Time series matching for fitness buddies

In this notebook, I will import relevant data from different sources from the Fitbit fitness tracker data publically available from Kaggle. I will then clean the data to prepare plots to motivate the project, and perform some preliminary analysis.

```
library(tidyverse)
library(lubridate)
library(hms)

# Read the number of steps per minute data
minute_steps <- read_csv("minuteStepsNarrow_merged.csv")
head(minute_steps)
```

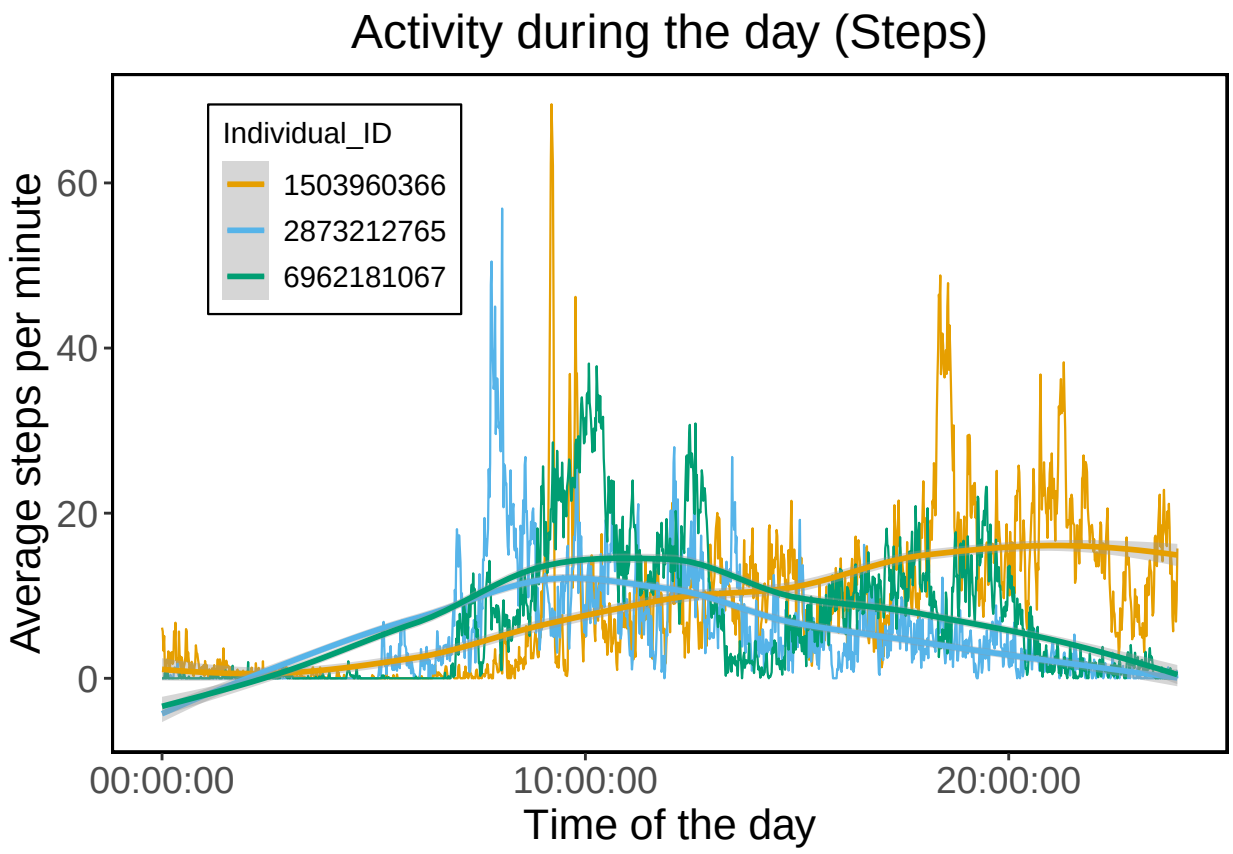
```
## # A tibble: 6 x 3
##           Id ActivityMinute        Steps
##        <dbl> <chr>                 <dbl>
## 1 1503960366 4/12/2016 12:00:00 AM     0
## 2 1503960366 4/12/2016 12:01:00 AM     0
## 3 1503960366 4/12/2016 12:02:00 AM     0
## 4 1503960366 4/12/2016 12:03:00 AM     0
## 5 1503960366 4/12/2016 12:04:00 AM     0
## 6 1503960366 4/12/2016 12:05:00 AM     0
```

```
# Convert the ActivityMinute column to date object and extract time in a separate column.
minute_steps$date_time <- as.POSIXct(minute_steps$ActivityMinute, format="%m/%d/%Y %I:%M:%S %p", tz=Sys
minute_steps$time <- as_hms(minute_steps$date_time)
minute_steps$Individual_ID <- factor(minute_steps$Id)

# Combine the average steps per minute for all different dates for each indivudual.
average_steps <- minute_steps %>% group_by(Individual_ID, time) %>% summarise(mean_steps = mean(Steps))
```

As a person interested to get fit who never managed to properly motivate myself to workout, I would've appreciated having a buddy to motivate me to workout. While some of my friends were happy to workout with me, it was never sustainable as we always had scheduling issues because we liked to workout at different times. So I aim to match people based on their workout schedules. In the plot below, I show average daily number of steps by the minute of 3 random individuals from the dataset. We could see that individuals in green and and blue are early birds that could work together, while individual in orange prefers to be active in the evenings.

```
# Plot the time series of Average steps per minute for 3
set.seed(42)
ggplot(subset(average_steps,Individual_ID %in% sample(unique(average_steps$Individual_ID),size = 3)), ae
  xlab("Time of the day") + ylab("Average steps per minute") + ggtitle("Activity during the day (Steps)
  scale_color_manual(values=c("#E69F00", "#56B4E9", "#009E73")) +
  geom_line(aes(color = Individual_ID), size = 0.4) + stat_smooth(aes(color = Individual_ID), method =
  theme_bw() +
  theme(panel.grid.major = element_blank(),
        panel.grid.minor = element_blank(),
        panel.border = element_rect(colour = "black", size=1),
```

```
        legend.direction = 'vertical',
        legend.position = c(.2,.8),
        legend.background = element_rect(colour = 'black', size = 0.4),
        legend.title = element_text(size=11),
        legend.text=element_text(size=11),
        plot.title = element_text(size=18, hjust = .5),
        axis.text.x = element_text(size=14),
        axis.text.y = element_text(size=14),
        axis.title.x = element_text(size=16),
        axis.title.y = element_text(size=16))
```

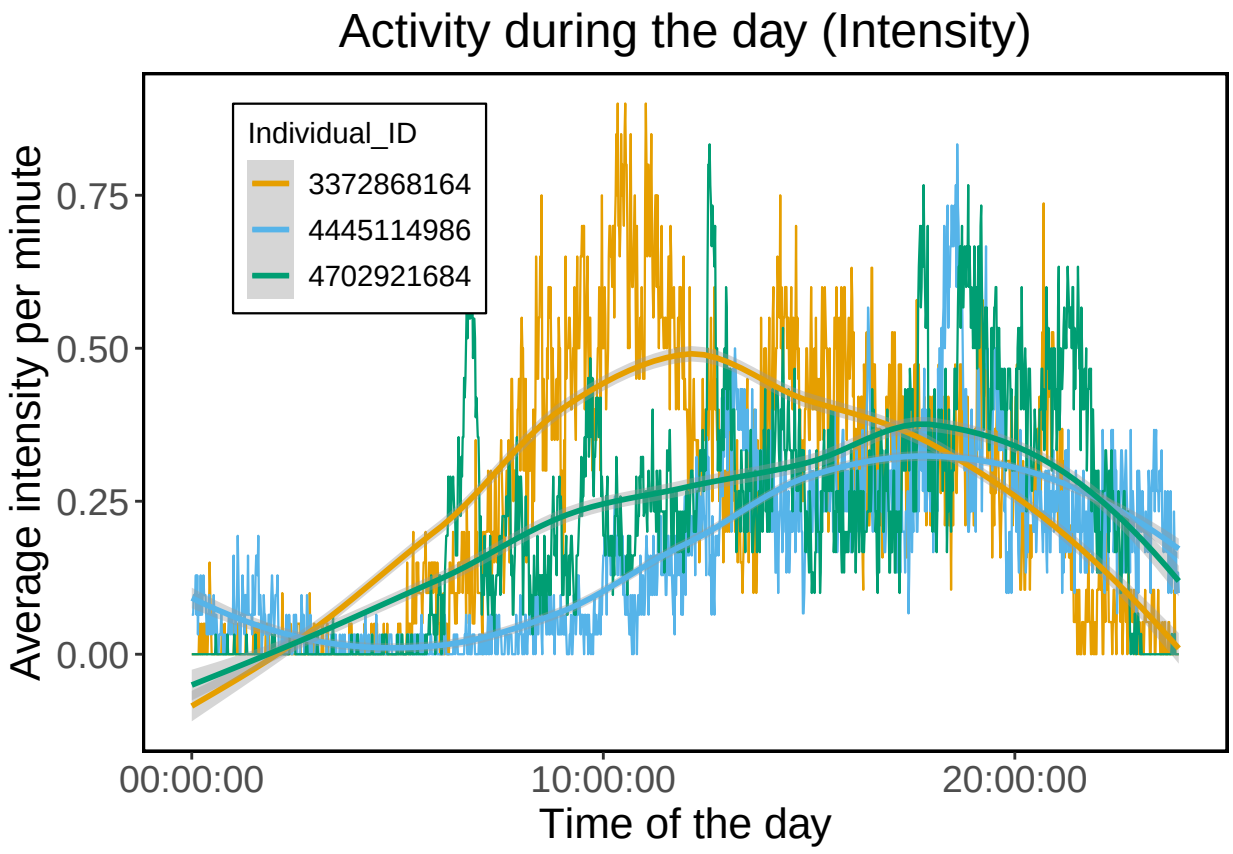


We could make a similar plot for the intensity of exercise. Here, we could see two individuals who prefer to be active in the evenings (green and blue), whereas the individual in orange seems to prefer to be more active in the afternoons.

```
# Data cleaning as above
minute_intensity <- read_csv("minuteIntensitiesNarrow_merged.csv")
head(minute_intensity)
```

```
## # A tibble: 6 x 3
##           Id ActivityMinute         Intensity
##        <dbl> <chr>                      <dbl>
## 1 1503960366 4/12/2016 12:00:00 AM         0
## 2 1503960366 4/12/2016 12:01:00 AM         0
## 3 1503960366 4/12/2016 12:02:00 AM         0
## 4 1503960366 4/12/2016 12:03:00 AM         0
## 5 1503960366 4/12/2016 12:04:00 AM         0
## 6 1503960366 4/12/2016 12:05:00 AM         0
```

```
minute_intensity$date_time <- as.POSIXct(minute_intensity$ActivityMinute, format="%m/%d/%Y %I:%M:%S %p"
minute_intensity$time <- as_hms(minute_intensity$date_time)
minute_intensity$Individual_ID <- factor(minute_intensity$Id)

average_intensity <- minute_intensity %>% group_by(Individual_ID, time) %>% summarise(mean_intensity = m

set.seed(44)
ggplot(subset(average_intensity,Individual_ID %in% sample(unique(average_intensity$Individual_ID),size =
  xlab("Time of the day") + ylab("Average intensity per minute") + ggtitle("Activity during the day (In
  scale_color_manual(values=c("#E69F00", "#56B4E9", "#009E73")) +
  geom_line(aes(color = Individual_ID), size = 0.4) + stat_smooth(aes(color = Individual_ID), method =
  theme_bw() +
  theme(panel.grid.major = element_blank(),
        panel.grid.minor = element_blank(),
        panel.border = element_rect(colour = "black", size=1),
        legend.direction = 'vertical',
        legend.position = c(.2,.8),
        legend.background = element_rect(colour = 'black', size = 0.4),
        legend.title = element_text(size=11),
        legend.text=element_text(size=11),
        plot.title = element_text(size=18, hjust = .5),
        axis.text.x = element_text(size=14),
        axis.text.y = element_text(size=14),
        axis.title.x = element_text(size=16),
        axis.title.y = element_text(size=16))
```



Next we perform time series clustering using the Dynamic Time Warping algorithm. We use two methods.

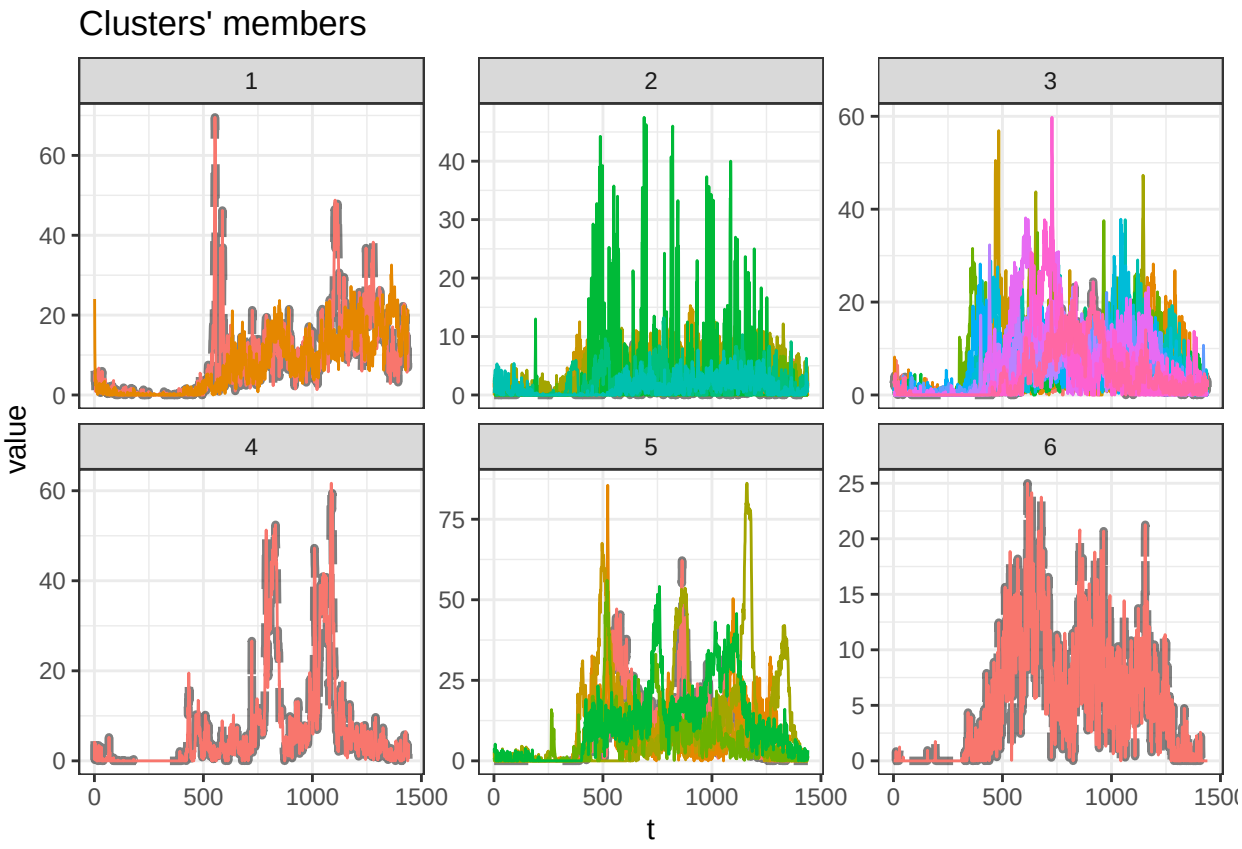
First, k-medoids clustering using Partition around medoids (PAM), and second, hierarchical clustering.

```
## Time series clustering using the Dynamic Time Warping algorithm
library(dtwclust)

# Prepare data by converting in wide form to be used in tsclust() function

average_steps_wide <-
  average_steps %>%
  spread(time, mean_steps)

# Convert the ID column to rownames

average_steps_wide <-
  average_steps_wide %>%
  remove_rownames %>%
  column_to_rownames("Individual_ID")

# Perform k-medoids clustering using Partition around medoids (PAM). Using 6 clusters for this example.
Time_series_clusters <- tsclust(average_steps_wide, type="partitional", k=6L, distance="dtw", clusterin
```

In this example, we made 6 clusters. Below we plot the time series of different individuals in these 6 clusters.

```
# Plot the time series results of the clusters.
plot(Time_series_clusters, type = "sc")
```

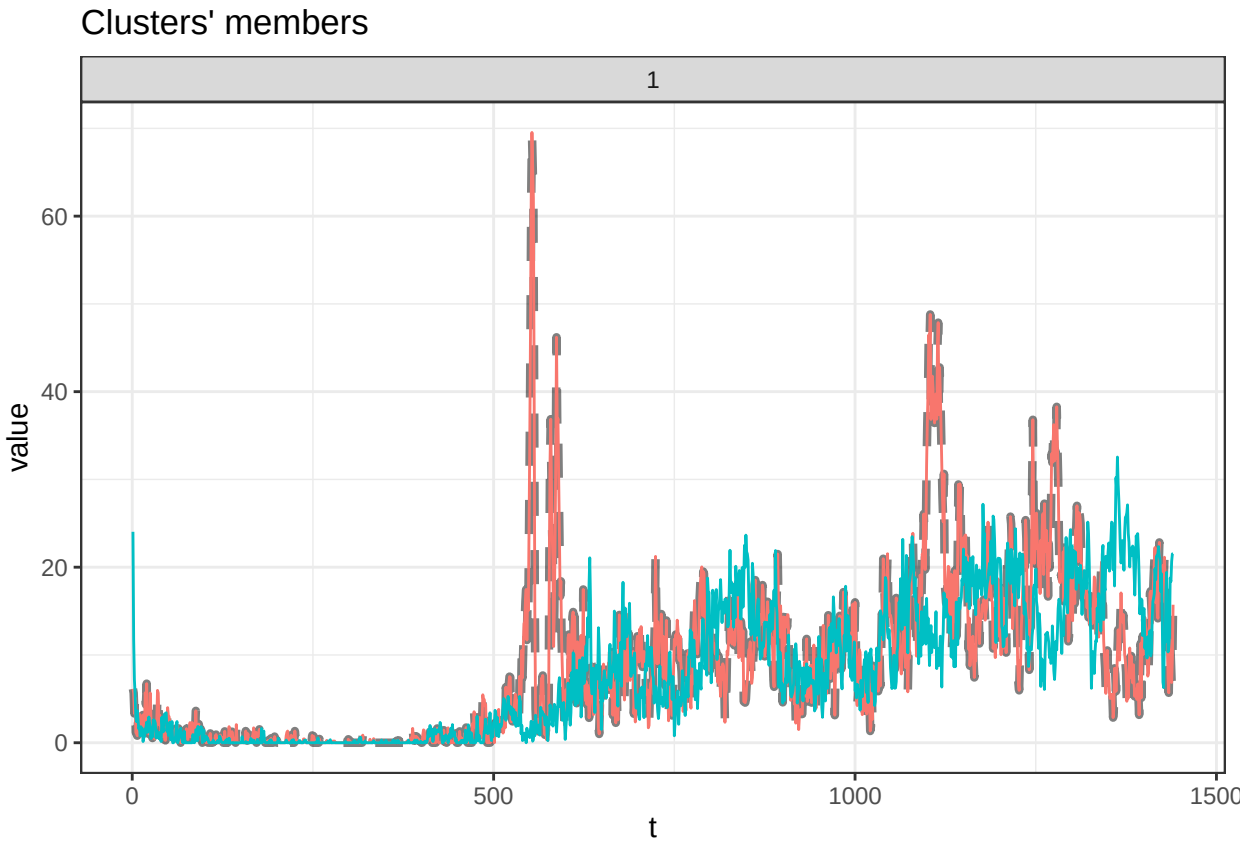
```
## Warning: `guides(<scale> = FALSE)` is deprecated. Please use `guides(<scale> =
## "none")` instead.
```



We can also inspect one cluster at a time. For example, below is cluster 1.

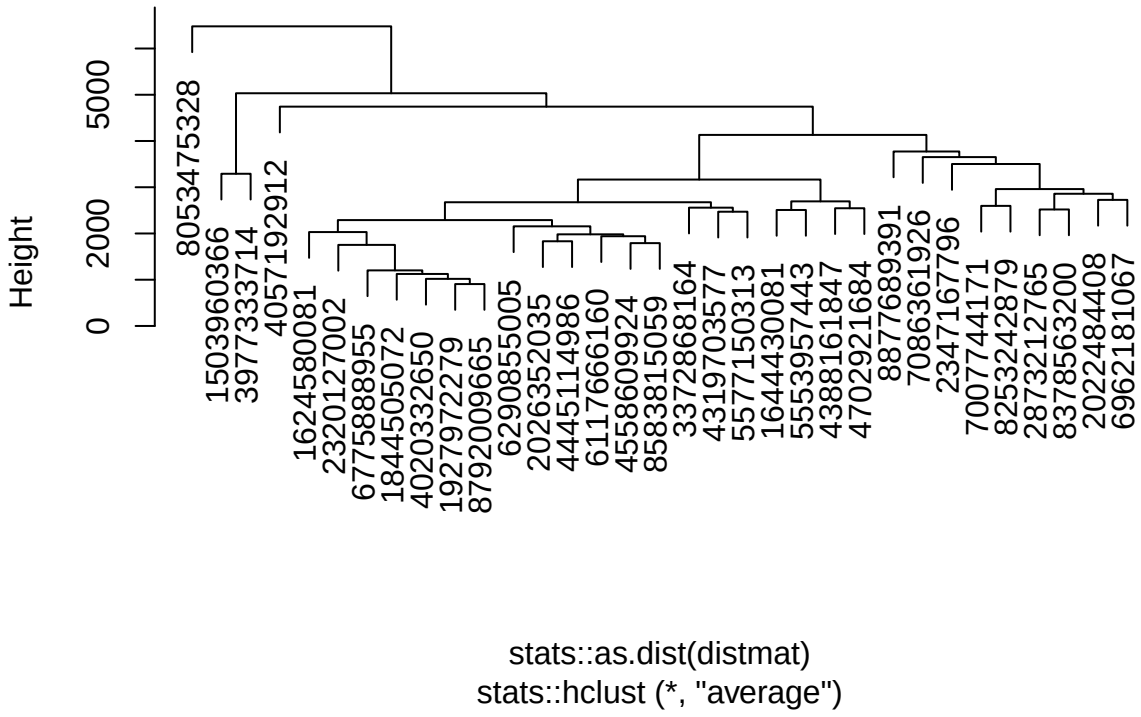
```
# Plot one particular cluster.
plot(Time_series_clusters, type = "sc", clus = 1L)
```

```
## Warning: `guides(<scale> = FALSE)` is deprecated. Please use `guides(<scale> =
## "none")` instead.
```



In the code below we perform hierarchical clustering and plot the dendrogram.

```
# Perform hierarchical clustering with 6 clusters.

Time_series_hierearchical_clusters <- tsclust(average_steps_wide, type = "h", k = 6L, distance = "dtw")

# Plot the dendrogram of the individuals clustered
plot(Time_series_hierearchical_clusters)
```



We can inspect which individuals belong to which cluster with the following line of code:

```
cutree(Time_series_hierearchical_clusters, k=6L)
```

```
## 1503960366 1624580081 1644430081 1844505072 1927972279 2022484408 2026352035
##          1          2          2          2          2          3          2
## 2320127002 2347167796 2873212765 3372868164 3977333714 4020332650 4057192912
##          2          3          3          2          1          2          4
## 4319703577 4388161847 4445114986 4558609924 4702921684 5553957443 5577150313
##          2          2          2          2          2          2          2
## 6117666160 6290855005 6775888955 6962181067 7007744171 7086361926 8053475328
##          2          2          2          3          3          3          5
## 8253242879 8378563200 8583815059 8792009665 8877689391
##          3          3          2          2          6
```

In the next stages of the project, I would include other variables such as heartrate and intensity to better match fitness buddies that are at a similar level of fitness to imporve the chances of a match working.

Below is some code to combine these different sources of data.

```
# Combine different data sources for future work

heartrate <- read_csv("heartrate_seconds_merged.csv")

heartrate$date_time <- as.POSIXct(heartrate$Time, format="%m/%d/%Y %I:%M:%S %p", tz=Sys.timezone())
minute_steps$time <- as_hms(minute_steps$date_time)
minute_steps$Individual_ID <- factor(minute_steps$Id)

new <- minute_steps %>% full_join(minute_intensity, by = c("Id", "date_time"))
```

```r
new1 <- new %>% left_join(heartrate, by = c("Id", "date_time"))

head(new1)
```

```
## # A tibble: 6 x 12
##           Id ActivityMinute.x      Steps date_time           time.x Individual_ID.x
##        <dbl> <chr>                 <dbl> <dttm>              <time> <fct>
## 1 1503960366 4/12/2016 12:00:00 AM     0 2016-04-12 00:00:00 00'00" 1503960366
## 2 1503960366 4/12/2016 12:01:00 AM     0 2016-04-12 00:01:00 01'00" 1503960366
## 3 1503960366 4/12/2016 12:02:00 AM     0 2016-04-12 00:02:00 02'00" 1503960366
## 4 1503960366 4/12/2016 12:03:00 AM     0 2016-04-12 00:03:00 03'00" 1503960366
## 5 1503960366 4/12/2016 12:04:00 AM     0 2016-04-12 00:04:00 04'00" 1503960366
## 6 1503960366 4/12/2016 12:05:00 AM     0 2016-04-12 00:05:00 05'00" 1503960366
## # ... with 6 more variables: ActivityMinute.y <chr>, Intensity <dbl>,
## #   time.y <time>, Individual_ID.y <fct>, Time <chr>, Value <dbl>
```