bayesian_example

Don Li

03/06/2020

Load stuff and functions

```
load( "Don/regression1.RData" )
difftime_mins = function( time1, time2 ){
    x = difftime( time1, time2, units = "secs" )
    as.numeric(x)
}
round2 = function(x, dp){
    round( x/dp ) * dp
}
```

Bayesian model proof of concept

Demonstrate a Bayesian approach with just one trip as an example.

Processing steps

We need to divide the city into grids.

```
grid_step = 0.01

plot_data = dataset[ trj_id == 10, {
    latrange = range( round2( rawlat, grid_step ) )
    lngrange = range( round2( rawlng, grid_step ) )

    latgrid = seq(latrange[1]-grid_step*1, latrange[2]+grid_step*1, by = grid_step )
    lnggrid = seq(lngrange[1]-grid_step*1, lngrange[2]+grid_step*1, by = grid_step )

    list( latrange = list( latrange ), lngrange = list( lngrange ),
        rawlng = list(rawlng), rawlat = list(rawlat),
        latgrid = list( latgrid ), lnggrid = list( lnggrid ) )
    } ]

plot_trj = function( data ){
    data[,{
        plot( rawlng[[1]], rawlat[[1]], type = "o",
            ylim = range(latgrid[[1]]), xlim = range(lnggrid[[1]]) )
        abline( h = latgrid[[1]] + grid_step/2 )
        abline( v = lnggrid[[1]] + grid_step/2 )
    } ]
}
```

Process the time spent in each grid. This is equivalent to the number of pings in each grid.

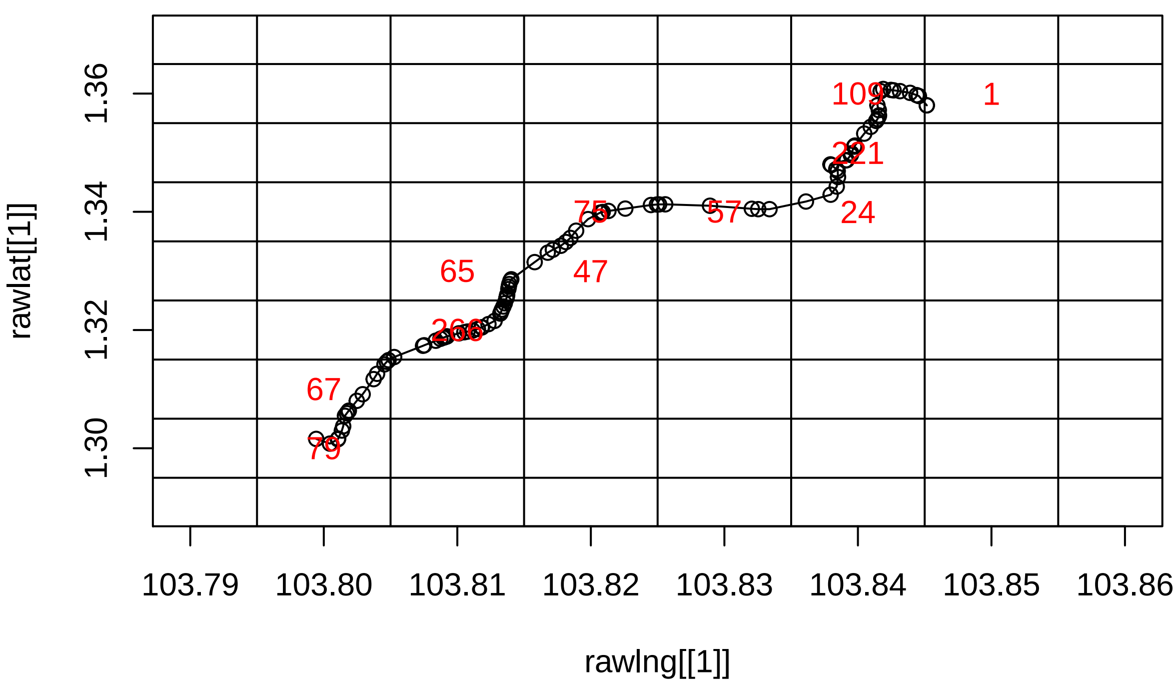
```
one_trip = dataset[ trj_id == 10 ]
one_trip[ , c("gridlat", "gridlng") := {
    grid_lat = round2( rawlat, grid_step )
    grid_lng = round2( rawlng, grid_step )
    list( gridlat = grid_lat, gridlng = grid_lng )
} ]
time_spent_in_grid = one_trip[ , {
    list( time_spent = difftime_mins( date[.N], date[1] ) )
    }, by = c("gridlat", "gridlng") ]
```

Visualise

```
plot_trj(plot_data)
```

```
## NULL
```

```
time_spent_in_grid[ , {
    text( y = gridlat , x = gridlng  ,
        time_spent, col = "red" )
} ]
```




```
## NULL
```

Make a random Bayesian model

Just a random model, doesn't really matter.

```r
cat( "
model{
    for ( i in 1:length(time_spent) ){
        time_spent[i] ~ dgamma( 10, 5 )
        time_pred[i] ~ dnorm( time_spent[i], 1/5^2 ) T(0,)
    }
}
", file = "jags_model.txt" )
jm = jags.model( "jags_model.txt", data = time_spent_in_grid, n.adapt = 100 )
```

```
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 11
##    Unobserved stochastic nodes: 11
##    Total graph size: 29
##
## Initializing model
```

```r
x = data.table( coda.samples( jm, "time_pred", n.iter = 5000 )[[1]] )
```

Label the average time spent in each square. The blue numbers are the posterior predicted means for time spent in each square.

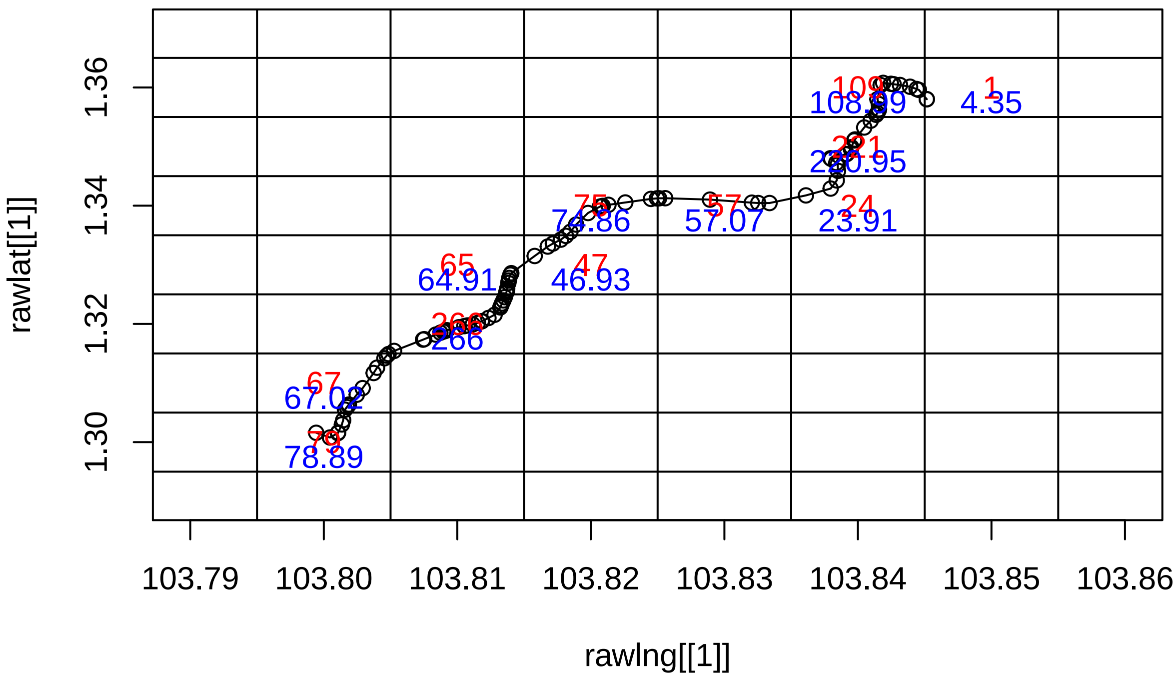
```r
plot_trj(plot_data)
```

```
## NULL
```

```r
time_spent_in_grid[ , {
    text( y = gridlat , x = gridlng  ,
        time_spent, col = "red" )
} ]
```

```
## NULL
```

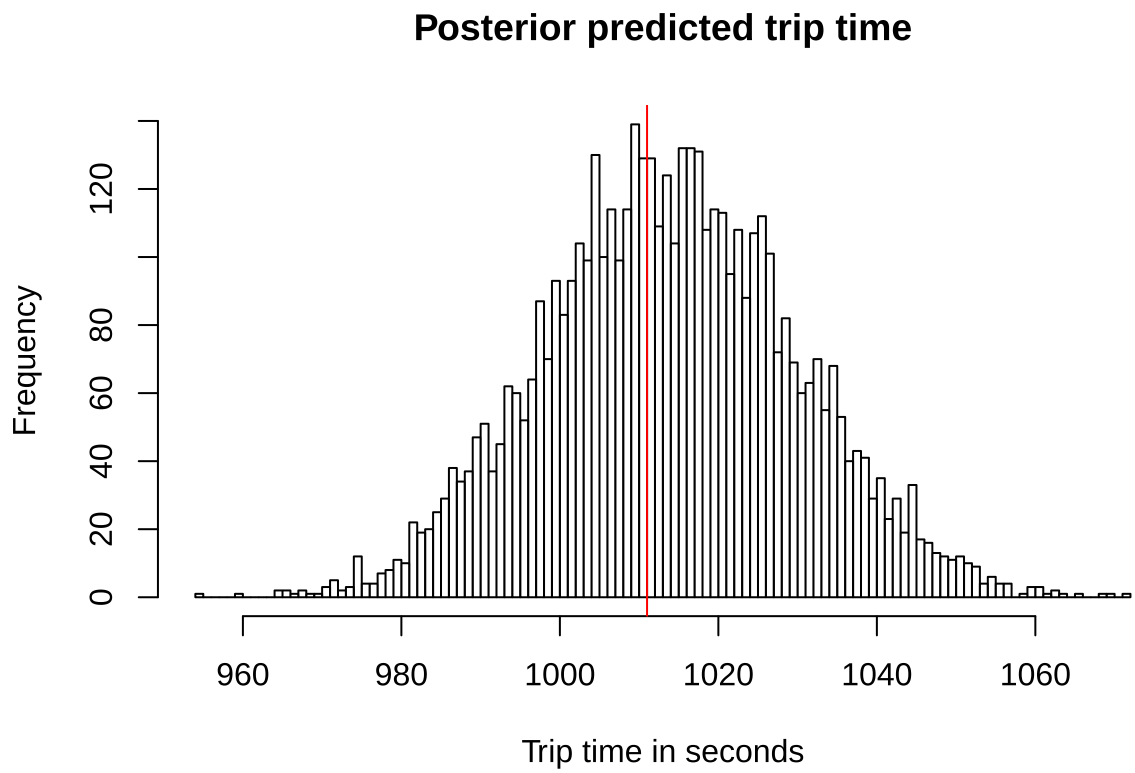
```r
time_spent_in_grid[ , {
    text( y = gridlat  - 0.0025, x = gridlng  ,
        round( colMeans( x ), 2 ), col = "blue" )
} ]
```




```
## NULL
```

Overall prediction is just the sum of all the travel times. The posterior predicted travel time is just the sum.

```r
hist( rowSums( x ), breaks = 100,
    main = "Posterior predicted trip time",
    xlab = "Trip time in seconds" )
abline( v = time_spent_in_grid[ , sum(time_spent) ], col = "red" )
```



Problems and things to do:

- Transition between squares can be problematic. In this one example, we lose about 100 seconds due to transitions between grid squares. Could be improved with smaller grids?
- Need a more reasonable probability model model. Could be based on sampling points?
- Need to process the data into squares.