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 fake 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

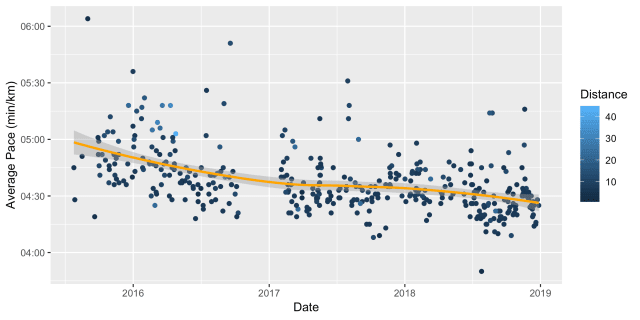
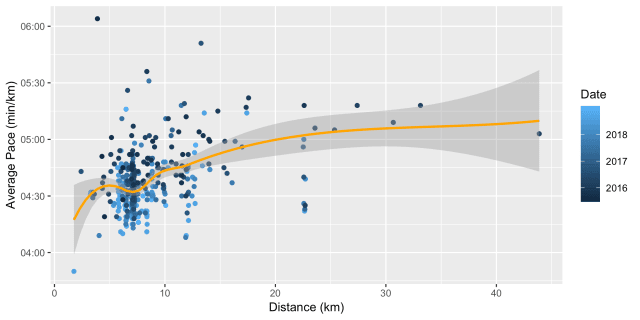
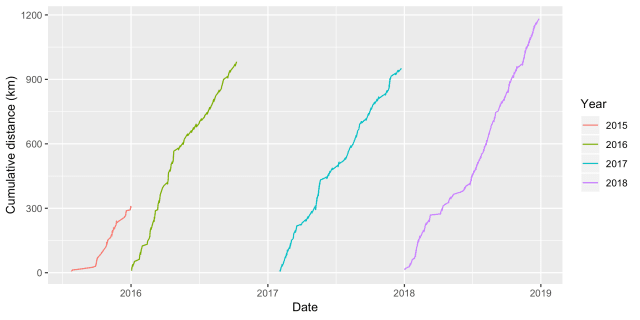
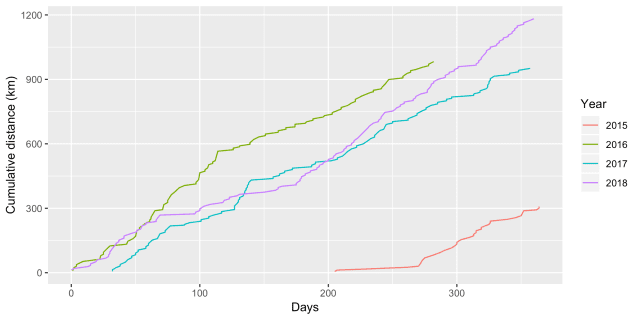
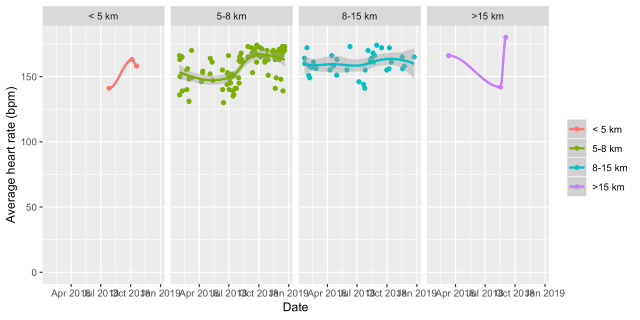
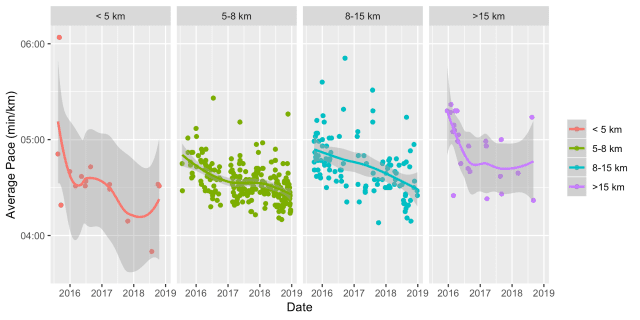
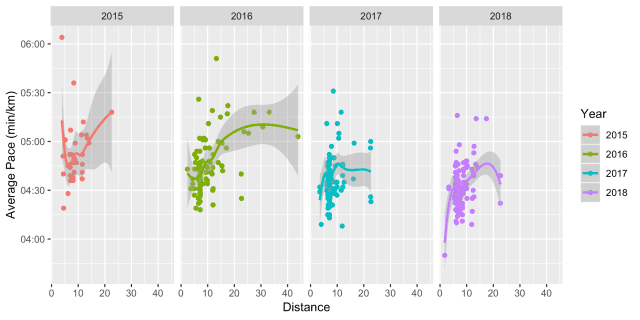
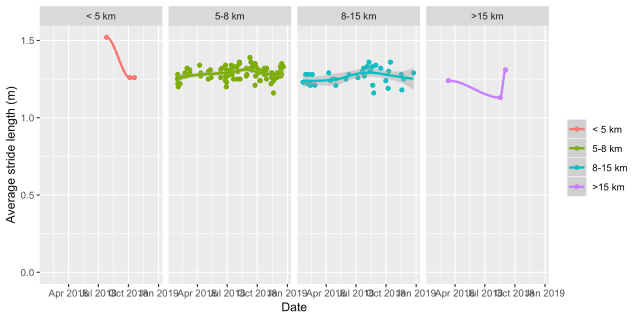
p1 <- ggplot( data = df1, aes(x = Date,y = Avg.Pace, color = Distance)) +

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.