A la Une du Support

IMPORTER ET AFFICHER LES PRIX À LA POMPE DES CARBURANTS DANS SAS VISUAL ANALYTICS À L'AIDE DE SAS STUDIO

Le gouvernement fournit en Open Data au format XML les prix des carburants par le biais du site http://www.data.gouv.fr.

Cet article explique pas-à-pas, avec du code SAS, comment importer, convertir puis charger en mémoire LASR ces fichiers XML. Ces opérations faites automatiquement grâce au code SAS, il ne reste plus qu'à les visualiser ensuite dans SAS Visual Analytics.

Caractéristiques :

Catégories : Visual Analytics, SAS Studio

OS : Windows, Unix

Version : SAS® 9.4 M3

Vérifié en décembre 2015

Sommaire

Importer et afficher les prix à la pompe des carburants dans SAS Visual Analytics. 1
1. Les données Open Data 2
2. Import des données dans une table SAS 3
2.1. Téléchargement des données 3
2.2. Téléchargement automatisé 3
2.3. Examen du contenu et correction des caractères accentués 4
2.3.1. Conversion des caractères HTML en caractères « texte » 4
2.3.2. Autres méthodes de conversion : 6
2.3.3. Contrôle de la conversion 6
2.4. Import du XML dans SAS 7
2.5. Création d'une map XML avec SAS XML Mapper. 7
2.6. Création des tables SAS. 9
2.7. Chargement des tables dans Hadoop 12
2.8. Chargement des données dans le serveur LASR Public 13
2.9. Association des tables via la PROC IMSTAT (création d'un schéma en étoile) 14
2.10. Sauvegarde de la table finale ESSENCE dans Hadoop. 16
3. Exploration dans SAS Visual Analytics 18
3.1. Validation des données : création d'une exploration 18
4. Création d'un rapport SAS Visual Analytics 21
5. Ressources 27
5.1. Code complet 27
5.2. Erreurs rencontrées : 27
5.2.1. La table n'apparaît pas dans SAS Visual Analytics 27
5.2.2. ERROR : Metadata Server denied access to operation 27
6. Liens utiles 29
7. Conclusion 29

1. LES DONNÉES OPEN DATA

Le site https://www.data.gouv.fr/fr/datasets/prix-des-carburants-en-france/ fournit sous Licence Ouverte le détail des prix des carburants. La licence ouverte se définit selon les caractéristiques suivantes :

«
- Une licence ouverte, libre et gratuite, qui apporte la sécurité juridique nécessaire aux producteurs et aux « ré-utilisateurs » des données publiques ;

- Une licence qui promeut la réutilisation la plus large en autorisant la reproduction, la redistribution, l'adaptation et l'exploitation commerciale des données ;

- Une licence qui s'inscrit dans un contexte international en étant compatible avec les standards des licences Open Data développées à l'étranger et notamment celles du gouvernement britannique (Open Government Licence) ainsi que les autres standards internationaux (ODC-BY, CC-BY 2.0). »

Source : http://wiki.data.gouv.fr/wiki/Licence_Ouverte_/_Open_Licence

La liste des jeux de données est disponible à cette adresse : https://www.data.gouv.fr/fr/datasets/

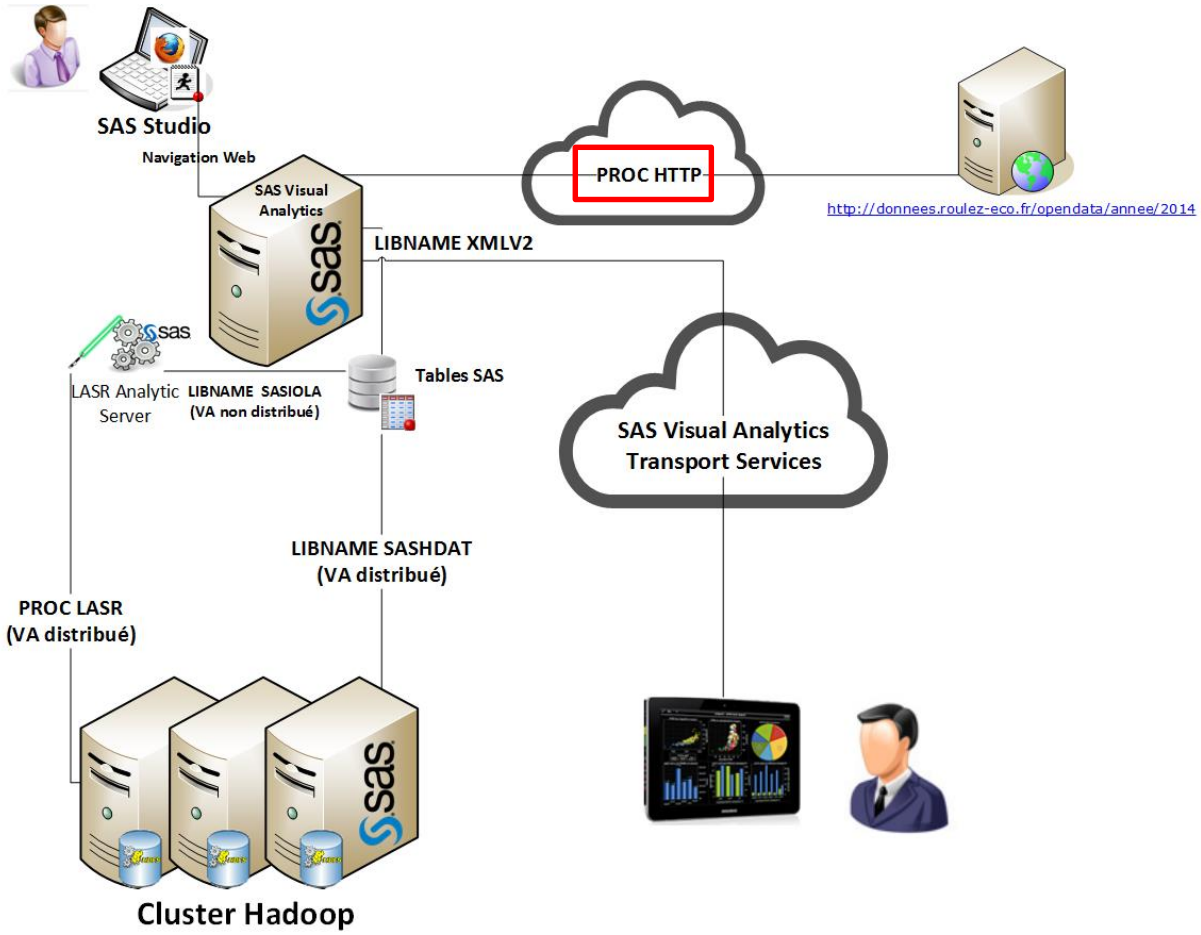
Autres exemples de données open data disponibles : vous pouvez trouver la liste officielle des codes postaux disponible à cette adresse : https://www.data.gouv.fr/fr/datasets/base-officielle-des-codes-postaux/. Cette liste est utile dans le sens où le code postal d'une commune n'est pas forcément le même que celui de son code INSEE. À titre d'exemple les communes de Cerdon et Isdes dans le Loiret ont le même code postal, mais chacune a un code INSEE unique.

2. IMPORT DES DONNÉES DANS UNE TABLE SAS

2.1. Téléchargement des données

Les données sont téléchargeables via l'adresse https://www.data.gouv.fr/fr/datasets/prix-des-carburants-en-france/. Nous allons utiliser les données complètes de l'année 2014 disponibles à l'adresse http://donnees.roulez-eco.fr/opendata/annee/2014 . Ces données ont la forme d'un fichier compressé ZIP dans lequel se trouve un fichier XML d'environ 153 Mo : PrixCarburants_annuel_2014.xml.

2.2. Téléchargement automatisé



SAS peut télécharger des données grâce à la PROC HTTP. Voici le code utilisé pour télécharger l'intégralité des données Open Data. La boucle télécharge les données jusqu'à l'année précédente. La dernière PROC HTTP télécharge les données de l'année courante.

```
OPTIONS SPOOL;
%let delim=%sysfunc(ifc(%eval(&sysscp. = WIN),\,/));
%let folder=/opt/sas/datas/lasradm;
%let proxyhost="";
%let proxyport=0;
%let stopdate = %sysfunc(year(%sysfunc(today())));
libname essence "&folder";
```

```
%macro download();
 %do i=2007 %to &stopdate-1;
      filename download "&folder.&delim.PrixCarburants_annuel_&i..zip";
      /* Telechargement du fichier avec la proc HTTP*/
      proc http  method='GET'  url="http://donnees.roulez-
 eco.fr/opendata/annee/&i"
       proxyhost=&proxyhost proxyport=&proxyport
       out=download;
      run;
 %end;

 filename download "&folder.&delim.PrixCarburants_annuel_&stopdate..zip";
 /* Telechargement du fichier avec la proc HTTP*/
 proc http  method='GET'  url="http://donnees.roulez-eco.fr/opendata/annee"
 proxyhost=&proxyhost proxyport=&proxyport  out=download;
 run;
%mend download;
%download();
```

2.3. Examen du contenu et correction des caractères accentués.

2.3.1. Conversion des caractères HTML en caractères « texte »

L'ouverture de ce fichier PrixCarburants_annuel_2014.xml dans un utilitaire Notepad évolué (le fichier fait 153M ce qui n'est pas supporté par le notepad natif de Windows), Notepad++ par exemple, donne plusieurs indications :

```
PrixCarburants_annuel_2014.xml
1  <?xml version="1.0" encoding="ISO-8859-1" standalone="yes"?><pdv_liste>
2  <pdv id="1000001" latitude="4620114" longitude="519791" cp="01000" pop="R">
3    <adresse>ROUTE NATIONALE</adresse>
4    <ville>SAINT-DENIS-L&#xE8;S-BOURG</ville>
5    <ouverture debut="01:00" fin="01:00" saufjour=""/>
6    <services>
7      <service>Automate CB</service>
8      <service>Vente de gaz domestique</service>
9      <service>Station de gonflage</service>
10   </services>
11   <prix nom="Gazole" id="1" maj="2014-01-02 11:08:03" valeur="1304"/>
12   <prix nom="Gazole" id="1" maj="2014-01-04 09:54:03" valeur="1304"/>
13   <prix nom="Gazole" id="1" maj="2014-01-05 10:27:09" valeur="1304"/>
```

- Le fichier XML est déclaré contenir des informations au format ISO-8859-1
- Un nom de ville est étrangement affiché.

Le code è est une balise HTML hexadécimale permettant d'encoder les caractères accentués mais cette balise HTML n'est pas lisible directement par SAS.
Par conséquent, il faut convertir ces balises en caractères accentués. Le tableau des correspondances entre ces codes hexa et les caractères est disponible à cette adresse : http://webdesign.about.com/od/localization/l/blhtmlcodes-fr.htm ou encore http://character-code.com/french-html-codes.php.

2.3.1.1. Conversion avec SAS

La méthode préconisée est d'utiliser un code SAS faisant appel à la PROC HTMLDECODE et FILENAME ZIP. Il est nécessaire que SAS ait le support de UTF-8 c'est-à-dire que le dossier <SASHome>\SASFoundation\9.4\nls\u8 existe ; sans pour autant lancer SAS en mode utf8. Sous Unix/Linux le chemin est le même : <SASHome>/SASFoundation/9.4/nls/u8.

Dans l'exemple ci-dessous le « filename zip » permet de lire le fichier xml directement dans le zip. Les appels TRANWRD ont été nécessaires pour convertir les caractères non pris en compte par SAS htmldecode.

Ce code nécessite d'avoir exécuté la création de xmlmap au préalable.

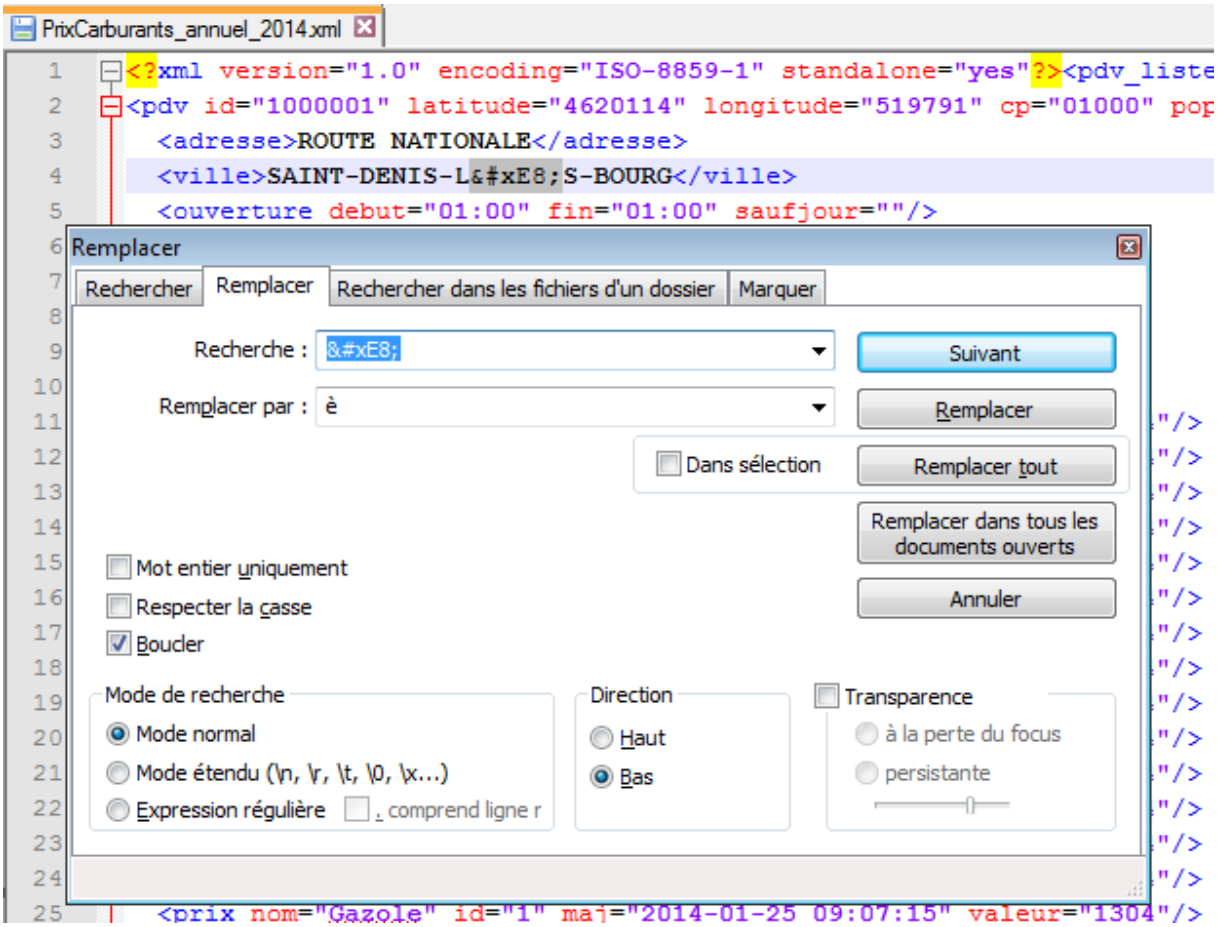
```
OPTIONS SPOOL;
%let delim=%sysfunc(ifc(%eval(&sysscp. = WIN),\,/));
%let folder=/opt/sas/datas/lasradm;
%let proxyhost="";
%let proxyport=0;
%let stopdate = %sysfunc(year(%sysfunc(today())));
libname essence "&folder";

%macro createtable();
proc datasets library=essence;
   delete pdv prix;
run;
/*%let stop = %sysfunc(year(%sysfunc(today())));*/
%do i=2007 %to &stopdate;
/*--- Declaration du fichier zip avec le filename "zip"---*/
filename inzip zip "&folder.&delim.PrixCarburants_annuel_&i..zip";
/*--- data _null_ car il n'y a pas de création de table SAS a ce stade ---
*/
data _null_;
/*--- le infile "inzip" va lire le fichier XML directement dans le zip*/
infile inzip("PrixCarburants_annuel_&i..xml");
/*--- déclaration du fichier de sortie*/
file "&folder.&delim.PrixCarburants_annuel_&i._final.xml";
/*--- remplissage de l'input buffer*/
input;

/*correction des scories*/
/* Fichier xml 2008 ligne 528841*/
_infile_ = tranwrd(_infile_, "35***", "35000");
/*--- _infile_ est la variable pour acceder au contenu de l input buffer.
htmldecode remplace la valeur de _infile_ avec la meme valeur dont les
balises HTML ont ete converties en caracteres.*/
_infile_ = tranwrd(_infile_, "&#x8C;", "OE");
/*_infile_ = tranwrd(_infile_, "&#xCODE;","œ");*/
_infile_ = tranwrd(_infile_, "&#xC3;&#x89;", "É");
_infile_ = tranwrd(_infile_, "&#xC3; ", "à");
/*_infile_ = tranwrd(_infile_, 'encoding="ISO-8859-1"', 'encoding="UTF-
8"');*/

_infile_ = htmldecode(_infile_);
/*--- ecriture de la ligne convertie vers le fichier de sortie*/
put _infile_;
run;
%end;
%mend createtable;
```

2.3.2. *Autres méthodes de conversion :*

2.3.2.1. Windows

Sous Windows peu de possibilités s'offrent à l'utilisateur pour effectuer la conversion :

- Pour Notepad++ le plugin HTMLTag http://sourceforge.net/projects/npp-plugins/files/HTMLTag/ provoque une erreur OutOfMemory.
- La modification des tags par « Rechercher-Remplacer » est fastidieuse mais fonctionne.



2.3.2.2. Linux

Sous Linux, un utilitaire existe pour convertir l'intégralité des caractères du document : ascii2uni.
La commande suivante permettra de convertir ces balises dans le caractère accentué correspondant.
ascii2uni -a H PrixCarburants_annuel_2014.xml > PrixCarburants_annuel_2014_final.xml

2.3.3. *Contrôle de la conversion*

Le fichier une fois corrigé des caractères HTML ne doit plus contenir de suite « &#x ».
Les noms accentués doivent s'afficher correctement.
Exemple :

```
1  <?xml version="1.0" encoding="ISO-8859-1" standalone="yes"?><pdv_liste>
2  <pdv id="1000001" latitude="4620114" longitude="519791" cp="01000" pop="R">
3    <adresse>596 AVENUE DE TREVOUX</adresse>
4    <ville>SAINT-DENIS-LèS-BOURG</ville>
5    <ouverture debut="01:00" fin="01:00" saufjour=""/>
6    <services>
```

2.4. Import du XML dans SAS

Les données XML peuvent être importées dans SAS avec le moteur libname xmlv2. Cet import nécessite l'utilisation d'une map XML. La map XML est fournie par le code complet donné en fin d'article mais nous allons détailler la création de cette map dans les étapes suivantes. Il est nécessaire d'installer SAS XML Mapper pour pouvoir créer la map XML.

2.5. Création d'une map XML avec SAS XML Mapper.

Une map XML Mapper est fournie en fin d'article via un code SAS de génération automatique. Cette partie vous permettra de vous familiariser avec le format XML et les données utilisées.

La map XML indique à SAS quelles sont les entités du fichier XML et dans quelle table les enregistrer. La création d'une map XML se fait avec l'outil SAS MXL Mapper disponible à cette adresse : http://support.sas.com/downloads/browse.htm?cat=12

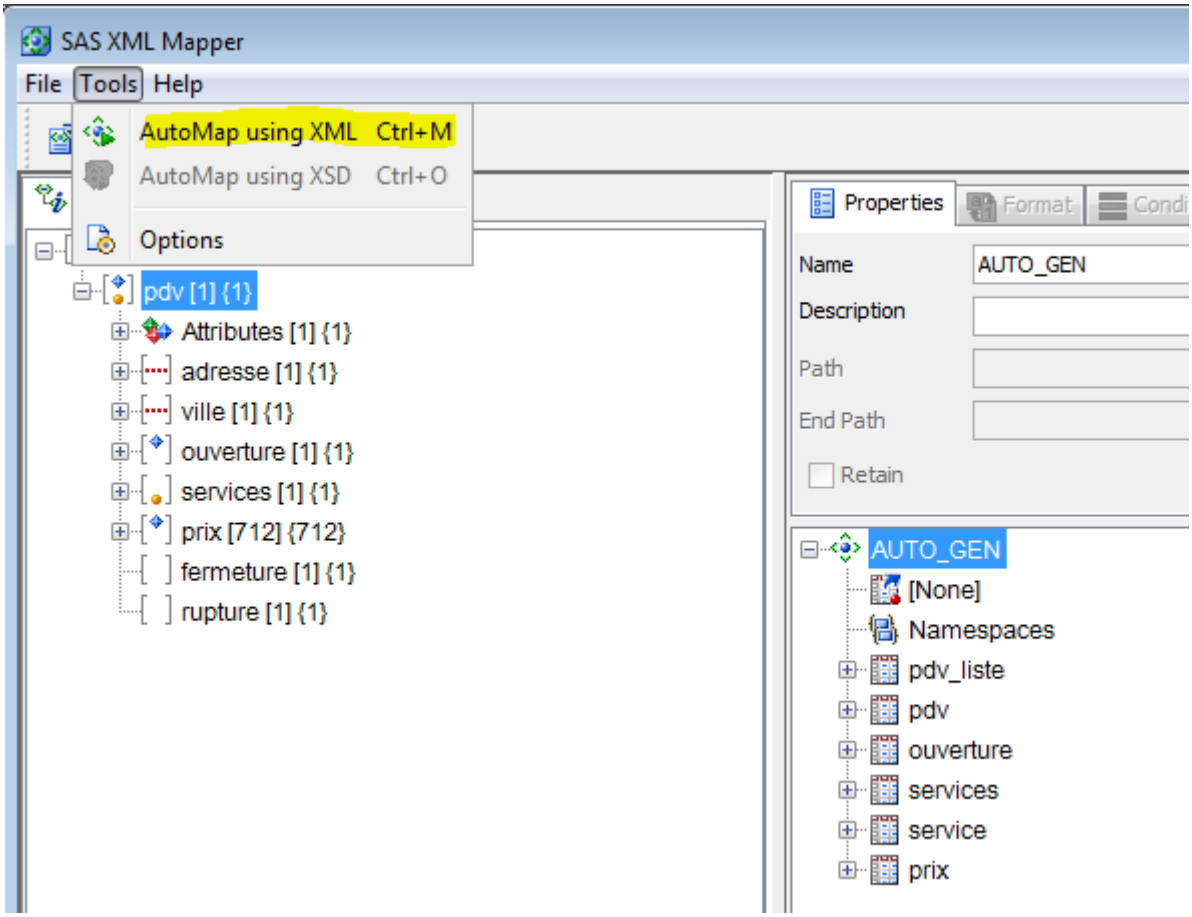
Une fois SAS XML Mapper installé, il est intuitif de vouloir ouvrir le fichier PrixCarburants_annuel_2014_final.xml. Toutefois, le nombre d'éléments dans ce fichier est inutilement trop élevé. Il faut donc échantillonner le fichier source.

Pour créer un échantillon, il faut ouvrir le fichier PrixCarburants_annuel_2014_final.xml et ne garder qu'une seule entité <pdv></pdv>. Pour conserver une structure XML valide, il faut insérer à la toute fin </pdv_liste>, puis l'enregistrer sous le nom PrixCarburants_annuel_2014_echantillon.xml.

Lancer SAS XML Mapper et ouvrir le fichier PrixCarburants_annuel_2014_echantillon.xml.

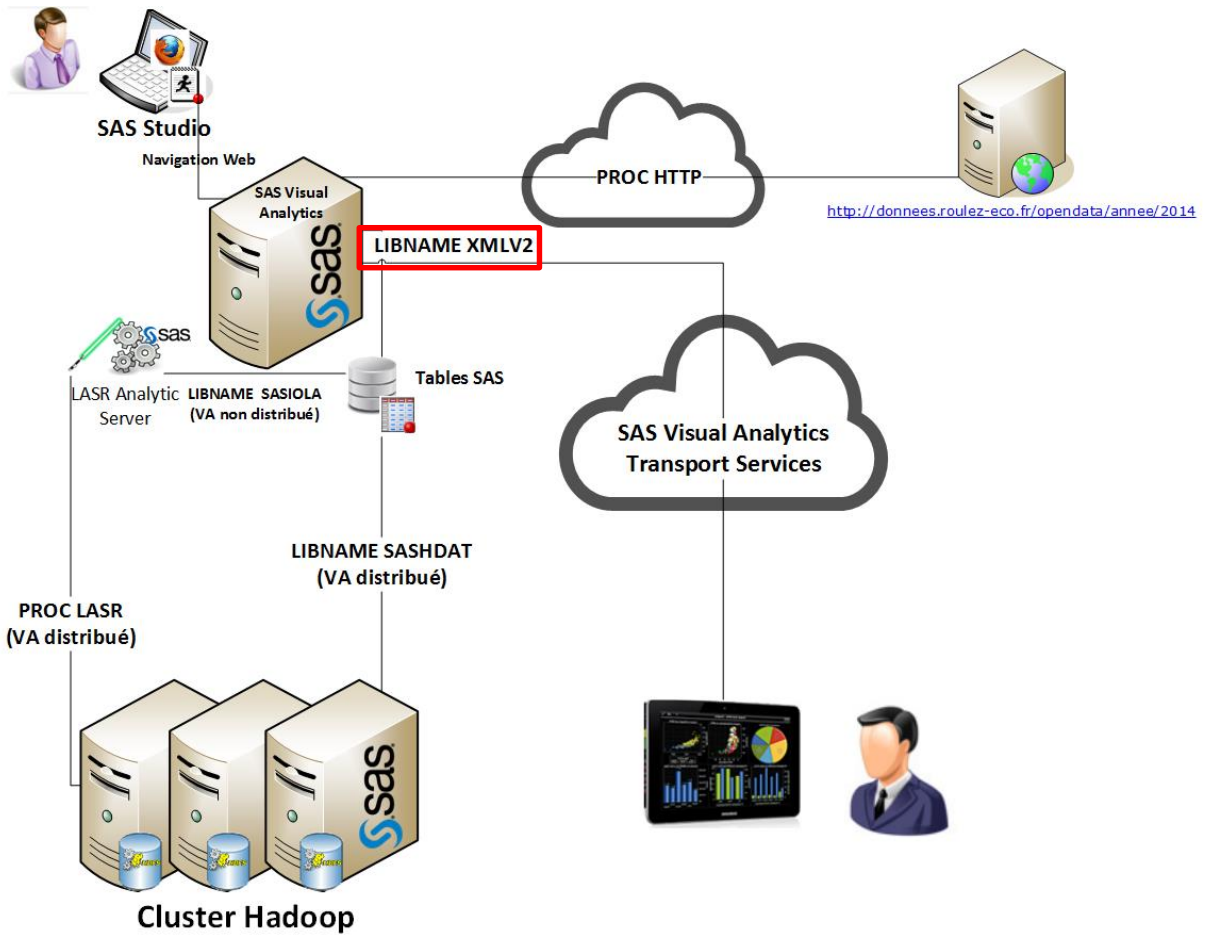
La structure du XML est visible dans la colonne de gauche. Celle de la map XML dans la fenêtre principale. La création de la map consiste à glisser-déposer les entités de l'échantillon (pdv, prix, services …) sur la map (SXLEMAP).

Dans l'article nous utiliserons la fonctionnalité auto-map disponible dans le menu « Tools ».



La map XML sera sauvegardée sous le nom prix_essence_annee.map .

2.6. Création des tables SAS.



La création des tables SAS peut se faire via SAS Base, SAS Enterprise Guide ou encore SAS Studio (interface de programmation en ligne).

Dans l'article c'est SAS Studio qui a été utilisé. C'est l'équivalent de SAS Base (à quelques différence près) et disponible via une interface web. Dans l'article le serveur s'appelle « vamaster ». Le chemin de l'application est SASStudio. L'accès complet à l'application est http://vamaster:7980/SASStudio. Cet URL est à adapter en fonction de votre environnement.

Deux tables seront créées : prix et pdv.
La table PDV (Point de vente) contient toutes les informations d'un point de vente :
Adresse, localisation GPS, horaires, services fournis. A noter que la version payante non OpenData contient l'entreprise d'appartenance des stations.
La table PRIX contient l'historique des prix par jour et par carburant. Chaque historique contient une clé PDV_ordinal. Le champ PDV_ordinal est la clé d'association avec la table PDV.

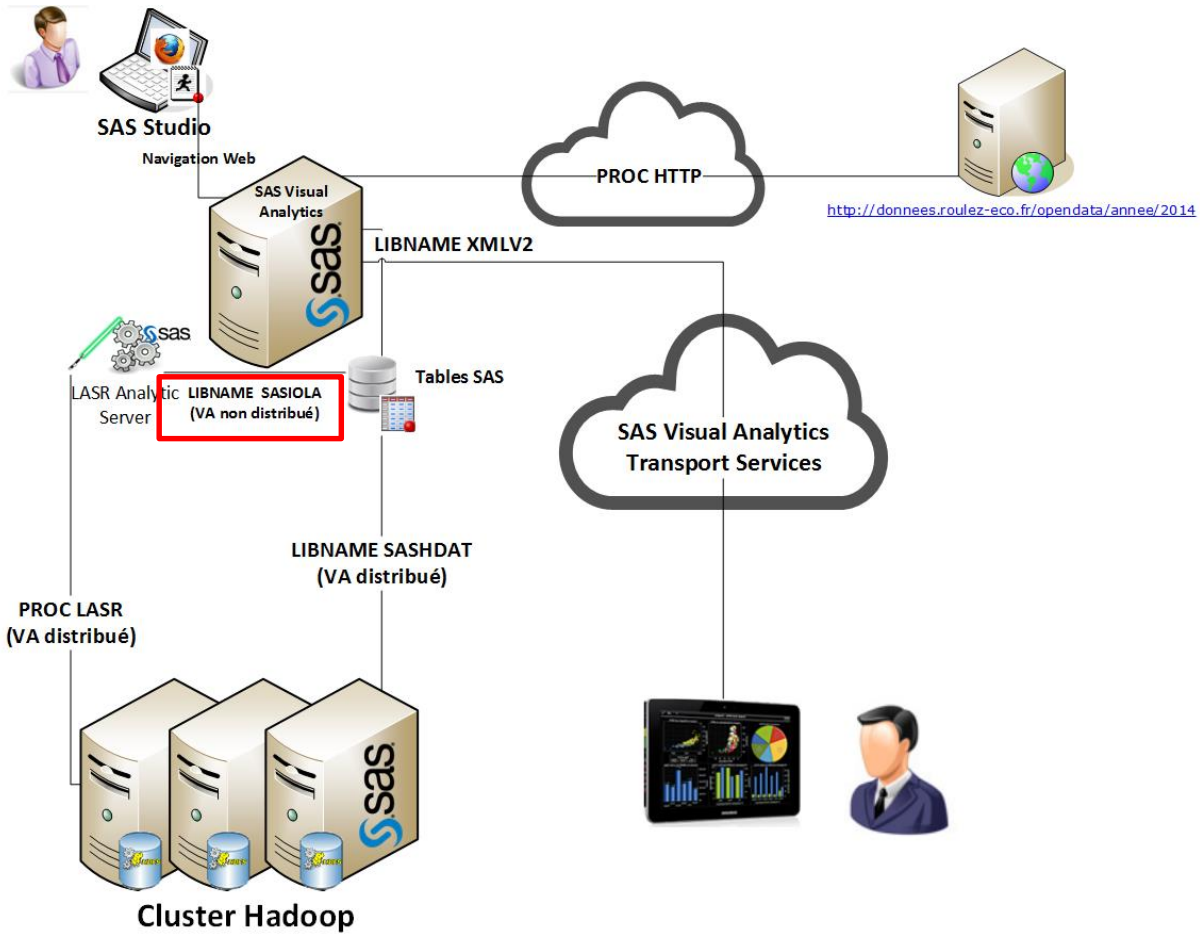
Le code de base pour importer un fichier XML à l'aide de la map associée est le suivant :

```
libname libxml xmlv2 "donnees.xml" xmlmap="xmlmap.map";
libname sortie " c:\dossier\";

data sortie.donnees;
        set libxml.data;
run;
```

Dans le cadre des données de l'essence le code d'import est exécuté en même temps que la conversion ce qui explique la présence de macro-variables qui seront utilisées dans une boucle. Le code complet est disponible en fin d'article.

```
libname essxml xmlv2 "&folder.&delim.PrixCarburants_annuel_&i._final.xml"
xmlmap="&folder.&delim.PrixCarburants_annuel_xmlmap.map";

data prix_tmp&i/*(drop=prix_maj)*/;
        set essxml.prix;
        format prix_majdate datetime21.2 prix_euros 5.3;
        prix_majdate=input(trim(prix_maj),ymddttm24.);
        prix_euros=trim(prix_valeur/1000);
run;

proc sort data=prix_tmp&i;
by pdv_ORDINAL;
run;

data pdv_tmp&i;
        set essxml.pdv;
        taille=1;
        ville=propcase(ville);
        adresse=propcase(adresse);
        pdv_latitude=pdv_latitude/100000;
        pdv_longitude=pdv_longitude/100000;
run;

proc sort data=pdv_tmp&i;
by pdv_ORDINAL;
run;
proc append base=essence.pdv data=pdv_tmp&i FORCE;
run;
proc append base=essence.prix data=prix_tmp&i FORCE;
run;
```

2.7. Chargement des tables dans le serveur LASR (SAS VA non-distribué)

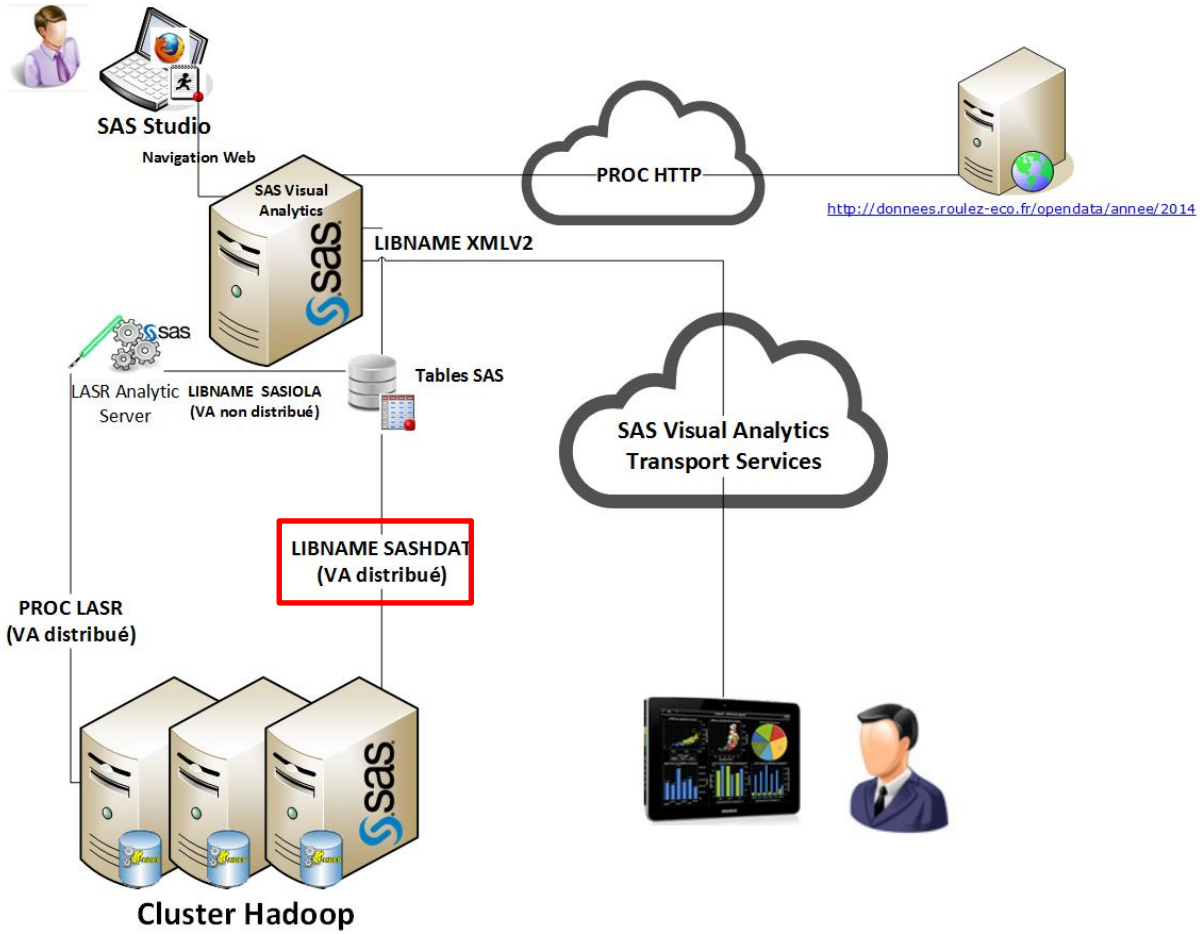


(Voir les chapitres 2.8 et 2.9 pour l'utilisation d'un serveur LASR distribué sur Hadoop.)

Les tables essence.pdv et essence.prix peuvent être chargées dans le serveur LASR non distribué via un libname SASIOLA et une étape data.

```
libname lasrlib sasiola startserver=(path="/opt/sas/config/LASRSigFiles/")
    host="vamaster" port=10010 tag='hps'
    signer="http://vamaster:7980/SASLASRAuthorization";

data lasrlib.pdv;
      set essence.pdv;
run;

data lasrlib.prix;
      set essence.prix;
run;
```

2.8. Chargement des tables dans Hadoop (SAS VA distribué)



/! Il est très important d'utiliser le libref « vapublic » ainsi que le path « /vapublic » pour charger les tables ensuite dans le serveur LASR Public. Ce nom doit correspondre à celui présent dans les propriétés de la librairie HDFS Public ainsi que du serveur LASR Public.

/! Les tables SAS Base sas7bdat ne peuvent pas être chargées dans le serveur LASR via hadoop (commande hadoop fs -put table.sas7bdat). Pour charger des tables qui seront utilisées par le serveur LASR, il faut passer par le format SASHDAT et son libname associé.

```
LIBNAME vapublic SASHDAT PATH="/vapublic"
SERVER = "vamaster" INSTALL = "/opt/TKGrid";

PROC DATASETS LIB=VAPUBLIC;
      delete pdv prix;
run;

data VAPUBLIC.pdv;
      set essence.pdv;
run;

data VAPUBLIC.prix;
      set essence.prix;
run;
```

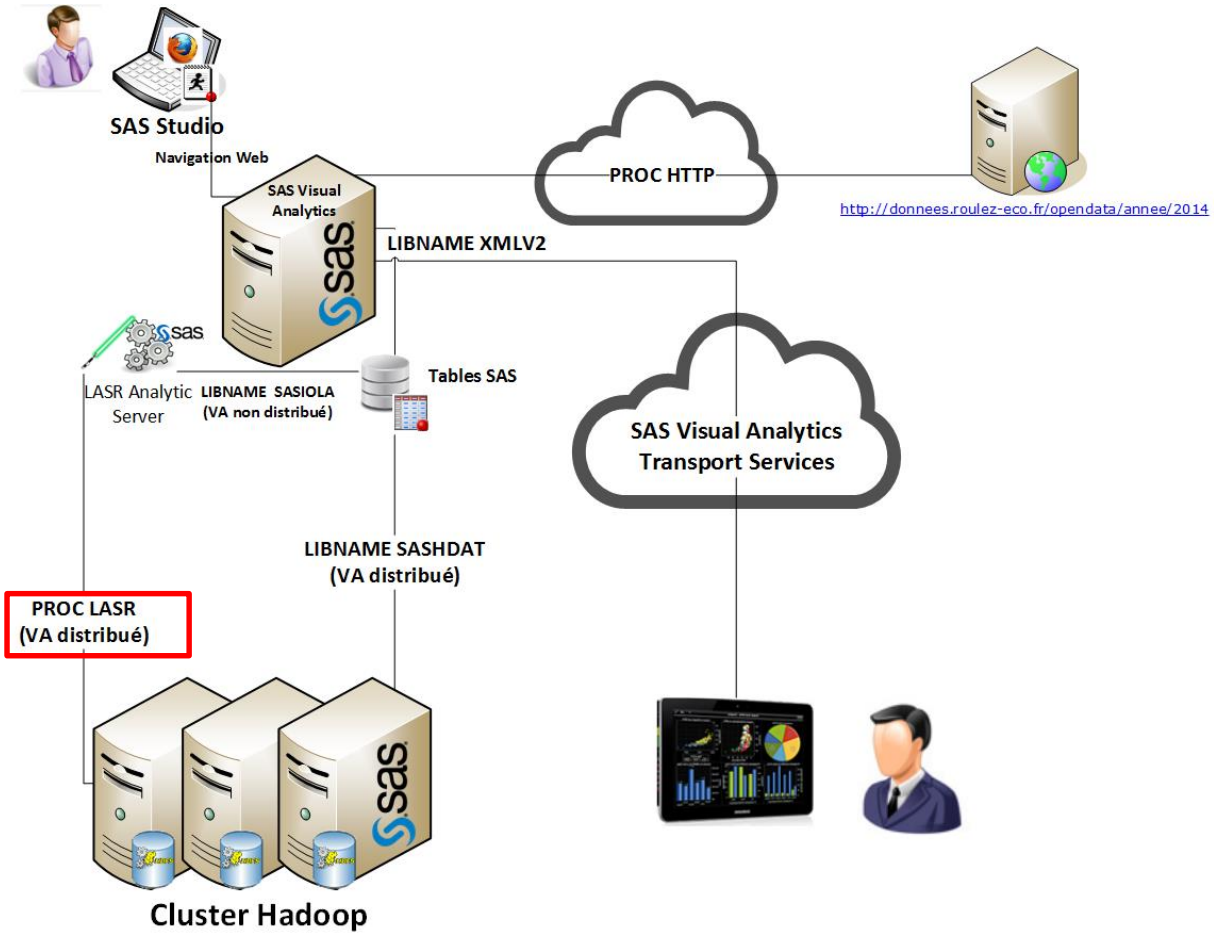
Une fois l'import effectué il est possible de vérifier qu'il s'est bien déroulé en vérifiant la présence des tables SASHDAT sur le cluster hadoop. Dans le cadre d'une installation de hadoop fournie par SAS l'url http://vamaster:50070/explorer.html#/vapublic doit afficher :

Browse Directory

/vapublic Go!

Permission	Owner	Group	Size	Replication	Block Size	Name
-rw-r--r--	lasradm	sas	60.04 MB	2	2 MB	pdv.sashdat
-rw-r--r--	lasradm	sas	1.44 GB	2	2 MB	prix.sashdat

2.9. Chargement des données dans le serveur LASR Public (SAS VA distribué)



Les tables stockées dans le cluster hadoop peuvent être chargées dans le serveur LASR. Pour cela c'est la PROC LASR qui va être utilisée.

```
proc lasr port=10031
   data=vapublic.pdv
   signer="http://vamaster:7980/SASLASRAuthorization"
    add noclass;
   performance
      host="vamaster";
run;

proc lasr port=10031
   data=vapublic.prix
   signer="http://vamaster:7980/SASLASRAuthorization"
    add noclass;
   performance
      host="vamaster";
run;
```

En l'absence de message d'erreur les tables sont bien chargées dans le serveur LASR. La log indique :

```
NOTE: The LASR procedure is executing in the distributed computing environment with 3 worker nodes.
 NOTE: Data loading into the LASR Analytic Server required 4.3530268669 seconds.
 NOTE: The HDFS file /vapublic/pdv has been added to the LASR Analytic Server.
 NOTE: PROCEDURE LASR used (Total process time):
      real time           16.69 seconds
      cpu time            2.11 seconds
```

2.10. Association des tables via la PROC IMSTAT (création d'un schéma en étoile)

A ce stade les tables sont chargées dans le serveur LASR mais elles sont encore indépendantes l'une par rapport à l'autre. Nous allons utiliser les outils IN-MEMORY du serveur LASR pour créer une seule table unique. La procédure IMSTAT est utilisée dans ce cas-là.

La proc IMSTAT permet une instruction merge comme dans un DATA STEP. La différence est que la PROC IMSTAT travaillant directement dans la mémoire RAM du cluster, le résultat est obtenu beaucoup plus rapidement.

La proc IMSTAT associe par la clé pdv_ORDINAL les informations d'un point de vente à un prix.
La PROC IMSTAT définit la table de base comme la « table de faits ».

La macro vdb_dt permet de supprimer toute table déjà chargée en mémoire. La table créée par le schéma en étoile est `LASRLIB.ESSENCE`

Ensuite les tables prix et pdv n'étant plus utiles elles sont supprimées du serveur LASR.

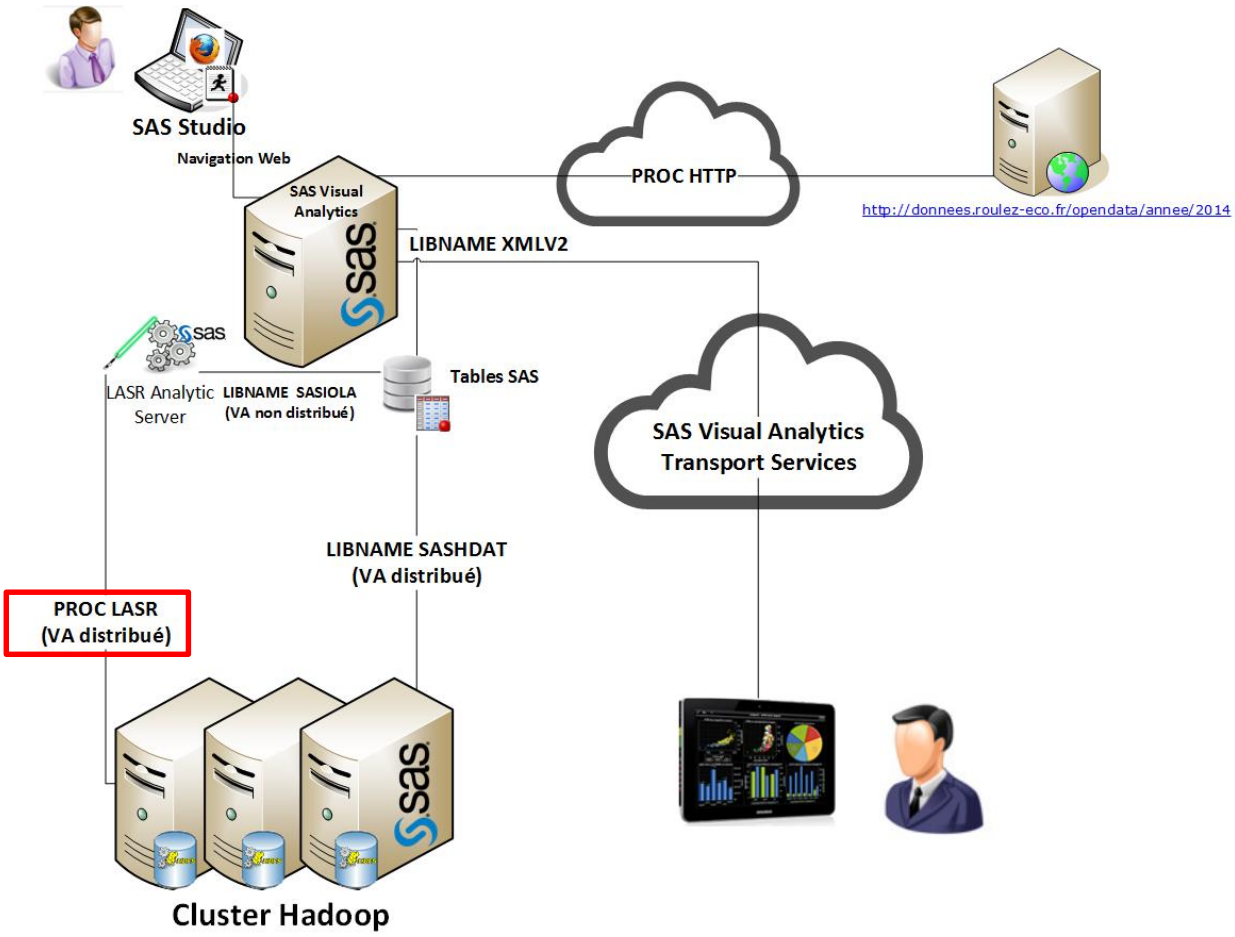
```
/** LIBNAME SASIOLA (port 10031 dans le cadre d'un serveur distribué) **/
LIBNAME LASRLIB SASIOLA  TAG=VAPUBLIC  PORT=10031 HOST="vamaster"
SIGNER="http://vamaster:7980/SASLASRAuthorization" ;

/** SUPRESSION DE LA TABLE ESSENCE **/
%vdb_dt(LASRLIB.ESSENCE);

/** LASR STAR SCHEMA CODE **/
proc imstat;
   table LASRLIB.PRIX;
   schema
      PDV ( pdv_ORDINAL = pdv_ORDINAL
       / prefix = PDV )
   ;
   run;
   table LASRLIB.&_templast_;
   promote ESSENCE;
   run;
quit;

proc lasr port=10031;
    remove vapublic.prix;
    remove vapublic.pdv;
      performance
    host="vamaster ";
run;
```

2.11. Enregistrement des tables dans les métadonnées

Pour que la table ESSENCE apparaisse dans les interfaces WEB de SAS Visual Analytics, une PROC METALIB est exécutée.

```
proc metalib;
 omr ( library="Visual Analytics Public LASR" REPNAME="Foundation" );
 folder="/Shared Data/SAS Visual Analytics/Public/LASR";
 select ("PDV" "PRIX" "ESSENCE");
 update_rule=(delete);
report;
run;
```

2.12. Sauvegarde de la table finale ESSENCE dans Hadoop (SAS VA distribué)



Afin de ne pas avoir à ré-exécuter le code d'import, il est nécessaire de sauvegarder la table ESSENCE en dehors de la mémoire RAM du cluster.

Le code suivant permet de sauvegarder la table dans Hadoop.

```
option set=GRIDHOST="vamaster";
option set=GRIDINSTALLLOC="/opt/TKGrid";

proc lasr port=10031;
      save vapublic.essence / path="/vapublic" copies=1 blocksize=32m;
run;
```

L'interface Hadoop doit afficher la table essence.sashdat :
http://vamaster:50070/explorer.html#/vapublic

Browse Directory

/vapublic Go!

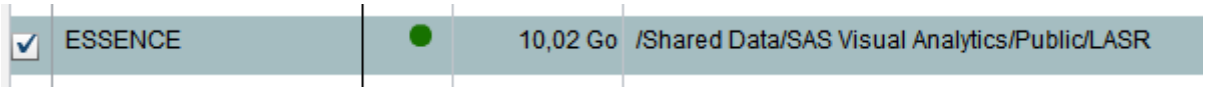
Permission	Owner	Group	Size	Replication	Block Size	Name
-rw-r--r--	lasradm	sas	10.06 GB	3	32 MB	essence.sashdat
-rw-r--r--	lasradm	sas	60.04 MB	2	2 MB	pdv.sashdat
-rw-r--r--	lasradm	sas	1.44 GB	2	2 MB	prix.sashdat

La suite se passe dans l'interface SAS Visual Analytics :

http://vamaster:7980/SASVisualAnalyticsAdministrator/

3. EXPLORATION DANS SAS VISUAL ANALYTICS

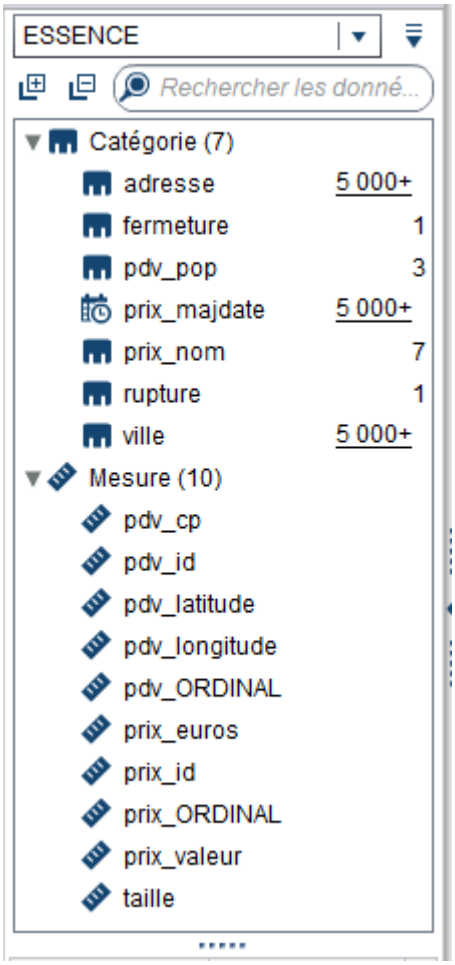
La première exploration peut se faire après avoir validé que la table ESSENCE a bien été chargée dans le serveur LASR. Pour cela l'exécution du code précédemment décrit ne doit pas renvoyer d'erreur. Ensuite dans l'interface d'administration la table ESSENCE doit s'afficher :



3.1. Validation des données : création d'une exploration

Une fois chargée dans SAS Visual Analytics, il faut cliquer sur « Créer une exploration »

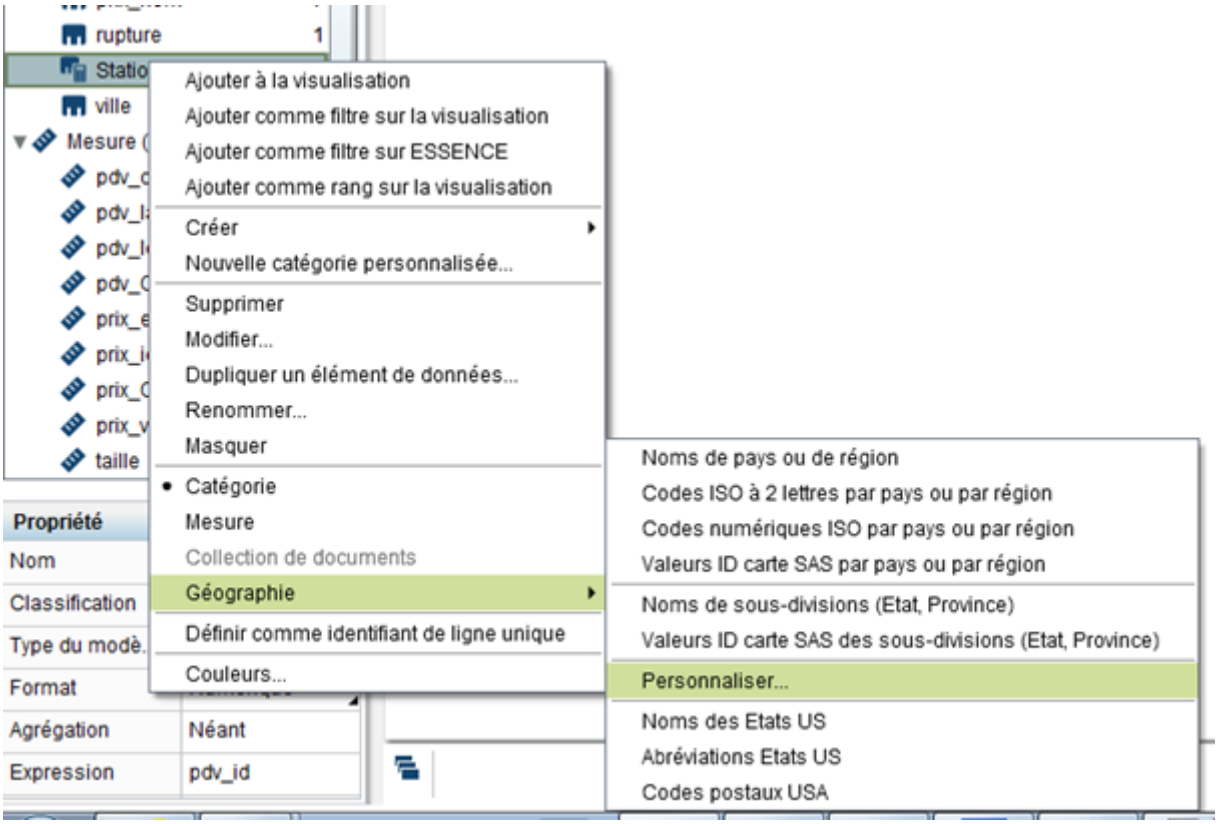
Puis Démarrer une nouvelle exploration -> « Sélectionnez une source de données » -> « ESSENCE » -> « Ouvrir »
Si la table Essence ne s'affiche pas c'est qu'elle n'est pas chargée dans le serveur LASR.
La liste des colonnes de la table s'affiche alors sur la droite :



Nous allons prendre la donnée pdv_id, la dupliquer (clic droit -> « dupliquer »), changer son type en « catégorie » (clic droit -> « Catégorie ») et la renommer « Station Essence » (clic droit -> « Renommer »). La variable doit s'afficher dans la liste sous la forme :

La deuxième étape sera de créer des points géo localisés. Pour cela il faut sélectionner le format « personnalisé » pour la variable « Station essence » comme suit :

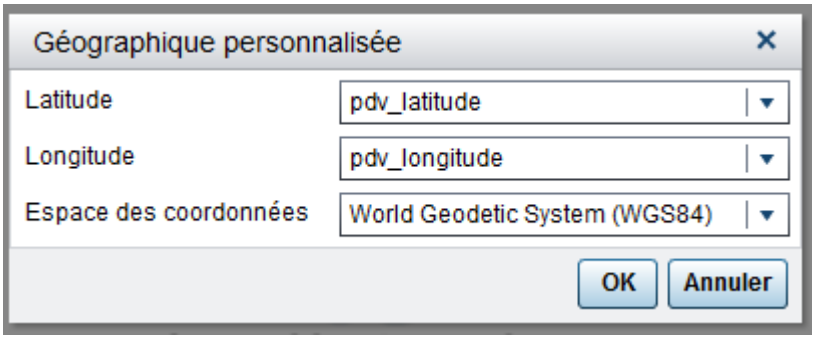
Clic droit -> Géographie -> Personnaliser



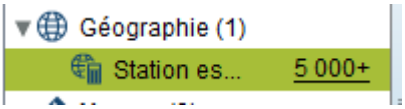
Puis faire correspondre les valeurs :

Latitude -> pdv_latitude

Longitude -> pdv_longitude

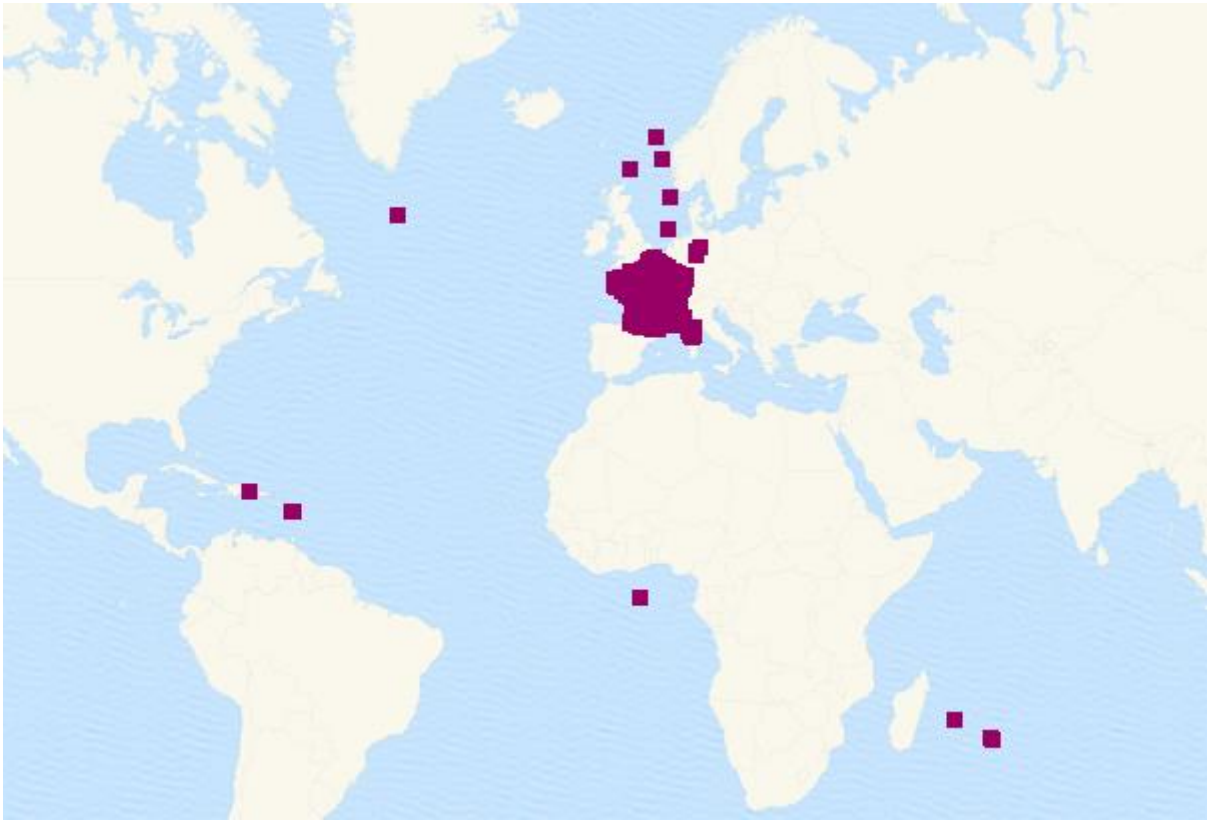


Cliquer sur OK. La variable Station essence s'affiche désormais sous la forme :



Il faut maintenant faire un glisser-déposer sur la zone principale après avoir accepté l'avertissement.

La carte du monde doit s'afficher sous la forme :



La première constatation est qu'il existe des points placés bizarrement en dehors de la métropole, des départements / régions / collectivité d'outre-mer.

Il s'agit de données dont les coordonnées GPS ne sont pas correctes. C'est un problème de qualité des données sources.

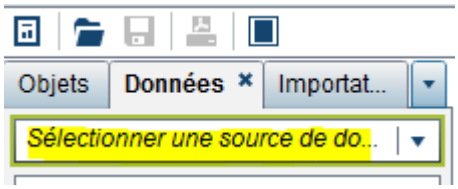
Les données GPS sont toutefois dans la grande majorité exploitables. Nous pouvons passer à la création d'un rapport.

4. Création d'un rapport SAS Visual Analytics

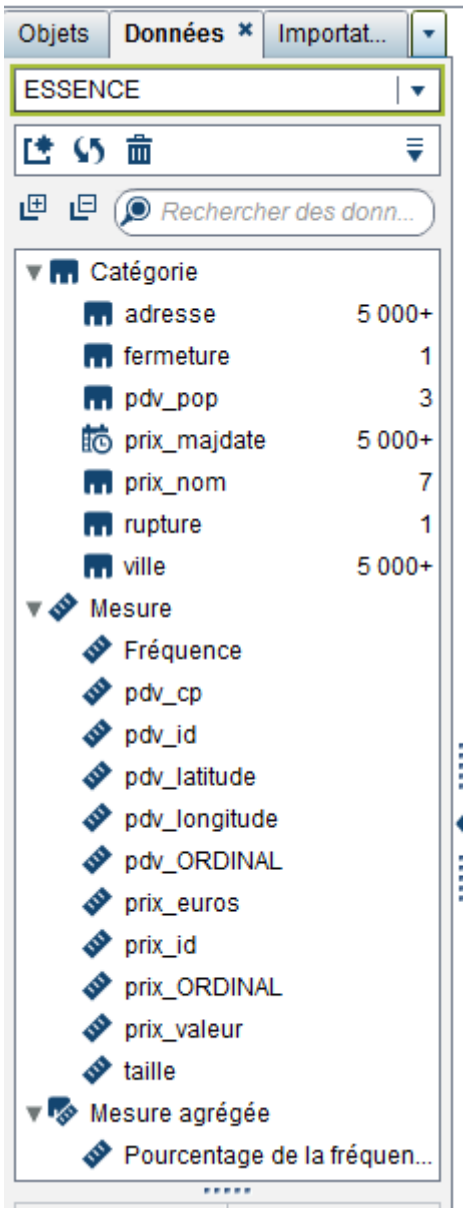
Les données ont été chargées dans le serveur LASR via la création de l'exploration. (vous pouvez aussi utiliser la fonction autoload de SAS Visual Analytics).

La création d'un rapport se fait via l'application SASVisualAnalyticsDesigner (https://vamaster:7980/SASVisualAnalyticsDesigner) ou via le bouton du Hub :

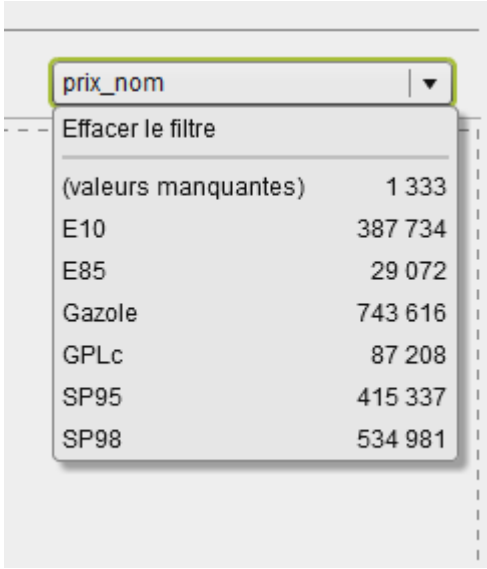
Il faut d'abord sélectionner la source de données ESSENCE.



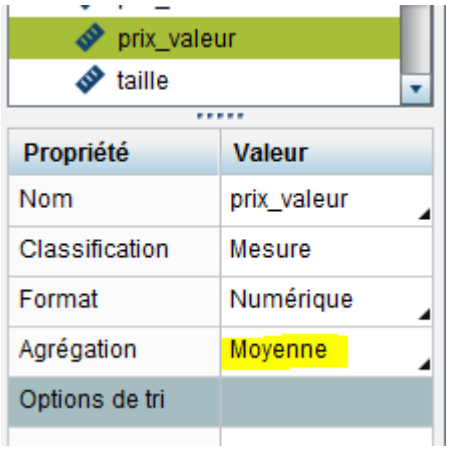
Les données doivent s'afficher comme ceci :



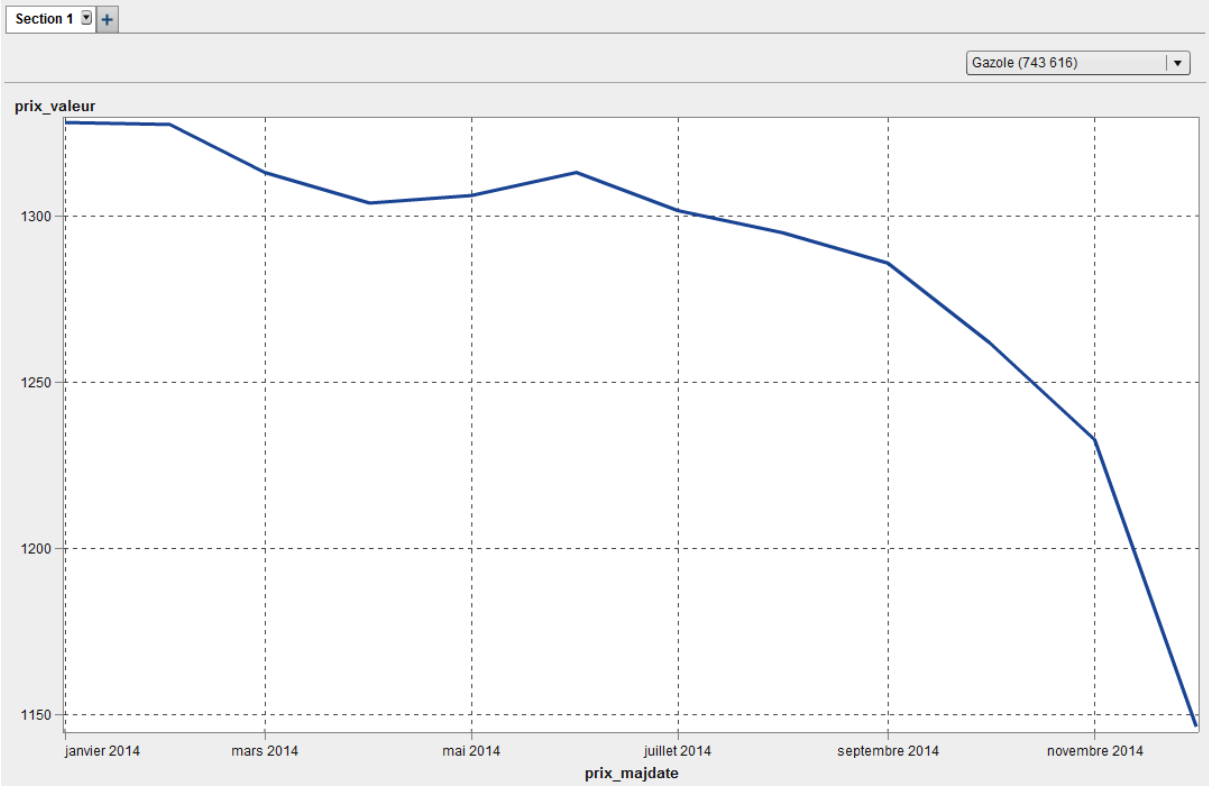
Un objet de type « liste déroulante est ensuite glissé-déposé sur l'en-tête du rapport. La variable prix_nom lui est affectée aussi par glisser-déposer:



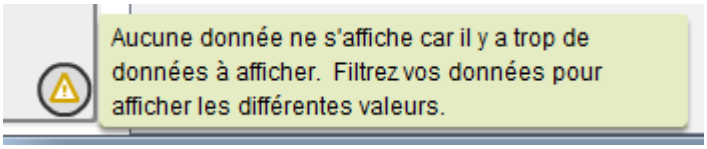
La variable prix_valeur doit avoir comme agrégation « Moyenne ». Il est possible aussi d'agréger par médiane.



Ensuite nous insérons un objet « graphique de série chronologique » dans le corps du rapport et nous glissons-déposons les valeurs prix_majdate et prix_valeur. Le résultat obtenu est le suivant :



Lors de l'ajout d'une carte géographique à bulles pour insérer les points de ventes le message suivant est affiché :

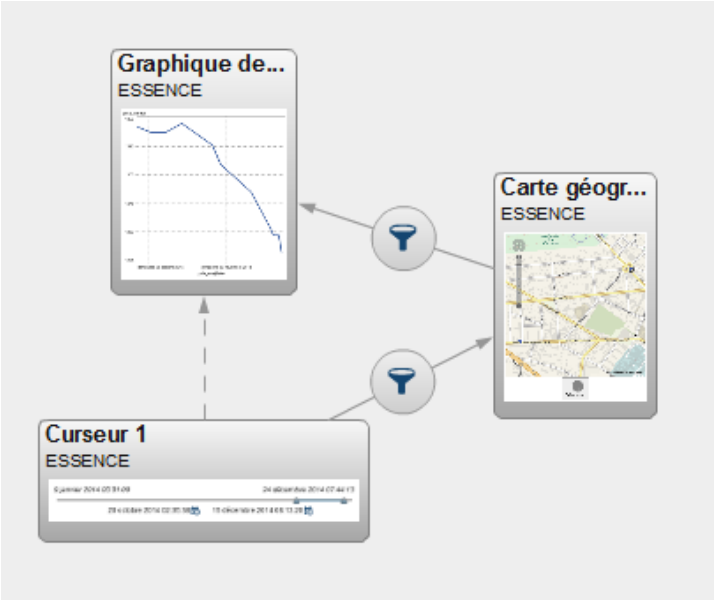


Il faut en effet filtrer les données car en l'état une agrégation de trop nombreuses valeurs ferait perdre l'intérêt analytique du rapport.

Nous insérons :

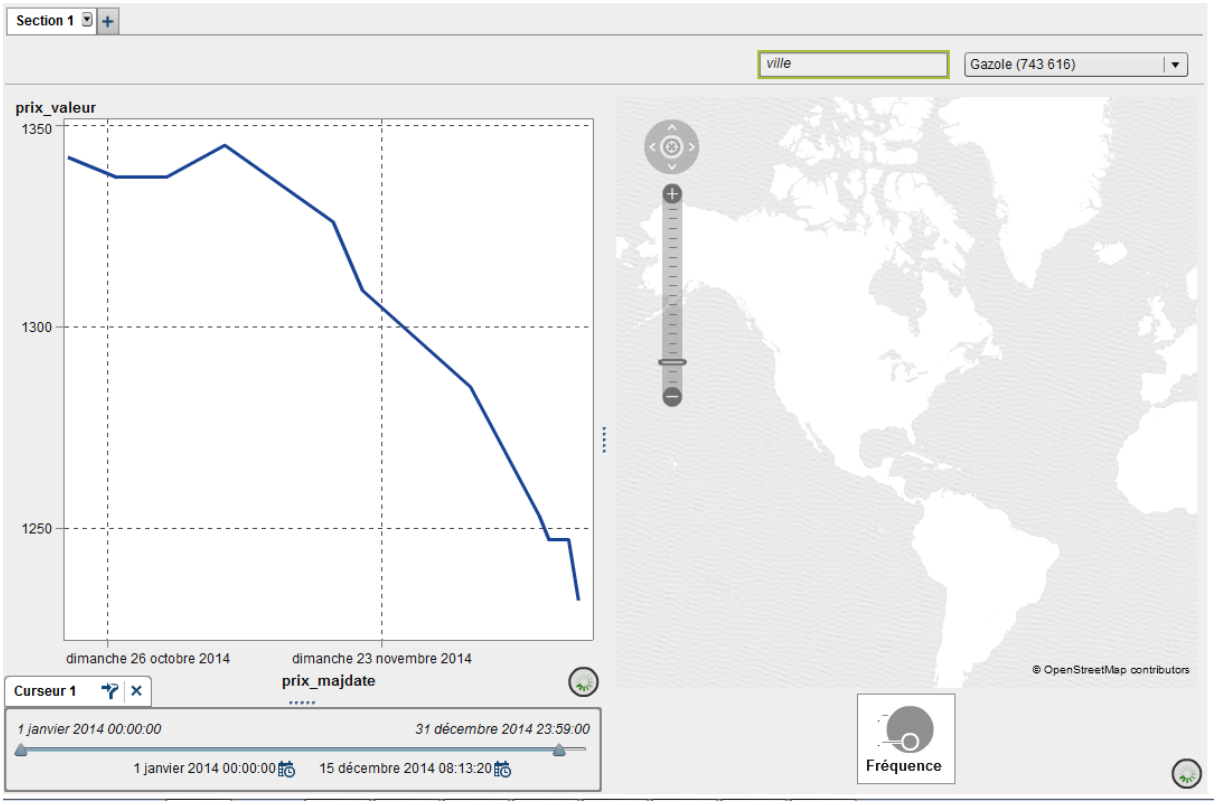
- Une zone de texte en en-tête
- Un curseur sous la série chronologique.

Nous ajoutons deux interactions (Affichage => Interactions) entre le curseur et la carte géographique puis une nouvelle interaction entre la carte géographique et la série chronologique.

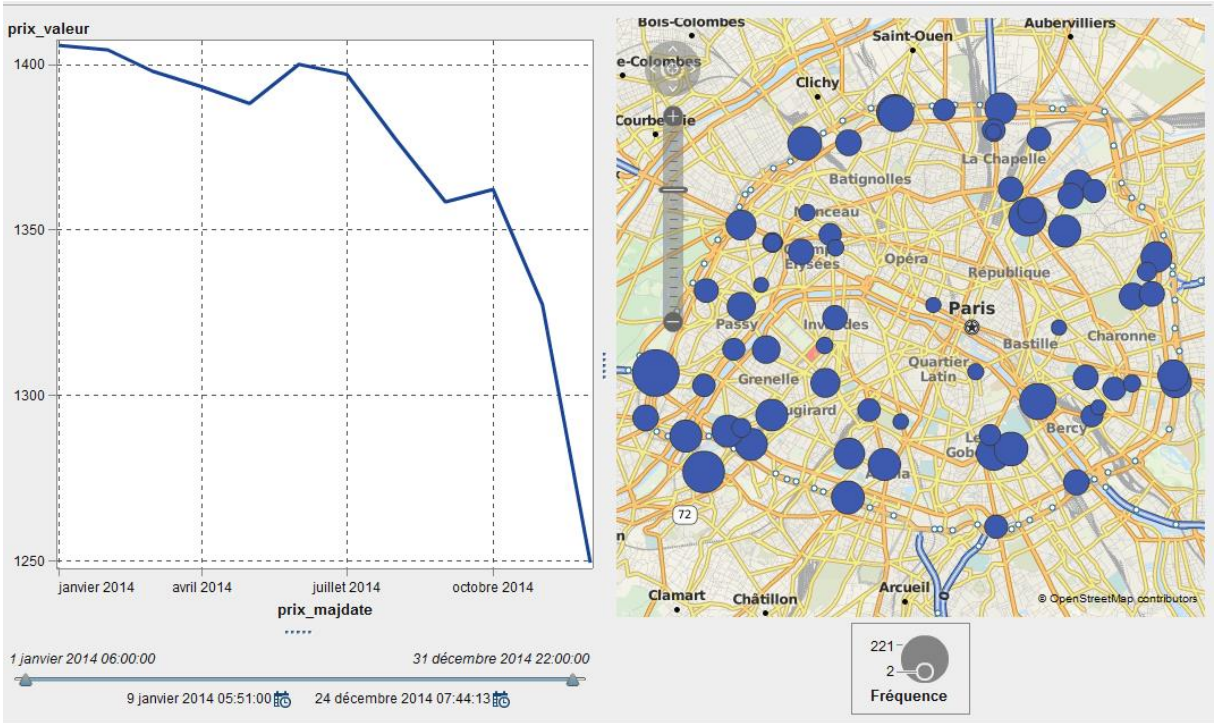


Au final nous obtenons :

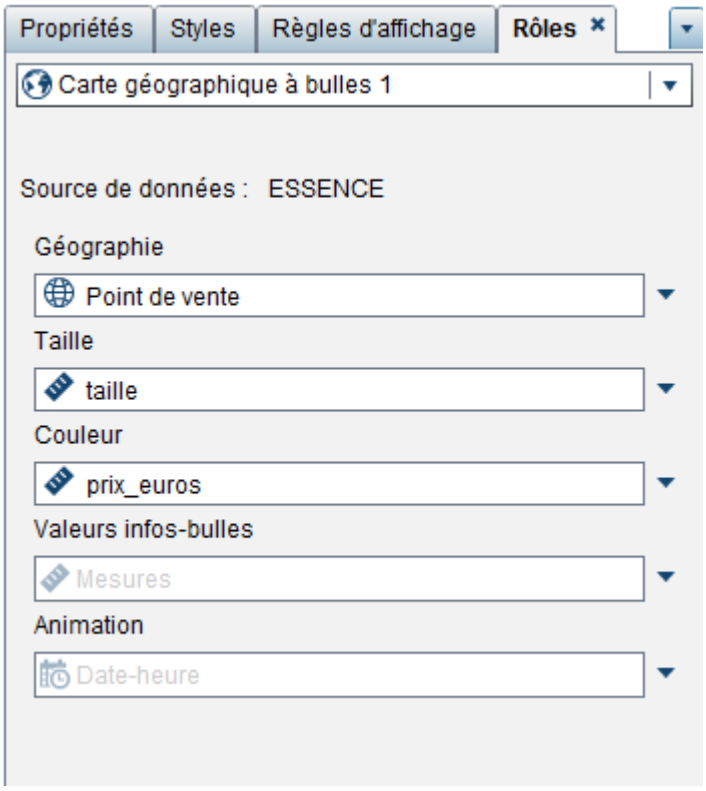
Pour revenir à la conception du rapport il faut cliquer sur « Affichage => Afficher la disposition ». Le rapport a la forme suivante :



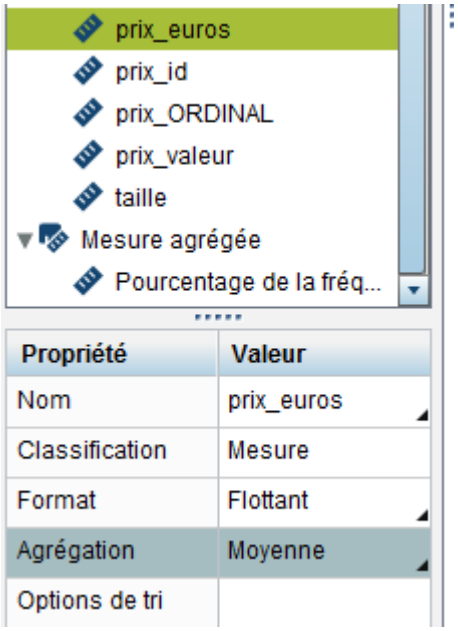
Dans le champ « ville » nous pouvons taper « Paris ». Les stations de la ville s'affichent sur la carte :



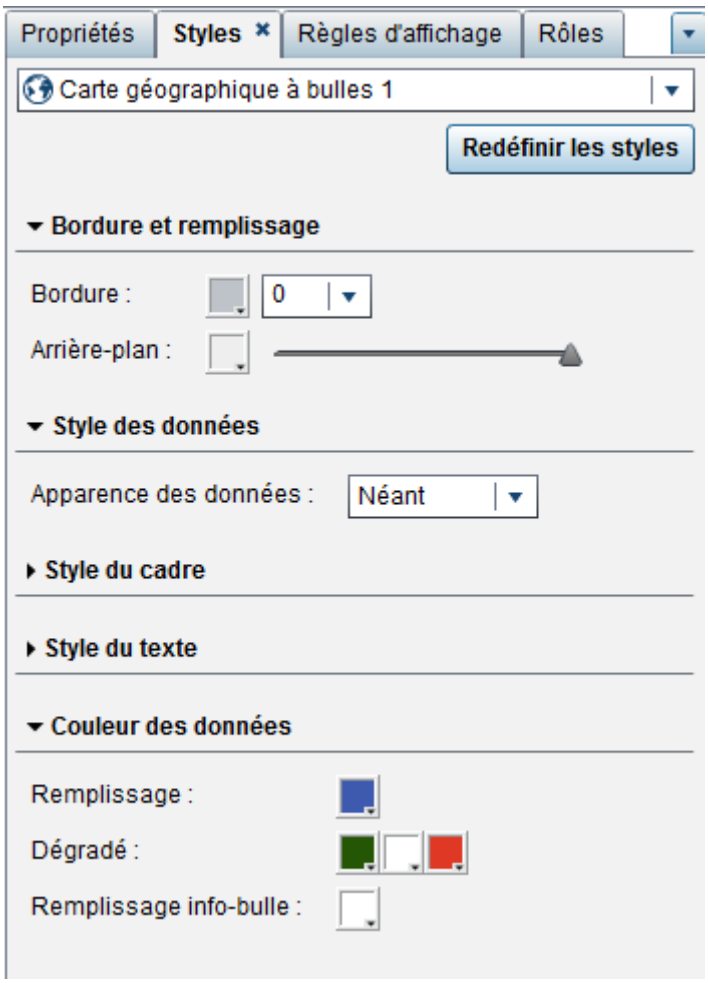
Pour améliorer le rendu, nous modifions les rôles de la manière suivante (menu de droite, onglet Rôles):



L'agrégation de la valeur « taille » et « prix_euro » sera « Moyenne »

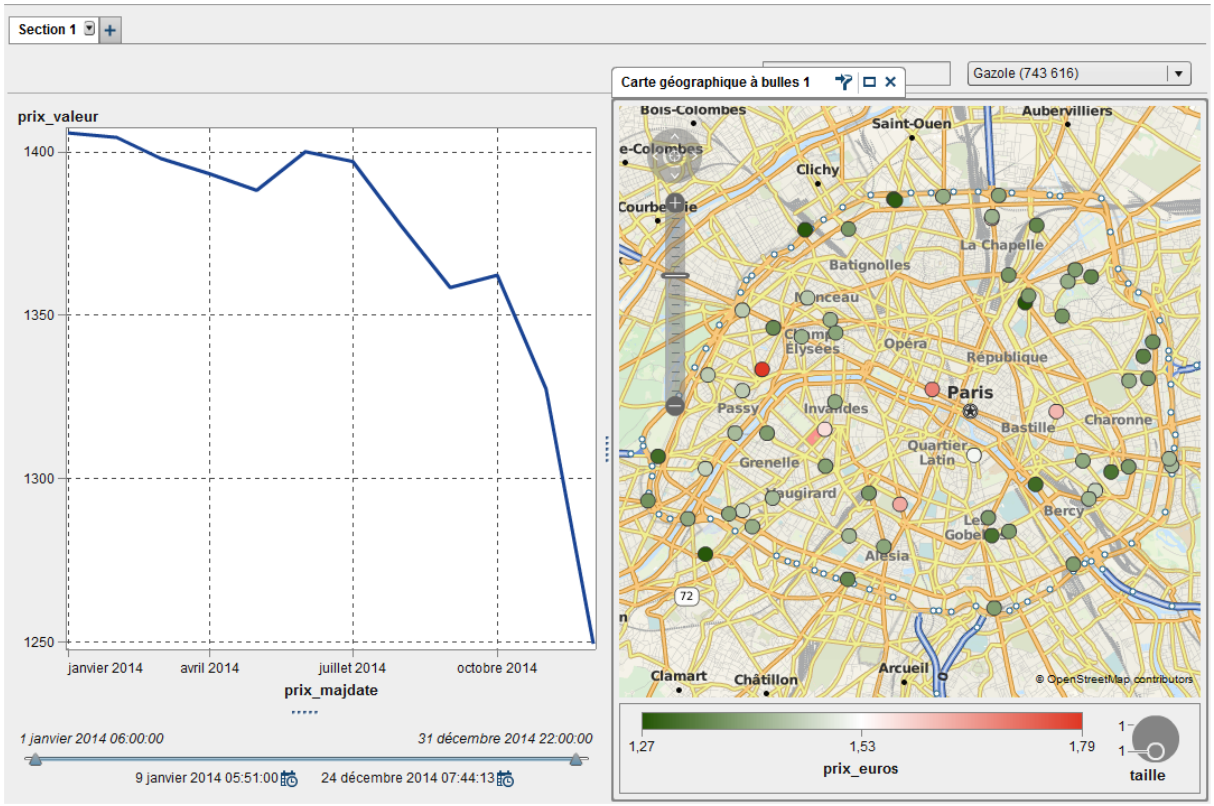


Pour donner des couleurs cohérentes nous sélectionnerons les options suivantes :
(menu de droite, onglet « Styles », étendre « couleurs des données »)



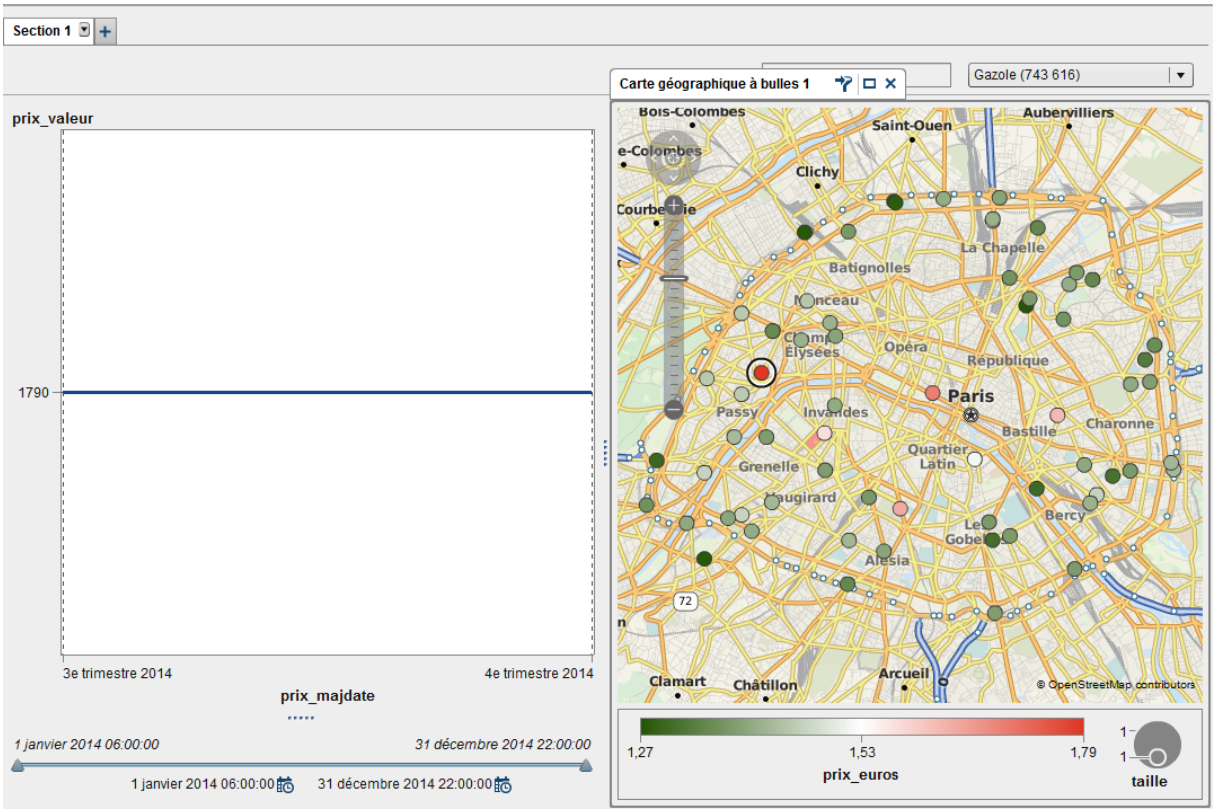
L'aspect du rapport est affiché ci-dessous.

Nous visualisons rapidement le prix moyen en station sur une période donnée dans une ville. Les stations où le prix moyen est le plus élevé et le plus bas sont identifiables facilement.

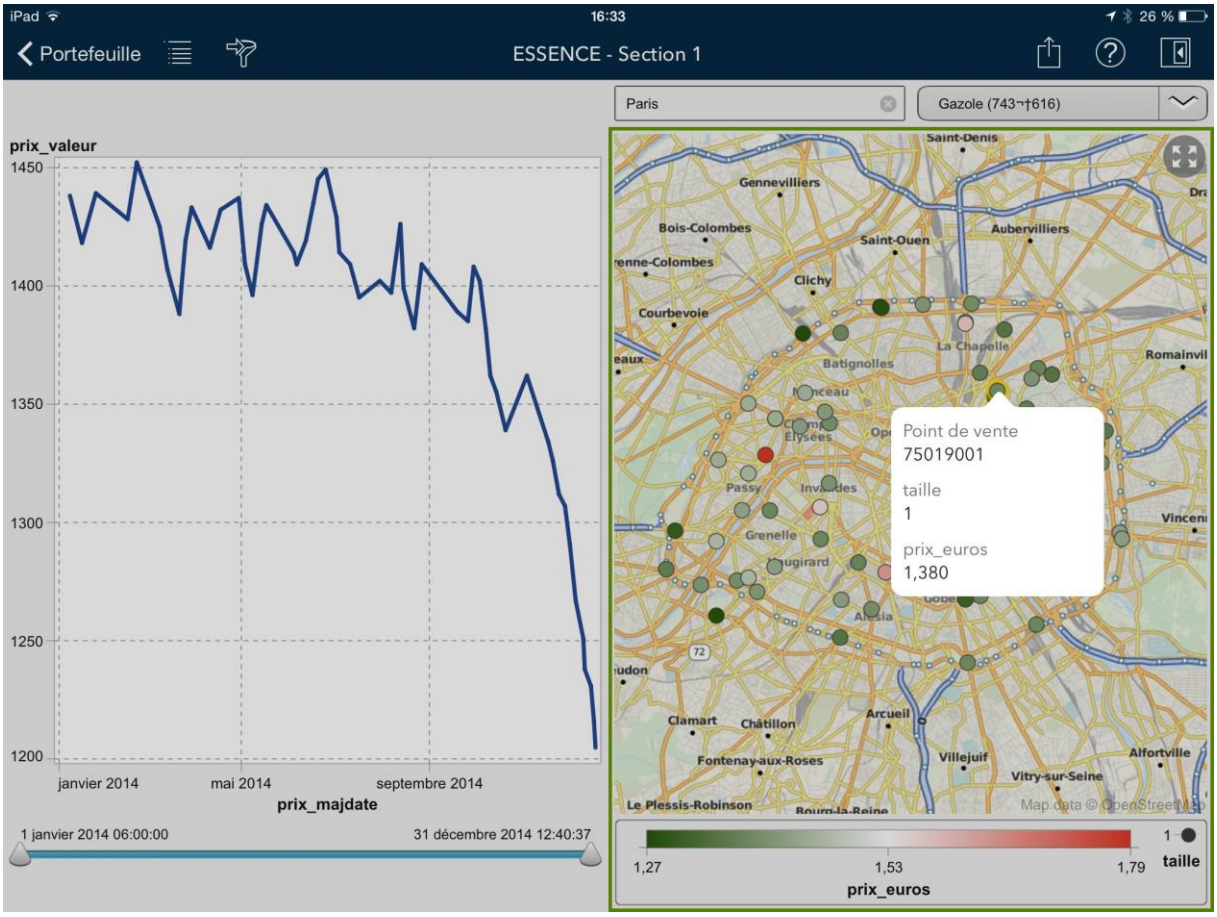


Pour connaitre l'évolution du prix d'une station, il suffit de cliquer dessus. La courbe est mise à jour automatiquement. Pour désélectionner la station, il faut faire CTRL + clic.

Ceci permet de détecter les stations dont les données ne sont pas représentatives :



Sur iPad l'aspect du rapport est le suivant :



5. Ressources

5.1. Code complet

Le code complet utilisé est disponible au téléchargement à l'adresse :
https://github.com/bdherouville/sas/blob/master/essenceopendata.sas

5.2. Erreurs rencontrées :

5.2.1. La table n'apparaît pas dans SAS Visual Analytics

Cela signifie que les métadonnées n'ont pas été mises à jour. La procédure METALIB n'a pas été exécutée ou bien a été exécutée en erreur.

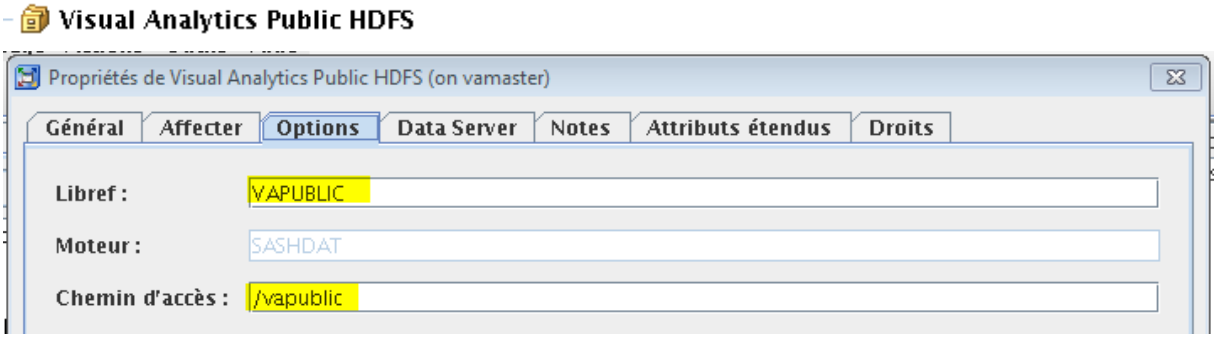
5.2.2. ERROR : Metadata Server denied access to operation

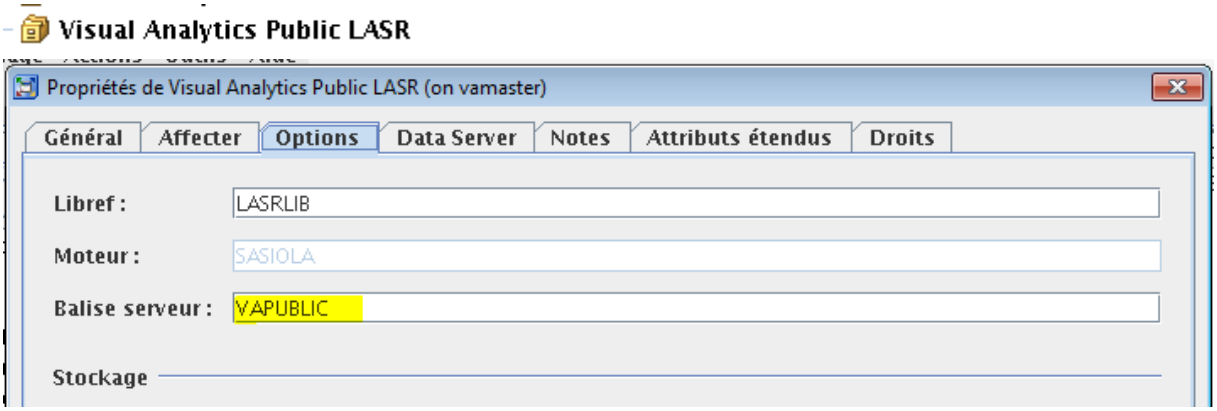
Le plus difficile a été de charger les tables pdv et prix de hadoop au serveur LASR.
Dans la LOG de SAS Studio. Au chargement des tables dans le serveur LASR il y avait le message suivant : « **ERROR : Metadata Server denied access to operation** »

Puis dans la LOG <Lev1>/Web/Logs/SASServer12_1/SASLASRAuthorization7.2.log :

APP 2015-08-07 10:07:45,610 WARN [ST-1970-nQxirc6uqNTFEEPocgUb-cas sasdemo] com.sas.lasr.authorization.service.AuthorizationServiceImpl - ** Access Error - Table not found **
APP 2015-08-07 10:07:45,610 WARN [ST-1970-nQxirc6uqNTFEEPocgUb-cas sasdemo] com.sas.lasr.authorization.service.AuthorizationServiceImpl - Action: 7
APP 2015-08-07 10:07:45,611 WARN [ST-1970-nQxirc6uqNTFEEPocgUb-cas sasdemo] com.sas.lasr.authorization.service.AuthorizationServiceImpl - **Table: VALIBLA.HPS.HPS.PRIX**
APP 2015-08-07 10:07:45,611 WARN [ST-1970-nQxirc6uqNTFEEPocgUb-cas sasdemo] com.sas.lasr.authorization.service.AuthorizationServiceImpl - User: sasdemo
APP 2015-08-07 10:07:45,611 WARN [ST-1970-nQxirc6uqNTFEEPocgUb-cas sasdemo] com.sas.lasr.authorization.service.AuthorizationServiceImpl - **Neither table nor library could be found in metadata.**
APP 2015-08-07 10:07:45,611 WARN [ST-1970-nQxirc6uqNTFEEPocgUb-cas sasdemo] com.sas.lasr.authorization.service.AuthorizationServiceImpl - **Does user have ReadMetadata permissions?**

Résolution : Il faut que le libname SASHDAT et le chemin d'accès fassent référence aux paramètres du serveur HDFS Public et LASR Public. Dans SAS Management Console **:**





6. LIENS UTILES

Données OpenData essence : https://www.data.gouv.fr/fr/datasets/prix-des-carburants-en-france/

Reading and updating ZIP files with FILENAME ZIP :
http://blogs.sas.com/content/sasdummy/2014/01/29/using-filename-zip/

Table de caractères UTF-8 : http://www.utf8-chartable.de/

Documentation SAS Visual Analytics : http://www.sas.com/en_us/software/business-intelligence/visual-analytics.html

LIBNAME SASHDAT :
http://support.sas.com/documentation/cdl/en/inmsref/68736/HTML/default/viewer.htm#p0kn1b8a7yt44fn1qwp8w1b8l92w.htm

PROC LASR :
http://support.sas.com/documentation/cdl/en/inmsref/68736/HTML/default/viewer.htm#p00z16nlibqearn11f67sudjjtd9.htm

PROC IMSTAT (Data and server management):
http://support.sas.com/documentation/cdl/en/inmsref/68736/HTML/default/viewer.htm#p10dosb1fybvpzn1hw38gxuotopk.htm

7. CONCLUSION

Dans cet exercice nous avons pu découvrir que la combinaison du langage SAS Base utilisé avec SAS Studio permet d'effectuer beaucoup d'actions spécifiques : téléchargement de fichier, conversion de caractères, assemblage.
Les différentes procédures : PROC LASR, IMSTAT permettent de manipuler les données in-memory. La procédure metalib enregistre la table qui devient alors disponible pour les utilisateurs de SAS Visual Analytics.

Les deux produits SAS Studio et SAS Visual Analytics sont tout à fait complémentaires et permettent des usages quasi illimités. Cette base ESSENCE in-memory pourra à l'avenir être alimentée automatiquement par l'ordonnancement d'un job.

Bertrand d'Hérouville
Consultant Support Clients SAS France