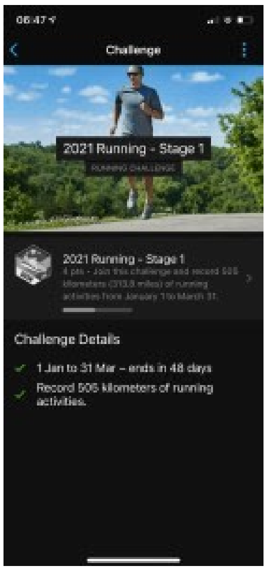
Many running races and events have been cancelled or disrupted due to the pandemic. To fill the void, I've been taking on Garmin's "Challenges".

In Garmin Connect, you can accept a challenge set by Garmin (I think users can challenge each other too). Completion of these challenges gives the user points, which I've become somewhat obsessed with.

In late December I accepted the **2021 Running Stage 1** challenge: to run 505 km by the end of March. The name Stage 1 suggests that there will be 4 similar challenges which would presumably take me over the 2000 km running total for the year. This sounds like a fun goal – off we go!



4 pts - Join this challenge and record 505 kilometers (313.8 miles) of running activities from January 1 to March 31.

1 Jan to 31 Mar – ends in 48 days

Record 505 kilometers of running activities.

Challenge accepted

The app shows progress with a bar and you can tap through to see the distance covered so far, but it is difficult to know whether I am "on target" with respect to the time remaining for the challenge.

Monitoring progress using R

As described previously, Garmin Connect in a browser allows for easy export of summary data in csv format.

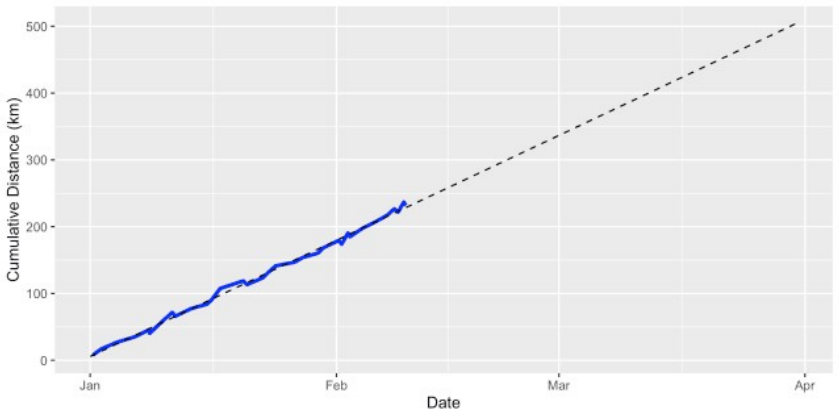
I downloaded my data, ensuring that it included activities from the start of 2021. This csv was added to the `Data/` folder of my RStudio project. I am currently using a uniform way of processing data using a folder structure which makes coding easier. The script starts by checking that this folder structure is in place. The csv goes into `Data/` and I can add further csv files, as the challenge progresses, which contain the latest activities, to keep monitoring the target.

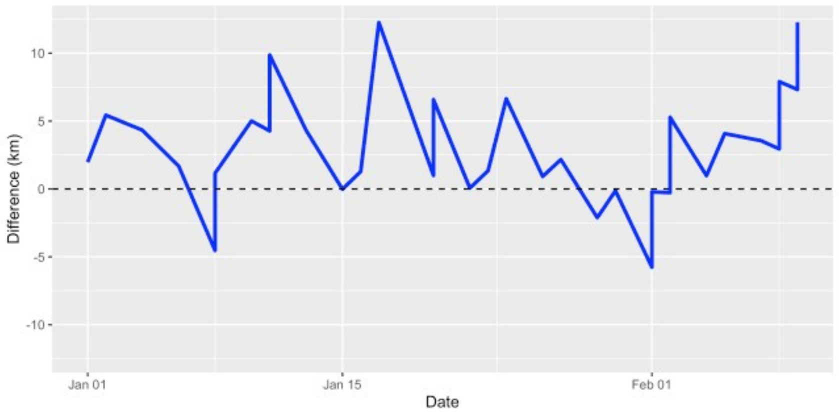
The script loads in all csv files and makes a data frame of them all, removing any duplicates. I created a function that allows me to filter the data for the time window we are interested in (for the challenge) and to only look at Running (and Treadmill Running) activities. It also calculates the cumulative distance run during this window. Now, this distance needs to be compared to the target, at that point in time.

I could do this by plotting Cumulative Distance by Date and adding an `abline`. However, I also wanted to look at my distance "balance", i.e. am I ahead or behind target?

So, another function creates a day-by-day data frame of the target. This can be used for plotting, and it allows me to calculate my distance "balance".

The script is below. This code is reusable for future stages of the challenge or for monitoring progress against an annual/monthly running target you've set yourself.

- 

Progress
- 

Balance

Two outputs to monitor progress towards a running goal

The challenge means running just over 5.5 km every day and so far progress is good.

Generally my running balance has been “in the black” and I could see that it was OK take a rest day yesterday!

Just show me the code

```
require(ggplot2)
require(hms)

## Setup preferred directory structure in wd
ifelse(!dir.exists("Data"), dir.create("Data"), "Folder exists already")
ifelse(!dir.exists("Output"), dir.create("Output"), "Folder exists already")
ifelse(!dir.exists("Output/Data"), dir.create("Output/Data"), "Folder exists
already")
ifelse(!dir.exists("Output/Plots"), dir.create("Output/Plots"), "Folder
exists already")
ifelse(!dir.exists("Script"), dir.create("Script"), "Folder exists already")

## functions

compare2target <- function(activity,fromStr,toStr,df) {
  # filter for activity
  df_window <- subset(df,grepl(tolower(activity),tolower(df_all$
Activity.Type)))
  # activities within the window
  fromDate <- as.Date(fromStr)
  toDate <- as.Date(toStr)
  df_window <- subset(df_window, as.Date(df_window$Date) >= fromDate &
as.Date(df_window$Date) <= toDate)
  # put them in order
  df_window <- df_window[order(as.Date(df_window$Date)),]
  df_window$Cumulative.Distance <- cumsum(df_window$Distance)

  return(df_window)
}

maketarget <- function(fromStr,toStr,km) {
  temp <- seq(as.Date(fromStr), as.Date(toStr), by="days")
  cumdist <- seq(km / length(temp), km, by = km / length(temp))
  df <- data.frame(Date = as.POSIXct(temp),
                   Cumulative.Distance = cumdist)

  return(df)
}

## main script

all_files <- list.files("Data", pattern = "*.csv", full.names = TRUE)
df_all <- read.csv(all_files[1], header = TRUE, stringsAsFactors=FALSE)
for (filename in all_files[-1]) {
    df_temp <- read.csv(filename, stringsAsFactors=FALSE)
    df_all <- rbind(df_all, df_temp)
}
# remove duplicates
df_all <- df_all[!duplicated(df_all), ]
# format Date column to POSIXct
df_all$Date <- as.POSIXct(strptime(df_all$Date, format = "%Y-%m-%d %H:%M:
%S"))
df_all <- compare2target("running","2021-01-01","2021-03-31",df_all)
df_target <- maketarget("2021-01-01","2021-03-31", 505)
# wrangle data frames to have matching date columns and then merge, then find
difference
# between the cumulative distance and the target
df_temp <- df_all
df_temp$Date <- as.Date(df_all$Date)
df_temp2 <- df_target
df_temp2$Date <- as.Date(df_target$Date)
df_merge <- merge(x = df_temp,
                  y = df_temp2,
                  by = "Date",
                  all.x = TRUE)
df_merge$Difference <- df_merge$Cumulative.Distance.x -
df_merge$Cumulative.Distance.y

# plot out cumulative distance over time compared to target
p1 <- ggplot(data = df_all, aes(x = Date, y = Cumulative.Distance)) +
  geom_line(colour = "blue", size = 1.2) +
  geom_line(data = df_target, linetype = 2) +
  labs(x = "Date", y = "Cumulative Distance (km)")
p1
# plot out how it's going wrt to target
```

```
p2 <- ggplot(data = df_merge, aes(x = Date, y = Difference)) +
  geom_line(colour = "blue", size = 1.2) +
  geom_hline(yintercept = 0, linetype = 2) +
  ylim(-max(abs(df_merge$Difference)),max(abs(df_merge$Difference))) +
  labs(x = "Date", y = "Difference (km)")
p2

# save all plots
ggsave("Output/Plots/progress.png", plot = p1, width = 8, height = 4, dpi =
"print")
ggsave("Output/Plots/difference.png", plot = p2, width = 8, height = 4, dpi =
"print")
```

The trickiest part, as usual with this type of data, is wrangling with date/time formats. I'm not happy with the part of the script where I switch Date/POSIX. It works but it's hacky. I'll use this code quite a bit this year, so I might edit this later. Always happy for suggestions to improve the code.