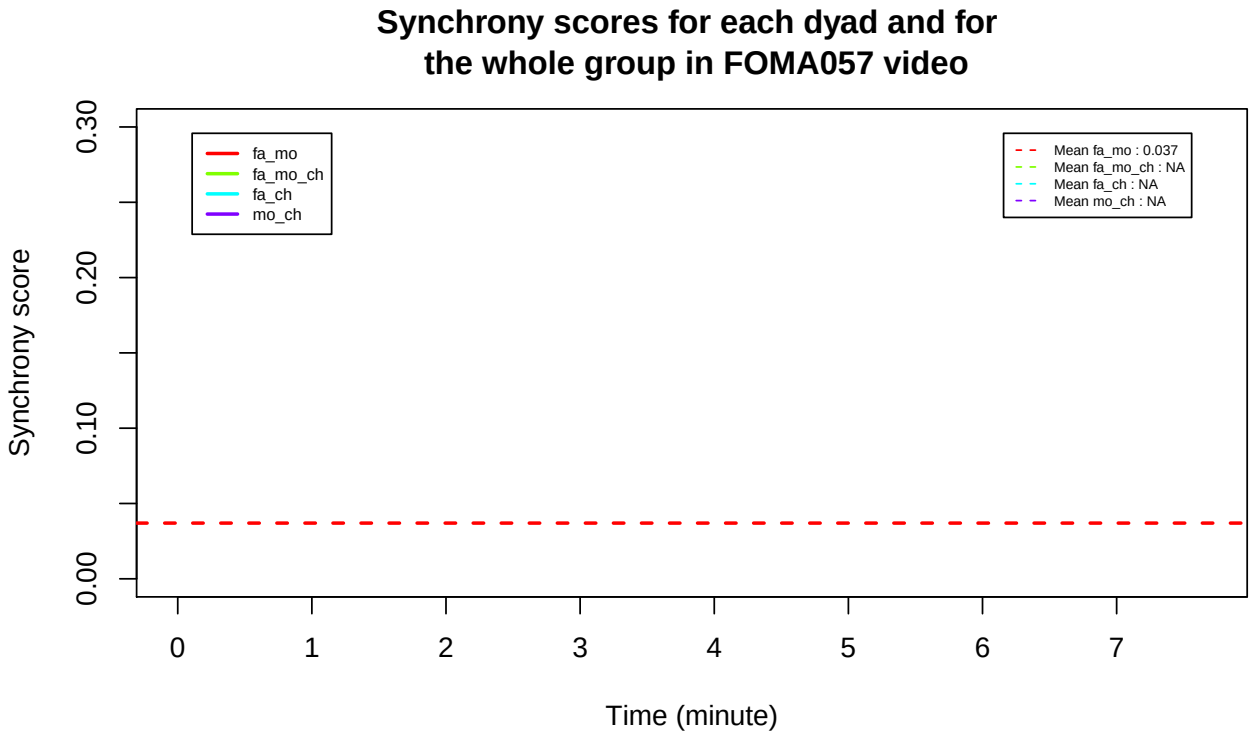

```
## Warning in mean.default(SSInoLog$SSI_fa_mo_ch, na.rm = TRUE): argument is
## not numeric or logical: returning NA

## Warning in mean.default(SSInoLog$SSI_fa_ch, na.rm = TRUE): argument is not
## numeric or logical: returning NA

## Warning in mean.default(SSInoLog$SSI_mo_ch, na.rm = TRUE): argument is not
## numeric or logical: returning NA

## Warning in mean.default(SSInoLog$SSI_fa_mo_ch, na.rm = TRUE): argument is
## not numeric or logical: returning NA

## Warning in mean.default(SSInoLog$SSI_fa_ch, na.rm = TRUE): argument is not
## numeric or logical: returning NA

## Warning in mean.default(SSInoLog$SSI_mo_ch, na.rm = TRUE): argument is not
## numeric or logical: returning NA
```



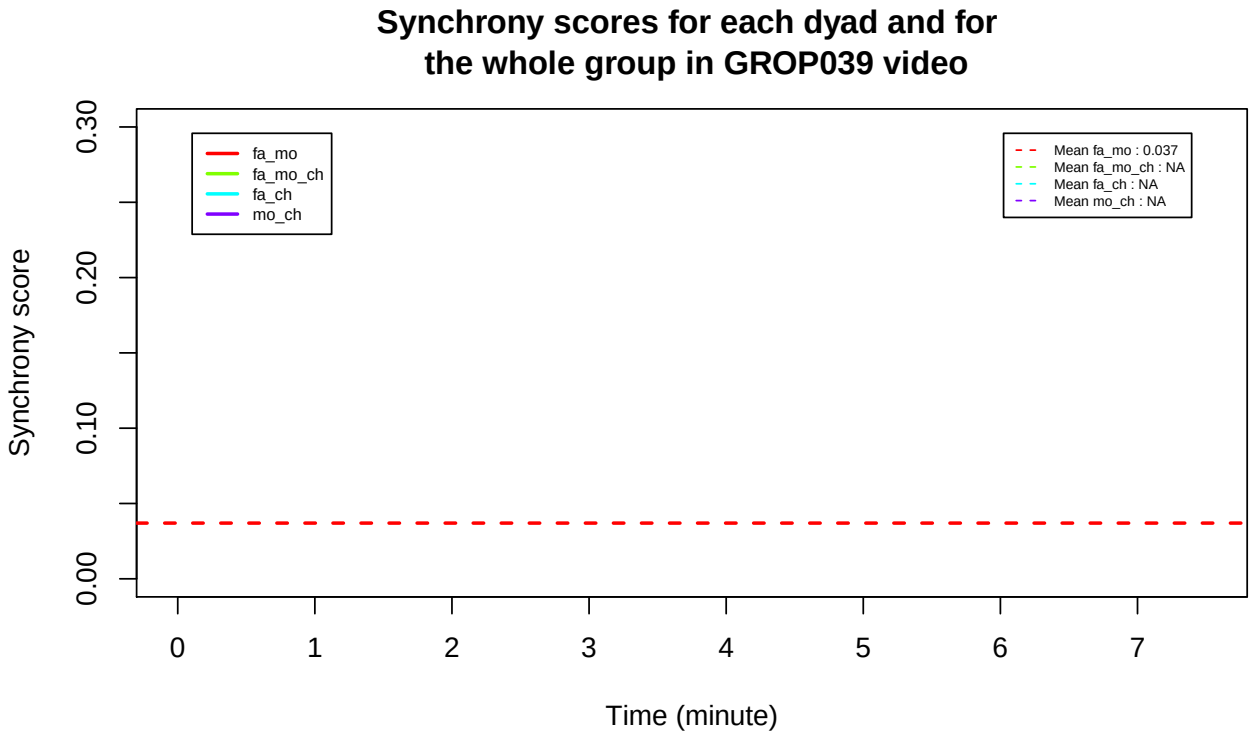

```
## Warning in mean.default(SSInoLog$SSI_fa_mo_ch, na.rm = TRUE): argument is
## not numeric or logical: returning NA

## Warning in mean.default(SSInoLog$SSI_fa_ch, na.rm = TRUE): argument is not
## numeric or logical: returning NA

## Warning in mean.default(SSInoLog$SSI_mo_ch, na.rm = TRUE): argument is not
## numeric or logical: returning NA

## Warning in mean.default(SSInoLog$SSI_fa_mo_ch, na.rm = TRUE): argument is
## not numeric or logical: returning NA

## Warning in mean.default(SSInoLog$SSI_fa_ch, na.rm = TRUE): argument is not
## numeric or logical: returning NA

## Warning in mean.default(SSInoLog$SSI_mo_ch, na.rm = TRUE): argument is not
## numeric or logical: returning NA
```



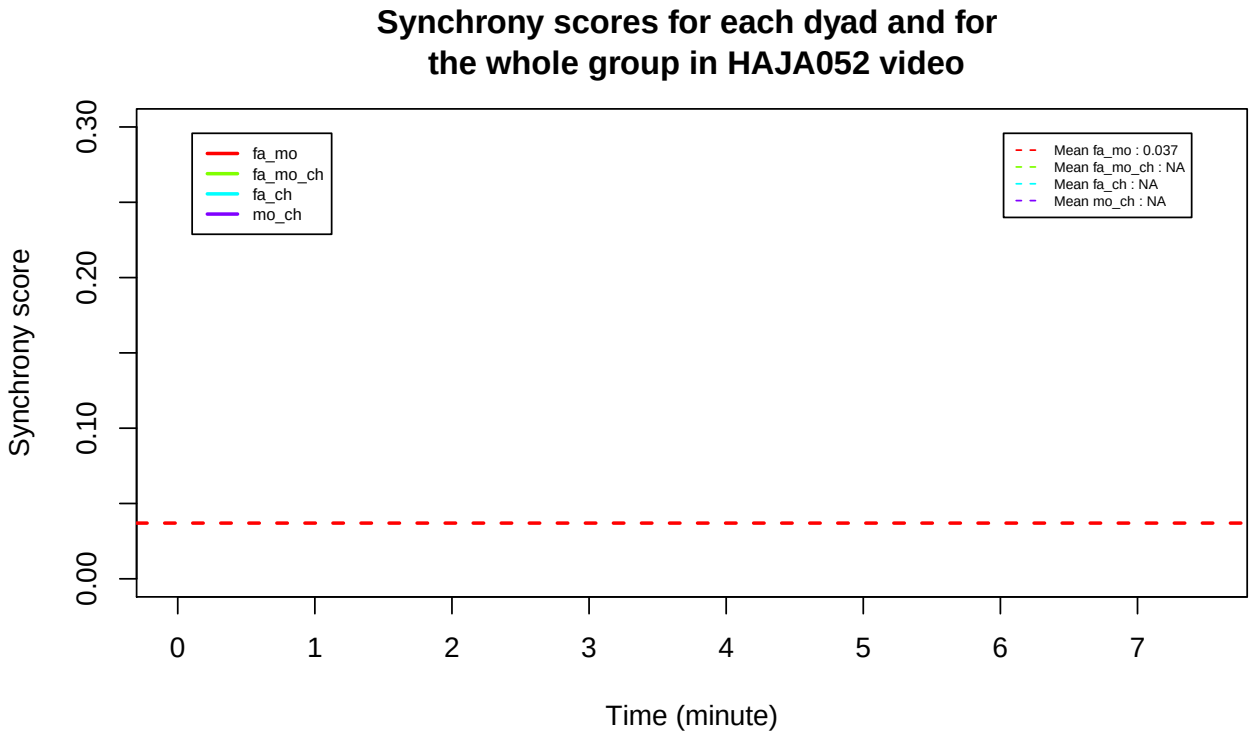

```
## Warning in mean.default(SSInoLog$SSI_fa_mo_ch, na.rm = TRUE): argument is
## not numeric or logical: returning NA

## Warning in mean.default(SSInoLog$SSI_fa_ch, na.rm = TRUE): argument is not
## numeric or logical: returning NA

## Warning in mean.default(SSInoLog$SSI_mo_ch, na.rm = TRUE): argument is not
## numeric or logical: returning NA

## Warning in mean.default(SSInoLog$SSI_fa_mo_ch, na.rm = TRUE): argument is
## not numeric or logical: returning NA

## Warning in mean.default(SSInoLog$SSI_fa_ch, na.rm = TRUE): argument is not
## numeric or logical: returning NA

## Warning in mean.default(SSInoLog$SSI_mo_ch, na.rm = TRUE): argument is not
## numeric or logical: returning NA
```



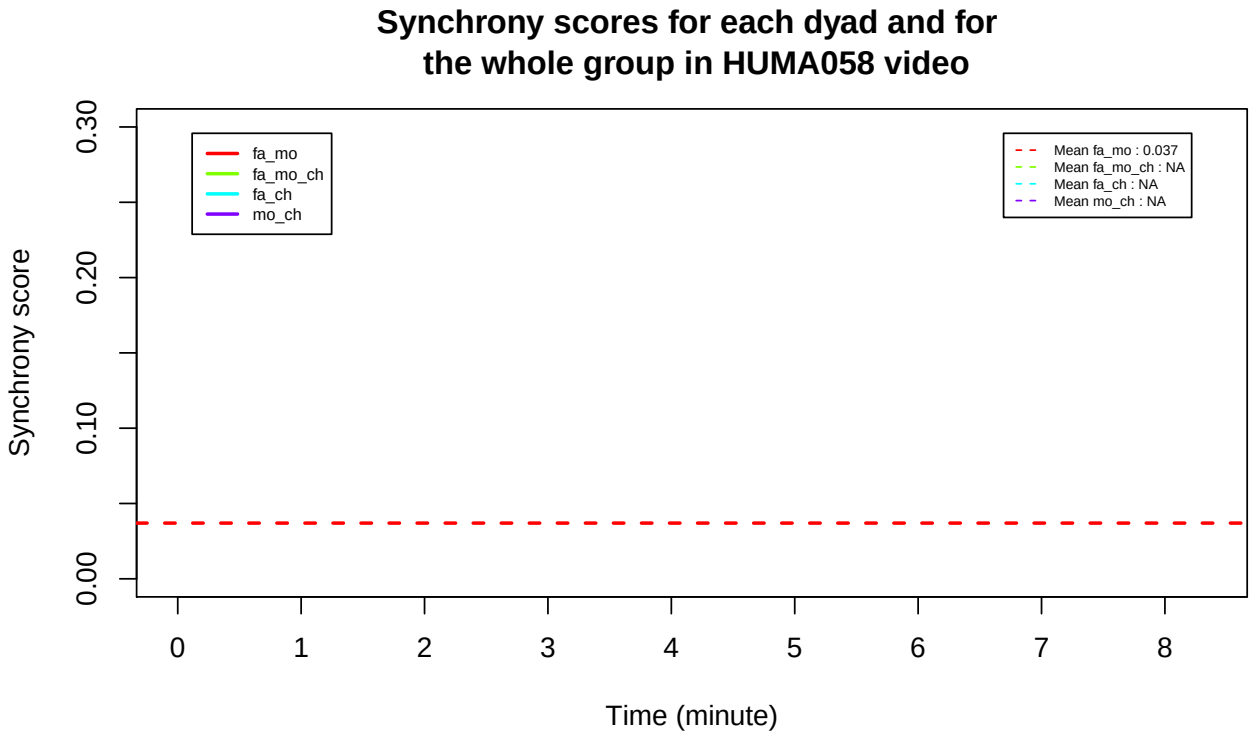

```
## Warning in mean.default(SSInoLog$SSI_fa_mo_ch, na.rm = TRUE): argument is
## not numeric or logical: returning NA

## Warning in mean.default(SSInoLog$SSI_fa_ch, na.rm = TRUE): argument is not
## numeric or logical: returning NA

## Warning in mean.default(SSInoLog$SSI_mo_ch, na.rm = TRUE): argument is not
## numeric or logical: returning NA

## Warning in mean.default(SSInoLog$SSI_fa_mo_ch, na.rm = TRUE): argument is
## not numeric or logical: returning NA

## Warning in mean.default(SSInoLog$SSI_fa_ch, na.rm = TRUE): argument is not
## numeric or logical: returning NA

## Warning in mean.default(SSInoLog$SSI_mo_ch, na.rm = TRUE): argument is not
## numeric or logical: returning NA
```



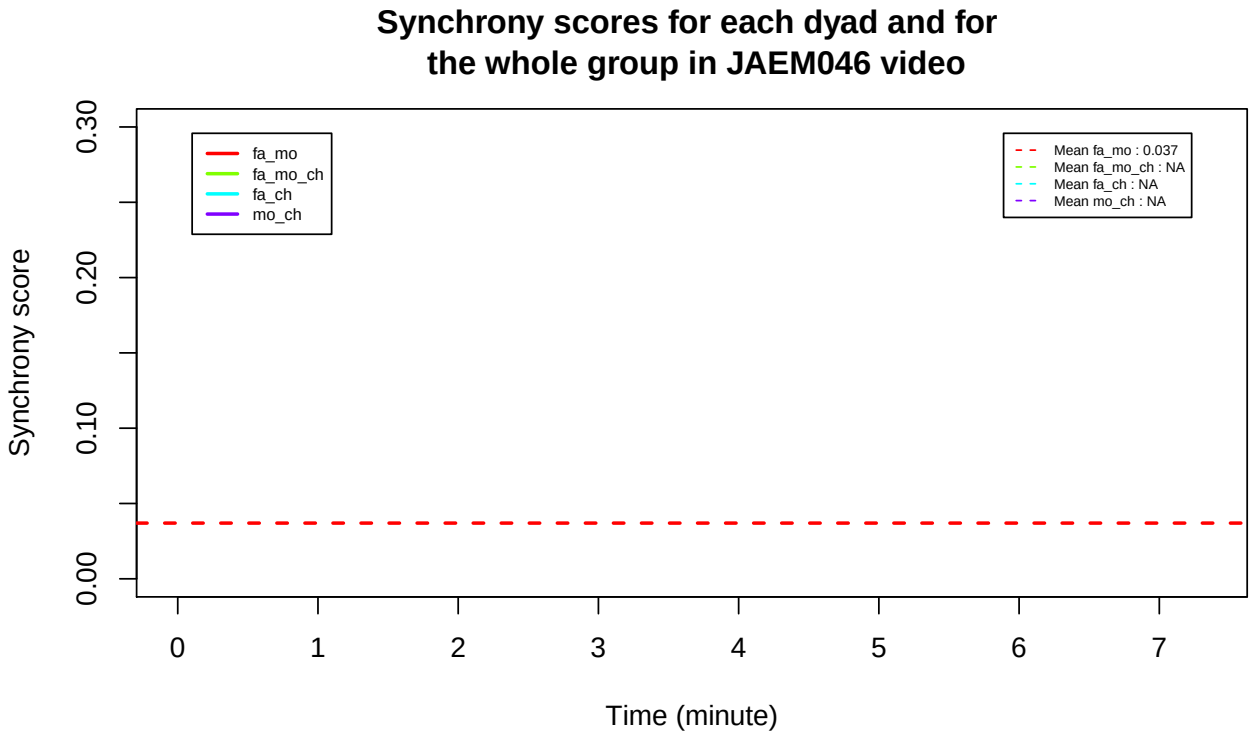

```
## Warning in mean.default(SSInoLog$SSI_fa_mo_ch, na.rm = TRUE): argument is
## not numeric or logical: returning NA

## Warning in mean.default(SSInoLog$SSI_fa_ch, na.rm = TRUE): argument is not
## numeric or logical: returning NA

## Warning in mean.default(SSInoLog$SSI_mo_ch, na.rm = TRUE): argument is not
## numeric or logical: returning NA

## Warning in mean.default(SSInoLog$SSI_fa_mo_ch, na.rm = TRUE): argument is
## not numeric or logical: returning NA

## Warning in mean.default(SSInoLog$SSI_fa_ch, na.rm = TRUE): argument is not
## numeric or logical: returning NA

## Warning in mean.default(SSInoLog$SSI_mo_ch, na.rm = TRUE): argument is not
## numeric or logical: returning NA
```



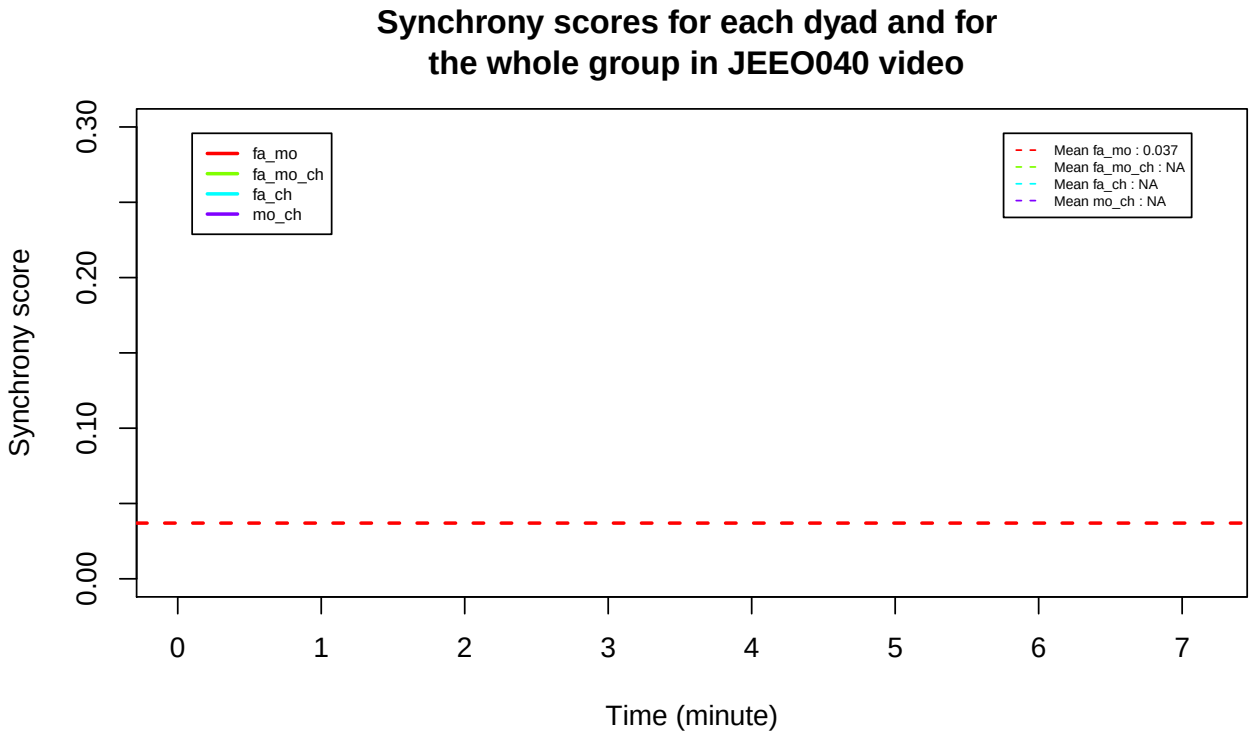

```
## Warning in mean.default(SSInoLog$SSI_fa_mo_ch, na.rm = TRUE): argument is
## not numeric or logical: returning NA

## Warning in mean.default(SSInoLog$SSI_fa_ch, na.rm = TRUE): argument is not
## numeric or logical: returning NA

## Warning in mean.default(SSInoLog$SSI_mo_ch, na.rm = TRUE): argument is not
## numeric or logical: returning NA

## Warning in mean.default(SSInoLog$SSI_fa_mo_ch, na.rm = TRUE): argument is
## not numeric or logical: returning NA

## Warning in mean.default(SSInoLog$SSI_fa_ch, na.rm = TRUE): argument is not
## numeric or logical: returning NA

## Warning in mean.default(SSInoLog$SSI_mo_ch, na.rm = TRUE): argument is not
## numeric or logical: returning NA
```



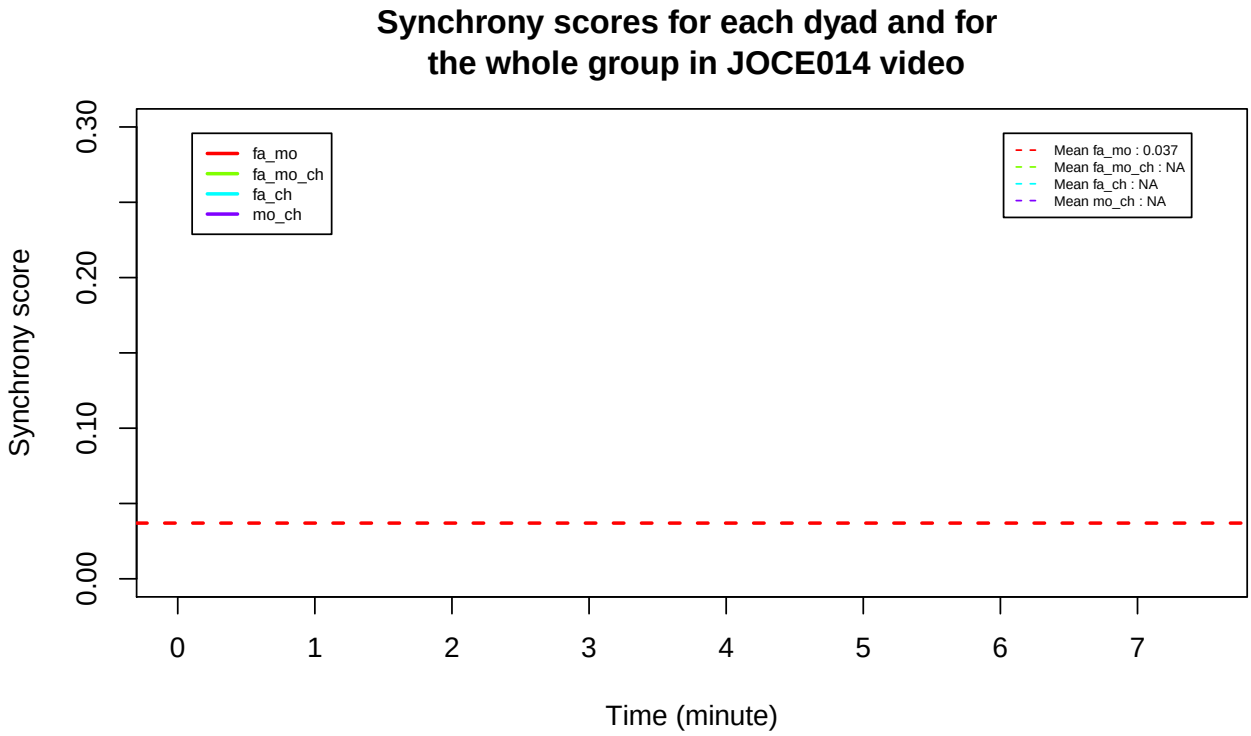

```
## Warning in mean.default(SSInoLog$SSI_fa_mo_ch, na.rm = TRUE): argument is
## not numeric or logical: returning NA

## Warning in mean.default(SSInoLog$SSI_fa_ch, na.rm = TRUE): argument is not
## numeric or logical: returning NA

## Warning in mean.default(SSInoLog$SSI_mo_ch, na.rm = TRUE): argument is not
## numeric or logical: returning NA

## Warning in mean.default(SSInoLog$SSI_fa_mo_ch, na.rm = TRUE): argument is
## not numeric or logical: returning NA

## Warning in mean.default(SSInoLog$SSI_fa_ch, na.rm = TRUE): argument is not
## numeric or logical: returning NA

## Warning in mean.default(SSInoLog$SSI_mo_ch, na.rm = TRUE): argument is not
## numeric or logical: returning NA
```



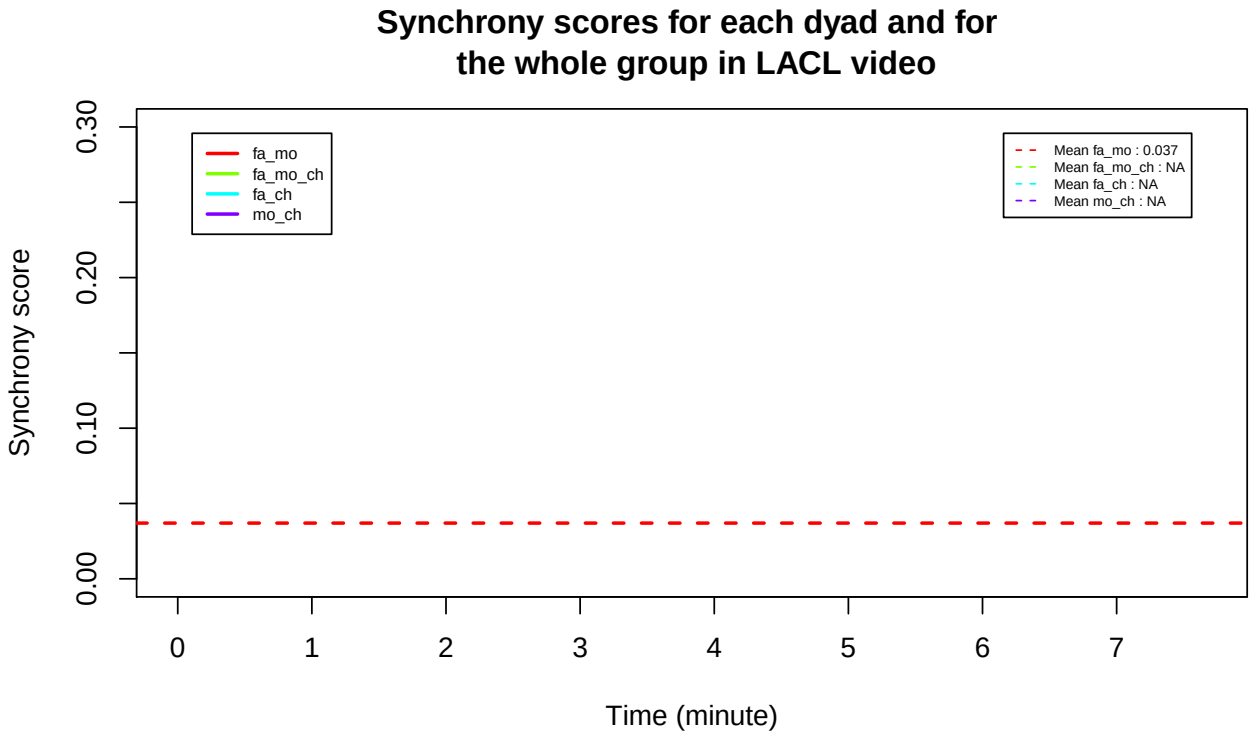

```
## Warning in mean.default(SSInoLog$SSI_fa_mo_ch, na.rm = TRUE): argument is
## not numeric or logical: returning NA

## Warning in mean.default(SSInoLog$SSI_fa_ch, na.rm = TRUE): argument is not
## numeric or logical: returning NA

## Warning in mean.default(SSInoLog$SSI_mo_ch, na.rm = TRUE): argument is not
## numeric or logical: returning NA

## Warning in mean.default(SSInoLog$SSI_fa_mo_ch, na.rm = TRUE): argument is
## not numeric or logical: returning NA

## Warning in mean.default(SSInoLog$SSI_fa_ch, na.rm = TRUE): argument is not
## numeric or logical: returning NA

## Warning in mean.default(SSInoLog$SSI_mo_ch, na.rm = TRUE): argument is not
## numeric or logical: returning NA
```



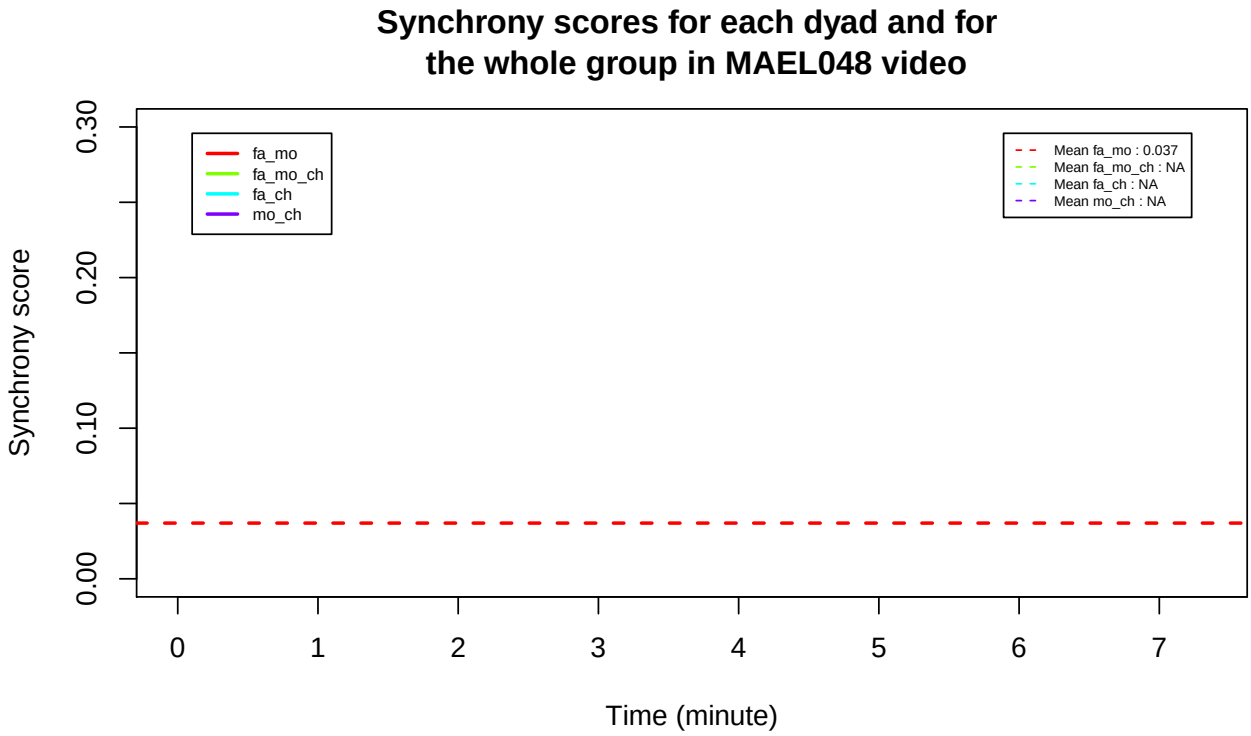

```
## Warning in mean.default(SSInoLog$SSI_fa_mo_ch, na.rm = TRUE): argument is
## not numeric or logical: returning NA

## Warning in mean.default(SSInoLog$SSI_fa_ch, na.rm = TRUE): argument is not
## numeric or logical: returning NA

## Warning in mean.default(SSInoLog$SSI_mo_ch, na.rm = TRUE): argument is not
## numeric or logical: returning NA

## Warning in mean.default(SSInoLog$SSI_fa_mo_ch, na.rm = TRUE): argument is
## not numeric or logical: returning NA

## Warning in mean.default(SSInoLog$SSI_fa_ch, na.rm = TRUE): argument is not
## numeric or logical: returning NA

## Warning in mean.default(SSInoLog$SSI_mo_ch, na.rm = TRUE): argument is not
## numeric or logical: returning NA
```



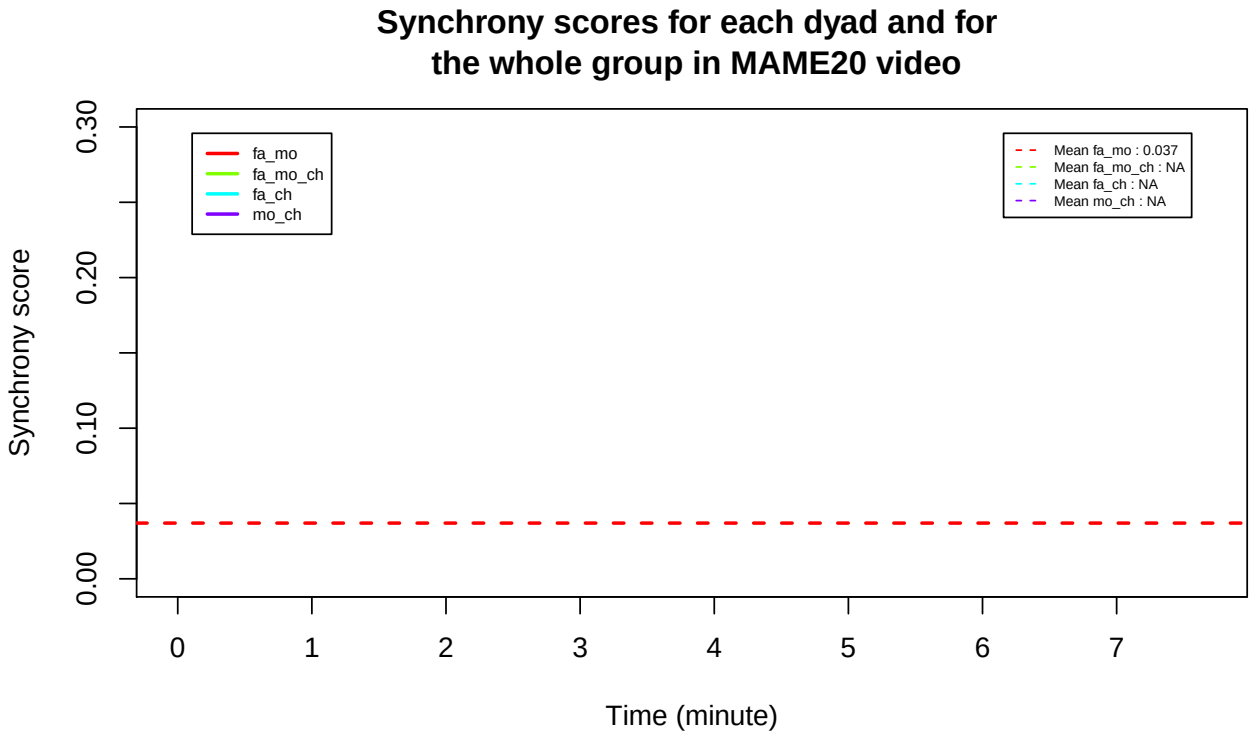

```
## Warning in mean.default(SSInoLog$SSI_fa_mo_ch, na.rm = TRUE): argument is
## not numeric or logical: returning NA

## Warning in mean.default(SSInoLog$SSI_fa_ch, na.rm = TRUE): argument is not
## numeric or logical: returning NA

## Warning in mean.default(SSInoLog$SSI_mo_ch, na.rm = TRUE): argument is not
## numeric or logical: returning NA

## Warning in mean.default(SSInoLog$SSI_fa_mo_ch, na.rm = TRUE): argument is
## not numeric or logical: returning NA

## Warning in mean.default(SSInoLog$SSI_fa_ch, na.rm = TRUE): argument is not
## numeric or logical: returning NA

## Warning in mean.default(SSInoLog$SSI_mo_ch, na.rm = TRUE): argument is not
## numeric or logical: returning NA
```



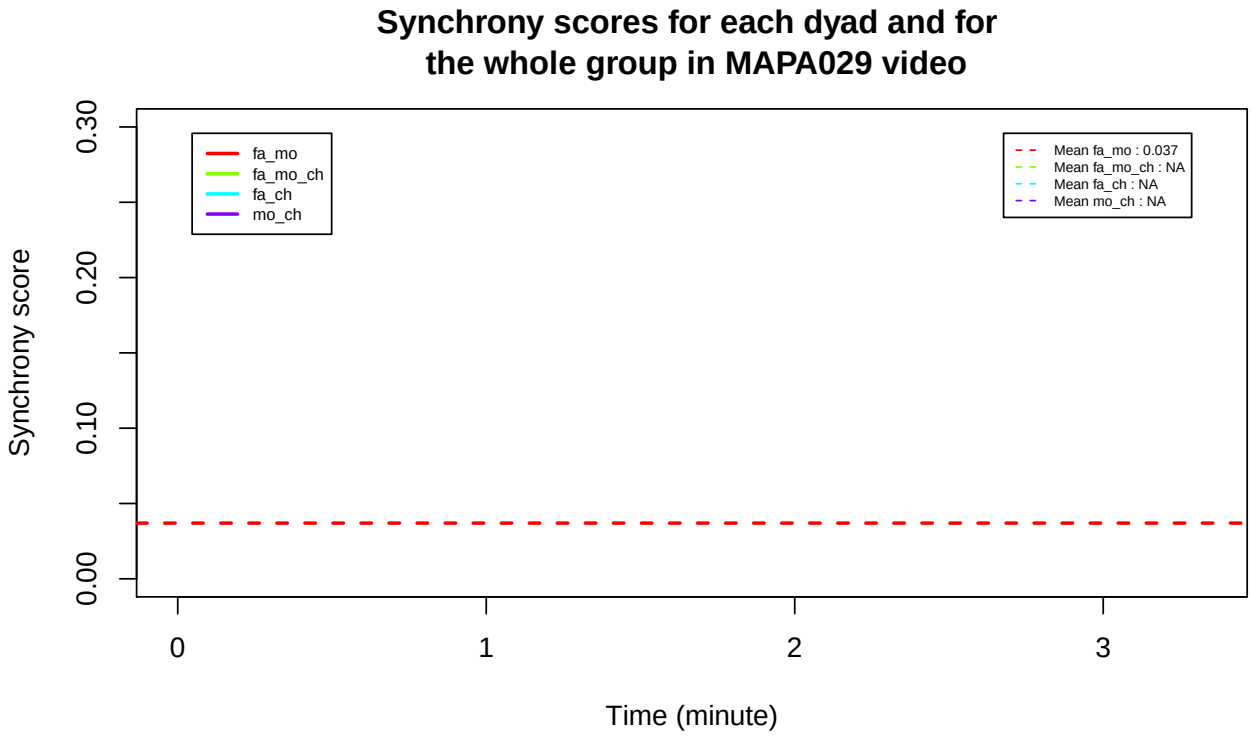

```
## Warning in mean.default(SSInoLog$SSI_fa_mo_ch, na.rm = TRUE): argument is
## not numeric or logical: returning NA

## Warning in mean.default(SSInoLog$SSI_fa_ch, na.rm = TRUE): argument is not
## numeric or logical: returning NA

## Warning in mean.default(SSInoLog$SSI_mo_ch, na.rm = TRUE): argument is not
## numeric or logical: returning NA

## Warning in mean.default(SSInoLog$SSI_fa_mo_ch, na.rm = TRUE): argument is
## not numeric or logical: returning NA

## Warning in mean.default(SSInoLog$SSI_fa_ch, na.rm = TRUE): argument is not
## numeric or logical: returning NA

## Warning in mean.default(SSInoLog$SSI_mo_ch, na.rm = TRUE): argument is not
## numeric or logical: returning NA
```



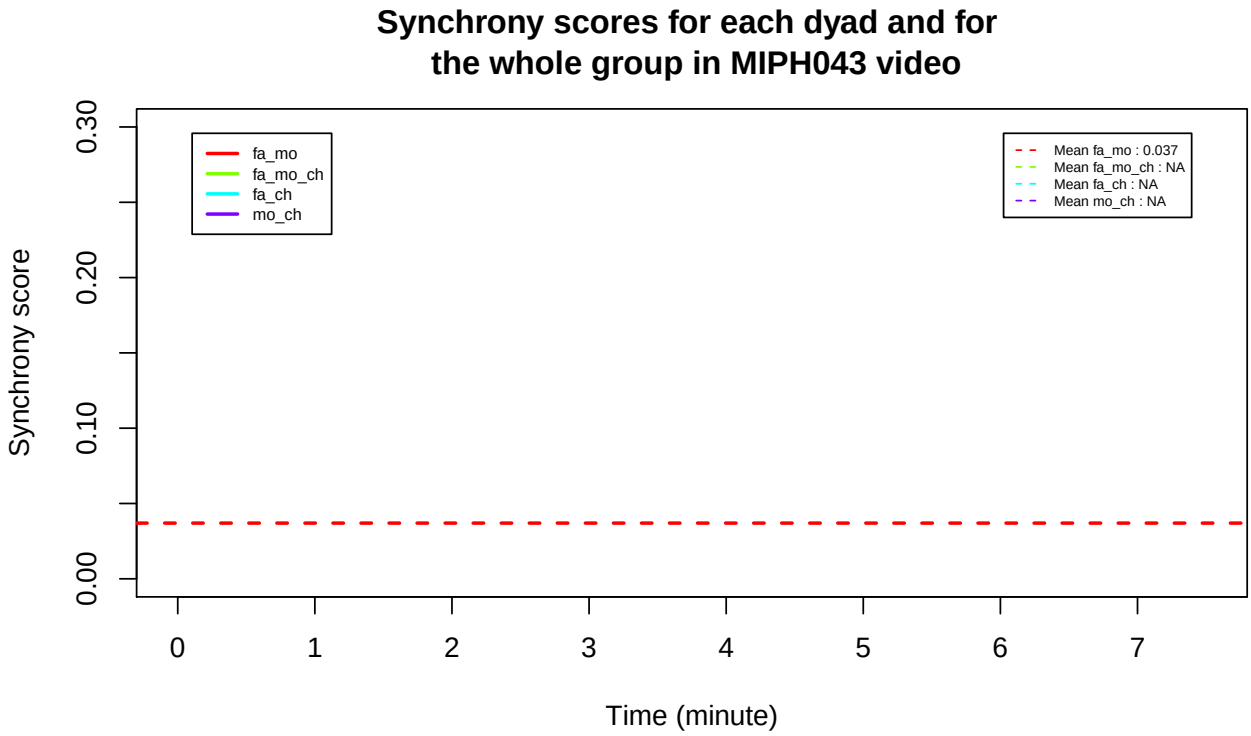

```
## Warning in mean.default(SSInoLog$SSI_fa_mo_ch, na.rm = TRUE): argument is
## not numeric or logical: returning NA

## Warning in mean.default(SSInoLog$SSI_fa_ch, na.rm = TRUE): argument is not
## numeric or logical: returning NA

## Warning in mean.default(SSInoLog$SSI_mo_ch, na.rm = TRUE): argument is not
## numeric or logical: returning NA

## Warning in mean.default(SSInoLog$SSI_fa_mo_ch, na.rm = TRUE): argument is
## not numeric or logical: returning NA

## Warning in mean.default(SSInoLog$SSI_fa_ch, na.rm = TRUE): argument is not
## numeric or logical: returning NA

## Warning in mean.default(SSInoLog$SSI_mo_ch, na.rm = TRUE): argument is not
## numeric or logical: returning NA
```



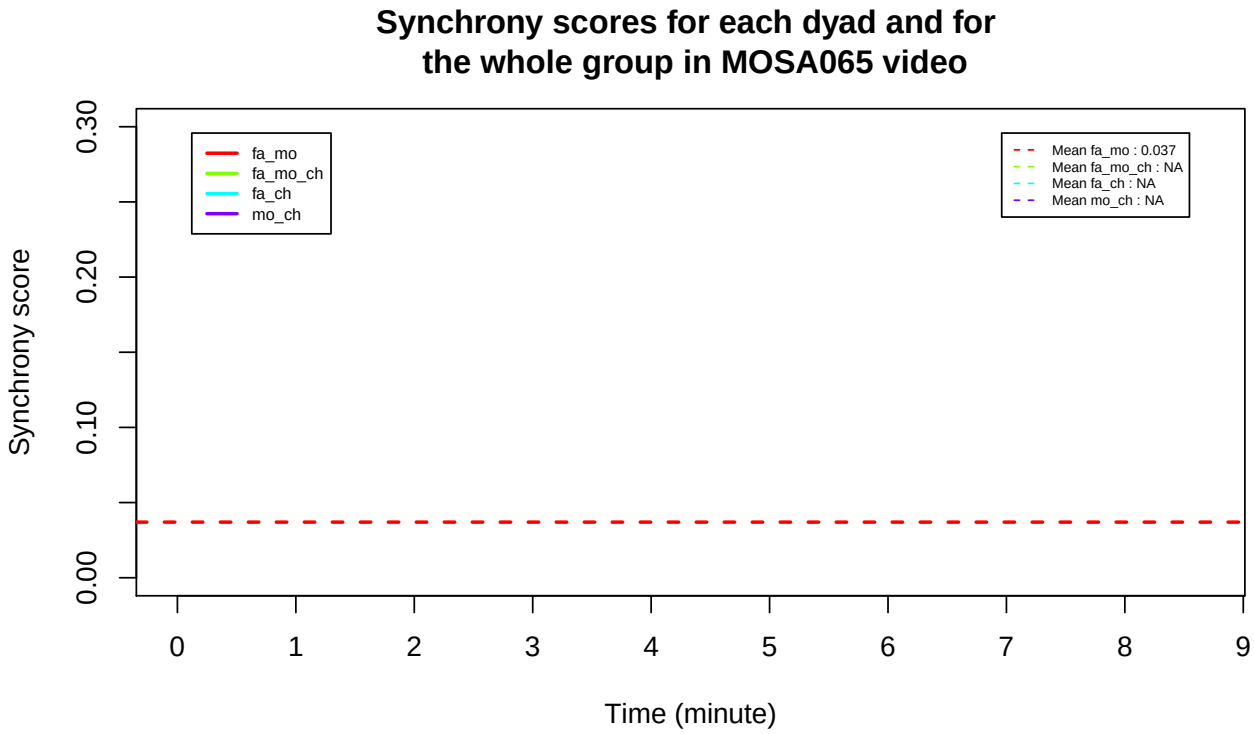

```
## Warning in mean.default(SSInoLog$SSI_fa_mo_ch, na.rm = TRUE): argument is
## not numeric or logical: returning NA

## Warning in mean.default(SSInoLog$SSI_fa_ch, na.rm = TRUE): argument is not
## numeric or logical: returning NA

## Warning in mean.default(SSInoLog$SSI_mo_ch, na.rm = TRUE): argument is not
## numeric or logical: returning NA

## Warning in mean.default(SSInoLog$SSI_fa_mo_ch, na.rm = TRUE): argument is
## not numeric or logical: returning NA

## Warning in mean.default(SSInoLog$SSI_fa_ch, na.rm = TRUE): argument is not
## numeric or logical: returning NA

## Warning in mean.default(SSInoLog$SSI_mo_ch, na.rm = TRUE): argument is not
## numeric or logical: returning NA
```



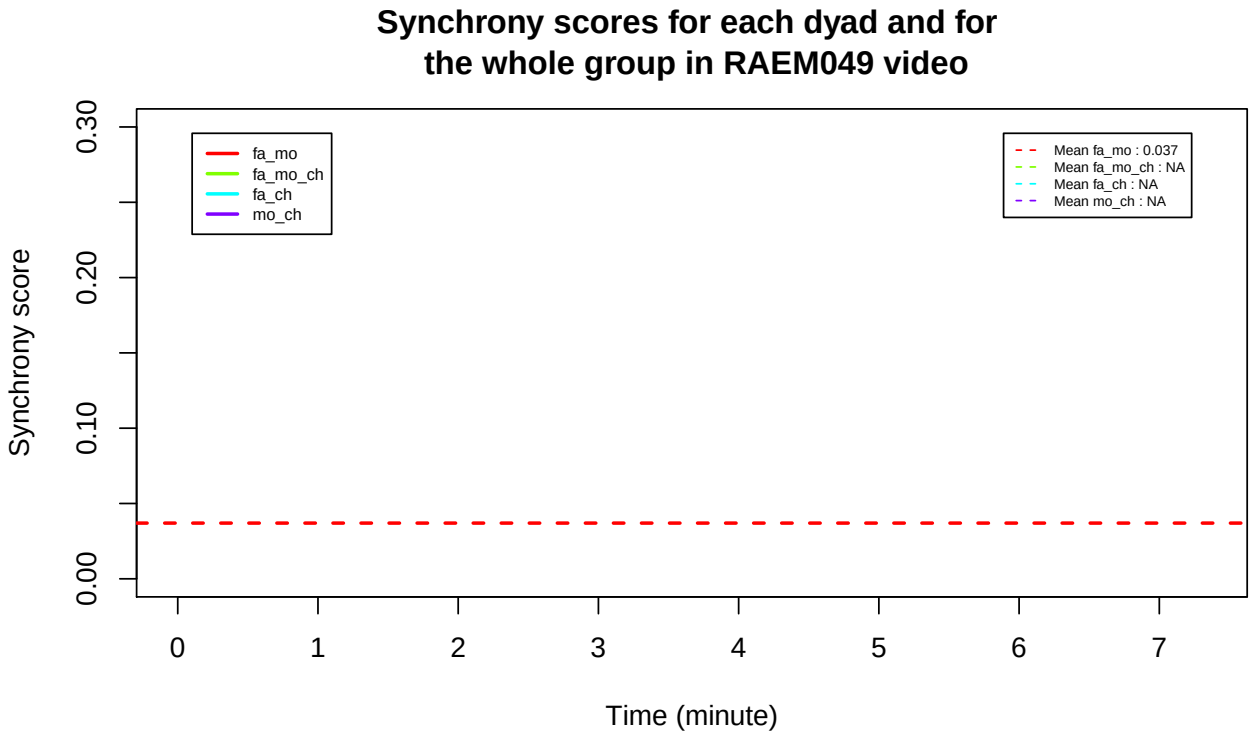

```
## Warning in mean.default(SSInoLog$SSI_fa_mo_ch, na.rm = TRUE): argument is
## not numeric or logical: returning NA

## Warning in mean.default(SSInoLog$SSI_fa_ch, na.rm = TRUE): argument is not
## numeric or logical: returning NA

## Warning in mean.default(SSInoLog$SSI_mo_ch, na.rm = TRUE): argument is not
## numeric or logical: returning NA

## Warning in mean.default(SSInoLog$SSI_fa_mo_ch, na.rm = TRUE): argument is
## not numeric or logical: returning NA

## Warning in mean.default(SSInoLog$SSI_fa_ch, na.rm = TRUE): argument is not
## numeric or logical: returning NA

## Warning in mean.default(SSInoLog$SSI_mo_ch, na.rm = TRUE): argument is not
## numeric or logical: returning NA
```



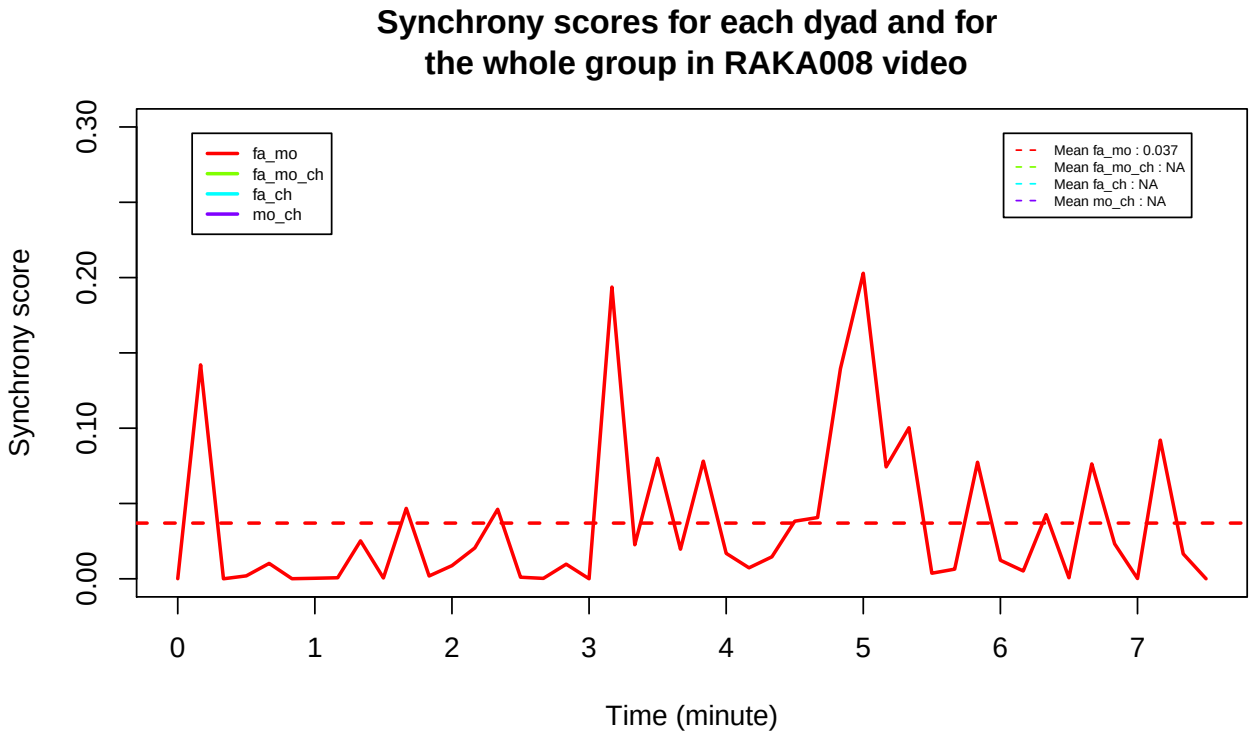

```
## Warning in mean.default(SSInoLog$SSI_fa_mo_ch, na.rm = TRUE): argument is
## not numeric or logical: returning NA

## Warning in mean.default(SSInoLog$SSI_fa_ch, na.rm = TRUE): argument is not
## numeric or logical: returning NA

## Warning in mean.default(SSInoLog$SSI_mo_ch, na.rm = TRUE): argument is not
## numeric or logical: returning NA

## Warning in mean.default(SSInoLog$SSI_fa_mo_ch, na.rm = TRUE): argument is
## not numeric or logical: returning NA

## Warning in mean.default(SSInoLog$SSI_fa_ch, na.rm = TRUE): argument is not
## numeric or logical: returning NA

## Warning in mean.default(SSInoLog$SSI_mo_ch, na.rm = TRUE): argument is not
## numeric or logical: returning NA
```



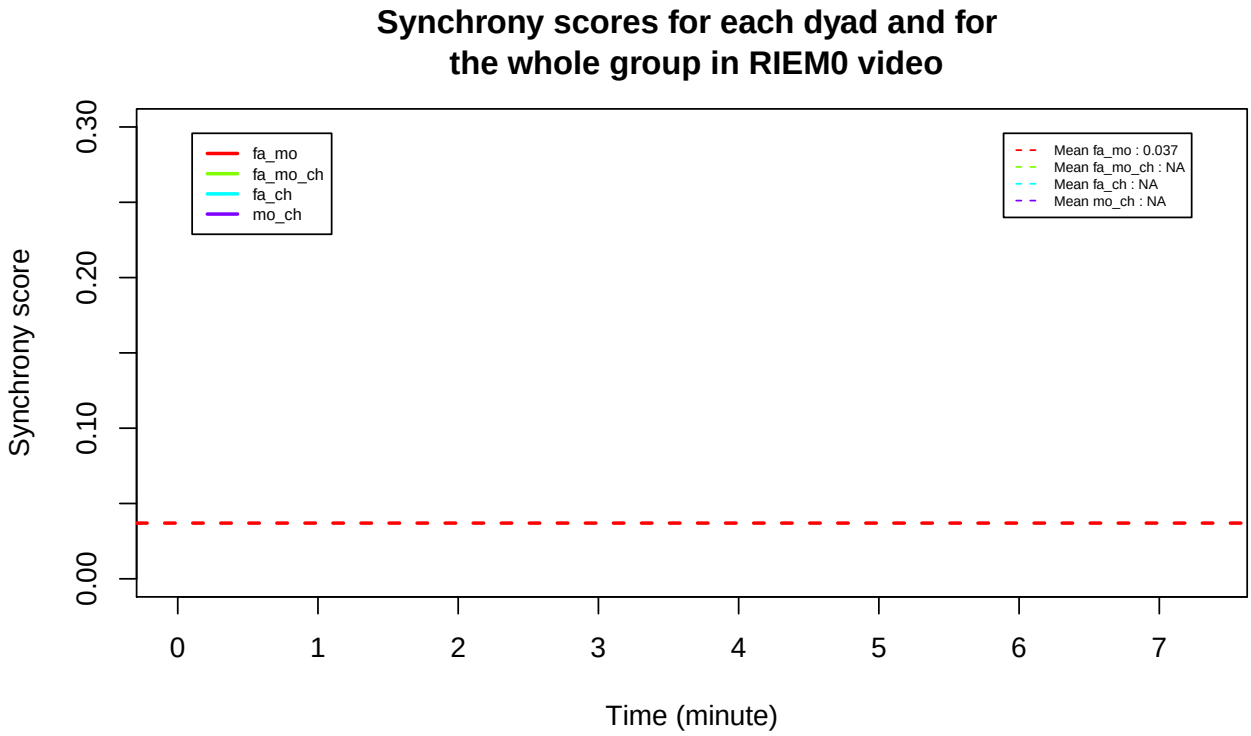

```
## Warning in mean.default(SSInoLog$SSI_fa_mo_ch, na.rm = TRUE): argument is
## not numeric or logical: returning NA

## Warning in mean.default(SSInoLog$SSI_fa_ch, na.rm = TRUE): argument is not
## numeric or logical: returning NA

## Warning in mean.default(SSInoLog$SSI_mo_ch, na.rm = TRUE): argument is not
## numeric or logical: returning NA

## Warning in mean.default(SSInoLog$SSI_fa_mo_ch, na.rm = TRUE): argument is
## not numeric or logical: returning NA

## Warning in mean.default(SSInoLog$SSI_fa_ch, na.rm = TRUE): argument is not
## numeric or logical: returning NA

## Warning in mean.default(SSInoLog$SSI_mo_ch, na.rm = TRUE): argument is not
## numeric or logical: returning NA
```



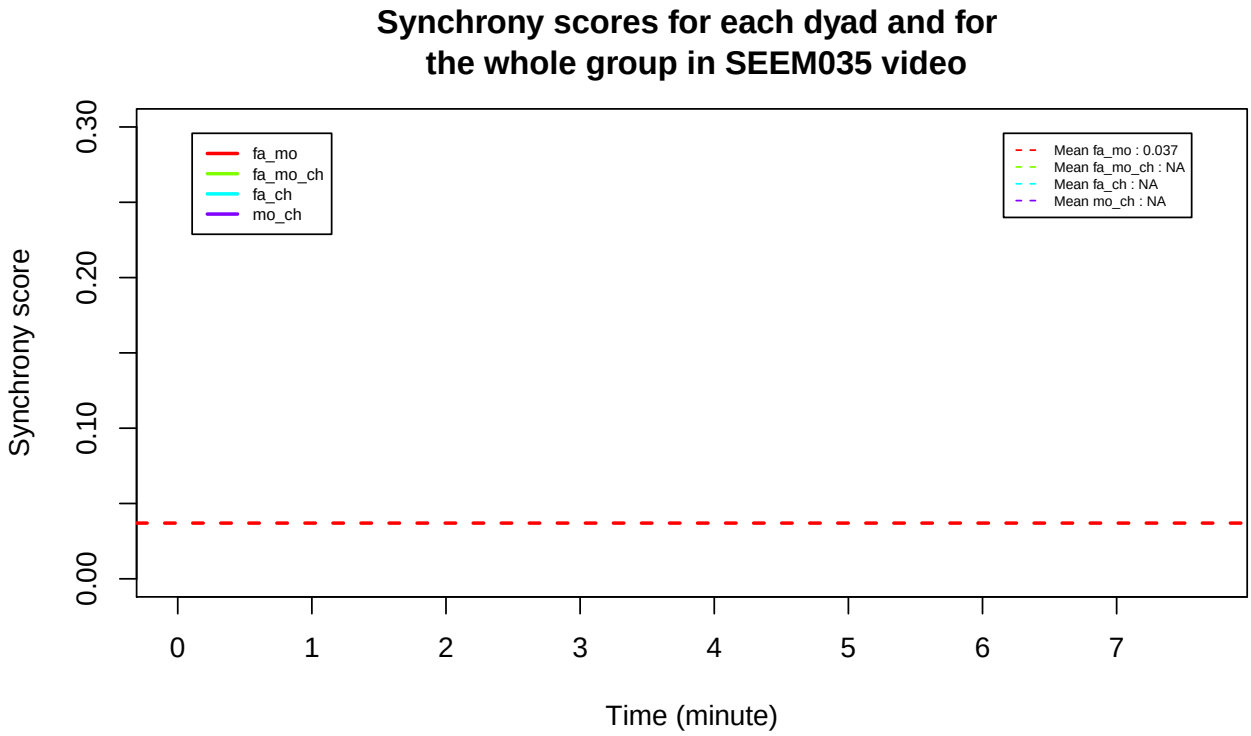

```
## Warning in mean.default(SSInoLog$SSI_fa_mo_ch, na.rm = TRUE): argument is
## not numeric or logical: returning NA

## Warning in mean.default(SSInoLog$SSI_fa_ch, na.rm = TRUE): argument is not
## numeric or logical: returning NA

## Warning in mean.default(SSInoLog$SSI_mo_ch, na.rm = TRUE): argument is not
## numeric or logical: returning NA

## Warning in mean.default(SSInoLog$SSI_fa_mo_ch, na.rm = TRUE): argument is
## not numeric or logical: returning NA

## Warning in mean.default(SSInoLog$SSI_fa_ch, na.rm = TRUE): argument is not
## numeric or logical: returning NA

## Warning in mean.default(SSInoLog$SSI_mo_ch, na.rm = TRUE): argument is not
## numeric or logical: returning NA
```



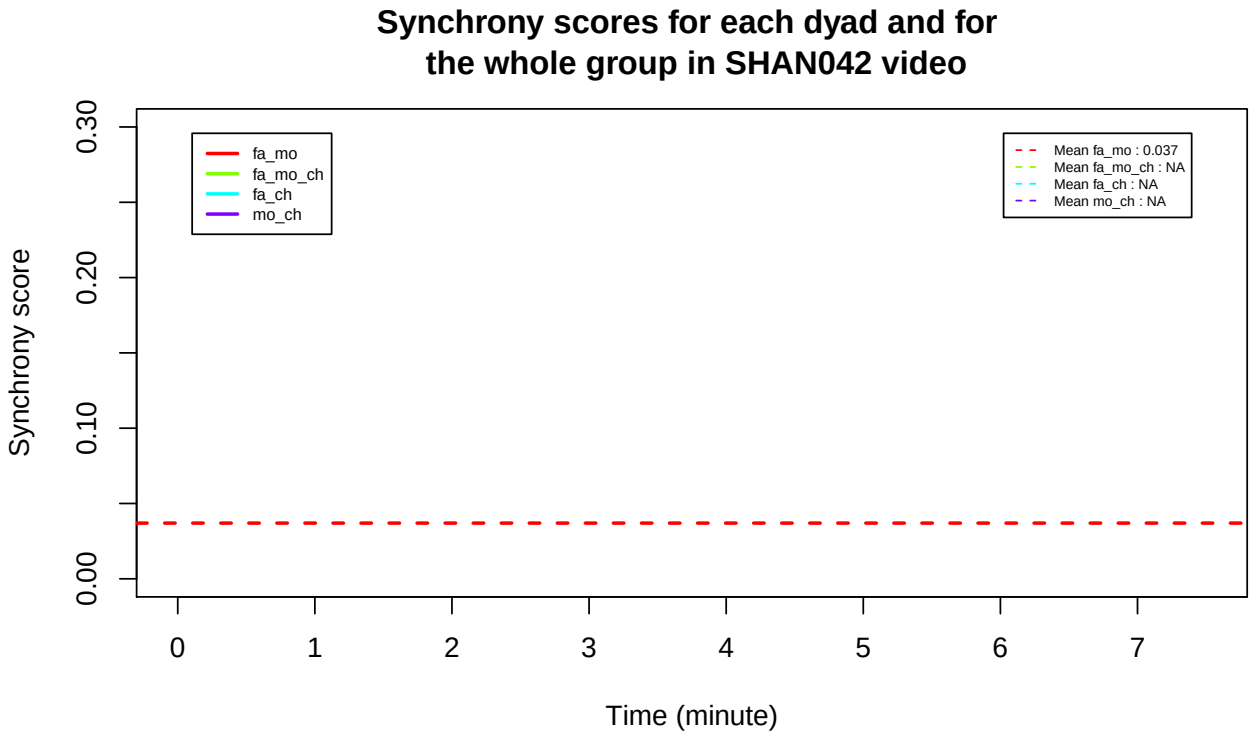

```
## Warning in mean.default(SSInoLog$SSI_fa_mo_ch, na.rm = TRUE): argument is
## not numeric or logical: returning NA

## Warning in mean.default(SSInoLog$SSI_fa_ch, na.rm = TRUE): argument is not
## numeric or logical: returning NA

## Warning in mean.default(SSInoLog$SSI_mo_ch, na.rm = TRUE): argument is not
## numeric or logical: returning NA

## Warning in mean.default(SSInoLog$SSI_fa_mo_ch, na.rm = TRUE): argument is
## not numeric or logical: returning NA

## Warning in mean.default(SSInoLog$SSI_fa_ch, na.rm = TRUE): argument is not
## numeric or logical: returning NA

## Warning in mean.default(SSInoLog$SSI_mo_ch, na.rm = TRUE): argument is not
## numeric or logical: returning NA
```



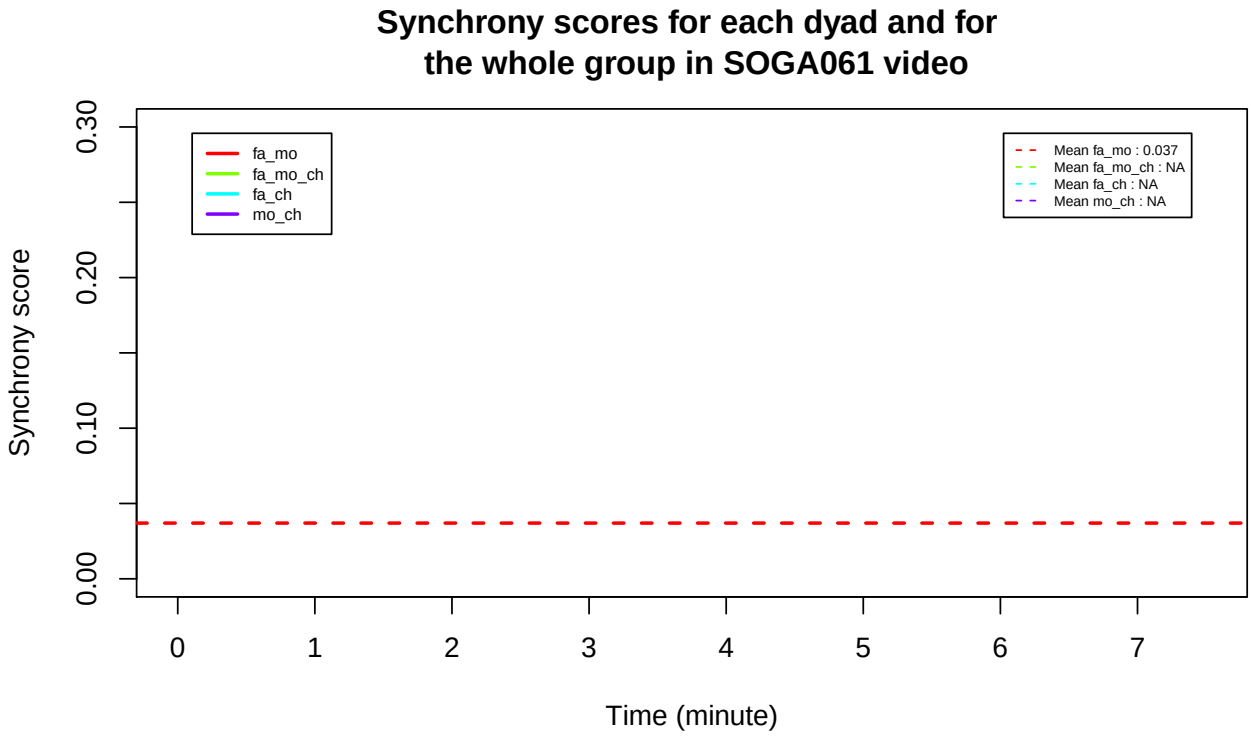

```
## Warning in mean.default(SSInoLog$SSI_fa_mo_ch, na.rm = TRUE): argument is
## not numeric or logical: returning NA

## Warning in mean.default(SSInoLog$SSI_fa_ch, na.rm = TRUE): argument is not
## numeric or logical: returning NA

## Warning in mean.default(SSInoLog$SSI_mo_ch, na.rm = TRUE): argument is not
## numeric or logical: returning NA

## Warning in mean.default(SSInoLog$SSI_fa_mo_ch, na.rm = TRUE): argument is
## not numeric or logical: returning NA

## Warning in mean.default(SSInoLog$SSI_fa_ch, na.rm = TRUE): argument is not
## numeric or logical: returning NA

## Warning in mean.default(SSInoLog$SSI_mo_ch, na.rm = TRUE): argument is not
## numeric or logical: returning NA
```



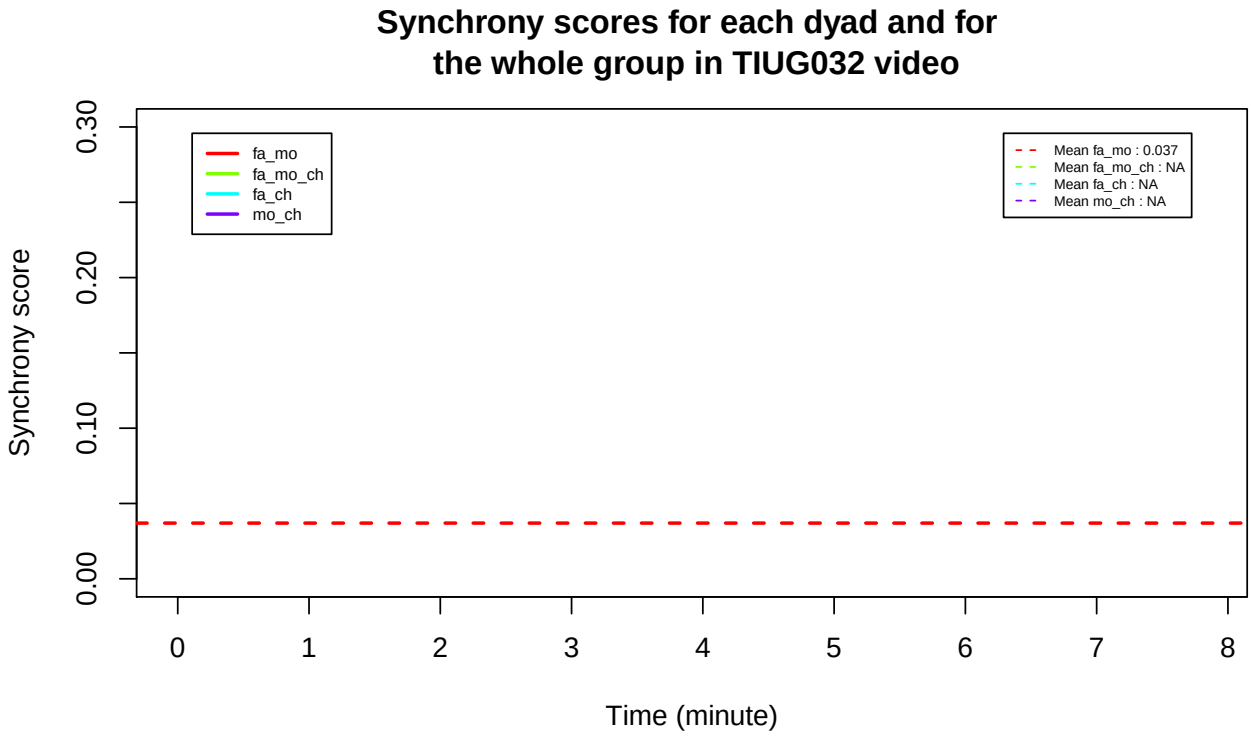

```
## Warning in mean.default(SSInoLog$SSI_fa_mo_ch, na.rm = TRUE): argument is
## not numeric or logical: returning NA

## Warning in mean.default(SSInoLog$SSI_fa_ch, na.rm = TRUE): argument is not
## numeric or logical: returning NA

## Warning in mean.default(SSInoLog$SSI_mo_ch, na.rm = TRUE): argument is not
## numeric or logical: returning NA

## Warning in mean.default(SSInoLog$SSI_fa_mo_ch, na.rm = TRUE): argument is
## not numeric or logical: returning NA

## Warning in mean.default(SSInoLog$SSI_fa_ch, na.rm = TRUE): argument is not
## numeric or logical: returning NA

## Warning in mean.default(SSInoLog$SSI_mo_ch, na.rm = TRUE): argument is not
## numeric or logical: returning NA
```



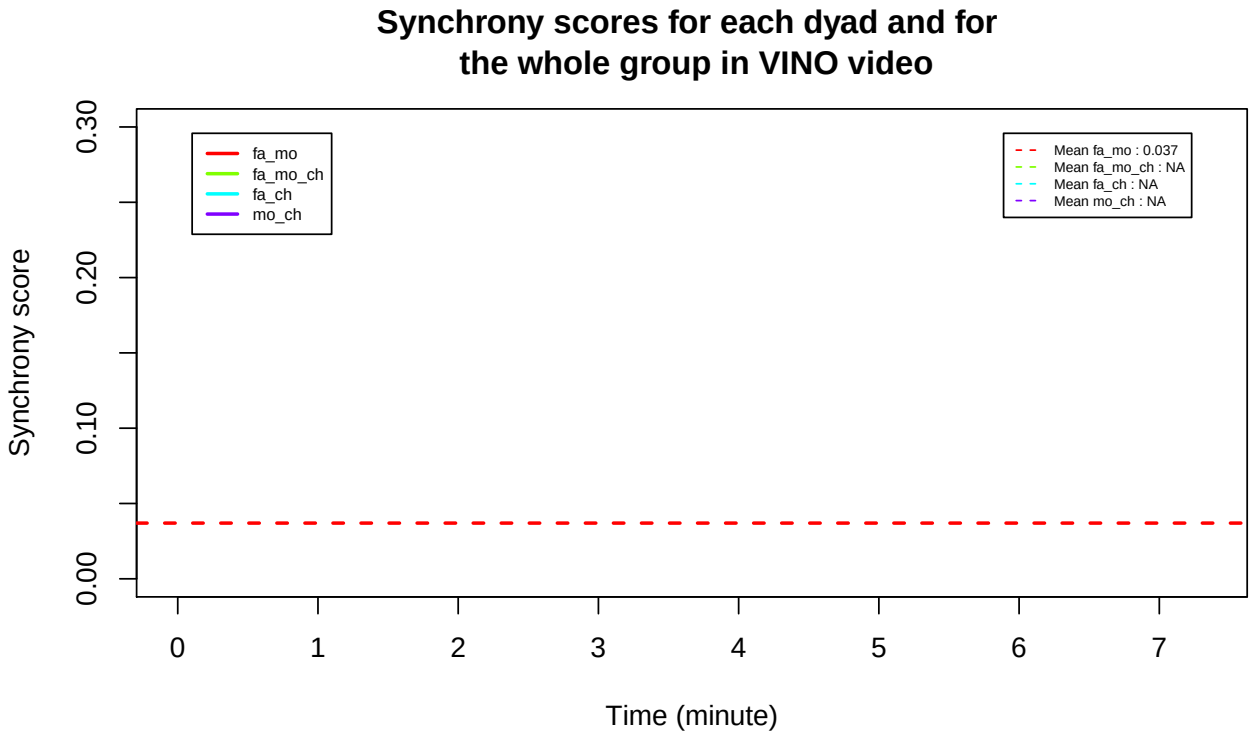

```
## Warning in mean.default(SSInoLog$SSI_fa_mo_ch, na.rm = TRUE): argument is
## not numeric or logical: returning NA

## Warning in mean.default(SSInoLog$SSI_fa_ch, na.rm = TRUE): argument is not
## numeric or logical: returning NA

## Warning in mean.default(SSInoLog$SSI_mo_ch, na.rm = TRUE): argument is not
## numeric or logical: returning NA

## Warning in mean.default(SSInoLog$SSI_fa_mo_ch, na.rm = TRUE): argument is
## not numeric or logical: returning NA

## Warning in mean.default(SSInoLog$SSI_fa_ch, na.rm = TRUE): argument is not
## numeric or logical: returning NA

## Warning in mean.default(SSInoLog$SSI_mo_ch, na.rm = TRUE): argument is not
## numeric or logical: returning NA
```




```
## Warning in mean.default(SSInoLog$SSI_fa_mo_ch, na.rm = TRUE): argument is
## not numeric or logical: returning NA

## Warning in mean.default(SSInoLog$SSI_fa_ch, na.rm = TRUE): argument is not
## numeric or logical: returning NA

## Warning in mean.default(SSInoLog$SSI_mo_ch, na.rm = TRUE): argument is not
## numeric or logical: returning NA

## Warning in mean.default(SSInoLog$SSI_fa_mo_ch, na.rm = TRUE): argument is
## not numeric or logical: returning NA

## Warning in mean.default(SSInoLog$SSI_fa_ch, na.rm = TRUE): argument is not
## numeric or logical: returning NA

## Warning in mean.default(SSInoLog$SSI_mo_ch, na.rm = TRUE): argument is not
## numeric or logical: returning NA
```




```
## Warning in mean.default(SSInoLog$SSI_fa_mo_ch, na.rm = TRUE): argument is
## not numeric or logical: returning NA

## Warning in mean.default(SSInoLog$SSI_fa_ch, na.rm = TRUE): argument is not
## numeric or logical: returning NA

## Warning in mean.default(SSInoLog$SSI_mo_ch, na.rm = TRUE): argument is not
## numeric or logical: returning NA

## Warning in mean.default(SSInoLog$SSI_fa_mo_ch, na.rm = TRUE): argument is
## not numeric or logical: returning NA

## Warning in mean.default(SSInoLog$SSI_fa_ch, na.rm = TRUE): argument is not
## numeric or logical: returning NA

## Warning in mean.default(SSInoLog$SSI_mo_ch, na.rm = TRUE): argument is not
## numeric or logical: returning NA
```



Evolution of synchrony through time, raw each second

```
par(mar=c(4,4,4,4))
    col <- 1
for (i in 5:length(SSI)){
    plot(1:length(SSI[,i]), SSI[,i], type="l",
    col=rainbow(4)[col], main = names(SSI)[i])
    col <- col+1}
```

Evolution of synchrony through time, mean by minute

```
par(mar=c(4,4,4,4))
    col = 1
for (indexSSI in 5:length(SSI)){
    IntervalNumbersVideo <- ceiling(length(SSI[,indexSSI])/6)
    SSIColumn <- SSI[,indexSSI]
    SSIMinute <- c()
    for (i in 1:IntervalNumbersVideo){
          borneInf <- 1+(i-1)*6
          borneSup <- i * 6
          SSIVectorInterval <- SSIColumn[borneInf:borneSup]
          mean <- mean(SSIVectorInterval, na.rm=TRUE)
          SSIMinute <- c(SSIMinute, mean)}
    plot(1:length(SSIMinute), SSIMinute, type="l", col=rainbow(11)[col], main = names(SSI)[indexSSI])
    col <- col+1}
```

Evolution of synchrony through time, mean by 10 minutes

```
par(mar=c(4,4,4,4))
    col = 1
for (indexSSI in 5:length(SSI)){
    IntervalNumbersVideo <- ceiling(length(SSI[,indexSSI])/60)
    SSIColumn <- SSI[,indexSSI]
    SSITenMinute <- c()
    for (i in 1:IntervalNumbersVideo){
          borneInf <- 1+(i-1)*60
          borneSup <- i * 60
          SSIVectorInterval <- SSIColumn[borneInf:borneSup]
          mean <- mean(SSIVectorInterval, na.rm=TRUE)
          SSITenMinute <- c(SSITenMinute, mean)}
    plot(1:length(SSITenMinute), SSITenMinute, type="l", col=rainbow(4)[col], main = names(SSI)[indexSS
    col <- col+1}
```

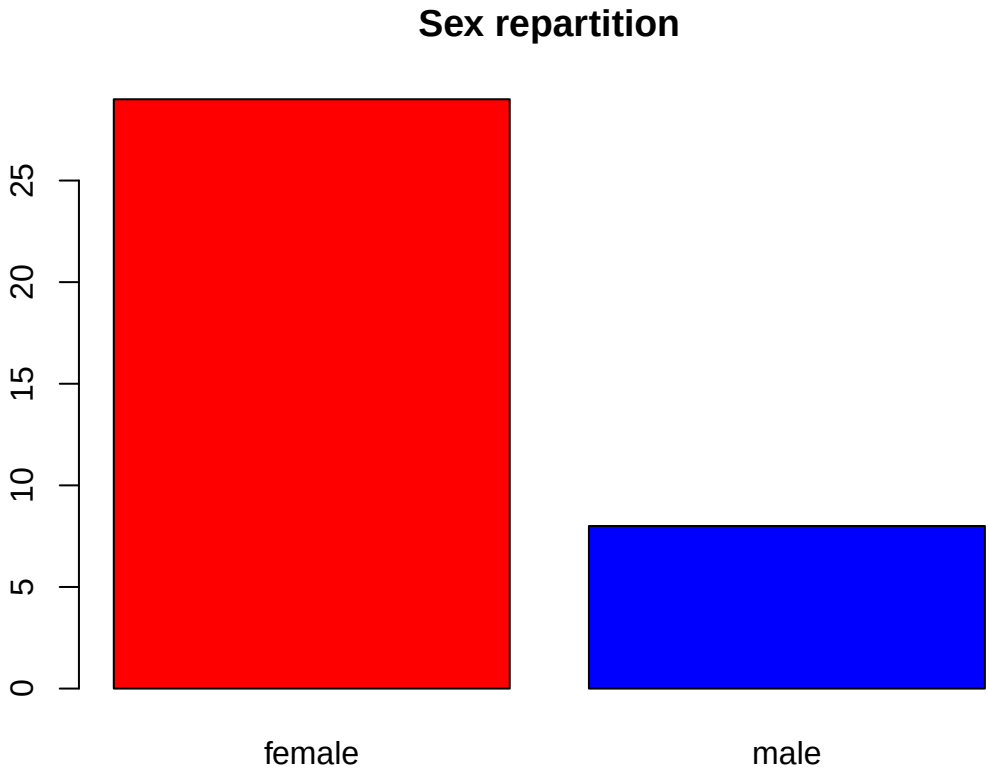
Psychometric database

```
psycho <- read.csv2("/Users/Ofix/Documents/Fac/internat/Recherche/projets/synchro/synchroData/Monrado/Da
psycho <- psycho[-(38:40),]
str(psycho)
#View(psycho)
```

Demographic description

Sex

```
psycho$Sex[which(psycho$Sex == 1 )] <- "male"
psycho$Sex[which(psycho$Sex == 2 )] <- "female"

par(mar=c(3,4,4,4))
barplot(table(psycho$Sex), col=c("red", "blue"), main ="Sex repartition")
```



Age

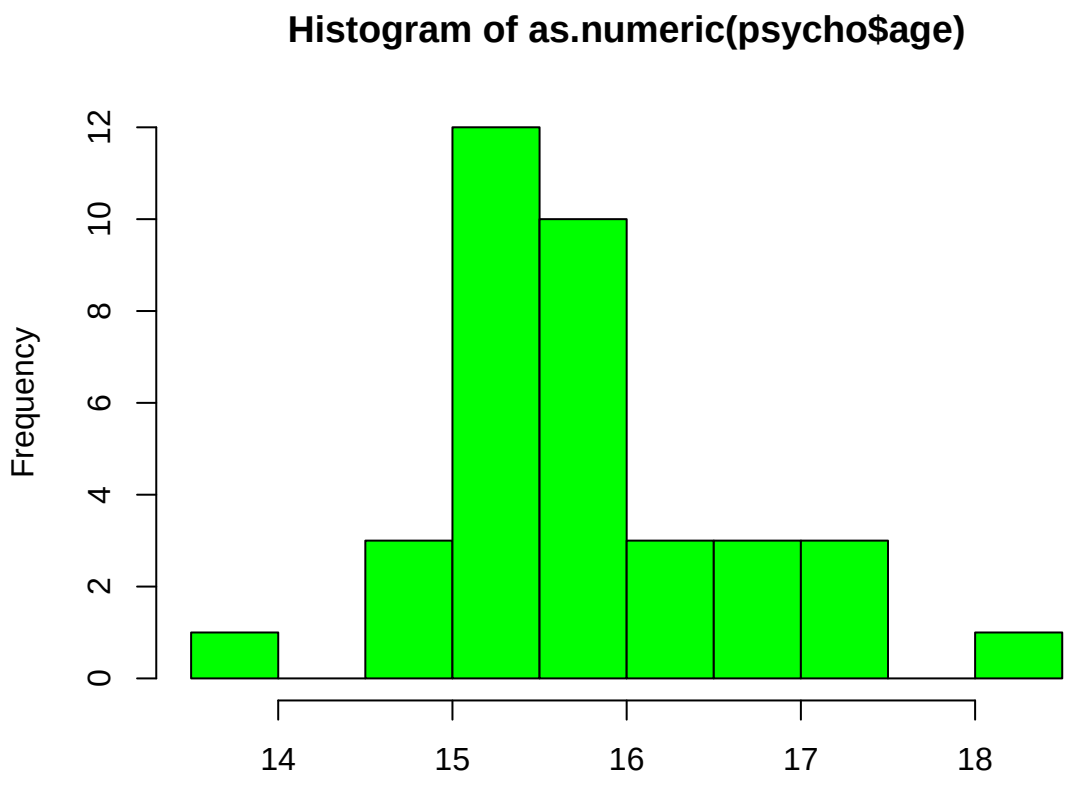
```
psycho$Birthday <- as.Date(psycho$Birthday, format="%d/%m/%y")
psycho$interview_date <- as.Date(psycho$interview_date, format="%d/%m/%y")
str(psycho$Birthday )
```

```
##  Date[1:37], format: "1997-06-18" "1997-12-03" "1997-04-07" "1999-09-08" ...
```

```
str(psycho$interview_date)
```

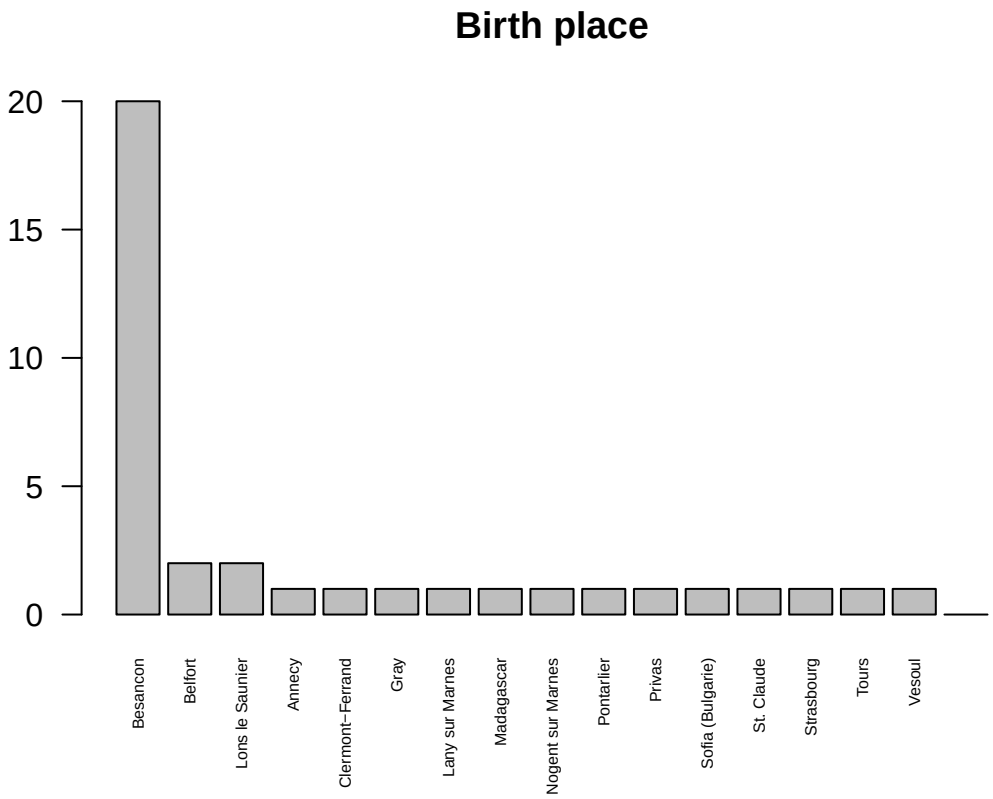
```
##  Date[1:37], format: "2014-01-09" "2014-01-16" "2014-04-17" "2014-04-29" ...
```

```
psycho$age <- (psycho$interview_date-psycho$Birthday)/365.25

par(mar=c(3,4,4,4))
hist(as.numeric(psycho$age), col="green")
```



Birth places

```
par(mar=c(5,4,4,4))
barplot(sort(table(psycho$Birth_place), decreasing = TRUE ), las=2, cex.names=0.5, main="Birth place")
```

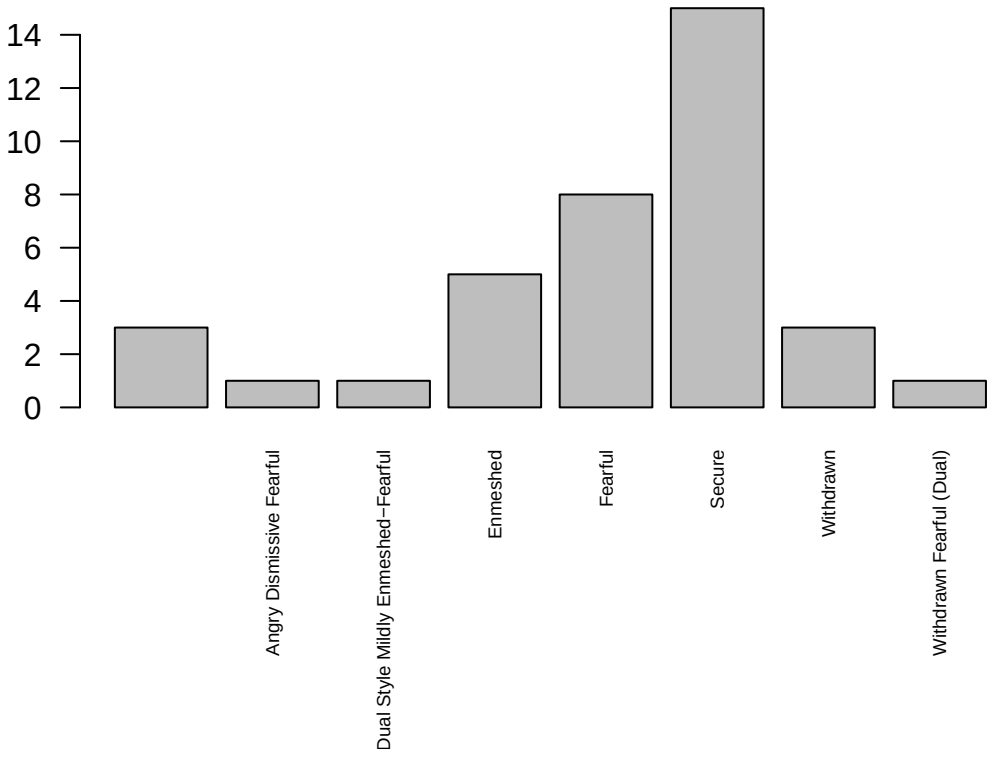



```
psycho$Birth_place
```

```
##  [1] Gray              Madagascar        Besancon
##  [4] Lany sur Marnes   Besancon          Besancon
##  [7] Sofia (Bulgarie)  Besancon          Besancon
## [10] Lons le Saunier   Besancon          Besancon
## [13] Besancon          Besancon          Pontarlier
## [16] Lons le Saunier   Besancon          Besancon
## [19] Besancon          Nogent sur Marnes Belfort
## [22] Strasbourg        Besancon          Besancon
## [25] Besancon          Besancon          Annecy
## [28] Belfort           Besancon          Besancon
## [31] St. Claude        Privas            Clermont-Ferrand
## [34] Tours             Besancon          Besancon
## [37] Vesoul
## 17 Levels:  Annecy Belfort Besancon Clermont-Ferrand ... Vesoul
```

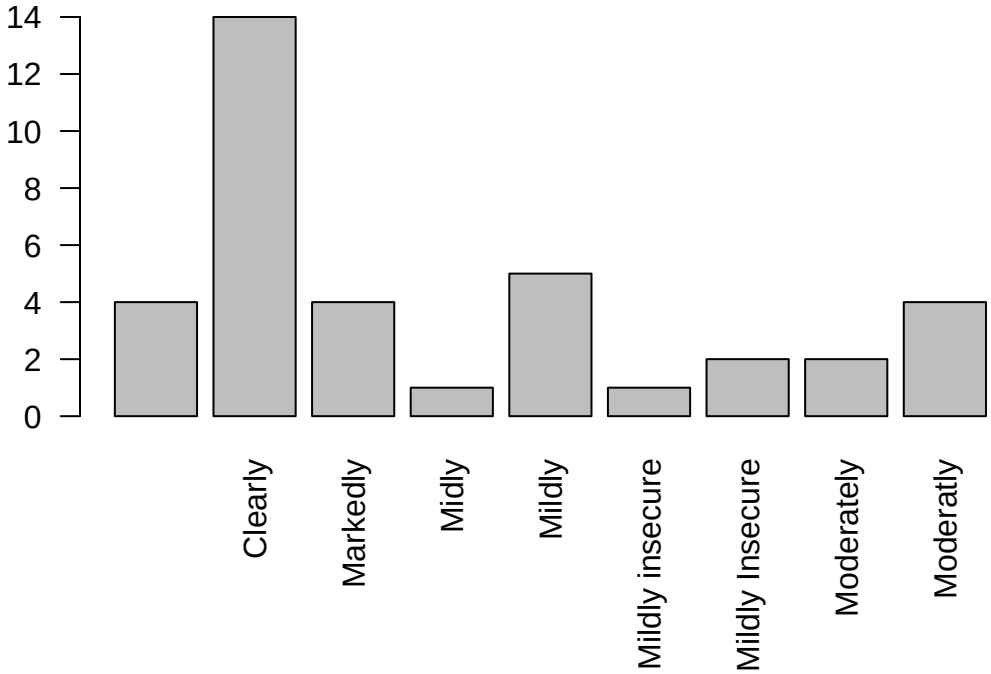
Attachement styles

```
par(mar=c(9,5,3,3))
barplot(table(psycho$attachement_style), las=2, cex.names=0.6)
```



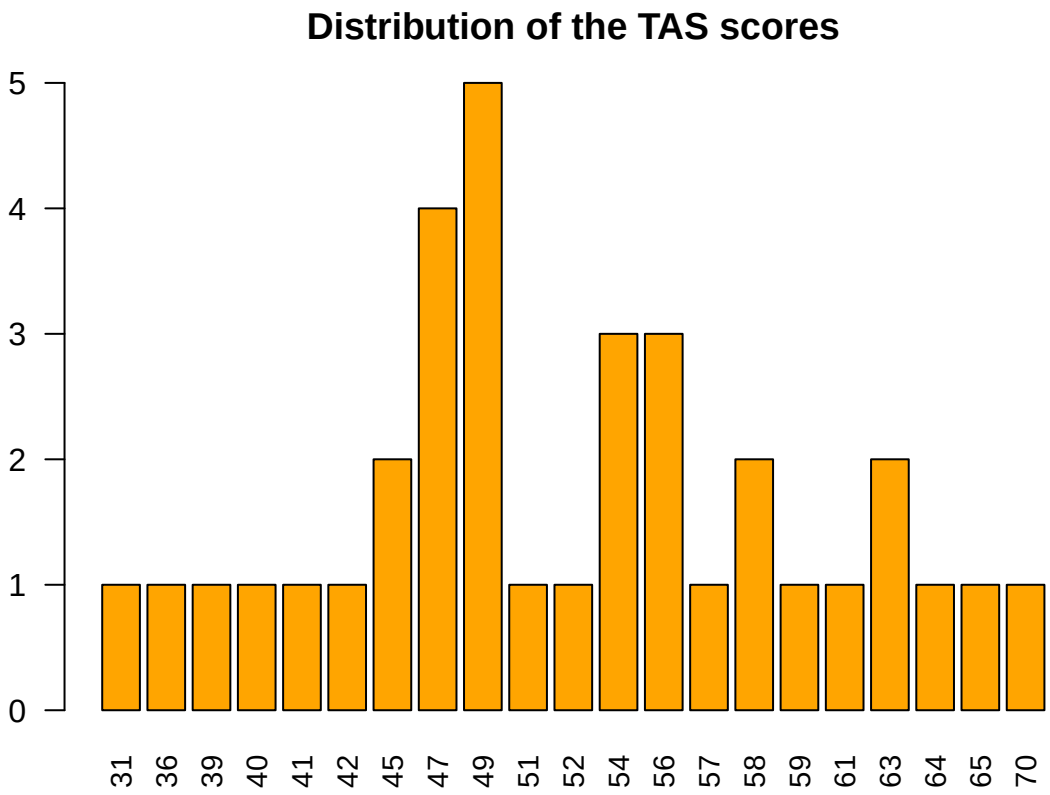
Insecurity level

```
par(mar=c(9,5,3,3))
barplot(table(psycho$insecurite_level), las=2)
```



TAS

```
par(mar=c(3,3,3,3))
barplot(table(psycho$TAS_total), las=2, col="orange", main="Distribution of the TAS scores", cex.name=0
```



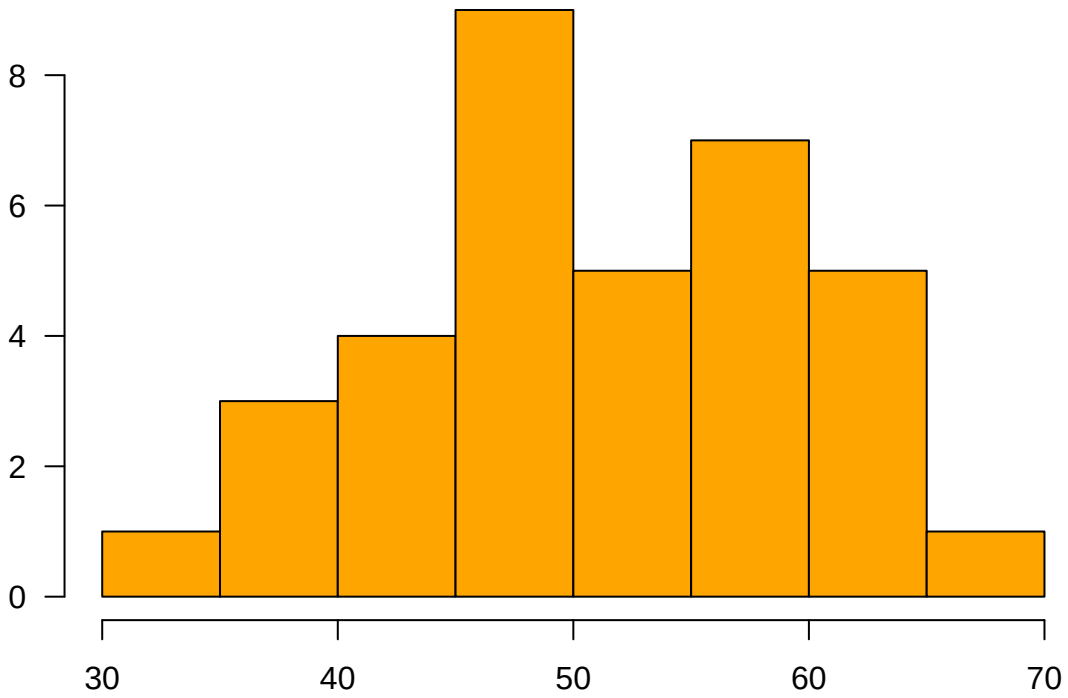

```
hist(psycho$TAS_total, las=1, col="orange", main="Distribution of the TAS scores", cex.name=0.9)
```

```
## Warning in plot.window(xlim, ylim, "", ...): "cex.name" is not a graphical
## parameter
```

```
## Warning in title(main = main, sub = sub, xlab = xlab, ylab = ylab, ...):
## "cex.name" is not a graphical parameter
```

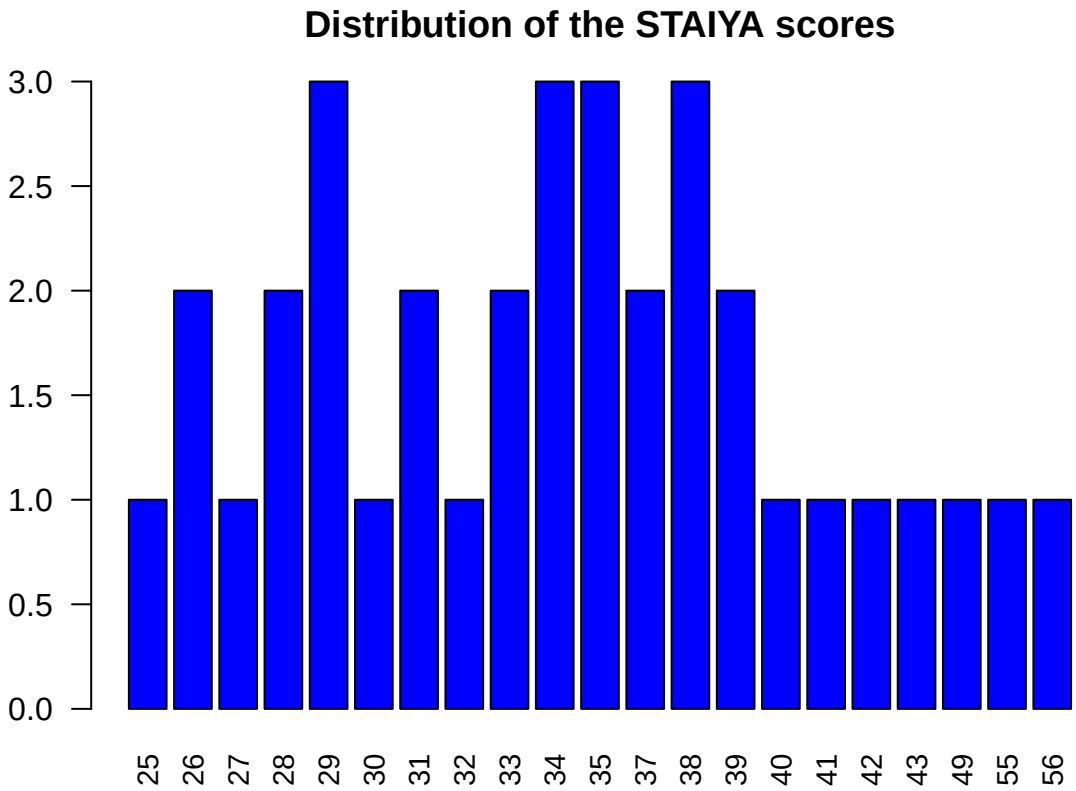
```
## Warning in axis(1, ...): "cex.name" is not a graphical parameter
```

```
## Warning in axis(2, ...): "cex.name" is not a graphical parameter
```



STAIYA

```
par(mar=c(3,3,3,3))
barplot(table(psycho$STAIYA_total), las=2, col="blue", main="Distribution of the STAIYA scores", cex.nar
```



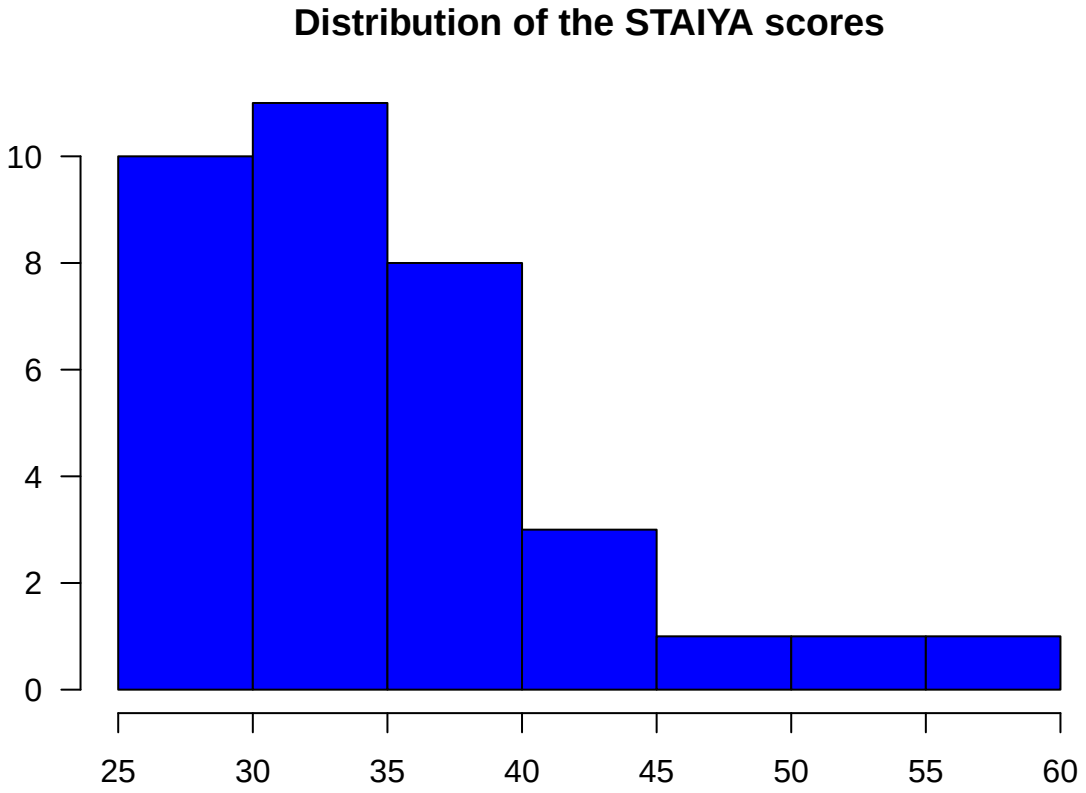

```
hist(psycho$STAIYA_total, las=1, col="blue", main="Distribution of the STAIYA scores", cex.name=0.9)
```

```
## Warning in plot.window(xlim, ylim, "", ...): "cex.name" is not a graphical
## parameter
```

```
## Warning in title(main = main, sub = sub, xlab = xlab, ylab = ylab, ...):
## "cex.name" is not a graphical parameter
```

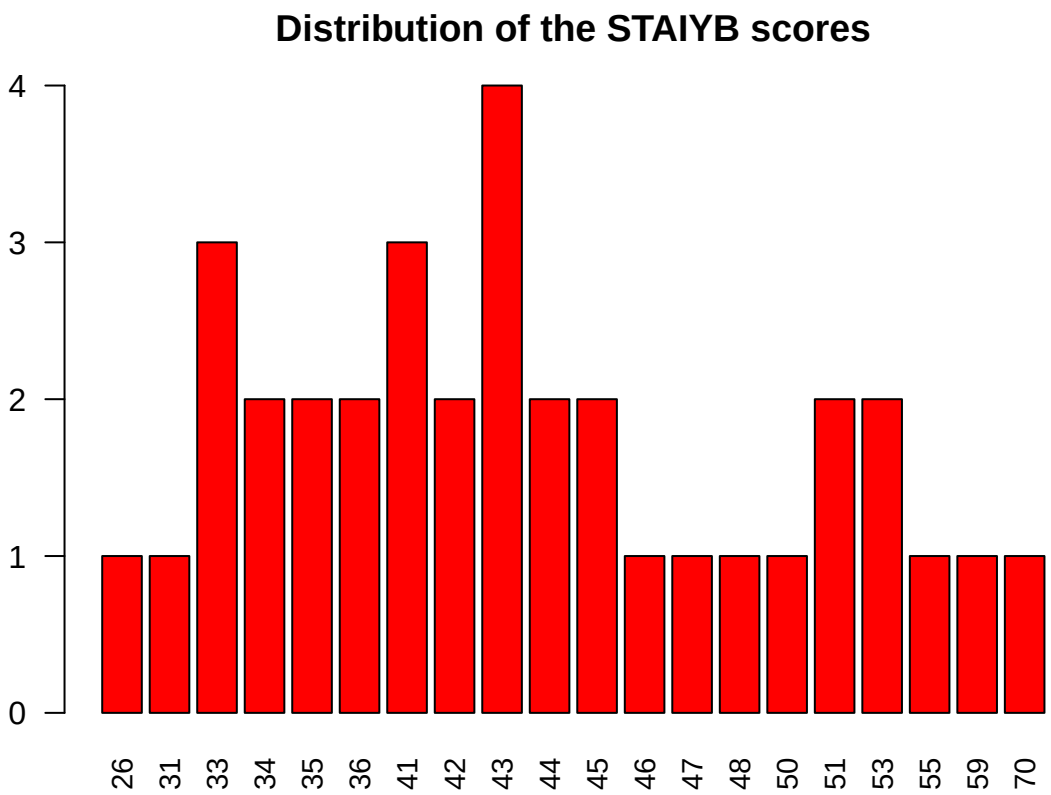
```
## Warning in axis(1, ...): "cex.name" is not a graphical parameter
```

```
## Warning in axis(2, ...): "cex.name" is not a graphical parameter
```



STAIYB

```
par(mar=c(3,3,3,3))
barplot(table(psycho$STAIYB_total), las=2, col="red", main="Distribution of the STAIYB scores", cex.nam
```



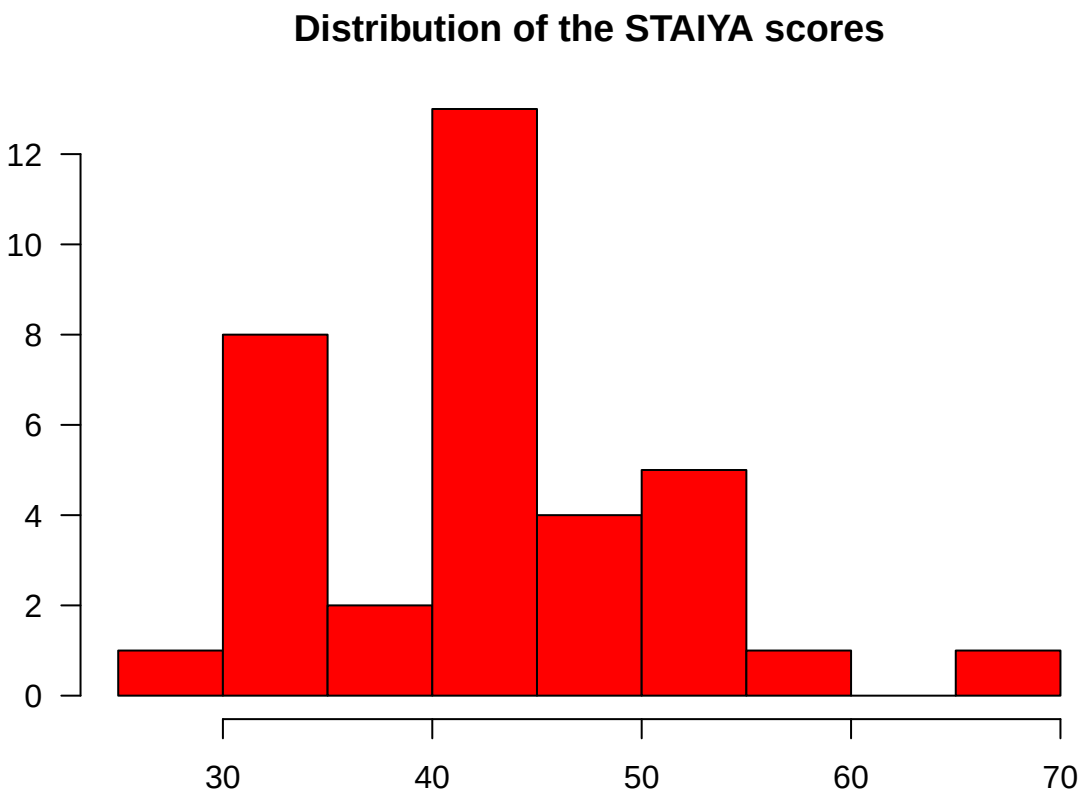

```
hist(psycho$STAIYB_total, las=1, col="red", main="Distribution of the STAIYA scores", cex.name=0.9)
```

```
## Warning in plot.window(xlim, ylim, "", ...): "cex.name" is not a graphical
## parameter
```

```
## Warning in title(main = main, sub = sub, xlab = xlab, ylab = ylab, ...):
## "cex.name" is not a graphical parameter
```

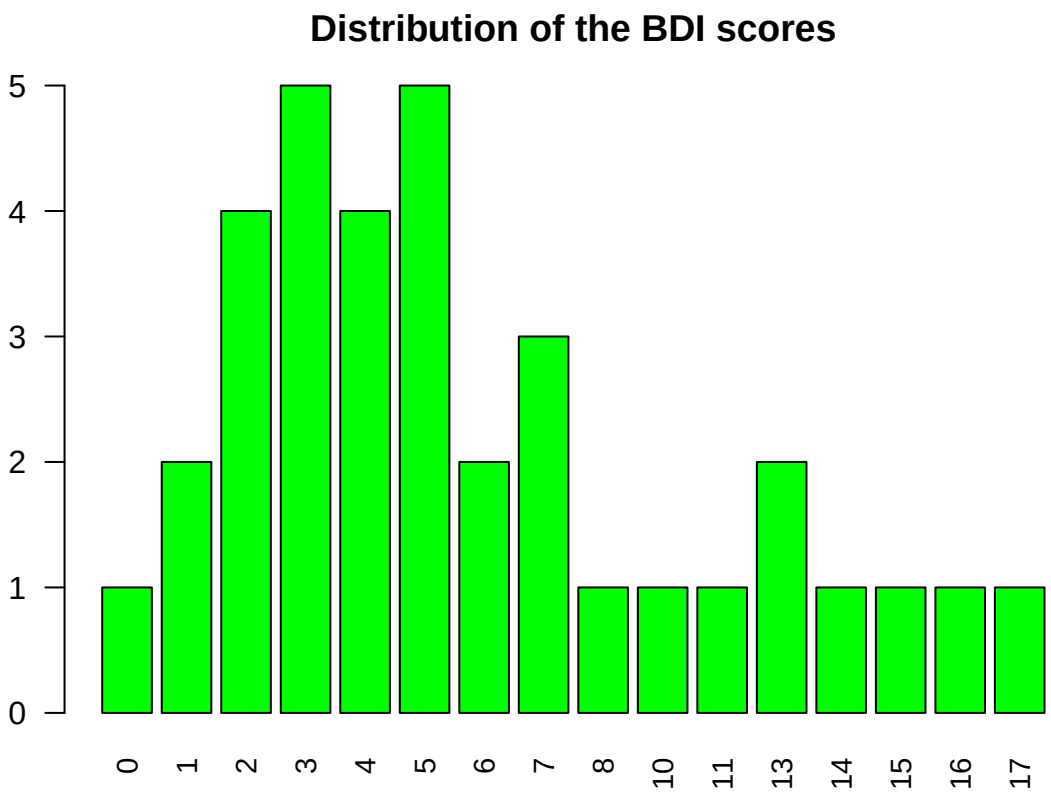
```
## Warning in axis(1, ...): "cex.name" is not a graphical parameter
```

```
## Warning in axis(2, ...): "cex.name" is not a graphical parameter
```



BDI total

```
par(mar=c(3,3,3,3))
barplot(table(psycho$BDI_total), las=2, col="green", main="Distribution of the BDI scores", cex.name=0.
```



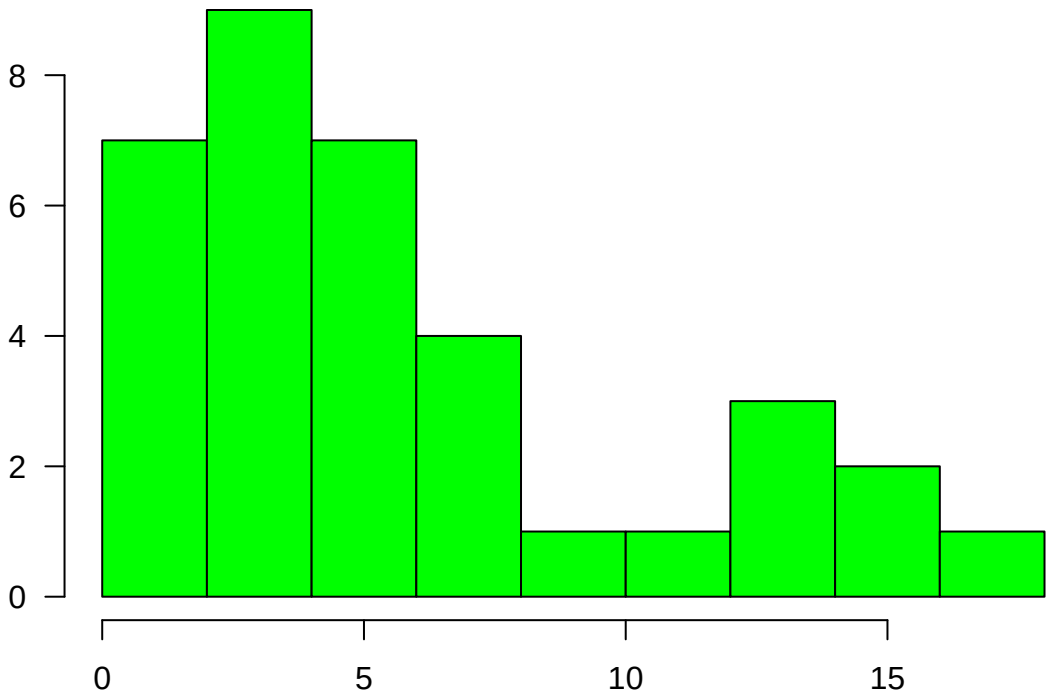

```
hist(psycho$BDI_total, las=1, col="green", main="Distribution of the BDI scores", cex.name=0.9)
```

```
## Warning in plot.window(xlim, ylim, "", ...): "cex.name" is not a graphical
## parameter
```

```
## Warning in title(main = main, sub = sub, xlab = xlab, ylab = ylab, ...):
## "cex.name" is not a graphical parameter
```

```
## Warning in axis(1, ...): "cex.name" is not a graphical parameter
```

```
## Warning in axis(2, ...): "cex.name" is not a graphical parameter
```



Models of synchrony

```
SSI_fa_th_lme <- lmer(SSI_fa_th ~ Time_min + (1|video), data=SSI)
summary(SSI_fa_th_lme)
#plot(SSI_fa_th_lme)
res <- residuals(SSI_fa_th_lme)
hist(SSI$SSI_fa_th)
qqnorm(res)
SSI_fa_th_List <- c()
for (i in families){
  SSI_fa_th_List <- c(SSI_fa_th_List, mean(SSI[which(SSI$video==i),]$SSI_fa_th, na.rm=TRUE))
}
print(SSI_fa_th_List)
#plot(SSI_fa_th_List, type="b")
```

```
# log of the data
log_SSI_fa_th <- hist(log(SSI$SSI_fa_th))
SSI_fa_th_log_lme <- lmer(log(SSI_fa_th) ~ Time_min + (1|video), data=SSI)
res_log <- residuals(SSI_fa_th_log_lme)
qqnorm(res_log)
summary(SSI_fa_th_log_lme)
```

```
# root square of the data
sq_SSI_fa_th <- hist(sqrt(SSI$SSI_fa_th))
SSI_fa_th_sq_lme <- lmer(sqrt(SSI_fa_th) ~ Time_min + (1|video), data=SSI)
res_sq <- residuals(SSI_fa_th_sq_lme)
qqnorm(res_sq)
summary(SSI_fa_th_sq_lme)
```