Replication of Ciccone 2002 by Laurens Hof

In [1]:
```
import matplotlib.pyplot as plt
import matplotlib.patches as mpatch
import pandas as pd
import numpy as np
import statsmodels.formula.api as smf
import eurostat
from linearmodels.iv import IV2SLS
```

Research question and motivation

I am going to replicate a 2002 paper by Antonio Ciccone00099-4), which tries to measure the size of agglomeration efficiencies in Europe. It does this by regressing labor productivity on density of employment for NUTS-3 areas in selected European countries, using land area as an instrumental variable.

This question is interesting because urban density has long been touted as one of the primary drivers of prosperity for developed (Rosenthal and Strange, 2004)80006-3) and developing (Dorosh and Thurlow, 2014) countries alike, as well as in historical contexts (Jedwab and Vollrath, 2015). It is therefore very relevant to policy to have an unbiased and up-to-date estimate of the size of this effect. The original paper was published in 2002 and uses data from the late 1980's, which can raise doubts as to whether the conclusions still hold today. However, replicating it is relatively straightforward since almost all the necessary data are made freely available by Eurostat.

Method and assumptions

I follow the original author in their set-up. They derive their estimating equation from the following specification of a production function for a unit of land:

$$q = \Omega_{sc}\left((nH)^{\beta}k^{1-\beta}\right)^{\alpha}\left(\frac{Q_{sc}}{A_{sc}}\right)^{(\lambda-1)/\lambda}$$

where $0 \leq \beta \leq 1$ and $0 \leq \alpha \leq 1$

Here, q denotes the output produced on the unit of land. Ω_{sc} denotes unexplained total factor productivity for region s within country (or larger region) c. n is the number of workers on the unit of land and H is their average human capital. k is the amount of physical capital employed. Q_{sc} and A_{sc} are supposed to be the source of the agglomeration effects: they denote total production and total land in the region respectively, again for region s within country c.

The specific form of this production function contains some assumptions that were likely chosen for convenience, but which are not very controversial. At its core it is a very standard Cobb-Douglas production function with labor-enhancing human capital. α captures the returns to

capital and labor on the unit of land, which are distributed according to β. The first step towards turning this production function into an estimable equation is the (rather strong) assumption that capital and labor are distributed among the units of land in the region. Then,

$$Q_{sc} = A_{sc}q$$

which, after substituting the first equation and solving for average labor productivity yields:

$$\frac{Q_{sc}}{N_{sc}} = \Omega^{\lambda}\left(H_{sc}^{\beta}\left(\frac{K_{sc}}{N_{sc}}\right)^{1-\beta}\right)^{\alpha\lambda}\left(\frac{N_{sc}}{A_{sc}}\right)^{\alpha\lambda-1}$$

This equation only uses local aggregates as variables: K_{sc} and N_{sc} are total capital and total amount of workers within area s, respectively.

This equation requires information on the capital density within each area, which is not available. However, this problem can be circumvented by assuming that the rental rate of capital, r, is equal across a country. Then, assuming that capital gets hired until the market clears, we can use capital demand as a substitute. This is equal to:

$$K_{sc} = \frac{\alpha(1-\beta)}{r_c}Q_{sc}$$

If we use this as a proxy for actual employed capital, the expression for average labor productivity becomes:

$$\frac{Q_{sc}}{N_{sc}} = \Lambda_c\Omega_{sc}H_{sc}\left(\frac{N_{sc}H_{sc}}{A_{sc}}\right)^{\theta}$$

Here, Λ_c depends on the country-wide rental rate of capital, and the convenient thing is that it can be accounted for using country-level fixed effects. This is very close the estimating equation, and the new variable is θ. This is the coefficient we are going to be estimating, and it is what is capturing the agglomeration effects. It is given by:

$$\theta = \frac{\alpha\lambda-1}{1-\alpha\lambda(1-\beta)}$$

θ measures the manner in which the density of human capital and employment affect local labor productivity. It should be thought of as a "net agglomeration effect", in the sense that it is determined by the product of α and λ. λ captures the extent to which density of production increases production efficiency, while α captures the extent to which there are diminishing returns to capital and labor within a unit of land. If this product is smaller than 1, θ is negative and increasing density reduces average labor productivity declines with employment density. If the product is equal to 1, θ is 0 and employment density is irrelevant. Finally, if $\alpha\lambda$ is greater than 1, it is positive and employment density increases regional labor productivity.

It should be emphasized that θ measures the effect of employment density on local **labor productivity**, not total factor productivity. A good way to think about it is to conceptualize θ as the "net agglomeration effect *after capital arbitrage*". After all, we our assumptions until this point imply that capital moves to the places where it can be employed most productively. As a

consequence, the magnitude of θ becomes larger as β (the relative productivity of labor) becomes smaller. One might ask why labor does not move to its most productive location. The easiest answer to this is that workers apparently experience amenities in less dense places, or disamenities in denser places (congestion, pollution, etc.). It could also be that workers are compensated in the form of lower-cost housing.

Taking logs of the capital-corrected average productivity expression implies that:

$$\ln Q_{sc} - \ln N_{sc} = \ln \Lambda_c + \theta \left(\ln N_{sc} - \ln A_{sc} \right) + (\theta + 1) \ln H_{sc} + \ln \Omega_{sc}$$

Which yields the estimation equation:

$$\ln Q_{sc} - \ln N_{sc} = \theta_0 + \theta_1 \left(\ln N_{sc} - \ln A_{sc} \right) + \delta_1 W_{sc} + \delta_2 Z + \varepsilon_{sc}$$

Where W is a collection of coefficients for each level of education, indicating which fraction of the population has attained that level of education, and Z is a set of country-level fixed effects. Since this is a log-log regression, θ_1 can be interpreted as an elasticity: "if employment density increases by 1 percent, this will have an x percent increase/decrease in labor productivity as a consequence".

Data

The data I will use comes from the Eurostat regio database, which provides statistics for subnational regions ("NUTS-regions", for "*Nomenclature des Unités Territoriales Statistiques*" or "Nomenclature of Territorial Units for Statistics") in the EFTA and potential members states. Like the original author, I will use data on the NUTS-3 level for Germany, the UK, Italy and Spain. The original paper includes France, but I have omitted it since there do not seem to be employment data available at the NUTS-3 level for France. NUTS-3 regions correspond to *kreise* in Germany, *counties* in the UK, *provincie* in Italy and *provincias* in Spain (or COROP-regions in the Netherlands).

This setup has some reverse causality problems: is production more efficient because it is close together, or does production cluster because it is more productive in that place? The author addresses this by selecting countries for which the NUTS-3 divisions are historically predetermined, and then using land area as an instrumental variable, since it turns out that land area and employment density are negatively correlated.The explanation the author posits is that, since the historical divisions were usually made for administrative purposes, which made it "common sense" to roughly equalize population size across regions. This caused a negative correlation between land area and population density, which in turn persisted to the present day in causing a negative correlation between area and employment density.

The argument for exclusion of this instrument rests on the supposition that late 19th century population density has no effect on current-day productivity, except through its effect on current-day employment density. While several papers on the long run persistence historical effects (such as Dell 2010 or Bartels, Jaeger and Obergruber 2020), they are typically theorized to

work either through institutional mechanisms or cultural attitudes, both of which can be expected to stongly be dampened by the country-level fixed effects. My IV will then look like this:

$$Y_{sc} = \theta_0 + \theta_1 \widehat{X_{sc}}\,(A_{sc}) + \delta_1 W_{sc} + \delta_2 Z + \varepsilon_{sc}$$

Where X is the difference in the log of employment and the log of area in that area, with the hat indicating that I use the fitted values resulting from regressing it on area. Y is the difference in log added value and log employment, indicating labor productivity.

Preview of the answers

Multiple different specifications fail to reject the null hypothesis at the 0.05 threshold. Whereas the original paper found a 2SLS point estimate of 0.0455, mine is 0.0092 with and upper bound of 0.0213. My naive OLS estimate is different, too: 0.027 vs 0.05.

Python code

Importing data

The "eurostat" package contains a convenient set of wrapper functions for retrieving eurostat data. I need data on:

- Land area of each NUTS-3 region. This is our instrumental variable.
- Value added in each NUTS-3 region, in current market prices. This is our dependent variable. We want to exlcude the agricultural sector from this measure. This is a lot easier in my replication than in the original paper, since now Eurostat has data available subdividing value added by top-level NACE indicator.
- Employment per NUTS-3 region. Again, it is simple to exclude agriculture.
- Education per NUTS-3 region. I have not been able to find a database of employment by education at NUTS-3 level, or even of population by education. There *is* data available on employment by education by NUTS-2 level, so a very crude solution is to make the (strong) assumption that average education level is spread evenly throughout each NUTS-2 region.

In [2]:
```
#area
area = eurostat.get_data_df("reg_area3")
#value added
va_NUTS3 = eurostat.get_data_df("nama_10r_3gva")
#employment
empl_NUTS3 = eurostat.get_data_df("nama_10r_3empers")
#employment by education
educ_NUTS2 = eurostat.get_data_df("lfst_r_lfe2eedu")
```

Collecting NUTS codes

I have downloaded a list of NUTS codes along with the names of the regions. Firstly, I need to read it in from the csv file.

In [3]:
```
codes = pd.read_csv("NUTS3.csv")
```

Step 2: basic sifting. Only keep those countries we will be analyzing, and get rid of "extra regions"

In [4]:
```
#create the list of country codes I will be analyzing
countrylist = ["DE", "ES", "IT", "UK"]
#select those rows of "codes" for which the first two symbols of the NUTS cod
# in the list
codes = codes[sum([codes["NUTS-Code"].str[:2] == c for c in countrylist]) == 1
#drop the "extra regions", whose code ends with "Z"
codes = codes[codes["NUTS-Code"].str[-1] != "Z"]
```

We want to exclude overseas territories and "outermost regions". I have manually created a list of which level 2 codes belong to overseas territories. There are no level 3 overseas codes that are under a non-overseas level 2 code.

In [5]:
```
overseas = ["ES70"]

#apply the same idea as with selecting the countries, except now we are exclu
codes = codes[sum([codes["NUTS-Code"].str[:4] == c for c in overseas]) == 0]
```

Find the codes that correspond to NUTS-3 regions. Unfortunately, using the "level" variable does not work, since it also includes some higher-level aggregates.

In [6]:
```
#select only the NUTS3 regions, which have a 5-character code.
level3 = codes[codes["NUTS-Code"].apply(len) == 5]

#select only the NUTS2 regions, which have a 4-character code
level2 = codes[codes["NUTS-Code"].apply(len) == 4]
```

Finally, convert the columns containing NUTS codes to lists, so that when filtering our actual data it is easier to check against.

In [7]:
```
NUTS3_selected = level3["NUTS-Code"].tolist()
NUTS2_selected = level2["NUTS-Code"].tolist()
```

Importing and shaping data

Data shaping: area

Area will be the "base" dataframe from which I will build my actual dataset.

Firstly, I keep only the values labeled "L0008". Contrary to the values labeled "TOTAL", this only includes land area. Secondly, keep only the regions I want to examine. Next, I will append two columns to the dataframe, indicating the country code and NUTS-2 code corresponding to the area, respectively. I will need this later to implement fixed effects.

In [8]:
```
#keep only the L0008 values
area = area[area["landuse"] == "L0008"]
#drop unused NUTS3 codes
area = area[area.iloc[:, 2].isin(NUTS3_selected)]
#add country code
area["country"] = area.iloc[:, 2].str[:2]
#add NUTS2 code
area["NUTS2"] = area.iloc[:, 2].str[:4]
```

I have two observations for area: one in 2013 and one in 2016, of which I will use the 2013 column since it has no missing values.

In [9]:
```
sum(area[2013].isna())
```

Out[9]: 0

In [10]:
```
sum(area[2016].isna())
```

Out[10]: 25

I will also drop other unnecessary columns, and rename the remaining ones. To this, I will append my other variables.

In [11]:
```
#subset columns
maindata1 = area.iloc[:, [2, 4, 5, 6]]
#rename columns
maindata1.columns.values[0] = "NUTS3"
maindata1.columns.values[1] = "area_km2"
maindata1
```

Out[11]:

	NUTS3	area_km2	country	NUTS2
210	DE111	206.0	DE	DE11
211	DE112	618.0	DE	DE11
212	DE113	641.0	DE	DE11
213	DE114	642.0	DE	DE11
214	DE115	685.0	DE	DE11
...	...	...	...	...
2177	UKN01	109.0	UK	UKN0
2178	UKN02	839.0	UK	UKN0
2179	UKN03	3137.0	UK	UKN0
2180	UKN04	3217.0	UK	UKN0
2181	UKN05	6260.0	UK	UKN0

683 rows × 4 columns

Data shaping: employment

The first steps will be to keep only the data for 2013 (to be consistent with the area data) and the data for the relevant NUTS-3 regions. I will only keep the rows labeled "EMP", which indicates employed persons rather than employees.

In [12]:
```
#filter relevant NUTS3
empl_NUTS3 = empl_NUTS3[empl_NUTS3["geo\\time"].isin(NUTS3_selected)]
#filter employed persons
empl_NUTS3 = empl_NUTS3[empl_NUTS3["wstatus"] == "EMP"]
#filter all years except 2013
empl_NUTS3 = empl_NUTS3[["nace_r2", "geo\\time", 2013]]
```

Then, I will need to remove agriculture from my employment data. The easiest way to do this I can think of is to pivot the NACE classifications into the columns and then subtract the "A" column from the "TOTAL" column. In-between those steps, I will substitute zeroes for the 13 missing values in agriculture so that they will not "infect" the subtraction. Since "TOTAL" has no missing values and all NA's for agriculture are in metropolitan regions, I think this can be done without biasing the results.

In [13]:
```
#pivot
empl_NUTS3_pivoted = empl_NUTS3.pivot(index = "geo\\time", columns = "nace_r2"
#set NA's to 0
empl_NUTS3_pivoted["A"][empl_NUTS3_pivoted["A"].isna()] = 0
#subtract columns
empl_NUTS3_pivoted["empl_ths"] = empl_NUTS3_pivoted["TOTAL"] - empl_NUTS3_pivo
```

Finally, I will add the employment data to the main dataset by joining on NUTS3.

In [14]:
```
#join with maindata:
maindata2 = pd.merge(maindata1, empl_NUTS3_pivoted, left_on = "NUTS3", right_
#shave off unnecessary columns:
maindata2 = maindata2[["NUTS3", "country", "NUTS2", "area_km2", "empl_ths"]]
maindata2
```

Out[14]:

	NUTS3	country	NUTS2	area_km2	empl_ths
0	DE111	DE	DE11	206.0	497.17
1	DE112	DE	DE11	618.0	219.76
2	DE113	DE	DE11	641.0	262.50
3	DE114	DE	DE11	642.0	118.33
4	DE115	DE	DE11	685.0	247.91
...	...	...	...	...	...
653	UKM62	UK	UKM6	7822.0	80.00
654	UKM63	UK	UKM6	14247.0	45.00

	NUTS3	country	NUTS2	area_km2	empl_ths
655	UKM64	UK	UKM6	3059.0	10.00
656	UKM65	UK	UKM6	989.0	10.00
657	UKM66	UK	UKM6	1466.0	14.00

Data shaping: education

Education data is only provided at the NUTS-2 level, and as absolute numbers of employed instead of frations. However, the first steps are similar to the previous two datasets: drop all unused years and regions. I will also drop the "not reported" category, since that is not an education level. Additionally, I can also get rid of the "sex" variable since there is no reason to expect the agglomeration effect to be different. I also select only the columns concerning all employed persons 15 and over, indicated by "Y_GE15".

In [15]:
```
#filter relevant NUTS2
educ_NUTS2 = educ_NUTS2[educ_NUTS2["geo\\time"].isin(NUTS2_selected)]
#filter out sex-specific columns
educ_NUTS2 = educ_NUTS2[educ_NUTS2["sex"] == "T"]
#filter out age-specific columns
educ_NUTS2 = educ_NUTS2[educ_NUTS2["age"] == "Y_GE15"]
#drop not reported:
educ_NUTS2 = educ_NUTS2[educ_NUTS2["isced11"] != "NRP"]
```

There are 16 missing values, evenly spread among the four categories. It turns out that there are only four regions that each have a missing value in all four categories.

In [16]:
```
educ_NUTS2["isced11"][educ_NUTS2[2013].isna()].value_counts()
```

Out[16]:
```
ED3_4    4
TOTAL    4
ED0-2    4
ED5-8    4
Name: isced11, dtype: int64
```

I will use the interpolate() method to try to fill in some of the values. I use backward interpolation (since the data are ordered 2019 to 1999 from left to right) and a limit of 2. This will successfully fill eight of the sixteen NA's. The others I will drop.

In [17]:
```
educ_NUTS2.loc[:, 2019:1999] = educ_NUTS2.loc[:, 2019:1999].interpolate(axis =
educ_NUTS2["isced11"][educ_NUTS2[2013].isna()].value_counts()
```

Out[17]:
```
ED3_4    2
TOTAL    2
ED0-2    2
ED5-8    2
Name: isced11, dtype: int64
```

In [18]:
```
#drop remaining NA's
educ_NUTS2 = educ_NUTS2[educ_NUTS2[2013].isna() == False]
#drop all years except 2013
educ_NUTS2 = educ_NUTS2[["isced11", "geo\\time", 2013]]
```

Next, I need to be able to create a proportion of employed with each education level for each NUTS-2 region. Again I will use a pivot. Afterards I will create the proportions of the population by dividing it by the total of persons employed within the region. I finish with renaming the education levels, since a including a variable named "ED0-2" will disrupt a regression due to the hyphen being interpreted as a minus sign.

In [19]:
```
#pivot education levels into columns
educ_pivoted = educ_NUTS2.pivot(index = "geo\\time", columns = "isced11", valu

#divide by total using a for loop:
for level in ["ED0-2", "ED3_4", "ED5-8"]:
    educ_pivoted[level] = educ_pivoted[level]/educ_pivoted["TOTAL"]

#drop total: we don't need it anymore
educ_pivoted = educ_pivoted.drop(columns = ["TOTAL"])

#rename columns
educ_pivoted.rename(columns = {"ED0-2":"ED0_2", "ED5-8":"ED5_8"}, inplace = T
```

Again, I merge with the main dataset.

In [20]:
```
maindata3 = pd.merge(maindata2, educ_pivoted, left_on = "NUTS2", right_on = "
maindata3
```

Out[20]:

	NUTS3	country	NUTS2	area_km2	empl_ths	ED0_2	ED3_4	ED5_8
0	DE111	DE	DE11	206.0	497.17	0.141150	0.536201	0.318846
1	DE112	DE	DE11	618.0	219.76	0.141150	0.536201	0.318846
2	DE113	DE	DE11	641.0	262.50	0.141150	0.536201	0.318846
3	DE114	DE	DE11	642.0	118.33	0.141150	0.536201	0.318846
4	DE115	DE	DE11	685.0	247.91	0.141150	0.536201	0.318846
...	...	...	...	...	...	...	...	...
653	UKM62	UK	UKM6	7822.0	80.00	0.130993	0.441353	0.419949
654	UKM63	UK	UKM6	14247.0	45.00	0.130993	0.441353	0.419949
655	UKM64	UK	UKM6	3059.0	10.00	0.130993	0.441353	0.419949
656	UKM65	UK	UKM6	989.0	10.00	0.130993	0.441353	0.419949
657	UKM66	UK	UKM6	1466.0	14.00	0.130993	0.441353	0.419949

658 rows × 8 columns

Data shaping: value added

Many of the steps I will take here are similar to what I have done previously. I will select only the columns denominated in euros, and only those concerning relevant NUTS-3 regions.

In [21]:
```
#select only EUR
va_NUTS3 = va_NUTS3[va_NUTS3["currency"] == "MIO_EUR"]
#select only relevant regions
va_NUTS3 = va_NUTS3[va_NUTS3["geo\\time"].isin(NUTS3_selected)]
```

There are around 10 "different" missing values here. I will take care of them before splitting the data by NACE classification. Again, I will try an interpolation with limit 2, before dropping what remains.

In [22]:
```
va_NUTS3["nace_r2"][va_NUTS3[2013].isna()].value_counts()
```

Out[22]:
```
O-Q    2
Name: nace_r2, dtype: int64
```

In [23]:
```
va_NUTS3["geo\\time"][va_NUTS3[2013].isna()].value_counts()
```

Out[23]:
```
DE501    1
DE502    1
Name: geo\time, dtype: int64
```

In [24]:
```
va_NUTS3.loc[:, 2018:1995] = va_NUTS3.loc[:, 2018:1995].interpolate(axis = 1,
va_NUTS3 = va_NUTS3[va_NUTS3[2013].notna()]
```

From here it is almost an exact repeat of the steps taken when processing the employment data: keep 2013, pivot the sectors and subtract agriculture from the total.

In [25]:
```
#drop unnecessary columns
va_NUTS3 = va_NUTS3[["nace_r2", "geo\\time", 2013]]
#pivot sectors to columns
va_pivoted = va_NUTS3.pivot(index = "geo\\time", columns = "nace_r2", values =
va_pivoted["A"][va_pivoted["A"].isna()] = 0
#subtract agriculture
va_pivoted["VA_mln"] = va_pivoted["TOTAL"] - va_pivoted["A"]
```

The final steps are dropping the last unnecessary columns and another join. Then, I will create additional columns to store the natural logs of employment density and labor productivity, after which the final version of the main dataset is ready.

In [26]:
```
#last column drop
va_pivoted = va_pivoted.iloc[:,15]
```

In [27]:
```
#last join
maindata4 = maindata3.merge(va_pivoted, left_on = "NUTS3", right_index = True)
#create logs of density and productivity
maindata4["logprod"] = np.log(maindata4["VA_mln"]/maindata4["empl_ths"])
maindata4["logdens"] = np.log(maindata4["empl_ths"]/maindata4["area_km2"])

maindata4
```

Out[27]:

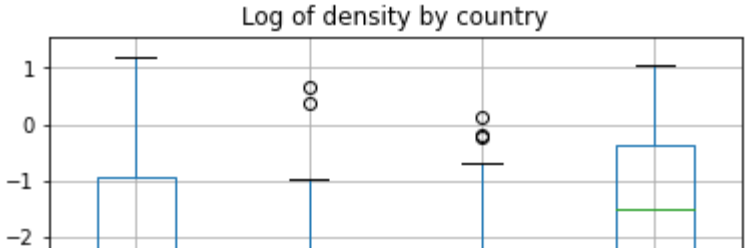
	NUTS3	country	NUTS2	area_km2	empl_ths	ED0_2	ED3_4	ED5_8	VA_mln	logprod
0	DE111	DE	DE11	206.0	497.17	0.141150	0.536201	0.318846	41431.64	4.422868
1	DE112	DE	DE11	618.0	219.76	0.141150	0.536201	0.318846	18305.95	4.422445
2	DE113	DE	DE11	641.0	262.50	0.141150	0.536201	0.318846	16535.70	4.143026
3	DE114	DE	DE11	642.0	118.33	0.141150	0.536201	0.318846	6711.16	4.038050
4	DE115	DE	DE11	685.0	247.91	0.141150	0.536201	0.318846	18330.19	4.303239
...	...	...	...	...	...	...	...	...	...	...
653	UKM62	UK	UKM6	7822.0	80.00	0.130993	0.441353	0.419949	5770.91	4.278558
654	UKM63	UK	UKM6	14247.0	45.00	0.130993	0.441353	0.419949	2424.47	3.986706
655	UKM64	UK	UKM6	3059.0	10.00	0.130993	0.441353	0.419949	580.51	4.061322
656	UKM65	UK	UKM6	989.0	10.00	0.130993	0.441353	0.419949	640.56	4.159758
657	UKM66	UK	UKM6	1466.0	14.00	0.130993	0.441353	0.419949	801.87	4.047889

658 rows × 11 columns

Analysis

In [28]:
```
maindata4.boxplot(column = "logdens", by = "country")
plt.suptitle('')
plt.title("Log of density by country")
plt.xlabel('')
plt.xticks([1,2,3,4],["Germany", "Spain", "Italy", "United Kingdom"])
```

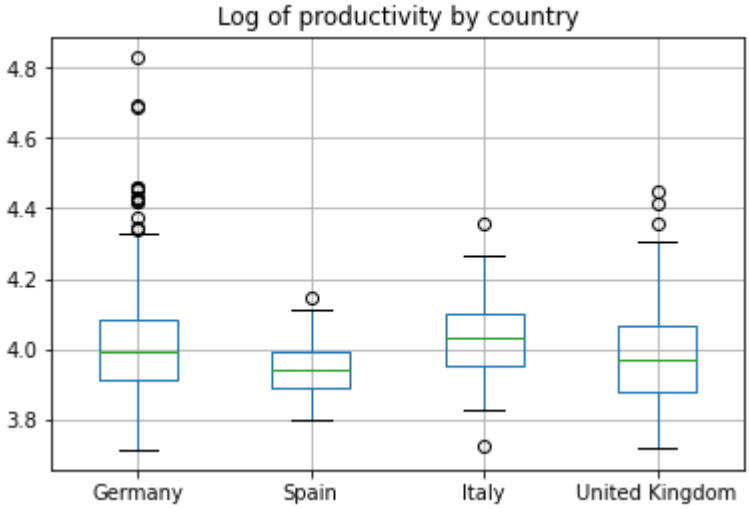
Out[28]:
```
([<matplotlib.axis.XTick at 0x1ed40a58610>,
  <matplotlib.axis.XTick at 0x1ed40a585e0>,
  <matplotlib.axis.XTick at 0x1ed40ad7c10>,
  <matplotlib.axis.XTick at 0x1ed41ea2700>],
 [Text(1, 0, 'Germany'),
  Text(2, 0, 'Spain'),
  Text(3, 0, 'Italy'),
  Text(4, 0, 'United Kingdom')])
```



Above, you see a boxplot describing the spread of the natural log of employment density, broken down by country. Spain and Italy have a small number of upward outliers. This is expected, as they are known for having a small number of relatively very large cities (places like Barcelona and Madrid in the case of Spain, and Rome and Milan in the case of Italy). Meanwhile, the UK has some downward outliers, which most likely represent sparse counties in the Scottish and Welsh Highlands.

In [29]:
```
maindata4.boxplot(column = "logprod", by = "country")
plt.suptitle('')
plt.title("Log of productivity by country")
plt.xlabel('')
plt.xticks([1,2,3,4],["Germany", "Spain", "Italy", "United Kingdom"])
```

Out[29]:
```
([<matplotlib.axis.XTick at 0x1ed425f37c0>,
  <matplotlib.axis.XTick at 0x1ed425f3790>,
  <matplotlib.axis.XTick at 0x1ed41ea2eb0>,
  <matplotlib.axis.XTick at 0x1ed42973880>],
 [Text(1, 0, 'Germany'),
  Text(2, 0, 'Spain'),
  Text(3, 0, 'Italy'),
  Text(4, 0, 'United Kingdom')])
```

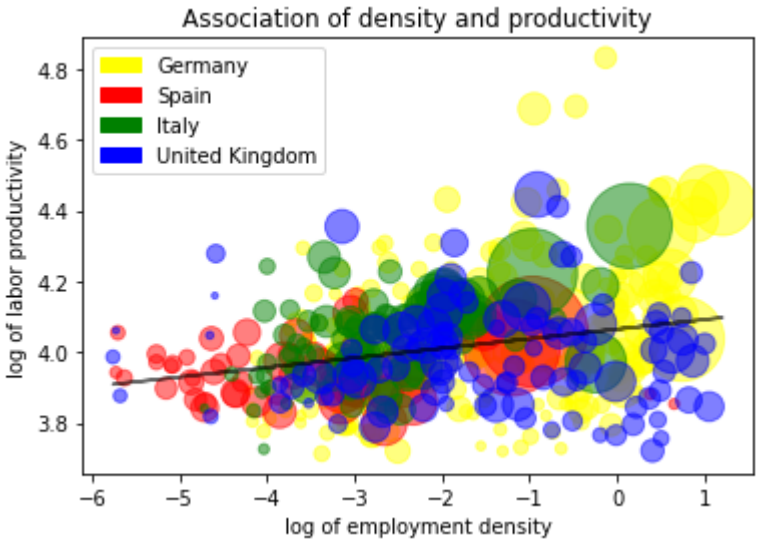


Meanwhile, the distributions for log labor productivity are a lot more compressed, and the country averages are a lot closer together than the average employment densities. Furthermore, there are a lot more upward outliers, especially in the UK and Germany.

In [30]:

```
#create color and label dictionaries
colors = {"DE": "yellow", "ES":"red", "IT":"green", "UK":"blue"}
labels = {"DE": "Germany", "ES":"Spain", "IT":"Italy", "UK":"United Kingdom"}

#run a loop drawing a scatterplot for each country
for c in maindata4.country.unique() :
    plt.scatter(x = "logdens",
                y = "logprod",
                c = colors[c],
                s = "empl_ths",
                data = maindata4[maindata4["country"] == c],
                alpha = 0.5)

#add a best fit line
b,a = np.polyfit(maindata4["logdens"], maindata4["logprod"], 1)
plt.plot(maindata4["logdens"], maindata4["logdens"]*b + a, color = "black", al

#create a list of custom legend entries using list comprehension
patches = [mpatch.Patch(color = colors[c], label = labels[c]) for c in maindat
#use custom legend contents and move the legend away from the top right
plt.legend(handles = patches, loc = "upper left")

#add labels
plt.xlabel("log of employment density")
plt.ylabel("log of labor productivity")
plt.title("Association of density and productivity")

#print naive OLS coefficient
b
```

Out[30]: 0.027045896856065076



While the naive OLS estimate indeed shows a positive relationship between density and productivity, the coefficient is a lot lower than it is in the original paper (0.027 versus 0.051). It seems that Germany and Italy are driving most of the correlation, with the UK almost exhibiting a hump-shaped pattern.

However, we cannot interpret this result as causal. As explained before, I will use area as an

instrumental variable. To test whether the instrument is still relevant in 2013, I check whether the reduced form (regressing the dependent variable directly on the instrument) yields a statistically significant result:

In [31]:
```
reducedform = smf.ols("logprod ~ area_km2", data = maindata4).fit()
reducedform.summary()
```

Out[31]:

OLS Regression Results

Dep. Variable:	logprod	**R-squared:**	0.014
Model:	OLS	**Adj. R-squared:**	0.013
Method:	Least Squares	**F-statistic:**	9.391
Date:	Tue, 26 Jan 2021	**Prob (F-statistic):**	0.00227
Time:	16:24:44	**Log-Likelihood:**	352.81
No. Observations:	658	**AIC:**	-701.6
Df Residuals:	656	**BIC:**	-692.6
Df Model:	1		
Covariance Type:	nonrobust		

	coef	**std err**	**t**	**P>|t|**	**[0.025**	**0.975]**
Intercept	4.0176	0.007	609.095	0.000	4.005	4.031
area_km2	-5.615e-06	1.83e-06	-3.065	0.002	-9.21e-06	-2.02e-06

Omnibus:	140.458	**Durbin-Watson:**	1.265
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	398.764
Skew:	1.047	**Prob(JB):**	2.57e-87
Kurtosis:	6.188	**Cond. No.**	4.30e+03

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 4.3e+03. This might indicate that there are

strong multicollinearity or other numerical problems.

Knowing that the effect is indeed there, and that it has the expected sign, I will now run the run the main regression (using White standard errors):

In [32]:
```
iv1 = IV2SLS.from_formula("logprod ~ ED0_2 + ED3_4 + ED5_8 + C(country) + [log
                          data = maindata4).fit(cov_type = "robust")

#extract the confidence interval for later use
confint1 = iv1.conf_int()

iv1
```

Out[32]:

IV-2SLS Estimation Summary

Dep. Variable:	logprod	**R-squared:**	0.1806
Estimator:	IV-2SLS	**Adj. R-squared:**	0.1718
No. Observations:	658	**F-statistic:**	7.802e+05
Date:	Tue, Jan 26 2021	**P-value (F-stat)**	0.0000
Time:	16:24:44	**Distribution:**	chi2(8)
Cov. Estimator:	robust		

Parameter Estimates

	Parameter	Std. Err.	T-stat	P-value	Lower CI	Upper CI
C(country)[DE]	8.3083	2.5413	3.2693	0.0011	3.3275	13.289
C(country)[ES]	7.8478	2.5451	3.0835	0.0020	2.8595	12.836
C(country)[IT]	8.2440	2.5528	3.2294	0.0012	3.2406	13.247
C(country)[UK]	8.0510	2.5089	3.2089	0.0013	3.1336	12.968
ED0_2	-3.7670	2.5736	-1.4637	0.1433	-8.8112	1.2772
ED3_4	-4.8465	2.5616	-1.8920	0.0585	-9.8672	0.1742
ED5_8	-3.3670	2.5237	-1.3341	0.1822	-8.3134	1.5795
logdens	0.0092	0.0062	1.4885	0.1366	-0.0029	0.0213

Endogenous: logdens
Instruments: area_km2
Robust Covariance (Heteroskedastic)
Debiased: False

id: 0x1ed42857bb0

Sensitivity analysis

As a first sensitivity check, I will respecify the model, but with NUTS-2 fixed effects instead of country-level fixed effects. Since the education data is at the NUTS-2 level as well, they can be omitted in this specification.

In [33]:
```
iv2 = IV2SLS.from_formula("logprod ~ C(NUTS2) + [logdens ~ area_km2]",
                          data = maindata4).fit(cov_type = "robust")

#again extract the confidence intervals
confint2 = iv2.conf_int()
```

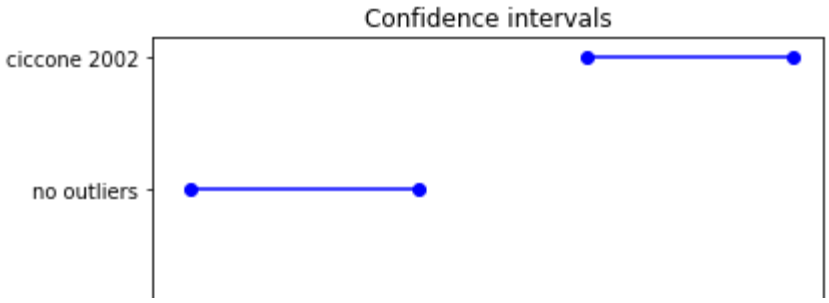
My second sensitivity check will be to re-run the main IV regression, but with all productivity and density outliers removed. This reduces the number of observations to 640.

In [34]:
```
#define cutoff thresholds for outliers
prod_upper = maindata4["logprod"].quantile(0.75) + 1.5*(maindata4["logprod"].
dens_upper = maindata4["logdens"].quantile(0.75) + 1.5*(maindata4["logdens"].
prod_lower = maindata4["logprod"].quantile(0.25) - 1.5*(maindata4["logprod"].
dens_lower = maindata4["logdens"].quantile(0.25) - 1.5*(maindata4["logdens"].

#filter outliers
no_outliers = maindata4[maindata4["logprod"] < prod_upper]
no_outliers = no_outliers[no_outliers["logprod"] > prod_lower]
no_outliers = no_outliers[no_outliers["logdens"] < dens_upper]
no_outliers = no_outliers[no_outliers["logdens"] > dens_lower]

#rerun regression
iv3 = IV2SLS.from_formula("logprod ~ ED0_2 + ED3_4 + ED5_8 + C(country) + [log
                          data = no_outliers).fit(cov_type = "robust")

#extract confint
confint3 = iv3.conf_int()
```

Below, I plot the 3 different confidence intervals, along with the confidence interval from the original paper. I use code adapted from this stackoverflow answer.

In [35]:
```
#Define the confidence interval found by the original paper
ciccone_upper = 0.0455 + 2*0.00507
ciccone_lower = 0.0455 - 2*0.00507

ci_dict = {}
ci_dict['category'] = ['main','NUTS2 dummies','no outliers', "ciccone 2002"]
ci_dict['lower'] = [confint1.iloc[-1,1], confint2.iloc[-1,1], confint3.iloc[-1
ci_dict['upper'] = [confint1.iloc[-1,0], confint2.iloc[-1,0], confint3.iloc[-1
cis = pd.DataFrame(ci_dict)

for lower,upper,y in zip(cis['lower'],cis['upper'],range(len(cis))):
    plt.plot((lower,upper),(y,y),'ro-',color = "blue")
plt.yticks(range(len(cis)),list(cis['category']))
plt.title("Confidence intervals")
```

Out[35]: Text(0.5, 1.0, 'Confidence intervals')



Clearly, my estimates are different from those in the orginal paper, no matter the specification.

Discussion and conclusion

In conclusion, my replication indicates that net agglomeration effects could be a lot lower than estimated in Ciccone (2002), and does not even conclusively reject the null hypothesis. There are several possible explanations for this. The first is simply that the exlcusion of French regions from the data is what causes the difference. After all, France is a country known for having a small number of prosperous metropolitan areas and rural areas that do poorly by comparison. Another explanation is that something has changed structurally within the European economy over the past 30 years, reducing the strength of agglomeration effects. Since the estimate measures net agglomeration, this could also be driven by things like a structural increase in congestion externalities.

Finally, we have no reason to expect the local elasticity of productivity to be constant along all possible densities. It could be that, if there is a different density spread now than there was 30 years ago, that drives the difference in estimates. If this is true, it is a very intuitive explanation: employers cluster to take advantage of agglomeration effects, and they stop this "arbitrage" once the net effects approach zero.

There are several ways to improve this project in later versions. The first, and most obvious one, is to find employment data for France and include it in the analysis. A second way is to find better data on education, such that it can be controlled for at the NUTS-3 level instead of at the NUTS-2 level. A final way is to control for spillover effects, like in the original paper. However, this had a negligible impact and I could not find an easy way to implement it.

Bibliography

- Bartels, C.; Jaeger, S.; Obergruber, N. (2020). Long-Term Effects of Equal Sharing: Evidence from Inheritance Rules for Land. NBER Working Paper No. w28230.
- Ciccone, A. (2002). Agglomeration effects in Europe. *European Economic Review* 46, 213-227. https://doi.org/10.1016/S0014-2921(00)00099-4
- Dell, M. (2010). The Persistent Effects of Peru's Mining *Mita*. *Econometrica* 78(6), 1862-1903. https://doi.org/10.3982/ECTA8121
- Dorosh, P.; Thurlow, J. (2014). Can Cities or Towns Drive African Development? Economywide Analysis for Ethiopia and Uganda. *World Development* (63), 113-123. https://doi.org/10.1016

/j.worlddev.2013.10.014

- Jedwad, R.; Vollrath, D. (2015). Urbanization without growth in historical perspective. *Explorations in Economic History* (58C), 1-21. https://doi.org/10.1016/j.eeh.2015.09.002
- Rosenthal, S.; Strange, W. (2004). Evidence on the Nature and Sources of Agglomeration Economies. *Handbook of Regional and Urban Economics* (4), 2119-2171. https://doi.org/10.1016/S1574-0080(04)80006-3