Data Analytics for EU Law

What is the proximity of political parties in the European Parliment ?

HEC Paris
European Business Law and Global Affairs
Année 2023 - 2024

Lucie Moutard
Julien Vacherot
Anatole de Rauglaudre

Our questions

Question: What is the proximity of political parties in the European Parliment ?

- Can we identify correlations among the politicial parties by analysing historical votes in plenary sessions?
- Can we predict a party's vote using a prediction model?

Method: scrapping the European Parliment Roll Call Votes and make a prediction model

1) Scrapping the European Parliament roll call votes from the European Parliament website, loading them in an object
2) Sanitize and optimize dataframe obtained (i.e. anomalies and missing values, dimension)
3) Analyse dataframe by identifying correlations between variables
4) Select and apply a model that predicts target party votes using independent variables
5) Gather and analyse results

Scrapping the European Parliament results of roll-call votes

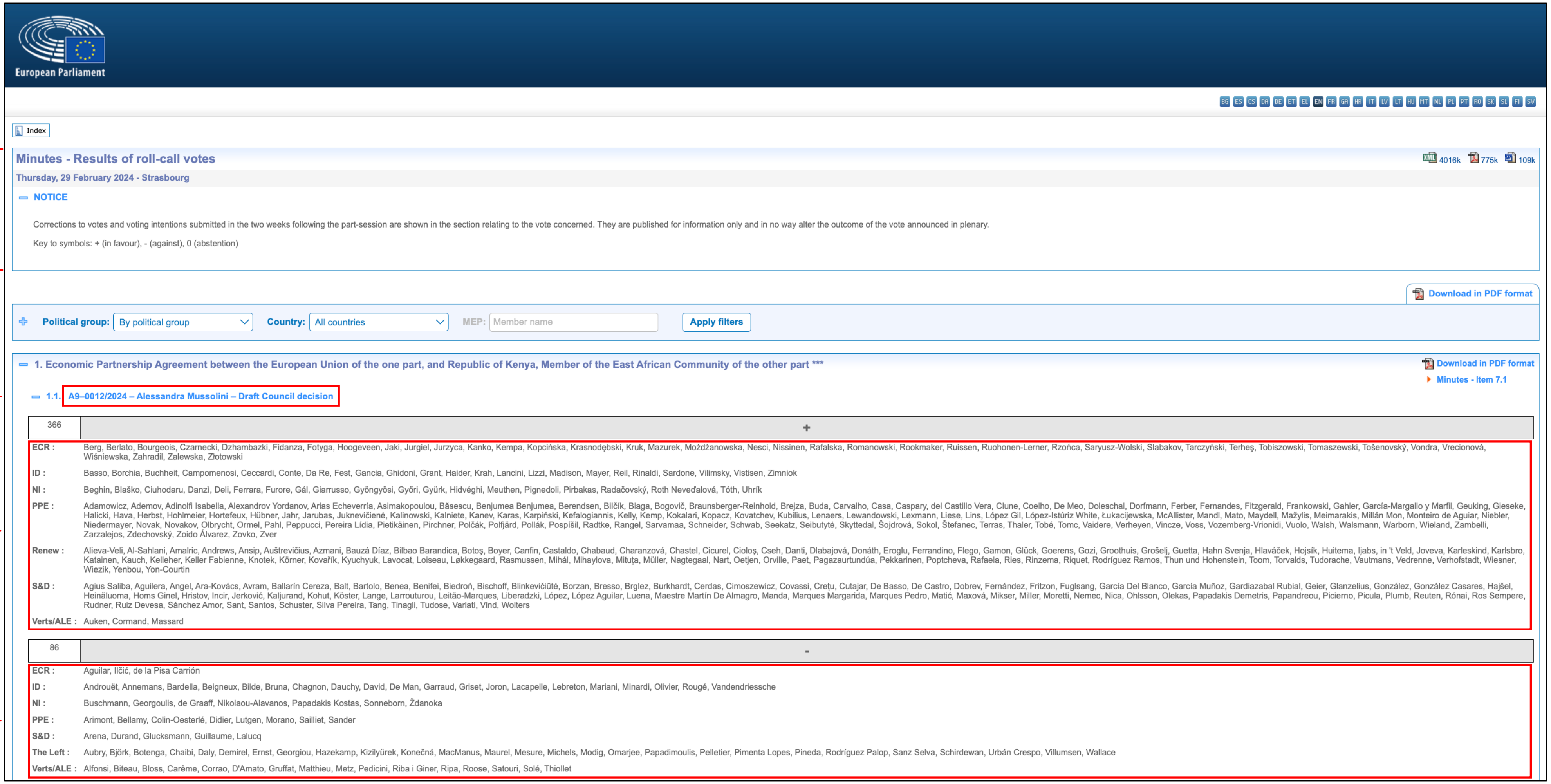
Structure of the results of roll-call votes page for each plenary

Results of roll-call votes section of Minutes of a particular plenary session

Title of the vote during the plenary

Members of each party who voted for

Members of each party who voted against



Scrapping the European Parliament results of roll-call votes

Using xml page

1) Create a function to generate the urls

```python
def generate_urls(start_year, end_year, end_month, end_day, urls=None):
    if urls is None:
        urls = []
    current_date = datetime.date(start_year, 1, 1)
    end_date = datetime.date(end_year, end_month, end_day)

    while current_date <= end_date:
        url = f"https://www.europarl.europa.eu/doceo/document/PV-9-{current_date.year}-{current_date.month:02d}-{current_date.day:02d}-RCV_EN.xml"
        urls.append(url)
        current_date += datetime.timedelta(days=1)

    return urls
```

1

2) Create a function to parse the target data

```python
def parse_xml(url):
    response = requests.get(url)

    if response.status_code == 200:
        soup = BeautifulSoup(response.content, "lxml-xml")

        roll_call_vote_descriptions = soup.find_all("RollCallVote.Description.Text")

        votes = []

        for roll_call_vote_description in roll_call_vote_descriptions:
            vote_name = roll_call_vote_description.text.strip()

            result_for_tag = roll_call_vote_description.find_previous("Result.For")
            result_against_tag = roll_call_vote_description.find_previous("Result.Against")

            if result_for_tag:
                for_identifiers = {}
                for political_group_list in result_for_tag.find_all("Result.PoliticalGroup.List"):
                    identifier = political_group_list["Identifier"]
                    members = political_group_list.find_all("PoliticalGroup.Member.Name")
                    for_identifiers[identifier] = len(members)
            else:
                for_identifiers = {}

            if result_against_tag:
                against_identifiers = {}
                for political_group_list in result_against_tag.find_all("Result.PoliticalGroup.List"):
                    identifier = political_group_list["Identifier"]
                    members = political_group_list.find_all("PoliticalGroup.Member.Name")
                    against_identifiers[identifier] = len(members)
            else:
                against_identifiers = {}

            vote = {
                "name": vote_name,
                "for": for_identifiers,
                "against": against_identifiers,
            }

            votes.append(vote)

    return votes
```

2

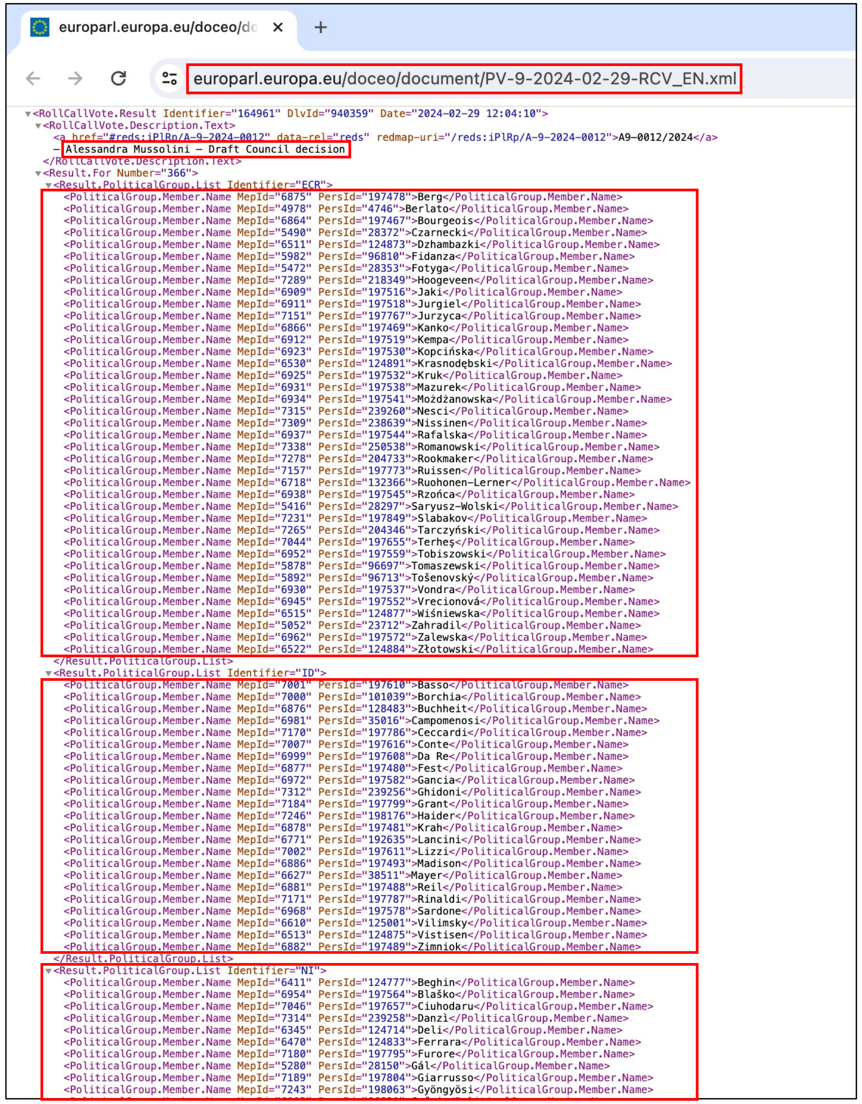
3

3

1 url structure

2 vote description

3 Members that voted for (against) for each party



Using xml page

3) Generating the object and converting to DataFrame

```
urls = generate_urls(2019, 2024, 3, 15)

all_votes = []

for url in urls:
    result = parse_xml(url)
    if result is not None:
        all_votes.extend(result)
```

```
df_all_votes = pd.DataFrame(all_votes)
df_all_votes.head(100)
```

	name	for	against
1	European Medicines Agency - A9-0216/2021 - Nic...	{'ECR': 59, 'ID': 68, 'NI': 21, 'PPE': 172, 'R...	{'ECR': 4, 'ID': 2, 'NI': 9, 'PPE': 1}
2	Objection under Rule 111(3): Determining cases...	{'ECR': 54, 'ID': 47, 'NI': 27, 'PPE': 174, 'R...	{'ECR': 7, 'ID': 18, 'NI': 6}
3	Objection under Rule 111(3): Determining cases...	{'ECR': 6, 'ID': 2, 'NI': 19, 'Renew': 73, 'S&...	{'ECR': 57, 'ID': 68, 'NI': 13, 'PPE': 172, 'R...
4	Digital Services Act - A9-0356/2021 - Christel...	{'ECR': 7, 'ID': 2, 'NI': 19, 'Renew': 74, 'S&...	{'ECR': 56, 'ID': 68, 'NI': 13, 'PPE': 172, 'R...
5	A9-0356/2021 - Christel Schaldemose - after Ar...	{'ECR': 44, 'ID': 26, 'NI': 26, 'PPE': 173, 'R...	{'ECR': 7, 'ID': 28, 'NI': 5, 'The Left': 3, '...
...	...	...	...
96	A9-0356/2021 - Christel Schaldemose - § 4 - Am...	{'ECR': 4, 'NI': 15, 'PPE': 1, 'Renew': 25, 'S...	{'ECR': 59, 'ID': 67, 'NI': 14, 'PPE': 172, 'R...
97	A9-0356/2021 - Christel Schaldemose - Recital ...	{'ECR': 3, 'ID': 23, 'NI': 15, 'Renew': 2, 'S&...	{'ECR': 57, 'ID': 47, 'NI': 14, 'PPE': 173, 'R...
98	A9-0356/2021 - Christel Schaldemose - Recital ...	{'ECR': 63, 'ID': 67, 'NI': 29, 'PPE': 173, 'R...	{'ID': 1, 'NI': 1, 'The Left': 2}
99	A9-0356/2021 - Christel Schaldemose - after Re...	{'ECR': 53, 'ID': 37, 'NI': 29, 'PPE': 117, 'R...	{'ECR': 7, 'ID': 32, 'NI': 1, 'PPE': 52, 'Rene...
100	A9-0356/2021 - Christel Schaldemose - Recital ...	{'ECR': 59, 'ID': 70, 'NI': 16, 'The Left': 2,...	{'NI': 13, 'PPE': 173, 'Renew': 100, 'S&D': 14...

4) Creating and cleaning a DF with only votes for

```
df_for_list = pd.DataFrame(for_list)
df_for_list.fillna(0, inplace=True) # replace NaN
df_for_list = df_for_list.drop(columns=['GUE/NGL']) #GUE/NGL is The Left, appears in some Minutes
df_for_list
```

	ECR	ID	NI	PPE	Renew	S&D	The Left	Verts/ALE
1	59.0	68.0	21.0	172.0	100.0	144.0	38.0	71.0
2	54.0	47.0	27.0	174.0	100.0	144.0	38.0	71.0
3	6.0	2.0	19.0	0.0	73.0	144.0	38.0	71.0
4	7.0	2.0	19.0	0.0	74.0	144.0	38.0	71.0
5	44.0	26.0	26.0	173.0	100.0	144.0	33.0	66.0
...	...	...	...	...	...	...	...	...
96	4.0	0.0	15.0	1.0	25.0	9.0	35.0	70.0
97	3.0	23.0	15.0	0.0	2.0	10.0	35.0	70.0
98	63.0	67.0	29.0	173.0	100.0	144.0	35.0	70.0
99	53.0	37.0	29.0	117.0	78.0	138.0	30.0	70.0
100	59.0	70.0	16.0	0.0	0.0	0.0	2.0	4.0

- ➢ Next? Dividing by number of members for each party to have percentage
- ➢ Some values are above the number of members in party, deleting those values

Using xml page

5) Making the dataframe percent

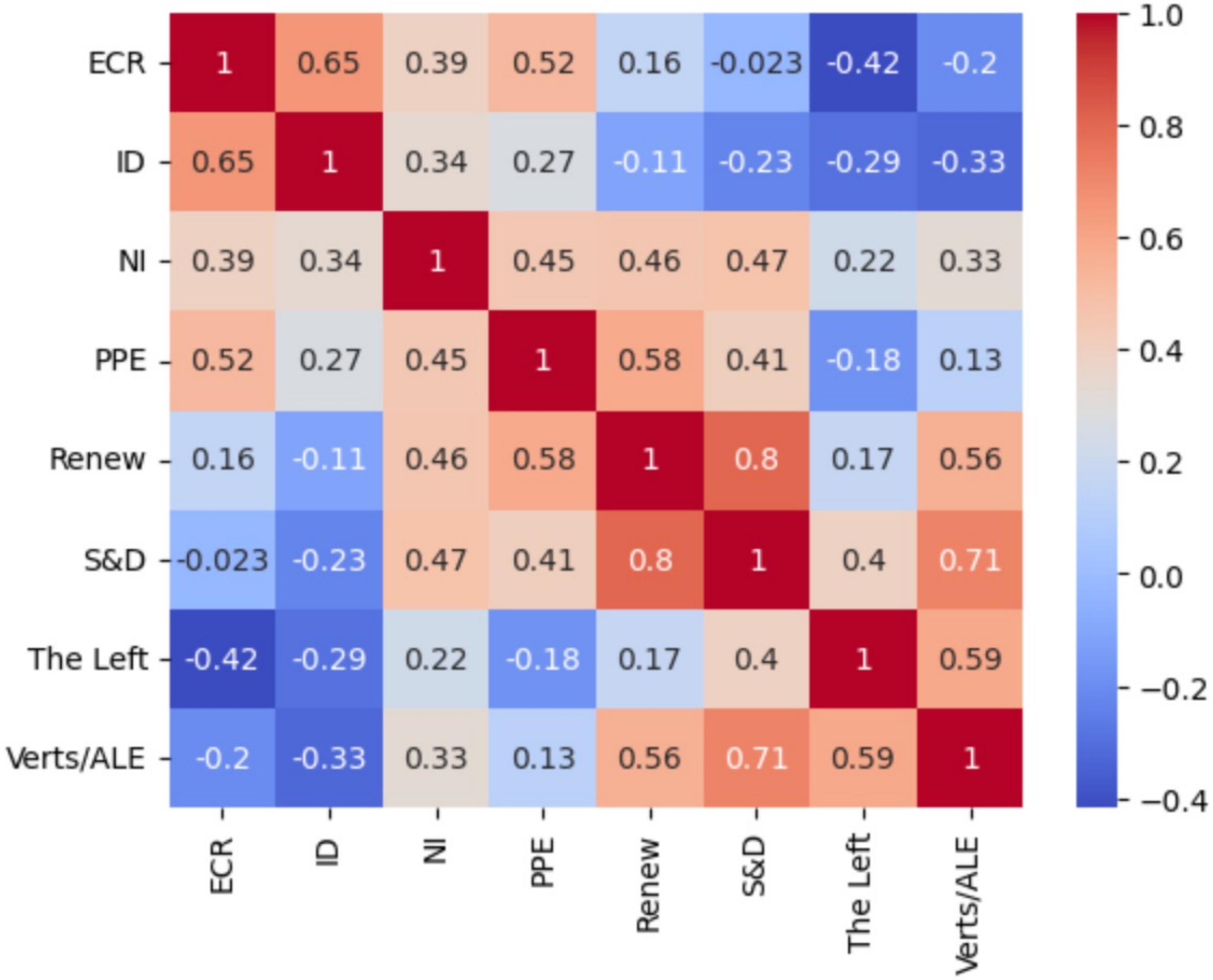
```
parliment = {'ECR':68.0, 'ID':59.0, 'NI':50.0, 'PPE':178.0, 'Renew':102.0, 'S&D':139.0, 'The Left':37.0, 'Verts/ALE':72.0}
parliment_df = pd.Series(parliment)
```

```
for party in df_for_list:
    for name in parliment_df.index:
        if party == name:
            df_for_list.drop(df_for_list[df_for_list[str(party)] > parliment_df[str(name)]].index, inplace = True)
        else:
            continue
```

```
df_for_percent = df_for_list / parliment_df * 100
```

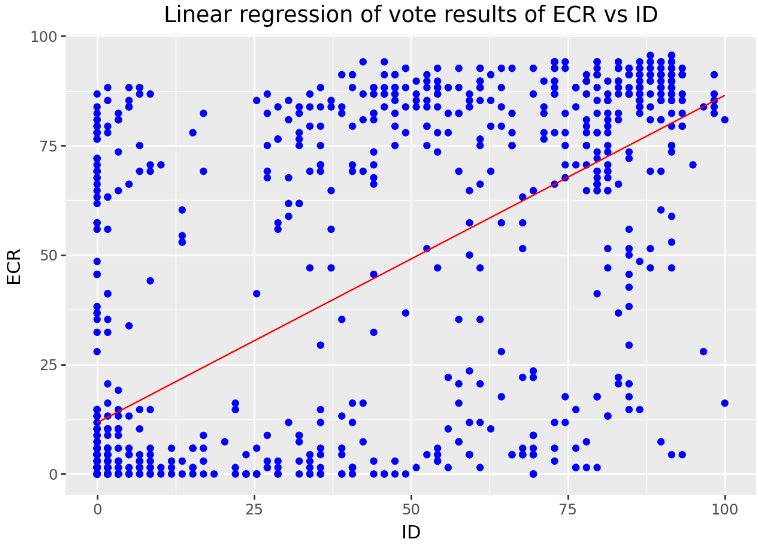
	ECR	ID	NI	PPE	Renew	S&D	The Left	Verts/ALE
7	76.470588	59.322034	50.0	67.415730	75.490196	100.000000	86.486486	97.222222
9	13.235294	38.983051	10.0	0.000000	14.705882	14.388489	10.810811	97.222222
11	23.529412	35.593220	32.0	0.000000	11.764706	16.546763	32.432432	97.222222
13	0.000000	52.542373	28.0	11.235955	0.000000	1.438849	97.297297	0.000000
16	23.529412	37.288136	30.0	0.000000	11.764706	17.266187	24.324324	97.222222
...	...	...	...	...	...	...	...	...
214	76.470588	33.898305	32.0	5.056180	2.941176	29.496403	94.594595	95.833333
215	75.000000	66.101695	32.0	7.865169	5.882353	23.741007	89.189189	95.833333
216	72.058824	71.186441	32.0	6.179775	28.431373	37.410072	91.891892	94.444444
217	8.823529	33.898305	30.0	2.247191	3.921569	15.107914	89.189189	95.833333
218	76.470588	33.898305	32.0	1.685393	0.000000	29.496403	91.891892	93.055556

A first approach : the correlation coefficient

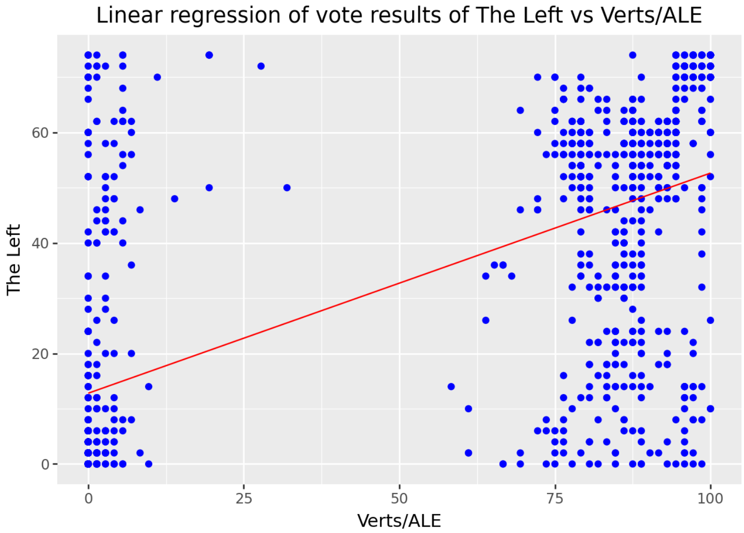


- Results that seems relevant with the current political dispositions of the European Parliament
- A strong opposition with the two extreme-wing-parties
- High correlation between left-wing and right wing parties
- The strongest correlation seems to be between Renew and S&D

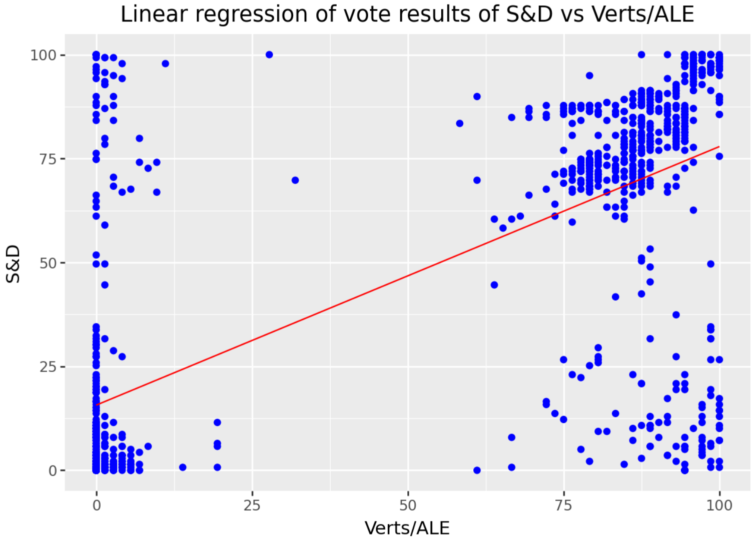
A second look : the simple linear regressions



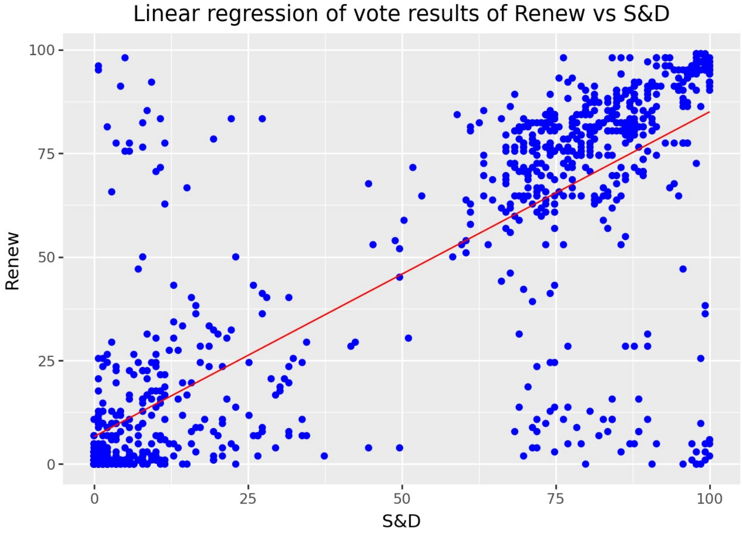
R2 =0.465335409100066



R2 =0.39331905561260894

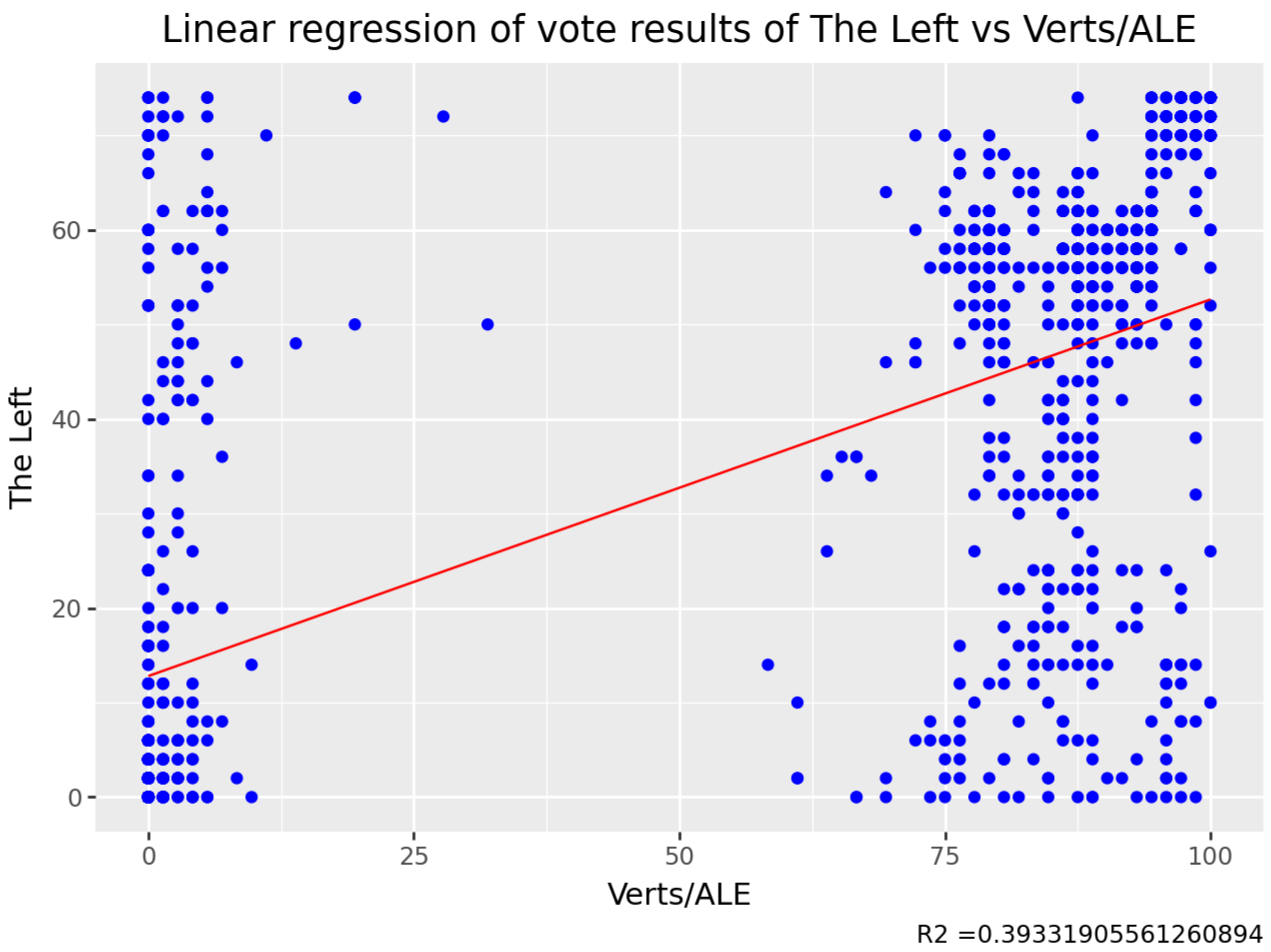


R2 =0.47539425779341293



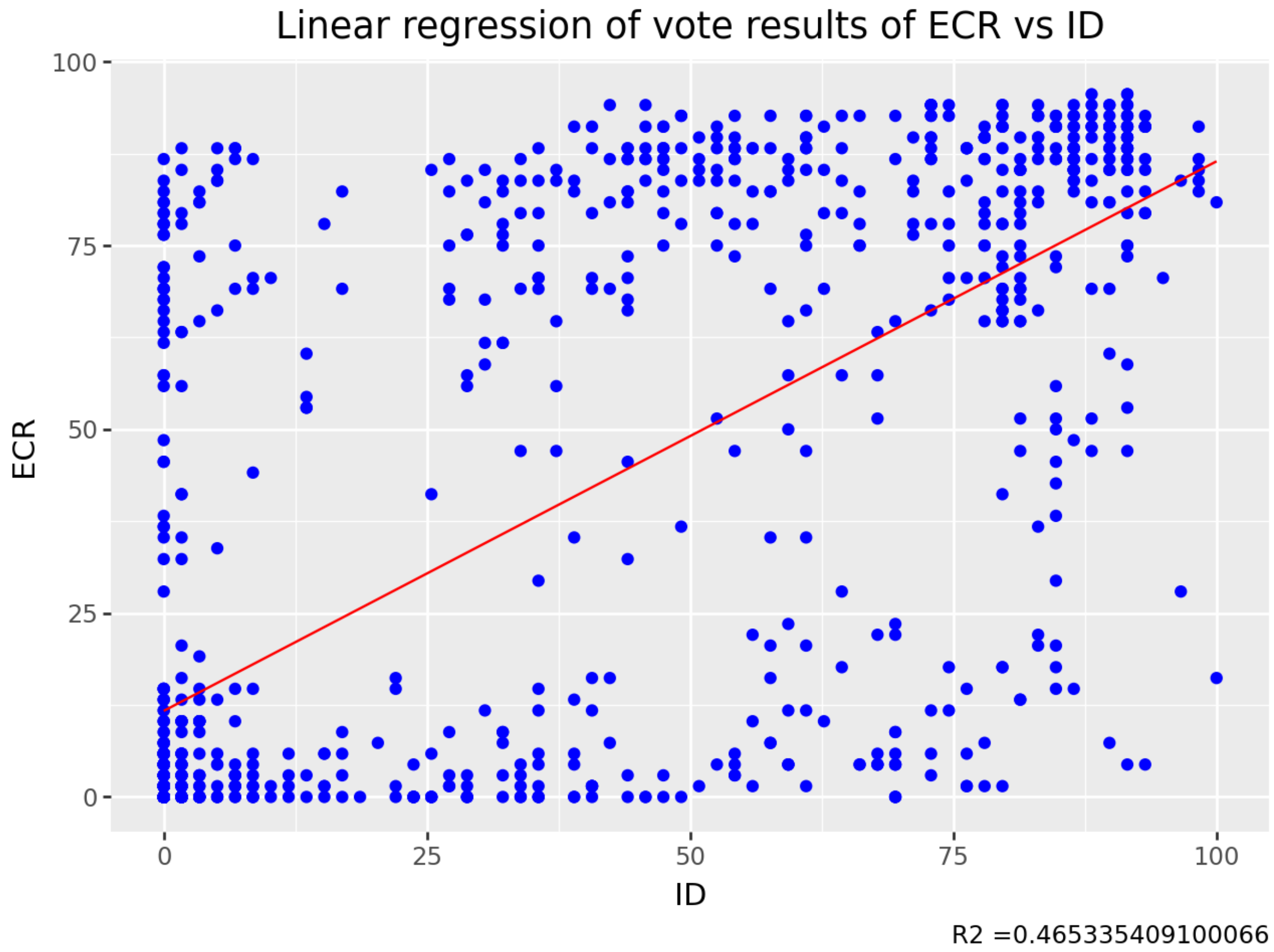
R2 =0.6512818489556982

Lesson 1 : No clear link between the vote of most parties seems decorrelated



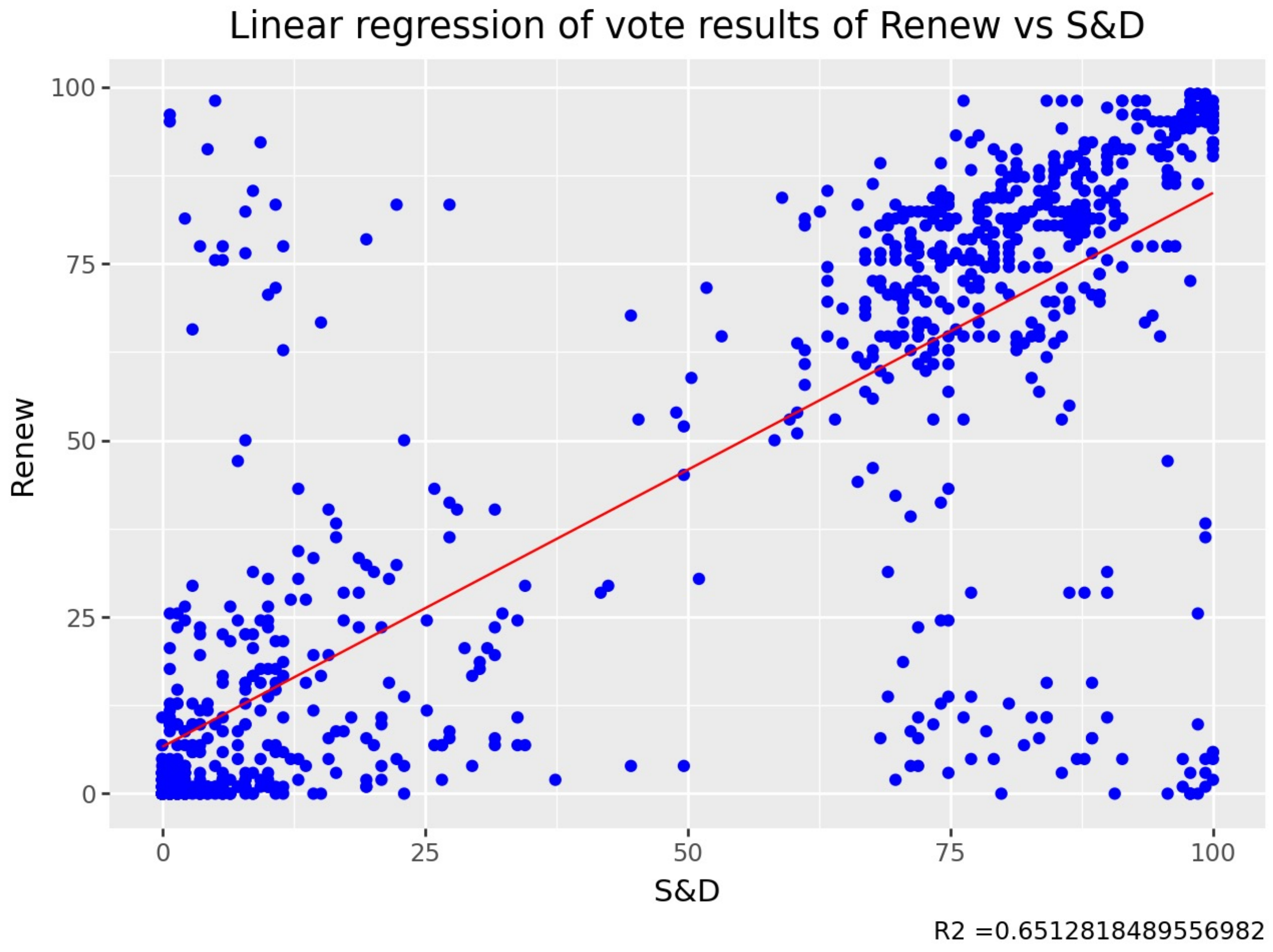
- We realized linear regressions between all the different groups in the European Parliament
- Unless for two political groups, we find no clear evidence of links between the votes of the different MPs.
- This results seems logical between parties which are not from the same side of the Parliament, but looks different from national parliaments, where parties with similar ideological basis usually vote often together

Lesson 2 : Even political groups with the same ideological basis seems not correlated



- Even political groups with strong similarities are not correlated with their votes
- The main example of this situation is the absence of evidence of links between ECR and ID.
- Indeed, they represent two extreme-right-wing political groups, but cannot match on many points.
- However, it has to be admitted that similarities exist on several votes.

Lesson 3 : One existing correlation : S&D and Renew



- A proximity in the speeches proved by our study (R^2>60% in the political field is an efficient proof of correlation)
- Concerning the votes where the political view of the two groups are clearly defined (more than 75% of vote for or against the text), clear proximity.
- This can looks quite strange from France, but is more logical in the EU, because of the central position of this two groups in the political landscape.

Conclusion : What about a third approach with multiple regression ?

```
#Defining our dependent and independent variables, and splitting for testing
y = test['S&D']
X = test[['Verts/ALE', 'Renew']]
X_train,X_test, y_train, y_test = train_test_split(X, y, test_size = 0.3, random_state = 100)
reg_model = linear_model.LinearRegression()
reg_model = LinearRegression().fit(X_train, y_train)

#Pair the feature names with the coefficients
list(zip(X, reg_model.coef_))

#Predicting the Test and Train set result
y_pred= reg_model.predict(X_test)
x_pred= reg_model.predict(X_train)

#Print return metric R2
r2 = metrics.r2_score(y_test, y_pred)
print('R2:', r2)
```

```
R2: 0.7916001834351883
```

- While linear regression gives no proper results about potential correlation between the dfferent political groups, it would make no sense to conclude that no proximity can be find between all groups in the Parliament

- As a result, we would need more solutions to find links between votes of more than two political groups.

- Multiple regression could be a solution. For example, we tested it with S&D as dependent variable and Les Verts/ALE et Renew as independent variables. The R^2 of 0,79 shows a strong strong fit