Graz University of Technology

# Transport Planning related Data generated by Mobility Apps

**MASTER THESIS**

prepared by

Alexander Schaffenberger, BSc

Supervisor:

Univ.-Prof. Dr.-Ing. Martin Fellendorf

Institute of Highway Engineering and Transport Planning

Supervising assistant:

Dipl.-Ing. Michael Cik

Institute of Highway Engineering and Transport Planning

Graz, October 21st, 2019

**Statutory Declaration**

I declare that I have authored this thesis independently, that I have not used other than the declared sources / resources, and that I have explicitly marked all material which has been quoted either literally or by content from the used sources. This document is identical with the electronic version uploaded via TUGRAZonline.

Graz, ______________________ ____________________________

Alexander Schaffenberger, BSc.

**Eidesstattliche Erklärung**

Ich erkläre an Eides Statt, dass ich die vorliegende Arbeit selbstständig verfasst, andere als die angegebenen Quellen/Hilfsmittel nicht benutzt und die den benutzten Quellen wörtliche und inhaltlich entnommene Stellen als solche kenntlich gemacht habe. Das in TUGRAZonline hochgeladene Textdokument ist mit der vorliegenden Masterarbeit/Diplomarbeit identisch.

Graz, ______________________ ____________________________

Alexander Schaffenberger, BSc.

# Acknowledgment

Primarily I would like to thank my family for their support over the years. Without them such a study would not have been possible.

Furthermore, I would like to thank all the teachers at the technical university for bringing me closer to the content of civil engineering and for investing a lot of time and effort in this.

I would also like to thank my fellow students for the support and the great time we have spent together.

For providing the data I used in this thesis I would like to thank Fluidtime Data Services GmbH, Holding Graz – Kommunale Dienstleistungen GmbH and Kapsch TrafficCom AG.

And finally, thanks to Univ.-Prof. Dr.-Ing. Martin Fellendorf and Dipl.-Ing. Michael Cik for supporting me while writing this master thesis.

Graz University of Technology

**Institut für Straßen- und Verkehrswesen**

**Vorstand Univ.-Prof. Dr.-Ing. Martin Fellendorf**

Rechbauerstraße 12
A-8010 Graz

Tel.: +43 (0) 316 873-6221
Fax: +43 (0) 316 873-4199
isv@tugraz.at

DVR: 008 1833 UID: ATU 574 77 929

**Task for the master thesis**

**by Alexander Schaffenberger, BSc**

Graz, 04.04.2018

# Information provided from mobility apps for transport planning purposes

## Problem statement

Many European cities grow steadily in population while rural areas are shrinking. The spatial concentration of residents leads to additional commuter traffic accompanied with various traffic problems. The cities must deal with increased traffic amount. Travelers are interested in new mobility solutions and the cities must be prepared. One solution will be the reduction of individual car trips by shared mobility solutions. European cities as well as cities worldwide are seeking for opportunities to enable intermodal trips and convince residents to reduce car ownership. The sharing community will open up a new market of sharing vehicles instead of owning them.

Various companies ranging from public transport operators, car manufacturers to start-ups with industry support try to get into this mobility market searching for Mobility-as-a-Service (MaaS) as an opportunity. Some companies offer only a part of Mobility-as-a-Service, while others develop complete solutions. Transport services between public and private transport are connected to shared mobility to provide intermodal trips. For this connection the data of different modes are collected and processed. Companies offering Mobility-as-a-Service have work together with various transport operators to provide multiple modes.

The enormous development of the internet and mobile devices provides new possibilities for mobility services. There are numerous mobility apps on the market. So far, they are mainly used for navigation purposes, such as finding the shortest journey for various travel modes. Several mobility apps offer additional functionality such as ticketing and pre-booking to simplify a journey. The customer's data is gathered and can be used to make journeys more efficient. However, data protection rules must be considered, especially the upcoming General Data Protection Regulation EU Directive 2016/679.

The additional data provided by journey planners and more advanced mobility apps may provide data which may be useful for transport planning purposes. Transport planners for urban areas try to provide solutions which make best usage of existing or future transport systems seeking for suitable compromises between accessibility, travel times, travel comfort and journey cost for users as well as operators. This thesis will focus on the opportunities of data provided by upcoming mobility data for transport planning purposes.

## Tasks

The master thesis is structured in two parts.
In the first part existing mobility data from Graz should be analyzed. The data to be examined comes from the mobility app "quando" and should be provided from the Holding Graz. Quando is a first generation mobility app (journey planner with real-time public transport information). In the second part the additional data opportunities of new mobility apps should be investigated by literature research.

The following list contains main items of the master thesis which will be adopted according to the work progress:

- Data collection of mobility-app quando and Graz transport operator (passenger counts, comparison time table, …)
- Statistical analysis of quando data: correlation between usage of quando services and travel demand in public transport
- Data analysis of mobility service platforms (moovel, wegfinder, Whim, VAO, …)
- Data analysis of urban traffic management systems (traffic signals, traffic congestion, parking and city tolling, …)
- Conclusions: priority list for integrated mobility management systems

The work will be done in close co-operation with Kapsch TrafficCom AG.

The student agrees, that all data provided by stakeholders such as Holding Graz, Kapsch and TU Graz will only be used for conducting this master thesis and data usage will be compliant with data protection regulations.

The master thesis will be provided in two printed copies accompanied with an electronic version of the master thesis, the data being collected and any presentations.

Univ.-Prof. Dr.-Ing. Martin Fellendorf
Tel. 0316 873 - 6220
martin.fellendorf@tugraz.at
Institute of Highway Engineering and Transport Planning
TU Graz
Supervisor

Dipl.-Ing. Michael Cik
Tel. 0316 873 - 6224
michael.cik@tugraz.at
Institute of Highway Engineering and Transport Planning
TU Graz
Supervising assistant

## Abstract

**Transport Planning related Data generated by Mobility Apps**

100 pages, 75 figures, 36 tables

This master thesis deals with information gained from mobility apps that may be of interest for transport planning purposes. For this, monitor and route requests of the mobility app “qando Graz” were evaluated and analyzed. Temporal distributions, frequencies and locations of requests and public transport assignments to the line network, which were created with the traffic model of Graz, were considered. In addition, the qando Graz data assigned to the lines of the Holding Graz were statistically compared with real occurring passenger counts in the public transport network of the city of Graz in order to be able to make a statement about the correlation between these different data.

The analysis of the data leads to the assumption that a large part of the users of this app are students at the universities in Graz or related employees. Due to this one-sided distribution of the user group, the qando data does not reflect the real occurring public transport amount and there is no correlation between the assigned qando data and the real occurring passenger counts after performing the statistical comparison.

# Kurzfassung

**Verkehrsplanungsdaten erzeugt durch Mobilitätsapplikationen**

100 Seiten, 75 Abbildungen, 36 Tabellen

Diese Masterarbeit beschäftigt sich mit Informationen, welche aus Mobilitätsapps gewonnen werden und für die Verkehrsplanung von Interesse sein können. Dafür wurden Monitor- und Routenabfragen von der Mobilitätsapp „qando Graz" ausgewertet und analysiert. Zeitliche Verteilungen, Häufigkeiten und Lage von Anfragen und Umlegungen des öffentlichen Verkehrs auf das Liniennetz, welche mit dem Verkehrsmodell von Graz erstellt wurden, wurden betrachtet. Zusätzlich wurden die auf die Linien der Holding Graz umgelegten qando Graz Daten statistisch mit wirklich auftretenden Fahrgastzahlen aus dem öffentlichen Verkehrsnetz der Stadt Graz verglichen, um eine Aussage über die Korrelation zwischen diesen unterschiedlichen Daten treffen zu können.

Die Analyse der Daten führt zu der Annahme, dass ein großer Teil der Appnutzer Studenten an den Universitäten oder zugehörige Bedienstete sind. Aufgrund der einseitigen Verteilung der Nutzergruppe spiegeln die qando Daten nicht den wirklich auftretenden öffentlichen Verkehr wieder und es gibt keine Korrelation zwischen den umgelegten qando Daten und den wirklich auftretenden Fahrgastzahlen nach der Durchführung des statistischen Vergleichs.

# Table of contents

# List of figures

# List of tables

# Index of abbreviations

| | |
|---|---|
| API | Application Programming Interface |
| FCD | Floating car data |
| GPS | Global positioning system |
| GUI | Graphical user interface |
| HGL | Holding Graz Linien |
| ID | Identification |
| KFU | Karl Franzens University |
| LKH | Landeskrankenhaus (state hospital) |
| MaaS | Mobility-as-a-Service |
| MOTUS | Mobility and tourism in urban areas |
| ÖBB | Österreichische Bundesbahnen (Austrian railways) |
| POI | Points of interest |
| TU | Technical University |

# 1 Introduction

In the cities, due to the constantly increasing population and the concentration of these into the urban area, the traffic amount increases more and more and this leads to problems on transportation networks. Transport planners are faced with these occurring issues and must provide new solutions for the cities. They are trying to find solutions that include the best possible use of available and future transport systems for users and transport operators. One possible solution is to reduce individual car trips and enable intermodal trips by using shared mobility. With this approach the behavior of owning a car is changed and it comes to a new market where vehicles are shared and not owned. Mobile devices offer the possibility for the implementation of these mobility services with the help of mobility apps.

Because of the massive development of the internet and mobile devices, the field around mobility apps has experienced a big upswing in the recent years and is an indispensable part of modern mobility. Travelers adapt their travel behaviors based on the information provided by these mobility apps and companies are increasingly realizing the potential of these new mobility platforms, are entering this market and offering solutions. Some companies offer complete solutions for the use of mobility services, like shared mobility, which include everything from the planning to the ticketing of the desired services and others offer only a part of this, like pure route planning.

The aim of this master thesis on the one hand is to find out which possibilities are available for using data coming from mobility apps for transport planning purposes and on the other hand the aim is to evaluate existing data of the mobility app “qando Graz”. From mobility apps users obtain information about their desired trips, like for example how to get from A to B or which means of transport nearby can be used for the trip. However, these apps do not only contain the information for the users, but also for planning in the fields of transportation the user requests can be evaluated and assessed. The stored user requests thus have to some extent a second life and can contribute transport planners to find effective solutions.

This thesis is structured as follows. Chapter 2 contains a literature research on the internet on currently available mobility apps. Furthermore, opportunities on data extraction for traffic management and transport planning purposes are investigated. The second part of this thesis, starting with chapter 3, deals with an investigation of data analysis of the mobility app “qando”, which is implemented in several cities in Austria. In this chapter the functionality and data availability of the qando implementation in Graz is described. The following chapter 4 includes the results of the analysis and these are presented in figures and tables and to that described. Temporal distributions, frequencies and locations of the requests, origin-destination information via spider matrices and public transport assignments are the included evaluations. At the end of this chapter a statistical comparison is made between the assigned qando data and the real occurring passenger counts in the public transport network of Graz. The entire thesis is summarized in chapter 5 and a short prospect into the future is made.

# 2 Literature research

The subsequent literature research only exists of a pure internet research. No literature is given from which the information must be taken. The fields of mobility apps and new mobility solutions are fast moving and there are always innovations coming out. Information out of older books would not be state of the art. The research includes existing mobility apps, what these are able to and how transport planner can use data out of these mobility apps for transport planning.

## 2.1 What is a mobility app?

A mobility app is an offer from a transport operator or a company for passengers which are on the way with the services of public transport or using various means of transport and it is a part of modern mobility solutions. Pathfinding from A to B should be simplified using these apps and a variety of solutions should demonstrate the user how the path of the chosen route can be taken. The range, in which the mobility apps provide their services, extends from urban areas over larger regions to the possibility for using them in a whole country. The expansion levels of such apps are very different and thus some apps include more functionality than others. Main functions which mobility apps may contain are:

- **Navigation:** Finding the optimal route for one or more means of transport
- **Booking:** Reservation of transport facility at a given time
- **Payment:** Paying a ticket or a service

For the calculation of a route a starting and an ending point are needed. These points can be stops, points of interest (POI), addresses or the current position of the user. Stops and POI are deposited in the apps and can be entered in the search field or selected on the integrated map. Often, intermediate points, on which the desired route should run, can also be specified. The time when the route should be started or ended can be determined by specifying a departure or arrival time.

After entering this information, an algorithm is run which calculates the optimal route. In order to find the personally optimally adapted route, the route calculation can be carried out according to different optimization criteria. It can be prioritized between the fastest, the shortest, the cheapest and the most comfortable route or even those with the fewest changes. As a result, the various possibilities, with which the trip can be traveled, are displayed. In addition to the textual display with departure and arrival times and the means of transport used, the selected route is also displayed on a map from the start to the end point. Thus, the exact paths and stop points of the stations can be retraced. For better organization, the possibility to send the information of the selected route via an export function to the used calendar exists.

When displaying various options, single mode trips, if present, with only one mean of transport or mostly multimodal combinations of multiple means of transport are displayed. Such means of transport can for example be busses, trams, trains or subways. The offer depends on the existing infrastructure. Most of the time footpaths to the stop, while changing the stop point and from the last stop to the destination are included in the calculated routes. Thereby, in the calculation is taken care to avoid large time holes when changing vehicles and thus to reduce the waiting times to a minimum. User-defined settings can be specified before calculating a route. This includes settings like the maximum walking time, whether the means of transport are equipped for wheelchairs, whether there is an elevator or no stairs when changing vehicles or which means of transport should be considered for the calculation and which of them should be excluded.

To calculate the route the timetable with the average travel times is deposited in the background. It is assumed, that the means of transport are always on time. This procedure is time independent. To always transmit current departure and arrival times and to detect deviations from the timetable, real-time data is used if it is available. Thus, delays of vehicles and the information about possibly unreachable connections can be passed on to the user and he can react accordingly. Also, these time dependent travel times can be used to respond to the current congestion level in the transport network. Disruptions and unexpected events are poor for route planning and are counteracted with the help of text messages or alternatively calculated routes.

Modern mobility apps also offer the possibility to use mobility as a service. The apps contain shared mobility services which are supplied by private transport providers and can be booked and paid directly via the app. Shared mobility services can for example be car sharing, bike sharing, scooter sharing or taxis. More and more apps also include a payment function to directly purchase the ticket or the used services. Fares are calculated simultaneously to the route. The price depends on the means of transport used and can be very different.

Beside the possibility of calculating a route there is also the opportunity to display only the departure times of public transport lines. This information can be retrieved quickly, if the desired route is already known. In this case, as with the route calculation, real-time data is used if it is available.

Mobility app functionality can be very different and depends on the desired content of the provider and the existing infrastructure in the field of application. The next chapter will take a closer look at several usual mobility apps from different areas and countries.

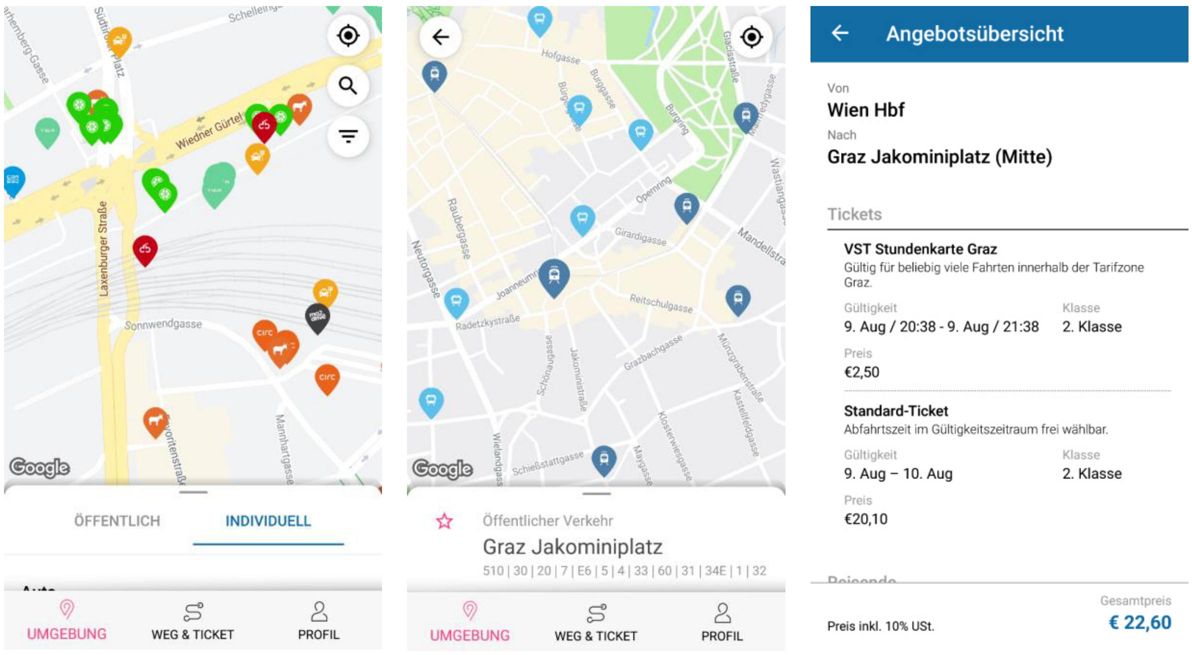
## 2.2 Common mobility apps

In this chapter several mobility apps are shown and compared in terms of the range of functions. A variety of such apps can be found on the market and the number is growing. Because of the large number of apps many differences occur between those as the focus varies from city to city or project to project. Those in this thesis considered apps are from local importance, in this case Graz, or they are in the pot with the market leaders. The considered mobility apps are:

- **wegfinder:** A journey planner with ticket reservation and payment for public transport and booking of shared mobility services in Austria
- **Whim:** A fully Mobility-as-a-Service (MaaS) solution, which includes different means of transport
- **REACH NOW:** A trip planner with included payment option for shared mobility services
- **qando Graz:** A journey planner with real-time information for the zone 101 in Graz, but no ticketing functionality
- **BusBahnBim:** An Austrian-wide trip planner without an opportunity to purchase tickets

### 2.2.1 wegfinder

wegfinder is a multiple award-winning mobility app for the whole country of Austria and is operated by iMobility GmbH. The iMobility GmbH is a 100 % subsidiary of the Austrian railways (ÖBB – Österreichische Bundesbahnen) and interested to make mobility easily available in Austria. For that, the public transport providers of Austria are combined with private mobility providers. This results in a combination of public transport and shared mobility. From trains and busses over car and bike sharing right down to flexible E-Scooters is everything available in the app. At the moment over 30 partners (as of 09.08.2019) provide their services. wegfinder is a schedule, a map of the surroundings, a route planner and a ticket shop in one. The routes are calculated by considering the available means of transport, no matter if city or countryside. The payment system includes countrywide tickets of all transport associations and the ÖBB. For all other services from private mobility providers, like car sharing or E-Scooter sharing, an extra registration at the partner's must be performed (iMobility GmbH, 2019).



Angebotsübersicht

Von
Wien Hbf
Nach
Graz Jakominiplatz (Mitte)

Tickets

VST Stundenkarte Graz
Gültig für beliebig viele Fahrten innerhalb der Tarifzone Graz.

| Gültigkeit | Klasse |
|---|---|
| 9. Aug / 20:38 - 9. Aug / 21:38 | 2. Klasse |

Preis
€2,50

Standard-Ticket
Abfahrtszeit im Gültigkeitszeitraum frei wählbar.

| Gültigkeit | Klasse |
|---|---|
| 9. Aug – 10. Aug | 2. Klasse |

Preis
€20,10

Preis inkl. 10% USt.
Gesamtpreis
€ 22,60

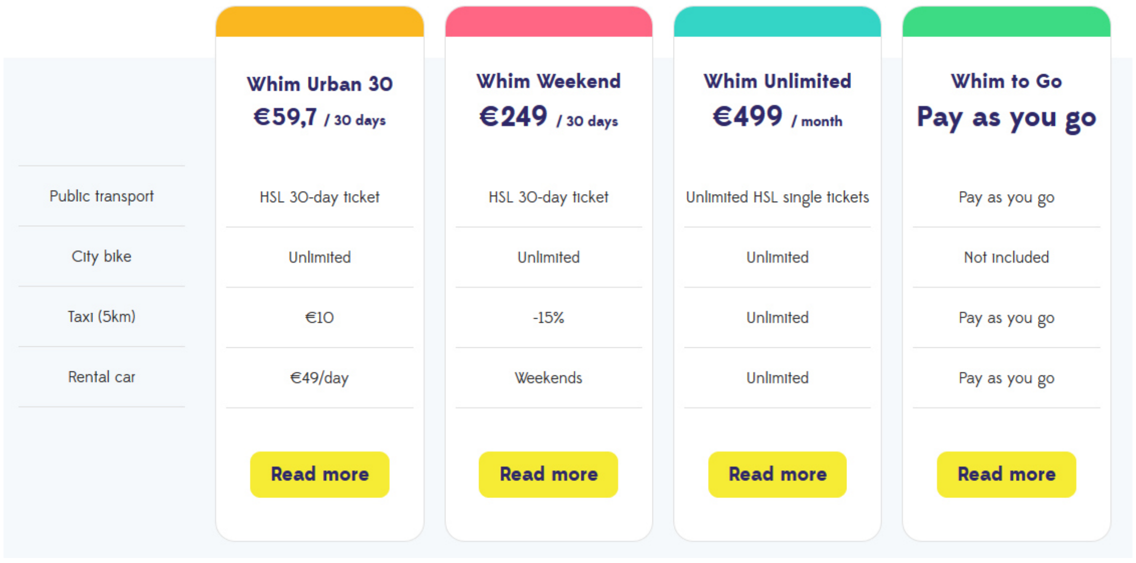
**Figure 1: wegfinder app with shared mobility services in Vienna (left), tram and bus stops in Graz (middle) and payment system (right)**

### 2.2.2 Whim

Whim is an award-winning mobility app from the first Mobility-as-a-Service operator MaaS Global. Maas Global believes that it makes no sense for car ownership anymore and so they developed with Whim a cheap alternative to own a car. It is a fully MaaS solution and includes different means of transport, like public transport, cars and bikes, whereby the availability depends on the area in which the service is needed. The idea behind is to cover the mobility needs in a single service. Route planning, booking and paying (MaaS Global, 2019).

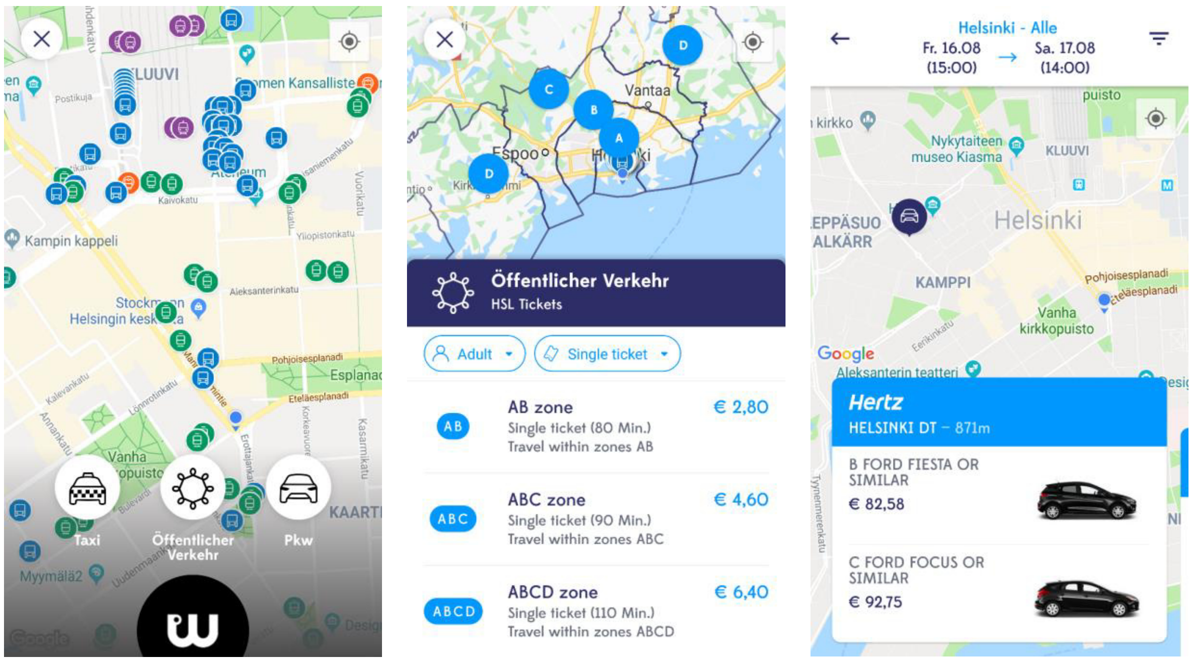
Mobility-as-a-service is a concept where you go away from personally owned mobility to consume services of mobility solutions by combining public and private transport modes.

Users have the choice between four different paying options when using Whim. The simplest way to pay for the services is to use "pay as you go". By utilizing this payment method no monthly fees are deducted, the trips are payed separately. The other paying options are valid for one month and include various possibilities depending on the price. In Figure 2 the paying options are listed (MaaS Global, 2019).



| | Whim Urban 30 €59,7 / 30 days | Whim Weekend €249 / 30 days | Whim Unlimited €499 / month | Whim to Go Pay as you go |
|---|---|---|---|---|
| Public transport | HSL 30-day ticket | HSL 30-day ticket | Unlimited HSL single tickets | Pay as you go |
| City bike | Unlimited | Unlimited | Unlimited | Not included |
| Taxi (5km) | €10 | -15% | Unlimited | Pay as you go |
| Rental car | €49/day | Weekends | Unlimited | Pay as you go |
| | Read more | Read more | Read more | Read more |

Figure 2: Paying options when using Whim, Source: https://whimapp.com/plans/ [10.08.2019]



| | | |
|---|---|---|
| AB | AB zone<br>Single ticket (80 Min.)<br>Travel within zones AB | € 2,80 |
| ABC | ABC zone<br>Single ticket (90 Min.)<br>Travel within zones ABC | € 4,60 |
| ABCD | ABCD zone<br>Single ticket (110 Min.)<br>Travel within zones ABCD | € 6,40 |

B FORD FIESTA OR SIMILAR
€ 82,58

C FORD FOCUS OR SIMILAR
€ 92,75

Figure 3: Whim app with public transport stops in Helsinki (left), fare zones around Helsinki (middle) and car rental in Helsinki (right)

### 2.2.3 REACH NOW

REACH NOW, former moovel, is the result of bundling mobility services from the BMW Group and the Daimler AG. Germany and Austria are the countries in which the app is available, and nine cities are incorporated right now (by 10.08.2019) in the App. Eigth cities are from Germany and Vienna is the only Austrian city. Local transport connections can be searched in the app, but there is no ticketing option for buying them. Long-distance traffic is also not completely supported. REACH NOW includes the mobility services from four private transport providers, one taxi service, on bike sharing service and two car sharing services. All but one can be booked and paid directly in the app (moovel Group GmbH, 2019).

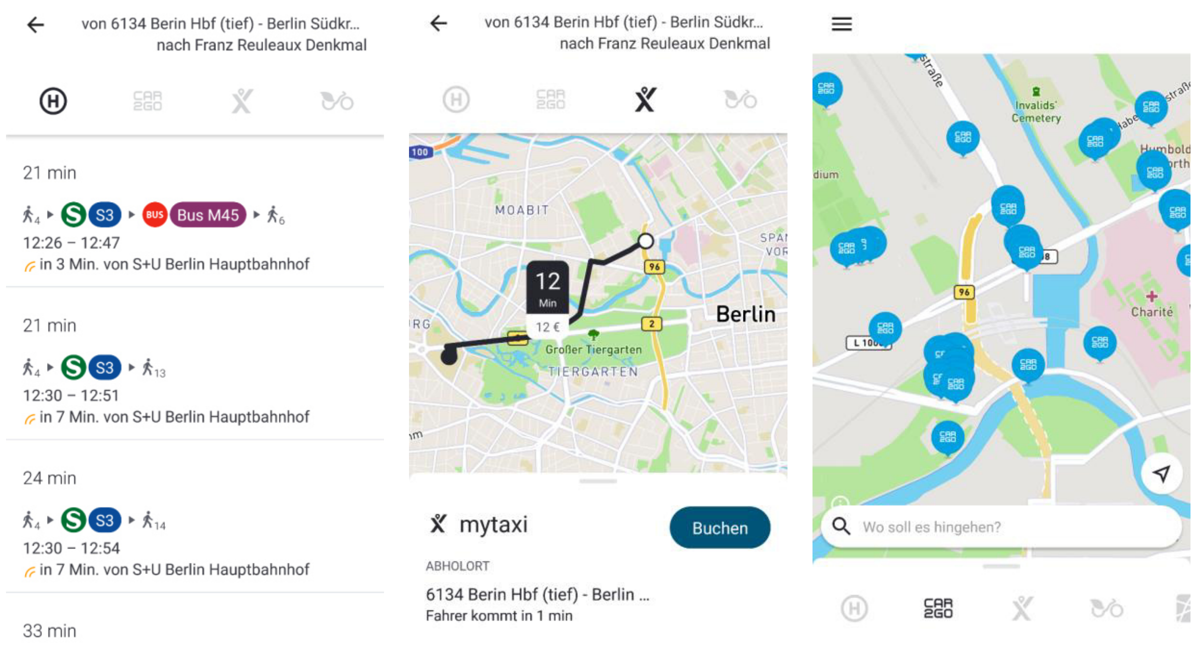
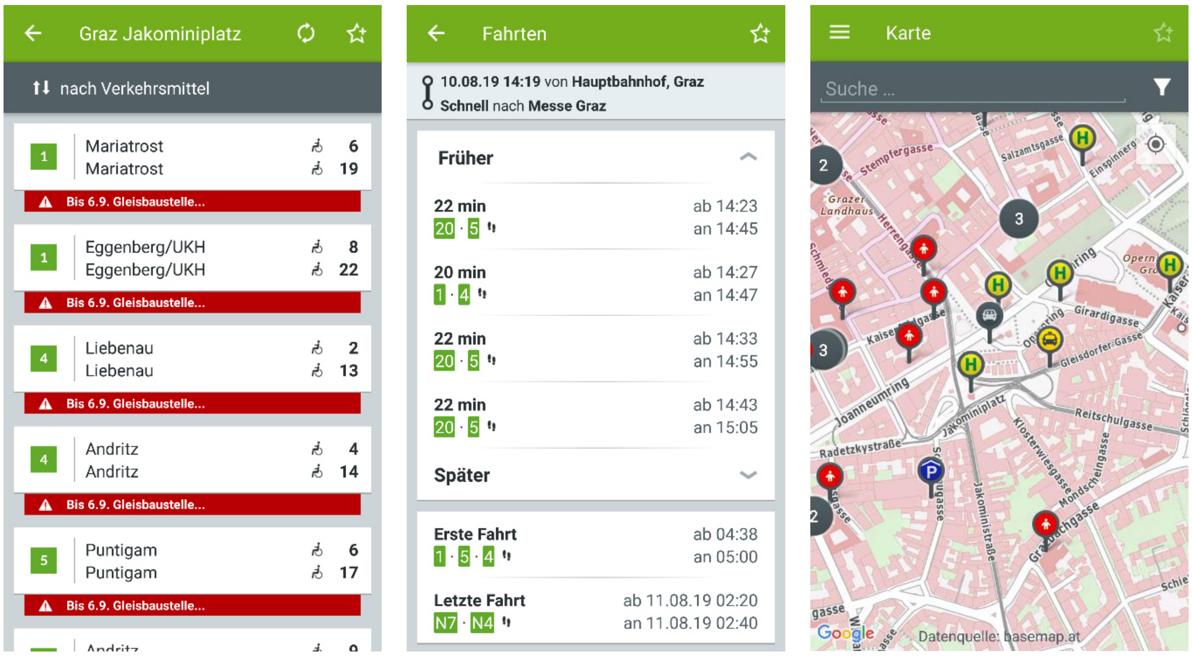

von 6134 Berin Hbf (tief) - Berlin Südkr...
nach Franz Reuleaux Denkmal
21 min
S3
Bus M45
12:26 – 12:47
in 3 Min. von S+U Berlin Hauptbahnhof
21 min
S3
12:30 – 12:51
in 7 Min. von S+U Berlin Hauptbahnhof
24 min
S3
12:30 – 12:54
in 7 Min. von S+U Berlin Hauptbahnhof
33 min
von 6134 Berin Hbf (tief) - Berlin Südkr...
nach Franz Reuleaux Denkmal
MOABIT
12
Min
12 €
Berlin
Großer Tiergarten
TIERGARTEN
mytaxi
Buchen
ABHOLORT
6134 Berin Hbf (tief) - Berlin ...
Fahrer kommt in 1 min
Invalids' Cemetery
Charité
Wo soll es hingehen?

**Figure 4: REACH NOW app with route planning in Berlin (left), taxi service in Berlin (middle) and car sharing around Berlin main railway station (right)**

### 2.2.4 qando Graz

qando is a mobile passenger information in Graz and is provided by the public transport operator Holding Graz Linien (HGL). Compared to the apps which have been presented so far, qando offers no possibilities for shared mobility, only information about public transport can be put out. Beside the information about the nearest stops, departure times at stops and the desired routes, schedules, possible disruptions and diversions can be displayed in the app. The zone 101 is in general the field of application, but it is possible to make route requests for the whole state Styria. Within the zone 101 departure times of trams and busses are shown in real time. The opportunity to buy tickets is not existing in the app (Holding Graz - Kommunale Dienstleistungen GmbH, 2019).



Graz Jakominiplatz

nach Verkehrsmittel

| | | | |
|---|---|---|---|
| 1 | Mariatrost | ♿ | 6 |
| | Mariatrost | ♿ | 19 |
| | Bis 6.9. Gleisbaustelle... | | |
| 1 | Eggenberg/UKH | ♿ | 8 |
| | Eggenberg/UKH | ♿ | 22 |
| | Bis 6.9. Gleisbaustelle... | | |
| 4 | Liebenau | ♿ | 2 |
| | Liebenau | ♿ | 13 |
| | Bis 6.9. Gleisbaustelle... | | |
| 4 | Andritz | ♿ | 4 |
| | Andritz | ♿ | 14 |
| | Bis 6.9. Gleisbaustelle... | | |
| 5 | Puntigam | ♿ | 6 |
| | Puntigam | ♿ | 17 |
| | Bis 6.9. Gleisbaustelle... | | |

Fahrten

10.08.19 14:19 von Hauptbahnhof, Graz
Schnell nach Messe Graz

Früher

| | |
|---|---|
| 22 min | ab 14:23 |
| 20 · 5 | an 14:45 |
| 20 min | ab 14:27 |
| 1 · 4 | an 14:47 |
| 22 min | ab 14:33 |
| 20 · 5 | an 14:55 |
| 22 min | ab 14:43 |
| 20 · 5 | an 15:05 |

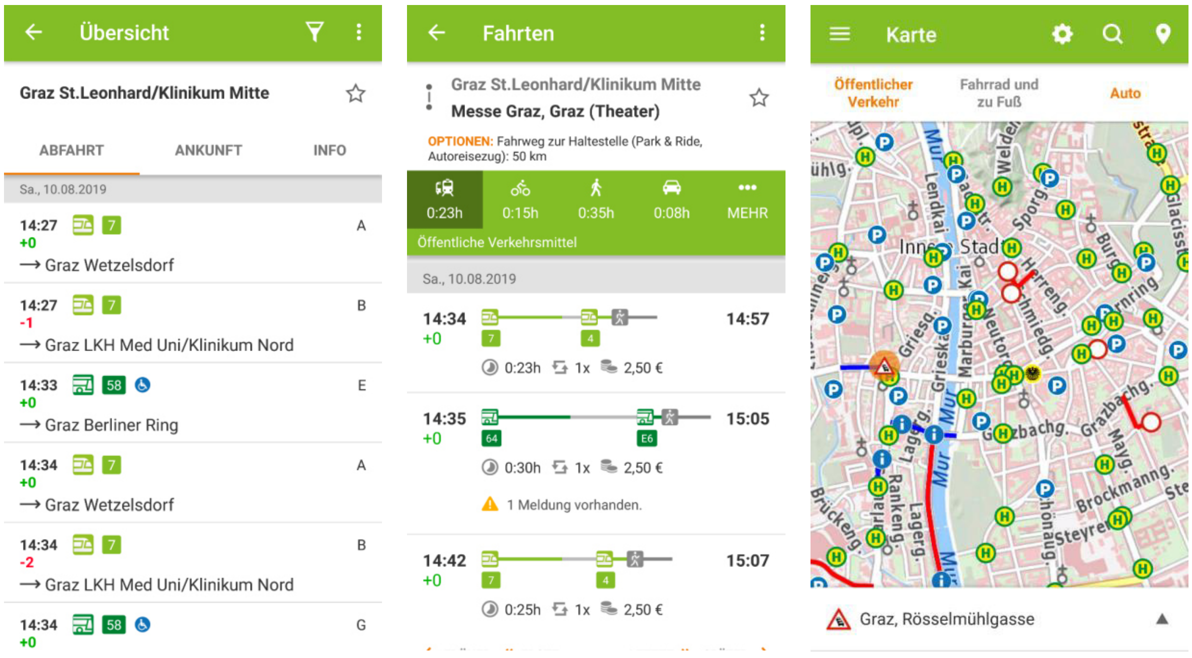
Später

| | |
|---|---|
| Erste Fahrt | ab 04:38 |
| 1 · 5 · 4 | an 05:00 |
| Letzte Fahrt | ab 11.08.19 02:20 |
| N7 · N4 | an 11.08.19 02:40 |

**Figure 5: qando Graz app with departure times for Jakominiplatz and disruption messages (left), route planning in Graz (middle) and stops and POI around Jakominiplatz (right)**

### 2.2.5 BusBahnBim

BusBahnBim is a mobility app to show information about the schedule and is provided by Verkehrsverbund Steiermark GmbH. The app is very similar to qando, but it has more functions for route planning. Departure times of stops and routes can be shown too, and also in real-time if this information is available. Compared to qando, BusBahnBim considers multimodal traffic. If it is possible, the routes are calculated for public transport, cars, bikes, footpath, Bike & Ride and Park & Ride. No services of shared mobility are included, no means of transport can be borrowed, the app assumes that means of transport, like cars and bikes, are the own property of the user. The main focus lays on connections in Styria, but route planning can be done for the whole country Austria. BusBahnBim has no ticketing system, but prices are displayed for routes in Styria (Verkehrsverbund Steiermark GmbH, 2019).



Übersicht

Graz St.Leonhard/Klinikum Mitte

ABFAHRT ANKUNFT INFO

Sa., 10.08.2019

14:27 +0 7 A
⟶ Graz Wetzelsdorf

14:27 -1 7 B
⟶ Graz LKH Med Uni/Klinikum Nord

14:33 +0 58 E
⟶ Graz Berliner Ring

14:34 +0 7 A
⟶ Graz Wetzelsdorf

14:34 -2 7 B
⟶ Graz LKH Med Uni/Klinikum Nord

14:34 +0 58 G

Fahrten

Graz St.Leonhard/Klinikum Mitte
Messe Graz, Graz (Theater)

OPTIONEN: Fahrweg zur Haltestelle (Park & Ride, Autoreisezug): 50 km

0:23h 0:15h 0:35h 0:08h MEHR

Öffentliche Verkehrsmittel

Sa., 10.08.2019

14:34 +0 7 4 14:57
0:23h 1x 2,50 €

14:35 +0 64 E6 15:05
0:30h 1x 2,50 €
1 Meldung vorhanden.

14:42 +0 7 4 15:07
0:25h 1x 2,50 €

Karte

Öffentlicher Verkehr | Fahrrad und zu Fuß | Auto

Graz, Rösselmühlgasse

**Figure 6: BusBahnBim app with departure times for St. Leonhard/Klinikum Mitte (left), route planning in Graz (middle) and stops, parking areas and information about traffic situation in the inner city (right)**

## 2.3 Comparison of mobility apps

In the following two tables the mobility apps are compared with each other. Because of lack of space the comparison is divided into two tables, the first one contains the apps which include services of shared mobility and the second one contains the two journey planners for public transport only without ticketing functionality. Last time updated August 2019.

**Table 1: Comparison of mobility apps, Source pictures: https://play.google.com/store [09.08.2019]**

| | wegfinder | Whim | REACH NOW |
|---|---|---|---|
| | | | REACH NOW |
| **Monitor function** | The app does not have its own search field for stops. It is possible to select a stop on the map to show the departing lines with the belonging departure times of the stop. | The app does not have its own search field for stops. It is possible to select a stop on the map to show the departing lines with the belonging departure times of the stop. | The app does not have its own search field for stops. It is possible to select a stop on the map to show the departing lines with the belonging departure times of the stop. |
| **Route function** | Routes can be calculated with the desired departure or arrival time. The possible connections are then listed, including connections using shared mobility, and can be displayed on the map. Thereby not desired means of transport can be filtered out so that only connections are shown which the user intends to use. | Routes can be calculated with the desired departure or arrival time. The route request has to be carried out with the actual position first, only then a different route start can be entered. The possible connections are then listed, not including connections using shared mobility, and can be displayed on the map. | Routes can be calculated with the desired departure or arrival time. The possible connections are then listed, including connections using shared mobility, and can be displayed on the map. |
| **Real-time data** | The departure times are shown in real-time if they are available. | The departure times are shown in real-time if they are available. | The departure times are shown in real-time if they are available. |
| **Taxi** | Taxi points are displayed on the map of the surrounding area and are considered for route planning. Call taxis and Uber services are available. | A own taxi function exists in the app. The start and endpoints of the route can be defined right away and so a taxi is requested directly within the app. | A own taxi function exists in the app. The pick-up location can be selected by using the needle on the map and so a taxi is requested directly within the app. |
| **Shared mobility** | The position of the different means of transport of the shared mobility services are shown on the map of the sourrounding area. When calculating the route, the possible connections for shared mobility are included. The services for car sharing, bike sharing, E-Scooter and motor scooter are available. | The shared mobility services have a own function in each case and are shown individual and not combined all together on the map. When calculating the route, the connections for shared mobility are not included. The services for car sharing, car rental and bike sharing are available. | The shared mobility services have a own function in each case and are shown individual and not combined all together on the map. When calculating the route, the possible connections for shared mobility are included. The services for car sharing and bike sharing are available. |

| | | | |
|---|---|---|---|
| **Own mobility** | Routes are also calculated for trips with own car or own bike. | Routes are not calculated for trips with own car or own bike. | Routes are not calculated for trips with own car or own bike. |
| **Payment system** | Trips by public transport can be paid directly in the app for all of Austria. To pay the services for shared mobility the user is redirected to the respective partner's page. Credit and debit cards are accepted for payment. | All services can be paid directly in the app. Credit cards are accepted for payment. | There is no opportunity to buy tickets for public transport. The shared mobility services, except one car sharing service, can be paid directly in the app. Credit cards and PayPal are accepted for payment. |
| **Abonnement** | There is no possibility to purchase the services for a longer period of time. | There is a possibility to purchase tickets that are valid for one month. Different ticket options exist (compare chapter 2.2.2). | There is no possibility to purchase the services for a longer period of time. |
| **Export route to calendar** | Route information can be exported to the calendar. | There is no possibility to export route information to the calendar, but public transport tickets can be purchased via calendar entries. | There is no possibility to export route information to the calendar. |

**Table 2: Comparison of mobility apps, Source pictures: https://play.google.com/store [09.08.2019]**

| | **qando Graz** | **BusBahnBim** |
|---|---|---|
| | | |
| **Monitor function** | The app does have its own monitor function to search stops and therefore to show the departing lines with the belonging departure times of the stop. By clicking on a stop on the map the same information can be seen as well. | The app does have its own monitor function to search stops and therefore to show the departing lines with the belonging departure times of the stop. By clicking on a stop on the map the same information can be seen as well. |
| **Route function** | Routes can be calculated with the desired departure or arrival time. The possible connections are then listed and can be displayed on the map. Thereby not desired means of transport can be filtered out in the app options so that only connections are shown which the user intends to use. | Routes can be calculated with the desired departure or arrival time. The possible connections are then listed and can be displayed on the map. Intermediate destinations can be used for route planning. Not desired means of transport can be filtered out so that only connections are shown which the user intends to use. Options to the means of transport, like velocity of bikes, can be adjusted. |
| **Real-time data** | The departure times are shown in real-time if they are available. | The departure times are shown in real-time if they are available. |
| **Taxi** | Not available in the app. | Not available in the app. |
| **Shared mobility** | Not available in the app. | Not available in the app. |

| | | |
|---|---|---|
| **Own mobility** | Routes are not calculated for trips with own car or own bike. | Routes are also calculated for trips with own car or own bike and by foot. |
| **Payment system** | Not available in the app. | Not available in the app. |
| **Abonnement** | Not available in the app. | Not available in the app. |
| **Export route to calendar** | Route information can be exported to the calendar. | Route information can be exported to the calendar. |

After a closer look at these five apps can be seen that the functions are partly very similar, but sometimes very different. The focus of different workgroups and cities lays on other features of the app. Also, the actual existing infrastructure in cities or countries influences the offers of an app.

A comparison of the five apps can not be made easily as they differ by functionality and purpose of application. Whim provides direct payment of public transport tickets and shared vehicles, while wegfinder requires users to link to other sites when ordering shared vehicles. REACH NOW is still in its infacy after a relaunch from moovel, the software stability has still to be improved. qando and BusBahnBim are journey planners for public transport in Graz respectively the state of Styria and not equipped with a payment system or shared mobility services. Thus, they are limited in region and functionality, but both are aimed to provide public transport users in that service area with real-time travel information.

## 2.4 Which data are interesting for users and transport planners?

This next part deals with public transport data from mobility apps and traffic management systems which could be of importance for the users or the transport planners.

Actually, every mobility app for public transport includes the function of a route planner with which a route from A to B can be calculated. The user of this app receives the information on how he can travel this trip, by tram, by bus or rather by foot. Various variants are listed, and the user can choose the most suitable one based on the displayed connections. After creating a route request, this one is stored in a database. In many cases these data atrophies in the stores without being evaluated. But what can be done with these data from the perspective of a transport planner to improve public transport and make it more efficient? If the app is known among the population, many such requests are made each day. The most varied requests are sent, routes over long distances from the outskirts to the center or just a short trip within the city center. Thus, at the end of the day a collection of route requests yield of which origin and destination are known. This information about the departure and destination locations can be summarized to traffic zones and used to create origin-destination matrices, which are then assigned to a public transport network in a traffic model. Such a public transport assignment will also be shown later in this thesis in chapter 4.2.5 including the theoretical fundamentals and in chapter 4.2.6 containing the assignment results. With the help of a model, it is possible to calculate volumes for lines out of the route requests received. The quality of the result depends heavily on the users of the app. If the user group is very one-sided, then the result from the model will also be very one-sided and does not reflect reality. If the user group is well distributed, interesting results can be achieved.

Of course, there are other ways to receive the volumes on the lines. If the app contains the possibility of recording the routes via the user’s global positioning system (GPS), the route and time are known and thereof it can also be suggested to the lines used. Also, in the presence of a payment system it can be closed to the lines used. After purchasing a ticket for a selected route, it can be assumed that the user will also take this journey.

Modern mobility apps increasingly include shared mobility services, various services of bike-sharing, car-sharing, E-scooter-sharing etc. are already available. The user indicates from where to where he wants to travel and can view the possible shared mobility services. If there is no station of the desired service available nearby, this service cannot be used. If there is an accumulation of requests in the same area, this shows a higher attractiveness of that area. With the help of these requests it is possible to estimate the attractiveness of an area, whether in the future an expansion of the system would make sense or not.

Car-sharing services are already widely used and can be utilized by people who otherwise have no car available. Just unlock the car via the mobility app and you are ready to go. As simple as this sounds it is afterwards often more difficult to find a suitable parking space near the destination, be it at a shopping center, at an event location or at a friend’s place of residence etc. So, it would be advantageous for the user if the app could display available parking spaces near the destination. This falls in the field of smart parking and several companies are working on such systems. A system that recognizes available parking spaces is among other things available from Bosch and is called community-based parking. Vehicles equipped with on-board sensors detect available parking spaces while passing by and send this information to a cloud. This information is processed and can be forwarded to car drivers, which are searching for an available parking space. In the searching car available parking spaces in the destination area are therefore displayed. Thus, no senseless wandering to find a parking space, but purposefully reaching an available parking space. This saves nerves, time and money. How the parking space detection and finding works is shown in Figure 7 (Robert Bosch GmbH, 2019).

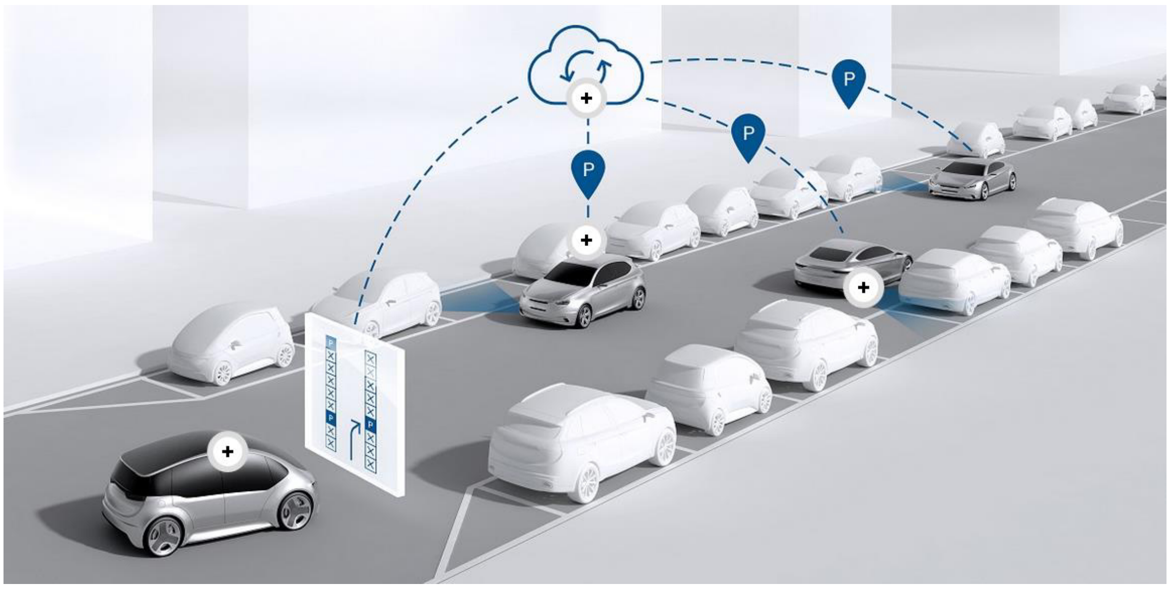


**Figure 7: Community-based parking principle (Robert Bosch GmbH, 2019)**

With such a system, car-sharing users would be able to find an available parking space faster and this would certainly make car-sharing in cities more attractive.

An important information for the transport operator is knowing how many passengers are on the lines. For this purpose, systems, which automatically measure the number of passengers, are used. Different systems are available on the market, but only one system of the German company iris-GmbH is declared in detail. 3D technology is used for automatic passenger counting and can be installed in trains or buses. The sensors are mounted above the entrances and use the time-of-flight technology. In this technology, infrared light, not visible for the human eye, is emitted and reflected by objects, like human beings. The time span, which the light needs from emission to reception, is called time of flight. 3D images, which recognize passengers and even bicycles or wheelchairs, are the result (iris-GmbH infrared & intelligent sensors, 2019).

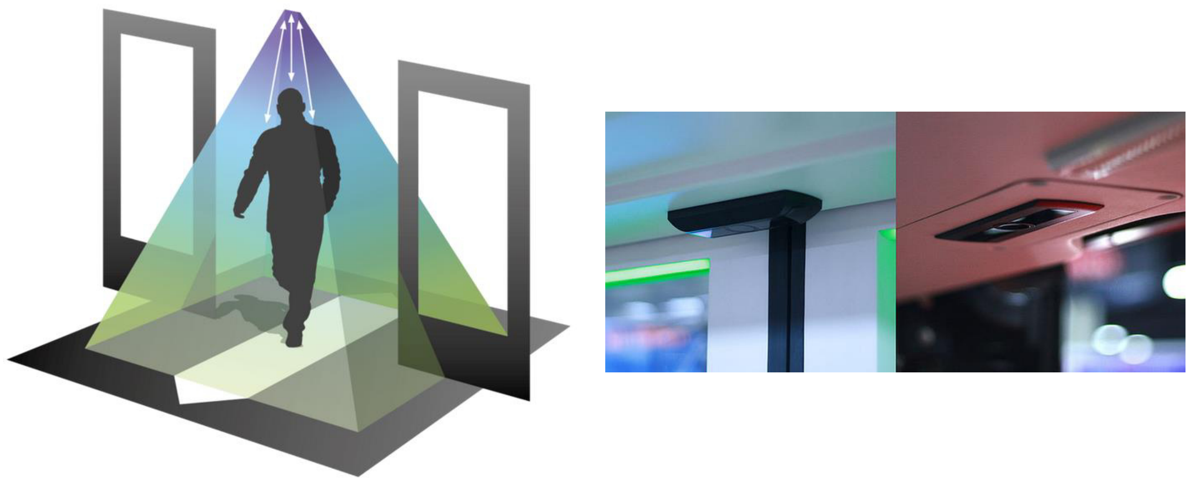


**Figure 8: Infrared light sensors (iris-GmbH infrared & intelligent sensors, 2019)**

The system is connected to a computer and thus the data is processed in real-time 24/7. With this real-time processing the transport operator has always access to the actual passenger numbers in the vehicles and is very flexible in responding to changes in travel demand. With the help of long-term countings it is possible for the transport operator to optimize the capacities of vehicles to the transport volume. Thus, it is for example known how many units a train needs to meet the travel demand. The information about the current occupancy rate of vehicles may also be of importance for the passengers. Number of passengers currently existing in the vehicles can be forwarded to waiting passengers at the next stop. By means of a system mimicking traffic lights the waiting passengers are informed about how much space is available and thus the passengers can arrange accordingly, and it results in a shorter transfer time. Figure 9 shows the directed passengers to train units, more passengers are directed to more empty train units (iris-GmbH infrared & intelligent sensors, 2019).

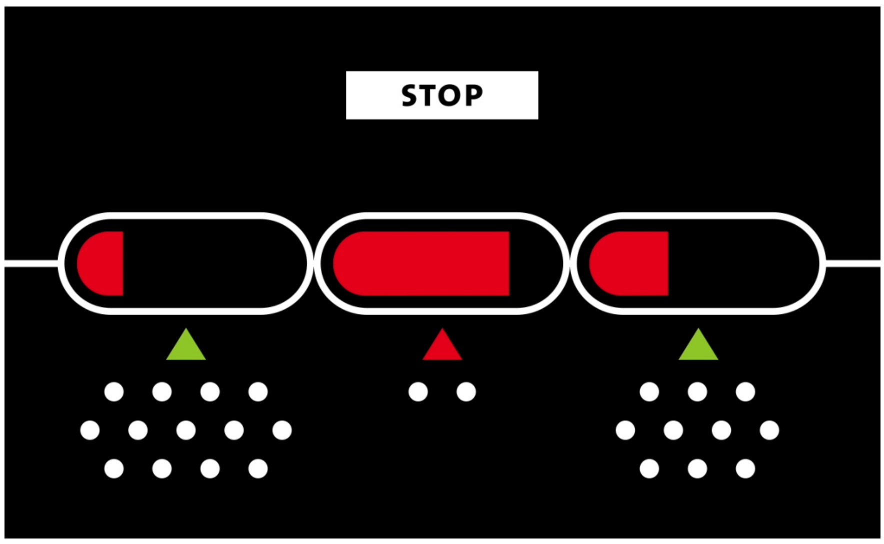


**Figure 9: Direct passengers to train units at a stop (iris-GmbH infrared & intelligent sensors, 2019)**

An addition to mobility apps would be the inclusion of the current numbers of passengers in the app. The user could look at the occupancy rates of the means of transport in real-time directly on the map or after calculating a route. Thereby it would be possible to recognize an already utilized vehicle and to choose an alternative. An appropriate example can be found in Graz. A large part of the operation area of trams 7 and 1 lays on the east side of Graz and they have also a not insignificant part on the same track. Line 7 is usually busier than line 1 and if the line 7 arrives now just before line 1 at the stop, most people get into the already utilized tram. The shortly following line 1 is by far not as busy and only takes on the leftover passengers. If the information of the occupancy rate could be seen at the station or in the app, the passengers would be able to arrange themselves better.

If no disruptions occur in a public transport network and the operation can proceed as planned, then there are usually no problems with the displayed information on mobility apps. Possible delays are taken into account by the use of real-time data when calculating a route and the user gets the information he needs. But if there are unexpected events or disruptions, such as an accident on the tram rail, traffic jam or the barrier of a stop, then there are usually no proposed solutions, but a message about the disruption on the respective part of the network is shown. An optimal solution for the user would be the calculation of an alternative route, in which the disrupted or locked area can be avoided. In Milano, such a system was developed and tested with the MOTUS (mobility and tourism in urban areas) transport planner. The basic idea was to develop an information system that flexibly manages unexpected events and disruptions in public transport in real-time and to pass on this information to the users. This allows users to organize their trips themselves and to choose the best solution. The data used in the system was provided by the public transport manager. Figure 10 shows the route calculation using MOTUS under normal conditions and with disruptions. It can be seen, that at the calculation with disruptions the locked stops are bypassed, and an alternative route is calculated. The routes calculated under normal conditions are similar to those that are provided by other journey planners (Bruglieri, et al., 2015).

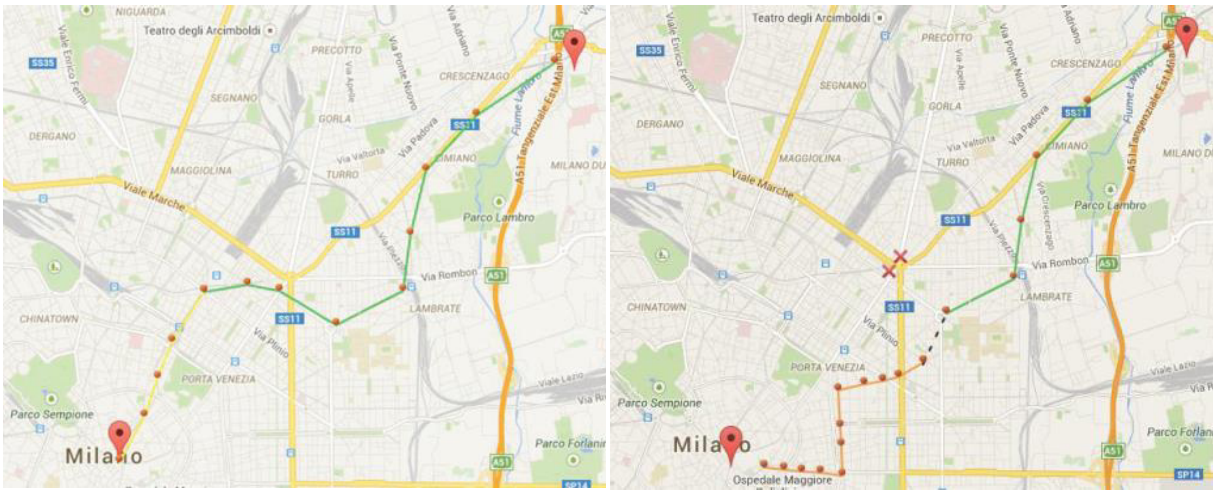


**Figure 10: MOTUS route calculation in normal conditions (left) and MOTUS route calculation with disruptions (right) (Bruglieri, et al., 2015)**

Using this system, public transport users could be deliberately directed. If there are too many passengers on a line or in an area, the users can be offered different routes and thus distributing the volume.

Another way to use traffic data comes from the private transport sector. This involves the use of floating car data (FCD) to determine route times and furthermore the occupancy rate of line sections. These traffic data exist of traces of GPS positions from vehicles and additionally information, such as travel times and vehicle speeds, is send to a data processing system. Thereby, these vehicles operate as moving sensors, which participate directly in traffic and thus fixed installed traffic sensors, like loop detectors or cameras, can be supplemented by this procedure. The more floating car data is present, the more reliable are the results which arise from that. With FCD recorded in the past and currently obtained FCD, it is possible to make predictions to estimate the future road traffic condition (Fabritiis, Ragona, & Valenti, 2008). FCD can be sent by any mean of transport which involves GPS, thus also from taxis, busses and shared mobility vehicles, and be used for traffic flow analysis by transport planners. Occupany rates and average travel times can be displayed on maps and influence navigation and route planning.

To get a good overview over the mentioned possible usage of data out of mobility apps and traffic management systems, Table 3 summarizes short and concise the interesting data for transport planners and users.

**Table 3: Interesting data for transport planners and users out of mobility apps and traffic management systems summarized in a table**

| | Transport planners … | Users … |
|---|---|---|
| **Public transport assignments out of mobility app data** | … can calculate the volumes for lines out of route requests with the help of public transport assignments in a transport model. | - |
| **Line volume out of GPS routes and payment system** | … can use GPS routes or tickets from payment systems to figure out the used lines of the users. | - |
| **Expansion of shared mobility services** | … can estimate the attractiveness of an area by the accumulation of requests and expand the amount of shared mobility services if necessary. | - |
| **Smart parking** | … can reduce the traffic caused by searching for an available parking space with a good smart parking system. | … can find an available parking space without annoying searching. This saves nerves, time and money. |
| **Passenger counting system** | … have always access to the actual passenger numbers in the vehicles and are very flexible in responding to changes in travel demand. The capacities of vehicles can be optimized to the transport volume. | … can check the occupancy rate at a stop or on the mobile phone and arrange themselves or choose a not so busy alternative. |
| **Disruption or unexpected event managing** | … can deliberately direct passengers to distribute the volume of an overloaded line or area. | … can avoid disruptions or locked areas by automatically calculating an alternative route for them. |
| **Floating Car Data** | … can use this data for traffic flow analysis to get average travel times and occupancy rates of line sections. | … can view occupancy rates on a map and take advantage in terms of route planning and navigation. |

# 3 Analysis of mobility data

Cities show a steady grow in population and must deal with increased traffic amount. Therefore, transport planners are searching for new mobility solutions to counteract occurring traffic problems. With mobile devices and the fast development of the internet new possibilities for mobility services are provided and mobility apps may provide data which can be interesting for transport planning purposes.

Figure 11 shows the methodology of the analysis with a flow chart. The mobility data to be analyzed is provided by the mobility app “qando Graz”. Information about the app can be found in chapter 2.2.4 and 3.1. The further extent of the master thesis focuses on the individual work steps, which are given in the flow chart, and explains them in detail.

The analysis deals with raw data that has been read out directly from the app. This raw data include data from monitor and route requests and thus these two sets of data build the two major parts of the analysis. Subsequently, the raw data are prepared and sorted for further processing in order to facilitate the data processing for the evaluations. Afterwards, the different types of evaluations are presented. A distinction between distribution and frequency plots is made in the monitor data. These figures show the temporal distributions over a certain time period, six months or one day, and the frequencies and locations of the requests. These representations are also available for the route requests and additionally three more. Spider matrices take into account the relationship between origin and destination, in which the route requests are related to defined areas, so-called traffic zones. The frequencies of routes between the same traffic zones are included in the figures. In order to be able to compare the qando route data with real occurring numbers of passengers, public transport assignments are carried out. From these assignments the line transportation on respective lines is obtained and can thus be compared with the real numbers. To be able to statistically prove the comparison between the assigned route data and the real occurring numbers of passengers, a $\chi^2$-test is carried out as the last evaluation.

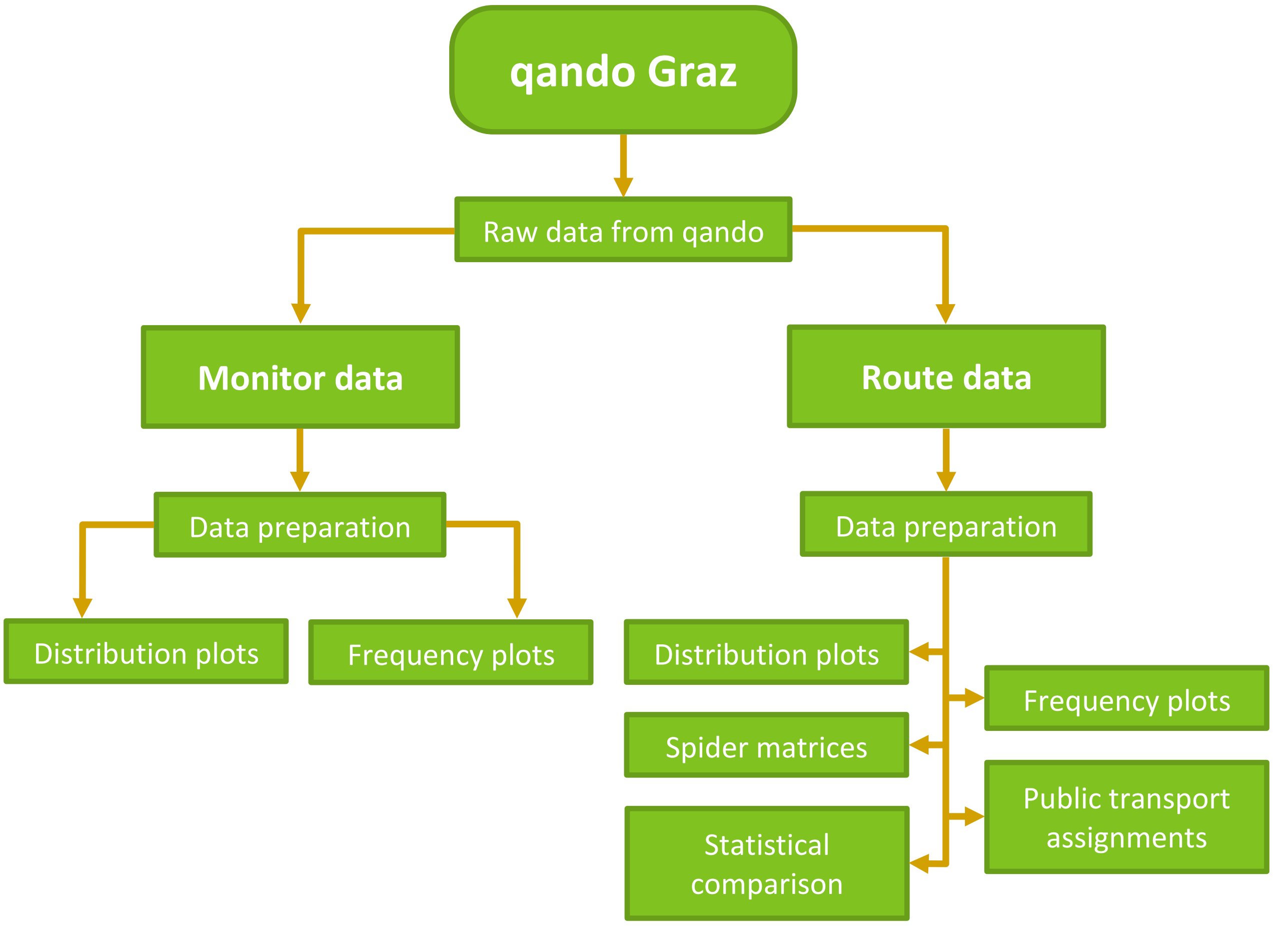


**Figure 11: Methodology of the analysis**

## 3.1 Mobility app qando Graz

The app has already been presented in chapter 2.2.4. As a short reminder the important information about the app is shown in the lines below.

qando Graz is a free first-generation mobility app. That means, that the HGL offer a journey planner with real-time public transport information for the zone 101. Information about the departure times of means of transport can be called up to an accuracy of one minute. Also, the route planning can be done for the whole state Styria. Additionally, useful information about disruptions in the service area or POIs are shown in the app (Holding Graz - Kommunale Dienstleistungen GmbH, 2019). The following Figure 12 shows the fare zone 101 of Graz.

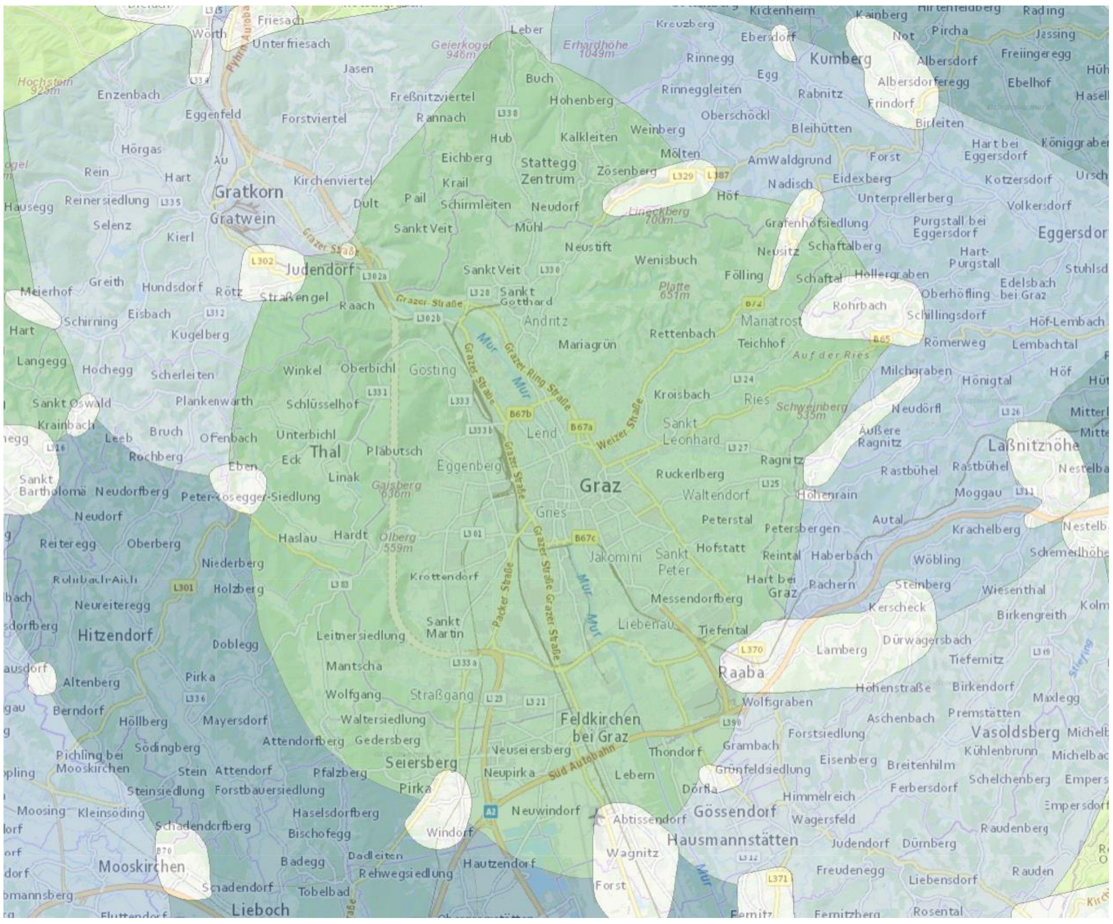


**Figure 12: Fare zone 101, Source: https://gis.stmk.gv.at/atlas/(S(oapfmedca4ghse5cxuec1yh5))/init.aspx?cms=da&karte=emptymap&layout=gisstmk&styles=gisstmk&template=gisstmk&gdiservices=hintergr%2cgel%2cdopags_tc%2copbmgrau%2copbm%2cuctc%2copoverlay&gdiservices=kat%2corient_adr&sichtbar=_ortsplanGrau&t=637067464260001353 [15.10.2019]**

### 3.1.1 Functions of the app

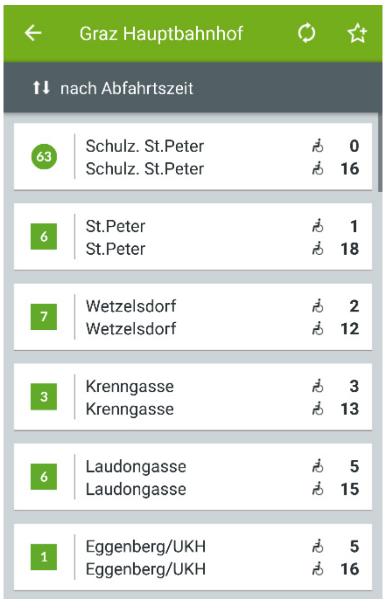
Several functions are offered by the app which should support passengers collecting information about public transport in Graz. These functions are shown in the following list (Holding Graz - Kommunale Dienstleistungen GmbH, 2019).

- **Favorites:** Saved routes and stops by the user
- **Monitor:** Real-time information about departure times at stops in the zone 101
- **Route:** Route planning within Styria
- **Map:** Shows entered route, positions of stops and POI
- **Disruptions:** Displays possible disruptions in the service area
- **Advices & news:** General information (diversions, transferred stops, etc.)
- **Lines & schedules:** Shows all the lines and current net plans of the zone 101 (Holding Graz - Kommunale Dienstleistungen GmbH, 2019)

In the further only monitor and route function are described in detail, because the analysis is limited on data out of these.

### 3.1.2 Monitor function

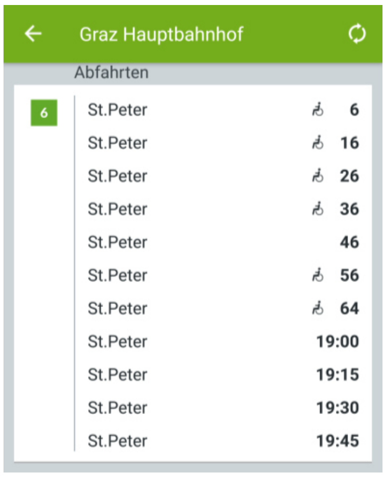
As mentioned before, the monitor function returns departure times in real-time for desired stops. For the request either a specific stop or the actual position of the user can be used. By requesting a specific stop the next departing lines are shown one below the other in a list. Each with the next two departure times. More departure times are displayed by selecting one line. In Figure 13 the graphical user interface (GUI) of the monitor function in qando can be seen. The left picture includes lines with departure times for the stop "Graz main railway station" and the right one departure times of line 6 for the stop "Graz main railway station". qando was created for the use in Austria and so there is no option to change the German language in the app.



Graz Hauptbahnhof

nach Abfahrtszeit

| | | | |
|---|---|---|---|
| 63 | Schulz. St.Peter | ♿ | 0 |
| | Schulz. St.Peter | ♿ | 16 |
| 6 | St.Peter | ♿ | 1 |
| | St.Peter | ♿ | 18 |
| 7 | Wetzelsdorf | ♿ | 2 |
| | Wetzelsdorf | ♿ | 12 |
| 3 | Krenngasse | ♿ | 3 |
| | Krenngasse | ♿ | 13 |
| 6 | Laudongasse | ♿ | 5 |
| | Laudongasse | ♿ | 15 |
| 1 | Eggenberg/UKH | ♿ | 5 |
| | Eggenberg/UKH | ♿ | 16 |



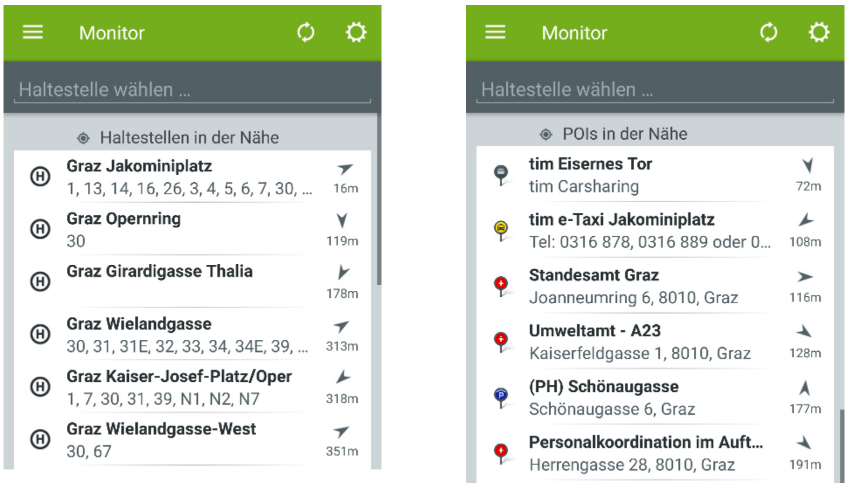
Graz Hauptbahnhof

Abfahrten

| | | | |
|---|---|---|---|
| 6 | St.Peter | ♿ | 6 |
| | St.Peter | ♿ | 16 |
| | St.Peter | ♿ | 26 |
| | St.Peter | ♿ | 36 |
| | St.Peter | | 46 |
| | St.Peter | ♿ | 56 |
| | St.Peter | ♿ | 64 |
| | St.Peter | | 19:00 |
| | St.Peter | | 19:15 |
| | St.Peter | | 19:30 |
| | St.Peter | | 19:45 |

**Figure 13: Lines with departure times for stop "Graz main railway station" (left) and departure times of line 6 for stop "Graz main railway station" (right)**

Using the actual position while opening the monitor function reflects all stops and POI nearby. Also, the directions and the distances to the stops are included. By clicking on a stop, the user is passed on to the previously shown GUI of the stop. If one POI is selected, the map opens and zooms to the chosen one. Figure 14 displays the GUI of stops nearby on the left side and POI nearby on the right side.


Monitor

Haltestelle wählen ...

Haltestellen in der Nähe

| | |
|---|---|
| **Graz Jakominiplatz** 1, 13, 14, 16, 26, 3, 4, 5, 6, 7, 30, ... | 16m |
| **Graz Opernring** 30 | 119m |
| **Graz Girardigasse Thalia** | 178m |
| **Graz Wielandgasse** 30, 31, 31E, 32, 33, 34, 34E, 39, ... | 313m |
| **Graz Kaiser-Josef-Platz/Oper** 1, 7, 30, 31, 39, N1, N2, N7 | 318m |
| **Graz Wielandgasse-West** 30, 67 | 351m |

Monitor

Haltestelle wählen ...

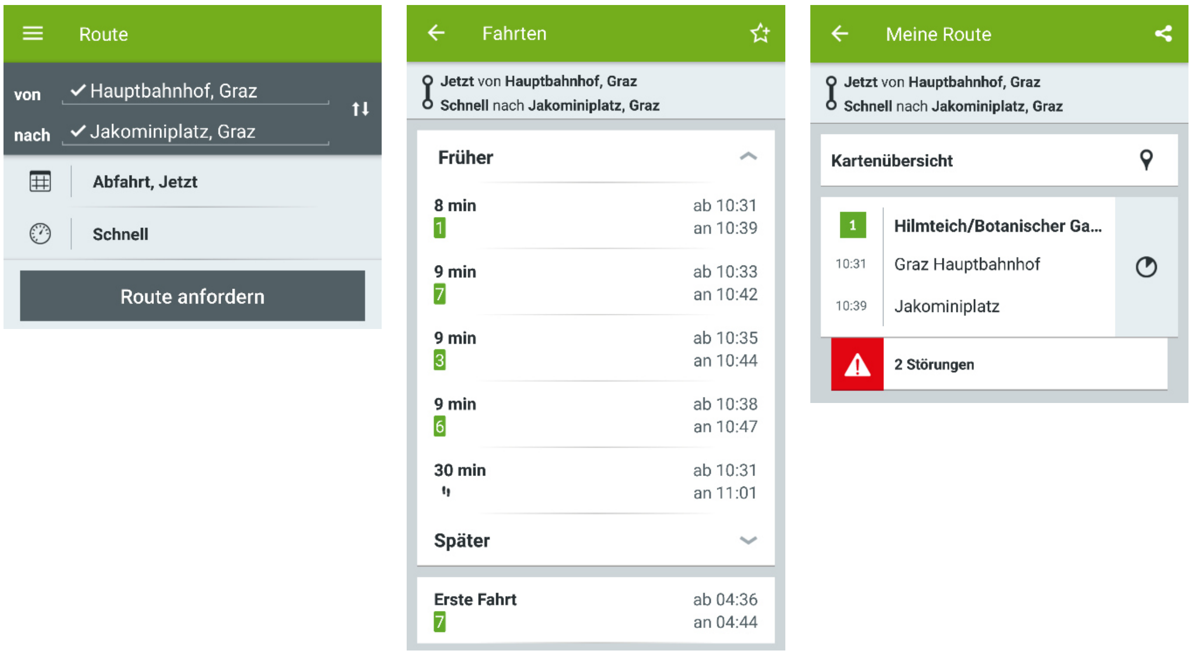
POIs in der Nähe

| | |
|---|---|
| **tim Eisernes Tor** tim Carsharing | 72m |
| **tim e-Taxi Jakominiplatz** Tel: 0316 878, 0316 889 oder 0... | 108m |
| **Standesamt Graz** Joanneumring 6, 8010, Graz | 116m |
| **Umweltamt - A23** Kaiserfeldgasse 1, 8010, Graz | 128m |
| **(PH) Schönaugasse** Schönaugasse 6, Graz | 177m |
| **Personalkoordination im Auft...** Herrengasse 28, 8010, Graz | 191m |

**Figure 14: Stops nearby (left) and POI nearby (right)**

## 3.1.3 Route function

To calculate the connection between a starting and an ending point the app integrates the route function. As already mentioned, routes can be determined for the city Graz and the whole state Styria (Holding Graz - Kommunale Dienstleistungen GmbH, 2019). Starting and ending point can be stops, addresses or the actual position of the user. Departure times and times of arrival can be chosen as “now” or as the desired date and time. After entering the start and end points, possible connections are displayed. The route is then shown with all means of transport to use and if a footpath is necessary it is also considered in the routing. To get an overview about the chosen route the connection can be viewed in the map. Figure 15 reflects the connection between „Graz main railway station“ and „Graz Jakominiplatz“.


Route

von ✓ Hauptbahnhof, Graz

nach ✓ Jakominiplatz, Graz

Abfahrt, Jetzt

Schnell

Route anfordern

Fahrten

**Jetzt** von **Hauptbahnhof, Graz**
**Schnell** nach **Jakominiplatz, Graz**

**Früher**

| | |
|---|---|
| **8 min** 1 | ab 10:31 an 10:39 |
| **9 min** 7 | ab 10:33 an 10:42 |
| **9 min** 3 | ab 10:35 an 10:44 |
| **9 min** 6 | ab 10:38 an 10:47 |
| **30 min** | ab 10:31 an 11:01 |

**Später**

| | |
|---|---|
| **Erste Fahrt** 7 | ab 04:36 an 04:44 |

Meine Route

**Jetzt** von **Hauptbahnhof, Graz**
**Schnell** nach **Jakominiplatz, Graz**

**Kartenübersicht**

1 **Hilmteich/Botanischer Ga...**
10:31 Graz Hauptbahnhof
10:39 Jakominiplatz

**2 Störungen**

**Figure 15: Route request from “Graz main railway station” to “Graz Jakominiplatz”**

## 3.2 Raw data from qando Graz

The qando Graz data was provided by the company Fluidtime Data Services GmbH. Fluidtime is a leading provider of IT-services in the fields of Mobility-as-a-Service (MaaS) and integrated mobility in Austria and since 2016 a subsidiary of Kapsch-Group (Fluidtime, 2019a). The company developed the components for server and client of qando themselves and menu navigation, visual design and range of functions in cooperation with Wiener Linien. All Austrian qando services were operated by Fluidtime since 2015 and so HGL is a partner/customer of Fluidtime (Fluidtime, 2019b).

At the beginning two days of qando data were provided by Fluidtime. These were the 08.01.2019 and 09.01.2019. Because this is a Tuesday and a Wednesday the dates are representative. No Monday or Friday and no weekend. The data includes monitor and route requests. With these two days first analyses were carried out and results were presented. In order to obtain differences between weekdays, weekends and seasons, data, which extends over a longer period of time, is needed. For that six months of qando data were read out by Fluidtime. The time period extends from 01.10.2018 to 31.03.2019. It contains with autumn, winter and spring three seasons. These data also contain monitor and route data and are each stored in a csv-file. During the six months 12.935.536 monitor requests have been stored, which corresponds to more than 3 GB of data. The route requests amount to 766.354, about 300 MB of data. Monitor requests are divided into:

- **Stop requests (2.716.766):** Requests where the user enters the stop name
- **Position requests (1.557.127):** Requests where the actual position of the user is used via GPS
- **Favorite requests (3.292.672):** Requests which are saved as favorite by the user
- **Departure board requests (5.368.971):** Requests from public placed departure boards at stops

### 3.2.1 Data structure

The data is separated with semicolon ( ; ) in the csv-files. This csv-files can be imported and displayed in spreadsheet programs. It can be seen, that the information is divided into rows and columns, where the division of the data is sometimes not necessarily advantageous, because much information is stored in one cell. Each line corresponds to a separate request. In the following two tables examples for the structure of the monitor data and of the route data are given.

**Table 4: Structure monitor data**

| | User | Date Time | Time Zone | Method |
|---|---|---|---|---|
| 1 | c7c4cbe3939e3450133063e1315aa863 | 08.01.2019 07:53:38 | GMT+0100 | GET |
| 2 | e6abca9efbd7eab3b8e6c2aff3cb961b | 08.01.2019 07:53:40 | GMT+0100 | GET |
| 3 | eb615f8d5e6bda537c75b3e6a2e71386 | 08.01.2019 07:54:16 | GMT+0100 | GET |
| 4 | 485190e0d1f2e4a6943a77252254ec5d | 08.01.2019 07:54:17 | GMT+0100 | GET |

| | URL | Response Code | Bytes Sent | User Agent |
|---|---|---|---|---|
| 1 | /monitor?&rblLineDirs=407911:5:R&rblLineDirs=4079216:34:H&rblLineDirs=4079216:34E:H | 200 | 1230 | qando-graz_/1.3.2+77 (Android; 7.0; Scale/2.0; Size/1536/2048;) |
| 2 | /monitor?diva=63203268 | 200 | 1485 | qando-graz_/1.3.2+77 (Android; 8.0.0; Scale/2.625; Size/1080/1920;) |
| 3 | /monitorNearby?coord= 15.394145%3A47.045532%3A WGS84&limit=10&radius=1000 | 200 | 4899 | widget/170 (iPhone; iOS 12.1.2; Scale/3.00) |
| 4 | /monitor?rblLineDirs=299248:7:R,2992116: 33:R,2992116:33E:R,29923172:62:H&_= 1546912813715 | 200 | 4364 | mpo_departureboard/1.0 (Monitor; MonitorOS) |

The data is split into the eight displayed columns. Each column has different amounts of content in the cells. Some cells include a lot of information, like URL or User Agent, and some others include very little information, like Method or Response Code. Each qando user is assigned an ID, which is stored with every request. The time of the request is included in Date Time and Timezone. The timezone is relevant when the time is changed from GMT+0100 (winter time) to GMT+0200 (summer time) and reverse. Information about which stops, stop points with lines and stops near the current position were requested is written in URL. Response Code indicates whether the request was successful or not (200 = successful) and Bytes Sent shows the amount of transferred data that was required. User Agent gives information about the device with which the request was made.

Requests from different devices are also stored partly different in the data. For example: Android and departure board requests use colon ( : ) as separator, in contrast iPhone devices specify the colon with %3A. Or Android and iPhone requests also use & as a separator, whereby departure board requests work with commas ( , ). Departure board requests come from the departure boards which are mounted in public at the stops.

**Table 5: Structure route data**

| | User | Date Time | Time Zone | Method |
|---|---|---|---|---|
| 1 | c0c2188eda4a699e50562ac5238ab1dc | 08.01.2019 08:41:51 | GMT+0100 | GET |
| 2 | e136c349c1f42fff2ad18ff828361ec0 | 08.01.2019 08:42:03 | GMT+0100 | GET |
| 3 | 13790acd31614265279758ca319b03ef | 08.01.2019 08:43 | GMT+0100 | GET |

| | URL | Response Code | Bytes Sent | User Agent |
|---|---|---|---|---|
| 1 | /route?from=63207334&to=63204213&date=2019-01-08T 19%3A41%3A40.685%2B0100&aSS=1&aPP=1&aflT=1&ptRO= ptMinTime&deparr=dep&ptMWT=5&walkMT=5&ptWS= ptNormal&ptV=ptTrain&ptV=ptTrainS&ptV=ptTrainR&ptV= ptBusCity&ptV=ptBusRegion&ptV=ptTram&version=1.1 | 200 | 13601 | qando-graz_/1.3.2+77 (Android; 8.0.0; Scale/3.0; Size/1080/1920;) |
| 2 | /route?from=15.43192:47.08979:WGS84:Grabenstra%C3%9Fe +115:a_618839365&overrideFromType=poi&to=63204076& date=2019-01-08T08%3A42%3A03.152%2B0100&aSS=1&aPP= 1&aflT=1&ptRO=ptMinTime&deparr=dep&ptMWT=15& walkMT=15&ptWS=ptNormal&ptV=ptTrain&ptV=ptTrainS& ptV=ptTrainR&ptV=ptBusCity&ptV=ptBusRegion&ptV=ptTram& version=1.1 | 200 | 7096 | qando-graz_/1.3.2+77 (Android; 8.0.0; Scale/3.0; Size/1080/1920;) |
| 3 | /route?version=1.1&from=15.458150%3A47.069543%3AWGS8 4%3ASchillerstra%C3%9Fe%2049%2C%208010%20Graz%3ACU RRENTLOCATION&to=63204227&deparr=dep&modality=pt&aP P=1&aSS=1&ptRO=ptMinTime&ptMWT=15&ptWS=ptFast&aflT =1&walkMT=15&ptV=ptTram&ptV=ptBusCity&ptV=ptBusNight | 200 | 9558 | qando_hgl_app/170 (iPhone; iOS 12.1.2; Scale/2.00) |

The column-wise division of the route data is exactly the same as in the monitor data. Differences can only be seen in the column URL, here is now other information stored which relates to the routes. It is always indicated from where to where the route was requested. For the requests stops and the actual position can be used. In addition, a second time indication is included in the requests. This does not indicate the time at which the request was made, but that to which the user wants to depart or arrive at the destination. The remaining factors are settings that the user can pre-define, such as maximum walking time, maximum number of transfers or which means of transport should be used for calculating the route. Also, in the route requests different devices have a partially different syntax.

### 3.2.2 Data preparation

With this division of the data reading out the desired information is very difficult and not very practical for further work. Therefore, the data is prepared and sorted, a separate column is created for each variable. If the corresponding variable does not exist for a request, a NA is entered in the cell. The final result is a huge table, which is very clear and facilitates the reading out of the desired information. Table 6 shows a small extract of the sorted monitor data, which was created with the same requests as above.

**Table 6: Structure sorted monitor data**

| | User | Date | Time | Platform | Request | Coord. Sys. |
|---|---|---|---|---|---|---|
| 1 | c7c4cbe3939e3450133063e1315aa863 | 08.01.2019 | 07:53:38 | Android | /monitor | NA |
| 2 | e6abca9efbd7eab3b8e6c2aff3cb961b | 08.01.2019 | 07:53:40 | Android | /monitor | NA |
| 3 | eb615f8d5e6bda537c75b3e6a2e71386 | 08.01.2019 | 07:54:16 | iPhone | /monitorNearby | WGS84 |
| 4 | 485190e0d1f2e4a6943a77252254ec5d | 08.01.2019 | 07:54:17 | Monitor | /monitor | NA |

| | Limit | Radius | Coordinate X | Coordinate Y | Diva (Stop) | rblLineDirs 1 | rblLineDirs 2 | … |
|---|---|---|---|---|---|---|---|---|
| 1 | NA | NA | NA | NA | NA | 407911:5:R | 4079216:34:H | … |
| 2 | NA | NA | NA | NA | 63203268 | NA | NA | … |
| 3 | 10 | 1000 | 15.394145 | 47.045532 | NA | NA | NA | … |
| 4 | NA | NA | NA | NA | NA | 299248:7:R | 2992116:33:R | … |

Following the same system as the monitor data to prepare and sort the data also the route data is processed this way. Of course, the column names differ and the division is adjusted to the route requests. Table 7 shows the sorted route data.

**Table 7: Structure sorted route data**

| | User | Date | Time | Platform | Request | From Diva |
|---|---|---|---|---|---|---|
| 1 | c0c2188eda4a699e50562ac5238ab1dc | 08.01.2019 | 08:41:51 | Android | /route | 63207334 |
| 2 | e136c349c1f42fff2ad18ff828361ec0 | 08.01.2019 | 08:42:03 | Android | /route | NA |
| 3 | 13790acd31614265279758ca319b03ef | 08.01.2019 | 08:43:04 | iPhone | /route | NA |

| | From X | From Y | Address From | To Diva | To X | To Y | Address To | … |
|---|---|---|---|---|---|---|---|---|
| 1 | NA | NA | NA | 63204213 | NA | NA | NA | … |
| 2 | 15.43192 | 47.08979 | Grabenstraße 115 | 63204076 | NA | NA | NA | … |
| 3 | 15.458150 | 47.069543 | Schillerstraße 49, 8010 Graz | 63204227 | NA | NA | NA | … |

The sorting and the preparation of the data was made in open source programm RStudio 1.1.463 with the programming language R. Standard programs, like spreadsheet program Microsoft Excel or the editor, are not able to manage the large data set of the monitor data, but with RStudio it is possible without problems to read in the big data set and to further process it. The use of loops is largely avoided for long runs, as they need an almost endlessly long processing time. Therefore, the vectorization is used, so that the processing steps can be accelerated enormously.

To reduce the runtime for big data sets, like six months of monitor data, not every variable is stored into the sorted table, only the data for further processing is considered and sorted.

# 4 Results of app data analysis

This chapter explains the evaluation results of the qando data (monitor and route data). It considers temporal distributions, frequencies and location of the requests, as well as public transport assignments of qando data by using a Visum model. To compensate possible outliers, medians instead of arithmetic means are used for the evaluations. The median always takes the value in the middle of a vector sorted in an ascending order.

For the implementation of the evaluations different time periods were considered. For the consideration of all days, which are contained in the volume of data, a long-term consideration over six months was used. It contains 182 days from 01.10.2018 – 31.03.2019. Due to the formation of the median over this period, this is called "Median 01.10.2018 – 31.03.2019". In order to involve various seasons of the year in the evaluations, the six months were divided into the three periods autumn, winter and spring. Autumn covers 01.10.2018 – 18.11.2018, winter 19.11.2018 – 10.02.2019 and spring 11.02.2019 – 31.03.2019. Additionally, a division based on the days of the week was carried out. The data was divided into the seven days of the week and only the respective day of the week is considered in the six months. For example, this means for the Mondays, that in the evaluations only Mondays are considered. The consideration period includes 26 weeks and thus the median was formed over 26 Mondays. For the other days of the week this is the same. In the following Table 8 the consideration periods for the evaluations can be seen. Median defines that the median was built over the period.

**Table 8: Consideration periods for the evaluations**

| |
|---|
| – **Median 01.10.2018 – 31.03.2019** |
| – **Median autumn 01.10.2018 – 18.11.2018**<br>– **Median winter 19.11.2018 – 10.02.2019**<br>– **Median spring 11.02.2019 – 31.03.2019** |
| – **Median Mondays 01.10.2018 – 31.03.2019**<br>– **Median Tuesdays 01.10.2018 – 31.03.2019**<br>– **Median Wednesdays 01.10.2018 – 31.03.2019**<br>– **Median Thursdays 01.10.2018 – 31.03.2019**<br>– **Median Fridays 01.10.2018 – 31.03.2019**<br>– **Median Saturdays 01.10.2018 – 31.03.2019**<br>– **Median Sundays 01.10.2018 – 31.03.2019** |

All graphic representations in this thesis are created with RStudio.

## 4.1 Results monitor data

The evaluations of the monitor data are based on the time of request, the used platform from which the request was sent and the current position or the requested stop.

### 4.1.1 Platforms used for monitor requests

The monitor requests were carried out from various platforms and their distribution is shown in Figure 16. Departure board requests make up the largest share. Thus, with more than five million requests in six months, most of the requests come from the own system of the Holding Graz. Second most frequent requested type are mobile devices requesting a stop. This kind of requesting stops is the most common one. Most users of mobile devices made the requests with an iPhone, then come the Android devices and few requests were also sent via iPads. A rather outdated method is to request with the web application. The proportion of these is just over 1 % compared to mobile applications. Others includes the remaining requests from platforms that are not often used. In total 12.935.536 requests were made in six months.

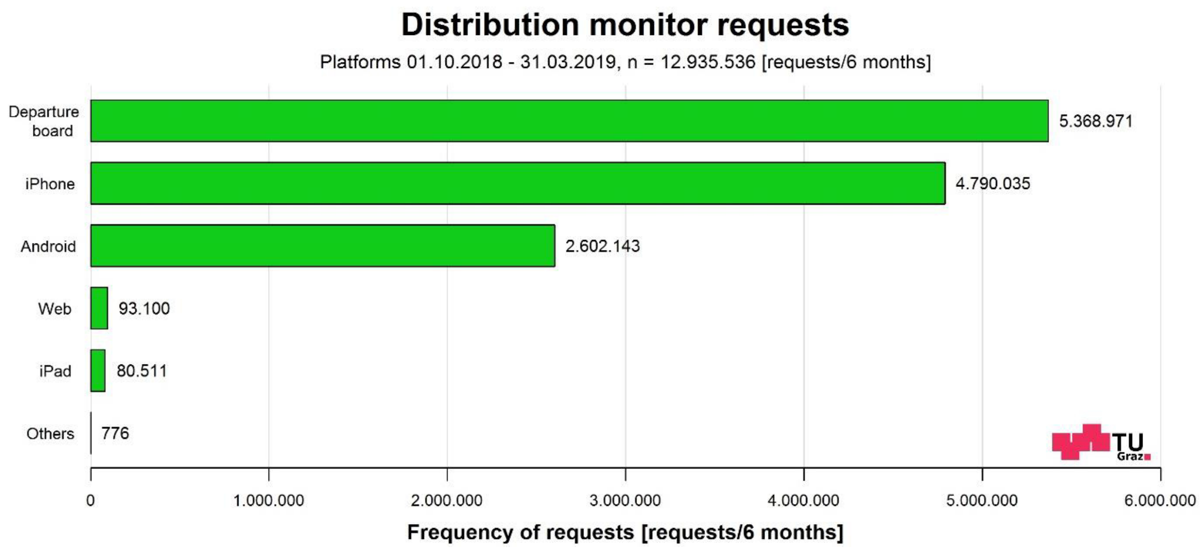


**Figure 16: Platforms used for monitor requests**

### 4.1.2 Distribution plots

The distribution plots display the temporal distribution of the requests, either over the entire six months or over one day. Based on the sorted monitor data four distribution tables are created. One for position, stop, favorite and departure board requests. Table 9 shows the distribution table of the stop requests in an exemplary presentation. All 182 days between 01.10.2018 – 31.03.2019 are included in the columns. At the right end of the table the sum and the eleven medians are attached. Each line stands for one hour of the day and the last one for the sum over a whole day. The frequencies are entered in the cells. For example, on 01.10.2018 27 stop requests were carried out in the first hour, from 00:00 to 01:00, and in total 12.788 stop requests were made over six months in the first hour. Over the whole day 18.357 stop requests were sent on 01.10.2018.

**Table 9: Distribution table stop requests, exemplary presentation**

| Hours | 01.10.2018 | 02.10.2018 | 03.10.2018 | … | Sum | Median | Median autumn | … |
|---|---|---|---|---|---|---|---|---|
| … | … | … | … | … | … | … | … | … |
| 7 | 1.010 | 1.128 | 961 | … | 129.896 | 872 | 858 | … |
| 8 | 1.908 | 2.174 | 1.982 | … | 233.480 | 1.641,5 | 1.551 | … |
| 9 | 1.325 | 1.432 | 998 | … | 157.272 | 973,5 | 928 | … |
| … | … | … | … | … | … | … | … | … |
| Sum | 18.357 | 17.982 | 16.060 | … | 2.716.766 | 15.676,5 | 15.199 | … |

The distribution tables of the other request types are built up the same way. How the distribution tables are created is visible in the following Pseudo-Code.

**Pseudo-Code:** Distribution tables for position, stop, favorite and departure board requests

```
##-----------------------------------------------------------------------

Read in sorted monitor table
Reduce time to hours (delete minutes and seconds)
Divide monitor table to 4 matrices --> position, stop, favorite and departure
   board matrix
Overwrite matrices with calculated frequencies for requests per hour and date
Write hours 1 to 24 in row names
Change format of date to "dd.mm.YYYY" and write it to column names
Calculate all medians for every hour and add them at the end of the table
Change the column names for the medians part
Build the sum for every column
Write all 4 tables to .csv

##-----------------------------------------------------------------------
```

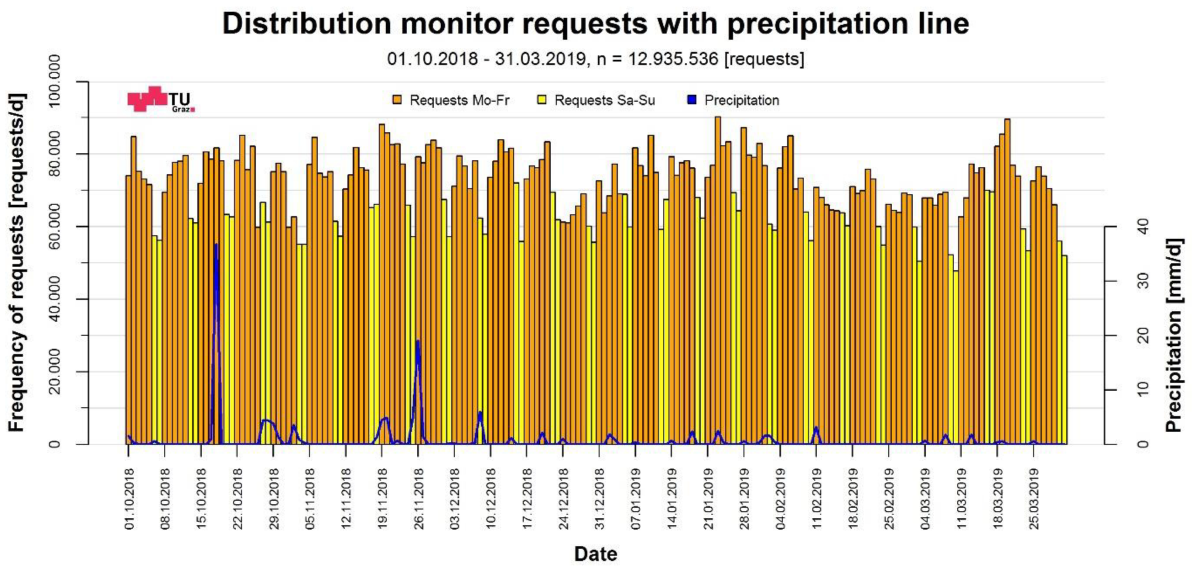


**Figure 17: Distribution monitor requests, every day of six months, with precipitation line**

In Figure 17 the distribution of the monitor requests over six months is shown. The orange bars are requests on weekdays, the yellow bars are requests on the weekends and the blue line is the precipitation. The x-axis contains the dates and every Monday is plotted. How much requests were performed per day is plotted on the y-axis. As mentioned before, a total of 12.935.536 requests were recorded between October 2018 and March 2019.

It is easy to see that there were more requests on weekdays than on weekends. In public transport, the number of passengers at the weekend is about one third of the number of passengers on weekdays. In comparison to the real passenger counts the proportion of requests for the qando data at the weekends is very high compared to during the week. This could be because passengers need more information about the schedule at the weekend. The lines do not run as often because they have higher headways and with a simple request to the monitor function in qando, the exact time of the next departure can be determined. Also, more non-daily trips are made at weekends which may result in multiple requests due to ignorance of the environment.

During the Christmas holidays a small collapse of more than 10.000 requests can be seen. As the holidays continue, the course increases steadily. A second collapse is noticeable in February and in the beginning of March during the semester break of the universities. This collapse is not as clear as in the Christmas holidays, but extends over a longer period of time.

The precipitation values were generated at a measuring station in Stremayrgasse, which is operated by the Technical University (TU) and are given in millimeters per day (mm/d). 15,451643/47,064669 (WGS84) are the coordinates of the station in central location. WGS84 means World Geodetic System 1984 and is the coordinate system in which the coordinates are located. The precipitation line is only used for visual comparison, whether occurred rain showers possibly had an influence on the number of requests. On days with higher precipitation, such as on 18.10.2018 and 26.11.2018, no increases in monitor requests are noticeable. When looking at the smaller amounts of rain also hardly any increases are visible. However, on certain days, like as on 19.11.2018, a higher number of requests can be seen, which may have arisen due to an occurred precipitation. But this is just a guess and is not proven.

By removing the requests from the public mounted departure boards and considering the requests sent by the users, it can be seen, that the departure board requests account more than 40 % of the total requests. The departure board requests show an approximate constant course, which can be seen in Figure 18 with the black line. Compared to the picture before (Figure 17) it looks very similar, but with a much smaller number of requests. The striking collapses during the holidays can also be recognized.

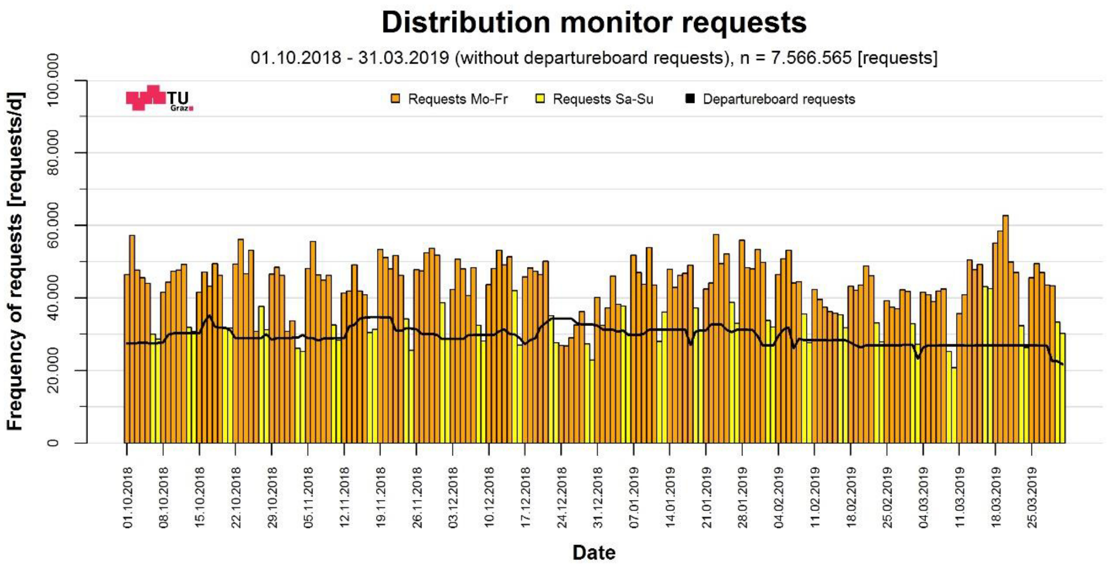


**Figure 18: Distribution monitor requests, every day of six months, without departure board requests**

In the previous figures, the distributions were considered over six months. The following illustrations show the daily courses. Figure 19 displays the sum of all requests over six months broken down into the four request types. On the x-axis the 24 hours of the day are plotted and on the y-axis the requests per hour. The departure board requests are very constant over the 24 hours of the day, they are only slightly lower during the night hours. This suggests that the public mounted departure boards request always very similar. Stop and favorite requests show beautiful courses. A striking morning peak and evening peak or evening sector is recognizable. The morning peak is a little bit higher than the evening peak. Such courses can be recognized at real public and private traffic and it is interesting that this picture also occurs with the qando data. Thus, it is easy to see that in the early morning many passengers request information about the departure times before leaving for work. In the evening the peak extends over a larger area, because the working hours are more variable than in the morning and leisure trips are added. The position requests have the least share of all four types and are rather constant during the day, but higher than the stop requests in the night hours.

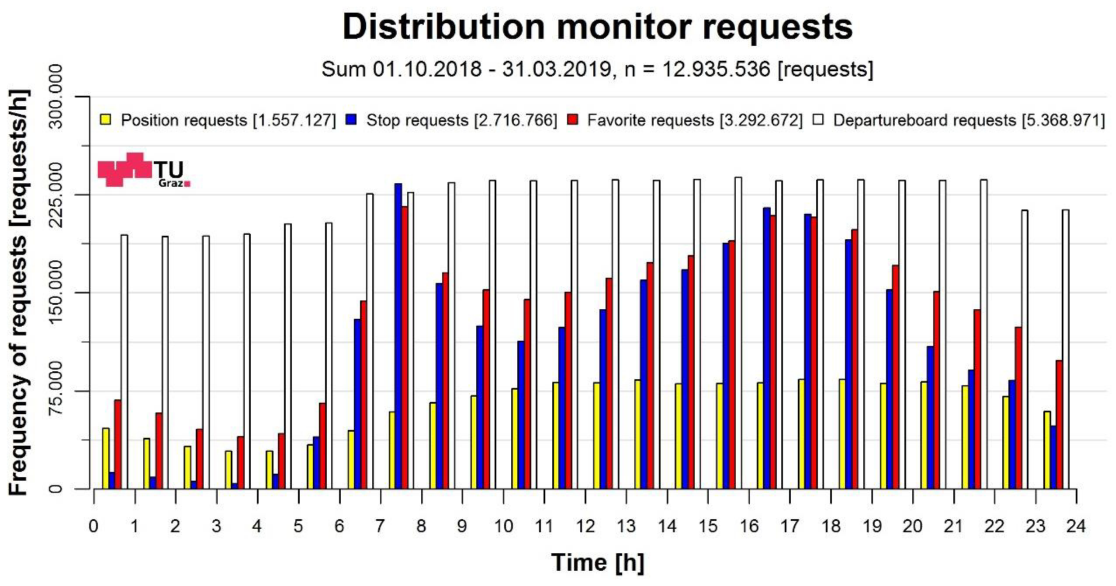


**Figure 19: Distribution monitor requests, daily courses, sum**

In general, the median over six months (Figure 20) has similar courses than the sum before, but based on requests per day. Departure board requests are again very constant and the user requests take again a corresponding representation. Between 07:00 and 08:00 o'clock the stop requests stand out with a very high value. On average 70.757,5 requests were made per day.

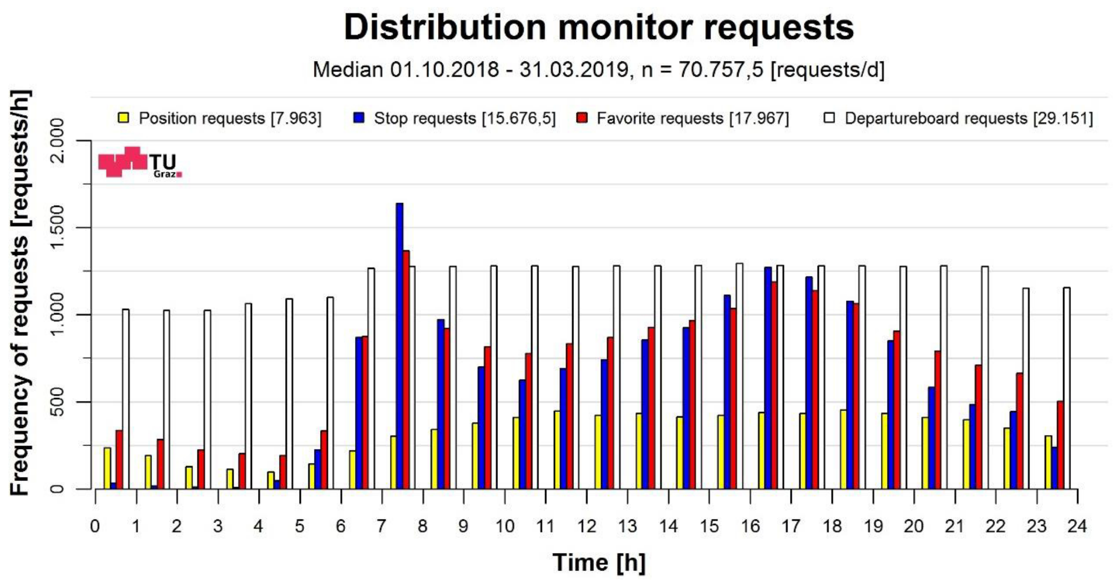


**Figure 20: Distribution monitor requests, daily courses, median six months**

Figure 21 shows the seasonal comparison of the monitor requests. Autumn in blue, winter in red and spring in white. All four types of requests are summed up to get one value for the season. In winter the most monitor requests were made with 73.830,5 requests per day, clearly visible in the course of the distribution. After winter comes autumn with 70.641 requests per day and is therefore almost the same as the average course over six months. Spring has the fewest number of requests with 67.257 requests per day. From this comparison can be deduced that during the cold season more people use public transport and thus more monitor requests arise. Because the temperatures are much higher in spring and the weather is more inviting for riding the bike or walking, it is not surprising that this season had received the fewest requests. The highest value of requests per day in winter was 4.779.

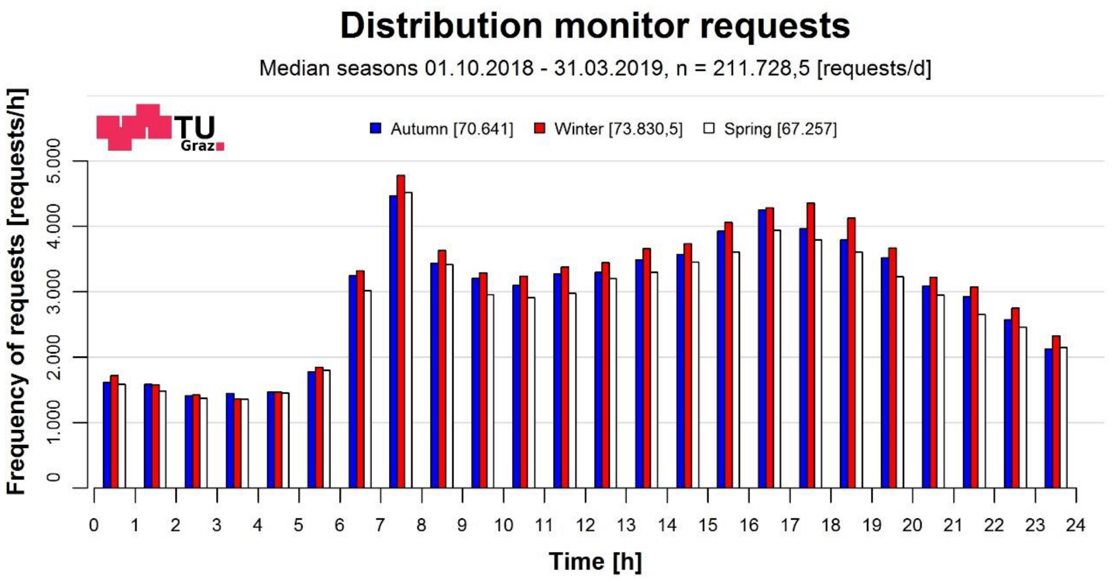


**Figure 21: Distribution monitor requests, daily courses, median seasons**

Thursdays are on average the most requested days of the week. Overall, the came to 74.604 requests per day. The course is very similar to the median, the values are slightly higher.

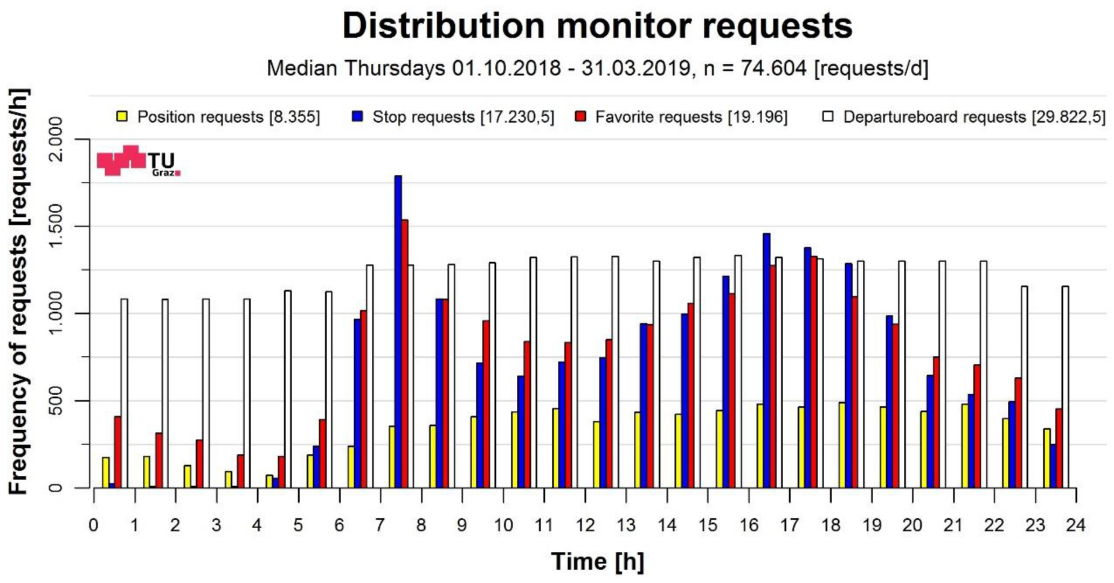


**Figure 22: Distribution monitor requests, daily courses, median Thursdays**

The courses of the weekdays look all pretty the same to the course of the median over six months. In contrast, the distributions at the weekend are completely different. Sundays, in particular, have the fewest requests per day and do not show any significant peaks. It is logical that the curve starts to increase later, because the workers and student traffic for the most part is eliminated in the morning hours. From midday to the evening hours it was then increasingly requested. On Sundays only 57.403 requests were made per day.

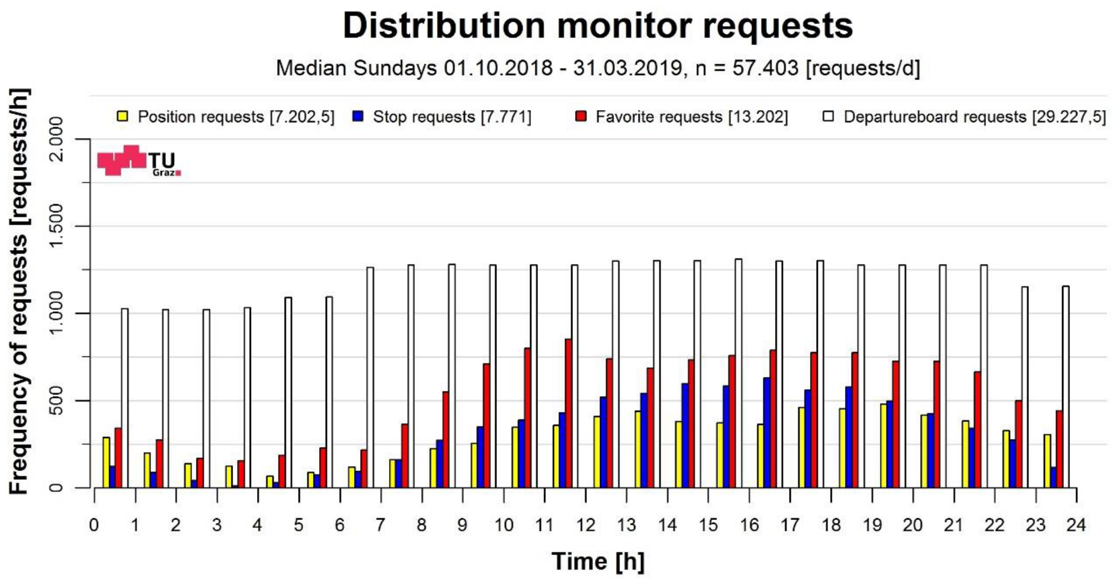


**Figure 23: Distribution monitor requests, daily courses, median Sundays**

In addition, the day with the highest amount of monitor requests is displayed in Figure 24. This is the 23.01.2019 with a total of 90.324 requests that were sent that day. Due to the superior night hours the departure board requests take again the largest part on that day, but not as striking as before. The stop and favorite requests achieve significantly higher morning peak values. In the morning peak hour between 07:00 and 08:00 o'clock over 3.000 stop requests were sent.

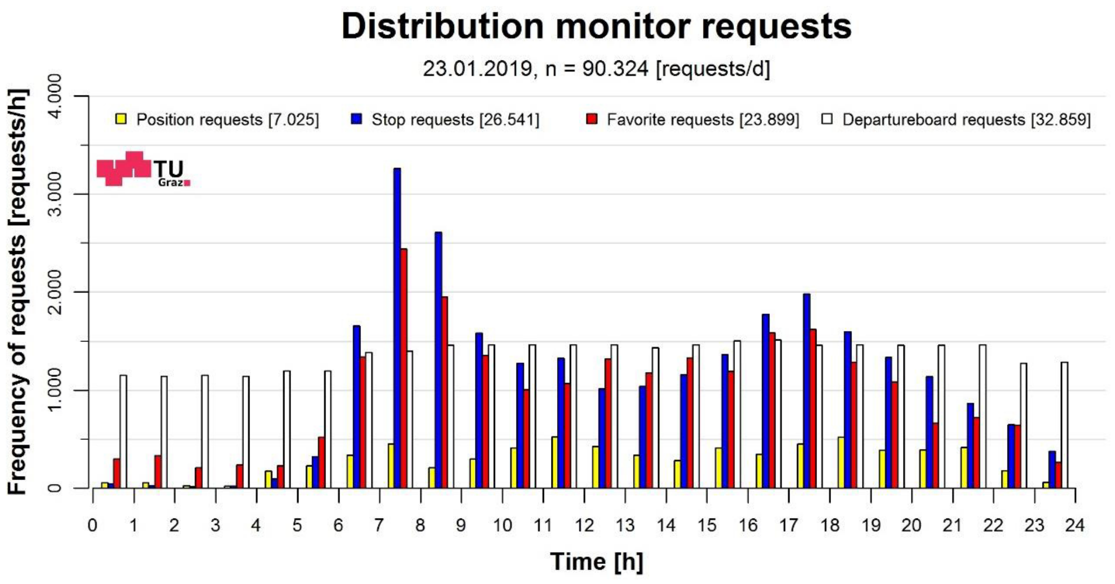


**Figure 24: Distribution monitor requests, daily courses, 23.01.2019, most requested day**

### 4.1.3 Frequency Plots

The distribution plots only contain the temporal distributions of the requests. They do not give information about individual stops or positions. So that the frequencies and the positions of the requests can also be shown frequency plots are used for representation. City districts of Graz are displayed in the background as the basis of the plots. There are 17 city districts in Graz, which are shown in Figure 25. For using the city districts in RStudio these are exported from the traffic model of the TU Graz out of the traffic planning software PTV Visum. The file format is a shapefile and it is used to save information about the geometric position and attributes of geographic objects. Geographic objects can be points, lines or polygons (Environmental Systems Research Institute, Inc, 2016). These shapefiles can be loaded and plotted in RStudio. By means of coordinates the positions of the points are determined and so these can be plotted at the exact location on the city districts. Each stop is assigned with the corresponding coordinates and the coordinates for positions requests are already available. Different frequencies in plots are shown with the help of the size of points. The bigger the circle of a point, the more requests he has received. Because of this representation it can be clearly seen where the hotspots of the requests lie. The circles are scaled based on the surface. That means, a value twice as high has twice the surface. Therefore, the circles cannot be compared linearly by the diameter, but only quadratically over the surface. How the frequency tables are created is visible in the following Pseudo-Code.

```
Pseudo-Code: Frequency tables for position and stop requests

##---------------------------------------------------------------------------

Read in sorted monitor table
Delete rows without coordinates in the monitor table
Divide the new monitor table to 2 matrices --> stop and position matrix
Overwrite matrices with calculated frequencies for requests per coordinate
   and date
Write coordinates into the first two columns of both matrices
Change format of date to "dd.mm.YYYY" and write it to column names
Calculate all medians for every coordinate and add them at the end of the
   table
Change the column names for the medians part
Build the sum for every column
Write all 2 tables to .csv

##---------------------------------------------------------------------------
```

To get a good overview of Graz, Figure 25 shows the city districts of Graz. The numbering begins in the center and forms helically outward. In Table 10 the belonging names of the 17 city districts are declared.

**Table 10: Graz city districts**

| Graz city districts | | |
|---|---|---|
| 1. Innere Stadt | 7. Liebenau | 13. Gösting |
| 2. St. Leonhard | 8. St. Peter | 14. Eggenberg |
| 3. Geidorf | 9. Waltendorf | 15. Wetzelsdorf |
| 4. Lend | 10. Ries | 16. Straßgang |
| 5. Gries | 11. Mariatrost | 17. Puntigam |
| 6. Jakomini | 12. Andritz | |

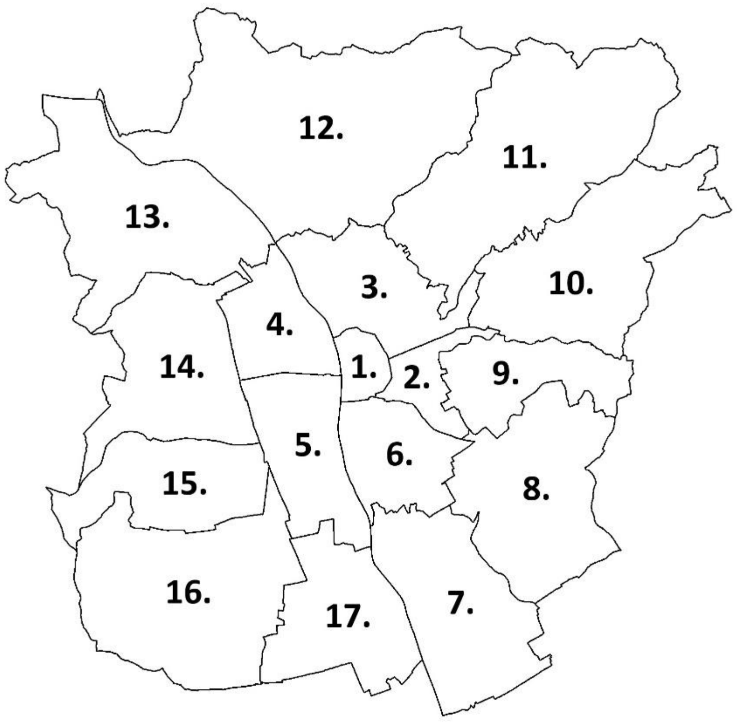


**Figure 25: Graz city districts**

The frequency plots for stop and position requests are shown below.

### 4.1.3.1 Stop requests

With blue circles and black borders, the stop requests are displayed in the representations. Figure 26 shows the median frequencies over six months, wherein 12.998 requests per day were carried out on average. It can be seen that the inner area of the city (zoomed in in Figure 27) contains most of the requests. The further the look goes out the smaller become the circles. In the city center, of course, the number of people using public transport is much larger than in outer residential areas. Also, the density of stops is higher in the central area. However, remarkable are end stations of tram lines or transfer points in Andritz, St. Leonhard, Liebenau and Puntigam. Here it comes to an encounter of trams, busses and in Puntigam and Liebenau also the railway is present. At this points several lines are coming together and therefore they are then interesting for stop requests to know when the corresponding connection leaves at the stop. Number of requests and departing lines for end stations can be seen in Table 11. With 171 requests per day the end station Liebenau/MURPARK was the most frequently requested station compared to the other three. The shopping center MURPARK and the railway station adjacent to the stop certainly contribute to this. The stops Andritz and St. Leonhard/Klinikum Mitte have more or less the same values and Puntigam Bahnhof only comes to 108 requests per day despite an additional connection to the railway.

**Table 11: End stations with number of requests and departing lines, median six months**

| Stop | Requests per day | Lines |
|---|---|---|
| Liebenau/MURPARK | 171 | 4, N4, 13, 64, 64E, 72, 74, 74E, 75U |
| Andritz | 146,5 | 4, 5, N5, 41, 41E, 52, 53, 53E |
| St. Leonhard/Klinikum Mitte | 142,5 | 7, N7, 41, 58, 64, 64E |
| Puntigam Bahnhof | 108 | 5, 62, 64, 65, 65E, 78, 80 |

St. Leonhard/Klinikum Mitte is strictly speaking not an end station of a tram line. It was until recently line 7 was extended by one stop. Now LKH Med Uni/Klinikum Nord is the new end station. St. Leonhard/Klinikum Mitte has remained a transfer point between trams and busses and is therefore listed here as an end station.

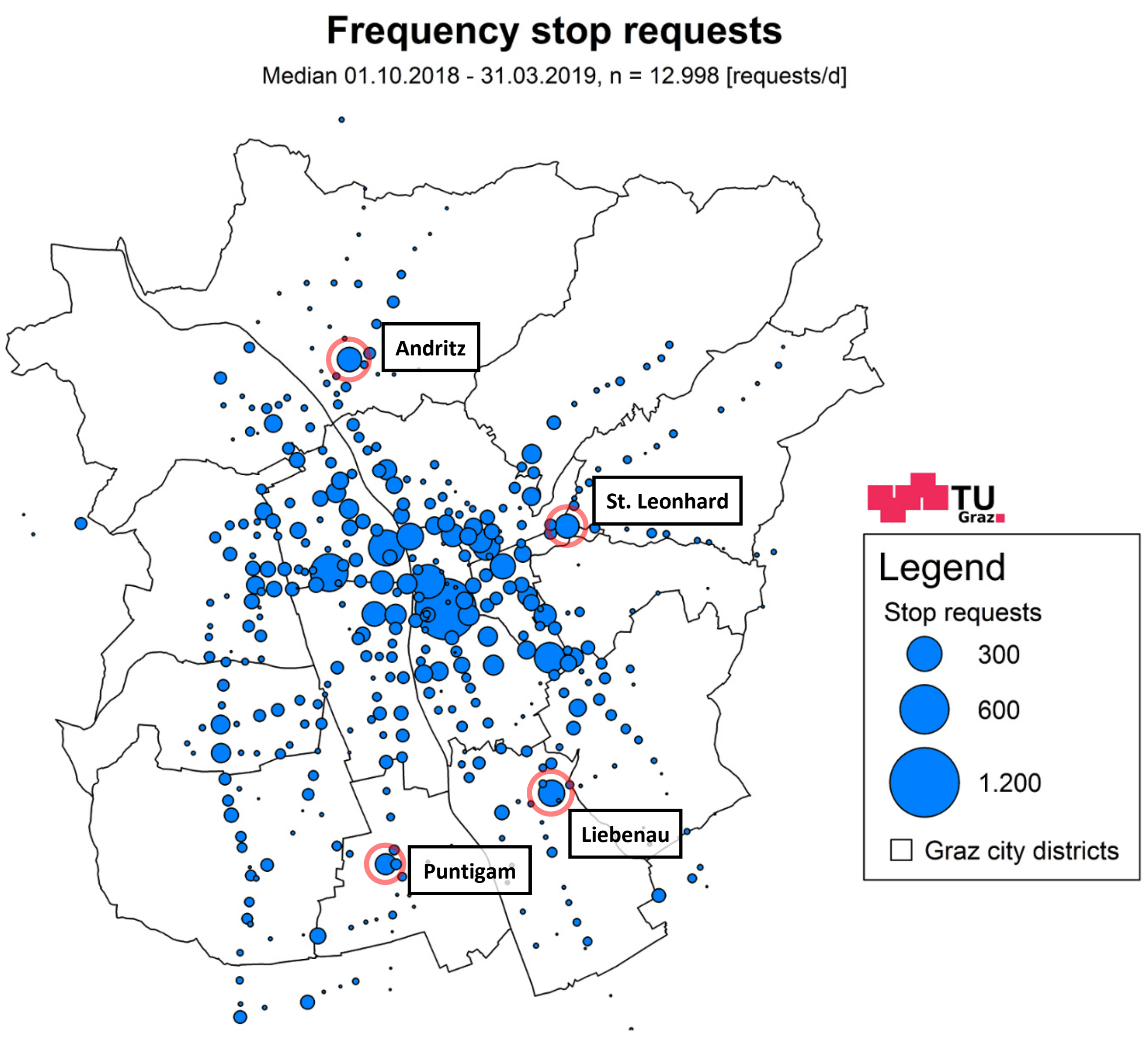


**Figure 26: Frequency stop requests, daily values, median six months**

When zooming to the inner city several hotspots can be seen in the city center. The largest circle has Jakominiplatz, the central transfer point of Graz. There, all tram lines and multiple bus lines come together star-shaped. Therefore, it is no surprise that this stop was the most frequently requested one on average. The main railway station also stands out clearly. This station is very much in demand with its transfer possibilities between the means of transport tram, bus and railway in local and long-distance traffic. Stop points of individual lines are quite far apart and require a longer walking time. Then it is good to know what the next departure times of the desired line are. If then a longer waiting time is displayed in the app this may be bridged for example by a short shopping in the supermarket or in the bookstore. Also, the main square and Lendplatz are blatant in the figure. At the Lendplatz only bus services are available, at the main square opposite only tram services. In Figure 27 additional two areas are identified. One area marks the area of Karl Franzens University (KFU) and the other one the TU and the school center next door. In these sections several stops were requested by qando users and if they are added together and considered as one, they yield a not extraneous share. On the basis of this information a slight trend is recognizable, that many students use qando for getting information, as the universities and the main railway station receive many requests. The table below shows the exact values of the most requested points in requests per day.

**Table 12: Most requested stops with number of requests, median six months**

| Stop | Requests per day |
|---|---|
| Jakominiplatz | 926,5 |
| KFU | 549,5 |
| TU, school center | 390,5 |
| Main railway station (Hauptbahnhof) | 352,5 |
| Lendplatz | 316 |
| Main square (Hauptplatz) | 284,5 |

Jakominiplatz is by far the most popular place with 926,5 requests per day. Followed by the areas KFU and TU, school center. The main railway station is the second separate stop behind Jakominiplatz with 352,5 requests per day. Below are than the Lendplatz with 316 and the main square with 284,5 requests per day.

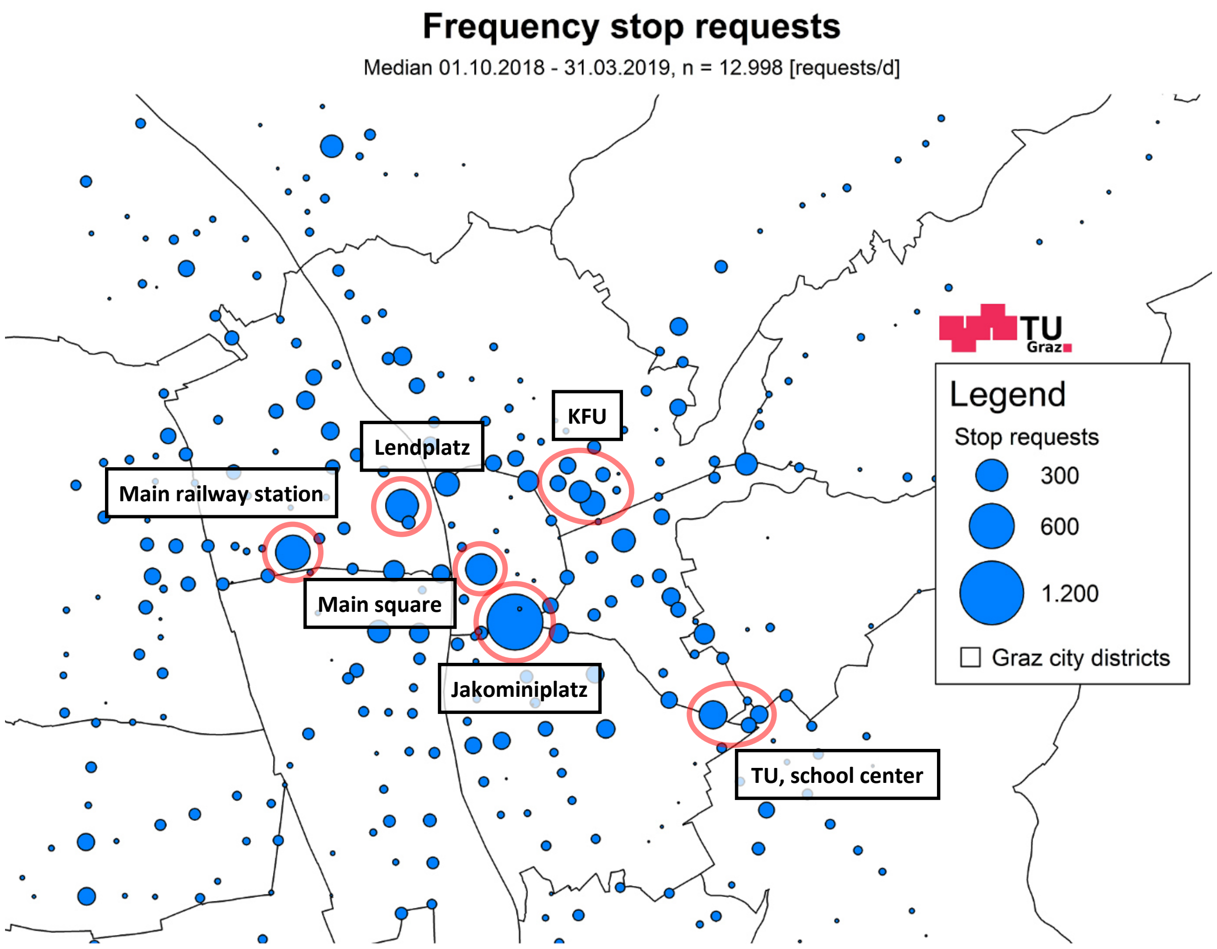


**Figure 27: Frequency stop requests, daily values, median six months, zoomed to inner city**

The highest average requests per day were sent on Wednesdays. On average 15.402 requests were made per day on these days. It is noticeable, that all circles of the stops are generally larger than the average values. One big reason for that is that the weekends with fewer requests are not included in this representation. Compared to Tuesdays (15.283 requests per day) and Thursdays (15.166 requests per day), which are also located during the week, Wednesdays can prevail.

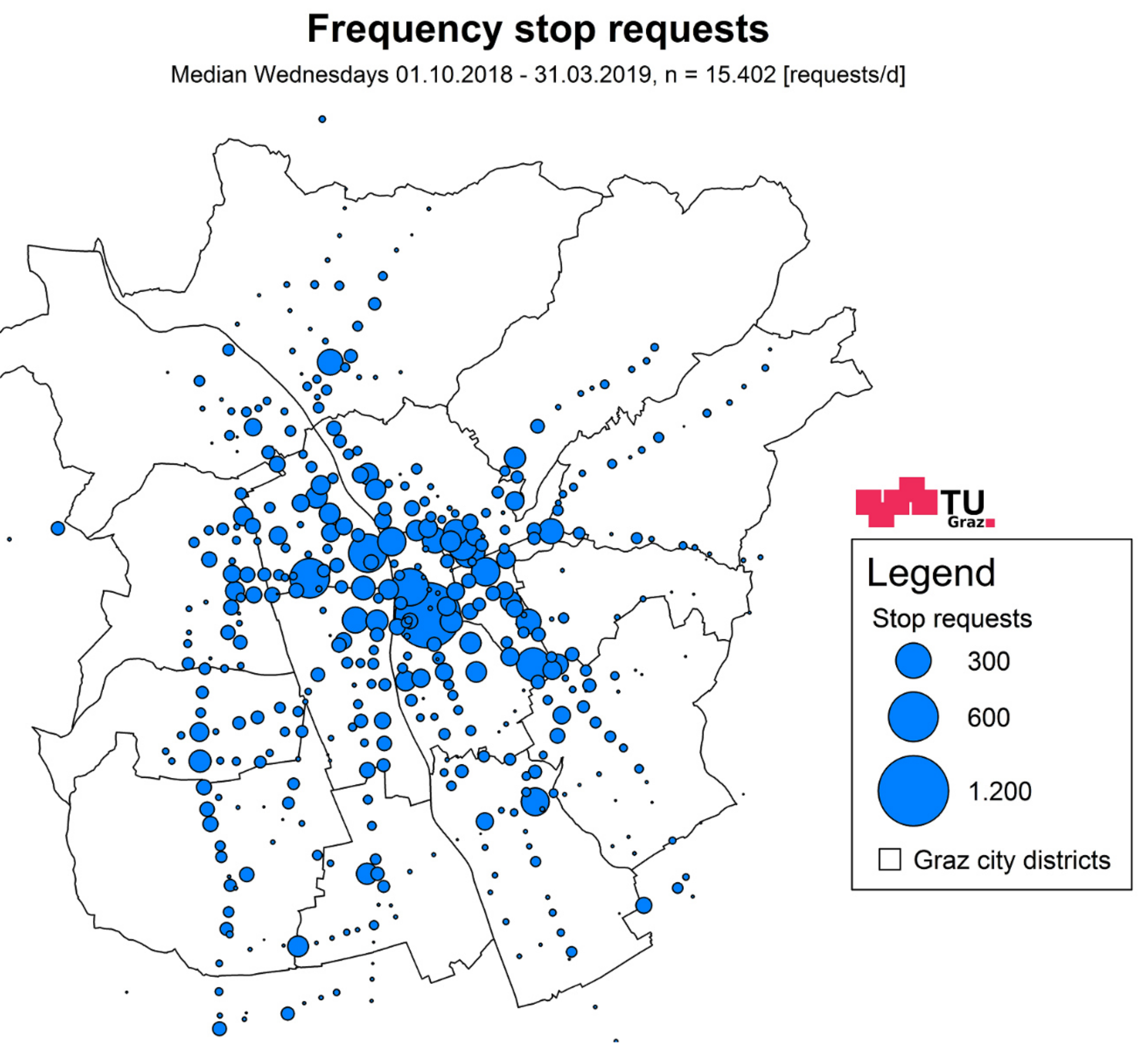


**Figure 28: Frequency stop requests, daily values, median Wednesdays**

For the sake of completeness, the frequency plot of the average values of Sundays are indicated to show that the number of requests at weekends is much lower. The number of requests is about half of the median over six months. Good to see in the sizes of the circles.

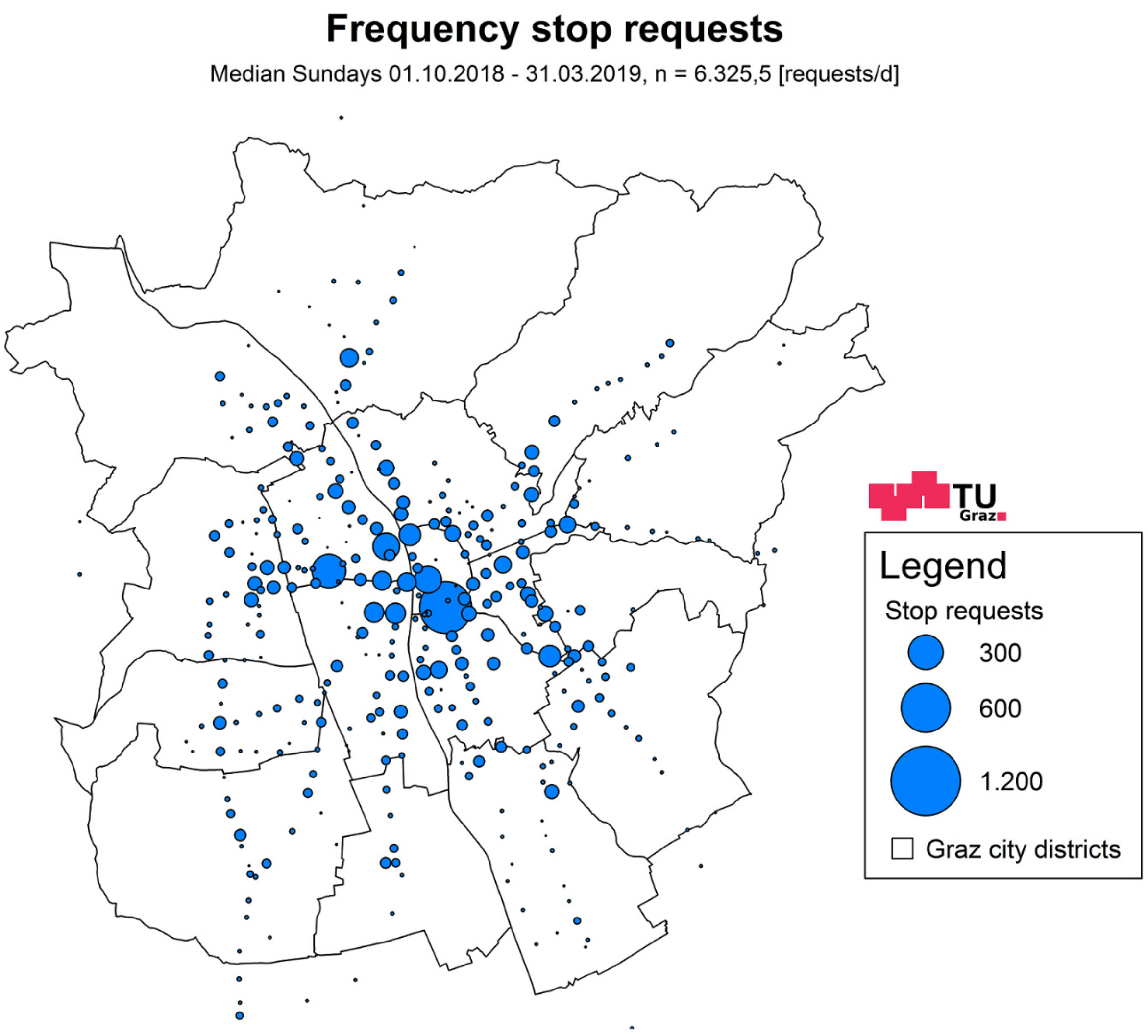


**Figure 29: Frequency stop requests, daily values, median Sundays**

Figure 30 shows the frequency of stop requests for the most requested day, the 23.01.2019. On this day 26.541 stop requests were made altogether, which is more than twice the average day. Particularly noteworthy in the figure is the stop Hertzgasse. This stop is located in the south-east of Graz in a residential area. In the surroundings no facilities are visible that would cause such high requests and only the bus line 64 operates at this stop. That is why this stop has been specially examined and it turns out that almost all requests from this stop came from one user. Every 30 seconds a request was sent by the web application of qando. Because of these requests comes the assumption that this user has positioned a monitor at home, on which the departure times of Hertzgasse are displayed. By reading out the application programming interface (API) of qando it is possible for the user to get the departure times. With requests every 30 seconds the user is always well informed about the departure times of the stop. Thus, this stop got 2.201 requests on that day. But why was the stop not distinctive in the representation of the average values? This is because the stop was especially requested in winter. Compared to the winter the number of requests in autumn and spring were very little and by building the median over six months the high and low values are eliminated and the middle one remains. Because of building the median such outliers are not as significant and stops like the Jakominiplatz, which was requested much more constantly over time, are strongly represented in the average values.

Also, the overall maximum value of all days in six months comes from the Hertzgasse. This happened on 13.12.2018 with a number of 2.859 requests per day.

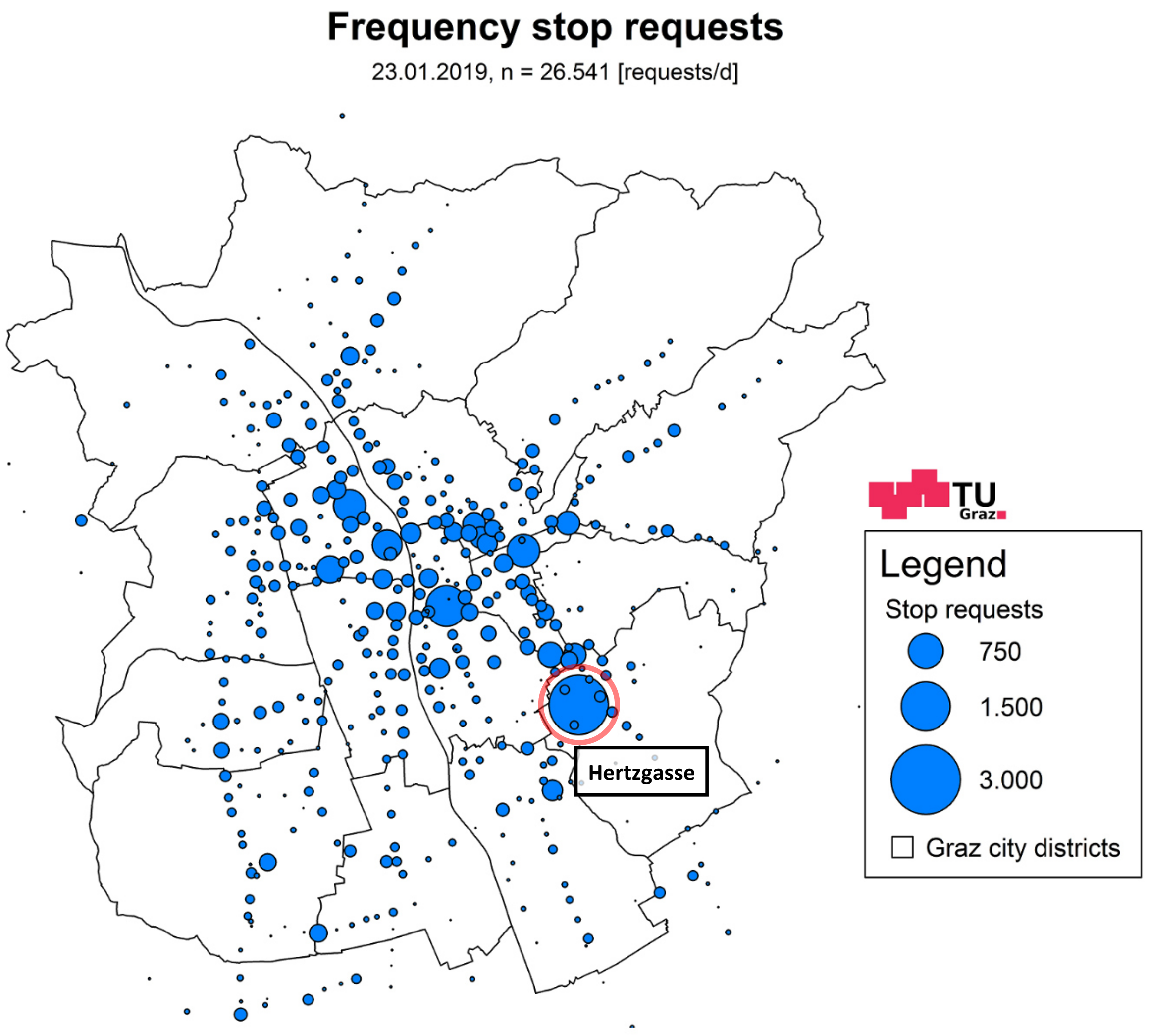


**Figure 30: Frequency stop requests, daily values, 23.01.2019, most requested day**

To give an overview about the hotspots Table 13 shows the medians at stops in requests per day. In the first column the total requests per day of all stops are listed. The maximum values of the requests are marked and occur more likely between Tuesday and Thursday.

**Table 13: Medians at hotspots in requests per day**

| | Total | Jakomini-platz | Main railway station | Lend-platz | Main square | KFU | TU, school center | Hertz-gasse |
|---|---|---|---|---|---|---|---|---|
| Median 6 months | 12.998 | 926,5 | 352,5 | 316 | 284,5 | 549,5 | 390,5 | 72,5 |
| Median autumn | 13.067 | 1.173 | 345 | 319 | 299 | 692 | 375 | 47 |
| Median winter | 13.871,5 | 929 | 372,5 | 325 | 294 | 535 | 430,5 | 364,5 |
| Median spring | 12.703 | 796 | 334 | 304 | 256 | 493 | 357 | 84 |
| Median Mondays | 14.834,5 | 916 | 398 | 356 | 288,5 | 710 | 455,5 | 146 |
| Median Tuesdays | 15.283 | 1.062 | 406,5 | 323 | 303 | 799,5 | 490 | 105 |
| Median Wednesdays | 15.402 | 1.041 | 379 | 353 | 337 | 815 | 458 | 73 |
| Median Thursdays | 15.166 | 1.021 | 416,5 | 352,5 | 334,5 | 688 | 437,5 | 67,5 |
| Median Fridays | 14.766,5 | 1.025,5 | 361 | 359 | 307,5 | 570,5 | 470,5 | 89 |
| Median Saturdays | 9.291 | 773 | 251,5 | 217 | 245,5 | 131,5 | 221,5 | 55,5 |
| Median Sundays | 6.325,5 | 675 | 283 | 180,5 | 182,5 | 65 | 176 | 34,5 |

So far, the stops have only been considered individually. Below all requests from stops are assigned to their respective districts, that means that the number of requests in a district are summed up. This gives information in which area most of the requests were made.

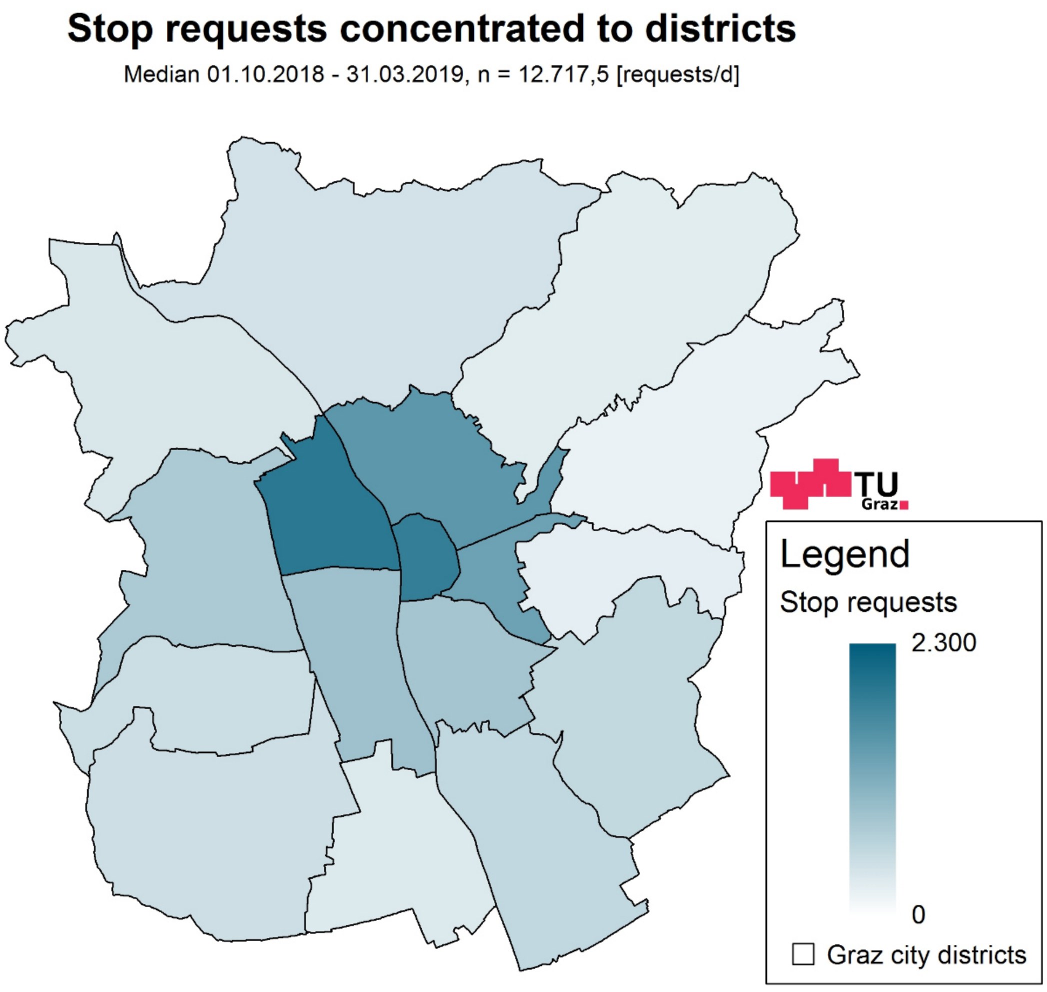

Stop requests concentrated to districts
Median 01.10.2018 - 31.03.2019, n = 12.717,5 [requests/d]
TU Graz
Legend
Stop requests
2.300
0
Graz city districts

**Figure 31: Stop requests concentrated to city districts, median six months**

The total values related to the city districts of Graz are lower compared to the median of the normal frequency plot in Figure 26. The reason for this is that stops outside the city districts are also present at the frequency plots of the stops and these naturally fall out when concentrating on the districts. 12.717,5 requests per day yield by building the median over six months for the requests concentrated to districts. Figure 31 shows the 17 city districts with the average occurred frequencies over six months in the districts. The darker the district is colored the more requests he got. As was to be expected from the figures above, the inner districts of Graz are the darkest. Outstanding are here the districts inner city, Lend, Geidorf and St. Leonhard. These districts include the stops with most of the requests, like Jakominiplatz, main railway station, Lendplatz and KFU. Outer districts have despite their size a much lower number of requests per day. Because of the highest average requests per day Figure 32 displays the median of Wednesdays related to the city districts in requests per day.

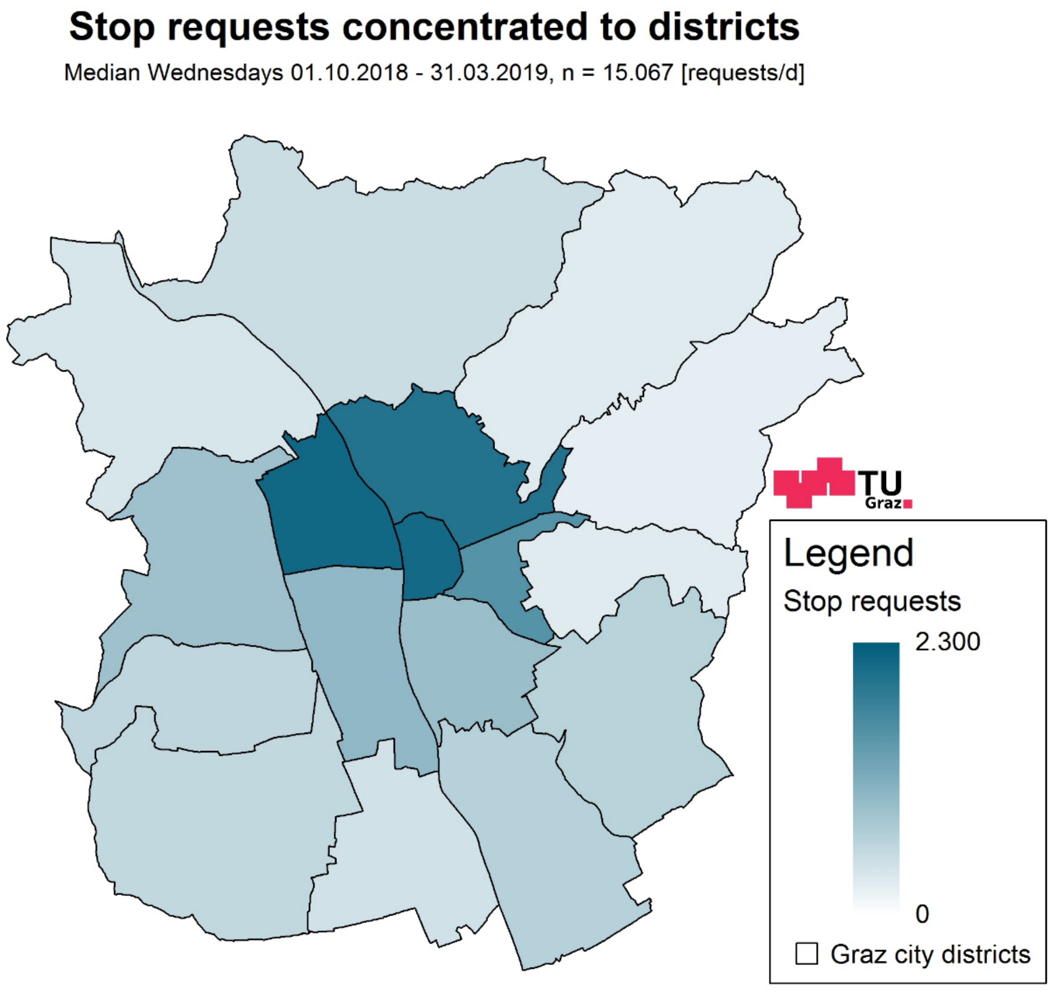

Stop requests concentrated to districts
Median Wednesdays 01.10.2018 - 31.03.2019, n = 15.067 [requests/d]
TU Graz
Legend
Stop requests
2.300
0
☐ Graz city districts

**Figure 32: Stop requests concentrated to city districts, median Wednesdays**

### 4.1.3.2 Position requests

With position requests the actual position of the users is used via GPS. As with the stop requests the frequency is displayed in circles. The accuracy of the coordinates is to the six decimal places, which makes a multiple demeanor of the same coordinates unlikely. In Figure 33, however, it can be seen that despite this accuracy a high number of requests can occur at one coordinate. These are then requests that likely came from the same user and were possibly generated by an automatic query. If the user is in a movement while requesting, a line of points results. Several of such lines can be seen in the northwest of Graz. The upper line shows requests on the Wienerstraße, the lower one shows requests on the railway track. In general, most requests arose in the inner city and also little areas with accumulations are noticeable.

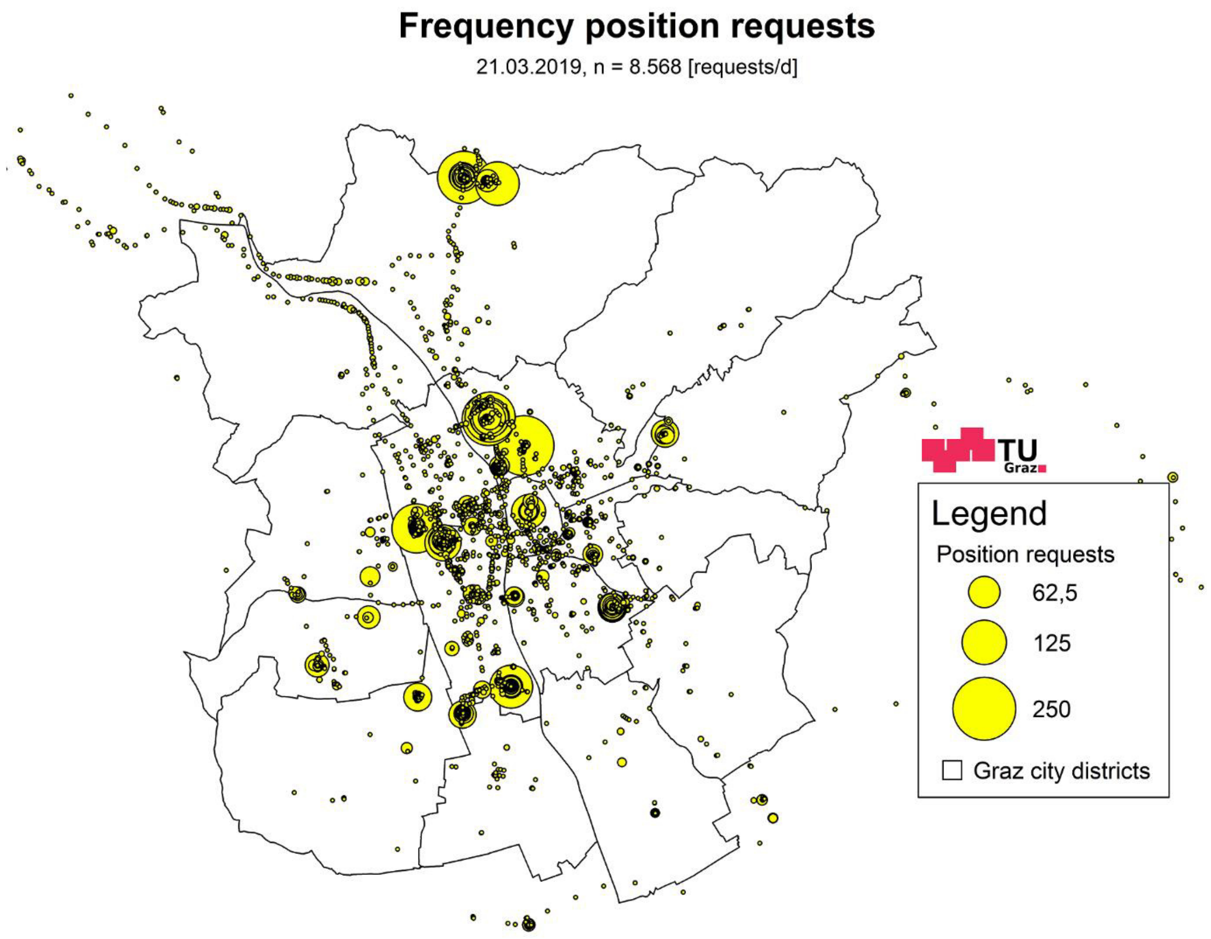

Frequency position requests
21.03.2019, n = 8.568 [requests/d]
TU Graz
Legend
Position requests
62,5
125
250
Graz city districts

Figure 33: Frequency position requests, daily values, 21.03.2019

Most position requests were carried out on 21.02.2019 with 16.567 requests per day. The highest frequency at one point is 2.859 request per day on 29.12.2018. There exist no medians for the position requests. Due to the fact that coordinates usually only occur on one day there are too much zeros in the matrix and so no median can be built.

## 4.2 Results route data

As well as with the monitor data the analysis results for the route data consider temporal distributions, frequencies and locations. In addition, representations of spider matrices and the results of public transport assignments of the qando data are displayed. These assignments are also compared to real passenger counts provided by the Holding Graz. Follow-up requests from the same user are not considered in the evaluations. These arise when after requesting a route earlier or later alternatives of the route should be displayed. In the data sets these follow-up requests are marked with a session ID (Identification).

### 4.2.1 Platforms used for route requests

The route requests were carried out from various platforms and their distribution is shown in Figure 34. Most of the route requests were sent from Android devices, nearly 300.000 requests in six months. After that are requests from iPhones with a number of 246.949 requests in six months. In the monitor data more requests were made with iPhones, in the route data this changed in favor of Android devices. A small number of requests were made by the web application and iPads. Others includes the remaining requests from platforms that are not often used, in this case only one request was made with such a platform. In total 547.041 requests were made in six months.

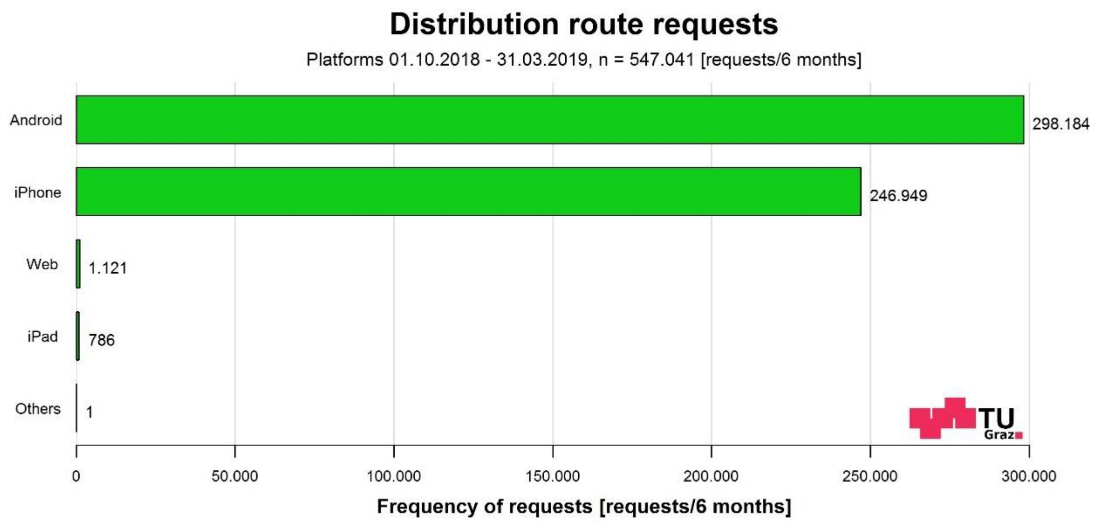


**Figure 34: Platforms used for route requests**

### 4.2.2 Distribution plots

The process of creating distribution plots is the same as for the monitor data, but there is no distinction between different types of requests.

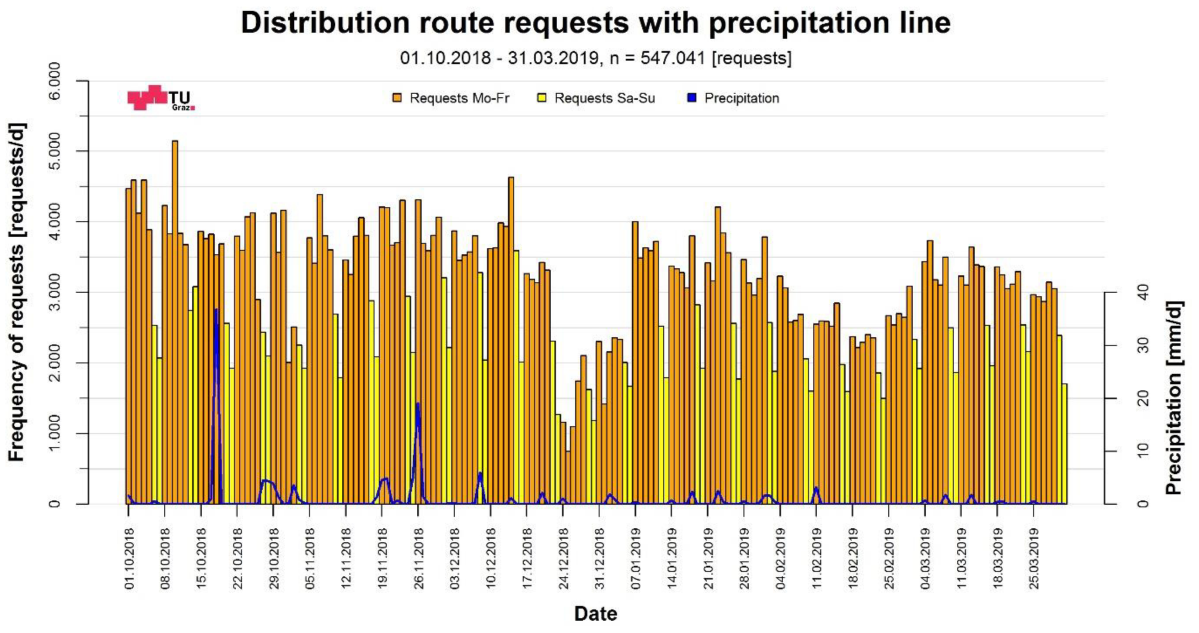


**Figure 35: Distribution route requests, every day of six months, with precipitation line**

In Figure 35 the distribution of the route requests over six months is shown. The representation corresponds to that of the monitor. Requests on weekdays are shown as orange bars, requests at the weekends as yellow bars and the blue line is the occurred precipitation. The x-axis contains the dates and every Monday is plotted, the y-axis shows the frequency of requests in requests per day.

Between October and March 547.041 route requests were made. Similar to the distribution of the monitor data the weekends are significantly lower than the other days of the week. The route requests have a bigger difference between Saturdays and Sundays. In autumn most requests were made, especially the first week in October stands out clearly. For students the winter semester began in this week, which suggests, because of the higher number of requests, that new students in Graz needed information about routes or already longer students refreshed their public transport knowledge. The day with the most route requests is the 10.10.2018 with 5.144 requests. A smaller number of requests is noticeable on National Day and on All Saint’s Day. In the Christmas holidays the collapse is more striking compared to the monitor data. The 25.12.2018 even has the lowest value of the whole six months, only 751 routes were requested on this day. As the holidays continued the course increased. In the weeks before Christmas shopping Saturdays in Graz are very popular, and this picture is also reflected in the qando route requests. Beginning in the middle of November a not be overlooked rising trend is evident. On 15.12.2018 this Saturday reached with 3.591 requests a similar value as the Monday and Tuesday of this week and is thus the most demanded Saturday. Also, during the semester break in February another collapse is visible, which is much more pronounced than in the monitor requests.

With regards to the seasons, most requests were sent in autumn. In winter was a decrease and spring got even fewer requests than the winter. In contrast to the monitor requests the route requests have several days of increased request values on which a precipitation was also measured. As with the monitor requests, the rainfall on 18.10.2018 was no reason for more requests. But the number of requests is above average on the following days and could be increased due to precipitation.

**Table 14: Weekdays with possible increased number of requests because of precipitation**

| Weekdays | |
|---|---|
| Monday, 29.10.2018 | Wednesday, 23.01.2019 |
| Monday, 19.11.2018 | Friday, 01.02.2019 |
| Tuesday, 20.11.2018 | Friday, 08.03.2019 |
| Monday, 26.11.2018 | Wednesday, 13.03.2019 |
| Friday, 18.01.2019 | |

The following results consider temporal distributions over a day. Figure 36 shows the distribution of the sum of all requests, with a total of 547.041 requests sent in six months. On the x-axis, the 24 hours of the day are plotted and on the y-axis the number of requests per hour. Similar to the monitor data a beautiful course arises with a morning peak and an evening peak or an evening area. In the night hours after midnight hardly any requests occurred, the course only begins to rise between five and six o'clock in the morning.

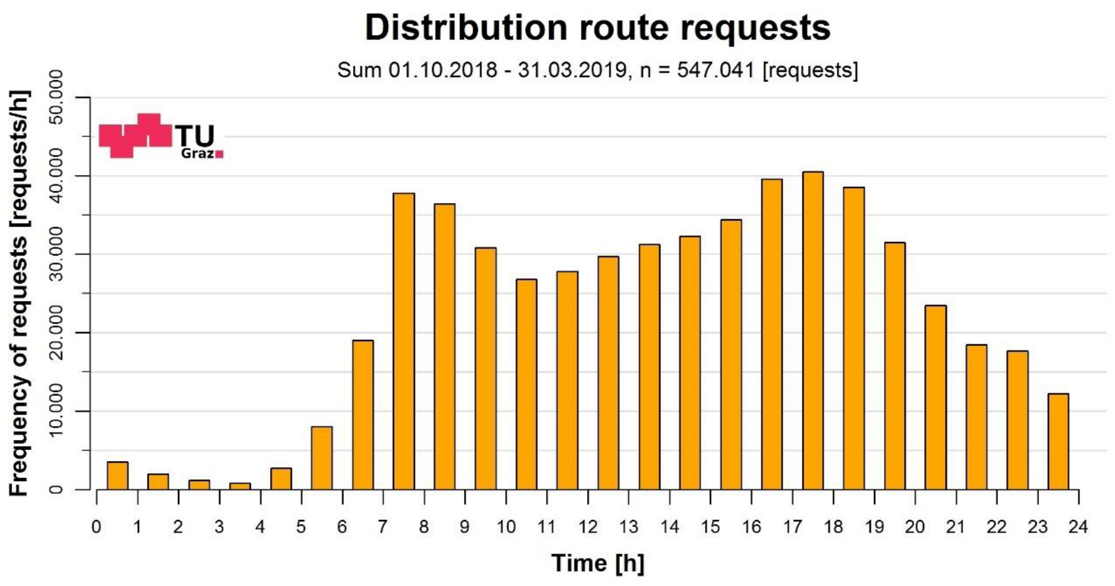


**Figure 36: Distribution route requests, daily courses, sum**

If the median of the temporally distributed route requests is considered over six months, it can be seen, that a beautiful course with two peaks has set again. In the night hours after 24 o'clock hardly any requests were made, which is very plausible, because in the early morning hours only on Saturdays, Sundays and public holidays connections are available. These connections are called nightline, are operated by buses and drive three times at 0:30, 1:30 and 2:30 star shaped from the Jakominplatz. In comparison Figure 40 shows the temporal distribution on Sundays and in this figure several requests are recognizable at night. In six months on average 3.042,5 route requests were made per day, which is very low compared to the monitor data with an average of 41.606,5 requests per day. This confirms the assumption that the information about departure times at a stop is preferred in relation to a route calculation.

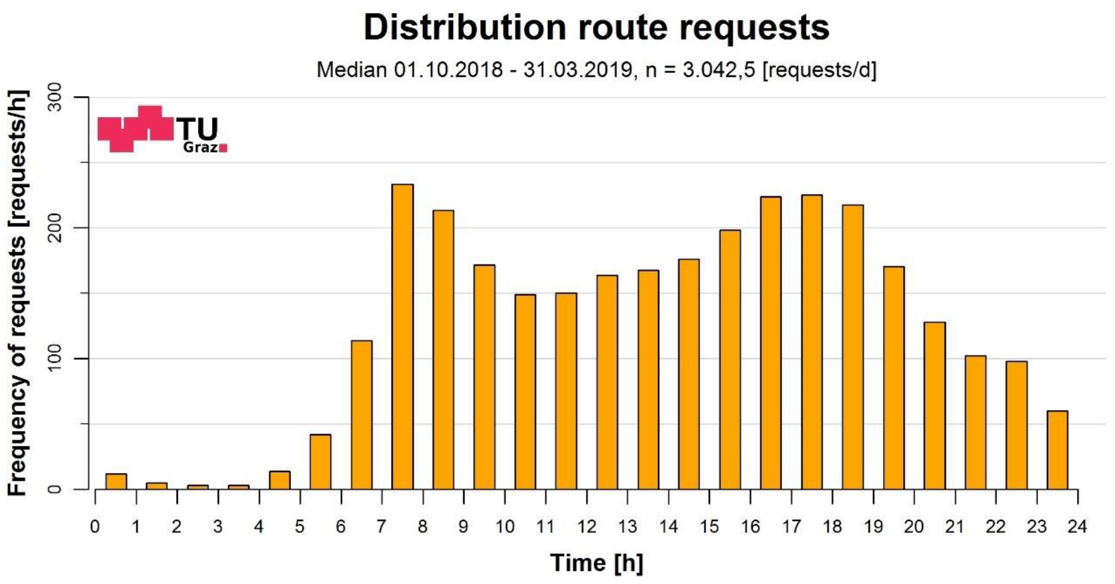


**Figure 37: Distribution route requests, daily courses, median six months**

Figure 35: Distribution route requests, every day of six months, with precipitation line already showed that in autumn more requests were made than in winter and spring. Figure 38 confirms this assumption and displays the temporal courses of the medians of autumn, winter and spring. Autumn has on average the largest share with 3.490 requests per day, in winter 3.062 and in spring 2.718 requests were made per day. That winter has fewer requests than autumn is largely due to the fact that Christmas holidays and a part of the semester break fall within this time span, which have far fewer requests for a longer period of time, as already seen in Figure 35. The lower number of requests in spring can be explained by the fact that the weather in February and March has already invited to travel more ways by bicycle or by foot and not to use public transportation. In the early morning hours, a similar course of all three seasons is present before in the later morning and in the afternoon, they are increasingly separated. On the later evening they become more uniform again.

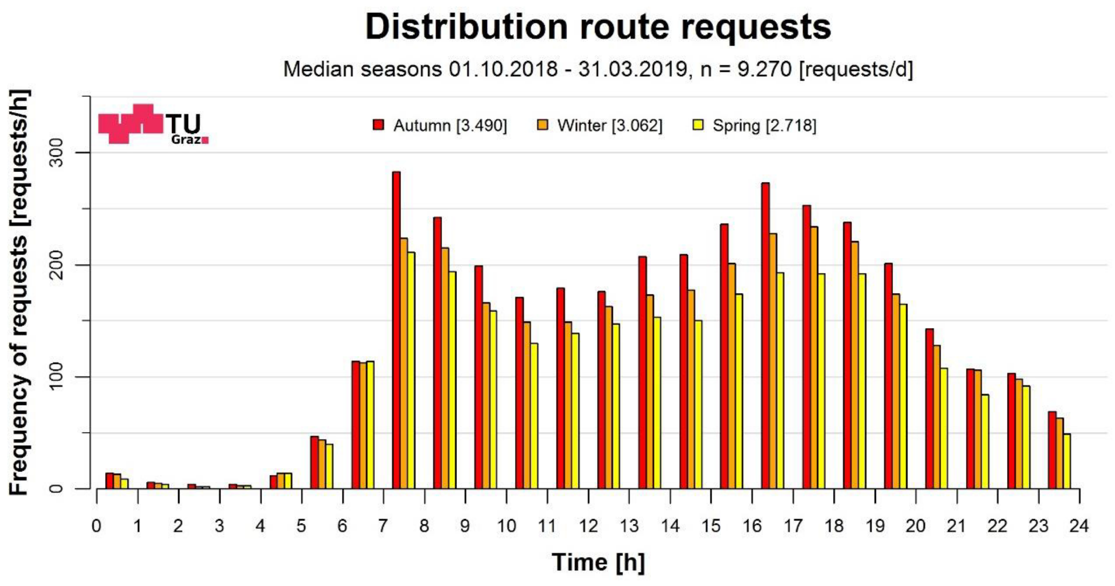


**Figure 38: Distribution route requests, daily courses, median seasons**

Mondays got on average the most route requests per day over six months. 3.468,5 were made and these are distributed according to the already known distribution. Hardly any requests in the night hours and from five o'clock in the morning a steep rise is visible. Between seven and eight o'clock lays the maximum with 309 requests per hour, the morning peak is only so distinctive on Mondays. At midday a little collapse in requests and in the afternoon again the evening peak.

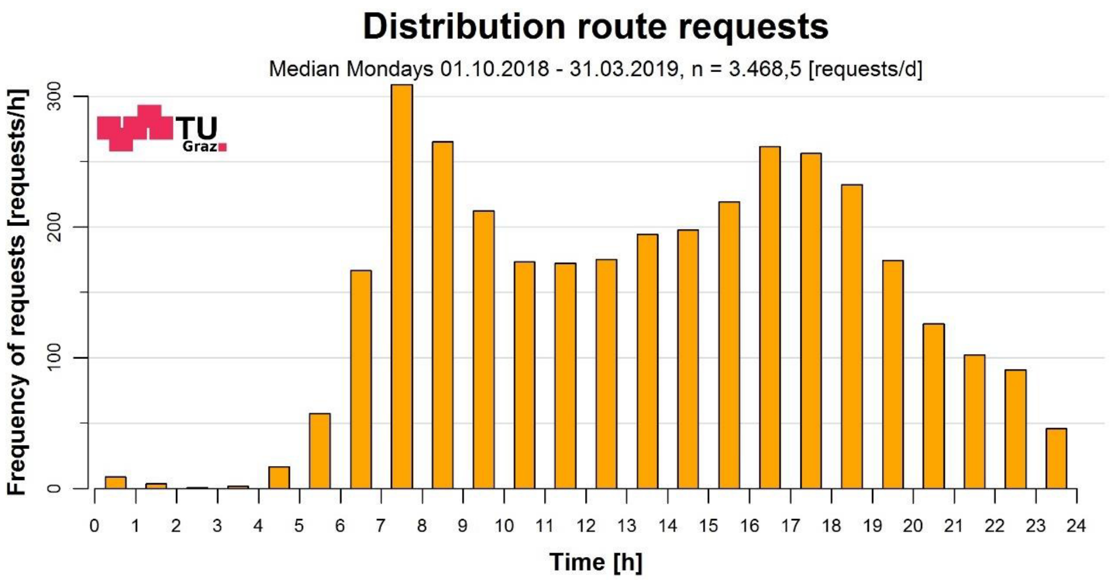


**Figure 39: Distribution route requests, daily courses, median Mondays**

On Sundays the fewest number of requests per day were carried out on average. With 1.838,5 requests per day the Sundays reached just over half compared to the Mondays. But as already mentioned, Saturdays and Sundays got the most requests in the morning hours. No morning and no evening peak are recognizable. The course increases very slowly, and most requests were made between five and six o'clock in the evening.

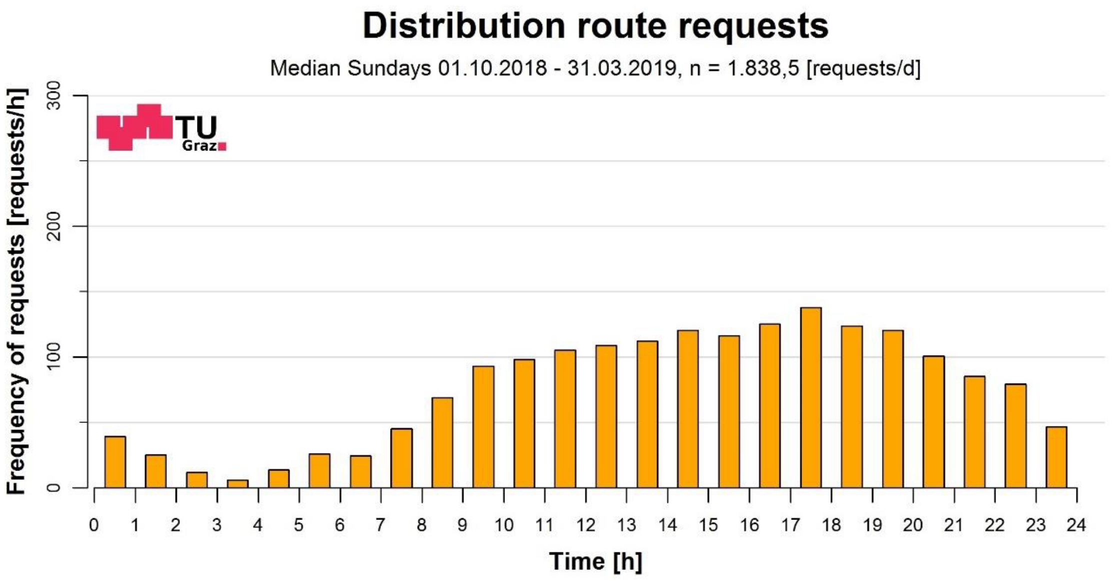


**Figure 40: Distribution route requests, daily courses, median Sundays**

### 4.2.3 Frequency plots

Frequency and location of the route requests are displayed in frequency plots. Thereby routes starting or ending at the actual position of the user are shown together with the requests at the stops in the figures. In the route requests a distinction between the start and the end of a route is made and these are displayed separately in the representations for the sake of clarity. The start of a route request is called route from request and the end of a route request is called route to request hereafter. The figures including route from requests are colored in green and the ones including route to requests are colored in orange. For the comparison between route from and route to requests the hotspots are summarized in a table. This comparison can be found in Table 23.

#### 4.2.3.1 Route from requests

The route from requests come to a median of 1.451,5 requests per day and are thus much lower than the monitor requests. As with the monitor requests most of it takes place in the inner city (zoomed in in Figure 42) and the further it goes out, the fewer requests were sent. However, there are again the already mentioned end stations, which stand out with higher numbers of requests in the outer areas. Of these the stop Liebenau/MURPARK had with 27 requests per day the most. Then there are the stations St. Leonhard/Klinikum Mitte and Andritz with 20 requests per day each and then Puntigam Bahnhof with 15. In Table 15 the end stations with number of requests and departing lines can be seen.

**Table 15: End stations with number of requests and departing lines, median six months**

| Stop | Requests per day | Lines |
|---|---|---|
| Liebenau/MURPARK | 27 | 4, N4, 13, 64, 64E, 72, 74, 74E, 75U |
| St. Leonhard/Klinikum Mitte | 20 | 7, N7, 41, 58, 64, 64E |
| Andritz | 20 | 4, 5, N5, 41, 41E, 52, 53, 53E |
| Puntigam Bahnhof | 15 | 5, 62, 64, 65, 65E, 78, 80 |

Furthermore, there are two more stops in the direction of Mariatrost, which were requested on average not insignificantly often and are already far outside of the city center. These stops are called Mariagrün and Kroisbach. Both of them are located at the line 1 and at Mariagrün there is also the possibility to change to the bus line 58. On average Mariagrün came to 13 requests per day and Kroisbach to 16, therefore they are comparable to Puntigam Bahnhof.

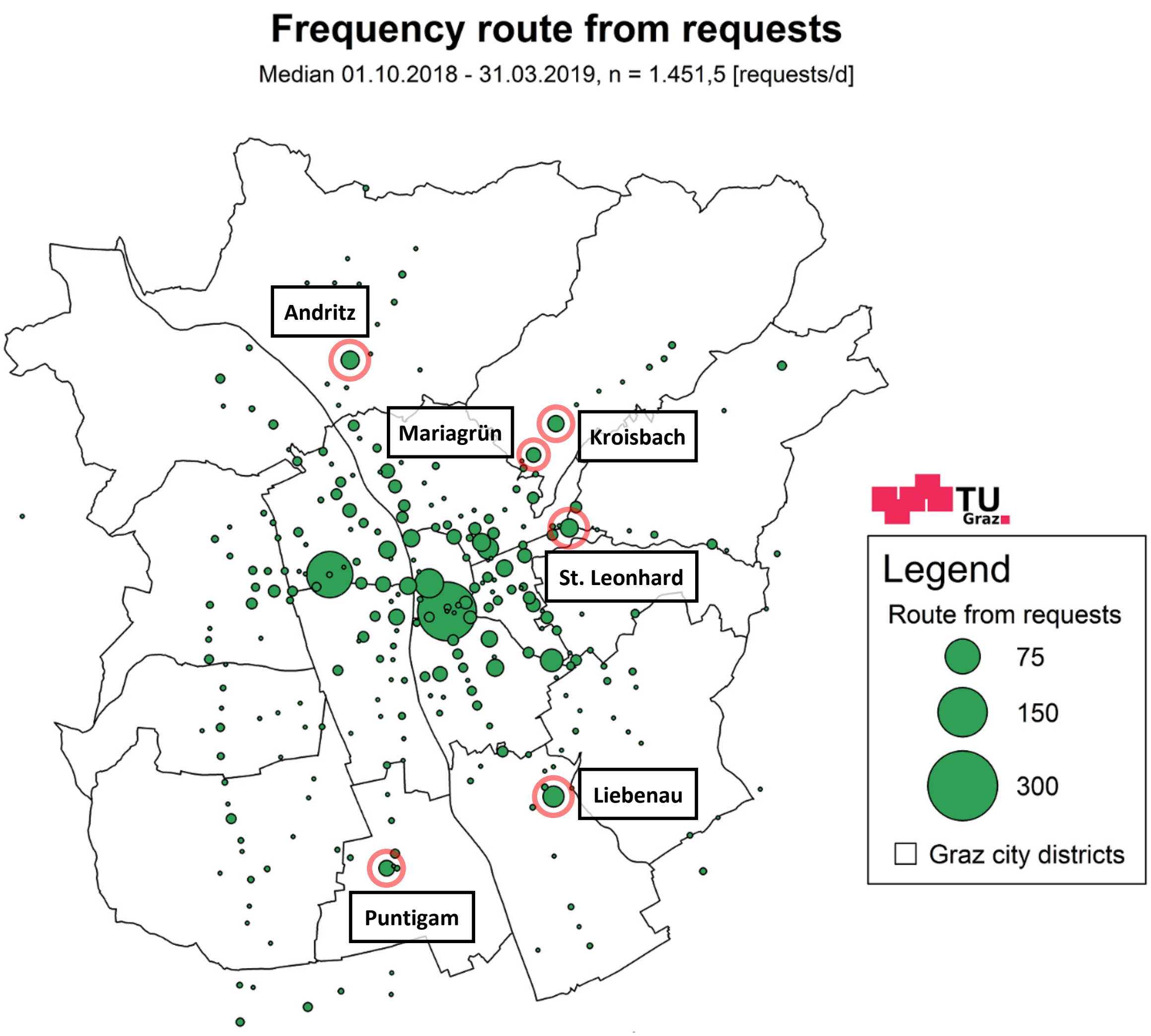


**Figure 41: Frequency route from requests, daily values, median six months**

As can be seen in Figure 42, the Jakominiplatz received by far the most route from requests with an average of 213 requests per day. Then come the main railway station with 131, KFU with 67 and main square with 51,5 requests per day. Less routes were beginning in the area of the TU and therefore this area does not really distinguish itself from the surrounding area. The following table shows the values of the most requested stops in requests per day.

**Table 16: Most requested stops with number of requests, median six months**

| Stop | Requests per day |
|---|---|
| Jakominiplatz | 213 |
| Main railway station | 131 |
| KFU | 67 |
| Main square | 51,5 |
| TU, school center | 42,5 |

Compared to the monitor requests the areas KFU and TU can be found further back in the route from requests. The main railway station is much more in demand in this representation. One reason for this is that a high number of requests in the area of KFU were using the actual position for calculating the route. As these are always different from day to day, they fall out at building the median. Good to see are the clusters of position requests in Figure 45 and Figure 46. In chapter 4.2.4 Origin-Destination information via spider matrices, requests using the actual position do not fall out.

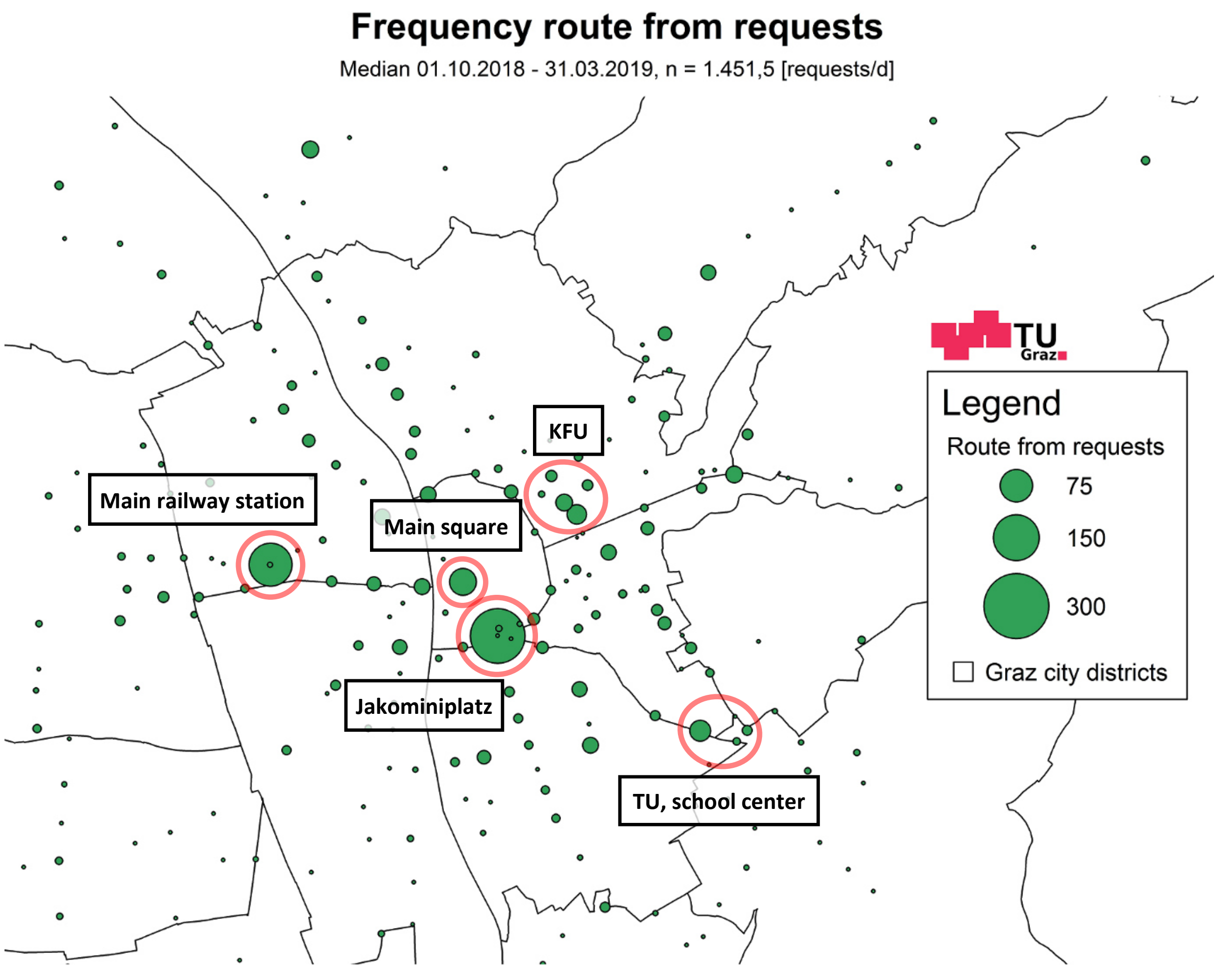


**Figure 42: Frequency route from requests, daily values, median six months, zoomed to inner city**

The highest average route from requests per day of the weekdays were sent on Mondays. On average 1.753 requests were made per day on these days. The frequency plot for the medians of the Mondays is shown in Figure 43 and is very similar to the figure of the median over six months, but the values are slightly higher. Also, the two stops Mariagrün and Kroisbach are in the same range as the end stations on Mondays. The comparison between the different medians can be seen in Table 17.

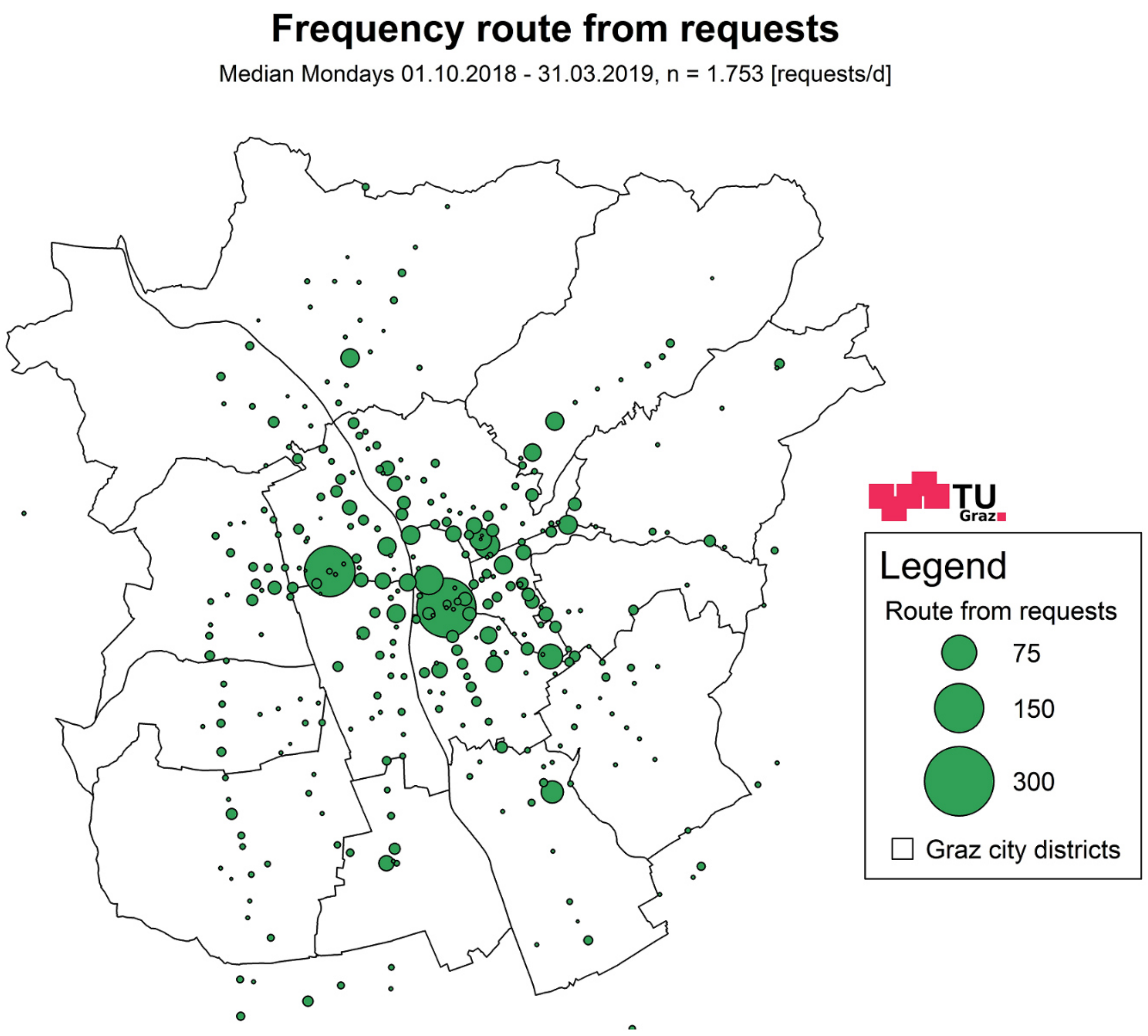

Frequency route from requests
Median Mondays 01.10.2018 - 31.03.2019, n = 1.753 [requests/d]
TU Graz
Legend
Route from requests
75
150
300
Graz city districts

**Figure 43: Frequency route from requests, daily values, median Mondays**

In contrast to the Mondays a completely different picture arises on Sundays. The median for the Sundays is displayed in Figure 44. Sundays, with 902 requests per day, have about the half of the requests of Mondays. With the Jakominiplatz (160,5 requests per day) and the main railway station (143 requests per day) only two stops stand out with higher numbers of requests, the other points are much smaller. The main railway station only reached a higher value than the 143 requests per day on Mondays. This indicates that on Sundays a large part of the qando users want to travel via the main railway station away from Graz or to Graz (compare chapter 4.2.4).

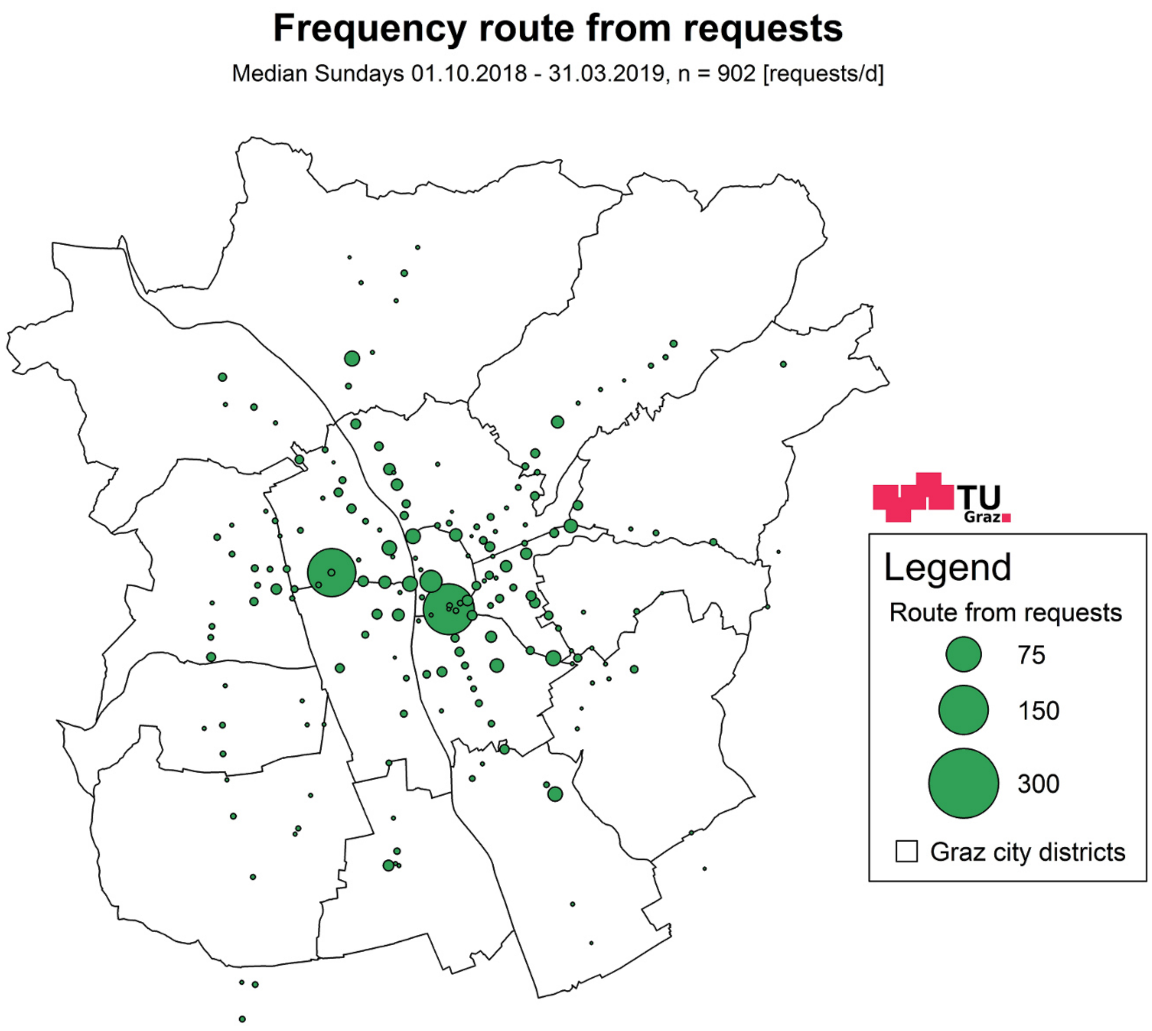

Frequency route from requests
Median Sundays 01.10.2018 - 31.03.2019, n = 902 [requests/d]
TU Graz
Legend
Route from requests
75
150
300
Graz city districts

**Figure 44: Frequency route from requests, daily values, median Sundays**

With 5.144 route requests per day Wednesday, 10.10.2018, is the most requested day in six months. This was already visible in the distribution of the route data over six months in Figure 35. Jakominiplatz (408 requests per day) and main railway station (244 requests per day) got the most requests and have thus the biggest circles. Also, other stops and the areas KFU and TU have higher values. In this large view of the entire urban area the details are difficult to see. Therefore, it is zoomed in to the inner city in Figure 46.

The small circles in those of the stops show points for which the actual position was used while requesting. As already mentioned, these position points fall out by building the median over longer periods of time, because the exact coordinates were not requested on several days. In the following two figures it is easy to see that these position requests are increasingly occurring in the areas of Jakominiplatz, main railway station, KFU and main square. With Jakominiplatz, main railway station and KFU three areas with lots of route from requests stand out clearly. In later evaluations this observation is confirmed. Due to the many requests in the area of KFU, there is a growing assumption that many students and people working in this area are using qando services.

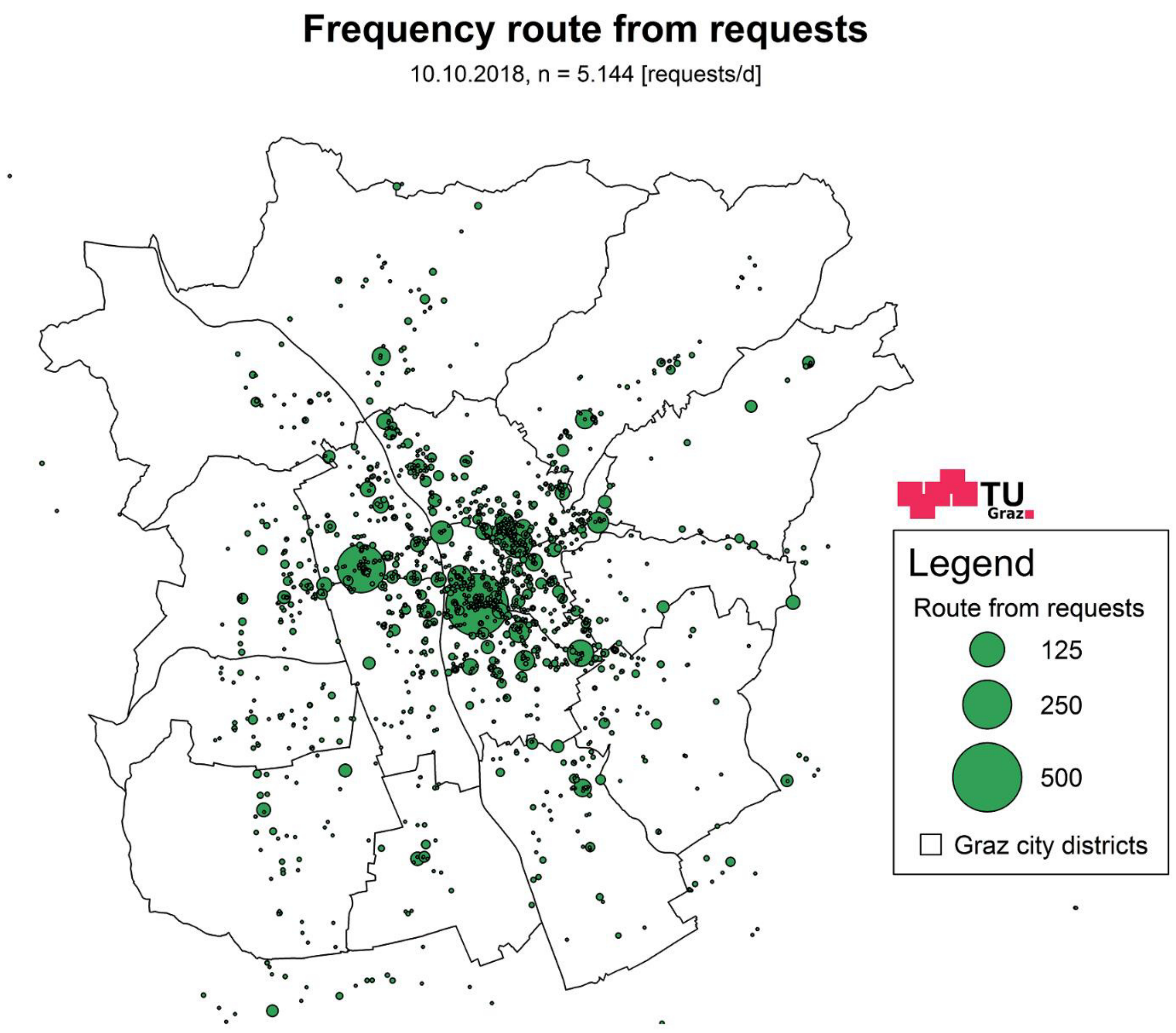

Frequency route from requests
10.10.2018, n = 5.144 [requests/d]
TU Graz
Legend
Route from requests
125
250
500
Graz city districts

**Figure 45: Frequency route from requests, daily values, 10.10.2018, most requested day**

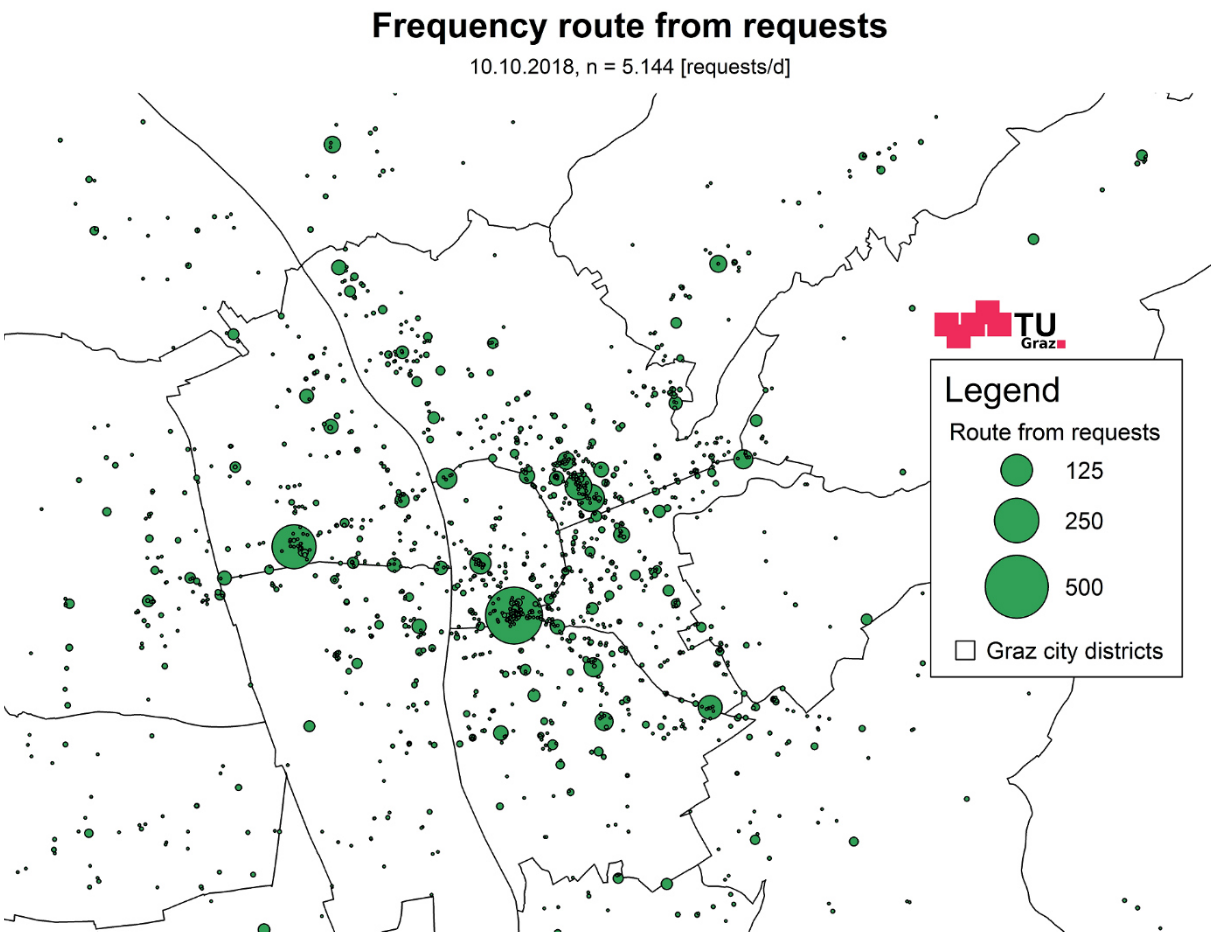

Frequency route from requests
10.10.2018, n = 5.144 [requests/d]
TU Graz
Legend
Route from requests
125
250
500
Graz city districts

**Figure 46: Frequency route from requests, daily values, 10.10.2018, most requested day, zoomed to inner city**

To give an overview about the hotspots Table 17 shows the medians at stops in requests per day, the first column shows the total requests per day of all stops. Maximum values of requests are marked and occur especially in autumn, on Mondays and on Wednesdays.

**Table 17: Medians route from at hotspots in requests per day**

| | Total | Jakomini-platz | Main railway station | Main square | KFU | TU, school center |
|---|---|---|---|---|---|---|
| Median 6 months | 1.451,5 | 213 | 131 | 51,5 | 67 | 42,5 |
| Median autumn | 1.719 | 265 | 155 | 53 | 119 | 48 |
| Median winter | 1.469,5 | 212,5 | 123 | 55,5 | 61,5 | 44 |
| Median spring | 1.315 | 187 | 125 | 44 | 42 | 35 |
| Median Mondays | 1.753 | 219 | 160,5 | 52 | 99 | 50 |
| Median Tuesdays | 1.643,5 | 221 | 121 | 54 | 107 | 53,5 |
| Median Wednesdays | 1.730 | 231 | 131,5 | 59 | 117 | 53,5 |
| Median Thursdays | 1.680 | 231 | 125 | 56,5 | 117,5 | 51,5 |
| Median Fridays | 1.714 | 258 | 139,5 | 57,5 | 82,5 | 45 |
| Median Saturdays | 1.192,5 | 182,5 | 112,5 | 50 | 26 | 26 |
| Median Sundays | 902 | 160,5 | 143 | 30 | 13 | 19 |

### 4.2.3.2 Route to requests

The route to requests came to a median of 1.784,5 requests per day and have more than 300 requests more than the median of the route from requests. But why is there a difference between route from and route to requests? Why do the route to requests have a higher number? This is because the median is built over six months and this is calculated separately for route from and route to request. The resulting higher route to requests are explained by the fact that the qando users use more consistent route ends, the number of requests at these points is thereby larger and therefore higher values remain after building the median. This is reinforced by increased use of the actual position as route start point, which then fall out with the median. Route from and route to requests are compared with hotspots in Table 23.

Similar to the route from requests a comparable picture results. The end stations have increased values again. In the route to requests the end station of tram line 7 LKH Med Uni/Klinikum Nord shows a higher number of requests. Tram line 7 is the only connection at this stop, a bus stop is a little further away, but this station is attractive because of the neighboring medical university and the dental clinic. Liebenau/MURPARK is, as with the route from requests, on average the most appealing end station and was requested 45 times per day. This is followed by the stations at the LKH and the medical university. The value at LKH Med Uni/Klinikum Nord is with 25 requests per day slightly smaller than the previous stop St. Leonhard/Klinikum Mitte with 32,5, but larger than at the end stations Andritz (21 requests per day) and Puntigam Bahnhof (19 requests per day). All end stations with number of requests and departing lines can be seen in Table 18.

**Table 18: End stations with number of requests and departing lines, median six months**

| Stop | Requests per day | Lines |
|---|---|---|
| Liebenau/MURPARK | 45 | 4, N4, 13, 64, 64E, 72, 74, 74E, 75U |
| St. Leonhard/Klinikum Mitte | 32,5 | 7, N7, 41, 64, 64E |
| LKH Med Uni/Klinikum Nord | 25 | 7 |
| Andritz | 21 | 4, 5, N5, 41, 41E, 52, 53, 53E |
| Puntigam Bahnhof | 19 | 5, 62, 64, 65, 65E, 78, 80 |

The stops Mariagrün and Kroisbach, which attracted positive attention in the route from requests, are not quite so attractive when it comes to requests at the end of the route compared to the route start.

It was previously mentioned that the values for the route to requests are larger than those for the route from requests and the qando users use more uniform route ends. Table 19 shows the comparison of the end stations between route from and route to requests and it is easy to see that the values at the stations in the route to requests are considerably larger. Only the station Andritz has about the same number of requests for route from and route to requests. Therefore, it can be said that more route requests end at end stations than they start there.

**Table 19: Comparison end stations between route from and route to requests, median six months**

| Stop | Route from [requests per day] | Route to [requests per day] | Lines |
|---|---|---|---|
| Liebenau/MURPARK | 27 | 45 | 4, N4, 13, 64, 64E, 72, 74, 74E, 75U |
| St. Leonhard/Klinikum Mitte | 20 | 32,5 | 7, N7, 41, 64, 64E |
| LKH Med Uni/Klinikum Nord | 8 | 25 | 7 |
| Andritz | 20 | 21 | 4, 5, N5, 41, 41E, 52, 53, 53E |
| Puntigam Bahnhof | 15 | 19 | 5, 62, 64, 65, 65E, 78, 80 |

The following Figure 47 shows the median of the route to requests over six months and the four distinctive points of the end stations.

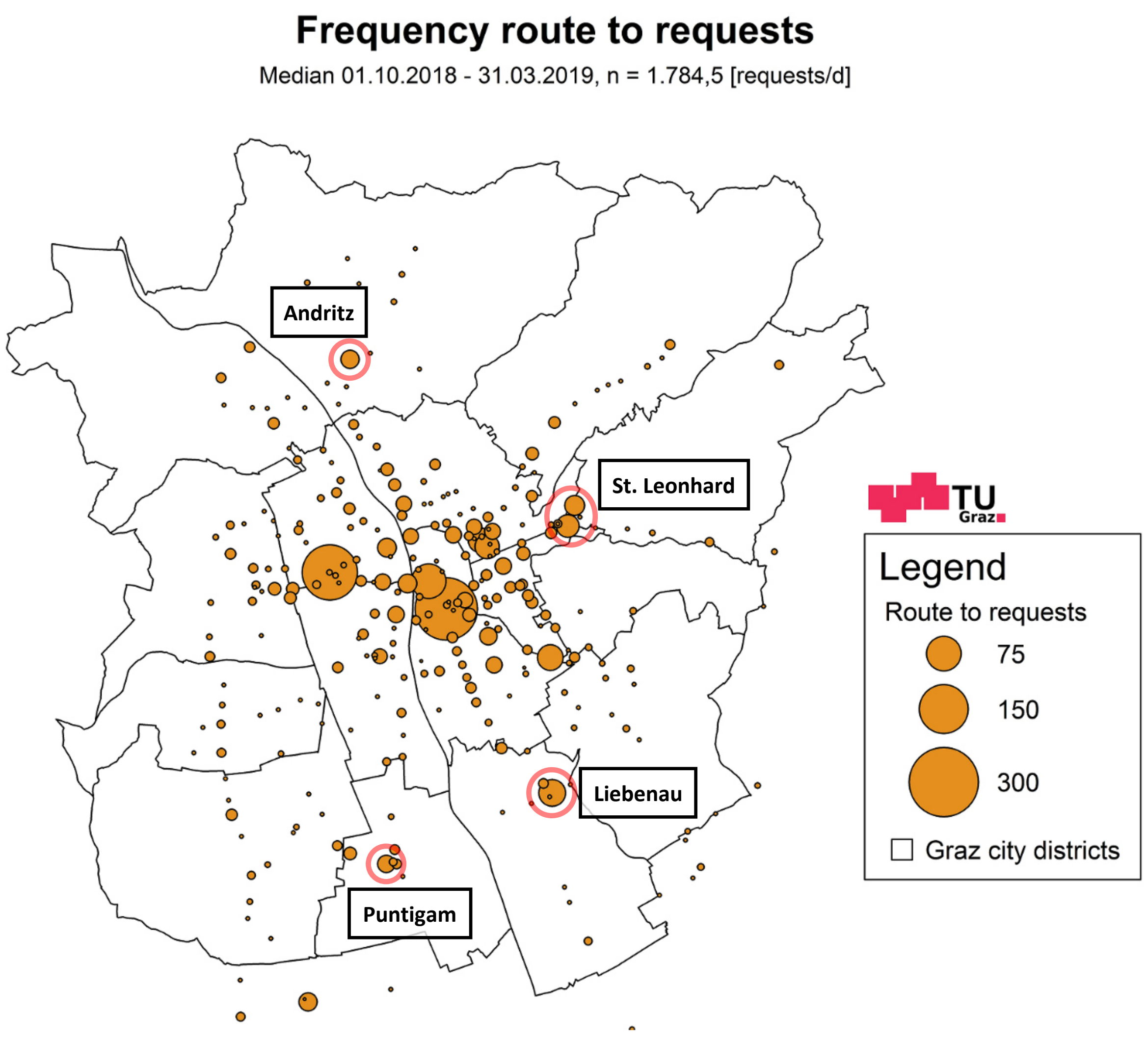


**Figure 47: Frequency route to requests, daily values, median six months**

As can be seen in Figure 48, Jakominiplatz received the most route to requests with 243,5 requests per day. Then comes the main railway station with 191 and the area around KFU with 116,5 requests per day. Table 20 below lists the exact values of the most requested stops in requests per day.

**Table 20: Most requested stops with number of requests, median six months**

| Stop | Requests per day |
|---|---|
| Jakominiplatz | 243,5 |
| Main railway station | 191 |
| KFU | 116,5 |
| Main square | 75 |
| TU, school center | 50 |

Table 21 shows a comparison between route from and route to requests for the most requested stops. As with the end stations there is also a big difference between route from and route to requests for the most requested stops. The stops or areas have significantly higher values for route to requests. So it also follows, that at the hotspots a larger number of route requests are ending than starting. Beside the facts that qando users had uniform route ends and used their actual position as route start, it could also be that in many cases the outward journeys were requested, and the return trips were travelled the same way, but therefore no additional request was needed.

**Table 21: Most requested stops with number of requests, median six months**

| Stop | Route from [requests per day] | Route to [requests per day] |
|---|---|---|
| Jakominiplatz | 213 | 243,5 |
| Main railway station | 131 | 191 |
| KFU | 67 | 116,5 |
| Main square | 51,5 | 75 |
| TU, school center | 42,5 | 50 |

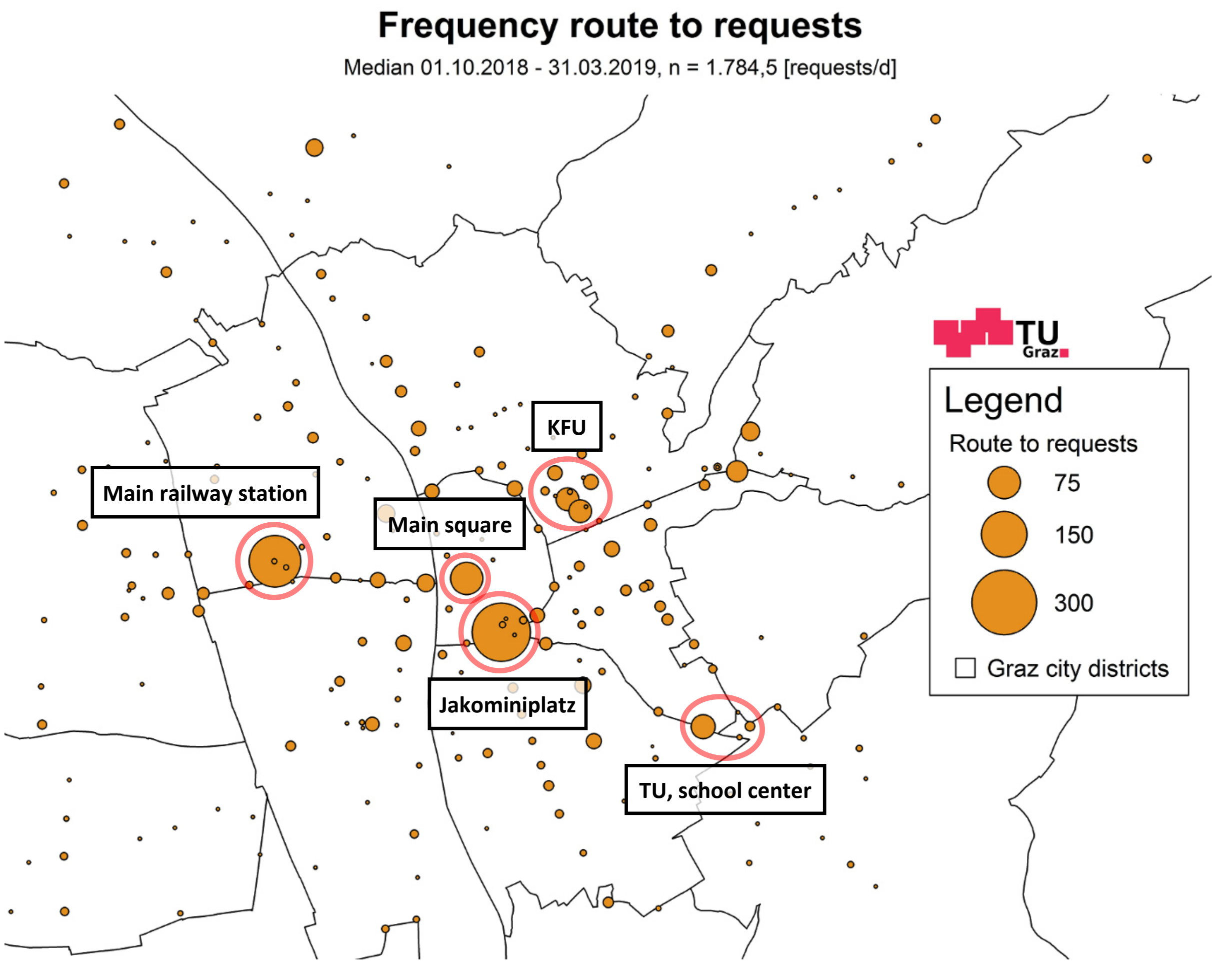


**Figure 48: Frequency route to requests, daily values, median six months, zoomed to inner city**

In the case of the route to requests the Mondays were also the most requested days related to the weekdays. 2.194,5 requests were made on average on these days per day, the route from requests had only 1.753 requests per day. This difference can be seen in Figure 49 at the most requested stops. Especially noteworthy is the area around KFU, where multiple points received many requests, which together make up a significant proportion. There are also points visible where no stops are located. These addresses or positions have been requested regularly and thus have not fallen out at building the median. A reason for this could be that users have used same addresses in the requests or individual users have used stored addresses regularly.

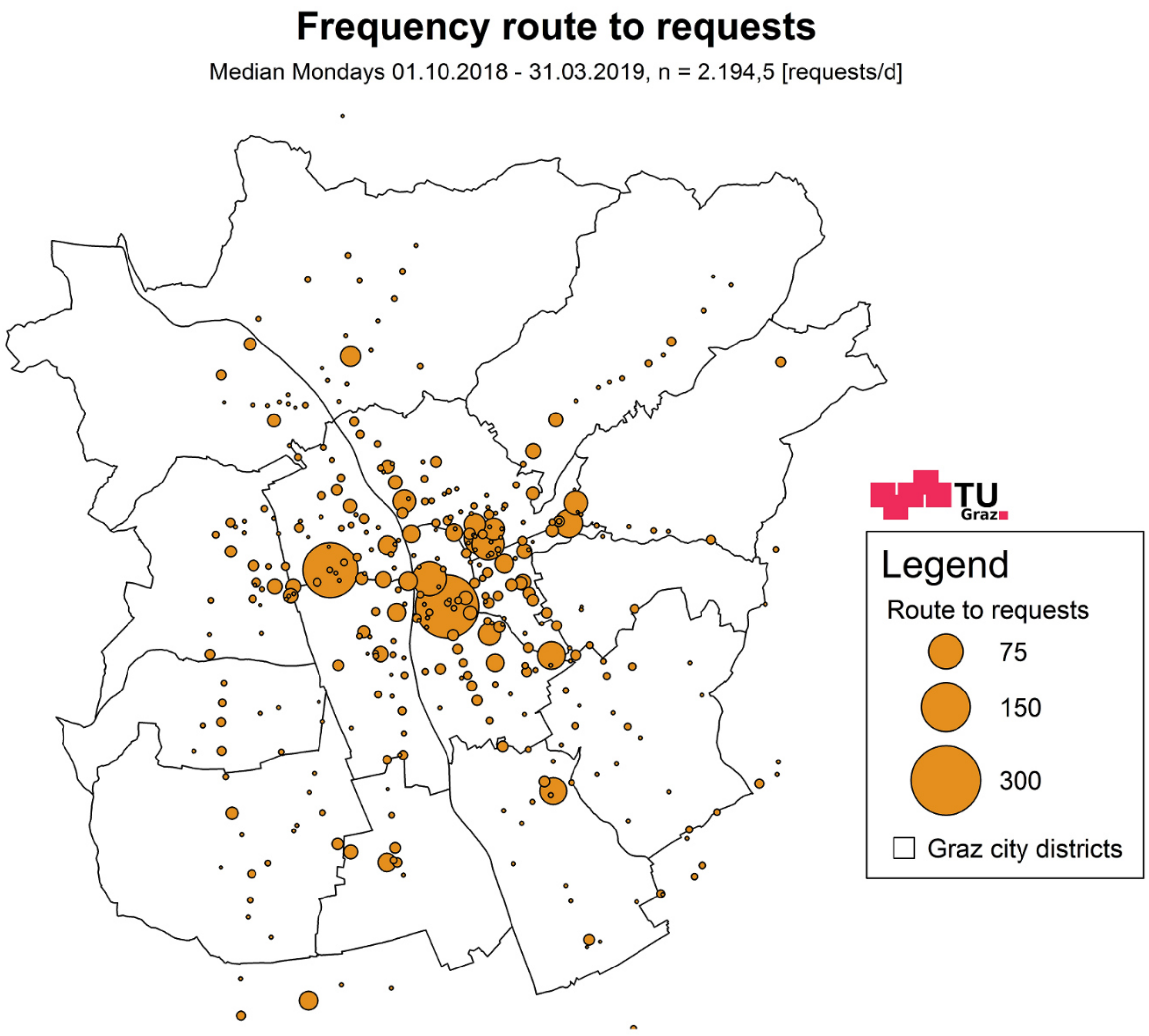

Frequency route to requests
Median Mondays 01.10.2018 - 31.03.2019, n = 2.194,5 [requests/d]
TU Graz
Legend
Route to requests
75
150
300
Graz city districts

**Figure 49: Frequency route to requests, daily values, median Mondays**

Only just over half were requested on Sundays compared to the Mondays, see Figure 50. In total the Sundays came to 1.138 requests per day. A similar picture justifies to the route from requests, but the values are in contrast slightly increased at the route to requests. Jakominiplatz, main railway station and this time also the main square, which was not that much requested at the route from requests, are striking again.

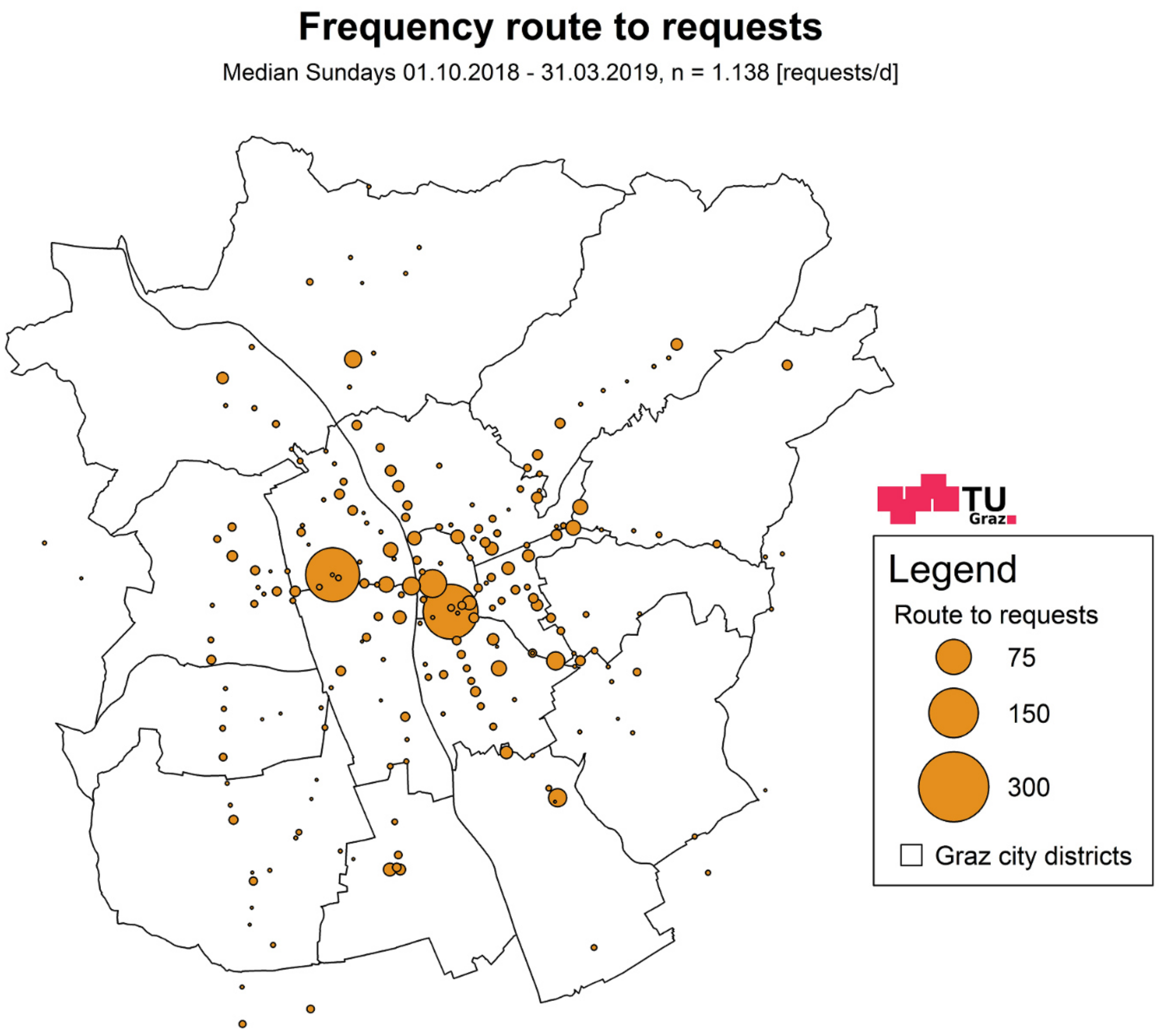

Frequency route to requests
Median Sundays 01.10.2018 - 31.03.2019, n = 1.138 [requests/d]
TU Graz
Legend
Route to requests
75
150
300
Graz city districts

**Figure 50: Frequency route to requests, daily values, median Sundays**

The day with the most requests (10.10.2018) got many requests in the city center at Jakominiplatz, at main railway station and especially a dense distribution in the area of KFU. This area has been the target of many requests and received a lot more than the Jakominiplatz on that day. At the route ends not so many position requests exist, because this route information was usually only used for the route start and thus the stops at the route ends stand out much clearer, because they were specified as destination.

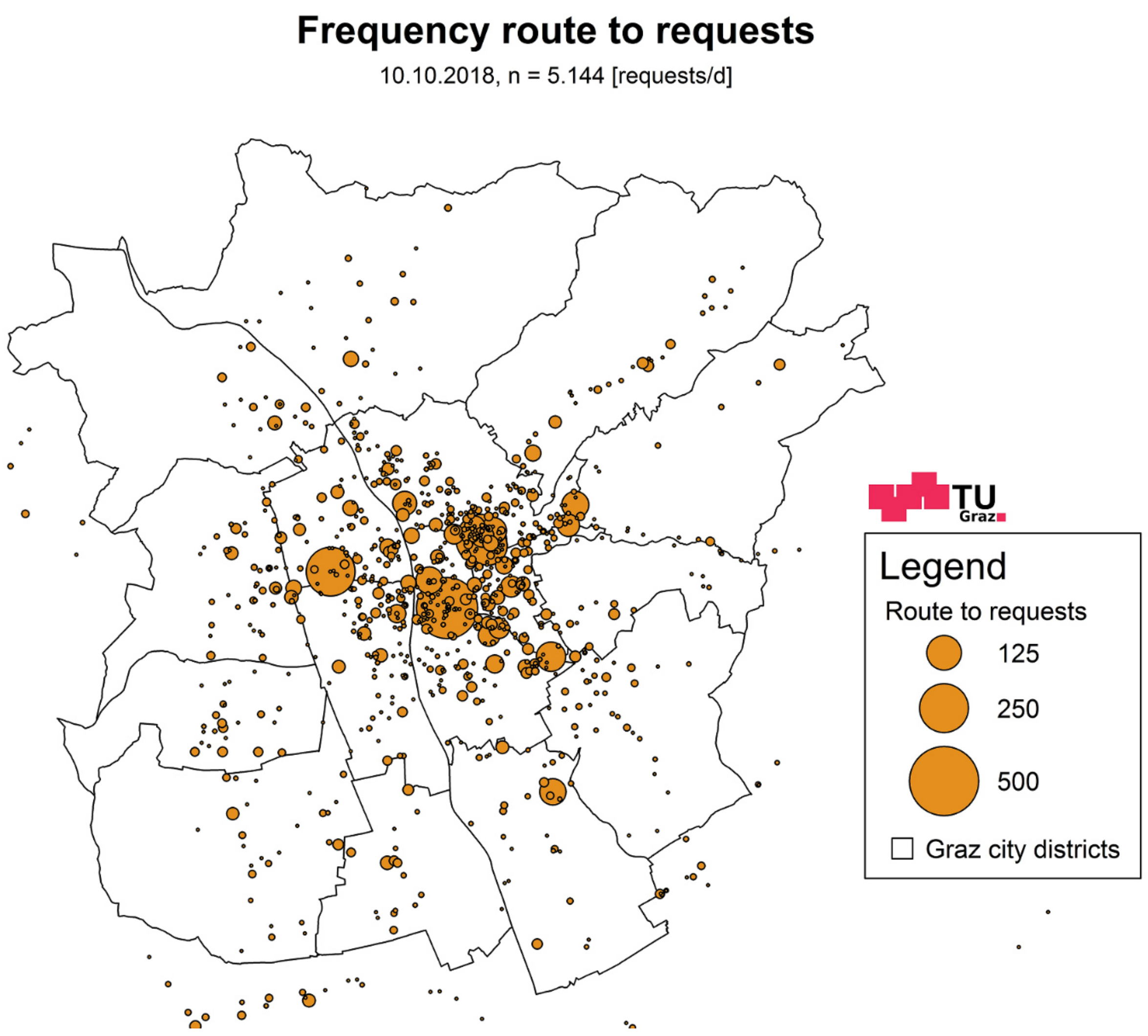

Frequency route to requests
10.10.2018, n = 5.144 [requests/d]
TU Graz
Legend
Route to requests
125
250
500
Graz city districts

**Figure 51: Frequency route to requests, daily values, 10.10.2018, most requested day**

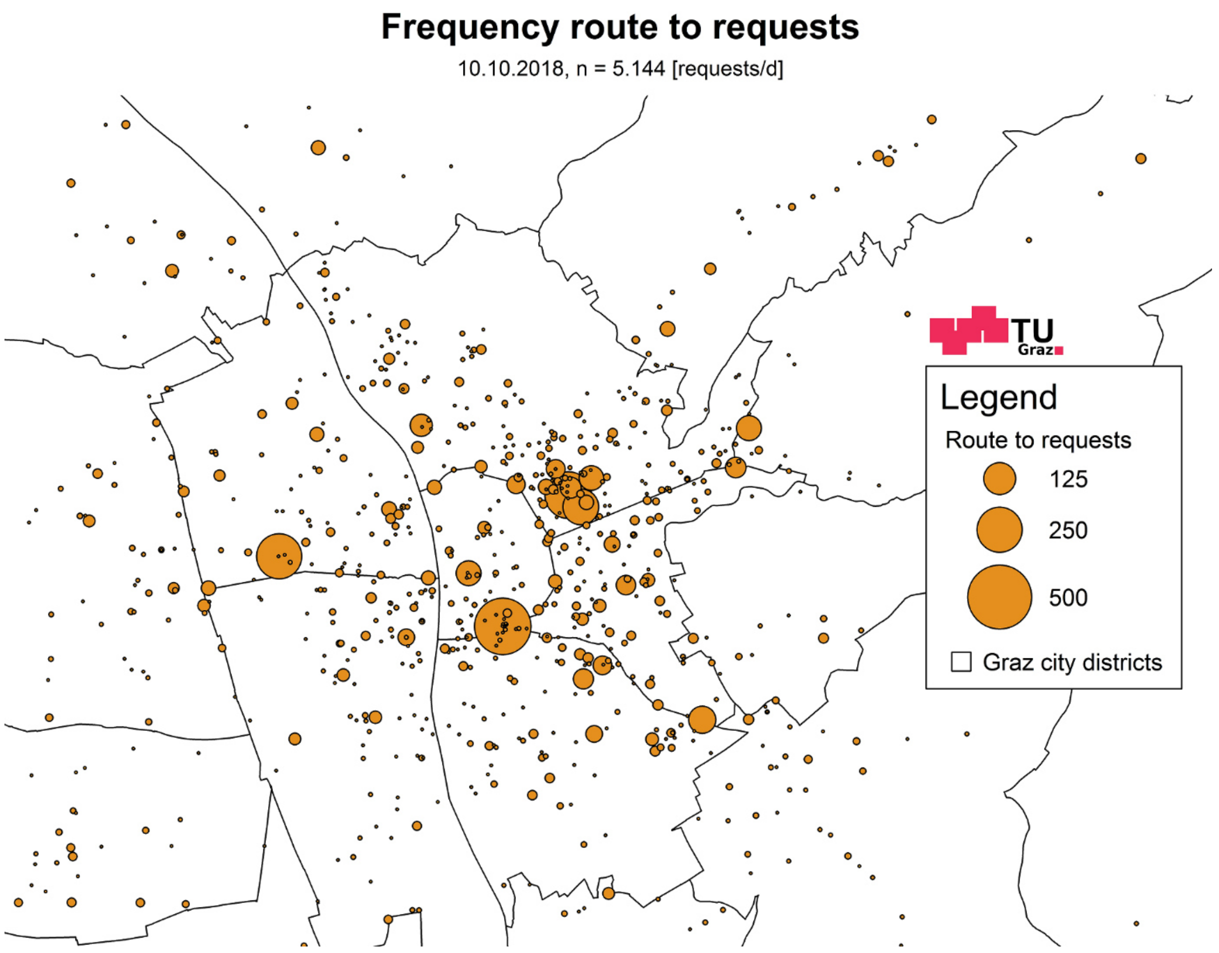

Frequency route to requests
10.10.2018, n = 5.144 [requests/d]
TU Graz
Legend
Route to requests
125
250
500
Graz city districts

**Figure 52: Frequency route to requests, daily values, 10.10.2018, most requested day, zoomed to inner city**

The distribution of this day, shown in Figure 53, is very different from the others in chapter 4.2.2. Alone between one and three o'clock in the afternoon more than 1.000 route requests were sent and also the remaining afternoon is far above the average. In conjunction with the previously seen frequency plots for route from (Figure 45) and route to (Figure 51) requests the very high number of requests on this day and the accumulation of requests in the area of KFU are indicators of an event at the university. Since the 10.10.2018 was the second Wednesday in the new semester it is quite likely that students have celebrated at the university.

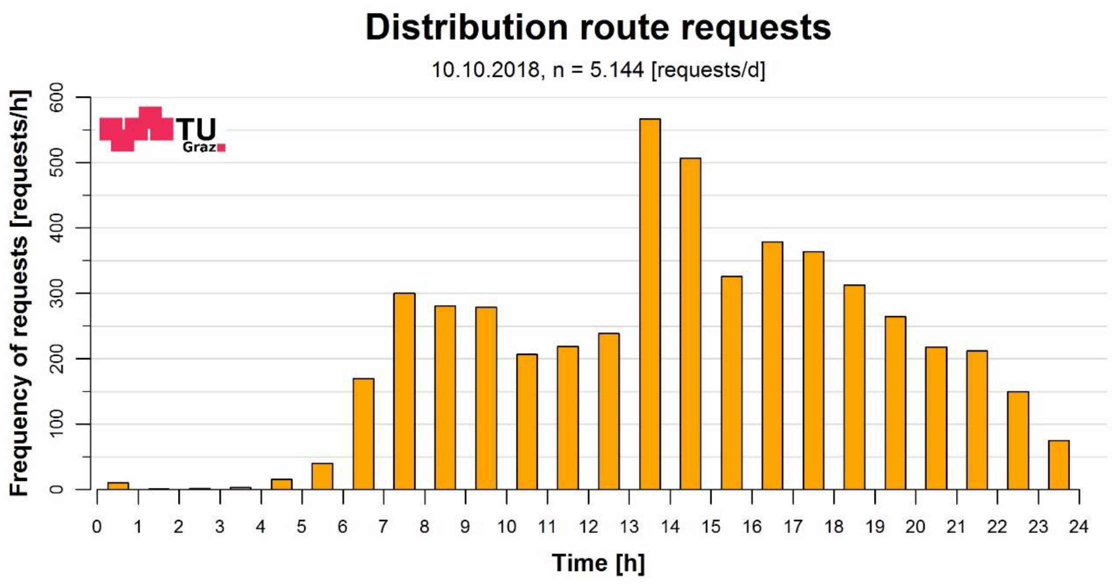


**Figure 53: Distribution route requests, daily courses, 10.10.2018, most requested day**

To give an overview about the hotspots Table 22 shows the medians at hotspots in requests per day. In the first column the total requests per day of all stops are indicated and the maximum values of the requests are marked. Friday is a strong day of travel, which is reflected in the maximum median values of Jakominiplatz and main railway station. Because of the high value on Fridays at the main railway station this indicates a strong home travel over the weekend out of the city or into the city. The main square received its maximum on Saturdays. In the area of the main square many shops and restaurants are located, which make it attractive for leisure activities on Saturdays. Saturdays before Christmas holidays certainly contribute to this higher value.

**Table 22: Medians route to at hotspots in requests per day**

| | Total | Jakomini-platz | Main railway station | Main square | KFU | TU, school center |
|---|---|---|---|---|---|---|
| Median 6 months | 1.784,5 | 243,5 | 191 | 75 | 115,5 | 50 |
| Median autumn | 2.171 | 266 | 216 | 73 | 230 | 58 |
| Median winter | 1.815,5 | 245,5 | 189 | 86,5 | 105 | 54 |
| Median spring | 1.571 | 213 | 184 | 69 | 73 | 34 |
| Median Mondays | 2.194,5 | 250,5 | 189 | 72 | 212 | 55 |
| Median Tuesdays | 2.059,5 | 247 | 177,5 | 70,5 | 220 | 57,5 |
| Median Wednesdays | 2.165,5 | 251 | 186,5 | 75,5 | 206,5 | 67 |
| Median Thursdays | 2.076 | 237,5 | 204 | 69,5 | 188,5 | 66 |
| Median Fridays | 2.186,5 | 269,5 | 295 | 91 | 134 | 52,5 |
| Median Saturdays | 1.549 | 247 | 189 | 95 | 41,5 | 26,5 |
| Median Sundays | 1.138 | 183,5 | 178 | 48 | 24,5 | 27 |

In Table 23 the comparison between medians of route from and route to requests are shown. This table links Table 17 and Table 22, so that the values can be compared. The route from requests are always the lower values with white background and the route to requests are always the upper values with light green background. With one exception the route to requests are always larger than the corresponding route from requests.

Therefore, it can be summarized that there are much more consistent route ends, which the qando users specify when entering the route, as route starts. Also, significantly more position requests were used in route calculation for route starts and thus a large number falls out by building the median.

**Table 23: Comparison between medians of route from (white) and route to (light green) requests at hotspots in requests per day**

| | Total | Jakomini-platz | Main railway station | Main square | KFU | TU, school center |
|---|---|---|---|---|---|---|
| Median 6 months | 1.784,5 | 243,5 | 191 | 75 | 115,5 | 50 |
| | 1.451,5 | 213 | 131 | 51,5 | 67 | 42,5 |
| Median autumn | 2.171 | 266 | 216 | 73 | 230 | 58 |
| | 1.719 | 265 | 155 | 53 | 119 | 48 |
| Median winter | 1.815,5 | 245,5 | 189 | 86,5 | 105 | 54 |
| | 1.469,5 | 212,5 | 123 | 55,5 | 61,5 | 44 |
| Median spring | 1.571 | 213 | 184 | 69 | 73 | 34 |
| | 1.315 | 187 | 125 | 44 | 42 | 35 |
| Median Mondays | 2.194,5 | 250,5 | 189 | 72 | 212 | 55 |
| | 1.753 | 219 | 160,5 | 52 | 99 | 50 |
| Median Tuesdays | 2.059,5 | 247 | 177,5 | 70,5 | 220 | 57,5 |
| | 1.643,5 | 221 | 121 | 54 | 107 | 53,5 |
| Median Wednesdays | 2.165,5 | 251 | 186,5 | 75,5 | 206,5 | 67 |
| | 1.730 | 231 | 131,5 | 59 | 117 | 53,5 |
| Median Thursdays | 2.076 | 237,5 | 204 | 69,5 | 188,5 | 66 |
| | 1.680 | 231 | 125 | 56,5 | 117,5 | 51,5 |
| Median Fridays | 2.186,5 | 269,5 | 295 | 91 | 134 | 52,5 |
| | 1.714 | 258 | 139,5 | 57,5 | 82,5 | 45 |
| Median Saturdays | 1.549 | 247 | 189 | 95 | 41,5 | 26,5 |
| | 1.192,5 | 182,5 | 112,5 | 50 | 26 | 26 |
| Median Sundays | 1.138 | 183,5 | 178 | 48 | 24,5 | 27 |
| | 902 | 160,5 | 143 | 30 | 13 | 19 |

### 4.2.4 Origin-Destination information via spider matrices

So far, the route evaluations had only the information from where they were started or stopped and with which frequency the individual points were requested. The missing information is how the routes went. Using spider matrices, the relationship between start and end point of the route requests can be displayed clearly. Start and end points are assigned to the respective traffic zone in which they fall and routes between the same traffic zones are summed up. The frequency between two traffic zones is considered linearly using the line thickness.

Because the city districts would be too large for these representations of spider matrices the traffic zones of Graz are used. With the help of these zones, the city is divided into 290 districts, which are finer in the inner city and coarser the further it goes out. Thus, the city of Graz is divided into equivalent zones. The zones are received from the traffic model of Graz and are normally used for traffic modeling. They contain information about residents and workplaces and can be utilized to deduce the produced and attracted traffic of the zone. This traffic is distributed among the other zones, the means of transport are selected and then assigned to the lines. Figure 54 shows the traffic zones and additionally the population density in the traffic zones of Graz. The data for the representation comes from the traffic model of Graz. Due to the large size of the traffic zones on the outer edge of the city, the population density is lower there. In the zones in the inner city the residents live together in a narrow space and therefore the population density is greater there. The values are given in residents per hectare.

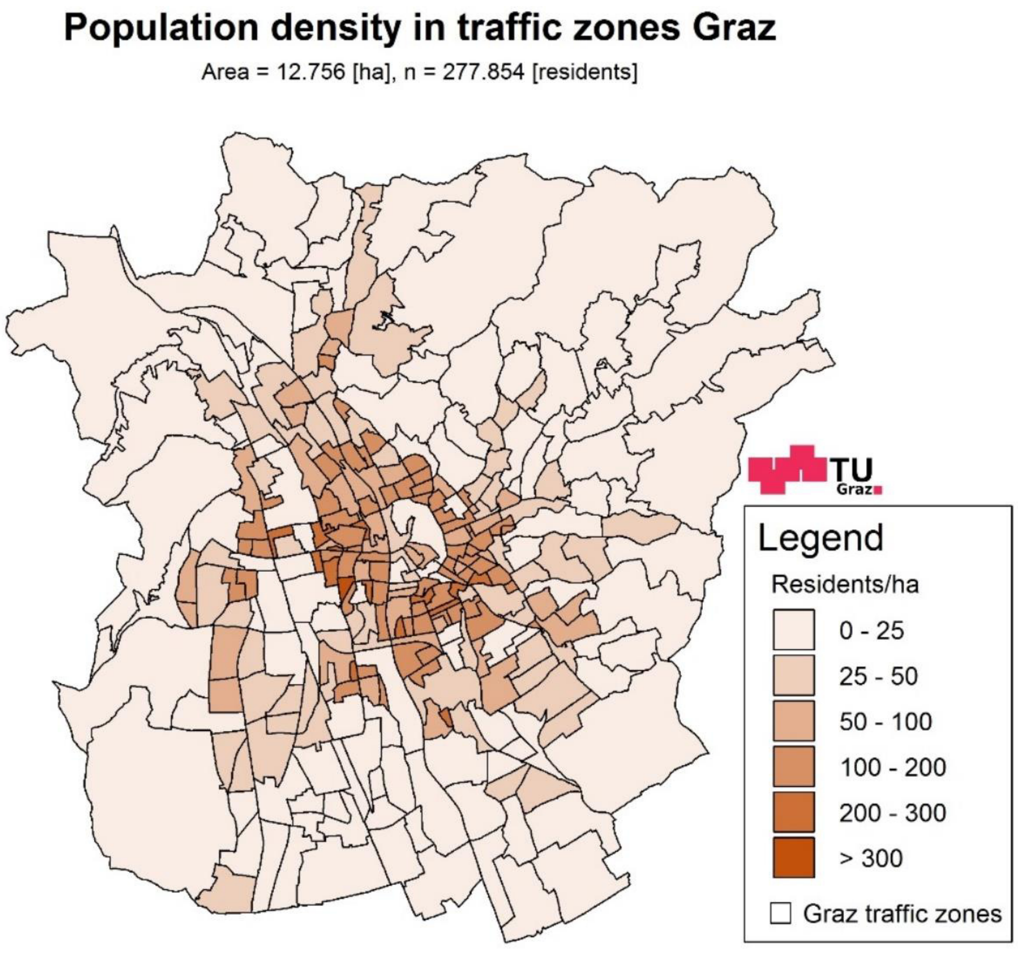


**Figure 54: Population density in traffic zones Graz**

The inner areas were more requested in the previous evaluations than the outer ones. As can be seen in Figure 54, because of the narrow space in which the people live in the inner city, a higher number of requests was already assumed in the city center. Interesting are the traffic zones around the main railway station, KFU and Jakominiplatz. These have only a small number of residents living there.

Areas with many route requests are easier to recognize with this classification of the traffic zones. In the spider matrices plots the traffic zones are shown in the background with black lines and the routes are displayed in blue. The fewer requests a traffic zone to traffic zone relationship has become, the thinner and more transparent becomes the line. The directions between two traffic zones are not distinguished, they are added up and treated as one. How the spider matrix table is created is visible in the following Pseudo-Code.

**Pseudo-Code:** Spider matrix table for route requests

```
##-----------------------------------------------------------------------

Read in sorted route table
Write each coordinate that occurs in the route table in a new matrix once
Write this new matrix to .csv
Blend these coordinates of the .csv-file with the traffic zones of Styria in
    a GIS-program and write them back to .csv
Read in .csv-file of blended coordinates
Remove rows without assigned traffic zones (coordinates out of traffic zones)
    in the coordinates table
Add the belonging traffic zones to the coordinates in the sorted route table
Remove rows without assigned traffic zones in the route table
Write each occurring traffic zone to traffic zone relationship in a vector
    once
Create a matrix out of this vector and add the coordinates of the centroids
    of the traffic zones
Create a matrix with zeros including all weekdays and add this at the end of
    the previously created matrix
Go through each line in the route table and increase the belonging entry in
    the matrix by one
Calculate the sum and the medians and add them at the end of the matrix
Calculate the sum under the columns for every weekday, the sum and the medians
Write this table to .csv

##-----------------------------------------------------------------------
```

Figure 55 shows a spider matrix with the median over six months and an average of 773 route requests were made per day. For the spider matrices not only the urban area of Graz is considered, but also the surrounding traffic zones of Styria. By using spider matrices for representation, it can be seen very well between which traffic zones many requests were made. Because of the fine division of the traffic zones in the inner city, the origin of the route requests is clearly visible. With Jakominiplatz, main railway station and KFU three points stand out again, as already in the chapter 4.2.3, and confirm accordingly these evaluations. Most route requests went with an average of 24 requests per day between Jakominiplatz – KFU. This connection is followed by KFU – main railway station with 18 and main railway station – Jakominiplatz with 15 requests per day. These three connections yield together a triangle with the most requests. The remaining traffic zone to traffic zone relationships received much lower numbers of requests.

**Table 24: Medians of most requested routes in requests per day, median six months**

| Route | Requests per day |
|---|---|
| Jakominiplatz – KFU | 24 |
| KFU – Main railway station | 18 |
| Main railway station – Jakominiplatz | 15 |

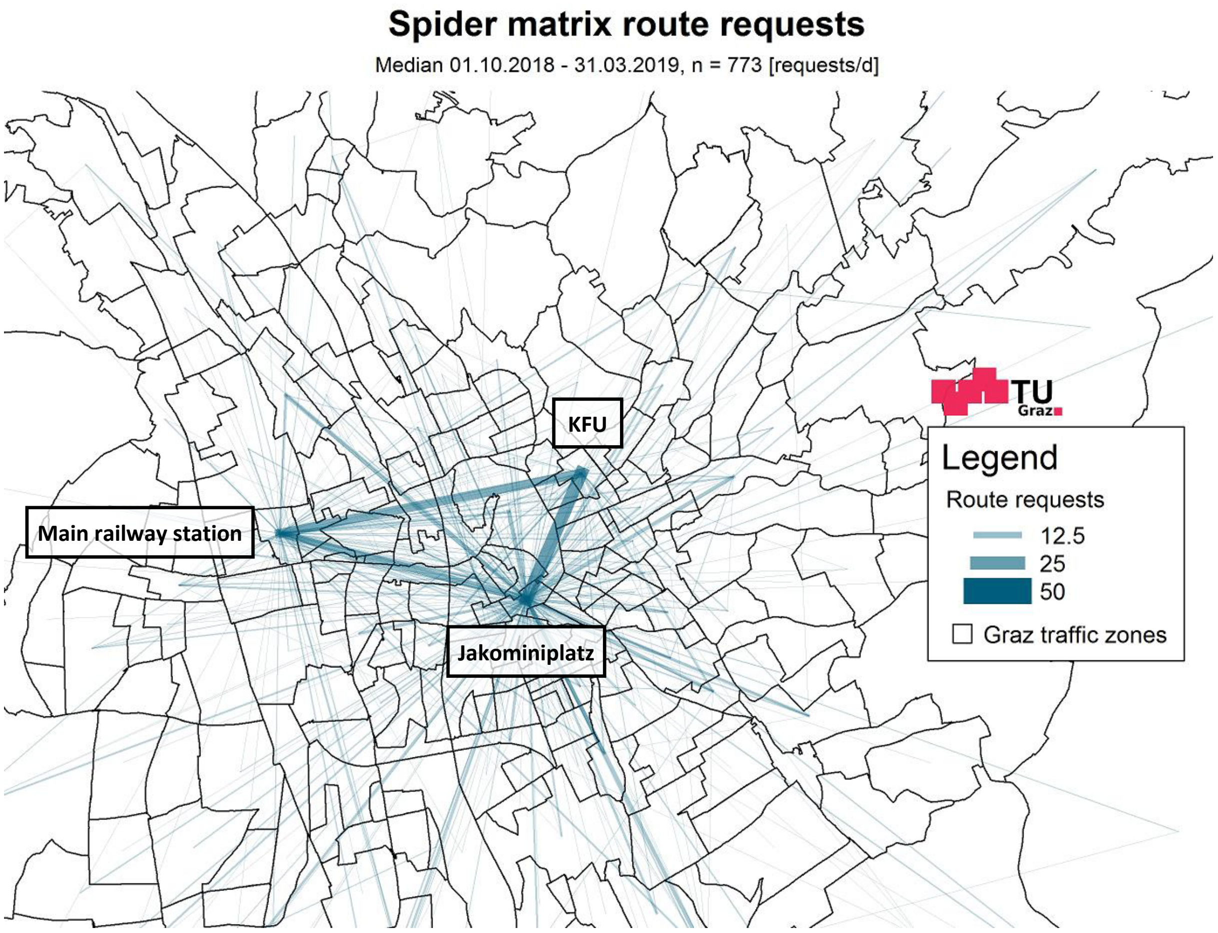


**Figure 55: Spider matrix route requests, daily values, median six months, zoomed to inner city**

On average a total of 1.079 route requests were sent per day in autumn. In the chapter 4.2.2 it was nice to see that most of the route requests were sent in autumn, this can also be seen in the following Figure 56. The connection between Jakominiplatz – KFU received 50 requests per day in autumn and this is also the highest value of all medians. The second most has again the route KFU – main railway station with 37 and then main railway station – Jakominiplatz with 22 requests per day. All other routes are, compared to the three strongest, very low. Jakominiplatz – St. Leonhard is the connection which comes after all to an average of 13 requests per day.

**Table 25: Medians of most requested routes in requests per day, median autumn**

| Route | Requests per day |
|---|---|
| Jakominiplatz – KFU | 50 |
| KFU – Main railway station | 37 |
| Main railway station – Jakominiplatz | 22 |

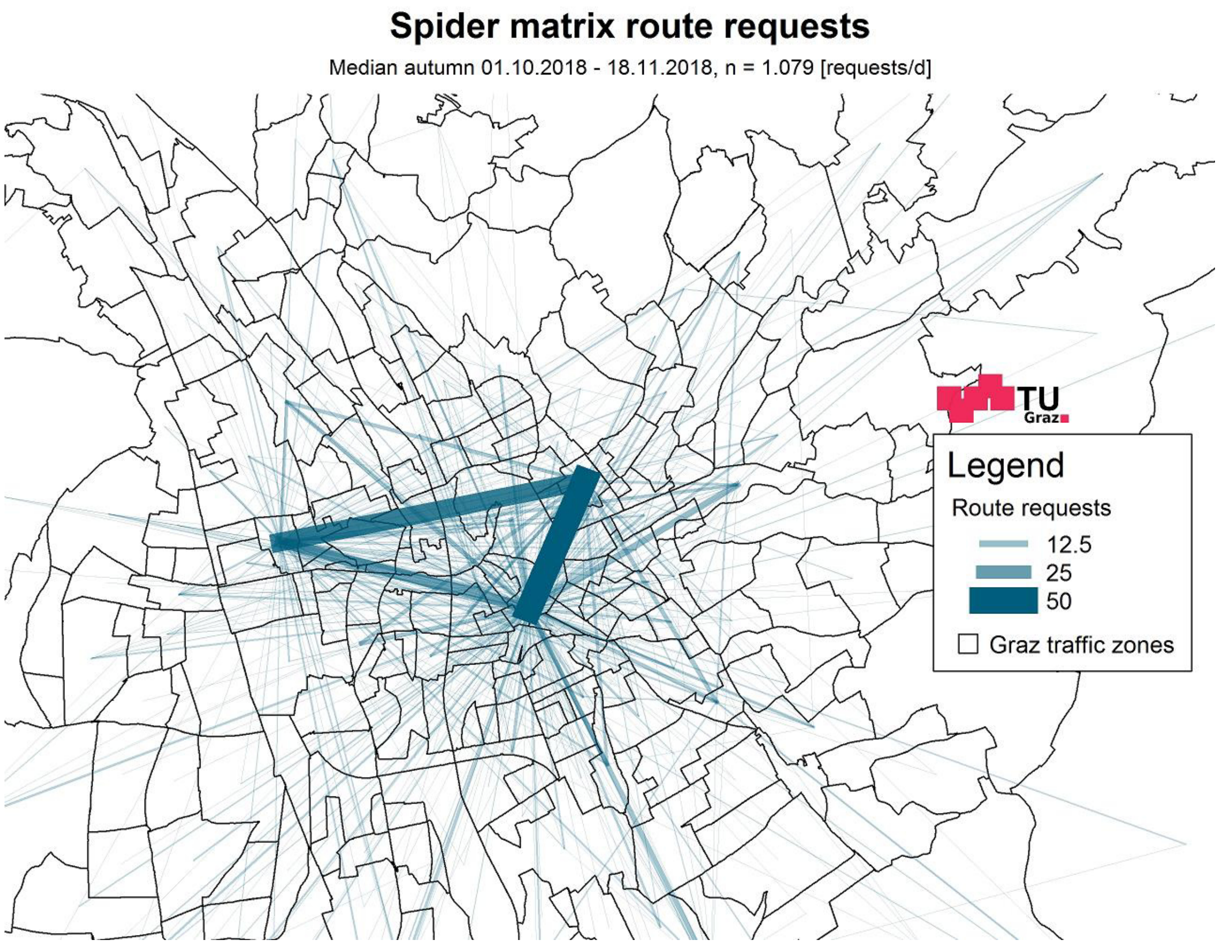

Spider matrix route requests
Median autumn 01.10.2018 - 18.11.2018, n = 1.079 [requests/d]
TU Graz
Legend
Route requests
12.5
25
50
Graz traffic zones

**Figure 56: Spider matrix route requests, daily values, median autumn, zoomed to inner city**

On Fridays, not the connection between Jakominiplatz – KFU had the strongest demand, but that between KFU – main railway station with 33 requests per day. Many students are heading home away from Graz on Fridays and let themselves calculate more routes between these traffic zones. This result is clearly visible in Figure 57. The next connection is then Jakominiplatz – KFU with 27 and subsequent main railway station – Jakominiplatz with 21 route requests per day. The total number of requests on Fridays lays by 1.102 requests per day.

**Table 26: Medians of most requested routes in requests per day, median Fridays**

| Route | Requests per day |
|---|---|
| KFU – Main railway station | 33 |
| Jakominiplatz – KFU | 27 |
| Main railway station – Jakominiplatz | 21 |

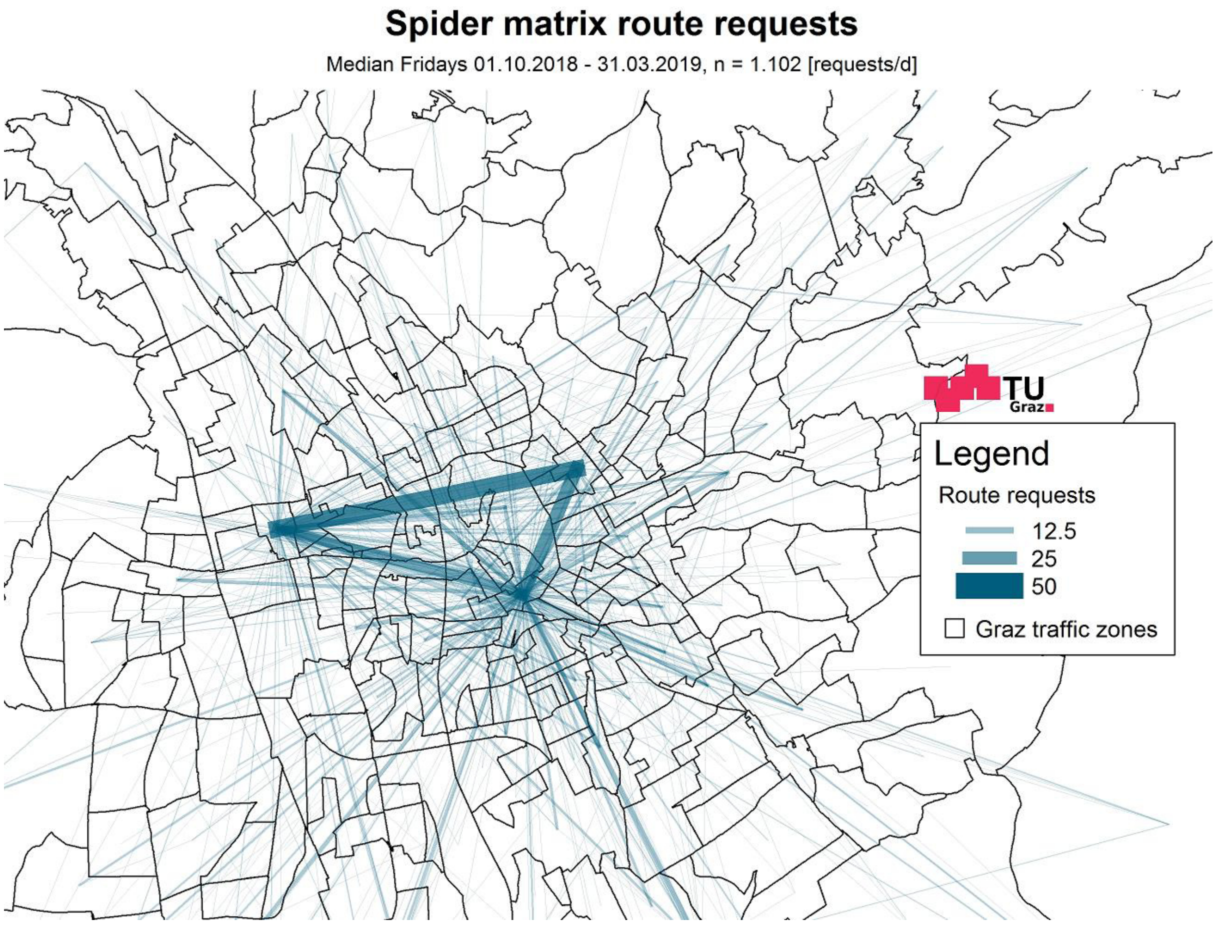

Spider matrix route requests
Median Fridays 01.10.2018 - 31.03.2019, n = 1.102 [requests/d]
TU Graz
Legend
Route requests
12.5
25
50
Graz traffic zones

**Figure 57: Spider matrix route requests, daily values, median Fridays, zoomed to inner city**

The days of the week on which the connection main railway station – Jakominiplatz was requested most are Saturdays and Sundays. Figure 58 shows the spider matrix including the median values for Sundays. It can be clearly seen that the connection between main railway station – Jakominiplatz has the greatest appeal in the qando requests. However, this connection comes only to 21 requests per day, but because of few requests in the remaining city area it stands out blatantly. Sundays are strong travel days and therefore this result is not surprising. The other two connections of the usual triangle were requested similarly rare then the rest of Graz.

**Table 27: Medians of most requested routes in requests per day, median Sundays**

| Route | Requests per day |
|---|---|
| Main railway station – Jakominiplatz | 21 |
| KFU – Main railway station | 8 |
| Jakominiplatz – KFU | 5,5 |

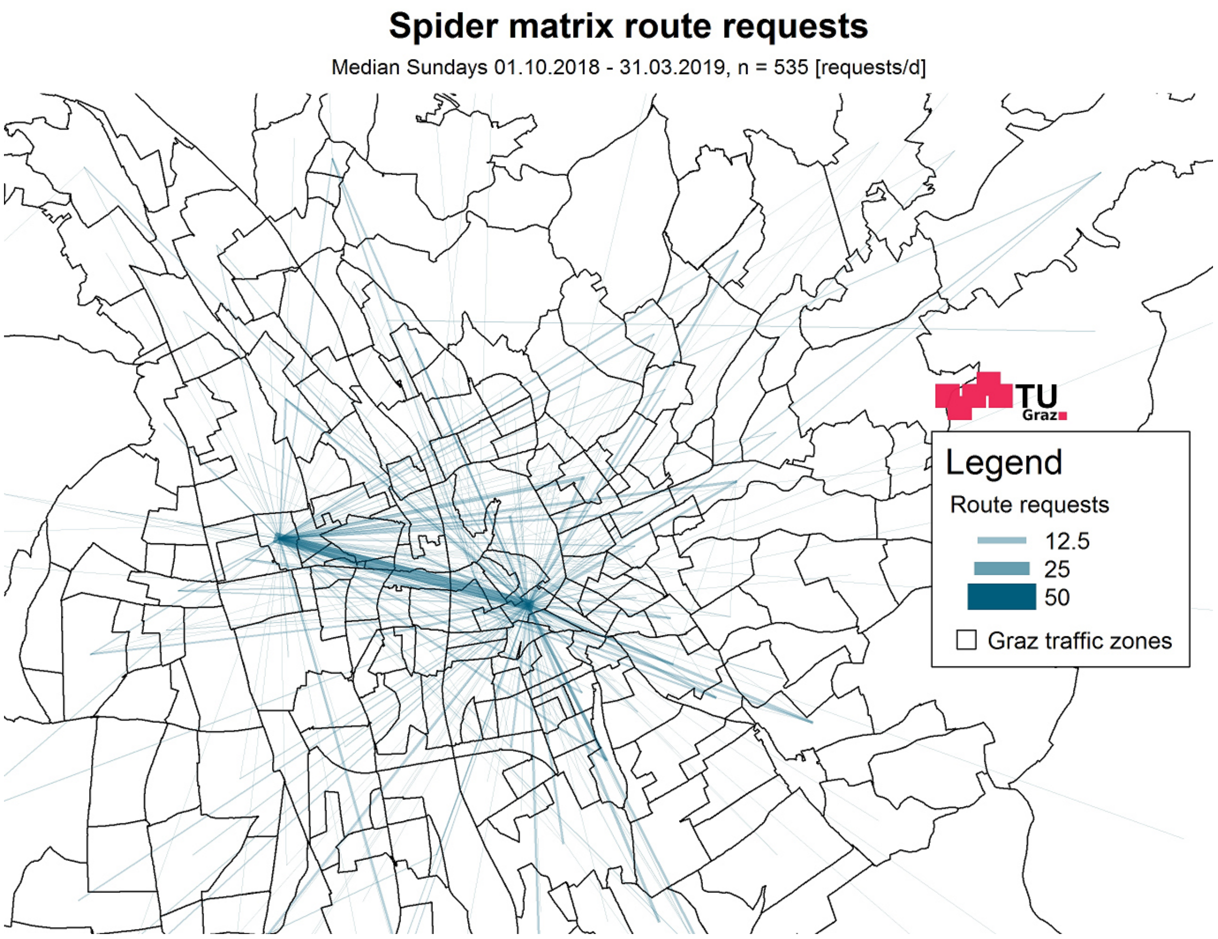

Spider matrix route requests
Median Sundays 01.10.2018 - 31.03.2019, n = 535 [requests/d]
TU Graz
Legend
Route requests
12.5
25
50
Graz traffic zones

**Figure 58: Spider matrix route requests, daily values, median Sundays, zoomed to inner city**

Table 28 lists all medians of the three main connections and compares them, the maximum value for each median is marked. In addition, the total number of requests is indicated. For the most part, the connection Jakominiplatz – KFU was the most requested one. Only on Fridays and on weekends, as previously mentioned, other routes were preferred.

**Table 28: Medians of most requested routes in requests per day**

| | Total | Jakominiplatz – KFU | KFU – Main railway station | Main railway station – Jakominiplatz |
|---|---|---|---|---|
| Median 6 months | 773 | 24 | 18 | 15 |
| Median autumn | 1.079 | 50 | 37 | 22 |
| Median winter | 802,5 | 24 | 16,5 | 12 |
| Median spring | 702 | 19 | 14 | 14 |
| Median Mondays | 1.100,5 | 37 | 26 | 14,5 |
| Median Tuesdays | 1.025 | 43,5 | 20,5 | 12,5 |
| Median Wednesdays | 1.080 | 44 | 22 | 11,5 |
| Median Thursdays | 1.035,5 | 41 | 29,5 | 13 |
| Median Fridays | 1.102 | 27 | 33 | 21 |
| Median Saturdays | 733 | 14 | 8 | 16 |
| Median Sundays | 535 | 5,5 | 8 | 21 |

The 10.10.2018 is by far the most requested day in six months. This can also be seen in the following spider matrix. The connection Jakominiplatz – KFU arrived on that day with 141 requests up to the total maximum in six months. This connection got almost double the number of requests as KFU – main railway station, which came up to 84 requests per day. Between main railway station – Jakominiplatz 50 users wanted a route information to be calculated. Striking on this day is the route between KFU – TU, school center. With 41 requets per day this connection was nearly as often requested as main railway station – Jakominiplatz. A reason for that could be, as already assumed in the chapter 4.2.3, an event in the area around KFU. The remaining route connections also have increased values compared to the other days of the week, but this is not so clear viewable in this representation, because of the high maximum of requests between Jakominiplatz – KFU and the resulting scale. In total, 5.125 route requests were sent on average on that day.

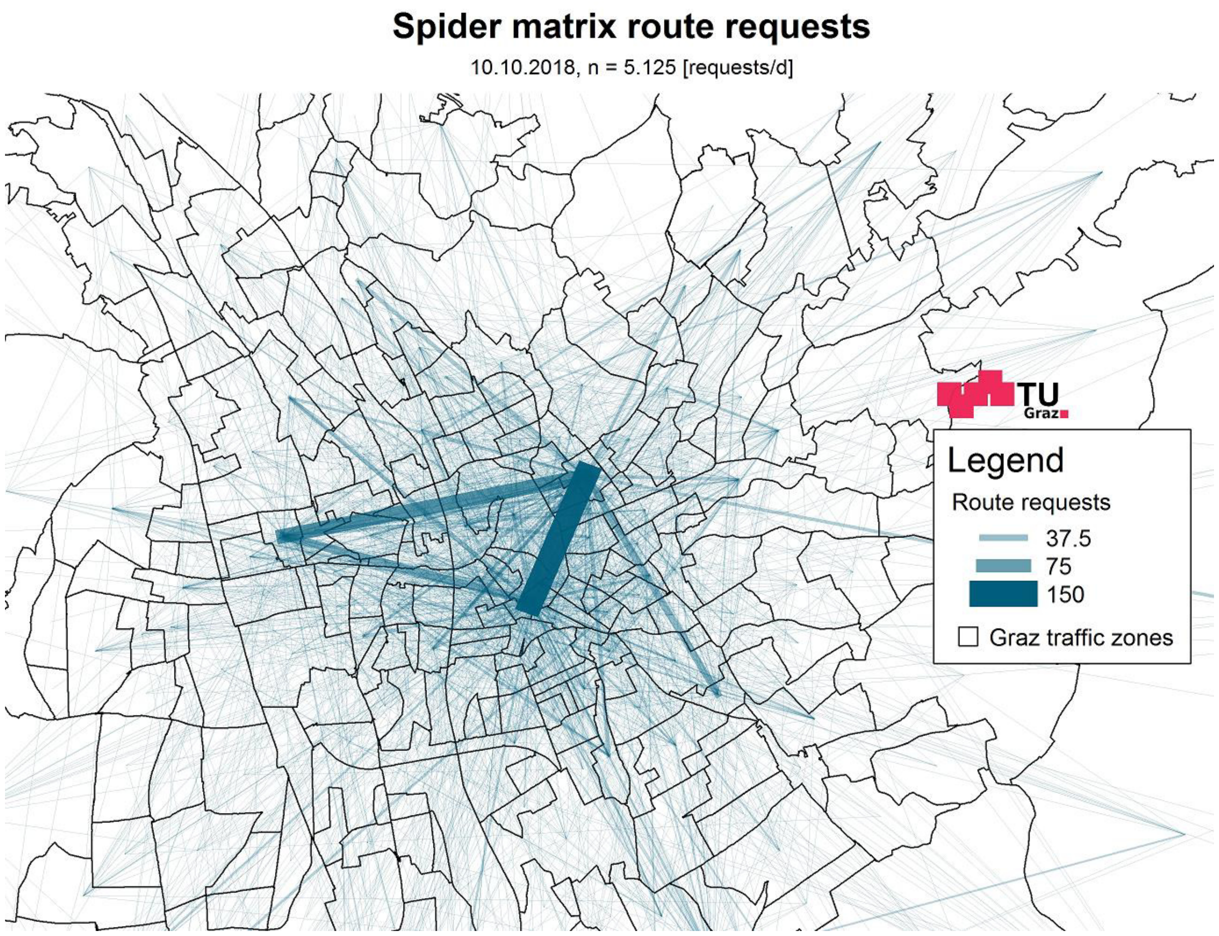

Spider matrix route requests
10.10.2018, n = 5.125 [requests/d]
TU Graz
Legend
Route requests
37.5
75
150
Graz traffic zones

**Figure 59: Spider matrix route requests, daily values, 10.10.2018, most requested day, zoomed to inner city**

The next three figures show routes, which were made from or to one traffic zone. Selected for this purpose are the three most requested traffic zones Jakominiplatz, KFU and main railway station. Figure 60 displays the median over six months for the Jakominiplatz, the central transfer point of Graz. This traffic zone is almost connected to all others, which is indicated by the number of different lines. It is good to see which traffic zones are most closely related to Jakominiplatz, the two thickest lines connect Jakominiplatz with KFU and main railway station. Several routes are also requested to the Kalvariengürtel in the north of the main railway station, to St. Leonhard, to the TU and to Liebenau. Overall, 384,5 routes were requested per day from or to the Jakominiplatz and this place has thus been requested most often.

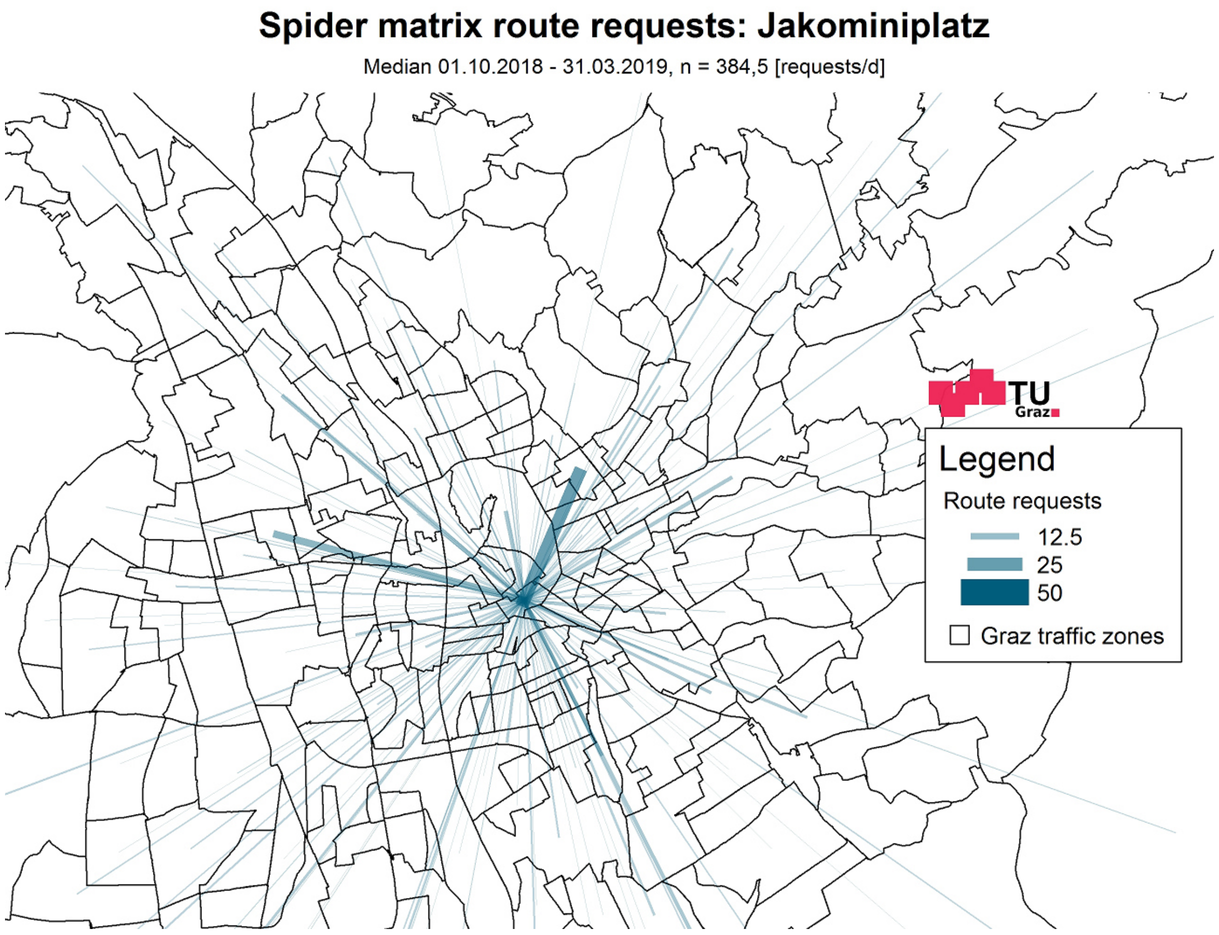

Spider matrix route requests: Jakominiplatz
Median 01.10.2018 - 31.03.2019, n = 384,5 [requests/d]
TU Graz
Legend
Route requests
12.5
25
50
Graz traffic zones

**Figure 60: Spider matrix route requests from/to Jakominiplatz, daily values, median six months**

The Jakominiplatz has a central location in the city and therefore the lines are evenly distributed over the city. Because the main railway station is located very far to the west, most requests lead towards the east in the city center. As expected, the connections to Jakominiplatz and KFU were requested most frequently in the six months and the other routes are rather the same and very low requested in comparison. With 195,5 requests per day, this traffic zone reached about half of the Jakominiplatz. The belonging spider matrix for the main railway station can be seen in Figure 61.

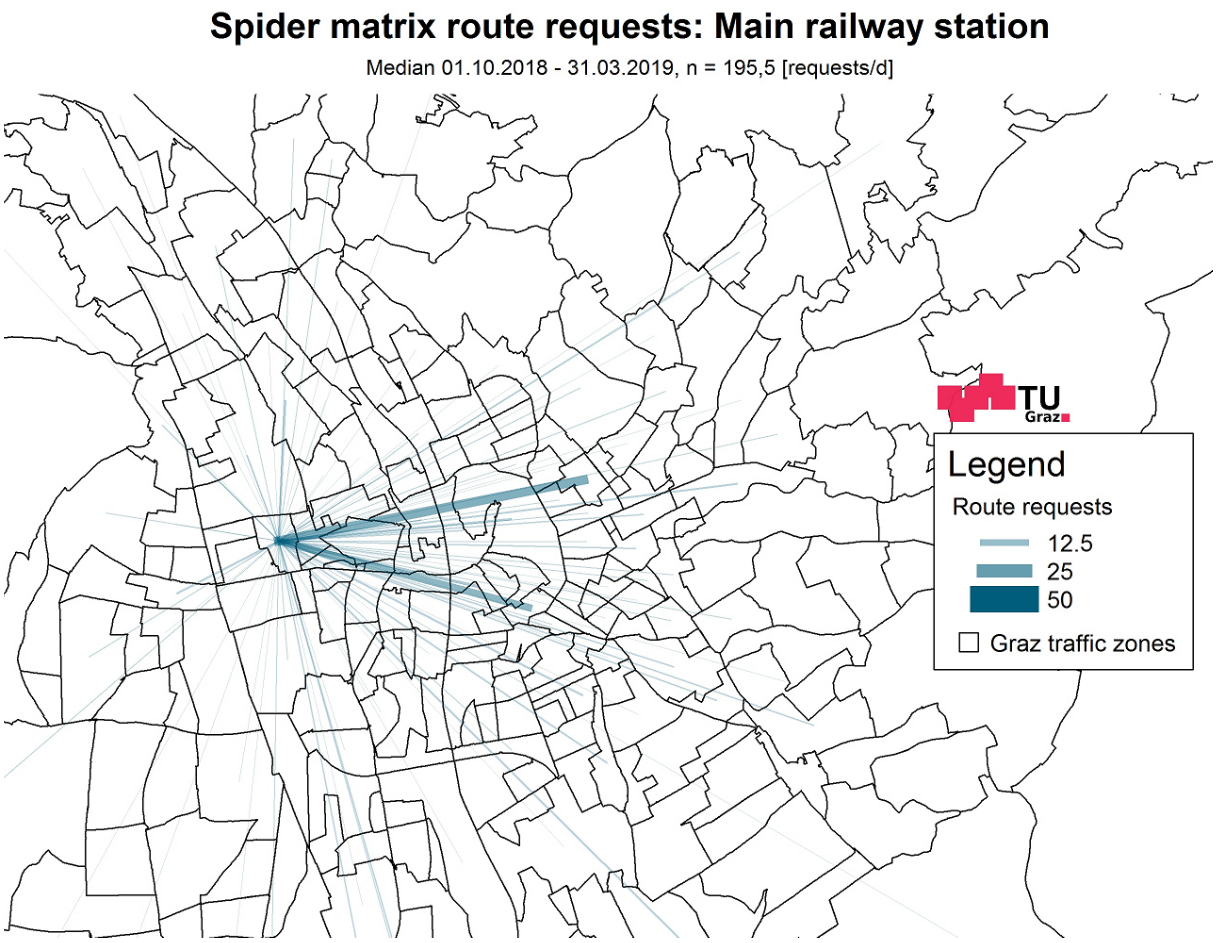

Spider matrix route requests: Main railway station
Median 01.10.2018 - 31.03.2019, n = 195,5 [requests/d]
TU Graz
Legend
Route requests
12.5
25
50
Graz traffic zones

**Figure 61: Spider matrix route requests from/to main railway station, daily values, median six months**

The traffic zone including KFU got with 110 requests per day the lowest number of route requests compared to the other two selected traffic zones. The reason for this is that not so many different routes were requested from or to the KFU, as can be seen in Figure 62. And those who were requested got only a small number of requests. Mainly only to Jakominiplatz and main railway station a strong requested connection exists.

In Table 24 the median values between these three traffic zones are displayed.

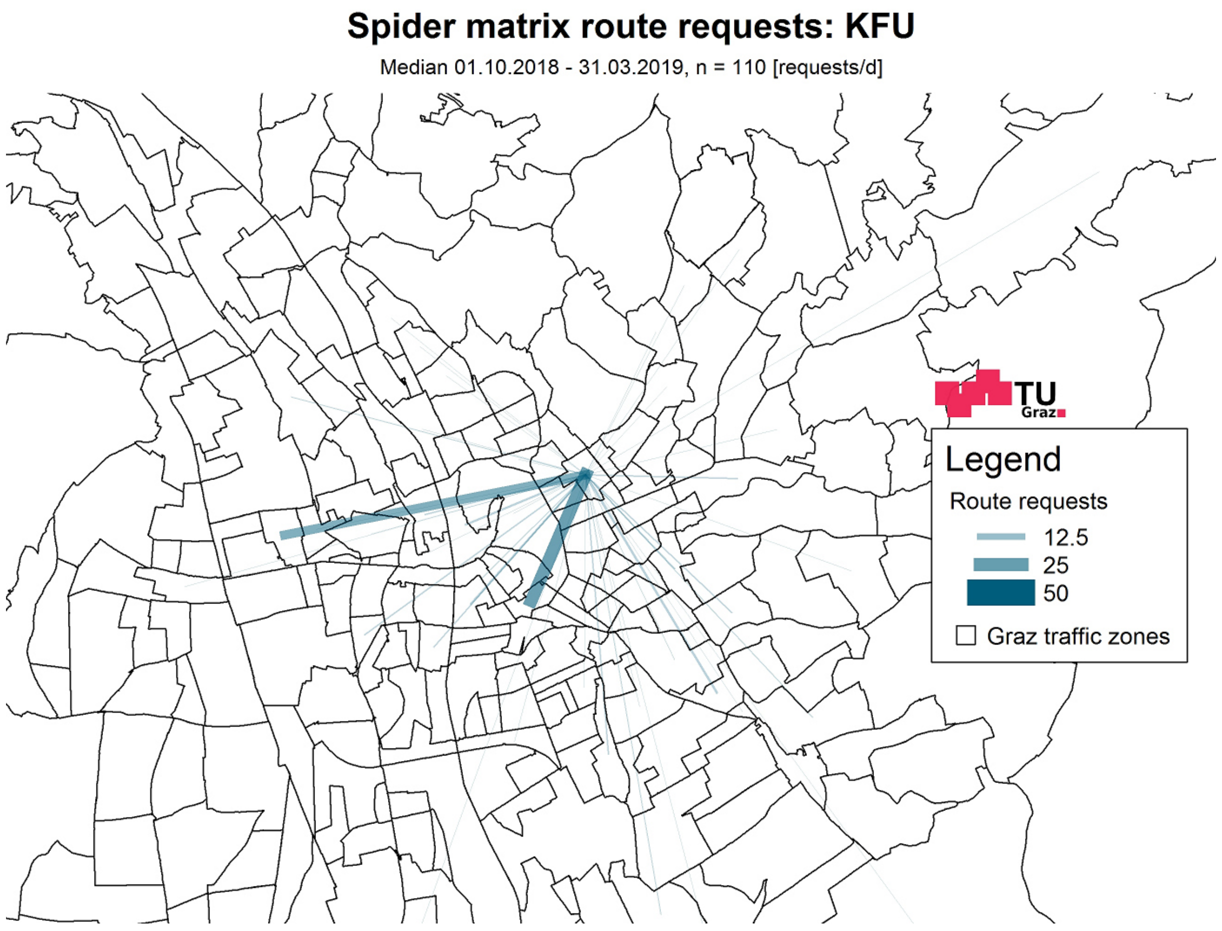

Spider matrix route requests: KFU
Median 01.10.2018 - 31.03.2019, n = 110 [requests/d]
TU Graz
Legend
Route requests
12.5
25
50
☐ Graz traffic zones

**Figure 62: Spider matrix route requests from/to KFU, daily values, median six months**

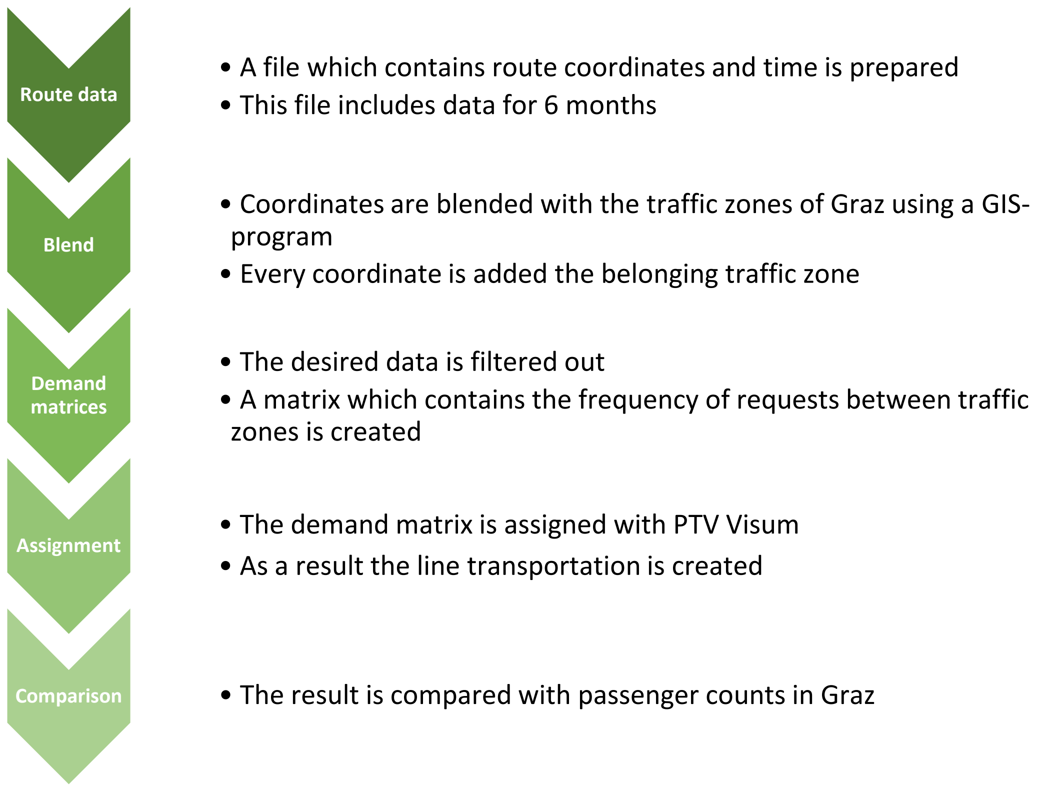
In the analysis of the spider matrices three strong connections have been revealed. The most frequently requested route is between Jakominiplatz – KFU and thus the most attractive one for the qando users. When this connection is compared with the actual occurring passengers, these results do not match. For the Holding Graz, this connection does not have as much importance or traffic volume like other routes in the city. Of course, the results depend heavily on who is using the app. The evaluations of the spider matrices strongly suggest that the majority of the users are students, because the traffic zone around the KFU is very significant. This statement is only an assumption based on the results obtained, because there are no personal data about the users, like age, place of residence, workplace, etc., available. An investigation would be very interesting, but the extend would be too large for this work.

What in addition can be recognized out of the spider matrices is that there were only a few route requests outside of Graz, to Graz or away from Graz. As a result, there are only a few commuters or long-distance travelers who requested routes via the app. Qando is a route planner for the whole state Styria, but it is mainly used in Graz.

### 4.2.5 Public transport assignments

With all evaluations that have been shown so far (distribution plots, frequency plots and spider matrices) only the qando data has been considered and compared with each other. By now no indication about the correlation between the qando data and the actual occurring numbers of passengers in public transport exist. Therefore, the qando data is assigned to the public transport network, compared with passenger counts from the Holding Graz and the correlation between these two is also checked.

The program that was used to carry out the public transport assignment is called PTV Visum and is provided by PTV Group. PTV Visum is a software for transport planners to support traffic planning in cities. In the fields of traffic planning software PTV Visum is the world's leader. It handles multimodal demand modelling including detailed public transport planning (PTV Group, 2019). Figure 63 shows the flow chart for the implementation of the public transport assignments.



- A file which contains route coordinates and time is prepared
- This file includes data for 6 months

- Coordinates are blended with the traffic zones of Graz using a GIS-program
- Every coordinate is added the belonging traffic zone

- The desired data is filtered out
- A matrix which contains the frequency of requests between traffic zones is created

- The demand matrix is assigned with PTV Visum
- As a result the line transportation is created

- The result is compared with passenger counts in Graz

**Figure 63: Flow chart public transport assignment qando data**

PTV Visum needs demand matrices as input data for the public transport assignment in order to distribute the route requests with corresponding weighting to the lines. These demand matrices contain the information how many routes were requested between the traffic zones. Overall, the demand matrices consider 983 traffic zones, which are considered in the traffic model.

Table 29 illustrates the structure of a demand matrix. In the rows and columns, the traffic zones are arranged in ascending order. In this example are traffic zones from A – D present. The values in the matrix specify how often a route relationship was requested on average on a day. For example, seven routes from C – A and four routes from A – C were requested on a day.

**Table 29: Schematic illustration of a demand matrix**

| TZ | A | B | C | D | … |
|---|---|---|---|---|---|
| A | 5 | 2 | 4 | 8 | … |
| B | 4 | 5 | 1 | 3 | |
| C | 7 | 0 | 6 | 9 | |
| D | 2 | 4 | 1 | 7 | |
| … | … | | | | |

TZ … Traffic zone

The following pseudocode explains the procedure how a demand matrix is created.

```
Pseudo-Code: Demand matrix for route requests

##---------------------------------------------------------------------------

Read in sorted route table
Write route from coordinates and route to coordinates to .csv
Blend these coordinates of the .csv-files with the traffic zones of Styria
   in a GIS-program and write them back to .csv
Read in .csv-files of blended coordinates
Replace coordinates in the sorted route table with the belonging traffic
   zones
Remove rows without assigned traffic zones in the route table
Reduce the route table to those entries used for the assignment
Create a demand matrix for every day out of the route table
Build the median over all demand matrices to get a big demand matrix for the
   desired time span
Add not occurred traffic zones to row names and column names, sort rows and
   columns and fill the added matrix sections with zeros
Write the demand matrix to .csv

##---------------------------------------------------------------------------
```

Due to the low number of qando route requests the demand matrices refer to a whole day. Dividing the demand matrices by the hour results in few or no entries in the matrix and therefore does not make sense. With the help of the belonging hydrograph, the different course of requests over the day is considered. For this, the percentage values per hour of the course are entered in the software before each assignment. The percentage values per hour for the hydrographs of all medians are shown in Table 30.

**Table 30: Percentage values per hour for the hydrographs of all medians**

| | Median | Median autumn | Median winter | Median spring | Median Mondays | Median Tuesdays – Thursdays | Median Fridays | Median weekends |
|---|---|---|---|---|---|---|---|---|
| 1 | 0.39 | 0.40 | 0.42 | 0.33 | 0.26 | 0.28 | 0.22 | 1.86 |
| 2 | 0.16 | 0.17 | 0.16 | 0.15 | 0.12 | 0.11 | 0.09 | 1.15 |
| 3 | 0.10 | 0.11 | 0.07 | 0.07 | 0.03 | 0.05 | 0.04 | 0.66 |
| 4 | 0.10 | 0.11 | 0.10 | 0.11 | 0.06 | 0.05 | 0.10 | 0.31 |
| 5 | 0.46 | 0.34 | 0.46 | 0.52 | 0.49 | 0.33 | 0.35 | 0.81 |
| 6 | 1.38 | 1.35 | 1.44 | 1.47 | 1.66 | 1.30 | 1.24 | 1.67 |
| 7 | 3.75 | 3.27 | 3.67 | 4.19 | 4.81 | 3.73 | 3.51 | 1.76 |
| 8 | 7.67 | 8.11 | 7.30 | 7.76 | 8.91 | 8.00 | 7.29 | 3.77 |
| 9 | 7.02 | 6.93 | 7.02 | 7.14 | 7.65 | 7.19 | 6.33 | 5.41 |
| 10 | 5.64 | 5.70 | 5.42 | 5.85 | 6.13 | 5.74 | 5.15 | 5.64 |
| 11 | 4.90 | 4.90 | 4.87 | 4.78 | 5.00 | 4.67 | 4.69 | 5.28 |
| 12 | 4.93 | 5.13 | 4.87 | 5.11 | 4.97 | 4.84 | 5.32 | 5.40 |
| 13 | 5.39 | 5.04 | 5.31 | 5.41 | 5.06 | 5.12 | 6.21 | 5.66 |
| 14 | 5.51 | 5.93 | 5.65 | 5.63 | 5.61 | 5.38 | 6.42 | 5.92 |
| 15 | 5.78 | 5.99 | 5.80 | 5.52 | 5.71 | 5.60 | 6.59 | 5.95 |
| 16 | 6.52 | 6.76 | 6.56 | 6.40 | 6.33 | 6.53 | 6.12 | 5.74 |
| 17 | 7.36 | 7.82 | 7.45 | 7.10 | 7.54 | 7.97 | 6.76 | 6.18 |
| 18 | 7.41 | 7.25 | 7.64 | 7.06 | 7.40 | 7.80 | 7.09 | 6.96 |
| 19 | 7.15 | 6.82 | 7.20 | 7.06 | 6.70 | 7.12 | 7.23 | 7.16 |
| 20 | 5.60 | 5.76 | 5.68 | 6.07 | 5.03 | 5.73 | 5.77 | 6.59 |
| 21 | 4.21 | 4.10 | 4.18 | 3.97 | 3.63 | 4.13 | 4.40 | 5.06 |
| 22 | 3.37 | 3.07 | 3.46 | 3.09 | 2.96 | 3.34 | 3.16 | 3.90 |
| 23 | 3.22 | 2.95 | 3.20 | 3.38 | 2.62 | 3.11 | 3.13 | 4.08 |
| 24 | 1.97 | 1.98 | 2.07 | 1.80 | 1.33 | 1.87 | 2.80 | 3.09 |

After the assignment the information about how many people used a line per day is received. These values are displayed in a list and in the traffic network for each line that is included in the model. Subsequently figures and comparisons are made with these assigned numbers of passengers.

### 4.2.6 Assignment results

Several public transport assignments were carried out to show the differences between individual time periods or weekdays. For the assignments, demand matrices with median values were used. The first assignment covers the entire time period of the received data and reflects the general average, the further three assignments were divided into autumn, winter and spring and show the seasonal difference in the qando data. Weekdays were distinguished between Mondays, Tuesdays to Thursdays, Fridays and weekends. Because Tuesdays, Wednesdays and Thursdays behave similarly, they were grouped together, and Saturdays and Sundays were also grouped into weekends. Time periods of assignments of the weekdays extend over the whole six months. For the comparison with the passenger counts of the Holding Graz, two further assignments had to be carried out with appropriate conditions. These assignments do not include weekends, only data from Monday to Friday in the specified period. Table 35 shows the data with which the public transport assignments were performed.

**Table 31: Data for which a public transport assignment was carried out**

| |
|---|
| – **Median 01.10.2018 – 31.03.2019** |
| – **Median autumn 01.10.2018 – 18.11.2018**<br>– **Median winter 19.11.2018 – 10.02.2019**<br>– **Median spring 11.02.2019 – 31.03.2019** |
| – **Median Mondays 01.10.2018 – 31.03.2019**<br>– **Median Tuesdays to Thursdays 01.10.2018 – 31.03.2019**<br>– **Median Fridays 01.10.2018 – 31.03.2019**<br>– **Median Weekends 01.10.2018 – 31.03.2019**<br>– **Median Mondays to Fridays 01.10.2018 – 29.11.2018**<br>– **Median Mondays to Fridays 07.01.2019 – 31.03.2019 (except 16.02.2019 – 24.02.2019)** |

#### 4.2.6.1 Median

Due to the low number of route requests, the lines in the assignment results do not have very much passengers per day. Figure 64 shows the distribution of the 15 strongest used lines of the assignment for the median over six months. Tram and bus lines are plotted on the x-axis and the y-axis shows the line transportation in persons per day. As a whole, these 15 lines come to 479 persons who travel on the lines per day. Tram line 7 is the most used line after the assignment, as this line is used by an average of 94 people per day. With these 94 people this line is also clearly ahead of the second line in the diagram, the tram line 1. 64 persons are using this line per day. Tram line 1 and 7 drive a large part along the same rails in the inner city and operate main railway station, Jakominiplatz, old campus of the TU and also pass near KFU. The third most used line is tram line 6 and also operates from main railway station to Jakominiplatz and then further to the two other locations of the TU and the school center. The most frequently used bus line is line 63 from the main railway station via KFU to the TU or the school center and there are 46 passengers per day on this line. Tram line 5 operates between the end stations Andritz and Puntigam Bahnhof, which already stood out in the frequency plots in chapter

4.2.3. Thereafter, the number of passengers on the lines declines and the last line mentioned carries only 9 persons per day. All lines with line transportation can be seen in Table 32.

**Table 32: Lines with number of passengers, median**

| Line | 7 | 1 | 6 | 63 | 5 | 3 | 4 | 31 | 40 | 32 | 39 | 30 | 33 | 64 | 85 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Line transportation [persons/d] | 94 | 64 | 51 | 46 | 44 | 34 | 29 | 26 | 18 | 17 | 14 | 12 | 11 | 10 | 9 |

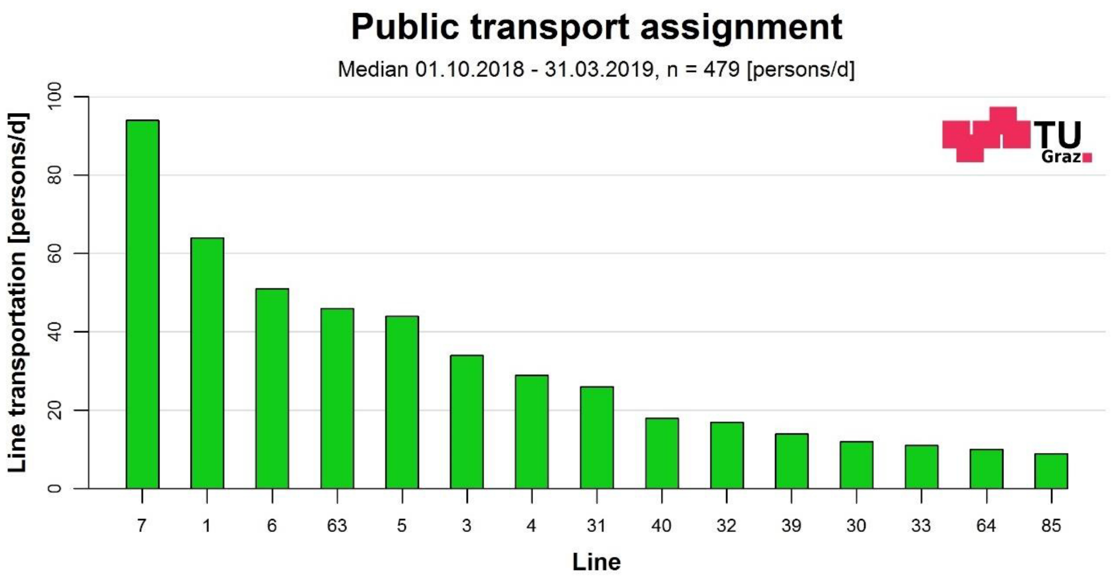


**Figure 64: Public transport assignment, median six months**

Another way of presenting the result of the qando assignment is the plot of the passenger trips per day on the traffic network. In this representation the passengers of all lines are summed up if they travel on the same route. Figure 65 shows pretty much the whole city, the inner part of the city is the most interesting. The background of the map is black, the streets are in grey and the assignment result is displayed in white. Most notably is the route between main railway station – Jakominiplatz. Also, the route from Jakominiplatz towards St. Leonhard has a high number of passenger trips. The routes from Jakominiplatz – TU, school center and main railway station – KFU stand out as well. A small amount of passenger trips received the tram lines 4 and 5, which operate between north and south of Graz.

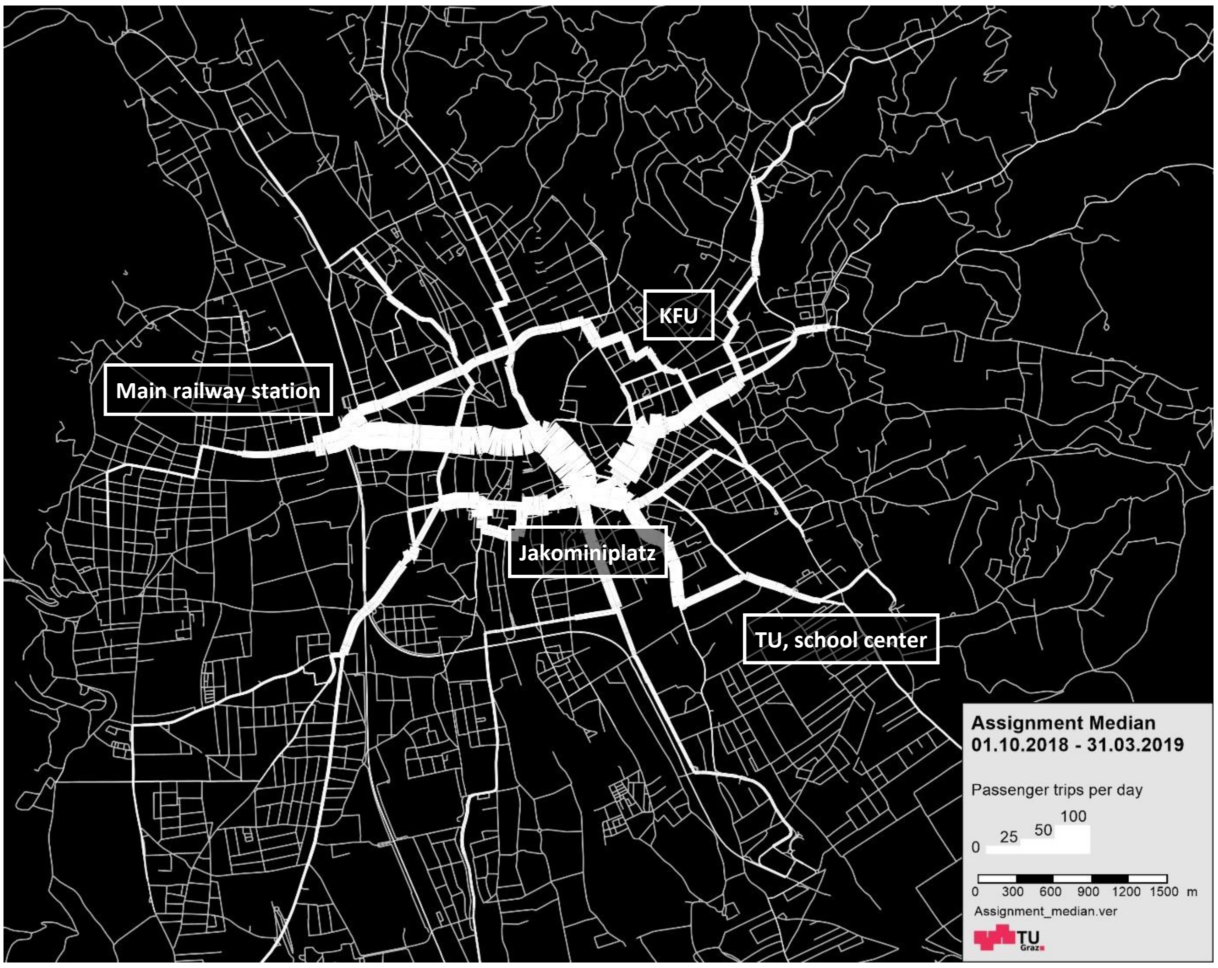


**Figure 65: Public transport assignment, traffic network, median six months**

Public transport assignments are carried out for Graz in order to be able to estimate the travel demand. These assignments fit quite well with reality and therefore the result from the qando assignment can be compared with the result from the public transport assignment. Because the number of passengers per day is very low in the qando assignment, the results are compared relatively. For each of these results the percentage of the total volume was calculated for each link section in Visum and then the differences between the qando and the public transport assignment were computed. These relative differences can be seen in Figure 66. In green an overrepresentation of the qando data is displayed and in red an underrepresentation. As was to see before, the routes between main railway station, KFU and Jakominiplatz are very striking. This also shows up in the comparison with the public transport assignment. It is also good to see, that the route from main square to the north is heavily underrepresented.

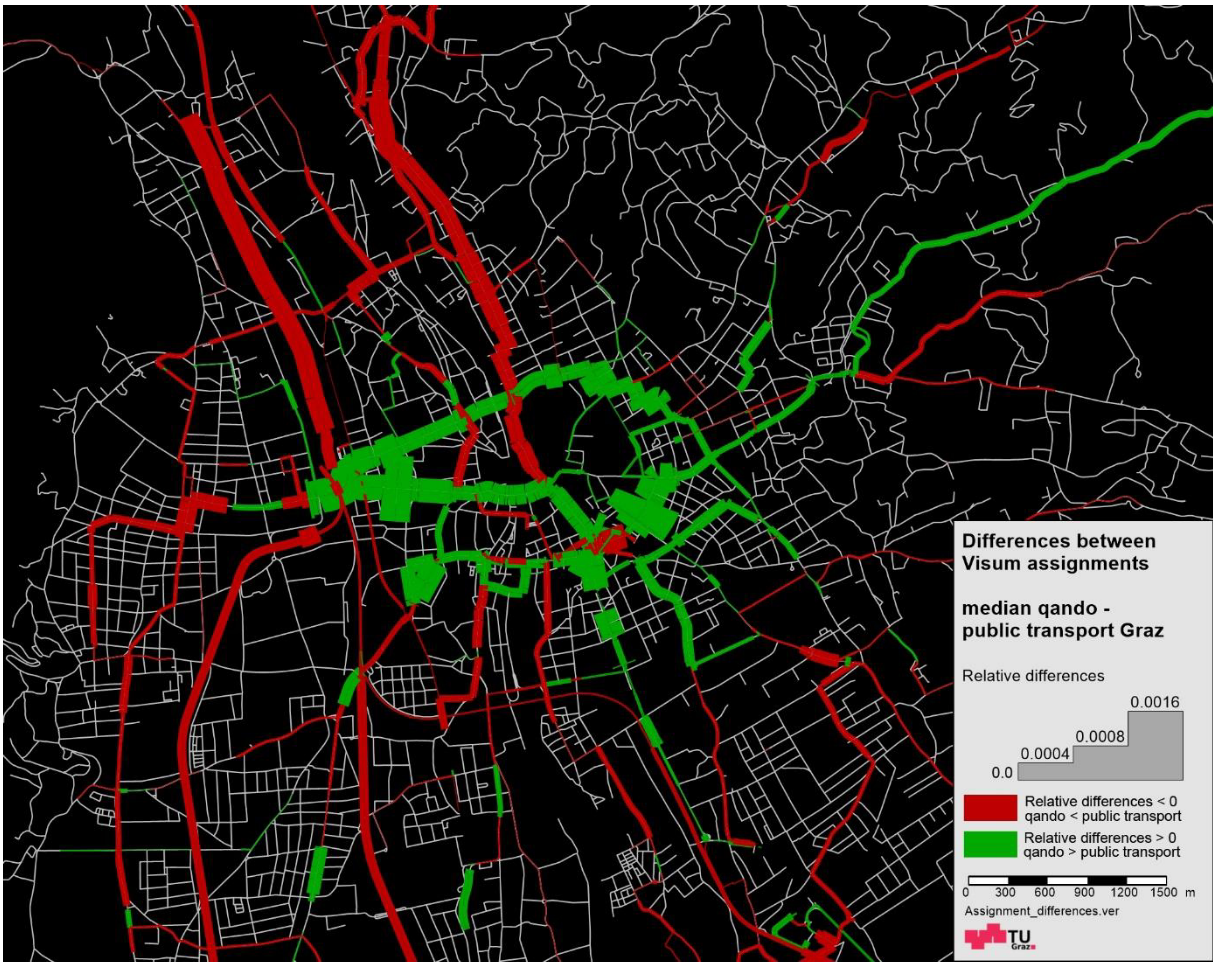

Differences between Visum assignments
median qando - public transport Graz
Relative differences
0.0016
0.0008
0.0004
0.0
Relative differences < 0 qando < public transport
Relative differences > 0 qando > public transport
0 300 600 900 1200 1500 m
Assignment_differences.ver
TU Graz

**Figure 66: Differences between Visum assignments, median qando – public transport Graz**

### 4.2.6.2 Comparison between different medians

In order to get a comparison between the individual qando assignments, different assignments are shown side by side in Figure 67. This figure shows the comparison between the median over six months, the median for autumn, winter and spring and the median for Mondays. On the x-axis the first 15 lines out of the assignment of the median over six months are plotted and based on these lines the other medians are added to the diagram. The y-axis shows the line transportation in persons per day. In dark green, the median over the six months is plotted, the same course as in Figure 64. All other applied medians are also shown in green, but they are getting brighter.

It can be clearly seen that the Mondays have the most passengers on these lines. Especially from the third line to the last one bigger differences between Mondays and the other medians are evident. The Mondays do not include weekends and therefore they come to higher numbers of people per day. Nearest to the weekdays Mondays is autumn. At the first two lines autumn has the most passengers per day and tram line 7 has overall the most passengers in this season with 153 persons per day. In winter, about as many people as in the median over six months use public transport. Only one passenger more is visible in winter compared to the six months at the first 15 lines. The fewest requests are in spring. At the distributions in Figure 35 and Figure 38 it was already viewable that autumn received the most requests compared to the other seasons and then winter and then spring. This is also the case in the assignments. Winter and spring, however, differ only by 50 passengers. In total, this figure shows 2.843 passengers per day.

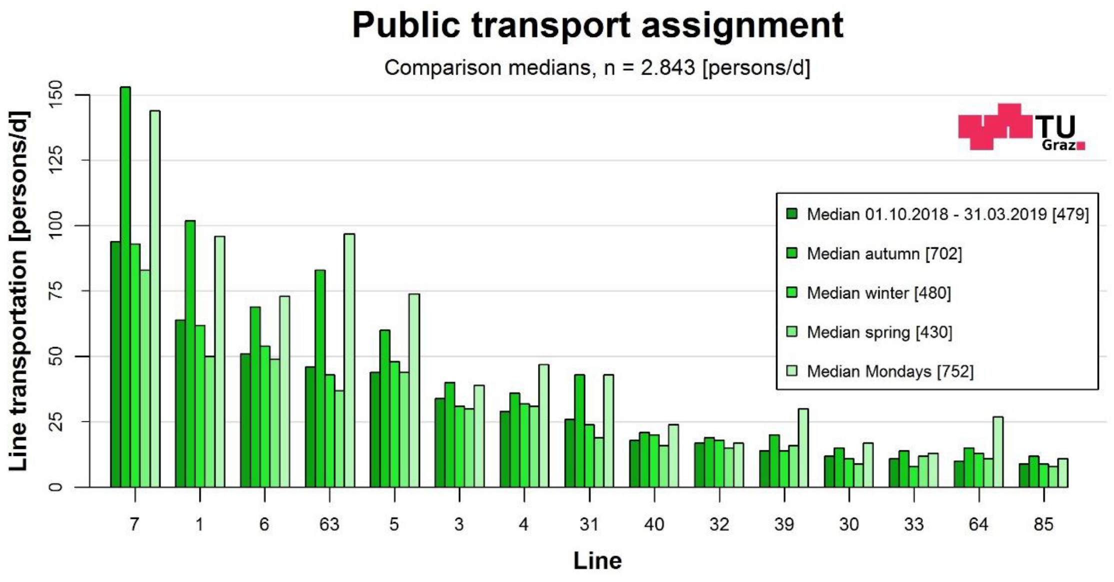


**Figure 67: Public transport assignment, comparison medians**

Figure 68 shows another comparison of medians. In this figure, the weekdays are compared, and the first bar contains again the median over six months. In the previous Figure 67, with the Mondays were already weekdays present, because these days have in general and compared to the other days of the week the most passengers per day. At nearly two-thirds of the lines the Mondays have the most passengers, only the Fridays have more on the other lines. The values from Tuesdays to Thursdays are usually just behind. Significantly fewer people use public transport at the weekends, these days have the lowest value on every line. In total, this figure shows 2.924 passengers per day.

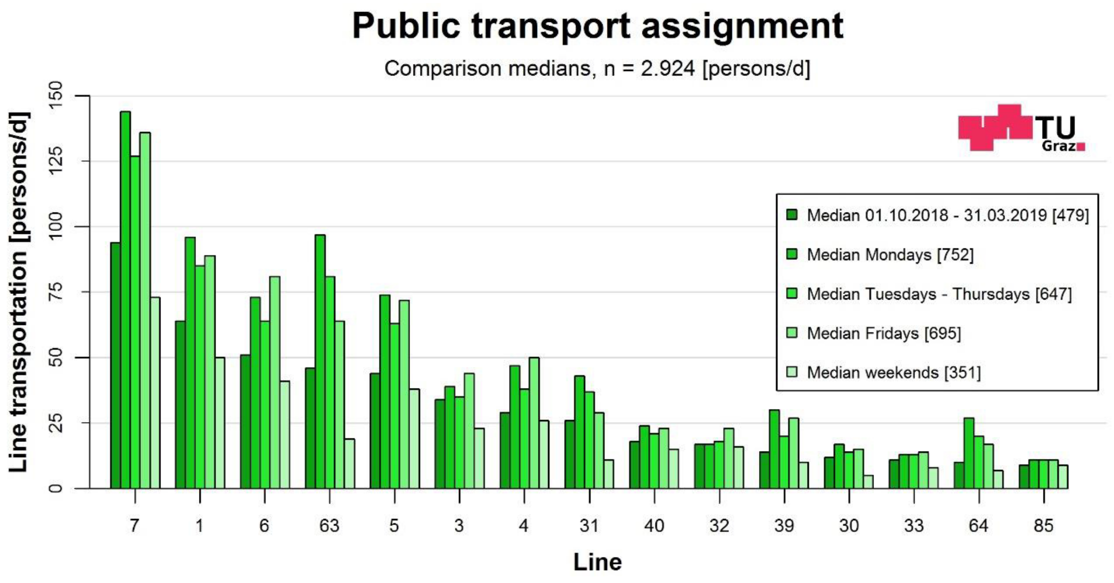


**Figure 68: Public transport assignment, comparison medians, weekdays**

A list of all assignments including the number of passengers per line and day is shown in Table 33.

**Table 33: Lines with number of passengers, all medians**

| | 7 | 1 | 6 | 63 | 5 | 3 | 4 | 31 | 40 | 32 | 39 | 30 | 33 | 64 | 85 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 01.10.2018 – 31.03.2019 | 94 | 64 | 51 | 46 | 44 | 34 | 29 | 26 | 18 | 17 | 14 | 12 | 11 | 10 | 9 |
| Autumn | 153 | 102 | 69 | 83 | 60 | 40 | 36 | 43 | 21 | 19 | 20 | 15 | 14 | 15 | 12 |
| Winter | 93 | 62 | 54 | 43 | 48 | 31 | 32 | 24 | 20 | 18 | 14 | 11 | 8 | 13 | 9 |
| Spring | 83 | 50 | 49 | 37 | 44 | 30 | 31 | 19 | 16 | 15 | 16 | 9 | 12 | 11 | 8 |
| Mondays | 144 | 96 | 73 | 97 | 74 | 39 | 47 | 43 | 24 | 17 | 30 | 17 | 13 | 27 | 11 |
| Tuesdays to Thursdays | 127 | 85 | 64 | 81 | 63 | 35 | 38 | 37 | 21 | 18 | 20 | 14 | 13 | 20 | 11 |
| Fridays | 136 | 89 | 81 | 64 | 72 | 44 | 50 | 29 | 23 | 23 | 27 | 15 | 14 | 17 | 11 |
| Weekends | 73 | 50 | 41 | 19 | 38 | 23 | 26 | 11 | 15 | 16 | 10 | 5 | 8 | 7 | 9 |
| 01.10.2018 – 29.11.2018 | 202 | 134 | 99 | 132 | 86 | 49 | 48 | 61 | 28 | 19 | 35 | 20 | 15 | 29 | 14 |
| 07.01.2019 – 31.03.2019 | 120 | 74 | 67 | 77 | 68 | 36 | 44 | 31 | 20 | 17 | 25 | 12 | 15 | 21 | 10 |

### 4.2.6.3 Comparison qando route assignments – passenger counts Holding Graz

To compare the results of the public transport assignments with the actual occurred passengers, Holding Graz provided passenger counts for autumn 2018 and winter 2019. The definition for autumn and winter is different in the data of the Holding Graz than in this work, therefore, two additional assignments for similar time periods were carried out. Table 34 shows the time periods of the passenger counts of the Holding Graz and those used for the qando assignments. Autumn starts at the passenger counts already on 08.09.2018 and ends on 29.11.2018. qando data is only available from the first of October and so the time period of the qando assignment starts on 01.10.2018 but ends at the same time as the period of the passenger counts on 29.11.2018. Winter begins at the passenger counts on 07.01.2019. This date lays in the middle of the six months of the qando data and is therefore available. However, the winter ends at the passenger counts on 12.04.2019. The qando data is only available until 31.03.2019 and thus nearly two weeks lack compared to the passenger counts. In February a gap of one week exists in the data of the passenger counts and therefore this week is also removed in the qando data and is not considered in the assignment. Only weekdays from Monday to Friday are considered.

**Table 34: Time span of passenger counts Holding Graz and qando assignment for comparison**

| | Passenger counts | qando assignment |
|---|---|---|
| Autumn 2018 | 08.09.2018 – 29.11.2018 | 01.10.2018 – 29.11.2018 |
| Winter 2019 | 07.01.2019 – 12.04.2019 (except 16.02.2019 – 24.02.2019 | 07.01.2019 – 31.03.2019 (except 16.02.2019 – 24.02.2019 |

The passenger counts for the time period autumn 2018 are shown in Figure 69. In descending order, the most used 20 lines in Graz can be seen in this figure. Tram line 7 is the strongest line in Graz with almost 40.000 passengers per day and has significantly more passengers than the other lines. All tram lines are at the beginning of this distribution, only then come the most attractive bus lines. Due to a usually low service rate, central trips within the city and more space in the vehicles than buses, trams transport more passengers than buses do. Lines 40 and 58 are the most used bus lines and the last line listed is the bus line 41 with just over 5.000 passengers per day. In total, these 20 lines carried 283.821 passengers per day.

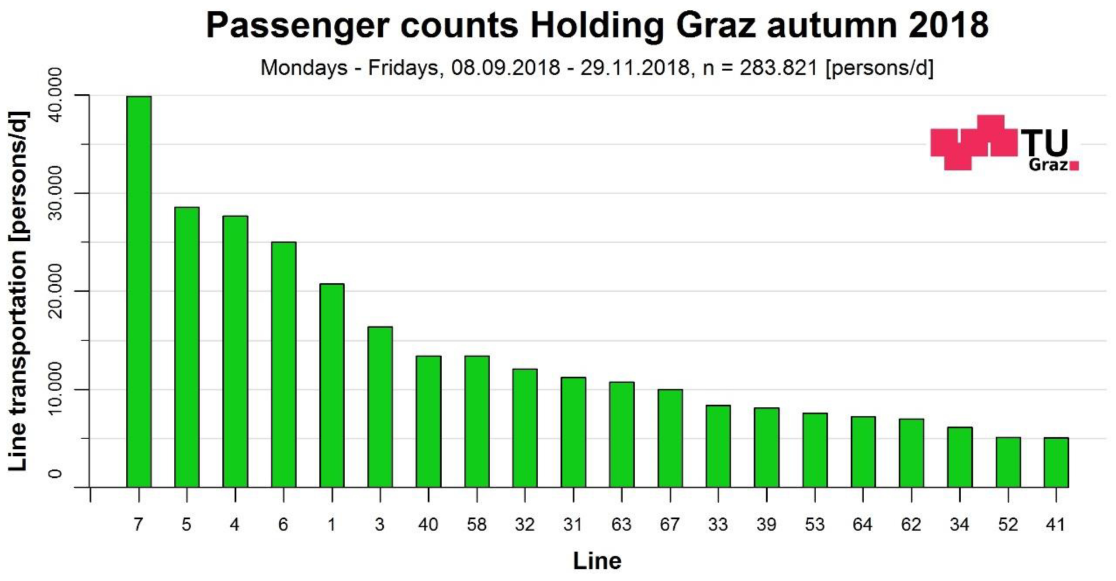


**Figure 69: Passenger counts Holding Graz, autumn 2018**

Due to the fact that the passenger counts of the Holding Graz are on a completely different scale, they are hardly comparable to the values out of the qando assignment. Therefore, the comparison of the data in Figure 70 is shown with percentages. The 20 lines represent 100 % for both distributions and out of these the shares for the individual lines were calculated. Thus, all bars for the qando assignment and for the passenger counts result to 100 % each. The lines on the x-axis contain the top 20 lines of the passenger counts and are sorted in descending order. It can be clearly seen, that several lines are overrepresented after the qando assignment. These are the lines 7, 1, 31, 41 and especially 63. This difference to the passenger counts is not particularly surprising since the stops or areas that were striking in previous evaluations are operated by these lines. Of course, at other lines larger shares are then missing, which have many more passengers in reality. Tram lines 4 and 5 operate between north and south of Graz and are underrepresented after the qando assignment. This axis is much more used in reality, but this result was expected to happen due to the previous evaluations.

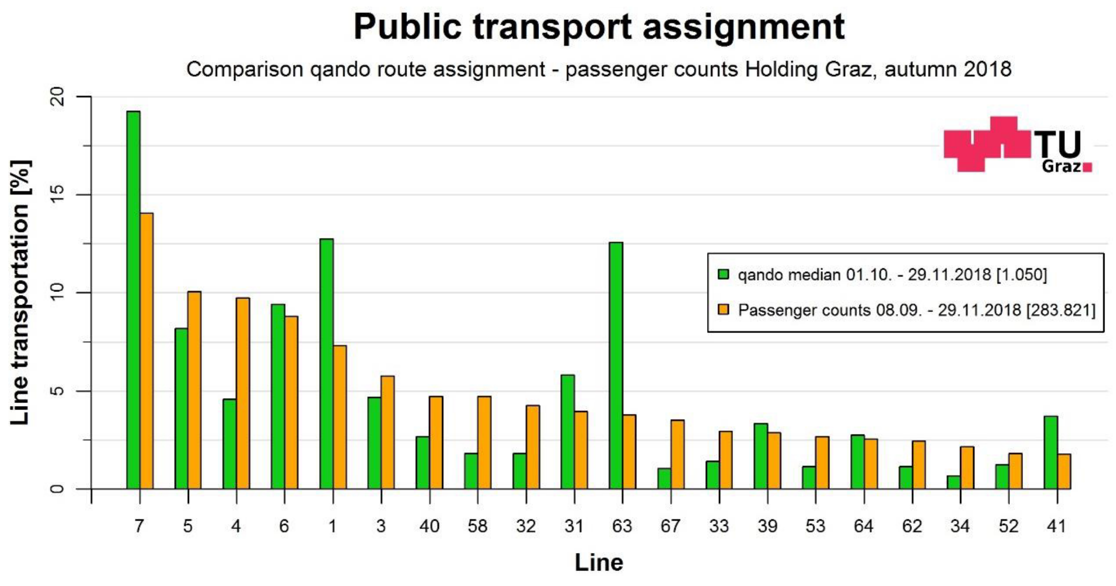


**Figure 70: Public transport assignment, comparison qando route assignment – passenger counts Holding Graz, autumn 2018**

The differences between the qando assignment and the passenger counts are shown in Figure 71. As mentioned earlier, the line 63 has received a large number of passengers in the qando assignment, which is clearly visible with a difference of 8,8 % compared to the passenger counts. Tram lines 1 and 7 also received over 5 % more trips and in contrast, tram line 4 got over 5 % less. Only a few lines exist that have received a nearly equal share of passengers in the qando assignment compared to the passenger counts.

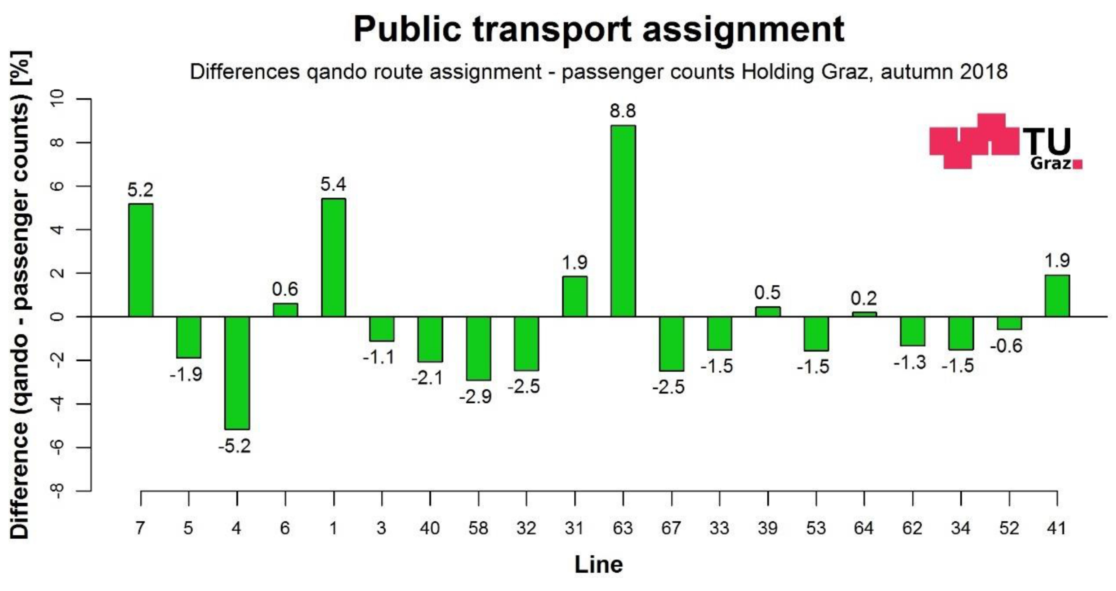


**Figure 71: Public transport assignment, differences qando route assignment – passenger counts Holding Graz, autumn 2018**

The passenger counts of the Holding Graz in winter are very similar to autumn, but the numbers of passengers on the lines are higher. In winter, additional 13.000 passengers used public transport in relation to the most used 20 lines and therefore the total is 296.934 passengers per day. Tram lines are one more time at the front of the distribution and the tram line 7 comes on average in winter to

45.636 passengers per day. On this line alone, this is an increase of almost 6.000 passengers per day (~14 %).

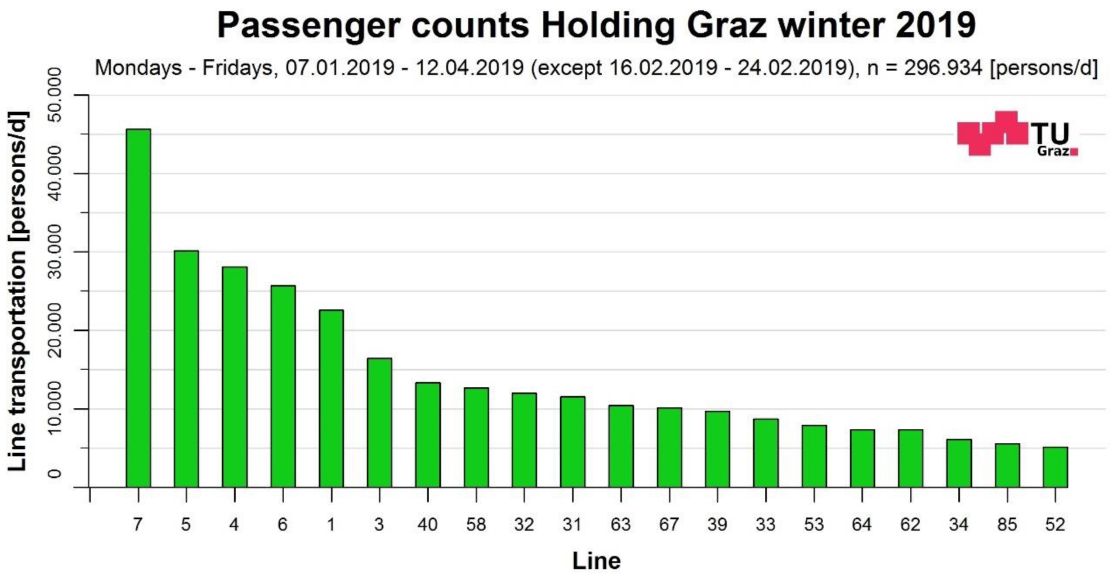


**Figure 72: Passenger counts Holding Graz, winter 2019**

As already known, fewer route requests were made in winter than in autumn. The qando assignment of the winter comes therefore only to 676 against the 1.050 passengers per day in autumn. The differences between the qando assignment and the passenger counts in winter are not so big as in autumn, but the previously mentioned lines a still burdened more heavily after the assignment. Especially the bus line 63 comes to a difference of 7,9 %. In Figure 73 the comparison of the qando assignment and the passenger counts in winter is shown and the associated differences in percentages can be found in Figure 74.

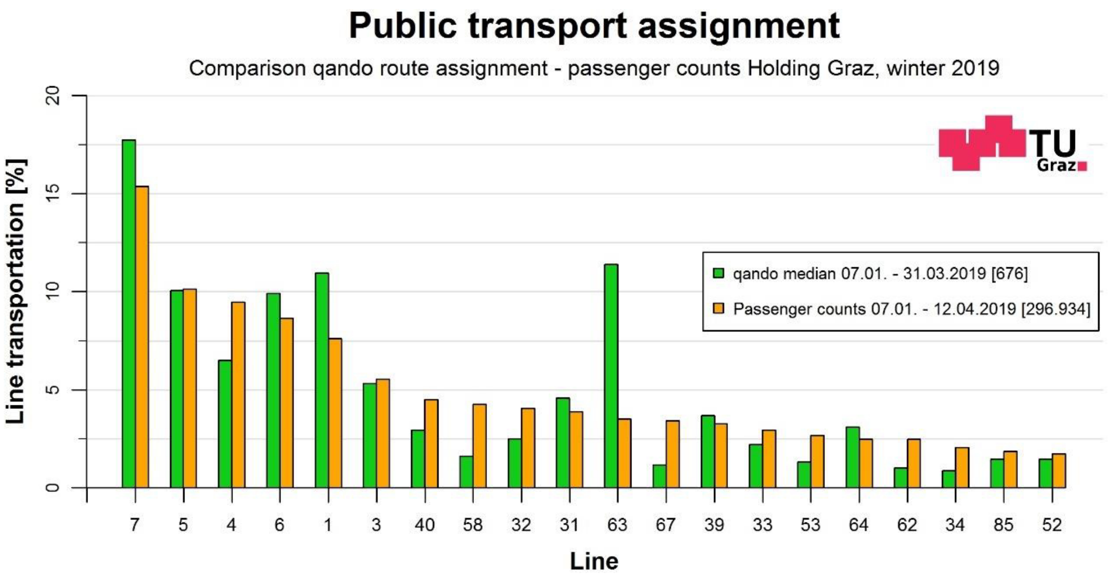


**Figure 73: Public transport assignment, comparison qando route assignment – passenger counts Holding Graz, winter 2019**

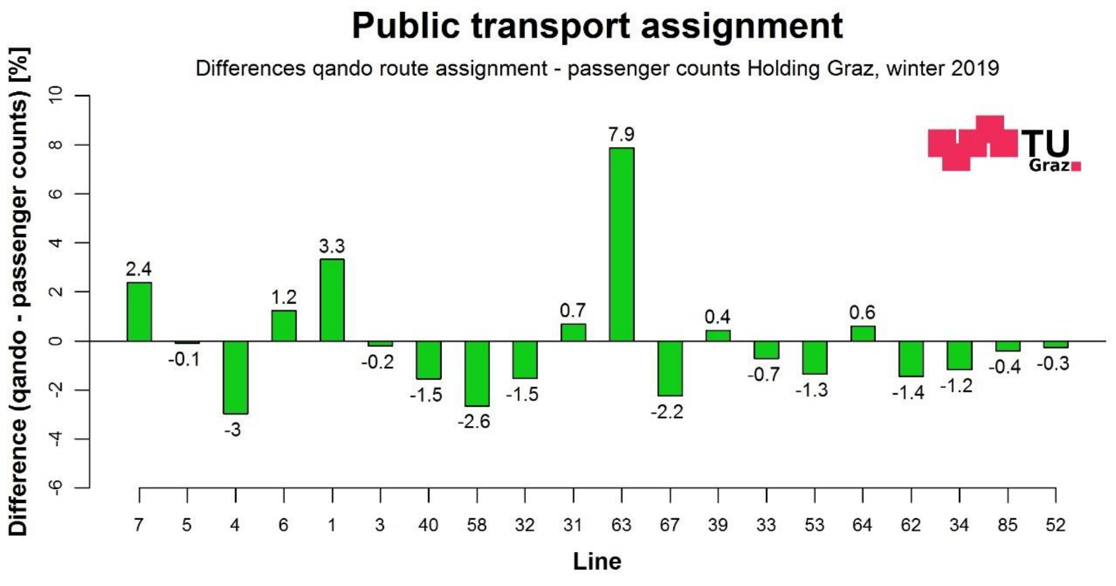


**Figure 74: Public transport assignment, differences qando route assignment – passenger counts Holding Graz, winter 2019**

### 4.2.7 Statistical comparison

In order to make not only a visual comparison between the qando assignments and the passenger counts, they are also statistically tested for significant differences. The procedure of a hypothesis test is explained below.

The hypothesis test works with two hypotheses that are established at the beginning of the test. These are the null hypothesis $H_0$ and the alternative hypothesis $H_1$. These hypotheses must exclude each other completely. The null hypothesis defines that there are no differences between the distributions, but in contrast the alternative hypothesis that there are differences.

Afterwards the significance level α is determined, which describes the probability of an error with which the null hypothesis is wrongly rejected although it is true. This probability of an error is also called α-error or type one error. The significance level is often chosen at 5 %, in other words $\alpha = 0{,}05$. This value promises a sufficient accuracy in most cases. An β-error or error of type two also exists. He is defined that the null hypothesis is not rejected even though it is wrong.

With the choosing of the significance level also the acceptance region and the critical region or rejection region is defined. The rejection region is the area in which the null hypothesis is rejected. In a one-sided test the rejection region is located at one side. By using a two-sided test, the rejection region is located on both sides of possible test statistics. Then the rejection region is divided on the left and on the right side to α/2 each. A density function with two-sided critical region can be seen in Figure 75.

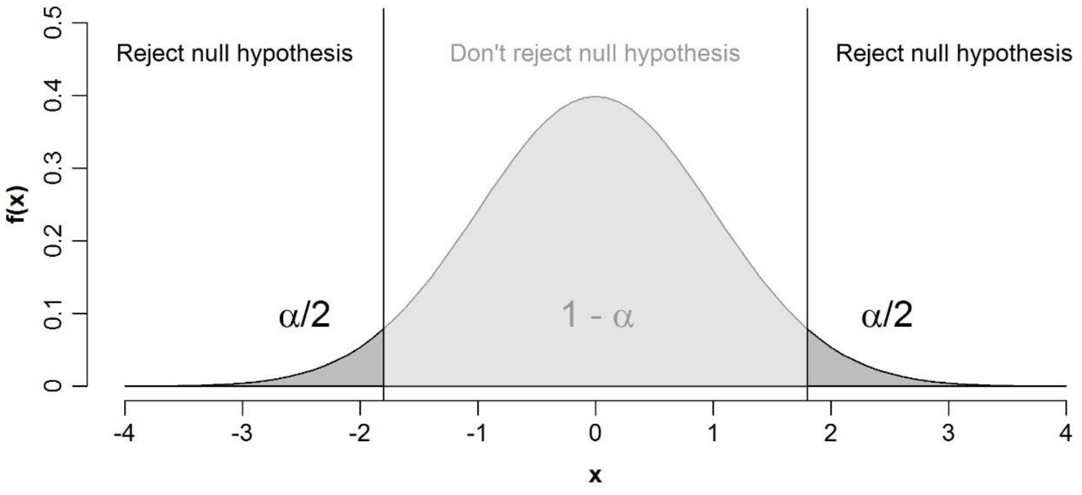


**Figure 75: Density function with two-sided critical region**

At the crossing from the acceptance region to the rejection region of the density function the critical value appears. He depends on the significance level α and on the degree of freedom of the sample and can be looked up in distribution tables. In these tables the critical values for different significance levels and degrees of freedom are indicated.

The test statistic is calculated for the selected test and with this value a decision about rejecting the null hypothesis or not can be made. Decisive is where the test statistic lays, whether she lays in the acceptance or in the rejection region. If she lays in the acceptance region the null hypothesis is not rejected. In this case the test statistic has a smaller value compared to the critical value. If the test statistic lays in the rejection region the null hypothesis is rejected, and the alternative hypothesis is accepted. When that happens, the probability is very high that the null hypothesis is wrong. If the null hypothesis can’t be rejected the test is not statistical secured that the null hypothesis is wright. In contrast the probability of an error by accepting the alternative hypothesis and rejecting the null hypothesis is very low.

Another possibility to decide is the consideration of the p-value. This indicates how likely an at least as extreme test statistic occurs as the one calculated, if the null hypothesis is correct. That means, an area under the density function that is below the calculated test statistic is determined. After getting the p-value he is compared to the significance level α. Is he lying below α then the result is significant, and the null hypothesis is rejected. Is he lying above α then the null hypothesis is not rejected (Zucchini, Schlegel, Nenadić, & Sperlich, 2009).

The statistical test, used for the comparison between qando assignments and passenger counts, is a chi-squared independence test.

#### 4.2.7.1 Chi-Squared ($\chi^2$) independence test

Given that the qando assignments and the passenger counts from the Holding Graz are compared to differences the following wordings are used for the null hypothesis $H_0$ and the alternative hypothesis $H_1$.

$H_0$: The number of passengers out of the qando assignment does <u>not</u> differ from the passenger counts from the Holding Graz.

$H_1$: The number of passengers out of the qando assignment does differ from the passenger counts from the Holding Graz.

The significance level is set to $\alpha = 0{,}05$. Thus, the probability of an error is 5 %, this is sufficiently accurate for this test. With the frequencies of the qando assignment and the passenger counts a cross table is created. The categories are listed in the upper part of the table and include the top 20 lines. Two rows are included in the table, one for the data of the qando assignment and one for the data of the passenger counts. The table is filled with the observed frequencies. Afterwards row and column sums and the total sum of the table are calculated. From these sums the expected frequencies can be calculated. For that a row sum is multiplied by a column sum and then divided by the total sum (Zucchini, Schlegel, Nenadić, & Sperlich, 2009). As an example, in Table 35 the first row has a sum of 1.050 and the first column of 40.093. Are these two sums multiplied and then divided by the total sum the expected frequency 148 is calculated for the first cell.

In Table 35 the cross table for the $\chi^2$-test with observed and expected frequencies in brackets is shown for autumn 2018. The cross table for winter 2019 can be seen in Table 36. Due to better readability the expected frequencies in the tables are rounded to a whole number. For the computation of the test statistics the exact values are used.

**Table 35: Cross table for $\chi^2$-test with observed and expected frequencies, autumn 2018**

| | 7 | 5 | 4 | 6 | 1 | 3 | 40 | 58 | 32 | 31 | 63 | 67 | 33 | 39 | 53 | 64 | 62 | 34 | 52 | 41 | ∑ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| qando | 202 (148) | 86 (106) | 48 (102) | 99 (93) | 134 (77) | 49 (60) | 28 (49) | 19 (49) | 19 (45) | 61 (42) | 132 (40) | 11 (37) | 15 (31) | 35 (30) | 12 (28) | 29 (27) | 12 (26) | 7 (23) | 13 (19) | 39 (19) | 1.050 |
| Passenger counts | 39.891 (39.945) | 28.588 (28.568) | 27.666 (27.612) | 25.016 (25.022) | 20.788 (20.845) | 16.364 (16.353) | 13.393 (13.372) | 13.386 (13.356) | 12.095 (12.069) | 11.228 (11.247) | 10.733 (10.825) | 9.988 (9.962) | 8.350 (8.334) | 8.141 (8.146) | 7.597 (7.581) | 7.245 (7.247) | 7.001 (6.987) | 6.144 (6.128) | 5.132 (5.126) | 5.075 (5.095) | 283.821 |
| ∑ | 40.093 | 28.674 | 27.714 | 25.115 | 20.922 | 16.413 | 13.421 | 13.405 | 12.114 | 11.289 | 10.865 | 9.999 | 8.365 | 8.176 | 7.609 | 7.274 | 7.013 | 6.151 | 5.145 | 5.114 | 284.871 |

**Table 36: Cross table for $\chi^2$-test with observed and expected frequencies, winter 2019**

| | 7 | 5 | 4 | 6 | 1 | 3 | 40 | 58 | 32 | 31 | 63 | 67 | 39 | 33 | 53 | 64 | 62 | 34 | 85 | 52 | ∑ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| qando | 120 (104) | 68 (69) | 44 (64) | 67 (59) | 74 (52) | 36 (37) | 20 (30) | 11 (29) | 17 (27) | 31 (26) | 77 (24) | 8 (23) | 25 (22) | 15 (20) | 9 (18) | 21 (17) | 7 (17) | 6 (14) | 10 (13) | 10 (12) | 676 |
| Passenger counts | 45.636 (45.652) | 30.132 (30.131) | 28.115 (28.095) | 25.727 (25.735) | 22.636 (22.658) | 16.430 (16.429) | 13.368 (13.358) | 12.679 (12.661) | 12.025 (12.015) | 11.578 (11.583) | 10.434 (10.487) | 10.168 (10.153) | 9.703 (9.706) | 8.739 (8.734) | 7.952 (7.943) | 7.400 (7.404) | 7.376 (7.366) | 6.119 (6.111) | 5.571 (5.568) | 5.146 (5.144) | 296.934 |
| ∑ | 45.756 | 30.200 | 28.159 | 25.794 | 22.710 | 16.466 | 13.388 | 12.690 | 12.042 | 11.609 | 10.511 | 10.176 | 9.728 | 8.754 | 7.961 | 7.421 | 7.383 | 6.125 | 5.581 | 5.156 | 297.610 |

With the now known observed and expected frequencies the test statistic $\chi^2$ can be calculated using the following formula (Zucchini, Schlegel, Nenadić, & Sperlich, 2009).

$$\chi^2 = \sum \frac{(observed\ frequencies - expected\ frequencies)^2}{expected\ frequencies}$$

Based on the size of the cross table the degree of freedom ν of the sample is determined with the following formula (Zucchini, Schlegel, Nenadić, & Sperlich, 2009).

$$\nu = (number\ of\ rows - 1) * (number\ of\ columns - 1)$$

The cross tables for autumn and winter have two rows and 20 columns. This results in a degree of freedom of $\nu = 19$.

By means of RStudio and the function chisq.test() the $\chi^2$-Test was carried out. A matrix, including the observed frequencies, is needed as input data. As results, the expected frequencies, $\chi^2$ and the p-value are outputted by the function. By looking at the p-value it is immediately obvious whether the null hypothesis can be rejected or not.

#### 4.2.7.2 Results autumn 2018

For autumn 2018, the test statistic yields to $\chi^2$ = 439,50. The critical value out of a distribution table is ${\chi^2}_{19;0.05} = 30{,}14$ and therefore the test statistic is bigger than the critical value and the null hypothesis is rejected. The significance level α = 0,01 with ${\chi^2}_{19;0.01} = 36{,}19$ is also smaller than the calculated test statistic, which means the result is highly significant. A value of p even can indicate the significance of the test. $p = 2{,}57 * 10^{-81}$ and therefore much smaller than the significance level α = 0,05 or α = 0,01.

Due to the rejection of the null hypothesis the alternative hypothesis is accepted. Thus, the number of passengers out of the qando assignment does differ from the passenger counts from the Holding Graz in autumn 2018.

#### 4.2.7.3 Results winter 2019

For winter 2019, the test statistic yields to $\chi^2$ = 185,72. The critical value out of a distribution table is ${\chi^2}_{19;0.05} = 30{,}14$ as well and therefore the test statistic is bigger than the critical value and the null hypothesis is rejected. The significance level α = 0,01 with ${\chi^2}_{19;0.01} = 36{,}19$ is also smaller than the calculated test statistic, which means the result is highly significant. The p-value in this test is $p = 2{,}30 * 10^{-29}$ and therefore much smaller than the significance level α = 0,05 or α = 0,01.

Again, in the second test the null hypothesis is rejected and the alternative hypothesis is accepted. Thus, the number of passengers out of the qando assignment does differ from the passenger counts from the Holding Graz in winter 2019.

### 4.2.8 Summary assignment results

There are big differences between the results of the qando assignments and the real occurred passenger counts. Because of many constant requests in the area of KFU the lines are heavily overrepresented in this area. This of course leads to underrepresented lines in the qando assignment, which are stronger loaded in reality. Partially little similarities are noticeable, but these are too small to conclude to real numbers of passengers out of the qando data. The number of route requests is additionally very small and that’s why a one-sided user group, like in this case probably students, carry more weight and pull the results to one side.

## 4.3 Differences between monitor and route requests

A monitor request gives the user a different result than a route request and therefore differences can be seen in the evaluations. Basically, there are much more monitor requests than route requests in six months, 12.935.536 monitor requests and with 547.041 route requests only approximately 4 % of the monitor requests were carried out. Of these, most requests were made by the own system in the monitor requests and most of the users used an iPhone for it. In the case of the route requests much more requests were sent via Android devices. An everyday user will prefer the monitor function instead of the route function. Routes are usually known for everyday trips and the departure times of the lines are much worth knowing.

During the Christmas and semester breaks collapses are recognizable in both request functions. These are by far not as pronounced in the monitor requests as in the route requests. At the beginning of the Christmas holidays the number of route requests dropped to a quarter and rose again over the duration of the holidays.

Looking at the seasons, winter was the strongest one in the monitor requests and the weakest one was spring. In the case of the route requests winter came only after autumn and spring is also the rarest requested of all.

On average Wednesdays got the most monitor requests compared to the other days of the week and Mondays got the most route requests.

The average temporal distributions of requests throughout the day are very similar. A striking morning peak and a pronounced evening area can be recognized in both request functions. Also, in the night hours declined courses occurred in monitor and route requests.

Most requests throughout one day were reached on different days. In the case of the route requests this was achieved on 10.10.2018 with 5.144 requests per day right at the beginning of the recordings. On the 23.01.2019 with 90.324 requests per day the maximum in the monitor requests was reached much later.

In terms of frequencies, the same stops or areas are the most interesting ones for the users. These are the stops Jakominiplatz, main railway station and main square and the areas around KFU and TU, school center. Lendplatz stands out only in the monitor requests and is to mention in addition. Also, the end stations stand out in the monitor and route requests.

# 5 Summary and conclusions

This last chapter summarizes the results of the master thesis and contains the conclusions that can be made due to the results observed. The conclusions are followed by a prospect into the future of mobility apps and their data.

## 5.1 Summary

The first part of the thesis dealt with a comparison of existing mobility apps and the opportunities of data usage, which are offered transport planners and users from mobility apps and traffic management systems. There are various mobility apps with different functions on the market. Some apps only serve to transmit information to the user and others offer complete solutions including planning, booking and paying for tickets or services. At the comparison two local apps from Styria, an Austria-wide app and two international apps, which are in the pot with the market leaders, were examined in detail. According to the comparison of the five apps the decision of the favored app, in terms of supply of information and services, has fallen on the Austria-wide mobility app “wegfinder”. This one convinces with a high number of different services, from car sharing to E-Scooter sharing, and the possibilities of uncomplicated, nationwide ticket purchase on the mobile device.

Subsequently, it was shown that there are definitely options for transport planners to obtain meaningful information from these systems. An important part of this is the knowledge of the line volumes in the public transport network. If the information about how many passengers are located at what time at which point in the transport network could be received from the system, transport planners would be able to use this data to make public transport more efficient. Most suitable for this purpose are passenger counting systems, which record the occupancy rate in vehicles in real-time and represent a realistic data basis. To equip all vehicles in the transport network with such systems is very costly and therefore often not nationwide feasible. Data from mobility apps are usually very inaccurate, because they depend on the inputs of the users and do not really reflect the reality, because not every passenger uses a mobility app and even app users do not request every trip.

Benefits that a user could gain from these systems make the journey from A to B more stress-free and faster for the user. Thereby the best route information can automatically be sent to the user or the information about appropriate alternatives can be transmitted to him and the user can then decide by himself which option to choose. By transmitting the degree of occupancy to stops or mobility apps, users can adapt to the conditions in the transport network. Thus, the duration of the boarding process at stops can be reduced and the operation of the mean of transport is not delayed unnecessarily.

The second part of the thesis dealt with the user requests of the mobility app “qando Graz”. In general, qando Graz is a journey planner with included real-time information in the transport network and offers no shared mobility services or ticketing. The considered data of this app are divided in monitor and route requests, whereby much more monitor requests were made in the six months. Compared to the real occurring number of passengers the proportion of qando requests is very low, since not everyone uses a mobility app and several apps from different providers can be used in Graz.

The proportion of requests at weekends is relatively high compared to the real occurring public transport amount. At weekends more information is needed and this affects the number of requests.

Due to higher headways of means of transport and non-daily trips the desired information is received through requests in qando.

Most frequently requested stops were Jakominiplatz, main railway station and main square. Jakominiplatz is by far the number one stop and reached an average of 926,5 monitor requests per day. Also very striking are the areas around KFU and TU, school center, which achieved a not insignificant proportion by the addition of all stops in these areas. A protruding stop is Hertzgasse in the southeast of Graz. The stop got a fairly large numbers of requests from one and the same user who sent a request to the system in 30-second intervals. This points to a monitor set up in the own four walls, which updates the departure times of the station every 30 seconds and thus always displays the times in real-time. After building the median over the six month observation period this stop is no longer conspicuous as it has not been requested consistently enough over the long period of time.

Requests to the app can also be sent using the actual position of the user. A long-term view over the six months was not possible with these requests, because the GPS coordinates are recorded very accurately to the sixth decimal place and a constant use of the same coordinate over a longer period of time is very unlikely. Nevertheless, it happened that some coordinates received several hundred requests per day. The positions were automatically requested on a regular basis and thus not the actual use of the service, and with it a trip, was reflected.

The low number of route requests makes it hard to deliver meaningful results. However, a picture worth seeing has set over the observation period. The majority of requested routes took place between the triangle Jakominiplatz – KFU – main railway station. During the week the axis Jakominiplatz – KFU was most frequently requested, on Fridays the axis KFU – main railway station and at the weekends the axis main railway station – Jakominiplatz. Due to this distribution of the route requests it can be seen that many qando users travel via the main railway station from Graz or to Graz at weekends. On average over the six months the route between Jakominiplatz – KFU received with 24 requests per day the most.

In order to get the volume on the line network of Graz out of the received route data from qando, public transport assignments were carried out in the transport planning software PTV Visum. Because of the small number of routes, daily matrices with the belonging hydrographs were used for the assignment. Creating hourly matrices, 24 matrices for one day, produced only little or no values in one matrix. Therefore, this approach was not considered as sufficient and discarded. Due to the distribution of the route requests, especially between Jakominiplatz – KFU – main railway station, the lines operating in these areas are strongly overrepresented after the public transport assignments of the qando data. These operating lines are the lines 7, 1, 63 and 31.

When comparing the assigned qando data with the real occurring passenger counts of the Holding Graz the overrepresentation of some lines is clearly visible. Lines in the area of the triangle have much higher values in percentage terms than they actually have in reality. Only logical that then many requests are missing on other lines, which are actually not so underrepresented. Primarily tram lines 4 and 5, which are operating between the north and south of Graz, are worth mentioning. Deriving the volume on the lines from app requests is possible, but does not reflect reality in this thesis. The assignment of the requests to the lines strongly depends on the user group of the app and this is quite ambiguous in the qando app.

To statistically evaluate this comparison and to check for a correlation between the mobility app data and the actual passengers, a hypothesis test was carried out with the $\chi^2$-test. After conducting the test in RStudio it came to the result that based on this overrepresentation of some lines there is statistically no correlation between the qando data and the real occurring passenger counts of the Holding Graz. The null hypothesis, which would have assumed a correlation between the two datasets, was clearly rejected and the alternative hypothesis was accepted. The result is very clear, because the statistical test received highest significance as an outcome for accepting the alternative hypothesis

## 5.2 Conclusions

Based on the results obtained in this master thesis it can be seen that the qando Graz user group is very one-sided distributed. It can be assumed that a large proportion of the qando users are students at the universities or related employees. This can only be assumed due to the obtained results, as there is no demographic information about it in the data. Observations confirming this assumption are in any case the collapses of requests in the non-lecture times during the Christmas holidays and the semester break in February, the accumulation of requests at the main railway station and in the vicinity of universities and in addition the route connections between Jakominiplatz – KFU and KFU – main railway station. To be able to make a reliable statement, an extra investigation would have to be carried out.

The well-known stops with many people movements are also very popular with the request in qando. These stations are located in the central area of Graz and many passengers travel from or to these points. At these stops a change between several lines or between different means of transport is possible, which makes them very attractive for requests. In the route requests the stops Jakominiplatz and main railway station as well as the area around the KFU play an essential role. Considering the six months, a tringle of route has stretched over these points and that leaves the other route connections a little lost due to the much larger dimension of these three points.

Compared to the seasons autumn and winter, the fewest monitor and route requests were sent in spring. This can be attributed to the fact that the weather in Graz was already relatively mild in February and March and the temperatures had already invited to switch from public transport to the bike or to travel the routes by foot. Of course, fewer passengers in the transport network lead to fewer requests via the qando app.

Based on the data discussed in this thesis it is not possible to conclude from the qando data to the real occurring traffic demand in public transport, since the user group is too one-sidedly distributed. The overrepresentation of several lines after the public transport assignments leads to the fact that in reality weaker used lines are displayed too strongly and conversely more heavily used lines receive a much smaller share of passengers.

## 5.3 Future prospects

Mobility apps are playing an increasingly important role in modern mobility, a big part of passengers uses them every day. There are already a large number of such apps on the market and this amount continues to increase. Many companies or transport operators are in this market or want to enter it. Thereby various solutions are offered, an increasingly number of companies offer complete solutions, with which not only information can be displayed, but also services can be used and paid directly and easily. In contrast, there are simple apps that only exist to give the user the best information he needs. The expansion level of the app depends heavily on the mobility offers in the respective cities. As the supply of mobility in cities increases, so will the information and service supply of mobility apps.

Processing big data and further use of it opens many possibilities for transport planners, if evaluated and interpreted correctly, and after that the user benefits as well. By including data from traffic management systems and mobility apps, the demand for mobility, whether public transport or private mobility services, can be better estimated, designed more efficient and better offered to the users. A reliable data basis of mobility apps could reduce costly installations of sensors in the vehicles and expensive investigations, because existing data, created by users, could be used.

If the user group of a mobility app is homogeneously spread relative to the usage of public transport lines, then the data will be more meaningful for transport planning purposes. However, some data analysis is already possible even with biased data. The awareness is directed here on the temporal level. On the basis of evaluations of monitor data, analyses of the temporal usage of certain stops can be made. Also, highly requested routes can be analysed for their temporal use. Long-term monitoring of several months or years may detect recurring behaviours.

In the future a lot will happen in the still very young market. Additional solutions will emerge through innovative ideas and the further development of technology. Mobility must change in cities in order to be able to handle the increasing traffic demand accordingly and to make it more efficient and comfortable.

# List of references

Bruglieri, M., Bruschi, F., Colorni, A., Lue, A., Nocerino, R., & Rana, V. (2015). *A real-time information system for public transport in case of delays and service disruptions.* Delft: Elsevier B.V.

Environmental Systems Research Institute, Inc. (2016). *What is a shapefile?* Retrieved from http://desktop.arcgis.com/en/arcmap/10.3/manage-data/shapefiles/what-is-a-shapefile.htm [11.07.2019]

Fabritiis, C., Ragona, R., & Valenti, G. (2008). *Traffic Estimation And Prediction Based On Real Time Floating Car Data.* China: IEEE.

Fluidtime. (2019a). *About Fluidtime*. Retrieved from https://www.fluidtime.com/en/about-us/ [05.07.2019]

Fluidtime. (2019b). *qando – Vienna, Linz, Salzburg, Graz, Klagenfurt*. Retrieved from https://www.fluidtime.com/en/project/qando/#toggle-id-4-closed [05.07.2019]

Holding Graz - Kommunale Dienstleistungen GmbH. (2019). *qando Graz*. Retrieved from https://www.holding-graz.at/de/app/qando.html [03.07.2019]

iMobility GmbH. (2019). *Wegfinder. Wie wohin. Vergleiche, kombiniere und buche neue Wege.* Retrieved from https://wegfinder.at/ [09.08.2019]

iris-GmbH infrared & intelligent sensors. (2019). *Passenger counting*. Retrieved from https://www.iris-sensing.com/ [24.08.2019]

MaaS Global. (2019). *whim. All your journeys*. Retrieved from https://whimapp.com/ [10.08.2019]

moovel Group GmbH. (2019). *REACH NOW. Frequently Asked Questions*. Retrieved from https://support.reach-now.com/hc/en-us [10.08.2019]

PTV Group. (2019). *PTV Visum*. Retrieved from https://www.ptvgroup.com/en/solutions/products/ptv-visum/ [02.08.2019]

Robert Bosch GmbH. (2019). *Community-based parking: helping one another find the nearest available parking space more quickly*. Retrieved from https://www.bosch-mobility-solutions.com/en/products-and-services/mobility-services/connected-parking/community-based-parking/ [25.08.2019]

Verkehrsverbund Steiermark GmbH. (2019). *BusBahnBim-App*. Retrieved from https://www.verbundlinie.at/fahrplan/fahrplaene/fahrplan-app [10.08.2019]

Zucchini, W., Schlegel, A., Nenadić, O., & Sperlich, S. (2009). *Statistik für Bachelor- und Masterstudenten.* Göttingen: Springer.

# Appendix A – R-Code for data processing, monitor data

```
# Used libraries
library(png) # For integrading pictures in png format
library(maptools) # For using readShapeSpatial to show shape files
library(scales) # For using alpha to make lines transparent
library(SDMTools) # For getting an legend with gradient

# Define the working directory for the project
working_directory <- "D:/qando Graz/"
```

## Data preparation

```
# diva = stop
# Set the working directory in which the exported files are stored
setwd(paste(working_directory, "Tables/Monitor", sep = ""))

# Read in monitor.csv
Monitor <- read.csv(paste(working_directory, "Data/monitor.csv", sep = ""),
           stringsAsFactors = FALSE, check.names = FALSE)

# Delete columns "Hashed Host", "Log Name", "Method" and "Bytes Sent"
Monitor <- Monitor[,-c(1,2,5,8)]

# Delete response code unequal 200 (200 = request successful)
RC_codes <- c(304,400,401,404,500,503)
for(i in 1:length(RC_codes)) {
  RC_del <- grep(RC_codes[i], Monitor[,4])
  if(identical(RC_del, integer(0)) == FALSE) {
    Monitor <- Monitor[-RC_del,]
  }
}

# Delete column "Response Code"
Monitor <- Monitor[,-c(4)]

# Sort table by date in increasing order
Monitor <- Monitor[order(Monitor[,1], decreasing = FALSE),]

# Split column "Date Time" and add the resulting two columns to the table
Monitor <- cbind(matrix(unlist(strsplit(Monitor[,1], " ")), nrow = nrow(Monitor),
           byrow = TRUE), Monitor, deparse.level = 0)

# Delete columns "Date Time" and "Time Zone"
Monitor <- Monitor[,-c(3,4)]

# Read out the used platform from the column "User Agent" and create a new column
  for it in the table
Monitor <- cbind(Monitor[,1:2], matrix(NA, nrow(Monitor), 1), Monitor[,3:4])
Platforms <- c("Monitor", "iPhone", "Android", "iPad", "Web", "Mozilla",
             "CFNetwork", "Apache")
for(i in 1:length(Platforms)) {
  Platforms_read_out <- grep(Platforms[i], Monitor[,5])
  if(identical(Platforms_read_out, integer(0)) == FALSE) {
    Monitor[Platforms_read_out,3] <- Platforms[i]
  }
}

# Delete column "User Agent"
Monitor <- Monitor[,-c(5)]

# Replace all %3A with :
Monitor[,4] <- gsub("%3A", ":", Monitor[,4], fixed = TRUE)

# Add a & to every diva request
rows_diva <- grep("diva=", Monitor[,4])
Monitor[rows_diva,4] <- paste(Monitor[rows_diva,4], "&", sep = "")

# Read out diva from column "URL" and write it into a new column in the table
Monitor <- cbind(Monitor[,1:3], matrix(NA, nrow(Monitor), 1), Monitor[,4])
Monitor[rows_diva,4] <- unlist(regmatches(Monitor[rows_diva,5],
                         regexec("diva=(.*?)&", Monitor[rows_diva,5])))[c(seq(2,
                         length(rows_diva)*2, by = 2))]

# Read out nearby coordinates from column "URL" and write them into two new columns
  in the table
```

```
Monitor <- cbind(Monitor, matrix(NA, nrow(Monitor), 2))
rows_nearby <- grep("Nearby", Monitor[,5])
Monitor[rows_nearby,6] <- unlist(regmatches(Monitor[rows_nearby,5],
                          regexec("coord=(.*?):",
                          Monitor[rows_nearby,5])))[c(seq(2, length(rows_nearby)*2,
                          by = 2))]
Monitor[rows_nearby,7] <- unlist(regmatches(Monitor[rows_nearby,5],
                          regexec(":(.*?):", Monitor[rows_nearby,5])))[c(seq(2,
                          length(rows_nearby)*2, by = 2))]

# Read out favorites from column "URL"
rows_monitor <- grep("Monitor", Monitor[,3])
Monitor[rows_monitor,5] <- NA
Monitor[rows_diva,5] <- NA
Monitor[rows_nearby,5] <- NA
Monitor <- cbind(Monitor, matrix(NA, nrow(Monitor), 1))

rows_rblLineDirs <- grep("rblLineDirs=", Monitor[,5])
Monitor[rows_rblLineDirs,8] <- "yes"
Monitor[rows_rblLineDirs,5] <- NA

rows_divaLineDirs <- grep("divaLineDirs=", Monitor[,5])
Monitor[rows_divaLineDirs,8] <- "yes"
Monitor[rows_divaLineDirs,5] <- NA

# Delete column "URL"
Monitor <- Monitor[,-c(5)]

# Add weekdays in a new column to the table
Monitor <- cbind(matrix(NA, nrow(Monitor), 1), Monitor)
Monitor[,1] <- weekdays(as.Date(Monitor[,2], "%Y-%m-%d"))

# Mark nearby and diva rows
Monitor <- cbind(Monitor, matrix(NA, nrow(Monitor), 2))
Monitor[rows_diva,9] <- "yes"
Monitor[rows_nearby,10] <- "yes"

# Read in the coordinates of the divas
Coordinates <- read.csv2(paste(working_directory, "Data/Graz_stop_coordinates.csv",
               sep = ""), stringsAsFactors = FALSE, check.names = FALSE)

# Add coordinates to divas
for(i in 1:nrow(Coordinates)) {
  rows_coordinate <- grep(Coordinates[i,1], Monitor[,5])
  Monitor[rows_coordinate,6] <- Coordinates[i,3]
  Monitor[rows_coordinate,7] <- Coordinates[i,4]
}

# Remove divas without coordinates
if(identical(grep(TRUE, is.na(Monitor[rows_diva,6])), integer(0)) == FALSE) {
  Monitor <- Monitor[-c(rows_diva[grep(TRUE, is.na(Monitor[rows_diva,6]))]),]
}

# Change the column names of the table
colnames(Monitor) <- c("Weekday", "Date", "Time", "Platform", "Divas", "Coord_X",
                     "Coord_Y", "Favorites", "Diva", "Nearby")

# Write monitor table to csv.file
write.csv2(Monitor, file = "Monitor_evaluations_start_file.csv", row.names = FALSE)
```

## Distribution tables for stop, position, monitor and favorite requests

```
# Set the working directory in which the exported files are stored
setwd(paste(working_directory, "Tables/Monitor/Distribution Tables", sep = ""))

# Read in sorted monitor table
Monitor <- read.csv2(paste(working_directory,
           "Tables/Monitor/Monitor_evaluations_start_file.csv", sep = ""),
           stringsAsFactors = FALSE, check.names = FALSE)

# Use first element of date
Monitor[,3] <- unlist(regmatches(Monitor[c(1:nrow(Monitor)),3],
               regexec("(.*?):", Monitor[c(1:nrow(Monitor)),3])))[c(seq(2,
               nrow(Monitor)*2, by = 2))]

# Create four matrices for stop, position, favorite and monitor requests
rows_monitor <- grep("Monitor", Monitor[,4])
```

```
rows_favorite <- grep("yes", Monitor[,8])
rows_diva <- grep("yes", Monitor[,9])
rows_nearby <- grep("yes", Monitor[,10])

Frequency_monitor<- Monitor[rows_monitor,]
Frequency_favorite <- Monitor[rows_favorite,]
Frequency_stop <- Monitor[rows_diva,]
Frequency_nearby <- Monitor[rows_nearby,]

#
## Frequencies stop requests ##

# Build a contingency table of the frequencies between time and date
Frequency_stop <- table(Frequency_stop[,3], Frequency_stop[,2])
Frequency_stop <- as.data.frame.matrix(Frequency_stop)
Frequency_stop <- cbind(matrix(NA, nrow(Frequency_stop),1), Frequency_stop)
rownames(Frequency_stop) <- c(1:nrow(Frequency_stop))
Frequency_stop[,1] <- rownames(Frequency_stop)

# Change the column name of the first column
colnames(Frequency_stop)[1] <- c("Frequency")

# Change the format of date and replace it in the column names
date <- as.integer(as.POSIXct(colnames(Frequency_stop)[2:ncol(Frequency_stop)],
        tryFormats = c("%Y-%m-%d"), origin = "1970-01-01"))
class(date) = c('POSIXct')
date <- as.character.Date(date, tryFormats = c("%d.%m.%Y"))
date <- format(as.Date(date), "%d.%m.%Y")
colnames(Frequency_stop)[2:ncol(Frequency_stop)] <- date

# Add sum and medians to the stop table
median <- matrix(NA, nrow(Frequency_stop), 12)
colnames(median) <- c("Sum",
                    "Median 01.10.2018 - 31.03.2019",
                    "Median autumn 01.10.2018 - 18.11.2018",
                    "Median winter 19.11.2018 - 10.02.2019",
                    "Median spring 11.02.2019 - 31.03.2019",
                    "Median Mondays 01.10.2018 - 31.03.2019",
                    "Median Tuesdays 01.10.2018 - 31.03.2019",
                    "Median Wednesdays 01.10.2018 - 31.03.2019",
                    "Median Thursdays 01.10.2018 - 31.03.2019",
                    "Median Fridays 01.10.2018 - 31.03.2019",
                    "Median Saturdays 01.10.2018 - 31.03.2019",
                    "Median Sundays 01.10.2018 - 31.03.2019")

Frequency_stop <- cbind(Frequency_stop, median)

# Define the time periods for autumn, winter and spring
start_autumn <- "01.10.2018"
end_autumn <- "18.11.2018"
start_winter <- "19.11.2018"
end_winter <- "10.02.2019"
start_spring <- "11.02.2019"
end_spring <- "31.03.2019"

# Calculate the sum and the different medians for every hour and write them into
  the belonging columns
Frequency_stop[,grep("Sum", colnames(Frequency_stop))] <-
     apply(Frequency_stop[,2:(grep("Sum", colnames(Frequency_stop))-1)], 1, sum)
Frequency_stop[,grep("Median 01.10.2018 - 31.03.2019", colnames(Frequency_stop))]
     <- apply(Frequency_stop[,2:(grep("Sum", colnames(Frequency_stop))-1)], 1,
     median)
Frequency_stop[,grep("Median autumn 01.10.2018 - 18.11.2018",
     colnames(Frequency_stop))] <- apply(Frequency_stop[,grep(start_autumn,
     colnames(Frequency_stop))[1]:grep(end_autumn, colnames(Frequency_stop))[1]],
     1, median)
Frequency_stop[,grep("Median winter 19.11.2018 - 10.02.2019",
     colnames(Frequency_stop))] <- apply(Frequency_stop[,grep(start_winter,
     colnames(Frequency_stop))[1]:grep(end_winter, colnames(Frequency_stop))[1]],
     1, median)
Frequency_stop[,grep("Median spring 11.02.2019 - 31.03.2019",
     colnames(Frequency_stop))] <- apply(Frequency_stop[,grep(start_spring,
     colnames(Frequency_stop))[1]:grep(end_spring, colnames(Frequency_stop))[1]],
     1, median)
Frequency_stop[,grep("Median Mondays 01.10.2018 - 31.03.2019",
     colnames(Frequency_stop))] <- apply(Frequency_stop[,seq(2, (grep("Sum",
     colnames(Frequency_stop))-1), by=7)], 1, median)
Frequency_stop[,grep("Median Tuesdays 01.10.2018 - 31.03.2019",
     colnames(Frequency_stop))] <- apply(Frequency_stop[,seq(3, (grep("Sum",
     colnames(Frequency_stop))-1), by=7)], 1, median)
Frequency_stop[,grep("Median Wednesdays 01.10.2018 - 31.03.2019",
     colnames(Frequency_stop))] <- apply(Frequency_stop[,seq(4, (grep("Sum",
     colnames(Frequency_stop))-1), by=7)], 1, median)
Frequency_stop[,grep("Median Thursdays 01.10.2018 - 31.03.2019",
     colnames(Frequency_stop))] <- apply(Frequency_stop[,seq(5, (grep("Sum",
     colnames(Frequency_stop))-1), by=7)], 1, median)
Frequency_stop[,grep("Median Fridays 01.10.2018 - 31.03.2019",
     colnames(Frequency_stop))] <- apply(Frequency_stop[,seq(6, (grep("Sum",
     colnames(Frequency_stop))-1), by=7)], 1, median)
Frequency_stop[,grep("Median Saturdays 01.10.2018 - 31.03.2019",
     colnames(Frequency_stop))] <- apply(Frequency_stop[,seq(7, (grep("Sum",
     colnames(Frequency_stop))-1), by=7)], 1, median)
Frequency_stop[,grep("Median Sundays 01.10.2018 - 31.03.2019",
     colnames(Frequency_stop))] <- apply(Frequency_stop[,seq(8, (grep("Sum",
     colnames(Frequency_stop))-1), by=7)], 1, median)

# Calculate the sum of requests on one day
sum_days <- matrix(NA, 1, ncol(Frequency_stop))
colnames(sum_days) <- colnames(Frequency_stop)
Frequency_stop <- rbind(Frequency_stop, sum_days)
for(i in 2:ncol(Frequency_stop)) {
  Frequency_stop[nrow(Frequency_stop), i] <-
  sum(Frequency_stop[1:(nrow(Frequency_stop)-1), i])
}

# Write frequency stop table to csv.file
write.csv2(Frequency_stop, file = "Distribution_table_stop.csv", row.names = FALSE)

# The processing for creating the distribution tables for position, favorite and
  monitor requests is equal to the stop requests
```

## Distribution plots: Medians six months

```
setwd(paste(working_directory, "Plots/Monitor/Distribution Plots/Medians", sep =
""))

# Read in frequency tables for stop, position, favorite and monitor requests
Frequency_monitor <- read.csv2(paste(working_directory,
                     "Tables/Monitor/Distribution Tables/
                     Distribution_table_monitor.csv", sep = ""),
                     stringsAsFactors = FALSE, check.names = FALSE)
Frequency_favorite <- read.csv2(paste(working_directory,
                      "Tables/Monitor/Distribution Tables/
                      Distribution_table_favorite.csv", sep = ""),
                      stringsAsFactors = FALSE, check.names = FALSE)
Frequency_stop <- read.csv2(paste(working_directory, "Tables/Monitor/
                  Distribution Tables/Distribution_table_stop.csv", sep = ""),
                  stringsAsFactors = FALSE, check.names = FALSE)
Frequency_nearby <- read.csv2(paste(working_directory, "Tables/Monitor/
                    Distribution Tables/Distribution_table_nearby.csv", sep = ""),
                    stringsAsFactors = FALSE, check.names = FALSE)

# Import the logo of the TU Graz
TULogo <- readPNG(paste(working_directory, "Data/TULogo.png", sep = ""))

for(i in (grep("Sum", colnames(Frequency_monitor))+1):ncol(Frequency_monitor)) {

  # Create a jpeg-file
  jpeg(filename = paste(i-184, ".Distribution_monitor_requests_", gsub(" ", "",
  sub(" ", "_", sub(" ", "_", colnames(Frequency_monitor)[i]))), ".jpg", sep = ""),
  width = 8.33, height = 4.58, units = "in", pointsize = 12, quality = 150, bg =
  "white", res = 300, family = "", restoreConsole = TRUE, type = c("windows",
  "cairo"))

  # Define the edges of the plot
  par(mgp = c(2,1,0), oma = c(0,0,0,0), pin = c(7,2.7))

  # Maximum value on the y-axis
  max_y <- max(c(max(Frequency_nearby[1:24,(grep("Sum",
           colnames(Frequency_monitor))+1):ncol(Frequency_monitor)]),
           max(Frequency_stop[1:24,(grep("Sum",
           colnames(Frequency_monitor))+1):ncol(Frequency_monitor)]),
           max(Frequency_favorite[1:24,(grep("Sum",
           colnames(Frequency_monitor))+1):ncol(Frequency_monitor)]),
           max(Frequency_monitor[1:24,(grep("Sum",
         colnames(Frequency_monitor))+1):ncol(Frequency_monitor)])))

  # Create a bar plot
  barplot(rbind(Frequency_nearby[1:24,i], Frequency_stop[1:24,i],
  Frequency_favorite[1:24,i], Frequency_monitor[1:24,i]), beside = TRUE, width =
  c(0.15), space = c(0,2.665), legend.text = FALSE, col=c("yellow", "blue", "red",
  "white"), names.arg = Frequency_monitor[1:24,i], xaxt = "n", yaxt = "n", xlim =
  c(1,24), ylim = c(0, (ceiling(max_y/100)*100)+350))

  # Add horizontal lines behind the bars
  abline(h = seq(250, (ceiling(max_y/100)*100)+350, by = 250), col = "gray87")

  # Repeat create a bar plot to bring the bars in front of the before added
    horizontal lines
  barplot(rbind(Frequency_nearby[1:24,i], Frequency_stop[1:24,i],
  Frequency_favorite[1:24,i], Frequency_monitor[1:24,i]), beside = TRUE, width =
  c(0.15), space = c(0,2.665), legend.text = FALSE, col=c("yellow", "blue", "red",
  "white"), names.arg = Frequency_monitor[1:24,i], xaxt = "n", yaxt = "n", xlim =
  c(1,24), ylim = c(0, (ceiling(max_y/100)*100)+350), add = TRUE)

  # Add a x-axis to the plot
  axis(side = 1, at = c(0:24)+0.19, labels = c(0:24), cex.axis = 0.75, padj = -0.7)
  title(xlab = "Time [h]", font.lab = 2, cex.lab = 1.0)

  # Add a y-axis to the plot
  axis(side = 2, at = seq(0, ceiling(max_y/100)*100+350, by = 500), labels =
  gsub(" ", "", format(seq(0, ceiling(max_y/100)*100+350, by = 500), big.mark =
  ".", decimal.mark = ",")), cex.axis = 0.75, padj = 0.5)
  axis(side = 2, at = seq(250, ceiling(max_y/100)*100, by = 500), labels = FALSE,
  lwd.ticks = 0.8, tck = -0.02)
  title(ylab = "Frequency of requests [requests/h]", font.lab = 2, cex.lab = 1.0)

  # Add a main title and a subtitle to the plot
  title(main = "Distribution monitor requests", font.main = 2, cex.main = 1.5,
  line = 2)
  mtext(paste(colnames(Frequency_monitor)[i], ", n = ",
  format((Frequency_nearby[25,i]+Frequency_stop[25,i] + Frequency_favorite[25,i] +
  Frequency_monitor[25,i]), big.mark = ".", decimal.mark = ","), " [requests/d]",
  sep = ""), side = 3, cex = 0.9, line = 0.5)

  # Add a legend to the plot
  legend(0.5, ceiling(max_y/100)*100+350, legend = paste("Position requests [",
  format(Frequency_nearby[25,i], big.mark = ".", decimal.mark = ",") , "]", sep =
  ""), pch = 22, bty = "n", cex = 0.7, col = "black", pt.bg = "yellow", pt.cex = 1)

  legend(6.25, ceiling(max_y/100)*100+350, legend = paste("Stop requests [",
  format(Frequency_stop[25,i], big.mark = ".", decimal.mark = ",") , "]", sep =
  ""), pch = 22, bty = "n", cex = 0.7, col = "black", pt.bg = "blue", pt.cex = 1)

  legend(11.7, ceiling(max_y/100)*100+350, legend = paste("Favorite requests [",
  format(Frequency_favorite[25,i], big.mark = ".", decimal.mark = ",") , "]", sep =
  ""), pch = 22, bty = "n", cex = 0.7, col = "black", pt.bg = "red", pt.cex = 1)

  legend(17.75, ceiling(max_y/100)*100+350, legend =
  paste("Departureboard requests [", format(Frequency_monitor[25,i], big.mark =
  ".", decimal.mark = ",") , "]", sep = ""), pch = 22, bty = "n", cex = 0.7, col =
  "black", pt.bg = "white", pt.cex = 1)

  # Add the TU logo to the plot
  rasterImage(TULogo, 0.2,1750,2.318,2000)
  dev.off()
}
# The other distribution plots are created the same way
```

## Frequency tables for stop and position requests

```
# Set the working directory in which the exported files are stored
setwd(paste(working_directory, "Tables/Monitor/Frequency Tables", sep = ""))

# Read in sorted monitor table
Monitor <- read.csv2(paste(working_directory,
           "Tables/Monitor/Monitor_evaluations_start_file.csv", sep = ""),
           stringsAsFactors = FALSE, check.names = FALSE)

# Delete rows which do not have coordinates
rows_monitor <- c(1:nrow(Monitor))
rows_coordinates <- grep(".", Monitor[,6])
```

```
rows_delete <- rows_monitor[-c(rows_coordinates)]
Monitor <- Monitor[-c(rows_delete),]

# Create one matrix for stop and one for position requests
rows_diva <- grep("yes", Monitor[,9])
rows_nearby <- grep("yes", Monitor[,10])
Stop_matrix <- Monitor[rows_diva,]
Position_matrix <- Monitor[rows_nearby,]

#
## Stop matrix ##

# Build a contingency table of the frequencies between coordinates and date
Stop_matrix <- table(paste(Stop_matrix[,6], "/", Stop_matrix[,7], sep = ""),
               Stop_matrix[,2])
Stop_matrix <- as.data.frame.matrix(Stop_matrix)
Stop_matrix <- cbind(matrix(NA, nrow(Stop_matrix),1), Stop_matrix)
Stop_matrix[,1] <- rownames(Stop_matrix)
rownames(Stop_matrix) <- c(1:nrow(Stop_matrix))

# Split the coordinates into x and y coordinates
Stop_matrix <- cbind(matrix(NA, nrow(Stop_matrix),2), Stop_matrix)
Stop_matrix[c(1:nrow(Stop_matrix)),1] <-
     unlist(regmatches(Stop_matrix[c(1:nrow(Stop_matrix)),3], regexec("(.*?)/",
     Stop_matrix[c(1:nrow(Stop_matrix)),3])))[c(seq(2, nrow(Stop_matrix)*2, by =
     2))]
Stop_matrix[c(1:nrow(Stop_matrix)),2] <-
     unlist(regmatches(Stop_matrix[c(1:nrow(Stop_matrix)),3], regexec("/(.*?)$",
     Stop_matrix[c(1:nrow(Stop_matrix)),3])))[c(seq(2, nrow(Stop_matrix)*2, by =
     2))]

# Change the column names of the first three columns
colnames(Stop_matrix)[1:3] <- c("Coord X", "Coord Y", "X/Y")

# Change the format of date and replace it in the column names
date <- as.integer(as.POSIXct(colnames(Stop_matrix)[4:ncol(Stop_matrix)],
        tryFormats = c("%Y-%m-%d"), origin = "1970-01-01"))
class(date) = c('POSIXct')
date <- as.character.Date(date, tryFormats = c("%d.%m.%Y"))
date <- format(as.Date(date), "%d.%m.%Y")
colnames(Stop_matrix)[4:ncol(Stop_matrix)] <- date

# Add sum and medians to the stop matrix
median <- matrix(NA, nrow(Stop_matrix), 12)
colnames(median) <- c("Sum",
                    "Median 01.10.2018 - 31.03.2019",
                    "Median autumn 01.10.2018 - 18.11.2018",
                    "Median winter 19.11.2018 - 10.02.2019",
                    "Median spring 11.02.2019 - 31.03.2019",
                    "Median Mondays 01.10.2018 - 31.03.2019",
                    "Median Tuesdays 01.10.2018 - 31.03.2019",
                    "Median Wednesdays 01.10.2018 - 31.03.2019",
                    "Median Thursdays 01.10.2018 - 31.03.2019",
                    "Median Fridays 01.10.2018 - 31.03.2019",
                    "Median Saturdays 01.10.2018 - 31.03.2019",
                    "Median Sundays 01.10.2018 - 31.03.2019")

Stop_matrix <- cbind(Stop_matrix, median)

# Define the time periods for autumn, winter and spring
start_autumn <- "01.10.2018"
end_autumn <- "18.11.2018"
start_winter <- "19.11.2018"
end_winter <- "10.02.2019"
start_spring <- "11.02.2019"
end_spring <- "31.03.2019"

# Calculate the sum and the different medians for every coordinate and write them
  into the belonging columns
Stop_matrix[,grep("Sum", colnames(Stop_matrix))] <-
     apply(Stop_matrix[,4:(grep("Sum", colnames(Stop_matrix))-1)], 1, sum)
Stop_matrix[,grep("Median 01.10.2018 - 31.03.2019", colnames(Stop_matrix))] <-
     apply(Stop_matrix[,4:(grep("Sum", colnames(Stop_matrix))-1)], 1, median)
Stop_matrix[,grep("Median autumn 01.10.2018 - 18.11.2018", colnames(Stop_matrix))]
     <- apply(Stop_matrix[,grep(start_autumn,
     colnames(Stop_matrix))[1]:grep(end_autumn, colnames(Stop_matrix))[1]], 1,
     median)
Stop_matrix[,grep("Median winter 19.11.2018 - 10.02.2019", colnames(Stop_matrix))]
```

```
      <- apply(Stop_matrix[,grep(start_winter,
      colnames(Stop_matrix))[1]:grep(end_winter, colnames(Stop_matrix))[1]], 1,
      median)
Stop_matrix[,grep("Median spring 11.02.2019 - 31.03.2019", colnames(Stop_matrix))]
      <- apply(Stop_matrix[,grep(start_spring,
      colnames(Stop_matrix))[1]:grep(end_spring, colnames(Stop_matrix))[1]], 1,
      median)
Stop_matrix[,grep("Median Mondays 01.10.2018 - 31.03.2019", colnames(Stop_matrix))]
      <- apply(Stop_matrix[,seq(4, (grep("Sum", colnames(Stop_matrix))-1), by=7)],
      1, median)
Stop_matrix[,grep("Median Tuesdays 01.10.2018 - 31.03.2019",
      colnames(Stop_matrix))] <- apply(Stop_matrix[,seq(5, (grep("Sum",
      colnames(Stop_matrix))-1), by=7)], 1, median)
Stop_matrix[,grep("Median Wednesdays 01.10.2018 - 31.03.2019",
      colnames(Stop_matrix))] <- apply(Stop_matrix[,seq(6, (grep("Sum",
      colnames(Stop_matrix))-1), by=7)], 1, median)
Stop_matrix[,grep("Median Thursdays 01.10.2018 - 31.03.2019",
      colnames(Stop_matrix))] <- apply(Stop_matrix[,seq(7, (grep("Sum",
      colnames(Stop_matrix))-1), by=7)], 1, median)
Stop_matrix[,grep("Median Fridays 01.10.2018 - 31.03.2019", colnames(Stop_matrix))]
      <- apply(Stop_matrix[,seq(8, (grep("Sum", colnames(Stop_matrix))-1), by=7)],
      1, median)
Stop_matrix[,grep("Median Saturdays 01.10.2018 - 31.03.2019",
      colnames(Stop_matrix))] <- apply(Stop_matrix[,seq(9, (grep("Sum",
      colnames(Stop_matrix))-1), by=7)], 1, median)
Stop_matrix[,grep("Median Sundays 01.10.2018 - 31.03.2019", colnames(Stop_matrix))]
      <- apply(Stop_matrix[,seq(10, (grep("Sum", colnames(Stop_matrix))-1), by=7)],
      1, median)

# Calculate the sum of requests on one day
sum_days <- matrix(NA, 1, ncol(Stop_matrix))
colnames(sum_days) <- colnames(Stop_matrix)
Stop_matrix <- rbind(Stop_matrix, sum_days)
for(i in 4:ncol(Stop_matrix)) {
  Stop_matrix[nrow(Stop_matrix), i] <- sum(Stop_matrix[1:(nrow(Stop_matrix)-1),i])
}

# Write stop matrix to csv.file
write.csv2(Stop_matrix, file = "Frequency_monitor_table_stop.csv", row.names =
FALSE)

# The processing for creating the frequency table for position requests is equal
  to the stop requests
```

### Frequency plots: Medians stop requests, zoomed out

```
setwd(paste(working_directory, "Plots/Monitor/Frequency Plots/Medians stop
requests/Zoomed out", sep = ""))

Frequency_stop <- read.csv2(paste(working_directory, "Tables/Monitor/Frequency
                  Tables/Frequency_monitor_table_stop.csv", sep = ""),
                  stringsAsFactors = FALSE, check.names = FALSE)

# Read in a shape-file including the city districts of Graz
shape_Graz <- readShapeSpatial(paste(working_directory,
              "Data/Graz_city_districts.shp", sep = ""), proj4string =
              CRS("+proj=longlat"))

# Maximum value of the medians
max <- max(Frequency_stop[1:(nrow(Frequency_stop)-1), (grep("Sum",
       colnames(Frequency_stop))+1):ncol(Frequency_stop)])

TULogo <- readPNG(paste(working_directory, "Data/TULogo.png", sep = ""))

for(i in (grep("Sum", colnames(Frequency_stop))+1):ncol(Frequency_stop)) {

  # Take one column, delete the row including the sum and sort the table in
    decreasing order
  Frequency_stop_sorted <- cbind(Frequency_stop[,1], Frequency_stop[,2],
                           Frequency_stop[,i])
  Frequency_stop_sorted <- Frequency_stop_sorted[-c(nrow(Frequency_stop_sorted)),]
  Frequency_stop_sorted <-
       Frequency_stop_sorted[order(as.numeric(Frequency_stop_sorted[,3]),
       decreasing = TRUE), ]

  jpeg(filename = paste(i-186, ".Frequency_stop_requests_", gsub(" ", "", sub(" ",
  "_", sub(" ", "_", colnames(Frequency_stop)[i]))), ".jpg", sep = ""), width = 10,
  height = 7.6, units = "in", pointsize = 12, quality = 150, bg = "white", res =
  900, family = "", restoreConsole = TRUE, type = c("windows", "cairo"))

  # Define the edges of the plot and the size of the shape-file and plot the shape-
    file
  par(mgp = c(2,1,0), oma = c(0,0,2,0), pin = c(9,6))
  plot(xlim = c(15.33,15.60), ylim = NULL, shape_Graz, xaxt="n")

  # Maximum circle size
  cex_max <- 6

  # Maximum rounded value in the legend
  legend_max <- 1200

  # Add points of requests to the plot
  for(j in 1:nrow(Frequency_stop_sorted)) {
    if(Frequency_stop_sorted[j,3] != 0) {
      points(as.numeric(Frequency_stop_sorted[j,1]),
      as.numeric(Frequency_stop_sorted[j,2]), col = "black", bg = "#0080ff", cex =
      cex_max/sqrt(legend_max/as.numeric(Frequency_stop_sorted[j,3])), pch = 21)
    }
  }

  title(main = "Frequency stop requests", font.main = 2, cex.main = 1.5, line = 2)
  mtext(paste(colnames(Frequency_stop)[i], ", n = ",
  format(Frequency_stop[nrow(Frequency_stop),i], big.mark = ".", decimal.mark =
  ","), " [requests/d]", sep = ""), side = 3, cex = 0.9, line = 0.5)

  # Add a legend to the plot surrounded by a rectangle
  rect(15.53, 47.0285, 15.588, 47.078, col = "white", border = "black", lty =
  "solid", lwd = 1)
  text(15.533, 47.073, labels = "Legend", face = 2, cex = 1.5, pos = 4, offset = 0)

  legend(15.533, 47.070, legend = gsub(" ", "",
  format(c(legend_max/4,legend_max/2,legend_max), big.mark = ".", decimal.mark =
  ",")), pt.cex = c(cex_max/sqrt(4),cex_max/sqrt(2),cex_max), pch = 21, bty = "n",
  col = "black", pt.bg = "#0080ff", x.intersp = 2.3, y.intersp = c(1.05,1.35,1.64),
  title = "Stop requests    ")

  legend(15.533, 47.0372, legend = "Graz city districts", pch = 22, bty = "n",
  col = "black", seg.len = 2, pt.cex = 2)

  rasterImage(TULogo, 15.53,47.078,15.56,47.088)
  dev.off()
}
# The other frequency plots are created the same way
```

## Plots where requests are related to city districts: Medians stop requests

```
setwd(paste(working_directory, "Plots/Monitor/Shape Plots"))

Frequency_stop <- read.csv2(paste(working_directory, "Tables/Monitor/
                  Frequency Tables/Frequency_monitor_table_stop.csv", sep = ""),
                  stringsAsFactors = FALSE, check.names = FALSE)

# Read in stop coordinates related to Graz city districts
Frequency <- read.csv2(paste(working_directory, "Plots/Monitor/
             Shape Plots/Frequency_stop_coordinates_with_Graz_city_districts.csv",
             sep = ""), stringsAsFactors = FALSE, check.names = FALSE)

# Create a new column in the stop table for districts
districts <- matrix(NA, nrow(Frequency_stop), 1)
colnames(districts) <- "Districts"
Frequency_stop <- cbind(Frequency_stop[,1:3], districts,
                  Frequency_stop[,4:ncol(Frequency_stop)])

# Add number of city districts to stop table if requested stop belongs to Graz
for(i in 1:length(Frequency[,1])) {
  for(j in 1:length(Frequency_stop[,1])){
    if(identical(paste(Frequency[i,1], "/", Frequency[i,2], sep = ""),
    Frequency_stop[j,3]) == TRUE) {
      Frequency_stop[j,4] <- Frequency[i,3]
    }
  }
}

# Delete requests out of Graz and adjust the row numbers
```

```
Frequency_stop <- Frequency_stop[grep("[0-9]", Frequency_stop[,4]),]
rownames(Frequency_stop) <- c(1:nrow(Frequency_stop))

# Calculate the sum of requests on one day
sum_days <- matrix(NA, 1, ncol(Frequency_stop))
colnames(sum_days) <- colnames(Frequency_stop)
Frequency_stop <- rbind(Frequency_stop, sum_days)
for(i in 5:ncol(Frequency_stop)) {
  Frequency_stop[nrow(Frequency_stop), i] <-
       sum(Frequency_stop[1:(nrow(Frequency_stop)-1),i])
}

# Read in shape-file of Graz and add a new column to control the color
shape_Graz <- readShapeSpatial(paste(working_directory,
              "Data/Graz_city_districts.shp", sep = ""),
              proj4string = CRS("+proj=longlat"))
shape_Graz@data$DISTRICTS <- c(0)

TULogo <- readPNG(paste(working_directory, "Data/TULogo.png", sep = ""))

maximum <- matrix(NA,0,1)

for(i in (grep("Sum", colnames(Frequency_stop))+1):ncol(Frequency_stop)) {

  jpeg(filename = paste(i-187, ".Stop_requests_concentrated_to_districts_",
  gsub(" ", "", sub(" ", "_", sub(" ", "_", colnames(Frequency_stop)[i]))), ".jpg",
  sep = ""), width = 10, height = 7.6, units = "in", pointsize = 12, quality = 150,
  bg = "white", res = 300, family = "", restoreConsole = TRUE, type = c("windows",
  "cairo"))

  # Get the number of requests for every district
  for(k in 1:length(shape_Graz$NO)) {
    if(length(grep(paste("\\b", shape_Graz$NO[k], "\\b", sep = ""),
    Frequency_stop[,4])) > 0) {
      shape_Graz@data$DISTRICTS[k] <- sum(Frequency_stop[grep(paste("\\b",
                                     shape_Graz$NO[k], "\\b", sep = ""),
                                     Frequency_stop[,4]),i])
    }
  }

  # Get the maximum value of the medians
  maximum <- rbind(maximum, max(shape_Graz@data$DISTRICTS))

  # Create a matrix with place for 17 different colors
  color <- matrix(NA, length(shape_Graz$NO), 1)

  # Calculate the colors related to a maximum of 2300
  for(j in 1:length(color)) {
    color[j,] <- alpha("#005d7c", ((shape_Graz@data$DISTRICTS[j]*1)/2300))
  }

  # Plot the data
  plot(shape_Graz, col=color, border="black")

  title(main = "Stop requests concentrated to districts", font.main = 2, cex.main =
  1.5, line = 2)
  mtext(paste(colnames(Frequency_stop)[i], ", n = ",
  format(Frequency_stop[nrow(Frequency_stop),i], big.mark = ".", decimal.mark =
  ","), " [requests/d]", sep = ""), side = 3, cex = 0.9, line = 0.5)

  rect(15.514, 47.01, 15.5735, 47.078, col = "white", border = "black", lty =
  "solid", lwd = 1)
  text(15.517, 47.073, labels = "Legend", face = 2, cex = 1.5, pos = 4, offset = 0)
  text(15.517, 47.066, labels = "Stop requests", face = 2, cex = 1.1, pos = 4,
  offset = 0)

  # Include a color legend
  color_legend <- alpha("#005d7c", c(1:2300)/2300)
  legend.gradient(cbind(x = c(15.532, 15.542,15.542,15.532), y =
  c(47.06,47.06,47.02,47.02)), cols = color_legend, limits = c("0", "2.300"), title
  = "", cex = 1)

  legend(15.517, 47.019, legend = "Graz city districts", pch = 22, bty = "n", col =
  "black", seg.len = 2, pt.cex = 2)

  rasterImage(TULogo, 15.514,47.078,15.545,47.0885)
  dev.off()
}
```

# Appendix B – R-Code for data processing, route data

## Data preparation

```
setwd(paste(working_directory, "Tables/Route", sep = ""))

Route <- read.csv2(paste(working_directory, "Data/route.csv", sep = ""),
         stringsAsFactors = FALSE)

# Split the data in several parts if necessary for processing
Route <- Route[1:10000, ]

# Generate a total matrix out of the first column of Route
Matrix_tot <- Route[,1]

# Split column "Date Time" and add them to the total matrix
Matrix_tot <- cbind(Matrix_tot, matrix(unlist(strsplit(Route[,2], " ")),
              nrow=length(Route[,2]), byrow = TRUE), deparse.level = 0)

# Add columns "Time Zone", "Method", "Response Code" and "Bytes Sent" to the total
  matrix
Matrix_tot <- cbind(Matrix_tot, Route[,3], deparse.level = 0)
Matrix_tot <- cbind(Matrix_tot, Route[,4], deparse.level = 0)
Matrix_tot <- cbind(Matrix_tot, Route[,6], deparse.level = 0)
Matrix_tot <- cbind(Matrix_tot, Route[,7], deparse.level = 0)

# Adapt the seperators of the column "User Agent" to prepare the column for
  splitting
Route[,8] <- gsub(" ","",Route[,8], fixed=TRUE)
Route[,8] <- gsub("/",":",Route[,8], fixed=TRUE)
Route[,8] <- gsub("(",":",Route[,8], fixed=TRUE)
Route[,8] <- gsub(";)","",Route[,8], fixed=TRUE)
Route[,8] <- gsub(";",":",Route[,8], fixed=TRUE)
Route[,8] <- gsub(")","",Route[,8], fixed=TRUE)

# Create a new matrix for the splitted data
Matrix_user_agent <- matrix(NA, length(Route[,8]), 10)

# Split column "User Agent" and at them to the matrix created before
for (i in 1:length(Route[,8])) {

  if(identical(grep("iPhone", Route[i,8], fixed=TRUE), integer(0)) == FALSE ||
  identical(grep("iPad", Route[i,8], fixed=TRUE), integer(0)) == FALSE) {
    Vector_user_agent <- unlist(strsplit(Route[i,8], ":"))
    Matrix_user_agent[i,1] <- Vector_user_agent[1]
    Matrix_user_agent[i,2] <- Vector_user_agent[2]
    Matrix_user_agent[i,3] <- Vector_user_agent[3]
    Matrix_user_agent[i,4] <- Vector_user_agent[4]
    Matrix_user_agent[i,5] <- Vector_user_agent[6]
  }

  if(identical(grep("Android", Route[i,8], fixed=TRUE), integer(0)) == FALSE) {
    Vector_user_agent <- unlist(strsplit(Route[i,8], ":"))
    Matrix_user_agent[i,1] <- Vector_user_agent[1]
    Matrix_user_agent[i,2] <- Vector_user_agent[2]
    Matrix_user_agent[i,3] <- Vector_user_agent[3]
    Matrix_user_agent[i,4] <- Vector_user_agent[4]
    Matrix_user_agent[i,5] <- Vector_user_agent[6]
    Matrix_user_agent[i,6] <- Vector_user_agent[8]
    Matrix_user_agent[i,7] <- Vector_user_agent[9]
  }

  if(identical(grep("Web", Route[i,8], fixed=TRUE), integer(0)) == FALSE) {
    Vector_user_agent <- unlist(strsplit(Route[i,8], ":"))
    Matrix_user_agent[i,1] <- Vector_user_agent[1]
    Matrix_user_agent[i,2] <- Vector_user_agent[2]
    Matrix_user_agent[i,3] <- Vector_user_agent[3]
    Matrix_user_agent[i,4] <- Vector_user_agent[4]
  }

  if(identical(grep("Mozilla", Route[i,8], fixed=TRUE), integer(0)) == FALSE &&
  identical(grep("Trident", Route[i,8], fixed=TRUE), integer(0)) == FALSE) {
    Vector_user_agent <- unlist(strsplit(Route[i,8], ":"))
    Matrix_user_agent[i,1] <- Vector_user_agent[1]
    Matrix_user_agent[i,2] <- Vector_user_agent[2]
    Matrix_user_agent[i,3] <- Vector_user_agent[4]
    Matrix_user_agent[i,4] <- Vector_user_agent[5]
    Matrix_user_agent[i,10] <- Vector_user_agent[3]
    Matrix_user_agent[i,8] <- Vector_user_agent[7]
    Matrix_user_agent[i,9] <- Vector_user_agent[9]
  }

  if(identical(grep("Mozilla", Route[i,8], fixed=TRUE), integer(0)) == FALSE &&
  identical(grep("AppleWebKit", Route[i,8], fixed=TRUE), integer(0)) == FALSE) {
    Vector_user_agent <- unlist(strsplit(Route[i,8], ":"))
    Matrix_user_agent[i,1] <- Vector_user_agent[1]
    Matrix_user_agent[i,2] <- Vector_user_agent[2]
    Matrix_user_agent[i,3] <- Vector_user_agent[3]
    Matrix_user_agent[i,4] <- Vector_user_agent[4]
  }
}

# Add the matrix with the splitted data to the total matrix
Matrix_tot <- cbind(Matrix_tot, Matrix_user_agent, deparse.level = 0)

# Split "URL" and add it to Matrix_tot
Route[,5] <- gsub("%C3%84","Ä",Route[,5], fixed=TRUE)
Route[,5] <- gsub("%C3%A4","ä",Route[,5], fixed=TRUE)
Route[,5] <- gsub("%C3%96","Ö",Route[,5], fixed=TRUE)
Route[,5] <- gsub("%C3%B6","ö",Route[,5], fixed=TRUE)
Route[,5] <- gsub("%C3%9C","Ü",Route[,5], fixed=TRUE)
Route[,5] <- gsub("%C3%BC","ü",Route[,5], fixed=TRUE)
Route[,5] <- gsub("%C3%9F","ß",Route[,5], fixed=TRUE)

Route[,5] <- gsub("%3A",":",Route[,5], fixed=TRUE)
Route[,5] <- gsub("%2B","+",Route[,5], fixed=TRUE)
Route[,5] <- gsub("%2C",",",Route[,5], fixed=TRUE)
Route[,5] <- gsub("%20%20"," ",Route[,5], fixed=TRUE)
Route[,5] <- gsub("%20"," ",Route[,5], fixed=TRUE)
Route[,5] <- gsub("%28","(",Route[,5], fixed=TRUE)
Route[,5] <- gsub("%29",")",Route[,5], fixed=TRUE)
Route[,5] <- gsub("%C2%B4","´",Route[,5], fixed=TRUE)
Route[,5] <- gsub("%EF%BF%BD","§",Route[,5], fixed=TRUE)

# Split "URL" and add the variables to the url matrix
# Different processing for different platforms
Matrix_url <- matrix(NA, length(Route[,5]), 42)
for (i in 1:length(Route[,5])) {

  ## Android ##
  if(identical(grep("Android", Matrix_tot[i,10], fixed=TRUE), integer(0)) == FALSE)
  {

    if(identical(grep("/route?", Route[i,5], fixed=TRUE), integer(0)) == FALSE) {
      Matrix_url[i,1] <- "/route"
      Route[i,5] <- sub("/route?", "", Route[i,5], fixed=TRUE)
    }

    if(identical(grep("overrideFromType", Route[i,5], fixed=TRUE), integer(0)) ==
    FALSE) {
      Matrix_url[i,19] <- unlist(regmatches(Route[i,5],
                          regexec("overrideFromType=(.*?)&", Route[i,5])))[2]
      Route[i,5] <- gsub(gsub(" ", "", paste("overrideFromType=", Matrix_url[i,19],
                    "&"), fixed=TRUE), "", Route[i,5], fixed=TRUE)
    }

    if(substr(Route[i,5],6,6) != "-" && as.numeric(substr(Route[i,5],6,6)) == 6) {
      Matrix_url[i,2] <- unlist(regmatches(Route[i,5], regexec("from=(.*?)&",
                         Route[i,5])))[2]
      Route[i,5] <- sub(gsub(" ", "", paste("from=", Matrix_url[i,2], "&"),
                    fixed=TRUE), "", Route[i,5], fixed=TRUE)
    } else {
      Matrix_url[i,3] <- unlist(regmatches(Route[i,5], regexec("from=(.*?):",
                         Route[i,5])))[2]
      Route[i,5] <- sub(gsub(" ", "", paste("from=", Matrix_url[i,3]), fixed=TRUE),
                    "", Route[i,5], fixed=TRUE)
      Matrix_url[i,4] <- unlist(regmatches(Route[i,5], regexec(":(.*?):",
                         Route[i,5])))[2]
      Route[i,5] <- sub(gsub(" ", "", paste(":", Matrix_url[i,4]), fixed=TRUE), "",
                    Route[i,5], fixed=TRUE)
      Matrix_url[i,7] <- unlist(regmatches(Route[i,5], regexec(":(.*?):",
                         Route[i,5])))[2]
      Route[i,5] <- sub(gsub(" ", "", paste(":", Matrix_url[i,7]), fixed=TRUE), "",
                   Route[i,5], fixed=TRUE)
    Matrix_url[i,5] <- unlist(regmatches(Route[i,5], regexec(":(.*?):",
                         Route[i,5])))[2]
    Route[i,5] <- sub(gsub(" ", "", paste(":", Matrix_url[i,5]), fixed=TRUE), "",
                   Route[i,5], fixed=TRUE)
    Matrix_url[i,6] <- unlist(regmatches(Route[i,5], regexec(":(.*?)&",
                         Route[i,5])))[2]
    Route[i,5] <- sub(gsub(" ", "", paste(":", Matrix_url[i,6], "&"),
                   fixed=TRUE), "", Route[i,5], fixed=TRUE)
  }

  if(substr(Route[i,5],4,4) != "-" && as.numeric(substr(Route[i,5],4,4)) == 6) {
    Matrix_url[i,8] <- unlist(regmatches(Route[i,5], regexec("to=(.*?)&",
                         Route[i,5])))[2]
    Route[i,5] <- sub(gsub(" ", "", paste("to=", Matrix_url[i,8], "&"),
                   fixed=TRUE), "", Route[i,5], fixed=TRUE)
  } else {
    Matrix_url[i,9] <- unlist(regmatches(Route[i,5], regexec("to=(.*?):",
                         Route[i,5])))[2]
    Route[i,5] <- sub(gsub(" ", "", paste("to=", Matrix_url[i,9]), fixed=TRUE),
                   "", Route[i,5], fixed=TRUE)
    Matrix_url[i,10] <- unlist(regmatches(Route[i,5], regexec(":(.*?):",
                          Route[i,5])))[2]
    Route[i,5] <- sub(gsub(" ", "", paste(":", Matrix_url[i,10]), fixed=TRUE),
                   "", Route[i,5], fixed=TRUE)
    Matrix_url[i,13] <- unlist(regmatches(Route[i,5], regexec(":(.*?):",
                          Route[i,5])))[2]
    Route[i,5] <- sub(gsub(" ", "", paste(":", Matrix_url[i,13]), fixed=TRUE),
                   "", Route[i,5], fixed=TRUE)
    Matrix_url[i,11] <- unlist(regmatches(Route[i,5], regexec(":(.*?):",
                          Route[i,5])))[2]
    Route[i,5] <- sub(gsub(" ", "", paste(":", Matrix_url[i,11]), fixed=TRUE),
                   "", Route[i,5], fixed=TRUE)
    Matrix_url[i,12] <- unlist(regmatches(Route[i,5], regexec(":(.*?)&",
                          Route[i,5])))[2]
    Route[i,5] <- sub(gsub(" ", "", paste(":", Matrix_url[i,12], "&"),
                   fixed=TRUE), "", Route[i,5], fixed=TRUE)
  }

  if(identical(grep("date=", Route[i,5], fixed=TRUE), integer(0)) == FALSE) {
    Matrix_url[i,14] <- unlist(regmatches(Route[i,5], regexec("date=(.*?)&",
                          Route[i,5])))[2]
    Route[i,5] <- sub(gsub(" ", "", paste("date=", Matrix_url[i,14], "&"),
                   fixed=TRUE), "", Route[i,5], fixed=TRUE)
  }

  if(identical(grep("aPP=", Route[i,5], fixed=TRUE), integer(0)) == FALSE) {
    Matrix_url[i,21] <- unlist(regmatches(Route[i,5], regexec("aPP=(.*?)&",
                          Route[i,5])))[2]
    Route[i,5] <- sub(gsub(" ", "", paste("aPP=", Matrix_url[i,21], "&"),
                   fixed=TRUE), "", Route[i,5], fixed=TRUE)
  }

  if(identical(grep("aflT=", Route[i,5], fixed=TRUE), integer(0)) == FALSE) {
    Matrix_url[i,23] <- unlist(regmatches(Route[i,5], regexec("aflT=(.*?)&",
                          Route[i,5])))[2]
    Route[i,5] <- sub(gsub(" ", "", paste("aflT=", Matrix_url[i,23], "&"),
                   fixed=TRUE), "", Route[i,5], fixed=TRUE)
  }

  if(identical(grep("deparr=", Route[i,5], fixed=TRUE), integer(0)) == FALSE) {
    Matrix_url[i,24] <- unlist(regmatches(Route[i,5], regexec("deparr=(.*?)&",
                          Route[i,5])))[2]
    Route[i,5] <- sub(gsub(" ", "", paste("deparr=", Matrix_url[i,24], "&"),
                   fixed=TRUE), "", Route[i,5], fixed=TRUE)
  }

  if(identical(grep("ptRO=", Route[i,5], fixed=TRUE), integer(0)) == FALSE) {
    Matrix_url[i,28] <- unlist(regmatches(Route[i,5], regexec("ptRO=(.*?)&",
                          Route[i,5])))[2]
    Route[i,5] <- sub(gsub(" ", "", paste("ptRO=", Matrix_url[i,28], "&"),
                   fixed=TRUE), "", Route[i,5], fixed=TRUE)
  }

  if(identical(grep("aSS=", Route[i,5], fixed=TRUE), integer(0)) == FALSE) {
    Matrix_url[i,22] <- unlist(regmatches(Route[i,5], regexec("aSS=(.*?)&",
                          Route[i,5])))[2]
    Route[i,5] <- sub(gsub(" ", "", paste("aSS=", Matrix_url[i,22], "&"),
                  fixed=TRUE), "", Route[i,5], fixed=TRUE)
    }

    if(identical(grep("ptMWT=", Route[i,5], fixed=TRUE), integer(0)) == FALSE) {
      Matrix_url[i,26] <- unlist(regmatches(Route[i,5], regexec("ptMWT=(.*?)&",
                          Route[i,5])))[2]
      Route[i,5] <- sub(gsub(" ", "", paste("ptMWT=", Matrix_url[i,26], "&"),
                  fixed=TRUE), "", Route[i,5], fixed=TRUE)
    }

    if(identical(grep("walkMT=", Route[i,5], fixed=TRUE), integer(0)) == FALSE) {
      Matrix_url[i,25] <- unlist(regmatches(Route[i,5], regexec("walkMT=(.*?)&",
                          Route[i,5])))[2]
      Route[i,5] <- sub(gsub(" ", "", paste("walkMT=", Matrix_url[i,25], "&"),
                  fixed=TRUE), "", Route[i,5], fixed=TRUE)
    }

    if(identical(grep("ptWS=", Route[i,5], fixed=TRUE), integer(0)) == FALSE) {
      Matrix_url[i,27] <- unlist(regmatches(Route[i,5], regexec("ptWS=(.*?)&",
                          Route[i,5])))[2]
      Route[i,5] <- sub(gsub(" ", "", paste("ptWS=", Matrix_url[i,27], "&"),
                  fixed=TRUE), "", Route[i,5], fixed=TRUE)
    }

    for (j in 32:41) {

      if(identical(grep("ptV=", Route[i,5], fixed=TRUE), integer(0)) == FALSE) {
        Matrix_url[i,j] <- unlist(regmatches(Route[i,5], regexec("ptV=(.*?)&",
                           Route[i,5])))[2]
        Route[i,5] <- sub(gsub(" ", "", paste("ptV=", Matrix_url[i,j], "&"),
                    fixed=TRUE), "", Route[i,5], fixed=TRUE)
      }
    }

    if(identical(grep("prevNext=", Route[i,5], fixed=TRUE), integer(0)) == FALSE) {
      Matrix_url[i,17] <- unlist(regmatches(Route[i,5], regexec("prevNext=(.*?)&",
                          Route[i,5])))[2]
      Route[i,5] <- sub(gsub(" ", "", paste("prevNext=", Matrix_url[i,17], "&"),
                  fixed=TRUE), "", Route[i,5], fixed=TRUE)
    }

    if(identical(grep("sessionId=", Route[i,5], fixed=TRUE), integer(0)) == FALSE)
    {
      Matrix_url[i,16] <- unlist(regmatches(Route[i,5], regexec("sessionId=(.*?)&",
                          Route[i,5])))[2]
      Route[i,5] <- sub(gsub(" ", "", paste("sessionId=", Matrix_url[i,16], "&"),
                  fixed=TRUE), "", Route[i,5], fixed=TRUE)
    }

    if(identical(grep("ptMC=", Route[i,5], fixed=TRUE), integer(0)) == FALSE) {
      Matrix_url[i,29] <- unlist(regmatches(Route[i,5], regexec("ptMC=(.*?)&",
                          Route[i,5])))[2]
      Route[i,5] <- sub(gsub(" ", "", paste("ptMC=", Matrix_url[i,29], "&"),
                  fixed=TRUE), "", Route[i,5], fixed=TRUE)
    }

    for (j in 30:31) {

      if(identical(grep("ptMobC=", Route[i,5], fixed=TRUE), integer(0)) == FALSE) {
        Matrix_url[i,j] <- unlist(regmatches(Route[i,5], regexec("ptMobC=(.*?)&",
                           Route[i,5])))[2]
        Route[i,5] <- sub(gsub(" ", "", paste("ptMobC=", Matrix_url[i,j], "&"),
                    fixed=TRUE), "", Route[i,5], fixed=TRUE)
      }
    }

    if(identical(grep("version=", Route[i,5], fixed=TRUE), integer(0)) == FALSE) {
      Matrix_url[i,15] <- unlist(regmatches(Route[i,5], regexec("version=(.*?)$",
                          Route[i,5])))[2]
      Route[i,5] <- sub(gsub(" ", "", paste("version=", Matrix_url[i,15]),
                  fixed=TRUE), "", Route[i,5], fixed=TRUE)
    }
  }
}
# The sorting of the other platforms is similar to the Android platform

# Add url matrix with variables from "URL" to the total matrix
Matrix_tot <- cbind(Matrix_tot, Matrix_url)
```

```
# Change signs with special characters
Matrix_tot <- gsub("°", " ", Matrix_tot, fixed=TRUE)
Matrix_tot <- gsub("++", " ", Matrix_tot, fixed=TRUE)
Matrix_tot[,31] <- gsub("+", "~", Matrix_tot[,31], fixed=TRUE)
Matrix_tot[,9] <- gsub("+", "~", Matrix_tot[,9], fixed=TRUE)
Matrix_tot <- gsub("+", " ", Matrix_tot, fixed=TRUE)
Matrix_tot[,31] <- gsub("~", "+", Matrix_tot[,31], fixed=TRUE)
Matrix_tot[,9] <- gsub("~", "+", Matrix_tot[,9], fixed=TRUE)

# Change colnames of the total matrix
colnames(Matrix_tot) <- c("User","Date","Time","Time zone","Method",
     "Response Code","Bytes Sent","App","App Version","Platform",
     "Platform Version","Scale","Size Heigth","Size Width","comp","Trident",
     "Trident 2","Request","From Diva","From X", "From Y","Address From",
     "Address From 2","KS From","To Diva","To X","To Y","Address To",
     "Address To 2","KS To","date_time","version","sessionId","prevNext",
     "modality","overrideFromType","overrideToType","aPP","aSS","aflT","deparr",
     "walkMT","ptMWT","ptWS","ptRO","ptMC","ptMobC","ptMobC 2","ptV","ptV 2",
     "ptV 3","ptV 4","ptV 5","ptV 6","ptV 7","ptV 8","ptV 9","ptV 10","_")

Route <- Matrix_tot

# Create a vector in which the weekdays to the date are stored
Weekday <- matrix(NA, length(Route[,2]), 1)
for (i in 1:length(Route[,2])) {
  Weekday[i,1] <- weekdays(as.Date(Route[i,2], "%d.%m.%Y"))
}

# Split the column "Date Time" into date, time and time zone
date_time <- matrix(NA, length(Route[,31]), 3)
for(i in 1:length(Route[,31])) {

  if(is.na(Route[i,31]) == FALSE) {

    date_time[i,1] <- paste(unlist(regmatches(unlist(regmatches(Route[i,31],
         regexec("-(.*?)T",Route[i,31])))[2], regexec("-(.*?)$",
         unlist(regmatches(Route[i,31],regexec("-(.*?)T",Route[i,31])))[2])))[2],
         ".", unlist(regmatches(Route[i,31],regexec("-(.*?)-",Route[i,31])))[2],
         ".", unlist(regmatches(Route[i,31],regexec("^(.*?)-",Route[i,31])))[2],
         sep = "")

    Route[i,31] <- gsub(".","~",Route[i,31], fixed=TRUE)
    date_time[i,2] <- unlist(regmatches(Route[i,31], regexec("T(.*?)~",
                      Route[i,31])))[2]
    Route[i,31] <- gsub("+","#",Route[i,31], fixed=TRUE)
    date_time[i,3] <- paste("GMT", unlist(regmatches(Route[i,31],
                      regexec("#(.*?)$", Route[i,31])))[2])
  }
}

# Change the colnames of date time
colnames(date_time) <- c("Date2","Time2","Timezone2")

# Add date time to the Route table
Route <- cbind(Route[,1:2], Weekday, Route[,3:30], date_time, Route[,32:59])

#
## Reduce Route table to required columns ##

# Read in stops of Graz with coordinates
Diva_coor <- read.csv2(paste(working_directory, "Data/Graz_stop_coordinates.csv",
             sep = ""), stringsAsFactors = FALSE)

# Create a new table with the required colums for evaluation
Route_req <- cbind(Route[,3], Route[,2], Route[,4], Route[,7], Route[,11],
             Route[,20], Route[,21], Route[,22], Route[,26], Route[,27],
             Route[,28], Route[,32], Route[,33], Route[,36])

# Change the colnames of the new table
colnames(Route_req) <- c("Date Request","Weekday Request","Time Request",
                       "Response Code","Platform","From Diva","From X","From Y",
                       "To Diva","To X","To Y","Date Ride","Time Ride",
                       "Session ID")

# Delete response code unequal 200 (200 = request successful)
RC_del <- matrix(NA, 0, 1)
for(i in 1:length(Route_req[,4])) {
  if(Route_req[i,4] == 404 || Route_req[i,4] == 500 || Route_req[i,4] == 503) {
    RC_del <- rbind(RC_del, i)
  }
}
if(length(RC_del) > 0) {
  Route_req <- Route_req[-c(RC_del),]
}

# Delete column "Response Code"
Route_req <- Route_req[,-c(4)]


# Delete all requests with an Session ID
SessionID_del <- grep("[0-9]", Route_req[,13])
if(length(SessionID_del) > 0) {
  Route_req <- Route_req[-c(SessionID_del),]
}

# Delete column "Session ID"
Route_req <- Route_req[,-c(13)]

# Date and Time exchange; Use date and time from the moment of the ride if
  available
overall_border_low <- as.integer(as.POSIXct("01.10.2018 00:00:00", tryFormats =
                        c("%d.%m.%Y %H:%M:%S")))
overall_border_high <- as.integer(as.POSIXct("31.03.2019 23:59:59", tryFormats =
                         c("%d.%m.%Y %H:%M:%S")))

row_del <- matrix(NA, 0, 1)

for(i in 1:length(Route_req[,11])) {

  if(is.na(Route_req[i,11]) == FALSE) {

    # If the request falls into the hour of clock changing in spring then the time
      is shifted one hour forwards
    if(Route_req[i,11] == "31.03.2019" && unlist(regmatches(Route_req[i,12],
    regexec("(.*?):", Route_req[i,12])))[2] == "02") {
      Route_req[i,12] <- paste("03:", unlist(regmatches(Route_req[i,12],
                           regexec(":(.*?)$", Route_req[i,12])))[2], sep = "")
    }

    # If the desired time of travel is later than the time of request then date and
      time are used from time of travel
    if(as.integer(as.POSIXct(paste(Route_req[i,11], Route_req[i,12]), tryFormats =
    c("%d.%m.%Y %H:%M:%S"))) > as.integer(as.POSIXct(paste(Route_req[i,2],
    Route_req[i,3]), tryFormats = c("%d.%m.%Y %H:%M:%S")))) {
      Route_req[i,2] <- Route_req[i,11]
      Route_req[i,3] <- Route_req[i,12]
    }

    # If the time of request lays outside of the borders then these rows are
      deleted
    if(as.integer(as.POSIXct(paste(Route_req[i,11], Route_req[i,12]), tryFormats =
    c("%d.%m.%Y %H:%M:%S"))) < overall_border_low ||
    as.integer(as.POSIXct(paste(Route_req[i,11], Route_req[i,12]), tryFormats =
    c("%d.%m.%Y %H:%M:%S"))) > overall_border_high) {
      row_del <- rbind(row_del, i)
    }
  }
}

# Delete the rows outside of the borders
if(identical(row_del[,1], logical(0)) == FALSE) {
  Route_req <- Route_req[-c(row_del),]
}

# Delete columns "Date Ride" and "Time Ride"
Route_req <- Route_req[,-c(11:12)]

#
## Coordinates for Divas ##

# Get the coordinates for the divas out of the table including the stops of Graz
  with coordinates (Diva_coor)
for(i in 1:length(Route_req[,5])) {
  if(is.na(Route_req[i,5]) == FALSE && length(grep(Route_req[i,5], Diva_coor[,1],
  fixed = TRUE)) > 0) {
    row_nr <- grep(Route_req[i,5], Diva_coor[,1], fixed = TRUE)
    Route_req[i,6] <- Diva_coor[row_nr,3]
    Route_req[i,7] <- Diva_coor[row_nr,4]
  }

  if(is.na(Route_req[i,8]) == FALSE && length(grep(Route_req[i,8], Diva_coor[,1],
  fixed = TRUE)) > 0) {
    row_nr <- grep(Route_req[i,8], Diva_coor[,1], fixed = TRUE)
    Route_req[i,9] <- Diva_coor[row_nr,3]
    Route_req[i,10] <- Diva_coor[row_nr,4]
  }
}

# Delete the colums "From Diva" and "To Diva"
Route_req <- Route_req[,-c(5,8)]

# Delete rows with NA
rows_NA <- matrix(NA, 0, 1)
for(i in 1:length(Route_req[,1])) {

  if(sum(is.na(Route_req[i,])) > 0) {
    rows_NA <- rbind(rows_NA, i)
  }
}
if(length(rows_NA) > 0) {
  Route_req <- Route_req[c(-rows_NA),]
}

# Sort table to date and time in increasing order
Route_req <- cbind(Route_req, matrix(NA, length(Route_req[,2]), 1))
for(i in 1:length(Route_req[,2])) {
  Route_req[i,9] <- as.integer(as.POSIXct(paste(Route_req[i,2], Route_req[i,3]),
                    tryFormats = c("%d.%m.%Y %H:%M:%S")))
}
Route_req <- Route_req[order(Route_req[,9], decreasing = FALSE),]

# Delete the just now created column
Route_req <- Route_req[,-c(9)]

# Update the weekdays
for(i in 1:length(Route_req[,2])) {
  Route_req[i,1] <- weekdays(as.Date(Route_req[i,2], "%d.%m.%Y"))
}

# Change the colnames of the route table
colnames(Route_req) <- c("Weekday", "Date", "Time", "Platform", "From X", "From Y",
                         "To X", "To y")

# Write the sorted table to csv.file
write.csv2(Route_req, file = "Routes_evaluation_start_file.csv", row.names = FALSE)
```

## Distribution tables

```
# Set the working directory in which the exported files are stored
setwd(paste(working_directory, "Tables/Route/Distribution Tables", sep = ""))

# Read in sorted route table
Route <- read.csv2(paste(working_directory,
         "Tables/Monitor/Routes_evaluation_start_file.csv", sep = ""),
         stringsAsFactors = FALSE)

# Use first element of date
Route[,3] <- unlist(regmatches(Route[c(1:nrow(Route)),3], regexec("(.*?):",
             Route[c(1:nrow(Route)),3])))[c(seq(2, nrow(Route)*2, by = 2))]

# Build a contingency table of the frequencies between time and date
Frequency_table <- table(Route[,3], as.integer(as.POSIXct(Route[,2], tryFormats =
                   c("%d.%m.%Y"))))
Frequency_table <- as.data.frame.matrix(Frequency_table)
Frequency_table <- cbind(matrix(NA, nrow(Frequency_table),1), Frequency_table)
rownames(Frequency_table) <- c(1:nrow(Frequency_table))
Frequency_table[,1] <- rownames(Frequency_table)

# Change the column name of the first column
colnames(Frequency_table)[1] <- c("Hours")

# Change the format of date and replace it in the column names
```

```
date <- as.POSIXct(as.integer(colnames(Frequency_table)[2:ncol(Frequency_table)]),
        tryFormats = c("%d.%m.%Y"), origin = "01.01.1970")
class(date) = c('POSIXct')
date <- as.character.Date(date, tryFormats = c("%d.%m.%Y"))
date <- format(as.Date(date), "%d.%m.%Y")
colnames(Frequency_table)[2:ncol(Frequency_table)] <- date

# Add sum and medians to the frequency table
median <- matrix(NA, nrow(Frequency_table), 12)
colnames(median) <- c("Sum",
                      "Median 01.10.2018 - 31.03.2019",
                      "Median autumn 01.10.2018 - 18.11.2018",
                      "Median winter 19.11.2018 - 10.02.2019",
                      "Median spring 11.02.2019 - 31.03.2019",
                      "Median Mondays 01.10.2018 - 31.03.2019",
                      "Median Tuesdays 01.10.2018 - 31.03.2019",
                      "Median Wednesdays 01.10.2018 - 31.03.2019",
                      "Median Thursdays 01.10.2018 - 31.03.2019",
                      "Median Fridays 01.10.2018 - 31.03.2019",
                      "Median Saturdays 01.10.2018 - 31.03.2019",
                      "Median Sundays 01.10.2018 - 31.03.2019")

Frequency_table <- cbind(Frequency_table, median)

# Define the time periods for autumn, winter and spring
start_autumn <- "01.10.2018"
end_autumn <- "18.11.2018"
start_winter <- "19.11.2018"
end_winter <- "10.02.2019"
start_spring <- "11.02.2019"
end_spring <- "31.03.2019"

# Calculate the sum and the different medians for every hour and write them into
the belonging columns
Frequency_table[,grep("Sum", colnames(Frequency_table))] <-
     apply(Frequency_table[,2:(grep("Sum", colnames(Frequency_table))-1)], 1, sum)
Frequency_table[,grep("Median 01.10.2018 - 31.03.2019", colnames(Frequency_table))]
     <- apply(Frequency_table[,2:(grep("Sum", colnames(Frequency_table))-1)], 1,
     median)
Frequency_table[,grep("Median autumn 01.10.2018 - 18.11.2018",
     colnames(Frequency_table))] <- apply(Frequency_table[,grep(start_autumn,
     colnames(Frequency_table))[1]:grep(end_autumn, colnames(Frequency_table))[1]],
     1, median)
Frequency_table[,grep("Median winter 19.11.2018 - 10.02.2019",
     colnames(Frequency_table))] <- apply(Frequency_table[,grep(start_winter,
     colnames(Frequency_table))[1]:grep(end_winter, colnames(Frequency_table))[1]],
     1, median)
Frequency_table[,grep("Median spring 11.02.2019 - 31.03.2019",
     colnames(Frequency_table))] <- apply(Frequency_table[,grep(start_spring,
     colnames(Frequency_table))[1]:grep(end_spring, colnames(Frequency_table))[1]],
     1, median)
Frequency_table[,grep("Median Mondays 01.10.2018 - 31.03.2019",
     colnames(Frequency_table))] <- apply(Frequency_table[,seq(2, (grep("Sum",
     colnames(Frequency_table))-1), by=7)], 1, median)
Frequency_table[,grep("Median Tuesdays 01.10.2018 - 31.03.2019",
     colnames(Frequency_table))] <- apply(Frequency_table[,seq(3, (grep("Sum",
     colnames(Frequency_table))-1), by=7)], 1, median)
Frequency_table[,grep("Median Wednesdays 01.10.2018 - 31.03.2019",
     colnames(Frequency_table))] <- apply(Frequency_table[,seq(4, (grep("Sum",
     colnames(Frequency_table))-1), by=7)], 1, median)
Frequency_table[,grep("Median Thursdays 01.10.2018 - 31.03.2019",
     colnames(Frequency_table))] <- apply(Frequency_table[,seq(5, (grep("Sum",
     colnames(Frequency_table))-1), by=7)], 1, median)
Frequency_table[,grep("Median Fridays 01.10.2018 - 31.03.2019",
     colnames(Frequency_table))] <- apply(Frequency_table[,seq(6, (grep("Sum",
     colnames(Frequency_table))-1), by=7)], 1, median)
Frequency_table[,grep("Median Saturdays 01.10.2018 - 31.03.2019",
     colnames(Frequency_table))] <- apply(Frequency_table[,seq(7, (grep("Sum",
     colnames(Frequency_table))-1), by=7)], 1, median)
Frequency_table[,grep("Median Sundays 01.10.2018 - 31.03.2019",
     colnames(Frequency_table))] <- apply(Frequency_table[,seq(8, (grep("Sum",
     colnames(Frequency_table))-1), by=7)], 1, median)

# Calculate the sum of requests on one day
sum_days <- matrix(NA, 1, ncol(Frequency_table))
colnames(sum_days) <- colnames(Frequency_table)
Frequency_table <- rbind(Frequency_table, sum_days)
for(i in 2:ncol(Frequency_table)) {
  Frequency_table[nrow(Frequency_table), i] <-
       sum(Frequency_table[1:(nrow(Frequency_table)-1),i])
}

# Write frequency monitor table to csv.file
write.csv2(Frequency_table, file = "Distribution_table.csv", row.names = FALSE)

# The distribution plots are created the same way as for the monitor requests
```

## Frequency tables

```
#
## Route from table ######

setwd(paste(working_directory, "Tables/Route/Frequency Tables", sep = ""))

# Read in sorted route table
Route <- read.csv2(paste(working_directory,
         "Tables/Monitor/Routes_evaluation_start_file.csv", sep = ""),
         stringsAsFactors = FALSE, check.names = FALSE)

# Build a contingency table of the frequencies between coordinates and date
Frequency_route_from <- table(paste(Route[,5], "/", Route[,6], sep = ""),
                        as.integer(as.POSIXct(Route[,2], tryFormats =
                        c("%d.%m.%Y"))))
Frequency_route_from <- as.data.frame.matrix(Frequency_route_from)
Frequency_route_from <- cbind(matrix(NA, nrow(Frequency_route_from),1),
                        Frequency_route_from)
Frequency_route_from[,1] <- rownames(Frequency_route_from)
rownames(Frequency_route_from) <- c(1:nrow(Frequency_route_from))
Frequency_route_from <- cbind(matrix(NA, nrow(Frequency_route_from), 2),
                        Frequency_route_from)

# Split the coordinates into x and y coordinates
Frequency_route_from[,1] <- unlist(regmatches(Frequency_route_from[,3],
                            regexec("^(.*?)/", Frequency_route_from[,3])))[seq(1,
                            nrow(Frequency_route_from)*2, 2)+1]
Frequency_route_from[,2] <- unlist(regmatches(Frequency_route_from[,3],
                            regexec("/(.*?)$", Frequency_route_from[,3])))[seq(1,
                            nrow(Frequency_route_from)*2, 2)+1]

# Change the column names of the first three columns
colnames(Frequency_route_from)[1:3] <- c("From X", "From Y", "X/Y")

# Change the format of date and replace it in the column names
date <- as.POSIXct(as.integer(colnames(Frequency_route_from)[4:ncol(
        Frequency_route_from)]), tryFormats = c("%d.%m.%Y"), origin = "01.01.1970")
class(date) = c('POSIXct')
date <- as.character.Date(date, tryFormats = c("%d.%m.%Y"))
date <- format(as.Date(date), "%d.%m.%Y")
colnames(Frequency_route_from)[4:ncol(Frequency_route_from)] <- date

# Add sum and medians to the route from table
median <- matrix(NA, nrow(Frequency_route_from), 12)
colnames(median) <- c("Sum",
                    "Median 01.10.2018 - 31.03.2019",
                    "Median autumn 01.10.2018 - 18.11.2018",
                    "Median winter 19.11.2018 - 10.02.2019",
                    "Median spring 11.02.2019 - 31.03.2019",
                    "Median Mondays 01.10.2018 - 31.03.2019",
                    "Median Tuesdays 01.10.2018 - 31.03.2019",
                    "Median Wednesdays 01.10.2018 - 31.03.2019",
                    "Median Thursdays 01.10.2018 - 31.03.2019",
                    "Median Fridays 01.10.2018 - 31.03.2019",
                    "Median Saturdays 01.10.2018 - 31.03.2019",
                    "Median Sundays 01.10.2018 - 31.03.2019")

Frequency_route_from <- cbind(Frequency_route_from, median)

# Define the time periods for autumn, winter and spring
start_autumn <- "01.10.2018"
end_autumn <- "18.11.2018"
start_winter <- "19.11.2018"
end_winter <- "10.02.2019"
start_spring <- "11.02.2019"
end_spring <- "31.03.2019"
```

```
# Calculate the sum and the different medians for every coordinate and write them
  into the belonging columns
Frequency_route_from[,grep("Sum", colnames(Frequency_route_from))] <-
     apply(Frequency_route_from[,4:(grep("Sum", colnames(Frequency_route_from))-
     1)], 1, sum)
Frequency_route_from[,grep("Median 01.10.2018 - 31.03.2019",
     colnames(Frequency_route_from))] <- apply(Frequency_route_from[,4:(grep("Sum",
     colnames(Frequency_route_from))-1)], 1, median)
Frequency_route_from[,grep("Median autumn 01.10.2018 - 18.11.2018",
     colnames(Frequency_route_from))] <-
     apply(Frequency_route_from[,grep(start_autumn,
     colnames(Frequency_route_from))[1]:grep(end_autumn,
     colnames(Frequency_route_from))[1]], 1, median)
Frequency_route_from[,grep("Median winter 19.11.2018 - 10.02.2019",
     colnames(Frequency_route_from))] <-
     apply(Frequency_route_from[,grep(start_winter,
     colnames(Frequency_route_from))[1]:grep(end_winter,
     colnames(Frequency_route_from))[1]], 1, median)
Frequency_route_from[,grep("Median spring 11.02.2019 - 31.03.2019",
     colnames(Frequency_route_from))] <-
     apply(Frequency_route_from[,grep(start_spring,
     colnames(Frequency_route_from))[1]:grep(end_spring,
     colnames(Frequency_route_from))[1]], 1, median)
Frequency_route_from[,grep("Median Mondays 01.10.2018 - 31.03.2019",
     colnames(Frequency_route_from))] <- apply(Frequency_route_from[,seq(4,
     (grep("Sum", colnames(Frequency_route_from))-1), by=7)], 1, median)
Frequency_route_from[,grep("Median Tuesdays 01.10.2018 - 31.03.2019",
     colnames(Frequency_route_from))] <- apply(Frequency_route_from[,seq(5,
     (grep("Sum", colnames(Frequency_route_from))-1), by=7)], 1, median)
Frequency_route_from[,grep("Median Wednesdays 01.10.2018 - 31.03.2019",
     colnames(Frequency_route_from))] <- apply(Frequency_route_from[,seq(6,
     (grep("Sum", colnames(Frequency_route_from))-1), by=7)], 1, median)
Frequency_route_from[,grep("Median Thursdays 01.10.2018 - 31.03.2019",
     colnames(Frequency_route_from))] <- apply(Frequency_route_from[,seq(7,
     (grep("Sum", colnames(Frequency_route_from))-1), by=7)], 1, median)
Frequency_route_from[,grep("Median Fridays 01.10.2018 - 31.03.2019",
     colnames(Frequency_route_from))] <- apply(Frequency_route_from[,seq(8,
     (grep("Sum", colnames(Frequency_route_from))-1), by=7)], 1, median)
Frequency_route_from[,grep("Median Saturdays 01.10.2018 - 31.03.2019",
     colnames(Frequency_route_from))] <- apply(Frequency_route_from[,seq(9,
     (grep("Sum", colnames(Frequency_route_from))-1), by=7)], 1, median)
Frequency_route_from[,grep("Median Sundays 01.10.2018 - 31.03.2019",
     colnames(Frequency_route_from))] <- apply(Frequency_route_from[,seq(10,
     (grep("Sum", colnames(Frequency_route_from))-1), by=7)], 1, median)

# Calculate the sum of requests on one day
sum_days <- matrix(NA, 1, ncol(Frequency_route_from))
colnames(sum_days) <- colnames(Frequency_route_from)
Frequency_route_from <- rbind(Frequency_route_from, sum_days)
for(i in 4:ncol(Frequency_route_from)) {
  Frequency_route_from[nrow(Frequency_route_from), i] <-
       sum(Frequency_route_from[1:(nrow(Frequency_route_from)-1),i])
}

# Write frequency route from table to csv.file
write.csv2(Frequency_route_from, file = "Frequency_table_route_from.csv", row.names
= FALSE)

# The frequency route to table is created the same way as the route from table and
  the frequency plots are created the same way as for the monitor requests
```

## Spider matrix

```
setwd(paste(working_directory, "Tables/Route/Spider Matrix Tables", sep = ""))

Route <- read.csv2(paste(working_directory,
         "Tables/Monitor/Routes_evaluation_start_file.csv", sep = ""),
         stringsAsFactors = FALSE, check.names = FALSE)

Coordinates <- matrix(NA, length(Route[,5]), 3)
count <- 1

# Write route from coordinates into matrix "Coordinates" once
for(i in 1:length(Route[,5])) {
  if(identical(grep(paste(Route[i,5], "/", Route[i,6], sep = ""), Coordinates[,3],
  fixed=TRUE), integer(0)) == TRUE) {
    Coordinates[count,1] <- Route[i,5]
    Coordinates[count,2] <- Route[i,6]
    Coordinates[count,3] <- paste(Route[i,5], "/", Route[i,6], sep = "")
    count <- count+1
  }
}

# Write route to coordinates into matrix "Coordinates" once
for(i in 1:length(Route[,7])) {
  if(identical(grep(paste(Route[i,7], "/", Route[i,8], sep = ""), Coordinates[,3],
  fixed=TRUE), integer(0)) == TRUE) {
    Coordinates[count,1] <- Route[i,7]
    Coordinates[count,2] <- Route[i,8]
    Coordinates[count,3] <- paste(Route[i,7], "/", Route[i,8], sep = "")
    count <- count+1
  }
}

# Delete rows with NA
rows_NA <- matrix(NA, 0, 1)
for(i in 1:length(Coordinates[,1])) {

  if(sum(is.na(Coordinates[i,])) == 3) {
    rows_NA <- rbind(rows_NA, i)
  }
}
if(length(rows_NA) > 0) {
  Coordinates <- Coordinates[c(-rows_NA),]
}

# Change the column names
colnames(Coordinates) <- c("X", "Y", "X/Y")

# Write coordinates to csv.file
write.csv2(Coordinates, file = "Coordinates_routes.csv", row.names = FALSE)

#
## Blend the Coordinates_routes.csv with the traffic zones in QGIS
## Then new file --> Coordinates_districts
#

# Read in blended coordinates with districts
Coordinates_districts <- read.csv2(paste(working_directory, "Tables/Route/
     Spider Matrix Tables/Coordinates_districts.csv", sep = ""),
     stringsAsFactors = FALSE, check.names = FALSE)

# Delete rows with NA
rows_NA <- matrix(NA, 0, 1)
for(i in 1:length(Coordinates_districts[,1])) {

  if(sum(is.na(Coordinates_districts[i,])) > 0) {
    rows_NA <- rbind(rows_NA, i)
  }
}
if(length(rows_NA) > 0) {
  Coordinates_districts <- Coordinates_districts[c(-rows_NA),]
}

# Read in sorted route table
Route <- read.csv2(paste(working_directory,
         "Tables/Monitor/Routes_evaluation_start_file.csv", sep = ""),
         stringsAsFactors = FALSE, check.names = FALSE)

# Add two columns for districts
Route_districts <- cbind(Route[,1:3], Route[,5:6], matrix(NA, nrow(Route), 1),
                   Route[,7:8], matrix(NA, nrow(Route), 1))

# Change the column names of these two columns
colnames(Route_districts)[6] <- "District from"
colnames(Route_districts)[9] <- "District to"

# Add the belonging districts to the coordinates
for(i in 1:nrow(Route_districts)) {

  if(identical(grep(paste(Route_districts[i,4], "/", Route_districts[i,5], sep =
  ""), Coordinates_districts[,3], fixed=TRUE), integer(0)) == FALSE) {
    row_nr_from <- grep(paste(Route_districts[i,4], "/", Route_districts[i,5],
                   sep = ""), Coordinates_districts[,3], fixed=TRUE)
    Route_districts[i,6] <- Coordinates_districts[row_nr_from,4]
  }

  if(identical(grep(paste(Route_districts[i,7], "/", Route_districts[i,8], sep =
  ""), Coordinates_districts[,3], fixed=TRUE), integer(0)) == FALSE) {
    row_nr_to <- grep(paste(Route_districts[i,7], "/", Route_districts[i,8], sep =
                 ""), Coordinates_districts[,3], fixed=TRUE)
    Route_districts[i,9] <- Coordinates_districts[row_nr_to,4]
  }
}

# Delete rows with NA
rows_NA <- matrix(NA, 0, 1)
for(i in 1:length(Route_districts[,1])) {

  if(sum(is.na(Route_districts[i,])) > 0) {
    rows_NA <- rbind(rows_NA, i)
  }
}
if(length(rows_NA) > 0) {
  Route_districts <- Route_districts[c(-rows_NA),]
}


# Matrix districts from-to
districts_from_to <- matrix(NA, 0, 2)
count <- 1

# Write district to district relationship into the matrix "districts_from_to"
for(i in 1:nrow(Route_districts)) {

  if(identical(grep(paste(Route_districts[i,6], "/", Route_districts[i,9], sep =
  ""), districts_from_to[,1], fixed = TRUE), integer (0)) == TRUE &&
  identical(grep(paste(Route_districts[i,6], "/", Route_districts[i,9], sep = ""),
  districts_from_to[,2], fixed = TRUE), integer (0)) == TRUE) {

    districts_from_to <- rbind(districts_from_to, cbind(paste(Route_districts[i,6],
    "/", Route_districts[i,9], sep = ""), paste(Route_districts[i,9], "/",
    Route_districts[i,6], sep = "")))

    count <- count+1
  }
}

# Change the column names
colnames(districts_from_to) <- c("Districts from/to", "Districts to/from")

# Read in the centroids of the Styrian districts
Centroids_styria <- read.csv2(paste(working_directory, "Data/Centroids_Styria.csv",
                    sep = ""), stringsAsFactors = FALSE, check.names = FALSE)

# Get every date once and add them to the vector "dates"
dates <- matrix(NA, 0, 1)
for(i in 1:nrow(Route_districts)) {
  if(identical(grep(Route_districts[i,2], dates[,1], fixed = TRUE), integer (0)) ==
  TRUE) {
    dates <- rbind(dates, Route_districts[i,2])
  }
}

# Create a matrix for all days with zeros and change the column names
Weekdays <- matrix(0, nrow(districts_from_to), nrow(dates))
colnames(Weekdays) <- dates

# Create a matrix for the centroids with coordinates from-to and change the column
  names
centroids <- matrix(NA, nrow(districts_from_to), 4)
colnames(centroids) <- c("From X", "From Y", "To X", "To Y")

# Combine the whole information to one matrix
spider_matrix <- cbind(districts_from_to, centroids, Weekdays)

# Go through every requests and increase the belonging entry by one every time
for(i in 1:nrow(Route_districts)) {

  if(identical(grep(paste(Route_districts[i,6], "/", Route_districts[i,9], sep =
  ""), spider_matrix[,1], fixed = TRUE), integer (0)) == FALSE) {
    row_nr <- grep(paste(Route_districts[i,6], "/", Route_districts[i,9], sep =
    ""), spider_matrix[,1], fixed = TRUE)
    col_nr <- grep(Route_districts[i,2], colnames(spider_matrix), fixed = TRUE)

    spider_matrix[row_nr, col_nr] <- spider_matrix[row_nr, col_nr]+1
    next
  }

  if(identical(grep(paste(Route_districts[i,6], "/", Route_districts[i,9], sep =
  ""), spider_matrix[,2], fixed = TRUE), integer (0)) == FALSE) {
    row_nr <- grep(paste(Route_districts[i,6], "/", Route_districts[i,9], sep =
    ""), spider_matrix[,2], fixed = TRUE)
    col_nr <- grep(Route_districts[i,2], colnames(spider_matrix), fixed = TRUE)

    spider_matrix[row_nr, col_nr] <- spider_matrix[row_nr, col_nr]+1
    next
  }
}

# Add the coordinates of the centroids to the spider matrix
for(i in 1:nrow(spider_matrix)) {

  from <-
unlist(regmatches(spider_matrix[i,1],regexec("(.*?)/",spider_matrix[i,1])))[2]
  to <-
unlist(regmatches(spider_matrix[i,1],regexec("/(.*?)$",spider_matrix[i,1])))[2]

  row_nr_from <- grep(from, Centroids_styria[,1], fixed = TRUE)
  if(length(row_nr_from) == 1) {
    spider_matrix[i,3] <- Centroids _styria[row_nr_from,4]
    spider_matrix[i,4] <- Centroids _styria[row_nr_from,5]
  } else {
    row_nr_from <- grep(from, Centroids _styria[,1], fixed = TRUE)[1]
    spider_matrix[i,3] <- Centroids _styria[row_nr_from,4]
    spider_matrix[i,4] <- Centroids _styria[row_nr_from,5]
  }

  row_nr_to <- grep(to, Centroids _styria[,1], fixed = TRUE)
  if(length(row_nr_to) == 1) {
    spider_matrix[i,5] <- Centroids _styria[row_nr_to,4]
    spider_matrix[i,6] <- Centroids _styria[row_nr_to,5]
  } else {
    row_nr_to <- grep(to, Centroids _styria[,1], fixed = TRUE)[1]
    spider_matrix[i,5] <- Centroids _styria[row_nr_to,4]
    spider_matrix[i,6] <- Centroids _styria[row_nr_to,5]
  }
}

# Add sum and medians to the spider matrix
median <- matrix(NA, nrow(spider_matrix), 12)
colnames(median) <- c("Sum",
                    "Median 01.10.2018 - 31.03.2019",
                    "Median autumn 01.10.2018 - 18.11.2018",
                    "Median winter 19.11.2018 - 10.02.2019",
                    "Median spring 11.02.2019 - 31.03.2019",
                    "Median Mondays 01.10.2018 - 31.03.2019",
                    "Median Tuesdays 01.10.2018 - 31.03.2019",
                    "Median Wednesdays 01.10.2018 - 31.03.2019",
                    "Median Thursdays 01.10.2018 - 31.03.2019",
                    "Median Fridays 01.10.2018 - 31.03.2019",
                    "Median Saturdays 01.10.2018 - 31.03.2019",
                    "Median Sundays 01.10.2018 - 31.03.2019")

spider_matrix <- cbind(spider_matrix, median)

# Define the time periods for autumn, winter and spring
start_autumn <- "01.10.2018"
end_autumn <- "18.11.2018"
start_winter <- "19.11.2018"
end_winter <- "10.02.2019"
start_spring <- "11.02.2019"
end_spring <- "31.03.2019"

# Calculate the sum and the different medians for every district to district
  relationship and write them into the belonging columns
spider_matrix[,grep("Sum", colnames(spider_matrix))] <-
     apply(spider_matrix[,7:(grep("Sum", colnames(spider_matrix))-1)], 1, sum)
spider_matrix[,grep("Median 01.10.2018 - 31.03.2019", colnames(spider_matrix))] <-
     apply(spider_matrix[,7:(grep("Sum", colnames(spider_matrix))-1)], 1, median)
spider_matrix[,grep("Median autumn 01.10.2018 - 18.11.2018",
     colnames(spider_matrix))] <- apply(spider_matrix[,grep(start_autumn,
     colnames(spider_matrix))[1]:grep(end_autumn, colnames(spider_matrix))[1]], 1,
     median)
spider_matrix[,grep("Median winter 19.11.2018 - 10.02.2019",
     colnames(spider_matrix))] <- apply(spider_matrix[,grep(start_winter,
     colnames(spider_matrix))[1]:grep(end_winter, colnames(spider_matrix))[1]], 1,
     median)
spider_matrix[,grep("Median spring 11.02.2019 - 31.03.2019",
     colnames(spider_matrix))] <- apply(spider_matrix[,grep(start_spring,
     colnames(spider_matrix))[1]:grep(end_spring, colnames(spider_matrix))[1]], 1,
     median)
spider_matrix[,grep("Median Mondays 01.10.2018 - 31.03.2019",
     colnames(spider_matrix))] <- apply(spider_matrix[,seq(7, (grep("Sum",
     colnames(spider_matrix))-1), by=7)], 1, median)
spider_matrix[,grep("Median Tuesdays 01.10.2018 - 31.03.2019",
     colnames(spider_matrix))] <- apply(spider_matrix[,seq(8, (grep("Sum",
     colnames(spider_matrix))-1), by=7)], 1, median)
spider_matrix[,grep("Median Wednesdays 01.10.2018 - 31.03.2019",
     colnames(spider_matrix))] <- apply(spider_matrix[,seq(9, (grep("Sum",
     colnames(spider_matrix))-1), by=7)], 1, median)
spider_matrix[,grep("Median Thursdays 01.10.2018 - 31.03.2019",
     colnames(spider_matrix))] <- apply(spider_matrix[,seq(10, (grep("Sum",
     colnames(spider_matrix))-1), by=7)], 1, median)
spider_matrix[,grep("Median Fridays 01.10.2018 - 31.03.2019",
     colnames(spider_matrix))] <- apply(spider_matrix[,seq(11, (grep("Sum",
     colnames(spider_matrix))-1), by=7)], 1, median)
spider_matrix[,grep("Median Saturdays 01.10.2018 - 31.03.2019",
     colnames(spider_matrix))] <- apply(spider_matrix[,seq(12, (grep("Sum",
     colnames(spider_matrix))-1), by=7)], 1, median)
spider_matrix[,grep("Median Sundays 01.10.2018 - 31.03.2019",
     colnames(spider_matrix))] <- apply(spider_matrix[,seq(13, (grep("Sum",
     colnames(spider_matrix))-1), by=7)], 1, median)

# Calculate the sum of requests on one day
sum_days <- matrix(NA, 1, ncol(spider_matrix))
colnames(sum_days) <- colnames(spider_matrix)
spider_matrix <- rbind(spider_matrix, sum_days)
for(i in 7:ncol(spider_matrix)) {
  spider_matrix[nrow(spider_matrix), i] <-
       sum(spider_matrix[1:(nrow(spider_matrix)-1),i])
}

# Write the spider matrix to csv-file
write.csv2(spider_matrix, file = "Spider_matrix.csv", row.names = FALSE)
```

## Spider matrices plots: Medians six months, zoomed in

```
setwd(paste(working_directory, "Plots/Route/Spider Matrix Plots/Medians/Zoomed in",
sep = ""))

spider_matrix <- read.csv2(paste(working_directory, "Tables/Route/
                 Spider Matrix Tables/Spider_matrix.csv", sep = ""),
                 stringsAsFactors = FALSE, check.names = FALSE)

shape_Graz <- readShapeSpatial(paste(working_directory, "Data/
              Graz_traffic_zones.shp", sep = ""), proj4string =
              CRS("+proj=longlat"))

max <- max(spider_matrix[1:(nrow(spider_matrix)-1), (grep("Sum",
       colnames(spider_matrix))+1):ncol(spider_matrix)])

TULogo <- readPNG(paste(working_directory, "Data/TULogo.png", sep = ""))

for(i in (grep("Sum", colnames(spider_matrix))+1):ncol(spider_matrix)) {

  jpeg(filename = paste(i-189, ".Spider_matrix_", gsub(" ", "", sub(" ", "_",
  sub(" ", "_", colnames(spider_matrix)[i]))), ".jpg", sep = ""), width = 10,
  height = 7.6, units = "in", pointsize = 12, quality = 150, bg = "white", res =
  300, family = "", restoreConsole = TRUE, type = c("windows", "cairo"))

  par(mgp = c(2,1,0), oma = c(0,0,2,0), pin = c(9,6))
  plot(xlim = c(15.39,15.52), ylim = c(47.05,47.1), shape_Graz, xaxt="n")

  # Define the line end style as square
  par(lend = "square")

  # Maximum line width in legend
  lwd_legend <- 18

  # Maximum value in legend
  legend_max <- 50

  # Draw the lines between the districts (alpha is for transparency)
  for(j in 1:(nrow(spider_matrix)-1)) {
    if(spider_matrix[j,i] != 0) {
      segments(as.numeric(spider_matrix[j,3]), as.numeric(spider_matrix[j,4]),
      as.numeric(spider_matrix[j,5]), as.numeric(spider_matrix[j,6]), col =
      alpha("#005d7c", ((spider_matrix[j,i]*0.8)/legend_max)+0.2), lwd =
      ((spider_matrix[j,i]*lwd_legend)/legend_max))
    }
  }

  title(main = "Spider matrix route requests", font.main = 2, cex.main = 1.5,
  line = 2)
  mtext(paste(colnames(spider_matrix)[i], ", n = ",
  format(spider_matrix[nrow(spider_matrix),i], big.mark = ".", decimal.mark = ","),
  " [requests/d]", sep = ""), side = 3, cex = 0.9, line = 0.5)

  rect(15.4885, 47.0637, 15.5171, 47.081445, col = "white", border = "black", lty =
  "solid", lwd = 1)

  text(15.49, 47.07902, labels = "Legend", face = 2, cex = 1.5, pos = 4, offset =
  0)

  legend(15.49, 47.07747, legend = c(legend_max/4, legend_max/2, legend_max),
  lty ="solid", bty = "n", lwd = c(lwd_legend/4, lwd_legend/2, lwd_legend), col =
  c(alpha("#005d7c", 0.4), alpha("#005d7c", 0.6), alpha("#005d7c", 1.0)), seg.len =
  2, title = "Route requests", pt.cex = 2)

  legend(15.49, 47.06785, legend = "Graz traffic zones", pch = 22, bty = "n", col =
  "black", seg.len = 2, pt.cex = 2)

  rasterImage(TULogo, 15.4885,47.081445,15.5037,47.08657)
  dev.off()
}
# The other spider matrix plots are created the same way
```

## Public transport assignment demand matrices

```
setwd(paste(working_directory, "Tables/Route/Assignment Tables", sep = ""))

# Read in sorted route table
Route <- read.csv2(paste(working_directory,
         "Tables/Monitor/Routes_evaluation_start_file.csv", sep = ""),
         stringsAsFactors = FALSE, check.names = FALSE)

# Delete column "Platform"
Route <- Route[,-c(4)]

# Add a column with increasing numbers beginning from one and change the column
  name
Route <- cbind(Route, c(1:nrow(Route)))
colnames(Route)[8] <- "Nr"

# Writ the coordinates to csv-file
write.csv2(Route[,c(4,5,8)], file = "Coordinates_route_from.csv", row.names =
FALSE)
write.csv2(Route[,6:8], file = "Coordinates_route_to.csv", row.names = FALSE)

#
## Blend the coordinates with the traffic zones in QGIS
## Then import coordinates data with districts
#

Coordinates_route_from <- read.csv2(paste(working_directory, "Tables/
     Route/Assignment Tables/Coordinates_districts_route_from.csv", sep = ""),
     stringsAsFactors = FALSE, check.names = FALSE)
Coordinates_route_to <- read.csv2(paste(working_directory, "Tables/Route/
     Assignment Tables/Coordinates_districts_route_to.csv", sep = ""),
     stringsAsFactors = FALSE, check.names = FALSE)

# Sort these tables in increasing order according to the column "Nr"
Coordinates_route_from <- Coordinates_route_from[order(Coordinates_route_from[,3],
                          decreasing = FALSE), ]
```

```
Coordinates_route_to <- Coordinates_route_to[order(Coordinates_route_to[,3],
                        decreasing = FALSE), ]

# Replace coordinates with districts in route table
Route <- Route[, -c(4:8)]
Route <- cbind(Route, Coordinates_route_from[,4], Coordinates_route_to[,4])
```

## Demand matrices: Median six months

```
Route_new <- Route

# Delete rows with NA
rows_delete_na <- grep("1", Route_new[,4])
Route_new <- Route_new[rows_delete_na,]
rows_delete_na <- grep("1", Route_new[,5])
Route_new <- Route_new[rows_delete_na,]

# Build an array with districts from, districts to and the date
time <- table(Route_new[,4], Route_new[,5], Route_new[,2])

# Build the median over all tables
result_time <- apply(time, c(1,2), median)

# Read in all districts
districts <- read.csv2(paste(working_directory, "Data/Districts_Styria.csv", sep =
             ""), stringsAsFactors = FALSE, check.names = FALSE)
districts <- districts[,1]

# Add not used districts to matrix, column
colname_matrix <- colnames(result_time)
for(j in 1:length(districts)) {
  x <- grep(districts[j], colnames(result_time))
  if(identical(x, integer(0)) == TRUE) {
    colname_matrix <- c(colname_matrix, districts[j])
  }
}

# Fill up the new entries with zeros and change the column names
result_time <- cbind(result_time, matrix(0, nrow(result_time),
               length(colname_matrix)-ncol(result_time)))
colnames(result_time) <- colname_matrix

# Add not used districts to matrix, row
rowname_matrix <- rownames(result_time)
for(j in 1:length(districts)) {
  x <- grep(districts[j], rownames(result_time))
  if(identical(x, integer(0)) == TRUE) {
    rowname_matrix <- c(rowname_matrix, districts[j])
  }
}

# Fill up the new entries with zeros and change the row names
result_time <- rbind(result_time, matrix(0, length(rowname_matrix)-
               nrow(result_time), ncol(result_time)))
rownames(result_time) <- rowname_matrix

# Sort the table column by column and row by row
result_time <- result_time[,order(as.numeric(colnames(result_time)),
               decreasing = FALSE)]
result_time <- result_time[order(as.numeric(rownames(result_time)),
               decreasing = FALSE),]

# Write result to csv-file with row names
write.csv2(result_time, file = "Median_01.10.2018_-_31.03.2019.csv", row.names =
TRUE)

# The other demand matrices are created the same way
```

## Public transport assignment plots: Median six months

```
setwd(paste(working_directory, "Plots/Route/Assignment Plots", sep = ""))

Assignment <- read.csv2(paste(working_directory, "Tables/Route/
              Assignment Tables/Comparison_qando_data.csv", sep = ""),
              stringsAsFactors = FALSE, check.names = FALSE)
```

```
# Round the column to whole numbers with no decimal places
Assignment[,2] <- round(as.numeric(Assignment[,2]), digits = 0)

TULogo <- readPNG(paste(working_directory, "Data/TULogo.png", sep = ""))

for(i in 1:1) {

  jpeg(filename = "Public_transport_assignment_median.jpg", width = 8.33, height =
  4.58, units = "in", pointsize = 12, quality = 150, bg = "white", res = 300,
  family = "", restoreConsole = TRUE, type = c("windows", "cairo"))

  par(mgp = c(2,1,0), oma = c(0,0,0,0), pin = c(7,2.7))

  barplot(Assignment[,2], width = c(0.5), space = 1, legend.text = FALSE,
  col=c("#11cd1a"), xaxt = "n", yaxt = "n", xlim = c(0.5,15), ylim = c(0, 100))

  abline(h = seq(20, 100, by = 20), col = "gray87")

  barplot(Assignment[,2], width = c(0.5), space = 1, legend.text = FALSE,
  col=c("#11cd1a"), xaxt = "n", yaxt = "n", xlim = c(0.5,15), ylim = c(0, 100),
  add = TRUE)

  axis(side = 1, at = c(0:15)-0.25, labels = c("", Assignment[,1]), cex.axis =
  0.75, padj = -0.7)
  title(xlab = "Line", font.lab = 2, cex.lab = 1.0)

  axis(side = 2, at = seq(0, 100, by = 20), labels = seq(0, 100, by = 20),
  cex.axis = 0.75, padj = 0.5)
  title(ylab = "Line transportation [persons/d]", font.lab = 2, cex.lab = 1.0)

  title(main = "Public transport assignment", font.main = 2, cex.main = 1.5,
  line = 2)
  mtext(paste("Median 01.10.2018 - 31.03.2019", ", n = ",
  format(sum(Assignment[,2]), big.mark = ".", decimal.mark = ","), " [persons/d]",
  sep = ""), side = 3, cex = 0.9, line = 0.5)

  rasterImage(TULogo, 13,80,15.4,100)
  dev.off()
}
# The other public transport assignment plots are created the same way
```