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:

```
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.id <- drivers.dir[random.driver[i]]
    j <- 99  #grab
    trip <- getTrip( driver.id, j )
    driver.df[i, "id"] = driver.id
    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)
}
```

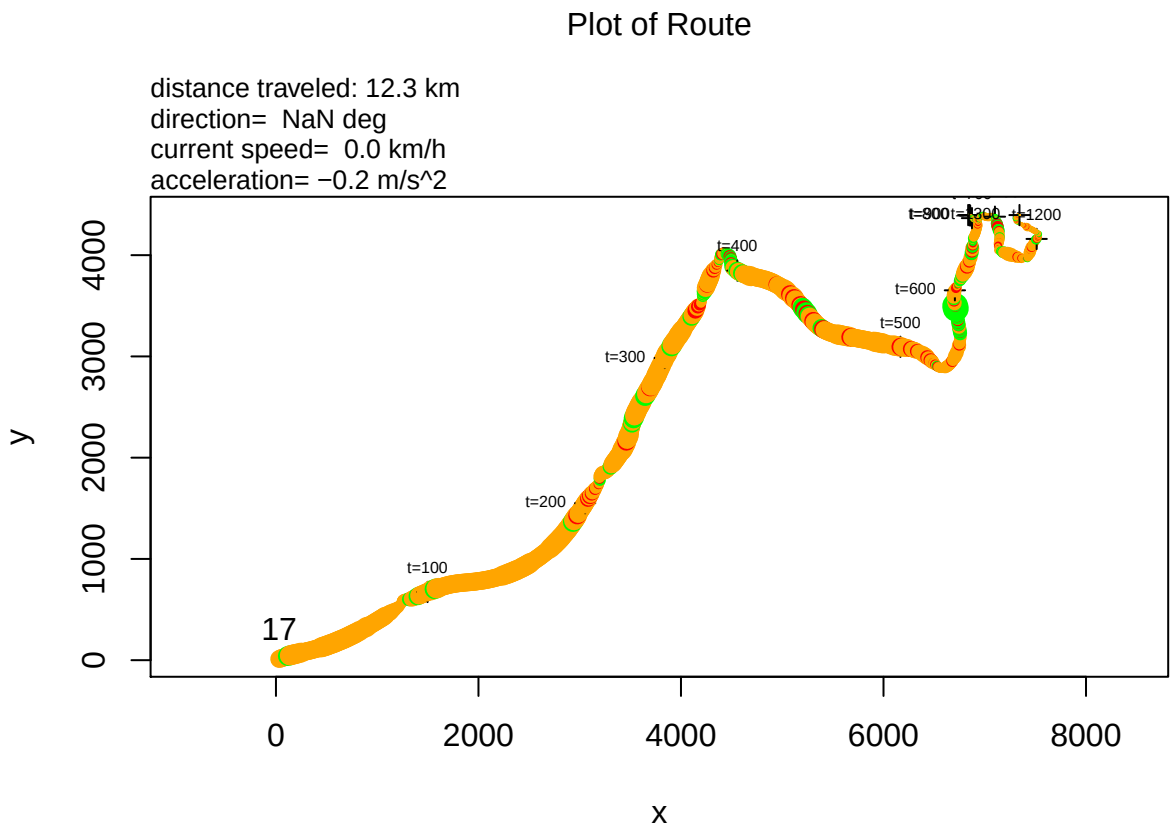
Now we've got the 99th trip for driver 2591 in memory. Here's the data for the first few lines (we are supplied the first 2, x and y, the rest is derived):

```
head(trip)
```

```
##       x    y  x.d y.d        v x.d2          y.d2          a  bearing
## 3  32.4 11.8 16.4 5.9 17.42900  0.4  0.000000e+00  0.3758466 19.78650
## 4  49.0 17.9 16.6 6.1 17.68530  0.2  2.000000e-01  0.2563058 20.17686
## 5  66.1 24.0 17.1 6.1 18.15544  0.5  3.552714e-15  0.4701353 19.63260
## 6  83.4 29.1 17.3 5.1 18.03607  0.2 -1.000000e+00 -0.1193650 16.42540
## 7 100.6 35.2 17.2 6.1 18.24966 -0.1  1.000000e+00  0.2135826 19.52711
## 8 117.9 41.4 17.3 6.2 18.37743  0.1  1.000000e-01  0.1277743 19.71674
```

Plot the entire trip:

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



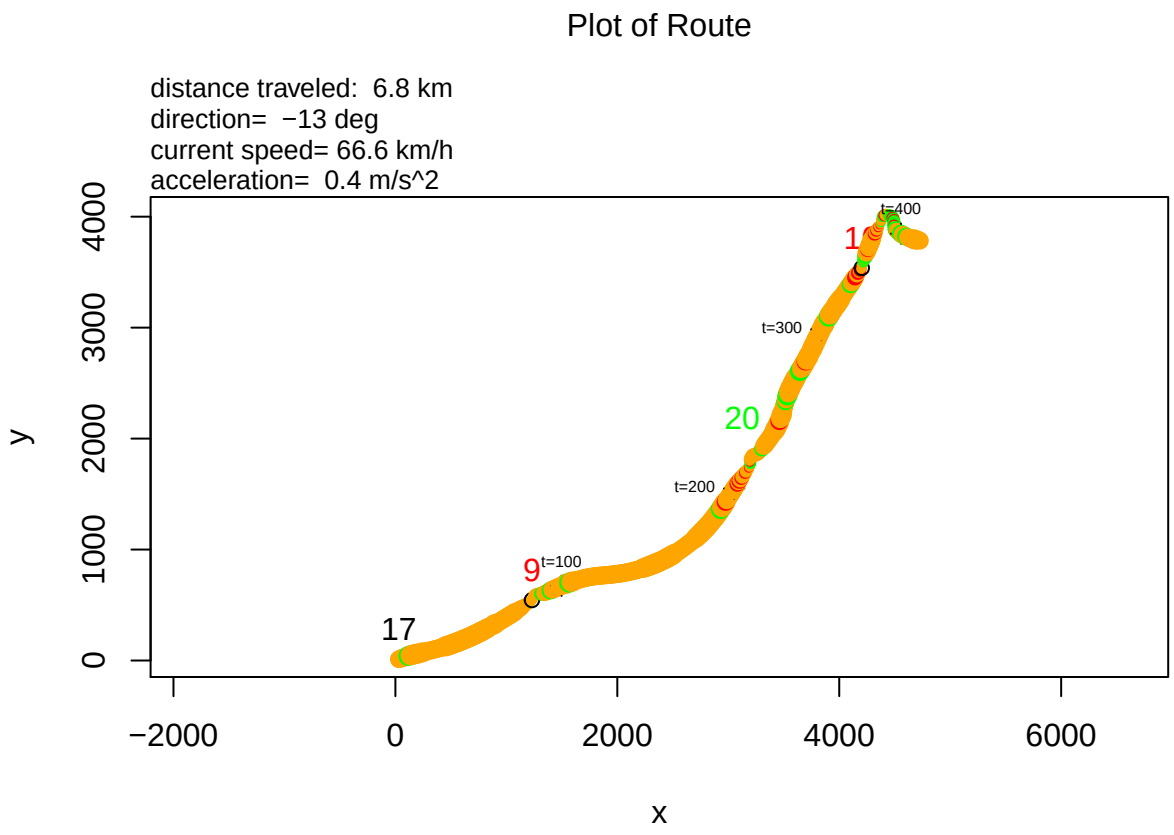
The first number (17) at x=y=0 is the initial speed. Since this isn't 0 in this case, we know that the beginning of the trip has been removed. The small "t=NNN" indicate where the driver is at different time points (in seconds) on the trip, this can be adjusted by calling `plotTrip` with `t.mark=100`.

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)

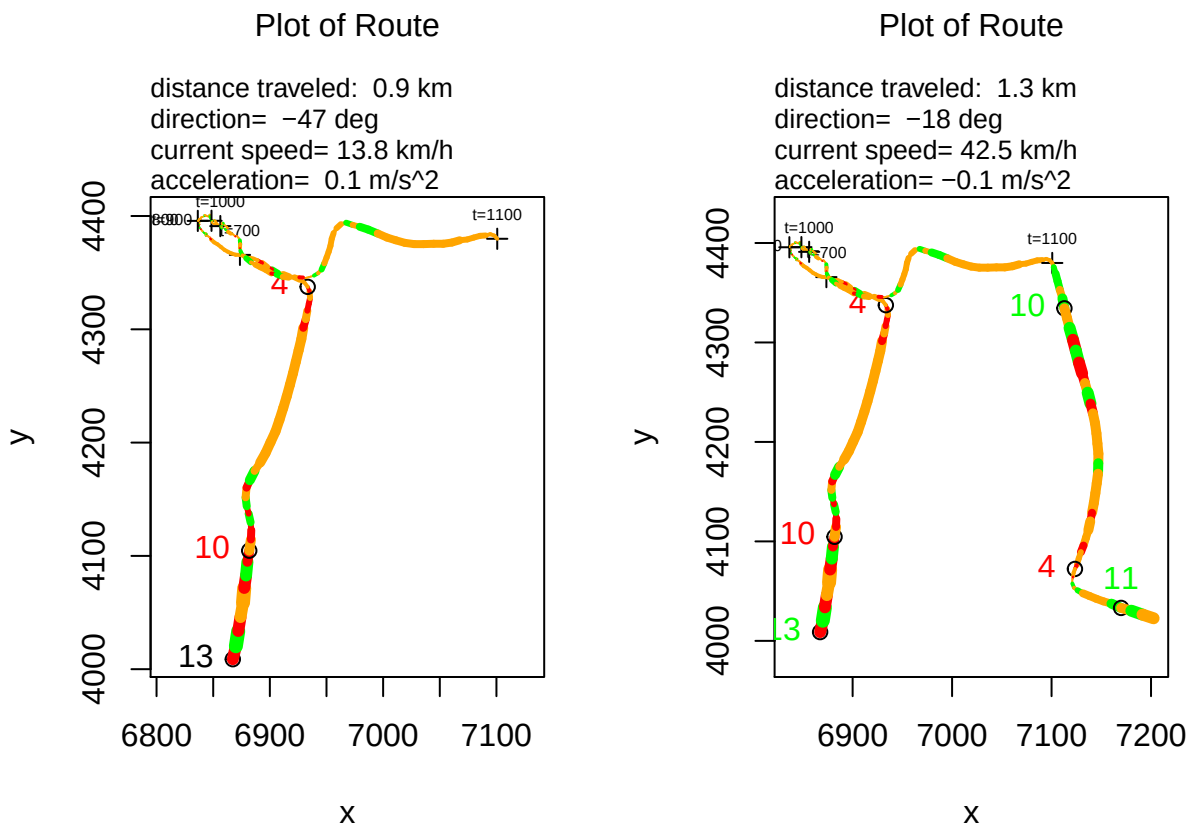
Note that the speed is shown in green when the driver is accelerating, and red when decelerating.

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



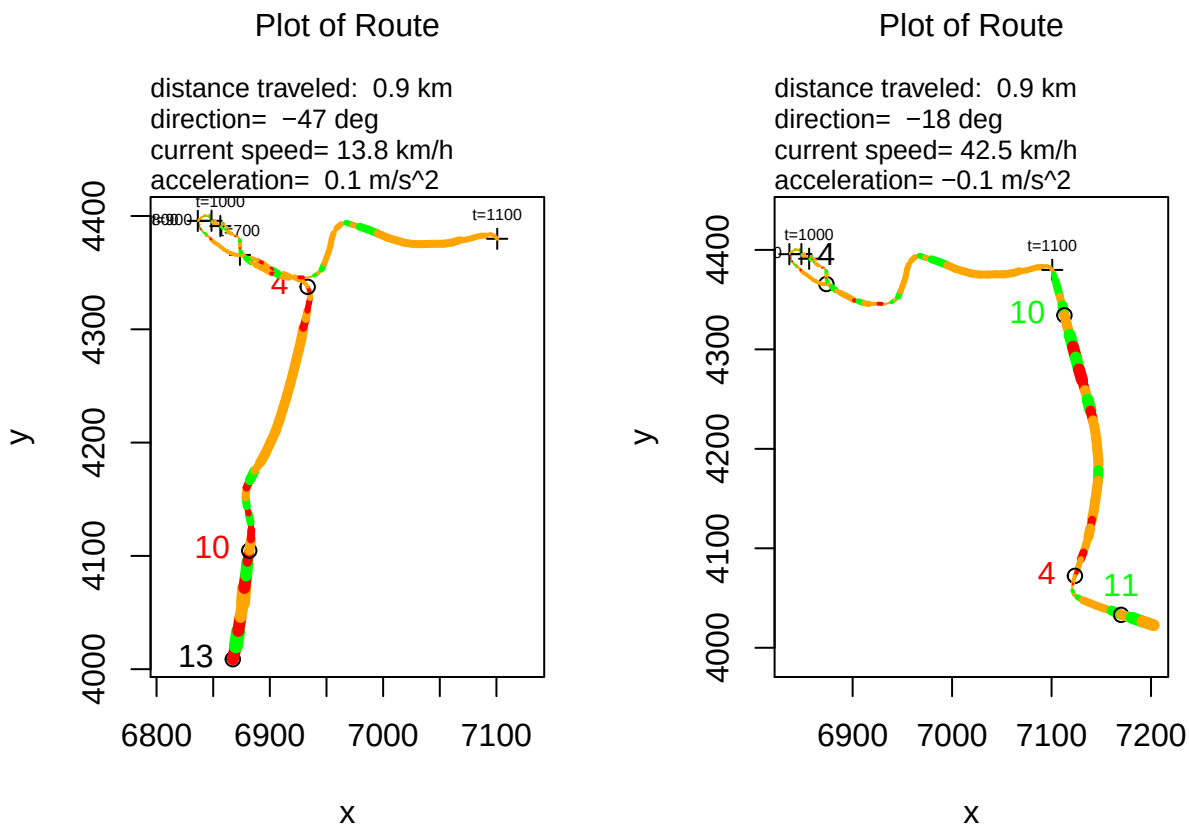
Putting this all together, you can do some quick detail analysis with some simple console commands. For this trip, the interesting stuff is near the end of the trip, so you can step through things like this:

```
par(mfrow=c(1,2))
i = 1050 #the time period we want to start with
i=i+50;  plotTrip(trip, tmin=650, tmax=i)
i=i+50;  plotTrip(trip, tmin=650, tmax=i)    # constant start point, grow segment
```



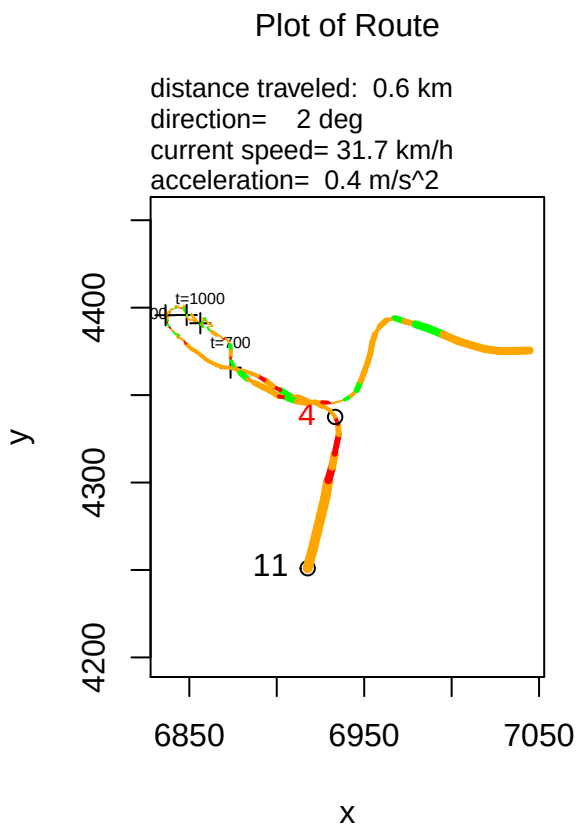
or

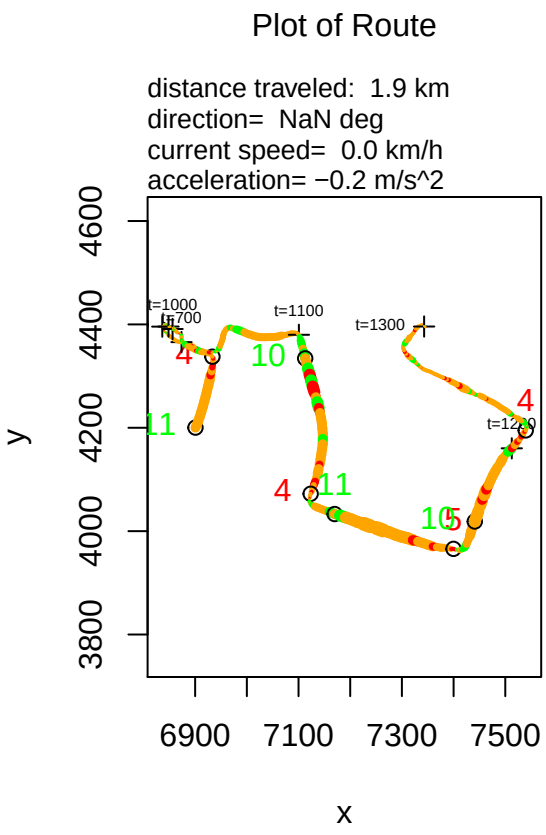
```
par(mfrow=c(1,2))
i = 1050 #the time period we want to start with
i=i+50;  plotTrip(trip, tmin=i-450, tmax=i)
i=i+50;  plotTrip(trip, tmin=i-450, tmax=i) # constant segment
```



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 ;-)

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

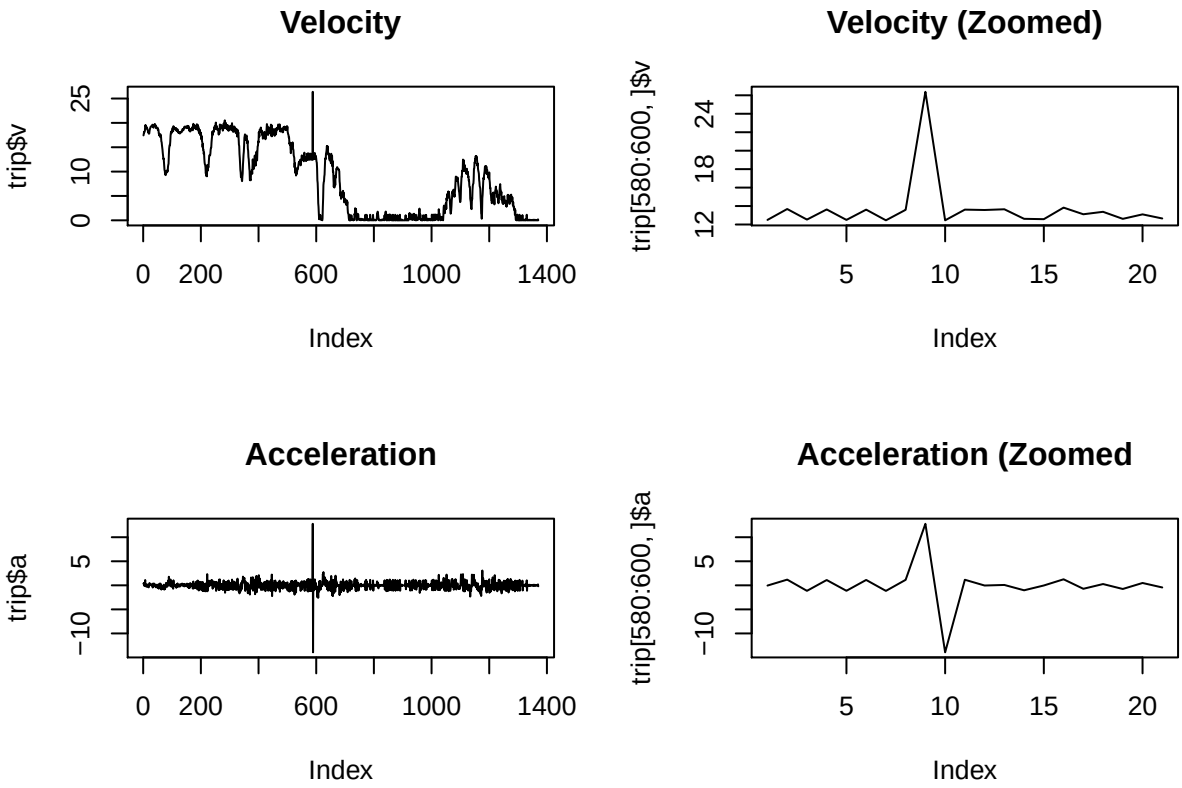




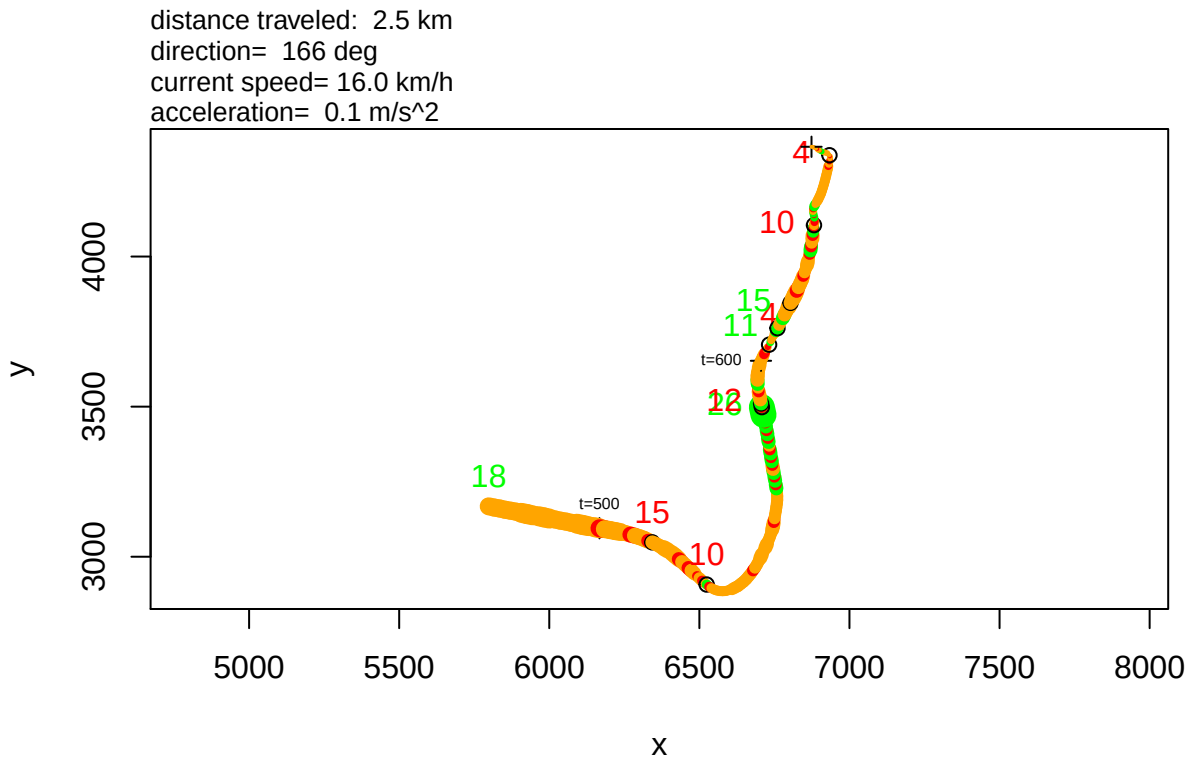

```
par(mfrow=c(1,1))
```

The next practical use of this tool is to look at the spikes in the velocity (and acceleration plots)

```
par(mfcol=c(2,2))
plot(trip$v, type="l", main="Velocity")
plot(trip$a, type="l", main="Acceleration")
plot(trip[580:600,]$v, type="l", main="Velocity (Zoomed)")
plot(trip[580:600,]$a, type="l", main="Acceleration (Zoomed")
```




```
par(mfrow=c(1,1))
plotTrip(trip, tmin=480, tmax=700)
```

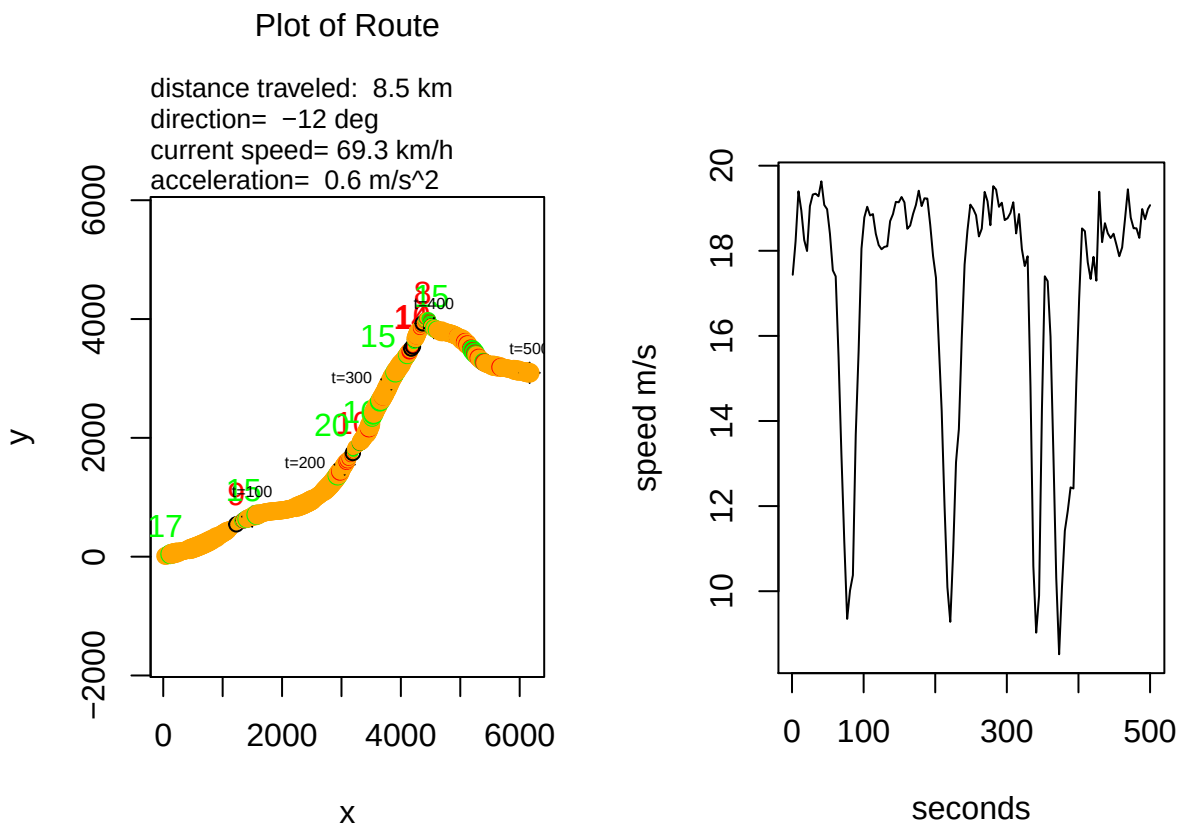


You can see it show up just before t=600, but there's nothing that would explain this other than an measurement or entry error, so we probably want to remove that data point and impute its value. That looks like a general data cleaning operation we'll need to do. (I remember reading something about this in the forums)

Moving on...

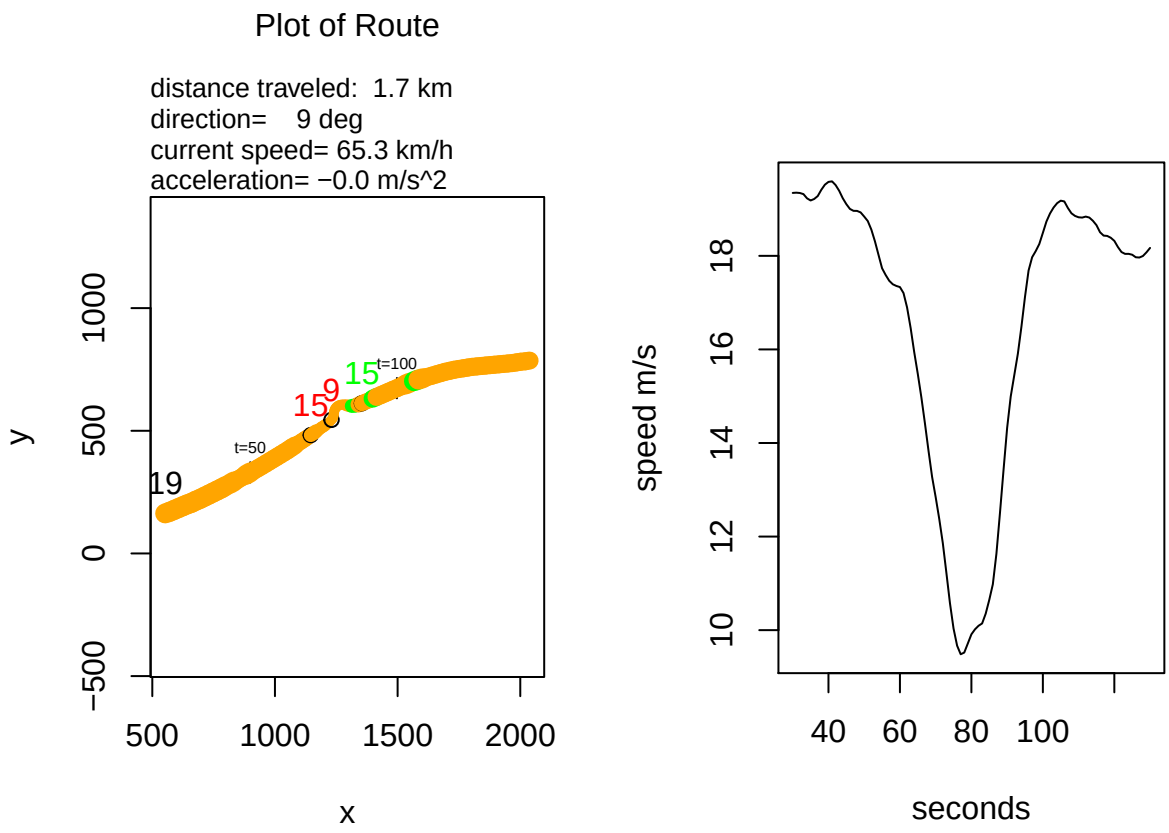
Looking at the velocity profile for the first part of the trip, it shows regular slow downs, but not stops. I wrote the functon `plotTripSegment` to help with zooming in on this.

```
plotTripSegment(trip, 1,500)
```



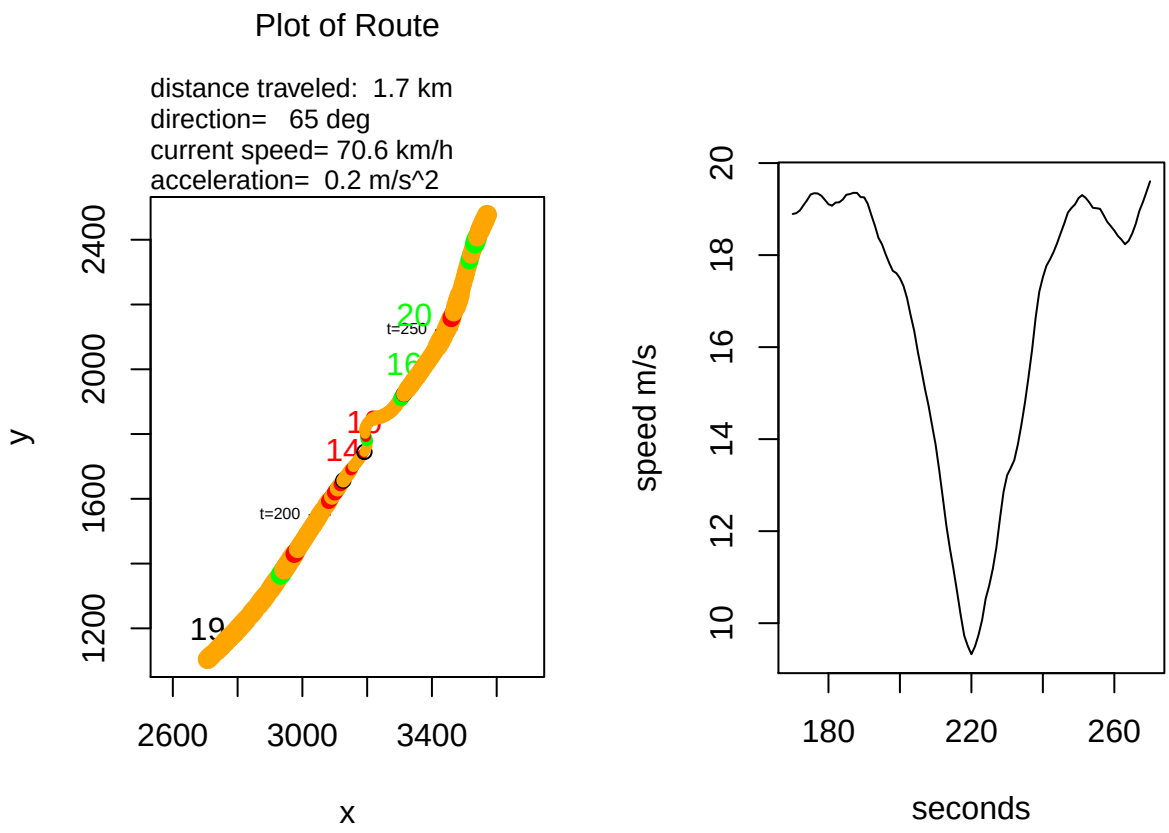
We'll narrow in on the first valley:

```
plotTripSegment(trip, 30, 130, t.mark=50, f=.06)
```



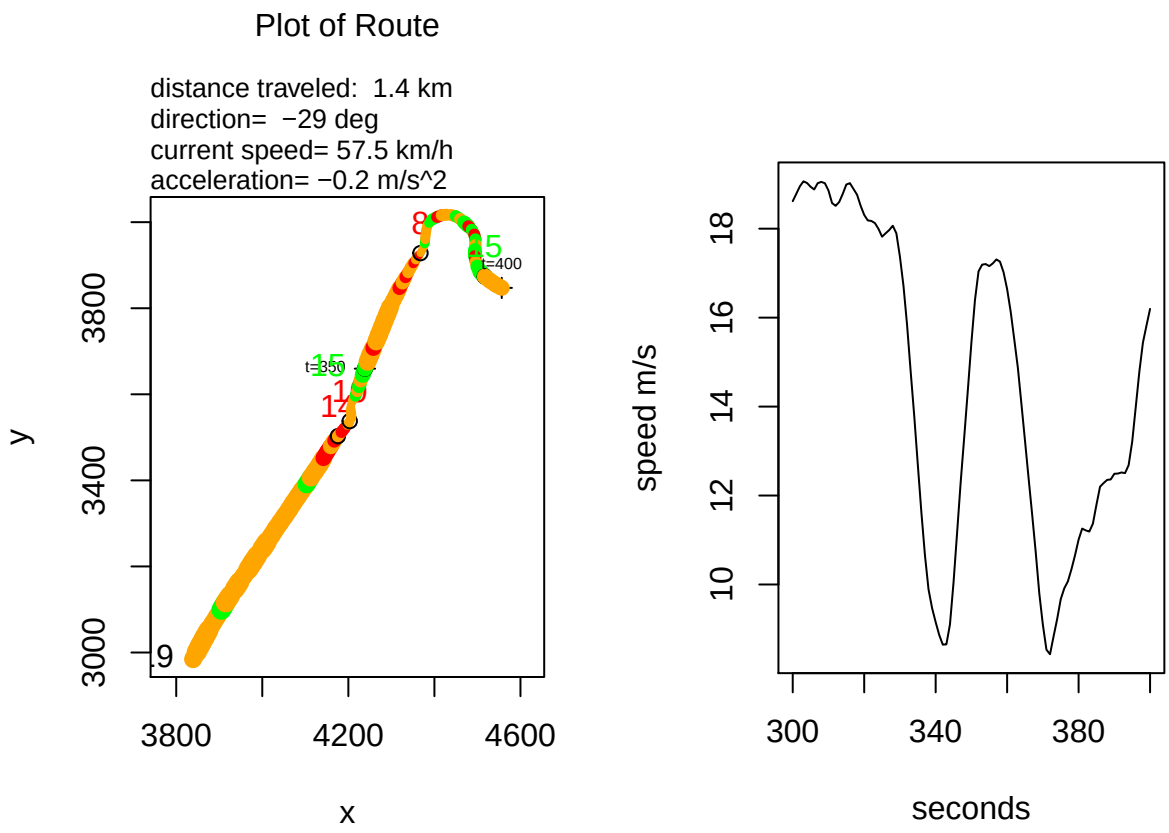
Hmmm... looks like a roundabout, and it goes to the left, so we are probably in England! The regularity of of the troughs is interesting. The next one appears to be identical. This suggests several possible features: When does the driver start to slow down going into a roundabout? When do they speed up? How fast do they take it?

```
plotTripSegment(trip, 170, 270, t.mark=50, f=.06)
```



The last trough looks like a roundabout, with an immediate obstacle (traffic)

```
plotTripSegment(trip, 300, 400, t.mark=50, f=.06)
```



From the map, it looks like a roundabout followed by a hairpin turn, from which the driver exited onto another road.

Conclusion

I'm starting to feel confident this tool will help decipher the trips. Most people seem to focussing on the routes themselves as the primary features, and its probably accurate to say that drivers will tend toward the same routes. I've heard just using trip length as a predictor gets you pretty far. That said, anyone who does that will totally miss when the driver takes a different route. So the next step is to try to break down the routes into segment types, and then see we can reconstruct the route generically. If we can do that, the differences in the driving behavior between similar segments should be strong predictors.