Fitbit R

Get your own Fitbit data

First step is to download your own data from Fitbit using `fitbitr` package (available at github: `devtools::install_github("teramonagi/fitbitr")`). You will need to get your own token (see instructions ~here) and then you will be using your own FITBIT_KEY and FITBIT_SECRET.

```
FITBIT_KEY    <- "22B7NV"
FITBIT_SECRET <- "688a2b1ce6d63f9cfd1c1a37ec06de23"
FITBIT_CALLBACK <- "http://localhost:1410/"

#token <- fitbitr::oauth_token(language = "en_US")
#saveRDS(token, file = "/Users/svetlanavinogradova/Documents/Projects/RFitbit/fitbit_token.rds")
token <- readRDS("/Users/svetlanavinogradova/Documents/Projects/RFitbit/fitbit_token.rds")
```

Whole day heart rate

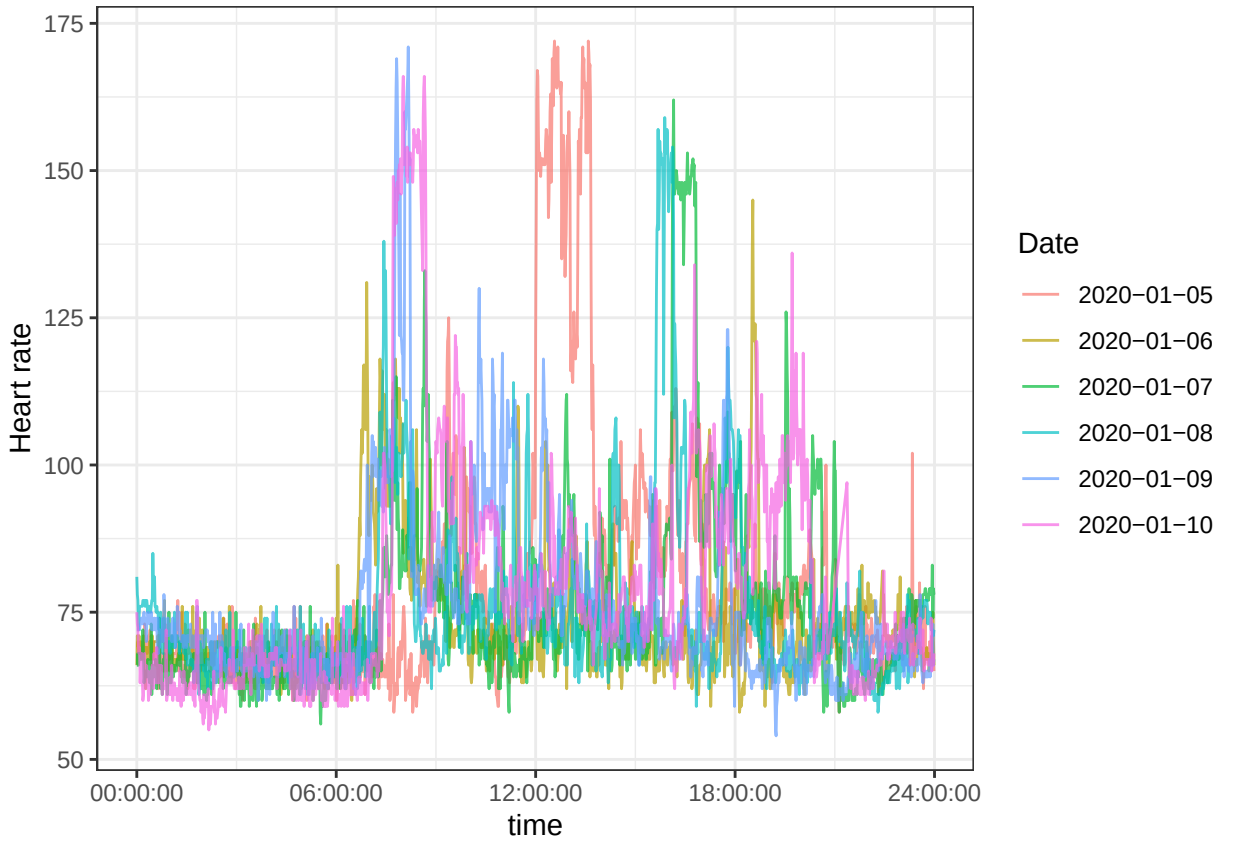
Let's asume we want to start with downloading your whole day heart rate for first 10 days for January:

```
df_hr <- get_heart_rate_intraday_time_series_multiple(token = token, date_start = "2020-01-05", date_en
df_hr$time <- times(df_hr$time)
kable(head(df_hr)) %>%  kable_styling(position = "center", full_width = F)
```

time	value	day
00:00:00	69	2020-01-05
00:01:00	69	2020-01-05
00:02:00	70	2020-01-05
00:03:00	68	2020-01-05
00:04:00	69	2020-01-05
00:05:00	69	2020-01-05

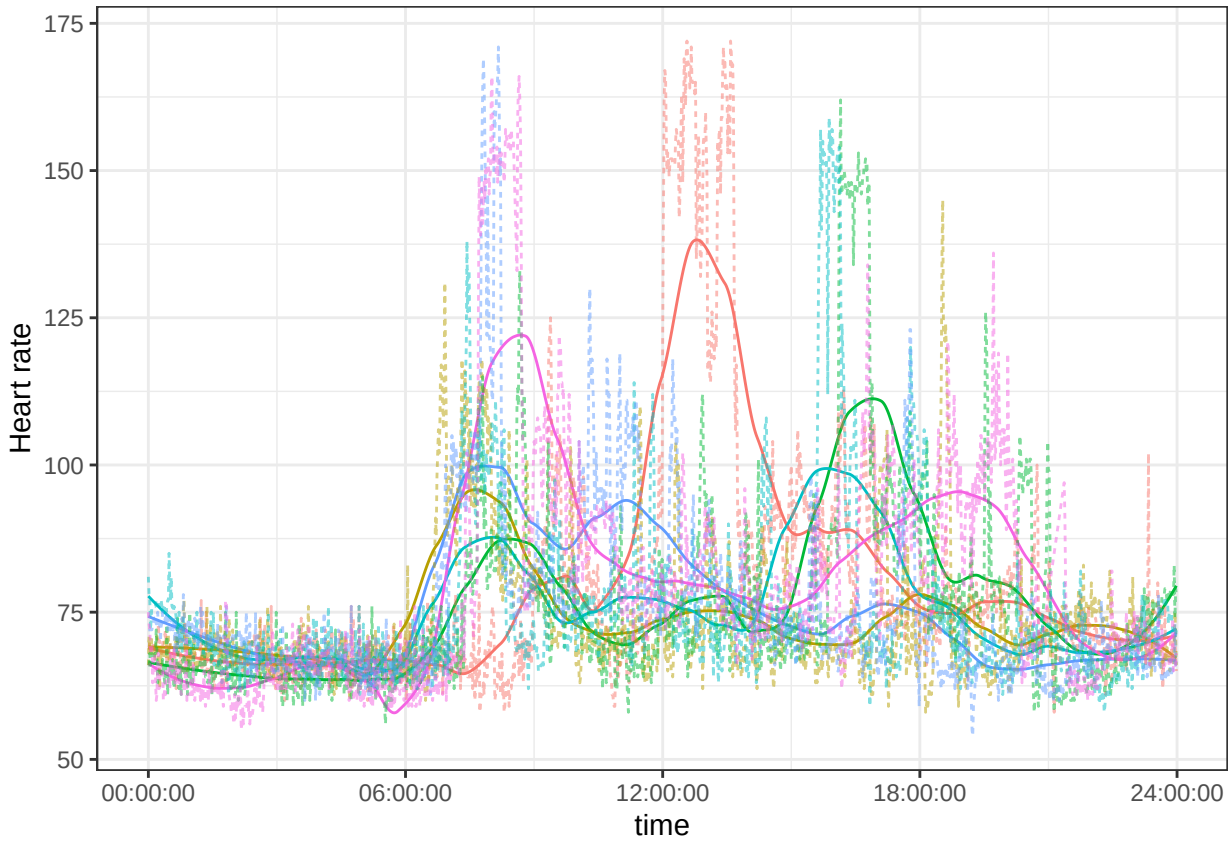
Let's now plot the heart rate:

```
breaks2 <- c(0.0000000, 0.2500000, 0.500000, 0.7500000, 0.9999999)
labels2 <- times(breaks2)
ggplot(df_hr, aes(x = time, y = value, col = as.factor(day))) +
  geom_line(alpha = 0.7) +
  scale_x_continuous(labels = labels2, breaks = breaks2) +
  ylab("Heart rate") +
  guides(color = guide_legend(title = "Date")) +
  theme_bw()
```



Now we can add loess lines to your heart rate to see if there any trends:

```
fit_loess <- function(day_selected) {
  df <- filter(df_hr, day == day_selected)
  df$time_conv <- (as.numeric(as.POSIXct((data.table::as.ITime(df$time)))) - as.numeric(as.POSIXct((data
  fit <- loess(value ~ time_conv, data = df, span = 0.25)
  preds <- predict(fit)
  df$preds <- preds
  return(df)
}
loess_res <- unique(df_hr$day) %>%
    purrr::map(fit_loess) %>%
    purrr::map_df(dplyr::bind_rows)

loess_res_long <- loess_res %>% gather(-day, -time, -time_conv, key = "class", value = "RHR")

ggplot(loess_res_long, aes(x = time, y = RHR, col = as.factor(day))) +
  geom_line(aes(linetype = class, alpha = class)) +
  scale_alpha_manual(values = c(1, 0.5)) +
  scale_x_continuous(labels = labels2, breaks = breaks2) +
  ylab("Heart rate") +
  theme_bw() +
  theme(legend.position = "None")
```

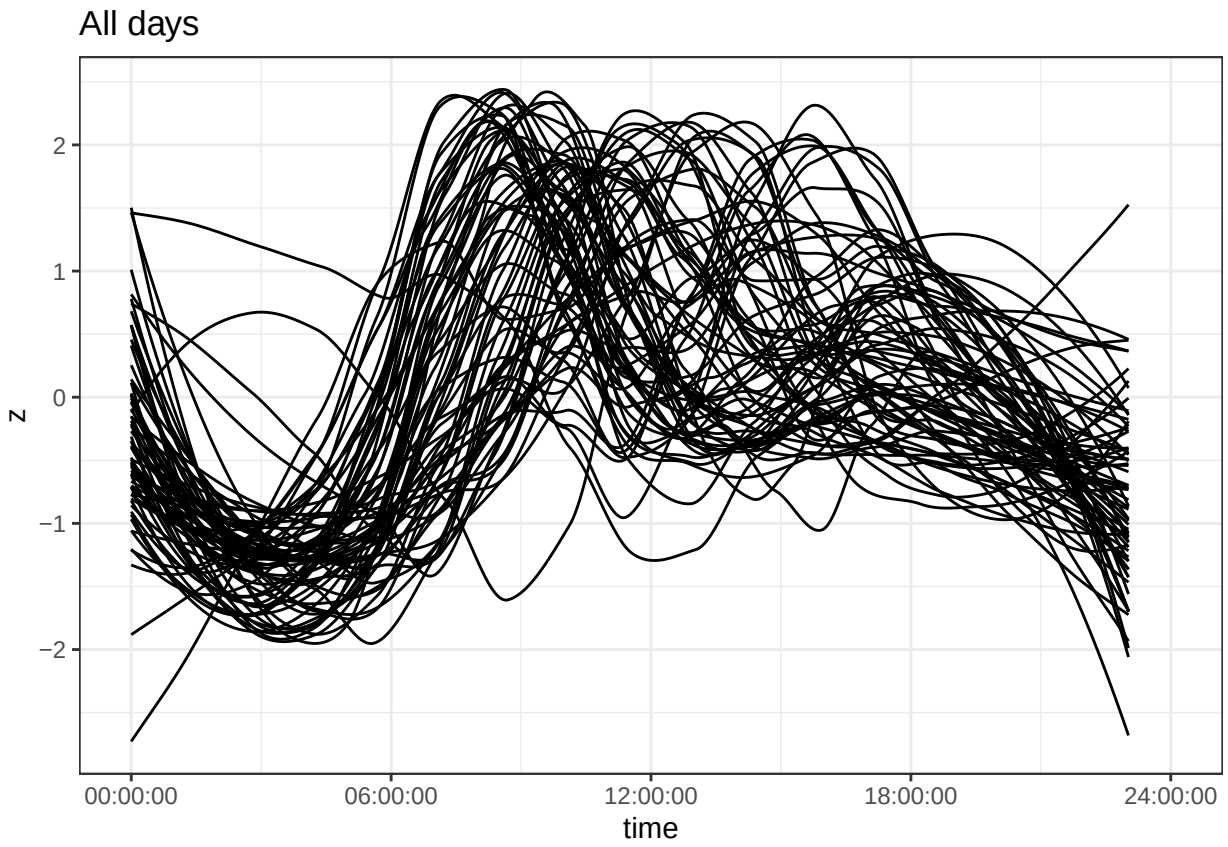


In order to have a real clustering, we will need more data:

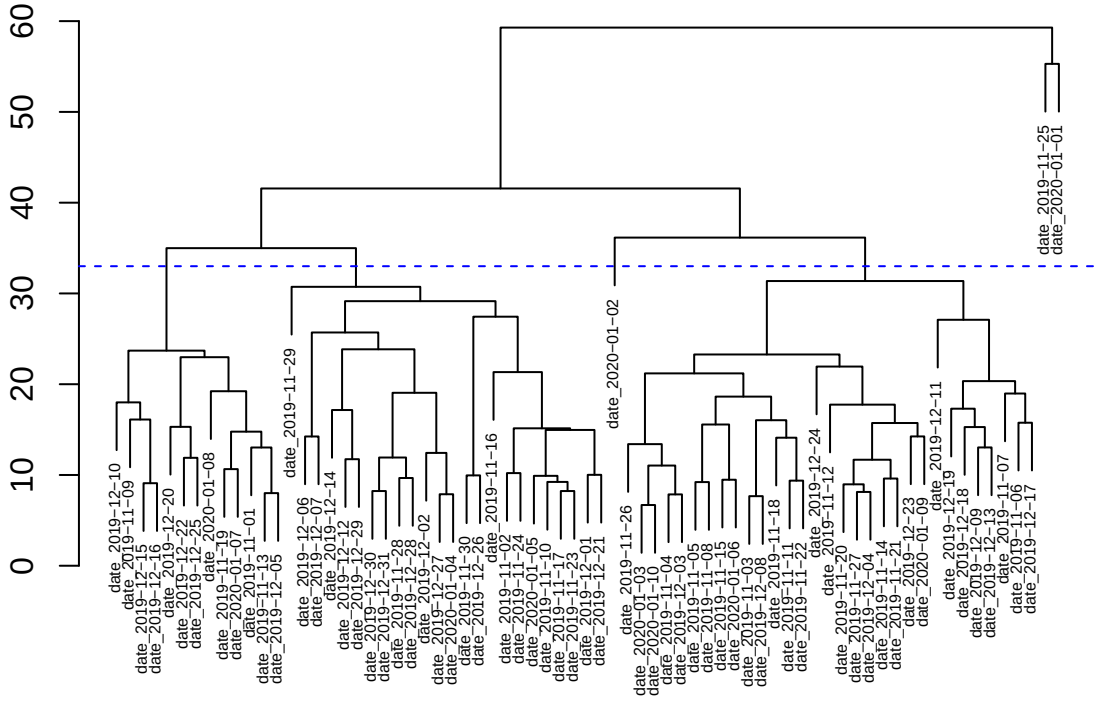
```
df_hr <- get_heart_rate_intraday_time_series_multiple(token = token, date_start = "2019-11-01", date_en
df_hr$time <- as.POSIXct(df_hr$time,format = "%H:%M:%S")

time_range <- 1:1500
breaks2 <- seq(min(time_range), max(time_range), length.out = 5)
fit_loess <- function(day_selected) {
  df <- filter(df_hr, day == day_selected)
  df$time_conv <- (as.numeric(as.POSIXct((data.table::as.ITime(df$time)))) - as.numeric(as.POSIXct((data
  fit <- loess(value ~ time_conv, data = df, span = 0.5)
  preds <- predict(fit, newdata = data.frame(time_conv = time_range))
  list(fit = fit, preds = preds)
}
loess_res <- lapply(unique(df_hr$day), fit_loess)
z_score_trajectories_wide <- scale(map_dfc(loess_res, "preds"))
colnames(z_score_trajectories_wide) <- paste0("date_", unique(df_hr$day))


z_score_trajectories_long <- z_score_trajectories_wide %>%
  as_tibble() %>%
  mutate(time = time_range) %>%
  mutate(time = as.integer(time)) %>%
  gather(key = day, value = z, -time)
z_score_trajectories_long %>%
  ggplot(aes(x = time, y = z, group = day)) +
  geom_line() +
```

```
  scale_x_continuous(labels = labels2, breaks = breaks2) +
  labs(title = "All days") +
  theme_bw()
```

```
## Warning: Removed 4331 rows containing missing values (geom_path).
```



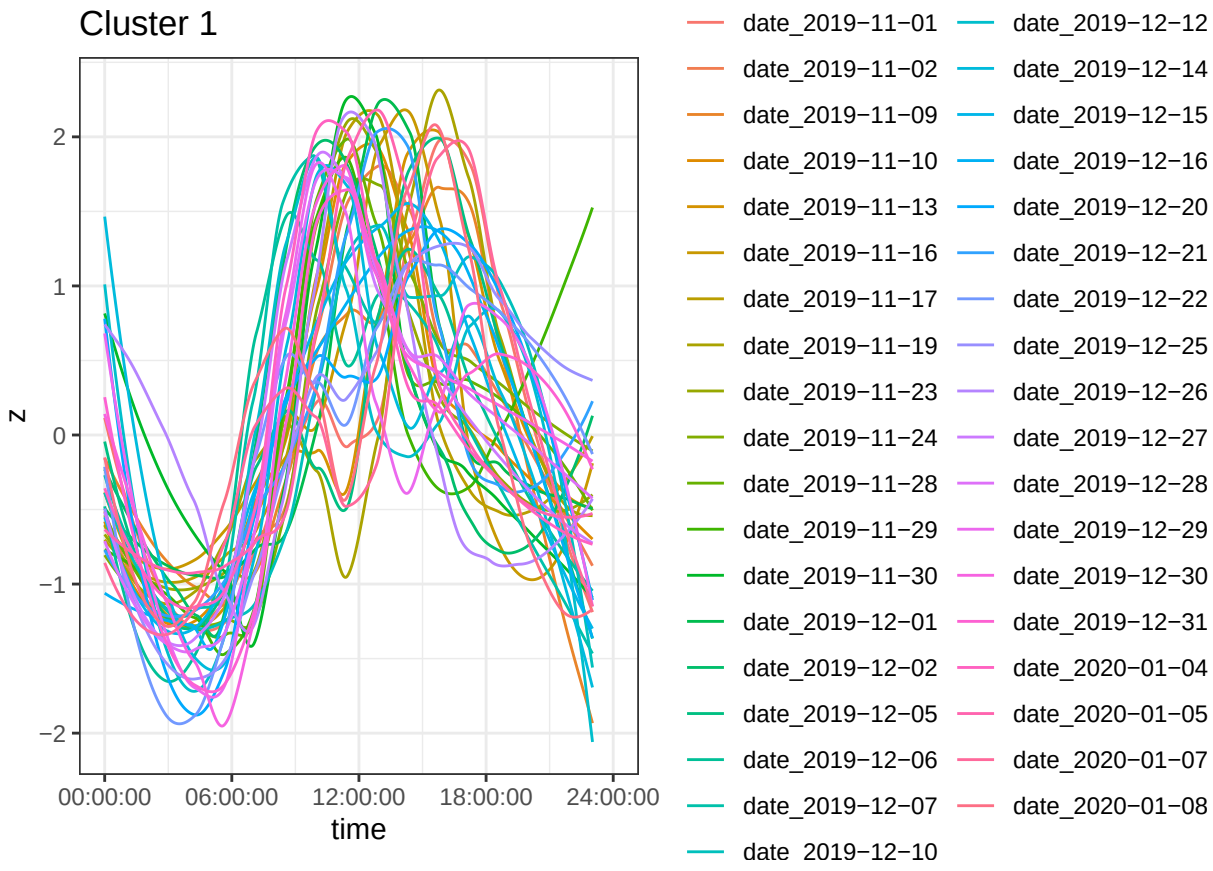

```
hc <- hclust(dist(t(z_score_trajectories_wide)), method = "average")
par(mar = c(0, 4, 4, 2))
plot(hc, main = "Heart rate trajectory-based hierarchical clustering", ylab = "", cex = 0.5)
abline(a = 33, b = 0, col = "blue", lty = 2)
```



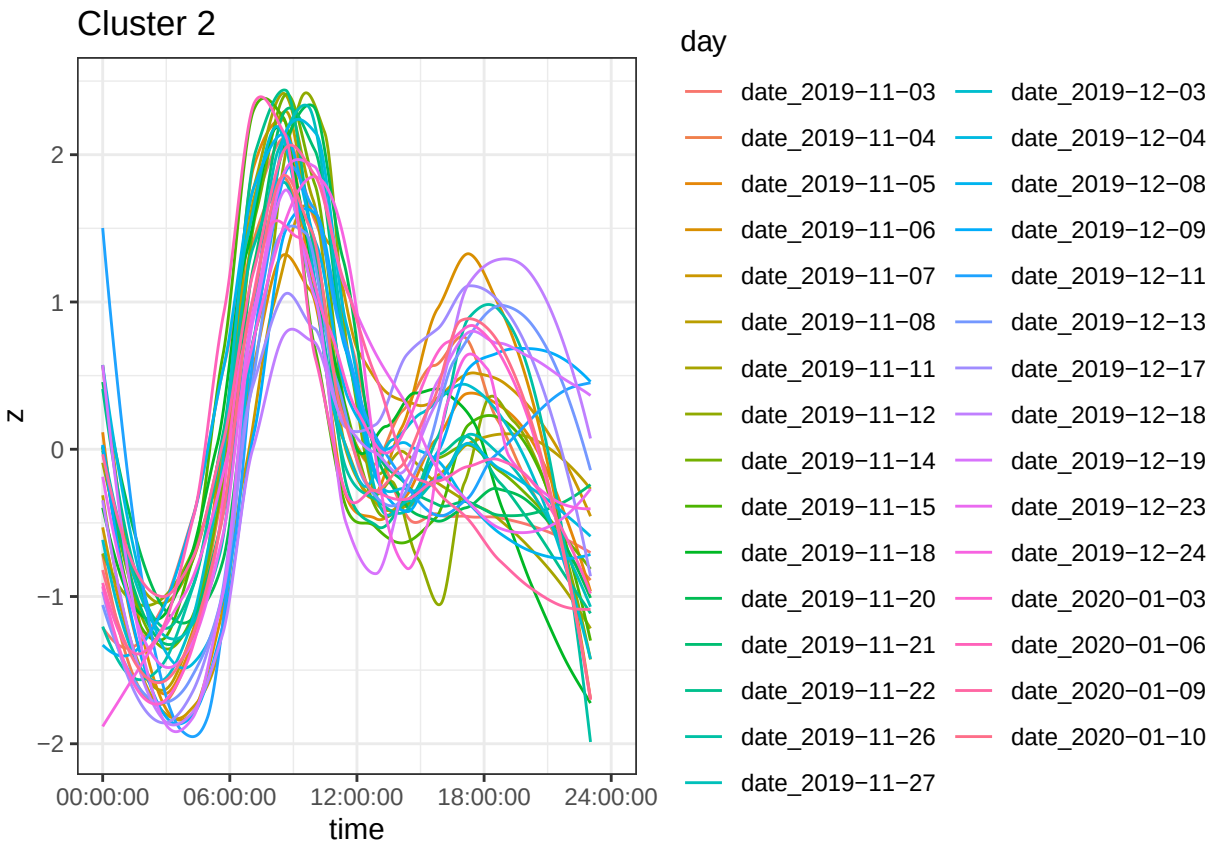

```
k <- 5
assignments <- cutree(hc, k)
```

```
age_trajectory_plts <- lapply(
  names(sort(table(assignments), decreasing = T)),
  function(clust) {
    z_score_trajectories_long %>%
      filter(day %in% names(assignments)[assignments == clust]) %>%
      ggplot(aes(x = time, y = z, color = day)) +
      geom_line() +
      labs(paste0("Cluster", clust)) +
      scale_x_continuous(labels = labels2, breaks = breaks2) +
      theme(legend.title = element_blank()) +
      theme_bw()
  }
)
silent <- lapply(seq(1, k), function(i) {
  print(age_trajectory_plts[[i]] + labs(title = paste("Cluster", i)))
})
```

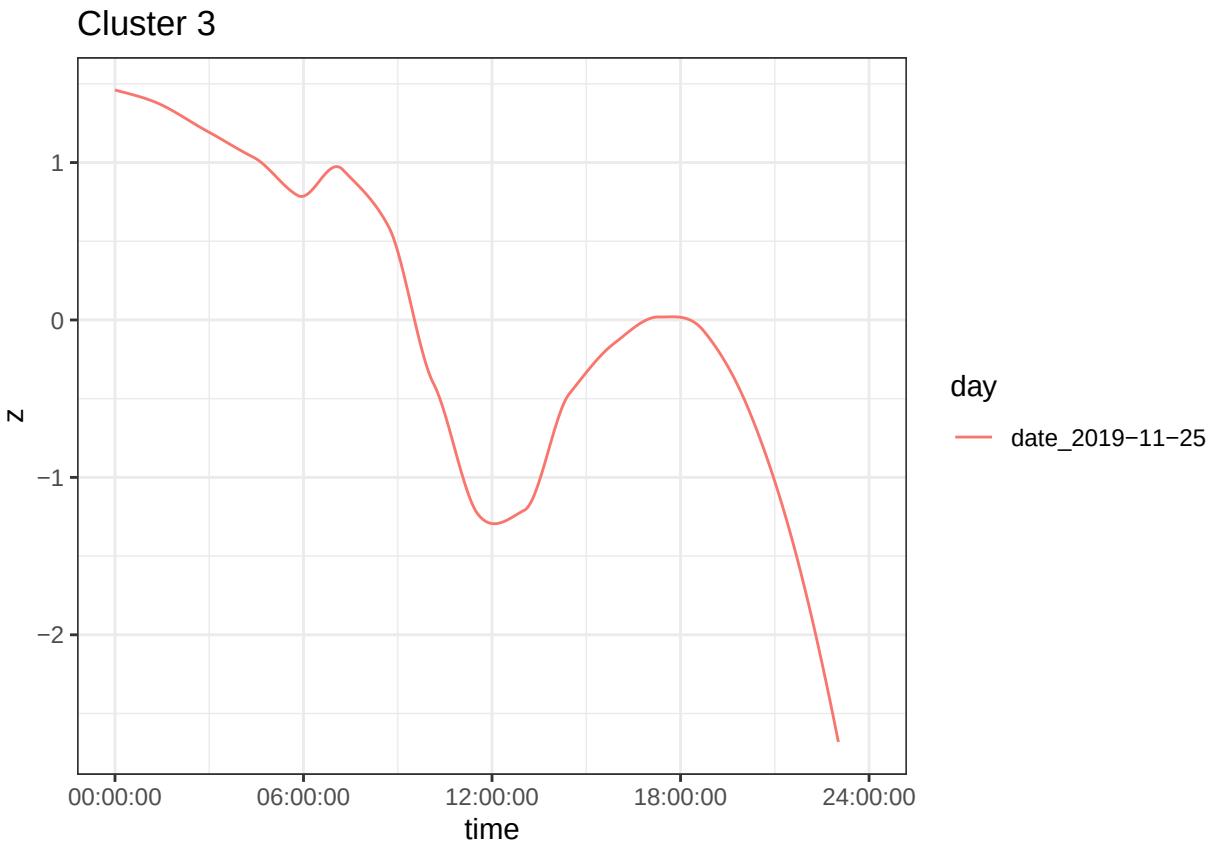
```
## Warning: Removed 2257 rows containing missing values (geom_path).
```



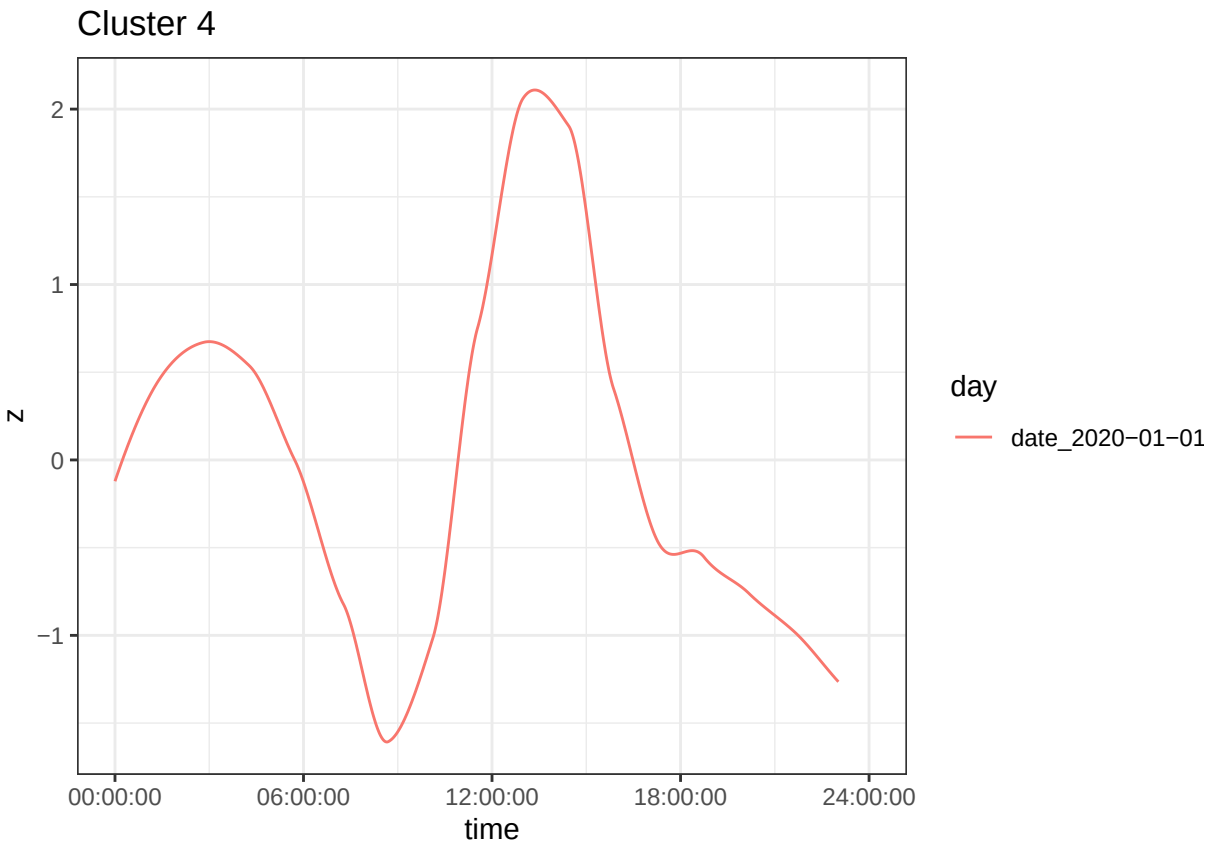

```
## Warning: Removed 1891 rows containing missing values (geom_path).
```



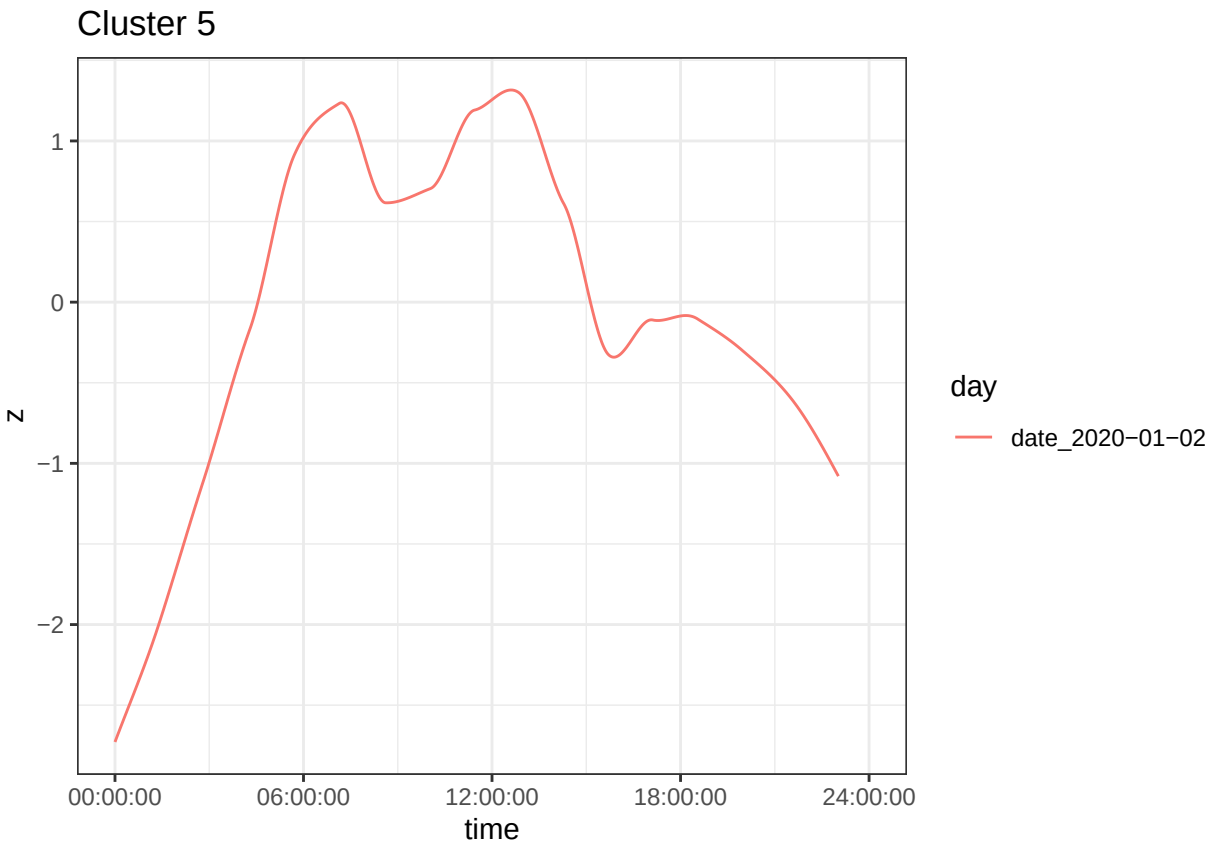

```
## Warning: Removed 61 rows containing missing values (geom_path).
```




```
## Warning: Removed 61 rows containing missing values (geom_path).
```



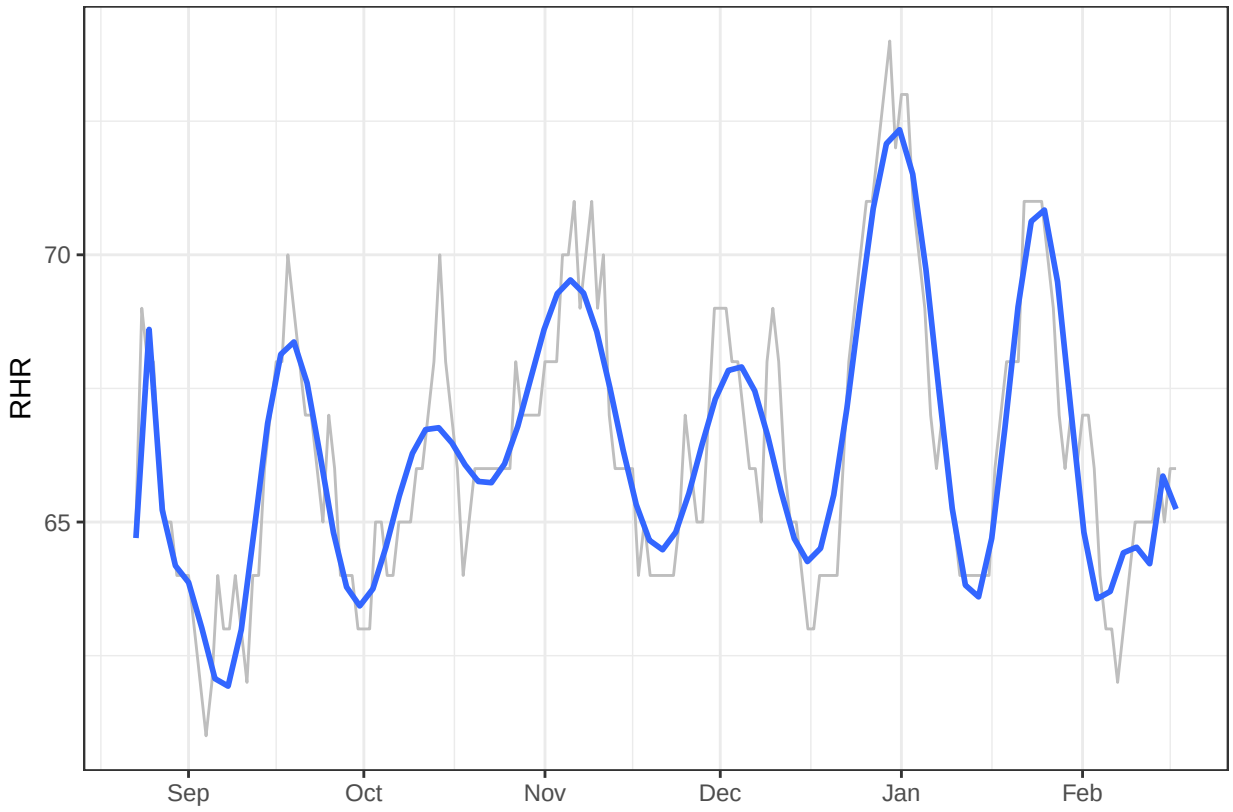

```
## Warning: Removed 61 rows containing missing values (geom_path).
```



Resting heart rate (RHR)

Now let's look at resting heart rate:

```
rhr <- get_activity_time_series(token, "restingHeartRate", date = as.character(Sys.Date()), period = "ma
rhr$dateTime <- as.Date(rhr$dateTime)
rhr$value <- as.numeric(rhr$value)
ggplot(rhr, aes(x = dateTime, y = value)) +
  geom_line(col = "gray") +
  stat_smooth(aes(x = dateTime, y = value), method = "lm", formula = y ~ poly(x, 21), se = FALSE) +
  theme_bw() +
  ylab("RHR") +
  xlab("")
```


RHR
70
65
Sep
Oct
Nov
Dec
Jan
Feb