July 2nd try 2

Chad Mirara

2025-07-12

R Markdown

This is an R Markdown document. Markdown is a simple formatting syntax for authoring HTML, PDF, and MS Word documents. For more details on using R Markdown see http://rmarkdown.rstudio.com.

When you click the **Knit** button a document will be generated that includes both content as well as the output of any embedded R code chunks within the document. You can embed an R code chunk like this:

```
setwd("~/Library/Mobile Documents/com~apple~CloudDocs/Desktop/Personal/Chad Mirara Technology Projects")

# Load required packages for mile split analysis
library(sf)
```

```
## Linking to GEOS 3.13.0, GDAL 3.8.5, PROJ 9.5.1; sf_use_s2() is TRUE
```

```
library(dplyr)
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

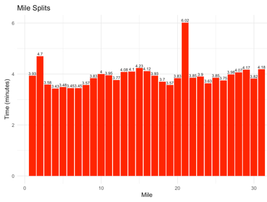
```
library(geosphere)
library(ggplot2)

# Import the GPX file
ride_data <- sf::st_read("Morning_Ride July 2nd 2025.gpx", layer="track_points")
```

```
## Reading layer `track_points' from data source
##   `/Users/chadmirara/Library/Mobile Documents/com~apple~CloudDocs/Desktop/Personal/Chad Mirara Technology Proj
ects/Morning_Ride July 2nd 2025.gpx'
##   using driver `GPX'
## Simple feature collection with 7412 features and 26 fields
## Geometry type: POINT
## Dimension:     XY
## Bounding box:  xmin: -83.00938 ymin: 40.15543 xmax: -82.97474 ymax: 40.2002
## Geodetic CRS:  WGS 84
```

```
# Convert time to POSIXct
ride_data$time <- as.POSIXct(ride_data$time)

# Extract coordinates
coords <- st_coordinates(ride_data$geometry)
ride_data$lon <- coords[,1]
ride_data$lat <- coords[,2]

# Calculate distance between consecutive points (in feet)
calculate_distances <- function(lon, lat) {
  n <- length(lon)
  dists <- numeric(n-1)

  for(i in 1:(n-1)) {
    dists[i] <- distHaversine(
      c(lon[i], lat[i]),
      c(lon[i+1], lat[i+1])
    )
  }

  return(c(0, dists))  # First point has 0 distance
}

ride_data$segment_dist <- calculate_distances(ride_data$lon, ride_data$lat)

# Calculate cumulative distance
ride_data$cum_dist <- cumsum(ride_data$segment_dist)

# Convert to miles
ride_data$cum_dist_miles <- ride_data$cum_dist / 1609.34

# Create mile splits
miles <- floor(max(ride_data$cum_dist_miles))
mile_splits <- data.frame(
  mile = 1:miles,
  split_time = numeric(miles),
  avg_hr = numeric(miles),
  elevation_gain = numeric(miles)
)

for(i in 1:miles) {
  # Find points in this mile
  mile_start <- ride_data[which(ride_data$cum_dist_miles >= (i-1) &
                                ride_data$cum_dist_miles < i), ]
  mile_end <- ride_data[which(ride_data$cum_dist_miles >= i)[1], ]

  # Calculate time for this mile (in minutes)
  time_diff <- as.numeric(difftime(mile_end$time[1], mile_start$time[1], units="mins"))
  mile_splits$split_time[i] <- time_diff


  ##--------


  # Calculate average heart rate for this mile
  mile_data <- ride_data[which(ride_data$cum_dist_miles >= (i-1) &
                               ride_data$cum_dist_miles < i), ]
  mile_splits$avg_hr[i] <- mean(as.numeric(mile_data$heart_rate), na.rm=TRUE)

  # Calculate elevation gain for this mile
  ele_diffs <- diff(as.numeric(mile_data$ele))
  mile_splits$elevation_gain[i] <- sum(pmax(0, ele_diffs))
}

# Create a pace column (minutes per mile)
mile_splits$pace <- mile_splits$split_time

# Display mile splits table
print(mile_splits)
```

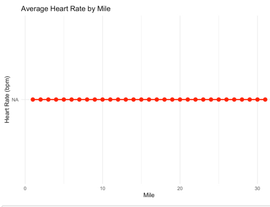
```
##    mile split_time avg_hr elevation_gain     pace
## 1     1   3.933333    NaN          17.6 3.933333
## 2     2   4.700000    NaN           7.3 4.700000
## 3     3   3.583333    NaN           8.9 3.583333
## 4     4   3.433333    NaN           4.5 3.433333
## 5     5   3.483333    NaN           4.5 3.483333
## 6     6   3.450000    NaN           3.9 3.450000
## 7     7   3.450000    NaN           4.2 3.450000
## 8     8   3.566667    NaN           2.4 3.566667
## 9     9   3.833333    NaN           6.5 3.833333
## 10   10   4.000000    NaN           2.3 4.000000
## 11   11   3.950000    NaN           4.9 3.950000
## 12   12   3.766667    NaN           5.4 3.766667
## 13   13   4.083333    NaN           4.6 4.083333
## 14   14   4.100000    NaN           4.4 4.100000
## 15   15   4.233333    NaN           7.8 4.233333
## 16   16   4.116667    NaN           7.7 4.116667
## 17   17   3.933333    NaN           5.6 3.933333
## 18   18   3.700000    NaN           8.2 3.700000
## 19   19   3.566667    NaN           7.5 3.566667
## 20   20   3.833333    NaN           5.7 3.833333
## 21   21   6.016667    NaN           1.3 6.016667
## 22   22   3.850000    NaN           4.6 3.850000
## 23   23   3.900000    NaN           4.8 3.900000
## 24   24   3.633333    NaN           5.1 3.633333
## 25   25   3.850000    NaN           7.6 3.850000
## 26   26   3.750000    NaN           1.9 3.750000
## 27   27   3.983333    NaN           4.9 3.983333
## 28   28   4.066667    NaN           4.7 4.066667
## 29   29   4.166667    NaN           9.0 4.166667
## 30   30   3.816667    NaN           3.9 3.816667
## 31   31   4.183333    NaN           2.9 4.183333
```

```
mile_splits <- mile_splits %>%
  mutate(
    split_time = ifelse(is.finite(split_time), split_time, NA),
    avg_hr = ifelse(is.finite(avg_hr), avg_hr, NA),
    elevation_gain = ifelse(is.finite(elevation_gain), elevation_gain, NA),
    pace = ifelse(is.finite(pace), pace, NA)
  )




# Plot mile splits
ggplot(mile_splits, aes(x=mile, y=split_time)) +
  geom_bar(stat="identity", fill="red", na.rm=TRUE) +
  geom_text(aes(label=round(split_time, 2)), vjust=-0.3, size=2.5, na.rm=TRUE) +
  labs(title="Mile Splits", x="Mile", y="Time (minutes)") +
  theme_minimal()
```

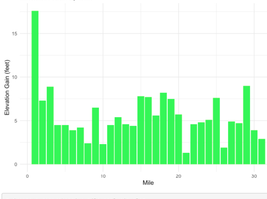



```
# Plot average heart rate by mile
ggplot(mile_splits, aes(x=mile, y=avg_hr)) +
  geom_line(color="red", size=1) +
  geom_point(color="red", size=3) +
  labs(title="Average Heart Rate by Mile", x="Mile", y="Heart Rate (bpm)") +
  theme_minimal()
```

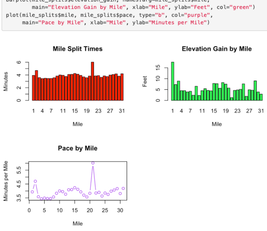
```
## Warning: Using `size` aesthetic for lines was deprecated in ggplot2 3.4.0.
## ℹ Please use `linewidth` instead.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```




```
# Plot elevation gain by mile
ggplot(mile_splits, aes(x=mile, y=elevation_gain)) +
  geom_bar(stat="identity", fill="green") +
  labs(title="Elevation Gain by Mile", x="Mile", y="Elevation Gain (feet)") +
  theme_minimal()
```




```
# Create a comprehensive mile analysis plot
par(mfrow=c(2,2))
barplot(mile_splits$split_time, names.arg=mile_splits$mile,
        main="Mile Split Times", xlab="Mile", ylab="Minutes", col="red")

barplot(mile_splits$elevation_gain, names.arg=mile_splits$mile,
        main="Elevation Gain by Mile", xlab="Mile", ylab="Feet", col="green")
plot(mile_splits$mile, mile_splits$pace, type="b", col="purple",
     main="Pace by Mile", xlab="Mile", ylab="Minutes per Mile")
```




```
summary(cars)
```

```
##      speed           dist
##  Min.   : 4.0   Min.   :  2.00
##  1st Qu.:12.0   1st Qu.: 26.00
##  Median :15.0   Median : 36.00
##  Mean   :15.4   Mean   : 42.98
##  3rd Qu.:19.0   3rd Qu.: 56.00
##  Max.   :25.0   Max.   :120.00
```

Note that the `echo = FALSE` parameter was added to the code chunk to prevent printing of the R code that generated the plot.