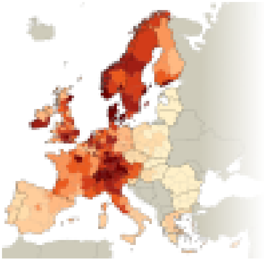
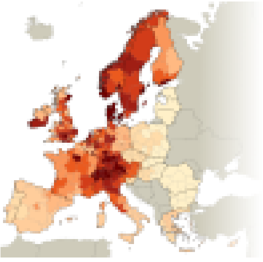
HyperAdmin User's Manual



HyperCarte

HyperCarte Research Group

HyperAdmin User's Manual:



HyperCarte

HyperCarte Research Group

Abstract

This document provides the minimum information on how to build a HyperAtlas dataset `hyp` file. The `.hyp` files are binary files that can be loaded by the HyperAtlas software. HyperAdmin is an integration tool that aims at generating `.hyp` files from a set of input files. This document describes the expected input files for HyperAdmin.

Table of Contents

List of Figures

List of Tables

Chapter 1. Introduction: HyperCarte Research Group

HyperCarte Research Group aims at providing projects and applications for interactive cartography. The projects focus on the development of an easily understood methodology that allows the analysis and visualization of spatial phenomena, taking into account its multiple possible representations.

Statistical observations of the territory are complex, and one representation, directly linked to a precise objective, is the result of a combination of different choices which are relative on one hand to the territories and their geographical scales, to the the statistical indicators on the other hand. This is of interest for researchers as well as for development policy decision-makers.

Thus, the principal innovative aspect of the HyperCarte project lies on this perspective based on the popularization of methods coming from spatial analysis such as the fitting of territorial scales, gradients, discontinuities…. This supposes an effort of multidisciplinary cooperation between geographers and computer scientists in order to create new maps in real time according to the different choices. An important effort has concerned ergonomics and time of calculus.

Main partners of the HyperCarte research group are:

RIATE [UMS 2414] http://www.ums-riate.com

CNRS UMR 8504 Géographie-Cités [UMR 8504] http://www.parisgeo.cnrs.fr

LIG-MESCAL [UMR 5217] http://mescal.imag.fr/

LIG-STeamer [UMR 5217] http://steamer.imag.fr/

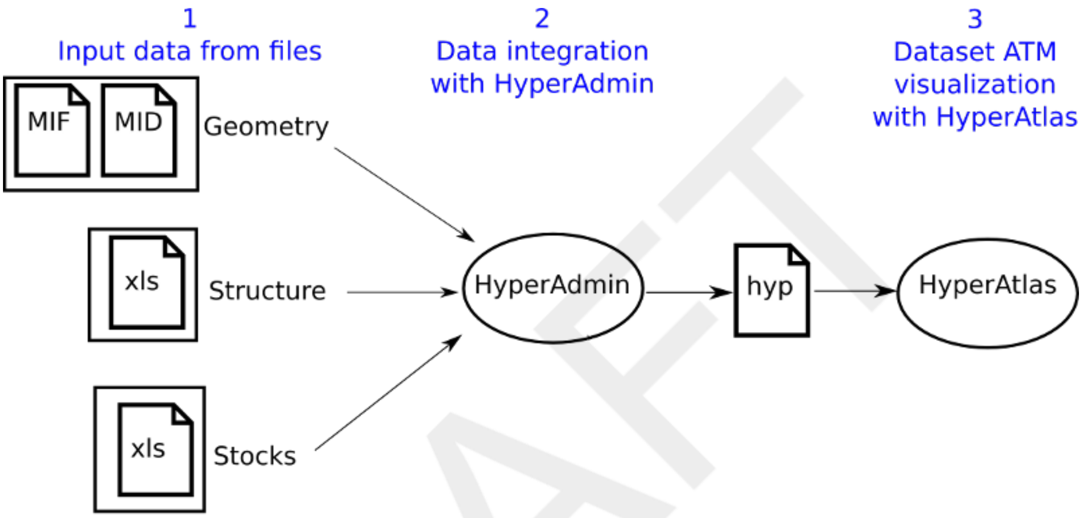
For more information, please visit HyperCarte Research Group Web site on http://hypercarte.imag.fr.

Chapter 2. HyperAdmin Overview

In order to perform Multiscalar Territorial Analysis with HyperAtlas, the datasets provided by geographers are serialized in a convenient format into a binary file named with the `.hyp` extension. As a convention, a HyperAtlas dataset input file is called an **hyp file** (example: `demography.hyp`).

HyperAdmin is the tool to generate hyp files from your a set of input well-formed files. The steps to generate an hyp file and the workflow between HyperAdmin and HyperAtlas is summarized in the Figure 2.1.

Figure 2.1. HyperCarte Workflow



HyperAdmin and HyperAtlas data flow.

To sum up, the main expected input files are:

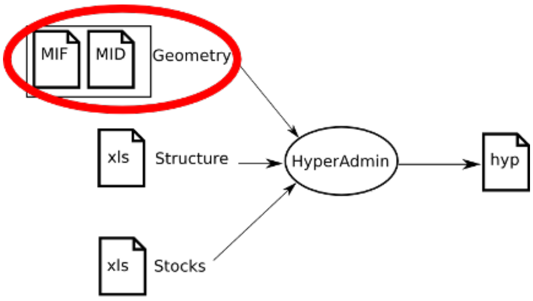
- the **geometry** of the dataset, in MapInfo MIF/MID formats:
  - the `MIF` file
  - the `MID` file
- the **structure** of the dataset, as an `xls` (Excel/OpenOffice) file
- the **stocks** of the dataset, as an `xls` (Excel/OpenOffice) file

As shown on Figure 2.1, creating a dataset `hyp` file consists in:

1. preparing your dataset geometry as a MIF/MID files pair (MapInfo format);
2. preparing your dataset structure as a speadsheet `structure.xls` file;
3. optionally, preparing a distance-time matrix as an `xls`file for custom contiguities;
4. preparing your dataset stocks as a spreadsheet (Excel/OpenOffice) `data.xls` file;
5. generating the dataset `hyp` file with HyperAdmin.

Following chapters describe each above step for integrating your data into an hyp file.

Chapter 3. Geometry input



This section describes the expected geometry input for HyperAdmin.

> **!** The maps are computed using the geometric information from the lowest level of territorial units, then aggregating this information to build the upper levels. So, the user must provide data without any hole, and territorial units at lowest level must be contiguous.

Expected geographical information must be provided by the user in the **MIF/MID** format (MapInfo format). For more information on this software and its format, please consult http://www.pbmapinfo.eu/ (last visit: 13rd may 2010).

3.1. The MID file

The `MID` file must be made of only one column where territorial units identifiers are listed, one per line, without any doublon. Example:

```
"AT111"
"AT112"
"AT125"
"AT126"
"AT127"
"AT13"
"AT211"
```

> **!** The given order of TU identifiers in the `MID` file must match the order of provided regions in the `MIF` file, see Data section of the MIF file [5]

Based on a naming convention of the identifiers for these territorial units, following exceptions are handled by HyperAtlas for particular display options. Please take into account the following exceptions when designing your dataset:

- **FR**, **ES**, **PT**, **MT** is the list of units identifiers for countries that own overseas units: France (Martinique, ...), Spain (Canarias, ...) and Portugal (Madeire). For example for European datasets, In HyperAtlas, the islands will be drawn in squares over the Russia.
- **SUR** and **BRA** (Surinam and Brazil) are examples of units identifiers that are treated differently when drawing them on the maps by HyperAtlas.
- Integer identifiers from **0** to **9** correspond to squares that must be drawn on the map, they are used for overseas in Europe dataset.
- A territorial unit with the identifier **no data** will be painted in white on the maps that are drawn by HyperAtlas. This exception is used for North Cyprus in Europe datasets.
- The **chypre** identifier is used to handle the particular case of the display of Cyprus island in the ESPON datasets.

3.2. The MIF file

> The information in this section is essentially based on the MapInfo Data Interchange Format document [1].

Geographical units are described in an ASCII file by their `X` and `Y` coordinates. The `.MIF` file is made of an header section then a data section.

Figure 3.1. MIF file header

```
VERSION n
Charset ”characterSetName”
[ DELIMITER ”<c>” ]
[ UNIQUE n,n.. ]
[ INDEX   n,n.. ]
[ COORDSYS...]
[ TRANSFORM...]
COLUMNS n
  <name> <type>
  <name> <type>

  etc.
```

As shown on Figure 3.1, the header can contain the following information:

- **VERSION**: the version of the MapInfo software;
- **CHARSET** clause specifies which character set was used to create text in the table (examples: `WindowsLatin1`, `MacRoman` or `Neutral`;
- **DELIMITER** shows the character that is used to separate columns values (if not specified, tabulation is the default delimiter);
- **UNIQUE** parameter must be a number that refers to a database column, this parameter is used to create related tables;
- **INDEX** parameter (a number or a comma-separated list of numbers) that shows the number(s) of the indexed column(s);
- the **COORDSYS** parameter sets the used coordinate system.

  This parameter is essential, in particular to compute the scale of the map. By default (when no `COORDSYS` clause is specified) data is assumed to be stored in longitude/latitude forms. All coordinates are stored with respect to the northeast quadrant. The coordinates for points in the West of Greenwich have a negative X while coordinates for points in the East of Greenwich have a positive X. Coordinates for points in the Northern hemisphere have a positive Y while coordinates for points in the Southern hemisphere have a negative Y. Examples:

  - The following example represents a map of Europe centered on 50°N 15°E with a Lamber Azimutal projection that can be associated to the following bounds pair: (X_{min}, Y_{min}) (X_{max}, Y_{max}). The `"m"` option stands for "meters" as the unit:

    ```
    CoordSys NonEarth Units "m" Bounds (-2217175, -1723801) (1783333, 2518193)
    ```

  - Another setting for a map of Rhône-Alpes may be:

    ```
    CoordSys NonEarth Units "m" Bounds (691594, 1893320) (993392, 2185448)
    ```

- **TRANSFORM** parameter can be used to convert coordinates which are given in a different quadrant than the default northeast one.

- **COLUMNS** parameter describes the data in the table of the associated `MID` file. The `n` parameter specifies the number of columns. Example:

```
Columns 1
   unit  Char(100)
```

specifies one column named `unit`, each value will be made of characters string type whose length is not longer than 100.

HyperAdmin is quite sensible on the format of the header of the MIF file (one information by line). Here are some examples of the expected formats for the header of the more frequently recent and used MIF files:

- European datasets (used for ESPON HyperAdmin maps with the EPSG 3035 Coordinates system and projection):

```
Version 300
Charset "Neutral"
Delimiter ","
CoordSys NonEarth Units "m" Bounds (2600301.93555, 1249109.375) (6593124.
Columns 1
  ID Char(50)
Data

(...)
```

- EUROMED dataset:

```
Version 300
Charset "WindowsLatin1"
Delimiter ","
CoordSys NonEarth Units "m" Bounds (-4487557.26071, -3722255.38453) (4487
Columns 1
  ID Char(10)
Data

(...)
```

- Metroborder dataset:

```
VERSION 300
Charset "WindowsLatin1"
DELIMITER ";"
COORDSYS NonEarth Units "m" Bounds (-743051.308162917,-145654.445989655)
COLUMNS 1
     SHN Char(14)
DATA

(...)
```

The **DATA** keyword specifies both the end of the header of the `MIF` file and the start of the enumeration of outlines.

If the MapInfo `MIF` file may set different types of graphical primitives (point, line, polyline, etc.), the HyperAdmin software only expects the polygon type in order to describe the outlines of territorial units. Eeach TU whose identifier is given in the `MID` file (see Section 3.1) must be associated to a new entry in the `MIF` file under the **data** section, IN THE SAME ORDER, as a **Region** entry. In MapInfo, a `Region` object consists of one or more polygons. Let us describe an expected **Region** entry using the definition example shown on Figure 3.2.

Figure 3.2. Example of two "Region" entries in the MIF file Data section

```
Data
Region  2 ❶
  7 ❷
108071.871 -293320.749
96339.456 -282096.297
102833.097 -261179.193
106485.534 -258631.56
123883.98 -262981.491
122621.886 -282959.13
108071.871 -293320.749
    Pen (1,2,0) ❸
    Brush (0,1)
    Center 110111.718 -275976.153
  5 ❹
-407753.01 -311500.065
-417000.993 -311417.496
-411718.965 -289228.641
-406514.985 -302217.573
-407753.01 -311500.065
    Pen (1,2,0)
    Brush (0,1)
    Center -411757.989 -300364.353
Region  1 ❺
  11 ❻
2186917.593 -1518464.703
2186829.009 -1692861.786
2129979.423 -1729141.275
1933829.46 -1729141.275
1928265.747 -1699690.677
1922979.324 -1671615.192
1928499.903 -1666190.274
1941660.768 -1656068.01
2005909.794 -1679948.187
2047505.1 -1676110.68
2186917.593 -1518464.703
    Pen (1,2,0)
    Brush (0,1)
    Center 2140313.457 -1623802.989
```

❶ Start of the entry for the first territorial unit in our data section. This region definition will be associated to the identifier on the firts entry of the `MID` file. The `2` parameter near `Region` shows that this region is made of two polyogons (example, France may be considered as a region made of two polygons: metropol and Corse island).

❷ The first polygon of this region is set with seven points whose coordinates in X Y forms are given on following lines.

❸ **Pen(a, b, c)**, **Brush(a, b)** and **Center x y** specifications are optional and they will not be read by HyperAdmin.

❹ The second polygon of this region is defined with five points whose coordinates are given on the five following lines.

❺ Here is the start of a new `Region` definition. As the second entry of the data section, this region definition will be associated to the identifier on the second line of the `MID` file. `Region 1` indicates that this region is made of one polygon.

❻ This line shows the number of points that compose the polygon: 11 points, whose coordinates are successively given on 11 folloging lines.

3.3. Layer of main cities

HyperAtlas can handle additional layers of information that can be displayed over the maps. Currently (May 2011), only a layer showing the main cities has been tested and can currently be supported.

The expected format for this "cities" layer incorporation into the dataset to be built is a **.csv** file. This file is only composed of three fields, these fields are separated by a comma character:

- the name of the city
- the X coodinate of this city, based on the MIF/MID projection and coordinates system
- the Y coodinate of this city, based on the MIF/MID projection and coordinates system

The following listing provides an example of the main cities layer definition csv file that has been used for European datasets (EPSG 3035):

```
Vilnius,5295673.924,3612560.328
Minsk,5460580.445,3560616.774
Dublin,3253284.971,3480193.09
Berlin,4547186.818,3272495.918
Amsterdam,3975886.565,3263689.867
Warszawa,5068508.328,3293815.926
London,3620060.313,3202333.12
Bruxelles/Brussel,3927032.583,3095975.903
Kyiv,5751996.553,3239855.146
Praha,4639737.703,3008973.669
Paris,3769691.587,2891825.057
Wien,4790135.661,2807741.98
Budapest,5003603.404,2753261.228
Bern,4128054.027,2651781.399
Beograd,5142183.84,2467117.484
Bucuresti,5593724.067,2506886.924
Sofiya,5408445.047,2274434.026
Tirana,5143864.946,2078891.927
Madrid,3164690.758,2032301.915
Ankara,6248076.399,2163898.451
Helsinki,5144699.201,4208069.911
Zagreb,4784474.809,2540154.601
Nicosia,6434072.209,1668719.112
Luxembourg,4054388.133,2965578.225
Bratislava,4859375.987,2822228.019
Tallinn,5154761.636,4105585.175
Sarajevo,4997878.051,2344715.534
Skopje,5274194.7,2172377.111
Athina,5518075.047,1777730.958
Kishinev,5733746.751,2835203.886
Copenhagen,4481880.455,3626362.309
Lisboa,2671218.026,1947183.08
Oslo,4362362.69,4091266.484
Reykjavik,2843090.801,4908517.82
Riga,5170116.607,3836021.74
Roma,4531433.066,2089563.772
Stockholm,4781578.636,4041161.089
```

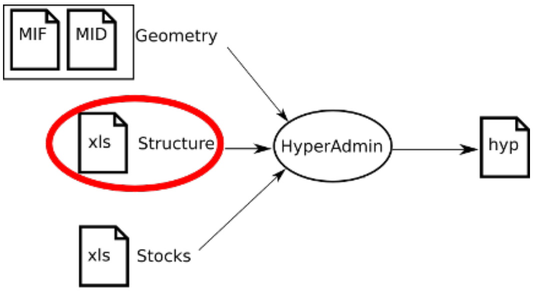
```
Valletta,4737055.11,1442089.281
Ljubljana,4670851.053,2559186.916
El-Jazair,3696198.974,1536632.051
Tounis,4344016.475,1511814.733
Podgorica,5085720.438,2197200.507
Vaduz,4287807.431,2668956.206
```

Chapter 4. Structure/Contiguity input

The Section 4.1 presents the expectations of the HyperAdmin about the structure input file, e.g. the information about the territorial units hierarchy and their relationships.

The Section 4.2 presents the optional steps that consists in creating a distance-time contiguity matrix input data for custom neighbourhood definitions (example: distance time, 2 hours by car, etc.).

4.1. Structure input



In the the input `structure.xls` Excel file, ten sheets must mandatory be provided in a unique `.xls` file.

Optionnally, complex contiguities must be defined as a set of seven sheets in an unique other xls file. Thus the input data may be composed of:

- `some_structure.xls`: to describe the structure;
- `some_contiguity.xls`: to optionally describe the contiguities (see Section 4.2).

Table 4.1 provides the list of these sheets names and a short description for each of them, as they are expected in the `structure` input definition. The expected columns and an example for each of them is described below this table.

The names of sheets is case-unsensitive. For example, the mandatory `UnitArea.txt` file can be named `unitarea.txt` or `UNITAREA.txt`. Suffixes and prefixes around the file basename are also possible while choosing the plain text option: thus, `my_UnitArea.txt` will be considered as a `UnitArea.txt` file, e.g. the input file that contains the information about the relationships between areas and territorial units.

Table 4.1. Overview of expected sheets for data structure input

Excel Sheet Name	Description
unit	Identifiers for the set of territorial units. See Content of unit.
Area	Identifiers for the set of study areas. See Content of area.
Zoning	Identifiers for the set of meshes. See Content of zoning.
UnitSup	Hierarchy between units: an `UTSup_ID` parent unit owns at least one child `UT_ID` unit. See Content of unitsup.
UnitArea	An UT belongs to one or several study areas. See Content of unitarea.
UnitZoning	An UT belongs to one or several meshes. See Content of unitzoning.
language	Provides a human readable name for used languages codes. See Content of language.
UnitLanguage	Names of UT in different languages. A translation may be missing. See Content of unitlanguage.
AreaLanguage	Names of the study areas in different languages. A translation may be missing. See Content of arealanguage.
ZoningLanguage	Names of the meshes in different languages. A translation may be missing. See Content of zoninglanguage.

Expected content for each of these sheets is following:

Unit This file/sheet must contain one column whose header cell must be `UT_ID`. Example:

Table 4.2. Sample input Unit sheet

UT_ID
AT11
AT12
AT13
AT21
etc...

Area This file/sheet must contain one column whose header cell must be `Area_ID`. Example:

Table 4.3. Sample input Area sheet

Area_ID
UE15
UE25
PECO
Arc_Atlantique
Nouveaux_UE
UE27
UE29

Zoning

This file/sheet must contain one column whose header cell must be `Zoning_ID`. An additional column named `Rank` may order given zonings. Example:

Table 4.4. Sample input Zoning sheet

Zoning_ID	Rank
Nuts_0	1
Nuts_1	2
Nuts_2	3
Nuts_3	5
Nuts_2_3	4

UnitSup

This file/sheet must contain two columns whose header cells must be named `UTSup_ID` and `UT_ID`. Example:

Table 4.5. Sample input UnitSup sheet

UT_ID	UTSup_ID
AT1	AT
AT2	AT
AT3	AT
BE1	BE
BE2	BE

UnitArea

This file/sheet must contain two columns whose header cells must be named `UT_ID` and `Area_ID`. Example:

Table 4.6. Sample input UnitArea sheet

UT_ID	Area_ID
AT	UE15
BE	UE15
DE	UE15
DK	UE15
ES	UE15

UnitZoning

This file/sheet must contain two columns whose header cells must be `UT_ID` and `Zoning_ID`. Example:

Table 4.7. Sample input UnitZoning sheet

UT_ID	**Zoning_ID**
AT	Nuts_0
BE	Nuts_0
BG	Nuts_0
CH	Nuts_0

Language

This file/sheet must contain two columns whose header cell are `Language_ID` and `Language_NAME`. Example:

Table 4.8. Sample input Language sheet

Language_ID	**Language_NAME**
DE	**allemand**
CS	tchèque
DA	danois
ET	estonien
EN	anglais
ES	espagnol

UnitLanguage

This file/sheet must contain three columns whose header cells must be `UT_ID`, `UT_NAME` and `Language_ID`. Example:

Table 4.9. Sample input UnitLanguage sheet

UT_ID	**Language_ID**	**UT_NAME**
AT11	DE	BURGENLAND
AT34	DE	VORARLBERG
BE24	NL	VLAAMS BRABANT
BE25	NL	WEST-VLAANDEREN
BE31	FR	BRABANT WALLON
BE32	FR	HAINAUT

AreaLanguage

This file/sheet must contain three columns whose header cells must be `Area_ID`, `Language_ID` and `Area_NAME`. Example:

Table 4.10. Sample input AreaLanguage sheet

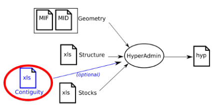
Area_ID	**Language_ID**	**Area_NAME**
UE15	FR	Union européenne des 15
UE25	FR	Union européenne des 25
PECO	FR	Pays d'Europe Centrale et Orientale

ZoningLanguage This file/sheet must contain three columns whose header cells must be `zoning_ID`, `Language_ID` and `zoning_NAME`. Example:

Table 4.11. Sample input ZoningLanguage sheet

Zoning_ID	Language_ID	Zoning_NAME
Nuts_0	FR	Nomenclature des unités territoriales de niveau 0
Nuts_1	FR	Nomenclature des unités territoriales de niveau 1
Nuts_2	FR	Nomenclature des unités territoriales de niveau 2
Nuts_3	FR	Nomenclature des unités territoriales de niveau 3
Nuts_2_3	FR	Nomenclature des unités territoriales de niveau 2-3

4.2. Contiguity input (optional)



This section presents the optional contigity definition input data file.

Table 4.12. Overview of expected sheets for contiguity input

Excel Sheet Name	Description
Contiguity	List of identifiers for contiguities.
ContiguityLanguage	Names of the contiguities in different languages. A translation may be missing. See Content of contiguitylanguage.
Neighbourhood	Unique code for a neighbourhood that is associated to a contiguity, a threshold and a comparator. The comparator shows if two UT are neighbours or not. See Content of neighbourhood.
Neighbourhood	Names of neighbourhoods for each language. A translation may be missing. See Content of neighbourhoodlanguage.
ContiguityZoning	A distance matrix is available for one or several meshes. See Content of ContiguityZoning.
ContiguityArea	A distance matrix is available for one or several study areas. See Content of ContiguityArea.
UnitContiguity_i	Each line provides the code of two UT, following columns show the distance for `Contiguity_ID`$_i$, the header cell of each contiguity column providing the identifier of this contiguity. See Content of UnitContiguity.

Following listing provides an example for each expected sheet describing a contiguity definition:

Contiguity.txt / Contiguity

This sheet must contain one column whose header cell is `ID`. Example: the following sample sets two possible computations for contiguity, distance-time matrixes will be available for a car and for a lorry.

Table 4.13. Sample input Contiguity sheet

ID
CAR
TRUCK

ContiguityLanguage.txt / ContiguityLanguage

This sheet must contain four columns whose header cells are `CONTIGUITY_ID`, `Language_ID` (see Note about expected languages identifiers), `Contiguity_NAME` (e.g. the name of this contiguity in this locale) and `Contiguity_DESC` (a description of this contiguity). Example:

Table 4.14. Sample input ContiguityLanguage sheet

Contiguity_ID	Language_ID	Contiguity_NAME	Contiguity_DESC
CAR	EN	car time	Time between units by car (in minutes)
TRUCK	EN	truck time	Time between units on a truck (minutes)
CAR	FR	temps voiture	Temps entre les unités en voiture (minutes)
TRUCK	FR	temps camion	Temps entre les unités en camion (minutes)

Neighbourhood.txt / Neighbourhood

This sheet must contain four columns whose header cells are `Neighbourhood_ID` (see Note about expected languages identifiers), `Contiguity_ID`, `Distance` and `Comparator`. Possible values for the `Comparator` cells are:

- `<`
- `<=`
- `==`
- `>=`
- `>`

Example:

Table 4.15. Sample input Neighbourhood sheet

Neighbourhood_ID	**Contiguity_ID**	**distance**	**comparator**
CAR <= 360	CAR	360	<=
TRUCK <= 360	TRUCK	360	<=
CAR <= 540	CAR	540	<=
TRUCK <= 540	TRUCK	540	<=
CAR <= 180	CAR	180	<=
TRUCK <= 180	TRUCK	180	<=

NeighbourhoodLanguage.txt / NeighbourhoodLanguage

This sheet must contain four columns whose header cells are `Neighbourhood_ID`, `Language_ID` (ISO-639 language in 2 digits, `Neighbourhood_NAME` (e.g the name of this neighbourhood) and `Neighbourhood_DESC` (e.g. a description of this neighbourhood). In the following example, neighbourhoods are translated in english (`EN`) and french (`FR`):

Table 4.16. Sample input NeighbourhoodLanguage sheet

Neighbourhood_ID	Language_ID	Neighbourhood_NAME	Neighbourhood_DESC
CAR <= 360	EN	6h car	Units at less than 6 hours by car
TRUCK <= 360	EN	6h truck	Units at less than 6 hours on a truck
CAR <= 540	EN	9h car	Units at less than 9 hours by car
TRUCK <= 540	EN	9h truck	Units at less than 9 hours on a truck
CAR <= 180	EN	3h car	Units at less than 3 hours by car
TRUCK <= 180	EN	3h truck	Units at less than 3 hours on a truck
CAR <= 360	FR	6h de voiture	Unités à moins de 6 heures en voiture
TRUCK <= 360	FR	6h de camion	Unités à moins de 6 heures en camion
CAR <= 540	FR	9h voiture	Unités à moins de 9 heures en voiture
TRUCK <= 540	FR	9h de camion	Unités à moins de 9 heures en camion
CAR <= 180	FR	3h de voiture	Unités à moins de 3 heures en voiture
TRUCK <= 180	FR	3h de camion	Unités à moins de 3 heures en voiture

ContiguityZoning.txt / ContiguityZoning

This sheet must contain two columns whose header cells are `Contiguity_ID` and `Zoning_ID`. In the following example, the distance-time by car and distance-time by lorry are available for `NUTS_2` zoning only:

Table 4.17. Sample input ContiguityZoning sheet

Contiguity_ID	Zoning_ID
CAR	Nuts_2
TRUCK	Nuts_2

> The identifiers that are given in the `Zoning_ID` column must be coherent with the identifiers that have been given in the `Zoning` sheet of the structure input, see Content of zoning.

ContiguityArea.txt / ContiguityArea

This sheet must contain two columns whose header cells are `Contiguity_ID` and `Area_ID`. In the following example, distance-time matrixes by car are available for `UE15` and `UE25`, distance-time matrixes by lorry are available for `UE27` and `UE29`:

Table 4.18. Sample input ContiguityArea sheet

Contiguity_ID	Area_ID
CAR	UE15
CAR	UE25
TRUCK	UE27
TRUCK	UE29

> The identifiers that are given in the `Area_ID` column must be coherent with the identifiers that have been given in the `Area` sheet of the structure input, see Content of area.

UnitContiguity.txt / UnitContiguity$_i$

This sheet must contain at least three columns whose header cells are `UT_ID1`, `UT_ID2`, then the identifier of a contiguity.. In the following example, contiguities between units are performed for `CAR` contiguity and `TRUCK` contiguity:

Table 4.19. Sample input UnitContiguity sheet

UT_ID1	UT_ID2	CAR	TRUCK
ES12	UKC1	1265.11	1820.5
ES13	UKC1	1138.85	1649.5
ES21	UKC1	1058.92	1529.9

> In Excel mode, each `UnitContiguity` can only contain 2^{16} rows, e.g. 65536. Several sheets can be created to import more results: just name your sheets `UnitContiguity1`, `UnitContiguity2`, etc. Note that only 30 `UnitContiguity`$_i$ sheets can be created. Nevertheless, on considering a symetric relationship for a distance between two units (e.g. distance between UT1 and UT2 equals the distance between UT2 and UT1), the number of needed rows can quasi be reduced by half. Thus, the number `S` of needed sheets for `n` units can be found by executing the formula which is shown on Figure 4.1:

Figure 4.1. Number S of needed sheets for n units

$$S = \frac{1 + \frac{n \cdot (n-1)}{2}}{65536}$$

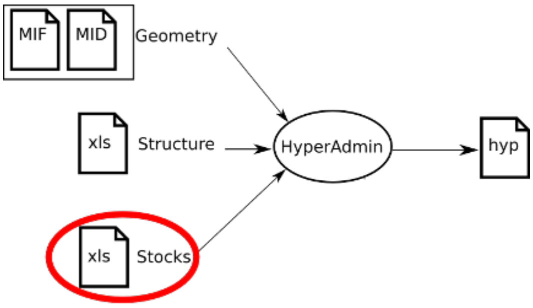
`S` is the number of needed sheets, `n` is the number of units. On the numerator, "`1 +`" stands for the header row that must be included on each sheet.

Note that **contiguities are not aggregable**: a distance matrix is set for a given level of mesh. For its upper level, the associated distance matrix must also be given. Table 4.20 provides an example of the number of needed sheets for different levels with several numbers of units.

Table 4.20. Example of needed sheets number

Mesh	Number of UT (n)	Number of sheets (S)
NUTS_0	29	1
NUTS_1	92	1
NUTS_2	280	2
NUTS_2_3	727	5
NUTS_3	1329	14
	Total	23

Chapter 5. Stocks input



The **stock** file mainly aims at describing the statistics of the dataset.

5.1. HyperAdmin input data file format

This section describes the stocks (statistical data) file that HyperAdmin expects as input.

Please note the following requirements for the input data file:

- the input data file must be a spreadsheet xls file (editable by Microsoft Excel and Open Office) named "`*data*.xls`": the filename must include the "`data`" characters sequence and the `.xls` extension is required.
- the values of stocks must be provided for the lowest level of territorial units. This list is available in the example data template that depends on the selected structure/geometry model at previous step.
- all values for all units must be filled;

Following sections describe the expected format (sheets, columns and possible values) for the version 2 of this `data.xls` "stocks" file.

5.1.1. About

Table 5.1 provides an example for this mandatory sheet in the data v2 input xls file.

Table 5.1. V2 sample About sheet

VERSION	**TIME_ENABLED**
2	TRUE

This sheet aims at identifying the version of the format of this data file. Currently (2010-2011), only the value **2** is possible for the `VERSION` column.

The expected value for the `TIME_ENABLED` column is a boolean: only **TRUE** or **FALSE** values are possible:

- The **TRUE** value shows that values are available for the sames labels of indicators at several dates: for example, the population in 2000, the population in 2002.
- The **FALSE** value shows that each indicator is given for a single date.

5.1.2. Data

Table 5.2 provides an example for this mandatory sheet in the data v2 input xls file.

Table 5.2. V2 sample Data sheet

UT_ID	pop2000	pop2002	area2000	gdp2000	gdp2002
AT111	1	15	2	7	10
AT112	3	16	4	8	11
AT113	5	17	6	9	12

This sheet must provide at least three columns: `UT_ID` then at least two indicators identifiers (in HyperAtlas, there must be at least one numerator stock and one denominator stock). The Table 5.2 shows five indicators identifiers: `pop2000`, `pop2002`, `area2000`, `gdp2000` and `gdp2002`. These identifiers must be described in the **StockInfo** sheet (see Section 5.1.8).

The `UT_ID` column must provide the list of territorial units at the lowest rank (example, at NUTS 3 level) of the dataset. The units are referenced by their identifiers that must match the given values in the associated `structure.xls` input file.

Then, each other cell provides a value for the given indicator column at the given unit row. For example in Table 5.2, 17 is the value for `pop2002` indicator in `AT113` territorial unit.

Each cell must be valuated. Missing values are not accepted here.

5.1.3. Default

Table 5.3 provides an example for this optional sheet in the data v2 input xls file.

Table 5.3. V2 sample Default sheet

DEFAULT_NUM	DEFAULT_DEN
pop	area

This sheet aims at providing a default indicator to be selected in HyperAtlas at startup for the denominator and for the numerator combo boxes. Expected values for both columns are valid indicators identifiers that must match two of those defined in the **StockInfo** sheet (see Section 5.1.8).

5.1.4. Label

Table 5.4 provides an example for this mandatory sheet in the data v2 input xls file.

Table 5.4. V2 sample Label sheet

LABEL_ID	LANG_CODE	NAME	DESC
1	EN	Total population	Total population in thousands
1	FR	Population totale	Population totale en milliers
2	EN	Area	Total area
2	FR	Superficie	Superficie totale
3	EN	GDP	Gross domestic product
3	FR	PIB	Produit intérieur brut
4	EN	GDP/Inhabitant	Gross domestic product per inhabitant
4	FR	PIB/Hab	PIB par habitant
5	EN	Density	Density of population
5	FR	Densité	Densité de population

This sheet aims at providing the internationalized names and descriptions for the indicators and predefined ratios. The `LABEL_ID` and `LANG_CODE` provides indexes for this table: for a given label identifier there may be several available translations. Thus, the `LABEL_ID = 1` is available in english (`LANG_CODE = EN`) and french (`LANG_CODE = FR`) languages. In the **StockInfo** sheet, each indicator reference a label identifier. As several indicators may be similarly named and described (when an indicator is valuated for several dates), these labels have been exported here.

The language identifier code must be a valid ISO Language Code. These codes are the lower-case, two-letter codes as defined by ISO-639. Nevertheless, the parser supports upper-cases. You can find a full list of these codes at a number of sites, such as: http://www.ics.uci.edu/pub/ietf/http/related/iso639.txt (2011-03-16).

Note that values in the `LABEL_ID` column may be referenced from the **StockInfo** sheet (see Section 5.1.8) and from the **RatioStock** sheet (see Section 5.1.7).

5.1.5. Metadata

Table 5.5 provides an example for this optional sheet in the data v2 input xls file.

Table 5.5. V2 sample Metadata sheet

UT_ID	STOCK_ID	PROVIDER_ID
AT111	pop2000	1
AT112	pop2000	2
	area	2
	pop2002	1

This draft sheet aims at providing some basic metadata information for an indicator relatively or not to a territorial unit. Currently, only the source of data may be given as metadata.

For example in Table 5.5, the values of the `pop2000` indicator identifier were retrieved from different sources for regions `AT111` and `AT112`. On the contrary, all values for the `area` indicator, whatever the unit is, were provided by the same source. Idem for the `pop2002` indicator.

The values in the `PROVIDER_ID` column must match the identifiers that are given in the **Provider** sheet (see Table 5.6). Likewise, the values in the `STOCK_ID` column must match the identifiers that are defined in the **StockInfo** sheet (see Table 5.8).

5.1.6. Provider

Table 5.6 provides an example for this optional sheet in the data v2 input xls file.

Table 5.6. V2 sample Provider sheet

PROVIDER_ID	NAME	CONTACT	URL
1	Eurostat	toto@eurostat.eu	http://www.eurostat.eu
2	INSEE	tata@insee.fr	http://www.insee.fr

This sheet aims at providing the list of data providers. Their different ids are referenced from the **Metadata** sheet.

5.1.7. RatioStock

Table 5.7 provides an example for this optional sheet in the data v2 input xls file.

Table 5.7. V2 sample RatioStock sheet

RATIO_ID	LABEL_ID	NUM_ID	DEN_ID	VALIDITY_START	VALIDITY_END
1	4	gdp2000	pop2000	2000	2000
2	4	gdp2002	pop2002	2002	2002
3	5	pop2000	area2000	2000	2000
4	5	pop2002	area2000	2002	2002

This sheet aims at defining relevant ratios for the HyperAtlas "ratio" combo box parameter. Table 5.7 shows the example of two such predefined ratios, each of them for two different dates:

- the GDP/Inhabitant:
  - in 2000 (second line)
  - in 2002 (third line)
- The density of population:
  - in 2000 (fourth line)
  - in 2002 (fifth line)

Each value in the `RATIO_ID` column must be unique. Doublons will overwrite the previous found value.

Note that the `LABEL_ID` references the sames labels for the given pairs of numerator/denominator at different dates (`4` for lines 2 and 3, `5` for lines 4 and 5). These labels identifiers must be set in the**Label** sheet (see Section 5.1.4).

The values in the `NUM_ID` column and the values in the `DEN_ID` column must match the identifiers of indicators that are defined in the **StockInfo** sheet (see Section 5.1.8).

The values in the `VALIDITY_START` column will only be considered if the value of the `TIME_ENABLED` column in the **About** sheet is `TRUE` (see Section 5.1.1). Then, one relevant ratio can be chosen in HyperAtlas for different dates. Identically for the values in the `VALIDITY_END` column. Though `VALIDITY_START` and `VALIDITY_END` columns are designed to handle time intervals, setting the same value in both columns makes the ratio associated to a timestamp.

> **!** The expected format for both valididy start/end date fields is currently a year in the `yyyy` pattern. An input like `2010-12-31` is possible but this version of the application will only take into account the year, that is to say `2010` for this example.

5.1.8. StockInfo

Table 5.8 provides an example for this mandatory sheet in the data v2 input xls file.

Table 5.8. V2 sample StockInfo sheet

STOCK_ID	LABEL_ID	MEASURE_UNIT	VALIDITY_START	VALIDITY_END	VISIBLE_FLAG
pop2000	1	*1000	2000	2000	TRUE
pop2002	1	*1000	2002	2002	TRUE
area2000	2	km2	2000	2000	TRUE
gdp2000	3	euros	2000	2000	TRUE
gdp2002	3	euros	2002	2002	TRUE

This sheet mainly aims at providing the identifiers of the indicators of the dataset. Here are a short description for each column of this sheet:

- STOCK_ID: each value in this column must be unique. Any doublon will overwrite the previous found identical value. This column lists the identifiers of the indicators that are referenced in the other sheets. Note that several indicators may be associated to the same label (lines 2 and 3 for example), though they exist to distinguish the values of the population in 2000 and 2002.
- LABEL_ID: each value in this column must reference an identifier defined in the **Label** sheet (see Section 5.1.4).
- MEASURE_UNIT: simply provides the unit of measure for this indicator.
- VALIDITY_START: shows the start date of validity for this indicator. This field will only be considerated if the value of the TIME_ENABLED column in the **About** sheet is TRUE (see Section 5.1.1 and Important note about expected date format).
- VALIDITY_END: shows the end date of validity for this indicator. VALIDITY_START and VALIDITY_END fields are able to manage time intervals, but they can be used to associate a timestamp to the current stock: just write the same value in both cells (please see Important note about expected date format).
- VISIBLE: this field acts like a flag, a boolean is expected for the values of this column. A TRUE value shows that this indicator will be available in the numerator and in the denominator combo boxes of HyperAtlas parameters panel. A FALSE value may be usefull to define relevant ratios whose indicators have no reason to be available in the numerator and denominator combo boxes. For example, the life expectancy pre-defined ratio considers indicators that have no sense out of this compute.

Chapter 6. How to use HyperAdminimal

6.1. Launching HyperAdminimal

Requirements

To execute HyperAdminimal, a Java Runtime Environment (version 1.6 or upper) is required on your environment.

To check this requirement, open a console (dos/Terminal/shell) and type the **java -version** command. If your environement is Java-enabled, a message as shwon below must be displayed:

```
$ java -version
java version "1.6.0_17"
Java(TM) SE Runtime Environment (build 1.6.0_17-b04)
Java HotSpot(TM) Server VM (build 14.3-b01, mixed mode)
```

For more information on how to install a Java Runtime Environment, please consult http://java.com.

This HyperAdminimal r668 distribution also requires a graphical environment. The command-line-only mode is not enabled on this version.

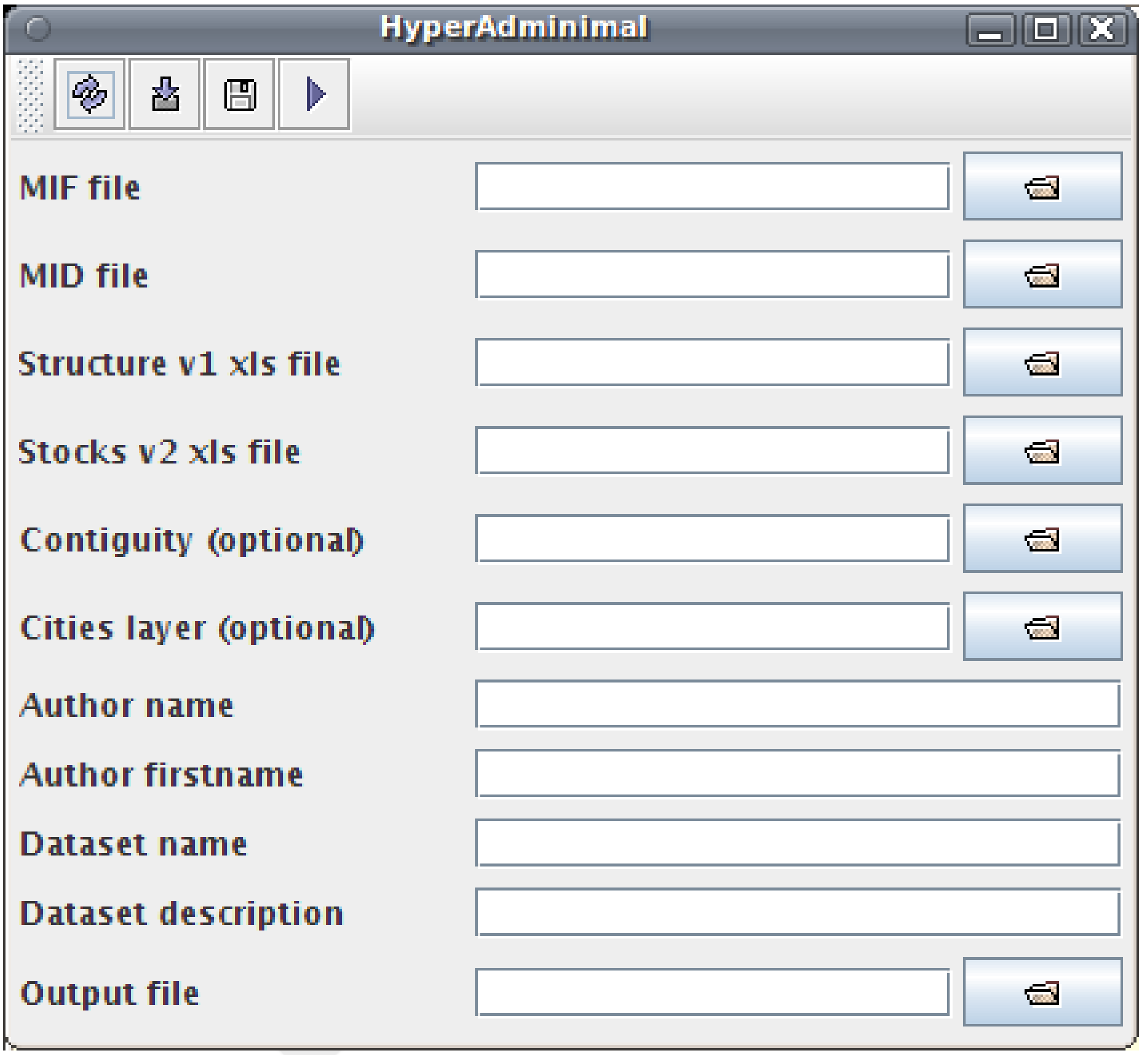
The HyperAdminimal application is delivered as a Java executable binary file named `hyperadminimal.jar`.

To execute HyperAdminimal software, open a console (depending on your platform: Windows-dos / Mac-Terminal / Linux-shell) and type the following command from the directory where `hyperadminimal.jar` is located on your disk:

```
java -jar hyperadminimal.jar
```

This command must display the main frame of the application as shown in Figure 6.1.

Figure 6.1. HyperAdminimal main frame at startup



HyperAdminimal at startup.

6.2. Using HyperAdminimal

As shown in Figure 6.1, the application is composed of a basic toolbar and of a form composed of 11 input text fields.

The text fields aim at specifying the input parameters for the generation of the dataset `.hyp` file. As most of expected inputs are files, the buttons on the right side allow to browse the user's file system to choose them. Nevertheless, the user may directly type the absolute paths to the input files in their respective text fields.

Note that most of parameters are required, only `Contiguity` and `Cities` fields are optional.

At this step, the user is expected to have prepared on his/her disk the four requested input files (see Expected Input Files [2]).

In addition to the requested input files, note that the user must mandatory provide some metadata about the dataset to be generated:

- the name of the author of this dataset

- the firstname of this author
- the name for the dataset
- a short sentence as an abstract of this dataset (`Dataset description` field value)

Finally, the `Output file` field consists in entering the absolute path and filename of the dataset `.hyp` file to be generated. For example, `/home/user/myDataset.hyp` (spaces and accentuated characters should be avoided for this value).

Then, to start the build of the dataset `hyp` file, simply click the "Play" button on the toolbar.

Build logs and eventual error messages or exceptions are displayed to the console (see the command to execute HyperAdminimal [24]). In the case of a failure, the displayed messages to the console may be usefull to fix the input files.

At the end of the process, a graphical box displays a message that indicates if the build failed or succeeded.

6.2.1. Saving and loading the build parameters

In order to replay several times a dataset build, the HyperAdminimal application proposes a functionnality to save and load the build parameters. This functionnality can be explained by describing the available buttons on the toolbar:

- **Reset parameters**

  This button erases the values in the fields.

- **Load parameters**

  Clicking this button opens a file chooser, the user is invited to select on his/her disk an XML file that has previously been saved via the save parameters [26] functionnality.

- **Save parameters**

  On clicking this button, the user is invited to select a directory and a filename on his/her disk. The content of the fields will be saved to this file in XML format. As shown in Figure 6.2, the format of this XML file is quite simple, it can thus be eventually easily copied/pasted and manually edited for various configurations.

  Figure 6.2. HyperAdminimal parameters XML file format sample

```
<?xml version="1.0" encoding="UTF-8"?>
<hyperadmin>
  <mif>/home/blerubrus/geom_nuts2_eu31.mif</mif>
  <mid>/home/blerubrus/geom_nuts2_eu31.mid</mid>
  <structure>/home/blerubrus/structure.xls</structure>
  <data>/home/blerubrus/data.xls</data>
  <contiguity />
  <authorname>HyperCarte Research Group</authorname>
  <authorfirstname>LIG STEAMER</authorfirstname>
  <datasetname>testDataset</datasetname>
  <datasetdescription>Built with HyperAdmininimal</datasetdescription>
  <output>/home/blerubrus/test.hyp</output>
</hyperadmin>
```

Appendix A. References

Internet

[1] Pitney Bowes MapInfo. *Appendix J: MapInfo Data Interchange Format*. [on line]. http://resource.mapinfo.com/static/files/document/1074660800077/interchange_file.pdf (last visit: 18.th may 2010).

Appendix B. About

This document is part of the HyperCarte Research Group projects. It has been generated at the following date, 2013-07-05 15:27:05, from the svn rev 1088 sources of the `docbench` project.

This document has been written by the LIG STeamer team.

For any comment question or suggestion, please please visit http://hypercarte.imag.fr or contact Jérôme Gensel <Jerome.Gensel@imag.fr>.

Laboratoire LIG UMR 5217, Equipe STeamer
681 rue de la Passerelle, Domaine Universitaire
BP 72
38402 Saint Martin d'Hères Cedex
FRANCE
Tel: (+33) (0)4 76 82 72 80
Fax: (+33) (0)4 76 82 72 87

Colophon

Based on DocBook technology [1], this document is written in XML format, sources are validated with DocBook DTD 4.5CR3, then sources are transformed to HTML and PDF formats by using DocBook xslt 1.73.2 stylesheets. The generation of the documents is automatized thanks to the `docbench` LIG STeamer project that is based on Ant [2], java [3], processors Xalan[4] and FOP [5]. Note that Xslt standard stylesheets are customized in order to get a better image resolution in PDF generated output for admonitions icons: the generated sizes of these icons were turned from 30 to 12 pt.

[1][on line] *DocBook.org* [http://www.docbook.org] (last visit: July 2011)
[2][on line] *Apache Ant - Welcome*. Version 1.7.1 [http://ant.apache.org] (last visit: July 2011)
[3][on line] *Developer Resources For Java Technology* [http://java.sun.com] (last visit: July 2011). Version 1.6.0_03-b05.
[4][on line] *Xalan-Java Version 2.7.1* [http://xml.apache.org/xalan-j/] (last visit: 18 november 2009). Version 2.7.1.
[5][on line] *Apache FOP* [http://xmlgraphics.apache.org/fop/download.html] (last visit: July 2011). Version 0.94.