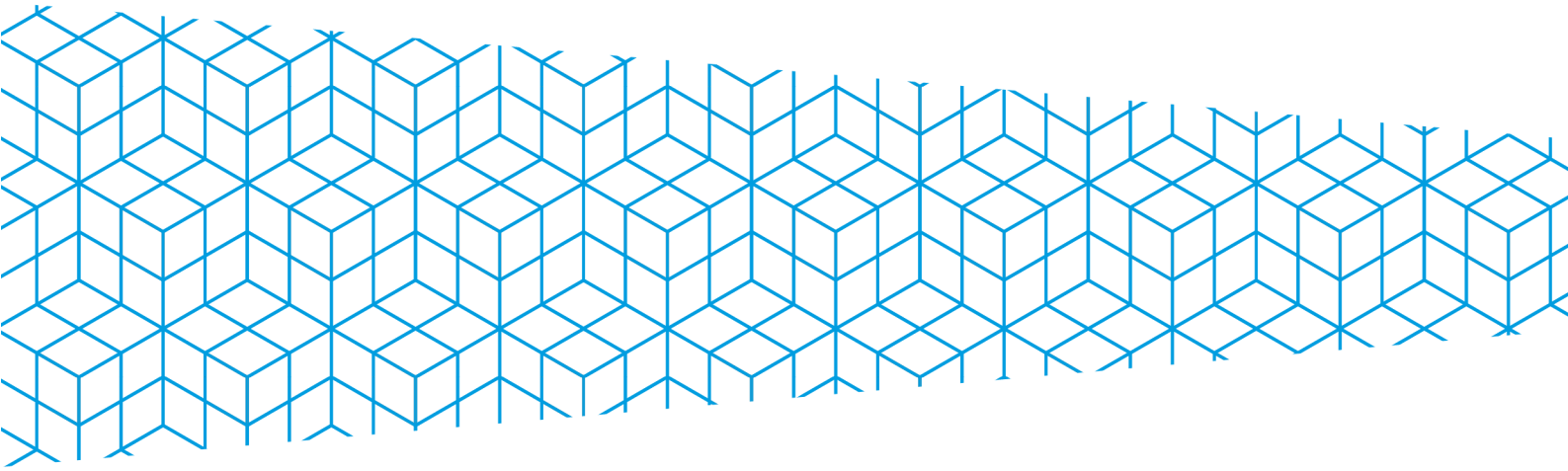


université de BORDEAUX

Table des matières ou plan

Complément de texte
Texte complémentaire

Visualisation Python: design, matplotlib et open source

Dataviz Universe - 2024

Introduction

Texte courant corps 11. Ro vellessed erum dolupienis nime et optatiisit quis pro inctusa ndistium eatempos accus incitas dia de vitatis sinum quid que litam voluptaecum laceped ea nimi, sit, sequam, ipicias utemo mincia quo mi, ea eum eniment ullorit earum sum nost, cum quis poria quo veruntisi doluptatur, ommostrupta secumque conecto modist, oditet dolo et verum ipsanda eceate doluptatur, te vero quatur as rempos aspid eum harchic iaersped que preperiberum auda si omnihillessi volo cupitatus ex etus senti alitia con re vendem quid eat pa veles nonsere none rem simillа dolecto volum quiscias eos archici enihicimet est, offic tem rectatem et, occus essima vel maior a nis auditatia exerum assunt, quo doluptatem ad moluptiore voluptatum dolenis qui cus aut reruptint, simi, sint.

Contexte

Texte courant en corps 11. Ro vellessed erum dolupienis nime et optatiisit quis pro inctusa ndistium[1] eatempos accus incitas dia de vitatis sinum quid que litam voluptaecum laceped ea quis poria quo veruntisi doluptatur, ommostrupta secumque conecto modist, oditet dolo et.

Tem rectatem et, occus essima vel maior a nis auditatia exerum assunt, quo doluptatem ad moluptiore voluptatum dolenis qui cus aut reruptint, simi, sint :

- Optis iditiberum aut venihit aspelit estem ;
- La doluptatia sunt ipsum nonsequundel mos architem nonsequo ;
- Dupide destotatur soluptat voluptas parum fugiae mint late nis doluptumque eostiosam nis auta deles simet experehent ;

Objectifs

Texte courant en corps 11. Ro vellessed erum dolupienis nime et optatiisit quis pro inctusa ndistium[2] eatempos accus incitas dia de vitatis sinum quid que litam voluptaecum laceped ea quis poria quo veruntisi doluptatur, ommostrupta secumque conecto modist, oditet dolo et.

Tem rectatem et, occus essima vel maior a nis auditatia exerum assunt, quo doluptatem ad moluptiore voluptatum dolenis qui cus aut reruptint, simi, sint

[1]Loi, référence etc.

[2]Loi, référence etc.

Plan du document (si nécessaire)

Texte courant corps 11. Ro vellessed erum dolupienis nime et optatiisit quis pro inctusa ndistium eatempos accus incitas dia de vitatis sinum quid que litam voluptaecum laceped ea nimi, sit, sequam, ipicias utemo mincia quo mi, ea eum eniment ullorit earum sum nost, cum quis poria quo veruntisi doluptatur, ommostrupta secumque conecto modist, oditet dolo et verum ipsanda eceate doluptatur, te vero quatur as rempos aspid eum harchic iaersped que preperiberum auda si omnihillessi volo cupitatus ex etus senti alitia con re vendem quid eat pa veles nonsere none rem simillа dolecto volum quiscias eos archici enihicimet est, offic tem rectatem et, occus essima vel maior a nis auditatia exerum assunt, quo doluptatem ad moluptiore voluptatum dolenis qui cus aut reruptint, simi, sint.

Partie I. Optis iditiberum aut venihit aspelit estem a doluptatia sunt ipsum nonsequundel mos architem nonsequo cupide destotatur soluptat voluptas parum fugiae mint late nis doluptumque eostiosam nis auta deles simet experehent.

Partie II. Archite nderit, sus et quaturem qui antiusc ienienis dolupta cum alicia secearciae premolorpos nossin nihicil igenti autae vitistis nam earuntio int officia dolorae modi volorent. Aqui des dolorati optio veni tem fugitet ut landipsus expeliqui quatias aut officie nditas atiur, nest autem aut del iumende aliquid itiore quatet audaepelis si dolupta dolest, sandiste nimenti busamus, sum et molesequi ilicima gnatqui beation eum dolore, est eumqui.

Partie I. Visualisation de données via Python

I/ De la donnée brute à la donnée traitée

Texte courant corps 11. Ro vellessed erum dolupienis nime et optatiisit quis pro inctusa ndistium eatempos accus incitas dia de vitatis sinum quid que litam voluptaecum laceped ea nimi, sit, sequam, ipicias utemo mincia quo mi, ea eum eniment ullorit earum sum nost. Optis iditiberum aut venihit aspelit estem a doluptatia sunt ipsum nonsequundel mos architem nonsequo cupide destotatur soluptat voluptas parum fugiae mint late nis.

Titre niveau 3 sur une ou plusieurs lignes

Texte courant corps 11. Ro vellessed erum dolupienis nime et optatiisit quis pro inctusa ndistium eatempos accus incitas dia de vitatis sinum quid que litam voluptaecum laceped ea nimi, sit, sequam, ipicias utemo mincia quo mi, ea eum eniment ullorit earum sum nost, cum quis poria quo veruntisi doluptatur, ommostrupta secumque conecto modist, oditet dolo et. Optis iditiberum aut venihit aspelit estem a doluptatia sunt ipsum nonsequundel mos architem nonsequo cupide destotatur soluptat voluptas parum fugiae mint late nis.

II/ De la donnée traitée à l’histoire

Texte courant corps 11. Ro vellessed erum dolupienis nime et optatiisit quis pro inctusa ndistium eatempos accus incitas dia de vitatis sinum quid que litam voluptaecum laceped ea nimi, sit, sequam, ipicias utemo mincia quo mi, ea eum eniment ullorit earum sum nost. Optis iditiberum aut venihit aspelit estem a doluptatia sunt ipsum nonsequundel mos architem nonsequo cupide destotatur soluptat voluptas parum fugiae mint late nis.

III/ De l’histoire au design

Texte courant corps 11. Ro vellessed erum dolupienis nime et optatiisit quis pro inctusa ndistium eatempos accus incitas dia de vitatis sinum quid que litam voluptaecum laceped ea nimi, sit, sequam, ipicias utemo mincia quo mi, ea eum eniment ullorit earum sum nost. Optis iditiberum aut venihit aspelit estem a doluptatia sunt ipsum nonsequundel mos architem nonsequo cupide destotatur soluptat voluptas parum fugiae mint late nis.

IV/ Du design au code

Texte courant corps 11. Ro vellessed erum dolupienis nime et optatiisit quis pro inctusa ndistium eatempos accus incitas dia de vitatis sinum quid que litam voluptaecum laceped ea

nimi, sit, sequam, ipicias utemo mincia quo mi, ea eum eniment ullorit earum sum nost. Optis iditiberum aut venihit aspelit estem a doluptatia sunt ipsum nonsequundel mos architem nonsequo cupide destotatur soluptat voluptas parum fugiae mint late nis.

Partie II. Développement d'outils open source pour matplotlib

I/ Matplotlib et ses extensions

Matplotlib est la plus célèbre librairie dédiée à la visualisation de données en Python. Créée il y a plus de 20 ans, elle possède aujourd'hui une utilisation massive et est accompagnée d'un large nombre d'extensions.

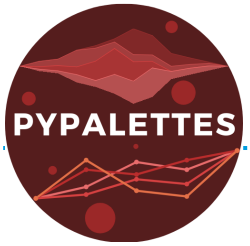
Malgré le fait que Matplotlib permet la création de graphiques en relativement peu de lignes de code, elle sert également de boîte à outil pour de nombreuses autres librairies écrites par dessus elle. Les plus connues sont Seaborn, un librairie qui reprend le coeur de matplotlib mais permettant un accès rapide et intuitif à des graphiques complexes, ou bien Plotnine, une librairie qui implémente uniquement à partir de Matplotlib la grammaire des graphiques (principalement connue via ggplot2) pour un usage en Python.

Ces 2 exemples ne représentent pourtant qu'une infime partie de tout ce que la communauté open source Python a créé autour de matplotlib. En effet, réaliser un graphique avec une qualité publiable dans un média grand publique requiert une quantité importante de code ainsi qu'une certaine complexité, si l'on utilise uniquement Matplotlib.

Même après plus de 20 d'existence, Matplotlib a toujours besoin de ses extensions et des gens qui les créent. A force de créer des graphiques avec matplotlib j'ai remarqué que :

beaucoup de mon code est redondant, facile à généraliser à des cas d'usages plus larges et complexe à documenter. Pour cette raison, Yan et moi avons créé 3 extensions pour (principalement mais pas uniquement) Matplotlib, simple d'utilisations et open sources.

II/ PyPalettes



Paletteer

En R, la librairie la plus populaire pour la gestion des couleurs dans un graphique est Paletteer, une agrégation de palettes de la plupart des packages R dédiés aux couleurs. Ce package donne accès à plus de 2500 palettes de couleur pré-faites accessibles via une simple API.

L'idée fut donc de trouver un moyen de rendre ces même palettes accessibles dans une interface Python. Etant donné que les palettes en question sont sous des licences libres, j'ai écris un script Python qui scrappe la documentation de Paletteer avec toutes les valeurs hexadécimales et le nom de chaque palette pour créer un fichier avec toutes les palettes.

Création de PyPalettes

Puis j'ai ajouté toutes les palette de couleurs déjà présentes dans Matplotlib et Seaborn afin d'avoir un outil le plus exhaustif possible. J'ai ensuite créé une API la plus simple possible pour les utilisateurs, avec une principale fonction : `load_cmap()`. Cette fonction prend un nom de palette et quelques autres arguments additionnels (pour des cas d'usages plus spécifiques) et retourne un object matplotlib dédié au palette : une colormap.

Nous avons effectué plusieurs itérations avant d'atteindre un résultat satisfaisant, la complexité résidant dans le fait que l'utilisateur n'est pas à se préoccuper de la complexité en arrière plan. PyPalettes est maintenant créé et fonctionnel.

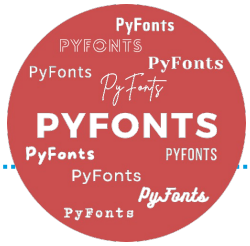
Publication et communication

Afin de communiquer au mieux sur cet outil, Yan a créé une application web qui permet de choisir une palette parmi la large liste et de voir automatiquement à quoi elle ressemble sur plusieurs types de graphiques Python. L'application permet une visualisation très rapide d'un grand nombre de palettes en simulant le style des graphiques matplotlib via des outils JavaScript, accessible via une simple url dans un navigateur.

L'étape finale fut alors de faire un publication de cette librairie sur PyPi, le gestionnaire de package Python le plus populaire. Cet outil est peu amené à évoluer fondamentalement, en dehors de simples corrections de bugs et d'ajout de palettes.

Nous avons communiqué fortement sur LinkedIn et Twitter, ainsi que sur divers blogs afin d'attirer les gens vers le projet. Aujourd'hui PyPalettes est installé entre 10 et 15 fois par jour et a environ 200 favoris sur le projet Github associé contenant le code source.

II/ PyFonts



Gestion des polices

La gestion des polices de caractères est surprenamment complexe, et par conséquent, leur gestion dans un logiciel en hérite. Même si Matplotlib fournit un set de polices nativement, leur nombre reste très limitant. Cependant, Matplotlib donne surtout accès à un gestionnaire de police (ou “font manager”), qui permet l’utilisation de n’importe quel police à condition d’avoir un accès local aux fichiers binaires associés à la police souhaité.

Les problèmes de ce type de fonctionnement sont les suivants :
- nécessite le téléchargement en amont de la police sur son ordinateur
- rend le code non-reproductible en :
    - nécessitant l’écriture en clair d’un chemin de dossier propre à un utilisateur
    - obligeant tout autre utilisateur du code d’avoir les fichiers de police

Création de PyFonts

Je me suis donc lancé dans la recherche d’un moyen d’accéder à n’importe quelle police sans installation quelconque. Dans la mesure où l’écrasant majorité des polices sous license libres sont répertoriées dans des projets Github, et ce que ce dernier permet facilement d’accéder à une url vers la version “brute” (ou “raw”) d’un fichier binaire, je me suis lancé dans la création d’un moyen d’aller chercher les fichiers à leur source même.

La première version actuelle de PyFonts permet d’accéder à une fonction load_font(), qui à partir d’une url vers n’importe quel police de charger un objet FontProperties via le gestionnaire de police de Matplotlib. Dans la pratique, la fonction va créer un fichier temporaire local du fichier binaire, l’utiliser pour initialiser l’objet FontProperties et ensuite supprimer le fichier. De cette manière le code est court (entre 3 et 4 lignes avec les outils Matplotlib traditionnels contre seulement 1 ligne avec PyFonts), simple (appel d’une fonction avec une url comme argument) et indépendant de l’ordinateur (charge depuis le net).

Futur du projet

Les prochaines étapes sont de simplifier encore plus le processus créant un algorithme qui va automatiquement chercher au sein de Google Font (plus grande base de données de police) la police voulue en question uniquement via son nom. Cette étape est encore en développement, mais PyFonts est aujourd’hui disponible sur PyPi et parfaitement fonctionnel.

II/ DrawArrow

université de BORDEAUX