Projet informatique de traitement des données en SHS

Nicolas Carrara

Parcours MASE/SCC/SISL licence MIASHS, S3, Lille 3.

Année 2018-2019

Informations diverses

- Me contacter : nicolas.carrara@protonmail.com
- Supports : moddle ou ncarrara.fr (à définir)

Objectif

- Lire/écrire des données.
- Manipuler des données.
- Afficher des données.

Evaluation

- Une évaluation écrite de 1h30/2h.
- Une présentation orale du projet.

Python

Utilisations

- Developpement Web.
- Dev logiciel.
- Mathématiques.
- Scripts.

Python

Avantages :

- Simple à apprendre.
- Langage avec interpreteur. Prototypage rapide.
- Syntaxe simple et naturelle pour les anglophones.
- Très peu de lignes de code.
- Objet/Functionnel/Procédural.
- Bon écosystème pour l'intelligence artificielle.

Inconvénients :

- Lent.
- Non typé.

Lancer un programme python

2 façons d'utiliser python :

- Avec la ligne de command python. Taper python dans un terminal pour ouvrir l'interpreteur.
- En créant un fichier avec une extention .py et taper `python exemple.py` dans un terminal.

Eléments basiques de python

Démonstration en direct de python.

- Syntaxe de base : Variable, Nombres, Cast, Chaîne de caractères, opérateurs.
- Structure de données : Listes, tuples, ensembles, dictonnaires.
- Syntaxe avancée : if then else, while, for

Eléments basiques de python

Retrouvez un tutorial complet :

- En français :
  https ://openclassrooms.com/fr/courses/235344-apprenez-a-programmer-en-python
- En anglais (simple) :
  https ://www.w3schools.com/python

Les fonctions

Une fonction **f** est un objet python.

- Assimilable au concept de fonction en mathématiques.
- Elle prend en entrée des paramètres (*arguments*).
- Sa sortie est le *return*.

La syntaxe d'une fonction

Par exemple la fonction $f(x) = x^2$:

```
def f(x):
    y=x**2
    return y
```

Notez l'**indentation** à la deuxième et troisième ligne.

Une fonction à deux arguments :

```
def f(x,y):
    z=x+y
    return z
```

Gestion de fichiers

Ouvrir un fichier avec la fonction `open()`. Différents modes :

- `"r"` - Lire (par defaut) (erreur si le fichier n'existe pas)
- `"a"` - Ajouter (création du fichier si inexistant)
- `"w"` - Ecrire (création du fichier si inexistant)
- `"x"` - Créer (erreur si le fichier existe déjà)
- `"t"` - Texte (par defaut)
- `"b"` - Binaire (e.g. images)

Utilisation :

- `f = open("fichier.txt")`
- `f = open("fichier.txt", "rt")`

Lire un fichier

Une fois ouvert en lecture (`f=open("fichier.txt")`) :

- `f.read()` : lire le fichier en entier
- `f.read(5)` : lire les 5 premiers caractères.
- `f.readLine()` : lire une ligne.
- Lire ligne par ligne avec une boucle `for x in f:`

L'utilisation de read ou readline change le pointeur dans le fichier (exemple en direct).

Ecrire dans un fichier

Attention, `"w"` écrase les données :

```
f = open("fichier.txt", "w")
f.write("ceci va remplacer l'ensemble du fichier")
```

Utiliser `"a"` pour ajouter :

```
f = open("fichier.txt", "a")
f.write("ceci va se rajouter à la fin du fichier")
```

Supprimer un fichier ou un dossier

Importer le package *operating system* avec `import os` puis :

- `os.remove("fichier.txt")` pour un fichier
- `os.rmdir("dossier")` pour un dossier

Vérifier qu'un fichier existe :

- `os.path.exists("fichier.txt")`

Retourne un booléen.

Gestion de données avec numpy

Numpy est une bibliothèque essentielle pour le calcul scientifique en python.

- L'élément de base est la matrice.
- Rapide, grâce à l'appel de code C.
- Intuitif pour le raisonnement matriciel.
- Permet de manipuler facilement des données numériques, faire des moyennes, médiannes ...

Numpy : le tableau

Numpy manipule des objets numpy.array.

```
import numpy as np
a = np.array([[ 0,  1,  2,  3,  4],
              [ 5,  6,  7,  8,  9],
              [10, 11, 12, 13, 14]])
```

Alors :

- `a.shape` est `(3,5)`
- `a.ndim` est `2`
- `a.dtype.name` est `'int64'`
- `a.size` est `15`

Numpy : instanciation d'un tableau

Il existe différentes méthodes pour créer un ndarray :

- Le constructeur de base : `np.array([1,2,3])`
- Un tableau de zeros : `np.zeros((3,4))`
- De uns : `np.ones((2,3,4),dtype=np.int16)`
- Un tableau vide : `np.empty( (2,3) )`
- Suite de nombres :
  - `np.arange( 10, 30, 5 )`
  - `np.arange( 0, 2, 0.3 )`
  - `np.linspace( 0, 2, 9)`

Numpy : opérations (1)

Différentes opérations matricielles :

- Addition elt/elt : `A+B`
- Multiplier par un scalaire : `k*A`
- Multiplication elt/elt (si même forme) : `A*B`
- Multiplication matricielle : `A @ B`

Numpy : opérations (2)

Les opérations suivantes sont élément par élément :

- La valeur absolue : `np.abs(A)`
- L'exponentielle : `np.exp(A)` (et `np.log(A)`)
- Trignométrie : `np.sin(A)` (ou cos, arccos, arcsin)
- Générer une matrice aléatoire $n \times p$: `np.random.rand(n,p)`

Numpy : statistiques

Différentes opérations matricielles :

- La médianne : `np.median(a[, axis, ...])`
- La moyenne : `np.mean(a[, axis, dtype, ...])`
- L'écart type : `np.std(a[, axis, dtype, ...])`
- L'histogramme : `np.histogram(a[, bins, ...])`

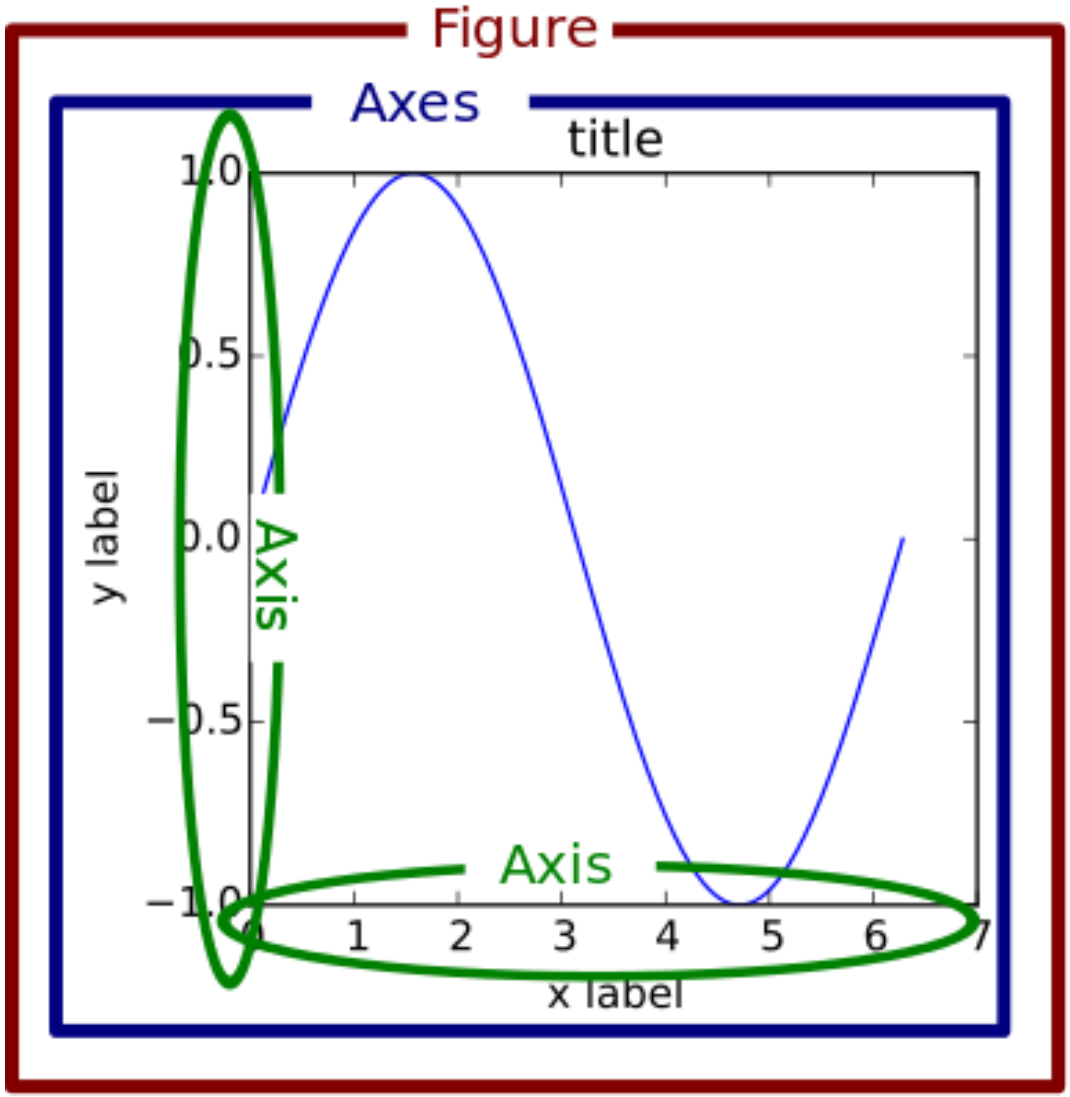
Numpy : documentation

Retrouver toutes la documentation sur
https ://docs.scipy.org/doc/numpy/reference/routines.html

Visualisation des données avec matplotlib

matplotlib est un package python introduit par le neurobiologist John D.Hunter en 2003. Le projet a été repris par la communauté en 2012 après la mort de John.

Nomenclature



Stateless vs Statefull

Approche *statefull* : manipulation autour du wrapper **plt**.
Opère toujours sur le graphe courant.

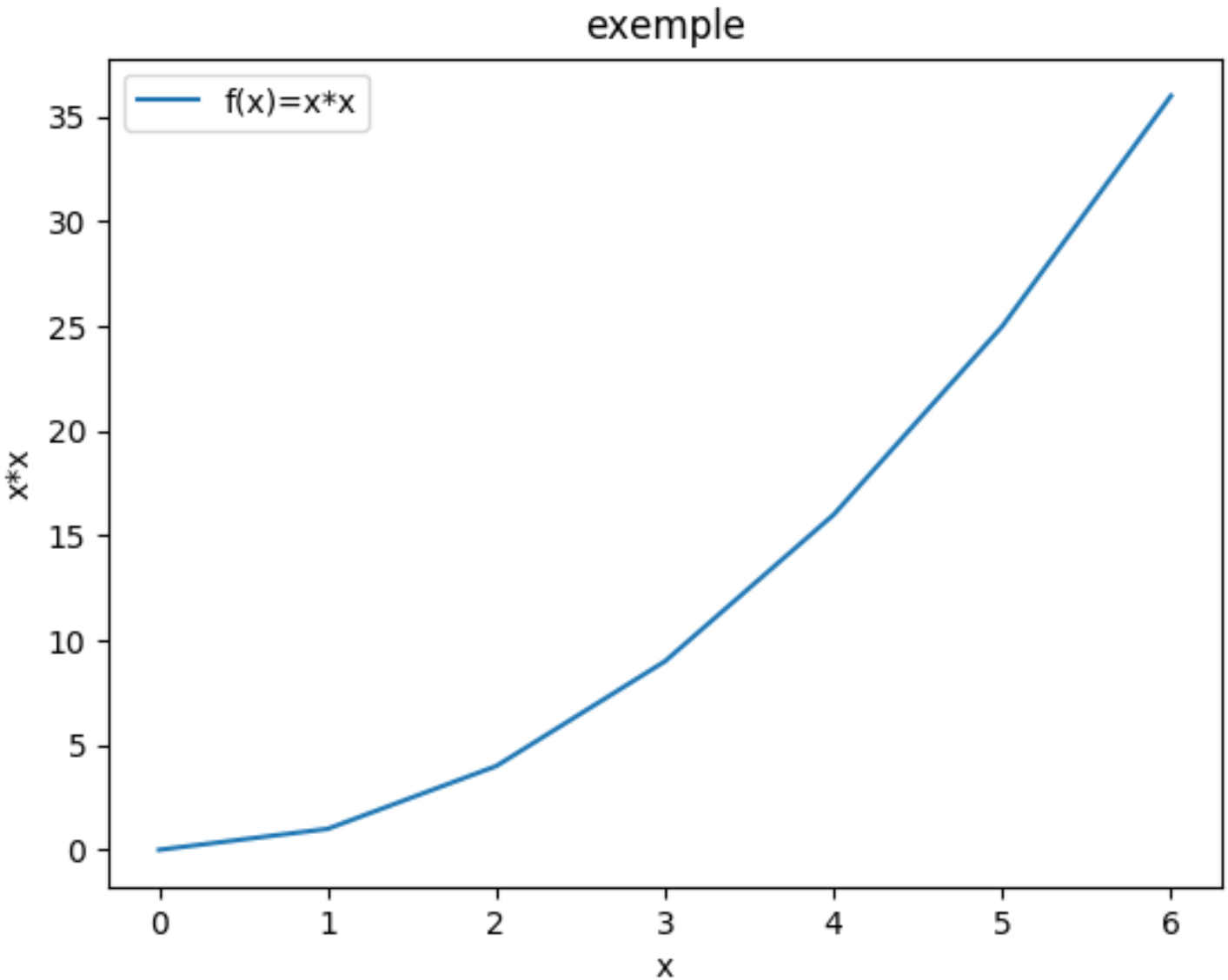
```
import matplotlib.pyplot as plt
plt.figure()
plt.title("exemple")
plt.plot([0,1,2,3,4,5,6],[0,1,4,9,16,25,36])
plt.legend(['f(x)=x*x'])
plt.xlabel('x')
plt.ylabel('x*x')
plt.show()
plt.close()
```

Stateless vs Statefull

Approche *stateless* : approche orienté objet, manipulation direct de la figure et des axes. Plus customisable mais plus compliqué.

```
import matplotlib.pyplot as plt
fig, ax = plt.subplots()
ax.set_title("exemple")
ax.plot([0,1,2,3,4,5,6],[0,1,4,9,16,25,36])
ax.set_ylabel('x*x')
ax.set_xlabel('x')
ax.legend(['f(x)=x*x'])
plt.show()
```

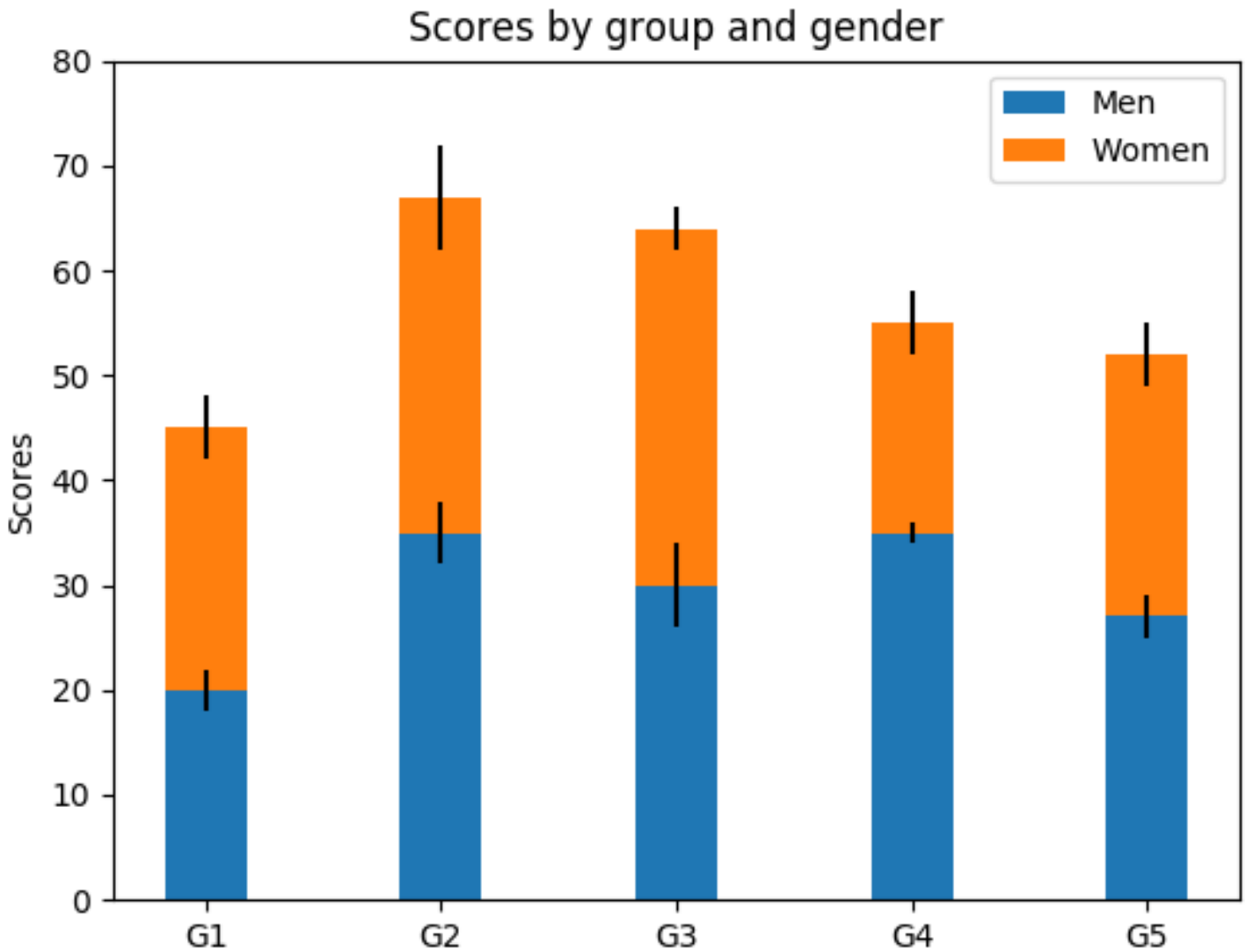
Plot



Barplot

```
import numpy as np
import matplotlib.pyplot as plt
N = 5
menMeans = (20, 35, 30, 35, 27)
womenMeans = (25, 32, 34, 20, 25)
menStd = (2, 3, 4, 1, 2)
womenStd = (3, 5, 2, 3, 3)
ind = np.arange(N)
width = 0.35
p1 = plt.bar(ind, menMeans, width, yerr=menStd)
p2 = plt.bar(ind, womenMeans, width,
             bottom=menMeans, yerr=womenStd)
plt.ylabel('Scores')
plt.title('Scores by group and gender')
plt.xticks(ind, ('G1', 'G2', 'G3', 'G4', 'G5'))
plt.yticks(np.arange(0, 81, 10))
plt.legend((p1[0], p2[0]), ('Men', 'Women'))
plt.show()
```

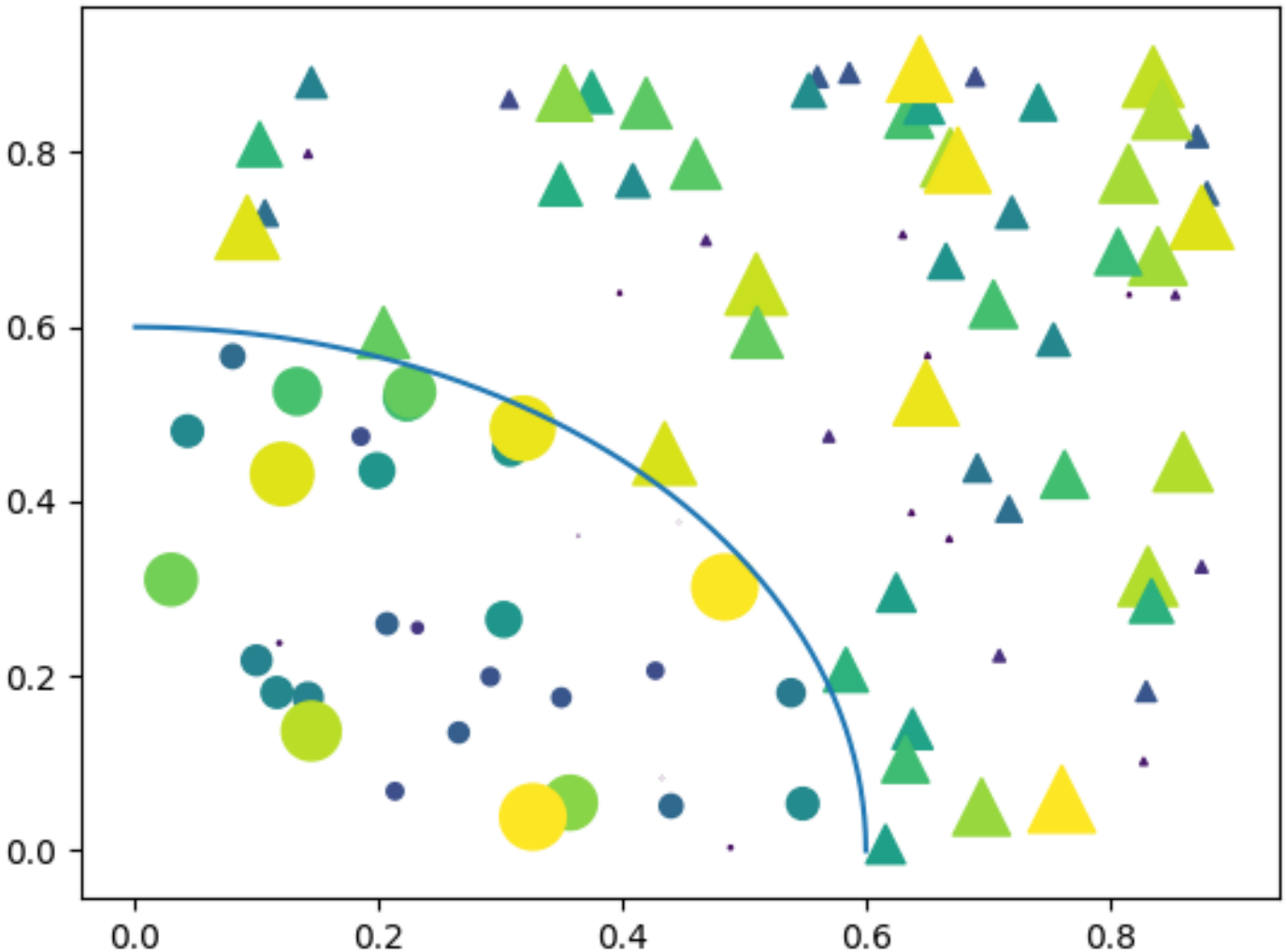
Barplot


Scores by group and gender
Men
Women
Scores
80
70
60
50
40
30
20
10
0
G1
G2
G3
G4
G5

Scatterplot

```
import matplotlib.pyplot as plt
import numpy as np
# Fixing random state for reproducibility
np.random.seed(19680801)
N = 100
r0 = 0.6
x = 0.9 * np.random.rand(N)
y = 0.9 * np.random.rand(N)
area = (20 * np.random.rand(N))**2  # 0 to 10 point radii
c = np.sqrt(area)
r = np.sqrt(x * x + y * y)
area1 = np.ma.masked_where(r < r0, area)
area2 = np.ma.masked_where(r >= r0, area)
plt.scatter(x, y, s=area1, marker='^', c=c)
plt.scatter(x, y, s=area2, marker='o', c=c)
# Show the boundary between the regions:
theta = np.arange(0, np.pi / 2, 0.01)
plt.plot(r0 * np.cos(theta), r0 * np.sin(theta))
plt.show()
```

Scatterplot



Le projet

- Travail en binômes.
- Trouver un/des fichier(s) texte à analyser, trouver des statistiques pertinentes et créer des graphiques.
- Trouver un/des fichier(s) csv Insee à analyser, trouver des statistiques pertinentes et créer des graphiques.
- Faire un site web vitrine pour présenter les graphiques, html et css seulement.
- Présentation du site web à l'oral avec une analyse des graphiques. Les difficultés rencontrées, les solutions.