Trip Analyzer

Dave Hurst

Wednesday, December 24, 2014

I think the key to creating a *fingerprint* for a driver is to analyze their patterns in specific driving circumstances. To figure out the important features, I wanted a tool to look at the data as trip, beyond just simple route plots.

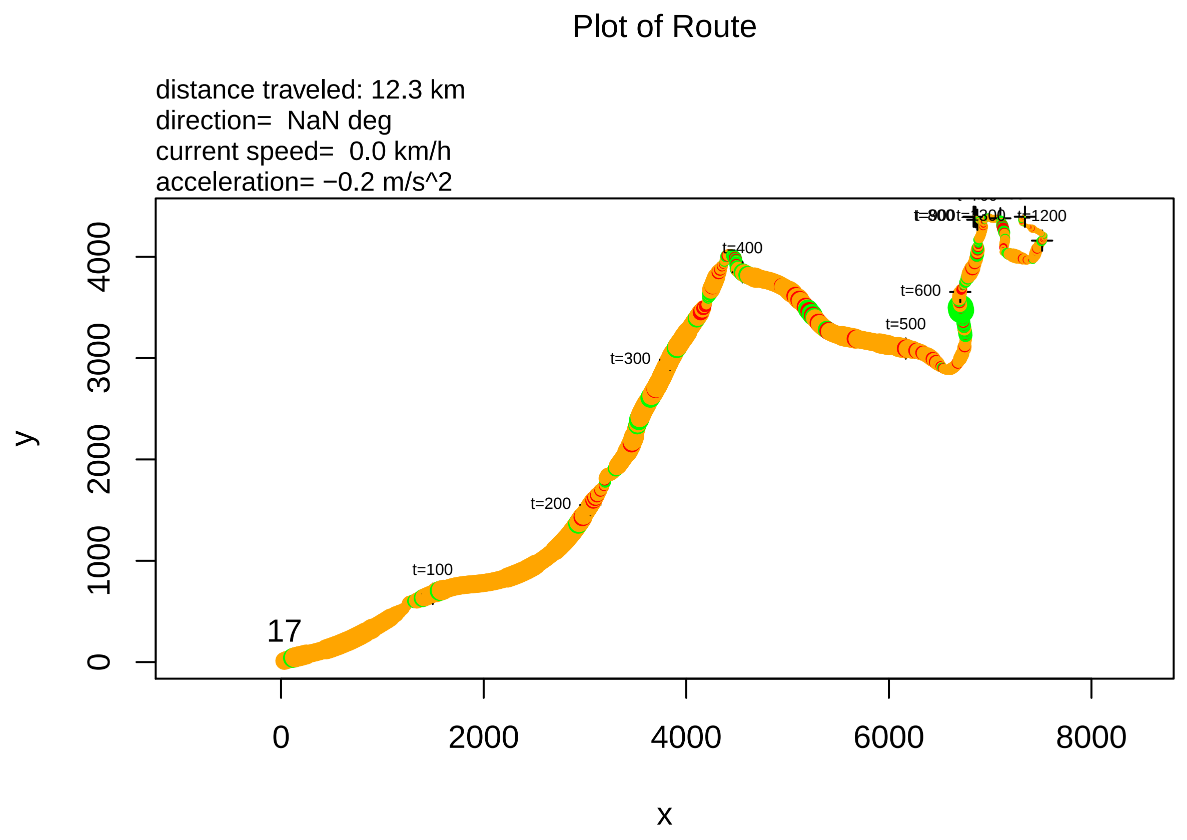
It's easiest to show this with a few examples.

Start by grabbing the data for a random driver:

```r
source("telematic_util.R")

data.dir <- "c:/NO_BACKUP/kaggle/telematics/drivers"

drivers.dir <- dir(data.dir)
set.seed(2349)

random.driver <- order(runif(1:length(drivers.dir), ))

N.DRIVERS <- 1
driver.df <- data.frame(id=integer()
                        , speed.avg=numeric(), speed.max=numeric()
                        , break.max=numeric(), accel.max=numeric())
for (i in 1:N.DRIVERS) {
    driver.dir <- paste(c(data.dir, drivers.dir[random.driver[i]]), collapse="/")
    driver.files <- dir(driver.dir)
    j <- length(driver.files)  #grab the last trip
    #for (j in 1:length(driver.files)) {
        trip <- read.csv( paste(c(driver.dir, driver.files[j] ), collapse="/"))
        trip.last <- trip[-nrow(trip), ]
        trip <- trip[-1, ]
        trip$x.d <- trip$x - trip.last$x
        trip$y.d <- trip$y - trip.last$y
        trip$v <- sqrt( trip$x.d^2 + trip$y.d^2 )  # distance travelled per second
        trip.last <- trip[-nrow(trip), 3:5 ]
        trip <- trip[-1, ]
        trip$x.d2 <- trip$x.d - trip.last$x.d
        trip$y.d2 <- trip$y.d - trip.last$y.d
        trip$a <- trip$v - trip.last$v
    #}
    driver.df[i,"id"] = drivers.dir[random.driver[1]]
    driver.df[i, "speed.avg"] =  mean(trip$v)
    driver.df[i, "speed.max"] =  max(trip$v)
    driver.df[i, "break.max"] =  min(trip$a)
    driver.df[i, "accel.max"] =  max(trip$a)
}
```

Now we've got the 200th trip for driver 2591 in memory. Plot the entire trip:

```r
plotTrip(trip, v.mark=50)
```



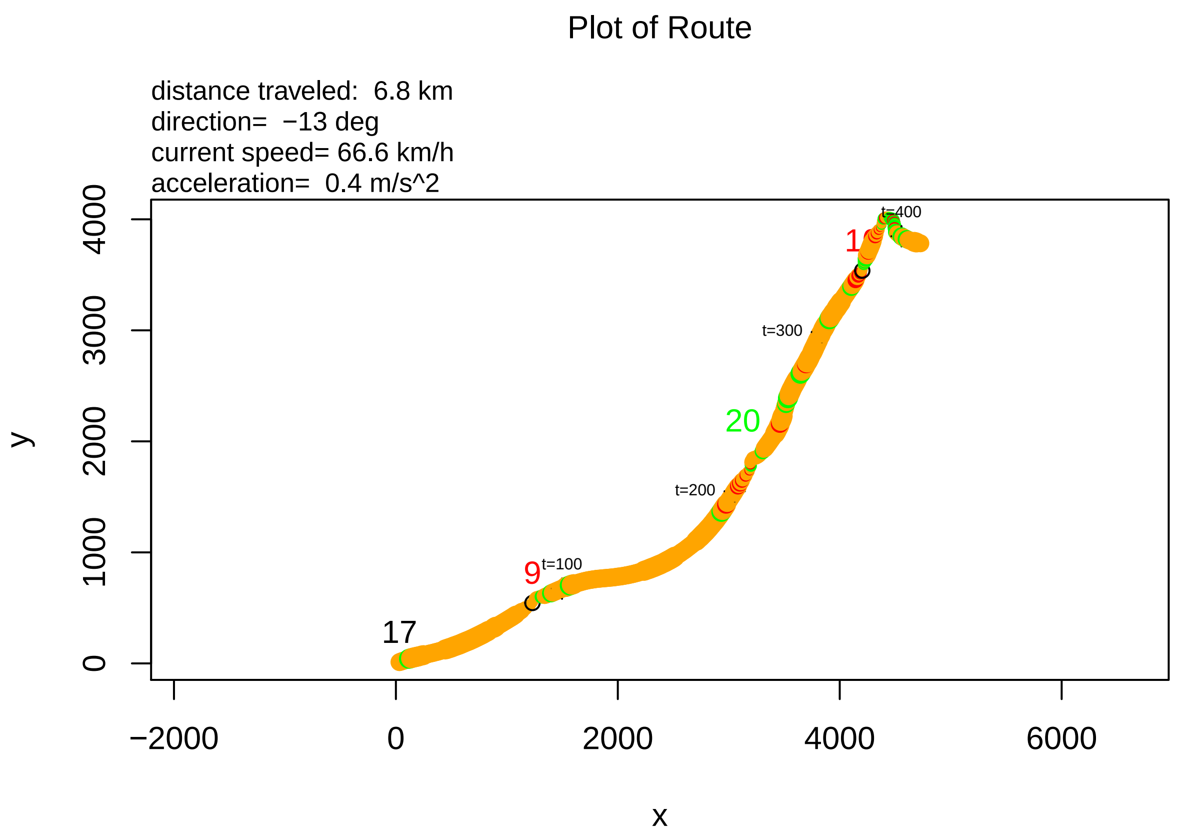
The first number (17) at x=y=0 is the initial speed. Since this isn't 0 in this case, we know that the beginning of the trip has been removed. The small "t=NNN" indicate where the driver is at different time points (in seconds) on the trip, this can be adjusted by calling `plotTrip` with `t.mark=100`.

The thickness of the line is relative to the speed, so as the driver slows down, so does that line. The green segments are segments where the driver is accelerating (beyond a fixed noise value of 1 m/s^2). Conversely, red segments show the driver braking (decelerating > 1 m/s^2).

The `v.mark=50` parameter controls printing of the drivers speed (in m/s) in predetermined thresholds, so setting it to 50 effectively turns it off (except for the first point which is always plotted). It's more useful for trip segments where the range of speed isn't too varied. Plot a segment by specifying `tmin` and `tmax` as shown (note `v.mark=5` by default)

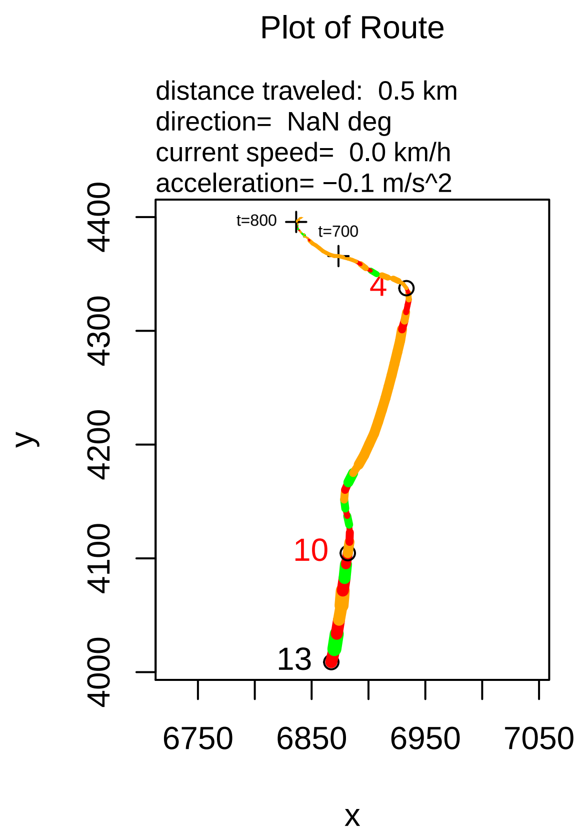
Note that the speed is shown in green when the driver is accelerating, and red when decelerating.

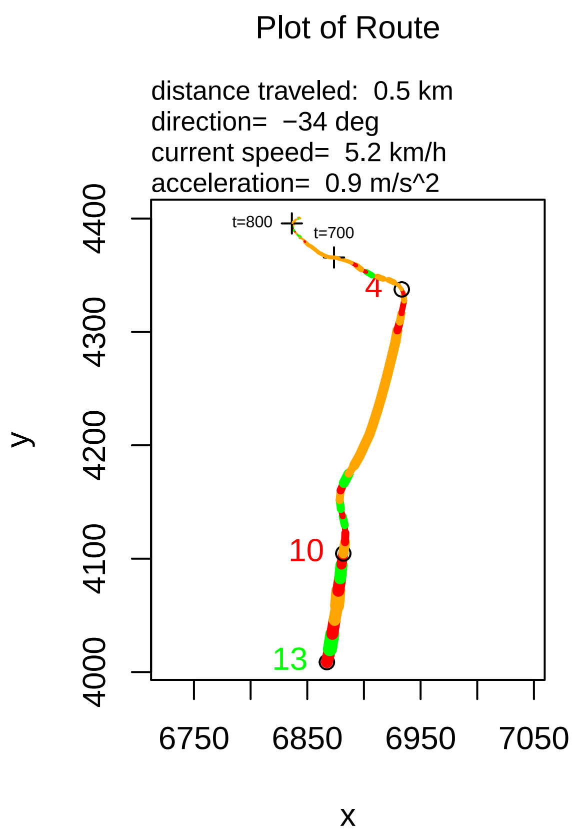
```r
plotTrip(trip, tmax=410, v.mark=10)
```



Putting this all together, you do some quick detail analysis with some simple console commands. For this trip, the interesting stuff is on the back end and you can step through things like this:

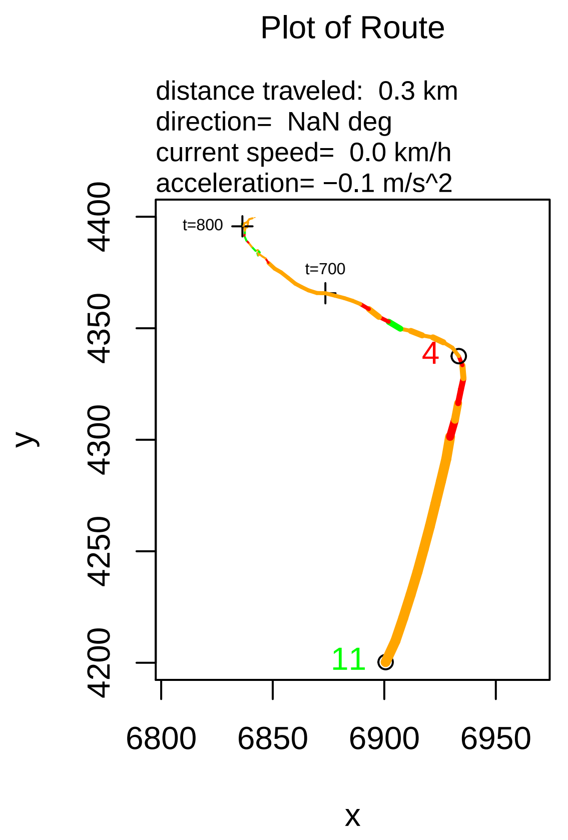
```
par(mfrow=c(1,2))
i = 800 #the time period we want to start with
i=i+20;  plotTrip(trip, tmin=650, tmax=i)
i=i+20;  plotTrip(trip, tmin=650, tmax=i)
```

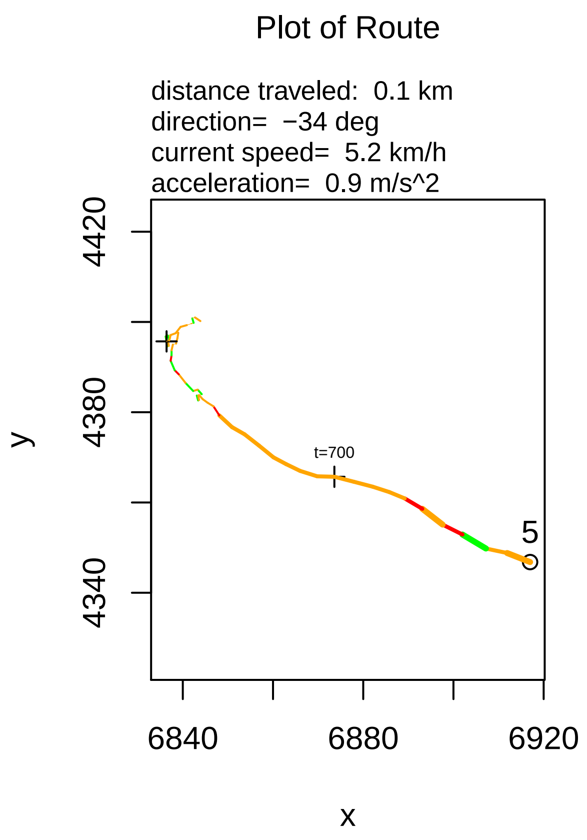




or

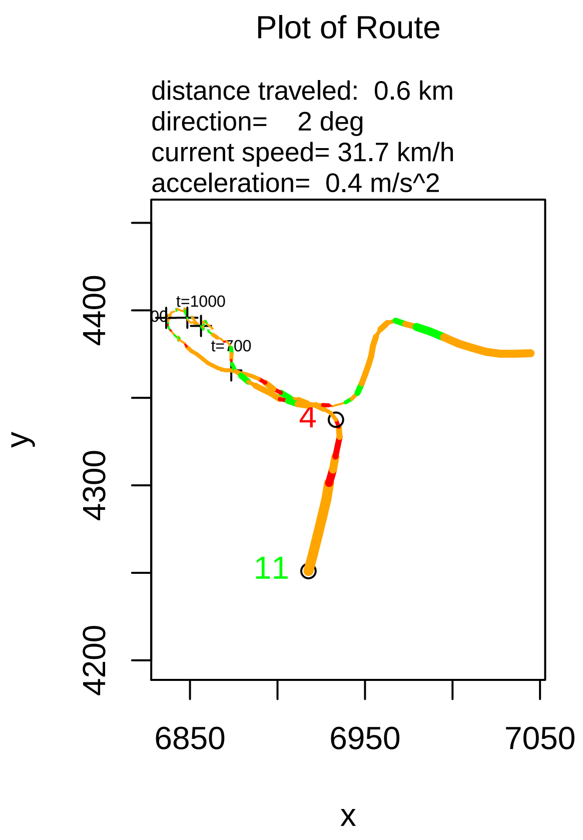
```
par(mfrow=c(1,2))
i = 800 #the time period we want to start with
i=i+20;  plotTrip(trip, tmin=i-150, tmax=i)
i=i+20;  plotTrip(trip, tmin=i-150, tmax=i)
```

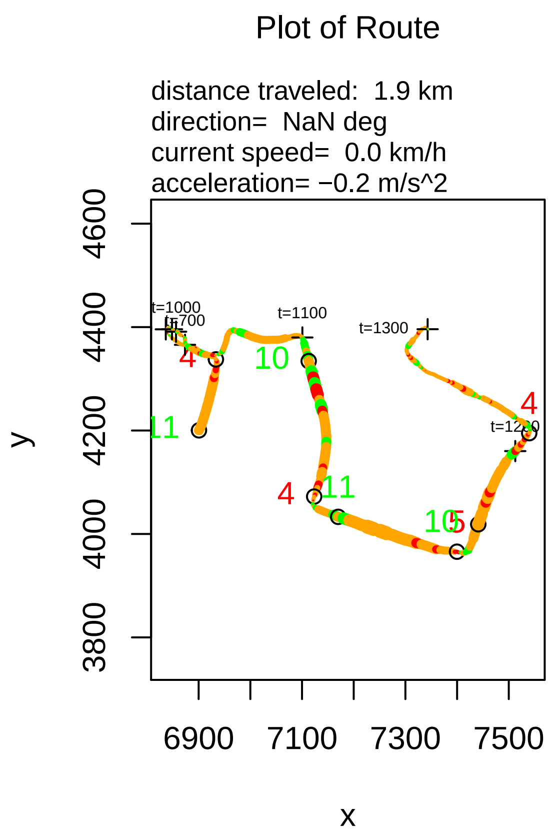




Soo.., putting it all together for this driver, I have to believe he stopped at the McDonalds drivethrough on his way home for work. I wonder if we can build a feature out of that ;-)

```r
par(mfrow=c(1,2))
plotTrip(trip, tmin=675, tmax=1090)
plotTrip(trip, tmin=670)
```





```r
par(mfrow=c(1,1))
```