STN Project: Modeling Train Commuter Stress for Dutch Cities

Leon Overweel (s1837379)

Abstract

Many tech companies in the Netherlands have offices in central Amsterdam, but housing in this area is too expensive for many of their employees. This raises the question of where these workers can live while avoiding a daily commute that is too physically and mentally taxing. In this work, we model for which Dutch train stations it is least stressful to commute to Amsterdam Central Station. Our contributions include a commuting stress model, a method for scraping relevant data, and a method for calculating and visualizing the resulting stress scores. We conclude that several large Randstad cities as well as the area North-West of Amsterdam are good living locations for a commuter seeking a low-stress commute to central Amsterdam.

1. Introduction

What is the best place to live if you work at a tech company in central Amsterdam? Companies like Google, Netflix, Salesforce and Adyen have their Dutch headquarters in the Amsterdam area, but living in the city's "sizzling housing market" is becoming increasingly unaffordable for recent graduates [1]. Because many of these offices are within biking distance from Amsterdam's central train station[1], however, commuting to work is a viable option for these workers. On the other hand, it has also been shown that commuting can negatively impact a worker's stress levels and motivation [4].

Combined, the untenability of the Amsterdam housing market and the potential stress of a commute raise the following question: From which Dutch city is it least stressful to commute to an office near the Amsterdam Central train station every weekday?

To answer this question, the rest of this paper is structured as follows. In Section 2, we describe related work in the area of commuter stress; in Section 3, we explain our approach to modeling commuter stress in terms of the Dutch rail network as well as our data collection process; in Section 4, we show the results of the model and discuss the work and its impact.

2. Background

Dutch workers tend to commute more by train than by car over long distances; for short distances (under 3 kilometers), they tend to prefer walking or cycling over using a car or public transport [3]. Driving is the most stressful mode of transportation [2]; car commuters, specifically, show higher stress and more negative moods than train commuters [5]. Especially highly educated commuters tend to not commute by car [3]. These facts justify our focus on train-based commuting.

According to [2], the biggest factors that influence stress in a commute are the commuter's sense of *control* and *comfort*. Factors that make a person feel less in control of their commute include train delays and transfers [2]. Factors that make a person feel less comfortable during their commute include crowding and excessive heat and noise.

3. Approach

Based on the related work, we identify the following properties of a commute as influences on a worker's stress levels: crowdedness c, duration d, flexibility f, punctuality p, and number of train switches s (transfers). These properties are further explained in the Description column of Table 1.

From the literature, there is no canonical way to weigh these factors: some commuters find it more stressful to be on a crowded train while others are more uncomfortable having to switch trains at many stops. Therefore, we define the stress S for commuting from a given station to Amsterdam Centraal as Equation 1:

$$S = \min_{\forall o \in O} \left(\alpha_c c_o + \alpha_d d_o + \alpha_p (1 - p_o) + \alpha_s s_o \right) + \alpha_f (1 - f) \tag{1}$$

In this equation, the free weights αs allow us to penalize certain properties more than others, depending on the commuter's preferences. Since there are often multiple options (different trains to take) for a station, we calculate properties c, d, p and s for each of these options O; in the final stress calculation, we then consider the least-stressful option. Note that we add the inverse of f and p because they are negatively correlated with stress (e.g. more flexibility or punctuality leads to less stress), unlike the other properties, which are positively correlated with stress; see Table 1.

Using this model, we can assign a commuting stress level S to each train station in the Netherlands. Commuters can use these scores to compare how stressful it would be to commute from the different cities that these train stations serve as a factor in their decision where to live.

3.1. Data

To assign stress scores, we need to calculate f for each station, and d, c, p, and s for each commuting option from that station. This data is not trivially available, so the collection process described in this section represents a major part of the contribution of this work.

3.1.1. Station List

The Dutch national railway operator Nederlandse Spoorwegen (NS) provides an API[2] endpoint to retrieve an XML list of all stations currently in service in their network. The output of calling this API is available as `stations-raw.xml`.

[1] Amsterdam Centraal (ASD)

[2] NS APIs require users to pass basic authentication usernames and passwords with each API call (available on request from NS). We provide raw API outputs as datasets where appropriate so that other parts of this work can be easily replicated. More information: https://www.ns.nl/en/travel-information/ns-api

Property		Description	Measurement	Corr.
Crowdedness	c	How busy the train is	`LOW` (0), `MEDIUM` (0.5) or `HIGH` (1); maximum for each leg of the journey	+
Duration	d	How much time it takes to get to ASD	Minutes from first train departure to final train arrival, divided by 120	+
Flexibility	f	How many ways there are to get to ASD	Number of train options arriving between 8:00 and 9:00, divided by 8	−
Punctuality	p	Whether the train(s) will arrive on time	Probability of all legs of the journey arriving on time	−
Switches	l	How many times the commuter has to switch trains	Number of trains in the journey−1, divided by TODO	+

Table 1: The different properties of a commute to Amsterdam Centraal (ASD) and their impact on stress. Measurements are normalized to be between 0 and 1. A positive correlation (**Corr.**) means that a higher value for the property leads to higher stress; a negative correlation means the opposite.

In total, there are 625 train stations in the NS network. Using `stations.py`, we extract the following properties from the 405 of these stations that are located in the Netherlands and store them in `stations.csv`:

- **Name**: The station's full name, such as Amsterdam Centraal.
- **Code**: The station's ID, like ASD for Amsterdam Centraal.
- **Latitude, Longitude**: The station's geographic location.

3.1.2. Routes

The NS provides a public API for finding the best routes given source and destination stations, but this API does not return statistics for crowdedness and punctuality, which we require for computing c and p. The NS's website, however, does show this information. We can automatically scrape this data as follows.

The `reload-page.js` script, when run in the Chrome Developer Tools (Menu > More Tools > Developer Tools), successively loads the directions from every Dutch NS station to Amsterdam Centraal[3] in the current tab, with a three second wait between loads to make sure all data has been retrieved via the private API.

Since the directions page dynamically loads its data using JavaScript (so just retrieving the page through e.g. `curl` yields no useful data), we can access the relevant information by listening to the network traffic through the Network tab in the Developer Tools window. By filtering the network requests on `?reisadvies`, we get a table of network requests, each of which contains the complete travel advice (including crowdedness and punctuality for each travel option).

The Developer Tools window has no native way of saving the responses to multiple HTTP requests in one go. To solve this, we can recursively open another Developer Tools window which inspects our original Developer Tools window that is listening to the network traffic. The `scrape-responses.js` script, when run in this secondary Developer Tools window, will dump a large (14-million character) JavaScript object that contains the JSON response for each `?reisadvies` call to the console. We can save this response to a file.

Finally, using `routes.py`, we can extract the following properties from all the travel options to ASD that we got from the API:

- **Station**: The unique identifier for the starting station.
- **Option ID**: An index for this travel option (there are often multiple routes one can take from the starting station to ASD).
- **Depart**: The departure time; in `hour:minute`.
- **Arrive**: The departure time; in `hour:minute`.
- **Minutes**: The travel time in minutes (= `arrive` − `depart`).
- **Legs**: The different parts of the journey, separated by spaces; e.g. `AH->UT UT->ASD` represents two legs: one from Arnhem to Utrecht, and one from Utrecht to Amsterdam.
- **Crowd**: How crowded each leg is; separated by spaces; possible values are `HIGH`, `MEDIUM`, `LOW`, and `UNKNOWN`.
- **Punctuality**: The percentage that each leg of the journey arrives on time; separated by spaces.

These are stored in `routes.csv`. This is all the data we need to compute stress scores S for each station.

3.2. Process

Using the data collected in Section 3.1, the definitions in Table 1, and Equation 1, we can calculate stress scores S for each station. We implement this in the `get_scores` function in `scores.py`, which takes the αs as argument. Some implementation notes:

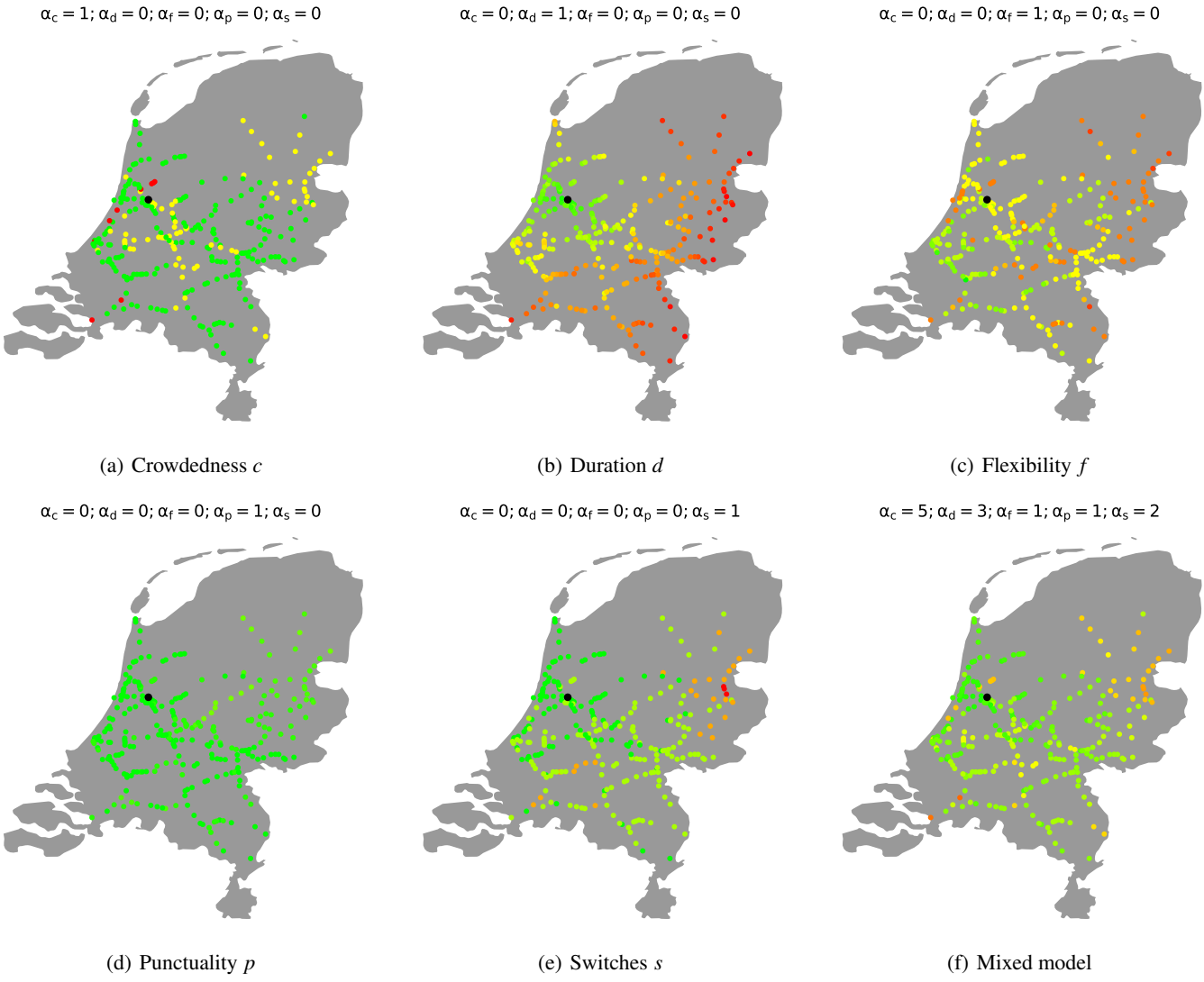
- For calculating crowdedness c, sometimes the API returns `UNKNOWN`; since (from inspecting the data) there are many more `LOW`-crowdedness trains than `MEDIUM` or `HIGH` ones, we assume `LOW` crowdedness for `UNKNOWN`.
- In calculating the duration d, we divide by 120; this is to normalize the data using the longest-allowed journey time of two hours.
- In calculating flexibility, we divide by 8; this is to normalize the data using the most flexible station in the data.
- In calculating punctuality, we assume 100% punctuality for journey legs that do not have this information available. To calculate the overall punctuality for a journey option, we multiply the punctuality of each leg together (assuming independence).

Then, using `map.py`, we can generate a map of the Netherlands that shows the score of each station, from the worst-scoring station (high stress, in red) to the best-scoring station (low stress, in green). Stations are not shown if they cannot reach Amstredam Centraal via a journey that arrives sometime between 8:00am and 9:00am and is less than two hours long.

4. Results and Conclusion

The results are shown in Figure 1, which plots each viable train station (that can reach ASD between 8:00am and 9:00am within two hours), colored by its stress level S, as definied in Equation 1. Unsurprisingly, duration and number of switches increase roughly by

[3] Setting arrival time to 9:00 on Monday, November 13, 2018.



(a) Crowdedness c (b) Duration d (c) Flexibility f

(d) Punctuality p (e) Switches s (f) Mixed model

Figure 1: Stress maps with different settings for αs; green indicates low stress and red indicates high stress. Amsterdam Central Station (ASD) is marked in black. The mixed model uses the author's preferences as α settings.

distance from ASD. Flexibility is highest in the Randstad megapolis (consisting of Amsterdam, Rotterdam, The Hague, and Utrecht; identifyable on the map roughly as the kite shape where Amsterdam forms the top corner), but this area also has relatively high crowdedness. Punctuality is generally high across the board.

Looking at our mixed model, which emphasizes in decreasing order crowdedness, duration, number of switches, flexibility, and punctuality, the area North-West of Amsterdam looks especially attractive. Table 2 (in the appendix) shows the thirty stations with the highest scores on this mixed model. Unsurprisingly, the top few stations are within the greater Amsterdam area. More interestingly, large cities such as Leiden, Rotterdam, Amersfoord and Hilversum also make the top thirty. All of these are very attractive commuting options for workers going to Amsterdam Centraal.

4.1. Ethical Considerations

In this work, we scraped data from a private API that is explicitly not available in the public version of this API, which may raise ethical concerns. In weighing these, we take into account that the data provider, NS, has received large government subsidies. We believe that data generated by publicly-funded organizations should be available to the public, especially for non-commercial use. We therefore believe that scraping the private API is justified.

4.2. Future Work

This model for commuter stress can be extended in several ways for it to be more useful to a person deciding in which city they want to live. For example, we could also consider housing availability and pricing[4]. We could also consider the other side of the commute: how flexibly can the person get home after work? Especially for younger workers, having the ability to take a late-night train home might be important. Finally, a natural extension to this work would be to apply it to different destinations than Amsterdam Central Station; using Section 3 and the code, this should be easy for cities within the Netherlands.

4.3. Conclusion

In this work, we have created a model for which Dutch train stations it is least stressful to commute to Amsterdam Central Station. Our contributions include a stress model, a method for scraping the required data, and a method for plotting the resulting stress scores. We conclude that several large Randstad cities as well as the area North-West of Amsterdam are good living locations for a commuter seeking a low-stress commute.

[4]Such data is available here: https://www.cbs.nl/nl-nl/publicatie/2015/52/demografische-kerncijfers-per-gemeente-2015.

References

[1] Hermse, John and Munsterman, Ruben. Sizzling amsterdam housing market pushes people to other cities, Jun 2018. URL https://www.bloomberg.com/news/articles/2018-06-18/sizzling-amsterdam-housing-market-pushes-people-to-other-cities.

[2] Legrain, Alexander, Eluru, Naveen, and El-Geneidy, Ahmed M. Am stressed, must travel: The relationship between mode choice and commuting stress. *Transportation Research Part F: Traffic Psychology and Behaviour*, 34:141–151, 2015. ISSN 13698478. doi: 10.1016/j.trf.2015.08.001. URL http://dx.doi.org/10.1016/j.trf.2015.08.001.

[3] Susilo, Yusak O. and Maat, Kees. The influence of built environment to the trends in commuting journeys in the Netherlands. *Transportation*, 34(5):589–609, 2007. ISSN 00494488. doi: 10.1007/s11116-007-9129-5.

[4] Wener, Richard, Evans, Gary, and Boately, Pier. Commuting stress: Psychophysiological effects of a trip and spillover into the workplace. *Transportation Research Record: Journal of the Transportation Research Board*, (1924):112–117, 2005.

[5] Wener, Richard E. and Evans, Gary W. Comparing stress of car and train commuters. *Transportation Research Part F: Traffic Psychology and Behaviour*, 14(2):111–116, 2011. ISSN 13698478. doi: 10.1016/j.trf.2010.11.008. URL http://dx.doi.org/10.1016/j.trf.2010.11.008.

A. Running the code

The following code files are included with this report:

- `reload-page.js`
- `scrape-responses.js`
- `stations.py`
- `routes.py`
- `scores.py`
- `map.py`

The JavaScript code can be run in Chrome as described in Section 3.1. Because these scripts take quite a bit of fine-tuning and time to run, we recommend using their output (`routes-raw.json`) directly with the Python scripts instead of re-scraping the data.

Running the Python scripts is easiest from a conda environmentv with the following packages installed:

```
conda activate
conda install numpy
conda install -c conda-forge cartopy
```

To create a map like the ones in Figure 1 and a ranking like in Table 2, run the scripts in the following order.

Run `stations.py` to create `stations.csv` from `stations-raw.xml`:

```
python3 stations.py
```

Run `routes.py` to create `routes.csv` from `routes-raw.json`:

```
python3 routes.py
```

Run `map.py` to calculate scores and create the map and ranking:

```
python3 map.py
```

You can tweak the settings for the αs in `map.py`; the map and ranking will be stored n the `data/plots` and `data/rankings` folders respectively, with a file name corresponding to the α settings.

Rank	**Code**	**Name**	**Stress** S
1	ASDM	Amsterdam Muiderpoort	0.0438
2	ASS	Amsterdam Sloterdijk	0.0438
3	ASA	Amsterdam Amstel	0.0458
4	ASDL	Amsterdam Lelylaan	0.0521
5	HWZB	Halfweg-Zwanenburg	0.0646
6	SHL	Schiphol Airport	0.0667
7	DVD	Duivendrecht	0.0687
8	HLM	Haarlem	0.0708
9	ASB	Amsterdam Bijlmer ArenA	0.0729
10	HLMS	Haarlem Spaarnwoude	0.0750
11	HFD	Hoofddorp	0.0792
12	KZ	Koog aan de Zaan	0.0798
13	ALM	Almere Centrum	0.0814
14	HAD	Heemstede Aerdenhout	0.0854
15	ZZS	Zaandijk Zaanse Schans	0.0861
16	HLO	Heiloo	0.0875
17	WM	Wormerveer	0.0923
18	KMA	Kromennie-Assendelft	0.0938
19	ASHD	Amsterdam Holendrecht	0.0981
20	UTG	Uitgeest	0.1021
21	LEDN	Leiden Centraal	0.1042
22	AC	Abcoude	0.1043
23	HN	Hoorn	0.1063
24	SSH	Sassenheim	0.1066
25	BSMZ	Bussum Zuid	0.1085
26	AMF	Amersfoort	0.1097
27	HVS	Hilversum	0.1106
28	RTD	Rotterdam Centraal	0.1108
29	BLL	Bloemendaal	0.1146
30	OVN	Overveen	0.1146

Table 2: Top thirty stations for the mixed model with $\alpha_c = 5, \alpha_d = 3, \alpha_f = 1, \alpha_p = 1, \alpha_s = 2$.