Trip Sequencing Part 2

Dave Hurst

Sunday, February 8, 2015

Continuing the work to convert a trip into a sequence of segments. The input will be a full trip, and the output will be broken up into several components: - A list of segments with details (or reference to details) on each segment - A sequence of segment ID's that comprise the trip - A of set data for each segment type

I'm going to try to reconstruct each trip out of segments made up of the following types:

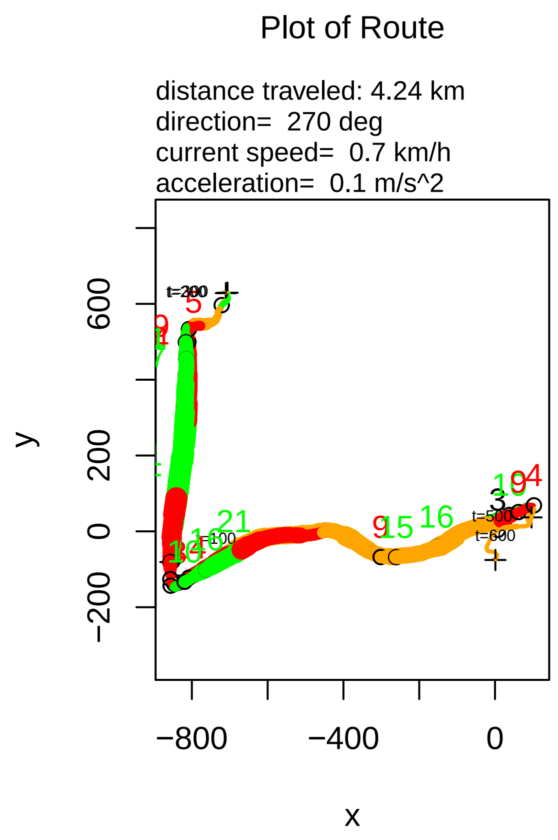
- stops *DONE: See part 1* (no driver profile for these, but we need them to describe the segment)
- sharp turns
- curves (differentiated from sharp turns by radii (as a function of speed))
- bends (differentiated from curves by time < 3 data points)
- straights
- uncategorized:
- – short straights (too short to collect profile info)
- – slow rolling sections (drive thrus, driveways, etc. – no profile info)
- – windy sections (ideally break these up later)
- – other?

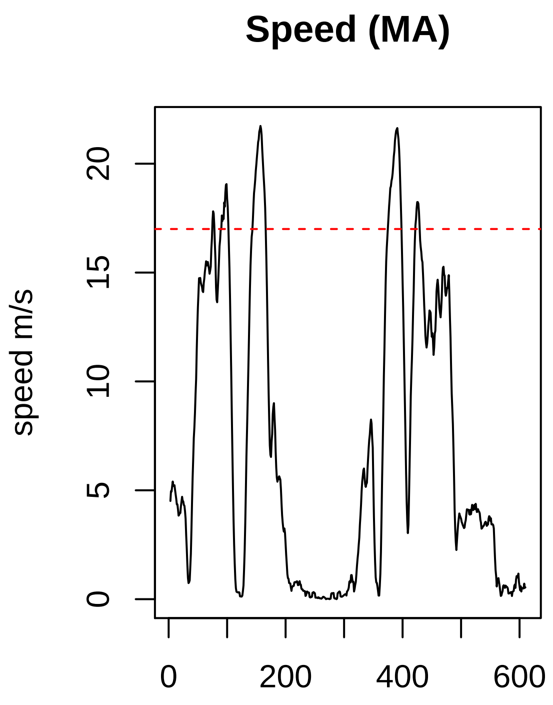
```
## Warning: package 'knitr' was built under R version 3.1.2
```

Again starting with the familiar 2591/49 trip:

```
driver.id <- 2591
trip.id <- 49
trip <- getTrip( driver.id, trip.id )

plotTripSegment(trip, 1, 99999)
```





Sharp Turns

Rules for determining a sharp turn * R < 20 m * Direction of curvature must be identical throughout curve (determined by cross product)

NOTE: Tunable parameters - Radius as a function of velocity?

First we need to parse the stops (from Part 1), and break the trip into segments

```
trip.seg <- segment.by.stops(trip)
trip.seg
```

```
##    id  t0  tn    type type.id
## 1   1   1 612 x.split      NA
## 2   2   1  31    <NA>      NA
## 3   3  32  38    stop       1
## 4   4  39 112    <NA>      NA
## 5   5 113 130    stop       2
## 6   6 131 204    <NA>      NA
## 7   7 205 313    stop       3
## 8   8 314 314 x.point      NA
## 9   9 315 324    stop       4
## 10 10 325 352    <NA>      NA
## 11 11 353 362    stop       5
## 12 12 363 558    <NA>      NA
## 13 13 559 595    stop       6
## 14 14 596 598    <NA>      NA
## 15 15 599 611    stop       7
```

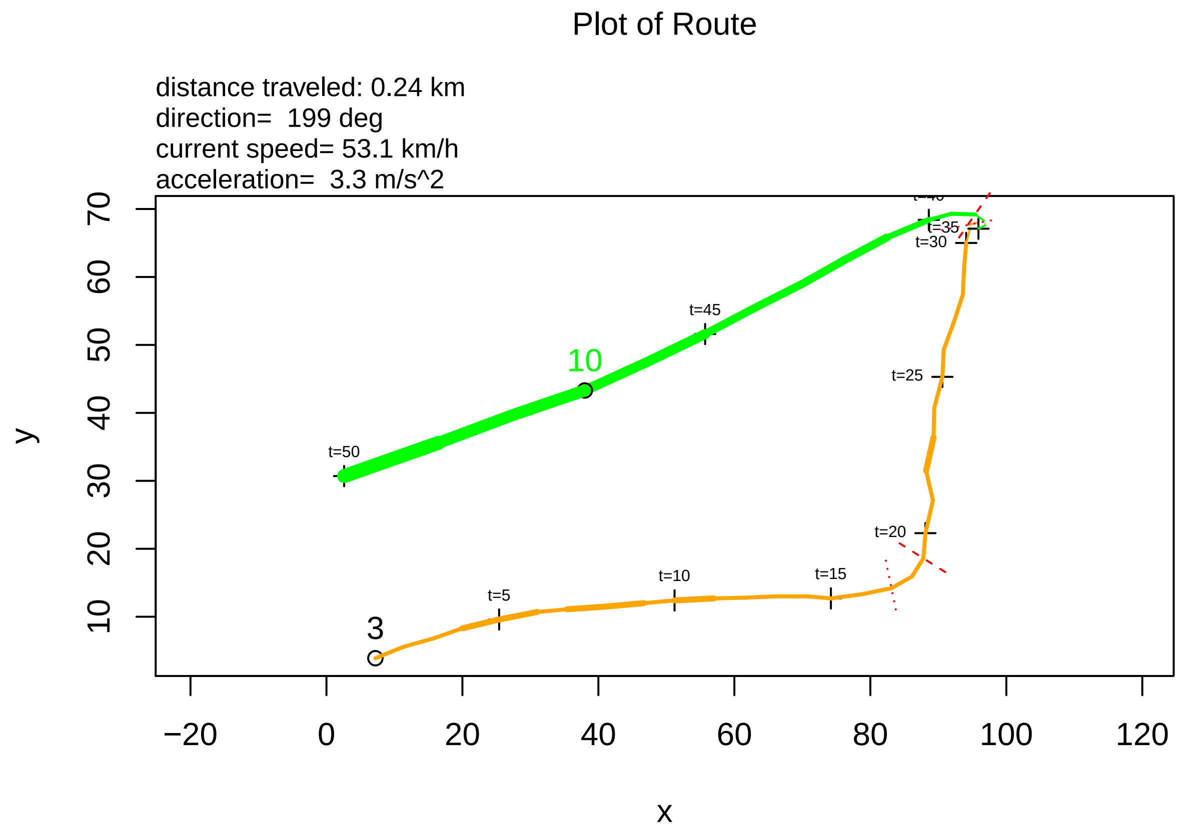
I created a function to parse turns call segment.parse.turns. We'll pass each unparsed segments (between stops above) to that to find all the sharp turns.

```
trip.seg <- segment.by.turns( trip, trip.seg)
trip.seg
```

```
##    id  t0  tn    type type.id
## 1   1   1 612 x.split      NA
## 2   2   1  31 x.split      NA
## 3   3  32  38    stop       1
## 4   4  39 112 x.split      NA
## 5   5 113 130    stop       2
## 6   6 131 204 x.split      NA
## 7   7 205 313    stop       3
## 8   8 314 314 x.point      NA
## 9   9 315 324    stop       4
## 10 10 325 352 x.split      NA
## 11 11 353 362    stop       5
## 12 12 363 558 x.split      NA
## 13 13 559 595    stop       6
## 14 14 596 598    <NA>      NA
## 15 15 599 611    stop       7
## 16 16   1  16    <NA>      NA
## 17 17  17  19    turn    2001
## 18 18  20  31    <NA>      NA
## 19 19  39 108    <NA>      NA
## 20 20 109 111    turn    4001
## 21 21 112 112 x.point      NA
## 22 22 131 172    <NA>      NA
## 23 23 173 175    turn    6001
## 24 24 176 199    <NA>      NA
## 25 25 200 203    turn    6002
## 26 26 204 204 x.point      NA
## 27 27 325 325 x.point      NA
## 28 28 326 328    turn   10001
## 29 29 329 352    <NA>      NA
## 30 30 363 407    <NA>      NA
## 31 31 408 411    turn   12001
## 32 32 412 488    <NA>      NA
## 33 33 489 491    turn   12002
## 34 34 492 492 x.point      NA
## 35 35 493 494    turn   12003
## 36 36 495 504    <NA>      NA
## 37 37 505 509    turn   12004
## 38 38 510 542    <NA>      NA
## 39 39 543 547    turn   12005
## 40 40 548 558    <NA>      NA
```

The modified `trip.seg` data frame is shown above. For a little graphical verification, we'll strip out the "turns" and "stops" and overlay those boundaries agains the plot. It's a little easier to see if we just take a small portion

```
plotTrip(trip, tmax=50, t.mark=5)
segs.1 <- trip.seg[grepl("stop|turn", trip.seg$type) & trip.seg$t0 <50,]
overlaySegmentBorders( trip, c( segs.1$t0, segs.1$tn), size= -1)
```



```
segs.1
```

```
##    id t0 tn type type.id
## 3   3 32 38 stop       1
## 17 17 17 19 turn    2001
```

Functions created/used

```
segment.clean.points
```

```
## function (s)
## {
##     for (i in 1:nrow(s)) s[i, "type"] <- ifelse(s[i, "t0"] ==
##         s[i, "tn"], "x.point", s[i, "type"])
##     s
## }
```

```
segment.by.stops
```

```
## function (trip, thresh.stop = 1, thresh.roll = 2)
## {
##     trip.seg <- data.frame(id = 1, t0 = 1, tn = nrow(trip), type = NA,
##         type.id = NA)
##     stops <- segment.parse.stops(trip, thresh.stop = 1, thresh.roll = 2)
##     stop.id <- 1:nrow(stops)
##     stops <- cbind(stop.id, stops)
##     t <- 1
##     i.seg <- orig.seg <- nrow(trip.seg)
##     for (i in stop.id) {
##         t0 <- stops[i, "t0"]
##         tn <- stops[i, "tn"]
##         if (t0 > t) {
##             i.seg <- i.seg + 1
##             trip.seg <- rbind(trip.seg, data.frame(id = i.seg,
##                 t0 = t, tn = t0 - 1, type = NA, type.id = NA))
##         }
##         i.seg <- i.seg + 1
##         trip.seg <- rbind(trip.seg, data.frame(id = i.seg, t0 = t0,
##             tn = tn, type = "stop", type.id = i))
##         t <- tn + 1
##     }
##     if (nrow(stops) > 0)
##         trip.seg[orig.seg, "type"] <- "x.split"
##     segment.clean.points(trip.seg)
## }
```

```
segment.parse.turns
```

```
## function (trip, tmin = 1, tmax = nrow(trip))
## {
##     tmin <- max(1, tmin)
##     tmax <- min(tmax, nrow(trip))
##     turn <- data.frame(t0 = integer(), tn = integer())
##     rinfo.trip <- calc.rinfo(trip)
##     t0 <- tmin
##     thresh <- 20
##     turning <- FALSE
##     xprod.last <- 0
##     for (t in (tmin + 1):tmax) {
##         rinfo <- rinfo.trip[t, ]
##         if (turning) {
##             if (xprod.last * rinfo$xprod < 0 | rinfo$r > thresh) {
##                 tn <- t - 1
##                 turn.seg <- data.frame(t0 = t0, tn = tn)
##                 turn.info <- cbind(trip[t0:tn, ], rinfo.trip[t0:tn,
##                   ])
##                 if (validate.turn(turn.info, rmax = thresh)) {
##                   turn <- rbind(turn, turn.seg)
##                 }
##                 turning <- FALSE
##                 t0 <- t
##             }
##         }
##         else {
##             if (rinfo$r <= thresh) {
##                 turning <- TRUE
##                 t0 <- t
##             }
##         }
##         xprod.last <- rinfo$xprod
##     }
##     return(turn)
## }
```

```
segment.by.turns
```

```
## function (trip, trip.seg)
## {
##     segs.unk <- trip.seg[is.na(trip.seg$type), ]
##     for (seg in segs.unk$id) {
##         seg.data <- segs.unk[segs.unk$id == seg, ]
##         t.beg <- seg.data$t0
##         t.fin <- seg.data$tn
##         turns <- with(seg.data, segment.parse.turns(trip, tmin = t.beg,
##             tmax = t.fin))
##         if (nrow(turns) > 0) {
##             t <- t.beg
##             i.seg <- nrow(trip.seg)
##             for (i in 1:nrow(turns)) {
##                 t0 <- turns[i, "t0"]
##                 tn <- turns[i, "tn"]
##                 if (t0 > t) {
##                   i.seg <- i.seg + 1
##                   trip.seg <- rbind(trip.seg, data.frame(id = i.seg,
##                     t0 = t, tn = t0 - 1, type = NA, type.id = NA))
##                 }
##                 i.seg <- i.seg + 1
##                 trip.seg <- rbind(trip.seg, data.frame(id = i.seg,
##                   t0 = t0, tn = tn, type = "turn", type.id = i +
##                     (1000 * seg)))
##                 t <- tn + 1
##             }
##             if (t <= t.fin) {
##                 i.seg <- i.seg + 1
##                 trip.seg <- rbind(trip.seg, data.frame(id = i.seg,
##                   t0 = t, tn = t.fin, type = NA, type.id = NA))
##             }
##             trip.seg[trip.seg$id == seg, "type"] <- "x.split"
##             turns$id <- 1:nrow(turns) + seg * 1000
##             if (exists("trip.turns")) {
##                 trip.turns <- rbind(trip.turns, turns)
##             }
##             else {
##                 trip.turns <- turns
##              }
##          }
##      }
##      segment.clean.points(trip.seg)
## }
```