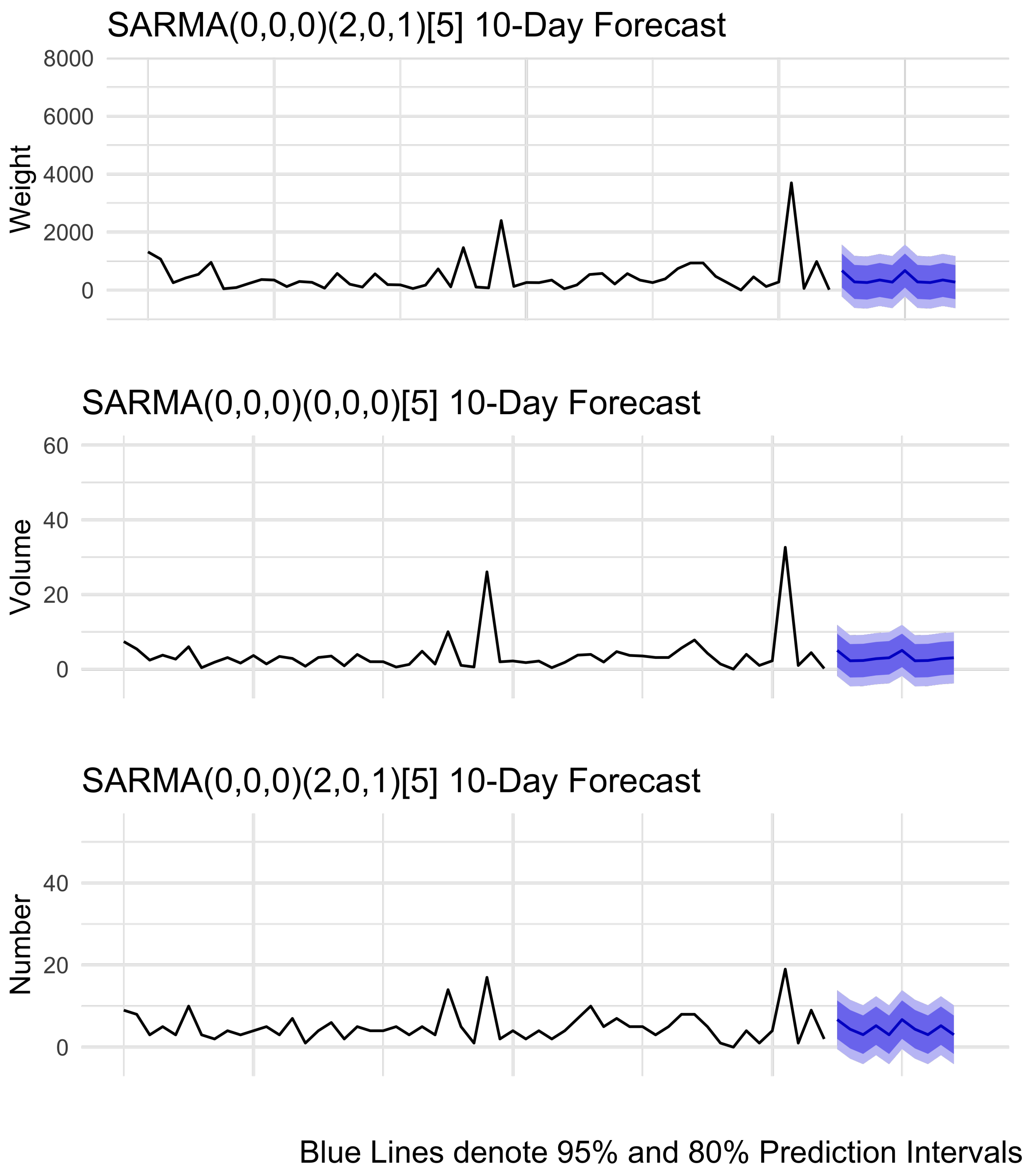
Project Report

Forecasting of Shipment Data and Vehicle Routing



Maastricht University
School of Business and Economics
Maastricht, June 30, 2022
Obbe Pulles (i6250802)
Tobias Schnabel (i6255807)
B.Sc. Econometrics and Operations Research
EBS2003: Second-Year Project II
Tutorial Group 1, Team 1

Contents

Introduction 1

1 Data Processing 2

2 Baseline Models 4

3 Seasonality and Model Identification 5

4 Model Estimation 7

5 Forecasting 10

6 Data Cleansing and Base Case 12

7 Finding good routing 13

8 Finding a local optimum 14

9 Business Report 17

10 Appendix A: Software Packages Used Including all Dependencies 19

11 Appendix B: Versions of Third-Party Software Packages Used 22

12 Appendix C: Source Code (Operations Research Part Data Cleaning) 23

Introduction

This report analyzes six months' worth of granular shipment data to provide insight on possible improvements of business practices on two fronts. Sections 1 through 5 are devoted to an econometric analysis of the seasonality properties of medium-range shipments between processing clusters. Sections 6 through 8 outline ways of algorithmically analyzing and improving the routing of trucks to pick up shipments from customers and transport them to the origin cluster. Our results are generalizable to broader sets of shipment data and could potentially also be used to optimize the vehicle routing at destination clusters to save costs on what is generally termed *last-mile* delivery. Our appendices provide all necessary information to reproduce all results starting from the raw data. In the interest of brevity, we proceed directly to our methodology.

1 Data Processing

We begin by identifying cluster 1 as our cluster of interest. The four most frequent destination clusters for this origin are clusters 15, 96, 206, and 12. We therefore subset our raw data to keep information on these four lanes. Subsequently, we aggregate the data to a daily frequency, add missing dates and replace missing values with 0. Finally, we drop weekends and convert our data to a balanced time series with frequency 5, that is, week-daily data, with a total of 204 observations across all 4 lanes.

Next, we use a custom function to generate dummy variable columns that are coded 1 for any observation where any of Weight W_t, Volume V_t, or Number N_t exceed their mean by x or more standard deviations, as given for $Weight$ by

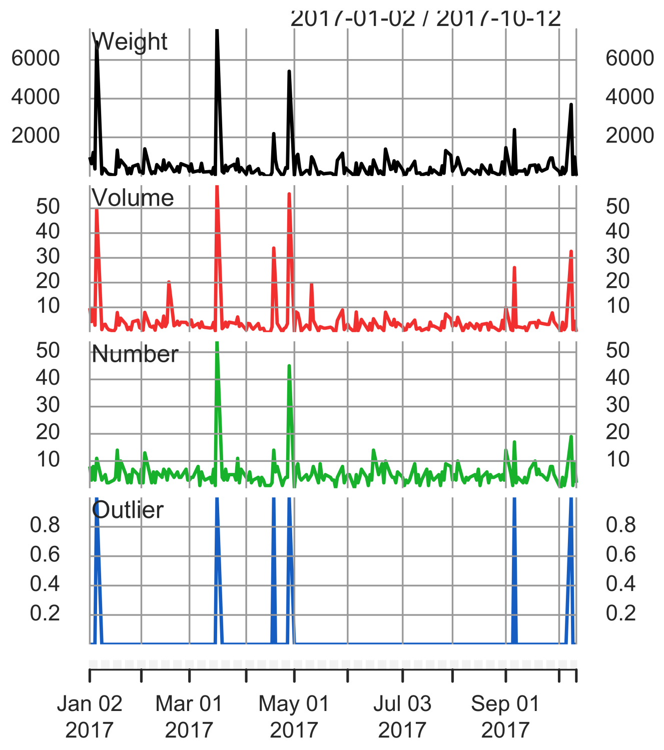
$$Flag_{Weight,i} = \begin{cases} 1 & \text{if } W_{t,i} \geq \mu_W + x \times \sigma_W \\ 0 & else \end{cases}$$

and use these three columns to create a fourth column:

$$Outlier = \begin{cases} 1 & \text{if } Flag_{Weight,i} = 1 \vee Flag_{Volume,i} = 1 \vee Flag_{Number,i} = 1 \\ 0 & \text{if } Flag_{Weight,i} = 0 \wedge Flag_{Volume,i} = 0 \wedge Flag_{Number,i} = 0 \end{cases}$$

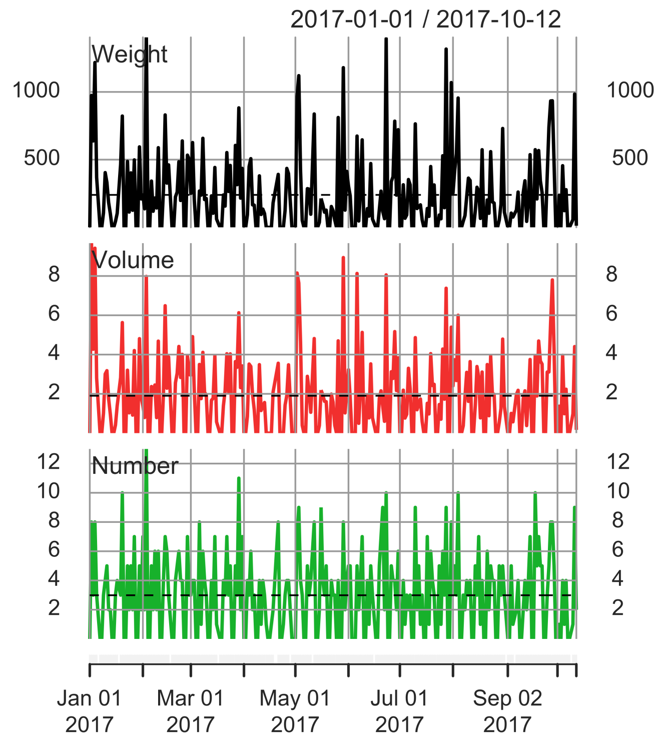
Figure 1 below demonstrates the result of this function for a threshold of $x = 2$ standard deviations, which is the value used to generate the final dataset used for the remainder of the analysis.

Figure 1: Outliers using a 2SD threshold



Having successfully identified all outliers, we plot a copy of our data that removes all outliers to gain an impression of whether they are stationary. Figure 2 below shows the three series and their respective means. All series fluctuate around their mean, we can discern no trend, hence we assume them to be stationary.

Figure 2: Stationarity



2 Baseline Models

To gain a rough understanding of the seasonality of these three series, we estimate the following linear models:

$$W_t = \alpha_0 + \alpha_1 Tuesday + \alpha_2 Wednesday + \alpha_3 Thursday + \alpha_4 Friday + \alpha_5 Outliers + \epsilon_t \tag{1}$$

$$V_t = \alpha_0 + \alpha_1 Tuesday + \alpha_2 Wednesday + \alpha_3 Thursday + \alpha_4 Friday + \alpha_5 Outliers + \epsilon_t \tag{2}$$

$$N_t = \alpha_0 + \alpha_1 Tuesday + \alpha_2 Wednesday + \alpha_3 Thursday + \alpha_4 Friday + \alpha_5 Outliers + \epsilon_t \tag{3}$$

We obtain the results reported in table 1 below. Across all three models, the estimated coefficient of the outliers are statistically significant at the 1% level. Of the weekdays, only Monday (the intercept) and Friday show significant estimated coefficients, which is indicative of a strong effect of these two days on the respective dependent variable. All models show high joint significance of estimated coefficients; $\bar{R}^2$ is noticeably lower for (3) than for (2) and (1).

Table 1: Baseline Regression Results

	Dependent variable:		
	Weight	Volume	Number
	OLS	*OLS*	*OLS*
	(1)	(2)	(3)
tuesday	−21.0671	0.0723	−1.3210
	(101.1564)	(0.7728)	(0.8079)
wednesday	67.3354	0.5663	0.8576
	(101.1564)	(0.7728)	(0.8079)
thursday	−6.6085	0.8012	−1.3454*
	(101.1564)	(0.7728)	(0.8079)
friday	391.4009***	2.7827***	2.3422***
	(102.1424)	(0.7803)	(0.8158)
Outlier	4,204.3500***	39.0742***	20.8399***
	(192.4248)	(1.4701)	(1.5368)
Constant	280.7378***	2.2310***	4.3698***
	(71.6053)	(0.5470)	(0.5719)
Observations	204	204	204
R^2	0.7343	0.7962	0.5524
Adjusted R^2	0.7276	0.7910	0.5411
Residual Std. Error (df = 198)	457.5120	3.4953	3.6540
F Statistic (df = 5; 198)	109.4428***	154.6670***	48.8782***
Note:			*p<0.1; **p<0.05; ***p<0.01

3 Seasonality and Model Identification

To identify seasonal dynamics more specifically, we first turn to two portmanteau tests for autocorrelation. Tables 2 and 3 below report test statistics and p-values for the LM- and Q-tests of the residuals of 1.

Table 2: LM Tests of Regression Residuals

	Lags	LM Statistic	p-value
Weight	25	39.649	0.03168
Volume	25	17.927	0.84547
Number	25	25.180	0.45234

Neither test finds statistically significant evidence for autocorrelation with the sole exception of $Weight$, the residuals of which show autocorrelation at the 5% significance level.

Table 3: Q-Tests of Regresion Residuals

	Q-Statistic	p-value
Weight, 5 Lags	4.802	0.44048
Volume, 5 Lags	2.820	0.72773
Number, 5 Lags	2.529	0.77206
Weight, 15 Lags	15.058	0.44728
Volume, 15 Lags	9.815	0.83121
Number, 15 Lags	15.609	0.40847
Weight, 25 Lags	21.232	0.67962
Volume, 25 Lags	20.052	0.74411
Number, 25 Lags	23.197	0.56605

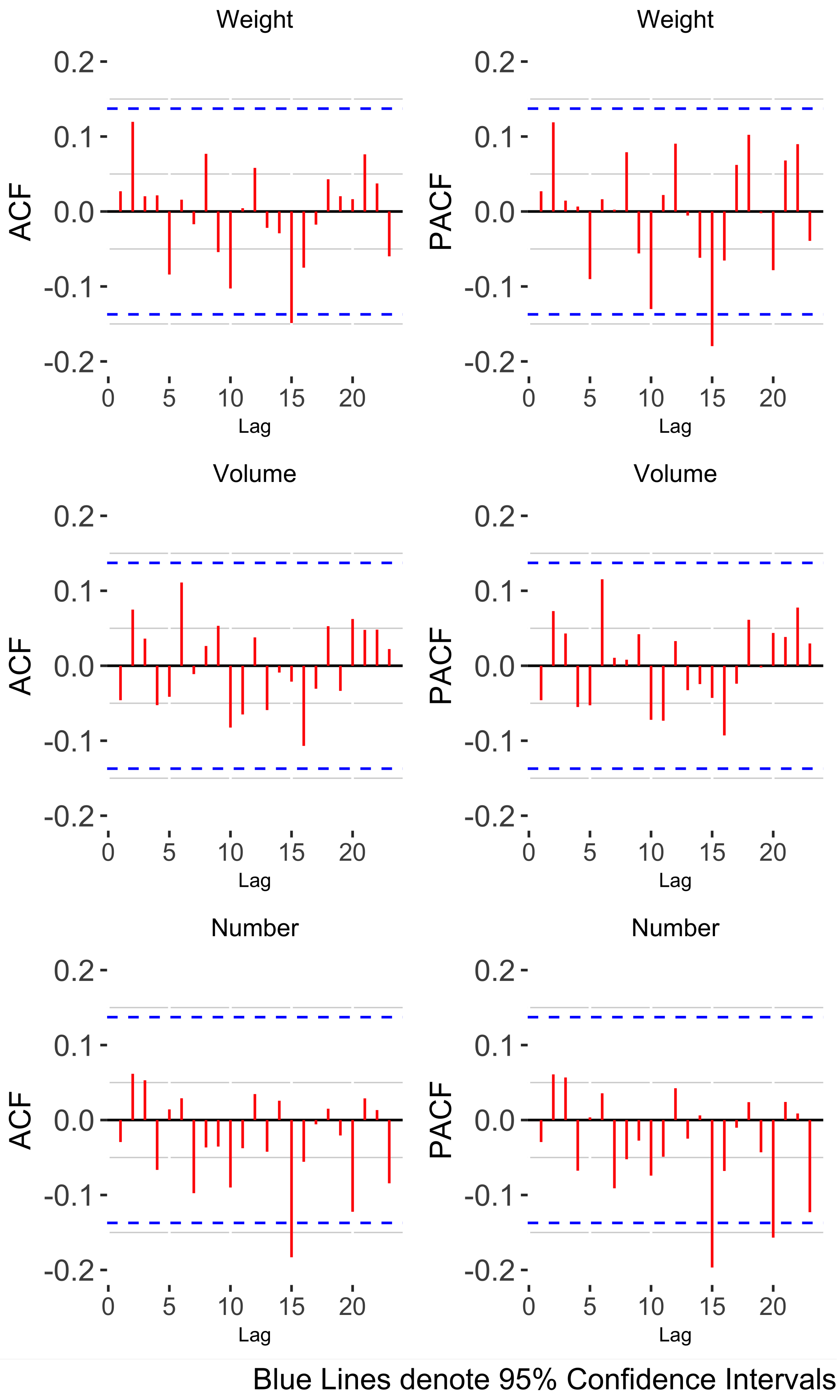
We note that the residuals of all three baseline regressions display heteroskedasticity at the 1% significance level, as shown in

Table 4: White Tests for Heteroskedasticity in Regresion Residuals

	Stat, No CT	p-value, No CT	Statistic, CT	p-value, CT
Weight	132.399	0.00000	153.664	0.00000
Volume	56.066	0.00000	153.664	0.00001
Number	115.968	0.00000	153.664	0.00000

We now visualize the (partial) autocorrelation factors of the residuals as shown in Figure 4. We cannot identify any patterns typical of autoregressive or moving-average processes in the three series generally. We also cannot identify any obvious sign of seasonality, though we note that significant (patial) autocorrelation appears at lags which are multiples of 5s, if they appear.

Figure 3: ACF / PACF at up to 25 Lags



4 Model Estimation

Before estimating $SARMA(p,q)(P,Q)$ specifications, we first re-estimate our three models as $SARMA(0,0(0,0)$ models to use as a baseline:

Table 5: ARMA(0,0)(0,0) Estimation Results

	Dependent variable:		
	Weight	Volume	Number
	ARIMA	*ARIMA*	*ARIMA*
	(1)	(2)	(3)
intercept	280.7378***	2.2310***	4.3698***
	(70.5445)	(0.5389)	(0.5634)
Outlier	4,204.3500***	39.0742***	20.8399***
	(189.5740)	(1.4483)	(1.5141)
tuesday	−21.0671	0.0723	−1.3210*
	(99.6578)	(0.7614)	(0.7959)
wednesday	67.3354	0.5663	0.8576
	(99.6578)	(0.7614)	(0.7959)
thursday	−6.6085	0.8012	−1.3454*
	(99.6578)	(0.7614)	(0.7959)
friday	391.4009***	2.7827***	2.3422***
	(100.6291)	(0.7688)	(0.8037)
Observations	204	204	204
Log Likelihood	−1,536.0820	−541.7077	−550.7674
σ^2	203,160.9000	11.8578	12.9592
Akaike Inf. Crit.	3,086.1650	1,097.4150	1,115.5350
Note:			*p<0.1; **p<0.05; ***p<0.01

Using R. Hyndman et al. (2022)'s *forecast::auto.arima*, we then identify the best-fitting specification for each model computationally, using the Akaike Information Criterion as our minimization objective. The optimal specifications we obtain are as follows:

$$Weight : SARMA(0,0,0)(2,0,1)[5]$$

$$Volume : SARMA(0,0,0)(0,0,0)[5]$$

$$Number : SARMA(0,0,0)(2,0,1)[5]$$

Table 7 below reports results. For $Weight$ and $Number$, the models we identified perform better (as measured by the Information Criteria in Table 6) than the $SARMA(0,0)(0,0)$ baseline results reported in Table 5, whereas the white-noise specification for $Volume$ performs best for this series. We attribute the latter fact to the absence of any significant (partial) autocorrelation in the series in our data. In all three specifications, the intercept and $Friday$ as well as the $Outliers$ are significant at the 1% level. In the specification for $Number$, we additionally now have significant estimated coefficients for $Tuesday$ and $Thursday$. In the models for $Weight$ and $Number$, the seasonal components are highly statistically significant.

Table 6: Information Criteria Comparison

	BIC Baseline	BIC Final	AIC Baseline	AIC Final
Weight	15.242	15.281	15.128	15.086
Volume	5.493	5.493	5.379	5.379
Number	5.582	5.619	5.468	5.456

Table 7: ARMA(p,q)(P,Q) Estimation Results

	Dependent variable:		
	Weight	Volume	Number
	ARIMA	*ARIMA*	*ARIMA*
	(1)	(2)	(3)
sar1	0.5582***		0.6363***
	(0.2064)		(0.2082)
sar2	−0.0988		−0.1783**
	(0.0969)		(0.0801)
sma1	−0.7263***		−0.6549***
	(0.2037)		(0.1997)
intercept	282.8321***	2.2310***	4.4237***
	(36.5122)	(0.5389)	(0.3636)
Outlier	4,151.9630***	39.0742***	21.0669***
	(192.3428)	(1.4483)	(1.5439)
tuesday	−25.5452	0.0723	−1.3545***
	(51.5271)	(0.7614)	(0.5133)
wednesday	39.9651	0.5663	0.6840
	(52.8975)	(0.7614)	(0.5199)
thursday	−8.1966	0.8012	−1.3919***
	(51.5516)	(0.7614)	(0.5133)
friday	380.5566***	2.7827***	2.4926***
	(53.0834)	(0.7688)	(0.5258)
Observations	204	204	204
Log Likelihood	−1,530.3360	−541.7077	−546.5224
σ^2	191,111.3000	11.8578	12.3890
Akaike Inf. Crit.	3,080.6720	1,097.4150	1,113.0450
Note:			*p<0.1; **p<0.05; ***p<0.01

We now test for a difference in daily effects, that is,

$$H_0 : \alpha_1 = \alpha_2 = \alpha_3 = \alpha_4 = 0$$

Table 6 below reports the F-statistic and associated p-values for these tests, the code for their computation can be found in the provided code. We reject H_0 only for $Weight$, pointing to a stronger effect of individual days in that series.

Table 8: F-Tests for Coefficient Restrictions

	F-statistic	95% Crit. Value	p-value
Weight	62016487.905	2.416	0.000
Volume	0.867	2.416	0.485
Number	0.924	2.416	0.451

Lastly, we test our specifications for misspecification. Although the results reported in Table 7 suggest that the residuals of our models are not normally distributed, Figures 5 and 5 suggests that they are at least approximately normally distributed and do not display significant autocorrelation (with the exception in the latter case of $Weight$).

Table 9: ARMA Residual Normality Test

	JB Statistic	JB p-value	Ljung-Box Statistic	LB p-value
Weight	1158.181	0.000	0.109	0.741
Volume	936.778	0.000	0.437	0.509
Number	2219.082	0.000	0.590	0.442

Figure 4: SARMA Misspecification: $Weight$

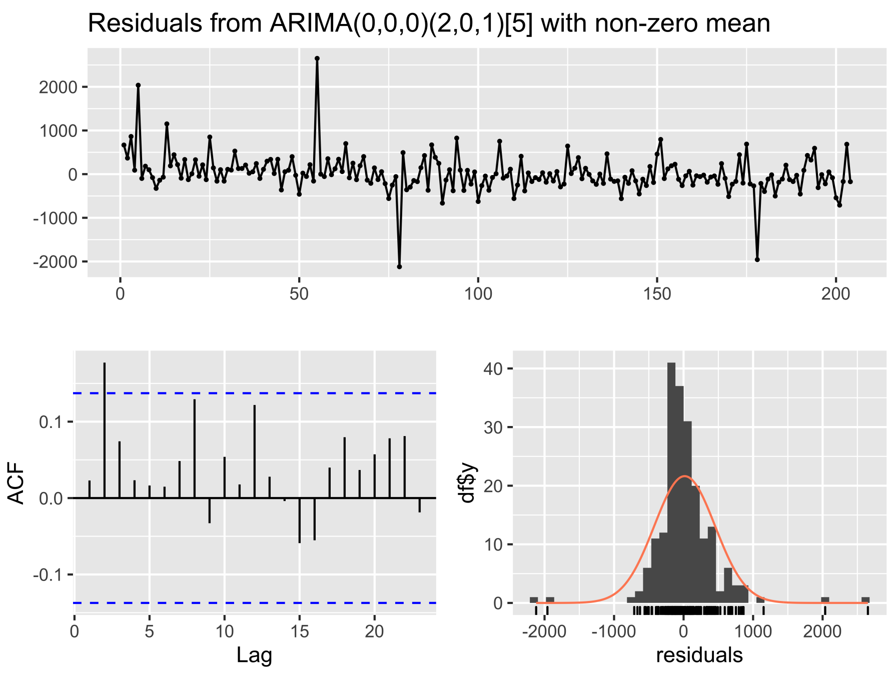
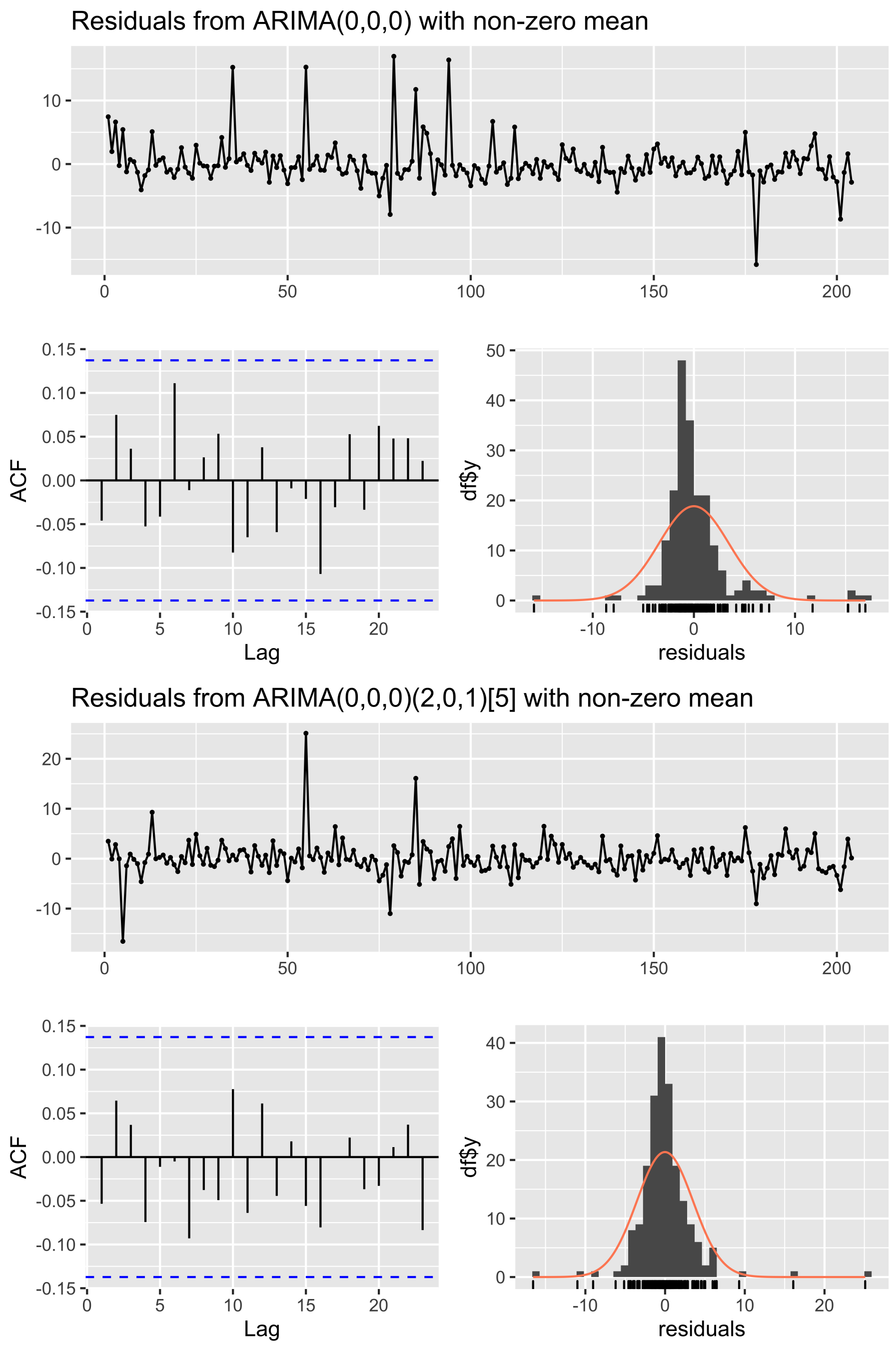


Figure 5: SARMA Misspecification: $Volume, Number$



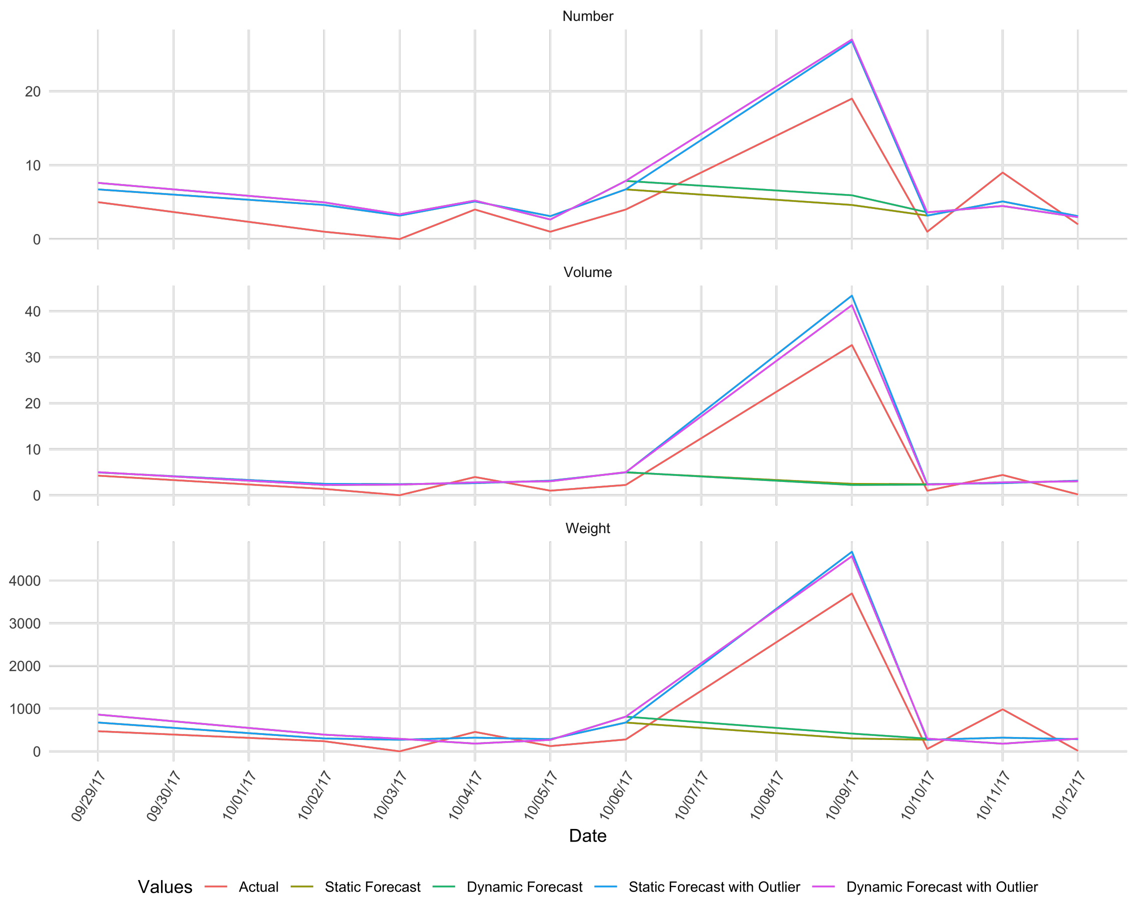
5 Forecasting

To forecast the last 10 days of data, we make a copy our data that omits the last 10 rows. We then use this copy to re-estimate our static models as well as our dynamic models. As the last 10 days of our data include an outlier identified by us, we repeat this process with a different set of forecast covariates that includes the information that this day is an outlier. Table 8 reports $RMSE$ values, we note that the dynamic model generally performs slightly better.

Table 10: RMSE of Forecast Models

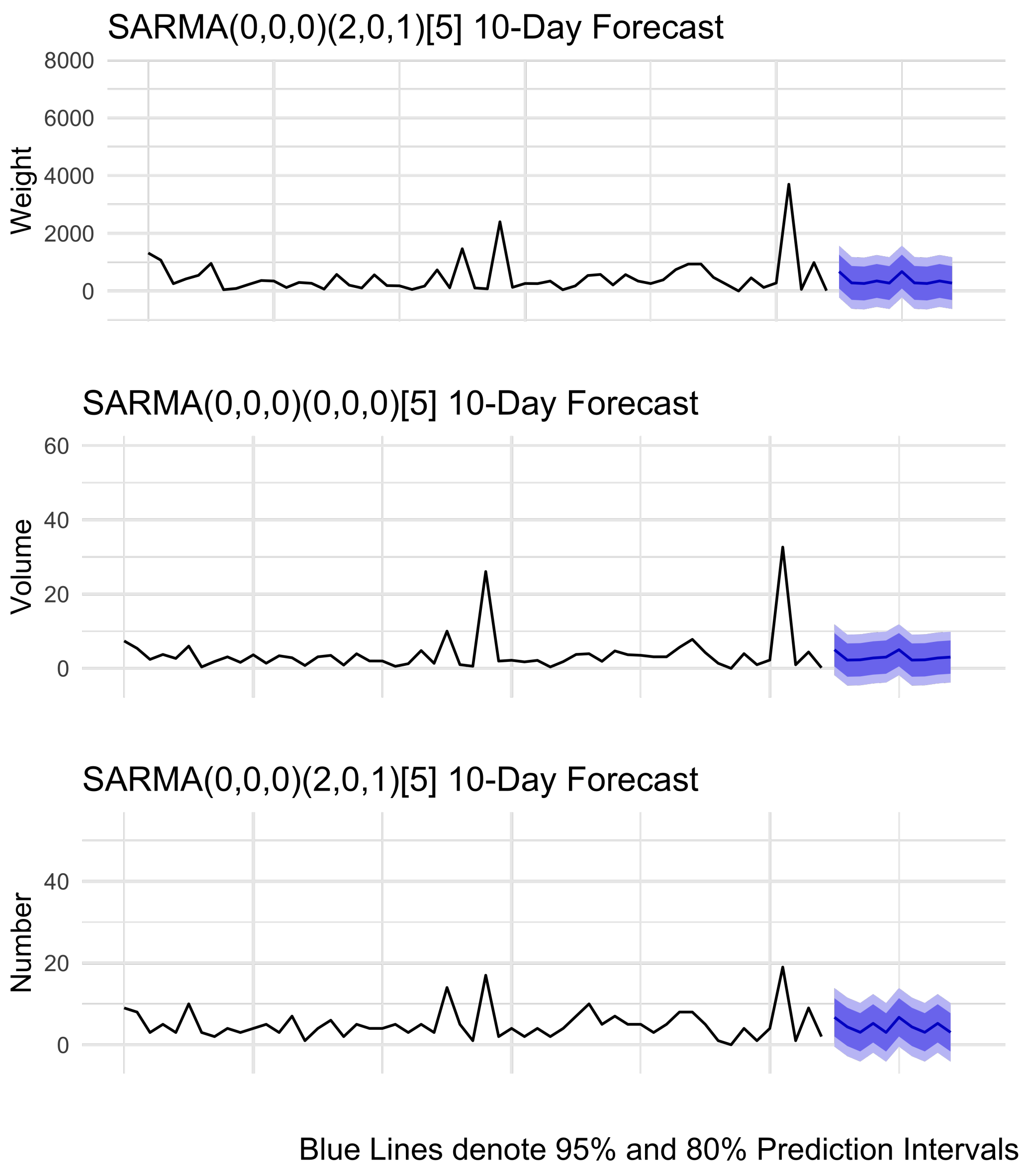
	Weight	Volume	Number
Static Forecast	1113.878	9.721	5.171
Dynamic Forecast	1104.308	9.781	5.021
Static with Outlier	428.418	3.886	3.469
Dynamic with Outlier	468.031	3.281	3.810

Figure 6: Forecast Comparison



Finally, we estimate our dynamic model over our full sample to forecast the next 10 observations, that is, 2 weeks into the future. Figure 7 visualizes the various forecasts per series. As we do not have actual data for these 10 future observations, we cannot forecast using a linear model. The reason why we can forecast our SARMA models is that they are based on lagged values of data, which enables us to estimate missing future values, which a linear model cannot do. Similarly, as we do not have the actual data for these 10 future observations, we cannot compute $RMSE$ of any kind.

Figure 7: Forecasts



6 Data Cleansing and Base Case

For the Operations Research part of this case, the first problem is to clean up the data and decide which cluster to use for the acquisition of the shipment data. As is suggested in the case description, we choose cluster 2 to be our cluster of interest and we select the relevant data by performing data cleansing in RStudio.

The initial stages of programming require us to plan ahead and choose which classes we want use. The obvious choice is to read the data into a "Shipment" class, which is exactly what we do. The code we use to retrieve the relevant data is provided in Appendix C. On closer inspection of the data, we find that many shipments store the same coordinates for their origin location so we decide on the creation of a new class "Customer" which stores relevant information which is specific to the customer, like their location and location code, which we use later as an identifier to select shipments and customers more efficiently. Last, we are required to have an object to store the shipments, we call this class "Truck". The truck has an identifier as well in order to make it easier to identify trucks while debugging. In the truck object, we store the shipments that are to be loaded in the truck as a shipment array-list, as well as the customer route which is also stored as an array-list. The choice of array-lists is an obvious one as they are dynamic when it comes to their size and we won't get issues regarding out-of-bounds indices. The last data we store in the truck is its current weight and volume, which is used to make sure that none of the 22000kg max weight and $82m^3$ max volume constraints are violated.

Now that all additional classes have been outlined, we start by reading the shipments into a large array-list containing all selected shipments on all selected dates. The base case itself is not particularly interesting as it involves creating a new truck for each shipment and sending it to the customer who has the shipment and going back to the cluster. The distance calculation is done by a haversine distance formula for distances on a sphere using the coordinates of two locations:

$$R = \text{radius of ball}$$

$$\phi_i,\ \psi_i,\ i = 1, 2$$

$$a = \sin^2(\Delta\phi/2) + \sin^2(\Delta\psi/2) * \cos(\phi_1) * \cos(\phi_2)$$

$$c = 2 * \arcsin(\sqrt{a})$$

$$\text{distance} = R * c$$

For a more adequate explanation on this, we invite the reader to have a look at the source code of the Shipment and Customer classes.

Given that this particular solution is not interesting, we do not include it within the source code, the storing of this process is only wasting space as the data is kept in memory and because it is a bad solution, it is better to immediately move on to a better routing heuristic in the next section.

7 Finding good routing

For an immediate improvement on the base case, we use a nearest neighbour approach. The idea is that we start by constructing a distance matrix for all customers to each other and to the cluster. We then initiate a new truck at the cluster and go to the nearest customer who has one or more shipment(s). We load the shipments into the truck and go from this customer, to its nearest neighbour who has one or more shipments until one of the capacity constraints is violated. We initiate a new truck at the cluster and repeat this process until there are no more shipments to be picked up. At the end we make all trucks return to the cluster.

To calculate the costs, we count how many trucks are used in total and multiply that amount by the fixed cost of 450, then to this total we add the total distance covered by the trucks and multiply this number by the variable cost of 1.5 to get the total cost of operation.

The total distance is calculated by summing up the route of each truck, which stores at the start and end the cluster, and in between those the customers it visits. By using the distance method from the Customer class, we calculate the individual distances of each route segment between index position i and index position $i-1$ for $1 \leq i < \text{route\_size}$ and sum all of these values up.

The algorithm makes sure that shipments are loaded one at a time, this means that when we are at a customer who has (for example) 4 shipments and we can only load 2 without violating a constraint, we do not remove that customer from the list of customers who still needs a truck for shipment transport but we keep those shipments, and thus the customer, in the list.

If one of the constraints is violated, we also do not immediately go back to the customer the previous truck ended at, as it is not certain whether this customer is in fact the nearest customer to the cluster, which is the starting point of a new truck.

When the list of shipments that are to be picked up is empty, we stop the program and return the costs. These results are a massive improvement over the previous ones as they save on truck spending because there are less trucks than in the base case. The route for each truck is longer in comparison, which is an obvious result of adding customers to the route of the truck. These improvements however, are still not optimal because the nearest neighbour algorithm is only a heuristic as it is obvious to save on costs.

Current shortcomings of this algorithm are manifold, however we focus only on two. A first issue is that the order of visiting the customers is not necessarily minimized because there may points in the route in which going to a customer who is further along the route first instead of the one who is closest, may actually make the route length shorter in the end. The improvement here is thus to exchange two points in the route, the case calls this type of improvement a "2-opt-exchange".

A second issue is that trucks that are designated as "full" may actually be able to carry more shipments still, that are lighter in weight and smaller in volume without violating any constraints. A solution to this is to see whether we can improve on the length of routes of two trucks by moving all shipments that belong to a customer from one truck to another truck and removing that customer from the former's route and adding it to the latter truck's route. The case calls this type of local search a "move".

8 Finding a local optimum

The difficulty of implementing these previous two improvements, 2-opt-exchange (2opt) and local search move (LSMove) depends greatly on the data structures used in solving the problem presented in the previous section. In our implementation of these local optimizations, we run into some issues regarding the data structures as well. Especially in finding and storing distance matrices and updating customers and shipments in lists quickly becomes rather confusing. However, by careful, step-by-step implementation of operations, we come to the following algorithms.

After we have gotten the initial routing from the previous section, we first implement the 2opt optimization because it is relatively easier compared to LSMove because we are looking at trucks one at a time instead of comparing and editing the properties of two trucks at a time. In pseudocode, the 2opt follows the following logic in Algorithm 1 below. Here, the reverse function of this algorithm reverses the route order from index i until index j. Meaning that if we have a route R = 1->2->5->3->4 and we want to reverse index 1 and 4, we get the route R* = 1->4->3->5->2. Furthermore, the updateRoute function updates the route that is stored in truck t. This way of reversing and swapping around the route effectively implements the 2opt algorithm and sometimes

Algorithm 1: Pseudocode 2opt

```
1  Take $t \in T$ a truck in the set of trucks;
2  Let R = Route(t);
3  Let bd = totaldistance(R);
4  Let foundImprovement = true;
5  Let newd = 0;
6  while foundImprovement do
7      foundImprovement = false;
8      for $1 \leq i < |R| - 2$ do
9          for $i + 1 \leq j < |R| - 1$ do
10             newR = reverse(R,i,j);
11             newd = totaldistance(newR);
12             if newd < bd then
13                 bd = newd;
14                 R = newR;
15 updateRoute(t,R);
16 return t;
```

leads to improvements in routing for trucks.

The second improvement to implement is the LSMove, which moves all shipments from a customer from one truck to another and checks whether there is an improvement of costs. In the algorithm, lines 10 and 11 perhaps require clarification. What is meant here is that we insert the customer "c" in position j from the route of truck 2, whose shipments we are considering moving from truck 2 to truck 1, into the route of truck 1 at position i, and we are removing that customer from the route of truck 2.

Algorithm 2: Pseudocode LSMove

```
1  Take $t_1, t_2 \in T$, where $t_1 \neq t_2$ are trucks;
2  Let $R_1$ =Route($t_1$), $R_2$ =Route($t_2$);
3  Let bd = totalDistance($R_1 + R_2$);
4  Let foundImprovement = true;
5  while foundImprovement do
6      foundImprovement = false;
7      for $1 \leq i < |R_1| - 1$ do
8          for $1 \leq j < |R_2| - 1$ do
9              Let c = R$_2$[j] gets customer in route 2 at position j;
10             Let $newR_1 = R_1 \cup c$ at position i;
11             Let $newR_2 = R_2 \backslash c$;
12             Let nd = totalDistiance($newR_1 + newR_2$);
13             if nd < bd AND no constraints are violated then
14                 bd = nd;
15                 Move all shipments with customer c from $t_2$ to $t_1$;
16                 $R_1 = newR_1, R_2 = newR_2$;
17                 foundImprovement = true;
18 updateRoute($t_1, R_1$);
19 updateRoute($t_2, R_2$);
20 return [$t_1, t_2$];
```

Finally, after implementing these algorithms, we end up with routes for trucks that are close to or at the local optimum. It is not necessarily the best solution possible, as we limit ourselves to the selection of the route made in the previous section and are thus constrained. But it is still better than the route we had before. Below are the results from the output of the program we have written for one day before local search optimizations and after local search optimizations showing long routes and a small improvement:

```
Date #10: 24/07/2017

Total cost of the feasible basic solution: 1493,39 EUR

Truck #1: Weight: 15,8/22 Volume: 78,77/82

Route: Cluster -> FR2055 > FR1207 > FR2006 > FR321 > FR34 > FR178 > FR4710 > FR5000 > FR3307 > FR743
    > FR47 > FR813 > FR3601 > FR2834 > FR1890 > FR953 > FR > Cluster


Truck #2: Weight: 3,6/22 Volume: 62,24/82

Route: Cluster -> FR > Cluster


Truck #3: Weight: 2,5/22 Volume: 38,34/82

Route: Cluster -> FR2754 > FR2845 > FR144 > FR3236 > FR2672 > FR3191 > FR849 > FR226 > Cluster


Optimal Total cost : 1491,58 EUR

Number of trucks used: 3

Method executed in 15 milliseconds, of which

Obtaining the feasible basic solution took 0 ms

Optimizing the basic solution took 15 ms
```

As we can see from the output, the optimization algorithms we employ do cut the cost of transport, so our implementation of these ideas is a correct starting point. For all results, we invite the reader to run the files that we provided in the assignment.

9 Business Report

This report serves as a *proof-of-concept* in helping the customer optimize their daily business practices. We show that medium-range shipments exhibit some seasonality, which can be exploited to forecast future demand with good accuracy in the short run. This enables the customers to pre-plan logistical efforts between clusters, which can be scaled to each week's requirements in an agile way. This analysis is extensible to long-range and perhaps also short-range shipments and ideally suited for permanent production implementation.

Our vehicle routing optimization program executes until all shipments to be picked up in the vicinity of a cluster have been assigned to a truck such that a truck starts and ends its route at the cluster origin. The basic solution uses a "Nearest Neighbour" approach, which makes it so that trucks go to the closest customer until they are full. We expand on this basic solution by recognizing that going to the nearest neighbour might not always be optimal. The first way to improve is by switching around the order of visited customers in the route of a single truck, and seeing whether it gives a route length improvement. Secondly, we check whether it is beneficial to move all shipments from a customer that are in one truck into another truck and see whether this move results in a reduction of the sum of the two routes of the trucks. We repeat this process until no more improvements can be made, in which case we call this solution the "local optimum".

The sample data we use to judge effectiveness of the program takes the busiest cluster from the provided shipment data, and takes those days on which there are more than 20 or more than 30 shipments to be picked up. Therefore, we have a sufficient benchmark for the performance of our program, as we are looking at some "worst case" scenarios for transporting shipments on busy days.

Our results are that of those 10 days, there are only 2 in which the local optimum does not improve on the basic solution. Given a fixed cost of €450 per truck and a variable cost of €1.5 per kilometer, the average amount we save by running the local optimum approach is approximately €10 per day. The running time of the algorithms is short, the computation of the basic solution for the 10 dates takes on average 1.6 milliseconds, while the computation of the local optimum averages 5.3 milliseconds. An example of the output of the program is given below.

Our methods are reliable and not cost-intensive. Given sufficient compute and robust data pipelines, they can be implemented at scale with reasonable accuracy, requiring moderate supervision. We are confident that both components can help the customer achieve substantial compounded cost savings, which can be (in part) productively redirected to the salaries of staff Econometricians.

```
Date: 26/05/2017

Total cost of the feasible basic solution: 1472,57 EUR

Truck #1: Weight: 2,5/22 Volume: 44,35/82

Route: Cluster > FR735 > FR849 > FR178 > Cluster


Truck #2: Weight: 5,1/22 Volume: 63,32/82

Route: Cluster -> FR > Cluster
```

```
Truck #3: Weight: 2,8/22 Volume: 60,04/82

Route: Cluster -> FR2055 > FR1890 > FR2834 > FR1808 > FR2754 > FR > Cluster


Optimal Total cost : 1455,96 EUR

Number of trucks used: 3

Method executed in 16 milliseconds, of which

Obtaining the feasible basic solution took 0 ms

Optimizing the basic solution took 16 ms
```

---

10 Appendix A: Software Packages Used Including all Dependencies

Zeileis, Achim and Torsten Hothorn (2002). "Diagnostic Checking in Regression Relationships". In: *R News* 2.3, pp. 7–10. URL: https://CRAN.R-project.org/doc/Rnews/.

Zeileis, Achim and Gabor Grothendieck (2005). "zoo: S3 Infrastructure for Regular and Irregular Time Series". In: *Journal of Statistical Software* 14.6, pp. 1–27. DOI: 10.18637/jss.v014.i06.

Hyndman, Rob J and Yeasmin Khandakar (2008). "Automatic time series forecasting: the forecast package for R". In: *Journal of Statistical Software* 26.3, pp. 1–22. DOI: 10.18637/jss.v027.i03.

Xie, Yihui (2014). "knitr: A Comprehensive Tool for Reproducible Research in R". In: *Implementing Reproducible Computational Research*. Ed. by Victoria Stodden, Friedrich Leisch, and Roger D. Peng. ISBN 978-1466561595. Chapman and Hall/CRC. URL: http://www.crcpress.com/product/isbn/9781466561595.

— (2015). *Dynamic Documents with R and knitr*. 2nd. ISBN 978-1498716963. Boca Raton, Florida: Chapman and Hall/CRC. URL: https://yihui.org/knitr/.

Wickham, Hadley (2016). *ggplot2: Elegant Graphics for Data Analysis*. Springer-Verlag New York. ISBN: 978-3-319-24277-4. URL: https://ggplot2.tidyverse.org.

Hamner, Ben and Michael Frasco (2018). *Metrics: Evaluation Metrics for Machine Learning*. R package version 0.1.4. URL: https://github.com/mfrasco/Metrics.

Dahl, David B. et al. (2019). *xtable: Export Tables to LaTeX or HTML*. R package version 1.8-4. URL: http://xtable.r-forge.r-project.org/.

Wickham, Hadley (2019). *stringr: Simple, Consistent Wrappers for Common String Operations*. R package version 1.4.0. URL: https://CRAN.R-project.org/package=stringr.

Wickham, Hadley, Mara Averick, et al. (2019). "Welcome to the tidyverse". In: *Journal of Open Source Software* 4.43, p. 1686. DOI: 10.21105/joss.01686.

Farrar, Thomas J. (2020). *skedastic: Heteroskedasticity Diagnostics for Linear Regression Models*. R Package Version 1.0.0. Bellville, South Africa. URL: https://github.com/tjfarrar/skedastic.

Henry, Lionel and Hadley Wickham (2020). *purrr: Functional Programming Tools*. R package version 0.3.4. URL: https://CRAN.R-project.org/package=purrr.

Kassambara, Alboukadel (2020). *ggpubr: ggplot2 Based Publication Ready Plots*. R package version 0.4.0. URL: https://rpkgs.datanovia.com/ggpubr/.

Pedersen, Thomas Lin (2020). *patchwork: The Composer of Plots*. R package version 1.1.1. URL: https://CRAN.R-project.org/package=patchwork.

Ryan, Jeffrey A. and Joshua M. Ulrich (2020). *xts: eXtensible Time Series*. R package version 0.12.1. URL: https://github.com/joshuaulrich/xts.

Arnold, Jeffrey B. (2021). *ggthemes: Extra Themes, Scales and Geoms for ggplot2*. R package version 4.2.4. URL: https://github.com/jrnold/ggthemes.

Csárdi, Gábor et al. (2021). *remotes: R Package Installation from Remote Repositories, Including GitHub*. R package version 2.4.2. URL: https://CRAN.R-project.org/package=remotes.

Dowle, Matt and Arun Srinivasan (2021). *data.table: Extension of ‘data.frame‘*. R package version 1.14.2. URL: https://CRAN.R-project.org/package=data.table.

Sax, Christoph (2021a). *tsbox: Class-Agnostic Time Series*. R package version 0.3.1. URL: https://CRAN.R-project.org/package=tsbox.

— (2021b). *tsbox: Class-Agnostic Time Series in in R*. R package. URL: https://www.tsbox.help.

Wickham, Hadley (2021a). *forcats: Tools for Working with Categorical Variables (Factors)*. R package version 0.5.1. URL: https://CRAN.R-project.org/package=forcats.

— (2021b). *tidyverse: Easily Install and Load the Tidyverse*. R package version 1.3.1. URL: https://CRAN.R-project.org/package=tidyverse.

Boshnakov, Georgi N. and Jamie Halliday (2022). *sarima: Simulation and Prediction with Seasonal ARIMA Models*. R package version 0.9. URL: https://CRAN.R-project.org/package=sarima.

Comtois, Dominic (2022). *summarytools: Tools to Quickly and Neatly Summarize Data*. R package version 1.0.1. URL: https://github.com/dcomtois/summarytools.

Farrar, Thomas (2022). *skedastic: Heteroskedasticity Diagnostics for Linear Regression Models*. R package version 1.0.4. URL: https://github.com/tjfarrar/skedastic.

Hlavac, Marek (2022). *stargazer: Well-Formatted Regression and Summary Statistics Tables*. R package version 5.2.3. URL: https://CRAN.R-project.org/package=stargazer.

Hothorn, Torsten et al. (2022). *lmtest: Testing Linear Regression Models*. R package version 0.9-40. URL: https://CRAN.R-project.org/package=lmtest.

Hyndman, Rob et al. (2022). *forecast: Forecasting Functions for Time Series and Linear Models*. R package version 8.16. URL: https://CRAN.R-project.org/package=forecast.

Müller, Kirill and Hadley Wickham (2022). *tibble: Simple Data Frames*. R package version 3.1.7. URL: https://CRAN.R-project.org/package=tibble.

R Core Team (2022). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing. Vienna, Austria. URL: https://www.R-project.org/.

Robinson, David, Alex Hayes, and Simon Couch (2022). *broom: Convert Statistical Objects into Tidy Tibbles*. R package version 0.8.0. URL: https://CRAN.R-project.org/package=broom.

Stoffer, David and Nicky Poison (2022). *astsa: Applied Statistical Time Series Analysis*. R package version 1.15. URL: https://CRAN.R-project.org/package=astsa.

Svetunkov, Ivan (2022). *greybox: Toolbox for Model Building and Forecasting*. R package version 1.0.5. URL: https://github.com/config-i1/greybox.

Trapletti, Adrian and Kurt Hornik (2022). *tseries: Time Series Analysis and Computational Finance*. R package version 0.10-51. URL: https://CRAN.R-project.org/package=tseries.

Wickham, Hadley and Jennifer Bryan (2022). *readxl: Read Excel Files*. R package version 1.4.0. URL: https://CRAN.R-project.org/package=readxl.

Wickham, Hadley, Winston Chang, et al. (2022). *ggplot2: Create Elegant Data Visualisations Using the Grammar of Graphics*. R package version 3.3.6. URL: https://CRAN.R-project.org/package=ggplot2.

Wickham, Hadley, Romain François, et al. (2022). *dplyr: A Grammar of Data Manipulation*. R package version 1.0.9. URL: https://CRAN.R-project.org/package=dplyr.

Wickham, Hadley and Maximilian Girlich (2022). *tidyr: Tidy Messy Data*. R package version 1.2.0. URL: https://CRAN.R-project.org/package=tidyr.

Wickham, Hadley, Jim Hester, and Jennifer Bryan (2022). *readr: Read Rectangular Text Data*. R package version 2.1.2. URL: https://CRAN.R-project.org/package=readr.

Xie, Yihui (2022). *knitr: A General-Purpose Package for Dynamic Report Generation in R*. R package version 1.39. URL: https://yihui.org/knitr/.

Zeileis, Achim, Gabor Grothendieck, and Jeffrey A. Ryan (2022). *zoo: S3 Infrastructure for Regular and Irregular Time Series (Z's Ordered Observations)*. R package version 1.8-10. URL: https://zoo.R-Forge.R-project.org/.

11 Appendix B: Versions of Third-Party Software Packages Used

Table 11: Versions of Packages used in this Report

	pkg	version
1	astsa	1.15
2	base	4.2.1
3	colorspace	2.0.3
4	curl	4.3.2
5	digest	0.6.29
6	ellipsis	0.3.2
7	evaluate	0.15
8	fansi	1.0.3
9	farver	2.1.0
10	forecast	8.16
11	fracdiff	1.5.1
12	glue	1.6.2
13	grateful	0.1.11
14	gtable	0.3.0
15	highr	0.9
16	isoband	0.2.5
17	knitr	1.39
18	labeling	0.4.2
19	lifecycle	1.0.1
20	lmtest	0.9.40
21	Metrics	0.1.4
22	munsell	0.5.0
23	pkgconfig	2.0.3
24	quadprog	1.5.8
25	quantmod	0.4.20
26	R6	2.5.1
27	RColorBrewer	1.1.3
28	Rcpp	1.0.8.3
29	RcppArmadillo	0.11.2.0.0
30	scales	1.2.0
31	stringi	1.7.6
32	tidyverse	1.3.1
33	timeDate	3043.102
34	tseries	0.10.51
35	TTR	0.24.3
36	urca	1.3.0
37	utf8	1.2.2
38	vctrs	0.4.1
39	viridisLite	0.4.0
40	withr	2.5.0
41	xfun	0.31
42	xts	0.12.1
43	yaml	2.3.5
44	zoo	1.8.10

12 Appendix C: Source Code (Operations Research Part Data Cleaning)

```r
##OR data cleaning

packages <- c("data.table", "dplyr", "zoo", "tidyr", "readxl", "summarytools")

installed_packages <- packages %in% rownames(installed.packages())
if (any(installed_packages == FALSE)) {
  install.packages(packages[!installed_packages])
}

#load packages
invisible(lapply(packages, library, character.only = TRUE))
rm(installed_packages)
Paths = c("/Users/ts/Git/Second-Year-Project-II-Econometrics",
↪  "C:/Users/obbep/Documents/R/Second-Year-Project-II-Econometrics/")
names(Paths) = c("ts", "obbep")
setwd(Paths[Sys.info()[7]])

ordat = as.data.table(read_excel('Data.xlsx'))
#location to cluster
#issue 1: shipment already at destination
#drop Lat-Long duplicates
orwdat = ordat[OriginClusterLat != OriginLat][OriginClusterLong != OriginLong]

#drop shipments over 22 tons
orwdat = orwdat[`TR Gross Weight (KG)` <= 20000][`TR Gross Volume (M3)` <= 82]

#sort on origin cluster and PU Date
setorder(orwdat, -`Nb of Ship Units`, -PUDate, na.last = TRUE)
#stview(dfSummary(orwdat))

#filter, drop unnecessary cols, rename vars, add counter var to id 20+/3-+ days, sort
orex = as_tibble(orwdat) %>%
  filter(OriginCluster == "Cluster2") %>%
  select(-c(2,4,7,8,9,16,17,18,19,20,28)) %>%
  rename(Weight = `TR Gross Weight (KG)`) %>%
  rename(Nb = `Nb of Ship Units`) %>%
  rename(EDay = `TR Pickup - Event Day`) %>%
  rename(Volume = `TR Gross Volume (M3)`) %>%
  rename(Date = PUDate) %>%
  rename(TRC = `TR Code`) %>%
  rename(SLC = `TR Source Location Code`) %>%
  rename(OrigID = OriginFull) %>%
  relocate(c(Date, Weight, Nb, Volume, SLC)) %>%
  add_count(Date, name = "NumPerDay") %>%
  relocate(NumPerDay, .after = Volume ) %>%
  arrange(desc(NumPerDay))

#grab more than 20
export1 = orex %>% filter(NumPerDay > 19) %>%
  filter(NumPerDay < 40) %>%
  arrange(desc(NumPerDay), Date)
#get dates
datelist1 = head(unique(export1$Date), 5)

#subset
export1 = export1 %>%
  filter(Date %in% datelist1) %>%
  distinct(SLC, Date, .keep_all = T) %>%
  select(-c(5,7,8,9,14,15,16,17,18,19))

#grab more than 30
export2 = orex %>% filter(NumPerDay > 30) %>%
  arrange(desc(NumPerDay), Date)
#get dates
datelist2 = head(unique(export2$Date), 5)

#subset
export2 = export2 %>%
  filter(Date %in% datelist2) %>%
  distinct(SLC, Date, .keep_all = T) %>%
  select(-c(5,7,8,9,14,15,16,17,18,19))

#combine
export = export1 %>%
  bind_rows(export2) %>%
  relocate(SLC, .after = Date)
#strip latter half of SLC identifier (first part is already unique)
SLC = export$SLC
SLC = sub(" .*", "", SLC)
SLC = sub("T011.", "", SLC)
export$SLC = SLC

#export to txt
print(length(union(datelist1, datelist2)))
write.table(export, file = "Data.txt", sep = " ",
            row.names = F, col.names = F)
```