Structurer et analyser des comptes bancaires avec Trifacta Wrangler

JP Gouigoux (MVP) – Avril 2016

Explications

Je viens juste de finir l'écriture d'un livre sur l'Open Data, dans lequel je montre comment chercher la donnée, puis la nettoyer, l'analyser et la mettre en forme. Afin que le livre ne soit pas trop rébarbatif, ni pour le lecteur ni pour moi-même pour l'écrire, je me suis fait un plaisir de tester plein d'outils divers et variés. Il reste bien sûr de l'Excel et du QlikView, mais je suis tombé sur un outil que je ne connaissais pas avant, et qui est vraiment top pour ce qui est du traitement des données non structurées.

Il s'agit de Trifacta Wrangler, et je vous assure que c'est très différent de tout ce qu'on a l'habitude de voir dans les outils de BI. Tout d'abord, l'ergonomie a été poussée dans ses derniers retranchements. L'outil propose des suggestions sur les données. Je suis en général réticent à ce genre d'approche, qui vise rarement juste, mais je dois avouer que j'ai été bluffé par les capacités de Trifacta Wrangler.

Il se trouve que le projet est issu d'un projet de recherche de Stanford University, mené par les docteurs Joe Hellerstein et Jeffrey Heer avec leur doctorant Sean Kandel, et précisément avec le but de monter un système de suggestions probant. Au final, le démonstrateur est extrêmement intuitif et permet d'atteindre des résultats très rapidement, car il propose, à chaque action, des suggestions qui montrent le résultat final si on applique cette proposition. Bref, l'utilisateur a un retour visuel immédiat sur les choix possibles.

Depuis ce démonstrateur, Joe, Jeff et Sean ont lancé la société Trifacta afin de continuer leur travail sur un outil que n'importe quel utilisateur sera capable d'utiliser. En plus de ceci, Trifacta Wrangler est pensé dès le début pour distribuer les calculs sur des machines séparées. On peut donc vraiment dire qu'il associe une grande simplicité avec une puissance impressionnante.

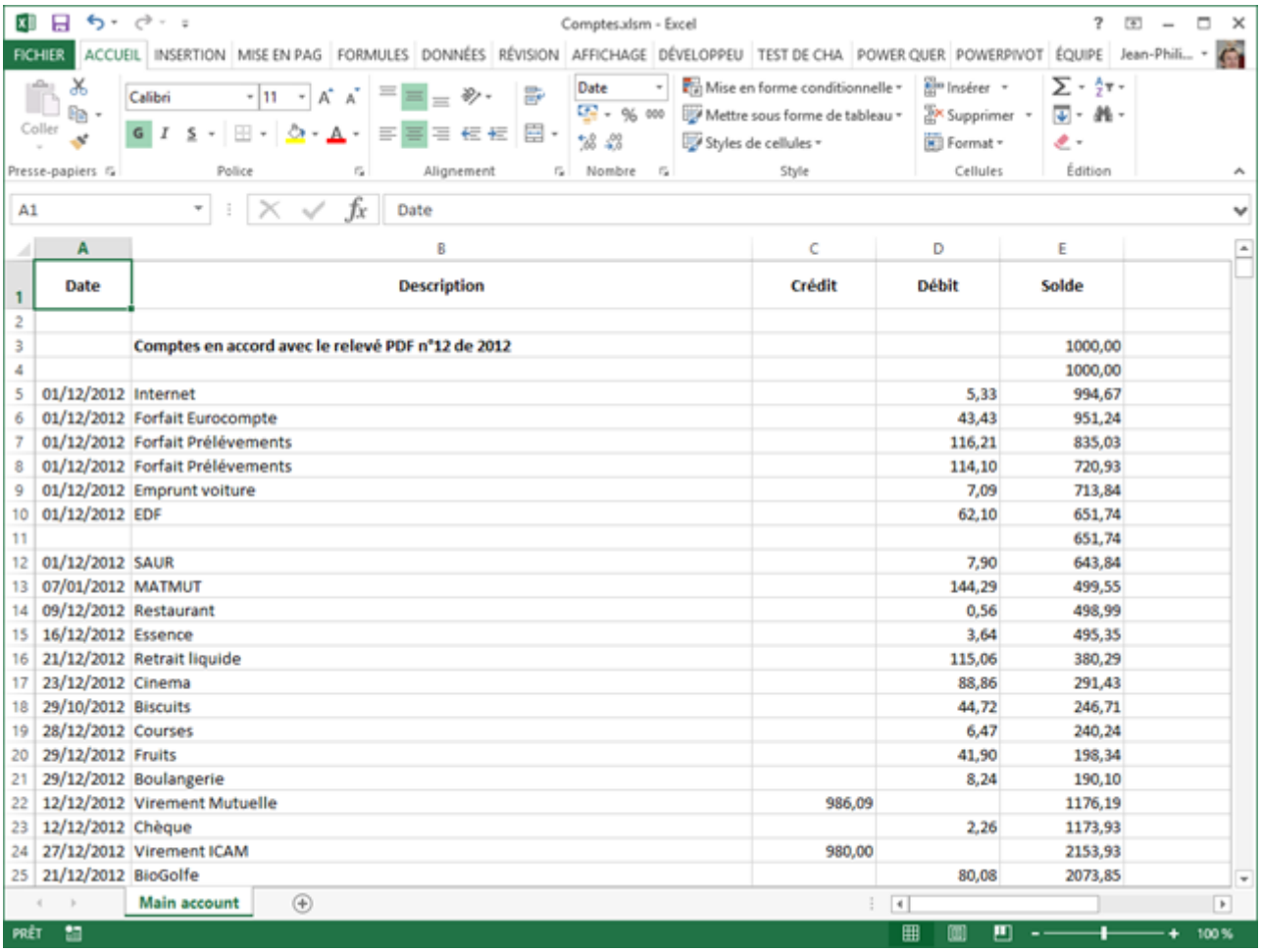
En plus du livre sur l'Open Data, je me suis retrouvé à utiliser Trifacta Wrangler pour débrouiller mes comptes bancaires. Il y a quelques semaines, je me suis rendu

compte que mes impôts avaient beaucoup augmenté, et je voulais savoir si, sur les dernières années, mes revenus avaient augmenté plus vite ou moins vite. J'avais une idée en gros, mais mes fichiers de comptes ne sont pas correctement catégorisés, et des raccourcis font qu'ils ne sont pas immédiatement analysables. Au final, je souhaitais savoir quelle était la part des impôts, des dépenses de loisirs, de celles qui sont incompressibles. Egalement, comme j'ai quelques revenus issus de location et de mes livres et vidéos, je voulais savoir comment ça participait à la balance au fur et à mesure du temps (ce n'est pas un scoop, mais au cas où vous ne le sauriez pas, non, la littérature informatique ne fait pas vivre).

A l'attaque de l'outil, maintenant !

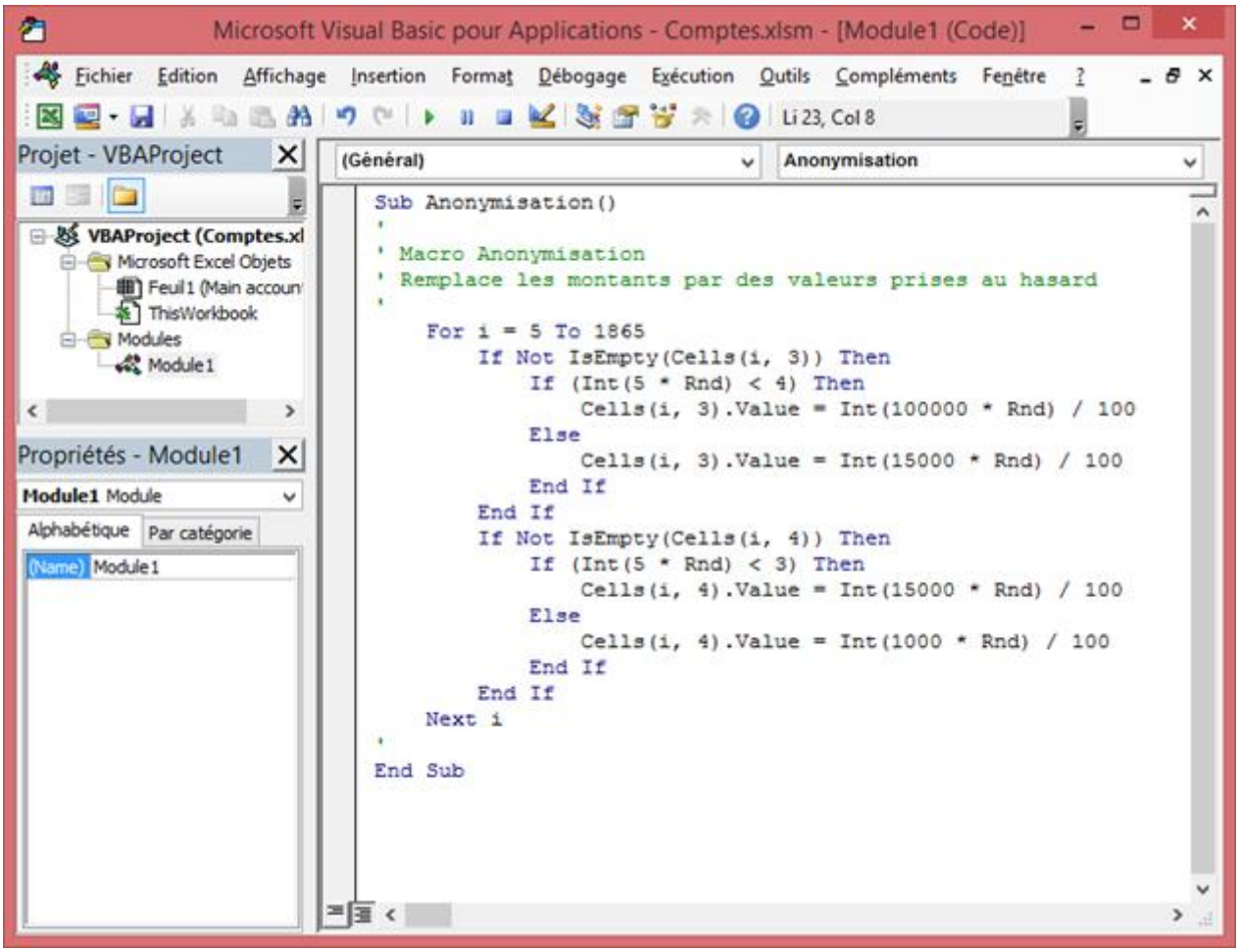
Préparation

Voici le fichier Excel qui me sert pour tenir mes comptes :

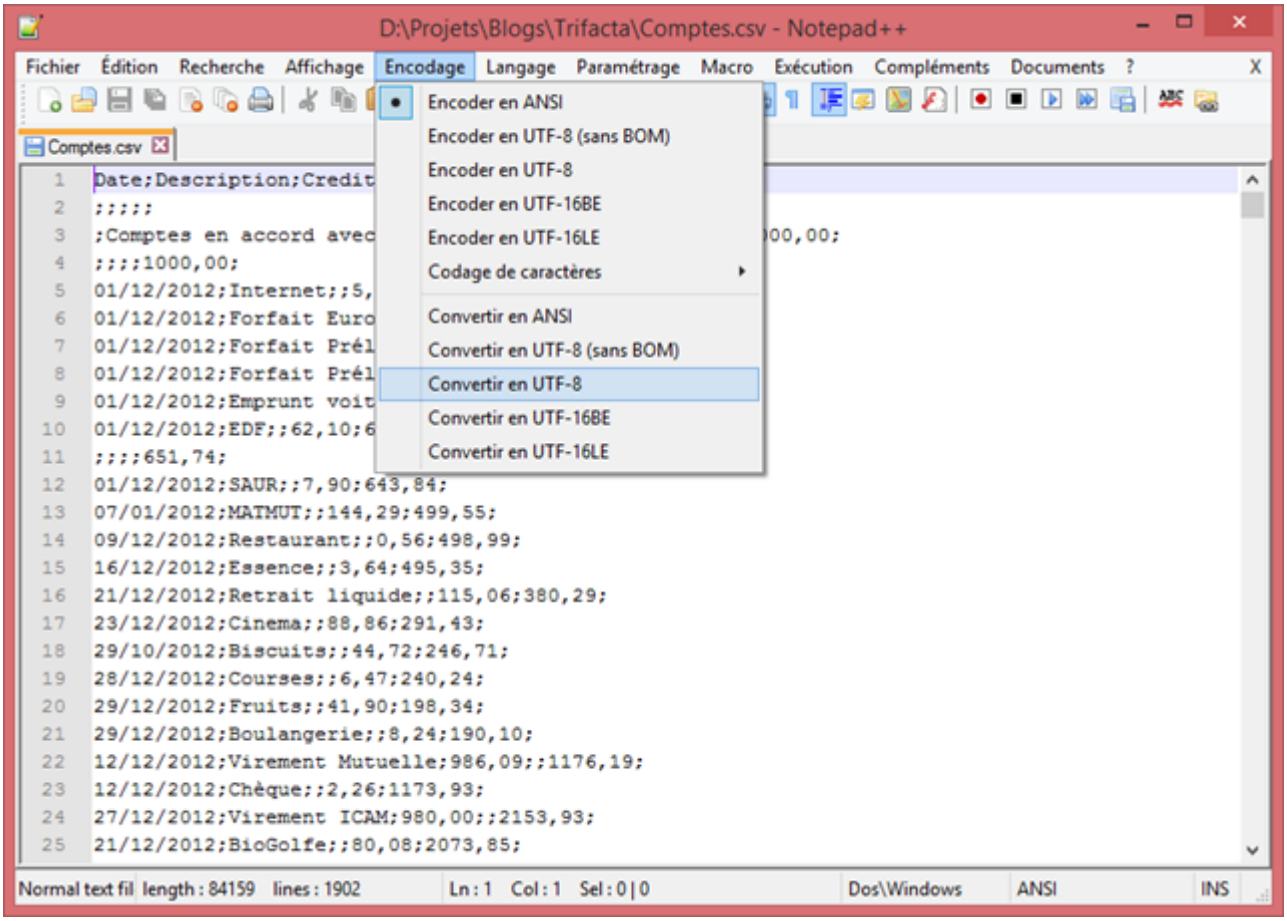


	A	B	C	D	E
1	Date	Description	Crédit	Débit	Solde
2					
3		Comptes en accord avec le relevé PDF n°12 de 2012			1000,00
4					1000,00
5	01/12/2012	Internet		5,33	994,67
6	01/12/2012	Forfait Eurocompte		43,43	951,24
7	01/12/2012	Forfait Prélévements		116,21	835,03
8	01/12/2012	Forfait Prélévements		114,10	720,93
9	01/12/2012	Emprunt voiture		7,09	713,84
10	01/12/2012	EDF		62,10	651,74
11					651,74
12	01/12/2012	SAUR		7,90	643,84
13	07/01/2012	MATMUT		144,29	499,55
14	09/12/2012	Restaurant		0,56	498,99
15	16/12/2012	Essence		3,64	495,35
16	21/12/2012	Retrait liquide		115,06	380,29
17	23/12/2012	Cinema		88,86	291,43
18	29/10/2012	Biscuits		44,72	246,71
19	28/12/2012	Courses		6,47	240,24
20	29/12/2012	Fruits		41,90	198,34
21	29/12/2012	Boulangerie		8,24	190,10
22	12/12/2012	Virement Mutuelle	986,09		1176,19
23	12/12/2012	Chèque		2,26	1173,93
24	27/12/2012	Virement ICAM	980,00		2153,93
25	21/12/2012	BioGolfe		80,08	2073,85

Vous vous doutez bien que je ne publie pas en ligne mes vrais revenus et dépenses, et que les chiffres que vous trouverez dans l'article sont passés par une moulinette d'anonymisation. Pour les plus curieux, voici la macro Excel qui a été utilisée :

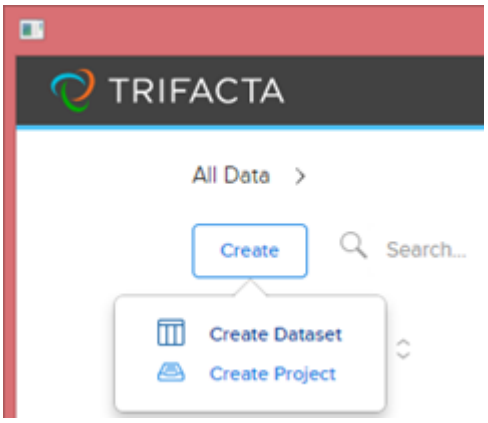


Afin d'éviter les problèmes d'encodage sur les caractères spéciaux (Excel enregistre en UTF-16 et Trifacta Wrangler lit uniquement de l'UTF-8 pour l'instant), la feuille du classeur est exportée en CSV (séparateur point-virgule) et le fichier est ensuite encodé en UTF-8 :

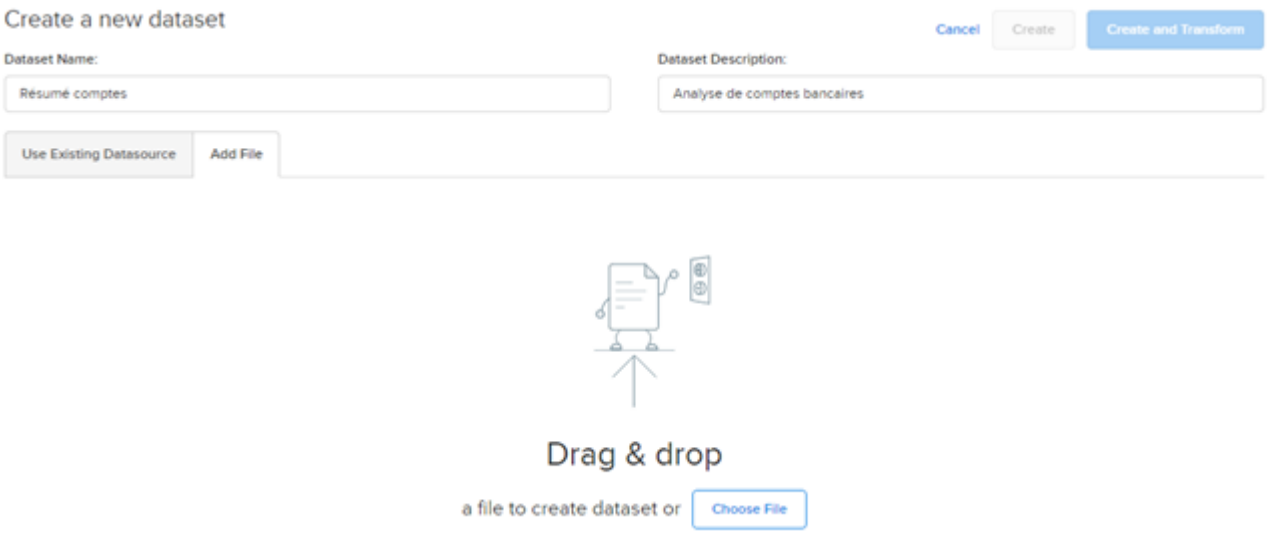


Au passage, j'en profite pour enlever les accents de la ligne d'entête, car ils ne sont pas affichés correctement sinon (pas de problème pour les données, toutefois, c'est juste l'entête).

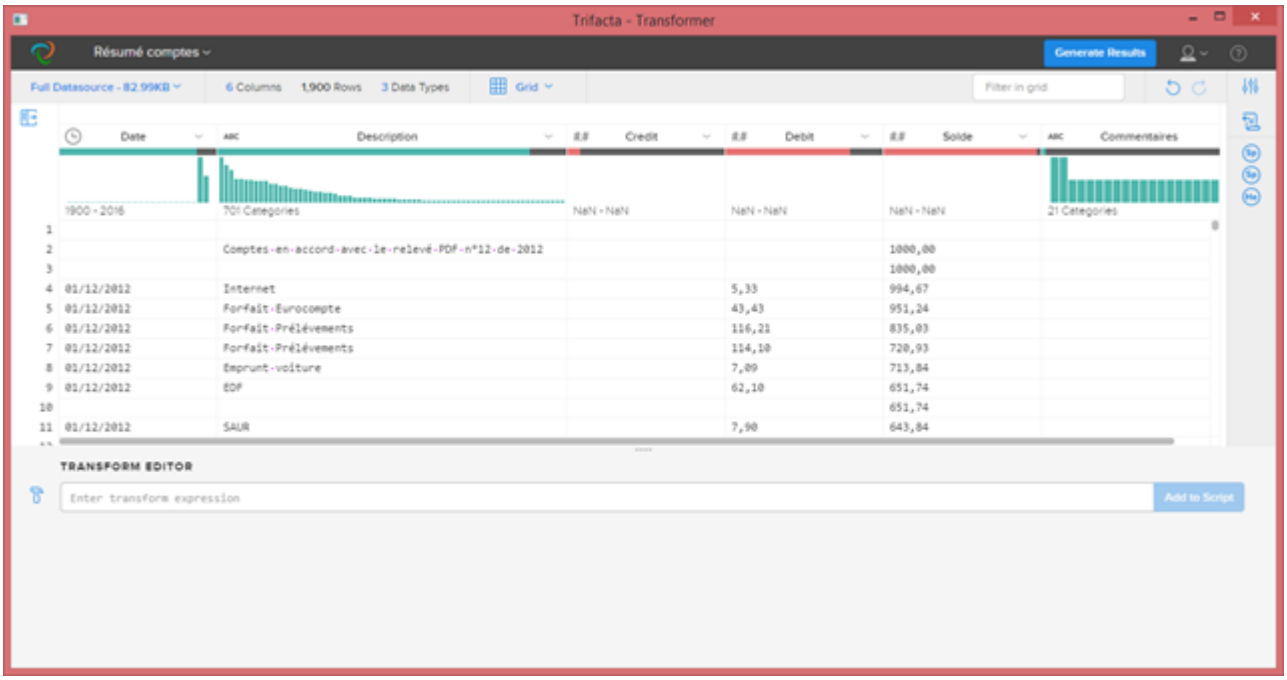
Le fichier étant prêt, on passe à l'outil Trifacta Wrangler, où on commence par créer un Dataset. Je passe sur l'installation de l'outil, qui n'appelle rien de particulier comme commentaire (à part qu'il est nécessaire de se connecter à chaque lancement de l'outil). Sur la première page, on voit la commande **Create Dataset** :



Ensuite, on renseigne le nom et la description, et on glisse-dépose le fichier CSV des comptes dans la zone du bas :



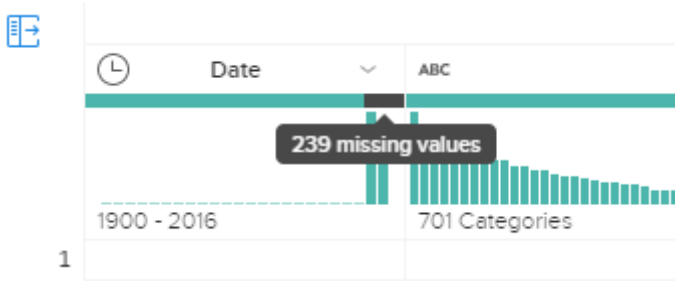
Un clic sur **Create and Transform**, et nous arrivons dans l'atelier de transformation du logiciel :



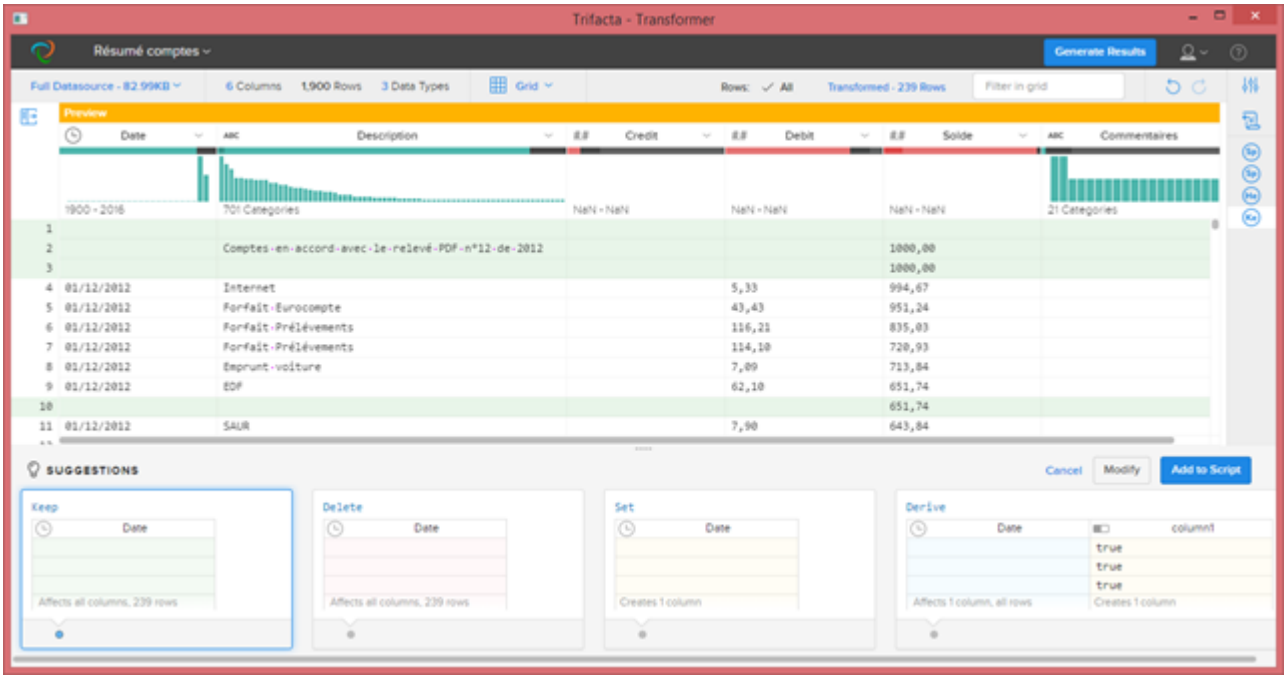
La phase préparatoire étant achevée, nous allons pouvoir passer à l'analyse et la structuration de données proprement dites.

Nettoyage des données non indispensables

La toute première chose à faire est de supprimer tout ce qui ne sera pas utile à notre analyse. Nous allons prendre les colonnes une par une, et dès cette première approche, on voit que Wrangler nous aide mieux que la plupart des outils concurrents. En plus de détecter le type des colonnes (ce qui est classique), il donne une indication sur les données manquantes, par exemple sur la première colonne :

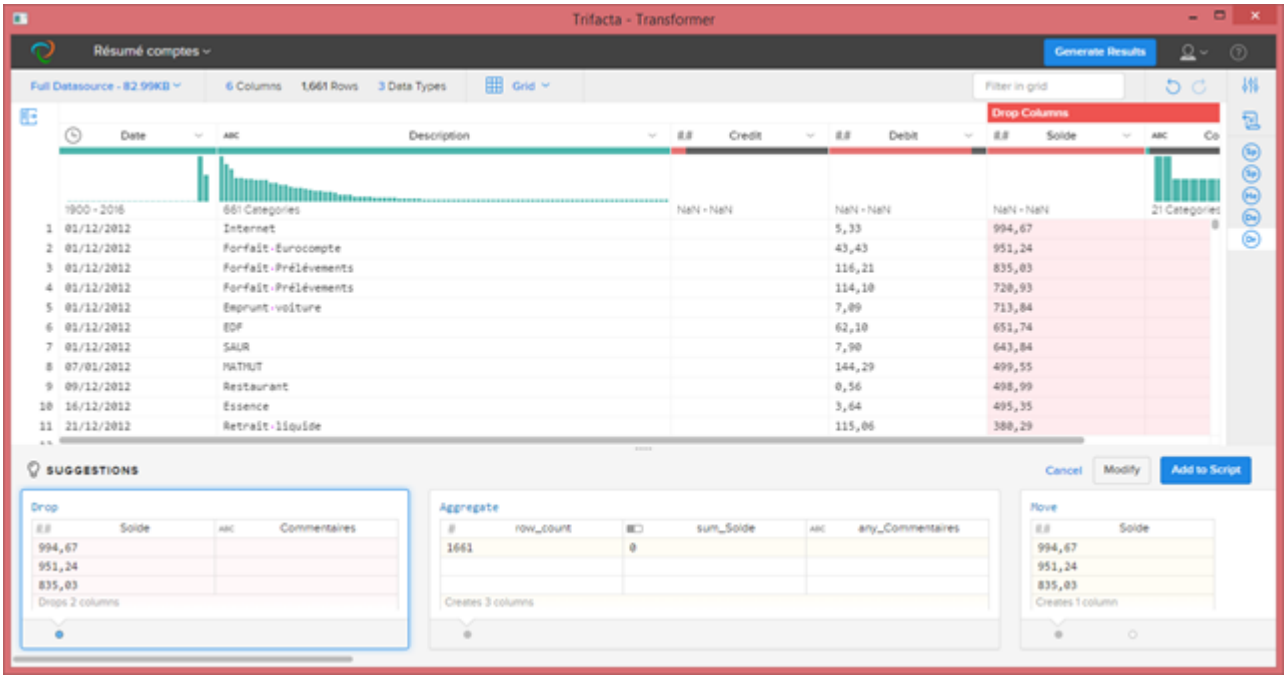


De la façon dont le fichier a été créé, toutes les lignes sans date peuvent être considérées comme inutiles, car un mouvement financier doit obligatoirement être inscrit dans le temps. Du coup, nous allons cliquer sur cette zone noire qui correspond aux données manquantes, et Wrangler va nous faire des suggestions de manipulation en lien :

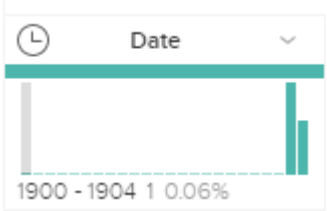


Ce qui nous intéresse dans le cas présent est le deuxième bloc de suggestion, qui correspond à la suppression des lignes qui sont sélectionnées dans le tableau (remarquez que la grille a automatiquement montré quelles lignes seront traitées). Nous cliquons donc sur le bloc **Delete**, puis sur **Add to Script**.

Tant que nous y sommes, les colonnes Solde et Commentaires ne nous seront pas utiles dans cette analyse, donc nous allons les supprimer également. Pour cela, nous les sélectionnons en cliquant simplement dans l'entête des deux colonnes (la sélection est incrémentale) :



Ce coup-ci, la première suggestion, pré-sélectionnée par défaut, à savoir **Drop** pour supprimer les colonnes, est la bonne, et nous pouvons cliquer directement sur le bouton d'acceptation **Add to Script**. La donnée est déjà plus propre, mais le spectre de répartition des dates en haut de la première colonne montre qu'il y a encore un petit quelque chose dont nous devons nous occuper :

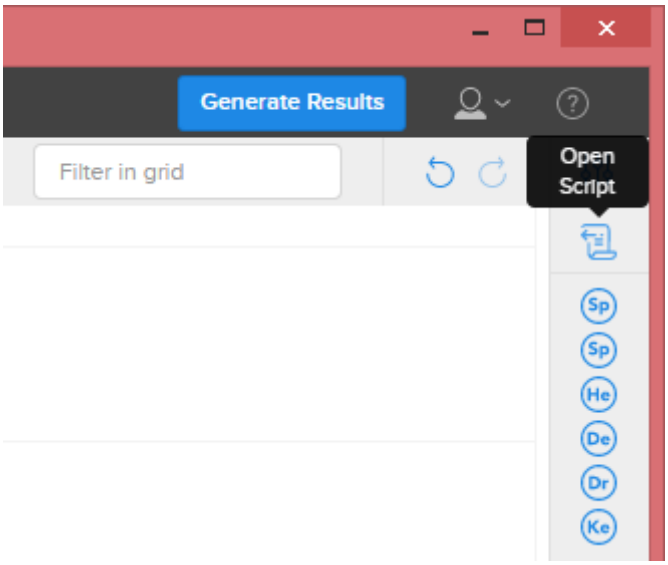


Apparemment, il y a une date incorrecte qui se situe entre 1900 et 1904. Pour retrouver précisément ce qui se passe, nous pourrions cliquer sur cette zone et sélectionner la suggestion **Keep**, pour ne garder que la donnée posant problème et l'analyser :

	Date	Description	Credit	Debit
	Jan 15 - Jan 15	1 Category	NaN - NaN	NaN - NaN
1	15/01/1900	Pizza·Del·Arte		101,76

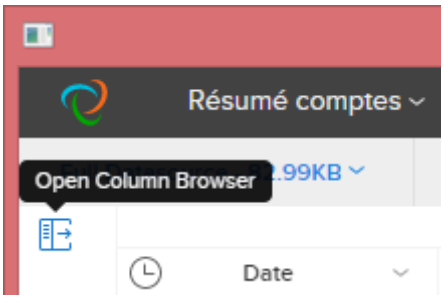
Il y a en effet une pizza dont la date a mal été entrée (le prix n'est pas bon non plus, mais c'est dû au brouillage des montants). Le logiciel ne peut pas la reporter comme posant problème, car il s'agit d'une vraie date sans erreur en soi. Par contre, nous allons voir un peu plus loin qu'il fait l'effort de signaler quelque chose de bizarre tout de même.

Pour l'instant, nous avons une action de filtre dont nous devons nous débarrasser, car elle n'était là que pour remonter la ligne posant problème. C'est l'occasion de montrer comment accéder à la liste des actions de Wrangler :

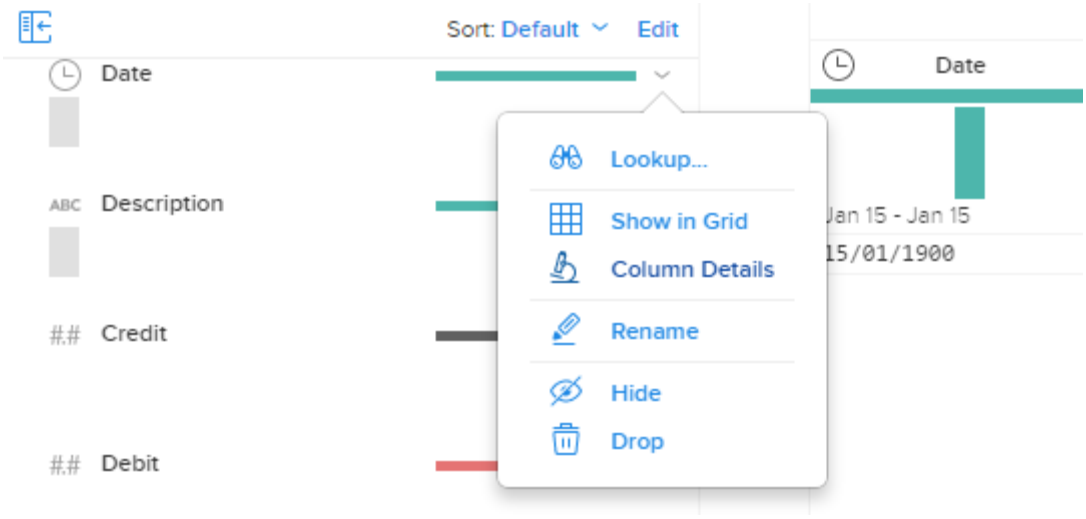


Le clic sur **Open Script** nous permet de voir ce qui se passe, et surtout de supprimer la dernière action de type **Keep**, qui n'est plus utile et ne fait pas réellement partie de l'analyse :

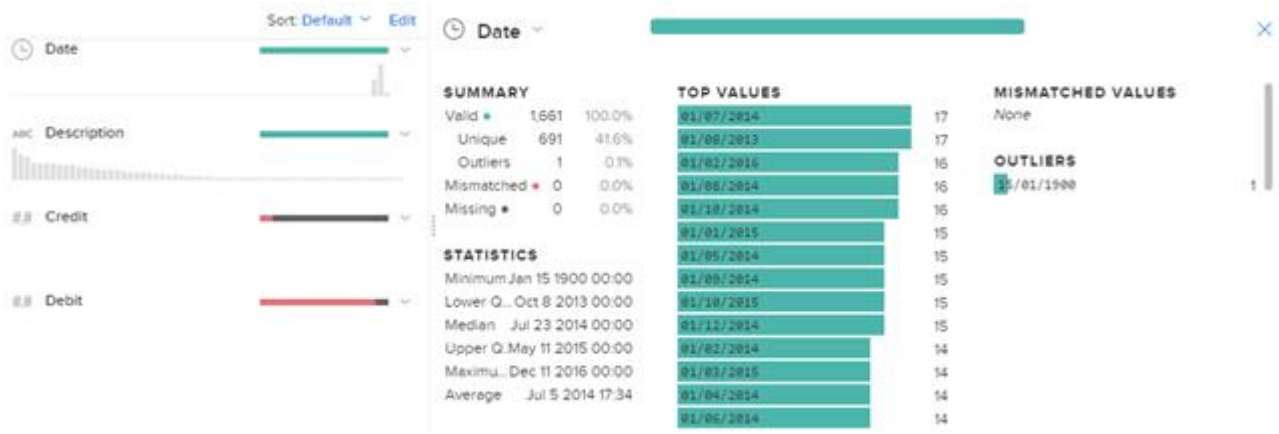
Comme dit précédemment, il était également possible d'utiliser les capacités d'analyse de valeurs de Wrangler pour voir le problème. Nous allons ouvrir le **Column Browser** :



En cliquant ensuite sur le menu **Column Details** pour la première colonne, nous allons accéder à plus de valeurs :

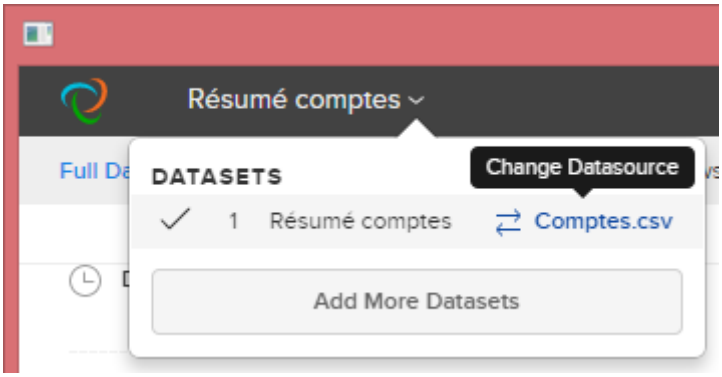


Comme nous pouvons le voir, la donnée "étrange mais sémantiquement correcte" a bien été détectée par Wrangler comme "Outlier" :

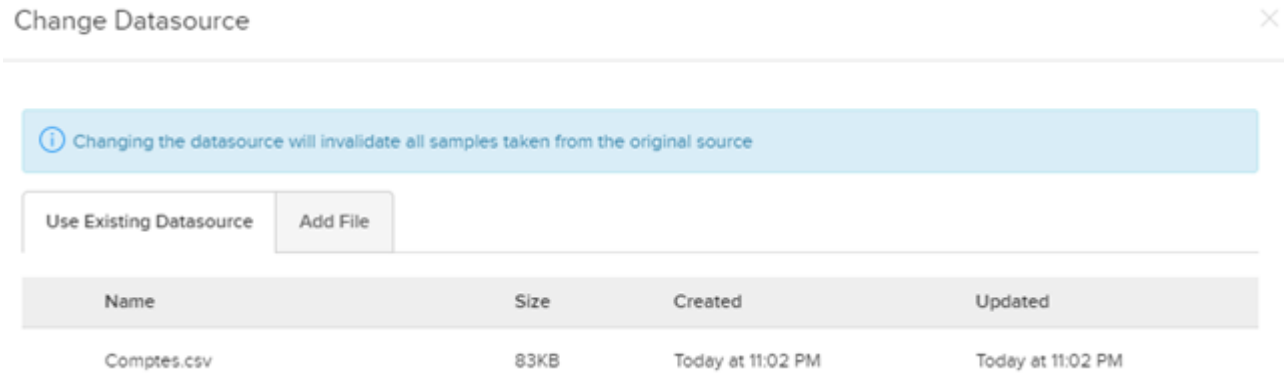


Du coup, la manipulation avec la commande Keep n'était pas indispensable, mais c'est intéressant de voir ce type de fonctionnement pour mieux comprendre le produit, et parce qu'il peut servir dans d'autres situations.

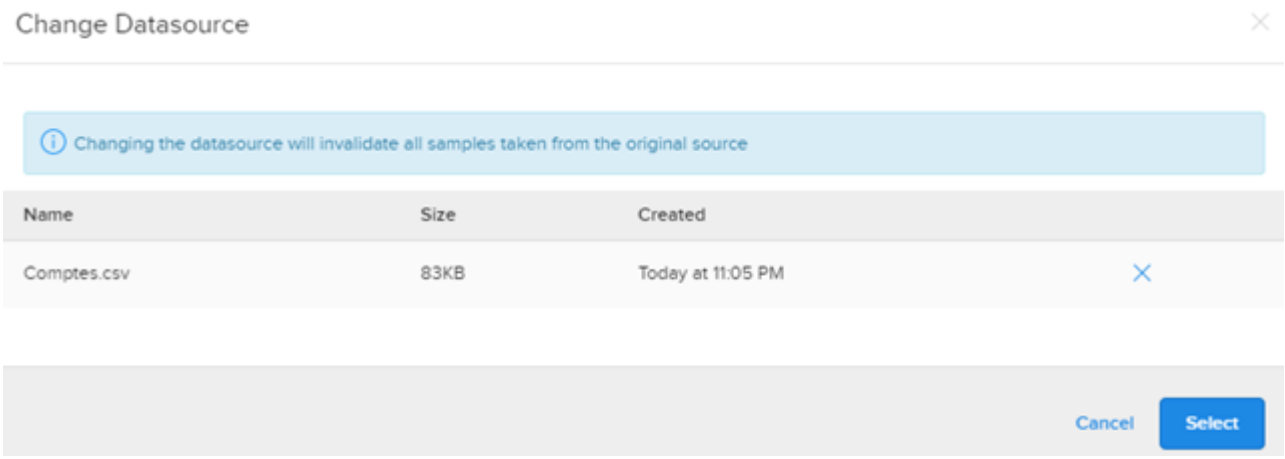
Une fois le fichier CSV source corrigé, il faut indiquer à Wrangler qu'il doit le recharger. Pour cela, le plus simple est d'utiliser la fonction permettant de changer de fichier, tout en repoussant le même :



Une fois dans la boîte de dialogue, nous utilisons l'onglet **Add File** :

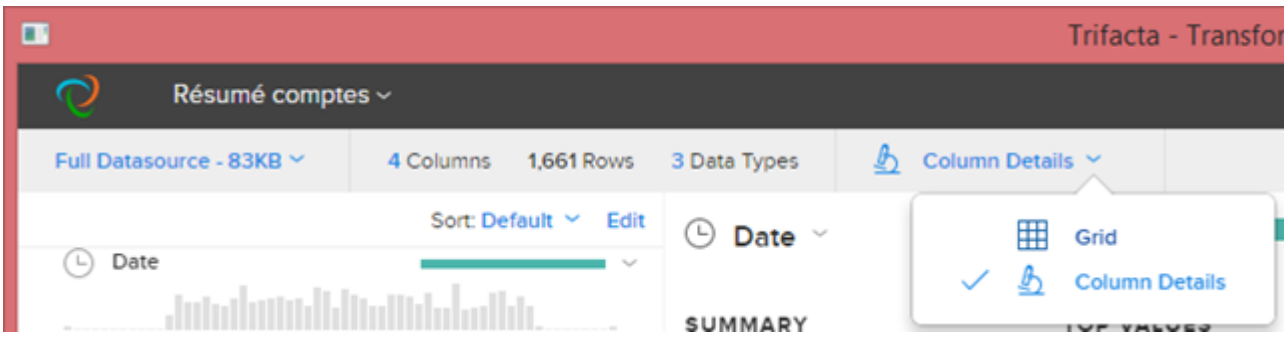


Ceci nous amène à la fenêtre suivante, sur laquelle nous pouvons glisser-déposer le fichier à nouveau et cliquer sur **Select** pour valider :

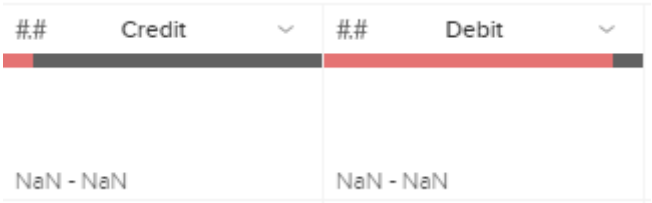


L'analyse de colonne paraît parfaite maintenant :

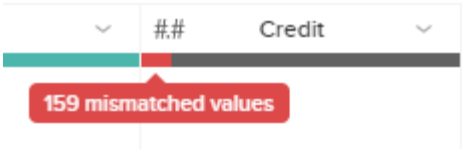
Pas de donnée "mismatched" (incorrecte) ni d' "outlier" (donnée en dehors du range standard) : nous sommes prêts à passer à la suite. Nous retournons donc sur la vue de type grille :



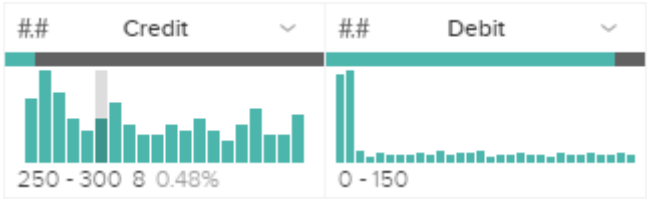
Et là, nous voyons le problème suivant : les colonnes numériques sont bien vues comme telles, et aucune portion n'est vue comme de la donnée valide (zone verte manquante) :



Le fait qu'il y ait de nombreuses valeurs manquantes (zone noire) dans la colonne Credit n'est pas illogique, car les crédits et les débits étant dans des colonnes différentes et la plupart des opérations sont des dépenses. Mais ce qui est gênant est que tout le reste est de la donnée incorrecte :



En fait, ce qui se passe est que Wrangler, ne voyant que la régionalisation américaine pour l'instant, ne supporte pas que les données utilisent des virgules comme séparateur décimal. Nous allons donc refaire la même manipulation que tout à l'heure après avoir remplacé toutes les virgules par des points dans un éditeur de texte, et une fois le fichier rechargé, cela va effectivement beaucoup mieux :



Catégorisation des données

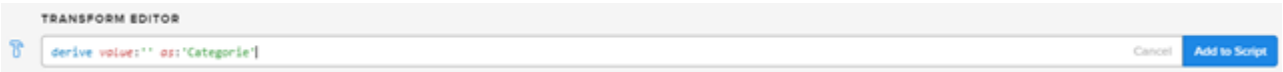
Nous ne sommes toutefois pas au bout de nos peines et comme souvent, la partie "nettoyage et structuration de la donnée" prend beaucoup plus de temps que l'analyse des résultats elle-même. Heureusement que nous avons le genre d'outils comme Wrangler pour rendre cela un peu plus ludique .

L'étape suivante d'analyse est de déterminer la catégorie des dépenses et des recettes, de façon à pouvoir obtenir un point de vue agrégé. Pour cela, nous allons créer de

toutes pièces une colonne qui contiendra cette information, et ajouter pas à pas des valeurs en fonction des mots-clés et des patterns trouvés dans la description des mouvements bancaires.

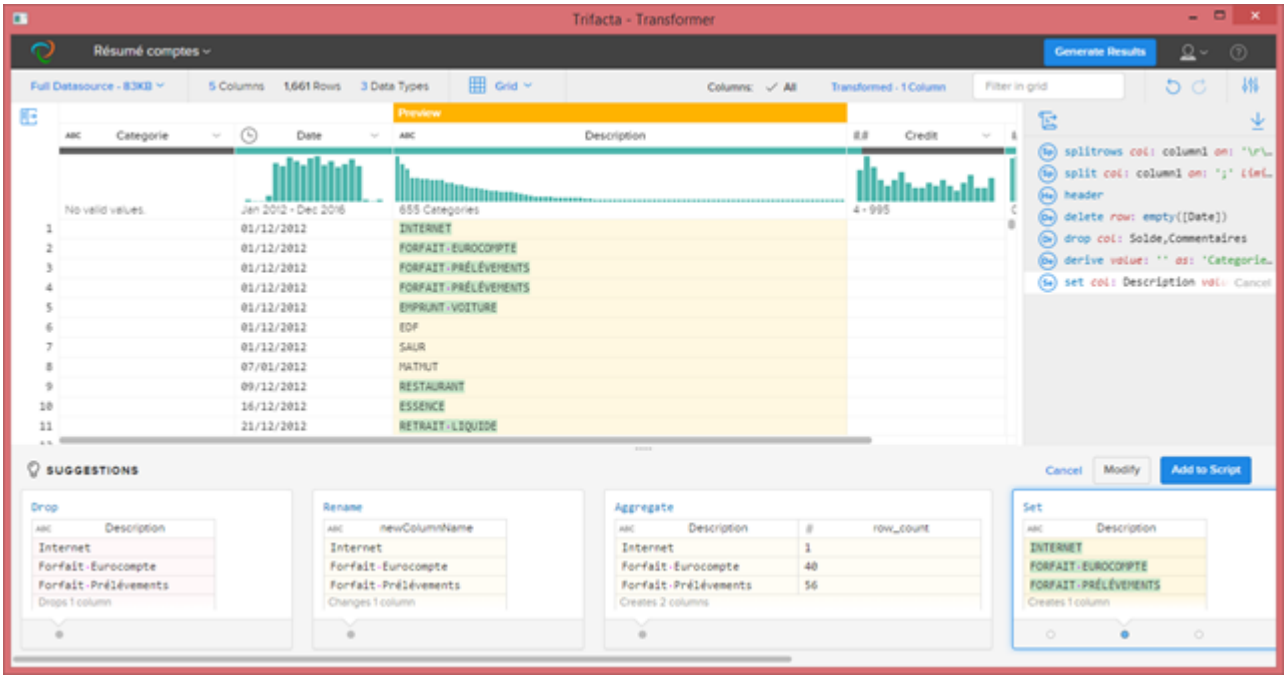
Dans un premier temps, il nous faut créer la colonne à vide. C'est la fonction **derive** que nous allons mettre en œuvre, et cette fois, nous n'allons pas utiliser le moteur de suggestion mais écrire directement la commande dans le **Transform Editor**, en bas de l'interface :

L'assistance à la saisie nous montre que le premier paramètre sera **value**. Dans notre cas, nous utiliserons une chaîne vide, de façon à garder la colonne vide par défaut. Ensuite, nous utiliserons le paramètre **as** pour spécifier le nom de la colonne à créer. Enfin, nous cliquerons sur **Add to Script** pour valider :



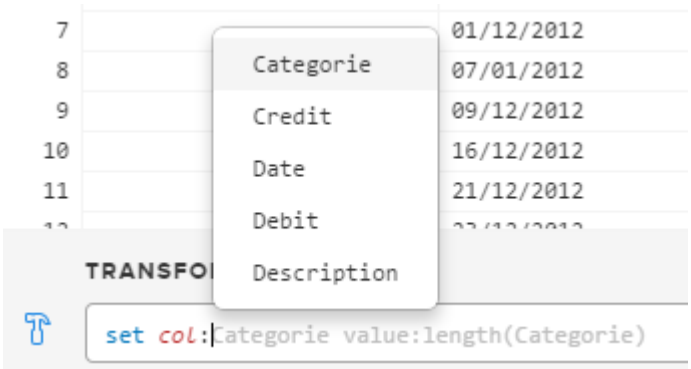
Attention à ne pas utiliser d'accent dans les noms de colonnes, Wrangler ne le supporte pour l'instant pas. On peut imaginer que cela va venir très vite, toutefois, puisqu'aux dernières nouvelles, ils ouvrent un bureau en France…

Avant de rentrer dans la classification proprement dite, comme nous allons utiliser les mots de la colonne Description et que certains sont en majuscules et d'autres en minuscules, nous allons d'abord tout passer en majuscules, pour simplifier la recherche. Pour cela, nous sélectionnons la colonne Description, et les suggestions vont nous aider :



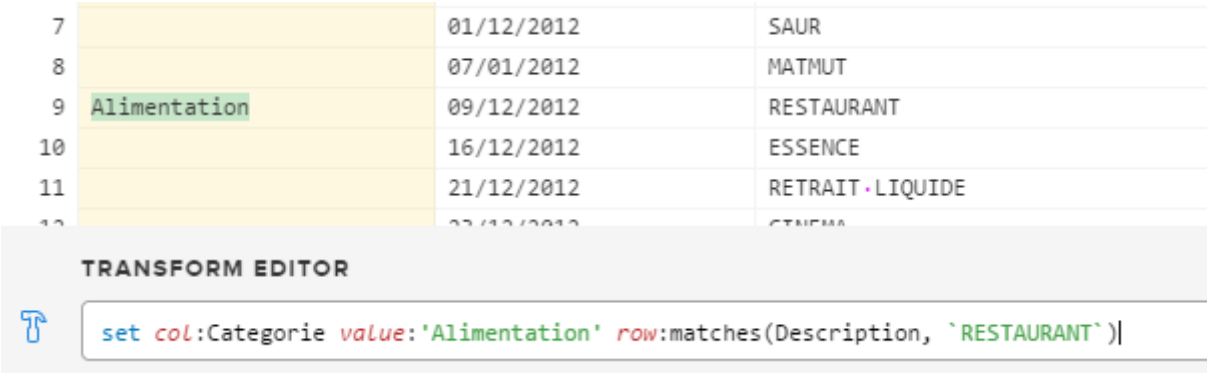
La deuxième option (les options sont matérialisées par les points en bas des blocs) sur le bloc **Set** permet de tout passer en majuscules, donc nous l'acceptons.

L'idée ensuite est d'utiliser le contenu de cette colonne pour déterminer la catégorie des lignes de compte. Pour cela, nous utiliserons à nouveau la commande s**et**, mais ce coup-ci en lui passant une valeur fixe non vide mais aussi une condition pour dire sur quelle ligne cette valeur s'applique. Une catégorie assez évidente est par exemple Alimentation, qui sera affectée chaque fois qu'on tombe sur un mot comme RESTAURANT, BOULANGERIE, BOUCHERIE ou CARREFOUR. Comme plus haut, nous allons entrer la formule directement dans le **Transform Editor**. La première information à passer, à part bien sûr le nom de l'opération lui-même, est la colonne sur laquelle cette dernière va s'appliquer.



L'option suivante est la valeur à utiliser. Dans cet exemple, elle est fixe :

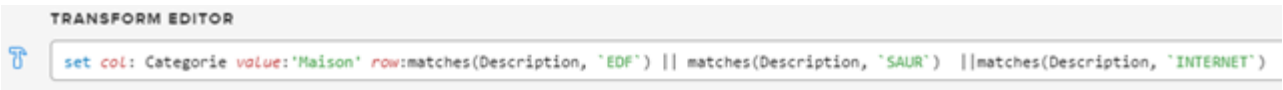
Enfin, il nous faut créer une condition sur le contenu de la description :



La fonction **matches** permet de filtrer les lignes pour lesquelles il y a correspondance entre le contenu de la colonne Description et l'expression régulière passée en second paramètre. Attention, les séparateurs sont des apostrophes inversées, et pas les apostrophes standards utilisées pour entourer les chaînes, comme dans le paramètre **value**. On voit dans la capture que Wrangler a tout de suite montré le résultat et trouvé en particulier une ligne qui correspond.

Il serait bien sûr possible de dupliquer l'opération pour tous les patterns recherchés, mais vu qu'il sera déjà nécessaire de créer une opération par catégorie détectée, il paraît plus propre de rechercher tous les patterns dans la même opération. Pour cela, nous allons utiliser l'opérateur booléen de Wrangler, à savoir un double pipe :

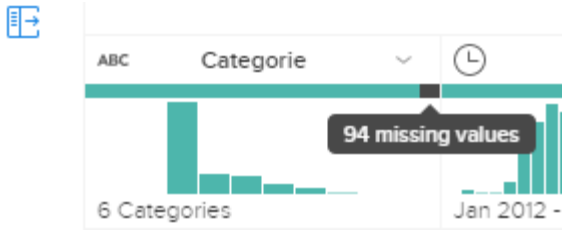
Cliquer sur **Add to Script** exécute la commande en l'ajoutant dans la liste, et nous allons créer autant de ces commandes que nécessaire pour toutes les catégories à retrouver. Par exemple, pour les mots EDF et SAUR (un fournisseur d'eau, en Bretagne), les lignes correspondront à une catégorie Maison, comme par exemple pour la connexion Internet :



Les mots FORFAIT et VIREMENT correspondront à la catégorie Banque :

```
TRANSFORM EDITOR
set col: Categorie value:'Banque' row:matches(Description, `FORFAIT`) || matches(Description, `VIREMENT`)
```

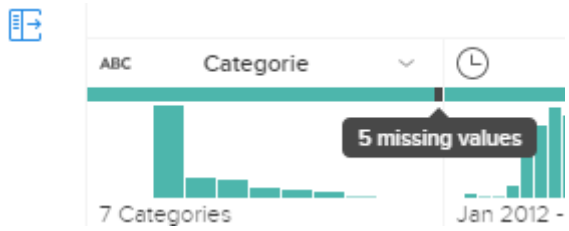
VOITURE et ESSENCE seront les patterns pour le transport, etc. Après quelques autres manipulations de ce type, le spectre du haut de la colonne Catégorie devrait montrer qu'il n'y a plus beaucoup de lignes sans catégorie associée :



La proportion est suffisamment faible pour que nous prenions le reste et la placions dans une catégorie "autres". Par contre, il est nécessaire de bien faire la différence entre les "autres recettes" et les "autres dépenses". Nous utiliserons le fait que seule une des deux colonnes Credit ou Debit peut être remplie à la fois pour déterminer le type de la ligne. La première opération est la suivante :

```
TRANSFORM EDITOR
set col: Categorie value:'Autres dépenses' row:empty([Categorie]) && empty([Credit])
```

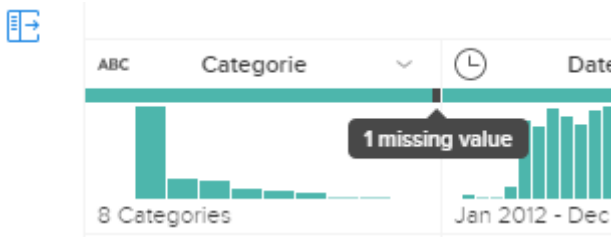
Ceci nous permet de réduire sensiblement le nombre de lignes sans catégorie :



Nous créons ensuite l'opération inverse :

```
TRANSFORM EDITOR
set col: Categorie value: 'Autres recettes' row: empty([Categorie]) && empty([Debit])
```

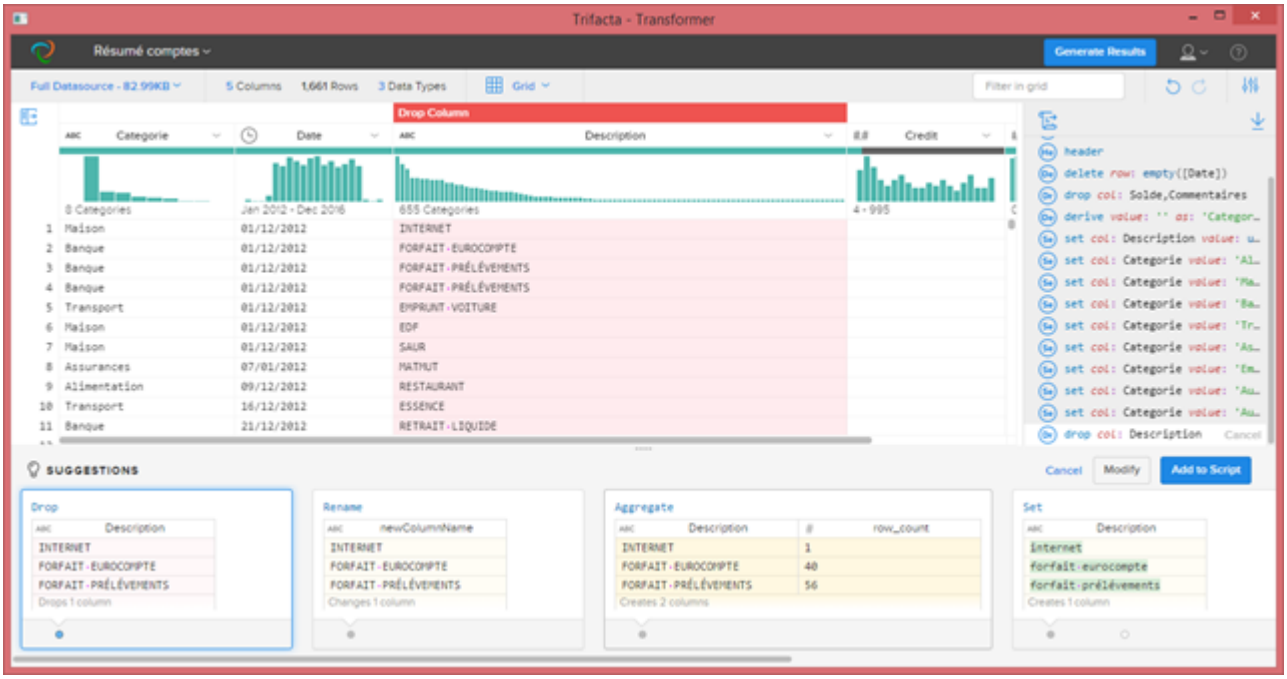
Logiquement, il ne devrait plus y avoir la moindre lignes sans catégorie, mais ce n'est pas le cas, et Wrangler nous signale le problème :



Puisque la commande **Column Details** ne montre pas la valeur exacte, nous allons utiliser la même astuce que montrée plus haut et ne garder que la ligne fautive en créant une opération temporaire de type **Keep** sur cette sélection. Le résultat est comme suit :

Encore une fois, il s'agit apparemment d'une entrée incorrecte car il ne peut pas normalement y avoir une entrée et une sortie à la fois sur une ligne financière. Du coup, le plus simple est de corriger le fichier CSV et de recharger. L'explication pratique de cette exception est que le paiement de ces tickets de cinéma n'est jamais passé sur la carte bleue, et que la banque ne m'a jamais débité. Du coup, en attendant de voir si le paiement arrivait plus tard, j'avais mis le même montant en crédit et en débit (ce qui n'est plus le cas ici à cause de l'obfuscation des montants).

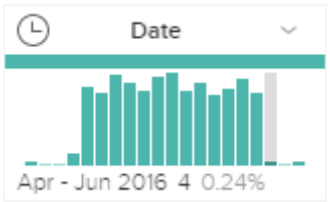
Les catégories sont maintenant correctes, ce qui fait que nous pouvons nous débarrasser de la colonne Description, qui ne sert plus (après bien sûr avoir annulé la commande **Keep** précédente qui n'était là que pour le diagnostic). Ceci sera la dernière opération avant l'analyse à proprement parler de nos données :



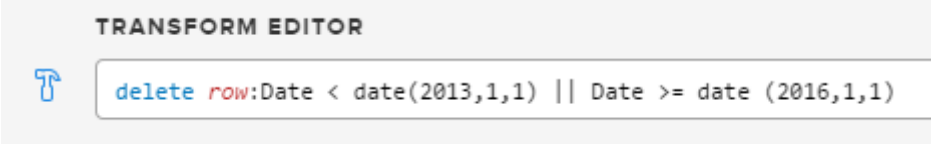
Grouper les valeurs

Les catégories sont maintenant révélées et la donnée nettoyée. Comme ce que nous voulons est une analyse des montants des différentes catégories sur des périodes agrégées de temps, nous allons par la suite agréger les données selon ces deux axes.

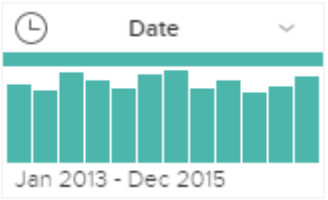
Nous commençons par opérer sur l'axe chronologique. Pour cela, nous commençons par supprimer les lignes correspondant aux années incomplètes en utilisant la commande **delete**, de façon à avoir des résultats qui puissent se comparer aisément les uns aux autres. Le spectre de répartition des valeurs en haut de la colonne Date montre qu'il y a trois années complètes, et que 2012 et 2016 comptent moins de valeurs :



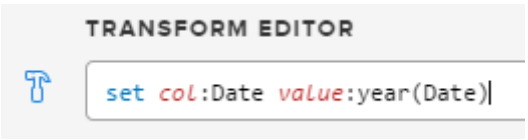
L'opération ci-dessous va nous permettre de nous débarrasser des lignes inutiles pour notre analyse (attention à la différence entre la colonne Date avec une majuscule au début et la fonction **date** qui permet de créer une date en grammaire Wrangler) :



Bien sûr, cette manipulation met à jour le spectre sur un aplat plus cohérent avec la répartition logique des mouvements financiers, c'est-à-dire assez homogène :

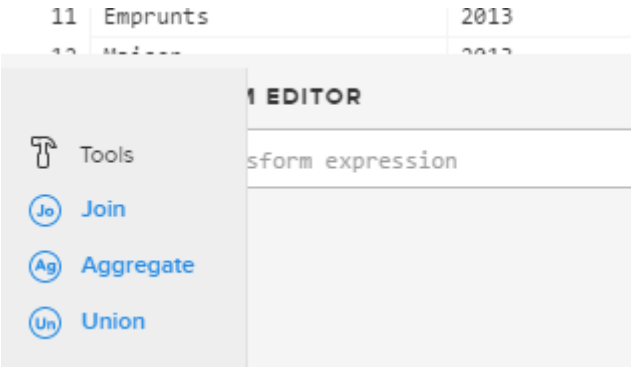


Dans notre exemple, nous nous intéressons à un agrégat des valeurs par année, donc nous allons ajouter une opération pour garder uniquement cette partie de la date :

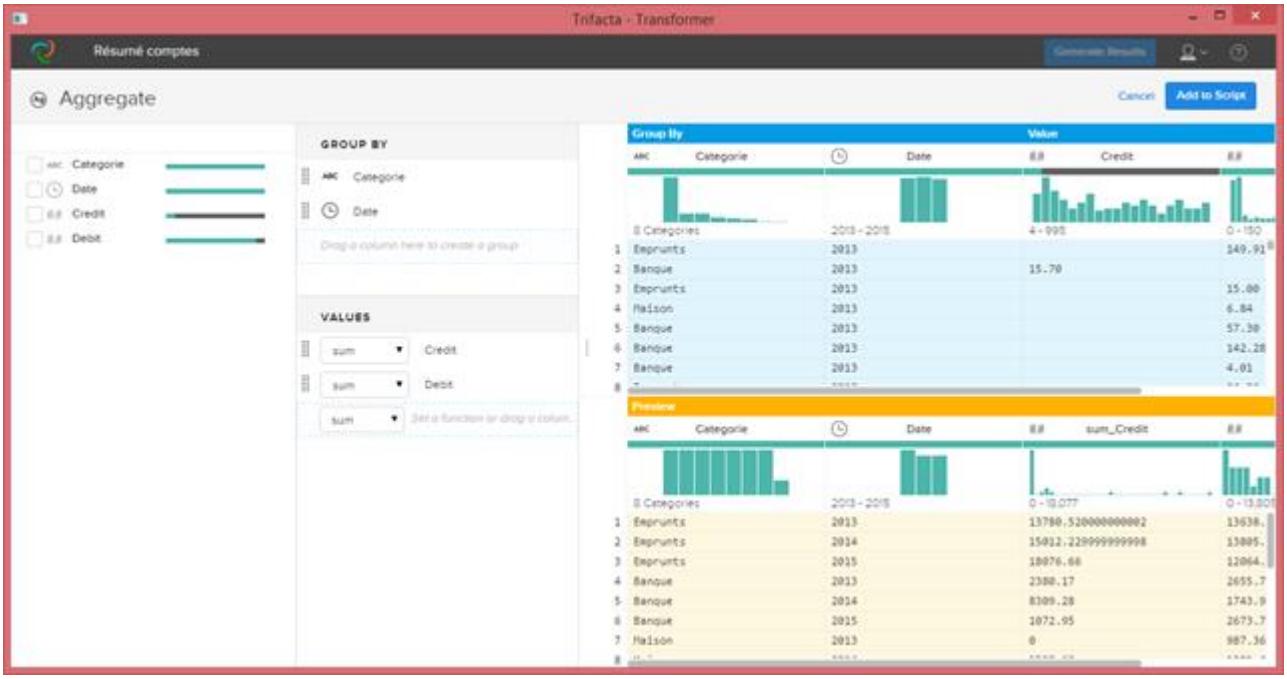


Notez que nous aurions bien sûr pu inverser les deux dernières opérations, et commencer par retrouver les années pour supprimer ensuite celles que nous ne souhaitions pas, ce qui aurait rendu la première opération un peu plus simple.

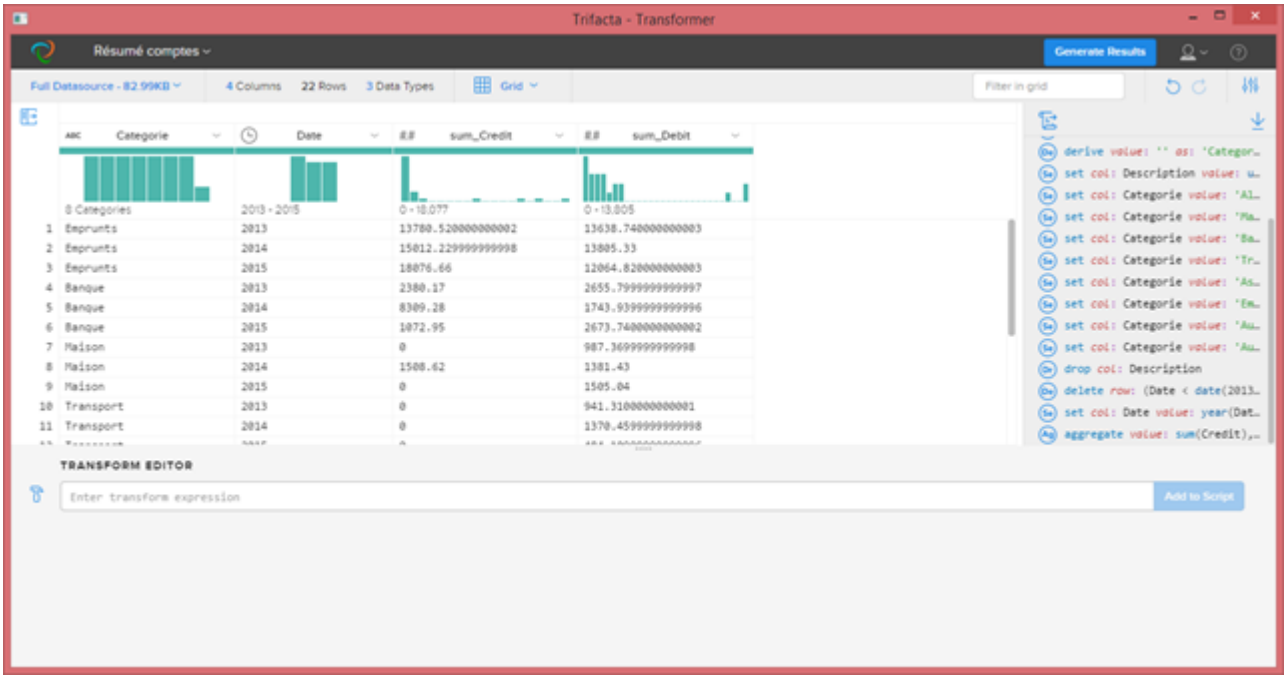
Arrivé à ce point, nous devrions être prêt pour réaliser l'agrégation. Nous ouvrons la Tools section en cliquant sur l'icône de marteau à gauche de la zone de saisie des commandes, et sélectionnons la commande **Aggregate** :



En utilisant les poignées à gauche des quatre lignes disponibles sur la gauche de l'écran, nous glissons et déposons les colonnes de façon à obtenir l'affichage ci-dessous :

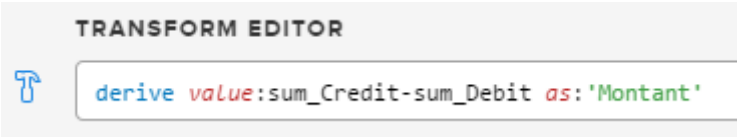


Comme nous avions pris de remplacer toutes les virgules par des points (Wrangler ne se sert pas de la régionalisation système du séparateur décimal), les résultats prévus sont bien affichés et nous pouvons donc cliquer sur **Add to Script** en haut de l'interface, ce qui nous amène aux résultats suivants :

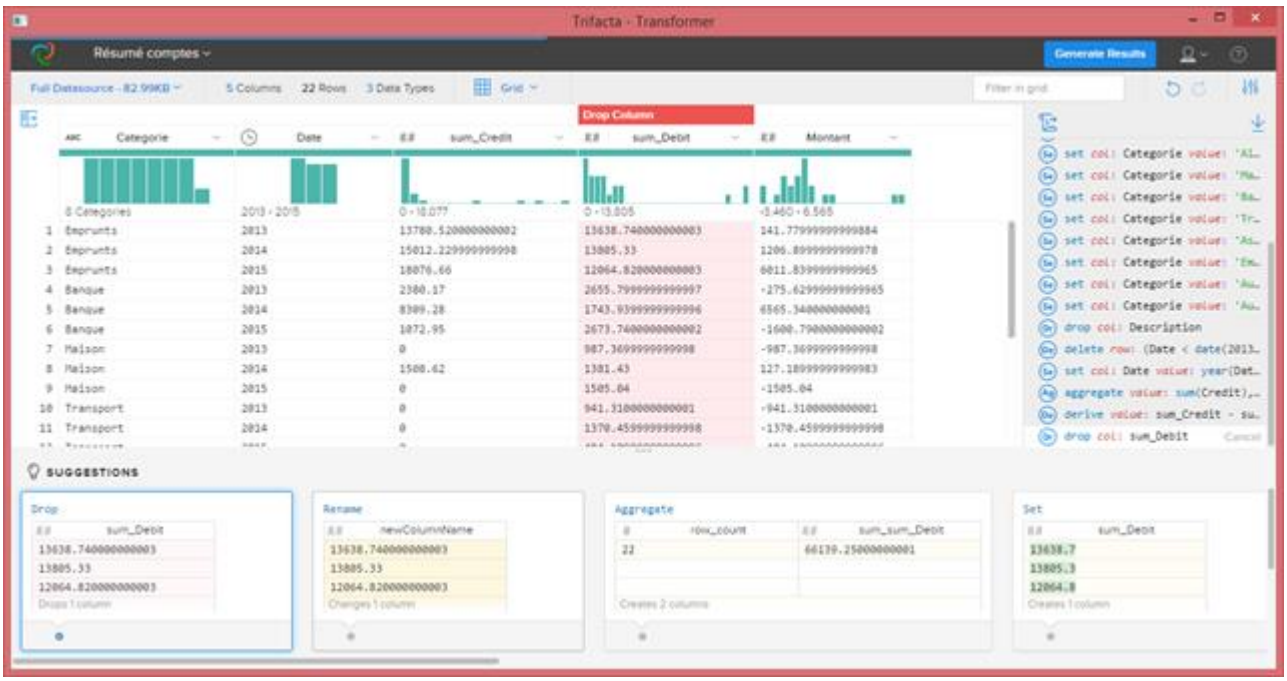


Enfin, de façon à rendre plus simple l'utilisation des montants, nous créons une colonne qui va reprendre les valeurs en crédit et en débit, mais avec un signe négatif

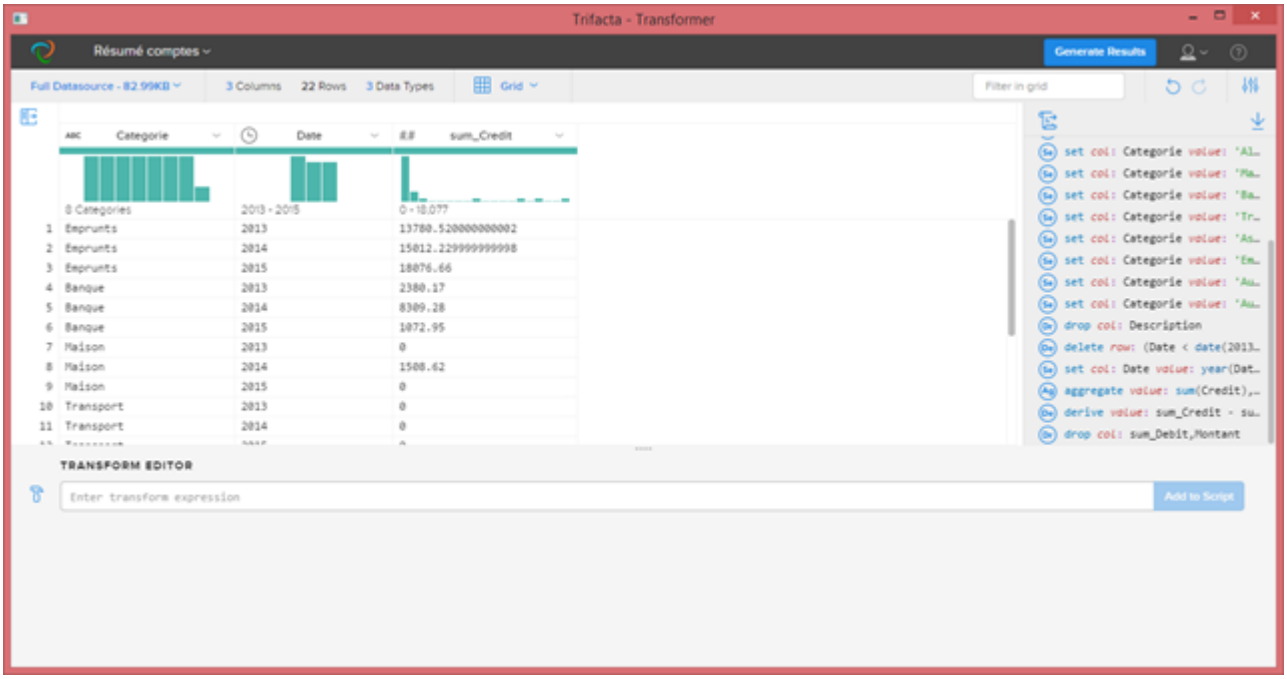
pour ce dernier. Le plus simple est de réaliser une simple soustraction dans une commande **derive** :



Les deux colonnes originales peuvent alors elles aussi être supprimées (il est important de faire du ménage au fur et à mesure, pour bien s'y retrouver dans la manipulation) :

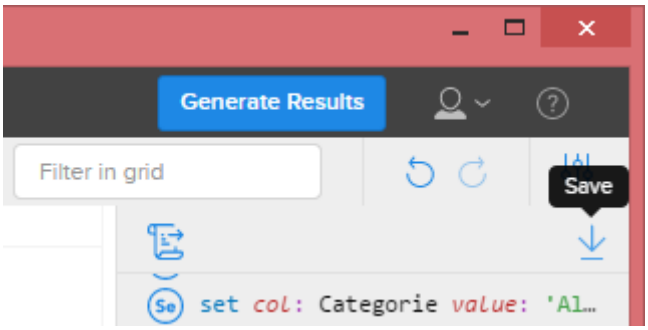


Les résultats sont bien ceux attendus, à part bien sûr que les montants sont de fausses valeurs dans la capture :

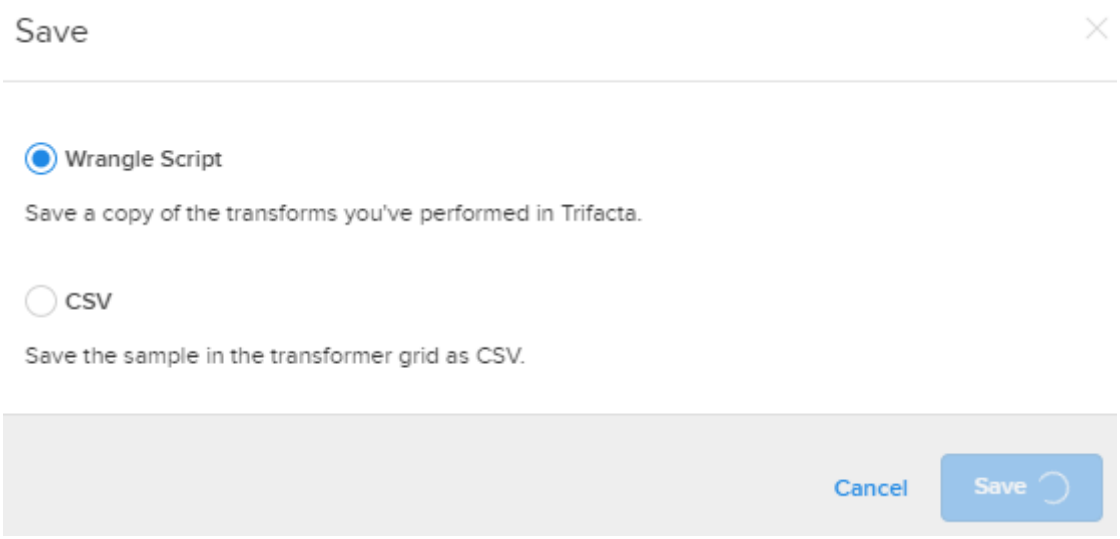


Export des résultats

Wrangler crée un script de réalisation des manipulations, pour exécution postérieure, voire distribuée sur des machines en cluster pour les très gros volumes. Ce script peut être visionné en utilisant la commande **Save** :



Il faut ensuite choisir l'option **Wrangle Script**, et cliquer sur **Save** :



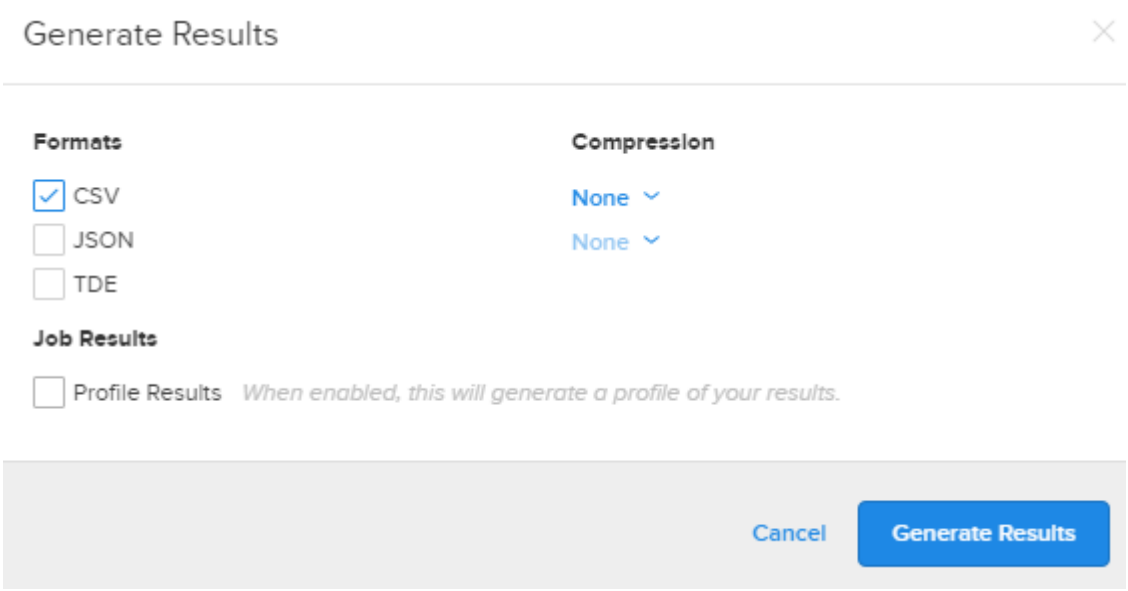
En résumé, le script est le suivant (toutes les catégories n'ont pas été montrées, pour simplifier le contenu de l'exemple par rapport à ma vraie analyse de comptes) :

View Script

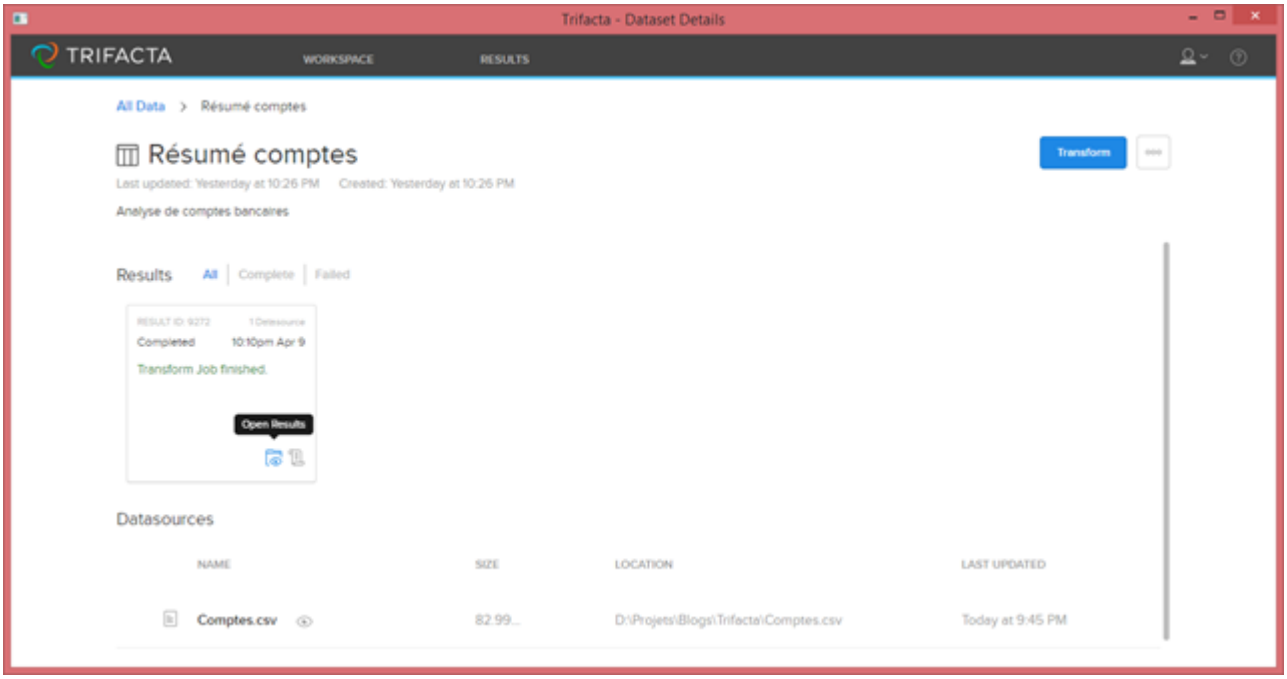
```
splitrows col: column1 on: '\r\n' quote: '\"'
split col: column1 on: ';' limit: 5 quote: '\"'
header
delete row: empty([Date])
drop col: Solde,Commentaires
derive value: '' as: 'Categorie'
set col: Description value: uppercase(Description)
set col: Categorie value: 'Alimentation' row: matches([Description], `RESTAURANT`)
set col: Categorie value: 'Maison' row: matches([Description], `EDF`) || (matches([
set col: Categorie value: 'Banque' row: matches([Description], `FORFAIT`) || (match
set col: Categorie value: 'Transport' row: matches([Description], `VOITURE`) || mat
set col: Categorie value: 'Assurances' row: matches([Description], `MATMUT`)
set col: Categorie value: 'Emprunts' row: (Categorie == '') && matches([Description
set col: Categorie value: 'Autres dépenses' row: empty([Categorie]) && empty([Credi
set col: Categorie value: 'Autres recettes' row: empty([Categorie]) && empty([Debit
drop col: Description
delete row: (Date < date(2013, 1, 1)) || (Date >= date(2016, 1, 1))
set col: Date value: year(Date)
aggregate value: sum(Credit),sum(Debit) group: Categorie,Date
derive value: sum_Credit - sum_Debit as: 'Montant'
drop col: sum_Debit,Montant
```

La dernière étape consiste à exporter les résultats, ou plus précisément lancer l'exécution avec export, car ce qui se passe dans l'éditeur est que Wrangler travaille sur un échantillon des données. Il se trouve dans notre cas que les données étaient suffisamment faibles pour que Wrangler charge tout et analyse tout en temps réel,

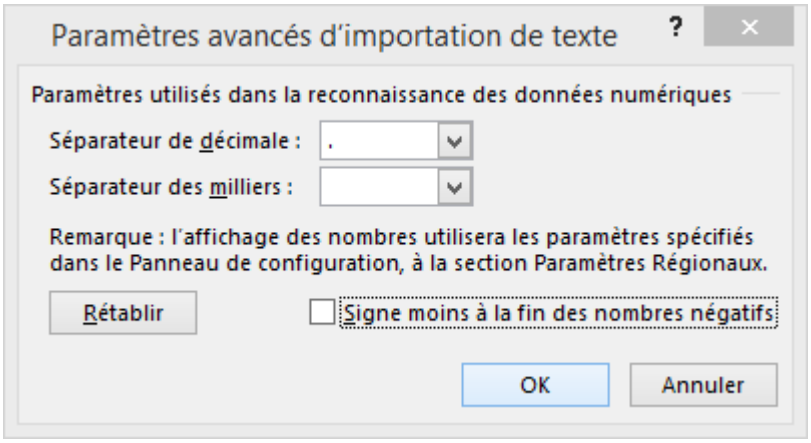
donc nous ne nous en sommes pas rendus compte. Nous cliquons donc sur **Generate Results** en haut à droite de l'interface, et utilisons les options suivantes :



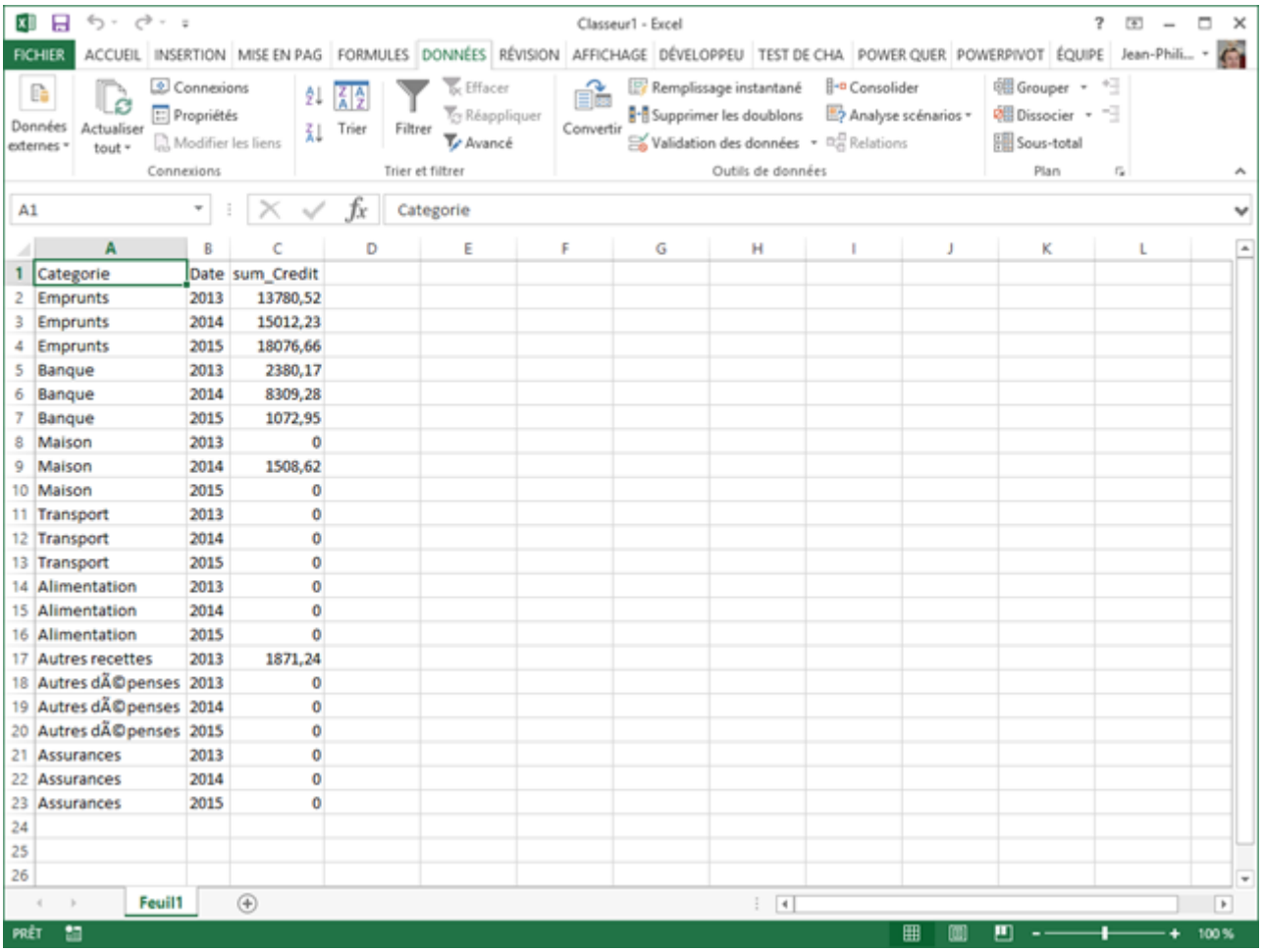
Une fois le processus fini, nous sommes ramenés à l'interface initiale, dans laquelle Wrangler propose une icône sur le dataset calculé pour voir les résultats :



Le fichier CSV résultant peut alors être importé dans un fichier Excel (attention à bien mettre le séparateur en point dans les options) :



Le résultat est le suivant :

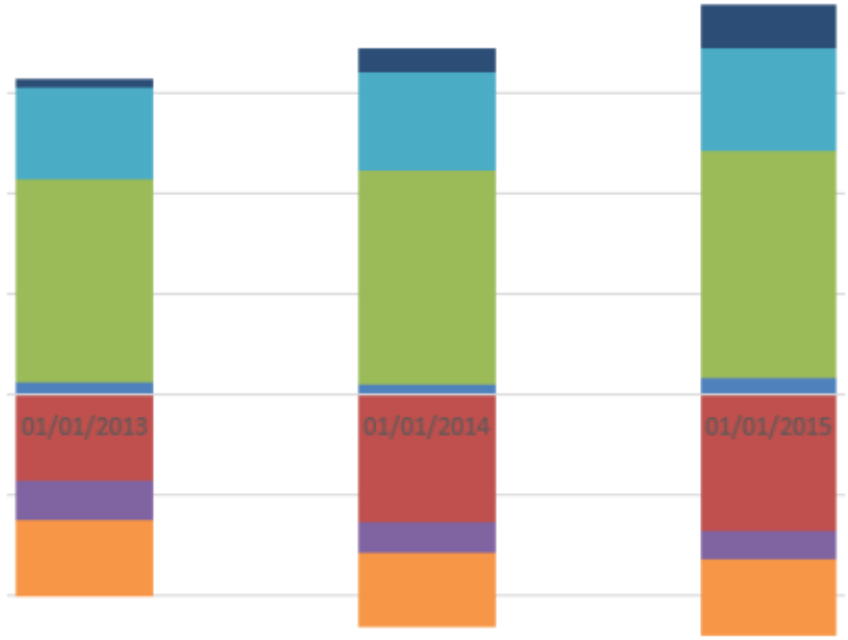


	A	B	C
1	Categorie	Date	sum_Credit
2	Emprunts	2013	13780,52
3	Emprunts	2014	15012,23
4	Emprunts	2015	18076,66
5	Banque	2013	2380,17
6	Banque	2014	8309,28
7	Banque	2015	1072,95
8	Maison	2013	0
9	Maison	2014	1508,62
10	Maison	2015	0
11	Transport	2013	0
12	Transport	2014	0
13	Transport	2015	0
14	Alimentation	2013	0
15	Alimentation	2014	0
16	Alimentation	2015	0
17	Autres recettes	2013	1871,24
18	Autres dÃ©penses	2013	0
19	Autres dÃ©penses	2014	0
20	Autres dÃ©penses	2015	0
21	Assurances	2013	0
22	Assurances	2014	0
23	Assurances	2015	0

On voit encore des problèmes d'accents, toujours à cause du non-support correct des encodages. Vivement que ça soit pris en compte…

A partir de là, bien sûr, tout est possible en termes de réalisation, comme ce genre de graphique avec les dépenses en-dessous du zéro et les recettes au-dessus, ce qui

permet de voir (quand toutes les catégories sont incluses) où en est le budget de la famille et comment il évolue :



L'avantage de l'approche est que le recalcul est ensuite très simple quand des lignes sont ajoutées.

Conclusion

Wrangler est un outil vraiment top que j'ai eu plaisir à découvrir. Il est très ludique et ergonomique, comme j'espère vous l'avoir montré dans ce blog. Sur un prochain article, je ferai peut-être voir le pendant de ceci côté puissance, où Wrangler semble très bon avec la possibilité de paralléliser des analyses (ce qui explique pourquoi, par exemple, il est nécessaire lors de duplication de valeurs précédentes de spécifier quelle colonne donne l'ordonnancement des lignes, comme ce qui se fait dans Power Query, par exemple). Et si vous voulez plus d'exemple, restez à l'écoute de ce site car l'annonce du bouquin sur l'analyse de données Open Data ne devrait pas tarder…