SAS Visual Analytics 6.4

Adding Belgium Regions

David Demeyer

Contents

1. Information

This document explains how to load Belgium geographical data into SAS Visual Analytics 6.4 in order to provide auto mapping for Geo Bubble Maps and Geo Region Maps. You can add different geographical levels:

1. Country
2. Region (Vlaanderen, …)
3. Province
4. Arrondisment
5. Municipality
6. Municipality Section (deelgemeenten)
7. Statistic Sector

Please note that:

- It is not supported by SAS Technical Support.
- Only tested on SAS Visual Analytics 6.4.
- Reports and Explorations based on this method are not guaranteed to migrate correctly.

This document would not be possible without the input and assistance of Andrew Christian and Falko Schulz. And is based on the '[How to load custom boundaries/polygon data into SAS Visual Analytics 6.4]' article of the SASPedia, written by Falko Schulz.

2. Overview of steps

General

1. Define Custom Maps Library.
2. Define Geographical Data Library.
3. Back-up of Lookup Tables.
4. Delete existing Data of Belgium.

For every Geographical Level

1. Define Macro Variables.
2. Import Shape File.
3. Add DENSITY Variable.
4. Add Custom Region Name to ATTRLOOKUP.
5. Create Custom Regions Dataset.
6. Add Custom Regions Dataset to ATTRLOOKUP.
7. Add Custom Region Name to CENTLOOKUP.
8. Add Custom Region Data to CENTLOOKUP.

3. General Steps

3.1 Define Custom Maps Library

In order to load any map data in SAS Visual Analytics, you will need to define a new custom maps library.

The location does not matter as long as the Workspace Server instance has sufficient access to it. As map data is loaded via SAS Stored Processes - we are going to add a new pre-assigned library in the following file:

<config-dir>/Lev1/SASApp/appserver_autoexec_usermods.sas:

```
libname MAPSCSTM "/opt/sas/maps";
```

Remember, to restart the object spawner so your autoexec changes take affect!

3.2 Define Geographical Data Library

SAS Visual Analytics uses centroid lookup tables in order to map regional names (e.g. Brussels) to ISO codes. These codes are the **unique** identifiers of your regions. So in order for VA to recognize your new regions we will need to add extra rows to the lookup tables.

The lookup tables are stored in the **valib** library definied in metadata.

libname valib "<config-dir>/Lev1/SASApp/Data/valib";

This library should contain the following two tables:

- ATTRLOOKUP (list of all regions on various levels and their attributes such as ISO code)
- CENTLOOKUP (list of all regions and their central lat/lng point)

```
libname valib "/sas/config/Lev1/SASApp/Data/valib";
```

3.3 Back-up of Lookup Tables

Before you make any changes to the Lookup Tables, it's a best practice to make a back-up of the original datasets.

```
data valib.atrrlookup_backup;
        set valib.attrlookup;
run;

data valib.centlookup_backup;
        set valib.centlookup;
run;
```

3.4 Delete existing Data of Belgium.

The make sure you don't have any duplicate data about Belgium, we are going to remove the already existing data.

```
data valib.attrlookup;
        set valib.attrlookup;
        where id NOT LIKE "BE%";
run;

data valib.centlookup;
        set valib.centlookup;
        where id NOT LIKE "BE%";
run;
```

Other reasons to delete the already existing data of Belgium:

- More detailed shape files of Belgium.
- You want to rename some areas: Belgie/Belgique instead of Belgium without changing the lookup tables.

4. Steps for one specific Geographical Level (eg. Municipalities)

4.1 Define Macro Variables

In order to make code more generic we are going to define some metadata (macro variables) to define our new regions:

```
%let REGION_LABEL=Belgium Municipality;
%let REGION_PREFIX=XM;
%let REGION_ISO=924;
%let PROVINCE_LABEL=Custom Municipality;
%let PROVINCE_DATASET=MAPSCSTM.CUSTOM_MUN1;
```

Make sure REGION_PREFIX and REGION_ISO doesn't conflict with any other country ISO 2-Letter Code or country ISO Code in your Lookup Tables.

TIP: Make sure the name of your datasets ends with the number 1.

4.2 Import Shape File.

The shape file is a common standard for representing geospatial vector data. SAS Visual Analytics is utilizing a SAS Stored Process stored in SAS Metadata* in order to load polygon data. The STP requires the map data set to contain the following columns:

- ID (Character - the unique identifier of your region)
- X (Number - longitude address)
- Y (Number - latitude address)

The following example imports the shape file (containing the municipalities) into the SAS dataset MunicipalityMap.

```
PROC MAPIMPORT  DATAFILE="/opt/sas/maps/AD_2_Municipality.shp"
                OUT=MAPSCSTM.MunicipalityMap;
                ID NISCODE;
RUN;
```

* /System/Applications/SAS Web Infrastructure Platform/Web Infrastructure Platform 9.4/Utilities/GeographicalMappingService

4.3 Add DENSITY Variable

Sometimes polygon data can be massive and contain far too much detail than you would ever need in order to render a basic map within VA. The following PROC GREDUCE creates a new column DENSITY (a number from 0-6) which you can use to filter the data if required.

```
PROC GREDUCE DATA=MAPSCSTM.MunicipalityMap OUT=MAPSCSTM.MunicipalityMap;
        ID NISCODE;
RUN;
```

4.4 Add Custom Region Name to ATTRLOOKUP

In order to support lookup of geographical data/values by name, we are going to use our new geographical data to add a list of all region names including their unique identifier to ATTRLOOKUP.

To distinguish our custom region from other regions in the lookup table, we are going to add a new region/level (ex. Belgium Municipality) to the ATTRLOOKUP table.

```
proc sql;
 insert into valib.attrlookup
 values (
  "&REGION_LABEL.",          /* IDLABEL=State/Province Label */
  "&REGION_PREFIX.",         /* ID=SAS Map ID Value */
  "&REGION_LABEL.",          /* IDNAME=State/Province Name */
  "",                        /* ID1NAME=Country Name */
  "",                        /* ID2NAME */
  "&REGION_ISO.",            /* ISO=Country ISO Numeric Code */
  "&REGION_LABEL.",          /* ISONAME */
  "&REGION_LABEL.",          /* KEY */
  "",                        /* ID1=Country ISO 2-Letter Code */
  "",                        /* ID2 */
  "",                        /* ID3 */
  "",                        /* ID3NAME */
  0                          /* LEVEL (0=country level) */
 );
;quit;
```

4.5 Create Custom Regions Boundary Dataset

This step is probably the most difficult step as it's very depending on the structure of your input data (Shape File). **It's important that you populate all columns below (see KEEP statement in code snippet).**

You can have only 1 name of each region in SAS Visual Analytics. Sounds logical, but this can be difficult as we are a multilingual country. In my code snippet (code in red) you will see one way of working. In Flanders I have used the Dutch name of the region, in the Wallonia the French name and in Brussels both.

TIP: Visual Analytics can't handle special characters in region names. So, you have to change *'België'* into *'Belgie'*.

As you all know: the ID and IDNAME should be unique. So you can't use Leuven as a municipality and in another level as an arrondissement. I have used the region name as-is for the municipality-level, and on the arrondissement-level I have added the word: 'Arr. ' before the name. In that way you have 'Leuven' and 'Arr. Leuven'.

Important: Maximum length of the ID column ($15.)

Tip: You can filter on the DENSITY. A DENSITY of 3 and lower is acceptable.

```
data &PROVINCE_DATASET.;
 SET MAPSCSTM.MunicipalityMap;
 ID = compress("&REGION_PREFIX.-" || put(NISCODE,8.));

 IF substr(left(niscode),1,1) in ('7','4','3','1') THEN DO;
        IF name_dut ne ' ' THEN IDNAME=name_dut;
 END;
 ELSE IF substr(left(niscode),1,1) in ('9','8','6','5') THEN DO;
        IF name_fre ne ' ' THEN IDNAME=name_fre;
        ELSE IF name_ger ne ' ' THEN IDNAME=name_ger;
 END;
 ELSE IF substr(left(niscode),1,1) = '2' THEN DO;
   code = substr(left(niscode),1,2);
        IF code in ('24', '23') THEN IDNAME = name_dut;
        ELSE IF code = '25' THEN IDNAME = name_fre;
        ELSE IF code = '21' THEN IDNAME = catx(' / ',name_fre,name_dut);
 END;

 LONG = X;
 LAT = Y;
 ISO = "&REGION  ISO.";
 RESOLUTION = 1;
 LAKE = 0;
 ISOALPHA2 = "&REGION_PREFIX.";
 AdminType = "regions";
 WHERE DENSITY <= 3;
 keep ID SEGMENT IDNAME LONG LAT X Y ISO DENSITY RESOLUTION LAKE ISOALPHA2
AdminType;
run;
```

If needed, you can also add this code to remove the special characters in the region names.

```
 IDNAME = TRANWRD(IDNAME,'é','e');
 IDNAME = TRANWRD(IDNAME,'è','e');
 IDNAME = TRANWRD(IDNAME,'ë','e');
 IDNAME = TRANWRD(IDNAME,'ê','e');
 IDNAME = TRANWRD(IDNAME,'ä','a');
 IDNAME = TRANWRD(IDNAME,'â','a');
 IDNAME = TRANWRD(IDNAME,'à','a');
 IDNAME = TRANWRD(IDNAME,'á','a');
 IDNAME = TRANWRD(IDNAME,'ï','i');
 IDNAME = TRANWRD(IDNAME,'ô','o');
 IDNAME = TRANWRD(IDNAME,'ö','o');
 IDNAME = TRANWRD(IDNAME,'ü','u');
 IDNAME = TRANWRD(IDNAME,'û','u');
 IDNAME = TRANWRD(IDNAME,'ú','u');
 IDNAME = TRANWRD(IDNAME,'ù','u');
```

4.6 Add Custom Regions Data to ATTRLOOKUP

The ATTRLOOKUP lookup table provides a list of all geographical levels and the regions. We are going to use our main boundary file to extract a distinct list of each custom region (ex. All the municipalities).

```
proc sql;
 insert into valib.attrlookup
 select distinct
  IDNAME,                          /* IDLABEL=State/Province Label */
  ID,                              /* ID=SAS Map ID Value */
  IDNAME,                          /* IDNAME=State/Province Name */
  "&REGION_LABEL.",                /* ID1NAME=Country Name */
  "",                              /* ID2NAME */
  "&REGION_ISO.",                  /* ISO=Country ISO Numeric Code */
  "&REGION_LABEL.",                /* ISONAME */
  IDNAME || "|&REGION_LABEL.",     /* KEY */
  "&REGION_PREFIX.",               /* ID1=Country ISO 2-Letter Code */
  "",                              /* ID2 */
  "",                              /* ID3 */
  "",                              /* ID3NAME */
  1                                /* LEVEL (1=state level) */
 from &PROVINCE_DATASET.
;quit;run;
```

4.7 Add Custom Region Name to CENTLOOKUP

The CENTLOOKUP lookup table provides the link between the unique identifiers of your regions and the actual polygon table.

```
proc sql;
  insert into valib.centlookup
  select distinct
     "&PROVINCE_DATASET." as mapname,
     "&REGION_PREFIX." as ID,
     avg(x) as x,
     avg(y) as y
  from &PROVINCE_DATASET.;
;quit;run;
```

4.8 Add Custom Region Data to CENTLOOKUP

```
proc sql;
  insert into valib.centlookup
  select distinct
     "&PROVINCE_DATASET." as mapname,
     ID as ID,
     avg(x) as x,
     avg(y) as y
  from &PROVINCE_DATASET.
  group by id;
;quit;run;
```

5. Create Test Data (Optional)

For testing purposes you may want to create a simple table containing all regional names and a few measures. This table will have the exact and correct mapping for all regions are we are going to create this table from our original shape file. Real data may later differ from the regional names dependent on data quality - so it's often good to have one test table to confirm that all polygons are rendered correctly.

```
proc sql;
  create table &PROVINCE_DATASET._TEST as
  select distinct
     ID as ID,
     IDNAME as NAME
  from &PROVINCE_DATASET.;
  create table &PROVINCE_DATASET._TEST as
  select *,
     round(ranuni(1) * 10000) as population,
     round(ranuni(1) * 100000) as avg_income format=dollar20.0
  from &PROVINCE_DATASET._TEST
  group by ID, NAME
  order by ID, NAME
;quit;run;
```

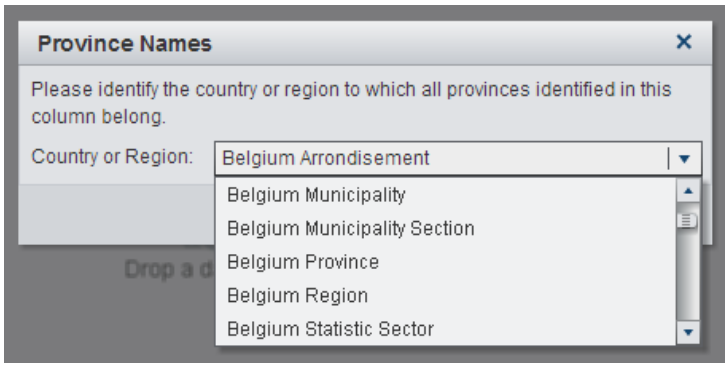
6. Load and Use New Regions

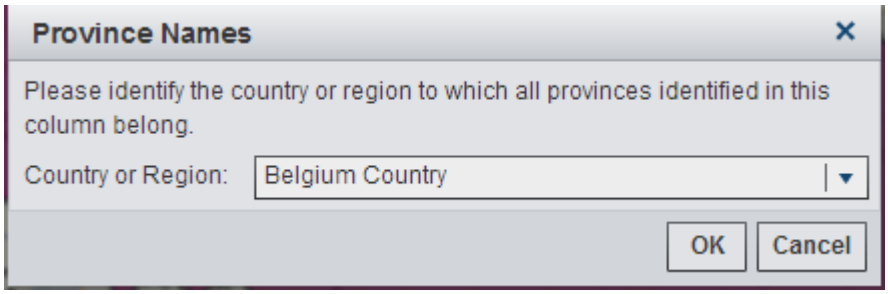
Now that we added new regions to our lookup tables it's time to load these into SAS Visual Analytics. As VA caches all lookup tables on the mid-tier during startup, we need to restart all the SAS services:

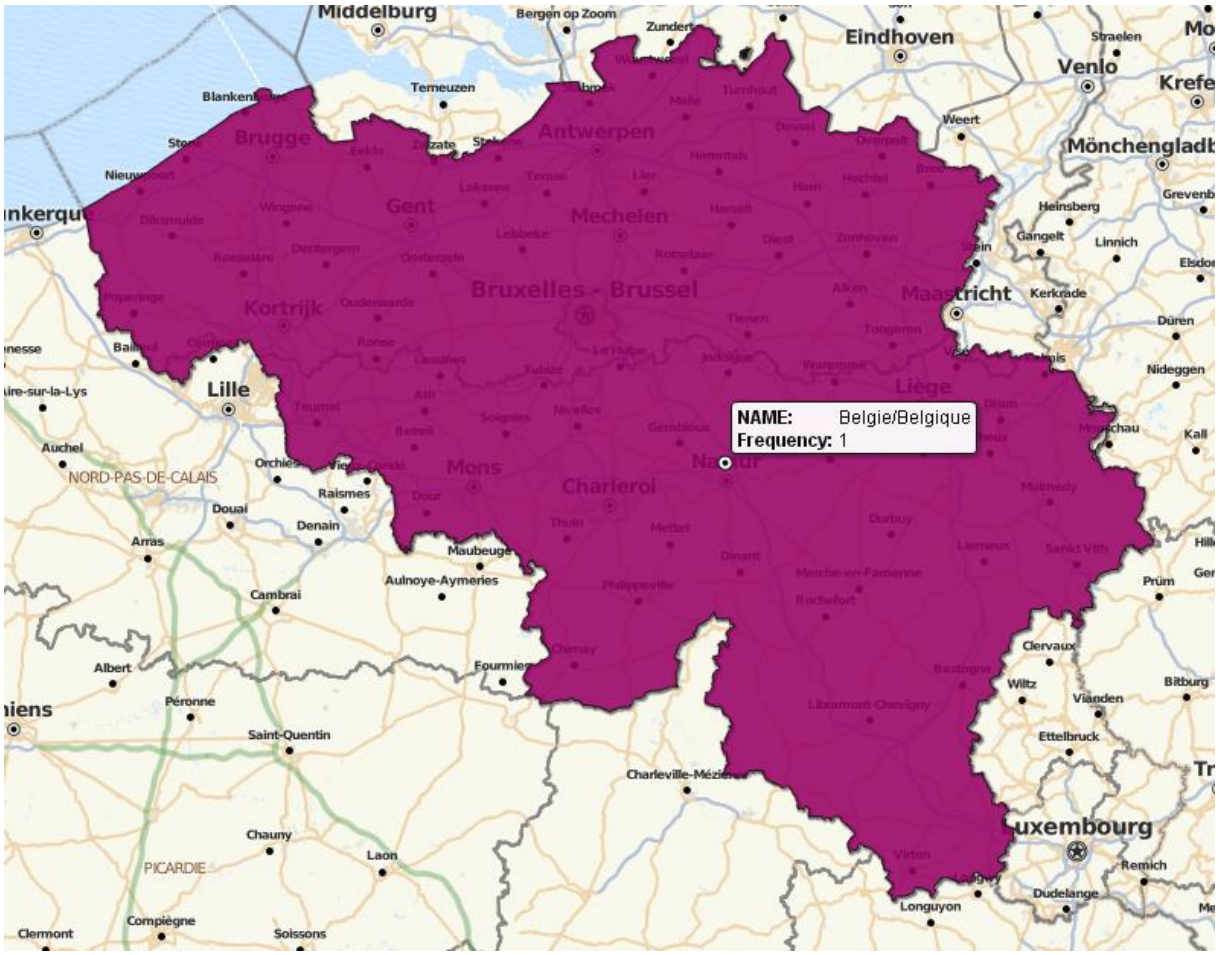
```
<config-dir>/Lev1/sas.servers restart
```

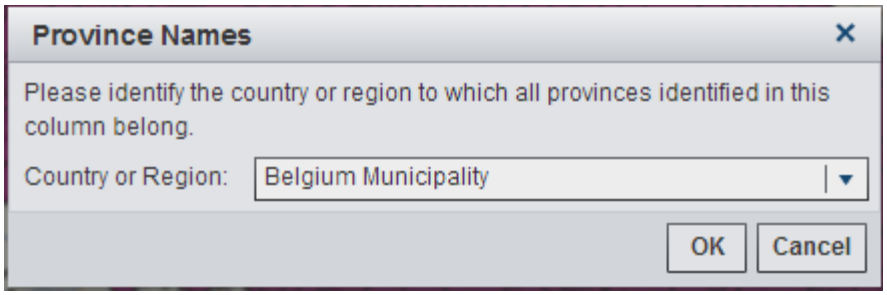
Once the VA environment is restarted you can load your data (or the Test Data) into LASR. You should be able to define both ID and NAME as geographical items:

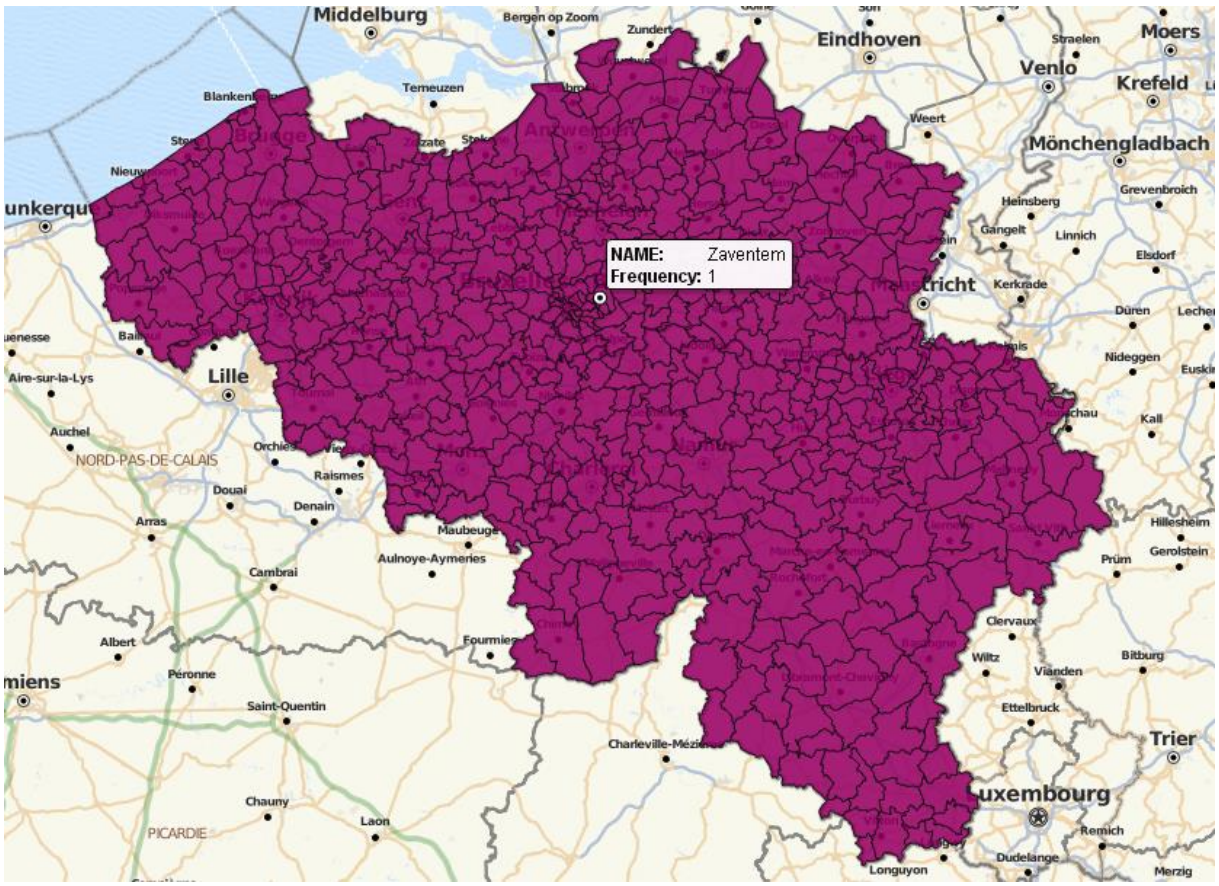
- ID = Subdivision (State, Province) SAS Map ID Values
- NAME = Subdivision (State, Province) Names

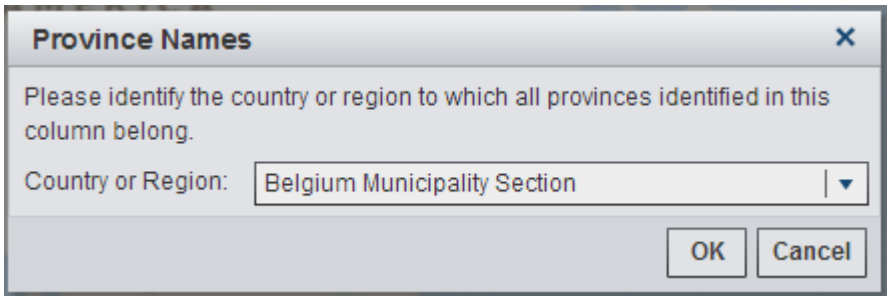


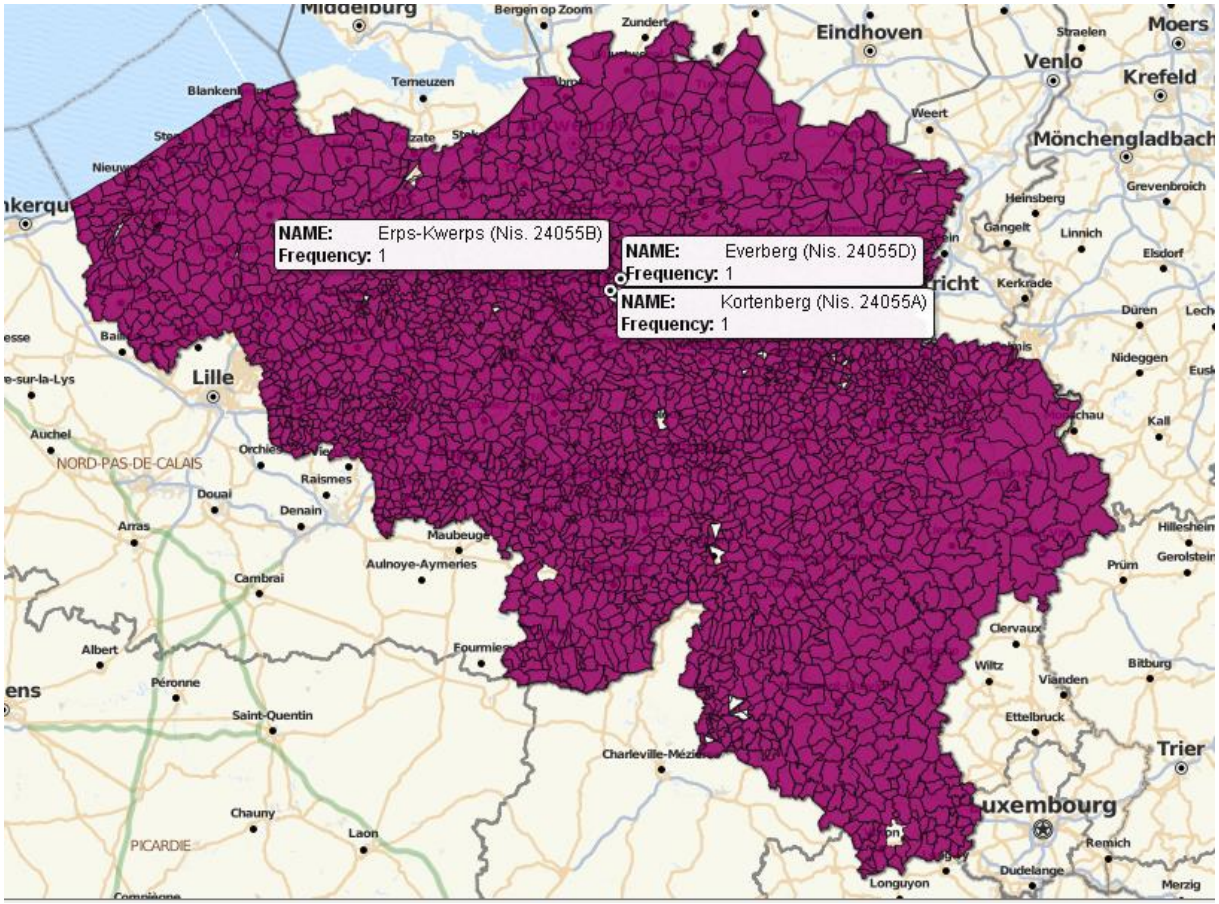

Province Names
Please identify the country or region to which all provinces identified in this column belong.
Country or Region:
Belgium Country
OK
Cancel


NAME: Belgie/Belgique
Frequency: 1
Middelburg
Bergen op Zoom
Zundert
Eindhoven
Venlo
Mönchengladbach
Weert
Heinsberg
Gangelt
Linnich
Kerkrade
Düren
Nideggen
Temeuzen
Blankenberge
Brugge
Antwerpen
Gent
Mechelen
Bruxelles - Brussel
Kortrijk
Lille
Liège
Charleroi
Mons
Namur
Arras
Douai
Denain
Raismes
Maubeuge
Aulnoye-Aymeries
Cambrai
Albert
Péronne
Saint-Quentin
Fourmies
Charleville-Mézières
Chauny
Laon
Clermont
Compiègne
Soissons
Clervaux
Wiltz
Vianden
Ettelbruck
Bitburg
Prüm
Remich
Dudelange
Longuyon
Luxembourg
NORD-PAS-DE-CALAIS
PICARDIE


Province Names
Please identify the country or region to which all provinces identified in this column belong.
Country or Region:
Belgium Municipality
OK
Cancel


NAME: Zaventem
Frequency: 1
Middelburg
Bergen op Zoom
Zundert
Eindhoven
Venlo
Moers
Krefeld
Mönchengladbach
Grevenbroich
Heinsberg
Gangelt
Linnich
Elsdorf
Kerkrade
Düren
Nideggen
Kall
Lille
Aire-sur-la-Lys
Auchel
NORD-PAS-DE-CALAIS
Orchies
Raismes
Douai
Denain
Arras
Maubeuge
Aulnoye-Aymeries
Cambrai
Albert
Péronne
Saint-Quentin
Fourmies
Charleville-Mézières
Clervaux
Wiltz
Vianden
Bitburg
Ettelbruck
Trier
Hillesheim
Gerolstein
Prüm
Remich
Dudelange
Longuyon
Merzig
Chauny
Laon
PICARDIE


Province Names
Please identify the country or region to which all provinces identified in this column belong.
Country or Region:
Belgium Municipality Section
OK
Cancel


NAME: Erps-Kwerps (Nis. 24055B)
Frequency: 1
NAME: Everberg (Nis. 24055D)
Frequency: 1
NAME: Kortenberg (Nis. 24055A)
Frequency: 1
Middelburg
Bergen op Zoom
Zundert
Eindhoven
Venlo
Krefeld
Moers
Mönchengladbach
Grevenbroich
Heinsberg
Linnich
Kerkrade
Düren
Nideggen
Kall
Lille
Auchel
NORD-PAS-DE-CALAIS
Douai
Arras
Denain
Raismes
Maubeuge
Aulnoye-Aymeries
Cambrai
Albert
Péronne
Saint-Quentin
Fourmies
Charleville-Mézières
Chauny
Laon
PICARDIE
Clervaux
Wiltz
Vianden
Bitburg
Ettelbruck
Trier
Prüm
Remich
Dudelange
Longuyon
Merzig

7. Tips and Tricks

- Take a back-up of the original lookup tables before making changes.
- Make sure that all the tables (especially the new tables with geographical data) have the correct permissions.
- Don't run your code twice! By adding the new region data again, your ID and IDNAME aren't unique.
- Make sure the REGION_PREFIX and REGION_ISO are unique for every geographical level that you add.
- Make sure that the name of your geographical datasets (PROVINCE_DATASET) ends with a 1.
- Use PROC GREDUCE and select only records up to density=3.
- No accents in names of regions are allowed.