Solving TSP using R in context of {sf} & HERE API

Jindra Lacko

2022-10-17

## 1 Nové Trenky

My Alma Mater, [Prague School of Economics](https://www.vse.cz/english/), is located in Žižkov. A formerly working class neighborhood, now rather gentrified, it has to this day retained some traces of its former rougher edges. One of these is an active night life.

A crawl through the bars of Žižkov is therefore a familiar activity for many VŠE students, and can serve as a gateway drug for serious optimization techniques. Such as the Travelling Salesman Problem (1).

The TSP is an optimization classic, with a number of well understood and highly standardized solutions available in the context of statistical programming language R.

In this blog post I would like to share a practical example of solving the TSP using Open Street Map (2) data of bars via {osmdata} (3) and HERE routing engine via {hereR} (4). The actual solution will be found by utilizing the {TSP} (5) package.

The first step in our exercise is acquiring data of Žižkov bars. A search is performed over the area of *core Žižkov*, defined as a polygon, using the OSM Overpass API. As there seems not to be a clear consensus over what constitutes a bar, restaurant or a pub in Prague I am including all three of the possible amenities.



We have located 74 bars, implying a distance matrix of 5476 elements. Not a huge one by today’s standards – but big enough to think twice about trying to solve using a pen and a piece of paper.

I have found that while it is not overly difficult to solve the TSP for *all* the Žižkov bars there is educational value in running the TSP over only a small sample. I have found it advantageous to be able to actually show the distance matrix – and this page will easily accommodate only about a 5×5 matrix.



The easiest distance matrix to calculate is plain “as the crow flies” distance. This can be calculated via a sf::st\_distance() call.

The resulting matrix will be based on pure distance, with some differences in interpretation depending on coordinate reference system of underlying data (Euclidean in projected CRS and spherical in unprojected CRS).

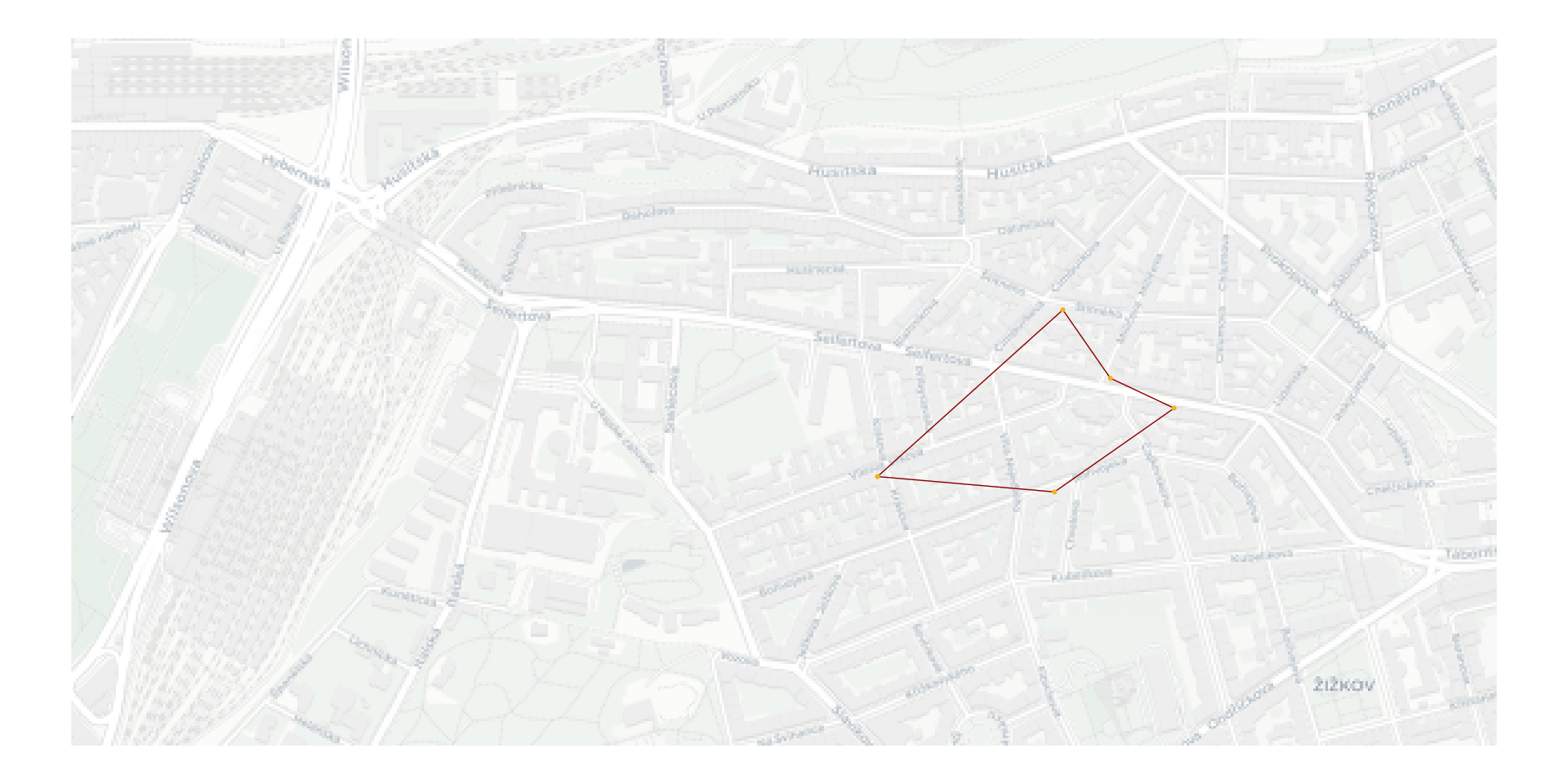
Calculating the distance matrix using plain distance is easy, and the resulting matrix is symmetrical (distance from A to B equals distance from B to A). It is also hollow (distance from A to A itself is zero).

Solving the TSP for such a matrix is straightforward, as the hard work has been outsourced to the {TSP} package internals.

The optimal route will thus be:

[1] "Chocobamba" "Blind Eye" "U Habásků"   
[4] "Pivní lokál Ostrý" "U Pádivce"

Once we have the optimal route calculated it can be visualized using {leaflet}. The sequence of stops needs to be completed (by repeating the first stop after the last) and cast from points to a linestring.



From the visual overview we can see an obvious shortcoming of the “as the crow flies” approach: it completely ignores other constraints except for distance – such as the road network.

Thus while the route shown is “optimal” in the sense that it forms the shortest path joining the five bars selected, it is not one that we could actually follow (unless we were a flying crow).

This shortcoming can be resolved by using an alternative distance matrix as input, while retaining the techniques of {TSP} for the actual route selection. A possible source of more applicable data are routing engines, available to R users via API interfacing packages.

The routing results give us several pieces of data:

* the routes as linestring objects in EPSG:4326 (for visualization later on)
* distance of the route (in meters)
* travel time (in seconds) both raw and adjusted for traffic
* petrol consumption

To these I have added three technical columns: indices of start & destination for easier joining of solved TSP results back and the name of the route as string for visualization purposes.

Having a variety of metrics will be helpful in construction of alternative distance matrices.

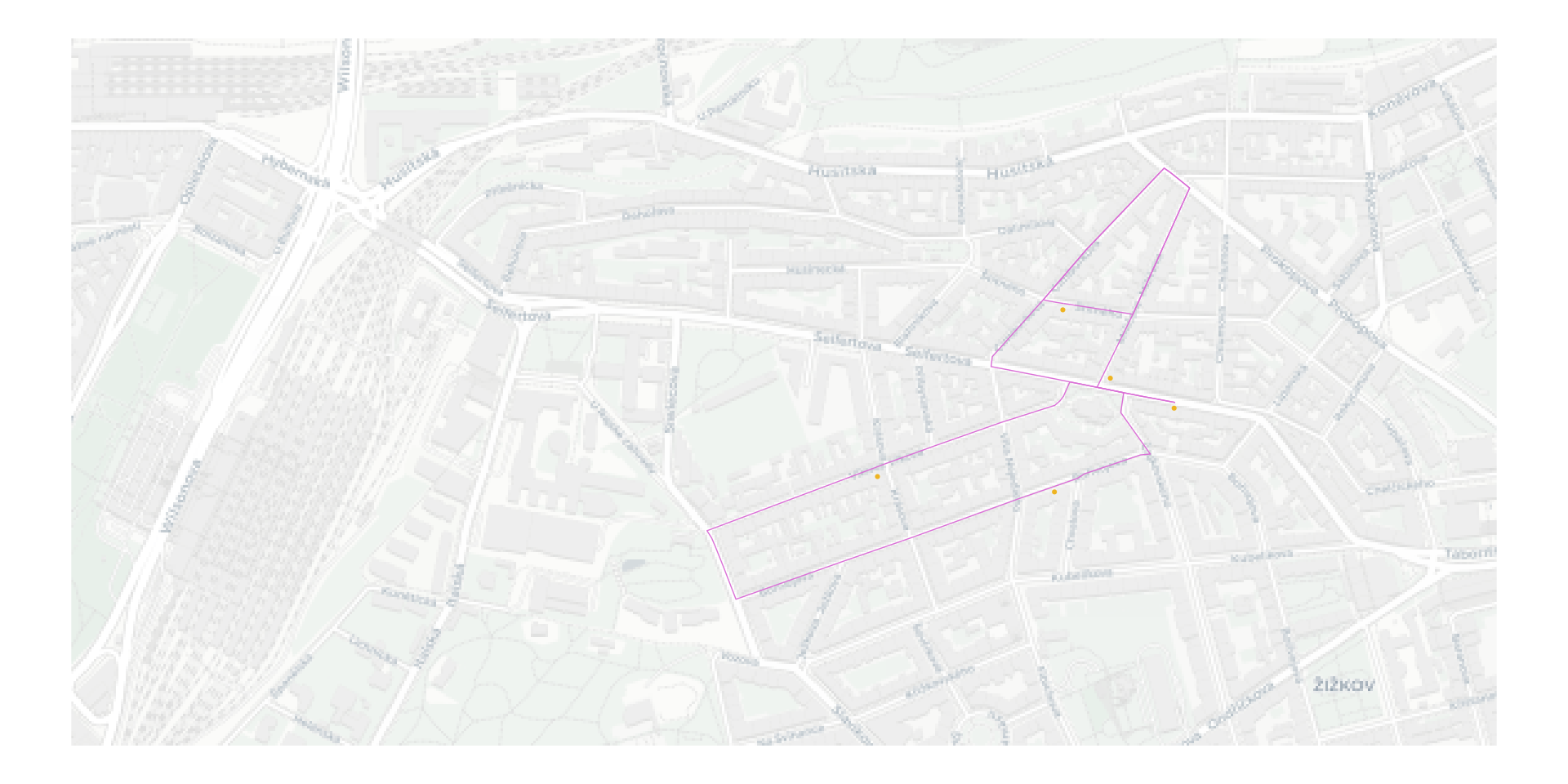
The first routing distance matrix will be based on route distance; notice that while the matrix is hollow it is not symmetrical. This is not surprising, as routing is not commutative – optimal route from A to B need not be the same as from B to A, due to constraints such as one way roads. Žižkov is a veritable warren of one way streets.

It will need to be declated as asymetrical to {TSP} solver; other than that the actual process of solving the matrix will be analogical to the “as the crow flies” matrix.

[1] "Pivní lokál Ostrý" "U Pádivce" "Blind Eye"   
[4] "Chocobamba" "U Habásků"

Once we have solved the TSP and figured the sequence of “cities” to visit it is time to report our results.

For this purpose it is advantageous to prepare a data frame of indices of start and destination, and join it back with the original dataset from HERE API (which contains routes as linestrings).



Since the HERE API is generous in terms of results provided it is not difficult to construct an alternative distance matrix, using a different metric. This could be either trip duration or petrol consumption.

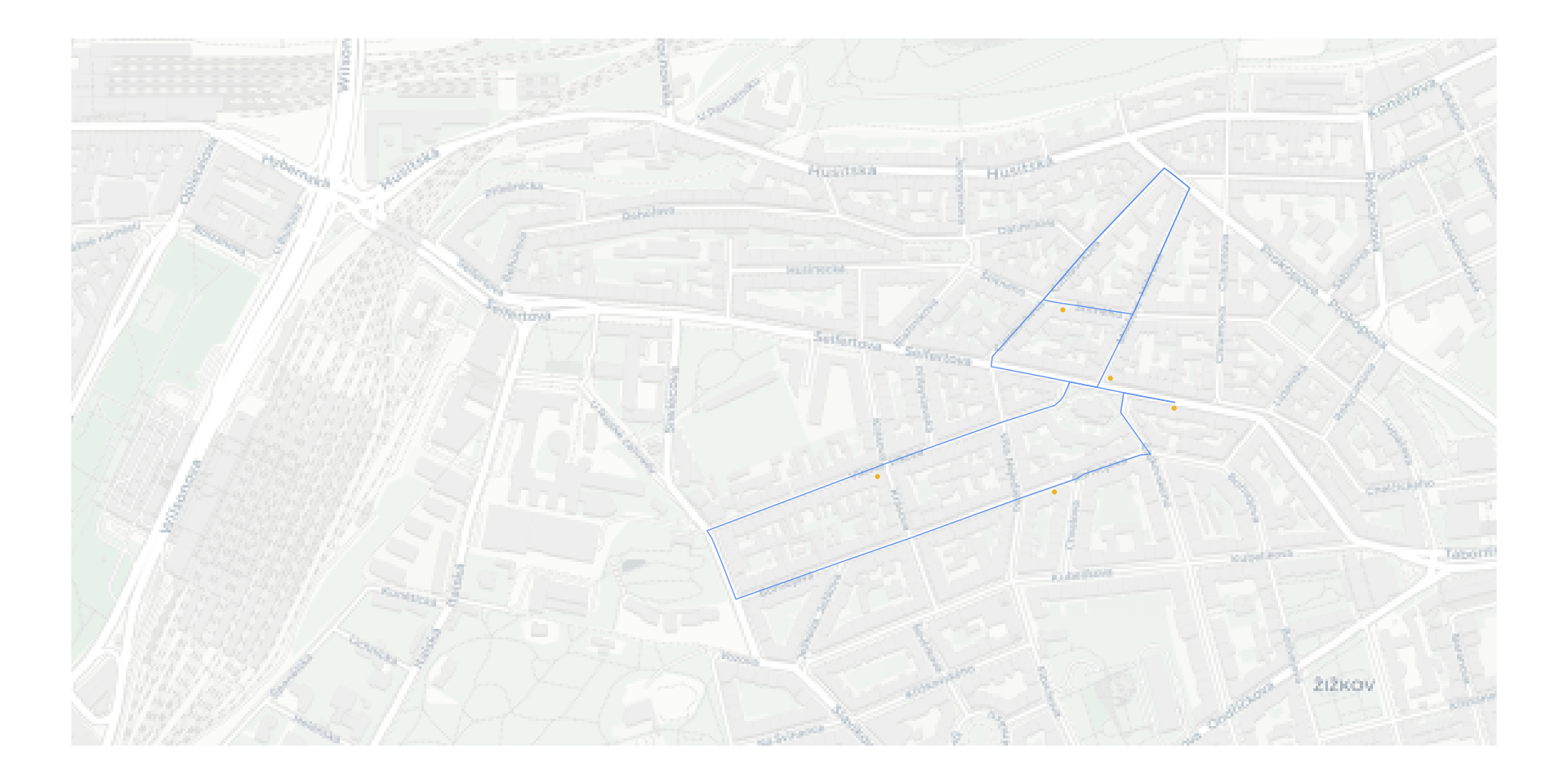
In our specific situation both of these can be expected to be be highly correlated with the plain distance results. All the streets in Žižkov are of very similar type, and the average speed & consumption are unlikely to vary greatly between the routes.

The most significant difference between the distance and time based TSP will be driven by current traffic, which is a factor HERE routing engine considers.

[1] "Pivní lokál Ostrý" "Blind Eye" "Chocobamba"   
[4] "U Pádivce" "U Habásků"

Once we have solved the trip duration optimized TSP we again need to report the results; in our use case the output is very similar to the distance based one.

This will not necessarily be the case in other contexts, especially ones with greater variation of road types (city streets vs. highways).



I believe my tongue in cheek example has succeeded in showing two things:

* the ease of applying a standardized solution (the {TSP} package) to a well known and well understood problem (the Travelling Salesman Problem) within the context of R ecosystem
* construction of distance matrices from HERE API routing results, with option to optimize for multiple metrics (such as minimizing the travel distance, travel time and petrol consumption)

1. GUTIN, G. and PUNNEN, A. P. *The Traveling Salesman Problem and Its Variations*. Springer Science & Business Media, 2006. ISBN 978-0-306-48213-7.

2. OPENSTREETMAP CONTRIBUTORS. *Planet dump retrieved from https://planet.osm.org*. Online. 2017. Available from: <https://www.openstreetmap.org/>

3. PADGHAM, Mark, LOVELACE, Robin, SALMON, Maëlle and RUDIS, Bob. osmdata. *Journal of Open Source Software*. Online. 26 June 2017. Vol. 2, no. 14, p. 305. [Accessed 7 October 2022]. DOI [10.21105/joss.00305](https://doi.org/10.21105/joss.00305).

4. UNTERFINGER, Merlin and POSSENRIEDE, Daniel. *hereR: “sf”- Based Interface to the “HERE” REST APIs*. Online. 0.9.0. R. 5 August 2022. [Accessed 7 October 2022]. Available from: <https://CRAN.R-project.org/package=hereR>

5. HAHSLER, Michael and HORNIK, Kurt. TSP—Infrastructure for the Traveling Salesperson Problem. *Journal of Statistical Software*. Online. 2008. Vol. 23, p. 1–21. [Accessed 7 October 2022]. DOI [10.18637/jss.v023.i02](https://doi.org/10.18637/jss.v023.i02).