Formation cartographie de l'IXXI, 2023

Éric Guichard et Antoine Chemardin

6 juillet 2023

Introduction

Programme

- initiation à une cartographie orientée maximisation des apports heuristiques de cette méthode.
- On part d'un fond (de coordonnées géographiques, sinon géométriques), d'un fichier de variables (« données » attachées aux objets de ce fond), on produit diverses cartes paramétrables et publiables en ligne.
- L'orientation prise est SVG + Python (+ JavaScript à la fin). Elle aurait pu être SVG + Perl (cf. http://barthes.enssib.fr/carto/fonds-sources-programme).
- On n'utilise pas de logiciel de cartographie (type QGIS, etc.).
- On commence simplement.

Formation orientée cartographie, angles épistémologiques et méthodologiques inclus.

Enjeux théoriques

Démarche théorique nourrie par l'expérience

- Être méthodique, choisir la simplicité, trouver des chemins de traverse quand on ne sait pas faire.
- Assumer le caractère calculatoire de la pensée cher à G. G. Granger.
- Apprendre, faire des exercices, raisonner technologie de l'intellect : l'écriture est une technique, nous a dit J. Goody.
- Par suite, on va beaucoup *jongler* avec des mots (des formes graphiques), produire du sens ou vérifier des hypothèses avec de telles manipulations, somme toute techniques.
- Possibilité d'inférer ce qu'est la culture numérique : la culture de l'écrit contemporain, non stabilisée, avec ses opportunités et ses formes de ségrégation.

On commence simplement : si on sait dessiner des ronds et des carrés, on sait produire une carte.

Détails de la combinatoire

- Des réarrangements permanents à partir de textes, de « données ».
- Même le fond sera manipulé, cassé, réordonné. Par exemple pour trouver les centres approximatifs de polygones.

Types des objets

- Un ou plusieurs fonds
  - Divers lieux surfaciques, chacun avec son **identifiant** suivi de son contour polygonal.
  - Un fond de centres (Londres n'est pas au centre de la Grande-Bretagne).
  - Un fond de lignes (routes, rivières, etc.). . .

  Ces fonds peuvent être regroupés en des calques.
- Fichiers de variables (souvent appelées « données »)
  - Toujours sous la forme : `identifiant` `valeur1` `valeur2`, etc.
  - Les variables peuvent être des noms. Ex. :
    0201 Aisne-1re C.
  - Les variables peuvent être issues de plusieurs fichiers de variables ou de diverses méthodes.
- Quelques scripts, pour aller plus vite (500 objets).

Conseils

- Comme le travail peut s'étaler sur plusieurs semaines, bien numéroter ses scripts : le pgm1 produit un fichier resu1, le pgm2 lit le fichier resu1 pour produire un fichier resu2, le pgm3 etc.
- Ainsi, en cas d'erreur ou de modification des hypothèses, il suffira de relancer toute la chaîne des opérations pour retrouver un nouveau résultat.
- Faire un **journal**, modifié à chaque étape, pour se remémorer (3 mois après ?) l'ensemble des choix, méthodes, hypothèses effectués.
- Se donner les moyens de retrouver aisément ses anciens scripts (outils Linux ?) : comment m'y suis-je pris la dernière fois pour utiliser la commande `split` ?
- Penser l'industrie de l'écriture contemporaine.

Thème de la journée

Cartographier l'abstention au premier tour de la précédente législative. Exercice limité aux 539 circonscriptions de la France métropolitaine et de la Corse.

- Le fond est donné.
- Les fichiers de variables aussi.
- Tous les scripts sont fournis.

Idée

Si l'on admet un pourcentage « normal » d'abstention aux alentours de 30%, et si on suppose que le reliquat de l'abstention donne des voix à un *parti des abstentionnistes* (fictif), combien de députés aurait ce parti au second tour ?

Ce projet ne pourra être réalisé dans sa totalité (on se limite au premier tour).

Mais on produira les cartes relatives à cette recherche, en paramétrant ces valeurs de l'abstention « légitime » et en choisissant seuils et plages de couleur, en commentant les cartes obtenues.

On commencera par des choses très simples, pour se familiariser avec le html, le svg, python.

Documentations

- Point d'histoire de la cartographie dans *Documentations*.
- Découvrir le SVG : dossier *B-DocSVG*.
- Apprendre python : dossier *C-Tutos Python*.
- Autres informations : fichier explicatif de la mesure du taux du « parti des abstentionnistes » : *D-PartiAbstention.pdf*.
- Ce document, *Explications.pdf*.

Les rudiments

Plan

1. Repères historiques de la cartographie.
2. Documentation du SVG seul.
3. Explorations, lectures, transformations :
   - dessin1.html dans *B-DocSVG*.
   - dessin2.html dans *B-DocSVG*.
4. Découverte de python seul (sans SVG).
   - Fichier `0-Mise-en-jambes.py` dans le dossier *Exercices*
5. Ensuite, premiers programmes python travaillant sur du SVG.

Exercice 1

Cf. `1-dessin.py`
Découvrir à la fois python et le svg, format graphique pour le web.

- Le svg a besoin d'un en-tête précis. Comme il est assez long, on va concaténer plusieurs lignes avec le + de python. Différences entre simples et doubles *quotes* (guillemets/apostrophes)
- Toujours svg : Attention à la boîte de visualisation, avec son B Majuscule : `viewBox`
  viewBox="xDépart yDépart largeur hauteur"
- Il n'y a pas d'échelle, juste des coordonnées avec l'axe positif des y qui descend (celui des x est normal) le point (0,0) est en haut à gauche de la fenêtre (ou de l'écran).
- On donne des noms français à nos objets et variables pour ne pas les confondre avec des fonctions python.
- Un *polygone* svg peut avoir beaucoup d'attributs : contour, couleur, etc. Le terme polygone est mal choisi car ce peut être une somme de polygones : Z D'où le nom de variable `polycheminsvg`.

Exercice 1, suite

- En théorie, chaque objet a un identifiant unique. En pratique, les navigateurs sont tolérants.
- Mais l'idéal est de décrire les différentes composantes connexes d'un même objet de la façon suivante :
  <objet id="Obj-1" première comp connexe></objet>
  <objet id="Obj-2" seconde comp connexe></objet>
  Et on fera en sorte que tous les Obj-* aient la même couleur. Cf. éclaté de la région parisienne.
- On découvre les fonctions `replace` et `write` de python
- On écrit du texte en svg
- Essai possible avec anaconda-navigator et JupyterLab, ou le `run` de *Visual Studio Code*.
- Le résultat est le fichier `dessin1.html`, que l'on peut ouvrir.

Exercice 2

Cf. `2-dessin.py`
Comme Ex1, mais on ajoute un cadre, des cercles, un calque.

- Le cadre est inséré dans un *calque* : `<g...> </g>` : plus propre, plus adapté à des CSS.
- Ensuite : python : première ouverture d'un fichier (cercles, en lecture), découverte de `csv.reader`, des listes et des boucles.
- On fabrique les cercles « à la main ». Ne pas se perdre avec les apostrophes et les guillemets.
- La taille de la police est « à l'estime ».
- Jusque là, moitié python, moitié svg.

Exercice 3

Cf. `3-dessin-cercles.py`
Quelques réflexes et sécurités en cas de bug.

- On introduit la bibliothèque `time` pour gérer le temps (fonction `sleep`).
- Comme pour les précédents exercices, profiter des commentaires.
- On découvre la fonction `type`, qui décrit le *type* d'un objet (variable, liste, dictionnaire, etc.).
- Des variables récupèrent directement le contenu d'une ligne-liste.
- Il peut être utile de faire une pause d'une seconde pour lire ce qui apparaît à l'écran.
- On ajoute le rayon (un nombre) à une liste.
- Si on ne fait pas attention, ce rayon est pris comme un caractère et non comme un nombre De ce fait, on découvre que 10 est plus petit que 5 (ordre alphanumérique).
- Le `min` est une fonction de base de python.
- La fonction `exit()` peut être utile aussi pour arrêter le programme, mais la syntaxe qui la suit doit être **correcte** (différence avec le `__END__` de Perl).

Exercice 4

Cf. `4-dessin.py`

Retour au svg

- Initiation aux feuilles de style, qui offrent des attributs identiques aux objets qui y font référence : directement ou via un méta-objet (parent).
- Ici la classe `cercle` servira de référence.
- Pour python : reprise de l'Ex2 avec une nuance : découverte de la fonction `format`, fort commode.
- Ce sera le moyen d'introduire `beautiful soup` et `json`.

Exercice 5

Cf. `5-lecture.py`
Oubli temporaire du svg.

Programme : lire les 565 fichiers des résultats du premier tour des législatives et les synthétiser dans un seul fichier. On s'intéresse aux taux d'abstention supérieurs à la « norme », choisie par l'expérimentateur (ex : 30%, transformé en 30pc pour le nom du fichier).

- On va donc produire un fichier du type `ResultatPourAbstentions30pc` (ou ...32pc, etc.) dans les lignes duquel on écrira
- l'identifiant de la circonscription, le total des inscrits, le pourcentage d'abstentions et ce que serait le pourcentage réel d'un parti des Abstentionnistes.
- Pour ce dernier calcul, cf. le fichier `partiAbstention.pdf`.
- Apparaissent des questions complexes d'identifiant des circonscriptions, qui se démultiplieront avec les fichiers sources (7501 ou 07501 ou 75001 ?).
- Conseil et commentaire : visualiser les deux types de fichiers du dossier `1erTourbis`.

Exercice 5, suite

- Fichiers des résultats électoraux : des fichiers html assez naïfs, non structurés, convertis en txt (Perl).
- Pour chaque fichier texte (*EGO), recherche des lignes Inscrits et Abstentions
- Syntaxe assez claire : `if vraieligne.startswith('Inscrits'):`
- On sépare les valeurs obtenues par des tabulations (choix SHS), on fait attention aux indentations propres à python, et on écrit le résultat dans le fichier `ResultatPourAbstentions30pc`.
- Méthode : ne pas hésiter à multiplier les fichiers intermédiaires quand on est profane, ni à leur donner des noms explicites.
- Preuve qu'on peut aisément travailler sur beaucoup de fichiers à la fois.

Exercice 6

Cf. `6-quantiles.py`.
Calcul de quelques indicateurs du fichier obtenu.

- On fabrique une liste à partir d'une colonne choisie (% effectif de l'abstention, sinon % du parti fictif associé).
- On calcule les quantiles (`import numpy`), le min, le max, l'écart-type, etc. de cette liste.
- Les résultats sont affichés à l'écran.
- Cela nous servira pour définir des seuils à l'origine des plages de couleurs.

Exercice 7

Cf. `7-carte-simple.py`.

- Usage de la bibliothèque `BeautifulSoup`, qui structure l'html/svg en dictionnaire.
- On commence par préciser les paramètres (seuils, couleurs).
- On attribue des couleurs aux circonscriptions.
- On ouvre le fichier `Fond.html`.
- On s'occupe des doublons de la région parisienne (zoomée, à droite de la carte).
- On insère **deux** couleurs dans le fichier du fond et on écrit le résultat dans le fichier `cartesimple.html`, que l'on peut visualiser.

Notes

- `BeautifulSoup` est très efficace mais sollicite de la culture et du doigté.
- Il est aisé de changer les variables, les seuils et les couleurs de la carte.

Sémiologie graphique

- les plages de couleurs sont réservées à la description des pourcentages.
- Les couleurs chaudes (jaune, orange, rouge, etc.) sont réservées aux valeurs (%) positives, les froides (bleu, etc.) aux négatives. Des exceptions sont possibles si elles font sens (rouge : dangereux, vert : calme).
- les cercles et analogues sont réservés aux valeurs entières (population, nombre de cas covid, etc.).
- C'est la surface des cercles et non leur rayon qui est proportionnelle à la valeur décrite. En d'autres termes, le rayon est proportionnel à la racine carrée de la valeur.

Pourquoi une telle sémiologie ?

- Imaginons le contraire, avec la population des Landes et de Paris : les Landes seront coloriés en jaune et attireront le regard car c'est le plus grand département de l'hexagone ; Paris en rouge mais invisible car trop petit.
- Par ailleurs, on ne sait pas représenter des cercles à rayon négatif...

Exercice 8

Cf. `8-carte1.py`.

- Développement de l'exercice 7 (plusieurs couleurs).
- On écrit le résultat dans le fichier `carte1.html`, que l'on peut visualiser.

Exercice 9a

Cf. `9a-carte2.py`.
Fabrication de la légende : des caissons décrivant les couleurs et leurs seuils.

- On structure ce calque via `BeautifulSoup` (un peu comme on l'avait fait pour les couleurs de l'exercice 8).
- On ajoute ce calque dans le fond.

À titre indicatif existe aussi un `9b-cartes2-avec-cercles.py` qui ne sera pas présenté en cours : ce script permet d'insérer des cercles dont la *surface* est proportionnelle à une variable donnée.

Comme la démocratie propre à la France fait que le nombre d'électeurs pour un député est relativement constant : environ 60 000. Et donc, tous les cercles auront à peu près la même taille. L'ajout de ces cercles n'apportera donc pas de réelle information.

Exercice 10

Cf. `10-carte-finale.py`

- Complète les précédents scripts en affichant des informations (au niveau de la Suisse) lorsque le curseur passe sur une circonscription.
- Produit la carte `carte3.html`.

Suites : ajouter un titre et un commentaire, analyser la carte, changer les paramètres, voire les variables, etc.

Dossiers 11JS

Vient ensuite une initiation à `javascript` (et au `html`) pour aboutir graduellement au fichier `tutoJS.html` du dossier JS3.