Hochschule Rhein-Waal

Rhine-Waal University of Applied Sciences

Faculty of Communication and Environment

Demand Forecast of Vytal in Cologne:

Correlation and Time Series Analysis of the Transactions

Master Thesis

by

Kültigin Bozdemir

24205

Hochschule Rhein-Waal

Rhine-Waal University of Applied Sciences

Faculty of Communication and Environment

Demand Forecast of Vytal in Cologne:

Correlation and Time Series Analysis of the Transactions

A Thesis Submitted in Partial
Fulfillment of the Requirements of the
degree of

Master of Science

In
Information Engineering and Computer Science

By

Kültigin Bozdemir

24205

Supervised by:

Prof. Dr. Agatha Kalhoff

Alexander Niclas Popovic

Submission Date:

06th July 2022

Abstract

Vytal is a startup company offering the customers environment-friendly reusable food containers, a substitution for disposable packaging, particularly for restaurants or bakeries. Customers order their food with the containers and return the containers to any partner store within two weeks after the check-out. However, continuous circulation of the containers between consumers and partner stores requires Vytal to rebalance the inventories at the stores where the forecasting is one of the major tasks as an input in such an optimization problem. Thus, this thesis aims to do correlation and time-series analyses to predict the demand at the stores in Cologne.

Internal interdependencies and correlations with external factors have been studied in this respect. Regarding interdependencies, association analysis for different types of containers has been analyzed to ascertain whether there is an association within the transactions regarding different container types. Only one interesting association was found though it has a low support value. The second interdependency is related to the geospatial autocorrelation analysis. Even though Moran's I global autocorrelation analysis showed some clustering concerning the transaction sizes, local autocorrelation analysis did not justify an interesting clustering of the stores in the space domain.

Regarding external factors, weather temperature, stores' proximity to public transport stations, proximity to green fields, proximity to event places, and corona cases were examined to whether they are correlated to the transaction sizes. Apart from Covid19 cases, there has not been found any significant correlating variable. However, Covid19 cases did not play any significant role after the 2021 autumn.

In the light of the abovementioned and time-series analyses, Autoregressive Integrated Moving Average forecasting modeling has been finally used to predict the aggregate demand in Cologne as a whole, while Facebook Prophet was used to predict the demand at the individual store level.

Python packages like Pysal, Scikit-learn, Scipy, NetworkX, and Facebook Prophet have been used to conduct the analyses.

The following Master thesis contains confidential data and information disclosed by Vytal GmbH. It may not be disclosed, published, or made known in any other manner, including in the form of extracts, without the explicit permission of the company. The thesis is made available to members of the Examination Board solely for the purpose of assessment.

Table of Contents

List of Abbreviations

ARIMA	Autoregressive Integrated Moving Average
ARIMAX	ARIMA with exogenous variables
ACF	Autocorrelation Function
AR	Autoregressive
ADF	Augmented Dickey-Fuller
CDF	Cumulative Distribution Function
CRS	Coordinate Reference System
KDE	Kernel Density Estimation
MA	Moving Average
MAE	Mean Absolute Error
ML	Machine Learning
NRW	North Rhine-Westphalia
OSM	Open Street Maps
PACF	Partial Autocorrelation Function
SARIMAX	Seasonal ARIMAX

List of Figures

List of Tables

Chapter 1 Introduction

There has been increasing public support for a greener environment for decades as more and more studies have justified global warming, plastic pollution (Rhodes, 2018), and other environmental concerns. Consequently, governments and societies worldwide started to discourage pollution or encourage greener solutions in almost every area, including recycling, as public opinion supports a greener environment.

One of the recent meta-analyses about plastic pollution in the world has been carried out by Bucci et al. (2020) shows the effect of plastic pollution on the environment. Ilyas et al. (2018) have recently studied literature reviews about plastic pollution, in them, they provided the most recent and relevant studies regarding plastic pollution.

Food disposable packages are one source of plastic pollution. As a company that is aware of this environmental thread and public support, Vytal is one of the food packaging service providers that offers its customers reusable food containers in favor of a greener environment and completion with disposable food packaging systems well-rooted in the industry.

Even though Vytal was founded in 2019, it has a dense network of partners, usually cafes and restaurants in Germany, and expands continuously into Europe. Food containers in different sizes circulate between partners and customers. Customers order their food with those reusable containers and return them to any partner in the world within two weeks of their order. Vytal generates revenue for every container to be used for that purpose. Such a system would be competitive against disposable packaging systems. Moreover, reusable containers are more environment-friendly in the long run. Vytal reached one million packs in May 2021, within two years after it was founded in 2019. (Vytal, 2022).

Source: (Vytal)

Figure 1 Vytal's Food Containers

The map below shows the locations of the partner stores in Germany, France, the UK, and Austria by March 2022. The map created is with Python's Folium package, which retrieves the underlying map from (Open Street Maps, 2022).

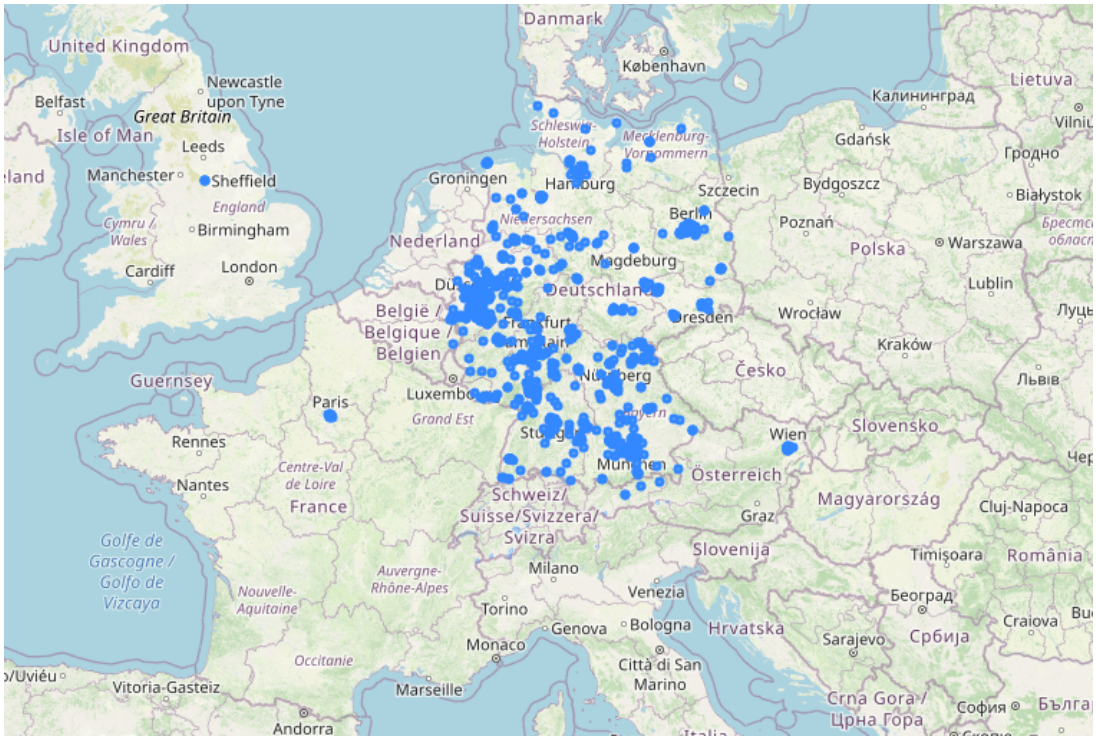


Figure 2 Vytal's partner stores

Vytal manages its operations with the help of a smartphone scanning app and a database where the transactions are recorded. The database enables to follow the transactions across this network of partners and customers. The database has the master tables for users (customers,

admin, store app users, etc.), stores, containers, as well as transactional tables that record the check-ins and check-outs of the containers.

Vytal has mainly two types of customers; partner stores and consumers. Consumers also have two sub-categories; ordinary and catering consumers. While ordinary consumers check out a few containers, the catering companies check-in and out in large quantities.

Every container, partner store, and customer has a unique id. Every check-in or check-out transaction is recorded with these three id numbers and timestamps. The employee at the store scans the barcode on the container that records it in the database. At the same time, the consumer also gets a notification about the transaction. Therefore, such a system identifies container flows between the customers and stores with timestamps. Similarly, container stocks can be calculated at the partners at a particular time. It is worth stating again that the customers can return the containers to any partner store of Vytal.

The company needs to rebalance these circulating containers across the partner stores such that the stores can meet the demand of the upcoming period. Although the company has different intra-city and inter-cities relocation models, it mostly plans one rebalancing tour a day in Cologne using a bike with a small cart enabling multiple hundreds of containers to be transported.

The rebalancing problem is two-fold; the first issue arises from spatial uncertainty because some containers are not necessarily returned to the same stores from which they were taken. The second aspect of the problem is related to the return time. Return times of the containers are stochastically distributed. Shortly, stochasticity in time and space dimensions is a challenge that results in a significant variance of check-ins or check-outs at the stores.

Inventory optimization is one of the fields that has been studied intensively. One comprehensive literature review about inventory management or inventory control has been conducted by Sing & Alay Verma. (2018). Classical inventory optimization usually focuses on one-directional replenishment, from warehouses to customers. However, Vytal's relocation operations differ from classical inventory optimization. Vytal has to take surplus containers from the partners and bring them to the partners where there is a shortage of containers. Regional warehouses play a secondary role in such a redistribution network. In that sense, the literature about reverse logistics concerns Vytal's relocation operations. A recent literature review about reverse logistics was studied by Rachih, Mhada, & Chieheb (2019).

From the inventory rebalancing aspect, sharing services like car- or bike-sharing is similar to the business of Vytal. Many studies have been conducted on optimizing inventory levels and the relocation of the system elements. Concerning Vytal's rebalancing effort, a bike-sharing service has more similarities with Vytal's operations since both operate in urban areas, particularly around city centers. Transportation costs between nodes play a secondary role rather than the inventory status in both cases. One of the studies concerning bike-sharing rebalancing was conducted by Leonardo Caggiani (2012). Recently, Zhou (2018) also studied the forecast of the bike inventory levels at the bike-sharing stations by using the Markov-Chain method after giving a list of methods that have been recently studied.

Nevertheless, Vytal's business differs from those sharing services in an important aspect. Container inventory level at the stores is in large numbers and usually not subject to storage constraints, while the inventory of sharing services, e.g., bike-sharing services, is constrained by the number of available docking points at the stations.

As mentioned above, check-ins distributions are stochastic to some degree. In addition, the demand in the stores also varies significantly, which increases the stochasticity even more. Thus, forecasting the demand will supply the inputs for the ultimate task, i.e., optimizing container rebalancing between the stores. Fortunately, time series analysis has a long history and is strengthened by recent developments in machine learning (ML) (Lazzeri, 2021). Faloutsos et al. (2018) provide a brief list of applied methods for time series forecasting, while Cerqueria, Torgo, & Soares (2019) show a prediction performance comparison between different methods, including the new ML approaches and the classical methods like the Autoregressive Integrated Moving Average (ARIMA) model.

1.1 Aim of the Thesis

This thesis aims to develop a demand prediction model for Vytal such that this prediction helps the rebalancing effort across the stores in Cologne. In this respect, relevant factors need to be identified in two categories, i.e., internal dependencies and external correlating variables. Concerning the first category, association analysis will reveal whether there are intra-dependencies in the transactions regarding the different container types. Spatial autocorrelation analysis fulfills the second part of internal dependence analysis by ultimately showing whether stores' proximity to other stores is relevant. Then the external factors, namely weather temperature, stores' proximity to public transport stations, proximity to green fields, proximity

to event places, and corona cases, will be examined to determine whether they are correlated to the transaction sizes.

Although the company's ultimate goal is to rebalance the containers in Cologne efficiently, the demand forecast analysis will lay the foundations to achieve it.

1.2 Limitations

This thesis analyzes Vytal's operations in Cologne and focuses mainly on the transactions of ordinary consumers. Except for the association analysis, only the container type-1 has been considered for the correlation and prediction analysis.

As mentioned above, the containers are circulating between stores and the customers continuously, which also means the stores are getting containers from the customers (check-ins) and borrowing them (check-outs) at the same time, of course, with different distributions.

The final limitation regulates the granularity. It is necessary to aggregate the transactions into hours, days, etc. Although demand at intraday intervals may also be of interest, the daily aggregate demand has been selected to decrease the sampling noise.

1.3 Data Sources

Vytal's Data

Vytal has a database where transactions are stored along with master data such as tables of customers, containers, and stores. As mentioned earlier, CSV files generated from the database were provided for this study. The data cleaning process, aggregations, and summaries are explained in Appendix A.

Weather Data

The weather data in Cologne as of the start of the Vytal's operations is downloaded from German Weather Service (Deuscher Wetterdienst, 2022) using a python script provided in Appendix A. The weather data is already aggregated daily. It contains temperature, humidity, wind condition, cloudiness, and other features.

Green Fields

The geolocations of green field polygons have been downloaded from Cologne Open Data (Offene Daten Köln, 2022). The data contains the area types of biotope areas, forest areas, cemeteries, green areas, allotments, special green areas, and playgrounds.

Public Transport Network Data

The Public Transport Data (Subway, Bus, Train Network) in Cologne is publicly available and downloaded from Verkehrsverbund Rhein-Sieg's website (2022). It has a graph structure where the vertices represent the stops, and the edges represent the lines of the buses or trains.

Covid19 Cases Data

Covid19 case numbers provided by Robert-Koch Institute are available in the GitHub repository (2022). The words Covid19 and corona are exchangeably used.

Events

Cologne's official website shows events in Cologne on an online calendar (Stadt-Köln, 2022). The events have been scrapped from this website using Python packages.

1.4 Method

First, the needed data is obtained from different sources. The data sources are given below. The data obtained are primarily processed in a Python environment and partly in the QGIS software. QGIS is a compact open-source software for geospatial analysis (QGIS, 2022).

Jupyter Notebook files accompany the chapters and are given in Appendix A. The Jupyter and relevant data files are organized so that the results of the analyses can be reproducible.

After data being obtained from different sources as well as from Vytal, data exploration and descriptive statistics have been accomplished. Distribution of the transaction sizes regarding the type of containers, time of the day, etc., have been plotted to capture basic patterns and variance. To illustrate the variance of multiple stores' transaction sizes, we used box plots to give a visual comparison.

Since the geolocations of the stores are prominent, spatial visualization will also be used with the help of QGIS and Python packages like Geopandas (2022), Shapely (2022), and Folium

(2022). The maps shown in the following chapters are created with the help of Folium unless otherwise stated. Folium gets the underlying map from Open Street Maps (2022).

Since the flows of the containers between stores are causing unbalanced container distribution and eventually rebalancing efforts, graphs with NetworkX will be constructed to detect possible patterns between stores.

After the exploration phase, association analysis will be carried out as the first internal dependency analysis. Association analysis aims to identify if one container type's check-outs are associated with another. Although the transactional data table has a Transaction ID column, a clustering algorithm, namely DBSCAN (Scikit-learn, 2022) will be deployed to detect transactions in the time dimension. After detecting further transactions, the Apriori module (Mlxtend, 2022) will be deployed to capture the associations between the container types.

The second test of the internal dependency analysis is the spatial autocorrelation test, which aims to explore if there is a spatial clustering in terms of transaction size and the geolocation of the stores. Python's spatial analysis package, Pysal (2022), particularly its Moran's I statistics simulation module, will be deployed for this task. Neighborhood types and corresponding parameters will be discussed in Chapter 4.

The second part of dependency analysis consists of external regressor analysis to see if there is a correlation between the size of check-outs and the aforementioned external factors.

After exploring relevant factors, the last part of the thesis contains the time series analysis. The forecasting will be studied at two levels which also refers to two planning horizons. The higher echelon is the forecast of the aggregated demand in Cologne for the time horizon of several weeks. Vytal can rebalance the inventory level across German cities or procure new containers. Autoregressive Integrated Moving Average (ARIMA) will be deployed to predict the daily demand in Cologne. At this stage, Statsmodels (2022) library is heavily used along with other complementing libraries like Pandas, NumPy, etc. The lower forecast echelon is the demand prediction at an individual store. Because the stores have stochastic transaction sizes and environments, a general prediction model for all stores is impossible. Therefore, one store will be selected to develop a prediction model with Prophet software. Facebook's Prophet is eventually deployed to take advantage of its easy handling of holidays and multi-seasonality in addition to the additional regressors.

An alternative to the prediction at one store, a more straightforward approach, kernel density estimation (KDE), will be implemented in Appendix C. KDEs are commonly used in inventory optimization to identify a service level (Vandeput, 2020).

The Python packages' versions will also be printed in the corresponding Jupyter files in Appendix A.

The personal pronoun 'we' used in the following sections refers to the author and the readers.

1.5 Structure of the Thesis

Chapter 2 is dedicated to data visualization and descriptive statistics of Vytal's transactions in Cologne. Geolocation of the stores and corresponding basic statistics will also be plotted to pave the path for further geospatial analysis in the following chapter.

Chapter 3 covers the association analysis of container types. DBSCAN clustering algorithm leverages detecting the transaction ids.

Chapter 4 comprises geospatial analyses of Vytal's stores, specifically Moron's I statistics.

Chapter 5 covers the correlation analysis with external factors such as weather, temperature, etc.

Time series analysis and prediction are conducted in chapter 6. The first section is devoted to the aggregated prediction in Cologne, while the second is allocated for a single store's prediction analysis.

Chapter 2 Vytal's Data Exploration

Vytal has two primary customers, i.e., consumers and partner stores to whom it offers the food containers. The following sections will introduce these customers and their interaction with the containers.

2.1 Vytal's Partner Stores

Vytal has more than 3000 partners in Europe by February 2022. However, we analyze the operations only in Cologne. Administrative boundaries of Germany are obtained from DIVA-GIS (2022), which enables the filtering of the 377 partner stores within the boundaries of Cologne. As seen on the map below, most stores are in the city center.

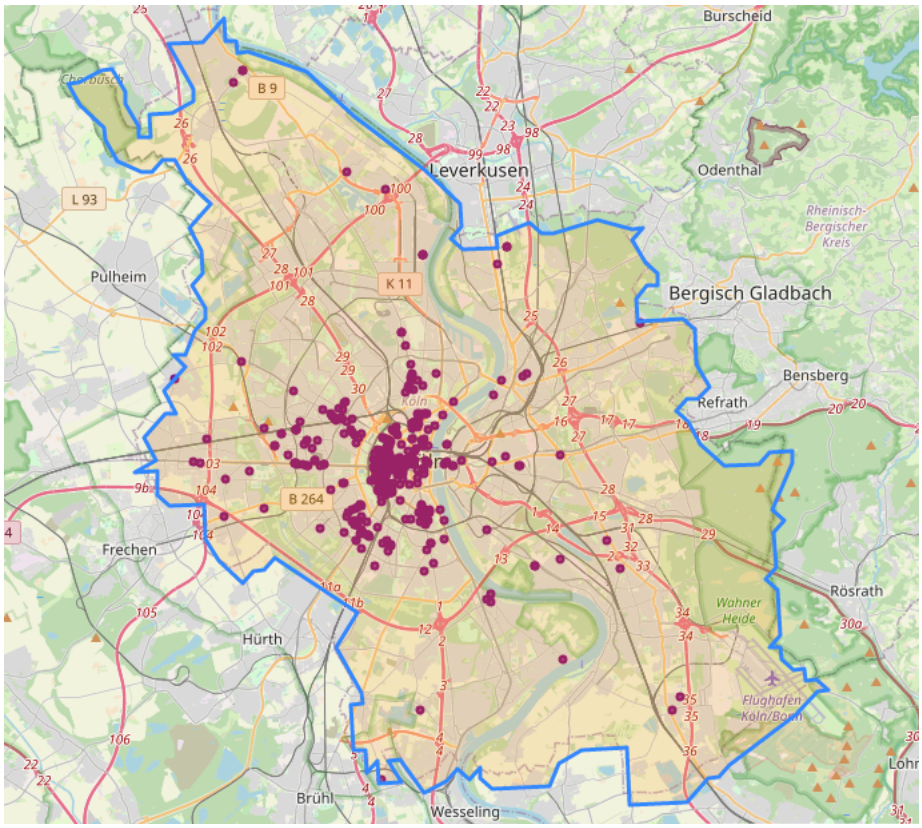


Figure 3 Partner stores in Cologne

Vytal has almost half-million check-outs and roughly 450000 check-ins in Cologne until March 2022. However, Vytal relocated less than 15000 containers in Cologne. Most of these

containers' relocations have been made with the cargo bike. That means the company depends heavily on the stock levels at the stores. The map below shows the total balance level as of March 2022. The red indicates the most negative total container balances, and dark green indicates the most positive total balances. This map shows where the sinks (red stores) are and where the containers should be relocated from other green stores.

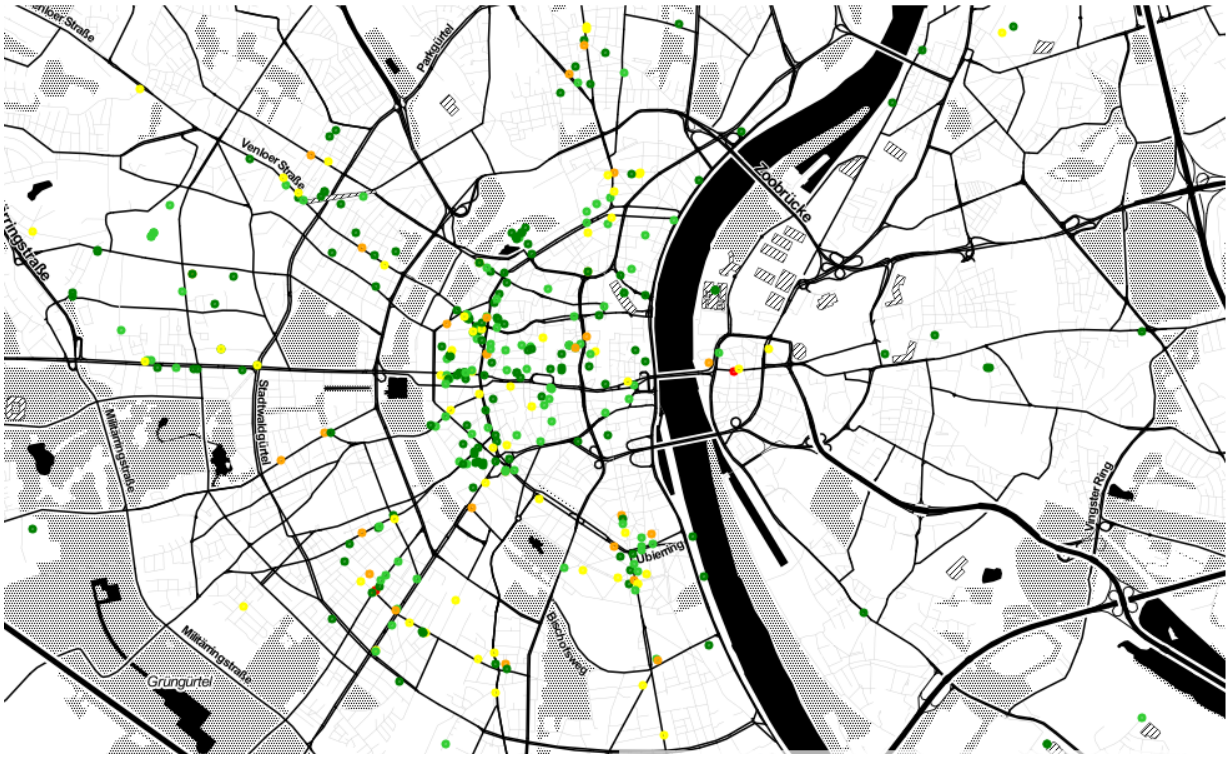


Figure 4 Container balance of the stores

However, this map does not reveal the daily balance level, which is more interesting. In order to capture this information, we used a heatmap in the following figure to plot the most recent 4000 balance levels at the stores. The x-axis shows the stores, y-axis shows the date, and the colors indicate the balance level at the corresponding particular date and store. The heatmap shows that most balance levels are recorded around zero, while only a few stores have a negative balance smaller than -30. That justifies the finding of the previous map as well. Any column-wise clustering of darker teal green on the heatmap shows that this store usually has negative daily balances, as seen on a few columns. Similarly, a row-wise darker teal clustering would show a peak day, like a black Friday phenomenon, that Vytal would have more check-outs than check-ins on this particular day, though it is not the case in this figure.

Figure 5 Heatmap of stores' balances

2.2 Vytal's Transactions' Distributions

In this section, we will explore transactional distributions of Vytal in Cologne. As stated earlier, every check-in or check-out of a transaction is recorded in the database with timestamps.

2.2.1 Consumers' Check-out Distribution

Vytal has more than 20000 ordinary consumers in Cologne besides the catering customers. Their daily check-out distribution is shown on the following histogram. As seen on the plot, most consumers have a few check-outs, though the distribution has a long right tail, up to 72 check-outs in a day. The daily check-outs larger than five are also shown in a second plot to show the tail. Since it is not plausible that ordinary consumers can have more than five check-outs in a day, we assess those consumers as the outliers and carry out the following analysis without them.

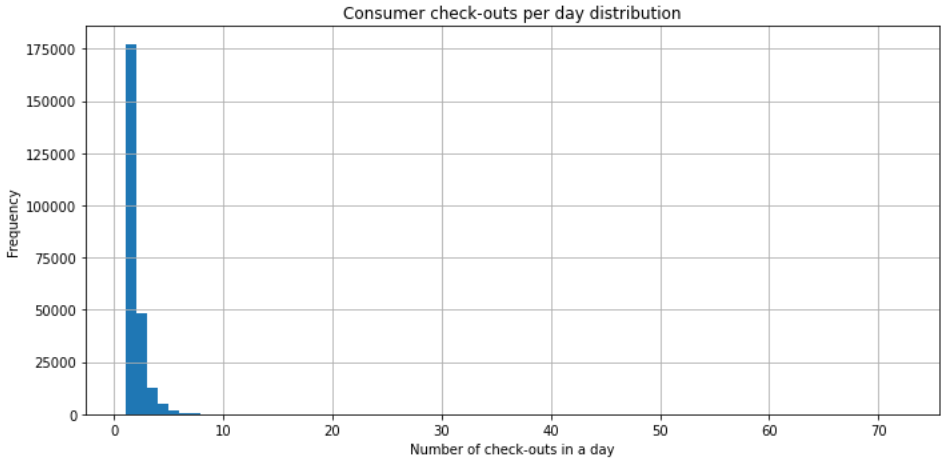


Figure 6 Daily check-out distribution

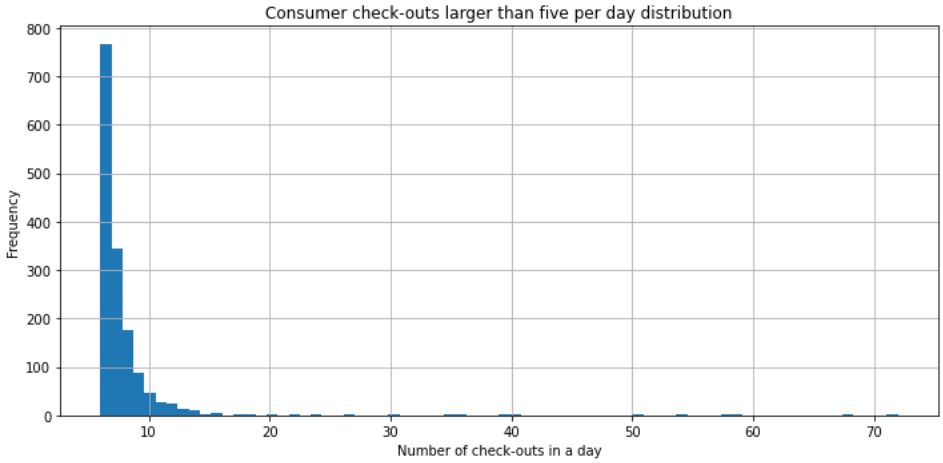


Figure 7 Distribution of daily check-outs larger than five

2.2.2 Container Type Distribution

The company has 16 different food containers of different sizes, and their distribution of the check-outs is shown in the histogram below. Type 1 (1250ml Bowl) is the most used container, as seen on the histogram. For the sake of simplicity, we will work only on type-1 in the following chapters, except for association analysis in Chapter 3.

Table 1 Type of containers

Id	Name	Description
1	1250ml Bowl	A standard bowl
2	500ml Bowl	A smaller Bowl
3	750ml Bowl	A medium bowl
4	Sushibox	A box for sushi
5	Cup	A reusable cocktail glass
6	Compartment Bowl	A bowl with two compartments
7	Weckglas	Weckglas
8	Pizzabox	Pizzabox
9	Snackpot	Snackpot
10	Cup_pp 0.3l	A coffee cup made of PP - size 0.3l
11	2250ml Bowl	A large bowl with restricted checkin for Heimk...
12	Cup_pp 0.2l	A coffee cup made of PP - size 0.2l
13	Glasbox Gastro	A glas container with two compartments
14	Glasbowl	Glasbowl - Milubo Glasbowl
15	2000ml rectangular bowl ("Landgasthof Hartmann")	2000ml rectangular bowl (nordic green) with re...
16	Sushi Box tall	Sushi Box tall

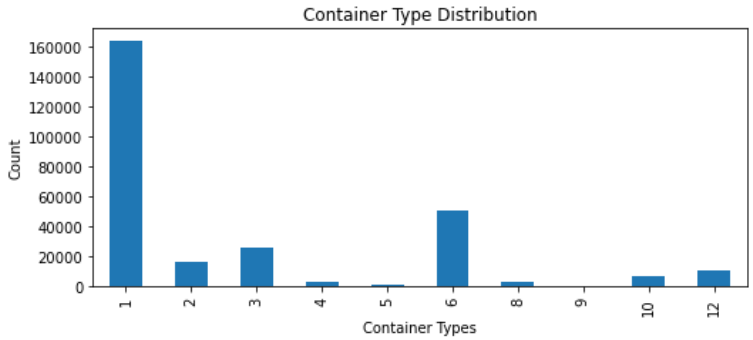


Figure 8 Container types distribution

2.2.3 Hourly Distribution of the Check-outs

The stores have peak hours around noon, as seen on the plot below. The check-in and check-out distributions are almost overlapping. Both distributions have a long tail until late evening, around 9 pm. Keeping the containers cleaned is the responsibility of the stores, although most of the consumers return the containers cleaned. That means stores need some time to clean the incoming containers and prepare them for the next check-outs. Since the stores have different internal processes, the cleaning process and length of this time are not transparent to Vytal and cannot be followed with the scanning system. The containers are needed at these peak hours, around noon. The demand for later hours can be met from the incoming containers again around noon. Therefore, the task is to estimate especially the demand at noon time. It holds

valid also for the rebalancing task such that the containers should be relocated before rush hours.

One further interpretation from this distribution is that we can aggregate the transactions on a daily basis for further forecasting analysis since the transactions are mostly clustered at noon time, and this is the most interesting finding for the relocation task. In other words, we can reduce the complexity of the problem by aggregation, thanks to the finding discussed above.

A second finding from the plot is that consumers are returning the old containers as they are checking out new containers as we would expect a plausible behavior.

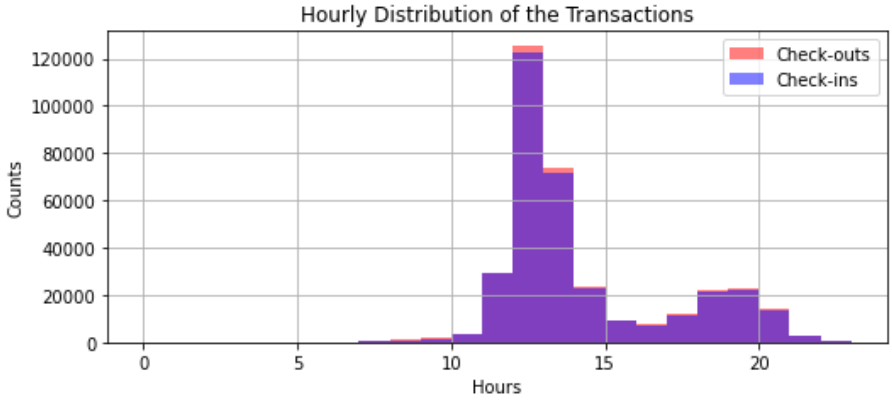


Figure 9 Hourly distributions of the transactions

2.2.4 Weekly Distribution of the Check-outs

The distribution of weekdays in check-outs is depicted below. Wednesday is the peak day for check-outs, while the weekends have low sales.

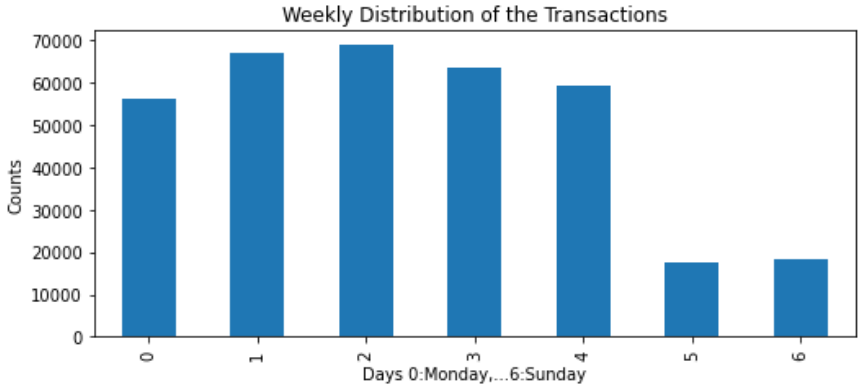


Figure 10 Weekly distribution of the check-outs

Chapter summary

The long-term and daily total balance analysis reveals that the majority of the store rely on their stock levels to meet the demand. Only a few stores have a significant negative balance which can be identified as sinks. That also implies that those stores must be resupplied often to meet the upcoming demand.

The number of check-outs of each consumer reveals that most of them borrow a few containers in a day as expected. However, the long tail in the distribution indicates some inconsistencies or false records in the database. Therefore, we assume that the consumers with more than five check-outs cannot be ordinary consumers. Consequently, we exclude them in the analysis as of Chapter 4.

Container type distribution reveals that the 1250ml bowl is the most demanded container type.

Noon time is the peak time in a day with a long left tail till 9 pm. This finding can help with a complexity reduction by aggregating the check-outs daily. The weekly distribution is not that much surprising. It has a slight peak on Wednesday and low check-outs at weekends.

Chapter 3 Association Analysis

This chapter aims to determine whether there is a dependency within the transactions regarding different container types. In other words, we hypothesize that check-outs of one type of container effects the check-out size of another type.

Secondarily, we can analyze the transactions of containers one by one if there is no significant interdependency between the container types so that we can develop prediction models independently. Briefly, association analysis can help us reduce the problem's complexity.

Vytal's check-outs are most frequently done with the type-1 (Bowl, 1250 ml). The distribution of container types is shown in Figure 8.

3.1 Definitions and Measurements

Association analysis, also known as market basket analysis, helps discover interesting relationships hidden in a large transactional data set (Tan, Steinbach, Karpatne, & Kumar, 2020, S. 213-306). The most relevant definitions are given below.

"The uncovered relationships can be represented in the form of sets of items present in many transactions , which are known as frequent itemsets, or association rules, that represent relationships between two itemsets" (Tan, Steinbach, Karpatne, & Kumar, 2020, S. 213).

Let I=$\{i_1, i_2, \ldots i_d\}$ be set of all itemsets and T=$\{t_1, t_2, \ldots t_N\}$ be set of all transactions. The rules are typically notated as {X}→{Y}, where X and Y are disjoint itemsets[1]. Support and confidence are the primary measurements. Support count is a property of an itemset and refers to the number of transactions in the dataset that contain a particular itemset, e.g., for itemset X, and the support count is notated as σ(X). Support returns how often a rule is applicable in a

[1] Itemset X is also called as antecedents while itemset Y is called as consequent.

dataset. Confidence determines how frequently items in Y appear in transactions that contain X (Tan, Steinbach, Karpatne, & Kumar, 2020, S. 214).

Support: $s(X \rightarrow Y) = \frac{\sigma(X \cup Y)}{N}$ **Equation 1**

Confidence: $c(X \rightarrow Y) = \frac{\sigma(X \cup Y)}{\sigma(X)}$ **Equation 2**

$\text{Lift} = \frac{c(X \rightarrow Y)}{s(Y)} = \frac{\sigma(X \cup Y)}{\sigma(X) \times \sigma(Y)}$ **Equation 3**

3.2 Rules Generations from Vytal's Transactions

3.2.1 Determination of Transactions

Since the transactional dataset has an inconsistent transaction id, it is necessary to generate a column of transaction id in the dataset. Furthermore, we also discovered that some consumers executed multiple transactions in a short time, i.e., in minutes. That means one consumer checked out another batch of containers only minutes after he or she had borrowed some containers in the previous transaction. We assume that those transactions close to each other in time dimension can be assessed as a single transaction instead of two separate transactions. Therefore, a clustering in time dimension and consumers can help us to identify those actual transactions. DBSCAN algorithm has been used with two attributes, i.e., time and consumers after the consumers are defined as categorical data with the help of Gower distance measurement, which was introduced by Gower (1971).

3.2.2 Association Rules Generation

After determining the actual transaction ids, the apriori algorithm has been used to find association rules in Vytal's transactions. The following table shows the rules after selecting item sets with a minimum 0.0001 support level. Antecedents in the table refer itemset X, and consequents refer itemset Y. Lift values close to 1 refer that X and Y item sets are independent. Higher lift values indicate the existence of an interesting rule. As seen below in the table, the rules can be redundant since the rules are generated in both X→ Y and Y→X directions.

Table 2 Associations rules of Vytal's transactions

	antecedents	consequents	antecedent support	consequent support	support	confidence	lift
2	(8.0)	(4.0)	0.010616	0.010837	0.000399	0.037600	3.469646
3	(4.0)	(8.0)	0.010837	0.010616	0.000399	0.036834	3.469646
4	(10.0)	(5.0)	0.026247	0.004140	0.000140	0.005339	1.289521
5	(5.0)	(10.0)	0.004140	0.026247	0.000140	0.033846	1.289521
0	(2.0)	(4.0)	0.064257	0.010837	0.000718	0.011168	1.030594
1	(4.0)	(2.0)	0.010837	0.064257	0.000718	0.066223	1.030594

The only interesting rule is in this dataset that container type 8 is likely checked out with container type 4. Note that support of this rule is 0.00039. it means that such a basket containing type 8 and type 4 will happen nearly four times in every 10000 transactions.

Chapter summary

We found only one interesting association rule between the container type-8 and type-4 with a low support level. Hence, this chapter concludes that the interdependency between container types is negligible. In other words, we can continue to analyze the transactions of different container types independently. For the sake of simplicity, we continue to analyze only container type 1 in the following chapters, as it is the most used type.

Chapter 4 Spatial Autocorrelation Analysis

This chapter covers the geospatial analysis and answers the question; Is there any spatial clustering between the stores regarding the size of the inter-store check-outs? To answer this question, we will show the containers' flow between the stores.

4.1 The flow of Containers Between the Stores

We introduce a new definition in this chapter. The ‘inter-stores flow’ refers to the movements of the containers which are returned to a different store from where they were initially checked out. To represent the inter-flows, we constructed a graph in NetworkX (2022) such that nodes represent the stores; edges represent the inter-store flows. Remember again; only the transactions are taken into accounts that are returned to a different store. The transactions, which are a loop, are ignored in this graph. The graph is imported to QGIS so that we can visualize the graph easily, as seen below. White arrowheads show the flow direction, and the color range of white-red shows the total size of the flows ranging up to 1000 container flows. On the map below, we have shown the total flows for more than two years. The white arrows dominate the graph, meaning most flows have small sizes. The filtered version of the graph (total flows>49) is shown next.

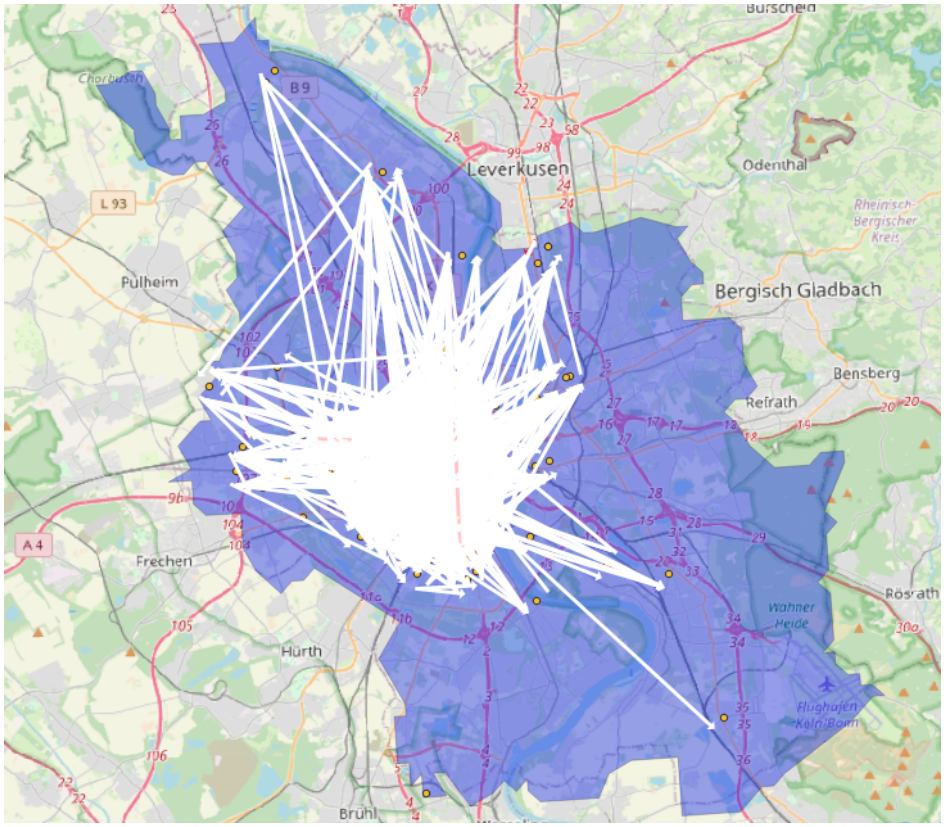


Figure 11 Container flows

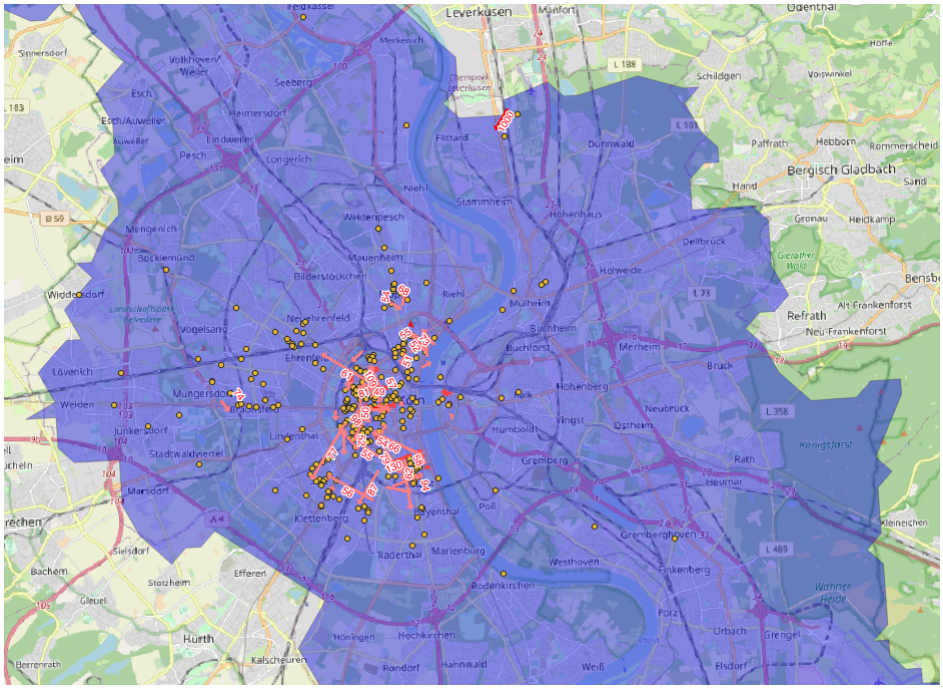


Figure 12 Container flows larger than 49

The two figures above show container flow volumes between the stores, implying a stochastic flow distribution. However, we get some properties from the graph for further analysis, namely in- and out-degrees of the nodes, but multiplied by the total number of edges so that the degrees can be institutively interpreted. The table below shows the five stores, which are the top five stores in terms of outbound check-outs. The vertex column shows the stores; the in-degree column indicates the incoming number of containers; the out-degree column shows the outgoing number of containers.

Table 3 Degree analysis of the graph

vertex	geometry	in_degree	out_degree
ebbeabd8-0201-4af9-b9fa-a997abcb6bfb	POINT (6.94467 50.93899)	5104.0	7795.0
a3541dbc-85f8-4ccd-8f5b-801ae99869b8	POINT (6.96179 50.94843)	7118.0	7448.0
791b56e7-ac51-432f-b75d-17d6556a42ae	POINT (6.99471 50.93822)	6552.0	6510.0
e20e6c15-6507-4427-8b68-e1d1f870b924	POINT (6.93986 50.94564)	4336.0	4915.0
74a627d5-9783-4c2b-87b1-f5aa3f617f19	POINT (6.96119 50.92230)	3068.0	4138.0

We have shown the flow of containers between the stores in this section. From the visual analysis of the graph on QGIS, any significant flow pattern cannot be detected. However, in- and out-degree properties of the stores have been found. This information will be utilized in the next section.

4.2 Spatial Autocorrelation Analysis of the Stores

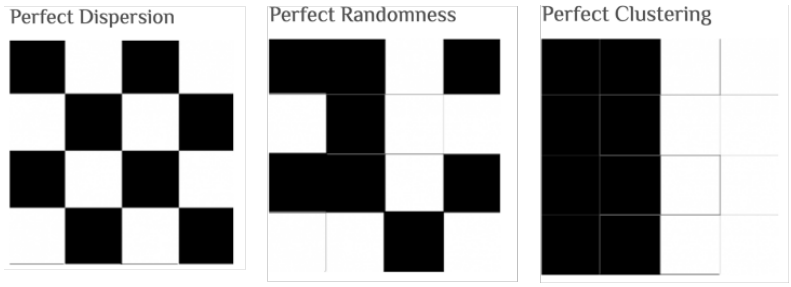
Although we could not detect any clear evidence from the previous graph analysis, we suspected some clustering in the city center in terms of inter-store container flows. We start this section with a hypothesis; the walking closeness of the other stores is correlated with the inter-flow sizes. We rephrase the hypothesis with simple words; consumers tend to return the containers to a nearby store instead of returning them to the original store from where they were taken away. Consumers might have different motivations for doing so, but this is beyond the scope of this section.

Autocorrelation is a similarity measurement of the nearby observations. Similarly, spatial autocorrelation measures the similarities of two values at spatial locations. Moran's I equation is given below. (rspatial.org, 2022).[2]

$$I = \frac{n}{\sum_{i=1}^{n}(y_i-\bar{y})^2} \frac{\sum_{i=1}^{n}\sum_{j=1}^{n} w_{ij}(y_i-\bar{y})(y_j-\bar{y})}{\sum_{i=1}^{n}\sum_{j=1}^{n} w_{ij}}$$ **Equation 4**

where y_i and y_j are the observed values,
$\bar{y}$ is the mean,
w_{ij} is the spatial weight between *i* and *j*.

Moran's I ranges between -1 and +1. The negative values refer a dispersion, while positive values refer a clustering. An example with extreme cases in terms of Moran's I calculation is shown in the figure below. The values are binary (white and black) for simplicity. The figure on the left would have a value of -1, the figure on the center would have the value of zero, and the right one would have +1. In addition to binary values, Moran's I can also handle continuous variables.



Source: **(Glen, 2016)**

Figure 13 Moran's I example

We have used Python's Pysal (2018) package to run Moran's I test. Pysal allows multiple spatial weight calculations. First, we need to identify the distance measure between the stores

[2] Readers can see (Michael J de Smith, 2018, S. 295-316) for a comprehensive tutorial about the spatial autocorrelation. Kassel University Statistics lecture notes (Statistics, 2022) give also a brief explanation for the spatial autocorrelation, spatial weight and Moran's I.

such that this choice will ultimately determine the spatial lag values. We decide on the '*Kernel'* weight, a continuous distance-based weight measurement because 'Kernel' weight can measure spatial similarity in a radius, enabling us to quantify the 'walking distance'. We assume that 300m is a reasonable walking distance between the stores. Next, we have selected a regenerated attributed total degree of the stores, namely the sum of in-degree and out-degree values of each store. To remind again, in or out-degrees refer the containers to be returned to a different store. Shortly, we test the correlation between spatial lag (weights) and total degrees of the stores. The figure below shows the correlation variance and fitting regression line followed by the same plot but with a logarithmic scale at the x-axis. Both figures are divided into four quadrants by the mean of the attributes. The letters H and L stand for High and Low, the first letter represents the attribute of interest (in our case; total degree of the store) axis on the scatter plot, and the second letter represents the spatial lag.

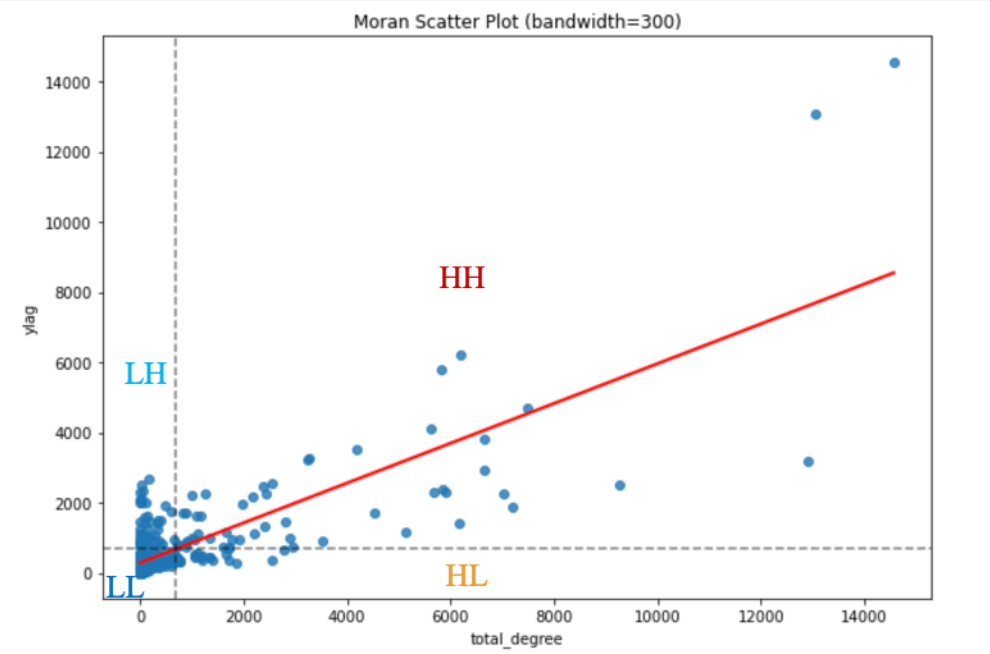


Figure 14 Correlation between total degrees and spatial lag

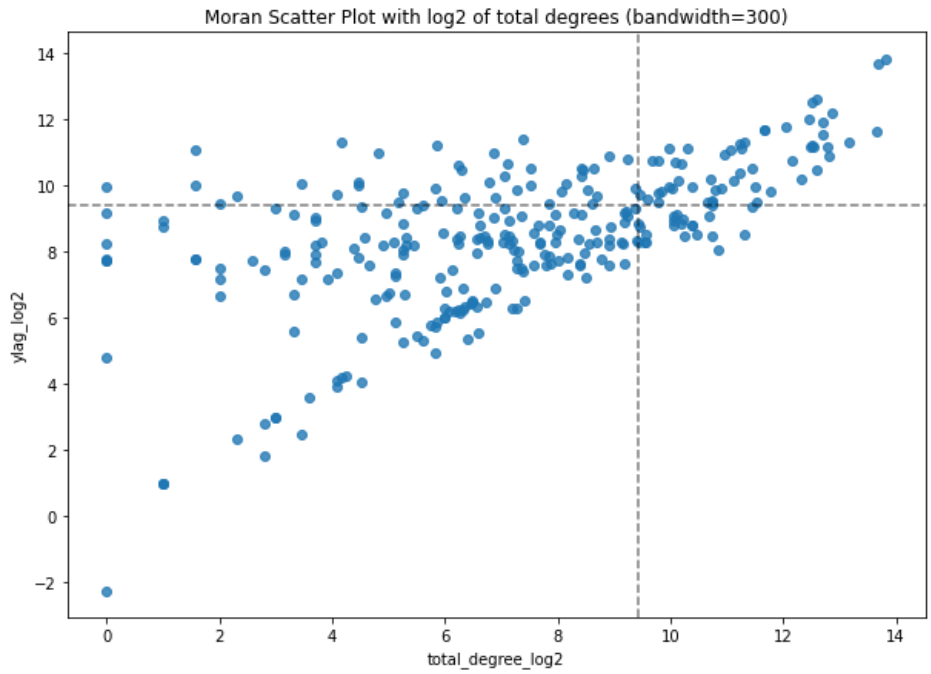


Figure 15 Correlation of total degrees and spatial lag with logarithm 2 scale at the x-axis

Pysal's Moran's I simulation runs a permutation test for the observations. It reshuffles the values among the stores and recalculates the Moran's I, and iterates this 999 times by default. Finally, we plot the distribution of these Moran's I resulting from those iterations. Shorter blue lines show the Moran's I values from these permutation tests; the red longer line is the actual Moran's I value. The p-value of this simulation is 0.035, which is traditionally statistically significant. In other words, there is clustering in our case with a value of Moran's I=0.57.

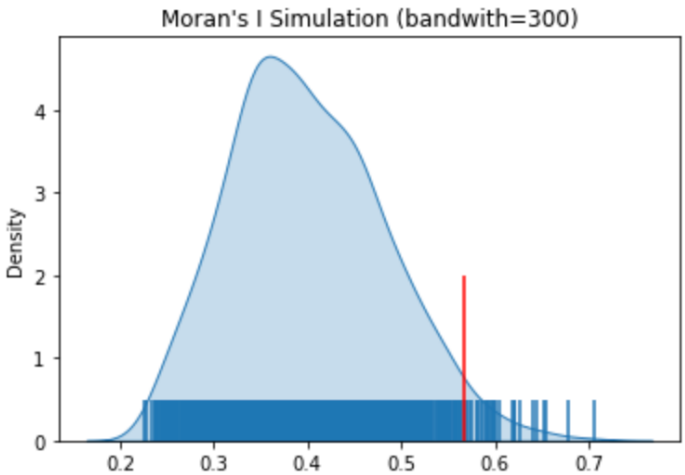


Figure 16 Moran's I simulation

Local spatial autocorrelation calculates the Moran's I for individual observations. The results of the local autocorrelation are shown below. The colors represent the four quadrants, while "ns" says the correlation is not statistically significant. Stores of interest in this plot are the HH ones because we are checking the hypothesis of if the nearby stores affect inter-store flows. The figure below shows that the red points (HH) do not form a clear cluster. Moreover, HH points are in close proximity to LL, and LH points which means the stores with low total degrees (inter-store flows) are also nearby.

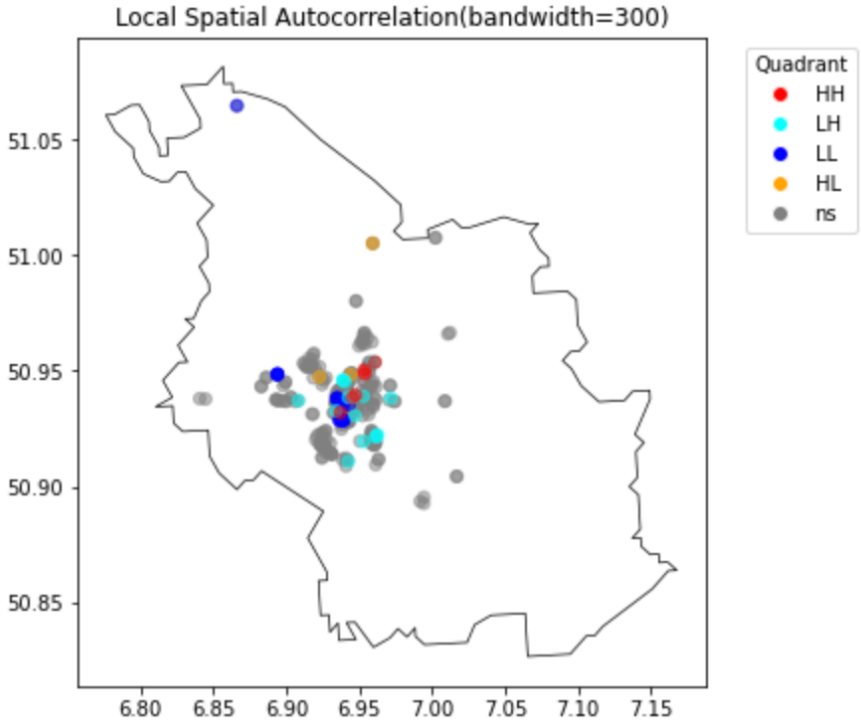


Figure 17 Local spatial autocorrelation

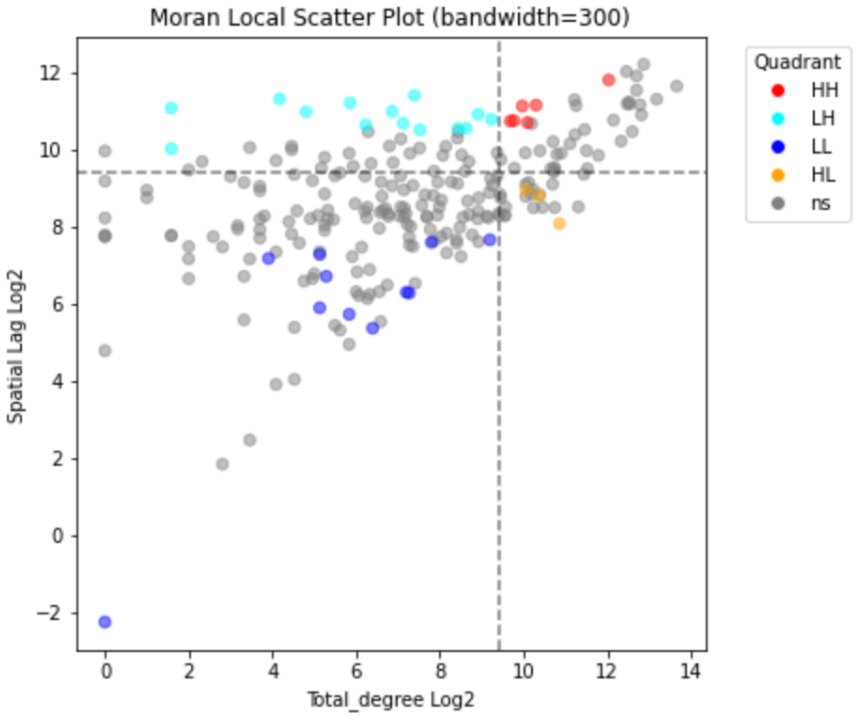


Figure 18 Moran's I Local Autocorrelation Scatter Plot

However, choosing 300m bandwidth for the spatial weight calculations was an assumption that people would tend to walk within this distance and return it to a different store. What if we choose 500m bandwidth? Similarly, we run the Moran's I with different bandwidths, and the results are given in the following table. Although the simulations with different bandwidths show a positive correlation, only the test with 300m bandwidth is statistically significant.

Table 4 Moran' I and p-values with different bandwidths

Bandwidth (m)	Moran's I	p-value
200	0.66	0.054
300	0.57	0.035
400	0.47	0.07
500	0.39	0.093
600	0.33	0.14
700	0.28	0.16

Chapter summary

In this chapter, we analyzed the spatial distribution of inter-store container flows. We answered if there is a spatial clustering concerning flow sizes.

Inter-store flow sizes have been selected as the attribute of interest. Alternatively, total transaction sizes (including self-loops and inter-store flows) can also be chosen. The null hypothesis would be accordingly modified; is there any spatial clustering regarding the transaction sizes of the stores? However, there is a strong correlation between the total size of transactions and the inter-store flow sizes. Indeed, inter-store flows are a fraction of total transaction sizes.

Lastly, kernel distance has been chosen since it addresses the null hypothesis at best. One improvement can be the usage of accurate walking path distances instead of Euclidean distances.

Even though the Moran's I tests indicate a positive correlation between the flow sizes and spatial lag, only one test width of 300m bandwidth is statistically significant among six tests. Moreover, the local autocorrelation analysis does not support the hypothesis we introduced at the beginning of the Chapter.

Chapter 5 Correlation Analysis with External Factors

We mainly analyzed the internal dependencies within the transactions so far. In this chapter, external factors will be analyzed to determine whether there is a correlation between the external factors and the transaction sizes.

The following factors will be analyzed in the following sections.

- Weather temperature
- Closeness to public transport (bus/train) stations
- Closeness to green fields
- Closeness to events
- Corona cases

5.1 Weather Data

As discovered in Figure 10 and Figure 9, we observe high check-outs on workdays, and noon time is the peak in check-outs. Note that most consumers return the containers on the same day; a similar pattern in a store will be soon seen in Figure 41. We infer from those plots that consumers buy the food with reusable containers instead of eating them on regular plates at the restaurant. Such behavior can be related to weather conditions. They can take their food with them during the lunch break and go to a nearby park while enjoying the fresh air instead of sitting in a restaurant. In this and the following sections, we test the hypothesis that daily transaction sizes correlate with the temperature and the distance to the green fields.

The daily weather data is imported from German Weather Service (Deuscher Wetterdienst, 2022).[3] The scatter plot below shows no indication of such a correlation between the temperature and the check-outs.

[3] Instead of hourly temperatures, we have gotten the daily data since the check-outs are clustered at noon time. For the same reason, we have taken also max daily temperature instead of mean temperature.

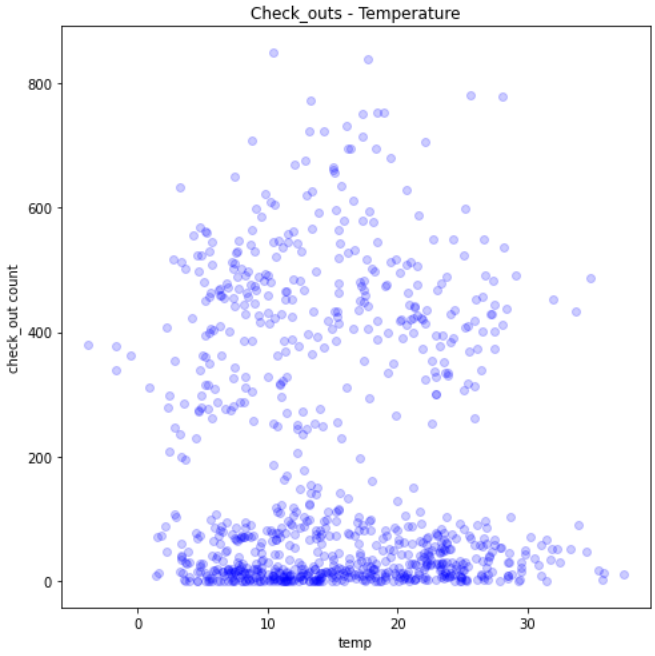


Figure 19 Temperature and check-out sizes

Though it can be easily interpretable on the plot above that there is no significant correlation between the check-out sizes and daily max temperatures, Pearson's correlation analysis also returns the value of -0.08.

5.2 Closeness to the green fields

As discussed above, closeness to green fields like parks or playgrounds can impact people's eating behaviors. The green field geo data is downloaded from Cologne Open Data (Offene Daten Köln, 2022) and processed in QGIS. The green field polygons are shown with the stores on the map below. The correlation results will be soon shown inTable 5.

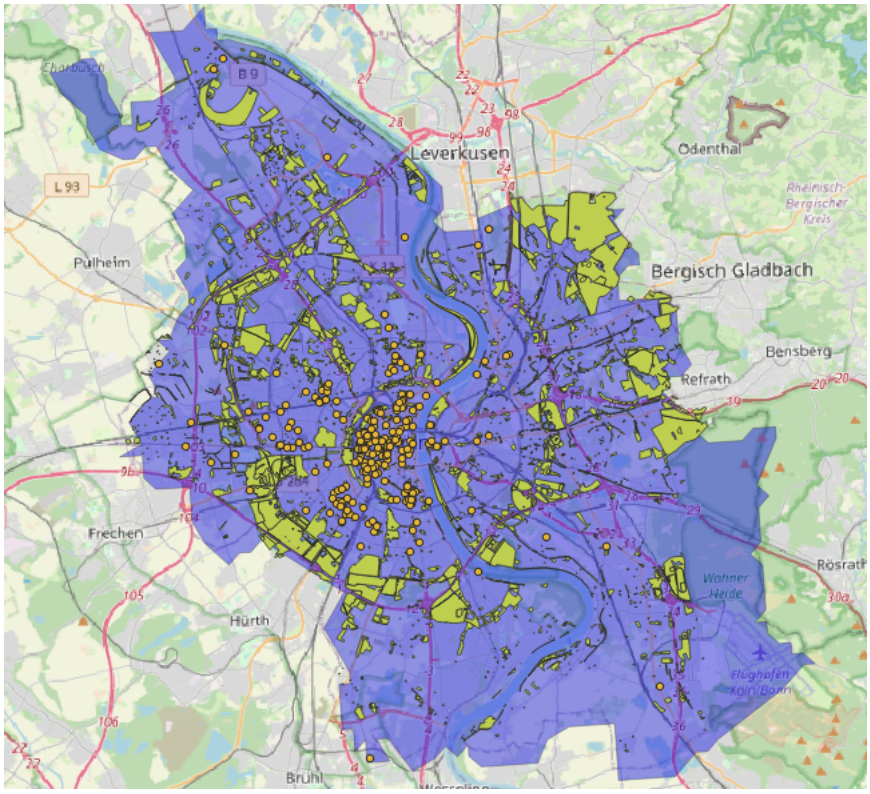


Figure 20 Green fields and the stores

5.3 Public Transport Data

This section questions if the closeness of stores to the public transport nodes is correlated with the transaction sizes since daily travel routine can impact people's eating behaviors. More clearly, they can visit a store to get food after or before taking a bus or train. The public transport network data in Cologne is downloaded from Verkehrsverbund Rhein-Sieg (2022). It contains bus, tram, and train networks, although only the bus network is shown below. The red lines represent bus lines; cyan points represent the bus stops. QGIS has been used to process the data and get the distances to the nearest public transport station.[4] Pearson's correlation analysis yields no significant correlation between the proximity to the public transport hubs and transaction sizes.

[4] "Distance to nearest hub (points)" function in Processing Tool of QGIS is used to get the distances to the closest public transport stop.

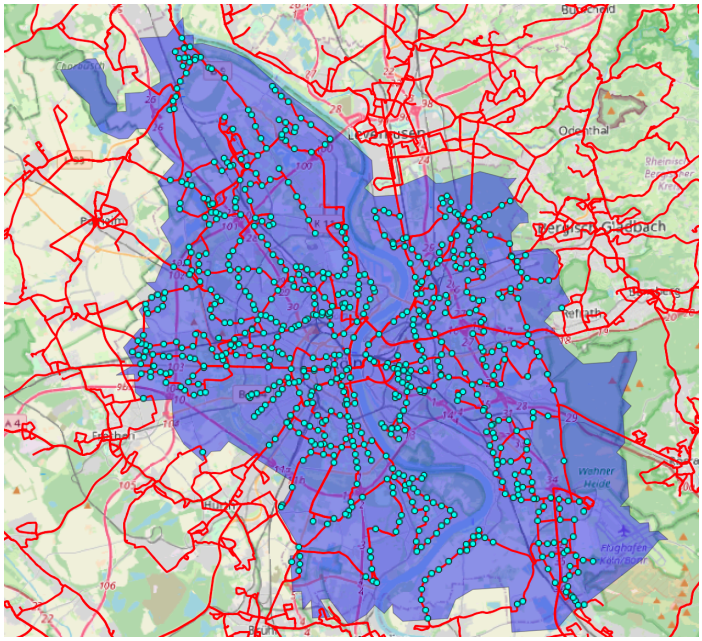


Figure 21 Bus lines and bus stops

Correlation analysis and corresponding p-values are given in the tables below. The column of check_ins represents the total number of returning containers; check_outs represent the outgoing containers, while in_degree and out_degree represent only the inter-store transactions, excluding looping transactions at the stores. Nearest_stop shows the distance between the store and the nearest public transport station. Greenfield_distance is the distance between the store and the nearest point of a green field polygon. As seen below, there is no significant correlation between green fields or bus/train stops with neither check-outs nor check-ins.

Table 5 Correlation Matrix

	check_ins	check_outs	in_degree	out_degree	nearest_stop	greenfield_distance
check_ins	1.000	-0.979	0.879	0.859	-0.019	-0.023
check_outs	-0.979	1.000	-0.845	-0.861	-0.002	0.005
in_degree	0.879	-0.845	1.000	0.974	-0.020	-0.034
out_degree	0.859	-0.861	0.974	1.000	0.007	-0.016
nearest_stop	-0.019	-0.002	-0.020	0.007	1.000	0.286
greenfield_distance	-0.023	0.005	-0.034	-0.016	0.286	1.000

Table 6 Corresponding p-values of the correlations

	check_ins	check_outs	in_degree	out_degree	nearest_stop	greenfield_distance
check_ins	0.000	0.000	0.000	0.000	0.715	0.661
check_outs	0.000	0.000	0.000	0.000	0.974	0.929
in_degree	0.000	0.000	0.000	0.000	0.701	0.515
out_degree	0.000	0.000	0.000	0.000	0.895	0.761
nearest_stop	0.715	0.974	0.701	0.895	0.000	0.000
greenfield_distance	0.661	0.929	0.515	0.761	0.000	0.000

5.4 Events in Cologne

We hypothesize that the nearby events increase the checks on that particular day that the event is held. The events are scrapped from the City of Cologne's official website until 31st October 2021 (Stadt-Köln, 2022). It is worth noting that the website's event calendar announces planned events with some relevant information, e.g., event types, location, and the date. It does not indicate the number of participants or whether the event took place as planned. We assume that event types of 'concert', 'fairs', ' talk', 'theater', `holiday program', and `dance' are the most attractive events concerning the number of people. We label the stores on a particular date if the event takes place in 500m.

The density distributions of the check-outs with and without event labels are given below. Surprisingly, the stores of interest record fewer check-outs on the days of events than the regular days. In contrast, we would expect a similar distribution but shifted to the right. Density and cumulative density plots are given below.

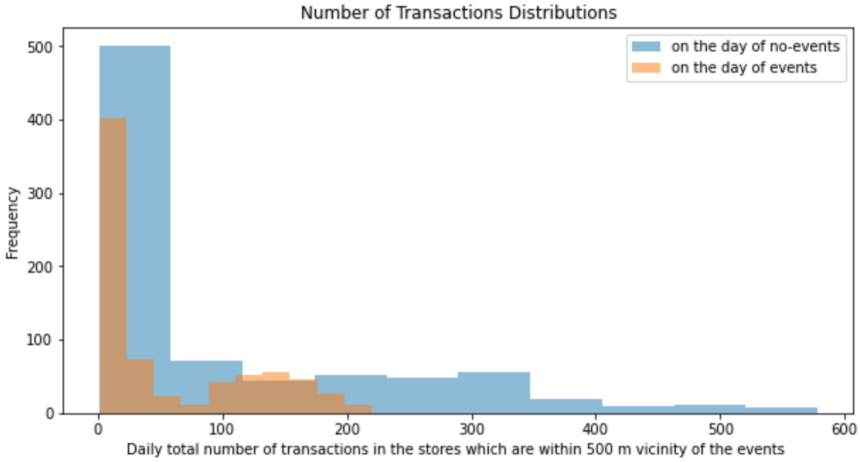


Figure 22 Distributions of the check-outs with and without event flag

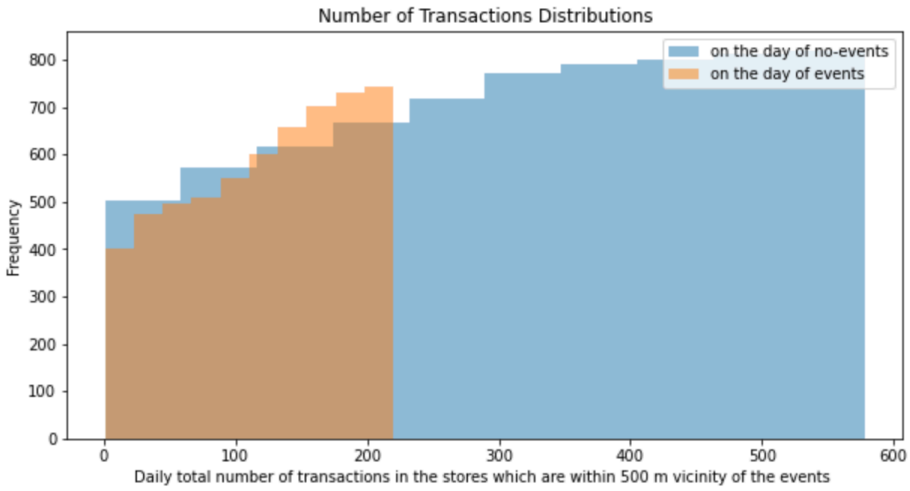


Figure 23 CDFs with and without event flag

Nevertheless, we perform a Kolmogorov-Smirnov test (National Institute of Standards and Technology, 2022) with the null hypothesis that the cumulative distribution function (CDF) of check-outs with event flags is smaller than the CDF of check-outs with non-event flags. More specifically, the orange CDF would not cross the blue CDF in the figure above and remains consistently below the blue CDF. After we run the test with the help of the Scipy library (2022), the null hypothesis is also rejected, as seen clearly in the CDF plot above.

5.5 Corona Cases

The corona cases and countermeasures had deeply affected people's travel and eating routines. The corona cases dataset is provided by Robert Koch Institute (2022) and shown with check-out sizes on the plot below. The sharp increases and decreases in check-outs are closely related to corona cases, as seen below. However, those sharp changes are primarily caused by the counter corona measures like lockdowns.

On 2nd November 2020, the federal government introduced a nationwide lockdown (Wiwo.de, 2022) (Wiwo, 2022) (ndr.de, 2022). Although the regulations changed from region to region slightly, the gastronomy stores (restaurants, etc.) were mainly allowed to provide pick-up food services while they were strictly forbidden for regular lunch or dinner services. That measure caused a continuous increase in check-outs until this measure was ruled out on 31st June 2021.

The measures were relaxed after October 2021, following significant progress on Corona vaccination. Meanwhile, local authorities also introduced many amending measures to react to corona cases in their regions. Consequently, we cannot quantify these measures and their effects precisely regarding the check-outs.

Nevertheless, 7-day corona cases in Cologne show an interesting correlation, as seen below on the plot.

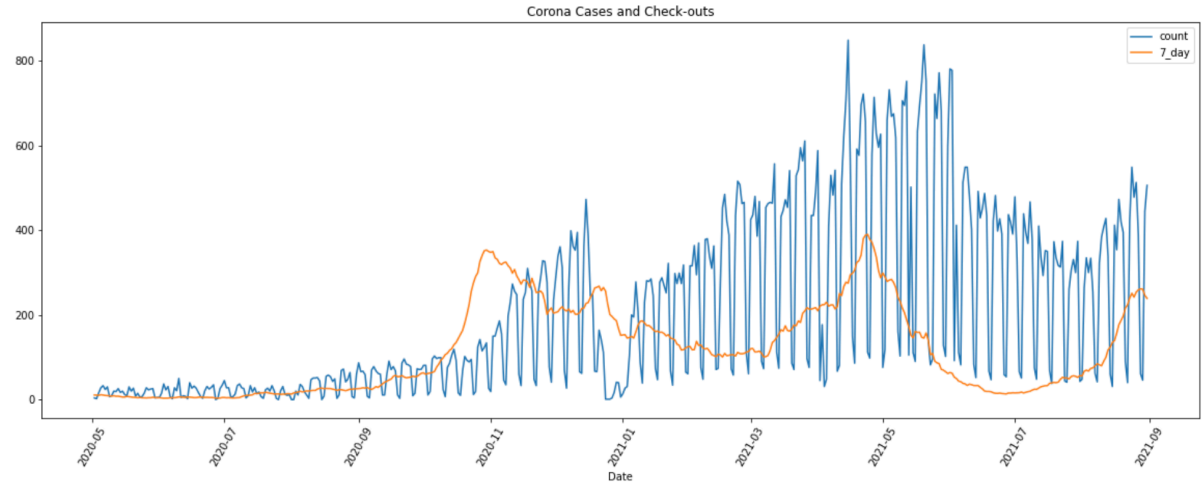


Figure 24 Corona Cases and Check-outs in Cologne

Pearson's correlation test also yields a significant positive correlation with a 0.36 correlation value. Besides that, we also implement linear regression analysis yields 0.13 R-squared, which means that the model can explain 13% of the variance. Below is the plot of residuals and predictions from the linear regression model.

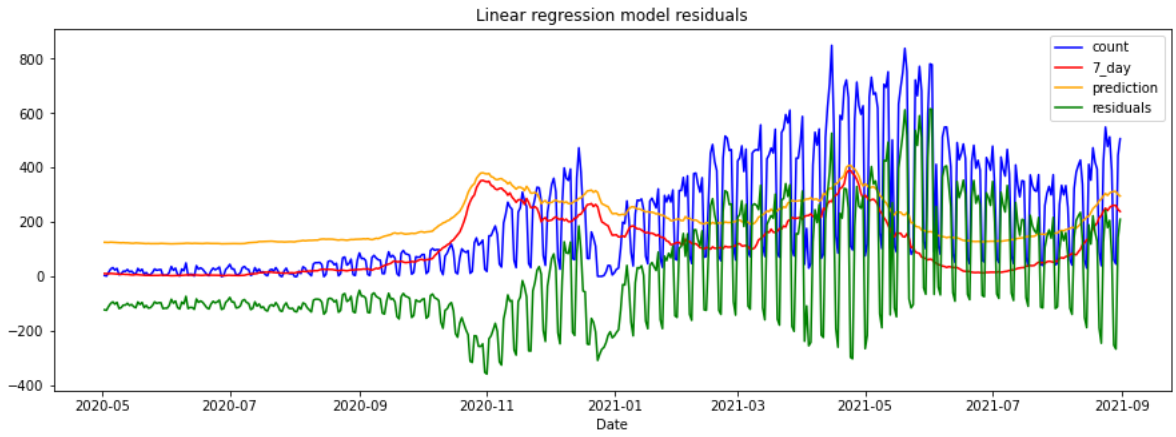


Figure 25 Linear regression model residuals

Chapter summary

In this chapter, we analyzed the correlations of the check-outs with the external factors. We discovered that only the corona cases are correlated with the check-outs, while temperature,

closeness to bus/train stops, closeness to the events, and closeness to green fields do not play a role in this context.

Closeness to bus or train stops cannot fully represent the flow volume at those stops. Unfortunately, at least in Cologne, we could not find any public data that gives such numbers. Similarly, closeness to green fields cannot capture all properties of a green field or a park. Availability of the banks, access routes, noise level, etc., can be more interesting to figure out the attractiveness of those places for the people who might mostly be looking for a calm place during the lunch break.

Events are scraped from the Cologne City calendar. The calendar does not imply any attendance number to those events even though it gives much other helpful information such as the event place, time, or type of the event. Other media sources, especially social media analysis, can reveal clues about the size of those events regarding attendance.

Lastly, we found out that the corona cases are statistically significant. However, the numbers are not relevant since 2021 autumn as most counter-corona restrictions are eased in parallel with the progress in the vaccination process.

Chapter 6 Time Series Analysis

This chapter is dedicated to a time series analysis of Vytal in Cologne. We will explore the essential components of the time series analysis and develop a prediction model at two echelons. The higher echelon is the aggregated transactions in Cologne that concerns both the local management in Cologne and the headquarters in Berlin. The lower echelon focuses on the transactions at an individual store that concerns mainly store management and Vytal's city management.

Management echelons are closely related to time horizons that we analyze and predict the demand. While next day prediction would suffice for the stores since the containers can be rebalanced before the rush hours on the same day, this prediction horizon would be too short for the rebalancing operations across Germany or procurement of new containers. Besides that, such a structure of echelons with time horizons indirectly dictates the aggregation level of the transactions. Therefore, we need to develop two separate models for those echelons since the time horizons differ. We start the time series analysis with the higher echelon and longer time horizon.

Time series analysis has some definitions and descriptions like 'level', 'trend', 'season', 'stationary', etc., that cannot be understood easily or intuitively. The readers might refer to (Hyndman & Athanasopoulos, 2021) and (Peixeiro, 2022) for those definitions.

6.1 Vytal's Time Series Analysis in Cologne

At this level, the company would be interested in

The company started its operations in 2019, and small check-out sizes throughout autumn 2020 can be seen in the plot below. Therefore, we name this period as 'the startup phase' in the following sections. With the boost of corona lockdowns, the check-outs increased rapidly in autumn 2020 and fluctuated further, at least for a year. Nevertheless, the Christmas breaks are very apparent in both years of 2020 and 2021.

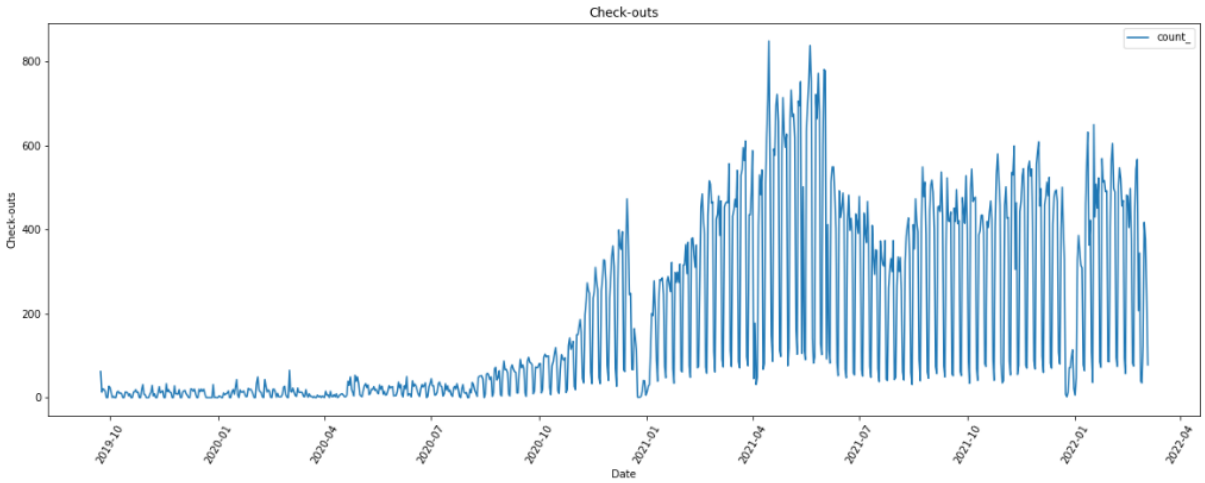


Figure 26 Vytal's check-out sizes

We discussed the effect of counter-corona measures like lockdowns above. Because of this corona disruption and startup phase, we cannot observe a yearly seasonality that would be more relevant for this management echelon. Therefore, we decided to deploy an ARIMA model to predict the demand in the following weeks. ARIMA stands for Autoregressive integrated moving average and will be soon briefly introduced in the following section. As discussed earlier, a time horizon of a few weeks would suffice either to rebalance the containers between the cities in Germany or even in Europe. Note that Vytal does not own a distribution network asset. Therefore, it outsources container shipments between cities unless they are directly procured from the suppliers.

6.2 Autoregressive Integrated Moving Average Forecasting

ARIMA, besides exponential smoothing, is one of the most used conventional models for time series forecasting. ARIMA aims to describe autocorrelations in the data with the condition that the series is stationary; it has neither trend nor seasonal fluctuations. (Hyndman & Athanasopoulos, 2021, S. 265). It mainly has two underlying components; autoregressive (AR) and moving average (MA), as ARIMA stands for Autoregressive Integrated Average Moving. Thus, we briefly introduce these components at first.

"A moving average model is denoted as MA(q) where q is the order. The model expresses the present value as a linear combination of the mean of the series μ, the present error term ϵ_t, and past error terms ϵ_{t-q}. The magnitude of the impact of past errors on the present value is quantified using a coefficient denoted as θ_q" (Peixeiro, 2022, S. 68).

The general expression of a MA(q) model is denoted below.

$$y_t = \mu + \epsilon_t + \theta_1\epsilon_{t-1} + \theta_2\epsilon_{t-2} + \cdots + \theta_q\epsilon_{t-q} \quad \textbf{Equation 5}$$

„An autoregressive process is denoted as an AR(p) process, where *p* is the order. In such a process, the present value y_t is a linear combination of a constant C, the present error term ϵ_t which is also white noise, and the past values of the series y_{t-p}. The magnitude of the influence of the past values on the present value is denoted as ϕ_p, which represent the coefficients of the AR(p) model“ (Peixeiro, 2022, S. 91).

$$y_t = C + \phi_1 y_{t-1} + \phi_2 y_{t-2} + \cdots + \phi_p y_{t-p} + \epsilon_t \quad \textbf{Equation 6}$$

One remaining component is differencing, which computes the differences between consecutive observations. “Differencing can help stabilize the mean of a time series by removing changes in the level of a time series, and therefore eliminating (or reducing) trend and seasonality” (Hyndman & Athanasopoulos, 2021, S. 267).

The basic ARIMA model is denoted by three parameters; ARIMA (*p, d, q*).
p= order of autoregressive part,
d= degree of differencing,
q=order of moving average part.

Although ARIMA has been extended as SARIMAX to capture seasonality and effects of exogenous predictors, we start with this simple model; then move forward to the more complex modeling.[5]

6.2.1 Data Preparation

The number of check-outs has fluctuated wildly due to the pandemic, as discussed in section 6.1. The other observation is the existence of weekly seasonality, as seen in Figure 26 and Figure 10. Vytal has a high demand on weekdays and low demand at the weekends. As stated above, we focus first on long-term planning; therefore, the granularity of the data can be

[5] In this Chapter, we will mainly use Python’s Statsmodels library for ARIMA forecasting (Statsmodels, 2022).

adjusted. Thus, we aggregate the check-outs weekly by taking the max of the daily sizes in every calendar week.

To note again, we decided to do analysis on a daily aggregate level, although intraday aggregate levels would also be interesting. In light of our findings so far, we can argue why we decided on the level of daily aggregation. Remember again, the check-outs and check-ins mostly happen at noon, as seen in Figure 9. Since the returning containers are almost simultaneously checked in at noon like the check-outs, the store is to have stocks, at least to meet this demand at noon. Those returning containers might be cleaned and reused for the afternoon demand. In this sense, demand at noon is the most interesting time slot for the intraday demand prediction. Since we do not know the internal processes of the stores, we cannot say how many minutes or hours later those returning containers would be serviced again after being washed. We assume the returning containers would be ready for the following day since they first need to be cleaned before the service in the worst case. That also means the returning containers (check-ins) allow the stores, at least partly, to replenish their stocks in addition to Vytal's rebalancing tours.

The second reason regarding the aggregation level is about the sampling noise, although we will show the variance in the data regarding the check-out sizes distribution in the following sections. Thus, a certain level of aggregation is necessary to detect the patterns in the data.

Therefore, we need to know the demand for a single day. Concerning the long-term planning, we do not need to know the demand every single day in a week since we are investigating whether the company needs to procure a new batch of containers or rebalance them between German cities. Thus, we need to determine a representative observation from a week that would help the company in this regard. The max of the week would be the best choice rather than the mean or sum of the week since the company aims to meet the daily demand fully by having no variable cost for maintaining the containers at the stores. Thus, the granularity of the data is finally set to weekly demand.

One data cleaning step has also been carried out to remove the Christmas weeks from the data. Consequently, this removal leaves the years with 50 weeks. Vytal has low demand these weeks, as seen clearly in Figure 27. Those idle weeks are not interesting because we want to predict the max demand levels. The simplified and aggregated check-out sizes are shown below.

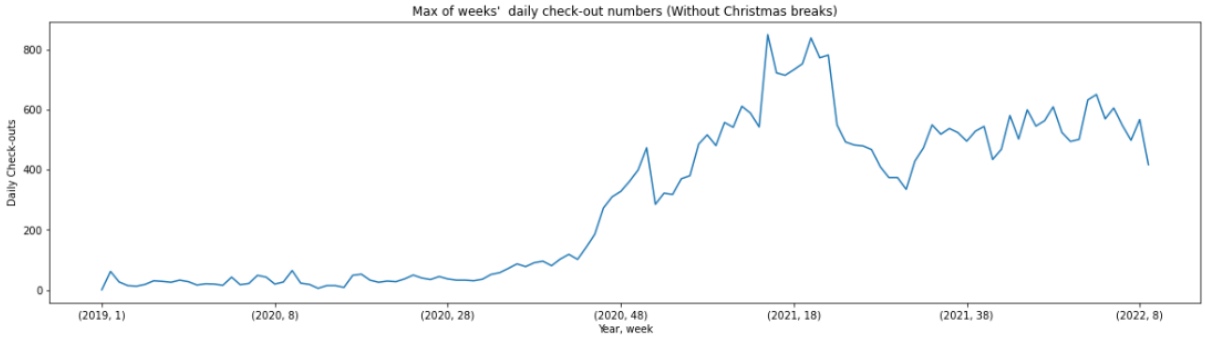


Figure 27 Weekly check-outs

6.2.2 ARIMA Model Implementation

As discussed above, we need to check the stationarity as a first step to start ARIMA forecasting. It is already apparent in the figure above that the data is not stationary. Nevertheless, the existence of non-stationarity is also justified with the Augmented Dickey-Fuller (ADF) test with the help of the Statsmodels library (Statsmodels, 2022)[6]. Thus, a differencing has to be applied. Apart from the weekly seasonality, which was already removed, we can expect a yearly seasonality. However, the plot above shows no evidence of it. As discussed in section 5.5, corona cases correlate with check-outs, implying that this regressor can mask the yearly seasonality. Removal of its contribution from the data can be thought of instantly. However, corona cases as a regressor in a linear model can explain only the portion of 0.13 of the variance, as discussed in section 5.5. Therefore, we assume the series has no yearly seasonality. Then, we need to implement only the differencing. Now the data is ready for ARIMA implementation.

However, before the differencing implementation, we briefly introduce the autocorrelation function (ACF) and partial autocorrelation function (PACF) plots. The points with the vertical lines show the degree of correlation between the lags. Blue zones on the plots represent the 0.95 confidence interval. As the figure below shows, there are strong correlations between the lags. The continuous smooth decrease also indicates a strong non-stationarity. ACF plots are

[6] Null hypothesis ($\mathbf{H_0}$): The time series is non-stationary. $\mathbf{H_A}$:The time series is stationary. Statsmodels ADF test returns the p-value=0.7 that means we fail to reject the null-hypothesis.

also helpful for detecting seasonality, which is not the case in this series. The partial autocorrelation shows the correlation between two observations that in-between lags do not explain. For example, in Figure 30. the third lag shows the correlation between 1st and 3rd observations that the 2nd lag does not explain. In other words, the accumulated correlation between the lags is removed in PACF.

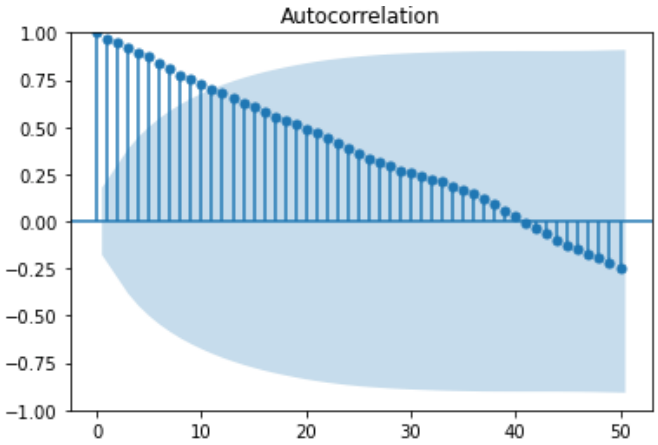


Figure 28 Autocorrelation of the check-outs

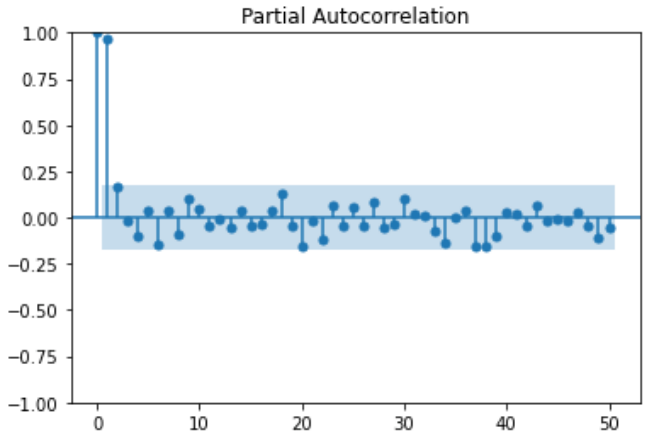


Figure 29 Partial Autocorrelation of the check-outs

The check-out sizes are shown below after the first differencing.

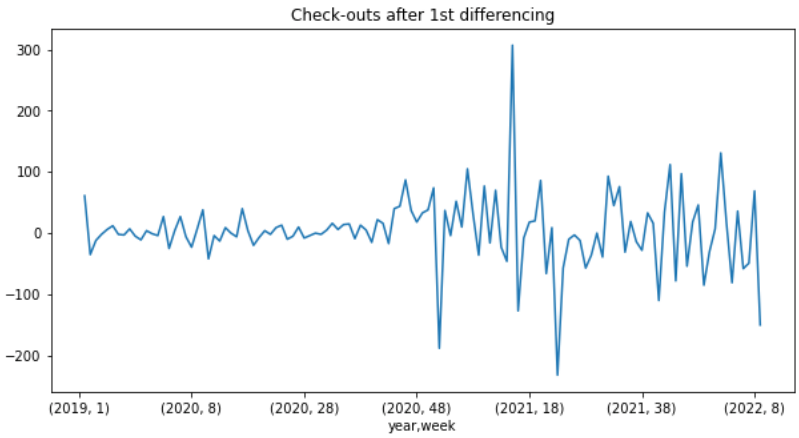


Figure 30 Check-outs after 1st differencing

As seen in the figure above, the series is now stationary. Additionally, the ADF test justifies the stationarity as well[7]. The corresponding ACF and PACF plots after the 1st differencing are given below. Regarding the ACF plot, the first lag is now negatively correlated rather than having multiple positive lags, as seen in Figure 29. The new ACF plot suggests MA(1), the equivalent of ARIMA(0,1,1) where differencing d=1 as we already found out. The PACF plot below also shows a significant correlation exceeding the confidence level zone. That also suggests AR(1) or equivalently ARIMA(1,1,0) model.

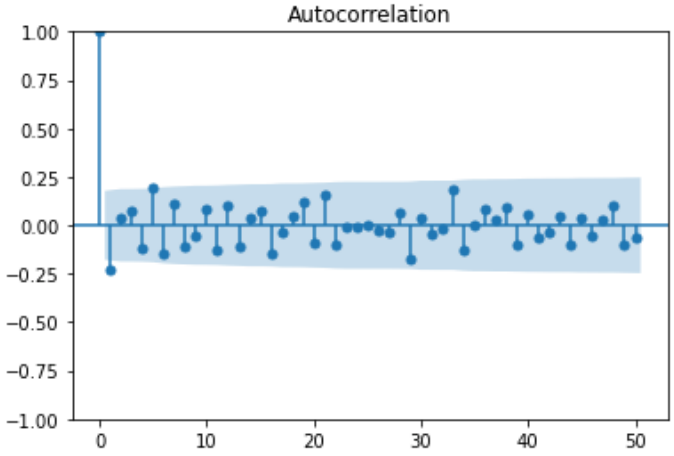


Figure 31 Autocorrelation after 1st differencing

[7] The ADF test after 1st differencing returns the p-value: 2.8187871855504284e-25. Thus, we reject the H_0.

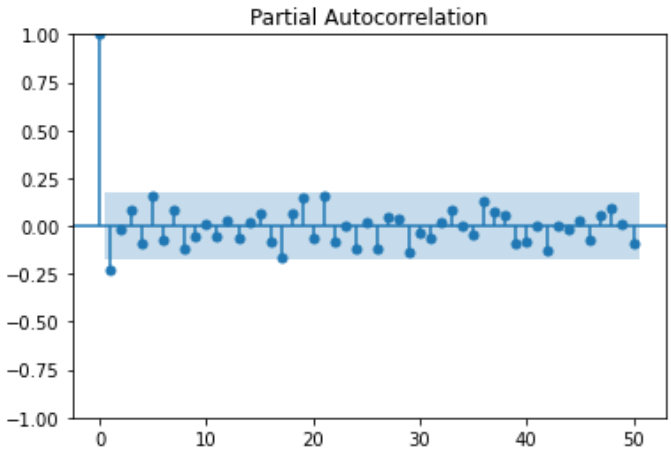


Figure 32 Partial autocorrelation after 1st differencing

Then, we can set the parameters of the model as ARIMA(1,1,1) following the analysis of the ACF and PACF plots. The model fitness is usually measured with Akaike's Information Criteria (AIC) or Bayesian Information Criteria (BIC) in ARIMA modeling (Hyndman & Athanasopoulos, 2021, S. 203).

Instead, we perform a grid search with a set of parameters and compare the fit of the models. Since we discussed and justified the degree of differencing is d=1, we do not search in this dimension. The reasonable orders for p and q are assumed up to 3 in this series. After reserving last 5 weeks as train set, we run the model ARIMA(p,1,q); p= (0,1,2), q = (0,1,2). According to AIC measurement, the ARIMA(2,1,2) is the best model. The diagnostics plot of the model is given below.

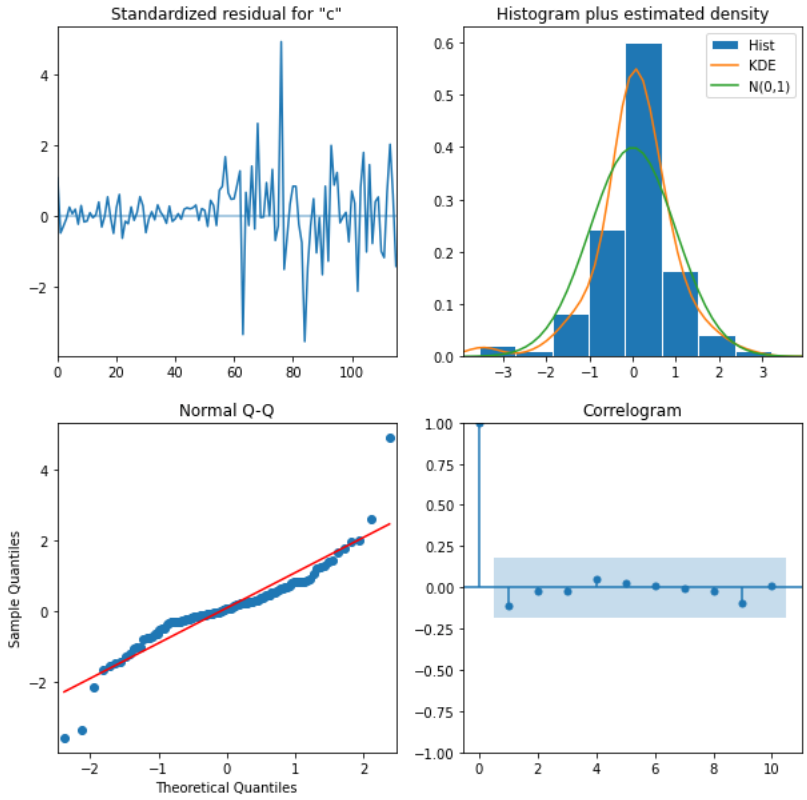


Figure 33 ARIMA(2,1,2) Diagnostics

In the figure above, the residual errors (top left) seem to fluctuate around zero. The histogram of the residuals on the top right also has a normal distribution. The bottom left Q-Q plot points out also a normal distribution of the residuals. Finally, the bottom right correlogram shows that the residual errors are not autocorrelated. Shortly, the model seems to have a good fit.

The model's predicted values with the actual ones are plotted below while it returns the mean absolute error (MAE) of 56.6 containers. Note that the model predicts the next five weeks in a row rather than rolling forecasting.

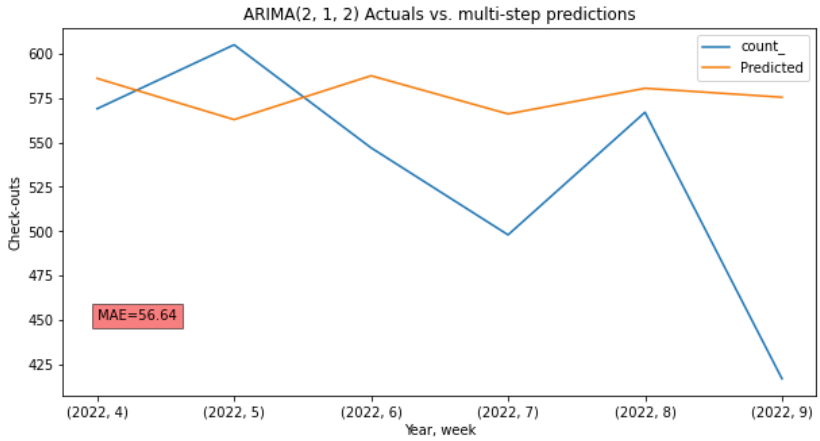


Figure 34 ARIMA(2,1,2) actual and predicted values

Due to the counter-corona measures, check-out sizes were disrupted extremely until autumn 2021, resulting in multi-step sharp decreases or increases in the series. That led to an ultimately higher order of moving averages and autoregressive.

Next, in the following section, we improve the model with an exogenous predictor, namely corona cases.

6.2.3 ARIMA modeling with corona cases

In the previous section, we discussed the possible contribution of corona cases in the model after finding a correlation between the check-outs and the corona cases. In this section, we will test to determine whether corona cases, as an exogenous regressor, in the ARIMA model is statistically significant. Corona cases are only relevant between March 2020 and Autumn 2021, as discussed in section 5.5.

Finally, the series is stripped after 1st October 2021, then a grid search for the model with the regressor of corona cases has been executed. ARIMA parameters are to d=1, q={0,1,2} and, p={0,1,2}. The coefficients and the p-values of the corona cases regressor are given below in Table 7 for the ARIMA model with different parameters. AIC column shows the model's fitness; lower values are better regarding the model's fitness. The coefficients are around 0.3, and p-values oscillate around 0.05. Four out of nine models have a statistically significant corona regressor.

Table 7 ARIMA models with the corona cases regressor

order	corona_p_value	corona_coeff	AIC
(0, 1, 0)	0.028	0.3230	876.725190
(1, 1, 0)	0.047	0.3114	875.763232
(2, 1, 0)	0.060	0.3091	877.361755
(0, 1, 1)	0.041	0.3147	876.212028
(1, 1, 1)	0.051	0.3107	877.523763
(2, 1, 1)	0.068	0.3067	879.333982
(0, 1, 2)	0.070	0.3064	877.311024
(1, 1, 2)	0.032	0.3270	878.303073
(2, 1, 2)	0.055	0.2964	873.741570

The table above casts doubt on the contribution of the corona cases regarding ARIMA modeling. Note that adding a regressor can be complicated if we predict multiple future steps since we also need to predict the regressor in those steps. Furthermore, as noted previously, corona cases are irrelevant after autumn 2021. Thus, it will not help predict the demand for the check-out in Vytal's case. In this regard, it seems unnecessary to include corona cases in the model.

Finally, we decided to use ARIMA(2,1,2) without any exogenous regressor. The following plot shows the predictions of a rolling forecast in recent weeks. The MAE is 76, which is worse than the multi-step prediction shown in Figure 34.

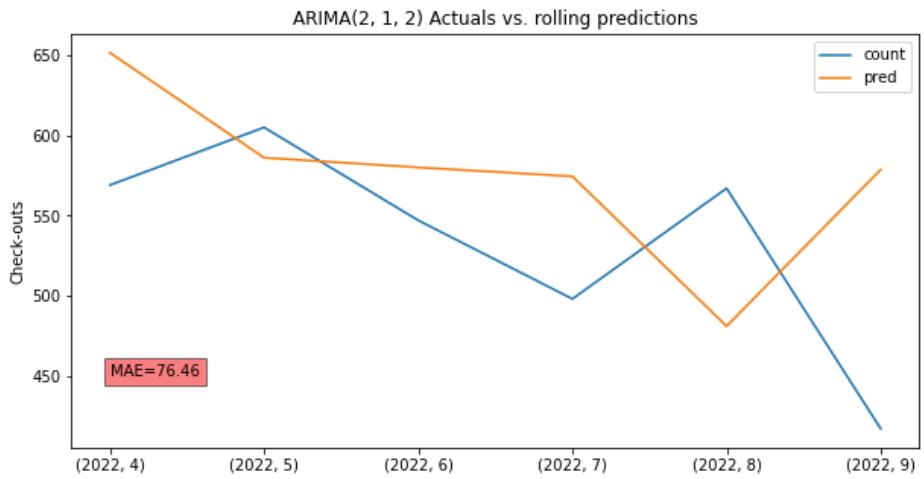


Figure 35 Rolling validation

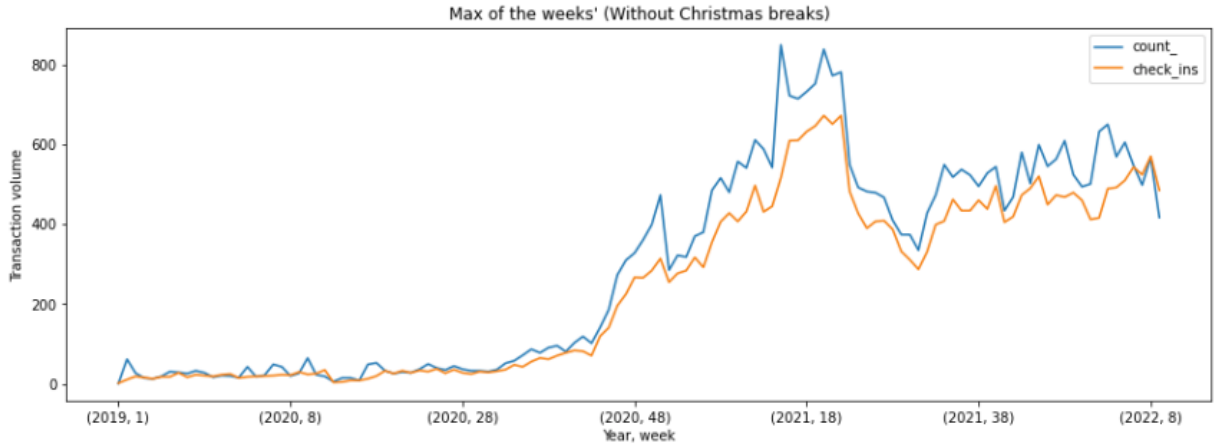
As discussed above, due to the counter-corona measures, check-out sizes were disrupted extremely until autumn 2021, resulting in sharp decreases or increases in the series. That led to an ultimately higher order of moving averages and autoregressive in the model fitting phase. However, this pattern is not representative anymore since the series seems to be stable after autumn 2021, as seen in Figure 27. Therefore, the model can be fitted on this recent portion of the series after 1st October 2021. A grid search returns ARIMA(0,0,0) on this portion which indicates the series is simply noise which can also be interpreted in Figure 27. In other words, the series would oscillate around the mean. The multi-step forecasting of ARIMA(2,1,2) tends to converge to the mean as the number of prediction steps increases (Hyndman & Athanasopoulos, 2021, S. 281) since the series is already stationary on this portion. That is also why multi-step forecasting outperformed the rolling forecast.

This section analyzed the aggregate demand in Cologne. Although weekly seasonality was evident in the series, it disappeared due to the aggregation in this section. The annual

seasonality is mainly masked by the corona cases. In addition, the two-year duration of the series is too short for the analysis of annual seasonality. However, a predictive model was built in Prophet, a forecasting software presented in the next section, that accounts for both corona cases and annual seasonality (see Appendix B).

6.3 Interaction between Check-ins and Check-outs

We had already found a strong correlation between check-ins and check-outs, as seen in Table 5. In this section, we investigate the bilateral interaction between check-ins and check-outs. The series of Check-ins is shown below with the check-outs. The plot also shows the interaction between the two series.



Vector autoregression (VAR) analysis allows us to predict the interacting time series. For VAR explanations, readers can see Chatfield (2004, S. 241-254) or Schumway&Stoffer (2016, S. 273-287). Nevertheless, the equation for one lag, VAR(1), is given below, reflecting the similarities with Equation 6.

$$\begin{bmatrix} y_{1,t} \\ y_{2,t} \end{bmatrix} = \begin{bmatrix} C_1 \\ C_2 \end{bmatrix} + \begin{bmatrix} \phi_{1,1} & \phi_{1,2} \\ \phi_{2,1} & \phi_{2,2} \end{bmatrix} \begin{bmatrix} y_{1,t-1} \\ y_{2,t-1} \end{bmatrix} + \begin{bmatrix} \epsilon_{1,t} \\ \epsilon_{2,t} \end{bmatrix} \qquad \text{Equation 7}$$

Statsmodels package provides VAR(p) method, with the p argument indicating the number of lags (Statsmodels, 2022). We passed those two time series, check-ins and check-outs shown in the figure above, into the VAR(1) model. As we did in the previous modeling analysis, we applied a grid search to find the best lag number that returned VAR(1) as the best one. Next, we apply Statsmodels's Granger causality (Statsmodels, 2022) test to check whether the series bilaterally interact with each other. As the hypothesis of "The series of check-outs causes check-ins" fails, we declined to deploy VAR in favor of an ARIMAX (ARIMA with an exogenous variable) model since one-directional interaction infers that check-outs series can be

a predictor to predict check-ins. The SARIMAX(1,1,0) is built with the steps similar to section 6.2. However, the model analysis with its whole steps is also given in Appendix A. Below are the model's fitted values and the prediction on the test set.

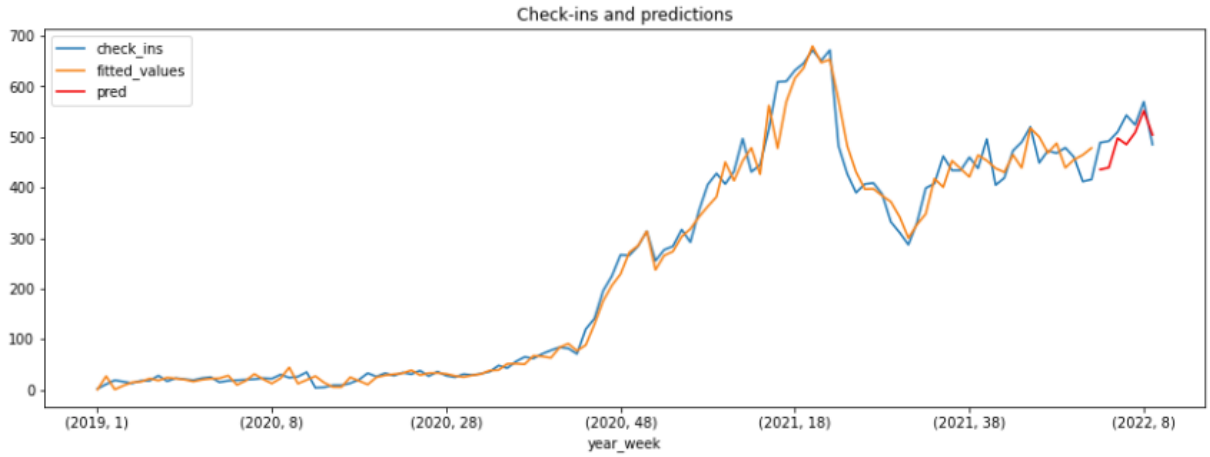


Figure 36 Check-ins and predictions with SARIMAX(1,1,0)

In this section, we proved that there is no bilateral interaction between check-ins and check-outs. However, check-outs can be used as a predictor for forecasting the check-ins. Since this thesis aims to develop a demand forecast model rather than an inventory forecast, we do not continue analyzing the check-ins in the next section, where we will develop a prediction model for the demand at an individual store.

6.4 Time Series Analysis at a Store

Vytal has more than 300 partner stores in Cologne. In the previous chapters, we have seen the density of hourly, daily, and weekly distributions of transactions. In this section, we analyze one of these stores. Most stores have lots of zero check-outs, as seen in the figure below that shows the boxplot distribution of randomly selected stores.

Figure 37 Boxplots of randomly selected stores

Since we want to show some patterns in the series, we selected the top 10 stores with the highest medians. The figure below shows the boxplot daily check-out size distributions of the top 10 stores according to the median of the check-outs. Furthermore, check-out or check-in sizes are aggregated by counting the unique consumers in this section to reduce variance.

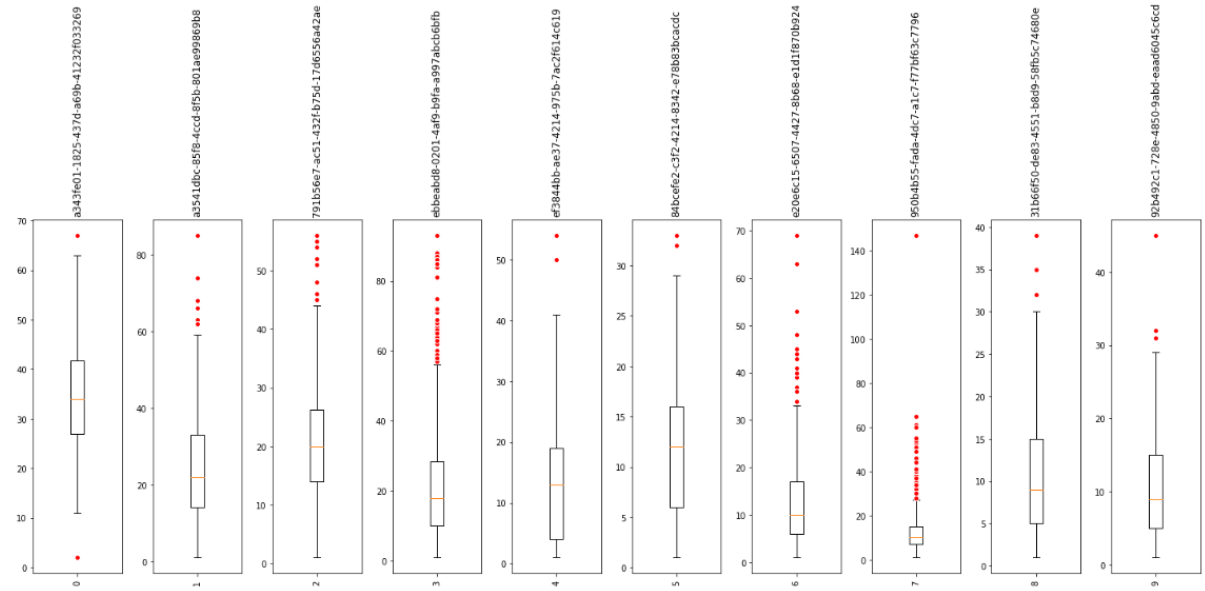


Figure 38 Top 10 Stores with highest medians regard check-outs

6.4.1 Time Series Analysis of the Store named “Polizeipräsidium Köln – Betriebsgastronomie”

For the considerations mentioned above, we selected the store labeled as “2” since it has a relatively more normal distribution. The selected store’s name is Polizeipräsidium Köln – Betriebsgastronomie and the location is shown on the map below.

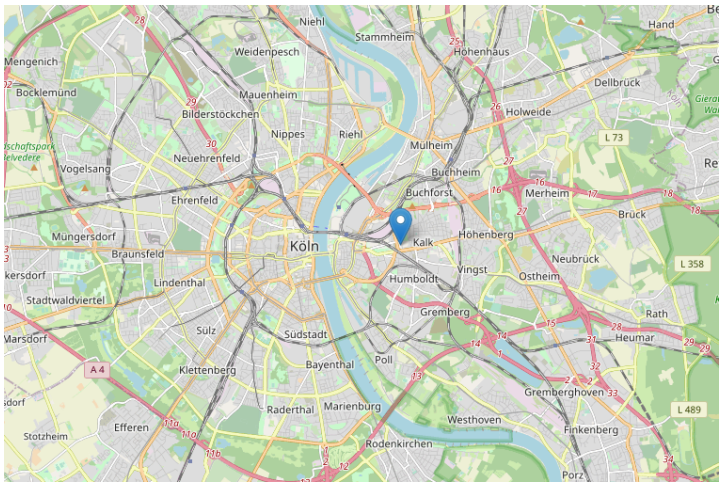


Figure 39 The location of the selected store

We will provide basic plots regarding time series starting from hourly to monthly distribution of the transactions. The following plot shows the hourly distribution of check-outs and check-ins at the store. As seen, the noon hours are the busiest. Although there is a strong correlation (0.74) between check-outs and check-ins on a daily basis, we will analyze the check-outs by assuming that the check-ins have no effect on check-outs.

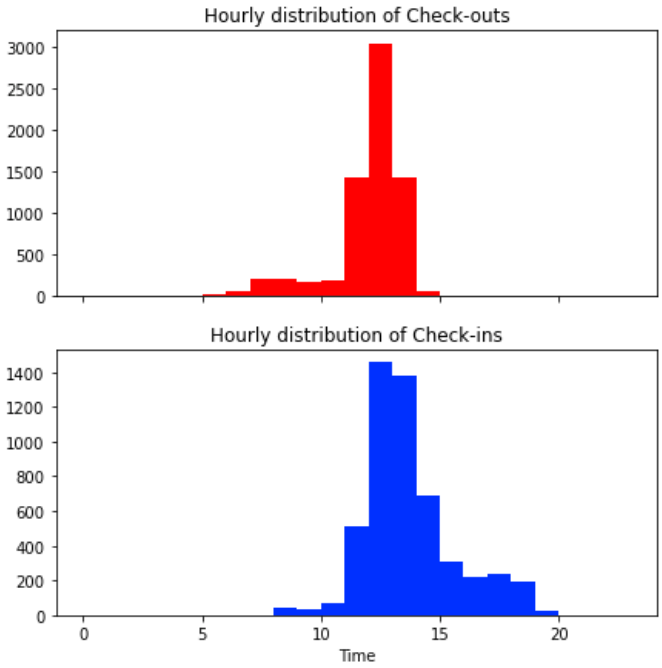


Figure 40 The distributions of check-outs and check-ins

Most customers of this store return the containers the same day after purchase, as seen in the figure below.

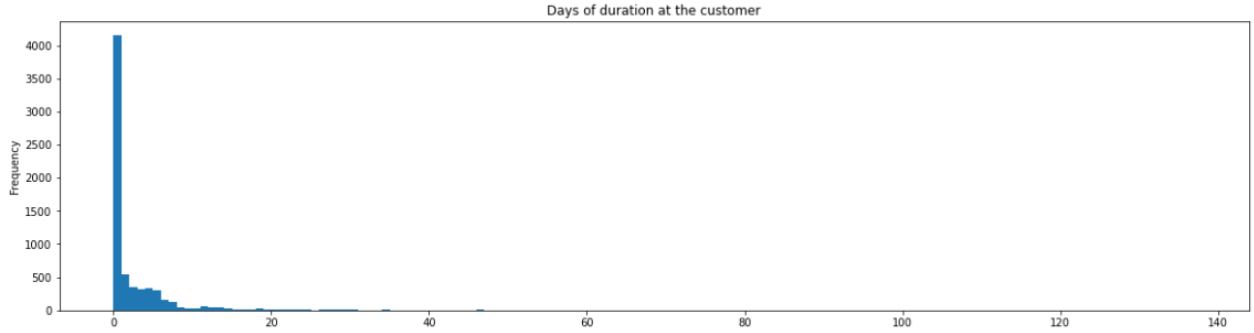


Figure 41 Days of duration of containers at consumers

The check-out sizes in the recent months are seen in the figure below. The weekly seasonality and Christmas break are evident.

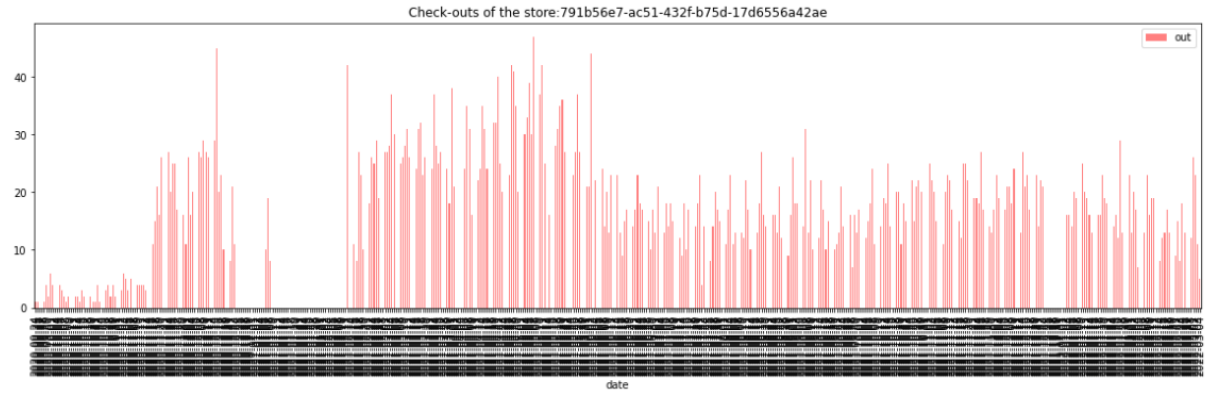


Figure 42 Daily check-outs

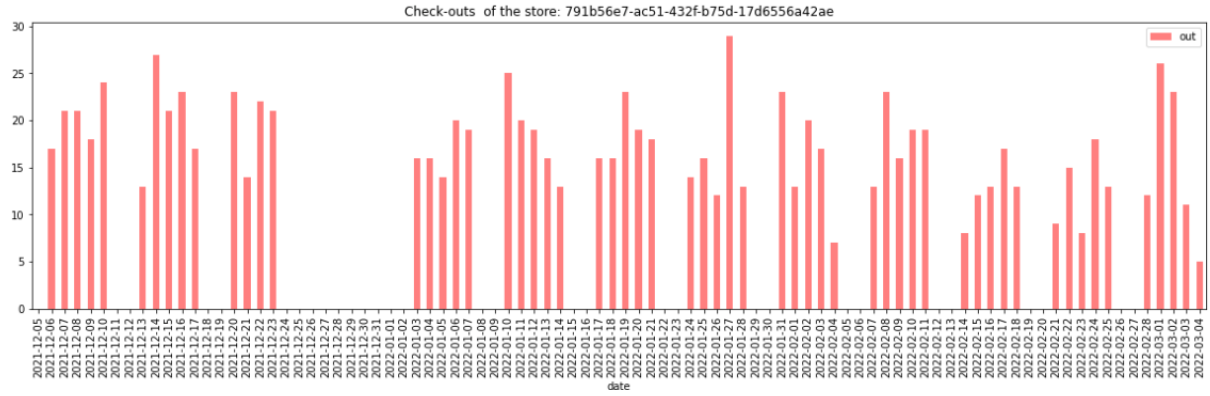


Figure 43 Daily check-outs of the store in recent months

Nevertheless, the ADF plot also confirms the weekly seasonality as 7-day lag periods are evident in the figure below.

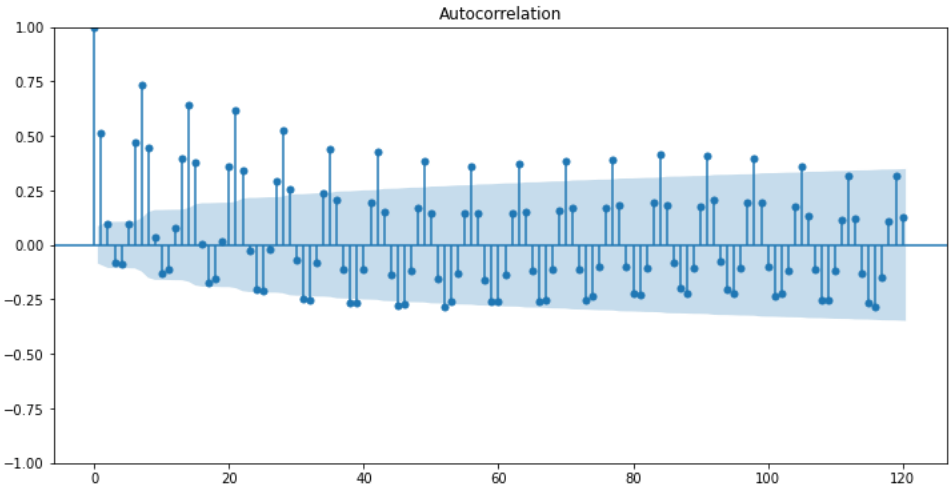


Figure 44 ACF Plot of the store

As discussed at the beginning of this chapter, the daily demand is relevant for the company. Therefore, we have taken the max daily demand for the weeks or months instead of weekly or monthly aggregation as we continue to decrease the granularity of the series. The plot below shows the max daily demand of every year-month pair. Neither the figure below nor Figure 42 indicates yearly seasonality, though the series is less than two years.

Figure 45 Max check-outs by year-month

6.4.2 Forecasting with Facebook Prophet

We used ARIMA modeling in the previous section to demonstrate long-term forecasting. Besides ARIMA, there are many other tools to forecast the time series, especially Machine Learning promises to enrich the time series analysis (Lazzeri, 2021). Prophet is one of those tools modeling more compactly (Prophet, 2022). This section will use Prophet to demonstrate short-term forecasting for a selected store.

Facebook analytics team produced an open-source software package in STAN, and R and Python APIs are also available. It is a compact forecasting solution for time series while providing parameters for complex multiple time series and exogenous predictors. Besides those, it allows us to model also the holidays or other special days.

Prophet's basic components are the trend, seasonality (yearly, weekly), holidays, and error (Rafferty, 2021).

The formula behind Prophet model is:

$$\mathbf{y}(t) = g_{(t)} + s_{(t)} + h_{(t)} + e_{(t)} \qquad \textbf{Equation 8}$$

$\mathbf{g}_{(t)}$: trend models non-periodic changes (i.e. growth over time),

$\mathbf{s}_{(t)}$: seasonality presents periodic changes (i.e. weekly, monthly, yearly),

$\mathbf{h}_{(t)}$: effects of holidays (on potentially irregular schedules ≥ 1 day(s))

$\mathbf{e}_{(t)}$: noise (Robson, 2019).

Trend component allows using linear or logistic function besides a flat function without a slope. The trend is the core component of the Prophet model.

The seasonal component allows multi-seasonal effect modeling. In the equation below, P denotes the length of the seasonal period in observations, while N denotes the number of terms in the Fourier series. Fourier series is controlled by the fourier-order in Prophet (Rafferty, 2021, S. 73-76).

‘Additive’ and ‘multiplicative' are critical parameters in Prophet, especially when parameterizing seasonal effects. In simple words, it is said to be additive if the effect of the season is constant from year to year or week to week, etc. On the other hand, if the size of the seasonal effect is proportional to the mean, then the seasonal effect is said to be multiplicative (Chatfield, 2004, S. 14).

$$s(t) = \sum_{n=1}^{N} \left(a_n \cos\left(\frac{2\pi nt}{P}\right) + b_n \sin\left(\frac{2\pi nt}{P}\right) \right)$$

Equation 9

The last remaining primary hyper parameter is the prior scale of the component, e.g., seasonality, regressors, etc. Reducing this number applies more regularization that would decrease the contribution of the particular component like the seasonality. That also means the prior scale is also directly related to the model’s fitness regarding over or under-fitting (Rafferty, 2021, S. 86-94).

Model implementation

The model is given in Appendix A. The hyper-parameters have been manually experimented, although a grid search can also be done while aware of the time-consuming computation time. The forecast of the model on the training set is shown below. Note that Christmas breaks are removed since the store has no check-outs these days, as seen in Figure 42 and Figure 43. As discussed above, the main component of the model is the trend. Those trends are fitted between the changepoints, shown with vertical red dashed lines in the figure below. Prophet identifies 25 changepoints on the first 0.8 portions of the training set by default. Then, selects the best changepoints after an internal optimization process. The manual setting of those changepoints would be incredibly relevant when we search the yearly seasonality. However, there is no proof

of any yearly seasonality in this series[8]. From the visual analysis of the plot below, changepoints selected by the model fit the series as major trends are overtly captured.

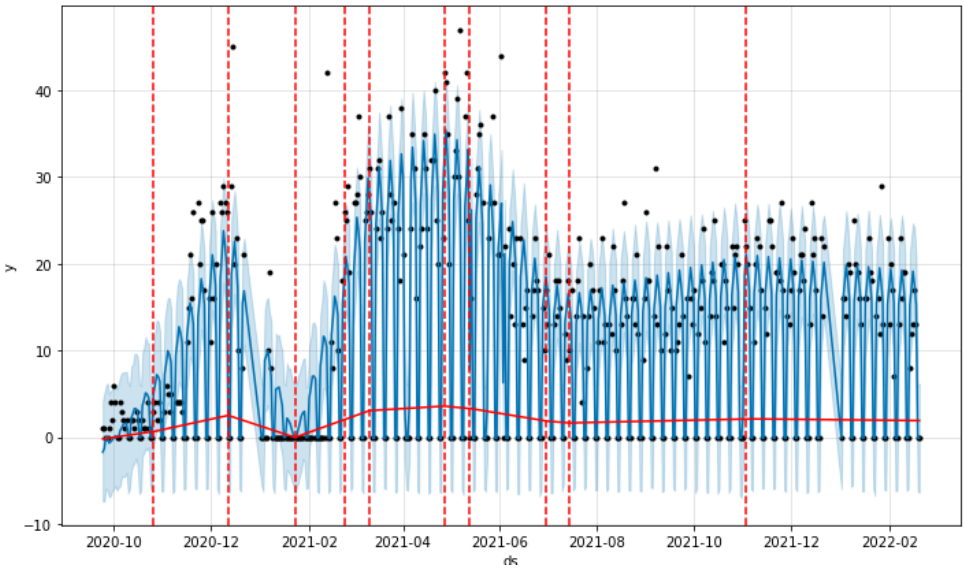


Figure 46 Prophet model forecast

The plot above apparently shows also the variance in the series. The black points are the observations, the blue line is the prediction, and the shadowed area is the confidence interval. Since the store has no check-outs at the weekend, some points are laid on the x-axis. On the other hand, some observations are well above the confidence interval.

With continuing variance analysis, we create the following plot. The red points on the plot show the observations one standard deviation away from the mean of MA(10). The red points at the lower side are mostly zeros, which are not interesting for the prediction analysis.

[8] The pattern of the series is similar to pattern of Vytal's aggregated data that discussed in 6.1 and Appendix B. See these sections for the discussions regarding yearly seasonality.

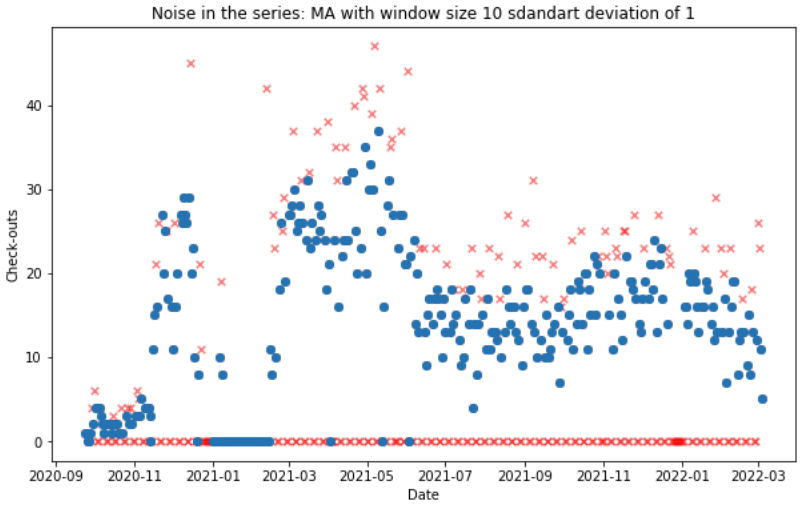


Figure 47 Noise in the series

The diagnostics of the Prophet model are given below. The trends between those changepoints are shown on the top. The second plot shows the weekly seasonality. Instead of using Prophet's holiday component, we defined holidays and weekends together as a regressor since the store has a similar pattern on those days, namely no check-outs on weekends or holidays[9]. Both weekly seasonality and the regressor of 'is_holiday' are modeled as multiplicative components that might be understood from the y-axis scaling with percentages.

[9] Prophet asses every holiday type individually. Since we have less than two years series, small number of these particular observations can lead over fitting of the, 'holidays' component of the model on those particular days. Holiday component is also a regressor in the model (Rafferty, 2021, S. 157).

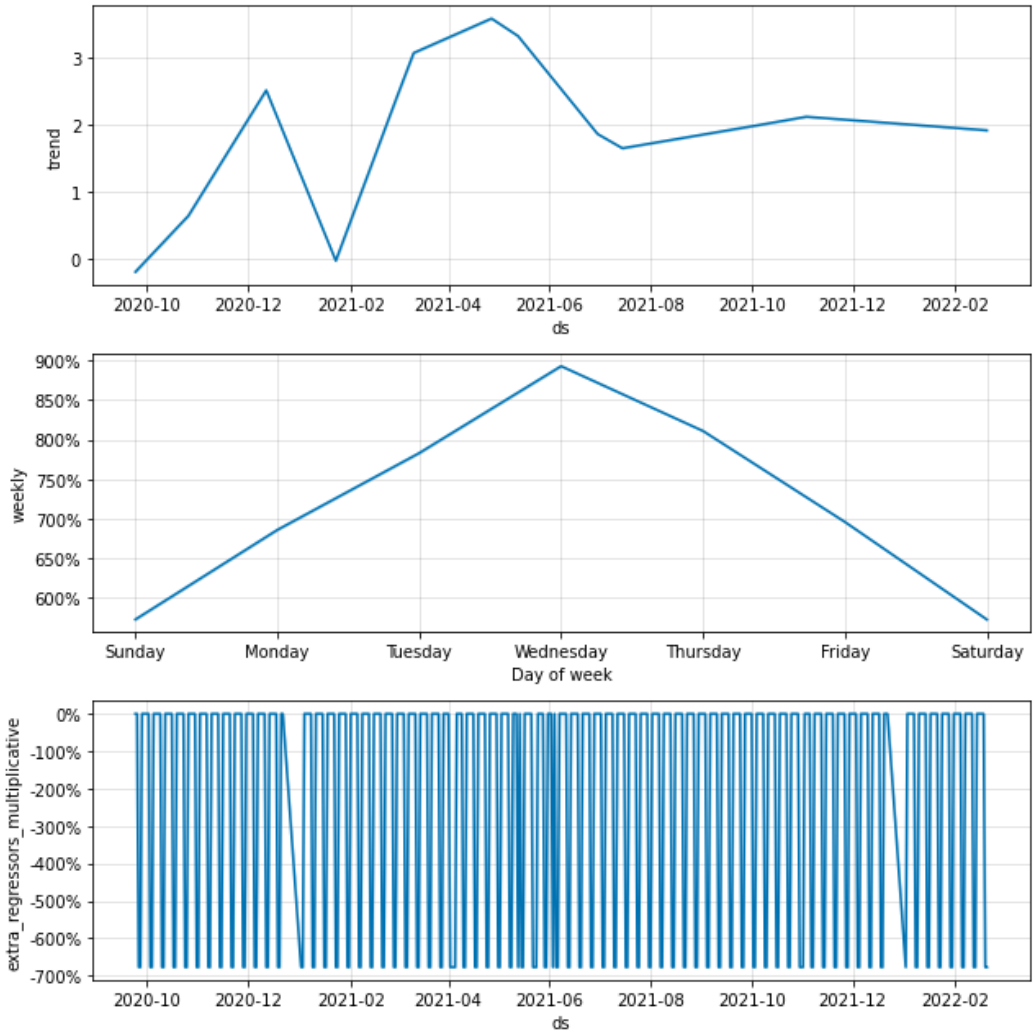


Figure 48 Diagnostics of the Prophet model for the selected store

The model is tested in the most recent two weeks as well. As seen in the figure below, the model captures the weekend perfectly while the weekdays have noise, as discussed above[10].

A cross-validation process for rolling prediction horizons for the model is also given in Appendix A.

[10] Since, the company's ultimate goal is to develop models to optimize the rebalancing effort between the stores, a service level model can also serve as a straight forward solution, which is a basic component of inventory optimization.

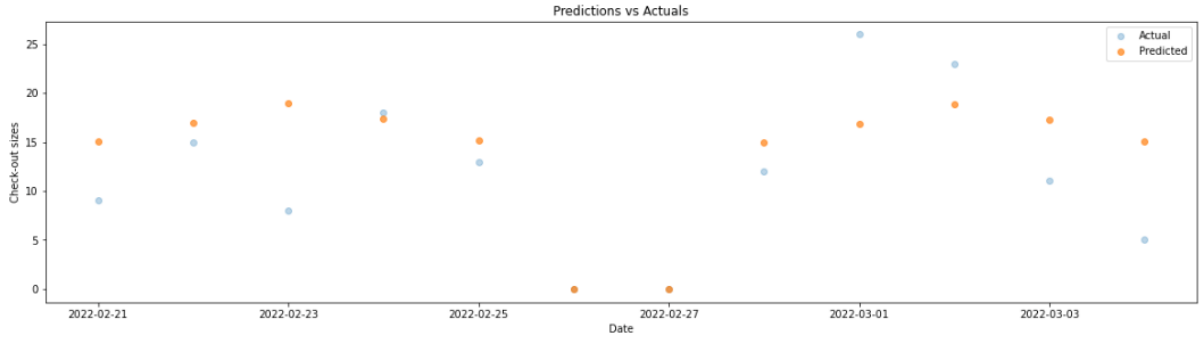


Figure 49 The predictions of the Prophet model for the selected store

Chapter summary

This chapter is mainly dedicated to forecasting demand at two aggregation levels, referring to the long-term and short-term planning horizons. The higher level or longer planning horizon concerns the overall demand in Cologne that can be predicted with an ARIMA forecast model given in section 6.2.2.

Although the correlation with the corona cases was proved earlier, the test in ARIMA modeling with different parameters cast doubt on it. Additionally, it has not been relevant after the 2021 autumn. Thus, we discard the corona cases in the final models.

The second part of the chapter is dedicated to the lower level or shorter planning horizon concerning the demand forecast at a single store where Prophet was deployed.

In this context, we selected one of the top stores according to the stores' median and the variance regarding the check-out sizes, following the boxplot distribution analysis. Prophet is preferred over ARIMA in this section since it can also be smoothly scaled up with multiple seasonality, regressors, and specific days like the opening days-hours of a particular store.

Conclusions and Recommendations

In this thesis, we have developed forecasting models for two levels; demand in a single store and aggregate demand in Cologne. To achieve this prediction, we analyzed internal dependencies and external correlating variables.

The association analysis of the container types did not reveal any significant dependencies, except for one rule showing an association between type-8 and type-4. Since this association rule is very rare, we discarded its dependency in the data and continued the analysis with the most used container, i.e., type-1.

Although spatial autocorrelation shows some indications for such a correlation, the local autocorrelation and the graphical analysis do not justify a significant correlation with the check-out sizes. Nevertheless, local autocorrelation returns a spatial property for the stores, which can be used for the upper-level predictions, e.g., as a regressor in a nationwide aggregated prediction model.

The corona cases correlate significantly with the check-out sizes. Indeed, corona cases prompted the decision-makers to apply counter-corona measures. Those decisions were rather more determinant than the corona cases, as discussed earlier in Chapter 5. Furthermore, a regressor in a multi-step prediction model also requires the prediction of this regressor. Although we demonstrated the possible models with corona cases, it was not modeled in the final model since it is irrelevant after the 2021 Autumn.

Finally, time series analyses and prediction models with ARIMA and Prophet have been studied. Daily and weekly seasonality are evident in both aggregate and individual store demand series. Midday is the peak period, not only for check-outs but also for check-ins. Since we decided to reduce granularity to the daily level, we did not examine daily seasonality. Indeed, it can be significant to know the exact number of containers available every hour in a store. Because after a time, e.g., 45 minutes, these returned containers can be cleaned and reserved again. Then, the store can update the inventory level accordingly. Unfortunately, we do not know the internal processes of the stores. Nevertheless, such an analysis would enable a pinpoint forecast for the store, supported by intraday balancing tours in Cologne.

By predicting the demand at an individual store level, Prophet software is preferred over ARIMA modeling because Prophet can be easily modeled with multi-seasons, regressors, and holidays as we assume that the stores have different characteristics such as food types, opening days and hours, space of the store and so on. In this regard, valuable data can be collected from the stores that can eventually be modeled in the Prophet model.

Customers are another vital source of data. Although our random customer analysis reveals stochastic time series of shopping behavior, they show some pattern in terms of time between check-in and check-out, as we knew from some sapling experiments. Most customers tend to return the containers immediately after the meal. Some return them the following days, usually during lunch, followed by another check-out. With privacy in mind, Vytal can collect additional data about customers, such as home address, work address, car usage, etc., which would help understand customers' shopping habits.

In future studies, researchers can focus on analyzing customer behavior, which is also related to the interaction between check-in and check-out.

Bibliography

Bucci, K., Tulio, M., & Rochman, C. (2020). What is known and unknown about the effects of plastic pollution: A meta-analysis and systematic review. Ecological Applications, 30(2).

Cerqueira, V., Torgo, L., & Soares, C. (2019, 9 29). Machine Learning vs Statistical Methods for Time Series Forecasting: Size Matters. From Machine Learning vs Statistical Methods for Time Series Forecasting: Size Matters: https://arxiv.org/abs/1909.13316

Chatfield, C. (2004). The analysis of Time Series (6th Edition ed.). New York: Chapman & Hall/CRC.

Deuscher Wetterdienst. (2022). Open data. Retrieved 5 5, 2022 from https://opendata.dwd.de/

DIVA-GIS. (2022). Download data by country. Retrieved 5 16, 2022 from http://www.diva-gis.org/gdata

Faloutsos, C., Gasthaus, J., Januschowski, T., & Wang, Y. (2018). Forecasting big time series: old and new. Proceedings of the VLDB Endowment, 2102.

Folium. (2022). Introduction. Retrieved 1 2, 2022 from https://pypi.org/project/folium/

Geopandas. (2022). Overview. Retrieved 1 22, 2022 from https://geopandas.org/

Glen, S. (2016). Statistics How To. Retrieved 6 1, 2022 from https://www.statisticshowto.com/morans-i/#:~:text=The%20Moran's%20statistic%20is%20calculated,%CC%84y)2)%2Fn

Gower, J. (1971). A General Coefficient of Similarity and Some of Its Properties. Biometrics, 27(4), 857-871.

Hassan, A., & Vijayaraghavan, J. (2019). Geospatial Data Science Quick Start Guide (1st Edition ed.). Mumbai: Packt.

Hyndman, R. J., & Athanasopoulos, G. (2021). Forecasting Principles and Practice (3rd Edition ed.). Otexts.

Ilyas, M., Ahmad, W., Khan, H., Yousaf, S., Khan, K., & Nazir, S. (2018). Plastic waste as a significant threat to environment – a systematic literature review. Reviews on Environmental Health, 383-406.

Kassel University. (2022). Statistics. Retrieved 5 19, 2022 from https://www.uni-kassel.de/fb07/ivwl/statistik/lehre/lehrveranstaltungen

Lazzeri, F. (2021). Machine Learning for Time Series Forecasting with Python (1st Edition ed.). Indianapolis: Wiley.

Leonardo Caggiani, M. O. (2012). A Modular Soft Computing based Method for Vehicles Repositioning in Bike-sharing Systems. Procedia - Social and Behavioral Sciences, 54, 675-684.

Michael J de Smith, M. F. (2018). Geospatial Analysis (6th Edition ed.). Drumlin Security Ltd .

Mlxtend. (2022). Mlxtend. Retrieved 2 2, 2022 from http://rasbt.github.io/mlxtend/

National Institute of Standards and Technology. (2022, 5 1). Kolmogorov-Smirnov Goodness-of-Fit Test. Retrieved 6 1, 2022 from https://www.itl.nist.gov/div898/handbook/eda/section3/eda35g.htm

ndr.de. (2022). Corona-Chronologie. Retrieved 4 1, 2022 from https://www.ndr.de/nachrichten/info/Corona-Chronologie-Die-Ereignisse-im-Norden,coronachronologieindex100.html

NetworkX. (2022, 6 4). NetworkX. Retrieved 7 29, 2022 from NetworkX: https://networkx.org/

Offene Daten Köln. (2022, 5 5). Grüne Flächen. Retrieved 1 1, 4 from Grüne Flächen: https://offenedaten-koeln.de/dataset/gruenflaechenkataster-koeln-flaechentypen

Open Street Maps. (2022). OpenStreetMap. Retrieved 5 11, 20222 from https://www.openstreetmap.org/#map=6/51.330/10.453

Peixeiro, M. (2022). Time Series Forecasting in Python (1st Edition ed.). New York: Manning.

Prophet. (2022). Introduction. Retrieved 1 1, 2022 from https://facebook.github.io/prophet/

Pysal Developers. (2018). Spatial autocorrelation. Retrieved 1 1, 2021 from https://pysal.org/esda/notebooks/spatialautocorrelation.html

Pysal. (2022). Home. Retrieved 1 1, 2022 from https://pysal.org/

QGIS. (2022). QGIS. Retrieved 5 15, 2022 from https://qgis.org/en/site/

Rachih, H., Mhada, F. Z., & Chiheb, R. (2019). Meta-heuristics for reverse logistics: A literature review and perspectives. Computers & Industrial Engineering, 45-62.

Rafferty, G. (2021). Forecasting Data Time Series Data with Facebook Prophet (1st Edition ed.). Mumbai: Pack.

Rhodes, C. J. (2018). Plastic Pollution and Potential Solutions. Science Progress(September), 207-260.

Robert Koch Institute. (2022, 4 4). SARS-CoV-2_Infektionen_in_Deutschland. Retrieved 3 1, 2022 from GitHub: https://github.com/robert-koch-institut/SARS-CoV-2_Infektionen_in_Deutschland

Robson, W. (2019). The Math of Prophet. Retrieved 2 2, 2023 from https://medium.com/future-vision/the-math-of-prophet-46864fa9c55a

rspatial.org. (2022). Spatial Data Science. Retrieved 5 18, 2022 from https://rspatial.org/raster/analysis/3-spauto.html

Schumway, R. H., & Stoffer, D. S. (2016). Time Series Analysis and Its Applications with R Examples. Cham: Springer.

Scikit-learn. (2022). Density Estimation. Retrieved 2 2, 2022 from https://scikit-learn.org/stable/modules/density.html

Scikit-learn. (2022). Scikit-learn home. Retrieved 1 2, 2022 from https://scikit-learn.org/stable/

Scipy. (2022). scipy.stats.ks_2samp. Retrieved 1 1, 2022 from https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.ks_2samp.html

Shapely. (2022). Shapely User Manual. Retrieved 6 1, 2022 from https://shapely.readthedocs.io/en/stable/manual.html

Singh, D., & Ajay Verma. (2018). Inventory Management in Supply Chain. Materials Today: Proceedings, 3867-3872.

Smith, M. J., Goodchild, M. F., & Longley, P. A. (2018). Geospatial Analysis (6th Edition ed.). London: The Winchelsea Press.

Stadt-Köln. (2022, 4 1). Veranstaltungen. Retrieved 3 2, 2022 from Veranstaltungen: https://www.stadt-koeln.de/leben-in-koeln/veranstaltungen

Statsmodels. (2022, 5 1). Granger casuality tests. From Granger casuality tests: https://www.statsmodels.org/dev/generated/statsmodels.tsa.stattools.grangercausalitytests.html?highlight=granger#statsmodels.tsa.stattools.grangercausalitytests

Statsmodels. (2022, 5 1). Statsmodels Homepage. Retrieved 6 2, 2022 from https://www.statsmodels.org/stable/index.html

Statsmodels. (2022, 6 1). Vector Autoregressions. From Vector Autoregressions: https://www.statsmodels.org/dev/vector_ar.html

Statsmodels. (2022). Augmented Dickey-Fuller. Retrieved 5 5, 2022 from https://www.statsmodels.org/dev/generated/statsmodels.tsa.stattools.adfuller.html

Statsmodels. (2022). statistical models, hypothesis tests, and data exploration. Retrieved 5 5, 2022 from https://www.statsmodels.org/dev/index.html#

Tan, P.-N., Steinbach, M., Karpatne, A., & Kumar, V. (2020). Introduction to Data Mining (2nd Edition Global Edition ed.). New York: Pearson.

Vandeput, N. (2020). Inventory Optimization (1st Edition ed.). Berlin: Walter De Gruyter GmbH.

Verkehrsverbund-Rhein-Sieg. (2022, 5 1). OpenData / OpenService. Retrieved 2 2, 2022 from https://www.vrs.de/fahren/fahrplanauskunft/opendata-/-openservice

Vytal. (2022, 5 5). Home Page. Retrieved 1 15, 2022 from https://en.vytal.org/

Wiwo. (2022). So ist der zweite Lockdown in Deutschland verlaufen. Retrieved 4 1, 2022 from https://www.wiwo.de/politik/deutschland/corona-lockdown-so-ist-der-zweite-lockdown-in-deutschland-verlaufen/27076474.html

Wiwo.de. (2022). Wie verlief der erste Lockdown in Deutschland? Retrieved 3 1, 2022 from https://www.wiwo.de/politik/deutschland/corona-wie-verlief-der-erste-lockdown-in-deutschland/26853384.html

Zhou, Y. &. (2018). A Markov Chain Based Demand Prediction Model for Stations in Bike Sharing Systems. Mathematical Problems in Engineering. , 1(8).

Appendix A : Jupyter Notebook Codes

Jupyter Notebook Codes are available in a GitHub repository (https://github.com/KultiginX/vytal). The repository is not public due to confidentiality. However, access permission to the repository can be granted to the persons in the examination of this thesis. Nevertheless, the codes of the last chapter are annexed as pdf files.

Appendix B : Forecasting with Prophet Modeling

In this section, we aim to show a possible yearly seasonal coefficient generated from a model with holidays, and an exogenous predictor, namely corona cases.

The model is provided in Appendix A. The parameters are determined after the manual experiments. Regarding the model components, the trend is defined as a linear function. We defined two seasonality, yearly and weekly. While weekly is defined as multiplicative, the yearly seasonality is modeled with the additive parameter after some experiments. It is noteworthy that the weekly seasonality is very strong, as shown in Chapter 6, while the yearly seasonality is not so apparent.

Regarding the holidays component, they are modeled like binary regressors. We modeled only the Christmas break, which refers to the dates between 23rd December and 1st January. Other national holidays or school holidays can also be modeled. However, it is not very necessary in the model since we had already simplified and aggregated our time series by taking the max of the calendar.

Below are given the results of the model.

The first plot shows the fitness of the model while red vertical dashed lines show the change points. The model is based on a linear trend line. Those change points refer to the intervals that those trend lines are to be calculated. Instead of default settings, we intentionally define those change points to capture the yearly seasonality. These are the dates when the company's transactions are disrupted significantly. The first change point in November 2020 refers to the end of the startup phase. The second and the third points indicate the Christmas break boundaries. The fourth point shows the start of the corona-counter measures relaxation. The last two points define the summer holiday in NRW.

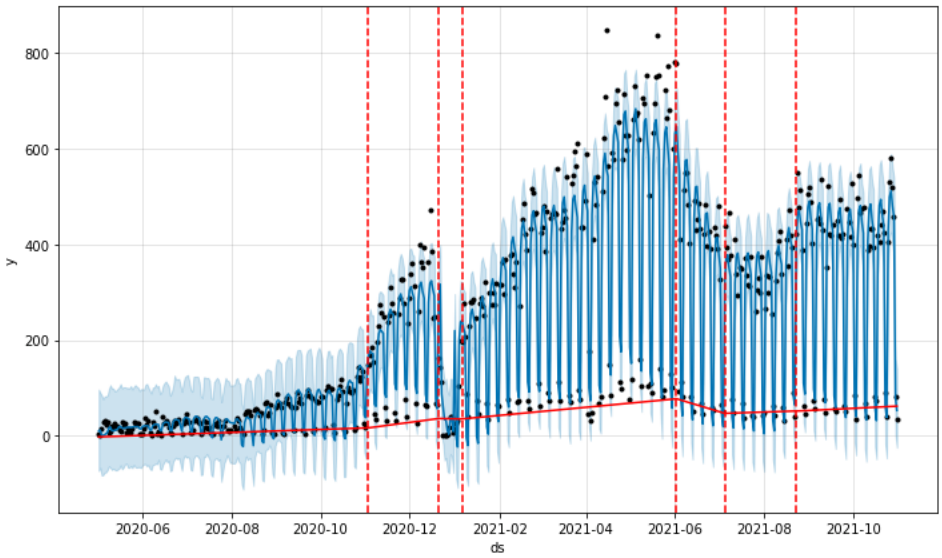


Figure B. 1 Prophet model forecast

The following plot shows those fitted trend lines between those change points.

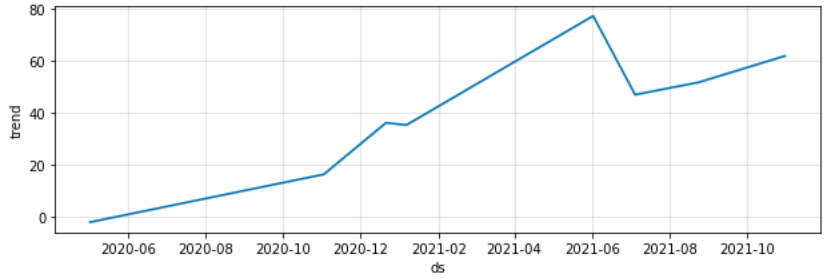


Figure B. 2 Trend plot of the model

The following plot shows the contribution of the holidays, which was modeled as a multiplicative seasonality. Thus, its contribution is shown as percentages on the y axis. We defined only the Christmas breaks as holidays such that we observe the contribution of only Christmas 2020.

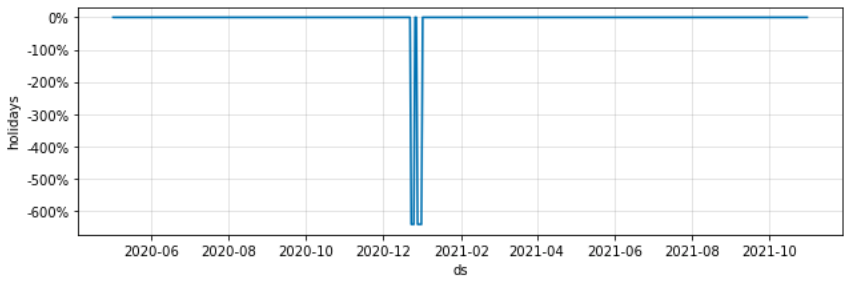


Figure B. 3 Holidays effect of the model

The weekly seasonality is also defined as a multiplicative seasonality. Weekdays have more check-outs than weekends, as discussed earlier.

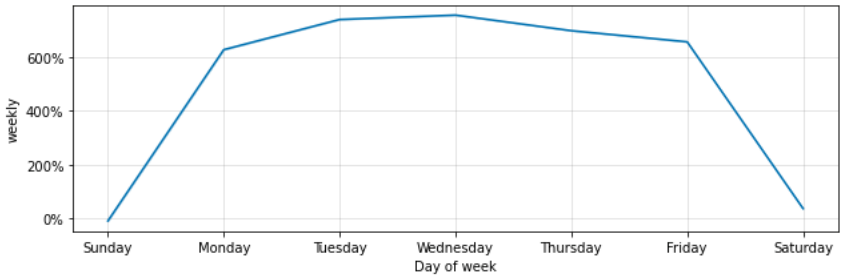


Figure B. 4 Weekly seasonality

The yearly seasonality is defined as an additive seasonality. Noe that there are two peaks in March and December.

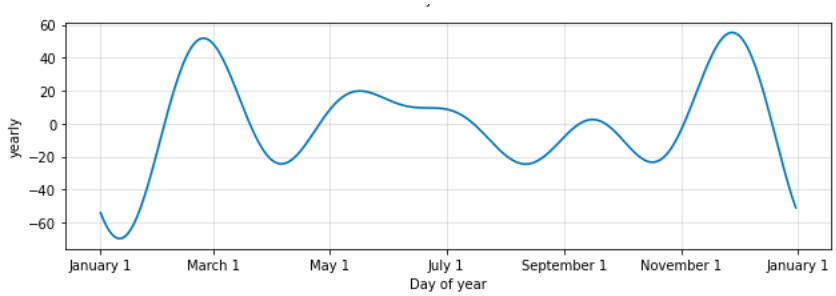


Figure B. 5 Yearly seasonality

The corona cases are also modeled as a multiplicative regressor. The cases after 1st October are replaced by the mean of the cases since they are not relevant afterward. Note that a regressor's predictions should also be passed to call the predict function since the model is already fitted with that regressor. That is why we replaced them with the mean instead of deleting them. Passing actual corona cases would mislead the model to forecast accordingly, although it is not the case after this date. To note again, we want to discover the seasonality with corona cases in the same model. Shortly, its effect is removed by taking the mean after 1st October 2021, while we can still get the yearly seasonality masked with the corona cases before this date.

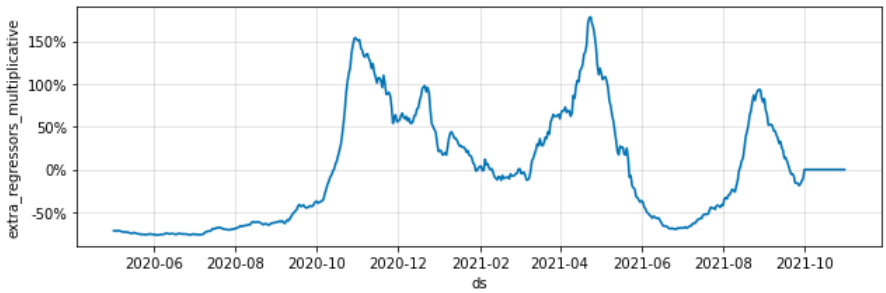


Figure B. 6 Corona cases contribution

After fitting the model, we tested the model on the test set of the series, namely after November 2021. The following plots show the diagnostics for the test portion of the series. The only

difference from the above discussion is that we have only one trend line, which is slightly positive.

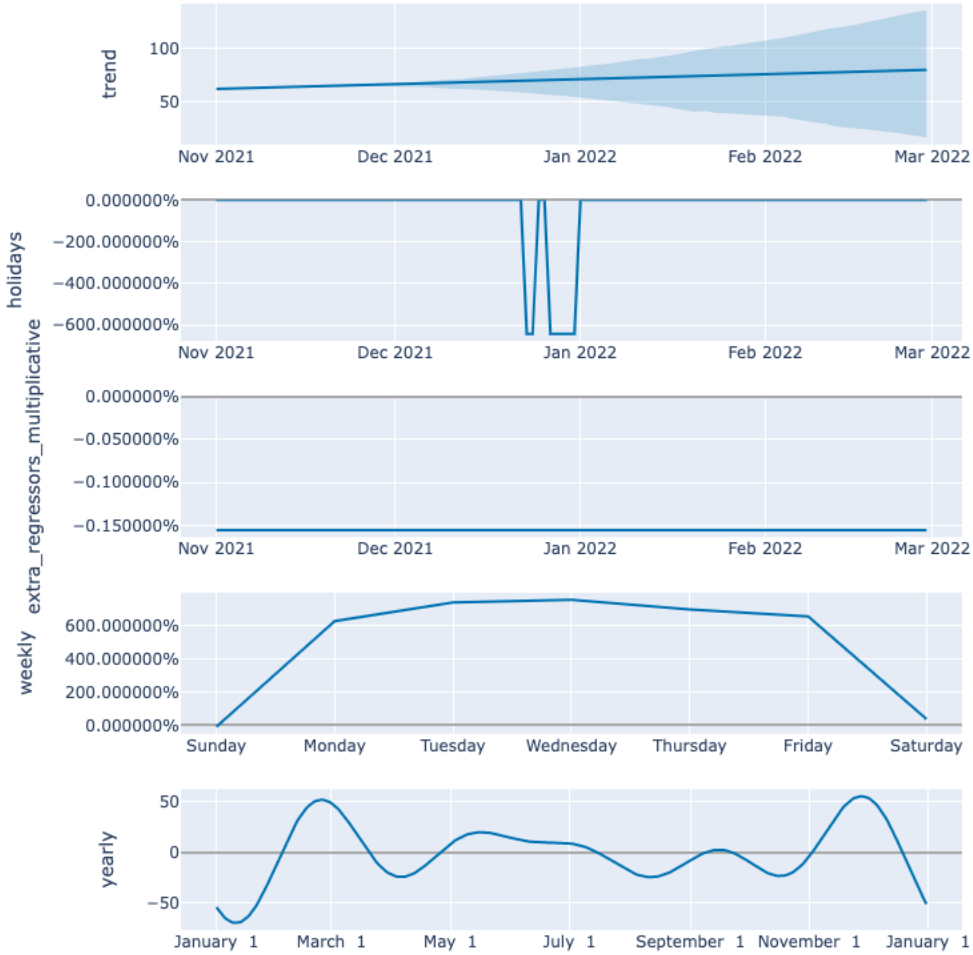


Figure B. 7 Model diagnostics on test set

The predictions on the test time series, which is a significantly large window from November 2021 to February 2023, are plotted below. We chose this larger window because we want to show the effect of the yearly seasonality in predictions. We see an increase towards March primarily due to the seasonality shown in the last figure above.

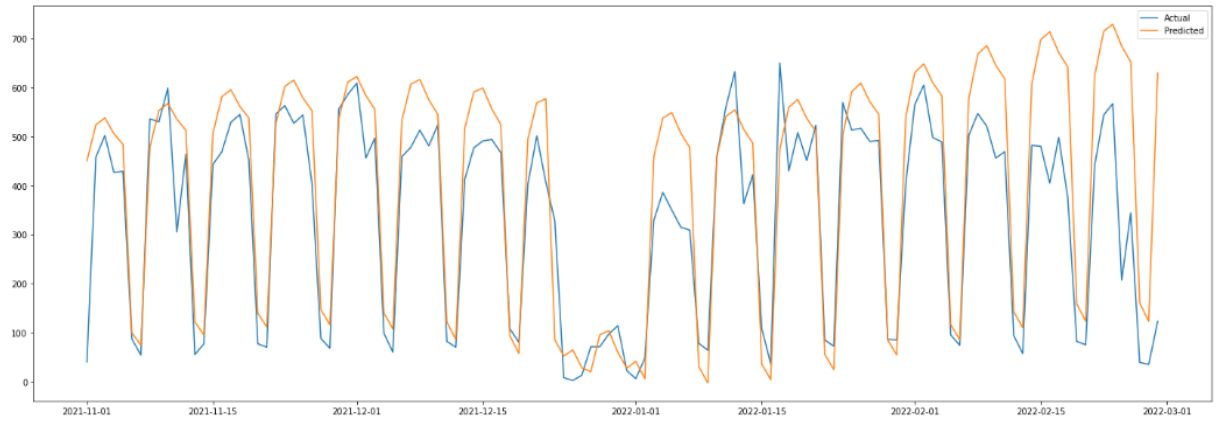


Figure B. 8 Actual vs. predictions

The model generates yearly seasonality from the series of two years after September 2019, which is relatively tiny data with respect to the yearly seasonality analysis. Analyzing the yearly seasonality plot reveals unplausible information. We do not expect an increase in check-outs in December or March. As discussed in Chapter 6, the counter-corona measures, specifically lockdowns, rather than 7-day corona cases, affect the check-outs. The corona cases regressor can explain only a small portion of the variance, as we proved in section 5.5. In addition, the yearly seasonality was generated from tiny data. Consequently, the predictions plot above falsifies such a yearly seasonality resulting in peaks in March.

Appendix C : Kernel Density Estimation of the Check-outs

We have shown the variance in the daily check-out numbers and analyzed the time series of a selected store regarding check-outs in section 6.4. The accompanying noise to the selected store's daily check-out numbers is shown in Figure 47. Since accurate forecasting is challenging with such noise, setting a service level can be helpful. Service level is a tool to manage the inventory by ensuring a certain inventory security level. As discussed in the introduction chapter, the company has no variable cost for keeping the containers at the stores but generates profit from every check-out. Therefore, maintaining a certain number of containers at the store can ensure keeping the backlogging below a certain level.

The following plot shows the daily check-out numbers of the selected store. To note again, this particular store has been intentionally selected to have less variance in the number of check-outs, as discussed in section 6.4. Since we do not know the reason behind the zeros, whether the store was closed or it did not record any check-out, we removed the zeros from the series.

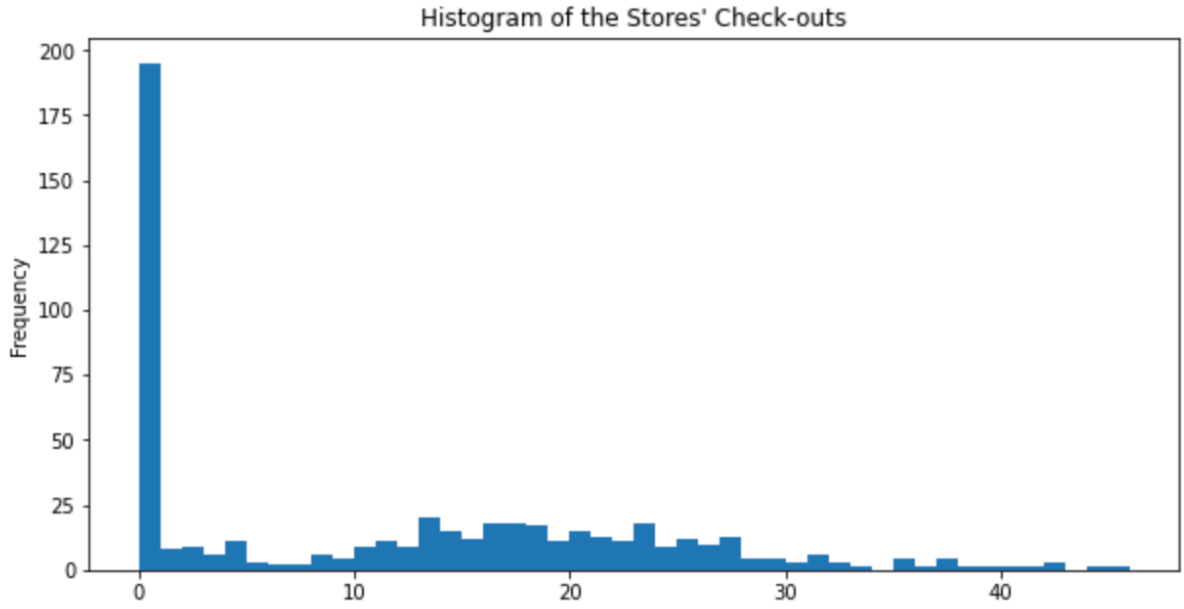


Figure C. 1 Daily check-outs histogram of the selected store

By estimating the function of this distribution, we can generate a probability function and use the cumulated distribution function(CDF) for the service level policy. Scikit-learn library offers six different kernels, e.i. "gaussian", "tophat", "epanechnikov", "exponential", "linear", "cosine" for kernel density estimation (KDE). Scikit-learns gives a brief mathematical introduction to the kernel on its web page (Scikit-learn, 2022)

The Gaussian kernel is used on the training set consisting of 2021 daily check-out numbers because we expect a distribution similar to the normal distribution regarding the daily check-outs. One main parameter is the bandwidth which refers to the smoothing of the function. A grid search ranging from 0.05 to 4 has been applied, resulting in the best bandwidth of 3.85. The top figure below shows the distribution of the check-outs, while the second one shows the estimated density function with a bandwidth of 3.85. Validation is done visually by plotting the check-out distribution on the test set at the bottom. The test set consists of the check-outs in January-February 2022.

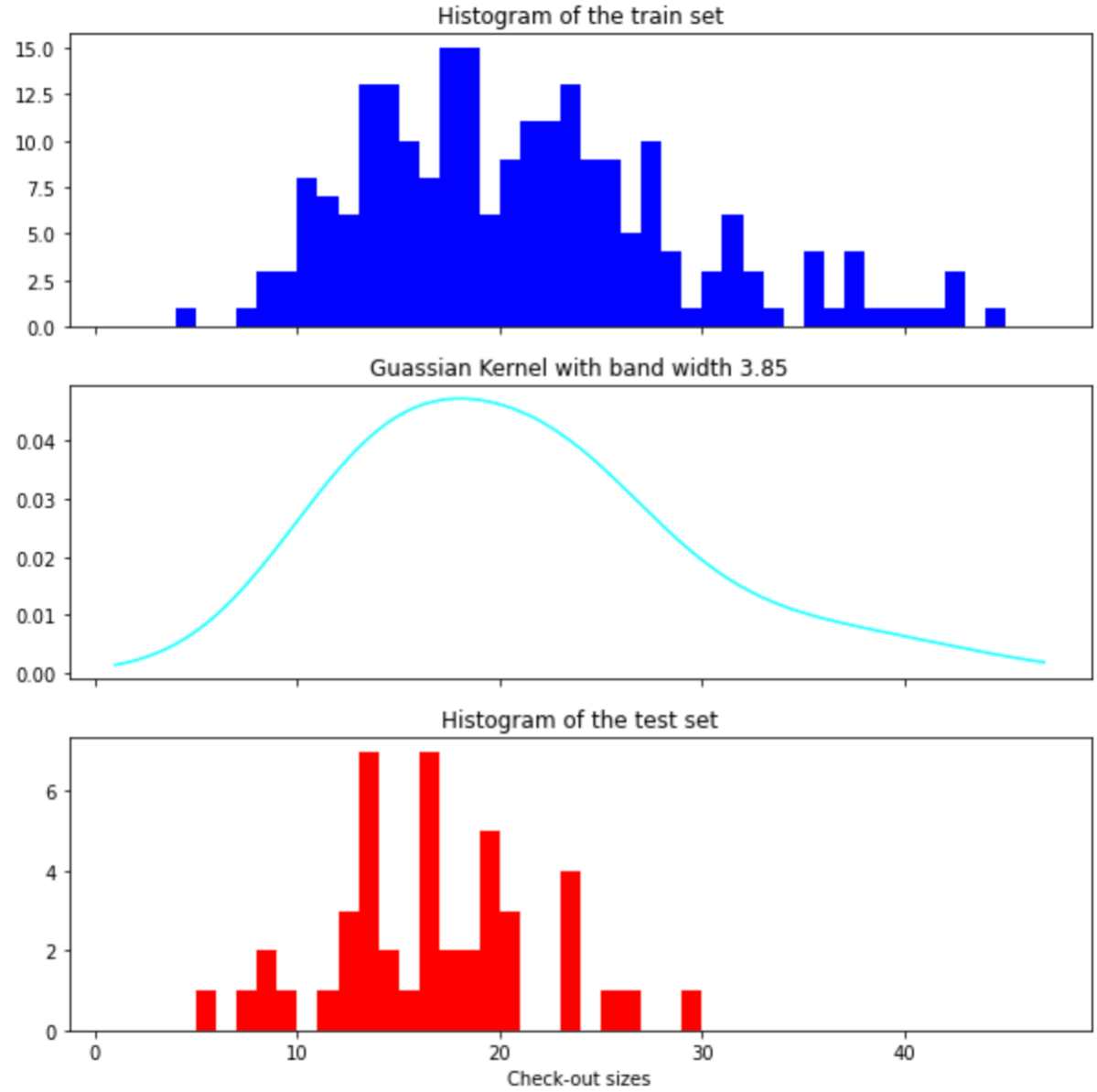


Figure C. 2 Histograms and the Gaussian KDE

Although we decide to use the Gaussian kernel, we look for the best kernel with its bandwidth resulting from a grid search with the different kernels and bandwidths. The grid search returns

the linear search with 0.1 bandwidth as the best one. However, comparing the density function with the test set distribution in the figure below shows how far the model is overfitting. Since the check-out numbers are discrete, the linear kernel density function results in such a function with the peaks at those discrete numbers. But of course, the test set has different discrete numbers, as seen in the bottom plot below.

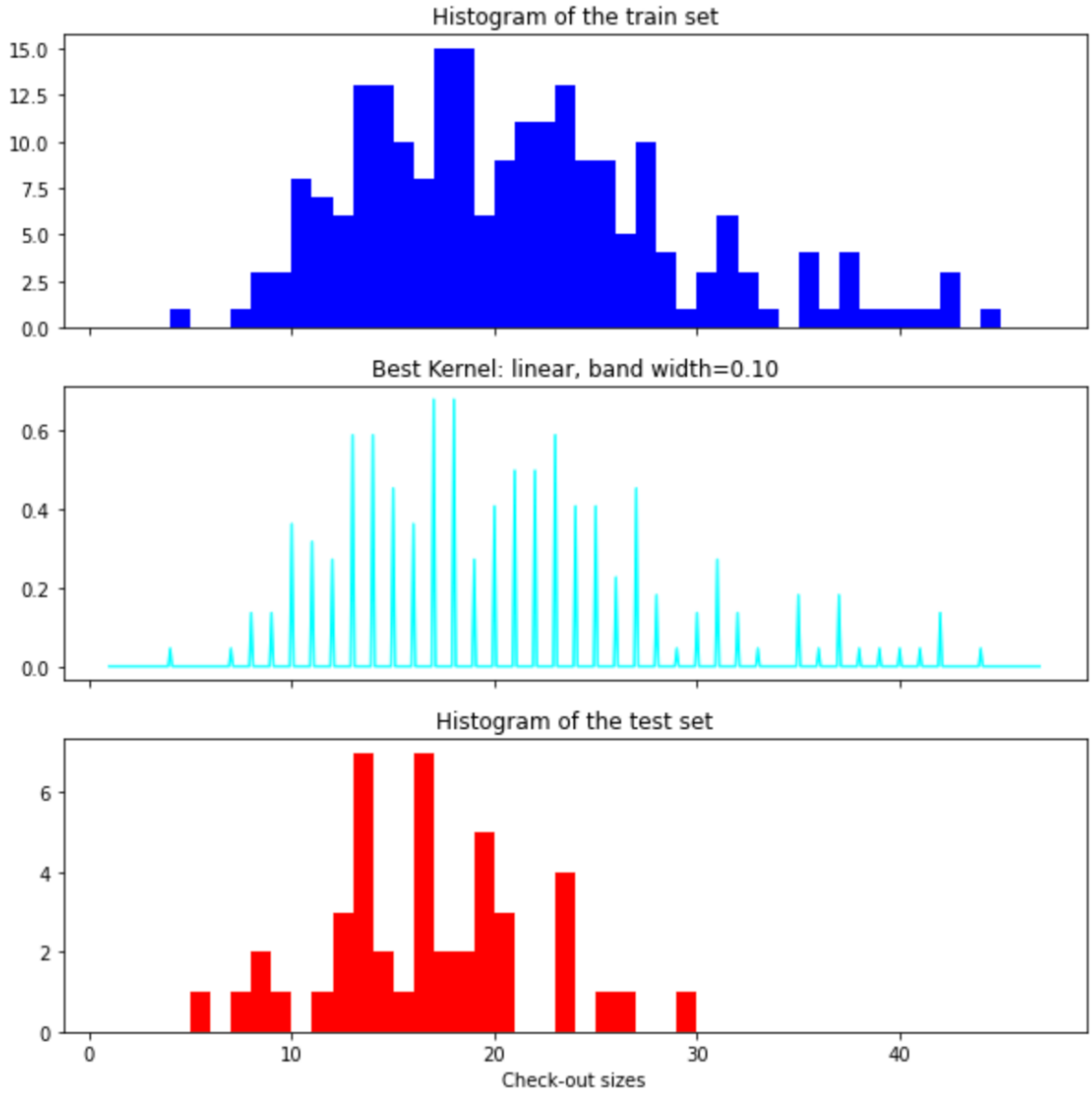


Figure C. 3 Best kernel and corresponding histograms

In this appendix, we showed a kernel density estimation for setting a service level. Despite having different kernels, we expect a normal distribution in check-outs. The best bandwidth for this store is 3.85, a well-smoothed density function. Thus, this function lays the foundation for initiating a service-level policy.

Declaration of Authenticity

I, Kültigin Bozdemir, hereby declare that the work presented herein is my own work completed without the use of any aids other than those listed. Any material from other sources or works done by others has been given due acknowledgement and listed in the reference section. Sentences or parts of sentences quoted literally are marked as quotations; identification of other references with regard to the statement and scope of the work is quoted. The work presented herein has not been published or submitted elsewhere for assessment in the same or a similar form. I will retain a copy of this assignment until after the Board of Examiners has published the results, which I will make available on request.

Chapter 6 Time Series Forecasting

Section 6.1 and 6.2

Prediction of demand with an ARIMA Model

In [1]:
```
#import packages
import pandas as pd
import numpy as np
import geopandas as gpd
import matplotlib.pyplot as plt
import statsmodels
import datetime as dt
import os
import warnings
```

In [2]:
```
print(statsmodels.__version__)
print(pd.__version__)
print(np.__version__)
print(gpd.__version__)
```

```
0.13.2
1.4.2
1.21.6
0.10.2
```

In [3]:
```
warnings.filterwarnings("ignore")
```

In [4]:
```
%store -r c_co_in
%store -r c_co_out
```

In [5]:
```
# get daily max check-outs
ops_day=c_co_out.groupby(c_co_out['OwnerSince'].dt.date).size().to_frame('count_')
ops_day['check_ins']=c_co_in.groupby(c_co_in['OwnerTill'].dt.date).size()
ops_day['date']=pd.to_datetime(ops_day.index)
print(ops_day.shape)
ops_day.head()
```

```
(837, 3)
```

Out[5]:

OwnerSince	count_	check_ins	date
2019-09-23	62	1.0	2019-09-23
2019-09-24	13	1.0	2019-09-24
2019-09-25	21	8.0	2019-09-25
2019-09-26	19	9.0	2019-09-26
2019-09-27	15	11.0	2019-09-27

In [6]:
```
ops_day.tail()
```

Out[6]:

OwnerSince	count_	check_ins	date
2022-02-28	123	172.0	2022-02-28
2022-03-01	417	453.0	2022-03-01
2022-03-02	377	478.0	2022-03-02
2022-03-03	244	485.0	2022-03-03
2022-03-04	78	362.0	2022-03-04

In [7]:
```
# create a bin including empty days between the range of earliest and latest day that a transaction took place
bins=pd.date_range(start=pd.to_datetime(ops_day.index).min(), end=pd.to_datetime(ops_day.index).max())
bins=pd.to_datetime(bins)
bins=pd.Series(bins).dt.date
bins=pd.DataFrame(bins, columns=['date'])


# add this bin as an index into df with merge operation
# now, we have all days in the df
ops_day['date']=pd.to_datetime(ops_day.index)
ops_day['date']=ops_day['date'].dt.date
ops_day.index=ops_day['date']

ops_day=pd.merge(bins,ops_day, how='left', left_on=bins['date'], \
                 right_index=True).fillna(0).drop(['date_y'], axis=1).rename(columns={'date_x':'date'}).set_index('date')
ops_day['date']=pd.to_datetime(ops_day.index)
print(ops_day.shape)
```

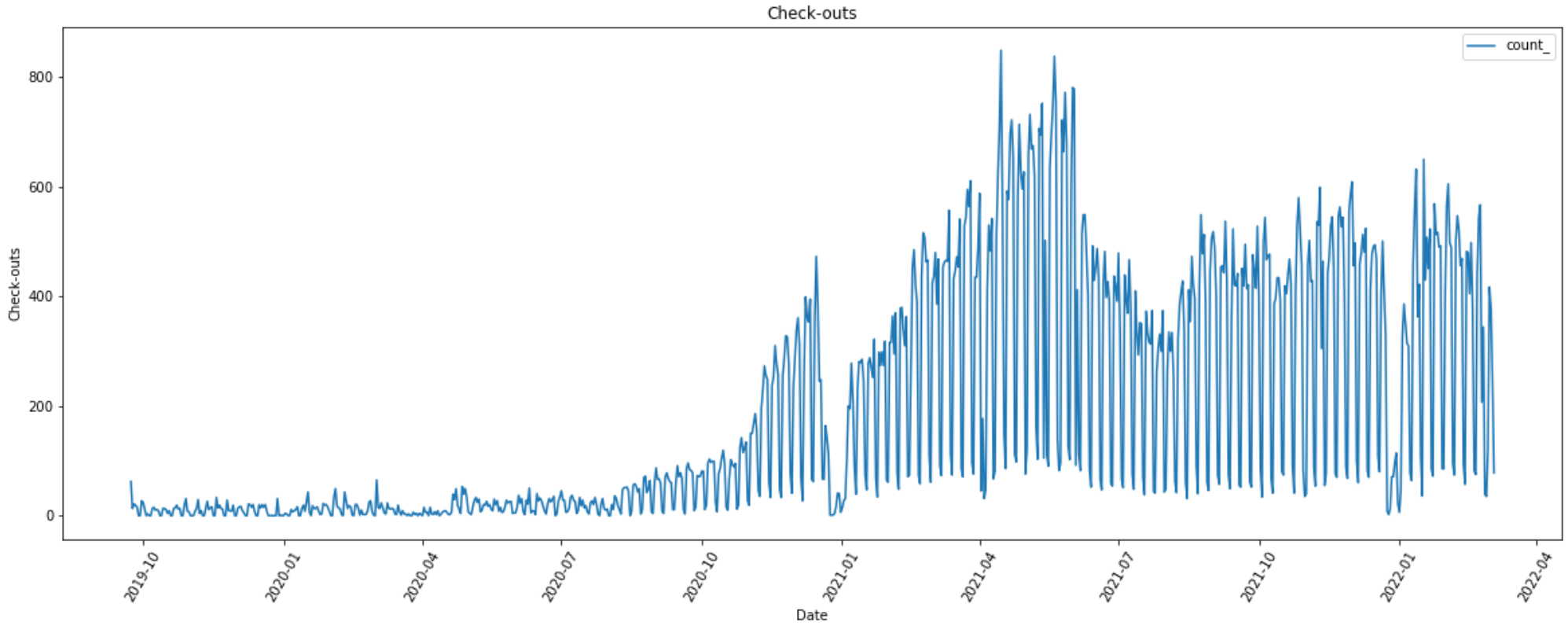
```
(894, 3)
```

In [8]:
```
# Corona cases
df=pd.read_csv('Aktuell_Deutschland_SarsCov2_Infektionen.csv')
df['Meldedatum']= pd.to_datetime(df['Meldedatum'])
df['Refdatum']= pd.to_datetime(df['Refdatum'])



cdf=df[(df['IdLandkreis']==5315)].groupby(df['Meldedatum'].dt.date)['AnzahlFall'].sum().to_frame('corona')



# join with ops transactional data
ops_day=ops_day.join(cdf, how='left')#.drop('key_0', axis=1)

ops_day.tail()
```

Out[8]:

date	count_	check_ins	date	corona
2022-02-28	123.0	172.0	2022-02-28	1620.0
2022-03-01	417.0	453.0	2022-03-01	3028.0
2022-03-02	377.0	478.0	2022-03-02	3821.0
2022-03-03	244.0	485.0	2022-03-03	6851.0
2022-03-04	78.0	362.0	2022-03-04	4985.0

In [9]:
```
# plot
ops_day[['count_']].plot(figsize=(20,7) )
plt.xticks(rotation=60)
plt.title('Check-outs')
plt.xlabel('Date')
plt.ylabel('Check-outs')
plt.show()
```



In [10]:
```
# 7_day_corona cases
for i,row in ops_day.iterrows():
    ops_day.loc[i,'7_day']=ops_day[:i].tail(7)['corona'].mean()

ops_day.tail()
```

Out[10]:

date	count_	check_ins	date	corona	7_day
2022-02-28	123.0	172.0	2022-02-28	1620.0	1670.000000
2022-03-01	417.0	453.0	2022-03-01	3028.0	1782.285714
2022-03-02	377.0	478.0	2022-03-02	3821.0	1996.000000
2022-03-03	244.0	485.0	2022-03-03	6851.0	2724.714286
2022-03-04	78.0	362.0	2022-03-04	4985.0	3173.000000

In [11]:
```
# crop the series
```

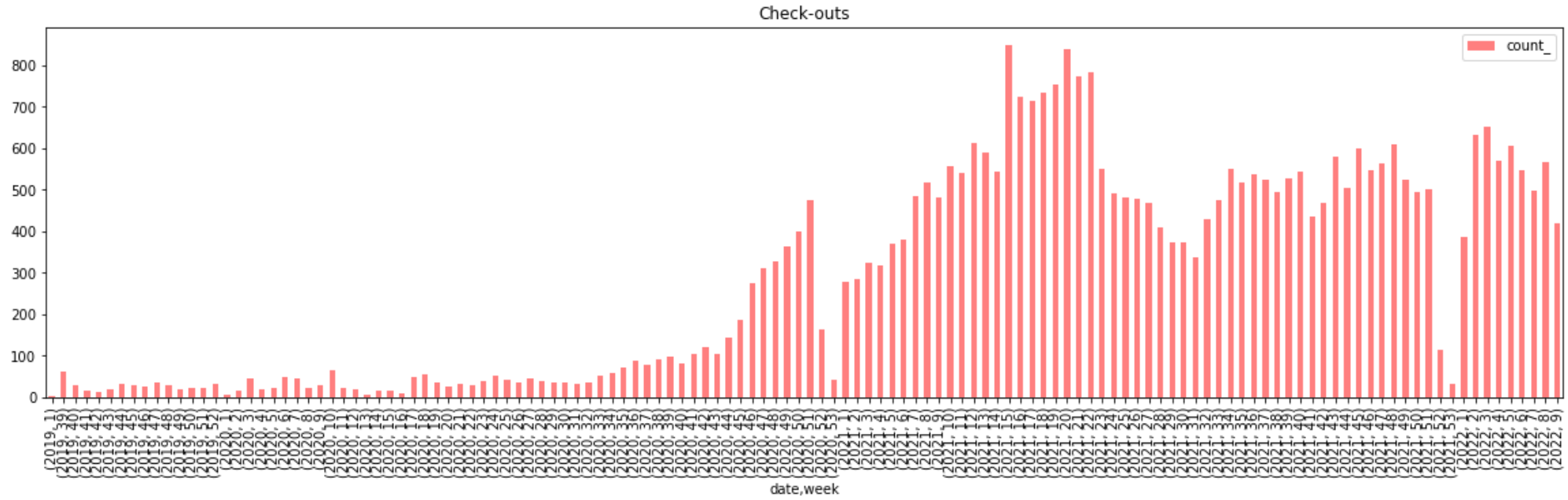
```
last_date=dt.datetime(2022, 3, 1).date()
ops_day=ops_day[ops_day.index<=last_date]
```

In [12]:
```
# group by week
ops_week=ops_day.groupby([ops_day['date'].dt.year, ops_day['date'].dt.isocalendar().week]).max()

ops_week['year_week']=ops_week.index

ops_week.tail()

ops_week.drop((2022, 52), inplace=True) # remove 52nd week from 2022?
ops_week.tail()
```

Out[12]:

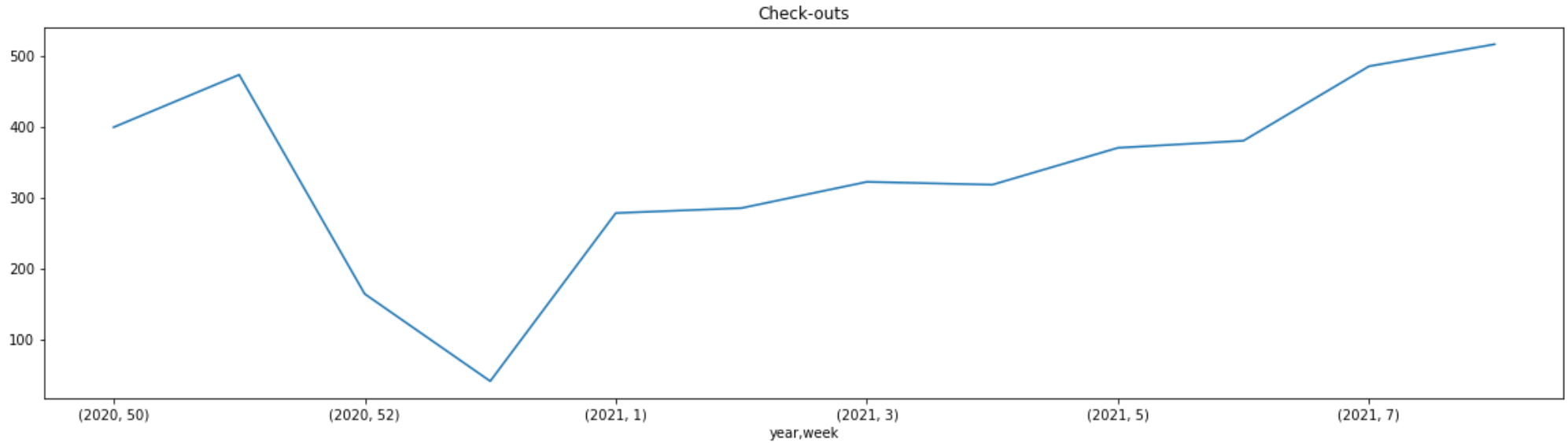
		count_	check_ins	date	corona	7_day	year_week
date	**week**						
2022	**5**	605.0	510.0	2022-02-06	4138.0	2855.857143	(2022, 5)
	6	547.0	543.0	2022-02-13	3566.0	2926.571429	(2022, 6)
	7	498.0	524.0	2022-02-20	2842.0	2471.000000	(2022, 7)
	8	567.0	570.0	2022-02-27	2325.0	1922.571429	(2022, 8)
	9	417.0	453.0	2022-03-01	3028.0	1782.285714	(2022, 9)

In [13]:
```
# plot

plt.figure(figsize=(20, 5))
plt.title('Check-outs')
ops_week['count_'].plot.bar(color='Red', legend=True,  alpha=0.5)
ops_week.index.names = ['year','week']

plt.show()
```

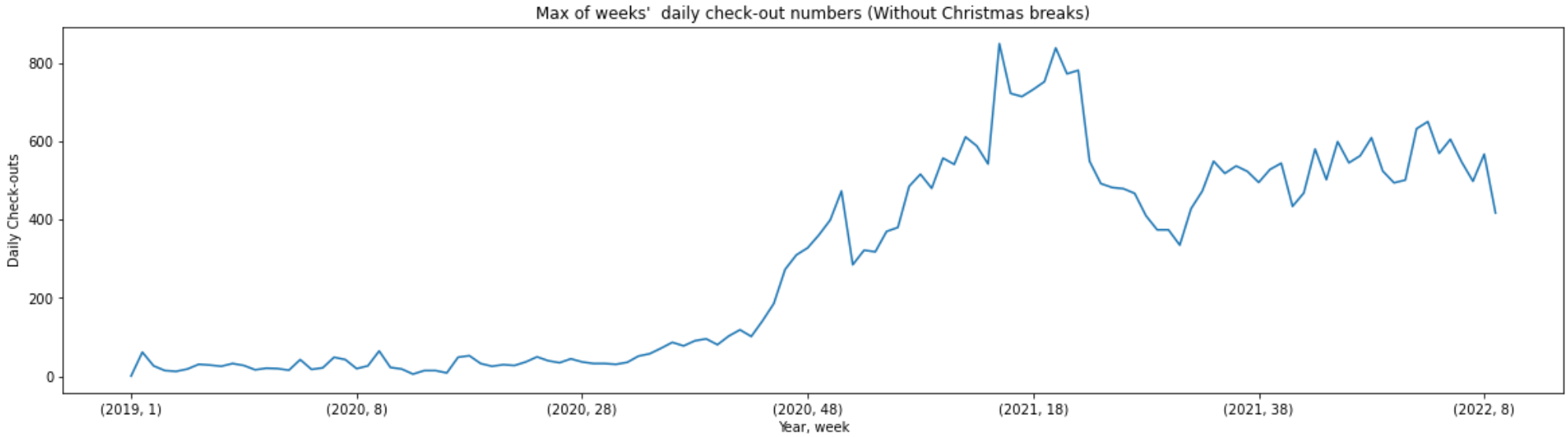


In [14]:
```
# plot

plt.figure(figsize=(20, 5))
plt.title('Check-outs')
ops_week[(2020,50):(2021,8)]['count_'].plot()
```

Out[14]:
```
<AxesSubplot:title={'center':'Check-outs'}, xlabel='year,week'>
```

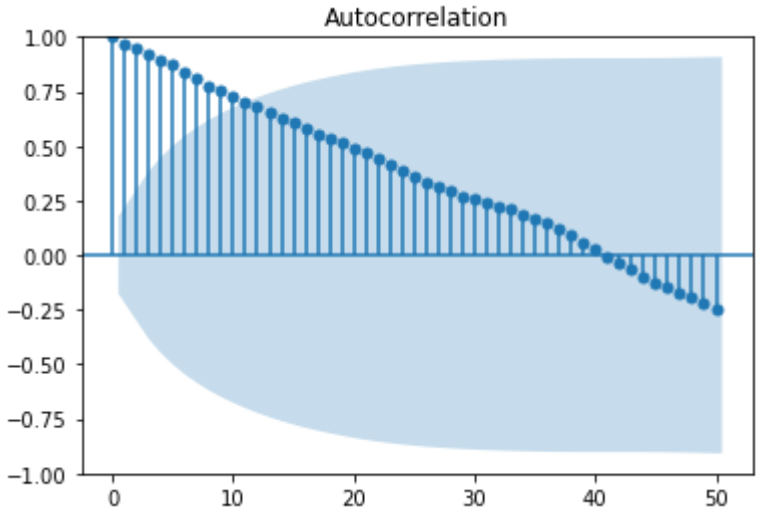


In [15]:
```
# Since we know drops at christmas time, we drop them...
#ops=ops_week[(2021,40):(2022,8)]['count_'].drop([(2021,52),(2021,53),(2022,1)]).reset_index()['count_']
ops=ops_week.drop([(2019,52),(2020,1),(2020,52),(2020,53),(2021,1),(2021,52),(2021,53),(2022,1)])
#ops=ops_week.drop([(2020,53),(2021,53)])#.reset_index() # drop fractional weeks
# plot
plt.figure(figsize=(20, 5))

plt.title("Max of weeks'  daily check-out numbers (Without Christmas breaks)")

ops['count_'].plot()
plt.xlabel('Year, week')
plt.ylabel('Daily Check-outs')
plt.show()
```

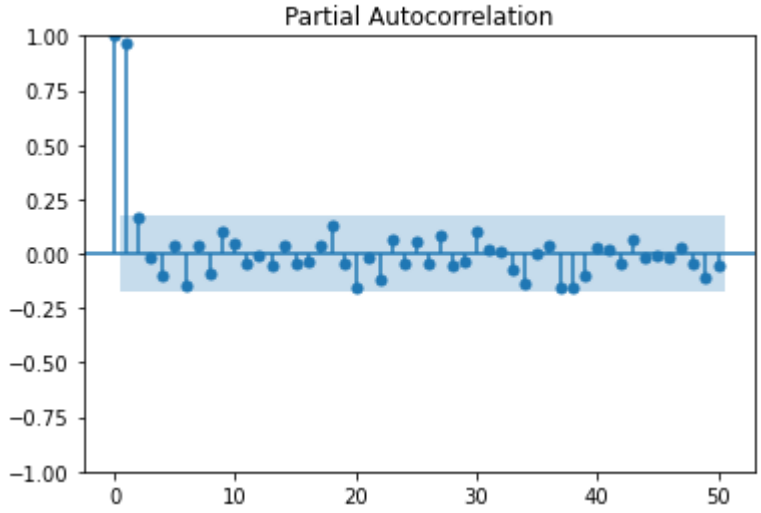


In [16]:
```
from statsmodels.graphics.tsaplots import plot_acf, plot_pacf
plt.figure(figsize=(10, 5))
#plt.title('Autocorrelation')
plot_acf(ops['count_'], lags=50)
plt.show()
```

```
<Figure size 720x360 with 0 Axes>
```



In [17]:
```
p=plot_pacf(ops['count_'], method='ywm', lags=50)
```



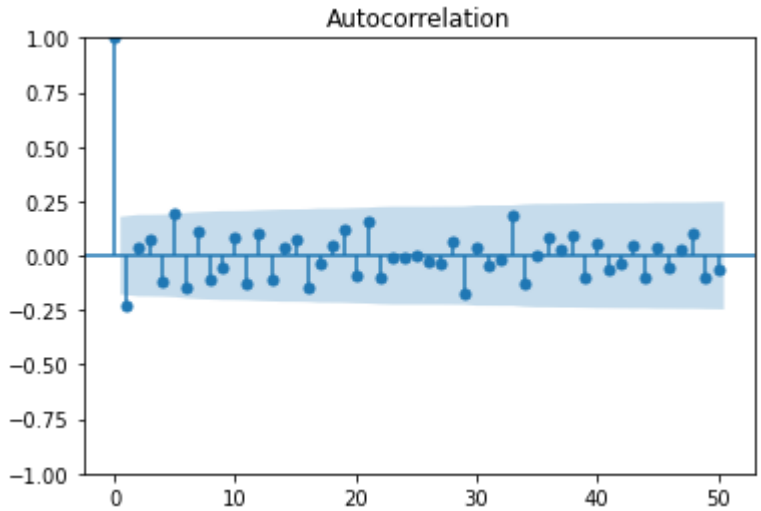
In [18]:
```
plt.figure(figsize=(10, 5))
plt.title('Check-outs after 1st differencing')
ops['count_'].diff().plot()
plt.show()
```

In [19]:
```
# adf füller test
from statsmodels.tsa.stattools import adfuller
```

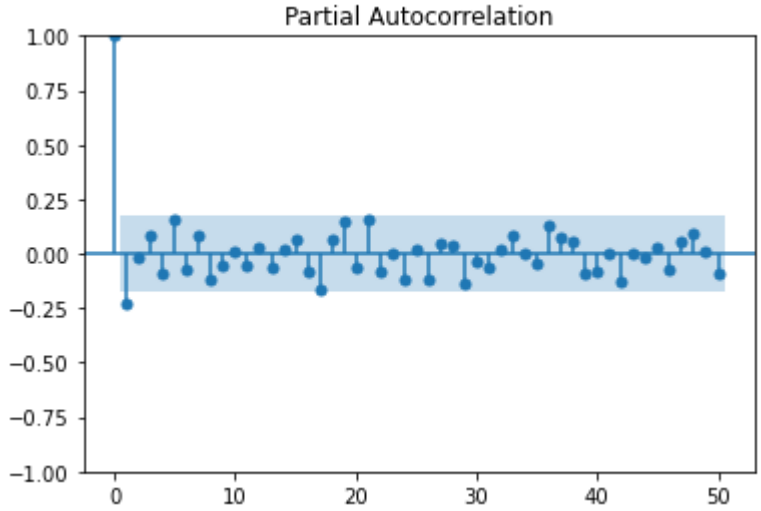
```
print('without differencing:',adfuller(ops['count_'].dropna())[1]) # p-value is not significant.
print('1st differencing:',adfuller(ops['count_'].diff().dropna())[1]) # p-value is significant
# The series is stationary...
```

```
without differencing: 0.7020704642633246
1st differencing: 2.8187871855504284e-25
```

In [20]:
```
# Autocorrelation plot
p=plot_acf(ops['count_'].diff().dropna(), lags=50)
```



In [21]:
```
# partial autocorrelation plot
p=plot_pacf(ops['count_'].diff().dropna(), method='ywm', lags=50) # 1st differencing
```



In [22]:
```
ops=ops.set_index('year_week')
ops.tail()
```

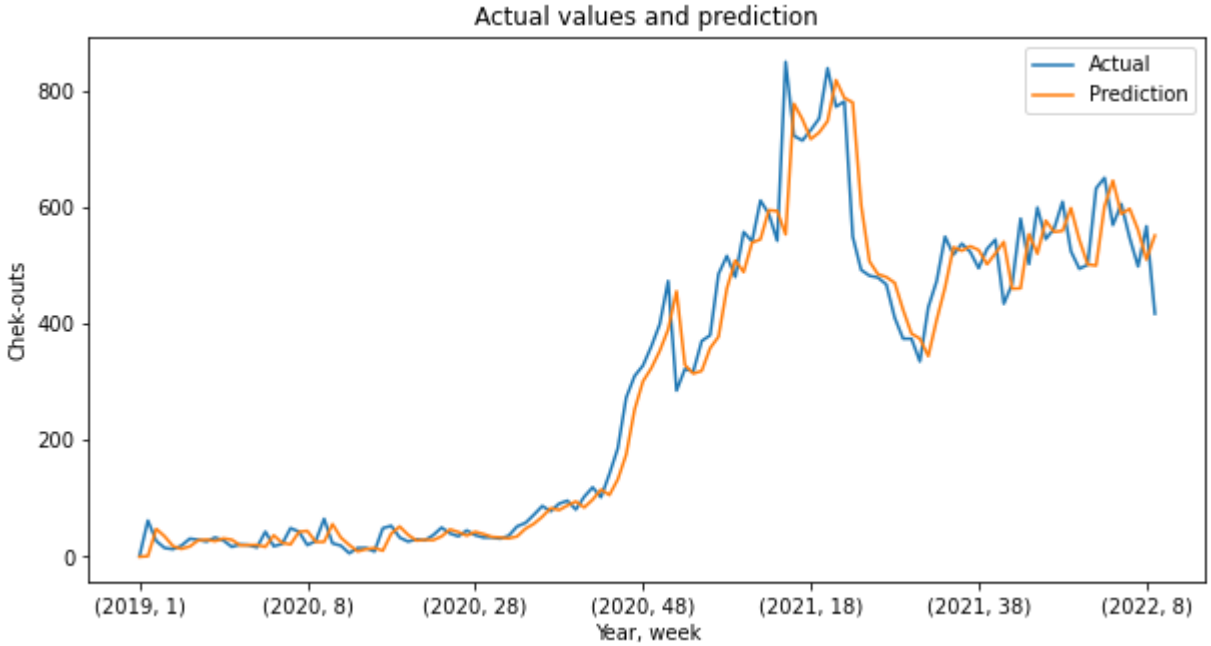
Out[22]:

year_week	count_	check_ins	date	corona	7_day
(2022, 5)	605.0	510.0	2022-02-06	4138.0	2855.857143
(2022, 6)	547.0	543.0	2022-02-13	3566.0	2926.571429
(2022, 7)	498.0	524.0	2022-02-20	2842.0	2471.000000
(2022, 8)	567.0	570.0	2022-02-27	2325.0	1922.571429
(2022, 9)	417.0	453.0	2022-03-01	3028.0	1782.285714

In [24]:
```
# surpress arnings
warnings.filterwarnings("ignore")
# ARIMA model
from statsmodels.tsa.arima.model import ARIMA
arima_model=ARIMA(ops['count_'], order=(1,1,1))
model=arima_model.fit()

print(model.summary())

fig = plt.figure(figsize=(10,5))
plt.title('Actual values and prediction')
plt.plot(ops['count_'].values ,label='Actual')
```

```
pd.Series(model.fittedvalues).plot(label='Prediction')
plt.ylabel('Chek-outs')
plt.xlabel('Year, week')
plt.legend()
plt.show()
```

```
                               SARIMAX Results
==============================================================================
Dep. Variable:                  count_   No. Observations:                  122
Model:                 ARIMA(1, 1, 1)   Log Likelihood                -662.195
Date:                Wed, 06 Jul 2022   AIC                           1330.390
Time:                        06:53:59   BIC                           1338.777
Sample:                             0   HQIC                          1333.796
                                  - 122
Covariance Type:                  opg
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
ar.L1         -0.2149      0.422     -0.509      0.610      -1.041       0.612
ma.L1         -0.0206      0.417     -0.049      0.961      -0.837       0.796
sigma2      3317.7059    211.614     15.678      0.000    2902.950    3732.462
===================================================================================
Ljung-Box (L1) (Q):                   0.01   Jarque-Bera (JB):               243.38
Prob(Q):                              0.93   Prob(JB):                         0.00
Heteroskedasticity (H):              15.81   Skew:                             0.34
Prob(H) (two-sided):                  0.00   Kurtosis:                         9.91
===================================================================================

Warnings:
[1] Covariance matrix calculated using the outer product of gradients (complex-step).
```

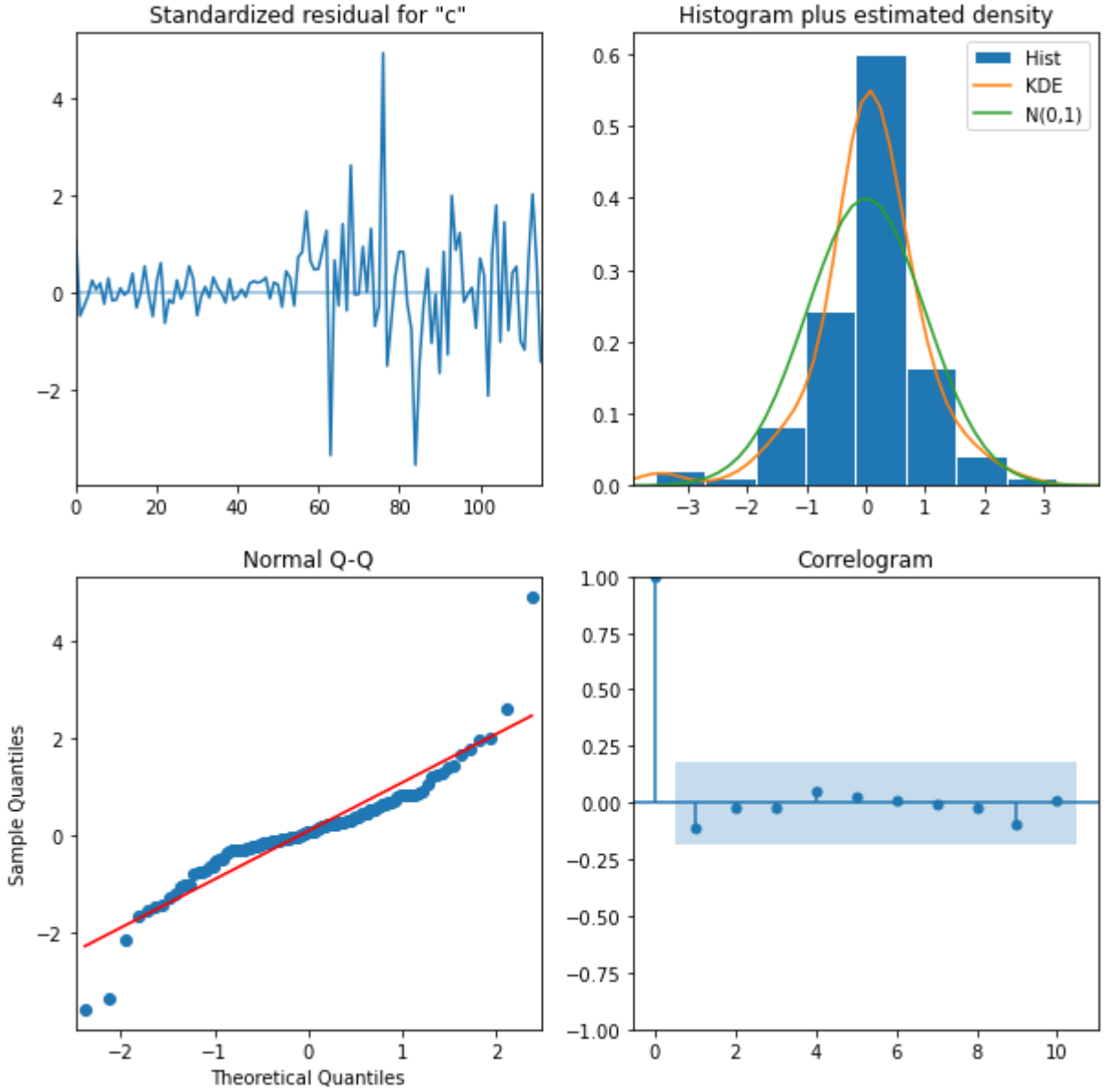


In [25]:

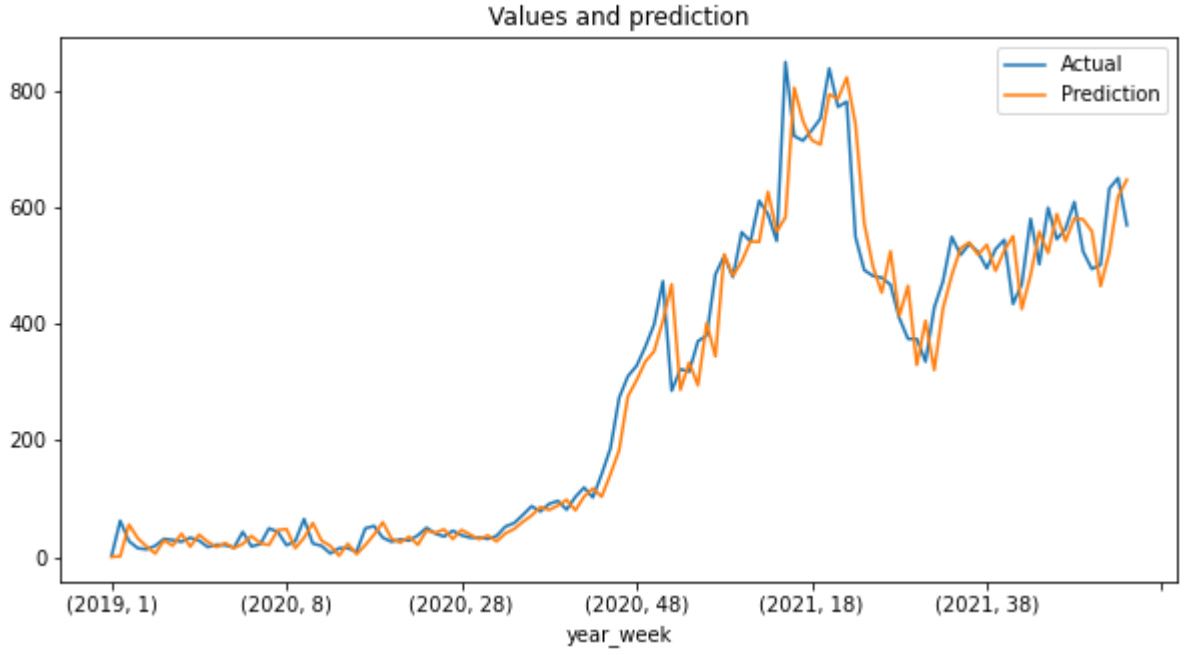
```
#split data for train and test
train_break=(2022,4)

train=ops[:train_break]['count_']

test=ops[train_break:]['count_']


# Grid search to find best parameters

best=10**1000
m_=()
for q in range(3):

    for p in range(3):
        #print(p,1,q)
        arima_model=ARIMA(train,  order=(p,1,q))
        model=arima_model.fit()
        #print(model.aic)
        if model.aic<best:
            best=model.aic
            m_=(p,1,q)
print('....result...')
print(m_, best)
```

```
arima_model=ARIMA(train,  order=m_)
model=arima_model.fit()
print(model.summary())

p=model.plot_diagnostics(figsize=(10,10))
```

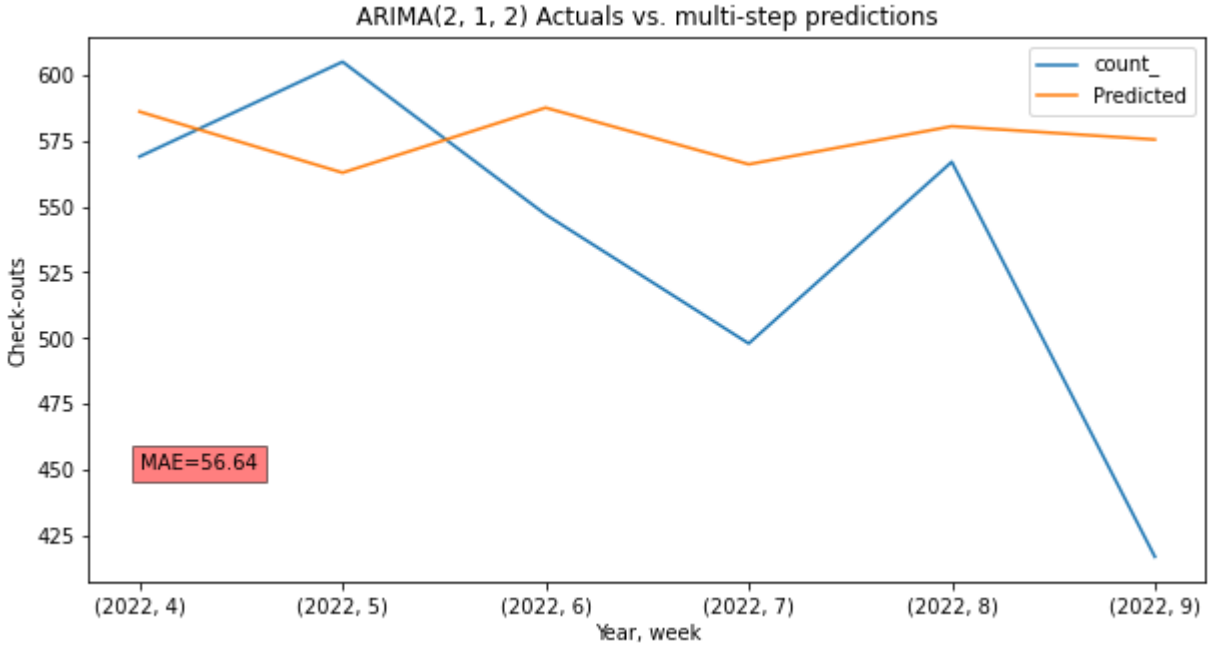
```
....result...
(2, 1, 2) 1266.6275291588727
                               SARIMAX Results
==========================================================================================
Dep. Variable:                     count_   No. Observations:                  117
Model:                 ARIMA(2, 1, 2)   Log Likelihood                -628.314
Date:                Wed, 06 Jul 2022   AIC                           1266.628
Time:                        06:54:28   BIC                           1280.395
Sample:                             0   HQIC                          1272.217
                                - 117
Covariance Type:                  opg
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
ar.L1         -1.7648      0.055    -32.336      0.000      -1.872      -1.658
ar.L2         -0.9516      0.049    -19.539      0.000      -1.047      -0.856
ma.L1          1.7571      0.082     21.341      0.000       1.596       1.918
ma.L2          0.8985      0.084     10.673      0.000       0.734       1.064
sigma2      2938.3103    250.424     11.733      0.000    2447.489    3429.132
===================================================================================
Ljung-Box (L1) (Q):                   1.51   Jarque-Bera (JB):               177.24
Prob(Q):                              0.22   Prob(JB):                         0.00
Heteroskedasticity (H):              12.71   Skew:                             0.29
Prob(H) (two-sided):                  0.00   Kurtosis:                         9.03
===================================================================================

Warnings:
[1] Covariance matrix calculated using the outer product of gradients (complex-step).
```



In [26]:
```
# Plot
fig = plt.figure(figsize=(10,5))
plt.title('Values and prediction')
train.plot(label='Actual')
pd.Series(model.fittedvalues).plot(label='Prediction')
plt.legend()
plt.show()
```



In [27]:
```
y_pred=pd.Series(model.forecast(steps=test.shape[0]).values, index=test.index, name='Predicted')
y_true=test
mae=np.mean(np.abs(y_pred-y_true)) # mean absolute error
df = pd.concat([y_true, y_pred], axis=1)

df.plot(figsize=(10,5))
plt.title('ARIMA{} Actuals vs. multi-step predictions'.format(m_))
plt.ylabel('Check-outs')
plt.xlabel('Year, week')
plt.text(0, 450, s='MAE={}'.format(round(mae,2)), bbox=dict(facecolor='red', alpha=0.5))
plt.show()
```



In [28]:
```
mape=np.mean(np.abs(y_pred-y_true)/np.abs(y_true)) #mean absolute percentage error
mae=np.mean(np.abs(y_pred-y_true)) # mean absolute error
mpe=np.mean((y_pred-y_true)/y_true) # mean percentage error
rmse= np.mean((y_pred-y_true)**2)**0.5
corr=np.corrcoef(y_pred,y_true)[0,1]

n=train.shape[0]
d=np.abs(train.diff()).sum()/(n-1)
errors=np.abs(y_true-y_pred)
mase=errors.mean()/d
print('mase:',mase)
print('mae;',mae)
```

```
mase: 1.6039691242212037
mae; 56.63670286905216
```

ARIMA with a exegenous regressor

In [29]:
```
ops_c=ops.loc[(2020,9):(2021,39)] # dropna of corona cases
ops_c
```

Out[29]:

year_week	count_	check_ins	date	corona	7_day
(2020, 9)	27.0	30.0	2020-03-01	2.0	1.666667
(2020, 10)	65.0	24.0	2020-03-08	10.0	4.142857
(2020, 11)	23.0	26.0	2020-03-15	89.0	34.142857
(2020, 12)	19.0	35.0	2020-03-22	104.0	80.142857
(2020, 13)	6.0	4.0	2020-03-29	87.0	78.142857
...	...	...	...	...	...
(2021, 35)	518.0	462.0	2021-09-05	244.0	246.285714
(2021, 36)	537.0	434.0	2021-09-12	214.0	199.571429
(2021, 37)	523.0	434.0	2021-09-19	195.0	169.428571
(2021, 38)	495.0	460.0	2021-09-26	147.0	118.571429
(2021, 39)	528.0	438.0	2021-10-03	184.0	104.000000

81 rows × 5 columns

In [34]:

```
train_break=(2021,33)
target_train=ops_c[:train_break]['count_']
exog_train=ops_c[:train_break][['7_day']]
model = ARIMA(
                target_train,
                exog_train,
                order=(2, 1, 2)).fit()



print(model.summary())
p=model.plot_diagnostics(figsize=(10,10))
```

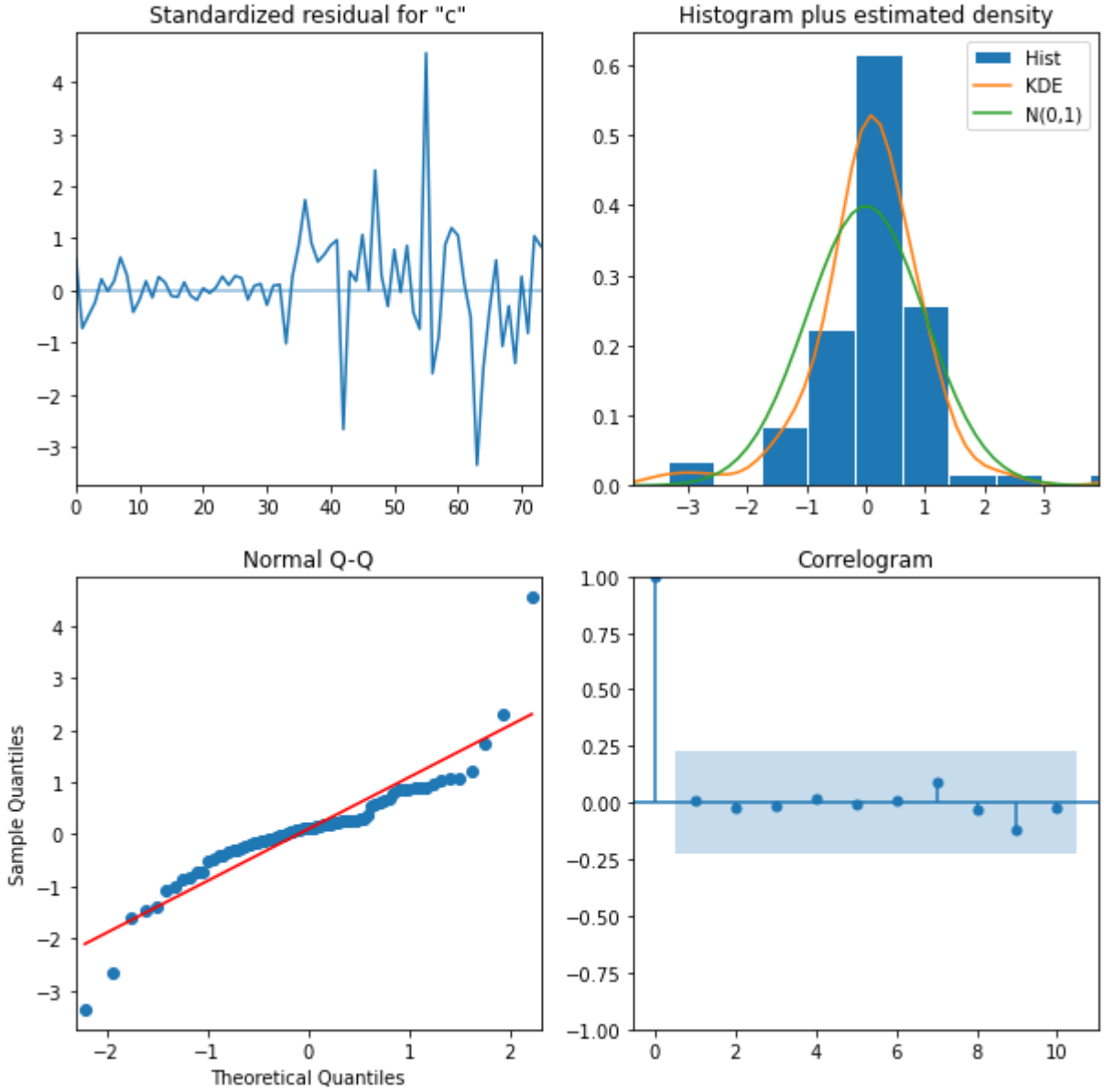
SARIMAX Results

Dep. Variable:	count_	No. Observations:	75
Model:	ARIMA(2, 1, 2)	Log Likelihood	-405.704
Date:	Wed, 06 Jul 2022	AIC	823.409
Time:	06:55:30	BIC	837.233
Sample:	0	HQIC	828.924
	- 75		
Covariance Type:	opg		

	coef	std err	z	P>|z|	[0.025	0.975]
7_day	0.2914	0.168	1.731	0.083	-0.039	0.621
ar.L1	-1.6726	0.123	-13.627	0.000	-1.913	-1.432
ar.L2	-0.9090	0.142	-6.391	0.000	-1.188	-0.630
ma.L1	1.5574	0.197	7.905	0.000	1.171	1.944
ma.L2	0.7667	0.207	3.704	0.000	0.361	1.172
sigma2	3361.1647	417.220	8.056	0.000	2543.429	4178.901

Ljung-Box (L1) (Q):	0.01	Jarque-Bera (JB):	116.16
Prob(Q):	0.93	Prob(JB):	0.00
Heteroskedasticity (H):	21.69	Skew:	0.43
Prob(H) (two-sided):	0.00	Kurtosis:	9.08

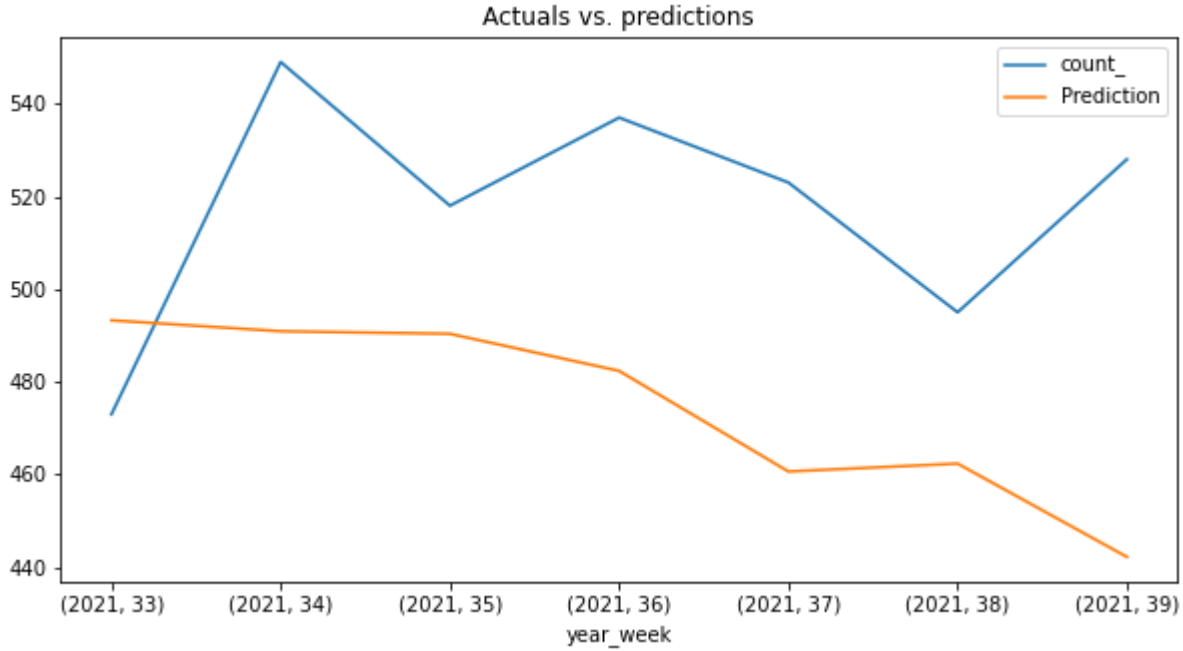
Warnings:
[1] Covariance matrix calculated using the outer product of gradients (complex-step).



In [35]:
```
predictions_=model.forecast(steps=ops_c.dropna()[train_break:].shape[0], exog=list(ops_c.dropna()[train_break:][['7_day']].valu
#predictions_
# create a df
y_pred=pd.Series(predictions_, index=ops_c.dropna()[train_break:].index, name='Prediction')
y_true=ops_c.dropna()[train_break:]['count_']


mape=np.mean(np.abs(y_pred-y_true)/np.abs(y_true)) #mean absolute percentage error
```

```
mae=np.mean(np.abs(y_pred-y_true)) # mean absolute error
mpe=np.mean((y_pred-y_true)/y_true) # mean percentage error
rmse= np.mean((y_pred-y_true)**2)**0.5
corr=np.corrcoef(y_pred,y_true)[0,1]


print('Mean Absolute Error',mae)

df = pd.concat([y_true, y_pred], axis=1)

df.plot(figsize=(10,5))
plt.title('Actuals vs. predictions')
plt.show()
```

```
Mean Absolute Error 48.777094311325776
```



In [36]:

```
# Grid search to find best parameters
# Check the contrinution of corona

best=10**1000
train_break=(2021,38)
target_train=ops_c[:train_break]['count_']
exog_train=ops_c[:train_break][['7_day',]]
m_=()
df=pd.DataFrame()
```

```
for q in range(3):

    for p in range(3):
        #print('........start.......')
        #print(p,1,q)
        arima_model=ARIMA(target_train,
                exog_train,
            order=(p,1,q))
        model=arima_model.fit()

        #print(model.summary())
        #print(model.aic)
        sum_ = pd.read_html(model.summary().tables[1].as_html(),header=None,index_col=0)[0]
        #print('p-value',sum_.iloc[1,3])
        #print('coeff',sum_.iloc[1,0])
        df.loc[str((p,1,q)),'corona_p_value']=sum_.iloc[1,3]
        df.loc[str((p,1,q)),'corona_coeff']=sum_.iloc[1,0]
        df.loc[str((p,1,q)),'AIC']=model.aic
        if model.aic<best:
            best=model.aic
            m_=(p,1,q)
df.index.name='order'
print('....result...')
#print(df)
print('Best:',m_, best)

arima_model=ARIMA(target_train,
                exog_train,
            order=m_)
model=arima_model.fit()
print(model.summary())

df
```

....result...
Best: (2, 1, 2) 873.7415703721825

SARIMAX Results

Dep. Variable:	count_	No. Observations:	80
Model:	ARIMA(2, 1, 2)	Log Likelihood	-430.871
Date:	Wed, 06 Jul 2022	AIC	873.742
Time:	06:55:34	BIC	887.958
Sample:	0	HQIC	879.437
	- 80		
Covariance Type:	opg		

	coef	std err	z	P>|z|	[0.025	0.975]
7_day	0.2964	0.154	1.919	0.055	-0.006	0.599
ar.L1	-1.7001	0.102	-16.603	0.000	-1.901	-1.499
ar.L2	-0.9410	0.110	-8.568	0.000	-1.156	-0.726
ma.L1	1.5941	0.172	9.280	0.000	1.257	1.931
ma.L2	0.8160	0.168	4.845	0.000	0.486	1.146
sigma2	3169.9763	364.304	8.701	0.000	2455.954	3883.999

Ljung-Box (L1) (Q):	0.00	Jarque-Bera (JB):	141.50
Prob(Q):	0.96	Prob(JB):	0.00
Heteroskedasticity (H):	20.61	Skew:	0.44
Prob(H) (two-sided):	0.00	Kurtosis:	9.50

Warnings:
[1] Covariance matrix calculated using the outer product of gradients (complex-step).

Out[36]:

order	corona_p_value	corona_coeff	AIC
(0, 1, 0)	0.028	0.3230	876.725190
(1, 1, 0)	0.047	0.3114	875.763232
(2, 1, 0)	0.060	0.3091	877.361755
(0, 1, 1)	0.041	0.3147	876.212028
(1, 1, 1)	0.051	0.3107	877.523763
(2, 1, 1)	0.068	0.3067	879.333982
(0, 1, 2)	0.070	0.3064	877.311024
(1, 1, 2)	0.032	0.3270	878.303073
(2, 1, 2)	0.055	0.2964	873.741570

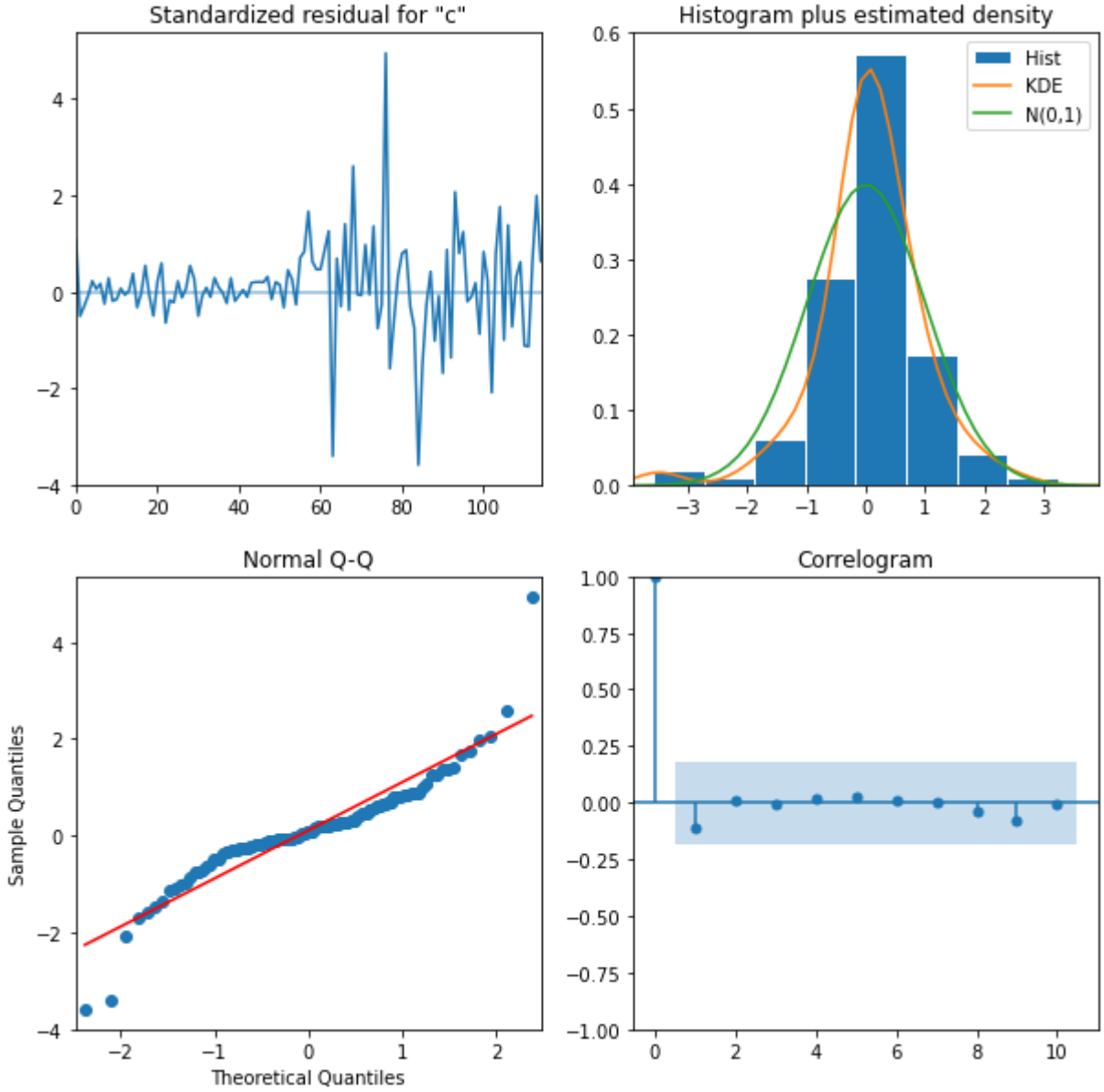
Rolling prediction without Exog

In [37]:

```
pred_steps=6 # pass how many rolling steps to be predicted


# Grid search to find best parameters
train_break=(ops.tail(pred_steps+1).head(1).index.values)[0]
best=10**1000

target_train=ops[:train_break]['count_']
#exog_train=ops[:train_break][['7_day']]
m_=()
df=pd.DataFrame()
for q in range(3):

    for p in range(3):
        #print('........start.......')
        #print(p,1,q)
        arima_model=ARIMA(target_train,
                #exog_train,
            order=(p,1,q))
        model=arima_model.fit()
```

```
        if model.aic<best:
            best=model.aic
            m_=(p,1,q)

print('....result...')

print('Best:',m_, best)

arima_model=ARIMA(target_train,
                  #exog_train,
              order=m_)
model=arima_model.fit()
print(model.summary())
p=model.plot_diagnostics(figsize=(10,10))
```

```
....result...
Best: (2, 1, 2) 1254.611550270068
                               SARIMAX Results
==============================================================================
Dep. Variable:                  count_   No. Observations:                  116
Model:                 ARIMA(2, 1, 2)   Log Likelihood                -622.306
Date:                Wed, 06 Jul 2022   AIC                           1254.612
Time:                        06:55:40   BIC                           1268.336
Sample:                             0   HQIC                          1260.182
                                - 116
Covariance Type:                  opg
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
ar.L1         -1.7828      0.049    -36.701      0.000      -1.878      -1.688
ar.L2         -0.9639      0.043    -22.373      0.000      -1.048      -0.880
ma.L1          1.7853      0.072     24.874      0.000       1.645       1.926
ma.L2          0.9257      0.074     12.531      0.000       0.781       1.070
sigma2      2900.0429    248.868     11.653      0.000    2412.270    3387.816
===================================================================================
Ljung-Box (L1) (Q):                   1.51   Jarque-Bera (JB):               187.14
Prob(Q):                              0.22   Prob(JB):                         0.00
Heteroskedasticity (H):              12.54   Skew:                             0.29
Prob(H) (two-sided):                  0.00   Kurtosis:                         9.22
===================================================================================

Warnings:
[1] Covariance matrix calculated using the outer product of gradients (complex-step).
```

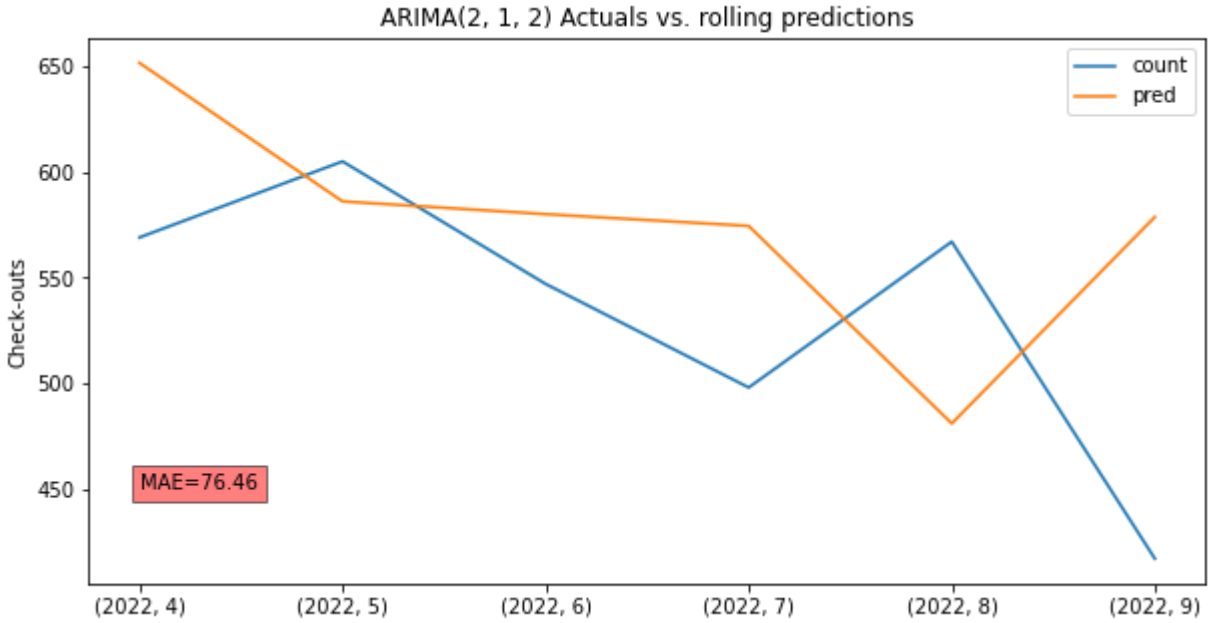


In [38]:
```
m_ # see the best order
```

Out[38]:
```
(2, 1, 2)
```

In [39]:
```
# Rolling prediction
df=pd.DataFrame()
```

```
for i,row in ops.reset_index().tail(pred_steps).iterrows():
    train_break=ops.reset_index().loc[i-1]['year_week']
    model=ARIMA(ops[:train_break]['count_'],order=m_).fit()
    predict=model.forecast(steps=1).values[0]
    df.loc[str(row['year_week']),'pred']=predict
df['count']=ops.tail(pred_steps)['count_'].values
mae=np.mean(np.abs(df['pred'].values-df['count'].values)) # mean absolute error
# plot
#print('Mean Absolute Error:',mae)
df[['count','pred']].plot( figsize=(10,5))
plt.ylabel('Check-outs')
plt.legend()
plt.title('ARIMA{} Actuals vs. rolling predictions'.format(m_))
plt.text(0, 450, s='MAE={}'.format(round(mae,2)), bbox=dict(facecolor='red', alpha=0.5))
plt.show()
```



What if:

Fitting an ARIMA Model on series after 1st Oct 2021

In [40]:
```
train_break=(2021,39)
best=10**1000

target_train=ops[train_break:]['count_']
```

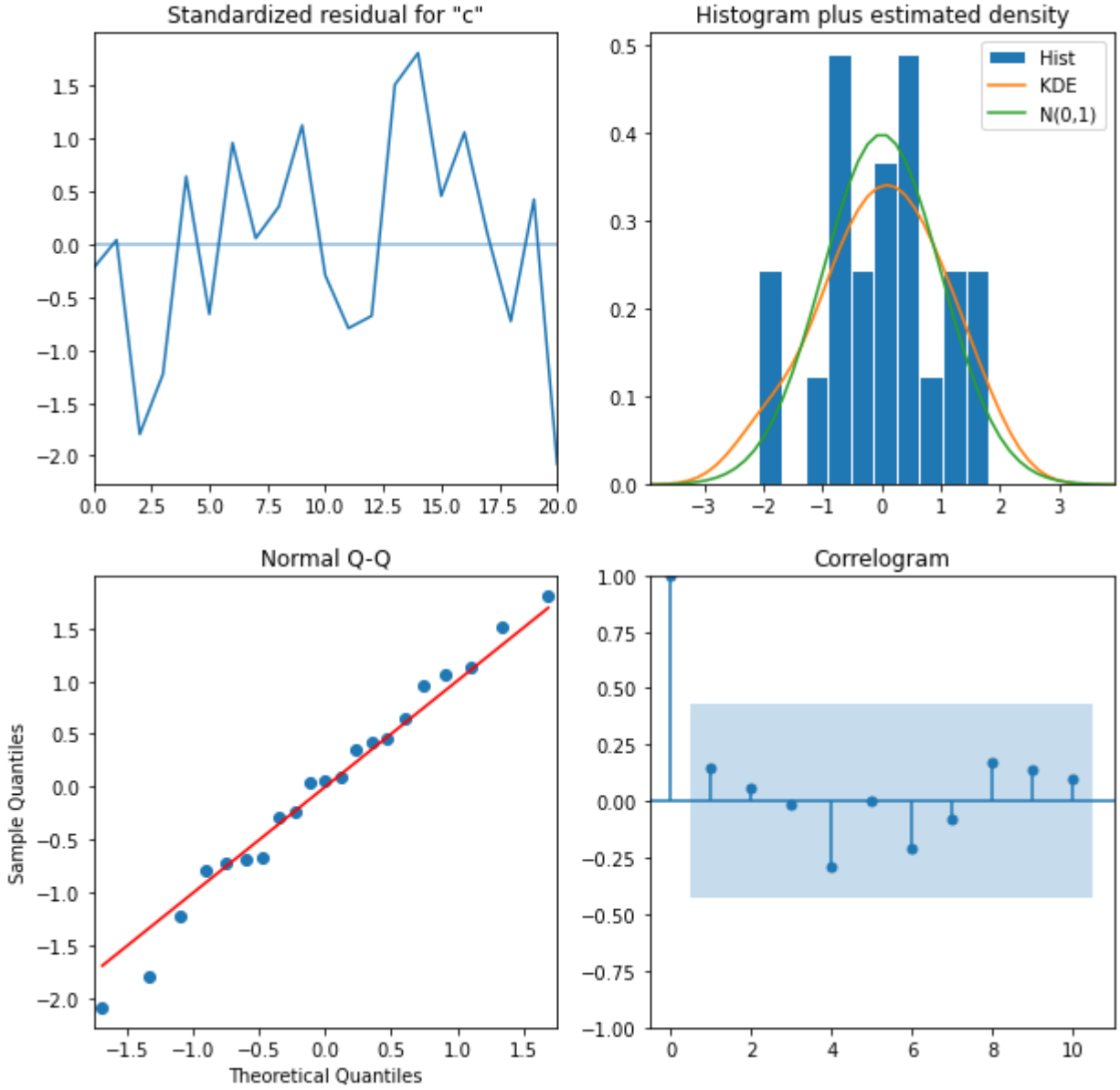
```
print('ADF test without differencing:',adfuller(target_train)[1]) # p-value is not significant.
```

```
ADF test without differencing: 0.025669272176180306
```

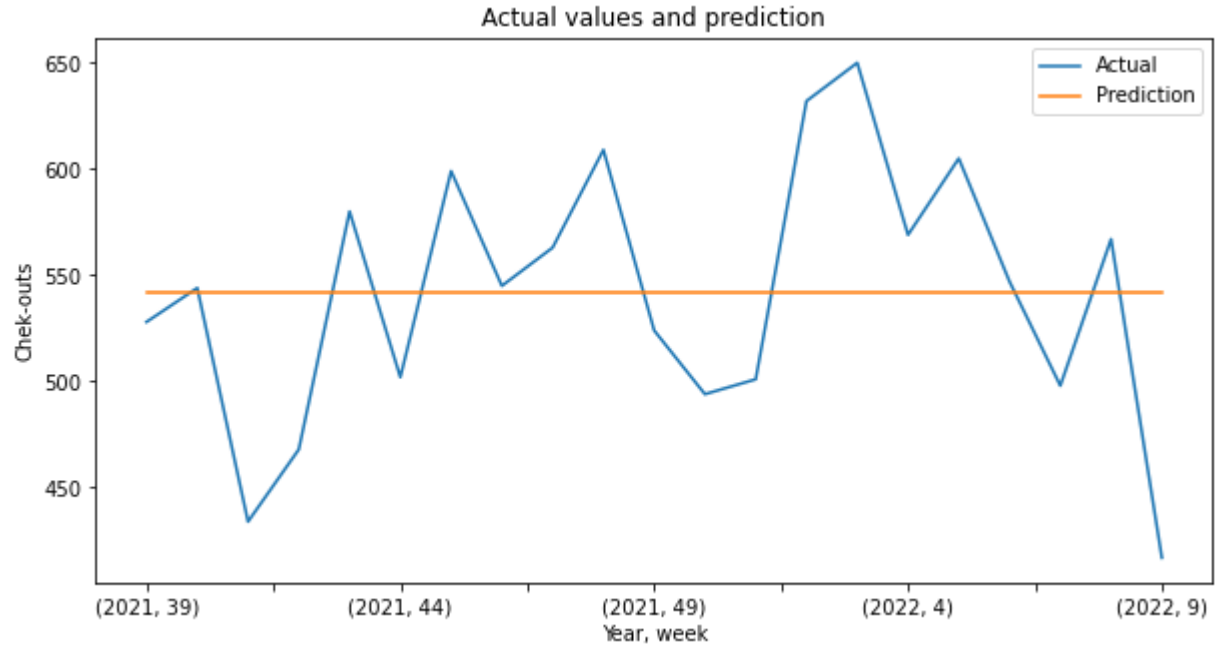
In [41]:

```
# Grid search to find best parameters


m_=()
df=pd.DataFrame()

for q in range(3):

    for p in range(3):


        #print('........start.......')
        #print(p,d,q)
        arima_model=ARIMA(target_train, order=(p,0,q))
        model=arima_model.fit()


        if model.aic<best:
            best=model.aic
            m_=(p,0,q)

print('....result...')

print('Best:',m_, best)

arima_model=ARIMA(target_train,

            order=m_)
model=arima_model.fit()
print(model.summary())
p=model.plot_diagnostics(figsize=(10,10))
```

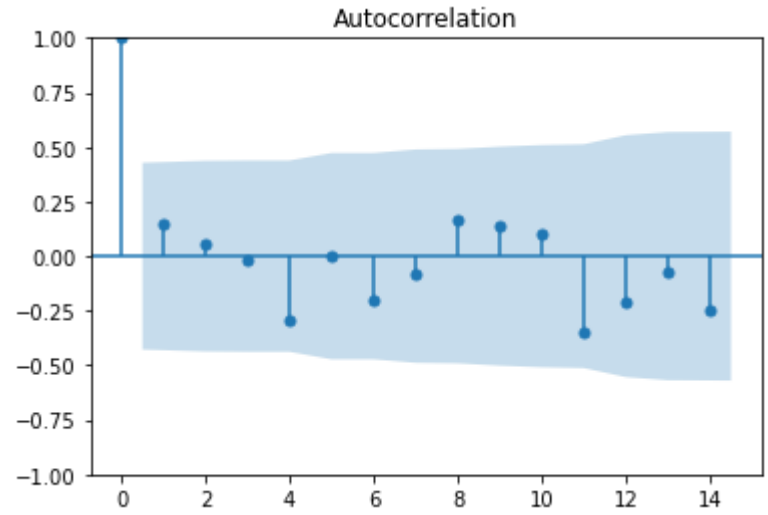
```
....result...
Best: (0, 0, 0) 235.54463185735972
                               SARIMAX Results
==============================================================================
Dep. Variable:                 count_   No. Observations:                   21
Model:                          ARIMA   Log Likelihood                -115.772
Date:                Wed, 06 Jul 2022   AIC                            235.545
Time:                        06:55:44   BIC                            237.634
Sample:                             0   HQIC                           235.998
                                  - 21
Covariance Type:                  opg
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
const        541.7143     13.340     40.608      0.000     515.568     567.861
sigma2      3597.6781   1314.254      2.737      0.006    1021.787    6173.569
===================================================================================
Ljung-Box (L1) (Q):                   0.51   Jarque-Bera (JB):                 0.43
Prob(Q):                              0.48   Prob(JB):                         0.81
Heteroskedasticity (H):               1.47   Skew:                            -0.24
Prob(H) (two-sided):                  0.62   Kurtosis:                         2.48
===================================================================================

Warnings:
[1] Covariance matrix calculated using the outer product of gradients (complex-step).
```



In [42]:
```
fig = plt.figure(figsize=(10,5))
plt.title('Actual values and prediction')
plt.plot(target_train.values ,label='Actual')
pd.Series(model.fittedvalues).plot(label='Prediction')
plt.ylabel('Chek-outs')
plt.xlabel('Year, week')
plt.legend()
plt.show()
```



In [43]:
```
p=plot_acf(target_train)
```



In [44]:
```
model.params
```

Out[44]:
```
const      541.714280
sigma2    3597.678072
dtype: float64
```

In [45]:
```
target_train.mean()
```

Out[45]:
```
541.7142857142857
```

In [ ]:
```

```