

Report

Artur Sahakyan

2023-08-07



Figure 1: Alice in Wonderland, Card soldiers as arches

Introduction

Presented with the task of computing Yerevan's walk score, our group started carrying out meetings. During the first 2 weeks we formed general idea of what walk score methodology used, what prioritized amenities they had and the discrete values they used as weights.

Each member of our group was assigned a task during this brainstorming process. Shoghik would be responsible for learning Shiny apps and constructing initial template, Sharlot would research taxes of Yerevan, Elen would study ggplot2, Gevorg would keep searching for documentations whenever needed. I would be responsible for algorithms.

Algorithm for Walk Score

After a week of thinking during walking, sports training sessions or waiting near coffee house I finally formed an initial idea of computing the walk score.

Let's represent Yerevan as a playing stage, where we simulate people walking. People have states - "not walking", "moving from A to B", "moving from B to C" and so on. Let's add to this probability of moving from "not walking" to "moving from A to B" and in this manner from each state to others. States connect all the points we queried earlier. Exponential distribution and specific decay rates (by amenity type and amenity's importance in real life) were used to compute those transitional probabilities. Below a sample chunk of transitional matrix is given.

```
print(transition_matrix[2:4,2:4])

##           [,1]      [,2]      [,3]
## [1,] 0.00000000 0.03049335 0.02920693
## [2,] 0.03050148 0.00000000 0.02962005
## [3,] 0.03006235 0.03047944 0.00000000
```

But this is not sufficient. Players have to randomly walk between points in order to replicate reality. Players have initial state vectors which alter transition matrix when simulations begin. Inside state vectors we have rates of success/(number of simulations) which was shown during presentation in form of cards game (choosing random card for 1 simulation, say spades, dividing number of found spades over the count of deck, suppose $[3/52, 1/52, 10/52, 0, 0, 1/52, \dots]$). R's functions came to help for those computations.

```
## Called inside a loop for random walk

next_point <- sample(1:n_points, 1, prob = transition_matrix[current_point, ])

### Called inside a loop for checking successes of random walk

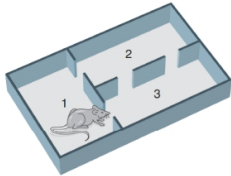
success_count <- sum(replicate(n_simulations, random_walk(start_point, end_point, steps = 30)))
```

Additionally, to provide academic structure to my algorithm, I would like to show "Rat Problem" from the lecturer Mher Martirosyan's slides. (Markov Chains are vividly shown here)

Markov Chains. Part 1
oooooooooooooooo●oooo

Example: Rat Problem

A psychologist places a rat in a cage with three compartments, as shown in the figure below. The rat has been trained to select a door at random whenever a bell is rung and to move through it into the next compartment.



(a) If the rat is initially in compartment 1, what is the probability that it will be in compartment 2 after the bell has rung twice? three times? (b) In the long run, what proportion of its time will the rat spend in each compartment?

Figure 2: Rat Problem

New tools have been added - such as interactive calculation based on user's square size input, also visualization based on user's square size input, note please that ggplot()'s square size input is little bit different from actual square division, taking into account ggplot()'s specifics

The geospatial points were queried from *openstreetmaps*, using *osmdata* and R. The query code is stored in "WalkScore\query_algo.R". Using *dplyr* new columns were added to specify amenity type and its decay rate. The distance function can be easily replaced by google's *mapdist()* function, which will provide more precise data on distances, using google roads API.

There is an issue that I have noticed regarding *openstreetmaps*. The library *osmdata* has its servers either shut down or in works in some periods of year and functionality may be unavailable for a while. This is a significant challenge, since in maps it is convenient to call bounding box of city by name, or query of points *depends solely on osmdata*.

Example of problematic function

```
getbb("Yerevan")
```

```
## error 405, http request error
```

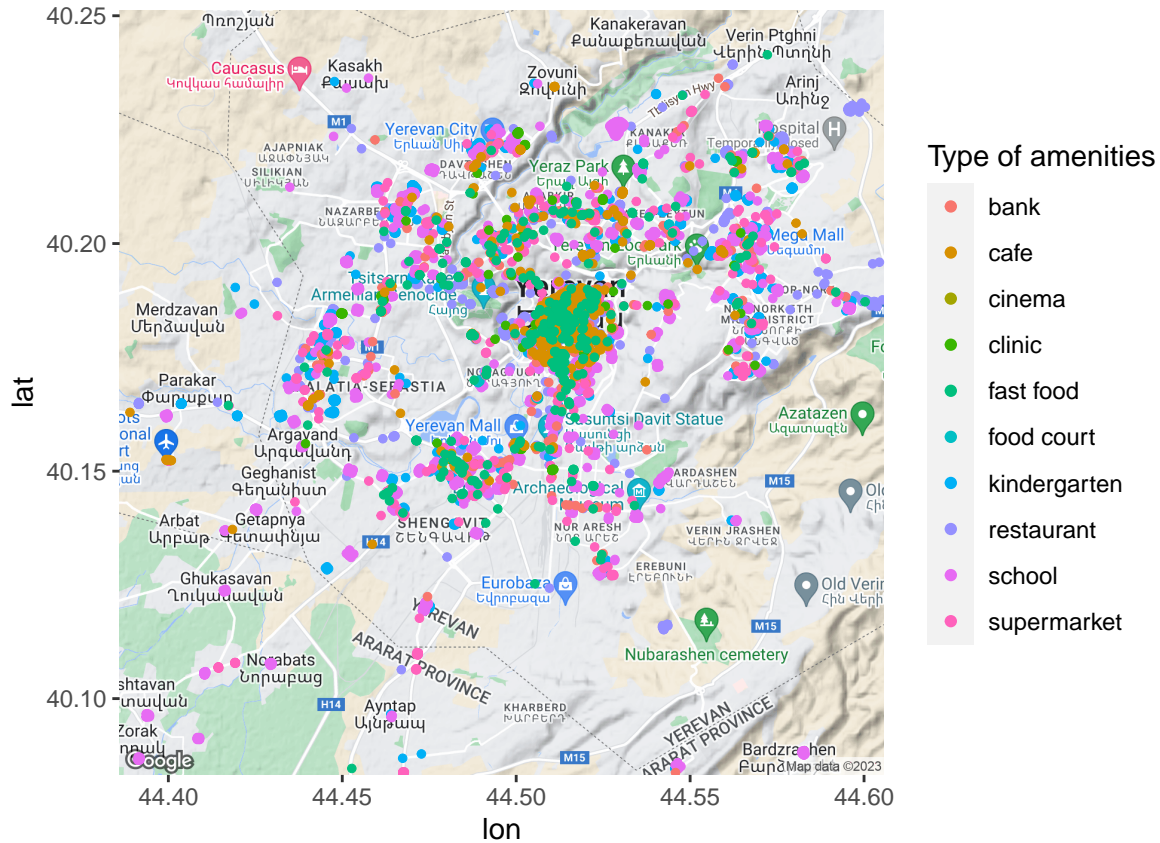
```
## Problem due to osmdata servers
```

```
ymin <- 44.39080 ## 44.36211
ymax <- 44.60057 #44.62177
xmin <- 40.10177 ##06585
xmax <- 40.23295 ##40.24177
yerevan_map <- get_map(location= c(lon = mean(c(ymin, ymax)), lat = mean(c(xmin,xmax))),
                        zoom =12, maptype="terrain", scale = 2)
```

```
## i <https://maps.googleapis.com/maps/api/staticmap?center=40.16736,44.495685&zoom=12&size=640x640&scale=2
```

```
ggmap(yerevan_map)+
  geom_point(data = final_data, aes(lat,lon, color = as.factor(type), stroke=0.001 )) + labs(c
```

```
## Warning: Removed 104 rows containing missing values (`geom_point()`).
```



For algorithms refer to “algorithm_fun.R” for use case on “final_data_addition.csv”, also in Shiny App’s interactive calculator tool code.

Map My Walk

The “mapmywalk” data we had was provided to Sharlot Abkarian. She turned .tcx files into excel sheets, grouped walks by sessions. After that I used dplyr to obtain delta-time in seconds for each row and the distance passed, after which units were converted to km/h for providing velocity column. Given the velocity, overall quantiles of data were compared with individual quantiles, which classified walking types. The most common way of walking is being in rush, the second one is relaxed walk. Meanwhile, brisk walk(used in fitness) is very uncommon.

Taxes

Analyzing Yerevan Taxes was insightful. Looking at data and its quantiles, it became evident that Kentron district was leading by the number of businesses, while Shengavit was paying almost the same size of total tax amount as Kentron district. First, we had to show the taxes geospatially. Clearing the data with dplyr, then giving the Kentron’s taxes yellow color created the necessary contrast. Although points are transparent, the number of observations is quite large, which creates the intense plot. Afterwards, boxplots using plotly inside shiny were added. Given the plots we were able to analyze data and make hypothesis clear.

Conclusion

To summarize, our main aim for this project was to develop the concepts of walk score methodology, advance it with more elaborate techniques of computation and introduce new algorithms. In addition, we wanted to have useful plots and interactive tools, that could help statisticians in their analysis. Other than that, regular users were taken into account, when for instance having colorful squares on map or telling about

algorithm in form of cards game.

We aimed to tell a story with the help of plots and analysis.

And hope this program will make a useful contribution to its users' life. Sincerely, Artur, Elen, Shoghik, Sharlot, Gevorg.