Cartographier le bout du monde

J.-M Friedt, 14 juin 2015

Après avoir acquis sur le web une série de données fournissant la durée du trajet entre une ville de référence et les autres gares françaises, nous nous proposons d'explorer quelques méthodes de traitement des données géoréférencées pour déformer les cartes en fonction de la grandeur physique d'intéret.

Une question d'une profondeur philosophiquee digne de la lutte entre `vi` et `emacs` [1] s'est posée lors d'une discussion sur `irc` : qui, de Besançon ou de Brest, se trouve le plus au bout du monde. Vaste question s'il en est, qui mérite une réponse quantitative et exacte. Il nous faut déjà définir "le bout du monde" : nous définirons la proximité au bout du monde [2] comme proportionnelle au temps nécessaire pour rejoindre ce site depuis Paris en train. Nous plaçons ainsi naturellement Paris au centre du monde, et cherchons à cartographier la durée du trajet entre Paris et un site sur le territoire métropolitain français. Le lecteur audacieux pourra essayer de remplacer Paris par une autre ville comme exercice de mise en pratique des concepts proposés ci-dessous.

Le problème de la représentation cartographique a déjà plusieurs fois été posée dans ces pages, et remplacer la grandeur caractéristique qu'est la longueur géométrique – approximée comme la longueur d'une droite dans une projection locale – par une durée ne facilite pas les choses. Notre objectif étant de rappeler au lecteur la géographie du site considéré, nous partons d'une représentation géographique, que nous allons tenter de déformer (pondérer) par la durée du trajet en train. Cette méthode de traitement de données spatialisées génère des anamorphoses [1] ou, pour faciliter la recherche sur les sites web anglophones (en remplaçant *amme* par *am*), cartogrammes [3].

1 Illustration des cartogrammes

Transformer une carte géographique de coordonnées sphériques vers une représentation planaire appropriée pour un écran n'est déjà par un problème simple et justifie l'utilisation d'un logiciel spécialisé de gestion d'informations spatialisées – QGis – mais déformer une carte pour tenir compte d'une grandeur physique caractérisant chaque pays est encore plus complexe. Nous recherchons donc des implémentations existantes des algorithmes décrits dans la littérature. Notre choix d'outils libres s'est focalisé sur `cart`, disponible à `http://www.umich.edu/~mejn/cart/` – après avoir rejeté l'outil java ScapeToad [2] – tous deux implémentant l'algorithme de diffusion décrit dans [3], et le *plugin Cartogram Generation* de QGis [4]. Le premier est un outil externe, que nous alimentons d'un fichier ASCII (matrice contenant les durées de trajets pour chaque abscisse et ordonnée) et qui nous fournit les nouvelles coordonnées après déformation selon un algorithme appliquant une équation de diffusion aux polygones qui forment la carte. Le second est un outil qui s'interface directement avec les structures de données de QGis, mais qui n'a a priori pas été porté ' a la dernière version de QGis accessible pour Debian/GNU Linux sur `http://qgis.org/debian-nightly jessie main` (version 2.7.0 à la date de rédaction de cette prose), et qu'il faudra donc légèrement pour le script Python le rendre utilisable.

Les seules modifications que nous avons eu à apporter au *plugin* pour le faire fonctionner avec une version récente de QGis sont d'une part étendre les versions de QGis accessibles (`qgisMaximumVersion=2.8`

1. lutte désespérée pour les défenseurs d'`emacs` tant la supériorité de `vi` est évidente
2. notre définition se démarque quelque peu de celle de Terry Pratchett dans *The Color of Magic* – disponible en version numérique à `http://img0.liveinternet.ru/images/attach/c/2/3820/3820555_pratchett_terry__discworld_01__colour_of_magic.pdf` :

There was a line of white on the foreshortened horizon, and the wizard fancied he could hear a distant roaring.
"What happens after a ship goes over the Rimfall ?" said Twoflower.
"Who knows ?"
"Well, in that case perhaps we'll just sail on through space and land on another world." A faraway look came into the little man's eyes. "I'd like that," he said.
[...]
Rincewind groaned and sat up.
This turned out to be a mistake. The edge of the world was a few feet away.
Beyond it, at a level just below that of the lip of the endless Rimfall, was something altogether magical.

3. `http://wwwf.imperial.ac.uk/~mgastner/cartogram/cartogram.html`

dans `metadata.txt`), et d'autre part d'ajouter deux tests sur une variable qui sinon induit une erreur à l'exécution du script Python (`if lfeat.dValue != NULL:` et `if dPolygonValue != NULL :` aux lignes 285 et 305 de `doCartogram.py`, qui aura déjà été corrigé tel que décrit à [5] en éliminant les lignes 58 à 60 du code original pris sur le `git` de `https://github.com/CristianCantoro/cartogram-plugin`). Plus important, cette petite modification est l'opportunité de découvrir la simplicité de développer un *plugin* QGis en Python, et en particulier grâce au *Plugin Reloader* qui évite de relancer QGis pour chaque modification du code [6].

Nous illustrons le bon fonctionnement du greffon après ces modifications en traitant la carte de la population mondiale disponible à `http://thematicmapping.org/downloads/world_borders.php` et en particulier le jeu de données `http://thematicmapping.org/downloads/TM_WORLD_BORDERS-0.3.zip` dont le format Shapefile est parfaitement approprié pour QGis. Le fichier se charge par `Add Vector Layer` pour donner la carte de la Fig. 1 (gauche). Appliquer le greffon *Cartogram Generation* fait croître la surface des pays les plus peuplés – le Nigéria, l'Inde, la Chine et l'Indonésie occupent ainsi une surface proportionnelle à leur population sur la carte et écrasent leurs voisins, tandis que la Russie, l'Australie, les États-Unis et le Canada rétrécissent pour devenir à peine visibles – avec un effet d'autant plus marqué que le nombre d'itérations est important. Noter que ce calcul prend tout de meme plus d'1,5 heures sur un Panasonic CF19 équipé d'un processeur Intel Core i5 cadencé à 2,6 GHz.

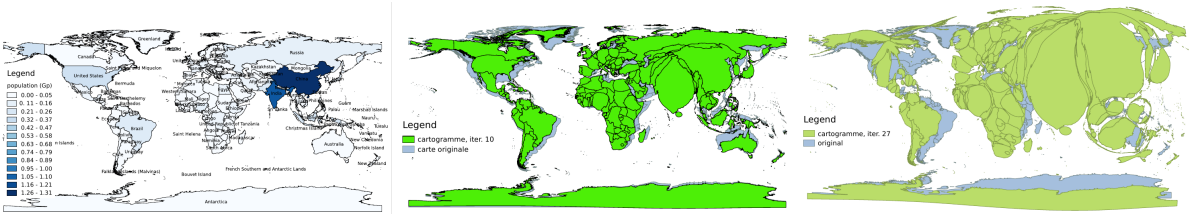

FIGURE 1 – De gauche à droite : carte originale dont le code de couleur indique la population en 2005 ; milieu : cartogramme après 10 itérations (l'arrière plan bleu est la carte originale, en vert la version déformée) ; droite : cartogramme après 27 itérations (l'arrière plan bleu est la carte originale, en vert la version déformée).

Le résultat est concluant et en accord avec nos attentes, nous pouvons donc nous engager sur l'étude d'un jeu de données original.

2 Obtenir des données sur le web

Tout d'abord, il nous faut une carte des durées des trajets en train. Il est bien connu que le site allemand de la Deutsche Bahn, `http://www.bahn.com`, est considérablement plus rapide et souple d'emploi que celui de la SNCF, et notre application va ici encore le prouver. Notre objectif est d'effectuer, pour chaque gare du territoire métropolitain français, une requête sur le site web de la Deutsche Bahn pour être informé de la durée du trajet en venant de Paris. Pour ce faire, nous allons générer des requêtes en format GET, au moyen de `wget`, pour sonder la durée du trajet entre Paris et une liste des gares françaises – l'avantage de passer par Deutsche Bahn est de plus que le concept s'étend trivialement au reste de l'Europe. La liste des gares françaises est issue d'une liste diponible depuis le site d'*opendata* de la SNCF, `https://ressources.data.sncf.com/explore/`, et en particulier des horaires de TER, sous hypothèse que les TER s'arrêtent aussi dans les gares TGV et des trains intercités. Alternativement, les archives du logiciel d'horaires de la SNCF `RIHO` (disponible à `http://lwdr.free.fr/hor.html` et `http://inforail.fr/`) feront l'affaire, mais la pérennité de cette source ne semble pas garantie.

Dans l'archive des horaires TER, le fichier `stops.txt` contient toutes les informations recherchées : nom de la gare et coordonnées GPS. L'objectif de nos manipulations de fichiers, en n'exploitant que les outils standards du shell, inutile de faire appel aux langages récents qui ne présentent que peu d'intéret tant que nous nous cantonnons aux traitements ligne par ligne, vise à extraire le nom des gares françaises en vue d'alimenter un script qui fera la requête de durée de trajet en train depuis Paris. Ayant obtenu ces informations, nous rajouterons les coordonnées GPS de chaque gare – le nom de la gare ne nous intéressera à ce moment plus – pour les positionner sur une carte. On notera simplement que la liste des gares est dédoublée dans le fichier, au début dans le contexte de `StopArea` et ensuite `StopPoint` :

seule une moitié de la liste nous intéresse donc. Nous éliminons de plus tous les caractères indésirables (commande `iconv`) pour une transaction http (cédilles, accents, remplacer des espaces par '+') et générer un fichier contenant uniquement les noms de gares qui alimenteront les requêtes :

```
cat stops.txt | grep pA | cut -d, -f2 | sed 's/"gare de //g' | sed 's/"$//g' \
| sed 's/ /+/g' | iconv -f utf8 -t ascii//TRANSLIT > gares.txt
```

génère ainsi la liste des 2620 noms de gares françaises.

Les requetes GET de Deutsche Bahn s'analysent trivialement et, après en avoir éliminé les champs inutiles, se résument en `http://reiseauskunft.bahn.de/bin/query.exe/fn?revia=yes&existOptimizePrice=1&country=FRA&dbkanal_007=L01_S01_D001_KIN0001_qf-bahn_LZ003&ignoreTypeCheck=yes&S=PARIS&REQ0JourneyStopsS0A=7&Z=${dest}&REQ0JourneyStopsZ0A=7&trip-type=single&date=Me%2C+07.01.15&time=08%3A37&timesel=depart&returnTimesel=depart&optimize=0&infant-number=0&tariffTravellerType.1=E&tariffTravellerReductionClass.1=0&tariffTravellerAge.1=&qf-trav-bday-1=&tariffClass=2&start=1&qf.bahn.button.suchen=` avec `${dest}` le nom de la gare de destination qui sera variable. Cette approche simple échoue sur les plus grosses villes qui proposent plusieurs noms de gare, mais nous pourrions corriger ces déficiences par quelques requêtes manuelles. Sur les 2620 requêtes par une boucle `while 'cat gares.txt' ; do wget -q -O-...;done'` en shell que nous avons faites, 418 se sont soldées par des résultats vides, soit un taux de succès de 84%, acceptable pour un processus complètement automatisé, surtout lorsque les échecs portent sur Lure, Lucy-sur-Cure-Bessy ou Blaisy-Bas. Chaque requête `wget` fournit une énorme séquence de commandes html dont les seuls champs d'intéret (`td class="duration lastrow" rowspan="2"`) sont la durée du trajet extrait par `| grep -A1 uratio | grep ^[0-9] | tr '\n' ' '`. La sortie de cette commande est de la forme

```
Wissembourg # 4:56 4:59 4:22
Bettembourg # 5:12 5:08 5:10
```

Ayant obtenu les durées de trajets, il nous reste à replacer les coordonnées GPS des gares dans le fichier qui contient le nom de la gare et la durée du trajet depuis Paris. Ces coordonées sont obtenues par `cat stops.txt | cut -d, -f3-5 | sed 's/,/ /g' > coord.txt` : les deux fichiers sont fusionnés ligne par ligne au moyen de `paste`. Le résultat est un fichier de la forme (nous ajoutons l'entête pour faciliter l'ouverture par QGis) :

```
Y X ID h m hh mm hhh mmm
27.14097341 -3.40456064 Martigny.
27.14097341 -3.40456064 Vernayaz+MC 5 38 4 51 5 38
27.14097341 -3.40456064 Salvan.
27.14097341 -3.40456064 Les+Marecottes.
27.14097341 -3.40456064 Tretien
27.14097341 -3.40456064 Finhaut.
27.14097341 -3.40456064 Chatelard-gietroz-ch 6 12 5 25 6 12
46.0326774 6.93300074   Vallorcine 6 19 7 41 7 32
52.14889855 11.64178642 Neustadt+(Weinstr)+Hbf 2 54 4 12 4 20
48.68303777 12.69557746 Landau+Hbf 3 11 4 39 3 57
49.09544004 8.1224056   WINDEN+PFALZ
49.03232444 7.9499969   Wissembourg 4 17 3 52 4 02
50.11606317 6.83789926  Bettembourg 2 35 3 50 2 24
49.35393027 6.16885555  Thionville 1 52 2 06 3 37
49.10978917 6.17720253  Metz-Ville 1 31 1 24 2 48
```

Chaque ligne contient les latitudes et longitude (X et Y dans la nomenclature de QGis), le nom de la gare (ou son indentifiant, qui servira de label à chaque point), et les trois durées issues de Deutsche Bahn sous forme de heure et minute séparées par un espace.

3 Cartographie des données récoltées

La première visualisation la plus évidente consiste à charger les points dans QGis et afficher, pour chaque gare, un point dont la couleur dépend de la durée du trajet. La seule subtilité tient au fait qu'une

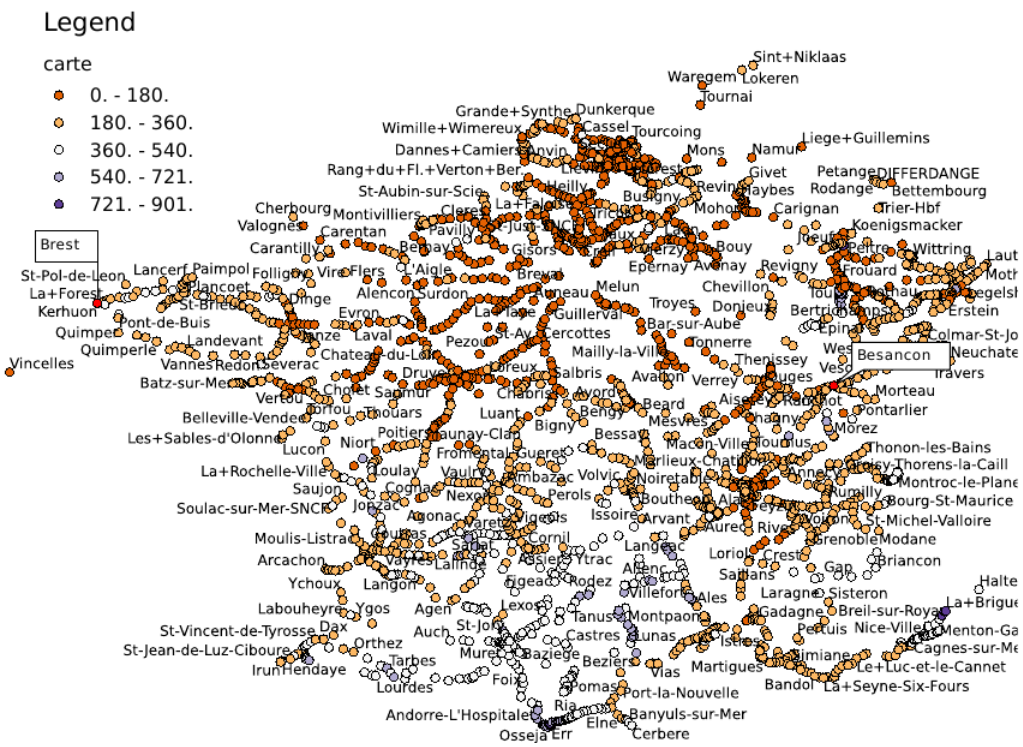

FIGURE 2 – Carte des mesures de temps de trajet en train depuis Paris : chaque gare est positionnée par sa latitude et longitude, et la couleur de chaque point est associée à la durée du trajet en minutes.

couche chargée depuis un fichier texte (icône présentant un symbole en forme de virgule) ne peut être modifiée afin d'ajouter un nouveau champ qui soit égale à la durée en minutes du trajet (HH×60+MM au lieu du format HH :MM), et il faut passer par une sauvegarde intermédiaire des données vectorielles (nom de la gare, coordonnées GPS et durée du trajet sans séparateur " :" entre minutes et secondes) en format Shapefile (`shp`) pour que la couche soit éditable (bouton de droite sur le nom de la couche, `Save As`). Sur la couche Shapefile, `Open Attribute Table`, cliquer sur le crayon en haut à gauche (rendre la table éditable), créer une nouvelle colonne que nous nommerons minutes et dont le contenu est `(h*60+m+hh*60+mm+hhh*60+mmm)/3` pour moyenner les 3 valeurs fournies par Deutsche Bahn. Le Le résultat est proposé sur la Fig. 2.

Nous avons un ensemble de points, qui présentent bien les principaux axes féroviaires. Cependant, une carte n'est pas faite de points mais de surfaces. Deux façons de remplir les surfaces sont d'une part une interpolation en diffusant les valeurs des voisins les plus proches de chaque point des surfaces (*nearest neighbour*) selon une pondération inversement proportionnelle à la distance (IDW), et d'autre part la définition des polygones de Voronoi qui se voient attribuer la valeur de durée de trajet du point qui est au centre de chaque surface.

Il est utile de prendre un peu de recul, avant de se lancer dans des traitements lourds, pour bien comprendre les divers modes d'interpolation et leurs conséquences. Soit un fichier synthétique contenant longitude, latitude et altitude que nous désirons exploiter pour fabriquer, de ces mesures ponctuelles, des surfaces :

```
X Y Z
0 0 0
0 1 0
1 0 0
1 1 1
0 2 0
1 2 2
2 2 0
```

```
2 0 0
2 1 0
1.3 1.3 2
```

Le résultat des diverses interpolations sont proposés en Fig. 3. Il est fondamental de bien comprendre les différences induites par les divers modes d'interpolations, bien qu'en pratique la différence ne soit pas toujours évidente [7].

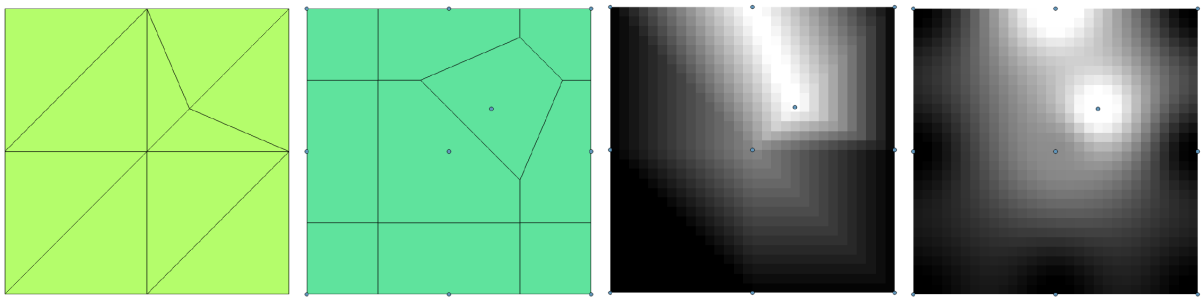

FIGURE 3 – Passage d'une série de mesures ponctuelles (points bleus) en des surfaces interpolées selon diverses méthodes : de gauche à droite, les triangles de Delaunay, les polygones de Voronoi qui optimisent la distribution des polygones autour de chaque point, une interpolation par TIN dont on voit clairement l'alignement le long des arrêtes des triangles de Delaunay, et finalement la pondération inversement proportionnelle à la distance (IDW).

L'application de ces concepts, et en particulier la distribution de polygones de Voronoi aux mesures acquises depuis le site Deutsche Bahn, est illustrée sur la Fig. 4.

Une fois la carte des polygones ou des surfaces interpolées générées, nous pouvons considérer les diverses méthodes de déformer ces cartes pour ne plus représenter la dépendance avec la distance géographique mais avec une grandeur physique, ici la durée du trajet par train depuis Paris.

Le *Cartogram Generation plugin* propose de sélectionner le nombre d'itérations, dont le résultat est illustré Fig. 5. Plus le nombre d'itérations est important, plus la carte convexe devient cylindrique, et plus la déformation dépendante de la grandeur analysée devient significative. Heureusement, chaque polygone conserve ses attributs, dont le nom de la gare du site central au polygone, nous permettant de retrouver la position du site initialement considéré. Nous aidons le lecteur à retrouver la position de Brest et Besançon en ajoutant manuellement des labels sur chaque carte pour chaque site.

Une alternative aux outils intégrés à QGis est l'utilisation de cart. Cet outil transforme une matrice (données de type *raster*) en une série de coordonnées qui indiquent la transformation effectuée sur chaque point de la carte originale. Malheureusement, étant un outil externe, les attributs de la carte initiale sont perdus et les retrouver est une tâche non-triviale. Après transformation de la matrice (issue de la sauvegarde de la carte raster au format ASCII par QGis au moyen de `Raster→Conversion→Translate` du plugin `GDAL`), le résultat – sous forme de matrice de 3 colonnes contenant la nouvelle abscisse, la nouvelle ordonnée et l'attribut d'origine correspondant chez nous au temps de trajet – est chargé dans QGis et interpolé tel que décrit auparavant (interpolation par IDW ou TIN). Le résultat de ces opérations est illustré sur la Fig. 6. Les difficultés à atteindre le sud-ouest de la France et, dans une moindre mesure, l'extrême sud-est, y sont manifestes.

4 Conclusion

Nous avons illustré, par une question triviale concernant la durée de trajet entre deux villes, diverses méthodes de cartographie de données acquises sur le web. L'anamorphose est une méthode de traitement souvent exploitée pour illustrer la distribution spatiale d'une grandeur, souvent la population ou le PIB pour insister sur les inhomogénéités de distributions spatiales.

Quant à la question de savoir qui, de Besançon ou de Brest, se trouve le plus au bout du monde, la Fig. 5 fournit une réponse évidente : alors que Brest reste en périphérie de la carte après les diverses déformations tenant compte de la durée du trajet en train, Besançon s'enfonce doucement, itération

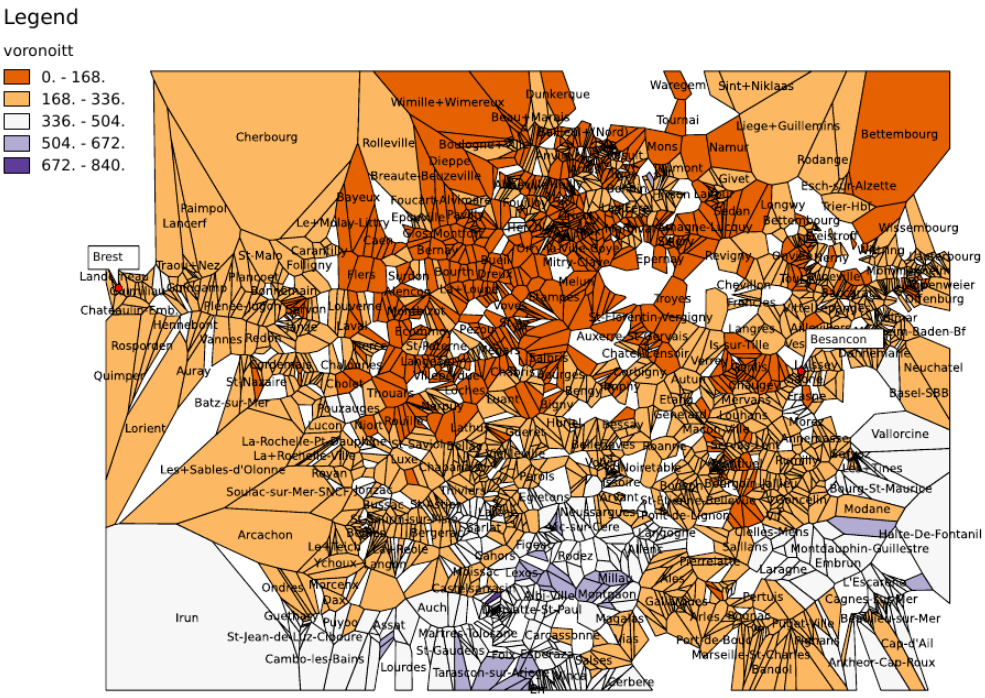

FIGURE 4 – Application de la distribution des polygones de Voronoi aux mesures effectuées par interrogation du site Deutsche Bahn. Chaque polygone se voit attribuer les caractéristiques du point en son centre. Pour faciliter la lecture, deux marqueurs indiquent la position de Besançon et Brest.

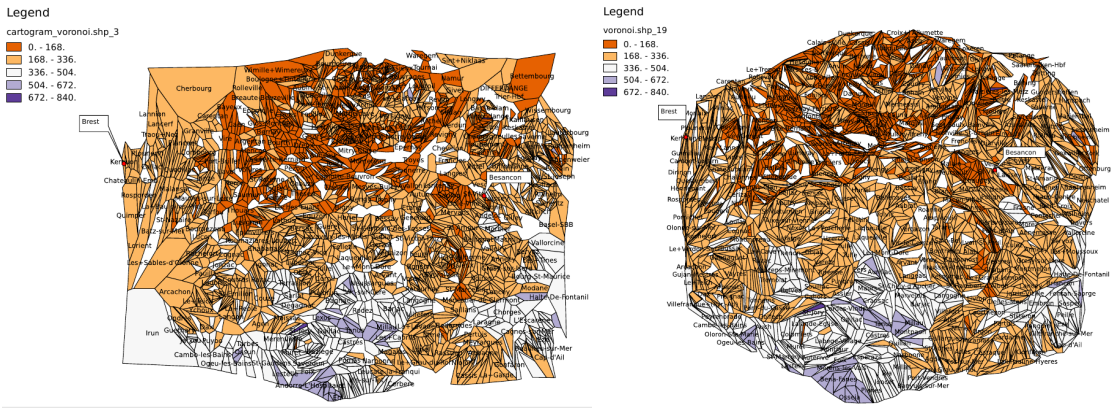

FIGURE 5 – Gauche : trois itération de `cartogram`. Droite : 19 itérations. La carte devient petit à petit circulaire, pour faire disparaître la caractéristiques géographiques initiales (caractéristique d'un algorithme de diffusion, comme une goutte qui finit par occuper une surface de contact circulaire lorsque déposée sur une surface homogène) et distribuer les polygones en fonction de leur pondération. Pour faciliter la lecture, deux marqueurs indiquent la position de Besançon et Brest.

après itération, pour se rapprocher de Paris et s'éloigner de la périphérie de la carte. Il s'en faut de peu pourtant, puisque des villages tels que Pontarlier et Morteau restent dangereusement proches du bord de la carte. Nous restons cependant loin des déboires qui attendent les voyageurs désireux d'atteindre Toulouse ou Nice en train ... heureusement, les trains de nuit permettent de moins se rendre compte de la durée du trajet !

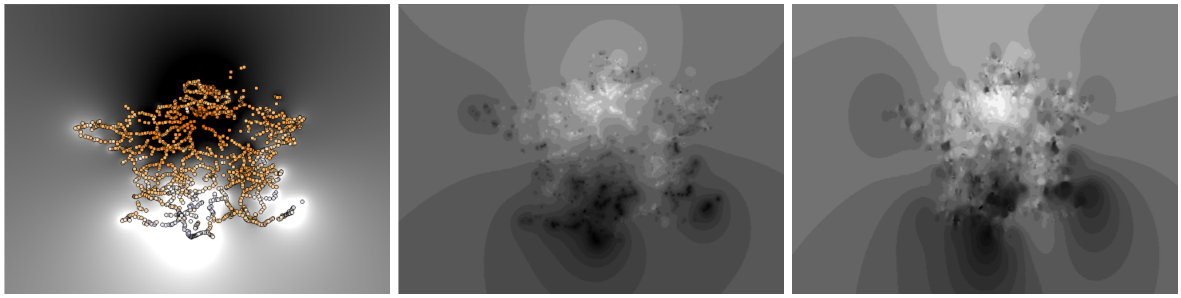

FIGURE 6 – De gauche à droite : fichier d'entrée de `cart`, avec la série de points acquis sur le web et interpolés ; la déformation après interpolation IDW et une coefficient de distance P=2 (milieu) ; la déformation après interpolation IDW et une coefficient de distance P=5 (droite).

Références

[1] J.-C. Denain & P. Langlois, *Cartographie en anamorphose*, MappeMonde 49 (1), 1619 (1998), disponible à `http://www.mgm.fr/PUB/Mappemonde/M198/LangloisDenain.pdf`

[2] `http://scapetoad.choros.ch/`

[3] M.T. Gastner & M. E. J. Newman, *Diffusion-based method for producing density equalizing maps*, Proc. Natl. Acad. Sci. USA **101**, 7499–7504 (2004), disponible à `http://www.pnas.org/content/101/20/7499.full.pdf`

[4] Le *cartogram* plugin est basé sur une implémentation de l'algorithme décrit dans J.A. Dougenik, N. R. Chrisman, & D. R. Niemeyer, *An algorithm to construct continuous cartograms*, Professional Geographer **37** 75-81 (1985)

[5] `http://gis.stackexchange.com/questions/72328/where-has-the-cartogram-plugin-for-qgis-gone/91209#91209`

[6] `http://geoapt.net/pluginbuilder/` fournit la description du *Plugin Builder* qui crée le squelette d'un greffon, dont l'architecture est détaillée dans `http://docs.qgis.org/testing/en/docs/pyqgis_developer_cookbook/plugins.html`. Le greffon `Hello World` est des plus instructifs.

[7] É. Bernard, F. Tolle, M. Griselin, D. Laffly, & C. Marlin. *Quantication des hauteurs de neige et des températures de l'air à la surface d'un glacier : du terrain à l'interpolation, confrontation de méthodes. Neuvièmes Rencontres de Théo Quant, Mars 2009, Besançon, France.*, disponible à `https://hal.archives-ouvertes.fr/hal-00762406/document`