Time Series Analysis (02417)

Assignment 3
Predicting the amount of legislation

Authors:
Grzegorz Wojciech Zaba, s213035
Andreas With Aspe, s174197
Evangelos Kalimantzalis Lianas, s210260
Francesco Maria D'Antiga, s212354

May 27, 2022

Student contributions

All students have participated actively in discussing and calculating the results presented in the sections of this report. The table below shows which students took the main responsibility of each question.

Question 2.1	Question 2.2	Question 2.3	Question 2.4	Question 2.5	Question 2.6
s210260	s213035	s174197	s174197	s212354	s210260

Question 2.1

Overview of data

Data was categorised by government and plotted. Results were presented in Figure 1. Number of words in danish legislation appears to grow exponentially as time follows. No matter what political party was in charge year by year amount of Danish legislation was clearly increasing. At first period, in years from 1919 to around 1950 trend showed rather linear properties, but after that it started grow significantly. Reason for this might have been the fact, that first described period was time of world wars, thus political environment could have been unstable. Concluding, inarguable divergence can be noticed on Figure 1, thus time series appears to be non-stationary. In Table 1 mean and variance values for first 20, 40, 60, 80 and 100 data points were calculated

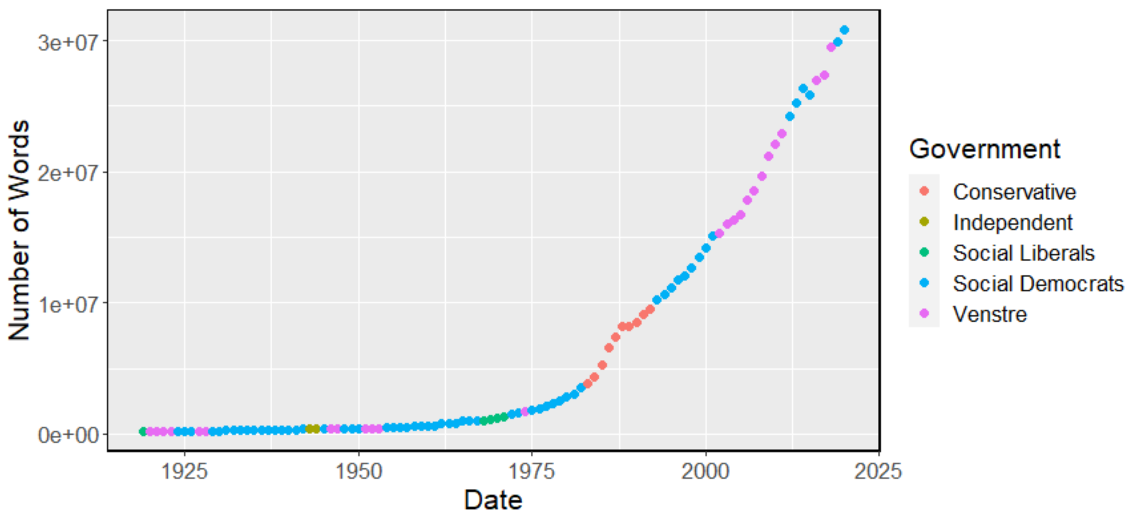

Figure 1: Total amount of words in the Danish legislation in function of year.

in order to prove hypothesis of series being non-stationary. Values of these moments are growing with growing number of data points included in their calculation, thus from the fact that both expected value and variance aren't stationary, non-stationarity of the process can be confirmed.

Number of data points	20	40	60	80	100
Mean	$2.3 \cdot 10^5$	$3.2 \cdot 10^5$	$6.3 \cdot 10^5$	$2.4 \cdot 10^6$	$6 \cdot 10^6$
Variance	$1.8 \cdot 10^9$	$1.2 \cdot 10^{10}$	$2.9 \cdot 10^{11}$	$1.2 \cdot 10^{13}$	$6.9 \cdot 10^{13}$

Table 1: Mean and variance development over time.

Data transformations

One can conclude, that due to process not being stationary some data transformation or differentiation shall be included while fitting ARIMA model. It was already said, that data seems to follow exponential trend, thus logarithmic transformation will be applied to whole data set. After transformation logarithm of first value was subtracted from all measurements in order to set starting point of series to zero. Therefore the transformation takes the following form:

$$\hat{Y}_t = \log(Y_t) - \mu \tag{1}$$

where μ turns out to be 12.91 in order to out the first data point to (0,0). When we have finished our analysis and made predictions, we will transform the data back.

Despite the fact, that log-transformed data are more likely to follow linear trend (Figure 2) than original data, the series is still non-stationary. Another proof of this fact as well as of

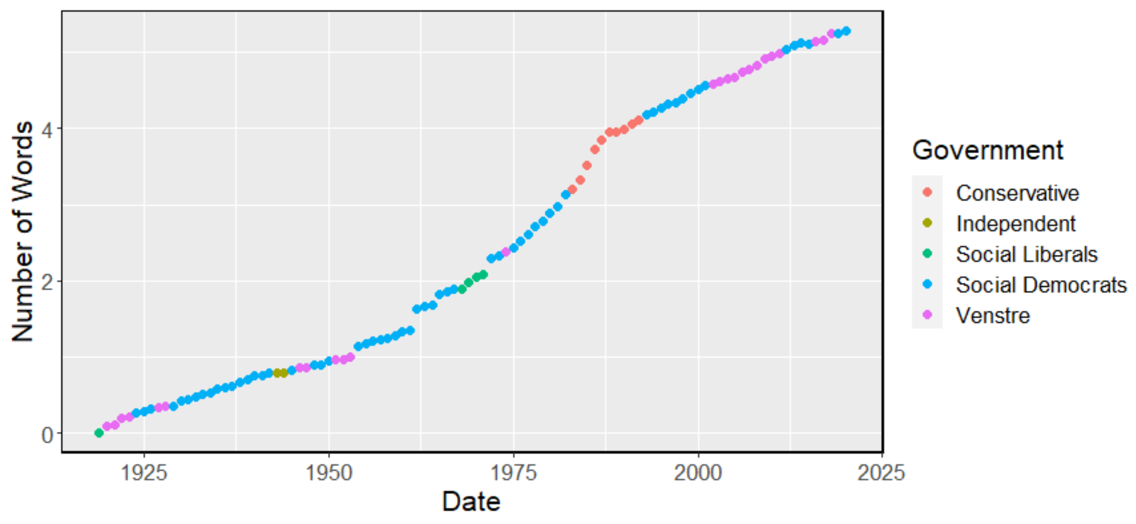

Figure 2: Time series after applying log transformation.

original data being non-stationary are ACF functions shown on Figure 3. Moreover, interpreting these plots can lead to deduction that no seasonality is expected to be included in model.

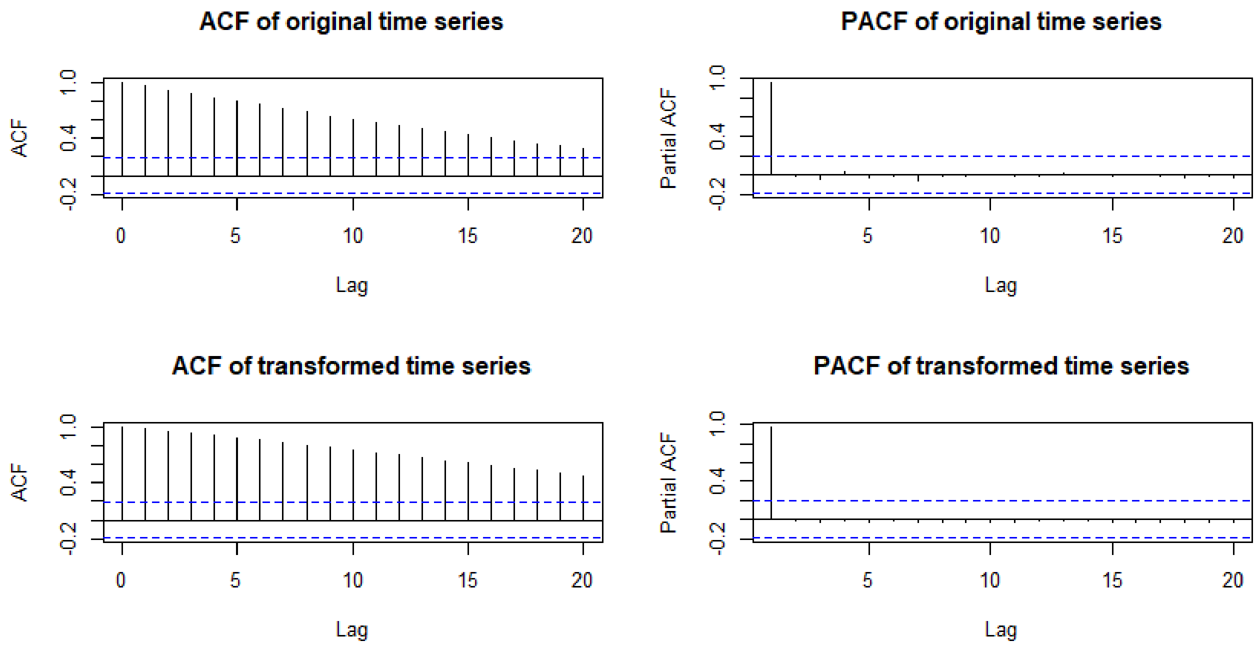

Figure 3: Auto- and partial autocorrelation functions of original and log-transformed data series.

Non-stationarity of the data can be handled by applying 1st order differentiation. Result of this type of transformation can be seen on Figure 17. Even though differentiation was applied on original data, the trend seems to still be non-stationary, while applying this transformation to logarithm data gives stationary result.

Another change to be considered is excluding some data from the beginning of analysed period. Due to unstable political scene in the world, caused by world wars measurements in that time can follow slightly different trend than after 1945. To let situation stabilize after second world war in further analysis data starting from 1950 only will be included. This won't alter stationarity of differentiated data, thus it's only shifting initial date from 1919 to 1950. Proof in form of plot can be seen in Appendix A. ACF and PACF funcitons of both transformed data including first 32 measurements and data without these values were calculated. Results can be seen on Figure 5. Even though after differenciation log-transformed data we can find stationarity in autocorrelation function, it performs slightly better when we eliminate year period 1919-1950. Some of spikes of ACF function in first case (whole dataset) are crossing threshold line, when in second case we don't see this behaviour even in first lags.

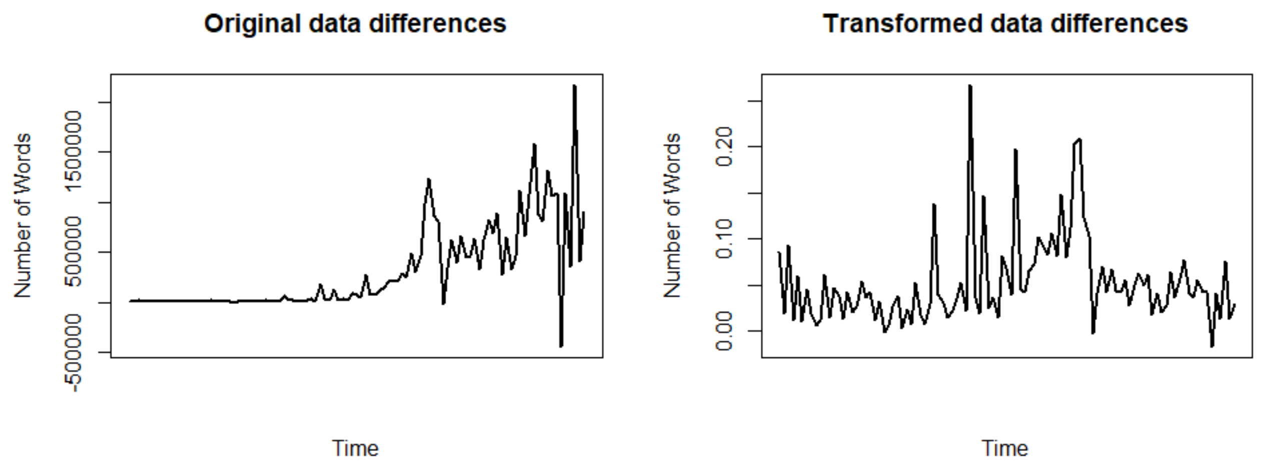

Figure 4: Differences of original data and log-transformed data.

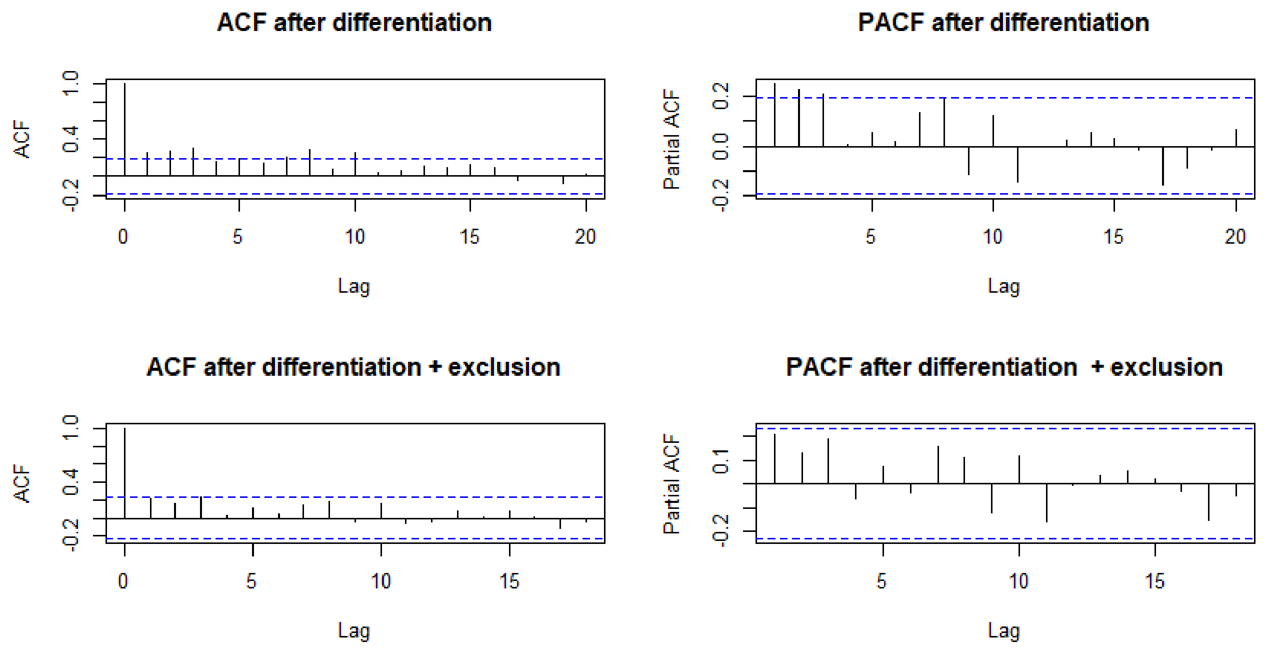

Figure 5: Auto- and partial autocorrelation functions of transformed data with and without exclusion of first 32 measurements.

Question 2.2

Taking all part 2.1 into consideration, data was transformed by taking logarithm of original values and differentiated once, thus d parameter in ARIMA(p,d,q) will be equal to 1. Since after data transformation first point of time series was set to 0, no intercept term in model will be included. No seasonal term will be included, what was justified in previous part, based on no seasonality observation in ACF and PACF functions. Based on Figure's 5 autocorrelation functions, one can conclude, that after 1st lag autocorrelation stabilizes, but it can be argued, whether 3rd lag spike exceeds threshold. Thus firstly models with moving average part (q) equal 1 and 3 will be tested. All of the spikes of PACF functions in mentioned figure can be considered equal to zero, although it's evolution can be interpreted as damped sinus oscillations since the beginning, hence AR part (p) equal to 1 will be considered.

ARIMA(1,1,3)

Model was investigated by looking at estimated coefficients, their statistical significance and ACF and PACF functions. Results in form of plots and output from R-studio are presented. Coefficient test was made with use of 'coeftest()' function from 'lmtest' library. This test calculates p-values for each coefficient in the model, to evaluate if they are significant due to a chance. The p-values are calculated from the null hypothesis that the coefficient is zero. If the value is above 5% we will then reject the hypothesis. From Figure 6 can be guessed, that ma2 and ma3 estimators can be excluded from our model, because their p-value is very close to 1. The next step is then to drop each coefficient at a time to see the difference. Sometimes dropping only one coefficient might change the scenario and make the other one significant. However, it turns out that when dropping one of the parameters the other will still be insignificant. Therefore we will next evaluate and present the results for the model while dropping these 2 coefficients, what will give ARIMA(1,1,1). ACF's and PACF's spikes for ARMA(1,1,3) do not all land within threshold, but they aren't substantially larger (Figure 7).

```
      Estimate Std. Error z value  Pr(>|z|)
ar1  0.9859409  0.0174719 56.4300 < 2.2e-16
ma1 -0.7991696  0.1278542 -6.2506 4.088e-10
ma2 -0.0212781  0.1948642 -0.1092    0.9130
ma3  0.0039606  0.1394654  0.0284    0.9773
```

Figure 6: Coefficients test for ARIMA(1,1,3).

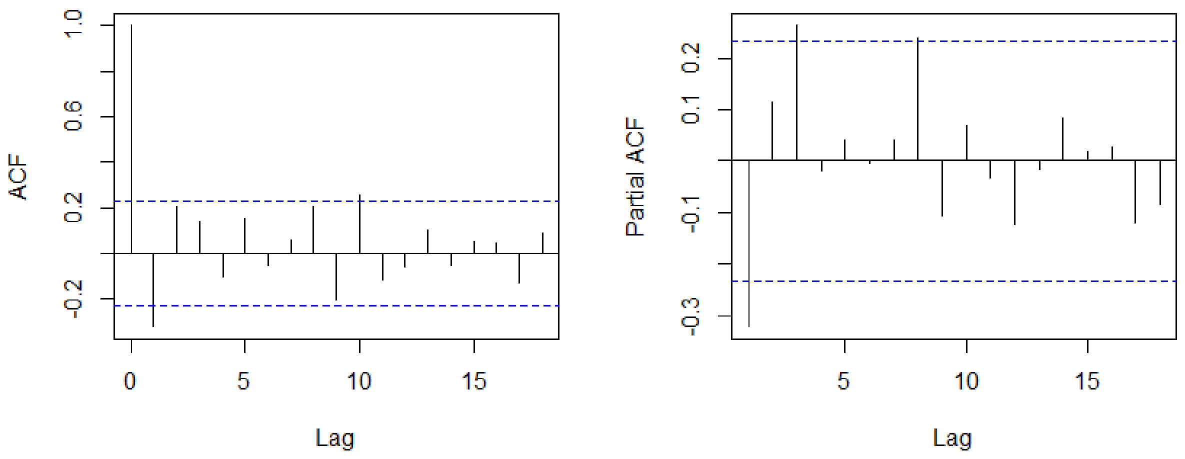

Figure 7: ACF and PACF functions of fitted ARIMA(1,1,3).

ARIMA(1,1,1)

In case of this model all coefficients are statistically significant and didn't appear due to chance. Moreover ACF and PACF functions look much more stable - there aren't any lags, whose spikes exceed set interval.

```
     Estimate Std. Error z value  Pr(>|z|)
ar1   0.98547    0.01715 57.4611 < 2.2e-16
ma1  -0.81280    0.08437 -9.6338 < 2.2e-16
```

Figure 8: Coefficients test for ARIMA(1,1,1).

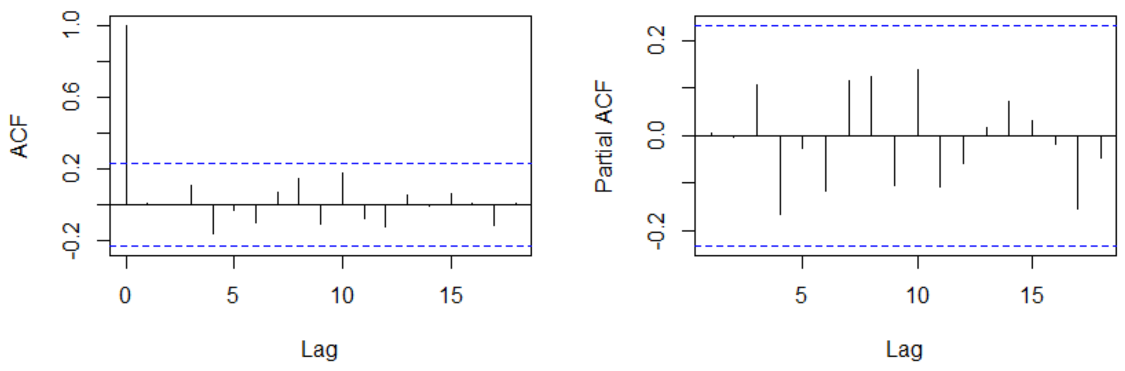

Figure 9: ACF and PACF functions of fitted ARIMA(1,1,1).

ARIMA(0,1,1)

Additional model will be tested, where AR part will be excluded. It was arguable whether the AR part should be included in the model, because both ARMA(1,1) as well as MA(1) models can result in harmonic oscillations in PACF functions, thus is is reasonable to test as well this model. Even though estimated ma1 parameter is statistically significant as seen on Figure 10, looking at ACF and PACF functions at Figure 11 it can be seen that autocorrelations at specific lags are performing visibly worse than in case of ARIMA(1,1,1) (Figure 8).

```
    Estimate Std. Error z value  Pr(>|z|)
ma1 0.498199   0.094741  5.2585 1.452e-07
```

Figure 10: Coefficients test for ARIMA(0,1,1).

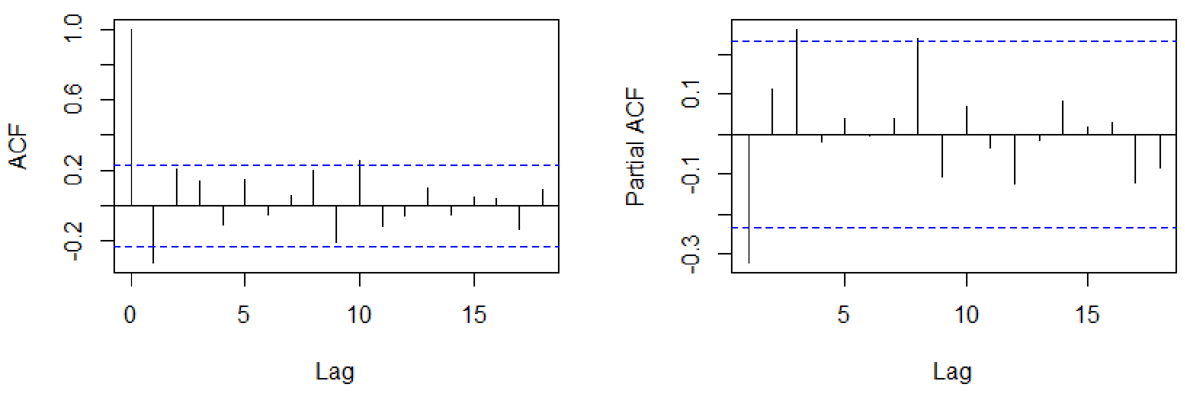

Figure 11: ACF and PACF functions of fitted ARIMA(0,1,1).

Other models

Since it was not crystal clear from ACF and PACF functions whether specific terms should definitely be included and tested models showed similar AIC value, few models are going to be compared. Tested MA parts will be in set 0,1,2,3,4 and AR in set 0,1,2. Results are presented in Table 2. As expected the best obtained model is the one with lowest value of AIC, thus ARIMA(1,1,1).

	MA part				
AR part	**q = 0**	**q = 1**	**q = 2**	**q = 3**	**q = 4**
p = 0	*151.8528*	*-173.7117*	*-178.9136*	*-191.3639*	*-192.4891*
p = 1	*-192.1526*	*-212.4106*	*-210.4300*	*-208.4308*	*-208.2413*
p = 2	*-201.0462*	*-210.4300*	*-208.9863*	*-207.1804*	*-206.5722*

Table 2: AIC evaluation for each pair of AR (p) and MA (q) part. Values in table apart from first row and column are values of AIC.

Final model - ARIMA(1,1,1)

Best obtained model is ARIMA(1,1,1). Both of his coefficients were already tested and proved to be statistically significant - weren't obtained due to a chance. Residuals of this models were investigated with use of 'checkresiduals()' function from 'forecast' library. Residuals, their histogram and acf graphs were presented (Figure 12). Starting with autocorrelation function, values of correlation for all lags lower or equal to 20 were inside threshold. One can conclude from residuals distribution that they seem to represent white noise, since their histogram looks similar to normally distributed. Of course it is not ideal Gaussian and some outliers can be notices, but this might have been caused by small amount of data points that were considered in analysis. General mathematical formulation of model looks as follows:

$$\phi(\mathrm{B})\nabla^d Y_t = \theta(\mathrm{B})\epsilon_k \tag{2}$$

The differentiating term in our model is equal d=1 and both autoregressive and moving average parts are equal to 1, thus polynomials $\phi(\mathrm{B}), \theta(\mathrm{B})$ will both be first order. The output from the function *arima* in R is 0.99 for the AR-part and -0.81 for the MA-part. However, *arima* estimates all the coefficients as being on the right hand side of the equation. Therefore, we need to change the sign for the AR-part in order to follow the convention defined in Equation 2. After considering these obtained coefficients, the factors in Equation 2 will be:

$$\nabla^d = \nabla^1 = (1-B)^1 = (1-B) \tag{3a}$$

$$\phi(\mathrm{B}) = 1 + \phi_1\mathrm{B} = 1 - 0.99\mathrm{B} \tag{3b}$$

$$\theta(\mathrm{B}) = 1 + \theta_1\mathrm{B} = 1 - 0.81\mathrm{B} \tag{3c}$$

Thus formulation of our model will look as follows:

$$(1 - 0.99\mathbf{B})\,(1 - \mathrm{B})\,\hat{Y}_t = (1 - 0.81\mathrm{B})\,\epsilon_t \tag{4}$$

With transfer function:

$$H(z) = \frac{1 - 0.81z^{-1}}{(1 - 0.99z^{-1})(1 - z^{-1})} \tag{5}$$

Moreover, considering the initial logarithm transformation of data it is to be remembered, that $\hat{Y}_t$ in above is in fact $\log(Y_t) - \mu$ of original data.

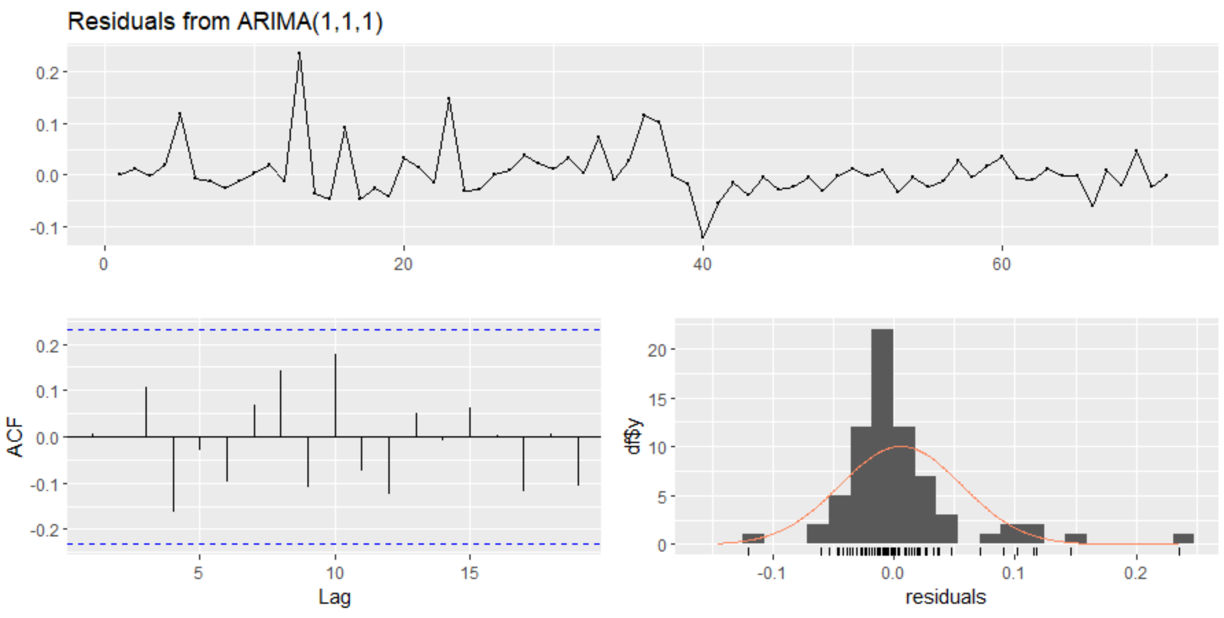

Figure 12: Residuals analysis of ARIMA(1,1,1).

Procedure above was carried for whole data set. We forgot to split data for training and test set before all procedures, which might have been the best practice to validate that the model actually predicts the correct values. We did this afterwards - splitting data to training and test, by leaving last 10 and 5 values for model validation (test set) and following the same procedure. The obtained optimal model were still ARIMA(1,1,1) and coefficients were basically the same. Hence it can be assumed, that proper investigation path would look very similar to what is presented. To prove that the model is correct, the new model was trained by leaving last 5 years of data set as a test set. After this the model was evaluated on this data and results are presented below in Table 3 and Figure 13. As can be seen predictions are virtually covered with original data, thus we can expect accurate forecast while using this model to estimate danish legislation for next 10 years.

Year	Original value	Prediction	Lower CI	Upper CI
2016	26999144	26730180	24151037	29584755
2017	27356201	27566151	23564808	32246930
2018	29523506	28414456	23066283	35002661
2019	29931409	29274860	22576647	37960349
2020	30837453	30147122	22073047	41174604

Table 3: Obtained values compared to original ones, evaluated on test data.

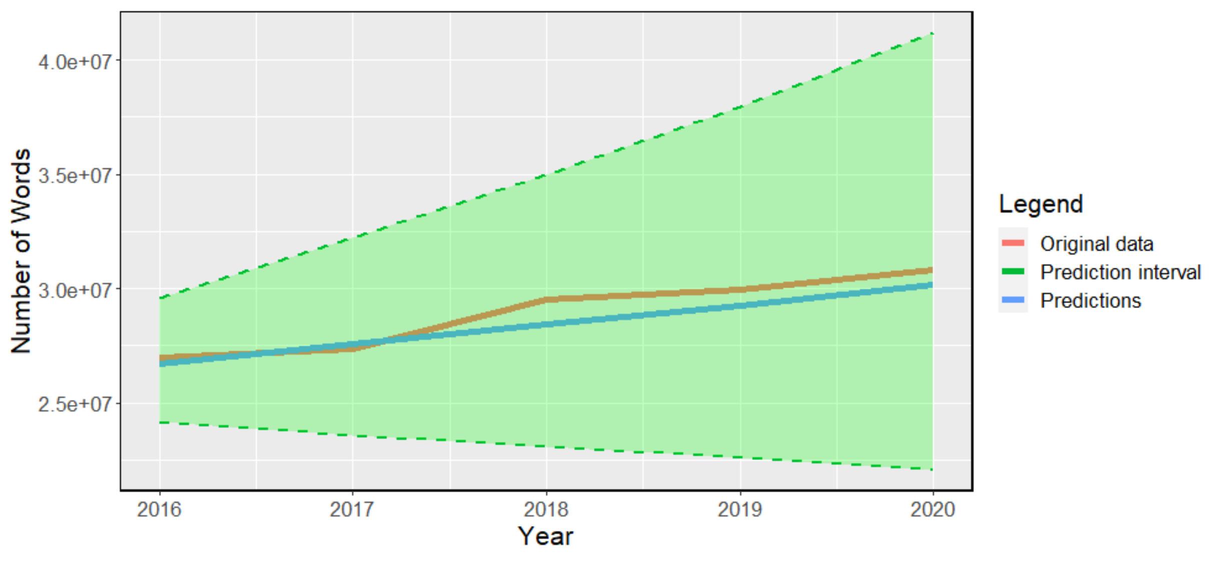

Figure 13: Original data, predictions and confidence interval for test data.

Question 2.3

The function *predict* in *R* has been used to do predictions for the best ARMA model found in the previous question; the $(1,1,1) \times (0,0,0)_0$ ARIMA model. The function also estimates the standard errors and hence the prediction intervals for the k-step prediction can be found using the following:

$$\hat{Y}_{t+k|t} \pm u_{\alpha/2} \cdot \sigma_\epsilon \tag{6}$$

where $u_{\alpha/2}$ is the $\alpha/2$-quantile in the standard normal distribution; which can be used due to the large size of our data sample. We use $\alpha = 5\%$. The predictions together with the prediction intervals are seen in Figure 14, after data has been transformed back to the original domain. Here it is clear that the predictions follow an increasing linear trend as expected and the prediction intervals are growing quite rapidly. Exact results were presented in Table 8. Prediction intervals growth goes in pair with variance growth. It is continuation of divergence of original data, in other words, the time series can be considered a non-stationary process. As time follows, the number of words in danish legislation will keep increasing rapidly, due to obtained model.

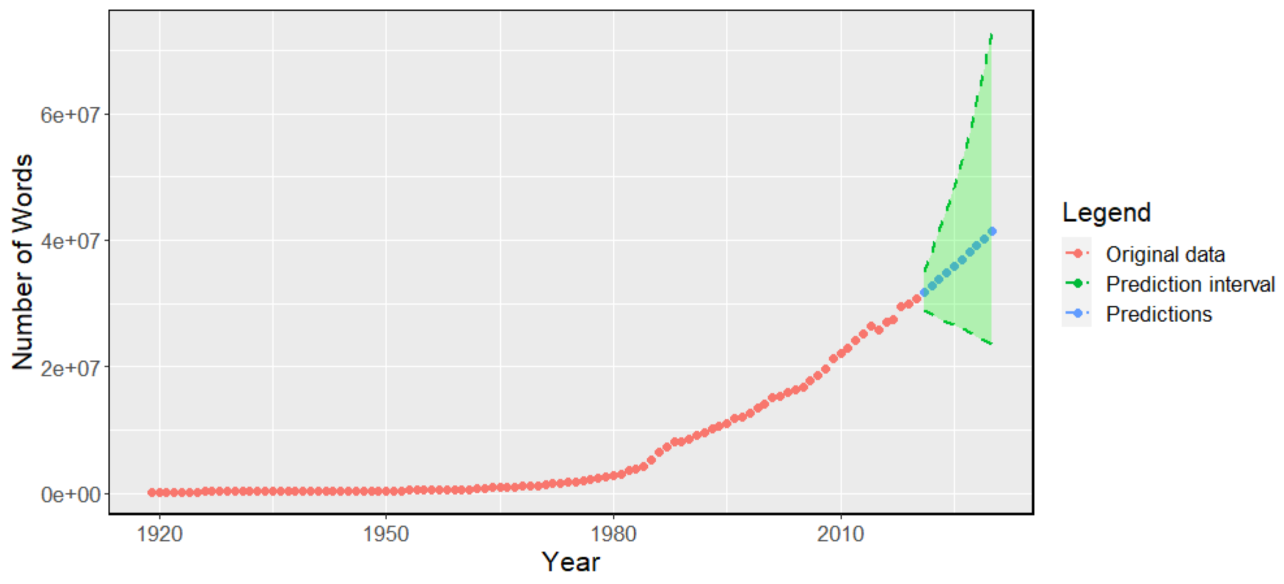

Figure 14: Predictions with ARIMA (1,1,1) model.

Year	$\log(\mathbf{Y_t}) - \mu$	**log(lowerPI)**$-\mu$	**log(upperPI)**$-\mu$	$\mathrm{Y_t}$	**lowerPI**	**upperPI**
2021	4.364629	4.265969	4.463289	31821622	28831995	35121247
2022	4.395589	4.243538	4.547640	32822218	28192459	38212275
2023	4.426099	4.224426	4.627772	33839055	27658768	41400313
2024	4.456166	4.205403	4.706928	34871935	27137583	44810618
2025	4.485796	4.185433	4.786158	35920650	26601029	48505384
2026	4.514995	4.164104	4.865886	36984981	26039662	52530974
2027	4.543771	4.141245	4.946296	38064699	25451166	56929468
2028	4.572128	4.116795	5.027461	39159566	24836427	61742843
2029	4.600073	4.090748	5.109399	40269333	24197875	67014943
2030	4.627613	4.063128	5.192098	41393744	23538682	72792609

Table 4: Predictions values from ARIMA(1,1,1) model.

Question 2.4

Obtained best arima model has following p, d, q values:

- $d = 1$ - this parameter informs that original data is differentiated (differences are calculated) once and than ARMA model is fitted to obtained new data
- $p = 1$ - (AR part) this parameter informs that only 1 step backward Y_{t-1} will be taken under consideration in estimating value of Y_t
- $q = 1$ - (MA part) this parameter informs that only 1 step backward noise parameter will impact current noise measurement

The obtained estimated parameters in the model is 0.99 for the AR-part and -0.81 for the MA-part, when both parts are on the right hand side of Equation 2. The coefficient for the AR-part tells us that the previous value of Y_t has a positive contribution to the current one. In other words; the number of words last year affects the current number of words a lot. This makes good sense, since the danish law will not be rewritten every year. Therefore the amount of words of the previous year affects the current year a lot. Two consecutive years have a quite similar amount of words and the number of words it not completely stochastic noise. The coefficient is positive since more and more words will be added to the already existing laws the year before. Once in a while a law is taken out but often laws are permanent and new laws are added (or existing laws will be elaborated). Since the best model is found to be ARIMA(1,1,1) it means that the previous number of words together with the previous noise estimate is sufficient in order to describe the new value. This makes sense intuitively when looking at the data in its raw form in Figure 1. The values are almost every year a little bit higher than the previous year, so with our fitted model we would be able to forecast the next year quite accurately only by knowing the value of the previous year.

The model also depends on the stochastic term of the previous year (the MA-part) along with noise term of the current year. There have to be a noise term no matter what, because the number of laws every year is not constant. But the dependency of the noise term for last year as well as the coefficient being negative might also be explainable from a real world perspective. The negative sign means that the previous noise term will contribute to a smaller value for the new number of words (if the previous value is positive obviously). If the stochastic term for the previous year is really high - in real life it can happen just by random or for instance because a new comprehensive financial law has been agreed on - then it is more likely that the legislation rate the next year will be a bit lower. So if a big law has been finished off the year before then this gives a legislation spike, which should be compensated for the next year. On the other hand if the noise term is negative, then this will give a positive contribution to the legislation rate the following year. The negative coefficient acts a bit like a buffer which counteracts the legislation rate for two consecutive year so they don't both show extremities, which is rare in real life. This could be the reason for the model behaviour.

After whole procedure of best model search in part 2.2, few facts appeared to be surprising. Firstly, before conducting any type of analysis we expected some seasonal trend to appear in data. In Denmark election takes place every 4 years, thus this was expected seasonality period. After conducting basic data preprocessing and plotting it appeared that this doesn't impact danish legislation. It was confirmed later by ACF and PACF functions analysis. It wasn't certain from ACF and PACF, till the very end of model fitting, whether the best model is going to have Moving Average parameter equal to 3 or to 1. Eventually, the simpler model proved to be best choice for considered data. Model could be improved by including more attributes into prediction and fit some type of multivariate Arima. Furthermore, there are only 102 data points available, thus by increasing this number we could obtain new model, more precisely fitted to analysed case.

Question 2.5

In this part we aimed to create a linear model that takes into consideration both the model calculated to the previous questions but also adds a linear regression model having one coefficient for each one of the parties. In order to achieve that we did not need to use any new libraries for forecasting or fitting, we just use the *arima* method of R as in the previous sections in this report. Also we needed to make an encoding for the parties by applying one-hot encoding. Here we used the function *one_hot* from the library *mltools*.

Mathematical formulation

We have decided to use a regression model with ARMA noise. This can be fitted by using $xreg$ in R. Our obtained model, applying the same transformation as the one presented in Equation 1, can be represented as:

$$\hat{Y}_t = \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_3 + \beta_4 x_4 + \eta_t, \tag{7}$$

where: x_1, x_2, x_3, x_4 are binary values originated from the one-hot encoding, whose values describe which political party was in charge during the investigated year (x_1 - Conservative, x_2 - Social Liberals, x_3 - Social Democrats, x_4 - Venstre, all 0 is Independent). The reason for not encoding 'The Independent' as x_5 is in order to avoid multicolinearity, which might break the optimizer. In this way our model assumes all values to be related to a baseline function, which simulates the amount of words per year if the Independent party was the ruling one in every single one, and then that amount is changed by the β_i coefficients from the regression model that are described below. Those β_i are generated from the differences in word increase between their corresponding party and the baseline party; the Independents. So it is a relative legislation rate. As mentioned the same transformation are used but this time we use all the data available in order to get the best possible regression. η_t is the noise term, described by ARIMA(1,1,1) model, which is as in equation 4, but instead of $\hat{Y}_t$ it would be η_t. Values of betas are as follows:

- $\beta_1 = -0.00372429$, standard deviation 0.0391 Conservatives
- $\beta_2 = -0.06933899$, standard deviation 0.0351 Social Liberals ,
- $\beta_3 = 0.01371953$, standard deviation 0.0266 Social Democrats
- $\beta_4 = 0.00060775$, standard deviation 0.0286 Venstre

Since we have a relative model, positive coefficients mean that the legislation rate is higher than for 'The Independent', and parties with negative coefficients mean that the legislation rate is lower than for 'The Independent' for that particular party. If we do not take into consideration the standard deviation it seems that having a Social Liberal or a Conservative president results in slower increment of the legislative body than while having a Social Democrat. However, the standard deviations of Social Democrats, Venstre and Conservatives are greater than the actual coefficient value. That means that it is not the best model to be used.

We chose the fit the model with $xreg$ and ARMA(1,1,1) noise, since this was found to be the best model in Question 2.2. The new values for the ARMA-coefficients are 0.98 for the AR-part and -0.78 for the MA-part. In the same way as in Equation 5 the transfer function can be written as:

$$H(z) = \frac{1 - 0.78z^{-1}}{(1 - 0.98z^{-1})(1 - z^{-1})} \tag{8}$$

and the whole model will be

$$\hat{Y}_t = \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_3 + \beta_4 x_4 + \frac{1-0.78z^{-1}}{(1-0.98z^{-1})(1-z^{-1})}\epsilon_t, \tag{9}$$

When 'The Independent' rules, all the x-variables are zero and the legislation rate is just given as the transfer function. When for instance the Social Democrats rules, the legislation rate is the transfer function plus the β_3 and then everything transformed back to get the exact number of words.

The greater the coefficient for each party, the bigger the prediction for the legislation body increase would be. Based on that Social Democrats produce the legislation body at the highest rate $\beta_3 = 0.01371953$.

In order to estimate the party that has the fastest change rate the increase trough time should be related not to the total amount of words added, but rather on the the change from one time step to the next. An other thing to take into consideration is that not each year has the same party of the previous one, if we took into account those instances we would be measuring the change in trend between those two parties rather than the quickness of either of them, so only couples of consecutive years with the same party should be evaluated (luckily parties retain power for at least four consecutive years). The resulting generalized equations are:

$$R_t(x_k) = \frac{\hat{Y}_t - \hat{Y_{t-1}}}{\hat{Y_{t-1}}}, \text{if } x_t = x_{t-1} = x_k \tag{10}$$

$$Q_k = \frac{\sum_{i=-\infty}^{\infty} R_{t-i}(x_k)}{N_k} \tag{11}$$

Where k=1,2,3,4 related to the five parties and N values are from:

$$N_k = \frac{\sum_{i=-\infty}^{\infty} R_{t-i}}{2}, \text{if } x_t = x_{t-1} = x_k \tag{12}$$

The only rate left is the one related to 'The Independent', which could be estimated by using and 11 and:

$$R_t(x_k) = \frac{\hat{Y}_t - \hat{Y_{t-1}}}{\hat{Y_{t-1}}}, \text{if } x_t = x_{t-1} = 0 \tag{13}$$

$$N_k = \frac{\sum_{i=-\infty}^{\infty} R_{t-i}}{2}, \text{if } x_t = x_{t-1} = 0 \tag{14}$$

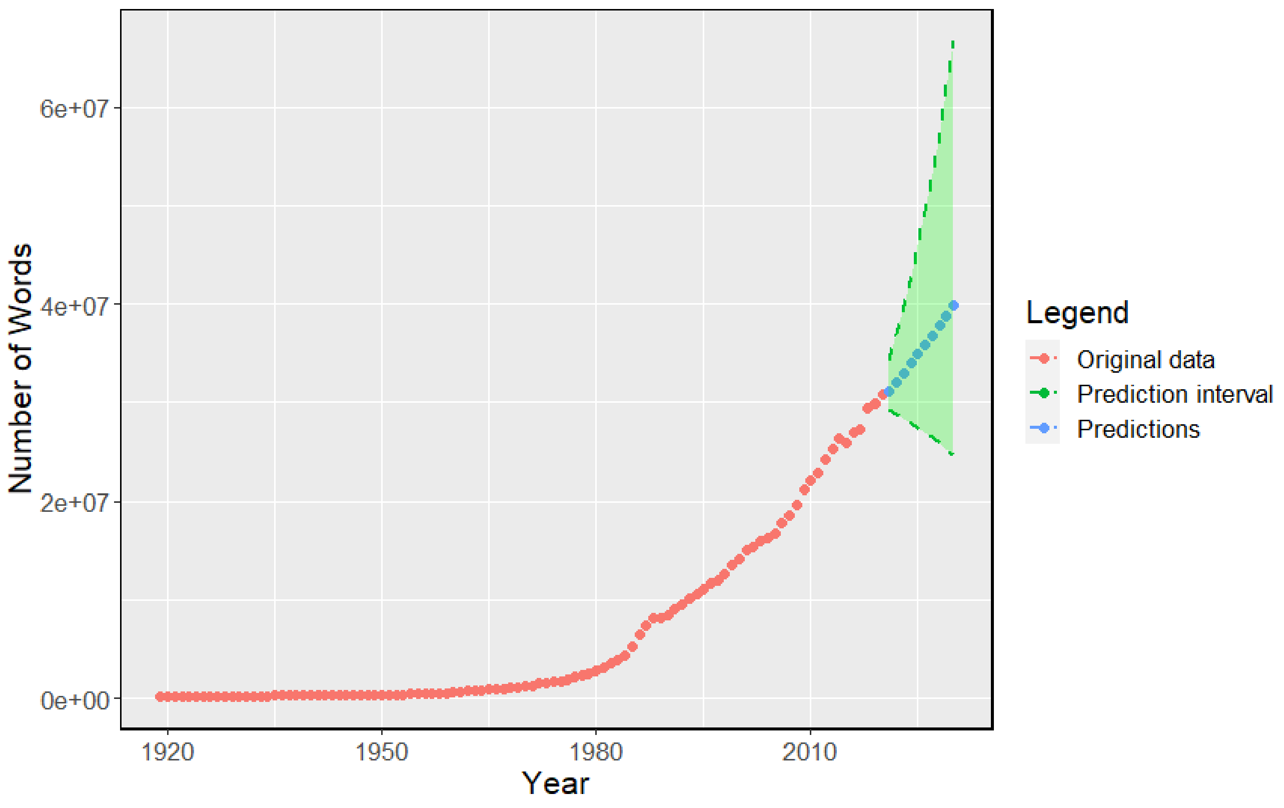

Figure 15: Social Democrats legislation body prediction

Also from figure 16 it seems that our assumption that the Social Democrats will produce bigger legislation bodies it is confirmed.

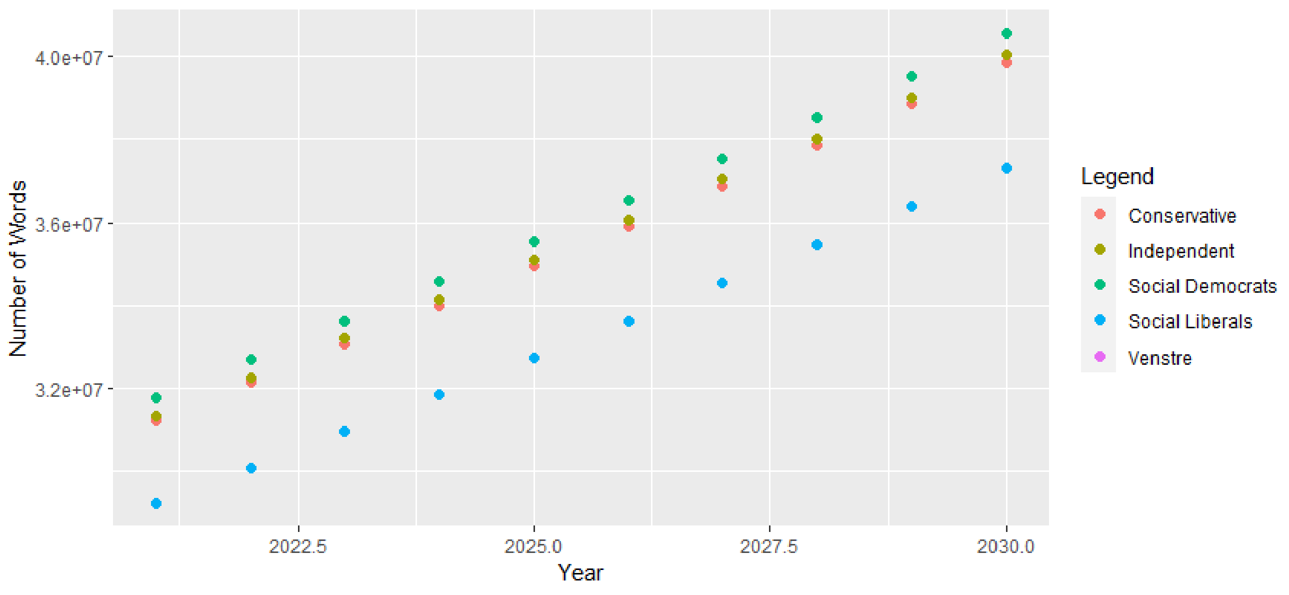

Figure 16: All parties predictions for period 2021-2030.

Also the actual predictions of each party were the following.

Year	Conservative	lower	upper
2021	4.364629	4.265969	4.463289
2022	4.395589	4.243538	4.547640
2023	4.426099	4.224426	4.627772
2024	4.456166	4.205403	4.706928
2025	4.485796	4.185433	4.786158
2026	4.514995	4.164104	4.865886
2027	4.543771	4.141245	4.946296
2028	4.572128	4.116795	5.027461
2029	4.600073	4.090748	5.109399
2030	4.627613	4.063128	5.192098

Table 5: Predictions values from ARIMA(1,1,1) model.

Year	$\log(\mathbf{Y_t}) - \mu$	$\mathbf{\log(lowerPI)}-\mu$	$\mathbf{\log(upperPI)}-\mu$	$\mathrm{Y_t}$	**lowerPI**	**upperPI**
2021	4.364629	4.265969	4.463289	31821622	28831995	35121247
2022	4.395589	4.243538	4.547640	32822218	28192459	38212275
2023	4.426099	4.224426	4.627772	33839055	27658768	41400313
2024	4.456166	4.205403	4.706928	34871935	27137583	44810618
2025	4.485796	4.185433	4.786158	35920650	26601029	48505384
2026	4.514995	4.164104	4.865886	36984981	26039662	52530974
2027	4.543771	4.141245	4.946296	38064699	25451166	56929468
2028	4.572128	4.116795	5.027461	39159566	24836427	61742843
2029	4.600073	4.090748	5.109399	40269333	24197875	67014943
2030	4.627613	4.063128	5.192098	41393744	23538682	72792609

Table 6: Predictions values from ARIMA(1,1,1) model.

Year	$\log(\mathbf{Y_t}) - \mu$	**log(lowerPI)**$-\mu$	**log(upperPI)**$-\mu$	$\mathrm{Y_t}$	**lowerPI**	**upperPI**
2021	4.364629	4.265969	4.463289	31821622	28831995	35121247
2022	4.395589	4.243538	4.547640	32822218	28192459	38212275
2023	4.426099	4.224426	4.627772	33839055	27658768	41400313
2024	4.456166	4.205403	4.706928	34871935	27137583	44810618
2025	4.485796	4.185433	4.786158	35920650	26601029	48505384
2026	4.514995	4.164104	4.865886	36984981	26039662	52530974
2027	4.543771	4.141245	4.946296	38064699	25451166	56929468
2028	4.572128	4.116795	5.027461	39159566	24836427	61742843
2029	4.600073	4.090748	5.109399	40269333	24197875	67014943
2030	4.627613	4.063128	5.192098	41393744	23538682	72792609

Table 7: Predictions values from ARIMA(1,1,1) model.

Year	$\log(\mathbf{Y_t}) - \mu$	**log(lowerPI)**$-\mu$	**log(upperPI)**$-\mu$	$\mathrm{Y_t}$	**lowerPI**	**upperPI**
2021	4.364629	4.265969	4.463289	31821622	28831995	35121247
2022	4.395589	4.243538	4.547640	32822218	28192459	38212275
2023	4.426099	4.224426	4.627772	33839055	27658768	41400313
2024	4.456166	4.205403	4.706928	34871935	27137583	44810618
2025	4.485796	4.185433	4.786158	35920650	26601029	48505384
2026	4.514995	4.164104	4.865886	36984981	26039662	52530974
2027	4.543771	4.141245	4.946296	38064699	25451166	56929468
2028	4.572128	4.116795	5.027461	39159566	24836427	61742843
2029	4.600073	4.090748	5.109399	40269333	24197875	67014943
2030	4.627613	4.063128	5.192098	41393744	23538682	72792609

Table 8: Predictions values from ARIMA(1,1,1) model.

A Appendix A

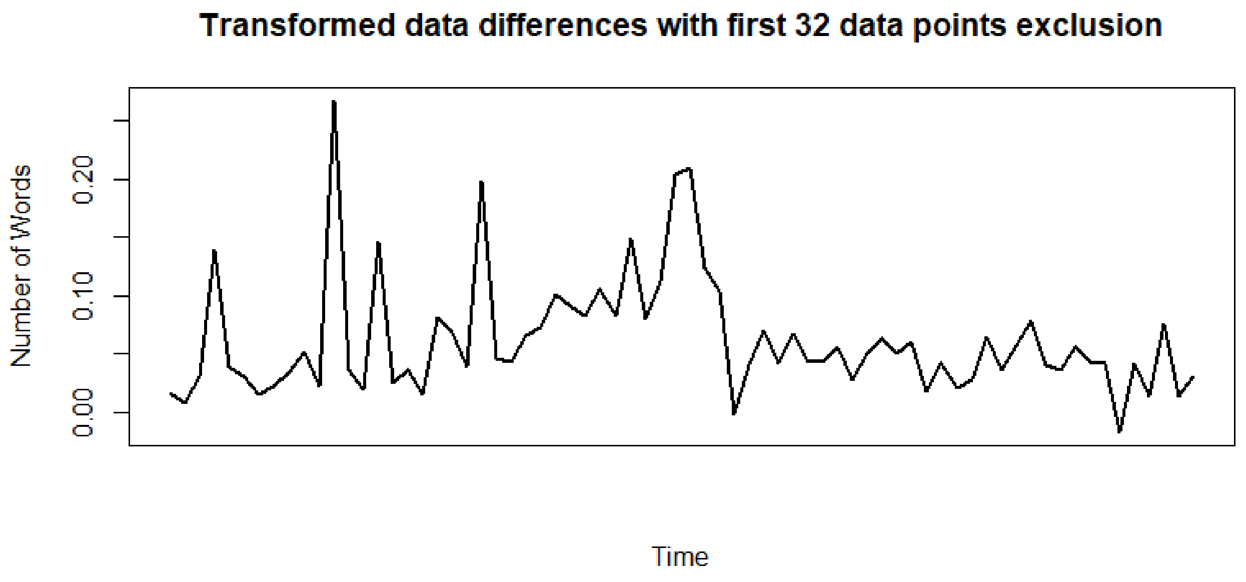

Figure 17: Differences of transformed data, after removing first 32 measurements.

B R code used in assignment

For part 1-4:

```
# ----------------- Imports -----------------

library(ggplot2)
library(rstudioapi)
library(forecast)
library(flexmix)
library(lmtest)

pathA=dirname(rstudioapi::getSourceEditorContext()$path)
data <- read.csv(paste(pathA, '/DataAssignment3.csv', sep=''), header=TRUE)
#data <- data[1:91,]
Y <- data[,2]

# ----------------- Question 2.1 - Data overview + stationarity -----------------

# plotting the data categorized by Government
ggplot(data, aes(x=Date, y=NumberWords, color = Government)) +
  geom_point(size=2) +
  labs(y='Number of Words') +
  theme(text = element_text(size = 16)) +
  theme(panel.border = element_rect(color = "black",
                                    fill = NA,
                                    size = 1))

# Calculating mean value and variation for few steps to show that they aren't
mean_var <- matrix(0,3,5)
mean_var <- data.frame(mean_var)
mean_var[1,] <- c(20, 40, 60, 80, 100)
for (i in 1:5)
{
  mean_var[2,i] <- mean(Y[1:mean_var[1,i]])
  mean_var[3,i] <- var(Y[1:mean_var[1,i]])
}
mean_var <- data.frame(mean_var)
```

```
rownames(mean_var) <- c('Observations', 'Mean', 'Variance')

# ——————— Question 2.1 - Data transformations ————————

# Log transformation
mu <- log(Y)[1]
Y_transformed <- log(Y) - mu

# Plotting log data
ggplot(data, aes(x=Date, y=Y_transformed, color = Government)) +
  geom_point(size=2) +
  labs(y='Number of Words') +
  theme(text = element_text(size = 16)) +
  theme(panel.border = element_rect(color = "black",
                                    fill = NA,
                                    size = 1))


# ACF and PACF original data
par(mfrow=c(2,2))
plot(acf(Y, plot=FALSE), main='ACF of original time series')
plot(pacf(Y, plot=FALSE), main= "PACF of original time series")
# ACF and PACF log transformed data
plot(acf(Y_transformed, plot=FALSE), main='ACF of transformed time series')
plot(pacf(Y_transformed, plot=FALSE), main= "PACF of transformed time series")


# Calculating differences
Y_diff <- diff(Y, lag = 1, differences = 1)
Y_diff_transformed <- diff(Y_transformed, lag = 1, differences = 1)


# Plots to compare the effect of differencing with and without transformation
par(mfrow=c(1,2))
plot(data$Date[2:n], Y_diff, main = "Original data differences", xlab = "Time"
,
     type = "l", lwd=2)
plot(data$Date[2:n], Y_diff_transformed, main = "Transformed data differences"
,
     type = "l", lwd=2)


# Removing some data
no_points <- 32
n <- length(Y)
Y_new <- data[no_points:n,2]
mu <- log(Y_new)[1]
Y_new_transformed <- log(Y_new) - mu


# Initial ACF and PACF - NOT IN REPORT
par(mfrow=c(2,2))
acf(Y_new_transformed)
pacf(Y_new_transformed)
acf(Y_new)
pacf(Y_new)


# Calculating differences of transformed data
Y_new_diff_transformed = diff(Y_new_transformed, lag = 1, differences = 1)
Y_new_diff = diff(Y_new, lag = 1, differences = 1)
```

```
# Plots to compare the effect of differencing with and without transformation
par(mfrow=c(2,2))
plot(Y_new)
plot(Y_new_transformed)
plot(Y_new_diff)
plot(Y_new_diff_transformed)

# AFC and PAFC after differencing
par(mfrow=c(2,2))
acf(Y_new_diff_transformed)
pacf(Y_new_diff_transformed)
acf(Y_new_diff)
pacf(Y_new_diff)
dev.off()
plot(data$Date[no_points:(n-1)], Y_new_diff_transformed, main = "Transformed d
,
     type = "l", xaxt = "n", lwd=2)

# AFC and PAFC after differencing
par(mfrow=c(2,2))
plot(acf(Y_diff_transformed, plot=FALSE), main='ACF after differentiation')
plot(pacf(Y_diff_transformed, plot=FALSE), main= "PACF after differentiation")
plot(acf(Y_new_diff_transformed, plot=FALSE), main='ACF after differentiation
plot(pacf(Y_new_diff_transformed, plot=FALSE), main= "PACF after differentiatio

# ———————— Question 2.2 ————————

# ———————— Starting with model ARIMA (1, 1, 1) ————————

# Fitting model
model_1 <- arima(Y_new_transformed, order=c(1, 1, 1))
model_1

# ACF, PACF and coefficients test to report
coeftest(model_1)
par(mfrow=c(1,2))
acf_model1=ggAcf(model_1$residuals,plot=FALSE)
plot(acf_model1,main="")
pacf_model1=ggPacf(model_1$residuals,plot=FALSE)
plot(pacf_model1,main="")

# ———————— Also checking ARIMA (1, 1, 3) ————————

model_2 <- arima(Y_new_transformed, order=c(1, 1, 3))
model_2

# ACF, PACF and coefficients test to report
coeftest(model_2)
par(mfrow=c(1,2))
acf_model2=ggAcf(model_2$residuals,plot=FALSE)
plot(acf_model1,main="")
pacf_model2=ggPacf(model_2$residuals,plot=FALSE)
plot(pacf_model1,main="")

# ———————— Also checking ARIMA (0, 1, 1) ————————

model_3 <- arima(Y_new_transformed, order=c(0, 1, 1))
model_3
```

```
# ACF, PACF and coefficients test to report
coeftest(model_3)
par(mfrow=c(1,2))
acf_model3=ggAcf(model_3$residuals,plot=FALSE)
plot(acf_model3,main="")
pacf_model3=ggPacf(model_3$residuals,plot=FALSE)
plot(pacf_model3,main="")

# ---------------- Investigating all models ----------------
# p - AR part = 0, 1, 2
# q - MA part = 0, 1, 2, 3, 4

ps <- c(0, 1, 2)
qs <- c(0, 1, 2, 3, 4)
aics <- matrix(nrow=length(ps), ncol=length(qs))

for (p in ps){
  for (q in qs){
    model <- arima(Y_new_transformed, order=c(p, 1, q))
    aics[which(ps==p), which(qs==q)]=model$aic
  }
}

# ---------------- ARIMA (1,1,1) ----------------
# Best obtained model is ARIMA (1,1,1)

# Fitting best model
model <- arima(Y_new_transformed, order=c(1, 1, 1))
model

# ACF, PACF and coefficients test to report
coeftest(model)
par(mfrow=c(1,2))
acf_model2=ggAcf(model_2$residuals,plot=FALSE)
plot(acf_model1,main="")
pacf_model2=ggPacf(model_2$residuals,plot=FALSE)
plot(pacf_model1,main="")
checkresiduals(model)

# ---------------- Question 2.3 ----------------

# log predictions and intervals
predictions <- predict(model, n.ahead=10)
conf = cbind(predictions$pred + qnorm(0.025)*predictions$se,predictions$pred -

Y_pred = exp(predictions$pred+mu)
conf_pred = exp(conf+mu)

Dates <- data[1]
Dates <- sapply(Dates,as.numeric)
new_dates <- 2021:2030
Dates <- append(Dates, new_dates)
Y_new2 <- sapply(Y,as.numeric)
Y_final <- append(Y_new2,Y_pred)
Y_final <- cbind(Dates, Y_final)
Y_finaldf <- data.frame(Y_final)
Y_finaldf['Col'] <- 'Original␣data'
Y_finaldf[103:112,'Col']='Predictions'
Y_finaldf[103:112,'lower']=conf_pred[,1]
Y_finaldf[103:112,'upper']=conf_pred[,2]
```

```
ggplot(Y_finaldf, aes(x=Dates)) +
  geom_point(aes(y=Y_final, color=Col), size=2) +
  geom_line(aes(y=lower, color='Prediction interval'), linetype = "dashed", si
  geom_line(aes(y=upper, color='Prediction interval'), linetype = "dashed", si
  geom_ribbon(data=subset(Y_finaldf, 2021 <= Dates & Dates <= 2030), aes(ymin=
  theme(text = element_text(size = 16)) +
  theme(panel.border = element_rect(color = "black",
                                    fill = NA,
                                    size = 1)) +
  labs(y='Number of Words') +
  labs(x='Year') +
  labs(color='Legend')

results <- cbind(predictions$pred, conf, Y_pred, conf_pred)
results <- data.frame(results)

# ------------- Training - test ----------------

data2 <- data[1:97,]
Y2 <- data2[,2]
n <- length(Y2)
Y_new <- data2[no_points:n,2]
mu <- log(Y_new)[1]
Y_new_transformed2 <- log(Y_new) - mu
model <- arima(Y_new_transformed2, order=c(1, 1, 1))

predictions <- predict(model, n.ahead=5)
conf = cbind(predictions$pred + qnorm(0.025)*predictions$se,predictions$pred -

Y_pred = exp(predictions$pred+mu)
conf_pred = exp(conf+mu)

#Dates <- data[1]
#Dates <- sapply(Dates,as.numeric)
#new_dates <- 2016:2020
#Dates <- append(Dates, new_dates)
#Y_new2 <- sapply(Y,as.numeric)
#Y_final <- append(Y_new2,Y_pred)
#Y_final <- cbind(Dates, Y_final)
Y_finaldf <- data[,1:2]
#Y_finaldf['Col'] <- 'Original data'
#Y_finaldf[98:102,'Col'] = 'Predictions'
Y_finaldf[98:102,'lower'] <- conf_pred[,1]
Y_finaldf[98:102,'upper'] <- conf_pred[,2]
Y_finaldf[98:102, 'preds'] <- Y_pred
Y_finaldf <- Y_finaldf[98:102,]

ggplot(Y_finaldf, aes(x=Date)) +
  geom_line(aes(y=NumberWords, color='Original data'), size=2) +
  geom_line(aes(y=preds, color='Predictions'), size=2) +
  geom_line(aes(y=lower, color='Prediction interval'), linetype = "dashed", si
  geom_line(aes(y=upper, color='Prediction interval'), linetype = "dashed", si
  geom_ribbon(data=subset(Y_finaldf, 2016 <= Date & Date <= 2020), aes(ymin=lo
  theme(text = element_text(size = 16)) +
  theme(panel.border = element_rect(color = "black",
                                    fill = NA,
                                    size = 1)) +
  labs(y='Number of Words') +
  labs(x='Year') +
  labs(color='Legend')
```

```
results <- cbind(predictions$pred, conf, Y_pred, conf_pred)
results <- data.frame(results)
```

For part 5:

```
dev.off()  # But only if there IS a plot

# Clear console
cat("\014")  # ctrl+L

## This empties the work space.
rm(list=ls())


# ———————— Importing data ————————

#remove.packages(c("flexmix", "mltools","data.table"))

#install.packages("flexmix",dependencies = TRUE)
#install.packages("mltools",dependencies = TRUE)
#install.packages("data.table",dependencies = TRUE)


library(ggplot2)
library(rstudioapi)
library(forecast)
library(flexmix)
library(lmtest)
library(mltools)
library(data.table)

pathA=dirname(rstudioapi::getSourceEditorContext()$path)
raw_data  <- read.csv(paste(pathA, '/DataAssignment3.csv', sep=''), header=TRU

# Transformation of data
Y <- raw_data[,2]
mu <- log(Y)[1]
#Y_new <- raw_data[33:length(Y),2]
#mu_new <- log(Y_new)[1]

##################
# THIS PART BELOW WAS USED ONLY TO TRY DIFFERENT COMBINATIONS OF DATA INPUT AN
# OUTPUT, TO FIX OPTIMIZER BREAKING WHILE ESTIMATING ARIMA
##################

# Different data input to try
  # Log transformation of all data                            (1) - 102 measurem
  Y_transformed <- log(Y) - mu
  # Log transformation excl. first 32 datapoints              (2) - 70 measureme
  #Y_new_transformed <- log(Y_new) - mu_new
  # Differentiated log data                                   (3) - 101 measurem
  #Y_diff_transformed <- diff(Y_transformed, 1)
  # Diffetentiated log data excl. first 32 datapoints         (4) - 69 measureme
  #Y_new_diff_transformed <- diff(Y_new_transformed, 1)
  # Differentiated original data                              (5) - 101 measurem
  #Y_diff <- diff(Y, 1)
  # Differentiated original data excl. first 32 datapoints   (6) - 69 measureme
  #Y_new_diff <- diff(Y_new, 1)

X <- data.frame(raw_data[3])
#X_sh <- data.frame(raw_data[33:102,3])
#colnames(X_sh) <- c("Government")
```

```
X$Government <- as.factor(X$Government)
#X_sh$Government <- as.factor(X_sh$Government)
# Encoding political parties
  # Whole dataset                                   (A) - 102 rows
  #X_encoded <- one_hot(as.data.table(X$Government))
  #X_encoded <- sapply(X_encoded, as.numeric)
  # Whole dataset 1 party as all zeros              (B) - 102 rows
   X_encoded_drop <- one_hot(as.data.table(X$Government))
   X_encoded_drop <- sapply(X_encoded_drop, as.numeric)
   X_encoded_drop <- subset(X_encoded_drop, select = -c(V1_Independent))
  # 70 measurements                                 (C) - 70 rows
  #X_encoded_sh <- one_hot(as.data.table(X_sh$Government))
  #X_encoded_sh <- sapply(X_encoded_sh, as.numeric)
  # 70 measurements 1 party as all zeros            (D) - 70 rows
  #X_encoded_sh_drop <- one_hot(as.data.table(X_sh$Government))
  #X_encoded_sh_drop <- sapply(X_encoded_sh_drop, as.numeric)
  #X_encoded_sh_drop <- subset(X_encoded_sh_drop, select = -c(V1_Venstre))


# ---------- Proper part ----------

#######################################
# ONE COLUMN WAS DROPPED FROM ORIGINAL ONE-HOT ENCODING MATRIX, BECAUSE THE DRO
# PARTY CAN BE INPUT TO OUR MODEL BY GIVING ALL ZERO ENTRIES, AND THAT WAY WE E
# MULTICOLINEARITY WHICH APPARENTLY WAS BRAKING OPTIMIZER
#######################################

Ys <- Y_transformed
Xs <- X_encoded_drop

# model estimation
model <- arima(Ys, order=c(1,1,1), xreg=Xs)
#model <- arima(Ys, order=c(1,1,1), xreg=Xs, method = 'CSS-ML')
#model <- arima(Ys, order=c(1,1,1), xreg=Xs, optim.method = "BFGS")

# printing coefficients and performing statistical test
model
coeftest(model)

# sample prediction for 1 time step and 2 time steps for Conservative party

tests_1 <- data.frame(t(c(1, 0, 0, 0)))
tests_2 <- data.frame(t(cbind(c(1, 0, 0, 0), c(1, 0, 0, 0))))
output_1 <- predict(model, n.ahead=1, newxreg = tests_1)
output_2 <- predict(model, n.ahead=2, newxreg = tests_2)
output_1$pred
output_2$pred   # predictions, but logarithmic
output_1$se     # to confidence interval estimation (in code for parts 1-4 and
output_2$se

# fixing dates
Dates <- raw_data[1]
Dates <- sapply(Dates,as.numeric)
new_dates <- 2021:2030
Dates <- append(Dates, new_dates)



#predictions for conservative Party
Conservative <- matrix(rep(t(cbind(1,0,0,0)),10),ncol=4,byrow=TRUE)
Conservtive_model_prediction <- predict(model, n.ahead=10, newxreg = Conservat
conf = cbind(Conservtive_model_prediction$pred + qnorm(0.025)*Conservtive_mode

Y_pred = exp(Conservtive_model_prediction$pred+mu)
conf_pred = exp(conf+mu)

Conservative_pred <- Y_pred


Y_new2 <- sapply(Y,as.numeric)

Y_final <- append(Y_new2,Y_pred)
Y_final <- cbind(Dates, Y_final)

Y_finaldf <- data.frame(Y_final)
Y_finaldf['Col'] <- 'Original data'
Y_finaldf[103:112,'Col']='Predictions'
Y_finaldf[103:112,'lower']=conf_pred[,1]
Y_finaldf[103:112,'upper']=conf_pred[,2]

ggplot(Y_finaldf, aes(x=Dates)) +
  geom_point(aes(y=Y_final, color=Col), size=2) +
  geom_line(aes(y=lower, color='Prediction interval'), linetype = "dashed", siz
  geom_line(aes(y=upper, color='Prediction interval'), linetype = "dashed", siz
  geom_ribbon(data=subset(Y_finaldf, 2021 <= Dates & Dates <= 2030), aes(ymin=
  theme(text = element_text(size = 16)) +
  theme(panel.border = element_rect(color = "black",
                                    fill = NA,
                                    size = 1)) +
  labs(y='Number of Words') +
  labs(x='Year') +
  labs(color='Legend')


# predictions for Social liberals
Social_Liberals <- matrix(rep(t(cbind(0,1,0,0)),10),ncol=4,byrow=TRUE)
Social_Liberals_prediction <- predict(model, n.ahead=10, newxreg = Social_Libe

conf = cbind(Social_Liberals_prediction$pred + qnorm(0.025)*Social_Liberals_pre

Y_pred = exp(Social_Liberals_prediction$pred+mu)
conf_pred = exp(conf+mu)

Social_Liberals_pred <- Y_pred

Y_new2 <- sapply(Y,as.numeric)

Y_final <- append(Y_new2,Y_pred)
Y_final <- cbind(Dates, Y_final)

Y_finaldf <- data.frame(Y_final)
Y_finaldf['Col'] <- 'Original data'
Y_finaldf[103:112,'Col']='Predictions'
Y_finaldf[103:112,'lower']=conf_pred[,1]
Y_finaldf[103:112,'upper']=conf_pred[,2]

ggplot(Y_finaldf, aes(x=Dates)) +
  geom_point(aes(y=Y_final, color=Col), size=2) +
  geom_line(aes(y=lower, color='Prediction interval'), linetype = "dashed", siz
  geom_line(aes(y=upper, color='Prediction interval'), linetype = "dashed", siz
  geom_ribbon(data=subset(Y_finaldf, 2021 <= Dates & Dates <= 2030), aes(ymin=
  theme(text = element_text(size = 16)) +
  theme(panel.border = element_rect(color = "black",
                                    fill = NA,
                                    size = 1)) +
  labs(y='Number of Words') +
  labs(x='Year') +
  labs(color='Legend')


# Social Democrats predictions
Social_Democrats <- matrix(rep(t(cbind(0,0,1,0)),10),ncol=4,byrow=TRUE)
Social_Democrats_prediction <- predict(model, n.ahead=10, newxreg = Social_Dem

conf = cbind(Social_Democrats_prediction$pred + qnorm(0.025)*Social_Democrats_

Y_pred = exp(Social_Democrats_prediction$pred+mu)
conf_pred = exp(conf+mu)

Social_Democrats_pred <- Y_pred

Y_new2 <- sapply(Y,as.numeric)

Y_final <- append(Y_new2,Y_pred)
Y_final <- cbind(Dates, Y_final)

Y_finaldf <- data.frame(Y_final)
Y_finaldf['Col'] <- 'Original data'
Y_finaldf[103:112,'Col']='Predictions'
Y_finaldf[103:112,'lower']=conf_pred[,1]
Y_finaldf[103:112,'upper']=conf_pred[,2]

ggplot(Y_finaldf, aes(x=Dates)) +
  geom_point(aes(y=Y_final, color=Col), size=2) +
  geom_line(aes(y=lower, color='Prediction interval'), linetype = "dashed", si
  geom_line(aes(y=upper, color='Prediction interval'), linetype = "dashed", si
  geom_ribbon(data=subset(Y_finaldf, 2021 <= Dates & Dates <= 2030), aes(ymin=
  theme(text = element_text(size = 16)) +
  theme(panel.border = element_rect(color = "black",
                                    fill = NA,
                                    size = 1)) +
  labs(y='Number of Words') +
  labs(x='Year') +
  labs(color='Legend')

# Venstre predictions
Venstre = matrix(rep(t(cbind(0,0,0,1)),10),ncol=4,byrow=TRUE)
Venstre_prediction <- predict(model, n.ahead=10, newxreg = Venstre)

conf = cbind(Venstre_prediction$pred + qnorm(0.025)*Venstre_prediction$se,Vens

Y_pred = exp(Venstre_prediction$pred+mu)
conf_pred = exp(conf+mu)

Venstre_pred <- Y_pred

Y_new2 <- sapply(Y,as.numeric)

Y_final <- append(Y_new2,Y_pred)
Y_final <- cbind(Dates, Y_final)

Y_finaldf <- data.frame(Y_final)
```

```
Y_finaldf['Col'] <- 'Original data'
Y_finaldf[103:112,'Col']='Predictions'
Y_finaldf[103:112,'lower']=conf_pred[,1]
Y_finaldf[103:112,'upper']=conf_pred[,2]

Y_finaldf

ggplot(Y_finaldf, aes(x=Dates)) +
  geom_point(aes(y=Y_final, color=Col), size=2) +
  geom_line(aes(y=lower, color='Prediction interval'), linetype = "dashed", si
  geom_line(aes(y=upper, color='Prediction interval'), linetype = "dashed", si
  geom_ribbon(data=subset(Y_finaldf, 2021 <= Dates & Dates <= 2030), aes(ymin=
  theme(text = element_text(size = 16)) +
  theme(panel.border = element_rect(color = "black",
                                    fill = NA,
                                    size = 1)) +
  labs(y='Number of Words') +
  labs(x='Year') +
  labs(color='Legend')

# Independent predictions
Independent = matrix(rep(t(cbind(0,0,0,0)),10),ncol=4,byrow=TRUE)
Independent_prediction <- predict(model, n.ahead=10, newxreg = Venstre)

conf = cbind(Independent_prediction$pred + qnorm(0.025)*Independent_prediction$

Y_pred = exp(Independent_prediction$pred+mu)
conf_pred = exp(conf+mu)

Independent_pred <- Y_pred

Y_new2 <- sapply(Y,as.numeric)
typeof(Y_new2)
Y_new2 <- ts(Y_new2, start =1)
ts(Dates,start =1)

Y_Data <- data.frame(time = seq (1919,2020,length(102)),M=as.numeric(Y_new2),i
Venstre <- data.frame(time = seq (2021,2030,length(10)),M=as.numeric(Venstre_p
Social_Democrats <- data.frame(time = seq (2021,2030,length(10)),M=as.numeric(
Social_Liberals <- data.frame(time = seq (2021,2030,length(10)),M=as.numeric(S
Conservative <- data.frame(time = seq (2021,2030,length(10)),M=as.numeric(Cons

Dates1=c(2021:2030)
Y_finaldf <- cbind (Dates1, Venstre_pred,Social_Democrats_pred,Social_Liberals_
Y_finaldf <- data.frame(Y_finaldf)


ggplot(Y_finaldf, aes(x =Dates1)) +
  #geom_point(aes(y= Y_new2, color = "green"), size=2)+
  geom_point(aes(y= Venstre_pred, color = "Venstre"), size=2)+
  geom_point(aes(y= Social_Democrats_pred, color = "Social Democrats"), size=2
  geom_point(aes(y= Social_Liberals_pred, color = "Social Liberals"), size=2)+
  geom_point(aes(y= Conservative_pred, color = "Conservative"), size=2)+
  geom_point(aes(y= Independent_pred, color = "Independent"), size=2)+
  #theme(text = element_text(size = 16)) +
  labs(y='Number of Words') +
  labs(x='Year') +
  labs(color='Legend')
```