basic_regression

Don Li

02/06/2020

```
library( data.table )
```

Our objective is to predict the estimated time of arrival for a journey. We will start with the most basic model, linear regression. Then we will build it up from there.

Workflow stuff

Read the data in with Python. I will use R for prototyping because we only have two weeks and I don't have time to get good at the snake language.

```
import numpy as np
import pandas as pd
import pickle
import os
from datetime import datetime
import pyarrow.parquet as pq
import matplotlib.pyplot as plt

pd.set_option( "display.max_columns", None )

filename = "part-00000-8bbff892-97d2-4011-9961-703e38972569.c000.snappy.parquet"
dataset = pq.read_table( filename ).to_pandas()
```

When this Py chunk gets evaluated, it returns an R dataframe. The result of the document will be working on that dataframe.

Variable transformations

The `pingtimestamp` has a lot of information. I think the day of the month (`monthday`), day of the year (`yearday`), day of the week (`weekday`), the month (`month`), hour (`hour`), and minute `min`, will be useful. Of couse, we also want the actual date so we can compute differences in time.

```
dataset = data.table( dataset )
dataset = dataset[ order( trj_id, pingtimestamp ) ]

dataset[ , c("monthday", "yearday", "weekday", "date", "month",
    "hour", "min") := {
    date_ = .POSIXct(pingtimestamp)
    monthday = as.numeric( format( date_, "%d" ) )

    yearday = format( date_, "%W" )

    weekday = format( date_, "%a" )
    weekday = factor( weekday,
        levels = c("Sun", "Mon", "Tue", "Wed",
```

```
            "Thu", "Fri", "Sat") )

    month_F = format( date_ , "%b" )

    month_F = factor( month_F, ordered = T )

    hour = as.numeric( format( date_, "%H" ) )

    min = as.numeric( format( date_, "%M" ) )

    list( day = monthday, yearday = yearday,
        weekday = weekday,
        date_ = date_,
        month = month_F,
        hour = hour, min = min )
    } ]
```

A function to get the time difference in seconds.

```
difftime_mins = function( time1, time2 ){
    x = difftime( time1, time2, units = "secs" )
    as.numeric(x)
}
```

First model

Our most basic model is regressing the estimated time of arrival based on trip-level information. So, this is not a moment-by-moment solution at the moment. This is just to see what is or could be important, and it also serves as a base case.

We will also add additional variables, such as the mean speed over the trip (`speed_avg`) and also the variance fo the speed (`speed_var`). We also use the Euclidean distance (`dist_`).

```
training_set = dataset[ , {
    indices = c(1,.N)

    lat_diff = diff( rawlat[indices] )
    lng_diff = diff( rawlng[indices] )
    dist_ = sqrt( lat_diff^2 + lng_diff^2 )

    timediff = difftime_mins( date[.N], date[1] )

    speed_avg = mean(speed)
    speed_var = var(speed)

    monthday_ = monthday[1]
    weekday_ = weekday[1]
    yearday_ = yearday[1]
    hour_ = hour[1]
    min_ = min[1]
    month_ = month[1]

    list( dist_ = dist_, timediff = timediff, speed_avg = speed_avg,
```

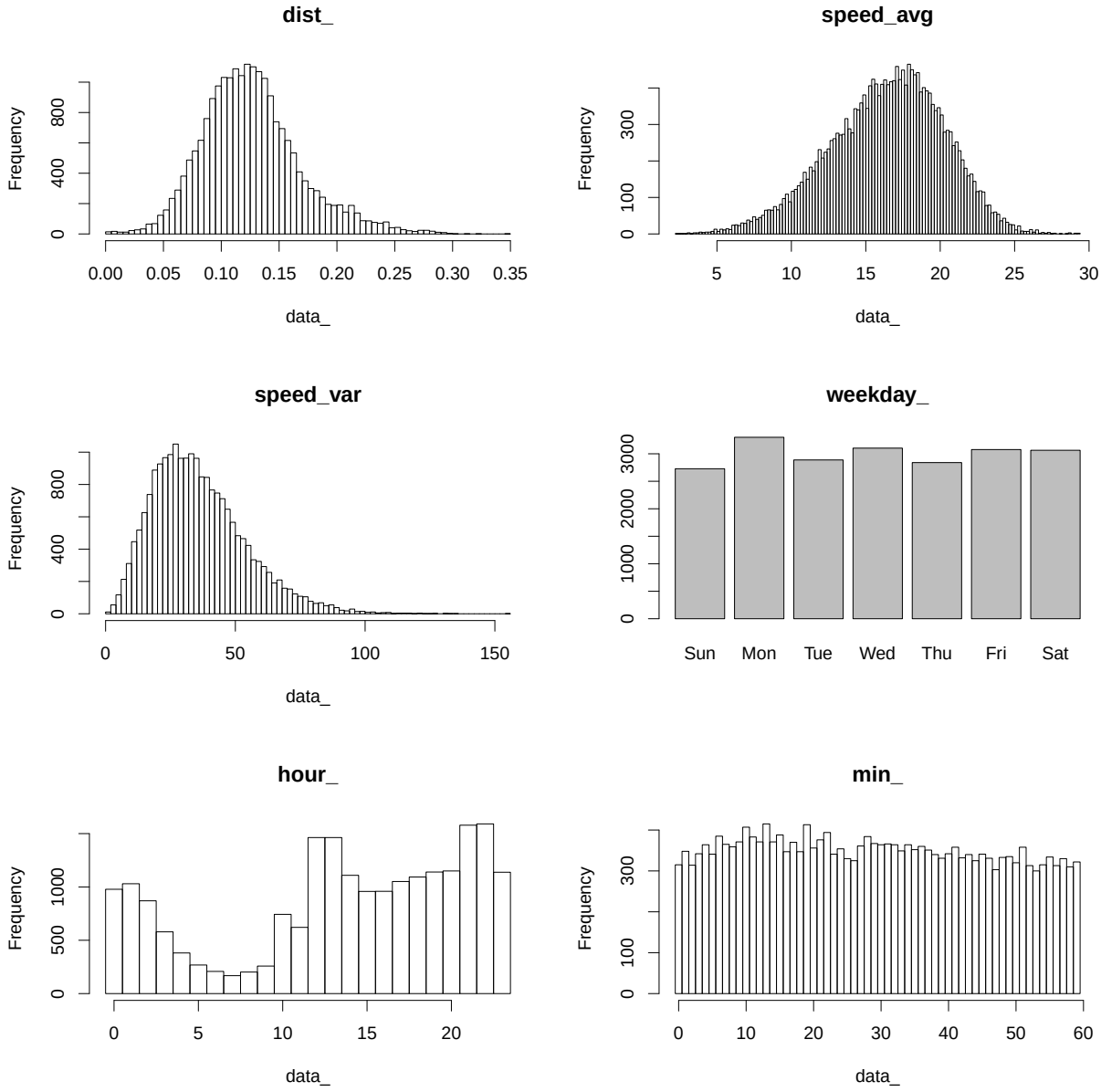
```
        speed_var = speed_var, weekday_ = weekday_,
        hour_ = hour_, min_ = min_, month_ = month_)
}, by = "trj_id" ]
save( dataset, training_set, file = "Don/regression1.RData" )
```

Exploratory analysis

```
load( "Don/regression1.RData" )
set.seed(1)
n = nrow( summary_data )
training_set_id = sample( 1:n, n * 0.75 )
training_set = summary_data[ training_set_id ]
test_set = summary_data[ -training_set_id ]
```

In the next figure, we have the empirical distributions of each of our covariates.

```
par( mfrow = c(3,2), cex = 1.5 )
varlist = setdiff( names(training_set), c("timediff","trj_id", "month_") )
for ( v in varlist ){
    data_ = training_set[[v]]
    if ( is.numeric( data_ ) ){
        breaks = 100
        if ( v == "hour_" ) breaks = 0:24 - 0.5
        if ( v == "min_" ) breaks = 0:60 - 0.5
        hist( data_, main = v, breaks = breaks )
    } else{
        plot(data_, main = v)
    }
}
```



We could explore different distance measures. The world is not flat (possible?), so maybe the Euclidean distance is not great. Routes are also not straight lines, so maybe an L1 distance would be nice? Not sure. I think the best would be to find how long the road distance is between two points.

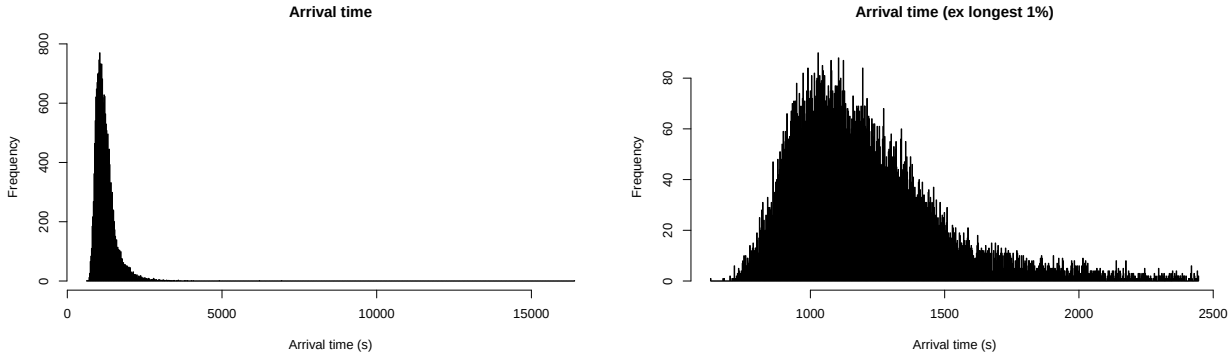
Could try logarithmic transformations for speed variance.

We see that there are fewer trips around 5am. Could be useful if we want to do a generalised least-squares solution. We could down-weight hours that are less frequent.

The next figure is the empirical distribution of the arrival times. There are some extreme outliers for some reason.

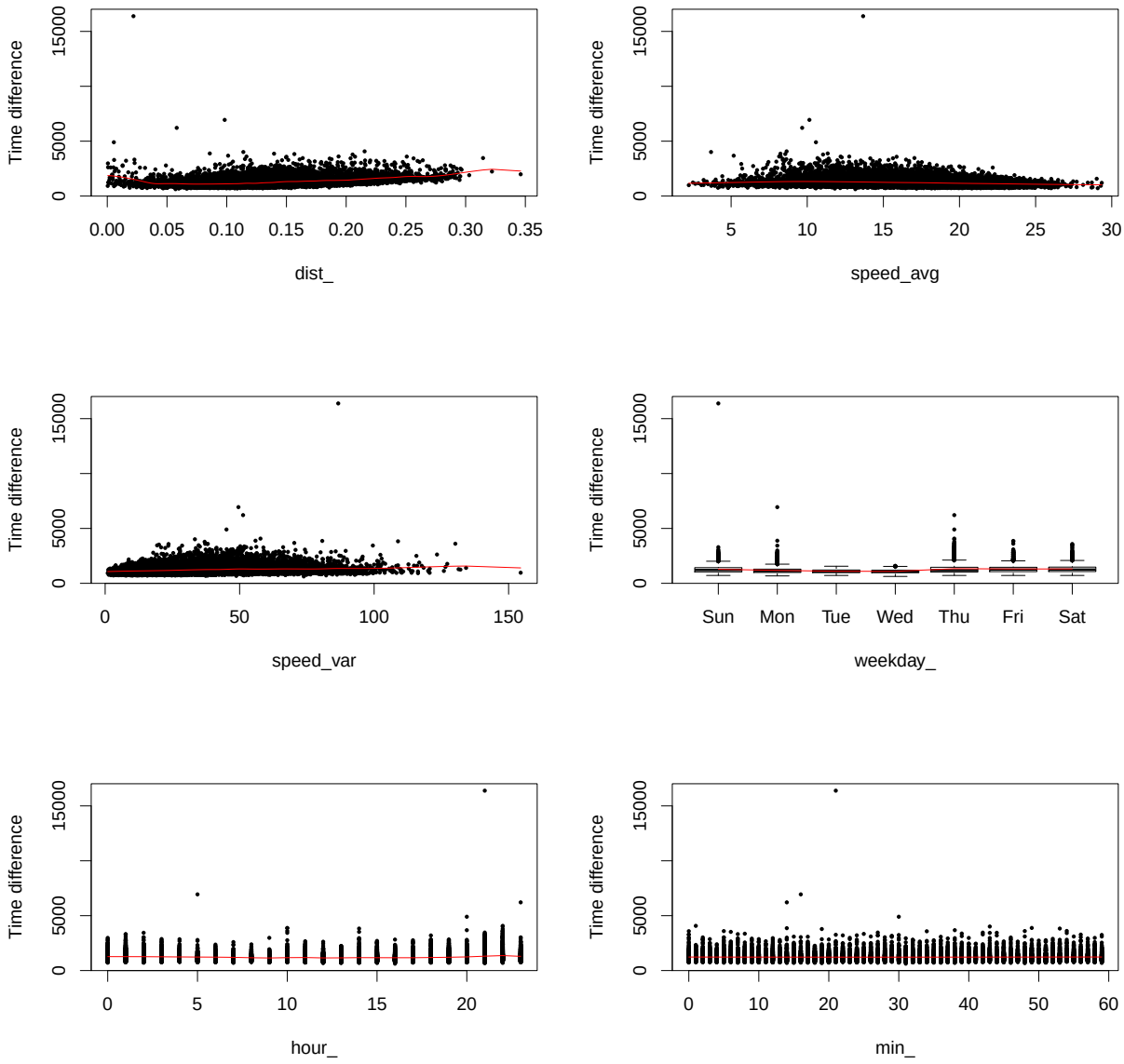
```
par( mfrow = c(1,2), cex = 1 )
hist( training_set$timediff, breaks = 1000, main = "Arrival time",
    xlab = "Arrival time (s)")
hist( training_set[timediff < quantile(timediff, 0.99)]$timediff,
```

```
    breaks = 1000, main = "Arrival time (ex longest 1%)",
    xlab = "Arrival time (s)")
```

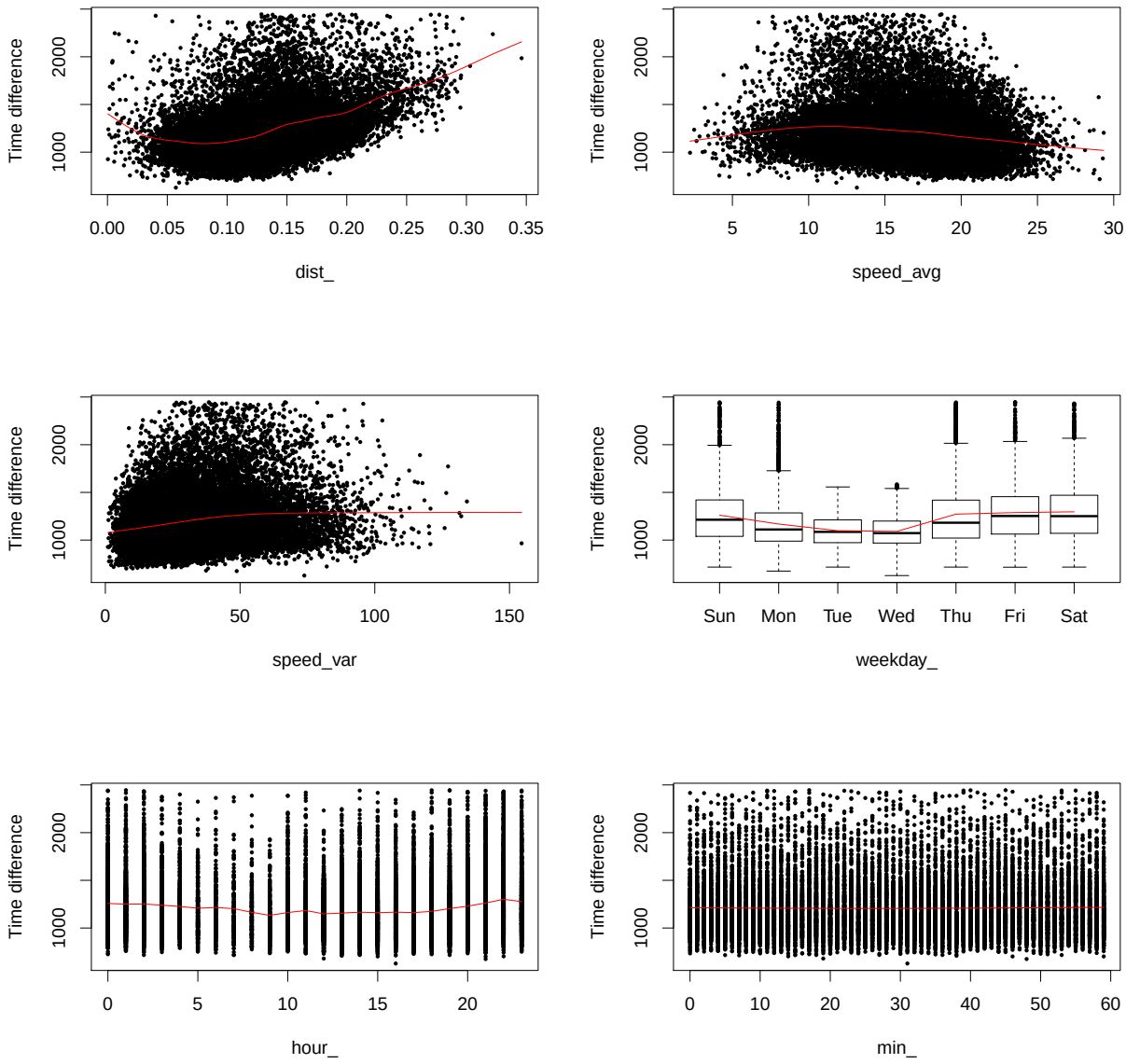


Next figure is a bivariate analysis. Again, the extreme outlier(s) really mess up the visualisation.

```
par( mfrow = c(3,2), cex = 1.5 )
for ( v in varlist ){
    plot( training_set[[v]], training_set$timediff,
        pch = 16, cex = 0.5,
        xlab = v, ylab = "Time difference")
    try({
    lines( smooth.spline( training_set[[v]],
        training_set$timediff ), col = "red" )
    })
}
```




```
par( mfrow = c(3,2), cex = 1.5 )
for ( v in varlist ){
    training_set[ timediff < quantile(timediff, 0.99), {
        var_ = get(v)
        plot( var_, timediff,
            pch = 16, cex = 0.5,
            xlab = v, ylab = "Time difference")
        try({
            lines( smooth.spline( var_,
                timediff ), col = "red" )
        })
        NULL
    } ]
}
```



Testing the model

Remove month because we only have one month in this subset. We have all the pairwise interactions.

```
training_set[ , month_ := NULL ]
training_set[ , trj_id := NULL ]
lm_ = lm( timediff ~ .*., training_set )
anova( lm_ )
```

```
## Analysis of Variance Table
##
## Response: timediff
##                        Df     Sum Sq    Mean Sq    F value    Pr(>F)
```

```
## dist_                   1  340582953 340582953  5582.0269 < 2.2e-16 ***
## speed_avg               1  717342861 717342861 11756.9804 < 2.2e-16 ***
## speed_var               1   20283414  20283414   332.4375 < 2.2e-16 ***
## weekday_                6   56639717   9439953   154.7173 < 2.2e-16 ***
## hour_                   1    1091656   1091656    17.8918 2.348e-05 ***
## min_                    1      20237     20237     0.3317   0.56468
## dist_:speed_avg         1   47415794  47415794   777.1271 < 2.2e-16 ***
## dist_:speed_var         1      81869     81869     1.3418   0.24673
## dist_:weekday_          6    6849920   1141653    18.7113 < 2.2e-16 ***
## dist_:hour_             1    1151865   1151865    18.8786 1.400e-05 ***
## dist_:min_              1     217570    217570     3.5659   0.05899 .
## speed_avg:speed_var     1      57254     57254     0.9384   0.33271
## speed_avg:weekday_      6   49193054   8198842   134.3759 < 2.2e-16 ***
## speed_avg:hour_         1    4913420   4913420    80.5291 < 2.2e-16 ***
## speed_avg:min_          1       5597      5597     0.0917   0.76200
## speed_var:weekday_      6    6333073   1055512    17.2994 < 2.2e-16 ***
## speed_var:hour_         1    4597250   4597250    75.3472 < 2.2e-16 ***
## speed_var:min_          1     120893    120893     1.9814   0.15926
## weekday_:hour_          6   17599856   2933309    48.0758 < 2.2e-16 ***
## weekday_:min_           6     598388     99731     1.6346   0.13306
## hour_:min_              1      50249     50249     0.8236   0.36415
## Residuals           20948 1278125649     61014
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

The ANOVA table suggests that the minute that the trip starts is not relevant.

Just with a simple model, our R^2 is about 50%, which is pretty good for a large dataset and the little work that I've done so far.

```
summary(lm_)
```

```
##
## Call:
## lm(formula = timediff ~ . * ., data = training_set)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
##  -645.0  -128.2   -31.6    83.4 15489.9
##
## Coefficients:
##                          Estimate Std. Error t value Pr(>|t|)
## (Intercept)             8.693e+02  6.335e+01  13.722  < 2e-16 ***
## dist_                   8.858e+03  3.351e+02  26.432  < 2e-16 ***
## speed_avg              -1.960e+01  3.302e+00  -5.937 2.95e-09 ***
## speed_var               4.595e+00  6.674e-01   6.886 5.91e-12 ***
## weekday_Mon            -8.575e+00  4.901e+01  -0.175 0.861111
## weekday_Tue            -1.842e+02  5.067e+01  -3.635 0.000279 ***
## weekday_Wed            -2.075e+02  4.980e+01  -4.166 3.11e-05 ***
## weekday_Thu             3.823e+01  5.217e+01   0.733 0.463678
## weekday_Fri            -1.748e+02  4.969e+01  -3.519 0.000435 ***
## weekday_Sat             3.506e+01  5.128e+01   0.684 0.494216
## hour_                   1.133e+01  1.816e+00   6.241 4.44e-10 ***
## min_                   -4.261e-01  7.703e-01  -0.553 0.580219
## dist_:speed_avg        -2.240e+02  1.115e+01 -20.086  < 2e-16 ***
## dist_:speed_var        -1.402e+01  3.088e+00  -4.540 5.65e-06 ***
```

```
## dist_:weekday_Mon      6.998e+02  2.001e+02   3.497 0.000472 ***
## dist_:weekday_Tue     -1.235e+03  2.431e+02  -5.078 3.85e-07 ***
## dist_:weekday_Wed     -1.126e+03  2.372e+02  -4.749 2.06e-06 ***
## dist_:weekday_Thu      2.210e+03  2.009e+02  11.000  < 2e-16 ***
## dist_:weekday_Fri      8.456e+02  1.861e+02   4.545 5.53e-06 ***
## dist_:weekday_Sat      9.539e+02  1.872e+02   5.095 3.52e-07 ***
## dist_:hour_            6.885e+01  7.888e+00   8.729  < 2e-16 ***
## dist_:min_             6.411e+00  3.165e+00   2.026 0.042797 *
## speed_avg:speed_var    7.090e-02  3.473e-02   2.041 0.041238 *
## speed_avg:weekday_Mon -1.294e+00  2.595e+00  -0.499 0.617999
## speed_avg:weekday_Tue  1.414e+01  2.751e+00   5.141 2.76e-07 ***
## speed_avg:weekday_Wed  1.435e+01  2.727e+00   5.261 1.44e-07 ***
## speed_avg:weekday_Thu -2.559e+01  2.650e+00  -9.659  < 2e-16 ***
## speed_avg:weekday_Fri  1.159e+00  2.699e+00   0.429 0.667674
## speed_avg:weekday_Sat -7.969e+00  2.667e+00  -2.988 0.002807 **
## speed_avg:hour_       -1.040e+00  9.641e-02 -10.791  < 2e-16 ***
## speed_avg:min_        -3.725e-02  3.746e-02  -0.995 0.319970
## speed_var:weekday_Mon -1.694e+00  3.767e-01  -4.498 6.90e-06 ***
## speed_var:weekday_Tue -2.101e+00  3.954e-01  -5.314 1.08e-07 ***
## speed_var:weekday_Wed -2.095e+00  3.868e-01  -5.415 6.18e-08 ***
## speed_var:weekday_Thu -2.132e+00  4.023e-01  -5.299 1.18e-07 ***
## speed_var:weekday_Fri  6.786e-01  3.829e-01   1.772 0.076381 .
## speed_var:weekday_Sat -9.001e-01  3.757e-01  -2.396 0.016594 *
## speed_var:hour_       -1.203e-01  1.504e-02  -7.996 1.35e-15 ***
## speed_var:min_        -8.466e-03  5.952e-03  -1.422 0.154951
## weekday_Mon:hour_     -7.965e+00  9.510e-01  -8.376  < 2e-16 ***
## weekday_Tue:hour_      1.112e+00  1.001e+00   1.111 0.266389
## weekday_Wed:hour_      1.079e+00  9.536e-01   1.131 0.258082
## weekday_Thu:hour_      8.551e+00  1.005e+00   8.510  < 2e-16 ***
## weekday_Fri:hour_     -1.256e+00  8.993e-01  -1.397 0.162439
## weekday_Sat:hour_     -2.497e+00  9.385e-01  -2.660 0.007814 **
## weekday_Mon:min_       8.138e-01  3.803e-01   2.140 0.032370 *
## weekday_Tue:min_       7.652e-01  3.956e-01   1.934 0.053081 .
## weekday_Wed:min_       7.372e-01  3.901e-01   1.890 0.058765 .
## weekday_Thu:min_       2.795e-01  3.961e-01   0.706 0.480425
## weekday_Fri:min_       5.511e-01  3.836e-01   1.437 0.150772
## weekday_Sat:min_       9.866e-01  3.832e-01   2.575 0.010040 *
## hour_:min_            -1.325e-02  1.460e-02  -0.908 0.364153
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 247 on 20948 degrees of freedom
## Multiple R-squared:  0.4994, Adjusted R-squared:  0.4982
## F-statistic: 409.8 on 51 and 20948 DF,  p-value: < 2.2e-16
```

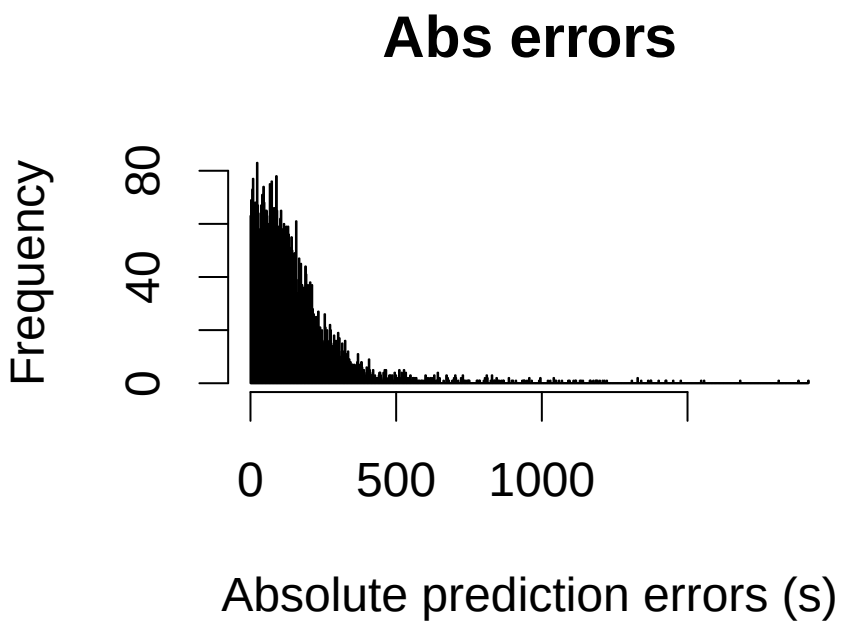
Our RMSE is about 220 seconds. So, about 3mins, 40 seconds on average.

```
yhat = predict( lm_, test_set )
rmse = sqrt( mean( (yhat - test_set$timediff)^2 ) )
rmse
```

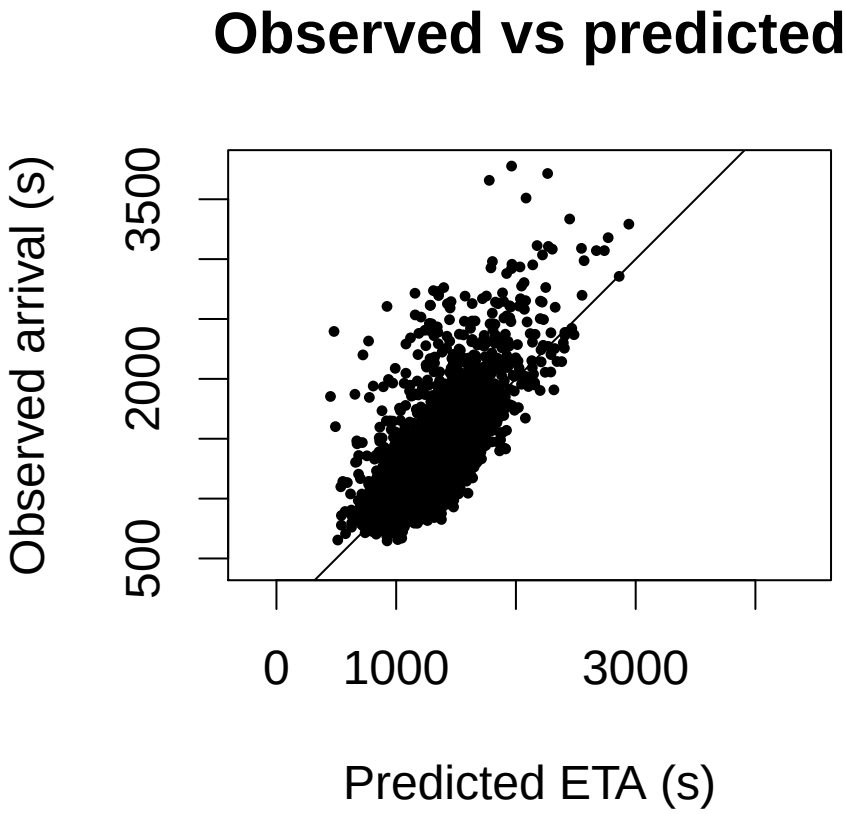
```
## [1] 217.3413
```

```
par( cex = 1.5 )
hist( abs(yhat - test_set$timediff), main = "Abs errors",
    xlab = "Absolute prediction errors (s)",
```

```
    breaks = 1000 )
```




```
par( cex = 1.5 )
total_range = range( c( yhat, test_set$timediff ) )
plot( yhat, test_set$timediff, pch = 16, cex = 0.5,
    main = "Observed vs predicted",
    xlab = "Predicted ETA (s)",
    ylab = "Observed arrival (s)", asp = 1,
    xlim = total_range, ylim = total_range
    )
abline( 0, 1 )
```



Taking the distribution of errors and the observed vs preidction plots together, it suggests that our basic model fails mostly because of under-prediction. There are a couple of very long journeys over short distances (e.g., fat loop around the city). Not sure how to handle this with the basic model.