European public procurement (2012–2022): indicators and maps of competition and SME participation for cross-border analysis based on TED data

Manuel J. García Rodríguez

Doctor of Engineering and expert in Public Procurement

Project Engineering Area, University of Oviedo, Spain

3/11/2023

Summary

The field of public procurement is underexplored from an analytical standpoint, with limited reports based on large-scale data analysis of tender information. European countries have different national contracting platforms, and, in general, public procurement data is of low quality. Electronic systems carry out the public procurement process, but it is generally designed to manage documents (unstructured data). Therefore, it is not easy to obtain massive structured data for macro and microanalysis. The main objective of this report is to create indicators, graphs and maps for cross-border analysis at the country and regional level in Europe. Particularly, it focuses on studying the competition between bidders and the participation of small and medium-sized enterprises (SMEs). This report analyses 30 European countries from 2012 to 2022: the EU-27 Member States and the United Kingdom, Switzerland and Norway. The study includes a total of 3.1 million public contracts were awarded for an amount of 5,127 billion €. The study uses open data from TED, the European procurement platform. The findings indicate notable variations among countries and regions, emphasising the necessity for further integration of the common market to ensure cohesive and effective public procurement practices across Europe.

Keywords: public procurement, cross-border analysis, competition, small and medium-sized enterprises (SMEs), open data, European procurement platform (TED).

Abbreviations: artificial intelligence (AI), contracting authority entity (CAE), contract award notice (CAN), common procurement vocabulary (CPV), European Economic Area (EEA), European Union (EU), Gross Domestic Product (GDP), small and medium-sized enterprises (SME), tenders electronic daily (TED).

1. Introduction

Public procurement provides works, goods or services to the public contracting authorities (purchasing entities). It is an intensive and complex process and thus can consume significant resources. The European Union (EU) spends around 16% of its Gross Domestic Product (GDP) on public procurement [1]. Comparable to other OECD countries, EU governments spent, on average, about 30% of their total budget on public tenders [2]. Public procurement is not only economically significant with respect to expenditures but also concerning the income of potential suppliers.

Despite the importance of public procurement, there have been few data-based reports conducted by academic or government institutions (for example, read [3–6]). Public procurement has been deeply digitalised in the last 10 years, at a different pace in different European countries, allowing the appearance of statistics and reports. Currently, data quality in public procurement is still not high, and data management is not simple, so programming and data science knowledge is needed to manipulate the data and generate reports.

The main objective of this report is to generate relevant public procurement information and cross-border comparisons in Europe based on public procurement data. Particularly, it focuses on studying two important issues in public procurement: competition and the participation of small and medium-sized enterprises (SMEs). Another paper titled '*Study on the measurement of cross-border penetration in the EU public procurement market*' [7] is a very exhaustive report on the European Commission's public procurement data (TED) from 2016 to 2019, with some sections covering an even wider time frame (2009–2016). This report is not as comprehensive, but it incorporates updated data (2012–2022) and analyses more tenders. It is an open door to future studies by stakeholders such as public institutions, economists, jurists, data engineers, academic researchers and journalists. It is desirable that multidisciplinary teams collaborate due to the transversality of knowledge that public procurement requires: legal, economic, mathematical-statistical and data processing.

EU competition rules aim to ensure that all companies compete fairly and equally in the single market to the benefit of consumers, businesses and the European economy as a whole. Thus, it is necessary to study public procurement from the perspective of competition [8,9] and how contracting authorities can improve their tender procurement performance [10]; that is to say, to analyse whether public procurement is a free market that facilitates the participation of companies without legal or administrative barriers.

SMEs are deeply woven into the fabric of Europe. There are 25 million SMEs in Europe; 2 out of 3 European jobs are with SMEs, and they support 50% of Europe's GDP [11]. They are essential to Europe's competitiveness and prosperity, as well as its economic and technological sovereignty. However, SME participation in public procurement is still limited compared to their role in national economies. Several barriers pose difficulties to SMEs competing for and winning tenders, necessitating analysis of their role in public procurement [12]. With SMEs' strategy for a sustainable and digital Europe, the Commission wants to support and empower SMEs of all sizes and sectors, including their access to public procurement.

The report has the following scope:

- **Countries**: All EU Member States (27 countries) and the United Kingdom, Switzerland and Norway. In 2022, this constituted 30 countries with approximately 530 million inhabitants and 19,000 billion € of GDP.[1]
- **Time period**: This report covers the last 10 years, 2012–2022. Information has been available since 2006, but some of this is unsuitable due to the low quality of the data and the limited use of electronic formats in public procurement.
- **Dataset size**: 3.1 million contract award notices (CAN) for a total award amount of 5,127 billion €.

Public contracts above the thresholds set at the EU level have to be published EU-wide on the Tenders Electronic Daily (TED) platform. The aim is to integrate procurement markets across the EU, increase competition and obtain better value for public money spent. For several years, the integrity, transparency and efficiency of public spending have increasingly become a focus of wider public attention and concern.

Supplying public procurement data in an open and reusable format not only serves to generate statistics and reports but also helps to improve public procurement processes and detect corruption [13], among other purposes [14,15]. For example, academic articles have created tools based on artificial intelligence (AI) to develop predictive models for detecting fraud and corruption [16–20], detecting collusion (bid-rigging by cartels) [21–25], forecasting the number of bidders [26,27] or the award price [28–30] and recommending bidders for tenders [31]. Hence, it is possible to create innovative applications in public procurement using disruptive technologies such as AI [32–34].

This report is structured in a summary and 5 sections. Section 1 introduces the study. Section 2 describes the data source and objectives of the report. Section 3 presents the methodology for data analysis, the pillar of this report. Section 4 shows the main results (graphs and maps) and explains the most significant findings. Finally, the conclusions and recommendations are summarised in Section 5.

[1] Information available in Eurostat: https://ec.europa.eu/eurostat/databrowser/view/namq_10_gdp/default/table?lang=en

2. Data source and objectives of the study

Tenders Electronic Daily (TED) is the data source for this study. TED is the online version[2] of the Supplement to the Official Journal of the EU (OJEU), dedicated to European public procurement. It serves as the primary source of information on public procurement notices within the EU, the European Economic Area (EEA), and beyond. TED provides access to a wide range of business opportunities by allowing suppliers to search for and respond to public procurement notices from various EU institutions, bodies, and agencies, as well as from contracting authorities from EU Member States and other countries.

TED has evolved significantly in response to the digital age, becoming a crucial platform for promoting transparency and fair competition in public procurement. It plays a vital role in facilitating the single market by ensuring that all European businesses, regardless of size or location, have equal access to information on public procurement opportunities. Additionally, TED ensures compliance with EU regulations on public procurement,[3] contributing to harmonising procurement practices across the EU Member States.

The primary objective of TED is to promote an open and competitive public procurement process. By providing a centralised platform for publishing and accessing procurement notices, TED fosters fair competition and enables SMEs to participate in public contracts alongside larger corporations. This not only supports the growth and competitiveness of European businesses but also encourages innovation and the development of a diverse and dynamic market. By providing a comprehensive and accessible platform for public procurement information, TED plays a significant role in fostering a competitive and fair business environment, encouraging cross-border trade, and supporting the growth and development of businesses within the EU and beyond.

What notices (or simply tenders) must be published on TED above the economic thresholds defined in the EU regulations?

1. Notices from the countries of the Public Procurement Network (EU Member States, candidate countries, EEA countries and Switzerland):
    a. Public works, supplies and services.
    b. Utilities contracts (water, energy, transport and telecommunications sectors).
    c. Concessions contracts.
2. Notices from the EU institutions, agencies and other bodies:
    a. Public works, supplies and services.
    b. Calls for expression of interest.
3. Notices for projects financed by the European Investment Bank, European Investment Fund, European Central Bank, European Bank for Reconstruction and Development and External Aid and European Development Fund.

TED has an open data service to reuse the information. All notices are published in XML format for reuse according to Commission decision 2011/833/EU on the reuse of Commission documents. The data source used for this study has been downloaded from the website of the official portal for European data, specifically, the dataset called '*Tenders Electronic Daily* –

[2] https://ted.europa.eu

[3] Directive 2014/23/EU on concession contracts; Directive 2014/24/EU on public procurement; Directive 2014/25/EU on water, energy, transport and postal services; Directive 2009/81/EC on public procurement in the defence field. More information is available at https://single-market-economy.ec.europa.eu/single-market/public-procurement/legal-rules-and-implementation/thresholds_en

public procurement notices.'[4] This dataset includes the most important fields from the contract notices and contract award notices, such as who bought what from whom, for how much, and which procedure and award criteria were used. Generally, the data consists of tenders above the procurement thresholds, but publishing below-threshold tenders in TED is considered good practice, and thus a non-negligible number of below-threshold tenders are also present.

The objective of this report is to answer relevant questions about public procurement by comparing European countries and regions based on TED data. This study will focus on the following topics, which contribute to promoting fairness, accountability, and effective decision-making in public procurement, ultimately fostering a competitive and inclusive marketplace.

- **General indicators**. Cross-border analysis of types of contract, types of procedure, types of contracting authority and the most important CPVs in economic terms. These indicators play a pivotal role in providing a comprehensive understanding of the overall landscape and dynamics of the public procurement process. These indicators facilitate the identification of trends, patterns, and potential areas for improvement or optimisation within public procurement.
- **SME indicators**. The evolution of SME winners for all countries, the percentage of tenders won by SMEs at the country level (and also at the regional level) and a comparison of SME indicators versus competition indicators. These indicators offer insights into the participation and evolution of SMEs in public tenders, enabling the assessment of their impact on the market. Analysing the percentage of tenders won by SMEs across countries and regions allows for the identification of potential disparities and the formulation of targeted policies to support SMEs in accessing public contracts. Comparing SME indicators with competition indicators provides a comprehensive view of the competitive landscape, aiding policymakers in evaluating the effectiveness of procurement regulations and their impact on market competition.
- **Competition indicators**. The percentage of tenders without competition (i.e. tenders with one bidder) at the country level and the regional level. This figure provides an understanding of potential monopolistic tendencies and the need for fostering a more competitive environment. The percentage of tenders whose winners are foreign (i.e. the winners are not from the contracting authority's country) at the country and regional level is also analysed, offering insights into the level of international participation and aiding in evaluating the openness of the public procurement market.

Currently, low data quality limits data-driven analyses of public procurement. The information published in TED notices is dependent on the accuracy of the data supplied by contracting authorities. The fields associated with economic amounts can contain errors or omissions, and the fields associated with the winners' information are sometimes poorly filled out or homogenised (for example, the same company is given different names). These issues will be examined in the methodology section. This study has attempted to homogenise the winners' information and discard tenders that contain anomalous values to achieve reliable results that show reality without distortions.

[4] https://data.europa.eu/data/datasets/ted-csv

3. Data analysis methodology

A robust methodology is crucial for ensuring reliable and replicable results in data analysis. A well-defined approach helps maintain consistency in data collection, processing, and analysis, minimising errors and ensuring result accuracy. By establishing clear guidelines for data cleaning and preparation, a methodology contributes to data quality, ensuring that the information used is precise and complete. This promotes the reliability of the findings and enables other researchers to replicate the analysis, reinforcing the credibility of the conclusions. Additionally, an effective methodology streamlines the analysis process, saving time and resources while also facilitating a deeper understanding of patterns and trends within the data. Ultimately, a strong methodology fosters transparency, consistency and confidence in the data analysis process, leading to more informed decision-making and meaningful insights.

The methodology for this report is described below.

1. **Gather tenders**. The first step is to gather the European public procurement information published in TED. As mentioned in the previous section, the dataset is '*Tenders Electronic Daily – public procurement notices*' from the website of the official portal for European data. This dataset has many fields which describe the contract award notice (tender) process from 2006 to 2022. Table 1 describes the fields that will be used for the study. For more details about the dataset and its fields, see the technical document '*TED CSV open data.*'[5] The scope and time period considered in this study are:
    a. Countries: EU Member States (27 countries) and the United Kingdom, Switzerland and Norway. All regions of the EU Member States are considered in this report, including the outermost regions[6] (if they report data), except on maps on which the outermost regions are not shown.
    b. Time period: Only the last 10 years, 2012–2022, have been considered for this report. Information has been available since 2006, but some of it is unsuitable due to the low quality of the data and the limited use of electronic formats in public procurement.
2. **Delete tenders (contract award notices)** without enough information, duplicates or outliers. The data is of low quality, as recognised in the previously mentioned technical document.[7] The following steps have been taken to select a set of tenders with high-quality data for the study.
    a. Delete tenders if the following fields are empty:
        i. WIN NAME[8] and WIN NATIONALID.
        ii. VALUE EURO, VALUE EURO FIN 1 and VALUE EURO FIN 2.
        iii. AWARD EST VALUE EURO, AWARD VALUE EURO and AWARD VALUE EURO FIN 1.

[5] https://data.europa.eu/api/hub/store/data/ted-csv-data-information-v3-5.pdf

[6] Some EU Member States have part of their territory located in areas of the globe that are remote from Europe, known as the outermost regions. There are 9 outermost regions: Guadeloupe, French Guiana, Réunion, Martinique, Mayotte and Saint-Martin (France), the Azores and Madeira (Portugal) and the Canary Islands (Spain). More information is available at https://www.europarl.europa.eu/factsheets/en/sheet/100/outermost-regions-ors-

[7] '*The data is provided "as is." The source of the data is unverified output from contracting authorities or entities across Europe. It is not uncommon for data to be input incorrectly or be missing, and thus great care must be taken with data management and interpretation*.'

[8] The prefix WIN means winner.

iv. ISO COUNTRY CODE, B MULTIPLE COUNTRY and ISO COUNTRY CODE ALL.

b. Delete tenders if the field CANCELLED is not 1.
c. Delete tenders if the field INFO ON NON AWARD is 'procurement unsuccessful' or 'procurement discontinued'.
d. Delete duplicate tenders; that is, tenders that have the same values in the fields ID NOTICE CAN, CAE NAME,[9] ID LOT, LOTS NUMBER, ID AWARD, ID LOT AWARDED and WIN NAME.
e. Delete tenders whose fields associated with economic amounts[10] have outliers (anomalous values). The maximum amount allowed is 20,000 million €.
f. Delete tenders whose fields associated with economic amounts have small values. The minimum amount allowed is 10,000 €. These are well below the economic thresholds defined by EU regulations and could distort the analysis.
g. Delete tenders if the award amount (AWARD_VALUE_EUR_FIN_1) is 10 times greater than tender amount (VALUE_EURO_FIN_2). The reason is that the tender amount is manually corrected by DG GROW if it has very large abnormal values.

3. **Clean and transform the tender data**. It is necessary to clean and process the data to make it usable for this project. This may involve correcting errors, filtering and formatting the data consistently and grouping firms that belong to the same company.
   a. Clean some fields of contracting authority (CAE NAME, CAE TOWN) and winner (WIN NAME, WIN TOWN, WIN POSTAL CODE): convert to uppercase, delete accentuation, delete multiple spaces, delete strange characters (commas, colons, quotation marks, parentheses and the like) and so on.
   b. Clean the IDs (national registration numbers) of the contracting authority (CAE NATIONALID) and the winner (WIN NATIONALID): convert to uppercase, delete multiple spaces, delete strange characters (commas, colons, quotation marks, parentheses, and the like), delete substrings (VAT, IDE, REGISTER and NUMBER), delete strings with abnormal sizes (too small or large) and so on.
   c. Unify the names of the winners and their IDs (national registration numbers) if the same company has multiple names (the ID is unique, so a company cannot have multiple IDs). Several functions have been programmed to unify the names and IDs or fill them when they are empty.
   d. Add geolocation information (latitude/longitude and NUTS[11]) to winners and contracting authorities. Regional analysis of public procurement can be carried out thanks to NUTS level 3. This regionally disaggregated analysis offers an innovative perspective in the field of public procurement.

[9] The prefix CAE means contracting authority entity.
[10] VALUE EURO, VALUE EURO FIN 1, VALUE EURO FIN 2, AWARD EST VALUE EURO, AWARD VALUE EURO and AWARD VALUE EURO FIN 1.
[11] The NUTS classification (nomenclature of territorial units for statistics) is a hierarchical system for dividing up the economic territory of the EU and the UK. NUTS subdivides each country into 3 levels: NUTS 1, NUTS 2 and NUTS 3. The second and third levels are subdivisions of the first and second levels. For example, NUTS 2: Provincies/Provinces in Belgium; Comunidades y ciudades autónomas in Spain; Régions in France; Länder in Austria. For example, NUTS 3: Amtskommuner in Denmark; départements in France; län in Sweden; megyék in Hungary; kraje in Czechia; oblasti in Bulgaria. More information available in Eurostat: https://ec.europa.eu/eurostat/web/nuts/background

4. **Analyse the tenders**. Use statistical techniques and descriptive analysis to identify and create indicators, ratios and trends. Create tables, graphs and maps to display the results clearly. All of these steps were developed with a custom Python program.[12]
5. **Interpret the results**. This involves drawing conclusions from the analysis and explaining what the results mean from a temporal, geographical and economic perspective. Technical and policy recommendations will be made based on the analysis. This will be covered in Sections 4 (Results and Discussion) and 5 (Conclusions and Recommendations).

The analysis for SME (see section 4.2 SME indicators) is based on the field called 'B CONTRACTOR SME' (see Table 1). Six considerations about this field:

- It was introduced in 2016 (version 2.0.9[13]). Furthermore, it has low data quality (it is not always informed) but it has increased over the years.
- However, it has been possible to fill this field in the period 2012–2015 thanks to clean, transform and get the contractor information from 2016–2022. If a company is an SME in 2016–2022, we assume that it was also an SME in 2012–2015 (and vice versa).
- After the data cleaning, percentage of tenders that do not have the field informed by year: 2012 (63.1%), 2013 (61.8%), 2014 (59.3%), 2015 (55.5%), 2016 (35.5%), 2017 (18.2%), 2018 (13.4%), 2019 (9.7%), 2020 (10.0%), 2021 (9.2%) and 2022 (8.9%).
- After the data cleaning, percentage of tenders that do not have the field informed by country: Cyprus (58.0%), Greece (47.5%), United Kingdom (40.8%), Italy (39.0%), Slovenia (37.8%), Belgium (35.7%), Luxembourg (31.1%), Hungary (30.7%), Malta (29.0%), Switzerland (28.6%), Poland (28.0%), Lithuania (27.9%), France (26.1%), Portugal (26.1%), Germany (25.2%), Romania (25.2%), Finland (22.2%), Austria (21.6%), Ireland (20.9%), Netherlands (19.7%), Estonia (19.7%), Norway (18.7%), Latvia (17.4%), Slovakia (17.3%), Denmark (16.3%), Bulgaria (16.2%), Spain (15.2%), Czech Rep. (14.1%), Croatia (14.0%) and Sweden (4.9%).
- To calculate the total number of tenders for the SME analysis, we only consider tenders that have the field informed.
- When a tender has several winners, the tender is considered as SME if one of the winners (at least) is an SME.

Finally, the criterion to classify the tenders by country is the field called ISO_COUNTRY_CODE (see Table 1). All types of contracting authorities are included, there are not filters. Hence, EU institutions and other unspecified contracting authorities are also included.

[12] Python is a high-level and versatile programming language with an extensive standard library and third-party modules. It has gained widespread popularity due to its user-friendly syntax and the extensive range of applications it supports. Python is used across various domains, including data analysis, artificial intelligence, scientific computing and automation.

[13] Version of the XML schema definition used by the Publications Office of the EU to publish the data. Higher versions mean better average quality of data.

Table 1. Description of the dataset

Topic	Field	Description
Tender information	ID NOTICE CAN	Unique identifier of the contract award notice (CAN)
	YEAR	Year of publication of the notice
	DT DISPATCH	The date when the buyer dispatched (sent) the notice for publication to TED
	CANCELLED	1 = this notice was later cancelled
	CAE NAME	Official name of Contracting Authority Entity (CAE)
	CAE NATIONALID	CAE's national registration number
	CAE TOWN	CAE's town
	ISO COUNTRY CODE	"Country" for the first listed authority
	B MULTIPLE COUNTRY	There are contracting authorities from at least 2 different countries
	ISO COUNTRY CODE ALL	If the variable above is yes, then this variable contains the list of all countries
	CAE TYPE	Type of contracting authority: Ministry or national authority, Regional/local authority, Utilities sectors, EU institution, Other international organisation, Body governed by public law, Other, National Agency, Regional/local Agency or Not specified
	TYPE OF CONTRACT	Type of contract: works, supplies or services
	B FRA AGREEMENT	The notice involves the establishment of a framework agreement
	B FRA CONTRACT	This notice is probably about specific contracts within a framework agreement
	CPV	The main Common Procurement Vocabulary code of the main object of the contract
	ID LOT	Lot identifier
	LOTS NUMBER	The number of lots for a given CAN. This value is based on the number of unique "Lot No" values
	VALUE EURO	CAN value, in EUR, without VAT. If the value was not present, the lowest bid is used instead
	VALUE EURO FIN 1	CAN value, in EUR, without VAT. If a value variable is missing, this variable looks for it in all other fields from which it could be taken
	VALUE EURO FIN 2	Generally the same value as VALUE EURO FIN 1, but -if available- overwritten by human-made estimates of values for large value contracts which seemed to be incorrect
	TOP TYPE	Type of procedure: award without prior publication of a contract notice, competitive dialogue, negotiated without a call for competition, negotiated with a call for competition, open, restricted or innovative partnership
Award information	ID AWARD	Unique contract award identifier. This is the identifier for all variables at the contract award level
	ID LOT AWARDED	Lot No, an identifier of a lot within this contract award
	INFO ON NON AWARD	If the variable is empty, then a contract was awarded
	WIN NAME	Official name of the winner (contractor)
	WIN NATIONALID	National registration number of the winner. E.g. a VAT number of a business registry number
	WIN TOWN	Winner's town
	WIN POSTAL CODE	Winner's postal code
	WIN COUNTRY CODE	Winner's country
	B CONTRACTOR SME	The contractor is an SME (SME as defined in Commission Recommendation 2003/361/EC)
	TITLE	Title of the contract award notice
	NUMBER OFFERS	Number of tenders received
	NUMBER TENDERS SME	Number of tenders received from SMEs
	NUMBER TENDERS OTHER EU	Number of tenders received from tenderers from other EU Member States
	NUMBER TENDERS NON EU	Number of tenders received from tenderers from non-EU countries
	AWARD EST VALUE EURO	Estimated CA value, in EUR, without VAT
	AWARD VALUE EURO	Total final CA value, in EUR, without VAT. If the value was not present, the lowest bid is included
	AWARD VALUE EURO FIN 1	CA value, in EUR, without VAT. If the value variable is missing, this variable looks for it in all other fields from which it could be taken
	DT AWARD	Date of contract award

4. Results and discussion

This section presents indicators, graphs and maps that help understand public procurement between European countries and also between their regions. The section is divided into 3 subsections: general indicators, SME indicators and competence indicators.

The results are based on a dataset of 3,093,902 contract awards[14] (sometimes simply called tenders) from 30 countries (27 EU Member States, the United Kingdom[15], Switzerland and Norway) over the period 2012 to 2022 after applying the methodology described in the previous section (initially there are 8,461,603 tenders). The total award amount is about 5,127 billion € (exactly 5,127,262,371,585 €.) Unless otherwise stated, the graphs and maps refer to the specified 10-year time interval. Countries that are not part of the study are coloured grey on the maps.

4.1 General indicators

For any data-based report, it is essential to know the data's main magnitudes to evaluate its scope and depth. Figure 1 shows the number of tenders for the 30 countries from 2012–2022. In fact, the number of tenders depends on several factors: the size of the country, the number of public administrations that purchase through public procurement, and the value of their tenders (if they are small, they are not required to be published on TED). France has the greatest number of tenders (626,614), and Malta has the least (4,084). There are substantial differences between the countries with the most tenders, France and Poland, and the rest of the countries. This is neither positive nor negative; the objective of this graph is to quantify the volume of tenders to be analysed.

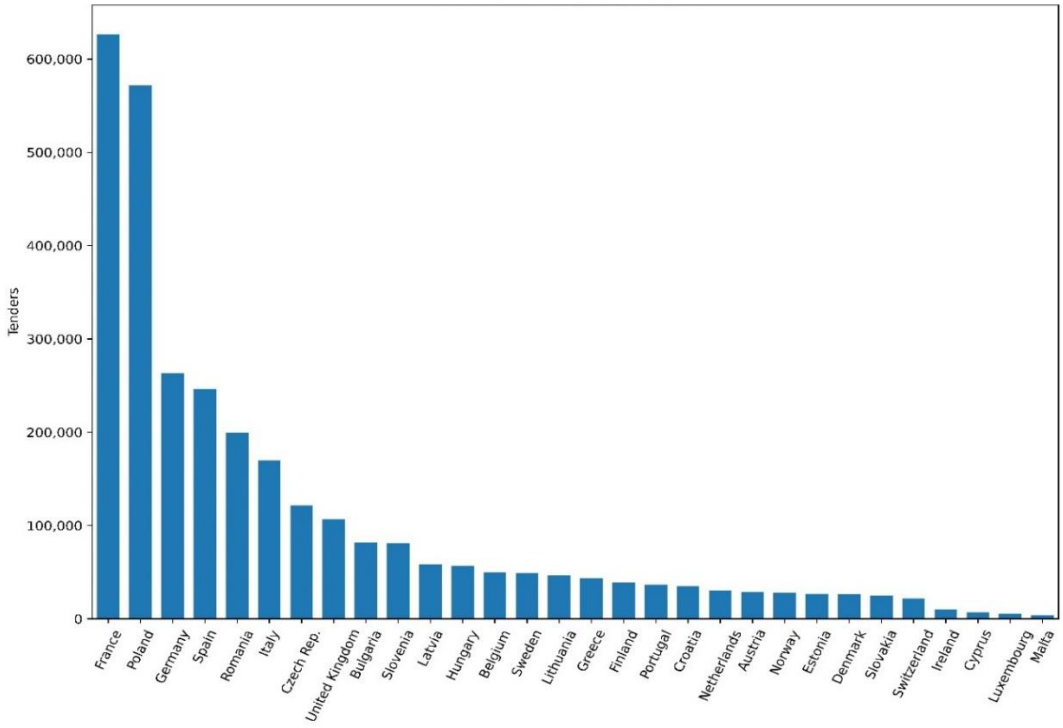

Figure 1. Total number of tenders by country for 2012–2022.

[14] This refers to a unique contract award identifier, which is the identifier for all variables at the contract award level. It is the field ID AWARD in Table 1.

[15] Note that the withdrawal of the United Kingdom from the EU has resulted in fewer tenders being published on TED. In this report, the United Kingdom had 9,033 tenders in 2016, 11,321 tenders in 2017, 15,209 tenders in 2018, 16,727 tenders in 2019, 12,436 tenders in 2020, 6,262 tenders in 2021 and 2,141 tenders in 2022.

Figure 2 shows the number of tenders by country represented on a map for a visual and simple interpretation. The 30 countries have been classified into 5 groups, from lowest to highest number of tenders. Each group comprises 6 countries:

- Purple (between 4,084 and 25,962 tenders): Malta, Luxembourg, Cyprus, Ireland, Switzerland and Slovakia.
- Dark blue (between 25,962 and 36,069 tenders): Denmark, Estonia, Norway, Austria, Netherlands and Croatia.
- Dark green (between 36,069 and 52,306 tenders): Portugal, Finland, Greece, Lithuania, Sweden and Belgium.
- Light green (between 52,306 and 131,259 tenders): Czech Republic, Hungary, Latvia, Slovenia, Bulgaria and the United Kingdom.
- Yellow (between 131,259 and 626,614 tenders): Italy, Romania, Spain, Germany, Poland and France.

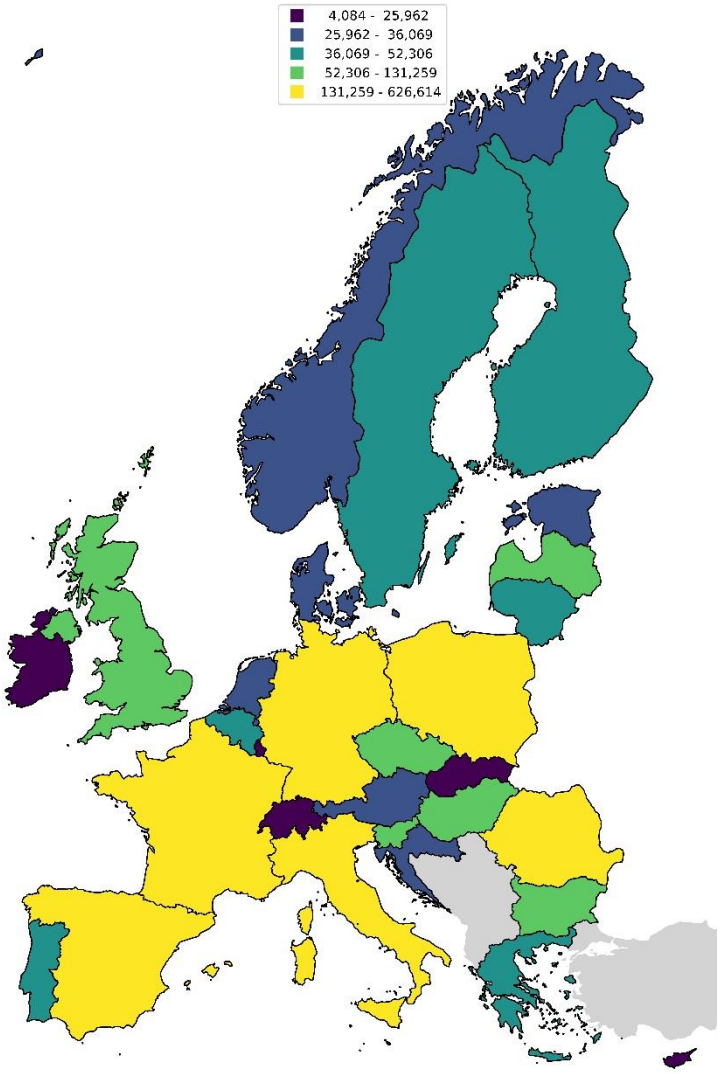

Figure 2. Total number of tenders by country for 2012–2022. The countries are classified into 5 groups by number of tenders (low to high: purple, dark blue, dark green, light green and yellow).

When maps are displayed in this report, countries will be classified by groups (usually 4 groups) for a better comparison between them. The classification criterion is that each group has the same number of countries (more or less). Therefore, if there are 30 countries classified into 4 groups, each group has 7 or 8 countries.

Figure 3 shows the volume of tenders in economic terms, according to the award price (contract award value,[16] in € without VAT). There are some notable differences in the number of tenders (see Figure 1). For example, the United Kingdom is in first place in economic terms, but it is in eighth place in the number of tenders. This means that their tenders have a very high average award price. The opposite case is Romania. After the United Kingdom, the countries that spend the most on tenders are France, Italy, Spain, Poland and Germany (all spend above 300 thousand million €). These countries also have high numbers of tenders. There are considerable differences between the 2 highest-spending countries, the United Kingdom and France, and the rest of the countries in economic terms during the 10 years 2012–2022.

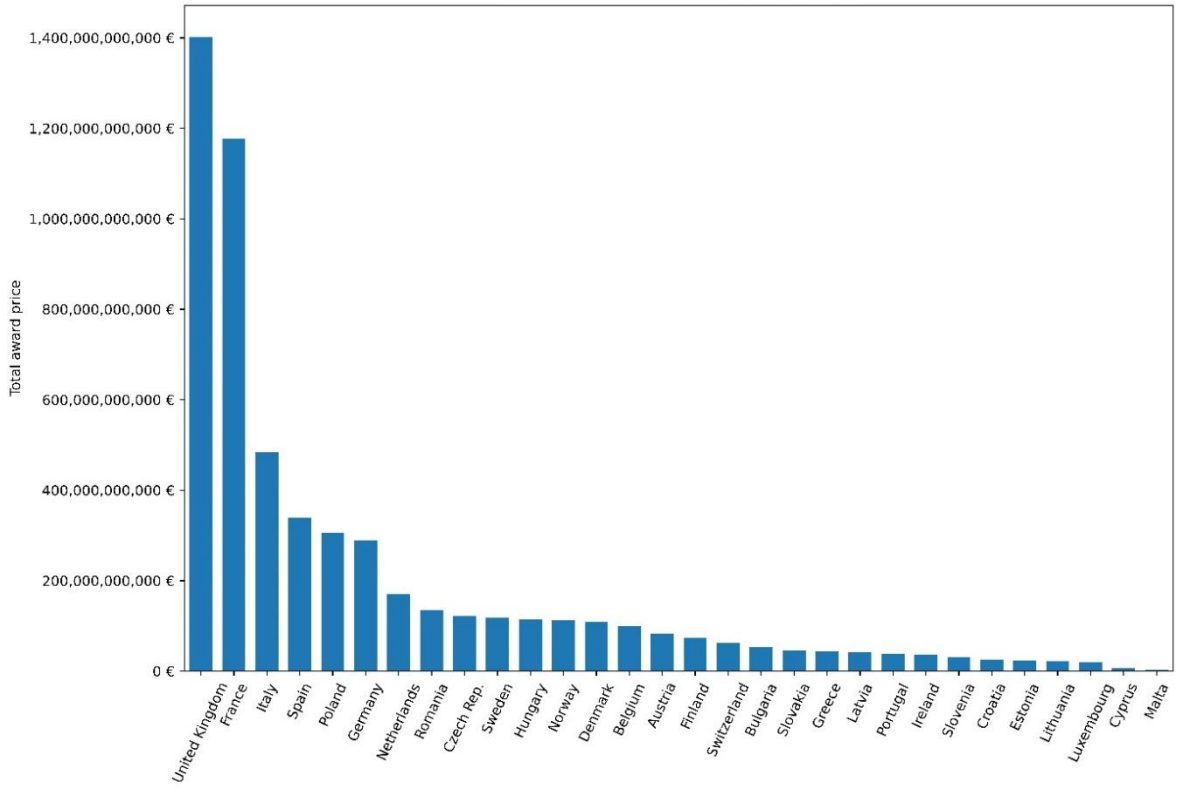

Figure 3. Total award price by country for 2012–2022.

Figure 4 shows the type of contract (supplies, services and works) by country according to the award price. Note that the award price is expressed in relative economic terms (a percentage) to allow comparisons between countries, not in absolute terms (€) as in the previous figure. In the following figures, the award price will be expressed in relative economic values. Figure 4 shows the significant differences between the countries. For example, the countries with the lowest percentage of supply contracts are France, Netherlands and Switzerland. The countries that allocate the most economic resources to the purchase of services are France and the United Kingdom. Among the large European countries, Spain spends the least on public works.

[16] The field AWARD VALUE EURO FIN 1 in Table 1.

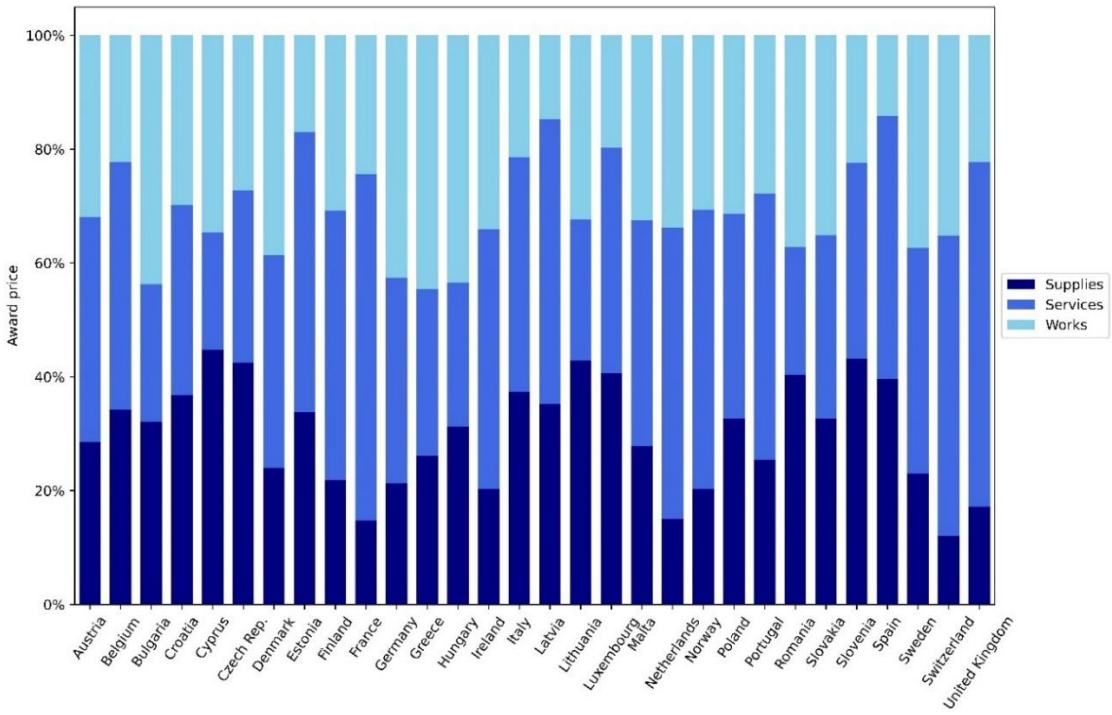

Figure 4. Type of contract over total award price by country for 2012–2022.

Figure 5 shows the type of procedure (open, negotiated with/without a call for competition, restricted and others) in tenders according to the total award price. There are significant differences between countries and no common behaviour. Open procedures facilitate the participation of companies and improve the free market. The countries with the lowest percentage of open procedures are Denmark, France, Ireland and the United Kingdom. Those with the highest percentage are Switzerland,[17] Malta and Greece.

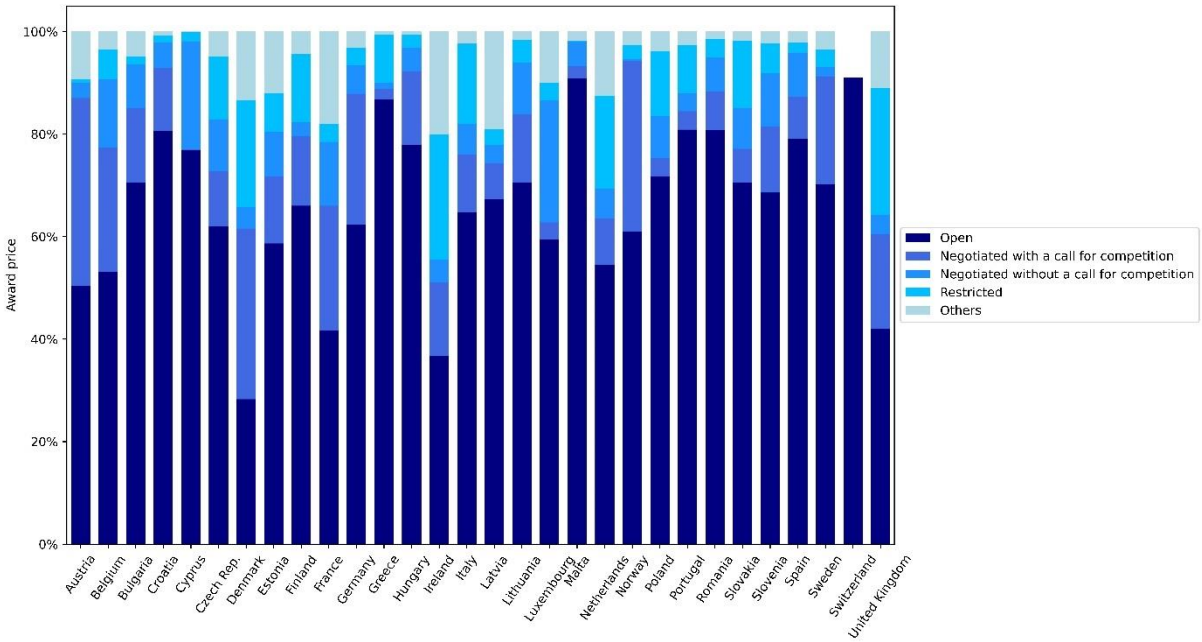

Figure 5. Type of procedure as a percentage of total award price by country for 2012–2022.

[17] Note that in Switzerland, only the open procedure appears and does not reach 100% due to tenders where the type of procedure is not given.

Figure 6 represents the type of contracting authority (see legend) of the tenders according to the award price. Geographical or administrative causes strongly influence this graph. For example, small countries like Cyprus, Luxembourg and Malta have almost no regional or local authorities. Belgium and Luxembourg have a higher percentage of EU institution or international organisation tenders because many of these organisations are headquartered there. The countries with the lowest percentage of national authority tenders are Austria, Belgium, Croatia, Finland, France, Germany, Hungary, Italy, Poland and Portugal.

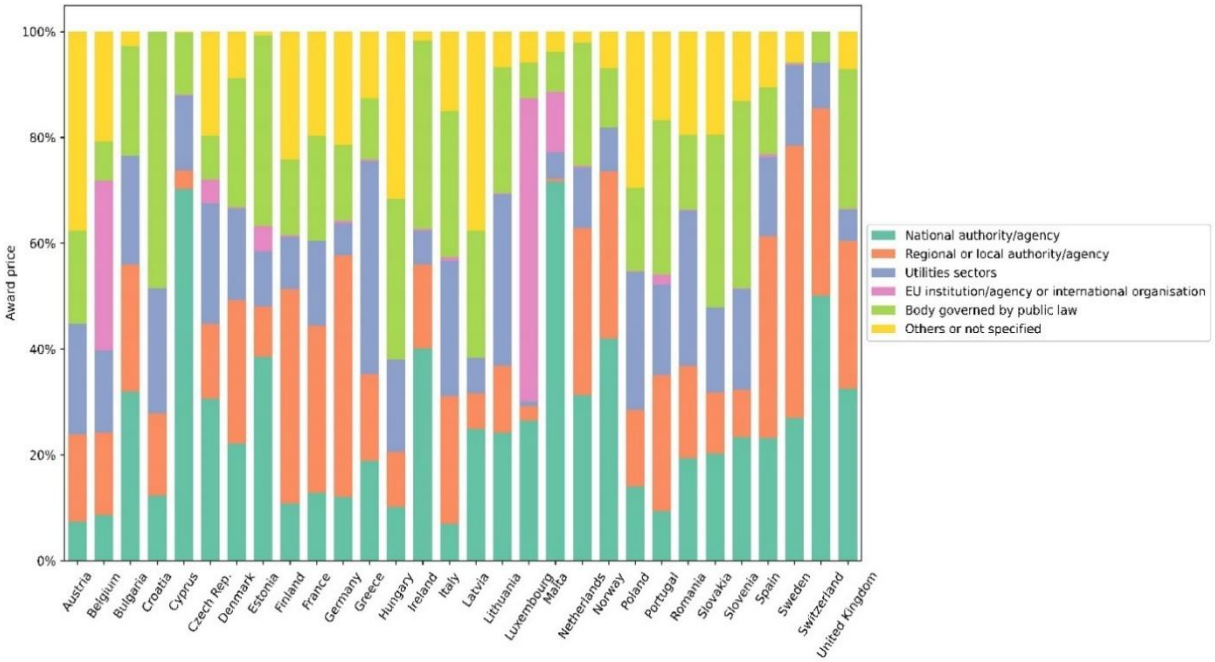

Figure 6. Type of contracting authority as a percentage of total award price by country for 2012–2022.

Figure 7 shows 9 CPV codes (see legend) which are very relevant in economic terms, accounting for more than 50% of the total tender award price of each country. In some countries (Bulgaria and Switzerland), they reach almost 80%. In this analysis, there is a clear pattern for all countries: construction work (CPV 45) is the most important (around 35% of the total award price). This is not new; historically, construction works have been the most significant projects in public procurement. In small countries, such as Cyprus and Luxembourg, medical equipment (CPV 33) constitutes a very high percentage of expenditure (around 30%); this percentage is much lower in the rest of the countries.

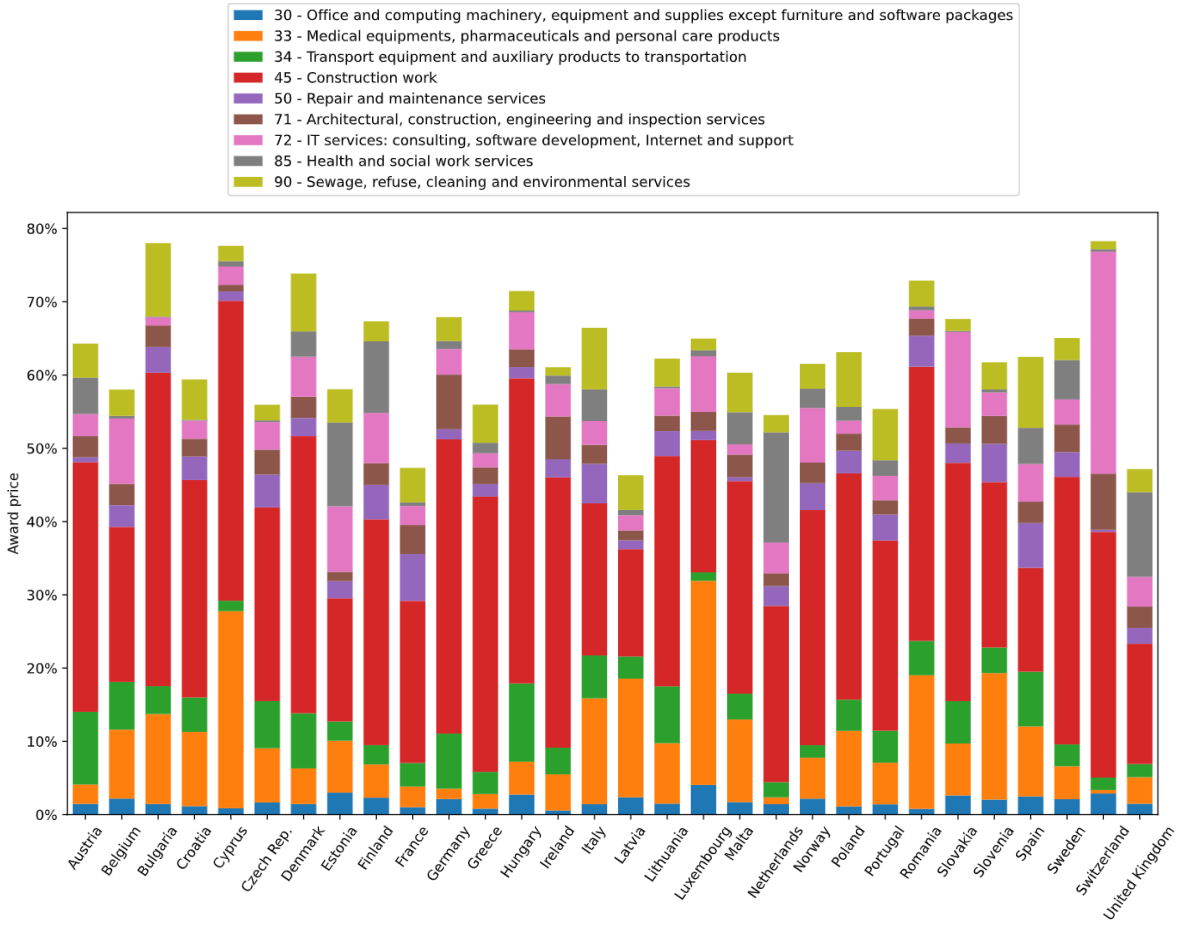

Figure 7. The most important CPVs in terms of total award price by country for 2012–2022.

Finally, Figure 8 shows the temporal evolution of the total number of companies that have won tenders in all 30 countries. Since 2019, this has been around 140 thousand companies. It is an approximation because, as mentioned in the methodology section, the data is low quality, particularly in the fields associated with the winner. It must also be considered that the actual number of public procurement contract winners will be higher because this study only considers tenders published on TED (that is, tenders published on national platforms are not taken into account).

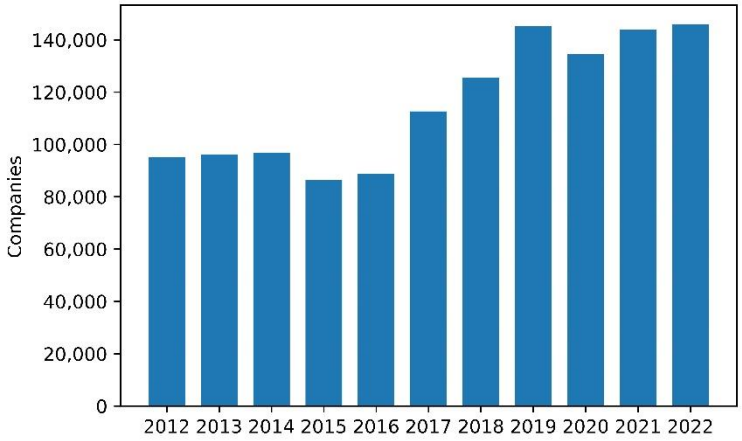

Figure 8. Evolution of the number of winners (different companies) for all countries.

4.2 SME indicators

It is important to look at indicators that show the participation of SMEs in public procurement to gauge the evolution and current reality of the market. Figure 9 shows the percentage of SME winners in all countries from 2012 to 2022 with a double perspective: according to the number of tenders (on the left) and according to the total award price (on the right). In 2022, SMEs won approximately 60% of tenders; however, they won barely 45% in 2012. Therefore, there has been a positive evolution over the years. In economic terms, on the right, it can be observed that the percentage of award prices for SMEs remained constant at around 30% from 2017 to 2022. Therefore, although SMEs win more tenders, large companies get more money (70% of the total award price in recent years).

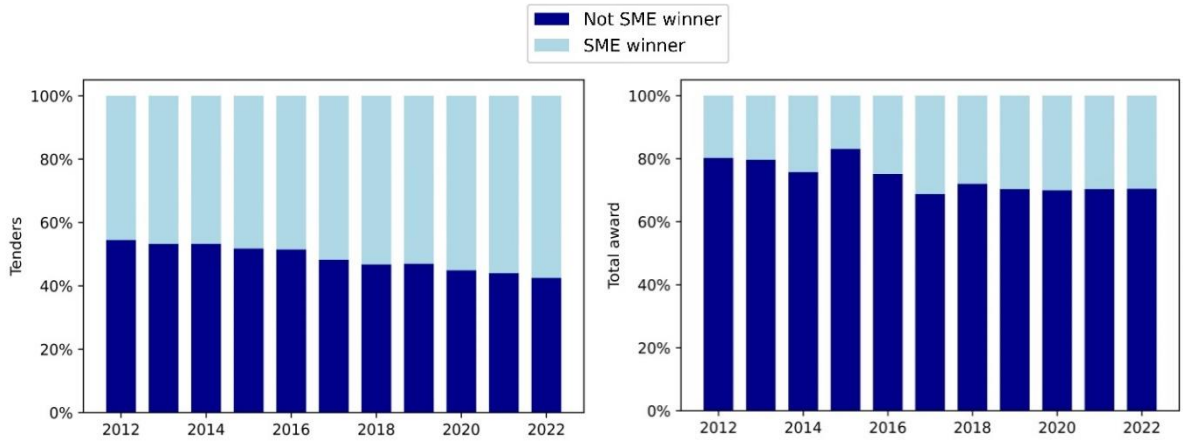

Figure 9. Evolution of SME winners for all countries. On the left is the percentage of tenders with SME winners (light blue) and non-SME winners (dark blue). On the right is the total award price for SME winners (light blue) and non-SME winners (dark blue).

Figure 10 shows the evolution of the percentage of the total award price for SMEs winners by country over 4 time periods (2016, 2018, 2020 and 2022). The countries with the lowest percentage are coloured red, and those with the highest percentage (60%) are coloured green. Some countries have a positive evolution (like Germany); others do not (like the United Kingdom). Historically, the 3 countries with the highest percentage are Switzerland, Estonia and Latvia. However, a country's small size does not seem to be directly related to a higher percentage because Ireland, Belgium and Denmark have low percentages.

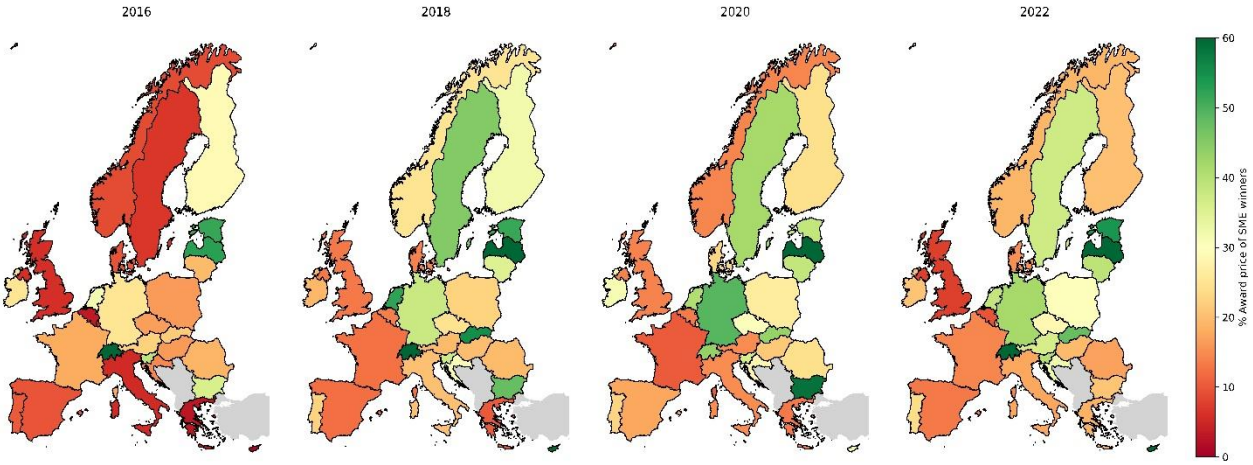

Figure 10. Percentage of total award price for SMEs winners by country and year (2016, 2018, 2020 and 2022).

Figure 11 shows the percentage of tenders won by SMEs for each country from 2012–2022. The 30 countries have been classified into 4 groups (each group has more or less the same number of countries):

- Dark red (30.8%–43.6%). 8 countries: Norway, Belgium, France, Portugal, Italy, Greece, Spain and Denmark.
- Light red (43.6%–60.5%). 7 countries: United Kingdom, Finland, Poland, Luxembourg, Austria, Croatia and Romania.
- Light green (60.5%–74.5%). 7 countries: Ireland, Netherlands, Germany, Sweden, Czech Republic, Hungary and Bulgaria.
- Dark green (74.5%–98.3%). 8 countries: Estonia, Latvia, Lithuania, Switzerland, Slovakia, Slovenia, Malta and Cyprus.

Of the 5 largest European economies (Germany, France, Spain, Italy and the United Kingdom), all are in the 2 worst groups (red), except Germany, which is in the light green group. There is no clear relationship between country size and the percentage of tenders won by SMEs (as seen in Figure 10). For example, small countries like Luxembourg, Norway and Denmark do not have high percentages.

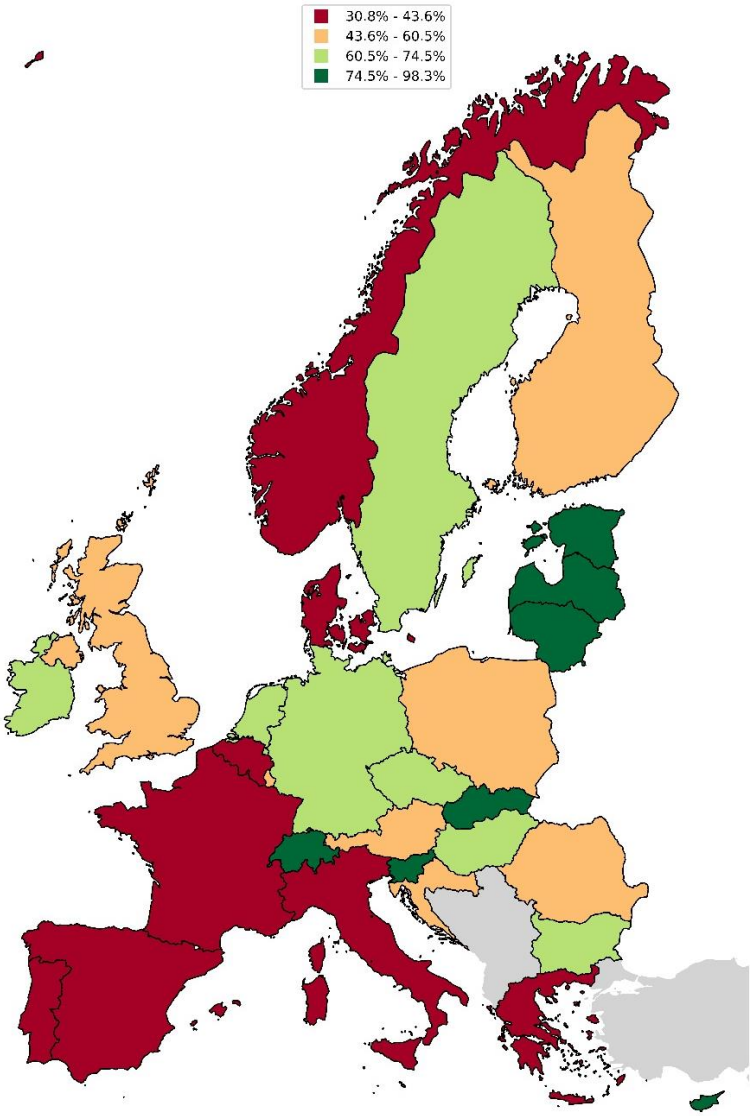

Figure 11. Percentage of tenders won by SMEs, 2012–2022. The countries are classified into 4 groups from smallest to greatest percentage (dark red, light red, light green and dark green).

For more information, you can consult the website Single Market Scoreboard[18] developed by the European Commission. You can compare these results with the "SME contractors" indicator on the website. Both are very similar and if there is any difference, note that the results on the website are for one year (2021 is the last year available) and this study is for 10 years. In addition, it has other indicators such as "SME bids" (it measures the proportion of bids from SMEs), "single bidder" (it will be studied in the next section), award cirteria, decision speed, etc.

Tenders are also segmented at the regional level (NUTS 3), as mentioned in the methodology section. Therefore, the country-level map can be disaggregated to the regional level[19] to analyse behaviour within countries and make cross-border comparisons.

Figure 12 shows the percentage of tenders won by SMEs in each region between 2012–2022. Analogously, the regions are classified into 4 groups: dark red (6.1%–40.3%), light red (40.3%–53.1%), light green (53.1%–66.6%) and dark green (66.6%–100%). There are 7 countries in which every region is in the best group (dark green): Estonia, Latvia, Lithuania, Slovakia, Switzerland and the islands of Malta and Cyprus. Conversely, almost all regions of Belgium are in the worst group (dark red). In general, Western and Southern European regions (except Ireland) have a lower percentage of SME winners than Eastern and Northern European regions (except Norway).

[18] Website https://single-market-scoreboard.ec.europa.eu/business-framework-conditions/public-procurement_en

[19] In this report, regions coloured white have no tenders, region borders are red, and country borders are black.

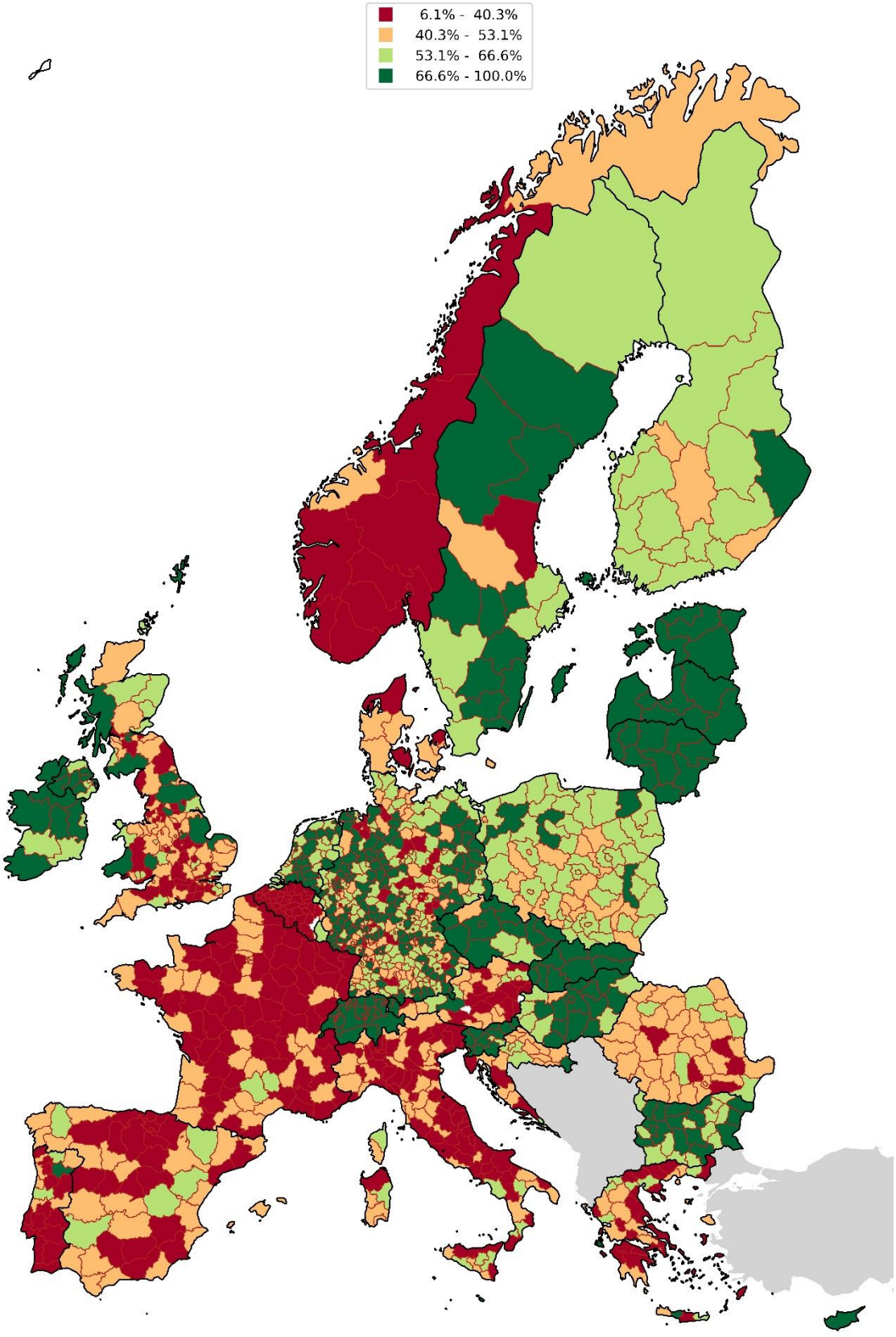

Figure 12. Percentage of tenders won by SMEs by region for 2012–2022, classified into 4 groups from smallest to greatest percentage (dark red, light red, light green and dark green) at the European regional (NUTS 3) level.

Figure 13 shows a scatter plot[20] of 2 variables at the country level for 2012–2022: SME winners (vertical axis) versus competition (horizontal axis). The vertical axis is the percentage of tenders whose winners are SMEs, and the horizontal axis is the percentage of tenders with 2 or more offers (bidders). The circle's size is the total number of tenders for each country (France has the biggest circle and Malta the smallest).

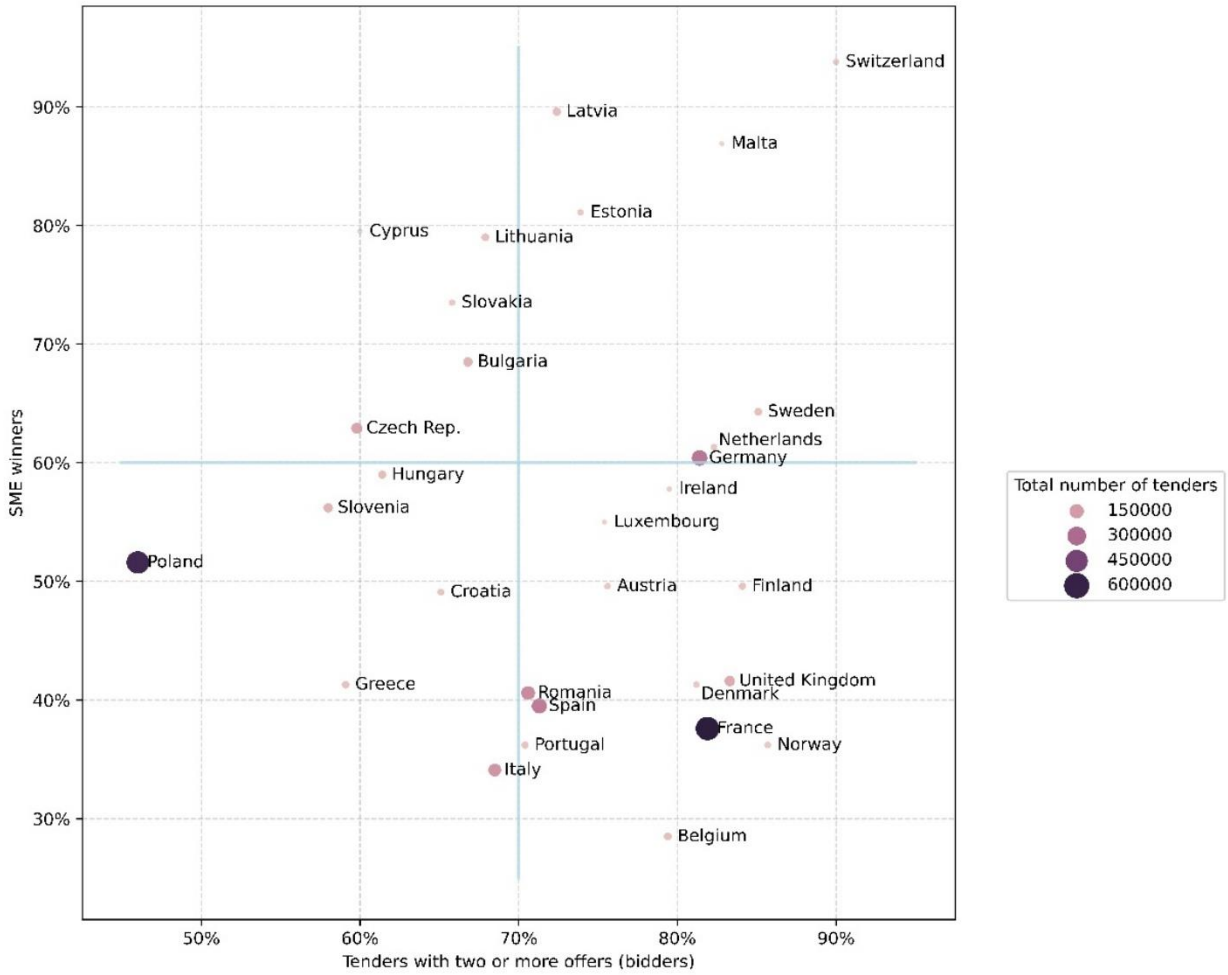

Figure 13. SME winners (vertical axis) versus competition (horizontal axis). The vertical axis is the percentage of tenders whose winners are SMEs, and the horizontal axis is the percentage of tenders with 2 or more offers (bidders).

If a country is located at the top right position in Figure 13, its market has a high percentage of SME winners and high competition. Conversely, if a country is located at the bottom left position, the country has a less inclusive and competitive market. If we divide the scatter plot into 4 quadrants along the blue lines in the graph, the countries are grouped in the following way:

- SME winners > 60% and competition > 70%. This quadrant has a high number of SME winners and high competition. It comprises 7 countries: Switzerland, Malta, Latvia, Estonia, Sweden, Netherlands and Germany.

[20] A scatter plot (also called a scatter graph or scatter chart) is a type of plot or mathematical diagram using Cartesian coordinates to display values, typically of two dataset variables. The data are displayed as a collection of points, where one variable determines the position on the horizontal axis, and the other determines the position on the vertical axis. If the points are coded (colour/shape/size), one additional variable can be displayed.

- SME winners > 60% and competition < 70%. 5 countries: Cyprus, Lithuania, Slovakia, Bulgaria and the Czech Republic.
- SME winners < 60% and competition > 70%. 12 countries: Ireland, Luxembourg, Austria, Finland, Romania, United Kingdom, Spain, Denmark, France, Portugal, Norway and Belgium.
- SME winners < 60% and competition < 70%. These countries have a low number of SME winners and low competition. 6 countries: Hungary, Slovenia, Croatia, Poland, Greece and Italy.

Of the 5 largest European economies, only Germany is in the best quadrant, but it is almost in the same quadrant as the United Kingdom, France and Spain. The cutoff values that create the quadrants (60% of SME winners and 70% of competition) are approximately the European average values.

There are substantial differences between countries, but there is no evidence of geographic, economic or country size patterns. Note that this graph (as well as the rest of the report) contains all the tenders for each country. However, we saw in the previous subsection that there are significant differences between countries depending on the type of contract, type of procedure, type of contracting authority and the most relevant CPVs. Moreover, there are more variables not shown in this report (for example, tender price, award criteria and number of lots per tender) that certainly also affect the participation of SMEs and competition. Therefore, it would be interesting to carry out more detailed studies which consider specific types of tenders (with the same 'configuration' of variables) to make more realistic comparisons between countries or regions.

4.3 Competition indicators

We will further analyse competition in public procurement, considering any type of winner (SME or not). Two indicators have been created at the country/regional level for 2012–2022, and the results are displayed on maps:

1. Tenders without competition. The percentage of tenders with one offer (bidder). This indicator is the opposite of the indicator referred to as 'tenders with 2 or more offers (bidders)' in the previous subsection.
2. Foreign winners. The percentage of tenders whose winners are not from the contracting authority's country. If foreign companies win public contracts, the country has easy and understandable public procurement procedures, and its market is internationalised (free transit of goods, people and capital).

Figure 14 shows the percentage of tenders with one offer (bidder) at the country level. A high percentage implies that the country has less competition. The 30 countries have been classified into 4 groups:

- Dark green (10%–18.2%). 8 countries: Norway, Sweden, Finland, United Kingdom, Netherlands, France, Switzerland and Malta.
- Light green (18.2%–26.9%). 7 countries: Ireland, Belgium, Germany, Denmark, Austria, Luxembourg and Estonia.
- Light red (26.9%–34%). 7 countries: Portugal, Spain, Italy, Romania, Bulgaria, Latvia and Lithuania.
- Dark red (34%–54%). 8 countries: Poland, Czech Republic, Slovakia, Hungary, Slovenia, Croatia, Greece and Cyprus.

The highest percentages of single bidder tenders are concentrated in Eastern and Mediterranean European and Mediterranean countries. Conversely, the Atlantic countries have the lowest percentages.

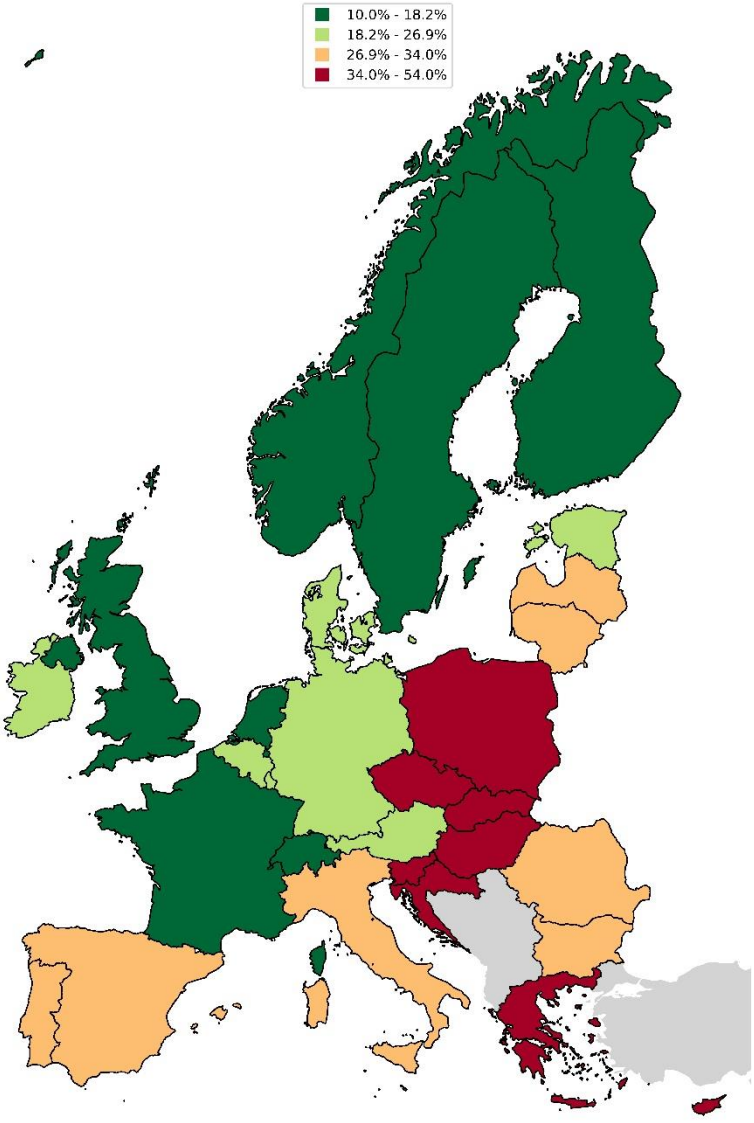

Figure 14. Tenders without competition by country for 2012–2022. Percentage of tenders with one offer (bidder). The countries are classified into 4 groups (low to high percentage: dark green, light green, light red and dark red).

Figure 15 shows the percentage of tenders with one offer (bidder) at the regional level. Analogously, the regions are classified into 4 groups: dark green (1.8%–15%), light green (15%–21%), light red (21%–30.3%) and dark red (30.3%–100%). Switzerland is the only country where all regions with available data are in the best group (dark green). On the opposite side, all of Poland's regions are in the worst group (dark red). Almost all of the Czech Republic and Hungary's regions are in this group.

There are 401 German regions (called *Kreise*) which have notable differences in percentages. They are small, and their administrative processes are probably diverse. The number of tenders in each region is also likely small, which could affect the statistics.

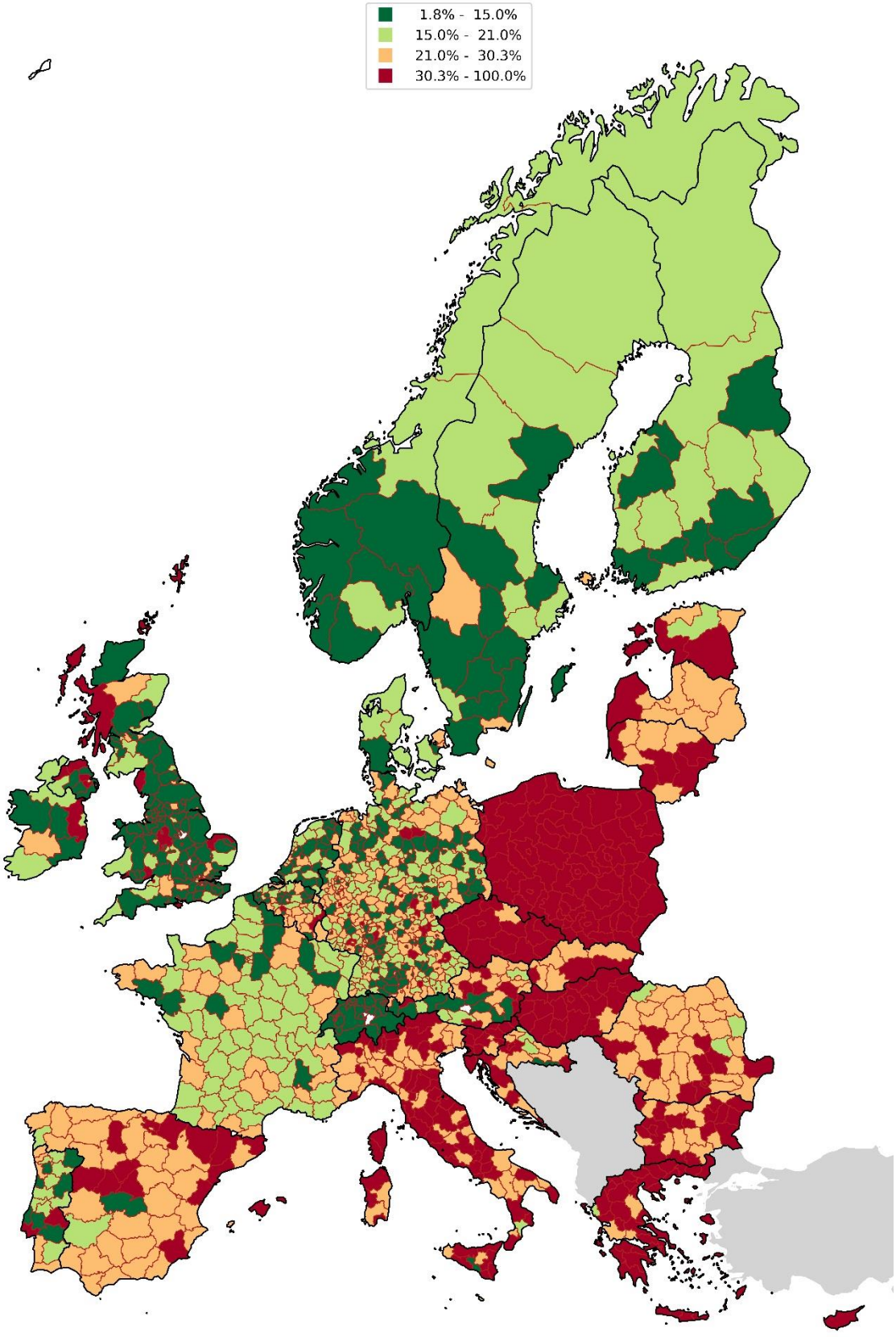

Figure 15. Tenders without competition by region for 2012–2022. Percentage of tenders with one offer (bidder), classified into 4 groups (low to high percentage: dark green, light green, light red and dark red) at the European regional (NUTS 3) level.

Figure 16 shows the foreign winners (percentage of tenders whose winners were not from the contracting authority's country) at the country level. It means, it is a direct cross-border (companies located in a foreign country) according to the classification of the report [7]. They have been classified into 4 groups:

- Dark red (0.6%–2%). 8 countries: Germany, France, Spain, Italy, Poland, Romania, Bulgaria and Greece.
- Light red (2%–3.7%). 8 countries: United Kingdom, Netherlands, Portugal, Czech Republic, Hungary, Slovenia, Croatia and Latvia.
- Light green (3.7%–5.7%). 6 countries: Sweden, Finland, Estonia, Lithuania, Slovakia and Switzerland.
- Dark green (5.7%–31.1%). 8 countries: Norway, Denmark, Austria, Belgium, Ireland, Luxembourg, Malta and Cyprus.

Of the 5 largest economies in Europe, only the United Kingdom had more than 2% of tenders won by foreigners. The reasons could be its proximity to Ireland and its highly internationalised economy. The 14 countries in the green groups are medium or small-sized (the largest is Belgium, with 11.6 million inhabitants in 2022). Country size seems to influence the percentage of foreign winners.

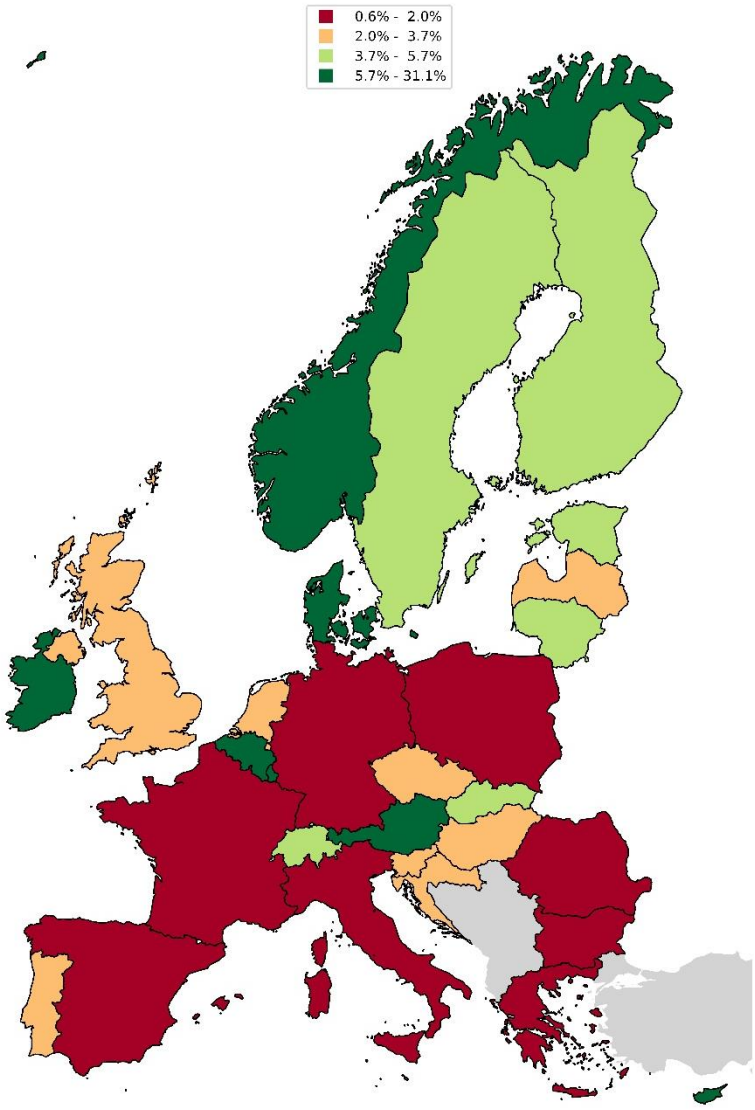

Figure 16. Foreign winners by country for 2012–2022. Percentage of tenders whose winners are not from the contracting authority's country. The countries are classified into 4 groups (low to high percentage: dark red, light red, light green and dark green).

Finally, Figure 17 shows the foreign winners at the regional level. Analogously, the regions are classified into 4 groups: dark red (0%–0.4%), light red (0.4%–1%), light green (1%–2%) and dark green (2%–33.6%). Note that these percentages are not the same as those on the previous map at the country level. 7 countries have all regions in the best group (dark green): Norway, Denmark, Ireland, Slovakia, Luxembourg, Malta and Cyprus. In general, the regions of Northern and Central Europe have higher percentages (except Germany and Poland). However, there are no trends related to the country's capital, coastal and border regions having a higher percentage.

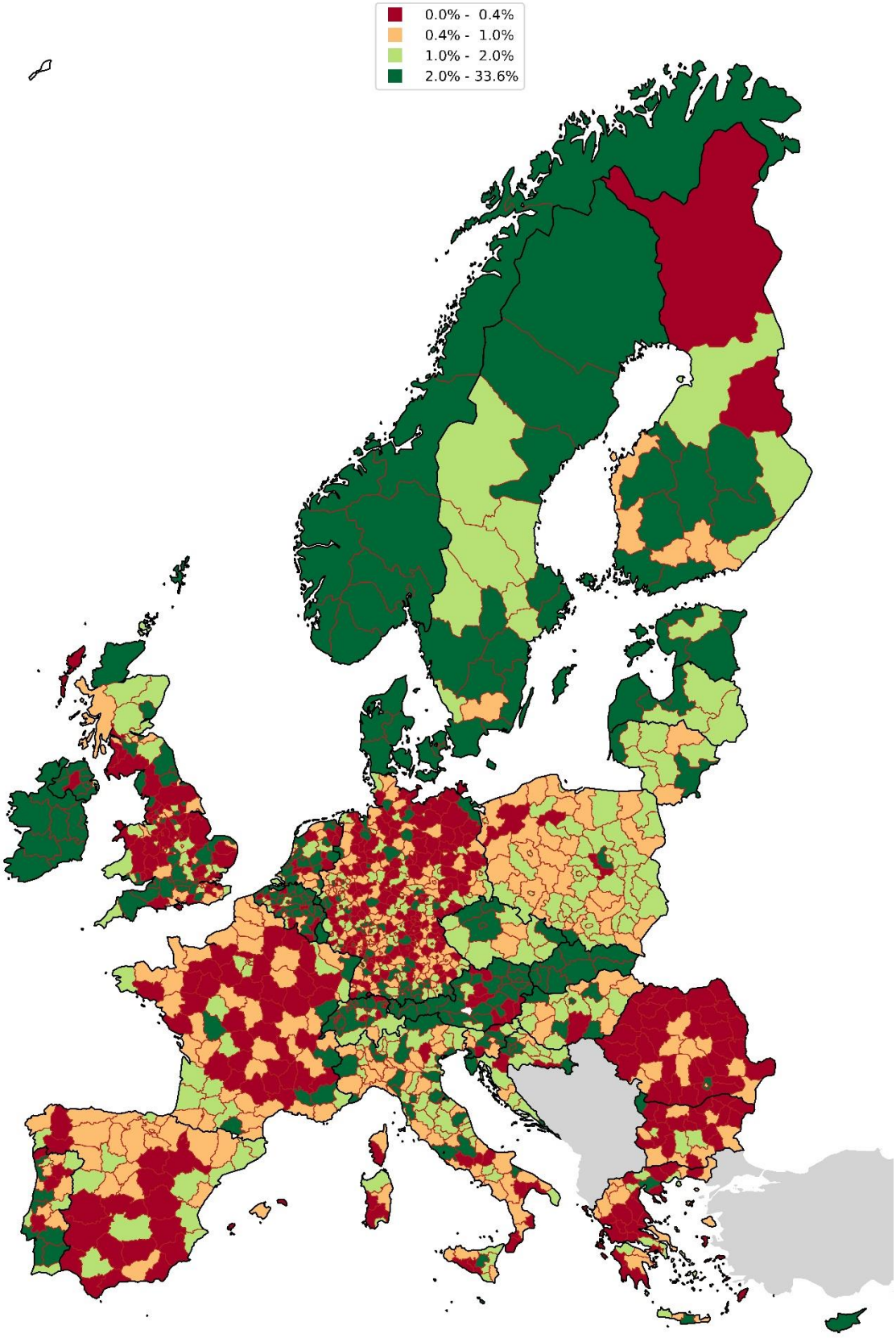

Figure 17. Foreign winners by region for 2012–2022. Percentage of tenders whose winners are not from the contracting authority's country, classified into 4 groups (low to high percentage: dark red, light red, light green and dark green) at the European regional (NUTS 3) level.

5. Conclusions and recommendations

This report has created several indicators to analyse European public procurement in 2012–2022 using TED data. The graphs and maps show that European public procurement is very diverse: there are significant differences between countries and also between regions. Public procurement has always had challenges and points for improvement [35]; it is a continuous process of legal, administrative and technical reforms.

This report shows countries and regions where they stand in relation to other areas. Key highlights from the report include:

- A data analysis based on TED information (2012–2022) for 30 European countries (EU 27 Member States, United Kingdom, Switzerland and Norway) equalling 3.1 million tenders and a total award price of 5,127 billion €.
- The most used type of procedure is open in economic terms: around 50% of the award price on average.
- Construction works (CPV 45) are clearly the most substantial tenders in economic terms: around 35% of the total award price on average.
- Approximately 140 thousand companies win tenders every year in 30 European countries.
- In recent years, the percentage of tenders won by SMEs was more or less 60%, but only 30% of the total award price went to SMEs.
- In half of the countries, the percentage of tenders won by SMEs is between 43.6% and 74.5%. Of the 5 largest European economies (Germany, France, Spain, Italy and the United Kingdom), Germany has the best percentage with 61%, more than 20% more than the next country (the United Kingdom). Respectively, in half of the regions, between 40.3% and 66.6% of tenders are won by SMEs.
- In half of the countries, the percentage of tenders with one offer (bidder) is between 18.2% and 34%. Respectively, in half of the regions, it is between 15% and 30.3%.
- In half of the countries, the percentage of tenders won by foreigners is between 2% and 5.7%. Of the 5 largest economies in Europe, only in the United Kingdom is it more than 2%. Respectively, in half of the regions, it is between 0.4% and 2%.
- There are only 7 countries with a percentage of tenders won by SMEs > 60% and a percentage of tenders with one offer < 30%: Switzerland, Malta, Latvia, Estonia, Sweden, Netherlands and Germany.

The report could have been expanded with more indicators, graphs or maps, disaggregating the information by sectors (CPV), types of contracts or procedures, price ranges, award criteria and analysis by year, among others. Future studies can focus on more particular or sectoral issues.

It is crucial to prioritise comprehensive research efforts at both the national and European levels to enhance data-based analysis in public procurement, given the current scarcity of analytical studies in the field. The first recommendation is for more extensive analytical research on public procurement. Despite the significance of this sector in economic and social development, there remains a notable gap in comprehensive studies and analyses focusing on the intricacies of public procurement and its impacts on the market. Undertaking in-depth analytical studies can shed light on critical areas, including the assessment of procurement practices, identification of potential loopholes, and evaluation of the effectiveness of existing policies and regulations.

Furthermore, there is a pressing demand to elevate the quantity and quality of the public procurement data used for analysis. The second recommendation is to ensure data accuracy,

completeness and consistency.[21] For this reason, the European Commission is developing the Public Procurement Data Space (PPDS[22]), which will connect European databases, including TED data on public procurement and national procurement data sets available in national portals. In many instances, the quality of public procurement data remains a subject of concern, as incomplete or erroneous information can significantly compromise the integrity and reliability of analytical insights. Implementing stringent data validation procedures, investing in robust data management systems, and fostering a culture of data integrity within public procurement entities are pivotal steps in bolstering data quality. Emphasising the importance of data quality assurance measures can substantially fortify the foundation for effective data-driven analysis, enabling policymakers and stakeholders to make informed decisions with more confidence and accuracy.

In addition, TED should expand the scope of the data fields collected within its open data to advance data-driven analysis in public procurement. Currently, TED's open data primarily focuses on essential procurement details, yet there is a notable absence of comprehensive information, such as the identities of other bidders involved in the tendering process. Including additional fields that capture pertinent data related to all participating bidders can facilitate comprehensive scrutiny of procurement activities. This can be instrumental in empowering competition authorities to conduct thorough analyses to detect potential anti-competitive practices, including collusive behaviour or oligopoly. Regulatory bodies can proactively monitor and investigate potential instances of market manipulation by enabling access to comprehensive data sets encompassing a broader spectrum of procurement information, fostering a more transparent and competitive procurement landscape.

[21] eForms is the notification standard for public procurement procedures in the EU. The use of eForms is mandatory as of 25/10/2023, meaning the Publications Office will only receive, process and publish eForms. More information is available at https://docs.ted.europa.eu/eforms/latest

[22] More information about PPDS is available at https://single-market-economy.ec.europa.eu/single-market/public-procurement/digital-procurement/public-procurement-data-space-ppds_en

References

[1] Public procurement contracts, European Parliament, Fact Sheets on the European Union. (2021). https://www.europarl.europa.eu/factsheets/en/sheet/34/public-procurement-contracts.

[2] OECD, Government at a Glance 2017, OECD, 2017. https://doi.org/10.1787/gov_glance-2017-en.

[3] European Commission, Public procurement, European Semester Thematic Factsheet. (2017) 1–12. https://ec.europa.eu/info/sites/info/files/file_import/european-semester_thematic-factsheet_public-procurement_en_0.pdf (accessed May 20, 2021).

[4] M. Fazekas, A. Abdou, Y. Kazmina, N. Regős, Development aid contracts database: World Bank, Inter-American Development Bank, and EuropeAid, Data in Brief. 42 (2022) 108121. https://doi.org/10.1016/j.dib.2022.108121.

[5] L. Jacques de Sousa, M.L. Simões, J. Poças Martins, L. Sanhudo, J. Moreira da Costa, Statistical Descriptive Analysis of Portuguese Public Procurement Data from 2015 to 2022, CivilEng. 4 (2023) 808–826. https://doi.org/10.3390/civileng4030045.

[6] M. Mendes, M. Fazekas, DIGIWHIST Recommendations for the Implementation of Open Public Procurement Data An Implementer's Guide, (2018) 1–16. https://opentender.eu/blog/2017-03-recommendations-for-implementation/.

[7] Prometeia SpA Bologna, Business Integration Partners S.p.A., Nova School of Business and Economics Economics for Policy Knowledge Center, I. Europäische Kommission Generaldirektion Binnenmarkt, Study on the measurement of cross-border penetration in the EU public procurement market, European Commission, 2021. https://doi.org/https://data.europa.eu/doi/10.2873/15626.

[8] M. Fazekas, B. TTth, The Effectiveness of the European Union in Safeguarding Competition in Public Procurement Markets, SSRN Electronic Journal. (2018) 1–47. https://doi.org/10.2139/ssrn.3101885.

[9] S.E. Weishaar, Cartels, competition and public procurement: Law and economics approaches to bid rigging, Cartels, Competition And Public Procurement: Law and Economics Approaches to Bid Rigging. (2013) 1–330. https://doi.org/10.4337/9780857936752.

[10] A. Rhode, Public Procurement in the European Union, Springer Fachmedien Wiesbaden, Wiesbaden, 2019. https://doi.org/10.1007/978-3-658-28073-4.

[11] The European Commission, Unleashing the Full Potential of SMEs, (2020) 3. https://doi.org/10.2775/296379.

[12] E. Commission, SME needs analysis in public procurement, 2021. https://doi.org/10.2873/86199.

[13] A. Sanchez-Graells, Procurement Corruption and Artificial Intelligence: Between the Potential of Enabling Data Architectures and the Constraints of Due Process Requirements, SSRN Electronic Journal. (2021) 1–17. https://doi.org/10.2139/ssrn.3952665.

[14] A. Young, S. Verhulst, The Global Impact of Open Data, 1st ed., O'Reilly Media, 2016.

[15] E. Huyer, L. van Knippenberg, The Economic Impact of Open Data Opportunities for value creation in Europe, 2020. https://doi.org/10.2830/63132.

[16] R. Nai, E. Sulis, R. Meo, Public Procurement Fraud Detection and Artificial Intelligence Techniques: a Literature Review, EKAW'22: Companion Proceedings of the 23rd International Conference on Knowledge Engineering and Knowledge Management. (2022). https://ceur-ws.org/Vol-3256/km4law4.pdf.

[17] Y. Torres Berru, V.F. López Batista, P. Torres-Carrión, M.G. Jimenez, Artificial Intelligence Techniques to Detect and Prevent Corruption in Procurement: A Systematic Literature Review, in: Springer (Ed.), Communications in Computer and Information Science, 2020: pp. 254–268. https://doi.org/10.1007/978-3-030-42520-3_21.

[18] T. Sun, L.J. Sales, Predicting Public Procurement Irregularity: An Application of Neural Networks, Journal of Emerging Technologies in Accounting. 15 (2018) 141–154. https://doi.org/10.2308/jeta-52086.

[19] Y. Torres-Berru, V.F. López Batista, Data Mining to Identify Anomalies in Public Procurement Rating Parameters, Electronics. 10 (2021) 2873. https://doi.org/10.3390/electronics10222873.

[20] N. Modrušan, K. Rabuzin, L. Mršic, Review of Public Procurement Fraud Detection Techniques Powered by Emerging Technologies, International Journal of Advanced Computer Science and Applications. 12 (2021) 575–583. https://doi.org/10.14569/IJACSA.2021.0120272.

[21] M. Huber, D. Imhof, Machine learning with screens for detecting bid-rigging cartels, International Journal of Industrial Organization. 65 (2019) 277–301. https://doi.org/10.1016/j.ijindorg.2019.04.002.

[22] H. Wallimann, D. Imhof, M. Huber, A Machine Learning Approach for Flagging Incomplete Bid-Rigging Cartels, Computational Economics. 0123456789 (2022). https://doi.org/10.1007/s10614-022-10315-w.

[23] Algorithms and Collusion: Competition Policy in the Digital Age, OECD. (2017). www.oecd.org/competition/algorithms-collusion-competition-policy-in-the-digital-age.htm.

[24] M.J. García Rodríguez, V. Rodríguez-Montequín, P. Ballesteros-Pérez, P.E.D. Love, R. Signor, Collusion detection in public procurement auctions with machine learning algorithms, Automation in Construction. 133 (2022) 104047. https://doi.org/10.1016/j.autcon.2021.104047.

[25] J.R. Kultima, Collusion Detection in Public Procurement Using Computational Methods, 2022. https://www.kfst.dk/media/dq2htrud/bid-viewer_56_seneste.pdf.

[26] P. Ballesteros-Pérez, M. Skitmore, E. Sanz-Ablanedo, P. Verhoeven, Forecasting the Number and Distribution of New Bidders for an Upcoming Construction Auction, Journal of Construction Engineering and Management. 145 (2019). https://doi.org/10.1061/(ASCE)CO.1943-7862.0001694.

[27] M.K. Gorgun, M. Kutlu, B.K. Onur Tas, Predicting The Number of Bidders in Public Procurement, in: 2020 5th International Conference on Computer Science and Engineering (UBMK), IEEE, 2020: pp. 360–365. https://doi.org/10.1109/UBMK50275.2020.9219404.

[28] J.S. Chou, C.W. Lin, A.D. Pham, J.Y. Shao, Optimized artificial intelligence models for predicting project award price, Automation in Construction. 54 (2015) 106–115. https://doi.org/10.1016/j.autcon.2015.02.006.

[29] M.J. García Rodríguez, V. Rodríguez Montequín, F. Ortega Fernández, J.M. Villanueva Balsera, Public Procurement Announcements in Spain: Regulations, Data Analysis, and Award Price Estimator Using Machine Learning, Complexity. 2019 (2019). https://doi.org/10.1155/2019/2360610.

[30] M.J. García Rodríguez, V. Rodríguez Montequín, A. Aranguen Ubierna, R. Santana Hermida, B. Sierra Araujo, A. Zelaia Jauregi, Award Price Estimator for Public Procurement Auctions Using Machine Learning Algorithms: Case Study with Tenders from Spain, Studies in Informatics and Control. 30 (2021) 67–76. https://doi.org/10.24846/v30i4y202106.

[31] M.J. García Rodríguez, V. Rodríguez Montequín, F. Ortega Fernández, J.M. Villanueva Balsera, Bidders Recommender for Public Procurement Auctions Using Machine Learning: Data Analysis, Algorithm, and Case Study with Tenders from Spain, Complexity. 2020 (2020) 1–20. https://doi.org/10.1155/2020/8858258.

[32] European Commission, Selected AI cases in the public sector, 2021. http://data.europa.eu/89h/7342ea15-fd4f-4184-9603-98bd87d8239a.

[33] World Bank, Disruptive technologies in public procurement, Washington, 2021. https://documents1.worldbank.org/curated/en/522181612428427520/pdf/Disruptive-Technologies-in-Public-Procurement.pdf.

[34] D.-G. for I.P. of the U. European Parliament, The Digital Single Market and the digitalisation of the public sector - GovTech and other innovations in public procurement, (2022) 64. https://doi.org/https://data.europa.eu/doi/10.2861/830794.

[35] A. Castelli, G. Piga, S. Saussier, T. Tátrai, The Challenges of Public Procurement Reforms, Routledge, 2020. https://doi.org/10.4324/9781003023470.

Appendix: extract, clean, filter and transform TED information (Python code)

Import libraries

```
import pandas as pd
pd.options.display.max_columns=100
import numpy as np
import glob
import os
from fuzzywuzzy import fuzz
import swifter
from tqdm.notebook import tqdm
tqdm.pandas()
import matplotlib.pyplot as plt
from matplotlib.ticker import PercentFormatter, FormatStrFormatter, StrMethodFormatter
from matplotlib_inline.backend_inline import set_matplotlib_formats
set_matplotlib_formats('svg')
import seaborn as sns
import re as re
import geopandas as gpd
from shapely.geometry import Point, Polygon, MultiPolygon
from functools import partial
from geopy.geocoders import Nominatim
import geopy
import geoplot as gplt
import ssl
from statistics import mode
from functools import reduce
```

Global variables

```
# Path TED_Contract_award_notices .csv files.
path_TED = './Data_TED/'
# Generate .pkl file from all TED_Contract_award_notices .csv files. If False, .pkl file has been creat
ed
generate_pickle = False
# Name pickle file with all contract award notices
dataset_name = 'TED_Dataset_filtered.pkl'
# Generate .pkl file of unique winning companies. If False, .pkl file has been created
generate_df_win_companies = False
# Name pickle file with unique winning companies
df_win_companies = 'TED_Dataset_companies.pkl'
# Generate .pkl file of unique winning companies without NATIONALID. If False, .pkl file has been creat
ed
generate_df_win_companies_without = False
# Name pickle file with unique winning companies
df_win_companies_without = 'TED_Dataset_companies_without_nationalid.pkl'
# Real GDP per capita from EU member states (from Eurostat)
UE_GDP_filename = 'UE_Real_GDP_per_capita.xlsx'
# Maps with geolocation information (latitude, longitude, NUTS, etc.) from Eurostat:
# https://ec.europa.eu/eurostat/web/gisco/geodata/reference-data/administrative-units-statistical-units
/nuts#nuts21
NUTS_RG_01M_2021_3857 = 'NUTS_RG_01M_2021_3857.geojson'
NUTS_RG_10M_2021_3857 = 'NUTS_RG_10M_2021_3857.geojson'
# Path to save figures
path_figures = './Data_TED/figures/'
if not os.path.exists(path_figures): os.makedirs(path_figures)
# Save figures in path or not
save_figures = True
```

1. Collect all TED information to generate pickle file

```
if generate_pickle is True:
    csv_files = glob.glob(os.path.join(path_TED, '*.csv'))
    df = pd.DataFrame()
    # loop over the list of csv files
    for file in csv_files:
        # read the csv file
        df_file = pd.read_csv(file, low_memory=False)
        # print filename and shape
        print('Filename {}: {} rows and {} columns'.format(file.split('\\')[-1], df_file.shape[0], df_f
ile.shape[1]))
```

```
        # concat dataframes
        df = pd.concat([df, df_file], sort=False)
```

1.1 Import and selection columns' dataframe from pickle file

```
if generate_pickle is True:
    selected_columns = ['ID_NOTICE_CAN',       # Unique identifier of the contract award notice (CAN)
                        'TED_NOTICE_URL',      # Webpage of the notice on the TED website. Note that TE
D hosts notices only for five years after publication, so for notices older than that the link will not
work.
                        'YEAR',                # Year of publication of the notice
                        'ID_TYPE',             # Standard form number, see the relevant TED webpage: ht
tps://simap.ted.europa.eu/web/simap/standard-forms-for-public-procurement
                        'DT_DISPATCH',         # The date when the buyer dispatched (sent) the notice f
or publication to TED
                        'XSD_VERSION',         # Version of the XML schema definition used by the Publi
cations Office. Higher versions mean better average quality of data.
                        'CANCELLED',           # 1 = this notice was later cancelled
                        'CAE_NAME',            # Official name of Contracting Authority Entity (CAE)
                        'CAE_NATIONALID',      # CAE's national registration number
                        'CAE_TOWN',            # CAE's town
                        'CAE_POSTAL_CODE',     # CAE's postal code
                        'ISO_COUNTRY_CODE',    # "Country" for the first listed authority
                        'B_MULTIPLE_COUNTRY',  # There are contracting authorities from at least 2 diff
erent countries
                        'ISO_COUNTRY_CODE_ALL', # If the variable above is yes, then this variable conta
ins the list of all countries.
                        'CAE_TYPE',            # Type of contracting authority: 1 (Ministry or any othe
r national authority), 3 (Regional or local authority), 4 (Utilities sectors), 5 (EU institution/agency
), 5A (Other international organisation), 6 (Body governed by public law), 8 (Other), N (National Agenc
y), R (Regional or local Agency) or Z (Not specified)
                        'EU_INST_CODE',        # EU institution (or type of EU institution)
                        'MAIN_ACTIVITY',       # COFOG (Classification of the Functions of Government)
divisions correspond only to the classical directive.
                        'TYPE_OF_CONTRACT',    # Type of contract. The values are W (works), U (supplie
s) or S (services)
                        'TAL_LOCATION_NUTS',   # The Nomenclature of Territorial Units for Statistics (
NUTS) code placement of the "Main site or location of work, place of delivery or of performance"
                        'B_FRA_AGREEMENT',     # The notice involves the establishment of a framework a
greement
                        'B_FRA_CONTRACT',      # This notice is probably about specific contracts withi
n a framework agreement.
                        'CPV',                 # The main Common Procurement Vocabulary code of the mai
n object of the contract
                        'ID_LOT',              # Lot identifier
                        'LOTS_NUMBER',         # The number of lots for a given CAN. This value is base
d on the number of unique "Lot No" values
                        'VALUE_EURO',          # CAN value, in EUR, without VAT. If the value was not p
resent, the lowest bid is used instead
                        'VALUE_EURO_FIN_1',    # CAN value, in EUR, without VAT. If a value variable is
missing, this variable looks for it in all other fields from which it could be taken
                        'VALUE_EURO_FIN_2',    # Generally the same value as VALUE_EURO_FIN_1, but - if
available - overwritten by human-made estimates of values for large value contracts which seemed to be
incorrect
                        'TOP_TYPE',            # Type of procedure. The values are AWP (award without p
rior publication of a contract notice), COD (competitive dialogue), NOC/NOP (negotiated without a call
for competition), NIC/NIP "negotiated with a call for competition), OPE (open), RES (restricted) or INP
(innovative partnership)
                        'CRIT_CODE',           # Award criteria. The values are L (lowest price) or M (
most economically advantageous tender)
                        'CRIT_PRICE_WEIGHT',   # Weight given to price
                        'CRIT_CRITERIA',       # Information on award criteria.
                        'CRIT_WEIGHTS',        # Information on award criteria weighing.
                        'NUMBER_AWARDS',       # The number of CAs for a given CAN
                        'ID_AWARD',            # Unique contract award identifier. This is the identifi
er for all variables at the contract award level
                        'ID_LOT_AWARDED',      # Lot No, an identifier of a lot within this contract aw
ard
                        'INFO_ON_NON_AWARD',   # If the variable is empty, then a contract was awarded
                        'B_AWARDED_TO_A_GROUP', #xThe contract has been awarded to a group of economic o
perators. Currently, only information about the economic operator listed first is available in the data
                        'WIN_NAME',            # Official name of the winner
                        'WIN_NATIONALID',      # National registration number of the winner. E.g. a VAT
number of a business registry number
                        'WIN_TOWN',            # Winner's town
```

```
                    'WIN_POSTAL_CODE',          # Winner's postal code
                    'WIN_COUNTRY_CODE',         # Winner's country
                    'B_CONTRACTOR_SME',         # The contractor is an SME
                    'TITLE',                    # Title of the contract award notice
                    'NUMBER_OFFERS',            # Number of tenders received
                    'NUMBER_TENDERS_SME',       # Number of tenders received from SMEs
                    'NUMBER_TENDERS_OTHER_EU', # Number of tenders received from tenderers from other E
U Member States
                    'NUMBER_TENDERS_NON_EU',    # Number of tenders received from tenderers from non-EU
countries
                    'AWARD_EST_VALUE_EURO',     # Estimated CA value, in EUR, without VAT
                    'AWARD_VALUE_EURO',         # Total final CA value, in EUR, without VAT. If the valu
e was not present, the lowest bid is included
                    'AWARD_VALUE_EURO_FIN_1',  # CA value, in EUR, without VAT. If the value variable i
s missing, this variable looks for it in all other fields from which it could be taken
                    'DT_AWARD']                 # Date of contract award
    df = df.filter(selected_columns)
```

1.2 Delete tenders without enough information and duplicates

```
if generate_pickle is True:
    # Delete rows if the fields WIN_NAME and WIN_NATIONALID are empty (NaN)
    df.dropna(subset=['WIN_NAME', 'WIN_NATIONALID'], how='all', inplace=True)
    # Delete rows if the fields VALUE_EURO, VALUE_EURO_FIN_1 and VALUE_EURO_FIN_2 are empty
    df.dropna(subset=['VALUE_EURO', 'VALUE_EURO_FIN_1', 'VALUE_EURO_FIN_2'], how='all', inplace=True)
    # Delete rows if the fields AWARD_EST_VALUE_EURO, AWARD_VALUE_EURO and AWARD_VALUE_EURO_FIN_1 are e
mpty
    df.dropna(subset=['AWARD_EST_VALUE_EURO', 'AWARD_VALUE_EURO', 'AWARD_VALUE_EURO_FIN_1'], how='all',
inplace=True)
    # Delete rows if the fields ISO_COUNTRY_CODE, B_MULTIPLE_COUNTRY and ISO_COUNTRY_CODE_ALL are empty
    df.dropna(subset=['ISO_COUNTRY_CODE', 'B_MULTIPLE_COUNTRY', 'ISO_COUNTRY_CODE_ALL'], how='all', inp
lace=True)
    # Delete rows if the field CANCELLED is not 1
    df = df[df['CANCELLED'] != 1]
    # Delete rows if the field INFO_ON_NON_AWARD is 'PROCUREMENT_UNSUCCESSFUL' or 'PROCUREMENT_DISCONTI
NUED'
    df = df[(df['INFO_ON_NON_AWARD'] != 'PROCUREMENT_UNSUCCESSFUL') & (df['INFO_ON_NON_AWARD'] != 'PROC
UREMENT_DISCONTINUED')]
    # Delete duplicates
    df.drop_duplicates(inplace=True)
    df.drop_duplicates(subset=['ID_NOTICE_CAN', 'CAE_NAME', 'ID_LOT', 'LOTS_NUMBER', 'ID_AWARD', 'ID_LO
T_AWARDED', 'WIN_NAME'], inplace=True)
    # Reset index
    df.reset_index(drop=True, inplace=True)
```

1.3 Transform data and delete outliers

```
if generate_pickle is True:
    # Delete outliers values in prices
    maximum_price = 20000000000
    minimum_price = 10000
    price_columns = ['VALUE_EURO', 'VALUE_EURO_FIN_1', 'VALUE_EURO_FIN_2', 'AWARD_EST_VALUE_EURO', 'AWA
RD_VALUE_EURO', 'AWARD_VALUE_EURO_FIN_1']
    for col in price_columns:
        df = df[((df[col] < maximum_price) & (df[col] > minimum_price)) | (df[col].isna())]
    # Delete award prices not coherent
    df['ID_CAN_SUM_VALUE'] = df.groupby(['ID_NOTICE_CAN'])['AWARD_VALUE_EURO'].transform('sum')
    security_margin = 1.00
    df = df[(df['VALUE_EURO']*security_margin >= df['ID_CAN_SUM_VALUE']) |
            (df['VALUE_EURO_FIN_1']*security_margin >= df['ID_CAN_SUM_VALUE']) |
            (df['VALUE_EURO_FIN_2']*security_margin >= df['ID_CAN_SUM_VALUE'])]
    df.drop(['ID_CAN_SUM_VALUE'], axis=1, inplace=True)
    # Delete outliers values in number offers
    maximum_offers = 940
    num_offers_columns = ['NUMBER_OFFERS', 'NUMBER_TENDERS_SME', 'NUMBER_TENDERS_OTHER_EU', 'NUMBER_TEN
DERS_NON_EU']
    for col in num_offers_columns:
        df[col] = df[col].apply(lambda x: x if x < maximum_offers and x > 0 else np.nan)
    # Dates. Convert to timestamp format
    df['DT_DISPATCH'] = pd.to_datetime(df['DT_DISPATCH'], errors='coerce', format='%d-%b-%y')
    df['DT_AWARD'] = pd.to_datetime(df['DT_AWARD'], errors='coerce', format='%d-%b-%y')
    # Delete Framework Contracts. A CAN is considered a framework contract if B_FRA_AGREEMENT or B_FRA_
CONTRACT is 'Y'
    # df = df[(df['B_FRA_AGREEMENT'] != 'Y') or (df['B_FRA_CONTRACT'] != 'Y')]
```

1.4 Save the pkl file

```
if generate_pickle is True:
    # Delete columns which are  more useful any more
    #df.drop(['CANCELLED', 'INFO_ON_NON_AWARD', 'B_FRA_AGREEMENT', 'B_FRA_CONTRACT'], axis=1, inplace=T
rue)
    # Save pickle
    df.to_pickle(path_TED + dataset_name, compression='infer', protocol=5)
```

2. Clean and transform TED dataset

```
df = pd.read_pickle(os.path.join(path_TED, dataset_name))
df.reset_index(drop=True, inplace=True)
```

2.1 Cleaning information of contractors (name winner and national ID) and contracting authorities

```
def clean_name(value, town=False):
    '''Clean and homogenize a name of a contracting authority or company'''

    if type(value) is str:
        # Everything is converted to uppercase
        value = value.upper()
        # Split values by '---'
        list_values = value.split('---')
        for i, value in enumerate(list_values):
            # Delete spaces on left and on right
            value = value.rstrip()
            value = value.lstrip()
            # If it is a town, the hyphen is preserved and we remove dots, commas, colons, quotation ma
rks, parentheses, |, etc.
            if town is True:
                list_char = ['+', '.', ';', ',', ':', '*', '%', '/', '?', '¿', '_', '$', '€', '#', '!',
'¡', ':', '"', "'",
                             '(', ')', '|', '[', ']', '<', '>', '{', '}', '·', 'ª', 'º', '\\', '@', '`', '´'
, '«', '»', '‘', '’',
                             '“', '”', '„', '§', '•', '‴', '‴', '´', '^', '¨', '°', '=', '\t']
                for char in list_char:
                    value = value.replace(char, '-')
            # If not, delete hyphens, periods, commas, colons, quotes, parentheses, |, etc.
            else:
                list_char = ['-', '–', '−', '+', '.', ';', ',', ':', '*', '%', '/', '?', '¿', '_', '$',
'€', '#', '!', '¡', ':', '"', "'",
                             '(', ')', '|', '[', ']', '<', '>', '{', '}', '·', 'ª', 'º', '\\', '@', '`',
'´', '«', '»', '‘', '’',
                             '“', '”', '„', '§', '•', '‴', '‴', '´', '^', '¨', '°', '=', '\t']
                for char in list_char:
                    value = value.replace(char, '')
            # Delete multiple spaces
            value = re.sub(' +', ' ', value)
            # Delete accentuation
            list_char_A = ['Á', 'À','Â', 'Ä']
            for char in list_char_A:
                value = value.replace(char, 'A')
            list_char_E = ['É', 'È', 'Ê', 'Ë']
            for char in list_char_E:
                value = value.replace(char, 'E')
            list_char_I = ['Í', 'Ì', 'Î', 'Ï']
            for char in list_char_I:
                value = value.replace(char, 'I')
            list_char_O = ['Ó', 'Ò', 'Ô', 'Ö']
            for char in list_char_O:
                value = value.replace(char, 'O')
            list_char_U = ['Ú', 'Ù', 'Û', 'Ü']
            for char in list_char_U:
                value = value.replace(char, 'U')
            # Delete spaces in the abbreviations of commercial companies from Spain
            if value[-4:] == ' S A': value = value[:-4] + ' SA'
            if value[-17:] == ' SOCIEDAD ANONIMA': value = value[:-17] + ' SA'
            if value[-4:] == ' S C': value = value[:-4] + ' SC'
            if value[-4:] == ' S L': value = value[:-4] + ' SL'
            if value[-18:] == ' SOCIEDAD LIMITADA': value = value[:-18] + ' SL'
            if value[-6:] == ' S R L': value = value[:-6] + ' SRL'
            if value[-6:] == ' S A U': value = value[:-6] + ' SAU'
```

```
            if value[-6:] == ' S L P': value = value[:-6] + ' SLP'
            if value[-6:] == ' S L U': value = value[:-6] + ' SLU'
            # Germany
            if value[-4:] == ' A G': value = value[:-4] + ' AG'
            if value[-8:] == ' G M B H': value = value[:-8] + ' GMBH'
            if value[-4:] == ' K G': value = value[:-4] + ' KG'
            if value[-6:] == ' O H G': value = value[:-6] + ' OHG'
            if value[-6:] == ' G B R': value = value[:-6] + ' GBR'
            # France
            if value[-6:] == ' S A S': value = value[:-6] + ' SAS'
            if value[-6:] == ' S C P': value = value[:-6] + ' SCP'
            if value[-8:] == ' S A R L': value = value[:-8] + ' SARL'
            # UK
            if value[-6:] == ' P L C': value = value[:-6] + ' PLC'
            if value[-6:] == ' L T D': value = value[:-6] + ' LTD'
            if value[-8:] == ' LIMITED': value = value[:-8] + ' LTD'
            if value[-10:] == '  U N L T D': value = value[:-10] + ' UNLTD'
            if value[-6:] == ' L L P': value = value[:-6] + ' LLP'
            # Poland
            if value[-8:] == ' SP Z OO': value = value[:-8] + ' SPZOO'
            if value[-9:] == ' SP Z O O': value = value[:-9] + ' SPZOO'
            if value[-7:] == ' SPZ OO': value = value[:-7] + ' SPZOO'
            if value[-10:] == ' S P Z O O': value = value[:-10] + ' SPZOO'
            if value[-8:] == ' S P ZOZ': value = value[:-8] + ' SPZOZ'
            if value[-5:] == ' SP J': value = value[:-5] + ' SPJ'
            if value[-6:] == ' S P J': value = value[:-6] + ' SPJ'
            # Slovakia
            if value[-6:] == ' S R O': value = value[:-6] + ' SRO'
            if value[-5:] == ' S RO': value = value[:-5] + ' SRO'
            # Slovenia
            if value[-4:] == ' D D': value = value[:-4] + ' DD'
            # Portugal
            if value[-6:] == ' E P E': value = value[:-6] + ' EPE'
            # Denmark
            if value[-4:] == ' E K': value = value[:-4] + ' EK'

            # Concat values
            if i == 0:
                return_list_values = value
            else:
                return_list_values = return_list_values + '---' + value
    return return_list_values

def clean_VAT(value, company):
    '''Clean and homogenize the national ID (VAT) of contracting authority or company'''

    if type(value) is str:
        # Everything is converted to uppercase
        value = value.upper()
        # Split values by '---'
        list_values = value.split('---')
        for i, value in enumerate(list_values):
            # Delete spaces on left and on right
            value = value.rstrip()
            value = value.lstrip()
            # Delete spaces, hyphens, periods, commas, colons, quotes, parentheses, |, etc.
            list_char = [' ', '-', '—', '–', '+', '.', ';', ',', ':', '*', '%', '/', '?', '¿', '_', '$'
, '€', '#', '!', '¡', ':', '"', "'",
                         '(', ')', '|', '[', ']', '<', '>', '{', '}', '·', 'ª', 'º', '\\', '@', '`', '´'
, '«', '»', '‘', '’',
                         '“', '”', '„', '§', '&', '\t']
            for char in list_char:
                value = value.replace(char, '')
            # Delete CIF, VAT, NIF etc.
            list_ID = ['VAT', 'ID', 'REGISTERED', # General
                       'REGISTER', 'REGISTRY',
                       'NUMBER', 'TEMP',
                       'SIRET',                # France
                       'CIF', 'NIF', 'UTE',
                       'ROMANIA',              # Spain, Portugal, Romania
                       'UID',                  # Austria
                       'TVA', 'BTW', 'MWST',   # Belgium, France, Luxembourg
                       'ДДС', 'НОМЕР',         # Bulgaria
                       'PDV', 'OIB',           # Croatia
```

```
                    'ΦΠΑ',                  # Cyprus
                    'DIČ',                  # Czech Republic
                    'CVR',                  # Denmark
                    'KMKR',                 # Estonia
                    'ALV', 'MOMSNU', 'MMER',# Finland
                    'UST', 'NR',            # Germany
                    'ΑΦΜ',                  # Greece
                    'ANUM',                 # Hungary
                    'CBL',                  # Ireland
                    'PIVA',                 # Italy
                    'PVN',                  # Latvia
                    'PVM', 'KODAS',         # Lithuania
                    'NIP', 'REGON',         # Poland
                    'IČ', 'DPH', 'IČ DPH',  # Slovakia
                    'DDV',                  # Slovenia
                    'MOMSNR',               # Sweden
                    'BTW', 'BTWNR']         # The Netherlands
            for char in list_ID:
                value = value.replace(char, '')
            # Delete ES because it is already Spanish CIF
            value = value.replace('ES', '')
            if company is True:
                # Delete IDs with abnormal sizes (too small or large)
                if len(value) < 8 or len(value) > 20:
                    value = ''
                # Delete zeros
                try:
                    if int(value) <= 1:
                        value = ''
                except:
                    pass
                # Delete VAT with only charachers
                if value.isalpha() is True:
                    value = ''

            if i == 0:
                return_list_values = value
            else:
                return_list_values = return_list_values + '---' + value
        return return_list_values

print('Number of contractors by name (WIN_NAME) before cleaning: {}. And NaN: {}'.format(df['WIN_NAME']
.nunique(), df['WIN_NAME'].isna().sum()))
print('Number of contractors by ID (WIN_NATIONALID) before cleaning: {}. And NaN: {}'.format(df['WIN_NA
TIONALID'].nunique(), df['WIN_NATIONALID'].isna().sum()))
df['WIN_NAMEC'] = df['WIN_NAME'].astype(str).apply(lambda x: clean_name(x, False))
df['WIN_TOWNC'] = df['WIN_TOWN'].astype(str).apply(lambda x: clean_name(x, True))
df['WIN_POSTAL_CODEC'] = df['WIN_POSTAL_CODE'].astype(str).apply(lambda x: clean_name(x, False))
df['WIN_NATIONALIDC'] = df['WIN_NATIONALID'].astype(str).apply(lambda x: clean_VAT(x, True))
print('Number of contractors by name (WIN_NAMEC) after cleaning: {}. And NaN: {}'.format(df['WIN_NAMEC'
].nunique(), df['WIN_NAMEC'].isna().sum()))
print('Number of contractors by ID (WIN_NATIONALIDC) after cleaning: {}. And NaN: {}'.format(df['WIN_NA
TIONALIDC'].nunique(), df['WIN_NATIONALIDC'].isna().sum()))
print('Rows and columns of the dataframe: {}'.format(df.shape))

def calculate_length(df):
    ''' Calculate length of elements in columns WIN_NAMEC, WIN_NATIONALIDC and WIN_COUNTRY_CODE'''

    columns = ['WIN_NAMEC', 'WIN_NATIONALIDC', 'WIN_COUNTRY_CODE']
    for col in columns:
        df[col+'_LEN'] = df[col].str.len()
        df[col+'_LEN'] = df[col+'_LEN'].fillna(0)
    df['WIN_NUMBER_OK'] = (df['WIN_NAMEC_LEN'] == df['WIN_NATIONALIDC_LEN']) & (df['WIN_NATIONALIDC_LEN
'] == df['WIN_COUNTRY_CODE_LEN'])
    return df

def assign_none_values(list):
    ''' Assign '' or 'NAN' values in list to None'''

    for i, c in enumerate(list):
        if c == '' or c == 'NAN': list[i] = None
    return list
```

```
columns = ['WIN_NAMEC', 'WIN_NATIONALIDC', 'WIN_COUNTRY_CODE', 'WIN_TOWNC', 'WIN_POSTAL_CODEC', 'B_CONT
RACTOR_SME']
for col in columns:
    # Convert to string
    df[col] = df[col].astype('string')
    # pd.NA values to None
    df[col] = df[col].apply(lambda x: None if x is pd.NA or x == '' or x == 'NAN' else x)
    # Split values (separator is '---')
    df[col] = df[col].str.split('---')
    # '' or 'NAN' values to None
    df[col] = df[col].apply(lambda list: assign_none_values(list) if list is not None and any(list) els
e None)
# Delete tenders without information about the winner (WIN_NAMEC and WIN_NATIONALIDC are None)
df.dropna(subset=['WIN_NAMEC', 'WIN_NATIONALIDC'], how='all', inplace=True)
# Calculate 1) Length of series and 2) By row, check number of companies, IDs and countries
df = calculate_length(df)
```

2.2 Generate a dataframe of unique contractors

```
def calculate_mode(x):
    ''' Calculate mode of series pandas '''

    if any(x):
        mode = x.mode(dropna=True)
        if mode[0] is not None: return mode[0]
        else: return None
    else: return None

def fuzz_similar_companies(row, df):
    ''' Calculate similar companies by name and nationalID (fuzz.ratio >= 160) '''

    if not df.empty:
        max_element_name = df['WIN_NAMEC'].apply(lambda x: fuzz.ratio(x, row['WIN_NAMEC'])).nlargest(1)
        ratio_nationalID = fuzz.ratio(df.loc[max_element_name.index[0]]['WIN_NATIONALIDC'] , row['WIN_N
ATIONALIDC'])
        if max_element_name.values[0] + ratio_nationalID >= 160:
            row['WIN_NAMEC_VALID'] = df.loc[max_element_name.index[0]]['WIN_NAMEC']
            row['WIN_NATIONALIDC_VALID'] = df.loc[max_element_name.index[0]]['WIN_NATIONALIDC']
    return None

def create_df_companies(df_aux):
    '''Create a dataframe which unique award companies, their ID and country'''

    df = df_aux.copy()
    columns_explode = ['WIN_NAMEC', 'WIN_NATIONALIDC', 'WIN_COUNTRY_CODE', 'WIN_TOWNC', 'WIN_POSTAL_COD
EC', 'B_CONTRACTOR_SME']
    for col in columns_explode:
        df[col] = df[col].apply(lambda x: [None] if x is None else x)
    # Delete values not correct by length in WIN_TOWNC and WIN_POSTAL_CODEC
    df['WIN_TOWNC'] = df.apply(lambda row: ([None] * int(row['WIN_NAMEC_LEN'])) if len(row['WIN_TOWNC']
) != row['WIN_NAMEC_LEN'] else row['WIN_TOWNC'], axis='columns')
    df['WIN_POSTAL_CODEC'] = df.apply(lambda row: ([None] * int(row['WIN_NAMEC_LEN'])) if len(row['WIN_
POSTAL_CODEC']) != row['WIN_NAMEC_LEN'] else row['WIN_POSTAL_CODEC'], axis='columns')
    df['B_CONTRACTOR_SME'] = df.apply(lambda row: ([None] * int(row['WIN_NAMEC_LEN'])) if len(row['B_CO
NTRACTOR_SME']) != row['WIN_NAMEC_LEN'] else row['B_CONTRACTOR_SME'], axis='columns')
    # Explode columns
    columns_explode = ['WIN_NAMEC', 'WIN_NATIONALIDC', 'WIN_COUNTRY_CODE', 'WIN_TOWNC', 'WIN_POSTAL_COD
EC', 'B_CONTRACTOR_SME']
    df = df.explode(columns_explode)[columns_explode]
    df.reset_index(drop=True, inplace=True)
     # Assign the most frequent WIN_NATIONALIDC when the WIN_NAMEC is the same for each country and yea
r
    df['WIN_NATIONALIDC'] = df.groupby(['WIN_NAMEC', 'WIN_COUNTRY_CODE'])['WIN_NATIONALIDC'].progress_t
ransform(calculate_mode)
    # And vice verda, Assign the most frequent WIN_NAMEC when the WIN_NATIONALIDC is the same for each
country
    df['WIN_NAMEC'] = df.groupby(['WIN_NATIONALIDC', 'WIN_COUNTRY_CODE'])['WIN_NAMEC'].progress_transfo
rm(calculate_mode)
    # Assign the most frequent WIN_TOWNC,  WIN_POSTAL_CODEC and B_CONTRACTOR_SME for the same WIN_NAMEC
and WIN_COUNTRY_CODE combination
    df['WIN_TOWNC'] = df.groupby(['WIN_NAMEC', 'WIN_COUNTRY_CODE'])['WIN_TOWNC'].progress_transform(cal
culate_mode)
    df['WIN_POSTAL_CODEC'] = df.groupby(['WIN_NAMEC', 'WIN_COUNTRY_CODE'])['WIN_POSTAL_CODEC'].progress
_transform(calculate_mode)
```

```
    df['B_CONTRACTOR_SME'] = df.groupby(['WIN_NAMEC', 'WIN_COUNTRY_CODE'])['B_CONTRACTOR_SME'].progress
_transform(calculate_mode)
    # Count tenders with same WIN_COUNTRY_CODE, WIN_NAMEC and WIN_NATIONALIDC
    df['WIN_COUNT_TENDERS'] = df.groupby(['WIN_COUNTRY_CODE', 'WIN_NAMEC', 'WIN_NATIONALIDC'])['WIN_NAM
EC'].transform('size')
    # Replace NaN by None
    df.replace({np.nan: None}, inplace=True)
    # Drop duplicates
    df.drop_duplicates(inplace=True)
    df.dropna(subset=['WIN_NAMEC', 'WIN_NATIONALIDC'], how='all', inplace=True)
    # Detect similar companies in the own dataframe generated
    df['WIN_NAMEC_VALID'] = None
    df['WIN_NATIONALIDC_VALID'] = None
    df.progress_apply(lambda row: fuzz_similar_companies(row, df[df['WIN_COUNTRY_CODE'].isin([row['WIN_
COUNTRY_CODE']]) &
                                                                 df['WIN_COUNT_TENDERS'].gt(row['WIN_CO
UNT_TENDERS'] * 3) &
                                                                 (df['WIN_TOWNC'].isin([row['WIN_TOWNC'
]]) | df['WIN_POSTAL_CODEC'].isin([row['WIN_POSTAL_CODEC']]))]), axis='columns')
    # Reset index
    df.reset_index(drop=True, inplace=True)
    return df

if generate_df_win_companies is True:
    df_co = create_df_companies(df[df['WIN_NUMBER_OK'] == True])
```

2.2.1 Add geolocation (latitude and longitude) and NUTS to dataframe of unique contractors

```
def get_lat_lon(row, geocode):
    ''' Get latitude and longitude from address'''

    query = row['WIN_COUNTRY_CODE']
    if row['WIN_TOWNC'] is not None: query = query + ', ' + row['WIN_TOWNC']
    if row['WIN_POSTAL_CODEC'] is not None: query = query + ', ' + row['WIN_POSTAL_CODEC']
    if row['WIN_TOWNC'] is not None or row['WIN_POSTAL_CODEC'] is not None:
        try:
            location = geocode(query)
            if location is not None: return (location.latitude, location.longitude)
            else:
                if row['WIN_TOWNC'] is not None: location = geocode(row['WIN_COUNTRY_CODE'] + ', ' + ro
w['WIN_TOWNC'])
                if location is not None: return (location.latitude, location.longitude)
        except Exception as e:
            print(e)
    return (None, None)

def get_nuts_from_point(point, gdf_map):
    '''Get NUTS ID from point'''

    nuts = gdf_map[gdf_map.contains(point)]
    if not nuts.empty:
        return nuts['NUTS_ID'].values[0]
    else: return None

if generate_df_win_companies is True:
    # Disable SSL to use the geo-service Nominatim
    ctx = ssl.create_default_context()
    ctx.check_hostname = False
    ctx.verify_mode = ssl.CERT_NONE
    # Create the service Nominatim
    geolocator = Nominatim(user_agent='myapp', timeout=5, ssl_context=ctx, adapter_factory=geopy.adapte
rs.URLLibAdapter)
    geocode = partial(geolocator.geocode, language='en')
    # Drop duplicates
    df_aux = df_co.drop_duplicates(subset=['WIN_COUNTRY_CODE', 'WIN_TOWNC', 'WIN_POSTAL_CODEC'])
    # Get latitude and longitude of each row (WIN_COUNTRY_CODE, WIN_TOWNC and WIN_POSTAL_CODEC)
    df_aux['LAT_LON'] = df_aux.progress_apply(lambda row: get_lat_lon(row, geocode), axis=1)
    # Merge df_companies with df_aux by WIN_COUNTRY_CODE, WIN_TOWNC and WIN_POSTAL_CODEC
    df_co = df_co.merge(df_aux[['WIN_COUNTRY_CODE', 'WIN_TOWNC', 'WIN_POSTAL_CODEC', 'LAT_LON']], how='
left', on=['WIN_COUNTRY_CODE', 'WIN_TOWNC', 'WIN_POSTAL_CODEC'])

if generate_df_win_companies is True:
    # Create geopandas from df_co
    df_nuts = pd.DataFrame([])
    df_nuts['LAT_LON'] = df_co.drop_duplicates(subset=['LAT_LON'])['LAT_LON']
```

```
    df_nuts['POINT'] = df_co.drop_duplicates(subset=['LAT_LON'])['LAT_LON'].apply(lambda x: Point(x[1], x[0]) if len(x) > 0 and any(x) else None)
    gdf_df_nuts = gpd.GeoDataFrame(df_nuts, geometry='POINT', crs='EPSG:4326').to_crs('EPSG:3857')
    # Get map with NUTS (level 3) information
    map = gpd.read_file(os.path.join(path_TED, NUTS_RG_01M_2021_3857), driver='GeoJSON')
    mask_EU = (-1.5*10**6, 0.4*10**7, 4*10**6, 1.2*10**7)
    gdf_map = gpd.GeoDataFrame(map[map['LEVL_CODE'].isin([3])], geometry='geometry', crs='EPSG:3857').clip(mask_EU)
    # Calculate NUTS (level 3) from point
    gdf_df_nuts['NUTS_ID'] = gdf_df_nuts['POINT'].progress_apply(lambda x: get_nuts_from_point(x, gdf_map) if x is not None else None)
    # Merge df_co and gdf_df_nuts
    df_co = df_co.merge(gdf_df_nuts[['NUTS_ID', 'LAT_LON']], how='left', on=['LAT_LON'])

if generate_df_win_companies is True:
    # Save pickle
    df_co.to_pickle(path_TED + df_win_companies, compression='infer', protocol=5)
else:
    # Load pickle
    df_co = pd.read_pickle(os.path.join(path_TED, df_win_companies))
```

2.3 Apply the dataframe of unique contractors to fill contractor's information for tenders with one winner

```
# Create tuples
columns = ['WIN_NAMEC', 'WIN_NATIONALIDC', 'WIN_COUNTRY_CODE', 'WIN_TOWNC', 'WIN_POSTAL_CODEC', 'B_CONTRACTOR_SME']
for col in columns:
    df[col] = df[col].apply(lambda x: tuple(x) if x is not None else None)

# 1. Merge df_companies with df by WIN_NAMEC and WIN_COUNTRY_CODE
df = df.merge(df_co.applymap(lambda x: tuple([x])), how='left', on=['WIN_NAMEC', 'WIN_COUNTRY_CODE'], suffixes=('', '_y'))
# Unify fields
df['WIN_NATIONALIDC'] = df['WIN_NATIONALIDC_y'].fillna(df['WIN_NATIONALIDC'])
df['WIN_TOWNC'] = df['WIN_TOWNC_y'].fillna(df['WIN_TOWNC'])
df['WIN_POSTAL_CODEC'] = df['WIN_POSTAL_CODEC_y'].fillna(df['WIN_POSTAL_CODEC'])
df['B_CONTRACTOR_SME'] = df['B_CONTRACTOR_SME_y'].fillna(df['B_CONTRACTOR_SME'])
# Delete temporal columns
df.drop(df.filter(regex='_y$').columns, axis=1, inplace=True)

# 2. Merge df_companies with df by WIN_NATIONALIDC and WIN_COUNTRY_CODE
df = df.merge(df_co.applymap(lambda x: tuple([x])), how='left', on=['WIN_NATIONALIDC', 'WIN_COUNTRY_CODE'], suffixes=('', '_y'))
# Unify fields
df['WIN_NAMEC'] = df['WIN_NAMEC_y'].fillna(df['WIN_NAMEC'])
df['WIN_TOWNC'] = df['WIN_TOWNC_y'].fillna(df['WIN_TOWNC'])
df['WIN_POSTAL_CODEC'] = df['WIN_POSTAL_CODEC_y'].fillna(df['WIN_POSTAL_CODEC'])
df['B_CONTRACTOR_SME'] = df['B_CONTRACTOR_SME_y'].fillna(df['B_CONTRACTOR_SME'])
df['NUTS_ID'] = df['NUTS_ID_y'].fillna(df['NUTS_ID'])
# Delete temporal columns
df.drop(df.filter(regex='_y$').columns, axis=1, inplace=True)

# Tuples to list
columns = ['WIN_NAMEC', 'WIN_NATIONALIDC', 'WIN_COUNTRY_CODE', 'WIN_TOWNC', 'WIN_POSTAL_CODEC', 'B_CONTRACTOR_SME']
for col in columns:
    df[col] = df[col].apply(lambda x: list(x) if x is not None else None)
# Update length columns and WIN_NUMBER_OK
df = calculate_length(df)

# If there is neither WIN_NATIONALIDC nor WIN_COUNTRY_CODE but WIN_NAMEC does match for the country ISO_COUNTRY_CODE in df_co,
# we assume that it is that company (a company from the same country of the contracting authority) to fill in its data (WIN_NATIONALIDC, WIN_COUNTRY_CODE, WIN_TOWNC and WIN_POSTAL_CODEC)
df_aux = df[df['WIN_NATIONALIDC'].isna() & df['WIN_COUNTRY_CODE'].isna()].copy()
df_aux['WIN_COUNTRY_CODE'] = df_aux['ISO_COUNTRY_CODE'].apply(lambda x: [x])

# Create tuples
columns = ['WIN_NAMEC', 'WIN_NATIONALIDC', 'WIN_COUNTRY_CODE', 'WIN_TOWNC', 'WIN_POSTAL_CODEC', 'B_CONTRACTOR_SME']
for col in columns:
    df_aux[col] = df_aux[col].apply(lambda x: tuple(x) if x is not None else None)
```

```
# 3. Merge
df_aux = df_aux.reset_index().merge(df_co.applymap(lambda x: tuple([x])), how='left', on=['WIN_NAMEC',
'WIN_COUNTRY_CODE'], suffixes=('', '_y')).set_index('index')
# Unify fields
df_aux['WIN_NATIONALIDC'] = df_aux['WIN_NATIONALIDC_y']
df_aux['WIN_TOWNC'] = df_aux['WIN_TOWNC_y'].fillna(df['WIN_TOWNC'])
df_aux['WIN_POSTAL_CODEC'] = df_aux['WIN_POSTAL_CODEC_y'].fillna(df['WIN_POSTAL_CODEC'])
df_aux['B_CONTRACTOR_SME'] = df_aux['B_CONTRACTOR_SME_y'].fillna(df['B_CONTRACTOR_SME'])
df_aux['NUTS_ID'] = df_aux['NUTS_ID_y'].fillna(df_aux['NUTS_ID'])
# Delete temporal columns
df_aux.drop(df_aux.filter(regex='_y$').columns, axis=1, inplace=True)
# Delete not filled
df_aux.dropna(subset=['WIN_NATIONALIDC'], inplace=True)

# Tuples to list
columns = ['WIN_NAMEC', 'WIN_NATIONALIDC', 'WIN_COUNTRY_CODE', 'WIN_TOWNC', 'WIN_POSTAL_CODEC', 'B_CONT
RACTOR_SME']
for col in columns:
    df_aux[col] = df_aux[col].apply(lambda x: list(x) if x is not None else None)
# Update length columns and WIN_NUMBER_OK
df_aux = calculate_length(df_aux)

# Update df with df_aux's information
df = df.reset_index().set_index('index')
df.update(df_aux)

# Create tuples
columns = ['WIN_NAMEC', 'WIN_NATIONALIDC', 'WIN_COUNTRY_CODE', 'WIN_TOWNC', 'WIN_POSTAL_CODEC', 'B_CONT
RACTOR_SME']
for col in columns:
    df[col] = df[col].apply(lambda x: tuple(x) if x is not None else None)

# 4. Merge df with df_co by WIN_NAME_VALID, WIN_NATIONALIDC_VALID and WIN_COUNTRY_CODE
df = df.merge(df_co.applymap(lambda x: tuple([x])), how='left', left_on=['WIN_NAMEC', 'WIN_NATIONALIDC'
, 'WIN_COUNTRY_CODE'], right_on=['WIN_NAMEC_VALID', 'WIN_NATIONALIDC_VALID', 'WIN_COUNTRY_CODE'], suffi
xes=('', '_y'))
# Unify all fields' adjudicator
df['WIN_NAMEC'] = df['WIN_NAMEC_VALID_y'].fillna(df['WIN_NAMEC'])
df['WIN_NATIONALIDC'] = df['WIN_NATIONALIDC_VALID_y'].fillna(df['WIN_NATIONALIDC'])
df['WIN_TOWNC'] = df['WIN_TOWNC_y'].fillna(df['WIN_TOWNC'])
df['WIN_POSTAL_CODEC'] = df['WIN_POSTAL_CODEC_y'].fillna(df['WIN_POSTAL_CODEC'])
df['B_CONTRACTOR_SME'] = df['B_CONTRACTOR_SME_y'].fillna(df['B_CONTRACTOR_SME'])
df['NUTS_ID'] = df['NUTS_ID_y'].fillna(df['NUTS_ID'])
# Delete df_co columns
df.drop(df.filter(regex='_y$').columns, axis=1, inplace=True)
df.drop(['WIN_COUNT_TENDERS', 'WIN_NAMEC_VALID', 'WIN_NATIONALIDC_VALID'], axis=1, inplace=True)

# Tuples to list
columns = ['WIN_NAMEC', 'WIN_NATIONALIDC', 'WIN_COUNTRY_CODE', 'WIN_TOWNC', 'WIN_POSTAL_CODEC', 'B_CONT
RACTOR_SME']
for col in columns:
    df[col] = df[col].apply(lambda x: list(x) if x is not None else None)
# Update length columns and WIN_NUMBER_OK
df = calculate_length(df)
```

2.4 Generate a dataframe of similar name contractors

```
def fuzz_name_company(row, df):
    ''' Calculate similar companies by name (fuzz.partial_ratio >= 95) '''

    if not df.empty:
        max_element = df['WIN_NAMEC'].apply(lambda x: fuzz.partial_ratio(x, row['WIN_NAMEC'])).nlargest
(1)
        if max_element.values[0] >= 95:
            row['WIN_NATIONALIDC'] = df_co.iloc[max_element.index[0]]['WIN_NATIONALIDC']
            row['WIN_TOWNC'] = df_co.iloc[max_element.index[0]]['WIN_TOWNC']
            row['WIN_POSTAL_CODEC'] = df_co.iloc[max_element.index[0]]['WIN_POSTAL_CODEC']
            row['B_CONTRACTOR_SME'] = df_co.iloc[max_element.index[0]]['B_CONTRACTOR_SME']
            row['NUTS_ID'] = df_co.iloc[max_element.index[0]]['NUTS_ID']
    return row

if generate_df_win_companies_without is True:
    # Quedarse con las filas únicas de WIN_NAMEC, WIN_NATIONALIDC=NaN y WIN_COUNTRY_CODE
    columns = ['WIN_NAMEC', 'WIN_NATIONALIDC', 'WIN_COUNTRY_CODE', 'WIN_TOWNC', 'WIN_POSTAL_CODEC', 'B_
```

```
CONTRACTOR_SME']
    df_aux = df[(df['WIN_NATIONALIDC'].isnull()) & (df['WIN_NAMEC_LEN'] == 1) & (df['WIN_COUNTRY_CODE_L
EN'] == 1)][columns]
    # Convert tuples of one element to own element
    for col in columns:
        if col != 'WIN_NATIONALIDC':
            df_aux[col] = df_aux[col].apply(lambda x: x[0] if x is not None and any(x) else None)
    # Drop duplicates and reset index
    df_aux.drop_duplicates(subset=['WIN_NAMEC', 'WIN_COUNTRY_CODE'], inplace=True)
    df_aux.reset_index(drop=True, inplace=True)
    # A cada fila, hacer el fuzz.partial_ratio con el df_companies filtrando por pais y (ciudad o códig
o postal)
    df_co_without_NATIONALID = df_aux.progress_apply(lambda row: fuzz_name_company(row, df_co[df_co['WI
N_COUNTRY_CODE'].isin([row['WIN_COUNTRY_CODE']]) &
                                                                                                (df_co['W
IN_TOWNC'].isin([row['WIN_TOWNC']]) | df_co['WIN_POSTAL_CODEC'].isin([row['WIN_POSTAL_CODEC']]))]), axi
s='columns')
    # Drop NaN in WIN_NATIONALIDC
    df_co_without_NATIONALID.dropna(subset=['WIN_NATIONALIDC'], inplace=True)
    # Replace NaN by None
    df_co_without_NATIONALID.replace({np.nan: None}, inplace=True)
    # Drop duplicates and reset index
    df_co_without_NATIONALID.drop_duplicates(inplace=True)
    df_co_without_NATIONALID.reset_index(drop=True, inplace=True)
    # Save pickle
    df_co_without_NATIONALID.to_pickle(path_TED + df_win_companies_without, compression='infer', protoc
ol=5)
else:
    # Load pickle
    df_co_without_NATIONALID = pd.read_pickle(os.path.join(path_TED, df_win_companies_without))
```

2.5 Apply the dataframe of similar name contractors to fill contractor's information for tenders with one winner

```
# Create tuples
columns = ['WIN_NAMEC', 'WIN_NATIONALIDC', 'WIN_COUNTRY_CODE', 'WIN_TOWNC', 'WIN_POSTAL_CODEC', 'B_CONT
RACTOR_SME']
for col in columns:
    df[col] = df[col].apply(lambda x: tuple(x) if x is not None else None)

# 5. Merge df_co_without_NATIONALID with df
df = df.merge(df_co_without_NATIONALID.applymap(lambda x: tuple([x])), how='left', on=['WIN_NAMEC', 'WI
N_COUNTRY_CODE'], suffixes=('', '_y'))
# Unify WIN_NATIONALIDC, WIN_TOWNC and WIN_POSTAL_CODEC
df['WIN_NATIONALIDC'] = df['WIN_NATIONALIDC_y'].fillna(df['WIN_NATIONALIDC'])
df['WIN_TOWNC'] = df['WIN_TOWNC_y'].fillna(df['WIN_TOWNC'])
df['WIN_POSTAL_CODEC'] = df['WIN_POSTAL_CODEC_y'].fillna(df['WIN_POSTAL_CODEC'])
df['B_CONTRACTOR_SME'] = df['B_CONTRACTOR_SME_y'].fillna(df['B_CONTRACTOR_SME'])
df['NUTS_ID'] = df['NUTS_ID_y'].fillna(df['NUTS_ID'])
# Delete temporal columns
df.drop(df.filter(regex='_y$').columns, axis=1, inplace=True)
# Merge df_companies with df to get the correct WIN_NAMEC
df = df.merge(df_co.applymap(lambda x: tuple([x])), how='left', on=['WIN_NATIONALIDC', 'WIN_COUNTRY_COD
E'], suffixes=('', '_y'))
# Unify WIN_NAMEC
df['WIN_NAMEC'] = df['WIN_NAMEC_y'].fillna(df['WIN_NAMEC'])
# Delete temporal columns
df.drop(df.filter(regex='_y$').columns, axis=1, inplace=True)

# 6. Merge df with df_co by WIN_NAME_VALID, WIN_NATIONALIDC_VALID and WIN_COUNTRY_CODE
df = df.merge(df_co.applymap(lambda x: tuple([x])), how='left', left_on=['WIN_NAMEC', 'WIN_NATIONALIDC'
, 'WIN_COUNTRY_CODE'], right_on=['WIN_NAMEC_VALID', 'WIN_NATIONALIDC_VALID', 'WIN_COUNTRY_CODE'], suffi
xes=('', '_y'))
# Unify all fields' adjudicator
df['WIN_NAMEC'] = df['WIN_NAMEC_VALID_y'].fillna(df['WIN_NAMEC'])
df['WIN_NATIONALIDC'] = df['WIN_NATIONALIDC_VALID_y'].fillna(df['WIN_NATIONALIDC'])
df['WIN_TOWNC'] = df['WIN_TOWNC_y'].fillna(df['WIN_TOWNC'])
df['WIN_POSTAL_CODEC'] = df['WIN_POSTAL_CODEC_y'].fillna(df['WIN_POSTAL_CODEC'])
df['B_CONTRACTOR_SME'] = df['B_CONTRACTOR_SME_y'].fillna(df['B_CONTRACTOR_SME'])
df['NUTS_ID'] = df['NUTS_ID_y'].fillna(df['NUTS_ID'])
# Delete df_co columns
df.drop(df.filter(regex='_y$').columns, axis=1, inplace=True)
df.drop(['WIN_COUNT_TENDERS', 'WIN_NAMEC_VALID', 'WIN_NATIONALIDC_VALID'], axis=1, inplace=True)
```

```
# Tuples to list
columns = ['WIN_NAMEC', 'WIN_NATIONALIDC', 'WIN_COUNTRY_CODE', 'WIN_TOWNC', 'WIN_POSTAL_CODEC', 'B_CONTRACTOR_SME']
for col in columns:
    df[col] = df[col].apply(lambda x: list(x) if x is not None else None)
# Update length columns and WIN_NUMBER_OK
df = calculate_length(df)
```

2.6 Apply the two dataframe (unique contractors and similar name contractors) to fill contractor's information for tenders with two or more winners

```
def assign_correct_values(row, df_co, df_co_without_NATIONALID):
    ''' '''

    for i, national_id in enumerate(row['WIN_NATIONALIDC']):
        if national_id is not None:
            # Assign correct WIN_NAMEC (and town and postal code) from WIN_NATIONALIDC in df_co
            register = df_co[(df_co['WIN_NATIONALIDC'].isin([national_id])) & (df_co['WIN_COUNTRY_CODE'].isin([row['WIN_COUNTRY_CODE'][i]]))]
            if register.empty is False:
                if register['WIN_NAMEC_VALID'].values[0] is None:
                    row['WIN_NAMEC'][i] = register['WIN_NAMEC'].values[0]
                    row['WIN_TOWNC'][i] = register['WIN_TOWNC'].values[0]
                    row['WIN_POSTAL_CODEC'][i] = register['WIN_POSTAL_CODEC'].values[0]
                    row['B_CONTRACTOR_SME'][i] = register['B_CONTRACTOR_SME'].values[0]
                    row['NUTS_ID'][i] = register['NUTS_ID'].values[0]
                else:
                    row['WIN_NAMEC'][i] = register['WIN_NAMEC_VALID'].values[0]
                    row['WIN_NATIONALIDC'][i] = register['WIN_NATIONALIDC_VALID'].values[0]
                    row['WIN_TOWNC'][i] = df_co[(df_co['WIN_NATIONALIDC'].isin([register['WIN_NATIONALIDC'].values[0]])) & (df_co['WIN_COUNTRY_CODE'].isin([row['WIN_COUNTRY_CODE'][i]]))]['WIN_TOWNC'].values[0]
                    row['WIN_POSTAL_CODEC'][i] = df_co[(df_co['WIN_NATIONALIDC'].isin([register['WIN_NATIONALIDC'].values[0]])) & (df_co['WIN_COUNTRY_CODE'].isin([row['WIN_COUNTRY_CODE'][i]]))]['WIN_POSTAL_CODEC'].values[0]
                    row['B_CONTRACTOR_SME'][i] = df_co[(df_co['WIN_NATIONALIDC'].isin([register['WIN_NATIONALIDC'].values[0]])) & (df_co['WIN_COUNTRY_CODE'].isin([row['WIN_COUNTRY_CODE'][i]]))]['B_CONTRACTOR_SME'].values[0]
                    row['NUTS_ID'][i] = df_co[(df_co['WIN_NATIONALIDC'].isin([register['WIN_NATIONALIDC'].values[0]])) & (df_co['WIN_COUNTRY_CODE'].isin([row['WIN_COUNTRY_CODE'][i]]))]['NUTS_ID'].values[0]
                    continue
            # Assign correct WIN_NATIONALIDC (and town and postal code) from WIN_NAMEC in df_co
            register = df_co[(df_co['WIN_NAMEC'].isin([row['WIN_NAMEC'][i]])) & (df_co['WIN_COUNTRY_CODE'].isin([row['WIN_COUNTRY_CODE'][i]]))]
            if register.empty is False:
                if register['WIN_NATIONALIDC_VALID'].values[0] is None:
                    row['WIN_NATIONALIDC'][i] = register['WIN_NATIONALIDC'].values[0]
                    row['WIN_TOWNC'][i] = register['WIN_TOWNC'].values[0]
                    row['WIN_POSTAL_CODEC'][i] = register['WIN_POSTAL_CODEC'].values[0]
                    row['B_CONTRACTOR_SME'][i] = register['B_CONTRACTOR_SME'].values[0]
                    row['NUTS_ID'][i] = register['NUTS_ID'].values[0]
                else:
                    row['WIN_NAMEC'][i] = register['WIN_NAMEC_VALID'].values[0]
                    row['WIN_NATIONALIDC'][i] = register['WIN_NATIONALIDC_VALID'].values[0]
                    row['WIN_TOWNC'][i] = df_co[(df_co['WIN_NATIONALIDC'].isin([register['WIN_NATIONALIDC'].values[0]])) & (df_co['WIN_COUNTRY_CODE'].isin([row['WIN_COUNTRY_CODE'][i]]))]['WIN_TOWNC'].values[0]
                    row['WIN_POSTAL_CODEC'][i] = df_co[(df_co['WIN_NATIONALIDC'].isin([register['WIN_NATIONALIDC'].values[0]])) & (df_co['WIN_COUNTRY_CODE'].isin([row['WIN_COUNTRY_CODE'][i]]))]['WIN_POSTAL_CODEC'].values[0]
                    row['B_CONTRACTOR_SME'][i] = df_co[(df_co['WIN_NATIONALIDC'].isin([register['WIN_NATIONALIDC'].values[0]])) & (df_co['WIN_COUNTRY_CODE'].isin([row['WIN_COUNTRY_CODE'][i]]))]['B_CONTRACTOR_SME'].values[0]
                    row['NUTS_ID'][i] = df_co[(df_co['WIN_NATIONALIDC'].isin([register['WIN_NATIONALIDC'].values[0]])) & (df_co['WIN_COUNTRY_CODE'].isin([row['WIN_COUNTRY_CODE'][i]]))]['NUTS_ID'].values[0]
            else:
                # Assign correct WIN_NAMEC (and id, town and postal code) from WIN_NAMEC in df_co_without_NATIONALID
                register = df_co_without_NATIONALID[(df_co_without_NATIONALID['WIN_NAMEC'].isin([row['WIN_NAMEC'][i]])) & (df_co_without_NATIONALID['WIN_COUNTRY_CODE'].isin([row['WIN_COUNTRY_CODE'][i]]))]
                if register.empty is False:
                    if  df_co[(df_co['WIN_NATIONALIDC'].isin([register['WIN_NATIONALIDC'].values[0]])) & (df_co['WIN_COUNTRY_CODE'].isin([row['WIN_COUNTRY_CODE'][i]]))]['WIN_NAMEC_VALID'].values[0] is None:
                        row['WIN_NAMEC'][i] = df_co[(df_co['WIN_NATIONALIDC'].isin([register['WIN_NATIONALIDC'].values[0]])) & (df_co['WIN_COUNTRY_CODE'].isin([row['WIN_COUNTRY_CODE'][i]]))]['WIN_NAMEC'].values
```

```
[0]
                    row['WIN_NATIONALIDC'][i] = register['WIN_NATIONALIDC'].values[0]
                    row['WIN_TOWNC'][i] = register['WIN_TOWNC'].values[0]
                    row['WIN_POSTAL_CODEC'][i] = register['WIN_POSTAL_CODEC'].values[0]
                    row['B_CONTRACTOR_SME'][i] = register['B_CONTRACTOR_SME'].values[0]
                    row['NUTS_ID'][i] = register['NUTS_ID'].values[0]
                else:
                    row['WIN_NAMEC'][i] = df_co[(df_co['WIN_NATIONALIDC'].isin([register['WIN_NATIONALI
DC'].values[0]])) & (df_co['WIN_COUNTRY_CODE'].isin([row['WIN_COUNTRY_CODE'][i]]))]['WIN_NAMEC_VALID'].
values[0]
                    row['WIN_NATIONALIDC'][i] = df_co[(df_co['WIN_NATIONALIDC'].isin([register['WIN_NAT
IONALIDC'].values[0]])) & (df_co['WIN_COUNTRY_CODE'].isin([row['WIN_COUNTRY_CODE'][i]]))]['WIN_NATIONAL
IDC_VALID'].values[0]
                    row['WIN_TOWNC'][i] = df_co[(df_co['WIN_NATIONALIDC'].isin([register['WIN_NATIONALI
DC'].values[0]])) & (df_co['WIN_COUNTRY_CODE'].isin([row['WIN_COUNTRY_CODE'][i]]))]['WIN_TOWNC'].values
[0]
                    row['WIN_POSTAL_CODEC'][i] = df_co[(df_co['WIN_NATIONALIDC'].isin([register['WIN_NA
TIONALIDC'].values[0]])) & (df_co['WIN_COUNTRY_CODE'].isin([row['WIN_COUNTRY_CODE'][i]]))]['WIN_POSTAL_
CODEC'].values[0]
                    row['B_CONTRACTOR_SME'][i] = df_co[(df_co['WIN_NATIONALIDC'].isin([register['WIN_NA
TIONALIDC'].values[0]])) & (df_co['WIN_COUNTRY_CODE'].isin([row['WIN_COUNTRY_CODE'][i]]))]['B_CONTRACTO
R_SME'].values[0]
                    row['NUTS_ID'][i] = df_co[(df_co['WIN_NATIONALIDC'].isin([register['WIN_NATIONALIDC
'].values[0]])) & (df_co['WIN_COUNTRY_CODE'].isin([row['WIN_COUNTRY_CODE'][i]]))]['NUTS_ID'].values[0]
            else:
                # Not found
                row['WIN_TOWNC'][i] = None
                row['WIN_POSTAL_CODEC'][i] = None
                row['B_CONTRACTOR_SME'][i] = None
                row['NUTS_ID'][i] = None
    return row

df_aux = df[(df['WIN_NAMEC_LEN'] == df['WIN_COUNTRY_CODE_LEN']) & (df['WIN_NAMEC_LEN'] > 1)].copy()

columns = ['WIN_NATIONALIDC', 'WIN_TOWNC', 'WIN_POSTAL_CODEC', 'B_CONTRACTOR_SME', 'NUTS_ID']
for col in columns:
    df_aux[col] = df_aux.apply(lambda row: ([None] * int(row['WIN_NAMEC_LEN'])) if row[col] is None or
row[col] is np.NAN or len(row[col]) != row['WIN_NAMEC_LEN'] else row[col], axis='columns')

# 7. Assign correct values using df_co and df_co_without_NATIONALID
df_aux = df_aux.progress_apply(lambda row: assign_correct_values(row, df_co, df_co_without_NATIONALID),
axis='columns')

# Update df with df_aux's information
df = df.reset_index().set_index('index')
df_aux = df_aux.reset_index().set_index('index')
df.update(df_aux)

# Tuples to list
columns = ['WIN_NAMEC', 'WIN_NATIONALIDC', 'WIN_COUNTRY_CODE', 'WIN_TOWNC', 'WIN_POSTAL_CODEC', 'B_CONT
RACTOR_SME', 'NUTS_ID']
for col in columns:
    df[col] = df[col].apply(lambda x: list(x) if x is not None and x is not np.NAN  else None)
# Update length columns and WIN_NUMBER_OK
df = calculate_length(df)
```

2.7 Add geolocation (latitude and longitude) and NUTS for tenders (with one winner) without this information

```
# Create dataframe with geolocation info thanks to df_co
df_co_lat_lon_unique = df_co[df_co['WIN_COUNTRY_CODE'].notna() & df_co['WIN_TOWNC'].notna() & df_co['LA
T_LON'].notna()].drop_duplicates(subset=['WIN_COUNTRY_CODE', 'WIN_TOWNC'])[['WIN_COUNTRY_CODE', 'WIN_TO
WNC', 'LAT_LON', 'NUTS_ID']]

# Create tuples
columns = ['WIN_COUNTRY_CODE', 'WIN_TOWNC']
for col in columns:
    df[col] = df[col].apply(lambda x: tuple(x) if x is not None else None)

# Create dataframe with particular conditions
df_aux = df[df['NUTS_ID'].isna() & df['WIN_COUNTRY_CODE'].notna() & df['WIN_TOWNC'].notna() & df['WIN_N
AMEC_LEN'] == 1].drop_duplicates(subset=['WIN_COUNTRY_CODE', 'WIN_TOWNC'])[['WIN_COUNTRY_CODE', 'WIN_TO
WNC', 'LAT_LON', 'NUTS_ID']]
```

```
# Convert tuples of one element to own element
columns = ['WIN_COUNTRY_CODE', 'WIN_TOWNC', 'NUTS_ID']
for col in columns:
    df[col] = df[col].apply(lambda x: x[0] if x is not None else None)
    df_aux[col] = df_aux[col].apply(lambda x: x[0] if x is not None else None)
# Delete columns NUTS_ID (it's empty)
df_aux.drop(['NUTS_ID'], axis=1, inplace=True)

# Merge df_aux and df_co_lat_lon_unique by WIN_COUNTRY_CODE and WIN_TOWNC
df_aux = df_aux.merge(df_co_lat_lon_unique, how='left', on=['WIN_COUNTRY_CODE', 'WIN_TOWNC'], suffixes=
('', '_y'))
# Unify fields
df_aux['LAT_LON'] = df_aux['LAT_LON_y'].fillna(df_aux['LAT_LON'])
# Delete temporal columns
df_aux.drop(df_aux.filter(regex='_y$').columns, axis=1, inplace=True)
# Delete rows without lat_lon
df_aux.dropna(subset=['LAT_LON'], inplace=True)
# To unify with df['LAT_LON']
df_aux['LAT_LON'] = df_aux['LAT_LON'].apply(lambda x: tuple([tuple(x)]) if x is not None else None)

# Merge df with df_aux by WIN_COUNTRY_CODE and WIN_TOWNC
df = df.merge(df_aux, how='left', on=['WIN_COUNTRY_CODE', 'WIN_TOWNC'], suffixes=('', '_y'))
# Unify fields
df['LAT_LON'] = df['LAT_LON_y'].fillna(df['LAT_LON'])
df['NUTS_ID'] = df['NUTS_ID_y'].fillna(df['NUTS_ID'])
# Delete temporal columns
df.drop(df.filter(regex='_y$').columns, axis=1, inplace=True)

# Tuples to list
columns = ['WIN_COUNTRY_CODE', 'WIN_TOWNC', 'NUTS_ID']
for col in columns:
    df[col] = df[col].apply(lambda x: [x] if x is not None else None)

# Create tuples
columns = ['WIN_COUNTRY_CODE', 'WIN_TOWNC']
for col in columns:
    df[col] = df[col].apply(lambda x: tuple(x) if x is not None else None)

# Disable SSL to use the geo-service Nominatim
ctx = ssl.create_default_context()
ctx.check_hostname = False
ctx.verify_mode = ssl.CERT_NONE
# Create the service Nominatim
geolocator = Nominatim(user_agent='myapp', timeout=5, ssl_context=ctx, adapter_factory=geopy.adapters.U
RLLibAdapter)
geocode = partial(geolocator.geocode, language='en')
# Create dataframe with particular conditions
df_aux = df[df['NUTS_ID'].isna() & df['WIN_COUNTRY_CODE'].notna() & df['WIN_TOWNC'].notna() & df['WIN_N
AMEC_LEN'] == 1].drop_duplicates(subset=['WIN_COUNTRY_CODE', 'WIN_TOWNC'])
# Convert tuples of one element to own element
columns = ['WIN_COUNTRY_CODE', 'WIN_TOWNC', 'NUTS_ID']
for col in columns:
    df[col] = df[col].apply(lambda x: x[0] if x is not None else None)
    df_aux[col] = df_aux[col].apply(lambda x: x[0] if x is not None else None)
# Delete columns NUTS_ID (it's empty)
df_aux.drop(['NUTS_ID'], axis=1, inplace=True)
# WIN_POSTAL_CODEC is not used
df_aux['WIN_POSTAL_CODEC'] = None
# Get latitude and longitude of each row (WIN_COUNTRY_CODE, WIN_TOWNC and WIN_POSTAL_CODEC)
df_aux['LAT_LON'] = df_aux.progress_apply(lambda row: get_lat_lon(row, geocode), axis=1)

# Create geopandas from df_aux
df_nuts = pd.DataFrame([])
df_nuts['LAT_LON'] = df_aux.drop_duplicates(subset=['LAT_LON'])['LAT_LON']
df_nuts['POINT'] = df_aux.drop_duplicates(subset=['LAT_LON'])['LAT_LON'].apply(lambda x: Point(x[1], x[
0]) if len(x) > 0 and any(x) else None)
gdf_df_nuts = gpd.GeoDataFrame(df_nuts, geometry='POINT', crs='EPSG:4326').to_crs('EPSG:3857')
# Get map with NUTS (level 3) information
map = gpd.read_file(os.path.join(path_TED, NUTS_RG_01M_2021_3857), driver='GeoJSON')
mask_EU = (-1.5*10**6, 0.4*10**7, 4*10**6, 1.2*10**7)
gdf_map = gpd.GeoDataFrame(map[map['LEVL_CODE'].isin([3])], geometry='geometry', crs='EPSG:3857').clip(
mask_EU)
# Calculate NUTS (level 3) from point
gdf_df_nuts['NUTS_ID'] = gdf_df_nuts['POINT'].progress_apply(lambda x: get_nuts_from_point(x, gdf_map)
```

```
if x is not None else None)
# Merge df and gdf_df_nuts
df_aux = df_aux.merge(gdf_df_nuts[['NUTS_ID', 'LAT_LON']], how='left', on=['LAT_LON'])
# To unify with df['LAT_LON']
df_aux['LAT_LON'] = df_aux['LAT_LON'].apply(lambda x: tuple([tuple(x)]) if x is not None else None)

# Merge df with df_aux by WIN_COUNTRY_CODE, WIN_TOWNC and WIN_POSTAL_CODEC
df = df.merge(df_aux[['WIN_COUNTRY_CODE', 'WIN_TOWNC', 'LAT_LON', 'NUTS_ID']], how='left', on=['WIN_COU
NTRY_CODE', 'WIN_TOWNC'], suffixes=('', '_y'))

# Unify fields
df['LAT_LON'] = df['LAT_LON_y'].fillna(df['LAT_LON'])
df['NUTS_ID'] = df['NUTS_ID_y'].fillna(df['NUTS_ID'])
# Delete temporal columns
df.drop(df.filter(regex='_y$').columns, axis=1, inplace=True)

# Tuples to list
columns = ['WIN_COUNTRY_CODE', 'WIN_TOWNC', 'NUTS_ID']
for col in columns:
    df[col] = df[col].apply(lambda x: [x] if x is not None else None)
```

2.8 Add geolocation (latitude and longitude) and NUTS3 for CAE (Contracting Authority Entity)

```
def get_lat_lon_CAE(row, geocode):
    ''' Get latitude and longitude from address'''

    if row['ISO_COUNTRY_CODE'] is not None and row['CAE_TOWNC'] is not None:
        query = row['ISO_COUNTRY_CODE'] + ', ' + row['CAE_TOWNC']
        try:
            location = geocode(query)
            if location is not None: return (location.latitude, location.longitude)
        except Exception as e:
            print(e)
    return (None, None)

# Rename columns to identify univocy lat_lon and nuts of winners
df.rename(columns={'LAT_LON': 'WIN_LAT_LON', 'NUTS_ID': 'WIN_NUTS3'}, inplace=True)

# Create tuples
columns = ['WIN_COUNTRY_CODE', 'WIN_TOWNC', 'WIN_NUTS3']
for col in columns:
    df[col] = df[col].apply(lambda x: tuple(x) if x is not None else None)
df['ISO_COUNTRY_CODE'] = df['ISO_COUNTRY_CODE'].apply(lambda x: tuple([x]) if x is not None else None)
# Clean CAE_TOWN
df['CAE_TOWNC'] = df['CAE_TOWN'].astype(str).apply(lambda x: clean_name(x, True))
# Consider only the first town when there are some towns (separator is '---')
df['CAE_TOWNC'] = df['CAE_TOWNC'].str.split('---').apply(lambda string: tuple([string[0]]) if string is
not None and any(string) else None)
# Create dataframe with geolocation info thanks to columns WIN_COUNTRY_CODE, WIN_TOWNC and WIN_NUTS3
df_aux = df.drop_duplicates(subset=['WIN_COUNTRY_CODE', 'WIN_TOWNC', 'WIN_NUTS3'])[['WIN_COUNTRY_CODE',
'WIN_TOWNC', 'WIN_NUTS3']]
# Merge previous dataframe with df (ISO_COUNTRY_CODE and CAE_TOWNC) to calculate CAE_NUTS3
df = df.merge(df_aux, how='left', left_on=['ISO_COUNTRY_CODE', 'CAE_TOWNC'], right_on=['WIN_COUNTRY_COD
E', 'WIN_TOWNC'], suffixes=('', '_y'))
df['CAE_NUTS3'] = df['WIN_NUTS3_y'].apply(lambda x: x[0] if x is not np.NAN and x is not None else None
)
df.drop(df.filter(regex='_y$').columns, axis=1, inplace=True)

# Calculate rest of NUTS3 which are empty
df_aux = df[df['CAE_NUTS3'].isna()].drop_duplicates(subset=['ISO_COUNTRY_CODE', 'CAE_TOWNC'])[['ISO_COU
NTRY_CODE', 'CAE_TOWNC']]
# Disable SSL to use the geo-service Nominatim
ctx = ssl.create_default_context()
ctx.check_hostname = False
ctx.verify_mode = ssl.CERT_NONE
# Create the service Nominatim
geolocator = Nominatim(user_agent='myapp', timeout=5, ssl_context=ctx, adapter_factory=geopy.adapters.U
RLLibAdapter)
geocode = partial(geolocator.geocode, language='en')
# Convert tuples of one element to own element
columns = ['ISO_COUNTRY_CODE', 'CAE_TOWNC']
for col in columns:
    df_aux[col] = df_aux[col].apply(lambda x: x[0] if x is not None else None)
    df[col] = df[col].apply(lambda x: x[0] if x is not None else None)
```

```
# Get latitude and longitude of each row (ISO_COUNTRY_CODE and CAE_TOWNC)
df_aux['CAE_LAT_LON'] = df_aux.progress_apply(lambda row: get_lat_lon_CAE(row, geocode), axis=1)

# Create geopandas from df_aux
df_aux['POINT'] = df_aux['CAE_LAT_LON'].apply(lambda x: Point(x[1], x[0]) if len(x) > 0 and any(x) else
None)
df_aux = df_aux.dropna(subset=['POINT'])
gdf_df_nuts = gpd.GeoDataFrame(df_aux, geometry='POINT', crs='EPSG:4326').to_crs('EPSG:3857')
# Get map with NUTS (level 3) information
map = gpd.read_file(os.path.join(path_TED, NUTS_RG_01M_2021_3857), driver='GeoJSON')
mask_EU = (-1.5*10**6, 0.4*10**7, 4*10**6, 1.2*10**7)
gdf_map = gpd.GeoDataFrame(map[map['LEVL_CODE'].isin([3])], geometry='geometry', crs='EPSG:3857').clip(
mask_EU)
# Calculate NUTS (level 3) from point
gdf_df_nuts['CAE_NUTS3'] = gdf_df_nuts['POINT'].apply(lambda x: get_nuts_from_point(x, gdf_map) if x is
not None else None)
# Merge df and gdf_df_nuts
df = df.merge(gdf_df_nuts[['ISO_COUNTRY_CODE', 'CAE_TOWNC', 'CAE_NUTS3']], how='left', on=['ISO_COUNTRY
_CODE', 'CAE_TOWNC'], suffixes=('', '_y'))
# Unify fields
df['CAE_NUTS3'] = df['CAE_NUTS3_y'].fillna(df['CAE_NUTS3'])
# Delete temporal columns
df.drop(df.filter(regex='_y$').columns, axis=1, inplace=True)

# Tuples to list
columns = ['WIN_COUNTRY_CODE', 'WIN_TOWNC', 'WIN_NUTS3']
for col in columns:
    df[col] = df[col].apply(lambda x: [x[0]] if x is not None else None)

# Save final dataframe
df.to_pickle(os.path.join(path_TED, 'df.pkl'), compression='infer', protocol=5)
```