

Synchrony in relationship, example with MONRADO Data

Thomas Gargot

May, 10th 2016

Contents

Fixed variables	2
Functions list	2
Import data	5
Clean dataframe	5
Presentation of the data	6
Length of the videos in minutes	7
Length of the videos in number of frames	8
Number of Available (True) and Not Available (False) data for each participant	9
Global Motion history	10
Mean Motion history by video by participant	10
Motion history box plots by frame (raw data), all videos	11
Raw data and mean of Motion History on sliding and non overlapping intervals on 00034 video	15
Raw data	15
Sliding interval	16
Non overlapping interval	16
Focus on the motion history of the first 10 seconds of the first video 00034	16
Sliding interval function on a 5 frames interval	16
Motion history of the father during 10-20 seconds of the first video 00034	18
Mean motion history by 10 sec plots	20
Export no log data in text files	40
Export log data in text files	41
SyncPy utilisation for creating synchrony dataframe	42
Description of SSIlog data frame	43
Description of noLogSSI data frame	43
Synchrony scores log for each dyad, triad and for the whole group	43
Synchrony scores noLog for each dyad, triad and for the whole group	64
Evolution of synchrony through time, raw each second	91
Evolution of synchrony through time, mean by minute	91
Evolution of synchrony through time, mean by 10 minutes	92
Psychometric database	92
Demographic description	92

Attachement styles	95
Insecurity level	96
TAS	97
STAIYA	98
STAIYB	100
BDI total	102
Models of synchrony	104

```
rm(list = ls(all.names = TRUE))
```

Fixed variables

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

# working directory
# where this report is
setwd("/Users/0fix/Documents/Fac/internat/Recherche/projets/synchro/synchroData/Git/Monrado/Reports/")

# blue will refer to father
# red will refer to mother
# green to child
colorOrderList <- 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)
    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

Mean motion history (non overlapping 5 frames intervals) 00034 video, 2nd second

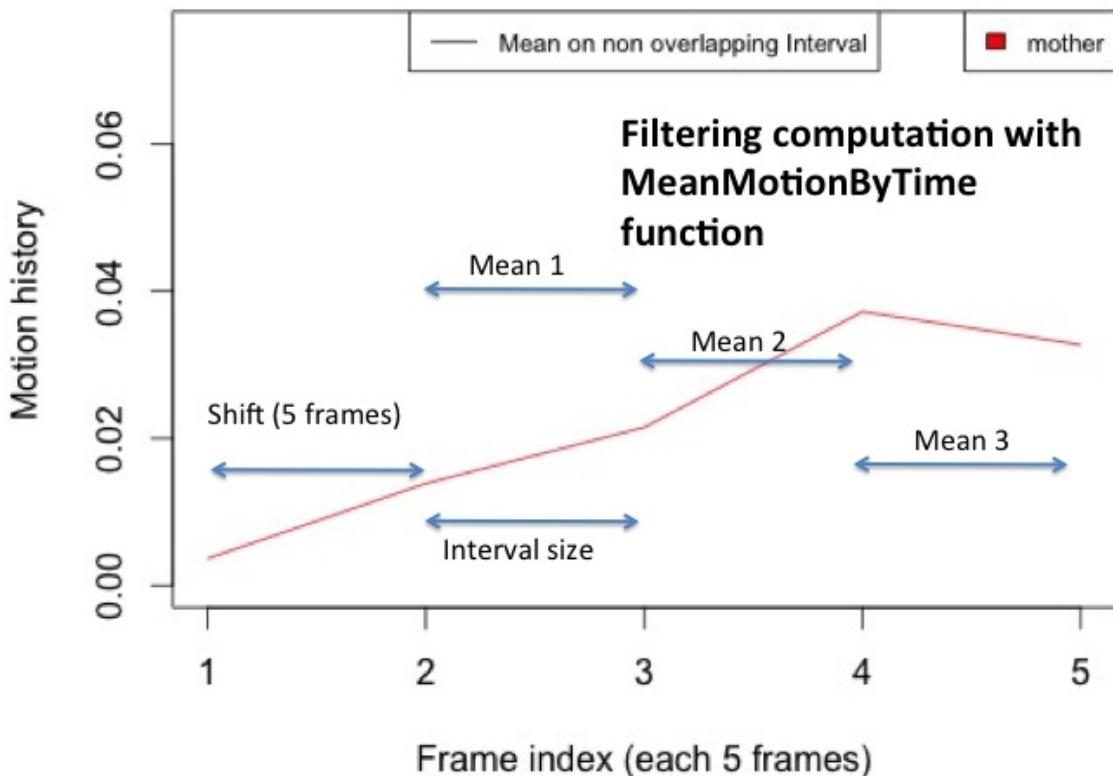


Figure 1:

```
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
      borneresup <- i*interval
      x <- rbind(x, mean(dataVector[borneinf:borneresup], na.rm=TRUE))
    }
  }
  return(x)
}
```

```

        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

Mean motion history (Sliding 5 frames interval) on 00034 video, 2nd second

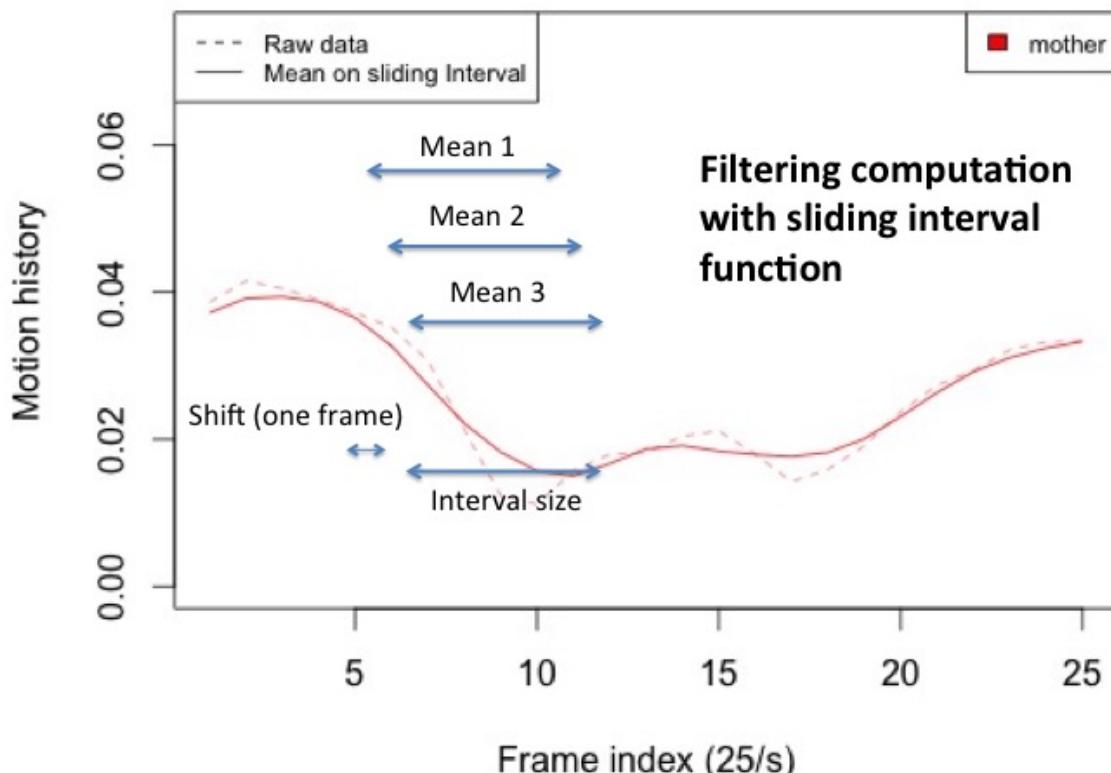


Figure 2:

```

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


```

```

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

```

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 fileList){
  name <- sub(FileExtension, "", i)
  families <- c(families, name)
}
families

## [1] "00034"   "00037"   "00041"   "00048"   "0206"    "1106"    "1606"
## [8] "BAJE059" "BALE050" "BALU062" "BEAL036" "BEAM031" "BICA"    "BRL0041"
## [15] "COL0022" "DIPE004" "DOMA"    "DRNE"    "FOMA057" "GROP039" "HAJA052"
## [22] "HUMA058" "JAEM046" "JEE0040" "JOCE014" "LACL"    "MAEL048" "MAME20"
## [29] "MAPA029" "MIPH043" "MOSA065" "RAEM049" "RAKA008" "RIEMO"   "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 data
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", "fatherShifted")]

```

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 ...
## $ 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 ...
## $ fatherShifted: num NA NA NA NA NA NA NA NA NA ...
## $ logFather    : num 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 ...

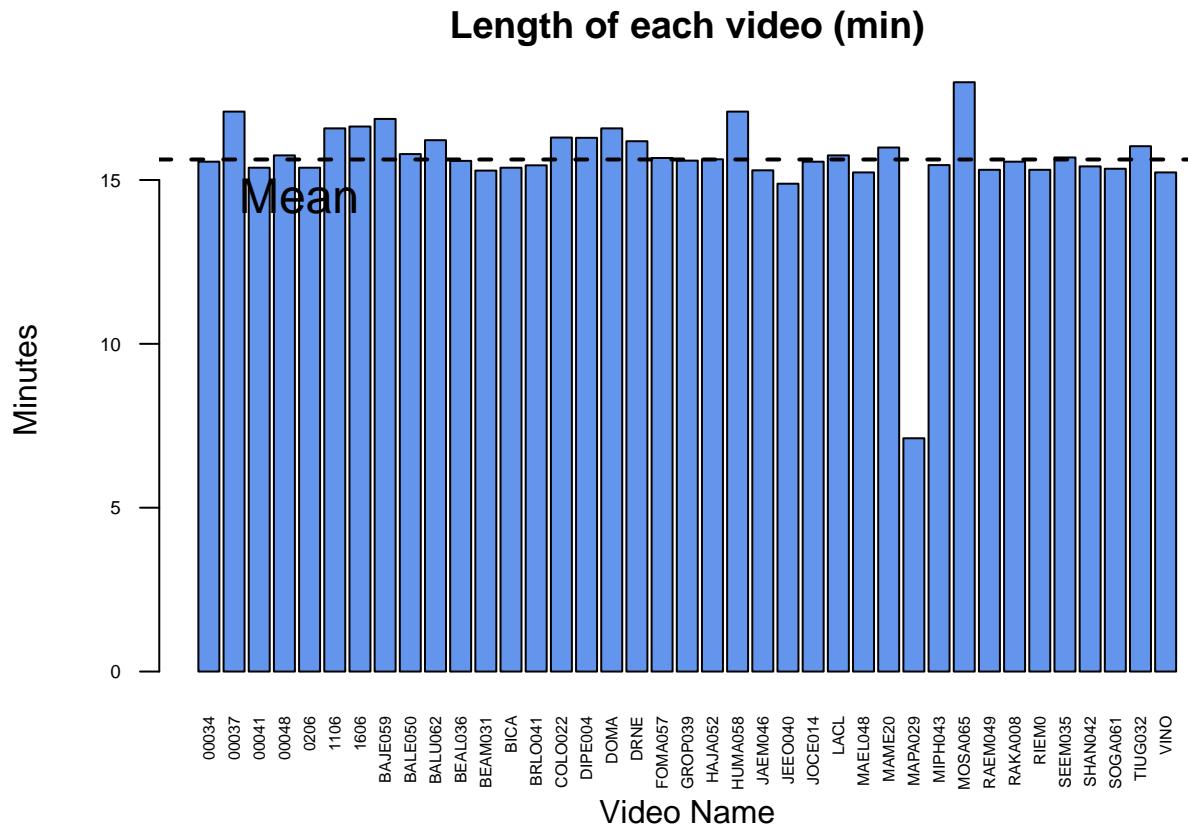
summary(data)

##      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

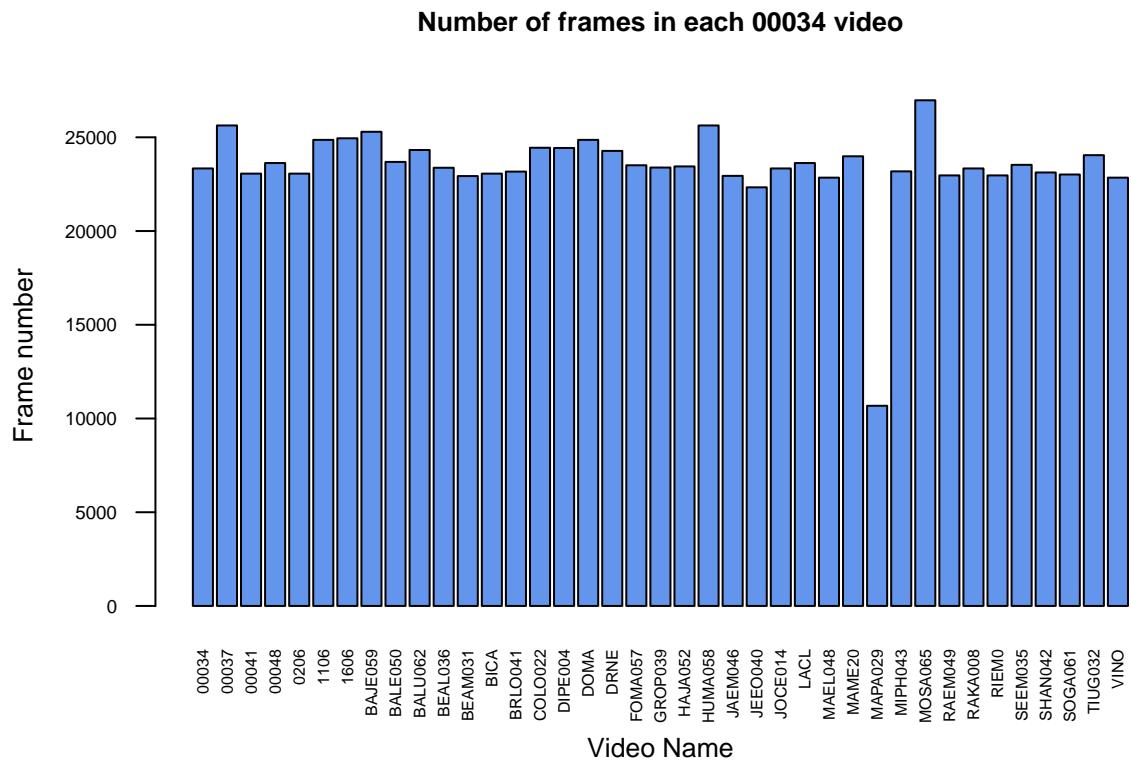
```

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

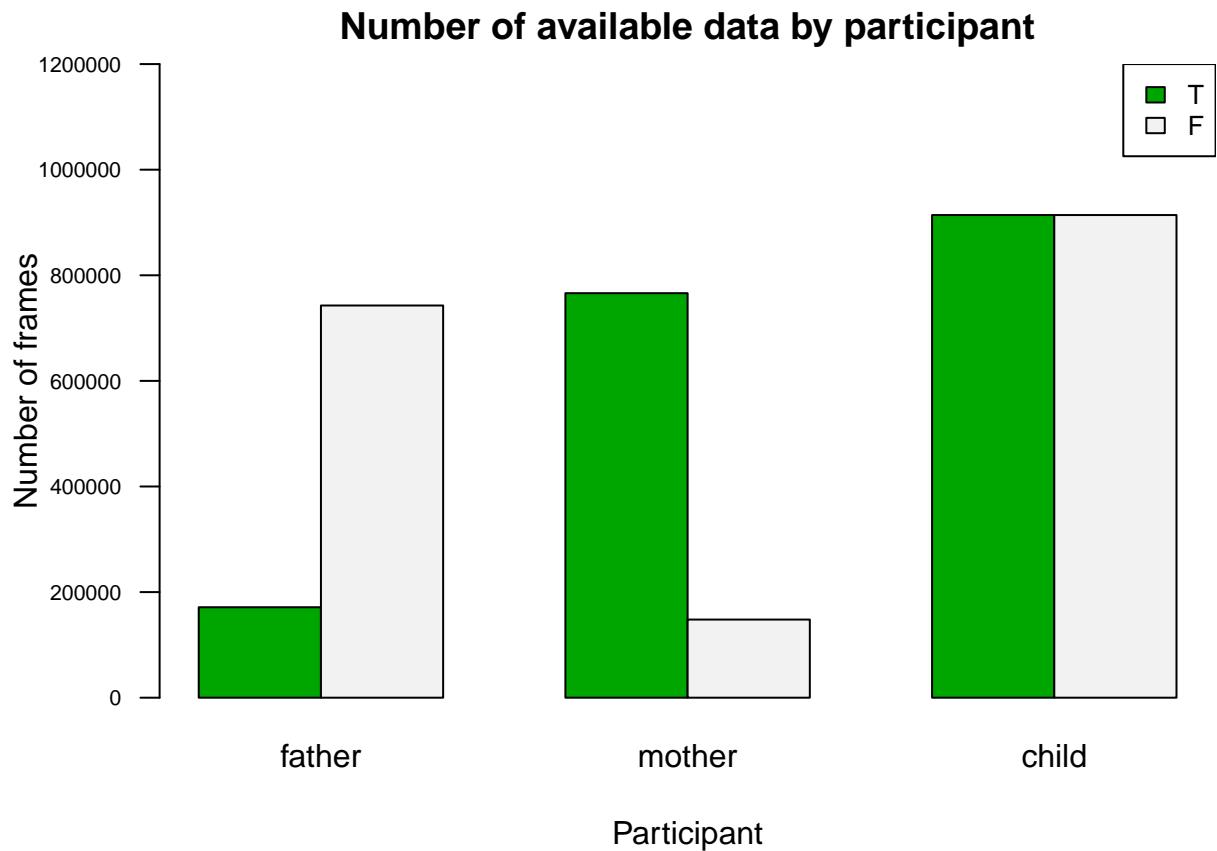
Length of the videos in minutes



Length of the videos in number of frames



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

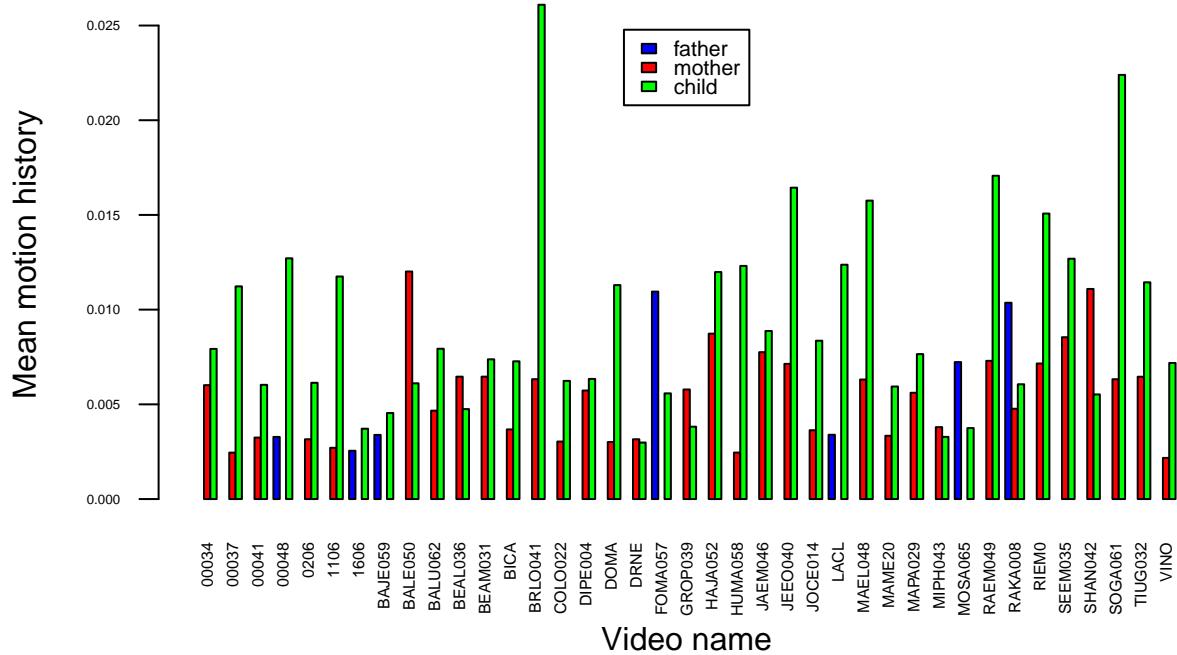


All the participants involved are filmed. * All the children are filmed and we have data for each. * More often there is the other with him/her sometimes, it is the father * In some videos for instance RAKA008, there are 3 subjects

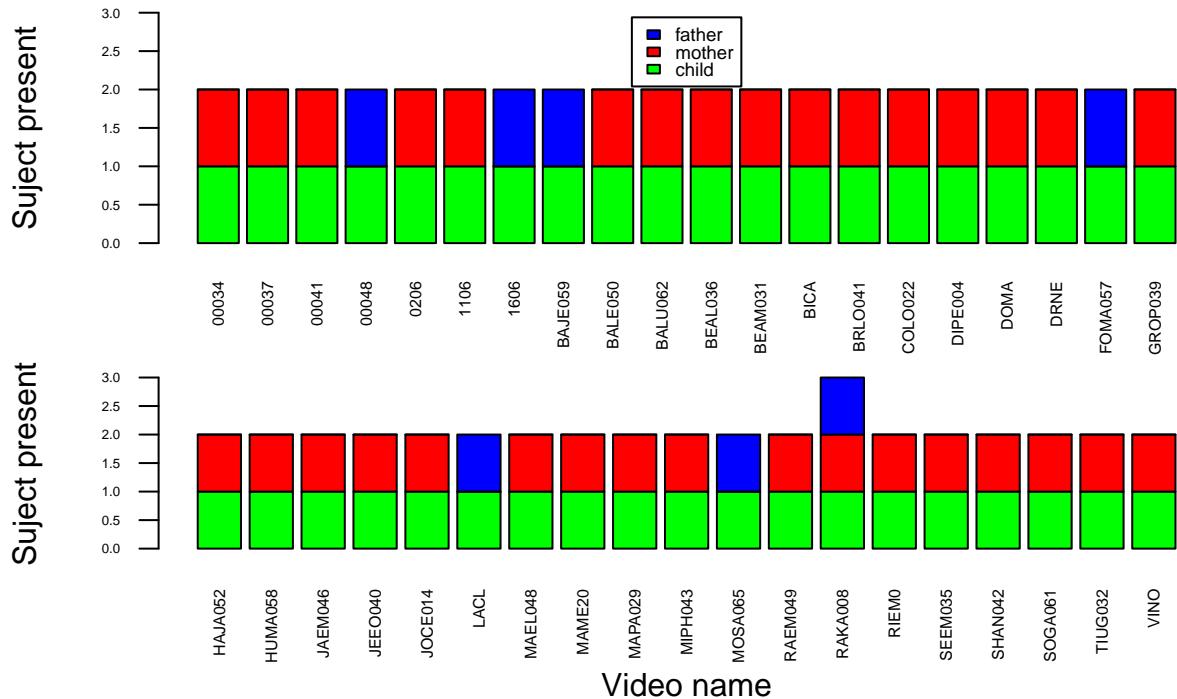
Global Motion history

Mean Motion history by video by participant

Mean Motion history in each video



Configuration of participant by video



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 comparisons of the videos easier than in the INCANT study.

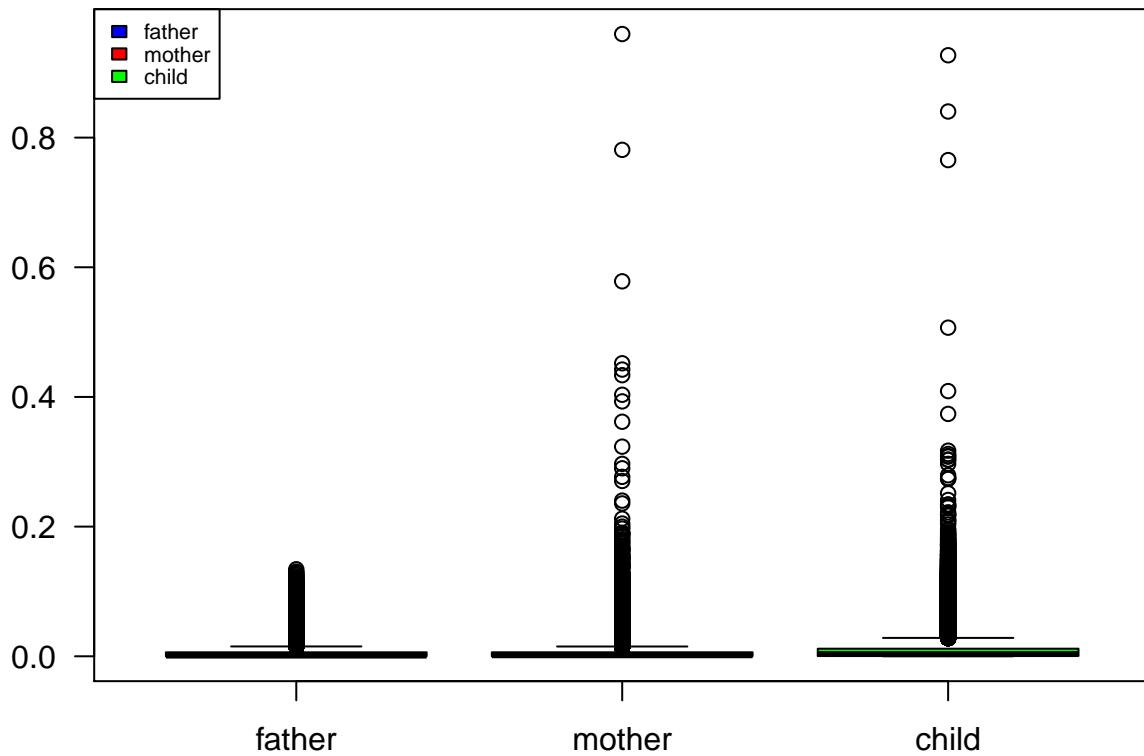
Motion history box plots by frame (raw data), all videos

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)
```

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

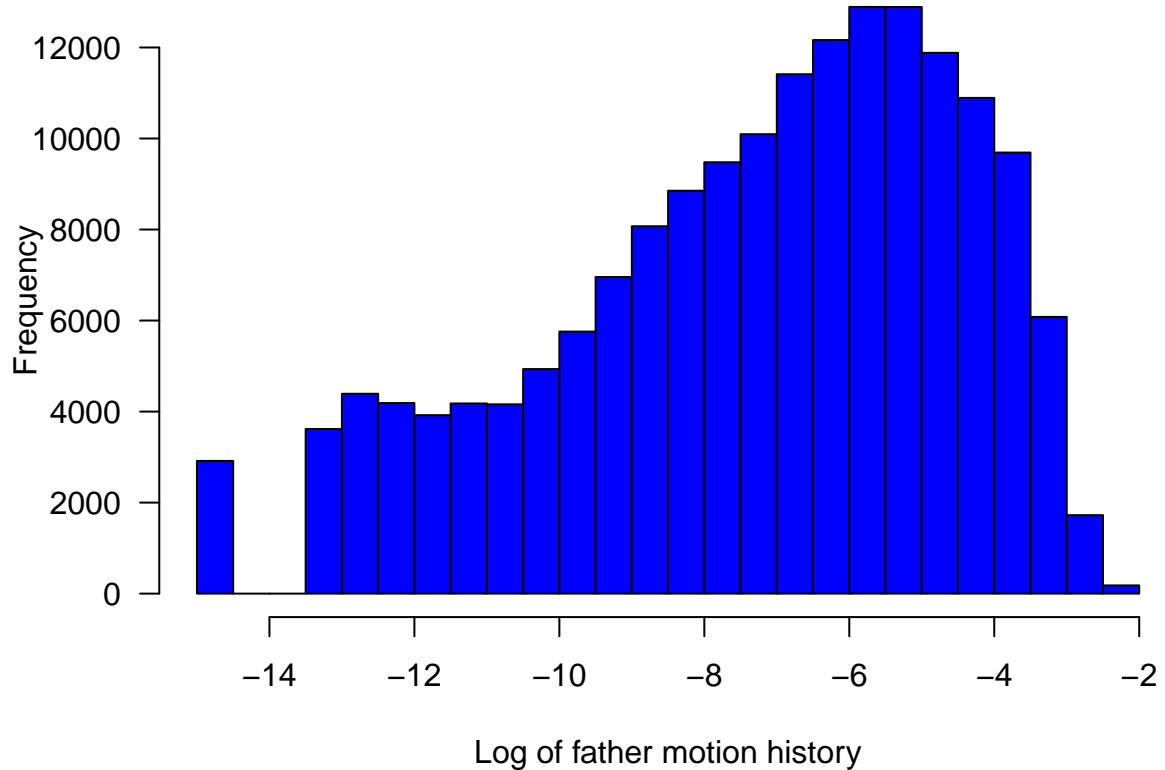


jects data distribution are very similar.

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

The sub-

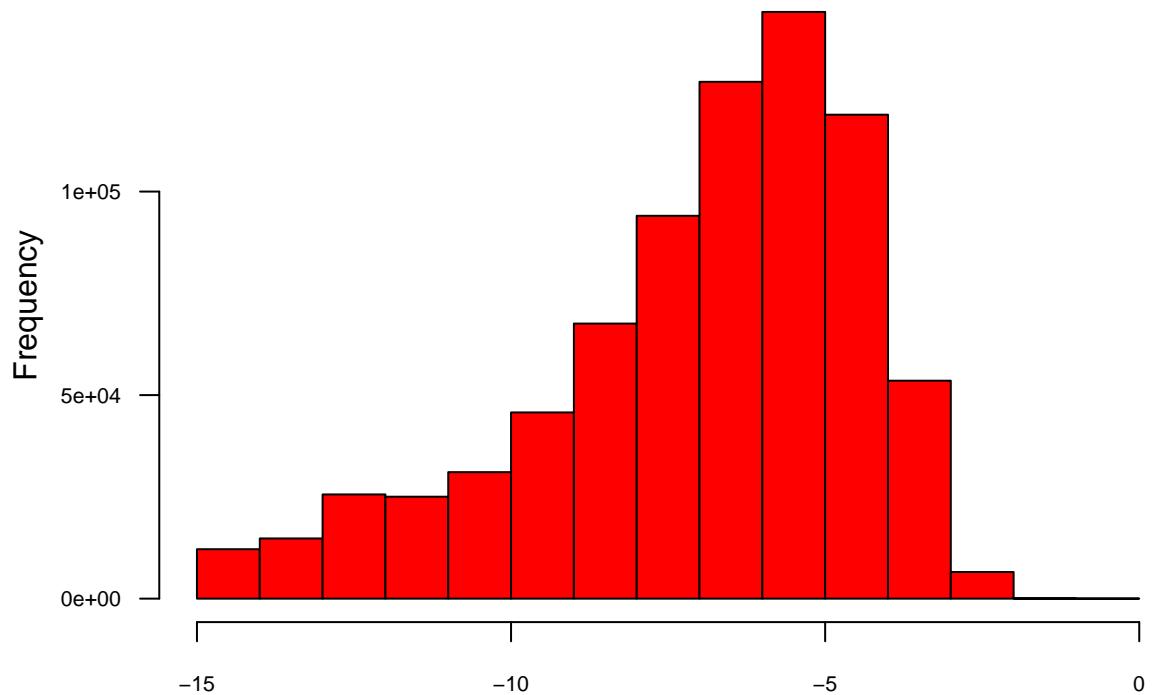
Histogram of data\$logFather



Log of father motion history

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

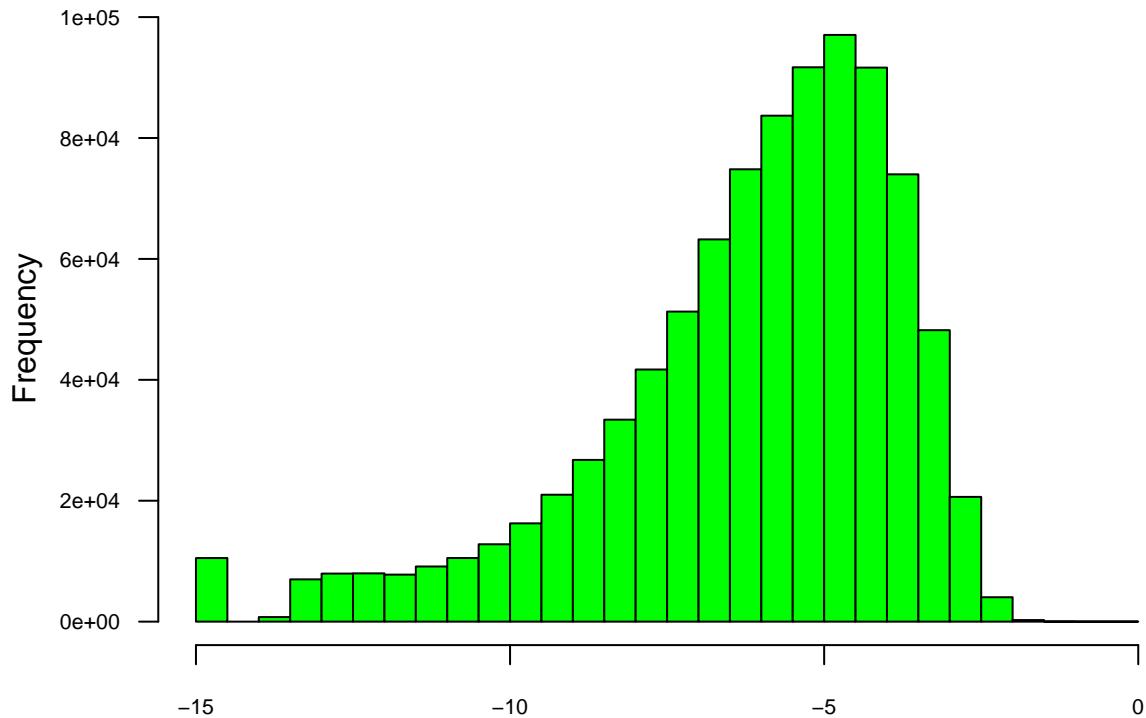
Histogram of data\$logMother



Log of mother motion history

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

Histogram of data\$logChild



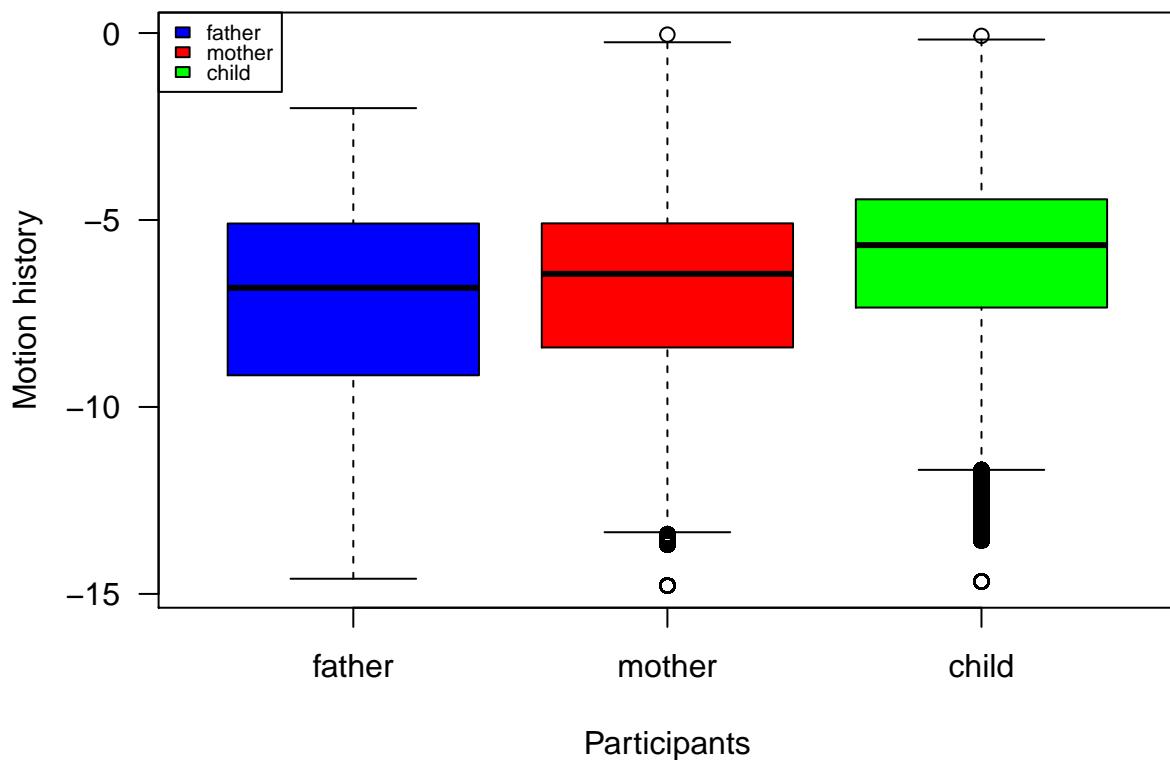
Log of child motion history

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

```
data$childShifted <- data$child + min(data$child[which(data$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)
```

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



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

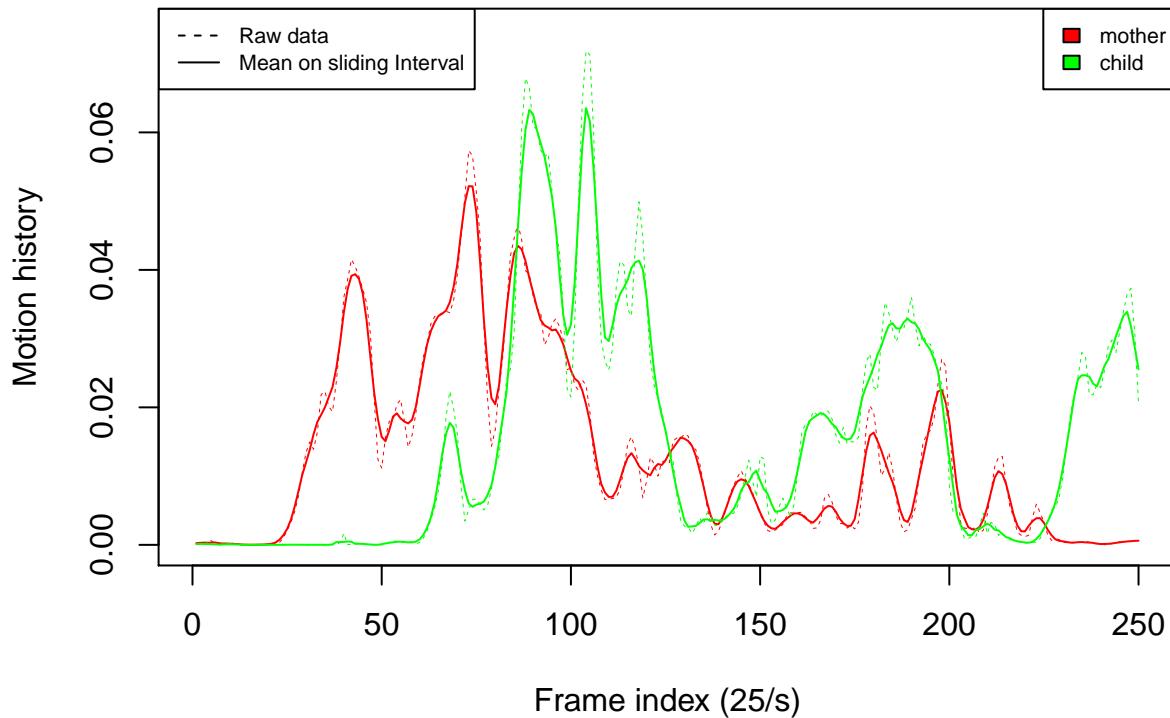
```
par(mar=c(4,4,4,2))  
plot(1:250, data$mother[3:252], main="Mean motion history (Sliding 5 frames interval)  
on 00034 video, first 10 seconds ", xlab="Frame index (25/s)",  
ylab="Motion history",  
col="red", type="l", lty=2, lwd=0.5, ylim=c(0, 0.075))  
lines(slidedMother[1:250], col="red", lty=1)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

```

Mean motion history (Sliding 5 frames interval) on 00034 video, first 10 seconds



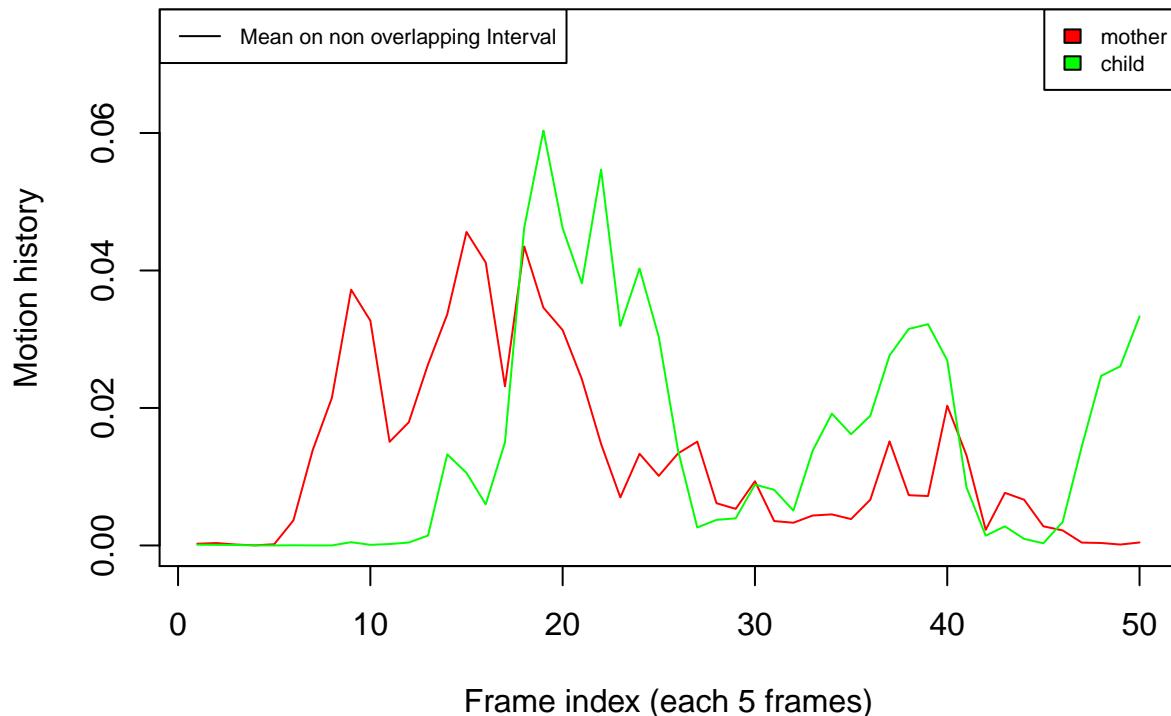
Non overlapping interval function on a 5 frames interval

```

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)

```

Mean Motion history (non overlapping 5 frames intervals) for father on 00034 video, first 10 seconds

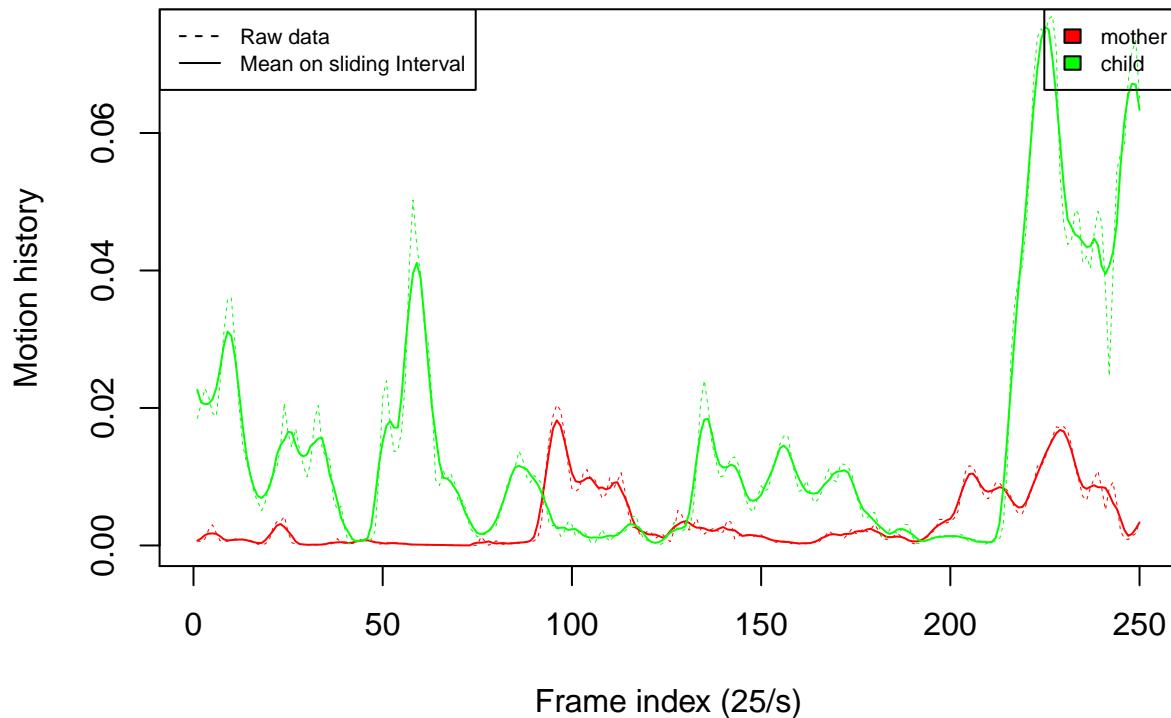


Motion history of the father during 10-20 seconds of the first video 00034

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)
```

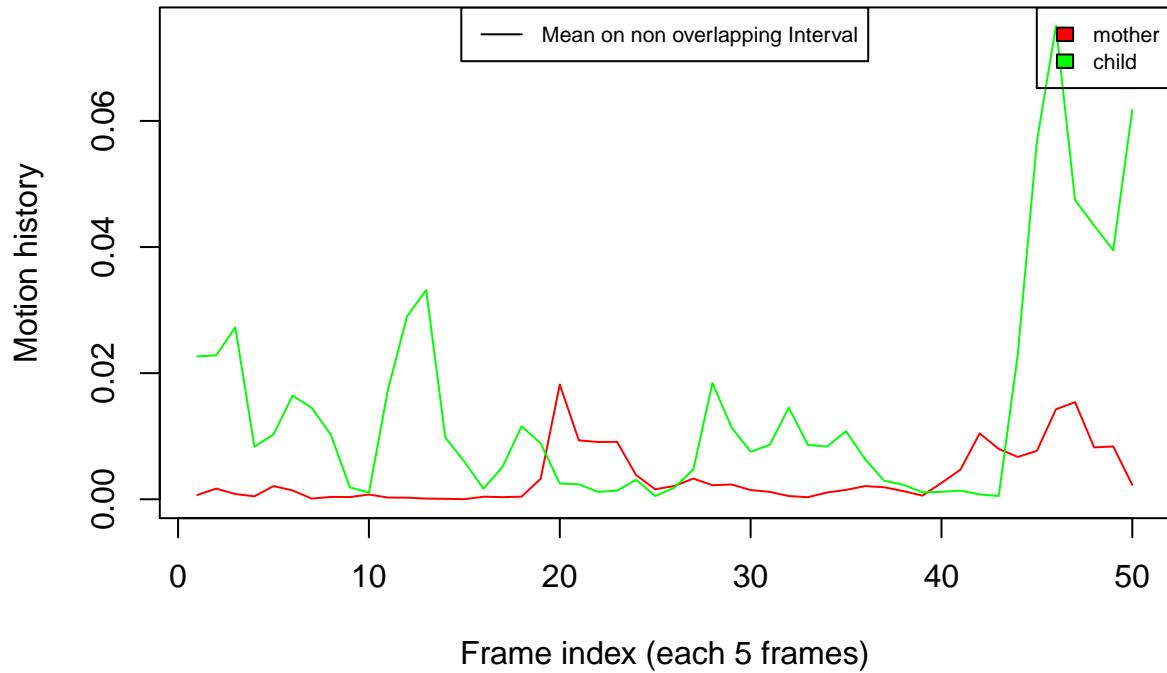
Mean motion history (Sliding 5 frames interval) for father on 00034 video, 10–20 seconds



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 (non overlapping 5 frames intervals) on 00034 video, between 10–20 seconds



Mean motion history by 10 sec plots

```

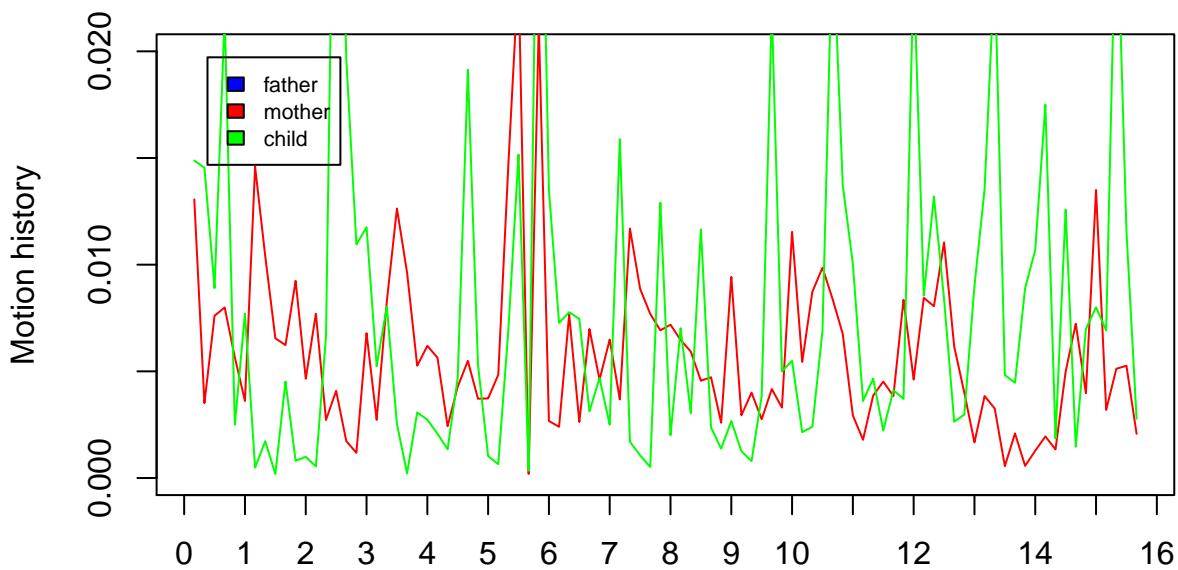
for (i in 1:Number0fvideos){
  fatherMinute<- MeanMotionByTime("father", index0fvideos=i, interval=250, data)

  motherMinute<- MeanMotionByTime("mother", index0fvideos=i, interval=250, data)

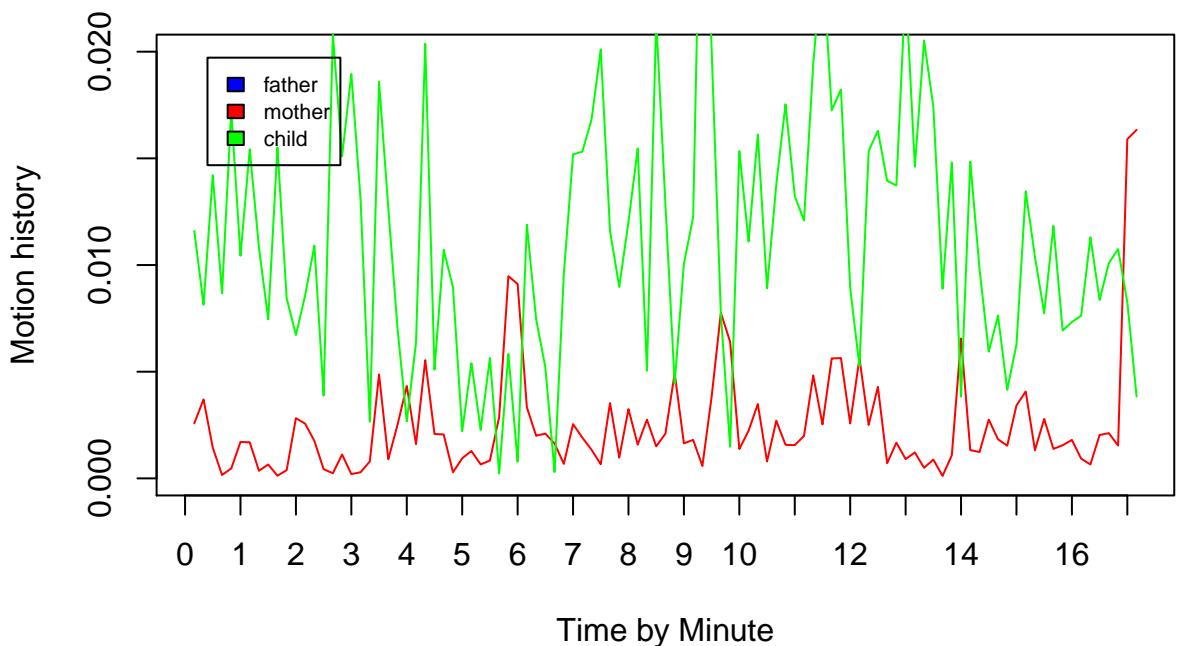
  childMinute<- MeanMotionByTime("child", index0fvideos=i, interval=250, data)

  par(mar=c(4,4,4,2))
    plot ((1:length(fatherMinute)/6), 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, (round(length(fatherMinute)/6)), round((length(fatherMinute)/6))))
    lines((1:length(fatherMinute)/6), motherMinute, col="red")
    lines((1:length(fatherMinute)/6), childMinute, col="green")
    legend("topleft", inset=.05, ParticipantsList[1:3],
    fill=colOrderList[1:3], cex=0.7)}
  
```

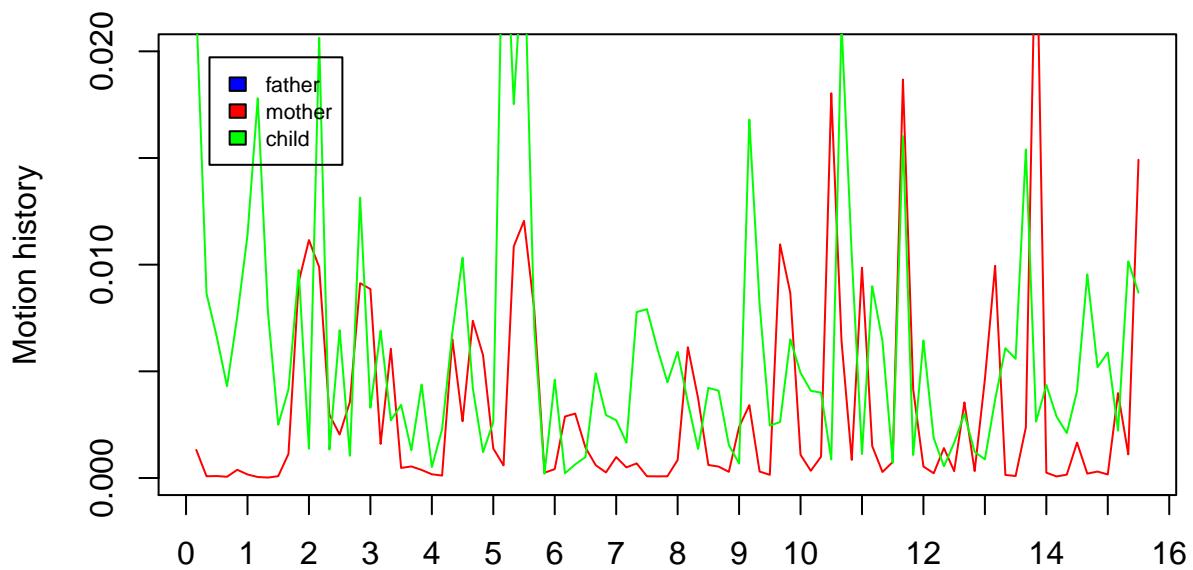
**Mean motion history (non overlapping 10 sec intervals)
on 00034 video**



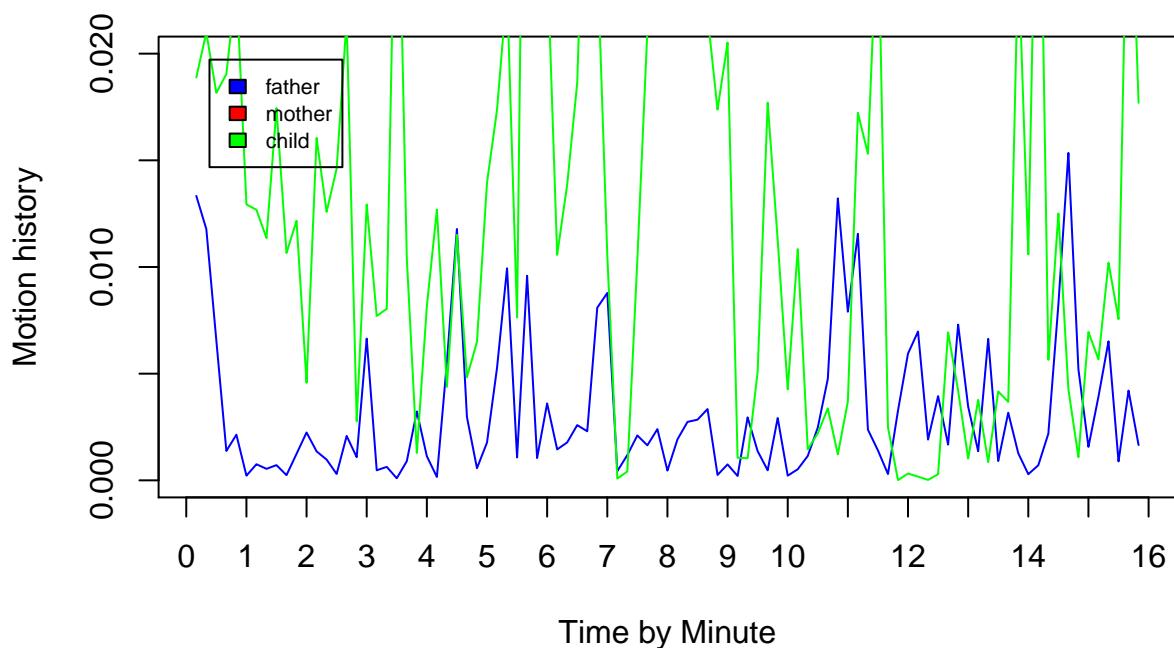
**Time by Minute
Mean motion history (non overlapping 10 sec intervals)
on 00037 video**



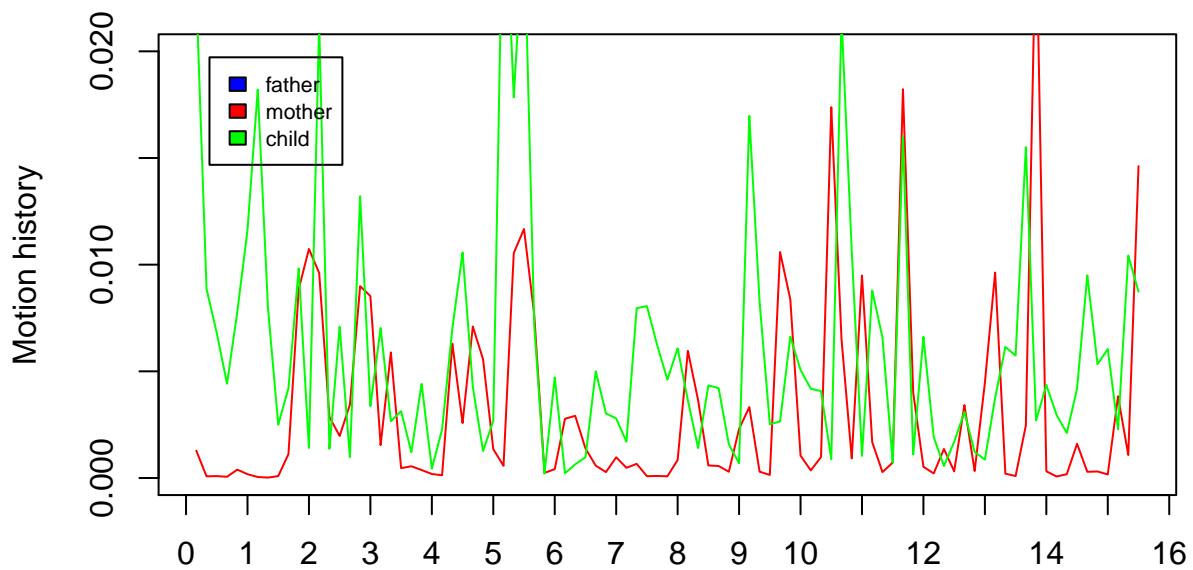
**Mean motion history (non overlapping 10 sec intervals)
on 00041 video**



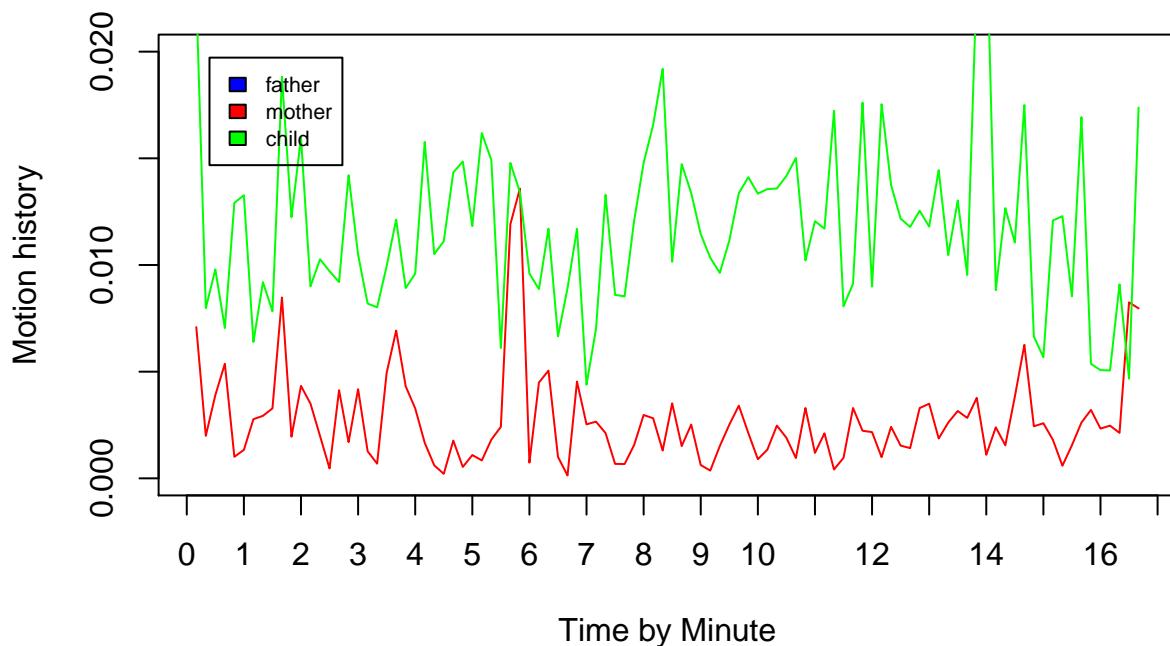
**Time by Minute
Mean motion history (non overlapping 10 sec intervals)
on 00048 video**



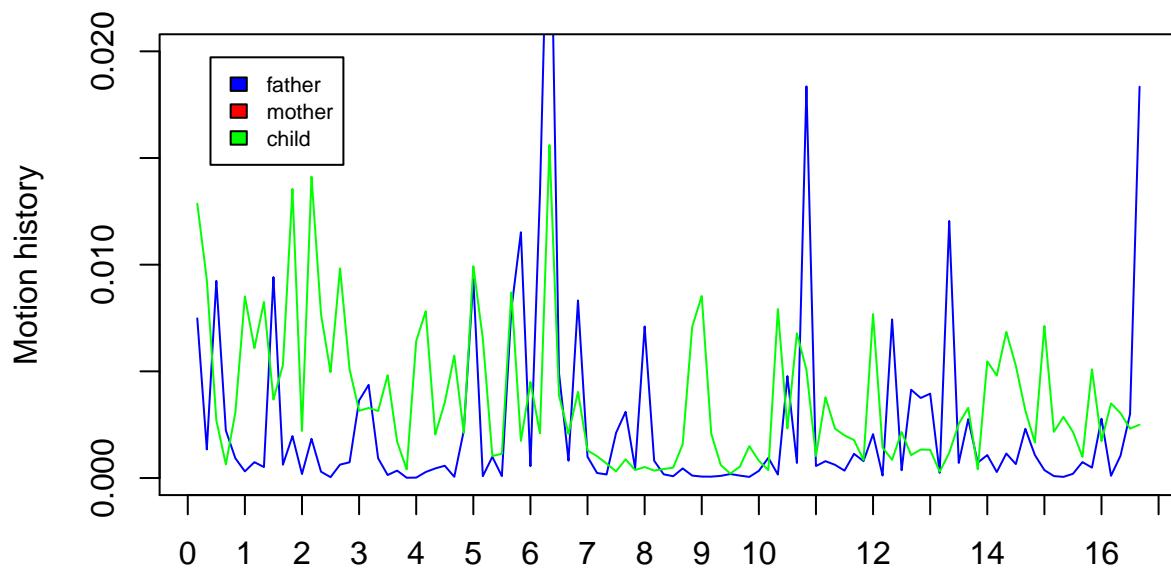
**Mean motion history (non overlapping 10 sec intervals)
on 0206 video**



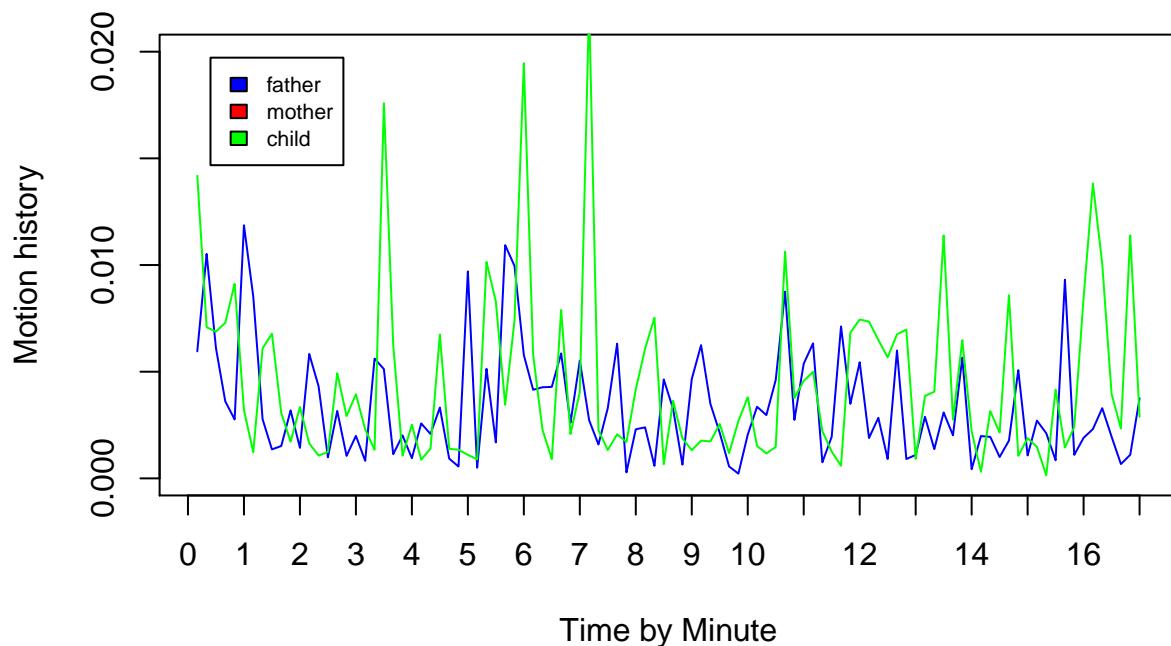
**Time by Minute
Mean motion history (non overlapping 10 sec intervals)
on 1106 video**



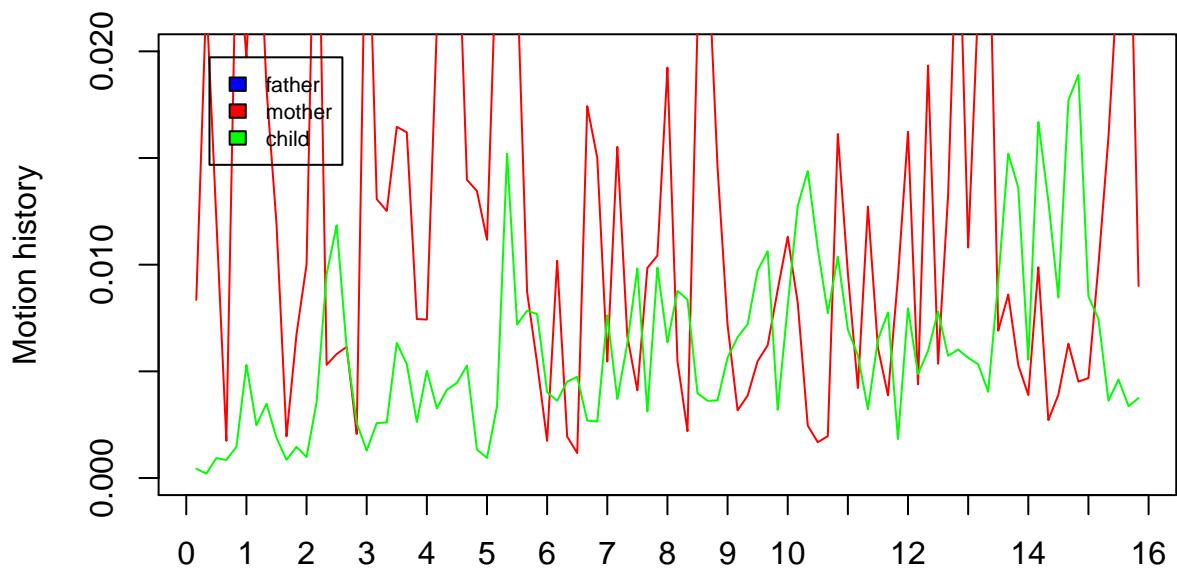
**Mean motion history (non overlapping 10 sec intervals)
on 1606 video**



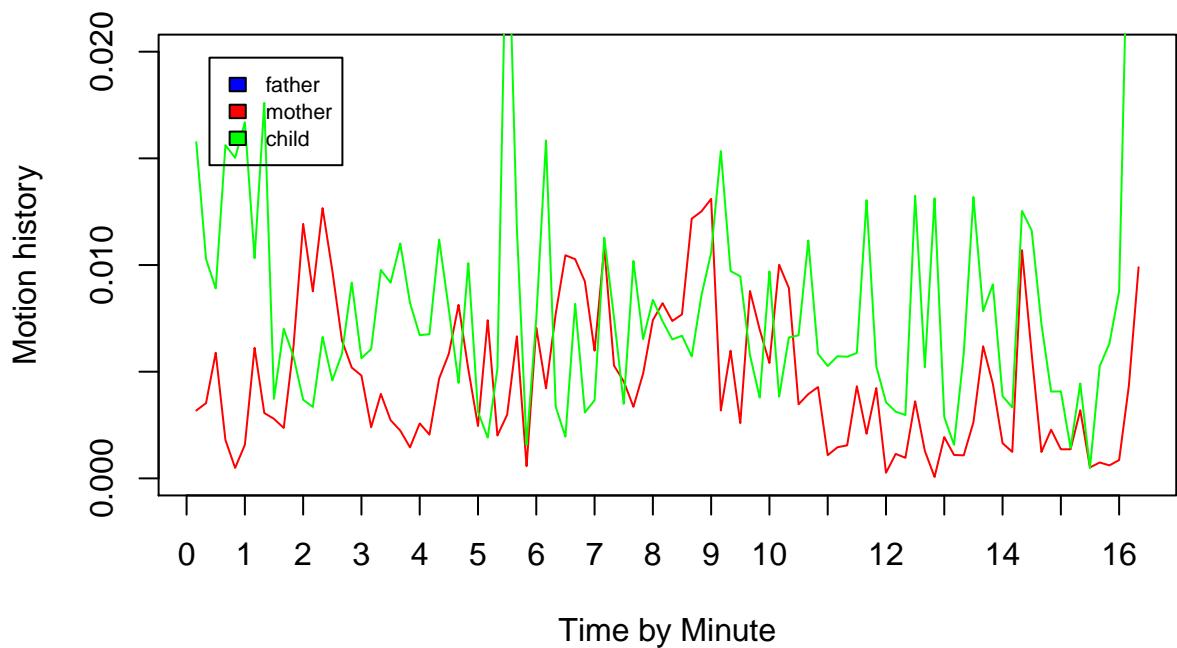
**Time by Minute
Mean motion history (non overlapping 10 sec intervals)
on BAJE059 video**



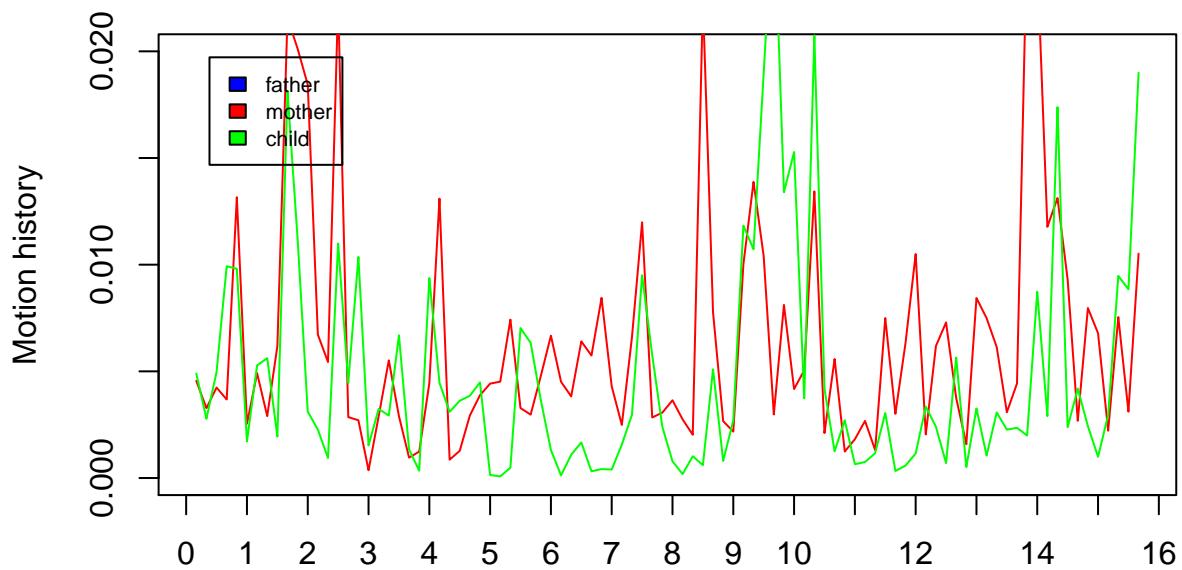
**Mean motion history (non overlapping 10 sec intervals)
on BALE050 video**



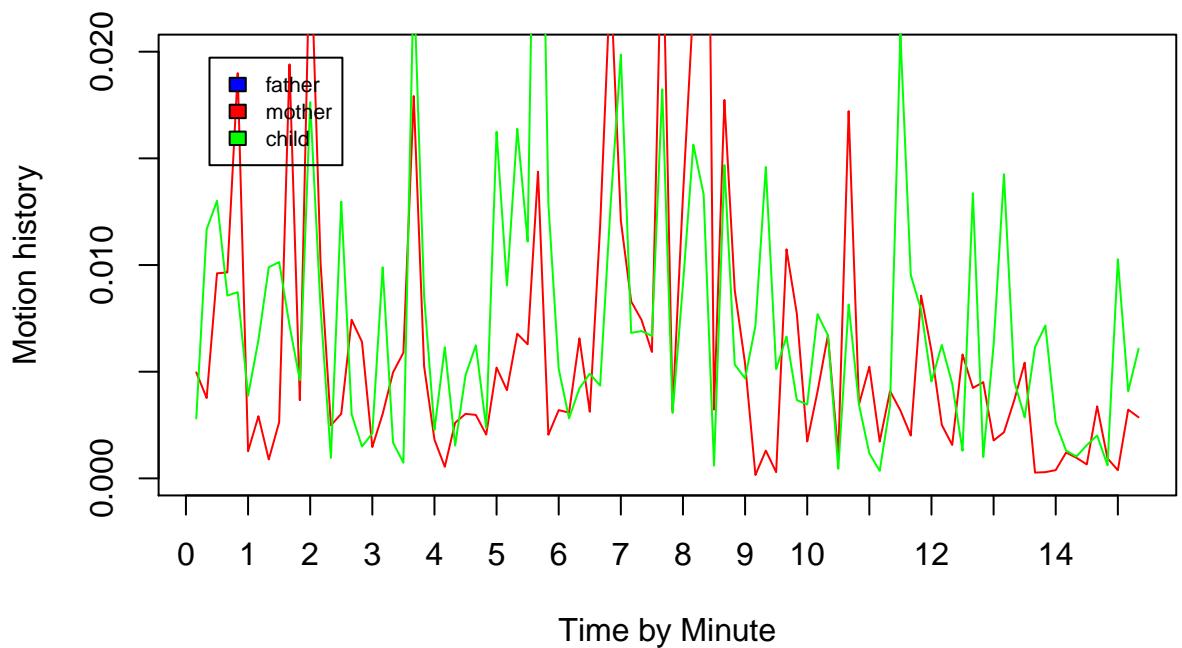
**Time by Minute
Mean motion history (non overlapping 10 sec intervals)
on BALU062 video**



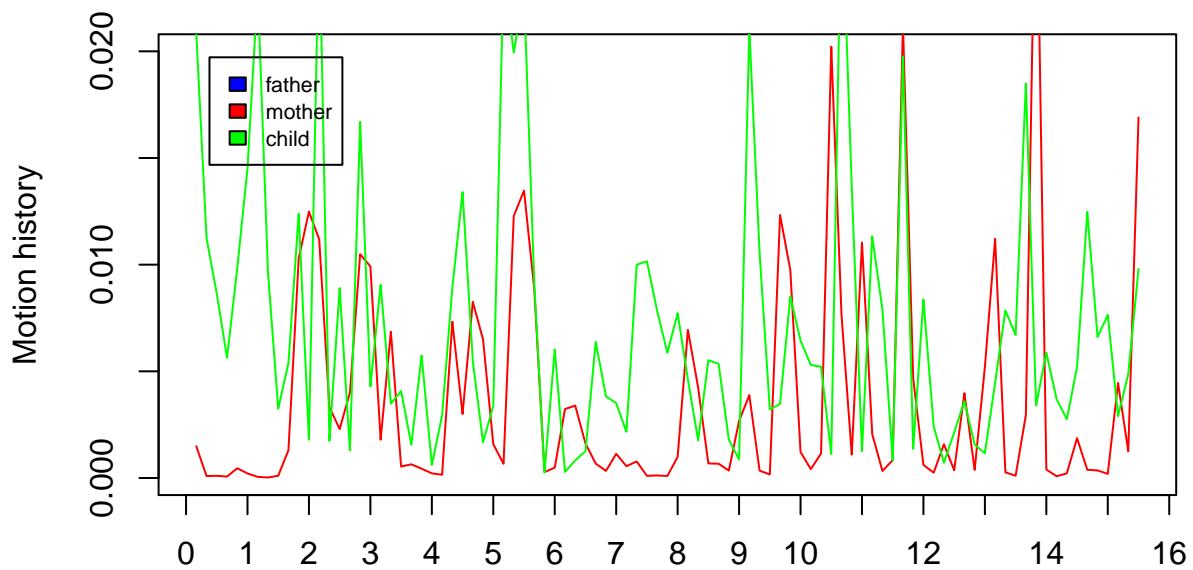
**Mean motion history (non overlapping 10 sec intervals)
on BEAL036 video**



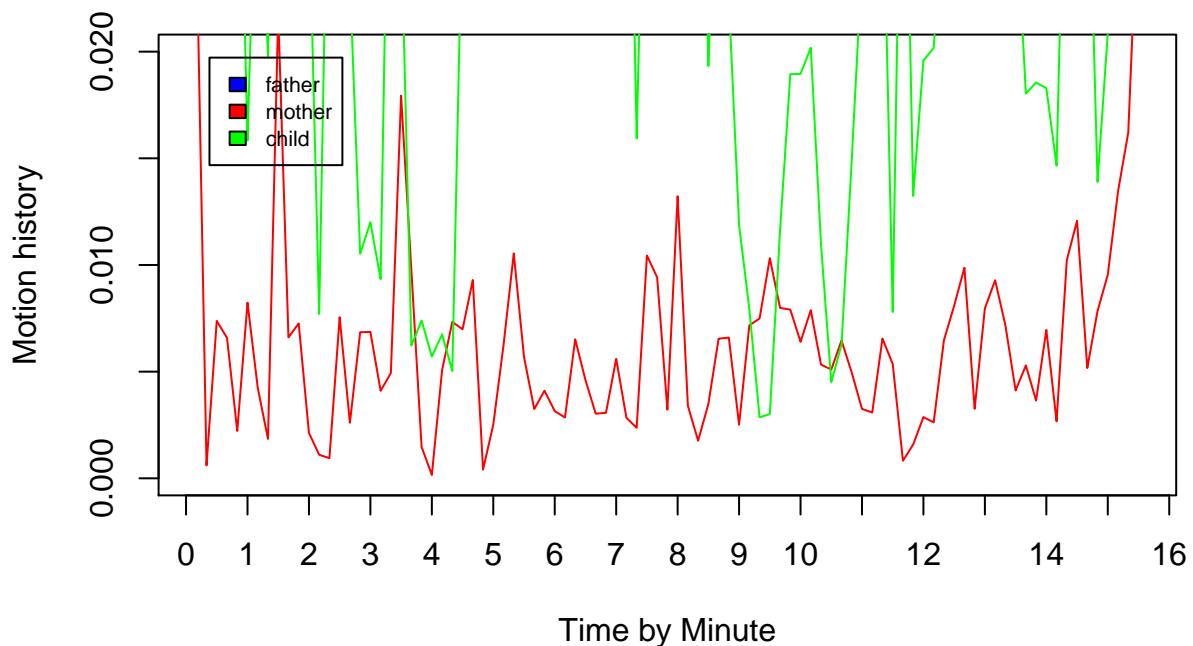
**Time by Minute
Mean motion history (non overlapping 10 sec intervals)
on BEAM031 video**



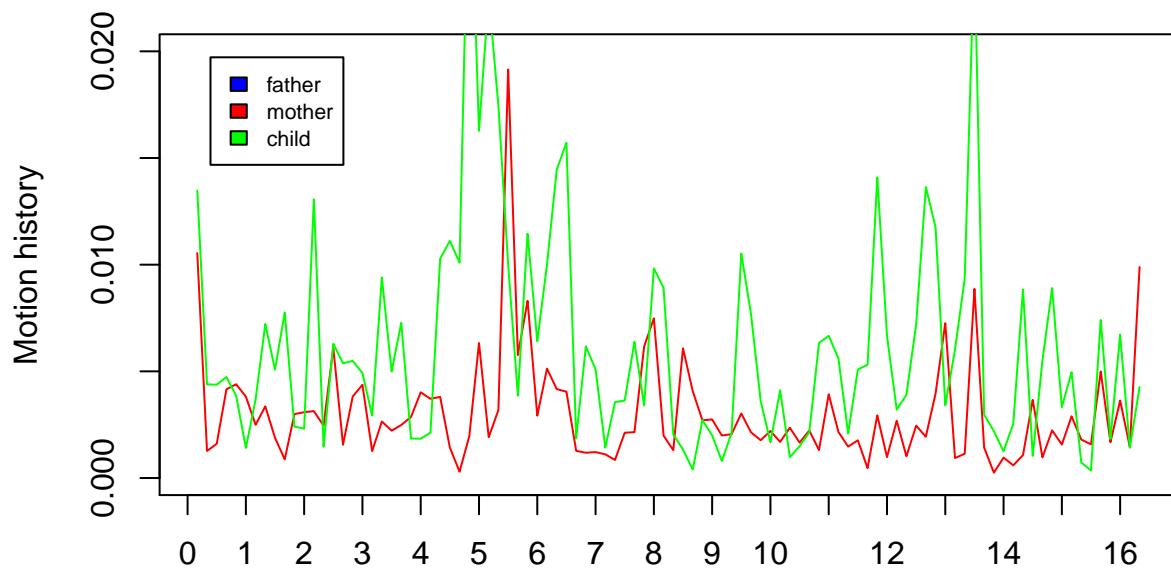
**Mean motion history (non overlapping 10 sec intervals)
on BICA video**



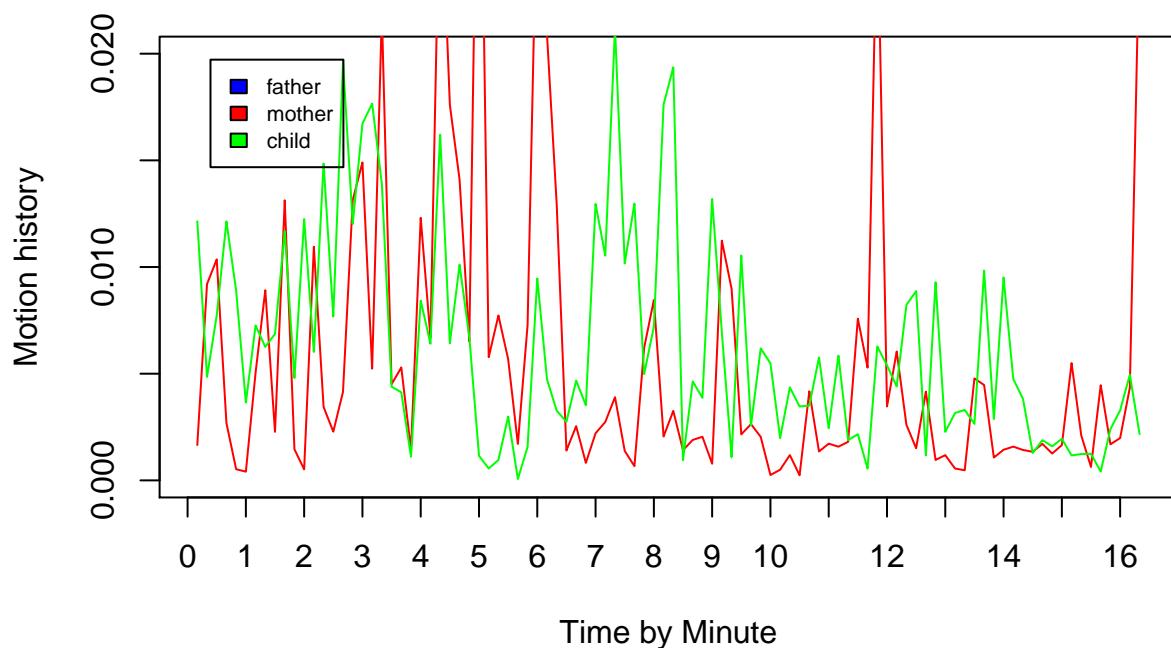
**Time by Minute
Mean motion history (non overlapping 10 sec intervals)
on BRLO041 video**



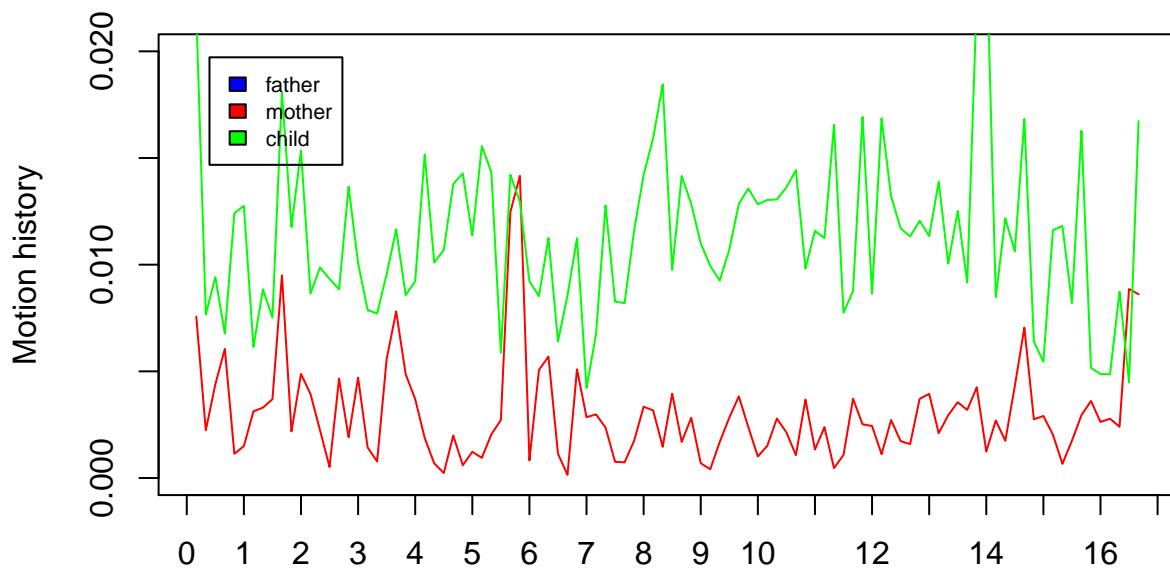
**Mean motion history (non overlapping 10 sec intervals)
on COLO022 video**



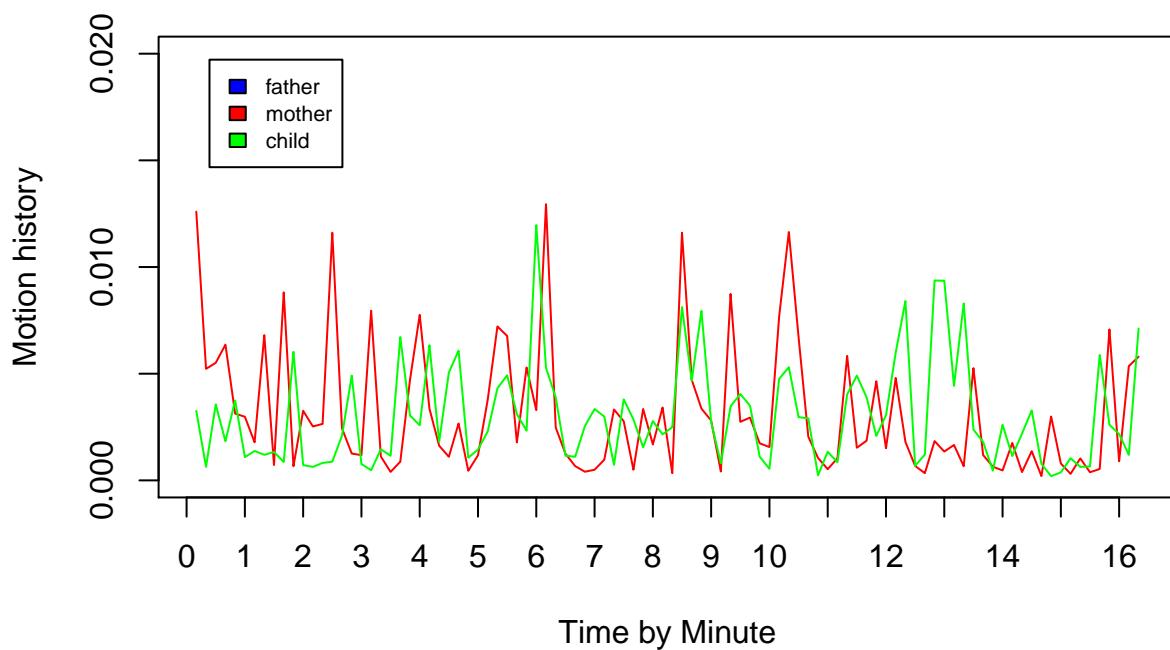
**Time by Minute
Mean motion history (non overlapping 10 sec intervals)
on Dipe004 video**



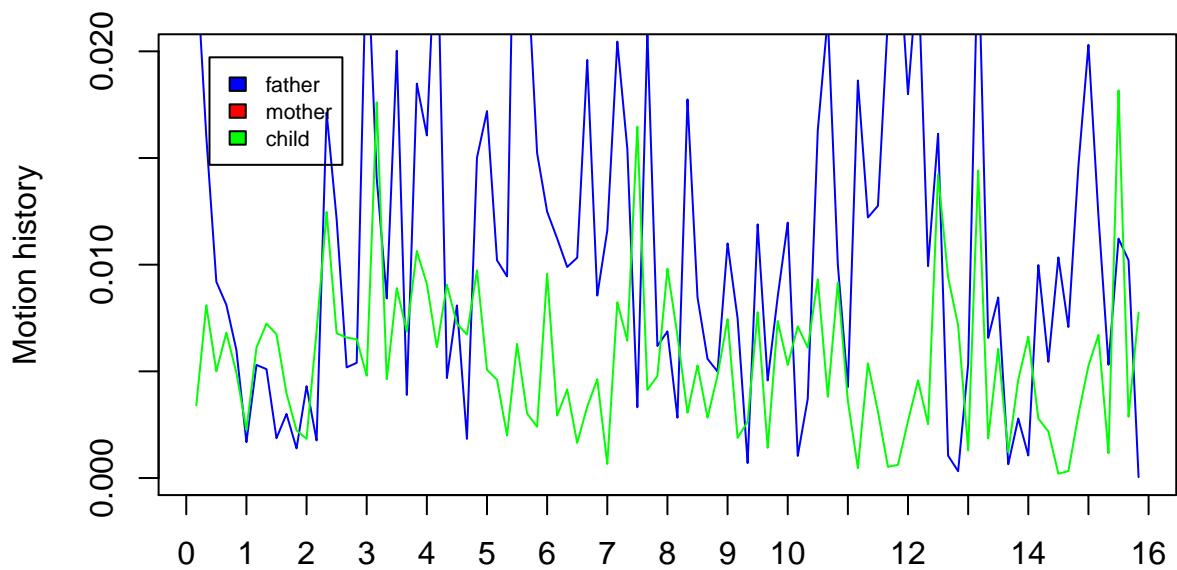
**Mean motion history (non overlapping 10 sec intervals)
on DOMA video**



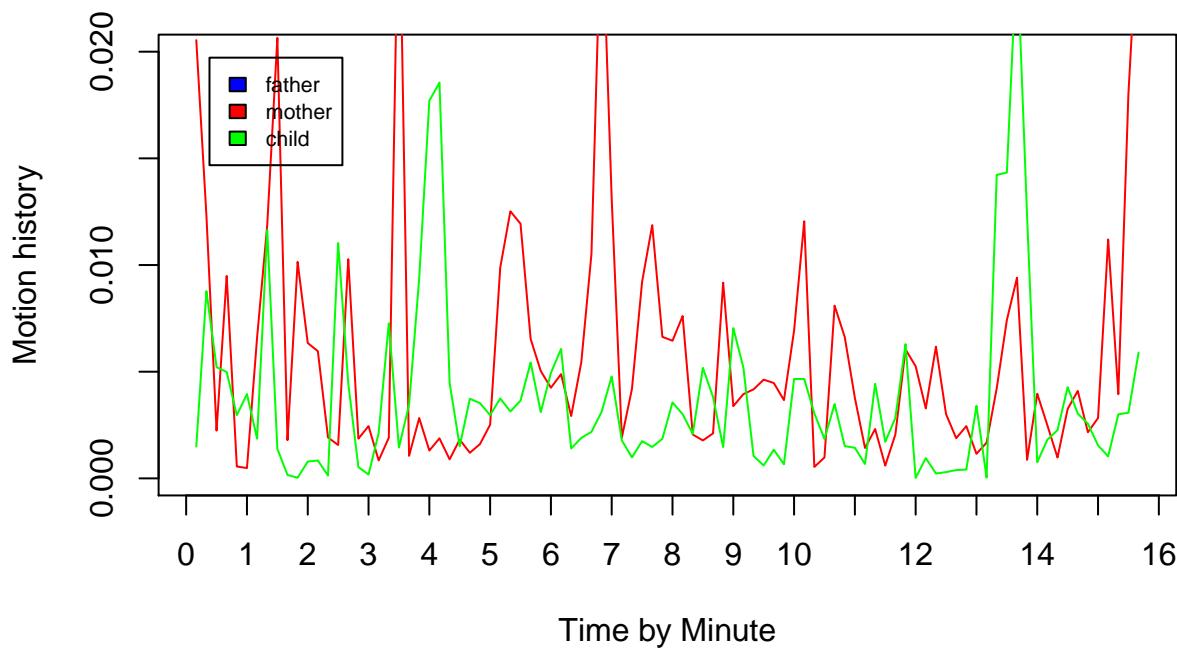
**Time by Minute
Mean motion history (non overlapping 10 sec intervals)
on DRNE video**



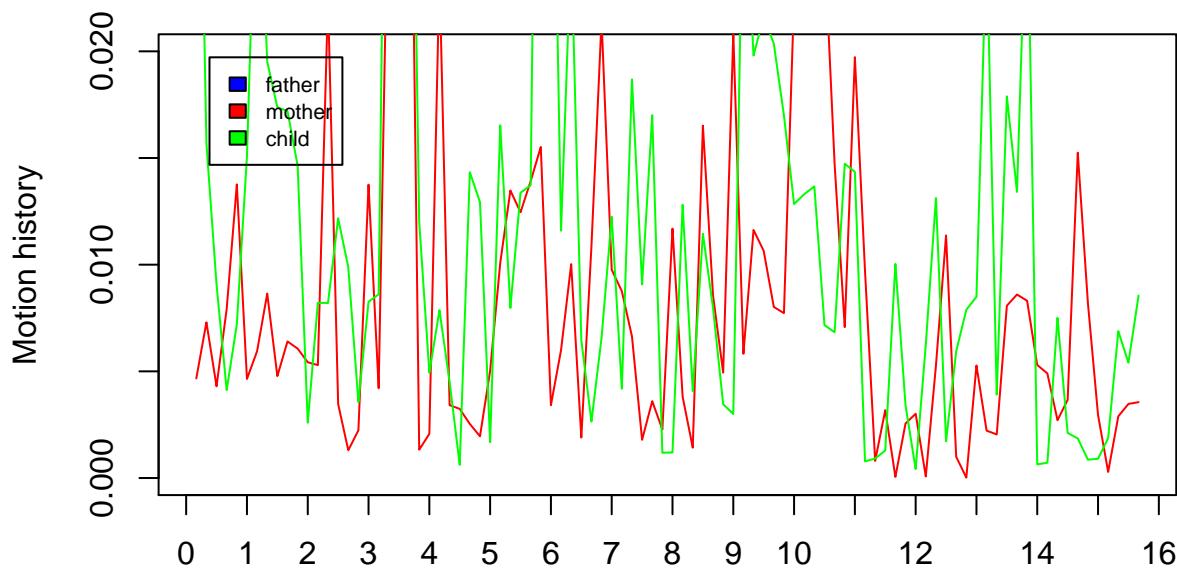
**Mean motion history (non overlapping 10 sec intervals)
on FOMA057 video**



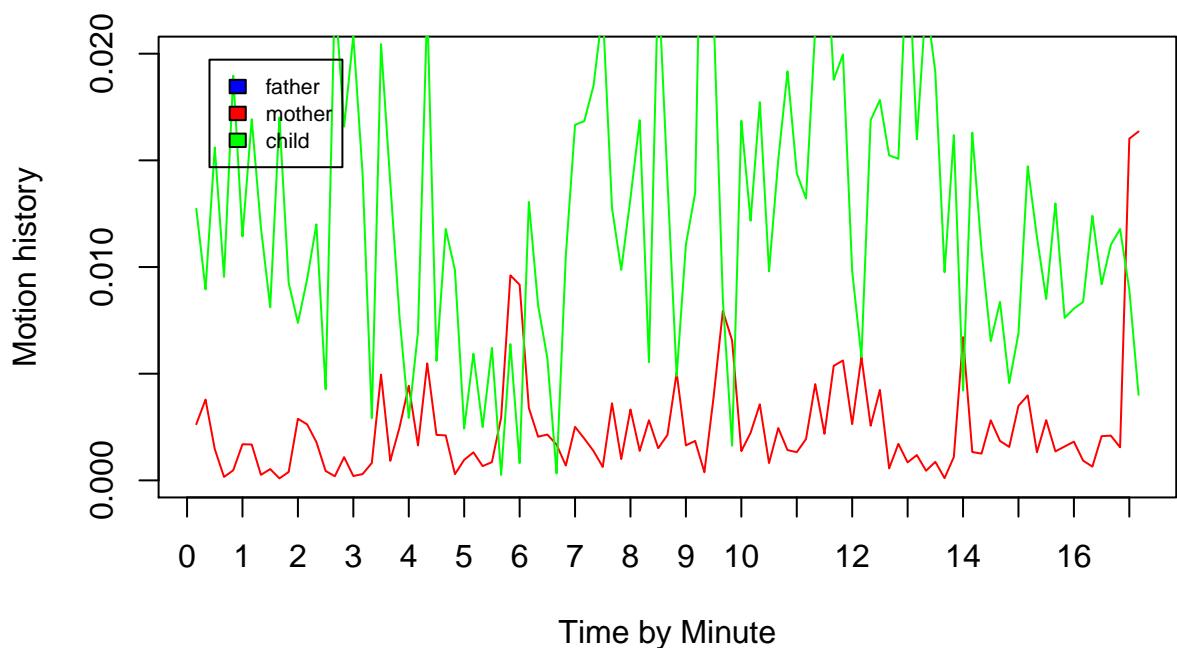
**Time by Minute
Mean motion history (non overlapping 10 sec intervals)
on GROP039 video**



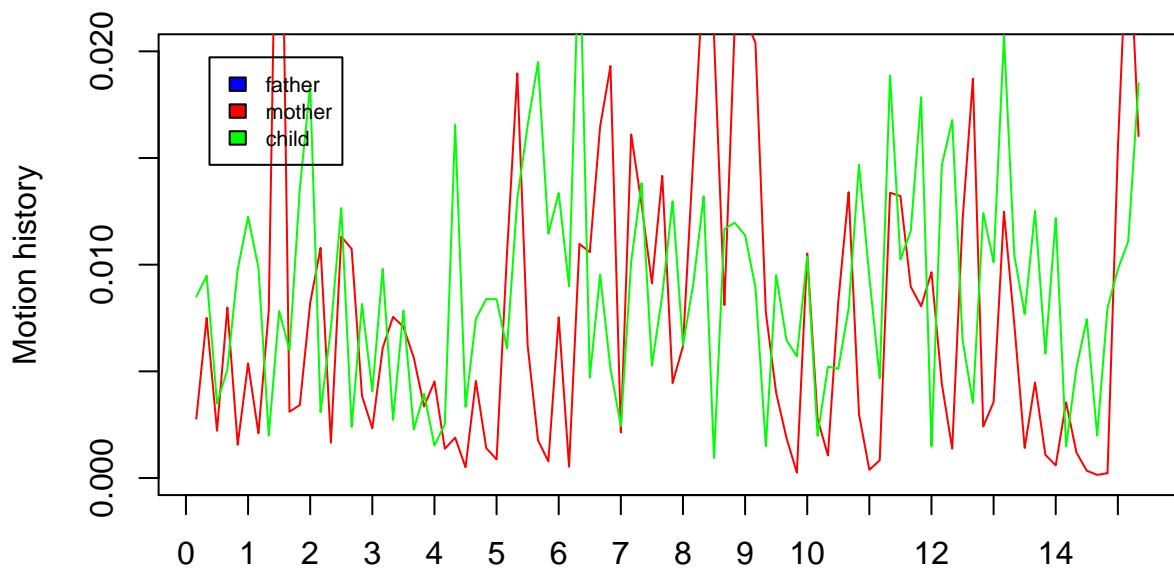
**Mean motion history (non overlapping 10 sec intervals)
on HAJA052 video**



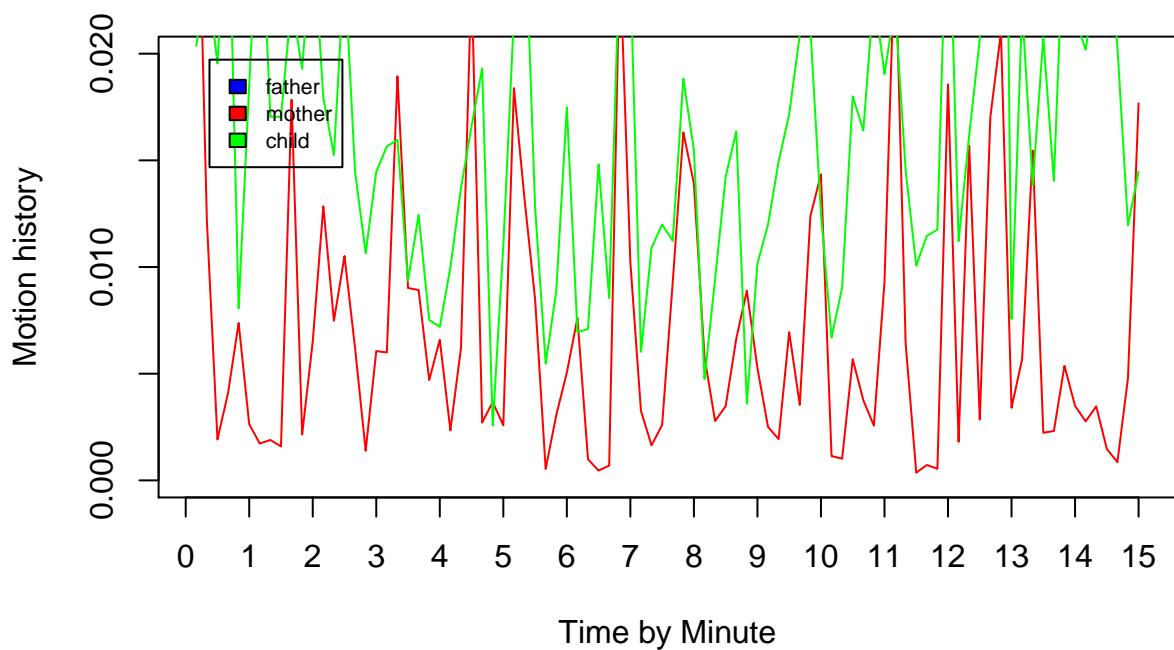
**Time by Minute
Mean motion history (non overlapping 10 sec intervals)
on HUMA058 video**



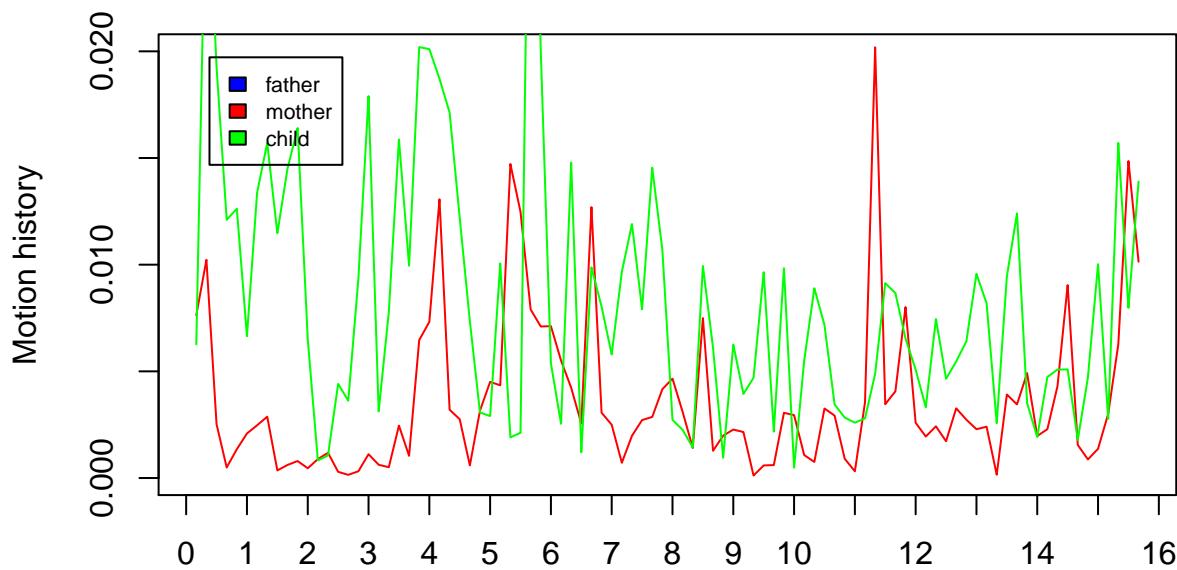
**Mean motion history (non overlapping 10 sec intervals)
on JAEM046 video**



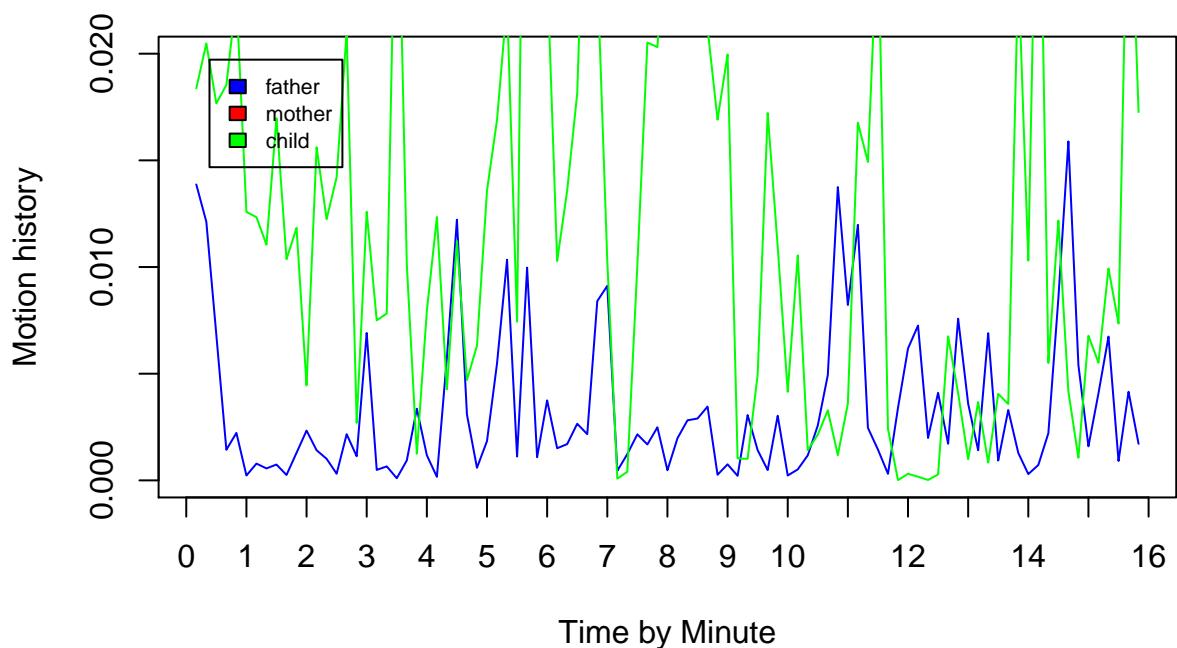
**Time by Minute
Mean motion history (non overlapping 10 sec intervals)
on JEEO040 video**



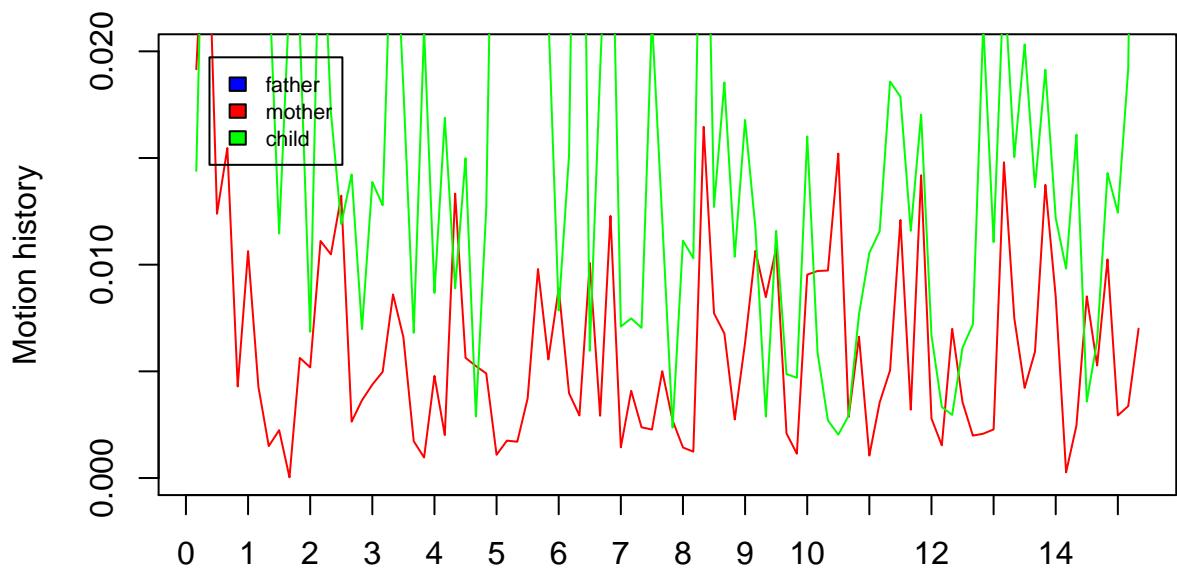
**Mean motion history (non overlapping 10 sec intervals)
on JOCE014 video**



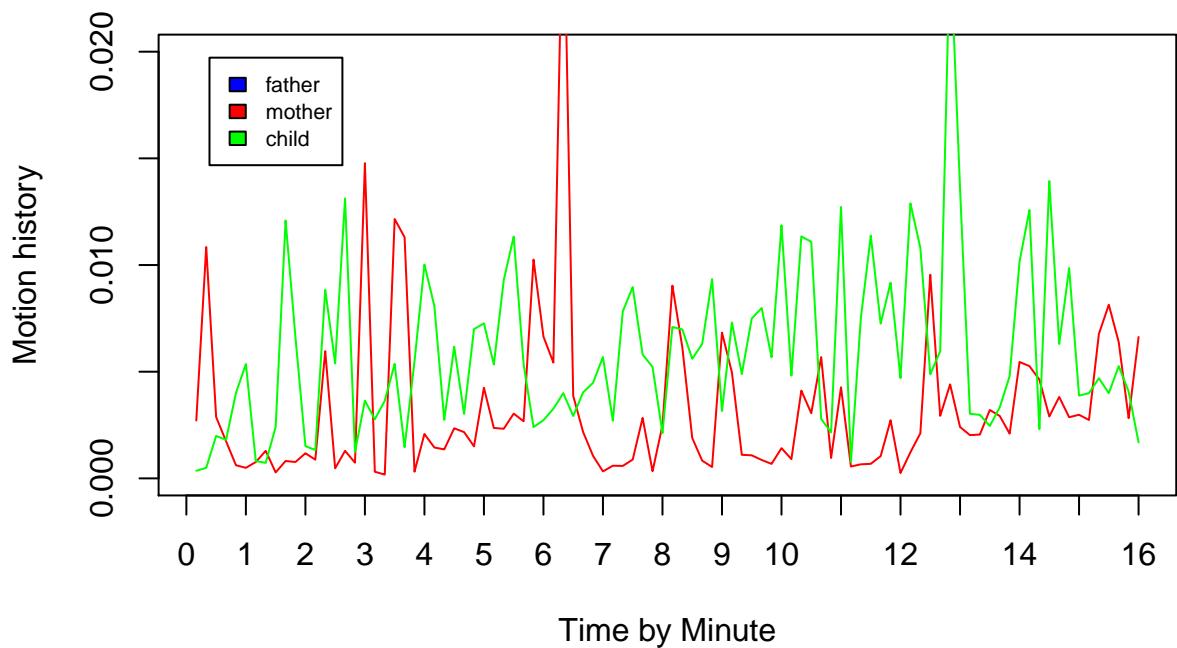
**Time by Minute
Mean motion history (non overlapping 10 sec intervals)
on LACL video**



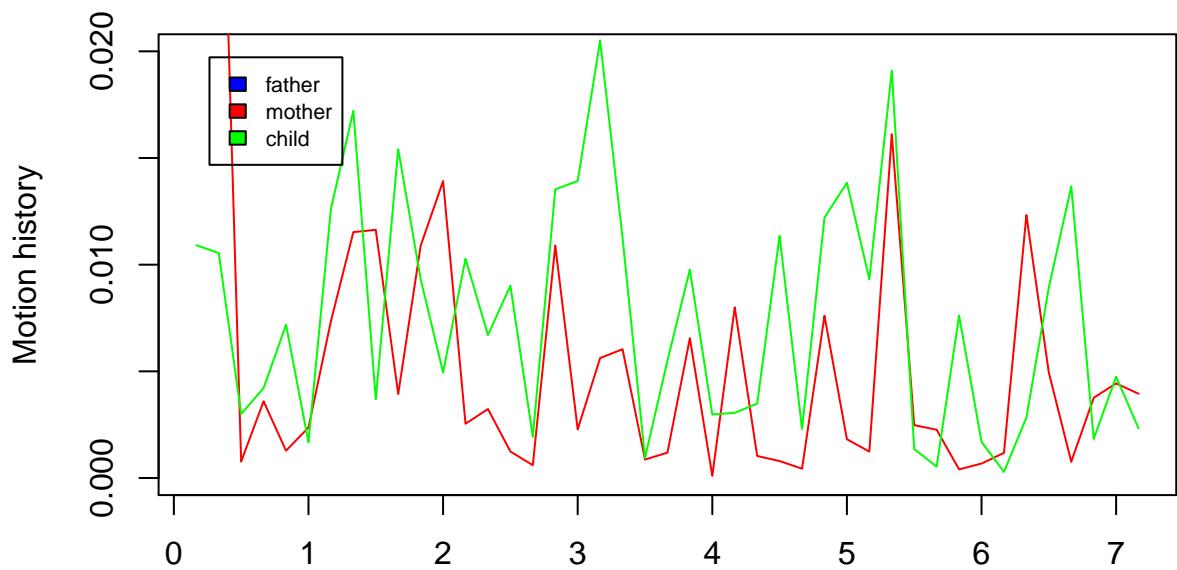
**Mean motion history (non overlapping 10 sec intervals)
on MAEL048 video**



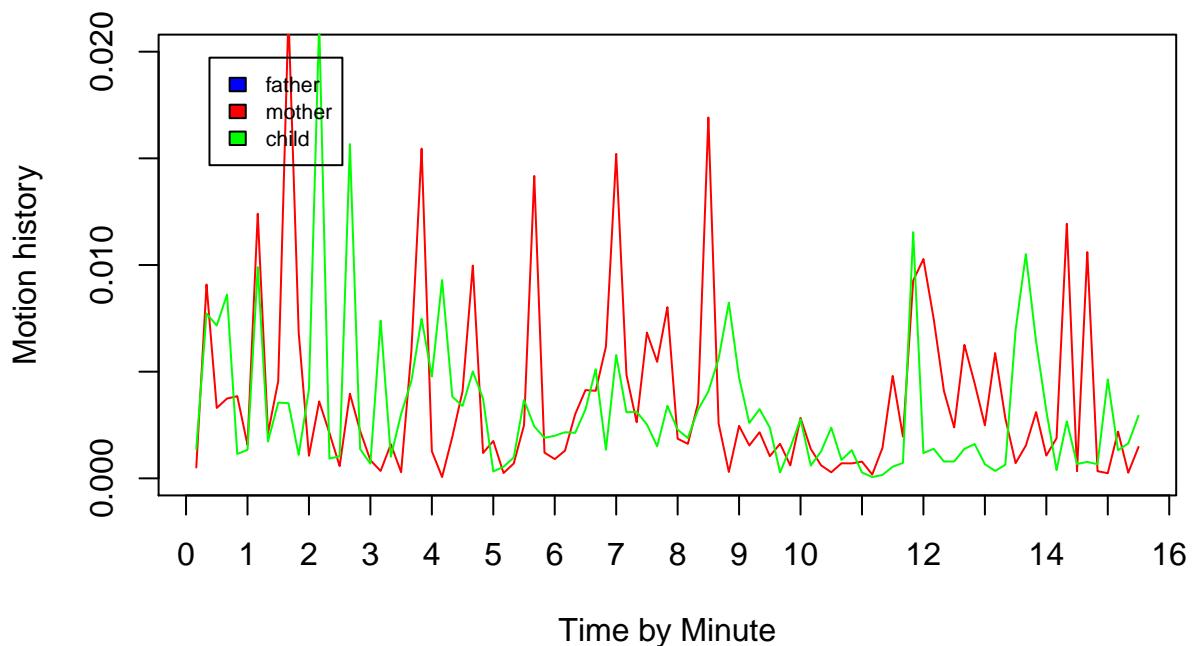
**Mean motion history (non overlapping 10 sec intervals)
on MAME20 video**



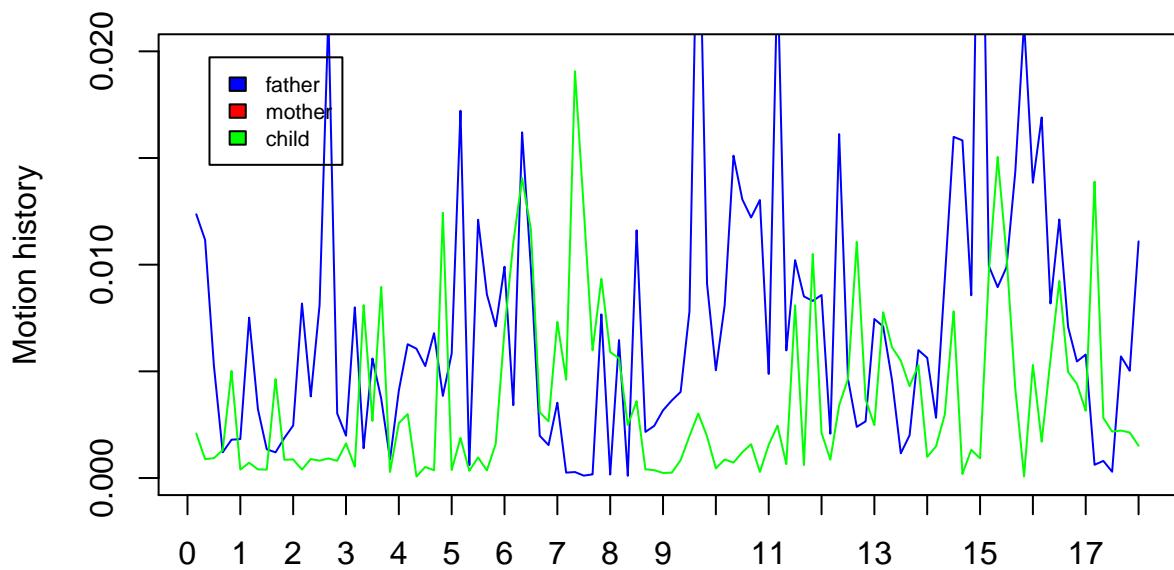
**Mean motion history (non overlapping 10 sec intervals)
on MAPA029 video**



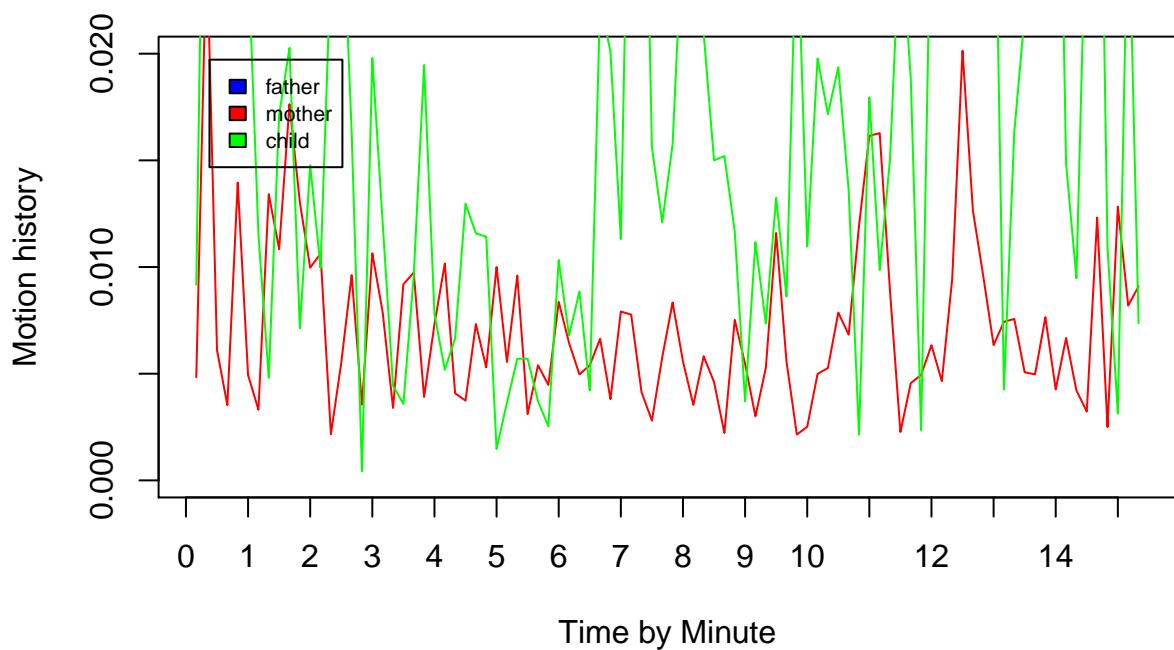
**Time by Minute
Mean motion history (non overlapping 10 sec intervals)
on MIPH043 video**



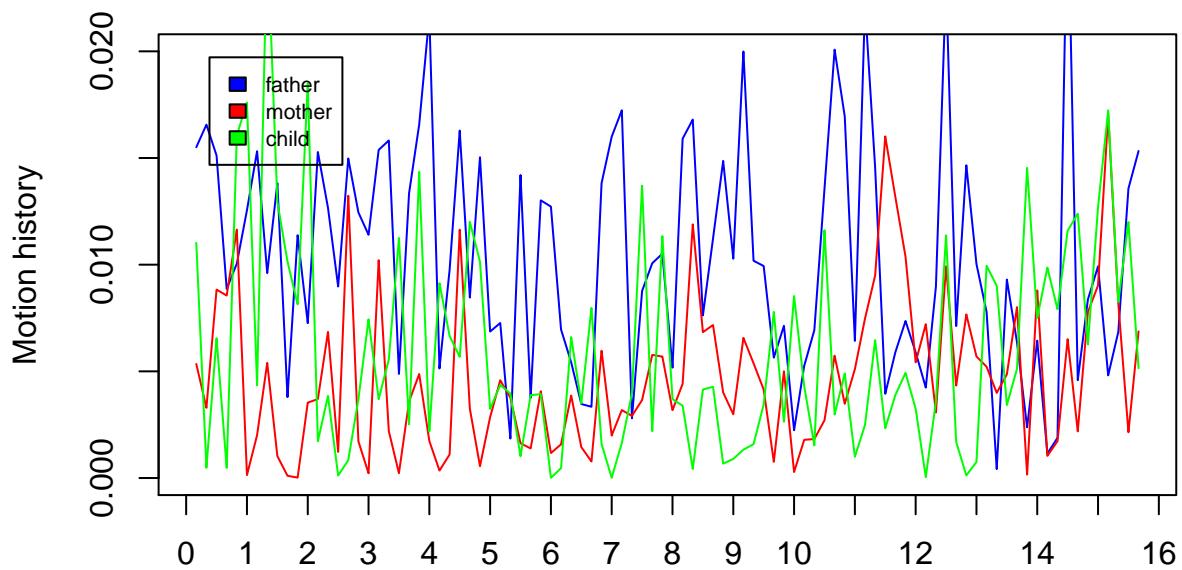
**Mean motion history (non overlapping 10 sec intervals)
on MOSA065 video**



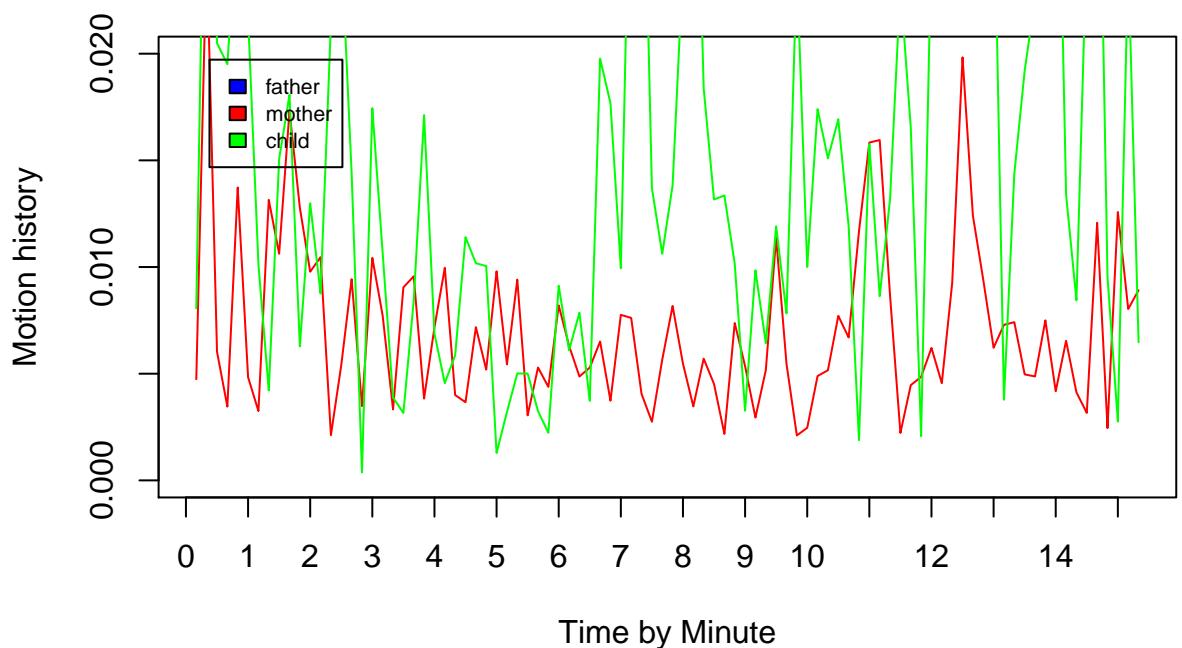
**Mean motion history (non overlapping 10 sec intervals)
on RAEM049 video**



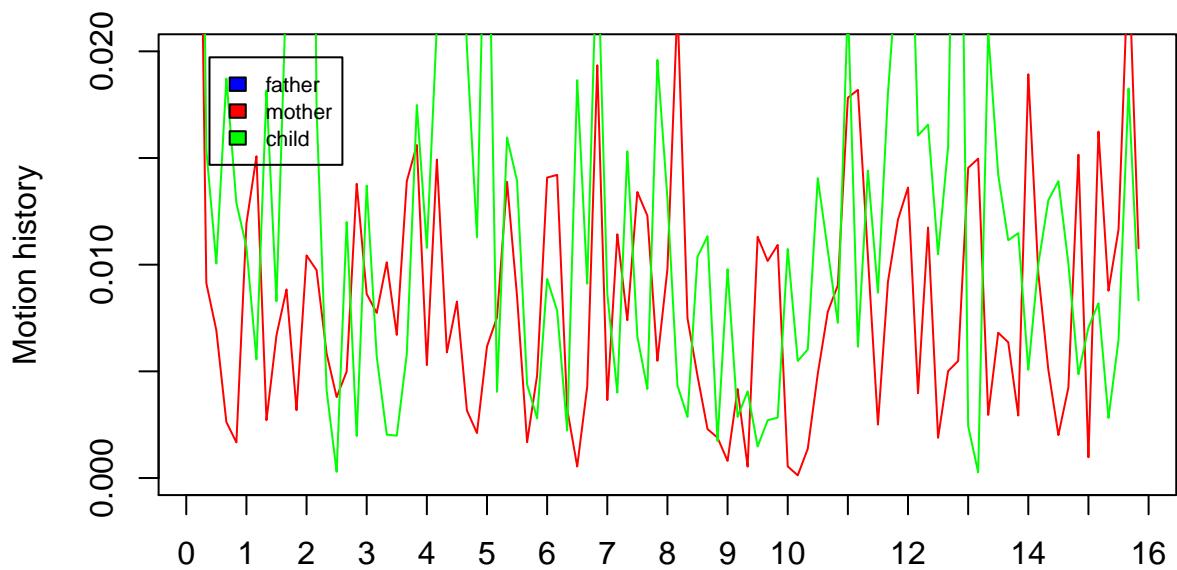
**Mean motion history (non overlapping 10 sec intervals)
on RAKA008 video**



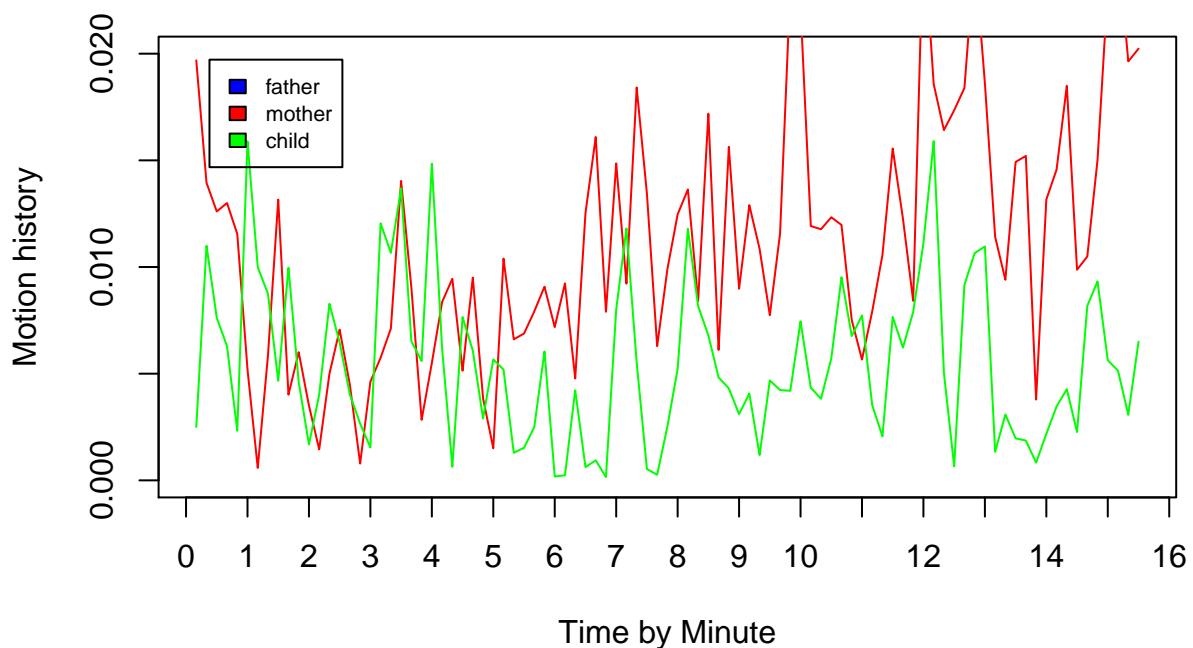
**Time by Minute
Mean motion history (non overlapping 10 sec intervals)
on RIEM0 video**



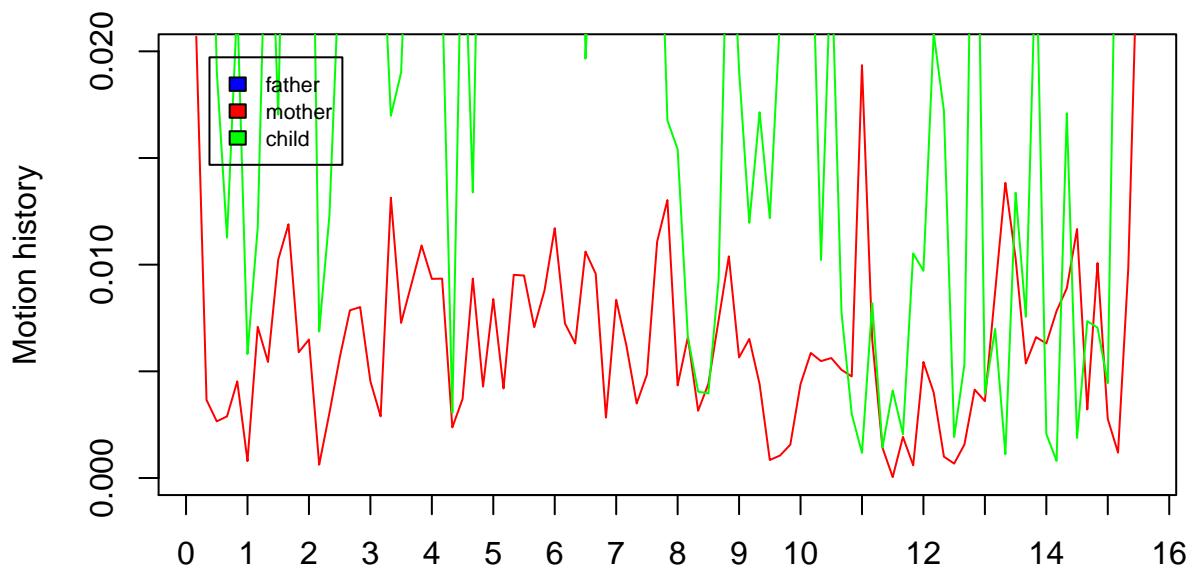
**Mean motion history (non overlapping 10 sec intervals)
on SEEM035 video**



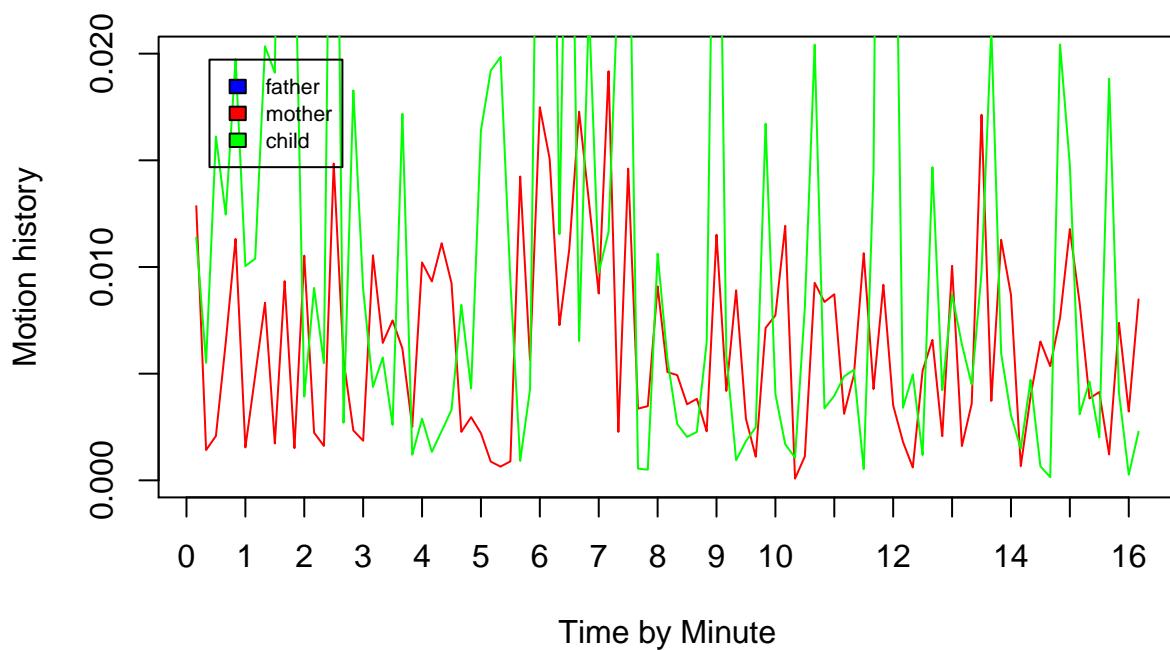
**Time by Minute
Mean motion history (non overlapping 10 sec intervals)
on SHAN042 video**



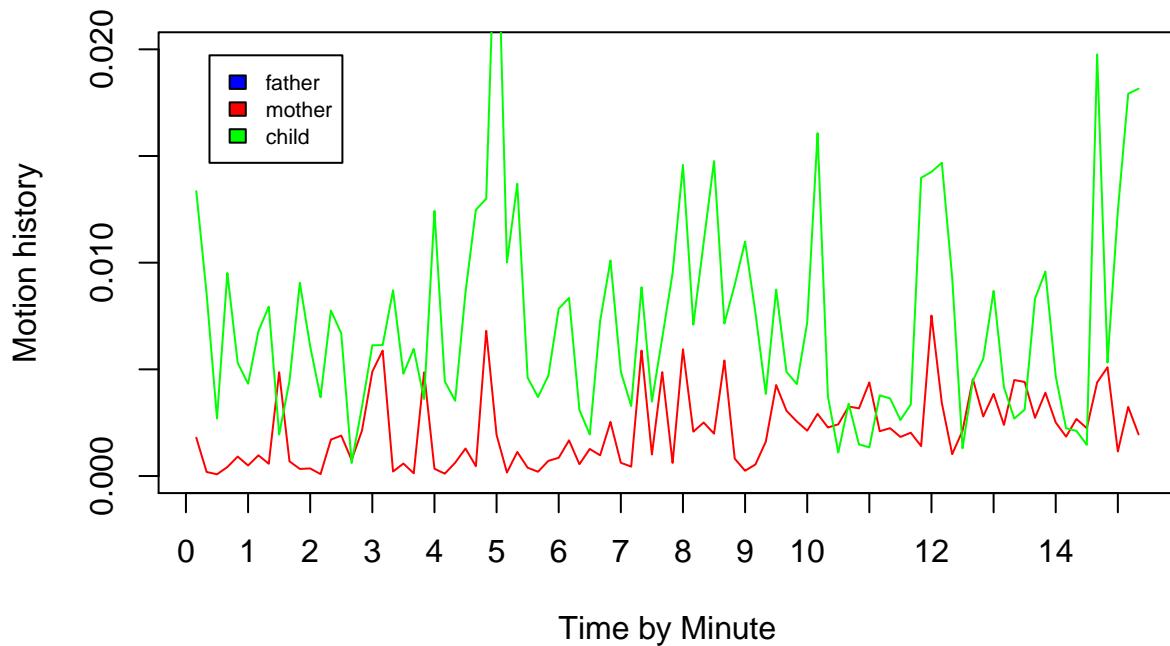
**Mean motion history (non overlapping 10 sec intervals)
on SOGA061 video**



**Time by Minute
Mean motion history (non overlapping 10 sec intervals)
on TIUG032 video**



Mean motion history (non overlapping 10 sec intervals) on VINO video



Export no log data in text files

```

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

  print(paste("Computing滑动父亲", 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滑动母亲", 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滑动孩子", 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", sep=""))
videoIndex <-(videoIndex+1)
}

```

SyncPy utilisation for creating synchrony dataframe

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

And after putting 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

```
print("SSI Files Directory")

## [1] "SSI Files Directory"
SSIlogFilesList <- list.files("../Data/CSV/Synchrony/log/S_estimator", full.name=TRUE)
#SSIlogFilesList

print("SSI Files Directory")

## [1] "SSI Files Directory"
SSInoLogFilesList <- list.files("../Data/CSV/Synchrony/noLog/S_estimator", full.name=TRUE)
#SSInoLogFilesList

SSIlog <- data.frame(video="Name")
for (file in SSIlogFilesList){
  SSIalone <- read.csv(file)
  # print(str(SSIalone))
  SSIlog <- rbind.fill(SSIlog, SSIalone)}
str(SSIlog)

## 'data.frame': 1201 obs. of 8 variables:
## $ video      : chr "Name" "34" "34" "34" ...
## $ X          : int NA 0 1 2 3 4 5 6 7 8 ...
## $ Interval   : int NA 1 2 3 4 5 6 7 8 9 ...
## $ Time_min   : num NA 0 0.5 1 1.5 2 2.5 3 3.5 4 ...
## $ SSI_mo_ch  : num NA 6.91e-03 5.31e-02 2.32e-05 5.74e-04 ...
## $ SSI_fa_ch  : num NA NA NA NA NA NA NA NA NA ...
## $ SSI_fa_mo  : num NA NA NA NA NA NA NA NA NA ...
## $ SSI_fa_mo_ch: num NA NA NA NA NA NA NA NA NA ...

SSIlog$video <- as.factor(SSIlog$video)
SSIlog <- SSIlog[-which(SSIlog$video=="Name"),]

SSInoLog <- data.frame(video="Name")
for (file in SSInoLogFilesList){
# print(file)
  SSIalone <- read.csv(file)
  # print(str(SSIalone))
```

```

SSIInoLog<- rbind.fill(SSIInoLog, SSIalone)
SSIInoLog$video <- as.factor(SSIInoLog$video)
SSIInoLog <- SSIInoLog[-which(SSIInoLog$video=="Name"),]
#SSIInoLog$action <- rep(NA, nrow(SSIInoLog))
#SSIInoLog[which(video == & Time_min==)]$action

```

Description of SSIlog data frame

```
str(SSIlog)
```

```

## 'data.frame': 1200 obs. of 8 variables:
## $ video : Factor w/ 40 levels "1106","1606",...: 4 4 4 4 4 4 4 4 4 ...
## $ 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 0.5 1 1.5 2 2.5 3 3.5 4 4.5 ...
## $ SSI_mo_ch : num 6.91e-03 5.31e-02 2.32e-05 5.74e-04 1.88e-02 ...
## $ SSI_fa_ch : num NA NA NA NA NA NA NA NA NA ...
## $ SSI_fa_mo : num NA NA NA NA NA NA NA NA NA ...
## $ SSI_fa_mo_ch: num NA NA NA NA NA NA NA NA NA ...

```

```
#View(SSI)
```

Description of noLogSSI data frame

```
str(SSIInoLog)
```

```

## 'data.frame': 1200 obs. of 8 variables:
## $ video : Factor w/ 40 levels "1106","1606",...: 4 4 4 4 4 4 4 4 4 ...
## $ 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 0.5 1 1.5 2 2.5 3 3.5 4 4.5 ...
## $ SSI_mo_ch : num 0.018627 0.024666 0.04506 0.001527 0.000817 ...
## $ SSI_fa_ch : num NA NA NA NA NA NA NA NA ...
## $ SSI_fa_mo : num NA NA NA NA NA NA NA NA ...
## $ SSI_fa_mo_ch: num NA NA NA NA NA NA NA NA ...

```

```
#View(SSIInoLog)
```

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

```

par(mar=c(4,4,4,3), mfrow=c(1,1))
for (i in unique(SSIlog$video)){
  if (all(!is.na(SSIlog[which(SSIlog$video==i),]$SSI_mo_ch)==TRUE)){
    #print(SSIlog[which(SSIlog$video==i),]$Time_min)
    #print(str(SSIlog[which(SSIlog$video==i),]$SSI_mo_ch))
    plot(SSIlog[which(SSIlog$video==i),]$Time_min, SSIlog[which(SSIlog$video==i),]$SSI_mo_ch,
         ylim=c(0, 0.3), main=paste("Synchrony scores in", i, "video"), xlab = "Time (minute)")

  else if(all(!is.na(SSIlog[which(SSIlog$video==i),]$SSI_fa_ch)==TRUE)){

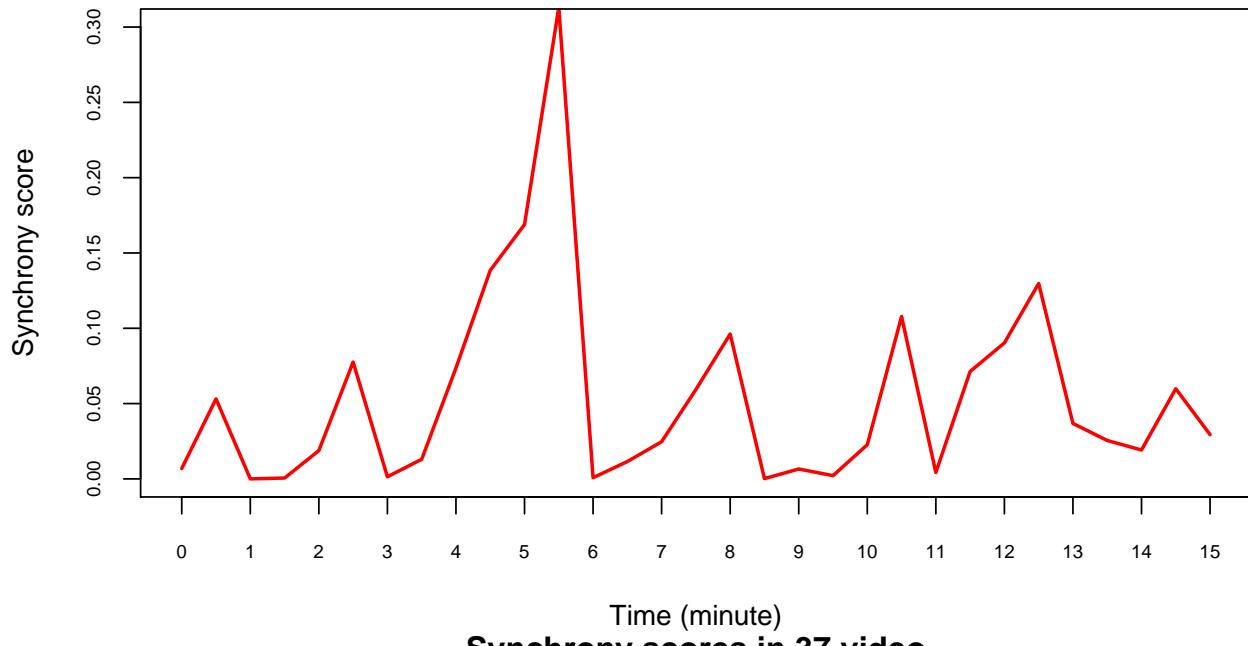
```

```

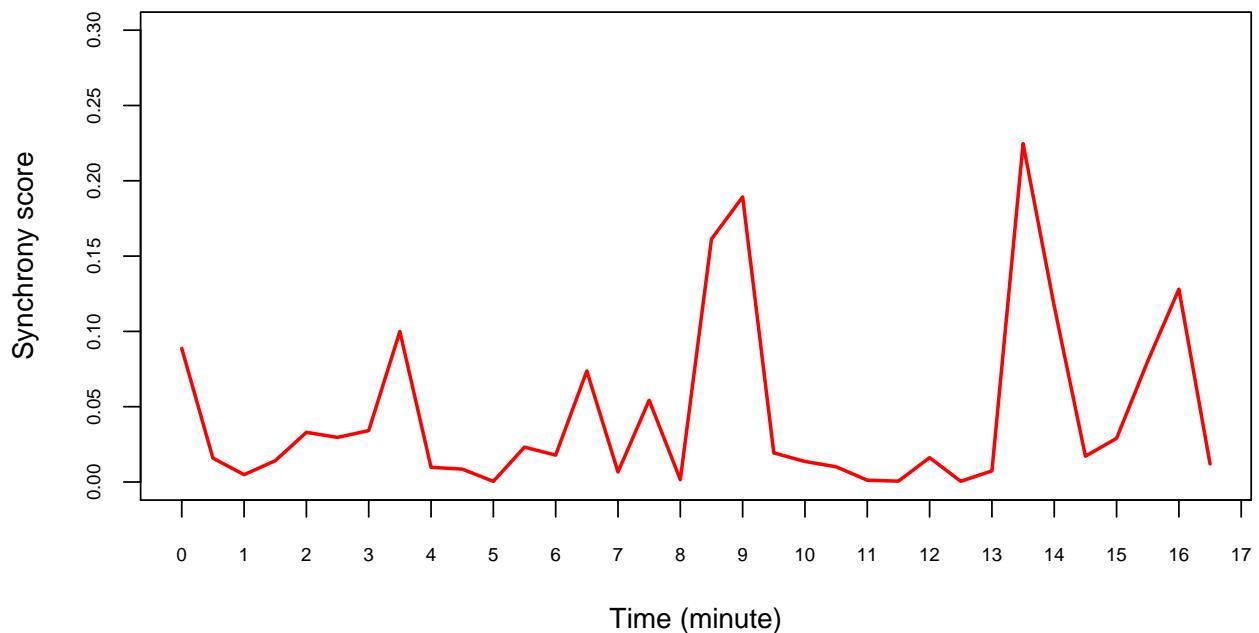
plot(SSIlog[which(SSIlog$video==i),]$Time_min, SSIlog[which(SSIlog$video==i),]$SSI_fa_ch,
      ylim=c(0, 0.3), main=paste("Synchrony scores in", i, "video"), xlab = "Time (minute)", yl
else{print("error")})

```

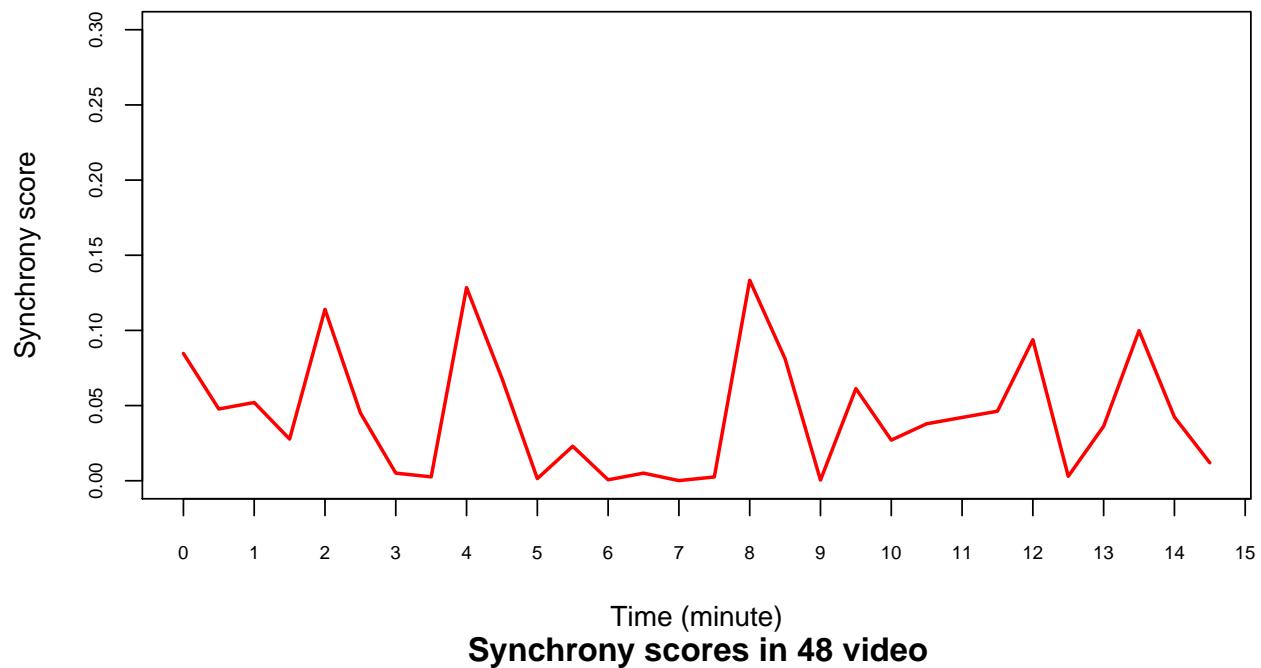
Synchrony scores in 34 video



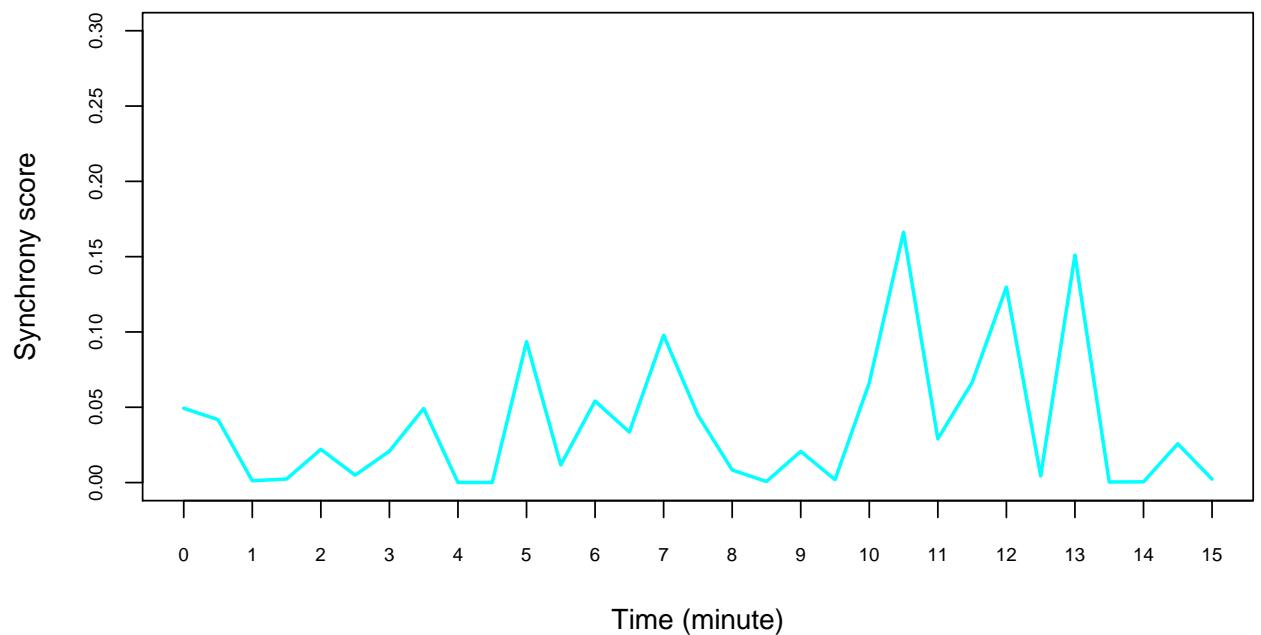
Synchrony scores in 37 video



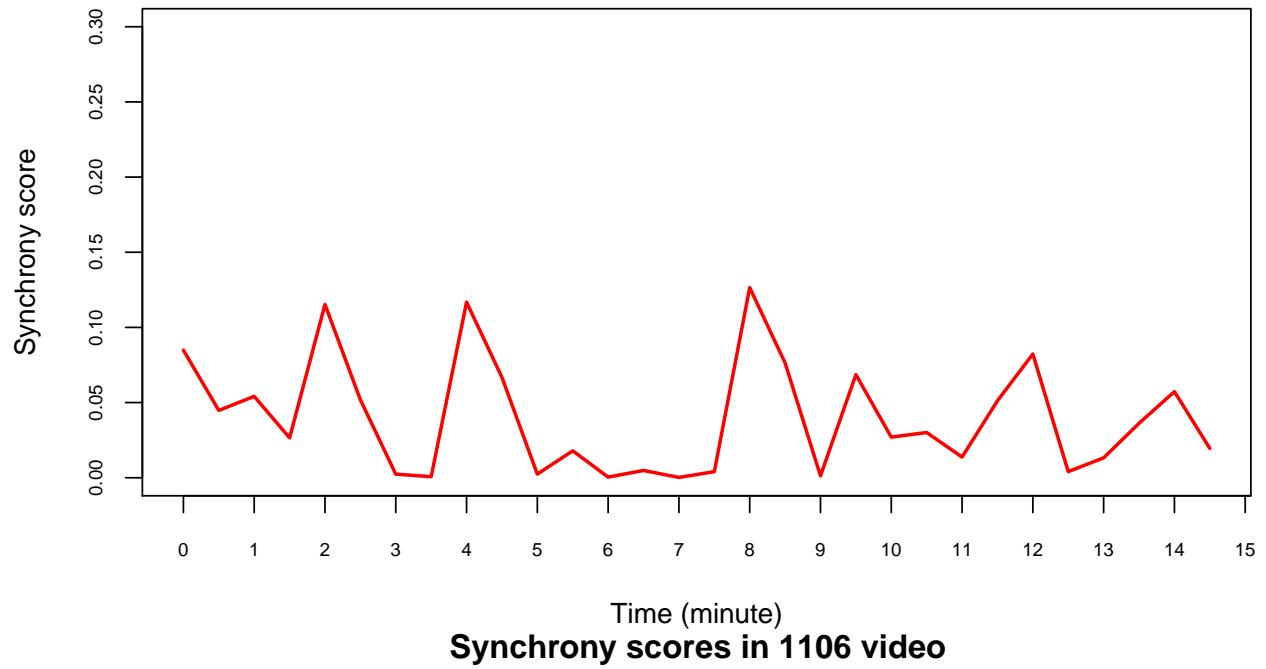
Synchrony scores in 41 video



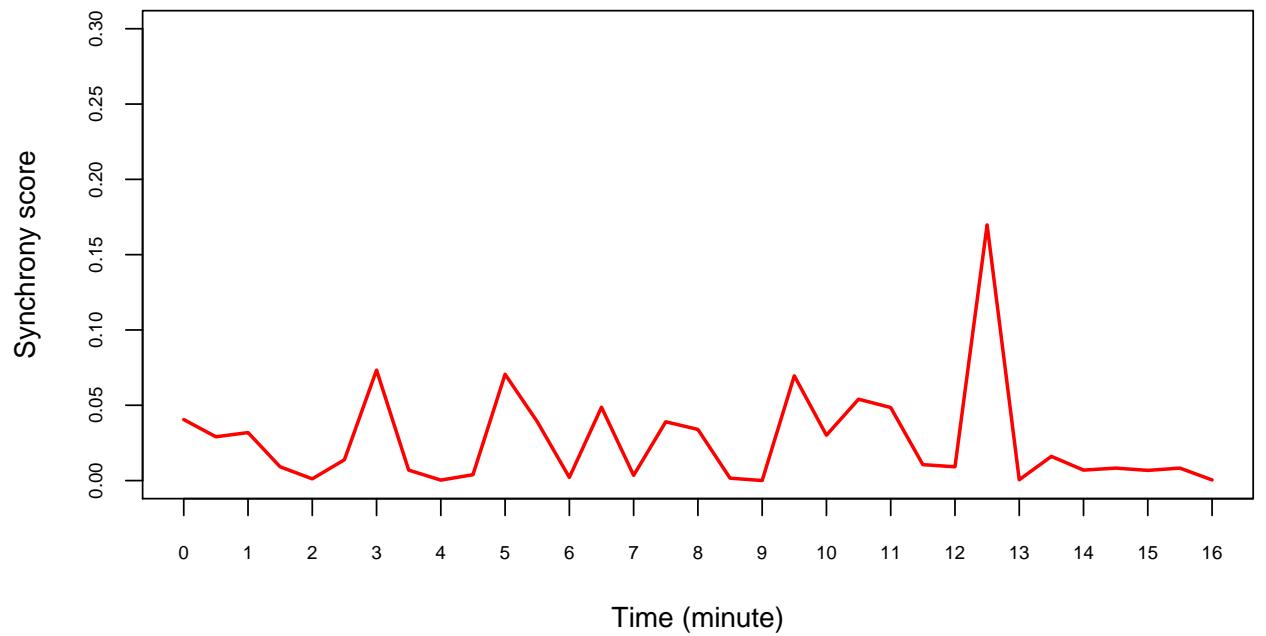
Synchrony scores in 48 video



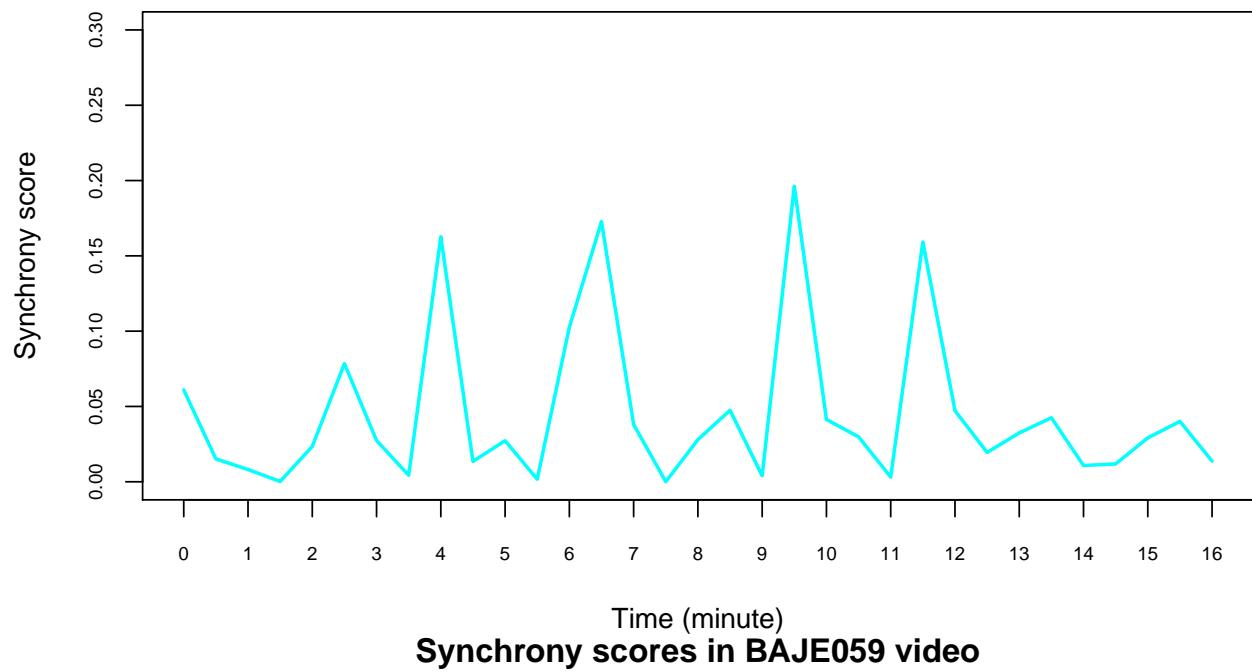
Synchrony scores in 206 video



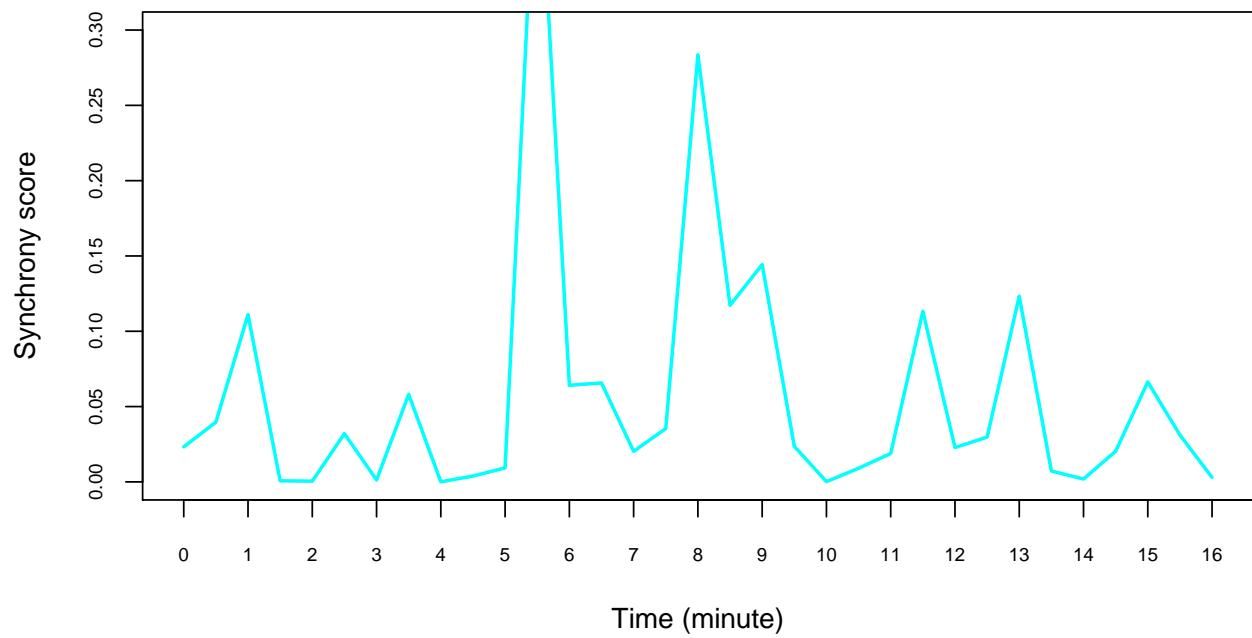
Synchrony scores in 1106 video



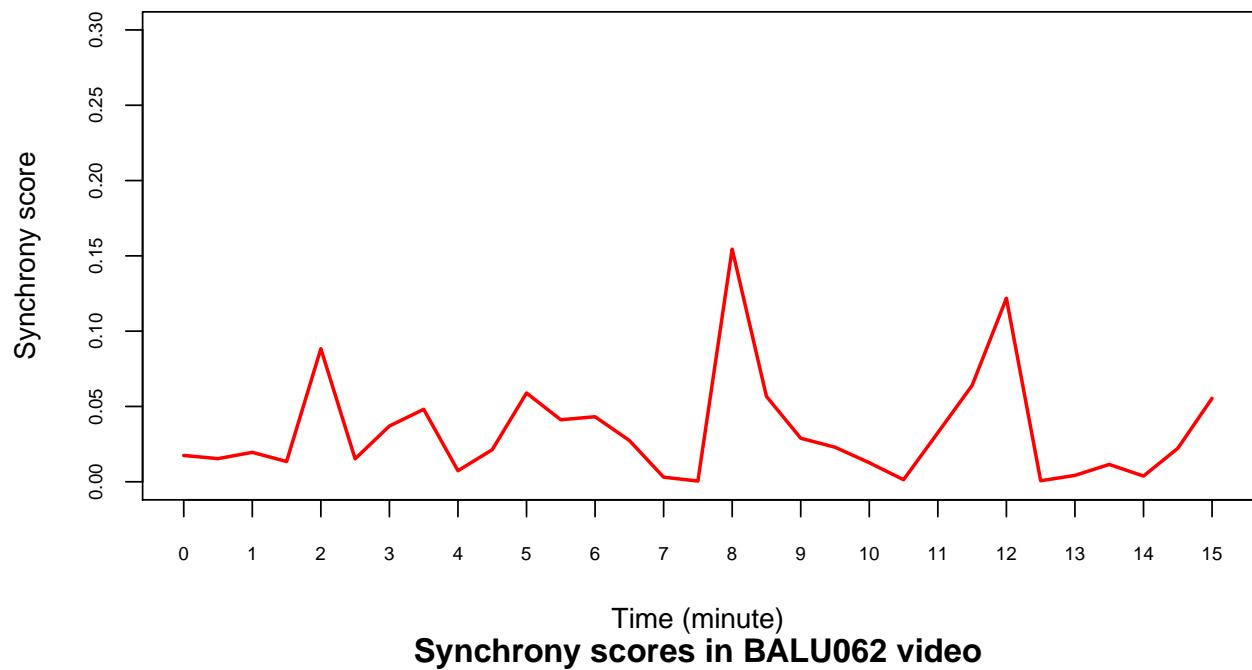
Synchrony scores in 1606 video



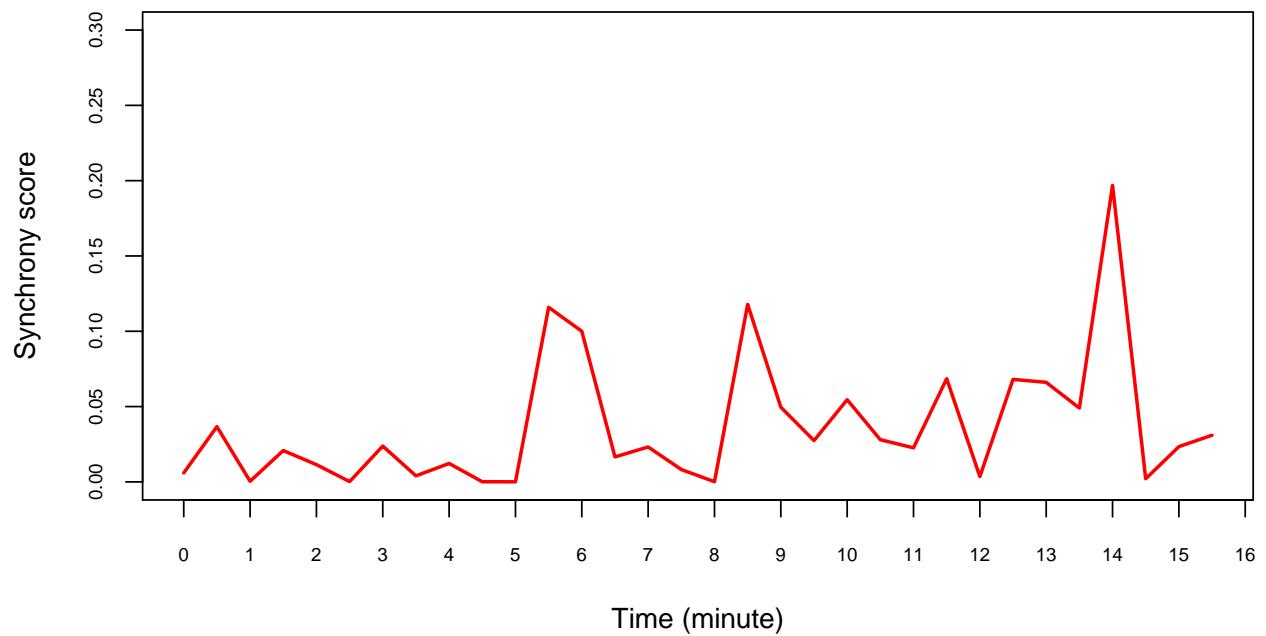
Synchrony scores in BAJE059 video



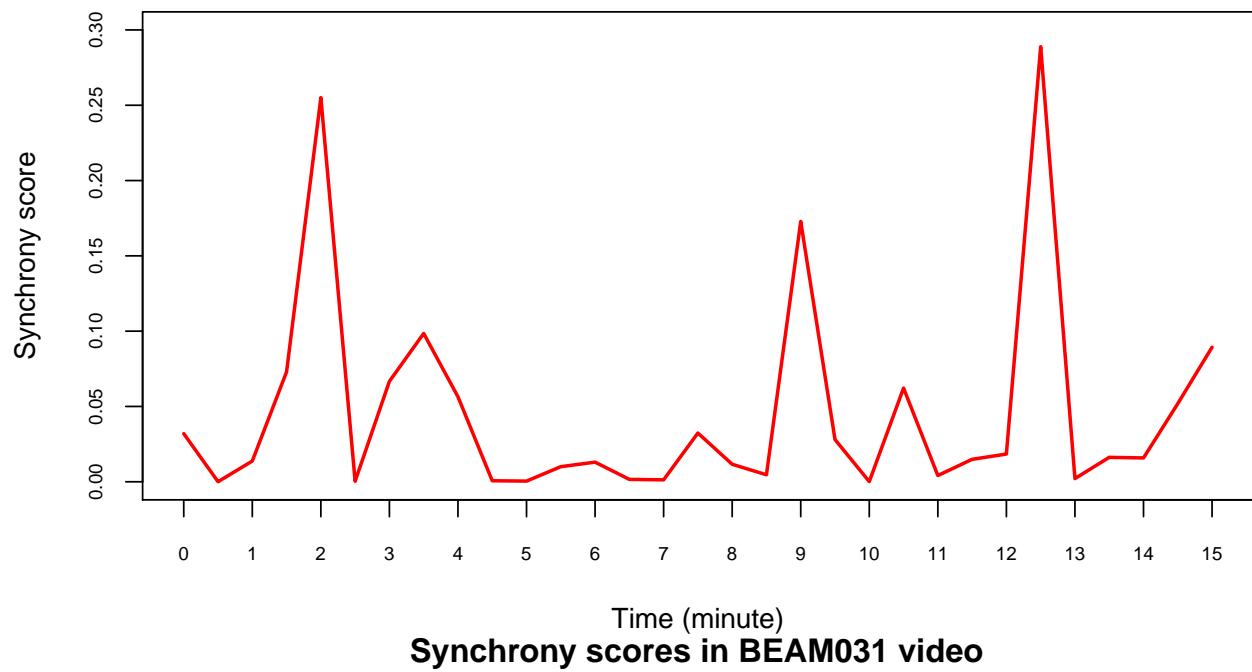
Synchrony scores in BALE050 video



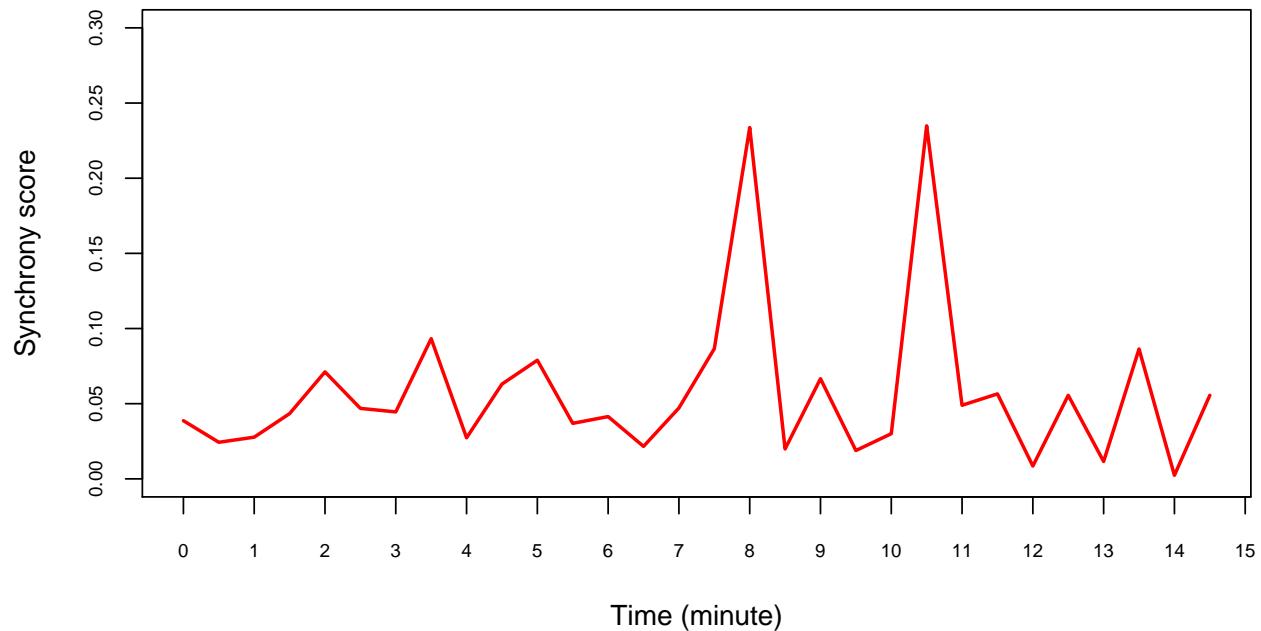
Synchrony scores in BALU062 video



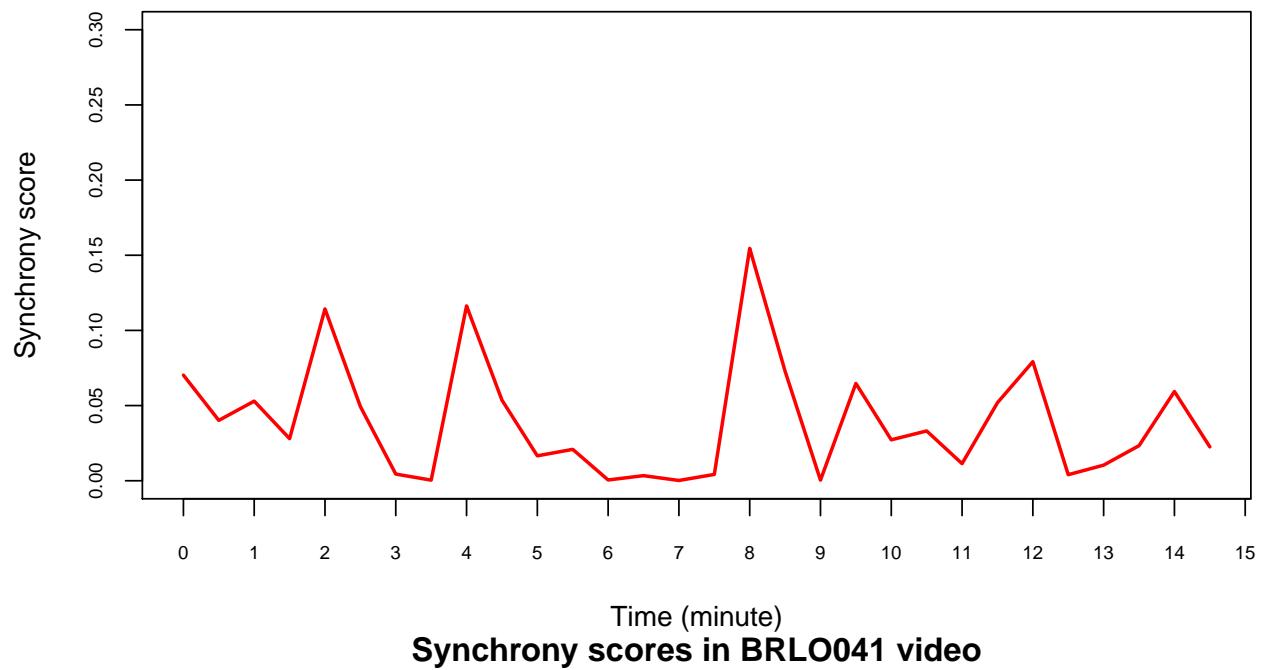
Synchrony scores in BEAL036 video



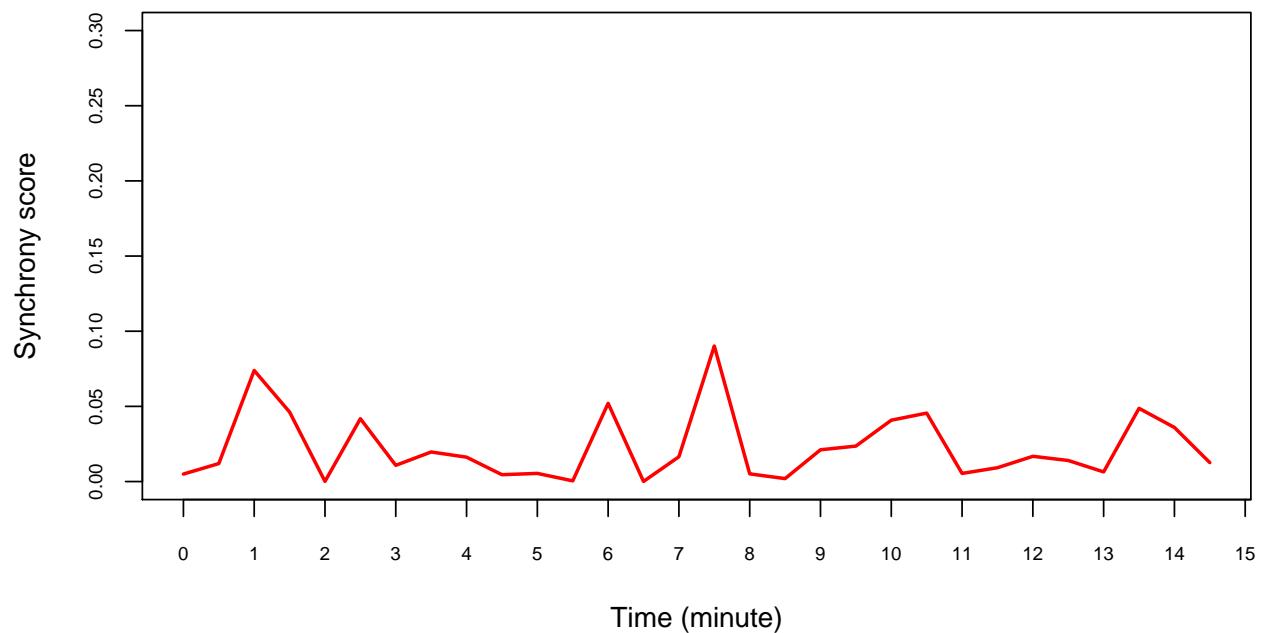
Synchrony scores in BEAM031 video



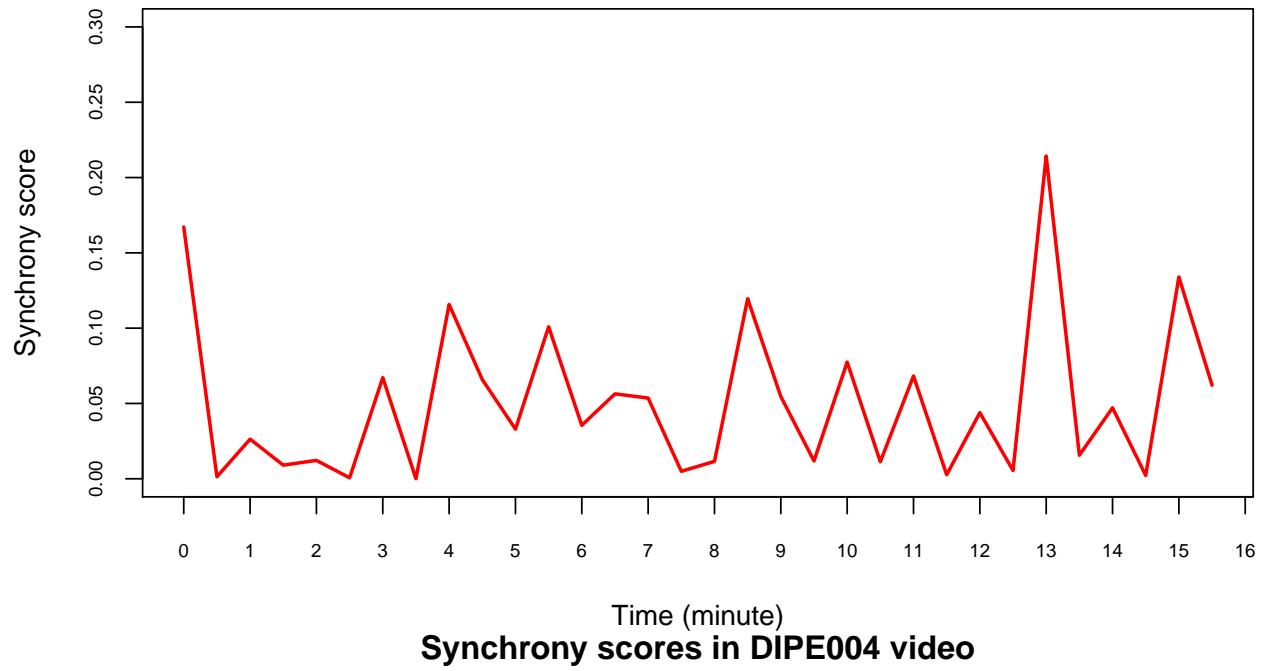
Synchrony scores in BICA video



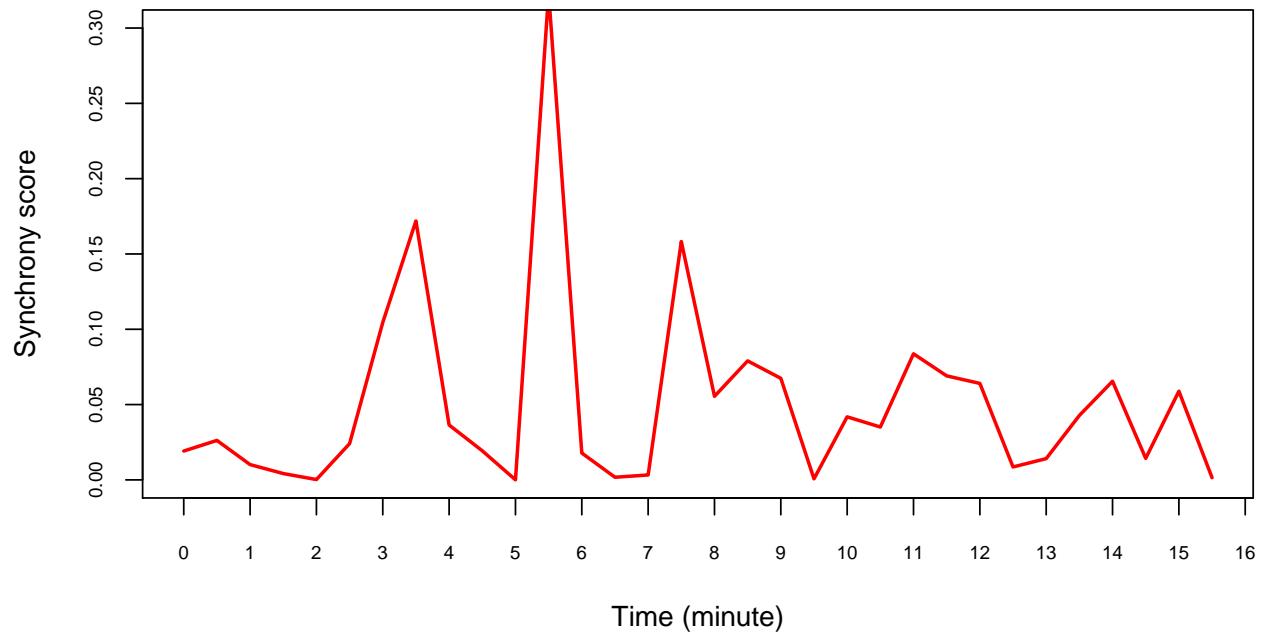
Synchrony scores in BRLO041 video



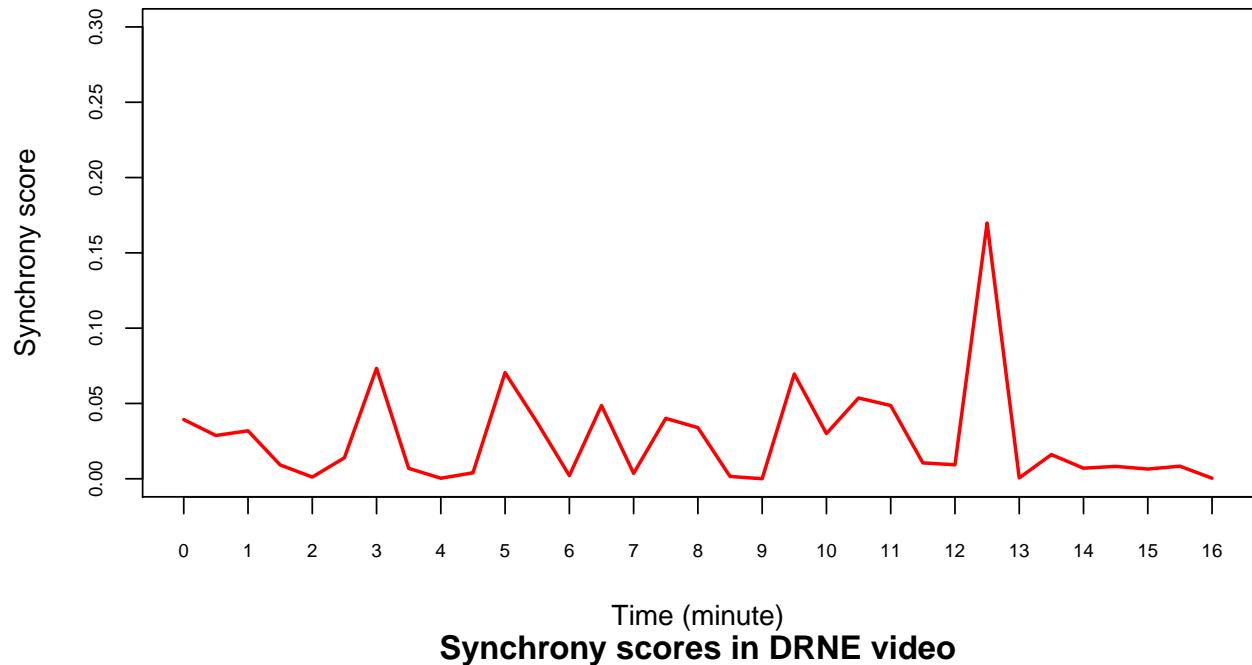
Synchrony scores in COLO022 video



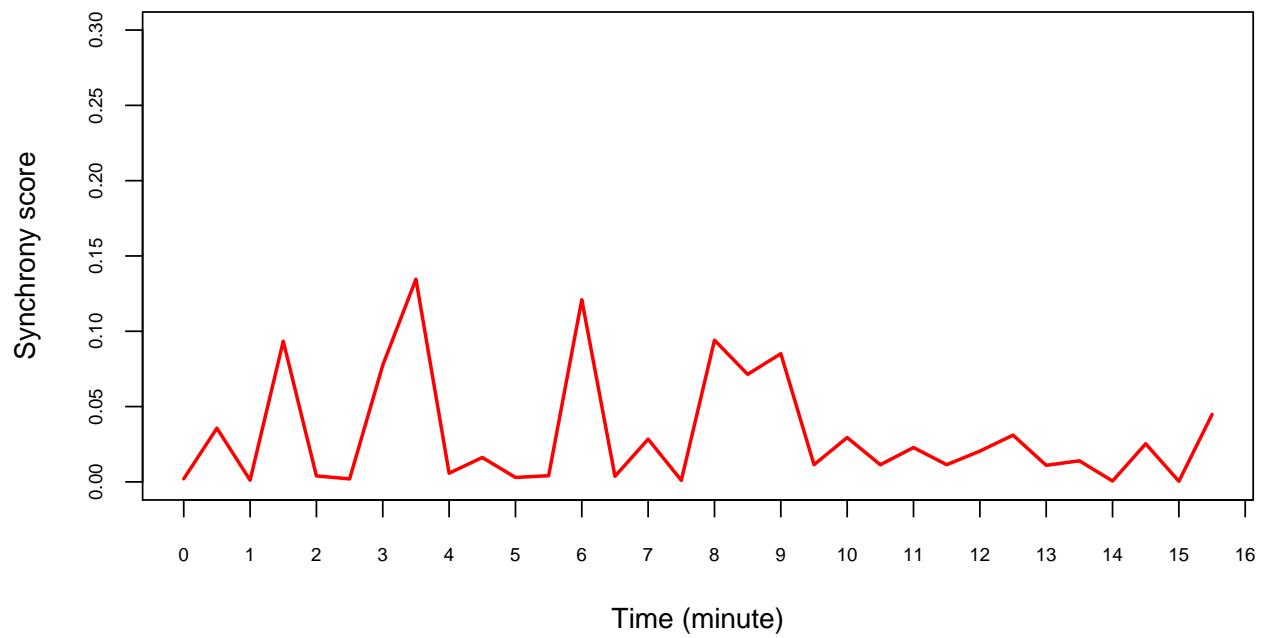
Synchrony scores in DIPE004 video



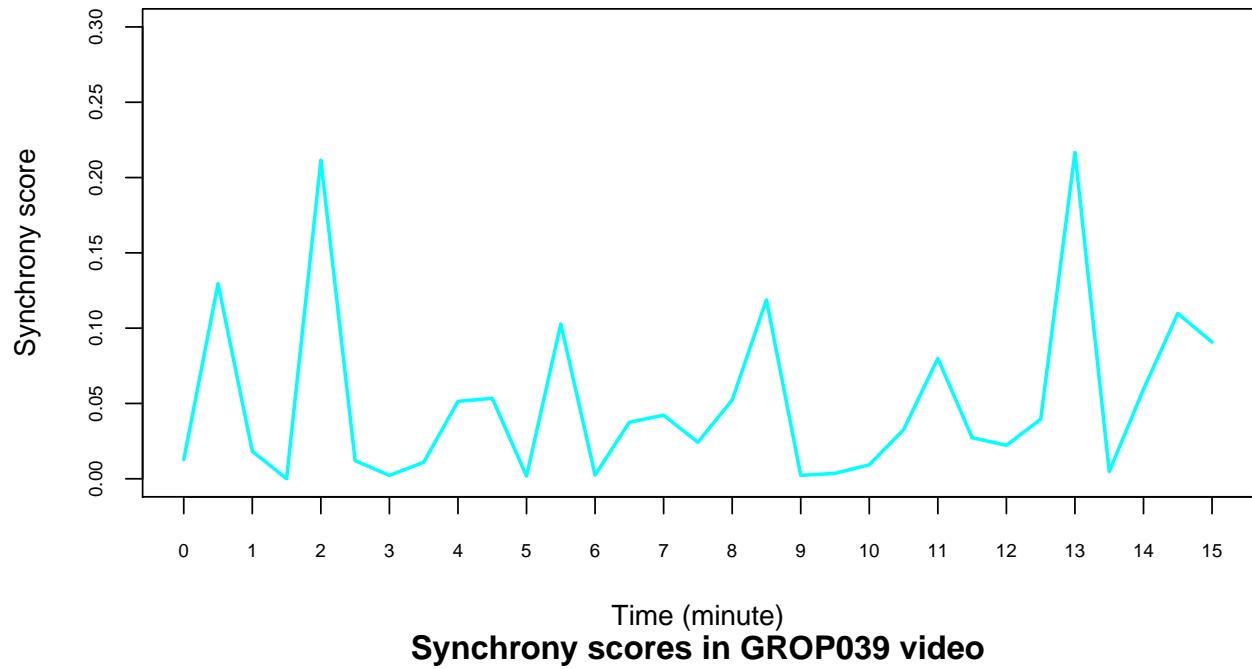
Synchrony scores in DOMA video



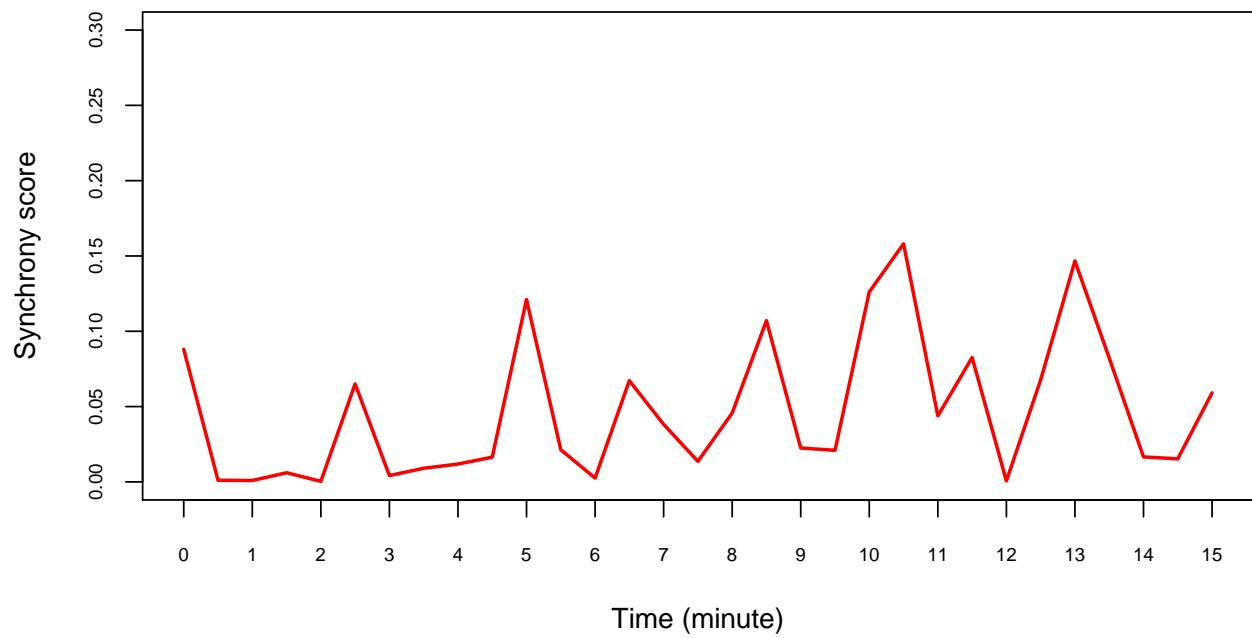
Synchrony scores in DRNE video



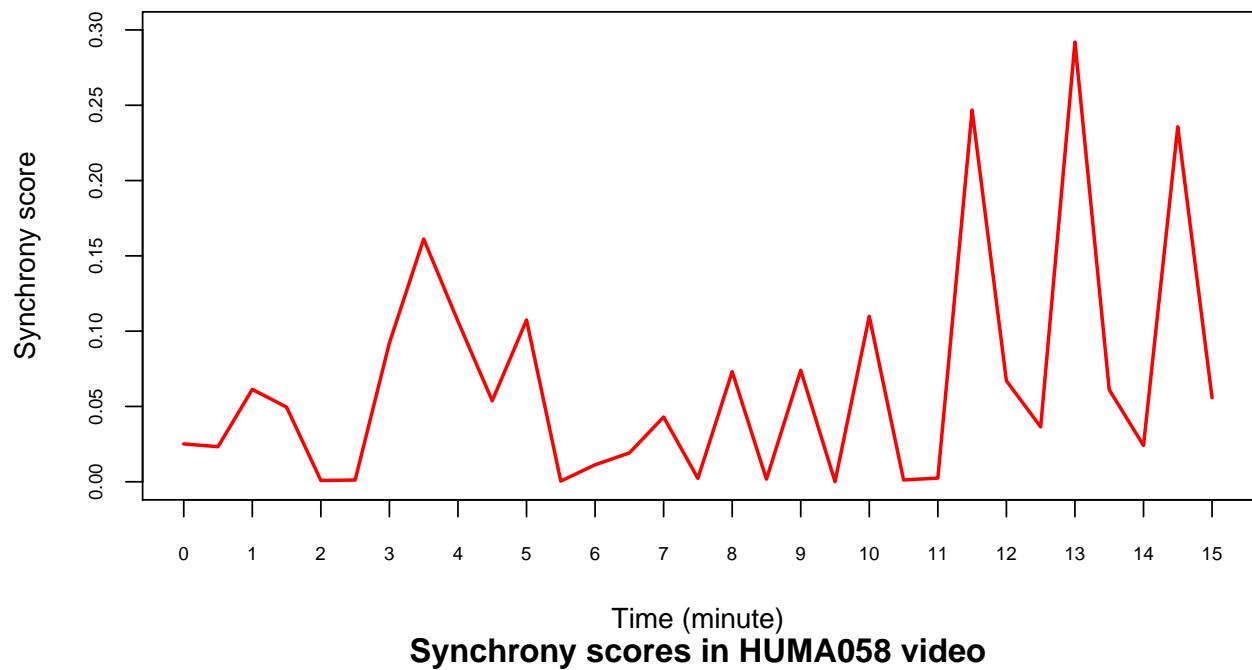
Synchrony scores in FOMA057 video



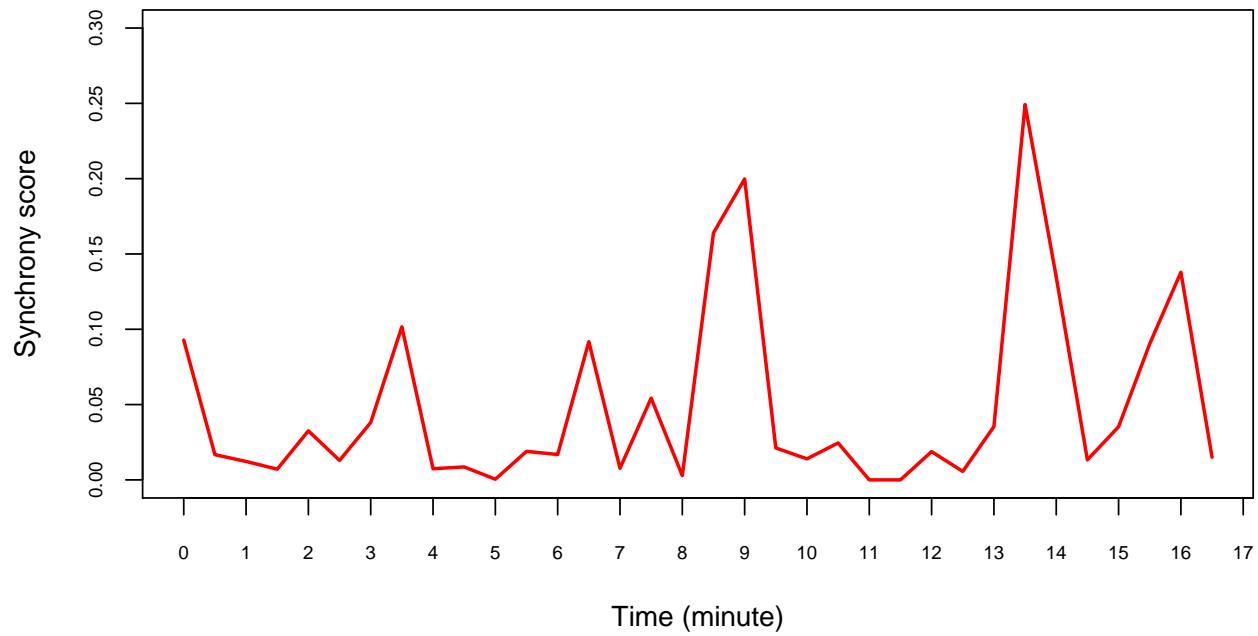
Synchrony scores in GROP039 video



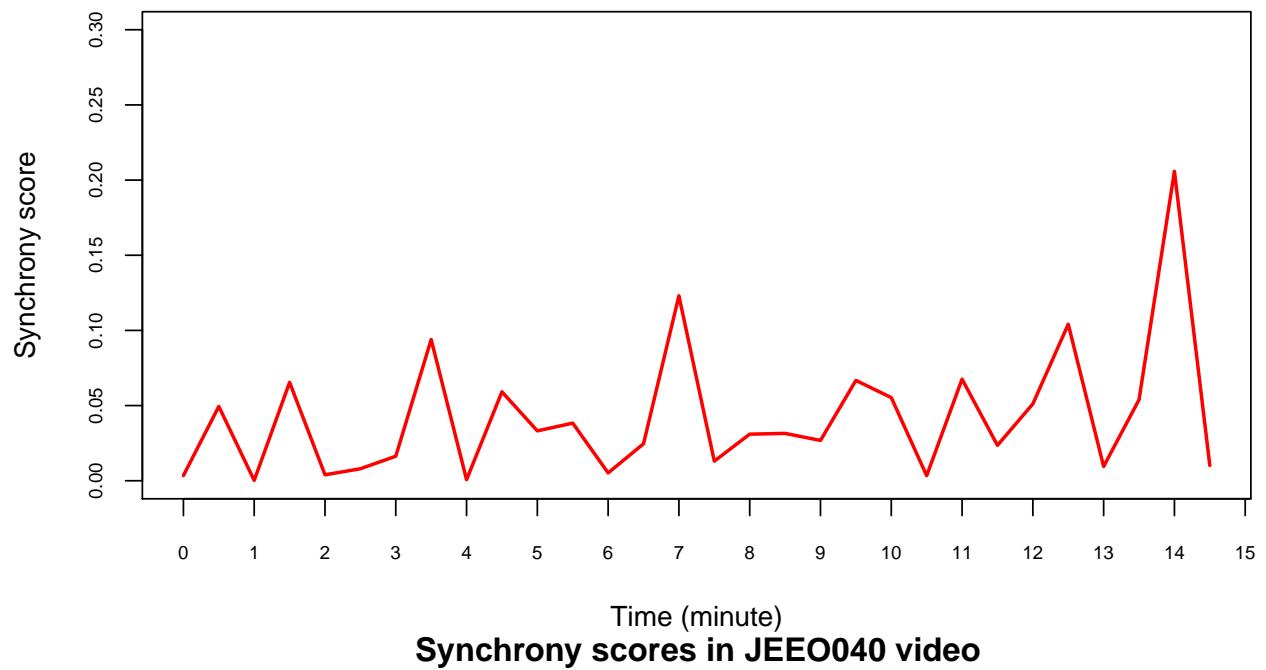
Synchrony scores in HAJA052 video



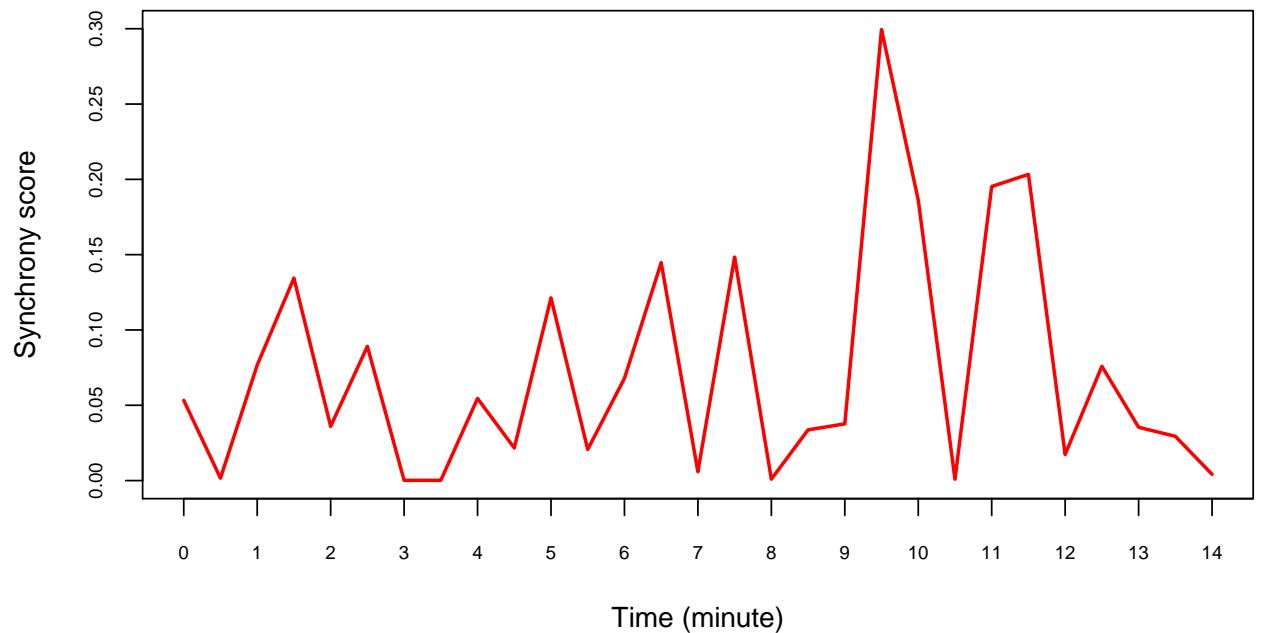
Synchrony scores in HUMA058 video



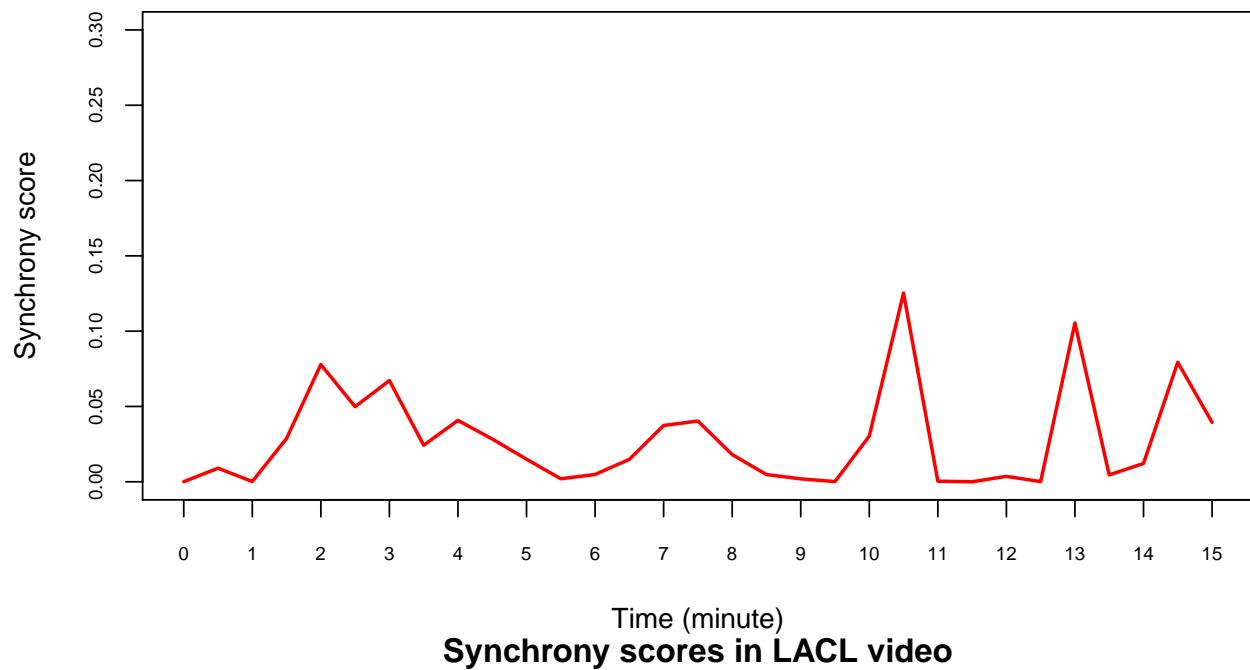
Synchrony scores in JAEM046 video



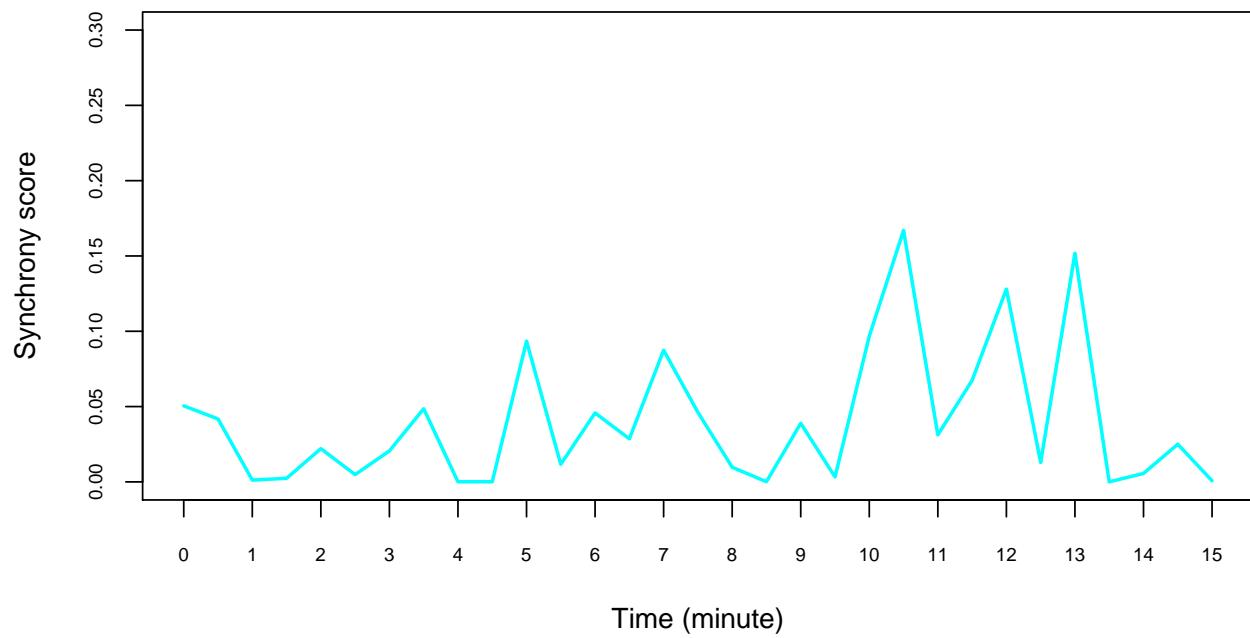
Synchrony scores in JEEO040 video



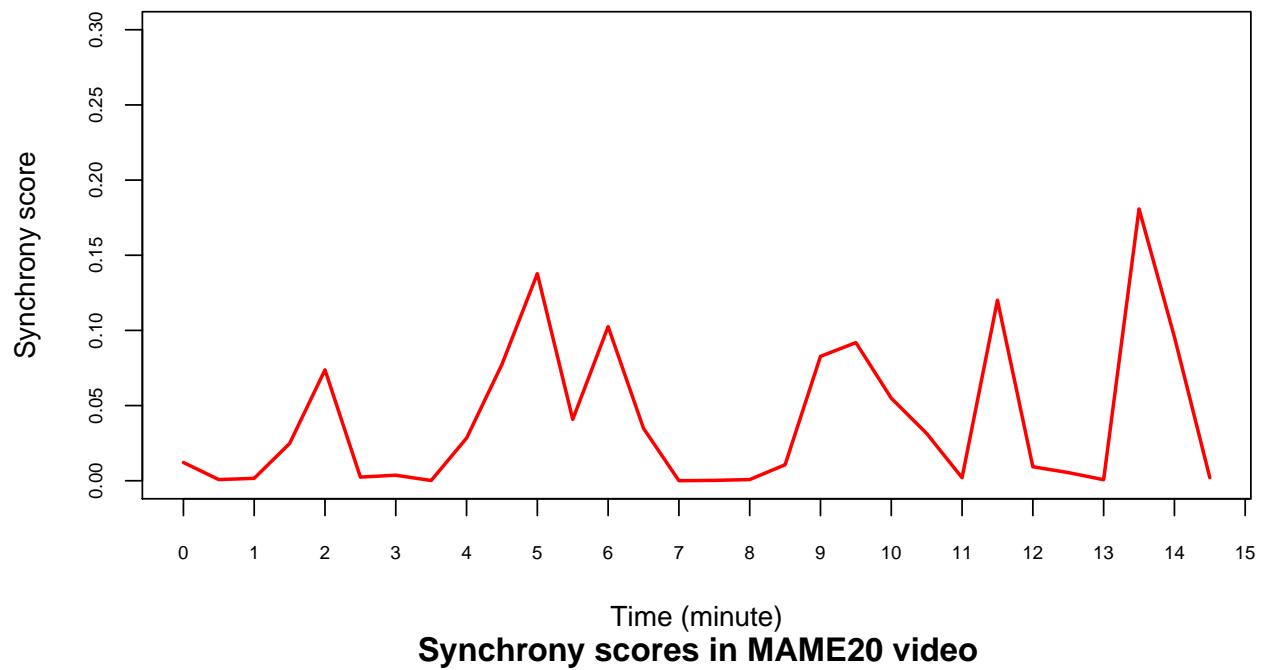
Synchrony scores in JOCE014 video



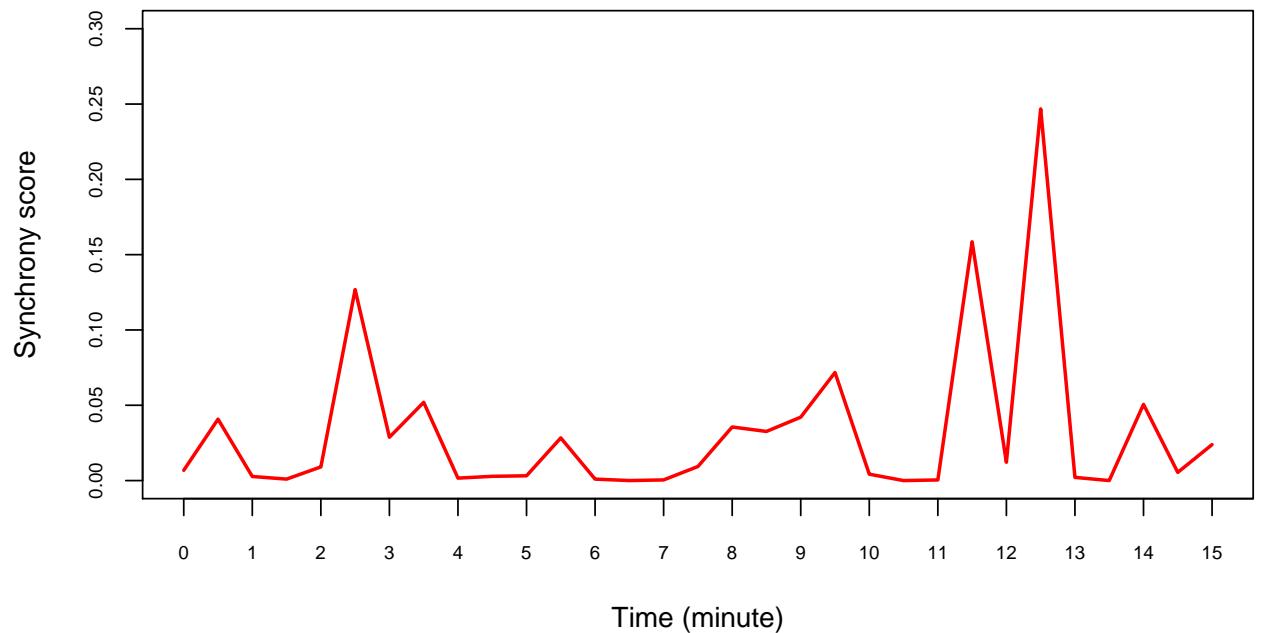
Synchrony scores in LACL video



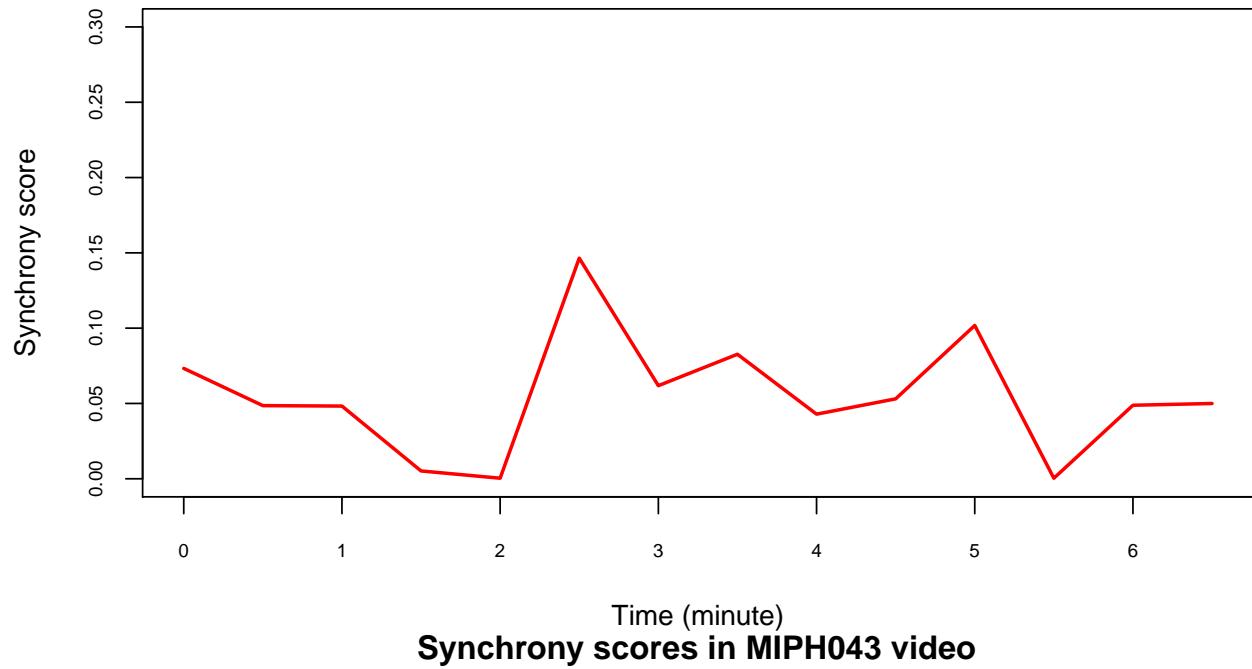
Synchrony scores in MAEL048 video



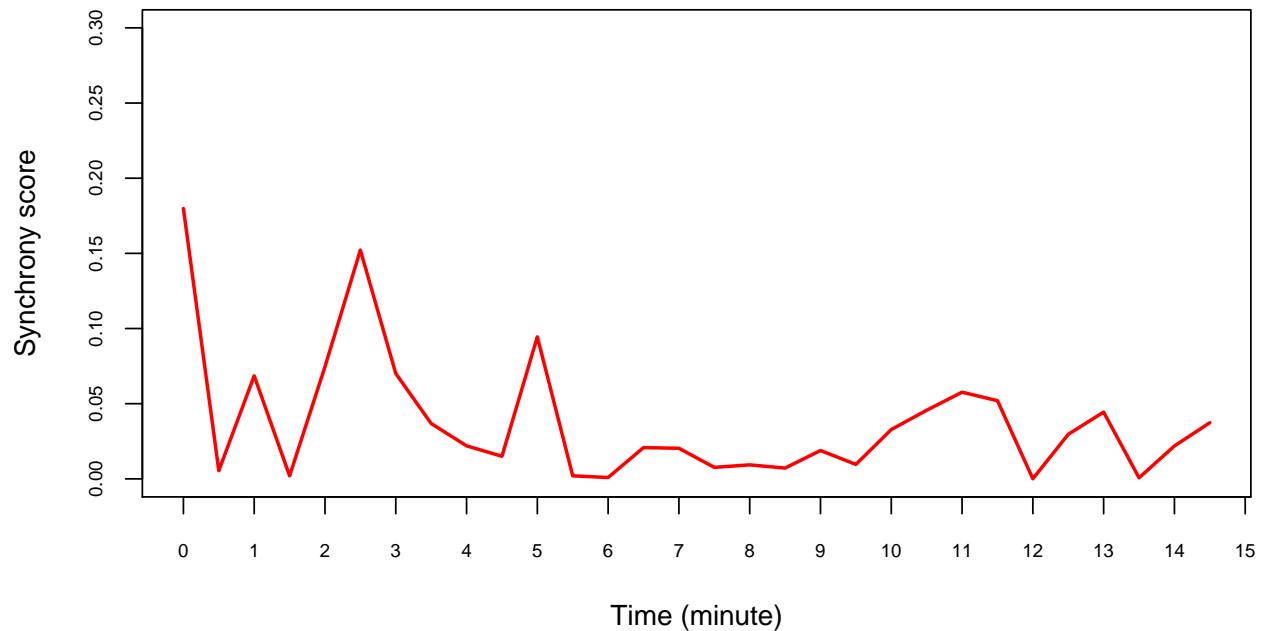
Synchrony scores in MAME20 video



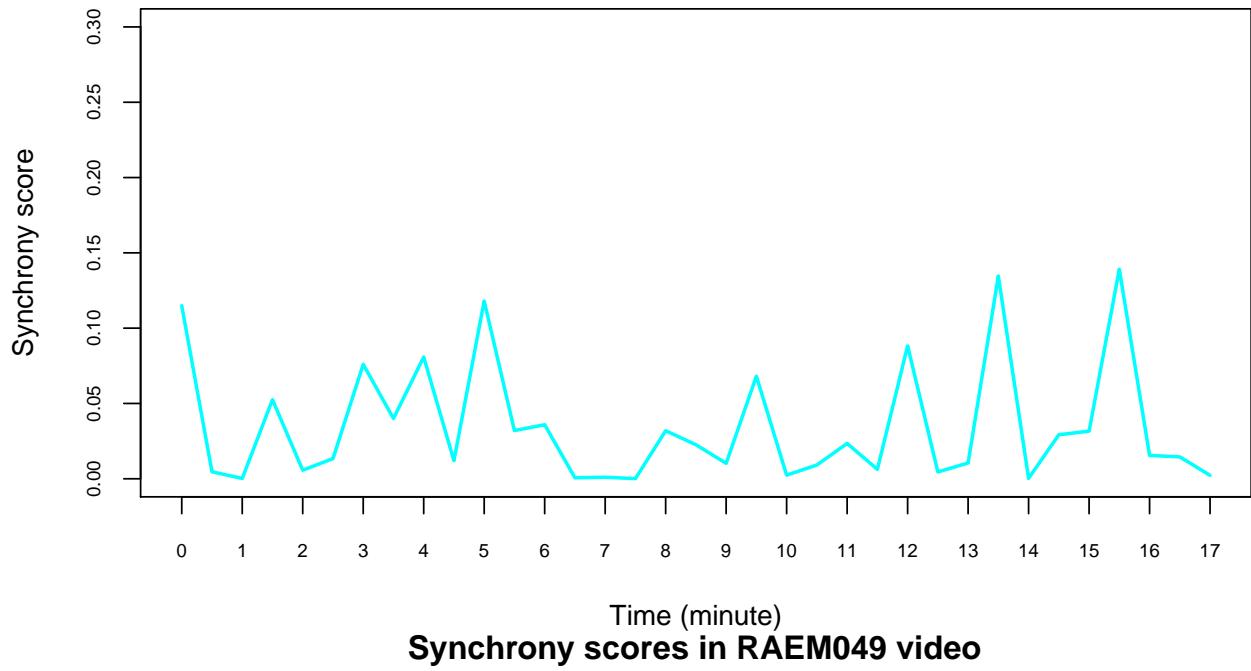
Synchrony scores in MAPA029 video



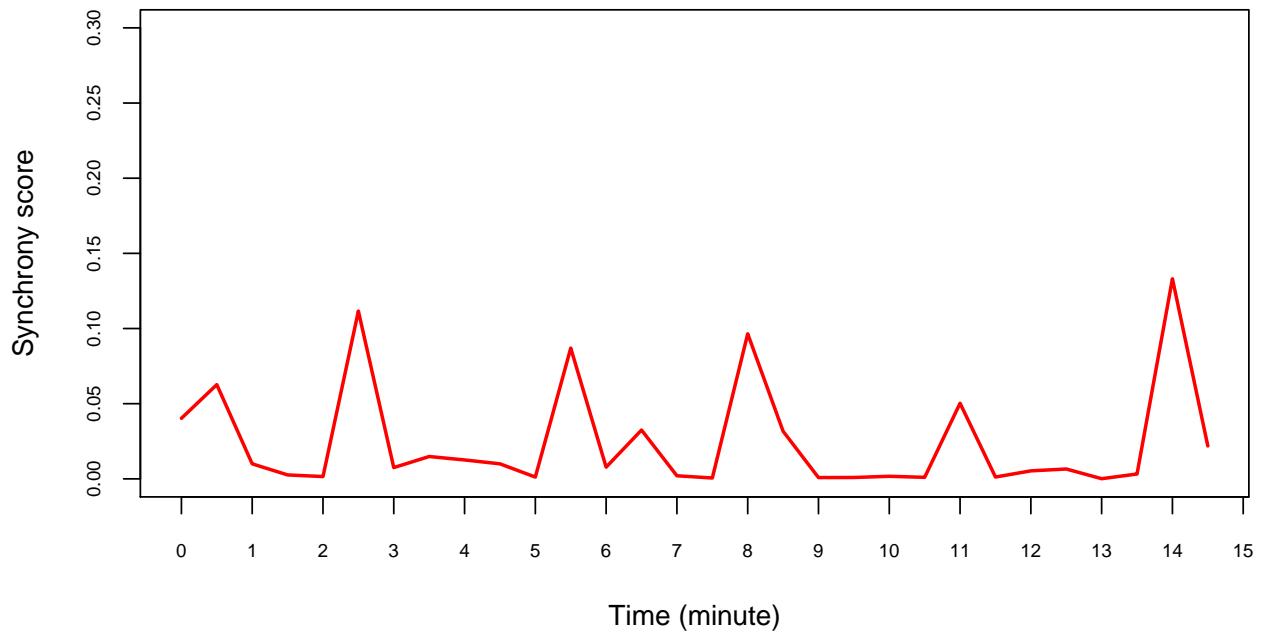
Synchrony scores in MIPH043 video



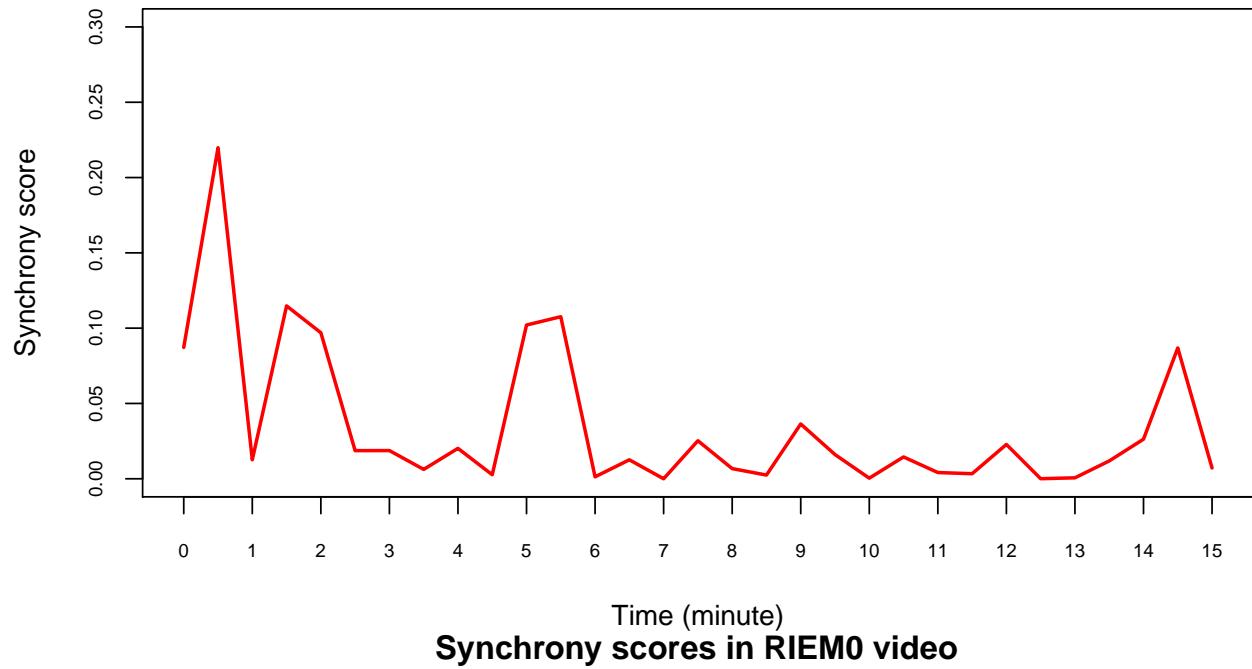
Synchrony scores in MOSA065 video



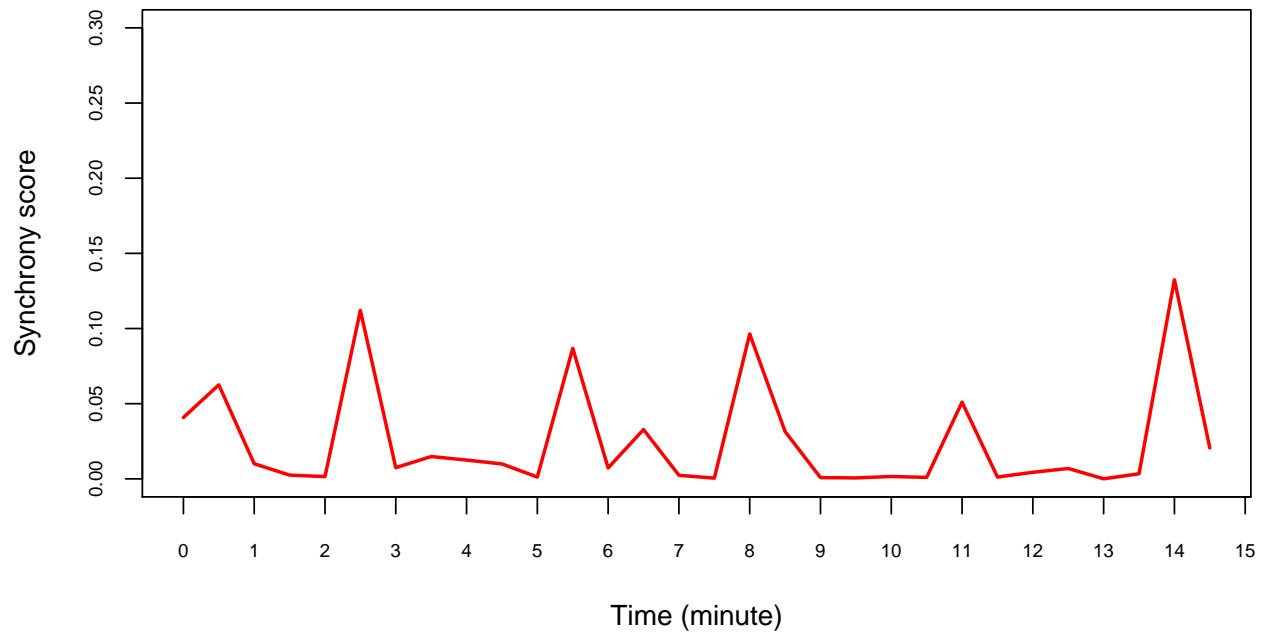
Synchrony scores in RAEM049 video



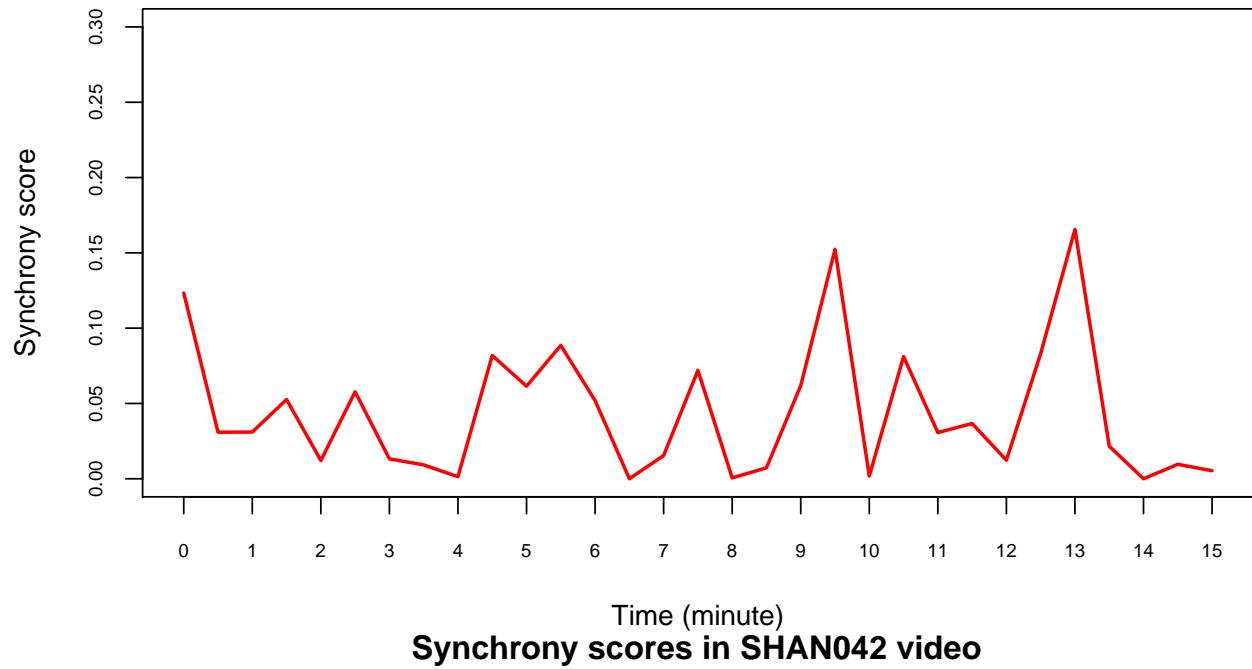
Synchrony scores in RAKA008 video



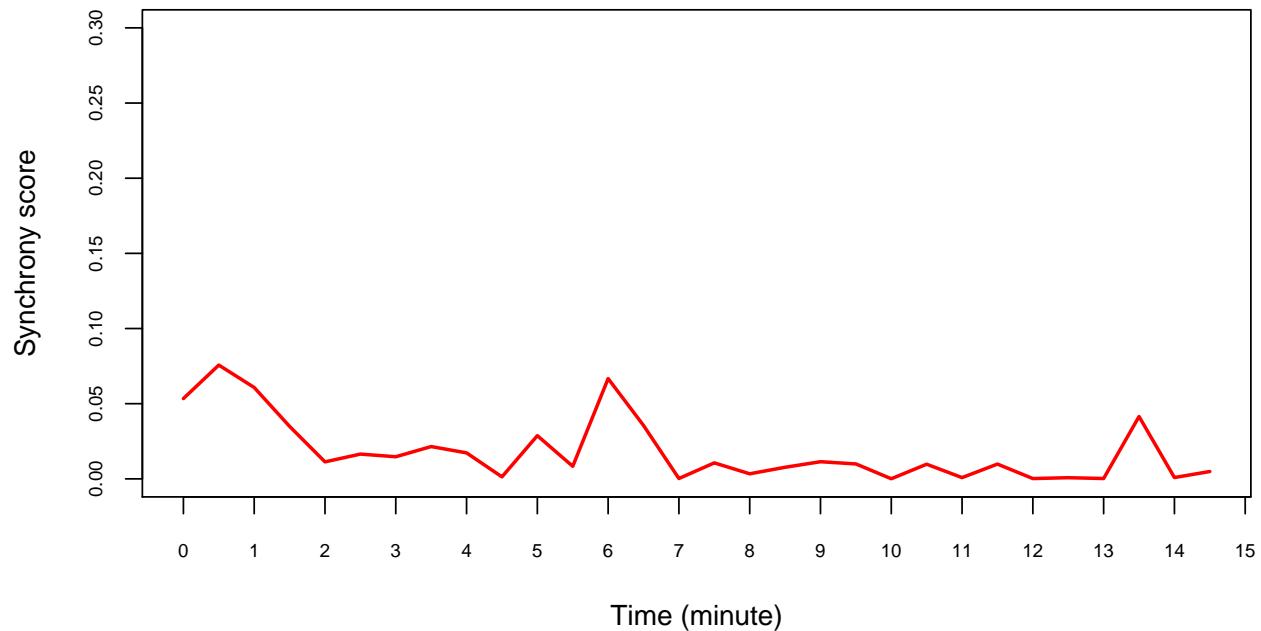
Synchrony scores in RIEM0 video



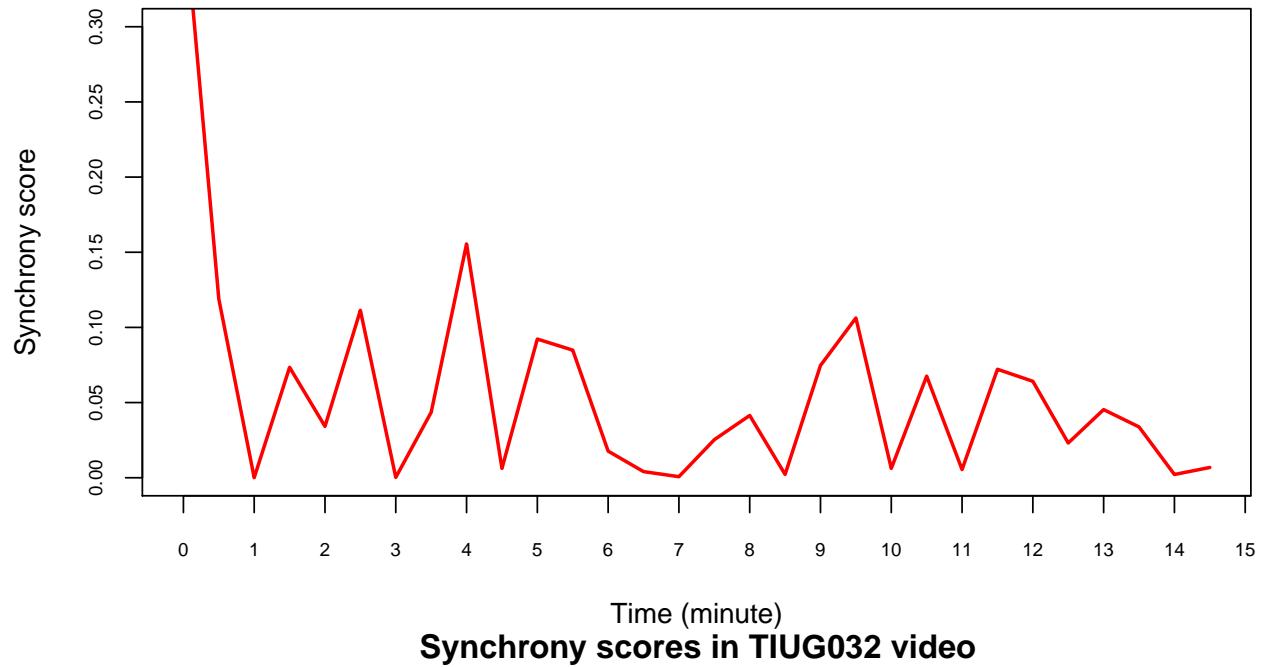
Synchrony scores in SEEM035 video



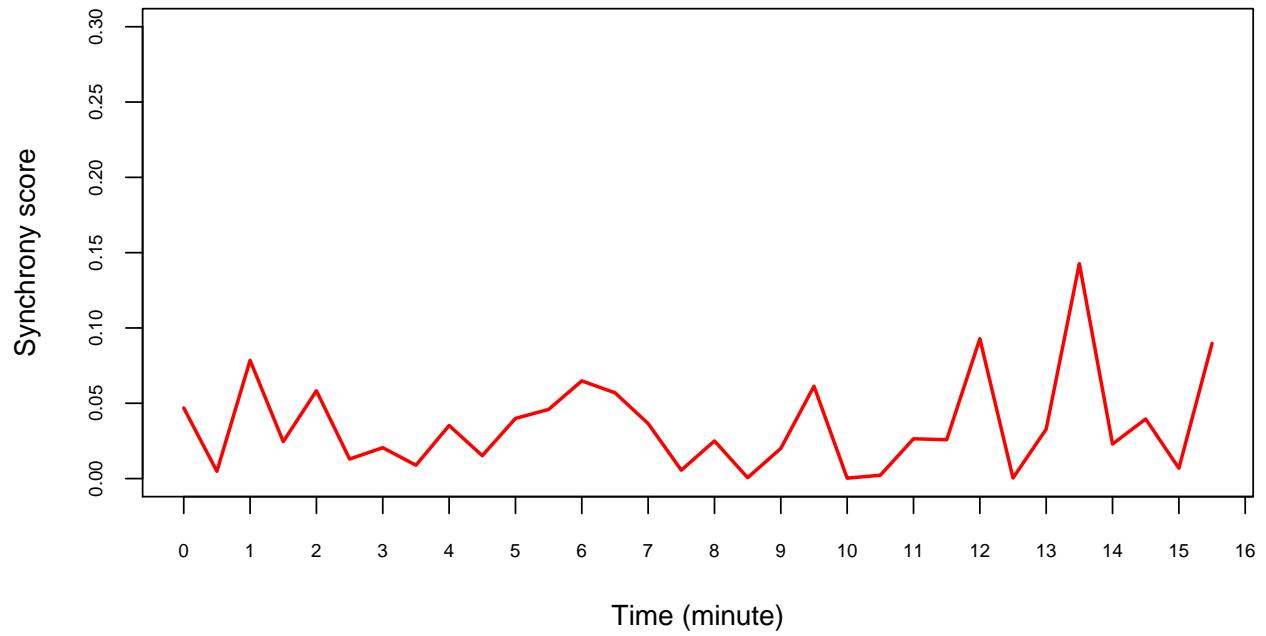
Synchrony scores in SHAN042 video



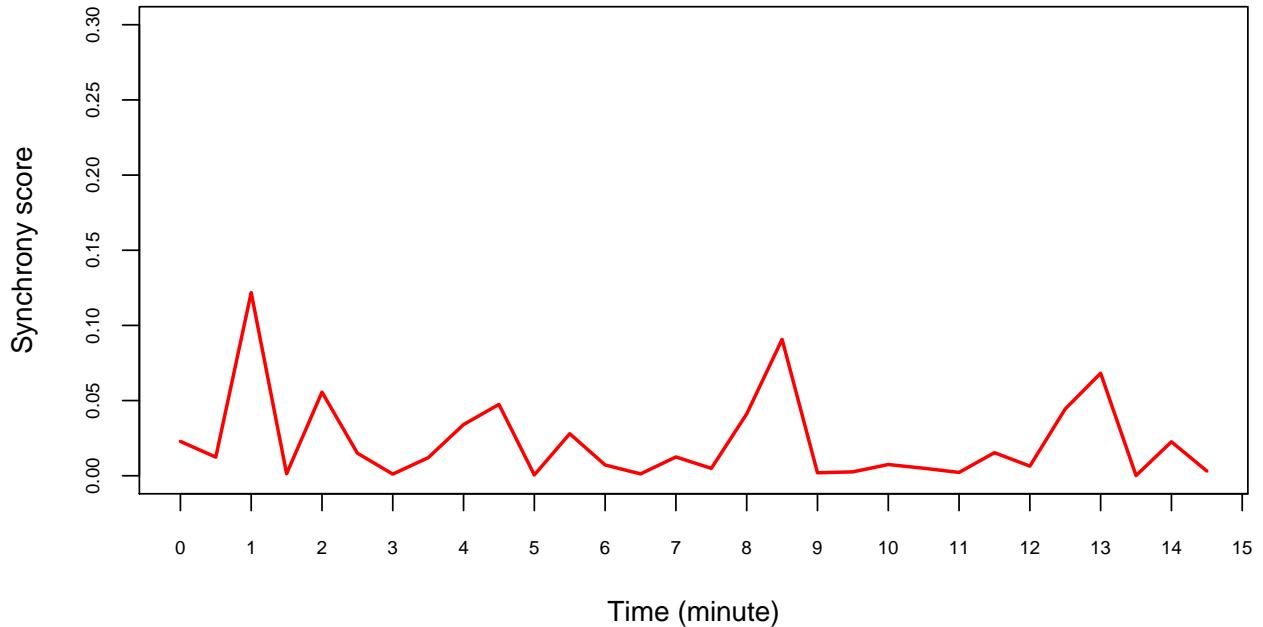
Synchrony scores in SOGA061 video



Synchrony scores in TIUG032 video



Synchrony scores in VINO video



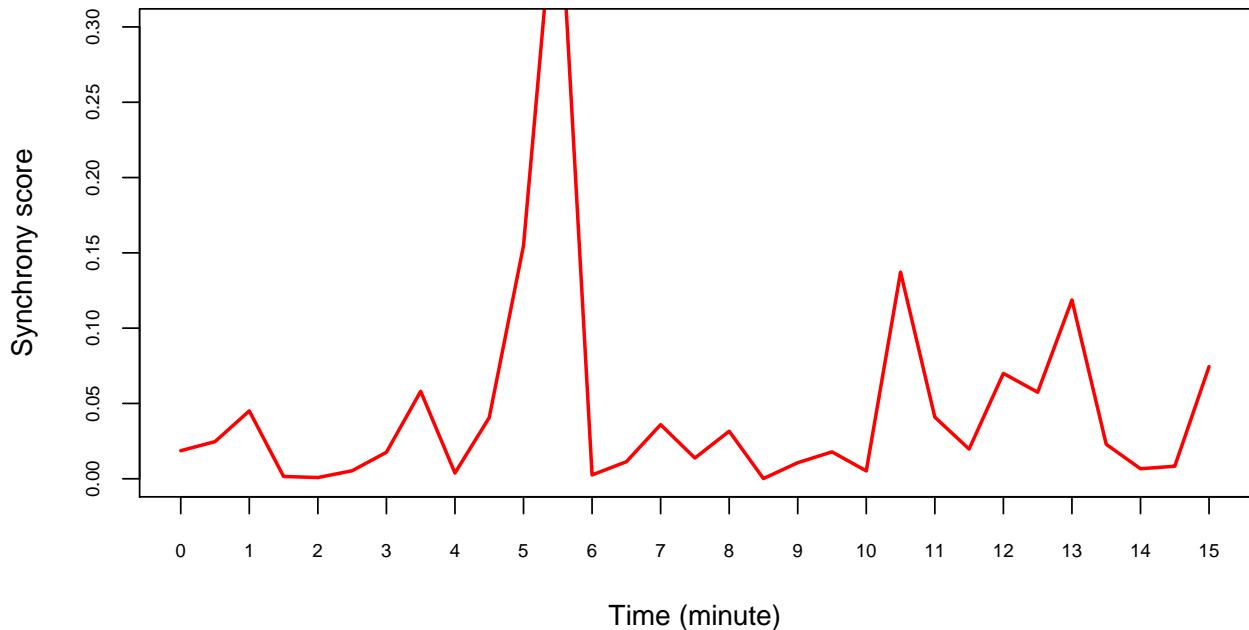
```
#par(mar=c(4,4,4,3), mfrow=c(1,1))

#for (i in unique(SSILog$video)){
#  plot(SSILog[which(SSILog$video==i),]$Time_min,
#       SSILog[which(SSILog$video==i),]$SSI_mo_ch, type="l", ylim=c(0, 0.3), #col=rainbow(4)[1], main=
#
#       lines(SSILog[which(SSILog$video==i),]$SSI_fa_mo_ch, col=rainbow(4)[2], lwd=2)
#       lines(SSILog[which(SSILog$video==i),]$SSI_fa_ch, col=rainbow(4)[3], lwd=2)
#
#       str(SSILog[which(SSILog$video==i),]$Time_min)
#       str(SSILog[which(SSILog$video==i),]$SSI_mo_ch)
#       str(SSILog[which(SSILog$video==i),]$SSI_fa_mo_ch)
#       str(SSILog[which(SSILog$video==i),]$SSI_fa_ch)}
#
#abline(h= mean(SSILog$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)}
```

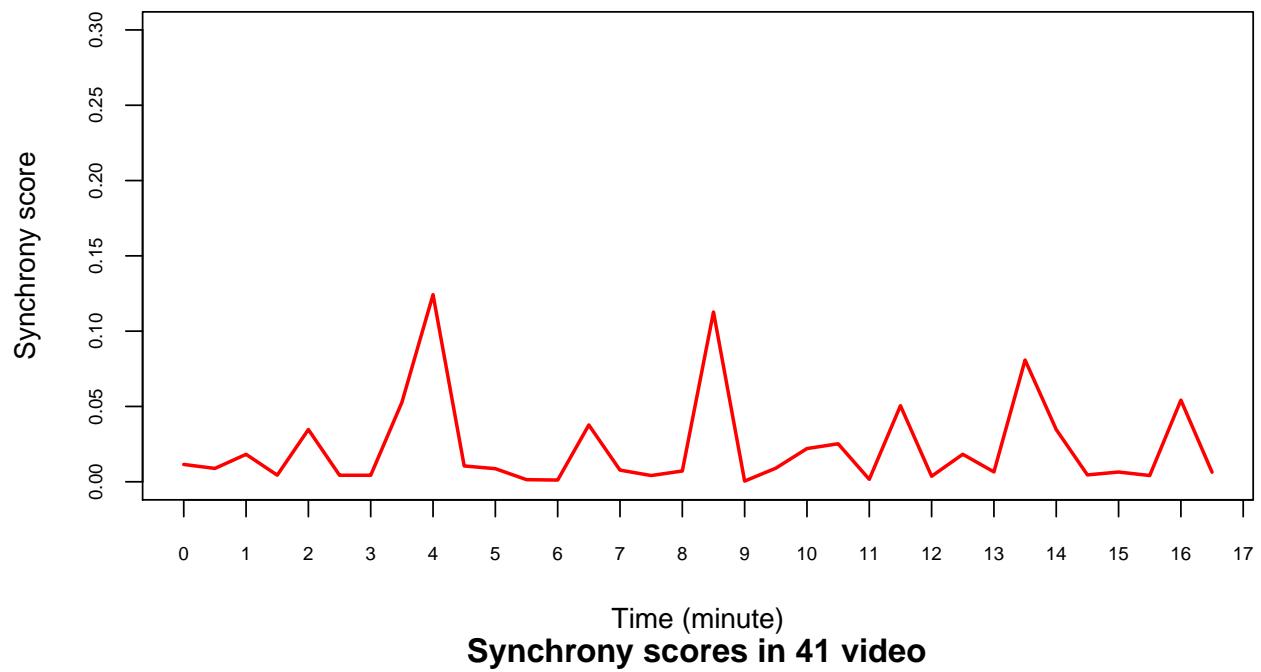
Synchrony scores noLog for each dyad, triad and for the whole group

```
par(mar=c(4,4,4,3), mflow=c(1,1))
for (i in unique(SSInoLog$video)){
  if (all(!is.na(SSInoLog[which(SSInoLog$video==i),]$SSI_mo_ch)==TRUE)){
    #   print(str(SSInoLog[which(SSInoLog$video==i),]$Time_min))
    #   print(str(SSInoLog[which(SSInoLog$video==i),]$SSI_mo_ch))
    plot(SSInoLog[which(SSInoLog$video==i),]$Time_min, SSInoLog[which(SSInoLog$video==i),]$SSI_mo_ch,
         ylim=c(0, 0.3), main=paste("Synchrony scores in", i, "video"), xlab = "Time (minute)", ylab="Synchrony score")
    else if(all(!is.na(SSInoLog[which(SSInoLog$video==i),]$SSI_fa_ch)==TRUE)){
      #   print(str(SSInoLog[which(SSInoLog$video==i),]$SSI_fa_ch))
      plot(SSInoLog[which(SSInoLog$video==i),]$Time_min, SSInoLog[which(SSInoLog$video==i),]$SSI_fa_ch, ylim=c(0, 0.3),
           main=paste("Synchrony scores in", i, "video"), xlab = "Time (minute)", ylab="Synchrony score")
    else{print("error")}}
  }
}
```

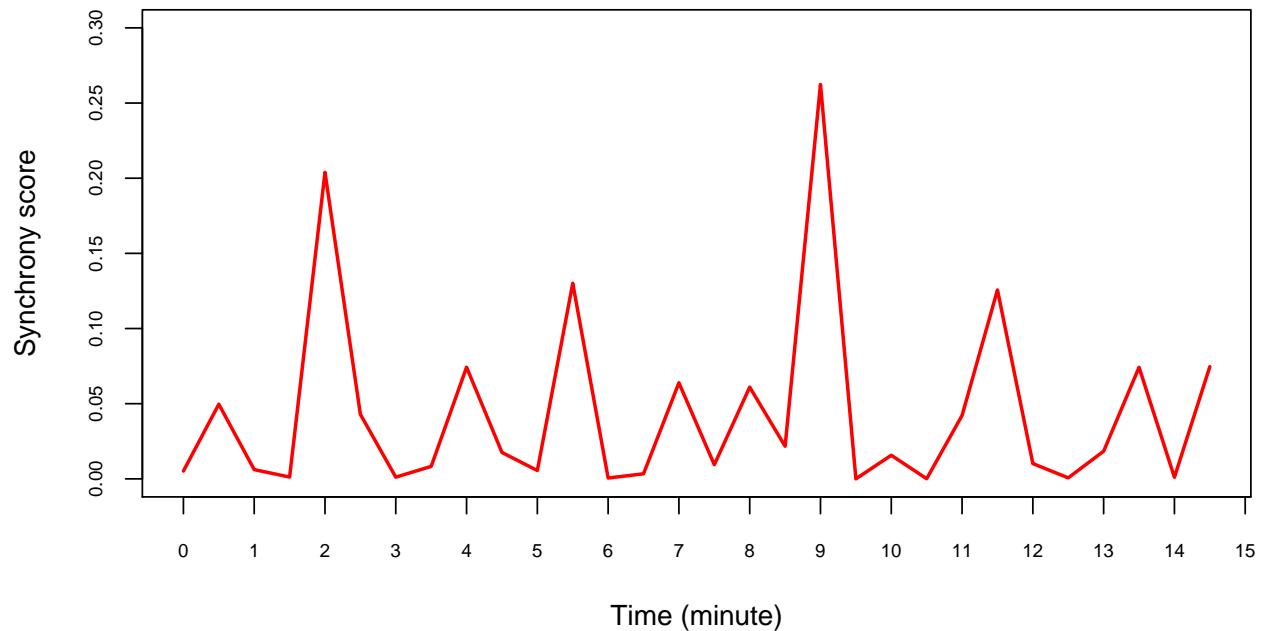
Synchrony scores in 34 video



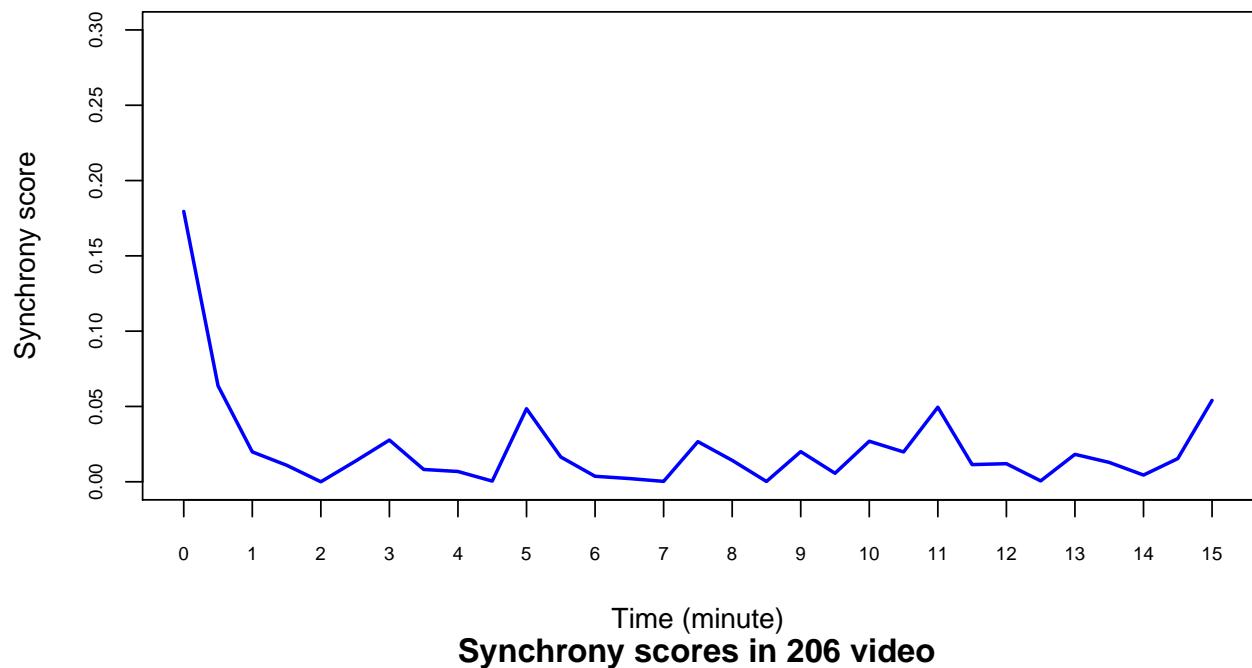
Synchrony scores in 37 video



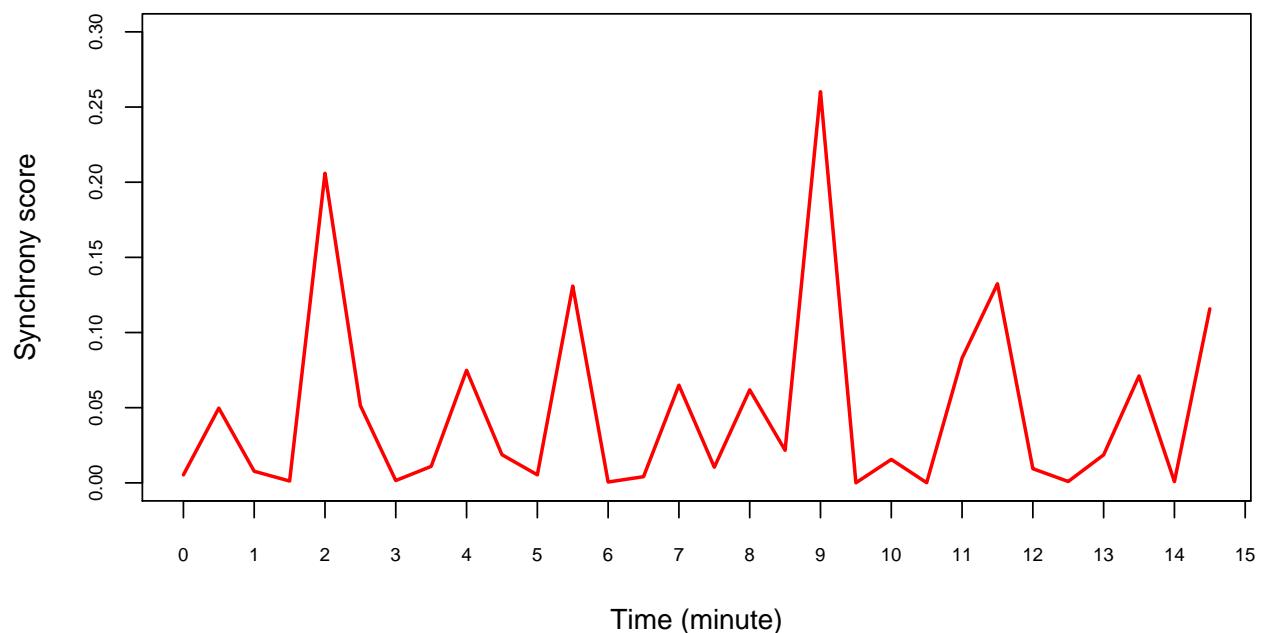
Synchrony scores in 41 video



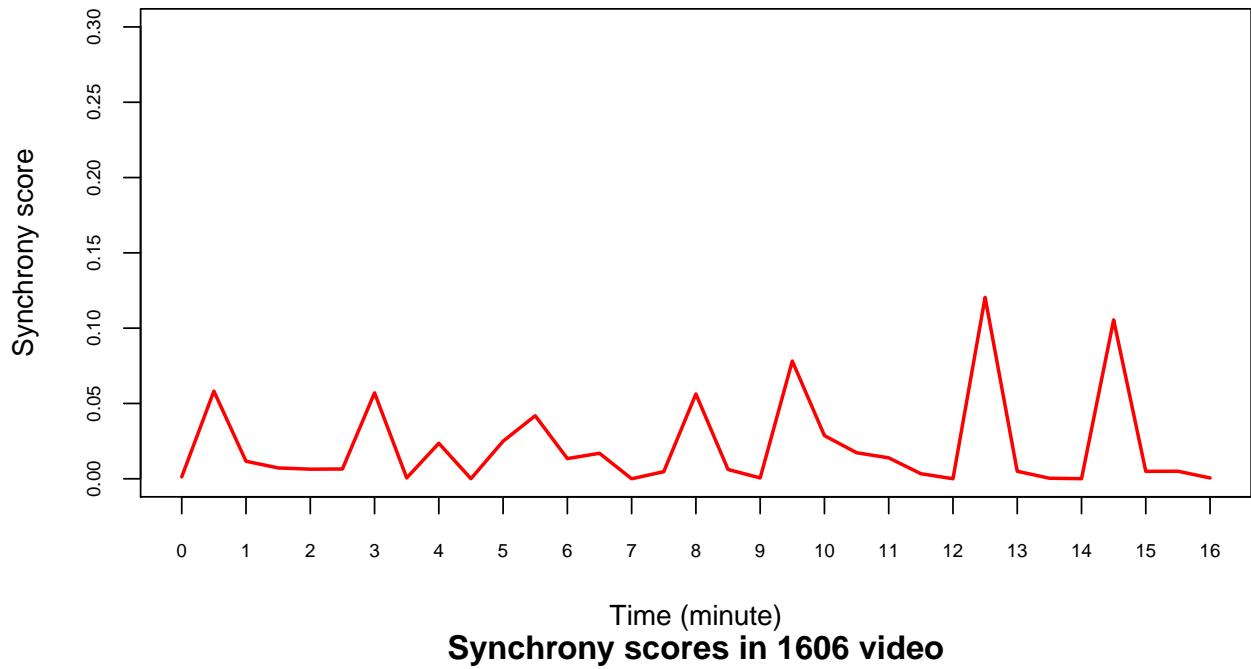
Synchrony scores in 48 video



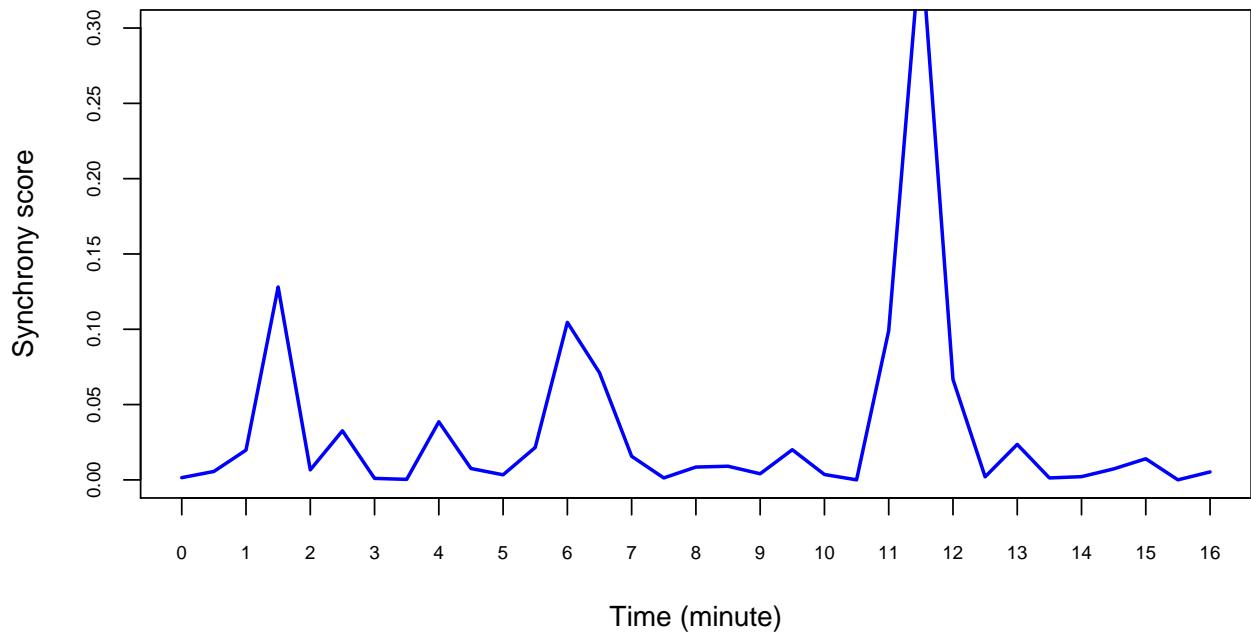
Synchrony scores in 206 video



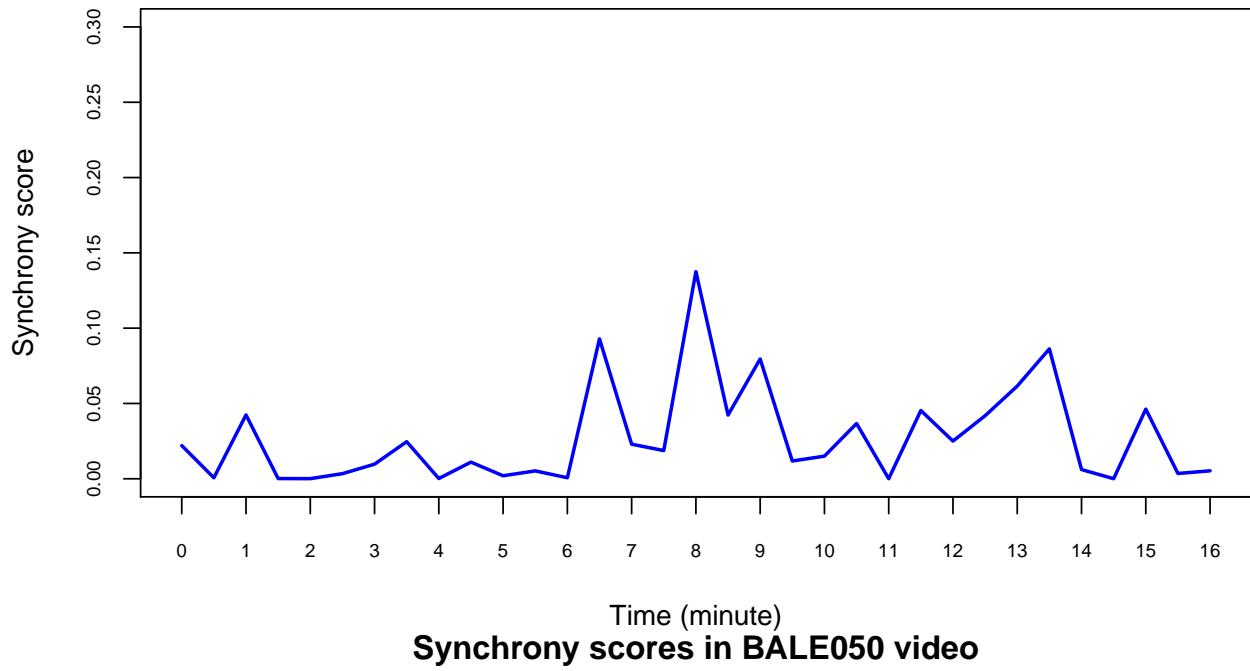
Synchrony scores in 1106 video



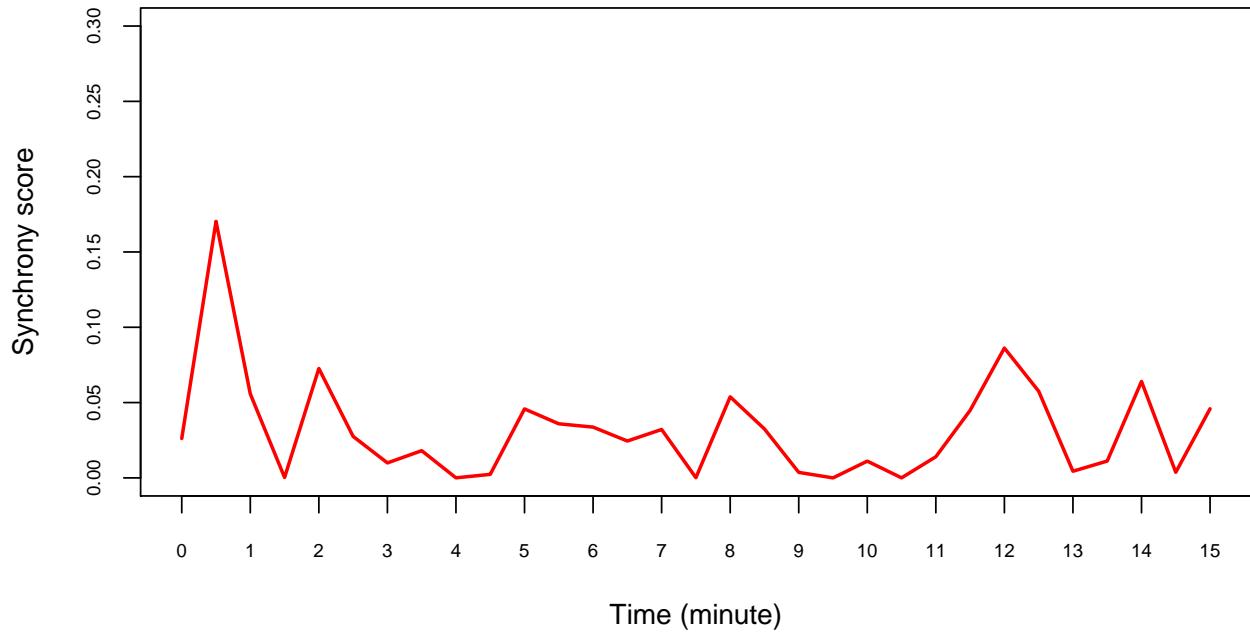
Synchrony scores in 1606 video



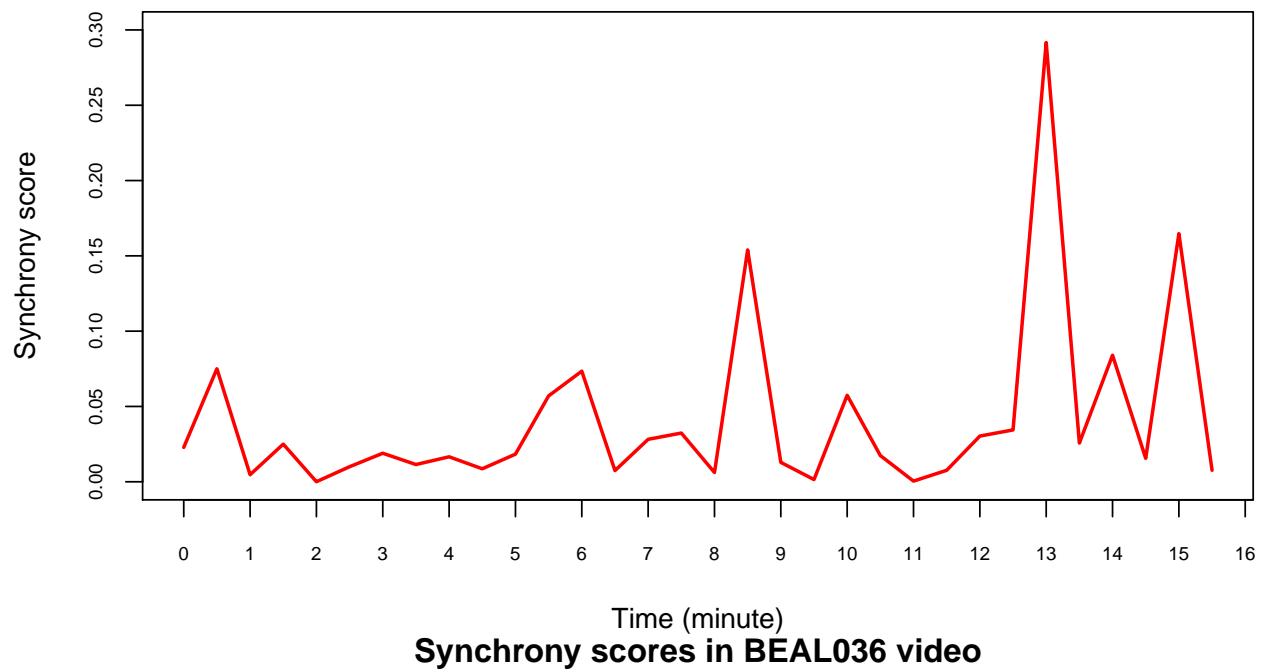
Synchrony scores in BAJE059 video



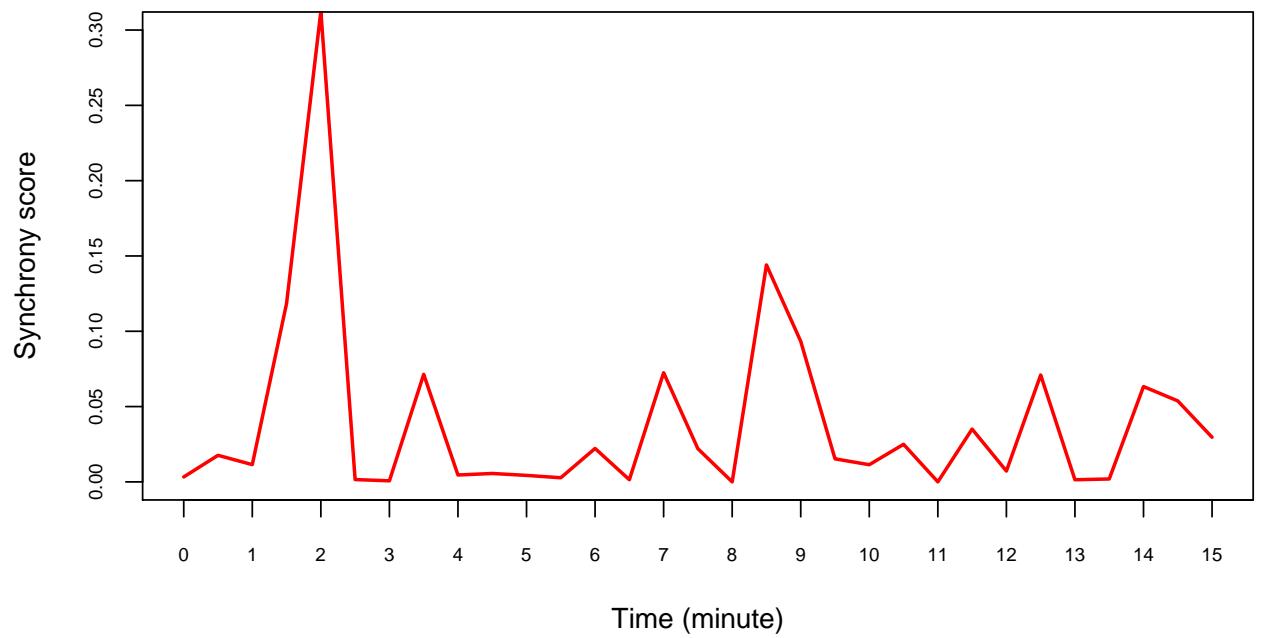
Synchrony scores in BALE050 video



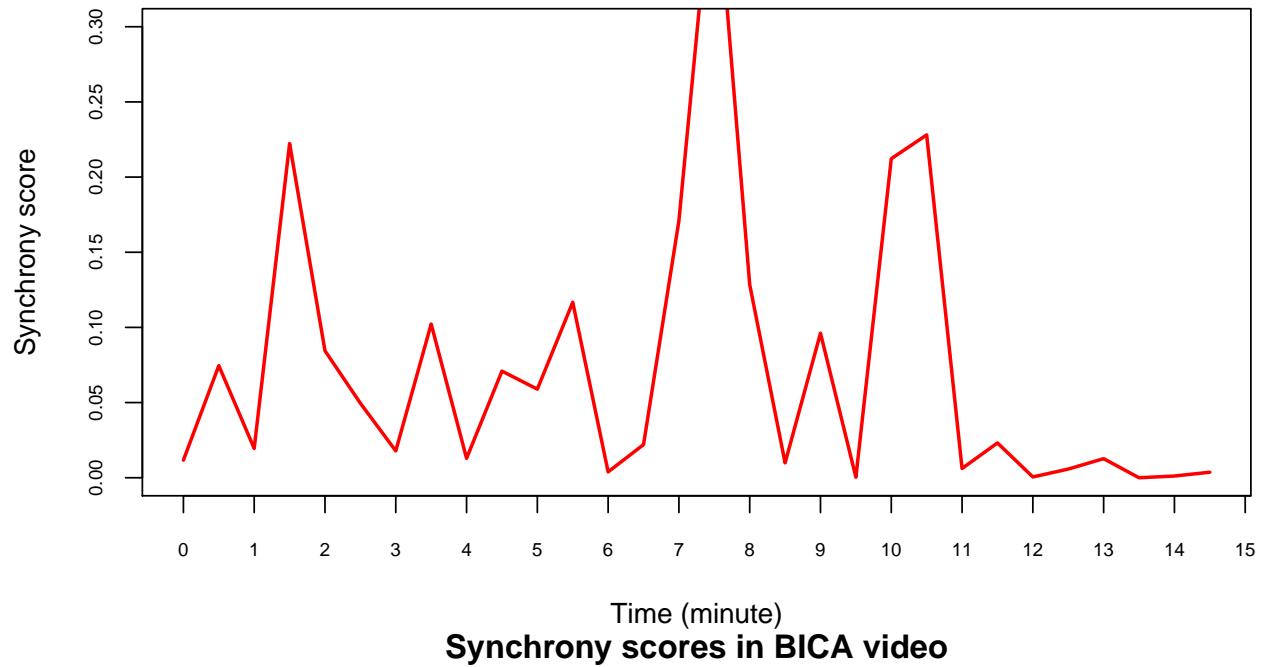
Synchrony scores in BALU062 video



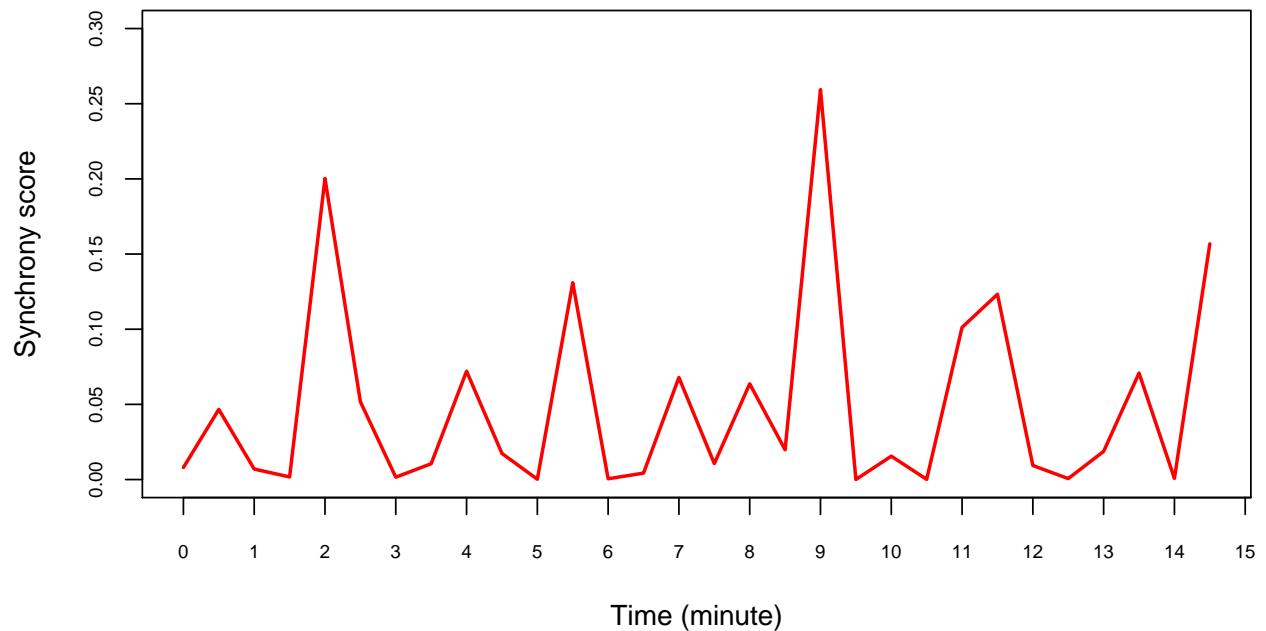
Synchrony scores in BEAL036 video



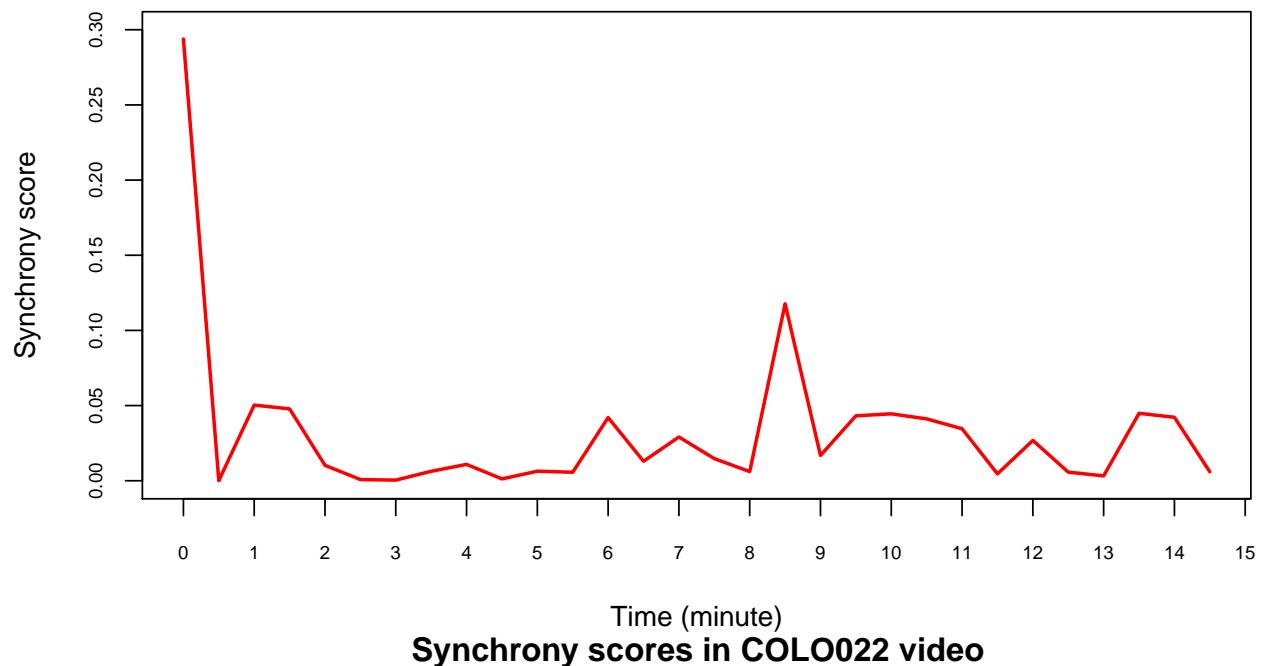
Synchrony scores in BEAM031 video



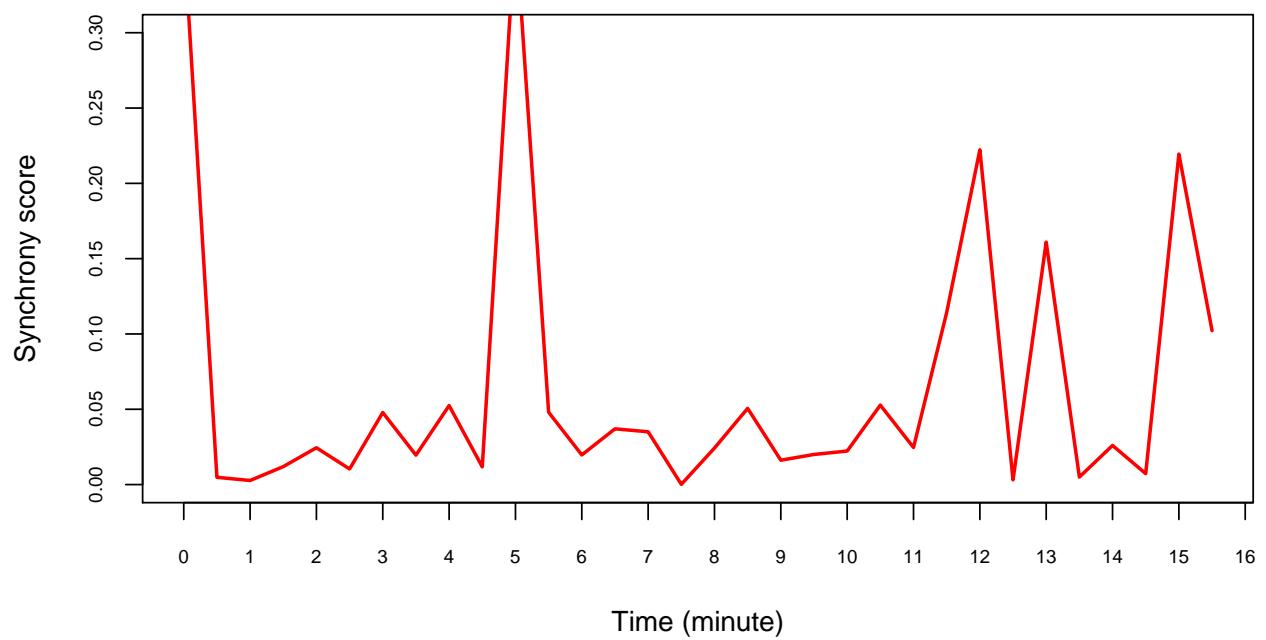
Synchrony scores in BICA video



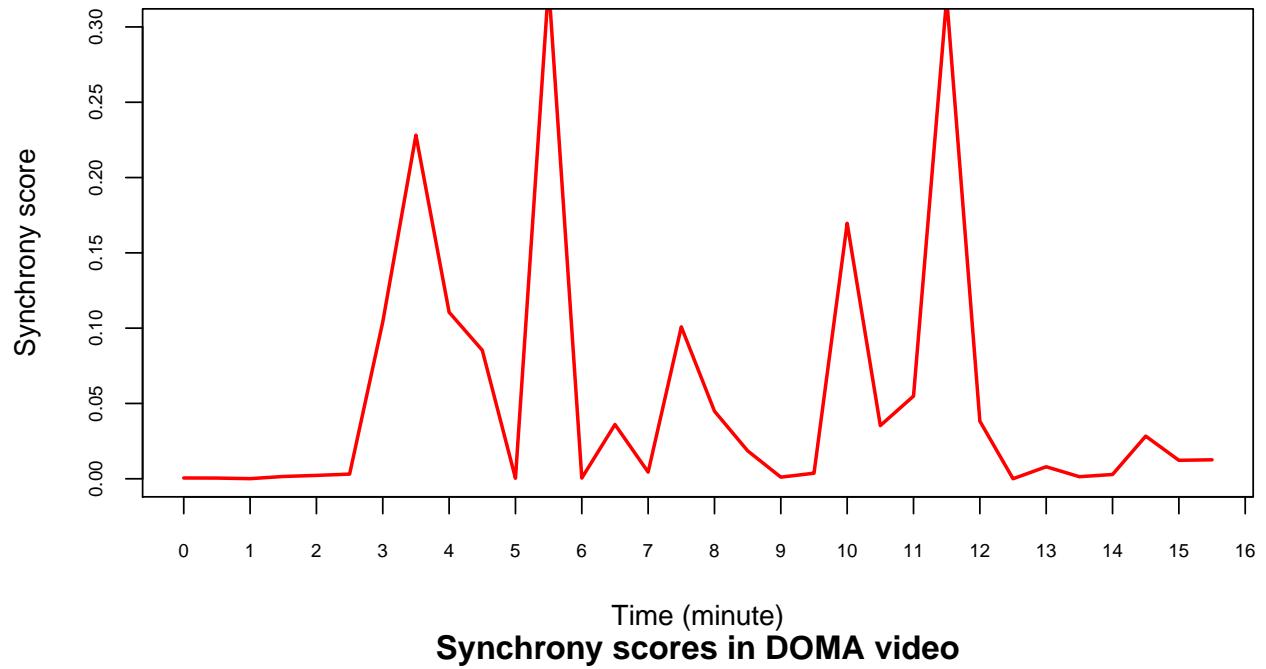
Synchrony scores in BRLO041 video



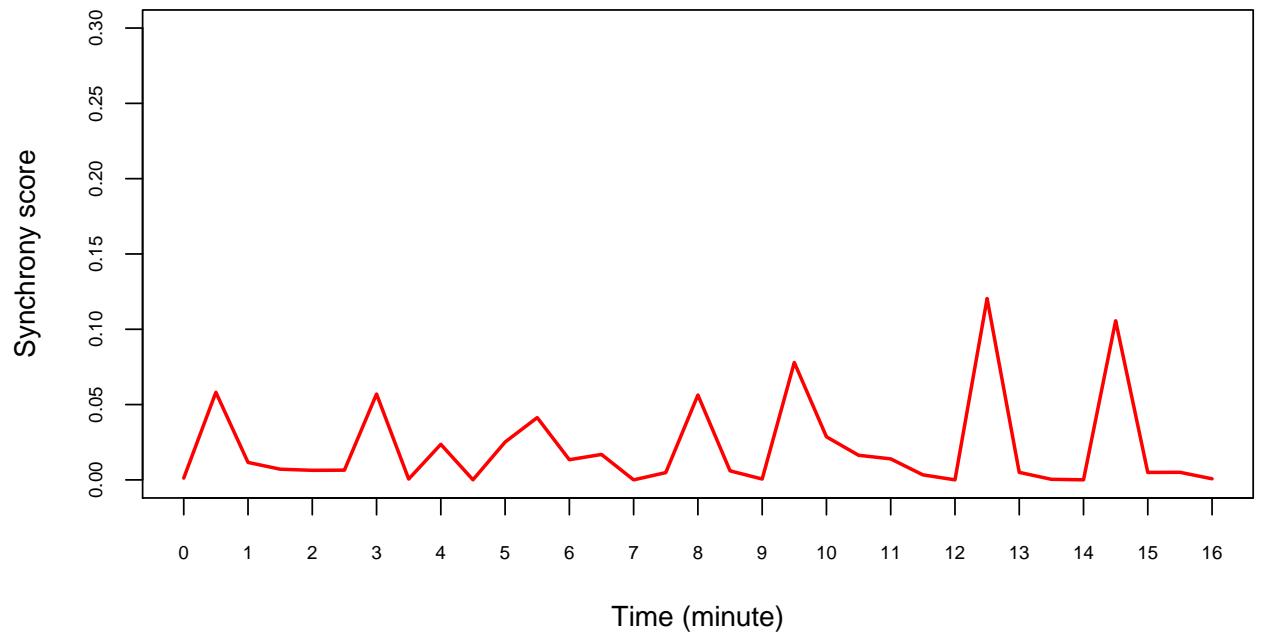
Synchrony scores in COLO022 video



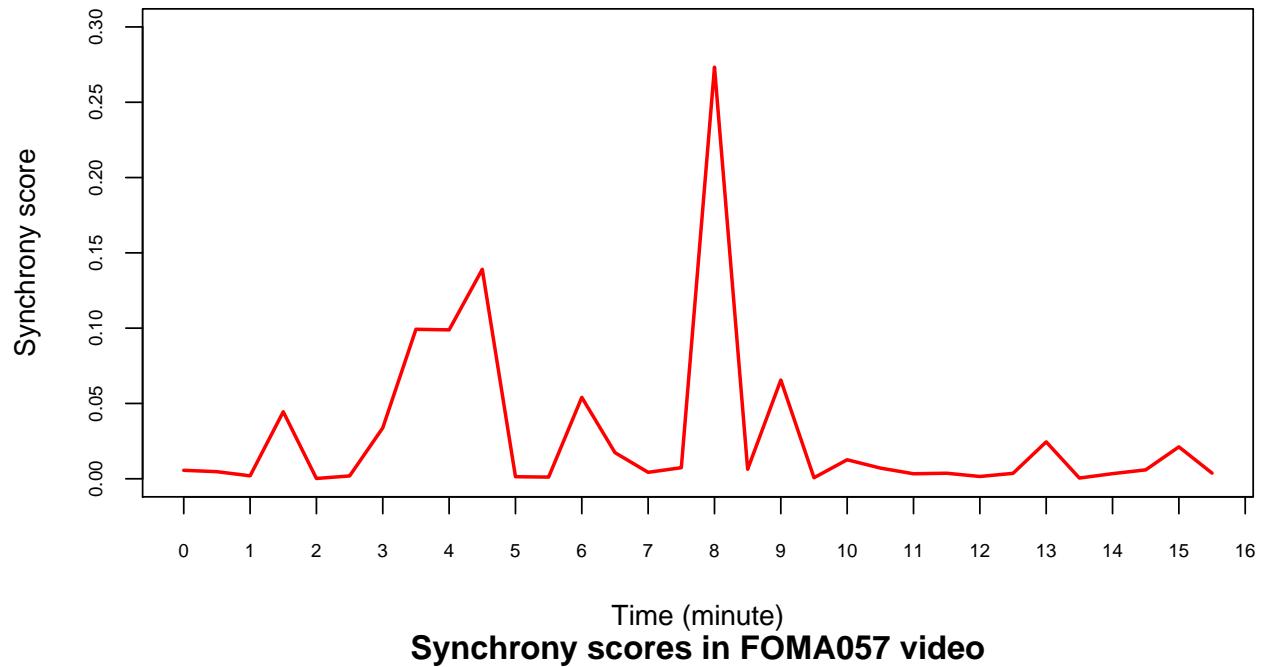
Synchrony scores in DIPE004 video



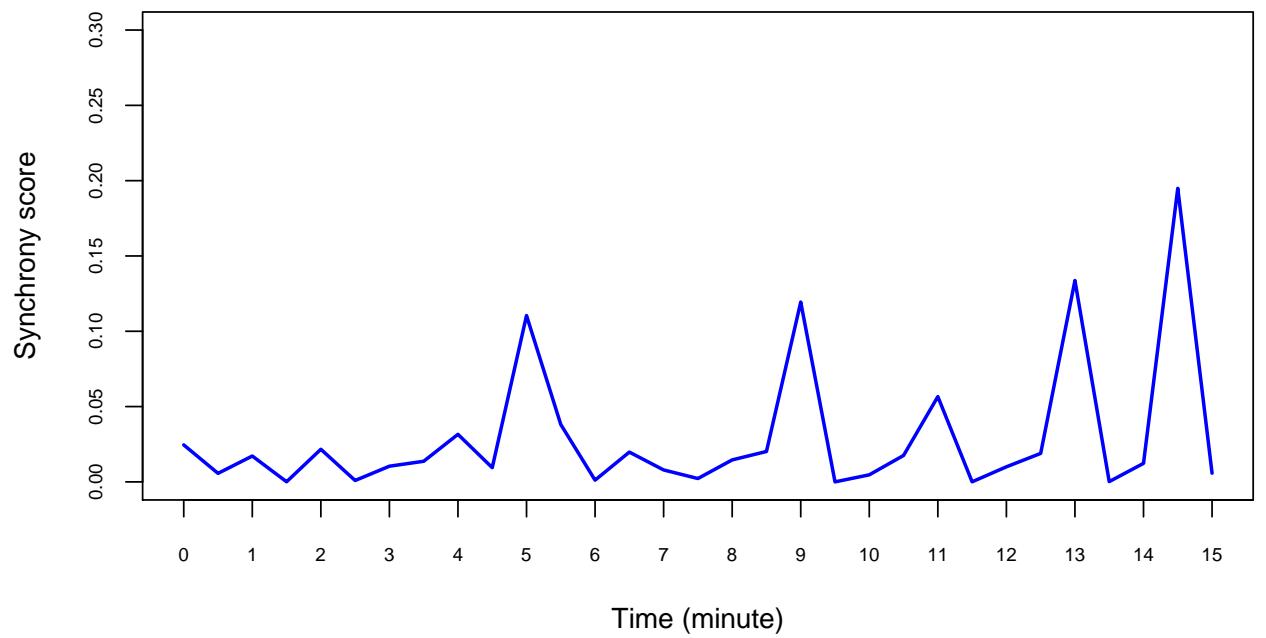
Synchrony scores in DOMA video



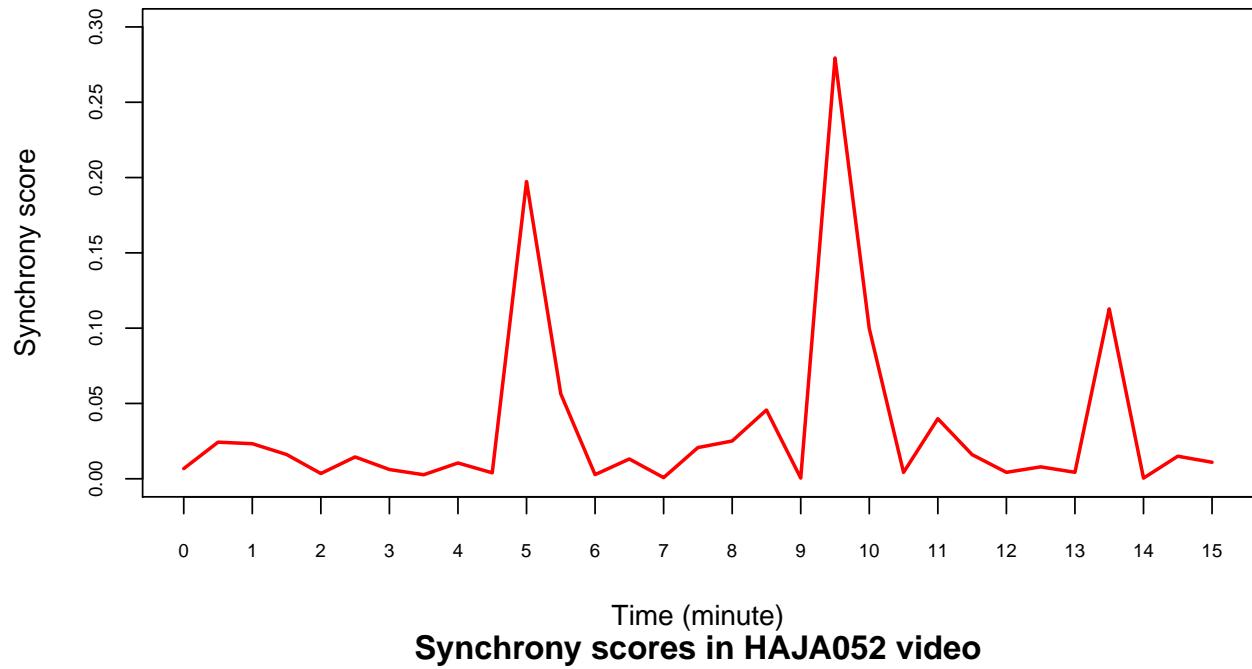
Synchrony scores in DRNE video



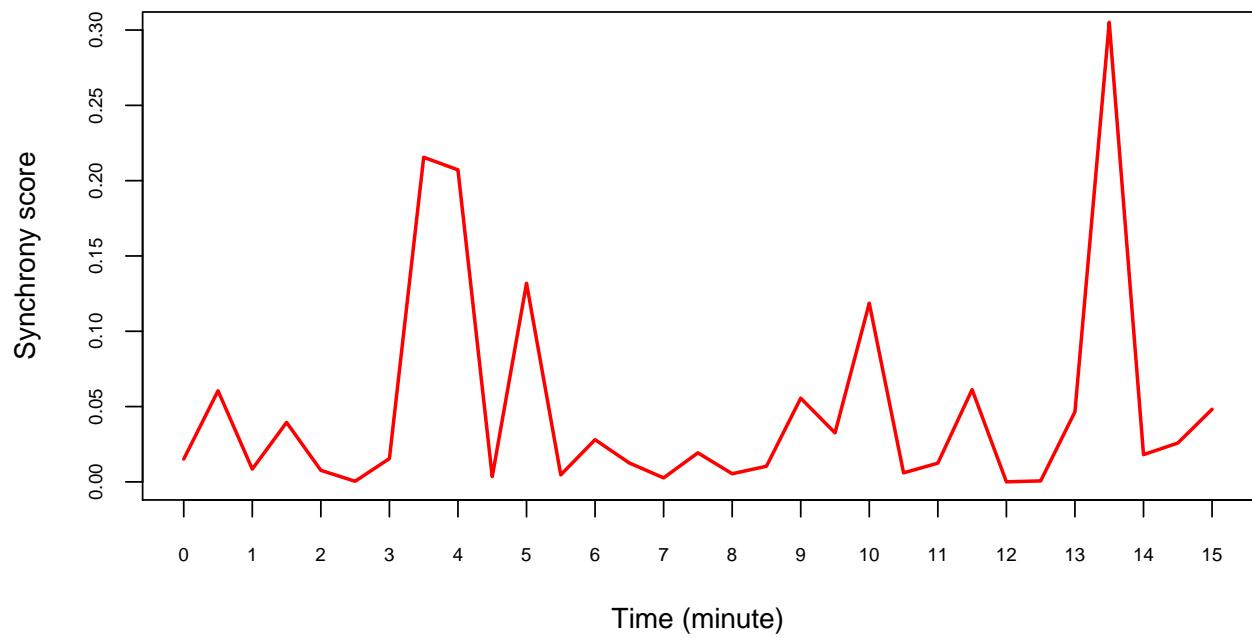
Synchrony scores in FOMA057 video



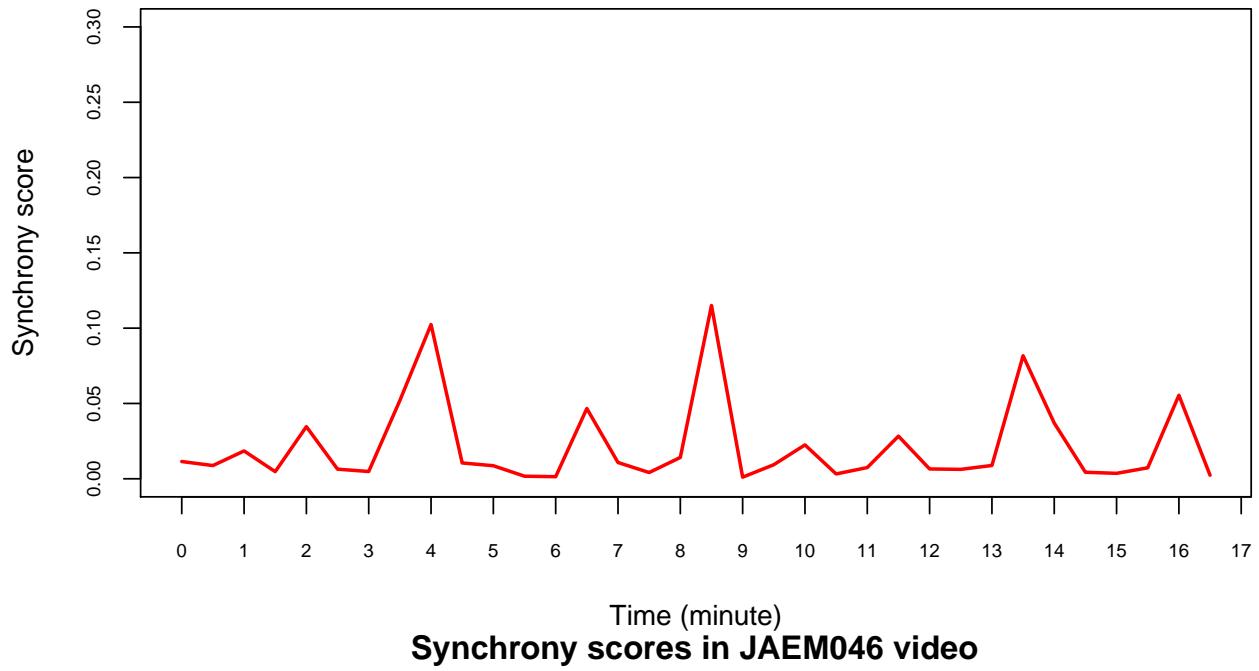
Synchrony scores in GROP039 video



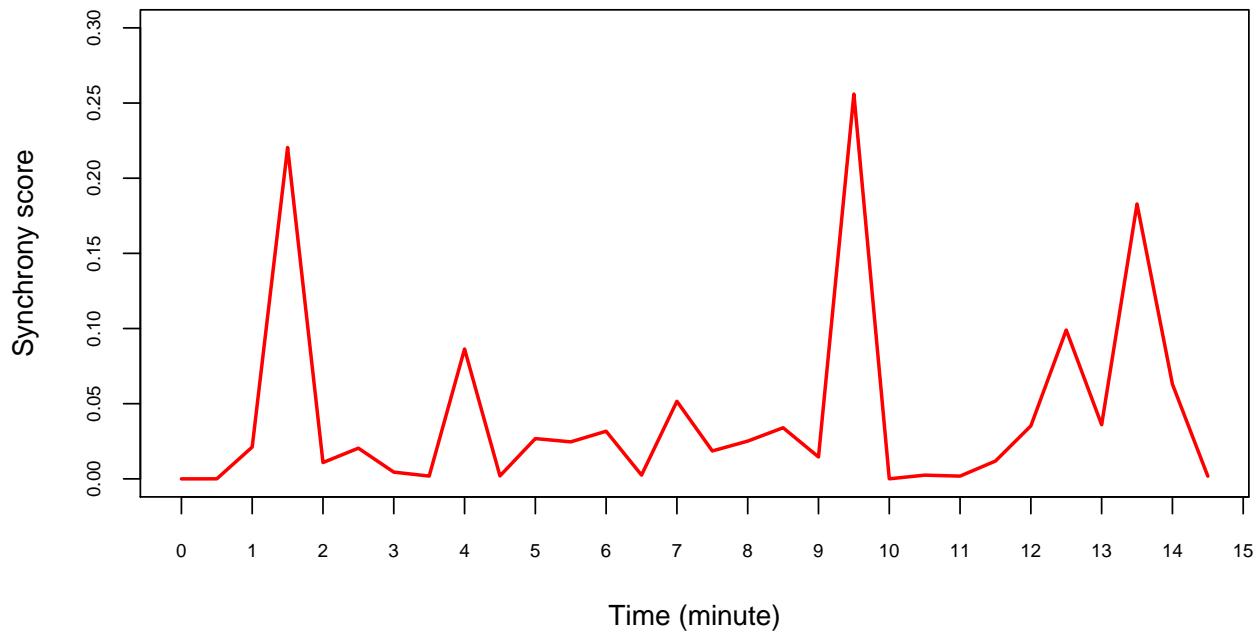
Synchrony scores in HAJA052 video



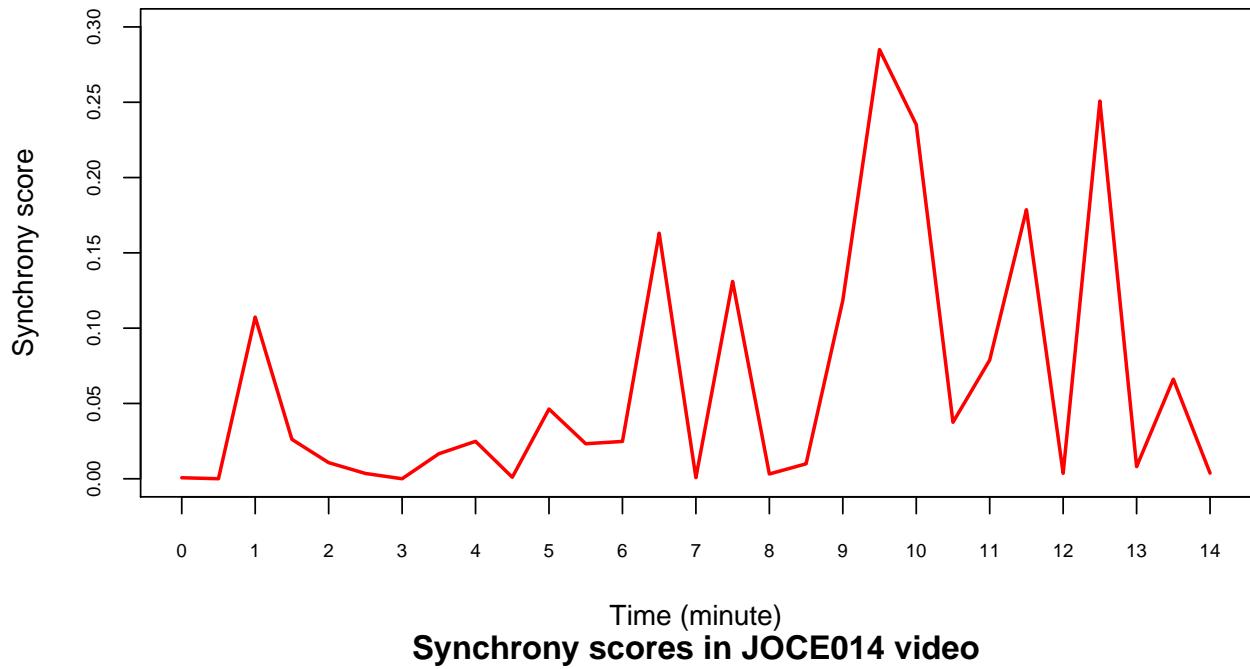
Synchrony scores in HUMA058 video



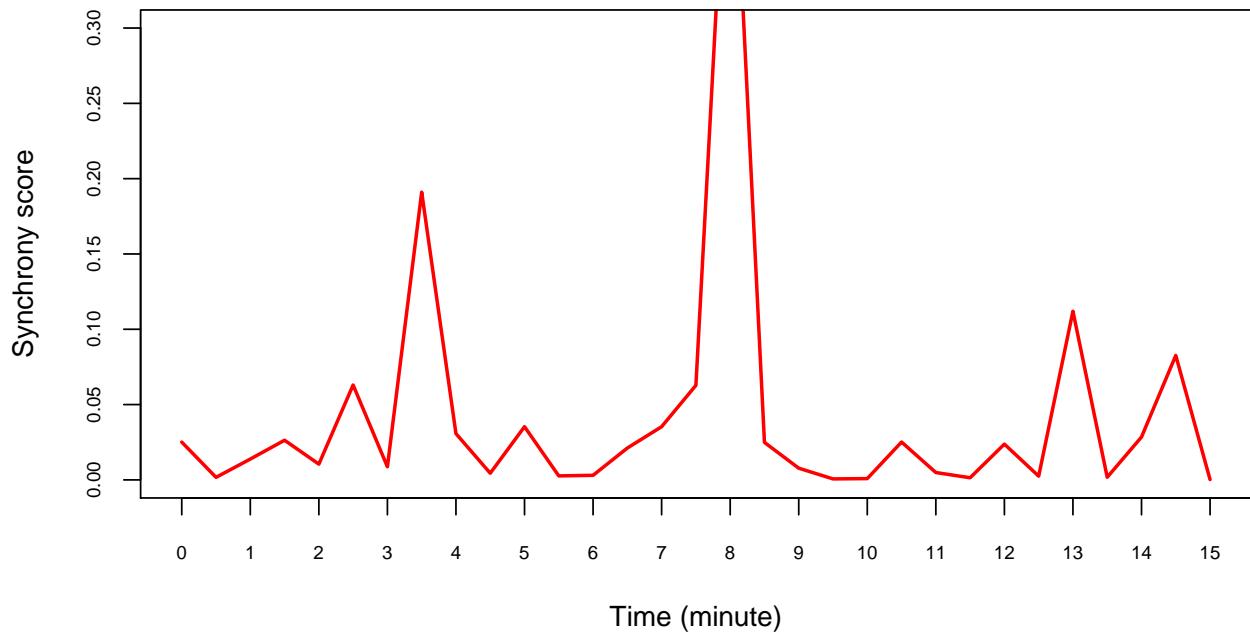
Synchrony scores in JAEM046 video



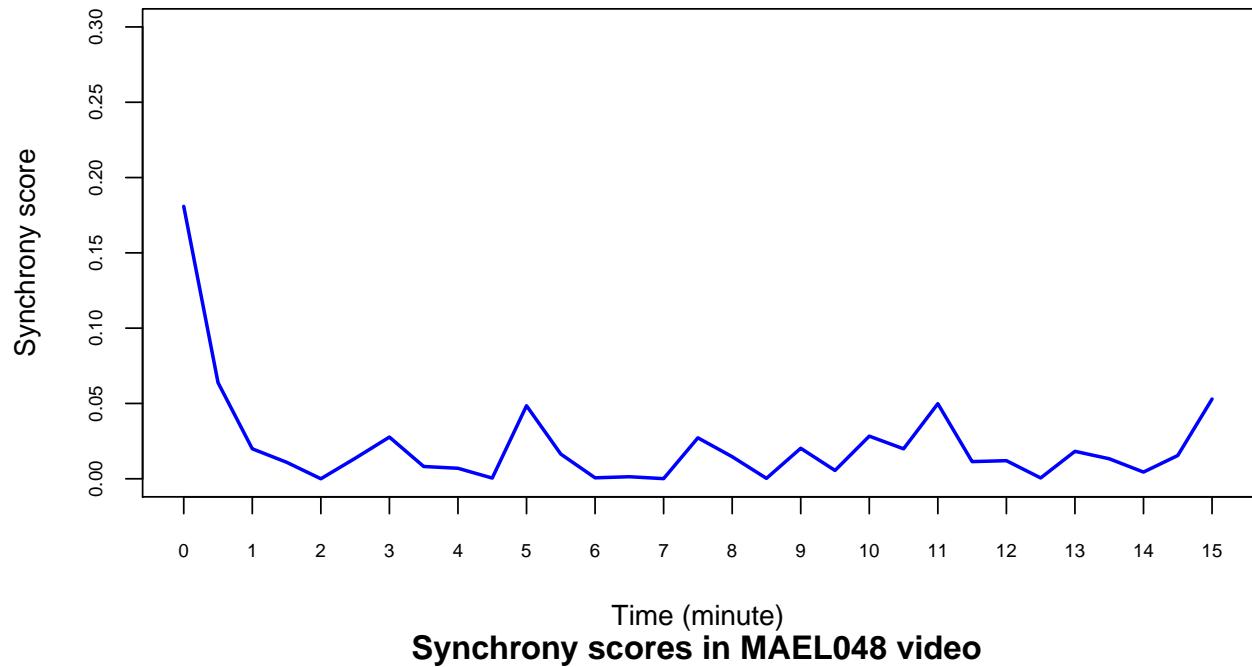
Synchrony scores in JEE0040 video



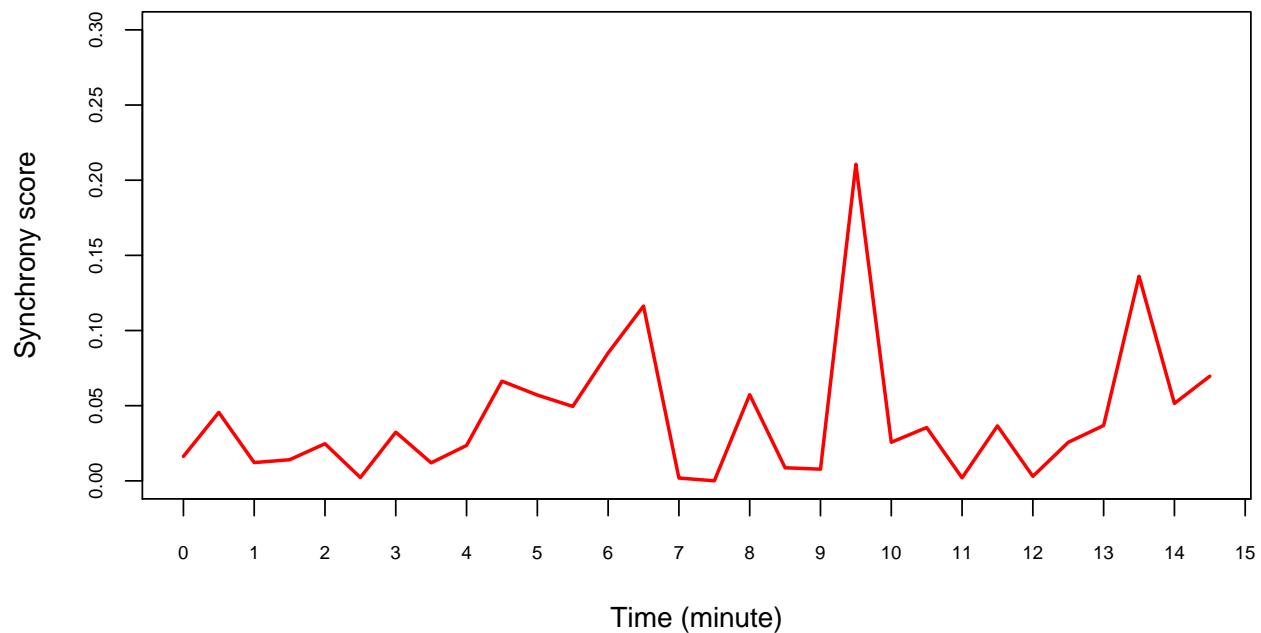
Synchrony scores in JOCE014 video



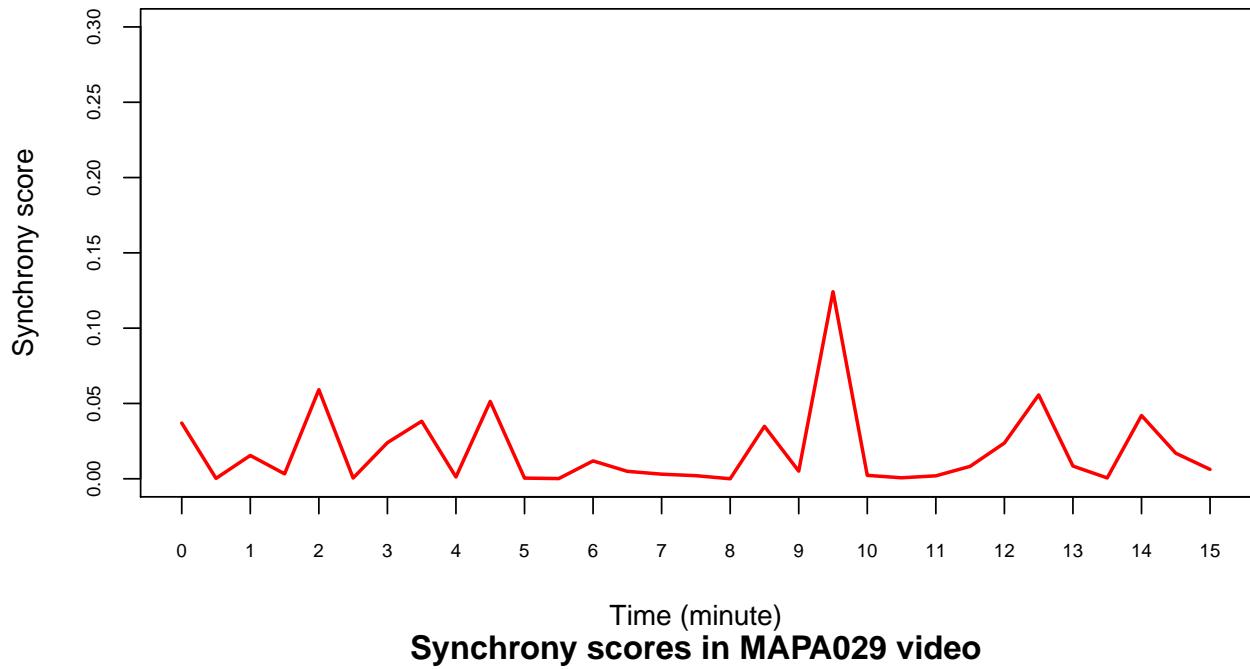
Synchrony scores in LACL video



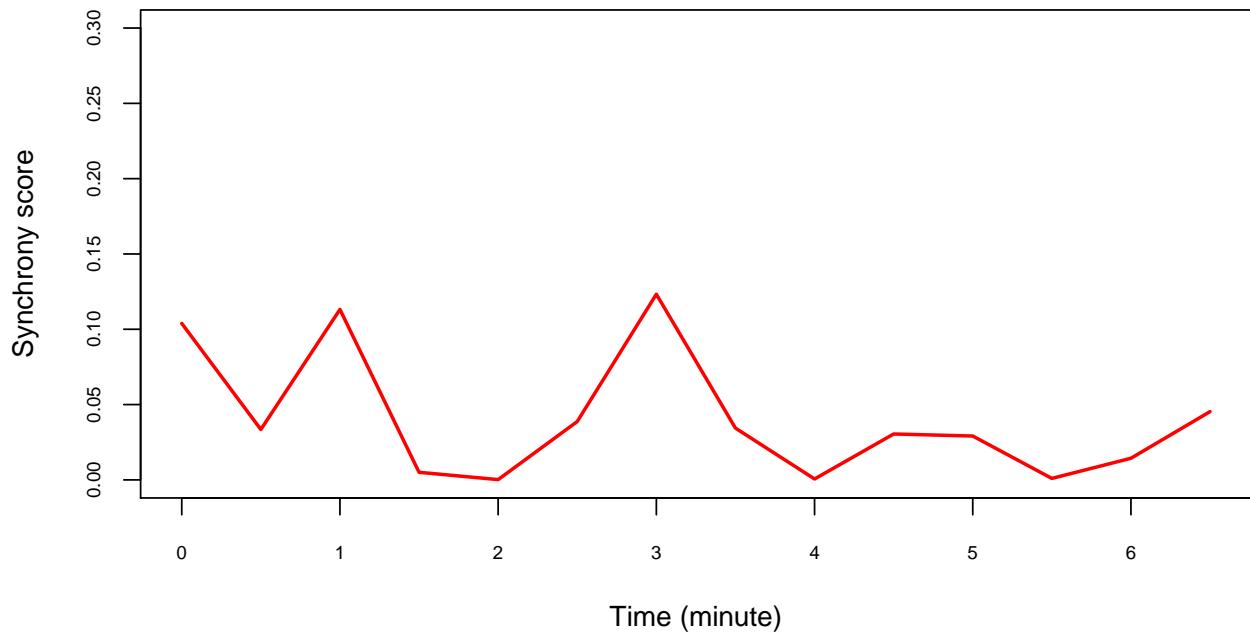
Synchrony scores in MAEL048 video



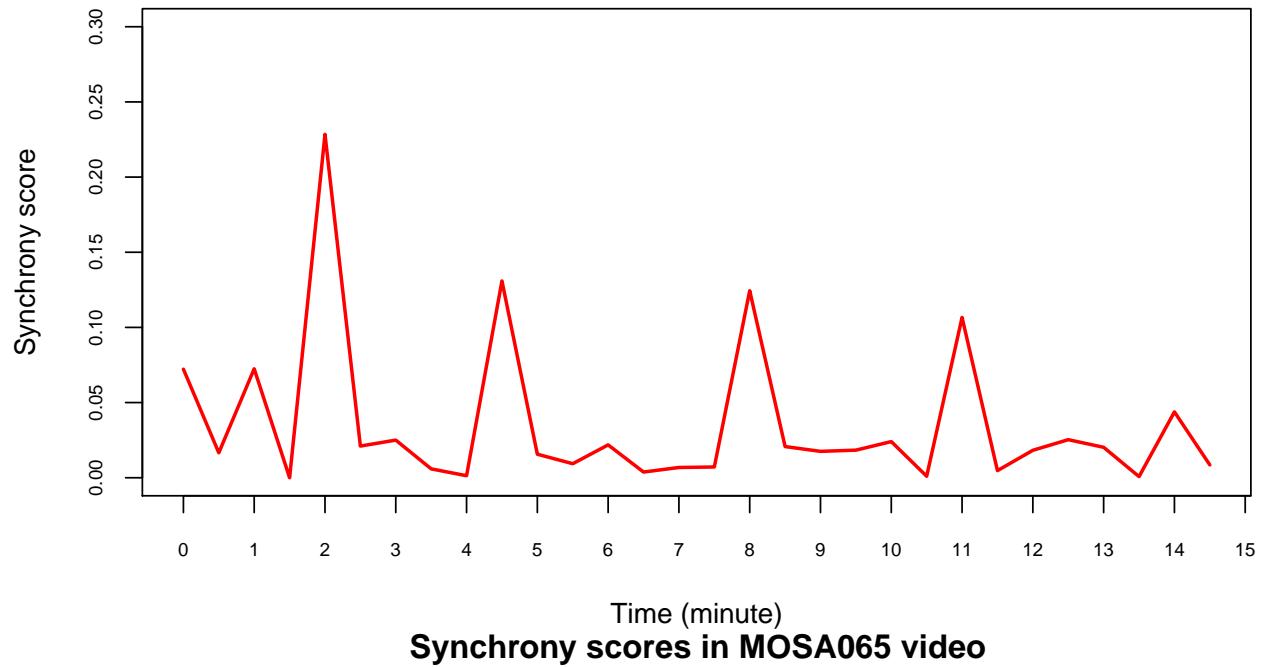
Synchrony scores in MAME20 video



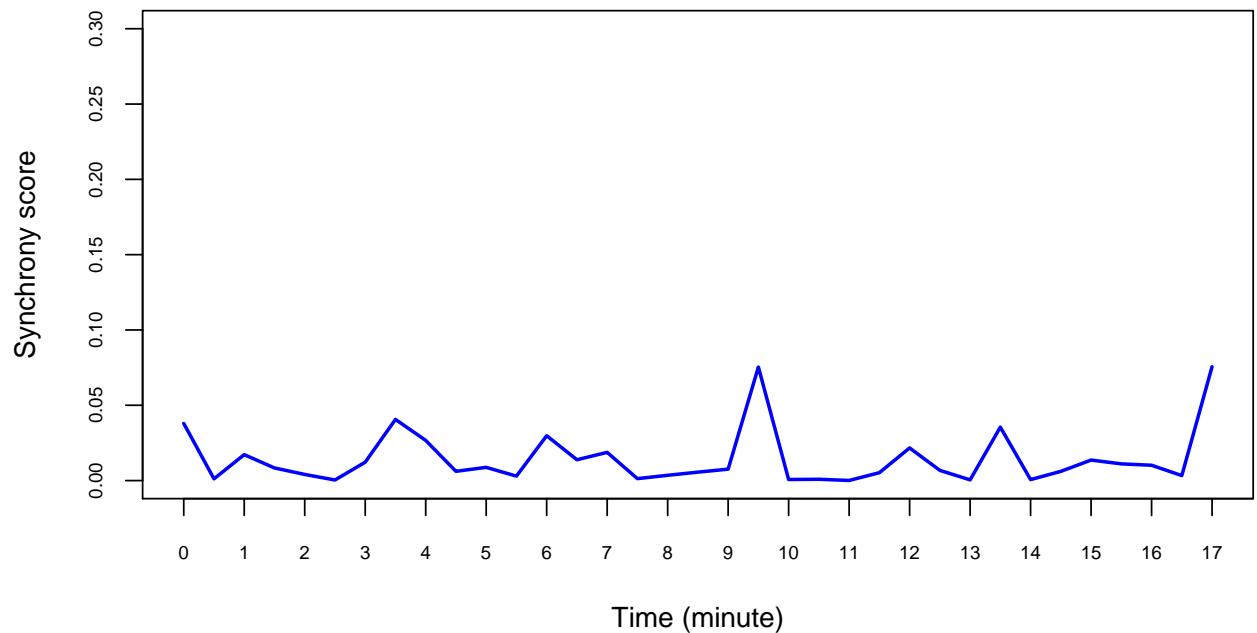
Synchrony scores in MAPA029 video



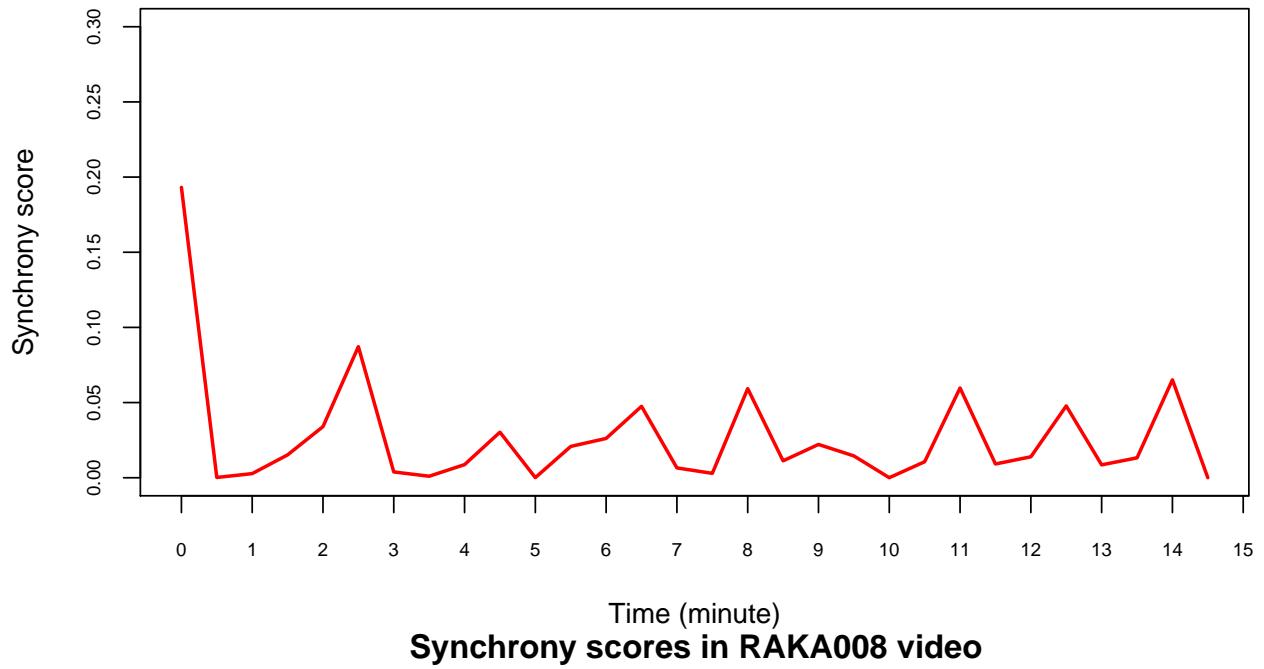
Synchrony scores in MIPH043 video



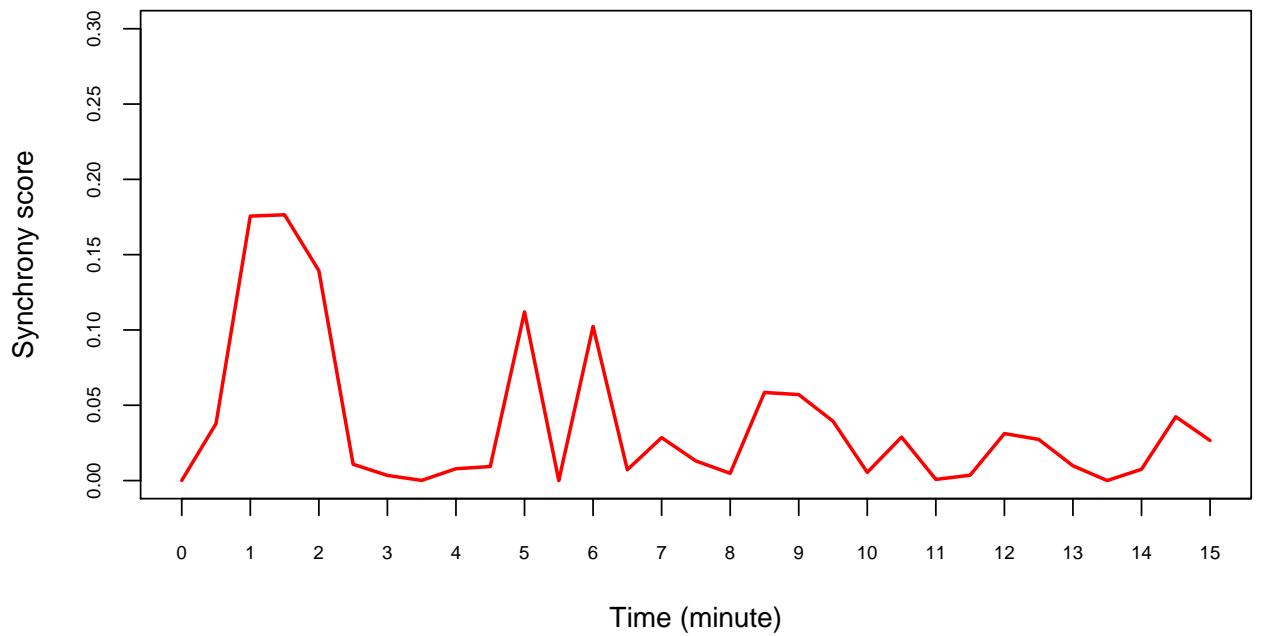
Synchrony scores in MOSA065 video



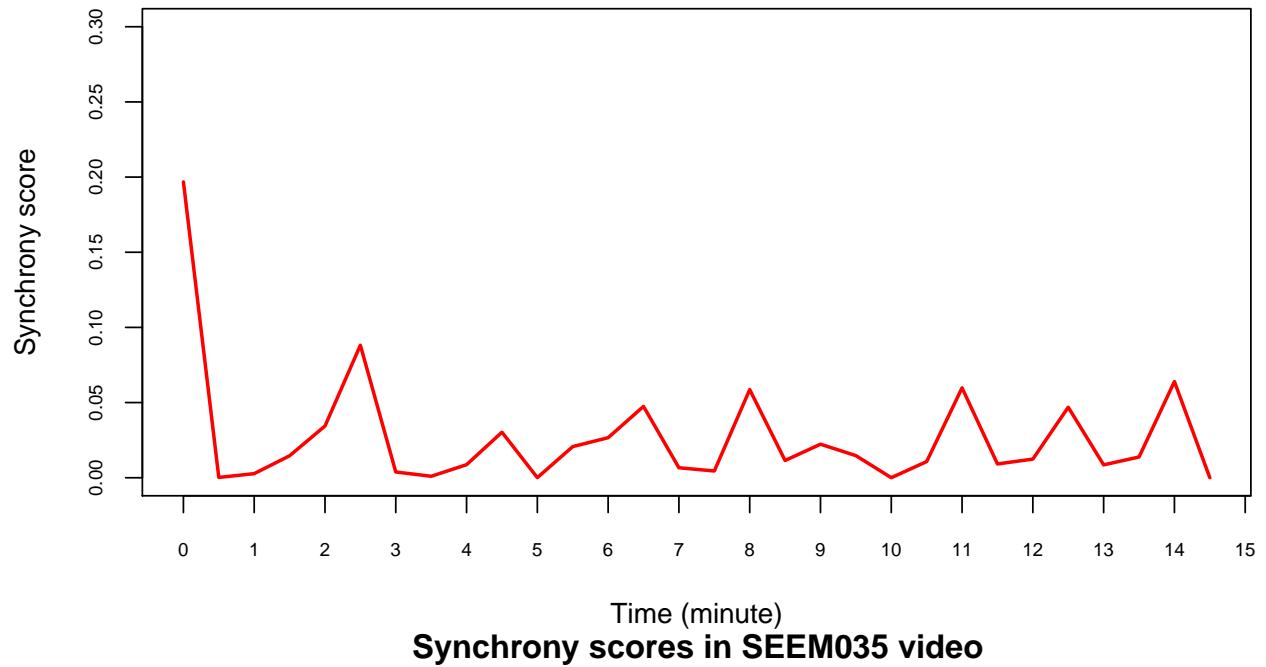
Synchrony scores in RAEM049 video



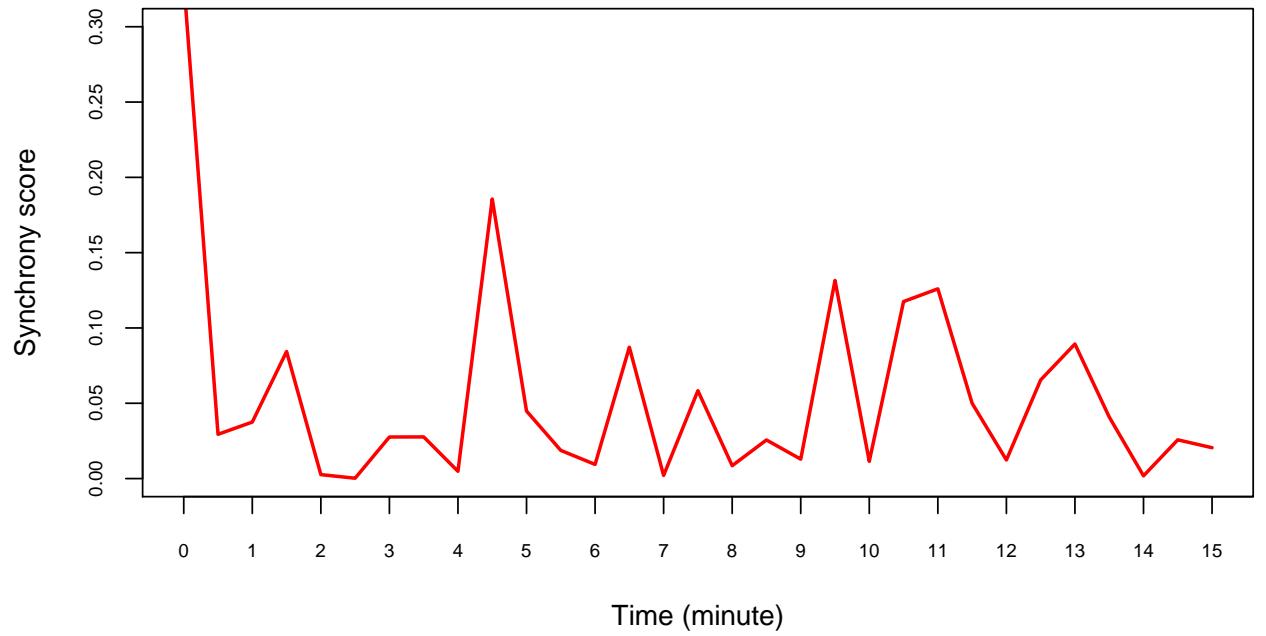
Synchrony scores in RAKA008 video



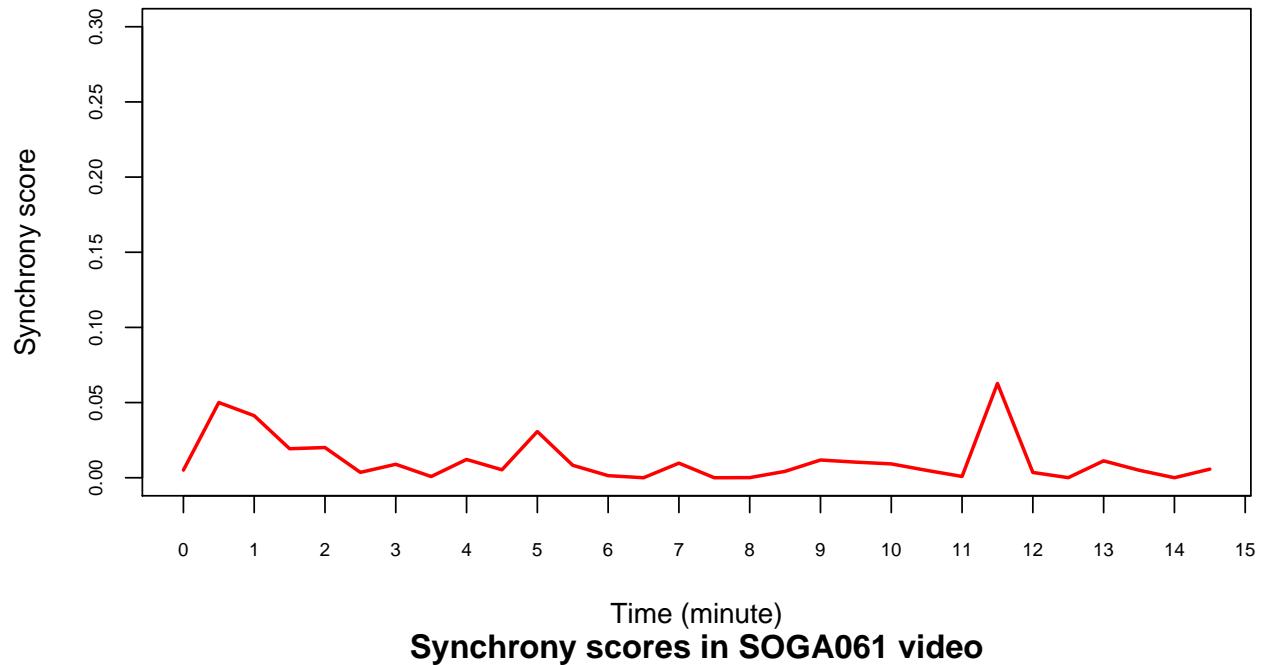
Synchrony scores in RIEM0 video



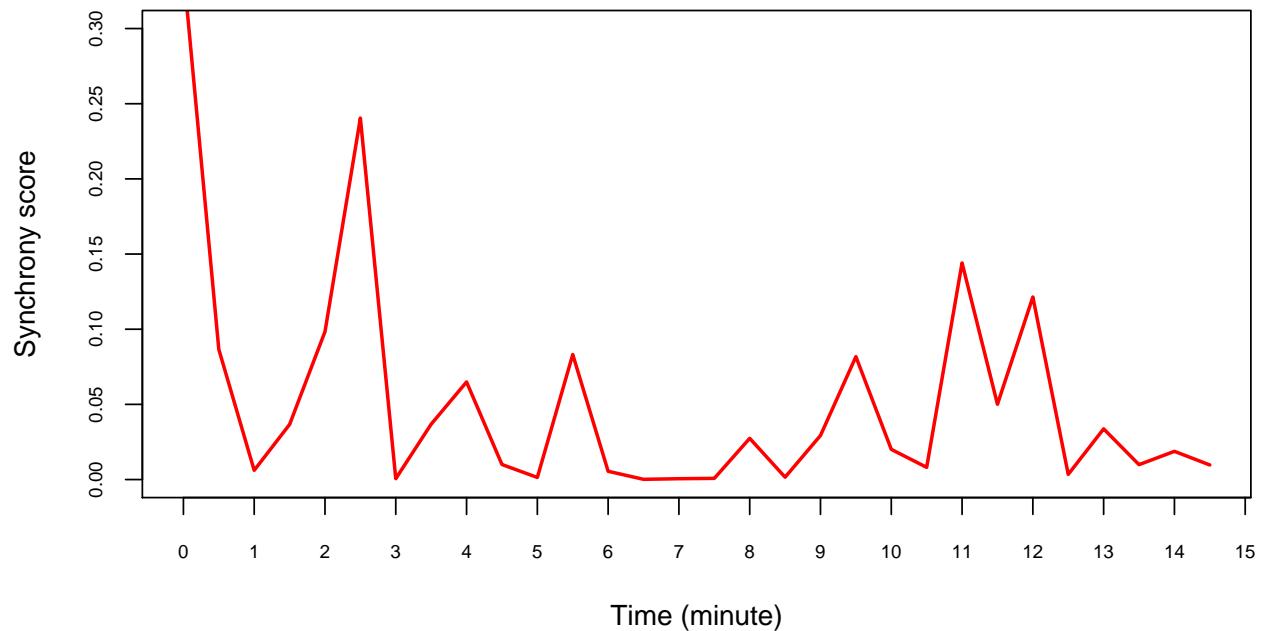
Synchrony scores in SEEM035 video



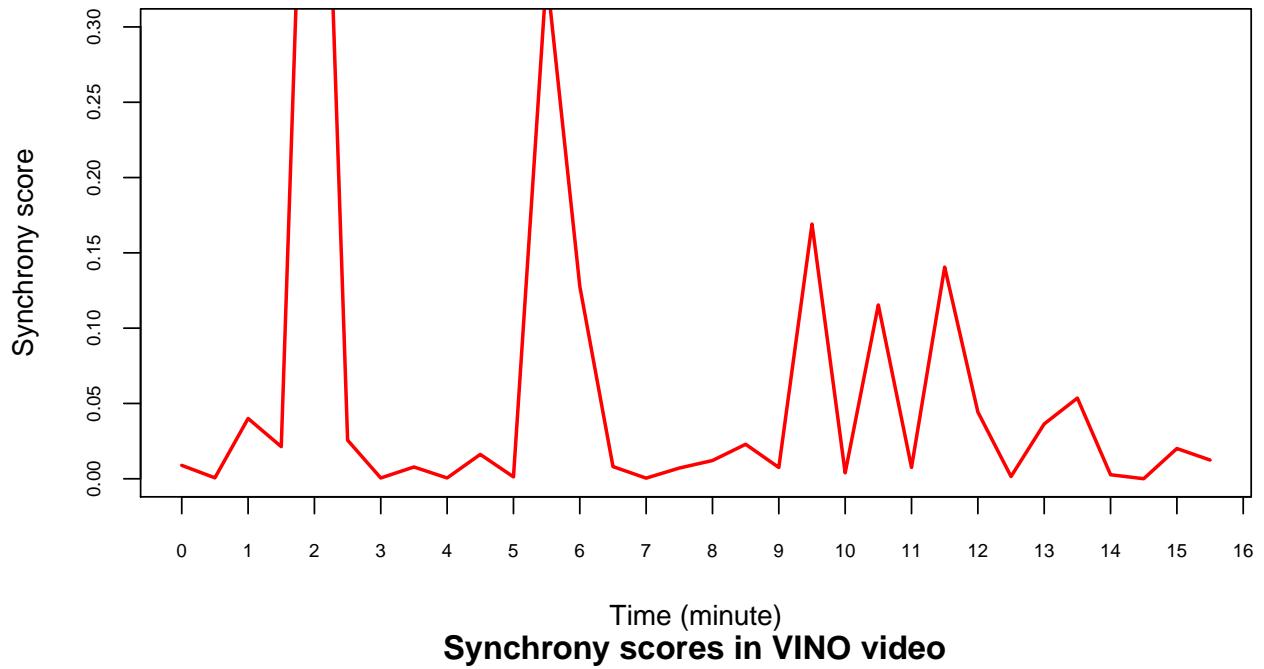
Synchrony scores in SHAN042 video



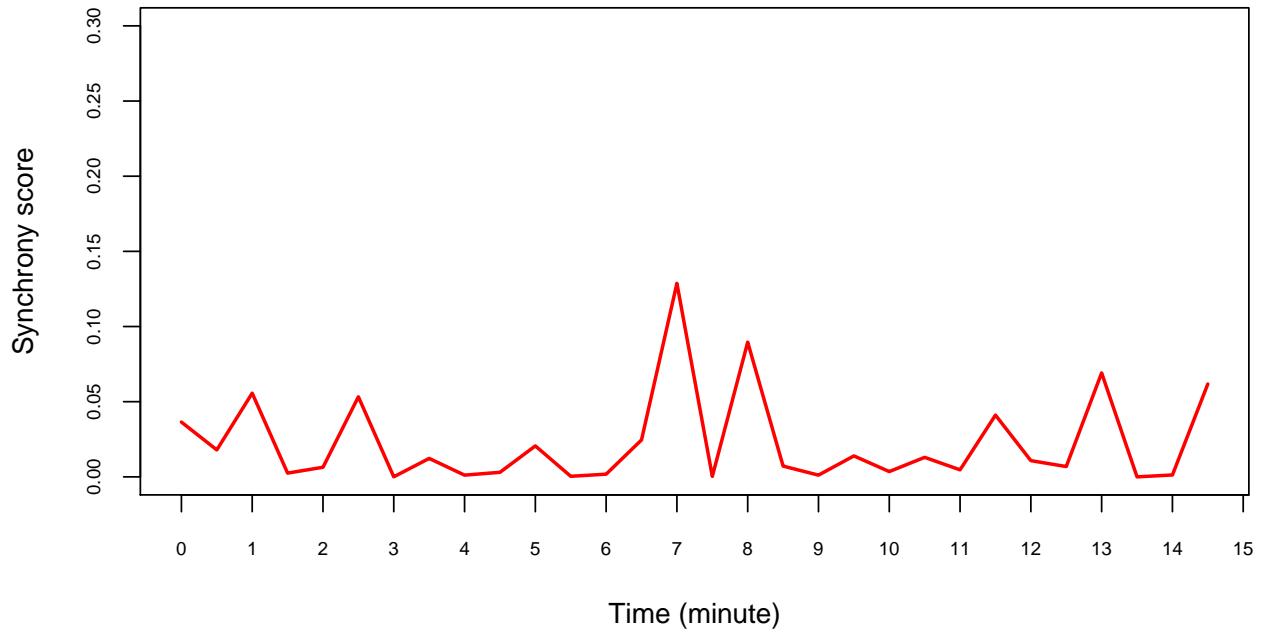
Synchrony scores in SOGA061 video



Synchrony scores in TIUG032 video



Synchrony scores in VINO video



```
#par(mar=c(4,4,4,3), mfrow=c(1,1))
#for (i in unique(SSInoLog$video))
#  {plot(SSInoLog[which(SSInoLog$video==i),]$Time_min,
#SSInoLog[which(SSInoLog$video==i),]$SSI_mo_ch, type="l", ylim=c(0, 0.3), #col=rainbow(4)[1], main=past
#lines(SSInoLog[which(SSInoLog$video==i),]$SSI_fa_mo_ch, col=rainbow(4)[2], lwd=2)
#lines(SSInoLog[which(SSInoLog$video==i),]$SSI_fa_ch, col=rainbow(4)[3], lwd=2)
#str(SSInoLog[which(SSInoLog$video==i),]$Time_min)
#str(SSInoLog[which(SSInoLog$video==i),]$SSI_mo_ch)
#str(SSInoLog[which(SSInoLog$video==i),]$SSI_fa_mo_ch)
```

```

#str(SSInoLog[which(SSInoLog$video==i),]$SSI_fa_ch]}

#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)}

cutFrames <- read.csv2("../Data/CSV/Cutframes.csv")
str(cutFrames)

## 'data.frame':   39 obs. of  7 variables:
##   $ indexList : Factor w/ 39 levels "1106","1606",...: 4 5 6 7 3 1 2 8 9 10 ...
##   $ CutBefore : Factor w/ 9 levels "00:06","00:07",...: NA NA NA NA NA NA NA 4 8 3 ...
##   $ CutMiddle1: Factor w/ 21 levels "04:26","04:43",...: NA NA NA NA NA NA NA 13 1 14 ...
##   $ CutMiddle2: Factor w/ 25 levels "05:02","05:08",...: NA NA NA NA NA NA NA 18 8 22 ...
##   $ CutFinal  : Factor w/ 24 levels "14:43","15:04",...: NA NA NA NA NA NA 23 13 18 ...
##   $ ChildSex  : Factor w/ 2 levels "Female","Male": NA NA NA NA NA NA 1 1 2 ...
##   $ ParentSex : Factor w/ 2 levels "Female","Male": NA NA NA NA NA NA 2 1 1 ...

#cutFrames$CutBefore
cutFramesCB <- strftime(cutFrames$CutBefore, format="%M:%S")
cutFrames$CutBefore <- as.character(cutFrames$CutBefore)
#cutFrames$CutBefore
cutFramesCB <- strsplit(cutFrames$CutBefore, split=":")

Cut <- c()
for (i in 1:nrow(cutFrames)){
  CutBeforeAlone <- (as.numeric(cutFramesCB[i][[1]][1]) + as.numeric(cutFramesCB[i][[1]][2])/60)
  Cut <- c(Cut, CutBeforeAlone)
}
cutFrames$CutBeforeMin <- Cut
Cut <- c()
#View(cutFrames)

Cut <- c()
cutFrames$CutMiddle1 <- as.character(cutFrames$CutMiddle1)
cutMiddleSplit <- strsplit(cutFrames$CutMiddle1, split=":")
for (i in 1:nrow(cutFrames)){
  CutAlone <- (as.numeric(cutMiddleSplit[i][[1]][1]) + as.numeric(cutMiddleSplit[i][[1]][2])/60)
  Cut <- c(Cut, CutAlone)
  # print(CutAlone)
}
cutFrames$CutMiddle1Min <- Cut
Cut <- c()
#cutFrames$CutMiddle1
#cutFrames$CutMiddle1Min

Cut <- c()
cutFrames$CutMiddle2 <- as.character(cutFrames$CutMiddle2)
cutMiddleSplit <- strsplit(cutFrames$CutMiddle2, split=":")

```

```

for (i in 1:nrow(cutFrames)){
  CutAlone <- (as.numeric(cutMiddleSplit[i][[1]][1]) + as.numeric(cutMiddleSplit[i][[1]][2])/60)
  Cut <- c(Cut, CutAlone)
  print(CutAlone)
}

## [1] NA
## [1] 5.933333
## [1] 5.483333
## [1] 6.116667
## [1] 5.666667
## [1] 5.133333
## [1] 5.533333
## [1] 5.933333
## [1] 5.8
## [1] 6.15
## [1] 5.866667
## [1] 5.983333
## [1] 5.783333
## [1] 5.55
## [1] 6.033333
## [1] 6.1
## [1] 5.233333
## [1] 5.033333
## [1] 5.483333
## [1] 5.6
## [1] 5.733333
## [1] 6.566667
## [1] NA
## [1] 6.266667
## [1] NA
## [1] NA
## [1] NA
## [1] 5.316667
## [1] 5.283333
## [1] 5.183333
## [1] 5.583333
## [1] 5.983333
## [1] 5.333333

cutFrames$CutMiddle2Min <- Cut
Cut <- c()
#cutFrames$CutMiddle2
#cutFrames$CutMiddle2Min

Cut <- c()
cutFrames$CutFinal <- as.character(cutFrames$CutFinal)
cutSplit <- strsplit(cutFrames$CutFinal, split=":")

```

```

for (i in 1:nrow(cutFrames)){
  CutAlone <- (as.numeric(cutSplit[i][[1]][1]) + as.numeric(cutSplit[i][[1]][2])/60)
  Cut <- c(Cut, CutAlone)
  print(CutAlone)
}

## [1] NA
## [1] 16.73333
## [1] 15.5
## [1] 16.08333
## [1] 15.46667
## [1] 15.13333
## [1] 15.21667
## [1] 15.3
## [1] 16.16667
## [1] 16.1
## [1] 16.41667
## [1] 16.06667
## [1] 15.5
## [1] 15.41667
## [1] 15.5
## [1] 16.83333
## [1] 15.11667
## [1] 14.71667
## [1] 15.35
## [1] 15.25
## [1] 15.06667
## [1] 15.9
## [1] NA
## [1] 16.43333
## [1] NA
## [1] NA
## [1] NA
## [1] 15.16667
## [1] 15.56667
## [1] 15.3
## [1] 15.21667
## [1] 15.88333
## [1] 15.1

cutFrames$CutFinalMin <- Cut
Cut <- c()
#cutFrames$CutFinal
#cutFrames$CutFinalMin
#View(cutFrames)

str(SSInoLog)

## 'data.frame': 1200 obs. of 8 variables:

```

```

## $ video      : Factor w/ 40 levels "1106","1606",...: 4 4 4 4 4 4 4 4 4 ...
## $ 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 0.5 1 1.5 2 2.5 3 3.5 4 4.5 ...
## $ SSI_mo_ch  : num  0.018627 0.024666 0.04506 0.001527 0.000817 ...
## $ SSI_fa_ch  : num  NA NA NA NA NA NA NA NA NA ...
## $ SSI_fa_mo  : num  NA NA NA NA NA NA NA NA NA ...
## $ SSI_fa_mo_ch: num  NA NA NA NA NA NA NA NA NA ...

a <- merge(SSInoLog, cutFrames, by.x="video", by.y="indexList")
#View(a)
a$LabelVideo <- rep(NA, nrow(a))
a[which(a$Time_min < a$CutBeforeMin),]$LabelVideo <- "Cut"

a[which(a$Time_min > a$CutBeforeMin & a$Time_min < a$CutMiddle1Min),]$LabelVideo <- "No Conflict"
a[which(a$Time_min > a$CutMiddle1Min & a$Time_min < a$CutMiddle2Min),]$LabelVideo <- "Cut"
a[which(a$Time_min > a$CutMiddle2Min & a$Time_min < a$CutFinalMin),]$LabelVideo <- "Conflict"
#a[which(a$Time_min >= a$CutFinalMin),]$LabelVideo <- "Cut"

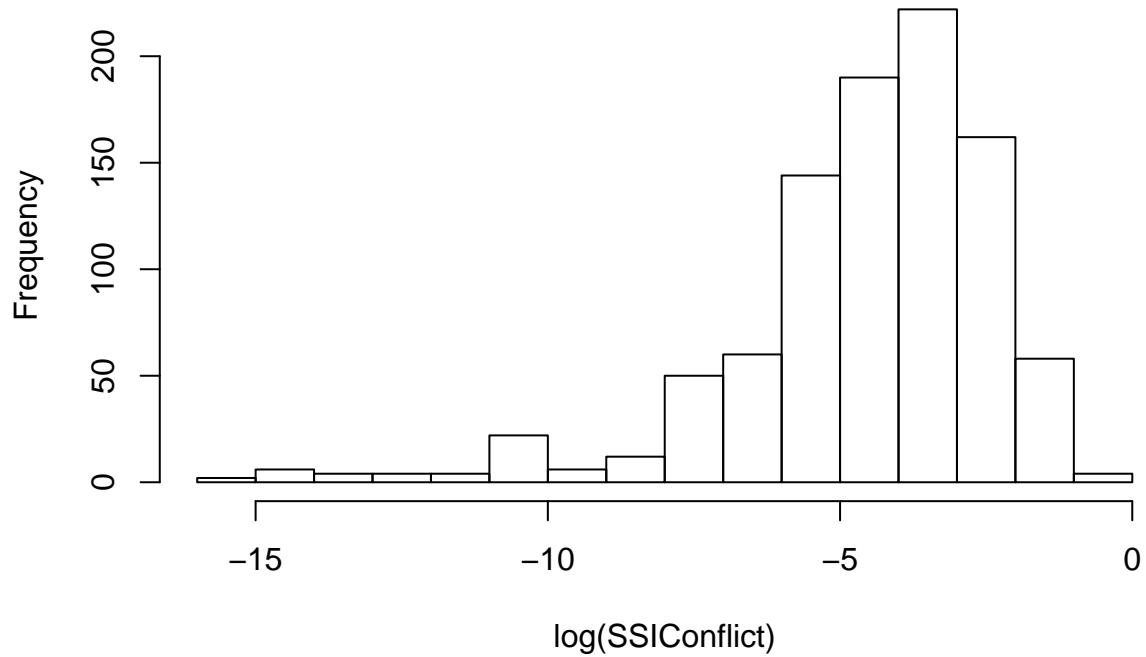
SSIConflict <- c(a[which(a$LabelVideo=="Conflict")],$SSI_mo_ch, a[which(a$LabelVideo=="Conflict")],$SSI...
mean(SSIConflict, na.rm=TRUE)

## [1] 0.03921076
SSINoConflict <- c(a[which(a$LabelVideo=="No Conflict")],$SSI_mo_ch, a[which(a$LabelVideo=="No Conflict")...
mean(SSINoConflict, na.rm=TRUE)

## [1] 0.04026161
hist(log(SSIConflict))

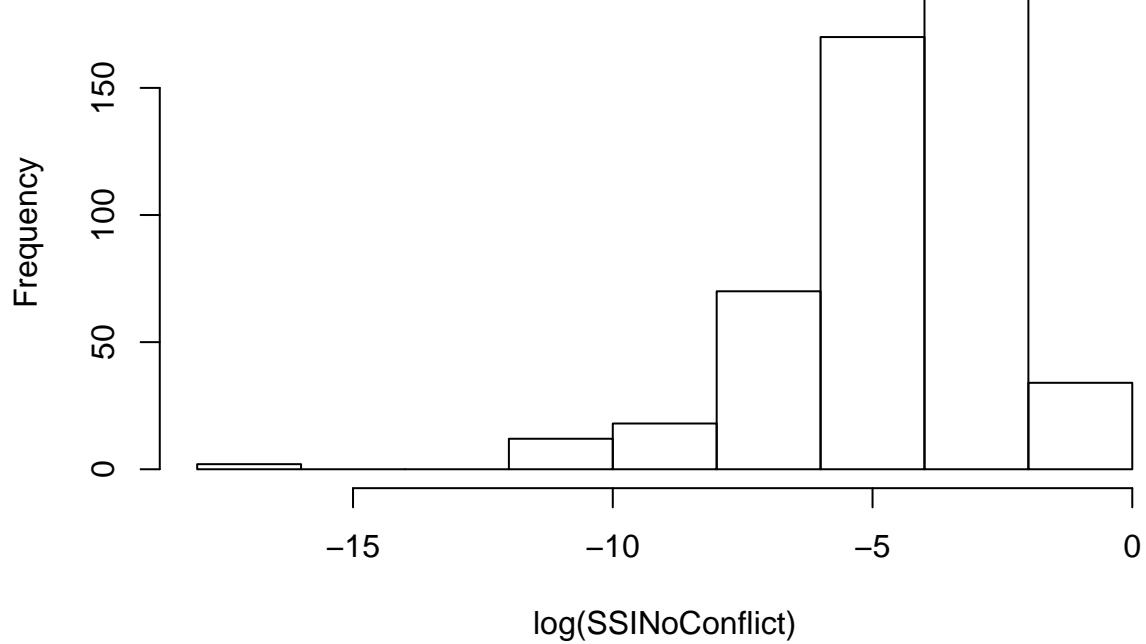
```

Histogram of log(SSIConflict)



```
hist(log(SSINoConflict))
```

Histogram of log(SSINoConflict)

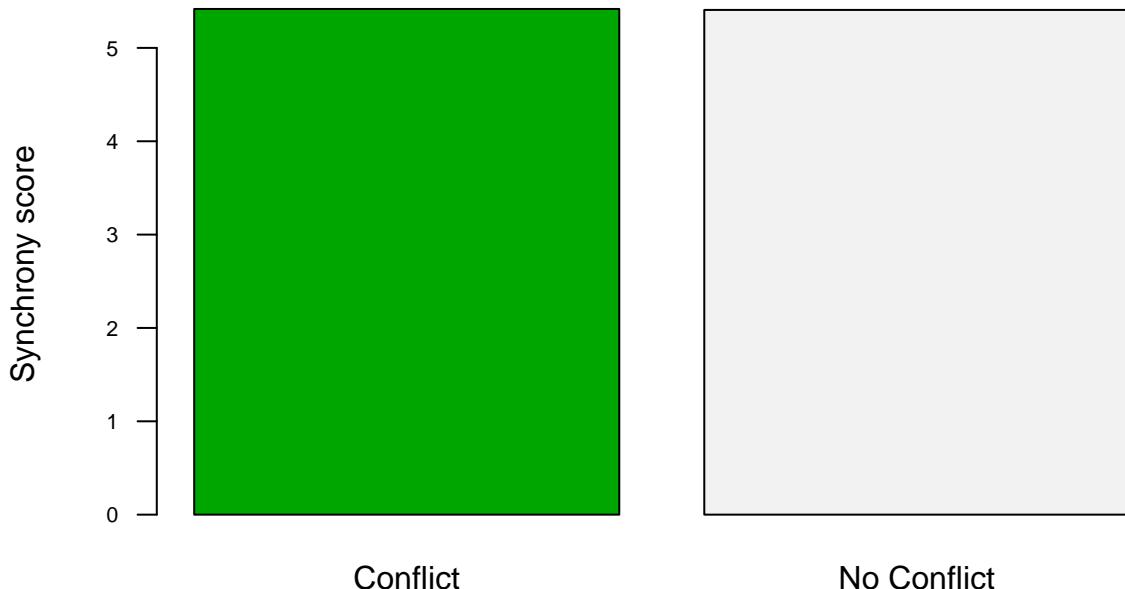


```
mean(log(SSINoConflict), na.rm=TRUE)
```

```
## [1] -4.582897
```

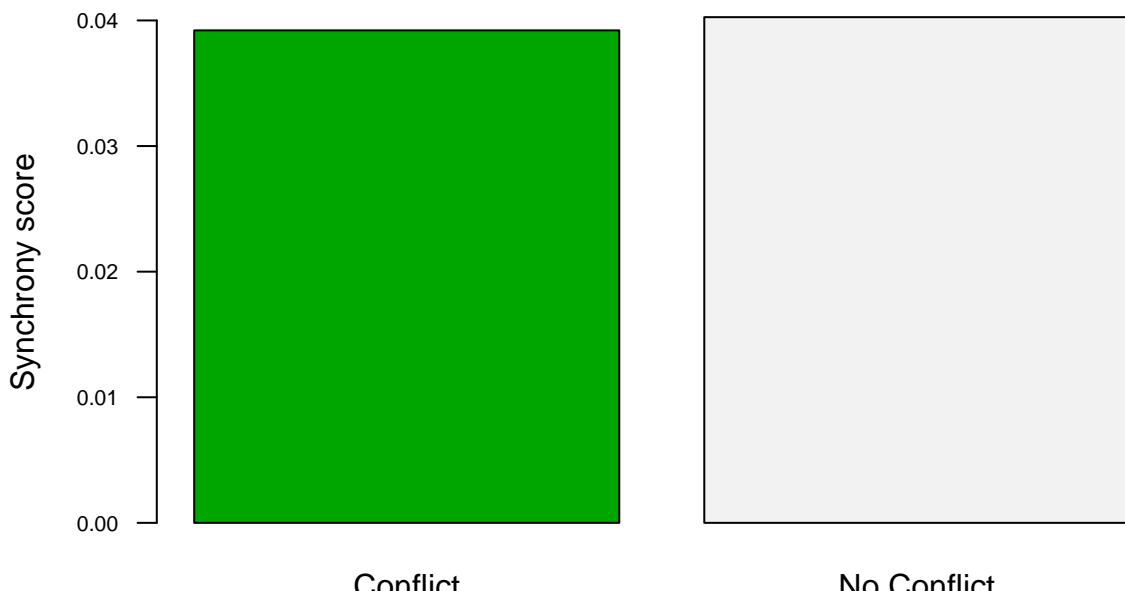
```
barplot(c(mean(log(SSIConflict), na.rm=TRUE)+10, mean(log(SSINoConflict), na.rm=TRUE)+10), names=c("Conflict", "No Conflict"))
```

Synchrony scores between 2 situations



```
barplot(c(mean(SSIConflict, na.rm=TRUE), mean(SSINoConflict, na.rm=TRUE)), names=c("Conflict", "No Conflict"))
```

Synchrony scores in 2 situations



```
df <- data.frame(NA, NA)
for (i in unique(a$video)){
  # print (i)
  b <- mean(a[which(a$LabelVideo=="No Conflict" & a$video==i),]$SSI_mo_ch)
  #print(b)
  e <- mean(a[which(a$LabelVideo=="Conflict" & a$video==i),]$SSI_mo_ch)
```

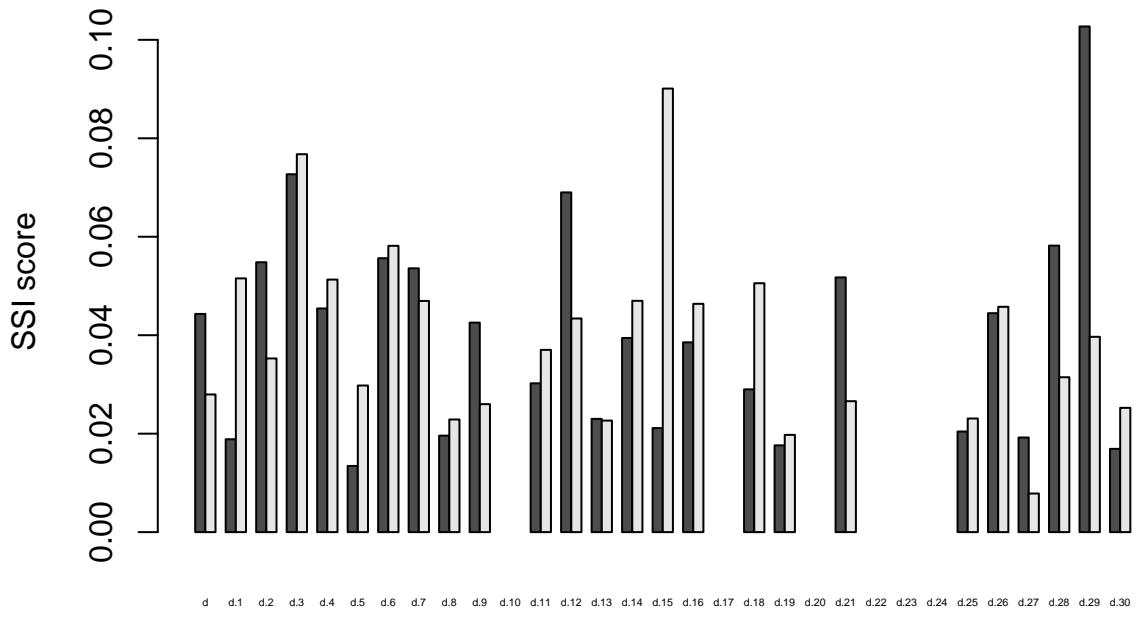
```

# print(c)
d <- c(b, e)
df <- cbind(df,d)
# print(df)

df <- df[,-(1:10)]

barplot(as.matrix(df), beside=TRUE, xlab="Video", ylab="SSI score", cex.names=0.3)

```

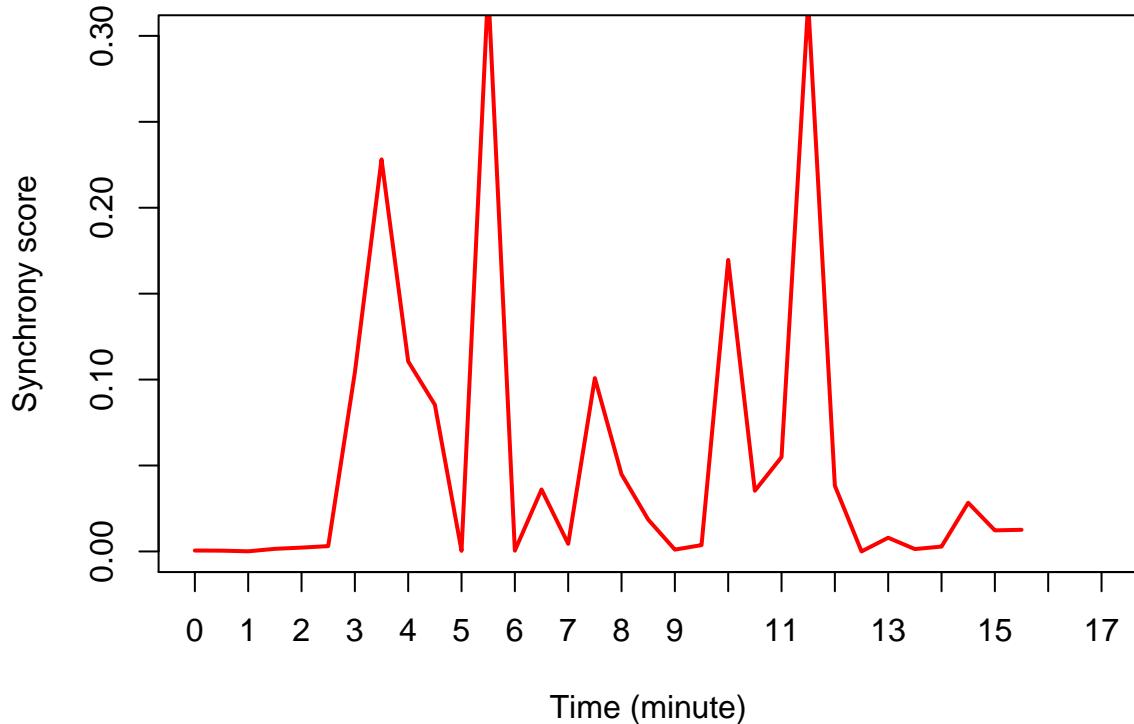


```

par(mar=c(4,4,4,3), mflow=c(1,1))
plot(SSInoLog[which(SSInoLog$video=="DIPE004"),]$Time_min, SSInoLog[which(SSInoLog$video=="DIPE004"),]$Time_min,
     DIPE004 video", xlab = "Time (minute)", ylab="Synchrony score", lwd=2, xaxp=c(0,length(SSInoLog$Time_min),

```

Synchrony scores for child–mother dyad in DIPE004 video



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(SI)[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(SI)){
  IntervalNumbersVideo <- ceiling(length(SI[,indexSSI])/60)
  SIColumn <- SI[,indexSSI]
  SITenMinute <- c()
  for (i in 1:IntervalNumbersVideo){
    borneInf <- 1+(i-1)*60
    borneSup <- i * 60
    SIVectorInterval <- SIColumn[borneInf:borneSup]
    mean <- mean(SIVectorInterval, na.rm=TRUE)
    SITenMinute <- c(SITenMinute, mean)}
  plot(1:length(SITenMinute), SITenMinute, type="l", col=rainbow(4)[col], main = names(SI)[indexSSI]
  col <- col+1)}
```

Psychometric database

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

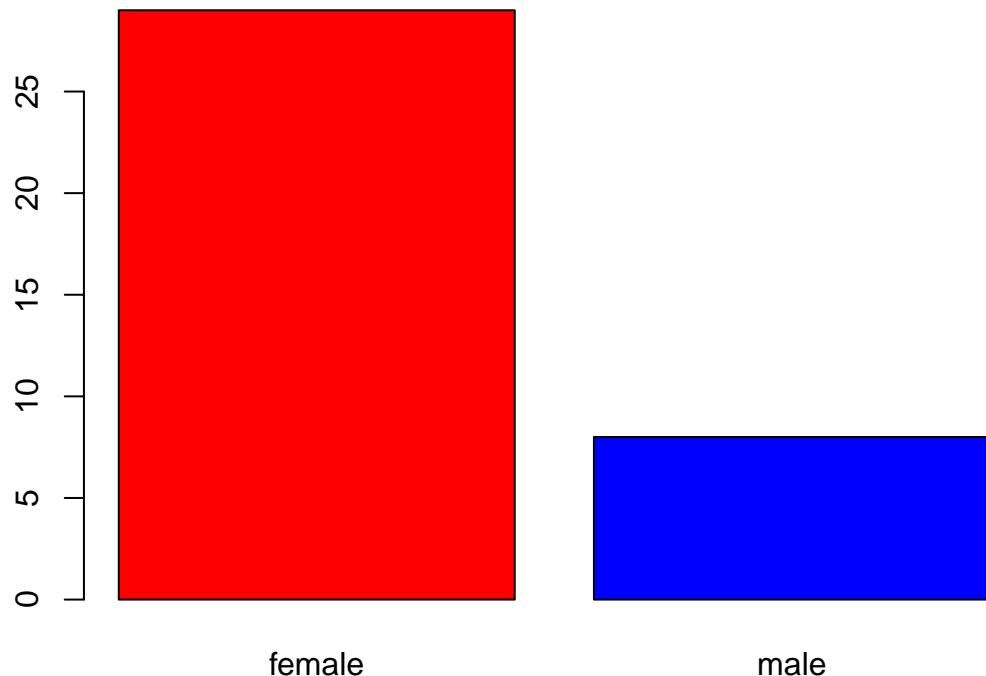
Demographic description

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")
```

Sex repartition



Age

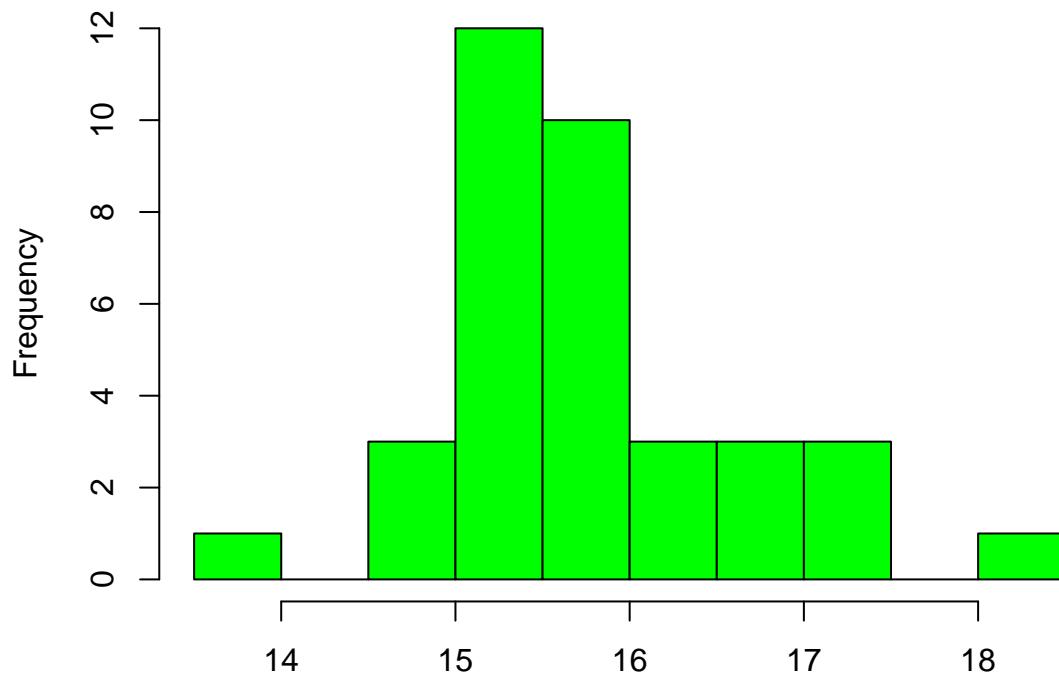
```
psycho$Birthday <- as.Date(psycho$Birthday, format="%d/%m/%y")
psycho$interview_date <- as.Date(psycho$interview_date, format="%d/%m/%y")
str(psycho$Birthday)

## 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-ps psycho$Birthday)/365.25

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

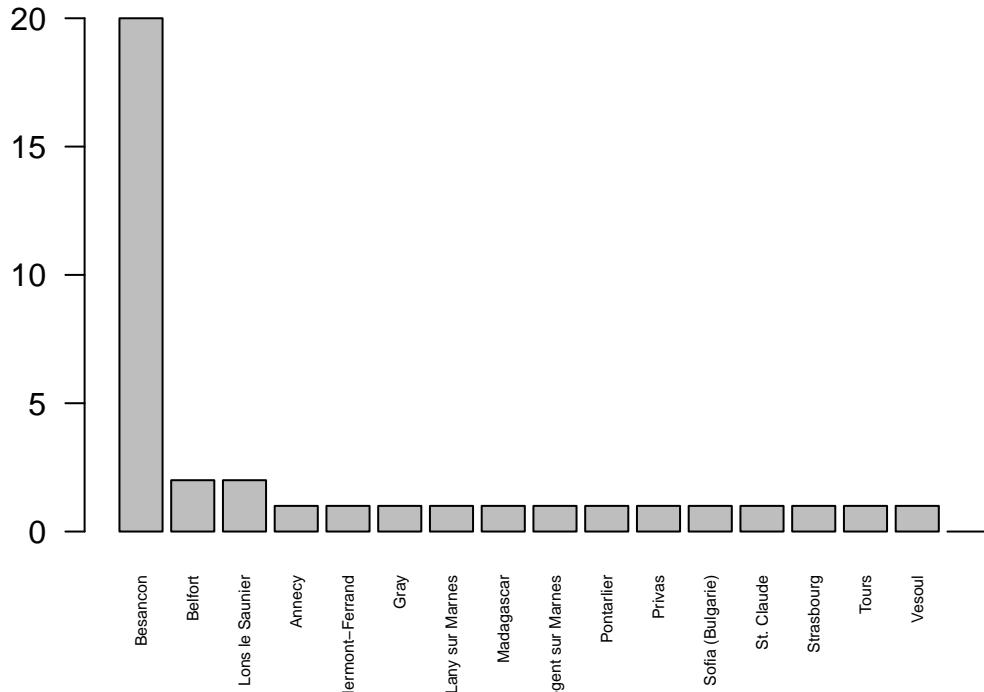
Histogram of as.numeric(psycho\$age)



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")
```

Birth place

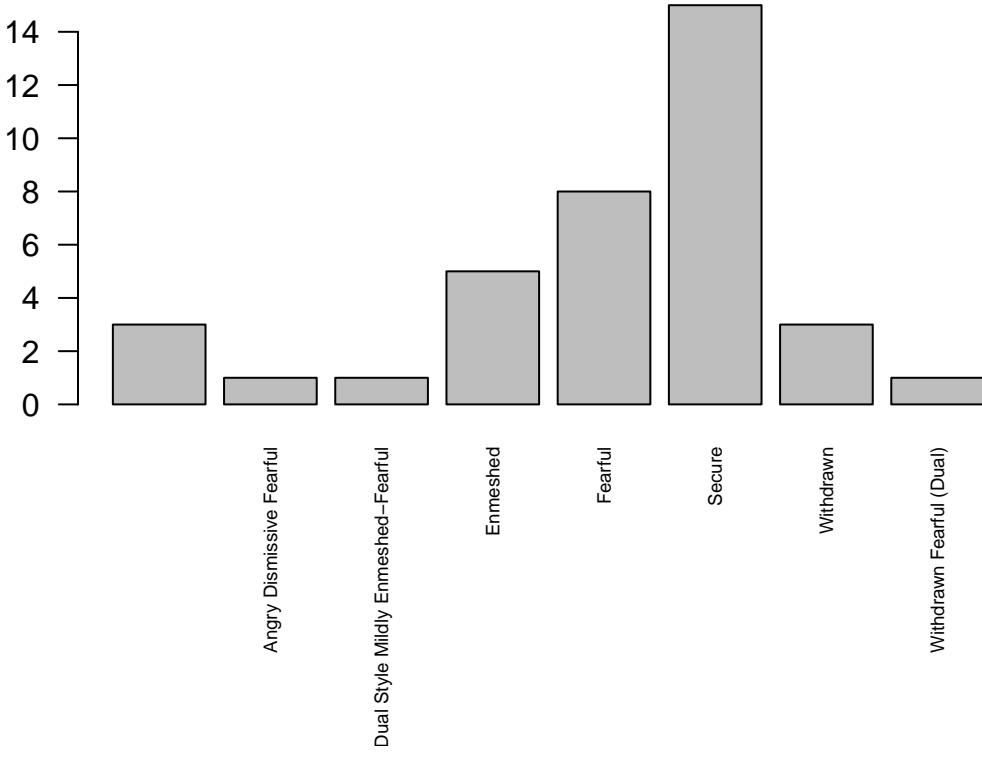


```
psycho$Birth_place
```

```
## [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         Besancon      Besancon
## 17 Levels:  Annecy Belfort Besancon Clermont-Ferrand ... Vesoul
```

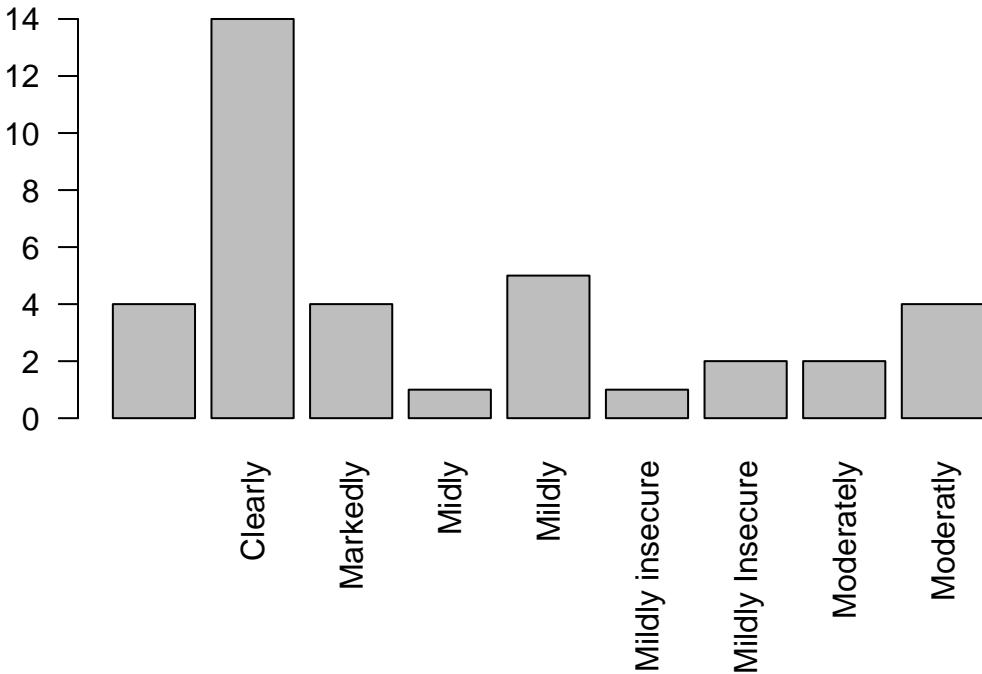
Attachement styles

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

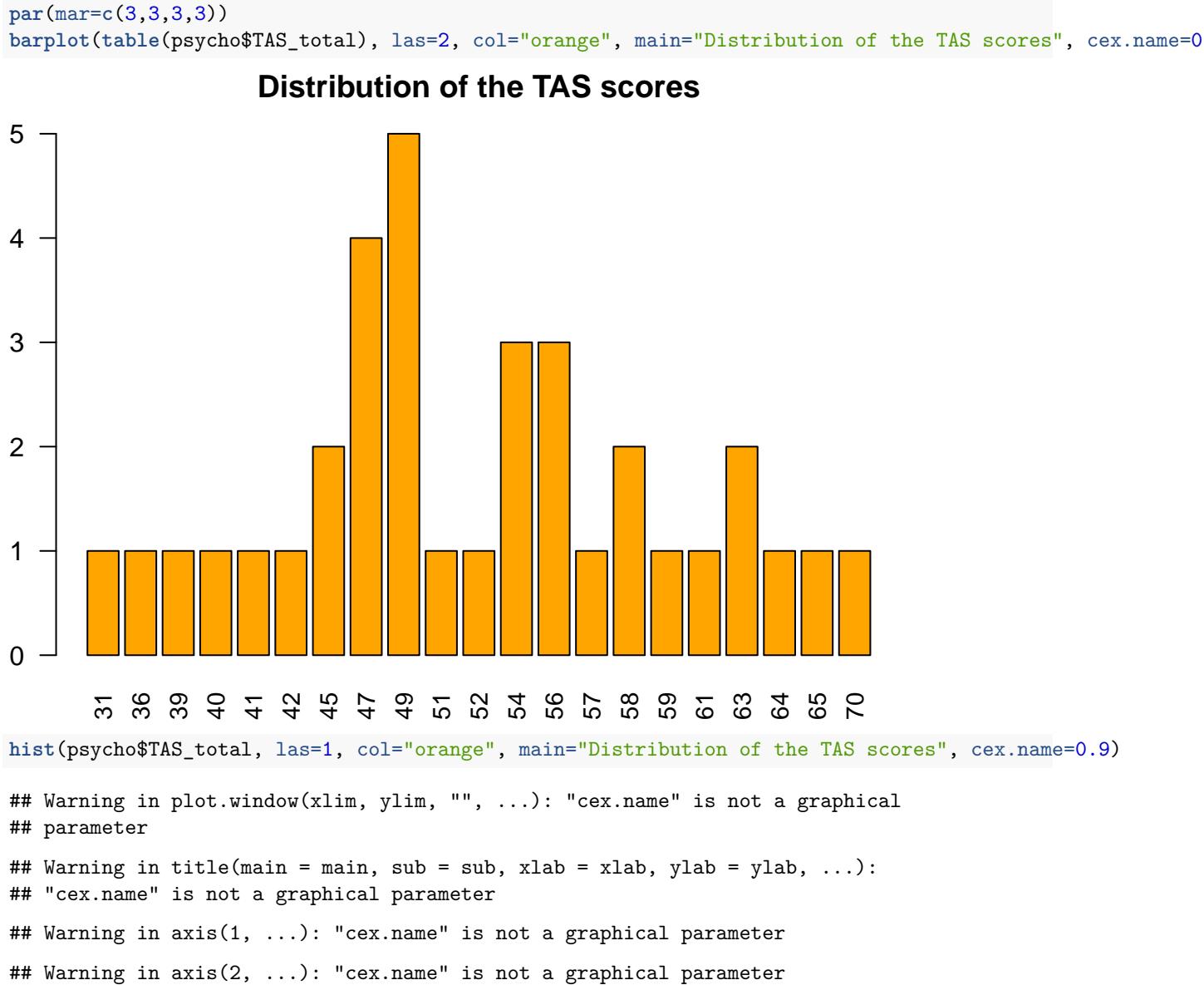


Insecurity level

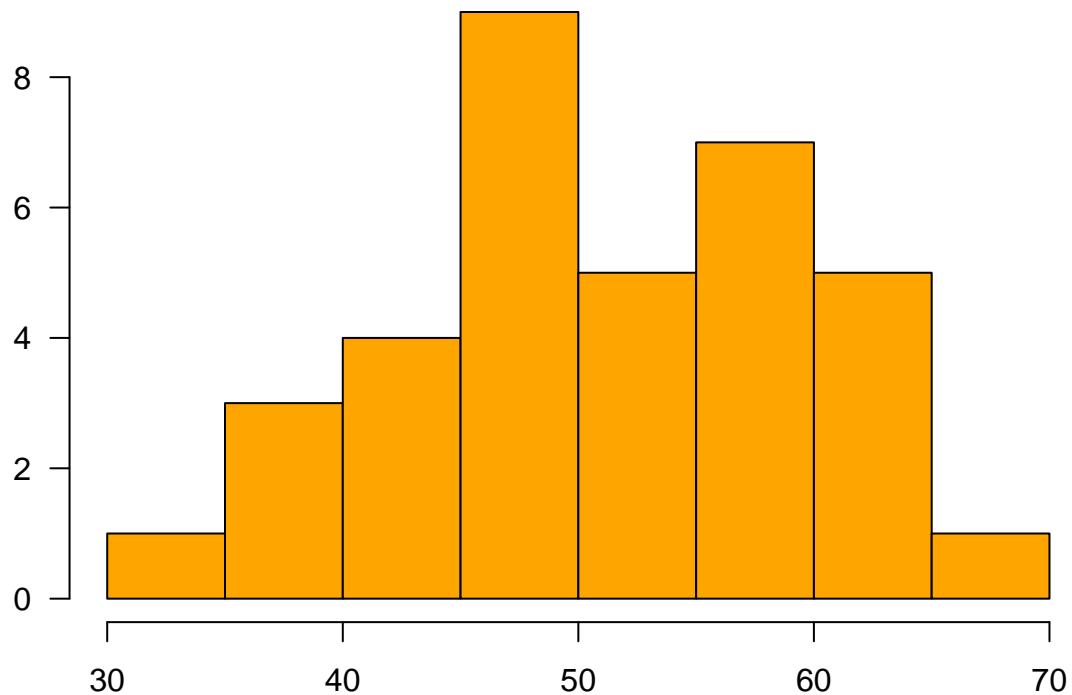
```
par(mar=c(9,5,3,3))
barplot(table(psycho$insecurite_level), las=2)
```



TAS



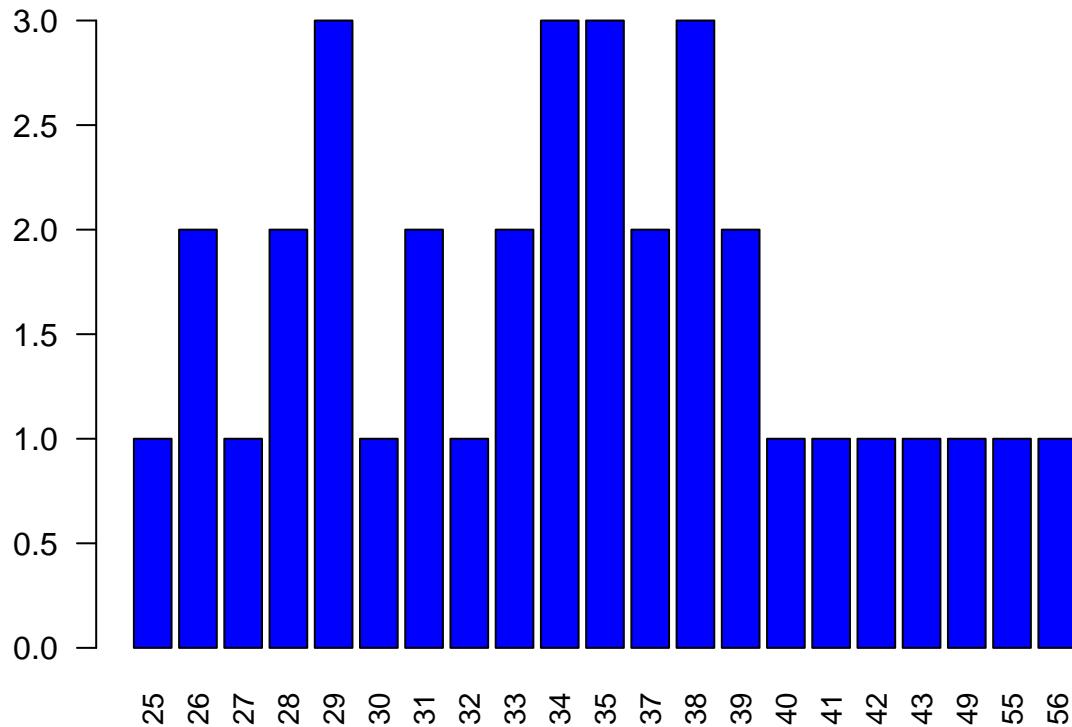
Distribution of the TAS scores



STAIYA

```
par(mar=c(3,3,3,3))
barplot(table(psycho$STAIYA_total), las=2, col="blue", main="Distribution of the STAIYA scores", cex.names=0.8)
```

Distribution of the STAIYA scores



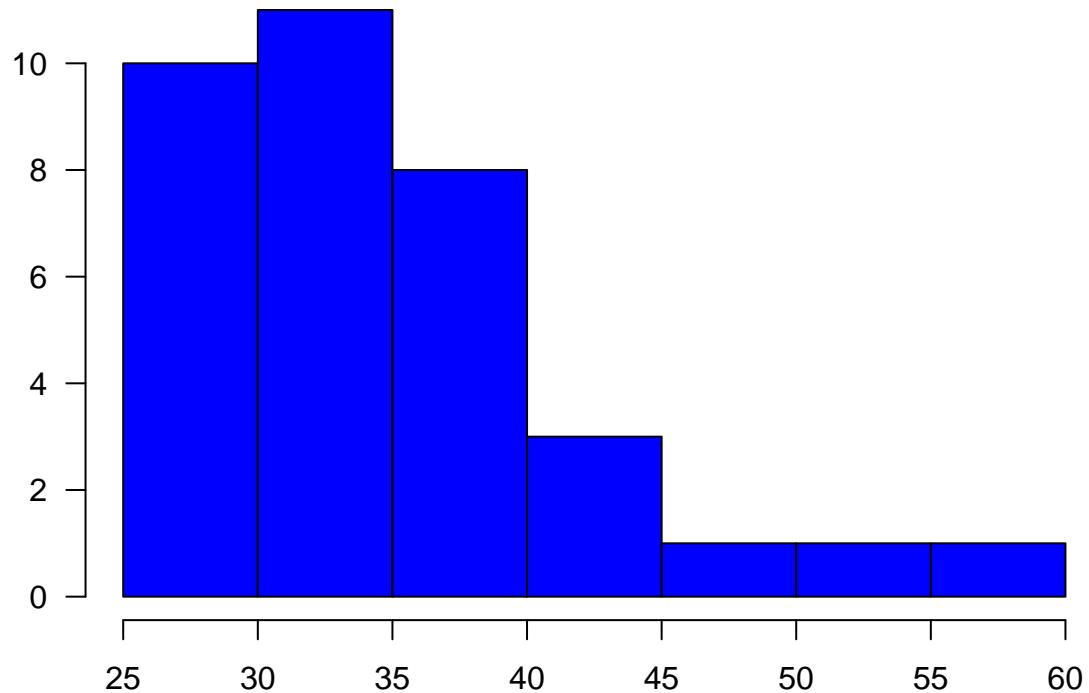
```
hist(psycho$STAIYA_total, las=1, col="blue", main="Distribution of the STAIYA scores", cex.name=0.9)

## 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
```

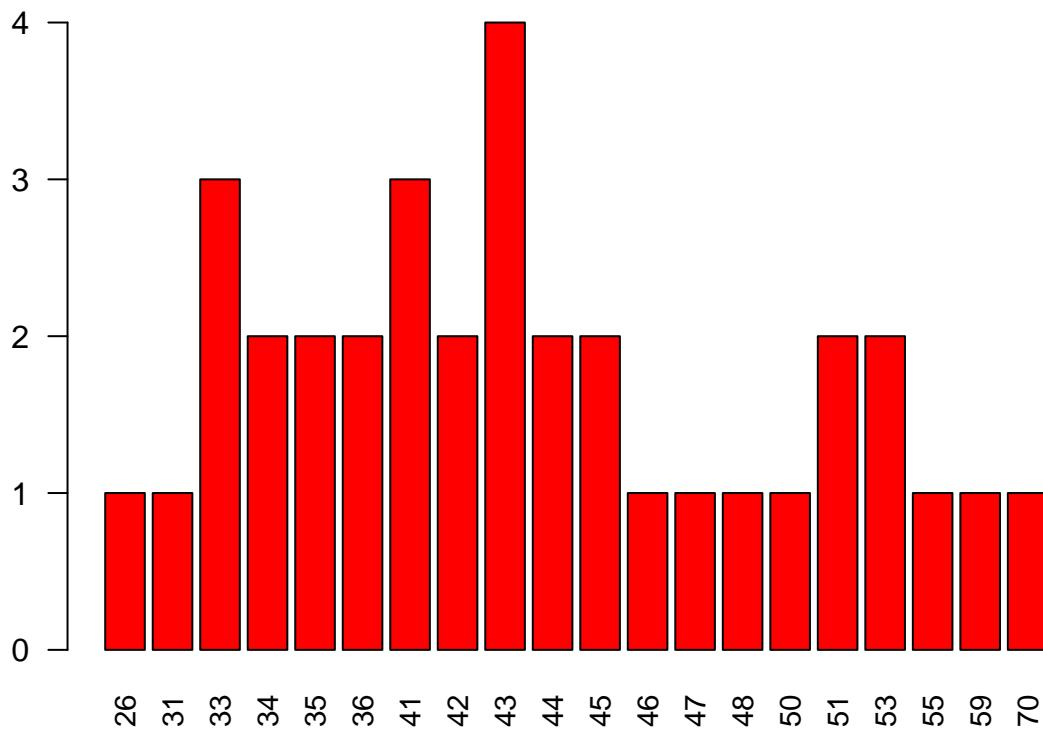
Distribution of the STAIYA scores



STAIYB

```
par(mar=c(3,3,3,3))
barplot(table(psycho$STAIYB_total), las=2, col="red", main="Distribution of the STAIYB scores", cex.names=1)
```

Distribution of the STAIYB scores



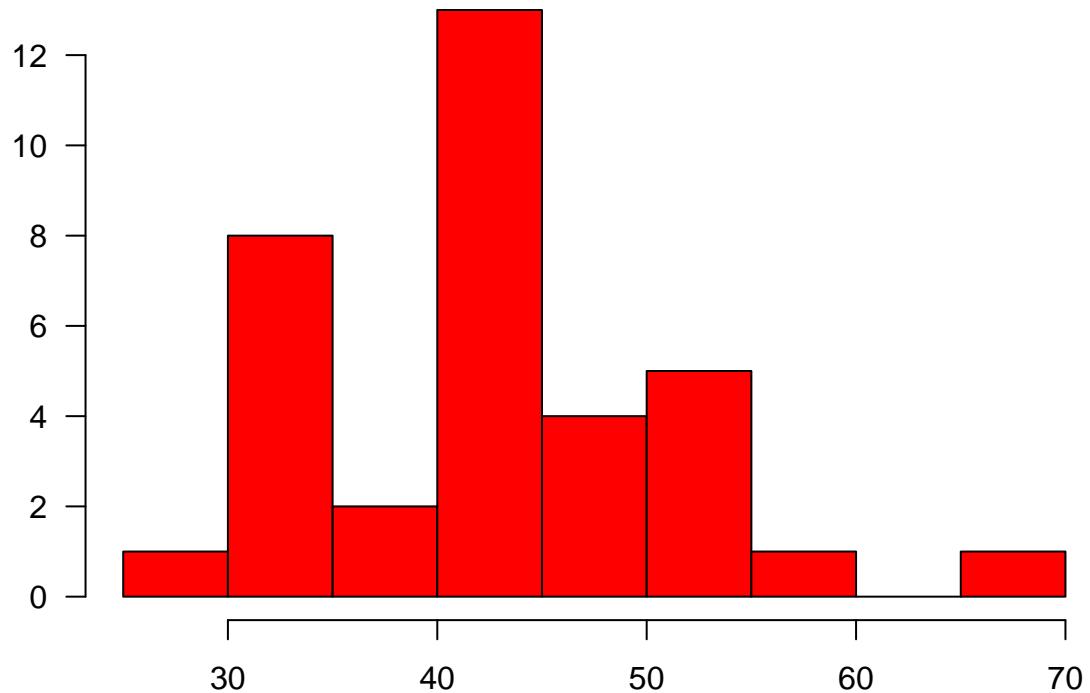
```
hist(psycho$STAIYB_total, las=1, col="red", main="Distribution of the STAIYA scores", cex.name=0.9)

## 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
```

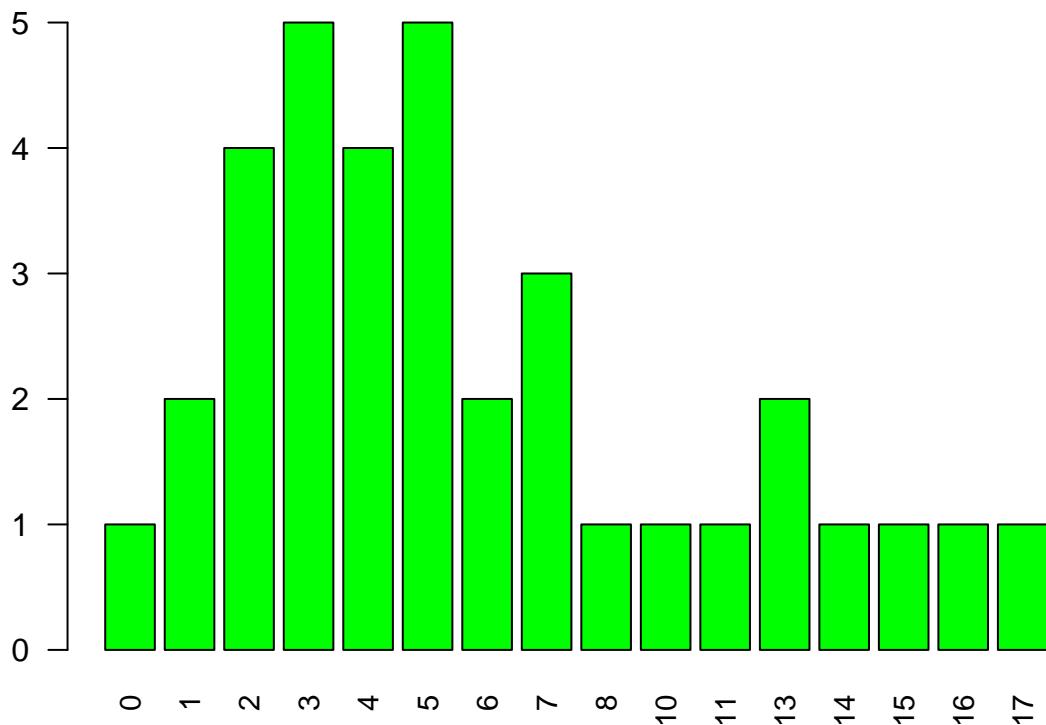
Distribution of the STAIYA scores



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=0.9)
```

Distribution of the BDI scores



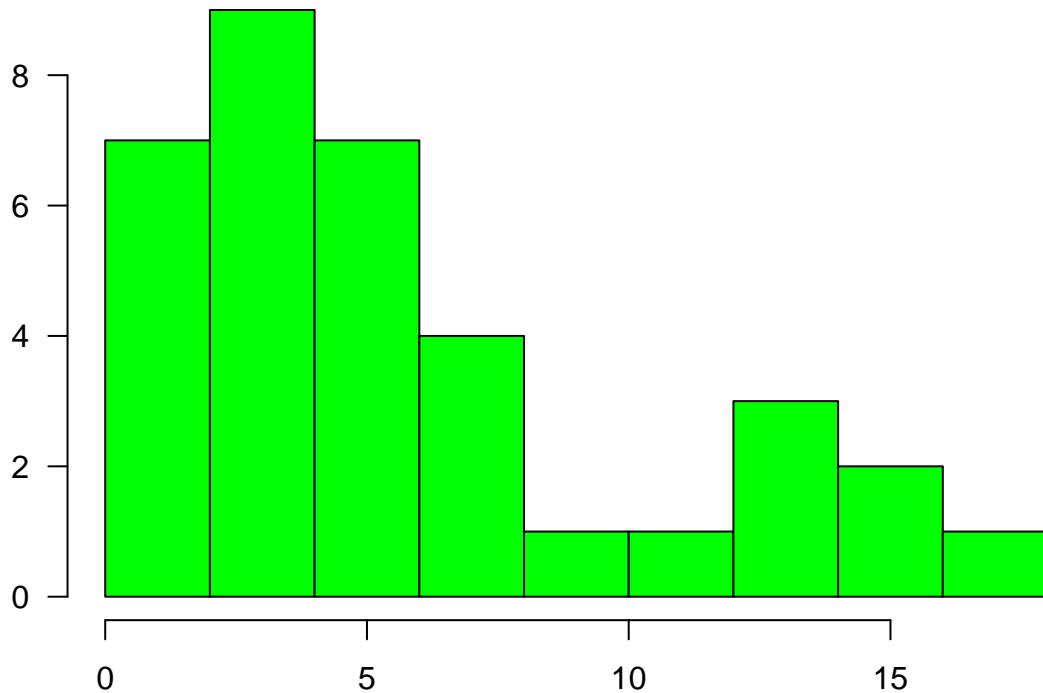
```
hist(psycho$BDI_total, las=1, col="green", main="Distribution of the BDI scores", cex.name=0.9)

## 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
```

Distribution of the BDI scores



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)
```

