A comprehensive analysis of Geneva's tram line18 and bus line 80: Identifying and pinpointing causes of bottlenecks

Ben Fall
July 2025

Under the supervision of
Dr. C.Cortes Balcells
Prof. Dr. K. Zhang

Context:

The city of Geneva is the second most populous city in Switzerland, with a municipal population of approximately 200,000 and a cantonal population nearing 530,000 (Etat de Genève). As a major European hub, Geneva faces significant challenges related to urban mobility and traffic congestion. Despite Switzerland's well-deserved reputation for efficient and punctual public transportation, Geneva is ranked as the second most congested city in the country, closely trailing Zurich. According to the 2024 TomTom Traffic Index, Geneva ranks 87th worldwide, with an average travel time of 25 minutes and 59 seconds per 10 kilometers traveled.

The city's public transit network plays a crucial role in managing commuter flows and reducing private vehicle use. Among the key components are tram line 18 and bus line 80, which serve as vital links between residential, commercial, and industrial zones. However, persistent bottlenecks on these lines have undermined the efficiency and reliability of the services, contributing to delays and commuter dissatisfaction.

Despite a population far from that of a megacity or even millionaire city, Geneva has some unique challenges. Due to its historic city design, as most European cities, the "Protestant Rome" has challenges in adapting its urban infrastructure to modern mobility demands. Narrow streets, limited space for expansion, and protected heritage zones constrain large-scale modifications, making it difficult to implement wide boulevards, dedicated bus lanes, or extensive cycling infrastructure. Additionally, Geneva's position as an international hub and cross-border commuter destination amplifies traffic volumes relative to its size, further complicating congestion management. Not only does its growing population add strain to the existing infrastructure but so too does, the increasing number of cross-border workers, "frontaliers" (specifically on the transportation lines coming in and out of the city) - a challenge very unique to the city of Geneva[1]. There are currently over 110,000 frontaliers from France working in Geneva and this number has grown unprecedentedly in recent years (Etat de Geneve).

The tramline 18, going from Palettes, Grand-Lancy to Meyrin CERN provided 17.797 million journeys in 2024, making it the 4th busiest line of all the TPG network, after lines 14,15 and 12. The average length of the line is 12 km and 837 meters and the average period/duration is 48 minutes and 28 seconds, giving a mean commercial speed[2] of 15.89 km per hour.

It may be noted that trams are by far, the most used form of public transportation in Geneva with 102.797 million journeys in 2024.

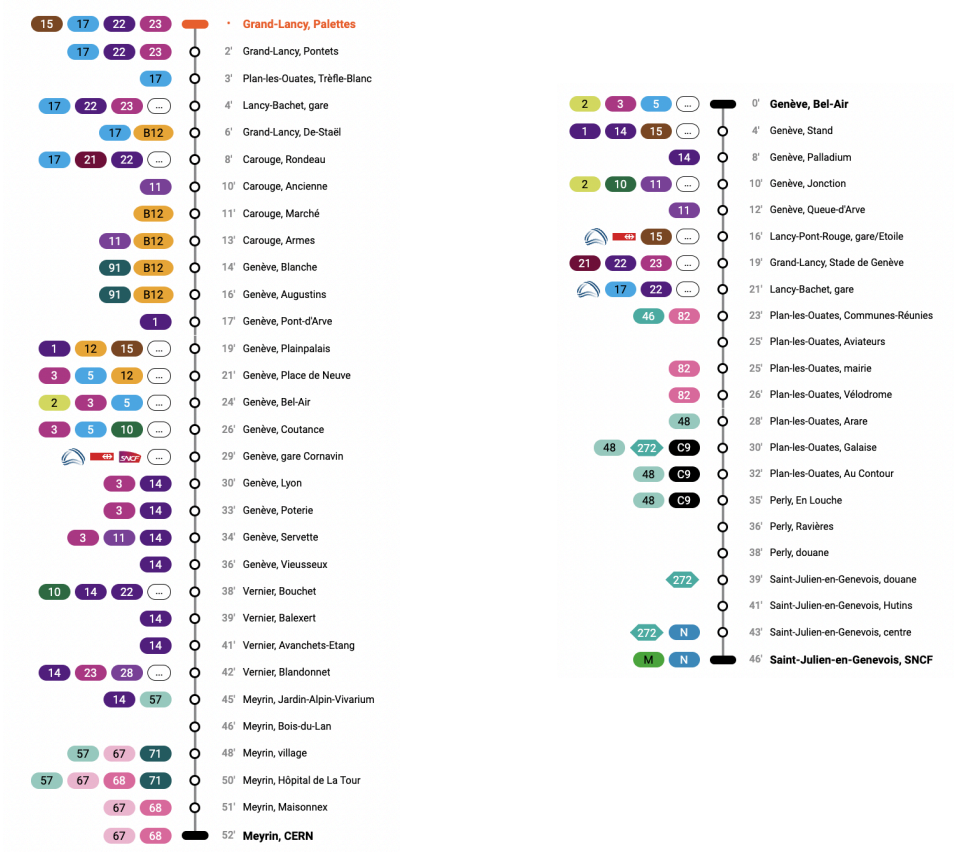
The bus line 80 (formerly Line D) is one of the most popular bus routes and has for the past couple of years been the most heavily used line in the cross-border network, with 5,942,000 journeys in 2024 (4,502,000 on the Swiss segment + 1,440,000 on the French segment) and 5,909,000 in 2023. The average length of the line is 10 km and 133 meters and the average period/duration is 36 minutes and 45 seconds, giving a mean commercial speed of 16.54 km per hour.

---

[1] An increasing population inevitably results in an increase in the number of vehicles in the city as well as a greater number of public transportation users.

[2] Commercial speed – the average speed of public transport vehicles, including time spent at stops - is a key indicator of the network's efficiency and resilience.

Line Overview



Tram Line 18 (left) and Bus Line 80 (right)

In real-world operations, schedules are frequently disrupted by factors such as congestion, signal delays, high passenger demand, vehicle interactions, and other unforeseen disturbances. These issues contribute not only to operational inefficiencies but also to increased air pollution, higher fuel consumption, and significant economic losses. The root causes are often linked to rapid population growth, accelerated urbanization, and poorly planned land use. Addressing these challenges is essential to alleviate urban traffic congestion and promote more sustainable and efficient transportation systems.

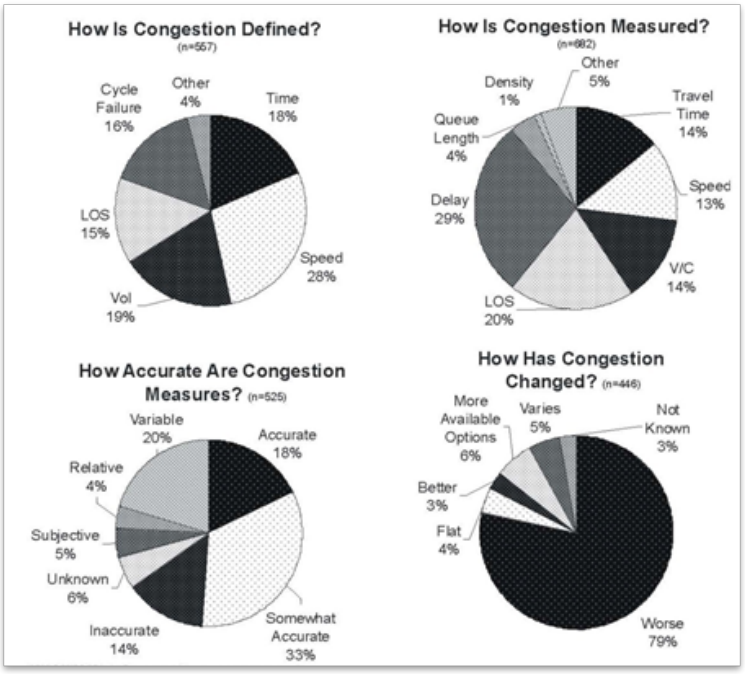
We will analyze data collected directly by the TPG in order to pinpoint bottlenecks and try to understand their causes, in order to improve network efficiency and passenger experience.

This brings us to the first challenge of understanding what a bottleneck is, in order to be able to identify one!

Understanding what a bottleneck is:

Traffic and congestion is a surprisingly difficult word to define in a metrizable manner. This is quintessential in order to quantify and analyse data.

Defining and measuring congestion remains a complex and somewhat subjective challenge, as highlighted by the responses to a study with over 500 participants from diverse professional backgrounds published by the US Department of Transportation's FHWA (Figure 1). While most researchers agree on the need to clearly define and quantify congestion to effectively address it, consensus on what exactly constitutes congestion is elusive. Some respondents attempted to pin it down with quantitative measures, such as speeds below the posted limit or falling under a certain speed threshold (e.g. less than 35 mph). However, others emphasized that congestion is more of a relative or perceptual concept - something people recognize intuitively but struggle to express in objective terms, often describing it as "I know it when I see it."



Understanding Congestion Study (Hale and Cronin)

This disparity reflects a fundamental difficulty: congestion is not just a straightforward physical state measurable by speed or density, but also a human experience influenced by expectations, context, and perception. Consequently, it complicates efforts to develop universal definitions or metrics that can capture congestion consistently across different scenarios and road users.

Moreover, beyond the question of how severe a traffic breakdown must be to qualify as a bottleneck, there remains the broader challenge of how to assess congestion at all levels - from minor slowdowns to severe gridlock. We must therefore focus on bottleneck identification and mitigation, rather than attempting to solve the broader, more nuanced

problem of defining and addressing congestion in all its forms, considering it always in both its dimensions: spatial and temporal.

One fairly straight-forward way was designed by Bucknell et al., which they introduced in their paper and then employed in Santiago, Chile as a case study, specifically on the bus network[3].

It is a simple two-criteria method, applying spatial discretization. Each bus route, r, is divided into x road segments of fixed length l (segments of $500\ m$).

Segments are labeled S_i where $i \in \{1,\ldots,\ x\}$, representing progression along the route, buses sequentially through segments S_1 through S_x.

A bottleneck along any segment is flagged if:

Its segment speed is less than 15 km/h (anything above being considered downstream speeds - meaning free-flowing traffic)

and

If improved to 15 km/h, the bus would save ≥ 1.5 minutes

Evidently, these are flexible variables (15 km/h, 1.5 mins, 500 meters) as otherwise, it is evident that these two conditions can be rewritten as one condition, however these were the heuristic thresholds chosen by Bucknell et al.

Time taken at $15\ km/hour$: $t_{15} = \frac{0.5\ km}{15\ km/h} = \frac{1}{30} hours\ =\ 2\ mins$

We want the speed to be such that the bus takes at least $1.5\ mins$ more.

$$2\ mins\ +\ 1.5\ mins\ =\ 3.5\ mins$$

$$t_{slow} \geq \frac{3.5}{60} = \frac{7}{120}\ hours$$

$$t_v = \frac{0.5}{v}$$

$$\frac{0.5}{v} = \frac{7}{120}$$

$$0.5 \times \frac{120}{7} = v$$

Hence threshold speed is 8.571 km/h, c. 43% less than the 15 km/hour standard, a typical commercial speed.

Considering the three variables as flexible ones: L (segment length in km), V (the commercial speed in km/s) and Δt (the target time lost), by simple rearrangements, we can generalise and find the threshold speed, v.

[3] The city of Santiago faced a critical issue: bus operating speeds were experiencing drops of c. 2.5% annually, as the number of vehicles in circulation was growing significantly, posing a significant challenge to its bus network (Garrido-Valenzuela et al.).

$$v = \frac{L}{(\frac{L}{V}+\frac{\Delta t}{60})} = \frac{60L}{60L+\Delta t \cdot V}$$

This evidently requires bus routes to not only be discretised into space but also time for systematic evaluation (temporal discretization necessary for the times), allowing us to very simply analyse bus speed profiles of road networks to detect operational bottlenecks.

We realise that this is a very simplified model and also gives us very little means of understanding the cause of the particular “bottleneck segment” we identify (Bucknell et al.).

It does however make the point that we should break down the journey into smaller segments. In our case, due to the design of both the bus and tram system, consecutive stops are typically at circa 400-600 meters distances, which makes them perfect candidates for segments.

The variability of the commercial speed of the segment is also a valid point to highlight. It also gives rise to the idea of differentiating between segments to categorize effectively. Some segments may be considered “short and slow”, “systematically slow”, “fast” or those of urban congestion!

Bucknell et al.’s method as phrased in Garrido-Valenzuela et al.’s paper, “Identifying and visualizing operational bottlenecks and Quick win opportunities for improving bus performance in public transport systems”, commenting on its limitations and proposing extensions.

- **Step 1: Detecting a bottleneck.** For each route r, time-step h, and segment i, a bottleneck is detected if the following conditions are met:

I. The speed of segment $(i+1)$ exceeds the speed of segment i, and the time saved for each bus on segment i, if the speed is increased to the speed of segment $(i+1)$, is at least γ s. This threshold ensures that the difference in speed is at least a significant value and not random.

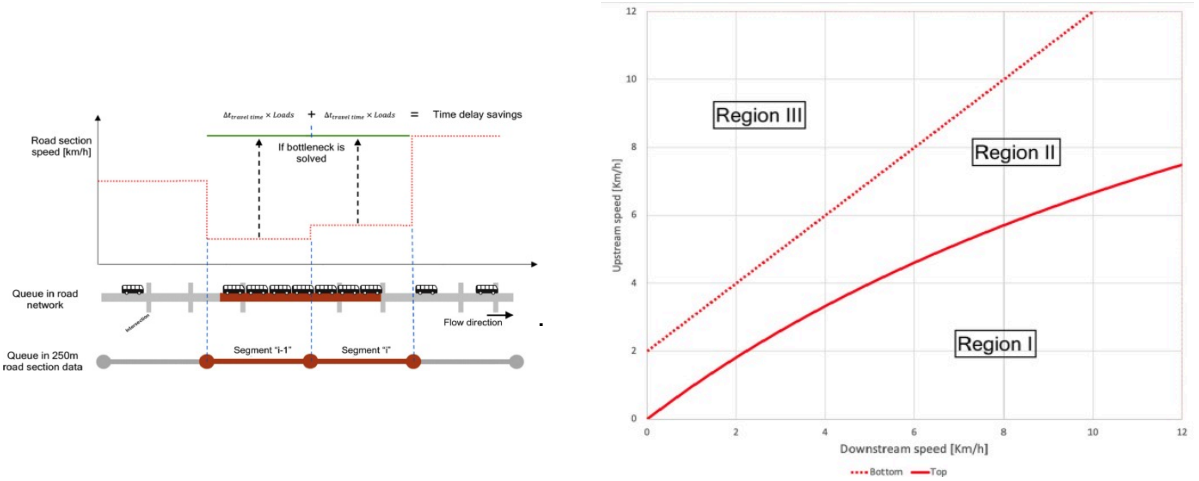
II. The speed on segment i is less than or equal to δ kilometers per hour. This upper bound filters out situations where, even though condition (I) is met, buses on segment i are traveling at a reasonable speed.

Therefore, if $\frac{l}{s^r_{h,i}} - \frac{l}{s^r_{h,i+1}} \geq \gamma$ and $s^r_{h,i} < \delta$, a bottleneck is identified on segment i at the time-step h.

Segments can then be examined in terms of their reliability - that is, how frequently delays occur. Does a road segment become congested only on rare occasions, or does it become congested routinely?

Garrido-Valenzuela et al. go on to develop Bucknell et al.’s method further. They highlight the limitation of the segment length i.e. either the fixed segment length is too long to identify precisely where the bottleneck is, or it is in fact shorter than the actual length of the bottleneck.

The primary value of the paper is providing an extension to very logically deduce the length of the bottleneck, allowing us to make segments even shorter and adding a consideration of the vehicle occupancy. Shorter segments make it easier to pinpoint specific issues - for instance if it is an issue with a specific turn or redlight. The method is otherwise the same as Bucknell et al. 's. At the end of each segment, one must compare the speeds from that segment to those in the next region to see if we are entering a bottleneck, exiting one or are in one that transcends the segment borders (proposing to use 250 meters instead of 500 meters).



Determining what happens at the boundaries (Garrido-Valenzuela et al.)

The method compares speeds between adjacent road segments. The figure above to the right shows three possible speed relationships:

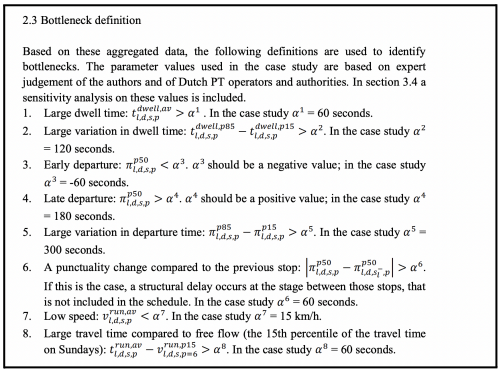
Region I: The upstream speed is much lower than the downstream speed. This indicates a new bottleneck is beginning - congestion starts here.
Region II: The upstream and downstream speeds are consistent. This suggests we are still within the same queue - the bottleneck continues.
Region III: The upstream speed is higher than the downstream speed. This means we have exited the bottleneck - the congestion ends at the downstream segment.

Given that this is a retrospective study/analysis (post hoc), the data has already been collected at specific points so Garrido-Valenzuela et al.'s extension unfortunately bears little direct value for the TPG case study.

While this is helpful at getting a first idea of identifying bottlenecks, it gives no ideas as to the underlying cause, nor does it, at all, tackle the issue of punctuality - really only considering speeds.

In a 2018 paper *Automatic bottleneck detection using AVL data: a case study in Amsterdam*, Brands. et al introduce a more comprehensive identification method which they then employ to analyse the tram network in the city of Amsterdam.

8 criteria are introduced which one then considers when aggregating the data: large dwell time, large variation in dwell time, early departure, late departure, large variation in dep time, punctuality change, low speed.



2.3 Bottleneck definition

Based on these aggregated data, the following definitions are used to identify bottlenecks. The parameter values used in the case study are based on expert judgement of the authors and of Dutch PT operators and authorities. In section 3.4 a sensitivity analysis on these values is included.

1. Large dwell time: $t_{l,d,s,p}^{dwell,av} > \alpha^1$. In the case study α^1 = 60 seconds.
2. Large variation in dwell time: $t_{l,d,s,p}^{dwell,p85} - t_{l,d,s,p}^{dwell,p15} > \alpha^2$. In the case study α^2 = 120 seconds.
3. Early departure: $\pi_{l,d,s,p}^{p50} < \alpha^3$. α^3 should be a negative value; in the case study α^3 = -60 seconds.
4. Late departure: $\pi_{l,d,s,p}^{p50} > \alpha^4$. α^4 should be a positive value; in the case study α^4 = 180 seconds.
5. Large variation in departure time: $\pi_{l,d,s,p}^{p85} - \pi_{l,d,s,p}^{p15} > \alpha^5$. In the case study α^5 = 300 seconds.
6. A punctuality change compared to the previous stop: $\left|\pi_{l,d,s,p}^{p50} - \pi_{l,d,s_i^-,p}^{p50}\right| > \alpha^6$. If this is the case, a structural delay occurs at the stage between those stops, that is not included in the schedule. In the case study α^6 = 60 seconds.
7. Low speed: $v_{l,d,s,p}^{run,av} < \alpha^7$. In the case study α^7 = 15 km/h.
8. Large travel time compared to free flow (the 15th percentile of the travel time on Sundays): $t_{l,d,s,p}^{run,av} - v_{l,d,s,p=6}^{run,p15} > \alpha^8$. In the case study α^8 = 60 seconds.

Such values are calculated for a specific stop on a specific line and direction/on the line segment leading to a specific stop. This is done for 6 different time periods, averaging values and 15, 50 and 85 percentile values for each period, namely, on weekdays: AM peak (7-9 AM), inter-peak period (9 AM-4 PM), PM peak (4-6 PM), and evening period (6 PM-12 AM) as well as Saturday and Sunday.
The AVL data required is very similar to what has been collected by the TPG.
Once processed, one should get a table like this:

Table 2 Number of bottleneck occurrences per time period and per criterion

Time period	Large dwell time	Large variation in dwell time	Early departure	Late departure	Large variation in departure time	Punctuality change compared to the previous stop	Low speed	Large travel time compared to free flow	At least one criterion
AM peak	4	0	17	0	27	11	87	7	**133**
Day period	4	0	11	0	60	10	87	13	**164**
PM peak	14	0	29	0	102	17	99	21	**219**
Evening	7	0	20	0	18	7	67	0	**109**
Saturdays	10	0	12	0	66	11	76	10	**158**
Sundays	6	0	29	0	0	9	64	10	**104**

This will again represent the number of bottleneck occurrences per time period and per criterion for a specific stop on a specific line and direction/on the line segment leading to a specific stop (Brands et al.). Differences in time periods are evidently extremely notable, especially due to the occupancy levels and general traffic: PM peak were noted to be the worst (more leisurely activity, pedestrians).
Evidently, it is essential to also have the number of trips to get a proportional idea as there is very likely a far greater number of trips in some time periods than others.
A sensitivity analysis showed that the number of detected bottlenecks is highly sensitive to the threshold values chosen for the criteria (alpha 1 to alpha 8), emphasizing the need for careful parameter selection.
This method is far more rigorous and diligent as it defines different types of bottlenecks however there seems to be the obvious issue in this method that a bottleneck will carry and may ripple through the trip result in us identifying “bottlenecks” in segments/zones where there are none due to criteria like realised punctuality. This will therefore need to be considered in analysis where “at least one criterion” cannot solely be used.

With the data we have, I named each segment by concatenating the ID of the departure station and the destination station. As stop IDs vary in length between 2 and 4 digits, I used the format XXXXYYYY so if you are on the A direction of the line 18 going from Genéve, Bel-Air (86) to Genéve, Coutance (3349), the segment is represented by 863349.

I deduced the segment speed by using station X's window exit time and station Y's window entry time. These were the only reliable pieces of data that I had for all rows/segments.
The windows are by default defined as being markings 35 meters before and 35 meters after each station. We also have the DistanceInterArre which represents the actual distance being travelled between the two consecutive (therefore also extremely similar to DistanceTheo). This means calculating the mean speed in km/hour for the majority of the segment length becomes quite simple (3.6*(DistanceInterArre-70)/(Delta between consecutive windows)). Note that there are a few stations where the windows are defined differently however the differences were considered (can be found in the appendix).
The dwell time was defined as the time spent within this window, even though it neither represents the time stopped per se at the station nor the time the doors were opened directly. I did however check and it indirectly gives an accurate indication for both the tram and bus. In the case of the tram, the door opening data was extremely dubious as the doors did not even have the ability to transmit this data while for the bus, while the data was there, it evidently didn't exist for every row given that there are times when no doors open/the bus doesn't stop unless requested.

Using the existing alpha values, without any modifications, already we can make some deductions which go to show the merit of the data.
While there is some obscurity in how criteria 2 (Large variation in dwell time) and criteria 5 (Large variation in departure time) are intended to be calculated - as well as punctuality change.
All are for instance quite unclear as they would appear to be a binary indicator, which when true would result in a 100% bottleneck. I have hence reached out to the researchers at Delft for a clearer explanation, however despite follow-ups, no explanation could be provided which may indicate a fallacy in their logic.

segment05221298_bottleneck_summary_with_n

period	large_dwell_time	large_var_in_dwell	early_departure	late_departure	large_var_in_departure	punct_change_prev	low_speed	large_travel_time_vs_freeflow	any_criterion	num_trips	any_criterion_percent
AM peak	479	0	63	578	0	0	4	29	994	2488	40.0
Day period	2743	0	237	1165	0	0	32	302	3543	5909	60.0
PM peak	1953	0	157	617	0	0	84	431	2211	2611	84.7
Evening	1877	0	236	653	0	0	33	134	2382	4700	50.7

Saturdays	1104	0	56	568	0	0	11	65	1339	2294	58.4
Sundays	402	0	56	332	0	0	5	23	665	1866	35.6
Other	168	0	38	94	0	0	0	6	278	1178	23.6

AM Peak was defined as being between 7 AM and 9 AM, Day Period between 9 AM and 4 PM, PM Peak between 4 PM and 6 PM, Evening between 6 PM and 12 AM, with Other being 12 AM to 7 AM on days other than Saturday, Sunday.
(using DTDepartTheo each time)

segment13010523_bottleneck_summary_with_n

period	**large_dwell_time**	**large_var_in_dwell**	**early_departure**	**late_departure**	**large_var_in_departure**	**punct_change_prev**	**low_speed**	**large_travel_time_vs_freeflow**	**any_criterion**	**num_trips**	**any_criterion_percent**
AM peak	1844	0	299	721	2629	0	426	1037	2629	2629	100.0
Day period	3641	0	211	1763	5625	0	112	537	5625	5625	100.0
PM peak	1880	0	265	486	0	0	93	353	2154	2631	81.9
Evening	2189	0	136	1016	0	0	22	183	2714	4279	63.4
Saturdays	1452	0	39	817	2285	0	6	14	2285	2285	100.0
Sundays	986	0	36	403	0	0	0	2	1129	1894	59.6
Other	780	0	28	216	0	0	5	98	902	1472	61.3

Segment 05221298 connects Lancy-Pont-Rouge, gare/Etoile and Grand-Lancy, Stade de Genève (line 80, direction A). While at first, the low AM peak may be surprising, it is quintessential to consider that the line 80 is a key frontalier line which means that it is very much direction R (from France to Switzerland) in the morning periods and direction A in the evenings - the higher occupancy inevitably leading to larger dwell times and bottlenecks.
We will conduct an analysis of the passenger data to establish where most people are going at particular times to determine a “directional arrow” of the passengers at different periods.
The PM peak dominates with 84.7% of journeys between 4 PM and 6 PM qualifying as bottlenecks. While it may be surprising to see that this figure stands at only 40.0% in the AM period, this does make sense considering that most traffic is on the other side of the street with very few going from Geneva outwards (centripetal vs centrifugal commute).
We also observe much calmer weekends with Sunday also noticeably less congested than Saturday which aligns with what one would expect considering blue laws and it traditionally being a day of rest.

If we take the same segment but in the opposite direction, we notice that the AM peak is particularly hit, which aligns with the fact that this would be the busy direction (going towards the CBD).
If we only consider the 4 very raw criteria, we still observe this same trend.

segment05221298_bottleneck_summary_with_n

period	large_dwell_time	early_departure	late_departure	low_speed	any_criterion	num_trips	any_criterion_percent
AM peak	479	63	578	4	982	2488	39.5
Day period	2743	237	1165	32	3429	5909	58.0
PM peak	1953	157	617	84	2162	2611	82.8
Evening	1877	236	653	33	2355	4700	50.1
Saturdays	1104	56	568	11	1333	2294	58.1
Sundays	402	56	332	5	662	1866	35.5
Other	168	38	94	0	274	1178	23.3

segment13010523_bottleneck_summary_with_n

period	large_dwell_time	early_departure	late_departure	low_speed	any_criterion	num_trips	any_criterion_percent
AM peak	1844	299	721	426	2226	2629	84.7
Day period	3641	211	1763	112	4336	5625	77.1
PM peak	1880	265	486	93	2101	2631	79.9
Evening	2189	136	1016	22	2657	4279	62.1
Saturdays	1452	39	817	6	1713	2285	75.0
Sundays	986	36	403	0	1128	1894	59.6
Other	780	28	216	5	863	1472	58.6

Doing this for all segments (that we consider) gave the following tables:

period	large_dwell_time	large_var_in_dwell	early_departure	late_departure	large_var_in_departure	punct_change_prev	low_speed	large_travel_time_vs_freeflow	any_criterion	num_trips	any_criterion_percent
AM peak	41811	0	19188	34832	0	16208	25571	17980	100809	214897	46.9
Day period	98066	0	50756	98867	0	29603	57813	29025	256755	584711	43.9
PM peak	52289	0	19325	48605	0	18793	31070	20797	119427	216672	55.1
Evening	54292	0	41786	67245	0	17708	33083	16819	170037	413777	41.1
Saturdays	32065	0	17362	43618	0	8305	15783	6488	93334	241161	38.7

Sundays	20061	0	18850	23060	0	5550	10639	3609	64897	197185	32.9
Other	15360	0	21399	14599	0	5667	11669	4686	58025	172779	33.6

period	large_dwell_time	early_departure	late_departure	low_speed	any_criterion	num_trips	any_criterion_percent
AM peak	41811	19188	34832	25571	97241	214897	45.3
Day period	98066	50756	98867	57813	250787	584711	42.9
PM peak	52289	19325	48605	31070	115961	216672	53.5
Evening	54292	41786	67245	33083	165742	413777	40.1
Saturdays	32065	17362	43618	15783	91982	241161	38.1
Sundays	20061	18850	23060	10639	63869	197185	32.4
Other	15360	21399	14599	11669	56664	172779	32.8

To familiarize myself with the lines and get an idea of which segments were generally “slow” or “fast”, I calculated the average of the segment speeds for each segment across all journeys.

Line 18, Direction A

Segment	Mean Speed (km/hour)
74221075	23.98
10751361	19.77
13610117	19.13
01171296	20.08
12960374	28.84
03740042	20.05
00425315	20.78
53150056	20.44
00560130	17.49
01300071	25.57
00711127	27.18
11271039	24.94
10391063	19.32
10630086	15.80
00863349	19.46
33490422	9.51
04220801	15.21
08011082	22.07

10821238	31.92
12381441	38.61
14410155	34.87
1550091	34.33
910077	33.35
770134	42.70
1340702	36.87
7023002	31.58
30020924	37.58
9240686	25.11
6860806	42.47
8060260	37.48

Line 18, Direction R

Segment	Mean Speed (km/hour)
02610807	37.12
08070687	37.03
06870925	27.28
09253003	36.71
30030703	31.09
07030137	35.23
01370078	37.97
00780092	32.25
00920156	35.29
01561440	35.78
14401237	24.71
12371081	28.63
10810800	23.32
08000427	14.60
04270356	9.88
03560087	23.96
00871064	16.40
10641041	17.79
10411128	25.06
11280072	26.84
00720131	27.10
01310057	15.64

00575316	20.68
53160043	21.07
00430375	19.67
03751297	26.29
12970118	22.28
01181362	19.82
13621077	21.98
10777423	19.42

Line 80, Direction A:

Segment	Mean Speed (km/hour)
23211310	17.31
13101022	27.68
10220720	29.44
07207166	41.39
71660522	40.23
05221298	39.69
12980122	46.93
01220335	22.85
03353303	24.90
33031037	33.94
10371414	30.45
14140052	38.28
00520630	32.04
06302117	37.41
21170782	29.23
07823301	27.09
33010982	32.40
09826495	32.27
64956497	42.27
64972521	31.22
25217444	17.97

Line 80, direction R

Segment	Mean Speed (km/hour)
74442522	19.09

25226498	38.63
64986496	43.48
64960983	26.13
09833302	31.71
33020780	37.63
07800353	30.22
03530631	34.74
06310053	45.53
00531415	38.10
14151038	inf
10383304	28.49
33040334	33.23
03340121	25.96
01211301	32.10
13010523	27.58
05237165	30.61
71650721	17.82
07211023	39.41
10231308	21.90
13082321	22.63

As one can see, notably from segment 14151038, where one single row led to the segment mean being inf. Without this row, the mean becomes 28.994471 km/hour. This goes to show that while these still remain good indicators, there remains sifting to do in cleaning the data of incorrect rows. While it is tempting to simply remove rows with segment travel times less than 5 seconds or so, speeds greater than 80 km/hour, and distances significantly different to the theoretical, we must come up with a more methodic process.

high_speed_14151038

IdCourse	IdArret	RangArretAs	EcartDepar	IdArretPreceden	DistanceInterArre	DTEntreeFenetreArretRea	DTSortieFenetreArretRea	DTDepartTheo	C_Lign	C_SensApp	Segment_XXXXYYYY	dwell_time	segment_travel_time	distance	segment_speed
85374483	1038	12	36	1415.0	218	2025-01-22 23:52:30	2025-01-22 23:52:43	2025-01-22 23:52:00.000	80	R	14151038	13.0	0.0	148.0	inf

Analyzing the data, I did quickly realise that the current dwell time alpha at 60 seconds was slightly too short (especially for line 80) as it was capturing far too many journeys. I conducted an analysis of the distributions as well as considering the delta between the door opening and window entry and exit of the window (time needed to travel the last 35 meters and first 35 meters) and established that 80 seconds may make for a more reasonable alpha1 value, specifically for line 80.

Dwell times for line 80 (a bus line) were quite close to a normal distribution while that for line 18 (a tram) was right-skewed (positive skewed) with quite a long tail and a small left bulge. A similar distinction between the bus and tram was noted for segment travel times, which is also logical, as trams are more constrained by their fixed tracks and regular stop patterns, leading to generally consistent travel times with occasional long delays (e.g., due to boarding congestion or priority at intersections), resulting in a right-skewed distribution, whereas

buses, which have more routing flexibility and variable traffic exposure, tend to show a bell-curve distribution with more balanced variation around a mean.

I was however encouraged to consider where people were going to determine the aforementioned directional arrows.

line18_directionA_boarding_alighting_ordered

IdArret	StationName	Boarding_AM peak	Boarding_Day period	Boarding_Evening	Boarding_Other	Boarding_PM peak	Boarding_Saturdays	Boarding_Sundays	Alighting_AM peak	Alighting_Day period	Alighting_Evening	Alighting_Other	Alighting_PM peak	Alighting_Saturdays	Alighting_Sundays	NetBoarding_AM peak	NetBoarding_Day period	NetBoarding_Evening	NetBoarding_Other	NetBoarding_PM peak	NetBoarding_Saturdays	NetBoarding_Sundays
7422	Grand-Lancy, Palettes	11281	25856	9841	1566	12870	10445	3527	13	5	20	0	0	2	13	11268	25951	9821	1566	12870	10443	3514
1075	Grand-Lancy, Pontets	18523	42342	12088	8346	12502	13283	7916	959	4698	1848	388	2185	1291	1105	17564	37644	10240	7958	10317	11992	6811
1361	Plan-les-Ouates, Trèfle-Blanc	12224	22776	6468	2776	6629	7116	3961	920	4139	1140	85	2070	998	474	11304	18637	5328	2691	4559	6118	3487
117	Lancy-Bachet, gare	27209	47080	18210	8121	15515	13999	8656	10854	22606	7601	3063	10389	6136	3712	16355	24474	10609	5058	5126	7863	4944
1296	Grand-Lancy, De-Staël	3527	20074	5544	1908	8517	3699	2427	14847	14971	1379	1662	1752	757	623	-11320	5103	4165	246	6765	2942	1804
374	Carouge, Rondeau	25920	68194	26354	7499	32489	20399	11780	7709	19759	6687	1926	8614	4180	2531	18211	48435	19667	5573	23875	16219	9249
42	Carouge, Ancienne	9179	29095	16109	3816	11618	10823	7048	3288	9812	3923	554	4252	2299	1564	5891	19283	12186	3262	7366	8524	5484
5315	Carouge, Marché	19097	59074	23135	7832	23132	22374	12579	5149	22268	7503	1144	9250	7133	3811	13948	36806	15632	6688	13882	15241	8768
56	Carouge, Armes	22851	72375	33258	8596	29369	27708	16688	4662	22757	7406	1043	10230	4855	2966	18189	49618	25852	7553	19139	22853	13722
130	Genève, Blanche	27871	62220	23162	8935	27464	22434	14614	5230	15117	7324	1289	7230	4591	3191	22641	47103	15838	7646	20234	17843	11423
71	Genève, Augustins	39255	140511	60504	12821	64761	39940	22489	16071	50679	14068	3966	19239	13270	5439	23184	89832	46436	8855	45522	26670	17050
1127	Genève, Pont-d'Arve	27819	96247	45921	10903	41367	33152	19456	12891	44886	14678	2284	16717	12277	6983	14928	51361	31243	8619	24650	20875	12473
1039	Genève, Plainpalais	19840	100952	49372	7965	45531	37208	23367	13464	57362	21204	2823	22710	17072	11292	6376	43590	28168	5142	22821	20136	12075
1063	Genève, Place de Neuve	7563	49115	30371	2959	26487	18007	12666	13641	43179	17817	2734	17446	13314	8546	-6078	5936	12554	225	9041	4693	4120
86	Genève, Bel-Air	47145	187782	110192	20355	78113	70998	37114	25097	103302	44390	6242	40662	34928	18592	22048	84480	65802	14113	37451	36070	18522
3349	Genève, Coutance	10435	72825	31181	3301	35143	29857	8366	19744	115204	33338	4758	44725	35890	10374	-9309	42379	-2157	-1457	-9582	-9033	-2008
422	Genève, gare Cornavin	90987	241862	177816	30992	119381	105389	81584	74908	221235	133050	31719	113071	87684	61806	16079	20627	44766	-727	6310	17705	19778
801	Genève, Lyon	12412	49111	31228	6701	23893	19720	12964	12800	53389	34552	3150	24360	20314	13424	-388	-4278	-3324	3551	-467	-594	-460
1082	Genève, Poterie	15272	48417	23248	4695	22323	17080	8526	11480	79299	55055	4880	37624	31275	20619	3792	-30882	-31807	-185	-15301	-14195	-12093
1238	Genève, Servette	27154	83326	41322	7490	39239	29615	14731	18907	92017	68331	7219	42000	38193	26815	8247	-8691	-27009	271	-2761	-8578	-12084
1441	Genève, Vieusseux	13645	34961	13627	4010	15455	12167	6179	15435	54744	36520	3990	27375	18667	13931	-1790	-19783	-22893	20	-11920	-6500	-7752
155	Vernier, Bouchet	16886	68963	24086	5368	36113	16897	8552	32895	66803	39250	7257	27448	19714	15476	-16009	2160	-15164	-1889	8665	-2817	-6924
91	Vernier, Balexert	7939	62591	28880	3411	37538	25329	6735	15705	122162	48465	4222	58821	49928	15969	-7766	-59571	-19585	-811	-21283	-24599	-9234
77	Vernier, Avanchets-Etang	9574	23970	12827	3805	11821	8259	5069	17219	47983	42659	5793	26799	19836	13991	-7645	-24013	-29832	-1988	-14978	-11577	-8922
134	Vernier, Blandonnet	13612	48850	34763	4869	27710	17207	10486	25848	59131	33519	10689	24918	21485	10365	-12236	-10281	1244	-5820	2792	-4278	121
702	Meyrin, Jardin-Alpin-Vivarium	11565	46933	24402	4062	25585	15552	7173	6004	34410	20692	4864	20887	13407	6155	5561	12523	3710	-802	4698	2145	1018
3002	Meyrin, Bois-du-Lan	3898	12907	6103	1964	6893	4157	2613	13654	64747	29357	6421	31183	24478	8557	-9756	-51840	-23254	-4457	-24290	-20321	-5944
924	Meyrin, village	3624	12179	6342	1630	6462	3810	2467	17177	82227	61466	5782	53284	27721	17373	-13553	-70048	-55124	-4152	-46822	-23911	-14906
686	Meyrin, Hôpital de La Tour	3523	8737	4590	1565	3689	3068	2003	34357	88095	44060	12941	40953	24287	15313	-30834	-79358	-39470	-11376	-37264	-21219	-13310
806	Meyrin, Maisonnex	1347	4215	2739	1388	1387	1860	1218	657	3034	2370	265	2703	1151	862	690	1181	369	1123	-1316	709	356
260	Meyrin, CERN	41	235	118	18	292	174	63	56086	126833	88784	9645	54792	52963	38212	-56045	-126598	-88666	-9627	-54500	-52789	-38149

line18_directionR_boarding_alighting_ordered

IdArret	StationName	Boarding_AM peak	Boarding_Day period	Boarding_Evening	Boarding_Other	Boarding_PM peak	Boarding_Saturdays	Boarding_Sundays	Alighting_AM peak	Alighting_Day period	Alighting_Evening	Alighting_Other	Alighting_PM peak	Alighting_Saturdays	Alighting_Sundays	NetBoarding_AM peak	NetBoarding_Day period	NetBoarding_Evening	NetBoarding_Other	NetBoarding_PM peak	NetBoarding_Saturdays	NetBoarding_Sundays
7423	Grand-Lancy, Palettes	4	120	18	0	285	15	0	11646	32541	12127	2594	14909	4598	7231	-11642	-32421	-12109	-2594	-14624	-4583	-7231
1077	Grand-Lancy, Pontets	5179	12737	5088	1898	4738	3495	2177	4332	35885	21611	1822	20259	11524	7644	847	-23148	-16523	76	-15521	-8029	-5467
1362	Plan-les-Ouates, Trèfle-Blanc	3468	7678	3485	1603	2638	2400	1597	3450	19337	11206	1286	10385	6392	3992	18	-11659	-7721	317	-7747	-3992	-2395
118	Lancy-Bachet, gare	7419	26112	14986	2753	15800	7737	5018	12212	47383	30425	5792	30915	14827	9238	-4793	-21271	-15439	-3039	-15115	-7090	-4220
1297	Grand-Lancy, De-Staël	3094	25671	5038	1387	12881	2687	1872	7616	14894	3833	2410	3012	2579	1631	-4522	10777	1205	-1023	9869	108	241
375	Carouge, Rondeau	10193	22870	11280	3199	11598	6082	3874	24168	63990	38121	6164	27149	19341	11855	-13975	-41120	-26841	-2965	-15551	-13259	-7981
43	Carouge, Ancienne	4834	12644	7322	2059	5667	4030	2505	5702	25203	20074	1097	11046	9476	6767	-868	-12559	-12752	962	-5379	-5446	-4262
5316	Carouge, Marché	6176	24329	12651	2676	10847	8881	4614	10776	53358	31494	3360	24583	22783	13652	-4600	-29029	-18843	-684	-13736	-13902	-9038
57	Carouge, Armes	9519	23659	11687	2393	9973	6660	4147	14898	66953	39684	4148	29347	24996	15009	-5379	-43294	-27997	-1755	-19374	-18336	-10862
131	Genève, Blanche	6788	18798	9506	2643	8478	5963	3982	19269	49993	39693	3610	21913	19584	14347	-12481	-31195	-30187	-967	-13435	-13621	-10365
72	Genève, Augustins	10569	66925	29998	3970	33086	19858	6892	44183	131856	63210	11145	52541	37918	23645	-33594	-64931	-33212	-7175	-19455	-18060	-16753
1128	Genève, Pont-d'Arve	12423	47125	25028	3416	21536	13830	8334	25886	93036	46903	3976	35406	28345	15987	-13463	-45911	-21875	-560	-13870	-14515	-7653
1041	Genève, Plainpalais	14413	63071	35259	4133	27060	21084	13343	18934	78498	36206	3561	28773	30447	18132	-4521	-15427	-947	572	-1713	-9363	-4789
1064	Genève, Place de Neuve	8497	36104	21283	2985	16718	11753	7147	10717	41328	16839	1402	13831	12834	8736	-2220	-5224	4444	1583	2887	-1081	-1589
87	Genève, Bel-Air	23566	81715	48230	7053	36473	30678	18817	52237	133330	65204	13454	56883	52318	27428	-28671	-51615	-16974	-6401	-20410	-21640	-8611
356	Genève, Coutance	17243	100637	38340	5043	41900	33735	11285	14079	61018	19246	2490	22414	21800	6890	3164	39619	19094	2553	19486	11935	4395
427	Genève, gare Cornavin	73596	196121	123379	14054	89358	79634	63945	75211	188620	107305	23955	97696	80344	56944	-1615	7501	16074	-9901	-8338	-710	7001
800	Genève, Lyon	12489	43097	22081	4751	19660	16761	9853	16286	40820	24677	3232	21479	15343	9166	-3797	2277	-2596	1519	-1819	1418	687
1081	Genève, Poterie	20421	58445	24813	5499	22380	22337	12752	10013	37710	21038	1506	18922	13377	6443	10408	20735	3775	3993	3458	8960	6309
1237	Genève, Servette	31982	90972	39284	9418	36180	35425	21927	25455	68357	38238	5935	33344	24109	12155	6527	22615	1046	3483	2836	11316	9772
1440	Genève, Vieusseux	24286	58445	21132	6315	22495	18600	12061	5177	24995	14664	851	13422	9356	4679	19109	33450	6468	5464	9073	9244	7382
156	Vernier, Bouchet	23165	65566	27935	5774	31488	18577	13980	33121	50834	18934	3701	19576	12998	6319	-9956	14732	9001	2073	11912	5579	7661
92	Vernier, Balexert	24246	90617	45895	7204	44193	44991	15020	14110	57240	16562	2014	24603	22366	5237	10136	33377	29333	5190	19590	22625	9783
78	Vernier, Avanchets-Etang	23694	51266	28558	6412	23145	20376	13874	5866	16589	11498	1913	11118	6215	3770	17828	34677	17060	4499	12027	14161	10104
137	Vernier, Blandonnet	12950	46738	29488	5234	29401	19474	9502	22139	42041	15850	7357	23126	14840	6542	-9189	4697	13638	-2123	6275	4634	2960
703	Meyrin, Jardin-Alpin-Vivarium	7360	26287	14929	2349	11709	11582	5300	4565	25249	12617	1005	11695	8322	3009	2795	1038	2312	1344	14	3260	2291
3003	Meyrin, Bois-du-Lan	20693	66691	23658	8522	28288	26399	8838	2502	9278	5262	890	5000	3199	1891	18191	57413	18396	7632	23288	23200	6947
925	Meyrin, village	47041	83462	28782	14531	30516	26913	15429	3871	10291	5508	690	5092	2839	1710	43170	73171	23274	13841	25424	24074	13719
687	Meyrin, Hôpital de La Tour	46249	115465	42412	13081	53909	29920	18360	1848	6387	2811	440	3560	1770	1089	44401	109078	39601	12641	50349	28150	17271
807	Meyrin, Maisonnex	3782	7351	4106	1863	2850	3040	2265	131	268	198	27	130	107	81	3651	7083	3908	1836	2720	2933	2184
261	Meyrin, CERN	67177	117804	69807	24879	78239	56625	34886	11	121	227	16	85	48	37	67166	117683	69580	24863	78154	56577	34849

line80_directionA_boarding_alighting_ordered

IdArret	StationName	Boarding_AM peak	Boarding_Day period	Boarding_Evening	Boarding_Other	Boarding_PM peak	Boarding_Saturdays	Boarding_Sundays	Alighting_AM peak	Alighting_Day period	Alighting_Evening	Alighting_Other	Alighting_PM peak	Alighting_Saturdays	Alighting_Sundays	NetBoarding_AM peak	NetBoarding_Day period	NetBoarding_Evening	NetBoarding_Other	NetBoarding_PM peak	NetBoarding_Saturdays	NetBoarding_Sundays
2321	Geneve, Bel-Air	4674	32566	44351	3253	29446	17736	9046	3	807	9	3	55	11	17	4671	31759	44342	3250	29391	17725	9029
1310	Genève, Stand	16608	43845	46499	6560	33148	18541	11003	308	1207	1006	152	727	432	229	16300	42638	45493	6408	32421	18109	10774
1022	Genève, Palladium	6308	16318	13855	1532	12662	5275	3088	1922	3679	3197	243	2634	1152	715	4386	12639	10658	1289	10028	4123	2373
720	Geneve, Jonction	30949	60845	37597	10882	34498	20521	12155	4017	10149	8087	704	6634	3205	2197	26932	50696	29510	10178	27864	17316	9958
7166	Genève, Queue-d'Arve	1906	10221	8216	273	12575	3260	1276	4437	8195	3469	1132	4147	3411	752	-2531	2026	4747	-859	8428	-151	524
522	Lancy-Pont-Rouge, gare/Etoile	14333	51363	38682	4613	40706	14848	7913	15714	24330	18290	2992	18161	5237	2682	-1381	27033	20392	1621	22545	9611	5231
1298	Grand-Lancy, Stade de Geneve	2717	27628	19934	538	19332	12903	4530	7796	43593	26684	2877	25609	23121	9634	-5079	-15965	-6750	-2339	-6277	-10218	-5104
122	Lancy-Bachet, gare	32684	55556	47909	11587	43004	16760	10711	6897	27192	25994	1626	20029	9345	5614	25787	28364	21915	9961	22975	7415	5097
335	Plan-les-Ouates, Communes-Réunies	4772	15906	10122	1736	11378	4445	2605	2081	10506	11521	532	9197	4382	2461	2691	5400	-1399	1204	2181	63	144
3303	Plan-les-Ouates, Aviateurs	1695	12936	4138	265	9623	2306	1131	5431	25798	19584	577	17254	7903	4785	-3736	-12862	-15446	-312	-7631	-5597	-3654
1037	Plan-les-Ouates, mairie	508	2118	1523	89	2146	616	317	4010	10045	8935	877	8321	4261	2350	-3502	-7927	-7412	-788	-6175	-3645	-2033
1414	Plan-les-Ouates, Vélodrome	1745	11104	3792	315	8598	2165	793	16774	29258	16382	7779	14943	7520	4609	-15029	-18154	-12590	-7464	-6345	-5355	-3816
52	Plan-les-Ouates, Arare	516	4129	4743	149	10730	671	377	23125	21220	11296	7491	11352	5041	3730	-22609	-17091	-6553	-7342	-622	-4370	-3353
630	Plan-les-Ouates, Galaise	375	4123	4371	88	7898	558	397	11977	9857	2781	2766	3423	1383	731	-11602	-5734	1590	-2678	4475	-825	-334
2117	Plan-les-Ouates, Au Contour	26	131	85	13	177	36	24	363	1177	795	463	779	309	162	-337	-1046	-710	-450	-602	-273	-138
762	Perly, En Louche	176	1131	884	10	1965	328	147	2097	12477	10301	807	9832	3489	2260	-1921	-11346	-9417	-797	-7867	-3161	-2113
3301	Perly, Ravieres	874	4294	2926	91	3498	1612	842	3971	21269	18753	1125	15327	7384	4241	-3097	-16975	-15827	-1034	-11829	-5772	-3399
982	Perly, douane	322	1246	1510	38	1413	484	304	2442	7348	4841	792	4886	2169	1319	-2120	-6102	-3331	-754	-3473	-1685	-1015
6495	Saint-Julien-en-Genevois, douane	79	878	775	10	1044	252	130	884	11667	14366	266	12339	4218	2199	-805	-10789	-13591	-256	-11295	-3966	-2069
6497	Saint-Julien-en-Genevois, Hutins	780	8274	1748	63	7528	933	405	1373	14160	22612	730	14998	7590	3879	-593	-5886	-20864	-667	-7470	-6657	-3474
2521	Saint-Julien-en-Genevois, centre	549	1823	1366	53	1277	544	331	7402	39116	62835	2602	44659	17495	9837	-6853	-37293	-61469	-2549	-43382	-16951	-9506
7444	Saint-Julien-en-Genevois, SNCF	170	365	1647	356	121	199	109	5505	28804	44197	2643	30690	12324	7717	-5335	-28439	-42550	-2287	-30569	-12125	-7608

line80_directionR_boarding_alighting_ordered

IdArret	StationName	Boarding_AM peak	Boarding_Day period	Boarding_Evening	Boarding_Other	Boarding_PM peak	Boarding_Saturdays	Boarding_Sundays	Alighting_AM peak	Alighting_Day period	Alighting_Evening	Alighting_Other	Alighting_PM peak	Alighting_Saturdays	Alighting_Sundays	NetBoarding_AM peak	NetBoarding_Day period	NetBoarding_Evening	NetBoarding_Other	NetBoarding_PM peak	NetBoarding_Saturdays	NetBoarding_Sundays
2321	Geneve, Bel-Air	33	30	38	112	44	52	23	34409	75438	38728	12713	35008	31762	16398	-34376	-75408	-38690	-12601	-34964	-31710	-16375
1308	Genève, Stand	429	1839	1263	128	1090	1163	460	19946	31876	15190	4218	14889	11028	6940	-19517	-30037	-13927	-4090	-13799	-9865	-6480
1023	Genève, Palladium	5867	11631	7076	1001	8401	3380	2312	10886	16862	9344	1441	7895	4991	2513	-5019	-5231	-2268	-440	506	-1611	-201
721	Geneve, Jonction	11242	21356	12656	2904	13793	7211	4778	26627	65258	34854	6322	34930	18943	11885	-15385	-43902	-22198	-3418	-21137	-11732	-7107
7165	Genève, Queue-d'Arve	1664	6223	3725	167	5113	2156	663	14099	14387	4493	3953	5692	3835	1546	-12435	-8164	-768	-3786	-579	-1679	-883
523	Lancy-Pont-Rouge, gare/Etoile	17672	26907	12838	2908	14148	4972	2858	31346	45908	15077	8866	16366	11478	6349	-13674	-19001	-2239	-5958	-2218	-6506	-3491
1301	Grand-Lancy, Stade de Geneve	8889	33907	18721	1658	17506	16664	7502	6512	28665	8336	1860	12726	11178	3977	2377	5242	10385	-202	4780	5486	3525
121	Lancy-Bachet, gare	21642	32543	11147	4799	15023	8392	4554	59634	81164	31115	22020	43323	20338	11638	-37992	-48621	-19968	-17221	-28300	-11946	-7084
334	Plan-les-Ouates, Communes-Réunies	5792	10881	3872	1903	4831	3825	2137	7042	12591	4164	1893	5771	3309	1858	-1250	-1710	-292	10	-940	516	279
3304	Plan-les-Ouates, Aviateurs	12840	31774	10172	3214	12570	8713	4801	7118	11217	2709	437	3399	1902	972	5722	20557	7463	2777	9171	6811	3829
1038	Plan-les-Ouates, mairie	5972	12264	5372	1007	5652	3887	2179	1834	2936	868	200	1140	590	414	4138	9328	4504	807	4512	3297	1765
1415	Plan-les-Ouates, Vélodrome	9866	28552	11359	2223	20960	6317	3556	5141	7578	2138	846	2405	1642	614	4725	20974	9221	1377	18555	4675	2942
53	Plan-les-Ouates, Arare	9664	21219	10893	1936	16953	4163	3011	11670	4023	1252	5185	1047	685	459	-2006	17196	9641	-3249	15906	3478	2552
631	Plan-les-Ouates, Galaise	4433	7525	4515	416	7435	1291	701	10588	4035	580	3133	1004	440	286	-6155	3490	3935	-2717	6431	851	415
353	Plan-les-Ouates, Au Contour	1350	1502	433	419	451	394	250	256	236	73	56	54	78	38	1094	1266	360	363	397	316	212
780	Perly, En Louche	8301	13175	2901	2487	5163	3254	1903	645	1369	410	398	555	408	227	7656	11806	2491	2089	4608	2846	1676
3302	Perly, Ravieres	13790	22979	7660	5592	7491	7235	3891	1602	4933	1760	461	2141	1886	1107	12188	18046	5900	5131	5350	5349	2784
983	Perly, douane	5503	11025	3182	1836	3507	3288	1659	1024	2183	733	680	1532	710	353	4479	8842	2449	1158	1975	2578	1306
6496	Saint-Julien-en-Genevois, douane	10439	10988	1599	5960	2221	2783	1545	605	914	244	180	295	225	86	9834	10074	1355	5780	1926	2558	1459
6498	Saint-Julien-en-Genevois, Hutins	16754	19444	4394	9746	5644	8140	3794	9465	8102	2220	547	3054	1392	575	7289	11342	2174	9199	2590	6748	3219
2522	Saint-Julien-en-Genevois, centre	45462	47492	11377	20868	16012	15537	7402	978	2352	728	423	900	613	326	44484	45140	10649	20445	15112	14924	7076
7444	Saint-Julien-en-Genevois, SNCF	45037	40823	13210	27987	12367	13986	7618	707	1024	134	1868	422	182	72	44330	39799	13076	26119	11945	13804	7546

line18_directionA_netboarding_TOTALS

IdArret	StationName	AM peak	Day period	Evening	Other	PM peak	Saturdays	Sundays	NbMontees	NbDescente	NetBoardingTotal	TotalMovement
7422	Grand-Lancy, Palettes	11268	25951	9821	1566	12870	10443	3514	75486	53	75433	75539
1075	Grand-Lancy, Pontets	17564	37644	10240	7958	10317	11992	6811	115000	12474	102526	127474
1361	Plan-les-Ouates, Tréfle-Blanc	11304	18637	5328	2691	4559	6118	3487	61950	9826	52124	71776
117	Lancy-Bachet, gare	16355	24474	10609	5058	5126	7863	4944	138790	64361	74429	203151
1296	Grand-Lancy, De-Staél	-11320	5103	4165	246	6765	2942	1804	45696	35991	9705	81687
374	Carouge, Rondeau	18211	48435	19667	5573	23875	16219	9249	192635	51406	141229	244041

42	Carouge, Ancienne	5891	19283	12186	3262	7366	8524	5484	87688	25692	61996	113380
5315	Carouge, Marché	13948	36806	15632	6688	13882	15241	8768	167223	56258	110965	223481
56	Carouge, Armes	18189	49618	25852	7553	19139	22853	13722	210845	53919	156926	264764
130	Genéve, Blanche	22641	47103	15838	7646	20234	17843	11423	186700	43972	142728	230672
71	Genéve, Augustins	23184	89832	46436	8855	45522	26670	17050	380281	122732	257549	503013
1127	Genéve, Pont-d'Arve	14928	51361	31243	8619	24650	20875	12473	274865	110716	164149	385581
1039	Genéve, Plainpalais	6376	43590	28168	5142	22821	20136	12075	284235	145927	138308	430162
1063	Genéve, Place de Neuve	-6078	5936	12554	225	9041	4693	4120	147168	116677	30491	263845
86	Genéve, Bel-Air	22048	84480	65802	14113	37451	36070	18522	551699	273213	278486	824912
3349	Genéve, Coutance	-9309	-42379	-2157	-1457	-9582	-9033	-2008	188108	264033	-75925	452141
422	Genéve, gare Cornavin	16079	20627	44766	-727	6310	17705	19778	848011	723473	124538	1571484
801	Genéve, Lyon	-388	-4278	-3324	3551	-467	-594	-460	156029	161989	-5960	318018
1082	Genéve, Poterie	3792	-30882	-31807	-185	-15301	-14195	-12093	139561	240232	-100671	379793
1238	Genéve, Servette	8247	-8691	-27009	271	-2761	-8578	-12084	242877	293482	-50605	536359
1441	Genéve, Vieusseux	-1790	-19783	-22893	20	-11920	-6500	-7752	100044	170662	-70618	270706
155	Vernier, Bouchet	-16009	2160	-15164	-1889	8665	-2817	-6924	176865	208843	-31978	385708
91	Vernier, Balexert	-7766	-59571	-19585	-811	-21283	-24599	-9234	172423	315272	-142849	487695
77	Vernier, Avanchets-Etang	-7645	-24013	-29832	-1988	-14978	-11577	-8922	75325	174280	-98955	249605
134	Vernier, Blandonnet	-12236	-10281	1244	-5820	2792	-4278	121	157497	185955	-28458	343452
702	Meyrin, Jardin-Alpin-Vivarium	5561	12523	3710	-802	4698	2145	1018	135272	106419	28853	241691
3002	Meyrin, Bois-du-Lan	-9756	-51840	-23254	-4457	-24290	-20321	-5944	38535	178397	-139862	216932

924	Meyrin, village	-13553	-70048	-55124	-4152	-46822	-23911	-14906	36514	265030	-228516	301544
686	Meyrin, Hépital de La Tour	-30834	-79358	-39470	-11376	-37264	-21219	-13310	27175	260006	-232831	287181
806	Meyrin, Maisonnex	690	1181	369	1123	-1316	709	356	14154	11042	3112	25196
260	Meyrin, CERN	-56045	-126598	-88666	-9627	-54500	-52789	-38149	941	427315	-426374	428256

line18_directionR_netboarding_TOTALS

IdArret	StationName	AM peak	Day period	Evening	Other	PM peak	Saturdays	Sundays	NbMontees	NbDescente	NetBoardingTotal	TotalMovement
261	Meyrin, CERN	67166	117683	69580	24863	78154	56577	34849	449417	545	448872	449962
807	Meyrin, Maisonnex	3651	7083	3908	1836	2720	2933	2184	25257	942	24315	26199
687	Meyrin, Hépital de La Tour	44401	109078	39601	12641	50349	28150	17271	319396	17905	301491	337301
925	Meyrin, village	43170	73171	23274	13841	25424	24074	13719	246674	30001	216673	276675
3003	Meyrin, Bois-du-Lan	18191	57413	18396	7632	23288	23200	6947	183089	28022	155067	211111
703	Meyrin, Jardin-Alpin-Vivarium	2795	1038	2312	1344	14	3260	2291	79516	66462	13054	145978
137	Vernier, Blandonnet	-9189	4697	13638	-2123	6275	4634	2960	152787	131895	20892	284682
78	Vernier, Avanchets-Etang	17828	34677	17060	4499	12027	14161	10104	167325	56969	110356	224294
92	Vernier, Balexert	101366	33737	29333	5190	195900	22625	9783	272166	142132	130034	414298
156	Vernier, Bouchet	-9956	14732	9001	2073	11912	5579	7661	186485	145483	41002	331968
1440	Genéve, Vieusseux	191099	33450	6468	5464	9073	9244	7382	163334	73144	90190	236478
1237	Genéve, Servette	6527	22615	1046	3483	2836	11316	9772	265188	207593	57595	472781
1081	Genéve, Poterie	10408	20735	3775	3993	3458	8960	6309	166647	109009	57638	275656
800	Genéve, Lyon	-3797	2277	-2596	1519	-1819	1418	687	128692	131003	-2311	259695

427	Genéve, gare Cornavin	-1615	7501	16074	-9901	-8338	-710	7001	640087	630075	10012	1270162
356	Genéve, Coutance	3164	39619	19094	2553	19486	11935	4395	248183	147937	100246	396120
87	Genéve, Bel-Air	-28671	-51615	-16974	-6401	-20410	-21640	-8611	246532	400854	-154322	647386
1064	Genéve, Place de Neuve	-2220	-5224	4444	1583	2887	-1081	-1589	104487	105687	-1200	210174
1041	Genéve, Plainpalais	-4521	-15427	-947	572	-1713	-9363	-4789	178363	214551	-36188	392914
1128	Genéve, Pont-d'Arve	-13463	-45911	-21875	-560	-13870	-14515	-7653	131692	249539	-117847	381231
72	Genéve, Augustins	-33594	-64931	-33212	-7175	-19455	-18060	-16753	171298	364478	-193180	535776
131	Genéve, Blanche	-12481	-31195	-30187	-967	-13435	-13621	-10365	56158	168409	-112251	224567
57	Carouge, Armes	-5379	-43294	-27997	-1755	-19374	-18336	-10862	68038	195035	-126997	263073
5316	Carouge, Marché	-4600	-29029	-18843	-684	-13736	-13902	-9038	70174	160006	-89832	230180
43	Carouge, Ancienne	-868	-12559	-12752	962	-5379	-5446	-4262	39061	79365	-40304	118426
375	Carouge, Rondeau	-13975	-41120	-26841	-2965	-15551	-13259	-7981	69096	190788	-121692	259884
1297	Grand-Lancy, De-Staél	-4522	10777	1205	-1023	9869	108	241	52630	35975	16655	88605
118	Lancy-Bachet, gare	-4793	-21271	-15439	-3039	-15115	-7090	-4220	79825	150792	-70967	230617
1362	Plan-les-Ouates, Tréfle-Blanc	18	-11659	-7721	317	-7747	-3992	-2395	22869	56048	-33179	78917
1077	Grand-Lancy, Pontets	847	-23148	-16523	76	-15521	-8029	-5467	35312	103077	-67765	138389
7423	Grand-Lancy, Palettes	-11642	-32421	-12109	-2594	-14624	-4583	-7231	442	85646	-85204	86088

line80_directionA_netboarding_TOTALS

IdArret	StationName	AM peak	Day period	Evening	Other	PM peak	Saturdays	Sundays	NbMontees	NbDescente	NetBoardingTotal	TotalMovement
2321	Geneve, Bel-Air	4671	31759	44342	3250	29391	17725	9029	141072	905	140167	141977

1310	Genéve, Stand	16300	42638	45493	6408	32421	18109	10774	176204	4061	172143	180265
1022	Genéve, Palladium	4386	12639	10658	1289	10028	4123	2373	59038	13542	45496	72580
720	Geneve, Jonction	26932	50696	29510	10178	27864	17316	9958	207447	34993	172454	242440
7166	Genéve, Queue-d'Arve	-2531	2026	4747	-859	8428	-151	524	37727	25543	12184	63270
522	Lancy-Pont-Rouge, gare/Etoile	-1381	27033	20392	1621	22545	9611	5231	172458	87406	85052	259864
1298	Grand-Lancy, Stade de Geneve	-5079	-15965	-6750	-2339	-6277	-10218	-5104	87582	139314	-51732	226896
122	Lancy-Bachet, gare	25787	28364	21915	9961	22975	7415	5097	218211	96697	121514	314908
335	Plan-les-Ouates, Communes-Réunies	2691	5400	-1399	1204	2181	63	144	50964	40680	10284	91644
3303	Plan-les-Ouates, Aviateurs	-3736	-12862	-15446	-312	-7631	-5597	-3654	32094	81332	-49238	113426
1037	Plan-les-Ouates, mairie	-3502	-7927	-7412	-788	-6175	-3645	-2033	7317	38799	-31482	46116
1414	Plan-les-Ouates, Vélodrome	-15029	-18154	-12590	-7464	-6345	-5355	-3816	28512	97265	-68753	125777
52	Plan-les-Ouates, Arare	-22609	-17091	-6553	-7342	-622	-4370	-3353	21315	83255	-61940	104570
630	Plan-les-Ouates, Galaise	-11602	-5734	1590	-2678	4475	-825	-334	17810	32918	-15108	50728
2117	Plan-les-Ouates, Au Contour	-337	-1046	-710	-450	-602	-273	-138	492	4048	-3556	**4540**
782	Perly, En Louche	-1921	-11346	-9417	-797	-7867	-3161	-2113	4641	41263	-36622	45904
3301	Perly, Ravieres	-3097	-16975	-15827	-1034	-11829	-5772	-3399	14137	72070	-57933	86207
982	Perly, douane	-2120	-6102	-3331	-754	-3473	-1685	-1015	5317	23797	-18480	29114
6495	Saint-Julien-en-Genevois, douane	-805	-10789	-13591	-256	-11295	-3966	-2069	3168	45939	-42771	49107
6497	Saint-Julien-en-Gene	-593	-5886	-20864	-667	-7470	-6657	-3474	19731	65342	-45611	85073

	vois, Hutins											
2521	Saint-Julien-en-Genevois, centre	-6853	-37293	-61469	-2549	-43382	-16951	-9506	5943	183946	-178003	189889
7444	Saint-Julien-en-Genevois, SNCF	-5335	-28439	-42550	-2287	-30569	-12125	-7608	2967	131880	-128913	134847

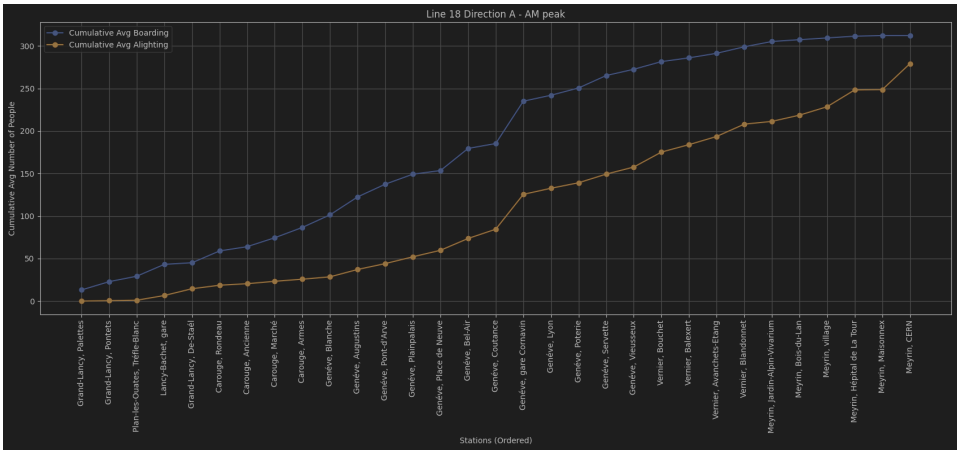
line80_directionR_netboarding_TOTALS

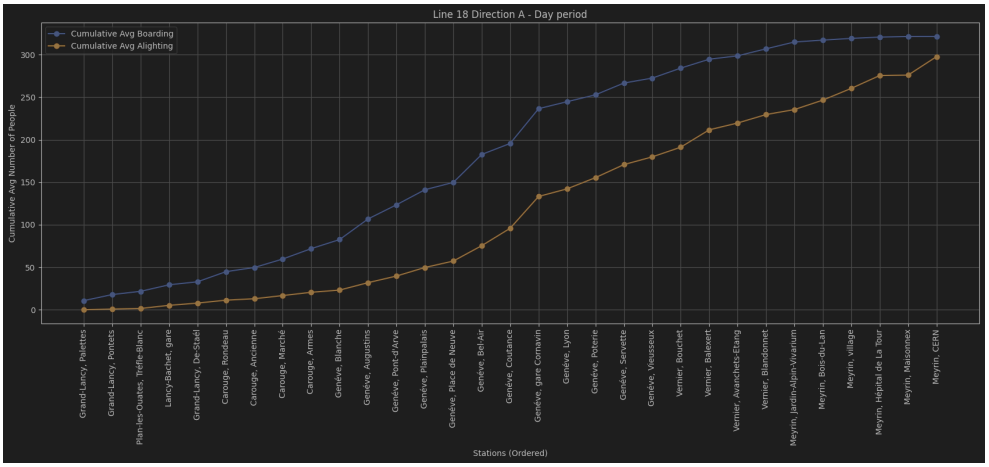
IdArret	StationName	AM peak	Day period	Evening	Other	PM peak	Saturdays	Sundays	NbMontees	NbDescente	NetBoardingTotal	TotalMovement
7444	Saint-Julien-en-Genevois, SNCF	44330	39799	13076	26119	11945	13804	7546	161028	4409	156619	165437
2522	Saint-Julien-en-Genevois, centre	44484	45140	10649	20445	15112	14924	7076	164150	6320	157830	170470
6498	Saint-Julien-en-Genevois, Hutins	7289	11342	2174	9199	2590	6748	3219	67916	25355	42561	93271
6496	Saint-Julien-en-Genevois, douane	9834	10074	1355	5780	1926	2558	1459	35535	2549	32986	38084
983	Perly, douane	4479	8842	2449	1156	1975	2578	1306	30000	7215	22785	37215
3302	Perly, Ravieres	12188	18046	5900	5131	5350	5349	2784	68638	13890	54748	82528
780	Perly, En Louche	7656	11806	2491	2089	4608	2846	1676	37184	4012	33172	41196
353	Plan-les-Ouates, Au Contour	1094	1266	360	363	397	316	212	4799	791	4008	**5590**
631	Plan-les-Ouates, Galaise	-6155	3490	3935	-2717	6431	851	415	26316	20066	6250	46382
53	Plan-les-Ouates, Arare	-2006	17196	9641	-3249	15906	3478	2552	67839	24321	43518	92160
1415	Plan-les-Ouates, Vélodrome	4725	20974	9221	1377	18555	4675	2942	82833	20364	62469	103197
1038	Plan-les-Ouates, mairie	4138	9328	4504	807	4512	3297	1765	36333	7982	28351	44315

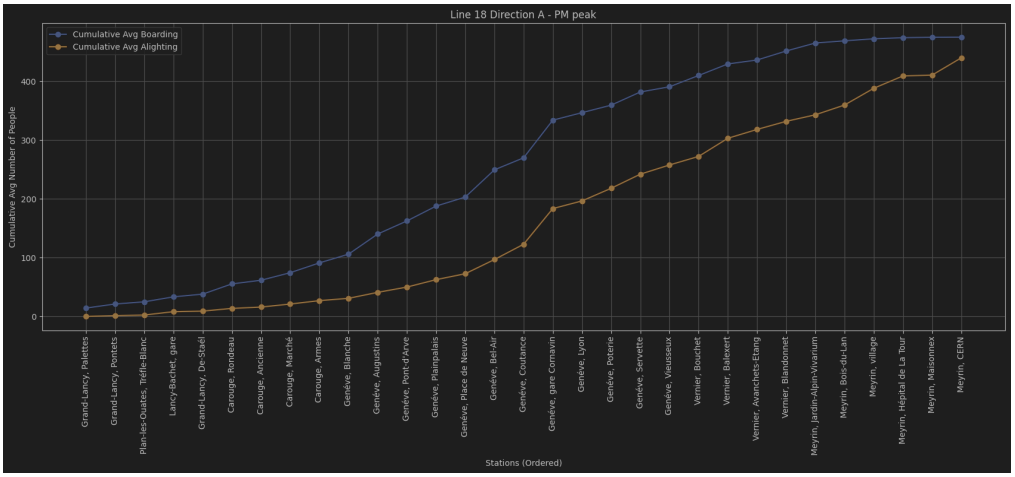
3304	Plan-les-Ouates, Aviateurs	5722	20557	7463	2777	9171	6811	3829	84084	27754	56330	111838
334	Plan-les-Ouates, Communes-Réunies	-1250	-1710	-292	10	-940	516	279	33241	36628	-3387	69869
121	Lancy-Bachet, gare	-37992	-48621	-19968	-17221	-28300	-11946	-7084	98100	269232	-171132	367332
1301	Grand-Lancy, Stade de Geneve	2377	5242	10385	-202	4780	5486	3525	104847	73254	31593	178101
523	Lancy-Pont-Rouge, gare/Etoile	-13674	-19001	-2239	-5958	-2218	-6506	-3491	82303	135390	-53087	217693
7165	Genéve, Queue-d'Arve	-12435	-8164	-768	-3786	-579	-1679	-883	19711	48005	-28294	67716
721	Geneve, Jonction	-15385	-43902	-22198	-3418	-21137	-11732	-7107	73940	198819	-124879	272759
1023	Genéve, Palladium	-5019	-5231	-2268	-440	506	-1611	-201	39668	53932	-14264	93600
1308	Genéve, Stand	-19517	-30037	-13927	-4090	-13799	-9865	-6480	6372	104087	-97715	110459
2321	Geneve, Bel-Air	-34376	-75408	-38690	-12601	-34964	-31710	-16375	332	244456	-244124	244788

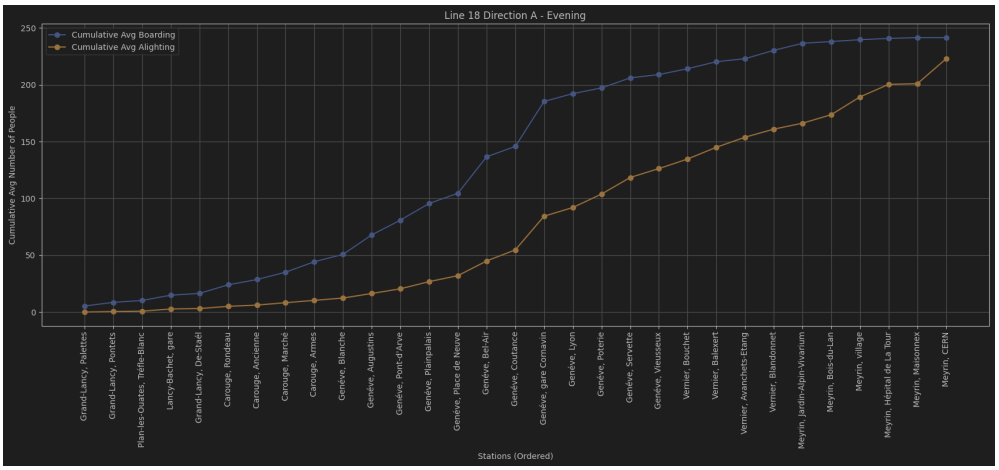
To visualize this, I graphed the average (rather than raw) cumulative number of passengers that mounted and descended by each stop for each line and direction for each time period.

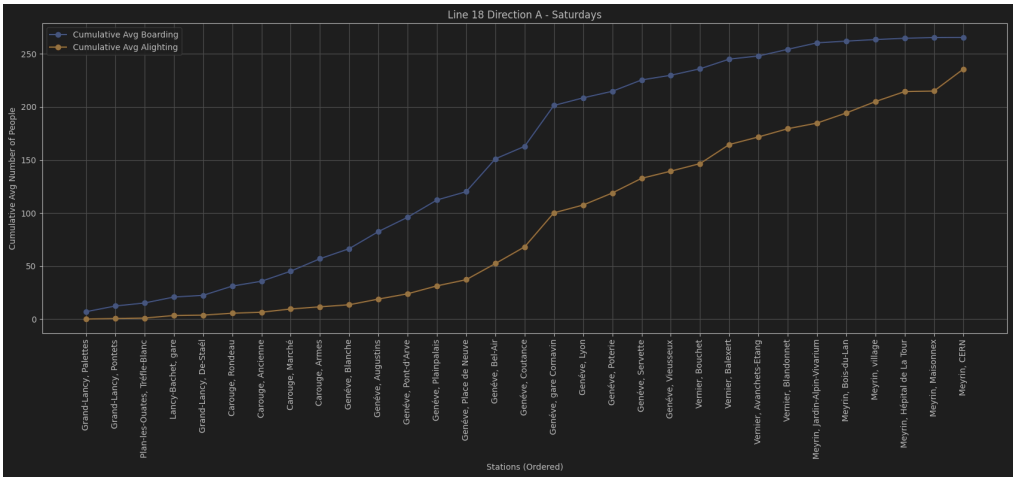
The scheduled number of buses operating in both directions is always the same so this does not pose any other limitations.

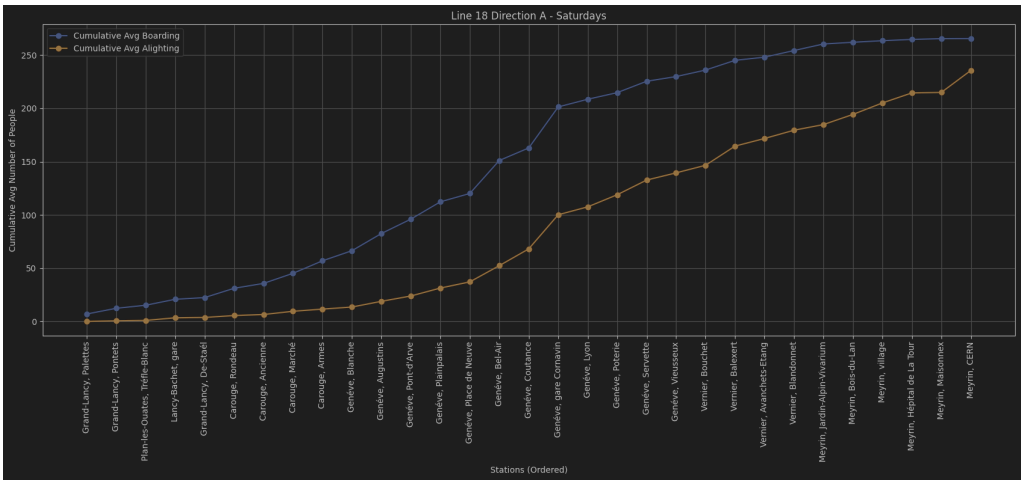


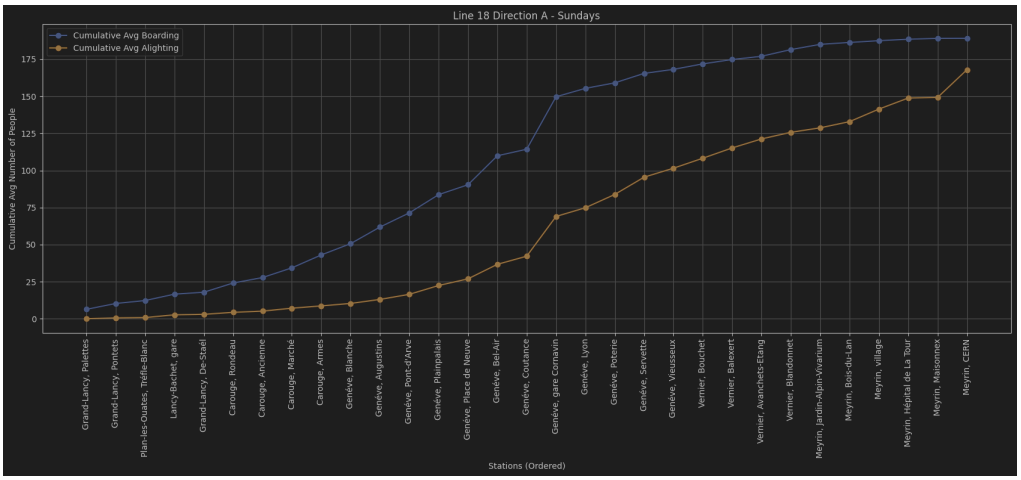

Line 18 Direction A - Day period
Cumulative Avg Boarding
Cumulative Avg Alighting
Cumulative Avg Number of People
0
50
100
150
200
250
300
Grand-Lancy, Palettes
Grand-Lancy, Pontets
Plan-les-Ouates, Trèfle-Blanc
Lancy-Bachet, gare
Grand-Lancy, De-Staël
Carouge, Rondeau
Carouge, Ancienne
Carouge, Marché
Carouge, Armes
Genève, Blanche
Genève, Augustins
Genève, Pont-d'Arve
Genève, Plainpalais
Genève, Place de Neuve
Genève, Bel-Air
Genève, Coutance
Genève, gare Cornavin
Genève, Lyon
Genève, Poterie
Genève, Servette
Genève, Vieusseux
Vernier, Bouchet
Vernier, Balexert
Vernier, Avanchets-Etang
Vernier, Blandonnet
Meyrin, Jardin-Alpin-Vivarium
Meyrin, Bois-du-Lan
Meyrin, village
Meyrin, Hôpital de La Tour
Meyrin, Maisonnex
Meyrin, CERN
Stations (Ordered)

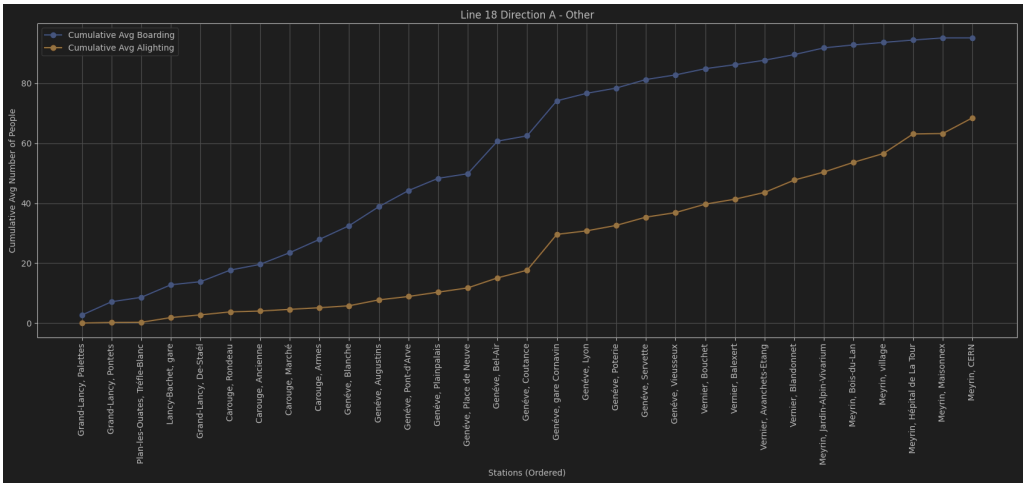

Line 18 Direction A - PM peak
Cumulative Avg Boarding
Cumulative Avg Alighting
Cumulative Avg Number of People
0
100
200
300
400
Grand-Lancy, Palettes
Grand-Lancy, Pontets
Plan-les-Ouates, Trèfle-Blanc
Lancy-Bachet, gare
Grand-Lancy, De-Staël
Carouge, Rondeau
Carouge, Ancienne
Carouge, Marché
Carouge, Armes
Genève, Blanche
Genève, Augustins
Genève, Pont-d'Arve
Genève, Plainpalais
Genève, Place de Neuve
Genève, Bel-Air
Genève, Coutance
Genève, gare Cornavin
Genève, Lyon
Genève, Poterie
Genève, Servette
Genève, Vieusseux
Vernier, Bouchet
Vernier, Balexert
Vernier, Avanchets-Etang
Vernier, Blandonnet
Meyrin, Jardin-Alpin-Vivarium
Meyrin, Bois-du-Lan
Meyrin, village
Meyrin, Hôpital de La Tour
Meyrin, Maisonnex
Meyrin, CERN
Stations (Ordered)

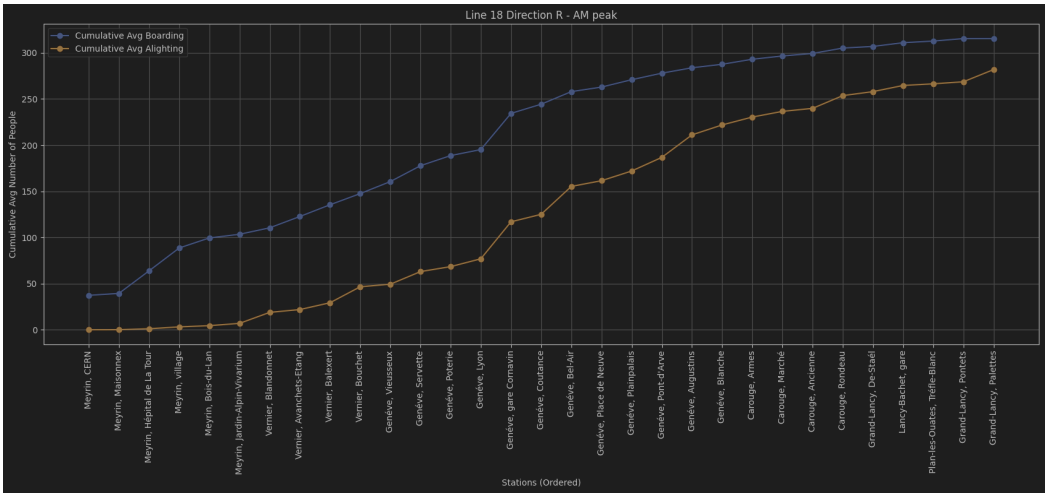

Line 18 Direction A - Evening
Cumulative Avg Boarding
Cumulative Avg Alighting
Cumulative Avg Number of People
0
50
100
150
200
250
Grand-Lancy, Palettes
Grand-Lancy, Pontets
Plan-les-Ouates, Trèfle-Blanc
Lancy-Bachet, gare
Grand-Lancy, De-Staël
Carouge, Rondeau
Carouge, Ancienne
Carouge, Marché
Carouge, Armes
Genève, Blanche
Genève, Augustins
Genève, Pont-d'Arve
Genève, Plainpalais
Genève, Place de Neuve
Genève, Bel-Air
Genève, Coutance
Genève, gare Cornavin
Genève, Lyon
Genève, Poterie
Genève, Servette
Genève, Vieusseux
Vernier, Bouchet
Vernier, Balexert
Vernier, Avanchets-Etang
Vernier, Blandonnet
Meyrin, Jardin-Alpin-Vivarium
Meyrin, Bois-du-Lan
Meyrin, village
Meyrin, Hôpital de La Tour
Meyrin, Maisonnex
Meyrin, CERN
Stations (Ordered)

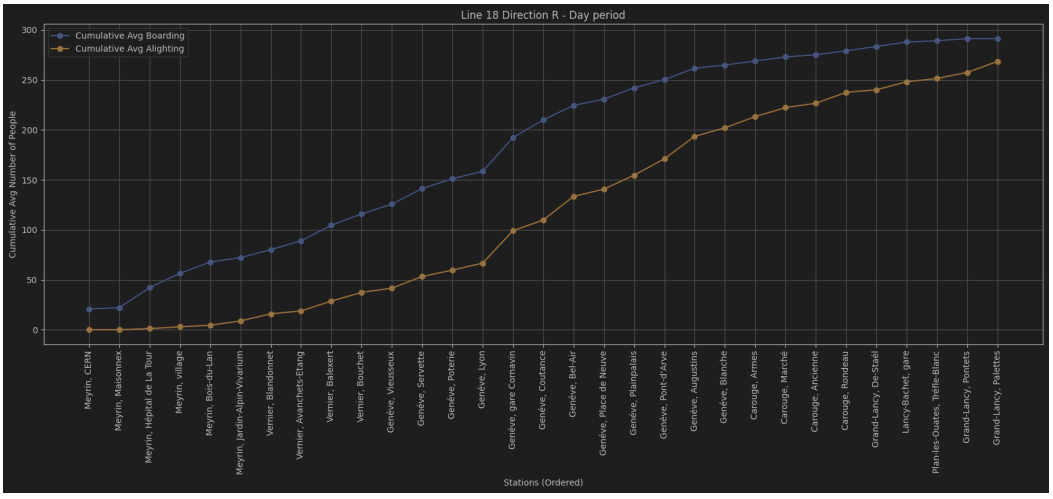

Line 18 Direction A - Saturdays
Cumulative Avg Boarding
Cumulative Avg Alighting
Cumulative Avg Number of People
0
50
100
150
200
250
Grand-Lancy, Palettes
Grand-Lancy, Pontets
Plan-les-Ouates, Trèfle-Blanc
Lancy-Bachet, gare
Grand-Lancy, De-Staël
Carouge, Rondeau
Carouge, Ancienne
Carouge, Marché
Carouge, Armes
Genève, Blanche
Genève, Augustins
Genève, Pont-d'Arve
Genève, Plainpalais
Genève, Place de Neuve
Genève, Bel-Air
Genève, Coutance
Genève, gare Cornavin
Genève, Lyon
Genève, Poterie
Genève, Servette
Genève, Vieusseux
Vernier, Bouchet
Vernier, Balexert
Vernier, Avanchets-Etang
Vernier, Blandonnet
Meyrin, Jardin-Alpin-Vivarium
Meyrin, Bois-du-Lan
Meyrin, village
Meyrin, Hôpital de La Tour
Meyrin, Maisonnex
Meyrin, CERN
Stations (Ordered)

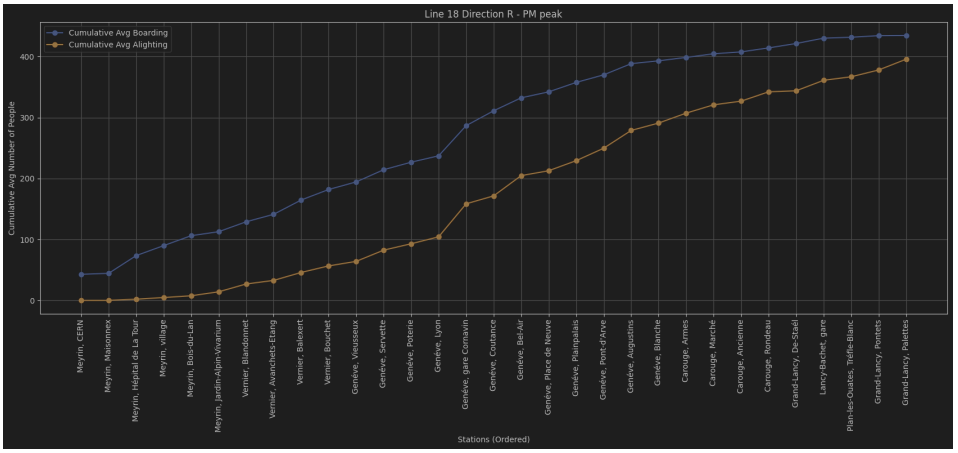

Line 18 Direction A - Saturdays
Cumulative Avg Boarding
Cumulative Avg Alighting
Cumulative Avg Number of People
0
50
100
150
200
250
Grand-Lancy, Palettes
Grand-Lancy, Pontets
Plan-les-Ouates, Trèfle-Blanc
Lancy-Bachet, gare
Grand-Lancy, De-Staël
Carouge, Rondeau
Carouge, Ancienne
Carouge, Marché
Carouge, Armes
Genève, Blanche
Genève, Augustins
Genève, Pont-d'Arve
Genève, Plainpalais
Genève, Place de Neuve
Genève, Bel-Air
Genève, Coutance
Genève, gare Cornavin
Genève, Lyon
Genève, Poterie
Genève, Servette
Genève, Vieusseux
Vernier, Bouchet
Vernier, Balexert
Vernier, Avanchets-Etang
Vernier, Blandonnet
Meyrin, Jardin-Alpin-Vivarium
Meyrin, Bois-du-Lan
Meyrin, village
Meyrin, Hôpital de La Tour
Meyrin, Maisonnex
Meyrin, CERN
Stations (Ordered)

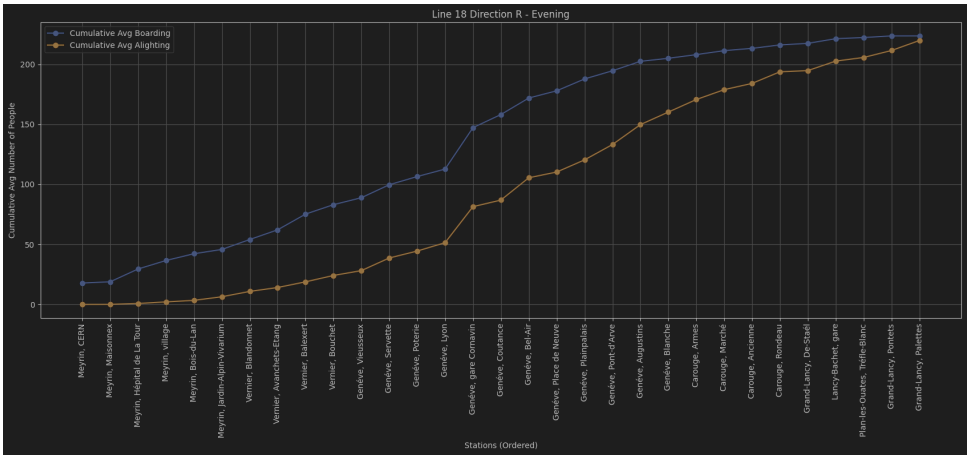

Line 18 Direction A - Sundays
Cumulative Avg Boarding
Cumulative Avg Alighting
Cumulative Avg Number of People
0
25
50
75
100
125
150
175
Grand-Lancy, Palettes
Grand-Lancy, Pontets
Plan-les-Ouates, Trèfle-Blanc
Lancy-Bachet, gare
Grand-Lancy, De-Staël
Carouge, Rondeau
Carouge, Ancienne
Carouge, Marché
Carouge, Armes
Genève, Blanche
Genève, Augustins
Genève, Pont-d'Arve
Genève, Plainpalais
Genève, Place de Neuve
Genève, Bel-Air
Genève, Coutance
Genève, gare Cornavin
Genève, Lyon
Genève, Poterie
Genève, Servette
Genève, Vieusseux
Vernier, Bouchet
Vernier, Balexert
Vernier, Avanchets-Etang
Vernier, Blandonnet
Meyrin, Jardin-Alpin-Vivarium
Meyrin, Bois-du-Lan
Meyrin, village
Meyrin, Hôpital de La Tour
Meyrin, Maisonnex
Meyrin, CERN
Stations (Ordered)

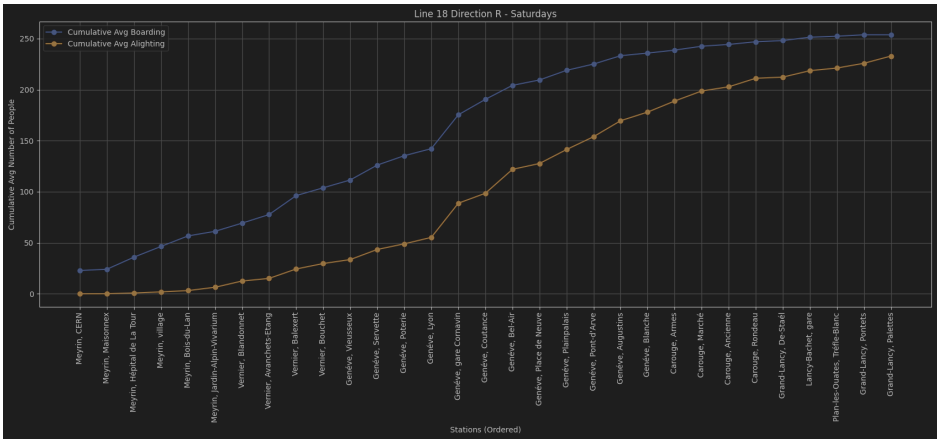

Line 18 Direction A - Other
Cumulative Avg Boarding
Cumulative Avg Alighting
Cumulative Avg Number of People
0
20
40
60
80
Grand-Lancy, Palettes
Grand-Lancy, Pontets
Plan-les-Ouates, Trèfle-Blanc
Lancy-Bachet, gare
Grand-Lancy, De-Staël
Carouge, Rondeau
Carouge, Ancienne
Carouge, Marché
Carouge, Armes
Genève, Blanche
Genève, Augustins
Genève, Pont-d'Arve
Genève, Plainpalais
Genève, Place de Neuve
Genève, Bel-Air
Genève, Coutance
Genève, gare Cornavin
Genève, Lyon
Genève, Potterie
Genève, Servette
Genève, Vieusseux
Vernier, Bouchet
Vernier, Balexert
Vernier, Avanchets-Etang
Vernier, Blandonnet
Meyrin, Jardin-Alpin-Vivarium
Meyrin, Bois-du-Lan
Meyrin, village
Meyrin, Hôpital de La Tour
Meyrin, Maisonnex
Meyrin, CERN
Stations (Ordered)

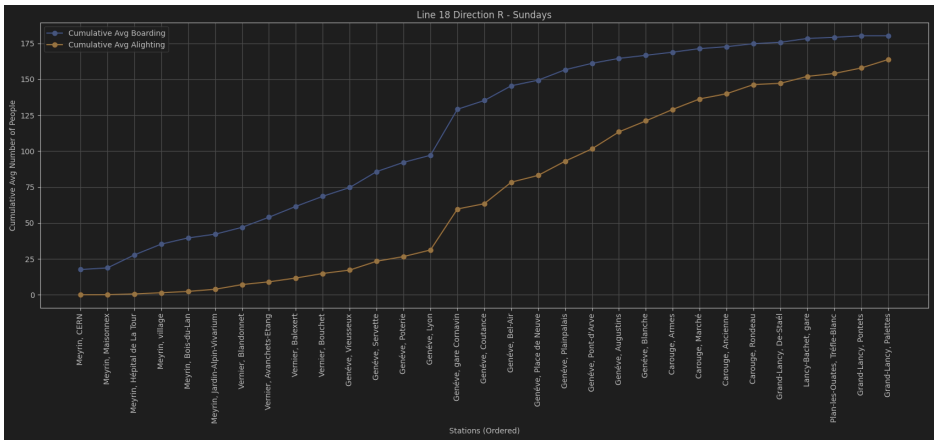

Line 18 Direction R - AM peak
Cumulative Avg Boarding
Cumulative Avg Alighting
Cumulative Avg Number of People
0
50
100
150
200
250
300
Meyrin, CERN
Meyrin, Maisonnex
Meyrin, Hôpital de La Tour
Meyrin, village
Meyrin, Bois-du-Lan
Meyrin, Jardin-Alpin-Vivarium
Vernier, Blandonnet
Vernier, Avanchets-Etang
Vernier, Balexert
Vernier, Bouchet
Genève, Vieusseux
Genève, Servette
Genève, Potterie
Genève, Lyon
Genève, gare Cornavin
Genève, Coutance
Genève, Bel-Air
Genève, Place de Neuve
Genève, Plainpalais
Genève, Pont-d'Arve
Genève, Augustins
Genève, Blanche
Carouge, Armes
Carouge, Marché
Carouge, Ancienne
Carouge, Rondeau
Grand-Lancy, De-Staël
Lancy-Bachet, gare
Plan-les-Ouates, Trèfle-Blanc
Grand-Lancy, Pontets
Grand-Lancy, Palettes
Stations (Ordered)

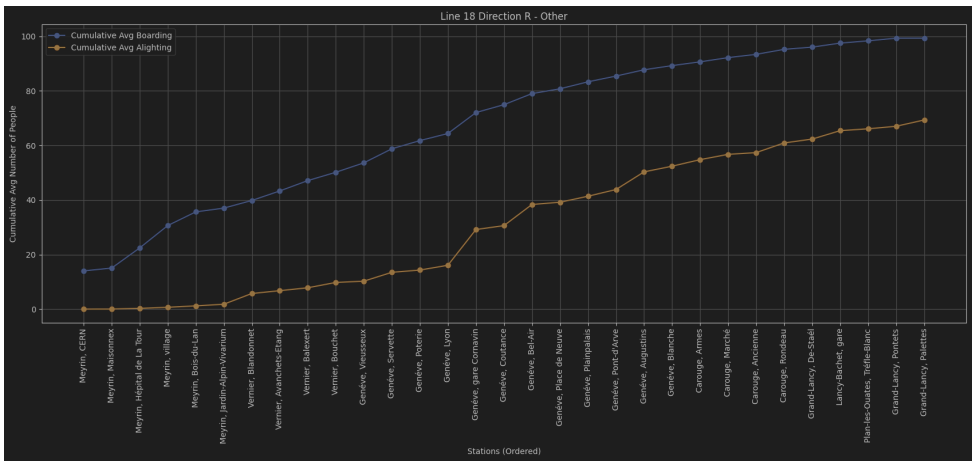

Line 18 Direction R - Day period
Cumulative Avg Boarding
Cumulative Avg Alighting
Cumulative Avg Number of People
0
50
100
150
200
250
300
Meyrin, CERN
Meyrin, Maisonnex
Meyrin, Hôpital de La Tour
Meyrin, village
Meyrin, Bois-du-Lan
Meyrin, Jardin-Alpin-Vivarium
Vernier, Blandonnet
Vernier, Avanchets-Etang
Vernier, Balexert
Vernier, Bouchet
Genève, Vieusseux
Genève, Servette
Genève, Potterie
Genève, Lyon
Genève, gare Cornavin
Genève, Coutance
Genève, Bel-Air
Genève, Place de Neuve
Genève, Plainpalais
Genève, Pont-d'Arve
Genève, Augustins
Genève, Blanche
Carouge, Armes
Carouge, Marché
Carouge, Ancienne
Carouge, Rondeau
Grand-Lancy, De-Staël
Lancy-Bachet, gare
Plan-les-Ouates, Trèfle-Blanc
Grand-Lancy, Pontets
Grand-Lancy, Palettes
Stations (Ordered)

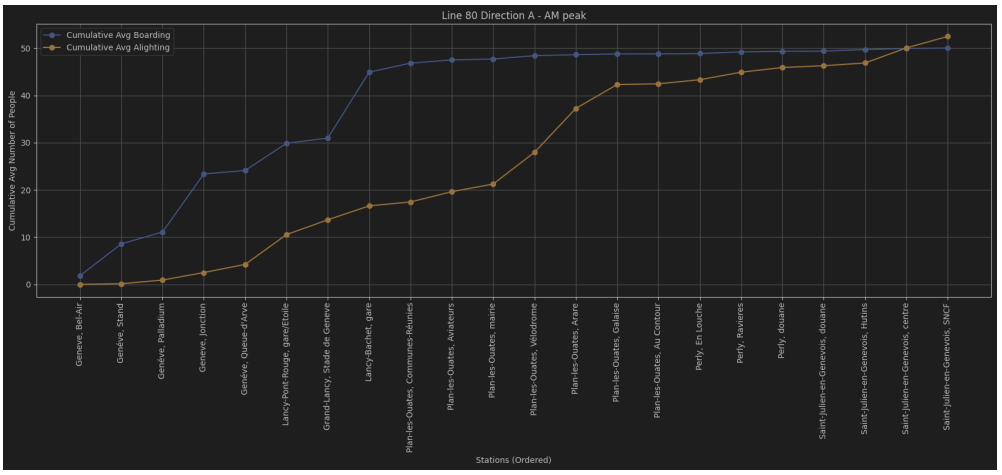

Line 18 Direction R - PM peak
Cumulative Avg Boarding
Cumulative Avg Alighting
Cumulative Avg Number of People
Stations (Ordered)

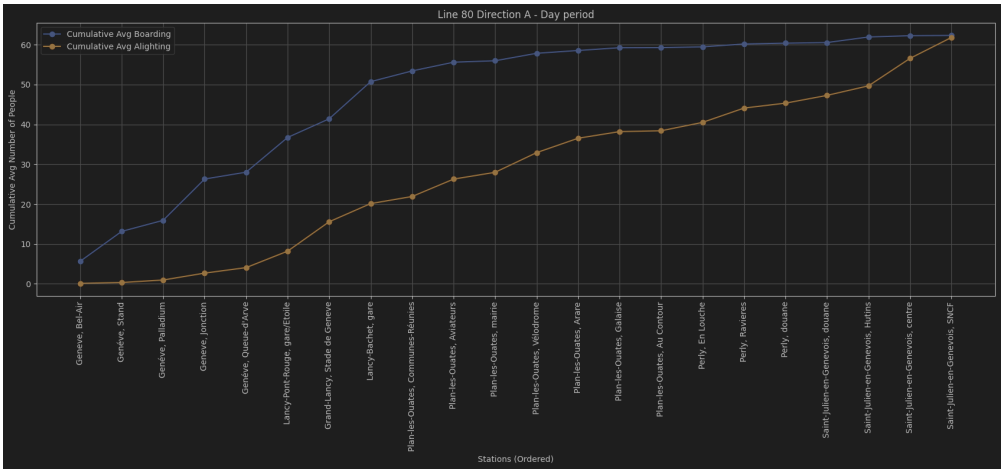

Line 18 Direction R - Evening
Cumulative Avg Boarding
Cumulative Avg Alighting
Cumulative Avg Number of People
Stations (Ordered)

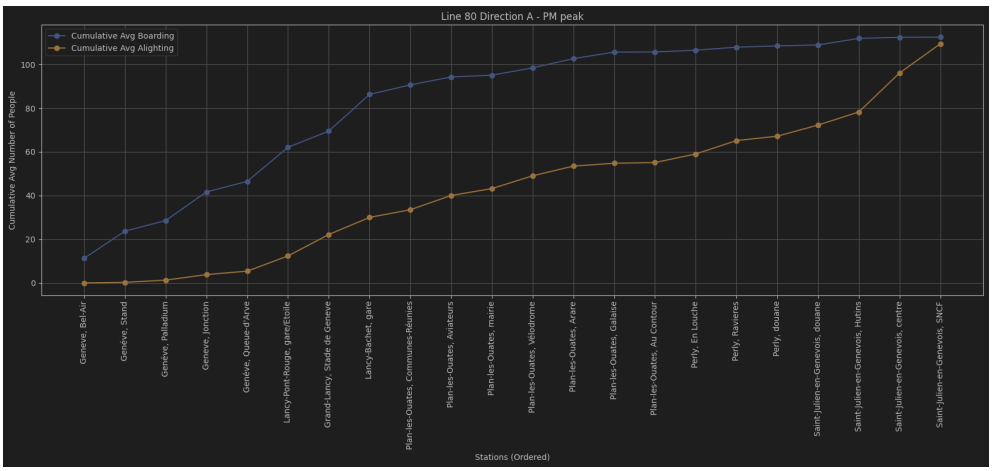

Line 18 Direction R - Saturdays
Cumulative Avg Boarding
Cumulative Avg Alighting
Cumulative Avg Number of People
Stations (Ordered)

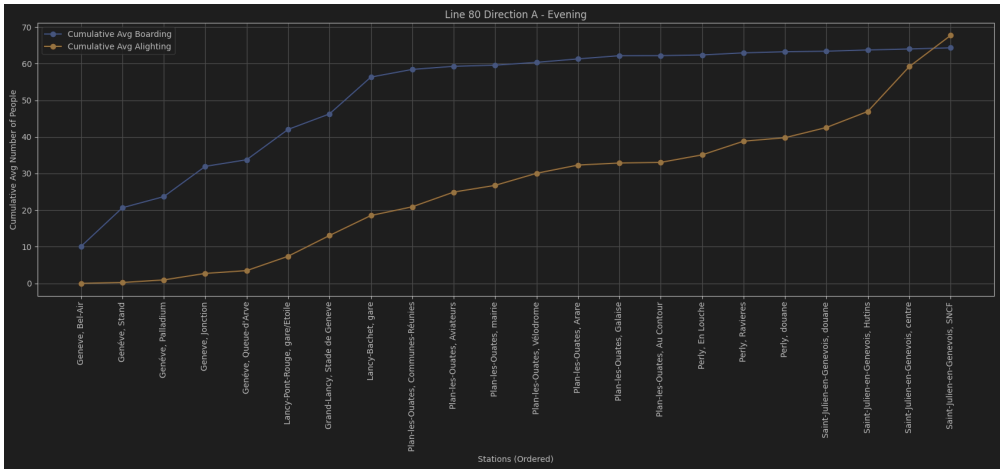

Line 18 Direction R - Sundays
Cumulative Avg Boarding
Cumulative Avg Alighting
Cumulative Avg Number of People
Stations (Ordered)

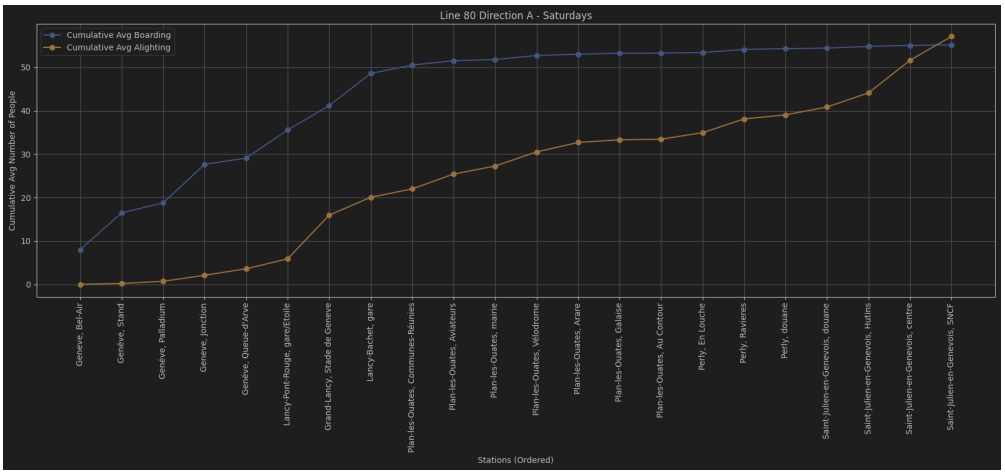

Line 18 Direction R - Other
Cumulative Avg Boarding
Cumulative Avg Alighting
Cumulative Avg Number of People
Stations (Ordered)
Meyrin, CERN
Meyrin, Maisonnex
Meyrin, Hôpital de La Tour
Meyrin, village
Meyrin, Bois-du-Lan
Meyrin, Jardin-Alpin-Vivarium
Vernier, Blandonnet
Vernier, Avanchets-Etang
Vernier, Balexert
Vernier, Bouchet
Genève, Vieusseux
Genève, Servette
Genève, Prieuré
Genève, Lyon
Genève, gare Cornavin
Genève, Coutance
Genève, Bel-Air
Genève, Place de Neuve
Genève, Plainpalais
Genève, Pont-d'Arve
Genève, Augustins
Genève, Blanche
Carouge, Armes
Carouge, Marché
Carouge, Ancienne
Carouge, Rondeau
Grand-Lancy, De-Staël
Lancy-Bachet, gare
Plan-les-Ouates, Tieffe-Blanc
Grand-Lancy, Pontets
Grand-Lancy, Palettes

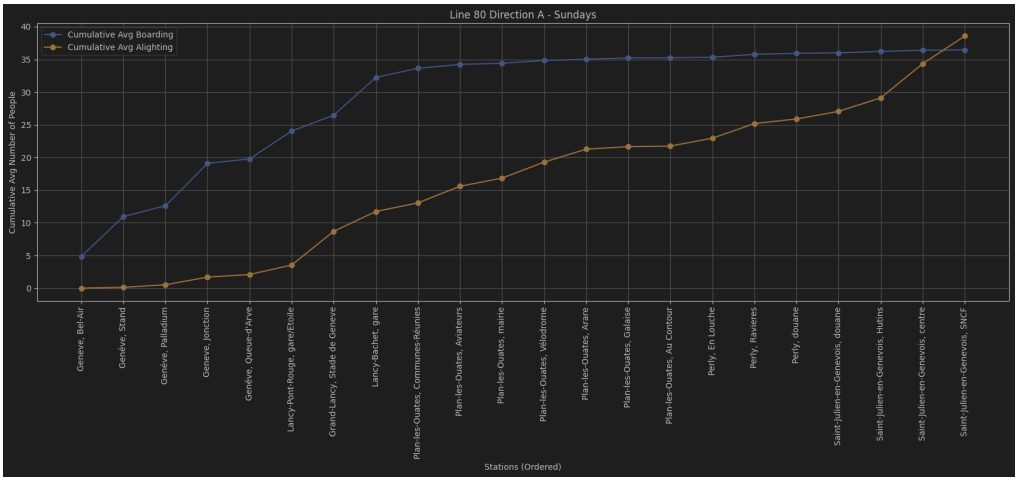

Line 80 Direction A - AM peak
Cumulative Avg Boarding
Cumulative Avg Alighting
Cumulative Avg Number of People
Stations (Ordered)
Geneve, Bel-Air
Genève, Stand
Genève, Palladium
Geneve, Jonction
Genève, Queue-d'Arve
Lancy-Pont-Rouge, gare/Etoile
Grand-Lancy, Stade de Geneve
Lancy-Bachet, gare
Plan-les-Ouates, Communes-Réunies
Plan-les-Ouates, Aviateurs
Plan-les-Ouates, mairie
Plan-les-Ouates, Vélodrome
Plan-les-Ouates, Arare
Plan-les-Ouates, Galaise
Plan-les-Ouates, Au Contour
Perly, En Louche
Perly, Ravieres
Perly, douane
Saint-Julien-en-Genevois, douane
Saint-Julien-en-Genevois, Hutins
Saint-Julien-en-Genevois, centre
Saint-Julien-en-Genevois, SNCF

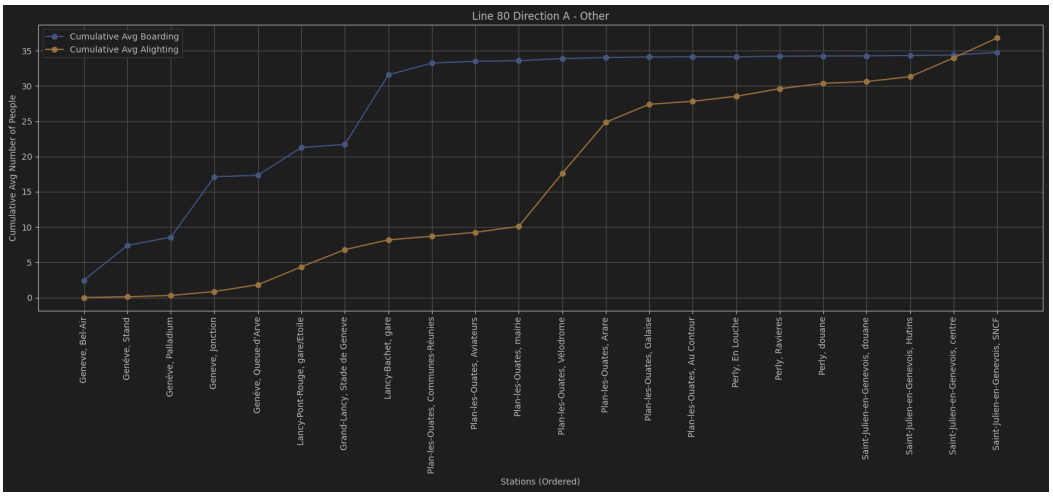

Line 80 Direction A - Day period
Cumulative Avg Boarding
Cumulative Avg Alighting
Cumulative Avg Number of People
Stations (Ordered)
Geneve, Bel-Air
Genève, Stand
Genève, Palladium
Geneve, Jonction
Genève, Queue-d'Arve
Lancy-Pont-Rouge, gare/Etoile
Grand-Lancy, Stade de Geneve
Lancy-Bachet, gare
Plan-les-Ouates, Communes-Réunies
Plan-les-Ouates, Aviateurs
Plan-les-Ouates, mairie
Plan-les-Ouates, Vélodrome
Plan-les-Ouates, Arare
Plan-les-Ouates, Galaise
Plan-les-Ouates, Au Contour
Perly, En Louche
Perly, Ravieres
Perly, douane
Saint-Julien-en-Genevois, douane
Saint-Julien-en-Genevois, Hutins
Saint-Julien-en-Genevois, centre
Saint-Julien-en-Genevois, SNCF

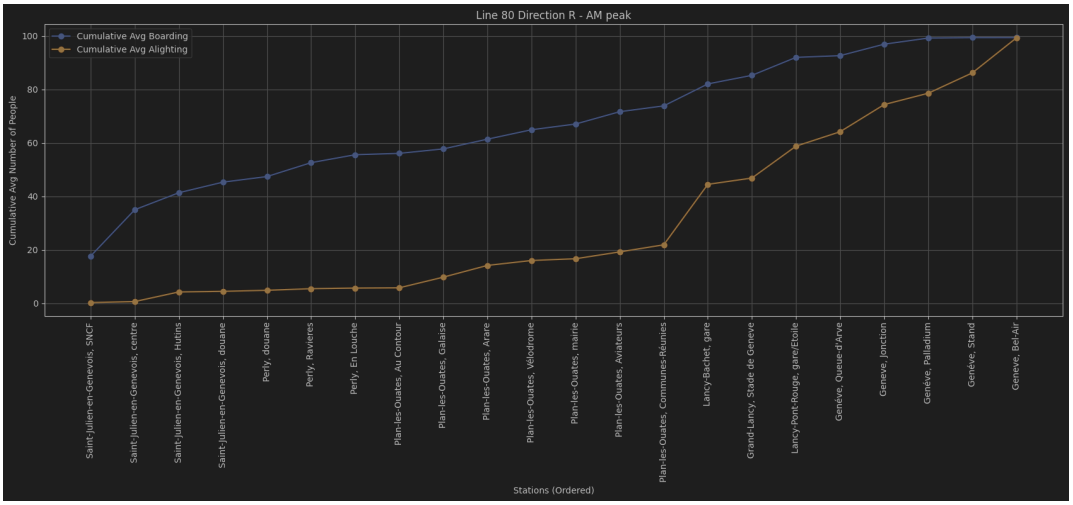

Line 80 Direction A - PM peak
Cumulative Avg Boarding
Cumulative Avg Alighting
Cumulative Avg Number of People
Stations (Ordered)
Genève, Bel-Air
Genève, Stand
Genève, Palladium
Genève, Jonction
Genève, Queue-d'Arve
Lancy-Pont-Rouge, gare/Etoile
Grand-Lancy, Stade de Genève
Lancy-Bachet, gare
Plan-les-Ouates, Communes-Réunies
Plan-les-Ouates, Aviateurs
Plan-les-Ouates, mairie
Plan-les-Ouates, Vélodrome
Plan-les-Ouates, Arare
Plan-les-Ouates, Galaise
Plan-les-Ouates, Au Contour
Perly, En Louche
Perly, Ravières
Perly, douane
Saint-Julien-en-Genevois, douane
Saint-Julien-en-Genevois, Hutins
Saint-Julien-en-Genevois, centre
Saint-Julien-en-Genevois, SNCF

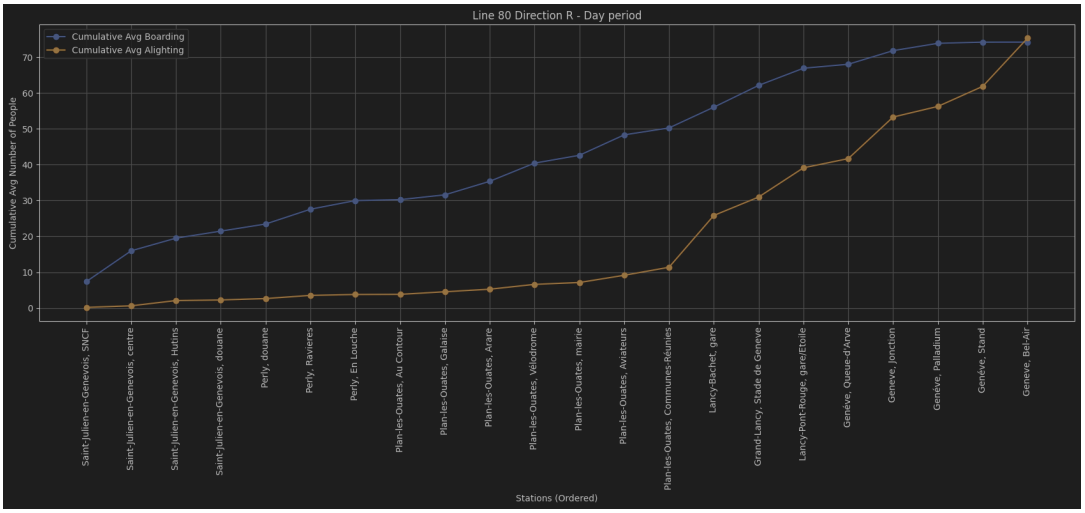

Line 80 Direction A - Evening
Cumulative Avg Boarding
Cumulative Avg Alighting
Cumulative Avg Number of People
Stations (Ordered)
Genève, Bel-Air
Genève, Stand
Genève, Palladium
Genève, Jonction
Genève, Queue-d'Arve
Lancy-Pont-Rouge, gare/Etoile
Grand-Lancy, Stade de Genève
Lancy-Bachet, gare
Plan-les-Ouates, Communes-Réunies
Plan-les-Ouates, Aviateurs
Plan-les-Ouates, mairie
Plan-les-Ouates, Vélodrome
Plan-les-Ouates, Arare
Plan-les-Ouates, Galaise
Plan-les-Ouates, Au Contour
Perly, En Louche
Perly, Ravières
Perly, douane
Saint-Julien-en-Genevois, douane
Saint-Julien-en-Genevois, Hutins
Saint-Julien-en-Genevois, centre
Saint-Julien-en-Genevois, SNCF

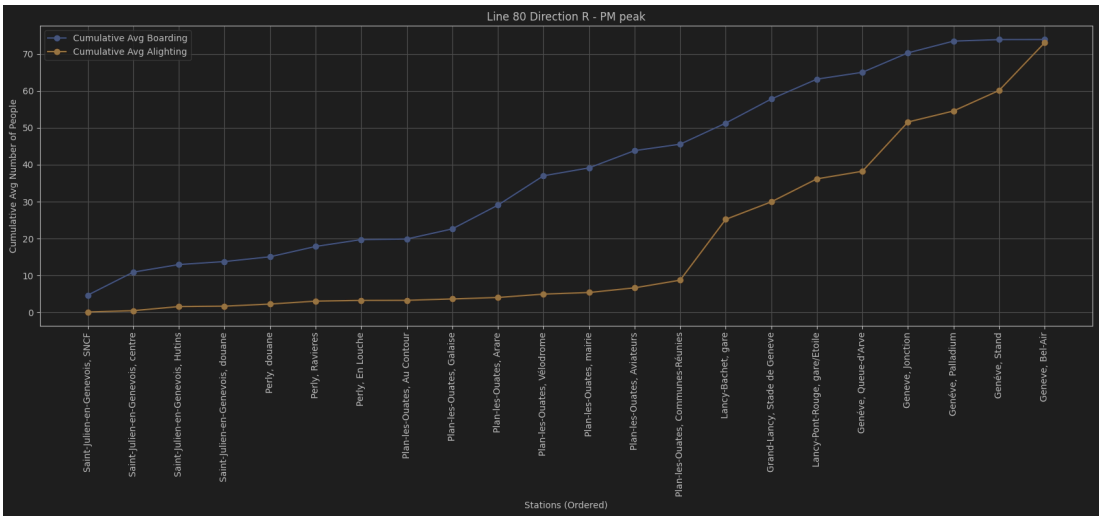

Line 80 Direction A - Saturdays
Cumulative Avg Boarding
Cumulative Avg Alighting
Cumulative Avg Number of People
Stations (Ordered)
Genève, Bel-Air
Genève, Stand
Genève, Palladium
Genève, Jonction
Genève, Queue-d'Arve
Lancy-Pont-Rouge, gare/Etoile
Grand-Lancy, Stade de Genève
Lancy-Bachet, gare
Plan-les-Ouates, Communes-Réunies
Plan-les-Ouates, Aviateurs
Plan-les-Ouates, mairie
Plan-les-Ouates, Vélodrome
Plan-les-Ouates, Arare
Plan-les-Ouates, Galaise
Plan-les-Ouates, Au Contour
Perly, En Louche
Perly, Ravières
Perly, douane
Saint-Julien-en-Genevois, douane
Saint-Julien-en-Genevois, Hutins
Saint-Julien-en-Genevois, centre
Saint-Julien-en-Genevois, SNCF

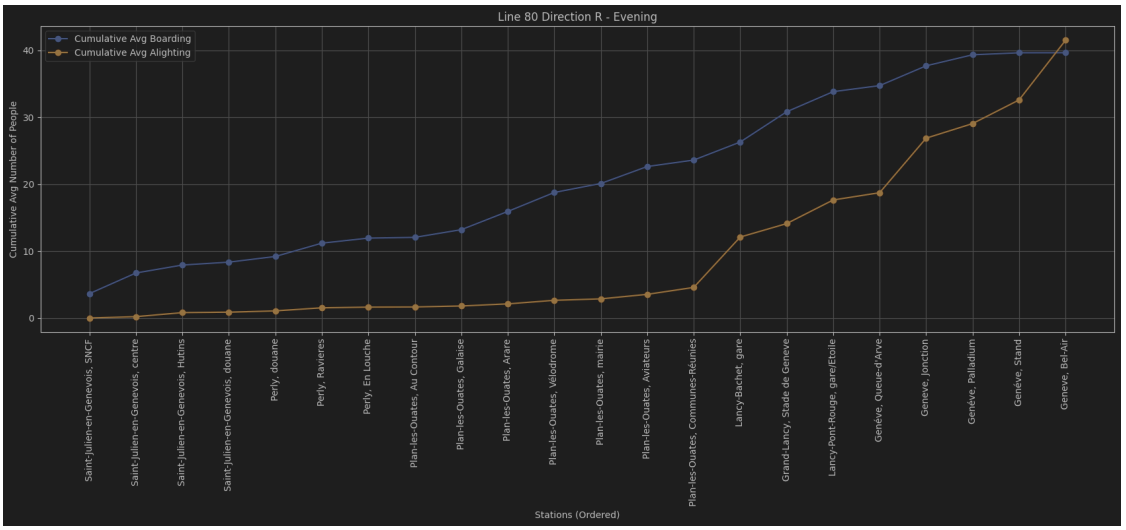

Line 80 Direction A - Sundays
Cumulative Avg Boarding
Cumulative Avg Alighting
Cumulative Avg Number of People
Stations (Ordered)
Geneve, Bel-Air
Genève, Stand
Genève, Palladium
Geneve, Jonction
Genève, Queue-d'Arve
Lancy-Pont-Rouge, gare/Etoile
Grand-Lancy, Stade de Geneve
Lancy-Bachet, gare
Plan-les-Ouates, Communes-Réunies
Plan-les-Ouates, Aviateurs
Plan-les-Ouates, mairie
Plan-les-Ouates, Vélodrome
Plan-les-Ouates, Arare
Plan-les-Ouates, Galaise
Plan-les-Ouates, Au Contour
Perly, En Louche
Perly, Ravieres
Perly, douane
Saint-Julien-en-Genevois, douane
Saint-Julien-en-Genevois, Hutins
Saint-Julien-en-Genevois, centre
Saint-Julien-en-Genevois, SNCF

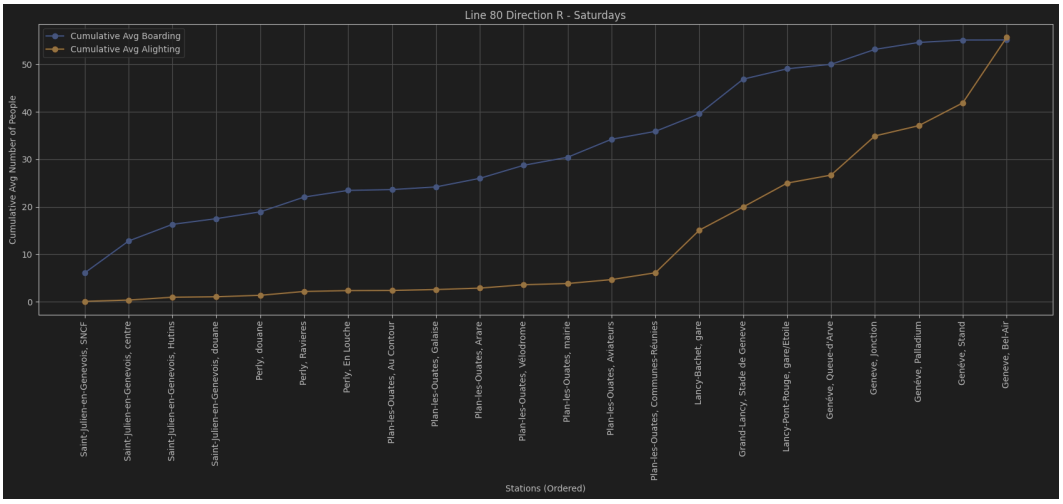

Line 80 Direction A - Other
Cumulative Avg Boarding
Cumulative Avg Alighting
Cumulative Avg Number of People
Stations (Ordered)
Geneve, Bel-Air
Genève, Stand
Genève, Palladium
Geneve, Jonction
Genève, Queue-d'Arve
Lancy-Pont-Rouge, gare/Etoile
Grand-Lancy, Stade de Geneve
Lancy-Bachet, gare
Plan-les-Ouates, Communes-Réunies
Plan-les-Ouates, Aviateurs
Plan-les-Ouates, mairie
Plan-les-Ouates, Vélodrome
Plan-les-Ouates, Arare
Plan-les-Ouates, Galaise
Plan-les-Ouates, Au Contour
Perly, En Louche
Perly, Ravieres
Perly, douane
Saint-Julien-en-Genevois, douane
Saint-Julien-en-Genevois, Hutins
Saint-Julien-en-Genevois, centre
Saint-Julien-en-Genevois, SNCF

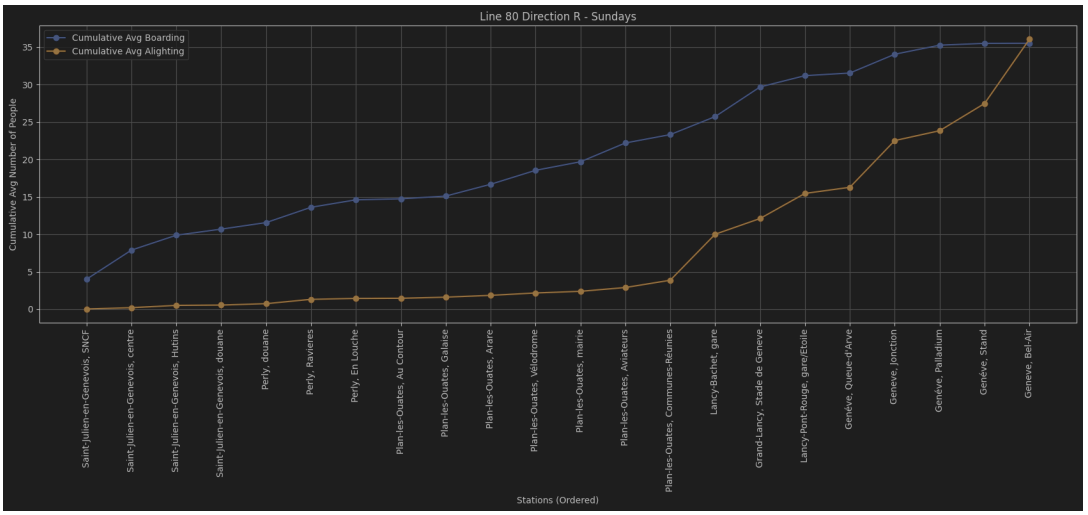

Line 80 Direction R - AM peak
Cumulative Avg Boarding
Cumulative Avg Alighting
Cumulative Avg Number of People
Stations (Ordered)
Saint-Julien-en-Genevois, SNCF
Saint-Julien-en-Genevois, centre
Saint-Julien-en-Genevois, Hutins
Saint-Julien-en-Genevois, douane
Perly, douane
Perly, Ravieres
Perly, En Louche
Plan-les-Ouates, Au Contour
Plan-les-Ouates, Galaise
Plan-les-Ouates, Arare
Plan-les-Ouates, Vélodrome
Plan-les-Ouates, mairie
Plan-les-Ouates, Aviateurs
Plan-les-Ouates, Communes-Réunies
Lancy-Bachet, gare
Grand-Lancy, Stade de Geneve
Lancy-Pont-Rouge, gare/Etoile
Genève, Queue-d'Arve
Geneve, Jonction
Genève, Palladium
Genève, Stand
Geneve, Bel-Air

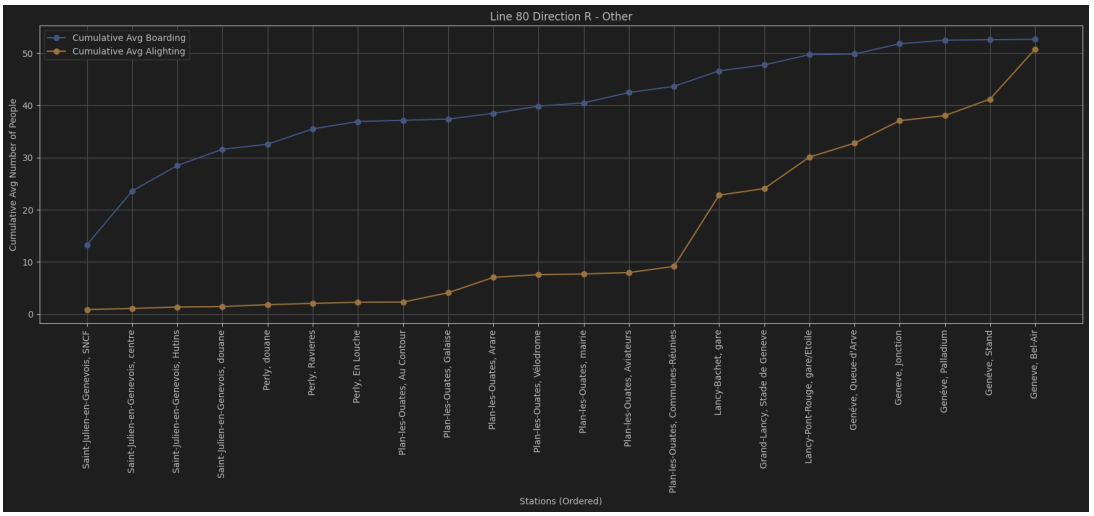

Line 80 Direction R - Day period
Cumulative Avg Boarding
Cumulative Avg Alighting
Cumulative Avg Number of People
Stations (Ordered)
Saint-Julien-en-Genevois, SNCF
Saint-Julien-en-Genevois, centre
Saint-Julien-en-Genevois, Hutins
Saint-Julien-en-Genevois, douane
Perly, douane
Perly, Ravieres
Perly, En Louche
Plan-les-Ouates, Au Contour
Plan-les-Ouates, Galaise
Plan-les-Ouates, Arare
Plan-les-Ouates, Vélodrome
Plan-les-Ouates, mairie
Plan-les-Ouates, Aviateurs
Plan-les-Ouates, Communes-Réunies
Lancy-Bachet, gare
Grand-Lancy, Stade de Geneve
Lancy-Pont-Rouge, gare/Etoile
Genève, Queue-d'Arve
Geneve, Jonction
Genève, Palladium
Genève, Stand
Geneve, Bel-Air


Line 80 Direction R - PM peak
Cumulative Avg Boarding
Cumulative Avg Alighting
Cumulative Avg Number of People
Stations (Ordered)
Saint-Julien-en-Genevois, SNCF
Saint-Julien-en-Genevois, centre
Saint-Julien-en-Genevois, Hutins
Saint-Julien-en-Genevois, douane
Perly, douane
Perly, Ravieres
Perly, En Louche
Plan-les-Ouates, Au Contour
Plan-les-Ouates, Galaise
Plan-les-Ouates, Arare
Plan-les-Ouates, Vélodrome
Plan-les-Ouates, mairie
Plan-les-Ouates, Aviateurs
Plan-les-Ouates, Communes-Réunies
Lancy-Bachet, gare
Grand-Lancy, Stade de Geneve
Lancy-Pont-Rouge, gare/Etoile
Genève, Queue-d'Arve
Geneve, Jonction
Genève, Palladium
Genève, Stand
Geneve, Bel-Air


Line 80 Direction R - Evening
Cumulative Avg Boarding
Cumulative Avg Alighting
Cumulative Avg Number of People
Stations (Ordered)
Saint-Julien-en-Genevois, SNCF
Saint-Julien-en-Genevois, centre
Saint-Julien-en-Genevois, Hutins
Saint-Julien-en-Genevois, douane
Perly, douane
Perly, Ravieres
Perly, En Louche
Plan-les-Ouates, Au Contour
Plan-les-Ouates, Galaise
Plan-les-Ouates, Arare
Plan-les-Ouates, Vélodrome
Plan-les-Ouates, mairie
Plan-les-Ouates, Aviateurs
Plan-les-Ouates, Communes-Réunies
Lancy-Bachet, gare
Grand-Lancy, Stade de Geneve
Lancy-Pont-Rouge, gare/Etoile
Genève, Queue-d'Arve
Geneve, Jonction
Genève, Palladium
Genève, Stand
Geneve, Bel-Air







Now to use a statistical analysis method to filter out the data. While initially tempting to simply filter the raw rows by removing all that for instance indicated a segment travel time less than 5 seconds, speeds more than 80 km per hour or significant distance variations with the theory, this would not have been very rigorous as a methodic process.

Note that it was not outliers that I wanted to eliminate as these are precisely the bottlenecks we need but rather the
Hence, I decided to eliminate a reasonable percentage of the extremes by trimming the edges of my distribution. Evidently, for dwell times for instance, this had to be symmetric and for each respective station. Once, I had verified that my selection of bounds would lead to a reasonable range, I had to e
0.01 to 0.99 for dwell times with physical bound that positive
0.01 to 0.99 for segment times with physical bound that positive
0.01 to 0.99 for segment speeds with physical bound that positive
They are all closely interlinked so we mustn't do this sequentially but rather find those that overlap and remove all those that satisfy one of all three in one go.

For line 18, minimums after this processing for dwell time are about 10 seconds (realistic if no stop at all) for less busy/significant stops or 30 seconds if more frequented stations which makes sense for a short stop.

See appendix 3 for this analysis. Having removed these, we find slight changes in the mean speeds:

Average segment speeds for Line 18 Direction A:
Segment 74221075: Average Speed = 23.95 km per hour
Segment 10751361: Average Speed = 19.80 km per hour
Segment 13610117: Average Speed = 19.14 km per hour
Segment 01171296: Average Speed = 20.13 km per hour
Segment 12960374: Average Speed = 28.89 km per hour
Segment 03740042: Average Speed = 20.16 km per hour
Segment 00425315: Average Speed = 20.81 km per hour
Segment 53150056: Average Speed = 20.46 km per hour
Segment 00560130: Average Speed = 17.53 km per hour
Segment 01300071: Average Speed = 25.67 km per hour
Segment 00711127: Average Speed = 27.29 km per hour
Segment 11271039: Average Speed = 25.00 km per hour
Segment 10391063: Average Speed = 19.38 km per hour
Segment 10630086: Average Speed = 15.84 km per hour
Segment 00863349: Average Speed = 19.52 km per hour
Segment 33490422: Average Speed = 9.49 km per hour
Segment 04220801: Average Speed = 15.26 km per hour
Segment 08011082: Average Speed = 22.14 km per hour
Segment 10821238: Average Speed = 32.03 km per hour
Segment 12381441: Average Speed = 38.66 km per hour
Segment 14410155: Average Speed = 34.95 km per hour
Segment 01550091: Average Speed = 34.44 km per hour
Segment 00910077: Average Speed = 33.45 km per hour
Segment 00770134: Average Speed = 42.78 km per hour
Segment 01340702: Average Speed = 36.94 km per hour
Segment 07023002: Average Speed = 31.60 km per hour
Segment 30020924: Average Speed = 37.78 km per hour

Segment 09240686: Average Speed = 25.20 km per hour
Segment 06860806: Average Speed = 42.57 km per hour
Segment 08060260: Average Speed = 37.52 km per hour

Average segment speeds for Line 18 Direction R:
Segment 02610807: Average Speed = 37.07 km per hour
Segment 08070687: Average Speed = 37.08 km per hour
Segment 06870925: Average Speed = 27.32 km per hour
Segment 09253003: Average Speed = 36.86 km per hour
Segment 30030703: Average Speed = 31.16 km per hour
Segment 07030137: Average Speed = 35.28 km per hour
Segment 01370078: Average Speed = 38.06 km per hour
Segment 00780092: Average Speed = 32.38 km per hour
Segment 00920156: Average Speed = 35.41 km per hour
Segment 01561440: Average Speed = 35.80 km per hour
Segment 14401237: Average Speed = 24.74 km per hour
Segment 12371081: Average Speed = 28.65 km per hour
Segment 10810800: Average Speed = 23.41 km per hour
Segment 08000427: Average Speed = 14.62 km per hour
Segment 04270356: Average Speed = 9.86 km per hour
Segment 03560087: Average Speed = 24.00 km per hour
Segment 00871064: Average Speed = 16.44 km per hour
Segment 10641041: Average Speed = 17.81 km per hour
Segment 10411128: Average Speed = 25.15 km per hour
Segment 11280072: Average Speed = 26.93 km per hour
Segment 00720131: Average Speed = 27.21 km per hour
Segment 01310057: Average Speed = 15.66 km per hour
Segment 00575316: Average Speed = 20.74 km per hour
Segment 53160043: Average Speed = 21.14 km per hour
Segment 00430375: Average Speed = 19.71 km per hour
Segment 03751297: Average Speed = 26.35 km per hour
Segment 12970118: Average Speed = 22.35 km per hour
Segment 01181362: Average Speed = 19.83 km per hour
Segment 13621077: Average Speed = 22.05 km per hour
Segment 10777423: Average Speed = 19.49 km per hour

Average segment speeds for Line 80 Direction A:
Segment 23211310: Average Speed = 17.30 km per hour
Segment 13101022: Average Speed = 27.73 km per hour
Segment 10220720: Average Speed = 29.48 km per hour
Segment 07207166: Average Speed = 41.46 km per hour
Segment 71660522: Average Speed = 40.22 km per hour
Segment 05221298: Average Speed = 39.68 km per hour
Segment 12980122: Average Speed = 46.99 km per hour
Segment 01220335: Average Speed = 22.59 km per hour
Segment 03353303: Average Speed = 24.88 km per hour
Segment 33031037: Average Speed = 34.03 km per hour
Segment 10371414: Average Speed = 30.40 km per hour

Segment 14140052: Average Speed = 38.25 km per hour
Segment 00520630: Average Speed = 31.84 km per hour
Segment 06302117: Average Speed = 37.32 km per hour
Segment 21170782: Average Speed = 29.21 km per hour
Segment 07823301: Average Speed = 27.05 km per hour
Segment 33010982: Average Speed = 32.37 km per hour
Segment 09826495: Average Speed = 32.38 km per hour
Segment 64956497: Average Speed = 42.31 km per hour
Segment 64972521: Average Speed = 31.23 km per hour
Segment 25217444: Average Speed = 17.97 km per hour

Average segment speeds for Line 80 Direction R:
Segment 74442522: Average Speed = 18.98 km per hour
Segment 25226498: Average Speed = 38.58 km per hour
Segment 64986496: Average Speed = 43.46 km per hour
Segment 64960983: Average Speed = 26.14 km per hour
Segment 09833302: Average Speed = 31.65 km per hour
Segment 33020780: Average Speed = 37.60 km per hour
Segment 07800353: Average Speed = 30.23 km per hour
Segment 03530631: Average Speed = 34.40 km per hour
Segment 06310053: Average Speed = 45.52 km per hour
Segment 00531415: Average Speed = 38.11 km per hour
Segment 14151038: Average Speed = 29.03 km per hour
Segment 10383304: Average Speed = 28.53 km per hour
Segment 33040334: Average Speed = 33.38 km per hour
Segment 03340121: Average Speed = 25.87 km per hour
Segment 01211301: Average Speed = 32.12 km per hour
Segment 13010523: Average Speed = 27.56 km per hour
Segment 05237165: Average Speed = 30.56 km per hour
Segment 71650721: Average Speed = 17.75 km per hour
Segment 07211023: Average Speed = 39.45 km per hour
Segment 10231308: Average Speed = 21.83 km per hour
Segment 13082321: Average Speed = 22.63 km per hour

With the trimming, if we redo the table for all segments, we evidently see a mild decrease but the overall trend is evidently still there (for the updated code, see appendix 4).

period	large_dwell_time	early_departure	late_departure	low_speed	any_criterion	num_trips	any_criterion_percent
AM peak	38034	18647	32200	22145	90833	207172	43.8
Day period	90787	49474	93199	51700	237280	567593	41.8
PM peak	46745	18845	43647	25866	106681	206340	51.7
Evening	50417	40353	63764	29980	157542	401079	39.3
Saturdays	29842	16391	41432	14359	87033	232795	37.4
Sundays	18537	17723	21768	9870	60070	189171	31.8
Other	14284	19253	13577	10294	51892	162988	31.8

And then we can do the same for independent segments again, like here for 13010523

segment13010523_bottleneck_summary_with_n

period	large_dwell_time	early_departure	late_departure	low_speed	any_criterion	num_trips	any_criterion_percent
AM peak	1715	294	592	296	2052	2454	83.6
Day period	3558	211	1692	85	4237	5520	76.8
PM peak	1793	264	416	49	2001	2530	79.1
Evening	2094	125	961	12	2540	4114	61.7
Saturdays	1411	38	769	2	1654	2200	75.2
Sundays	939	33	382	0	1069	1792	59.7
Other	766	28	206	4	847	1438	58.9

We can also use the EcartDepart to determine the average delay resulting from each segment (see code in Appendix 5).

Line 18 Direction A:

Segment_XXXXYYYY	average_delay_from_segment in seconds
74221075	8.226335463644040
10751361	6.640504969110930
13610117	7.831324004305710
1171296	0.7679136528028930
12960374	-3.6941531798495600
3740042	4.373444444444440
425315	0.6852805938729160
53150056	-3.9070025524359100
560130	4.688679245283020
1300071	-10.041574032903500
711127	0.35177052643282000
11271039	5.3250546165344400
10391063	-9.37675995632435
10630086	5.602446307568620
863349	**-35.624949605482900**
33490422	**39.39553694355210**
4220801	17.81249097820330
8011082	-12.264701639659600
10821238	-0.2152552093290000
12381441	-0.8108056418838150
14410155	-4.883306866849530
1550091	-2.5799827271854900

910077	5.945400650219930
770134	-3.3627629935632600
1340702	-1.2878402301394100
7023002	8.076444492534080
30020924	-6.839882854100110
9240686	-11.015969016924000
6860806	-9.09518483008416
8060260	-4.75479925551715

Line 18 Direction R:

Segment_XXXXYYYY	average_delay_from_segment in seconds
2610807	-1.4692532151734600
8070687	14.871828378525500
6870925	-8.752084669660040
9253003	-6.6849205074674800
30030703	3.9348722617963700
7030137	3.948658308277080
1370078	3.141085529974420
780092	7.883366688903710
920156	-4.605736158881620
1561440	-17.166972273156600
14401237	17.875484871993800
12371081	-17.201117936797800
10810800	-22.022859356838400
8000427	**33.4156656518129**
4270356	**38.98712072866510**
3560087	**-35.860397025238600**
871064	-2.4768208058778900
10641041	6.8339838061396900
10411128	0.47140451721535200
11280072	1.6784921163668700
720131	-13.844941818384300
1310057	15.331854929261200
575316	4.749046657694030
53160043	-11.97679218967920
430375	7.842111154642640
3751297	3.2358142137660900
12970118	-2.6235775547956700
1181362	4.377883622173760

13621077	10.396590664659100
10777423	-5.428756020204390

For line 18, in both directions the segments connecting Bel-Air, Genéve and Coutance, Genéve (863349 in direction A and 3560087 in direction R) and that connecting Coutance, Genéve and gare Cornavin, Genéve (33490422 in direction A and 4270356 in direction R) have significant delays with circa -36 seconds and +39 seconds in each direction respectively. The segment, 8000427, connecting Lyon, Genève and gare Cornavin, Genève (Line 18 direction R) also has a significant segment delay at +33.42 seconds.
The significant positive delay in the Cornavin segment is logical and aligns with all expectations.

Line 80 Direction A:

Segment_XXXXYYYY	average_delay_from_segment
23211310	23.67220147370550
13101022	**-31.139237877686600**
10220720	**31.939719410041600**
7207166	-3.36873840445269
71660522	-0.49173574996138200
5221298	-3.4360143442623000
12980122	7.382223134558140
1220335	1.0747126436781600
3353303	1.7332659251769500
33031037	2.327000604716790
10371414	10.643032766320400
14140052	-12.442611442611400
520630	-8.176355519887650
6302117	-12.205144625893900
21170782	-4.412017601537610
7823301	18.212239975881800
33010982	**70.49143259132710**
9826495	0.8266767523953610
64956497	3.2607971745711400
64972521	8.038574143510200
25217444	**37.626987060998200**

Line 80 Direction R:

Segment_XXXXYYYY	average_delay_from_segment
74442522	8.352531162554060
25226498	-17.88013144854210
64986496	-25.382327254045700
64960983	-8.297307094769550

9833302	-7.892246705107080
33020780	9.980516491945790
7800353	-17.252956402578300
3530631	11.383333333333300
6310053	-6.249858619094130
531415	0.6177341914393710
14151038	11.28345776829700
10383304	24.12545482498850
33040334	-9.55797768907993
3340121	0.1939126417902790
1211301	2.3519665531124200
13010523	3.1995459002012500
5237165	**-55.35668427047740**
71650721	**68.8041878304136**
7211023	-12.355368050522800
10231308	-0.9540729247478670
13082321	8.685169885366110

Investigating further, by isolating the high delay trips, I realized that often large delays skewed the average. As some segment delays were as large as 30 minutes, these are unrealistic as they would indicate a bottleneck with speeds of 1 km per hour or less. Hence I will immediately filter out segment delays of magnitude +300 or more and -300 or less (having analyzed the distributions and considering the frequencies).
I redid the averages - also giving them by time periods but then also decided on a better indicator which is a number of significantly + and - delays as a percentage. As the average delay time could be misleading if for instance the + and - delay cancel each other out, I will also have a count for the number of significant delays (i.e. +45 to +300 or -45 to -300) and then find a relative percentage per time period.

Average segment delays
Line 18 Direction A:

Segment_XXXXYYYY	AM peak	Day period	PM peak	Evening	Saturdays	Sundays	Other	net
74221075	9.607398568019090	8.26423982869379	7.9325	7.691625069328900	8.064182194616980	4.428301886792450	10.753731343283600	8.124231093957300
10751361	10.600889382899060	8.98503207412687	8.76555023923445	2.947280334728030	5.803786574870910	4.170111287758350	4.342741935483870	6.640504969110930
13610117	12.304031354983200	8.267012893982810	8.257142857142860	5.675515391132450	6.621413276231260	5.856542301629010	10.943903835146000	7.958225667527990
1171296	0.910061832490163	1.068633011911510	0.840596330 2752290	1.748322147 6510100	-2.13077274 80567000	-1.48235974 55176400	3.753846153846150	0.767913652 8028930
12960374	3.084411930219470	-7.123971685479240	-7.752298850574710	-2.115596330 2752300	2.452061139 4164000	-6.711764705882350	-3.908059701492540	-3.694153179 8495600

3740042	6.3441483198146	5.761343012704170	6.889332572732460	4.382663847780130	3.071591427268580	1.3337047353760400	-0.8607372732592160	4.275268100238930
425315	-0.4924942263279450	0.6718439588224670	0.5401891252955080	1.374668630 3387300	2.5134764732754700	-0.4038888888888890	-0.5	0.6852805938729160
53150056	-5.169737576772750	-3.8586263435962800	-4.612786489746680	-3.2415501165501200	-2.345031905195990	-4.877448237269170	-4.391980849790550	-3.9070025524359100
560130	7.520425293788470	4.355584319526630	5.046109510086460	4.415835777 1261	4.481965734896300	3.40672268907563	4.568276684555750	4.688679245283020
1300071	-10.701195219123500	-8.06753975678204	-5.555045871559630	-9.668640805448620	-11.153327338129500	-13.954469739033390	-11.777202072538890	-9.711203780928550
711127	-0.8733031674208150	0.5671045117075960	2.42445620223398	0.42484057090798700	-1.182738881549980	0.4462678665960830	0.09376723662437950	0.29380698243273300
11271039	5.108998732572880	5.033062535424150	5.695937873357230	4.599304237824160	3.722744360 9022600	6.816085489313840	7.750140370578330	5.325054616 5344400
10391063	-11.914319952067100	-11.203510082150900	-11.913287585776700	-9.755665942253960	-5.872857142857140	-5.581096196868010	-8.299819385912100	-9.570582825612140
10630086	5.982255294 79107	7.180835197613720	14.073394495412800	3.185219257896350	6.621864647420730	0.4983240223463690	1.143831370117790	5.602446307568620
863349	**-38.854060193072100**	**-30.800562324273700**	-26.40105193951350	**-38.099968953741100**	**-34.575502571295**	**-40.635540406406340**	**-47.669549104385400**	**-35.624949605482900**
33490422	**41.09398715703440**	**39.39940276222470**	**33.921495327102800**	**37.566926677 06710**	**35.517193308 55020**	**45.662902315 07620**	**44.901105651 10570**	**39.395536943 55210**
4220801	11.945614035087700	17.14	16.479154433352900	18.023139745916500	18.281814686659000	21.728950923007700	17.835946924004800	17.562415784408100
8011082	-14.521942656524300	-7.737222418047230	-8.333762886597940	-13.43326715199650	-13.696197718 63120	-12.014765100671100	-20.149756493506500	-12.264701639659600
10821238	-3.3406408094435100	2.037443399512370	10.39602053915280	-3.777533523851400	2.172087745839640	1.3046417305092400	1.257438016 5289300	0.7866673042896080
12381441	0.2395005675368900	-3.451353711790390	-0.36829117828500900	-2.225628584 0317600	1.196752265 8610300	3.835283528 3528400	-0.5743440233236150	-0.9426194233252040
14410155	-0.7000572409845450	-6.529949784791970	-3.2041297935103200	-6.706839674510670	-4.082238077356370	-1.970135746606340	-5.417840375586850	-4.883306866849530
1550091	-3.0146370023419200	0.6711695698542480	1.5494117647058800	-4.640964378801040	-5.055306427503740	-4.8515660463005000	-4.324010327022380	-2.608847943956620
910077	8.700463499420630	6.326329022988510	6.150803461063040	5.458518990027580	3.749160134 3785000	6.337825696316260	5.985750209555740	5.945400650219930
770134	-1.565597667638480	-0.368713083944872	9.402939375382730	-3.282215357458080	-1.273863636 3636400	-10.339173405211100	-15.840241448692200	-3.362762993 5632600
1340702	-7.654747948 41735	-3.831373989713450	-5.438995913601870	2.84369011771147	3.624890829694320	1.11395101 171459	-0.5933908045977010	-1.2878402301394100
7023002	9.682521489971350	9.829451677975430	6.943859649122810	2.643281165 67728	8.173710904480140	6.582938386255900	6.489681050 65666	7.272594436 62734
30020924	-4.230861965 03918	-4.707174490699730	0.3632580261593340	-10.470317159121700	-8.172342192 69103	-8.607883817427390	-11.629734313171300	-6.839882854100110
9240686	-10.280764635603300	-13.316183015844800	-5.028555111364930	-11.391965255157400	-12.901993355481700	-2.4790481117 43400	-16.27384960718290	-11.015969016924000
6860806	-15.673469387755100	-15.554435846679900	-10.168870192307700	-15.796391058443300	-10.395911556111800	-10.727509100364000	-17.165814877791030	-14.130592632141000

8060260	-4.868360277136260	-5.2788880813953500	-7.623453152622270	-6.339617631439190	-1.5817882159044100	-1.7809132888660900	-4.511560693641620	-4.75479925551715

Line 18 Direction R:

Segment_XXXXYYYY	AM peak	Day period	PM peak	Evening	Saturdays	Sundays	Other	net
2610807	-1.6887340301974400	-1.8077705827937100	-1.4778353483016700	-2.3989100817438700	-1.1896697713801900	-1.3659439450002640	-1.2286549707602300	-1.7053576255609000
8070687	11.591191417278400	18.13538973032880	17.299820466786400	14.627563576702200	13.700511945392500	13.582980972515900	8.78726035868893	14.871828378525500
6870925	-15.553215077605300	-10.938701487123700	-2.7329376854599400	-6.3060637741766 90	-6.0012615643397800	-8.1351069379238 40	-10.582070707070700	-8.752084669660040
9253003	-7.736312849162010	-8.811081322609470	-6.51198963058976	-6.168148336296670	-5.837826453243470	-4.360020931449500	-4.2488539620170300	-6.754671521122240
30030703	4.166192373363690	2.619942713927 6800	4.853566958698370	4.66452772073922	3.537200504413620	5.378898128898130	4.454604833442200	3.934872261796 3700
7030137	-1.8050991501416400	6.16790976896489	-0.29531812725090000	-2.3193529059317000	14.104982206405700	9.145454545454550	-0.9160246533127890	3.948658308277080
1370078	8.524073049252910	3.566437465155180	-2.09886363 6363640	3.342882991556090	3.607111111111110	3.014900662 2516600	0.5549516908212560	3.141085529974420
780092	5.770996640537510	7.856773239696910	9.964591661907480	8.3002414001207	8.909009812 66726	7.261605679 9563100	6.056295399515740	7.841774614999720
920156	-3.1153203342618400	-5.5981532777931670	-2.6256781193490100	-4.3056872037 91470	-7.0398582816651900	-4.620950323974080	-2.2567324 9551167	-4.6057361588 81620
1561440	-15.343266475644700	-21.193613138686100	-19.828209764918600	-13.520303827052300	-15.563405002193900	-16.988004362050200	-15.808524173028 00	-17.428079146287 20
14401237	18.447012401352900	15.084522304497 00	20.485919712402600	17.16603553743080	19.433021806853 60	17.422674738580100	24.147864882090500	17.875484871993800
12371081	-18.441011235955100	-17.301096530648 90	-18.025547445255500	-18.141569767441 90	-15.395814781834400	-18.005503577325 30	-16.479365079365100	-17.401716025463600
10810800	-24.290741743367600	-23.097800399927300	-21.786442711457 70	-21.071751582755500	-21.309711286089 20	-21.650595882990 30	-19.380536494073600	-22.022859356838400
8000427	**32.955555555555600**	**32.680180180180 20**	**38.366398016119000**	**34.797114517583 40**	**35.244871235268 40**	**33.340333871836 30**	29.792103639728600	**33.739520460712 10**
4270356	**34.990974729241 90**	**40.550686966208 70**	**50.950609364977 60**	**38.741729893778500**	**35.374461979913 90**	**34.335680751173 70**	**34.592977893368 00**	**38.835335073977 40**
3560087	**-36.601092896174 90**	**-37.985217065868 30**	**-53.340673575129500**	**-40.510826471485 20**	-28.679611650485400	-23.998236331569700	-22.6491344873502	**-35.860397025238600**
871064	0.583793738489871	-2.357932197040640	-1.7105943152454800	-0.9042714328991220	-6.995664739884390	-6.687717265 3534200	-2.972495088 4086400	-2.792254753295230
10641041	6.943952802359880	6.109070034443170	5.993777224642190	7.361563517915310	7.592843326 88588	8.208624708 62471	6.321380243572400	6.833983806 1396900
10411128	-3.4	-1.373851059 8386800	1.248301420 6300200	-0.22276657060518700	8.309859154929580	-1.571929824 5614000	-0.113785557 9868710	0.147851335 6562140
11280072	-2.529310344827590	1.961684438 8216200	6.107883817 4273900	0.9185949261221080	6.945236999539810	-3.043962159154150	0.373913043 4782610	1.678492116 3668700

720131	-11.207058823529400	-14.935536888239600	-20.54159292035400	-16.339370298133200	-15.954859511174570	-11.770011273957200	-10.378552971576200	-14.809948198072700
1310057	17.815421979356400	18.10408719346050	14.038254632396900	12.34108527131780	14.406922357343300	15.832762557077600	11.773063973064000	15.331854929261200
575316	3.9771497294046900	7.2605646466646290	5.125	2.506196640044070	5.099014547160960	4.738274464389110	0.6996563573883160	4.749046657694030
53160043	-8.660479041916170	-10.262235867798400	-15.656307129798900	-14.376531445684700	-12.378913443830600	-10.024798154555590	-13.773030707610100	-11.976792118967920
430375	10.982174688057000	9.654499910538560	16.872847682119200	7.310972297664310	5.830547818013000	6.872193436960280	5.528276779773790	8.827640273266880
3751297	9.301820317087490	9.976216022889840	2.9961953075459700	-11.026830631637800	-0.9260286638927420	16.729081334113500	-3.657622739018090	3.235814213760900
12970118	-4.2982768865121800	-5.394149908592320	-1.5256723716381400	-4.2640924464487000	3.2677824267782400	3.0459242250287000	-2.9658172777812310	-2.6235775547956700
1181362	3.6972909305064800	3.0952925481227400	4.55741775294848	3.771530038472920	4.455258302583030	3.0667040358744400	0.8871493803000650	3.392365097588980
13621077	10.8986562150056	10.905223478729100	12.575882352941200	10.517640417725100	10.611348122866900	9.724760892667380	6.38361581920904	10.396590664659100
10777423	-8.813953488372090	-10.861121157323700	-11.202247191011200	-8.772727272727270	6.506666666666670	6.648881239242690	0.5285132382892060	-5.428756020204390

Line 80 Direction A:

Segment_XXXYYYY	AM peak	Day period	PM peak	Evening	Saturdays	Sundays	Other	net
23211310	**30.663303909205600**	22.76034236804570	11.958465189873400	22.956163092151800	29.053562888570100	24.585553582001200	18.761866452131900	22.942166725645800
13101022	**-32.980867346938800**	**-34.783136899365400**	**-37.790636042402800**	**-33.662726008344900**	-18.611896635787400	-19.639354438237100	-25.812121212121200	**-31.139237877686600**
10220720	**34.027153160797600**	**33.126964059960300**	**41.507792207792200**	29.935379644588000	29.645446096654300	27.733656174334100	24.905882352941200	**32.209863210943100**
7207166	-9.035364294844480	-4.847802197802200	-3.360148514851490	-1.272471910112360	-1.182955071792500	0.7941527446300720	-3.799812030075190	-3.428615606638490
71660522	9.901050788091070	2.4296619411123200	-10.389583333333300	-5.319801533603970	4.819339622641510	-4.228158390949090	-0.4557981651376150	-0.491735749961382
5221298	-2.530901287553650	-6.758419000354480	-5.175894378194210	0.427027027027000	-0.490883590462833	-3.168888888888890	-6.448374760994260	-3.436014344262300
12980122	4.272317403065830	1.012682754976220	13.685085354896700	15.199154428126400	8.974707259953160	-1.011778563015310	11.959537572254300	7.382223134558140
1220335	3.475826972010180	-0.3134941912421810	1.5463917525773200	1.949704788978790	1.465300419189570	-4.677345537757440	6.5625	1.012817252428670
3353303	-0.4721157939548740	1.7460146874440300	-1.4099302421009400	2.846652267818570	4.277545327754530	3.768384481760280	0.2126245847176080	1.733265925176950
33031037	3.3828160143949600	2.485561497326200	4.2340511986997200	4.350021394950790	3.468505747126440	1.371264367816090	1.772967265047520	3.218608169440240
10371414	13.165763324299900	13.562998405103700	0.5939849624060150	9.133347289093570	12.500909090909100	10.722222222222200	10.061158798283300	10.365890383457100

14140052	-15.0508764 42924300	-10.2063435 49536700	-29.7813333 33333300	-11.1881065 4552960	-6.57721746 9608290	-5.88511601 5846070	-10.3863885 83973700	-12.4426114 42611400
520630	-8.01456953 642384	-6.04834517 4146600	-11.8500226 55188000	-8.43605257 6674320	-8.32815356 4899450	-10.8654836 86319400	-6.37991718 426501	-8.17635551 9887650
6302117	-15.9986046 51162800	-7.90678280 2075610	**-33.7716020 55114400**	-15.3351523 41973600	-4.69490695 3966700	-4.38705738 70573900	-2.91942148 7603310	-12.6199754 44402900
21170782	2.65205959 6844870	-9.50659301 4967930	2.67480777 92853900	-5.72765509 98948500	-3.71157407 4074070	-5.62984384 0370160	-0.84050880 62622310	-4.41201760 1537610
7823301	10.5425268 37324500	15.85560569 2467900	35.4364427 2595460	19.37048698 572630	20.0657657 65765800	17.0737142 85714300	12.08662280 7017500	18.49042665 460580
33010982	2.30742509 7698650	-2.98438661 7100370	-5.75516372 7959700	-5.42454429 8431540	1.57467532 46753200	-1.17139479 90543700	1.626794258 3732100	-2.35899318 3009960
9826495	-4.95870337 4777980	0.078103207 81032080	-0.34440039 6432111	3.276396249 49042	4.03165988 2406150	2.35787089 4677240	-1.42933049 94686500	0.826676752 3953610
64956497	-3.14537246 0496610	2.706552706 5527100	0.56832151 30023640	5.315916885 212830	8.18383017 163505	5.77733485 1936220	0.611404435 0580780	3.260797174 5711400
64972521	12.9741816 50530200	7.005043900 616480	-11.2335190 3435470	6.257429718 875500	17.8615730 33707900	16.6543352 60115600	17.88461538 4615400	8.038574143 510200
25217444	**61.2634615 3846150**	**36.54949067 845470**	24.7800995 02487600	27.04854563 691070	**48.1303129 37879500**	**41.6040070 713023**	**52.69117647 058820**	**37.97195225 015850**

Line 80 Direction R:

Segment_X XXXYYYY	AM peak	Day period	PM peak	Evening	Saturdays	Sundays	Other	net
74442522	-10.7291242 36252500	9.21173806 378562	17.1534941 7637270	16.43580104 10642	14.6711469 53405000	14.52957435 0469900	-12.5225088 5179570	7.88519914 4341450
25226498	-29.0156931 12467300	-17.8225308 6419750	-18.4387430 38981700	-14.1225146 19883000	-13.0853211 00917400	-12.8222735 14137300	-20.8152762 73022800	-17.8801314 4854210
64986496	-29.3328947 36842100	-26.0137956 748695	-29.1452492 211838	-23.8747880 15828200	-24.0109539 02327700	-22.3079080 2019070	-20.6507754 1642730	-25.3823272 54045700
64960983	-8.45994718 3098590	-7.13263954 5884580	-10.1849642 00477300	-9.37876254 180602	-6.22434701 4925370	-8.17439910 5645610	-9.35913853 3178120	-8.29730709 4769550
9833302	-1.81873111 78247700	-8.38128116 6091290	-2.92865683 5392590	-6.33052459 9075840	-2.95746691 87145600	-6.10742857 1428570	-7.82444061 9621340	-5.64652317 8807950
33020780	-0.13500597 371565100	11.7467382 96239400	8.70581440 123219	10.20114472 60834	11.7985074 62686600	12.23492605 2332200	15.0210843 37349400	9.85432970 1805530
7800353	11.98905908 09628	-25.6334387 20476100	-28.8411428 57142900	-17.1305767 1381940	-18.7369838 42010800	-14.9452130 60321000	-12.9425219 94134900	-17.2529564 02578300
3530631	56.30482758 62069	8.27844588 3441260	1.81960934 5078510	0.872414719 3123830	-1.05176362 80348100	-0.90529247 91086350	30.1244212 96296300	11.3833333 33333300
6310053	-25.4470438 11144200	-6.41400075 2728640	-6.74580433 8927550	0.361420243 77318500	-1.95289179 1044780	-0.22164351 851851900	-4.69652327 6370070	-6.24985861 9094130
531415	-20.8645753 63427700	1.89096749 81103600	3.67732677 3267730	5.708497763 746380	6.91805555 5555560	7.362682215 74344	-1.03403982 01669900	0.61773419 14393710
14151038	6.685855890 202060	16.0983050 84745800	13.5265486 72566400	11.98440343 1245100	13.8678107 60667900	12.50318471 3375800	6.17019867 5496690	12.3891856 14731200

10383304	15.859937402190900	34.1966519145661	24.65326633165830	20.357273918741800	21.216241299303900	20.6905311778291	18.39850136239780	24.00128139415680
33040334	-16.665951972555700	-10.352431209951000	-14.54576144036010	-4.527229676400950	-8.491284403669730	-7.021387283237000	-4.153241650294700	-9.557977689907993
3340121	-0.5683610867659950	-1.208946877912400	-13.75583696082310	5.49522087315939	3.1929499072356200	3.862860506180110	8.654648956356740	0.3007032493198500
1211301	3.2925639983746400	3.279056603773590	-6.499019992159940	-1.780614903595620 0	15.80057251908400	8.275353773584910	-4.664089950808150	2.1990704879938000
13010523	-7.868354430379750	4.481970260223050	-4.964183964183960	6.298466180538090	11.676597789524300	6.612280701754390	5.755698005698010	3.1995459002012500
5237165	**-60.18666089216110**	**-61.641263940520400**	**-70.89889415481830**	**-58.1200208279094**	**-45.57540056550420**	**-33.5787037037037**	**-34.13948497854080**	**-55.712998446400 80**
71650721	**74.8081615481 7000**	**72.488621403046 80**	**82.994075327972 90**	**66.3954787909576 0**	**59.546517064048 60**	**57.2686397268 0710**	**56.098290598290600**	**68.804187830413 6**
7211023	-16.426961602671100	-15.671524663677 10	-12.697585768742100	-15.704060913705600	-16.378441437237500	-14.808620689655200	-16.428466076696200	-15.461893405226000
10231308	-3.527375201288250	-0.650908416759362 0	-1.282546201232030 0	-1.604251012145750	-1.924003707136240 0	2.7007553747821000	2.345088161209070	-0.954072924747867 0
13082321	9.775344687753450	9.718214352985500	12.584415584415600	10.499248496994000	6.553044280442810	6.633657698912420	7.733718054410550	9.494188749418880

Appendix:

This report particularly investigates tram line 18 going between Palettes, Grand-Lancy and CERN Meyrin and the bus line 80 going between Bel Air, Geneva and SNCF, Saint-Julien-en-Genevois.

The TPG has been made aware of concerns regarding these lines through numerous channels, both official and informal, yet the issues persist.

Various users on platforms such as Reddit have repeatedly highlighted long-standing issues with these two lines, particularly regarding congestion and inefficiency:

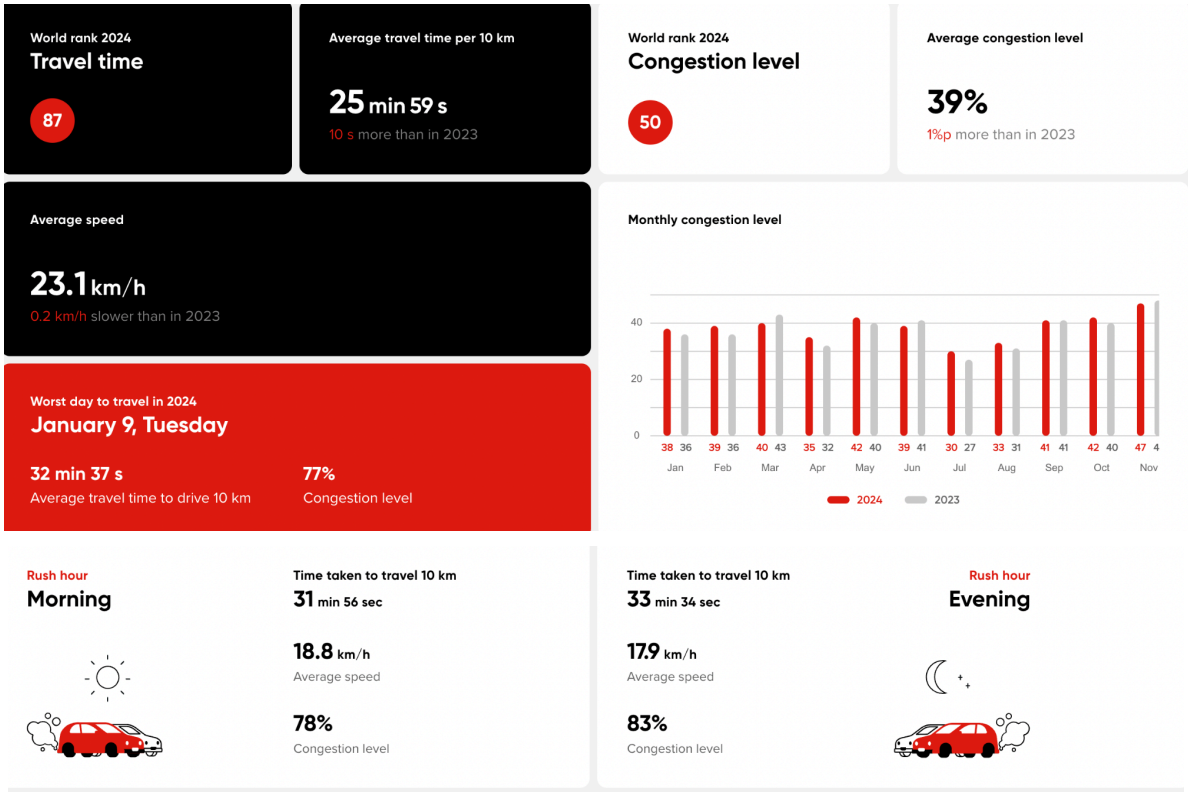
“Bus 80, overpacked, two places where it doesn’t move (au contour between Perly and PLO) and at P+R Étoile because of the cars changing lanes” – October 2024

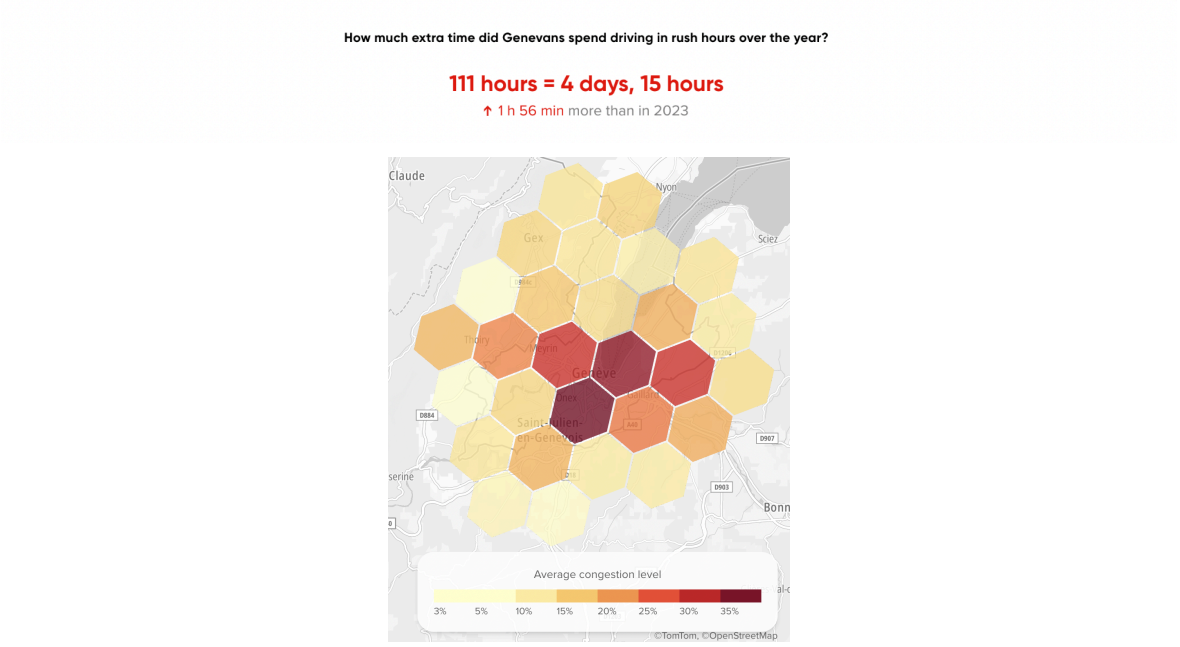
“Tram 18 shares a lane with cars between Meyrin Village and Hôpital de La Tour…” and “they share the road and the tram gets stuck VERY badly.” – November 2024

Several users have echoed these frustrations, noting that Tram 18 frequently becomes gridlocked in mixed traffic, especially around Meyrin Village, where shared lanes severely hinder service reliability.

In an age where data is abundant and accessible through diverse platforms, it is essential that transport authorities embrace modern methods of gathering and analyzing public feedback, including online forums and social media. Doing this allows for a more responsive, data-driven approach to public service improvement.

TomTom Geneva Report Data for 2024





Congestion levels are logically highest within the CBD, with congestion levels in the Meyrin–Vernier sector, Cologny sector and Carouge–Onex sector at 28%, 33% and 36%

respectively. TomTom calculates the congestion level percentage by measuring the extra time it takes to travel a certain distance compared to the time it would take under free-flowing traffic conditions (TomTom).

Congestion is a major issue commonly found in surface public transport systems, impacting travel times, service reliability, and overall operational costs (Garrido-Valenzuela et al.).

TomTom data indicated that peak morning and evening rush hours are particularly between 8-9 AM and 5-6 PM (4-6 PM on Fridays) respectively - aligning with the general global standard of 7-9 AM and 4-6 PM.

Note: During the time of the report, due to summer maintenance works, the tram line 18 only went between Bel Air, Geneva and CERN Meyrin and the bus line 80 went between Bel Air, Geneva and SNCF, Saint-Julien-en-Genevois, however the standard routes were evidently used in the analysis.
More information available on TPG website or in local newspapers.

Initial questions: What is the problem? How can one fix it?
Has any city had the same problem? What did they do? What was successful? What are some new ideas?

Notes on academic research:

Integrating public transit signal priority into max-pressure signal control: Methodology and simulation study on a downtown network

There's a need to balance private vehicle efficiency with public transit priority (TSP) to improve overall urban mobility and equity.

Proposed Solution:
The authors developed a modified max-pressure signal control method that integrates Transit Signal Priority (TSP) specifically for Bus Rapid Transit (BRT) systems with dedicated lanes.
The approach is decentralized, relying only on local traffic conditions at each intersection.
It introduces constraints to ensure that buses receive priority at signals without causing instability in the overall traffic network.

Methodology:
The study designed dynamic queuing models for both buses and private vehicles.
A conflict region model was incorporated to manage interactions between BRT and other traffic.
Stability analysis mathematically proved that the modified system maintains maximum stability (i.e., no indefinite vehicle queue growth).

Simulation & Results:
Simulations were conducted on a real-world downtown network in Austin, Texas using BRT routes and actual timetables.

The modified system:
Reduced bus travel time.
Reduced private vehicle travel time.

Maintained or improved network stability compared to existing control systems.
The approach proved more effective than both traditional fixed-time and adaptive traffic signal control with conventional TSP.

Policy Implications:
The findings support the adoption of smart, adaptive traffic control systems that prioritize public transit without sacrificing the efficiency of private vehicle flows.
The proposed method can promote a shift from private car use to public transit by improving bus reliability and speed.

Analysis of the causal effects of exclusive bus lanes on performance indicators of public transport corridors

The study developed a new, more accurate way to **measure how exclusive bus lanes affect bus speed and reliability.**

Using advanced statistical methods, the researchers found that exclusive bus lanes made buses go about 20% faster and improved reliability by 30% within 10 months.
Over time, as more people started using the buses, these benefits were reduced—bus speed improvements dropped by 27%.
The study points out that earlier research often missed these indirect effects and didn't fully capture the real impact of bus lanes.
The findings help transit agencies understand both the short-term benefits and long-term challenges of exclusive bus lanes.
The authors recommend more detailed studies in the future, especially to look at how bus lanes affect regular traffic and how changes in land use might play a role.

Automatic bottleneck detection using AVL data: a case study in Amsterdam

The paper introduces a method for automatic detection of bottlenecks in public transport operations by analyzing AVL data.

- Key performance indicators such as dwell time, run time, and punctuality are calculated from AVL data. The method uses eight specific criteria to systematically identify bottlenecks in the network.
- The approach was applied to the Amsterdam tram network before the opening of a new metro line, using a dataset of approximately 1.9 million AVL records.
- Variable sensitivity values -> therefore, it is important to set these values such that a reasonable number of bottlenecks are identified (almost says to play around in order to identify an amount that we can work with)
- Most bottlenecks were found during the **PM peak period**, with issues mainly related to low speeds and unreliable departure times. These problems were especially **concentrated in the city center (pedestrian, leisure activity)**

The low speeds are a result of a lot of people that are in the center, so the tram driver must consider these pedestrians into their driving behavior. This results in delays and low speeds. Also, the occupation at the tram stops is high as well, which causes larger dwell times.

- The study anticipates that the opening of the new metro line will alleviate tram bottlenecks and improve operations by reducing congestion in the city center.

Future research will focus on comparing network performance before and after the metro opening, integrating smart card data to assess passenger impacts, and developing real-time bottleneck detection systems connected to operational control tools.
The extension study in 2020, after the implementation of the NZ (Noord-Zuidlijn) was conducted by a student of Prof. N. Van Oorts, Bobby Kartoidjojo, using the same method and metrics as used in Brands et al.'s paper.

Coding:
Originally, having received the .csv (Comma-separated values) file, I tried to open it via Excel or Google Sheets. Neither were possible due to the file having 2,301,409 data rows. Tables in Apple Numbers can't support more than a million rows however I was able to open it there and take a first look at the first million rows (Apple Numbers saying there were 2,301,413 rows).
There were 73 columns and I had to figure out what each column represented.

Rows 1 to 855164 were all for line 18

Column Name	Description
IdCourse (RECAR_ID)	Unique identifier for the vehicle course (the whole paths going from RangArretAsc 1 to end)
IdArret	Identifier for the stop (arrêt)
RangArretAsc	Sequence number for this stop within the trip (ascending) 1 to 31 for line 18 1 to 23 for line 80 (1 additional due to border crossing)
DateCourse	Date when the course/trip took place YYYY-MM-DD Format
HDepartTheo (RECAR_HRE_DEP_THEO)	Scheduled (theoretical) departure time at this stop HH:MM:SS format
HArriveeTheo (RECAR_HRE_ARR_THEO)	Scheduled (theoretical) arrival time at this stop - meant to be **Identical to HDeparTheo which must be an error!** HH:MM:SS Format
DistanceTheo (RECAR_DIST_THEO)	Theoretical (scheduled) distance covered from trip start to this stop (in meters)
TempsInterArretTheo (RECAR_TPS_THEO)	Scheduled/planned travel time between two consecutive stops (in seconds) Document says Case-Formula **Case** / **Formula** **Origin Terminus (first stop)** / Not applicable (set to NaN or 0) **Intermediate stops** / HDepartThe(i) – HDepartThe(i-1)(difference in seconds) **Destination Terminus (last stop, n)** / HArriveeThe(n) – HDepartThe(n-1)(difference in seconds) **Regulation stop (if flagged as such)** / HArriveeThe(i) – HDepartThe(i-1)(difference in seconds) However this does not align with the values in the table So this is unreliable data
EcartDepart (RECAR_ECART_DEP)	Difference (in seconds) between actual and scheduled departure time
EcartDistance (RECAR_ECART_DIST)	Deviation (in meters) from scheduled distance i.e. discrepancy between theoretical and realised distances
TempsPasseArret (RECAR_TPS_PASSE)	Time (in seconds) spent at the stop (door open, boarding, etc.)
TempsInterArretRealise (RECAR_TPS_REAL)	Actual time (in seconds) spent traveling between two stops Document says

	<table><tr><th>Case</th><th>Formula</th></tr><tr><td>First stop (origin terminus)</td><td>Not applicable(NaNor0)</td></tr><tr><td>Intermediate stops (2 to n-1)</td><td>HDepartRealis(i) - HDepartRealis(i-1)(in seconds)</td></tr><tr><td>Regulation stop</td><td>HArriveeRealis(i) - HDepartRealis(i-1)(in seconds)</td></tr><tr><td>Last stop (destination terminus)</td><td>HArriveeRealis(n) - HDepartRealis(n-1)(in seconds)</td></tr></table> However this does not align with the values in the table So this is unreliable data
VitesseInterArret (RECAR_VIT)	Average speed between stops (I suppose km/h - going from the magnitudes) Document says “Vitesse réelle inter arrêts : = Distance IA Réelle / Temps IA réel”
NbMontees (RECAR_NB_MONT)	Number of passengers who boarded at this stop
NbDescentes (RECAR_NB_DESC)	Number of passengers who got off at this stop
Charge (RECAR_CHARGE)	Occupancy of the vehicle after this stop (in passengers)
FlagArretReference (RECAR_TYPE_PT)	Reference stop flag (e.g., a major point/terminal,'Y' for yes, else 'N')
FlagDeviation (RECAR_FLAG_DEVIA)	Indicates if the trip deviated from its planned route ('Y' or 'N')
FlagDelocalisation (RECAR_FLAG_DELOC)	Indicates GPS or localization issue for this data point ('Y' or 'N')
FlagRecalage (RECAR_FLAG_RECAL)	Flag for recalibrated data or stop timing ('Y' or 'N')
FlagRegulation (RECAR_FLAG_REGUL)	Trip was regulated/adjusted by dispatching ('Y' or 'N')
IdArretPrecedent (RECAR_ID_PREC)	ID of the previous stop (if available)
DistanceInterArret (RECAR_LG_INT_ARRET)	Distance (in meters) between this stop and the previous one (not theoretical as varies but very similar to DistanceThe(i) - DistanceThe(i-1))
NbMonteesPorteX (RECAR_NB_MONT_X)	Number of passengers who boarded at this stop via door X (0-5)
NbDescentesPorteX (RECAR_NB_DESC_X)	Number of passengers who alighted at this stop via door X (0-5)
HMarquageArretTheo (RECAR_HRE_MARQ_THEO)	Scheduled time when stop is marked/confirmed (“Valeur de l’heure de marquage de l’arrêt théorique”)
HMarquageArretAppl (RECAR_HRE_MARQ_APPL)	Applied/adjusted marking time for stop (“Valeur de l’heure de marquage de l’arrêt applicable”)
HMarquageArretReal (RECAR_HRE_MARQ_REAL)	Actual time of stop marking (“Valeur de l’heure de marquage de l’arrêt réalisée”)
HEntreeFenetreArretReal (RECAR_HRE_ENTREE_FEN_REAL)	Actual window entrance time at stop Window begins 35 meters before the stop location
HSortieFenetreArretReal (RECAR_HRE_SORTIE_FEN_REAL)	Actual window exit time at stop Window begins 35 meters before the stop location

HOuverturePortesReal (RECAR_HRE_OUV_PORTE_REAL)	Actual time doors opened
HFermetureportesReal (RECAR_HRE_FERM_PORTE_REAL)	Actual time doors closed
DTArriveeTheo	Datetime (scheduled) of arrival at stop Same as DTDepartTheo so clearly false
DTDepartTheo	Datetime (scheduled) of departure at stop
DTEntreeFenetreArretReal	Datetime of entry into stop's time window (actual)
DTSortieFenetreArretReal	Datetime of exit from stop's time window (actual)
DTOuverturePortes	Datetime when doors opened (actual)
DTFermeturesPortes	Datetime when doors closed (actual)
DTEntreeArretAtp	Datetime of ATP system registering arrival at stop
DTSortieArretAtp	Datetime of ATP system registering departure from stop
DTMarquageArretTheo	Datetime, theoretical marking of stop
DTMarquageArretReal	Datetime, real marking of stop
Latitude / Longitude	Geographic coordinates (stop location)
C_DExploitCourse	Seems useless - either blank or TPG Must be provider/operator of the line
C_DistanceApp	The DistanceTheo of the last course of a trip (largest RangArretAsc of a specific trip of id x) Same for all courses of a specific IdCourse
C_DistanceBattemen	Basically always 0
C_DistanceDeviatio	Basically always 0 Except if there is a deviation (we remove rows with this flag)
C_DistanceLign	???
C_DistanceThe	Seems to be same as C_DistanceApp
C_Ligne	Line number (e.g., 18 for tram, 80 for bus)
C_NoParcoursAppl	Applied trip/run number
C_NumVoiture	Vehicle/unit/car number
C_ParcoursTypeAppl	????
C_SensAppl	Direction of trip (A or R - Aller/Retour) A for Palettes to CERN for line 18 A for Bel-Air to St-Julien for line 80
C_ServiceVoiture	Vehicle service code (may relate to depot or shift????)
C_TypeAppl (REFCO_TYPE_REC)	Tells us what type -> 0 is for commercial trips HLP out of depot: 1 HLP leaving depot: 2 HLP returning to depot: 3
CodeLong	Involves Arret (differentiates between the actual stops at a particular station) Short identifier
Arret	Stop name (e.g., CERN, PALETTES, etc.))

I had an internal TPG file 40014 CPT 06688 B0 Modèle Physique de Données du SAE-TD Recueil Section TABLE REC_ARRET which was extremely helpful too.

Each row in the Geneva TPG dataset corresponds to a specific event at a single stop for a particular vehicle trip on a specific line and date. This is best understood as:
One row = the state and events of one vehicle at one stop during one trip run.

Initially encountered issues opening the file on Jupyter Notebook and via PyCharm's IDE. To try to find the delimiter, I used the most common and standard way, using the Sniffer class of the csv module.

```
import csv

with open('/Users/benfall/Desktop/arrets_ben.csv', 'r') as f:
   sample = f.read(1024)  # Read a small sample of the file - 2^10
   sniffer = csv.Sniffer()
   dialect = sniffer.sniff(sample)
   print(f"The detected delimiter is: '{dialect.delimiter}'")
```

The output of the cell was
The detected delimiter is: ' '
This was the first data bottleneck I encountered, before any traffic bottlenecks. Fortunately Dr. C. Cortes Balcells provided invaluable help and insights, solving the congestion issue! The csv file extension had thrown me off. The file was not in fact delimited but was in fact misnamed a csv file when it was in fact a fixed width file[4] (usually .txt, .dat or .fwf) - a mismatch that is missed by the Sniffer.

Initially loaded the data into a pandas DataFrame for analysis…

```
import re
import pandas as pd

path = r'/Users/benfall/Desktop/arrets_ben.csv'

# 1) Grab the header line + dash line
with open(path, encoding='utf-8') as f:
   header_line = f.readline().rstrip('\n')
   dash_line   = f.readline().rstrip('\n')

# 2) Infer colspecs from runs of dashes
colspecs = [(m.start(), m.end()) for m in re.finditer(r'-+', dash_line)]

# 3) Slice out the raw names from the header
raw_names = [header_line[start:end].strip() for start, end in colspecs]

# 4) De-duplicate any repeated names by appending a counter
seen = {}
names = []
for nm in raw_names:
   if nm in seen:
       seen[nm] += 1
       names.append(f"{nm}_{seen[nm]}")
   else:
       seen[nm] = 0
       names.append(nm)

# 5) Read the data as fixed-width, skipping the two top lines
```

[4] No delimiter, fixed width format, column width inferred from the dashed line

```
df = pd.read_fwf(
   path,
   colspecs=colspecs,
   names=names,
   skiprows=[0, 1],      # drop both the header and the dash-line
   na_values=['NULL'],   # turn "NULL" → NaN
   dtype=str             # all data is read as string
)

print(df.shape)           # expect (n_rows, 73)
print(df.columns.tolist())
df.head()
```

And then:

```
# 1) Strip any leading/trailing whitespace from column names
df.columns = df.columns.str.strip()

# 2) Print them out to see the exact spellings
print(df.columns.tolist())
```

This lets us see all our DataFrame column names exactly as they exist in memory and cleans up column names by removing any leading or trailing spaces.
When importing from fixed-width files, CSVs, or spreadsheets, column names often contain hidden spaces due to formatting.
Spaces in column names are invisible in outputs, but they make code like df['MyColumn'] fail if the true name is 'MyColumn ' (with a space).
By stripping spaces, one standardizes all column names for clean and predictable access.

The output of the cell was:
['\ufeffIdCourse', 'IdArret', 'RangArretAs', 'DateCours', 'HDepartThe', 'HArriveeThe', 'DistanceThe', 'TempsInterArretThe', 'EcartDepar', 'EcartDistanc', 'TempsPasseArre', 'TempsInterArretRealis', 'VitesseInterArre', 'NbMontees', 'NbDescente', 'Charge', 'FlagArretReferenc', 'FlagDeviatio', 'FlagDelocalisatio', 'FlagRecalag', 'FlagRegulatio', 'IdArretPreceden', 'DistanceInterArre', 'NbMonteesPorte', 'NbDescentesPorte', 'NbMonteesPorte_1', 'NbDescentesPorte_1', 'NbMonteesPorte_2', 'NbDescentesPorte_2', 'NbMonteesPorte_3', 'NbDescentesPorte_3', 'NbMonteesPorte_4', 'NbDescentesPorte_4', 'NbMonteesPorte_5', 'NbDescentesPorte_5', 'HMarquageArretThe', 'HMarquageArretApp', 'HMarquageArretRea', 'HEntreeFenetreArretRea', 'HSortieFenetreArretRea', 'HOuverturePortesRea', 'HFermetureportesRea', 'DTArriveeTheo', 'DTDepartTheo', 'DTEntreeFenetreArretRea', 'DTSortieFenetreArretRea', 'DTOuverturePortes', 'DTFermeturesPortes', 'DTEntreeArretAtp', 'DTSortieArretAtp', 'DTMarquageArretTheo', 'DTMarquageArretReal', 'Latitude', 'Longitude', 'C_DExploitCours', 'C_DistanceApp', 'C_DistanceBattemen', 'C_DistanceDeviatio', 'C_DistanceLign', 'C_DistanceThe', 'C_EcartDepar', 'C_ExploitantLign', 'C_IdVehicul', 'C_Lign', 'C_NoParcoursApp', 'C_NumVoitur', 'C_ParcoursTypeApp', 'C_SensApp', 'C_ServiceVoitur', 'C_TempsBattemen', 'C_TypeAppl', 'CodeLong', 'Arret']

* The appearance of `ufeefIdCourse` (more precisely, `\ufeffIdCourse`) instead of `IdCourse` is caused by the presence of a Byte Order Mark (BOM) - specifically, the Unicode character `\ufeff`—at the start of your data file. This is an invisible marker that's sometimes included in UTF-8 or UTF-16 encoded text files to signal the encoding type to programs. This can evidently be removed - either using `str.replace('\ufeff', '')` method or during file reading (`df = pd.read_fwf('yourfile.csv', encoding='utf-8-sig')`).
'NbMonteesPorte' and 'NbDescentesPorte' clearly symbolized the number of people who entered through another door so I decided to call this the 0th door and for consistency, rename 'NbMonteesPorte_0' and ''NbMonteesPorte_0''. However I anyways removed these columns for the analysis as it wasn't very relevant to know how many people got
-> I recognize that later analysis may be interesting to see if dwell time can be reduced by speeding boarding and deboarding times by getting a more even spread in the use of doors
This in itself is however quite challenging as given the infrastructure, it would be difficult to implement such a scheme. Santiago for instance could significantly decrease dwell time by having automatic ticket machines installed at stations meaning the drivers didn't have to also check everyone in during boarding time, something much easier to implement (that Geneva already has done).

```
# For example, if your course-ID column is actually named "IdCourse"
course_col = '\ufeffIdCourse'      # replace with the exact string you saw
rank_col   = 'RangArretAs'  # adjust if necessary

# 3) Ensure C_TypeAppl is numeric and filter
df['C_TypeAppl'] = pd.to_numeric(df['C_TypeAppl'], errors='coerce')
```

```
df0 = df[df['C_TypeAppl'] == 0]   *

# 4) Sort by the actual course-ID and rank columns
df0 = df0.sort_values([course_col, rank_col])
```

This block finds and sorts all rows where C_TypeAppl is zero (commercial rides, removing depot trips), using exactly the (possibly messy) column names as they appear in your DataFrame - so the code won't fail because of invisible or odd characters in a column name.

Then df0, implicitly calling display to see the new dataframe, which I then also saved - df0 now containing only the rows from df where C_TypeAppl was 0, with it now ordered.

```
# Save the DataFrame to a CSV file
df0.to_csv('/Users/benfall/Desktop/saved_file.csv', index=False)
```

I then opened the saved file in Apple Numbers, the file now having 2,226,102 data rows (2,226,104 rows Apple Numbers) due to the removal of the 75,311 non-commercial steps. Although it was good to originally read everything as strings, I realised there was a flaw in my logic of ordering as like this the sorting (4) is done on something of type string, meaning lexicographical (dictionary) order is used instead of numerical.

We must therefore explicitly convert RangArretAs to a numeric type before sorting…

```
# 1) Define the column names we want to use for grouping/sorting
course_col = '\ufeffIdCourse'  # Use the exact column name from your DataFrame
rank_col = 'RangArretAs'       # Adjust if necessary

# 2) Ensure needed columns are numeric (vital for correct sorting)
df[course_col] = pd.to_numeric(df[course_col], errors='coerce')
df[rank_col] = pd.to_numeric(df[rank_col], errors='coerce')
df['C_TypeAppl'] = pd.to_numeric(df['C_TypeAppl'], errors='coerce')

# 3) Filter for commercial trips
df0 = df[df['C_TypeAppl'] == 0]

# 4) Sort by course and stop-sequence in numeric order
df0 = df0.sort_values([course_col, rank_col])
```

I then noticed that line 80 has 23 stops in the system as an additional “stop” was added between Perly, douane and Saint-Julien-en-Genevois, douane called Passage Douane (2541 on A and 2542 on R).
Saint-Julien Douane (FR) and Perly Douane (CH) are only c. 240 meters apart.

~~Despite the short distance, I decided it best to keep this stop as it really accurately gives an idea of the time spent waiting to cross the border.~~

Given the fact that HDepartThe and HArriveeThe seem to contain the exact same things, it is reasonable to deduce that they both in fact represent the theoretical departure time. Hence along with all the individual doors data, we can also delete HArriveeThe, DTArriveeTheo,

According to our theoretical data, the full line 18 is 13001 meters long in the A direction (Palettes to CERN) and 13057 meters long in the R direction (CERN to Palettes). The full line 80 is 10530 m long in the A direction (Bel Air to Saint Julien) and 9717 m long in the R direction (St Julien to Bel Air).

As I quickly noticed, for line 18, on the 1st of October 2024, I had 398 rows with RangArretAsc=1, 377 with 2, 243 with 3 243 with 4, 237 with 5, 237 with 6, 228 with 7, ….., and 192 with RangArretAsc=31. This is however totally normal as sometimes a vehicle starts in the middle of the line, or for instance, as can be observed with trams of line 18, between midnight and 2 AM, as vehicles retire for the night, lots of vehicles only do about 9 stops or about 25 depending on the direction to stop at Blandonettes, which is right next to the Dépôt TPG En Chardon.

Using the same way in which we isolated for only rows where C_TypeAppl==0, one can isolate to only keep all the rows where all 4 of the 5 flags other than FlagArretReference (which is just an attribute of the station in question i.e. if it is considered a reference terminal/stop like Cornavin for instance). For the moment, we will simply ignore the 20,454 that have at least one of the four flags.

The code below acted on the original file so we will then again need to sort `with_all_flags_n.csv` and then remove rows where non-commercial rides are present.

```
flag_cols = ["FlagDeviatio", "FlagDelocalisatio", "FlagRecalag", "FlagRegulatio"]

# Get subsets
mask_any_y = df[flag_cols].eq('Y').any(axis=1)
mask_all_n = df[flag_cols].eq('N').all(axis=1)

df_any_y = df[mask_any_y]
df_all_n = df[mask_all_n]

# Save
df_any_y.to_csv('with_flag_y.csv', index=False, encoding='utf-8')
df_all_n.to_csv('with_all_flags_n.csv', index=False, encoding='utf-8')
```

After this, the second file had 2,280,955 rows[5] (Apple Numbers Count 2,280,957) which by elimination, tells us that the former has 20,454 data rows given that ben_arrets.csv has 2,301,409 data rows.

Now we do what we had done before:

```
import pandas as pd

# 1. File paths
input_path = '/Users/benfall/Desktop/Data Files/with_all_flags_n.csv'
output_path = '/Users/benfall/Desktop/Data Files/all_flags_n_type0_sorted.csv'

# 2. Load data
df_all_n = pd.read_csv(input_path, encoding='utf-8')

# 3. Use correct column names (without BOM!)
course_col = 'IdCourse'
rank_col = 'RangArretAs'
type_col = 'C_TypeAppl'

# 4. Convert to numeric
df_all_n[course_col] = pd.to_numeric(df_all_n[course_col], errors='coerce')
df_all_n[rank_col]   = pd.to_numeric(df_all_n[rank_col], errors='coerce')
```

[5] Python ((len(df_all_n))) only counts data rows (no header, no extra line) while Apple Numbers probably includes header row (+1), and maybe an empty row (+1). When it was above 2 million, the error seemed to increase to 4… Needless to say, the important number is clearly the python value.

```
df_all_n[type_col]   = pd.to_numeric(df_all_n[type_col], errors='coerce')

# 5. Filter only type 0
df0 = df_all_n[df_all_n[type_col] == 0]

# 6. Sort by course and rank
df0_sorted = df0.sort_values([course_col, rank_col])

# 7. Save result
df0_sorted.to_csv(output_path, index=False, encoding='utf-8')

print(f"Saved {len(df0_sorted)} rows to {output_path}")
```

Saved 2213032 rows to /Users/benfall/Desktop/Data Files/all_flags_n_type0_sorted.csv
(Apple Numbers here only overshoots by 2 -> might try to understand why by 4 in the other as extension)

It came to my attention that the door opening and closing times made no sense for line 18 (sometimes indicating that several people boarded within 1 or even, at times, 0 seconds. After making inquiries, I learned that the tram doors do not even have the ability to transmit such data. As such, this means that this cannot be used in the analysis. For line 80, the data is reasonable - with empty cells if the doors were not opened.

For the method introduced by Brand et al, we will need something indicative of the dwell time for #1 and #2. While ideally this would be the door openings, the only data that we always have (every row) and can verify the accuracy of is the entry and exit of the window of the station (specifically using datetime version so **DTEntreeFenetreArretReal** and **DTSortieFenetreArretReal**).
-> Will possibly calculate a new reasonable value for alpha 1 given that 60 seconds was intended for the time spent solely at the stop and more specific to the Dutch network
For #3, #4, #5, I will use **EcartDepart** which is supposedly exactly that (otherwise will try subtracting the theoretical from the MarquageArretReal or again HEntreeFenetreArretReal). Punctuality change #6 is then in terms of that. Low speed #7 evidently uses the **DistanceInterArre** - 70 m and the time between the windows and then #8 reuses what we already have.

Conception: Either XXXX-XXXX where first 4 represent ID of departure station and last 4 of arrival station - use theoretical departure time to then categorize or same idea but only use one of the window times. Logically, it makes sense to use (theoretical) departure time - although it shouldn't make a considerable difference (but then you need to go to the row before.

Hence the only relevant columns for now are the **IdCourse**, **RangArretAsc**, two station Ids (**IdArret** and **IdArretPrecedent**), **DTDepartTheo**, **EcartDepart**, **DTEntreeFenetreArretReal**, **DTSortieFenetreArretReal, DistanceInterArre**, line (**C_Ligne**), direction (**C_SensAppl** -> will just make it easier later, in theory redundant as XXXX-XXXX order will)

- IdCourse
- IdArret
- RangArretAs
- EcartDepar
- IdArretPreceden

- DistanceInterArre
- DTEntreeFenetreArretRea
- DTSortieFenetreArretReal
- DTDepartTheo
- C_Lign
- C_SensApp

Effects of things like EcartDistanc were considered negligible, especially when no deviation flag…
Ignore DistanceInterArre for RangArretAs=1 -> should theoretically anyways always be 0

```
import pandas as pd

# Load the CSV
input_path = '/Users/benfall/Desktop/Data Files/all_flags_n_type0_sorted.csv'
output_path = '/Users/benfall/Desktop/Data Files/all_flags_n_type0_sorted_special2.csv'

df_check = pd.read_csv(input_path, encoding='utf-8')

# Print the actual column names
print("Actual column names:")
print(df_check.columns.tolist())

# Define the columns one wants TO KEEP, exactly as in one's DataFrame
columns_to_keep = [
   'IdCourse',
   'IdArret',
   'RangArretAs',
   'EcartDepar',
   'IdArretPreceden',
   'DistanceInterArre',
   'DTEntreeFenetreArretRea',
   'DTSortieFenetreArretRea',
   'DTDepartTheo',
   'C_Lign',
   'C_SensApp'
]

# Subset and save
special_df = df_check[columns_to_keep]
special_df.to_csv(output_path, index=False, encoding='utf-8')

print(f"Saved {len(special_df)} rows with these columns:")
print(columns_to_keep)
print(f"to {output_path}")
```

Same number of rows evidently -> 2213032

Now we want to remove the artificial stops when crossing the border named Passage Douane (2541 on A and 2542 on R). For line 80, this meant removing the row corresponding to this stop, updating the RangArretAs after so that it remains 1,2,3,....,22, as well as IdArretPreceden to “skip” the removed stop and DistanceInterArre to be sum of segment before+after.

```
import pandas as pd
import numpy as np

# Load file
df = pd.read_csv('/Users/benfall/Desktop/Data Files/all_flags_n_type0_sorted_special.csv', encoding='utf-8')

# Force numeric type for calculations
for col in ['C_Lign', 'RangArretAs', 'IdArret', 'DistanceInterArre', 'IdArretPreceden']:
   df[col] = pd.to_numeric(df[col], errors='coerce')

# These are the stops to remove per direction for line 80
to_remove = {'R': 2542, 'A': 2541}
```

```
def adjust_group(g):
   sens = g['C_SensApp'].iloc[0]
   keep = to_remove.get(str(sens))
   # Where to drop?
   drop_idxs = g.index[g['IdArret'] == keep].tolist()
   if not drop_idxs:
       # Nothing to drop
       g['RangArretAs'] = range(1, len(g)+1)
       return g
   for idx in drop_idxs:
       # Only process if not last row
       next_idx = g.index.get_loc(idx) + 1 if idx+1 in g.index else None
       if next_idx is not None and next_idx < len(g):
           next_real_idx = g.index[next_idx]
           prev_real_idx = g.index[next_idx - 2] if next_idx-2 >= 0 else None
           # Update IdArretPreceden of next row
           if prev_real_idx is not None:
               g.at[next_real_idx, 'IdArretPreceden'] = g.at[prev_real_idx, 'IdArret']
           else:
               g.at[next_real_idx, 'IdArretPreceden'] = np.nan
           # Update DistanceInterArre
           if prev_real_idx is not None:
               # Add previous and current distances
               to_add = g.at[idx, 'DistanceInterArre'] if not np.isnan(g.at[idx, 'DistanceInterArre']) else 0
               g.at[next_real_idx, 'DistanceInterArre'] = (g.at[next_real_idx, 'DistanceInterArre'] or 0) +
to_add
   # Drop the actual stop
   g = g[g['IdArret'] != keep].copy()
   g = g.sort_values('RangArretAs') # Or whatever your journey order is
   g['RangArretAs'] = range(1, len(g)+1)
   return g

# Only work on line 80
mask = (df['C_Lign'] == 80) & (df['C_SensApp'].isin(to_remove.keys()))
df_fixed = pd.concat([
   adjust_group(grp) if m else grp
   for m, grp in df.groupby('IdCourse', group_keys=False)
])

# Save
output_path = '/Users/benfall/Desktop/Data Files/all_flags_n_type0_sorted_special_nostop.csv'
df_fixed.to_csv(output_path, index=False, encoding='utf-8')
print(f"Removed specified stops for line 80. New file: {output_path}")
```

This code does exactly that - the table now has 2,171,081 data rows (Apple Numbers counting 2,171,083).

While I could have just added c. 25 to the 60 seconds, I realised that 85 seconds would be quite large. Looking at the data, I saw that it was generally 35-50 seconds (depending if people got off) with 10-15 seconds at the stop for line 80 - still below 60 seconds for the inter-window when people get on and off, with similar values for inter-window observed for line 18.
For now, I will keep 60 seconds however I may adjust depending on how many bottlenecks I identify…

Ligne 18

Station	Direction A	Direction R
Grand-Lancy, Palettes	7422	7423
Grand-Lancy, Pontets	1075	1077
Plan-les-Ouates, Tréfle-Blanc	1361	1362
Lancy-Bachet, gare	117	118
Grand-Lancy, De-Staél	1296	1297

Carouge, Rondeau	374	375
Carouge, Ancienne	42	43
Carouge, Marché	5315	5316
Carouge, Armes	56	57
Genéve, Blanche	130	131
Genéve, Augustins	71	72
Genéve, Pont-d'Arve	1127	1128
Genéve, Plainpalais	1039	1041
Genéve, Place de Neuve	1063	1064
Genéve, Bel-Air	86	87
Genéve, Coutance	3349	356
Genéve, gare Cornavin	422	427
Genéve, Lyon	801	800
Genéve, Poterie	1082	1081
Genéve, Servette	1238	1237
Genéve, Vieusseux	1441	1440
Vernier, Bouchet	155	156
Vernier, Balexert	91	92
Vernier, Avanchets-Etang	77	78
Vernier, Blandonnet	134	137
Meyrin, Jardin-Alpin-Vivarium	702	703
Meyrin, Bois-du-Lan	3002	3003
Meyrin, village	924	925
Meyrin, Hépital de La Tour	686	687
Meyrin, Maisonnex	806	807
Meyrin, CERN	260	261

Ligne 80

Station	Direction A	Direction R
Geneve, Bel-Air	2321	2321
Genéve, Stand	1310	1308
Genéve, Palladium	1022	1023
Geneve, Jonction	720	721
Genéve, Queue-d'Arve	7166	7165
Lancy-Pont-Rouge, gare/Etoile	522	523
Grand-Lancy, Stade de Geneve	1298	1301
Lancy-Bachet, gare	122	121
Plan-les-Ouates, Communes-Réunies	335	334
Plan-les-Ouates, Aviateurs	3303	3304
Plan-les-Ouates, mairie	1037	1038
Plan-les-Ouates, Vélodrome	1414	1415
Plan-les-Ouates, Arare	52	53
Plan-les-Ouates, Galaise	630	631
Plan-les-Ouates, Au Contour	2117	353
Perly, En Louche	782	780
Perly, Ravieres	3301	3302
Perly, douane	982	983
Saint-Julien-en-Genevois, douane	6495	6496
Saint-Julien-en-Genevois, Hutins	6497	6498
Saint-Julien-en-Genevois, centre	2521	2522
Saint-Julien-en-Genevois, SNCF	7444	7444

Segments for line 18 direction A:

74221075, 10751361, 13610117, 01171296, 12960374, 03740042, 00425315, 53150056, 00560130, 01300071, 00711127, 11271039, 10391063, 10630086, 00863349, 33490422, 04220801, 08011082, 10821238, 12381441, 14410155, 01550091, 00910077, 00770134, 01340702, 07023002, 30020924, 09240686, 06860806, 08060260

Segments for line 18 direction R:
02610807, 08070687, 06870925, 09253003, 30030703, 07030137, 01370078, 00780092, 00920156, 01561440, 14401237, 12371081, 10810800, 08000427, 04270356, 03560087, 00871064, 10641041, 10411128, 11280072, 00720131, 01310057, 00575316, 53160043, 00430375, 03751297, 12970118, 01181362, 13621077, 10777423

Segments for line 80 direction A:
23211310, 13101022, 10220720, 07207166, 71660522, 05221298, 12980122, 01220335, 03353303, 33031037, 10371414, 14140052, 00520630, 06302117, 21170782, 07823301, 33010982, 09826495, 64956497, 64972521, 25217444

Segments for line 80 direction R:
74442522, 25226498, 64986496, 64960983, 09833302, 33020780, 07800353, 03530631, 06310053, 00531415, 14151038, 10383304, 33040334, 03340121, 01211301, 13010523, 05237165, 71650721, 07211023, 10231308, 13082321

- Standard is DistanceInterArre-70

Bel Air entry is 60 meters before the stop (irrelevant to our calculations)
Stand's entry window is 60 meters before the stop so 23211310 is DistanceInterArre-95 (for A)
The "dwell-time" will also be influenced (would be longer for Stand for A) - 95 meters
Stand's exit window is 10 meters after the stop so 13082321 is DistanceInterArre-45 (for R)
The "dwell-time" will also be influenced (would be shorter for Stand for R) - 45 meters
The one for Bel-Air again doesn't matter

Now for line 18
Lyon's entry window is 60 meters before the stop so 04220801 is DistanceInterArre-95 (for A)
The "dwell-time" will also be influenced (would be longer for Lyon) - 95 meters
Poterie's entry window is 60 meters before the stop so 08011082 is DistanceInterArre-95 (for A)
The "dwell-time" will also be influenced (would be longer for Poterie) - 95 meters

Trefle-blanc's exit window is 10 meters after the stop so 13621077 is DistanceInterArre-45 (for R - doesn't matter as identified by the XXXXXXXX)
The "dwell-time" will also be influenced (would be shorter for Trefle-blanc) - 45 meters

```
import pandas as pd
import numpy as np

# Path to your source file
file = '/Users/benfall/Desktop/Data Files/all_flags_n_type0_sorted_special_nostop.csv'
df = pd.read_csv(file, encoding='utf-8')

# -- Ensure time columns are datetime
```

```
df['DTEntreeFenetreArretRea'] = pd.to_datetime(df['DTEntreeFenetreArretRea'])
df['DTSortieFenetreArretRea'] = pd.to_datetime(df['DTSortieFenetreArretRea'])

# --- 1. Build Segment Label ('XXXXYYYY'), zero-padded, robust to NaN (no dash)
def pad4(val):
    try:
        return str(int(val)).zfill(4)
    except:
        return "null"

df['Segment_XXXXYYYY'] = df['IdArretPreceden'].apply(pad4) + df['IdArret'].apply(pad4)

# --- 2. Compute dwell_time (only for through-stops, else NaN)
df['dwell_time'] = (df['DTSortieFenetreArretRea'] - df['DTEntreeFenetreArretRea']).dt.total_seconds()
df.loc[df['RangArretAs'] == 1, 'dwell_time'] = np.nan

# --- 3. Compute segment_travel_time (current IN minus previous OUT, per trip)
df = df.sort_values(['IdCourse', 'RangArretAs']).reset_index(drop=True)
df['segment_travel_time'] = (
    df['DTEntreeFenetreArretRea'] -
    df.groupby('IdCourse')['DTSortieFenetreArretRea'].shift(1)
).dt.total_seconds()

# --- 4. Compute distance with special rules, NaN for first stop
def compute_distance(row):
    if row['RangArretAs'] == 1:
        return np.nan
    seg = row['Segment_XXXXYYYY']
    d = row['DistanceInterArre']
    if pd.isna(d):
        return np.nan
    if seg in ['23211310', '04220801', '08011082']:
        return d - 95
    elif seg in ['13621077', '13082321']:
        return d - 45
    else:
        return d - 70

df['distance'] = df.apply(compute_distance, axis=1)

# --- 5. Compute segment_speed in km/h (distance / segment_travel_time * 3.6)
df['segment_speed'] = df['distance'] / df['segment_travel_time'] * 3.6

# --- 6. Save to file (all original columns + new ones appended)
out_file = '/Users/benfall/Desktop/Data
Files/all_flags_n_type0_sorted_special_WITH_SEG_AND_TIMES_DIST_SPEED.csv'
df.to_csv(out_file, index=False, encoding='utf-8')

print('✅ Finished! All columns added and file saved as:')
print(out_file)
print('\nSample of new columns:')
print(df[['Segment_XXXXYYYY', 'dwell_time', 'segment_travel_time', 'distance',
'segment_speed']].head(12))
```

This file was then renamed OG.csv

-> To get /Users/benfall/Desktop/Data Files/all_flags_n_type0_sorted_special.csv
We started from the original "ben_arrets.csv", filtering to standard passenger service (TypeCourse == 0) if present, sorting for per-journey/segment analysis (code from Dr. C. Cortes Balcells), then dropping all rows with at least one Y in the four flags (FlagDeviatio, FlagDelocalisatio, FlagRecala)
Defined the column names we want to use for grouping/sorting
course_col = '\ufeffIdCourse' # Use the exact column name from your DataFrame
rank_col = 'RangArretAs' # Adjust if necessary

Ensured needed columns are numeric (vital for correct sorting)
df[course_col] = pd.to_numeric(df[course_col], errors='coerce')
df[rank_col] = pd.to_numeric(df[rank_col], errors='coerce')

df['C_TypeAppl'] = pd.to_numeric(df['C_TypeAppl'], errors='coerce')

Filtered for commercial trips
df0 = df[df['C_TypeAppl'] == 0]

Sorted by course and stop-sequence in numeric order
df0 = df0.sort_values([course_col, rank_col]).

Removed the artificial douane stop

```
import pandas as pd
import numpy as np

# PARAMETERS
alpha1 = 60   # sec, large dwell time
alpha2 = 120  # sec, large dwell time variation
alpha3 = -60  # sec, early departure (threshold)
alpha4 = 180  # sec, late departure (threshold)
alpha5 = 300  # sec, large dep. time variation
alpha6 = 60   # sec, punctuality change to previous
alpha7 = 15   # km/h, low speed
alpha8 = 60   # sec, travel time excess over free flow

# 1. Load your data
file = '/Users/benfall/Desktop/Data Files/all_flags_n_type0_sorted_special_nostop_segments.csv'
df = pd.read_csv(file, encoding='utf-8')
df['DTDepartTheo'] = pd.to_datetime(df['DTDepartTheo'])
df['DTEntreeFenetreArretRea'] = pd.to_datetime(df['DTEntreeFenetreArretRea'])
df['DTSortieFenetreArretRea'] = pd.to_datetime(df['DTSortieFenetreArretRea'])

# 2. Period assignment using *theoretical* departure time
def assign_period(dt):
   hour = dt.hour
   day = dt.dayofweek
   if day == 5:
       return "Saturdays"
   if day == 6:
       return "Sundays"
   if 7 <= hour < 9:
       return "AM peak"
   elif 9 <= hour < 16:
       return "Day period"
   elif 16 <= hour < 19:
       return "PM peak"
   elif 19 <= hour < 23:
       return "Evening"
   else:
       return "Other"
df['period'] = df['DTDepartTheo'].apply(assign_period)

# 3. Filter for one segment if desired
segment_id = "74221075"
df_seg = df[df["SegmentID"] == segment_id].copy()
```

```
# 4. BOTTLE NECK LOGIC

# Dwell time
df_seg['dwell_time'] = (df_seg['DTSortieFenetreArretRea'] -
df_seg['DTEntreeFenetreArretRea']).dt.total_seconds()

# Large dwell time
df_seg["large_dwell_time"] = df_seg["dwell_time"] > alpha1

# Large variation in dwell time per period
period_dwell_var = df_seg.groupby('period')["dwell_time"].agg(lambda x: np.percentile(x,85) -
np.percentile(x,15) if len(x.dropna())>=2 else 0)
df_seg["large_var_in_dwell"] = df_seg["period"].map(lambda p: period_dwell_var[p] > alpha2)

# Early/late departure
df_seg["early_departure"] = df_seg["EcartDepar"] < alpha3
df_seg["late_departure"]  = df_seg["EcartDepar"] > alpha4

# Large var in departure per period
period_dep_var = df_seg.groupby('period')["EcartDepar"].agg(lambda x: np.percentile(x,85) - np.percentile(x,15)
if len(x.dropna())>=2 else 0)
df_seg["large_var_in_departure"] = df_seg["period"].map(lambda p: period_dep_var[p] > alpha5)

# Punctuality change previous
df_seg = df_seg.sort_values(["DTDepartTheo"])
df_seg["EcartDepar_prev"] = df_seg["EcartDepar"].shift(1)
df_seg["punct_change_prev"] = (df_seg["EcartDepar"] - df_seg["EcartDepar_prev"]).abs() > alpha6

# Low speed
df_seg["low_speed"] = df_seg["segment_speed"] < alpha7

# Large travel time vs free flow (15th percentile, Sundays, all data)
sundays = df[df['period'] == 'Sundays']
freeflow_15th = sundays.groupby('SegmentID')['segment_travel_time'].quantile(0.15)
df_seg["freeflow_15th"] = df_seg["SegmentID"].map(freeflow_15th)
df_seg["large_travel_time_vs_freeflow"] = (df_seg["segment_travel_time"] - df_seg["freeflow_15th"]) > alpha8

# At least one
criteria_cols = [
   "large_dwell_time", "large_var_in_dwell", "early_departure", "late_departure",
   "large_var_in_departure", "punct_change_prev", "low_speed", "large_travel_time_vs_freeflow"
]
df_seg["any_criterion"] = df_seg[criteria_cols].any(axis=1)

# 5. SUMMARY: Count per period, *plus* number of trips
summary = df_seg.groupby("period")[criteria_cols + ["any_criterion"]].sum().astype(int)
summary["num_trips"] = df_seg.groupby("period").size()

summary = summary.reset_index()

print(summary)
summary.to_csv("/Users/benfall/Desktop/segment74221075_bottleneck_summary_with_n.csv", index=False)
```

Evening period here is only until 11 PM - using DTDepartTheo
Has other which captures the rest
Instead evening -> 7 PM - 7 AM (no spillover for Friday which would otherwise take from Saturday for instance)
8686 instances of segment 74221075
Following code combines them:

```
import pandas as pd
```

```
import numpy as np

# === PARAMETERS (Brand et al. style, can be changed) ===
alpha1 = 60   # sec, large dwell time
alpha2 = 120  # sec, large dwell time variation
alpha3 = -60  # sec, early departure (threshold)
alpha4 = 180  # sec, late departure (threshold)
alpha5 = 300  # sec, large dep. time variation
alpha6 = 60   # sec, punctuality change to previous
alpha7 = 15   # km/h, low speed
alpha8 = 60   # sec, travel time excess over free flow

# === LOAD DATA ===
file = '/Users/benfall/Desktop/Data Files/all_flags_n_type0_sorted_special_nostop_segments.csv'
df = pd.read_csv(file, encoding='utf-8')
df['DTDepartTheo'] = pd.to_datetime(df['DTDepartTheo'])
df['DTEntreeFenetreArretRea'] = pd.to_datetime(df['DTEntreeFenetreArretRea'])
df['DTSortieFenetreArretRea'] = pd.to_datetime(df['DTSortieFenetreArretRea'])

# === PERIOD ASSIGNMENT using DTDepartTheo ===
def assign_period(dt):
   hour = dt.hour
   day = dt.dayofweek
   if day == 5:
       return "Saturdays"
   if day == 6:
       return "Sundays"
   if 7 <= hour < 9:
       return "AM peak"
   elif 9 <= hour < 16:
       return "Day period"
   elif 16 <= hour < 19:
       return "PM peak"
   else:
       return "Other"
df['period'] = df['DTDepartTheo'].apply(assign_period)

# === FILTER FOR A SEGMENT (or remove for whole dataset) ===
segment_id = "74221075"    # Set to segment, or comment to process all
df_seg = df[df["SegmentID"] == segment_id].copy()

# === BOTTLE NECK CALCULATIONS ===
# Dwell time
df_seg['dwell_time'] = (df_seg['DTSortieFenetreArretRea'] -
df_seg['DTEntreeFenetreArretRea']).dt.total_seconds()

# 1. Large dwell time
df_seg["large_dwell_time"] = df_seg["dwell_time"] > alpha1

# 2. Large variation in dwell time per period
period_dwell_var = df_seg.groupby('period')["dwell_time"].agg(
   lambda x: np.percentile(x, 85) - np.percentile(x, 15) if len(x.dropna()) >= 2 else 0
)
df_seg["large_var_in_dwell"] = df_seg["period"].map(lambda p: period_dwell_var[p] > alpha2)

# 3. Early departure
df_seg["early_departure"] = df_seg["EcartDepar"] < alpha3
# 4. Late departure
df_seg["late_departure"]  = df_seg["EcartDepar"] > alpha4
```

```
# 5. Large variation in departure per period
period_dep_var = df_seg.groupby('period')["EcartDepar"].agg(
    lambda x: np.percentile(x, 85) - np.percentile(x, 15) if len(x.dropna()) >= 2 else 0
)
df_seg["large_var_in_departure"] = df_seg["period"].map(lambda p: period_dep_var[p] > alpha5)

# 6. Punctuality change compared to previous stop (in sequence by theoretical dep)
df_seg = df_seg.sort_values(["DTDepartTheo"])
df_seg["EcartDepar_prev"] = df_seg["EcartDepar"].shift(1)
df_seg["punct_change_prev"] = (df_seg["EcartDepar"] - df_seg["EcartDepar_prev"]).abs() > alpha6

# 7. Low speed
df_seg["low_speed"] = df_seg["segment_speed"] < alpha7

# 8. Large travel time vs free flow (15th percentile on Sundays, from whole DF)
sundays = df[df['period'] == 'Sundays']
freeflow_15th = sundays.groupby('SegmentID')['segment_travel_time'].quantile(0.15)
df_seg["freeflow_15th"] = df_seg["SegmentID"].map(freeflow_15th)
df_seg["large_travel_time_vs_freeflow"] = (df_seg["segment_travel_time"] - df_seg["freeflow_15th"]) > alpha8

# At least one criterion
criteria_cols = [
    "large_dwell_time", "large_var_in_dwell", "early_departure", "late_departure",
    "large_var_in_departure", "punct_change_prev", "low_speed", "large_travel_time_vs_freeflow"
]
df_seg["any_criterion"] = df_seg[criteria_cols].any(axis=1)

# === CREATE SUMMARY TABLE ===
summary = (
    df_seg.groupby("period")[criteria_cols + ["any_criterion"]].sum().astype(int)
)
summary["num_trips"] = df_seg.groupby("period").size()
summary = summary.reset_index()

# === SAVE/PRINT RESULT ===
print(summary)
summary.to_csv("/Users/benfall/Desktop/segment74221075_bottleneck_summary_FULL.csv", index=False)
```

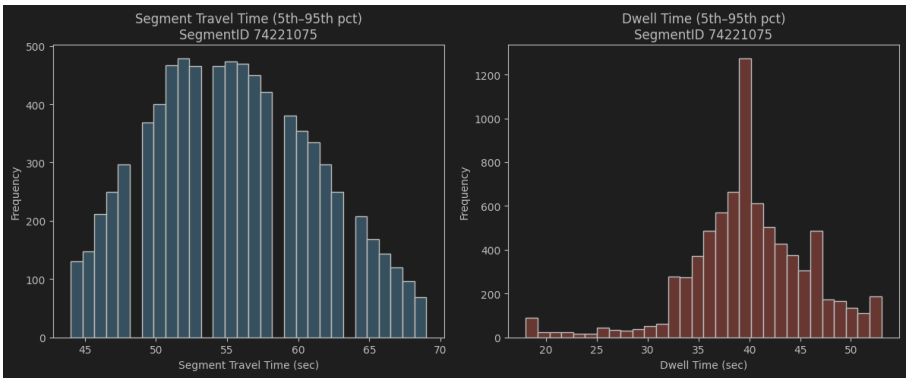
Using Data Wrangler, I was able to observe that there were rows with RangArretAsc larger than 31.
I isolated these and got 5746 rows (with none of the 4 flags raised, most of which were non-commercial), corresponding to line 18. While sometimes representing known segments, often completely different stations were included such as Ziplo which are not part of the line.

```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np

file = '/Users/benfall/Desktop/Data Files/all_flags_n_type0_sorted_special_nostop_segments.csv'
segment_id = '74221075'

df = pd.read_csv(file, encoding='utf-8')
df['SegmentID'] = df['SegmentID'].astype(str).str.zfill(8)
df_seg = df[df['SegmentID'] == segment_id].copy()

# Calculate dwell time if missing
df_seg['DTEntreeFenetreArretRea'] = pd.to_datetime(df_seg['DTEntreeFenetreArretRea'])
df_seg['DTSortieFenetreArretRea'] = pd.to_datetime(df_seg['DTSortieFenetreArretRea'])
df_seg['dwell_time'] = (
    df_seg['DTSortieFenetreArretRea'] - df_seg['DTEntreeFenetreArretRea']
).dt.total_seconds()

segment_times = df_seg['segment_travel_time'].dropna()
dwell_times = df_seg['dwell_time'].dropna()

print(f"\nNumber of segment travel time datapoints: {len(segment_times)}")
print(f"Number of dwell time datapoints:         {len(dwell_times)}")
```

```
# Compute 5th-95th percentile focus
p5_st, p95_st = np.percentile(segment_times, 5), np.percentile(segment_times, 95)
p5_dw, p95_dw = np.percentile(dwell_times, 5), np.percentile(dwell_times, 95)

segment_center = segment_times[(segment_times >= p5_st) & (segment_times <= p95_st)]
dwell_center = dwell_times[(dwell_times >= p5_dw) & (dwell_times <= p95_dw)]

print(f"\nSegment travel time 5th-95th percentile: [{p5_st:.1f}, {p95_st:.1f}] sec, showing
{len(segment_center)}/{len(segment_times)} values")
print(f"Dwell time 5th-95th percentile:          [{p5_dw:.1f}, {p95_dw:.1f}] sec, showing
{len(dwell_center)}/{len(dwell_times)} values")

plt.figure(figsize=(12, 5))

plt.subplot(1,2,1)
plt.hist(segment_center, bins=30, color='skyblue', edgecolor='black')
plt.title(f'Segment Travel Time (5th-95th pct)\nSegmentID {segment_id}')
plt.xlabel('Segment Travel Time (sec)')
plt.ylabel('Frequency')

plt.subplot(1,2,2)
plt.hist(dwell_center, bins=30, color='salmon', edgecolor='black')
plt.title(f'Dwell Time (5th-95th pct)\nSegmentID {segment_id}')
plt.xlabel('Dwell Time (sec)')
plt.ylabel('Frequency')

plt.tight_layout()
plt.show()
```

Number of segment travel time datapoints: 8619
Number of dwell time datapoints: 8686

Segment travel time 5th–95th percentile: [44.0, 69.0] sec, showing 7914/8619 values
Dwell time 5th–95th percentile: [18.0, 53.0] sec, showing 7854/8686 values



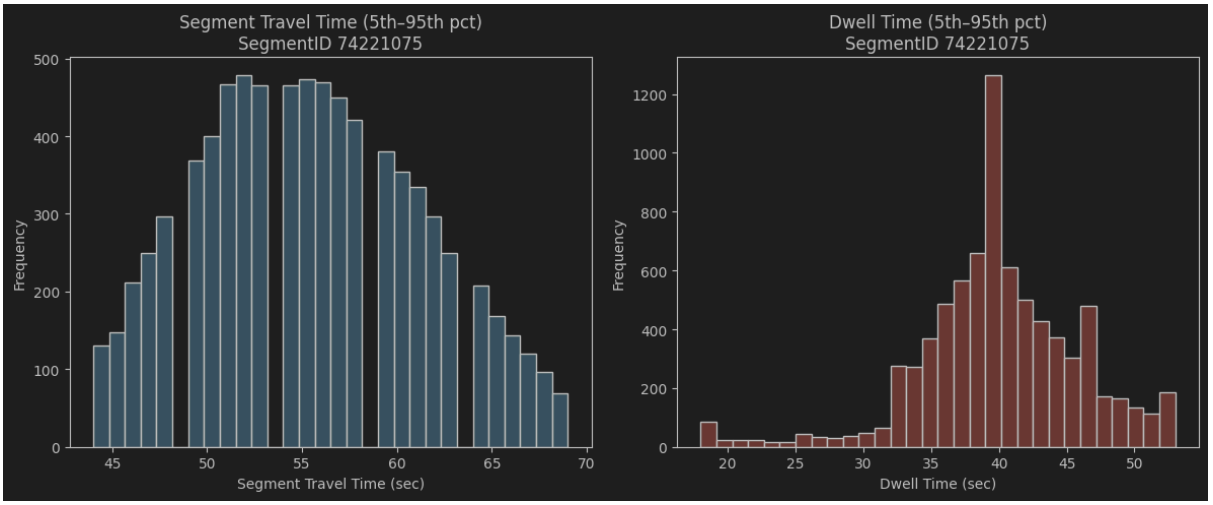
While it originally bothered me that there were more dwell time data points than segment travel time datapoints, this simply indicated that sometimes the segment did appear as the first stop of the journey (for which the segment_travel_time is not defined while dwell_time is). I hadn't considered this for this particular segment as I naively expected it to always fall in line with the schedule which evidently isn't realistic. Verifying, there were exactly 67.

```
import pandas as pd

file = '/Users/benfall/Desktop/Data Files/all_flags_n_type0_sorted_special_nostop_segments.csv'
segment_id = '74221075'

df = pd.read_csv(file, encoding='utf-8')
df['SegmentID'] = df['SegmentID'].astype(str).str.zfill(8)  # ensure 8-char format

# Filter for desired segment
df_seg = df[df['SegmentID'] == segment_id].copy()

# Select rows where segment_travel_time is NaN (i.e., first stop of the journey for this segment)
first_stop_rows = df_seg[df_seg['segment_travel_time'].isna()]

print(f"Number of 'first stop' rows for SegmentID {segment_id}: {len(first_stop_rows)}")
print(first_stop_rows[['IdCourse', 'RangArretAs', 'DTEntreeFenetreArretRea',
'DTSortieFenetreArretRea']].head())

# Optionally, save to CSV for review:
first_stop_rows.to_csv('/Users/benfall/Desktop/Data Files/first_stop_of_segment_74221075.csv', index=False,
encoding='utf-8')
```

Reflecting further, I realised that just as I didn't have segment travel times for the first trips, I did not want dwell times, as they would be meaningful as the vehicle may have simply entered the zone before to start.

```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np

file = '/Users/benfall/Desktop/Data Files/all_flags_n_type0_sorted_special_nostop_segments.csv'
segment_id = '74221075'

df = pd.read_csv(file, encoding='utf-8')
df['SegmentID'] = df['SegmentID'].astype(str).str.zfill(8)
df_seg = df[df['SegmentID'] == segment_id].copy()

# Calculate dwell time (if not present already)
df_seg['DTEntreeFenetreArretRea'] = pd.to_datetime(df_seg['DTEntreeFenetreArretRea'])
df_seg['DTSortieFenetreArretRea'] = pd.to_datetime(df_seg['DTSortieFenetreArretRea'])
df_seg['dwell_time'] = (
   df_seg['DTSortieFenetreArretRea'] - df_seg['DTEntreeFenetreArretRea']
).dt.total_seconds()

# Keep only rows where segment_travel_time exists (i.e., NOT first stop in journey)
valid = df_seg[df_seg['segment_travel_time'].notnull()]

print(f"Number of data points used for both metrics: {len(valid)}")

# Now proceed just as before, but only using these rows:
segment_times = valid['segment_travel_time']
dwell_times = valid['dwell_time']

p5_st, p95_st = np.percentile(segment_times, 5), np.percentile(segment_times, 95)
p5_dw, p95_dw = np.percentile(dwell_times, 5), np.percentile(dwell_times, 95)

segment_center = segment_times[(segment_times >= p5_st) & (segment_times <= p95_st)]
dwell_center   = dwell_times[(dwell_times >= p5_dw) & (dwell_times <= p95_dw)]

print(f"\nSegment travel time 5th–95th percentile: [{p5_st:.1f}, {p95_st:.1f}] sec, showing
{len(segment_center)}/{len(segment_times)} values")
print(f"Dwell time 5th–95th percentile:          [{p5_dw:.1f}, {p95_dw:.1f}] sec, showing
{len(dwell_center)}/{len(dwell_times)} values")

plt.figure(figsize=(12, 5))
plt.subplot(1,2,1)
plt.hist(segment_center, bins=30, color='skyblue', edgecolor='black')
plt.title(f'Segment Travel Time (5th–95th pct)\nSegmentID {segment_id}')
plt.xlabel('Segment Travel Time (sec)')
plt.ylabel('Frequency')
plt.subplot(1,2,2)
plt.hist(dwell_center, bins=30, color='salmon', edgecolor='black')
plt.title(f'Dwell Time (5th–95th pct)\nSegmentID {segment_id}')
plt.xlabel('Dwell Time (sec)')
plt.ylabel('Frequency')
plt.tight_layout()
plt.show()
```

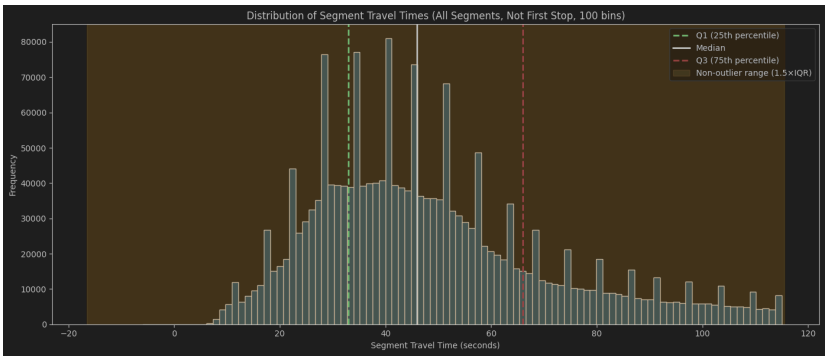
Number of data points used for both metrics: 8619
Segment travel time 5th–95th percentile: [44.0, 69.0] sec, showing 7914/8619 values
Dwell time 5th–95th percentile: [18.0, 53.0] sec, showing 7794/8619 values



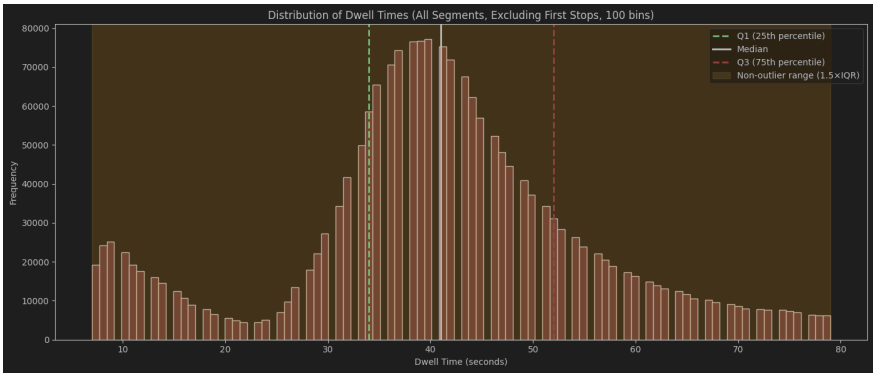
While tempting to remove outliers, this would not be rigorous considering we may remove serious bottlenecks.

Originally used +/- 35 meters methods which in the case of a few segments was false!

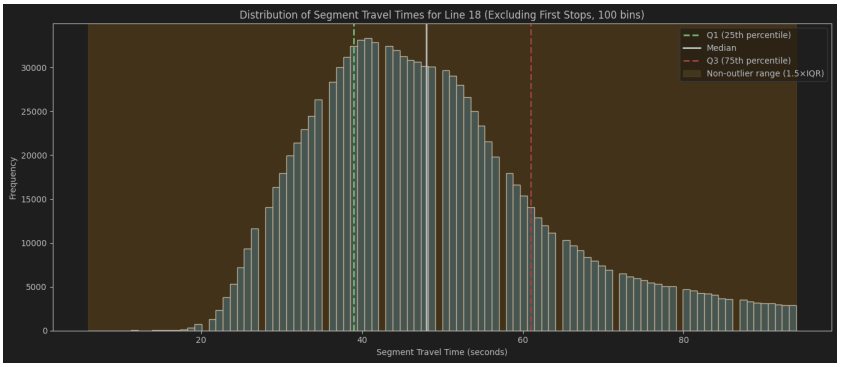
Number of datapoints (non-first stops): 2080647
Median: 46.00 sec | Q1: 33.00 sec | Q3: 66.00 sec | IQR: 33.00 sec
Outlier bounds: [-16.50, 115.50] sec
Number of points in non-outlier range: 1940857



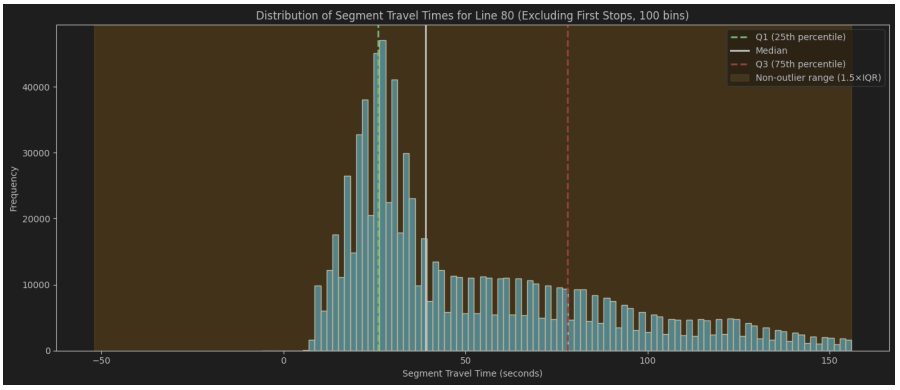
Number of dwell time datapoints (not first stop in trip): 2080647
Median: 41.00 sec | Q1: 34.00 sec | Q3: 52.00 sec | IQR: 18.00 sec
Outlier bounds: [7.00, 79.00] sec
Number in non-outlier range: 1927501



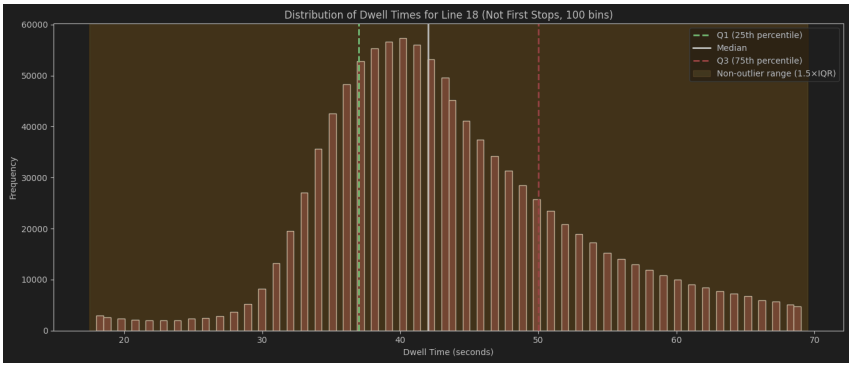
Number of datapoints (line 18, valid stops): 1203397
Median: 48.00 sec | Q1: 39.00 sec | Q3: 61.00 sec | IQR: 22.00 sec
Outlier bounds: [6.00, 94.00] sec
Number of points in non-outlier range: 1106872



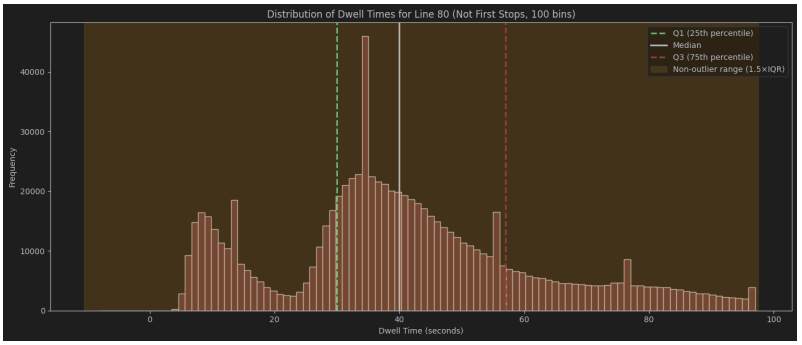
Number of datapoints (line 80, valid stops): 877250
Median: 39.00 sec | Q1: 26.00 sec | Q3: 78.00 sec | IQR: 52.00 sec
Outlier bounds: [-52.00, 156.00] sec
Number of points in non-outlier range: 846238



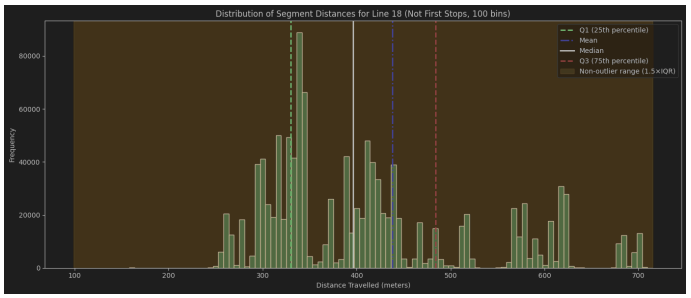
Number of dwell time datapoints (line 18, not first in trip): 1203397
Median: 42.00 sec | Q1: 37.00 sec | Q3: 50.00 sec | IQR: 13.00 sec
Outlier bounds: [17.50, 69.50] sec
Number in non-outlier range: 1067344



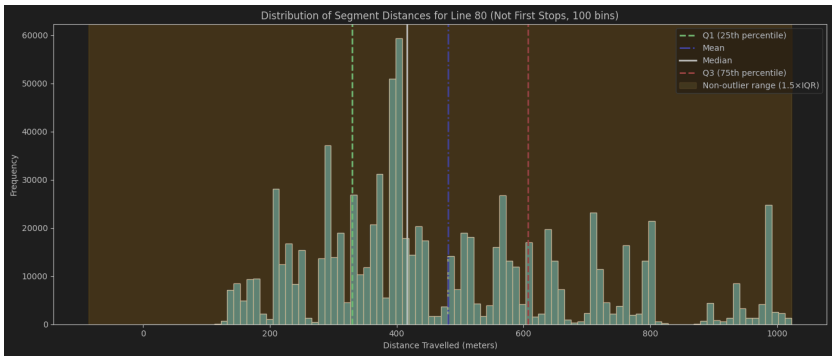
Number of dwell time datapoints (line 80, not first in trip): 877250
Median: 40.00 sec | Q1: 30.00 sec | Q3: 57.00 sec | IQR: 27.00 sec
Outlier bounds: [-10.50, 97.50] sec
Number in non-outlier range: 825595



Datapoints (line 18, not first in trip): 1203397
Mean: 437.99 m | Median: 396.00 m | Q1: 330.00 m | Q3: 484.00 m | IQR: 154.00 m
Outlier bounds: [99.00, 715.00] m | Non-outlier points: 1157705



Datapoints (line 80, not first in trip): 877250
Mean: 481.12 m | Median: 416.00 m | Q1: 330.00 m | Q3: 607.00 m | IQR: 277.00 m
Outlier bounds: [-85.50, 1022.50] m | Non-outlier points: 871873



*The last two graphs are somewhat misleading due to some error in the +/- 35 method

Using the +/- 35 meter method and all given alpha values

segment74221075_bottleneck_summary_with_n

period	large_dwell_time	large_var_in_dwell	early_departure	late_departure	large_var_in_departure	punct_change_prev	low_speed	large_travel_time_vs_freeflow	any_criterion	num_trips
AM peak	19	0	4	11	0	181	1	0	195	867
Day period	53	0	43	34	0	671	5	1	709 (28.9%)	2435
Evening	15	0	57	36	0	426	4	0	445 (36.3%)	1226
Other	8	0	17	25	0	307	2	0	316	883
PM peak	27	0	21	100	0	478	4	0	501	1207

Saturdays	27	0	24	36	0	463	1	0	484	1510
Sundays	5	0	21	6	0	155	0	0	164	558

Clearly something wrong with large var in dwell, large_var_in_departure, large_travel_time_vs_freeflow
Reached out to inquire how these are calculated but no response

Appendix 2:

Given the large dataset, I only now really began to appreciate testing on smaller samples to check before that everything seems to be going well.
Example preview tool:

```
import re
import pandas as pd

path = '/Users/benfall/Desktop/arrets_ben.csv'

# 1) Grab the header line + dash line
with open(path, encoding='utf-8') as f:
    header_line = f.readline().rstrip('\n')
    dash_line   = f.readline().rstrip('\n')

# 2) Infer colspecs from runs of dashes
colspecs = [(m.start(), m.end()) for m in re.finditer(r'-+', dash_line)]

# 3) Slice out the raw names from the header
raw_names = [header_line[start:end].strip() for start, end in colspecs]

# 4) De-duplicate any repeated names by appending a counter
seen = {}
names = []
for nm in raw_names:
    if nm in seen:
        seen[nm] += 1
        names.append(f"{nm}_{seen[nm]}")
    else:
        seen[nm] = 0
        names.append(nm)

# 5) Read only the first 10,000 rows for a fast preview!
df_preview = pd.read_fwf(
    path,
    colspecs=colspecs,
    names=names,
    skiprows=[0, 1],      # skip header and dash line
    na_values=['NULL'],   # treat 'NULL' as NaN
    dtype=str,
    nrows=10000           # only load first 10,000 rows for testing
)

# 6) Clean column names, just in case
df_preview.columns = df_preview.columns.str.strip()

# 7) Show what we've got!
# print(df_preview.shape)            # Rows/columns loaded
# print(df_preview.columns.tolist()) # List column names to verify they're clean
```

```
df_preview.head()
```

len() in pandas does not include the row with the column names. It only counts the number of data rows, not the header.
Initial inspection at the deltas to reflect on alpha value

```
import re
import pandas as pd
from difflib import get_close_matches

# ====== 1. File path ======
path = r'/Users/benfall/Desktop/arrets_ben.csv'

# ====== 2. Read header + dashes to infer structure ======
with open(path, encoding='utf-8') as f:
    header_line = f.readline().rstrip('\n')
    dash_line = f.readline().rstrip('\n')

# ====== 3. Infer fixed-width column specs ======
colspecs = [(m.start(), m.end()) for m in re.finditer(r'--+', dash_line)]

# ====== 4. Extract and clean column names ======
raw_names = [header_line[start:end].strip() for start, end in colspecs]

# De-duplicate any repeated names
seen = {}
names = []
for nm in raw_names:
    if nm in seen:
        seen[nm] += 1
        names.append(f"{nm}_{seen[nm]}")
    else:
        seen[nm] = 0
        names.append(nm)

# ====== 5. Read the fixed-width file into DataFrame ======
df = pd.read_fwf(
    path,
    colspecs=colspecs,
    names=names,
    skiprows=[0, 1],
    na_values=['NULL'],
    dtype=str
)

print(f" Loaded shape: {df.shape}")
print(" First few cleaned column names:", df.columns[:10].tolist())

# ====== 6. Clean column names (remove all whitespace) ======
df.columns = [''.join(col.split()) for col in df.columns]

# ====== 7. Define expected columns ======
expected = [
    'DTEntreeFenetreArretReal',
    'DTSortieFenetreArretReal',
    'DTOuverturePortes',
    'DTFermeturesPortes'
]

# ====== 8. Try to match actual columns ======
actual_keep = []
for col in expected:
    if col in df.columns:
        actual_keep.append(col)
        print(f" Found exact match: {col}")
    else:
        match = get_close_matches(col, df.columns, n=1)
        if match:
            print(f" Closest match for '{col}': '{match[0]}' — using it")
            actual_keep.append(match[0])
        else:
            raise KeyError(f" Column not found and no close match: {col}")
```

```
# ====== 9. Select and copy relevant columns ======
df_clean = df[actual_keep].copy()

# ====== 10. Convert to datetime ======
for col in actual_keep:
   df_clean[col] = pd.to_datetime(df_clean[col], errors='coerce')

# Rename for clarity
df_clean.columns = ['entree_fenetre', 'sortie_fenetre', 'ouverture_portes', 'fermeture_portes']

# ====== 11. Compute deltas ======
df_clean['delta_fenetre'] = df_clean['sortie_fenetre'] - df_clean['entree_fenetre']
df_clean['delta_portes'] = df_clean['fermeture_portes'] - df_clean['ouverture_portes']
df_clean['diff_delta'] = df_clean['delta_fenetre'] - df_clean['delta_portes']

# Optional: also in seconds
df_clean['delta_fenetre_sec'] = df_clean['delta_fenetre'].dt.total_seconds()
df_clean['delta_portes_sec'] = df_clean['delta_portes'].dt.total_seconds()
df_clean['diff_delta_sec'] = df_clean['diff_delta'].dt.total_seconds()

# ====== 12. Show preview ======
print(df_clean[['delta_fenetre', 'delta_portes', 'diff_delta']].head())

# ====== 13. Save to CSV ======
output_path = r'/Users/benfall/Desktop/arrets_ben_deltas.csv'
df_clean.to_csv(output_path, index=False)
print(f" Saved to: {output_path}")
```

```
 Loaded shape: (2301409, 73)
 First few cleaned column names: ['\ufeffIdCourse', 'IdArret', 'RangArretAs', 'DateCours', 'HDepartThe', 'HArriveeThe', 'DistanceThe',
'TempsInterArretThe', 'EcartDepar', 'EcartDistanc']
 Closest match for 'DTEntreeFenetreArretReal': 'DTEntreeFenetreArretRea' — using it
 Closest match for 'DTSortieFenetreArretReal': 'DTSortieFenetreArretRea' — using it
Found exact match: DTOuverturePortes
 Found exact match: DTFermeturesPortes
   delta_fenetre    delta_portes      diff_delta
0 0 days 00:09:03 0 days 00:00:01 0 days 00:09:02
1 0 days 00:12:46 0 days 00:00:01 0 days 00:12:45
2 0 days 00:00:06           NaT           NaT
3 0 days 00:00:37 0 days 00:00:01 0 days 00:00:36
4 0 days 00:00:24 0 days 00:00:00 0 days 00:00:24
 Saved to: /Users/benfall/Desktop/arrets_ben_deltas.csv
```

```
import pandas as pd
import numpy as np

# PARAMETERS (Brand et al. reference values)
alpha1 = 60   # sec, large dwell time
alpha2 = 120  # sec, large dwell time variation
alpha3 = -60  # sec, early departure (threshold)
alpha4 = 180  # sec, late departure (threshold)
alpha5 = 300  # sec, large dep. time variation
alpha6 = 60   # sec, punctuality change to previous
alpha7 = 15   # km/h, low speed
alpha8 = 60   # sec, travel time excess over free flow

# == 1. Load (already processed) data ==
file = '/Users/benfall/Desktop/Data Files/all_flags_n_type0_sorted_special_nostop_segments.csv'
df = pd.read_csv(file, encoding='utf-8')
df['DTEntreeFenetreArretRea'] = pd.to_datetime(df['DTEntreeFenetreArretRea'])
df['DTSortieFenetreArretRea'] = pd.to_datetime(df['DTSortieFenetreArretRea'])

# == 2. Time period assignment for each row ==
def assign_period(dt):
   hour = dt.hour
   day = dt.dayofweek
   if day == 5:
       return "Saturdays"
   if day == 6:
       return "Sundays"
   if 7 <= hour < 9:
       return "AM peak"
   elif 9 <= hour < 16:
       return "Day period"
   elif 16 <= hour < 19:
```

```
        return "PM peak"
    elif 19 <= hour < 23:
        return "Evening"
    else:
        return "Other"
df['period'] = df['DTEntreeFenetreArretRea'].apply(assign_period)
WILL MODIFY TO USE DTDepartTheo
# == 3. Filter for one segment if desired ==
segment_id = "74221075"
df_seg = df[df["SegmentID"] == segment_id].copy()

# == 4. Compute dwell time (for this segment) ==
df_seg["dwell_time"] = (df_seg["DTSortieFenetreArretRea"] -
df_seg["DTEntreeFenetreArretRea"]).dt.total_seconds()

# == 5. Large dwell time ==
df_seg["large_dwell_time"] = df_seg["dwell_time"] > alpha1

# == 6. Large variation in dwell time per period (requires at least 2 observations) ==
period_dwell_var = df_seg.groupby('period')["dwell_time"].agg(lambda x: np.percentile(x,85) -
np.percentile(x,15) if len(x.dropna())>=2 else 0)
df_seg["large_var_in_dwell"] = df_seg["period"].map(lambda p: period_dwell_var[p] > alpha2)

# == 7. Early departure ==
df_seg["early_departure"] = df_seg["EcartDepar"] < alpha3

# == 8. Late departure ==
df_seg["late_departure"] = df_seg["EcartDepar"] > alpha4

# == 9. Large variation in departure per period ==
period_dep_var = df_seg.groupby('period')["EcartDepar"].agg(lambda x: np.percentile(x,85) - np.percentile(x,15)
if len(x.dropna())>=2 else 0)
df_seg["large_var_in_departure"] = df_seg["period"].map(lambda p: period_dep_var[p] > alpha5)

# == 10. Punctuality change compared to previous stop ==
df_seg = df_seg.sort_values(["DTEntreeFenetreArretRea"])
df_seg["EcartDepar_prev"] = df_seg["EcartDepar"].shift(1)
df_seg["punct_change_prev"] = (df_seg["EcartDepar"] - df_seg["EcartDepar_prev"]).abs() > alpha6

# == 11. Low speed ==
df_seg["low_speed"] = df_seg["segment_speed"] < alpha7

# == 12. Large travel time compared to free flow ==

# First, compute free flow time (15th percentile of segment travel time for Sundays, *from whole dataset*)
sundays = df[df['period'] == 'Sundays']
freeflow_15th = sundays.groupby('SegmentID')['segment_travel_time'].quantile(0.15)
df_seg["freeflow_15th"] = df_seg["SegmentID"].map(freeflow_15th)
df_seg["large_travel_time_vs_freeflow"] = (df_seg["segment_travel_time"] - df_seg["freeflow_15th"]) > alpha8

# == 13. At least one criterion ==
criteria_cols = [
    "large_dwell_time", "large_var_in_dwell", "early_departure", "late_departure",
    "large_var_in_departure", "punct_change_prev", "low_speed", "large_travel_time_vs_freeflow"
]
df_seg["any_criterion"] = df_seg[criteria_cols].any(axis=1)

# == 14. Count for table: count # of TRUEs per period and criterion ==
summary = df_seg.groupby("period")[criteria_cols + ["any_criterion"]].sum().astype(int)
summary = summary.reset_index()

print(summary)
# Optionally save:
summary.to_csv("/Users/benfall/Desktop/segment74221075_bottleneck_summary.csv", index=False)
```

More or less complete code (minor rounding differences due to python limitations):

```
import re
import pandas as pd
import numpy as np

def pad4(val):
    try:
        return str(int(val)).zfill(4)
```

```
    except:
        return "null"

def clean_columns(df):
    # Strip whitespace and remove BOM characters from columns
    df.columns = df.columns.str.strip()
    df.columns = df.columns.str.replace('\ufeff', '', regex=False)
    return df

def adjust_group(g, to_remove, rank_col='RangArretAs'):
    sens = str(g['C_SensApp'].iloc[0])
    keep_stop = to_remove.get(sens)
    if keep_stop is None:
        g = g.sort_values(rank_col).reset_index(drop=True)
        g[rank_col] = range(1, len(g) + 1)
        return g

    g = g.sort_values(rank_col).reset_index(drop=True)
    drop_positions = g.index[g['IdArret'] == keep_stop].tolist()

    for pos in drop_positions:
        if pos == len(g) - 1:
            continue
        prev_pos = pos - 1 if pos - 1 >= 0 else None
        next_pos = pos + 1

        if next_pos < len(g):
            # Update IdArretPreceden for next row
            if prev_pos is not None:
                g.at[next_pos, 'IdArretPreceden'] = g.at[prev_pos, 'IdArret']
            else:
                g.at[next_pos, 'IdArretPreceden'] = np.nan

            # Handle DistanceInterArre safely as float and round
            prev_distance = float(g.at[pos, 'DistanceInterArre']) if pd.notna(g.at[pos, 'DistanceInterArre'])
else 0.0
            next_distance = float(g.at[next_pos, 'DistanceInterArre']) if pd.notna(g.at[next_pos,
'DistanceInterArre']) else 0.0
            g.at[next_pos, 'DistanceInterArre'] = round(next_distance + prev_distance, 2)

    # Drop the stop(s)
    g = g[g['IdArret'] != keep_stop].copy()
    g = g.sort_values(rank_col).reset_index(drop=True)
    g[rank_col] = range(1, len(g) + 1)
    return g

def compute_distance(row):
    if row['RangArretAs'] == 1:
        return np.nan
    seg = row['Segment_XXXXYYYY']
    d = float(row['DistanceInterArre']) if pd.notna(row['DistanceInterArre']) else np.nan
    if pd.isna(d):
        return np.nan
    if seg in ['23211310', '04220801', '08011082']:
        result = d - 95
    elif seg in ['13621077', '13082321']:
        result = d - 45
    else:
        result = d - 70
    return round(result, 2)

def main():
    # Step 1: Read fixed-width data
    fwf_path = '/Users/benfall/Desktop/arrets_ben.csv'
    with open(fwf_path, encoding='utf-8') as f:
        header_line = f.readline().rstrip('\n')
        dash_line = f.readline().rstrip('\n')

    colspecs = [(m.start(), m.end()) for m in re.finditer(r'-+', dash_line)]
    raw_names = [header_line[start:end].strip() for start, end in colspecs]

    seen = {}
    names = []
    for nm in raw_names:
        if nm in seen:
            seen[nm] += 1
```

```
            names.append(f"{nm}_{seen[nm]}")
        else:
            seen[nm] = 0
            names.append(nm)

    df = pd.read_fwf(fwf_path, colspecs=colspecs, names=names,
                     skiprows=[0, 1], na_values=['NULL'], dtype=str)
    df = clean_columns(df)

    # Canonical column names, adapt if necessary
    course_col = 'IdCourse'
    rank_col = 'RangArretAs'
    type_col = 'C_TypeAppl'

    df[course_col] = pd.to_numeric(df[course_col], errors='coerce')
    df[rank_col] = pd.to_numeric(df[rank_col], errors='coerce')
    df[type_col] = pd.to_numeric(df[type_col], errors='coerce')

    df0 = df[df[type_col] == 0].copy()

    df0 = df0.sort_values([course_col, rank_col]).reset_index(drop=True)
    df0[rank_col] = df0.groupby(course_col).cumcount() + 1

    # Step 2: Flag filtering
    flag_cols = ["FlagDeviatio", "FlagDelocalisatio", "FlagRecalag", "FlagRegulatio"]
    mask_any_y = df0[flag_cols].eq('Y').any(axis=1)
    mask_all_n = df0[flag_cols].eq('N').all(axis=1)

    df_any_y = df0[mask_any_y].reset_index(drop=True)
    df_all_n = df0[mask_all_n].reset_index(drop=True)

    df_any_y.to_csv('/Users/benfall/Desktop/Data Files/with_flag_y.csv', index=False, encoding='utf-8')
    df_all_n.to_csv('/Users/benfall/Desktop/Data Files/with_all_flags_n.csv', index=False, encoding='utf-8')

    # Step 3: Reload and reprocess all_flags_n subset
    df_all_n = pd.read_csv('/Users/benfall/Desktop/Data Files/with_all_flags_n.csv', encoding='utf-8',
low_memory=False)
    df_all_n = clean_columns(df_all_n)

    df_all_n[course_col] = pd.to_numeric(df_all_n[course_col], errors='coerce')
    df_all_n[rank_col] = pd.to_numeric(df_all_n[rank_col], errors='coerce')
    df_all_n[type_col] = pd.to_numeric(df_all_n[type_col], errors='coerce')

    df0_clean = df_all_n[df_all_n[type_col] == 0].copy()
    df0_clean = df0_clean.sort_values([course_col, rank_col]).reset_index(drop=True)
    df0_clean[rank_col] = df0_clean.groupby(course_col).cumcount() + 1

    out_sorted_path = '/Users/benfall/Desktop/Data Files/all_flags_n_type0_sorted.csv'
    df0_clean.to_csv(out_sorted_path, index=False, encoding='utf-8')

    # Step 4: Keep specific columns
    df_check = pd.read_csv(out_sorted_path, encoding='utf-8')
    df_check = clean_columns(df_check)

    columns_to_keep = [
        'IdCourse', 'IdArret', 'RangArretAs', 'EcartDepar', 'IdArretPreceden',
        'DistanceInterArre', 'DTEntreeFenetreArretRea', 'DTSortieFenetreArretRea',
        'DTDepartTheo', 'C_Lign', 'C_SensApp'
    ]
    available_cols = [col for col in columns_to_keep if col in df_check.columns]

    special_df = df_check[available_cols].copy()
    special_out_path = '/Users/benfall/Desktop/Data Files/all_flags_n_type0_sorted_special.csv'
    special_df.to_csv(special_out_path, index=False, encoding='utf-8')

    # Step 5: Remove designated stops on line 80
    df_special = pd.read_csv(special_out_path, encoding='utf-8')
    df_special = clean_columns(df_special)

    for col in ['C_Lign', 'RangArretAs', 'IdArret', 'DistanceInterArre', 'IdArretPreceden']:
        if col in df_special.columns:
            df_special[col] = pd.to_numeric(df_special[col], errors='coerce')

    to_remove = {'R': 2542, 'A': 2541}

    def process_df_stop_removal(df):
```

```
        processed_groups = []
        for course_id, group in df.groupby('IdCourse', group_keys=False):
            mask = (group['C_Lign'] == 80) & (group['C_SensApp'].isin(to_remove.keys()))
            if mask.any():
                new_group = adjust_group(group, to_remove, rank_col)
                processed_groups.append(new_group)
            else:
                group = group.sort_values(rank_col).reset_index(drop=True)
                group[rank_col] = range(1, len(group) + 1)
                processed_groups.append(group)

        df_out = pd.concat(processed_groups, ignore_index=True)
        df_out = df_out.sort_values([course_col, rank_col]).reset_index(drop=True)
        df_out[rank_col] = df_out.groupby(course_col).cumcount() + 1
        return df_out

    df_fixed = process_df_stop_removal(df_special)

    fixed_out_path = '/Users/benfall/Desktop/Data Files/all_flags_n_type0_sorted_special_nostop.csv'
    df_fixed.to_csv(fixed_out_path, index=False, encoding='utf-8')

    # Step 6: Add segment labels and compute times, distances, speeds
    df_final = pd.read_csv(fixed_out_path, encoding='utf-8')
    df_final = clean_columns(df_final)

    df_final['DTEntreeFenetreArretRea'] = pd.to_datetime(df_final['DTEntreeFenetreArretRea'], errors='coerce')
    df_final['DTSortieFenetreArretRea'] = pd.to_datetime(df_final['DTSortieFenetreArretRea'], errors='coerce')

    df_final['Segment_XXXXYYYY'] = df_final['IdArretPreceden'].apply(pad4) + df_final['IdArret'].apply(pad4)

    df_final['dwell_time'] = (df_final['DTSortieFenetreArretRea'] -
df_final['DTEntreeFenetreArretRea']).dt.total_seconds()
    df_final.loc[df_final[rank_col] == 1, 'dwell_time'] = np.nan

    df_final = df_final.sort_values([course_col, rank_col]).reset_index(drop=True)
    df_final['segment_travel_time'] = (df_final['DTEntreeFenetreArretRea'] -

df_final.groupby(course_col)['DTSortieFenetreArretRea'].shift(1)).dt.total_seconds()

    df_final['DistanceInterArre'] = pd.to_numeric(df_final['DistanceInterArre'], errors='coerce').round(2)
    df_final['distance'] = df_final.apply(compute_distance, axis=1)

    df_final['segment_speed'] = (df_final['distance'] / df_final['segment_travel_time']) * 3.6

    df_final['distance'] = pd.to_numeric(df_final['distance'], errors='coerce').round(2)
    df_final['segment_speed'] = pd.to_numeric(df_final['segment_speed'], errors='coerce').round(2)

    out_file_final = '/Users/benfall/Desktop/Data
Files/all_flags_n_type0_sorted_special_WITH_SEG_AND_TIMES_DIST_SPEED.csv'
    df_final.to_csv(out_file_final, index=False, encoding='utf-8')

    print("✅ Finished processing. Output saved to:")
    print(out_file_final)
    print("\nSample of key columns:")
    print(df_final[[course_col, rank_col, 'Segment_XXXXYYYY', 'dwell_time', 'segment_travel_time', 'distance',
'segment_speed']].head(12))

if __name__ == "__main__":
    main()
```

```
import pandas as pd

# Load OG data
file_path = '/Users/benfall/Desktop/Data Files/OG.csv'  # Replace with your actual file path
df = pd.read_csv(file_path, encoding='utf-8', low_memory=False)

# Clean columns from BOM and whitespace
df.columns = df.columns.str.strip().str.replace('\ufeff', '', regex=False)

# Exclude first stops where segment_speed is generally NaN or invalid
df_valid = df[df['RangArretAs'] != 1].copy()

# Convert segment_speed to numeric to avoid issues
df_valid['segment_speed'] = pd.to_numeric(df_valid['segment_speed'], errors='coerce')

# Your segment groups
```

```
all_segments = [
   '74221075', '10751361', '13610117', '01171296', '12960374', '03740042', '00425315', '53150056',
   '00560130', '01300071', '00711127', '11271039', '10391063', '10630086', '00863349', '33490422',
   '04220801', '08011082', '10821238', '12381441', '14410155', '01550091', '00910077', '00770134',
   '01340702', '07023002', '30020924', '09240686', '06860806', '08060260'
]

segments_line18_R = [
   '02610807', '08070687', '06870925', '09253003', '30030703', '07030137', '01370078', '00780092',
   '00920156', '01561440', '14401237', '12371081', '10810800', '08000427', '04270356', '03560087',
   '00871064', '10641041', '10411128', '11280072', '00720131', '01310057', '00575316', '53160043',
   '00430375', '03751297', '12970118', '01181362', '13621077', '10777423'
]

segments_line80_A = [
   '23211310', '13101022', '10220720', '07207166', '71660522', '05221298', '12980122', '01220335',
   '03353303', '33031037', '10371414', '14140052', '00520630', '06302117', '21170782', '07823301',
   '33010982', '09826495', '64956497', '64972521', '25217444'
]

segments_line80_R = [
   '74442522', '25226498', '64986496', '64960983', '09833302', '33020780', '07800353', '03530631',
   '06310053', '00531415', '14151038', '10383304', '33040334', '03340121', '01211301', '13010523',
   '05237165', '71650721', '07211023', '10231308', '13082321'
]

# Function to calculate mean segment speed for given segments
def mean_segment_speed(df, segments):
   subset = df[df['Segment_XXXXYYYY'].isin(segments)]
   means = subset.groupby('Segment_XXXXYYYY')['segment_speed'].mean().sort_index()
   return means

# Calculate means for each segment group
mean_all_segments = mean_segment_speed(df_valid, all_segments)
mean_18_R = mean_segment_speed(df_valid, segments_line18_R)
mean_80_A = mean_segment_speed(df_valid, segments_line80_A)
mean_80_R = mean_segment_speed(df_valid, segments_line80_R)

print("Mean segment speeds (all segments):")
print(mean_all_segments)

print("\nMean segment speeds (line 18 direction R):")
print(mean_18_R)

print("\nMean segment speeds (line 80 direction A):")
print(mean_80_A)

print("\nMean segment speeds (line 80 direction R):")
print(mean_80_R)
```

```
import pandas as pd
import matplotlib.pyplot as plt

# ---- Load the data ----
data_path = '/Users/benfall/Desktop/Data Files/all_flags_n_type0_sorted.csv'
df = pd.read_csv(data_path, encoding='utf-8', low_memory=False)
df.columns = df.columns.str.strip().str.replace('\ufeff','', regex=False)
df['DTDepartTheo'] = pd.to_datetime(df['DTDepartTheo'], errors='coerce')

# ---- Assign time period ----
def assign_period(dt):
   if pd.isnull(dt): return 'Unknown'
   dow = dt.dayofweek
   hr = dt.hour
   if dow == 5: return 'Saturdays'
   elif dow == 6: return 'Sundays'
   elif 7 <= hr < 9: return 'AM peak'
   elif 9 <= hr < 16: return 'Day period'
   elif 16 <= hr < 18: return 'PM peak'
   elif 18 <= hr < 24: return 'Evening'
   else: return 'Other'
df['TimePeriod'] = df['DTDepartTheo'].apply(assign_period)
for col in ['NbMontees', 'NbDescente']:
   df[col] = pd.to_numeric(df[col], errors='coerce').fillna(0).astype(int)
```

```
# ---- EXAMPLES: Station orders and mappings ----
# (Set these for each line/direction)
line18_A_order =
[7422,1075,1361,117,1296,374,42,5315,56,130,71,1127,1039,1063,86,3349,422,801,1082,1238,1441,155,91,77,134,702,
3002,924,686,806,260]
stations_18_A = {7422:'Grand-Lancy, Palettes', 1075:'Grand-Lancy, Pontets', 1361:'Plan-les-Ouates,
Tréfle-Blanc',
117:'Lancy-Bachet, gare', 1296:'Grand-Lancy, De-Staél', 374:'Carouge, Rondeau', 42:'Carouge, Ancienne',
5315:'Carouge, Marché', 56:'Carouge, Armes', 130:'Genéve, Blanche', 71:'Genéve, Augustins',
1127:"Genéve, Pont-d'Arve", 1039:'Genéve, Plainpalais', 1063:'Genéve, Place de Neuve',
86:'Genéve, Bel-Air', 3349:'Genéve, Coutance', 422:'Genéve, gare Cornavin', 801:'Genéve, Lyon',
1082:'Genéve, Poterie', 1238:'Genéve, Servette', 1441:'Genéve, Vieusseux', 155:'Vernier, Bouchet', 91:'Vernier,
Balexert',
77:'Vernier, Avanchets-Etang', 134:'Vernier, Blandonnet', 702:'Meyrin, Jardin-Alpin-Vivarium',
3002:'Meyrin, Bois-du-Lan', 924:'Meyrin, village', 686:'Meyrin, Hépital de La Tour', 806:'Meyrin, Maisonnex',
260:'Meyrin, CERN'}

# For direction R, reverse the order and use the correct mapping!
line18_R_order = [261, 807, 686, 924, 3003, 703, 137, 78, 92, 156, 1440, 1237, 1081, 800, 427, 356, 87, 1064,
1041, 1128, 72, 131, 57, 5316, 43, 375, 1297, 118, 1362, 1077, 7423]
stations_18_R = {
   7423:'Grand-Lancy, Palettes', 1077:'Grand-Lancy, Pontets', 1362:'Plan-les-Ouates, Tréfle-Blanc',
118:'Lancy-Bachet, gare',
   1297:'Grand-Lancy, De-Staél', 375:'Carouge, Rondeau', 43:'Carouge, Ancienne', 5316:'Carouge, Marché',
57:'Carouge, Armes',
   131:'Genéve, Blanche', 72:'Genéve, Augustins', 1128:"Genéve, Pont-d'Arve", 1041:'Genéve, Plainpalais',
1064:'Genéve, Place de Neuve',
   87:'Genéve, Bel-Air', 356:'Genéve, Coutance', 427:'Genéve, gare Cornavin', 800:'Genéve, Lyon',
   1081:'Genéve, Poterie', 1237:'Genéve, Servette', 1440:'Genéve, Vieusseux', 156:'Vernier, Bouchet',
92:'Vernier, Balexert',
   78:'Vernier, Avanchets-Etang', 137:'Vernier, Blandonnet', 703:'Meyrin, Jardin-Alpin-Vivarium', 3003:'Meyrin,
Bois-du-Lan',
   925:'Meyrin, village', 687:'Meyrin, Hépital de La Tour', 807:'Meyrin, Maisonnex', 261:'Meyrin, CERN'
}

# ---- General plotting function ----
def plot_cumulative_board_alight(
   df, line_number, direction, stations_order, station_map, time_period=None, title=None
):
   sub = df[
       (df['C_Lign'] == line_number) &
       (df['C_SensApp'] == direction) &
       (df['IdArret'].isin(stations_order))
   ].copy()
   if time_period is not None:
       sub = sub[sub['TimePeriod'] == time_period].copy()
   # Compute mean number on/off per station (per run)
   summary = sub.groupby('IdArret')[['NbMontees', 'NbDescente']].mean().reindex(stations_order).fillna(0)
   summary['StationName'] = [station_map.get(s, str(s)) for s in summary.index]
   summary['Cum_Boarding'] = summary['NbMontees'].cumsum()
   summary['Cum_Alighting'] = summary['NbDescente'].cumsum()
   plt.figure(figsize=(15,8))
   plt.plot(summary['StationName'], summary['Cum_Boarding'], label='Cumulative Avg Boarding', marker='o',
color='royalblue')
   plt.plot(summary['StationName'], summary['Cum_Alighting'], label='Cumulative Avg Alighting', marker='o',
color='orange')
   plt.xticks(rotation=90)
   plt.xlabel('Station (Ordered)')
   plt.ylabel('Cumulative Average Number of People')
   per_str = f" ({time_period})" if time_period else ' (All periods)'
   plt.title(title or f"Cumulative Avg Boarding and Alighting - Line {line_number} {direction}{per_str}")
   plt.legend()
   plt.grid(True)
   plt.tight_layout()
   plt.show()

# --- Example usage ---
# Line 18, Direction A, AM peak:
plot_cumulative_board_alight(df, 18, 'A', line18_A_order, stations_18_A, time_period="AM peak", title="Line 18
Direction A - AM peak")

# Line 18, Direction R, PM peak:
plot_cumulative_board_alight(df, 18, 'R', line18_R_order, stations_18_R, time_period="PM peak", title="Line 18
Direction R - PM peak")

# Just repeat with current order and mapping for any other line/direction/time_period!
```

```
# For Line 80 and others, insert corresponding station order/mapping and rerun.
```

Appendix 3:

=== dwell_time by IdArret ====

IdArret=42:
Before: min=9.000 max=1317.000
1% cutoff=12.000 99% cutoff=71.000
After: min=12.000 max=71.000

IdArret=43:
Before: min=2.000 max=530.000
1% cutoff=14.000 99% cutoff=72.950
After: min=14.000 max=72.000

IdArret=52:
Before: min=-8.000 max=545.000
1% cutoff=6.000 99% cutoff=61.000
After: min=6.000 max=61.000

IdArret=53:
Before: min=5.000 max=305.000
1% cutoff=18.000 99% cutoff=125.000
After: min=18.000 max=125.000

IdArret=56:
Before: min=12.000 max=653.000
1% cutoff=25.000 99% cutoff=88.000
After: min=25.000 max=88.000

IdArret=57:
Before: min=16.000 max=842.000
1% cutoff=29.000 99% cutoff=73.000
After: min=29.000 max=73.000

IdArret=71:
Before: min=9.000 max=724.000
1% cutoff=31.000 99% cutoff=81.300
After: min=31.000 max=81.000

IdArret=72:
Before: min=5.000 max=641.000
1% cutoff=11.000 99% cutoff=97.000
After: min=11.000 max=97.000

IdArret=77:
Before: min=4.000 max=897.000
1% cutoff=11.000 99% cutoff=61.000
After: min=11.000 max=61.000

IdArret=78:
Before: min=9.000 max=422.000
1% cutoff=23.000 99% cutoff=77.630
After: min=23.000 max=77.000

IdArret=86:
Before: min=10.000 max=3452.000
1% cutoff=37.000 99% cutoff=102.000
After: min=37.000 max=102.000

IdArret=87:
Before: min=8.000 max=324.000
1% cutoff=39.000 99% cutoff=109.000
After: min=39.000 max=109.000

IdArret=91:
Before: min=7.000 max=525.000
1% cutoff=12.000 99% cutoff=70.000
After: min=12.000 max=70.000

IdArret=92:
Before: min=8.000 max=1389.000
1% cutoff=23.000 99% cutoff=78.000
After: min=23.000 max=78.000

IdArret=117:
Before: min=5.000 max=816.000
1% cutoff=9.000 99% cutoff=79.000
After: min=9.000 max=79.000

IdArret=118:
Before: min=5.000 max=4203.000
1% cutoff=11.000 99% cutoff=82.000
After: min=11.000 max=82.000

IdArret=121:
Before: min=4.000 max=735.000
1% cutoff=33.000 99% cutoff=205.000
After: min=33.000 max=205.000

IdArret=122:
Before: min=2.000 max=370.000
1% cutoff=33.000 99% cutoff=156.980
After: min=33.000 max=156.000

IdArret=124:
Before: min=11.000 max=168.000
1% cutoff=12.570 99% cutoff=166.430
After: (no data left after trimming)

IdArret=125:
Before: min=6.000 max=47.000
1% cutoff=6.430 99% cutoff=45.280
After: min=7.000 max=43.000

IdArret=130:
Before: min=8.000 max=1696.000
1% cutoff=15.000 99% cutoff=63.000
After: min=15.000 max=63.000

IdArret=131:
Before: min=10.000 max=298.000
1% cutoff=26.000 99% cutoff=60.000
After: min=26.000 max=60.000

IdArret=134:
Before: min=2.000 max=958.000
1% cutoff=7.000 99% cutoff=130.000
After: min=7.000 max=130.000

IdArret=137:
Before: min=9.000 max=708.000
1% cutoff=30.000 99% cutoff=142.000
After: min=30.000 max=142.000

IdArret=155:
Before: min=6.000 max=1707.000
1% cutoff=24.000 99% cutoff=95.590
After: min=24.000 max=95.000

IdArret=156:
Before: min=3.000 max=2423.000
1% cutoff=27.000 99% cutoff=105.000
After: min=27.000 max=105.000

IdArret=260 (end station):
Before: min=-7.000 max=1975.000
1% cutoff=6.000 99% cutoff=44.000
After: min=6.000 max=44.000

IdArret=261 (start station)
No dwell times as it is the first stop each time

IdArret=314:
Before: min=11.000 max=132.000
1% cutoff=15.360 99% cutoff=73.820
After: min=17.000 max=73.000

IdArret=315:

Before: min=25.000 max=175.000
1% cutoff=31.000 99% cutoff=97.420
After: min=31.000 max=96.000

IdArret=334:
Before: min=7.000 max=349.000
1% cutoff=11.000 99% cutoff=123.000
After: min=11.000 max=123.000

IdArret=335:
Before: min=8.000 max=359.000
1% cutoff=10.000 99% cutoff=75.000
After: min=10.000 max=75.000

IdArret=353:
Before: min=0.000 max=1475.000
1% cutoff=5.000 99% cutoff=94.000
After: min=5.000 max=94.000

IdArret=356:
Before: min=12.000 max=726.000
1% cutoff=26.000 99% cutoff=78.000
After: min=26.000 max=78.000

IdArret=374:
Before: min=8.000 max=4708.000
1% cutoff=35.000 99% cutoff=108.740
After: min=35.000 max=108.000

IdArret=375:
Before: min=3.000 max=350.000
1% cutoff=17.000 99% cutoff=85.000
After: min=17.000 max=85.000

IdArret=422:
Before: min=11.000 max=658.000
1% cutoff=42.000 99% cutoff=142.000
After: min=42.000 max=142.000

IdArret=427:
Before: min=8.000 max=644.000
1% cutoff=45.000 99% cutoff=158.000
After: min=45.000 max=158.000

IdArret=522:
Before: min=5.000 max=1476.000
1% cutoff=42.000 99% cutoff=192.000
After: min=42.000 max=192.000

IdArret=523:
Before: min=4.000 max=730.000
1% cutoff=31.000 99% cutoff=228.000
After: min=31.000 max=228.000

IdArret=630:
Before: min=3.000 max=371.000
1% cutoff=6.000 99% cutoff=93.000
After: min=6.000 max=93.000

IdArret=631:
Before: min=5.000 max=574.000
1% cutoff=8.000 99% cutoff=117.000
After: min=8.000 max=117.000

IdArret=638:
Before: min=8.000 max=216.000
1% cutoff=11.440 99% cutoff=110.000
After: min=12.000 max=110.000

IdArret=686:
Before: min=5.000 max=556.000
1% cutoff=12.000 99% cutoff=93.000
After: min=12.000 max=93.000

IdArret=687:

Before: min=1.000 max=525.000
1% cutoff=12.000 99% cutoff=108.000
After: min=12.000 max=108.000

IdArret=699:
Before: min=19.000 max=563.000
1% cutoff=38.000 99% cutoff=216.540
After: min=38.000 max=175.000

IdArret=702:
Before: min=4.000 max=483.000
1% cutoff=32.000 99% cutoff=87.000
After: min=32.000 max=87.000

IdArret=703:
Before: min=4.000 max=424.000
1% cutoff=9.000 99% cutoff=63.000
After: min=9.000 max=63.000

IdArret=720:
Before: min=9.000 max=410.000
1% cutoff=34.000 99% cutoff=172.000
After: min=34.000 max=172.000

IdArret=721:
Before: min=8.000 max=545.000
1% cutoff=35.000 99% cutoff=181.000
After: min=35.000 max=181.000

IdArret=780:
Before: min=4.000 max=437.000
1% cutoff=6.000 99% cutoff=132.000
After: min=6.000 max=132.000

IdArret=782:
Before: min=4.000 max=337.000
1% cutoff=7.000 99% cutoff=75.000
After: min=7.000 max=75.000

IdArret=800:
Before: min=14.000 max=680.000
1% cutoff=33.000 99% cutoff=111.000
After: min=33.000 max=111.000

IdArret=801:
Before: min=8.000 max=454.000
1% cutoff=14.000 99% cutoff=72.000
After: min=14.000 max=72.000

IdArret=806:
Before: min=1.000 max=193.000
1% cutoff=7.000 99% cutoff=41.000
After: min=7.000 max=41.000

IdArret=807:
Before: min=4.000 max=3298.000
1% cutoff=6.000 99% cutoff=46.000
After: min=6.000 max=46.000

IdArret=924:
Before: min=6.000 max=1080.000
1% cutoff=14.000 99% cutoff=104.000
After: min=14.000 max=104.000

IdArret=925:
Before: min=3.000 max=479.000
1% cutoff=13.000 99% cutoff=70.000
After: min=13.000 max=70.000

IdArret=982:
Before: min=5.000 max=734.000
1% cutoff=7.000 99% cutoff=61.190
After: min=7.000 max=61.000

IdArret=983:

Before: min=4.000 max=550.000
1% cutoff=10.000 99% cutoff=124.000
After: min=10.000 max=124.000

IdArret=1022:
Before: min=8.000 max=427.000
1% cutoff=23.000 99% cutoff=184.000
After: min=23.000 max=184.000

IdArret=1023:
Before: min=8.000 max=518.000
1% cutoff=13.000 99% cutoff=70.000
After: min=13.000 max=70.000

IdArret=1037:
Before: min=5.000 max=230.000
1% cutoff=7.000 99% cutoff=60.000
After: min=7.000 max=60.000

IdArret=1038:
Before: min=6.000 max=482.000
1% cutoff=9.000 99% cutoff=90.130
After: min=9.000 max=90.000

IdArret=1039:
Before: min=2.000 max=534.000
1% cutoff=35.000 99% cutoff=128.000
After: min=35.000 max=128.000

IdArret=1041:
Before: min=4.000 max=421.000
1% cutoff=32.000 99% cutoff=115.000
After: min=32.000 max=115.000

IdArret=1042:
Before: min=13.000 max=213.000
1% cutoff=18.000 99% cutoff=141.000
After: min=18.000 max=141.000

IdArret=1045:
Before: min=19.000 max=503.000
1% cutoff=22.000 99% cutoff=203.260
After: min=22.000 max=202.000

IdArret=1063:
Before: min=11.000 max=743.000
1% cutoff=21.000 99% cutoff=78.000
After: min=21.000 max=78.000

IdArret=1064:
Before: min=8.000 max=359.000
1% cutoff=17.000 99% cutoff=71.000
After: min=17.000 max=71.000

IdArret=1075:
Before: min=2.000 max=488.000
1% cutoff=13.000 99% cutoff=65.000
After: min=13.000 max=65.000

IdArret=1077:
Before: min=8.000 max=578.000
1% cutoff=11.000 99% cutoff=56.000
After: min=11.000 max=56.000

IdArret=1081:
Before: min=8.000 max=1125.000
1% cutoff=21.000 99% cutoff=72.000
After: min=21.000 max=72.000

IdArret=1082:
Before: min=8.000 max=2170.000
1% cutoff=13.000 99% cutoff=79.000
After: min=13.000 max=79.000

IdArret=1127:

Before: min=4.000 max=539.000
1% cutoff=10.000 99% cutoff=84.000
After: min=10.000 max=84.000

IdArret=1128:
Before: min=7.000 max=721.000
1% cutoff=25.000 99% cutoff=71.000
After: min=25.000 max=71.000

IdArret=1237:
Before: min=7.000 max=2190.000
1% cutoff=31.000 99% cutoff=78.000
After: min=31.000 max=78.000

IdArret=1238:
Before: min=6.000 max=2503.000
1% cutoff=31.000 99% cutoff=127.000
After: min=31.000 max=127.000

IdArret=1296:
Before: min=2.000 max=1687.000
1% cutoff=31.000 99% cutoff=119.000
After: min=31.000 max=119.000

IdArret=1297:
Before: min=7.000 max=441.000
1% cutoff=13.000 99% cutoff=110.250
After: min=13.000 max=110.000

IdArret=1298:
Before: min=7.000 max=1201.000
1% cutoff=26.000 99% cutoff=158.900
After: min=26.000 max=158.000

IdArret=1301:
Before: min=7.000 max=1345.000
1% cutoff=12.000 99% cutoff=119.930
After: min=12.000 max=119.000

IdArret=1307:
Before: min=21.000 max=302.000
1% cutoff=28.000 99% cutoff=102.000
After: min=28.000 max=102.000

IdArret=1308:
Before: min=7.000 max=1108.000
1% cutoff=22.000 99% cutoff=105.000
After: min=22.000 max=105.000

IdArret=1309:
Also Stand but go from Cirque?!
Ignored
Before: min=12.000 max=130.000
1% cutoff=21.370 99% cutoff=99.000
After: min=22.000 max=99.000

IdArret=1310:
Before: min=11.000 max=458.000
1% cutoff=44.000 99% cutoff=230.000
After: min=44.000 max=230.000

IdArret=1311:
Before: min=17.000 max=294.000
1% cutoff=22.000 99% cutoff=179.880
After: min=22.000 max=177.000

IdArret=1361:
Before: min=7.000 max=366.000
1% cutoff=10.000 99% cutoff=56.000
After: min=10.000 max=56.000

IdArret=1362:
Before: min=9.000 max=1314.000
1% cutoff=11.000 99% cutoff=51.000
After: min=11.000 max=51.000

IdArret=1414:
Before: min=5.000 max=1436.000
1% cutoff=7.000 99% cutoff=60.000
After: min=7.000 max=60.000

IdArret=1415:
Before: min=5.000 max=524.000
1% cutoff=8.000 99% cutoff=78.000
After: min=8.000 max=78.000

IdArret=1440:
Before: min=7.000 max=1728.000
1% cutoff=13.000 99% cutoff=71.000
After: min=13.000 max=71.000

IdArret=1441:
Before: min=7.000 max=228.000
1% cutoff=14.000 99% cutoff=68.000
After: min=14.000 max=68.000

IdArret=1722:
Before: min=18.000 max=164.000
1% cutoff=18.920 99% cutoff=130.880
After: min=19.000 max=128.000

IdArret=1765:
Before: min=1.000 max=4.000
1% cutoff=1.050 99% cutoff=3.950
After: min=2.000 max=3.000

IdArret=2117:
Before: min=3.000 max=440.000
1% cutoff=5.000 99% cutoff=52.000
After: min=5.000 max=52.000

IdArret=2321 (first stop and end stop):
Before: min=-1.000 max=92.000
1% cutoff=7.000 99% cutoff=30.000
After: min=7.000 max=30.000

IdArret=2521:
Before: min=6.000 max=462.000
1% cutoff=17.000 99% cutoff=94.000
After: min=17.000 max=94.000

IdArret=2522:
Before: min=1.000 max=466.000
1% cutoff=19.000 99% cutoff=137.000
After: min=19.000 max=137.000

IdArret=3002:
Before: min=5.000 max=431.000
1% cutoff=11.000 99% cutoff=53.000
After: min=11.000 max=53.000

IdArret=3003:
Before: min=6.000 max=582.000
1% cutoff=11.000 99% cutoff=64.000
After: min=11.000 max=64.000

IdArret=3301:
Before: min=5.000 max=224.000
1% cutoff=10.000 99% cutoff=69.000
After: min=10.000 max=69.000

IdArret=3302:
Before: min=6.000 max=870.000
1% cutoff=18.000 99% cutoff=115.000
After: min=18.000 max=115.000

IdArret=3303:
Before: min=-14.000 max=346.000
1% cutoff=7.000 99% cutoff=81.650
After: min=7.000 max=81.000

IdArret=3304:
Before: min=6.000 max=750.000
1% cutoff=19.000 99% cutoff=161.000
After: min=19.000 max=161.000

IdArret=3349:
Before: min=2.000 max=950.000
1% cutoff=30.930 99% cutoff=143.000
After: min=31.000 max=143.000

IdArret=5315:
Before: min=5.000 max=297.000
1% cutoff=18.000 99% cutoff=78.000
After: min=18.000 max=78.000

IdArret=5316:
Before: min=10.000 max=402.000
1% cutoff=16.000 99% cutoff=73.000
After: min=16.000 max=73.000

IdArret=6495:
Before: min=-11.000 max=256.000
1% cutoff=6.000 99% cutoff=55.000
After: min=6.000 max=55.000

IdArret=6496:
Before: min=5.000 max=390.000
1% cutoff=7.000 99% cutoff=113.000
After: min=7.000 max=113.000

IdArret=6497:
Before: min=6.000 max=582.000
1% cutoff=12.000 99% cutoff=72.000
After: min=12.000 max=72.000

IdArret=6498:
Before: min=6.000 max=439.000
1% cutoff=18.000 99% cutoff=118.000
After: min=18.000 max=118.000

IdArret=6948:
Before: min=21.000 max=43.000
1% cutoff=21.080 99% cutoff=42.960
After: min=23.000 max=42.000

IdArret=7165:
Before: min=6.000 max=304.000
1% cutoff=10.000 99% cutoff=145.000
After: min=10.000 max=145.000

IdArret=7166:
Before: min=6.000 max=326.000
1% cutoff=14.450 99% cutoff=165.000
After: min=15.000 max=165.000

IdArret=7422 (first stop):
Before: min=-4.000 max=125.000
1% cutoff=10.000 99% cutoff=48.000
After: min=10.000 max=48.000
*Dwell times aren't calculated when it is in fact the first stop

IdArret=7423 (end stop):
Before: min=4.000 max=61.000
1% cutoff=10.000 99% cutoff=49.000
After: min=10.000 max=49.000

IdArret=7444 (first and last stop):
Before: min=2.000 max=100.000
1% cutoff=7.000 99% cutoff=33.000
After: min=7.000 max=33.000

IdArret=7686:
Before: min=8.000 max=410.000
1% cutoff=12.000 99% cutoff=60.550

After: min=12.000 max=60.000

IdArret=7687:
Before: min=9.000 max=404.000
1% cutoff=11.000 99% cutoff=58.000
After: min=11.000 max=58.000

IdArret=7688:
Before: min=11.000 max=215.000
1% cutoff=15.000 99% cutoff=67.400
After: min=15.000 max=67.000

IdArret=7689:
Before: min=11.000 max=302.000
1% cutoff=14.000 99% cutoff=52.000
After: min=14.000 max=52.000

IdArret=7690:
Before: min=10.000 max=661.000
1% cutoff=14.000 99% cutoff=70.460
After: min=14.000 max=70.000

IdArret=7691:
Before: min=3.000 max=127.000
1% cutoff=13.000 99% cutoff=71.000
After: min=13.000 max=71.000

IdArret=7693:
Before: min=5.000 max=390.000
1% cutoff=9.000 99% cutoff=46.510
After: min=9.000 max=46.000

==== segment_speed by Segment_XXXXYYYY ====

Segment_XXXXYYYY=00425315:
Before: min=0.743 max=38.743
1% cutoff=14.529 99% cutoff=26.013
After: min=14.529 max=26.013

Segment_XXXXYYYY=00430375:
Before: min=1.708 max=49.629
1% cutoff=6.715 99% cutoff=29.250
After: min=6.715 max=29.250

Segment_XXXXYYYY=00520630:
Before: min=-5.478 max=112.200
1% cutoff=7.545 99% cutoff=58.226
After: min=7.546 max=58.226

Segment_XXXXYYYY=00531415:
Before: min=-7.636 max=99.900
1% cutoff=15.128 99% cutoff=60.120
After: min=15.129 max=60.120

Segment_XXXXYYYY=00560130:
Before: min=0.609 max=26.900
1% cutoff=10.413 99% cutoff=22.009
After: min=10.413 max=22.009

Segment_XXXXYYYY=00575316:
Before: min=1.282 max=62.400
1% cutoff=9.200 99% cutoff=27.733
After: min=9.200 max=27.733

Segment_XXXXYYYY=00711127:
Before: min=0.754 max=871.200
1% cutoff=10.711 99% cutoff=39.764
After: min=10.711 max=39.764

Segment_XXXXYYYY=00720131:
Before: min=0.907 max=57.257
1% cutoff=10.840 99% cutoff=34.843
After: min=10.840 max=34.843

Segment_XXXXYYYY=00770134:

Before: min=4.578 max=146.640
1% cutoff=22.291 99% cutoff=54.900
After: min=22.291 max=54.900

Segment_XXXXYYYY=00780092:
Before: min=2.848 max=60.857
1% cutoff=11.368 99% cutoff=42.480
After: min=11.373 max=42.480

Segment_XXXXYYYY=00863349:
Before: min=0.436 max=42.218
1% cutoff=8.029 99% cutoff=26.949
After: min=8.031 max=26.949

Segment_XXXXYYYY=00871064:
Before: min=1.577 max=40.670
1% cutoff=8.695 99% cutoff=22.874
After: min=8.695 max=22.874

Segment_XXXXYYYY=00910077:
Before: min=1.173 max=81.840
1% cutoff=11.623 99% cutoff=44.229
After: min=11.649 max=44.229

Segment_XXXXYYYY=00920156:
Before: min=4.239 max=66.414
1% cutoff=13.783 99% cutoff=46.114
After: min=13.783 max=46.114

Segment_XXXXYYYY=01171296:
Before: min=1.198 max=44.673
1% cutoff=8.774 99% cutoff=26.537
After: min=8.781 max=26.537

Segment_XXXXYYYY=01181362:
Before: min=2.839 max=35.259
1% cutoff=11.798 99% cutoff=27.082
After: min=11.798 max=27.082

Segment_XXXXYYYY=01211301:
Before: min=0.819 max=83.057
1% cutoff=6.414 99% cutoff=48.715
After: min=6.432 max=48.715

Segment_XXXXYYYY=01220335:
Before: min=0.746 max=84.764
1% cutoff=9.333 99% cutoff=52.869
After: min=9.335 max=52.869

Segment_XXXXYYYY=01300071:
Before: min=-7.200 max=50.589
1% cutoff=9.552 99% cutoff=32.520
After: min=9.552 max=32.520

Segment_XXXXYYYY=01310057:
Before: min=2.557 max=27.635
1% cutoff=9.588 99% cutoff=20.720
After: min=9.588 max=20.720

Segment_XXXXYYYY=01340702:
Before: min=4.598 max=55.096
1% cutoff=24.346 99% cutoff=45.759
After: min=24.346 max=45.759

Segment_XXXXYYYY=01346948:
Before: min=9.171 max=17.439
1% cutoff=9.202 99% cutoff=17.256
After: min=10.206 max=11.360

Segment_XXXXYYYY=01370078:
Before: min=2.356 max=61.103
1% cutoff=24.281 99% cutoff=48.409
After: min=24.281 max=48.409

Segment_XXXXYYYY=01550091:

Before: min=2.575 max=61.763
1% cutoff=14.271 99% cutoff=44.000
After: min=14.271 max=44.000

Segment_XXXXYYYY=01561440:
Before: min=2.210 max=59.520
1% cutoff=27.471 99% cutoff=43.727
After: min=27.471 max=43.727

Segment_XXXXYYYY=02610807:
Before: min=0.301 max=153.771
1% cutoff=22.543 99% cutoff=54.338
After: min=22.555 max=54.327

Segment_XXXXYYYY=03141308:
Before: min=1.766 max=19.579
1% cutoff=1.853 99% cutoff=19.441
After: min=2.345 max=18.660

Segment_XXXXYYYY=03141309:
Before: min=0.496 max=30.388
1% cutoff=3.755 99% cutoff=27.274
After: min=3.774 max=27.095

Segment_XXXXYYYY=03151045:
Before: min=2.400 max=33.660
1% cutoff=3.584 99% cutoff=30.902
After: min=3.660 max=30.686

Segment_XXXXYYYY=03151722:
Before: min=2.326 max=21.240
1% cutoff=2.760 99% cutoff=19.158
After: min=2.770 max=19.093

Segment_XXXXYYYY=03340121:
Before: min=-2.965 max=90.600
1% cutoff=9.612 99% cutoff=45.847
After: min=9.615 max=45.847

Segment_XXXXYYYY=03353303:
Before: min=-405.000 max=95.100
1% cutoff=7.690 99% cutoff=44.400
After: min=7.704 max=44.400

Segment_XXXXYYYY=03530631:
Before: min=-178.200 max=522.000
1% cutoff=7.981 99% cutoff=76.114
After: min=7.981 max=*76.114*

Segment_XXXXYYYY=03560087:
Before: min=0.874 max=48.505
1% cutoff=12.575 99% cutoff=31.779
After: min=12.575 max=31.779

Segment_XXXXYYYY=03740042:
Before: min=0.392 max=56.400
1% cutoff=2.908 99% cutoff=26.585
After: min=*2.908* max=26.585

Segment_XXXXYYYY=03751297:
Before: min=3.764 max=43.200
1% cutoff=10.350 99% cutoff=37.291
After: min=10.350 max=37.291

Segment_XXXXYYYY=04220801:
Before: min=-6.840 max=34.071
1% cutoff=8.559 99% cutoff=20.371
After: min=8.559 max=20.371

Segment_XXXXYYYY=04270356:
Before: min=0.997 max=21.443
1% cutoff=4.311 99% cutoff=17.869
After: min=4.314 max=17.869

Segment_XXXXYYYY=04270699:

Before: min=1.002 max=22.255
1% cutoff=2.620 99% cutoff=20.425
After: min=2.635 max=20.325

Segment_XXXXYYYY=04300422:
Before: min=2.430 max=16.033
1% cutoff=3.907 99% cutoff=14.723
After: min=3.908 max=14.709

Segment_XXXXYYYY=04620118:
Before: min=9.220 max=9.220
1% cutoff=9.220 99% cutoff=9.220
After: min=9.220 max=9.220

Segment_XXXXYYYY=04621765:
Before: min=-29.400 max=-16.364
1% cutoff=-29.400 99% cutoff=-16.670
After: min=-29.400 max=-22.500

Segment_XXXXYYYY=05221298:
Before: min=2.123 max=103.636
1% cutoff=16.212 99% cutoff=60.631
After: min=16.237 max=60.621

Segment_XXXXYYYY=05237165:
Before: min=-1.826 max=60.676
1% cutoff=16.921 99% cutoff=47.989
After: min=16.927 max=47.983

Segment_XXXXYYYY=06302117:
Before: min=0.990 max=322.800
1% cutoff=14.971 99% cutoff=61.680
After: min=14.971 max=61.680

Segment_XXXXYYYY=06310053:
Before: min=2.265 max=130.320
1% cutoff=19.549 99% cutoff=76.659
After: min=19.551 max=76.659

Segment_XXXXYYYY=06380422:
Before: min=1.406 max=26.205
1% cutoff=3.143 99% cutoff=16.314
After: min=3.207 max=16.298

Segment_XXXXYYYY=06860806:
Before: min=2.537 max=106.094
1% cutoff=23.610 99% cutoff=52.251
After: min=23.610 max=52.251

Segment_XXXXYYYY=06870925:
Before: min=-5.143 max=114.120
1% cutoff=6.111 99% cutoff=40.469
After: min=6.122 max=40.469

Segment_XXXXYYYY=06991307:
Before: min=1.184 max=37.626
1% cutoff=2.999 99% cutoff=31.586
After: min=3.008 max=31.524

Segment_XXXXYYYY=07023002:
Before: min=-6.462 max=74.200
1% cutoff=23.464 99% cutoff=40.036
After: min=23.464 max=40.036

Segment_XXXXYYYY=07030137:
Before: min=2.845 max=53.200
1% cutoff=23.501 99% cutoff=43.896
After: min=23.501 max=43.870

Segment_XXXXYYYY=07036948:
Before: min=48.738 max=48.738
1% cutoff=48.738 99% cutoff=48.738
After: min=48.738 max=48.738

Segment_XXXXYYYY=07207166:

Before: min=-6.300 max=97.579
1% cutoff=14.574 99% cutoff=63.665
After: min=14.576 max=*63.643*

Segment_XXXXYYYY=07211023:
Before: min=-8.129 max=83.000
1% cutoff=18.000 99% cutoff=56.057
After: min=18.000 max=56.057

Segment_XXXXYYYY=07800353:
Before: min=2.344 max=84.240
1% cutoff=6.060 99% cutoff=56.749
After: min=6.061 max=56.747

Segment_XXXXYYYY=07823301:
Before: min=0.707 max=126.000
1% cutoff=6.550 99% cutoff=48.618
After: min=6.550 max=48.600

Segment_XXXXYYYY=07826495:
Before: min=3.980 max=6.445
1% cutoff=3.997 99% cutoff=6.413
After: min=4.814 max=4.814

Segment_XXXXYYYY=08000427:
Before: min=0.968 max=27.720
1% cutoff=6.252 99% cutoff=21.047
After: min=6.253 max=21.046

Segment_XXXXYYYY=08011082:
Before: min=0.533 max=43.329
1% cutoff=9.071 99% cutoff=29.631
After: min=9.071 max=29.631

Segment_XXXXYYYY=08060260:
Before: min=3.227 max=102.764
1% cutoff=23.082 99% cutoff=49.950
After: min=23.082 max=49.950

Segment_XXXXYYYY=08070687:
Before: min=-4.941 max=88.800
1% cutoff=23.514 99% cutoff=48.264
After: min=23.580 max=48.221

Segment_XXXXYYYY=09240686:
Before: min=0.707 max=46.523
1% cutoff=7.354 99% cutoff=35.576
After: min=7.355 max=35.576

Segment_XXXXYYYY=09253003:
Before: min=1.015 max=64.656
1% cutoff=14.880 99% cutoff=47.224
After: min=14.967 max=47.224

Segment_XXXXYYYY=09826495:
Before: min=1.910 max=102.150
1% cutoff=8.980 99% cutoff=45.600
After: min=8.980 max=45.600

Segment_XXXXYYYY=09833302:
Before: min=1.961 max=83.520
1% cutoff=9.165 99% cutoff=58.320
After: min=9.169 max=58.320

Segment_XXXXYYYY=10220720:
Before: min=-3.036 max=78.240
1% cutoff=7.579 99% cutoff=55.714
After: min=7.579 max=55.714

Segment_XXXXYYYY=10231308:
Before: min=-1.626 max=58.080
1% cutoff=9.360 99% cutoff=38.640
After: min=9.360 max=38.640

Segment_XXXXYYYY=10371414:

Before: min=0.646 max=583.200
1% cutoff=10.179 99% cutoff=48.900
After: min=10.179 max=48.900

Segment_XXXXYYYY=10383304:
Before: min=1.014 max=86.400
1% cutoff=3.960 99% cutoff=48.000
After: min=3.960 max=48.000

Segment_XXXXYYYY=10391063:
Before: min=1.034 max=39.024
1% cutoff=11.297 99% cutoff=24.570
After: min=11.297 max=24.570

Segment_XXXXYYYY=10411128:
Before: min=0.351 max=70.714
1% cutoff=10.494 99% cutoff=34.510
After: min=10.497 max=34.510

Segment_XXXXYYYY=10420314:
Before: min=1.268 max=28.224
1% cutoff=4.576 99% cutoff=23.115
After: min=4.751 max=22.413

Segment_XXXXYYYY=10451128:
Before: min=2.918 max=18.667
1% cutoff=3.019 99% cutoff=18.607
After: min=3.834 max=18.129

Segment_XXXXYYYY=10630086:
Before: min=0.809 max=25.920
1% cutoff=7.703 99% cutoff=20.765
After: min=7.703 max=20.765

Segment_XXXXYYYY=10640314:
Before: min=6.246 max=16.532
1% cutoff=6.314 99% cutoff=16.244
After: min=6.864 max=13.920

Segment_XXXXYYYY=10641041:
Before: min=1.429 max=33.269
1% cutoff=8.031 99% cutoff=25.768
After: min=8.033 max=25.768

Segment_XXXXYYYY=10751361:
Before: min=0.657 max=38.974
1% cutoff=13.977 99% cutoff=25.425
After: min=13.980 max=25.425

Segment_XXXXYYYY=10777422:
Before: min=1.956 max=35.319
1% cutoff=13.702 99% cutoff=26.564
After: min=13.725 max=26.550

Segment_XXXXYYYY=10777423:
Before: min=2.891 max=29.972
1% cutoff=10.103 99% cutoff=24.928
After: min=10.171 max=24.923

Segment_XXXXYYYY=10777691:
Before: min=4.178 max=40.566
1% cutoff=8.344 99% cutoff=25.406
After: min=8.400 max=25.393

Segment_XXXXYYYY=10810800:
Before: min=0.379 max=39.900
1% cutoff=9.836 99% cutoff=30.870
After: min=9.836 max=30.870

Segment_XXXXYYYY=10821238:
Before: min=-9.692 max=292.800
1% cutoff=9.754 99% cutoff=46.800
After: min=9.760 max=46.800

Segment_XXXXYYYY=11271039:

Before: min=0.477 max=354.000
1% cutoff=9.528 99% cutoff=34.843
After: min=9.529 max=34.843

Segment_XXXXYYYY=11271042:
Before: min=1.545 max=24.379
1% cutoff=4.256 99% cutoff=20.577
After: min=4.269 max=20.488

Segment_XXXXYYYY=11280072:
Before: min=0.296 max=78.545
1% cutoff=4.332 99% cutoff=39.436
After: min=4.335 max=39.436

Segment_XXXXYYYY=12371081:
Before: min=0.768 max=59.040
1% cutoff=12.181 99% cutoff=40.582
After: min=12.181 max=40.582

Segment_XXXXYYYY=12381441:
Before: min=0.928 max=78.975
1% cutoff=23.842 99% cutoff=50.535
After: min=23.842 max=*50.529*

Segment_XXXXYYYY=12960374:
Before: min=0.886 max=46.457
1% cutoff=13.448 99% cutoff=39.035
After: min=13.451 max=39.035

Segment_XXXXYYYY=12970118:
Before: min=1.035 max=63.840
1% cutoff=11.012 99% cutoff=29.925
After: min=11.048 max=29.925

Segment_XXXXYYYY=12970124:
Before: min=18.623 max=21.927
1% cutoff=18.656 99% cutoff=21.894
After: (no data left after trimming)

Segment_XXXXYYYY=12980122:
Before: min=0.425 max=213.943
1% cutoff=11.553 99% cutoff=74.158
After: min=11.553 max=*74.057*

Segment_XXXXYYYY=13010523:
Before: min=-2.769 max=79.855
1% cutoff=9.897 99% cutoff=44.628
After: min=9.897 max=44.628

Segment_XXXXYYYY=13070315:
Before: min=3.696 max=40.800
1% cutoff=10.753 99% cutoff=34.777
After: min=12.105 max=34.538

Segment_XXXXYYYY=13080086:
Before: min=4.425 max=19.756
1% cutoff=4.812 99% cutoff=19.373
After: min=7.005 max=17.200

Segment_XXXXYYYY=13082321:
Before: min=-123.600 max=168.171
1% cutoff=6.507 99% cutoff=41.523
After: min=6.512 max=41.400

Segment_XXXXYYYY=13090638:
Before: min=2.883 max=29.661
1% cutoff=6.315 99% cutoff=26.286
After: min=6.873 max=26.238

Segment_XXXXYYYY=13101022:
Before: min=2.660 max=79.615
1% cutoff=8.447 99% cutoff=42.026
After: min=8.454 max=42.026

Segment_XXXXYYYY=13110638:

Before: min=0.277 max=35.169
1% cutoff=6.204 99% cutoff=28.926
After: min=6.703 max=28.723

Segment_XXXXYYYY=13610117:
Before: min=3.538 max=36.947
1% cutoff=11.580 99% cutoff=25.056
After: min=11.593 max=25.056

Segment_XXXXYYYY=13610125:
Before: min=6.991 max=26.426
1% cutoff=7.223 99% cutoff=24.516
After: min=7.531 max=21.986

Segment_XXXXYYYY=13621077:
Before: min=0.951 max=48.695
1% cutoff=13.405 99% cutoff=28.368
After: min=13.409 max=28.364

Segment_XXXXYYYY=14140052:
Before: min=1.485 max=131.760
1% cutoff=16.532 99% cutoff=57.836
After: min=16.532 max=*57.789*

Segment_XXXXYYYY=14151038:
Before: min=0.465 max=inf
1% cutoff=6.556 99% cutoff=50.760
After: min=6.562 max=50.760

Segment_XXXXYYYY=14401237:
Before: min=0.368 max=50.954
1% cutoff=10.211 99% cutoff=39.050
After: min=10.214 max=38.965

Segment_XXXXYYYY=14410155:
Before: min=0.686 max=71.700
1% cutoff=13.147 99% cutoff=47.501
After: min=13.157 max=47.446

Segment_XXXXYYYY=17221128:
Before: min=0.912 max=40.114
1% cutoff=2.834 99% cutoff=38.362
After: min=14.640 max=27.600

Segment_XXXXYYYY=17650374:
Before: min=8.583 max=38.285
1% cutoff=8.683 99% cutoff=36.725
After: min=8.833 max=34.386

Segment_XXXXYYYY=21170782:
Before: min=-3.706 max=94.176
1% cutoff=8.214 99% cutoff=57.441
After: min=8.214 max=57.424

Segment_XXXXYYYY=23211310:
Before: min=-2.085 max=39.825
1% cutoff=8.321 99% cutoff=27.647
After: min=8.321 max=27.626

Segment_XXXXYYYY=25217444:
Before: min=1.566 max=41.125
1% cutoff=7.543 99% cutoff=28.634
After: min=7.546 max=28.631

Segment_XXXXYYYY=25220983:
Before: min=3.840 max=3.884
1% cutoff=3.840 99% cutoff=3.883
After: (no data left after trimming)

Segment_XXXXYYYY=25226498:
Before: min=1.057 max=131.600
1% cutoff=22.074 99% cutoff=59.580
After: min=22.075 max=59.580

Segment_XXXXYYYY=25420983:

Before: min=1.576 max=4.582
1% cutoff=1.659 99% cutoff=4.578
After: min=4.338 max=4.464

Segment_XXXXYYYY=30020924:
Before: min=1.218 max=113.914
1% cutoff=13.438 99% cutoff=48.327
After: min=13.440 max=48.327

Segment_XXXXYYYY=30030703:
Before: min=1.367 max=90.960
1% cutoff=13.897 99% cutoff=42.075
After: min=13.918 max=42.075

Segment_XXXXYYYY=33010982:
Before: min=2.590 max=92.400
1% cutoff=10.137 99% cutoff=52.457
After: min=10.139 max=52.457

Segment_XXXXYYYY=33016495:
Before: min=2.787 max=9.686
1% cutoff=2.840 99% cutoff=9.604
After: min=3.843 max=8.045

Segment_XXXXYYYY=33020780:
Before: min=-7.412 max=107.280
1% cutoff=20.100 99% cutoff=58.400
After: min=20.106 max=58.400

Segment_XXXXYYYY=33031037:
Before: min=-72.000 max=385.200
1% cutoff=10.108 99% cutoff=50.232
After: min=10.119 max=50.000

Segment_XXXXYYYY=33040334:
Before: min=0.833 max=123.429
1% cutoff=3.414 99% cutoff=50.612
After: min=3.416 max=50.612

Segment_XXXXYYYY=33490422:
Before: min=0.382 max=21.032
1% cutoff=3.937 99% cutoff=17.374
After: min=3.937 max=17.374

Segment_XXXXYYYY=53150056:
Before: min=1.996 max=45.450
1% cutoff=14.590 99% cutoff=25.033
After: min=14.592 max=25.033

Segment_XXXXYYYY=53160043:
Before: min=1.505 max=37.161
1% cutoff=12.130 99% cutoff=26.707
After: min=12.164 max=26.707

Segment_XXXXYYYY=64956497:
Before: min=4.194 max=392.400
1% cutoff=12.953 99% cutoff=65.700
After: min=12.960 max=65.700

Segment_XXXXYYYY=64960780:
Before: min=5.576 max=5.679
1% cutoff=5.577 99% cutoff=5.678
After: (no data left after trimming)

Segment_XXXXYYYY=64960983:
Before: min=1.442 max=210.000
1% cutoff=10.012 99% cutoff=40.500
After: min=10.012 max=40.500

Segment_XXXXYYYY=64963302:
Before: min=11.797 max=11.797
1% cutoff=11.797 99% cutoff=11.797
After: min=11.797 max=11.797

Segment_XXXXYYYY=64972521:

Before: min=3.911 max=72.327
1% cutoff=19.301 99% cutoff=43.103
After: min=19.301 max=43.103

Segment_XXXXYYYY=64980983:
Before: min=3.600 max=8.794
1% cutoff=3.725 99% cutoff=8.589
After: min=4.121 max=7.938

Segment_XXXXYYYY=64986496:
Before: min=2.161 max=198.600
1% cutoff=24.759 99% cutoff=66.400
After: min=24.759 max=66.400

Segment_XXXXYYYY=70621064:
Before: min=5.944 max=21.135
1% cutoff=6.202 99% cutoff=21.098
After: min=7.556 max=20.903

Segment_XXXXYYYY=70621311:
Before: min=2.220 max=26.760
1% cutoff=6.002 99% cutoff=19.499
After: min=6.272 max=19.317

Segment_XXXXYYYY=71650721:
Before: min=0.546 max=79.952
1% cutoff=6.884 99% cutoff=40.240
After: min=6.886 max=40.240

Segment_XXXXYYYY=71660522:
Before: min=-3.652 max=96.039
1% cutoff=14.740 99% cutoff=70.085
After: min=14.742 *max=70.085*

Segment_XXXXYYYY=74221075:
Before: min=11.668 max=41.923
1% cutoff=16.927 99% cutoff=32.913
After: min=16.938 max=32.850

Segment_XXXXYYYY=74231075:
Before: min=1.471 max=40.050
1% cutoff=16.758 99% cutoff=32.663
After: min=16.770 max=32.663

Segment_XXXXYYYY=74440983:
Before: min=1.769 max=1.769
1% cutoff=1.769 99% cutoff=1.769
After: min=1.769 max=1.769

Segment_XXXXYYYY=74442522:
Before: min=-1.156 max=60.565
1% cutoff=10.600 99% cutoff=32.691
After: min=10.604 max=32.686

Segment_XXXXYYYY=76867690:
Before: min=4.681 max=31.425
1% cutoff=17.853 99% cutoff=28.406
After: min=17.871 max=28.392

Segment_XXXXYYYY=76877689:
Before: min=5.488 max=50.054
1% cutoff=22.851 99% cutoff=42.003
After: min=22.869 max=41.979

Segment_XXXXYYYY=76887686:
Before: min=2.298 max=47.608
1% cutoff=20.220 99% cutoff=42.203
After: min=20.473 max=42.163

Segment_XXXXYYYY=76897693:
Before: min=9.189 max=36.442
1% cutoff=21.560 99% cutoff=32.341
After: min=21.560 max=32.279

Segment_XXXXYYYY=76901075:

Before: min=4.000 max=41.824
1% cutoff=13.757 99% cutoff=29.475
After: min=13.800 max=29.475

Segment_XXXXYYYY=76917687:
Before: min=0.537 max=31.800
1% cutoff=11.822 99% cutoff=24.502
After: min=11.833 max=24.480

Segment_XXXXYYYY=76927688:
Before: min=16.705 max=36.623
1% cutoff=21.023 99% cutoff=33.592
After: min=21.023 max=33.537

==== segment_travel_time by Segment_XXXXYYYY ====

Segment_XXXXYYYY=00425315:
Before: min=21.000 max=1071.000
1% cutoff=31.000 99% cutoff=56.000
After: min=31.000 max=56.000

Segment_XXXXYYYY=00430375:
Before: min=14.000 max=413.000
1% cutoff=24.000 99% cutoff=104.990
After: min=24.000 max=104.000

Segment_XXXXYYYY=00520630:
Before: min=12.000 max=889.000
1% cutoff=23.000 99% cutoff=177.000
After: min=23.000 max=177.000

Segment_XXXXYYYY=00531415:
Before: min=12.000 max=1265.000
1% cutoff=20.000 99% cutoff=79.000
After: min=20.000 max=79.000

Segment_XXXXYYYY=00560130:
Before: min=36.000 max=1596.000
1% cutoff=44.000 99% cutoff=93.000
After: min=44.000 max=93.000

Segment_XXXXYYYY=00575316:
Before: min=12.000 max=584.000
1% cutoff=27.000 99% cutoff=81.000
After: min=27.000 max=81.000

Segment_XXXXYYYY=00711127:
Before: min=1.000 max=1165.000
1% cutoff=22.000 99% cutoff=81.000
After: min=22.000 max=81.000

Segment_XXXXYYYY=00720131:
Before: min=20.000 max=1083.000
1% cutoff=28.000 99% cutoff=91.000
After: min=28.000 max=91.000

Segment_XXXXYYYY=00770134:
Before: min=15.000 max=482.000
1% cutoff=40.000 99% cutoff=99.000
After: min=40.000 max=99.000

Segment_XXXXYYYY=00780092:
Before: min=21.000 max=450.000
1% cutoff=30.000 99% cutoff=113.000
After: min=30.000 max=113.000

Segment_XXXXYYYY=00863349:
Before: min=22.000 max=2161.000
1% cutoff=35.000 99% cutoff=116.000
After: min=35.000 max=116.000

Segment_XXXXYYYY=00871064:
Before: min=37.000 max=943.000
1% cutoff=65.000 99% cutoff=171.000
After: min=65.000 max=171.000

Segment_XXXXYYYY=00910077:
Before: min=15.000 max=1040.000
1% cutoff=28.000 99% cutoff=106.000
After: min=28.000 max=106.000

Segment_XXXXYYYY=00920156:
Before: min=29.000 max=456.000
1% cutoff=42.000 99% cutoff=140.000
After: min=42.000 max=140.000

Segment_XXXXYYYY=01171296:
Before: min=22.000 max=796.000
1% cutoff=36.000 99% cutoff=108.000
After: min=36.000 max=108.000

Segment_XXXXYYYY=01181362:
Before: min=34.000 max=421.000
1% cutoff=44.000 99% cutoff=101.000
After: min=44.000 max=101.000

Segment_XXXXYYYY=01211301:
Before: min=28.000 max=2707.000
1% cutoff=47.000 99% cutoff=362.930
After: min=47.000 max=362.000

Segment_XXXXYYYY=01220335:
Before: min=22.000 max=2481.000
1% cutoff=35.000 99% cutoff=198.000
After: min=35.000 max=198.000

Segment_XXXXYYYY=01300071:
Before: min=19.000 max=698.000
1% cutoff=30.000 99% cutoff=101.000
After: min=30.000 max=101.000

Segment_XXXXYYYY=01310057:
Before: min=34.000 max=366.000
1% cutoff=45.000 99% cutoff=98.000
After: min=45.000 max=98.000

Segment_XXXXYYYY=01340702:
Before: min=69.000 max=826.000
1% cutoff=83.000 99% cutoff=157.000
After: min=83.000 max=157.000

Segment_XXXXYYYY=01346948:
Before: min=109.000 max=210.000
1% cutoff=110.740 99% cutoff=209.340
After: min=167.000 max=188.000

Segment_XXXXYYYY=01370078:
Before: min=37.000 max=964.000
1% cutoff=47.000 99% cutoff=93.000
After: min=47.000 max=93.000

Segment_XXXXYYYY=01550091:
Before: min=32.000 max=769.000
1% cutoff=45.000 99% cutoff=138.000
After: min=45.000 max=138.000

Segment_XXXXYYYY=01561440:
Before: min=30.000 max=813.000
1% cutoff=41.000 99% cutoff=65.000
After: min=41.000 max=65.000

Segment_XXXXYYYY=02610807:
Before: min=7.000 max=3630.000
1% cutoff=20.000 99% cutoff=48.000
After: min=20.000 max=48.000

Segment_XXXXYYYY=03141308:
Before: min=57.000 max=632.000
1% cutoff=57.150 99% cutoff=608.600
After: min=58.000 max=476.000

Segment_XXXXYYYY=03141309:
Before: min=34.000 max=2076.000
1% cutoff=38.000 99% cutoff=284.450
After: min=38.000 max=275.000

Segment_XXXXYYYY=03151045:
Before: min=20.000 max=270.000
1% cutoff=21.000 99% cutoff=180.630
After: min=21.000 max=180.000

Segment_XXXXYYYY=03151722:
Before: min=50.000 max=455.000
1% cutoff=54.840 99% cutoff=386.960
After: min=55.000 max=386.000

Segment_XXXXYYYY=03340121:
Before: min=18.000 max=1374.000
1% cutoff=34.000 99% cutoff=161.000
After: min=34.000 max=161.000

Segment_XXXXYYYY=03353303:
Before: min=-2.000 max=4017.000
1% cutoff=18.000 99% cutoff=100.000
After: min=18.000 max=100.000

Segment_XXXXYYYY=03530631:
Before: min=-6.000 max=401.000
1% cutoff=14.000 99% cutoff=129.000
After: min=14.000 max=129.000

Segment_XXXXYYYY=03560087:
Before: min=19.000 max=1046.000
1% cutoff=29.000 99% cutoff=73.000
After: min=29.000 max=73.000

Segment_XXXXYYYY=03740042:
Before: min=12.000 max=1756.000
1% cutoff=26.000 99% cutoff=230.750
After: min=26.000 max=230.000

Segment_XXXXYYYY=03751297:
Before: min=46.000 max=528.000
1% cutoff=53.000 99% cutoff=192.000
After: min=53.000 max=192.000

Segment_XXXXYYYY=04220801:
Before: min=23.000 max=2154.000
1% cutoff=41.000 99% cutoff=97.000
After: min=41.000 max=97.000

Segment_XXXXYYYY=04270356:
Before: min=46.000 max=1011.000
1% cutoff=54.000 99% cutoff=226.000
After: min=54.000 max=226.000

Segment_XXXXYYYY=04270699:
Before: min=44.000 max=966.000
1% cutoff=48.000 99% cutoff=365.410
After: min=48.000 max=350.000

Segment_XXXXYYYY=04300422:
Before: min=97.000 max=637.000
1% cutoff=105.860 99% cutoff=399.260
After: min=106.000 max=398.000

Segment_XXXXYYYY=04620118:
Before: min=82.000 max=82.000
1% cutoff=82.000 99% cutoff=82.000
After: min=82.000 max=82.000

Segment_XXXXYYYY=04621765:
Before: min=6.000 max=11.000
1% cutoff=6.000 99% cutoff=10.850
After: min=6.000 max=8.000

Segment_XXXXYYYY=05221298:
Before: min=33.000 max=1445.000
1% cutoff=54.000 99% cutoff=194.900
After: min=54.000 max=194.000

Segment_XXXXYYYY=05237165:
Before: min=55.000 max=1994.000
1% cutoff=69.000 99% cutoff=195.360
After: min=69.000 max=195.000

Segment_XXXXYYYY=06302117:
Before: min=3.000 max=967.000
1% cutoff=16.000 99% cutoff=63.000
After: min=16.000 max=63.000

Segment_XXXXYYYY=06310053:
Before: min=10.000 max=596.000
1% cutoff=17.000 99% cutoff=67.000
After: min=17.000 max=67.000

Segment_XXXXYYYY=06380422:
Before: min=40.000 max=640.000
1% cutoff=55.000 99% cutoff=282.360
After: min=55.000 max=275.000

Segment_XXXXYYYY=06860806:
Before: min=17.000 max=681.000
1% cutoff=35.000 99% cutoff=77.000
After: min=35.000 max=77.000

Segment_XXXXYYYY=06870925:
Before: min=10.000 max=629.000
1% cutoff=29.000 99% cutoff=188.000
After: min=29.000 max=188.000

Segment_XXXXYYYY=06991307:
Before: min=31.000 max=988.000
1% cutoff=37.000 99% cutoff=394.670
After: min=37.000 max=389.000

Segment_XXXXYYYY=07023002:
Before: min=18.000 max=2222.000
1% cutoff=33.000 99% cutoff=56.000
After: min=33.000 max=56.000

Segment_XXXXYYYY=07030137:
Before: min=72.000 max=1335.000
1% cutoff=86.000 99% cutoff=161.000
After: min=86.000 max=161.000

Segment_XXXXYYYY=07036948:
Before: min=39.000 max=39.000
1% cutoff=39.000 99% cutoff=39.000
After: min=39.000 max=39.000

Segment_XXXXYYYY=07207166:
Before: min=19.000 max=869.000
1% cutoff=29.000 99% cutoff=122.000
After: min=29.000 max=122.000

Segment_XXXXYYYY=07211023:
Before: min=15.000 max=1427.000
1% cutoff=21.000 99% cutoff=66.000
After: min=21.000 max=66.000

Segment_XXXXYYYY=07800353:
Before: min=25.000 max=894.000
1% cutoff=36.000 99% cutoff=343.000
After: min=36.000 max=343.000

Segment_XXXXYYYY=07823301:
Before: min=9.000 max=1335.000
1% cutoff=20.000 99% cutoff=144.100
After: min=20.000 max=144.000

Segment_XXXXYYYY=07826495:
Before: min=124.000 max=199.000
1% cutoff=124.900 99% cutoff=198.400
After: min=169.000 max=169.000

Segment_XXXXYYYY=08000427:
Before: min=30.000 max=849.000
1% cutoff=40.000 99% cutoff=133.000
After: min=40.000 max=133.000

Segment_XXXXYYYY=08011082:
Before: min=27.000 max=2163.000
1% cutoff=39.000 99% cutoff=127.000
After: min=39.000 max=127.000

Segment_XXXXYYYY=08060260*:
Before: min=11.000 max=357.000
1% cutoff=23.000 99% cutoff=50.000
After: min=23.000 max=50.000

Segment_XXXXYYYY=08070687:
Before: min=21.000 max=768.000
1% cutoff=39.000 99% cutoff=79.000
After: min=39.000 max=79.000

Segment_XXXXYYYY=09240686:
Before: min=26.000 max=1589.000
1% cutoff=34.000 99% cutoff=163.030
After: min=34.000 max=163.000

Segment_XXXXYYYY=09253003:
Before: min=25.000 max=1462.000
1% cutoff=34.000 99% cutoff=108.630
After: min=34.000 max=108.000

Segment_XXXXYYYY=09826495:
Before: min=8.000 max=430.000
1% cutoff=18.000 99% cutoff=91.000
After: min=18.000 max=91.000

Segment_XXXXYYYY=09833302:
Before: min=15.000 max=639.000
1% cutoff=20.000 99% cutoff=129.000
After: min=20.000 max=129.000

Segment_XXXXYYYY=10220720:
Before: min=15.000 max=601.000
1% cutoff=21.000 99% cutoff=153.000
After: min=21.000 max=153.000

Segment_XXXXYYYY=10231308:
Before: min=29.000 max=1029.000
1% cutoff=45.000 99% cutoff=184.000
After: min=45.000 max=184.000

Segment_XXXXYYYY=10371414:
Before: min=1.000 max=925.000
1% cutoff=12.000 99% cutoff=58.000
After: min=12.000 max=58.000

Segment_XXXXYYYY=10383304:
Before: min=6.000 max=511.000
1% cutoff=11.000 99% cutoff=133.000
After: min=11.000 max=133.000

Segment_XXXXYYYY=10391063:
Before: min=25.000 max=954.000
1% cutoff=40.000 99% cutoff=87.000
After: min=40.000 max=87.000

Segment_XXXXYYYY=10411128:
Before: min=14.000 max=2853.000
1% cutoff=29.000 99% cutoff=95.000
After: min=29.000 max=95.000

Segment_XXXXYYYY=10420314:
Before: min=25.000 max=551.000
1% cutoff=30.060 99% cutoff=152.640
After: min=31.000 max=147.000

Segment_XXXXYYYY=10451128:
Before: min=54.000 max=338.000
1% cutoff=54.220 99% cutoff=329.640
After: min=56.000 max=262.000

Segment_XXXXYYYY=10630086:
Before: min=55.000 max=1766.000
1% cutoff=69.000 99% cutoff=186.000
After: min=69.000 max=186.000

Segment_XXXXYYYY=10640314:
Before: min=76.000 max=200.000
1% cutoff=77.540 99% cutoff=198.020
After: min=90.000 max=182.000

Segment_XXXXYYYY=10641041:
Before: min=29.000 max=680.000
1% cutoff=38.000 99% cutoff=121.000
After: min=38.000 max=121.000

Segment_XXXXYYYY=10751361:
Before: min=23.000 max=1260.000
1% cutoff=32.000 99% cutoff=59.000
After: min=32.000 max=59.000

Segment_XXXXYYYY=10777422:
Before: min=37.000 max=670.000
1% cutoff=49.000 99% cutoff=95.750
After: min=49.000 max=95.000

Segment_XXXXYYYY=10777423:
Before: min=43.000 max=447.000
1% cutoff=52.000 99% cutoff=126.860
After: min=52.000 max=126.000

Segment_XXXXYYYY=10777691:
Before: min=41.000 max=349.000
1% cutoff=56.600 99% cutoff=168.800
After: min=57.000 max=168.000

Segment_XXXXYYYY=10810800:
Before: min=33.000 max=2960.000
1% cutoff=41.000 99% cutoff=126.940
After: min=41.000 max=126.000

Segment_XXXXYYYY=10821238:
Before: min=3.000 max=969.000
1% cutoff=19.000 99% cutoff=90.000
After: min=19.000 max=90.000

Segment_XXXXYYYY=11271039:
Before: min=3.000 max=2059.000
1% cutoff=28.000 99% cutoff=103.180
After: min=28.000 max=103.000

Segment_XXXXYYYY=11271042:
Before: min=57.000 max=890.000
1% cutoff=66.640 99% cutoff=320.360
After: min=68.000 max=319.000

Segment_XXXXYYYY=11280072:
Before: min=11.000 max=2812.000
1% cutoff=22.000 99% cutoff=197.000
After: min=22.000 max=197.000

Segment_XXXXYYYY=12371081:
Before: min=15.000 max=1162.000
1% cutoff=22.000 99% cutoff=73.000
After: min=22.000 max=73.000

Segment_XXXXYYYY=12381441:
Before: min=16.000 max=1381.000
1% cutoff=25.000 99% cutoff=53.000
After: min=25.000 max=53.000

Segment_XXXXYYYY=12960374:
Before: min=42.000 max=2247.000
1% cutoff=51.000 99% cutoff=148.000
After: min=51.000 max=148.000

Segment_XXXXYYYY=12970118:
Before: min=15.000 max=936.000
1% cutoff=32.000 99% cutoff=87.000
After: min=32.000 max=87.000

Segment_XXXXYYYY=12970124:
Before: min=44.000 max=52.000
1% cutoff=44.080 99% cutoff=51.920
After: (no data left after trimming)

Segment_XXXXYYYY=12980122:
Before: min=7.000 max=3506.000
1% cutoff=20.000 99% cutoff=129.980
After: min=20.000 max=129.000

Segment_XXXXYYYY=13010523:
Before: min=33.000 max=1470.000
1% cutoff=59.000 99% cutoff=265.000
After: min=59.000 max=265.000

Segment_XXXXYYYY=13070315:
Before: min=27.000 max=299.000
1% cutoff=32.000 99% cutoff=103.070
After: min=32.000 max=91.000

Segment_XXXXYYYY=13080086:
Before: min=41.000 max=179.000
1% cutoff=41.600 99% cutoff=168.800
After: min=45.000 max=111.000

Segment_XXXXYYYY=13082321:
Before: min=-3.000 max=848.000
1% cutoff=9.000 99% cutoff=60.000
After: min=9.000 max=60.000

Segment_XXXXYYYY=13090638:
Before: min=46.000 max=472.000
1% cutoff=52.000 99% cutoff=216.630
After: min=52.000 max=198.000

Segment_XXXXYYYY=13101022:
Before: min=26.000 max=724.000
1% cutoff=47.000 99% cutoff=221.000
After: min=47.000 max=221.000

Segment_XXXXYYYY=13110638:
Before: min=39.000 max=4647.000
1% cutoff=46.120 99% cutoff=217.960
After: min=47.000 max=203.000

Segment_XXXXYYYY=13610117:
Before: min=38.000 max=349.000
1% cutoff=50.000 99% cutoff=108.000
After: min=50.000 max=108.000

Segment_XXXXYYYY=13610125:
Before: min=47.000 max=172.000
1% cutoff=50.870 99% cutoff=168.130
After: min=56.000 max=163.000

Segment_XXXXYYYY=13621077:
Before: min=19.000 max=954.000
1% cutoff=33.000 99% cutoff=69.050
After: min=33.000 max=69.000

Segment_XXXXYYYY=14140052:
Before: min=10.000 max=841.000
1% cutoff=22.000 99% cutoff=76.000
After: min=22.000 max=76.000

Segment_XXXXYYYY=14151038:
Before: min=0.000 max=806.000
1% cutoff=10.000 99% cutoff=76.000
After: min=10.000 max=76.000

Segment_XXXXYYYY=14401237:
Before: min=26.000 max=3594.000
1% cutoff=34.000 99% cutoff=129.000
After: min=34.000 max=129.000

Segment_XXXXYYYY=14410155:
Before: min=24.000 max=2698.000
1% cutoff=39.000 99% cutoff=139.000
After: min=39.000 max=139.000

Segment_XXXXYYYY=17221128:
Before: min=14.000 max=608.000
1% cutoff=14.980 99% cutoff=527.640
After: min=21.000 max=34.000

Segment_XXXXYYYY=17650374:
Before: min=52.000 max=229.000
1% cutoff=54.400 99% cutoff=228.200
After: min=58.000 max=227.000

Segment_XXXXYYYY=21170782:
Before: min=25.000 max=2018.000
1% cutoff=41.000 99% cutoff=284.000
After: min=41.000 max=284.000

Segment_XXXXYYYY=23211310:
Before: min=64.000 max=1524.000
1% cutoff=91.000 99% cutoff=296.000
After: min=91.000 max=296.000

Segment_XXXXYYYY=25217444:
Before: min=59.000 max=1602.000
1% cutoff=87.000 99% cutoff=331.000
After: min=87.000 max=331.000

Segment_XXXXYYYY=25220983:
Before: min=165.000 max=165.000
1% cutoff=165.000 99% cutoff=165.000
After: min=165.000 max=165.000

Segment_XXXXYYYY=25226498:
Before: min=9.000 max=1117.000
1% cutoff=20.000 99% cutoff=54.000
After: min=20.000 max=54.000

Segment_XXXXYYYY=25420983:
Before: min=55.000 max=169.000
1% cutoff=55.600 99% cutoff=166.270
After: min=75.000 max=78.000

Segment_XXXXYYYY=30020924:
Before: min=14.000 max=1247.000
1% cutoff=33.000 99% cutoff=118.000
After: min=33.000 max=118.000

Segment_XXXXYYYY=30030703:
Before: min=15.000 max=993.000
1% cutoff=32.000 99% cutoff=97.000
After: min=32.000 max=97.000

Segment_XXXXYYYY=33010982:
Before: min=12.000 max=467.000
1% cutoff=21.000 99% cutoff=108.000
After: min=21.000 max=108.000

Segment_XXXXYYYY=33016495:
Before: min=84.000 max=248.000
1% cutoff=84.700 99% cutoff=245.950
After: min=98.000 max=207.000

Segment_XXXXYYYY=33020780:
Before: min=10.000 max=685.000
1% cutoff=19.000 99% cutoff=53.000
After: min=19.000 max=53.000

Segment_XXXXYYYY=33031037:
Before: min=-5.000 max=177.000
1% cutoff=7.000 99% cutoff=36.000
After: min=7.000 max=36.000

Segment_XXXXYYYY=33040334:
Before: min=7.000 max=976.000
1% cutoff=17.000 99% cutoff=251.000
After: min=17.000 max=251.000

Segment_XXXXYYYY=33490422:
Before: min=38.000 max=2121.000
1% cutoff=46.000 99% cutoff=203.720
After: min=46.000 max=203.000

Segment_XXXXYYYY=53150056:
Before: min=24.000 max=543.000
1% cutoff=44.000 99% cutoff=74.000
After: min=44.000 max=74.000

Segment_XXXXYYYY=53160043:
Before: min=31.000 max=768.000
1% cutoff=43.000 99% cutoff=95.000
After: min=43.000 max=95.000

Segment_XXXXYYYY=64956497:
Before: min=2.000 max=188.000
1% cutoff=12.000 99% cutoff=60.000
After: min=12.000 max=60.000

Segment_XXXXYYYY=64960780:
Before: min=258.000 max=266.000
1% cutoff=258.080 99% cutoff=265.920
After: (no data left after trimming)

Segment_XXXXYYYY=64960983:
Before: min=3.000 max=442.000
1% cutoff=16.000 99% cutoff=64.000
After: min=16.000 max=64.000

Segment_XXXXYYYY=64963302:
Before: min=130.000 max=130.000
1% cutoff=130.000 99% cutoff=130.000
After: min=130.000 max=130.000

Segment_XXXXYYYY=64972521:
Before: min=22.000 max=405.000
1% cutoff=37.000 99% cutoff=83.000
After: min=37.000 max=83.000

Segment_XXXXYYYY=64980983:
Before: min=70.000 max=178.000
1% cutoff=71.920 99% cutoff=171.760
After: min=78.000 max=152.000

Segment_XXXXYYYY=64986496:
Before: min=6.000 max=558.000
1% cutoff=18.000 99% cutoff=49.000
After: min=18.000 max=49.000

Segment_XXXXYYYY=70621064:
Before: min=62.000 max=215.000
1% cutoff=62.000 99% cutoff=208.120
After: min=62.000 max=172.000

Segment_XXXXYYYY=70621311:
Before: min=30.000 max=360.000
1% cutoff=43.000 99% cutoff=133.760
After: min=43.000 max=128.000

Segment_XXXXYYYY=71650721:
Before: min=30.000 max=3289.000
1% cutoff=44.000 99% cutoff=254.000
After: min=44.000 max=254.000

Segment_XXXXYYYY=71660522:
Before: min=31.000 max=3194.000
1% cutoff=46.000 99% cutoff=222.000
After: min=46.000 max=222.000

Segment_XXXXYYYY=74221075:
Before: min=31.000 max=112.000
1% cutoff=40.000 99% cutoff=77.000
After: min=40.000 max=77.000

Segment_XXXXYYYY=74231075:
Before: min=32.000 max=871.000
1% cutoff=40.000 99% cutoff=78.000
After: min=40.000 max=78.000

Segment_XXXXYYYY=74440983:
Before: min=348.000 max=348.000
1% cutoff=348.000 99% cutoff=348.000
After: min=348.000 max=348.000

Segment_XXXXYYYY=74442522:
Before: min=34.000 max=1199.000
1% cutoff=64.000 99% cutoff=197.220
After: min=64.000 max=197.000

Segment_XXXXYYYY=76867690:
Before: min=48.000 max=323.000
1% cutoff=53.000 99% cutoff=84.460
After: min=53.000 max=84.000

Segment_XXXXYYYY=76877689:
Before: min=52.000 max=429.000
1% cutoff=56.000 99% cutoff=103.760
After: min=56.000 max=103.000

Segment_XXXXYYYY=76887686:
Before: min=49.000 max=1026.000
1% cutoff=56.000 99% cutoff=116.650
After: min=56.000 max=115.000

Segment_XXXXYYYY=76897693:
Before: min=106.000 max=420.000
1% cutoff=119.000 99% cutoff=178.510
After: min=119.000 max=178.000

Segment_XXXXYYYY=76901075:
Before: min=34.000 max=342.000
1% cutoff=47.570 99% cutoff=102.430
After: min=48.000 max=102.000

Segment_XXXXYYYY=76917687:
Before: min=48.000 max=3064.000
1% cutoff=60.580 99% cutoff=127.420
After: min=61.000 max=127.000

Segment_XXXXYYYY=76927688:
Before: min=104.000 max=228.000
1% cutoff=113.600 99% cutoff=181.800
After: min=114.000 max=181.000

```
import pandas as pd

# ---- Set your OG file path ----
og_file_path = '/Users/benfall/Desktop/Data Files/OG.csv'
```

```
df = pd.read_csv(og_file_path, encoding='utf-8', low_memory=False)
df.columns = df.columns.str.strip().str.replace('\ufeff', '', regex=False)

# Find correct segment and stop columns
segment_id_col = 'Segment_XXXXYYYY'
stop_id_col = 'IdArret'

# Make needed columns numeric
for col in ['dwell_time', 'segment_speed', 'segment_travel_time']:
   if col in df.columns:
       df[col] = pd.to_numeric(df[col], errors='coerce')

def trim_and_show(df, group_col, val_col, exclude_first_stops=False):
   print(f"\n==== {val_col} by {group_col} ====")
   if exclude_first_stops and 'RangArretAs' in df.columns:
       df = df[df['RangArretAs'] > 1].copy()
   if group_col not in df.columns:
       print(f"Column '{group_col}' not found in your file!")
       return
   groups = sorted(df[group_col].dropna().unique())
   for group in groups:
       sub = df[df[group_col] == group][[val_col]].dropna()
       if len(sub) == 0:
           continue
       before_min, before_max = sub[val_col].min(), sub[val_col].max()
       q01 = sub[val_col].quantile(0.01)
       q99 = sub[val_col].quantile(0.99)
       trimmed = sub[(sub[val_col] >= q01) & (sub[val_col] <= q99)]
       after_min, after_max = (trimmed[val_col].min(), trimmed[val_col].max()) if len(trimmed) else (None,
None)
       print(f"\n{group_col}={group}:")
       print(f"  Before:   min={before_min:.3f}  max={before_max:.3f}")
       print(f"  1% cutoff={q01:.3f}  99% cutoff={q99:.3f}")
       if after_min is not None:
           print(f"  After:    min={after_min:.3f}  max={after_max:.3f}")
       else:
           print(f"  After:    (no data left after trimming)")

# ----- Dwell time by STOP, EXCLUDING first stops -----
trim_and_show(df, stop_id_col, 'dwell_time', exclude_first_stops=True)

# ----- Segment speed by SEGMENT -----
trim_and_show(df, segment_id_col, 'segment_speed')

# ----- Segment travel time by SEGMENT -----
trim_and_show(df, segment_id_col, 'segment_travel_time')
```

```
import pandas as pd

# ---- Load OG file ----
og_file_path = '/Users/benfall/Desktop/Data Files/OG.csv'
df = pd.read_csv(og_file_path, encoding='utf-8', low_memory=False)
df.columns = df.columns.str.strip().str.replace('\ufeff', '', regex=False)

# Column names as in your file
stop_id_col     = 'IdArret'
segment_id_col  = 'Segment_XXXXYYYY'

# Ensure numeric
for col in ['dwell_time', 'segment_speed', 'segment_travel_time']:
   df[col] = pd.to_numeric(df[col], errors='coerce')

# --- Compute bounds for each group ---
# Dwell time: by stop, excluding first stops
use_for_dwell = df[df['RangArretAs'] > 1]
dwell_bounds = use_for_dwell.groupby(stop_id_col)['dwell_time'].quantile([0.01, 0.99]).unstack()
dwell_bounds.columns = ['dwell_min', 'dwell_max']

# Segment speed: by segment
speed_bounds = df.groupby(segment_id_col)['segment_speed'].quantile([0.01, 0.99]).unstack()
speed_bounds.columns = ['speed_min', 'speed_max']

# Segment travel time: by segment
time_bounds = df.groupby(segment_id_col)['segment_travel_time'].quantile([0.01, 0.99]).unstack()
```

```
time_bounds.columns = ['time_min', 'time_max']

# --- Merge bounds back to records ---
result = df.copy()

# Set NaN boundaries for first stops for dwell_time, so we don't exclude them for missing bounds
result = result.merge(dwell_bounds, left_on=stop_id_col, right_index=True, how='left')
result = result.merge(speed_bounds, left_on=segment_id_col, right_index=True, how='left')
result = result.merge(time_bounds, left_on=segment_id_col, right_index=True, how='left')

# --- Build mask for trimming ---
# For dwell_time, only enforce for RangArretAs > 1
not_first_stop = result['RangArretAs'] > 1
dwell_ok = (~not_first_stop) | ((result['dwell_time'] >= result['dwell_min']) & (result['dwell_time'] <=
result['dwell_max']))
speed_ok = (result['segment_speed'] >= result['speed_min']) & (result['segment_speed'] <= result['speed_max'])
time_ok  = (result['segment_travel_time'] >= result['time_min']) & (result['segment_travel_time'] <=
result['time_max'])

# --- Only keep rows where all are within bounds ---
mask = dwell_ok & speed_ok & time_ok
trimmed = result[mask].copy()

# --- Drop bound columns (optional) ---
trimmed = trimmed[df.columns]  # Keep only original columns

# --- Save new file ---
trimmed_path = og_file_path.replace('.csv', '_trimmed.csv')
trimmed.to_csv(trimmed_path, index=False)
print(f"Trimmed OG file (all three 1-99% filters applied) saved to:\n{trimmed_path}")
print(f"Kept {len(trimmed)} of {len(df)} rows. Example rows:")
print(trimmed.head())
```

```
import pandas as pd

# ---- Load OG file ----
og_file_path = '/Users/benfall/Desktop/Data Files/OG.csv'
df = pd.read_csv(og_file_path, encoding='utf-8', low_memory=False)
df.columns = df.columns.str.strip().str.replace('\ufeff', '', regex=False)

stop_id_col = 'IdArret'
segment_id_col = 'Segment_XXXXYYYY'

# Ensure numeric columns
for col in ['dwell_time', 'segment_speed', 'segment_travel_time']:
   df[col] = pd.to_numeric(df[col], errors='coerce')

# Compute quantile bounds

# Dwell time bounds (exclude first stops for quantile calc)
dwell_bounds = (
   df[df['RangArretAs'] > 1]
   .groupby(stop_id_col)['dwell_time']
   .quantile([0.01, 0.99])
   .unstack()
)
dwell_bounds.columns = ['dwell_min', 'dwell_max']

# Segment speed bounds
speed_bounds = (
   df.groupby(segment_id_col)['segment_speed']
   .quantile([0.01, 0.99])
   .unstack()
)
speed_bounds.columns = ['speed_min', 'speed_max']

# Segment travel time bounds
time_bounds = (
   df.groupby(segment_id_col)['segment_travel_time']
   .quantile([0.01, 0.99])
   .unstack()
)
time_bounds.columns = ['time_min', 'time_max']

# Merge bounds back into dataframe
result = df.copy()
```

```
result = result.merge(dwell_bounds, left_on=stop_id_col, right_index=True, how='left')
result = result.merge(speed_bounds, left_on=segment_id_col, right_index=True, how='left')
result = result.merge(time_bounds, left_on=segment_id_col, right_index=True, how='left')

# Fill missing bounds with open intervals so no unintentional exclusions
result['dwell_min'] = result['dwell_min'].fillna(-float('inf'))
result['dwell_max'] = result['dwell_max'].fillna(float('inf'))
result['speed_min'] = result['speed_min'].fillna(-float('inf'))
result['speed_max'] = result['speed_max'].fillna(float('inf'))
result['time_min'] = result['time_min'].fillna(-float('inf'))
result['time_max'] = result['time_max'].fillna(float('inf'))

# Build masks
is_first_stop = result['RangArretAs'] == 1
not_first_stop = result['RangArretAs'] > 1

# Conditions
speed_ok = (result['segment_speed'] >= result['speed_min']) & (result['segment_speed'] <= result['speed_max'])
time_ok = (result['segment_travel_time'] >= result['time_min']) & (result['segment_travel_time'] <=
result['time_max'])
dwell_ok = (result['dwell_time'] >= result['dwell_min']) & (result['dwell_time'] <= result['dwell_max'])

# Final mask:
# Keep all first stops unconditionally, and only trim other stops meeting all three conditions
mask = is_first_stop | (not_first_stop & dwell_ok & speed_ok & time_ok)

trimmed = result[mask].copy()

# Remove helper bound columns before saving (keep only original columns)
trimmed = trimmed[df.columns]

# Save trimmed data to new file
trimmed_path = og_file_path.replace('.csv', '_trimmed_with_first_stops.csv')
trimmed.to_csv(trimmed_path, index=False)

# Print summary
print(f"Trimmed data saved to: {trimmed_path}")
print(f"Total rows before: {len(df)}")
print(f"Total rows after trimming: {len(trimmed)}")
print(f"Rows with RangArretAs == 1 before: {(df['RangArretAs'] == 1).sum()}")
print(f"Rows with RangArretAs == 1 after:  {(trimmed['RangArretAs'] == 1).sum()}")
```

```
import pandas as pd


# Path to your trimmed OG file
file_path = '/Users/benfall/Desktop/Data Files/OG_trimmed_with_first_stops.csv'


# Load the trimmed data
df = pd.read_csv(file_path, low_memory=False)


# Segments for each line/direction as per your query
segments_18_A = [
  "74221075", "10751361", "13610117", "01171296", "12960374", "03740042", "00425315", "53150056",
  "00560130", "01300071", "00711127", "11271039", "10391063", "10630086", "00863349", "33490422",
  "04220801", "08011082", "10821238", "12381441", "14410155", "01550091", "00910077", "00770134",
  "01340702", "07023002", "30020924", "09240686", "06860806", "08060260"
]


segments_18_R = [
  "02610807", "08070687", "06870925", "09253003", "30030703", "07030137", "01370078", "00780092",
  "00920156", "01561440", "14401237", "12371081", "10810800", "08000427", "04270356", "03560087",
  "00871064", "10641041", "10411128", "11280072", "00720131", "01310057", "00575316", "53160043",
  "00430375", "03751297", "12970118", "01181362", "13621077", "10777423"
]


segments_80_A = [
  "23211310", "13101022", "10220720", "07207166", "71660522", "05221298", "12980122", "01220335",
  "03353303", "33031037", "10371414", "14140052", "00520630", "06302117", "21170782", "07823301",
  "33010982", "09826495", "64956497", "64972521", "25217444"
]
```

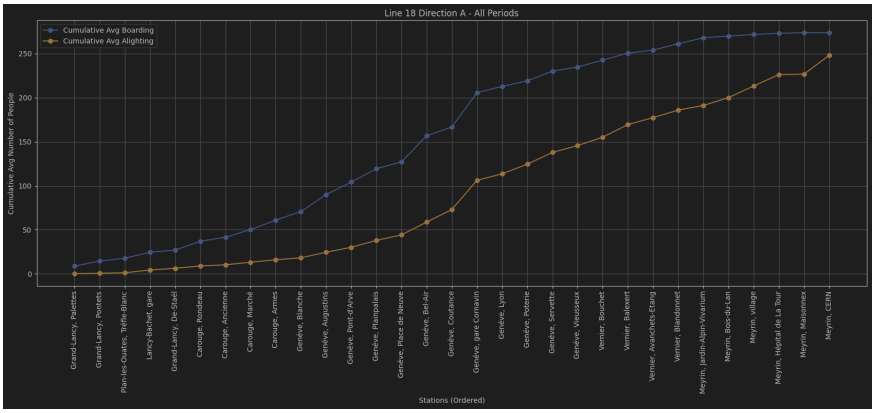
```
segments_80_R = [
  "74442522", "25226498", "64986496", "64960983", "09833302", "33020780", "07800353", "03530631",
  "06310053", "00531415", "14151038", "10383304", "33040334", "03340121", "01211301", "13010523",
  "05237165", "71650721", "07211023", "10231308", "13082321"
]


segment_sets = [
  (segments_18_A, "Line 18 Direction A"),
  (segments_18_R, "Line 18 Direction R"),
  (segments_80_A, "Line 80 Direction A"),
  (segments_80_R, "Line 80 Direction R")
]


# Calculate and print average segment speeds
for segment_list, label in segment_sets:
  print(f"\nAverage segment speeds for {label}:")
  results = []
  for segment in segment_list:
      seg_data = df[df['Segment_XXXXYYYY'] == segment]
      avg_speed = seg_data['segment_speed'].mean()
      results.append((segment, avg_speed))
  # Print results
  for segment, speed in results:
      print(f"Segment {segment}: Average Speed = {speed:.2f} (km per hour)")
```

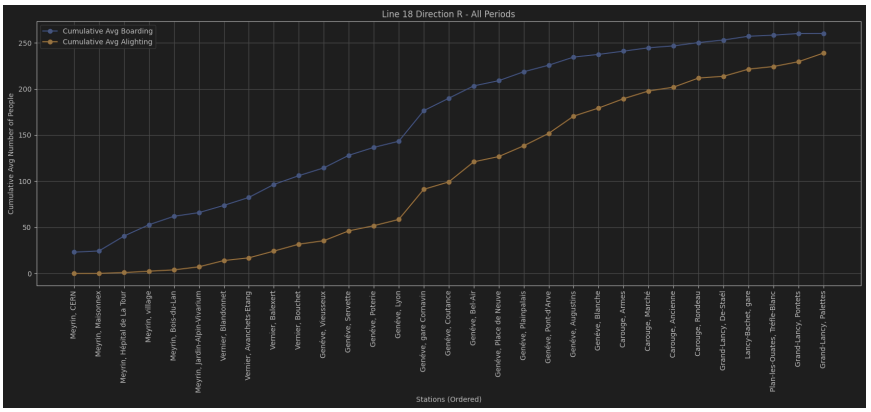
Appendix 4:

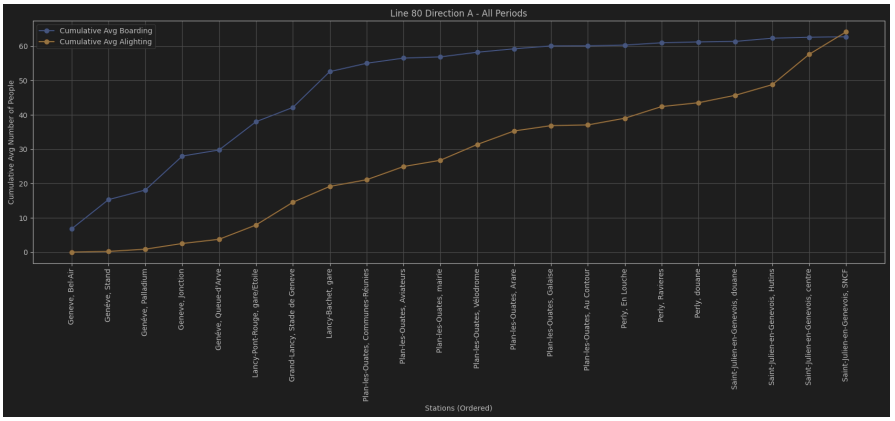
```
import pandas as pd
import numpy as np

# PARAMETERS (your thresholds)
alpha1 = 60     # sec, large dwell time
alpha3 = -60    # sec, early departure
alpha4 = 180    # sec, late departure
alpha7 = 15     # km/h, low speed

# --- 1. Load OG data ---
file_og = '/Users/benfall/Desktop/Data Files/OG_trimmed_with_first_stops.csv'
df = pd.read_csv(file_og, encoding='utf-8', low_memory=False)
df.columns = df.columns.str.strip().str.replace('\ufeff', '', regex=False)

# Ensure relevant columns are numeric/datetime
for col in ['dwell_time', 'EcartDepar', 'segment_speed', 'segment_travel_time', 'RangArretAs']:
   if col in df.columns:
       df[col] = pd.to_numeric(df[col], errors='coerce')
for col in ['DTDepartTheo', 'DTEntreeFenetreArretRea', 'DTSortieFenetreArretRea']:
   if col in df.columns:
       df[col] = pd.to_datetime(df[col], errors='coerce')

# Time period assignment
def assign_period(dt):
   hour = dt.hour if pd.notnull(dt) else -1
   day = dt.dayofweek if pd.notnull(dt) else -1
   if day == 5:
       return "Saturdays"
   if day == 6:
       return "Sundays"
   if 7 <= hour < 9:
       return "AM peak"
   elif 9 <= hour < 16:
       return "Day period"
   elif 16 <= hour < 18:
       return "PM peak"
   elif 18 <= hour < 24:
       return "Evening"
   else:
       return "Other"
df['period'] = df['DTDepartTheo'].apply(assign_period)

# ------ Your combined segments list -------
segments_all = [
   # Line 18 direction A
   '74221075', '10751361', '13610117', '01171296', '12960374', '03740042', '00425315', '53150056',
```

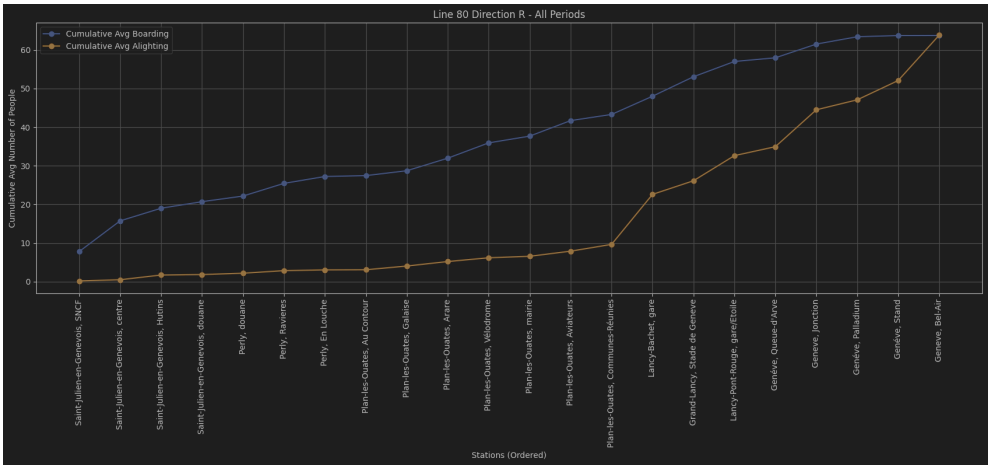
```
    '00560130', '01300071', '00711127', '11271039', '10391063', '10630086', '00863349', '33490422',
    '04220801', '08011082', '10821238', '12381441', '14410155', '01550091', '00910077', '00770134',
    '01340702', '07023002', '30020924', '09240686', '06860806', '08060260',
    # Line 18 direction R
    '02610807', '08070687', '06870925', '09253003', '30030703', '07030137', '01370078', '00780092',
    '00920156', '01561440', '14401237', '12371081', '10810800', '08000427', '04270356', '03560087',
    '00871064', '10641041', '10411128', '11280072', '00720131', '01310057', '00575316', '53160043',
    '00430375', '03751297', '12970118', '01181362', '13621077', '10777423',
    # Line 80 direction A
    '23211310', '13101022', '10220720', '07207166', '71660522', '05221298', '12980122', '01220335',
    '03353303', '33031037', '10371414', '14140052', '00520630', '06302117', '21170782', '07823301',
    '33010982', '09826495', '64956497', '64972521', '25217444',
    # Line 80 direction R
    '74442522', '25226498', '64986496', '64960983', '09833302', '33020780', '07800353', '03530631',
    '06310053', '00531415', '14151038', '10383304', '33040334', '03340121', '01211301', '13010523',
    '05237165', '71650721', '07211023', '10231308', '13082321'
]

# --- 2. Filter for all relevant segment IDs ---
df_seg = df[df["Segment_XXXXYYYY"].isin(segments_all)].copy()

# --- 3. Bottleneck criteria for ALL rows in all chosen segments ---
# 1. Large dwell time
df_seg["large_dwell_time"] = df_seg["dwell_time"] > alpha1

# 2. Early and late departure
df_seg["early_departure"] = df_seg["EcartDepar"] < alpha3
df_seg["late_departure"]  = df_seg["EcartDepar"] > alpha4

# 3. Low speed
df_seg["low_speed"] = df_seg["segment_speed"] < alpha7

# --- 4. Flag any criterion ---
criteria_cols = [
    "large_dwell_time", "early_departure", "late_departure", "low_speed"
]
df_seg["any_criterion"] = df_seg[criteria_cols].any(axis=1)

# --- 5. Summary table (one per period, for ALL listed segments together) ---
summary = (
    df_seg.groupby("period")[criteria_cols + ["any_criterion"]].sum().astype(int)
)
summary["num_trips"] = df_seg.groupby("period").size()
summary["any_criterion_percent"] = (100.0 * summary["any_criterion"] / summary["num_trips"]).round(1)
summary = summary.reset_index()

# Order periods for pretty output
order = ["AM peak", "Day period", "PM peak", "Evening", "Saturdays", "Sundays", "Other"]
summary["period"] = pd.Categorical(summary["period"], categories=order, ordered=True)
summary = summary.sort_values("period")

print(summary)
summary.to_csv("/Users/benfall/Desktop/bottleneck_summary_ALLSEGMENTS.csv", index=False)

import pandas as pd
import numpy as np

# ------------------- PARAMETERS (as per your table definitions) -------------------
alpha1 = 60     # sec, large dwell time
alpha3 = -60    # sec, early departure
alpha4 = 180    # sec, late departure
alpha7 = 15     # km/h, low speed

# ------------------- LOAD DATA -------------------
file_og = '/Users/benfall/Desktop/Data Files/OG_trimmed_with_first_stops.csv'
df = pd.read_csv(file_og, encoding='utf-8', low_memory=False)
df.columns = df.columns.str.strip().str.replace('\ufeff', '', regex=False)

# Ensure relevant columns are numeric/datetime
for col in ['dwell_time', 'EcartDepar', 'segment_speed', 'RangArretAs']:
    if col in df.columns:
        df[col] = pd.to_numeric(df[col], errors='coerce')
for col in ['DTDepartTheo', 'DTEntreeFenetreArretRea', 'DTSortieFenetreArretRea']:
    if col in df.columns:
        df[col] = pd.to_datetime(df[col], errors='coerce')
```

```python
# Period assignment using theoretical departure time
def assign_period(dt):
   hour = dt.hour if pd.notnull(dt) else -1
   day = dt.dayofweek if pd.notnull(dt) else -1
   if day == 5:
       return "Saturdays"
   if day == 6:
       return "Sundays"
   if 7 <= hour < 9:
       return "AM peak"
   elif 9 <= hour < 16:
       return "Day period"
   elif 16 <= hour < 18:
       return "PM peak"
   elif 18 <= hour < 24:
       return "Evening"
   else:
       return "Other"
df['period'] = df['DTDepartTheo'].apply(assign_period)

# ------------------- SELECT SEGMENT ------------------
segment_id = "13010523"  # Change this to the segment of interest
df_seg = df[df["Segment_XXXXYYYY"] == segment_id].copy()

# ------------------- BOTTLE NECK LOGIC (revised) ------------------

# 1. Large dwell time
df_seg["large_dwell_time"] = df_seg["dwell_time"] > alpha1

# 3. Early departure
df_seg["early_departure"] = df_seg["EcartDepar"] < alpha3

# 4. Late departure
df_seg["late_departure"]  = df_seg["EcartDepar"] > alpha4

# 7. Low speed
df_seg["low_speed"] = df_seg["segment_speed"] < alpha7

# At least one bottleneck criterion (row-wise)
criteria_cols = [
   "large_dwell_time", "early_departure", "late_departure", "low_speed"
]
df_seg["any_criterion"] = df_seg[criteria_cols].any(axis=1)

# ------------------- SUMMARY TABLE ------------------
summary = (
   df_seg.groupby("period")[criteria_cols + ["any_criterion"]].sum().astype(int)
)
summary["num_trips"] = df_seg.groupby("period").size()

# Add any_criterion percent column
summary["any_criterion_percent"] = (100.0 * summary["any_criterion"] / summary["num_trips"]).round(1)

summary = summary.reset_index()

# Order periods
desired_order = ["AM peak", "Day period", "PM peak", "Evening", "Saturdays", "Sundays", "Other"]
summary["period"] = pd.Categorical(summary["period"], categories=desired_order, ordered=True)
summary = summary.sort_values("period")

print(summary)
summary.to_csv(f"/Users/benfall/Desktop/segment{segment_id}_bottleneck_summary_with_n.csv", index=False)
```


Line 18 Direction A - All Periods
Cumulative Avg Boarding
Cumulative Avg Alighting
Cumulative Avg Number of People
Grand-Lancy, Palettes
Grand-Lancy, Pontets
Plan-les-Ouates, Trèfle-Blanc
Lancy-Bachet, gare
Grand-Lancy, De-Staël
Carouge, Rondeau
Carouge, Ancienne
Carouge, Marché
Carouge, Armes
Genève, Blanche
Genève, Augustins
Genève, Pont-d'Arve
Genève, Plainpalais
Genève, Place de Neuve
Genève, Bel-Air
Genève, Coutance
Genève, gare Cornavin
Genève, Lyon
Genève, Poterie
Genève, Servette
Genève, Vieusseux
Vernier, Bauchet
Vernier, Balexert
Vernier, Avanchets-Etang
Vernier, Blandonnet
Meyrin, Jardin-Alpin-Vivarium
Meyrin, Bois-du-Lan
Meyrin, village
Meyrin, Hôpital de la Tour
Meyrin, Maisonnex
Meyrin, CERN
Stations (Ordered)


Line 18 Direction R - All Periods
Cumulative Avg Boarding
Cumulative Avg Alighting
Cumulative Avg Number of People
Meyrin, CERN
Meyrin, Maisonnex
Meyrin, Hôpital de la Tour
Meyrin, village
Meyrin, Bois-du-Lan
Meyrin, Jardin-Alpin-Vivarium
Vernier, Blandonnet
Vernier, Avanchets-Etang
Vernier, Balexert
Vernier, Bauchet
Genève, Vieusseux
Genève, Servette
Genève, Poterie
Genève, Lyon
Genève, gare Cornavin
Genève, Coutance
Genève, Bel-Air
Genève, Place de Neuve
Genève, Plainpalais
Genève, Pont-d'Arve
Genève, Augustins
Genève, Blanche
Carouge, Armes
Carouge, Marché
Carouge, Ancienne
Carouge, Rondeau
Grand-Lancy, De-Staël
Lancy-Bachet, gare
Plan-les-Ouates, Trèfle-Blanc
Grand-Lancy, Pontets
Grand-Lancy, Palettes
Stations (Ordered)


Line 80 Direction A - All Periods
Cumulative Avg Boarding
Cumulative Avg Alighting
Cumulative Avg Number of People
Genève, Bel-Air
Genève, Stand
Genève, Palladium
Genève, Jonction
Genève, Queue-d'Arve
Lancy-Pont-Rouge, gare/Etoile
Grand-Lancy, Stade de Genève
Lancy-Bachet, gare
Plan-les-Ouates, Communes-Réunies
Plan-les-Ouates, Aviateurs
Plan-les-Ouates, mairie
Plan-les-Ouates, Vélodrome
Plan-les-Ouates, Arare
Plan-les-Ouates, Galaise
Plan-les-Ouates, Au Contour
Perly, En Louche
Perly, Ravenes
Perly, douane
Saint-Julien-en-Genevois, douane
Saint-Julien-en-Genevois, Hutins
Saint-Julien-en-Genevois, centre
Saint-Julien-en-Genevois, SNCF
Stations (Ordered)


Line 80 Direction R - All Periods
Cumulative Avg Boarding
Cumulative Avg Alighting
Cumulative Avg Number of People
Saint-Julien-en-Genevois, SNCF
Saint-Julien-en-Genevois, centre
Saint-Julien-en-Genevois, Hutins
Saint-Julien-en-Genevois, douane
Perly, douane
Perly, Ravenes
Perly, En Louche
Plan-les-Ouates, Au Contour
Plan-les-Ouates, Galaise
Plan-les-Ouates, Arare
Plan-les-Ouates, Vélodrome
Plan-les-Ouates, mairie
Plan-les-Ouates, Aviateurs
Plan-les-Ouates, Communes-Réunies
Lancy-Bachet, gare
Grand-Lancy, Stade de Genève
Lancy-Pont-Rouge, gare/Etoile
Genève, Queue-d'Arve
Genève, Jonction
Genève, Palladium
Genève, Stand
Genève, Bel-Air
Stations (Ordered)

Appendix 5:

```python
import pandas as pd

file_path = '/Users/benfall/Desktop/Data Files/OG_trimmed_with_first_stops.csv'
df = pd.read_csv(file_path, encoding='utf-8', low_memory=False)
df.columns = df.columns.str.strip().str.replace('\ufeff', '', regex=False)

df['EcartDepar'] = pd.to_numeric(df['EcartDepar'], errors='coerce')
df['RangArretAs'] = pd.to_numeric(df['RangArretAs'], errors='coerce')
df = df.sort_values(['IdCourse', 'RangArretAs'])

# Compute previous stop order/EcartDepar for each trip
df['prev_RangArretAs'] = df.groupby('IdCourse')['RangArretAs'].shift(1)
df['prev_EcartDepar'] = df.groupby('IdCourse')['EcartDepar'].shift(1)

# Only compute difference for strictly consecutive stops
df['delay_from_segment'] = df['EcartDepar'] - df['prev_EcartDepar']
df.loc[df['RangArretAs'] - df['prev_RangArretAs'] != 1, 'delay_from_segment'] = pd.NA

# Segment lists (in required order)
segments_18A = [
   '74221075', '10751361', '13610117', '01171296', '12960374', '03740042', '00425315', '53150056',
   '00560130', '01300071', '00711127', '11271039', '10391063', '10630086', '00863349', '33490422',
   '04220801', '08011082', '10821238', '12381441', '14410155', '01550091', '00910077', '00770134',
   '01340702', '07023002', '30020924', '09240686', '06860806', '08060260'
]
segments_18R = [
   '02610807', '08070687', '06870925', '09253003', '30030703', '07030137', '01370078', '00780092',
   '00920156', '01561440', '14401237', '12371081', '10810800', '08000427', '04270356', '03560087',
   '00871064', '10641041', '10411128', '11280072', '00720131', '01310057', '00575316', '53160043',
   '00430375', '03751297', '12970118', '01181362', '13621077', '10777423'
]
segments_80A = [
   '23211310', '13101022', '10220720', '07207166', '71660522', '05221298', '12980122', '01220335',
   '03353303', '33031037', '10371414', '14140052', '00520630', '06302117', '21170782', '07823301',
   '33010982', '09826495', '64956497', '64972521', '25217444'
]
segments_80R = [
   '74442522', '25226498', '64986496', '64960983', '09833302', '33020780', '07800353', '03530631',
   '06310053', '00531415', '14151038', '10383304', '33040334', '03340121', '01211301', '13010523',
   '05237165', '71650721', '07211023', '10231308', '13082321'
]

def avg_delay_table(df, segment_list, line, direction):
   # Filter only these segments and with a valid delay
   dsub = df[df['delay_from_segment'].notnull() & df['Segment_XXXXYYYY'].astype(str).isin(segment_list)]
   # Compute average delay for each segment
   avgdelays = dsub.groupby('Segment_XXXXYYYY')['delay_from_segment'].mean()
   # Create output table in the specified input order, even for missing segments
   out = pd.DataFrame({'Segment_XXXXYYYY': segment_list})
   out['average_delay_from_segment'] = out['Segment_XXXXYYYY'].map(avgdelays)
   out['line'] = line
   out['direction'] = direction
   return out

# Get all 4 tables
table_18A = avg_delay_table(df, segments_18A, 18, 'A')
table_18R = avg_delay_table(df, segments_18R, 18, 'R')
table_80A = avg_delay_table(df, segments_80A, 80, 'A')
table_80R = avg_delay_table(df, segments_80R, 80, 'R')

# Print them clearly
print("\nLine 18 Direction A:")
print(table_18A.to_string(index=False))

print("\nLine 18 Direction R:")
print(table_18R.to_string(index=False))

print("\nLine 80 Direction A:")
print(table_80A.to_string(index=False))

print("\nLine 80 Direction R:")
print(table_80R.to_string(index=False))

# Optionally save all 4 tables individually
table_18A.to_csv('/Users/benfall/Desktop/avg_delay_segment_18A.csv', index=False)
table_18R.to_csv('/Users/benfall/Desktop/avg_delay_segment_18R.csv', index=False)
```

```
table_80A.to_csv('/Users/benfall/Desktop/avg_delay_segment_80A.csv', index=False)
table_80R.to_csv('/Users/benfall/Desktop/avg_delay_segment_80R.csv', index=False)
```

```
import pandas as pd

# --- PARAMETERS ---
UPPER_LIMIT = 300
LOWER_LIMIT = -300

# --- LOAD AND PREPARE DATA ---
file_path = '/Users/benfall/Desktop/Data Files/OG_trimmed_with_first_stops.csv'
df = pd.read_csv(file_path, encoding='utf-8', low_memory=False)
df.columns = df.columns.str.strip().str.replace('\ufeff', '', regex=False)

df['DTDepartTheo'] = pd.to_datetime(df['DTDepartTheo'], errors='coerce')
df['EcartDepar'] = pd.to_numeric(df['EcartDepar'], errors='coerce')
df['RangArretAs'] = pd.to_numeric(df['RangArretAs'], errors='coerce')

df = df.sort_values(['IdCourse', 'RangArretAs'])

# --- Time period assignment (match your previous convention) ---
def assign_period(dt):
   hour = dt.hour if pd.notnull(dt) else -1
   day = dt.dayofweek if pd.notnull(dt) else -1
   if day == 5: return "Saturdays"
   if day == 6: return "Sundays"
   if 7 <= hour < 9: return "AM peak"
   elif 9 <= hour < 16: return "Day period"
   elif 16 <= hour < 18: return "PM peak"
   elif 18 <= hour < 24: return "Evening"
   else: return "Other"

df['period'] = df['DTDepartTheo'].apply(assign_period)

# --- STRICT DELAY LOGIC ---
df['prev_RangArretAs'] = df.groupby('IdCourse')['RangArretAs'].shift(1)
df['prev_EcartDepar'] = df.groupby('IdCourse')['EcartDepar'].shift(1)

df['delay_from_segment'] = df['EcartDepar'] - df['prev_EcartDepar']
df.loc[(df['RangArretAs'] - df['prev_RangArretAs'] != 1), 'delay_from_segment'] = pd.NA

# --- SEGMENT LISTS ---
segments_18A = [
   '74221075', '10751361', '13610117', '01171296', '12960374', '03740042', '00425315', '53150056',
   '00560130', '01300071', '00711127', '11271039', '10391063', '10630086', '00863349', '33490422',
   '04220801', '08011082', '10821238', '12381441', '14410155', '01550091', '00910077', '00770134',
   '01340702', '07023002', '30020924', '09240686', '06860806', '08060260'
]
segments_18R = [
   '02610807', '08070687', '06870925', '09253003', '30030703', '07030137', '01370078', '00780092',
   '00920156', '01561440', '14401237', '12371081', '10810800', '08000427', '04270356', '03560087',
   '00871064', '10641041', '10411128', '11280072', '00720131', '01310057', '00575316', '53160043',
   '00430375', '03751297', '12970118', '01181362', '13621077', '10777423'
]
segments_80A = [
   '23211310', '13101022', '10220720', '07207166', '71660522', '05221298', '12980122', '01220335',
   '03353303', '33031037', '10371414', '14140052', '00520630', '06302117', '21170782', '07823301',
   '33010982', '09826495', '64956497', '64972521', '25217444'
]
segments_80R = [
   '74442522', '25226498', '64986496', '64960983', '09833302', '33020780', '07800353', '03530631',
   '06310053', '00531415', '14151038', '10383304', '33040334', '03340121', '01211301', '13010523',
   '05237165', '71650721', '07211023', '10231308', '13082321'
]
period_order = ["AM peak", "Day period", "PM peak", "Evening", "Saturdays", "Sundays", "Other"]

def avg_delay_by_period(df_in, segment_list, line, direction):
   # Only segments in your list and with valid delay
   dsub = df_in[
       df_in['delay_from_segment'].notnull()
       & df_in['Segment_XXXXYYYY'].astype(str).isin(segment_list)
   ].copy()
   # Exclude outliers
   dsub = dsub[
       (dsub['delay_from_segment'] < UPPER_LIMIT)
       & (dsub['delay_from_segment'] > LOWER_LIMIT)
```

```
    ].copy()

    # Net (overall) average per segment
    net_avg = dsub.groupby('Segment_XXXXYYYY')['delay_from_segment'].mean()

    # Average per segment per period
    period_avg = dsub.groupby(['Segment_XXXXYYYY', 'period'])['delay_from_segment'].mean().unstack()
    # Reorder periods and segments
    period_avg = period_avg.reindex(segment_list).reindex(period_order, axis=1)
    net_avg = net_avg.reindex(segment_list)

    # Output table
    out = pd.DataFrame({'Segment_XXXXYYYY': segment_list})
    for per in period_order:
        out[per] = period_avg[per].values if per in period_avg.columns else pd.NA
    out['net'] = net_avg.values
    out['line'] = line
    out['direction'] = direction
    return out

# Compute all tables
table_18A = avg_delay_by_period(df, segments_18A, 18, 'A')
table_18R = avg_delay_by_period(df, segments_18R, 18, 'R')
table_80A = avg_delay_by_period(df, segments_80A, 80, 'A')
table_80R = avg_delay_by_period(df, segments_80R, 80, 'R')

# Print a few rows as demo:
print("\nLine 18 Direction A:")
print(table_18A.head())

print("\nLine 18 Direction R:")
print(table_18R.head())

print("\nLine 80 Direction A:")
print(table_80A.head())

print("\nLine 80 Direction R:")
print(table_80R.head())

# Optional: Save to CSV
table_18A.to_csv('/Users/benfall/Desktop/avg_delay_segment_18A_periods.csv', index=False)
table_18R.to_csv('/Users/benfall/Desktop/avg_delay_segment_18R_periods.csv', index=False)
table_80A.to_csv('/Users/benfall/Desktop/avg_delay_segment_80A_periods.csv', index=False)
table_80R.to_csv('/Users/benfall/Desktop/avg_delay_segment_80R_periods.csv', index=False)
```

Appendix 6:

```
import pandas as pd

file_path = '/Users/benfall/Desktop/Data Files/OG_trimmed_with_first_stops.csv'
df = pd.read_csv(file_path, encoding='utf-8', low_memory=False)
df.columns = df.columns.str.strip().str.replace('\ufeff', '', regex=False)

df['EcartDepar'] = pd.to_numeric(df['EcartDepar'], errors='coerce')
df['DTDepartTheo'] = pd.to_datetime(df['DTDepartTheo'], errors='coerce')

# Time period function
def assign_period(dt):
    hour = dt.hour if pd.notnull(dt) else -1
    day = dt.dayofweek if pd.notnull(dt) else -1
    if day == 5:  return "Saturdays"
    if day == 6:  return "Sundays"
    if 7 <= hour < 9:   return "AM peak"
    elif 9 <= hour < 16:return "Day period"
    elif 16 <= hour < 18:return "PM peak"
    elif 18 <= hour < 24:return "Evening"
    else:               return "Other"
df['period'] = df['DTDepartTheo'].apply(assign_period)

# Segment lists (in required order)
segments_18A = [
    '74221075','10751361','13610117','01171296','12960374','03740042','00425315','53150056','00560130',
    '01300071','00711127','11271039','10391063','10630086','00863349','33490422','04220801','08011082',
    '10821238','12381441','14410155','01550091','00910077','00770134','01340702','07023002','30020924',
```

```
    '09240686','06860806','08060260'
]
segments_18R = [
    '02610807','08070687','06870925','09253003','30030703','07030137','01370078','00780092','00920156',
    '01561440','14401237','12371081','10810800','08000427','04270356','03560087','00871064','10641041',
    '10411128','11280072','00720131','01310057','00575316','53160043','00430375','03751297','12970118',
    '01181362','13621077','10777423'
]
segments_80A = [
    '23211310','13101022','10220720','07207166','71660522','05221298','12980122','01220335','03353303',
    '33031037','10371414','14140052','00520630','06302117','21170782','07823301','33010982','09826495',
    '64956497','64972521','25217444'
]
segments_80R = [
    '74442522','25226498','64986496','64960983','09833302','33020780','07800353','03530631','06310053',
    '00531415','14151038','10383304','33040334','03340121','01211301','13010523','05237165','71650721',
    '07211023','10231308','13082321'
]
segment_map = {}
for v in segments_18A: segment_map[v] = (18, "A")
for v in segments_18R: segment_map[v] = (18, "R")
for v in segments_80A: segment_map[v] = (80, "A")
for v in segments_80R: segment_map[v] = (80, "R")
all_segments = segments_18A + segments_18R + segments_80A + segments_80R

periods = ["AM peak", "Day period", "PM peak", "Evening", "Saturdays", "Sundays", "Other"]

# Only for valid EcartDepar
valid_mask = (df['EcartDepar'] > -300) & (df['EcartDepar'] < 300)
df_valid = df[valid_mask & df['Segment_XXXXYYYY'].astype(str).isin(all_segments)].copy()

# Substantial delay flags for each row/segment/period
df_valid['delay_pos'] = (df_valid['EcartDepar'] >= 45)
df_valid['delay_neg'] = (df_valid['EcartDepar'] <= -45)
df_valid['substantial_delay'] = df_valid['delay_pos'] | df_valid['delay_neg']

df_valid['line'] = df_valid['Segment_XXXXYYYY'].map(lambda x: segment_map.get(str(x), [None])[0])
df_valid['direction'] = df_valid['Segment_XXXXYYYY'].map(lambda x: segment_map.get(str(x), [None, None])[1])

grouped = df_valid.groupby(['Segment_XXXXYYYY', 'period', 'line', 'direction'])
result = grouped.agg(
    num_substantial_delays = ('substantial_delay', 'sum'),
    num_pos_delays         = ('delay_pos', 'sum'),
    num_neg_delays         = ('delay_neg', 'sum'),
    num_valid_trips        = ('EcartDepar', 'count')
).reset_index()
result['percent_substantial_delays'] = (100 * result['num_substantial_delays'] /
result['num_valid_trips']).round(1)

# Ensure output is ordered by segment (your input order) and then by period
ordered_segments = segments_18A + segments_18R + segments_80A + segments_80R
ordered_periods = periods
from pandas import Categorical

result['Segment_XXXXYYYY'] = pd.Categorical(result['Segment_XXXXYYYY'], categories=ordered_segments,
ordered=True)
result['period'] = pd.Categorical(result['period'], categories=ordered_periods, ordered=True)
result = result.sort_values(['Segment_XXXXYYYY', 'period'])

# Keep columns as requested
result =
result[['Segment_XXXXYYYY','period','line','direction','num_substantial_delays','num_pos_delays','num_neg_delay
s','num_valid_trips','percent_substantial_delays']]

# Output
result.to_csv('/Users/benfall/Desktop/delay_summary_segment_all_periods.csv', index=False)
print(result.head(20))
```

If we consider “percent substantial delay” to be the trips where delay is longer than 45 seconds out of those that are less than 300 seconds.

Segment_XXXXYYYY	period	line	direction	num_substantial_delays	num_pos_delays	num_neg_delays	num_valid_trips	percent_substantial_delays

74221075	AM peak	18	A	141	120	21	838	16.8
74221075	Day period	18	A	557	454	103	2347	23.7
74221075	PM peak	18	A	228	196	32	773	29.5
74221075	Evening	18	A	507	401	106	1768	28.7
74221075	Saturdays	18	A	383	322	61	1441	26.6
74221075	Sundays	18	A	128	95	33	533	24.0
74221075	Other	18	A	147	122	25	533	27.6
10751361	AM peak	18	A	589	548	41	1832	32.2
10751361	Day period	18	A	1891	1694	197	5709	33.1
10751361	PM peak	18	A	533	480	53	1667	32.0
10751361	Evening	18	A	1189	919	270	3578	33.2
10751361	Saturdays	18	A	784	647	137	2368	33.1
10751361	Sundays	18	A	575	435	140	1938	29.7
10751361	Other	18	A	657	581	76	1792	36.7
13610117	AM peak	18	A	849	816	33	1827	46.5
13610117	Day period	18	A	2439	2258	181	5690	42.9
13610117	PM peak	18	A	656	606	50	1663	39.4
13610117	Evening	18	A	1417	1141	276	3513	40.3
13610117	Saturdays	18	A	973	847	126	2382	40.8
13610117	Sundays	18	A	740	589	151	1934	38.3
13610117	Other	18	A	854	792	62	1783	47.9
1171296	AM peak	18	A	841	813	28	1814	46.4
1171296	Day period	18	A	2439	2258	181	5391	45.2
1171296	PM peak	18	A	722	678	44	1731	41.7
1171296	Evening	18	A	1345	1186	159	3252	41.4

1171296	Saturdays	18	A	936	801	135	2210	42.4
1171296	Sundays	18	A	728	569	159	1771	41.1
1171296	Other	18	A	911	860	51	1729	52.7
12960374	AM peak	18	A	906	863	43	1801	50.3
12960374	Day period	18	A	2412	2038	374	5366	44.9
12960374	PM peak	18	A	744	632	112	1734	42.9
12960374	Evening	18	A	1396	1162	234	3255	42.9
12960374	Saturdays	18	A	988	841	147	2205	44.8
12960374	Sundays	18	A	744	516	228	1799	41.4
12960374	Other	18	A	917	837	80	1740	52.7
3740042	AM peak	18	A	997	954	43	1758	56.7
3740042	Day period	18	A	2829	2463	366	5658	50.0
3740042	PM peak	18	A	819	713	106	1730	47.3
3740042	Evening	18	A	1559	1312	247	3284	47.5
3740042	Saturdays	18	A	1074	928	146	2253	47.7
3740042	Sundays	18	A	846	588	258	1854	45.6
3740042	Other	18	A	969	865	104	1805	53.7
425315	AM peak	18	A	994	944	50	1747	56.9
425315	Day period	18	A	2867	2485	382	5603	51.2
425315	PM peak	18	A	831	725	106	1674	49.6
425315	Evening	18	A	1656	1399	257	3340	49.6
425315	Saturdays	18	A	1114	968	146	2225	50.1
425315	Sundays	18	A	877	605	272	1841	47.6
425315	Other	18	A	1010	895	115	1832	55.1
53150056	AM peak	18	A	988	921	67	1819	54.3

53150056	Day period	18	A	2854	2382	472	5593	51.0
53150056	PM peak	18	A	815	669	146	1640	49.7
53150056	Evening	18	A	1663	1348	315	3366	49.4
53150056	Saturdays	18	A	1130	952	178	2250	50.2
53150056	Sundays	18	A	893	567	326	1838	48.6
53150056	Other	18	A	935	789	146	1753	53.3
560130	AM peak	18	A	1055	1001	54	1796	58.7
560130	Day period	18	A	2958	2526	432	5520	53.6
560130	PM peak	18	A	914	763	151	1707	53.5
560130	Evening	18	A	1738	1436	302	3350	51.9
560130	Saturdays	18	A	1204	1046	158	2257	53.3
560130	Sundays	18	A	948	632	316	1845	51.4
560130	Other	18	A	978	829	149	1784	54.8
1300071	AM peak	18	A	945	857	88	1787	52.9
1300071	Day period	18	A	2871	2267	604	5425	52.9
1300071	PM peak	18	A	951	737	214	1741	54.6
1300071	Evening	18	A	1728	1274	454	3313	52.2
1300071	Saturdays	18	A	1156	886	270	2264	51.1
1300071	Sundays	18	A	978	516	462	1854	52.8
1300071	Other	18	A	932	718	214	1840	50.7
711127	AM peak	18	A	958	856	102	1779	53.9
711127	Day period	18	A	2963	2342	621	5392	55.0
711127	PM peak	18	A	965	742	223	1702	56.7
711127	Evening	18	A	1724	1269	455	3226	53.4
711127	Saturdays	18	A	1200	903	297	2327	51.6

711127	Sundays	18	A	1006	543	463	1941	51.8
711127	Other	18	A	1023	754	269	1927	53.1
11271039	AM peak	18	A	928	837	91	1584	58.6
11271039	Day period	18	A	3116	2560	556	5388	57.8
11271039	PM peak	18	A	976	768	208	1675	58.3
11271039	Evening	18	A	1771	1355	416	3149	56.2
11271039	Saturdays	18	A	1162	865	297	2150	54.0
11271039	Sundays	18	A	948	573	375	1825	51.9
11271039	Other	18	A	1020	800	220	1814	56.2
10391063	AM peak	18	A	990	832	158	1708	58.0
10391063	Day period	18	A	3019	2221	798	5372	56.2
10391063	PM peak	18	A	921	648	273	1588	58.0
10391063	Evening	18	A	1781	1231	550	3185	55.9
10391063	Saturdays	18	A	1177	825	352	2156	54.6
10391063	Sundays	18	A	971	534	437	1828	53.1
10391063	Other	18	A	948	663	285	1734	54.7
10630086	AM peak	18	A	1024	876	148	1738	58.9
10630086	Day period	18	A	3161	2412	749	5377	58.8
10630086	PM peak	18	A	898	692	206	1515	59.3
10630086	Evening	18	A	1839	1290	549	3199	57.5
10630086	Saturdays	18	A	1199	895	304	2149	55.8
10630086	Sundays	18	A	977	543	434	1824	53.6
10630086	Other	18	A	981	678	303	1727	56.8
863349	AM peak	18	A	993	578	415	1759	56.5
863349	Day period	18	A	3211	1771	1440	5329	60.3

863349	PM peak	18	A	968	579	389	1523	63.6
863349	Evening	18	A	1933	823	1110	3173	60.9
863349	Saturdays	18	A	1239	585	654	2171	57.1
863349	Sundays	18	A	1131	281	850	1803	62.7
863349	Other	18	A	1074	383	691	1697	63.3
33490422	AM peak	18	A	1060	861	199	1684	62.9
33490422	Day period	18	A	3328	2547	781	5238	63.5
33490422	PM peak	18	A	1036	815	221	1574	65.8
33490422	Evening	18	A	1942	1278	664	3136	61.9
33490422	Saturdays	18	A	1212	866	346	2131	56.9
33490422	Sundays	18	A	996	558	438	1816	54.8
33490422	Other	18	A	1039	643	396	1730	60.1
4220801	AM peak	18	A	1105	953	152	1678	65.9
4220801	Day period	18	A	3582	3018	564	5396	66.4
4220801	PM peak	18	A	1082	900	182	1577	68.6
4220801	Evening	18	A	2731	1975	756	4236	64.5
4220801	Saturdays	18	A	1637	1265	372	2563	63.9
4220801	Sundays	18	A	1286	886	400	2230	57.7
4220801	Other	18	A	1579	958	621	2545	62.0
8011082	AM peak	18	A	1029	814	215	1666	61.8
8011082	Day period	18	A	3633	2897	736	5454	66.6
8011082	PM peak	18	A	1008	797	211	1437	70.1
8011082	Evening	18	A	2877	1898	979	4358	66.0
8011082	Saturdays	18	A	1627	1130	497	2582	63.0
8011082	Sundays	18	A	1321	813	508	2219	59.5

8011082	Other	18	A	1639	774	865	2592	63.2
10821238	AM peak	18	A	1080	842	238	1748	61.8
10821238	Day period	18	A	3732	2950	782	5465	68.3
10821238	PM peak	18	A	1051	865	186	1460	72.0
10821238	Evening	18	A	2912	1830	1082	4335	67.2
10821238	Saturdays	18	A	1689	1164	525	2561	66.0
10821238	Sundays	18	A	1374	852	522	2222	61.8
10821238	Other	18	A	1622	792	830	2513	64.5
12381441	AM peak	18	A	1053	813	240	1712	61.5
12381441	Day period	18	A	3693	2875	818	5406	68.3
12381441	PM peak	18	A	1060	854	206	1442	73.5
12381441	Evening	18	A	2899	1788	1111	4311	67.2
12381441	Saturdays	18	A	1687	1179	508	2554	66.1
12381441	Sundays	18	A	1378	887	491	2213	62.3
12381441	Other	18	A	1611	780	831	2480	65.0
14410155	AM peak	18	A	1105	833	272	1721	64.2
14410155	Day period	18	A	3681	2723	958	5337	69.0
14410155	PM peak	18	A	1101	867	234	1505	73.2
14410155	Evening	18	A	2972	1723	1249	4346	68.4
14410155	Saturdays	18	A	1709	1140	569	2576	66.3
14410155	Sundays	18	A	1388	863	525	2197	63.2
14410155	Other	18	A	1640	731	909	2440	67.2
1550091	AM peak	18	A	1079	791	288	1684	64.1
1550091	Day period	18	A	3740	2758	982	5340	70.0
1550091	PM peak	18	A	1110	871	239	1523	72.9

1550091	Evening	18	A	3017	1698	1319	4367	69.1
1550091	Saturdays	18	A	1722	1097	625	2578	66.8
1550091	Sundays	18	A	1401	818	583	2212	63.3
1550091	Other	18	A	1698	707	991	2440	69.6
910077	AM peak	18	A	1067	826	241	1656	64.4
910077	Day period	18	A	3739	2863	876	5303	70.5
910077	PM peak	18	A	1055	847	208	1427	73.9
910077	Evening	18	A	3086	1843	1243	4436	69.6
910077	Saturdays	18	A	1757	1139	618	2579	68.1
910077	Sundays	18	A	1410	867	543	2209	63.8
910077	Other	18	A	1719	741	978	2455	70.0
770134	AM peak	18	A	1070	816	254	1664	64.3
770134	Day period	18	A	3781	2894	887	5291	71.5
770134	PM peak	18	A	1062	896	166	1420	74.8
770134	Evening	18	A	2969	1753	1216	4242	70.0
770134	Saturdays	18	A	1704	1117	587	2523	67.5
770134	Sundays	18	A	1458	808	650	2215	65.8
770134	Other	18	A	1849	706	1143	2561	72.2
1340702	AM peak	18	A	1079	787	292	1644	65.6
1340702	Day period	18	A	3613	2721	892	5127	70.5
1340702	PM peak	18	A	1090	893	197	1481	73.6
1340702	Evening	18	A	2367	1576	791	3450	68.6
1340702	Saturdays	18	A	1470	1084	386	2201	66.8
1340702	Sundays	18	A	1190	736	454	1853	64.2
1340702	Other	18	A	890	511	379	1359	65.5

7023002	AM peak	18	A	1159	903	256	1713	67.7
7023002	Day period	18	A	3627	2850	777	5120	70.8
7023002	PM peak	18	A	1080	899	181	1450	74.5
7023002	Evening	18	A	2393	1625	768	3460	69.2
7023002	Saturdays	18	A	1535	1174	361	2294	66.9
7023002	Sundays	18	A	1227	824	403	1917	64.0
7023002	Other	18	A	1157	800	357	1798	64.3
30020924	AM peak	18	A	1095	788	307	1605	68.2
30020924	Day period	18	A	3740	2851	889	5268	71.0
30020924	PM peak	18	A	1070	880	190	1429	74.9
30020924	Evening	18	A	2385	1478	907	3425	69.6
30020924	Saturdays	18	A	1569	1119	450	2290	68.5
30020924	Sundays	18	A	1254	755	499	1918	65.4
30020924	Other	18	A	1181	737	444	1811	65.2
9240686	AM peak	18	A	1109	730	379	1607	69.0
9240686	Day period	18	A	3768	2599	1169	5190	72.6
9240686	PM peak	18	A	1137	911	226	1494	76.1
9240686	Evening	18	A	2413	1394	1019	3407	70.8
9240686	Saturdays	18	A	1594	1023	571	2289	69.6
9240686	Sundays	18	A	1288	750	538	1905	67.6
9240686	Other	18	A	1205	641	564	1812	66.5
6860806	AM peak	18	A	1140	675	465	1649	69.1
6860806	Day period	18	A	3721	2341	1380	5163	72.1
6860806	PM peak	18	A	1072	826	246	1413	75.9
6860806	Evening	18	A	2532	1308	1224	3452	73.3

6860806	Saturdays	18	A	1606	931	675	2295	70.0
6860806	Sundays	18	A	1317	685	632	1917	68.7
6860806	Other	18	A	1261	535	726	1811	69.6
8060260	AM peak	18	A	1184	668	516	1682	70.4
8060260	Day period	18	A	3779	2327	1452	5192	72.8
8060260	PM peak	18	A	1098	817	281	1461	75.2
8060260	Evening	18	A	2570	1281	1289	3483	73.8
8060260	Saturdays	18	A	1640	938	702	2313	70.9
8060260	Sundays	18	A	1342	687	655	1924	69.8
8060260	Other	18	A	1217	503	714	1731	70.3
2610807	AM peak	18	R	390	303	87	1750	22.3
2610807	Day period	18	R	1235	881	354	5472	22.6
2610807	PM peak	18	R	409	281	128	1721	23.8
2610807	Evening	18	R	918	598	320	3643	25.2
2610807	Saturdays	18	R	505	366	139	2407	21.0
2610807	Sundays	18	R	495	353	142	1942	25.5
2610807	Other	18	R	369	274	95	1752	21.1
8070687	AM peak	18	R	571	513	58	1843	31.0
8070687	Day period	18	R	1804	1647	157	5555	32.5
8070687	PM peak	18	R	580	510	70	1688	34.4
8070687	Evening	18	R	1252	1032	220	3717	33.7
8070687	Saturdays	18	R	682	594	88	2399	28.4
8070687	Sundays	18	R	609	511	98	1934	31.5
8070687	Other	18	R	457	377	80	1649	27.7
6870925	AM peak	18	R	552	380	172	1829	30.2

6870925	Day period	18	R	1767	1395	372	5569	31.7
6870925	PM peak	18	R	644	527	117	1693	38.0
6870925	Evening	18	R	1403	1008	395	3807	36.9
6870925	Saturdays	18	R	726	565	161	2430	29.9
6870925	Sundays	18	R	650	471	179	1957	33.2
6870925	Other	18	R	472	311	161	1647	28.7
9253003	AM peak	18	R	593	317	276	1831	32.4
9253003	Day period	18	R	1871	1208	663	5663	33.0
9253003	PM peak	18	R	619	455	164	1541	40.2
9253003	Evening	18	R	1520	983	537	3903	38.9
9253003	Saturdays	18	R	745	509	236	2427	30.7
9253003	Sundays	18	R	686	459	227	1952	35.1
9253003	Other	18	R	537	312	225	1607	33.4
30030703	AM peak	18	R	589	357	232	1793	32.8
30030703	Day period	18	R	1988	1352	636	5634	35.3
30030703	PM peak	18	R	660	525	135	1599	41.3
30030703	Evening	18	R	1646	1148	498	3866	42.6
30030703	Saturdays	18	R	847	612	235	2434	34.8
30030703	Sundays	18	R	721	519	202	1964	36.7
30030703	Other	18	R	556	357	199	1577	35.3
7030137	AM peak	18	R	697	390	307	1792	38.9
7030137	Day period	18	R	2236	1656	580	5538	40.4
7030137	PM peak	18	R	758	561	197	1599	47.4
7030137	Evening	18	R	1405	1061	344	3280	42.8
7030137	Saturdays	18	R	954	815	139	2264	42.1

7030137	Sundays	18	R	794	647	147	1819	43.7
7030137	Other	18	R	453	314	139	1307	34.7
1370078	AM peak	18	R	747	518	229	1835	40.7
1370078	Day period	18	R	2458	1872	586	5474	44.9
1370078	PM peak	18	R	874	621	253	1694	51.6
1370078	Evening	18	R	1511	1152	359	3309	45.7
1370078	Saturdays	18	R	1060	921	139	2300	46.1
1370078	Sundays	18	R	872	718	154	1870	46.6
1370078	Other	18	R	680	483	197	1695	40.1
780092	AM peak	18	R	797	594	203	1830	43.6
780092	Day period	18	R	2667	2160	507	5461	48.8
780092	PM peak	18	R	903	699	204	1672	54.0
780092	Evening	18	R	1644	1334	310	3261	50.4
780092	Saturdays	18	R	1195	1064	131	2283	52.3
780092	Sundays	18	R	964	837	127	1877	51.4
780092	Other	18	R	755	582	173	1754	43.0
920156	AM peak	18	R	868	595	273	1818	47.7
920156	Day period	18	R	2797	2085	712	5474	51.1
920156	PM peak	18	R	899	658	241	1598	56.3
920156	Evening	18	R	1714	1321	393	3314	51.7
920156	Saturdays	18	R	1184	984	200	2315	51.1
920156	Sundays	18	R	982	791	191	1889	52.0
920156	Other	18	R	787	559	228	1738	45.3
1561440	AM peak	18	R	899	478	421	1782	50.4
1561440	Day period	18	R	2897	1650	1247	5553	52.2

1561440	PM peak	18	R	944	576	368	1621	58.2
1561440	Evening	18	R	1786	1132	654	3369	53.0
1561440	Saturdays	18	R	1148	789	359	2315	49.6
1561440	Sundays	18	R	956	619	337	1878	50.9
1561440	Other	18	R	800	424	376	1681	47.6
14401237	AM peak	18	R	938	655	283	1786	52.5
14401237	Day period	18	R	2976	2062	914	5587	53.3
14401237	PM peak	18	R	958	703	255	1590	60.3
14401237	Evening	18	R	1913	1428	485	3364	56.9
14401237	Saturdays	18	R	1229	993	236	2271	54.1
14401237	Sundays	18	R	1060	814	246	1867	56.8
14401237	Other	18	R	863	653	210	1664	51.9
12371081	AM peak	18	R	961	513	448	1783	53.9
12371081	Day period	18	R	3047	1678	1369	5564	54.8
12371081	PM peak	18	R	972	596	376	1594	61.0
12371081	Evening	18	R	1886	1117	769	3369	56.0
12371081	Saturdays	18	R	1259	866	393	2298	54.8
12371081	Sundays	18	R	999	611	388	1865	53.6
12371081	Other	18	R	911	547	364	1678	54.3
10810800	AM peak	18	R	1095	379	716	1859	58.9
10810800	Day period	18	R	3328	1281	2047	5554	59.9
10810800	PM peak	18	R	1016	481	535	1611	63.1
10810800	Evening	18	R	1880	849	1031	3255	57.8
10810800	Saturdays	18	R	1327	690	637	2323	57.1
10810800	Sundays	18	R	1045	458	587	1890	55.3

10810800	Other	18	R	948	397	551	1700	55.8
8000427	AM peak	18	R	981	585	396	1795	54.7
8000427	Day period	18	R	3153	1947	1206	5524	57.1
8000427	PM peak	18	R	988	730	258	1562	63.3
8000427	Evening	18	R	1863	1312	551	3264	57.1
8000427	Saturdays	18	R	1291	992	299	2290	56.4
8000427	Sundays	18	R	1041	726	315	1890	55.1
8000427	Other	18	R	887	589	298	1696	52.3
4270356	AM peak	18	R	989	827	162	1654	59.8
4270356	Day period	18	R	3306	2750	556	5270	62.7
4270356	PM peak	18	R	1027	931	96	1420	72.3
4270356	Evening	18	R	2130	1867	263	3134	68.0
4270356	Saturdays	18	R	1392	1262	130	2070	67.2
4270356	Sundays	18	R	1125	984	141	1721	65.4
4270356	Other	18	R	1034	883	151	1596	64.8
3560087	AM peak	18	R	963	544	419	1641	58.7
3560087	Day period	18	R	3232	1966	1266	5322	60.7
3560087	PM peak	18	R	984	674	310	1443	68.2
3560087	Evening	18	R	1962	1353	609	3168	61.9
3560087	Saturdays	18	R	1308	993	315	2083	62.8
3560087	Sundays	18	R	1090	810	280	1744	62.5
3560087	Other	18	R	1013	705	308	1577	64.2
871064	AM peak	18	R	967	541	426	1645	58.8
871064	Day period	18	R	3305	1952	1353	5290	62.5
871064	PM peak	18	R	1014	669	345	1440	70.4

871064	Evening	18	R	2034	1348	686	3194	63.7
871064	Saturdays	18	R	1350	950	400	2091	64.6
871064	Sundays	18	R	1061	729	332	1733	61.2
871064	Other	18	R	1051	693	358	1625	64.7
10641041	AM peak	18	R	1035	629	406	1698	61.0
10641041	Day period	18	R	3299	2064	1235	5223	63.2
10641041	PM peak	18	R	1026	712	314	1475	69.6
10641041	Evening	18	R	2095	1468	627	3202	65.4
10641041	Saturdays	18	R	1366	1033	333	2060	66.3
10641041	Sundays	18	R	1076	788	288	1728	62.3
10641041	Other	18	R	958	676	282	1481	64.7
10411128	AM peak	18	R	1012	594	418	1663	60.9
10411128	Day period	18	R	3403	2087	1316	5301	64.2
10411128	PM peak	18	R	1020	725	295	1473	69.2
10411128	Evening	18	R	2148	1478	670	3261	65.9
10411128	Saturdays	18	R	1397	1100	297	2065	67.7
10411128	Sundays	18	R	1071	771	300	1720	62.3
10411128	Other	18	R	923	646	277	1421	65.0
11280072	AM peak	18	R	1045	607	438	1717	60.9
11280072	Day period	18	R	3561	2219	1342	5433	65.5
11280072	PM peak	18	R	1058	775	283	1513	69.9
11280072	Evening	18	R	2259	1534	725	3358	67.3
11280072	Saturdays	18	R	1474	1193	281	2141	68.8
11280072	Sundays	18	R	1151	788	363	1825	63.1
11280072	Other	18	R	1019	704	315	1594	63.9

720131	AM peak	18	R	1063	555	508	1672	63.6
720131	Day period	18	R	3628	2005	1623	5389	67.3
720131	PM peak	18	R	1078	709	369	1524	70.7
720131	Evening	18	R	2281	1368	913	3361	67.9
720131	Saturdays	18	R	1462	1076	386	2124	68.8
720131	Sundays	18	R	1121	690	431	1766	63.5
720131	Other	18	R	1014	637	377	1551	65.4
1310057	AM peak	18	R	999	631	368	1630	61.3
1310057	Day period	18	R	3611	2338	1273	5429	66.5
1310057	PM peak	18	R	1061	750	311	1488	71.3
1310057	Evening	18	R	2280	1505	775	3335	68.4
1310057	Saturdays	18	R	1458	1158	300	2085	69.9
1310057	Sundays	18	R	1110	782	328	1745	63.6
1310057	Other	18	R	971	685	286	1518	64.0
575316	AM peak	18	R	1002	664	338	1631	61.4
575316	Day period	18	R	3621	2471	1150	5397	67.1
575316	PM peak	18	R	1061	772	289	1480	71.7
575316	Evening	18	R	2290	1533	757	3343	68.5
575316	Saturdays	18	R	1455	1176	279	2101	69.3
575316	Sundays	18	R	1114	796	318	1723	64.7
575316	Other	18	R	965	673	292	1512	63.8
53160043	AM peak	18	R	1060	640	420	1645	64.4
53160043	Day period	18	R	3641	2330	1311	5385	67.6
53160043	PM peak	18	R	1066	690	376	1472	72.4
53160043	Evening	18	R	2343	1411	932	3387	69.2

53160043	Saturdays	18	R	1461	1092	369	2104	69.4
53160043	Sundays	18	R	1129	740	389	1741	64.8
53160043	Other	18	R	1016	588	428	1558	65.2
430375	AM peak	18	R	1045	698	347	1640	63.7
430375	Day period	18	R	3635	2464	1171	5362	67.8
430375	PM peak	18	R	991	718	273	1383	71.7
430375	Evening	18	R	2369	1473	896	3379	70.1
430375	Saturdays	18	R	1460	1106	354	2082	70.1
430375	Sundays	18	R	1157	789	368	1729	66.9
430375	Other	18	R	993	618	375	1525	65.1
3751297	AM peak	18	R	1061	764	297	1639	64.7
3751297	Day period	18	R	3576	2581	995	5281	67.7
3751297	PM peak	18	R	1076	784	292	1435	75.0
3751297	Evening	18	R	2349	1342	1007	3303	71.1
3751297	Saturdays	18	R	1496	1112	384	2082	71.9
3751297	Sundays	18	R	1143	853	290	1688	67.7
3751297	Other	18	R	1080	614	466	1607	67.2
12970118	AM peak	18	R	1081	748	333	1620	66.7
12970118	Day period	18	R	3606	2490	1116	5215	69.1
12970118	PM peak	18	R	1080	787	293	1434	75.3
12970118	Evening	18	R	2325	1302	1023	3281	70.9
12970118	Saturdays	18	R	1501	1121	380	2067	72.6
12970118	Sundays	18	R	1204	917	287	1744	69.0
12970118	Other	18	R	1072	589	483	1574	68.1
1181362	AM peak	18	R	1163	801	362	1792	64.9

1181362	Day period	18	R	3582	2469	1113	5258	68.1
1181362	PM peak	18	R	1069	786	283	1452	73.6
1181362	Evening	18	R	2247	1290	957	3224	69.7
1181362	Saturdays	18	R	1547	1136	411	2236	69.2
1181362	Sundays	18	R	1254	946	308	1844	68.0
1181362	Other	18	R	1127	595	532	1777	63.4
13621077	AM peak	18	R	1110	824	286	1722	64.5
13621077	Day period	18	R	3607	2651	956	5261	68.6
13621077	PM peak	18	R	1078	838	240	1448	74.4
13621077	Evening	18	R	2229	1380	849	3215	69.3
13621077	Saturdays	18	R	1540	1195	345	2190	70.3
13621077	Sundays	18	R	1272	1003	269	1829	69.5
13621077	Other	18	R	1112	624	488	1757	63.3
10777423	AM peak	18	R	534	384	150	816	65.4
10777423	Day period	18	R	1848	1257	591	2615	70.7
10777423	PM peak	18	R	527	383	144	683	77.2
10777423	Evening	18	R	884	507	377	1245	71.0
10777423	Saturdays	18	R	414	330	84	560	73.9
10777423	Sundays	18	R	812	650	162	1127	72.0
10777423	Other	18	R	625	348	277	1028	60.8
23211310	AM peak	80	A	1598	1579	19	2261	70.7
23211310	Day period	80	A	3584	3509	75	5379	66.6
23211310	PM peak	80	A	1520	1421	99	2498	60.8
23211310	Evening	80	A	2487	2455	32	4192	59.3
23211310	Saturdays	80	A	1373	1368	5	1995	68.8

23211310	Sundays	80	A	1045	1043	2	1669	62.6
23211310	Other	80	A	758	745	13	1243	61.0
13101022	AM peak	80	A	1396	1265	131	2318	60.2
13101022	Day period	80	A	3057	2554	503	5525	55.3
13101022	PM peak	80	A	1267	913	354	2340	54.1
13101022	Evening	80	A	2314	1891	423	4386	52.8
13101022	Saturdays	80	A	1261	1193	68	2071	60.9
13101022	Sundays	80	A	943	900	43	1676	56.3
13101022	Other	80	A	593	534	59	1199	49.5
10220720	AM peak	80	A	1639	1574	65	2253	72.7
10220720	Day period	80	A	3454	3219	235	5263	65.6
10220720	PM peak	80	A	1577	1411	166	2349	67.1
10220720	Evening	80	A	2792	2567	225	4284	65.2
10220720	Saturdays	80	A	1557	1527	30	2076	75.0
10220720	Sundays	80	A	1233	1209	24	1760	70.1
10220720	Other	80	A	717	692	25	1192	60.2
7207166	AM peak	80	A	1568	1413	155	2301	68.1
7207166	Day period	80	A	3487	3127	360	5328	65.4
7207166	PM peak	80	A	1559	1365	194	2320	67.2
7207166	Evening	80	A	2794	2586	208	4277	65.3
7207166	Saturdays	80	A	1513	1454	59	2050	73.8
7207166	Sundays	80	A	1173	1130	43	1696	69.2
7207166	Other	80	A	652	607	45	1114	58.5
71660522	AM peak	80	A	1580	1463	117	2193	72.0
71660522	Day period	80	A	3363	3019	344	5321	63.2

71660522	PM peak	80	A	1424	1210	214	2294	62.1
71660522	Evening	80	A	2851	2515	336	4411	64.6
71660522	Saturdays	80	A	1449	1366	83	1999	72.5
71660522	Sundays	80	A	979	920	59	1619	60.5
71660522	Other	80	A	672	626	46	1110	60.5
5221298	AM peak	80	A	1558	1446	112	2237	69.6
5221298	Day period	80	A	3376	2971	405	5377	62.8
5221298	PM peak	80	A	1402	1146	256	2226	63.0
5221298	Evening	80	A	2787	2448	339	4390	63.5
5221298	Saturdays	80	A	1467	1381	86	2026	72.4
5221298	Sundays	80	A	996	924	72	1657	60.1
5221298	Other	80	A	601	534	67	1057	56.9
12980122	AM peak	80	A	1441	1352	89	2096	68.8
12980122	Day period	80	A	3386	2981	405	5322	63.6
12980122	PM peak	80	A	1361	1156	205	2072	65.7
12980122	Evening	80	A	3044	2804	240	4413	69.0
12980122	Saturdays	80	A	1466	1397	69	1968	74.5
12980122	Sundays	80	A	1065	969	96	1699	62.7
12980122	Other	80	A	668	613	55	1090	61.3
1220335	AM peak	80	A	1553	1421	132	2215	70.1
1220335	Day period	80	A	3455	2990	465	5239	65.9
1220335	PM peak	80	A	1475	1215	260	2144	68.8
1220335	Evening	80	A	3115	2809	306	4444	70.1
1220335	Saturdays	80	A	1470	1375	95	1960	75.0
1220335	Sundays	80	A	1056	924	132	1680	62.9

1220335	Other	80	A	573	528	45	930	61.6
3353303	AM peak	80	A	1524	1389	135	2178	70.0
3353303	Day period	80	A	3391	2970	421	5162	65.7
3353303	PM peak	80	A	1467	1215	252	2148	68.3
3353303	Evening	80	A	3137	2833	304	4433	70.8
3353303	Saturdays	80	A	1476	1383	93	1942	76.0
3353303	Sundays	80	A	1056	940	116	1673	63.1
3353303	Other	80	A	605	558	47	954	63.4
33031037	AM peak	80	A	1445	1316	129	2059	70.2
33031037	Day period	80	A	3467	3014	453	5203	66.6
33031037	PM peak	80	A	1474	1228	246	2142	68.8
33031037	Evening	80	A	3185	2888	297	4428	71.9
33031037	Saturdays	80	A	1465	1370	95	1933	75.8
33031037	Sundays	80	A	1067	943	124	1688	63.2
33031037	Other	80	A	628	574	54	975	64.4
10371414	AM peak	80	A	1540	1433	107	2081	74.0
10371414	Day period	80	A	3549	3206	343	5100	69.6
10371414	PM peak	80	A	1406	1167	239	2029	69.3
10371414	Evening	80	A	3294	3021	273	4442	74.2
10371414	Saturdays	80	A	1487	1419	68	1903	78.1
10371414	Sundays	80	A	1087	987	100	1665	65.3
10371414	Other	80	A	617	584	33	907	68.0
14140052	AM peak	80	A	1575	1418	157	2168	72.6
14140052	Day period	80	A	3527	3033	494	5152	68.5
14140052	PM peak	80	A	1363	992	371	2001	68.1

14140052	Evening	80	A	3276	2881	395	4534	72.3
14140052	Saturdays	80	A	1471	1386	85	1930	76.2
14140052	Sundays	80	A	1089	956	133	1666	65.4
14140052	Other	80	A	604	553	51	926	65.2
520630	AM peak	80	A	1428	1268	160	2061	69.3
520630	Day period	80	A	3589	2991	598	5204	69.0
520630	PM peak	80	A	1381	983	398	2007	68.8
520630	Evening	80	A	3119	2687	432	4466	69.8
520630	Saturdays	80	A	1415	1290	125	1901	74.4
520630	Sundays	80	A	1063	891	172	1639	64.9
520630	Other	80	A	647	578	69	971	66.6
6302117	AM peak	80	A	1332	1105	227	1997	66.7
6302117	Day period	80	A	3201	2509	692	4865	65.8
6302117	PM peak	80	A	1458	808	650	2048	71.2
6302117	Evening	80	A	2849	2180	669	4147	68.7
6302117	Saturdays	80	A	1315	1163	152	1800	73.1
6302117	Sundays	80	A	1010	810	200	1592	63.4
6302117	Other	80	A	623	532	91	997	62.5
21170782	AM peak	80	A	1434	1120	314	2061	69.6
21170782	Day period	80	A	3470	2520	950	5178	67.0
21170782	PM peak	80	A	1425	843	582	1911	74.6
21170782	Evening	80	A	3111	2260	851	4435	70.1
21170782	Saturdays	80	A	1383	1171	212	1905	72.6
21170782	Sundays	80	A	1143	859	284	1678	68.1
21170782	Other	80	A	667	539	128	1052	63.4

7823301	AM peak	80	A	1493	1218	275	2122	70.4
7823301	Day period	80	A	3479	2706	773	5089	68.4
7823301	PM peak	80	A	1330	930	400	1799	73.9
7823301	Evening	80	A	3192	2534	658	4421	72.2
7823301	Saturdays	80	A	1431	1266	165	1886	75.9
7823301	Sundays	80	A	1112	907	205	1638	67.9
7823301	Other	80	A	589	513	76	895	65.8
33010982	AM peak	80	A	1393	1133	260	1986	70.1
33010982	Day period	80	A	3316	2498	818	4750	69.8
33010982	PM peak	80	A	1260	877	383	1669	75.5
33010982	Evening	80	A	3081	2392	689	4308	71.5
33010982	Saturdays	80	A	1349	1197	152	1773	76.1
33010982	Sundays	80	A	1069	857	212	1569	68.1
33010982	Other	80	A	575	481	94	853	67.4
9826495	AM peak	80	A	1328	1042	286	1895	70.1
9826495	Day period	80	A	3349	2453	896	4753	70.5
9826495	PM peak	80	A	1211	863	348	1591	76.1
9826495	Evening	80	A	3174	2459	715	4389	72.3
9826495	Saturdays	80	A	1324	1162	162	1745	75.9
9826495	Sundays	80	A	1074	847	227	1571	68.4
9826495	Other	80	A	586	470	116	868	67.5
64956497	AM peak	80	A	1373	1028	345	1913	71.8
64956497	Day period	80	A	3272	2375	897	4651	70.4
64956497	PM peak	80	A	1260	902	358	1633	77.2
64956497	Evening	80	A	3196	2492	704	4378	73.0

64956497	Saturdays	80	A	1338	1176	162	1737	77.0
64956497	Sundays	80	A	1085	868	217	1562	69.5
64956497	Other	80	A	581	461	120	863	67.3
64972521	AM peak	80	A	1322	1047	275	1865	70.9
64972521	Day period	80	A	3249	2396	853	4575	71.0
64972521	PM peak	80	A	1310	916	394	1721	76.1
64972521	Evening	80	A	3235	2522	713	4330	74.7
64972521	Saturdays	80	A	1339	1203	136	1703	78.6
64972521	Sundays	80	A	1089	915	174	1530	71.2
64972521	Other	80	A	592	492	100	830	71.3
25217444	AM peak	80	A	1304	1196	108	1668	78.2
25217444	Day period	80	A	3245	2731	514	4350	74.6
25217444	PM peak	80	A	1190	876	314	1512	78.7
25217444	Evening	80	A	3321	2744	577	4225	78.6
25217444	Saturdays	80	A	1284	1226	58	1509	85.1
25217444	Sundays	80	A	1145	1043	102	1453	78.8
25217444	Other	80	A	623	585	38	784	79.5
74442522	AM peak	80	R	1526	1478	48	2441	62.5
74442522	Day period	80	R	4139	4131	8	5070	81.6
74442522	PM peak	80	R	1923	1920	3	2306	83.4
74442522	Evening	80	R	2864	2864	0	3328	86.1
74442522	Saturdays	80	R	1635	1635	0	1977	82.7
74442522	Sundays	80	R	1398	1398	0	1715	81.5
74442522	Other	80	R	1289	1264	25	1962	65.7
25226498	AM peak	80	R	1282	1111	171	2408	53.2

25226498	Day period	80	R	3852	3807	45	5152	74.8
25226498	PM peak	80	R	1900	1883	17	2506	75.8
25226498	Evening	80	R	2744	2734	10	3438	79.8
25226498	Saturdays	80	R	1558	1552	6	2002	77.8
25226498	Sundays	80	R	1291	1288	3	1739	74.2
25226498	Other	80	R	1054	974	80	1868	56.4
64986496	AM peak	80	R	1127	725	402	2415	46.7
64986496	Day period	80	R	3304	3130	174	5274	62.6
64986496	PM peak	80	R	1621	1521	100	2545	63.7
64986496	Evening	80	R	2480	2431	49	3547	69.9
64986496	Saturdays	80	R	1356	1321	35	2039	66.5
64986496	Sundays	80	R	1126	1104	22	1792	62.8
64986496	Other	80	R	908	770	138	1796	50.6
64960983	AM peak	80	R	1115	633	482	2409	46.3
64960983	Day period	80	R	3142	2905	237	5225	60.1
64960983	PM peak	80	R	1519	1370	149	2491	61.0
64960983	Evening	80	R	2362	2271	91	3559	66.4
64960983	Saturdays	80	R	1306	1255	51	2011	64.9
64960983	Sundays	80	R	1065	1026	39	1773	60.1
64960983	Other	80	R	843	643	200	1779	47.4
9833302	AM peak	80	R	1233	717	516	2458	50.2
9833302	Day period	80	R	3038	2693	345	5213	58.3
9833302	PM peak	80	R	1544	1363	181	2515	61.4
9833302	Evening	80	R	2331	2198	133	3680	63.3
9833302	Saturdays	80	R	1285	1215	70	1992	64.5

9833302	Sundays	80	R	1005	940	65	1765	56.9
9833302	Other	80	R	898	621	277	1827	49.2
33020780	AM peak	80	R	1206	791	415	2636	45.8
33020780	Day period	80	R	3266	3068	198	5157	63.3
33020780	PM peak	80	R	1649	1520	129	2532	65.1
33020780	Evening	80	R	2510	2445	65	3628	69.2
33020780	Saturdays	80	R	1395	1359	36	2044	68.2
33020780	Sundays	80	R	1124	1088	36	1785	63.0
33020780	Other	80	R	812	685	127	1698	47.8
7800353	AM peak	80	R	1408	892	516	2322	60.6
7800353	Day period	80	R	2868	2350	518	5291	54.2
7800353	PM peak	80	R	1455	1140	315	2617	55.6
7800353	Evening	80	R	2251	2081	170	3691	61.0
7800353	Saturdays	80	R	1322	1217	105	2084	63.4
7800353	Sundays	80	R	1025	956	69	1805	56.8
7800353	Other	80	R	813	599	214	1740	46.7
3530631	AM peak	80	R	1591	1366	225	2205	72.2
3530631	Day period	80	R	3013	2579	434	5250	57.4
3530631	PM peak	80	R	1371	1149	222	2498	54.9
3530631	Evening	80	R	2286	2137	149	3693	61.9
3530631	Saturdays	80	R	1252	1153	99	2039	61.4
3530631	Sundays	80	R	996	917	79	1778	56.0
3530631	Other	80	R	953	872	81	1706	55.9
6310053	AM peak	80	R	1567	1283	284	2366	66.2
6310053	Day period	80	R	2793	2356	437	5149	54.2

6310053	PM peak	80	R	1246	1001	245	2402	51.9
6310053	Evening	80	R	2324	2176	148	3777	61.5
6310053	Saturdays	80	R	1227	1133	94	2037	60.2
6310053	Sundays	80	R	978	898	80	1740	56.2
6310053	Other	80	R	927	826	101	1690	54.9
531415	AM peak	80	R	1591	1141	450	2507	63.5
531415	Day period	80	R	2899	2461	438	5201	55.7
531415	PM peak	80	R	1309	1081	228	2401	54.5
531415	Evening	80	R	2446	2305	141	3805	64.3
531415	Saturdays	80	R	1288	1206	82	2043	63.0
531415	Sundays	80	R	1031	962	69	1758	58.6
531415	Other	80	R	893	790	103	1568	57.0
14151038	AM peak	80	R	1621	1206	415	2500	64.8
14151038	Day period	80	R	3143	2847	296	5096	61.7
14151038	PM peak	80	R	1474	1297	177	2438	60.5
14151038	Evening	80	R	2663	2552	111	3820	69.7
14151038	Saturdays	80	R	1378	1321	57	2023	68.1
14151038	Sundays	80	R	1139	1083	56	1745	65.3
14151038	Other	80	R	879	784	95	1496	58.8
10383304	AM peak	80	R	1576	1303	273	2373	66.4
10383304	Day period	80	R	3556	3391	165	4851	73.3
10383304	PM peak	80	R	1676	1548	128	2430	69.0
10383304	Evening	80	R	2949	2868	81	3727	79.1
10383304	Saturdays	80	R	1526	1487	39	1989	76.7
10383304	Sundays	80	R	1315	1276	39	1741	75.5

10383304	Other	80	R	997	941	56	1483	67.2
33040334	AM peak	80	R	1385	1000	385	2217	62.5
33040334	Day period	80	R	3434	3178	256	4890	70.2
33040334	PM peak	80	R	1629	1380	249	2521	64.6
33040334	Evening	80	R	2849	2733	116	3748	76.0
33040334	Saturdays	80	R	1487	1428	59	2020	73.6
33040334	Sundays	80	R	1225	1184	41	1727	70.9
33040334	Other	80	R	985	918	67	1540	64.0
3340121	AM peak	80	R	1342	952	390	2123	63.2
3340121	Day period	80	R	3352	3052	300	4822	69.5
3340121	PM peak	80	R	1531	1189	342	2374	64.5
3340121	Evening	80	R	2840	2704	136	3742	75.9
3340121	Saturdays	80	R	1432	1367	65	1971	72.7
3340121	Sundays	80	R	1235	1189	46	1675	73.7
3340121	Other	80	R	1007	945	62	1525	66.0
1211301	AM peak	80	R	1436	1087	349	2249	63.9
1211301	Day period	80	R	3131	2786	345	4662	67.2
1211301	PM peak	80	R	1518	1169	349	2406	63.1
1211301	Evening	80	R	2830	2655	175	3757	75.3
1211301	Saturdays	80	R	1333	1265	68	1850	72.1
1211301	Sundays	80	R	1252	1221	31	1681	74.5
1211301	Other	80	R	893	848	45	1371	65.1
13010523	AM peak	80	R	1316	941	375	2115	62.2
13010523	Day period	80	R	3259	2952	307	4643	70.2
13010523	PM peak	80	R	1433	1079	354	2353	60.9

13010523	Evening	80	R	2903	2731	172	3845	75.5
13010523	Saturdays	80	R	1388	1325	63	1821	76.2
13010523	Sundays	80	R	1240	1182	58	1672	74.2
13010523	Other	80	R	940	895	45	1372	68.5
5237165	AM peak	80	R	1400	683	717	2180	64.2
5237165	Day period	80	R	3009	1935	1074	4825	62.4
5237165	PM peak	80	R	1557	590	967	2483	62.7
5237165	Evening	80	R	2466	1819	647	3817	64.6
5237165	Saturdays	80	R	1241	1001	240	1891	65.6
5237165	Sundays	80	R	1074	902	172	1721	62.4
5237165	Other	80	R	807	639	168	1411	57.2
71650721	AM peak	80	R	1384	1138	246	2084	66.4
71650721	Day period	80	R	3291	2979	312	4440	74.1
71650721	PM peak	80	R	1500	1223	277	2261	66.3
71650721	Evening	80	R	2909	2644	265	3750	77.6
71650721	Saturdays	80	R	1374	1290	84	1770	77.6
71650721	Sundays	80	R	1245	1178	67	1638	76.0
71650721	Other	80	R	976	916	60	1339	72.9
7211023	AM peak	80	R	1371	1077	294	2098	65.3
7211023	Day period	80	R	3216	2752	464	4557	70.6
7211023	PM peak	80	R	1507	1149	358	2262	66.6
7211023	Evening	80	R	2797	2451	346	3791	73.8
7211023	Saturdays	80	R	1349	1214	135	1796	75.1
7211023	Sundays	80	R	1199	1102	97	1633	73.4
7211023	Other	80	R	938	841	97	1334	70.3

10231308	AM peak	80	R	1450	1113	337	2127	68.2
10231308	Day period	80	R	3243	2717	526	4535	71.5
10231308	PM peak	80	R	1611	1167	444	2308	69.8
10231308	Evening	80	R	2713	2297	416	3709	73.1
10231308	Saturdays	80	R	1335	1167	168	1792	74.5
10231308	Sundays	80	R	1176	1075	101	1600	73.5
10231308	Other	80	R	842	747	95	1188	70.9
13082321	AM peak	80	R	1442	1156	286	2097	68.8
13082321	Day period	80	R	3227	2789	438	4405	73.3
13082321	PM peak	80	R	1594	1235	359	2252	70.8
13082321	Evening	80	R	2817	2465	352	3744	75.2
13082321	Saturdays	80	R	1350	1199	151	1787	75.5
13082321	Sundays	80	R	1209	1114	95	1633	74.0
13082321	Other	80	R	877	782	95	1224	71.7

For compact table with one row per segment:

```
import pandas as pd

file_path = '/Users/benfall/Desktop/Data Files/OG_trimmed_with_first_stops.csv'
df = pd.read_csv(file_path, encoding='utf-8', low_memory=False)
df.columns = df.columns.str.strip().str.replace('\ufeff', '', regex=False)

df['EcartDepar'] = pd.to_numeric(df['EcartDepar'], errors='coerce')
df['DTDepartTheo'] = pd.to_datetime(df['DTDepartTheo'], errors='coerce')

def assign_period(dt):
   hour = dt.hour if pd.notnull(dt) else -1
   day = dt.dayofweek if pd.notnull(dt) else -1
   if day == 5:  return "Saturdays"
   if day == 6:  return "Sundays"
   if 7 <= hour < 9:   return "AM peak"
   elif 9 <= hour < 16:return "Day period"
   elif 16 <= hour < 18:return "PM peak"
   elif 18 <= hour < 24:return "Evening"
   else:               return "Other"
df['period'] = df['DTDepartTheo'].apply(assign_period)

# ---- Segment lists ----
segments_18A = [
   '74221075','10751361','13610117','01171296','12960374','03740042','00425315','53150056','00560130',
   '01300071','00711127','11271039','10391063','10630086','00863349','33490422','04220801','08011082',
   '10821238','12381441','14410155','01550091','00910077','00770134','01340702','07023002','30020924',
   '09240686','06860806','08060260'
]
segments_18R = [
   '02610807','08070687','06870925','09253003','30030703','07030137','01370078','00780092','00920156',
   '01561440','14401237','12371081','10810800','08000427','04270356','03560087','00871064','10641041',
   '10411128','11280072','00720131','01310057','00575316','53160043','00430375','03751297','12970118',
```

```
   '01181362','13621077','10777423'
]
segments_80A = [
   '23211310','13101022','10220720','07207166','71660522','05221298','12980122','01220335','03353303',
   '33031037','10371414','14140052','00520630','06302117','21170782','07823301','33010982','09826495',
   '64956497','64972521','25217444'
]
segments_80R = [
   '74442522','25226498','64986496','64960983','09833302','33020780','07800353','03530631','06310053',
   '00531415','14151038','10383304','33040334','03340121','01211301','13010523','05237165','71650721',
   '07211023','10231308','13082321'
]

# Combine for mapping
segment_map = {}
for v in segments_18A: segment_map[v] = (18, "A")
for v in segments_18R: segment_map[v] = (18, "R")
for v in segments_80A: segment_map[v] = (80, "A")
for v in segments_80R: segment_map[v] = (80, "R")
all_segments = segments_18A + segments_18R + segments_80A + segments_80R

valid_mask = (df['EcartDepar'] > -300) & (df['EcartDepar'] < 300)
df_valid = df[valid_mask & df['Segment_XXXXYYYY'].astype(str).isin(all_segments)].copy()

df_valid['delay_pos'] = (df_valid['EcartDepar'] >= 45)
df_valid['delay_neg'] = (df_valid['EcartDepar'] <= -45)
df_valid['substantial_delay'] = df_valid['delay_pos'] | df_valid['delay_neg']
df_valid['line'] = df_valid['Segment_XXXXYYYY'].map(lambda x: segment_map.get(str(x), [None])[0])
df_valid['direction'] = df_valid['Segment_XXXXYYYY'].map(lambda x: segment_map.get(str(x), [None, None])[1])

grouped = df_valid.groupby(['Segment_XXXXYYYY','line','direction','period']).agg(
   num_substantial_delays = ('substantial_delay', 'sum'),
   num_pos_delays         = ('delay_pos', 'sum'),
   num_neg_delays         = ('delay_neg', 'sum'),
   num_valid_trips        = ('EcartDepar', 'count')
).reset_index()

grouped['percent_substantial_delays'] = (100 * grouped['num_substantial_delays'] /
grouped['num_valid_trips']).round(1)

periods = ["AM peak", "Day period", "PM peak", "Evening", "Saturdays", "Sundays", "Other"]
metrics = ['num_substantial_delays','num_valid_trips','percent_substantial_delays']

def build_wide_table(segment_list, line, direction):
   # Subset for this line and direction
   sub = grouped[(grouped['line'] == line) & (grouped['direction'] == direction)].copy()
   # Pivot to wide format: one row per segment
   wide = sub.pivot(index='Segment_XXXXYYYY', columns='period', values=metrics)
   # Flatten columns: metric_period
   wide.columns = [f"{period}_{metric}" for metric, period in wide.columns]
   # Reindex for desired segment order, add line/direction columns
   wide = wide.reindex(segment_list)
   wide['line'] = line
   wide['direction'] = direction
   wide = wide.reset_index()
   # Move segment first as column
   columns = ['Segment_XXXXYYYY','line','direction'] + [col for col in wide.columns if col not in
['Segment_XXXXYYYY','line','direction']]
   wide = wide[columns]
   return wide

table_18A = build_wide_table(segments_18A, 18, "A")
table_18R = build_wide_table(segments_18R, 18, "R")
table_80A = build_wide_table(segments_80A, 80, "A")
table_80R = build_wide_table(segments_80R, 80, "R")

table_18A.to_csv('/Users/benfall/Desktop/delay_summary_segment_wide_18A.csv', index=False)
table_18R.to_csv('/Users/benfall/Desktop/delay_summary_segment_wide_18R.csv', index=False)
table_80A.to_csv('/Users/benfall/Desktop/delay_summary_segment_wide_80A.csv', index=False)
table_80R.to_csv('/Users/benfall/Desktop/delay_summary_segment_wide_80R.csv', index=False)

# For all together if you want:
all_tables_wide = pd.concat([table_18A, table_18R, table_80A, table_80R], ignore_index=True)
all_tables_wide.to_csv('/Users/benfall/Desktop/delay_summary_segment_wide_all.csv', index=False)

print(table_18A.head(10))
```

Appendix 7:

```
import pandas as pd

# ---- Load trimmed file with first stops ----
file_path = '/Users/benfall/Desktop/Data Files/OG_trimmed_with_first_stops.csv'

df = pd.read_csv(file_path, encoding='utf-8', low_memory=False)
df.columns = df.columns.str.strip().str.replace('\ufeff', '', regex=False)

# Convert types
df['Segment_XXXXYYYY'] = df['Segment_XXXXYYYY'].astype(str)
df['segment_speed'] = pd.to_numeric(df['segment_speed'], errors='coerce')
df['DTDepartTheo'] = pd.to_datetime(df['DTDepartTheo'], errors='coerce')

# Assign time periods
def assign_period(dt):
   if pd.isnull(dt):
       return "Other"
   hour = dt.hour
   day = dt.dayofweek  # Monday=0, Sunday=6
   if day == 5:  return "Saturdays"
   if day == 6:  return "Sundays"
   if 7 <= hour < 9:   return "AM peak"
   elif 9 <= hour < 16:return "Day period"
   elif 16 <= hour < 18:return "PM peak"
   elif 18 <= hour < 24:return "Evening"
   else:               return "Other"

df['time_period'] = df['DTDepartTheo'].apply(assign_period)

# Segment lists
line_18_A = ['74221075', '10751361', '13610117', '01171296', '12960374', '03740042', '00425315', '53150056',
            '00560130', '01300071', '00711127', '11271039', '10391063', '10630086', '00863349', '33490422',
            '04220801', '08011082', '10821238', '12381441', '14410155', '01550091', '00910077', '00770134',
            '01340702', '07023002', '30020924', '09240686', '06860806', '08060260']

line_18_R = ['02610807', '08070687', '06870925', '09253003', '30030703', '07030137', '01370078', '00780092',
            '00920156', '01561440', '14401237', '12371081', '10810800', '08000427', '04270356', '03560087',
            '00871064', '10641041', '10411128', '11280072', '00720131', '01310057', '00575316', '53160043',
            '00430375', '03751297', '12970118', '01181362', '13621077', '10777423']

line_80_A = ['23211310', '13101022', '10220720', '07207166', '71660522', '05221298', '12980122', '01220335',
            '03353303', '33031037', '10371414', '14140052', '00520630', '06302117', '21170782', '07823301',
            '33010982', '09826495', '64956497', '64972521', '25217444']

line_80_R = ['74442522', '25226498', '64986496', '64960983', '09833302', '33020780', '07800353', '03530631',
            '06310053', '00531415', '14151038', '10383304', '33040334', '03340121', '01211301', '13010523',
            '05237165', '71650721', '07211023', '10231308', '13082321']

# Calculate average speeds ignoring NaNs
def avg_speed(segment_list, period=None):
   result = {}
   for segment in segment_list:
       seg_str = str(segment)
       if period:
           speeds = df[(df['Segment_XXXXYYYY'] == seg_str) & (df['time_period'] ==
period)]['segment_speed'].dropna()
       else:
           speeds = df[df['Segment_XXXXYYYY'] == seg_str]['segment_speed'].dropna()
       if speeds.empty:
           result[seg_str] = None
       else:
           result[seg_str] = speeds.mean()
   return result

# Calculate overall and Sunday averages
overall_18_A = avg_speed(line_18_A)
sundays_18_A = avg_speed(line_18_A, period='Sundays')

overall_18_R = avg_speed(line_18_R)
sundays_18_R = avg_speed(line_18_R, period='Sundays')

overall_80_A = avg_speed(line_80_A)
sundays_80_A = avg_speed(line_80_A, period='Sundays')
```

```
overall_80_R = avg_speed(line_80_R)
sundays_80_R = avg_speed(line_80_R, period='Sundays')

# Compute and print Sunday speed as % of overall speed
def print_percentage(overall_dict, sunday_dict, label):
   print(f"Sunday average speed as % of overall for {label}:")
   for seg in overall_dict.keys():
       ov = overall_dict[seg]
       su = sunday_dict.get(seg)
       if ov is None or su is None:
           print(f"Segment {seg}: No sufficient data")
       else:
           percent = (su / ov) * 100
           print(f"Segment {seg}: {percent:.1f}%")
   print()

print_percentage(overall_18_A, sundays_18_A, "line 18 direction A")
print_percentage(overall_18_R, sundays_18_R, "line 18 direction R")
print_percentage(overall_80_A, sundays_80_A, "line 80 direction A")
print_percentage(overall_80_R, sundays_80_R, "line 80 direction R")
```

For each segment, we print the Sunday speed as a percentage of the overall speed:

- Values below 100% mean Sunday speeds are slower than average.
- Values around or above 100% mean Sunday speeds are comparable to or faster than the overall average.

Sunday average speed as % of overall for line 18 direction A:
Segment 74221075: 98.9%
Segment 10751361: 100.4%
Segment 13610117: 101.8%
Segment 01171296: 102.9%
Segment 12960374: 103.3%
Segment 03740042: 104.9%
Segment 00425315: 101.3%
Segment 53150056: 101.6%
Segment 00560130: 102.0%
Segment 01300071: 103.4%
Segment 00711127: 104.7%
Segment 11271039: 103.4%
Segment 10391063: 101.1%
Segment 10630086: 108.2%
Segment 00863349: 105.5%
Segment 33490422: 104.0%
Segment 04220801: 100.9%
Segment 08011082: 106.1%
Segment 10821238: 106.0%
Segment 12381441: 100.2%
Segment 14410155: 105.1%
Segment 01550091: 105.3%
Segment 00910077: 103.6%
Segment 00770134: 101.7%
Segment 01340702: 101.5%
Segment 07023002: 100.8%
Segment 30020924: 102.4%
Segment 09240686: 106.6%
Segment 06860806: 101.5%
Segment 08060260: 101.4%

Sunday average speed as % of overall for line 18 direction R:
Segment 02610807: 100.7%
Segment 08070687: 102.0%
Segment 06870925: 105.0%
Segment 09253003: 102.2%
Segment 30030703: 102.2%
Segment 07030137: 101.3%
Segment 01370078: 102.2%
Segment 00780092: 103.5%
Segment 00920156: 105.1%
Segment 01561440: 101.4%

Segment 14401237: 101.8%
Segment 12371081: 107.3%
Segment 10810800: 105.5%
Segment 08000427: 103.9%
Segment 04270356: 102.3%
Segment 03560087: 103.3%
Segment 00871064: 107.7%
Segment 10641041: 101.6%
Segment 10411128: 105.4%
Segment 11280072: 105.7%
Segment 00720131: 101.8%
Segment 01310057: 103.6%
Segment 00575316: 105.1%
Segment 53160043: 102.5%
Segment 00430375: 108.8%
Segment 03751297: 103.7%
Segment 12970118: 101.2%
Segment 01181362: 101.5%
Segment 13621077: 101.3%
Segment 10777423: 101.9%

Sunday average speed as % of overall for line 80 direction A:
Segment 23211310: 115.8%
Segment 13101022: 109.5%
Segment 10220720: 112.7%
Segment 07207166: 110.7%
Segment 71660522: 121.6%
Segment 05221298: 112.8%
Segment 12980122: 107.9%
Segment 01220335: 118.5%
Segment 03353303: 103.1%
Segment 33031037: 108.0%
Segment 10371414: 113.5%
Segment 14140052: 110.5%
Segment 00520630: 119.7%
Segment 06302117: 107.1%
Segment 21170782: 126.5%
Segment 07823301: 121.4%
Segment 33010982: 111.7%
Segment 09826495: 107.5%
Segment 64956497: 106.9%
Segment 64972521: 104.7%
Segment 25217444: 109.8%

Sunday average speed as % of overall for line 80 direction R:
Segment 74442522: 107.2%
Segment 25226498: 106.2%
Segment 64986496: 105.0%
Segment 64960983: 108.5%
Segment 09833302: 116.9%
Segment 33020780: 108.3%
Segment 07800353: 115.6%
Segment 03530631: 123.5%
Segment 06310053: 108.8%
Segment 00531415: 108.6%
Segment 14151038: 116.3%
Segment 10383304: 119.0%
Segment 33040334: 112.0%
Segment 03340121: 111.1%
Segment 01211301: 119.4%
Segment 13010523: 109.4%
Segment 05237165: 100.7%
Segment 71650721: 127.4%
Segment 07211023: 107.5%
Segment 10231308: 105.6%
Segment 13082321: 118.4%

```
import pandas as pd

# ---- Load trimmed file with first stops ----
file_path = '/Users/benfall/Desktop/Data Files/OG_trimmed_with_first_stops.csv'

df = pd.read_csv(file_path, encoding='utf-8', low_memory=False)
df.columns = df.columns.str.strip().str.replace('\ufeff', '', regex=False)
```

```python
# Convert types
df['Segment_XXXXYYYY'] = df['Segment_XXXXYYYY'].astype(str)
df['segment_speed'] = pd.to_numeric(df['segment_speed'], errors='coerce')
df['DTDepartTheo'] = pd.to_datetime(df['DTDepartTheo'], errors='coerce')

# Assign time periods
def assign_period(dt):
   if pd.isnull(dt):
       return "Other"
   hour = dt.hour
   day = dt.dayofweek  # Monday=0, Sunday=6
   if day == 5:  return "Saturdays"
   if day == 6:  return "Sundays"
   if 7 <= hour < 9:   return "AM peak"
   elif 9 <= hour < 16:return "Day period"
   elif 16 <= hour < 18:return "PM peak"
   elif 18 <= hour < 24:return "Evening"
   else:               return "Other"

df['time_period'] = df['DTDepartTheo'].apply(assign_period)

# Segment lists
line_18_A = ['74221075', '10751361', '13610117', '01171296', '12960374', '03740042', '00425315', '53150056',
            '00560130', '01300071', '00711127', '11271039', '10391063', '10630086', '00863349', '33490422',
            '04220801', '08011082', '10821238', '12381441', '14410155', '01550091', '00910077', '00770134',
            '01340702', '07023002', '30020924', '09240686', '06860806', '08060260']

line_18_R = ['02610807', '08070687', '06870925', '09253003', '30030703', '07030137', '01370078', '00780092',
            '00920156', '01561440', '14401237', '12371081', '10810800', '08000427', '04270356', '03560087',
            '00871064', '10641041', '10411128', '11280072', '00720131', '01310057', '00575316', '53160043',
            '00430375', '03751297', '12970118', '01181362', '13621077', '10777423']

line_80_A = ['23211310', '13101022', '10220720', '07207166', '71660522', '05221298', '12980122', '01220335',
            '03353303', '33031037', '10371414', '14140052', '00520630', '06302117', '21170782', '07823301',
            '33010982', '09826495', '64956497', '64972521', '25217444']

line_80_R = ['74442522', '25226498', '64986496', '64960983', '09833302', '33020780', '07800353', '03530631',
            '06310053', '00531415', '14151038', '10383304', '33040334', '03340121', '01211301', '13010523',
            '05237165', '71650721', '07211023', '10231308', '13082321']

# All comparison periods except Sunday itself (which is baseline)
comparison_periods = ['AM peak', 'PM peak', 'Day period', 'Evening']

# Function to compute avg speed per segment & period
def avg_speed(segment_list, period=None):
   result = {}
   for segment in segment_list:
       seg_str = str(segment)
       if period:
           speeds = df[(df['Segment_XXXXYYYY'] == seg_str) & (df['time_period'] ==
period)]['segment_speed'].dropna()
       else:
           speeds = df[df['Segment_XXXXYYYY'] == seg_str]['segment_speed'].dropna()
       if speeds.empty:
           result[seg_str] = None
       else:
           result[seg_str] = speeds.mean()
   return result

# For one line-direction, compute Sunday baseline and comparisons
def compute_and_print_all(segment_list, label):
   # Sunday baseline speeds
   sunday_speeds = avg_speed(segment_list, period='Sundays')

   print(f"Comparisons to Sunday speeds for {label}:\n")
   for comp_period in comparison_periods:
       comp_speeds = avg_speed(segment_list, period=comp_period)
       print(f"--- {comp_period} compared to Sundays ---")
       for seg in segment_list:
           seg_str = str(seg)
           sun_spd = sunday_speeds.get(seg_str)
           comp_spd = comp_speeds.get(seg_str)
           if sun_spd is None or comp_spd is None:
               print(f"Segment {seg_str}: No sufficient data")
           else:
               percent = (comp_spd / sun_spd) * 100
```

```
            print(f"Segment {seg_str}: {percent:.1f}%")
    print()

# Run for all lines/directions
compute_and_print_all(line_18_A, "line 18 direction A")
compute_and_print_all(line_18_R, "line 18 direction R")
compute_and_print_all(line_80_A, "line 80 direction A")
compute_and_print_all(line_80_R, "line 80 direction R")
```

Comparisons to Sunday speeds for line 18 direction A:

--- AM peak compared to Sundays ---
Segment 74221075: 100.5%
Segment 10751361: 97.8%
Segment 13610117: 95.5%
Segment 01171296: 95.3%
Segment 12960374: 93.7%
Segment 03740042: 91.8%
Segment 00425315: 98.9%
Segment 53150056: 99.0%
Segment 00560130: 95.1%
Segment 01300071: 95.6%
Segment 00711127: 94.9%
Segment 11271039: 95.3%
Segment 10391063: 97.9%
Segment 10630086: 89.0%
Segment 00863349: 93.2%
Segment 33490422: 94.0%
Segment 04220801: 98.5%
Segment 08011082: 91.5%
Segment 10821238: 93.3%
Segment 12381441: 98.6%
Segment 14410155: 89.9%
Segment 01550091: 93.1%
Segment 00910077: 94.5%
Segment 00770134: 97.0%
Segment 01340702: 97.1%
Segment 07023002: 97.9%
Segment 30020924: 96.0%
Segment 09240686: 88.3%
Segment 06860806: 98.0%
Segment 08060260: 97.1%

--- PM peak compared to Sundays ---
Segment 74221075: 103.3%
Segment 10751361: 100.2%
Segment 13610117: 98.8%
Segment 01171296: 97.7%
Segment 12960374: 94.6%
Segment 03740042: 91.2%
Segment 00425315: 98.2%
Segment 53150056: 97.5%
Segment 00560130: 96.2%
Segment 01300071: 95.3%
Segment 00711127: 92.4%
Segment 11271039: 95.4%
Segment 10391063: 98.7%
Segment 10630086: 85.7%
Segment 00863349: 87.3%
Segment 33490422: 88.7%
Segment 04220801: 97.5%
Segment 08011082: 85.6%
Segment 10821238: 84.4%
Segment 12381441: 99.6%
Segment 14410155: 91.0%
Segment 01550091: 94.3%
Segment 00910077: 98.0%
Segment 00770134: 99.0%
Segment 01340702: 99.0%
Segment 07023002: 100.6%
Segment 30020924: 93.1%
Segment 09240686: 84.8%
Segment 06860806: 94.1%

Segment 08060260: 98.5%

--- Day period compared to Sundays ---
Segment 74221075: 103.5%
Segment 10751361: 100.1%
Segment 13610117: 98.5%
Segment 01171296: 97.3%
Segment 12960374: 93.7%
Segment 03740042: 93.3%
Segment 00425315: 99.1%
Segment 53150056: 97.7%
Segment 00560130: 97.1%
Segment 01300071: 96.3%
Segment 00711127: 92.1%
Segment 11271039: 94.2%
Segment 10391063: 99.4%
Segment 10630086: 89.3%
Segment 00863349: 91.7%
Segment 33490422: 95.0%
Segment 04220801: 99.3%
Segment 08011082: 90.4%
Segment 10821238: 91.2%
Segment 12381441: 100.1%
Segment 14410155: 93.3%
Segment 01550091: 93.0%
Segment 00910077: 96.3%
Segment 00770134: 99.4%
Segment 01340702: 98.6%
Segment 07023002: 99.9%
Segment 30020924: 98.8%
Segment 09240686: 92.1%
Segment 06860806: 100.5%
Segment 08060260: 98.8%

--- Evening compared to Sundays ---
Segment 74221075: 98.4%
Segment 10751361: 99.2%
Segment 13610117: 98.0%
Segment 01171296: 97.3%
Segment 12960374: 100.8%
Segment 03740042: 96.6%
Segment 00425315: 97.5%
Segment 53150056: 98.3%
Segment 00560130: 98.9%
Segment 01300071: 96.0%
Segment 00711127: 96.7%
Segment 11271039: 98.8%
Segment 10391063: 98.3%
Segment 10630086: 92.9%
Segment 00863349: 95.9%
Segment 33490422: 95.5%
Segment 04220801: 98.5%
Segment 08011082: 93.3%
Segment 10821238: 94.9%
Segment 12381441: 99.8%
Segment 14410155: 94.6%
Segment 01550091: 94.1%
Segment 00910077: 95.0%
Segment 00770134: 96.7%
Segment 01340702: 98.6%
Segment 07023002: 98.7%
Segment 30020924: 96.6%
Segment 09240686: 93.7%
Segment 06860806: 96.7%
Segment 08060260: 97.7%

Comparisons to Sunday speeds for line 18 direction R:

--- AM peak compared to Sundays ---
Segment 02610807: 96.7%
Segment 08070687: 95.9%
Segment 06870925: 92.4%
Segment 09253003: 95.0%
Segment 30030703: 94.7%
Segment 07030137: 97.3%

Segment 01370078: 94.4%
Segment 00780092: 96.3%
Segment 00920156: 92.5%
Segment 01561440: 96.9%
Segment 14401237: 97.9%
Segment 12371081: 90.6%
Segment 10810800: 92.9%
Segment 08000427: 95.5%
Segment 04270356: 98.9%
Segment 03560087: 94.9%
Segment 00871064: 89.9%
Segment 10641041: 100.0%
Segment 10411128: 93.8%
Segment 11280072: 95.6%
Segment 00720131: 96.1%
Segment 01310057: 95.9%
Segment 00575316: 96.7%
Segment 53160043: 95.9%
Segment 00430375: 92.1%
Segment 03751297: 91.9%
Segment 12970118: 97.9%
Segment 01181362: 97.0%
Segment 13621077: 96.6%
Segment 10777423: 95.9%

--- PM peak compared to Sundays ---
Segment 02610807: 100.3%
Segment 08070687: 96.0%
Segment 06870925: 86.4%
Segment 09253003: 96.5%
Segment 30030703: 97.4%
Segment 07030137: 99.3%
Segment 01370078: 97.9%
Segment 00780092: 95.8%
Segment 00920156: 92.6%
Segment 01561440: 99.2%
Segment 14401237: 95.1%
Segment 12371081: 87.4%
Segment 10810800: 88.5%
Segment 08000427: 91.3%
Segment 04270356: 90.8%
Segment 03560087: 93.4%
Segment 00871064: 87.7%
Segment 10641041: 95.9%
Segment 10411128: 92.2%
Segment 11280072: 90.0%
Segment 00720131: 98.9%
Segment 01310057: 92.1%
Segment 00575316: 90.2%
Segment 53160043: 95.2%
Segment 00430375: 74.0%
Segment 03751297: 92.4%
Segment 12970118: 99.7%
Segment 01181362: 98.5%
Segment 13621077: 98.5%
Segment 10777423: 98.7%

--- Day period compared to Sundays ---
Segment 02610807: 99.9%
Segment 08070687: 98.7%
Segment 06870925: 93.4%
Segment 09253003: 98.0%
Segment 30030703: 98.6%
Segment 07030137: 99.2%
Segment 01370078: 98.4%
Segment 00780092: 95.1%
Segment 00920156: 94.0%
Segment 01561440: 99.5%
Segment 14401237: 98.8%
Segment 12371081: 89.9%
Segment 10810800: 93.2%
Segment 08000427: 95.2%
Segment 04270356: 97.2%
Segment 03560087: 96.1%
Segment 00871064: 90.0%

Segment 10641041: 97.7%
Segment 10411128: 92.5%
Segment 11280072: 91.9%
Segment 00720131: 99.1%
Segment 01310057: 94.0%
Segment 00575316: 91.7%
Segment 53160043: 96.7%
Segment 00430375: 90.5%
Segment 03751297: 93.9%
Segment 12970118: 100.4%
Segment 01181362: 99.4%
Segment 13621077: 99.3%
Segment 10777423: 99.6%

--- Evening compared to Sundays ---
Segment 02610807: 99.0%
Segment 08070687: 97.3%
Segment 06870925: 96.2%
Segment 09253003: 97.8%
Segment 30030703: 96.8%
Segment 07030137: 98.6%
Segment 01370078: 98.2%
Segment 00780092: 96.7%
Segment 00920156: 95.7%
Segment 01561440: 97.9%
Segment 14401237: 101.8%
Segment 12371081: 94.1%
Segment 10810800: 93.9%
Segment 08000427: 94.6%
Segment 04270356: 96.2%
Segment 03560087: 96.7%
Segment 00871064: 93.2%
Segment 10641041: 98.5%
Segment 10411128: 95.7%
Segment 11280072: 96.2%
Segment 00720131: 97.4%
Segment 01310057: 97.6%
Segment 00575316: 96.3%
Segment 53160043: 98.1%
Segment 00430375: 91.6%
Segment 03751297: 99.8%
Segment 12970118: 97.3%
Segment 01181362: 97.2%
Segment 13621077: 98.1%
Segment 10777423: 96.7%

Comparisons to Sunday speeds for line 80 direction A:

--- AM peak compared to Sundays ---
Segment 23211310: 83.8%
Segment 13101022: 91.7%
Segment 10220720: 86.9%
Segment 07207166: 88.6%
Segment 71660522: 68.2%
Segment 05221298: 91.0%
Segment 12980122: 95.4%
Segment 01220335: 80.9%
Segment 03353303: 94.7%
Segment 33031037: 90.5%
Segment 10371414: 75.3%
Segment 14140052: 89.7%
Segment 00520630: 77.9%
Segment 06302117: 92.0%
Segment 21170782: 70.0%
Segment 07823301: 81.4%
Segment 33010982: 89.1%
Segment 09826495: 96.8%
Segment 64956497: 100.4%
Segment 64972521: 95.2%
Segment 25217444: 87.5%

--- PM peak compared to Sundays ---
Segment 23211310: 74.3%
Segment 13101022: 80.8%
Segment 10220720: 81.5%

Segment 07207166: 85.0%
Segment 71660522: 81.1%
Segment 05221298: 75.2%
Segment 12980122: 77.0%
Segment 01220335: 68.7%
Segment 03353303: 92.6%
Segment 33031037: 86.8%
Segment 10371414: 78.1%
Segment 14140052: 73.8%
Segment 00520630: 61.3%
Segment 06302117: 94.4%
Segment 21170782: 50.6%
Segment 07823301: 48.1%
Segment 33010982: 68.8%
Segment 09826495: 76.9%
Segment 64956497: 77.1%
Segment 64972521: 89.0%
Segment 25217444: 76.6%

--- Day period compared to Sundays ---
Segment 23211310: 82.7%
Segment 13101022: 88.4%
Segment 10220720: 85.1%
Segment 07207166: 86.8%
Segment 71660522: 76.5%
Segment 05221298: 85.0%
Segment 12980122: 94.5%
Segment 01220335: 80.3%
Segment 03353303: 95.8%
Segment 33031037: 89.0%
Segment 10371414: 84.6%
Segment 14140052: 88.4%
Segment 00520630: 82.2%
Segment 06302117: 85.3%
Segment 21170782: 79.6%
Segment 07823301: 82.3%
Segment 33010982: 88.9%
Segment 09826495: 96.6%
Segment 64956497: 96.9%
Segment 64972521: 95.5%
Segment 25217444: 90.9%

--- Evening compared to Sundays ---
Segment 23211310: 85.7%
Segment 13101022: 91.7%
Segment 10220720: 86.8%
Segment 07207166: 91.5%
Segment 71660522: 86.9%
Segment 05221298: 90.8%
Segment 12980122: 91.5%
Segment 01220335: 86.8%
Segment 03353303: 98.4%
Segment 33031037: 94.2%
Segment 10371414: 93.4%
Segment 14140052: 92.7%
Segment 00520630: 86.7%
Segment 06302117: 97.1%
Segment 21170782: 79.0%
Segment 07823301: 83.3%
Segment 33010982: 90.3%
Segment 09826495: 88.1%
Segment 64956497: 87.5%
Segment 64972521: 94.9%
Segment 25217444: 92.8%

Comparisons to Sunday speeds for line 80 direction R:

--- AM peak compared to Sundays ---
Segment 74442522: 90.0%
Segment 25226498: 91.9%
Segment 64986496: 91.3%
Segment 64960983: 78.2%
Segment 09833302: 68.8%
Segment 33020780: 89.0%
Segment 07800353: 56.8%

Segment 03530631: 33.6%
Segment 06310053: 87.6%
Segment 00531415: 89.2%
Segment 14151038: 73.2%
Segment 10383304: 84.9%
Segment 33040334: 81.8%
Segment 03340121: 71.7%
Segment 01211301: 65.3%
Segment 13010523: 74.5%
Segment 05237165: 91.7%
Segment 71650721: 68.6%
Segment 07211023: 91.7%
Segment 10231308: 93.2%
Segment 13082321: 81.1%

--- PM peak compared to Sundays ---
Segment 74442522: 86.3%
Segment 25226498: 89.8%
Segment 64986496: 92.6%
Segment 64960983: 92.7%
Segment 09833302: 75.9%
Segment 33020780: 84.1%
Segment 07800353: 79.2%
Segment 03530631: 91.4%
Segment 06310053: 86.5%
Segment 00531415: 82.9%
Segment 14151038: 78.1%
Segment 10383304: 69.9%
Segment 33040334: 77.3%
Segment 03340121: 88.1%
Segment 01211301: 81.8%
Segment 13010523: 88.6%
Segment 05237165: 98.4%
Segment 71650721: 61.5%
Segment 07211023: 83.7%
Segment 10231308: 91.6%
Segment 13082321: 71.5%

--- Day period compared to Sundays ---
Segment 74442522: 90.9%
Segment 25226498: 97.0%
Segment 64986496: 99.4%
Segment 64960983: 93.2%
Segment 09833302: 88.0%
Segment 33020780: 94.1%
Segment 07800353: 89.5%
Segment 03530631: 79.2%
Segment 06310053: 90.7%
Segment 00531415: 91.8%
Segment 14151038: 82.3%
Segment 10383304: 71.9%
Segment 33040334: 88.5%
Segment 03340121: 87.8%
Segment 01211301: 77.2%
Segment 13010523: 90.0%
Segment 05237165: 99.3%
Segment 71650721: 72.7%
Segment 07211023: 92.6%
Segment 10231308: 93.1%
Segment 13082321: 79.8%

--- Evening compared to Sundays ---
Segment 74442522: 96.3%
Segment 25226498: 94.9%
Segment 64986496: 94.5%
Segment 64960983: 95.9%
Segment 09833302: 94.0%
Segment 33020780: 92.7%
Segment 07800353: 93.5%
Segment 03530631: 100.1%
Segment 06310053: 94.3%
Segment 00531415: 93.0%
Segment 14151038: 91.4%
Segment 10383304: 90.1%
Segment 33040334: 91.4%

Segment 03340121: 96.5%
Segment 01211301: 95.4%
Segment 13010523: 96.7%
Segment 05237165: 101.6%
Segment 71650721: 80.6%
Segment 07211023: 92.2%
Segment 10231308: 94.4%
Segment 13082321: 84.2%

If the percentage value for a segment's average speed during a comparison period (e.g., AM peak, PM peak, Day period, or Evening) relative to Sundays is more than 100%, it means:

- The average speed during that comparison period is higher than the average speed on Sundays for that segment.
- In other words, transit is moving faster on that segment during that time period compared to Sundays.

So percentages above 100 indicate relatively better/faster travel speeds compared to Sunday baseline, and below 100 indicate slower travel speeds.

```
import pandas as pd

# ---- Load & prepare data ----
file_path = '/Users/benfall/Desktop/Data Files/OG_trimmed_with_first_stops.csv'
df = pd.read_csv(file_path, encoding='utf-8', low_memory=False)
df.columns = df.columns.str.strip().str.replace('\ufeff', '', regex=False)

# Convert datatypes
df['Segment_XXXXYYYY'] = df['Segment_XXXXYYYY'].astype(str)
df['segment_speed'] = pd.to_numeric(df['segment_speed'], errors='coerce')
df['DTDepartTheo'] = pd.to_datetime(df['DTDepartTheo'], errors='coerce')

# ---- Assign time periods ----
def assign_period(dt):
   if pd.isnull(dt):
       return "Other", -1
   hour = dt.hour
   day = dt.dayofweek  # Monday=0, Sunday=6
   if day == 5:  return "Saturdays", 5
   if day == 6:  return "Sundays", 6
   if 7 <= hour < 9:   return "AM peak", day
   elif 9 <= hour < 16:return "Day period", day
   elif 16 <= hour < 18:return "PM peak", day
   elif 18 <= hour < 24:return "Evening", day
   else:               return "Other", day

df['time_period'], df['weekday_num'] = zip(*df['DTDepartTheo'].apply(assign_period))

# ---- Line segment lists ----
line_18_A = ['74221075', '10751361', '13610117', '01171296', '12960374', '03740042', '00425315', '53150056',
            '00560130', '01300071', '00711127', '11271039', '10391063', '10630086', '00863349', '33490422',
            '04220801', '08011082', '10821238', '12381441', '14410155', '01550091', '00910077', '00770134',
            '01340702', '07023002', '30020924', '09240686', '06860806', '08060260']
line_18_R = ['02610807', '08070687', '06870925', '09253003', '30030703', '07030137', '01370078', '00780092',
            '00920156', '01561440', '14401237', '12371081', '10810800', '08000427', '04270356', '03560087',
            '00871064', '10641041', '10411128', '11280072', '00720131', '01310057', '00575316', '53160043',
            '00430375', '03751297', '12970118', '01181362', '13621077', '10777423']
line_80_A = ['23211310', '13101022', '10220720', '07207166', '71660522', '05221298', '12980122', '01220335',
            '03353303', '33031037', '10371414', '14140052', '00520630', '06302117', '21170782', '07823301',
            '33010982', '09826495', '64956497', '64972521', '25217444']
line_80_R = ['74442522', '25226498', '64986496', '64960983', '09833302', '33020780', '07800353', '03530631',
            '06310053', '00531415', '14151038', '10383304', '33040334', '03340121', '01211301', '13010523',
            '05237165', '71650721', '07211023', '10231308', '13082321']

lines = {
   "line 18 direction A": line_18_A,
   "line 18 direction R": line_18_R,
```

```
   "line 80 direction A": line_80_A,
   "line 80 direction R": line_80_R,
}

# ---- Precompute averages ----
# Groupby reduces expensive filtering later
grouped = (
   df.groupby(["Segment_XXXXYYYY", "time_period", "weekday_num"])["segment_speed"]
     .mean()
     .reset_index()
)

# Sunday averages (all time periods combined)
sunday_avg = (
   df[df['weekday_num'] == 6]
     .groupby("Segment_XXXXYYYY")["segment_speed"]
     .mean()
)

# Weekday subsets (Mon-Fri)
df_weekdays = df[df['weekday_num'].isin([0,1,2,3,4])]

# Build all comparisons
comparison_periods = [
   ("All Weekday", None),
   ("Weekday AM peak", "AM peak"),
   ("Weekday Day period", "Day period"),
   ("Weekday PM peak", "PM peak"),
   ("Weekday Evening", "Evening"),
   ("Weekday Other", "Other"),
]

def compute_comparisons(segment_list, label):
   results = []
   for seg in segment_list:
       sunday_speed = sunday_avg.get(seg, None)
       for name, tp in comparison_periods:
           if tp is None:
               wk_speeds = df_weekdays[df_weekdays["Segment_XXXXYYYY"] == seg]["segment_speed"].dropna()
           else:
               wk_speeds = df_weekdays[
                   (df_weekdays["Segment_XXXXYYYY"] == seg) &
                   (df_weekdays["time_period"] == tp)
               ]["segment_speed"].dropna()

           wk_speed = wk_speeds.mean() if not wk_speeds.empty else None

           if sunday_speed is None or wk_speed is None or sunday_speed == 0:
               pct = None
           else:
               pct = (wk_speed / sunday_speed) * 100

           results.append({
               "line": label,
               "segment": seg,
               "comparison_period": name,
               "weekday_avg_speed": wk_speed,
               "sunday_avg_speed": sunday_speed,
               "percent_vs_sunday": pct
           })
   return pd.DataFrame(results)

# ---- Run for all lines ----
all_results = pd.concat([compute_comparisons(segs, label) for label, segs in lines.items()],
                        ignore_index=True)

# Example: display first rows
print(all_results.head(20))

# Optionally save for analysis
all_results.to_csv("weekday_vs_sunday_results.csv", index=False)
```

line	segment	comparison_period	weekday_avg_speed	sunday_avg_speed	percent_vs_sunday
line 18 direction A	74221075	All Weekday	23.969727831815900	23.68976049555070	101.18180737334900
line 18 direction A	74221075	Weekday AM peak	23.809358883453100	23.68976049555070	100.50485266799100
line 18 direction A	74221075	Weekday Day period	24.527843108192100	23.68976049555070	103.53774202486700
line 18 direction A	74221075	Weekday PM peak	24.464439859347700	23.68976049555070	103.27010213523500
line 18 direction A	74221075	Weekday Evening	23.320904206720000	23.68976049555070	98.44297164211530
line 18 direction A	74221075	Weekday Other	23.235673782678000	23.68976049555070	98.08319415910510
line 18 direction A	10751361	All Weekday	19.761491946497400	19.877834318232600	99.41471304231280
line 18 direction A	10751361	Weekday AM peak	19.446758697127700	19.877834318232600	97.83137531884170
line 18 direction A	10751361	Weekday Day period	19.898710872768300	19.877834318232600	100.10502429088300
line 18 direction A	10751361	Weekday PM peak	19.913049897246800	19.877834318232600	100.17716003891800
line 18 direction A	10751361	Weekday Evening	19.718857468396600	19.877834318232600	99.20023053170230
line 18 direction A	10751361	Weekday Other	19.588343508759600	19.877834318232600	98.54365015404370
line 18 direction A	13610117	All Weekday	19.074542161922500	19.49581281792220	97.83917367316760
line 18 direction A	13610117	Weekday AM peak	18.615573507951900	19.49581281792220	95.48498275916440
line 18 direction A	13610117	Weekday Day period	19.211460300147200	19.49581281792220	98.54146877367640
line 18 direction A	13610117	Weekday PM peak	19.26731557648130	19.49581281792220	98.82796760732760
line 18 direction A	13610117	Weekday Evening	19.097420167279200	19.49581281792220	97.95652197544320
line 18 direction A	13610117	Weekday Other	18.87531606063800	19.49581281792220	96.81728193084740
line 18 direction A	1171296	All Weekday	19.993131410163100	20.721224845644200	96.48624325586580
line 18 direction A	1171296	Weekday AM peak	19.74535131386500	20.721224845644200	95.29046405775450
line 18 direction A	1171296	Weekday Day period	20.166943644820500	20.721224845644200	97.32505580653390
line 18 direction A	1171296	Weekday PM peak	20.247144571024400	20.721224845644200	97.71210303371890
line 18 direction A	1171296	Weekday Evening	20.163040043739100	20.721224845644200	97.30621714660660
line 18 direction A	1171296	Weekday Other	19.12207176730220	20.721224845644200	92.2825359492291
line 18 direction A	12960374	All Weekday	28.699342778581400	29.853962992413100	96.13243905298230

line 18 direction A	12960374	Weekday AM peak	27.96339325348140	29.853962992413100	93.66727379071210
line 18 direction A	12960374	Weekday Day period	27.986114742287600	29.853962992413100	93.74338257671120
line 18 direction A	12960374	Weekday PM peak	28.232012030587800	29.853962992413100	94.56704973394150
line 18 direction A	12960374	Weekday Evening	30.08059408562900	29.853962992413100	100.7591323579840
line 18 direction A	12960374	Weekday Other	29.4913440060634	29.853962992413100	98.78535728592600
line 18 direction A	3740042	All Weekday	19.964463963014400	21.14977544738580	94.39563087882410
line 18 direction A	3740042	Weekday AM peak	19.424093230037100	21.14977544738580	91.84065938836270
line 18 direction A	3740042	Weekday Day period	19.738830041445900	21.14977544738580	93.32879249971290
line 18 direction A	3740042	Weekday PM peak	19.288798907709900	21.14977544738580	91.2009631293465
line 18 direction A	3740042	Weekday Evening	20.43841451826440	21.14977544738580	96.63655564149570
line 18 direction A	3740042	Weekday Other	20.991505700205600	21.14977544738580	99.25167173725370
line 18 direction A	425315	All Weekday	20.789743719299900	21.084751983710900	98.60084546102860
line 18 direction A	425315	Weekday AM peak	20.857457255036300	21.084751983710900	98.92199477209790
line 18 direction A	425315	Weekday Day period	20.890799578954000	21.084751983710900	99.08012954144930
line 18 direction A	425315	Weekday PM peak	20.710269767314000	21.084751983710900	98.22391927262810
line 18 direction A	425315	Weekday Evening	20.55361392923240	21.084751983710900	97.48093762312790
line 18 direction A	425315	Weekday Other	20.93155693827320	21.084751983710900	99.27343207282690
line 18 direction A	53150056	All Weekday	20.436772732329300	20.784829705210200	98.32542783454390
line 18 direction A	53150056	Weekday AM peak	20.58584255452410	20.784829705210200	99.04263275904440
line 18 direction A	53150056	Weekday Day period	20.307300826505100	20.784829705210200	97.70251243104760
line 18 direction A	53150056	Weekday PM peak	20.273847745419900	20.784829705210200	97.54156292335580
line 18 direction A	53150056	Weekday Evening	20.4352471539576	20.784829705210200	98.31808796987670
line 18 direction A	53150056	Weekday Other	20.856453061404400	20.784829705210200	100.3445943854730
line 18 direction A	560130	All Weekday	17.448644523169500	17.886438054916800	97.5523716326124
line 18 direction A	560130	Weekday AM peak	17.007701142184300	17.886438054916800	95.08713299968120
line 18 direction A	560130	Weekday Day period	17.373647012816400	17.886438054916800	97.13307344633990

line 18 direction A	560130	Weekday PM peak	17.201881960935200	17.886438054916800	96.17276457235460
line 18 direction A	560130	Weekday Evening	17.681188935184100	17.886438054916800	98.85248745947870
line 18 direction A	560130	Weekday Other	17.918336465318700	17.886438054916800	100.17833852835300
line 18 direction A	1300071	All Weekday	25.473111487262200	26.54929598634040	95.94646690582000
line 18 direction A	1300071	Weekday AM peak	25.392597371740500	26.54929598634040	95.64320419194930
line 18 direction A	1300071	Weekday Day period	25.56728901055190	26.54929598634040	96.30119391379060
line 18 direction A	1300071	Weekday PM peak	25.311050064075100	26.54929598634040	95.33604987905380
line 18 direction A	1300071	Weekday Evening	25.499678218793600	26.54929598634040	96.04653257816430
line 18 direction A	1300071	Weekday Other	25.380190090446400	26.54929598634040	95.59647119646630
line 18 direction A	711127	All Weekday	26.99885152703750	28.581396675298900	94.4630237414915
line 18 direction A	711127	Weekday AM peak	27.113088341629700	28.581396675298900	94.8627131474715
line 18 direction A	711127	Weekday Day period	26.335295402987200	28.581396675298900	92.14138728828170
line 18 direction A	711127	Weekday PM peak	26.418521245445300	28.581396675298900	92.43257614585760
line 18 direction A	711127	Weekday Evening	27.646257328775600	28.581396675298900	96.72815378077220
line 18 direction A	711127	Weekday Other	28.174491402817100	28.581396675298900	98.57632824209250
line 18 direction A	11271039	All Weekday	24.814920721925800	25.839177756737300	96.03603085030670
line 18 direction A	11271039	Weekday AM peak	24.619171660775000	25.839177756737300	95.27846393779250
line 18 direction A	11271039	Weekday Day period	24.331770873427800	25.839177756737300	94.16619639564020
line 18 direction A	11271039	Weekday PM peak	24.662450841867200	25.839177756737300	95.44595835847270
line 18 direction A	11271039	Weekday Evening	25.52212312622580	25.839177756737300	98.77296935105120
line 18 direction A	11271039	Weekday Other	25.321617018927400	25.839177756737300	97.99699223139980
line 18 direction A	10391063	All Weekday	19.332167698612800	19.600102961348600	98.63299053446730
line 18 direction A	10391063	Weekday AM peak	19.18941833076450	19.600102961348600	97.90468125910370
line 18 direction A	10391063	Weekday Day period	19.483709826110700	19.600102961348600	99.40616059279180
line 18 direction A	10391063	Weekday PM peak	19.34756598635530	19.600102961348600	98.71155281433320
line 18 direction A	10391063	Weekday Evening	19.273864382296000	19.600102961348600	98.33552619751080

line 18 direction A	10391063	Weekday Other	19.09299775743320	19.600102961348600	97.41274214265440
line 18 direction A	10630086	All Weekday	15.550869435459700	17.142095307569700	90.71743655860250
line 18 direction A	10630086	Weekday AM peak	15.255185135105900	17.142095307569700	88.99253481789600
line 18 direction A	10630086	Weekday Day period	15.30874632688220	17.142095307569700	89.30498898884360
line 18 direction A	10630086	Weekday PM peak	14.689865659285500	17.142095307569700	85.69469131815360
line 18 direction A	10630086	Weekday Evening	15.92557792335380	17.142095307569700	92.90333321342150
line 18 direction A	10630086	Weekday Other	16.686384374064600	17.142095307569700	97.34156807946430
line 18 direction A	863349	All Weekday	19.311435175418600	20.58803264241090	93.799322697971
line 18 direction A	863349	Weekday AM peak	19.185804061130700	20.58803264241090	93.18910842218260
line 18 direction A	863349	Weekday Day period	18.873228274563900	20.58803264241090	91.67086822897990
line 18 direction A	863349	Weekday PM peak	17.963721228341200	20.58803264241090	87.25321909261160
line 18 direction A	863349	Weekday Evening	19.73400427868400	20.58803264241090	95.85182140246030
line 18 direction A	863349	Weekday Other	21.290145087779100	20.58803264241090	103.41029401673800
line 18 direction A	33490422	All Weekday	9.391629035009280	9.878792192458760	95.06859595830590
line 18 direction A	33490422	Weekday AM peak	9.283928978006290	9.878792192458760	93.97838113340850
line 18 direction A	33490422	Weekday Day period	9.382917653496820	9.878792192458760	94.98041330051990
line 18 direction A	33490422	Weekday PM peak	8.764259009061030	9.878792192458760	88.71792055461460
line 18 direction A	33490422	Weekday Evening	9.438183542975680	9.878792192458760	95.53985304175720
line 18 direction A	33490422	Weekday Other	10.050202834539200	9.878792192458760	101.73513764376300
line 18 direction A	4220801	All Weekday	15.214198097549400	15.393133079299300	98.83756620027850
line 18 direction A	4220801	Weekday AM peak	15.159338350019500	15.393133079299300	98.48117515729030
line 18 direction A	4220801	Weekday Day period	15.2793311663632	15.393133079299300	99.26069688120160
line 18 direction A	4220801	Weekday PM peak	15.008466206450200	15.393133079299300	97.50104887115940
line 18 direction A	4220801	Weekday Evening	15.161543411735400	15.393133079299300	98.49550012742170
line 18 direction A	4220801	Weekday Other	15.339768759869000	15.393133079299300	99.65332386100120
line 18 direction A	8011082	All Weekday	21.818344070542900	23.477851933900100	92.93160265245110

line 18 direction A	8011082	Weekday AM peak	21.493902564404900	23.477851933900100	91.54969809384250
line 18 direction A	8011082	Weekday Day period	21.233852515952800	23.477851933900100	90.44205822464050
line 18 direction A	8011082	Weekday PM peak	20.096459645230800	23.477851933900100	85.59752272827430
line 18 direction A	8011082	Weekday Evening	21.911115256466500	23.477851933900100	93.32674606755050
line 18 direction A	8011082	Weekday Other	24.222016865293500	23.477851933900100	103.16964658218500
line 18 direction A	10821238	All Weekday	31.659254743113000	33.959567122360600	93.22632007952490
line 18 direction A	10821238	Weekday AM peak	31.696631577943400	33.959567122360600	93.33638283355150
line 18 direction A	10821238	Weekday Day period	30.975907670738800	33.959567122360600	91.21408279183520
line 18 direction A	10821238	Weekday PM peak	28.650272028535700	33.959567122360600	84.36583400873490
line 18 direction A	10821238	Weekday Evening	32.224850701875600	33.959567122360600	94.89181821948840
line 18 direction A	10821238	Weekday Other	34.1425555126454	33.959567122360600	100.53884193996200
line 18 direction A	12381441	All Weekday	38.58594819817810	38.72985837457660	99.62842576131650
line 18 direction A	12381441	Weekday AM peak	38.176165124590500	38.72985837457660	98.57037109552250
line 18 direction A	12381441	Weekday Day period	38.77071952879590	38.72985837457660	100.10550297866900
line 18 direction A	12381441	Weekday PM peak	38.58053088115230	38.72985837457660	99.61443831789900
line 18 direction A	12381441	Weekday Evening	38.65076701098560	38.72985837457660	99.79578710867950
line 18 direction A	12381441	Weekday Other	38.33442632289090	38.72985837457660	98.97899948958950
line 18 direction A	14410155	All Weekday	34.55336611774280	36.723566742979400	94.0904415945615
line 18 direction A	14410155	Weekday AM peak	33.00201510648870	36.723566742979400	89.86603980344010
line 18 direction A	14410155	Weekday Day period	34.267543548305800	36.723566742979400	93.3121332906938
line 18 direction A	14410155	Weekday PM peak	33.41406550398840	36.723566742979400	90.9880724218498
line 18 direction A	14410155	Weekday Evening	34.72276458210380	36.723566742979400	94.55172158282240
line 18 direction A	14410155	Weekday Other	36.82016970058240	36.723566742979400	100.26305439849900
line 18 direction A	1550091	All Weekday	33.93070608883620	36.26312514219040	93.56806937017
line 18 direction A	1550091	Weekday AM peak	33.77193314144250	36.26312514219040	93.1302335610081
line 18 direction A	1550091	Weekday Day period	33.71171363373860	36.26312514219040	92.96417090791950

line 18 direction A	1550091	Weekday PM peak	34.188072186319000	36.26312514219040	94.27778784168500
line 18 direction A	1550091	Weekday Evening	34.110778538337900	36.26312514219040	94.06464115982010
line 18 direction A	1550091	Weekday Other	34.02129967640570	36.26312514219040	93.81789226109370
line 18 direction A	910077	All Weekday	33.093325428102	34.65186014930230	95.50230575073000
line 18 direction A	910077	Weekday AM peak	32.75114635516100	34.65186014930230	94.51482897035890
line 18 direction A	910077	Weekday Day period	33.36659069069690	34.65186014930230	96.2909077519428
line 18 direction A	910077	Weekday PM peak	33.97381236248190	34.65186014930230	98.04325717609720
line 18 direction A	910077	Weekday Evening	32.9321392740756	34.65186014930230	95.03714701659020
line 18 direction A	910077	Weekday Other	32.422611110334900	34.65186014930230	93.56672620355070
line 18 direction A	770134	All Weekday	42.594489290531600	43.49871805073590	97.92125193402340
line 18 direction A	770134	Weekday AM peak	42.21169349641600	43.49871805073590	97.04123566855750
line 18 direction A	770134	Weekday Day period	43.23248978587560	43.49871805073590	99.3879629635296
line 18 direction A	770134	Weekday PM peak	43.0462958695181	43.49871805073590	98.9599183573867
line 18 direction A	770134	Weekday Evening	42.05466661278470	43.49871805073590	96.68024368840720
line 18 direction A	770134	Weekday Other	42.10650919721040	43.49871805073590	96.79942555571010
line 18 direction A	1340702	All Weekday	36.798417975644200	37.47952676901080	98.18271773396630
line 18 direction A	1340702	Weekday AM peak	36.37683676308460	37.47952676901080	97.05788706265650
line 18 direction A	1340702	Weekday Day period	36.9545543409898	37.47952676901080	98.59930881396560
line 18 direction A	1340702	Weekday PM peak	37.08741351277330	37.47952676901080	98.95379347062190
line 18 direction A	1340702	Weekday Evening	36.96260929312980	37.47952676901080	98.62080041974190
line 18 direction A	1340702	Weekday Other	35.882625690048200	37.47952676901080	95.7392709657077
line 18 direction A	7023002	All Weekday	31.53551543510120	31.860863900975600	98.97884606366740
line 18 direction A	7023002	Weekday AM peak	31.190482681287200	31.860863900975600	97.8959100990734
line 18 direction A	7023002	Weekday Day period	31.842856663708200	31.860863900975600	99.94348164154190
line 18 direction A	7023002	Weekday PM peak	32.04770187741370	31.860863900975600	100.5864184254980
line 18 direction A	7023002	Weekday Evening	31.45812322503440	31.860863900975600	98.73593924762080

line 18 direction A	7023002	Weekday Other	30.519089578254800	31.860863900975600	95.78864425368040
line 18 direction A	30020924	All Weekday	37.51819097458460	38.67622508532090	97.00582435803480
line 18 direction A	30020924	Weekday AM peak	37.12307341187580	38.67622508532090	95.98422113321870
line 18 direction A	30020924	Weekday Day period	38.22867159651460	38.67622508532090	98.84282013609400
line 18 direction A	30020924	Weekday PM peak	36.02390300672740	38.67622508532090	93.14224158965250
line 18 direction A	30020924	Weekday Evening	37.375536477635700	38.67622508532090	96.63698149233570
line 18 direction A	30020924	Weekday Other	37.36267637558160	38.67622508532090	96.6037308272936
line 18 direction A	9240686	All Weekday	24.743779568125300	26.869989973230100	92.08704429282210
line 18 direction A	9240686	Weekday AM peak	23.72815036914280	26.869989973230100	88.30725427431320
line 18 direction A	9240686	Weekday Day period	24.736262930452100	26.869989973230100	92.05907019353640
line 18 direction A	9240686	Weekday PM peak	22.784087466770600	26.869989973230100	84.79380710402130
line 18 direction A	9240686	Weekday Evening	25.188257796017700	26.869989973230100	93.74122514043400
line 18 direction A	9240686	Weekday Other	26.728878171644200	26.869989973230100	99.47483493024560
line 18 direction A	6860806	All Weekday	42.42173434410300	43.20906882044670	98.1778490075418
line 18 direction A	6860806	Weekday AM peak	42.360411211573100	43.20906882044670	98.0359271050248
line 18 direction A	6860806	Weekday Day period	43.416123982130100	43.20906882044670	100.47919376032800
line 18 direction A	6860806	Weekday PM peak	40.6385732800485	43.20906882044670	94.05102768800750
line 18 direction A	6860806	Weekday Evening	41.79976057220680	43.20906882044670	96.73839708488930
line 18 direction A	6860806	Weekday Other	42.4226473451655	43.20906882044670	98.17996199235600
line 18 direction A	8060260	All Weekday	37.40418175717030	38.05757317829740	98.28315006302180
line 18 direction A	8060260	Weekday AM peak	36.9605633486098	38.05757317829740	97.11749925685450
line 18 direction A	8060260	Weekday Day period	37.61232471596410	38.05757317829740	98.83006606793540
line 18 direction A	8060260	Weekday PM peak	37.480524983438600	38.05757317829740	98.48374936532260
line 18 direction A	8060260	Weekday Evening	37.18461590894840	38.05757317829740	97.70621929764350
line 18 direction A	8060260	Weekday Other	37.59067681447130	38.05757317829740	98.7731840870708
line 18 direction R	2610807	All Weekday	36.961359493580200	37.3333141510977	99.00369236973690

line 18 direction R	2610807	Weekday AM peak	36.10201482662090	37.3333141510977	96.70187511482790
line 18 direction R	2610807	Weekday Day period	37.28245064408710	37.3333141510977	99.86375839336210
line 18 direction R	2610807	Weekday PM peak	37.45205165037050	37.3333141510977	100.31804703646800
line 18 direction R	2610807	Weekday Evening	36.95139499916560	37.3333141510977	98.97700174598380
line 18 direction R	2610807	Weekday Other	36.3350808856791	37.3333141510977	97.3261595223545
line 18 direction R	8070687	All Weekday	36.86190224180970	37.80406174518970	97.50778233902380
line 18 direction R	8070687	Weekday AM peak	36.24678055206	37.80406174518970	95.8806511225534
line 18 direction R	8070687	Weekday Day period	37.293991083467800	37.80406174518970	98.65075169657720
line 18 direction R	8070687	Weekday PM peak	36.294281012022900	37.80406174518970	96.00630021360350
line 18 direction R	8070687	Weekday Evening	36.78801910734760	37.80406174518970	97.31234531175380
line 18 direction R	8070687	Weekday Other	36.847609263289200	37.80406174518970	97.46997428914590
line 18 direction R	6870925	All Weekday	26.97546338214850	28.674694232676100	94.07410995653690
line 18 direction R	6870925	Weekday AM peak	26.500164823897400	28.674694232676100	92.41655589721760
line 18 direction R	6870925	Weekday Day period	26.776176386811900	28.674694232676100	93.37911738322660
line 18 direction R	6870925	Weekday PM peak	24.770895460260800	28.674694232676100	86.38590967792530
line 18 direction R	6870925	Weekday Evening	27.593625915783000	28.674694232676100	96.22988720255940
line 18 direction R	6870925	Weekday Other	29.049101471077900	28.674694232676100	101.30570612318900
line 18 direction R	9253003	All Weekday	36.69544175113700	37.6570492104494	97.44640783206790
line 18 direction R	9253003	Weekday AM peak	35.78974223979450	37.6570492104494	95.04128175253650
line 18 direction R	9253003	Weekday Day period	36.914409323059100	37.6570492104494	98.02788613828980
line 18 direction R	9253003	Weekday PM peak	36.33641412886040	37.6570492104494	96.49299371756800
line 18 direction R	9253003	Weekday Evening	36.810891691153500	37.6570492104494	97.75299037753320
line 18 direction R	9253003	Weekday Other	37.01664476041190	37.6570492104494	98.29937697332970
line 18 direction R	30030703	All Weekday	30.967745034618100	31.84635659693260	97.24109236910590
line 18 direction R	30030703	Weekday AM peak	30.166835227050200	31.84635659693260	94.72617420215600
line 18 direction R	30030703	Weekday Day period	31.39670447338440	31.84635659693260	98.58805787663780

line 18 direction R	30030703	Weekday PM peak	31.0138112524632	31.84635659693260	97.38574382304810
line 18 direction R	30030703	Weekday Evening	30.817768111242400	31.84635659693260	96.77015333744880
line 18 direction R	30030703	Weekday Other	30.659971324136000	31.84635659693260	96.27465933446550
line 18 direction R	7030137	All Weekday	35.16458054991880	35.74671923335100	98.37149059853010
line 18 direction R	7030137	Weekday AM peak	34.77915499615910	35.74671923335100	97.29327821421680
line 18 direction R	7030137	Weekday Day period	35.47700120957980	35.74671923335100	99.24547474689790
line 18 direction R	7030137	Weekday PM peak	35.50037530088960	35.74671923335100	99.31086282113520
line 18 direction R	7030137	Weekday Evening	35.2303575715218	35.74671923335100	98.55549915375890
line 18 direction R	7030137	Weekday Other	33.741975747143100	35.74671923335100	94.39181125092600
line 18 direction R	1370078	All Weekday	37.829163674718100	38.889001301331600	97.2747111235864
line 18 direction R	1370078	Weekday AM peak	36.72721950354730	38.889001301331600	94.44114858843070
line 18 direction R	1370078	Weekday Day period	38.24816150954120	38.889001301331600	98.35213101302130
line 18 direction R	1370078	Weekday PM peak	38.06786895151880	38.889001301331600	97.88852291821480
line 18 direction R	1370078	Weekday Evening	38.18493470456140	38.889001301331600	98.18954827017860
line 18 direction R	1370078	Weekday Other	36.68308457586060	38.889001301331600	94.32765910243240
line 18 direction R	780092	All Weekday	32.09323710863620	33.5212155682115	95.74007554508420
line 18 direction R	780092	Weekday AM peak	32.287963651921500	33.5212155682115	96.32098092093190
line 18 direction R	780092	Weekday Day period	31.89373799550960	33.5212155682115	95.14493270868950
line 18 direction R	780092	Weekday PM peak	32.1084943462443	33.5212155682115	95.78559071316350
line 18 direction R	780092	Weekday Evening	32.42875323237300	33.5212155682115	96.74098233813890
line 18 direction R	780092	Weekday Other	31.856443697652700	33.5212155682115	95.03367690479120
line 18 direction R	920156	All Weekday	34.95166470478370	37.21854622874350	93.90926902403000
line 18 direction R	920156	Weekday AM peak	34.43993018008200	37.21854622874350	92.5343240663291
line 18 direction R	920156	Weekday Day period	34.98337039540190	37.21854622874350	93.99445690435020
line 18 direction R	920156	Weekday PM peak	34.47044448077250	37.21854622874350	92.6163108814589
line 18 direction R	920156	Weekday Evening	35.62386751559640	37.21854622874350	95.71536538975420

line 18 direction R	920156	Weekday Other	34.53384496080360	37.21854622874350	92.78665735238600
line 18 direction R	1561440	All Weekday	35.662220187891100	36.294047734078400	98.25914279163180
line 18 direction R	1561440	Weekday AM peak	35.157828051578600	36.294047734078400	96.86940489298780
line 18 direction R	1561440	Weekday Day period	36.126186048103900	36.294047734078400	99.53749527414430
line 18 direction R	1561440	Weekday PM peak	35.99670267132970	36.294047734078400	99.18073325706950
line 18 direction R	1561440	Weekday Evening	35.51815525660160	36.294047734078400	97.86220461503330
line 18 direction R	1561440	Weekday Other	34.602168778578500	36.294047734078400	95.33841205065900
line 18 direction R	14401237	All Weekday	24.61662621777070	25.184395656264700	97.74555067255420
line 18 direction R	14401237	Weekday AM peak	24.667242936325000	25.184395656264700	97.94653512040500
line 18 direction R	14401237	Weekday Day period	24.88209871485600	25.184395656264700	98.79966569166600
line 18 direction R	14401237	Weekday PM peak	23.945448699314200	25.184395656264700	95.08049756738040
line 18 direction R	14401237	Weekday Evening	25.639962353799100	25.184395656264700	101.80892447749100
line 18 direction R	14401237	Weekday Other	22.185704522099900	25.184395656264700	88.0930589913962
line 18 direction R	12371081	All Weekday	28.261640885778400	30.757248307518600	91.88611608948730
line 18 direction R	12371081	Weekday AM peak	27.85283252305480	30.757248307518600	90.55697130177360
line 18 direction R	12371081	Weekday Day period	27.664094721638100	30.757248307518600	89.94333447857770
line 18 direction R	12371081	Weekday PM peak	26.89206709927350	30.757248307518600	87.43326721038230
line 18 direction R	12371081	Weekday Evening	28.95682984675740	30.757248307518600	94.14636041963120
line 18 direction R	12371081	Weekday Other	30.65563462771700	30.757248307518600	99.66962688344010
line 18 direction R	10810800	All Weekday	23.087550789373700	24.68303322366010	93.53611681421330
line 18 direction R	10810800	Weekday AM peak	22.937525161097800	24.68303322366010	92.92830809428580
line 18 direction R	10810800	Weekday Day period	22.9937453355642	24.68303322366010	93.15607659403630
line 18 direction R	10810800	Weekday PM peak	21.840464896760700	24.68303322366010	88.4837155095888
line 18 direction R	10810800	Weekday Evening	23.16510286409970	24.68303322366010	93.85030864802580
line 18 direction R	10810800	Weekday Other	24.672316341699800	24.68303322366010	99.95658198948580
line 18 direction R	8000427	All Weekday	14.468495934747300	15.190584200644000	95.24647468221780

line 18 direction R	8000427	Weekday AM peak	14.51204975293560	15.190584200644000	95.53319056893400
line 18 direction R	8000427	Weekday Day period	14.46359145405340	15.190584200644000	95.2141883617631
line 18 direction R	8000427	Weekday PM peak	13.874632352633200	15.190584200644000	91.33705570089270
line 18 direction R	8000427	Weekday Evening	14.371037045343700	15.190584200644000	94.60490034829890
line 18 direction R	8000427	Weekday Other	15.228554247119200	15.190584200644000	100.2499577763020
line 18 direction R	4270356	All Weekday	9.755715032029160	10.081861220205600	96.7650200587682
line 18 direction R	4270356	Weekday AM peak	9.974928667483200	10.081861220205600	98.93935702558440
line 18 direction R	4270356	Weekday Day period	9.796685852212130	10.081861220205600	97.17140157194420
line 18 direction R	4270356	Weekday PM peak	9.151750191470740	10.081861220205600	90.77441150577620
line 18 direction R	4270356	Weekday Evening	9.695180846433500	10.081861220205600	96.16459336895880
line 18 direction R	4270356	Weekday Other	10.118679650961300	10.081861220205600	100.36519477853800
line 18 direction R	3560087	All Weekday	23.83797929146270	24.805138242159900	96.10097334973370
line 18 direction R	3560087	Weekday AM peak	23.551034774690600	24.805138242159900	94.94417868094040
line 18 direction R	3560087	Weekday Day period	23.840460494504300	24.805138242159900	96.11097612826030
line 18 direction R	3560087	Weekday PM peak	23.169289227983900	24.805138242159900	93.40520097809550
line 18 direction R	3560087	Weekday Evening	23.97864556079740	24.805138242159900	96.66805855587690
line 18 direction R	3560087	Weekday Other	24.50957209379480	24.805138242159900	98.80844788898310
line 18 direction R	871064	All Weekday	16.196328521236900	17.712187701125800	91.4417168253441
line 18 direction R	871064	Weekday AM peak	15.916091312274700	17.712187701125800	89.85954519476460
line 18 direction R	871064	Weekday Day period	15.948842649972100	17.712187701125800	90.04445367840340
line 18 direction R	871064	Weekday PM peak	15.537856276031300	17.712187701125800	87.72409449479650
line 18 direction R	871064	Weekday Evening	16.507257472205200	17.712187701125800	93.19716881250070
line 18 direction R	871064	Weekday Other	17.304597034386000	17.712187701125800	97.69881240184750
line 18 direction R	10641041	All Weekday	17.75581992787020	18.087528889943000	98.16609021557790
line 18 direction R	10641041	Weekday AM peak	18.084979386579500	18.087528889943000	99.98590463418730
line 18 direction R	10641041	Weekday Day period	17.667659569962400	18.087528889943000	97.67868058409010

line 18 direction R	10641041	Weekday PM peak	17.34035052066800	18.087528889943000	95.8690964707153
line 18 direction R	10641041	Weekday Evening	17.808563924260600	18.087528889943000	98.45769442923990
line 18 direction R	10641041	Weekday Other	18.034133758852500	18.087528889943000	99.70479587667600
line 18 direction R	10411128	All Weekday	24.89418257210700	26.51359770489430	93.89213357307470
line 18 direction R	10411128	Weekday AM peak	24.85887493867870	26.51359770489430	93.75896555181510
line 18 direction R	10411128	Weekday Day period	24.533156042862800	26.51359770489430	92.5304680108881
line 18 direction R	10411128	Weekday PM peak	24.44659779570690	26.51359770489430	92.20400063320770
line 18 direction R	10411128	Weekday Evening	25.38310694600540	26.51359770489430	95.73618498903210
line 18 direction R	10411128	Weekday Other	25.6280265002678	26.51359770489430	96.65993572625180
line 18 direction R	11280072	All Weekday	26.747183622699700	28.47723058861490	93.92480613403890
line 18 direction R	11280072	Weekday AM peak	27.230794692160900	28.47723058861490	95.62304384699450
line 18 direction R	11280072	Weekday Day period	26.173136587159000	28.47723058861490	91.90899552438560
line 18 direction R	11280072	Weekday PM peak	25.629490119553500	28.47723058861490	89.99993886273500
line 18 direction R	11280072	Weekday Evening	27.387330180046300	28.47723058861490	96.17273033212590
line 18 direction R	11280072	Weekday Other	27.96277504018100	28.47723058861490	98.19344951106470
line 18 direction R	720131	All Weekday	27.078716576512000	27.688974348584400	97.79602608464420
line 18 direction R	720131	Weekday AM peak	26.621237612821000	27.688974348584400	96.14381983846240
line 18 direction R	720131	Weekday Day period	27.44378492503220	27.688974348584400	99.11448715844280
line 18 direction R	720131	Weekday PM peak	27.38335243732530	27.688974348584400	98.8962324591314
line 18 direction R	720131	Weekday Evening	26.965434797756300	27.688974348584400	97.38690374833240
line 18 direction R	720131	Weekday Other	26.22219699420460	27.688974348584400	94.70266635407230
line 18 direction R	1310057	All Weekday	15.543503599364400	16.213322948048500	95.86871025248580
line 18 direction R	1310057	Weekday AM peak	15.549066673810200	16.213322948048500	95.90302200007520
line 18 direction R	1310057	Weekday Day period	15.244976705611300	16.213322948048500	94.02746589616460
line 18 direction R	1310057	Weekday PM peak	14.937326064977000	16.213322948048500	92.12994839392220
line 18 direction R	1310057	Weekday Evening	15.816750061803900	16.213322948048500	97.55403079606040

line 18 direction R	1310057	Weekday Other	16.637514926103300	16.213322948048500	102.61631732997600
line 18 direction R	575316	All Weekday	20.569730038749600	21.79861862566760	94.36253916809680
line 18 direction R	575316	Weekday AM peak	21.07685625311700	21.79861862566760	96.68895362158090
line 18 direction R	575316	Weekday Day period	19.99981197032070	21.79861862566760	91.74807043401930
line 18 direction R	575316	Weekday PM peak	19.66492417123280	21.79861862566760	90.21179052179760
line 18 direction R	575316	Weekday Evening	21.001361159395600	21.79861862566760	96.34262390675880
line 18 direction R	575316	Weekday Other	22.074321627084700	21.79861862566760	101.26477281038600
line 18 direction R	53160043	All Weekday	21.042601011615000	21.666249870784800	97.12156527830560
line 18 direction R	53160043	Weekday AM peak	20.770679117025600	21.666249870784800	95.86651700640310
line 18 direction R	53160043	Weekday Day period	20.960185171610900	21.666249870784800	96.74117716086180
line 18 direction R	53160043	Weekday PM peak	20.63411015684700	21.666249870784800	95.23618660315790
line 18 direction R	53160043	Weekday Evening	21.256032622554000	21.666249870784800	98.10665320174350
line 18 direction R	53160043	Weekday Other	21.552157917498100	21.666249870784800	99.47341162422180
line 18 direction R	430375	All Weekday	19.357697235810700	21.441063159221600	90.28329002186220
line 18 direction R	430375	Weekday AM peak	19.7458253570466	21.441063159221600	92.09349933076480
line 18 direction R	430375	Weekday Day period	19.413901225918900	21.441063159221600	90.54542249957940
line 18 direction R	430375	Weekday PM peak	15.868664697657800	21.441063159221600	**74.01062428582440**
line 18 direction R	430375	Weekday Evening	19.648176956784100	21.441063159221600	91.63807228623180
line 18 direction R	430375	Weekday Other	21.61175313634720	21.441063159221600	100.79608914846300
line 18 direction R	3751297	All Weekday	26.196418246612200	27.31814855863720	95.89382746924700
line 18 direction R	3751297	Weekday AM peak	25.114928408042200	27.31814855863720	91.93495801567260
line 18 direction R	3751297	Weekday Day period	25.644141036618200	27.31814855863720	93.87217798297790
line 18 direction R	3751297	Weekday PM peak	25.247699524449700	27.31814855863720	92.42097600522470
line 18 direction R	3751297	Weekday Evening	27.25109907856040	27.31814855863720	99.75456067261320
line 18 direction R	3751297	Weekday Other	27.814878648839400	27.31814855863720	101.81831535594700
line 18 direction R	12970118	All Weekday	22.268222979197300	22.623704449456000	98.42872120676400

line 18 direction R	12970118	Weekday AM peak	22.13863485702660	22.623704449456000	97.85592322639660
line 18 direction R	12970118	Weekday Day period	22.71453306254250	22.623704449456000	100.40147542277800
line 18 direction R	12970118	Weekday PM peak	22.55397541836530	22.623704449456000	99.69178773862390
line 18 direction R	12970118	Weekday Evening	22.011451898179500	22.623704449456000	97.29375641091670
line 18 direction R	12970118	Weekday Other	21.15187899832950	22.623704449456000	93.49432161114560
line 18 direction R	1181362	All Weekday	19.774195477191000	20.125917558641500	98.25239231739030
line 18 direction R	1181362	Weekday AM peak	19.523152665049300	20.125917558641500	97.00503148819990
line 18 direction R	1181362	Weekday Day period	20.011081959411800	20.125917558641500	99.42941434150720
line 18 direction R	1181362	Weekday PM peak	19.815495066326900	20.125917558641500	98.45759831118190
line 18 direction R	1181362	Weekday Evening	19.56749989254240	20.125917558641500	97.22538033621540
line 18 direction R	1181362	Weekday Other	19.652524438047400	20.125917558641500	97.64784328856200
line 18 direction R	13621077	All Weekday	21.97056605484010	22.33922426514210	98.34972689325980
line 18 direction R	13621077	Weekday AM peak	21.578429418592500	22.33922426514210	96.59435422859900
line 18 direction R	13621077	Weekday Day period	22.18497515758310	22.33922426514210	99.30951448569480
line 18 direction R	13621077	Weekday PM peak	22.012357283000800	22.33922426514210	98.53680244997900
line 18 direction R	13621077	Weekday Evening	21.923363006082900	22.33922426514210	98.13842569409160
line 18 direction R	13621077	Weekday Other	21.74049268616920	22.33922426514210	97.31981929244020
line 18 direction R	10777423	All Weekday	19.40303402156300	19.858085019159200	97.70848499662890
line 18 direction R	10777423	Weekday AM peak	19.044013869113800	19.858085019159200	95.90055562124980
line 18 direction R	10777423	Weekday Day period	19.781763562746100	19.858085019159200	99.61566557732300
line 18 direction R	10777423	Weekday PM peak	19.59611068669620	19.858085019159200	98.68076739418590
line 18 direction R	10777423	Weekday Evening	19.200831769312300	19.858085019159200	96.69024858533540
line 18 direction R	10777423	Weekday Other	18.787357715035000	19.858085019159200	94.60810393806290
line 80 direction A	23211310	All Weekday	16.766072460218300	20.024447573611500	83.72801496063700
line 80 direction A	23211310	Weekday AM peak	16.779105706732700	20.024447573611500	83.79310163264850
line 80 direction A	23211310	Weekday Day period	16.56129643867010	20.024447573611500	82.70538489408720

line 80 direction A	23211310	Weekday PM peak	14.882982971301100	20.024447573611500	**74.32406270679880**
line 80 direction A	23211310	Weekday Evening	17.153912053621200	20.024447573611500	85.6648453874299
line 80 direction A	23211310	Weekday Other	20.16335162530070	20.024447573611500	100.69367232818100
line 80 direction A	13101022	All Weekday	27.258542336454500	30.366392223497000	89.76549514289120
line 80 direction A	13101022	Weekday AM peak	27.86091254669560	30.366392223497000	91.74916908679490
line 80 direction A	13101022	Weekday Day period	26.852249934717500	30.366392223497000	88.42752783104650
line 80 direction A	13101022	Weekday PM peak	24.54224830577130	30.366392223497000	80.82042846953970
line 80 direction A	13101022	Weekday Evening	27.832217996554600	30.366392223497000	91.65467465383830
line 80 direction A	13101022	Weekday Other	31.220856079953600	30.366392223497000	102.81384713128800
line 80 direction A	10220720	All Weekday	28.690101354848900	33.22976254221340	86.33856868041860
line 80 direction A	10220720	Weekday AM peak	28.872385669899300	33.22976254221340	86.8871260612271
line 80 direction A	10220720	Weekday Day period	28.268414452894800	33.22976254221340	85.069565023175
line 80 direction A	10220720	Weekday PM peak	27.0814512116236	33.22976254221340	81.49757668963400
line 80 direction A	10220720	Weekday Evening	28.84849904544530	33.22976254221340	86.8152428377954
line 80 direction A	10220720	Weekday Other	33.09262938717290	33.22976254221340	99.5873182817174
line 80 direction A	7207166	All Weekday	40.63583442616890	45.89232879127000	88.54602827193860
line 80 direction A	7207166	Weekday AM peak	40.67024743288370	45.89232879127000	88.62101467515940
line 80 direction A	7207166	Weekday Day period	39.850156715614600	45.89232879127000	86.83402600217420
line 80 direction A	7207166	Weekday PM peak	39.00338627064940	45.89232879127000	84.98890184467800
line 80 direction A	7207166	Weekday Evening	41.991069385715400	45.89232879127000	91.49910342685280
line 80 direction A	7207166	Weekday Other	42.86311995106140	45.89232879127000	93.39931330574620
line 80 direction A	71660522	All Weekday	38.95798050720920	48.894218007158900	**79.67809302422030**
line 80 direction A	71660522	Weekday AM peak	33.32734665701540	48.894218007158900	**68.16214271416660**
line 80 direction A	71660522	Weekday Day period	37.414693572233200	48.894218007158900	**76.52171380827710**
line 80 direction A	71660522	Weekday PM peak	39.65109683716120	48.894218007158900	81.09567644860510
line 80 direction A	71660522	Weekday Evening	42.47990803672650	48.894218007158900	86.88125052026960

line 80 direction A	71660522	Weekday Other	42.93733837421380	48.894218007158900	87.81680150386510
line 80 direction A	5221298	All Weekday	38.911513376994400	44.76133238910760	86.93108828561890
line 80 direction A	5221298	Weekday AM peak	40.738059410460300	44.76133238910760	91.01172202902020
line 80 direction A	5221298	Weekday Day period	38.04066271046740	44.76133238910760	84.98554596137170
line 80 direction A	5221298	Weekday PM peak	33.677622359390200	44.76133238910760	**75.23820351600050**
line 80 direction A	5221298	Weekday Evening	40.64197716493520	44.76133238910760	90.79706745911170
line 80 direction A	5221298	Weekday Other	43.99792554822160	44.76133238910760	98.29449482368020
line 80 direction A	12980122	All Weekday	46.456835478709700	50.69451791666950	91.64074812798190
line 80 direction A	12980122	Weekday AM peak	48.346512282414300	50.69451791666950	95.36832436573370
line 80 direction A	12980122	Weekday Day period	47.93137162321520	50.69451791666950	94.54941795087940
line 80 direction A	12980122	Weekday PM peak	39.02927366994320	50.69451791666950	**76.98914058932110**
line 80 direction A	12980122	Weekday Evening	46.363010778497700	50.69451791666950	91.4556695355269
line 80 direction A	12980122	Weekday Other	51.09818697942440	50.69451791666950	100.79627754507600
line 80 direction A	1220335	All Weekday	21.854880543559000	26.76757944375020	81.64683171851660
line 80 direction A	1220335	Weekday AM peak	21.655084401659600	26.76757944375020	80.90042077643190
line 80 direction A	1220335	Weekday Day period	21.48162414200750	26.76757944375020	80.2523970729191
line 80 direction A	1220335	Weekday PM peak	18.385222266103800	26.76757944375020	**68.68466498713020**
line 80 direction A	1220335	Weekday Evening	23.23373547803660	26.76757944375020	86.79804435384360
line 80 direction A	1220335	Weekday Other	26.934539846054600	26.76757944375020	100.62374112928400
line 80 direction A	3353303	All Weekday	24.712935672545000	25.636457265745400	96.39762396329850
line 80 direction A	3353303	Weekday AM peak	24.287970999160100	25.636457265745400	94.73996639782550
line 80 direction A	3353303	Weekday Day period	24.561180817998000	25.636457265745400	95.80567456493250
line 80 direction A	3353303	Weekday PM peak	23.74956767865240	25.636457265745400	92.63981927169720
line 80 direction A	3353303	Weekday Evening	25.23310671090680	25.636457265745400	98.42665251810130
line 80 direction A	3353303	Weekday Other	26.623565717539700	25.636457265745400	103.8504089764120
line 80 direction A	33031037	All Weekday	33.395469235276400	36.73683221381200	90.90459689314410

line 80 direction A	33031037	Weekday AM peak	33.26122454216770	36.73683221381200	90.53917427769510
line 80 direction A	33031037	Weekday Day period	32.697705429256800	36.73683221381200	89.00523931664270
line 80 direction A	33031037	Weekday PM peak	31.884293002835200	36.73683221381200	86.791078820475
line 80 direction A	33031037	Weekday Evening	34.60030372792080	36.73683221381200	94.18423321462120
line 80 direction A	33031037	Weekday Other	35.82841203296200	36.73683221381200	97.52722233761780
line 80 direction A	10371414	All Weekday	29.47346428968420	34.49483690704160	85.44311825306130
line 80 direction A	10371414	Weekday AM peak	25.975576975988700	34.49483690704160	**75.30279689679070**
line 80 direction A	10371414	Weekday Day period	29.18085286588310	34.49483690704160	84.59484224993200
line 80 direction A	10371414	Weekday PM peak	26.937293272186200	34.49483690704160	**78.09079760190850**
line 80 direction A	10371414	Weekday Evening	32.235082227398000	34.49483690704160	93.44900604767830
line 80 direction A	10371414	Weekday Other	32.44446593709750	34.49483690704160	94.05600619168160
line 80 direction A	14140052	All Weekday	37.33634417882930	42.28583740530950	88.29515144978850
line 80 direction A	14140052	Weekday AM peak	37.915583525805900	42.28583740530950	89.66497024141990
line 80 direction A	14140052	Weekday Day period	37.40024158573070	42.28583740530950	88.44625974235690
line 80 direction A	14140052	Weekday PM peak	31.227524586767700	42.28583740530950	**73.84866069330040**
line 80 direction A	14140052	Weekday Evening	39.184487838290400	42.28583740530950	92.66574872978720
line 80 direction A	14140052	Weekday Other	40.80983206691360	42.28583740530950	96.50945699798170
line 80 direction A	520630	All Weekday	30.604120475561300	38.10558743499670	80.31399733114770
line 80 direction A	520630	Weekday AM peak	29.701444063881000	38.10558743499670	**77.94511530506640**
line 80 direction A	520630	Weekday Day period	31.340654706898900	38.10558743499670	82.24687458331960
line 80 direction A	520630	Weekday PM peak	23.375164029265900	38.10558743499670	**61.34314047550370**
line 80 direction A	520630	Weekday Evening	33.051135967744500	38.10558743499670	86.73566842166030
line 80 direction A	520630	Weekday Other	33.63706576058020	38.10558743499670	88.27331639476970
line 80 direction A	6302117	All Weekday	36.781085430690600	39.96403261063370	92.03547046677140
line 80 direction A	6302117	Weekday AM peak	36.75019016792280	39.96403261063370	91.9581627959743
line 80 direction A	6302117	Weekday Day period	34.1069048727962	39.96403261063370	85.34400220592610

line 80 direction A	6302117	Weekday PM peak	37.73660685504520	39.96403261063370	94.42642393651800
line 80 direction A	6302117	Weekday Evening	38.811559655528000	39.96403261063370	97.1162245653883
line 80 direction A	6302117	Weekday Other	39.96432512731610	39.96403261063370	100.0007319498640
line 80 direction A	21170782	All Weekday	27.606173765550100	36.94838577484270	**74.71550701504940**
line 80 direction A	21170782	Weekday AM peak	25.880742361815900	36.94838577484270	**70.04566456442480**
line 80 direction A	21170782	Weekday Day period	29.39755440941210	36.94838577484270	**79.56383964527130**
line 80 direction A	21170782	Weekday PM peak	18.67941924027280	36.94838577484270	**50.55544064658760**
line 80 direction A	21170782	Weekday Evening	29.182230284636300	36.94838577484270	**78.98106959927290**
line 80 direction A	21170782	Weekday Other	33.238608727694500	36.94838577484270	89.95956935776580
line 80 direction A	7823301	All Weekday	25.856387184821300	32.83824015084790	**78.73865062818740**
line 80 direction A	7823301	Weekday AM peak	26.74218630294640	32.83824015084790	81.43611283705130
line 80 direction A	7823301	Weekday Day period	27.040133954374100	32.83824015084790	82.34343201755250
line 80 direction A	7823301	Weekday PM peak	15.781582198589200	32.83824015084790	**48.058550415899100**
line 80 direction A	7823301	Weekday Evening	27.36819212396280	32.83824015084790	83.34244465672470
line 80 direction A	7823301	Weekday Other	32.49334069193950	32.83824015084790	98.94970175830380
line 80 direction A	33010982	All Weekday	31.656518550590800	36.165308954797	87.53283039876200
line 80 direction A	33010982	Weekday AM peak	32.2242516918037	36.165308954797	89.10265838480950
line 80 direction A	33010982	Weekday Day period	32.16633217474740	36.165308954797	88.94250624252110
line 80 direction A	33010982	Weekday PM peak	24.872736780229200	36.165308954797	**68.77512593993770**
line 80 direction A	33010982	Weekday Evening	32.65111150260780	36.165308954797	90.28296023523080
line 80 direction A	33010982	Weekday Other	37.56297115603060	36.165308954797	103.86464886275500
line 80 direction A	9826495	All Weekday	31.835932434989900	34.80068576324700	91.48076176306800
line 80 direction A	9826495	Weekday AM peak	33.7020056319219	34.80068576324700	96.84293539845870
line 80 direction A	9826495	Weekday Day period	33.62958697339580	34.80068576324700	96.63483990568950
line 80 direction A	9826495	Weekday PM peak	26.769058642856500	34.80068576324700	**76.92106651279650**
line 80 direction A	9826495	Weekday Evening	30.67366763190540	34.80068576324700	88.14098618797870

line 80 direction A	9826495	Weekday Other	34.05806867917820	34.80068576324700	97.86608491246160
line 80 direction A	64956497	All Weekday	41.683390881059	45.21531879203010	92.18864755279870
line 80 direction A	64956497	Weekday AM peak	45.37765694371160	45.21531879203010	100.35903352230700
line 80 direction A	64956497	Weekday Day period	43.818822276197300	45.21531879203010	96.91145268209630
line 80 direction A	64956497	Weekday PM peak	34.8548532632765	45.21531879203010	**77.08638177161370**
line 80 direction A	64956497	Weekday Evening	39.56992790313050	45.21531879203010	87.51442865002040
line 80 direction A	64956497	Weekday Other	47.26080389166320	45.21531879203010	104.52387631953100
line 80 direction A	64972521	All Weekday	30.945311877046100	32.7043100892076	94.62150949717810
line 80 direction A	64972521	Weekday AM peak	31.11905297185370	32.7043100892076	95.15275780767170
line 80 direction A	64972521	Weekday Day period	31.248115198332700	32.7043100892076	95.54739150007200
line 80 direction A	64972521	Weekday PM peak	29.1109796780374	32.7043100892076	89.01267019127240
line 80 direction A	64972521	Weekday Evening	31.0319776381986	32.7043100892076	94.88650747731000
line 80 direction A	64972521	Weekday Other	32.97536629084630	32.7043100892076	100.8288088050150
line 80 direction A	25217444	All Weekday	17.674704567146700	19.73535698024990	89.55857542802300
line 80 direction A	25217444	Weekday AM peak	17.272432717568900	19.73535698024990	87.52024467991250
line 80 direction A	25217444	Weekday Day period	17.930509325895400	19.73535698024990	90.85475040476460
line 80 direction A	25217444	Weekday PM peak	15.125784057819900	19.73535698024990	**76.6430730032245**
line 80 direction A	25217444	Weekday Evening	18.309562959606100	19.73535698024990	92.77543333991570
line 80 direction A	25217444	Weekday Other	19.5641283365164	19.73535698024990	99.13237625291060
line 80 direction R	74442522	All Weekday	18.727344782980900	20.348116657569100	92.03478188245340
line 80 direction R	74442522	Weekday AM peak	18.321978194560300	20.348116657569100	90.04262410568080
line 80 direction R	74442522	Weekday Day period	18.506104580974900	20.348116657569100	90.94750581789610
line 80 direction R	74442522	Weekday PM peak	17.563467032210900	20.348116657569100	86.31495153964370
line 80 direction R	74442522	Weekday Evening	19.597103947394500	20.348116657569100	96.30917827525220
line 80 direction R	74442522	Weekday Other	19.713643888318000	20.348116657569100	96.88190912245910
line 80 direction R	25226498	All Weekday	37.9936263191541	40.97732440832590	92.71866054640330

line 80 direction R	25226498	Weekday AM peak	37.651446234930800	40.97732440832590	91.88361314112720
line 80 direction R	25226498	Weekday Day period	39.74688217584600	40.97732440832590	96.99726067954330
line 80 direction R	25226498	Weekday PM peak	36.79856651650950	40.97732440832590	89.80226759029840
line 80 direction R	25226498	Weekday Evening	38.87094873994570	40.97732440832590	94.85965543433030
line 80 direction R	25226498	Weekday Other	33.42300203097890	40.97732440832590	81.56462754358830
line 80 direction R	64986496	All Weekday	42.83779901906030	45.64062378345470	93.85892537820550
line 80 direction R	64986496	Weekday AM peak	41.67071118582930	45.64062378345470	91.30180030741710
line 80 direction R	64986496	Weekday Day period	45.38799727365820	45.64062378345470	99.4464876050004
line 80 direction R	64986496	Weekday PM peak	42.24318501762130	45.64062378345470	92.55610794902200
line 80 direction R	64986496	Weekday Evening	43.1358658049217	45.64062378345470	94.5119988052376
line 80 direction R	64986496	Weekday Other	37.024969780926600	45.64062378345470	81.12283906678030
line 80 direction R	64960983	All Weekday	25.6525506085069	28.357360210412600	90.46170171752260
line 80 direction R	64960983	Weekday AM peak	22.184530827786800	28.357360210412600	**78.23200277873810**
line 80 direction R	64960983	Weekday Day period	26.439544844325100	28.357360210412600	93.23697497983860
line 80 direction R	64960983	Weekday PM peak	26.283567138926100	28.357360210412600	92.68693187201180
line 80 direction R	64960983	Weekday Evening	27.208485365283800	28.357360210412600	95.94858323692990
line 80 direction R	64960983	Weekday Other	23.943898813579400	28.357360210412600	84.43627557683380
line 80 direction R	9833302	All Weekday	30.688432021207000	36.99237011729720	82.95881535543310
line 80 direction R	9833302	Weekday AM peak	25.443817435003200	36.99237011729720	**68.78125774132530**
line 80 direction R	9833302	Weekday Day period	32.57018634115070	36.99237011729720	88.04568682102730
line 80 direction R	9833302	Weekday PM peak	28.076102529461400	36.99237011729720	**75.89700914117250**
line 80 direction R	9833302	Weekday Evening	34.76130842171350	36.99237011729720	93.96885982566330
line 80 direction R	9833302	Weekday Other	27.635160800362800	36.99237011729720	**74.70502893633440**
line 80 direction R	33020780	All Weekday	36.86003343276660	40.7149890964302	90.53185141586700
line 80 direction R	33020780	Weekday AM peak	36.22660338983150	40.7149890964302	88.97608520545510
line 80 direction R	33020780	Weekday Day period	38.3219833959352	40.7149890964302	94.12254367838010

line 80 direction R	33020780	Weekday PM peak	34.23099930505540	40.7149890964302	84.07468616528920
line 80 direction R	33020780	Weekday Evening	37.74912412339900	40.7149890964302	92.71554521110930
line 80 direction R	33020780	Weekday Other	35.401282737771600	40.7149890964302	86.94901686925790
line 80 direction R	7800353	All Weekday	29.089653498653900	34.93802204367690	83.26073371379810
line 80 direction R	7800353	Weekday AM peak	19.852720574505800	34.93802204367690	**56.82268031569570**
line 80 direction R	7800353	Weekday Day period	31.264041714496000	34.93802204367690	89.48429214284660
line 80 direction R	7800353	Weekday PM peak	27.66883986142840	34.93802204367690	**79.19406492685490**
line 80 direction R	7800353	Weekday Evening	32.666280605027300	34.93802204367690	93.49779608070050
line 80 direction R	7800353	Weekday Other	29.332949279573000	34.93802204367690	83.95709763678980
line 80 direction R	3530631	All Weekday	32.65126970762670	42.47843662134960	**76.86551649411740**
line 80 direction R	3530631	Weekday AM peak	14.276955428543500	42.47843662134960	**33.60988907338440**
line 80 direction R	3530631	Weekday Day period	33.63704902560110	42.47843662134960	**79.18617468302770**
line 80 direction R	3530631	Weekday PM peak	38.810919413443000	42.47843662134960	91.36616716712380
line 80 direction R	3530631	Weekday Evening	42.50962768889630	42.47843662134960	100.07342800260000
line 80 direction R	3530631	Weekday Other	24.434451701298300	42.47843662134960	**57.522012684001500**
line 80 direction R	6310053	All Weekday	44.69625286998940	49.527581880038700	90.24517485680740
line 80 direction R	6310053	Weekday AM peak	43.39607754066840	49.527581880038700	87.62002079119980
line 80 direction R	6310053	Weekday Day period	44.944394251606600	49.527581880038700	90.74619140596630
line 80 direction R	6310053	Weekday PM peak	42.83312737163060	49.527581880038700	86.48338106911250
line 80 direction R	6310053	Weekday Evening	46.707386776400400	49.527581880038700	94.30580901270500
line 80 direction R	6310053	Weekday Other	44.07074952318450	49.527581880038700	88.98223545403190
line 80 direction R	531415	All Weekday	37.383362311087500	41.40703510347520	90.28263486546990
line 80 direction R	531415	Weekday AM peak	36.929474420731900	41.40703510347520	89.18647357495180
line 80 direction R	531415	Weekday Day period	37.992590922848200	41.40703510347520	91.75395153964930
line 80 direction R	531415	Weekday PM peak	34.335920564592500	41.40703510347520	82.92291510074990
line 80 direction R	531415	Weekday Evening	38.505994657145500	41.40703510347520	92.99384648265660

line 80 direction R	531415	Weekday Other	38.144408924401900	41.40703510347520	92.1205993838484
line 80 direction R	14151038	All Weekday	27.9449238786272	33.76926933224830	82.75252746419650
line 80 direction R	14151038	Weekday AM peak	24.70534228598460	33.76926933224830	**73.15924440921200**
line 80 direction R	14151038	Weekday Day period	27.79620012597580	33.76926933224830	82.3121159433308
line 80 direction R	14151038	Weekday PM peak	26.386548140849500	33.76926933224830	**78.13775264498090**
line 80 direction R	14151038	Weekday Evening	30.852638299553800	33.76926933224830	91.36306147462540
line 80 direction R	14151038	Weekday Other	29.320183884792900	33.76926933224830	86.82504674980720
line 80 direction R	10383304	All Weekday	27.334781461121800	33.944766017840900	80.52723488138070
line 80 direction R	10383304	Weekday AM peak	28.81997757742280	33.944766017840900	84.9025666056306
line 80 direction R	10383304	Weekday Day period	24.39540628535940	33.944766017840900	**71.8679465120999**
line 80 direction R	10383304	Weekday PM peak	23.742256764187700	33.944766017840900	**69.94379266514640**
line 80 direction R	10383304	Weekday Evening	30.588927407147600	33.944766017840900	90.113826063996
line 80 direction R	10383304	Weekday Other	33.05337418215130	33.944766017840900	97.37399328302600
line 80 direction R	33040334	All Weekday	32.4353287230504	37.367225150445700	86.80154491659800
line 80 direction R	33040334	Weekday AM peak	30.5761332810217	37.367225150445700	81.82607394024560
line 80 direction R	33040334	Weekday Day period	33.070350021891700	37.367225150445700	88.50095207430010
line 80 direction R	33040334	Weekday PM peak	28.868324815077400	37.367225150445700	**77.25573600621800**
line 80 direction R	33040334	Weekday Evening	34.144628414069800	37.367225150445700	91.37587358065450
line 80 direction R	33040334	Weekday Other	34.97840494224760	37.367225150445700	93.607177951854
line 80 direction R	3340121	All Weekday	25.33747817919160	28.733891952885400	88.17976423359930
line 80 direction R	3340121	Weekday AM peak	20.589950648972000	28.733891952885400	**71.65736783145500**
line 80 direction R	3340121	Weekday Day period	25.24187153234320	28.733891952885400	87.84703295234760
line 80 direction R	3340121	Weekday PM peak	25.31428609195670	28.733891952885400	88.09905088201840
line 80 direction R	3340121	Weekday Evening	27.74254020783220	28.733891952885400	96.54988698823430
line 80 direction R	3340121	Weekday Other	26.97401242290880	28.733891952885400	93.87524832047720
line 80 direction R	1211301	All Weekday	31.357021477956000	38.34233506413650	81.78172097631530

line 80 direction R	1211301	Weekday AM peak	25.040037540765600	38.34233506413650	**65.30650128345160**
line 80 direction R	1211301	Weekday Day period	29.589695676958200	38.34233506413650	**77.17238824255900**
line 80 direction R	1211301	Weekday PM peak	31.372821695812100	38.34233506413650	81.82292925909130
line 80 direction R	1211301	Weekday Evening	36.584069858106000	38.34233506413650	95.4142980517767
line 80 direction R	1211301	Weekday Other	35.233675965807000	38.34233506413650	91.89235842540750
line 80 direction R	13010523	All Weekday	26.968091953577800	30.141498226415300	89.47163724576750
line 80 direction R	13010523	Weekday AM peak	22.44836109408310	30.141498226415300	**74.47659345085210**
line 80 direction R	13010523	Weekday Day period	27.121649886060900	30.141498226415300	89.98109411260830
line 80 direction R	13010523	Weekday PM peak	26.720303866013900	30.141498226415300	88.64955439606140
line 80 direction R	13010523	Weekday Evening	29.147358253390100	30.141498226415300	96.70175660958360
line 80 direction R	13010523	Weekday Other	28.29907835064190	30.141498226415300	93.88743100315190
line 80 direction R	5237165	All Weekday	30.393415206251000	30.784282980830500	98.73030086546780
line 80 direction R	5237165	Weekday AM peak	28.224628715139500	30.784282980830500	91.68519121499480
line 80 direction R	5237165	Weekday Day period	30.55566921763930	30.784282980830500	99.2573685626085
line 80 direction R	5237165	Weekday PM peak	30.281683219898200	30.784282980830500	98.3673494645132
line 80 direction R	5237165	Weekday Evening	31.261736583116800	30.784282980830500	101.55096547996100
line 80 direction R	5237165	Weekday Other	31.288143627480000	30.784282980830500	101.63674641037900
line 80 direction R	71650721	All Weekday	16.773759275037900	22.609724685532800	**74.18825088910010**
line 80 direction R	71650721	Weekday AM peak	15.506610382686000	22.609724685532800	**68.58380895105790**
line 80 direction R	71650721	Weekday Day period	16.434492670380200	22.609724685532800	**72.68771689597810**
line 80 direction R	71650721	Weekday PM peak	13.903026294749700	22.609724685532800	**61.49135599004310**
line 80 direction R	71650721	Weekday Evening	18.216684001311200	22.609724685532800	80.57012747690580
line 80 direction R	71650721	Weekday Other	21.192294249870300	22.609724685532800	93.73088148848830
line 80 direction R	7211023	All Weekday	38.855574644333000	42.401643251449500	91.63695476119260
line 80 direction R	7211023	Weekday AM peak	38.887587622759200	42.401643251449500	91.71245414275270
line 80 direction R	7211023	Weekday Day period	39.284448929373500	42.401643251449500	92.64841151652900

line 80 direction R	7211023	Weekday PM peak	35.482974037901000	42.401643251449500	83.68301631019460
line 80 direction R	7211023	Weekday Evening	39.10248164762250	42.401643251449500	92.21926003135700
line 80 direction R	7211023	Weekday Other	42.37485401688510	42.401643251449500	99.93682029159690
line 80 direction R	10231308	All Weekday	21.59882936260040	23.056570978337800	93.6775437375011
line 80 direction R	10231308	Weekday AM peak	21.478670329319100	23.056570978337800	93.15639497954310
line 80 direction R	10231308	Weekday Day period	21.460873708776200	23.056570978337800	93.07920821764520
line 80 direction R	10231308	Weekday PM peak	21.112585254633400	23.056570978337800	91.56862603059730
line 80 direction R	10231308	Weekday Evening	21.766947045872600	23.056570978337800	94.40669675609230
line 80 direction R	10231308	Weekday Other	22.90394154882540	23.056570978337800	99.33802199097270
line 80 direction R	13082321	All Weekday	21.77244023198900	26.78625001705210	81.28215117132380
line 80 direction R	13082321	Weekday AM peak	21.711077136183600	26.78625001705210	81.05306686214890
line 80 direction R	13082321	Weekday Day period	21.36527127763180	26.78625001705210	**79.76208414403160**
line 80 direction R	13082321	Weekday PM peak	19.15021046440070	26.78625001705210	**71.49268916779940**
line 80 direction R	13082321	Weekday Evening	22.541492175950200	26.78625001705210	84.15322100555440
line 80 direction R	13082321	Weekday Other	26.298116845663600	26.78625001705210	98.17767260785760

```
import pandas as pd

# ---- Load & prepare data ----
file_path = '/Users/benfall/Desktop/Data Files/OG_trimmed_with_first_stops.csv'
df = pd.read_csv(file_path, encoding='utf-8', low_memory=False)
df.columns = df.columns.str.strip().str.replace('\ufeff', '', regex=False)

# Convert datatypes
df['Segment_XXXXYYYY'] = df['Segment_XXXXYYYY'].astype(str)
df['segment_speed'] = pd.to_numeric(df['segment_speed'], errors='coerce')
df['DTDepartTheo'] = pd.to_datetime(df['DTDepartTheo'], errors='coerce')

# ---- Assign time periods ----
def assign_period(dt):
   if pd.isnull(dt):
       return "Other", -1
   hour = dt.hour
   day = dt.dayofweek  # Monday=0, Sunday=6
   if day == 5:  return "Saturdays", 5
   if day == 6:  return "Sundays", 6
   if 7 <= hour < 9:   return "AM peak", day
   elif 9 <= hour < 16:return "Day period", day
   elif 16 <= hour < 18:return "PM peak", day
   elif 18 <= hour < 24:return "Evening", day
   else:               return "Other", day

df['time_period'], df['weekday_num'] = zip(*df['DTDepartTheo'].apply(assign_period))

# ---- Line segment lists ----
line_18_A = ['74221075','10751361','13610117','01171296','12960374',
            '03740042','00425315','53150056','00560130','01300071',
            '00711127','11271039','10391063','10630086','00863349',
            '33490422','04220801','08011082','10821238','12381441',
```

```
            '14410155','01550091','00910077','00770134','01340702',
            '07023002','30020924','09240686','06860806','08060260']
line_18_R = ['02610807','08070687','06870925','09253003','30030703',
            '07030137','01370078','00780092','00920156','01561440',
            '14401237','12371081','10810800','08000427','04270356',
            '03560087','00871064','10641041','10411128','11280072',
            '00720131','01310057','00575316','53160043','00430375',
            '03751297','12970118','01181362','13621077','10777423']
line_80_A = ['23211310','13101022','10220720','07207166','71660522',
            '05221298','12980122','01220335','03353303','33031037',
            '10371414','14140052','00520630','06302117','21170782',
            '07823301','33010982','09826495','64956497','64972521',
            '25217444']
line_80_R = ['74442522','25226498','64986496','64960983','09833302',
            '33020780','07800353','03530631','06310053','00531415',
            '14151038','10383304','33040334','03340121','01211301',
            '13010523','05237165','71650721','07211023','10231308',
            '13082321']

lines = {
   "line 18 direction A": line_18_A,
   "line 18 direction R": line_18_R,
   "line 80 direction A": line_80_A,
   "line 80 direction R": line_80_R,
}

# ---- Precompute Aggregates ----
sunday_by_period = (
   df[df['weekday_num'] == 6]  # Sunday = 6
     .groupby(["Segment_XXXXYYYY", "time_period"])["segment_speed"]
     .mean()
     .reset_index()
)

sunday_avg = (
   df[df['weekday_num'] == 6]
     .groupby("Segment_XXXXYYYY")["segment_speed"]
     .mean()
)

df_weekdays = df[df['weekday_num'].isin([0,1,2,3,4])]

comparison_periods = [
   ("All Weekday", None),
   ("Weekday AM peak", "AM peak"),
   ("Weekday Day period", "Day period"),
   ("Weekday PM peak", "PM peak"),
   ("Weekday Evening", "Evening"),
   ("Weekday Other", "Other"),
]

# ---- Updated Comparison function with NaN filtering ----
def compute_comparisons(segment_list, label):
   results = []
   for seg in segment_list:
       for name, tp in comparison_periods:
           if tp is None:
               wk_speeds = df_weekdays[df_weekdays["Segment_XXXXYYYY"] == seg]["segment_speed"].dropna()
               sunday_speed = sunday_avg.get(seg, None)
           else:
               wk_speeds = df_weekdays[
                   (df_weekdays["Segment_XXXXYYYY"] == seg) &
                   (df_weekdays["time_period"] == tp)
               ]["segment_speed"].dropna()

               row = sunday_by_period[
                   (sunday_by_period["Segment_XXXXYYYY"] == seg) &
                   (sunday_by_period["time_period"] == tp)
               ]
               sunday_speed = row["segment_speed"].iloc[0] if not row.empty else None

           wk_speed = wk_speeds.mean() if not wk_speeds.empty else None

           # Skip if missing data or zero Sunday speed
           if sunday_speed is None or wk_speed is None or sunday_speed == 0:
               continue  # Skip adding this row
```

```
            pct = (wk_speed / sunday_speed) * 100

            results.append({
                "line": label,
                "segment": seg,
                "comparison_period": name,
                "weekday_avg_speed": wk_speed,
                "sunday_avg_speed": sunday_speed,
                "percent_vs_sunday": pct
            })
    return pd.DataFrame(results)

# ---- Run for all lines ----
all_results = pd.concat(
    [compute_comparisons(segs, label) for label, segs in lines.items()],
    ignore_index=True
)

print(all_results.head(20))

all_results.to_csv("weekday_vs_sunday_results.csv", index=False)
```

Appendix 8:

```
import pandas as pd
import numpy as np

file_path = '/Users/benfall/Desktop/Data Files/OG_trimmed_with_first_stops.csv'
df = pd.read_csv(file_path, encoding='utf-8', low_memory=False)
df.columns = df.columns.str.strip().str.replace('\ufeff', '', regex=False)

df['EcartDepar'] = pd.to_numeric(df['EcartDepar'], errors='coerce')

# ---- Segment lists ----
segments_18A = [
    '74221075','10751361','13610117','01171296','12960374','03740042','00425315','53150056','00560130',
    '01300071','00711127','11271039','10391063','10630086','00863349','33490422','04220801','08011082',
    '10821238','12381441','14410155','01550091','00910077','00770134','01340702','07023002','30020924',
    '09240686','06860806','08060260'
]
segments_18R = [
    '02610807','08070687','06870925','09253003','30030703','07030137','01370078','00780092','00920156',
    '01561440','14401237','12371081','10810800','08000427','04270356','03560087','00871064','10641041',
    '10411128','11280072','00720131','01310057','00575316','53160043','00430375','03751297','12970118',
    '01181362','13621077','10777423'
]
segments_80A = [
    '23211310','13101022','10220720','07207166','71660522','05221298','12980122','01220335','03353303',
    '33031037','10371414','14140052','00520630','06302117','21170782','07823301','33010982','09826495',
    '64956497','64972521','25217444'
]
segments_80R = [
    '74442522','25226498','64986496','64960983','09833302','33020780','07800353','03530631','06310053',
    '00531415','14151038','10383304','33040334','03340121','01211301','13010523','05237165','71650721',
    '07211023','10231308','13082321'
]

all_segments = segments_18A + segments_18R + segments_80A + segments_80R

# Filter to only relevant segments
df_segments = df[df['Segment_XXXXYYYY'].astype(str).isin(all_segments)].copy()

# Function to calculate trimmed ranges for different percentages
def calculate_trimmed_ranges(trim_percentages=[1, 2, 3, 5]):
    results = []

    for trim_pct in trim_percentages:
        lower_pct = trim_pct / 100
        upper_pct = 1 - (trim_pct / 100)

        # Calculate percentiles for each segment
        quantiles = df_segments.groupby('Segment_XXXXYYYY')['EcartDepar'].quantile([lower_pct,
upper_pct]).unstack()
        quantiles.columns = ['min_value', 'max_value']
        quantiles['trim_percentage'] = trim_pct
```

```
        quantiles['range_width'] = quantiles['max_value'] - quantiles['min_value']

        results.append(quantiles.reset_index())

    return pd.concat(results, ignore_index=True)

# Calculate ranges for different trim percentages
trim_results = calculate_trimmed_ranges([1, 2, 3, 5])

# Print results organized by trim percentage
for trim_pct in [1, 2, 3, 5]:
    print(f"\n{'='*60}")
    print(f"TRIMMING {trim_pct}% FROM EACH SIDE")
    print(f"{'='*60}")

    subset = trim_results[trim_results['trim_percentage'] == trim_pct]

    # Overall statistics
    print(f"\nOverall Statistics (all segments):")
    print(f"Min of all min_values: {subset['min_value'].min():.1f}")
    print(f"Max of all max_values: {subset['max_value'].max():.1f}")
    print(f"Average range width: {subset['range_width'].mean():.1f}")
    print(f"Median range width: {subset['range_width'].median():.1f}")

    # Show some example segments
    print(f"\nSample segments (first 10):")
    sample = subset.head(10)[['Segment_XXXXYYYY', 'min_value', 'max_value', 'range_width']]
    print(sample.to_string(index=False, float_format='%.1f'))

# Save detailed results
trim_results.to_csv('/Users/benfall/Desktop/segment_trimmed_ranges.csv', index=False)
print(f"\n\nDetailed results saved to: segment_trimmed_ranges.csv")

# Also show segments with extreme ranges for each trim level
print(f"\n{'='*60}")
print("SEGMENTS WITH WIDEST/NARROWEST RANGES")
print(f"{'='*60}")

for trim_pct in [1, 2, 3, 5]:
    subset = trim_results[trim_results['trim_percentage'] == trim_pct].copy()

    print(f"\nTrim {trim_pct}% - Widest ranges:")
    widest = subset.nlargest(5, 'range_width')[['Segment_XXXXYYYY', 'min_value', 'max_value', 'range_width']]
    print(widest.to_string(index=False, float_format='%.1f'))

    print(f"\nTrim {trim_pct}% - Narrowest ranges:")
    narrowest = subset.nsmallest(5, 'range_width')[['Segment_XXXXYYYY', 'min_value', 'max_value',
'range_width']]
    print(narrowest.to_string(index=False, float_format='%.1f'))
```

Segment_XXXXYYYY	min_value	max_value	trim_percentage	range_width
425315	-97.0	465.0	1	562.0
430375	-173.0	790.3600000000010	1	963.3600000000010
520630	-146.0	819.0	1	965.0
531415	-131.0	608.8399999999970	1	739.8399999999970
560130	-105.0	477.0	1	582.0
575316	-163.0	825.9599999999990	1	988.9599999999990
711127	-124.0	466.84999999999900	1	590.8499999999990
720131	-177.0	770.5899999999930	1	947.5899999999930
770134	-210.0	683.5800000000020	1	893.5800000000020
780092	-98.0	570.7000000000010	1	668.7000000000010
863349	-172.0	462.91999999999800	1	634.9199999999980

871064	-158.0	667.3200000000000	1	825.3200000000000
910077	-189.98	700.0	1	889.98
920156	-111.0	589.609999999997	1	700.609999999997
1171296	-73.30000000000000	374.59999999999900	1	447.8999999999990
1181362	-186.0	728.4399999999990	1	914.4399999999990
1211301	-133.0	790.5299999999950	1	923.5299999999950
1220335	-115.0	696.7400000000020	1	811.7400000000020
1300071	-119.0	473.0	1	592.0
1310057	-163.39	798.3899999999990	1	961.7799999999990
1340702	-163.0	720.7700000000000	1	883.7700000000000
1370078	-98.0	539.7700000000000	1	637.7700000000000
1550091	-190.0	712.0	1	902.0
1561440	-130.0	583.9500000000010	1	713.9500000000010
2610807	-82.0	381.0	1	463.0
3340121	-130.94	737.0	1	867.94
3353303	-120.80000000000000	760.7999999999990	1	881.5999999999990
3530631	-113.0	598.5799999999980	1	711.5799999999980
3560087	-148.0	632.739999999998	1	780.739999999998
3740042	-90.82	459.8200000000000	1	550.6400000000000
3751297	-186.0	802.239999999998	1	988.239999999998
4220801	-145.0	640.4399999999990	1	785.4399999999990
4270356	-109.43	643.2900000000010	1	752.7200000000010
5221298	-110.0	614.1600000000040	1	724.1600000000040
5237165	-178.0	780.0	1	958.0
6302117	-173.0	808.5800000000020	1	981.5800000000020
6310053	-118.0	602.9399999999990	1	720.9399999999990
6860806	-210.0	835.0	1	1045.0
6870925	-92.0	436.2000000000010	1	528.2000000000010
7023002	-159.0	695.0	1	854.0
7030137	-97.0	543.6899999999990	1	640.6899999999990
7207166	-93.0	589.6399999999990	1	682.6399999999990
7211023	-137.0	911.2200000000010	1	1048.2200000000000
7800353	-135.0	549.5200000000040	1	684.5200000000040
7823301	-176.0	868.0999999999990	1	1044.1000000000000
8000427	-133.0	585.1200000000030	1	718.1200000000030
8011082	-166.0	670.0	1	836.0
8060260	-218.0	771.0	1	989.0
8070687	-76.0	386.0	1	462.0
9240686	-193.0	712.25	1	905.25

9253003	-107.0	447.3799999999970	1	554.3799999999970
9826495	-189.0	2077.8800000000000	1	2266.8800000000000
9833302	-121.0	507.0	1	628.0
10220720	-80.0	598.1999999999970	1	678.1999999999970
10231308	-153.0	924.8200000000000	1	1077.8200000000000
10371414	-113.0	818.0	1	931.0
10383304	-97.0	651.0	1	748.0
10391063	-130.0	450.98000000000000	1	580.9800000000000
10411128	-162.0	677.6899999999990	1	839.6899999999990
10630086	-129.0	473.0	1	602.0
10641041	-160.0	677.0	1	837.0
10751361	-83.0	399.6800000000000	1	482.6800000000000
10777423	-190.0	682.0900000000060	1	872.0900000000060
10810800	-160.0	552.2200000000010	1	712.2200000000010
10821238	-175.0	678.0	1	853.0
11271039	-120.80000000000000	456.0	1	576.8
11280072	-165.01	756.0299999999950	1	921.0399999999950
12371081	-140.0	616.0	1	756.0
12381441	-177.0	688.0299999999990	1	865.0299999999990
12960374	-88.0	416.64000000000300	1	504.64000000000300
12970118	-192.78	744.2399999999910	1	937.0199999999910
12980122	-101.71000000000000	657.1299999999970	1	758.8399999999970
13010523	-131.0	835.5299999999990	1	966.5299999999990
13082321	-145.0	879.0	1	1024.0
13101022	-98.0	576.0	1	674.0
13610117	-83.0	407.8500000000020	1	490.8500000000020
13621077	-178.0	740.4300000000110	1	918.4300000000110
14140052	-133.0	806.3000000000030	1	939.3000000000030
14151038	-125.0	602.3200000000000	1	727.3200000000000
14401237	-119.0	603.9900000000020	1	722.9900000000020
14410155	-182.0	690.6000000000060	1	872.6000000000060
21170782	-192.0	823.4000000000020	1	1015.4000000000000
23211310	-48.0	523.6000000000060	1	571.6000000000060
25217444	-188.0	901.0	1	1089.0
25226498	-56.69	541.0	1	597.69
30020924	-174.0	708.8400000000000	1	882.8400000000000
30030703	-107.0	542.0800000000020	1	649.0800000000020
33010982	-184.0	2033.7599999999900	1	2217.7599999999900
33020780	-106.0	528.7400000000020	1	634.7400000000020

33031037	-120.0	809.4300000000040	1	929.4300000000040
33040334	-120.0	676.8899999999990	1	796.8899999999990
33490422	-146.0	546.0	1	692.0
53150056	-104.0	462.3200000000000	1	566.3200000000000
53160043	-180.0	809.2799999999990	1	989.2799999999990
64956497	-191.0	2114.0	1	2305.0
64960983	-107.69	553.0699999999960	1	660.7599999999960
64972521	-199.0	2113.3500000000000	1	2312.3500000000000
64986496	-92.0	541.0	1	633.0
71650721	-119.0	870.9800000000000	1	989.9800000000000
71660522	-103.0	584.7999999999990	1	687.7999999999990
74221075	-75.41	370.23000000000000	1	445.64000000000000
74442522	-30.0	552.0	1	582.0
425315	-80.0	293.0	2	373.0
430375	-150.0	547.6800000000000	2	697.6800000000000
520630	-114.0	610.739999999998	2	724.739999999998
531415	-96.0	441.6800000000000	2	537.6800000000000
560130	-85.5	308.0	2	393.5
575316	-141.0	549.0	2	690.0
711127	-104.0	311.0	2	415.0
720131	-156.0	517.4799999999960	2	673.4799999999960
770134	-181.0	512.0	2	693.0
780092	-81.0	356.6800000000000	2	437.6800000000000
863349	-152.0	332.84000000000000	2	484.84000000000000
871064	-139.0	447.3200000000000	2	586.3200000000000
910077	-161.0	509.91999999999800	2	670.9199999999980
920156	-91.46000000000000	363.0	2	454.46
1171296	-60.0	252.0	2	312.0
1181362	-162.0	538.9599999999990	2	700.9599999999990
1211301	-97.0	631.0200000000000	2	728.0200000000000
1220335	-93.48000000000000	540.4800000000000	2	633.9600000000000
1300071	-101.0	303.40000000000100	2	404.40000000000100
1310057	-143.0	535.0	2	678.0
1340702	-140.0	531.0	2	671.0
1370078	-83.0	323.5400000000010	2	406.5400000000010
1550091	-162.0	504.0	2	666.0
1561440	-113.0	354.0	2	467.0
2610807	-66.0	232.0	2	298.0
3340121	-100.0	572.0	2	672.0

3353303	-91.0	570.5999999999990	2	661.5999999999990
3530631	-86.0	444.0	2	530.0
3560087	-130.74	428.0	2	558.74
3740042	-75.0	293.6399999999990	2	368.6399999999990
3751297	-157.0	554.239999999998	2	711.239999999998
4220801	-122.0	459.9599999999990	2	581.9599999999990
4270356	-91.0	446.72000000000100	2	537.7200000000010
5221298	-84.0	479.0	2	563.0
5237165	-153.0	584.3200000000000	2	737.3200000000000
6302117	-140.0	599.5799999999980	2	739.5799999999980
6310053	-90.0	440.0	2	530.0
6860806	-184.0	593.9800000000000	2	777.9800000000000
6870925	-74.0	262.0	2	336.0
7023002	-137.0	524.0	2	661.0
7030137	-81.0	319.0	2	400.0
7207166	-72.0	453.27999999999900	2	525.2799999999990
7211023	-113.0	712.4399999999990	2	825.4399999999990
7800353	-106.0	392.2599999999980	2	498.2599999999980
7823301	-141.0	671.0	2	812.0
8000427	-115.0	398.0	2	513.0
8011082	-143.0	467.0	2	610.0
8060260	-189.86	579.8600000000010	2	769.7200000000010
8070687	-59.78000000000000	251.0	2	310.78
9240686	-167.0	546.5	2	713.5
9253003	-88.0	270.380000000001	2	358.380000000001
9826495	-154.0	1868.0	2	2022.0
9833302	-95.0	374.0	2	469.0
10220720	-62.0	449.09999999999900	2	511.09999999999900
10231308	-127.0	718.2799999999990	2	845.2799999999990
10371414	-88.0	617.1800000000000	2	705.1800000000000
10383304	-72.0	508.40000000000100	2	580.4000000000020
10391063	-110.0	318.0	2	428.0
10411128	-139.0	471.0	2	610.0
10630086	-110.0	344.40000000000100	2	454.40000000000100
10641041	-137.0	464.0	2	601.0
10751361	-66.0	247.6800000000000	2	313.6800000000000
10777423	-159.0	523.2199999999990	2	682.2199999999990
10810800	-145.0	347.0	2	492.0
10821238	-148.0	479.4599999999990	2	627.4599999999990

11271039	-100.0	322.0	2	422.0
11280072	-142.0	506.0	2	648.0
12371081	-121.06	377.0	2	498.06
12381441	-149.0	490.0	2	639.0
12960374	-72.0	266.7599999999980	2	338.7599999999980
12970118	-163.56	538.0	2	701.56
12980122	-80.0	518.4199999999980	2	598.4199999999980
13010523	-101.0	656.0600000000010	2	757.0600000000010
13082321	-120.0	704.6399999999990	2	824.6399999999990
13101022	-83.0	404.0	2	487.0
13610117	-66.0	256.5400000000010	2	322.5400000000010
13621077	-153.0	555.9599999999990	2	708.9599999999990
14140052	-106.30000000000000	603.0	2	709.3
14151038	-92.0	446.0	2	538.0
14401237	-102.0	374.0	2	476.0
14410155	-155.0	489.2999999999990	2	644.2999999999990
21170782	-156.0	631.7999999999990	2	787.7999999999990
23211310	-38.0	384.2999999999990	2	422.2999999999990
25217444	-141.5	725.5	2	867.0
25226498	-40.0	417.0	2	457.0
30020924	-149.0	540.0	2	689.0
30030703	-86.54000000000000	314.5400000000010	2	401.08000000000100
33010982	-144.0	1828.380000000000	2	1972.380000000000
33020780	-79.0	392.0	2	471.0
33031037	-94.0	600.0	2	694.0
33040334	-90.0	524.0	2	614.0
33490422	-125.0	402.9399999999990	2	527.9399999999990
53150056	-87.0	293.3200000000000	2	380.3200000000000
53160043	-157.0	533.2799999999990	2	690.2799999999990
64956497	-156.0	1905.0400000000000	2	2061.040000000000
64960983	-83.0	400.380000000001	2	483.380000000001
64972521	-160.0	1831.7000000000000	2	1991.7000000000000
64986496	-70.0	404.0	2	474.0
71650721	-97.0	699.9599999999990	2	796.9599999999990
71660522	-80.0	452.0	2	532.0
74221075	-60.0	220.45999999999900	2	280.4599999999990
74442522	-16.0	418.3600000000010	2	434.3600000000010
425315	-68.0	245.0	3	313.0
430375	-137.0	451.02000000000000	3	588.0200000000000

520630	-95.0	515.1100000000010	3	610.1100000000010
531415	-78.0	369.2599999999980	3	447.2599999999980
560130	-75.0	258.0	3	333.0
575316	-128.0	451.0	3	579.0
711127	-92.0	260.0	3	352.0
720131	-144.0	417.0	3	561.0
770134	-160.37	432.0	3	592.37
780092	-72.0	281.0	3	353.0
863349	-141.0	268.0	3	409.0
871064	-127.0	351.0	3	478.0
910077	-145.0	428.9399999999990	3	573.9399999999990
920156	-82.0	282.0	3	364.0
1171296	-52.0	205.0	3	257.0
1181362	-145.0	455.0	3	600.0
1211301	-81.0	533.5299999999990	3	614.5299999999990
1220335	-78.0	460.0	3	538.0
1300071	-90.0	254.0	3	344.0
1310057	-130.17000000000000	435.0	3	565.1700000000000
1340702	-128.0	451.3100000000010	3	579.3100000000010
1370078	-75.0	258.0	3	333.0
1550091	-147.0	423.0	3	570.0
1561440	-103.0	273.0	3	376.0
2610807	-59.0	176.0	3	235.0
3340121	-81.0	481.0	3	562.0
3353303	-76.0	485.0	3	561.0
3530631	-72.0	368.0	3	440.0
3560087	-120.0	338.6100000000010	3	458.6100000000010
3740042	-65.0	241.0	3	306.0
3751297	-142.0	464.0	3	606.0
4220801	-108.0	389.0	3	497.0
4270356	-79.0	364.0	3	443.0
5221298	-71.0	405.0	3	476.0
5237165	-135.0	499.0	3	634.0
6302117	-118.0	507.0	3	625.0
6310053	-74.82000000000010	368.0	3	442.82000000000000
6860806	-169.0	488.97000000000100	3	657.9700000000010
6870925	-66.0	202.0	3	268.0
7023002	-123.60000000000000	445.59999999999900	3	569.1999999999990
7030137	-72.0	250.0	3	322.0

7207166	-60.0	382.0	3	442.0
7211023	-98.0	595.0	3	693.0
7800353	-88.39000000000000	324.0	3	412.39
7823301	-120.0	571.0	3	691.0
8000427	-104.0	312.0	3	416.0
8011082	-127.0	389.0	3	516.0
8060260	-174.0	482.0	3	656.0
8070687	-52.0	194.0	3	246.0
9240686	-152.0	464.0	3	616.0
9253003	-78.0	209.0	3	287.0
9826495	-132.0	1761.0	3	1893.0
9833302	-80.88	312.0	3	392.88
10220720	-52.0	374.0	3	426.0
10231308	-108.0	605.0	3	713.0
10371414	-70.0	528.0	3	598.0
10383304	-53.0	435.0	3	488.0
10391063	-99.0	264.0	3	363.0
10411128	-126.0	380.0	3	506.0
10630086	-98.0	285.0	3	383.0
10641041	-126.0	374.619999999999	3	500.619999999999
10751361	-55.52000000000000	184.0	3	239.52000000000000
10777423	-144.0	454.6100000000010	3	598.6100000000010
10810800	-135.0	263.0	3	398.0
10821238	-132.0	399.0	3	531.0
11271039	-89.0	269.0	3	358.0
11280072	-129.0	416.02999999999900	3	545.0299999999990
12371081	-110.0	284.0	3	394.0
12381441	-133.0	412.09000000000000	3	545.0900000000000
12960374	-64.0	223.0	3	287.0
12970118	-149.0	457.0	3	606.0
12980122	-68.0	436.130000000001	3	504.130000000001
13010523	-82.0	554.0	3	636.0
13082321	-101.0	606.0	3	707.0
13101022	-73.0	324.0	3	397.0
13610117	-56.0	197.0	3	253.0
13621077	-138.0	471.0	3	609.0
14140052	-88.0	513.0	3	601.0
14151038	-73.0	376.9599999999990	3	449.9599999999990
14401237	-91.97000000000000	293.0	3	384.97

14410155	-141.0	412.4500000000010	3	553.4500000000010
21170782	-136.0	534.0	3	670.0
23211310	-29.0	309.0	3	338.0
25217444	-114.0	640.0	3	754.0
25226498	-29.07000000000010	348.0	3	377.07000000000000
30020924	-136.0	456.0	3	592.0
30030703	-77.0	242.0	3	319.0
33010982	-124.0	1723.0	3	1847.0
33020780	-61.22000000000000	327.0	3	388.22
33031037	-77.0	507.0	3	584.0
33040334	-71.0	440.0	3	511.0
33490422	-112.0	336.41000000000000	3	448.41000000000000
53150056	-75.0	245.0	3	320.0
53160043	-143.0	439.0	3	582.0
64956497	-133.0	1791.5600000000000	3	1924.5600000000000
64960983	-70.0	321.0	3	391.0
64972521	-138.0	1022.3000000000000	3	1160.3000000000000
64986496	-56.0	326.0	3	382.0
71650721	-80.0	593.9399999999990	3	673.9399999999990
71660522	-66.0	388.0	3	454.0
74221075	-52.0	165.45999999999900	3	217.45999999999900
74442522	-8.0	357.77000000000000	3	365.77000000000000
425315	-55.0	192.0	5	247.0
430375	-116.0	350.0	5	466.0
520630	-73.0	416.0	5	489.0
531415	-58.0	292.0	5	350.0
560130	-60.0	207.0	5	267.0
575316	-109.0	346.0	5	455.0
711127	-77.0	208.0	5	285.0
720131	-125.0	315.0	5	440.0
770134	-135.0	351.0	5	486.0
780092	-59.0	210.0	5	269.0
863349	-124.0	210.0	5	334.0
871064	-110.0	273.7999999999990	5	383.7999999999990
910077	-124.0	346.0	5	470.0
920156	-69.0	213.0	5	282.0
1171296	-41.0	158.0	5	199.0
1181362	-123.0	369.3999999999980	5	492.3999999999980
1211301	-57.0	418.0	5	475.0

1220335	-58.0	369.0	5	427.0
1300071	-74.5	201.5	5	276.0
1310057	-112.0	336.0	5	448.0
1340702	-108.85000000000000	366.0	5	474.8500000000000
1370078	-63.0	188.0	5	251.0
1550091	-129.0	339.0	5	468.0
1561440	-90.0	201.0	5	291.0
2610807	-50.0	120.0	5	170.0
3340121	-57.0	383.0	5	440.0
3353303	-56.0	389.0	5	445.0
3530631	-53.0	296.0	5	349.0
3560087	-103.0	267.0	5	370.0
3740042	-53.0	190.0	5	243.0
3751297	-121.0	366.0	5	487.0
4220801	-90.0	313.0	5	403.0
4270356	-63.0	296.0	5	359.0
5221298	-54.0	326.59999999999900	5	380.59999999999900
5237165	-112.0	389.0	5	501.0
6302117	-97.0	406.4500000000010	5	503.4500000000010
6310053	-54.0	292.7000000000010	5	346.7000000000010
6860806	-146.0	388.0	5	534.0
6870925	-56.0	145.0	5	201.0
7023002	-104.0	371.0	5	475.0
7030137	-61.0	177.0	5	238.0
7207166	-47.0	305.2000000000010	5	352.2000000000010
7211023	-73.0	481.0	5	554.0
7800353	-68.0	256.0	5	324.0
7823301	-95.0	463.0	5	558.0
8000427	-91.0	239.79999999999900	5	330.7999999999990
8011082	-108.0	313.0	5	421.0
8060260	-152.0	383.1499999999980	5	535.1499999999980
8070687	-41.0	139.0	5	180.0
9240686	-130.0	375.0	5	505.0
9253003	-66.0	145.0	5	211.0
9826495	-106.0	874.4000000000020	5	980.4000000000020
9833302	-62.0	250.0	5	312.0
10220720	-38.25	300.0	5	338.25
10231308	-83.0	488.0	5	571.0
10371414	-50.0	422.0	5	472.0

10383304	-35.0	351.0	5	386.0
10391063	-84.0	214.89999999999800	5	298.8999999999980
10411128	-109.0	297.4500000000010	5	406.4500000000010
10630086	-83.0	233.0	5	316.0
10641041	-107.0	291.0	5	398.0
10751361	-44.0	135.20000000000100	5	179.20000000000100
10777423	-125.0	369.0	5	494.0
10810800	-121.0	196.0	5	317.0
10821238	-113.0	322.0	5	435.0
11271039	-73.0	216.0	5	289.0
11280072	-111.0	318.0	5	429.0
12371081	-96.0	215.0	5	311.0
12381441	-113.0	330.0	5	443.0
12960374	-52.0	172.0	5	224.0
12970118	-127.0	367.0	5	494.0
12980122	-50.0	352.0	5	402.0
13010523	-60.0	440.0	5	500.0
13082321	-75.0	493.09999999999900	5	568.0999999999990
13101022	-57.0	246.0	5	303.0
13610117	-43.0	149.0	5	192.0
13621077	-115.45000000000000	383.0	5	498.45000000000000
14140052	-67.0	410.0	5	477.0
14151038	-52.0	306.0	5	358.0
14401237	-77.0	227.0	5	304.0
14410155	-121.0	330.0	5	451.0
21170782	-111.0	432.0	5	543.0
23211310	-18.0	242.75	5	260.75
25217444	-81.0	542.0	5	623.0
25226498	-16.0	290.0	5	306.0
30020924	-115.0	374.0	5	489.0
30030703	-65.0	169.0	5	234.0
33010982	-100.0	875.9000000000020	5	975.9000000000020
33020780	-44.0	262.0	5	306.0
33031037	-57.0	408.0	5	465.0
33040334	-51.0	360.0	5	411.0
33490422	-97.0	273.0	5	370.0
53150056	-61.0	193.0	5	254.0
53160043	-123.0	336.0	5	459.0
64956497	-106.0	864.5999999999990	5	970.5999999999990

64960983	-53.0	251.0	5	304.0
64972521	-107.0	631.0	5	738.0
64986496	-41.0	260.0	5	301.0
71650721	-57.0	479.0	5	536.0
71660522	-51.0	316.0	5	367.0
74221075	-43.0	122.0	5	165.0
74442522	4.0	304.0	5	300.0

```
import pandas as pd
import numpy as np

file_path = '/Users/benfall/Desktop/Data Files/OG_trimmed_with_first_stops.csv'
df = pd.read_csv(file_path, encoding='utf-8', low_memory=False)
df.columns = df.columns.str.strip().str.replace('\ufeff', '', regex=False)

df['EcartDepar'] = pd.to_numeric(df['EcartDepar'], errors='coerce')

# ---- Segment lists ----
segments_18A = [
   '74221075','10751361','13610117','01171296','12960374','03740042','00425315','53150056','00560130',
   '01300071','00711127','11271039','10391063','10630086','00863349','33490422','04220801','08011082',
   '10821238','12381441','14410155','01550091','00910077','00770134','01340702','07023002','30020924',
   '09240686','06860806','08060260'
]
segments_18R = [
   '02610807','08070687','06870925','09253003','30030703','07030137','01370078','00780092','00920156',
   '01561440','14401237','12371081','10810800','08000427','04270356','03560087','00871064','10641041',
   '10411128','11280072','00720131','01310057','00575316','53160043','00430375','03751297','12970118',
   '01181362','13621077','10777423'
]
segments_80A = [
   '23211310','13101022','10220720','07207166','71660522','05221298','12980122','01220335','03353303',
   '33031037','10371414','14140052','00520630','06302117','21170782','07823301','33010982','09826495',
   '64956497','64972521','25217444'
]
segments_80R = [
   '74442522','25226498','64986496','64960983','09833302','33020780','07800353','03530631','06310053',
   '00531415','14151038','10383304','33040334','03340121','01211301','13010523','05237165','71650721',
   '07211023','10231308','13082321'
]

all_segments = segments_18A + segments_18R + segments_80A + segments_80R

# Filter to only relevant segments
df_segments = df[df['Segment_XXXXYYYY'].astype(str).isin(all_segments)].copy()

# ------------------------------------------------------------
# Function to find required symmetric trim percentage per segment
# ------------------------------------------------------------
def find_min_trim_for_bounds(lower=-300, upper=300, step=0.1):
   results = []

   for seg, values in df_segments.groupby('Segment_XXXXYYYY')['EcartDepar']:
       values = values.dropna().to_numpy()

       if len(values) == 0:
           results.append([seg, np.nan, np.nan, np.nan, np.nan])
           continue

       # Search from 0% trim up to 20% (or 50 if needed)
       trim_needed = None
       for trim_pct in np.arange(0, 50.1, step):
           low_q = np.percentile(values, trim_pct)
           high_q = np.percentile(values, 100 - trim_pct)

           if low_q > lower and high_q < upper:
               trim_needed = trim_pct
               break
```

```
        # If not possible, record 50% as "cap"
        if trim_needed is None:
            trim_needed = 50

        results.append([seg, trim_needed, low_q, high_q, high_q - low_q])

    return pd.DataFrame(results, columns=['Segment', 'TrimPct', 'FinalMin', 'FinalMax', 'RangeWidth'])

# Run the calculation
trim_requirements = find_min_trim_for_bounds()

# Save results
output_path = '/Users/benfall/Desktop/segment_trim_requirements.csv'
trim_requirements.to_csv(output_path, index=False)

print("Results saved:", output_path)
print("\nSample output:")
print(trim_requirements.head(10))
```

Segment	TrimPct	FinalMin	FinalMax	RangeWidth
425315	2.0	-80.0	293.0	373.0
430375	**7.2**	-102.0	298.0	400.0
520630	**10.4**	-41.0	297.0	338.0
531415	4.800000000000000	-60.0	299.6160000000020	359.6160000000020
560130	2.2	-83.0	294.0	377.0
575316	**6.9**	-96.0	298.0	394.0
711127	2.2	-101.0	295.0	396.0
720131	**5.6000000000000000**	-120.0	298.0	418.0
770134	**7.2**	-116.0	298.0	414.0
780092	2.7	-74.0	299.0	373.0
863349	2.5	-146.0	294.2999999999990	440.2999999999990
871064	4.1000000000000000	-117.0	299.0	416.0
910077	**6.9**	-109.0	299.0	408.0
920156	2.8000000000000000	-84.0	294.0	378.0
1171296	1.4000000000000000	-68.0	297.619999999999	365.619999999999
1181362	**8.0**	-101.0	299.0	400.0
1211301	**10.4**	-27.0	298.0	325.0
1220335	**8.4**	-38.0	299.0	337.0
1300071	2.1	-99.0	296.0	395.0
1310057	**6.4**	-102.0	299.4959999999990	401.4959999999990
1340702	**7.9**	-89.0	299.0	388.0
1370078	2.3000000000000000	-81.0	297.67099999999800	378.67099999999800
1550091	**6.5**	-116.0	299.0	415.0
1561440	2.6	-107.0	297.6700000000020	404.6700000000020
2610807	1.4000000000000000	-73.0	289.0	362.0
3340121	**8.9**	-33.0	299.0	332.0
3353303	**9.4**	-32.0	299.5199999999970	331.5199999999970

3530631	4.9	-54.0	297.0	351.0
3560087	4.0	-111.0	297.0	408.0
3740042	1.9000000000000000	-76.0	299.0	375.0
3751297	**7.9**	-100.0	298.0	398.0
4220801	**5.6000000000000000**	-86.0	299.0	385.0
4270356	4.9	-64.0	299.0	363.0
5221298	**6.2**	-46.0	299.0	345.0
5237165	**8.3**	-88.0	299.0	387.0
6302117	**9.9**	-62.0	298.270999999997	360.270999999997
6310053	4.7	-57.0	299.0	356.0
6860806	**8.6**	-121.0	297.0	418.0
6870925	1.7000000000000000	-78.0	289.23999999999800	367.23999999999800
7023002	**8.200000000000000**	-80.0	299.0	379.0
7030137	2.3000000000000000	-77.0	292.0	369.0
7207166	**5.2**	-45.0	299.0	344.0
7211023	**13.0**	-27.0	299.0	326.0
7800353	3.7	-80.0	296.3809999999980	376.3809999999980
7823301	**12.8**	-43.0	299.0	342.0
8000427	3.3000000000000000	-102.0	296.0	398.0
8011082	**5.5**	-104.0	298.0	402.0
8060260	**8.3**	-128.0	299.0690000000000	427.0690000000000
8070687	1.5	-66.0	297.0049999999970	363.0049999999970
9240686	**8.4**	-105.0	299.0	404.0
9253003	1.7000000000000000	-92.0	298.0	390.0
9826495	**17.5**	-28.0	299.0	327.0
9833302	3.3000000000000000	-78.0	297.0	375.0
10220720	**5.1000000000000000**	-38.0	297.0	335.0
10231308	**13.8**	-29.0	299.0	328.0
10371414	**11.3**	-15.0	299.0	314.0
10383304	**7.4**	-20.0	299.0	319.0
10391063	2.3000000000000000	-106.0	298.0	404.0
10411128	5.0	-109.0	297.4500000000010	406.4500000000010
10630086	2.7	-101.0	298.1899999999990	399.1899999999990
10641041	4.800000000000000	-109.0	297.0	406.0
10751361	1.6	-71.54400000000000	294.63200000000500	366.17600000000500
10777423	**8.200000000000000**	-101.0	299.0	400.0
10810800	2.6	-139.0	291.08600000000300	430.08600000000300
10821238	**5.9**	-106.0	299.0	405.0
11271039	2.4000000000000000	-94.0	293.52000000000000	387.52000000000000

11280072	**5.7**	-106.0	299.0	405.0
12371081	2.8000000000000000	-112.0	298.4840000000000	410.4840000000000
12381441	**6.2**	-105.0	299.0	404.0
12960374	1.7000000000000000	-76.0	289.39600000000100	365.39600000000100
12970118	**7.800000000000000**	-105.0	299.0	404.0
12980122	**7.300000000000000**	-37.0	299.0	336.0
13010523	**11.100000000000000**	-23.0	299.0	322.0
13082321	**14.8**	-16.0	299.0	315.0
13101022	3.5	-68.0	299.0550000000000	367.0550000000000
13610117	1.6	-70.03200000000000	294.0	364.032
13621077	**8.8**	-88.0	298.0	386.0
14140052	**10.5**	-33.0	298.8250000000010	331.8250000000010
14151038	**5.2**	-50.0	299.0	349.0
14401237	3.0	-91.97000000000000	293.0	384.97
14410155	**6.2**	-113.0	296.9300000000000	409.9300000000000
21170782	**10.9**	-70.0	299.0	369.0
23211310	3.3000000000000000	-27.79500000000000	298.0	325.79500000000000
25217444	**20.5**	21.0	299.0	278.0
25226498	4.6000000000000000	-18.0	299.0	317.0
30020924	**8.4**	-90.0	299.0	389.0
30030703	2.2	-84.0	297.89400000000000	381.89400000000000
33010982	**17.2**	-25.0	299.0	324.0
33020780	3.7	-54.0	298.0	352.0
33031037	**10.200000000000000**	-29.0	298.0620000000020	327.0620000000020
33040334	**7.7**	-33.0	299.0	332.0
33490422	4.0	-103.88	299.880000000001	403.760000000001
53150056	1.9000000000000000	-89.0	299.9539999999980	388.9539999999980
53160043	**6.4**	-113.0	299.0	412.0
64956497	**18.0**	-26.0	299.0	325.0
64960983	3.5	-65.0	297.4150000000010	362.4150000000010
64972521	**17.400000000000000**	-25.0	299.0	324.0
64986496	3.6	-51.0	299.0	350.0
71650721	**13.600000000000000**	-7.0	299.0	306.0
71660522	**5.9**	-45.0	297.0	342.0
74221075	1.4000000000000000	-68.0	297.9480000000000	365.9480000000000
74442522	**5.300000000000000**	5.0	299.0	294.0

Works Cited:

1. Office cantonal de la statistique et des études – État de Genève. *Statistique Genève*. https://statistique.ge.ch/. Accessed 11 July 2025.
2. Hale, David, and Brian Cronin. "Traffic Bottlenecks: Identification and Solutions." *Federal Highway Administration*, November 2016, https://www.fhwa.dot.gov/publications/research/operations/16064/003.cfm.
3. Francisco Garrido-Valenzuela, Diego Cruz, Marina Dragicevic, Alejandro Schmidt, Jaime Moya, Sebastián Tamblay, Juan C. Herrera, Juan C. Muñoz, Identifying and visualizing operational bottlenecks and Quick win opportunities for improving bus performance in public transport systems, Transportation Research Part A: Policy and Practice, Volume 164, 2022, Pages 324-336, ISSN 0965-8564, https://doi.org/10.1016/j.tra.2022.08.005. (https://www.sciencedirect.com/science/article/pii/S0965856422002014)
4. Bucknell, C., Schmidt, A., Cruz, D., & Muñoz, J. C. (2017). Identifying and Visualizing Congestion Bottlenecks with Automated Vehicle Location Systems: Application to Transantiago, Chile. *Transportation Research Record*, *2649*(1), 61-70. https://doi.org/10.3141/2649-07 (Original work published 2017)
5. Brands, Ties, et al. "Automatic Bottleneck Detection Using AVL Data: A Case Study in Amsterdam." *Proceedings of the Conference on Advanced Systems in Public Transport (CASPT)*, 23–25 July 2018, Brisbane, Australia. Article 82. *TU Delft Repository*, https://repository.tudelft.nl/record/uuid:dfda29f4-afcf-4852-aadf-2c6e1f78333d.
6. Transports Publics Genevois (TPG). "Rapport Annuel de Gestion 2024." TPG, May 2025, www.tpg.ch/sites/default/files/2025-05/tpg_Rapport%20annuel%20de%20gestion%202024_WEB.pdf. Accessed 14 July 2025.

Bibliography:

1. "Geneva Sees Record Number of Cross-Border Permits Issued in 2024." Swissinfo.ch, 1 Feb. 2025, swissinfo.ch/eng/workplace-switzerland/geneva-sees-record-number-of-cross-border-permits-issued-in-2024/88812679. Accessed 11 July 2025.
2. "Bottlenecks and Delays: Which Swiss Cities Have the Worst Traffic?" *The Local*, 10 Feb. 2022, www.thelocal.ch/20220210/bottlenecks-and-delays-which-swiss-cities-have-the-worst-traffic?utm_source=chatgpt.com.
3. Li, Zhi-Chun et al. "Fifty years of the bottleneck model: A bibliometric review and future research directions." Transportation Research Part B: Methodological vol. 139 (2020): 311–342. doi:10.1016/j.trb.2020.06.009,[6] https://pmc.ncbi.nlm.nih.gov/articles/PMC7333998/
4. TomTom. "TomTom Traffic Index 2025." TomTom Newsroom, 7 Jan. 2025, www.tomtom.com/newsroom/press-releases/general/605041959/tomtom-traffic-index-2025/.Accessed 14 July 2025.

[6] It is important to note that congestion is a relatively new concept, with the entire automotive industry and its mass adoption being a fairly modern phenomenon. This article discusses the history of the term "bottleneck," coined in 1969 by Prof. William S. Vickrey.

5. Xu, T., Barman, S., Levin, M.W., Chen, R., & Li, T. (2022). Integrating public transit signal priority into max-pressure signal control: Methodology and simulation study on a downtown network. Transportation Research Part C: Emerging Technologies, Vol. 138, 103614. https://www.sciencedirect.com/science/article/pii/S0968090X22000602?ref=pdf_download&fr=RR-2&rr=95f1ac34ea973b51
6. Francisco Nilso De Brito-Filho, Francisco Moraes Oliveira-Neto, Analysis of the causal effects of exclusive bus lanes on performance indicators of public transport corridors, Transportation Research Part A: Policy and Practice, Volume 196, 2025, 104485, ISSN 0965-8564, https://doi.org/10.1016/j.tra.2025.104485. (https://www.sciencedirect.com/science/article/pii/S0965856425001132)
7. Kartoidjojo, Bobby. "Tram bottlenecks in Amsterdam after implementation of the Noord-Zuidlijn." Bachelor of Science Thesis, Delft University of Technology, 2020. yufeiyuan.eu/wp-content/uploads/2020/06/2020-06-2912.pdf. Accessed 24 July 2025.
8. Aemmer, Zack, Andisheh Ranjbari, and Don MacKenzie. "Measurement and Classification of Transit Delays Using GTFS-RT Data." Public Transport, vol. 14, no. 2, 2022, pp. 263–285. Springer, https://doi.org/10.1007/s12469-022-00291-7. Accessed 28 July 2025.

Used PyCharm, Jupyter Notebook, Visual Studio Code Data Wrangler, Python (Pandas, Matplotlib, numpy
I also made use of ChatGPT o4 and 5 for rapid Python code improvements.

Useful links:

1. https://www.waze.com/live-map/ (unfortunately doesn't allow to backtrack data)
2. https://www.tpg.ch/fr/lignes/80
3. https://www.tpg.ch/fr/lignes/18
4. https://map.sitg.ge.ch/app/?mapresources=MOBILITE_GENERAL