Garmin Connect has a number of plots built in, but to take a deeper dive into all your fitness data, you need to export a CSV and fire up R. This post is a quick guide to some possibilities for running data.

There’s a few things that I wanted to look at. For example, how does my speed change through the year? How does that compare to previous years? If I see some trends, is that the same for short runs and long runs? I wanted to look at the cumulative distance I’d run each year… There’s a lot of things that would be good to analyse.

Garmin Connect has a simple way to export data as a CSV. There are other ways to get your data, but the web interface is pretty straightforward. To export a CSV of your data, head to the Garmin Connect website, login and select Activities, All Activities. On this page, filter the activities for whatever you want to export. I clicked Running (you can filter some more if you want), and then scrolled down letting the data load onto the page until I went back as far as I wanted. In the top right corner, you click Export CSV and you will download whatever is displayed on the page.

The code to generate these plots, together with some data to play with can be found [here](https://github.com/quantixed/GarminCSVr).

Now in R, load in the CSV file

require(ggplot2)

require(dplyr)

require(hms)

file\_name <- file.choose()

df1 <- read.csv(file\_name, header = TRUE, stringsAsFactors = FALSE)

We have a data frame, but we need to rejig the Dates and a few other columns before we can start making plots.

# format Date column to POSIXct

df1$Date <- as.POSIXct(strptime(df1$Date, format = "%Y-%m-%d %H:%M:%S"))

# format Avg.Pace to POSIXct

df1$Avg.Pace <- as.POSIXct(strptime(df1$Avg.Pace, format = "%M:%S"))

# make groups of different distances using ifelse

df1$Type <- ifelse(df1$Distance < 5, "< 5 km", ifelse(df1$Distance < 8, "5-8 km", ifelse(df1$Distance < 15, "8-15 km", ">15 km")))

# make factors for these so that they're in the right order when we make the plot

df1$Type\_f = factor(df1$Type, levels=c("< 5 km","5-8 km","8-15 km", ">15 km"))

Now we can make the first plot. The code for the first one is below, with all the code for the other plots shown below that.

# plot out average pace over time

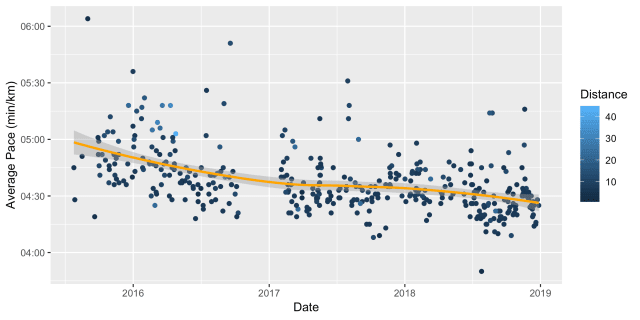
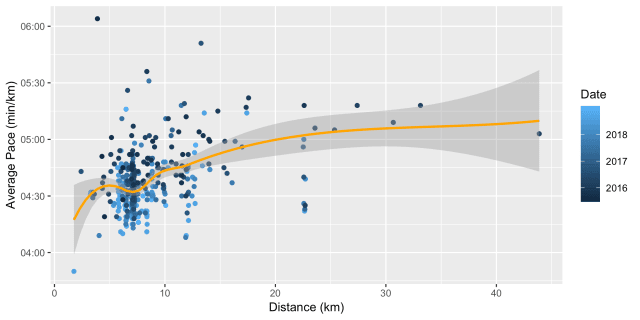
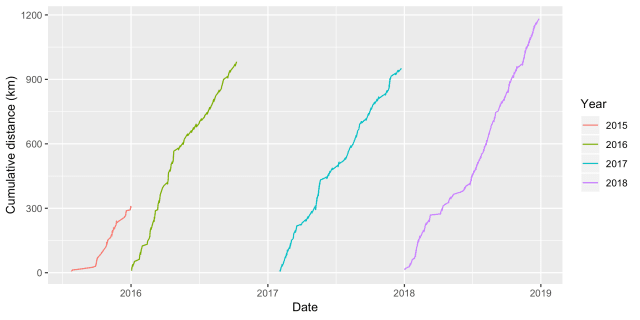
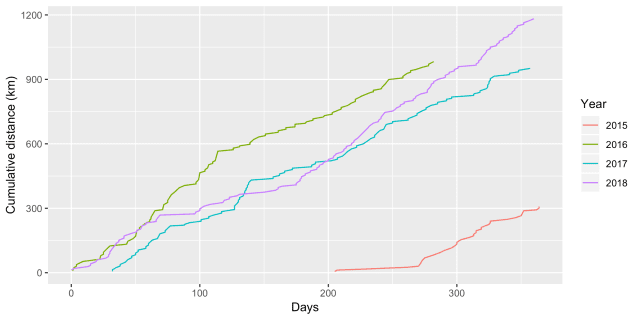
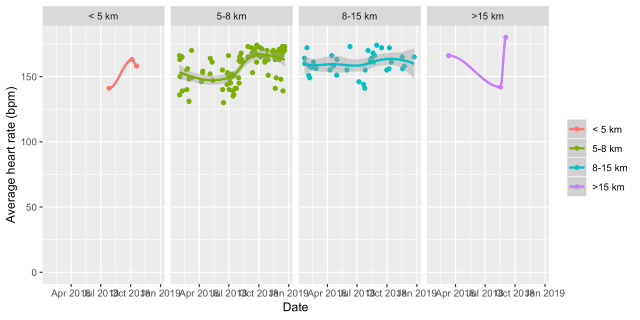
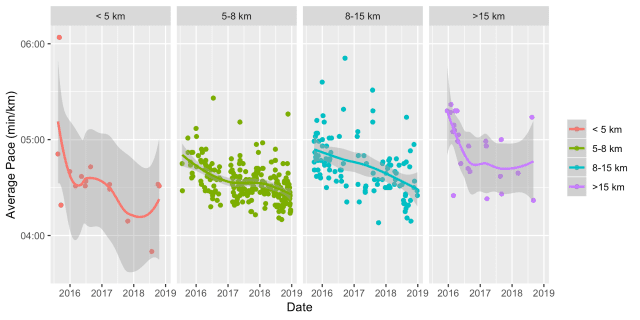
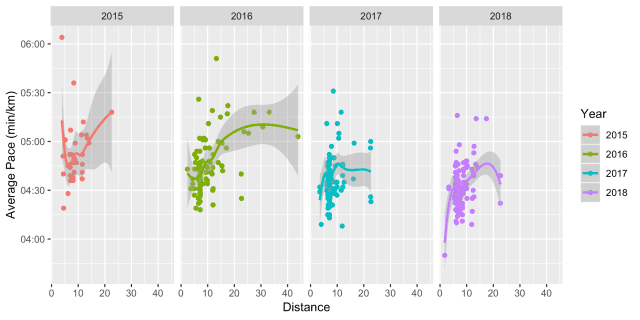
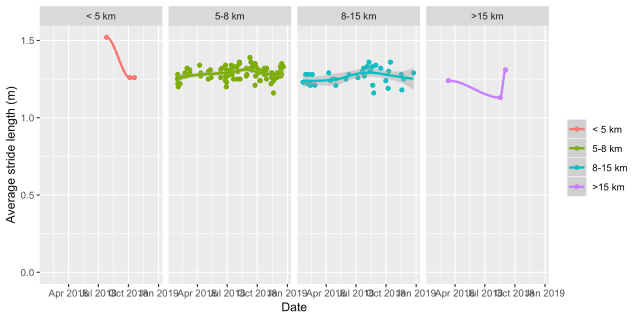
p1 <- ggplot( data = df1, aes(x = Date,y = Avg.Pace, color = Distance)) +

geom\_point() +

scale\_y\_datetime(date\_labels = "%M:%S") +

geom\_smooth(color = "orange") +

labs(x = "Date", y = "Average Pace (min/km)")

* 
* 
* 
* 
* 
* 
* 
* 

The remainder of the code for the other plots is shown below. The code is commented. For some of the plots, a bit of extra work on the data frame is required.

# plot out same data grouped by distance

p2 <- ggplot( data = df1, aes(x = Date,y = Avg.Pace, group = Type\_f, color = Type\_f)) +

geom\_point() +

scale\_y\_datetime(date\_labels = "%M:%S") +

geom\_smooth() +

labs(x = "Date", y = "Average Pace (min/km)", colour = NULL) +

facet\_grid(~Type\_f)

# now look at stride length. first remove zeros

df1[df1 == 0] <- NA

# now find earliest valid date

date\_v <- df1$Date

# change dates to NA where there is no avg stride data

date\_v <- as.Date.POSIXct(ifelse(df1$Avg.Stride.Length > 0, df1$Date, NA))

# find min and max for x-axis

earliest\_date <- min(date\_v, na.rm = TRUE)

latest\_date <- max(date\_v, na.rm = TRUE)

# make the plot

p3 <- ggplot(data = df1, aes(x = Date,y = Avg.Stride.Length, group = Type\_f, color = Type\_f)) +

geom\_point() +

ylim(0, NA) + xlim(as.POSIXct(earliest\_date), as.POSIXct(latest\_date)) +

geom\_smooth() +

labs(x = "Date", y = "Average stride length (m)", colour = NULL) +

facet\_grid(~Type\_f)

df1$Avg.HR <- as.numeric(as.character(df1$Avg.HR))

p4 <- ggplot(data = df1, aes(x = Date,y = Avg.HR, group = Type\_f, color = Type\_f)) +

geom\_point() +

ylim(0, NA) + xlim(as.POSIXct(earliest\_date), as.POSIXct(latest\_date)) +

geom\_smooth() +

labs(x = "Date", y = "Average heart rate (bpm)", colour = NULL) +

facet\_grid(~Type\_f)

# plot out average pace per distance coloured by year

p5 <- ggplot( data = df1, aes(x = Distance,y = Avg.Pace, color = Date)) +

geom\_point() +

scale\_y\_datetime(date\_labels = "%M:%S") +

geom\_smooth(color = "orange") +

labs(x = "Distance (km)", y = "Average Pace (min/km)")

# make a date factor for year to group the plots

df1$Year <- format(as.Date(df1$Date, format="%d/%m/%Y"),"%Y")

p6 <- ggplot( data = df1, aes(x = Distance,y = Avg.Pace, group = Year, color = Year)) +

geom\_point() +

scale\_y\_datetime(date\_labels = "%M:%S") +

geom\_smooth() +

labs(x = "Distance", y = "Average Pace (min/km)") +

facet\_grid(~Year)

# Cumulative sum over years

df1 <- df1[order(as.Date(df1$Date)),]

df1 <- df1 %>% group\_by(Year) %>% mutate(cumsum = cumsum(Distance))

p7 <- ggplot( data = df1, aes(x = Date,y = cumsum, group = Year, color = Year)) +

geom\_line() +

labs(x = "Date", y = "Cumulative distance (km)")

# Plot these cumulative sums overlaid

# Find New Year's Day for each and then work out how many days have elapsed since

df1$nyd <- paste(df1$Year,"-01-01",sep = "")

df1$Days <- as.Date(df1$Date, format="%Y-%m-%d") - as.Date(as.character(df1$nyd), format="%Y-%m-%d")

# Make the plot

p8 <- ggplot( data = df1, aes(x = Days,y = cumsum, group = Year, color = Year)) +

geom\_line() +

scale\_x\_continuous() +

labs(x = "Days", y = "Cumulative distance (km)")

Finally, we can save all of the plots using ggsave.

# save all plots

ggsave("allPace.png", plot = p1, width = 8, height = 4, dpi = "print")

ggsave("paceByDist.png", plot = p2, width = 8, height = 4, dpi = "print")

ggsave("strideByDist.png", plot = p3, width = 8, height = 4, dpi = "print")

ggsave("HRByDist.png", plot = p4, width = 8, height = 4, dpi = "print")

ggsave("allPaceByDist.png", plot = p5, width = 8, height = 4, dpi = "print")

ggsave("paceByDistByYear.png", plot = p6, width = 8, height = 4, dpi = "print")

ggsave("cumulativeDistByYear.png", plot = p7, width = 8, height = 4, dpi = "print")

ggsave("cumulativeDistOverlay.png", plot = p8, width = 8, height = 4, dpi = "print")

I think the code might fail if you don’t record all of the fields that I do. For example if heart rate data is missing or stride length is not recorded, I’m not sure what the code will do. The aim here is to give an idea of what sorts of plots can be generated just using the summary data in the CSV provided by Garmin. Feel free to make suggestions in the comments below.

—

The post title comes from “Garmonbozia” by Superdrag from the Regretfully Yours album. Apparently Garmonbozia is something eaten by the demons in the Black Lodge in the TV series Twin Peaks.

The code snippet are below:

CSV2r.R

|  |
| --- |
| ## The aim of this script is to load and process CSV data from the Garmin Connect website. |
|  | ## It will analyse various aspects of running performance over time. The focus is year-on-year comparisons. |
|  | ## This script will load all csv files in Data/ (in current wd) and filter for Running (and Treadmill Running) |
|  | ## Place one or moe Garmin CSV outputs into the Data folder for inclusion. Dates for activities can be overlapping |
|  | ## duplicates are dealt with, so you can just keep adding csvs with the latest data and use the script again. |
|  |  |
|  | require(ggplot2) |
|  | require(dplyr) |
|  | 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 |
|  |  |
|  | load\_garmin\_data <- function(activity) { |
|  | 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), ] |
|  | # filter for activity |
|  | df\_all <- subset(df\_all,grepl(tolower(activity),tolower(df\_all$Activity.Type))) |
|  |  |
|  | return(df\_all) |
|  | } |
|  |  |
|  | # main script |
|  |  |
|  | df1 <- load\_garmin\_data("running") |
|  | # format Date column to POSIXct |
|  | df1$Date <- as.POSIXct(strptime(df1$Date, format = "%Y-%m-%d %H:%M:%S")) |
|  | # format Avg.Pace to POSIXct |
|  | df1$Avg.Pace <- as.POSIXct(strptime(df1$Avg.Pace, format = "%M:%S")) |
|  | # make groups of different distances using ifelse |
|  | df1$Type <- ifelse(df1$Distance < 5, "< 5 km", ifelse(df1$Distance < 8, "5-8 km", ifelse(df1$Distance < 15, "8-15 km", ">15 km"))) |
|  | # make factors for these so that they're in the right order when we make the plot |
|  | df1$Type\_f = factor(df1$Type, levels=c("< 5 km","5-8 km","8-15 km", ">15 km")) |
|  | # plot out average pace over time |
|  | p1 <- ggplot( data = df1, aes(x = Date,y = Avg.Pace, color = Distance)) + |
|  | geom\_point() + |
|  | scale\_y\_datetime(date\_labels = "%M:%S") + |
|  | geom\_smooth(color = "orange") + |
|  | labs(x = "Date", y = "Average Pace (min/km)") |
|  | p1 |
|  | # plot out same data grouped by distance |
|  | p2 <- ggplot( data = df1, aes(x = Date,y = Avg.Pace, group = Type\_f, color = Type\_f)) + |
|  | geom\_point() + |
|  | scale\_y\_datetime(date\_labels = "%M:%S") + |
|  | geom\_smooth() + |
|  | labs(x = "Date", y = "Average Pace (min/km)", colour = NULL) + |
|  | facet\_grid(~Type\_f) |
|  | p2 |
|  | # now look at stride length. first remove zeros |
|  | df1[df1 == 0] <- NA |
|  | # now find earliest valid date |
|  | date\_v <- df1$Date |
|  | # change dates to NA where there is no avg stride data |
|  | date\_v <- as.Date.POSIXct(ifelse(df1$Avg.Stride.Length > 0, df1$Date, NA)) |
|  | # find min and max for x-axis |
|  | earliest\_date <- min(date\_v, na.rm = TRUE) |
|  | latest\_date <- max(date\_v, na.rm = TRUE) |
|  | # make the plot |
|  | p3 <- ggplot(data = df1, aes(x = Date,y = Avg.Stride.Length, group = Type\_f, color = Type\_f)) + |
|  | geom\_point() + |
|  | ylim(0, NA) + xlim(as.POSIXct(earliest\_date), as.POSIXct(latest\_date)) + |
|  | geom\_smooth() + |
|  | labs(x = "Date", y = "Average stride length (m)", colour = NULL) + |
|  | facet\_grid(~Type\_f) |
|  | p3 |
|  | df1$Avg.HR <- as.numeric(as.character(df1$Avg.HR)) |
|  | p4 <- ggplot(data = df1, aes(x = Date,y = Avg.HR, group = Type\_f, color = Type\_f)) + |
|  | geom\_point() + |
|  | ylim(0, NA) + xlim(as.POSIXct(earliest\_date), as.POSIXct(latest\_date)) + |
|  | geom\_smooth() + |
|  | labs(x = "Date", y = "Average heart rate (bpm)", colour = NULL) + |
|  | facet\_grid(~Type\_f) |
|  | p4 |
|  | # plot out average pace per distance coloured by year |
|  | p5 <- ggplot( data = df1, aes(x = Distance,y = Avg.Pace, color = Date)) + |
|  | geom\_point() + |
|  | scale\_y\_datetime(date\_labels = "%M:%S") + |
|  | geom\_smooth(color = "orange") + |
|  | labs(x = "Distance (km)", y = "Average Pace (min/km)") |
|  | p5 |
|  | # make a date factor for year to group the plots |
|  | df1$Year <- format(as.Date(df1$Date, format="%d/%m/%Y"),"%Y") |
|  | p6 <- ggplot( data = df1, aes(x = Distance,y = Avg.Pace, group = Year, color = Year)) + |
|  | geom\_point() + |
|  | scale\_y\_datetime(date\_labels = "%M:%S") + |
|  | geom\_smooth() + |
|  | labs(x = "Distance", y = "Average Pace (min/km)") + |
|  | facet\_grid(~Year) |
|  | p6 |
|  | # Cumulative sum over years |
|  | df1 <- df1[order(as.Date(df1$Date)),] |
|  | df1 <- df1 %>% group\_by(Year) %>% mutate(cumsum = cumsum(Distance)) |
|  | p7 <- ggplot( data = df1, aes(x = Date,y = cumsum, group = Year, color = Year)) + |
|  | geom\_line() + |
|  | labs(x = "Date", y = "Cumulative distance (km)") |
|  | p7 |
|  | # Plot these cumulative sums overlaid |
|  | # Find New Year's Day for each and then work out how many days have elapsed since |
|  | df1$nyd <- paste(df1$Year,"-01-01",sep = "") |
|  | df1$Days <- as.Date(df1$Date, format="%Y-%m-%d") - as.Date(as.character(df1$nyd), format="%Y-%m-%d") |
|  | # Make the plot |
|  | p8 <- ggplot( data = df1, aes(x = Days,y = cumsum, group = Year, color = Year)) + |
|  | geom\_line() + |
|  | scale\_x\_continuous() + |
|  | labs(x = "Days", y = "Cumulative distance (km)") |
|  | p8 |
|  |  |
|  | # save all plots |
|  | ggsave("Output/Plots/allPace.png", plot = p1, width = 8, height = 4, dpi = "print") |
|  | ggsave("Output/Plots/paceByDist.png", plot = p2, width = 8, height = 4, dpi = "print") |
|  | ggsave("Output/Plots/strideByDist.png", plot = p3, width = 8, height = 4, dpi = "print") |
|  | ggsave("Output/Plots/HRByDist.png", plot = p4, width = 8, height = 4, dpi = "print") |
|  | ggsave("Output/Plots/allPaceByDist.png", plot = p5, width = 8, height = 4, dpi = "print") |
|  | ggsave("Output/Plots/paceByDistByYear.png", plot = p6, width = 8, height = 4, dpi = "print") |
|  | ggsave("Output/Plots/cumulativeDistByYear.png", plot = p7, width = 8, height = 4, dpi = "print") |
|  | ggsave("Output/Plots/cumulativeDistOverlay.png", plot = p8, width = 8, height = 4, dpi = "print") |

ReadAndCalC.R

|  |
| --- |
| ## The aim of this script is to load and process CSV data from the Garmin Connect website. |
|  | ## It will check whether you are on track to meet a running goal, i.e. n km between date1 and date2. |
|  | ## This script will load all csv files in Data/ (in current wd) and filter for Running (and Treadmill Running) |
|  | ## Place one or more Garmin CSV outputs into the Data folder for inclusion. Dates for activities can be overlapping |
|  | ## duplicates are dealt with, so you can just keep adding csvs with the latest data and use the script again. |
|  |  |
|  |  |
|  | 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$Activity.Type))) |
|  | # activities within the window |
|  | fromDate <- as.Date(fromStr) |
|  | toDate <- as.Date(toStr) |
|  | df\_window <- subset(df\_window, as.Date(df\_window$Date) >= fromDate & as.Date(df\_window$Date) <= toDate) |
|  | # put them in order |
|  | df\_window <- df\_window[order(as.numeric(df\_window$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) |
|  | } |
|  |  |
|  | process\_data <- function(activityStr,fromStr,toStr,km) { |
|  | all\_files <- list.files("Data", pattern = "\*.csv", full.names = TRUE) |
|  | df\_all <- read.csv(all\_files[1], header = TRUE, stringsAsFactors=FALSE) |
|  | df\_all <- subset(df\_all, select = c(Activity.Type,Date,Title,Distance,Time)) |
|  | for (filename in all\_files[-1]) { |
|  | df\_temp <- read.csv(filename, stringsAsFactors=FALSE) |
|  | # subset data because Garmin can add or remove columns and we don't need them all |
|  | df\_temp <- subset(df\_temp, select = c(Activity.Type,Date,Title,Distance,Time)) |
|  | 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(activityStr,fromStr,toStr,df\_all) |
|  | df\_target <- maketarget(fromStr,toStr,km) |
|  | # wrangle data frames to have matching date columns and then merge, then find difference |
|  | # between the cumulative distance and the target |
|  | df\_temp <- data.frame(Date = as.Date(df\_all$Date), |
|  | Cumulative.Distance = df\_all$Cumulative.Distance) |
|  | # add today as final row if we are within the window |
|  | if(as.Date(toStr) >= Sys.Date()) { |
|  | df\_temp[nrow(df\_temp) + 1,1] = Sys.Date() |
|  | df\_temp[nrow(df\_temp),2] = df\_temp[nrow(df\_temp) - 1,2] |
|  | } |
|  | 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$Cumulative.Distance.y |
|  |  |
|  | # make dataframe to show more granular "balance" of km |
|  | df\_debit <- data.frame(Date = df\_target$Date, |
|  | Distance = -(km / nrow(df\_target))) |
|  | df\_credit <- data.frame(Date = df\_all$Date, |
|  | Distance = df\_all$Distance) |
|  | df\_balance <- rbind(df\_debit,df\_credit) |
|  | df\_balance <- df\_balance[order(as.numeric(df\_balance$Date)),] |
|  | df\_balance <- subset(df\_balance, as.Date(df\_balance$Date) <= as.Date(Sys.Date())) |
|  | df\_balance$Cumulative.Distance <- cumsum(df\_balance$Distance) |
|  | df\_balance$Date <- as.Date(df\_balance$Date, format = "%Y-%m-%d %H:%M:%S") |
|  |  |
|  | # save data |
|  | write.table(df\_all, file = paste0("Output/Data/alldata\_",fromStr,"\_",toStr,".txt"), sep="\t", row.names=FALSE, col.name=TRUE) |
|  | write.table(df\_merge, file = paste0("Output/Data/mergedata\_",fromStr,"\_",toStr,".txt"), sep="\t", row.names=FALSE, col.name=TRUE) |
|  | write.table(df\_target, file = paste0("Output/Data/targetdata\_",fromStr,"\_",toStr,".txt"), sep="\t", row.names=FALSE, col.name=TRUE) |
|  | write.table(df\_balance, file = paste0("Output/Data/balancedata\_",fromStr,"\_",toStr,".txt"), sep="\t", row.names=FALSE, col.name=TRUE) |
|  |  |
|  | # 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) + |
|  | geom\_line(data = df\_target, linetype = 2) + |
|  | labs(x = "Date", y = "Cumulative Distance (km)") |
|  | # 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) + |
|  | geom\_hline(yintercept = 0, linetype = 2) + |
|  | xlim(as.Date(fromStr), as.Date(toStr)) + |
|  | ylim(-max(abs(df\_merge$Difference)),max(abs(df\_merge$Difference))) + |
|  | labs(x = "Date", y = "Difference (km)") |
|  | # more accurate "balance" graph |
|  | p3 <- ggplot(data = df\_balance, aes(x = Date, y = Cumulative.Distance)) + |
|  | geom\_line(colour = "blue", size = 1) + |
|  | geom\_hline(yintercept = 0, linetype = 2) + |
|  | xlim(as.Date(paste0(fromStr," 00:00:00"), format = "%Y-%m-%d %H:%M:%S"), as.Date(paste0(toStr," 23:59:59"), format = "%Y-%m-%d %H:%M:%S")) + |
|  | ylim(-max(abs(df\_balance$Cumulative.Distance)),max(abs(df\_balance$Cumulative.Distance))) + |
|  | labs(x = "Date", y = "Balance (km)") |
|  |  |
|  | # save all plots |
|  | ggsave(paste0("Output/Plots/progress\_",fromStr,"\_",toStr,".png"), plot = p1, width = 8, height = 4, dpi = "print") |
|  | ggsave(paste0("Output/Plots/difference\_",fromStr,"\_",toStr,".png"), plot = p2, width = 8, height = 4, dpi = "print") |
|  | ggsave(paste0("Output/Plots/balance\_",fromStr,"\_",toStr,".png"), plot = p3, width = 8, height = 4, dpi = "print") |
|  |  |
|  | # report distances |
|  | print(paste0("Last ",activityStr," activity on: ", toString(df\_all[nrow(df\_all),2]),". Today is ", toString(Sys.Date()))) |
|  | print(paste0("Total ",activityStr," distance between ", fromStr," and ",toStr, " is ", toString(df\_merge[nrow(df\_merge),2])," km. Goal is ", km, " km.")) |
|  | if(df\_merge[nrow(df\_merge),4] < 0) { |
|  | print(paste0(toString(df\_merge[nrow(df\_merge),4] \* -1)," km behind target.")) |
|  | } else { |
|  | print(paste0(toString(df\_merge[nrow(df\_merge),4])," km ahead of target.")) |
|  | } |
|  | if(km > df\_merge[nrow(df\_merge),2]) { |
|  | print(paste0(toString(km - df\_merge[nrow(df\_merge),2]), " km to go!")) |
|  | } else { |
|  | print("You did it!") |
|  | } |
|  | print(paste0("In this period you have run ",sum(floor(df\_all$Distance / 21.1))," half-marathons.")) |
|  | } |
|  |  |
|  | # to process the data |
|  | # Garmin 2021 Running - Stage 1 |
|  | process\_data("running","2021-01-01","2021-03-31",505) |
|  | # Garmin 2021 Running - Stage 2 |
|  | process\_data("running","2021-04-01","2021-06-30",505) |
|  | # Garmin 2021 Running - Stage 3 |
|  | process\_data("running","2021-07-01","2021-09-30",505) |
|  | # Garmin 2021 Running - Stage 4 |
|  | process\_data("running","2021-10-01","2021-12-31",506) |
|  |  |
|  | # Overall |
|  | process\_data("running","2021-01-01","2021-12-31",2021) |