Computing trip distances/times from OSRM and Azure Maps

Don Li

08/06/2020

```
library( data.table )
load( "../../dataset/OSRM.RData" )
```

Computing trips and distances using OSRM and Azure Maps

Given only the start and end points on a trajectory, we are asked to predict the estiamted time of arrival (ETA). Because the ETA is the sum of a waiting process, it is useful to get information from the underlying movement that comprises the trip.

To do this, we will use existing tools, `Open Source Routing Machine` and `Azure Maps`. My interest here is using these tools to predict the path distance of the trip and the predicted time. Because path distance is not given as a model input, I will treat it as a missing value in the new data, and we will use model-based imputation to get those values.

In other words: * For a new trip, we predict (impute) the path distance and get a rough estimate of speed * Use these imputed values in a model to predict the ETA for the trip

OSRM

To get the OSRM data, I used the `osrmr` package in R. It actually had some stuff that bypassed `try` statements, which caused it to fail on an error regardless of what happened. I extracted the functions and put them together without that nonsense. You can source the function from `osrm_route_fx.R`.

Let's look at how close OSRM was to our data. In the data frame below, we have summaries for each trip. `lat1` and `lng1` are the longitudes and latitudes for the start; `lat2` and `lng2` are the corresponding coordinates for the termination. `known_dist` is the Haversine path distance - this is computed from the distance between each of the GPS pings in the trip, and then summed over the trip. `known_time` is the known travel time in seconds. `OSMR_dist` is the path distance computed from OSRM - OSRM returns a path for us, and I just computed the distance between each of points and summed them over the trip. `OSMR_duration` is an estimate of the trip duration from OSMR.

```
load( "../../dataset/OSRM.RData" )
head( trip_summary )
```

```
##    trj_id     lat1     lng1     lat2     lng2 known_dist known_time OSMR_dist
## 1:     10 1.301775 103.7993 1.358001 103.8452   10.05098       1149  9.850573
## 2:    100 1.345079 103.9385 1.335207 103.8422   13.26885        993 12.791949
## 3:   1000 1.435317 103.7886 1.368858 103.8610   13.89963        856 14.905366
## 4:  10001 1.375079 103.8334 1.328267 103.7506   17.15718       1250 17.208198
## 5:  10004 1.433899 103.7687 1.392301 103.9088   18.82800        851 18.816671
## 6:  10005 1.371245 103.8712 1.301454 103.8431   10.22355       1028 10.159012
##    OSMR_duration
## 1:         704.0
## 2:         860.3
```

```
## 3:         856.9
## 4:        1193.3
## 5:         970.4
## 6:         574.9
```

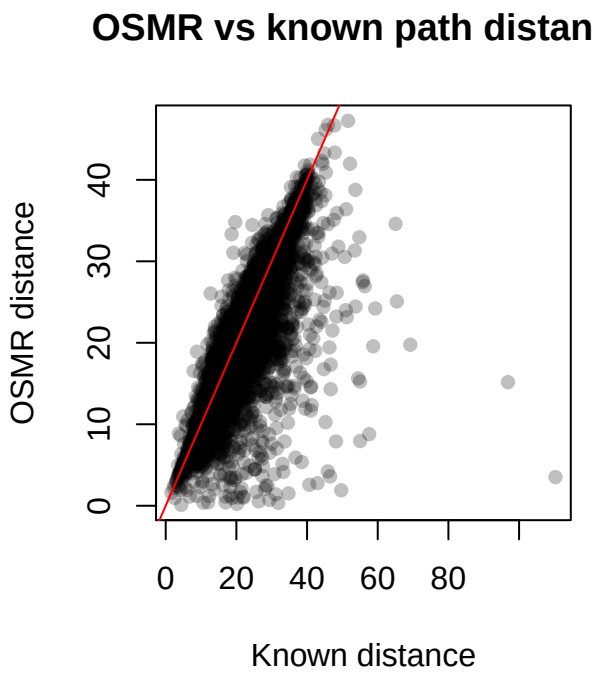
First, assess the distances. The OSMR distance accounts for about 80% of the variability in the trip distance. Based on my a priori expectations, this is quite good, since I expected drivers to just do whatever they want.

```
dist_model = lm( known_dist ~ OSMR_dist, data = trip_summary )
summary_dist_model = summary( dist_model )
summary_dist_model
```

```
##
## Call:
## lm(formula = known_dist ~ OSMR_dist, data = trip_summary)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -14.928  -0.954  -0.630   0.140 105.196
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 1.770433   0.053275   33.23   <2e-16 ***
## OSMR_dist   0.942848   0.002774  339.89   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.709 on 27998 degrees of freedom
## Multiple R-squared:  0.8049, Adjusted R-squared:  0.8049
## F-statistic: 1.155e+05 on 1 and 27998 DF,  p-value: < 2.2e-16
```

In the figure below, we can see that there are a couple of trips that are far longer than what OSMR predicts based on the start and the end (below the 0-1 line). These are trips where they take a fat loop around the city.

```
col = rgb( 0, 0, 0, 0.25 )
par( pty = "s" )
trip_summary[ , {
    plot( known_dist, OSMR_dist, pch = 16, col= col,
        main = "OSMR vs known path distance", ylab = "OSMR distance",
        xlab = "Known distance" )
    abline( 0, 1, col = "red" )
} ]
```

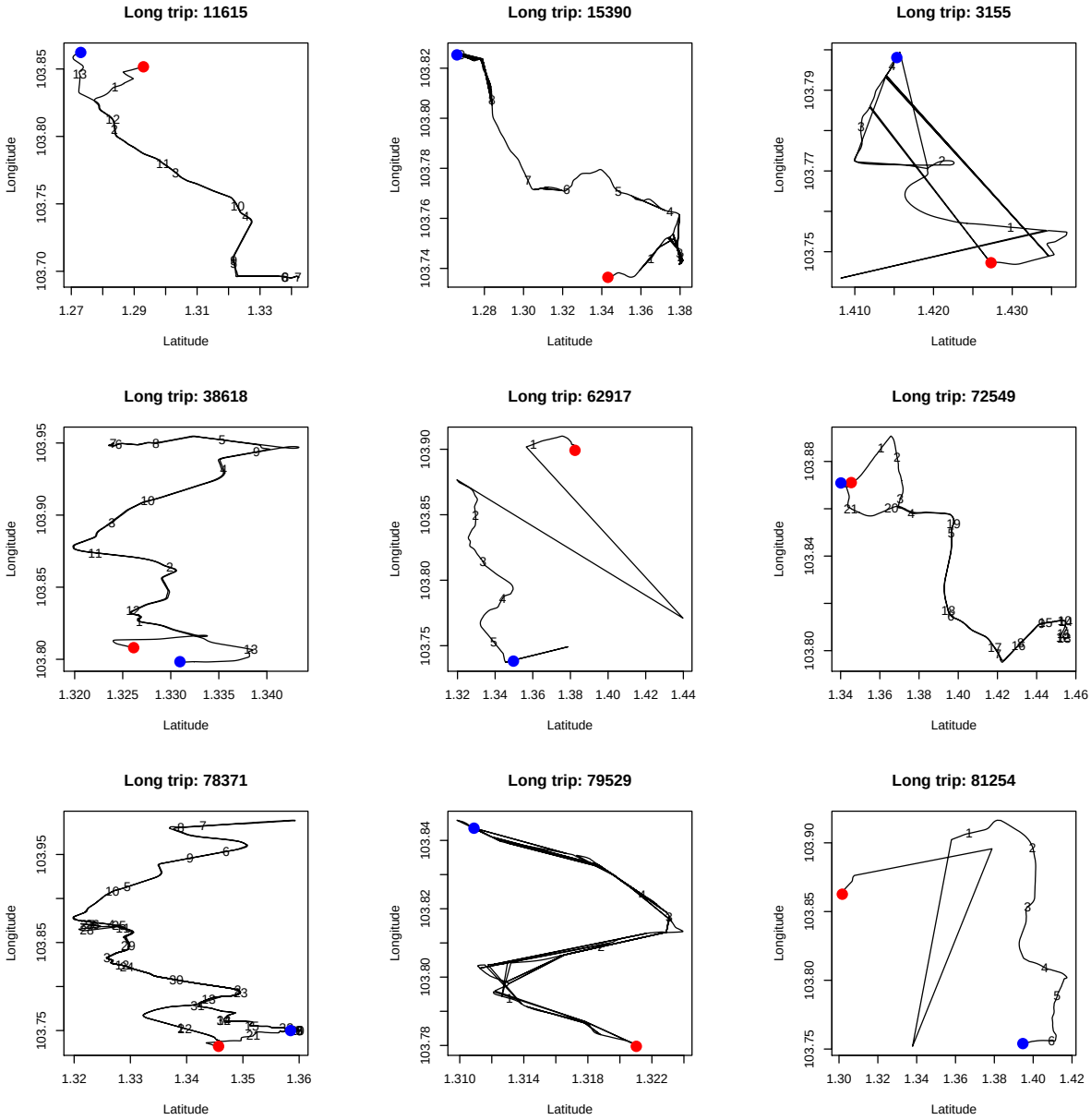



```
## NULL
```

We can see below, some of the trips with the most extreme deviation between the known path distance and the OSRM distance involve the driver taking some loops around the city. The red dot is the start and the blue dot is the end. The numbers are checkpoints that I labelled to get an indication of the sequence that the line points follow. Clearly, these are defective trips and we should consider removing them.

```
load( "../../dataset/all_SNG.RData" )
long_trips = trip_summary[ abs(OSMR_dist - known_dist) > 40, trj_id ]
all_data = all_data[ order( date_  ) ]
long_long = all_data[ trj_id %in% long_trips ]
par( mfrow = c(3,3), pty = "s" )

for ( trip_id in long_trips[1:9] ){
    temp_data = long_long[ trj_id == trip_id ]
    N = nrow( temp_data )
    checkpts = (1:N %% 200) == 0
    plot( temp_data$rawlat, temp_data$rawlng, type = "l",
        main = paste0( "Long trip: ", trip_id ),
        xlab = "Latitude", ylab = "Longitude" )
    text( temp_data$rawlat[ checkpts ], temp_data$rawlng[ checkpts ],
        1:sum(checkpts) )
    points( temp_data$rawlat[c(1,N)], temp_data$rawlng[c(1,N)], col = c("red","blue"),
        pch = 16, cex = 2 )
}
```



We have some other trips with very big jumps in their locations. It is possible that these are just due to the data not being ordered properly, but I did explicitly reorder them by date in the chunk above, to prove there isn't a pre-processing error. If we look at some trips with large jumps:

```
all_data[ trj_id == 81254, summary(speed) ]
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   -1.00   17.34   20.88   19.04   23.98   28.90
```

```
all_data[ trj_id == 62917, summary(speed) ]
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   -1.00   16.04   19.44   18.03   22.16   25.18
```

We can see that some of their GPS speeds are negative. Speed can't be negative without time travel, so there is probably some sort of error there.

Removing negative speed trips

Let's remove these invalid trips and see if the OSRM is more accurate.

```
invalid_trips = all_data[ , any( speed < 0 ), by = "trj_id" ]
valid_id = invalid_trips[, trj_id[!V1] ]
invalid_trips[ , sum(V1) ]
```

```
## [1] 10092
```

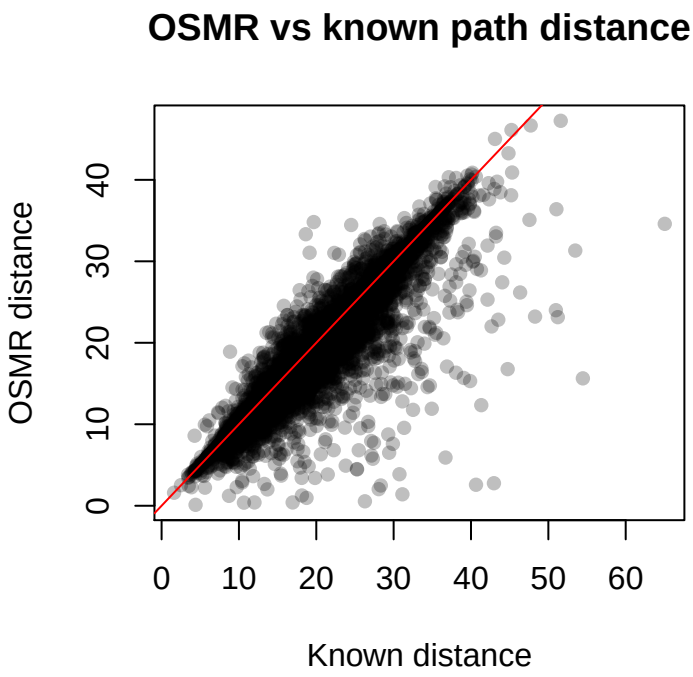
Almost half the trips have some kind of error with negative GPS speed. Need to think hard about this problem.

With the trips with defective pings removed, the OSRM routes explain 86% of the variability in the path distances. Seems good enough for what we want.

```
valid_trips = trip_summary[ trj_id %in% valid_id ]
dist_model_valid = lm( known_dist ~ OSMR_dist, data = valid_trips )
summary( dist_model_valid )
```

```
##
## Call:
## lm(formula = known_dist ~ OSMR_dist, data = valid_trips)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -14.875  -0.787  -0.510   0.123  38.842
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 1.497206   0.053167   28.16   <2e-16 ***
## OSMR_dist   0.949191   0.002777  341.81   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.171 on 17906 degrees of freedom
## Multiple R-squared:  0.8671, Adjusted R-squared:  0.8671
## F-statistic: 1.168e+05 on 1 and 17906 DF,  p-value: < 2.2e-16
```

```
valid_trips[ , {
    plot( known_dist, OSMR_dist, pch = 16, col= col,
        main = "OSMR vs known path distance", ylab = "OSMR distance",
        xlab = "Known distance" )
    abline( 0, 1, col = "red" )
} ]
```




```
## NULL
```

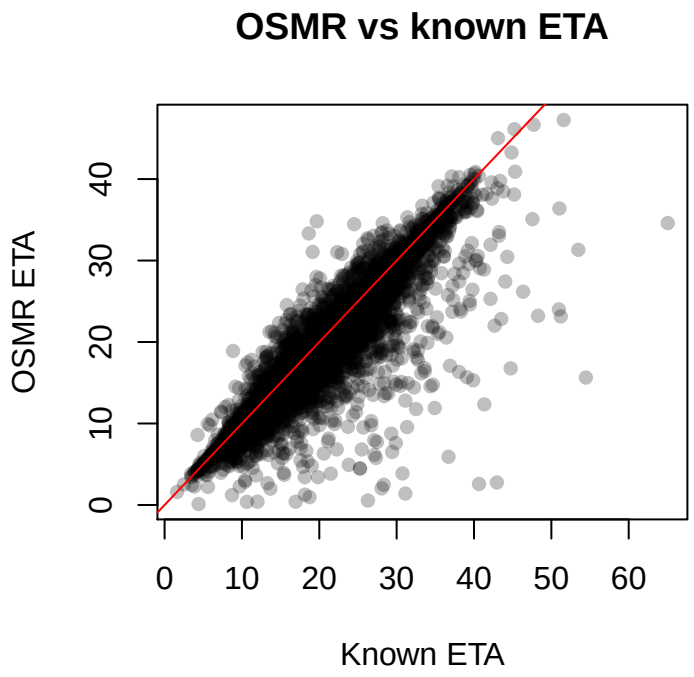
OSRM times

OSRM predicted ETA accounts for 23% of the variability in the trip time. Not too bad for a single covariate. Note that this is only using the valid trips.

```
time_model = lm( known_time ~ OSMR_duration, valid_trips )
summary( time_model )
```

```
##
## Call:
## lm(formula = known_time ~ OSMR_duration, data = valid_trips)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -798.8 -171.3  -49.9  101.0 5767.0
##
## Coefficients:
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept)   6.899e+02  7.502e+00   91.96   <2e-16 ***
## OSMR_duration 4.857e-01  6.499e-03   74.74   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 273.2 on 17906 degrees of freedom
## Multiple R-squared:  0.2378, Adjusted R-squared:  0.2377
## F-statistic:  5585 on 1 and 17906 DF,  p-value: < 2.2e-16
```

```
valid_trips[ , {
    plot( known_dist, OSMR_dist, pch = 16, col= col,
        main = "OSMR vs known ETA", ylab = "OSMR ETA",
        xlab = "Known ETA" )
    abline( 0, 1, col = "red" )
} ]
```




```
## NULL
```

Predicting time using OSMR data

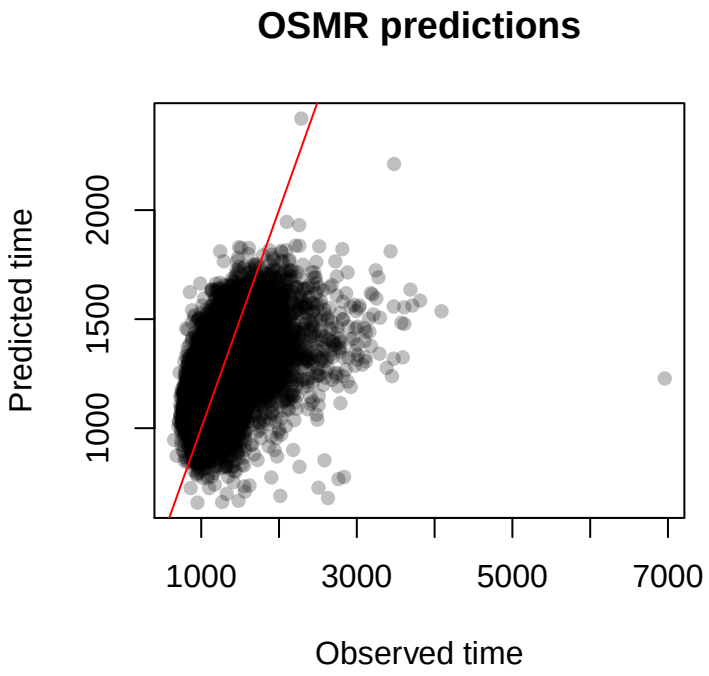
Our R-squared is now 26%, so seems okay.

```
time_model2 = lm( known_time ~ OSMR_duration + OSMR_dist, valid_trips )
summary( time_model2 )
```

```
##
## Call:
## lm(formula = known_time ~ OSMR_duration + OSMR_dist, data = valid_trips)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -769.8 -167.0  -48.2  100.9 5730.9
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)   641.4781     7.6772   83.56   <2e-16 ***
## OSMR_duration   1.0124     0.0235   43.08   <2e-16 ***
## OSMR_dist     -29.4327     1.2635  -23.30   <2e-16 ***
```

```
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 269.2 on 17905 degrees of freedom
## Multiple R-squared:  0.2602, Adjusted R-squared:  0.2601
## F-statistic:  3149 on 2 and 17905 DF,  p-value: < 2.2e-16
```

```
col = rgb( 0, 0, 0, 0.25 )
plot(
    valid_trips$known_time, predict( time_model2 ),
    xlab = "Observed time", ylab = "Predicted time",
    main = "OSMR predictions", col = col,
    pch = 16
)
abline( 0, 1, col = "red" )
```



Azure maps

Do all the same stuff, but with Azure maps instead.

```
load( "../../dataset/Azure_part.RData" )
head( trip_summary )
```

```
##    trj_id     lat1     lng1     lat2     lng2 known_dist known_time azure_dist
## 1:     10 1.301775 103.7993 1.358001 103.8452   10.05098       1149      9.797
## 2:    100 1.345079 103.9385 1.335207 103.8422   13.26885        993     14.696
## 3:   1000 1.435317 103.7886 1.368858 103.8610   13.89963        856     13.798
## 4:  10001 1.375079 103.8334 1.328267 103.7506   17.15718       1250     17.130
## 5:  10004 1.433899 103.7687 1.392301 103.9088   18.82800        851     18.786
```

```
## 6:  10005 1.371245 103.8712 1.301454 103.8431   10.22355       1028     10.111
##    azure_ETA
## 1:       754
## 2:      1056
## 3:       724
## 4:      1263
## 5:       854
## 6:       549
```

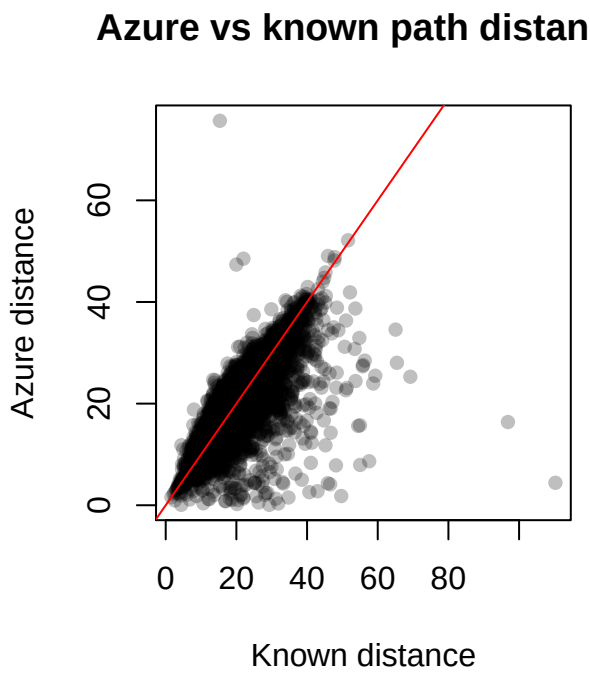
```
dist_model_Azure = lm( known_dist ~ azure_dist, data = trip_summary )
summary_dist_model_Azure = summary( dist_model_Azure )
summary_dist_model_Azure
```

```
##
## Call:
## lm(formula = known_dist ~ azure_dist, data = trip_summary)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -56.903  -0.620  -0.242   0.176 104.735
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 1.409534   0.053762   26.22   <2e-16 ***
## azure_dist  0.936178   0.002727  343.26   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.687 on 27998 degrees of freedom
## Multiple R-squared:  0.808,  Adjusted R-squared:  0.808
## F-statistic: 1.178e+05 on 1 and 27998 DF,  p-value: < 2.2e-16
```

The Azure maps distance is about the same, R-squared is 0.808, which is about the same as for the OSRM 0.805.

We notice a few overpredicted outliers that the OSRM did not have, but it's only 3 data points, so should be negligible.

```
col = rgb( 0, 0, 0, 0.25 )
par( pty = "s" )
trip_summary[ , {
    plot( known_dist, azure_dist, pch = 16, col= col,
        main = "Azure vs known path distance", ylab = "Azure distance",
        xlab = "Known distance" )
    abline( 0, 1, col = "red" )
} ]
```




```
## NULL
```

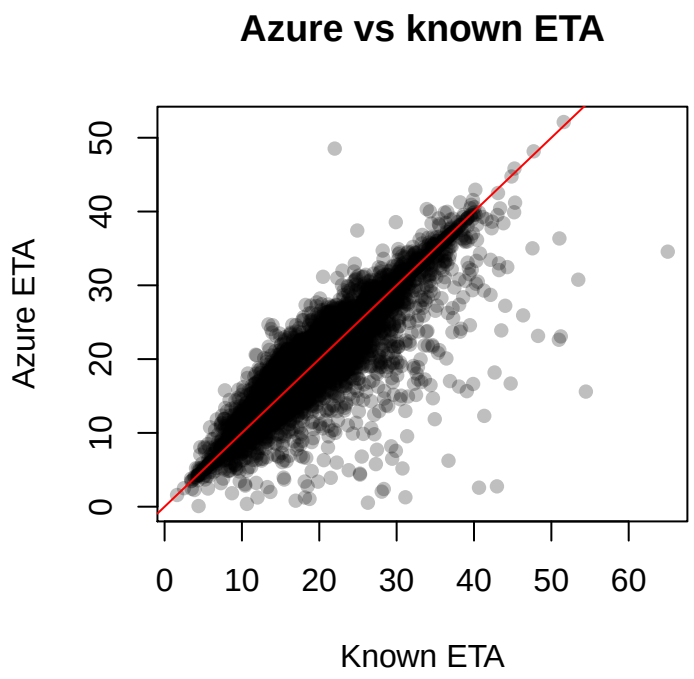
Azure times

Azure predicted ETA accounts for 24% of the variability in the trip time. Quite similar to OSRM.

```
valid_trips = trip_summary[ trj_id %in% valid_id ]
time_model_Azure = lm( known_time ~ azure_dist, valid_trips )
summary( time_model )
```

```
##
## Call:
## lm(formula = known_time ~ OSMR_duration, data = valid_trips)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -798.8 -171.3  -49.9  101.0 5767.0
##
## Coefficients:
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept)   6.899e+02  7.502e+00   91.96   <2e-16 ***
## OSMR_duration 4.857e-01  6.499e-03   74.74   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 273.2 on 17906 degrees of freedom
## Multiple R-squared:  0.2378, Adjusted R-squared:  0.2377
## F-statistic:  5585 on 1 and 17906 DF,  p-value: < 2.2e-16
```

```
valid_trips[ , {
    plot( known_dist, azure_dist, pch = 16, col= col,
        main = "Azure vs known ETA", ylab = "Azure ETA",
        xlab = "Known ETA" )
    abline( 0, 1, col = "red" )
} ]
```




```
## NULL
```

Predicting time using Azure data

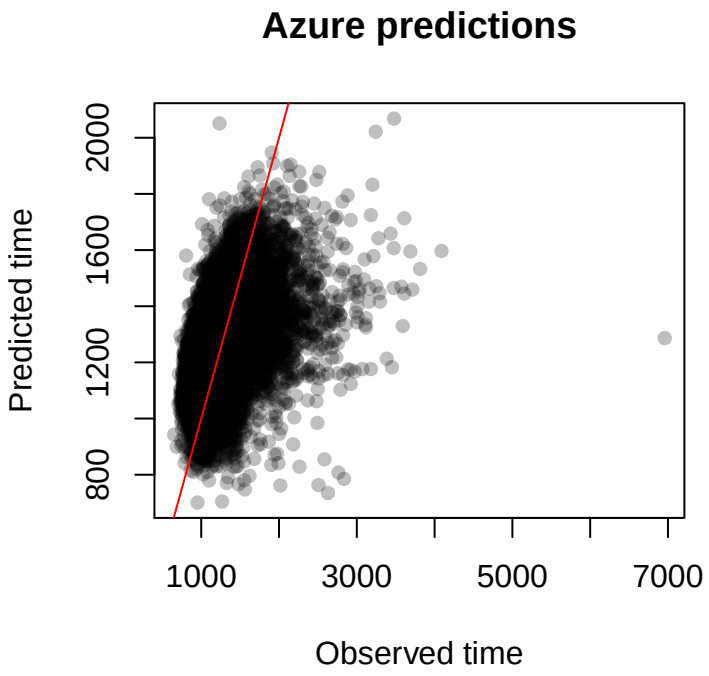
Our R-squared is now 26%, similar to OSRM.

```
time_model2_Azure = lm( known_time ~ azure_ETA + azure_dist, valid_trips )
summary( time_model2 )
```

```
##
## Call:
## lm(formula = known_time ~ OSMR_duration + OSMR_dist, data = valid_trips)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -769.8 -167.0  -48.2  100.9 5730.9
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)   641.4781     7.6772   83.56   <2e-16 ***
## OSMR_duration   1.0124     0.0235   43.08   <2e-16 ***
## OSMR_dist     -29.4327     1.2635  -23.30   <2e-16 ***
```

```
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 269.2 on 17905 degrees of freedom
## Multiple R-squared:  0.2602, Adjusted R-squared:  0.2601
## F-statistic:  3149 on 2 and 17905 DF,  p-value: < 2.2e-16
```

```
col = rgb( 0, 0, 0, 0.25 )
plot(
    valid_trips$known_time, predict( time_model2_Azure ),
    xlab = "Observed time", ylab = "Predicted time",
    main = "Azure predictions", col = col,
    pch = 16
)
abline( 0, 1, col = "red" )
```

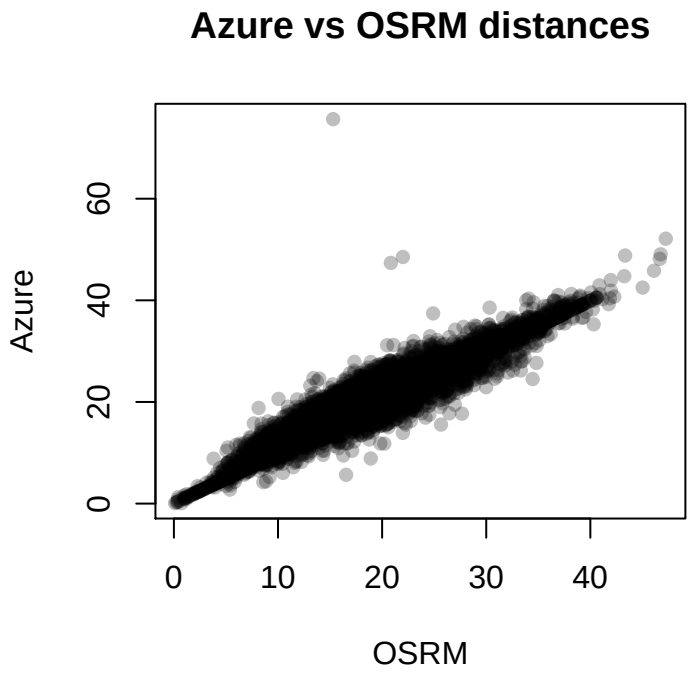


Comparisons

```
Azure_trip_summary = trip_summary
load( "../../dataset/OSRM.RData" )
# Perform a join
trip_summary[ Azure_trip_summary, c("azure_dist", "azure_eta") :={
    list( i.azure_dist, i.azure_ETA )
}, on = "trj_id" ]
```

In the plot below, we see a good match between the Azure and OSRM routes. A couple of overpredictions the Azure maps though.

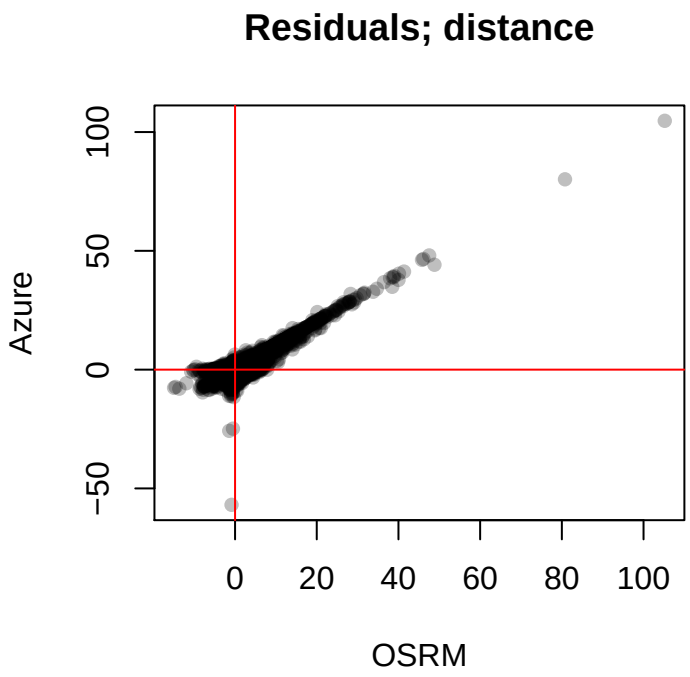
```
trip_summary[ , {
    plot( OSMR_dist, azure_dist, col = col, pch = 16,
        main = "Azure vs OSRM distances",
        xlab = "OSRM", ylab = "Azure" )
    } ]
```




```
## NULL
```

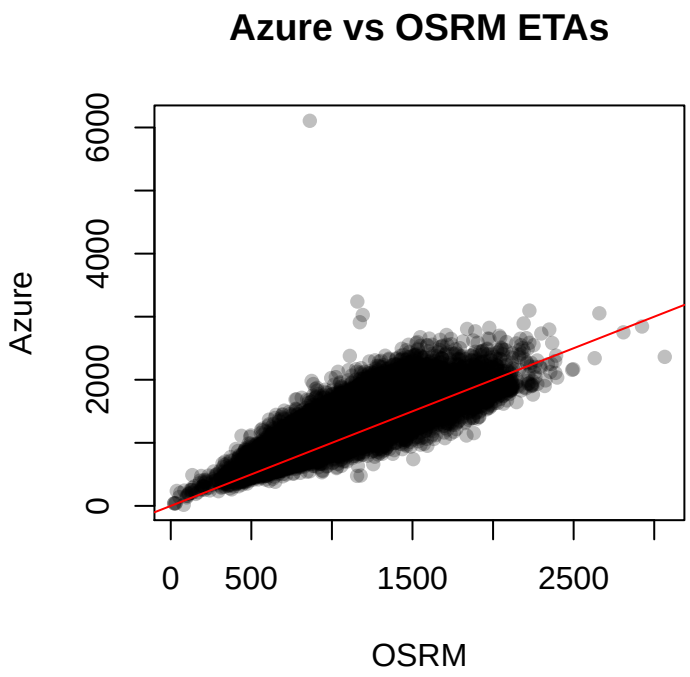
To see if having both distances will be useful, we will study the correlations in the residuals using a linear model. Since we see some spread around (0,0), it might be possible to combine the two distances into a better distance estimate.

```
plot(
    dist_model$residuals, dist_model_Azure$residuals,
    col = col, pch = 16,
    main = "Residuals; distance",
    xlab = "OSRM", ylab = "Azure"
)
abline( v = 0, h = 0, col = "red" )
```



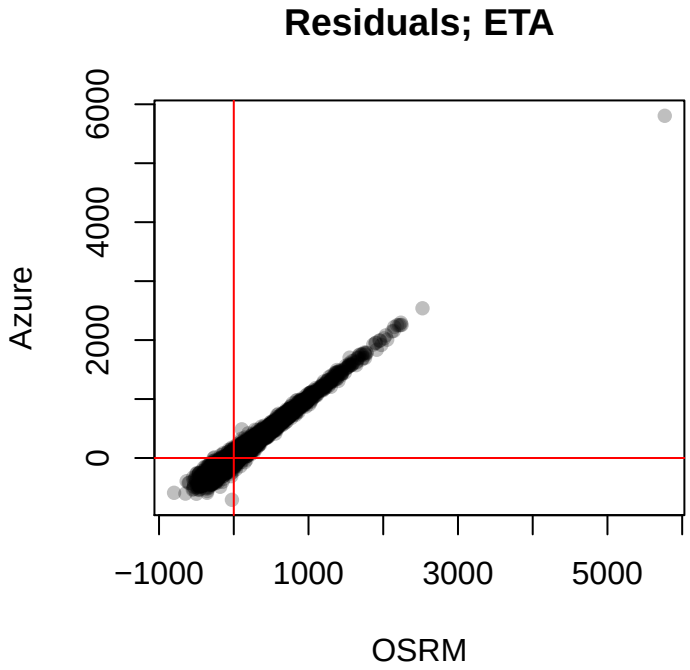
We can do the same thing with the travel time. Same conclusions as the distance; Azure has a bit of overprediction.

```
trip_summary[ , {
    plot( OSMR_duration, azure_eta, col = col, pch = 16,
        main = "Azure vs OSRM ETAs",
        xlab = "OSRM", ylab = "Azure" )
    abline( 0, 1, col = "red" )
    } ]
```




```
## NULL
```

```
plot(
    time_model$residuals, time_model_Azure$residuals,
    col = col, pch = 16,
    main = "Residuals; ETA",
    xlab = "OSRM", ylab = "Azure"
)
abline( v = 0, h = 0, col = "red" )
```



Conclusions

Azure Maps and OSRM give very similar routing/ETAs.

There are a lot of trips with negative speed that represent defective pathing data. Not sure what to do that this moment, or if it even matters.

Next steps: * Use the Azure and OSRM routes to reconstruct path distances as covariates * Use the Azure and OSRM ETAs to predict ETA