Final Assessment GIS

Arthur Maenhout - UCFNAMA

1/3/2019

## Part One

For this assessment I used different data sources, all form the open data portal of the City Bruges which can be found here: [OpenDataBrugge](https://www.brugge.be/opendata).

At the beginning I was not sure which data I wanted to map this because I had no idea what insights the data could give me. Therefore I started to with QGIS and imported almost every interesting file that is available on the website. To get a first look at the data, in my opinion, it is easier to use QGIS then R. This because in QGIS you can create quite fast some nice maps where it takes longer to create a map in R.

The most important file is a base layer of the different district in Bruges, I used this one on top of the openstreets layer to point out the districts. Because this map was very (to) detailed I used the dissolve function in QGIS (with the provided “sectorname” as unique field) to create a new shapefile with the different sectors in Bruges. Now I could use two shapefiles (districts and sectors). The other data files I used in both maps are:

* The road accidents from 2014 – 2016 (KML file - QGIS)
* Population per district (csv file - RStudio)
* Age of inhabitants per district (csv file - RStudio)

For both maps I used quite a different approach on how to map the valuable data. Therefore I will review both workflows separately.

In QGIS I started to map the accident file and the sector file. The accidents where illustrated with points inside the sector (polygons), which was very unclear (chaotic). In the dataset the year of the accident was stated which gave me the option to create new layers for every year (filtered and saved the new layer). By using the points per polygon built-in function, I could extract the amount of accidents per sector. I did this for every year separately and the total accidents, after I joined each file based on their sector name, this gave me a table with the sectors, accidents in 2014, 2015, 2016 and the total accidents. I used the total accidents to color the different polygons and used the different year accidents to create histograms.

In RStudio I used the shapefiles created using QGIS, this because I could not find a solution to use KML files directly into R. I started with the population data which I appended with the different districts. After doing some data cleaning (drop Na rows) and data expansion (add total population), I aggregated the dataframe with the Sectornames as unique value, and took the sum of the other numeric values (Male, Female, Total, Surface). Now I could append the sector shapefile in order to be able to plot the values. Hereafter I did some data enriching calculated:

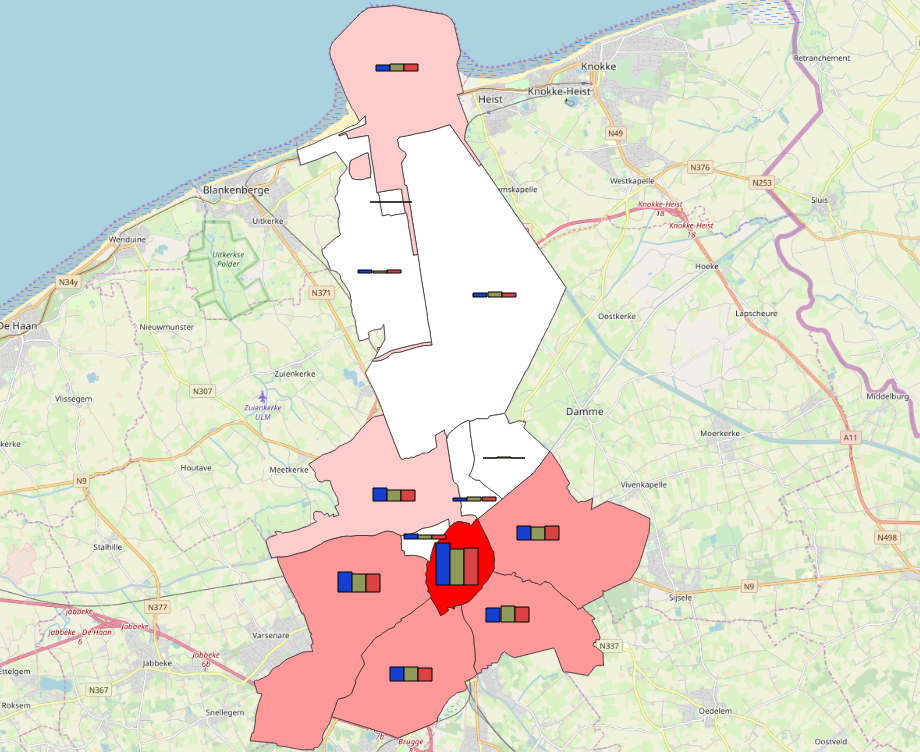
* Total Males in Bruges
* Total Females in Bruges
* Total population

Which made it possible to calculate:

* Rate of Males
* Rate of Females
* Rate of Population
* Population density

Besides the use of the population I added the age of the population per sector. After creating bins manually, I appended the district file, aggregated on the “sectorname”, and appended the sector file. Therefore, I could use the different bins to see how diverse the ages per sector in Bruges are.

After using both programs, I think that QGIS is an interesting tool get the first insights in your data. Also, to create new layers and filtering is QGIS quite fast. But when you get to the point to create maps and especially enrich your data, I would propose RStudio. Since the command line also makes it possible to go back to previous step and redo or change them which is clearly harder in QGIS.

## Part two

For this second part of the final assessment I received data regarding the “famous” CASA treasure hunt. In this hunt, the goal is to gain as much points as possible by visiting different attractions throughout the city. Each participating group chose its own path, some groups recorded their path to analyze theirs in this part; My group did not record our path, therefore the data of the winning team of last year was used. After doing the first assessment in QGIS and R and attend the lectures got really into R. Besides I think that it is easier to get around in a new tool as QGIS or ArcGis then learning a new programming language. So I choose to work in Rstudio for this part.

I started with importing the Path of the winning team, the file is a geojson which can easily be imported with R as a SpatialLineDataFrame. I choose to convert it to a simple feature (SF) data frame because there are some interesting benefits of working with a SF data frame. The most important one is the high number of functions to use. For example, the st\_length calculates the distance of the line (path), with st\_buffer you can draws a buffer around the line. Also,the TFL data source was imported and transformed to a simple feature data frame.

Apart from st\_length and the buffer also the st\_intersection function is used to calculate the amount of stations that were passed within a 100-meter radius. This result can then be used to filter the stations in order to map only these passed stations. In order to get the info about the London wards, I used the LondonWardShapefile to have a good base layer. To answer some questions, more information about the different wards was added (Life expectancies, population, ….). Some questions could be answered on more than one way, I would almost choose for the easiest to understand way/function. Because otherwise If someone want to reuse the code or you want to reproduce it for some other projects It could take a while to understand it.

All maps that are produced in this part are done with the tmap package. In most lectures we used this package, but also online blogs and resources showed that tmap is easy and fast to use. I used tmap to show the hunt path, stations and so on, but after almost every piece of code I would create a map to show the results. This was somehow my base for my sequence of steps to answer the questions. Each map show what the previous lines of code would change to the results.

Overall it took me some time to get the different files into the right types (Sp, Sf, …). I had troubles with transforming the types of the dataset. Which led to frustrations but after looking at previous practical’s, finding solutions on google or asking help from other students I got to the end of the second part.

## Part 3: Mini Project

### Introduction

For this final mini project part of the assessment I wanted to create a shiny app tool in order to create some awareness in Belgium about the use of realtime datasets and improve my knowledge of R. As I started my first part, I noticed that there is a lack of interesting and useful available open datasets in the city of Bruges. The data that is available is categorized in 5 different areas: population, land use, environment, mobility and work and economy. These datasets are all static data that are enriched with the newest at the end of a Quarter or the year. Therefore, I sent an email to some people at different departments in Bruges to get realtime or some more static data but unfortunately, they could not help me because they just do not have any more datasets. Since other cities in the world share so much realtime data I wanted to create something with a live dataset to show to cities and inhabitants from Belgium how this realtime data can be used in real-life cases.

When one thinks of realtime shared data, this is often about transport and mobility. Companies like Waze, Google, Uber and more have a lot of data available which is used to direct people through cities and can be used by developers to create own application based on these datasets. But multiple worldwide cities also share their public transport data to citizens, which creates an enormous number of apps and tools that are created.

For this project was mostly a trial and error project. I tried a lot of different datasets which leaded to some different attempts. First, I’ll explain why I chose to work with R and Shiny, what the basic datasets is about. Aftr, two attempts will be discussed because they both the base of my final product.

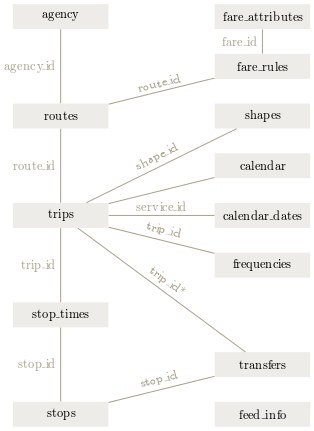
### R and shiny

First of all, the mini project is completely ran in Rstudio using R. This for the same reasons as part 2. I still felt (feel) eager to learn R because every week in the lectures and at home I learned new things that can be used in daily tools.

The main goal of this project is to create a dashboard to show realtime data from the Public transport in Bruges. I hope that this shiny app can also be used on a daily basis for some citizens. I choose to create a shiny app because It seems to be a completely different way of showing data using dashboard and R. The link of the shiny app can be found here: [ShinyAppArthurMaenhout](https://amaenhout.shinyapps.io/shiny/). (there seems to be a problem with the realtime data on the website, but it is working in Rstudio when running App.R (zipped file))

### General Transit Feed Specification (GTFS)

My project is based on GTFS data, GTFS defines a common format that is used in cities and countries. It includes transportations schedules and geographic information. The static part is composed of text files which can be collected in a ZIP file. Each of these files have information about a particular aspect of transit: stops, routes, trips, stop times, and other data. Figure 1 show the full dataset with the relationships. This static data is general for every city which makes it handy to use in different cities. The realtime part of GTFS is not that general across cities. Each city or country choses which information they want to push to the GTFS feed. When cities choose to open this live data is allows users to efficiently plan their trips based on current arrival and departure times.



GTFS Class diagram

### Attempt 1

Belgium is divided in three highly autonomous regions (Flanders, Brussels and Wallonia), defined on a territorial basis but each of these regions have their own language economy history and landscape. Each of this regions has his own public transport company. For busses this is De Lijn in Flanders, TEC in Wallonia and MIVB in Brussels. While the NMBS/ SNCB (Nationale Maatschappij der Belgische Spoorwegen, translated; National society of Belgian railways) is responsible for the Belgian transport by rails. My first focus was on the Belgian railway system is known for the delays that occur for almost every trip. (Treinen opnieuw minder stipt | De Tijd, no date) Therefore I wanted to create a map that would show the realtime location of the trains which could be used by commuters to plan their trips. Besides the NMBS I also wanted to include some information about the busses in Belgian cities.

The NMBS shares static and realtime info about their rail routes on their website. The Static data can be downloaded quite as a ZIP file while the realtime information is shared as a protocol buffer. I started with importing the static text files in R as data frame and exploring them further. Before adding realtime data, I wanted to have a look at the bus data that was shared by De Lijn, using their API’s I could get a lot of information that could be used. after adding some functions, it was possible to produce a basic Shiny app which showed a map of Belgium with the different routes and station. This map could be filtered by a user through text, select input and a time range.

After adding the static data of the NMBS and De Lijn I wanted to start with realtime data that is available on the NMBS website. Realtime data is shared as a protocol buffer, which is a “language-neutral, platform-neutral, extensible way of serializing structured data for use in communications protocols, data storage, and more.” (Developer Guide | Protocol Buffers, no date) Protocol buffers are mostly used by Java C++ and python developers which made it harder to find the right packages in R. After spending a long time on the internet and executing hundreds different lines of code I used python to get the first insight in the data. Here I saw that the shared data was a bit of a disappointment because it only had the minutes of delay of the trains. So, no realtime location of the trains. At this point I made the decision to pivot my idea to something else. Of course, I did not want to lose this already done work. That’s why I changed my location area from Belgium to the City of Brussels and the focus from railway and busses to busses and metrolines. This new focus will be described in the second attempt. But a lot of the functions and the sequence of step could be copied. So, the second attempt is actually using the same steps but other datasets.

### Attempt 2

The city of Brussels has a very wide spread of open transport data which is based on GTFS format and some other files. This data can be imported using an API and is divided in seven links and returned as a zip (orange) or json file (red), these links can be found in the project diagram.

For each link I created a function to connect with the API and get the asked datasets. I used a different approach for the zip files and json files. Zip files are saved and imported, json files are only imported when using the data.

#### Zip files

The two zip files hold a lot of basic information about the routes and the stations in Brussels. The GTFS data gives as mentioned static information about the planned routes in the city. I choose to use this static information to calculate some numbers (number of trips a day, etc). The shapefile is the base for mapping the routes and stations. After unzipping and reading the files, lines and stops, it seemed that the longitude and the latitude were not in a general known format, after doing some online research I found that it is a typical Belgian reference system; Lambert 1972 (EPSG: 31370). After converting it was possible to plot all the routes and stations in Brussels using the tmap package.

#### Json files

The json files are based on the realtime GTFS data, by adding different arguments to the URL, as route numbers or station numbers. You can get the information about the routes and stations, the information is return as json file but I chose to transform these to data frames when calling the URL functions (which are self-written). The different functions I used be found in the project diagram.

#### Results:

The result of this mini project is a working shiny app which includes two parts; One is focused on giving information about routes and the is about the stations, with some realtime information.

##### Route

The routes dashboard has his main focus on showing realtime positions of vehicles. Users can choose the kind of transport (Metro, Bus, Tram) method and the line number of the route they want to look at. The blue lines indicate the route of the line ( back and forth), grey dots all the stations, green dots location of vehicle with destination of green station and this is the same for the black dots for the blue station.

At the top right corner some basic informative number are shown about the selected line: - The amount of stations a single trip takes - The frequency of route on a day - The length of the total route - From where it leaves - To where it goes

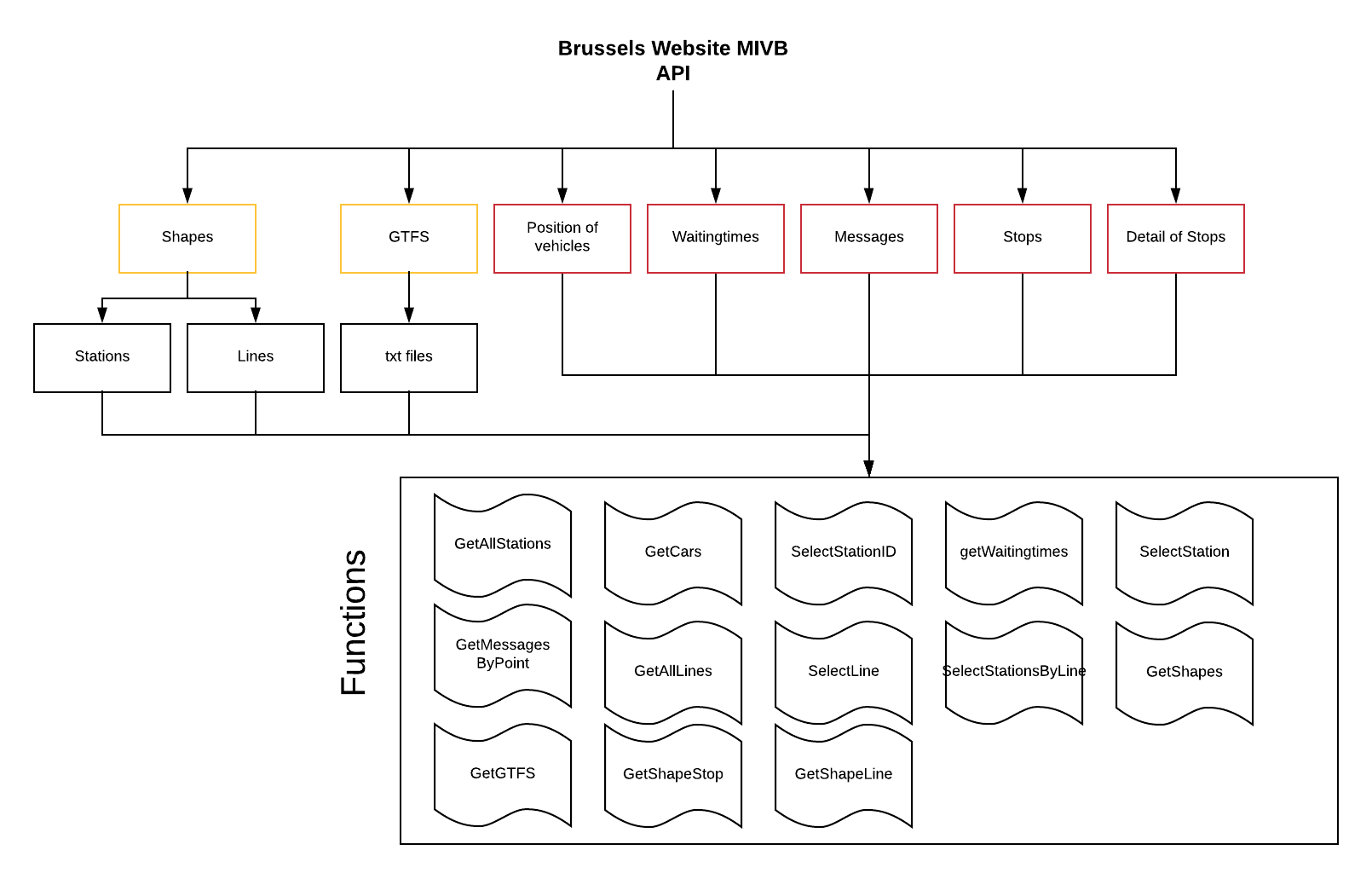
##### Stations

The station tab points out some more information about the stations in the city center. The first the map shows all the different stations for the selected transport method (Metro, bus and trams). Based on the filter the map will change again. You can enter or choose one or multiple stations. This will change the map to an overview of the chosen stations. In order to get messages and waiting times you need to choose the Station ID as well because the MIVB API has a problem when multiple id is asked. After choosing stations and stations ID, you’ll get a table with the line numbers the time when the vehicle will arrive and the destination. If there is a message about the station, for example when it is closed due to events, accidents etc., it will also be show here in a message box. Unfortunately (and luckily) there aren’t a lot of message so this will mostly be empty.

#### Discussion and conclusion

The different attempts show that I had to change my ideas multiple time, therefore the end product is not totally what I hoped for, but this product can be used by citizens that want to get realtime location and information about routes and stations in Brussels. The main goal for this project is achieved because with this dashboard, cities can be persuaded to change their mindset about opening data, especially realtime data.

But this product is clearly not an end product, there are still countless possibilities to take this product further. In the future, analysis could be added to the dashboard but also data enriching is necessary. For example, if it would be possible to add the number of commuters, price tickets this shiny app could show even more information to the public and city councils.



Project diagram of datasources, and created functions

## Bibliography

Developer Guide | Protocol Buffers (no date) Google Developers. Available at: <https://developers.google.com/protocol-buffers/docs/overview> (Accessed: 2 January 2019).

Treinen opnieuw minder stipt | De Tijd (no date). Available at: <https://www.tijd.be/politiek-economie/belgie/federaal/treinen-opnieuw-minder-stipt/10070723.html> (Accessed: 2 January 2019).