CZECH TECHNICAL UNIVERSITY IN PRAGUE

FACULTY OF TRANSPORTATION SCIENCES

Bc. Viktor Beneš

OCCUPANCY PREDICTION OF PUBLIC PARKING SPACES

Master thesis

2021

CZECH TECHNICAL UNIVERSITY IN PRAGUE
Faculty of Transportation Sciences
Dean’s office
Konviktská 20, 110 00 Prague 1, Czech Republic

K620 .... Department of Transport Telematics

MASTER'S THESIS ASSIGNMENT

(PROJECT, WORK OF ART)

Student’s name and surname (including degrees):

Bc. Viktor Beneš

Code of study programme code and study field of the student:

N 3710 – IS – Intelligent Transport Systems

Theme title (in Czech): **Predikce obsazenosti veřejných parkovacích stání**

Theme title (in English): Occupancy prediction of public parking spaces

Guides for elaboration

During the elaboration of the master's thesis follow the outline below:

- introduction the current state-of-the-art of public parking spaces
- integration of public parking spaces to the Smart City concept
- description of public parking spaces datasets
- data analysis and prediction model
- evaluation of the designed system and proposal for future implementation

CTU
CZECH TECHNICAL UNIVERSITY IN PRAGUE

Graphical work range:	As required by the supervisor
Accompanying report length:	55 pages
Bibliography:	G. James, D. Witten, T. Hastie, and R. Tibshirani, An introduction to statistical learning: with applications in R. New York: Springer, [2013].
	Marr B, Big data: using SMART big data, analytics and metrics to make better decisions and improve performance. Hoboken: John Wiley & Sons; 2015.

Master's thesis supervisor: **Ing. Patrik Horažďovský**
PhD David Gundlegård

Date of master's thesis assignment: **July 9, 2020**
(date of the first assignment of this work, that has be minimum of 10 months before the deadline of the theses submission based on the standard duration of the study)

Date of master's thesis submission: **May 17, 2021**
a) date of first anticipated submission of the thesis based on the standard study duration and the recommended study time schedule
b) in case of postponing the submission of the thesis, next submission date results from the recommended time schedule

ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE
Fakulta dopravní
L. S.

[illegible] [illegible]

prof. Ing. Zdeněk Votruba, CSc.
head of the Department
of Transport Telematics

doc. Ing. Pavel Hrubeš, Ph.D.
dean of the faculty

I confirm assumption of master's thesis assignment.

[illegible]

Bc. Viktor Beneš
Student's name and signature

Prague ....July 9, 2020

Acknowledgment

I would like to thank Czech Technical University in Prague and Linköpings University for the education that allowed the origin of this Master's Thesis. I would like to thank the supervisors of my master thesis Ing. Patrik Horažďovský, Ing. Jiří Růžička, and shadow supervisor PhD David Gundlegård for the guidance, support, opinions, professional consultations, useful critique, and for all the help during this project work. I would also thank my colleagues Ing. Zuzana Purkrábková and Ing. Kristýna Navrátilová for their support, consultations, and help with the administration. Finally, I wish to thank my family for their moral and material support, consultations, and background.

Declaration

I hereby declare that the presented master thesis is my work and that I have cited all sources of information following the Guideline for adhering to ethical principles when elaborating an academic final thesis. I acknowledge that my thesis is subject to the rights and obligations stipulated by the Act No. 121/2000 Goll., the Copyright Act, as amended that the Czech Technical University in Prague has the right to conclude a license agreement on the utilization of this thesis as schoolwork under the provisions of Article 60(1) of the Act.

In Prague 16th of May 2021 ....

Signature

CZECH TECHNICAL UNIVERSITY IN PRAGUE

FACULTY OF TRANSPORTATION SCIENCES

OCCUPANCY PREDICTION OF PUBLIC PARKING SPACES

Master thesis

May 2021

Bc. Viktor Beneš

ABSTRACT

The subject of this master thesis is to develop a system that recommends to the user whether there are free parking places in the selected parking lot or not. This recommendation is based on the short-term occupancy prediction of public parking spaces. The system is based on historical parking data that are used for the data analysis and for the training of the prediction method.

KEY WORDS

Data analysis, Parking prediction, Occupancy prediction, Parking occupancy prediction, Available parking space, Smart City, Smart Cities, Off-street parking prediction, On-street parking prediction

Content

Abbreviations

API	Application Programming Interface
BR	Bagging Regression
CNN	Convolution Neural Networks
CSV	Comma-Separated Values
CTU in Prague	Czech Technical University in Prague
FNN	Multi-layer Feed-forward decoder
GeoJSON	Geographical JavaScript Object Notation
GUI	Graphical User Interface
ICT	Information and Communication Technologies
IoT	Internet of Things
ITS	Intelligent Transportation System
JSON	JavaScript Object Notation
K+R	Kiss & Ride
LiU	Linköping University
LPWAN	Low Power Wide Area Network
LSTM	Long Short-Term Memory
MSE	Mean square error
NARX	Nonlinear Autoregressive with External Input Neural Network
NN	Neural network
P+R	Park & Ride
R or r	Correlation coefficient
R^2 or R-squared	Coefficient of determination

RSS	Residual sum of squares
SUMP	Sustainable Urban Mobility Plans
TSK	Technická správa komunikací hl. m. Prahy, a.s. ("Technical administration of roads")
TSS	Total sum of squares
VMS	Variable Message Sign
VTS	Variable Traffic Sign

1 Introduction

Transportation is one of the very important parts of the Smart City concept. Transportation in a Smart City has three general categories: infrastructure and coordination, data, and services. The infrastructure covers roads, traffic signs, toll collection gates, parking places, etc. Traffic data are data about the traffic flow, closed roads, parking occupancy, weather, etc. The last category, services, covers the transportation itself. It means the mode of transport such as individual car transport, bike, walk, bike/car sharing, etc.

1.1 Problem description

Whereas motorization increases every year, the infrastructure increases much slower. It is because it is easier to use a car than to build a parking place for the car. However, to improve the efficiency of the current infrastructure data can be used. In general, data supports traffic coordination in real-time (for example via VMS – Variable Message Sign or VTS – Variable Traffic sign). Data can be used also for parking. If there is enough data from each parking place (parking house or paid parking zone on the street) then this data can be used for the coordination of the drivers to the unoccupied parking lots. This can reduce traffic because the driver will drive to the exact unoccupied parking space and he/she does not have to look for it.

The real-time parking occupancy data would not be enough. If the one driver parks his car, the unoccupied parking lot will change for the following driver. For this purpose, a short-term (approximately 15 to 30 minutes) prediction of the parking occupancy is useful. The prediction can support the coordination of drivers for the available parking lots.

1.2 The aim of the thesis

The aim of this thesis is to develop a system that recommends whether is good to go to the parking lot or not based on the time of the day and occupancy prediction. The system does a short-term occupancy prediction of public parking spaces. The occupancy prediction is based on the already gathered historical parking data that is collected by the municipality.

The main purpose of the occupancy prediction is to inform the driver about the occupancy at the arrival time. The short-term occupancy prediction could be presented in the mobile phone apps (such as navigation apps) or on the VMS/VTS by the main roads. Therefore, this system could help in the future with route planning and choosing the right transport mode for each passenger in the road system.

1.3 Research questions

The aim of the thesis leads to several research questions. The questions below are answered in the document.

- What type of data is required for occupancy prediction of public parking spaces?
- Is this data available (for example as open data) for citizens?
- Is the data structure from different cities and different types of parking lots sufficient for the purpose of occupancy prediction?

1.4 Limitations

The thesis can be limited by the data availability of the cities or of the parking places. The options of extension of this work to other cities is limited by the data structure and collecting data about the parking in each city. This work can be extended to other cities with the same or very similar approach to parking.

In the case of parking without a verification system (such as entrance gate or camera, etc.) on the on-street parking, the limitation is the respect of drivers to the rules. Somebody can park his car on the on-street parking and do not purchase the parking ticket. This driver broke the law, and the administrative body will perform its duty. However, this car is not in the payment system of the on-street parking and therefore it creates inaccuracy in the data because by the system there is no car but available parking space.

Another limitation can be the different citizen behavior during the COVID-19 pandemic. It means that the year 2020 was complicated for every citizen. Common life was affected by measures / restrictions (for example so-called lockdown) and the citizen behavior was completely changed. It means that parking data for the year 2020 is particularly challenging to use for prediction.

2 Background

Public parking spaces, cities, and an overview of the related word are introduced in this section. In addition, there is a description of the data sources used in this work.

2.1 Public parking spaces

At first, it is important to define the public parking space. For the purpose of this thesis, public parking space means a parking lot provided by the city. It means that municipality or city companies provide the parking service. These parking lots are maintained from the earnings and from the city budget. Hence, the information about each parking lot (such as capacity, price – per hour or per day, location) should be available.

There are two main parking forms: on-street and off-street parking.

On-street parking

On-street parking is a common type of parking in the city, city center respectively. It is a parking lot placed on the roads itself and is defined by the traffic sign (IP11a-e).[1] The traffic sign specifies the place on the road where cars are allowed to stop and park. Usually, drivers can park a car anywhere on or along the curbs of the streets.

These parking lots are efficient because it allows parking a car near to the driver's destination in the city center without the needs of a parking house. It allows parallel or angular parking to the road direction.[2] Kiss & Ride (K+S) parking placed near the big transfer point of public transport is also on-street parking, but it has a strict time limit (such as 3 minutes maximum) (see Figure 1). But the standard on-street parking (see Figure 2) has a fee. This fee is usually per hour of parking and the price is set by the municipality. However, anybody can park there for 3 minutes without any fee. It is a manipulation time set by the law, Act on Traffic on Roads and on Amendments to Certain Acts (see [3]).

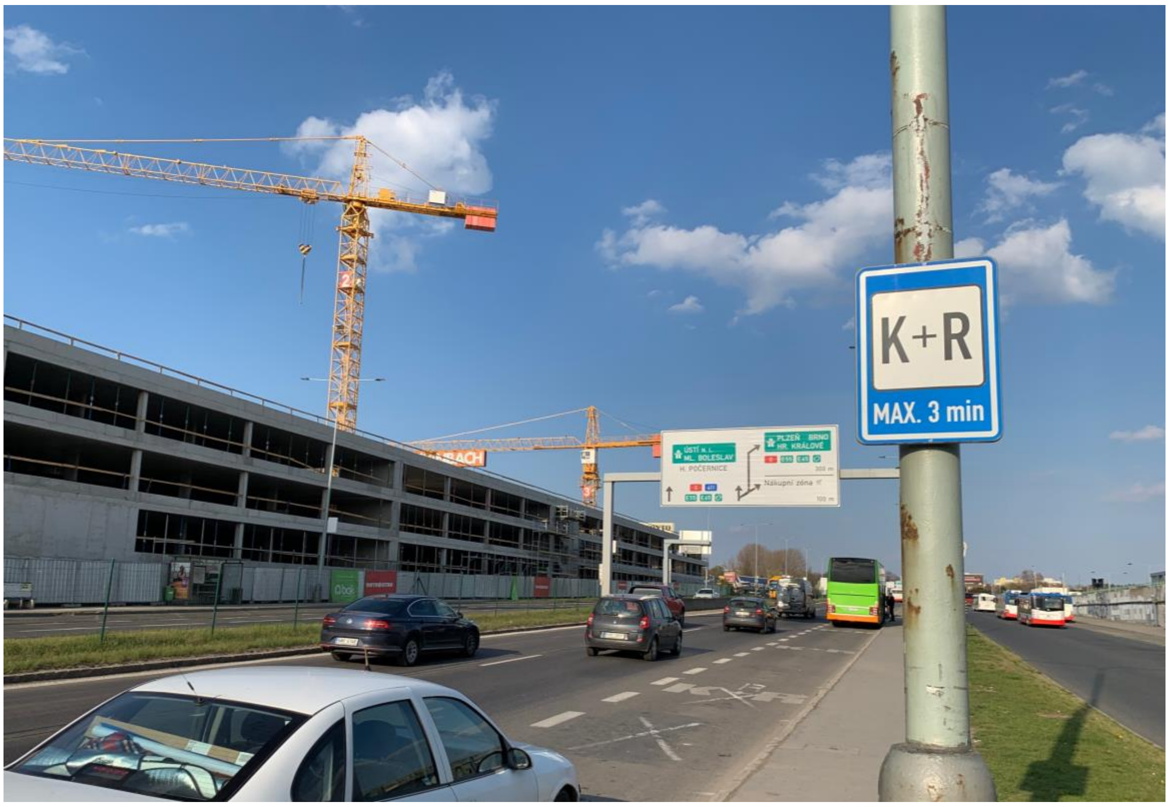


Figure 1: K+R Černý Most (Chlumecká street, direction to Horní Počernice) (28.04.2021)

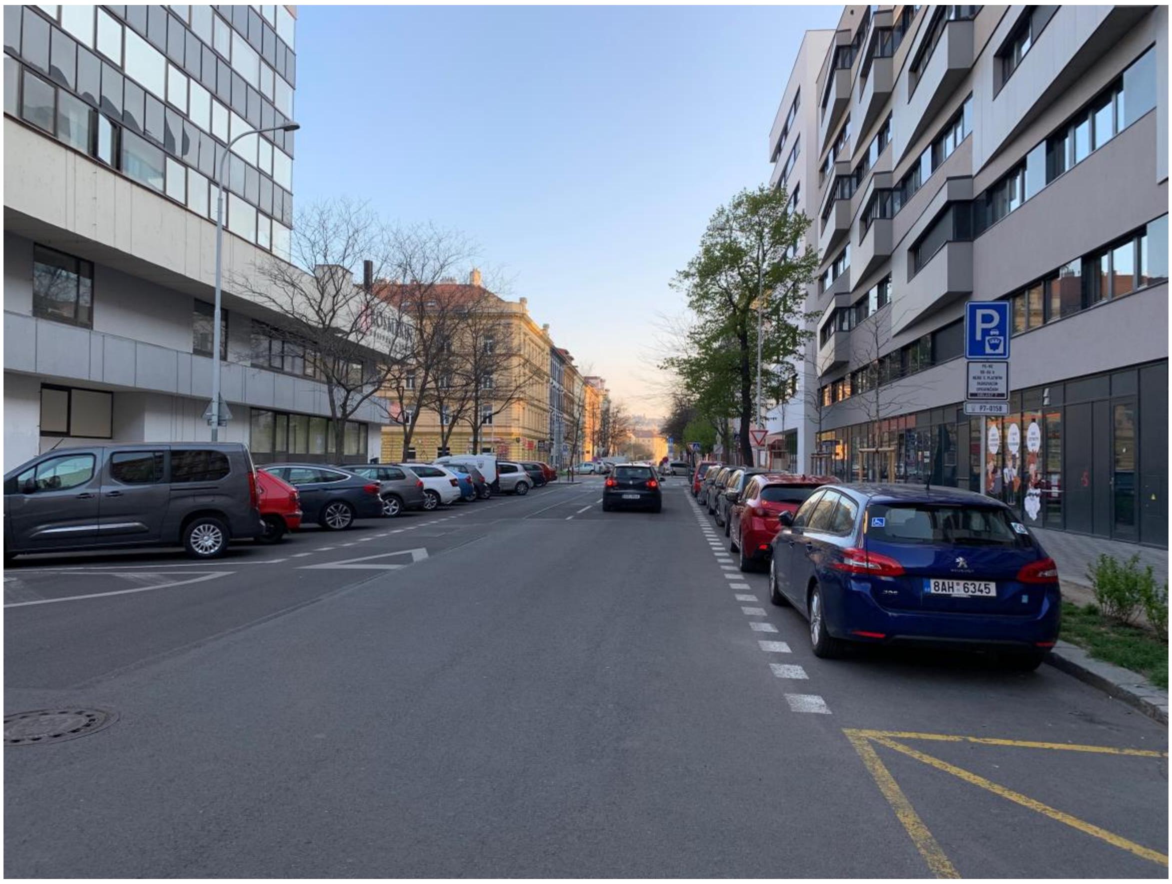


Figure 2: On-street parking – Paid parking zone (Osadní street) (28.04.2021)

Off-street parking

On the other hand, off-street parking is parking almost anywhere but not on the roads. It is the second common type of parking. Off-street parking is generally like a parking house (e.g., P+R – Park and Ride, private parking house, shopping center parking lot). These parking lots are equipped with an entrance/exit gate. The gate is similar to the tollgate. This gate allows counting the number of cars in the parking lot and compares it to the capacity – if there is enough parking space for other approaching cars.

For this thesis, off-street parking means P+R parking (see Figure 3) provided especially by the municipality. P+R parking is located at the huge public transport node. It is not necessary to place the P+R in the city. It can be placed in the big train node in the surrounding of the big city. The aim of this type of off-street parking is to change the traffic mode of the drivers. They use a car on the route to public transport and to the city center they use public transport itself. The car is parked on the big parking lot in the near surrounding of the public transport and usually, there is a fee per day that can be in the tariff with the long-term public transport coupon.

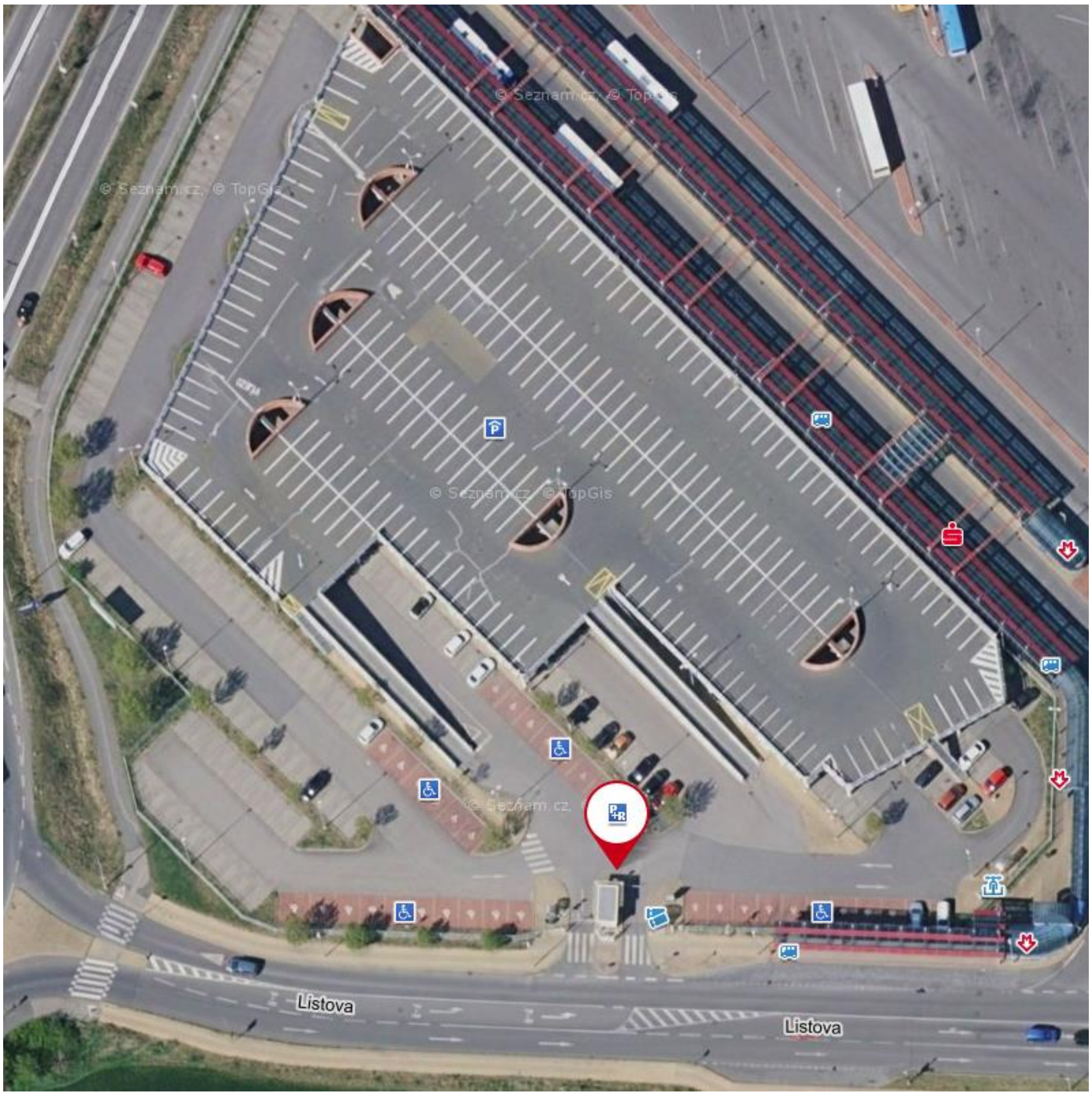


Figure 3: P+R Parking Prague Letňany [6]

2.2 Cities

This thesis is focused on two cities: Prague (Czech Republic) and Norrköping (Sweden). These cities were chosen based on the location of each faculty – Faculty of Transportation Sciences of Czech Technical University in Prague (CTU) and Faculty of Science and Engineering of Linköping University (LiU). Therefore, the parking situation is well-known. Each city has its differences, Prague represents a big city in the center of Czechia and Norrköping represents a standard-size city. Both cities are historical, but their approach is different. The behavior of drivers at each parking lot is presented in the Data analysis chapter based on the data. This chapter introduces the cities and their traffic, the parking mainly.

2.2.1 Prague

Prague is the capital city of the Czech Republic. It is placed almost in the center of the Bohemia region. The city of Prague had 1 324 277 inhabitants in 2019. The area of the city is 496 km^2 with the total length of the road network 4 047 km.[4]

The motorization (all vehicles) is 861 vehicles per 1 000 citizens, which is 1,2 citizens per vehicle. The motorization (personal vehicles) is 689 vehicles per 1 000 citizens (1,5 citizen per vehicle) (see. Figure 4). This data is for the year 2019.[4]

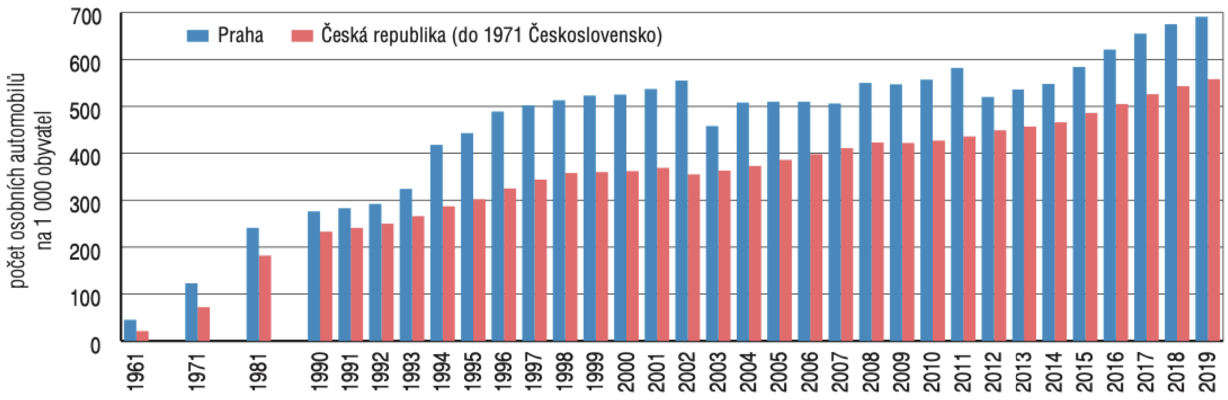


Figure 4: The motorization (personal vehicles) in Prague [4] (Prague – blue, the Czech Republic – red)

(During the years 2003 – 2007 a different algorithm was used, and the central vehicle register of the Czech Ministry of transportation is used from the year 2012.)

Motorization is on a high level and the trend is increasing. The number of commuters to Prague is 164.3 thousand people.[7] Therefore, the demand for parking space is increasing also. More paid parking zones (on-street parking) are built and P+R parking for the commuters is expanding or building new ones.

On-street parking

The on-street parking in Prague is called Paid parking zones. These zones are divided into three types:

- blue (residents)
- orange (visitors)
- purple (mixed)

The blue zones are preferably for the residents and then as paid parking for visitors. For paid parking in the blue zones there is a strict time limit (such as a 1-hour maximum). The orange zones are for paid parking only (no residents). And the purple zones are a combination of the blue and orange zones.[4]

The parking ticket is digital-only. The system is based on the license plate of each car. The visitors can pay either via ticket machine where they have to write their license plate or via mobile apps. Many mobile apps implement it into their user interface (e.g., PID Lítačka, Moje Praha, MPLA, Citymove, and many others). The control of the parking permit is provided by the monitoring car. The monitoring car is equipped with cameras and the system compares the license plate with the central evidence. Electric vehicles with a license plate that start with "EL" letters have automatically paid parking with no fee.[4]

In the whole of Prague, there are more than 70 thousand single blue (residential) zones, more than 40 thousand purple (mixed) zones, and more than 1 thousand orange (visitors) zones. Overall, it is 111 431 single paid parking zones.[4]

Off-street parking

In Prague is 21 P+R parking with an overall capacity of 3 946 parking spaces. 14 P+R parking are paid (20, - Czech crowns per day), have the maintenance and opening hours (from 4 am to 1 am). The rest parking (7 of them) is without maintenance and there is no fee.[4] This thesis is focused on the mentioned 14 paid P+R parking, because of the entrance/exit gate and the data collection.

Most of the P+R parking is placed near the subway stations. The rest of them are placed near the train stations (see Figure 5).

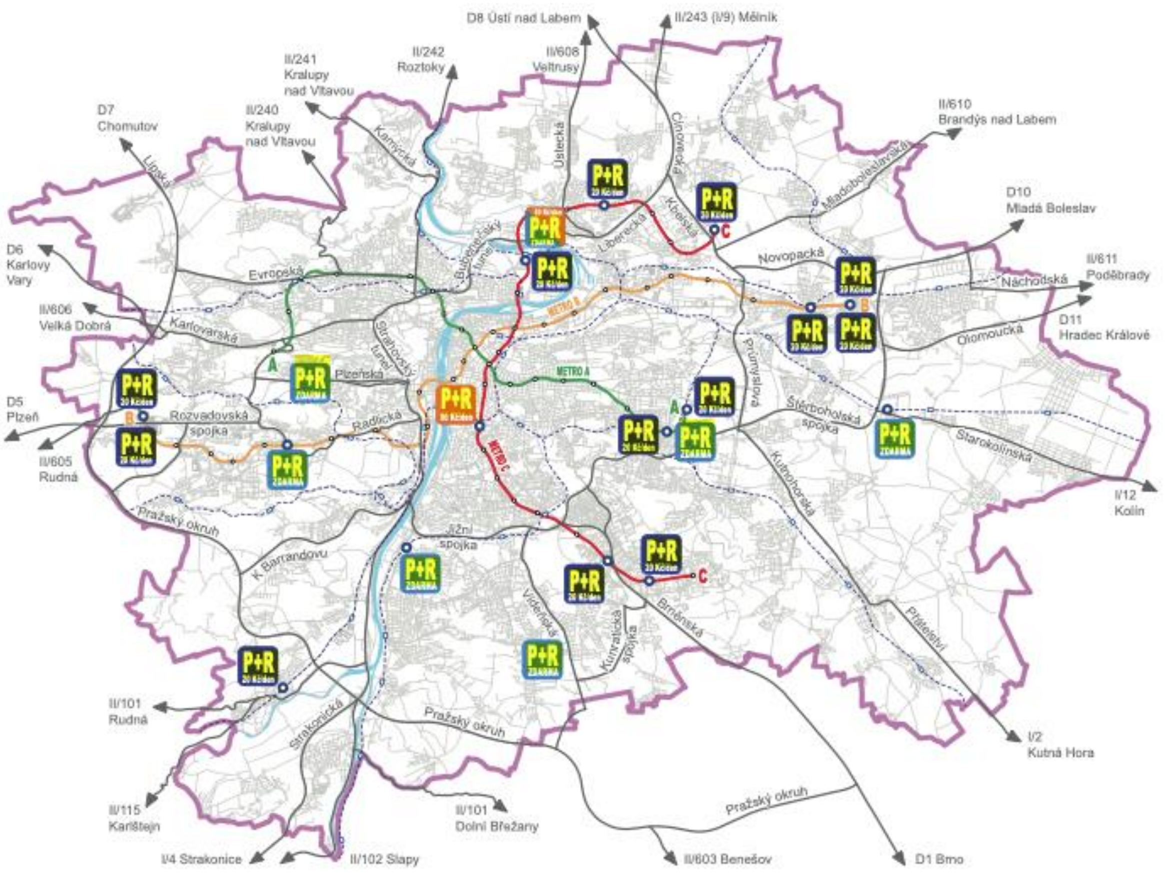


Figure 5: Map of Prague's P+R parking [4]

When the driver enters the parking, he/she gets the parking ticket from the entrance gate. On each parking, there is a ticket machine for paying the fee. So, the system is based on paper parking tickets. However, the entrance/exit gate counts how many cars already are in the parking lot. In 2019 the P+R Letňany (Figure 3) was equipped with cameras for license plate recognition. This system allows paying via the mobile apps mentioned with the on-street parking. It also speeds up the exit by the automatic gate opening. This was a pilot project of this system[1]. The other two P+R parking (P+R Černý Most 2 and P+R Rajská zahrada) were equipped with electric chargers for the electric vehicles.[4]

The next development of P+R parking in Prague do the year 2030 is in the SUMP (Sustainable Urban Mobility Plans).[8] It is expected in the future to create more than 9 thousand new parking spaces in Prague and almost 9 thousand new parking spaces in the Prague surroundings (by the big train nodes).[4]

[1] The final project report is available on the website: https://operatorict.cz/wp-content/uploads/2020/04/End-report-PR.pdf

2.2.2 Norrköping

The city of Norrköping is located in eastern Sweden in the Östergötland region. The city is about 160 km southwest of the capital city Stockholm. The number of citizens is 141 676 in 2020.[38] This number is for the whole Norrköping municipality which covers 1 495 km^2.[38]

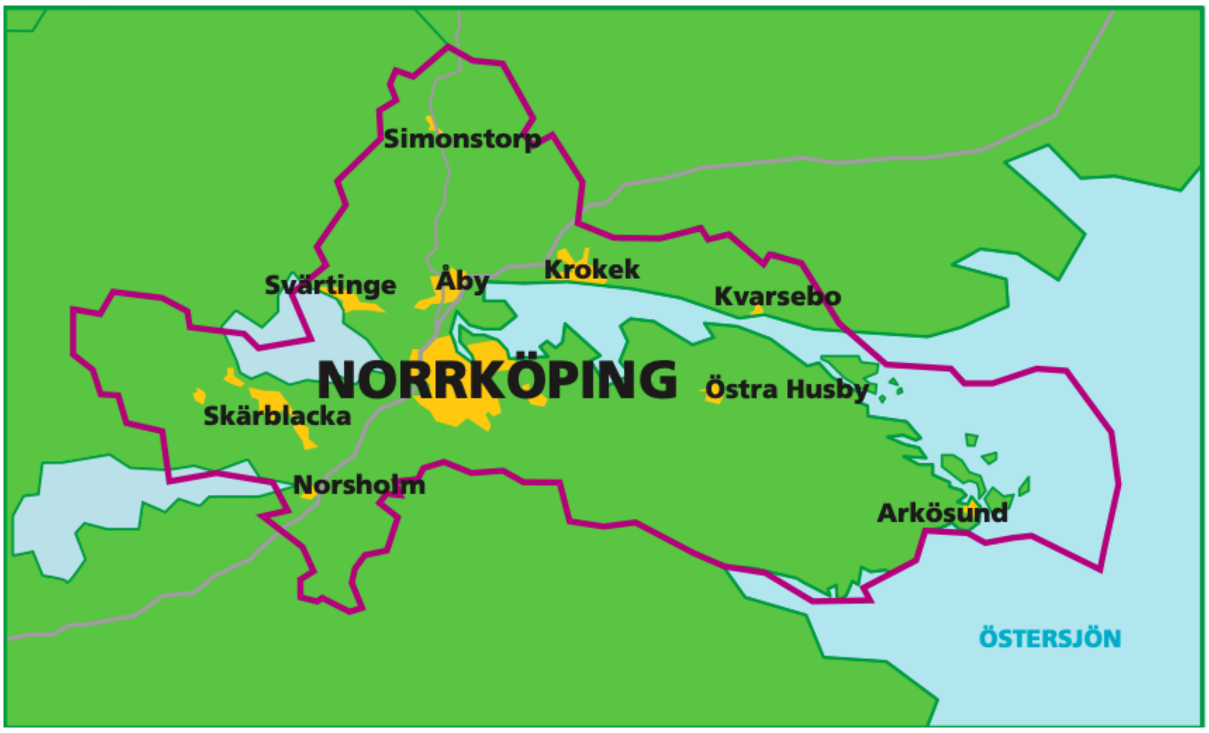


Figure 6: Map of the city of Norrköping and the whole Norrköping municipality [38]

Near the west side of the city is highway E4. This highway is going from the north of Sweden (border with Finland) to the south of Sweden to Helsingborg city. From the Finland border to Stockholm (north part) the highway is placed near the east coast of Sweden. From Stockholm to Helsingborg (south part) the highway is going through the upcountry. In the south of Norrköping begins another European route E22 and ends in the south of Sweden in Trelleborg city. This route has mixed standards containing motorway, two-lane expressway 2+1 road, and wide ordinary road.

On-street parking

The on-street parking in Norrköping is called City-P. Based on the information on the website of Norrköping [39] the City-P (Norrköping city parking) has two zones:

- blue
- red

It applies to most public parking spaces placed in residential parking areas and promenades. Other public parking spaces in the municipality are free except for a few places. The minimum parking time is one hour. The ticket can be bought via the Easypark or Parkster mobile app or on the ticket machine.

The day-ticket of the blue zones enables cars to park also in the red zones. Each ticket from the red zone enables cars to park in the blue zones. There are two parking cards: Parking card 300 and Parking card 400. Drivers with Parking card 300 can park in specially marked parking spaces for up to 24 hours. This card is valid for 30-days. Parking card 400 is for drivers that need to park for a shorter period on several occasions each day. The maximum parking time is 3 consecutive hours. Therefore, these drivers have to use a parking clock/disc.

There is also residential parking that enables parking specially designated places for a lower fee.

Off-street parking

Based on the information on the website of Norrköping [39] the public office provides public parking garages. These garages are open every day 05-00. The parking facilities are located in central Norrköping. The parking ticket can be purchased in the ticket machine or via the mobile app Easypark (the same as for the on-street parking). These garages are free of charge on Sundays and public holidays.

There are also parking places for the commuting drivers. These parking places are located in several places, generally near the roads with a high traffic flow (such as at North of Ingelsta next to the E4 and road 51). These parking places help to reduce car traffic on congested roads and reduce the pressure on parking spaces in the city. Most of the parking places are free of charge. The rest of them are charged (SEK 50/day, SEK 100/7 days, SEK 300/30 days). It can be paid by card or mobile.

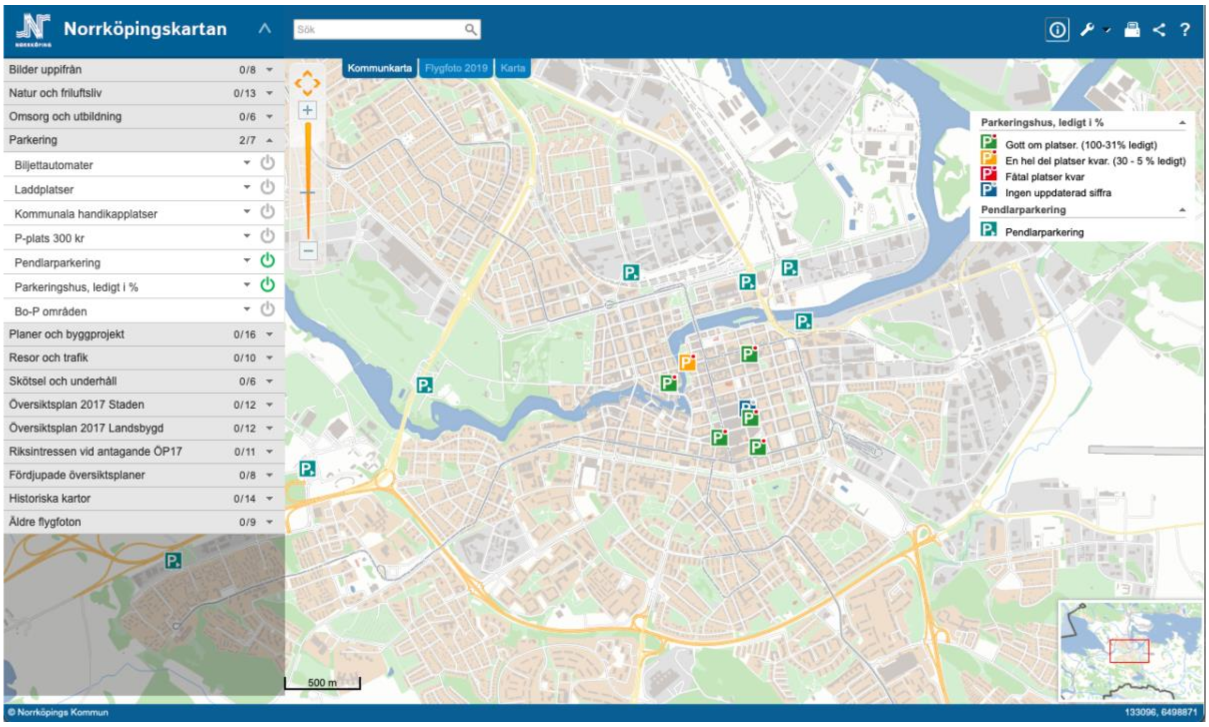

Norrköpingskartan
Sök
Bilder uppifrån 0/8
Natur och friluftsliv 0/13
Omsorg och utbildning 0/6
Parkering 2/7
Biljettautomater
Laddplatser
Kommunala handikapplatser
P-plats 300 kr
Pendlarparkering
Parkeringshus, ledigt i %
Bo-P områden
Planer och byggprojekt 0/16
Resor och trafik 0/10
Skötsel och underhåll 0/6
Översiktsplan 2017 Staden 0/12
Översiktsplan 2017 Landsbygd 0/12
Riksintressen vid antagande ÖP17 0/11
Fördjupade översiktsplaner 0/8
Historiska kartor 0/14
Äldre flygfoton 0/9
Kommunkarta
Flygfoto 2019
Karta
Parkeringshus, ledigt i %
Gott om platser. (100-31% ledigt)
En hel del platser kvar. (30 - 5 % ledigt)
Fåtal platser kvar
Ingen uppdaterad siffra
Pendlarparkering
Pendlarparkering
500 m
© Norrköpings Kommun
133096, 6498871

Figure 7: Map of Norrköping's parking garages (including parking for incoming drivers) [40]

Note: Green, Blue, and Yellow P signs of parking spaces represent parking garages. The cyan P sign represents parking for the incoming drivers.

2.3 Public parking spaces in cities

Each vehicle needs a parking lot for each trip – at the origin and the destination. As mentioned above, the motorization increases and therefore the parking demand is amplified as well.[9] This trend is mentioned also in [30]. The general prediction is that most of the world's population (about 70%) will live in cities by 2050.[33] And in the year 2006 was found that vehicles spend over 95% of their time parked.[10]

On-street parking has some positive and some negative consequences for the urban environment. The parked vehicles by the sidewalk can obstruct the side view. It means that the driver could have trouble seeing the coming pedestrian to the crosswalk, although the on-street parking can be parallel or angled (to the road).[9] However, a few pieces of research [11], [12] have a view that on-street parking could provide a safer environment for road users. Although, it depends on different factors such as culture and attitude in the location. Parked vehicles can create the following safety measures:

a) traffic calming measure that leads to lowering speed
b) a bumper that separates the road and the pavement

Different studies [13], [14] proved that urban on-street parking can slow down vehicle speed and thereby increase safety.

On the other hand, on-street parking reduces the traffic flow because the road is narrower, and the parking maneuvers limit the traffic for a while. Which leads to congestions. In [18], [30] is mentioned that 30% of urban congestions are created by drivers looking for a free parking space. The effects of the on-street parking depend on the road category. The literature [15] indicates an increase of 93% in traffic accidents when the major streets are equipped with on-street parking. It also depends on the type of on-street parking, whether it is parallel or angled. The parking maneuver, unparking maneuver respectively, in the angled on-street parking causes more than double crashes per unit distance than the parallel parking.[16] Angled parking also causes a higher decrease in traffic flow on the street. It is done by the same unparking maneuver.[17]

For future cities, it is important to a combination of the different parking possibilities (on-/off-street parking). Each possibility has its own positive and negative consequences. The combination of these possibilities leads to big off-street parking at the big public transportation nodes and on-street parking that allows parking at the destination. This on-street parking can be used by the shared vehicles, not private only. However, compromise should be in the middle – create on-street parking on secondary roads, ensure the safety of pedestrians and drivers by taking appropriate measures and enable drivers to park as close as possible to their destination.

In the Smart City concept, there is an emphasis on data. For parking, it means the occupancy measurement, sharing this data, and predicting the occupancy to the future (short-term, midterm, long-term prediction). The data sharing can navigate drivers into the exact position of the available parking spot in real-time. But in real-time, another driver can be at the parking spot faster, and therefore the navigation (mobile application or VTS/VMS) might route drivers to parking lots that are about to be occupied. The short-term prediction can avoid these problems. On the VTS's or in navigation can be shared information what will be at a time when the driver arrives at the parking destination (with a certain degree of accuracy).

2.4 Prediction of parking occupancy

The aim of the Smart parking system for the purpose of smart cities is to provide high-quality services to drivers. The success of such a system is to correctly predict available parking lots.[19] These systems have become one of the main parts of ITS (intelligent transportation system). The main purpose of the parking system is to find and provide an appropriate available parking lot.[20] Because of the increase in traffic flow (motorization) in urban areas, finding available parking lots is very challenging.[19] Providing real-time information is sufficient for the drivers in a closer area only. For the drivers that are further away is more useful to use prediction with the time when they reach the destination.[21] For the prediction is important to distinguish days. It means separate workdays, weekends, holidays, etc.[19]

In the literature, it is mentioned that drivers appreciate parking information provided by the parking guidance systems.[25] These systems have positive effects on the traffic and trips/routes of road users because such systems increase the probability of finding an available parking lot and decrease congestion associated with looking for the parking space. It also has a positive effect on the driver. The driver has a higher level of certainty that he/she parks at his destination. If parking guidance systems (such as VMS or mobile app and others) can show the "future" (in the time of arrival) occupancy with a high level of certainty the driver will be calmer on his route. The driver will be calmer because he will have a high probability that he will park in time at the destination. So, he will not be late for the appointment.

Data sources and sensors

One of the key elements of smart cities is the innovation and information and communication technologies (ICT) associated with different types of sensors. Sensors based on the IoT (Internet of Things) concept are a novel type of smart city infrastructure. It could be many types of devices, such as radiofrequency identification tags, mobile phones, and so on. And these elements can interact with each other and cooperate which leads to reaching goals.[26], [27]

There are several approaches (studies) that combine different data sources, for example, parking meter transactions, traffic speed, weather conditions, and many others that can improve prediction accuracy. For multiple data sources are used different prediction methods (such as neural networks, convolution neural networks, regression models). Generally, there are two methods how to predict short-term parking [22]:

1) model individual drivers' stochastic behavior in a microscopic manner based on the arrivals and departures
2) model of parking occupancy based on historical and real-time occupancy observations of parking

Based on the literature [23] the sensors implemented into the ground are not that efficient. These sensors are placed under each parking lot and can detect the vehicle above. However, the installation and maintenance of the sensors are expensive. Therefore, the usage of data from the parking ticket machine for the prediction can be more efficient.

Prediction methods

For the prediction in transportation problems (such as parking occupancy), it is popular to use a time-series modeling approach (for example Neural Network Models for Time Series Prediction).[28] It is suitable for this case due to the temporal structure of the parking data.[24]

The prediction method mentioned in the [22] is using Graph CNN (Convolution Neural Networks), LSTM (Long Short-Term Memory), and FNN (multi-layer feed-forward decoder). The connection of these systems works better for business areas than recreational locations, and for block-level parking. Weather information and traffic speed were also incorporated into this model. The weather information was very useful for the parking prediction in recreational areas.[22]

The short-term prediction based on the neural networks for the transportation problems has two parts training and testing. The training part is using a simple back-propagation algorithm. This algorithm has a genetically optimized learning rate and momentum. With the detailed description in [29]. The neural networks can adequately capture the parking occupancy in time and may accurately predict it for the next 30 minutes. This is the conclusion of the [24] based on the sensors in the Smart Santander project in Spain. This project uses IoT sensors for occupancy measuring. IoT sensors are used for the Bagging Regression (BR) approach that is mentioned in [30].

Data from the parking machine of on-street parking offers rich spatiotemporal information. This information leads to patterns of parking availability and therefore can be used for predicting parking availability. Methods such as regression trees and neural networks were used.[31]

In [32], four algorithms for on-street parking prediction are used. These algorithms were analyzed and compared. Two methods are based on historical data only (Mean and Variation of availability and Normally Distributed Availability). The other two methods are using real-time data (Normally Distributed Availability Variation and Non-homogeneous Poisson Distributed Arrivals and Departures). The conclusion is that real-time information improves performance to a certain prediction horizon.

In [34], three methods (regression tree, neural network, and support vector regression) are used for parking occupancy prediction. These methods were analyzed on datasets from San Francisco and Melbourne. Based on these datasets the conclusion is that the regression tree with a feature set (time of data, day of the week, ...) has the best performance.

2.5 Data sources

In this master thesis, data collected by the public authorities is used. The data is anonymized and therefore drivers cannot have a fear about the possible abuse. These data cannot be matched with a person. Therefore, it complies with the wording of paragraph 26 of the REGULATION (EU) 2016/679 OF THE EUROPEAN PARLIAMENT AND OF THE COUNCIL of 27 April 2016 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data and repealing Directive 95/46/EC (General Data Protection Regulation).[42] The data structure is described in the Data structure chapter.

Prague

In the case of Prague, the provider of traffic data (parking included) is the public company Technická správa komunikací hl. m. Prahy, a.s. ("Technical administration of roads"), also known as TSK.[35], [36] However, the data for the paid parking zones and for the P+R parking is based on different sources. The paid parking zones data is saved as the CSV (Comma-separated values) file. This CSV file includes all Prague's paid parking zones. This file is updated every 24h and after one quarter of the year is created a new file. It means that for each year it is 4 files. On the other side, the P+R parking has its own API (Application Programming Interface) with 20 seconds update period. It was mentioned that the TSK is the data provider, but the manager of the data is another public company called Operator ICT, a.s. The data for the paid parking zones are available on the Prague open data portal.[35] The API data are available on the Golemio website managed by the Operator ICT.[37] The documentation for these datasets is managed by the Operator ICT on their Golemio website.

Norrköping

In Norrköping is a LoRa network, which is a type of Low Power Wide Area Network (LPWAN). Through this network is connected many IoT elements such as traffic lights, ticket machines, etc. including parking guidance systems.[41] The data about parking is available in the management of the system. This system is provided by the company Infracontrol. In the management system, an authorized person can download parking data for each parking garage. Data is aggregated in 15 minutes intervals for each month. Data can be downloaded to a Microsoft Excel file.

3 Methodology

In this chapter, the methods and procedures are presented. At first, a method of how to answer the research questions and fulfill the aim of this thesis is presented. Then the data description (structure, inaccuracies, etc.) is presented and followed by the data filtering and data cleaning. Finally, there is described the functional principle of the proposed system.

3.1 The research questions and the aim of the thesis

The research questions and the aim of the thesis are intertwined. The research questions are focused on the parking data itself which is the input of the purposed system. Therefore, there was performed a research about the parking data itself. This research consisted of whether the municipalities (Prague and Norrköpng) monitor the parking occupancy and whether this data is publicly available. This research was based on the available internet sources based on the chosen cities and their data. The research was performed via the internet searcher Google. The searched expressions were: The searched expressions were: *Prague/Norrköping data platform, Prague/Norrköping parking data, Prague/Norrköping data,* and *Prague/Norrköping Smart City*. Results of this research are available in the Data sources chapter.

Data availability is the key part of this project. There is no prediction method without the data. The data availability of each city is described in the Data sources chapter above. For the occupancy prediction, there is necessary to know general information about the parking lot itself (such as location, and capacity) and about the occupancy in time (date and time, and occupancy in time). The location and capacity are not necessary for the data file, but this information must be available. The data structure is described in the Data structure chapter below.

The last research question is focused on the comparison of data structures from different cities and countries whether there are similarities. There is an assumption whether the data structure from different cities and countries should be similar because the parking is based on the same principle. Similarities are described in the Data structure similarities chapter below.

The smart city concept brings data to the management of the city. This data helps the municipality to measure different situations in the city and the municipality can more precisely solve problems and plan the city development. In the case of the parking lot, the data helps to understand the behavior of the drivers – based on the occupancy in time it is possible to observe peak hours, etc.

The aim of this thesis is to design and develop a system that will do a short-term occupancy prediction. This tool can help drivers with planning their route to their workplace/meetings/etc. The city can use this system for navigating drivers to the parking lot with a free parking space (with a high level of probability). This system is based on historical parking data. For the purpose of this thesis, there were chosen data from the beginning of 2018 to the end of 2020. Because of the COVID-19 pandemic, the data from 2020 was used from the first two months only. The restrictions connected with the pandemic negatively affected mobility. Therefore, it is the main limitation of this thesis. Based on the historical data a prediction model is developed. This model has two steps of the short-term prediction: 15-minutes and 30-minutes. The model predicts the occupancy in the next 15 or 30 minutes. The number of free parking spaces is evaluated (based on the trends for the chosen predicted time). Based on this evaluation there is a recommendation to the driver (whether the probability that the driver will park there is high or not). The principle of the system is in the Principle of the system chapter below.

3.2 Data structure

This chapter is focused on the data structure. It is divided into three parts based on the data inputs (P+R Prague, ZPS Prague, Parking garages Norrköping). Each data input is described in its chapter. In this chapter, it is verified if the data input has enough information (location, date and time, capacity, occupancy in time). In the subchapters below are presented real data used for the purpose of this thesis. These subchapters are used for the understanding of what parameters are necessary and how the structure of the data can look like. It is presented in this chapter because there is a possible purpose of extension of the system by additional cities/parking lots.

3.2.1 P+R Prague

This dataset is available at Golemio [37] via API. The update frequency is 20 seconds. It means that every 20 seconds data is updated. For the historic data, the API query is a little bit different. The difference is in the link. For the purpose of this thesis was used code:

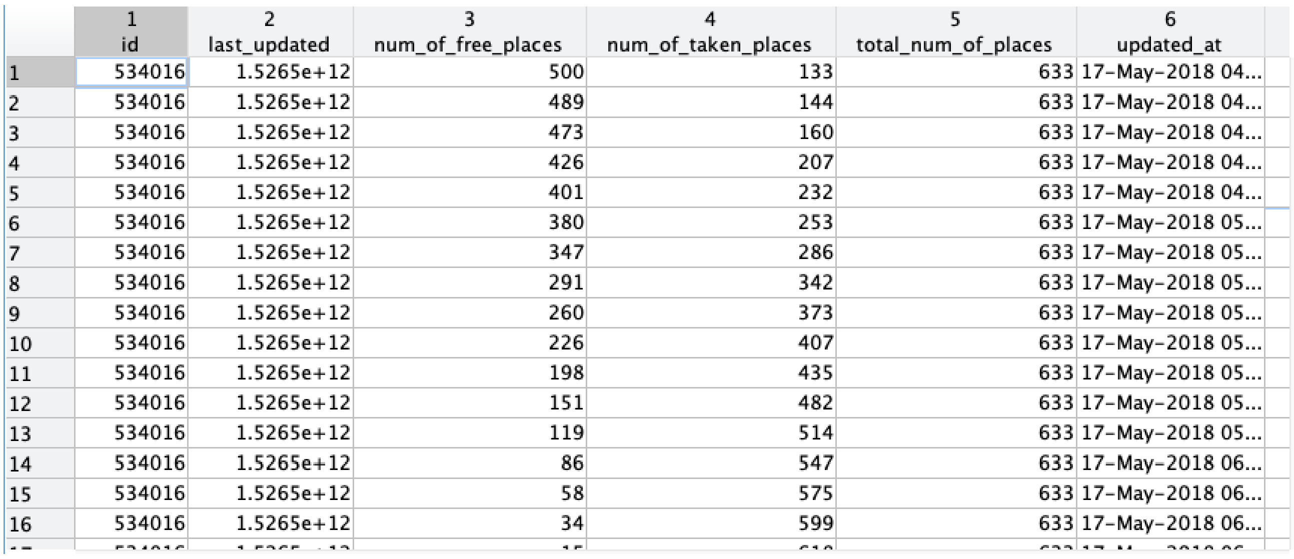
https://api.golemio.cz/v2/parkings/history?sensorId=534016&sensorId=534011&from=2018-12-01T00:00:00.000Z&to=2020-12-31T23:59:59.000Z

Where: https://api.golemio.cz/v2/parkings/history is the link to the server; sensorId=534016 represents P+R Letňany; sensorId=534011 represents P+R Černý Most 2; from=2018-12-01T00:00:00.000Z&to=2020-12-31T23:59:59.000Z is the date and time

This code was used in the Postman application. Through this application was downloaded all the needed data files (it means all files from January 2018 to December 2020). Because of the limitation of the servers which is 10 thousand rows, there is a need to download each month separately. All the files are in JSON (JavaScript Object Notation) format. The example of the JSON data follows:

[{"id":534016,"last_updated":1527810749000,"num_of_free_places":622,"num_of_taken_places":11,"total_num_of_places":633,"updated_at":"2018-05-31T23:52:29.000Z"},{"id":534016,"last_updated":1527810385000,"num_of_free_places":622,"num_of_taken_places":11,"total_num_of_places":633,"updated_at":"2018-05-31T23:46:25.000Z"},…]

This data was transformed to the table format in the Matlab application.



	1 id	2 last_updated	3 num_of_free_places	4 num_of_taken_places	5 total_num_of_places	6 updated_at
1	534016	1.5265e+12	500	133	633	17-May-2018 04...
2	534016	1.5265e+12	489	144	633	17-May-2018 04...
3	534016	1.5265e+12	473	160	633	17-May-2018 04...
4	534016	1.5265e+12	426	207	633	17-May-2018 04...
5	534016	1.5265e+12	401	232	633	17-May-2018 04...
6	534016	1.5265e+12	380	253	633	17-May-2018 05...
7	534016	1.5265e+12	347	286	633	17-May-2018 05...
8	534016	1.5265e+12	291	342	633	17-May-2018 05...
9	534016	1.5265e+12	260	373	633	17-May-2018 05...
10	534016	1.5265e+12	226	407	633	17-May-2018 05...
11	534016	1.5265e+12	198	435	633	17-May-2018 05...
12	534016	1.5265e+12	151	482	633	17-May-2018 05...
13	534016	1.5265e+12	119	514	633	17-May-2018 05...
14	534016	1.5265e+12	86	547	633	17-May-2018 06...
15	534016	1.5265e+12	58	575	633	17-May-2018 06...
16	534016	1.5265e+12	34	599	633	17-May-2018 06...

Figure 8: Screenshot of the P+R Prague data in the Matlab environment transformed to the table

In the data preview above it is possible to see that these data are anonymized. Thus, there is no chance of tracking specific people.

Each variable is described in the table below:

variable	**example**	**description**
id	*534016*	identification of the P+R parking lot
last_updated	*1527810749000*	timestamp of measurement by vendor
num_of_free_places	*622*	number of free places
num_of_taken_places	11	number of taken places
total_num_of_places	*633*	total number of places
updated_at	*2018-05-31T23:52:29.000Z*	timestamp of requesting vendor API

Table 1: Variables of the P+R Prague parking dataset

There is no information about the location of the parking place. Therefore, there is a GeoJSON file [43] with the coordinates of the parking lots in the Prague open data portal and other API for this purpose. The API code is:

https://api.golemio.cz/v2/parkings/534016

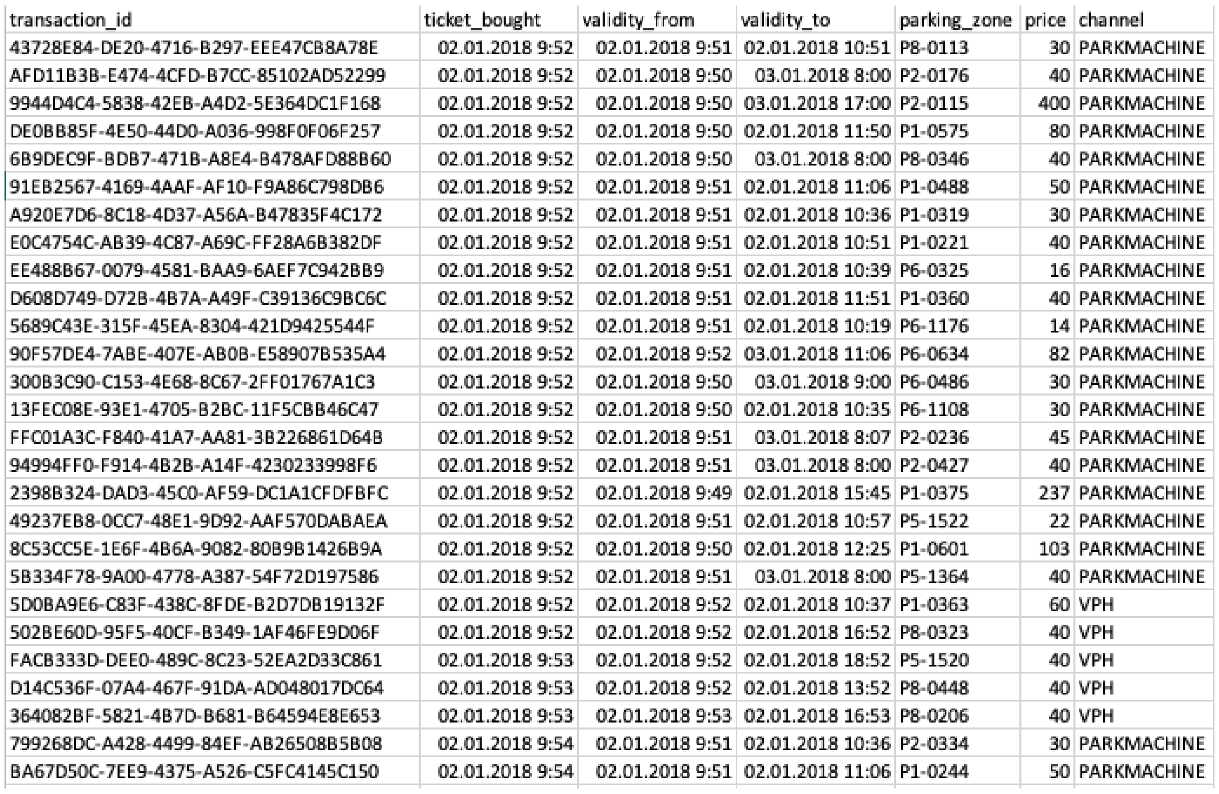
The answer is another JSON file. The example of the answer is following:

{"geometry": {"coordinates": [14.514741, 50.125168], "type": "Point" }, "properties": {"address": { "address_country": "Česko", "address_formatted": "Listova, 19900 Praha-Letňany, Česko", "street_address": "Listova", "postal_code": "19900", "address_locality": "Praha", "address_region": "Letňany"},…}

Where the location is (as coordinates).

3.2.2 ZPS Prague

This dataset is available at Prague open data portal [35] as a CSV file. The update frequency is 24 hours. It means that every 24 hours the last CSV file is updated. Each CSV file represents a quarter of a year. In each file is every paid parking zone in Prague. Records in the datasets represent bought parking tickets. The data is shown in the screenshot below.



transaction_id	ticket_bought	validity_from	validity_to	parking_zone	price	channel
43728E84-DE20-4716-B297-EEE47CB8A78E	02.01.2018 9:52	02.01.2018 9:51	02.01.2018 10:51	P8-0113	30	PARKMACHINE
AFD11B3B-E474-4CFD-B7CC-85102AD52299	02.01.2018 9:52	02.01.2018 9:50	03.01.2018 8:00	P2-0176	40	PARKMACHINE
9944D4C4-5838-42EB-A4D2-5E364DC1F168	02.01.2018 9:52	02.01.2018 9:50	03.01.2018 17:00	P2-0115	400	PARKMACHINE
DE0BB85F-4E50-44D0-A036-998F0F06F257	02.01.2018 9:52	02.01.2018 9:50	02.01.2018 11:50	P1-0575	80	PARKMACHINE
6B9DEC9F-BDB7-471B-A8E4-B478AFD88B60	02.01.2018 9:52	02.01.2018 9:50	03.01.2018 8:00	P8-0346	40	PARKMACHINE
91EB2567-4169-4AAF-AF10-F9A86C798DB6	02.01.2018 9:52	02.01.2018 9:51	02.01.2018 11:06	P1-0488	50	PARKMACHINE
A920E7D6-8C18-4D37-A56A-B47835F4C172	02.01.2018 9:52	02.01.2018 9:51	02.01.2018 10:36	P1-0319	30	PARKMACHINE
E0C4754C-AB39-4C87-A69C-FF28A6B382DF	02.01.2018 9:52	02.01.2018 9:51	02.01.2018 10:51	P1-0221	40	PARKMACHINE
EE488B67-0079-4581-BAA9-6AEF7C942BB9	02.01.2018 9:52	02.01.2018 9:51	02.01.2018 10:39	P6-0325	16	PARKMACHINE
D608D749-D72B-4B7A-A49F-C39136C9BC6C	02.01.2018 9:52	02.01.2018 9:51	02.01.2018 11:51	P1-0360	40	PARKMACHINE
5689C43E-315F-45EA-8304-421D9425544F	02.01.2018 9:52	02.01.2018 9:51	02.01.2018 10:19	P6-1176	14	PARKMACHINE
90F57DE4-7ABE-407E-AB0B-E58907B535A4	02.01.2018 9:52	02.01.2018 9:52	03.01.2018 11:06	P6-0634	82	PARKMACHINE
300B3C90-C153-4E68-8C67-2FF01767A1C3	02.01.2018 9:52	02.01.2018 9:50	03.01.2018 9:00	P6-0486	30	PARKMACHINE
13FEC08E-93E1-4705-B2BC-11F5CBB46C47	02.01.2018 9:52	02.01.2018 9:50	02.01.2018 10:35	P6-1108	30	PARKMACHINE
FFC01A3C-F840-41A7-AA81-3B226861D64B	02.01.2018 9:52	02.01.2018 9:51	03.01.2018 8:07	P2-0236	45	PARKMACHINE
94994FF0-F914-4B2B-A14F-4230233998F6	02.01.2018 9:52	02.01.2018 9:51	03.01.2018 8:00	P2-0427	40	PARKMACHINE
2398B324-DAD3-45C0-AF59-DC1A1CFDFBFC	02.01.2018 9:52	02.01.2018 9:49	02.01.2018 15:45	P1-0375	237	PARKMACHINE
49237EB8-0CC7-48E1-9D92-AAF570DABAEA	02.01.2018 9:52	02.01.2018 9:51	02.01.2018 10:57	P5-1522	22	PARKMACHINE
8C53CC5E-1E6F-4B6A-9082-80B9B1426B9A	02.01.2018 9:52	02.01.2018 9:50	02.01.2018 12:25	P1-0601	103	PARKMACHINE
5B334F78-9A00-4778-A387-54F72D197586	02.01.2018 9:52	02.01.2018 9:51	03.01.2018 8:00	P5-1364	40	PARKMACHINE
5D0BA9E6-C83F-438C-8FDE-B2D7DB19132F	02.01.2018 9:52	02.01.2018 9:52	02.01.2018 10:37	P1-0363	60	VPH
502BE60D-95F5-40CF-B349-1AF46FE9D06F	02.01.2018 9:52	02.01.2018 9:52	02.01.2018 16:52	P8-0323	40	VPH
FACB333D-DEE0-489C-8C23-52EA2D33C861	02.01.2018 9:53	02.01.2018 9:52	02.01.2018 18:52	P5-1520	40	VPH
D14C536F-07A4-467F-91DA-AD048017DC64	02.01.2018 9:53	02.01.2018 9:52	02.01.2018 13:52	P8-0448	40	VPH
364082BF-5821-4B7D-B681-B64594E8E653	02.01.2018 9:53	02.01.2018 9:53	02.01.2018 16:53	P8-0206	40	VPH
799268DC-A428-4499-84EF-AB26508B5B08	02.01.2018 9:54	02.01.2018 9:51	02.01.2018 10:36	P2-0334	30	PARKMACHINE
BA67D50C-7EE9-4375-A526-C5FC4145C150	02.01.2018 9:54	02.01.2018 9:51	02.01.2018 11:06	P1-0244	50	PARKMACHINE

Figure 9: Screenshot of the ZPS Prague data in the Excel environment

In the data preview above it is possible to see that these data are anonymized. This data cannot be connected to specific people.

variable	**example**	**description**
transaction_id	*43728E84-DE20-4716-B297-EEE47CB8A78E*	identification of the parking ticket
ticket_bought	*02.01.2018 9:52*	timestamp of purchasing a ticket
validity_from	02.01.2018 9:50	timestamp of validity starts
validity_to	03.01.2018 17:00	timestamp of validity ends
parking_zone	*P2-0115*	id of the parking zone
price	*400*	price of the ticket
channel	*PARKMACHINE*	channel of purchasing the ticket (ticket machine – parkmachine or mobile app - VPH)

Table 2: Variables of the ZPS Prague parking dataset

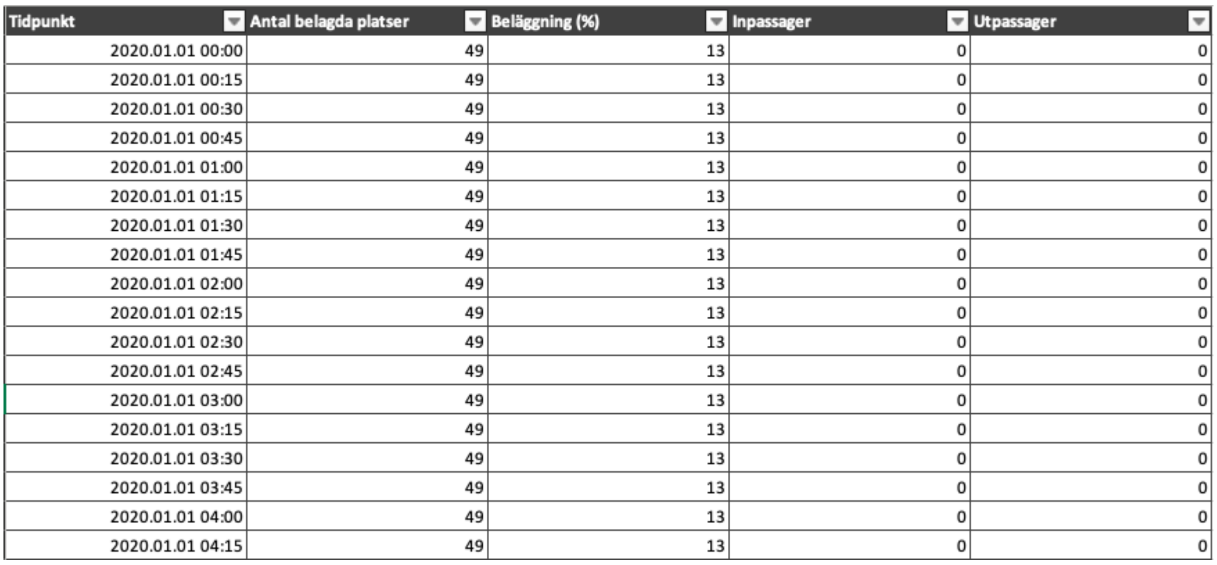
There is no information about the location of the parking zone and about the capacity. Therefore, there is another GeoJSON file with the coordinates and capacity of the parking lots in the Prague open data portal.[44] The example of the GeoJSON file is:

{"type" : "FeatureCollection", "name" : "DOP_ZPS_USEKY_p", "crs" : {"type" : "name", "properties" : {"name" : "EPSG:5514"}}, "features" : [{"type" : "Feature", "geometry" : {"type" : "Polygon", "coordinates" : [[[-741207.2478820011, -1043433.4962281734], [-741205.3908695057, -1043433.7941877544], [-741207.4218844026, -1043445.8723263592], [-741213.8947823942, -1043485.4672951102], [-741215.8427468538, -1043485.1603277177], [-741207.2478820011, -1043433.4962281734]]]}, "properties" : {"OBJECTID" : 1, "ZPS_ID" : "4875", "TYPZONY" : "2", "TARIFTAB" : "P3-0105", "PS_ZPS" : "9", "Shape_Length" : 108.59519045513692, "Shape_Area" : 99.76609455113356 }},…}

In the GeoJSON file is the location of the zone (as coordinates) and the capacity (as PS_ZPS).

3.2.3 Parking garages Norrköping

This dataset is available at Infracontrol management center[2]. Data is available for download as an Excel file. Data is aggregated every 15 minutes. The data is shown in the screenshot below.



Tidpunkt	Antal belagda platser	Beläggning (%)	Inpassager	Utpassager
2020.01.01 00:00	49	13	0	0
2020.01.01 00:15	49	13	0	0
2020.01.01 00:30	49	13	0	0
2020.01.01 00:45	49	13	0	0
2020.01.01 01:00	49	13	0	0
2020.01.01 01:15	49	13	0	0
2020.01.01 01:30	49	13	0	0
2020.01.01 01:45	49	13	0	0
2020.01.01 02:00	49	13	0	0
2020.01.01 02:15	49	13	0	0
2020.01.01 02:30	49	13	0	0
2020.01.01 02:45	49	13	0	0
2020.01.01 03:00	49	13	0	0
2020.01.01 03:15	49	13	0	0
2020.01.01 03:30	49	13	0	0
2020.01.01 03:45	49	13	0	0
2020.01.01 04:00	49	13	0	0
2020.01.01 04:15	49	13	0	0

Figure 10: Screenshot of the Parking garages Norrköping data in the Excel environment

In the data preview above it is possible to see that these data are anonymized. This data cannot be connected to specific people.

[2] The Infracontrol management center is available here: https://online.infracontrol.com/

variable	**example**	**description**
Tidpunkt	*2020.01.01 00:00*	timestamp of the record
Antal belagda platser	*49*	number of occupied places
Beläggning (%)	13	occupancy
Inpassager	0	arrivals in the time section
Utpassager	*0*	departures in the time section

Table 3: Variables of the Parking garages Norrköping parking dataset

There is no information about the capacity and the location. The capacity information and the location information were found on the city website.[40]

3.2.4 Data structure similarities

In a general manner, each parking facility has an original data structure. However, similarities can be found between the P+R Prague and Parking garages Norrköping. In the data structure of Parking garages Norrköping is missing the total number of places and number of free places. The total number of places is on the city website [40]. Then the number of free places can be calculated for each data record. After that, the data structure is very similar. Therefore, under certain circumstances (supplementation of the mentioned data) the system has the possible usage as a multi-cities (multi-countries) tool.

3.3 Data preparation

Data preparation consists of data cleaning and data filtering. These procedures are integral parts of working with data. For achieving suitable results from data analysis, well-prepared data is very important.

3.3.1 Data filtering

Data filtering is usually a second step. The first one is data cleaning. Data filtering is the process of choosing a part of the whole dataset. This smaller part of the dataset is called a data subset which can be used for the analysis. The filtering does not remove the rest records in the dataset, the dataset is kept. Data filtering can help with the calculations, especially for the analysis. For example, filtration can be based on the period of time.

For the purpose of this thesis, data filtering has two parts. First of all, it is to choose the P+R parking facilities in Prague, Zones of paid parking in Prague, and parking garages in

Norrköping. In this case, it is to choose two parking facilities of P+R Prague and Parking garages Norrköping and one paid parking zone of ZPS Prague. Each of these parking facilities must be chosen based on the following rules:

- data availability
- station of the public transport or a big destination hub for drivers in the near surroundings (walking distance)
- user/personal experience

As the next step, it is necessary to do the data cleaning.

Then, the second part of the filtering is a time-based filter. This filter is based on the days of the week. Each day of the week is analyzed in the Data analysis chapter. Based on the occupancy characteristics will be analyzed each day of the week and based on the similarities in the characteristics there will be created groups of the days (such as weekdays, weekends, holidays, public holidays).

3.3.2 Data cleaning

Data cleaning assures getting rid of inaccurate data and errors in data. These faults can be presented in the collected raw data. Data cleaning is a process of detecting and "repairing" (removing or correcting) inaccuracies in the dataset. Inaccurate data and errors could be caused by the human factor (errors during maintenance, user mistakes, etc.), machine errors, or errors in the data transmission.

The errors and inaccuracies in the data may be of the following nature:

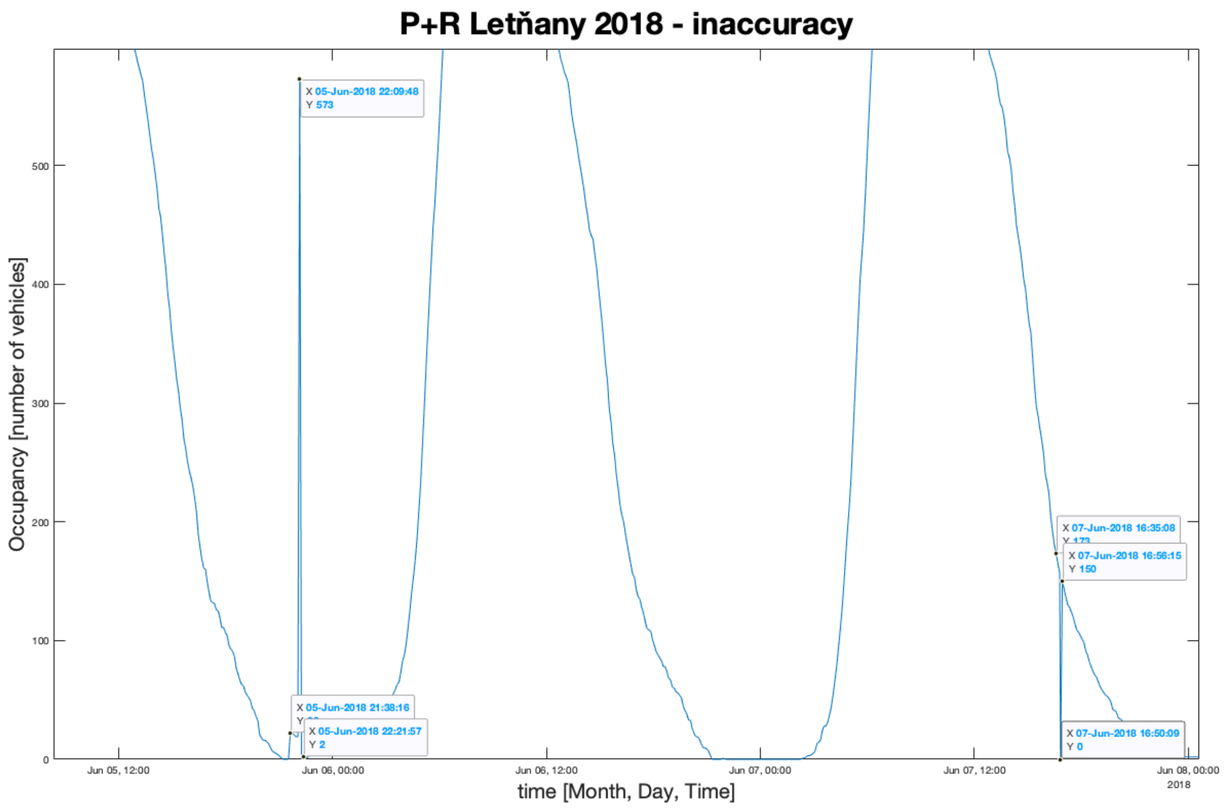
- short-time outage
- long-time outage

The short-time outage represents error or inaccuracy in a little number of consecutive records. The little number is based on the frequency of records, as follows:

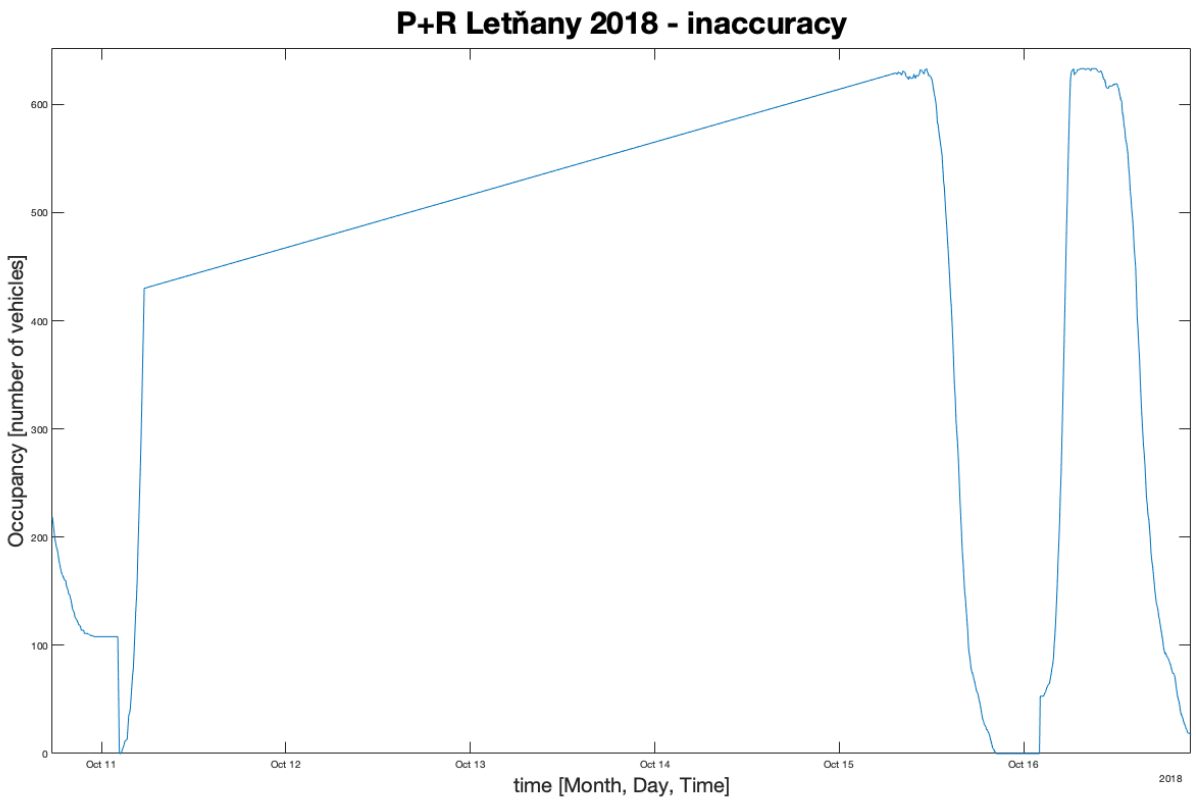
- P+R Prague: at a maximum consecutive 5 records (30 minutes of incorrect data)
- ZPS Prague: at a maximum of 0 records because no record is equal to no parked car (there is zero tolerance)
- Parking garages Norrköping: at a maximum of 1 record (30 minutes of incorrect data)

In the case of ZPS Prague, there is no information about the error. This error cannot be "repaired". However, errors of P+R Prague and Parking garages Norrköping can be "repaired". Based on the data analysis the inaccurate data can be deleted or recalculated. For the recalculation purpose, is used function *movmean* in the Matlab application. This function computes the moving average with a set size of the moving window.

The long-term outage represents more errors or inaccuracies than it is mentioned for short-time outage or if there is more than one short-time outage in a row. This data is irreparable and brings an error rate to the prediction model. This data is deleted.



Graph 1: Data cleaning – Short-time outage in the P+R Letňany 2018



Graph 2: Data cleaning – Long-time outage in the P+R Letňany 2018

3.4 Principle of the system

The principle of the proposed system is described in this section. At first, the major modules are described then the use cases of the modules and the necessary activity diagrams.

3.4.1 Major modules

Diagram 1 below introduces the major modules of the system and their connection to each other. There are 8 modules (4 out of scope and 4 in scope).

Out of scope:

- User
- ZPS Prague
- P+R Prague
- Parking garages Norrköping

In scope:

- Database
- Set destination
- Prediction
- Evaluation/Management

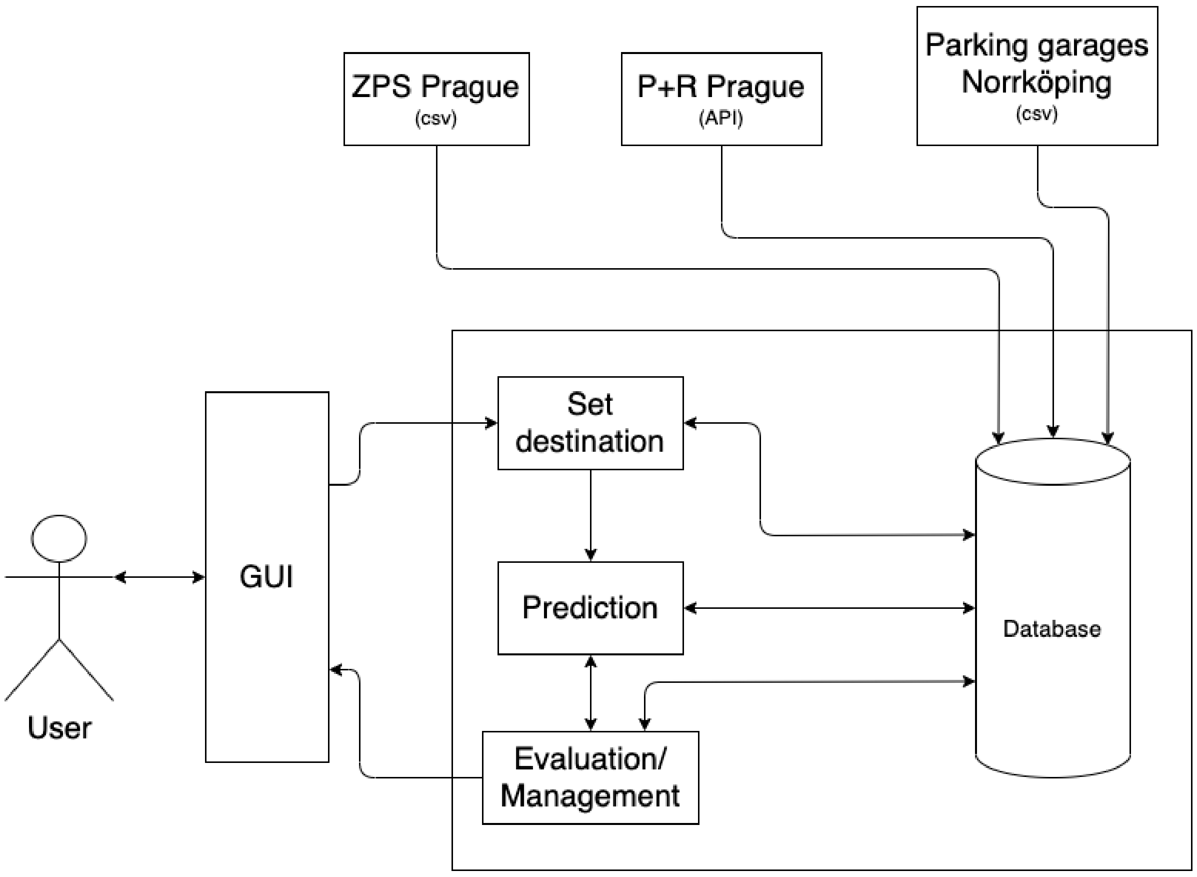


Diagram 1: Major modules

Out of scope

The user of this system should be the driver of a personal vehicle who wants to be at the destination in the next 15 or 30 minutes. There is no need for user registration. The user is connected to the system via GUI (Graphical User Interface). The GUI is used as a navigator in the systems that helps the user with the orientation in the system – to get the information about the future occupancy at the destination. The user has to choose its destination's parking facility. Then the system calculates the occupancy, and the user gets the results and recommendation.

Other out-of-scope major modules represent data input to the system. It is data input of ZPS Prague, P+R Prague, and Parking garages Norrköping. These modules create a database.

There is a need to have the following information: location, date and time, capacity, occupancy in time. Each data input is described in the Data structure chapter.

In scope

The database module – Because the structure of each data source is a little bit different, the database is divided into three parts. Each part is based on a specific data source.

The Set destination module represents the user's choice of the parking facility. In the case of ZPS Prague, this module chooses other parking facilities in the very near surroundings. As the very near surroundings are mentioned other paid parking zones located within walking distance (approximately 5-7 minutes) from the original destination.

The Prediction module represents the prediction model itself. This module gets information about the destination (and near surroundings) from the Set destination module. The prediction module looks to the database for the historical data and loads them. Based on the historical data and the date the prediction for the chosen parking facilities is done.

The Evaluation/Management module represents the decision-making module. The results of the prediction module (and the input for the Evaluation/Management module) are the number of possible free parking spaces. This number is compared with the capacity of the parking facility and with the free parking spaces in the predicted time plus/minus 3 minutes. The comparison with the capacity is done because there is a need for the context of the parking facility (the rate of free and occupied parking spaces). The capacity information is obtained from the database module. The comparison with the plus/minus 3 minutes is done for the context of the occupancy trend of the parking facility in chosen time. The plus/minus 3 minutes prediction of the predicted time is done by cooperation with the Prediction module. The plus/minus 3 minutes can be changed to plus/minus 1 timestep (based on the frequency of the data).

3.4.2 Actors of the system

There are several actors in the proposed system. The introduction of the actors is set to several categories: actors connected via GUI; actors provide the data to the system; the system's actors.

Actors connected via GUI

The User and the Administrator & maintenance are the only ones who are connected via GUI.

The User is the primary actor because for him/her is the whole system designed and created. A user uses this system to ride to another place (his/her destination). GUI is used for setting the destination.

The administrator & maintenance is there for making the system work – they do some corrections of the calculations, adding other parking facilities, update the GUI, etc.

Actors provide the data to the system

The data sources are also actors. For this case, these actors are the following: ZPS Prague; P+R Prague; Parking garages Norrköping. These actors are databases accessible to the public that contain parking data.

System's actors

These actors are modules inside of the system (major modules). Their descriptions are above.

There is a database that aggregates data from parking facilities. The next actor is the Set destination module. This actor uses the destination from the user and in the case of paid parking zones, it chooses another nearest zone. Prediction actor provides the computation. The last actor is the Evaluation/Management. This actor provides the estimation of the probability that the driver will park at the destination. In case of not, it provides a recommendation for the route.

3.4.3 Use case diagram

All actors who were introduced before are presented Diagram 2 below. The icon of a human represents the actor. In the diagram below is distinguished the category of the actor by the color of the icon (Actors connected via GUI = Red; Actors provide the data to the system = Green; System's actors = Orange). The numbers in the name of the use case (e.g., UC1) represent the ID of the use case. It is not the sequence.

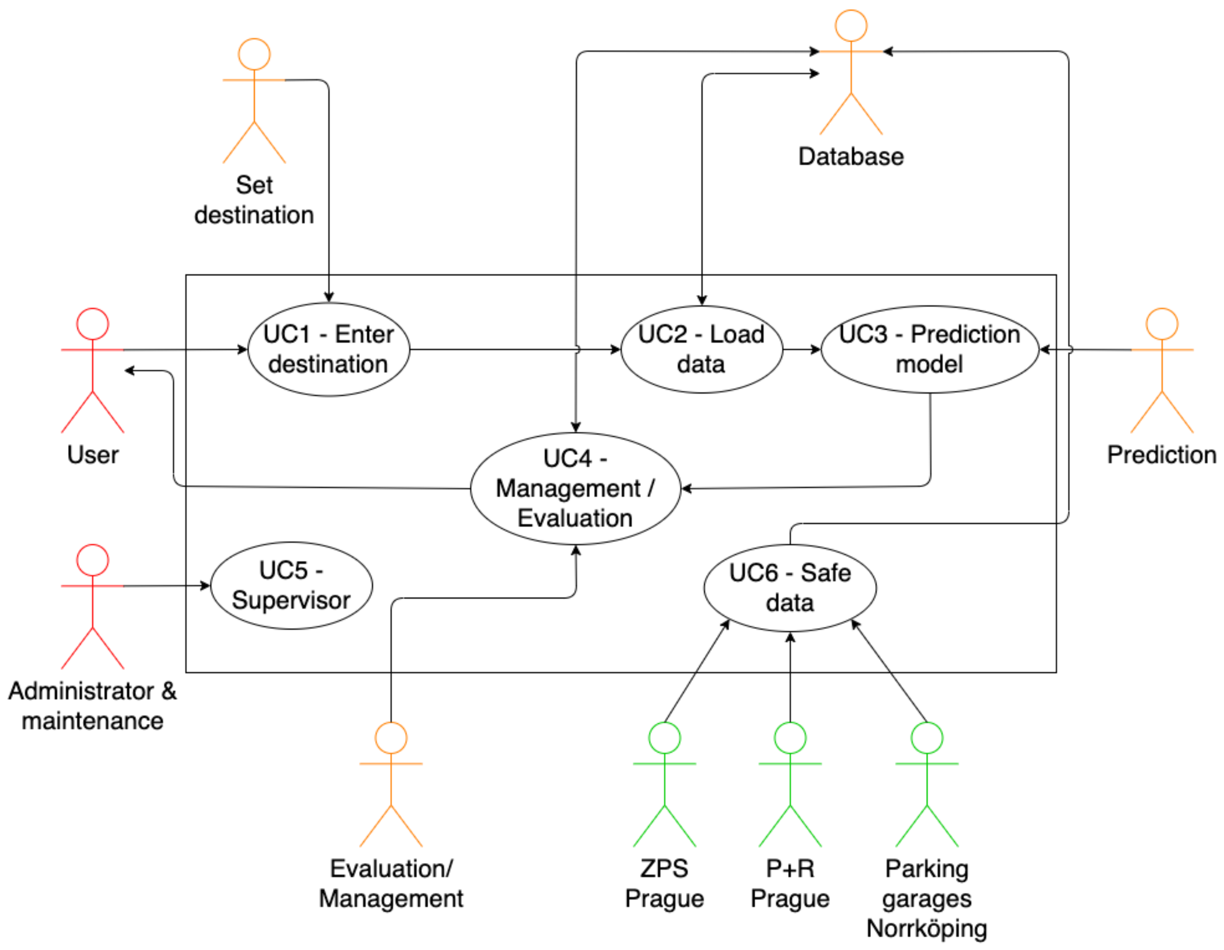


Diagram 2: Use case diagram

UC1 – Enter destination

This use case is started by the user. The user has to enter his destination (choose from a list of possible parking facilities). This step starts the whole system.

UC2 – Load Data

This use case chooses in the database the historical data of the selected parking facility and loads them. In the case of the parking zone, this use case selects the nearest zones and loads their data too.

UC3 – Prediction model

The UC3 gets data from the previous use case (UC2). This historical data is used for the prediction for the next 15 and 30 minutes. Also, there is another computation – for 12 and 18 minutes and for 27 and 33 minutes. These plus/minus 3 minutes are used as a control prediction for the following use case (UC4).

UC4 – Management/Evaluation

This use case is used for the management of the system. This system compares the free predicated parking spaces with the capacity and with the plus/minus 3 minutes. Then these results are evaluated. Based on the evaluation there is also a recommendation for the user. The recommendation can be as following: the parking facility is free and the user can park here, or there is no free parking space, so it is better to use another transport mode or use another parking facility.

UC5 – Supervisor

Administrator & maintenance supervise the entire system. It means that they check the functional status of the system. They can add another parking facility, upgrade the prediction model, etc.

UC6 – Safe data

This use case is used for saving historical data from the parking facilities. These data are saved into the database.

3.4.4 Activity diagrams

In this part are presented activity diagrams of use cases. At first, it is presented the diagram itself, and below it is its description. There aren't activity diagrams for each use case but the important ones only.

UC1 – Enter destination

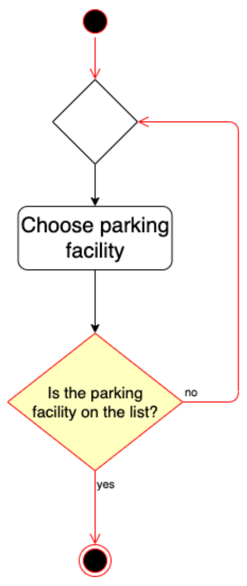


Diagram 3: Activity diagram – UC1 – Enter destination

The Set destination module starts with the system. This activity is started by the user by choosing a parking facility. The next step is to check whether it fits the list of available parking facilities. If the data doesn't fit it is necessary to set the destination again. If data fits it is getting to the next step. It is the end for this use case and the destination is the input for the following use case UC2.

UC4 – Management/Evaluation

The activity diagram is in ANNEX A for its size (it is the A3 format of the paper). The counting numbers or the number in parentheses (commented as results) are explained in the Management of the system. It is the evaluation method of each state.

The input for this use case is the results from the prediction model (UC3) – the number of free parking spaces. It means the main result: 15 or 30 minutes. And the other results are 12 and 18 minutes or 27 and 33 minutes (one timestep backward and forward).

The system checks if there is no free parking space. If not, the system evaluates it by 0 and checks if there is another parking facility in the surroundings. If there are no other facilities, the evaluation is 0, but if there is another facility, the system evaluates that one. And if there is free parking space the system continues.

If there is free parking space the evaluation counts plus 50, and then it checks whether the time is in the peak hour. If it is not in the peak hour the evaluation counts plus 20 and then checks the trends. It checks the minus 3 minutes - is there free parking space? If no, it checks the plus 3 minutes. If there is no free space, the result is 70. And it goes to the end. But if in the plus 3 minutes is free space the chance is good (the result is 85).

But if the minus 3 has free space and plus three minutes not, the chance is still good (the result is 80). And if the minus 3 has free space and plus three minutes also, there is a very high probability that you will park there (the result is 90). And it goes to the end. But if the peak hour is positive and the minus three minutes has no free space and the plus three minutes has also no free parking space there is a very low chance to park (the result is 50). But if the plus three minutes has free parking space there is a low chance (the result is 65). And if the minus 3 minutes has free parking space and the plus three minutes not, the chance is also low (the result is 60). But if the plus 3 minutes has free parking space, the result is 70.

The end is the same for each step. It checks if the probability is over the limit (the limit is 75). If not, it checks for another parking facility. If there is, the whole process starts again but for another parking facility. If there are no other parking facilities or the probability is sufficient it is the end.

4 Data analysis

This chapter introduces the data preparation and data analysis for each data source. It is mentioned above that data has different structures therefore it is divided into three segments based on the data source. Days of the week are divided into several categories based on the data analysis. In the beginning, are investigating the following categories of days:

- Monday
- Tuesday
- Wednesday
- Thursday
- Friday
- Saturday
- Sunday
- Each day for a Public holiday
- Each day for the Summer holidays

Workdays are divided into Monday, Workdays, Friday. It is because there is a thought that Monday as the day after the weekend is with higher peaks and in earlier time because drivers are going to the city. During workdays (Tuesday, Wednesday, Thursday) drivers stay in the city, and on Friday they are going out of the city.

The weekend is divided into two segments. The thought here is that Saturday is calmer because drivers already are out of the city and on Sunday they are going back to the city.

4.1 P+R Prague

There were two P+R parking lots chosen. It is P+R Letňany and P+R Černý Most 2. Both are well-known. P+R Letňany is located in the north part of the city of Prague, directly above the subway station Letňany. It is the end station of the subway line C/Red line. The capacity is 633 parking places. This parking lot is not in direct connection to the highway, but it is between highways D8, D10, and D11. P+R Černý Most 2 is one of two P+R Černý Most. It was chosen because P+R Černý Most 1 is now (first half of the year 2021) in the process of reconstruction and the capacity is significantly increased there. P+R Černý Most 2 has a capacity of 131 cars and is located near the subway station Černý Most. The subway station Černý most is the end station of the subway line B/Yellow line. It is located near the station which means that it is necessary to walk approximately 5 minutes from the parking lot to the subway station. This parking lot is located at the end of two highways, D10 and D11.

On the picture below it is shown the map of Prague and mentioned P+R parking lots are marked with the blue color.

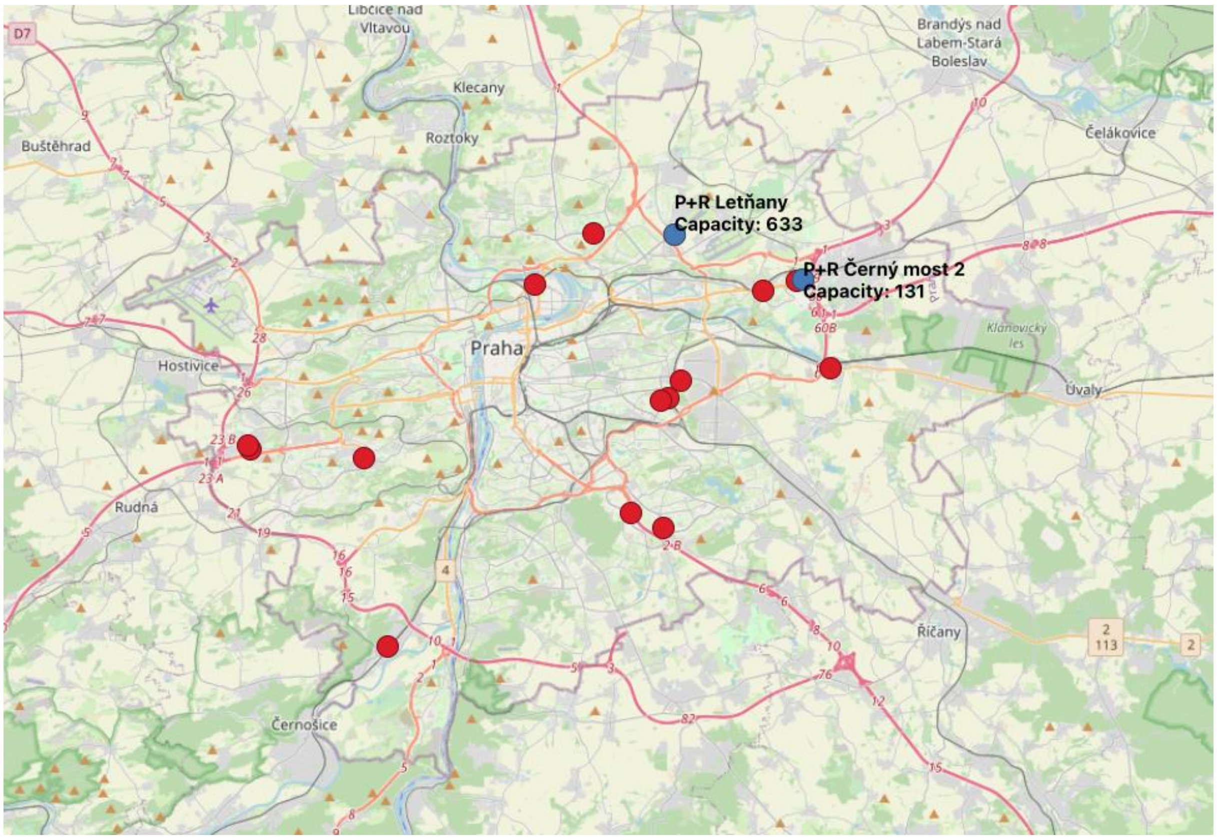


Figure 11: Map of Prague – P+R parking with marked P+R Letňany and P+R Černý Most 2 (blue) [43]

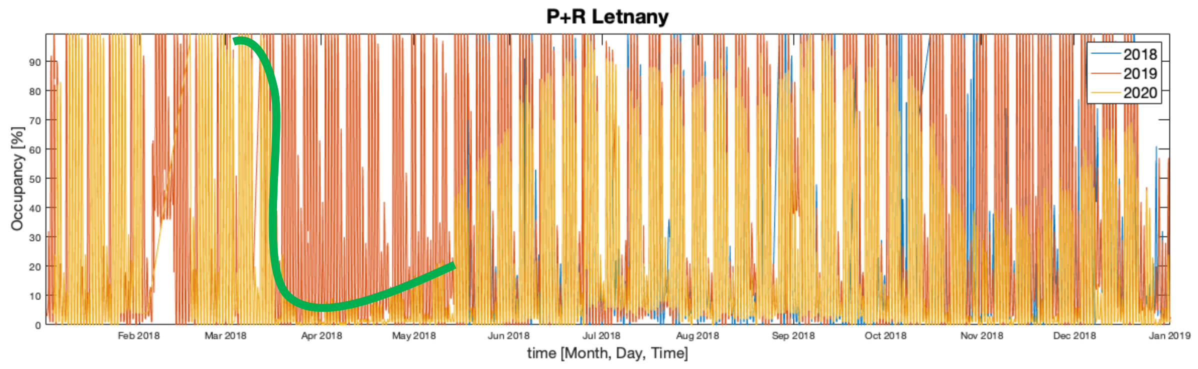
4.1.1 Data preparation

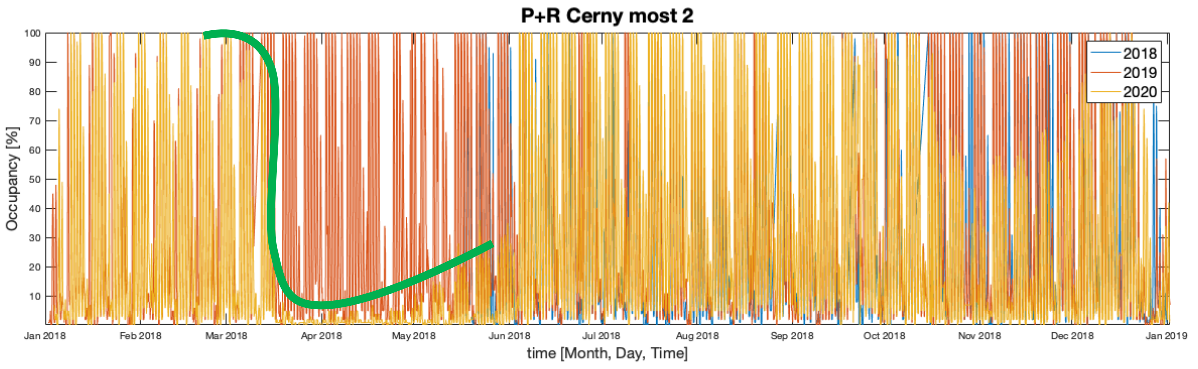
Data of P+R parking was downloaded via API. For the data download was used *Jupyter Notebook* application with the Python programming language. This tool was used for the API reading and data download. The code is in ANNEX B.

For data preparation (cleaning and filtering) was used Matlab software. The commented code is in ANNEX C.

Data is shown in the figures below. Both figures have the x-axis as a one-year duration. Data for years 2019 and 2020 is not with the correct date but with the correct weekday. For the purpose of the graph the year 2019 and 2020 was moved in time by one day (2019) and two days (2020). It was done for the comparison each weekday – weekends and workdays.

The data starts on the 18th of May 2018. In the data, it is possible to see the drop in demand in 2020 (yellow) because of the pandemic situation. It is highlighted by the green curve. In the P+R Letňany, the lost demand was during the rest of the year. There is a drop in the demand in October and November, also because of the pandemic situation. Therefore, the end data for data analysis is the end of February 2020 (29th of February 2020).

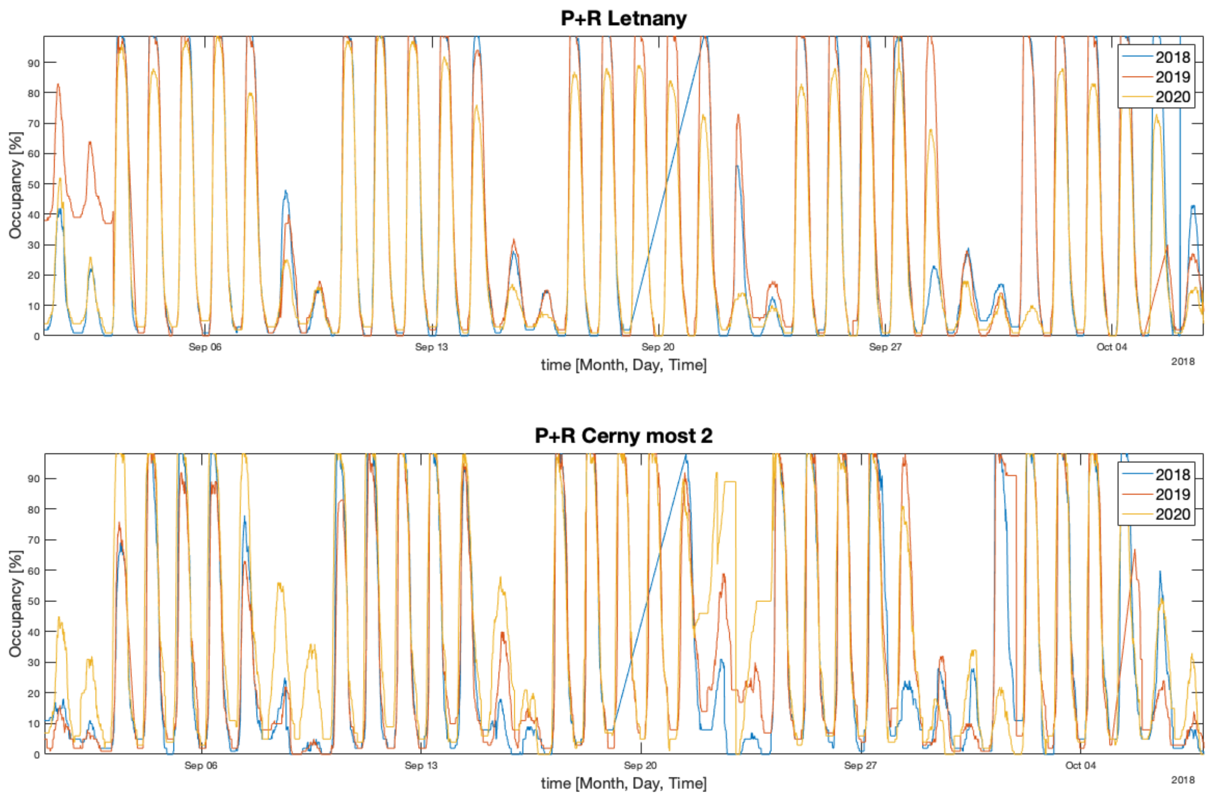




Graph 3: P+R Prague year overview

Note – Legend: blue = 2018; red = 2019; yellow = 2020; green = highlighted drop of the demand because of the COVID-19 pandemic

In the detailed view (picture below) it is possible to see that there are inaccuracies in the data (inaccurate values/peaks or outage for a few days). However, the days (Monday to Sunday) have a trend of occupancy. These trends are analyzed below.

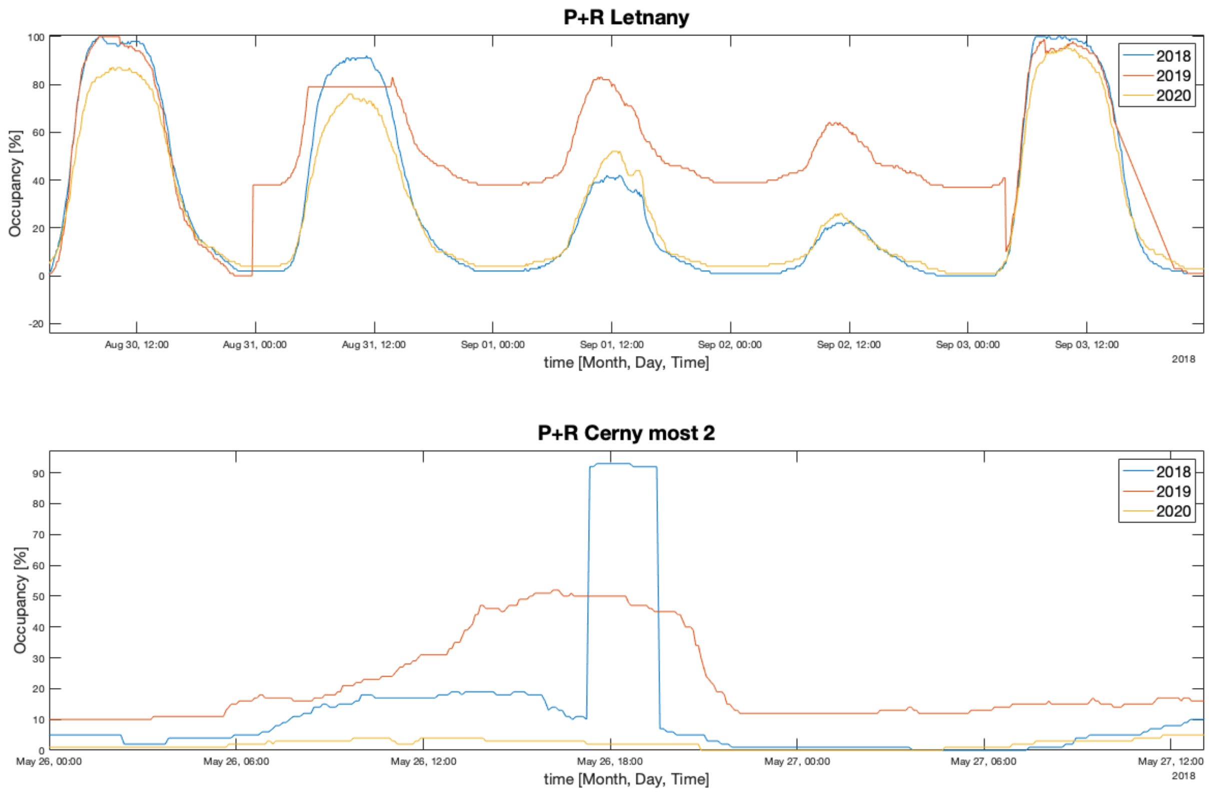


Graph 4: P+R Prague month overview

Note – Legend: blue = 2018; red = 2019; yellow = 2020

Data cleaning

First, there were observed outliers and time outages in the data. There were also observed, "jumped data". It is presented in the figure below. There is possible to see the trend but for an unknown reason, the data are jumped to different values. This problem was investigated and for the data for P+R Černý Most 2, this problem was correctly fixed. But data for P+R Letňany reported a few problems together and the fixing process from the first P+R was modified. Unfortunately, the complexity of the problem in the P+R Letňany did not allow the correct fixing of the problem. The only way how to correct the data is by manual work.



Graph 5: P+R Prague – Jumped data

The next part of the cleaning process was deleting outliers. The definition of an outlier for this case is:

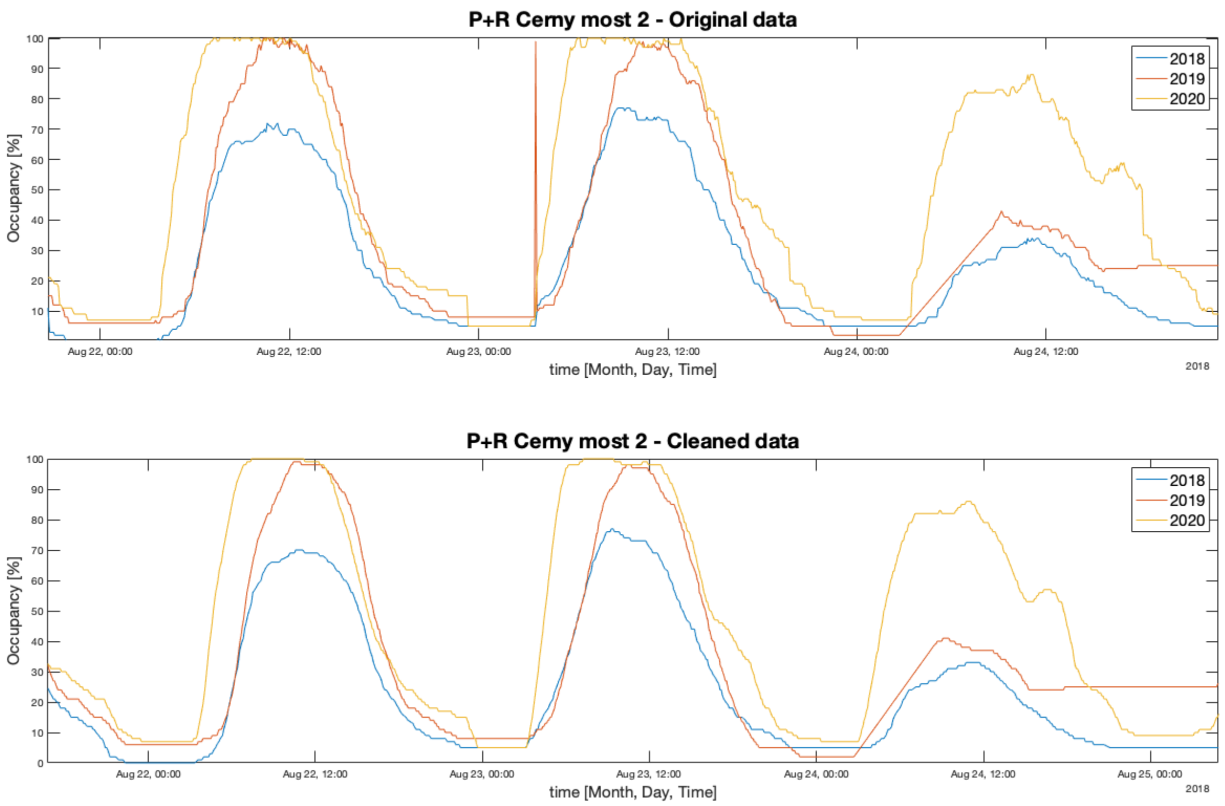
the difference in the number of free places between the i-th and the i+1 record is higher than half of the capacity of the parking lot

and

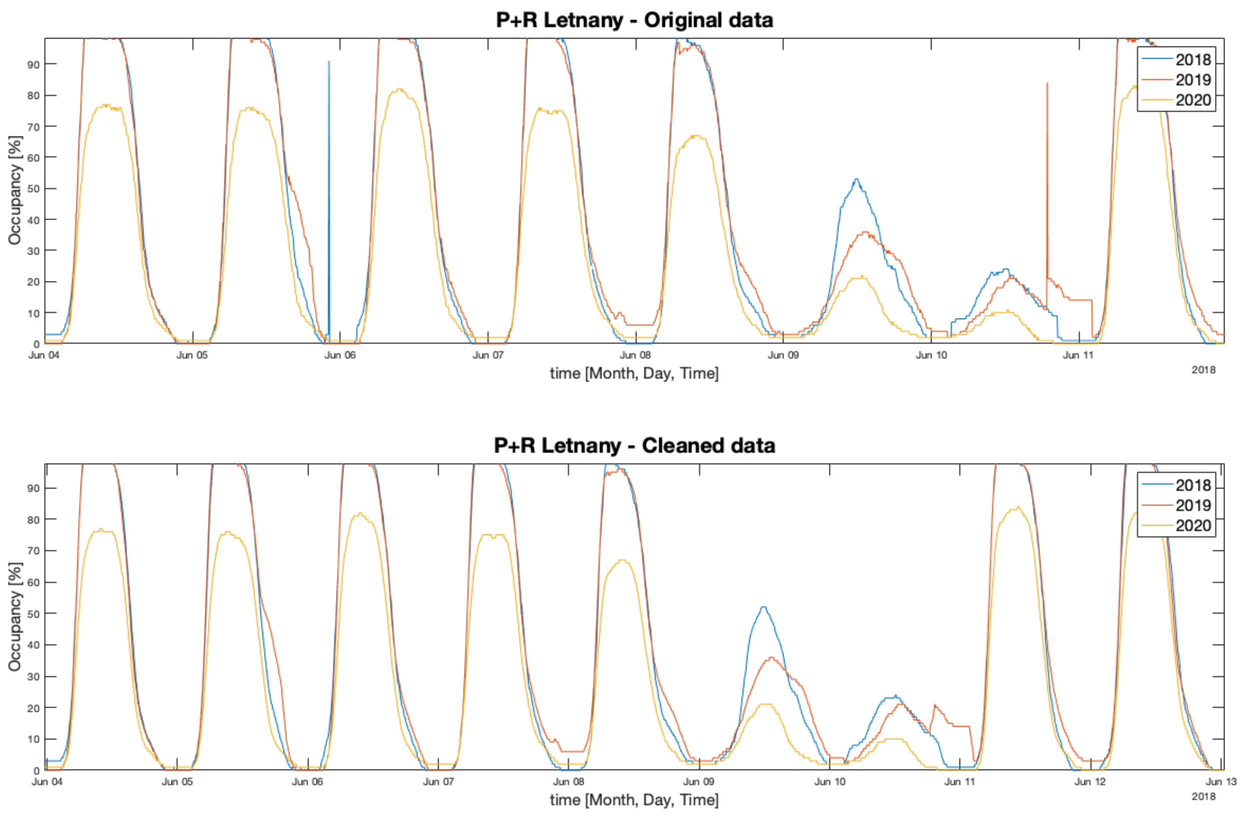
the difference in time between i-th and i+1 record is lower than 15 minutes

If there was observed an outlier this record was deleted.

The next part of the cleaning process was the smoothing process that helps to balance the beginning and end of the jumped data. For this purpose, was used the *movmean* function that calculates the mean of the floating window. This window was set to 30 minutes in both directions – it means 11 steps overall. This function was used for the consecutive timestamp only. It means that if there was a time distance higher than 30 minutes it was another time segment for which was done another *movmean* computation. The figures below show the original and cleaned data for both parking spaces.



Graph 6: P+R Černý Most 2 – cleaned data example



Graph 7: P+R Letňany – cleaned data example

Data filtering

The data filtering process is focused on the individual days of the week. Because of the irregularity in the minute timestamp (such as 06; 11 and 05; 10), the minutes were rounded to the exact 5-minute interval (for example 5; 10). And then for each timestamp record was generated the day of the week in the following code:

- name of the day (such as Mon)
- name of the day + SH (such as MonSH) – the day during summer holidays (July and August)
- name of the day + PH (such as MonPH) – the day for a public holiday
- name of the day + SH + PH (such as MonSHPH) – the day during summer holidays (July and August) and public holiday

For each day a table was created and saved as a CSV file. Then tables were used for the graphical comparison of the days. These figures are in ANNEX D because of their size. It requires an A3 paper format in the landscape. The figures were done by the *mesh* plot function in Matlab. The name is the name code, the x-axis represents the number of these days (weeks) in the time selection, the y-axis is a 5-minute interval from 00:00 to 23:55, the z-axis (colors) represents the number of free places (yellow is the maximum number of free places and dark blue is the zero number of free places). There is one code name added, it is 1day+name day. It is because this day appears in the data only once. So, this was done for the easier plot process.

4.1.2 Data evaluation

Based on the graphical comparison the Tuesday, Wednesday, and Thursday are very similar, therefore these days can be connected to one table, which means that data for these days can substitute/supplement each other. Monday has a little faster morning approach therefore it will be separate. Friday is very different and the same is for Saturday and Sunday. These days will be separate. The same works for summer holidays days. There is not enough data for separating each public holiday day. Therefore, public holidays will be merged into one table.

There are few inaccuracies in the standard days, it could be done by the data inaccuracy, or in the case of Monday, Tuesday can be a public holiday and therefore Monday is another day off. The same works for Thursday as the public holiday and Friday as the day off.

This data represents also driver's behavior – at what time they are approaching each parking lot. The behavior is standard with morning and afternoon peaks for workdays and easy for weekends and something between is for Fridays.

4.2 ZPS Prague

The ZPS Prague is done as one main paid parking zone (chosen was P7-0158) and other nearest zones in the walk distance (approximately 5-7 minutes). Therefore, other zones are within the air distance of 300 meters. These zones were found by the API. For this purpose, was used Postman application with the following code:

https://api.golemio.cz/v2/parkingzones/?latlng=50.10271354204963%2C14.446911303403967&range=300

Where 50.10271354204963%2C14.446911303403967 represents the latitude and longitude of the main parking zone and range=300 represents air distance from the coordinates.

The answer is a JSON file with coordinates and capacity of each parking zone. There were resident parking zones and mixed parking zones in the answered file. The resident parking zones were deleted from the answer because the commuter is not allowed to park there for a couple of hours. The commuter can use mixed zones only. Therefore, it is focused on them.

The surrounding parking zones are: P7-0134, P7-0179, P7-0156, P7-0161.

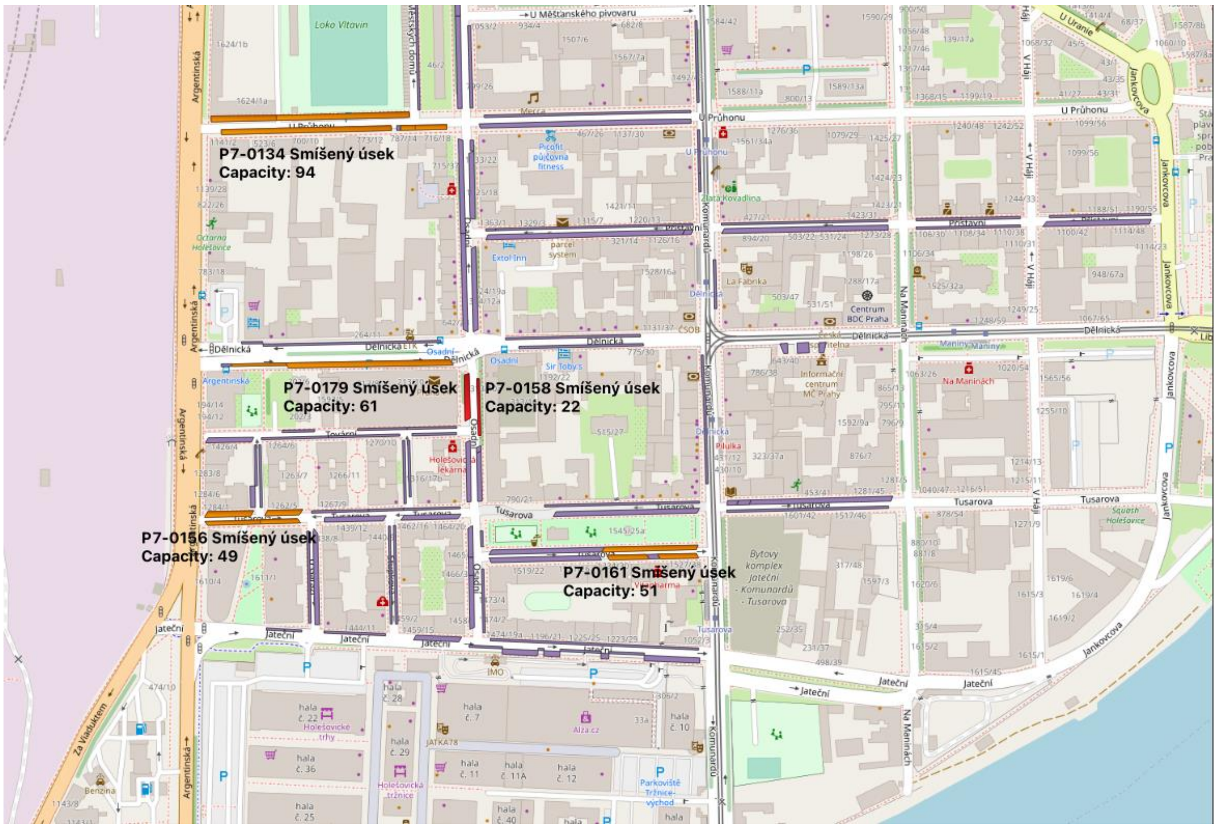


Figure 12: Map of Prague (Prague 7 – Holešovice) with ZPS

Legend: Red – main ZPS – P7-0158, orange – compensation ZPS for the main one (P7-0134, P7-0179, P7-0156, P7-0161), purple – all surrounding zones.

4.2.1 Data preparation

Data of ZPS Prague were downloaded as CSV files for each quarter of a year from January 2018 to December 2020. These files are available as open data at the Prague open data portal.

For data preparation (cleaning and filtering) was used Matlab software. The commented code is in ANNEX C.

This data is very different from the P+R parking data. It is not aggregated as a number of taken or free places in time but as sold tickets in days. Each ticket/record includes information about the valid parking time.

The data starts at the beginning of the year 2018. Based on the information about the pandemic from P+R parking the end for the data analysis is the end of February 2020 (29th of February 2020).

Data filtering – Paid parking zone

The data is aggregated into files for year quarters. But these files contain each paid parking zone in Prague. Therefore, it is necessary to filter the data for paid parking zones that we are interested in. The data in CSV files were loaded into the Matlab environment and connected into one table. Then, the paid parking zones that we are interested in were filtered into separated tables. These tables were saved as a new CSV file for each paid parking zone.

Data cleaning

Inaccurate records based on the date were observed. The date record was inaccurate and therefore these records were deleted. Unfortunately, these dates cannot be repaired because there is no information about valid parking time in this data. Each problem like this leads to deleting data without the opportunity of repairing or replacing it.

Data filtering – Days

The validity time was recalculated into the daytime segments. It means that for each ticket a record was created from 00:00 to 23:55 with a 5-minute step. In the record were 1 if the parking time was valid and 0 if not. Then each day was summarized. So, the end information is how many tickets were sold (how many parking places are taken) for each 5-minute period of the day.

As the following part was the filtering process of the data above to the individual days of the week. For each timestamp record was generated the day of the week (as for the P+R Prague) in the following code:

- name of the day (such as Mon)
- name of the day + SH (such as MonSH) – the day during summer holidays (July and August)
- name of the day + PH (such as MonPH) – the day for a public holiday
- name of the day + SH + PH (such as MonSHPH) – the day during summer holidays (July and August) and public holiday

For each day a table was created and saved as a CSV file. Then tables were used for the graphical comparison of the days. These figures are in ANNEX D for their size. It requires an A3 paper format in the landscape. The figures were done by the mesh plot function in Matlab. The name is the name code, the x-axis represents the number of these days (weeks) in the time selection, the y-axis is a 5-minute interval from 00:00 to 23:55, the z-axis (colors) represents the number of free places (yellow is the maximum number of free places and dark blue is the zero number of free places). There is one code name added, it is 1day+name day. It is because this day appears in the data only once.

4.2.2 Data evaluation

Based on the graphical comparison of the data the days are very similar (almost the same). The very details create the differences between them. This behavior can be caused by the same drivers, that they pay there every day for the same amount of time (work there), or by car-sharing vehicles. Therefore, this behavior is necessary to investigate more. Unfortunately, data are from the past and therefore there is no chance to verify the data. The pandemic situation is in progress these days (April 2020) and the behavior of drivers is a little bit different. Therefore, the data cannot be verified at this time. Based on this knowledge the days of ZPS Prague will be separated for the prediction. It is also because of the variability and not significant trends there. There is not enough data for separating each public holiday day. Therefore, public holidays will be merged into one table.

4.3 Parking garages Norrköping

Two Parking garages in Norrköping were chosen. It is P-hus Spiran and P-hus Vårdtornet. Both garages are located in the city center. However, Spiran parking is part of the Spiran shopping center that is placed in the surroundings of other shopping centers. Its capacity is 388 vehicles. In the Vårdtornet's surroundings are restaurants, museums, and housing. Its capacity is 84 vehicles. Based on the different surroundings of each parking garage different usage of them can be expected.

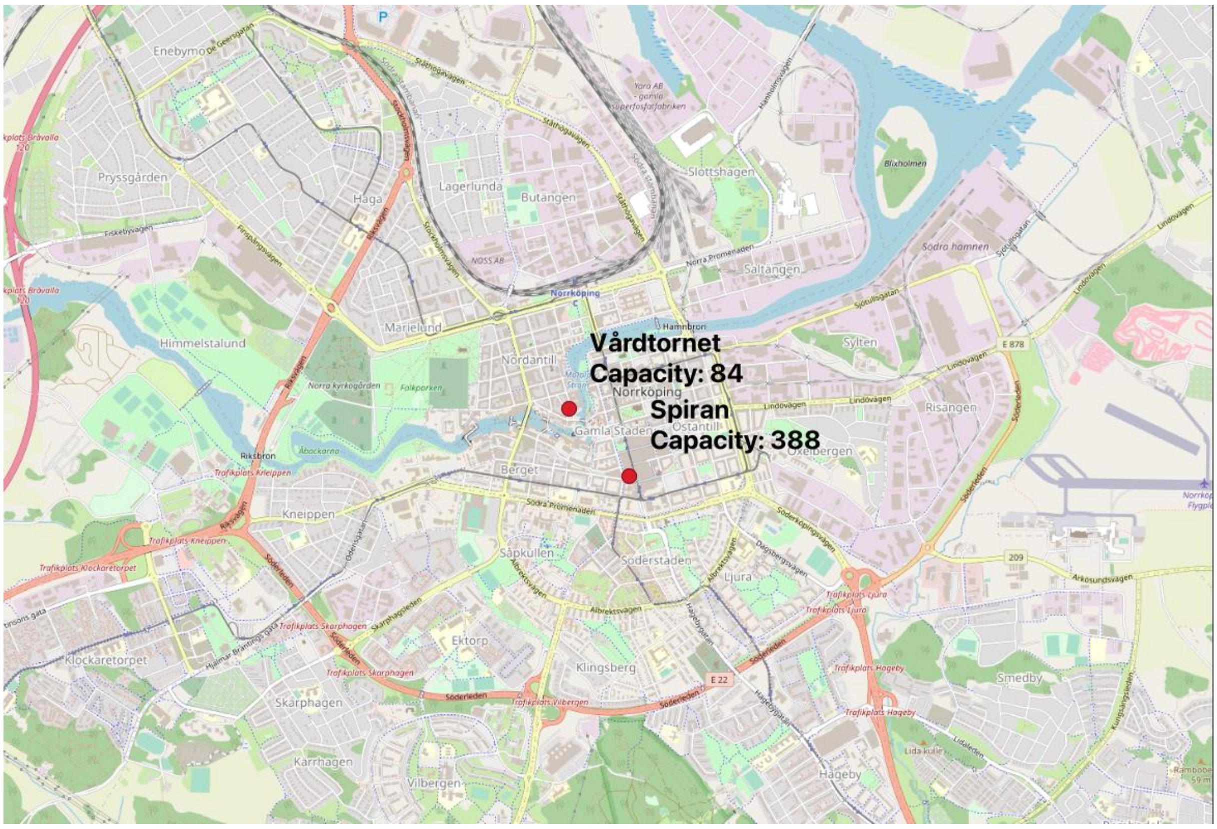


Figure 13: Map of Norrköping with Parking garages

4.3.1 Data preparation

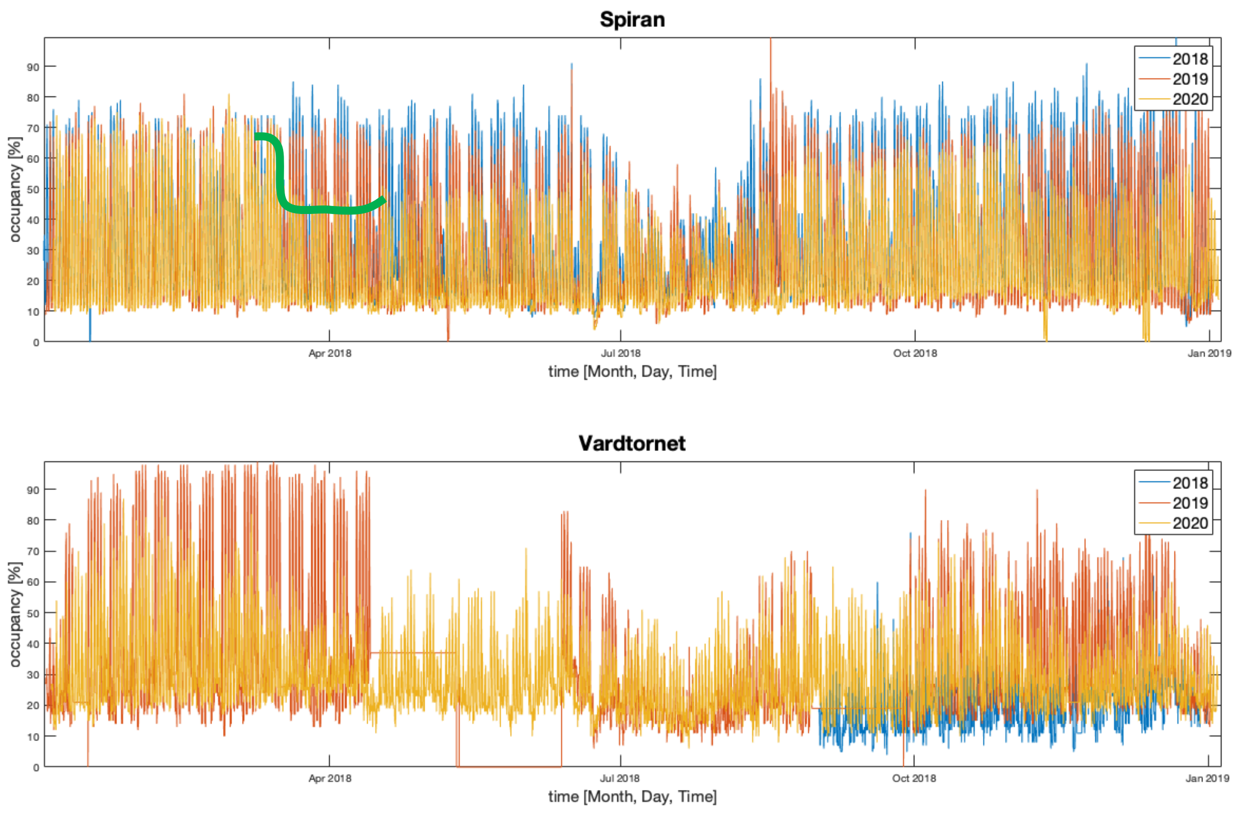
Data of Norrköping garages were downloaded via Infracontrol management center. It was downloaded manually in the XLSX file (MS Excel file).

For data preparation (cleaning and filtering) was used Matlab application. The commented code is in ANNEX C.

Data is shown in the figures below. Both figures have the x-axis as a one-year duration. Data for years 2019 and 2020 is not with a correct date but with a correct weekday. For the purpose of the graph the year 2019 and 2020 was moved in time by one day (2019) and two days (2020). It was done for the comparison of each weekday – weekends and workdays.

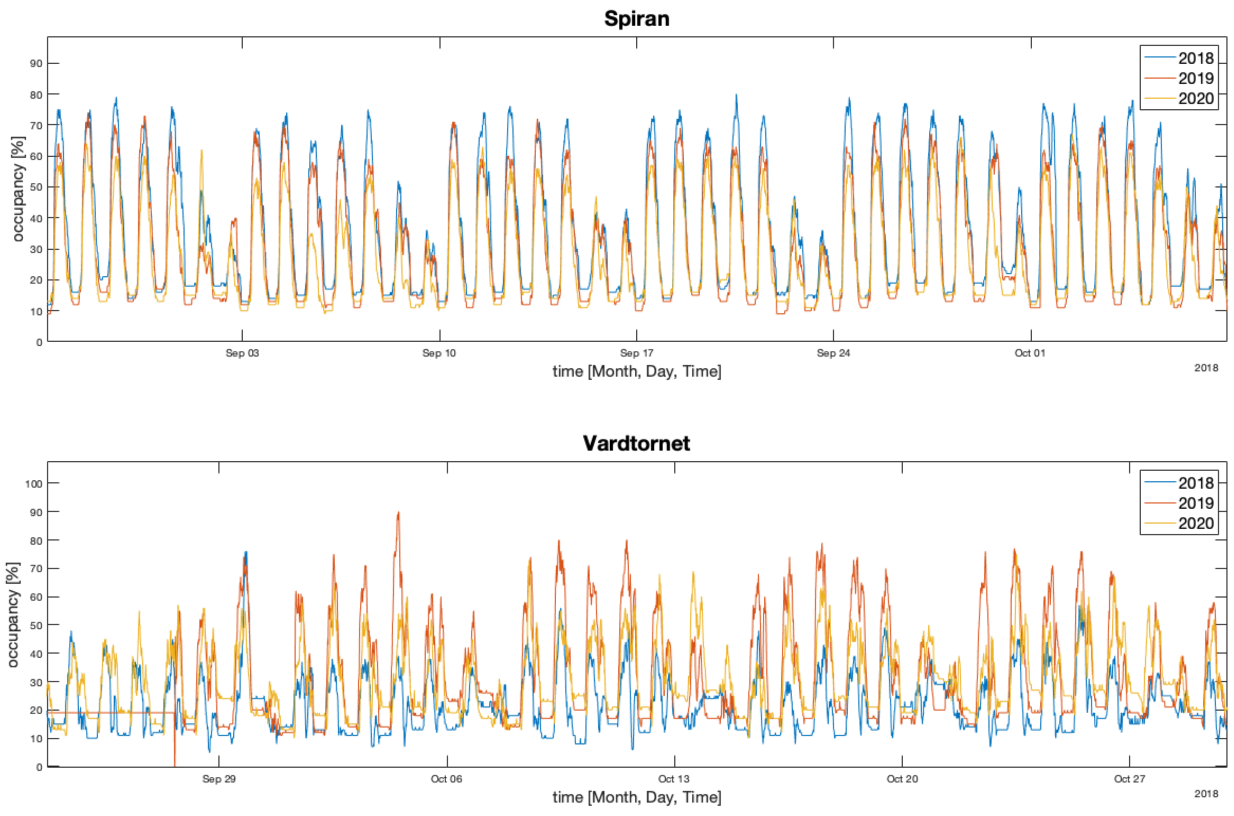
The data starts on the 1st of January 2018 for Spiran and on the 1st of September 2018 for Vårdtornet. It is expected that the Vårdtornet was opened that day. In the data of Spiran, it is possible to see the drop in demand in 2020 (yellow) because of the pandemic situation. It is highlighted by the green curve. The data of Vårdtornet is not stable as it is for Spiran. There is also untrusted data for almost two months in 2019 (red) from April until June. It is hard to find the drop in demand because of the pandemic because 2020 (yellow) has generally lower demand. However, in both datasets it is possible to see the summer vacation drop and in the detailed view, it is possible to see the trend of each day across the years.

The data for data analysis ends in February 2020 (29th of February 2020) because of the pandemic.



Graph 8: Norrköping garages year overview

Note – Legend: blue = 2018; red = 2019; yellow = 2020; green = highlighted drop of the demand because of the COVID-19 pandemic – it is the same for all the graphs

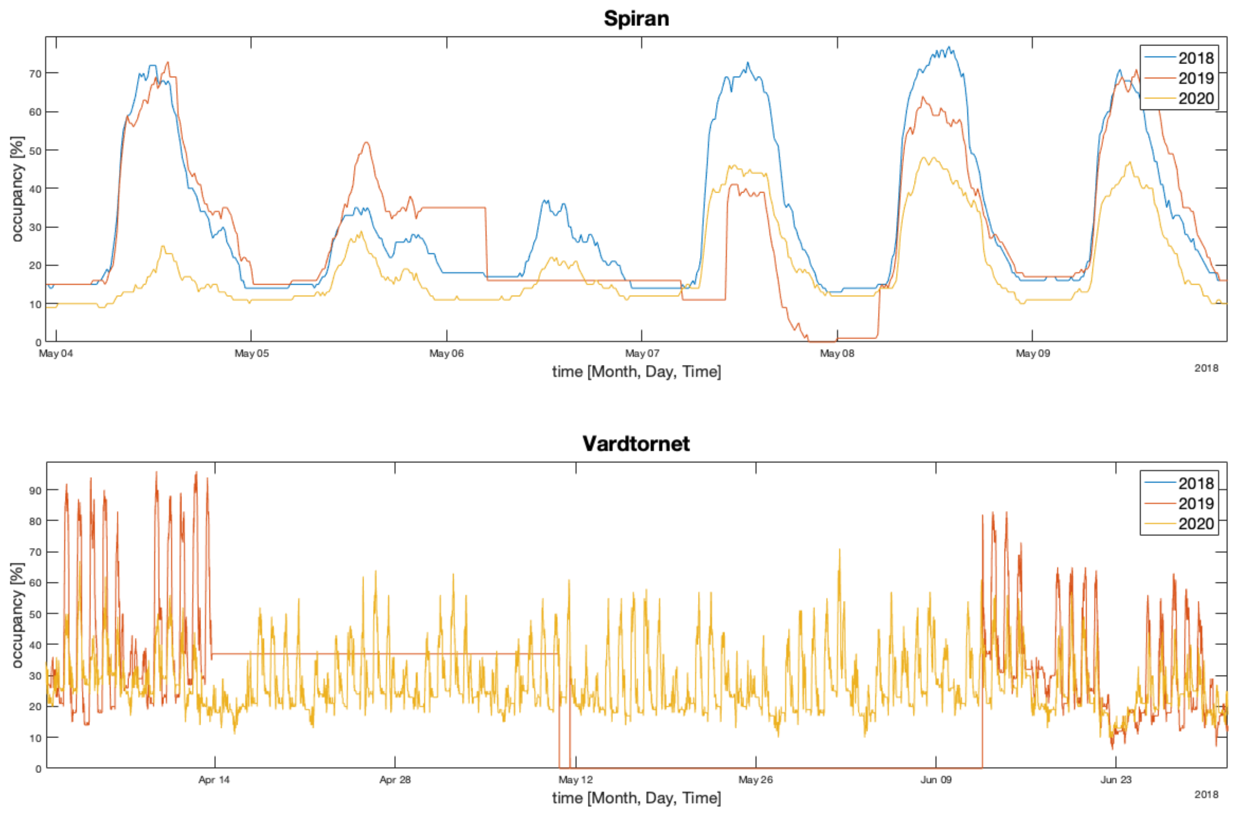


Graph 9: Norrköping garages month overview

The data is not getting to the 0 % occupancy (even at night). So, the idea is that there are parked residents.

Data cleaning

First, were investigated outliers and time outages as in the P+R Prague data. For this data, there were just a few outliers based on the same rules as in the P+R Prague case. Although the parameter of the data outlier rule was modified to correspond to the data periodicity (15-minute interval). However, untrusted data were observed. This data is shown in the graph below. Untrusted data is data without value change for an unexpected time. It means that if for more than 12 hours the value wasn't changed, we observed untrusted data. (It is possible to observe a couple of days without a change of the value.) This data was recorded and deleted.



Graph 10: Norrköping garages – Untrusted data

Data filtering

The data filtering process is based on the same principle as for the P+R Prague parking. It is focused on the individual days of the week. Because the timestamp is sometimes in minutes 15:00 and sometimes 14:59 the minutes were rounded to the 15-minute interval. Then, was generated each weekday for each timestamp as in the case of P+R Prague. The days have the same codes:

- name of the day (such as Mon)
- name of the day + SH (such as MonSH) – the day during summer holidays (July and August)
- name of the day + PH (such as MonPH) – the day for a public holiday
- name of the day + SH + PH (such as MonSHPH) – the day during summer holidays and public holiday

For each day a table was created and saved as a CSV file. Then tables were used for the graphical comparison of the days. These figures are in ANNEX D for their size. It requires an A3 paper format in the landscape. The figures were done by the *mesh* plot function in Matlab. The name is the name code, the x-axis represents the number of these days (weeks) in the time selection, the y-axis represents a 15-minute interval from 00:00 to 23:45, the z-axis (colors) represents the number of free places (yellow is the maximum number of free places and dark blue is the zero number of free places). There is one code name added, it is 1day+name day. It is because this day appears in the data only once.

4.3.2 Data evaluation

Based on the graphical comparison there is a minimum time with 100 % occupancy or close to it. This is true for both garages. The Vårdtornet does not have such stable trends as Spiran. Monday is different from other workdays because the approach time is early in the morning. Tuesday, Wednesday, and Thursday are very similar; therefore, these days can be connected to one table, which means that data for these days can substitute/supplement each other. Friday has lower occupancy than other workdays. Therefore, it will be separate such as Monday. Saturday has higher occupancy than Sunday, thus weekend days will be separated. The same is for the summer holidays. There is not enough data for separating each public holiday day. Even though the data are variable the trend is not going to the full occupancy, and therefore, public holidays will be merged into one table.

The Vårdtornet data is without any strong trend however it is not all the time with 100 % occupancy. Therefore, this data will be also used for prediction, but it can be assumed that better prediction results will be with longer observation (which is not optimal during the pandemic situation).

There are few inaccuracies in the standard workdays, it could be done by the data inaccuracy, or in the case of Monday, Tuesday can be a public holiday and therefore Monday is another day off. The same works for Thursday as the public holiday and Friday as the day off.

This data represents also driver's behavior – at what time they are approaching each parking lot. The behavior is standard with morning and afternoon peaks for workdays and easy for weekends and something between is for Friday.

5 Prediction methods

The parking occupancy datasets have as variables time and occupancy of the parking place. Therefore, there were investigated the time relationships only. Although, there are many other variables for the parking occupancy prediction, such as weather, road closure, culture or sports events, etc. These variables are not covered in this thesis, but they represent possibilities of improvement of this system.

The time variable was modified to the time of the day and the day of the week (includes summer and public holidays). Then, the null hypothesis for one parking lot was checked. The null hypothesis is used in statistics as a hypothesis that there is no effect between certain characteristics. In this case, the null hypothesis is, that time has no effect on the occupancy. Then the p-value was calculated. It was done on the cleaned data of the P+R Černý Most 2 by the *corrcoef* Matlab function. The P-values range is from 0 to 1. Values close to 0 correspond to a significant correlation and a low probability of observing the null hypothesis. Therefore, we are looking for a value under 0.05 because it indicates that there is a significant relationship. The relationships were investigated between variables *number of free places*, *date, weekday*, and *time of the day*. There was also calculated correlation coefficient R. Its range is from -1 to +1, where -1 represents a direct negative correlation; 0 represents no correlation; and +1 represents a direct positive correlation. This coefficient is a measure of the linear dependence of the variables by the Pearson correlation.

In the table below is important the first row or the first column because the table is diagonally symmetric. The number of free places has a very strong relationship with each time variable (Date, Weekday, and Time of the day) (p = 0 means so low a number that the system – Matlab, gives 0 as a result, but this p-value is not equal to zero but very close to zero). However, the correlation coefficient is weak (the highest absolute R-value is 0.1819). Therefore, it can be assumed that the relationship is not linear.

p-value (p) *correlation coefficient (R)*	**Number of free places**	**Date**	**Weekday**	**Time of the day**

Number of free places	p = 1
	R = 1
		p = 5.8226e-10
		R = -0.0156
			p = 0
			R = -0.1819
				p = 2.1770e-09
				R = -0.0151

Date	p = 5.8226e-10
	R = -0.0156
		p = 1
		R = 1
			p = 2.3779e-06
			R = -0.0119
				p = 0.8820
				R = -3.7381e-04

Weekday	p = 0
	R = -0.1819
		p = 2.3779e-06
		R = -0.0119
			p = 1
			R = 1
				p = 0.3801
				R = -0.0022

Time of the day	p = 2.1770e-09
	R = -0.0151
		p = 0.8820
		R = -3.7381e-04
			p = 0.3801
			R = -0.0022
				p = 1
				R = 1

Table 4: p-value and correlation coefficient for P+R Černý Most 2, cleaned data

As was already mentioned in chapter 2.4, for the prediction of transportation problems is popular to use the time-series approach. The time-series approach can be represented by Neural Network models. The neural networks can adequately capture the parking occupancy in time and may accurately predict it for the next 30 minutes. Another approach that was used is the regression tree. Because there are more possibilities of the prediction methods each method is tested in this chapter. The method with the best results of MSE and R-squared will be used in the system.

The MSE is an abbreviation for mean squared error. It is used for the purpose of evaluating the performance of a statistical learning method on a given dataset. It measures how good predictions are according to match the observed data – How close is the predicted response value to the true value.[48] The formula for MSE is:

$$MSE = \frac{1}{n}\sum_{i=1}^{n}\left(y_i - \hat{f}(x_i)\right)^2$$

Equation 1: Mean square error [48]

Where: $\hat{f}(x_i)$ is the prediction that $\hat{f}$ gives for the *i*-th observation.

If the MSE is small the predicted responses are very close to the true responses, and if the MSE is large the predicted and true values are very different.[48]

The R-squared statistics provide another measure of fit. It is independent of the y-scale and it takes the form of variance proportion. The R-squared explains the proportion of the variability in the response that was explained by the regression. [48] The formula is:

$$R^2 = \frac{TSS - RSS}{TSS} = 1 - \frac{RSS}{TSS}$$

Equation 2: R-squared [48]

Where: TSS is the total sum of squares $TSS = \sum(y_i - \bar{y})^2$, and RSS is the residual sum of squares $RSS = \sum_{i=1}^{n}(y_i - \hat{y}_i)^2$.

The R-squared statistic range is from 0 to 1. The result close to 1 indicates that a large proportion of the variability in the response was explained by the prediction method. A number close to 0 indicates that the prediction method did not explain much of the variability in the response. In fact, the R-squared can be also explained by the correlation coefficient. The R-squared (Coefficient of determination) is equal to the square of the correlation coefficient.[48]

Tested methods

The tested methods are:

- Polynomial regression (Quadratic)
- Regression tree
- Neural Networks

The polynomial regression represents a more general method, and it is used for comparison with more complex prediction methods. This method is based on the standard linear model:

$$y_i = \beta_0 + \beta_1 x_i + \epsilon_i$$

Equation 3: Linear regression [48]

but there is added a polynomial function:

$$y_i = \beta_0 + \beta_1 x_i + \beta_2 x_i^2 + \beta_3 x_i^3 + \cdots + \beta_d x_i^d + \epsilon_i$$

Equation 4: Polynomial regression [48]

Where: ϵ_i is the error term, and *d* represents the degree.

Polynomial regression can produce a non-linear curve with a large enough degree. But with a higher degree, the flexibility of the regression can become overly flexible for the training data and it leads to overfitting. The coefficients can be estimated by using least-squares linear regression.[48]

Decision trees can be used for regression and also for classification. These trees consist of a series of splitting rules, starting at the top of the tree. It is branched from the first splitting rule to two branches and then more and more branches.[48]

For the case of Neural Networks (NNs), there are often used Evolutionary Algorithms. These algorithms can solve optimization problems of the neural network.[46] Because the main aim of this thesis is to develop a system not a comparison of the prediction methods, the parameters of the NN are taken over from best practices for the same or similar purpose (prediction of parking occupancy).

Generally speaking, NNs are models inspired by the human brain and they are able to learn. Because in this case, enough data is available the supervised learning is done. Supervised learning is based on the supply of training data (input – X, and output – Y) and the NNs provides a new output (Z). This new output is compared to the original output (Y) by the error function (distance). By the iterative process, this error is minimized.[46]

5.1 Comparison

At first, the representative of each group of parking places (P+R Prague, ZPS Prague, Parking garages Norrköping) was selected. On these representatives are compared different predictive methods. It means that the predictive methods are trained and evaluated on these selected datasets. It is P+R Černý Most 2 for P+R Prague, ZPS-P7-0158 (the main paid parking zone) for ZPS Prague, and Spiran for Parking garages Norrköping. The next step was to get together the data for workdays (Tuesday, Wednesday, Thursday), the same for the workdays during summer holidays, and the data for the public holidays. For the NN there was necessary to separate train and test versions of the data. It means that the dataset was divided into two parts – training and test. The test version contains data for one day only. The training version contains all the data without the test day. Generally, the test day was chosen as the last day in the dataset. The code is in ANNEX E.

Polynomial regression (Quadratic)

For the Polynomial regression (Quadratic) and Regression tree was used the Regression learner app in Matlab. For the Polynomial regression (Quadratic) was chosen the Interactions Linear method with the parameters of Quadratic and Pure Quadratic regression. The Interactions Linear method was chosen for the ability of higher flexibility. The Quadratic parameters can be used for the constant term, linear terms, and quadratic terms, and the Pure Quadratic for the constant term, linear terms, and terms that are purely quadratic in each of the predictors. It was chosen because the data for most of the days are constant from the beginning of the day (midnight) to the morning peak hour where it starts to be quadratic till the afternoon peak where the data are more linear and at the evening, they are constant again.

Regression tree

For the Regression tree method was used all methods included in the Regression learner app. It is a Fine Tree (high flexibility – many small leaves for a highly flexible response function), Medium Tree (medium flexibility – medium-sized leaves for a less flexible response function), and Coarse Tree (low flexibility – Few large leaves for a coarse response function). To predict the response of any regression tree, it is necessary to follow the tree from the root (beginning) node down to a leaf node. The last node (the leaf node) contains the value of the response.

For all of these methods (Regression tree and Polynomial Quadratic regression) was used Cross-Validation with 5 folds. It is used for protecting against overfitting by partitioning the data set into folds and estimating accuracy on each fold.

Neural Networks

The Neural networks have many possibilities of parameters (such as structure, number of hidden layers, delay). Therefore, the NNs were compared first between each other to choose the one or two with parameters as good as possible.

Therefore, the NN with different parameters was tested on two datasets (P+R Černý Most 2 Monday and Workday). The assessment of the best parameters is based on the MSE and R-squared the same as the overall prediction method assessment. The chosen parameters are based on [46]. They have the expert-defined architecture in 2 layers. This parameter was taken over as the sufficient number of layers. They also have defined a maximum number of hidden neurons as 64. At first, it was tried 60 hidden neurons but after two hours of computing, this solution was denied. For the purpose of this thesis, the solution has to compute the prediction in the shortest time possible. Therefore, there were tried 10 and 20 hidden neurons as standard values offered by the Matlab Neural Net Time Series app. This app was used for the training and testing of the NNs. Another parameter of the NN is the structure. There is the possibility to pick from Nonlinear input-output and Nonlinear Autoregressive with External Input (NARX). The Nonlinear input-output predicts series y(t) given d past values of series x(t). And the NARX predicts series y(t) given d past values of y(t) and another series x(t). It means that NARX has a loop from output to the beginning. Where d is input delay. The delay allows the network to have a dynamic response to the time series input. It was tested low numbers of delay (input 1 and output 1 and input 1 and output 2). The result is a network with outputs that are produced d timesteps later.

NN parameters	**measure**	**PR_CM2_Mon**	**PR_CM_Workday**	**Overall R-squared**	**Overall MSE**
NARX: 2 layers, 10 hidden neurons, 2 delays	R-squared	0,9999	0,9998	0,9999	0,2807
	MSE	0,1909	0,3706		
NARX: 2 layers, 10 hidden neurons, 1 delay	R-squared	0,9998	0,9999	0,9998	0,2993
	MSE	0,2651	0,3335		
NARX: 2 layers, 20 hidden neurons, 2 delay	R-squared	0,9999	0,9998	0,9998	0,3757
	MSE	0,2266	0,5248		
NARX: 2 layers, 20 hidden neurons, 1 delay	R-squared	0,9999	0,9998	0,9998	0,3303
	MSE	0,1772	0,4834		
NN: 2 layers, 10 hidden neurons, 2 delays	R-squared	0,9998	0,9998	0,9998	0,4797
	MSE	0,4367	0,5226		
NN: 2 layers, 10 hidden neurons, 1 delay	**R-squared**	**0,9999**	**0,9999**	**0,9999**	**0,2165**
	MSE	**0,1307**	**0,3023**		
NN: 2 layers, 20 hidden neurons, 2 delay	R-squared	0,9432	0,9996	0,9714	1,8442
	MSE	2,6800	1,0084		
NN: 2 layers, 20 hidden neurons, 1 delay	**R-squared**	**0,9998**	**0,9999**	**0,9998**	**0,3235**
	MSE	**0,3245**	**0,3225**		

Table 5: Neural network parameters comparison

The table above represents the comparison of NN parameters. There were chosen two NNs with the highest R-squared and the lowest MSE over all datasets (marked with the bold text). The second NN with 2 layers, 20 hidden neurons, and 1 delay is not the second-best but NARX has higher computational time. Which was not the main criterion, but it affected the decision-making.

5.2 Prediction methods comparison

All the above-mentioned prediction methods were tested on the same datasets. The datasets are:

- Monday (Mon)
- Workday (Workday)
- Saturday summer holidays (SatSH)

for P+R Černý Most 2 and Spiran and

- Monday (Mon)
- Wednesday (Wed)
- Saturday summer holidays (SatSH)

for ZPS-P7-0158 (the main paid parking zone).

The table with the results is in the ANNEX F.

Based on the highest R-squared and lowest MSE was chosen the most accurate predictive method. It is a Neural network with 2 hidden layers and 10 neurons. This method has the best values and the maximum MSE is approximately 47, which means that the approximate error is 7 parking places. Overall, the higher error is for the Spiran dataset. It could be because the Spiran has 15-minute periods of data and the rest datasets have 5-minute periods of data. It means that for the whole day there are fewer records (96 per day) and the rest datasets have 288 records per day.

6 Recommendation decision-making

This chapter is based on Use case 4 – Management/Evaluation mentioned in section 3.4.3 and it relates to previous chapters Data analysis and Prediction methods. Based on this connection the decision-making about the recommendation is done. The decision-making part evaluates each combination of the prediction and recommends to the driver what to do. The Activity diagram of the decision-making is described in section 3.4.4 and the whole Matlab code is in the ANNEX G.

6.1 Step-by-step system description

There is a step-by-step description of how the whole system works in this chapter next part.

6.1.1 Selection of parking lot and prediction time

When this system is run, the user must choose the parking lot. It is based on choosing from the list. The options are:

- P+R Černý Most 2 (Prague)
- P+R Letňany (Prague)
- ZPS-P7-0158 (Prague) (the main paid parking zone)
- Spiran (Norrköping)
- Vardtornet (Norrköping)

The selection from the list also selects the right folder with the input (training) data. In the case of ZPS, it chooses more folders because of the comparison between zones.

Then the user must choose the prediction time. It could be:

- now + 15 minutes
- now + 30 minutes

By this selection, it is computed the name of the day (includes summer or public holidays) and one timestep backward and forward. The timestep differs based on location. For Prague, it is 5 minutes and for Norrköping it is 15 minutes (based on the input data). These steps are computed because of the actual trend of the parking lot. If the occupancy is increasing, stabilized, or decreasing. In the case of Prague, the timestep is 5 minutes. This is (after rounding to 5 minutes step) compliance with the law of 27§ o) of 361/2000 Road Traffic Act [49], because:

The driver must not stop and stand in a reserved car park unless it is a vehicle for which the car park is reserved; this does not apply in the case of stops and parking which do not exceed a period of three minutes and which do not endanger or restrict other road users or restrict the drivers of vehicles for which the car park is reserved.

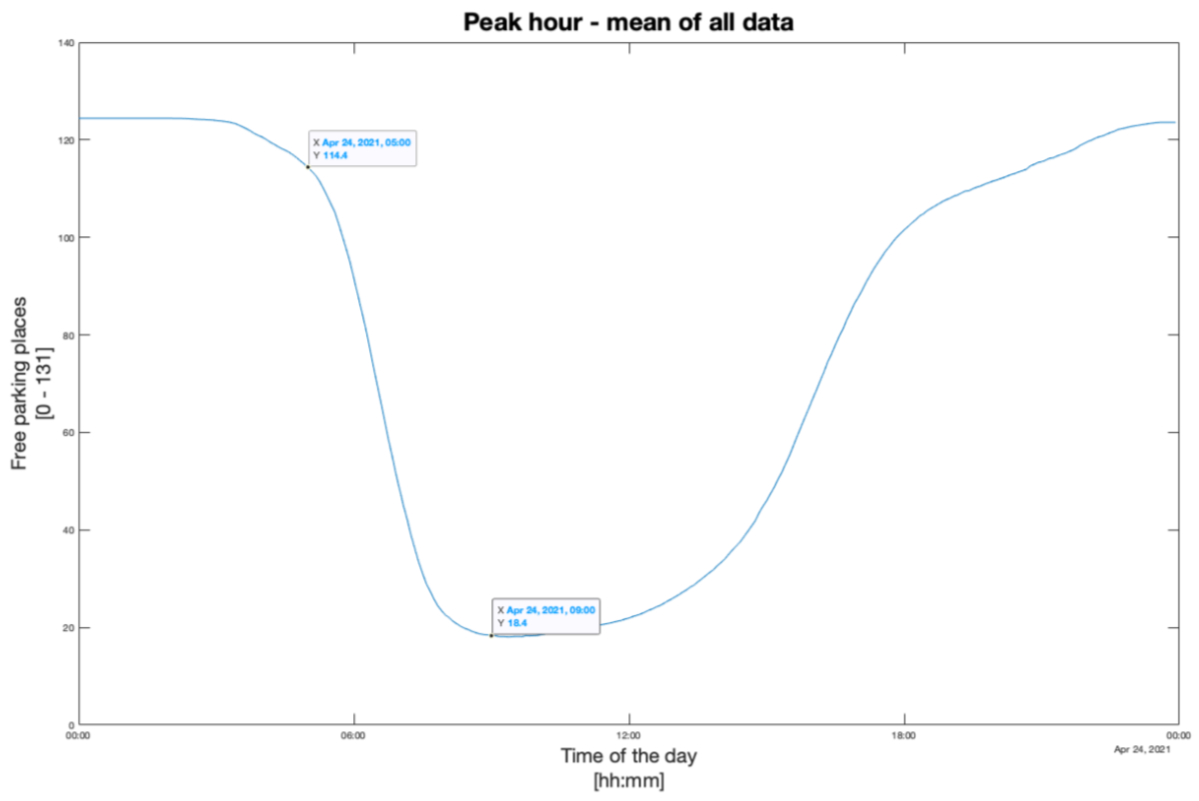
6.1.2 Peak hours

There is also defined if the predicted time is in the peak hour. It is important to know whether the predicted time is in peak hours or not because the demand for parking is much higher in the peak hours than in the regular time of the day.

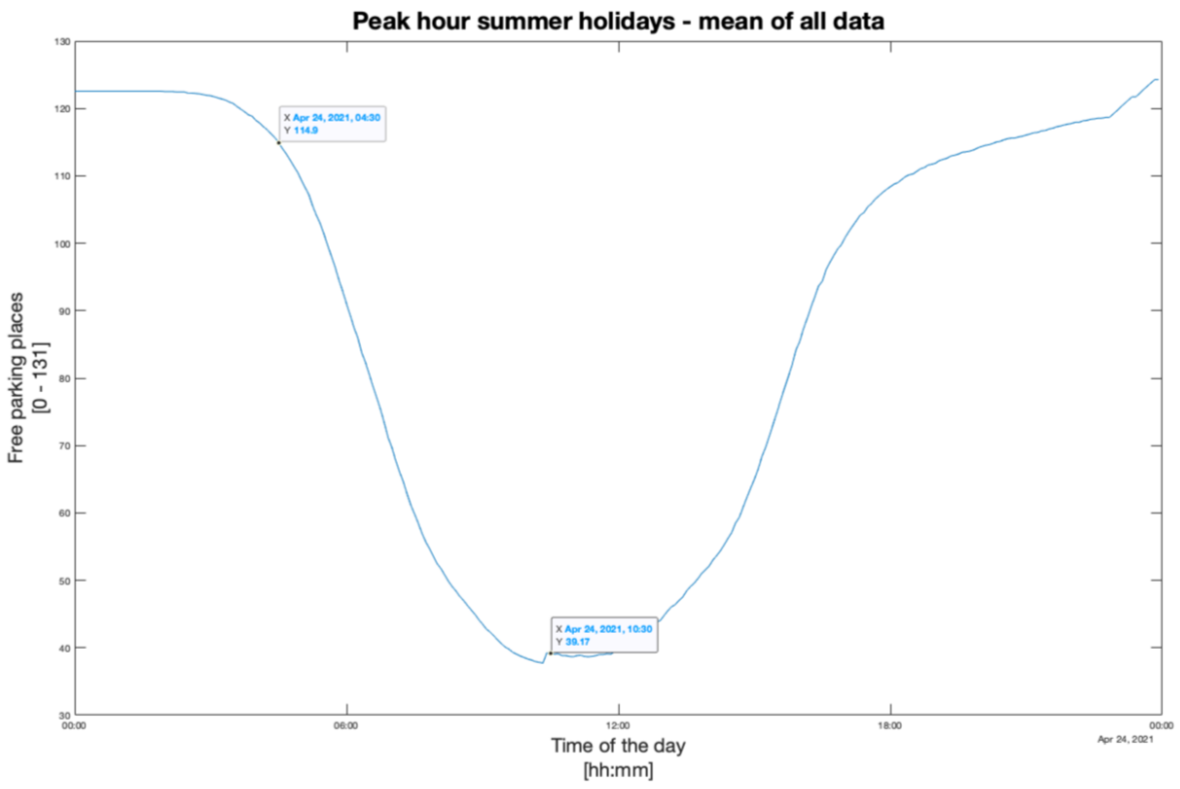
The peak hours were selected based on the Data analysis. It was discovered that for the ZPS Prague and all weekends the peak hours are not significant. But in the case of P+R Prague and Parking Garages Norrköping, the peak hours are significant. The peak hours were established for the workdays and workdays in summer holidays. For the P+R Prague it is:

- workdays – Monday to Friday -> from 5:00 to 9:00
- workdays summer holidays – Monday to Friday -> from 4:30 to 10:30

These values were established based on the graphs below. There are rendered mean values of Monday which has the longest peak hours in the graphs.



Graph 11: P+R Prague – peak hours

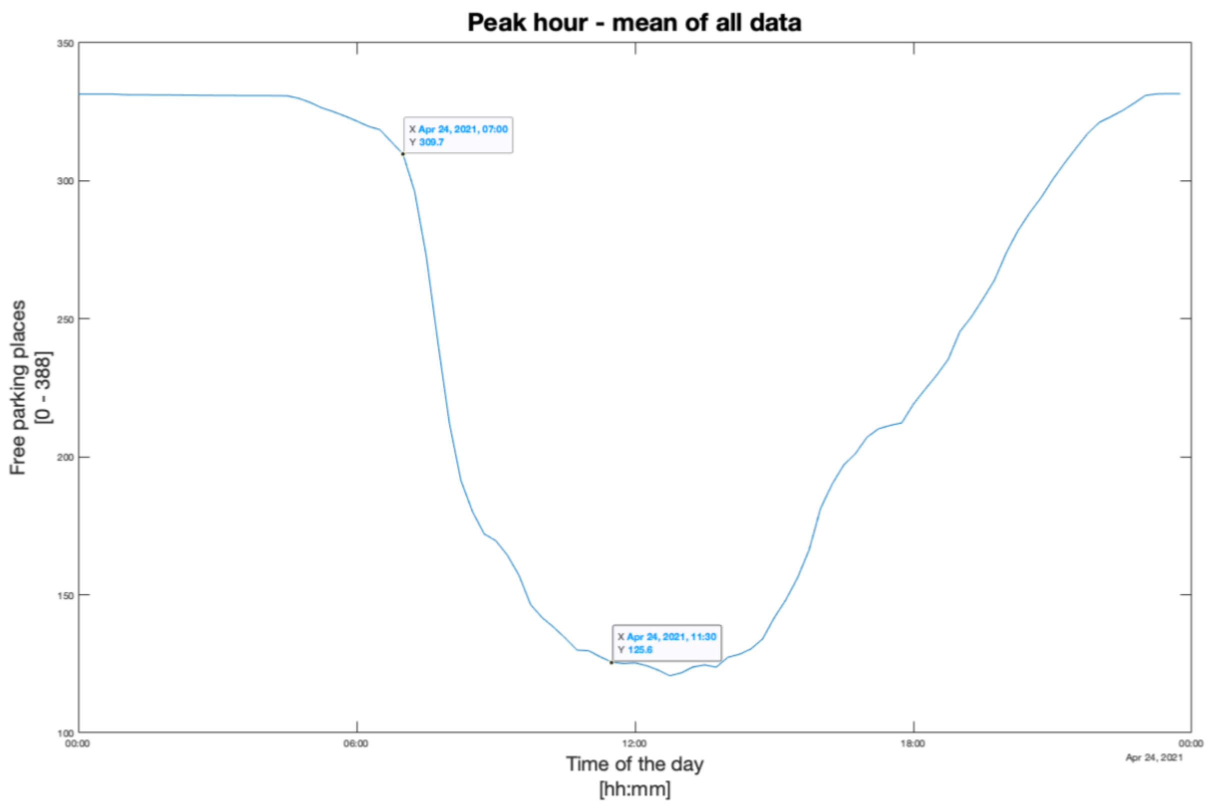


Graph 12: P+R Prague – peak hours summer holidays

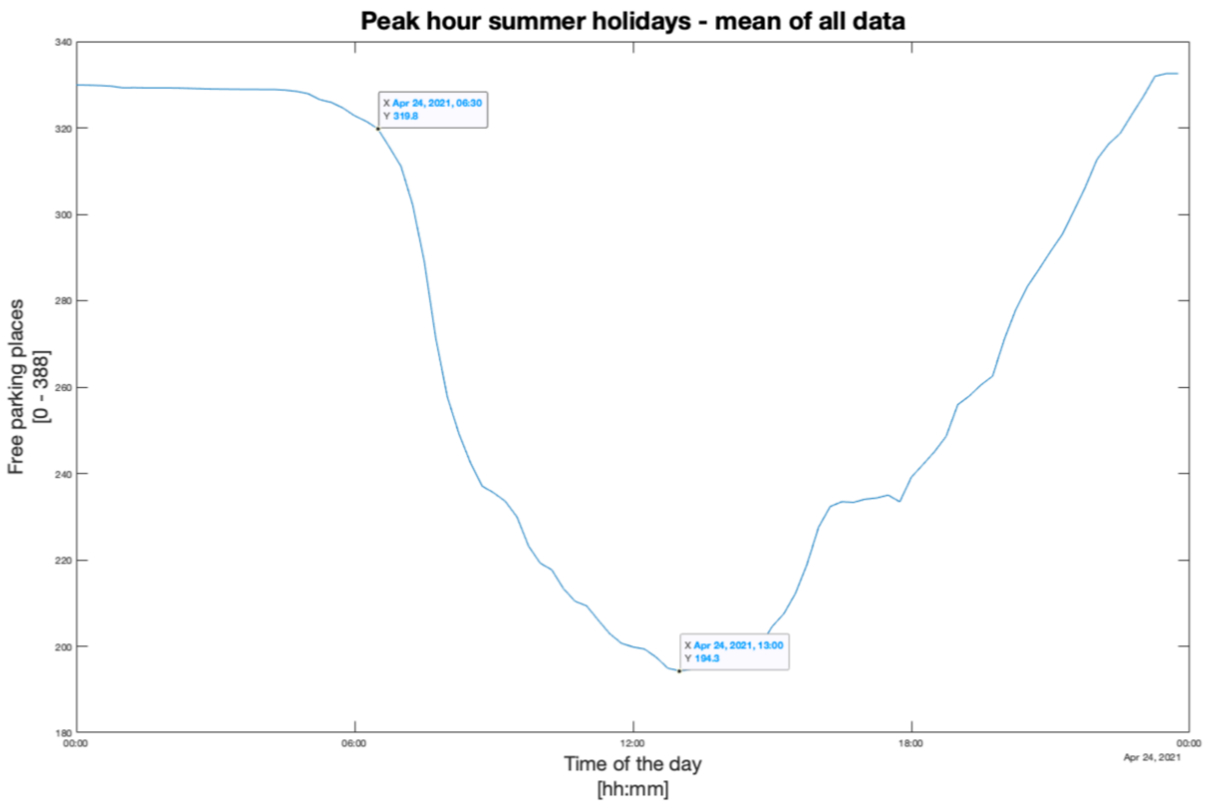
For the Parking Garages Norrköping it is:

- workdays – Monday to Friday -> from 7:00 to 11:30
- workdays summer holidays – Monday to Friday -> from 6:30 to 13:00

These values were established based on the graphs below. There are rendered mean values of Tuesday which has the longest peak hours in the graphs.



Graph 13: Parking garages Norrköping – peak hours



Graph 14: Parking garages Norrköping – peak hours summer holidays

6.1.3 Decision-making

The prediction system is Neural Network with 10 hidden neurons. This system is always trained based on the historical data of each parking lot and each day according to the user's choice. So, the CSV file with the historical data is imported to the NN and the NN is trained and tested to this data. Then the number of free parking places for the chosen time and one timestep forward and backward (for the purpose of the actual trend) is predicted.

Residents can park in the paid parking zones without a parking ticket and drivers may disrespect the parking rules, therefore, the TSK company was contacted. This company manages the paid parking zones. This company made available data from the last half a year (from August of 2020 to March 2021) about the residents and respect of the rules and other parameters in the selected zones. The obtained values of residents and don't respect (percentage to all the parking spaces) were added to the system as occupied parking spaces. Therefore, these values are subtracted from the predicted number of free parking places.

The main part of the decision-making is the evaluation of each state, that can happen. The states are defined in the activity diagram of the UC4 – Management and evaluation. There are the following states:

- is there parking space (50)
- is it in peak hour (20)
- trends (0-20)

The number in the brackets represents the evaluation of each state. This number is called P and is described below. The trends (one timestep forward and backward) are evaluated together and can and the range is 0 to 20 points. There is described the evaluation of the trends in the following part. The backward/forward represents the number of free places in the one timestep backward/forward and the number in the brackets represents the achieved points.

- increasing or stabilize trend (20)
  - backward is lower or equal to the predicted time that is lower or equal to forward
- beginning of increase or stable (15)
  - backward is 0 or higher to the predicted time that is lower or equal to forward
- beginning of decrease (10)
  - backward is lower or equal to the predicted time that is higher to forward
- decreasing (0)
  - backward is higher to the predicted time that is higher to forward

6.1.4 Recommendation to the user

The result of this system is a recommendation to the user whether it is good to go to the selected parking lot or not. For this purpose, the numbers in the parentheses were created. The number itself is called P. These numbers represent the evaluation of each state and are added up based on the criteria met. The P number range is 0 to 90 points and the limit for the successful probability, that driver can park in the chosen parking lot is P >= 75. There can happen 8 states based on the P value. These states have the following overall P value:

1. P=0 – There is no free parking space in the selected parking lot.
2. P=50 – There is a free parking space in the selected parking lot, but it is in the peak hour and the trend is decreasing.
3. P=60 – There is a free parking space in the selected parking lot, but it is in the peak hour and the trend begins decreasing.
4. P=65 – There is a free parking space in the selected parking lot, but it is in the peak hour, but the trend begins increasing.
5. P=70 – There is a free parking space in the selected parking lot, but it is in the peak hour, but the trend is increasing. Or there is a free parking space in the selected parking lot, but the trend is decreasing, but it is not in the peak hour.
6. P=80 – There is a free parking space in the selected parking lot, but the trend begins decreasing, but it is not in the peak hour.
7. P=85 – There is a free parking space in the selected parking lot, the trend begins increasing, and it is not in the peak hour.
8. P=90 – There is a free parking space in the selected parking lot, the trend is increasing, and it is not in the peak hour.

The Activity diagrams is in the UC4 – Management/Evaluation and in ANNEX A.

If the P value is equal to or over the limit (75) the recommendation for the driver is:

The probability, that you can park here in the next (based on the selected prediction time) minutes, is sufficient. There should be approximately (based on the predicted number of free places) free parking places.

But if the P value is lower than the limit (75) the recommendation is:

We are sorry but the probability, that you can park here in the next (based on the selected prediction time) minutes, is NOT sufficient. There should be approximately (based on the predicted number of free places) free parking places. We recommend choosing another parking lot or use a different mode of transport.

Or in the case of the paid parking zone, there should be recommendations for other paid parking zone in the surroundings.

When the recommendation is shown to the user, the process is finished, and the system ends.

6.2 Verification of the results

The system works properly and can recommend to the user whether the parking lot should have free parking space or not. But there is another question - whether the prediction is accurate or as close as possible to reality. Therefore, the system was tested in the real-life. The occupancy of P+R Prague [50] and Parking Garages Norrköping [40] can be checked by the internet website because the real occupancy is presented on the provider's websites. However, the ZPS Prague cannot be tested in the same way, therefore the check must be provided personally at the site.

The predicted and measured data are described in the table below. There is a time when the prediction was calculated. The prediction is for the next 15, respectively 30 minutes. There are written the predicted and real values for all the parking lots. The predicted values have a note about the prediction - whether the system recommends going there (marked as OK in the table) or not (marked as NOK in the table). There is also provided a calculation of the difference between predicted and real values in the table. The difference is calculated in the number of free places and percentage also.

The overall difference in percentage is approximately 20%. This deviation is high for this type of prediction system. On the other hand, there is the importance of the current pandemic situation. The testing was performed during an ongoing pandemic of COVID-19 disease all around the world. In connection with this situation, the overall parking demand is lower than the demand before the pandemic. And the prediction is based on the historical data before the pandemic. Therefore, the deviation is that higher.

Time of prediction	Predicted time	Prediction/ real data	P+R Černý Most 2 [131]	P+R Letňany [633]	ZPS-P7-0158 [22]	Spiran [388]	Vardtornet [84]
7:00	7:15	prediction	NOK 47	NOK 102	NOK 5	NOK 250	NOK 48
		real	70	403	1	334	66
	7:30	prediction	NOK 33	NOK 78	NOK 5	NOK 200	NOK 48
		real	67	379	1	319	68
9:00	9:15	prediction	OK 5	NOK 52	OK 5	NOK 133	OK 43
		real	41	259	0	244	59
	9:30	prediction	OK 3	OK 50	OK 5	OK 133	NOK 40
		real	40	253	0	244	57
12:00	12:15	prediction	OK 10	OK 48	OK 5	NOK 107	OK 40
		real	30	237	1	208	53
	12:30	prediction	OK 10	OK 50	OK 5	OK 117	OK 40
		real	31	239	0	200	50
16:00	16:15	prediction	OK 70	OK 326	OK 5	NOK 192	NOK 50
		real	73	338	2	241	67
	16:30	prediction	OK 79	OK 361	OK 5	OK 191	NOK 44
		real	77	349	1	236	67
17:00	17:15	prediction	OK 98	OK 438	OK 5	OK 213	NOK 36
		real	95	405	0	258	64
	17:30	prediction	OK 102	OK 460	OK 5	OK 224	OK 36
		real	95	416	2	257	63
+ 15 minutes	Difference [free places]		17	148	4	78	18
	Difference [%]		12,98	23,44	19,09	20,10	21,90
+ 30 minutes	Difference [free places]		20	150	4	78	19
	Difference [%]		15,42	23,67	19,09	20,15	23,10
Overall difference [%]			19,89				

Table 6: Verification of the prediction

There was observed that the recommendation from the decision-making was negative (not to go there) but there were predicted enough free parking places. Therefore, the decision-making was updated. The updated parts of the codes are in ANNEX H. Into the UC4 was added a loop that checks if the predicted number of free places and the number in the next time step is higher than the limit. This limit was determined as the 35 approaching cars per 15 minutes, 70 approaching cars per 30 minutes. If this limit is fulfilled and the P < 75 then the prediction is not sufficient but there should be enough places. Therefore, the answer for the user is updated in this way. This update is for the P+R Prague and Parking garages Norrköping only. The ZPS Prague works in a different way. The manual research gives information about the parking of the delivery services in the monitored zone. Therefore, for the ZPS Prague were modified the answers. For the positive recommendation (go there) there is also information, that the maximum number of free places should be in another zone and the number of free places is also stated there. The same is for the negative recommendation (not to go there).

The added recommendation is as following:

We are sorry but the probability, that you can park here in the next (based on the selected prediction time) minutes, is NOT sufficient. However, there should be enough free places to go there. There should be approximately (based on the predicted number of free places) free parking places. We recommend choosing another parking lot or use a different mode of transport.

7 Discussion

The aim of this thesis is to develop a system for a short-time occupancy prediction with decision-making that provides a recommendation to the user whether is good to go to the selected parking lot or not. This prediction is focused on public parking spaces because there is an assumption that these parking spaces should be part of the open data. The data should be based on occupancy in time or the number of free places in time. In a general manner, it is important to record changes in the parking occupancy in time. It is necessary to have additional data like capacity and location. The capacity is used for the calculation of free places and it is used at the end for the verification of the prediction in real situations. The location is used for computing if there is another parking lot in the surroundings.

The prediction method itself is inspired by the acquired knowledge from the previous theoretical part that is based on similar already performed projects. Based on the theoretical part several prediction methods were chosen. These methods were compared on the same datasets. The most accurate one was chosen for the system. It is Neural Network with 10 hidden neurons. To the prediction method was added decision-making that provides recommendation whether it is good to go to the selected parking lot or not. This decision-making provides an evaluation of the predicted value, capacity of the parking lot, and trend of the occupancy in the prediction time.

The final system was tested/verified in real situations on all parking lots. Most of the parking lots have data online, however, the ZPS Prague does not. So, this measurement was provided manually and data from the rest of the parking lots were read online. The verification was done by calculating the deviation between the predicted and real value. The deviation is approximately 20% across all the parking lots and daytime. However, the testing measurement was provided during the pandemic of COVID-19 and ongoing mobility restrictions, so the demand for parking is lower than in the no-pandemic situation. Because the historical data for training the prediction method is before pandemic only, the deviation is higher. There was also observed that the decision-making system recommends not to go to the parking lot although there were enough free parking places. Therefore, based on the verification, the decision-making was updated.

This thesis also observed different approaches to the data between chosen cities. Prague has the historical data openly available and the data of P+R is in the 5-minutes interval (288 records per day). Norrköping has openly real-time data that can be read on the Norrköping's website (NorrköpingsKartan) but the historical data was provided by the shadow supervisor. And these data are in the 15-minutes interval (96 records per day). The difference in the number of records per day has an influence on the prediction method, on the level of training respectively, but in the verification, this difference was not observed. And the structure of the data is very similar and therefore the system can be used anywhere. But the data must be analyzed for the trends in the driver's behavior – when they arrive and depart the parking lot. And the data must be transformed to the corresponding data structure for the prediction.

There was observed that in Norrköping the parking garages are not full (after peak hours) or empty (night). From this information can be assumed that the garages have enough capacity or people don't use a vehicle that much (for the case of full occupancy) and at night there is an assumption that there are cars of residents. The Prague case is that during the night the P+R is almost empty and during the day (mainly after peak hours) it is full. The case of the paid parking zone is different based on each zone.

For more accurate results should be more detailed data analysis that will study each day separately and predict each day (not gathered workdays). Also, information about weather, traffic situation, and others can be considered for the prediction. The improvement can be also in the decision-making that should compare the prediction in the 15 and 30 minutes and can recommend to the driver to wait another 15 minutes because in 30 minutes the chance is higher. Or the trend should be established by more timesteps than one timestep backward and forward. And like the system development should be an extension of real-time data gathering and comparison. Also, this data can be saved in the database for more accurate future results. This can help with more accurate results during the pandemic (the demand is not as high as before) or other unexpected situations.

Two of the limitations are with significant influence. On the ZPS Prague, there are a few percentages of drivers that don't respect the rule which is buying the ticket. Therefore, the accuracy of the prediction could be not sufficient. To eliminate this limitation is necessary to check each paid parking zone and perform a measurement there. The second limitation is the pandemic of the COVID-19 disease. Because of the restrictions in mobility, there is a significant decrease in the demand for parking spaces. Also more used is a home office. Therefore, mobility is decreasing.

8 Conclusion

The aim of this thesis was to develop an occupancy prediction system, that provides a recommendation to the user whether go to the parking lot or not. This system was performed in Matlab. This system can do a short-time prediction of the occupancy of public parking spaces in Prague (the Czech Republic) and Norrköping (Sweden). There is a different position of these cities. Prague is the capital city with more than 1 million inhabitants and has different problems with traffic than Norrköping which has approximately 100 thousand inhabitants. Also, there is a difference in the behavior of the inhabitants - in Sweden, the personal vehicle is not used that much as in the Czech Republic in terms of mobility in the city. But both cities monitor the public parking spaces and based on the historical data of the last three years (2018 - 2020) the prediction is made. The historical data after the 29th of February 2020 were excluded because of the pandemic of COVID-19 disease. The pandemic situation and the mobility restrictions caused a significant decrease in the demand for parking places. Because the demand was around 100% of capacity in Prague and during the pandemic, the demand was around 10% (in the worst times) therefore, these data were excluded. This significant unique decrease in demand can negatively affect the prediction.

To learning the trends in the occupancy was performed data analysis for all the selected parking lots. Based on the trends similar days were gathered (such as Tuesday, Wednesday, and Thursday). There was also considered the summer and public holidays. The input data was analyzed and then transformed to the structure for the prediction method.

The prediction method is a Neural network with 10 hidden neurons. This neural network can predict the number of free places with high accuracy and low computing time. The predicted number of free places is the input to the decision-making that provides recommendations to the user whether go to the parking lot or not. The recommendation is based on the number of predicted free parking places and the trends in the predicted time.

The designed system was tested in real life. There was performed a measurement in all the datasets for the purpose of checking the deviation between predicted and real value. The overall deviation is 20% across all the datasets and the daytime. The deviation is high because of the lower demand at the time of the pandemic. The deviation can be more accurate after the pandemic.

One of the important thoughts of this thesis was whether such a system can be made generally for any parking lot or city. This system needs a data analysis of each parking lot, investigate trends, transform the data, and therefore it is challenging but yes - such a system can be made across the world. The prediction and decision-making itself are flexible to other cities/parking lots but it requires correspond data structure.

The future development of the system can be in the extension of real-time data that can correct the decision-making part of the system. The real-time data can be gathered in the database and saved for future predictions. This solution can be helpful in the case of unexpected situations such as the pandemic situation (lower demand for parking) or any sport or culture event (high demand for parking).

The predicted information can be presented to the drivers by VMS or navigation system, mobile apps, or city dashboards. If the driver will have the information about the future parking situation, he/she can modify his/her route to the destination and make the roads and parking lots more effective. And in the case of on-street parking, it can reduce traffic jams that are due to looking for a free parking space. But it is in connection to the respect of the parking rules.

Bibliography

[1] Ministry of Transportation, Department of Roads, 2013. *TP65 – Principles for traffic signs on roads*. Available on the website [08.01.2021]: http://www.pjpk.cz/data/USR_001_2_8_TP/TP_65.pdf

[2] Biswas, Subhadip & Chandra, Satish & Ghosh, Indrajit. (2017). *Effects of On-Street Parking In Urban Context: A Critical Review*. Transportation in Developing Economies. 3. 1-14. 10.1007/s40890-017-0040-2.

[3] ČESKO. *§ 27 odst. 1 písm. o) zákona č. 361/2000 Sb., o provozu na pozemních komunikacích a o změnách některých zákonů*. In: <i>Zákony pro lidi.cz</i> [online]. © AION CS 2010-2021 Available on the website [08.01.2021]: https://www.zakonyprolidi.cz/cs/2000-361#p27-1-o

[4] TSK hl. m. Prahy, a.s. – Úsek dopravního inženýrství 2020. *Ročenka dopravy Praha 2019 (Prague transportation year book 2019)*. SOFIPRIN Praha, Praha 2020. Available on the website [08.01.2021]: CZ version: http://www.tsk-praha.cz/static/udi-rocenka-2019-cz.pdf; EN version: http://www.tsk-praha.cz/static/udi-rocenka-2019-en.pdf

[5] Seznam.cz, a.s. (2020). *Mapy.cz*, Available on the website [08.01.2021]: https://en.mapy.cz/zakladni?x=14.4469049&y=50.1025948&z=18&pano=1&source=traf&id=270&pid=69676183&yaw=0.536&fov=1.257&pitch=0.245

[6] Seznam.cz, a.s. (2020). *Mapy.cz*, Available on the website [08.01.2021]: https://en.mapy.cz/zakladni?x=14.5145352&y=50.1256639&z=19&l=0&base=ophoto&source=traf&id=270

[7] Mejstřík J. (IPR Praha) (2018). Dojížďka a vyjížďka do zaměstnání do/z hl. m. Prahy [aktualizace 2018] (Commuting to and from work to / from hl. City of Prague [update 2018]). Available on the website [08.01.2021]: https://www.iprpraha.cz/uploads/assets/dokumenty/ssp/analyzy/Obyvatelstvo/dojizdka_2018_mejstrik.pdf

[8] Polaď Prahu (2019). *The sustainable mobility plan for Prague and its suburbs.* Luova Publishing, s.r.o., Available on the website [08.01.2021]: https://poladprahu.cz/wp-content/uploads/2019/11/Brožura_Plán_mobility_CZ.pdf

[9] Biswas, Subhadip & Chandra, Satish & Ghosh, Indrajit. (2017). *Effects of On-Street Parking In Urban Context: A Critical Review.* Transportation in Developing Economies. 3. 1-14. 10.1007/s40890-017-0040-2.

[10] Stephen Ison, Tom Rye (2006), *Parking,Transport Policy*, Volume 13, Issue 6, Pages 445-446, ISSN 0967-070X, https://doi.org/10.1016/j.tranpol.2006.05.001.

[11] Peprah, C., Oduro, C.Y., & Ocloo, K. (2014). *On-Street Parking and Pedestrian Safety in the Kumasi Metropolis: Issues of Culture and Attitude*. Developing Country Studies, 4, 85-94.

[12] Eric Dumbaugh & J. L. Gattis (2005). *Safe Streets, Livable Streets*, Journal of the American Planning Association, 71:3, 283-300, DOI: 10.1080/01944360508976699

[13] Gattis JL (2000). *Urban Street Cross Section and Speed Issues*. TRB Circ E-C019 Urban Str Symp 1–17

[14] Gårder P, Ivan JN, Du J (2002). *Traffic Calming of State High- ways: Application New England*. Technical Report, Project No. UCNR13-5, New England University Transportation Center, Massachusetts Institute of Technology

[15] Marshall W, Garrick N, Hansen G (2008). *Reassessing on-street parking*. Transp Res Rec J Transp Res Board 2046:45–52. doi:10.3141/2046-06

[16] Box, P. (2004). *Curb-Parking Problems: Overview*. Journal of Transportation Engineering-asce, 130, 1-5.

[17] Elliot MA, McColl VA, Kennedy JV (2003) *Road design meas- ures to reduce drivers' speed via "psychological" processes: a lit- erature review*. Report No. TRL564, Transport Research Labora- tory, Crowthorne, Berkshire, UK

[18] Herin K.J., Jisha Akkara (2019). *Study of "On-Street" and "Off-Street" Parking Choice Behaviour.* International Journal of Advanced Research in Computer and Communication Engineering, Vol. 8, Special Issue 1, ISSN (Online) 2278-1021

[19] Alajali, Walaa & Wen, Sheng & Zhou, Wanlei. (2017). *On-Street Car Parking Prediction in Smart City: A Multi-source Data Analysis in Sensor-Cloud Environment*. 10.1007/978-3-319-72395-2_58.

[20] Ji, Z., Ganchev, I., O'Droma, M., & Zhang, X. (2014). *A cloud-based intelligent car parking services for smart cities.* 2014 XXXIth URSI General Assembly and Scientific Symposium (URSI GASS), 1-4.

[21] Rajabioun, T., & Ioannou, P.A. (2015). *On-Street and Off-Street Parking Availability Prediction Using Multivariate Spatiotemporal Models.* IEEE Transactions on Intelligent Transportation Systems, 16, 2913-2924.

[22] Shuguan Yang, Wei Ma, Xidong Pi, Sean Qian (2019). *A deep learning approach to real-time parking occupancy prediction in transportation networks incorporating multiple spatio-temporal data sources*, Transportation Research Part C: Emerging Technologies, Volume 107, Pages 248-265, ISSN 0968-090X, https://doi.org/10.1016/j.trc.2019.08.010.

[23] SFMTA (2014). *Parking Sensor Technology Performance Evaluation*. Technical Report.

[24] Eleni I. Vlahogianni, Konstantinos Kepaptsoglou, Vassileios Tsetsos & Matthew G. Karlaftis (2016). *A Real-Time Parking Prediction System for Smart Cities,* Journal of Intelligent Transportation Systems, 20:2, 192-204, DOI: 10.1080/15472450.2015.1037955

[25] Teodorovic, D., & Lucic, P. (2006). *Intelligent parking systems*. European Journal of Operational Research, 175(3), 1666–1681. http://dx.doi.org/10.1016/j.ejor.2005.02.033

[26] Komninos, N. (2009). *Intelligent cities: Towards interactive and global innovation environments*. International Journal of Innovation and Regional Development, 1(4), 337–355. http://dx.doi.org/10.1504/IJIRD.2009.022726

[27] Atzori, L., Iera, A., & Morabito, F. (2010). *The Internet of Things: A survey. Computer Networks*, 54(15), 2787–2805. http://dx.doi.org/10.1016/j.comnet.2010.05.010

[28] Karlaftis, M. G., & Vlahogianni, E. I. (2011). *Statistics versus neural networks in transportation research: Differences, similarities and some insights.* Transportation Research Part C: Emerging Technolo- gies, 19(3), 387–399. http://dx.doi.org/10.1016/j.trc.2010.10.004

[29] Vlahogianni, E. I., Karlaftis, M. G., & Golias, J. C. (2005). *Optimized and meta-optimized neural networks for short- term traffic flow prediction: A genetic approach. Transportation*, Research Part C: Emerging Technologies, 13(3), 211–234. http://dx.doi.org/10.1016/j.trc.2005.04.007

[30] Stéphane Cédric Koumetio Tekouabou, El Arbi Abdellaoui Alaoui, Walid Cherif, Hassan Silkan (2020). *Improving parking availability prediction in smart cities with IoT and ensemble-based model*, Journal of King Saud University - Computer and Information Sciences, ISSN 1319-1578, https://doi.org/10.1016/j.jksuci.2020.01.008.

[31] Liu, Kin Sum & Gao, Jie & Wu, Xiaobing. (2018). *On-Street Parking Guidance with Real-Time Sensing Data for Smart Cities*. 1-9. 10.1109/SAHCN.2018.8397113.

[32] Monteiro, Fernando & Ioannou, Petros. (2018). *On-Street Parking Prediction Using Real-Time Data*. 2478-2483. 10.1109/ITSC.2018.8569921.

[33] J. Belissent (2013). *Getting clever about smart cities: New opportunities re-quire new business models*, in http://www.forrester.com/rb/Research/ getting clever about smart cities new opportunities/q/id/56701/t/2

[34] Yanxu Zheng, S. Rajasegarar and C. Leckie (2015). *Parking availability prediction for sensor-enabled car parks in smart cities*, IEEE Tenth International Conference on Intelligent Sensors, Sensor Networks and Information Processing (ISSNIP), Singapore, 2015, pp. 1-6, doi: 10.1109/ISSNIP.2015.7106902.

[35] Prague Opendata portal [online], provided by *ckan*. Available: 25th of April 2021, https://opendata.praha.eu/dataset/zps_tickets

[36] Prague Opendata portal [online], provided by *ckan*. Available: 25th of April 2021, https://opendata.praha.eu/dataset/parkovani-pr-api

[37] Golemio API [online], provided by Apiary, Available: 25th of April 2021, https://golemioapi.docs.apiary.io/#reference/parking/parking-lots

[38] Municipal facts 2019, Let's create Norrköping, Available: 25th of April 2021, https://www.norrkoping.se/download/18.2b7f0b9316b3fde7751b7f/1561450381664/NKPG_i_siffror_2019_engelska.pdf

[39] Norrköping website, Perkering [online], Available: 25th of April 2021, https://www.norrkoping.se/boende-trafik-och-miljo/trafik-och-parkering/parkering

[40] Norrköping website, Norrköpingskartan [online], Available: 25th of April 2021, https://kartor.norrkoping.se/spatialmap?selectorgroups=resor_och_trafik&mapext=129659%206496117.3%20133755%206498059.7&layers=kulisskartan_ej_text%20tk_parkering_pendlar&mapheight=612&mapwidth=1285&profile=kartor

[41] Infracontol, Special report, February 2019, Norrköping is getting smarter with IoT [online], Available: 25th of April 2021, https://www.infracontrol.com/InfracontrolSpecialReports/en2019/259.pdf

[42] REGULATION (EU) 2016/679 OF THE EUROPEAN PARLIAMENT AND OF THE COUNCIL of 27 April 2016 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing Directive 95/46/EC (General Data Protection Regulation), https://eur-lex.europa.eu/legal-content/EN/TXT/PDF/?uri=CELEX:32016R0679&from=EN

[43] Prague Opendata portal [online], provided by *ckan*. Available: 25th of April 2021, https://opendata.praha.eu/dataset/ipr-zachytna_parkoviste_p_r

[44] Prague Opendata portal [online], provided by *ckan*. Available: 25th of April 2021, https://opendata.praha.eu/dataset/ipr-useky_parkovani_v_zonach_placeneho_stani

[45] Zhu, Xiaobo & Guo, Jianhua & Huang, Wei & Yu, Fengquan & Park, B.. (2018). *Real Time Short-term Forecasting Method of Remaining Parking Space in Urban Parking Guidance Systems*. PROMET - Traffic&Transportation. 30. 173. 10.7307/ptt.v30i2.2388.

[46] Camero, Andrés & Toutouh, Jamal & Stolfi, Daniel H. & Alba, Enrique. (2019). *Evolutionary Deep Learning for Car Park Occupancy Prediction in Smart Cities: 12th International Conference, LION 12, Kalamata, Greece, June 10–15, 2018, Revised Selected Papers*. 10.1007/978-3-030-05348-2_32.

[47] Ji, Y., Tang, D., Blythe, P., Guo, W. and Wang, W. (2015) *Short-term forecasting of available parking space using wavelet neural network model*. IET Intell. Transp. Syst., 9: 202-209. https://doi.org/10.1049/iet-its.2013.0184

[48] Gareth James, Daniela Witten, Trevor Hastie, Robert Tibshirani. (2013). *An introduction to statistical learning: with applications in R*. New York: Springer.

[49] Zákony pro lidi [online]. © AION CS, s.r.o. 2010-2021 Available: 25th of April 2021, https://www.zakonyprolidi.cz/cs/2000-361

[50] P+R Parkoviště, Copyright 2021, Technická správa komunikací hlavního města Prahy, a.s., Available: 25th of April 2021, https://www.tsk-praha.cz/wps/portal/root/aktualni-doprava/parkoviste/x

List of figures

List of tables

List of graphs

List of diagrams

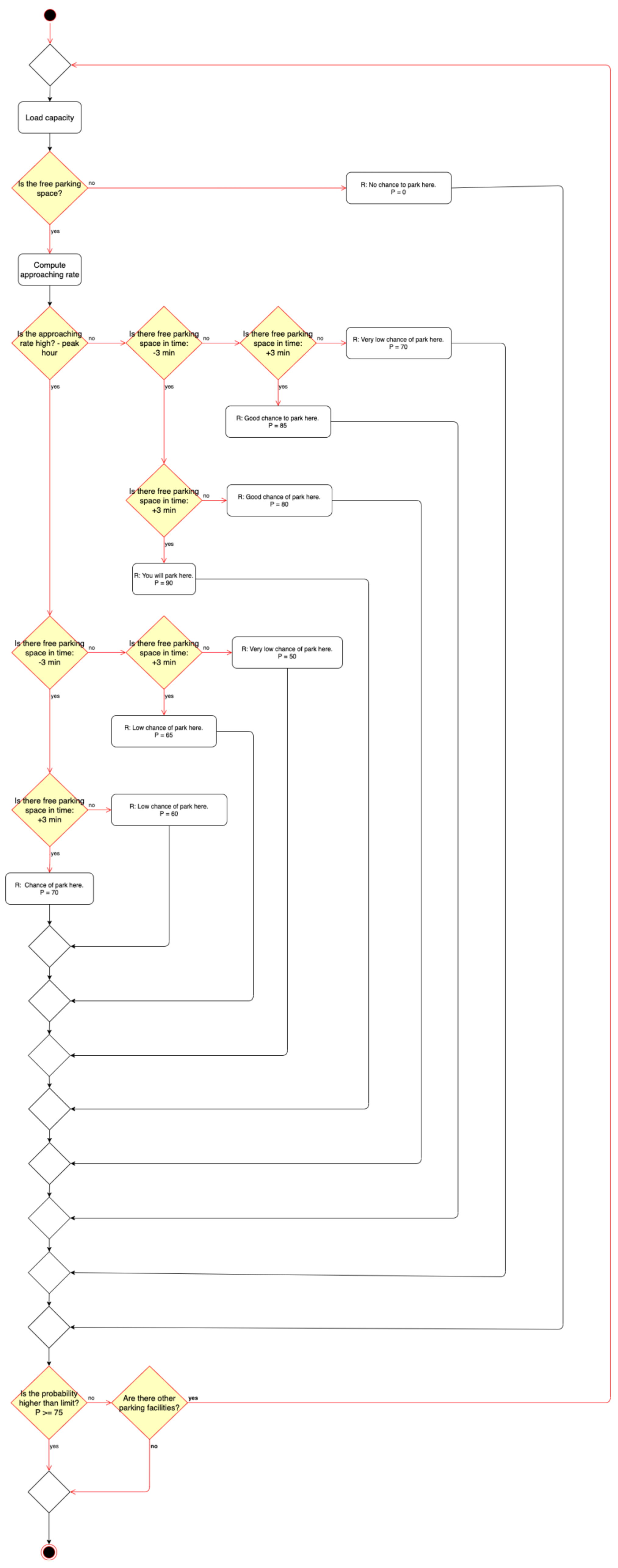
List of equations

List of attachments

Annexes mentioned below are presented on the following pages.

Annex A

The activity diagram of the UC4 – Management and evaluation for Activity diagrams chapter.


Load capacity
Is the free parking space?
no
R: No chance to park here. P = 0
yes
Compute approaching rate
Is the approaching rate high? - peak hour
no
Is there free parking space in time: -3 min
no
Is there free parking space in time: +3 min
no
R: Very low chance of park here. P = 70
yes
R: Good chance to park here. P = 85
yes
Is there free parking space in time: +3 min
no
R: Good chance of park here. P = 80
yes
R: You will park here. P = 90
yes
Is there free parking space in time: -3 min
no
Is there free parking space in time: +3 min
no
R: Very low chance of park here. P = 50
yes
R: Low chance of park here. P = 65
yes
Is there free parking space in time: +3 min
no
R: Low chance of park here. P = 60
yes
R: Chance of park here. P = 70
Is the probability higher than limit? P >= 75
no
Are there other parking facilities?
yes
no
yes

Annex B

```
import requests

import pandas as pd

import json


yy=20 # represents years - has to be changed

mm="12" # represents months - has to be changed

dd=31 # represents days - has to be changed


fromm="20{}-{}-01".format(yy,mm) # time from for the URL

to="20{}-{}-{}".format(yy,mm,dd) # time to for the URL


# First P+R


sensorid=534016 # P+R Letňany


# URL of the API

url = "https://api.golemio.cz/v2/parkings/history?sensorId={}&from={}T00:00:00.000Z&to={}T23:59:59.000Z".format(sensorid, fromm, to)


# parts for the URL API read (payload and API-token)

payload={}

headers = {

  'x-access-token': 'eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJlbWFpbCI6ImJlbmVzdmkxQGZkLmN2dXQuY3oiLCJpZCI6Mzk0LCJuYW1lIjpudWxsLCJzdXJuYW1lIjpudWxsLCJpYXQiOjE2MDA5NTA1NjYsImV4cCI6MTE2MDA5NTA1NjYsImlzcyI6ImdvbGVtaW8iLCJqdGkiOiI2NWFjYjRhMi00ZjY0LTRkMDItYTJlOC02MGM2NGFmYjczZDcifQ.jvfsGgghJMRPGWPMcwBq4PcHqIVffmT_IqIxDZhZFK8'

}


# variable data which includes the answer from the server
```

```
data = requests.request("GET", url, headers=headers, data=payload)


# part for saving data as a json file

f = open("{}_{}.json".format(sensorid,fromm), "w") # create json file

f.write(data.text) # write data to json file

f.close() # close file


# Second P+R


sensorid=534011 # P+R Černý most 2


# URL of the API

url                                                                                                                                    =
"https://api.golemio.cz/v2/parkings/history?sensorId={}&from={}T00:00:00.000Z&to={}T23:59:
59.000Z".format(sensorid, fromm, to)


# parts for the URL API read (payload and API-token)

payload={}

headers = {

  'x-access-token':
'eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJlbWFpbCI6ImJlbmVzdmkxQGZkLmN2dXQuY
3oiLCJpZCI6Mzk0LCJuYW1lIjpudWxsLCJzdXJuYW1lIjpudWxsLCJpYXQiOjE2MDA5NTA1N
jYsImV4cCI6MTE2MDA5NTA1NjYsImlzcyI6ImdvbGVtaW8iLCJqdGkiOiI2NWFjYjRhMi00ZjY
0LTRkMDItYTJlOC02MGM2NGFmYjczZDcifQ.jvfsGgghJMRPGWPMcwBq4PcHqIVffmT_IqI
xDZhZFK8'

}


# variable data which includes the answer from the server

data = requests.request("GET", url, headers=headers, data=payload)


# part for saving data as a json file

f = open("{}_{}.json".format(sensorid,fromm), "w") # create json file

f.write(data.text) # write data to json file

f.close() # close file
```

Annex C

P+R Prague

The main code

```
%% Initialization

clear
close all

%% P+R ÄŒernÃ½ Most 2 - data input (json file)
% load json data
directoryCM = '/Users/viktorbenes/diplomka/Prague P+R/cerny most 2/';
PR_CM2 = PR_CernyMost2(directoryCM);

%% P+R LetÅˆany - data input (json file)
% load json data
directoryL = '/Users/viktorbenes/diplomka/Prague P+R/letnany/';
PR_L = PR_Letnany(directoryL);

%% Figure 1 - Original data - year comparison
% plot raw data
PR_figure_yearcomparison(PR_L,PR_CM2)

%% Cleaning - P+R ÄŒernÃ½ most 2
% data cleaning
PR_CM2_cleaned = PR_Cleaning(PR_CM2);

%% Cleaning - P+R LetÅˆany
% data cleaning
PR_L_cleaned = PR_Cleaning(PR_L);

%% Figure 2 - Cleaned data - year comparison
% plot cleaned data
PR_figure_yearcomparison2(PR_L, PR_L_cleaned, PR_CM2, PR_CM2_cleaned);

%% Set public holidays

% dates of Czech Public Holidays
PublicHolidays=['01-01-2018'; '30-03-2018'; '02-04-2018'; '01-05-2018';
    '08-05-2018'; '05-07-2018'; '06-07-2018'; '28-09-2018'; '28-10-2018';
    '17-11-2018'; '24-12-2018'; '25-12-2018'; '26-12-2018';
    '01-01-2019'; '19-04-2019'; '22-04-2019'; '01-05-2019'; '08-05-2019';
    '05-07-2019'; '06-07-2019'; '28-09-2019'; '28-10-2019'; '17-11-2019';
    '24-12-2019'; '25-12-2019'; '26-12-2019';
    '01-01-2020'; '10-04-2020'; '13-04-2020'; '01-05-2020'; '08-05-2020';
    '05-07-2020'; '06-07-2020'; '28-09-2020'; '28-10-2020'; '17-11-2020';
    '24-12-2020'; '25-12-2020'; '26-12-2020'];
PublicHolidays=datetime(PublicHolidays,'InputFormat','dd-MM-yyyy');


%% Day comparison - P+R ÄŒernÃ½ most 2

directoryCM_cleaned = '/Users/viktorbenes/diplomka/Prague
P+R/days_PR_CM2/';
```

```
PR_days(PR_CM2_cleaned,PublicHolidays,directoryCM_cleaned);

%% Figure: Day comparison - P+R ÄŒernÃ½ most 2

PR_CM2_days_figure(directoryCM_cleaned);

%% Day comparison - P+R LetÅˆany

directoryL_cleaned = '/Users/viktorbenes/diplomka/Prague P+R/days_PR_L/';
PR_days(PR_L_cleaned,PublicHolidays,directoryL_cleaned);

%% Figure: Day comparison - P+R LetÅˆany

PR_L_days_figure(directoryL_cleaned);
```

Function PR_CernyMost2

```
function CernyMost = PR_CernyMost2(directory)

%% Read json files

alljson=string('*.json');

fileName = dir(directory+alljson);
fileName_cell=struct2cell(fileName);

CernyMost=table();
for k = 1:length(fileName_cell)
    filetoread=string(fileName_cell(1,k));
    openfile=directory+filetoread;
    file=fileread(openfile);
    CernyMost=[CernyMost;(struct2cell(jsondecode(file))')];
end


%% Set table
CernyMost.b6=datetime(CernyMost.b6,'InputFormat', 'yyyy-MM-
dd''T''HH:mm:ss.SSS''Z');
CernyMost = sortrows(CernyMost,'b6','ascend');

CernyMost.Properties.VariableNames = {'id' 'last_updated'
'num_of_free_places' 'num_of_taken_places' 'total_num_of_places'
'updated_at'};

%% Occupancy computing

for k = 1:height(CernyMost)

Oc(k)=CernyMost.num_of_taken_places(k)/CernyMost.total_num_of_places(k)*100
;
    Oc(k)=round(Oc(k));
end
Oc=Oc';
Oc=array2table(Oc);
Oc.Properties.VariableNames = {'Occupancy'};
CernyMost=[CernyMost Oc];

end
```

Function PR_Letnany

```
function Letnany = PR_Letnany(directory)

%% Read json files

alljson=string('*.json');

fileName = dir(directory+alljson);
fileName_cell=struct2cell(fileName);

Letnany=table();
for k = 1:length(fileName_cell)
    filetoread=string(fileName_cell(1,k));
    openfile=directory+filetoread;
    file=fileread(openfile);
    Letnany=[Letnany;(struct2cell(jsondecode(file))')];
end


%% Set table
Letnany.b6=datetime(Letnany.b6,'InputFormat', 'yyyy-MM-
dd''T''HH:mm:ss.SSS''Z');
Letnany = sortrows(Letnany,'b6','ascend');

Letnany.Properties.VariableNames = {'id' 'last_updated'
'num_of_free_places' 'num_of_taken_places' 'total_num_of_places'
'updated_at'};

%% Occupancy computing

for k = 1:height(Letnany)

Oc(k)=Letnany.num_of_taken_places(k)/Letnany.total_num_of_places(k)*100;
    Oc(k)=round(Oc(k));
end
Oc=Oc';
Oc=array2table(Oc);
Oc.Properties.VariableNames = {'Occupancy'};
Letnany=[Letnany Oc];

end
```

Function PR_figure_yearcomparison

```
function PR_figure_yearcomparison(PR_L,PR_CM2)

%% Figure prepairing

% Letnany - 2018
PR_L_2018 = PR_L.updated_at(PR_L.updated_at.Year==2018);
PR_L_2018 = datetime(PR_L_2018,'InputFormat','dd-MMM-yyyy HH:mm:ss',
'Format','dd-MMM HH:mm:ss');
PR_L_2018_Oc = PR_L.Occupancy(PR_L.updated_at.Year==2018);
```

```
% Letnany - 2019
PR_L_2019 = PR_L.updated_at(PR_L.updated_at.Year==2019);
PR_L_2019 = PR_L_2019 - calyears(1) + days(1);
PR_L_2019 = datetime(PR_L_2019,'InputFormat','dd-MMM-yyyy HH:mm:ss',
'Format','dd-MMM HH:mm:ss');
PR_L_2019_Oc = PR_L.Occupancy(PR_L.updated_at.Year==2019);

% Letnany - 2020
PR_L_2020 = PR_L.updated_at(PR_L.updated_at.Year==2020);
PR_L_2020 = PR_L_2020 - calyears(2) + days(3);
PR_L_2020 = datetime(PR_L_2020,'InputFormat','dd-MMM-yyyy HH:mm:ss',
'Format','dd-MMM HH:mm:ss');
PR_L_2020_Oc = PR_L.Occupancy(PR_L.updated_at.Year==2020);

% Cerny Most 2 - 2018
PR_CM2_2018 = PR_CM2.updated_at(PR_CM2.updated_at.Year==2018);
PR_CM2_2018 = datetime(PR_CM2_2018,'InputFormat','dd-MMM-yyyy HH:mm:ss',
'Format','dd-MMM HH:mm:ss');
PR_CM2_2018_Oc = PR_CM2.Occupancy(PR_CM2.updated_at.Year==2018);

% Cerny Most 2 - 2019
PR_CM2_2019 = PR_CM2.updated_at(PR_CM2.updated_at.Year==2019);
PR_CM2_2019 = PR_CM2_2019 - calyears(1) + days(1);
PR_CM2_2019 = datetime(PR_CM2_2019,'InputFormat','dd-MMM-yyyy HH:mm:ss',
'Format','dd-MMM HH:mm:ss');
PR_CM2_2019_Oc = PR_CM2.Occupancy(PR_CM2.updated_at.Year==2019);

% Cerny Most 2 - 2020
PR_CM2_2020 = PR_CM2.updated_at(PR_CM2.updated_at.Year==2020);
PR_CM2_2020 = PR_CM2_2020 - calyears(2) + days(3);
PR_CM2_2020 = datetime(PR_CM2_2020,'InputFormat','dd-MMM-yyyy HH:mm:ss',
'Format','dd-MMM HH:mm:ss');
PR_CM2_2020_Oc = PR_CM2.Occupancy(PR_CM2.updated_at.Year==2020);


%% Figure

figure(1)
subplot(2,1,1)
plot(PR_L_2018,PR_L_2018_Oc,PR_L_2019,PR_L_2019_Oc,PR_L_2020,PR_L_2020_Oc)
legend('2018','2019','2020','FontSize',15)
title('P+R Letnany','FontSize',20)
xlabel('time [Month, Day, Time]','FontSize',15)
ylabel('Occupancy [%]','FontSize',15)

subplot(2,1,2)
plot(PR_CM2_2018,PR_CM2_2018_Oc,PR_CM2_2019,PR_CM2_2019_Oc,PR_CM2_2020,PR_C
M2_2020_Oc)
legend('2018','2019','2020','FontSize',15)
title('P+R Cerny most 2','FontSize',20)
xlabel('time [Month, Day, Time]','FontSize',15)
ylabel('Occupancy [%]','FontSize',15)

end
```

Function PR_Cleaning

```
function PR_parking = PR_Cleaning(input)

%% Cleaning short

% set new variable based on the number of free places
input.free_places_cleaning=input.num_of_free_places;

% Outliers
errorwhen=[]; % set array with timestamp of outlier
errorvalue=[]; % set array with value of the outlier
% loop below finds the outlier based on the difference of previous and
% following value of free places
for i = 1:(height(input)-1)
    k(i,1)=abs(input.free_places_cleaning(i+1)-
input.free_places_cleaning(i));
    k(i,2)=25; % threshold for the standard approach

    % function below marks each outlier to the array
    if k(i,1) > 25 % threshold for the standard approach
        errorwhen=[errorwhen; input.updated_at(i)];
        errorvalue=[errorvalue; k(i)];
    end

end
error=[array2table(errorwhen) array2table(errorvalue)]; % creates table

% Time Outage
date_of_split_CM2_a=[]; % end of previous segment
date_of_split_CM2_b=[]; % start new segment
% loop below finds and records each time outage - records the timestamp
for i = 1:(height(input)-1)
    if abs(input.updated_at(i+1)-input.updated_at(i)) > minutes(30)
        date_of_split_CM2_a=[date_of_split_CM2_a; input.updated_at(i)]; %
end of the previous segment
        date_of_split_CM2_b=[date_of_split_CM2_b; input.updated_at(i+1)]; %
start of the following segment
    else
    end
end

%% Deleting extreme outliers
% deletes extreme outliers
deleterows=[];
% loop that detects outliers and records row of the outlier
for i = 1:(height(input)-1)
    if abs(input.num_of_free_places(i)-input.num_of_free_places(i+1)) >
input.total_num_of_places(1)/2 && (input.updated_at(i+1)-
input.updated_at(i)) < minutes(15)
        deleterows=[deleterows; i];
    end
end
% delete outliers from the table
input(deleterows,:)=[];

%% Cleaning - Movmean
% cleaning have to be used on the correct time data - therefore it is used
% in the continuous time segments

window=11; % set moving window to 11 records (30 minutes in previous and
following direction)
```

```
% first time segment
cleaned_free_places_first =
movmean(input.free_places_cleaning(isbetween(input.updated_at,input.updated
_at(1),date_of_split_CM2_a(1))), window);
% for free places in each segment it computes the moving average
cleaned_free_places_first = round(cleaned_free_places_first); % rounding to
the integer number of free parking places

% center time segments
cleaned_free_places_center=[];
for i = 1:(length(date_of_split_CM2_b)-1)
    movemean =
movmean(input.free_places_cleaning(isbetween(input.updated_at,date_of_split
_CM2_b(i),date_of_split_CM2_a(i+1))), window);
    % for free places in each segment it computes the moving average
    cleaned_free_places_center = [cleaned_free_places_center; movemean]; %
save it as new array
end
cleaned_free_places_center = round(cleaned_free_places_center); % rounding
to the integer number of free parking places

% last time segment
cleaned_free_places_last =
movmean(input.free_places_cleaning(isbetween(input.updated_at,date_of_split
_CM2_b(end),input.updated_at(end))), window);
cleaned_free_places_last = round(cleaned_free_places_last); % rounding to
the integer number of free parking places

% together
cleaned_free_places = [cleaned_free_places_first;
cleaned_free_places_center; cleaned_free_places_last]; % set array

% set cleaned data as a new variable
input.cleaned_num_of_free_places = cleaned_free_places;

%% Compute new occupancy - Occupancy2

for k = 1:height(input)
    Oc(k)=(input.total_num_of_places(k) -
input.cleaned_num_of_free_places(k))/input.total_num_of_places(k)*100;
    Oc(k)=round(Oc(k));
end
Oc=Oc';
Oc=array2table(Oc);
Oc.Properties.VariableNames = {'Occupancy2'};
input=[input Oc];

%% Set output table
% creates output variable
PR_parking = input;

end
```

Function PR_figure_yearcomparison2

```
function PR_figure_yearcomparison2(PR_L, PR_L_cleaned, PR_CM2,
PR_CM2_cleaned)
```

```
%% Figure prepairing

% Letnany - 2018
PR_L_2018 = PR_L.updated_at(PR_L.updated_at.Year==2018);
PR_L_2018 = datetime(PR_L_2018,'InputFormat','dd-MMM-yyyy HH:mm:ss',
'Format','dd-MMM HH:mm:ss');
PR_L_2018_Oc = PR_L.Occupancy(PR_L.updated_at.Year==2018);

PR_L_2018_2 = PR_L_cleaned.updated_at(PR_L_cleaned.updated_at.Year==2018);
PR_L_2018_2 = datetime(PR_L_2018_2,'InputFormat','dd-MMM-yyyy HH:mm:ss',
'Format','dd-MMM HH:mm:ss');
PR_L_2018_Oc2 =
PR_L_cleaned.Occupancy2(PR_L_cleaned.updated_at.Year==2018);

% Letnany - 2019
PR_L_2019 = PR_L.updated_at(PR_L.updated_at.Year==2019);
PR_L_2019 = PR_L_2019 - calyears(1) + days(1);
PR_L_2019 = datetime(PR_L_2019,'InputFormat','dd-MMM-yyyy HH:mm:ss',
'Format','dd-MMM HH:mm:ss');
PR_L_2019_Oc = PR_L.Occupancy(PR_L.updated_at.Year==2019);

PR_L_2019_2 = PR_L_cleaned.updated_at(PR_L_cleaned.updated_at.Year==2019);
PR_L_2019_2 = PR_L_2019_2 - calyears(1) + days(1);
PR_L_2019_2 = datetime(PR_L_2019_2,'InputFormat','dd-MMM-yyyy HH:mm:ss',
'Format','dd-MMM HH:mm:ss');
PR_L_2019_Oc2 =
PR_L_cleaned.Occupancy2(PR_L_cleaned.updated_at.Year==2019);

% Letnany - 2020
PR_L_2020 = PR_L.updated_at(PR_L.updated_at.Year==2020);
PR_L_2020 = PR_L_2020 - calyears(2) + days(3);
PR_L_2020 = datetime(PR_L_2020,'InputFormat','dd-MMM-yyyy HH:mm:ss',
'Format','dd-MMM HH:mm:ss');
PR_L_2020_Oc = PR_L.Occupancy(PR_L.updated_at.Year==2020);

PR_L_2020_2 = PR_L_cleaned.updated_at(PR_L_cleaned.updated_at.Year==2020);
PR_L_2020_2 = PR_L_2020_2 - calyears(2) + days(3);
PR_L_2020_2 = datetime(PR_L_2020_2,'InputFormat','dd-MMM-yyyy HH:mm:ss',
'Format','dd-MMM HH:mm:ss');
PR_L_2020_Oc2 =
PR_L_cleaned.Occupancy2(PR_L_cleaned.updated_at.Year==2020);

% Cerny Most 2 - 2018
PR_CM2_2018 = PR_CM2.updated_at(PR_CM2.updated_at.Year==2018);
PR_CM2_2018 = datetime(PR_CM2_2018,'InputFormat','dd-MMM-yyyy HH:mm:ss',
'Format','dd-MMM HH:mm:ss');
PR_CM2_2018_Oc = PR_CM2.Occupancy(PR_CM2.updated_at.Year==2018);

PR_CM2_2018_2 =
PR_CM2_cleaned.updated_at(PR_CM2_cleaned.updated_at.Year==2018);
PR_CM2_2018_2 = datetime(PR_CM2_2018_2,'InputFormat','dd-MMM-yyyy
HH:mm:ss', 'Format','dd-MMM HH:mm:ss');
PR_CM2_2018_Oc2 =
PR_CM2_cleaned.Occupancy2(PR_CM2_cleaned.updated_at.Year==2018);

% Cerny Most 2 - 2019
PR_CM2_2019 = PR_CM2.updated_at(PR_CM2.updated_at.Year==2019);
PR_CM2_2019 = PR_CM2_2019 - calyears(1) + days(1);
```

```
PR_CM2_2019 = datetime(PR_CM2_2019,'InputFormat','dd-MMM-yyyy HH:mm:ss',
'Format','dd-MMM HH:mm:ss');
PR_CM2_2019_Oc = PR_CM2.Occupancy(PR_CM2.updated_at.Year==2019);

PR_CM2_2019_2 =
PR_CM2_cleaned.updated_at(PR_CM2_cleaned.updated_at.Year==2019);
PR_CM2_2019_2 = PR_CM2_2019_2 - calyears(1) + days(1);
PR_CM2_2019_2 = datetime(PR_CM2_2019_2,'InputFormat','dd-MMM-yyyy
HH:mm:ss', 'Format','dd-MMM HH:mm:ss');
PR_CM2_2019_Oc2 =
PR_CM2_cleaned.Occupancy2(PR_CM2_cleaned.updated_at.Year==2019);

% Cerny Most 2 - 2020
PR_CM2_2020 = PR_CM2.updated_at(PR_CM2.updated_at.Year==2020);
PR_CM2_2020 = PR_CM2_2020 - calyears(2) + days(3);
PR_CM2_2020 = datetime(PR_CM2_2020,'InputFormat','dd-MMM-yyyy HH:mm:ss',
'Format','dd-MMM HH:mm:ss');
PR_CM2_2020_Oc = PR_CM2.Occupancy(PR_CM2.updated_at.Year==2020);

PR_CM2_2020_2 =
PR_CM2_cleaned.updated_at(PR_CM2_cleaned.updated_at.Year==2020);
PR_CM2_2020_2 = PR_CM2_2020_2 - calyears(2) + days(3);
PR_CM2_2020_2 = datetime(PR_CM2_2020_2,'InputFormat','dd-MMM-yyyy
HH:mm:ss', 'Format','dd-MMM HH:mm:ss');
PR_CM2_2020_Oc2 =
PR_CM2_cleaned.Occupancy2(PR_CM2_cleaned.updated_at.Year==2020);


%% Figure

figure(1)
subplot(2,1,1)
plot(PR_L_2018,PR_L_2018_Oc,PR_L_2019,PR_L_2019_Oc,PR_L_2020,PR_L_2020_Oc)
legend('2018','2019','2020','FontSize',15)
title('P+R Letnany - Original data','FontSize',20)
xlabel('time [Month, Day, Time]','FontSize',15)
ylabel('Occupancy [%]','FontSize',15)

subplot(2,1,2)
plot(PR_L_2018_2,PR_L_2018_Oc2,PR_L_2019_2,PR_L_2019_Oc2,PR_L_2020_2,PR_L_2
020_Oc2)
legend('2018','2019','2020','FontSize',15)
title('P+R Letnany - Cleaned data','FontSize',20)
xlabel('time [Month, Day, Time]','FontSize',15)
ylabel('Occupancy [%]','FontSize',15)

figure(2)
subplot(2,1,1)
plot(PR_CM2_2018,PR_CM2_2018_Oc,PR_CM2_2019,PR_CM2_2019_Oc,PR_CM2_2020,PR_C
M2_2020_Oc)
legend('2018','2019','2020','FontSize',15)
title('P+R Cerny most 2 - Original data','FontSize',20)
xlabel('time [Month, Day, Time]','FontSize',15)
ylabel('Occupancy [%]','FontSize',15)

subplot(2,1,2)
plot(PR_CM2_2018_2,PR_CM2_2018_Oc2,PR_CM2_2019_2,PR_CM2_2019_Oc2,PR_CM2_202
0_2,PR_CM2_2020_Oc2)
legend('2018','2019','2020','FontSize',15)
title('P+R Cerny most 2 - Cleaned data','FontSize',20)
```

```
xlabel('time [Month, Day, Time]','FontSize',15)
ylabel('Occupancy [%]','FontSize',15)

end
```

Function PR_Days

```
function PR_days(data,PublicHolidays,directory)

%% Round time to the nearest 5th minute
% set new format of timestamp data - withou seconds
data.updated_at = datetime(data.updated_at,'InputFormat', 'dd-MMM-yyyy
HH:mm:ss','Format', 'dd-MMM-yyyy HH:mm');
% round timestamp to 5 minute intervals
data.updated_at.Minute = 5 * round(data.updated_at.Minute/5);

%% Set timestamp do dates
% set new format of timestamp data - days withou hours and minutes
data.updated_at2 = datetime(data.updated_at,'InputFormat', 'dd-MMM-yyyy
HH:mm','Format', 'dd-MMM-yyyy');
% round timestamp days
data.updated_at2 = dateshift(data.updated_at2, 'start', 'day');

%% Set weekdays (takes about 12 minutes of computing)

DayNames=[];
% loop below creates new variable with a name of the day by the function
% weekday
for i = 1:height(data)
    [DayNumber, DayName] = weekday(data.updated_at2(i));
    DayNames = [DayNames; convertCharsToStrings(DayName)];
end
data.days=DayNames; % add variable to the table

%% Set summer holidays
% set code SH (Summer Holidays) to the exact days
data.days(month(data.updated_at2)==7) =
data.days(month(data.updated_at2)==7)+'SH';
data.days(month(data.updated_at2)==8) =
data.days(month(data.updated_at2)==8)+'SH';

%% Set public holidays CZ

% loop below sets code PH (Public Holidays) to the exact days set above
for i = 1:length(PublicHolidays)

data.days(isbetween(data.updated_at2,PublicHolidays(i),PublicHolidays(i)))
=
data.days(isbetween(data.updated_at2,PublicHolidays(i),PublicHolidays(i)))+
'PH';
end


%% Calculated days

% start is the first whole day in 2018 and end is before the pandemic in
2020
data2 = data(isbetween(data.updated_at2, datetime('18-May-2018'),
datetime('29-Feb-2020')),:);
```

```
% writedown unique days
Unique_Days = unique(data2.days);

%% Set time of day
% set the beginning and ending of the day
start_of_day = datetime('00:00','Format', 'HH:mm');
end_of_day = datetime('23:55','Format', 'HH:mm');
% day time in 5 minute intervals
daytime = (start_of_day:minutes(5):end_of_day)';

%% Creates CSV files for each unique day
% change type of directory of the export (from char to string)
directory = convertCharsToStrings(directory);

for i = 1:length(Unique_Days)

    the_weekday_all = []; % cleaned table after round of calculation
    the_day = []; % cleaned table after round of calculation

    % creates table of timestamp, day and number of free places for the
    % i-th unique day (such as Friday)
    the_weekday_all.updated_at =
data2.updated_at(data2.days==Unique_Days(i));
    the_weekday_all.updated_at2 =
data2.updated_at2(data2.days==Unique_Days(i));
    the_weekday_all.cleaned_num_of_free_places =
data2.cleaned_num_of_free_places(data2.days==Unique_Days(i));
    % change structure to table
    the_weekday_all=struct2table(the_weekday_all);
    % sets unique dats for the unique day (such as Friday 18-May-2018)
    Unique_Dates = unique(the_weekday_all.updated_at2);

    % loop below provides the whole computation - the final is a table for
    % each unique day
    for j = 1:length(Unique_Dates)

        the_weekday = []; % cleaned table after round of calculation
        % creates table of time of day and number of free places for each
        % unique day in unique date
        the_weekday.time_of_day =
the_weekday_all.updated_at(the_weekday_all.updated_at2==Unique_Dates(j));
        the_weekday.cleaned_num_of_free_places =
the_weekday_all.cleaned_num_of_free_places(the_weekday_all.updated_at2==Uni
que_Dates(j));
        % change structure to table and format of the timestamp
        the_weekday=struct2table(the_weekday);

the_weekday.time_of_day=datetime(the_weekday.time_of_day,'Format','HH:mm');

        % it is used for the computation purposes - the data are not
        % complete for all days
        l = 1;

        for k = 1:length(daytime) % 5 minute interval

            if (the_weekday.time_of_day(l) - the_weekday.time_of_day(end))
== minutes(0)
                % if the data ends before the end of the day takes previous
                % value
                the_day(k,j) = the_weekday.cleaned_num_of_free_places(l-1);

            elseif the_weekday.time_of_day(l).Hour == daytime(k).Hour &&
the_weekday.time_of_day(l).Minute == daytime(k).Minute
                % if data is correct it records the value
                the_day(k,j) = the_weekday.cleaned_num_of_free_places(l);
                l=l+1; % move in the weekday table - the data are not
complete for all days

            elseif the_weekday.time_of_day(l).Hour == daytime(2).Hour &&
the_weekday.time_of_day(l).Minute == daytime(2).Minute
                % if the data records start later than day it takes
                % following value
                the_day(k,j) = the_weekday.cleaned_num_of_free_places(l+1);

            elseif l-1 == 0
                % if there are no previsou records - day starts at 4 a.m.
                the_day(k,j) = 0;

            else
                % in the case - there are no data record it takes average
                % of previous and following data record
                the_day(k,j) =
round((the_weekday.cleaned_num_of_free_places(l-
1)+the_weekday.cleaned_num_of_free_places(l+1))/2);

            end
        end
    end
    the_day = array2table(the_day); % change structure to table
    % save table as a CSV file
    % colums are dates and rows is daytime
    day_name = sprintf('%s.csv',Unique_Days(i));
    writetable(the_day,directory+day_name,'Delimiter',',');
end

end
```

Function PR_CM2_days_figure

```
function PR_days_figure(directory)

%% Read csv files

csvfile=string('*.csv');

fileName = dir(directory+csvfile);
fileName_cell=struct2cell(fileName);

figure(1)
for i = 1:length(fileName_cell)
   subplot(6,4,i)
   a = table2array(readtable(directory+string(fileName_cell(1,i))));
   b = size(a);
   if b(2) == 1
       a(:,2) = a(:,1);
       mesh(a);
       title(sprintf('1day-%s',string(fileName_cell(1,i))),'FontSize',15);
       xlabel({'Week'; '[05-2018 to 02-2020]'},'FontSize',10);
```

```
        ylabel({'5-min interval'; '[00:00 to 23:55]'},'FontSize',10);
        zlabel({'Free parking places'; '[0 - 131]'},'FontSize',10);
    else
        mesh(a);
        title(sprintf('%s',string(fileName_cell(1,i))),'FontSize',15);
        xlabel({'Week'; '[05-2018 to 02-2020]'},'FontSize',10);
        ylabel({'5-min interval'; '[00:00 to 23:55]'},'FontSize',10);
        zlabel({'Free parking places'; '[0 - 131]'},'FontSize',10);
    end
end

end
```

Function PR_CM2_days_figure

```
function PR_L_days_figure(directory)

%% Read csv files

csvfile=string('*.csv');

fileName = dir(directory+csvfile);
fileName_cell=struct2cell(fileName);

figure(1)
for i = 1:length(fileName_cell)
    subplot(6,4,i)
    a = table2array(readtable(directory+string(fileName_cell(1,i))));
    b = size(a);
    if b(2) == 1
        a(:,2) = a(:,1);
        mesh(a);
        title(sprintf('1day-%s',string(fileName_cell(1,i))),'FontSize',15);
        xlabel({'Week'; '[05-2018 to 02-2020]'},'FontSize',10);
        ylabel({'5-min interval'; '[00:00 to 23:55]'},'FontSize',10);
        zlabel({'Free parking places'; '[0 - 633]'},'FontSize',10);
    else
        mesh(a);
        title(sprintf('%s',string(fileName_cell(1,i))),'FontSize',15);
        xlabel({'Week'; '[05-2018 to 02-2020]'},'FontSize',10);
        ylabel({'5-min interval'; '[00:00 to 23:55]'},'FontSize',10);
        zlabel({'Free parking places'; '[0 - 633]'},'FontSize',10);
    end
end

end
```

ZPS Prague

The main code

```
%% Initialization

clear
close all
```

```
%% Read CSV files from the current folder
ds = datastore('/Users/viktorbenes/diplomka/Prague ZPS/data/*.csv');
% set folder as a datastore

Table_alldata = readall(ds);
% read all data from datastore to one table

%% Filter the zone P7-0158
% other zones: 'P7-0179' 'P7-0161' 'P7-0156' 'P7-0134'
zone = 'P7-0158';
zone = convertCharsToStrings(zone);
% filter for the specific paid parking zone
ZPS_filter(Table_alldata,zone);

%% Import tables (csv)

P7_0158=readtable('/Users/viktorbenes/diplomka/Prague ZPS/ZPS_P7-
0158.csv');
P7_0179=readtable('/Users/viktorbenes/diplomka/Prague ZPS/ZPS_P7-
0179.csv');
P7_0161=readtable('/Users/viktorbenes/diplomka/Prague ZPS/ZPS_P7-
0161.csv');
P7_0156=readtable('/Users/viktorbenes/diplomka/Prague ZPS/ZPS_P7-
0156.csv');
P7_0134=readtable('/Users/viktorbenes/diplomka/Prague ZPS/ZPS_P7-
0134.csv');

%% Cleaning ZPS

P7_0158_cleaned = ZPS_cleaning(P7_0158);
P7_0179_cleaned = ZPS_cleaning(P7_0179);
P7_0161_cleaned = ZPS_cleaning(P7_0161);
P7_0156_cleaned = ZPS_cleaning(P7_0156);
P7_0134_cleaned = ZPS_cleaning(P7_0134);

%% Set public holidays

% dates of Czech Public Holidays
PublicHolidays=['01-01-2018'; '30-03-2018'; '02-04-2018'; '01-05-2018';
    '08-05-2018'; '05-07-2018'; '06-07-2018'; '28-09-2018'; '28-10-2018';
    '17-11-2018'; '24-12-2018'; '25-12-2018'; '26-12-2018';
    '01-01-2019'; '19-04-2019'; '22-04-2019'; '01-05-2019'; '08-05-2019';
    '05-07-2019'; '06-07-2019'; '28-09-2019'; '28-10-2019'; '17-11-2019';
    '24-12-2019'; '25-12-2019'; '26-12-2019';
    '01-01-2020'; '10-04-2020'; '13-04-2020'; '01-05-2020'; '08-05-2020';
    '05-07-2020'; '06-07-2020'; '28-09-2020'; '28-10-2020'; '17-11-2020';
    '24-12-2020'; '25-12-2020'; '26-12-2020'];
PublicHolidays=datetime(PublicHolidays,'InputFormat','dd-MM-yyyy');

%% Day comparison - P7-0158

directoryP7_0158 = '/Users/viktorbenes/diplomka/Prague ZPS/P7-0158/';
capacity0158 = 22;
ZPS_days(P7_0158_cleaned,PublicHolidays,directoryP7_0158,capacity0158)

%% Figure Day comparison - P7-0158

P7_0158_days_figure(directoryP7_0158)
```

```
%% Day comparison - P7-0179

directoryP7_0179 = '/Users/viktorbenes/diplomka/Prague ZPS/P7-0179/';
capacity0179 = 61;
ZPS_days(P7_0179_cleaned,PublicHolidays,directoryP7_0179,capacity0179)

%% Figure Day comparison - P7-0179

P7_0179_days_figure(directoryP7_0179)

%% Day comparison - P7-0161

directoryP7_0161 = '/Users/viktorbenes/diplomka/Prague ZPS/P7-0161/';
capacity0161 = 51;
ZPS_days(P7_0161_cleaned,PublicHolidays,directoryP7_0161,capacity0161)

%% Figure Day comparison - P7-0161

P7_0161_days_figure(directoryP7_0161)

%% Day comparison - P7-0156

directoryP7_0156 = '/Users/viktorbenes/diplomka/Prague ZPS/P7-0156/';
capacity0156 = 49;
ZPS_days(P7_0156_cleaned,PublicHolidays,directoryP7_0156,capacity0156)

%% Figure Day comparison - P7-0156

P7_0156_days_figure(directoryP7_0156)

%% Day comparison - P7-0134

directoryP7_0134 = '/Users/viktorbenes/diplomka/Prague ZPS/P7-0134/';
capacity0134 = 94;
ZPS_days(P7_0134_cleaned,PublicHolidays,directoryP7_0134,capacity0134)

%% Figure Day comparison - P7-0134

P7_0134_days_figure(directoryP7_0134)
```

ZPS_cleaning

```
function output = ZPS_cleaning(input)

%% Round time to the nearest 5th minute
% set new format of timestamp (velidity from) data - without seconds
input.validity_from = datetime(input.validity_from,'InputFormat', 'dd-MMM-
yyyy HH:mm:ss','Format', 'dd-MMM-yyyy HH:mm');
% round timestamp (velidity from) to 5 minute intervals
input.validity_from.Minute = 5 * round(input.validity_from.Minute/5);

% set new format of timestamp (velidity to) data - without seconds
input.validity_to = datetime(input.validity_to,'InputFormat', 'dd-MMM-yyyy
HH:mm:ss','Format', 'dd-MMM-yyyy HH:mm');
% round timestamp (velidity to) to 5 minute intervals
input.validity_to.Minute = 5 * round(input.validity_to.Minute/5);
```

```
%% Set timestamp do dates
% set new format of timestamp data - days withou hours and minutes
input.validity_from2 = datetime(input.validity_from,'InputFormat', 'dd-MMM-
yyyy HH:mm','Format', 'dd-MMM-yyyy');
% round timestamp days
input.validity_from2 = dateshift(input.validity_from2, 'start', 'day');

%% Deleting nonaccurate dates
% start of the measuring
starting = datetime('01-Jan-2018','InputFormat', 'dd-MMM-yyyy');
% end of the measuring because of the pandemic situation
ending = datetime('29-Feb-2020','InputFormat', 'dd-MMM-yyyy');

delete_records=[];
% loop that detects invalid dates and records row of the them into
% delete_records array
for i = 1:height(input)
    if input.validity_from2(i) >= starting && input.validity_from2(i) <=
ending
    else
        delete_records=[delete_records; i];
    end
end
% delete invalid dates from the table
input(delete_records,:)=[];

output = input;

end
```

ZPS_days

```
function Overall = ZPS_days(data,PublicHolidays,directory,capacity)

%% Set unique dates
% writedown unique dates
Unique_dates = unique(data.validity_from2);

%% Set time of day
% set the beginning and ending of the day
start_of_day = datetime('00:00','Format', 'HH:mm');
end_of_day = datetime('23:55','Format', 'HH:mm');
% day time in 5 minute intervals
daytime = (start_of_day:minutes(5):end_of_day)';

%% Computing when each ticket is valid during day
% matrix of zeros for the loop below
Tickets = zeros(height(data),length(daytime));
% loop creates for each ticket daytime
% 0 is for invalid time and 1 is for valid time of the ticket
for j = 1:height(data)

    time_lower =
datetime(sprintf('%d:%d',data.validity_from(j).Hour,data.validity_from(j).M
inute),'Format','HH:mm');
```

```
    time_upper =
datetime(sprintf('%d:%d',data.validity_to(j).Hour,data.validity_to(j).Minut
e),'Format','HH:mm');

    for k = find(daytime==time_lower):find(daytime==time_upper)
            Tickets(j,k) = 1;
    end
end
Tickets = array2table(Tickets);
Tickets.date = data.validity_from2;

%% Sum of tickets for each day

Overall = [];
Overall = array2table(Overall);

% this loop sum tickets per day and creates overall table where is daytime
% per day in one table
for i = 1:length(Unique_dates)

    TicketsPerDay = Tickets(Tickets.date==Tickets.date(i),1:end-1);

    for j = 1:(width(Tickets)-1)

        SumOfTickets(i,j) = sum(TicketsPerDay{:,j});

        Overall = [Overall; table(SumOfTickets(i,j), daytime(j),
Unique_dates(i))];

    end
end

Overall.Properties.VariableNames = {'num_of_taken_places' 'time' 'date'};

%% Set day names

DayNames=[];
% loop below creates new variable with a name of the day by the function
% weekday
for i = 1:height(Overall)
    [DayNumber, DayName] = weekday(Overall.date(i));
    DayNames = [DayNames; convertCharsToStrings(DayName)];
end
Overall.days=DayNames; % add variable to the table

%% Set summer holidays
% set code SH (Summer Holidays) to the exact days
Overall.days(month(Overall.date)==7) =
Overall.days(month(Overall.date)==7)+'SH';
Overall.days(month(Overall.date)==8) =
Overall.days(month(Overall.date)==8)+'SH';

%% Set public holidays CZ

% loop below sets code PH (Public Holidays) to the exact days set above
for i = 1:length(PublicHolidays)

Overall.days(isbetween(Overall.date,PublicHolidays(i),PublicHolidays(i))) =
Overall.days(isbetween(Overall.date,PublicHolidays(i),PublicHolidays(i)))+'
PH';
end

%% Set unique days
% writedown unique days
Unique_Days = unique(Overall.days);

%% Counting number of free places

Overall.free_places = capacity - Overall.num_of_taken_places;

%% Creates CSV files for each unique day
% change type of directory of the export (from char to string)
directory = convertCharsToStrings(directory);

for i = 1:length(Unique_Days)

    the_weekday_all = []; % cleaned table after round of calculation
    the_day = []; % cleaned table after round of calculation

    % creates table of timestamp, day and number of free places for the
    % i-th unique day (such as Friday)
    the_weekday_all.time = Overall.time(Overall.days==Unique_Days(i));
    the_weekday_all.date = Overall.date(Overall.days==Unique_Days(i));
    the_weekday_all.free_places =
Overall.free_places(Overall.days==Unique_Days(i));
    % change structure to table
    the_weekday_all=struct2table(the_weekday_all);
    % unique dates in for each day
    Unique_dates2 = unique(the_weekday_all.date);

    % loop below provides the whole computation - the final is a table for
    % each unique day
    for j = 1:length(Unique_dates2)

        the_weekday = []; % cleaned table after round of calculation
        % creates table of time of day and number of free places for each
        % unique day in unique date
        the_weekday.time_of_day =
the_weekday_all.time(the_weekday_all.date==Unique_dates2(j));
        the_weekday.free_places =
the_weekday_all.free_places(the_weekday_all.date==Unique_dates2(j));
        % change structure to table and format of the timestamp
        the_weekday=struct2table(the_weekday);

the_weekday.time_of_day=datetime(the_weekday.time_of_day,'Format','HH:mm');

        % it is used for the computation purposes - the data are not
        % complete for all days
        l = 1;

        for k = 1:length(daytime) % 5 minute interval

            if (the_weekday.time_of_day(l) - the_weekday.time_of_day(end))
== minutes(0)
                % if the data ends before the end of the day takes previous
                % value
                the_day(k,j) = the_weekday.free_places(l-1);
            elseif the_weekday.time_of_day(l).Hour == daytime(k).Hour &&
the_weekday.time_of_day(l).Minute == daytime(k).Minute
                % if data is correct it records the value
                the_day(k,j) = the_weekday.free_places(l);
                l=l+1; % move in the weekday table - the data are not
complete for all days

            elseif the_weekday.time_of_day(l).Hour == daytime(2).Hour &&
the_weekday.time_of_day(l).Minute == daytime(2).Minute
                % if the data records start later than day it takes
                % following value
                the_day(k,j) = the_weekday.free_places(l+1);

            elseif l-1 == 0
                % if there are no previsou records - day starts at 4 a.m.
                the_day(k,j) = 0;

            else
                % in the case - there are no data record it takes average
                % of previous and following data record
                the_day(k,j) = round((the_weekday.free_places(l-
1)+the_weekday.free_places(l+1))/2);

            end
        end
    end
    the_day = array2table(the_day); % change structure to table
    % save table as a CSV file
    % colums are dates and rows is daytime
    day_name = sprintf('%s.csv',Unique_Days(i));
    writetable(the_day,directory+day_name,'Delimiter',',');
end
```

P7_0134_days_figure

```
function P7_0134_days_figure(directory)

%% Read csv files

csvfile=string('*.csv');

fileName = dir(directory+csvfile);
fileName_cell=struct2cell(fileName);

figure(1)
for i = 1:length(fileName_cell)
   subplot(6,4,i)
   a = table2array(readtable(directory+string(fileName_cell(1,i))));
   b = size(a);
   if b(2) == 1
       a(:,2) = a(:,1);
       mesh(a);
       title(sprintf('1day-%s',string(fileName_cell(1,i))),'FontSize',15);
       xlabel({'Week'; '[01-2018 to 02-2020]'},'FontSize',10);
       ylabel({'5-min interval'; '[00:00 to 23:55]'},'FontSize',10);
       zlabel({'Free parking places'; '[0 - 94]'},'FontSize',10);
   else
```

```
        mesh(a);
        title(sprintf('%s',string(fileName_cell(1,i))),'FontSize',15);
        xlabel({'Week'; '[01-2018 to 02-2020]'},'FontSize',10);
        ylabel({'5-min interval'; '[00:00 to 23:55]'},'FontSize',10);
        zlabel({'Free parking places'; '[0 - 94]'},'FontSize',10);
    end
end

end
```

P7_0156_days_figure

```
function P7_0156_days_figure(directory)

%% Read csv files

csvfile=string('*.csv');

fileName = dir(directory+csvfile);
fileName_cell=struct2cell(fileName);

figure(1)
for i = 1:length(fileName_cell)
    subplot(6,4,i)
    a = table2array(readtable(directory+string(fileName_cell(1,i))));
    b = size(a);
    if b(2) == 1
        a(:,2) = a(:,1);
        mesh(a);
        title(sprintf('1day-%s',string(fileName_cell(1,i))),'FontSize',15);
        xlabel({'Week'; '[01-2018 to 02-2020]'},'FontSize',10);
        ylabel({'5-min interval'; '[00:00 to 23:55]'},'FontSize',10);
        zlabel({'Free parking places'; '[0 - 49]'},'FontSize',10);
    else
        mesh(a);
        title(sprintf('%s',string(fileName_cell(1,i))),'FontSize',15);
        xlabel({'Week'; '[01-2018 to 02-2020]'},'FontSize',10);
        ylabel({'5-min interval'; '[00:00 to 23:55]'},'FontSize',10);
        zlabel({'Free parking places'; '[0 - 49]'},'FontSize',10);
    end
end

end
```

P7_0158_days_figure

```
function P7_0158_days_figure(directory)

%% Read csv files

csvfile=string('*.csv');

fileName = dir(directory+csvfile);
fileName_cell=struct2cell(fileName);

figure(1)
for i = 1:length(fileName_cell)
    subplot(6,4,i)
   a = table2array(readtable(directory+string(fileName_cell(1,i))));
   b = size(a);
   if b(2) == 1
       a(:,2) = a(:,1);
       mesh(a);
       title(sprintf('1day-%s',string(fileName_cell(1,i))),'FontSize',15);
       xlabel({'Week'; '[01-2018 to 02-2020]'},'FontSize',10);
       ylabel({'5-min interval'; '[00:00 to 23:55]'},'FontSize',10);
       zlabel({'Free parking places'; '[0 - 22]'},'FontSize',10);
   else
       mesh(a);
       title(sprintf('%s',string(fileName_cell(1,i))),'FontSize',15);
       xlabel({'Week'; '[01-2018 to 02-2020]'},'FontSize',10);
       ylabel({'5-min interval'; '[00:00 to 23:55]'},'FontSize',10);
       zlabel({'Free parking places'; '[0 - 22]'},'FontSize',10);
   end
end

end
```

P7_0161_days_figure

```
function P7_0161_days_figure(directory)

%% Read csv files

csvfile=string('*.csv');

fileName = dir(directory+csvfile);
fileName_cell=struct2cell(fileName);

figure(1)
for i = 1:length(fileName_cell)
    subplot(6,4,i)
    a = table2array(readtable(directory+string(fileName_cell(1,i))));
    b = size(a);
    if b(2) == 1
        a(:,2) = a(:,1);
        mesh(a);
        title(sprintf('1day-%s',string(fileName_cell(1,i))),'FontSize',15);
        xlabel({'Week'; '[01-2018 to 02-2020]'},'FontSize',10);
        ylabel({'5-min interval'; '[00:00 to 23:55]'},'FontSize',10);
        zlabel({'Free parking places'; '[0 - 51]'},'FontSize',10);
    else
        mesh(a);
        title(sprintf('%s',string(fileName_cell(1,i))),'FontSize',15);
        xlabel({'Week'; '[01-2018 to 02-2020]'},'FontSize',10);
        ylabel({'5-min interval'; '[00:00 to 23:55]'},'FontSize',10);
        zlabel({'Free parking places'; '[0 - 51]'},'FontSize',10);
    end
end

end
```

P7_0179_days_figure

```
function P7_0179_days_figure(directory)
```

```
%% Read csv files

csvfile=string('*.csv');

fileName = dir(directory+csvfile);
fileName_cell=struct2cell(fileName);

figure(1)
for i = 1:length(fileName_cell)
   subplot(6,4,i)
   a = table2array(readtable(directory+string(fileName_cell(1,i))));
   b = size(a);
   if b(2) == 1
       a(:,2) = a(:,1);
       mesh(a);
       title(sprintf('1day-%s',string(fileName_cell(1,i))),'FontSize',15);
       xlabel({'Week'; '[01-2018 to 02-2020]'},'FontSize',10);
       ylabel({'5-min interval'; '[00:00 to 23:55]'},'FontSize',10);
       zlabel({'Free parking places'; '[0 - 61]'},'FontSize',10);
   else
       mesh(a);
       title(sprintf('%s',string(fileName_cell(1,i))),'FontSize',15);
       xlabel({'Week'; '[01-2018 to 02-2020]'},'FontSize',10);
       ylabel({'5-min interval'; '[00:00 to 23:55]'},'FontSize',10);
       zlabel({'Free parking places'; '[0 - 61]'},'FontSize',10);
   end
end

end
```

Parking garages Norrköping

The main code

```
%% Initialization

clear
close all

%% input Spiran Parkering

capacityS = 388; % capacity of the parking
directoryS = '/Users/viktorbenes/diplomka/Norrkoping Parking/Spiran/';
Spiran = data_input(directoryS, capacityS);

%% input Vardtornet Parkering

capacityV = 84; % capacity of the parking
directoryV = '/Users/viktorbenes/diplomka/Norrkoping Parking/Vardtornet/';
Vardtornet = data_input(directoryV, capacityV);

%% Figure 1 - Original data - year comparison
% plot raw data
Norrkoping_figure_yearcomparison(Spiran, Vardtornet);

%% Cleaning - Spiran
```

```
% data cleaning
Spiran_cleaned = Norrkoping_Cleaning(Spiran, capacityS);

%% Cleaning - Vardtornet
% data cleaning
Vardtornet_cleaned = Norrkoping_Cleaning(Vardtornet, capacityV);

%% Figure 2 - Cleaned data - year comparison
% plot cleaned data - not necessary
Norrkoping_figure_yearcomparison2(Spiran, Spiran_cleaned, Vardtornet,
Vardtornet_cleaned);

%% Set public holidays

% dates of Swedish Public Holidays
PublicHolidays = ['01-01-2018'; '06-01-2018'; '30-03-2018'; '02-04-2018';
    '01-05-2018'; '10-05-2018'; '06-06-2018'; '23-06-2018'; '03-11-2018';
    '25-12-2018'; '26-12-2018';
    '01-01-2019'; '06-01-2019'; '19-04-2019'; '22-04-2019'; '01-05-2019';
    '30-05-2019'; '06-06-2019'; '22-06-2019'; '02-11-2019'; '25-12-2019';
    '26-12-2019';
    '01-01-2020'; '06-01-2020'; '10-04-2020'; '13-04-2020'; '01-05-2020';
    '21-05-2020'; '06-06-2020'; '10-06-2020'; '31-10-2020'; '25-12-2020';
    '26-12-2020'];
PublicHolidays = datetime(PublicHolidays,'InputFormat','dd-MM-yyyy');

%% Set summer holidays

SH_dates = ['09-06-2018'; '19-08-2018';
    '08-06-2019'; '18-08-2019';
    '06-06-2020'; '16-08-2020'];
SH_dates = datetime(SH_dates,'InputFormat','dd-MM-yyyy');


%% Day comparison - Spiran

directoryS_days = '/Users/viktorbenes/diplomka/Norrkoping
Parking/Spiran_days/'
Norrkoping_days(Spiran_cleaned,PublicHolidays,SH_dates,directoryS_days)

%% Figure: Day comparison - Spiran

Spiran_days_figure(directoryS_days);

%% Day comparison - Vardtornet

directoryV_days = '/Users/viktorbenes/diplomka/Norrkoping
Parking/Vardtornet_days/'
Norrkoping_days(Vardtornet_cleaned,PublicHolidays,SH_dates,directoryV_days)

%% Figure: Day comparison - Vardtornet

Vardtornet_days_figure(directoryV_days);
```

Data input

```
function data = data_input(directory, capacity)

%% Import data

allxlsx=string('*.xlsx');

fileName = dir(directory+allxlsx);
fileName_cell=struct2cell(fileName);

data=table();
for k = 1:length(fileName_cell)
    filetoread=string(fileName_cell(1,k));
    openfile=directory+filetoread;
    file=readtable(openfile);
    data=[data;file];
end

%% Set table

data.Properties.VariableNames = {'Timestamp' 'num_of_taken_places'
'occupancy' 'arrivals' 'departures'};

data.Timestamp=datetime(data.Timestamp,'InputFormat', 'yyyy-MM-
dd''T''HH:mm:ss.SSS''Z');

%% Free places computing

data.num_of_free_places = capacity - data.num_of_taken_places;

end
```

Norrköping_figure_yearcomparison

```
function Norrkoping_figure_yearcomparison(Spiran,Vardtornet)

%% Figure prepairing

% Spiran - 2018
Spiran_2018 = Spiran.Timestamp(Spiran.Timestamp.Year==2018);
Spiran_2018 = datetime(Spiran_2018,'InputFormat','dd-MMM-yyyy HH:mm:ss',
'Format','dd-MMM HH:mm:ss');
Spiran_2018_Oc = Spiran.occupancy(Spiran.Timestamp.Year==2018);

% Spiran - 2019
Spiran_2019 = Spiran.Timestamp(Spiran.Timestamp.Year==2019);
Spiran_2019 = Spiran_2019 - calyears(1) + days(1);
Spiran_2019 = datetime(Spiran_2019,'InputFormat','dd-MMM-yyyy HH:mm:ss',
'Format','dd-MMM HH:mm:ss');
Spiran_2019_Oc = Spiran.occupancy(Spiran.Timestamp.Year==2019);

% Spiran - 2020
Spiran_2020 = Spiran.Timestamp(Spiran.Timestamp.Year==2020);
Spiran_2020 = Spiran_2020 - calyears(2) + days(3);
Spiran_2020 = datetime(Spiran_2020,'InputFormat','dd-MMM-yyyy HH:mm:ss',
'Format','dd-MMM HH:mm:ss');
Spiran_2020_Oc = Spiran.occupancy(Spiran.Timestamp.Year==2020);
```

```
% Vardtornet - 2018
Vardtornet_2018 = Vardtornet.Timestamp(Vardtornet.Timestamp.Year==2018);
Vardtornet_2018 = datetime(Vardtornet_2018,'InputFormat','dd-MMM-yyyy
HH:mm:ss', 'Format','dd-MMM HH:mm:ss');
Vardtornet_2018_Oc = Vardtornet.occupancy(Vardtornet.Timestamp.Year==2018);

% Vardtornet - 2019
Vardtornet_2019 = Vardtornet.Timestamp(Vardtornet.Timestamp.Year==2019);
Vardtornet_2019 = Vardtornet_2019 - calyears(1) + days(1);
Vardtornet_2019 = datetime(Vardtornet_2019,'InputFormat','dd-MMM-yyyy
HH:mm:ss', 'Format','dd-MMM HH:mm:ss');
Vardtornet_2019_Oc = Vardtornet.occupancy(Vardtornet.Timestamp.Year==2019);

% Vardtornet - 2020
Vardtornet_2020 = Vardtornet.Timestamp(Vardtornet.Timestamp.Year==2020);
Vardtornet_2020 = Vardtornet_2020 - calyears(2) + days(3);
Vardtornet_2020 = datetime(Vardtornet_2020,'InputFormat','dd-MMM-yyyy
HH:mm:ss', 'Format','dd-MMM HH:mm:ss');
Vardtornet_2020_Oc = Vardtornet.occupancy(Vardtornet.Timestamp.Year==2020);


%% Figure

figure(1)
subplot(2,1,1)
plot(Spiran_2018,Spiran_2018_Oc,Spiran_2019,Spiran_2019_Oc,Spiran_2020,Spir
an_2020_Oc)
legend('2018','2019','2020','FontSize',15)
title('Spiran','FontSize',20)
xlabel('time [Month, Day, Time]','FontSize',15)
ylabel('occupancy [%]','FontSize',15)

subplot(2,1,2)
plot(Vardtornet_2018,Vardtornet_2018_Oc,Vardtornet_2019,Vardtornet_2019_Oc,
Vardtornet_2020,Vardtornet_2020_Oc)
legend('2018','2019','2020','FontSize',15)
title('Vardtornet','FontSize',20)
xlabel('time [Month, Day, Time]','FontSize',15)
ylabel('occupancy [%]','FontSize',15)

end
```

Norrköping_Cleaning

```
function cleaned_data = Norrkoping_Cleaning(input, capacity)

%% Cleaning short

% Outliers
errorwhen=[]; % set array with timestamp of outlier
errorvalue=[]; % set array with value of the outlier
% loop below finds the outlier based on the difference of previous and
% following value of free places
for i = 1:(height(input)-1)
    k(i,1)=abs(input.num_of_free_places(i+1)-input.num_of_free_places(i));
    k(i,2)=70; % threshold for the standard approach
    % function below marks each outlier to the array
    if k(i,1) > 70 % threshold for the standard approach
        errorwhen=[errorwhen; input.Timestamp(i)];
        errorvalue=[errorvalue; k(i)];
    end

end
error=[array2table(errorwhen) array2table(errorvalue)]; % creates table

% Time Outage
date_of_split_a=[]; % end of previous segment
date_of_split_b=[]; % start new segment
% loop below finds and records each time outage - records the timestamp
for i = 1:(height(input)-1)
    if abs(input.Timestamp(i+1)-input.Timestamp(i)) > minutes(35)
        date_of_split_a=[date_of_split_a; input.Timestamp(i)]; % end of the
previous segment
        date_of_split_b=[date_of_split_b; input.Timestamp(i+1)]; % start of
the following segment
    else
    end
end

%% Deleting extreme outliers
% deletes extreme outliers
deleterows=[];
% loop that detects outliers and records row of the outlier
for i = 1:(height(input)-1)
    if abs(input.num_of_free_places(i)-input.num_of_free_places(i+1)) >
capacity/2 && (input.Timestamp(i+1)-input.Timestamp(i)) < minutes(35)
        deleterows=[deleterows; i];
    end
end
% delete outliers from the table
input(deleterows,:)=[];

%% Untrusted data
% computes difference between variable
input.difference = [0; diff(input.num_of_free_places)];

deleteuntrusted = [];
dateuntrusted = [];

% loop that detects Untrusted data and records row of the Untrusted data
% and timestamp
% 48 records represents 12 hours
for i = 1:height(input)-48
    if sum(abs(input.difference(i:i+48))) == 0
        deleteuntrusted = [deleteuntrusted; i];
        dateuntrusted = [dateuntrusted; input.Timestamp(i)];
    end
end
% delete Untrusted data from the table
input(deleteuntrusted,:)=[];

%% Set output table
% creates output variable
cleaned_data = input;
```

```
end
```

Norrköping_figure_yearcomparison2

```
function PR_figure_yearcomparison2(Spiran, Spiran_cleaned, Vardtornet,
Vardtornet_cleaned)

%% Figure prepairing

% Spiran - 2018
Spiran_2018 = Spiran.Timestamp(Spiran.Timestamp.Year==2018);
Spiran_2018 = datetime(Spiran_2018,'InputFormat','dd-MMM-yyyy HH:mm:ss',
'Format','dd-MMM HH:mm:ss');
Spiran_2018_Oc = Spiran.occupancy(Spiran.Timestamp.Year==2018);

Spiran_2018_2 =
Spiran_cleaned.Timestamp(Spiran_cleaned.Timestamp.Year==2018);
Spiran_2018_2 = datetime(Spiran_2018_2,'InputFormat','dd-MMM-yyyy
HH:mm:ss', 'Format','dd-MMM HH:mm:ss');
Spiran_2018_Oc2 =
Spiran_cleaned.occupancy(Spiran_cleaned.Timestamp.Year==2018);

% Spiran - 2019
Spiran_2019 = Spiran.Timestamp(Spiran.Timestamp.Year==2019);
Spiran_2019 = Spiran_2019 - calyears(1) + days(1);
Spiran_2019 = datetime(Spiran_2019,'InputFormat','dd-MMM-yyyy HH:mm:ss',
'Format','dd-MMM HH:mm:ss');
Spiran_2019_Oc = Spiran.occupancy(Spiran.Timestamp.Year==2019);

Spiran_2019_2 =
Spiran_cleaned.Timestamp(Spiran_cleaned.Timestamp.Year==2019);
Spiran_2019_2 = Spiran_2019_2 - calyears(1) + days(1);
Spiran_2019_2 = datetime(Spiran_2019_2,'InputFormat','dd-MMM-yyyy
HH:mm:ss', 'Format','dd-MMM HH:mm:ss');
Spiran_2019_Oc2 =
Spiran_cleaned.occupancy(Spiran_cleaned.Timestamp.Year==2019);

% Spiran - 2020
Spiran_2020 = Spiran.Timestamp(Spiran.Timestamp.Year==2020);
Spiran_2020 = Spiran_2020 - calyears(2) + days(3);
Spiran_2020 = datetime(Spiran_2020,'InputFormat','dd-MMM-yyyy HH:mm:ss',
'Format','dd-MMM HH:mm:ss');
Spiran_2020_Oc = Spiran.occupancy(Spiran.Timestamp.Year==2020);

Spiran_2020_2 =
Spiran_cleaned.Timestamp(Spiran_cleaned.Timestamp.Year==2020);
Spiran_2020_2 = Spiran_2020_2 - calyears(2) + days(3);
Spiran_2020_2 = datetime(Spiran_2020_2,'InputFormat','dd-MMM-yyyy
HH:mm:ss', 'Format','dd-MMM HH:mm:ss');
Spiran_2020_Oc2 =
Spiran_cleaned.occupancy(Spiran_cleaned.Timestamp.Year==2020);

% Vardtornet - 2018
Vardtornet_2018 = Vardtornet.Timestamp(Vardtornet.Timestamp.Year==2018);
Vardtornet_2018 = datetime(Vardtornet_2018,'InputFormat','dd-MMM-yyyy
HH:mm:ss', 'Format','dd-MMM HH:mm:ss');
Vardtornet_2018_Oc = Vardtornet.occupancy(Vardtornet.Timestamp.Year==2018);
```

```matlab
Vardtornet_2018_2 =
Vardtornet_cleaned.Timestamp(Vardtornet_cleaned.Timestamp.Year==2018);
Vardtornet_2018_2 = datetime(Vardtornet_2018_2,'InputFormat','dd-MMM-yyyy
HH:mm:ss', 'Format','dd-MMM HH:mm:ss');
Vardtornet_2018_Oc2 =
Vardtornet_cleaned.occupancy(Vardtornet_cleaned.Timestamp.Year==2018);

% Vardtornet - 2019
Vardtornet_2019 = Vardtornet.Timestamp(Vardtornet.Timestamp.Year==2019);
Vardtornet_2019 = Vardtornet_2019 - calyears(1) + days(1);
Vardtornet_2019 = datetime(Vardtornet_2019,'InputFormat','dd-MMM-yyyy
HH:mm:ss', 'Format','dd-MMM HH:mm:ss');
Vardtornet_2019_Oc = Vardtornet.occupancy(Vardtornet.Timestamp.Year==2019);

Vardtornet_2019_2 =
Vardtornet_cleaned.Timestamp(Vardtornet_cleaned.Timestamp.Year==2019);
Vardtornet_2019_2 = Vardtornet_2019_2 - calyears(1) + days(1);
Vardtornet_2019_2 = datetime(Vardtornet_2019_2,'InputFormat','dd-MMM-yyyy
HH:mm:ss', 'Format','dd-MMM HH:mm:ss');
Vardtornet_2019_Oc2 =
Vardtornet_cleaned.occupancy(Vardtornet_cleaned.Timestamp.Year==2019);

% Vardtornet - 2020
Vardtornet_2020 = Vardtornet.Timestamp(Vardtornet.Timestamp.Year==2020);
Vardtornet_2020 = Vardtornet_2020 - calyears(2) + days(3);
Vardtornet_2020 = datetime(Vardtornet_2020,'InputFormat','dd-MMM-yyyy
HH:mm:ss', 'Format','dd-MMM HH:mm:ss');
Vardtornet_2020_Oc = Vardtornet.occupancy(Vardtornet.Timestamp.Year==2020);

Vardtornet_2020_2 =
Vardtornet_cleaned.Timestamp(Vardtornet_cleaned.Timestamp.Year==2020);
Vardtornet_2020_2 = Vardtornet_2020_2 - calyears(2) + days(3);
Vardtornet_2020_2 = datetime(Vardtornet_2020_2,'InputFormat','dd-MMM-yyyy
HH:mm:ss', 'Format','dd-MMM HH:mm:ss');
Vardtornet_2020_Oc2 =
Vardtornet_cleaned.occupancy(Vardtornet_cleaned.Timestamp.Year==2020);


%% Figure

figure(1)
subplot(2,1,1)
plot(Spiran_2018,Spiran_2018_Oc,Spiran_2019,Spiran_2019_Oc,Spiran_2020,Spir
an_2020_Oc)
legend('2018','2019','2020','FontSize',15)
title('Spiran - Original data','FontSize',20)
xlabel('time [Month, Day, Time]','FontSize',15)
ylabel('occupancy [%]','FontSize',15)

subplot(2,1,2)
plot(Spiran_2018_2,Spiran_2018_Oc2,Spiran_2019_2,Spiran_2019_Oc2,Spiran_202
0_2,Spiran_2020_Oc2)
legend('2018','2019','2020','FontSize',15)
title('Spiran - Cleaned data','FontSize',20)
xlabel('time [Month, Day, Time]','FontSize',15)
ylabel('occupancy [%]','FontSize',15)

figure(2)
```

```
subplot(2,1,1)
plot(Vardtornet_2018,Vardtornet_2018_Oc,Vardtornet_2019,Vardtornet_2019_Oc,
Vardtornet_2020,Vardtornet_2020_Oc)
legend('2018','2019','2020','FontSize',15)
title('Vardtornet - Original data','FontSize',20)
xlabel('time [Month, Day, Time]','FontSize',15)
ylabel('occupancy [%]','FontSize',15)

subplot(2,1,2)
plot(Vardtornet_2018_2,Vardtornet_2018_Oc2,Vardtornet_2019_2,Vardtornet_201
9_Oc2,Vardtornet_2020_2,Vardtornet_2020_Oc2)
legend('2018','2019','2020','FontSize',15)
title('Vardtornet - Cleaned data','FontSize',20)
xlabel('time [Month, Day, Time]','FontSize',15)
ylabel('occupancy [%]','FontSize',15)

end
```

Norrköping_days

```
function PR_days(data,PublicHolidays,SummerHolidays,directory)

%% Round time to the nearest 15th minute
% set new format of timestamp data - withou seconds
data.Timestamp = datetime(data.Timestamp,'InputFormat', 'dd-MMM-yyyy
HH:mm:ss','Format', 'dd-MMM-yyyy HH:mm');
% round timestamp to 5 minute intervals
data.Timestamp.Minute = 15 * round(data.Timestamp.Minute/15);

%% Set timestamp do dates
% set new format of timestamp data - days withou hours and minutes
data.Timestamp2 = datetime(data.Timestamp,'InputFormat', 'dd-MMM-yyyy
HH:mm','Format', 'dd-MMM-yyyy');
% round timestamp days
data.Timestamp2 = dateshift(data.Timestamp2, 'start', 'day');

%% Set weekdays

DayNames=[];
% loop below creates new variable with a name of the day by the function
% weekday
for i = 1:height(data)
    [DayNumber, DayName] = weekday(data.Timestamp2(i));
    DayNames = [DayNames; convertCharsToStrings(DayName)];
end
data.days=DayNames; % add variable to the table

%% Set summer holidays
% set code SH (Summer Holidays) to the exact days

for i = 1:2:length(SummerHolidays)

data.days(isbetween(data.Timestamp2,SummerHolidays(i),SummerHolidays(i+1)))
=
data.days(isbetween(data.Timestamp2,SummerHolidays(i),SummerHolidays(i+1)))
+'SH';
end

%% Set public holidays CZ
```

```
% loop below sets code PH (Public Holidays) to the exact days set above
for i = 1:length(PublicHolidays)

data.days(isbetween(data.Timestamp2,PublicHolidays(i),PublicHolidays(i))) =
data.days(isbetween(data.Timestamp2,PublicHolidays(i),PublicHolidays(i)))+'
PH';
end


%% Calculated days

% start is the first whole day in 2018 and end is before the pandemic in
2020
data2 = data(isbetween(data.Timestamp2, datetime('01-Jan-2018'),
datetime('29-Feb-2020')),:);
% writedown unique days
Unique_Days = unique(data2.days);

%% Set time of day
% set the beginning and ending of the day
start_of_day = datetime('00:00','Format', 'HH:mm');
end_of_day = datetime('23:55','Format', 'HH:mm');
% day time in 5 minute intervals
daytime = (start_of_day:minutes(15):end_of_day)';

%% Creates CSV files for each unique day
% change type of directory of the export (from char to string)
directory = convertCharsToStrings(directory);

for i = 1:length(Unique_Days)

    the_weekday_all = []; % cleaned table after round of calculation
    the_day = []; % cleaned table after round of calculation

    % creates table of timestamp, day and number of free places for the
    % i-th unique day (such as Friday)
    the_weekday_all.Timestamp =
data2.Timestamp(data2.days==Unique_Days(i));
    the_weekday_all.Timestamp2 =
data2.Timestamp2(data2.days==Unique_Days(i));
    the_weekday_all.num_of_free_places =
data2.num_of_free_places(data2.days==Unique_Days(i));
    % change structure to table
    the_weekday_all=struct2table(the_weekday_all);
    % sets unique dats for the unique day (such as Friday 18-May-2018)
    Unique_Dates = unique(the_weekday_all.Timestamp2);

    % loop below provides the whole computation - the final is a table for
    % each unique day
    for j = 1:length(Unique_Dates)

        the_weekday = []; % cleaned table after round of calculation
        % creates table of time of day and number of free places for each
        % unique day in unique date
        the_weekday.time_of_day =
the_weekday_all.Timestamp(the_weekday_all.Timestamp2==Unique_Dates(j));
```

```
        the_weekday.num_of_free_places =
the_weekday_all.num_of_free_places(the_weekday_all.Timestamp2==Unique_Dates
(j));
        % change structure to table and format of the timestamp
        the_weekday=struct2table(the_weekday);

the_weekday.time_of_day=datetime(the_weekday.time_of_day,'Format','HH:mm');

        % it is used for the computation purposes - the data are not
        % complete for all days
        l = 1;

        for k = 1:length(daytime) % 5 minute interval

            if (the_weekday.time_of_day(l) - the_weekday.time_of_day(end))
== minutes(0)
                % if the data ends before the end of the day takes previous
                % value
                the_day(k,j) = the_weekday.num_of_free_places(l-1);

            elseif the_weekday.time_of_day(l).Hour == daytime(k).Hour &&
the_weekday.time_of_day(l).Minute == daytime(k).Minute
                % if data is correct it records the value
                the_day(k,j) = the_weekday.num_of_free_places(l);
                l=l+1; % move in the weekday table - the data are not
complete for all days

            elseif the_weekday.time_of_day(l).Hour == daytime(2).Hour &&
the_weekday.time_of_day(l).Minute == daytime(2).Minute
                % if the data records start later than day it takes
                % following value
                the_day(k,j) = the_weekday.num_of_free_places(l+1);

            elseif l-1 == 0
                % if there are no previsou records - day starts at 4 a.m.
                the_day(k,j) = 0;

            else
                % in the case - there are no data record it takes average
                % of previous and following data record
                the_day(k,j) = round((the_weekday.num_of_free_places(l-
1)+the_weekday.num_of_free_places(l+1))/2);

            end
        end
    end
    the_day = array2table(the_day); % change structure to table
    % save table as a CSV file
    % colums are dates and rows is daytime
    day_name = sprintf('%s.csv',Unique_Days(i));
    writetable(the_day,directory+day_name,'Delimiter',',');
end

end
```

Spiran_days_figure

```
function PR_L_days_figure(directory)

%% Read csv files

csvfile=string('*.csv');

fileName = dir(directory+csvfile);
fileName_cell=struct2cell(fileName);

figure(1)
for i = 1:length(fileName_cell)
   subplot(6,4,i)
   a = table2array(readtable(directory+string(fileName_cell(1,i))));
   b = size(a);
   if b(2) == 1
       a(:,2) = a(:,1);
       mesh(a);
       title(sprintf('1day-%s',string(fileName_cell(1,i))),'FontSize',15);
       xlabel({'Week'; '[01-2018 to 02-2020]'},'FontSize',10);
       ylabel({'15-min interval'; '[00:00 to 24:55]'},'FontSize',10);
       zlabel({'Free parking places'; '[0 - 388]'},'FontSize',10);
   else
       mesh(a);
       title(sprintf('%s',string(fileName_cell(1,i))),'FontSize',15);
       xlabel({'Week'; '[01-2018 to 02-2020]'},'FontSize',10);
       ylabel({'15-min interval'; '[00:00 to 23:45]'},'FontSize',10);
       zlabel({'Free parking places'; '[0 - 388]'},'FontSize',10);
   end
end

end
```

Vårdtornet_days_figure

```
function PR_L_days_figure(directory)

%% Read csv files

csvfile=string('*.csv');

fileName = dir(directory+csvfile);
fileName_cell=struct2cell(fileName);

figure(1)
for i = 1:length(fileName_cell)
   subplot(6,4,i)
   a = table2array(readtable(directory+string(fileName_cell(1,i))));
   b = size(a);
   if b(2) == 1
       a(:,2) = a(:,1);
       mesh(a);
       title(sprintf('1day-%s',string(fileName_cell(1,i))),'FontSize',15);
       xlabel({'Week'; '[09-2018 to 02-2020]'},'FontSize',10);
       ylabel({'15-min interval'; '[00:00 to 24:55]'},'FontSize',10);
       zlabel({'Free parking places'; '[0 - 84]'},'FontSize',10);
   else
       mesh(a);
```

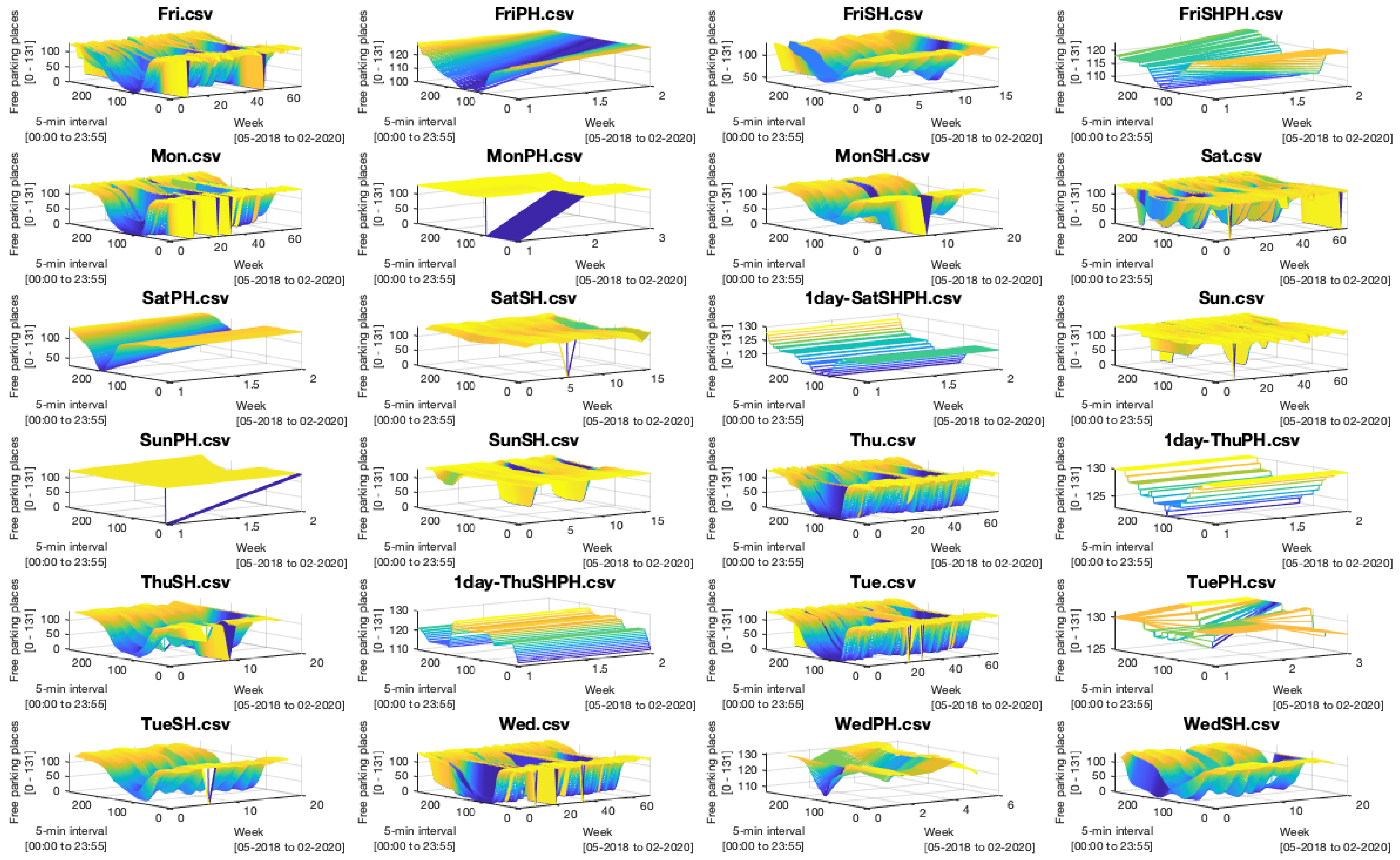
```
        title(sprintf('%s',string(fileName_cell(1,i))),'FontSize',15);
        xlabel({'Week'; '[09-2018 to 02-2020]'},'FontSize',10);
        ylabel({'15-min interval'; '[00:00 to 23:45]'},'FontSize',10);
        zlabel({'Free parking places'; '[0 - 84]'},'FontSize',10);
    end
end

end
```

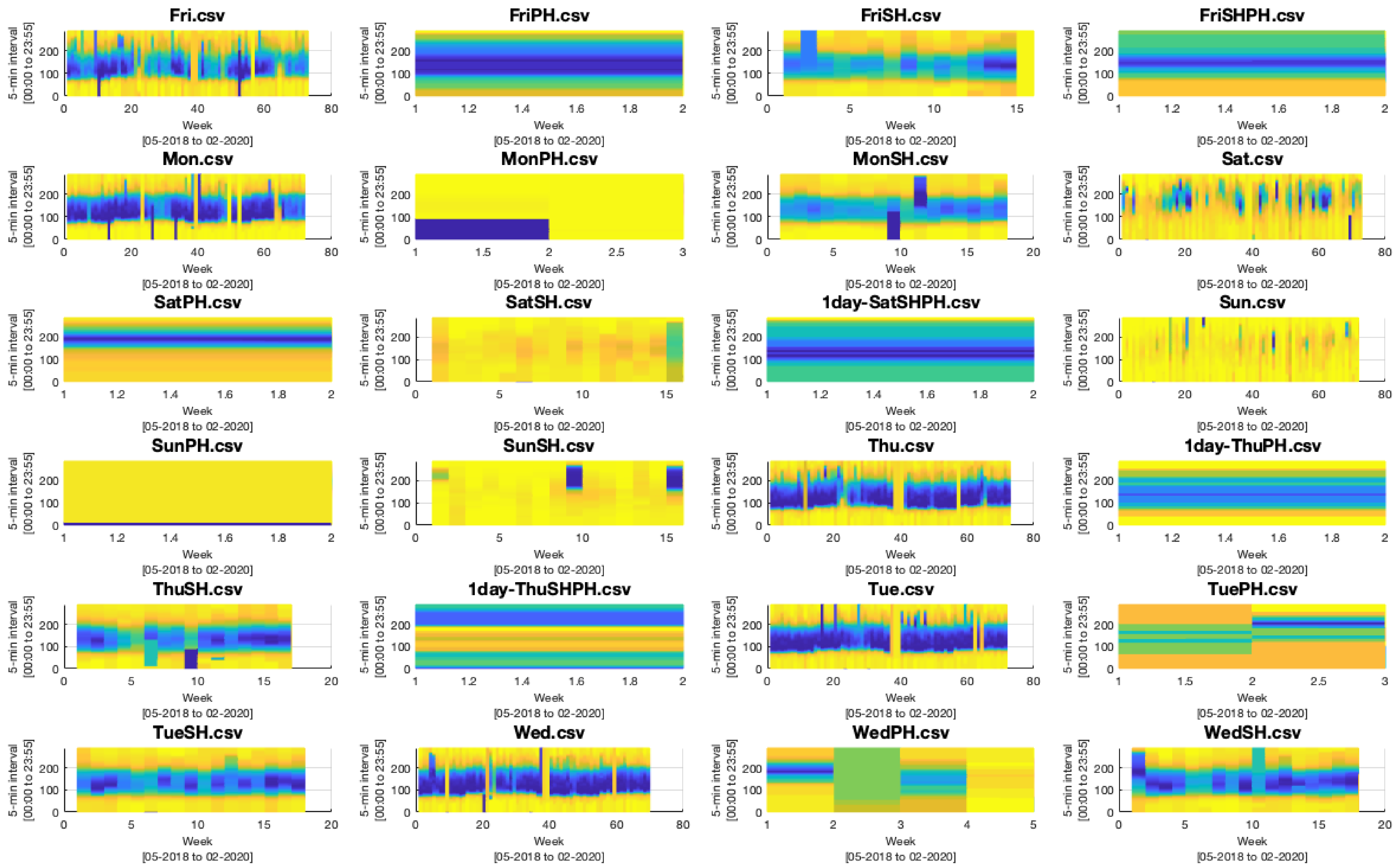
Annex D

P+R Prague

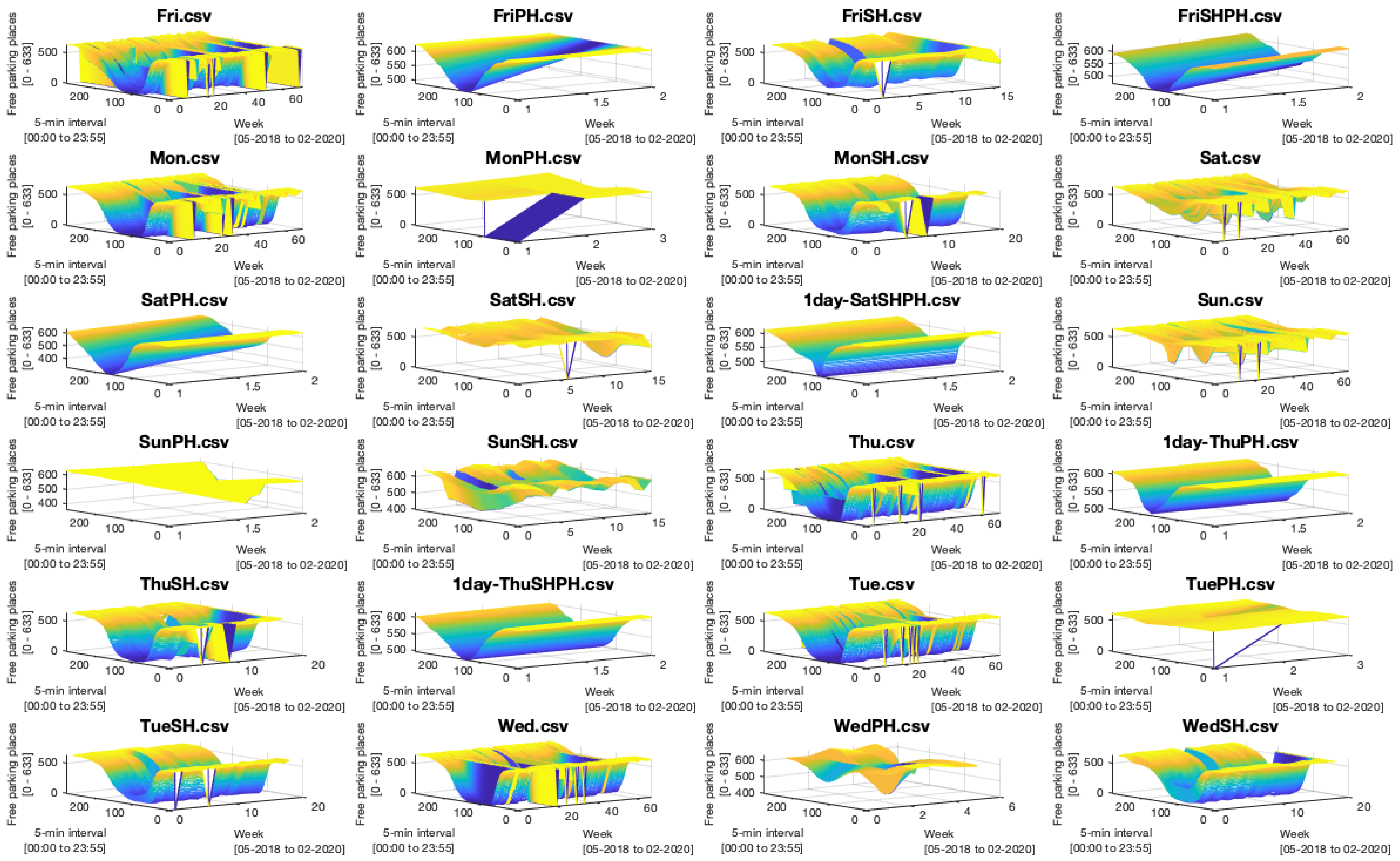
P+R Černý most 2 3D



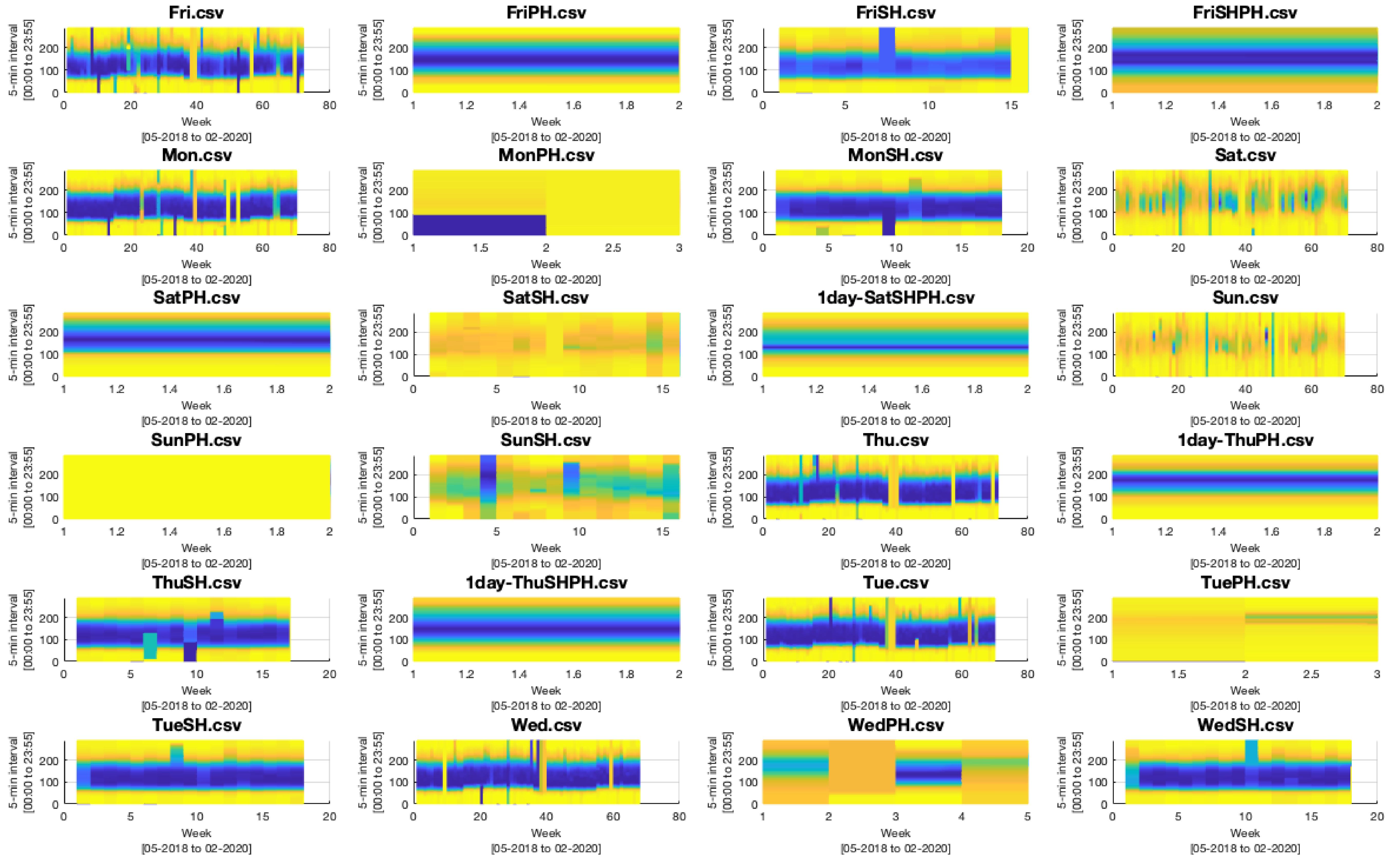
P+R Černý most 2 2D



P+R Letňany 3D

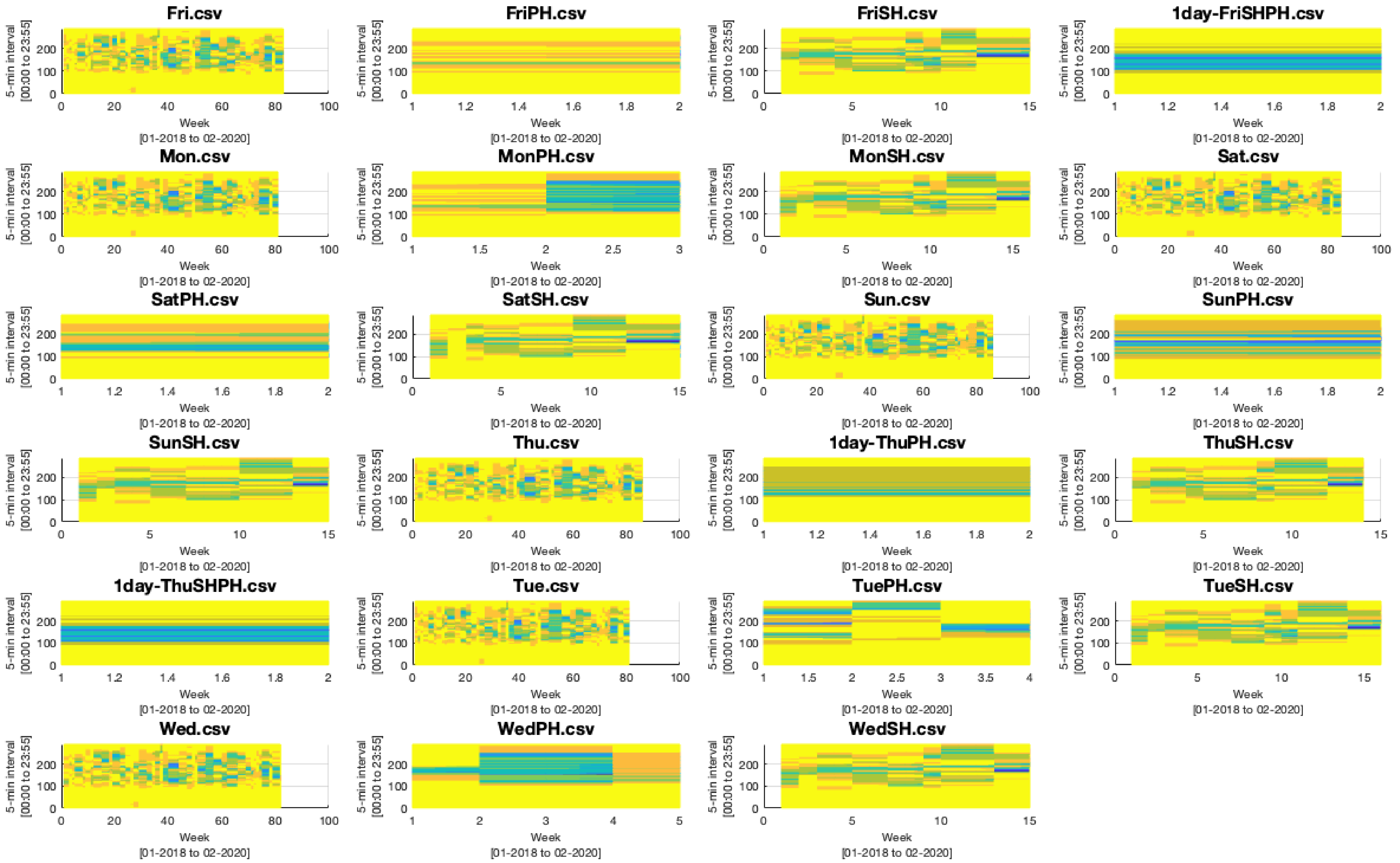


P+R Letňany 2D

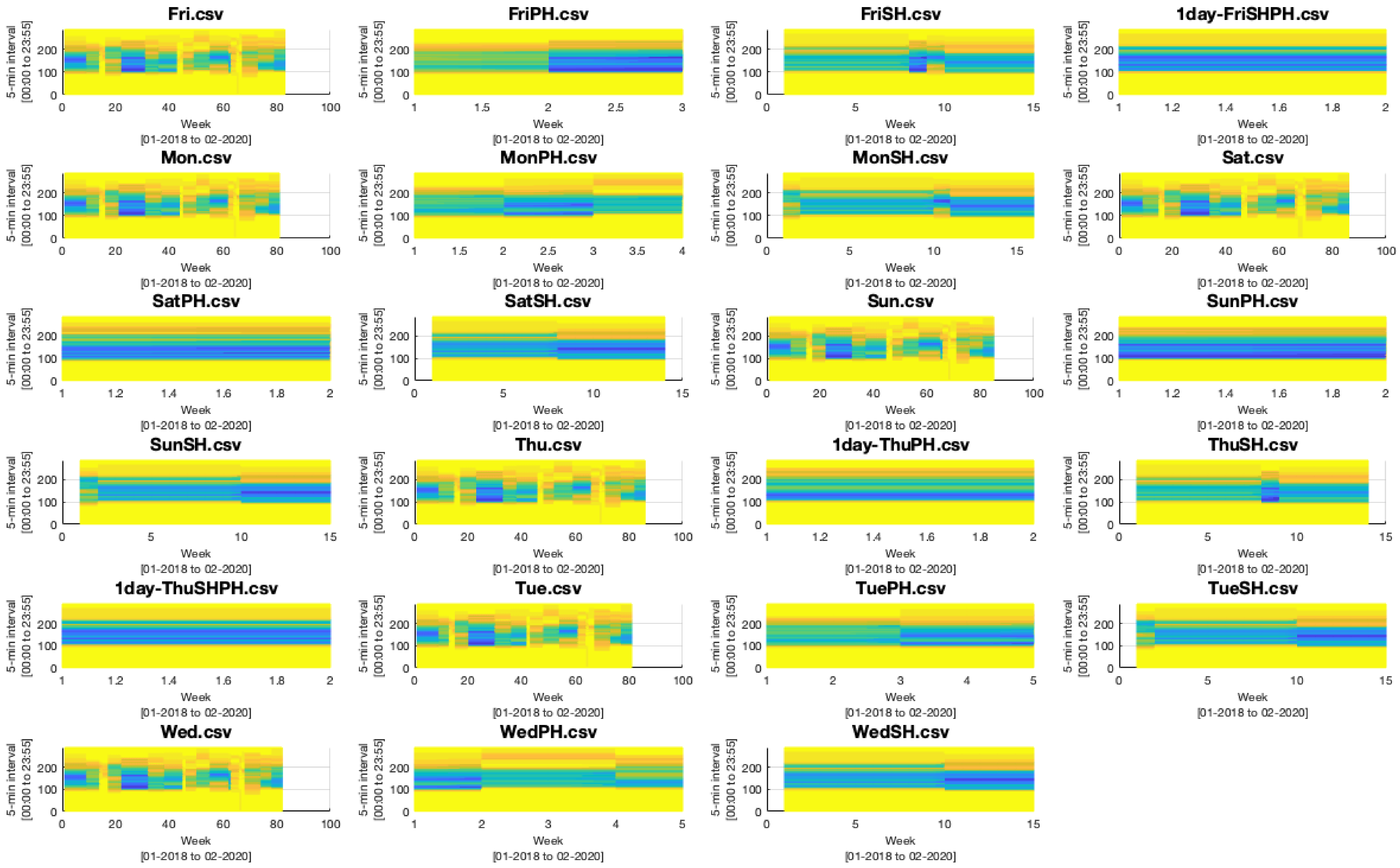


ZPS Prague

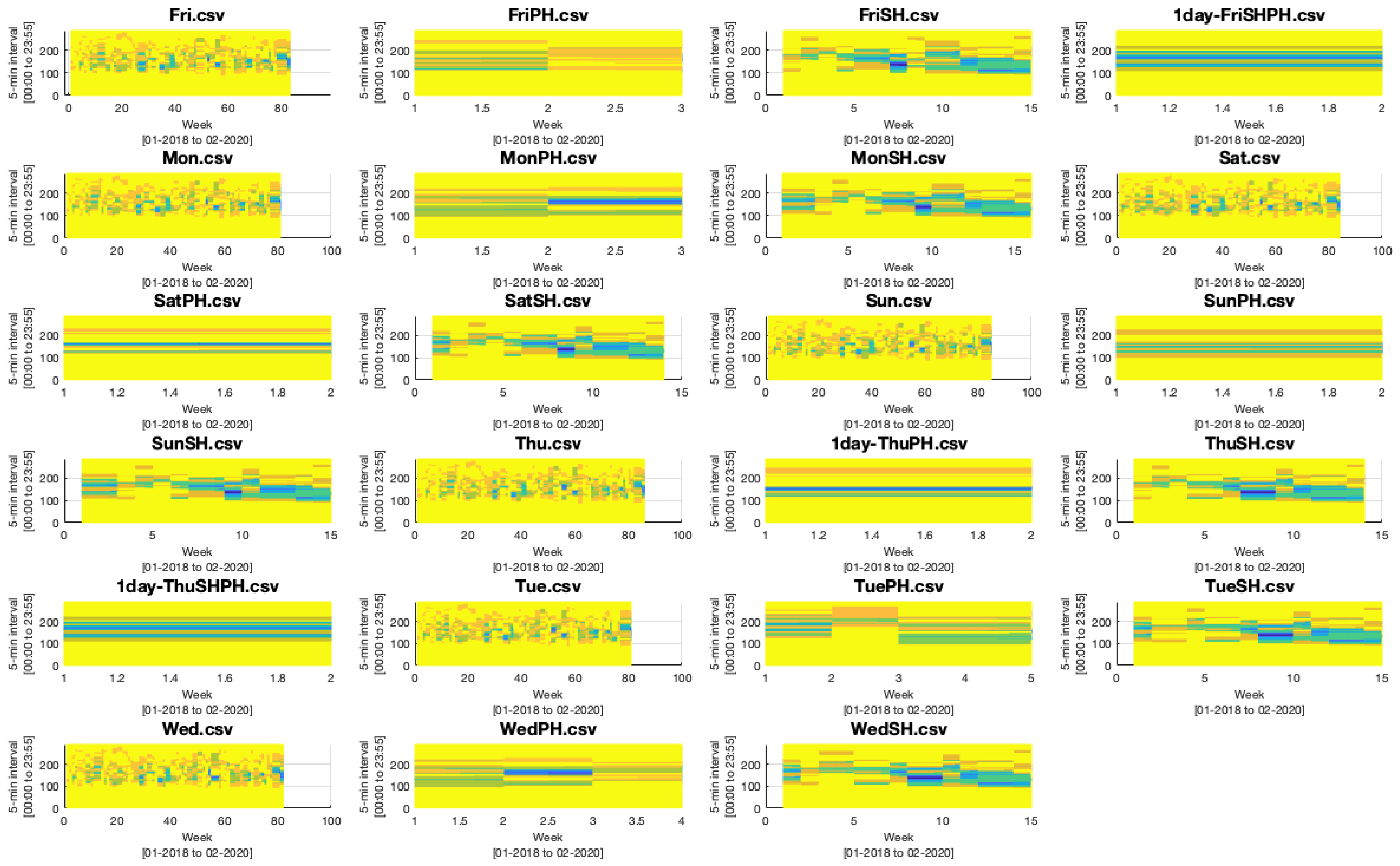
ZPS Prague – P7-0158



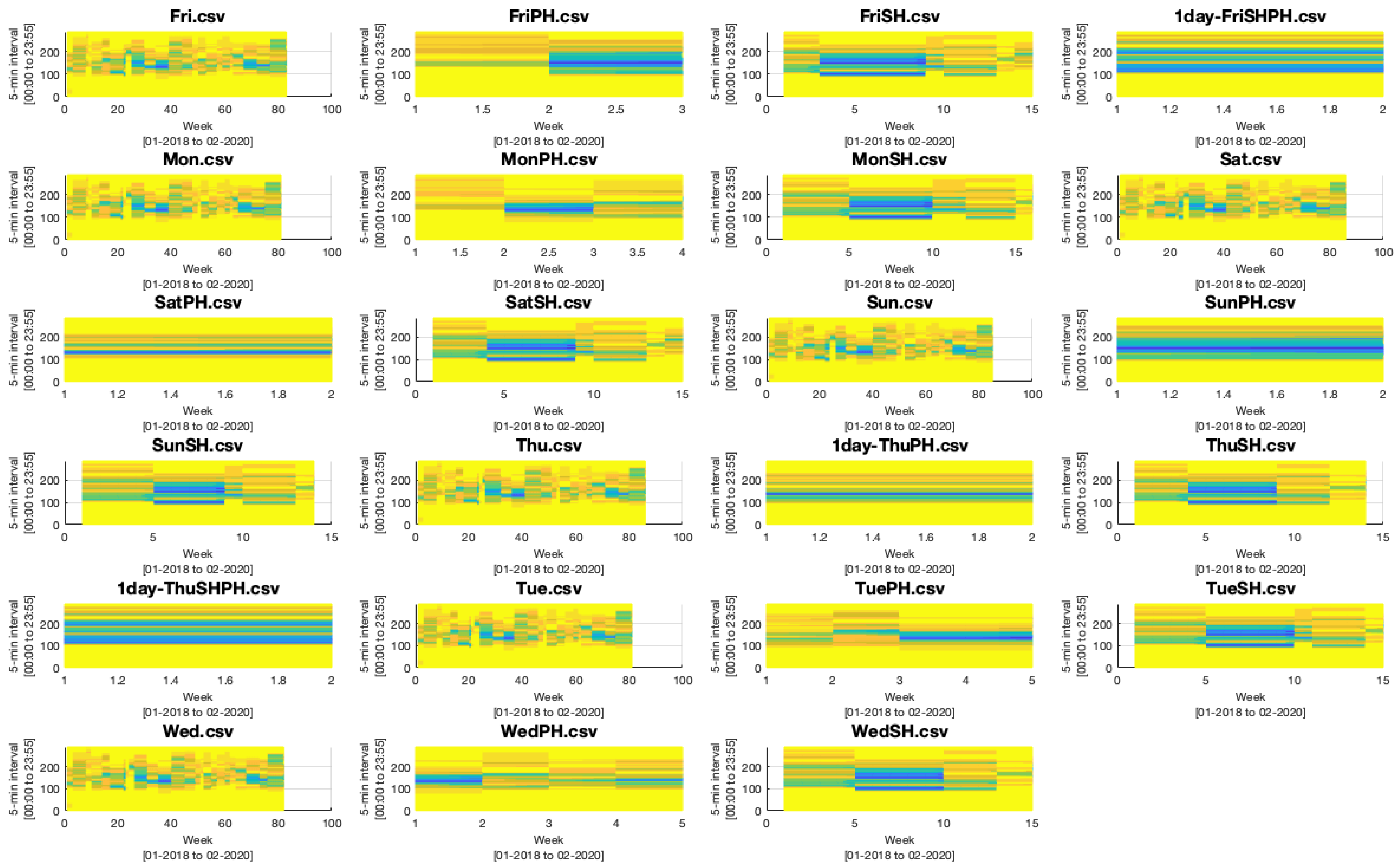
ZPS Prague – P7-0134



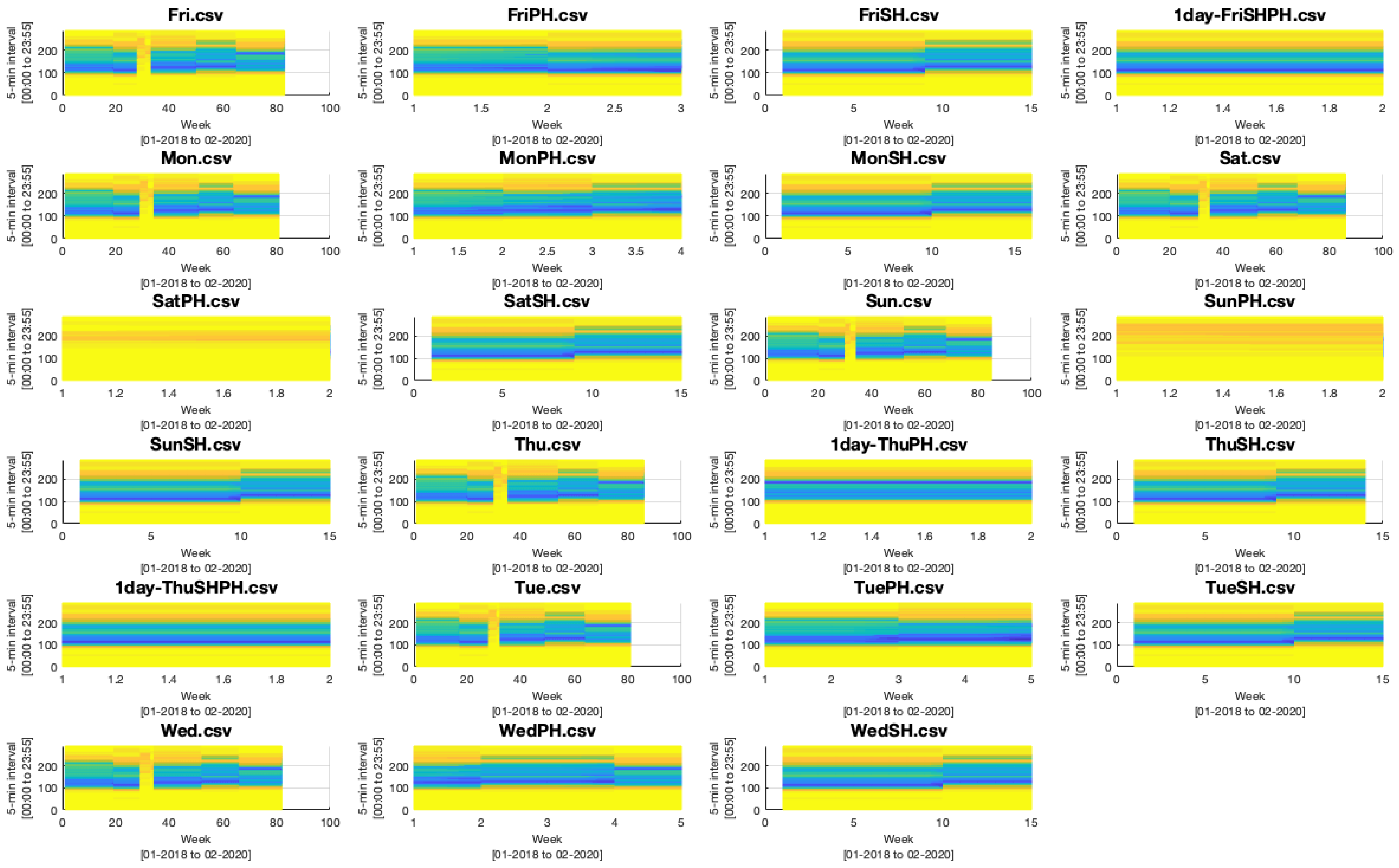
ZPS Prague – P7-0156



ZPS Prague – P7-0161

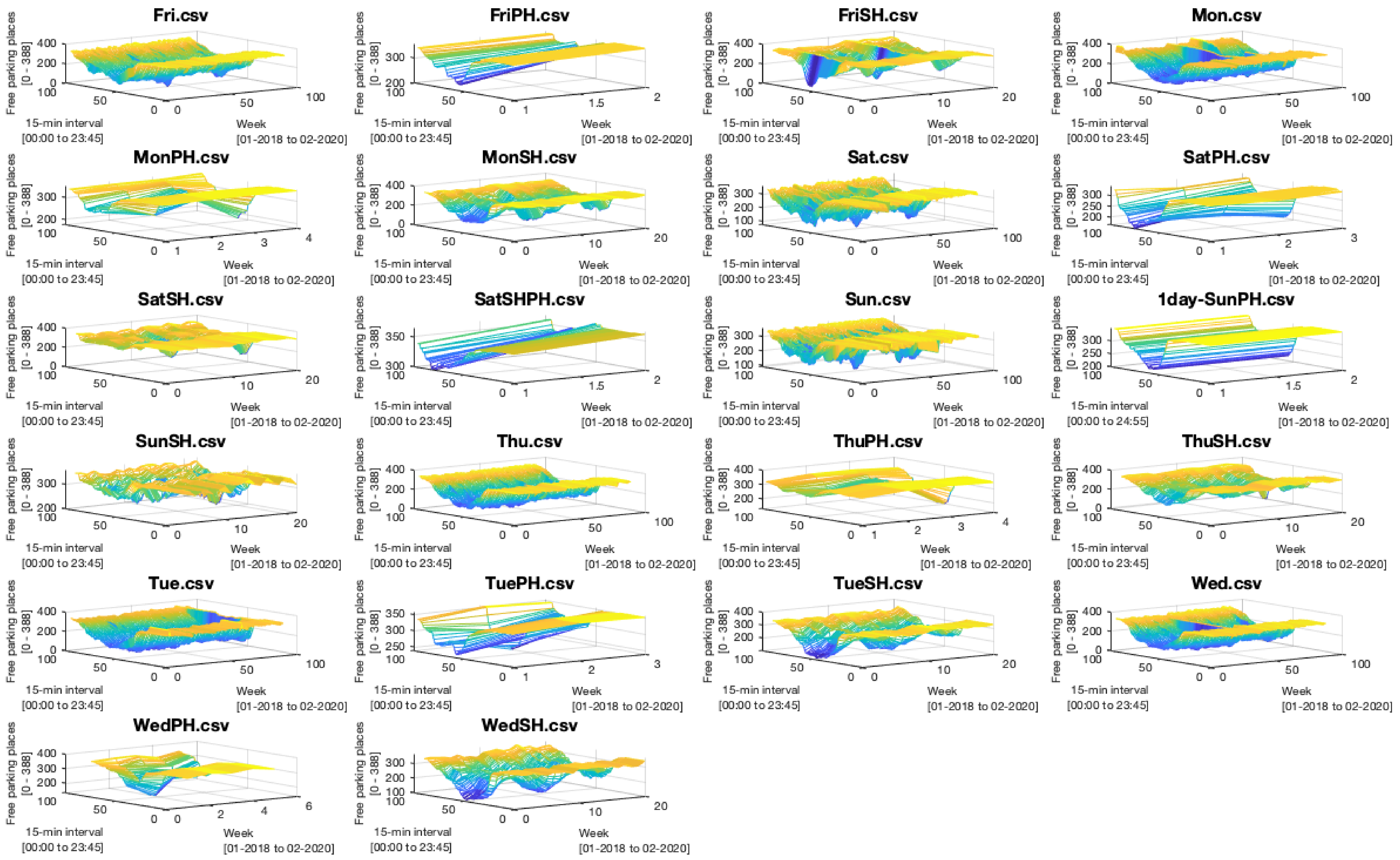


ZPS Prague – P7-0179

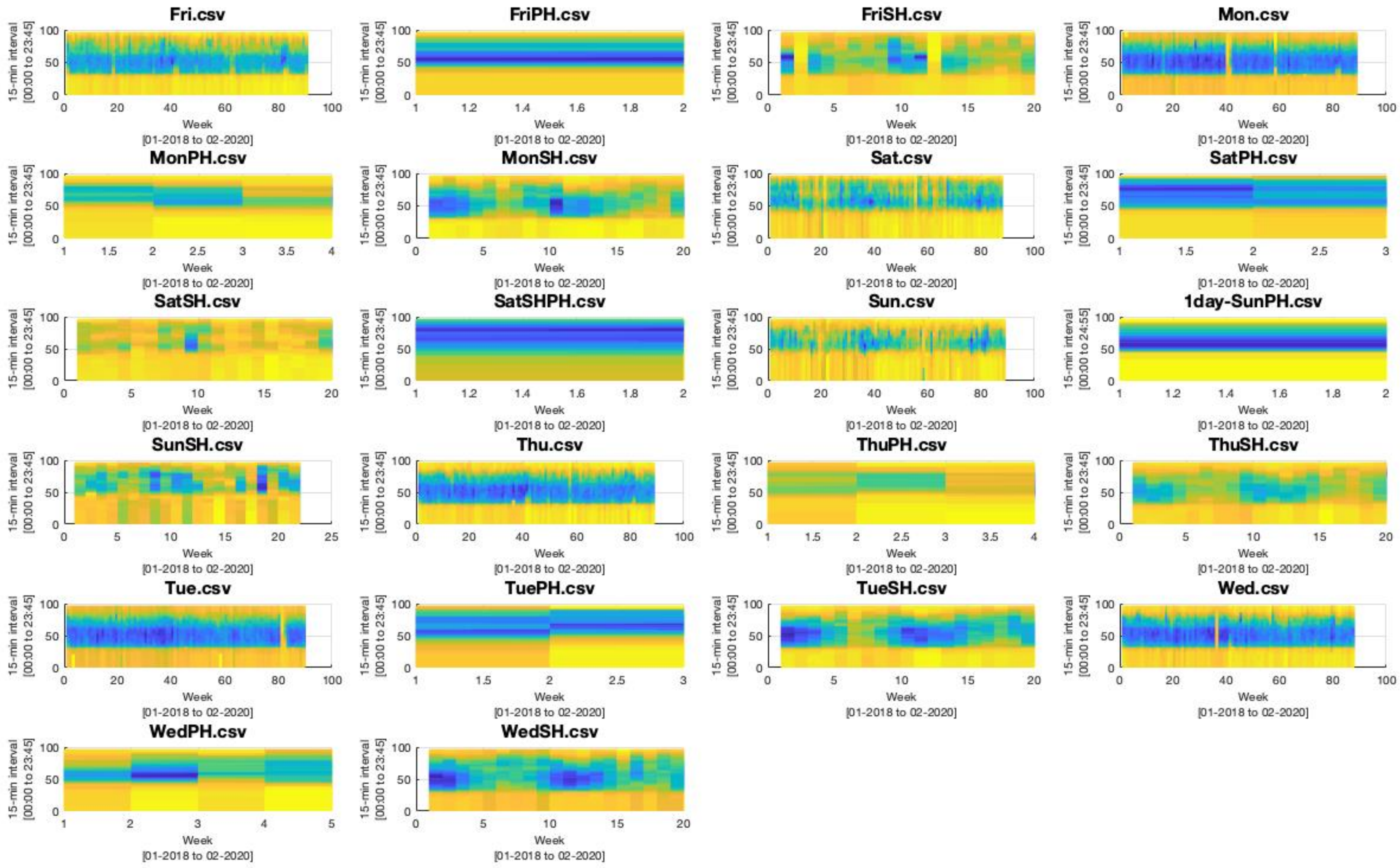


Parking garages Norrköping

Spiran 3D

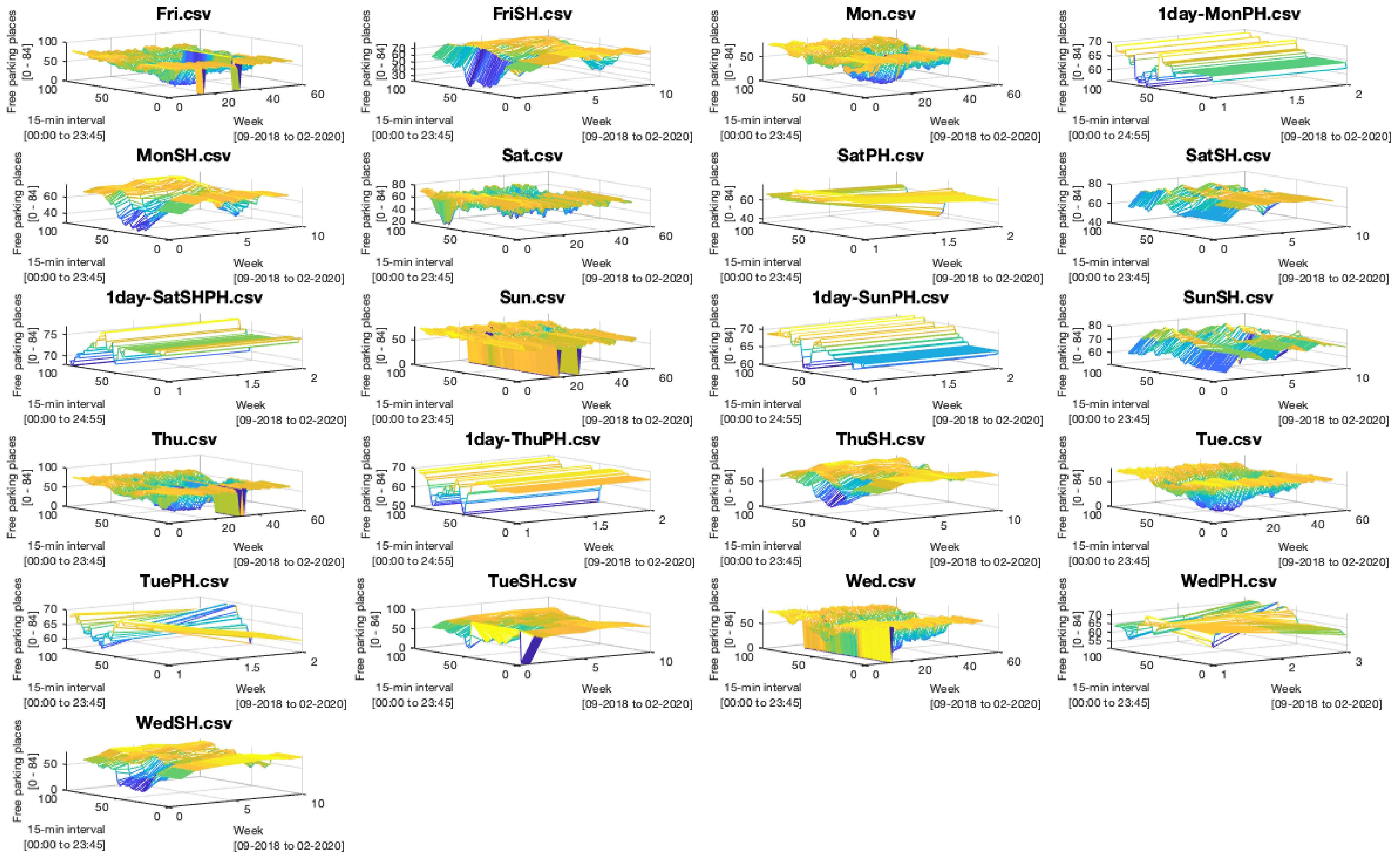


Spiran 2D

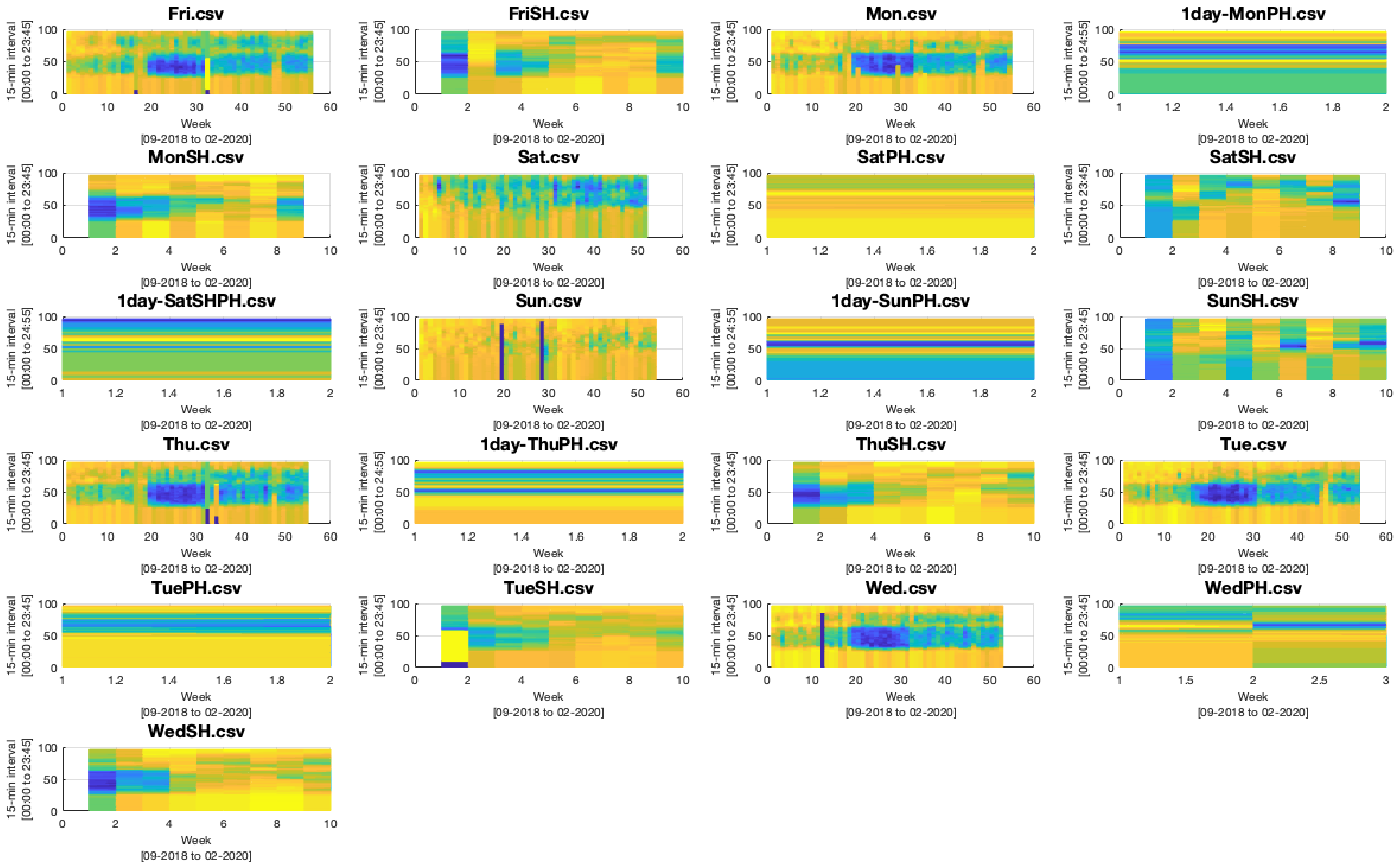


Fri.csv
FriPH.csv
FriSH.csv
Mon.csv
MonPH.csv
MonSH.csv
Sat.csv
SatPH.csv
SatSH.csv
SatSHPH.csv
Sun.csv
1day-SunPH.csv
SunSH.csv
Thu.csv
ThuPH.csv
ThuSH.csv
Tue.csv
TuePH.csv
TueSH.csv
Wed.csv
WedPH.csv
WedSH.csv
15-min interval [00:00 to 23:45]
15-min interval [00:00 to 24:55]
Week [01-2018 to 02-2020]

Vårdtornet 3D



Vårdtornet 2D



Annex E

The main code

```
%% Initialization

clear
close all

%% Load PR_CM2

directoryCM2 = '/Users/viktorbenes/diplomka/Prague P+R/days_PR_CM2/';

[Fri, FriSH, Mon, MonSH, Sat, SatSH, Sun, SunSH, workday, workdaySH,
publicholidays] = loadPR(directoryCM2);

%% Save wordays, workdaysSH and publicholidays to CSV files

workday_export = string('export/workday.csv');
writetable(workday,directoryCM2+workday_export,'Delimiter',',');

workday_exportSH = string('export/workdaySH.csv');
writetable(workdaySH,directoryCM2+workday_exportSH,'Delimiter',',');

publicholidays_export = string('export/publicholidays.csv');
writetable(publicholidays,directoryCM2+publicholidays_export,'Delimiter',',
');

%% Files for prediction method comparison

directory_TEST_CM2 =
'/Users/viktorbenes/diplomka/general_code/test_PR_CM2/';
directory_TEST_CM2 = convertCharsToStrings(directory_TEST_CM2);

MonTRAIN = string('MonTRAIN');
writetable(Mon(:,1:(end-1)),directory_TEST_CM2+MonTRAIN,'Delimiter',',');
MonTEST = string('MonTEST');
writetable(Mon(:,end),directory_TEST_CM2+MonTEST,'Delimiter',',');

workdayTRAIN = string('workdayTRAIN');
writetable(workday(:,1:(end-
1)),directory_TEST_CM2+workdayTRAIN,'Delimiter',',');
workdayTEST = string('workdayTEST');
writetable(workday(:,end),directory_TEST_CM2+workdayTEST,'Delimiter',',');

SatSHTRAIN = string('SatSHTRAIN');
writetable(SatSH(:,1:(end-
1)),directory_TEST_CM2+SatSHTRAIN,'Delimiter',',');
SatSHTEST = string('SatSHTEST');
writetable(SatSH(:,end),directory_TEST_CM2+SatSHTEST,'Delimiter',',');


%% Initialization

clear
close all
```

```
%% Load PR_L

directoryL = '/Users/viktorbenes/diplomka/Prague P+R/days_PR_L/';

[Fri, FriSH, Mon, MonSH, Sat, SatSH, Sun, SunSH, workday, workdaySH,
publicholidays] = loadPR(directoryL);

%% Save wordays, workdaysSH and publicholidays to CSV files

workday_export = string('export/workday.csv');
writetable(workday,directoryL+workday_export,'Delimiter',',');

workday_exportSH = string('export/workdaySH.csv');
writetable(workdaySH,directoryL+workday_exportSH,'Delimiter',',');

publicholidays_export = string('export/publicholidays.csv');
writetable(publicholidays,directoryL+publicholidays_export,'Delimiter',',')
;

%% Initialization

clear
close all

%% Load Spiran

directorySpiran = '/Users/viktorbenes/diplomka/Norrkoping
Parking/Spiran_days/';

[Fri, FriSH, Mon, MonSH, Sat, SatSH, Sun, SunSH, workday, workdaySH,
publicholidays] = loadPR(directorySpiran);

%% Save wordays, workdaysSH and publicholidays to CSV files

workday_export = string('export/workday.csv');
writetable(workday,directorySpiran+workday_export,'Delimiter',',');

workday_exportSH = string('export/workdaySH.csv');
writetable(workdaySH,directorySpiran+workday_exportSH,'Delimiter',',');

publicholidays_export = string('export/publicholidays.csv');
writetable(publicholidays,directorySpiran+publicholidays_export,'Delimiter'
,',');

%% Files for prediction method comparison

directory_TEST_Spiran =
'/Users/viktorbenes/diplomka/general_code/test_Spiran/';
directory_TEST_Spiran = convertCharsToStrings(directory_TEST_Spiran);

MonTRAIN = string('MonTRAIN');
writetable(Mon(:,1:(end-
1)),directory_TEST_Spiran+MonTRAIN,'Delimiter',',');
MonTEST = string('MonTEST');
writetable(Mon(:,end),directory_TEST_Spiran+MonTEST,'Delimiter',',');
```

```
workdayTRAIN = string('workdayTRAIN');
writetable(workday(:,1:(end-
1)),directory_TEST_Spiran+workdayTRAIN,'Delimiter',',');
workdayTEST = string('workdayTEST');
writetable(workday(:,end),directory_TEST_Spiran+workdayTEST,'Delimiter',','
);

SatSHTRAIN = string('SatSHTRAIN');
writetable(SatSH(:,1:(end-
1)),directory_TEST_Spiran+SatSHTRAIN,'Delimiter',',');
SatSHTEST = string('SatSHTEST');
writetable(SatSH(:,end),directory_TEST_Spiran+SatSHTEST,'Delimiter',',');

%% Initialization

clear
close all

%% Load Vardtornet

directoryV = '/Users/viktorbenes/diplomka/Norrkoping
Parking/Vardtornet_days/';

[Fri, FriSH, Mon, MonSH, Sat, SatSH, Sun, SunSH, workday, workdaySH,
publicholidays] = loadPR(directoryV);

%% Save wordays, workdaysSH and publicholidays to CSV files

workday_export = string('export/workday.csv');
writetable(workday,directoryV+workday_export,'Delimiter',',');

workday_exportSH = string('export/workdaySH.csv');
writetable(workdaySH,directoryV+workday_exportSH,'Delimiter',',');

publicholidays_export = string('export/publicholidays.csv');
writetable(publicholidays,directoryV+publicholidays_export,'Delimiter',',')
;

%% Initialization

clear
close all

%% Load ZPS 0158

directoryZPS0158 = '/Users/viktorbenes/diplomka/Prague ZPS/P7-0158/';
[Fri, FriSH, Mon, MonSH, Sat, SatSH, Sun, SunSH, Tue, TueSH, Wed, WedSH,
Thu, ThuSH, publicholidays] = loadZPS(directoryZPS0158);

%% Save publicholidays to CSV file

publicholidays_export = string('export/publicholidays.csv');
writetable(publicholidays,directoryZPS0158+publicholidays_export,'Delimiter
',',');

%% Files for prediction method comparison
```

```
directory_TEST_ZPS0158 =
'/Users/viktorbenes/diplomka/general_code/test_ZPS0158/';
directory_TEST_ZPS0158 = convertCharsToStrings(directory_TEST_ZPS0158);

MonTRAIN = string('MonTRAIN');
writetable(Mon(:,1:(end-
1)),directory_TEST_ZPS0158+MonTRAIN,'Delimiter',',');
MonTEST = string('MonTEST');
writetable(Mon(:,end),directory_TEST_ZPS0158+MonTEST,'Delimiter',',');

WedTRAIN = string('WedTRAIN');
writetable(Wed(:,1:(end-
1)),directory_TEST_ZPS0158+WedTRAIN,'Delimiter',',');
WedTEST = string('WedTEST');
writetable(Wed(:,end),directory_TEST_ZPS0158+WedTEST,'Delimiter',',');

SatSHTRAIN = string('SatSHTRAIN');
writetable(SatSH(:,1:(end-
1)),directory_TEST_ZPS0158+SatSHTRAIN,'Delimiter',',');
SatSHTEST = string('SatSHTEST');
writetable(SatSH(:,end),directory_TEST_ZPS0158+SatSHTEST,'Delimiter',',');

%% Initialization

clear
close all

%% Load ZPS 0134

directoryZPS0134 = '/Users/viktorbenes/diplomka/Prague ZPS/P7-0134/';
[Fri, FriSH, Mon, MonSH, Sat, SatSH, Sun, SunSH, Tue, TueSH, Wed, WedSH,
Thu, ThuSH, publicholidays] = loadZPS(directoryZPS0134);

%% Save publicholidays to CSV file

publicholidays_export = string('export/publicholidays.csv');
writetable(publicholidays,directoryZPS0134+publicholidays_export,'Delimiter
',',');

%% Initialization

clear
close all

%% Load ZPS 0156

directoryZPS0156 = '/Users/viktorbenes/diplomka/Prague ZPS/P7-0156/';
[Fri, FriSH, Mon, MonSH, Sat, SatSH, Sun, SunSH, Tue, TueSH, Wed, WedSH,
Thu, ThuSH, publicholidays] = loadZPS(directoryZPS0156);

%% Save publicholidays to CSV file

publicholidays_export = string('export/publicholidays.csv');
writetable(publicholidays,directoryZPS0156+publicholidays_export,'Delimiter
',',');

%% Initialization
```

```
clear
close all

%% Load ZPS 0161

directoryZPS0161 = '/Users/viktorbenes/diplomka/Prague ZPS/P7-0161/';
[Fri, FriSH, Mon, MonSH, Sat, SatSH, Sun, SunSH, Tue, TueSH, Wed, WedSH,
Thu, ThuSH, publicholidays] = loadZPS(directoryZPS0161);

%% Save publicholidays to CSV file

publicholidays_export = string('export/publicholidays.csv');
writetable(publicholidays,directoryZPS0161+publicholidays_export,'Delimiter
',',');

%% Initialization

clear
close all

%% Load ZPS 0179

directoryZPS0179 = '/Users/viktorbenes/diplomka/Prague ZPS/P7-0179/';
[Fri, FriSH, Mon, MonSH, Sat, SatSH, Sun, SunSH, Tue, TueSH, Wed, WedSH,
Thu, ThuSH, publicholidays] = loadZPS(directoryZPS0179);

%% Save publicholidays to CSV file

publicholidays_export = string('export/publicholidays.csv');
writetable(publicholidays,directoryZPS0179+publicholidays_export,'Delimiter
',',');
```

loadPR

```
function [Fri, FriSH, Mon, MonSH, Sat, SatSH, Sun, SunSH, workday,
workdaySH, publicholidays] = loadPR(directory)

%% Load directories

allcsv=string('*.csv');

directory = convertCharsToStrings(directory);
directoryCM2_wd = 'workday/';
directoryCM2_wd = convertCharsToStrings(directoryCM2_wd);
directoryCM2_wdSH = 'workdaySH/';
directoryCM2_wdSH = convertCharsToStrings(directoryCM2_wdSH);
directoryCM2_PH = 'PH/';
directoryCM2_PH = convertCharsToStrings(directoryCM2_PH);

%% Set table for separate days

fileName_all = dir(directory+allcsv);
fileName_cell_all = struct2cell(fileName_all);

Fri = readtable(directory+string(fileName_cell_all(1,1)));
FriSH = readtable(directory+string(fileName_cell_all(1,2)));
```

```
Mon = readtable(directory+string(fileName_cell_all(1,3)));
MonSH = readtable(directory+string(fileName_cell_all(1,4)));
Sat = readtable(directory+string(fileName_cell_all(1,5)));
SatSH = readtable(directory+string(fileName_cell_all(1,6)));
Sun = readtable(directory+string(fileName_cell_all(1,7)));
SunSH = readtable(directory+string(fileName_cell_all(1,8)));

%% Set table for Workdays days

fileName_wd = dir(directory+directoryCM2_wd+allcsv);
fileName_cell_wd = struct2cell(fileName_wd);

workday = [];
sizewd = size(fileName_cell_wd);
for k = 1:sizewd(2)
    workday=[workday
table2array(readtable(directory+directoryCM2_wd+string(fileName_cell_wd(1,k
))))];
end
workday = array2table(workday);

%% Set table for Workdays Summer holidays days

fileName_wdSH = dir(directory+directoryCM2_wdSH+allcsv);
fileName_cell_wdSH = struct2cell(fileName_wdSH);

workdaySH = [];
sizewdSH = size(fileName_cell_wdSH);
for k = 1:sizewdSH(2)
    workdaySH=[workdaySH
table2array(readtable(directory+directoryCM2_wdSH+string(fileName_cell_wdSH
(1,k))))];
end
workdaySH = array2table(workdaySH);

%% Set table for Public holidays days

fileName_PH = dir(directory+directoryCM2_PH+allcsv);
fileName_cell_PH = struct2cell(fileName_PH);

publicholidays = [];
sizePH = size(fileName_cell_PH);
for k = 1:sizePH(2)
    publicholidays=[publicholidays
table2array(readtable(directory+directoryCM2_PH+string(fileName_cell_PH(1,k
))))];
end
publicholidays = array2table(publicholidays);

end
```

loadZPS

```
function [Fri, FriSH, Mon, MonSH, Sat, SatSH, Sun, SunSH, Tue, TueSH, Wed,
WedSH, Thu, ThuSH, publicholidays] = loadZPS(directory)
```

```
%% Load directories

allcsv=string('*.csv');

directory = convertCharsToStrings(directory);
directoryCM2_wd = 'workday/';
directoryCM2_wd = convertCharsToStrings(directoryCM2_wd);
directoryCM2_wdSH = 'workdaySH/';
directoryCM2_wdSH = convertCharsToStrings(directoryCM2_wdSH);
directoryCM2_PH = 'PH/';
directoryCM2_PH = convertCharsToStrings(directoryCM2_PH);

%% Set table for separate days

fileName_all = dir(directory+allcsv);
fileName_cell_all = struct2cell(fileName_all);

Fri = readtable(directory+string(fileName_cell_all(1,1)));
FriSH = readtable(directory+string(fileName_cell_all(1,2)));
Mon = readtable(directory+string(fileName_cell_all(1,3)));
MonSH = readtable(directory+string(fileName_cell_all(1,4)));
Sat = readtable(directory+string(fileName_cell_all(1,5)));
SatSH = readtable(directory+string(fileName_cell_all(1,6)));
Sun = readtable(directory+string(fileName_cell_all(1,7)));
SunSH = readtable(directory+string(fileName_cell_all(1,8)));

Thu = readtable(directory+string(fileName_cell_all(1,9)));
ThuSH = readtable(directory+string(fileName_cell_all(1,10)));
Tue = readtable(directory+string(fileName_cell_all(1,11)));
TueSH = readtable(directory+string(fileName_cell_all(1,12)));
Wed = readtable(directory+string(fileName_cell_all(1,13)));
WedSH = readtable(directory+string(fileName_cell_all(1,14)));

%% Set table for Public holidays days

fileName_PH = dir(directory+directoryCM2_PH+allcsv);
fileName_cell_PH = struct2cell(fileName_PH);

publicholidays = [];
sizePH = size(fileName_cell_PH);
for k = 1:sizePH(2)
    publicholidays=[publicholidays
table2array(readtable(directory+directoryCM2_PH+string(fileName_cell_PH(1,k
))))];
end
publicholidays = array2table(publicholidays);

end
```

Annex F

Prediction method comparison

prediction method	**measure**	**PR_CM2**			**Spiran**			**ZPS-P7-0158**			**Overall R-squared**	**Overall MSE**
		Mon	**Workday**	**SatSH**	**Mon**	**Workday**	**SatSH**	**Mon**	**Wed**	**SatSH**		
Linear Regression - Pure Quadratic	R-squared	1,0000	0,9500	0,9800	0,9500	0,9900	0,9400	1,0000	1,0000	1,0000	0,9789	69,2737
	MSE	0,2955	157,4200	0,9151	337,2600	91,1900	36,3830	0,0000	0,0000	0,0000		
Linear Regression - Quadratic	R-squared	1,0000	Failed - too many terms	0,9900	0,9600	Failed - too many terms	-2,8800	0,9900	0,9900	1,0000	Falied	Failed
	MSE	3,4474		0,4604	287,2200		2285,0000	0,0233	0,0333	0,0000		
Fine Regression Tree	R-squared	1,0000	1,0000	0,9900	0,9800	0,9800	0,9300	1,0000	1,0000	0,9900	0,9856	43,0640
	MSE	12,0540	8,2880	0,4116	120,8900	201,9300	43,9740	0,0068	0,0115	0,0100		
Medium Regression Tree	R-squared	0,9900	0,9900	0,9900	0,9500	0,9600	0,8700	0,9800	1,0000	0,9900	0,9689	89,4429
	MSE	32,7790	20,3120	0,5597	329,9200	345,2000	76,1410	0,0492	0,0102	0,0146		
Coarse Regression Tree	R-squared	0,9700	0,9700	0,9700	0,8400	0,8500	0,7900	0,8900	0,9100	0,8700	0,8956	301,8703
	MSE	77,2070	80,6780	1,3290	1109,3000	1324,8000	122,7900	0,3036	0,2510	0,1737		
NN: 2 layers, 10 hidden neurons, 1 delay	**R-squared**	**0,9999**	**0,9999**	**0,9964**	**0,9969**	**0,9970**	**0,9937**	**0,9684**	**0,9642**	**0,7680**	**0,9649**	**11,0705**
	MSE	**0,1307**	**0,3023**	**0,1126**	**25,5841**	**25,1915**	**47,6578**	**0,0722**	**0,0030**	**0,5807**		
NN: 2 layers, 20 hidden neurons, 1 delay	R-squared	0,9998	0,9999	0,9972	0,9932	0,9976	0,9923	0,9557	0,9613	0,8286	0,9695	17,1554
	MSE	0,3245	0,3225	0,0922	56,0084	39,3327	57,8374	0,0988	0,0032	0,3789		

Annex G

The main code

```
%% Initialization

clear
close all

%% Choose parking lot

final_directory = ChooseParkingLot;

%% Choose time

if contains(final_directory(1),'Spiran') ||
contains(final_directory(1),'Vardtornet')
    [NextMinutes, WeekDayName, time, PH] = TimeAndDate_Norrkoping;
elseif contains(final_directory(1),'ZPS')
    [NextMinutes, WeekDayName, time, PH] = TimeAndDate_ZPS;
else
    [NextMinutes, WeekDayName, time, PH] = TimeAndDate;
end

%% Prediction & Result

if contains(final_directory(1),'ZPS')
    NN_ZPS(final_directory, WeekDayName, time, PH, NextMinutes);
else
    NN(final_directory, WeekDayName, time, PH, NextMinutes);
end
```

ChooseParkingLot

```
function final_directory = ChooseParkingLot

%% Choose parking lot
% list of parking lots
list = {'P+R ÄŒernÃ½ Most 2 (Prague)','P+R LetÅˆany (Prague)',...
'ZPS-P7-0158 (Prague)',...
'Spiran (NorrkÃ¶ping)','Vardtornet (NorrkÃ¶ping)'};
[indx,tf] = listdlg('ListString',list);

%% Directories
% general directory
directory = '/Users/viktorbenes/diplomka/my_system/';
directory = convertCharsToStrings(directory);
% chosen file of the chosen parking lot

switch indx
    case 1
        parkinglot = 'PR_CM2/';
    case 2
        parkinglot = 'PR_L/';
    case 3
```

```
        parkinglot = 'ZPS_0158/';
    case 4
        parkinglot = 'Spiran/';
    case 5
        parkinglot = 'Vardtornet/';
end

% for ZPS Prague is more than one possibility
if contains(parkinglot,'ZPS')
     parkinglot =
{'ZPS_0158/';'ZPS_0179/';'ZPS_0161/';'ZPS_0156/';'ZPS_0134/'};
     lengthOfchar = size(parkinglot);
     for i = 1:lengthOfchar(1)
         final_directory(i,1) =
directory+convertCharsToStrings(parkinglot(i));
     end
else
    parkinglot = convertCharsToStrings(parkinglot);
    % final directory of the chosen parking lot
    final_directory = directory+parkinglot;
end

end
```

TimeAndDate_Norrkoping

```
function [NextMinutes, WeekDayName, time, PH] = TimeAndDate_Norrkoping

%% Set time of day
% set the beginning and ending of the day
start_of_day = datetime('00:00','Format', 'HH:mm');
end_of_day = datetime('23:45','Format', 'HH:mm');
% day time in 5 minute intervals
daytime = (start_of_day:minutes(15):end_of_day)';

%% Choose time for prediction

% list of time
list2 = {'now + 15 minutes','now + 30 minutes'};
[indx,tf] = listdlg('ListString',list2);

switch indx
    case 1
        ChosenTime = datetime('now')+minutes(15);
        NextMinutes = 15;
    case 2
        ChosenTime = datetime('now')+minutes(30);
        NextMinutes = 30;
end

%% Round current time to the nearest 5th minute
% set new format of timestamp data - withou seconds
now = datetime(ChosenTime,'InputFormat', 'dd-MM-yyyy HH:mm:ss','Format',
'dd-MM-yyyy HH:mm');
% round timestamp to 5 minute intervals
now.Minute = 15 * round(now.Minute/15);
% defines time for NN calculation
time = find(sprintf('%d:%d',now.Hour, now.Minute)==daytime);
```

```matlab
%% Set weekdays

% new variable with a name of the day by the function weekday
[DayNumber, DayName] = weekday(now);
WeekDayName = convertCharsToStrings(DayName);

%% Check the date - is it summer or public holidays?

% Set summer holidays
% set code SH (Summer Holidays) if the day is during summer holidays
summerholidays = ['12-06-2021'; '17-08-2021'];
summerholidays = datetime(summerholidays,'InputFormat','dd-MM-yyyy');

if isbetween(now,summerholidays(1),summerholidays(2))
    WeekDayName = WeekDayName+'SH';
end

% Public holidays 2021
% dates of Sweden Public Holidays
PublicHolidays=['01-01-2021'; '06-01-2021'; '02-04-2021'; '05-04-2021';
    '01-05-2021'; '13-05-2021'; '06-06-2021'; '26-06-2021'; '06-11-2021';
    '25-12-2021'; '26-12-2021'];
PublicHolidays=datetime(PublicHolidays,'InputFormat','dd-MM-yyyy','Format',
'dd-MM-yyyy');
% check if the date is public holidays
for i = 1:length(PublicHolidays)
    if now.Month == PublicHolidays(i).Month && now.Day ==
PublicHolidays(i).Day
        WeekDayName = 'PH';
    end
end

% check if the day is workday, workday during holidays or not
if WeekDayName == 'Tue' || WeekDayName == 'Wed' || WeekDayName == 'Thu'
    WeekDayName = 'workday';
elseif WeekDayName == 'TueSH' || WeekDayName == 'WedSH' || WeekDayName ==
'ThuSH'
    WeekDayName = 'workdaySH';
end

%% Peak hours

PH = 0; % generally - no peak hours

% defines where the peak hours starts and ends based on the analysis
if contains(WeekDayName,'SH')
    PeakTimeStart = find('4:30'==daytime);
    PeakTimeEnd = find('10:30'==daytime);
    if time >= PeakTimeStart && time <= PeakTimeEnd
        PH = 1; % it is in peak hours of summer holidats
    end
else
    PeakTimeStart = find('5:00'==daytime);
    PeakTimeEnd = find('9:00'==daytime);
    if time >= PeakTimeStart && time <= PeakTimeEnd
        PH = 1; % it is in peak hours
    end
end
```

```
if contains(WeekDayName,'Sat') || contains(WeekDayName,'SatSH') ||
contains(WeekDayName,'Sun') || contains(WeekDayName,'SunSH')
    PH = 0; % for weekends - no peak hours
end

end
```

TimeAndDate

```
function [NextMinutes, WeekDayName, time, PH] = TimeAndDate

%% Set time of day
% set the beginning and ending of the day
start_of_day = datetime('00:00','Format', 'HH:mm');
end_of_day = datetime('23:55','Format', 'HH:mm');
% day time in 5 minute intervals
daytime = (start_of_day:minutes(5):end_of_day)';

%% Choose time for prediction

% list of time
list2 = {'now + 15 minutes','now + 30 minutes'};
[indx,tf] = listdlg('ListString',list2);

switch indx
    case 1
        ChosenTime = datetime('now')+minutes(15);
        NextMinutes = 15;
    case 2
        ChosenTime = datetime('now')+minutes(30);
        NextMinutes = 30;
end

%% Round current time to the nearest 5th minute
% set new format of timestamp data - withou seconds
now = datetime(ChosenTime,'InputFormat', 'dd-MM-yyyy HH:mm:ss','Format',
'dd-MM-yyyy HH:mm');
% round timestamp to 5 minute intervals
now.Minute = 5 * round(now.Minute/5);
% defines time for NN calculation
time = find(sprintf('%d:%d',now.Hour, now.Minute)==daytime);

%% Set weekdays

% new variable with a name of the day by the function weekday
[DayNumber, DayName] = weekday(now);
WeekDayName = convertCharsToStrings(DayName);

%% Check the date - is it summer or public holidays?

% Set summer holidays
% set code SH (Summer Holidays) if the day is during summer holidays
if month(now) == 7 || month(now) == 8
    WeekDayName = WeekDayName+'SH';
end

% Public holidays 2021
% dates of Czech Public Holidays
PublicHolidays=['01-01-2021'; '02-04-2021'; '05-04-2021'; '01-05-2021';
    '08-05-2021'; '05-07-2021'; '06-07-2021'; '28-09-2021'; '28-10-2021';
    '17-11-2018'; '24-12-2018'; '25-12-2018'; '26-12-2018'];
PublicHolidays=datetime(PublicHolidays,'InputFormat','dd-MM-yyyy','Format',
'dd-MM-yyyy');
% check if the date is public holidays
for i = 1:length(PublicHolidays)
    if now.Month == PublicHolidays(i).Month && now.Day ==
PublicHolidays(i).Day
        WeekDayName = 'PH';
    end
end

% check if the day is workday, workday during holidays or not
if WeekDayName == 'Tue' || WeekDayName == 'Wed' || WeekDayName == 'Thu'
    WeekDayName = 'workday';
elseif WeekDayName == 'TueSH' || WeekDayName == 'WedSH' || WeekDayName ==
'ThuSH'
    WeekDayName = 'workdaySH';
end

%% Peak hours

PH = 0; % generally - no peak hours

% defines where the peak hours starts and ends based on the analysis
if contains(WeekDayName,'SH')
    PeakTimeStart = find('4:30'==daytime);
    PeakTimeEnd = find('10:30'==daytime);
    if time >= PeakTimeStart && time <= PeakTimeEnd
        PH = 1; % it is in peak hours of summer holidats
    end
else
    PeakTimeStart = find('5:00'==daytime);
    PeakTimeEnd = find('9:00'==daytime);
    if time >= PeakTimeStart && time <= PeakTimeEnd
        PH = 1; % it is in peak hours
    end
end

if contains(WeekDayName,'Sat') || contains(WeekDayName,'SatSH') ||
contains(WeekDayName,'Sun') || contains(WeekDayName,'SunSH')
    PH = 0; % for weekends - no peak hours
end

end
```

TimeAndDate_ZPS

```
function [NextMinutes, WeekDayName, time, PH] = TimeAndDate

%% Set time of day
% set the beginning and ending of the day
start_of_day = datetime('00:00','Format', 'HH:mm');
end_of_day = datetime('23:55','Format', 'HH:mm');
% day time in 5 minute intervals
daytime = (start_of_day:minutes(5):end_of_day)';

%% Choose time for prediction
```

```
% list of time
list2 = {'now + 15 minutes','now + 30 minutes'};
[indx,tf] = listdlg('ListString',list2);

switch indx
    case 1
        ChosenTime = datetime('now')+minutes(15);
        NextMinutes = 15;
    case 2
        ChosenTime = datetime('now')+minutes(30);
        NextMinutes = 30;
end

%% Round current time to the nearest 5th minute
% set new format of timestamp data - withou seconds
now = datetime(ChosenTime,'InputFormat', 'dd-MM-yyyy HH:mm:ss','Format',
'dd-MM-yyyy HH:mm');
% round timestamp to 5 minute intervals
now.Minute = 5 * round(now.Minute/5);
% defines time for NN calculation
time = find(sprintf('%d:%d',now.Hour, now.Minute)==daytime);

%% Set weekdays

% new variable with a name of the day by the function weekday
[DayNumber, DayName] = weekday(now);
WeekDayName = convertCharsToStrings(DayName);

%% Check the date - is it summer or public holidays?

% Set summer holidays
% set code SH (Summer Holidays) if the day is during summer holidays
if month(now) == 7 || month(now) == 8
    WeekDayName = WeekDayName+'SH';
end

% Public holidays 2021
% dates of Czech Public Holidays
PublicHolidays=['01-01-2021'; '02-04-2021'; '05-04-2021'; '01-05-2021';
    '08-05-2021'; '05-07-2021'; '06-07-2021'; '28-09-2021'; '28-10-2021';
    '17-11-2018'; '24-12-2018'; '25-12-2018'; '26-12-2018'];
PublicHolidays=datetime(PublicHolidays,'InputFormat','dd-MM-yyyy','Format',
'dd-MM-yyyy');
% check if the date is public holidays
for i = 1:length(PublicHolidays)
    if now.Month == PublicHolidays(i).Month && now.Day ==
PublicHolidays(i).Day
        WeekDayName = 'PH';
    end
end

%% Peak hours

PH = 0; % generally - no peak hours

% defines where the peak hours starts and ends based on the analysis
if contains(WeekDayName,'SH')
    PeakTimeStart = find('4:30'==daytime);
    PeakTimeEnd = find('10:30'==daytime);
    if time >= PeakTimeStart && time <= PeakTimeEnd
        PH = 1; % it is in peak hours of summer holidats
    end
else
    PeakTimeStart = find('5:00'==daytime);
    PeakTimeEnd = find('9:00'==daytime);
    if time >= PeakTimeStart && time <= PeakTimeEnd
        PH = 1; % it is in peak hours
    end
end

if contains(WeekDayName,'Sat') || contains(WeekDayName,'SatSH') ||
contains(WeekDayName,'Sun') || contains(WeekDayName,'SunSH')
    PH = 0; % for weekends - no peak hours
end

end
```

NN_ZPS

```
function NN_ZPS(final_directory, WeekDayName, time, PH, NextMinutes)

%% Import specific data
% zones: 0158; 0179; 0161; 0156; 0134
capacity = [22; 61; 51; 49; 94]; % capacity of each paid parking zone
dontResp = [19; 13; 7; 5; 6]; % percentage of don't respect of rules
residents = [59; 45; 72; 77; 47]; % percentage of residents
for i = 1:length(capacity)
    discoveredplaces(i,1) = round(capacity(i)*dontResp(i)/100);
    discoveredplaces(i,2) = round(capacity(i)*residents(i)/100);
end

% set final CSV of the data input
for i = 1:length(final_directory)
    final_file(i,1)=
final_directory(i,1)+convertCharsToStrings(WeekDayName);
    % import the data
    the_day = table2array(readtable(final_file(i,1)));

%% Set and train the prediction method (NN)

% Defined variables: the_day(:,1:(end-1)) - input time series;
the_day(:,end) - target time series.
X = tonndata(the_day(:,1:(end-1)),false,false);
T = tonndata(the_day(:,end),false,false);

% Training Function Levenberg-Marquardt backpropagation
trainFcn = 'trainlm';

% Creates a Time Delay Network
inputDelays = 1:1; % delay
hiddenLayerSize = 10; % hidden neurons
net = timedelaynet(inputDelays,hiddenLayerSize,trainFcn); % defining NN

% Prepares the Data for training and for simulation
% function PREPARETS prepares timeseries data for a particular network
```

```
[x,xi,ai,t] = preparets(net,X,T);

% Setup Division of Data for Training, Validation, Testing
net.divideParam.trainRatio = 65/100; % Training
net.divideParam.valRatio = 20/100; % Validation
net.divideParam.testRatio = 15/100; % Testing

% Trains the Network
[net,tr] = train(net,x,t,xi,ai);

% Tests the Network (if necessary)
%y = net(x,xi,ai);
%e = gsubtract(t,y);
%performance = perform(net,t,y);

% For the diagram of the NN (if necessary)
%view(net)

%% Output from the NN
% calculates predicted value for defined time
output = round(cell2mat(net(X(time),x(time),ai)));
% minus the untrusted places & residents
output = output - discoveredplaces(i,1) - discoveredplaces(i,2);

% calculates predicted value for defined time one timestep backward
output_minus = round(cell2mat(net(X(time-1),x(time-1),ai)));
% minus the untrusted places & residents
output_minus = output_minus - discoveredplaces(i,1) -
discoveredplaces(i,2);

% calculates predicted value for defined time one timestep forward
output_plus = round(cell2mat(net(X(time+1),x(time+1),ai)));
% minus the untrusted places & residents
output_plus = output_plus - discoveredplaces(i,1) - discoveredplaces(i,2);

%% Management / Decision making system

P = 0; % set probability value

if output <= 0
    P = P + 0; % 0 -> there is no free parking place
elseif PH == 1
    P = P + 50; % 50 -> there are free parking places
    if output_minus > output && output >= output_plus || output_minus == 0
&& output_plus == 0
        P = P + 0; % +0 -> not enough before and after
    elseif output_minus > output && output < output_plus
        P = P + 15; % 15 -> beginning of increasing
    elseif output_minus > 0 && output_minus <= output && output >
output_plus
        P = P + 10; % 10 -> beginning of decreasing
    elseif output_minus > 0 && output_minus <= output && output <=
output_plus
        P = P + 20; % 20 -> fulfill requirements
    end
elseif PH == 0
    P = P + 50; % 50 -> there are free parking places
    P = P + 20; % 20 -> not in peak hour
    if output_minus > output && output >= output_plus || output_minus == 0
&& output_plus == 0
        P = P + 0; % +0 -> not enough before and after
    elseif output_minus > output && output < output_plus
        P = P + 15; % 15 -> beginning of increasing
    elseif output_minus > 0 && output_minus <= output && output >
output_plus
        P = P + 10; % 10 -> beginning of decreasing
    elseif output_minus > 0 && output_minus <= output && output <=
output_plus
        P = P + 20; % 20 -> fulfill requirements
    end
end

P_overall(i,1) = P;
output_overall(i,1) = output;
end

%% Answers
% succesful answer in the same zone
ASucces_1 = sprintf('The probability, that you can park here in the next %d
minutes, is sufficient.\nThere should be approximately %d free parking
places.',NextMinutes,output_overall(find(max(P_overall))));
% succesful answer in the other zone
ASucces_2 = sprintf('We are sorry, but the chosen zone should be full. But
we found free place in the zone %s.\nThe probability, that you can park
there in the next %d minutes, is sufficient.\nThere should be approximately
%d free parking
places.',extractBetween(final_directory(find(max(P_overall))),'system/','/'
),NextMinutes,output_overall(find(max(P_overall))));
% NOT succesful answer
ANotSucces = sprintf('We are sorry but we cannot recommend available
parking spot in the chosen zone or nearest surroundings.\nPlease choose a
different type of transport mode.');
% printing answer
if max(P_overall) >= 75
    if find(max(P_overall)) == 1
        msgbox(ASucces_1,'Success')
    else
        msgbox(ASucces_2,'Success')
    end
else
    msgbox(ANotSucces,'Unsuccessful')
end

end
```

NN

```
function NN(final_directory, WeekDayName, time, PH, NextMinutes)

%% Import specific data
% set final CSV of the data input
final_file = final_directory+convertCharsToStrings(WeekDayName);
% import the data
the_day = table2array(readtable(final_file));

%% Set and train the prediction method (NN)
```

```
% Defined variables: the_day(:,1:(end-1)) - input time series;
the_day(:,end) - target time series.
X = tonndata(the_day(:,1:(end-1)),false,false);
T = tonndata(the_day(:,end),false,false);

% Training Function Levenberg-Marquardt backpropagation
trainFcn = 'trainlm';

% Creates a Time Delay Network
inputDelays = 1:1; % delay
hiddenLayerSize = 10; % hidden neurons
net = timedelaynet(inputDelays,hiddenLayerSize,trainFcn); % defining NN

% Prepares the Data for training and for simulation
% function PREPARETS prepares timeseries data for a particular network
[x,xi,ai,t] = preparets(net,X,T);

% Setup Division of Data for Training, Validation, Testing
net.divideParam.trainRatio = 65/100; % Training
net.divideParam.valRatio = 20/100; % Validation
net.divideParam.testRatio = 15/100; % Testing

% Trains the Network
[net,tr] = train(net,x,t,xi,ai);

% Tests the Network (if necessary)
%y = net(x,xi,ai);
%e = gsubtract(t,y);
%performance = perform(net,t,y);

% For the diagram of the NN (if necessary)
%view(net)

%% Output from the NN
% calculates predicted value for defined time
output = round(cell2mat(net(X(time),x(time),ai)));
% calculates predicted value for defined time one timestep backward
output_minus = round(cell2mat(net(X(time-1),x(time-1),ai)));
% calculates predicted value for defined time one timestep forward
if time < length(x)
    output_plus = round(cell2mat(net(X(time+1),x(time+1),ai)));
else
    output_plus = output;
end

%% Management / Decision making system

P = 0; % set probability value

if output == 0
    P = P + 0; % 0 -> there is no free parking place
elseif PH == 1 % in peak hours
    P = P + 50; % 50 -> there are free parking places
    if output_minus > output && output >= output_plus || output_minus == 0
&& output_plus == 0
        P = P + 0; % +0 -> not enough before and after
    elseif output_minus > output && output < output_plus
        P = P + 15; % 15 -> beginning of increasing
    elseif output_minus > 0 && output_minus <= output && output >
output_plus
        P = P + 10; % 10 -> beginning of decreasing
    elseif output_minus > 0 && output_minus <= output && output <=
output_plus
        P = P + 20; % 20 -> fulfill requirements
    end
elseif PH == 0 % out of peak hours
    P = P + 50; % 50 -> there are free parking places
    P = P + 20; % 20 -> not in peak hour
    if output_minus > output && output >= output_plus || output_minus == 0
&& output_plus == 0
        P = P + 0; % +0 -> not enough before and after
    elseif output_minus > output && output < output_plus
        P = P + 15; % 15 -> beginning of increasing
    elseif output_minus > 0 && output_minus <= output && output >
output_plus
        P = P + 10; % 10 -> beginning of decreasing
    elseif output_minus > 0 && output_minus <= output && output <=
output_plus
        P = P + 20; % 20 -> fulfill requirements
    end
end

%% Answers
% succesful answer
ASucces = sprintf('The probability, that you can park here in the next %d
minutes, is sufficient.\nThere should be approximately %d free parking
places.',NextMinutes,output);
% NOT succesful answer
ANotSucces = sprintf('We are sorry but the probability, that you can park
here in the next %d minutes, is NOT sufficient.\nThere should be
approximately %d free parking places.\nWe recommend choosing another
parking lot or use a different mode of transport.',NextMinutes,output);
% printing answer
if P >= 75
    msgbox(ASucces,'Success')
else
    msgbox(ANotSucces,'Unsuccessful')
end

end
```

Annex H

NN

```
function NN(final_directory, WeekDayName, time, PH, NextMinutes)

%% Import specific data
% set final CSV of the data input
final_file = final_directory+convertCharsToStrings(WeekDayName);
% import the data
the_day = table2array(readtable(final_file));

%% Set and train the prediction method (NN)

% Defined variables: the_day(:,1:(end-1)) - input time series;
the_day(:,end) - target time series.
X = tonndata(the_day(:,1:(end-1)),false,false);
T = tonndata(the_day(:,end),false,false);

% Training Function Levenberg-Marquardt backpropagation
trainFcn = 'trainlm';

% Creates a Time Delay Network
inputDelays = 1:1; % delay
hiddenLayerSize = 10; % hidden neurons
net = timedelaynet(inputDelays,hiddenLayerSize,trainFcn); % defining NN

% Prepares the Data for training and for simulation
% function PREPARETS prepares timeseries data for a particular network
[x,xi,ai,t] = preparets(net,X,T);

% Setup Division of Data for Training, Validation, Testing
net.divideParam.trainRatio = 65/100; % Training
net.divideParam.valRatio = 20/100; % Validation
net.divideParam.testRatio = 15/100; % Testing

% Trains the Network
[net,tr] = train(net,x,t,xi,ai);

% Tests the Network (if necessary)
%y = net(x,xi,ai);
%e = gsubtract(t,y);
%performance = perform(net,t,y);

% For the diagram of the NN (if necessary)
%view(net)

%% Output from the NN
% calculates predicted value for defined time
output = round(cell2mat(net(X(time),x(time),ai)));
% calculates predicted value for defined time one timestep backward
output_minus = round(cell2mat(net(X(time-1),x(time-1),ai)));
% calculates predicted value for defined time one timestep forward
if time < length(x)
    output_plus = round(cell2mat(net(X(time+1),x(time+1),ai)));
else
    output_plus = output;
```

```
end

%% Management / Decision making system

P = 0; % set probability value

if output == 0
    P = P + 0; % 0 -> there is no free parking place
elseif PH == 1 % in peak hours
    P = P + 50; % 50 -> there are free parking places
    if output_minus > output && output >= output_plus || output_minus == 0
&& output_plus == 0
        P = P + 0; % +0 -> not enough before and after
    elseif output_minus > output && output < output_plus
        P = P + 15; % 15 -> beginning of increasing
    elseif output_minus > 0 && output_minus <= output && output >
output_plus
        P = P + 10; % 10 -> beginning of decreasing
    elseif output_minus > 0 && output_minus <= output && output <=
output_plus
        P = P + 20; % 20 -> fulfill requirements
    end
elseif PH == 0 % out of peak hours
    P = P + 50; % 50 -> there are free parking places
    P = P + 20; % 20 -> not in peak hour
    if output_minus > output && output >= output_plus || output_minus == 0
&& output_plus == 0
        P = P + 0; % +0 -> not enough before and after
    elseif output_minus > output && output < output_plus
        P = P + 15; % 15 -> beginning of increasing
    elseif output_minus > 0 && output_minus <= output && output >
output_plus
        P = P + 10; % 10 -> beginning of decreasing
    elseif output_minus > 0 && output_minus <= output && output <=
output_plus
        P = P + 20; % 20 -> fulfill requirements
    end
end

%% Update for accuracy
% Check if the capacity is sufficient

CPM = 35; % max capacity per 15 minutes

if NextMinutes == 15
    TimeVar = 1; % variable for calculation
elseif NextMinutes == 30
    TimeVar = 2; % variable for calculation
end

TimeStepCap = TimeVar*CPM; % time value

if output >= TimeStepCap && output_plus >= TimeStepCap
    P2 = 1; % capacity check value
else
    P2 = 0;
end

%% Answers - updated
% succesful answer
```

```
ASucces = sprintf('The probability, that you can park here in the next %d
minutes, is sufficient.\nThere should be approximately %d free parking
places.',NextMinutes,output);
% NOT succesful answer
ANotSucces = sprintf('We are sorry but the probability, that you can park
here in the next %d minutes, is NOT sufficient.\nThere should be
approximately %d free parking places.\nWe recommend choosing another
parking lot or use a different mode of transport.',NextMinutes,output);
% NOT succesful but with enough capacity
ANotSuccesBut = sprintf('We are sorry but the probability, that you can
park here in the next %d minutes, is NOT sufficient.\nHowever, there should
be enough free places to go there. There should be approximately %d free
parking places.\nWe recommend choosing another parking lot or use a
different mode of transport.',NextMinutes,output);
% printing answer
if P >= 75
    msgbox(ASucces,'Success')
elseif P < 75 && P2 == 1
    msgbox(ANotSuccesBut,'Unsuccessful BUT')
else
    msgbox(ANotSucces,'Unsuccessful')
end

end
```

NN_ZPS

```
function NN_ZPS(final_directory, WeekDayName, time, PH, NextMinutes)

%% Import specific data
% zones: 0158; 0179; 0161; 0156; 0134
capacity = [22; 61; 51; 49; 94]; % capacity of each paid parking zone
dontResp = [19; 13; 7; 5; 6]; % percentage of don't respect of rules
residents = [59; 45; 72; 77; 47]; % percentage of residents
for i = 1:length(capacity)
    discoveredplaces(i,1) = round(capacity(i)*dontResp(i)/100);
    discoveredplaces(i,2) = round(capacity(i)*residents(i)/100);
end

% set final CSV of the data input
for i = 1:length(final_directory)
    final_file(i,1)=
final_directory(i,1)+convertCharsToStrings(WeekDayName);
    % import the data
    the_day = table2array(readtable(final_file(i,1)));

%% Set and train the prediction method (NN)

% Defined variables: the_day(:,1:(end-1)) - input time series;
the_day(:,end) - target time series.
X = tonndata(the_day(:,1:(end-1)),false,false);
T = tonndata(the_day(:,end),false,false);

% Training Function Levenberg-Marquardt backpropagation
trainFcn = 'trainlm';

% Creates a Time Delay Network
inputDelays = 1:1; % delay
```

```
hiddenLayerSize = 10; % hidden neurons
net = timedelaynet(inputDelays,hiddenLayerSize,trainFcn); % defining NN

% Prepares the Data for training and for simulation
% function PREPARETS prepares timeseries data for a particular network
[x,xi,ai,t] = preparets(net,X,T);

% Setup Division of Data for Training, Validation, Testing
net.divideParam.trainRatio = 65/100; % Training
net.divideParam.valRatio = 20/100; % Validation
net.divideParam.testRatio = 15/100; % Testing

% Trains the Network
[net,tr] = train(net,x,t,xi,ai);

% Tests the Network (if necessary)
%y = net(x,xi,ai);
%e = gsubtract(t,y);
%performance = perform(net,t,y);

% For the diagram of the NN (if necessary)
%view(net)

%% Output from the NN
% calculates predicted value for defined time
output = round(cell2mat(net(X(time),x(time),ai)));
% minus the untrusted places & residents
output = output - discoveredplaces(i,1) - discoveredplaces(i,2);

% calculates predicted value for defined time one timestep backward
output_minus = round(cell2mat(net(X(time-1),x(time-1),ai)));
% minus the untrusted places & residents
output_minus = output_minus - discoveredplaces(i,1) -
discoveredplaces(i,2);

% calculates predicted value for defined time one timestep forward
output_plus = round(cell2mat(net(X(time+1),x(time+1),ai)));
% minus the untrusted places & residents
output_plus = output_plus - discoveredplaces(i,1) - discoveredplaces(i,2);

%% Management / Decision making system

P = 0; % set probability value

if output <= 0
    P = P + 0; % 0 -> there is no free parking place
elseif PH == 1
    P = P + 50; % 50 -> there are free parking places
    if output_minus > output && output >= output_plus || output_minus == 0
&& output_plus == 0
        P = P + 0; % +0 -> not enough before and after
    elseif output_minus > output && output < output_plus
        P = P + 15; % 15 -> beginning of increasing
    elseif output_minus > 0 && output_minus <= output && output >
output_plus
        P = P + 10; % 10 -> beginning of decreasing
    elseif output_minus > 0 && output_minus <= output && output <=
output_plus
        P = P + 20; % 20 -> fulfill requirements
    end
elseif PH == 0
    P = P + 50; % 50 -> there are free parking places
    P = P + 20; % 20 -> not in peak hour
    if output_minus > output && output >= output_plus || output_minus == 0
&& output_plus == 0
        P = P + 0; % +0 -> not enough before and after
    elseif output_minus > output && output < output_plus
        P = P + 15; % 15 -> beginning of increasing
    elseif output_minus > 0 && output_minus <= output && output >
output_plus
        P = P + 10; % 10 -> beginning of decreasing
    elseif output_minus > 0 && output_minus <= output && output <=
output_plus
        P = P + 20; % 20 -> fulfill requirements
    end
end

P_overall(i,1) = P;
output_overall(i,1) = output;
end

%% Answers
% max and indicates of max number of free places
[M_output, I_output] = max(output_overall);
% max and indicates of max P number
[M_P, I_P] = max(P_overall);

% succesful answer in the same zone
ASucces_1 = sprintf('The probability, that you can park here in the next %d
minutes, is sufficient.\nThere should be approximately %d free parking
places.\nBut the maximum free places should be in the zone %s with %d
number of free
places.',NextMinutes,output_overall(I_P),extractBetween(final_directory(I_o
utput),'system/','/'),M_output);
% succesful answer in the other zone
ASucces_2 = sprintf('We are sorry, but the chosen zone should be full. But
we found free place in the zone %s.\nThe probability, that you can park
there in the next %d minutes, is sufficient.\nThere should be approximately
%d free parking
places.',extractBetween(final_directory(I_P),'system/','/'),NextMinutes,out
put_overall(I_P));
% NOT succesful answer
ANotSucces = sprintf('We are sorry but we cannot recommend any parking spot
in the chosen zone or nearest surroundings.\nPlease choose a different type
of transport mode.\nBut in the zone %s shoudl be %d number of free
places',extractBetween(final_directory(I_output),'system/','/'),M_output);
% printing answer
if M_P >= 75
    if I_P == 1
        msgbox(ASucces_1,'Success')
    else
        msgbox(ASucces_2,'Success')
    end
else
    msgbox(ANotSucces,'Unsuccessful')
end


end
```