Appendix A

Data Management and Gravity Estimation

A.1 Creating the Database for the Gravity Analysis

As discussed in Chapter 2, for the gravity model to be estimated, it is necessary to merge databases from different sources. Moreover, to make the fixed effects comparable over time and across countries two assumptions need to hold: first, the database must be subset to take into account only those countries importing a certain good in every year; second, the exporters must ship their products to an overlapping group of importer countries. Hence, hereafter will be shown step by step how to deal with either these aspects. The software used is Stata version 14, so commands might not perfectly work for previous version, especially for those developed before version 13.

Starting from bilateral trade flows data, the Comtrade database can have many different formats, the one chosen for the SITC revision 2 two-digits industries features the following variables: *commoditycode*, *commoditydescription*, *netweightkg*, *classification*, *partneriso*3, *partnercode*, *value*[1], *year*, *reporteriso*3, *reportercode*. Not all the variables are meaningful for the analysis, those that must be kept will also be renamed for clarity as follows:

```
drop commoditydescription netweightkg ///
     classification partnercode reportercode
```

[1]The value of the bilateral trade flows used refers to imports

```
rename commoditycode HS2_product
rename partneriso3 exporter_iso3
rename reporteriso3 importer_iso3

order value HS2_product exporter_iso3 importer_iso3 year
```

The Comtrade database will afterwards be displayed as follows:

Value of Import	HS2 Commodity Code	Exporter	Importer	Year
20320440	0	WLD	DZA	1984
2058614	0	BEL	DZA	1984
...	...	...	...	...

This database features 244 exporters, 209 importers, and 100 two-digits industries, moreover it is constructed such to not report zero-value trade flows.

The first aspect to account for is the presence of "Other Countries" and "Free Zones" which can be easily dropped by the following command:

```
drop if importer_iso3 == . | exporter_iso3 == .
```

For what concern some specific countries it is necessary to aggregate the value of bilateral trade flows. These countries are Germany, due to historical division between East and West, Belgium and Luxembourg, for which data on gravity variables are considered together, Yemen and Panama. Such task can be easily performed as follows:

```
sort importer_iso3 exporter_iso3 year HS2_product

replace importer_iso3 = "DEU" if importer_iso3 == "DDR"
collapse (sum) value, by(year importer_iso3 exporter_iso3 HS2_product)
replace exporter_iso3 = "DEU" if exporter_iso3 == "DDR"
drop if exporter_iso3 == "DEU" & importer_iso3 == "DEU"

replace importer_iso3 = "BEL" if importer_iso3 == "LUX"
replace exporter_iso3 = "BEL" if exporter_iso3 == "LUX"
drop if exporter_iso3 == "BEL" & importer_iso3 == "BEL"

replace exporter_iso3 = "YEM" if exporter_iso3 == "YDM"
replace importer_iso3 = "YEM" if importer_iso3 == "YDM"
```

```
drop if exporter_iso3 == "YEM" & importer_iso3 == "YEM"

replace importer_iso3 = "PAN" if importer_iso3 == "PCZ"
collapse (sum) value, by(year importer_iso3 exporter_iso3 HS2_product)
replace exporter_iso3 = "PAN" if exporter_iso3 == "PCZ"
```

The Comtrade database is ready to be merged, hence it is sufficient to make the CEPII's database on gravity variables, and the Larch's database on preferential trade agreements comparable, which means to rename those variables that are in common across databases. The merge command, keeping open in Stata the file with trade flows, need to be specified as follows for the CEPII's database:

```
merge m:1 importer_iso3 exporter_iso3 using "file_path"
drop if year == .
drop _merge
```

While for the RTA database in this way:

```
merge m:1 year importer_iso3 exporter_iso3 using "file_path"
drop if value == .
drop _merge
```

The reason is that RTAs might change year by year, while gravity variables are constant over time. As a last step is required to get rid of all the missing values, and log-linearize the numeric variables:

```
drop if importer_iso3 == .
drop if exporter_iso3 == .
drop if border == .

destring dist, replace dpcomma ignore(".")
destring distcap, replace dpcomma ignore(".")
destring distw, replace dpcomma ignore(".")
destring distwces, replace dpcomma ignore(".")

generate ln_value = ln(value)
generate ln_gdp_importer = ln(gdp_importer)
generate ln_gdp_exporter = ln(gdp_exporter)
```

```
generate ln_dist = ln(dist)
generate ln_distcap = ln(distcap)
generate ln_distw = ln(distw)
generate ln_distwces = ln(distwces)
```

The command *destring* along with the option *replacedpcommaignore*(".") can be used to convert those numeric variables that were imported in Stata featuring a decimal comma separator. The conversion substitute the comma with a dot, in order to make Stata recognize the character in the cell as a numeric value and not a string. After this operation is completed it is possible to apply any mathematical function to the destringed variable.

The database now can be subset to satisfy the two aforementioned assumption. With this loop command:

```
codebook importer_iso3

egen group = group(importer_iso3)
sort group
foreach i of num 1/n {
   tabulate year if group == `i'
   }
```

it is possible first to evaluate the number, n, of importers, and second to visualize whether any of these countries has some missing values at different point in time. In this case, after the correction, the number of importers shrank to 61. A further correction is done to take into account those importers for which there is a small number of observation, in percentage relatively to the other countries, so that only 55 importers are finally left.

For the exporter condition to be met, it is possible to follow the same approach previously adopted, and then drop all those exporters than do not ship their products towards the same set of countries:

```
codebook exporter_iso3

egen group = group(exporter_iso3)
sort group
order group, after(exporter_iso3)
foreach i of num 1/n {
```

```
codebook importer_iso3 if group == `i'
}
```

This will leave 104 exporters in the sample, along with 66 industries.

To speed up the subsequent estimation of the gravity model it is possible to generate importer and exporter fixed effects, that will then estimated as importer- and exporter-industry-year fixed effects:

```
levelsof importer, clean local(imp)
foreach i in `imp'{
        gen importer_`i' = importer == "`i'"
}
levelsof exporter, clean local(exp)
foreach i in `exp'{
        gen exporter_`i' = exporter == "`i'"
```

A.2 Gravity Estimation

The identification of the fixed effects of the gravity model, following Hanson et al. (2016), requires to estimate equation (2.1) by mean of OLS. To not incur in the curse of dimensionality, the database can be split in a number of industry-year sub-databases on which to perform the OLS. Such operation can be accomplished in two steps. First, it is necessary to subset the main database:

```
levelsof importer, clean local(imp)
foreach i in `imp'{
        gen importer_`i' = importer == "`i'"
}
levelsof exporter, clean local(exp)
foreach i in `exp'{
        gen exporter_`i' = exporter == "`i'"
}
```

Second, a column reporting the set of fixed effect can be generated at which will be attached the estimated values of the regression, that must be carried out for each industry-year sub-database:

```
gen fixed_effect = ""

local i = 1
foreach var of varlist importer_ARG-exporter_ZWE {
replace fixed_effect = "`var'" in `i'
local i = `i' + 1
}

drop importer_USA exporter_USA

regress ln_value border lang lang_9perc colony common_colonizer
///
        curr_colonial after45_colonial same_country rta ln_distw ///
            importer_* exporter_*, robust

gen yearX = .
foreach var of varlist importer_ARG-exporter_ZWE {
replace yearX = _b[`var'] if fixed_effect == "`var'"
}

replace yearX = 0 if fixed_effect == "importer_USA"
replace yearX = 0 if fixed_effect == "exporter_USA"
```

The choice to drop the importer and exporter fixed effects associated to the US is made to allow for a common industry-year normalization between the different sub-databases, which derives from the presence of multicollinearity in the estimation of the model. Taking the US as the benchmark makes necessary to manually input the zero value in the cells of both fixed effects referring to the country.

It is possible, at this point, to manually extract the list of fixed effects and create a specific database on which the calculate the exporter capability A_{ist} and the demand shock D_{idt}.

A.3 Main Empirical Specification

In this part of the appendix are reported the scripts used to carry out the empirical analysis, hence the estimation of equations (2.8) and (2.9). The two main components to retrieve are the export capability and the effective foreign demand[2]. Once the databases with the export capability and demand shock are stored, in order to define the effective foreign demand, it is necessary to first determine the weights attributed:

```
bysort year exporter HS2_product: egen tot_exp_ist = total(value)
gen omega_isdt = value/tot_exp_ist
```

This procedure must be applied to the initial, non modified, Comtrade database because this would generate the true weights, that is weights not biased by the exclusion of bilateral trade flows. This database containing the weights needs to be merged with the databases containing A_{ist} and D_{idt}[3]:

```
merge m:1 exporter HS2_product year using "folder_path\dataset_A.dta"
drop _merge
merge m:1 importer HS2_product year using "folder_path\dataset_D.dta"
drop _merge
sort HS2_product
```

At this point, it is possible to generate the effective foreign demand:

```
gen Dw_idt = omega_isdt*D_idt
```

The resulting database can be exploited to perform the estimation of equations (2.8) and (2.9).

[2]The export capability measure, A_{ist}, and the demand shock, D_{idt}, were calculated using excel and then stored in two separated files.

[3]Note the three databases must be comparable as in the previously specified case with the gravity model variables

Appendix B

Tables

Table B.1: **List of Importing Countries (ISO-ALPHA 3 Code)**

Algeria (DZA)	Jamaica (JAM)
Argentina (ARG)	Japan (JPN)
Australia (AUS)	Kenya (KEN)
Austria (AUT)	Malaysia (MYS)
Barbados (BRB)	Mauritius (MUS)
Belgium & Luxembourg (BEL)	Morocco (MAR)
Bolivia (BOL)	Netherlands (NLD)
Brazil (BRA)	New Zealand (NZL)
Canada (CAN)	Norway (NOR)
Chile (CHL)	Oman (OMN)
China (CHN)	Peru (PER)
Colombia (COL)	Philippines (PHL)
Cyprus (CYP)	Poland (POL)
Denmark (DNK)	Portugal (PRT)
Ecuador (ECU)	Rep. of Korea (KOR)
Egypt (EGY)	Singapore (SGP)
Finland (FIN)	Spain (ESP)
France (FRA)	Sweden (SWE)
Germany (DEU)	Switzerland (CHE)
Greece (GRC)	Thailand (THA)
Hong Kong (HKG)	Trinidad & Tobago (TTO)
Hungary (HUN)	Tunisia (TUN)
India (IND)	Turkey (TUR)
Indonesia (IDN)	United Kingdom (GBR)
Ireland (IRL)	United States of America (USA)
Island (ISL)	Uruguay (URY)
Israel (ISR)	Venezuela (VEN)
Italy (ITA)	

Table B.2: **List of Exporting Countries (ISO-ALPHA 3 Code)**

Albania (ALB)	Ethipia (ETH)	Madagascar (MDG)	Rep. of Korea (KOR)
Algeria (DZA)	Fiji (FJI)	Malaysia (MYS)	Russia (RUS)
Argentina (ARG)	Finland (FIN)	Mali (MLI)	Rwanda (RWA)
Australia (AUS)	France (FRA)	Malta (MLT)	Saudi Arabia (SAU)
Austria (AUT)	Gabon (GAB)	Mauritania (MRT)	Sierra Leone (SLE)
Bahamas (BHS)	Germany (DEU)	Mauritius (MUS)	Singapore (SGP)
Bahrain(BHR)	Ghana (GHA)	Mexico (MEX)	Slovenia (SLV)
Barbados (BRB)	Greece (GRC)	Mongolia (MNG)	Spain (ESP)
Belgium & Luxembourg (BEL)	Guatemala (GTM)	Morocco (MAR)	Sri Lanka (LAK)
Bolivia (BOL)	Guinea (GIN)	Mozambique (MOZ)	Sweden (SWE)
Brazil (BRA)	Guyana (GUY)	Nepal (NPL)	Switzerland (CHE)
Brunei Darussalam (BRN)	Haiti (HTI)	Netherlands (NLD)	Syria (SYR)
Cambodia (KHM)	Honduras (HND)	New Caledonia (NCL)	Tanzania (TZA)
Canada (CAN)	Hong Kong (HKG)	New Zealand (NZL)	Thailand (THA)
Chile (CHL)	Hungary (HUN)	Nicaragua (NIC)	Trinidad & Tobago (TTO)
China (CHN)	India (IND)	Niger (NER)	Tunisia (TUN)
Colombia (COL)	Indonesia (IDN)	Nigeria (NGA)	Turkey (TUR)
Comoros (CMR)	Iran (IRN)	Norway (NOR)	Uganda (UGA)
Congo (COG)	Ireland (IRL)	Oman (OMN)	United Arab Emirates (ARE)
Costa Rica (CRI)	Island (ISL)	Pakistan (PAK)	United Kingdom (GBR)
Côte d'Ivoire (CIV)	Italy (ITA)	Panama (PAN)	United States of America (USA)
Cyprus (CYP)	Jamaica (JAM)	Papua New Guinea (PNG)	Uruguay (URY)
Denmark (DNK)	Japan (JPN)	Peru (PER)	Venezuela (VEN)
Ecuador (ECU)	Jordan (JOR)	Philippines (PHL)	Yemen (YEM)
Egypt (EGY)	Kenya (KEN)	Poland (POL)	Zambia (ZMB)
Ethiopia (ETH)	Lebanon (LBN)	Portugal (PRT)	Zimbabwe (ZWE)