onsetsynch: Package for analysis of onsets in music

Tuomas Eerola

23 Jan 2018

Contents

1 Introduction

This is a simple package meant to facilitate the analysis of onset data extracted from audio prepared in the context of *Interpersonal Entrainment Music Performance* (IEMP, an AHRC-funded research project, Principal Investigator Martin Clayton). The aim is to create facilities to replicate similar kinds of basic analyses of onset structures from different sources. This extension is available at https://github.com/tuomaseerola/onsetsynch.

1.1 Load onsetsynch library

Install and load the onsetsynch library and necessary other packages.

```
#devtools::install_github("tuomaseerola/onsetsynch") # install the developmental package synchrony
#library(onsetsynch)
library("devtools")
install("~/Dropbox/IEMP_Durham_Analysis/Analysis_tools/onsetsynch/developmental/")
library(onsetsynch)
packageVersion("onsetsynch")
```

```
## [1] '0.0.1.2000'
```

```
library(dplyr)
library(ggplot2)
library(reshape2)
```

1.2 Load and explore example onset data

We will first read in data which consists of extracted onsets (done with MIR toolbox for Matlab) and hand annotated labelling of beats and structures. This is a Cuban song, recorded by Adrian Poole at the Open University, performed by (more details here).

1.2.1 Read, filter and diagnose data

```
asere<-read.csv(url('https://raw.githubusercontent.com/tuomaseerola/onsetsynch/master/data/Asere_OU_2.cs
head(asere)
```

```
##    Piece N Label.SD Beat.pos SD.pos SD Clave_. Section Virtual.SD
## 1 Song_2 7    01:01        1      1  1       Y     Son   5.037333
## 2 Song_2 7    01:02        1      2  2       N     Son   5.260063
## 3 Song_2 7    01:03        1      3  3       N     Son   5.482792
## 4 Song_2 7    01:04        1      4  4       Y     Son   5.705521
## 5 Song_2 7    01:05        2      1  5       N     Son   5.928250
## 6 Song_2 7    01:06        2      2  6       N     Son   6.150979
##      Tactus    Tempo Clave     Bass   Guitar     Tres    Bongo Bell
## 1 0.8909167 67.34637    NA       NA       NA       NA       NA   NA
## 2 0.8909167 67.34637    NA       NA 5.281932       NA       NA   NA
## 3 0.8909167 67.34637    NA       NA 5.480643       NA 5.477695   NA
## 4 0.8909167 67.34637    NA 5.714555 5.707537 5.730943 5.718635   NA
## 5 0.8909167 67.34637    NA 5.927078 5.939071 5.917083 5.926234   NA
## 6 0.8909167 67.34637    NA       NA 6.153243 6.144901 6.155149   NA
##   Cl_Dens Bs_Dens Gt_Dens Tr_Dens Bn_Dens Bl_Dens
## 1      NA      NA      NA      NA      NA      NA
## 2      NA      NA      NA      NA      NA      NA
## 3      NA      NA      NA      NA     2.0      NA
## 4      NA      NA      NA      NA     2.0      NA
## 5      NA      NA      NA      NA     2.5      NA
## 6      NA      NA      NA      NA     2.5      NA
```

There is quite a bit of extra data here, which we won't need at this point. I also want to handle the data in `tidyverse` fashion and give few diagnostics about the data.

```
asere <- as_tibble(asere)
asere <- select(asere,SD,Section,Tempo,SD,Virtual.SD,Clave,Bass,Guitar,Tres,Bongo,Bell) # keep only key
print(dim(asere))
```

```
## [1] 1585   10
```

```
round(colSums(!is.na(asere[,which(colnames(asere)=="Clave"):
                           which(colnames(asere)=="Bell")]),na.rm = T)/nrow(asere),2) # percentage of
```

```
##  Clave   Bass Guitar   Tres  Bongo   Bell
##   0.31   0.31   0.89   0.58   0.40   0.25
```

Note: It would be convenient to have an uniform labelling of the key variables:

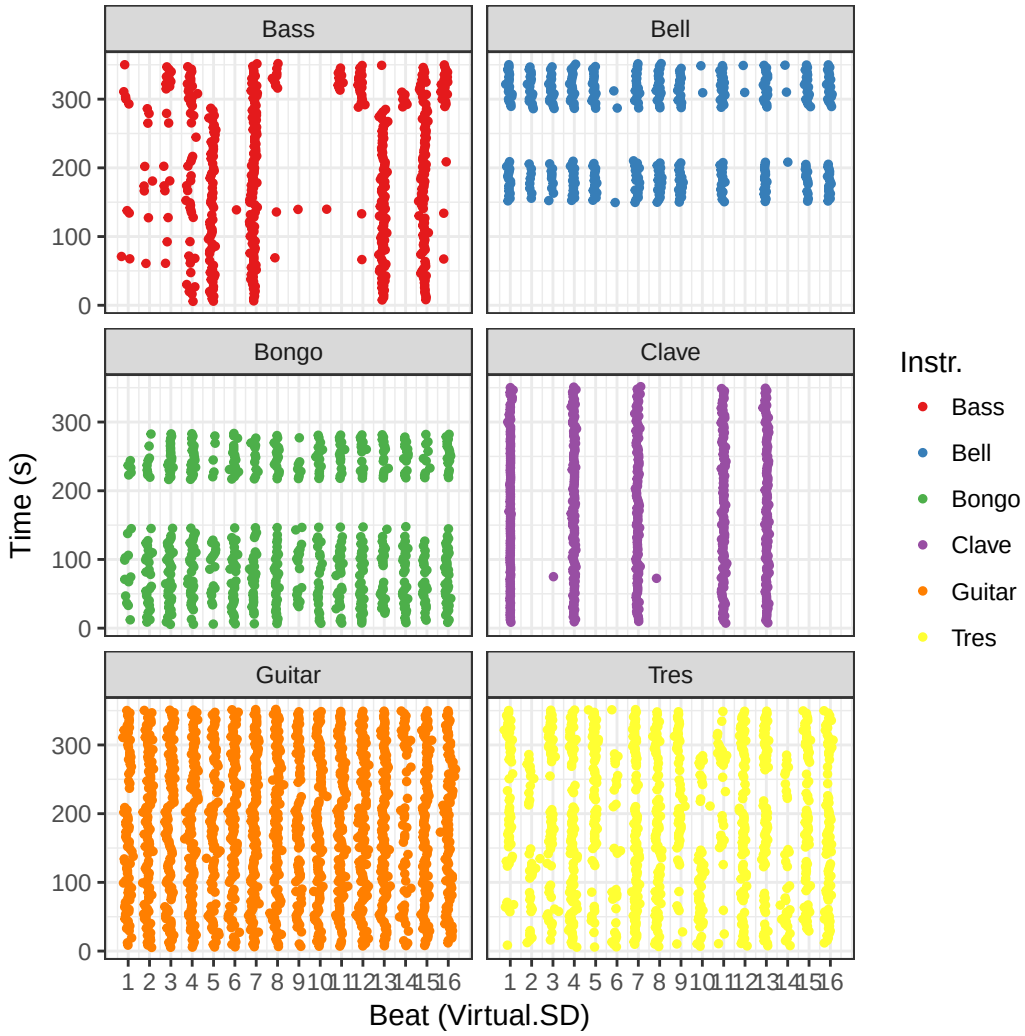
- `Corpus` = Simple string e.g. 'Cuban'
- `Piece` = String e.g. 'Song_2'
- `Section` = String e.g. 'Son'
- `Beat.pos` = Integer for ?
- `SD.pos` = ?
- `SD` = ?

- `Virtual.SD` = ?

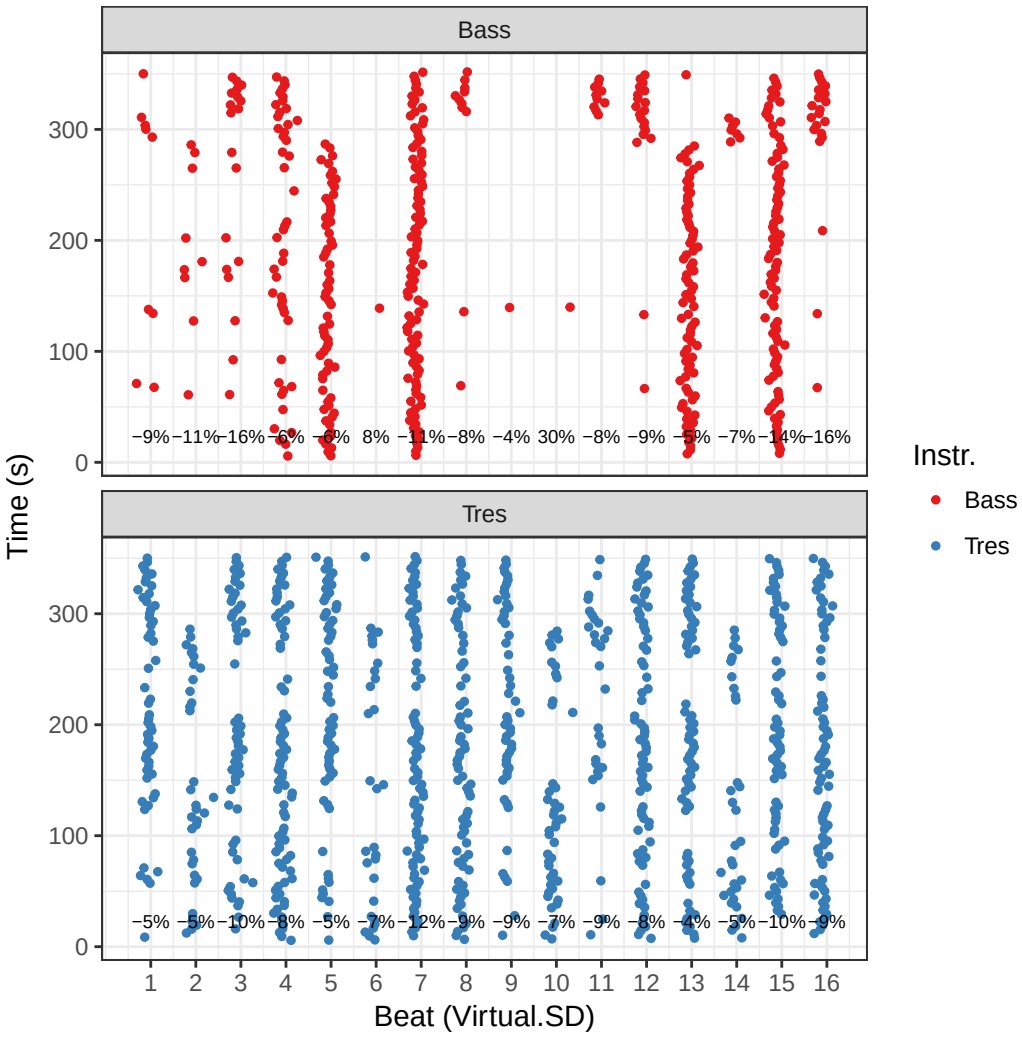
2 Visualise onset data

Use a function to display the onset relative to beats. First show all instruments, then two selected ones with relative timing deviations from the pre-defined beats.

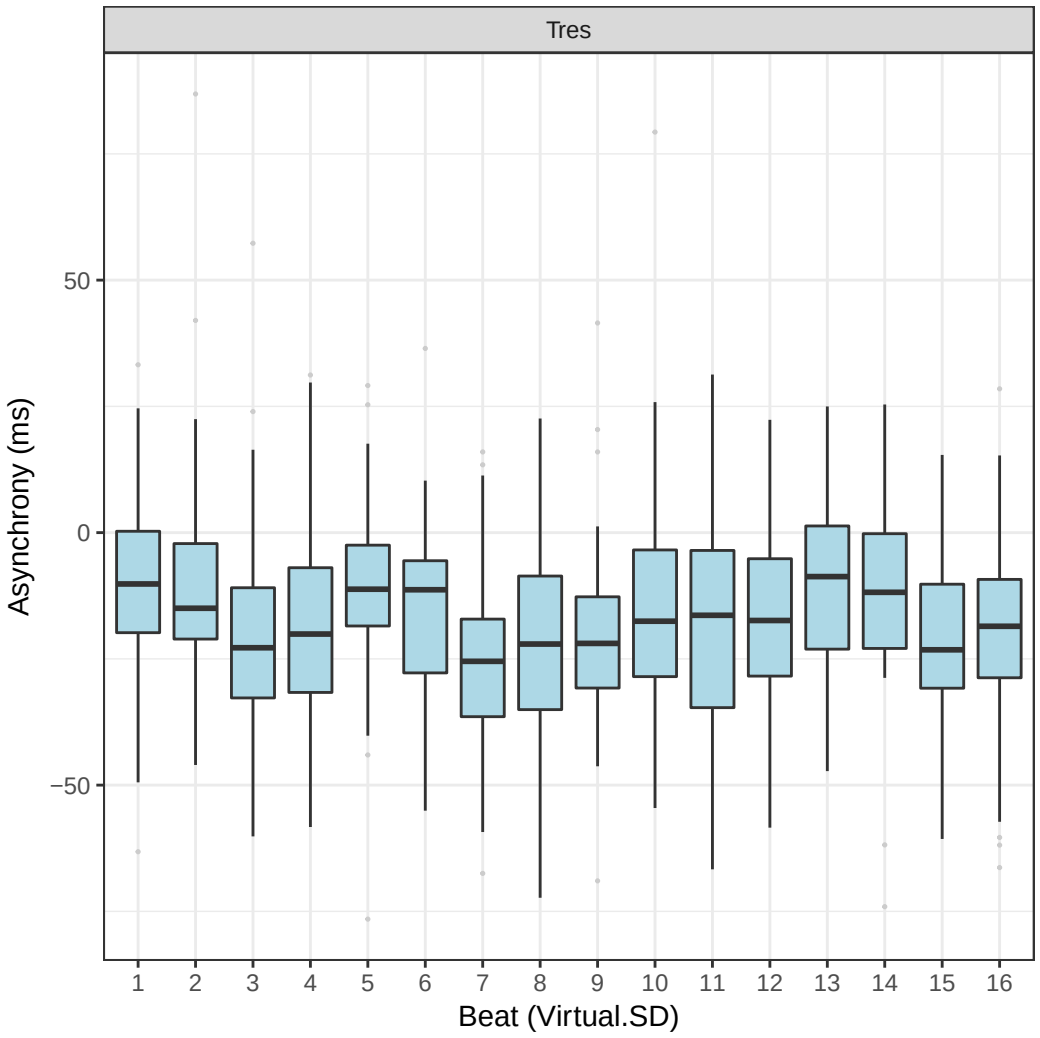
```
fig1 <- onsetsynch_by_beat_plot(asere,
   c('Bass','Clave','Guitar','Tres','Bongo','Bell'),'SD','Virtual.SD',pcols=2)
print(fig1)
```




```
fig2 <- onsetsynch_by_beat_plot(asere,c('Bass','Tres'),'SD','Virtual.SD',
                                pcols=1,griddeviations = TRUE)
print(fig2)
```

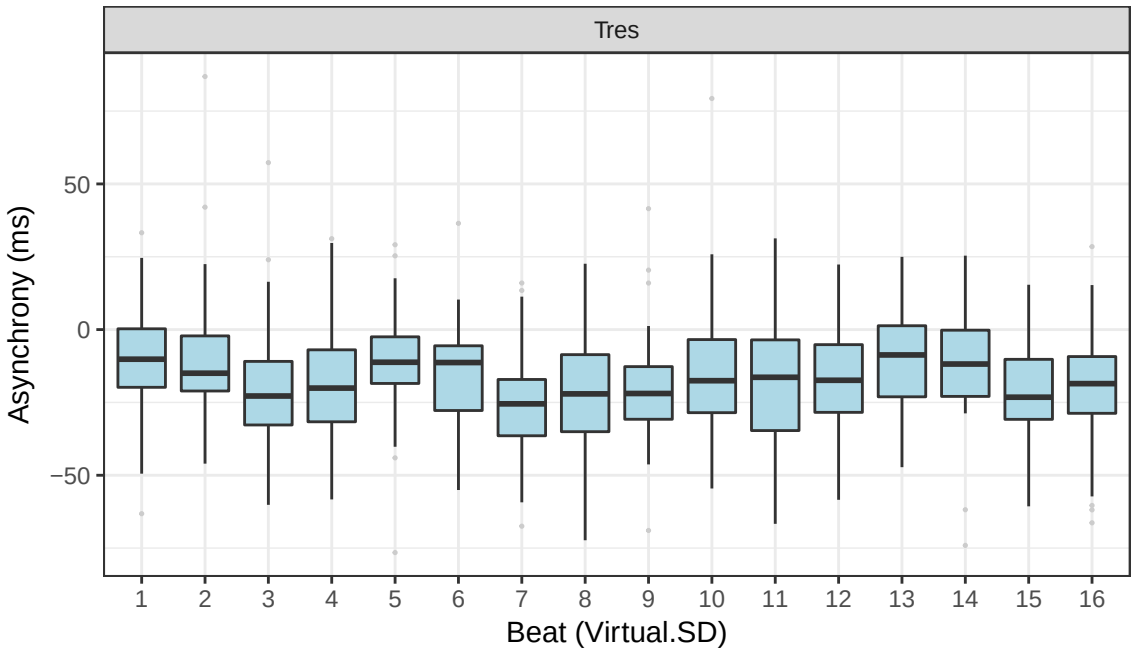



```
fig3 <- onsetsynch_by_beat_plot(asere,'Tres','SD','Virtual.SD',
                                pcols=1,griddeviations = TRUE,box=TRUE)
print(fig3)
```



The asynchronies can be displayed as a boxplot, here's an example for one instrument (Tres).

```
fig3 <- onsetsynch_by_beat_plot(asere,'Tres','SD','Virtual.SD',
                                pcols=1,griddeviations = TRUE,box=TRUE)
print(fig3)
```



3 Synchrony between specific instruments

How well are the pairs of instruments synchronised? Since the instrument play different amounts of onsets, and these are bound to be at different sub-beats, the mutual amount of onsets for each pair varies.

3.1 Randomly sample shared onsets

In order to keep the mean and standard deviations comparable, we will randomly sample the joint onsets for both instruments.

```
set.seed(1201) # set random seed before doing any sampling in order replicate results
N <- 100 # Let's select 100 onsets
d1 <- onsetsynch_sample_paired(asere,'Clave','Bass',N,1,'SD',TRUE) # Instr. pair
```

```
## [1] "onsets in common: 241"
```

```
print(paste('Asynchrony mean=',round(mean(d1$asynch*1000),4),
            'ms & sd. dev=',round(sd(d1$asynch*1000),3)))
```

```
## [1] "Asynchrony mean= 18.0883 ms & sd. dev= 21.525"
```

The first example might still be in accurate since we now know that there are at least 241 shared onset times between the Clave and the Bass. Let's redo the random sampling 10 times so we get more observations whilst still always sampling 100 joint onsets.

```
d10 <- onsetsynch_sample_paired(asere,'Clave','Bass',N,10,'SD',TRUE) # Execute 10 times
```

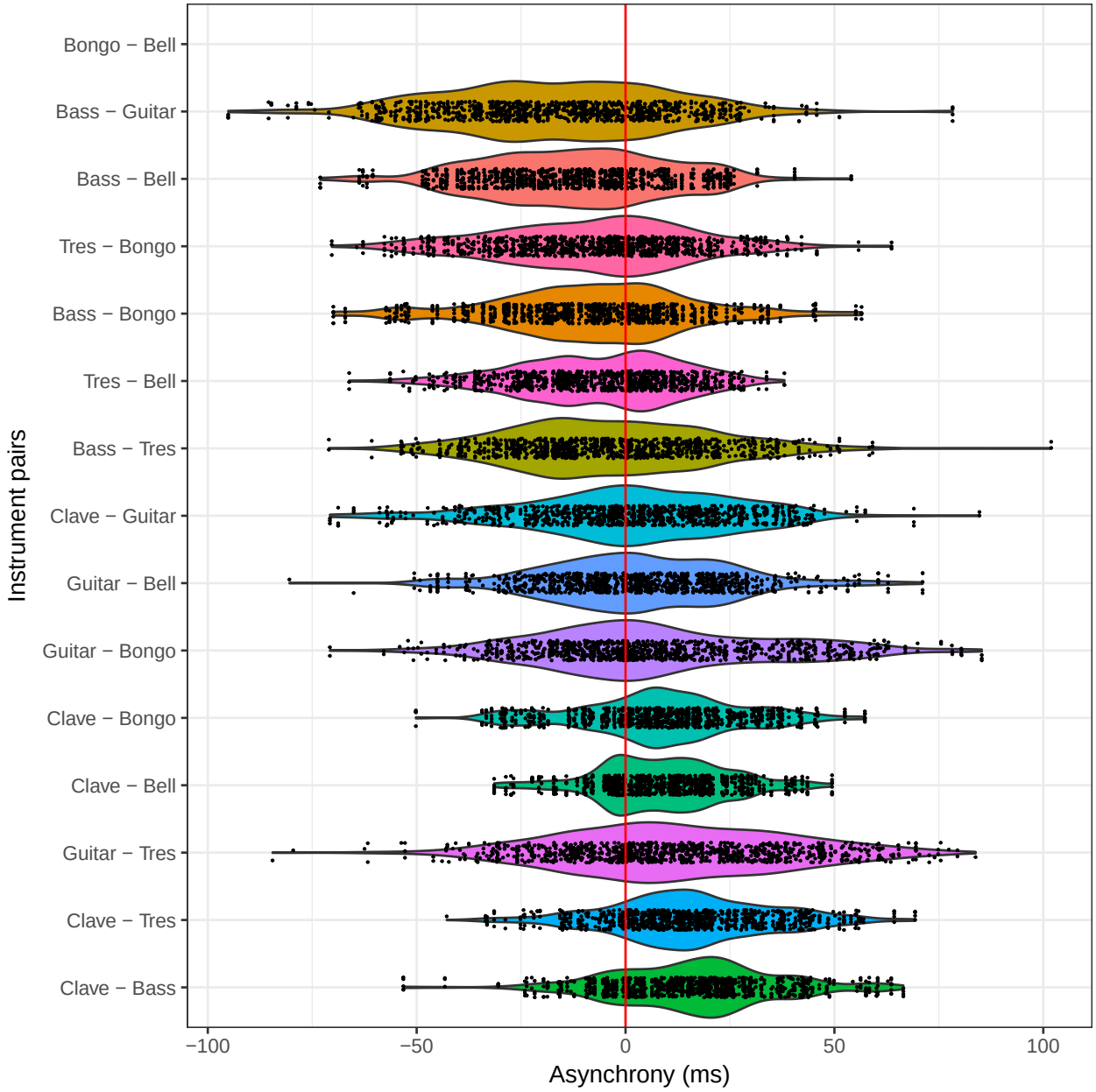
```
## [1] "onsets in common: 241"
```

```
print(paste('Asynchrony mean=',round(mean(d10$asynch*1000),4),
            'ms & sd. dev=',round(sd(d10$asynch*1000),3)))
```

```
## [1] "Asynchrony mean= 16.9891 ms & sd. dev= 19.692"
```

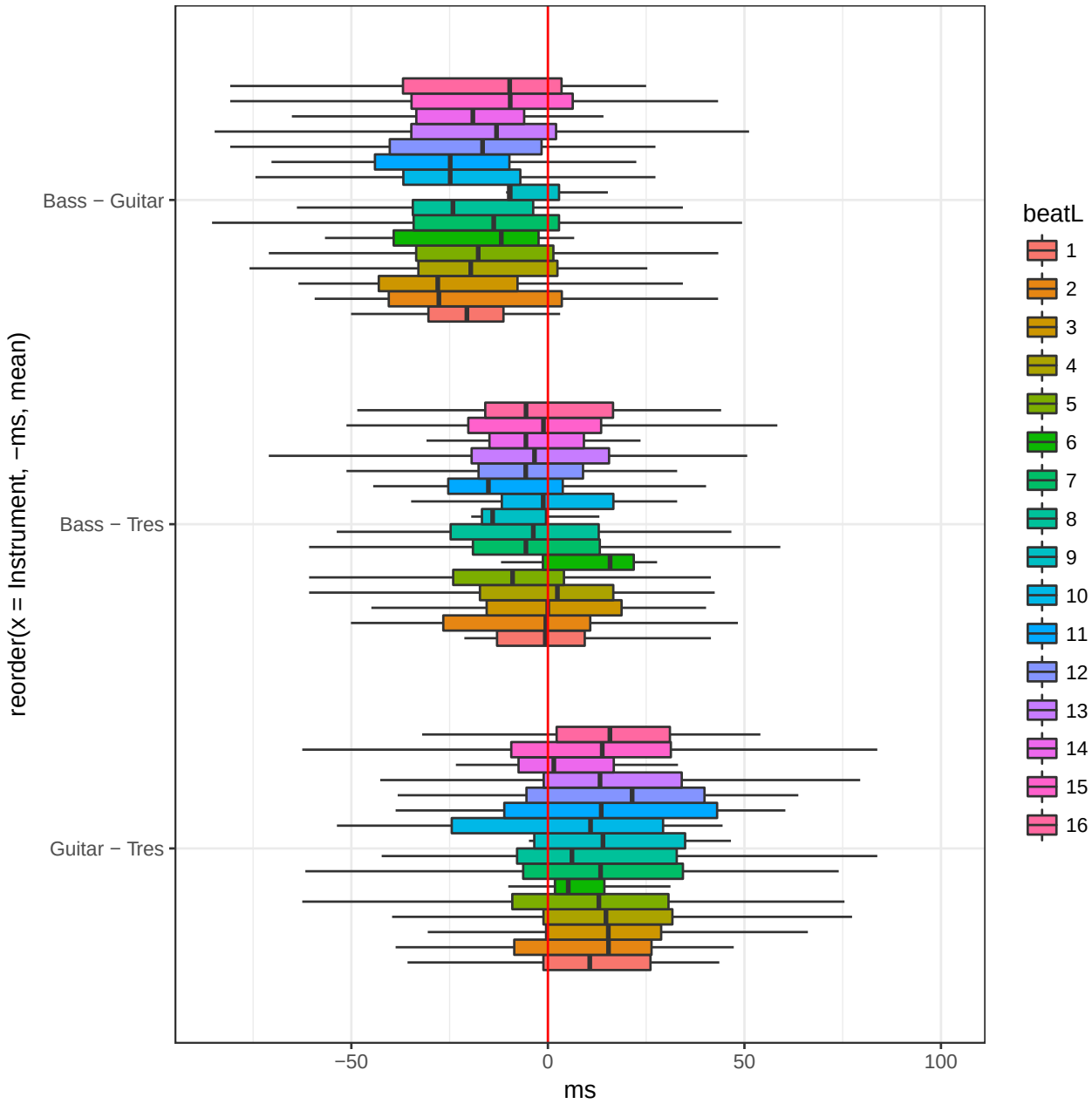
Carry this out for all possible pairings of the instruments and visualise the results.

```
inst<-c('Clave','Bass','Guitar','Tres','Bongo','Bell') # Define instruments
dn <- onsetsynch_execute_pairs(asere,inst,100,10,'SD') # Carry out pairwise comparisons
fig4 <- onsetsynch_by_pair_plot(dn)                    # plot
print(fig4)
```



Compare only three interesting instruments in terms of the synchrony across the beats.

```
source('~/Dropbox/IEMP_Durham_Analysis/Analysis_tools/onsetsynch/developmental/R/onsetsynch_execute_pai
inst<-c('Bass','Guitar','Tres') # Define instruments
dn3 <- onsetsynch_execute_pairs(asere,inst,100,10,'SD')
fig5 <- onsetsynch_by_pair_plot(dn3,bybeat=TRUE) # plot by beats
print(fig5)
```



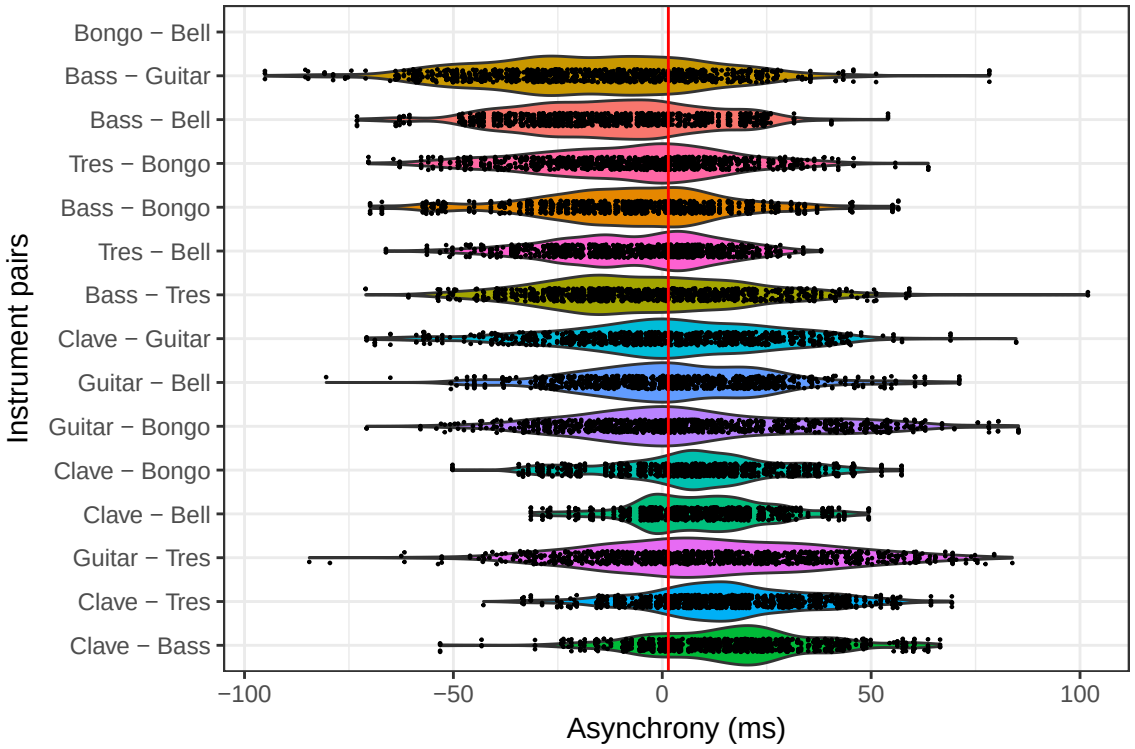
Calculate simple t-statistics (deviation from 0 synchrony) for the instrument pairs and visualise with the grand mean asynchrony of all instruments.

```
T <- data.frame(onsetsynch_by_pair_stats(dn))
print(T)
```

```
##                          M         SD          T          pval
## Clave - Bass     0.016689119 0.01954329  27.004475 4.879224e-121
## Clave - Guitar   0.001623855 0.02458017   2.089115  3.694992e-02
## Clave - Tres     0.015633945 0.01897775  26.050962 1.387498e-114
## Clave - Bongo    0.008214641 0.01923114  13.507764  2.600454e-38
## Clave - Bell     0.009285632 0.01520986  19.305734  8.017994e-71
## Bass - Guitar   -0.016364550 0.02663848 -19.426505  1.457484e-71
## Bass - Tres     -0.003722138 0.02370786  -4.964782  8.084035e-07
## Bass - Bongo    -0.007450633 0.02291069 -10.283831  1.198671e-23
```

```
## Bass - Bell    -0.011927713 0.02181613 -17.289385  8.602083e-59
## Guitar - Tres   0.012190932 0.02692082  14.320184  1.915035e-42
## Guitar - Bongo  0.006956416 0.02737060   8.037134  2.582065e-15
## Guitar - Bell   0.002992539 0.02203599   4.294446  1.922086e-05
## Tres - Bongo   -0.007476025 0.02279810 -10.369839  5.332948e-24
## Tres - Bell    -0.006301666 0.01887798 -10.556010  9.072574e-25
## Bongo - Bell             NA         NA         NA            NA
```

```
#source('~/Dropbox/IEMP_Durham_Analysis/Analysis_tools/onsetsynch/developmental/R/onsetsynch_by_pair_pl
fig6 <- onsetsynch_by_pair_plot(dn,bybeat = F,reference = mean(T$M*1000,na.rm = TRUE))
print(fig6)
```



Note that you can also get the full shared onsets with the function `onsetsynch_sample_paired` by specifying sample of 0.

```
d0 <- onsetsynch_sample_paired(asere,'Clave','Bass',N=0,beat='SD') # Determine the instrument pair
```

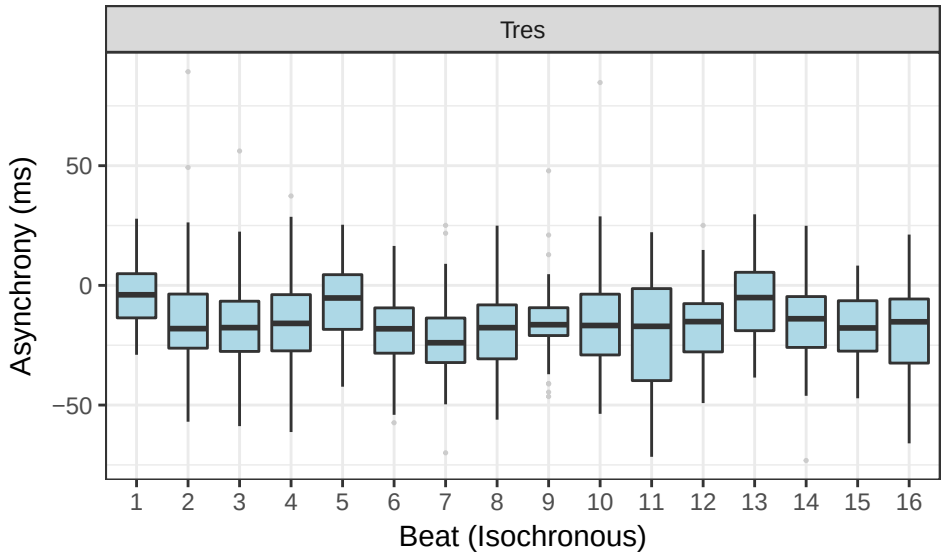
```
## [1] "take all onsets: 241"
```

3.2 Determine isochronous beat division

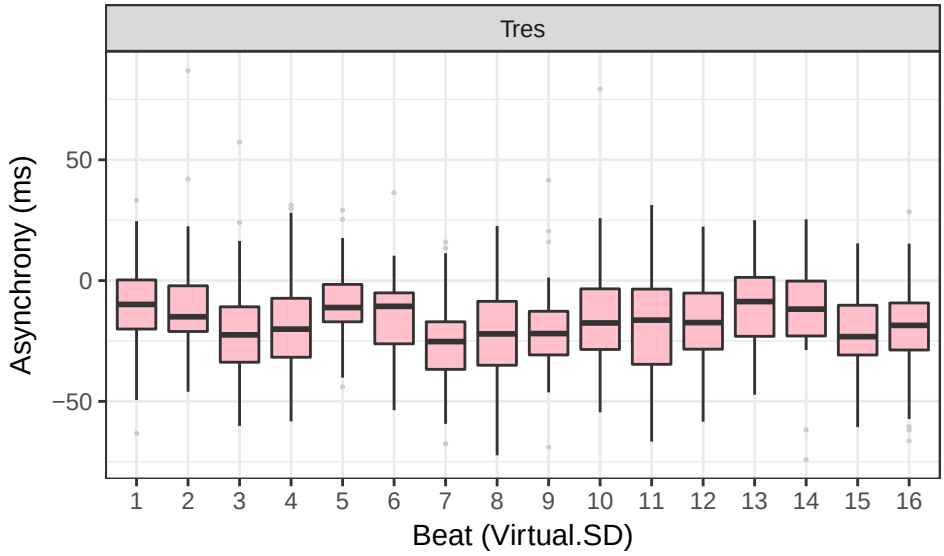
The assumption in the onset data is that it contains the raw onset times and labelling of the beats they represent. However, the Virtual beat structure is based on annotation (or tapping) and could be systematically off by a certain amount, so it would be advantageous to infer the timing information related to the beat structure from the onsets themselves. `onsetsynch_add_isobeats` is an attempt to do that. It calculates the mean onset time for the first beat of the cycle and sets an isochronous beat timings for the rest of the beats within the cycle.

```
instruments <- c('Bass','Tres','Bell','Guitar','Bongo','Bell','Clave') # use all the instruments
asere2 <- onsetsynch_add_isobeats(asere,instruments,'SD')              # add the isochronous timing
fig7 <- onsetsynch_by_beat_plot(asere2,'Tres','SD','Isochronous',box=TRUE) # plot
```

```
print(fig7)
```



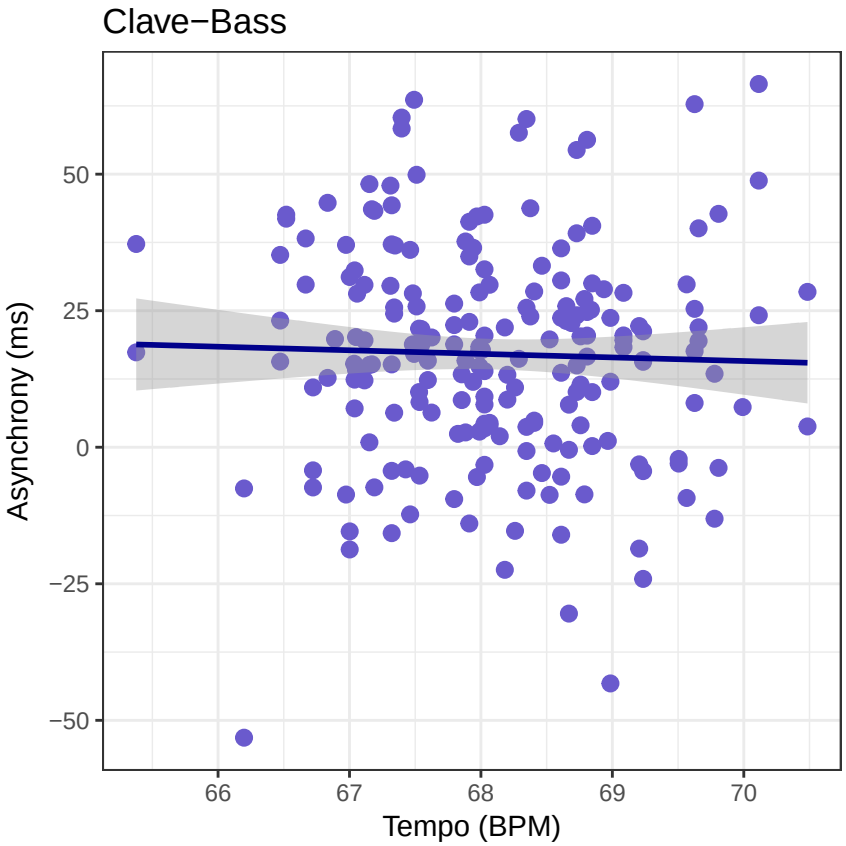

```
fig8 <- onsetsynch_by_beat_plot(asere2,'Tres','SD','Virtual.SD',box=TRUE,colour = 'pink')
print(fig8)
```



3.3 Synchrony and other variables

Visualise synchronies against another variable, for instance, tempo.

```
d1 <- onsetsynch_sample_paired(asere,'Clave','Bass',200,1,'Tempo')
fig9 <- onsetsynch_by_X_plot(d1,meta = 'Clave-Bass',xlab='Tempo (BPM)')
print(fig9)
```



4 Process multiple datasets

Take two datasets, choose the pairings, and combine the datasets and carry out the desired calculations.

4.1 Onsets in Indian and Cuban performances

In this example, the column names are slightly different to the Cuban example and the extract is taken from the middle of performance, so the timing information could be adjusted to remove the long lead time until the first onset.

```
DebBh_Drut <- read.csv('../data/DebBh_Drut.csv')
DebBh_Drut <- as_tibble(DebBh_Drut)
DebBh_Drut <- select(DebBh_Drut,Matra,Vibhag,Beat.pos,Virtual.beat,Inst.1,Tabla)
colnames(DebBh_Drut)[5]<-'Guitar'

d1 <- onsetsynch_sample_paired(asere,'Tres','Bass',N=0,beat='SD')
```

```
## [1] "take all onsets: 354"
```

```
d1<-data.frame(d1); d1$dataset<-'Asere_OU2'

d2 <- onsetsynch_sample_paired(DebBh_Drut,'Tabla','Guitar',N=0,beat='Beat.pos')
```

```
## [1] "take all onsets: 794"
```

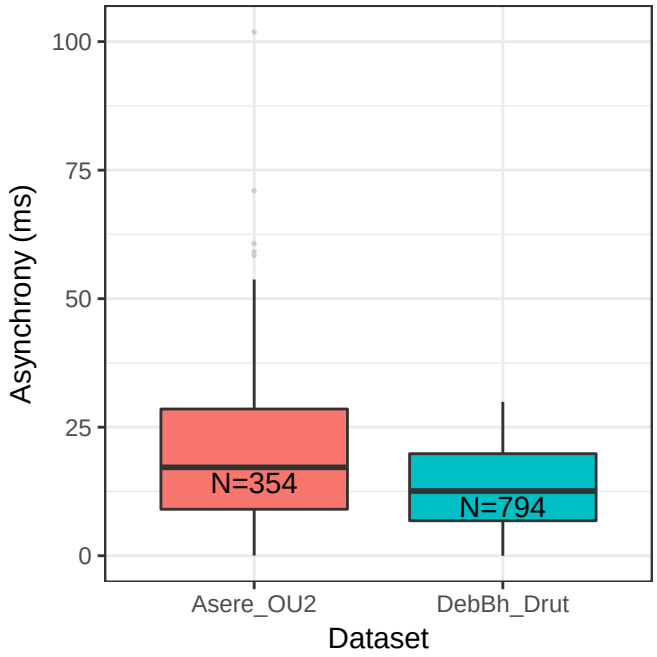
```
d2<-data.frame(d2);d2$dataset<-'DebBh_Drut'

D <- suppressMessages(full_join(d1,d2)); D$beatL<-factor(D$beatL); D$dataset<-factor(D$dataset)
D$abs_asynch_ms <- abs(D$asynch*1000)
```

D contains onset synchronies from both datasets now.

4.2 Summary statistics across datasets

The new data frame can be subjected to statistical analyses and visualisations.

```
fig10 <- onsetsynch_by_dataset_plot(D,'abs_asynch_ms','dataset',box=TRUE)
print(fig10)
```




```
library(knitr)
library(xtable)

# Summary statistics
output<-aov(abs(asynch) ~ beatL * dataset, data=D)
kable(xtable(output), caption = "Anova table")
```

Table 1: Anova table

	Df	Sum Sq	Mean Sq	F value	Pr(>F)
beatL	12	0.0170845	0.0014237	13.5943225	0.0000000
dataset	1	0.0027037	0.0027037	25.8162865	0.0000004
beatL:dataset	3	0.0001400	0.0000467	0.4454849	0.7205423
Residuals	1131	0.1184474	0.0001047	NA	NA

```
grandmeans <- summarise(group_by(D,dataset),
                        M=mean(abs(asynch*1000)),
```

Table 2: Grand mean asynchronies.

dataset	M	SD
Asere_OU2	20.02	14.55
DebBh_Drut	13.41	8.17

Table 3: Asynchronies across beats.

dataset	beatL	M	SD
Asere_OU2	1	20.45	11.64
Asere_OU2	2	17.21	15.66
Asere_OU2	3	22.39	18.42
Asere_OU2	4	19.69	14.10
Asere_OU2	5	16.72	13.25
Asere_OU2	7	20.09	13.30
Asere_OU2	8	24.37	10.19
Asere_OU2	10	101.90	NA
Asere_OU2	11	35.88	15.25
Asere_OU2	12	19.82	14.98
Asere_OU2	13	18.84	12.10
Asere_OU2	15	18.23	14.50
Asere_OU2	16	25.98	15.41
DebBh_Drut	1	12.85	8.07
DebBh_Drut	2	13.32	7.84
DebBh_Drut	3	13.79	8.17
DebBh_Drut	4	14.05	8.60

```
                        SD=sd(abs(asynch*1000)))
knitr::kable(grandmeans,caption="Grand mean asynchronies.",format="latex",digits=2)
```

```
mean_across_beats <- summarise(group_by(D,dataset,beatL),
                               M=mean(abs(asynch*1000)),
                               SD=sd(abs(asynch*1000)))
knitr::kable(mean_across_beats,format="latex",caption="Asynchronies across beats.",digits=2)
```

4.3 Batch processing (not yet implemented)

Define the desired instrument pairings for each dataset and let the metafunction to churn out the desired summaries of asynchronies (this is not yet done).

This would be easier with uniform variable names for the beat structures.

```
corpus<-NULL
corpus$fn[1] <- 'DebBh_Drut.csv';corpus$instr1[1] <- 'Guitar';
  corpus$instr2[1] <- 'Tabla';corpus$beat[1] <- 'Beat.pos';
corpus$fn[2] <- 'asere.csv';corpus$instr1[2] <- 'Bass';
  corpus$instr2[2] <- 'Tres';corpus$beat[2] <- 'SD';

# a <- onsetsynch_synchrony_across_corpora(path,corpus) # not ready yet
```

5 Analysis across time

Appy the analyses across specific segments (cycle, section, etc.).

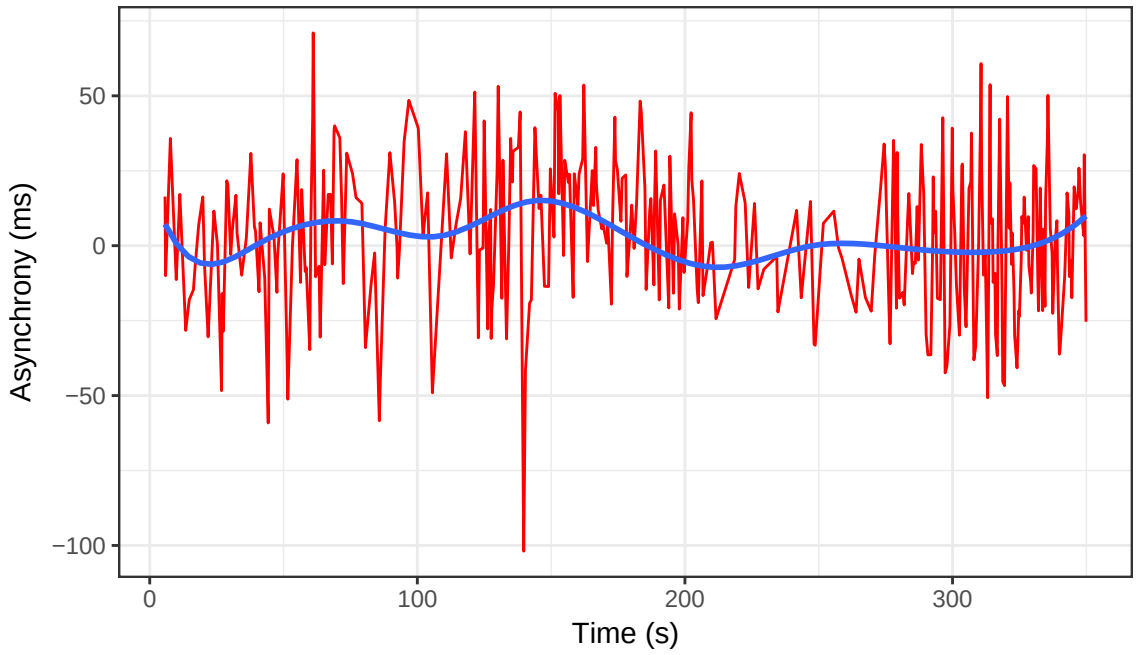
```
#source('onsetsynch_sample_paired.R')
d2 <- onsetsynch_sample_paired(asere2,'Tres','Bass',N=0,beat='SD')
```

```
## [1] "take all onsets: 352"
```

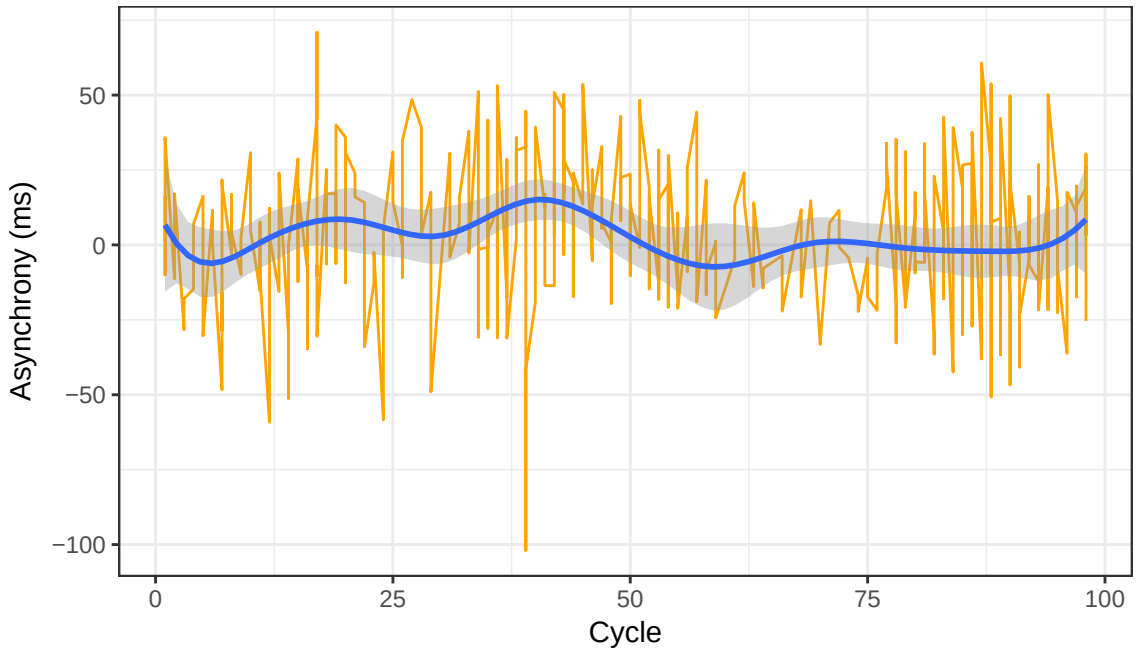
```
asere2_mutual <- d2$df2
asere2_mutual$asynch <- d2$asynch*1000
```

Visualise the temporal evolution of the asynchrony. First just raw time and asynchrony with spline interpolation, then across cycles with standard error, and finally explore visually whether the *absolute asynchronies* might be related to *tempo*.

```
fig11<-ggplot(asere2_mutual,aes(x=mean_onset,y=asynch))+
  geom_line(colour='red') +
#  stat_smooth(aes(x=mean_onset,y=asynch),
# method = lm, formula = y ~ poly(x, 12), se = FALSE)+
  stat_smooth(aes(x=mean_onset,y=asynch), method = lm,
              formula = y ~ splines::bs(x, 12), se = FALSE)+
  xlab('Time (s)')+
  ylab('Asynchrony (ms)')+
  theme_bw()
fig11
```




```
fig12 <- ggplot(asere2_mutual,aes(x=Cycle,y=asynch))+
  geom_line(colour='orange') +
  stat_smooth(aes(x=Cycle,y=asynch), method = lm,
              formula = y ~ splines::bs(x, 12), se = TRUE)+
  xlab('Cycle')+
  ylab('Asynchrony (ms)')+
  theme_bw()
fig12
```




```
fig13<-ggplot(asere2_mutual,aes(x=mean_onset,y=abs(asynch),colour=Tempo))+
  geom_line() +
  stat_smooth(aes(x=mean_onset,y=asynch), method = lm,
              formula = abs(y) ~ splines::bs(x, 12), se = FALSE)+
  xlab('Time (s)')+
  ylab('Absolute asynchrony (ms)')+
  scale_color_continuous(low='red',high = 'blue')+
  theme_bw()+
  theme(legend.position="top")
fig13
```