Strive

Bjørn Fjukstad

Mon Mar 12 11:05:24 2018

Introduction

In this short example I'll show how to access the Strava API to get and plot data from my runs.

Setup

We need two packages: `rStrava` to get data and `dplyr` to wrangle it.

```
library("rStrava")
library(dplyr)
```

Before you start, you'll need to create a Strava app and put the different tokens etc. as environment variables in your .Renviron file. See the rStrava README for more information: https://github.com/fawda123/rStrava. We also need a Google Maps key. Again, see the rStrava README for details. In the example code below I'm using a cached oauth token, but you can use the commented code to generate a new one.

```
app_name = Sys.getenv("STRAVA_APP_NAME")
app_client_id = Sys.getenv("STRAVA_APP_CLIENT_ID")
app_client_secret = Sys.getenv("STRAVA_CLIENT_SECRET")
google_maps_key = Sys.getenv("GOOGLE_MAPS_KEY")
if (app_name == "" ||
    app_client_id == "" ||
    app_client_secret == "" || google_maps_key == "") {
  stop(
    "Please set STRAVA_APP_NAME, STRAVA_APP_CLIENT_ID and STRAVA_APP_SECRET, GOOGLE_MAPS_KEY"
  )
}

# token = strava_oauth(app_name,
#                      app_client_id,
#                      app_client_secret)
# stoken = httr::config(token=stoken)
stoken <- httr::config(token = readRDS('.httr-oauth')[[1]])
```

Retrieving and exploring data

Get all activites using the `get_activity_list` function. It will retrieve the full list of all my activities.

```
activities = rStrava::get_activity_list(stoken)
```

```
## Warning in strptime(x, fmt, tz = "GMT"): unknown timezone 'zone/tz/2018c.
## 1.0/zoneinfo/Europe/Oslo'
```

To filter out some activities we only select runs that are recorded after 1. January 2017 and have hartrate data.

```
applicable_runs = lapply(activities, function(x) {
  date = as.Date(x$start_date)
  validDate = FALSE
  startDate = as.Date(as.POSIXct("2017-01-01"))
  if(date > startDate) {
    validDate = TRUE
  }
  if (x$type == "Run" && x$has_heartrate && validDate) {
    return(TRUE)
  } else {
    return(FALSE)
  }
})
runs = activities[unlist(applicable_runs)]
```

My latest run to Keiservarden in Bodø.

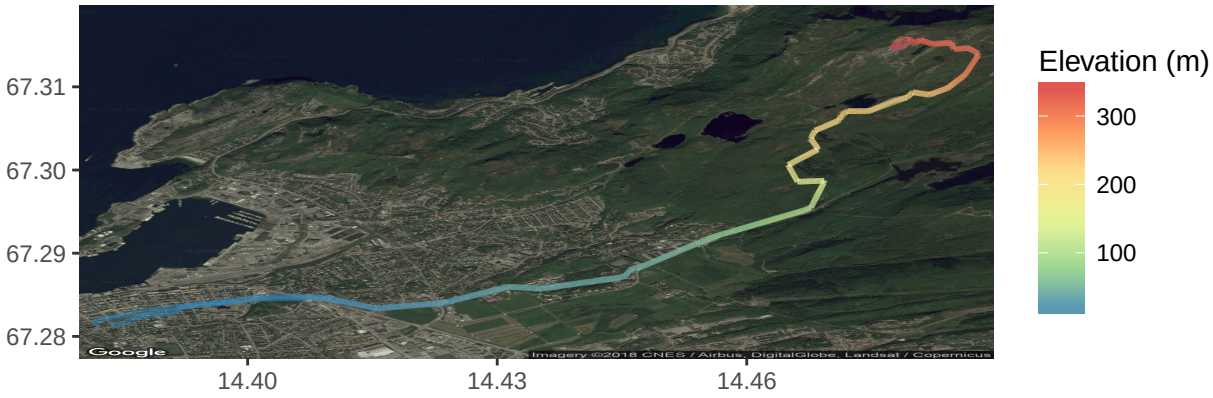
First, print out *some* of the information we get from Strava. We get an ID, my athlete id, how it was uploaded (from my Garmin account) and the name of the activity.

```
head(runs[[1]])
```

```
## $id
## [1] 1445422746
##
## $resource_state
## [1] 2
##
## $external_id
## [1] "garmin_push_2545972834"
##
## $upload_id
## [1] 1558794690
##
## $athlete
## $athlete$id
## [1] 2746741
##
## $athlete$resource_state
## [1] 1
##
##
## $name
## [1] "Lørdagstur til Keiservarden"
```

`rStrava` even provides functions to plot the activity on a map. Well use a satalite image and color the trace using the elevation.

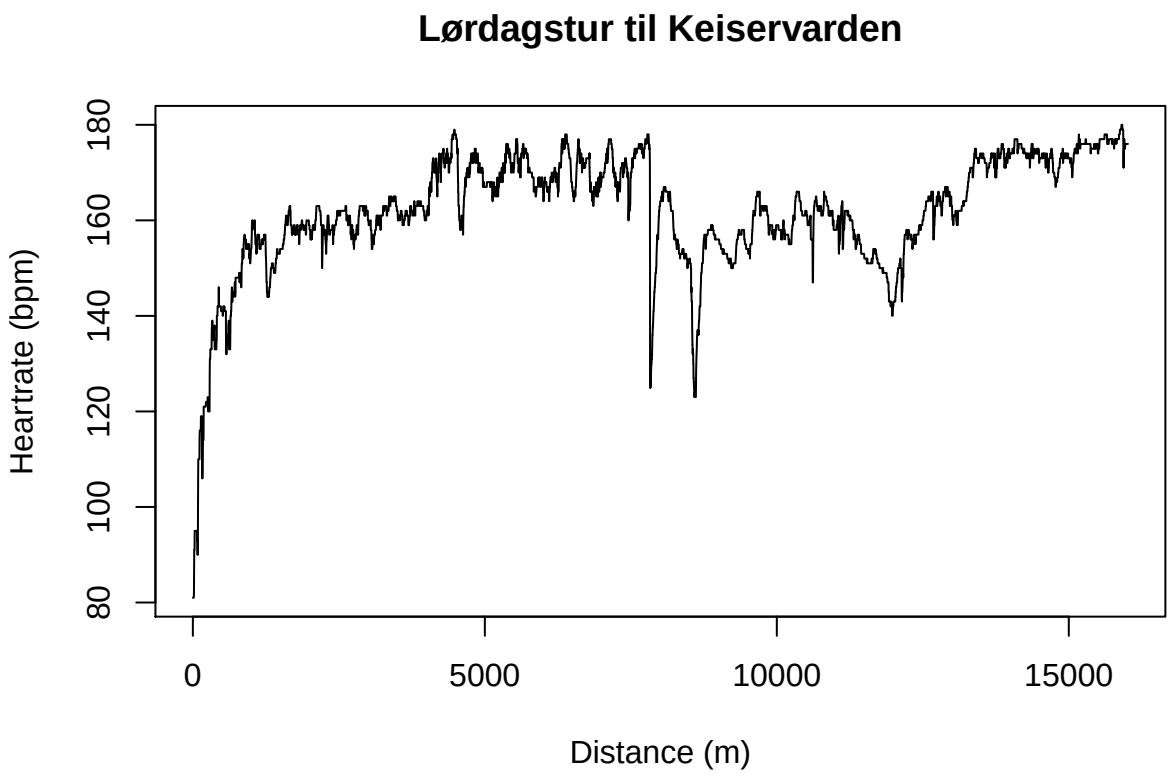
```
rStrava::get_heat_map(runs, key=google_maps_key, acts=1, col = 'Spectral',
                      maptype = 'satellite',  size = 1,  dist = F, add_elev = T,
                      f = 0.5)
```



To get detailed information from each activity we'll need to use the `get_streams` function. Here we indicate that we want to get all heartrate measurements. We then plot the heartrate against distance traveled.

```
heartrate = rStrava::get_streams(stoken, id = runs[[1]]$id,
                                 types = list('heartrate'))

distance = heartrate[[1]]$data
hr = heartrate[[2]]$data
plot(distance, hr, type = "l", main=runs[[1]]$name, xlab="Distance (m)",
     ylab="Heartrate (bpm)")
```



An overview of multiple runs

We can also explore average heartrates, distances and average paces for each run.

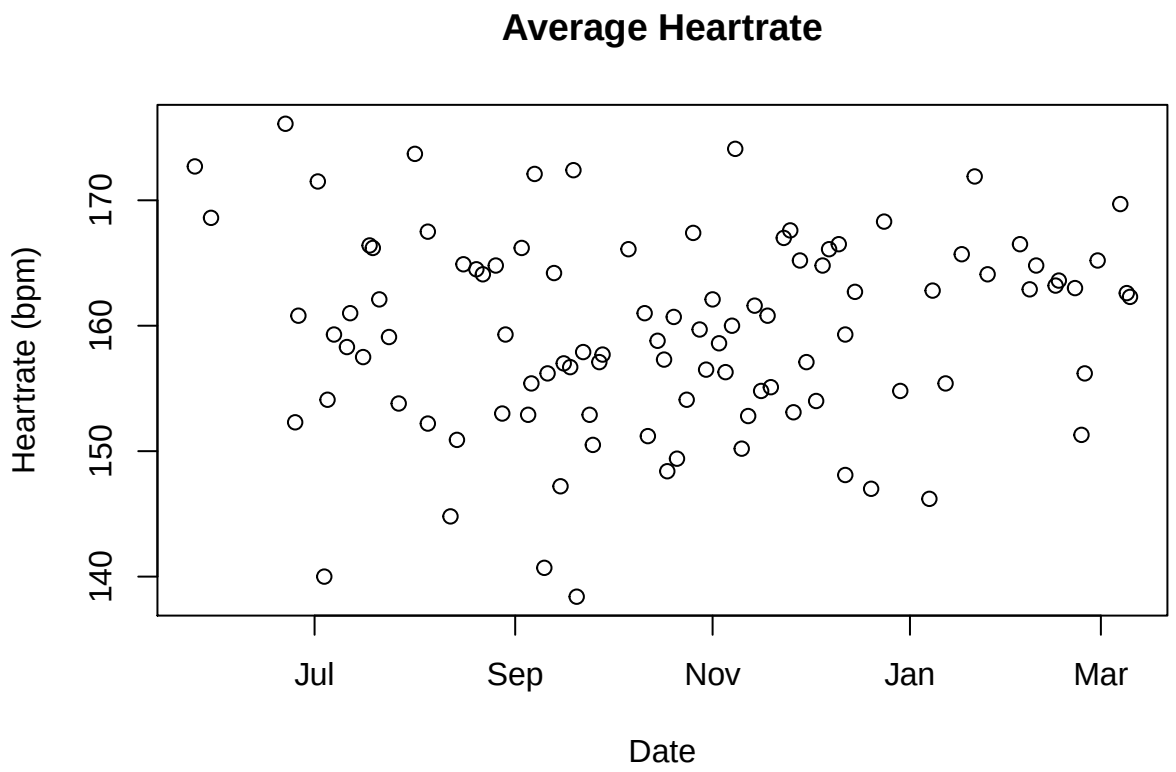
```
dates = unlist(lapply(runs, function(x) {
  x$start_date
}))
dates = as.Date(dates)
```

Get and plot heartrate, distance and elevation data for all my runs.

```
heartrates = unlist(lapply(runs, function(x) {
  x$average_heartrate
}))
summary(heartrates)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   138.4   154.1   159.7   159.5   164.9   176.1
```

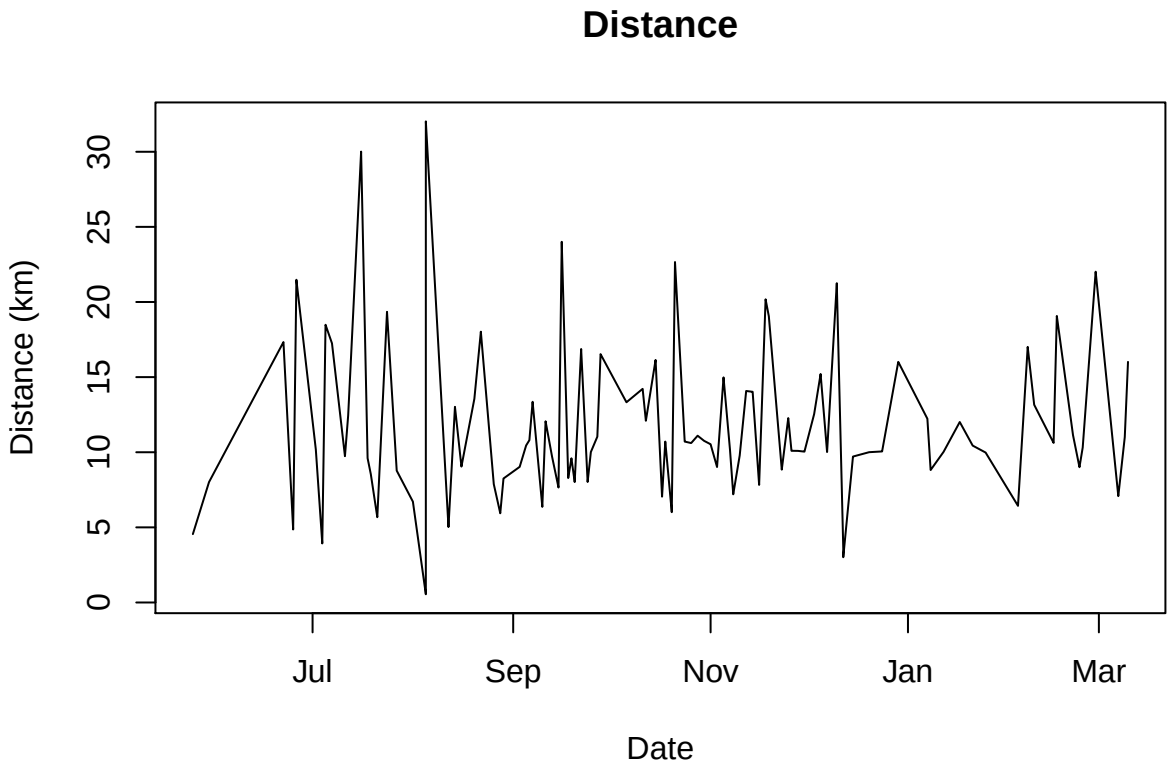
```
plot(dates, heartrates, main="Average Heartrate", xlab="Date",
     ylab="Heartrate (bpm)")
```




```
# Divide the distance by 1000 to get it in kilometers instead of meters.
distances = unlist(lapply(runs, function(x) {
  x$distance/1000
}))
summary(distances)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.5471  8.7752 10.4426 11.7169 14.0219 32.0238
```

```
plot(dates,distances, type = "l", main="Distance", xlab="Date",
     ylab="Distance (km)")
```

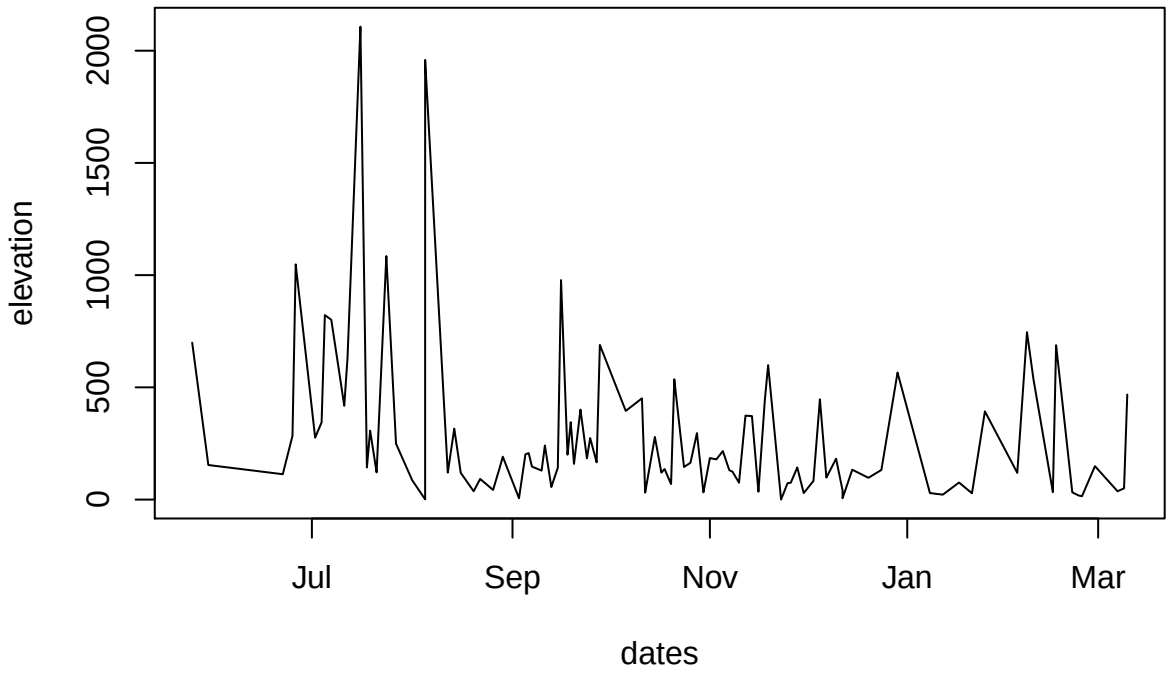


The first peak on the plot is when I ran the Tromsø Skyrace course to get a feeling it it. The second peak is the actual race. The same trend is on the elevation plot below.

```
elevation = unlist(lapply(runs, function(x) {
  x$total_elevation_gain
}))
summary(elevation)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##     0.0    76.0   149.0   279.7   372.0  2107.0
```

```
plot(dates, elevation, type = "l")
```

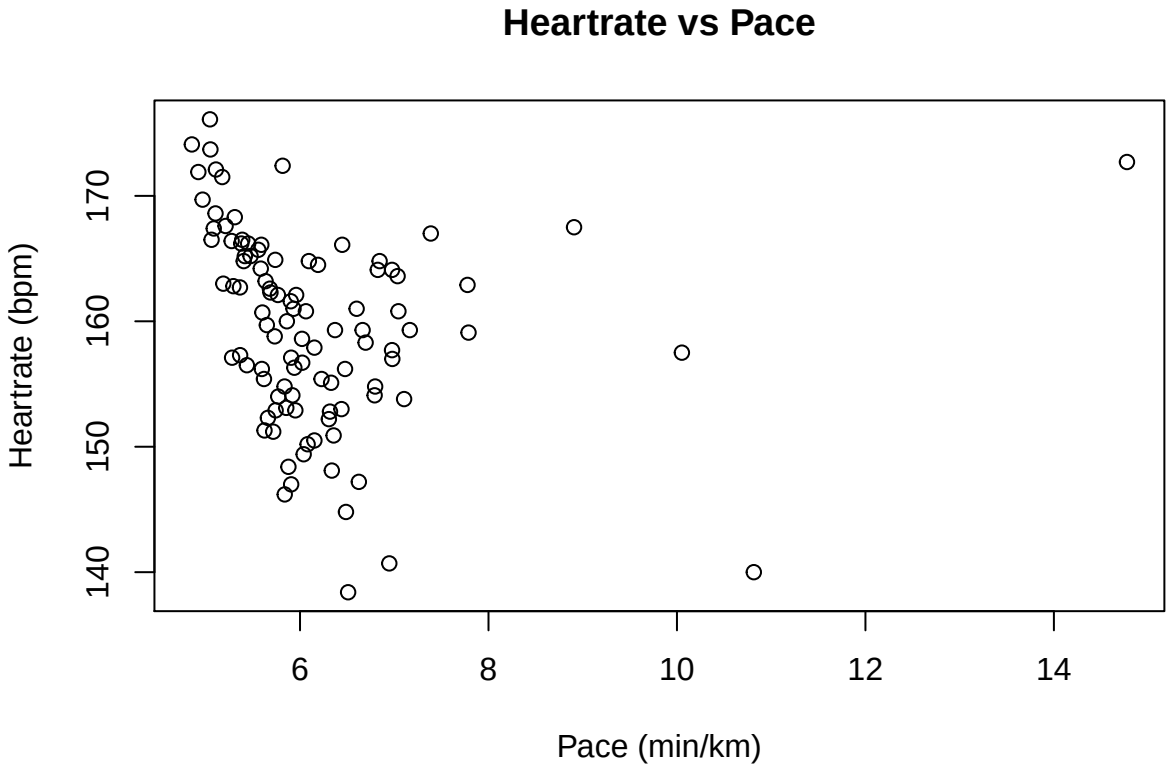


A look at average pace and heartrate.

Hopefully this could give an idication that I run faster at lower heartrates! Since Strava provides average speed at m/s, we'll convert them to average pace which is what I usually use.

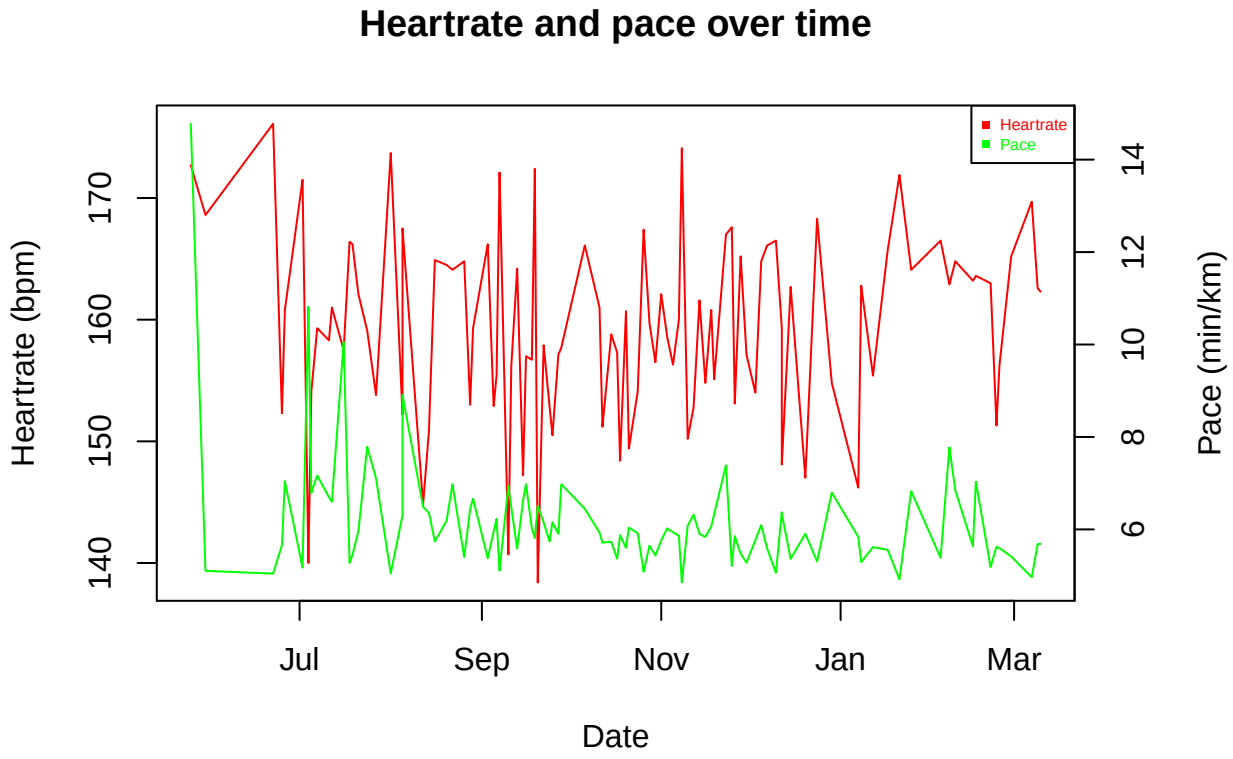
```
pace = unlist(lapply(runs, function(x) {
  km_hr = x$average_speed  * (3600 / 1000)
  min_km = 60 / km_hr
  return(min_km)
}))

plot(pace, heartrates, main="Heartrate vs Pace", xlab="Pace (min/km)",
     ylab="Heartrate (bpm)")
```



From the plot we can see a trend that I have a higher heartrate when I run at a faster pace. This is what we could expect. We can also se an outlier (top right) when I did a hill repeat in Tromsø. The "*course*" was 1km and had over 300m of climb, so while my heartrate was high up there, my pace wasn't very impressive. It could be interesting here to use Strava's grade adjusted pace (GAP) and not the raw pace metric. The last plot I want to generate is to see how the average pace of each run evloves over the year, in addition to the heartrate of the run. I did not want to spend that much time working on it, so here's a first example! To get a better overview I think we'll have to adjust the pace using the elevation gain of each activity.

```
par(cex=1)
par(mar = c(5, 4, 4, 4) + 0.3)
plot(dates, heartrates, type="l", col="red", main="Heartrate and pace over time",
     xlab="Date", ylab="Heartrate (bpm)")
par(new = TRUE)
plot(dates, pace, type="l", col="green",
     axes = FALSE, bty = "n",
     xlab="",ylab="")
axis(side=4, at = pretty(range(pace)))
mtext("Pace (min/km)", side=4, line=3)
par(cex = 0.5)
legend("topright",legend=c("Heartrate","Pace"),
       text.col=c("red","green"),pch=c(15,15),col=c("red","green"))
```


Heartrate and pace over time
Heartrate
Pace
Heartrate (bpm)
140
150
160
170
Pace (min/km)
6
8
10
12
14
Jul
Sep
Nov
Jan
Mar
Date