DS 2

Santé publique France

La mission Cassini-Huygens

Concours CCINP – PSI 2016

1 Introduction

Pour rester proches des préoccupations actuelles, nous proposons dans ce devoir de réaliser une analyse des données gouvernementales relatives aux résultats des tests virologiques COVID-19. Ces données sont librement téléchargeables à l'adresse
`https://www.data.gouv.fr/fr/datasets/donnees-relatives-aux-resultats-des-tests-virologiques-covi`

Objectif • Tracer les courbes de tests ou de cas positifs suivant plusieurs critères.

Les données disponibles sont initialement récupérées sous forme du tableau suivant. Seules les données pour la région « 1 » du 13 mai 2020 sont données en exemple.

Code région	Date de prélèvement	Nb de cas positifs ♀	Nb de cas positif chez les ♂	Nb de prélèvements chez les ♀	Nb de prélèvements chez les ♂	Classe d'âge
`code_regions list[int]`	`liste_jours list[str]`	`nb_cas_pos_f list[int]`	`nb_cas_pos_h list[int]`	`nb_prelev_f list[int]`	`nb_prelev_h list[int]`	`cl_age list[int]`
1	13/05/2020	0	0	0	0	9
1	13/05/2020	0	0	1	0	19
1	13/05/2020	0	0	1	0	29
1	13/05/2020	0	0	5	1	39
1	13/05/2020	0	0	2	1	49
1	13/05/2020	0	0	5	2	59
1	13/05/2020	0	0	1	1	69
1	13/05/2020	0	0	1	4	79
1	13/05/2020	0	0	1	0	89
1	13/05/2020	0	0	0	0	90
1	13/05/2020	0	0	17	9	0

Chacune des données de ce tableau sont stockées dans des listes dont le nom est donné dans la deuxième ligne. On précise aussi dans cette colonne le type de données. Ainsi, par exemple, le nombre cas positifs chez les femmes est donné dans la liste `nb_cas_pos_f = [0,0,0,0,...]`.

La correspondance entre le code de région est donné dans une liste Python nommée `liste_regions=list[int,str]` :

```
liste_regions = [[1, 'Guadeloupe'], [2, 'Martinique'], [3, 'Guyane'],
    [4, 'Réunion'], [6, 'Mayotte'], [11, 'Ile de France'], [24, 'Centre
    Val de Loire'], [27, 'Bourgogne et Franche Comté'], [28, 'Normandie'
    ], [32, 'Hauts de France'], [44, 'Grand Est'], [52, 'Pays de Loire'
    ], [53, 'Bretagne'], [75, 'Nouvelle Aquitaine'], [76, 'Occitanie'],
    [84, 'Auvergne et Rhône-Alpes'], [93, 'Provence-Alpes-Côte dAzur'],
    [94, 'Corse'], [975, 'Saint-Pierre et Miquelon'], [977, 'Saint-
    Barthelemy'], [978, 'Saint-Martin']].
```

2 Tracer du nombre de prélèvements par jour

Pour traiter les données fournies, il est nécessaire de calculer le nombre de prélèvement réalisé chacun des jours.

Question 1 *Donner l'instruction permettant de savoir combien il existe d'éléments dans la liste des jours* `liste_jours`.

Correction
```
>>> len(jours)
```

Question 2 *Écrire la fonction d'en-tête* `def is_test(jours:list, jour:str) -> bool :` *permettant de savoir si le jour* `jour` *est dans la liste des jours* `jours` *et renvoyant un booléen. Vous utiliserez une boucle* `for`.

Correction
```
def is_test(jours:list,jour:str) -> bool :
    for d in dates :
        if d == jour :
            return True
    return False
```

Question 3 *Écrire la fonction d'en-tête* `def is_test(jours:list, jour:str) -> bool :` *permettant de savoir si le jour* `jour` *est dans la liste des jours* `jours` *et renvoyant un booléen. Vous utiliserez une boucle* `while`.

Correction
```
def is_test_while(jours:list,jour:str) -> bool :
    i = 0
    while i<len(jours):
        if jours[i] == jour :
            return True
        i=i+1
    return False
```

Question 4 *Écrire instruction permettant de savoir si des tests ont été faits le 13 mai 2020. Que renvoie cette instruction ?*

Correction
```
>>> is_test(liste_jours,"2020-05-13")
```
Cette instruction renvoie le booléen True.

Question 5 *Écrire la fonction d'en-tête* `def indices_jour(jours:list, jour:str) -> list :` *permettant de renvoyer* `liste_indices = list[int]` *une liste d'indices .*

Correction
```
def indices_jour(jours:list, jour:str) -> list :
    liste_indices = []
    for i in range(len(jours)):
        if jours[i] == jour :
            liste_indices.append(i)
    return liste_indices
```

Question 6 *Écrire la fonction d'en-tête* `def compte_test_jour(jours:list, jour:str, tests:list) -> int :` *permettant de renvoyer le nombres de tests faits le jour* `jour`.

Correction

```
def compte_test_jour(jours:list, jour:str, tests:list) -> int :
    liste_indices = indices_jour(jours, jour)
    nb_tests = 0
    for i in liste_indices :
        nb_tests = nb_tests + tests[i]
    return nb_tests
```

Question 7 *Écrire les instruction permettant de déterminer le nombre de tests réalisés sur les femmes le 5 novembre 2020 et le nombre de tests positifs dénombrés le même jour chez les hommes.*

Correction

```
>>> compte_test_jour(liste_dates,'2020-11-05',nb_prelev_f)
>>> compte_test_jour(liste_dates,'2020-11-05',nb_cas_pos_h)
```

Question 8 *Écrire la fonction d'en-tête* `def compte_test_jour(jours:list, jour:str, tests:list) -> int :` *permettant de renvoyer la liste de nombres des tests faits le jour* `jour`.

Correction

```
def compte_test_jour(jours:list, jour:str, tests:list) -> int :
    liste_indices = indices_jour(jours, jour)
    nb_tests = 0
    for i in liste_indices :
        nb_tests = nb_tests + tests[i]
    return nb_tests
```

Question 9 *Écrire la fonction d'en-tête* `def creer_liste_jours(jours:list) -> list :` *permettant de renvoyer la liste des jours où des tests ont été faits.*

Correction

```
def creer_liste_jours(jours:list) -> list :
    liste_jours = []
    for d in jours :
        if not(d in liste_jours) :
            liste_jours.append(d)
    return liste_jours
```

Question 10 *Écrire la fonction d'en-tête* `def creer_liste_test(dates:list, jours:list) -> list :` *permettant de renvoyer la liste du nombre de tests faits chaque jour.*

Correction

```
def creer_liste_test(dates:list, jours:list) -> list :
    liste_jours = []
    for d in jours :
        if not(d in liste_jours) :
            liste_jours.append(d)
    return liste_jours
```

Question 11 *Écrire les instructions permettant d'obtenir la liste du nombre de tests réalisés par les femmes chaque jour ainsi que le nombre de cas positifs chez les femmes chaque jour.*

Correction

```
>>> l_test_f = creer_liste_test(liste_jours,nb_prelev_f)
>>> l_pos_f = creer_liste_test(liste_jours,nb_cas_pos_f)
```

Question 12 *Écrire la fonction d'en-tête* `def nb_tests_jour(dates:list, test1:list, test2:list) -> list :` *permettant de renvoyer la liste du nombre de tests faits chaque jour.*

Correction

```
def nb_tests_jour(dates:list, tests1:list, tests2:list) -> list :
    nb_tests_h = nb_tests_jour_sexe(dates,tests1)
    nb_tests_f = nb_tests_jour_sexe(dates,tests2)

    nb_tests = [nb_tests_h[i][1]+nb_tests_f[i][1] for i in range(len(
       nb_tests_h))]
    return nb_tests
```

Question 13 *Donner les instructions permettant de charger les bibliothèques nécessaire au tracer de graphes.*

Correction

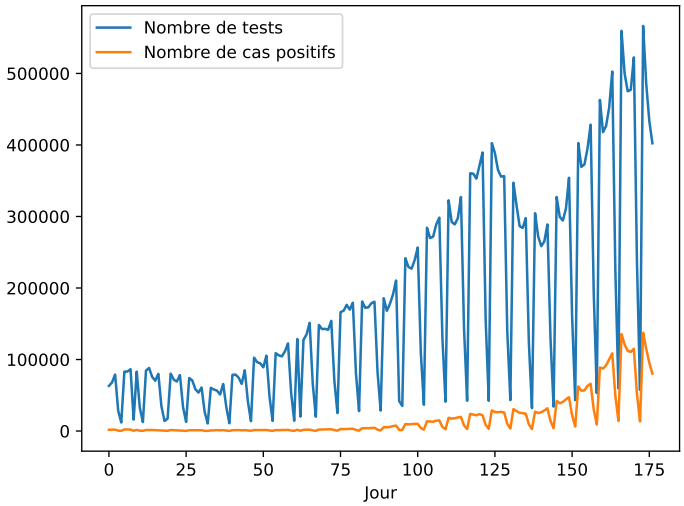
```
>>> import numpy as np
>>> import matplotlib.pyplot as plt
```

Question 14 *Donner les instructions permettant de tracer les courbes suivantes : nombre de tests en fonction du jour, nombre de cas positifs en fonction du jour.*

Correction

```
plt.plot(nb_test,label="Nombre de tests")
plt.plot(nb_pos,label="Nombre de cas positifs")
plt.xlabel('Jour')
plt.legend()
plt.show()
```

Le résultat obtenu est le suivant.



3 Traitement des mesures

On dispose des listes suivantes : `nb_tests = list[int]` la liste du nombre de prélèvement par jour et de `nb_pos = list[int]` la liste du nombre de tests positifs par jour ainsi que `liste_jours = list[int]`

Question 15 *Écrire la fonction d'en-tête* `def recherche_max(liste:list) -> int :` *permettant de renvoyer le maximum d'une liste. On pourrait ainsi avoir le maximum de tests réalisés en une journée, ou le nombre maximum de cas.* ***On ne pourra pas utiliser la fonction max.***

Correction

Afin de lisser les courbes réalisées précédemment, on se propose de réaliser un lissage sur n jours. Pour cela on fait la moyenne des cas du jour 0 au jour n, puis du jour 1 au jour $n+1$ *etc.* Cette méthode se nomme moyenne glissante.

Question 16 *Compléter la fonction d'en-tête* `def moyenne_glissante(tests:list, nb:int) -> list :` *permettant de réaliser la moyenne glissante sur n jours.*

```
def moyenne_glissante(tests,nb):
    res = []
    for i in range(     à compléter     ):
        s = 0
        for j in range(     à compléter     ):
            s = s+tests[j]
        res.append(     à compléter     )
    return res
```

Correction

```
def moyenne_glissante(tests,nb):
    res = []
    for i in range(len(tests)-nb):
        s = 0
        for j in range(i,i+nb):
            s = s+tests[j]
        res.append(s/nb)
    return res
```

On souhaiterait savoir s'il y a déjà eu un jour où n tests ont été faits. *Ok, cette question est artificielle, mais on voulait faire une question sur la dichotomie :)).*

On souhaite écrire la fonction d'en-tête `def recherche_dicho(tests:list, nb:int) -> bool :` permettant de savoir si un nombre de test nb a déjà été fait.

Question 17 *Quelle condition doit-il y avoir sur la liste* `tests` *pour pouvoir faire une recherche dichotomique ?*

Correction

Question 18 *Écrire la fonction d'en-tête* `def recherche_dicho(tests:list, nb:int) -> bool :` *permettant de savoir si un nombre de test nb a déjà été fait. On utilisera donc une recherche par dichotomie.*

Correction