Aabu

Se débarasser de la palette « jet » avec Jet Killer

22 mars 2019

Table des matières

1. Introduction

Ce billet présente Jet Killer , le petit projet ayant occupé mes soirées ces derniers temps. L'idée de ce projet est venue d'une discussion sur le forum .

2. Contexte

Dans le monde de la visualisation de données, la palette de couleur « jet » faisait jusque récemment office de reine, et était utilisée comme palette par défaut, notamment dans Matlab . Cependant, cette palette présente des défauts importants :

— elle n'est pas perceptuellement uniforme, ce qui peut masquer des détails ou créer des artefacts ;
— elle peut poser des difficultés aux personnes atteintes de daltonisme , ce qui est assez courant chez les hommes.

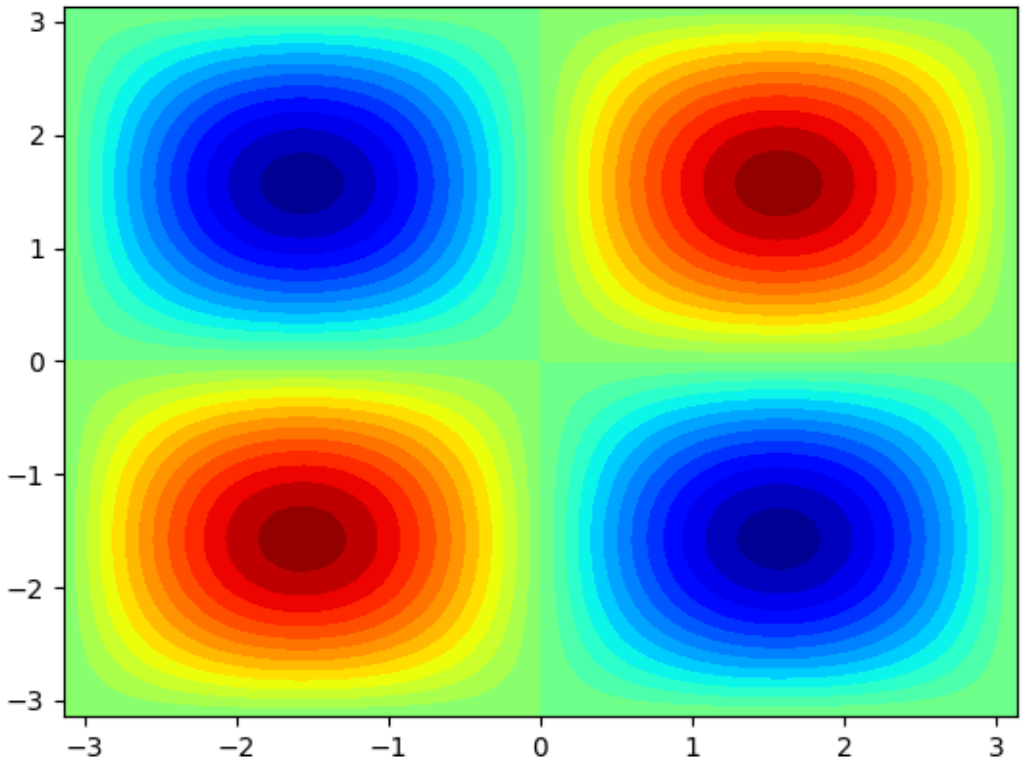

FIGURE 2. – Exemple de visualisation utilisant « jet ».

Ces différents défauts ont poussé à l'adoption de nouvelles palettes de couleurs à la fois perceptuellement uniformes et adaptées au daltonisme. Ces palettes de couleur ont été choisies comme valeur par défaut, et on trouve donc de plus en plus de visualisations utilisant « viridis » (palette par défaut de matplotlib) ou « parula » (palette par défaut de Matlab).

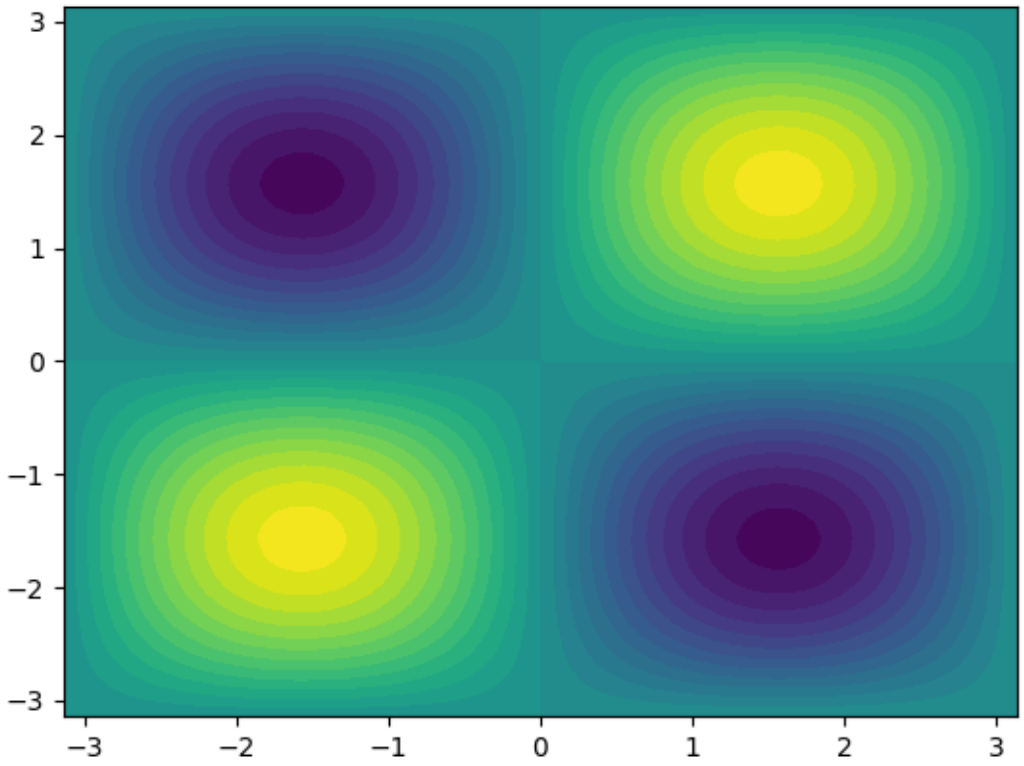

FIGURE 2. – Exemple de visualisation utilisant « viridis ».

3. Le problème

Au vu des défauts de « jet », il est conseillé de l'éviter en premier lieu, et de choisir la palette de couleur par défaut de son outil préféré (« viridis » dans mon cas). Si on a une vielle figure utilisant « jet » et qu'on dispose encore des données d'origine, il est également conseillé de régénérer la figure avec une meilleure palette de couleur (« viridis » est souvent amplement suffisante).

Cependant, quand on ne dispose pas des données d'origine, il n'est pas possible de faire tout cela... Pourtant, la situation peut arriver dans des cas relativement courants :

— récupération de figures depuis une source ayant fait un choix malheureux de palette de couleurs,
— utilisation de graphes tracés à partir de données ayant été possiblement perdues,
— visualisation rapide d'un mauvais graphe sans se fatiguer à le régénérer depuis les données originelles.

Que faire alors ?

4. Une solution : Jet Killer

Pour répondre à ce genre de problématiques, j'ai écrit un petit outil en Python, qui s'appelle Jet Killer .

Le principe est très simple : à partir d'une image utilisant « jet », Jet Killer génère une image utilisant une meilleure palette de couleur.

FIGURE 4. – Principe de fonctionnement de Jet Killer.

Le traitement se fait directement sur les pixels, sans avoir besoin de connaissance des données sous-jacentes. Pour chaque pixel de l'image (à l'exception des gris), on trouve la couleur la plus proche sur la palette « jet », qu'on remplace par la couleur correspondante sur la palette d'arrivée. Les pixels gris ne sont pas modifiés, ce qui permet, dans une certaine mesure, de conserver les textes et cadres des visualisations.

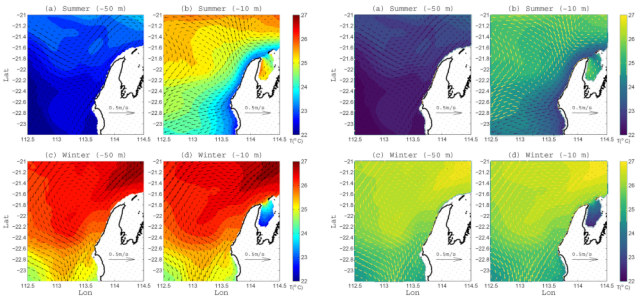

FIGURE 4. – Exemple de conversion. À gauche l'image d'origine (source), à droite le résultat.

4.1. Installation

L'outil est disponible sur PyPi . Pour l'installer, il suffit d'utiliser la commande suivante, qui installe le *package* `jetkiller` et la commande du même nom :

```
pip install jetkiller
```

Si vous souhaitez seulement tester, l'utilisation d'un *virtualenv* est évidemment conseillée.

Si vous ne disposez pas de Python, ni de `pip`, il vous faudra les installer.

4.2. Utilisation

4.2.1. En ligne de commande

Jet Killer est conçu principalement pour être utilisé en ligne de commande.

Utilisez la commande suivante pour convertir `input_file` en `output_file` :

```
jetkiller input_file output_file
```

Si vous omettez l'argument `output_file`, Jet Killer convertit par défaut `input_file` en "output.png" :

```
jetkiller input_file
```

Si le fichier "output.png" existe déjà, il est écrasé sans avertissement.

Il est possible de changer la palette de couleur d'arrivée (par défaut "viridis") en utilisant l'option `--colormap` (ou sa dénomination courte `-cm`). N'importe quelle valeur des palettes de matplotlib est acceptée. Voilà un exemple utilisant « inferno » :

```
jetkiller input_file output_file --colormap inferno
```

4.2.2. Depuis Python

L'utilisation depuis Python se fait en important le *package* `jetkiller` et en utilisant la fonction `jetkiller` :

```
import jetkiller as jk
jk.jetkiller("input_image.png", "output_image.png")
```

Similairement à la ligne de commande, le nom du fichier de sortie peut être omis et prend alors la valeur par défaut ”output.png” :

```
1 import jetkiller as jk
2 jk.jetkiller("input_image.png")
```

Toujours pareil qu'en ligne de commande, si le fichier ”output.png” existe déjà, il est écrasé sans avertissement.

Le changement de la palette de couleur se fait avec l'argument optionel `colormap`, qui vaut ”viridis” par défaut, mais qui peut prendre n'importe quelle valeur prise dans la liste des palettes de matplotlib .

```
1 import jetkiller as jk
2 jk.jetkiller("input_image.png", "output_image.png",
      colormap="inferno")
```

4.3. Formats d'image pris en charge

Jet Killer prend notamment en charge PNG et JPEG, ainsi que la plupart des formats pris en charge par le *package* Pillow sur lequel Jet Killer se repose pour la manipulation des images.

5. Conclusion

6. En savoir plus

Pour en savoir plus (mais vous savez presque déjà tout), rendez-vous sur le depôt du projet .