ESPON 2020 MapKits. MapKits creation process

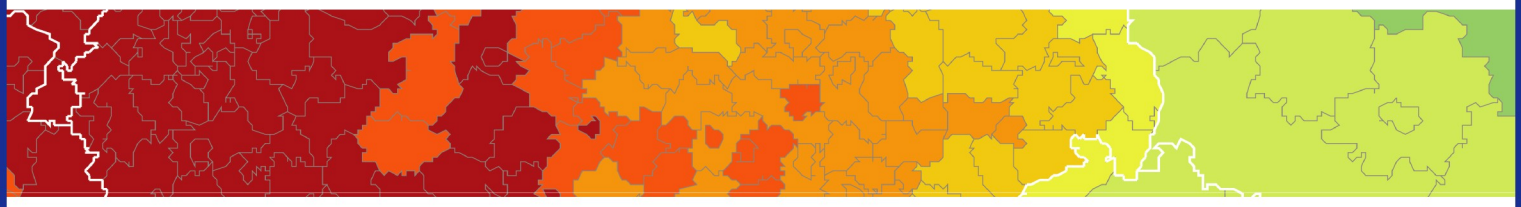
Ronan Ysebaert, Christine Zanin, Nicolas Lambert, Jacques Michelet, Erik Gløersen, Sebastian Hans

ESPON

EUROPEAN UNION

Co-financed by the European Regional Development Fund

Inspire policy making by territorial evidence



ESPON 2020 MapKits
MapKits creation process

Final Report

Version 11/05/2017

This research activity is conducted within the framework of the ESPON 2020 Cooperation Programme, partly financed by the European Regional Development Fund.

The ESPON EGTC is the Single Beneficiary of the ESPON 2020 Cooperation Programme. The Single Operation within the programme is implemented by the ESPON EGTC and co-financed by the European Regional Development Fund, the EU Member States and the Partner States, Iceland, Liechtenstein, Norway and Switzerland.

This delivery does not necessarily reflect the opinion of the members of the ESPON 2020 Monitoring Committee.

Authors

Ysebaert Ronan, UMS RIATE – University Paris Diderot (France)
Zanin Christine, UMS RIATE – University Paris Diderot (France)
Lambert Nicolas, UMS RIATE – CNRS (France)
Michelet Jacques, Universi
Gloersen Erik, Spatial Fore
Hans Sebastian, Spatial Fo

Advisory Group
ESPON EGTC: Martin Gauk, Marjan Van Herwijnen, Ilona Raugze (Director)

Information on ESPON and its projects can be found on www.espon.eu.

The web site provides the possibility to download and examine the most recent documents produced by finalised and ongoing ESPON projects.

This delivery exists only in an electronic version.

Contact: info@espon.eu

ISBN: ONLY FOR (DRAFT) FINAL REPORT, REMOVE OTHERWISE

ESPON 2020 MapKits

MapKits creation process

Table of contents

List of Figures

List of Maps

List of Tables

Abbreviations

EBM	EuroBoundaryMap
EC	European Commission
ECL	ESPON Cartographic Language
EFTA	European Free Trade Association
ESPON	European Territorial Observatory Network
EU	European Union
FUA	Functional Urban Areas
FYROM	Former Yugoslav Republic of Macedonia
GADM	Global Administrative Areas
GDB	GeoDataBase
GEOSPECS	Geographical Specificities and Development Potential in Europe
GIS	Geographical Information System
GISCO	Geographical Information System for the Commission
GREAT	Generalised Representation for European Areas and Territories
ITAN	Integrated Territorial Analysis of the Neighbourhood
LAU	Local Administrative Units
LUZ	Large Urban Zone
NUTS	Nomenclature of Territorial Units for Statistics
M4D	Multi Dimensional Database Design and Development
NMCA	National Mapping Cadastral Agencies
OGC	Open Geospatial Consortium
OSGEO	Open Source Geospatial Consortium
PUSH	Potential Urban Strategic Horizons
SNUTS	Similar to NUTS
TeDi	Territorial Diversity
QGIS	Quantum Geographical Information System
UMZ	Urban Morphological Zone
WUTS	World Units Territorial System

Executive summary

This technical report aims at describing how updating ESPON MapKits and generalised layers included in the MapKits in the future. It is divided in three parts.

The first part is dedicated to the GREAT layers, used for displaying NUTS territories in ESPON thematic maps (ESPON Narrow, Wide and Neighbourhood MapKits). The second part is dedicated to Voronoi layers, used for displaying LAU2 units in ESPON Transnational and Zoom-in MapKits. For these two layers of reference, created by UMS RIATE and University of Geneva, methods for updating these layers are explained (reference documents, quality control procedure, metadata and statistical codes).

The third part describes how a mapkit is organised: geographical layers selections, layers overlay and graphic styles. A step by step procedure for creating an ESPON MapKit in QGIS and ArcGIS softwares is explained.

1 Updating GREAT-Europe layer

1.1 Relevant data sources and formalization of NUTS change within a GIS framework

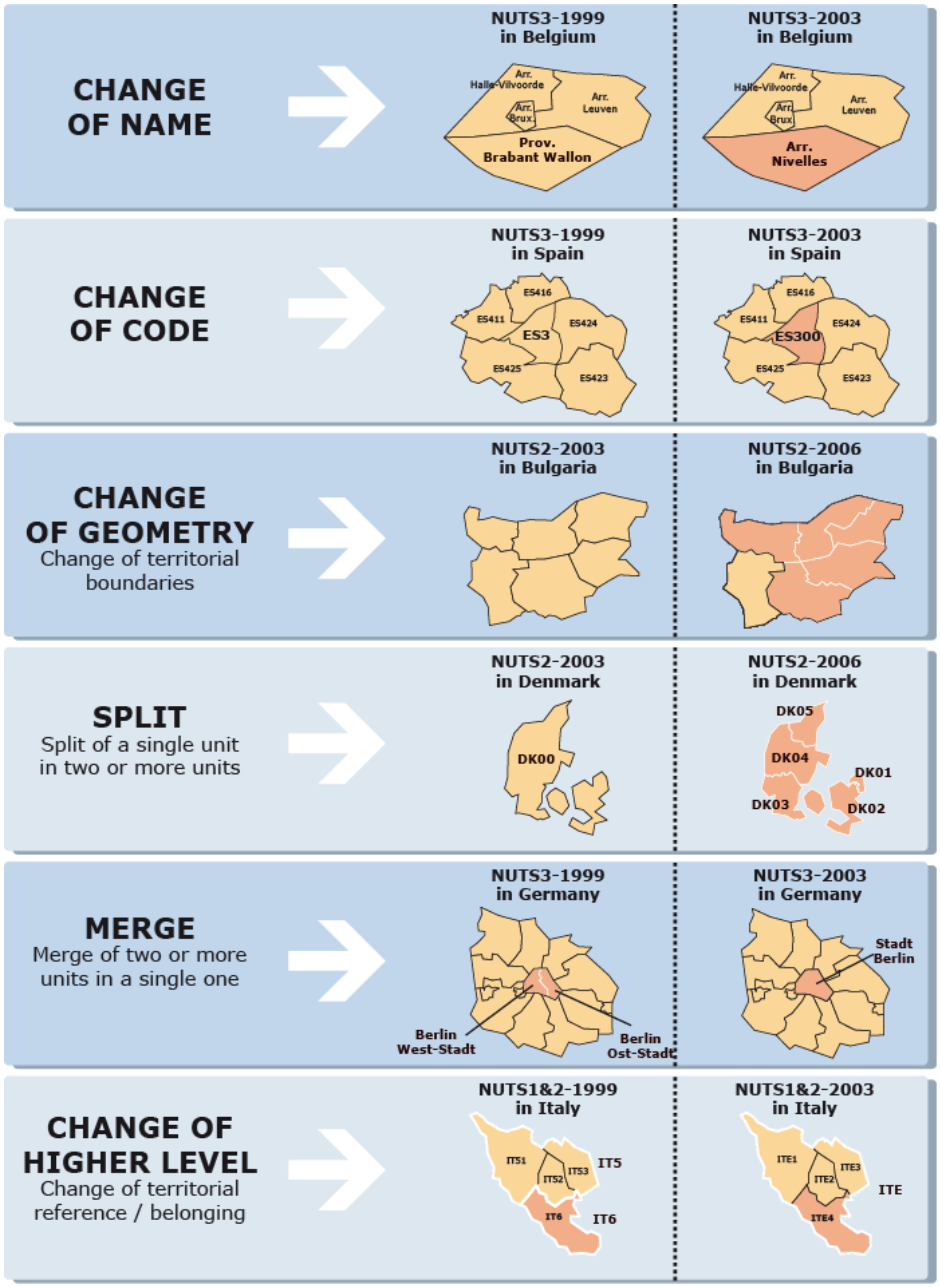
The NUTS nomenclature is the official division of the EU for regional statistics. Sometimes national interests require changing the regional breakdown of a country. When this happens, the concerned country informs the European Commission about the changes. The Commission in turn amends the classification at the end of period of stability according the rules of the NUTS Regulation.

The ESPON Database project proposes the following typology of changes[1]:

- **Change of name**: two cases can be distinguished. If the unit in question belongs to two levels (it is at the same time a NUTS 2 and a NUTS 3) the change of name can concern either one or the two levels.
- **Change of code:** This may result either from a political decision or from a territorial reorganisation.
- **Change of geometry:** This is the most complicated change type. Generally, the shape of a spatial unit can change in three different ways: a loss of area, a gain of area, or a redistribution of boundaries even while keeping the same area value.
- **Change of hierarchy:** This imply that the higher level units to which a given unit belongs are modified. As shown by Figure 1 -1, Italian Nuts 2 level have been assigned to new Nuts 1 units because of a reform of the nuts 1 level in 2003.

1 Ben Rebah M., Plumejeaud C., Ysebaert R., Peeters D., 2011, Modeling territorial changes and time series database building process: empirical approach and applications, Technical Report, ESPON Database Project.

Figure 1-1: Formalisation of NUTS changes (Source : ESPON Database Project, 2011)



The GREAT update consists in managing these territorial changes in the following perspectives:

(1) Ensure the quality and relevance of the geometries (boundaries and cartographic generalisation).

(2) Ensure the relevance of the layers codes (compliant with official nomenclatures)

(3) Document the data sources and the territorial nomenclature created in this context.

The section below proposes a step-by-step procedure to update the GREAT layer consequently. This update is systematically realised by UMS RIATE, out of project activities.

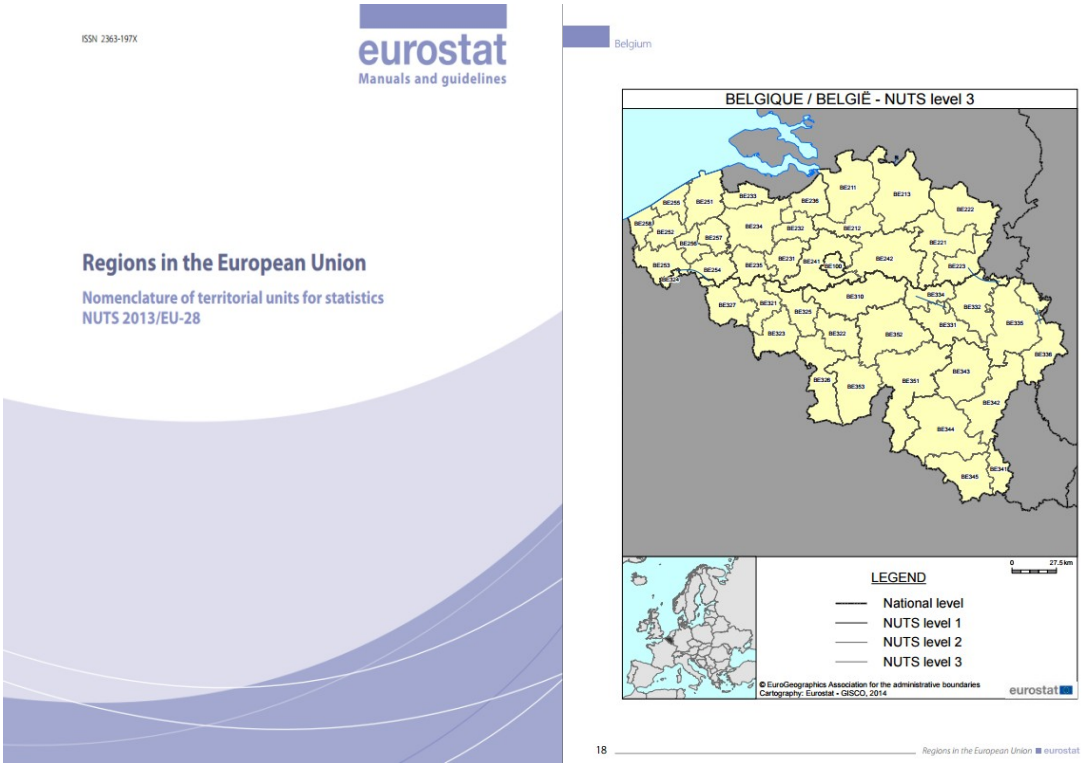
1.2 Step by step procedure to update GREAT Europe layers

1.2.1 Reference resources to be considered (Eurostat)

Reference documentation useful for managing the GREAT Europe update is provided by Eurostat. Three resources should be considered:

- *The Regions of European Union* report provides a map of each levels and versions[2] of the NUTS nomenclature, together with the codes, and official names of its territorial units. The same publication is available for EFTA and Candidate Countries. This report is particularly useful to check the quality and validity of the GREAT Europe layer.

Figure 1-2: Regions in the European Union, a useful document to check the validity of NUTS changes



- *The lists of changes between the various NUTS version* is certainly one of the most important resources since it provides textual information regarding the correspondence between two NUTS versions. In this Excel sheet, it is possible to highlight quickly territorial units concerned by the NUTS reform and their specificities (code change, territorial merge, boundary shift, etc.)

2 In May 2017, this document is not yet available for the NUTS2016 version.

Figure 1-3: List of changes between the NUTS versions 2013 and 2016

Code 2013	Code 2016	Country	Labels (version 2016 except in *italics = version 2013*) NUTS level 1	NUTS level 2	NUTS level 3	Change	NUTS level	Sorting order countries	NUTS 2013	NUTS 2016
FR223	FRE23				Somme	recoded	3	10	819	844
	FRF		ALSACE-CHAMPAGNE-ARDENNE-LORRAINE			new NUTS 1 region, merge of ex-NUTS 2 regions FR 21, FR 41 and FR42	1	10		845
FR42	FRF1			Alsace		recoded	2	10	849	846
FR421	FRF11				Bas-Rhin	recoded	3	10	850	847
FR422	FRF12				Haut-Rhin	recoded	3	10	851	848
FR21	FRF2			Champagne-Ardenne		recoded	2	10	811	849
FR211	FRF21				Ardennes	recoded	3	10	812	850
FR212	FRF22				Aube	recoded	3	10	813	851
FR213	FRF23				Marne	recoded	3	10	814	852
FR214	FRF24				Haute-Marne	recoded	3	10	815	853
FR41	FRF3			Lorraine		recoded	2	10	844	854
FR411	FRF31				Meurthe-et-Moselle	recoded	3	10	845	855
FR412	FRF32				Meuse	recoded	3	10	846	856
FR413	FRF33				Moselle	recoded	3	10	847	857
FR414	FRF34				Vosges	recoded	3	10	848	858
	FRG		PAYS DE LA LOIRE			new NUTS 1 region, identical to ex-NUTS 2 region FR51	1	10		859
FR51	FRG0			Pays de la Loire		recoded	2	10	858	860
FR511	FRG01				Loire-Atlantique	recoded	3	10	859	861
FR512	FRG02				Maine-et-Loire	recoded	3	10	860	862
FR513	FRG03				Mayenne	recoded	3	10	861	863
FR514	FRG04				Sarthe	recoded	3	10	862	864
FR515	FRG05				Vendée	recoded	3	10	863	865
	FRH		BRETAGNE			new NUTS 1 region, identical to ex-NUTS 2 region FR52	1	10		866
FR52	FRH0			Bretagne		recoded	2	10	864	867
FR521	FRH01				Côtes-d'Armor	recoded	3	10	865	868
FR522	FRH02				Finistère	recoded	3	10	866	869
FR523	FRH03				Ille-et-Vilaine	recoded	3	10	867	870
FR524	FRH04				Morbihan	recoded	3	10	868	871
	FRI		AQUITAINE-LIMOUSIN-POITOU-CHARENTES			new NUTS 1 region, merge of ex-NUTS 2 regions FR53, FR61 and FR63	1	10		872
FR61	FRI1			Aquitaine		recoded	2	10	875	873
FR611	FRI11				Dordogne	recoded	3	10	876	874
FR612	FRI12				Gironde	recoded	3	10	877	875
FR613	FRI13				Landes	recoded	3	10	878	876
FR614	FRI14				Lot-et-Garonne	recoded	3	10	879	877
FR615	FRI15				Pyrénées-Atlantiques	recoded	3	10	880	878
FR63	FRI2			Limousin		recoded	2	10	890	879
FR631	FRI21				Corrèze	recoded	3	10	891	880
FR632	FRI22				Creuse	recoded	3	10	892	881
FR633	FRI23				Haute-Vienne	recoded	3	10	893	882
FR53	FRI3			Poitou-Charentes		recoded	2	10	869	883
FR531	FRI31				Charente	recoded	3	10	870	884
FR532	FRI32				Charente-Maritime	recoded	3	10	871	885
FR533	FRI33				Deux-Sèvres	recoded	3	10	872	886
FR534	FRI34				Vienne	recoded	3	10	873	887

- The *GISCO layers* provided in Eurostat Website provide official geometries for the latest NUTS versions. These layers are especially useful for having the spatial reference of official boundaries and for splitting NUTS units (cf step 1.2.3 – Case 3)

Figure 1-4: Official geometries of NUTS units provided by Eurostat

GISCO: GEOGRAPHICAL INFORMATION AND MAPS

Overview
+ Geodata
+ GISCO activities
Frequently asked questions (FAQ)

NUTS

NUTS

Please be aware that there are specific download provisions for the datasets shown below which must respected. The download and usage of these data is subject to their acceptance.

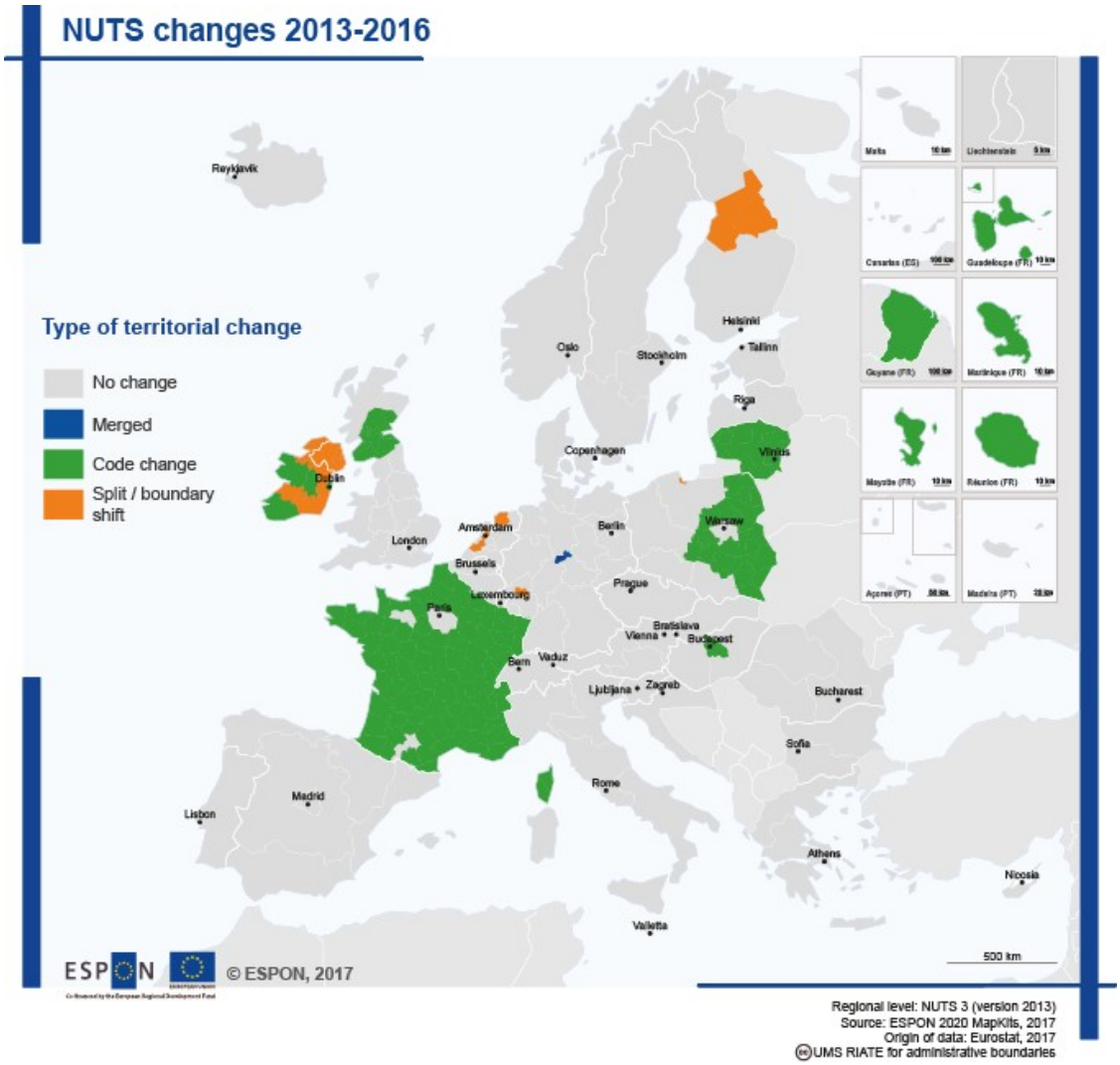
Administrative or Statistical unit	Scale	Coverage	Feature type	Format	Period	Coordinate reference system	Version date	File to download
NUTS 2013	1:1 Million	Europe	Point / Line / Polygon	Personal GBD	2013	ETRS89	03/12/2015	NUTS_2013_01M.zip
	1:1 Million	Europe	Point / Line / Polygon	Shapefile	2013	ETRS89	03/12/2015	NUTS_2013_01M_SH.zip
	1:3 Million	Europe	Point / Line / Polygon	Personal GDB	2013	ETRS89	03/12/2015	NUTS_2013_03M.zip
	1:3 Million	Europe	Point / Line / Polygon	Shapefile	2013	ETRS89	03/12/2015	NUTS_2013_03M_SH.zip
	1:10 Million	Europe	Point / Line / Polygon	Personal GDB	2013	ETRS89	03/12/2015	NUTS_2013_10M.zip
	1:10 Million	Europe	Point / Line / Polygon	Shapefile	2013	ETRS89	03/12/2015	NUTS_2013_10M_SH.zip

1.2.2 Mapping territorial changes using reference document

With a simple join between the GREAT layer of the previous version (here, 2013) and the table provided by Eurostat, it is possible to map territorial changes occurring between two NUTS versions (Map 1 -1).

This map makes it easier to plan the work required when updating the GREAT layer. For example, most of the changes between the NUTS 2013 abd 2016 nomenclatures will concern code changes (France, Poland, Lithuania, UK and Hungary). There are two mergers (Germany). The most complicated types of change (territorial split, in orange) will occur in Finland, Ireland, Germany, and the Netherlands.

Map 1-1: NUTS 3 units to be updated in the NUTS 2016 nomenclature



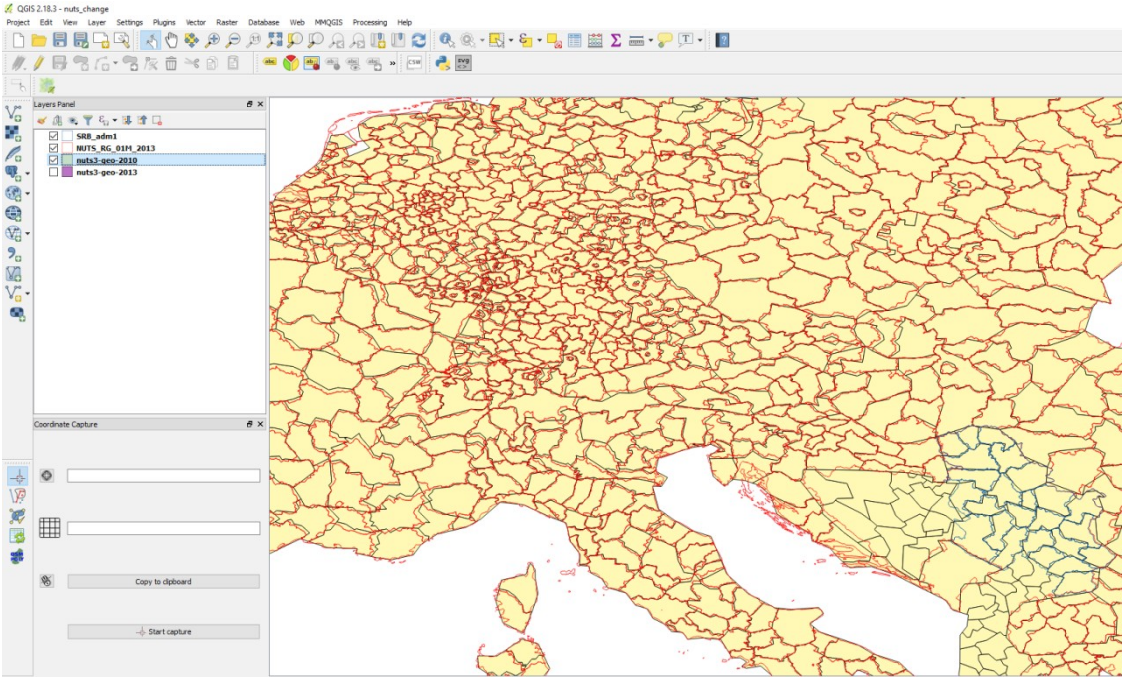
1.2.3 Update of the NUTS3 layer (new version)

Following the typology of NUTS change (Figure 1 -1), 4 cases must be considered: code change (C1), aggregation of territorial units (C2), Split of territorial units / boundary change (C3), and new country to be georeferenced (C4).

To highlight the methodology to be followed, the example of the NUTS2010 update into the NUTS2013 version is taken. In February 2017, the reference layers required for managing the NUTS2016 division have not yet been published. The screenshots displayed below use QGIS software, but the process would be exactly the same using ArcGIS.

The first step consists in uploading the NUTS3 layer in the previous NUTS version (nuts3-geo-2010 in this example) of the GREAT Europe layer in a chosen GIS and in duplicating it with the new name of the NUTS version (nuts3-geo-2013 if the update is for the NUTS 2013 version). The official layer provided by Eurostat is also uploaded in the GIS (NUTS_RG_01M_2013). It will be useful to process C3-type changes. Finally when adding new countries in the GREAT layer (C4-type change), GADM layers provide useful information[3]. **All edits that are described in this part will be applied to nuts3-geo-2013 layer**. The other layers are useful to check the quality of the process.

Figure 1-5: Update of the NUTS3 layer / Step 1 : Upload useful reference NUTS3 layer in a GIS



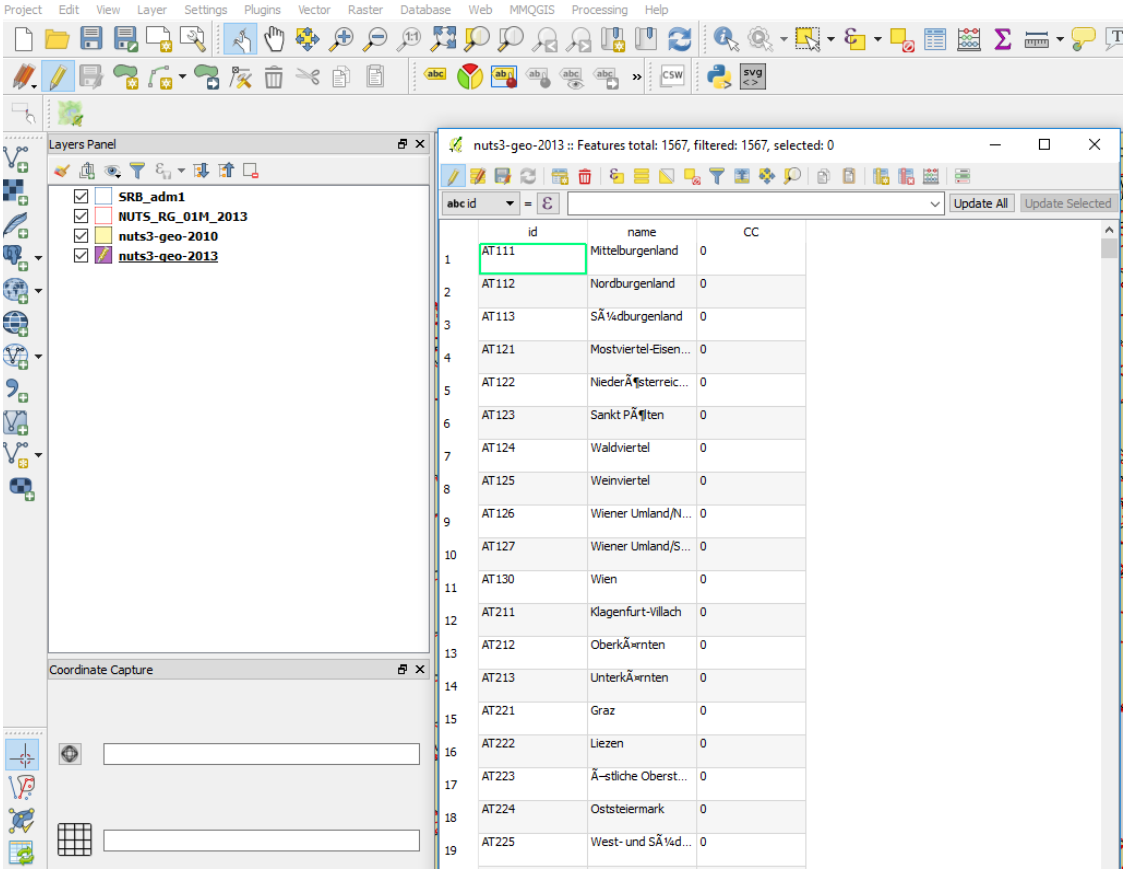
Case 1 – Code/name change

Updates of NUTS codes and names are managed in the attribute table of the new version of the NUTS layer (

3 Serbia is taken as an example in this guidance document. However, in the ESPON 2020 MapKit project, Serbian NUTS3 have been imported from the ITAN project (the layer is seamless with GREAT Europe layers.

). The data source used for this task is the list of changes between the NUTS versions provided by Eurostat (Figure 1 -3). It can be processed by a simple join with the table provided by Eurostat, or manually.

Figure 1-6: Update of the NUTS3 layer / Step 2 : Manage the code changes in the attribute table of the new NUTS layer

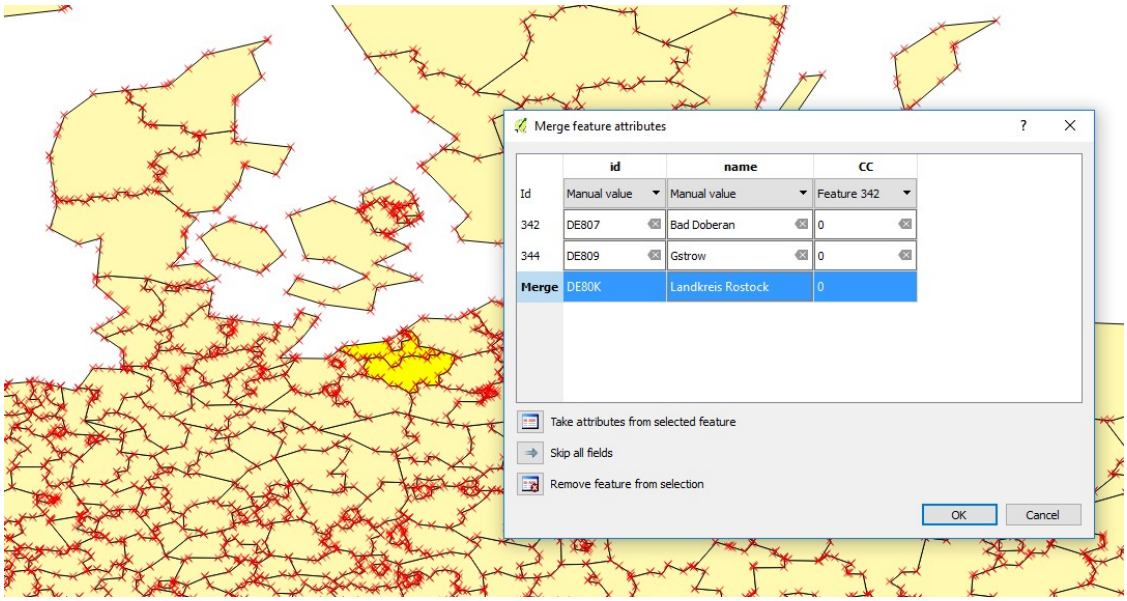


	id	name	CC
1	AT111	Mittelburgenland	0
2	AT112	Nordburgenland	0
3	AT113	SÃ¼dburgenland	0
4	AT121	Mostviertel-Eisen...	0
5	AT122	NiederÃ¶sterreic...	0
6	AT123	Sankt PÃ¶lten	0
7	AT124	Waldviertel	0
8	AT125	Weinviertel	0
9	AT126	Wiener Umland/N...	0
10	AT127	Wiener Umland/S...	0
11	AT130	Wien	0
12	AT211	Klagenfurt-Villach	0
13	AT212	OberkÃ¤rnten	0
14	AT213	UnterkÃ¤rnten	0
15	AT221	Graz	0
16	AT222	Liezen	0
17	AT223	Ã–stliche Oberst...	0
18	AT224	Oststeiermark	0
19	AT225	West- und SÃ¼d...	0

Case 2 – Territorial merge

In the case of merge of two or more territorial units in a single one, the merge feature option is used. In the example displayed in Figure 1 -7, DE807 and DE809 (NUTS 2010) are aggregated into DE80K (NUTS 2013).

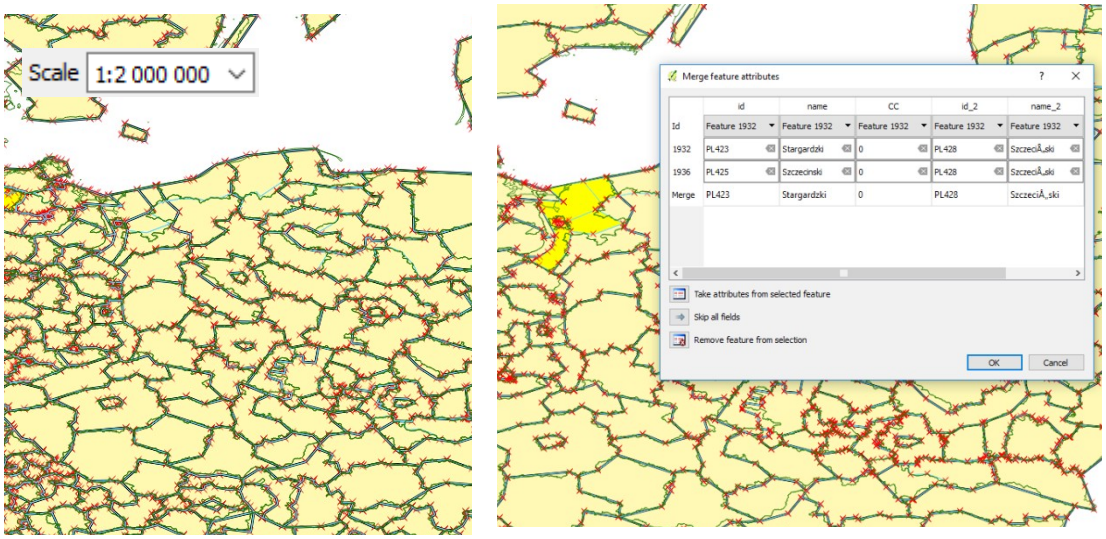
Figure 1-7: Update of the NUTS3 layer / Step 3 : Manage territorial merge



Case 3 – Split of territorial units

In case of boundary change or territorial split, NUTS3 units are split using the Eurostat layer as a reference. Optionally, territories are merged in a second steps (multi-part polygons in a single one, as displayed on Figure 1 -8 (right).

Figure 1-8: Update of the NUTS3 layer / Step 4 : Manage boundary changes by firstly splitting territorial units (on the left) and secondly merging territorial units belonging to the same territorial units (on the right).



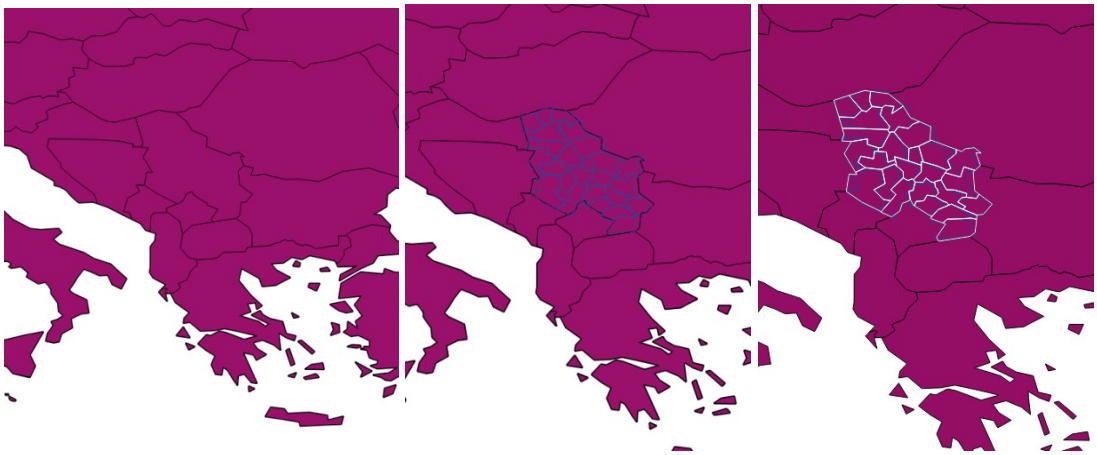
When splitting a territory in the GIS, levels of generalisation must correspond to those used in the rest of the map. A 1:2 000 000 scale is acceptable for the GREAT layer.

Nevertheless, the process of manual boundary creation and map generalization requires specific knowledge in layers creation and map harmonization. Several principles shall be considered: territories selection, territories schematisation, territories harmonisation (Lambert, Zanin, 2017) to create an adapted generalised layers.

Case 4 – New country to be georeferenced

This section describes the procedure to be followed to include new countries in the GREAT Europe layer. It has previously been applied when creating the SNUTS layer for Serbia within the ITAN and M4D project (Figure 1 -9). First, a layer at country level for all the countries of the World was created. This layer needed to be seamless with the GREAT layer (on the left). Then, GADM[4] data source were used to display the territorial division in Serbia. Finally, the polygon of Serbia was split using the same level of map generalization as the GREAT.

Figure 1-9: Using GADM reference to add new countries without Eurostat reference in the GREAT World layer (work done within M4D and ITAN projects for the European Neighbourhood).



1.2.4 Aggregation of the NUTS3 into NUTS2, 1 and 0 and correction of topological mistakes

When the territorial changes have been incorporated in the NUTS3 layer called “new version” (geometries and codes), this NUTS3 layer is merged at NUTS2, NUTS1 and NUTS0 levels. It is then necessary to correct potential topological problems generated as part of the dissolve process (Figure 1 -10 and Figure 1 -11). Some automatic tools exist in QGIS or ArcGIS to correct these topological errors more or less easily.

Figure 1-10: In the NUTS3 attribute table, create NUTS2, NUTS1 and NUTS0 fields

4 http://gadm.org/

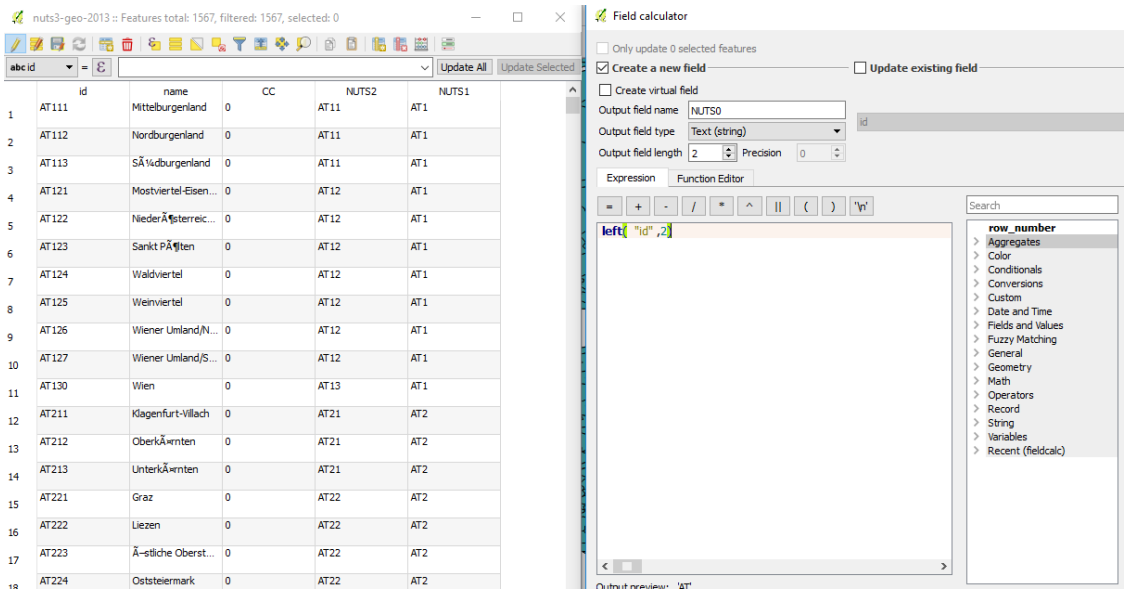
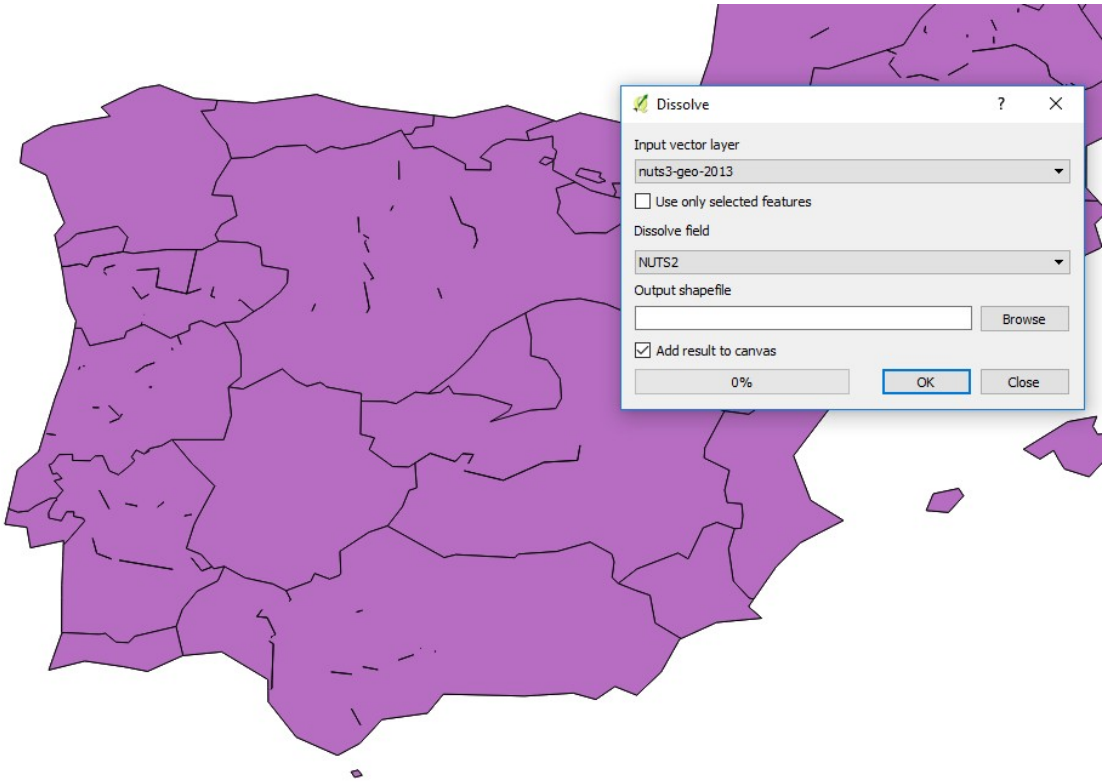


Figure 1-11: Dissolve the NUTS3 layer by attribute of the NUTS2 field and correction of topological problems afterwards.



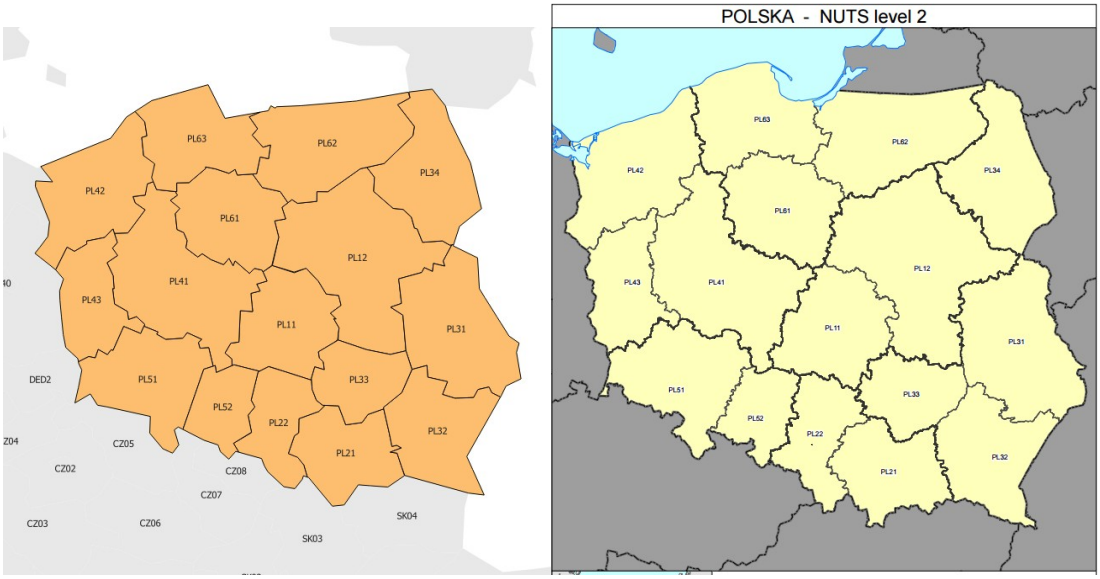
1.2.5 Quality check

Resulting layers are finally compared wto *the Regions of European Union* (Figure 1 -12). One needs to check that:

- It is possible to recognize territories in the map (level of generalization is not too important).
- NUTS codes are correct.
- No geometric and topological problems remains in the layer.

This is done for each level of the NUTS nomenclature and for each country.

Figure 1-12: Quality check of the NUTS2 in Poland using the Regions of European Union document



1.2.6 Documentation of territorial nomenclatures and geometries

The study area of the ESPON Program for regional analysis is EU28+4 (EFTA Countries), plus possibly Candidate Countries. This makes it necessary to combine several

nomenclatures: the official ones (EU28 territorial units), and regional nomenclatures coming from gentlemen agreement with authorities of EFTA and Candidate Countries. On top of that, some Candidate Country nomenclatures and geometries are not documented (Bosnia-Herzegovina, Serbia, etc.).

Consequently, regional nomenclatures included in the ESPON MapKits combine "official" and "non-official" regional nomenclatures, documented in metadata (metadata folder of the delivery, as displayed in Figure 1 -13).

It is necessary to ensure the lineage of this information. This is the reason why metadata are systematically added when a new nomenclature is created, as displayed in Figure 1 -13 for the NUTS 2013 nomenclature. This file also makes it possible to provide the official names of these territorial units, in national alphabets (Cyrillic, Greek) or in Latin alphabet (as provided by Eurostat).

Figure 1-13: Metadata for the ESPON NUTS nomenclatures

Unit code	Object type	Version	name_official	source	name_latin	source
RS221	NUTS3	2013	Борски округ	4	Borski okrug	4
RS222	NUTS3	2013	Браничевски округ	4	Braničevski okrug	4
RS223	NUTS3	2013	Зајечарски округ	4	Zaječarski okrug	4
RS224	NUTS3	2013	Јабланички округ	4	Jablanički okrug	4
RS225	NUTS3	2013	Нишавски округ	4	Nišavski okrug	4
RS226	NUTS3	2013	Пиротски округ	4	Pirotski okrug	4
RS227	NUTS3	2013	Подунавски округ	4	Podunavski okrug	4
RS228	NUTS3	2013	Пчињски округ	4	Pčinjski okrug	4
RS229	NUTS3	2013	Топлички округ	4	Toplički okrug	4
ME000	NUTS3	2013	Црна Гора	3	Crna Gora	3
XK001	NUTS3	2013	Gjakovë / Đakovica	7	Gjakovë / Đakovica	7
XK002	NUTS3	2013	Gnjilan / Gnjilane	7	Gnjilan / Gnjilane	7
XK003	NUTS3	2013	Mitrovicë / Kosovska Mitrovica	7	Mitrovicë / Kosovska Mitrovica	7
XK004	NUTS3	2013	Pejë / Peć	7	Pejë / Peć	7
XK005	NUTS3	2013	Prishtinë / Priština	7	Prishtinë / Priština	7
XK006	NUTS3	2013	Prizren / Prizren	7	Prizren / Prizren	7
XK007	NUTS3	2013	Ferizaj / Uroševac	7	Ferizaj / Uroševac	7
XK	NUTS0	2013	Republika e Kosovës / Republika Kosovo	7	Republika e Kosovës / Republika Kosovo	7
XK0	NUTS1	2013	Republika e Kosovës / Republika Kosovo	7	Republika e Kosovës / Republika Kosovo	7
XK00	NUTS2	2013	Republika e Kosovës / Republika Kosovo	7	Republika e Kosovës / Republika Kosovo	7
BA	NUTS0	2013	Bosnia and Herzegovina	7	Bosnia and Herzegovina	7
BA0	NUTS1	2013	Bosnia and Herzegovina	7	Bosnia and Herzegovina	7
BA01	NUTS2	2013	Distrikt Brcko	7	Distrikt Brcko	7
BA011	NUTS3	2013	Brcko	7	Brcko	7
BA02	NUTS2	2013	Federacija Bosna i Hercegovina	7	Federacija Bosna i Hercegovina	7
BA021	NUTS3	2013	Posavina	7	Posavina	7

Source Reference		
Label	1	
Date	2016-11-02	
Copyright	© Eurostat	
Provider	Name	Eurostat
	URI	http://ec.europa.eu/eurostat/fr/web/nuts/history
Publication	Title	Regions in the European Union Nomenclature of territorial units for statistics NUTS 2013 /EU-28. Eurostat, methodologies and Working Papers.
	URI	
	Reference	ISSN 1977-0375
Methodolog	Description	
	Formula	
	URI	
Access Rule	public	
Estimation	false	
Quality Leve	high	

Source Reference		
Label	2	
Date	2016-11-02	
Copyright	© Eurostat	
Provider	Name	Eurostat
	URI	http://ec.europa.eu/eurostat/fr/web/nuts/statistical-regions-outside-eu
Publication	Title	Statistical regions for the EFTA countries and the Candidate countries 2008. Eurostat, methodologies and Working Papers.
	URI	
	Reference	ISSN 1977-0375
Methodolog	Description	
	Formula	
	URI	
Access Rule	public	
Estimation	false	
Quality Leve	high	

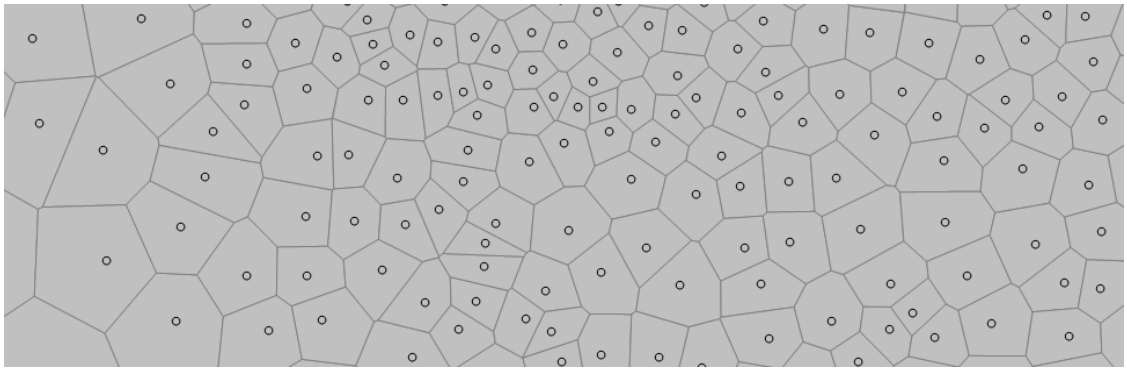
2 Updating Voronoi

2.1 Introduction

2.1.1 What is a Voronoi?

A Voronoi cell, also known as Thiessen polygon, is the basic unit of a Voronoi diagram. A Voronoi diagram divides – tessellates - a set of points of a plane into areas, so that the borders of the resulting areas are equidistant from the nearest points. It means that every location within a Voronoi cell is closer to the point around which it is drawn than to any other point.

Figure 2-14: Voronoi diagram methodology



The use of Voronoi diagrams is not new in cartography. One of the most well-known examples was produced by the British physician John Snow. Back in 1854, he used a method similar of Voronoi diagram to illustrate how the majority of people who died in the Broad Street cholera outbreak in London, lived closer to the infected Broad Street pump than to any other water pump.

Most of GIS systems offer nowadays automatic functions that allow tiling a set of points into areas using Voronoi operators.

2.1.2 Why should one use Voronoi diagram in ESPON Mapkits?

There are several advantages that have pledge in favour of Voronoi diagrams for displaying local statistical units (LAU) at regional level and above:

- Generating seamless terrestrial borders to maps of regional and local units is relatively easy.
- Far from adding accuracy, excessive level of details generates visually counter-productive ‘noise’ to maps above a certain level. In that sense the Voronoi diagram is best appropriate to generate cartographic message due to its “pixelisation effect”.

- Because of its high level of generalisation, only a small amount of computing power is required when processing these files in a GIS or in Adobe Illustrator. This makes the production of maps faster and easier.
- For the same reason, 'on the fly' production of maps, e.g. on an online mapping portal becomes feasible, even when displaying all of Europe.
- Based on Creative Commons license (CC-BY-NC-SA), the mapping layer can be distributed, used and updated by the potential users such as European institutions, students, academics, etc. In that sense, a seamless geographical layer, adapted to cartographic needs and matching with EU statistics fills an identified need.

2.2 Creation of the Voronoi mapping layer

In view of making it possible to update the Voronoi layer – as a whole or for some countries – this chapter describes the process of generating such a map step by step.

2.2.1 Preliminary work to the creation of the LAU Voronoi

This part describes the preparatory work required to generate Voronoi2 layers. Basically it consists of two steps:

1) Crafting the nomenclature upon which the Voronoi2 mapping layer will be based (steps a) and b)). The objective is to use a nomenclature that is fully compatible with the Eurostat Census 2011.
2) Generating the NUTS0 geometries that are seamlessly compatible both with the rest of the world countries as proposed by the Natural Earth 50m mapping layer and with the coastline created by UNIGE for the original Voronoi map (c, d and e)

a) Adjusting EuroBoundaryMap units and versions to the Census 2011 nomenclature

The detailed version of the Eurostat map of LAU2 units that is supposed to use the same nomenclature and codes as in the 2011 census data at a scale of 1: 100 000 was used. This shapefile was first adapted from EuroBoundaryMap versions 6.0 and , 6.2 by GISCO. In order to make this modified file fully compatible with census data, the ESPON MapKit project then made some manual adjustments to this map when this was possible (e.g. merging units in BG, splitting parishes that crossed NUTS borders in the UK), and used elements from EuroBoundaryMap v6 and v10 in other cases (e.g. Danish municipalities).

Also, for countries where no geometries or non-matching geometries in the 2011 census files were provided, different versions of the EuroBoundaryMaps were used either as reference or as full source of information (i.e. EL). This information is also specified in the corresponding metadata.

In the Danish case, 4 uninhabited church parishes (‘Sogn’) could not be associated with any geometry. Therefore, these 4 records were deleted (further information is provided in the metadata file).

The resulting shapefile has been named ‘ESPON LAU2 Census 2011’.

b) Generating the centroids from the resulting mapping layer

Centroids are points that GIS create at the centre of a polygon. They are the starting point for generating a Voronoi mapping layer.

There are many cases where centroids are placed outside the geographic object they represent and “fall” in the neighbouring object or in the sea (green dot in Figure 2 -15). This is the reason why the command “inner centroid” has to be used in order to guaranty that centroid computed are forced to fall within their original geographic object (orange square in Figure 2 -15).

Figure 2-15: Forcing centroids to fall within their geographical objects

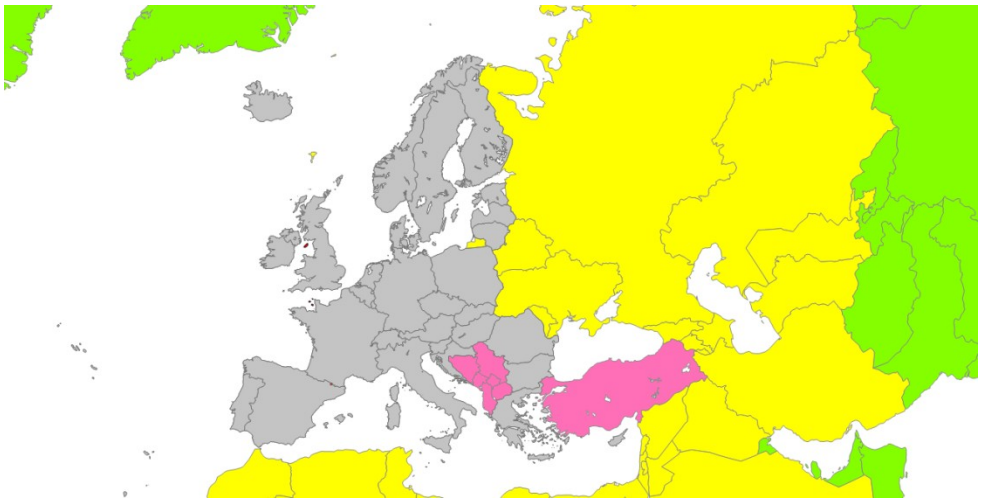
Once run over the all LAU units resulting from step a), the resulting file made of centroids looks as follow.

Figure 2-16: LAU2 centroids for the Voronoi layer

c) Adjusting the national borders of Natural Earth 50m layer

The rational for using national borders of the Natural Earth free public maps and dataset (http://www.naturalearthdata.com/) as a starting point is about ensuring a coherent degree of generalisation for national borders, when integrating the Voronoi file (Figure 2 -17, grey) with neighbouring countries (Figure 2 -17, yellow and pink) and the rest of world countries (Figure 2 -17, green).

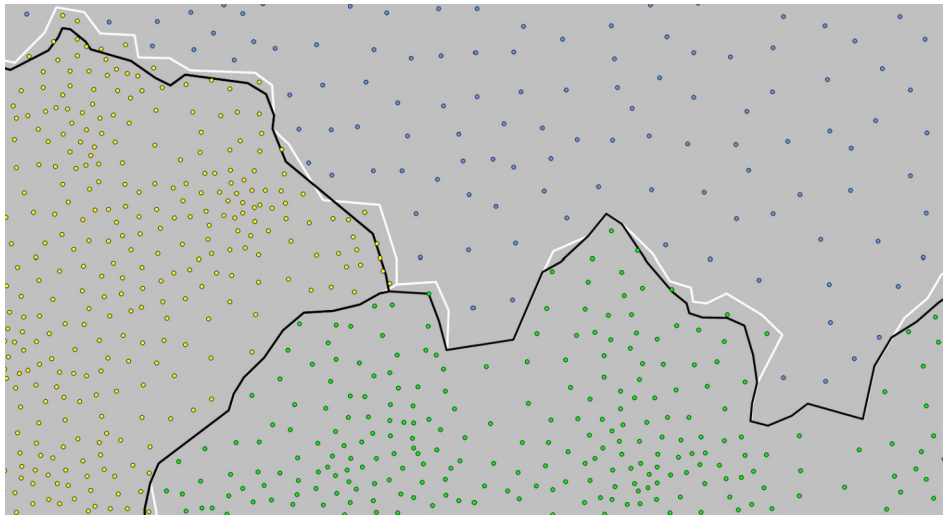
Figure 2-17: Ensuring coherent degree of generalisation for neighbouring countries



As a reminder, the degree of generalisation depends on the scale at which the mapping layer will be used. The Voronoi layer may for example be used to represent local data at macro-regional level. It has to be neither too precise nor not too coarse, in order to map to convey its message in the most efficient way.

While the Natural Earth 50m map fullfills has an appropriate level of generalisation, some problems occur because its generalised national borders in some cases position centroids that have been generalised at LAU2 level in the wrong wountry. This national border is shown in black in Figure 2 -18.

Figure 2-18: Adaptation of the national borders following the centroid structure of the country



This is why the the Natural Earth 50m national borders have been adapted so that they both: position all the centroids in the correct country, while preserving the original degree of generalisation (in white in Figure 2 -18).

d) Replacing coastline of Natural Earth 50m by the UNIGE Voronoi coastline

Similarly, the Natural Earth 50m map (in grey, Figure 2 -19) is too generalised for the purpose of mapping LAU units along the costline. Indeed many islands, some of them being LAU2 islands, are not represented in the Natural Earth (islands in white, witout a grey shape in the background Figure 2 -19).

This is the reason why the Natural Earth 50m coastline has been replaced with the Voronoi coastline especially developed by UNIGE to allow generalised mapping of LAU structure along the coasts (in green, Figure 2 -19).

The UNIGE coastline is based on GADM map. In order to generalise the coast line, improve visibility for small islands and facilitate the creation of Voronoi polygons, a buffer of 1’500 meters was applied to the original GADM coastline. Major estuaries and fjords that were closed by this buffering) were drawn manually. As a consequence, costalines are located a bit further into the sea when compared to their actual geographical position.

Figure 2-19: Creation of a dedicated coastline for the Voronoi layer

e) Producing the NUTS0 framework for Voronoi mapping layer

Once national borders and coasts have been adapted and fixed, NUTS0 entities have been created within these “land” and Sea” boundaries.

The last step consists in inserting lake surfaces. Lake geometries are based on a file produced by the European Environmental Agency (EEA). Only 39 lakes of more than 250 km2 have been taken into account. Their contours have been simplified by UNIGE.

Figure 2-20: Insert lakes of more than 250 km² in the Voronoi layer

2.2.2 Steps to generate the Voronoi diagrams, country per country

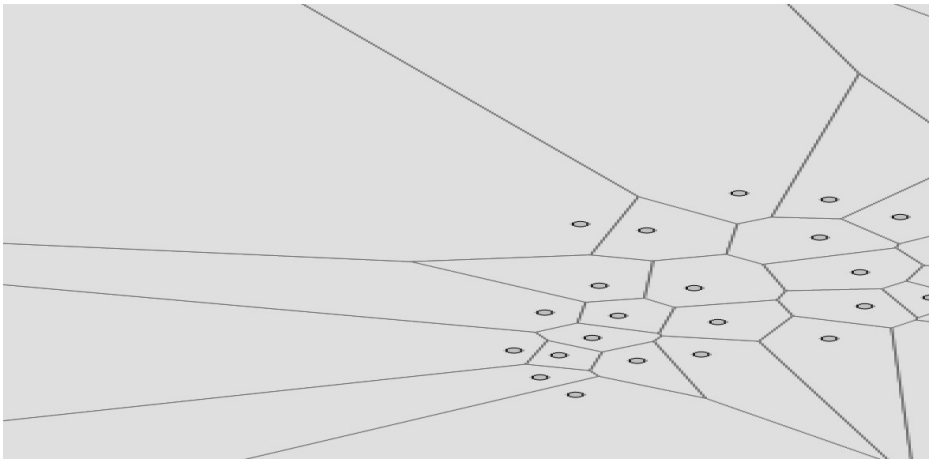
This part is about generating the Voronoi diagrams and the adjustments necessary for ensuring geographical coherence of the final output with regard to the input mapping layer, the EuroBoundaryMap at LAU2 level in the nomenclature of the Eurostat Census 2011.

Due to the substantial number of LAU units of the census 2011 (117'709 objects), it has been decided not to generate the Voronoi layer as a whole, but country by country. This will also make future updates of the Voronoi LAU2 layer easier, as each country can be processed separately to reflect significant changes in LAU2 boundaries.

f) Generating the Voronoi diagram

The Voronoi diagram is generated from the centroids obtained as the result of the process exposed in step b). The Voronoi areas are computed by using the appropriate GIS command (the name of the command is specific to each software).

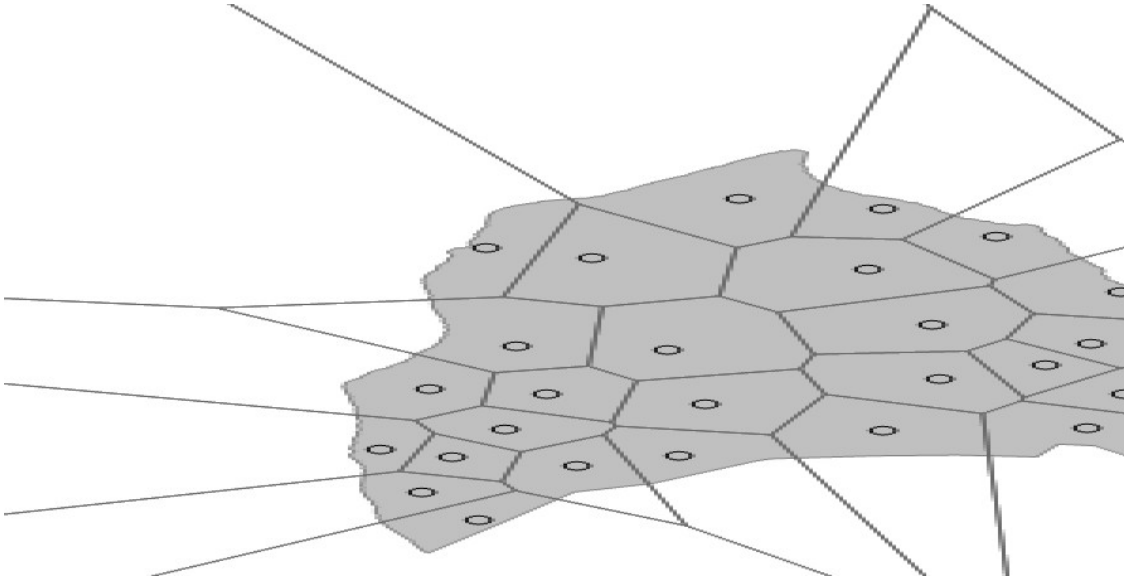
Figure 2-21: Voronoi diagram creation



g) Clipping the Voronoi diagram

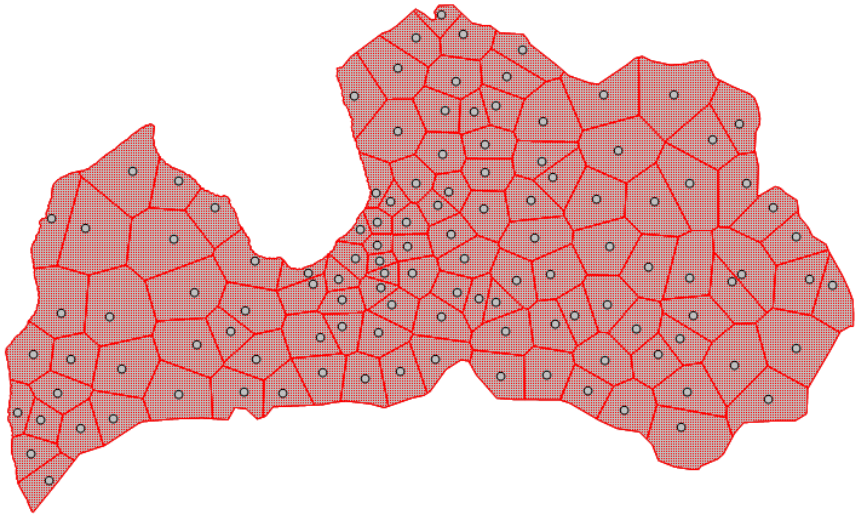
The next step consists in clipping the Voronoi diagram with the polygon of its respective country, obtained as the result of step e).

Figure 2-22: Clip the Voronoi diagram with the polygon of its respective country



Once clipped, the end result of the process should provide this result (Figure 2 -23)

Figure 2-23: Resulting LAU2 layer (Latvia)



h) Adjusting manually the geographical coherence (contiguity)

As the result of the process of generating Voronoi polygons, a municipality that is contiguous to a coast, a major lake or a national border in the EuroBoundaryMap may lose this feature in the resulting geometry. Inversely, a municipality that is not contiguous to a coast or border a national in the EuroBoundaryMap may be assigned a coastline or a portion of national border.

In order to ensure that the output map is topological consistent with the actual situation with regards to coastlines, major laes and national borderlines, manual adjustments have been carried on so far as possible. In a few cases, manual corrections have not been possible, e.g. LAU2 bordering long and narrow fjords.

Figure 2-24: Example of manual adjustment to re-establish contiguity to a coast or a border

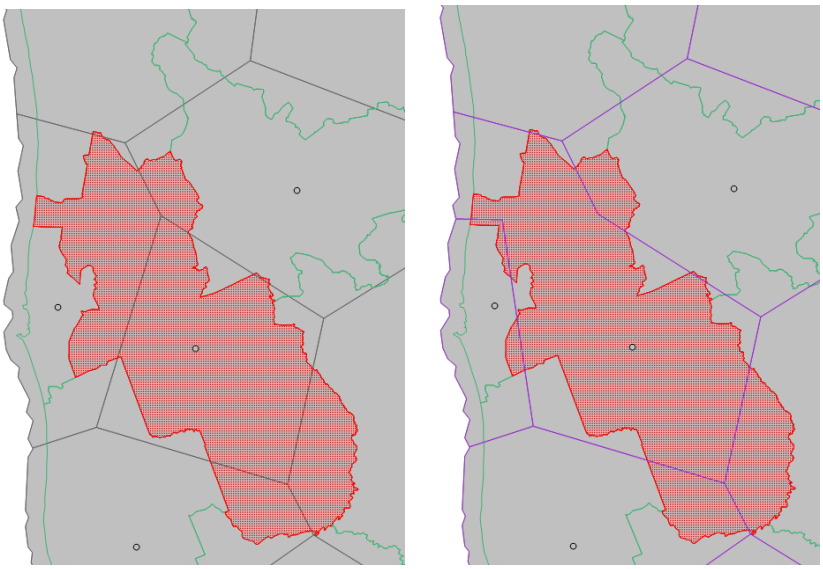
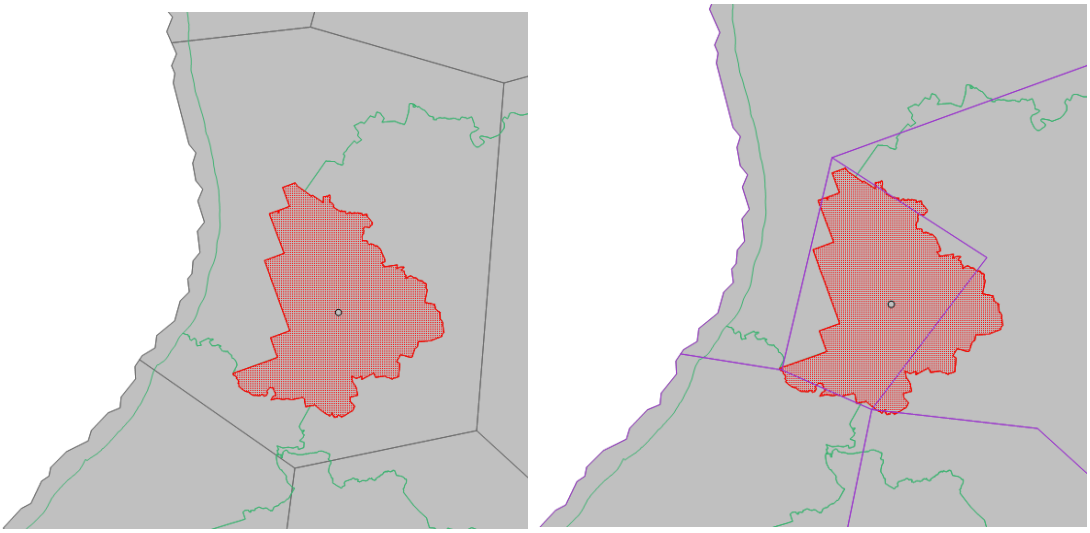


Figure 2-25: Example of manual adjustment to re-establish discontinuity to a coast or a border

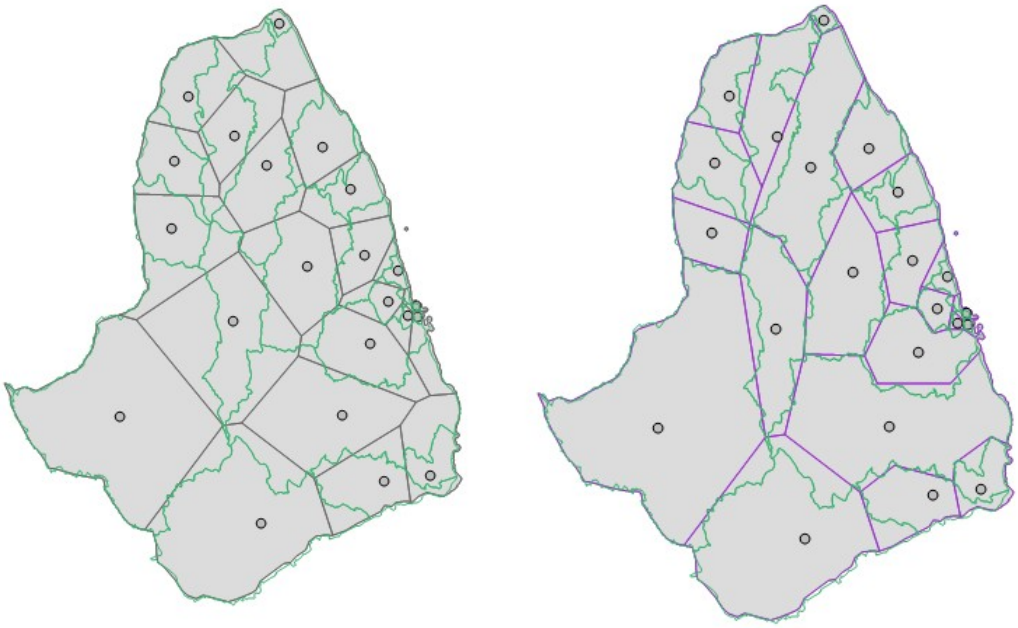


i) Adjusting manually the geographical coherence (geographical characteristics)

In the case of Europe's largest municipalities (e.g. in the Nordic countries, Iceland and Guyana) the generation of the Voronoi diagram led to differences that are so obvious that they disturb the cartographic message, even at a macro-regional level. At a much narrower scale, the same is true for small islands with only few municipalities.

In both cases manual adjustment is necessary so as to ensure minimum congruence between the Voronoi layer and the original geometry in order not to convey the desired the cartographic message.

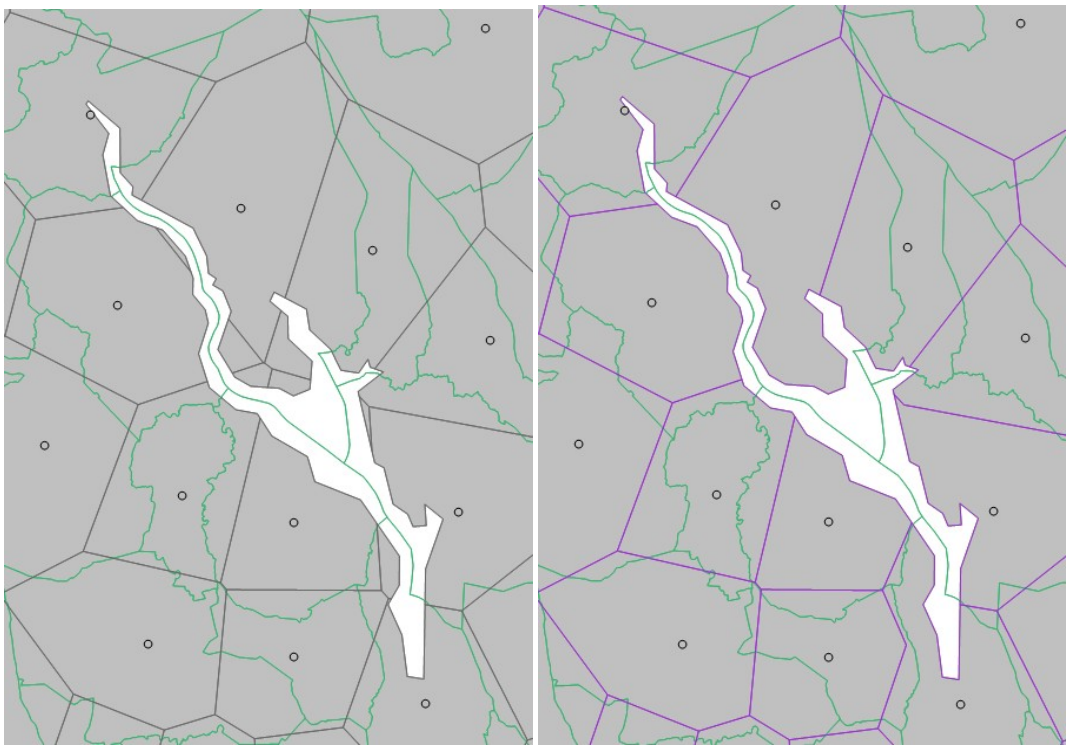
Figure 2-26: Manual adjustment of the LAU2 layer for large municipalies (French Guyane)



Lakes and estuaries are easily identifiable geographical structures as well. Bordering municipalities therefore require particular attention so as to ensure minimum coherence with regard to their relative position in relation with the lake.

When adjusting municipalities that are contiguous to a major lake, small fragments (so-called “branched objects”) may result from the clipping of the Voronoi polygons with the lake. These fragments must be reattributed to the polygon that makes the most sense, from a geographic perspective.

Figure 2-27: Fix “branched objects”

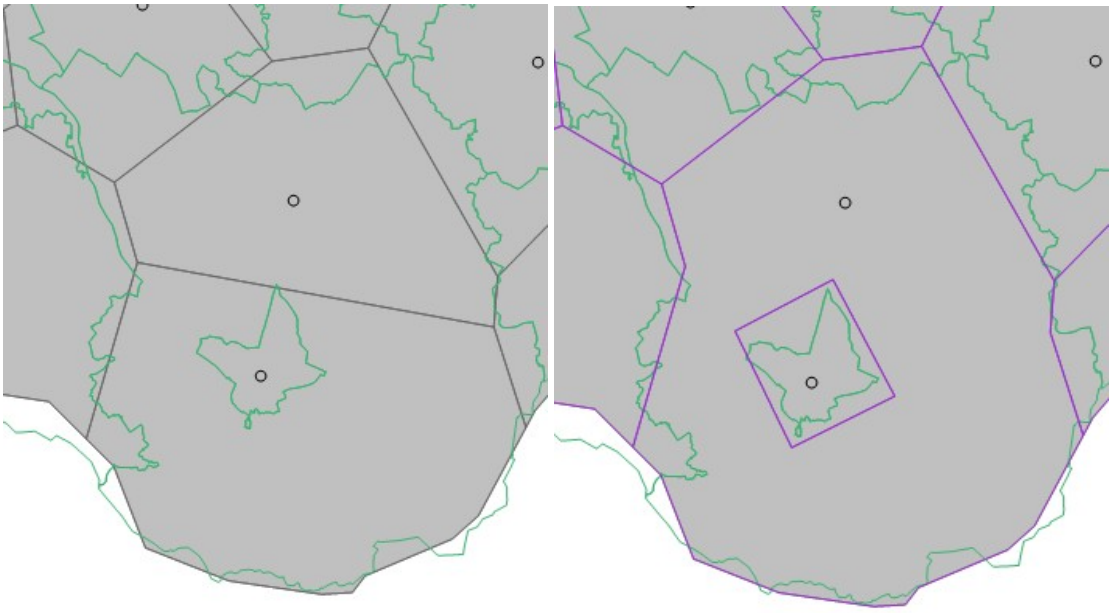


Some small municipalities are totally surrounded by a much larger one. In addition to be a prominent geographical situation that is easily identifiable, this type of situation often refers to a city surrounded by a more rural territory (Figure 2 -28). While such LAU2 boundaries are most common in countries of north-eastern Europe, they also occur in a number of other countries

Figure 2-28: Specific geographical pattern: an urban object surrounded by rural territory

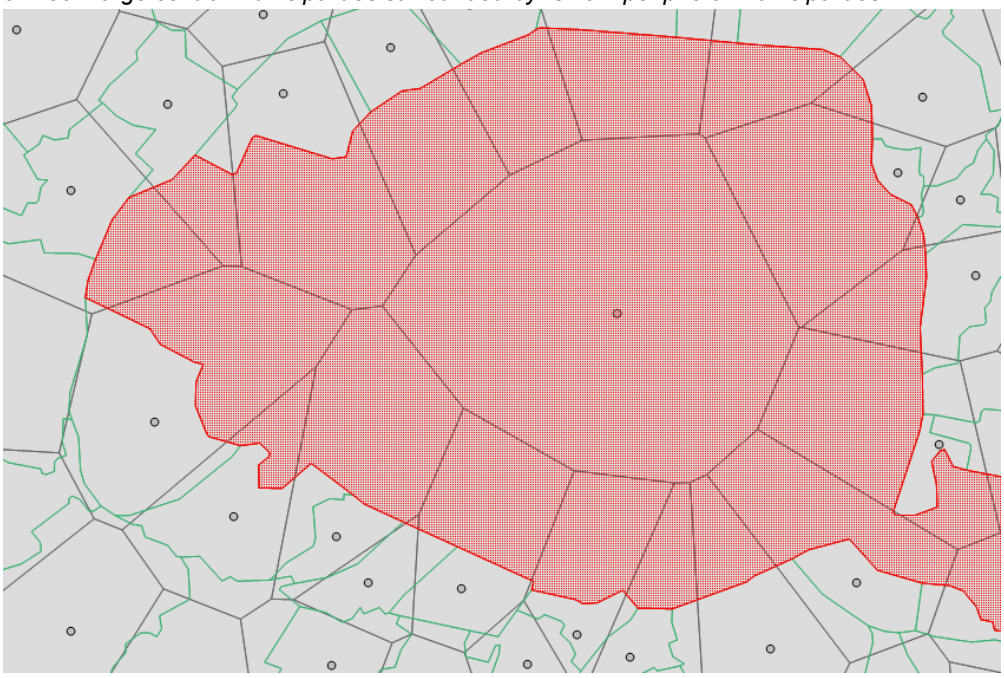
With regard to cartographic message, it is therefore essential to reproduce manually this type of territorial structure that is blurred when generating the Voronoi.

Figure 2-29: Manual adjustments in the Voronoi layer for urban object surrounded by rural territories



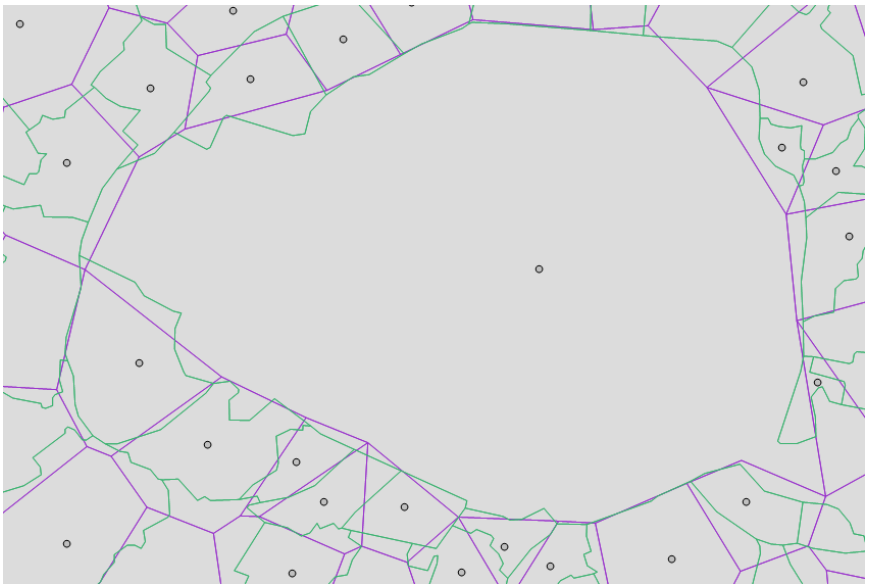
Oppositely, the urban structure is in other cases composed by a large central municipality surrounded by small peripheral municipalities. In such a case, owing to the nature of the Voronoi (closer to the point around which it is drawn than to any other point), the central unit will become much smaller, while the surrounding units will be crafted much larger than they really are. Resulting effect is that important territorial structure - because often referring to metropolitan areas - is lost.

Figure 2-30: Large central municipalities surrounded by "small" peripheral municipalities



Once again it is necessary to reproduce manually this type of territorial structure that is blurred when generating the Voronoi.

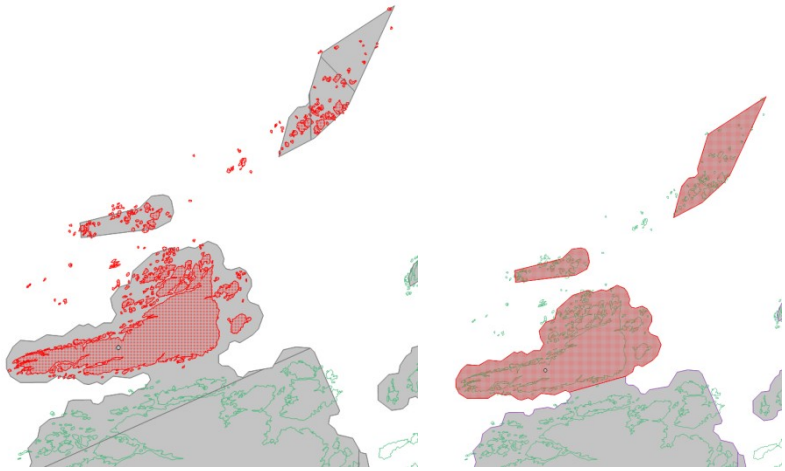
Figure 2-31: Manual adjustment in the Voronoi layer for large central municipalities surrounded by "small" peripheral municipalities



j) Adjusting manually remaining blunders

Similarly to the issue noticed for the lakes (step i), fragments resulting of the clipping of the Voronoi polygons with the coastline must be fixed, with a special concern in the case of islands. They must be reattributed to the LAU unit they correspond to and all fragments must be unified into a single object.

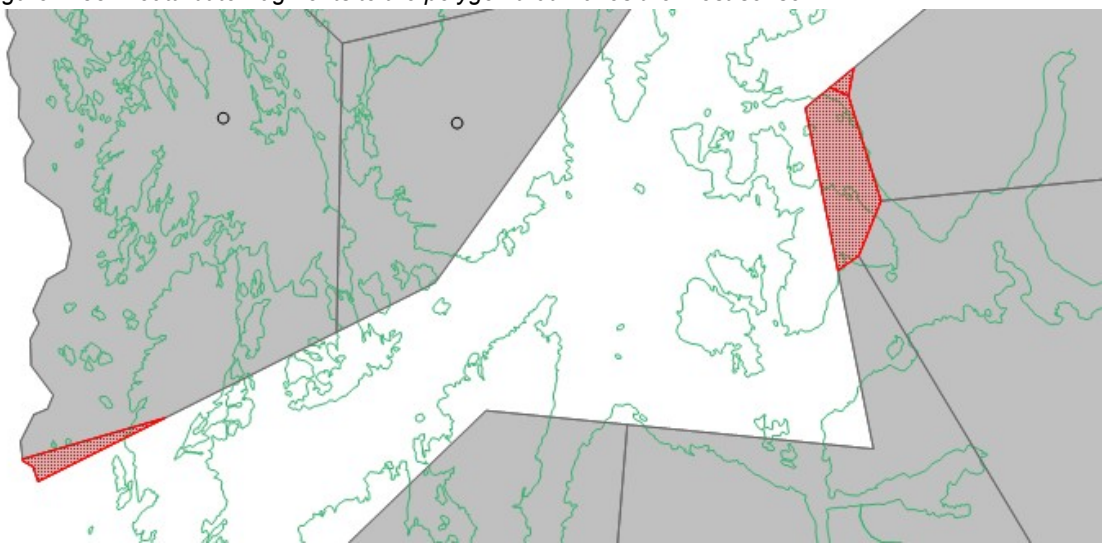
Figure 2-32: Fragments to be unified



Finally, one has to reassign numerous small fragments. These fragments can be identified by sorting objects of the Voronoi diagram by their size. Small fragments are likely to rank among

smallest objects. Once identified, they must be reattributed to the polygon that makes the most sense, from a geographic perspective.

Figure 2-33: Reattribute fragments to the polygon that makes the most sense



k) Cleaning topological errors

While the Voronoi diagram is originally generated without topological errors, there two causes that pledge in favour of controlling topological errors and correcting them. In both cases, it is recommended to set the precision to the maximum value.

- First, errors may have been introduced when making the manual adjustments listed above (steps h), i), j)). Topology control will be run only within the Voronoi mapping layer itself.
- Second, topological errors are generated along national border when Voronoi mapping layers from two contiguous countries are merged into a single shapefile. In the initial shapefile of each country, straight lines along their external border will be subdivided in as many segments are there are LAU2 units bordering the neigbouring country. The position and length of these segments will not coincide along the two countries' shared border, as they will in each case reflect each country's respective LAU2 delineation. When the shared border is transformed into a single line in the merged file, one therefore has to subdivide it into new segments, that take into account both countries' LAU2 delineation along the border. This is done in the following way:

    1) compose a single shapefile with all elements (Voronoi diagrams, lakes, microstates, neighbouring countries and other if relevant)

    2) run a topological tool with best precision on this file

    3) correct (if the tool does not make it automatically) all topological errors

4) select elements by type (Voronoi diagrams, lakes, microstates, neighbouring countries and other if relevant) and to export them in distinct file

Even if few topological errors are visible on output maps, one should not underestimate the importance of this task. Indeed, cleaning topological errors is essential for further processing of the resulting GIS file, such as R automatized treatments, running cartograms software, etc.

2.3 Updating Voronoi mapping layer

The ESPON 2020 programme may in the future need to adapt the Voronoi thematic mapping layer for some countries, to present data in updated territorial nomenclature. In order to do so, the project team is invited to follow some of the steps detailed previously:

1) Adjusting EuroBoundaryMap units (a)
2) Generating the centroids from the resulting mapping layer (b)
3) Generating the Voronoi diagram (f)
4) Clipping the Voronoi diagram with the Voronoi NUTS0 polygon for the country concerned (g)
5) Adjusting manually the geographical coherence (contiguity) (h)
6) Adjusting manually the geographical coherence (geographical characteristics) (i)
7) Adjusting manually remaining blunders (j)
8) Cleaning topological errors (k)

However before going into such a process, one should be informed that the method is quite time consuming.

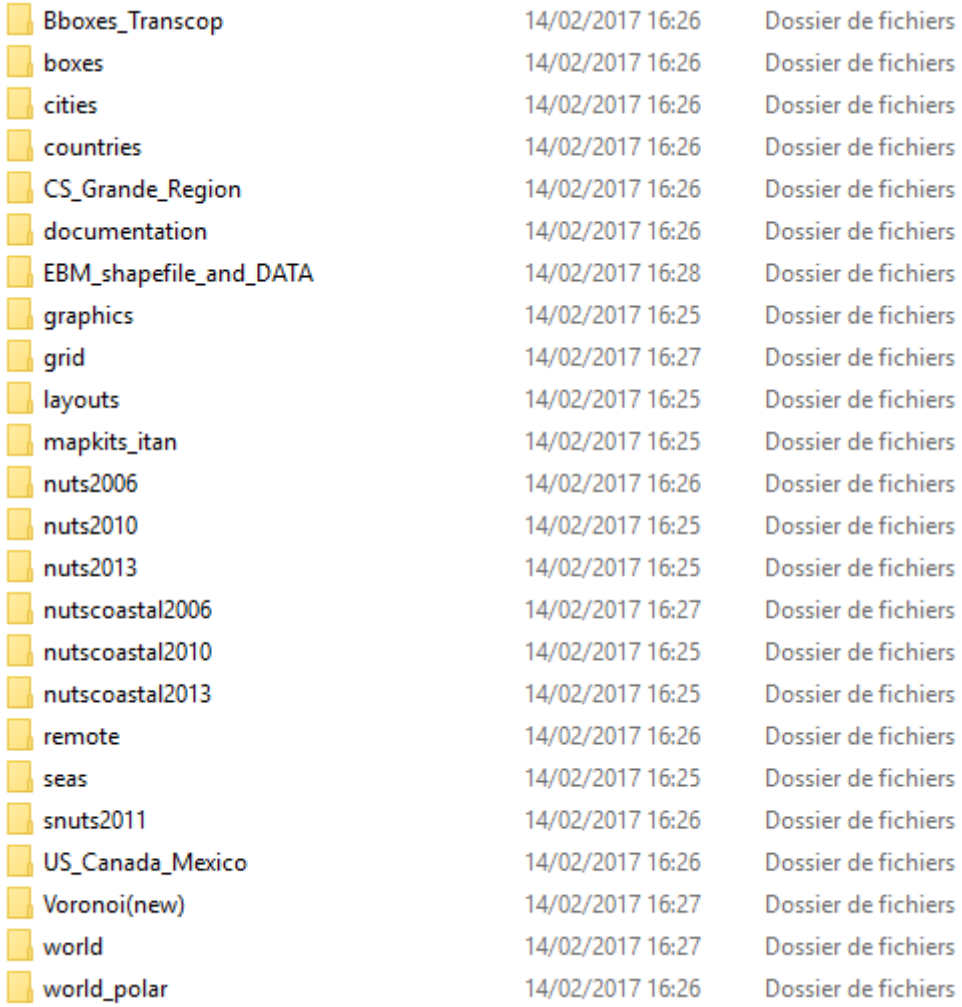
3 Creating a MapKit in QGIS, ArcGIS and Adobe Illustrator formats

This section describes the MapKit creation process for the ESPON EGTC. It is not intended to be disseminated to ESPON Projects in that form (cf Using ESPON MapKits guidance document).

3.1 Input layers preparation

When creating a MapKit, the most time-consuming task consists in preparing all the requested geographical layers and checking their **topologic validity** and **codes**. All geographical layers used by ESPON 2020 MapKits are located in the “input” folder. All are plotted using the World Geodetic System (WGS84) reference system. It corresponds to the reference coordinate system used by the Global Positioning System.

Figure 3-34: Input layers used by ESPON 2020 MapKits



3.2 Layout creation

Starting from a template designed in a CAD (Computer Aided Drafting, such as Adobe Illustrator), the template creation is designed with a R programme in order to be sure that the layout (ESPON blue stripes) will always have the same width, whatever the geographical extent of the Mapkit.

This R Programme displayed in Annex 1 is quite simple to use. In an R environment, one first needs to install the required packages (rgeos and rgdal), and then to to execute the function 'BuildTemplate'. The function is designed in a way to be fully compatible with the layout provided in an Adobe Illustrator format. The 'BuildTemplate' function takes in entry the **bounding box parameters** of the **desired layout** (xmin, xmax, ymin, ymax) and the **cartographic projection parameters** (prj). It returns a **shapefile (template.shp)** with the layout correctly sized in the desired cartographic projection.

3.3 Layers and territories selection / intersection with the layout

Irrespective of the template, a four steps procedure needs to be followed when creating a MapKit: (a.) selection of layers, (b.) selection of territories, (c.) intersection of layers intersection and (d.) definition of scale value.

In the ESPON 2020 MapKit project, all this process has been designed in a R Programme. This programming technology allows to reproduce easily the MapKits using systematically the same parameters. Annex 2 provides the code used for the Alpine Space MapKit. It is important to remind that these four steps could also be fully realised using a GIS, but it would be more time-consuming ("click-button" procedure).

a) Selection of layers

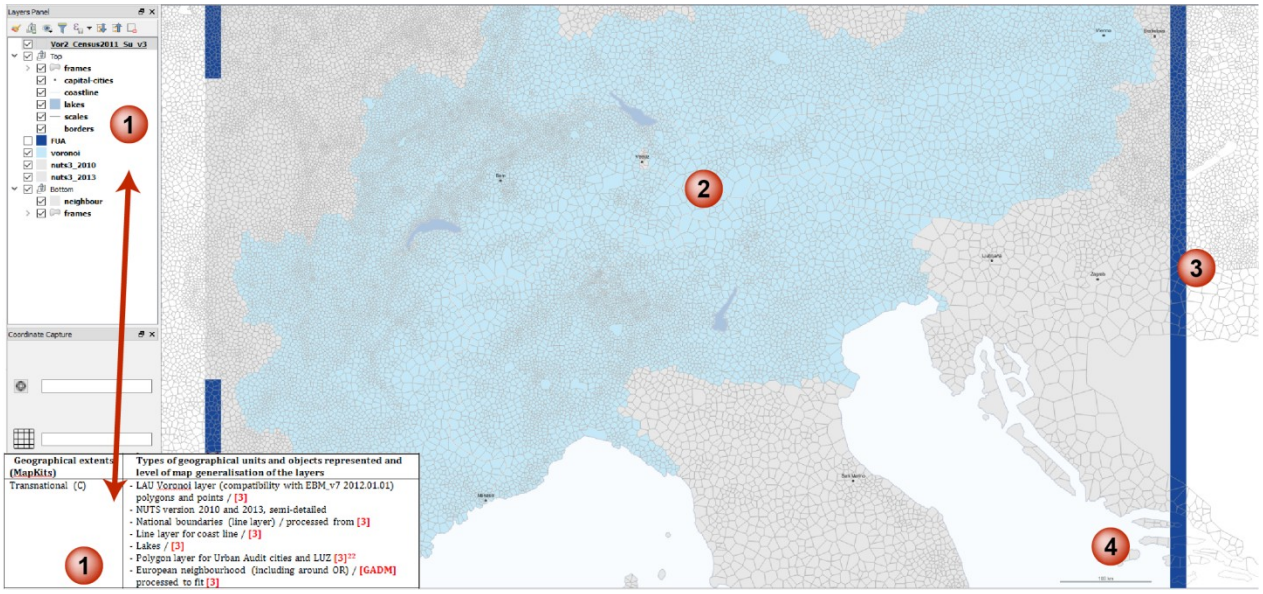
The document specifies all the layers that should be included in the ESPON MapKits. First step consists in selecting in the "input" folder all the relevant layers (1 in Figure 3 -35).

b) Selection of territories

This second step is adapted for ESPON Transnational MapKits, where not all LAU2 units of the bounding box are displayed in the map. In such cases, it implies to select territorial units to be included in the MapKit (2 in Figure 3 -35).

In the ESPON 2020 MapKits project, it is done with the R programme (prg folder of the delivery). But it could be done also directly within a GIS environment. NUTS0-1-2-3 codes must be specified for each LAU2 territorial unit.

Figure 3-35: From the input layers to the MapKits, a GIS process to follow



c) Layers intersection

All selected input layers have to be intersected with the layout (3 in Figure 3 -35) in order to delete territories located outside of the template.

d) Scale layer creation

A scale shapefile is created. The right side of the scale bar is always located at the same distance from the right blue stripe (4 in Figure 3 -35)

3.4 Layer organisation and styles definition in the GIS

Whatever the software used, it is compulsory to import all layers to be included in the ESPON MapKit in the GIS. It is also mandatory to define their graphic styles.

Layers are imported and superposed as follow, from the top to the bottom of the map template, with their associated styles (Table 3 -1, Table 3 -2, Table 3 -3)

Due to software implementation and bounding box extent, it is possible that minor differences occur between software implementations. The table below summarizes the styles of all the layers and font implemented in ESPON MapKits. These styles (in particular line/dot width) fit with a A4 templated map (with 20 mm margins on the top, on the left and on the right).

Table 3-1: Layer styles - Layout (top)

Layer	Background colour (RGB)	Line colour (RGB)	Line/dot width in mm	Line /dot width in pixels
Capital cities (dots)	(53, 53, 53)		1 mm	2,835 pt
Frames (blue stripes)	(3, 78, 162)	None	None	None
Frames (boxes)	None	(187, 189, 192)	0,2 mm	0,567 pt
Scales	None	(76, 80, 81)	0,15 mm	0,425 pt
Coastline	None	(210, 219, 232)	0,3 mm	0,85 pt
North Cyprus	(255, 255, 255)	None	None	None
National Borders	None	(255, 255, 255)	0,3 mm	0,85 pt
Regional Borders (Transnational MapKits)	None	(255, 255, 255)	0,1 mm	0,2835 pt
Disputed Borders	None	(200, 200, 200)	0,2 mm	0,567 pt
Lakes	(247, 252, 254)	None	None	None
Remote areas (non ESPON Space)	(230, 230, 230)	None	None	None

Table 3-2: Layer styles - Statistical layers used to create ESPON thematic maps

Layer	Background colour (RGB)	Line colour (RGB)	Line/dot width in mm	Line/dot width in pixels
LAU2-Voronoi / GREAT Europe GREAT World layers	(194, 232, 247)	Optional – If displayed (255, 255, 255)	Optional – if displayed 0,1 mm	Optional – if displayed 0,2835 pt
FUA polygons (for transnational Mapkits)	(3, 78, 162)	Optional – If displayed (255, 255, 255)	Optional – if displayed 0,1 mm	Optional – if displayed 0,2835 pt

Table 3-3: Layer styles – Layout (bottom)

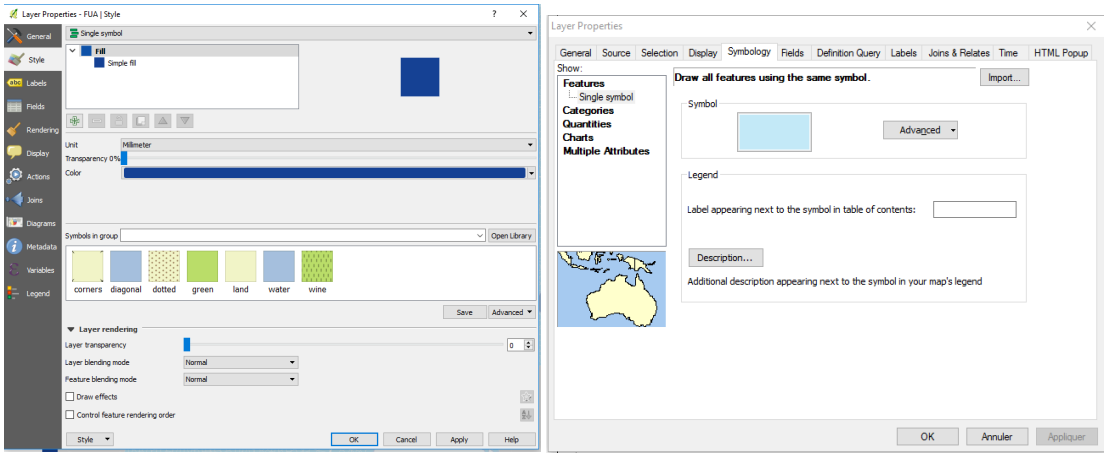
Layer	Background colour (RGB)	Line colour (RGB)	Line/dot width in mm	Line/dot width in pixels
Frames (boxes)	(247, 252, 254)	None	None	None
Countries	(30, 230, 230)	None	None	None
Frames (main / sea)	(247, 252, 254)	None	None	None

The table below summarizes the font to be used in the ESPON MapKits. For the readability of the map, elements must be graphically hierarchised.

Name	Font colour (RGB)	Font and size
Title	(3, 78, 162)	Arial, 14, bold
Legend title	(0, 0, 0)	Arial, 8, bold
Legend values	(0, 0, 0)	Arial, 6, normal
ESPON copyright	(53, 53, 53)	Arial, 8, bold
Metadata box	(26, 23, 23)	Arial, 6, normal
Capital cities	(0, 0, 0)	Arial, 5.5, normal
Scale (main)	(76, 80, 81)	Arial, 5.5, normal
Overseas territories + Malta and Liechtenstein names	(0, 0, 0)	Arial, 4 , normal
Legend (overseas territories + Malta and Liechtenstein)	(0, 0, 0)	Arial, 3.5 , normal

In QGIS, layer styles and fonts are managed in the layer properties (Figure 3 -36, on the left for QGIS and on the right for ArcGIS).

Figure 3-36: Layer properties in QGIS and ArcGIS



3.5 Labels (capital cities) edition and exceptions

By default, all labels of capital cities are displayed 1 mm from the top of the dot. Unfortunately in some cases (Vienna-Bratislava-Tallin in the ESPON Narrow MapKit) labels overlap. In these cases, exceptions are edited (SQL query on capital name) in order to display the labels

on the right or 1 mm on the left of the dot position. Screenshots in QGIS and ArcGIS are shown below.

Figure 3-37: Rule-based labeling in QGIS : Vienna and Ljubljana labels are placed 1 mm from the left of the dot instead of from the top

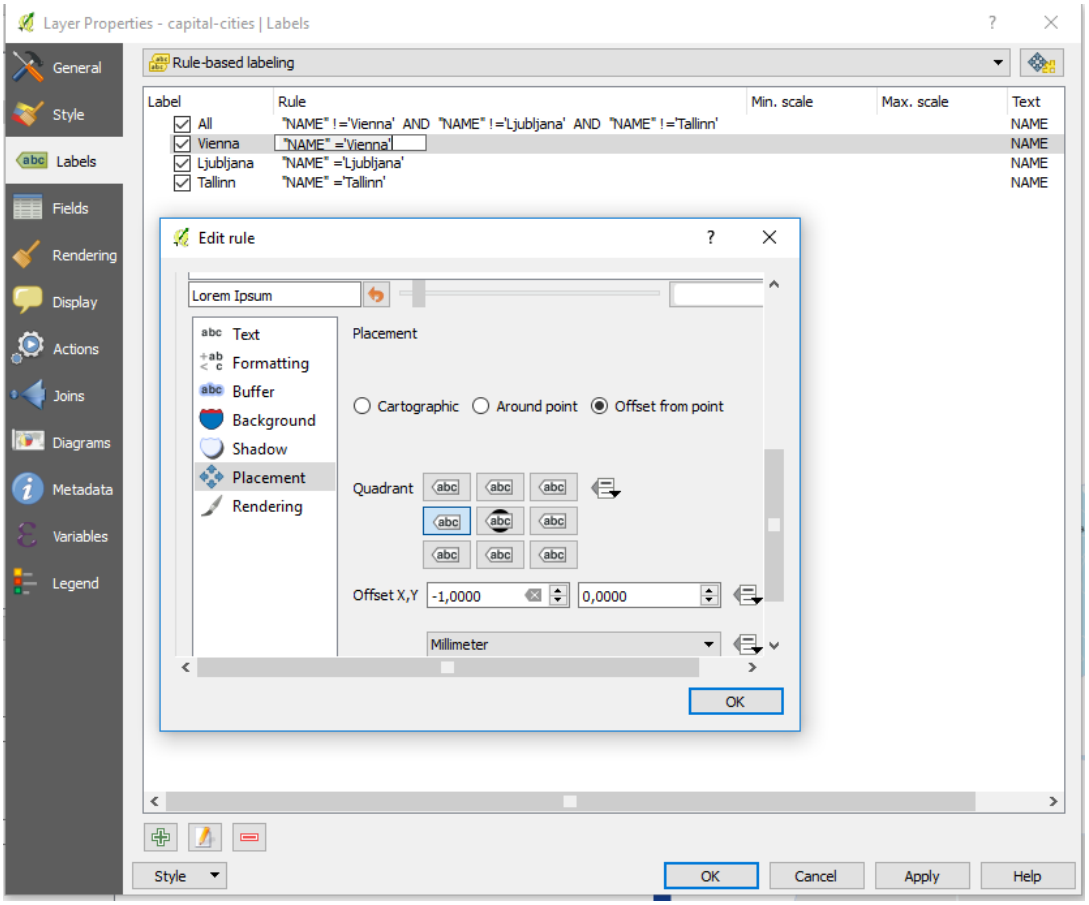
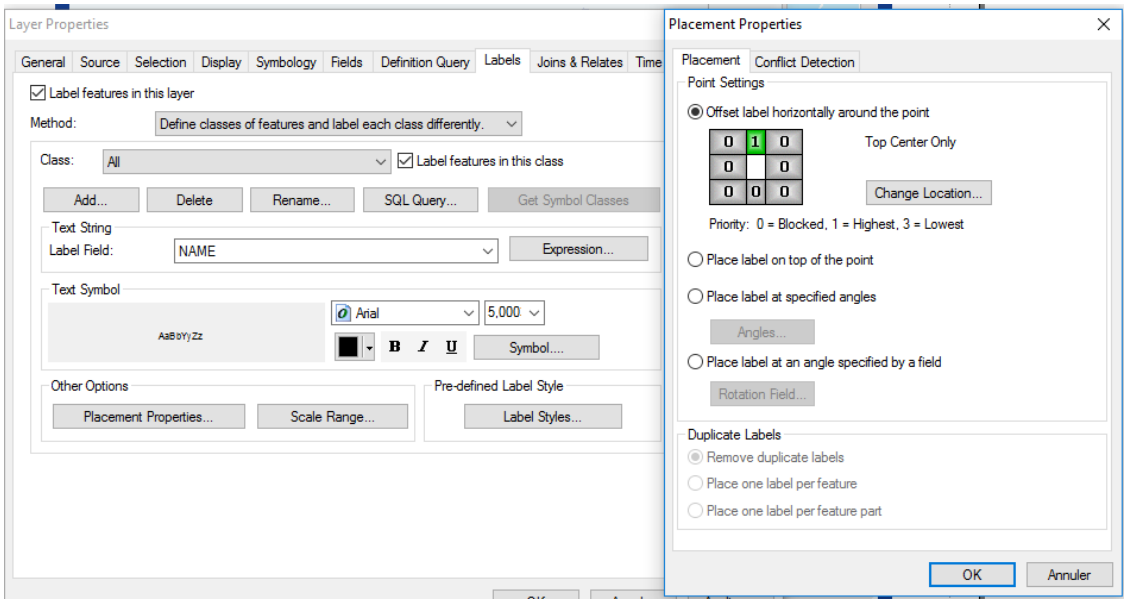
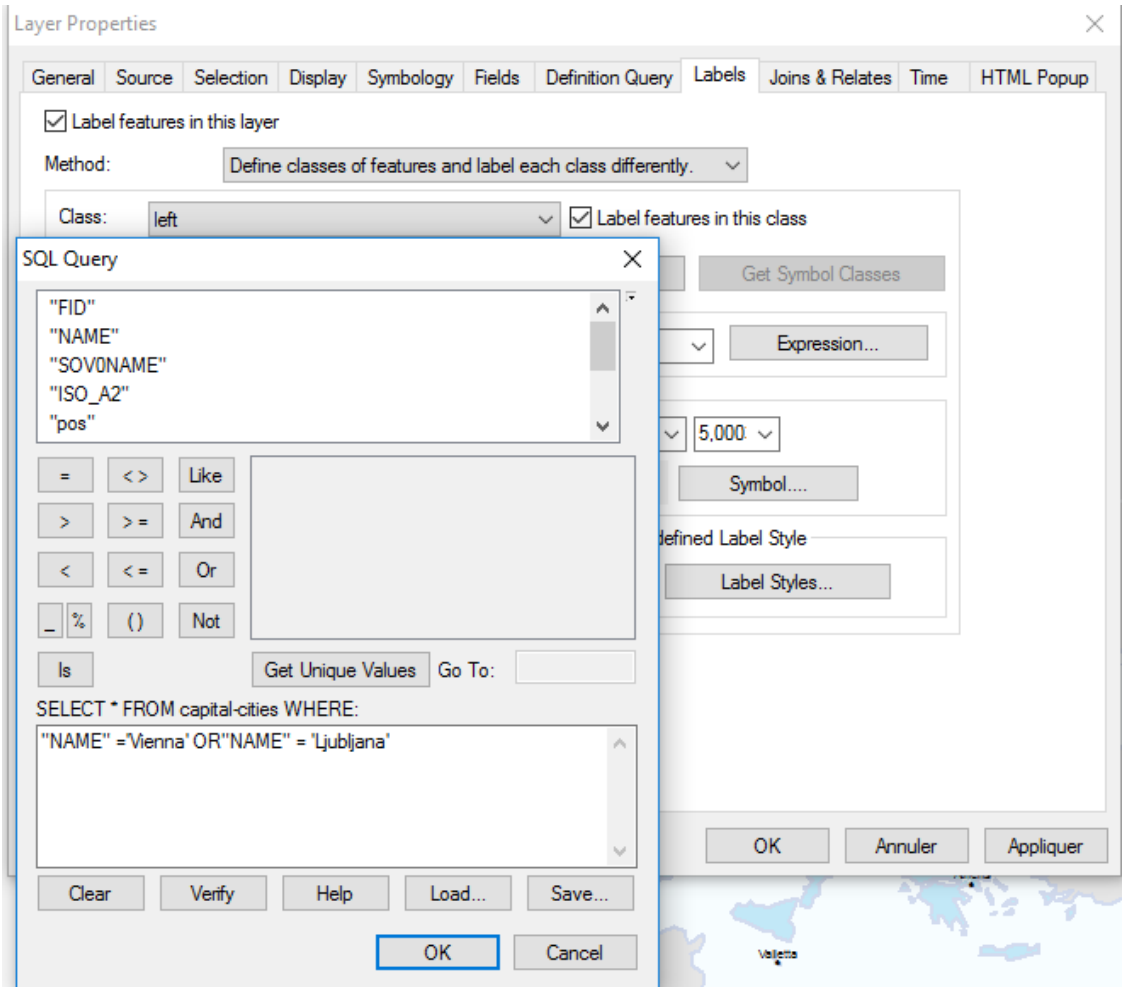


Figure 3-38: Rule-based labeling in ArcGIS : Vienna and Ljubljana labels are placed 1 mm from the left of the dot instead of from the top

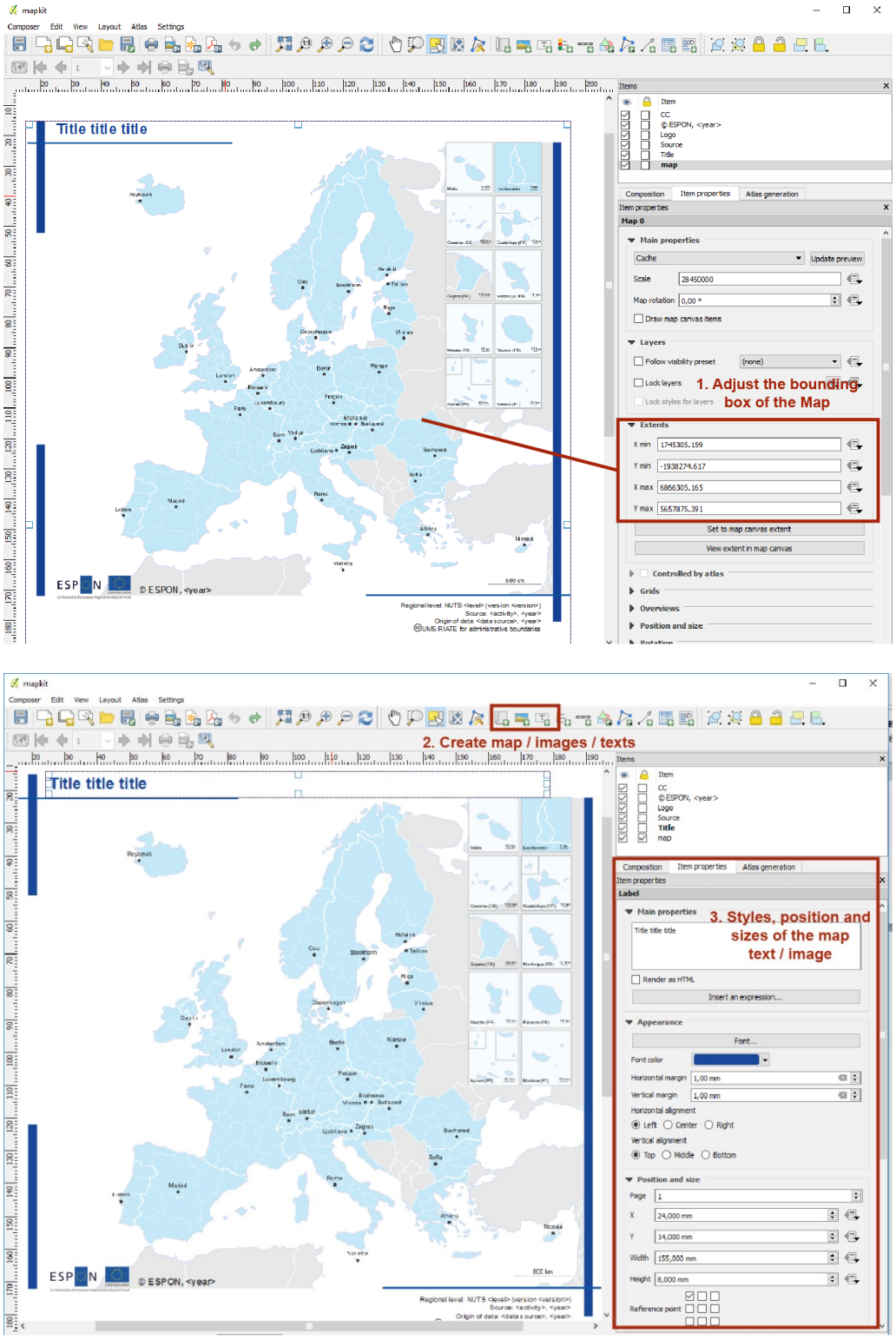



Layer Properties
General
Source
Selection
Display
Symbology
Fields
Definition Query
Labels
Joins & Relates
Time
HTML Popup
Label features in this layer
Method:
Define classes of features and label each class differently.
Class:
left
Label features in this class
Get Symbol Classes
Expression...
5,000
Symbol...
defined Label Style
Label Styles...
OK
Annuler
Appliquer
SQL Query
"FID"
"NAME"
"SOV0NAME"
"ISO_A2"
"pos"
=
< >
Like
>
> =
And
<
< =
Or
_
%
( )
Not
Is
Get Unique Values
Go To:
SELECT * FROM capital-cities WHERE:
"NAME" ='Vienna' OR"NAME" = 'Ljubljana'
Clear
Verify
Help
Load...
Save...
OK
Cancel

3.6 ESPON Template design in QGIS

Figure 3 -39 highlights the main steps to follow to implement the ESPON MapKit in design QGIS.

Figure 3-39: Implementation process of the ESPON 2020 MapKit in QGIS

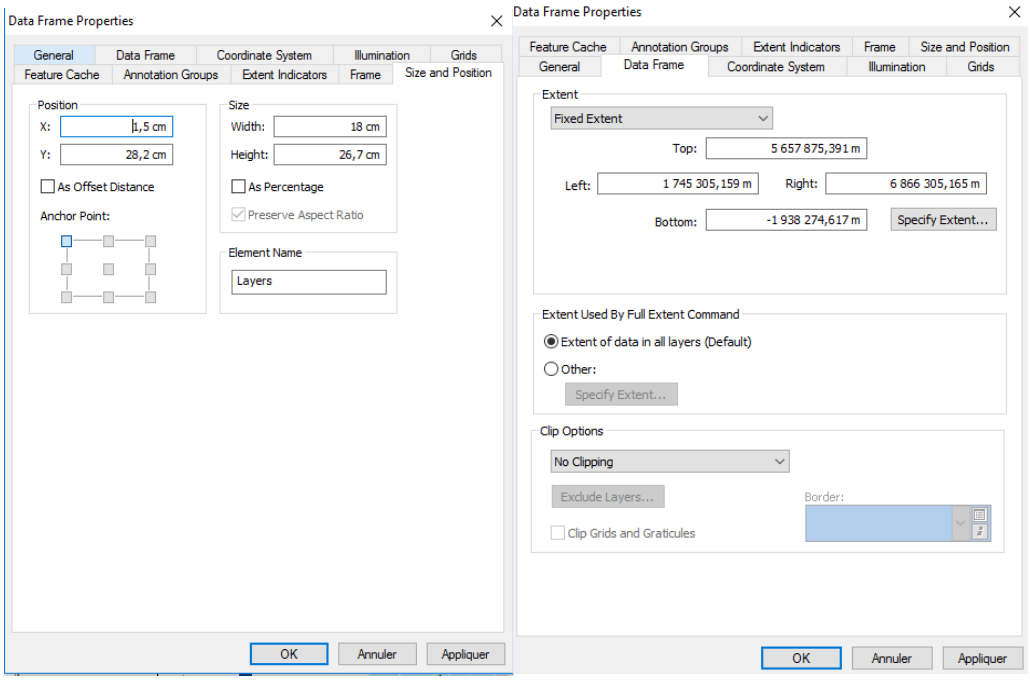


One has to print a new Map Composer (named “ mapkit”). Firstly, the map must be imported and the bounding box must be adjusted to the A4 page (point 1 in the Figure above). Secondly, all the requested elements of the map template must be imported/created (ESPON logos and European Commission in a vector format, title, data sources, point 2 in the figure above). Finally, the styles, sizes and position in the map template must be fixed.

3.7 ESPON Template design in ArcGIS

In ArcGIS, the process consists in defining the size and the position of the A4 page (Data Frame Properties -> Size and Positions[5]). Then, it requires specifying the extent of the map template (Data Frame Properties -> Data Frame).

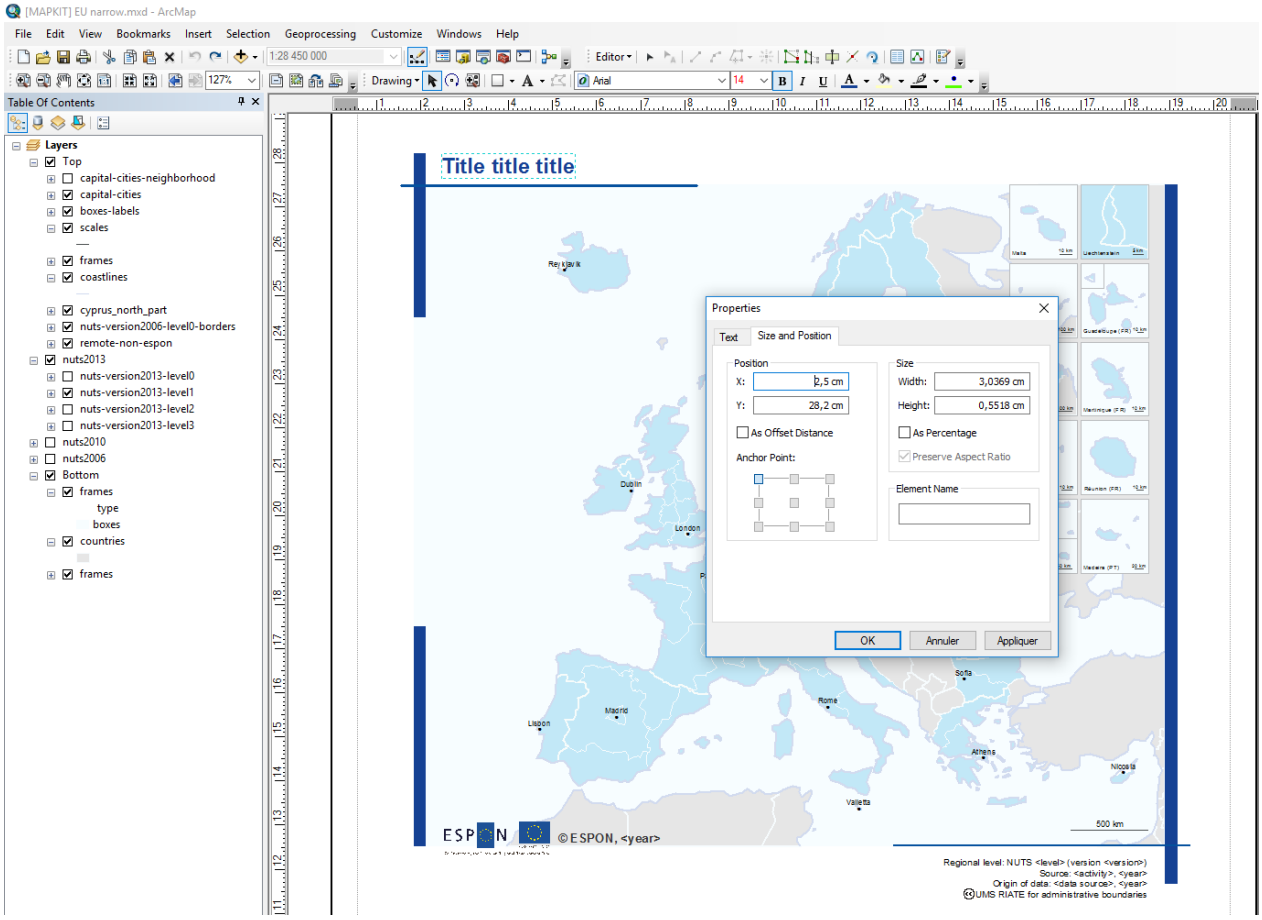
Figure 3-40: Implementation process of the ESPON 2020 MapKit in ArcGIS



Finally, logos and texts are imported/created in the ESPON map template. To do this, one first has to activate the “layout view” in ArcGIS. Text size, font and colours can then be adapted. One needs to ensure that all the graphical elements are positioned in the same way than as in the QGIS template (Figure 3 -41).

Figure 3-41: Set the size and position properties of all the graphical elements in ArcGIS

5 Be careful, the reference position (X/Y) of the graphical elements are not specified in the same way in QGIS (reference : top-left) and in ArcGIS (reference : back-left). It must consequently be adjusted manually to ensure a perfect harmonization between the two templates.



3.8 Apply relative paths, duplicate MapKit folder and copy-paste .qgs and .mxd documents

Thanks to the relative path option, which points to the MapKit folder (Figure 3 -42), it is then possible to copy/paste the graphics folder, the .mxd and the .qgs files to avoid wasting time to affect systematically the styles to each layer of each MapKit[6]. Applying relative paths also makes it possible for every user to open the .qgs and .mxd files without specifying the file path to access to the shapefiles of the ESPON Map templates.

However, the process of layer selection/intersection must be processed for each MapKit creation systematically (shapefiles included in the .shp folder) and all the layers must be named in the same wa

Figure 3-42 - Store relative paths in QGIS (left) and ArcGIS (right)

6 It requires only to rename the name of the document, for example [MAPKIT] Transnat-Alpine.qgs instead of [MAPKIT] Transnat-BSR.qgs

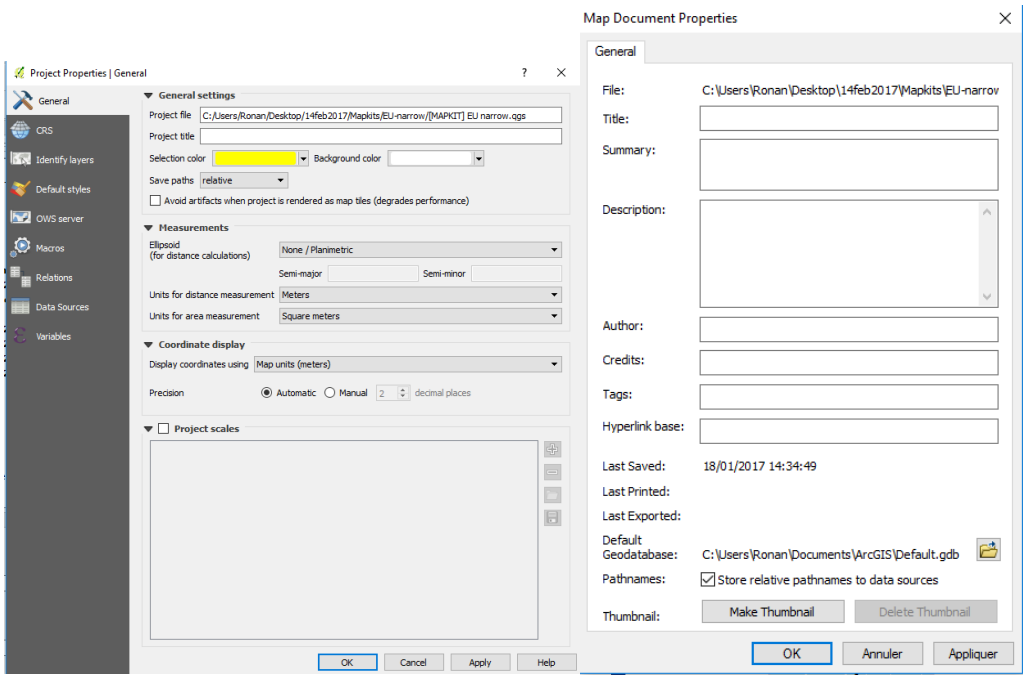
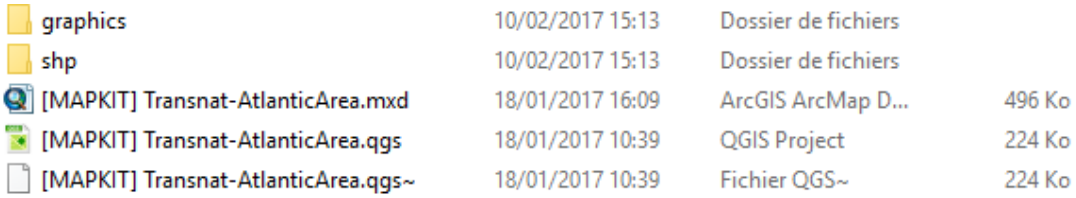


Figure 3-43 - Copy-paste .mxd, .qgs and graphics folder (the shp folder must be created systematically with intersected layers)

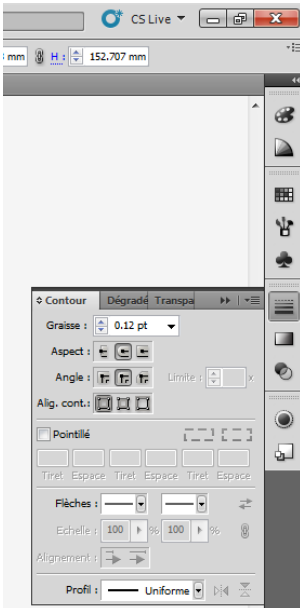


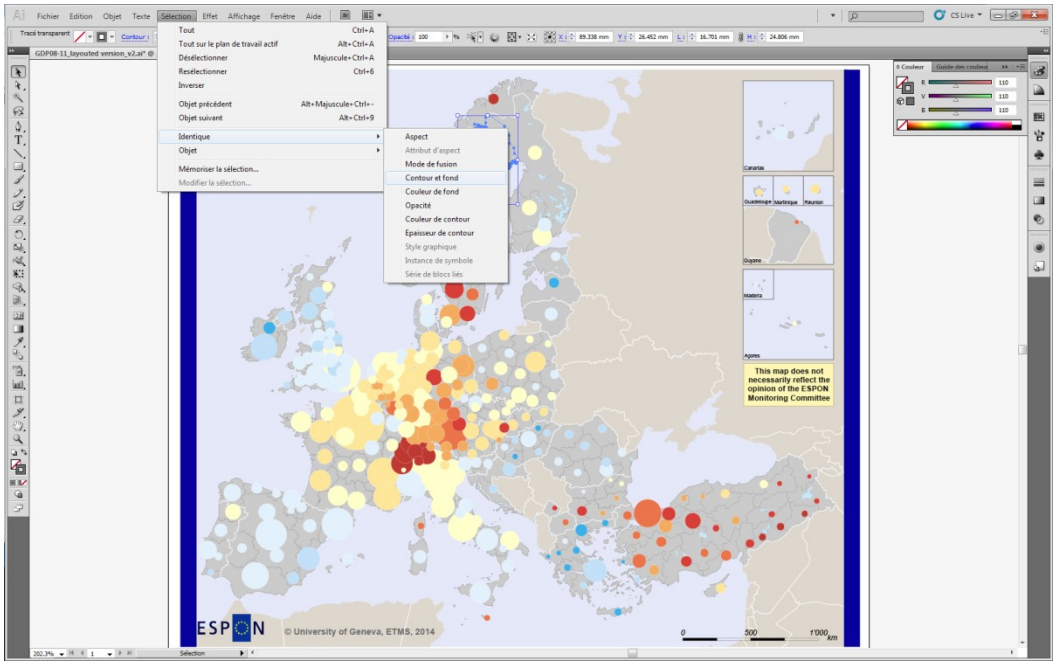
3.9 ESPON Template design in Adobe Illustrator

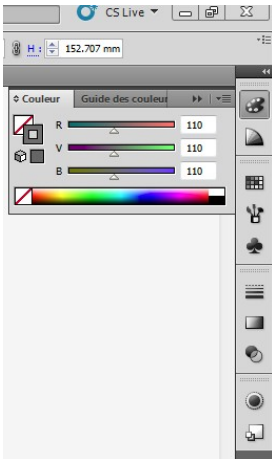
ESPON 2020 MapKits are exported from ArcGIS to Adobe Illustrator (export options in ArcGIS). One then has to check the styles in the Adobe Illustrator document. The export options can change the styles (police, colour, border width) initially set in the ArcGIS and QGIS templates. Figure 3 -44 shows how one can apply homogeneous styles in Adobe Illustrator.

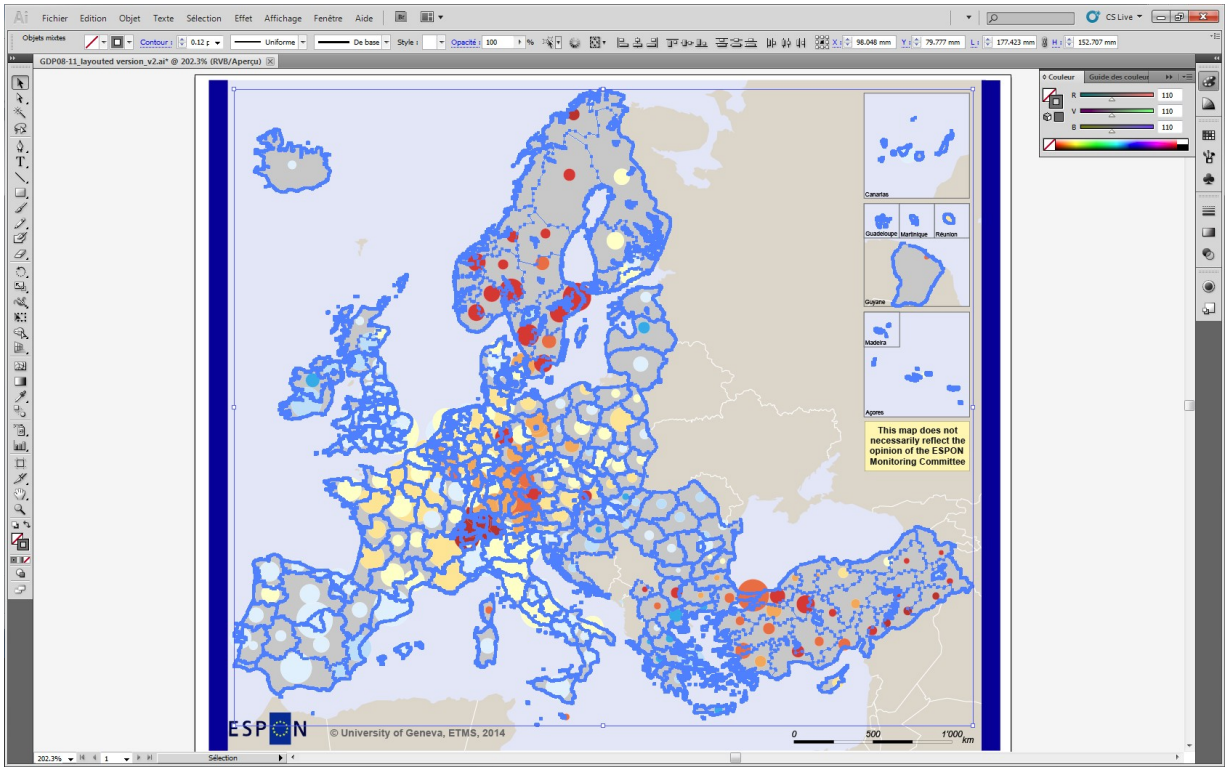
Note that all the MapKits delivered to the ESPON EGTC in .ai format are carefully checked in this way.

Figure 3-44 - Adjust graphic styles in Adobe Illustrator Environment






CS Live
152.707 mm
Couleur
Guide des couleurs
R
V
B
110
110
110


Fichier Edition Objet Texte Sélection Effet Affichage Fenêtre Aide
CS Live
GDP08-11_layouted version_v2.ai* @ 202.3% (RVB/Aperçu)
Canarias
Guadeloupe
Martinique
Réunion
Guyane
Madeira
Azores
This map does not necessarily reflect the opinion of the ESPON Monitoring Committee
ESPON
© University of Geneva, ETMS, 2014
0
500
1'000
km

*

References

Ben Rebah M., Plumejeaud C., Ysebaert R., Peeters D. (2011), Modeling territorial changes and time series database building process: empirical approach and applications, Technical Report, ESPON Database Project. Available at: https://www.espon.eu/main/Menu_Projects/Menu_ESPON2013Projects/Menu_ScientificPlatform/espondatabase2013.html

Reference documents for territorial nomenclatures (Voronoi / LAU2)

EBM Version 6.0 : http://www.eurogeographics.org/sites/default/files/EBM_v60_Specification.pdf

EBM Version 6.2: http://ec.europa.eu/eurostat/web/gisco/geodata/reference-data/administrative-units-statistical-units/census#censusunits11

EBM Version 6.4/ http://www.eurogeographics.org/sites/default/files/EBM_v10_Specification.pdf

Reference documents for territorial nomenclatures (GREAT Europe / NUTS)

Eurostat, Regions in the European Union - Nomenclature of territorial units for statistics http://ec.europa.eu/eurostat/fr/web/nuts/publications

Eurostat, List of changes between NUTS version, http://ec.europa.eu/eurostat/en/web/nuts/history

Eurostat, repository of georeferenced layers, http://ec.europa.eu/eurostat/fr/web/gisco/geodata/reference-data/administrative-units-statistical-units

Other useful ressources for creating layers in the ESPON Context

GADM : http://gadm.org

Natural Earth : http://www.naturalearthdata.com/

User manuals

Adobe Illustrator CC Help (2016). Available at: https://helpx.adobe.com/pdf/illustrator_reference.pdf

ArcGIS Desktop Documentation. Available at: http://desktop.arcgis.com/fr/documentation/

QGIS User Manual. Available at: http://docs.qgis.org/2.14/en/docs/user_manual/

List of Annexes

Annex 1 – R Programme – Build a ESPON Map template correctly sized

```
##############################################
#        Install required librairies        #
##############################################

library("rgeos")
library("rgdal")


##############################################
#        Function BuildTemplate              #
##############################################

# This function has been designed following a template designed in Adobe Illustrator and provided by ESPON EGTC
# It takes in entry the geographical coordinates (xmin, xmax, ymin, ymax) of the bounding box required and the cartographic projection system

buildTemplate <- function(xmin, xmax, ymin, ymax, prj){

# Template parameters
width <- (xmax - xmin) / 60 #  Blue stripes width
extralength <- width * 3 # Height of the extra length on the top and on the bottom of the template
lengthtopleft <- (ymax - ymin) / 5 # Height of the stripe exceeding in top left
lengthbottomleft <- (ymax - ymin) / 3 # Height of the strip exceeding in bottom left
linewidth <- (xmax - xmin) / 300 # Width of the little horizontal stripes
linesize <- (xmax - xmin) / 2.5 # Length of the little horizontale stripes

# Main frame
id <- 1
type <- "mainframe"
rect.sp <- rgeos::readWKT(paste("POLYGON((",xmin," ",ymin,",",xmin," ",ymax,",",xmax," ",ymax,",",xmax," ",ymin,",",xmin," ",ymin,"))",sep=""))
x1 <- xmin - width ; x2 <- xmin ; y1 <- ymin + lengthbottomleft ; y2 <- ymax - lengthtopleft
r.sp <- rgeos::readWKT(paste("POLYGON((",x1," ",y1,",",x1," ",y2,",",x2," ",y2,",",x2," ",y1,",",x1," ",y1,"))",sep=""))
rect.sp <- gUnion(rect.sp, r.sp)
rect.spdf <- SpatialPolygonsDataFrame(rect.sp, data.frame(id=id, type=type))

# Band right
id <- id + 1
type <- "stripe"
x1 <- xmax ; x2 <- xmax + width ; y1 <- ymin  - extralength; y2 <- ymax
r.sp <- rgeos::readWKT(paste("POLYGON((",x1," ",y1,",",x1," ",y2,",",x2," ",y2,",",x2," ",y1,",",x1," ",y1,"))",sep=""))
r.spdf <- SpatialPolygonsDataFrame(r.sp, data.frame(id=id, type=type))
row.names(r.spdf) <- as.character(id)
rect.spdf <- rbind(rect.spdf,r.spdf)

# Band top left
id <- id + 1
type <- "stripe"
x1 <- xmin - width ; x2 <- xmin ; y1 <- ymax - lengthtopleft; y2 <- ymax + extralength
r.sp <- rgeos::readWKT(paste("POLYGON((",x1," ",y1,",",x1," ",y2,",",x2," ",y2,",",x2," ",y1,",",x1," ",y1,"))",sep=""))
r.spdf <- SpatialPolygonsDataFrame(r.sp, data.frame(id=id, type=type))
row.names(r.spdf) <- as.character(id)
rect.spdf <- rbind(rect.spdf,r.spdf)

# Band bottom left
id <- id + 1
type <- "stripe"
x1 <- xmin - width ; x2 <- xmin ; y1 <- ymin ; y2 <- ymin + lengthbottomleft
```

```
r.sp <- rgeos::readWKT(paste("POLYGON((",x1," ",y1,",",x1," ",y2,",",x2," ",y2,",",x2," ",y1,",",x1," ",y1,"))",sep=""))
r.spdf <- SpatialPolygonsDataFrame(r.sp, data.frame(id=id, type=type))
row.names(r.spdf) <- as.character(id)
rect.spdf <- rbind(rect.spdf,r.spdf)

# Small line (top)
id <- id+1
type <- "line"
x1 <- xmin - width*2 ; x2 <- xmin - width*2 + linesize  ; y1 <- ymax - linewidth/2 ; y2 <- ymax - linewidth/2
l.sp  <- rgeos::readWKT(paste("LINESTRING(",x1," ",y1,",",x2," ",y2,")",sep=""))
r.sp <- gBuffer(l.sp, byid=FALSE, id=NULL, width=linewidth/2, quadsegs=5, capStyle="ROUND",joinStyle="ROUND", mitreLimit=1.0)
row.names(r.sp) <- "1"
r.spdf <- SpatialPolygonsDataFrame(r.sp, data.frame(id=id, type=type))
row.names(r.spdf) <- as.character(id)
rect.spdf <- rbind(rect.spdf,r.spdf)

# Small line (bottom)
id <- id+1
type <- "line"
x1 <- xmax + width*2 - linesize ; x2 <- xmax + width*2  ; y1 <- ymin + linewidth/2 ; y2 <- ymin + linewidth/2
l.sp  <- rgeos::readWKT(paste("LINESTRING(",x1," ",y1,",",x2," ",y2,")",sep=""))
r.sp <- gBuffer(l.sp, byid=FALSE, id=NULL, width=linewidth/2, quadsegs=5, capStyle="ROUND",joinStyle="ROUND", mitreLimit=1.0)
row.names(r.sp) <- "1"
r.spdf <- SpatialPolygonsDataFrame(r.sp, data.frame(id=id, type=type))
row.names(r.spdf) <- as.character(id)
rect.spdf <- rbind(rect.spdf,r.spdf)

proj4string(rect.spdf) <- prj

return (rect.spdf)
}


################################################
#       EXAMPLE - ESPON NARROW MAPKIT       #
################################################

# Set working directory
setwd("/xxx")

# Set ESPON Narrow MapKit bbox and projection
prj <- "+proj=laea +lat_0=52 +lon_0=10 +x_0=4321000 +y_0=3210000 +ellps=GRS80 +units=m +no_defs"
xmin <- 1930663
ymin <- 1189404
xmax <- 6702156
ymax <- 5456938

# Launch builTemplate function
template.spdf <- buildTemplate(xmin, xmax, ymin, ymax, prj)

# Export
# Set the folder
folder <- "../Mapkits/EU-narrow/shp/layout"
dir.create(folder)

# Write the shapefile in selected folder
writeOGR(obj=template.spdf, dsn=folder, layer="template", driver="ESRI Shapefile",overwrite_layer=TRUE,verbose=F)
```

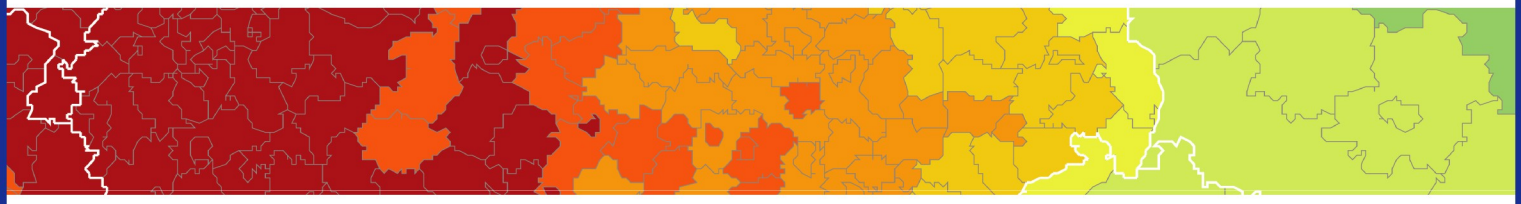
Annex 2 – R Programme – Select the layers and intersect with the template layer – Alpine Space example

```
# This R Programme build the ESPON Transnational MapKit for the Alpine Space

# Aim of the Programme
# 1 – Build the template shapefile using the bounding box parameters
# 2 – Select the territories to be displayed in the MapKit (SQL request)
# 3 – Build the layout (cf Annex 1)
# 4 – Create the folder architecture to export the shapefiles correctly
# 5 – Import input layers and intersect with the template in the appropriate cartographic projection system
# 6 – Create a scale shapefile after having specified the scale value

# Requires R librairies
library("rgdal")
library("mapinsetr")
library("rgeos")
library("maptools")
library("foreign")
library("sqldf")
library("reshape2")

# Set working directory and import the BuildEmptyTemplate function for creating the template shapefile
setwd("/home/nlambert/Documents/ESPON/ESPON-MapKits-2016/prg")
source("sources/BuildEmptyTemplate.R")

# params
# Scale value (m)
scaleVal <- 100000

## Territories selection
sql <- "select CENS_ID from tb where SN0_2013 = 'AT' OR SN0_2013 = 'CH' OR SN0_2013 = 'SI' OR SN2_2013 = 'FR42' OR SN2_2013 = 'FR43' OR SN2_2013 = 'FR71' OR SN2_2013 = 'FR82' OR SN2_2013 = 'ITC1' OR SN2_2013 = 'ITC2' OR SN2_2013 = 'ITC3' OR SN2_2013 = 'ITC4' OR SN2_2013 = 'ITH1' OR SN2_2013 = 'ITH2' OR SN2_2013 = 'ITH3' OR SN2_2013 = 'ITH4' OR SN2_2013 = 'DE13' OR SN2_2013 = 'DE14' OR SN2_2013 = 'DE27' OR SN2_2013 = 'DE21'"

## Bounding box of the layout
xmin <- 3819145
ymin <- 2192065
xmax <- 4867673
ymax <- 2902955

folder <- "Transnat-Alpine"
prj <- "+proj=laea +lat_0=52 +lon_0=10 +x_0=4321000 +y_0=3210000 +ellps=GRS80 +units=m +no_defs"

# folder
dir.create(paste("../Mapkits/",folder,sep=""))
shpfolder <- paste("../Mapkits/",folder,"/shp",sep="")
dir.create(shpfolder)

# template
template.spdf <- buildTemplate(xmin, xmax, ymin, ymax, prj)
writeOGR(obj=template.spdf, dsn=shpfolder, layer="template", driver="ESRI Shapefile",overwrite_layer=TRUE,verbose=F)

# Voronoi
voronoi.spdf <- readOGR(dsn = "../input/Voronoi(new)/Census2011_LAU", layer = "Vor2_Census2011_Su_v3", verbose = TRUE)
voronoi.spdf@data$SN3_2013 <- voronoi.spdf@data$NUTS_3_202
voronoi.spdf@data$SN2_2013 <- substr(voronoi.spdf@data$NUTS_3_202,1,4)
voronoi.spdf@data$SN1_2013 <- substr(voronoi.spdf@data$NUTS_3_202,1,3)
voronoi.spdf@data$SN0_2013 <- substr(voronoi.spdf@data$NUTS_3_202,1,2)
```

```
com.spdf <- voronoi.spdf
tb <- as.data.frame(com.spdf@data)
result <- sqldf(sql)
result$area <- 1
com.spdf@data <- data.frame(com.spdf@data, result[match(com.spdf@data[,"CENS_ID"],
result[,"CENS_ID"]),])
com.spdf <- com.spdf[!is.na(com.spdf@data$area),]
writeOGR(obj=com.spdf, dsn=shpfolder, layer="voronoi", driver="ESRI
Shapefile",overwrite_layer=TRUE,verbose=F)

# neighbour
neighbour.spdf <- readOGR(dsn = "../input/Voronoi(new)/Neighbour_NUTS0", layer =
"Vor2_NUTS0_Context_Su", verbose = TRUE)
wbtr.spdf <- readOGR(dsn = "../input/Voronoi(new)/WB&TR_NUTS0", layer =
"Vor2_NUTS0_OtherESPONspace_Su", verbose = TRUE)
microstates.spdf <- readOGR(dsn = "../input/Voronoi(new)/MicroStates", layer = "Vor2_MicroStates",
verbose = TRUE)
Other_worldCC.spdf <- readOGR(dsn = "../input/Voronoi(new)/Other_worldCC", layer =
"Ne_50m_OtherCC_Su", verbose = TRUE)
neighbour.spdf <- inset_rbinder(l = list(neighbour.spdf,wbtr.spdf,microstates.spdf,Other_worldCC.spdf))
sr <- gIntersection(neighbour.spdf, template.spdf[template.spdf@data$type=="mainframe",],
byid=TRUE)
if(!is.null(sr)){
  ids <- (do.call('rbind', (strsplit(as.character(row.names(sr))," "))))[,1]
  row.names(sr) <- ids
  data <-as.data.frame(neighbour.spdf@data[row.names(neighbour.spdf@data) %in% ids,])
  colnames(data) <- colnames(neighbour.spdf@data)
  row.names(data) <- ids
  neighbour.spdf <- sp::SpatialPolygonsDataFrame(Sr = sr, data = data, match.ID = TRUE)
            writeOGR(obj=neighbour.spdf, dsn=shpfolder, layer="neighbour", driver="ESRI
Shapefile",overwrite_layer=TRUE,verbose=F)
}else {rm(neighbour.spdf)}

# Lakes
lakes.spdf <- readOGR(dsn = "../input/Voronoi(new)/Lakes", layer = "Vor2_Lakes_Su", verbose =
TRUE)
sr <- gIntersection(lakes.spdf, template.spdf[template.spdf@data$type=="mainframe",], byid=TRUE)
if(!is.null(sr)){
  ids <- (do.call('rbind', (strsplit(as.character(row.names(sr))," "))))[,1]
  row.names(sr) <- ids
  data <-as.data.frame(lakes.spdf@data[row.names(lakes.spdf@data) %in% ids,])
  colnames(data) <- colnames(lakes.spdf@data)
  row.names(data) <- ids
  lakes.spdf <- sp::SpatialPolygonsDataFrame(Sr = sr, data = data, match.ID = TRUE)
              writeOGR(obj=lakes.spdf, dsn=shpfolder, layer="lakes", driver="ESRI
Shapefile",overwrite_layer=TRUE,verbose=F)
}else {rm(lakes.spdf)}

# NUTS 2013
nuts3_2013.spdf <- readOGR(dsn = "../input/Voronoi(new)", layer = "NUTS3_2013", verbose = TRUE)
sr <- gIntersection(nuts3_2013.spdf, template.spdf[template.spdf@data$type=="mainframe",],
byid=TRUE)
if(!is.null(sr)){
  ids <- (do.call('rbind', (strsplit(as.character(row.names(sr))," "))))[,1]
  row.names(sr) <- ids
  data <-as.data.frame(nuts3_2013.spdf@data[row.names(nuts3_2013.spdf@data) %in% ids,])
  colnames(data) <- colnames(nuts3_2013.spdf@data)
  row.names(data) <- ids
  nuts3_2013.spdf <- sp::SpatialPolygonsDataFrame(Sr = sr, data = data, match.ID = TRUE)
          writeOGR(obj=nuts3_2013.spdf, dsn=shpfolder, layer="nuts3_2013", driver="ESRI
Shapefile",overwrite_layer=TRUE,verbose=F)
}

# NUTS 2010
nuts3_2010.spdf <- readOGR(dsn = "../input/Voronoi(new)", layer = "NUTS3_2010", verbose = TRUE)
sr <- gIntersection(nuts3_2010.spdf, template.spdf[template.spdf@data$type=="mainframe",],
byid=TRUE)
```

```
if(!is.null(sr)){
  ids <- (do.call('rbind', (strsplit(as.character(row.names(sr))," "))))[,1]
  row.names(sr) <- ids
  data <-as.data.frame(nuts3_2010.spdf@data[row.names(nuts3_2010.spdf@data) %in% ids,])
  colnames(data) <- colnames(nuts3_2010.spdf@data)
  row.names(data) <- ids
  nuts3_2010.spdf <- sp::SpatialPolygonsDataFrame(Sr = sr, data = data, match.ID = TRUE)
            writeOGR(obj=nuts3_2010.spdf, dsn=shpfolder, layer="nuts3_2010", driver="ESRI
Shapefile",overwrite_layer=TRUE,verbose=F)
}

# FUA
FUA.spdf <- readOGR(dsn = "../input/Voronoi(new)", layer = "FUA", verbose = TRUE)
all <- FUA.spdf
studyarea <- gBuffer(com.spdf,width=0,byid = F)
studyarea <- gBuffer(studyarea,width=-1000,byid = F)
tmp <- gWithinDistance(FUA.spdf, spgeom2 = studyarea, dist=0,byid=TRUE, hausdorff=FALSE,
densifyFrac = NULL)
tmp <- melt(tmp)
head(tmp)
FUA.spdf@data <- data.frame(FUA.spdf@data, tmp[match(row.names(FUA.spdf),
tmp[,"Var2"]),"value"])
FUA.spdf <- FUA.spdf[FUA.spdf@data[,3],]
FUA.spdf@data <- FUA.spdf@data[,c(1,2)]
writeOGR(obj=FUA.spdf, dsn=shpfolder, layer="FUA", driver="ESRI
Shapefile",overwrite_layer=TRUE,verbose=F)

# Borders
nuts0.spdf <- readOGR(dsn = "../input/Voronoi(new)", layer = "NUTS0_2013", verbose = TRUE)
sr <- gIntersection(nuts0.spdf, template.spdf[template.spdf@data$type=="mainframe",], byid=TRUE)
ids <- (do.call('rbind', (strsplit(as.character(row.names(sr))," "))))[,1]
row.names(sr) <- ids
data <-as.data.frame(nuts0.spdf@data[row.names(nuts0.spdf@data) %in% ids,])
colnames(data) <- colnames(nuts0.spdf@data)
row.names(data) <- ids
nuts0.spdf <- sp::SpatialPolygonsDataFrame(Sr = sr, data = data, match.ID = TRUE)
borders.spdf <- getBorders(nuts0.spdf)
writeOGR(obj=borders.spdf, dsn=shpfolder, layer="borders", driver="ESRI
Shapefile",overwrite_layer=TRUE,verbose=F)

# CoastLine
poly <- gBuffer(nuts3_2013.spdf,byid=F,width=1)
if (exists("neighbour.spdf")){poly <- gUnion(poly,neighbour.spdf)}
poly <- gBuffer(poly,byid=F,width=100)
sr <- as(poly , 'SpatialLines')
sr2 <- gBuffer(as(template.spdf[template.spdf@data$type=="mainframe",] , 'SpatialLines'),width=2000)
sr <- gDifference(sr, sr2, byid=FALSE, id=NULL,
drop_lower_td=FALSE,unaryUnion_if_byid_false=TRUE, checkValidity=FALSE)
row.names(sr) <- "1"
data <-as.data.frame(c("id","name"))
data <- as.data.frame("id")
data[1]<-"coastline"
coastline.spdf <- sp::SpatialLinesDataFrame(sr, data = data, match.ID = TRUE)
writeOGR(obj=coastline.spdf, dsn=shpfolder, layer="coastline", driver="ESRI
Shapefile",overwrite_layer=TRUE,verbose=F)

# SCale
bb <- template.spdf[template.spdf@data$type=="mainframe",]@bbox
xmax <- bb[3]
ymin <- bb[2]
w <- bb[3] - bb[1]
h <- bb[4] - bb[2]
xpos <- xmax - scaleVal - w/50
ypos <- ymin + h/50
scale.sp <- rgeos::readWKT(paste("LINESTRING(",xpos," ",ypos,",",xpos+scaleVal," ",ypos,")",sep=""))
scale.spdf <- SpatialLinesDataFrame(scale.sp, data.frame(id="main",
value=paste((scaleVal/1000),"km",sep=" ")))
```

```
proj4string(scale.spdf) <- prj
writeOGR(obj=scale.spdf, dsn=shpfolder, layer="scale", driver="ESRI Shapefile",overwrite_layer=TRUE,verbose=F)

# Capital cities
cities.spdf <- readOGR(dsn = "../input/world", layer = "capital", verbose = TRUE,encoding = " ISO-8859-1")
cities.spdf  <- spTransform(x =  cities.spdf, CRSobj = prj)
sr <- gIntersection(cities.spdf, template.spdf[template.spdf@data$type=="mainframe",], byid=TRUE)
ids <- (do.call('rbind', (strsplit(as.character(row.names(sr))," "))))[,1]
row.names(sr) <- ids
data <-as.data.frame(cities.spdf@data[row.names(cities.spdf@data) %in% ids,])
row.names(data) <- ids
cities.spdf <- sp::SpatialPointsDataFrame(coords = sr@coords, data = data, match.ID = TRUE)
proj4string(cities.spdf) <-  prj
#cities.spdf <- cities.spdf[!cities.spdf@data$NAME %in% c("Dublin","Luxembourg"),]
writeOGR(obj=cities.spdf, dsn=shpfolder, layer="capital-cities", driver="ESRI Shapefile",overwrite_layer=TRUE,verbose=F)

# bbox for QGIS Print Composer
template.spdf@bbox
```



ESPON 2020 – More information

ESPON EGTC