Introduction à l'apprentissage par renforcement

TP 1 - les manchots multi-bras

1/4 de la note finale est liée à la mise en forme :

- pensez à nettoyer les outputs inutiles (installation, messages de débuggage, ...)
- soignez vos figures : les axes sont-ils faciles à comprendre ? L'échelle est adaptée ?
- commentez vos résultats : vous attendiez-vous à les avoir ? Est-ce étonnant ? Faites le lien avec la théorie.

Ce TP reprend l'exemple d'un médecin et de ses vaccins. Vous allez comparer plusieurs stratégies et trouver celle optimale. Un TP se fait en groupe de 2 à 4. Aucun groupe de plus de 4 personnes.

Vous allez rendre le TP dans une archive ZIP. L'archive ZIP contient ce notebook au format `ipynb`, mais aussi exporté en PDF & HTML. L'archive ZIP doit aussi contenir un fichier txt appelé `groupe.txt` sous le format:

```
Nom1, Prenom1, Email1, NumEtudiant1
Nom2, Prenom2, Email2, NumEtudiant2
Nom3, Prenom3, Email3, NumEtudiant3
Nom4, Prenom4, Email4, NumEtudiant4
```

Un script vient extraire vos réponses : ne changez pas l'ordre des cellules et soyez sûrs que les graphes sont bien présents dans la version notebook soumise.

```
! pip install matplotlib tqdm numpy ipympl opencv-python torch tqdm
!jupyter labextension install @jupyter-widgets/jupyterlab-manager
!jupyter labextension install jupyter-matplotlib
```

```
Requirement already satisfied: matplotlib in
./.venv/lib/python3.10/site-packages (3.9.2)
Requirement already satisfied: tqdm in ./.venv/lib/python3.10/site-
packages (4.66.5)
Requirement already satisfied: numpy in ./.venv/lib/python3.10/site-
packages (2.1.1)
Requirement already satisfied: ipympl in ./.venv/lib/python3.10/site-
packages (0.9.4)
Requirement already satisfied: opencv-python in
./.venv/lib/python3.10/site-packages (4.10.0.84)
Requirement already satisfied: torch in ./.venv/lib/python3.10/site-
packages (2.4.1)
Requirement already satisfied: contourpy>=1.0.1 in
./.venv/lib/python3.10/site-packages (from matplotlib) (1.3.0)
```

```
Requirement already satisfied: pillow>=8 in
./.venv/lib/python3.10/site-packages (from matplotlib) (10.4.0)
Requirement already satisfied: packaging>=20.0 in
./.venv/lib/python3.10/site-packages (from matplotlib) (24.1)
Requirement already satisfied: pyparsing>=2.3.1 in
./.venv/lib/python3.10/site-packages (from matplotlib) (3.1.4)
Requirement already satisfied: kiwisolver>=1.3.1 in
./.venv/lib/python3.10/site-packages (from matplotlib) (1.4.7)
Requirement already satisfied: fonttools>=4.22.0 in
./.venv/lib/python3.10/site-packages (from matplotlib) (4.53.1)
Requirement already satisfied: python-dateutil>=2.7 in
./.venv/lib/python3.10/site-packages (from matplotlib) (2.9.0.post0)
Requirement already satisfied: cycler>=0.10 in
./.venv/lib/python3.10/site-packages (from matplotlib) (0.12.1)
Requirement already satisfied: ipython<9 in
./.venv/lib/python3.10/site-packages (from ipympl) (8.27.0)
Requirement already satisfied: ipython-genutils in
./.venv/lib/python3.10/site-packages (from ipympl) (0.2.0)
Requirement already satisfied: traitlets<6 in
./.venv/lib/python3.10/site-packages (from ipympl) (5.14.3)
Requirement already satisfied: ipywidgets<9,>=7.6.0 in
./.venv/lib/python3.10/site-packages (from ipympl) (8.1.5)
Requirement already satisfied: typing-extensions>=4.8.0 in
./.venv/lib/python3.10/site-packages (from torch) (4.12.2)
Requirement already satisfied: nvidia-cuda-cupti-cu12==12.1.105
in ./.venv/lib/python3.10/site-packages (from torch) (12.1.105)
Requirement already satisfied: nvidia-cusparse-cu12==12.1.0.106
in ./.venv/lib/python3.10/site-packages (from torch) (12.1.0.106)
Requirement already satisfied: fsspec in ./.venv/lib/python3.10/site-
packages (from torch) (2024.9.0)
Requirement already satisfied: nvidia-cufft-cu12==11.0.2.54
in ./.venv/lib/python3.10/site-packages (from torch) (11.0.2.54)
Requirement already satisfied: nvidia-cuda-runtime-cu12==12.1.105
in ./.venv/lib/python3.10/site-packages (from torch) (12.1.105)
Requirement already satisfied: nvidia-curand-cu12==10.3.2.106
in ./.venv/lib/python3.10/site-packages (from torch) (10.3.2.106)
Requirement already satisfied: networkx in
./.venv/lib/python3.10/site-packages (from torch) (3.3)
Requirement already satisfied: nvidia-cublas-cu12==12.1.3.1
in ./.venv/lib/python3.10/site-packages (from torch) (12.1.3.1)
Requirement already satisfied: filelock in
./.venv/lib/python3.10/site-packages (from torch) (3.16.0)
Requirement already satisfied: nvidia-cusolver-cu12==11.4.5.107
in ./.venv/lib/python3.10/site-packages (from torch) (11.4.5.107)
Requirement already satisfied: sympy in ./.venv/lib/python3.10/site-
packages (from torch) (1.13.2)
Requirement already satisfied: nvidia-nccl-cu12==2.20.5 in
./.venv/lib/python3.10/site-packages (from torch) (2.20.5)
Requirement already satisfied: nvidia-cuda-nvrtc-cu12==12.1.105
```

```
in ./.venv/lib/python3.10/site-packages (from torch) (12.1.105)
Requirement already satisfied: nvidia-nvtx-cu12==12.1.105 in
./.venv/lib/python3.10/site-packages (from torch) (12.1.105)
Requirement already satisfied: triton==3.0.0 in
./.venv/lib/python3.10/site-packages (from torch) (3.0.0)
Requirement already satisfied: nvidia-cudnn-cu12==9.1.0.70
in ./.venv/lib/python3.10/site-packages (from torch) (9.1.0.70)
Requirement already satisfied: jinja2 in ./.venv/lib/python3.10/site-
packages (from torch) (3.1.4)
Requirement already satisfied: nvidia-nvjitlink-cu12 in
./.venv/lib/python3.10/site-packages (from nvidia-cusolver-
cu12==11.4.5.107->torch) (12.6.68)
Requirement already satisfied: jedi>=0.16 in
./.venv/lib/python3.10/site-packages (from ipython<9->ipympl) (0.19.1)
Requirement already satisfied: pygments>=2.4.0 in
./.venv/lib/python3.10/site-packages (from ipython<9->ipympl) (2.18.0)
Requirement already satisfied: matplotlib-inline in
./.venv/lib/python3.10/site-packages (from ipython<9->ipympl) (0.1.7)
Requirement already satisfied: stack-data in
./.venv/lib/python3.10/site-packages (from ipython<9->ipympl) (0.6.3)
Requirement already satisfied: exceptiongroup in
./.venv/lib/python3.10/site-packages (from ipython<9->ipympl) (1.2.2)
Requirement already satisfied: decorator in
./.venv/lib/python3.10/site-packages (from ipython<9->ipympl) (5.1.1)
Requirement already satisfied: pexpect>4.3 in
./.venv/lib/python3.10/site-packages (from ipython<9->ipympl) (4.9.0)
Requirement already satisfied: prompt-toolkit<3.1.0,>=3.0.41
in ./.venv/lib/python3.10/site-packages (from ipython<9->ipympl)
(3.0.47)
Requirement already satisfied: widgetsnbextension~=4.0.12 in
./.venv/lib/python3.10/site-packages (from ipywidgets<9,>=7.6.0-
>ipympl) (4.0.13)
Requirement already satisfied: comm>=0.1.3 in
./.venv/lib/python3.10/site-packages (from ipywidgets<9,>=7.6.0-
>ipympl) (0.2.2)
Requirement already satisfied: jupyterlab-widgets~=3.0.12 in
./.venv/lib/python3.10/site-packages (from ipywidgets<9,>=7.6.0-
>ipympl) (3.0.13)
Requirement already satisfied: six>=1.5 in
./.venv/lib/python3.10/site-packages (from python-dateutil>=2.7-
>matplotlib) (1.16.0)
Requirement already satisfied: MarkupSafe>=2.0 in
./.venv/lib/python3.10/site-packages (from jinja2->torch) (2.1.5)
Requirement already satisfied: mpmath<1.4,>=1.1.0 in
./.venv/lib/python3.10/site-packages (from sympy->torch) (1.3.0)
Requirement already satisfied: parso<0.9.0,>=0.8.3 in
./.venv/lib/python3.10/site-packages (from jedi>=0.16->ipython<9-
>ipympl) (0.8.4)
Requirement already satisfied: ptyprocess>=0.5 in
```

```
./.venv/lib/python3.10/site-packages (from pexpect>4.3->ipython<9-
>ipympl) (0.7.0)
Requirement already satisfied: wcwidth in ./.venv/lib/python3.10/site-
packages (from prompt-toolkit<3.1.0,>=3.0.41->ipython<9->ipympl)
(0.2.13)
Requirement already satisfied: asttokens>=2.1.0 in
./.venv/lib/python3.10/site-packages (from stack-data->ipython<9-
>ipympl) (2.4.1)
Requirement already satisfied: executing>=1.2.0 in
./.venv/lib/python3.10/site-packages (from stack-data->ipython<9-
>ipympl) (2.1.0)
Requirement already satisfied: pure-eval in
./.venv/lib/python3.10/site-packages (from stack-data->ipython<9-
>ipympl) (0.2.3)

[notice] A new release of pip is available: 23.0.1 -> 24.2
[notice] To update, run: pip install --upgrade pip
usage: jupyter [-h] [--version] [--config-dir] [--data-dir] [--
runtime-dir]
               [--paths] [--json] [--debug]
               [subcommand]

Jupyter: Interactive Computing

positional arguments:
  subcommand     the subcommand to launch

options:
  -h, --help     show this help message and exit
  --version      show the versions of core jupyter packages and exit
  --config-dir   show Jupyter config dir
  --data-dir     show Jupyter data dir
  --runtime-dir  show Jupyter runtime dir
  --paths        show all Jupyter paths. Add --json for machine-
readable
                 format.
  --json         output paths as machine-readable json
  --debug        output debug information about paths

Available subcommands: kernel kernelspec migrate run troubleshoot

Jupyter command `jupyter-labextension` not found.
usage: jupyter [-h] [--version] [--config-dir] [--data-dir] [--
runtime-dir]
               [--paths] [--json] [--debug]
               [subcommand]

Jupyter: Interactive Computing

positional arguments:
  subcommand     the subcommand to launch

options:
  -h, --help     show this help message and exit
  --version      show the versions of core jupyter packages and exit
  --config-dir   show Jupyter config dir
  --data-dir     show Jupyter data dir
  --runtime-dir  show Jupyter runtime dir
  --paths        show all Jupyter paths. Add --json for machine-
readable
                 format.
  --json         output paths as machine-readable json
  --debug        output debug information about paths

Available subcommands: kernel kernelspec migrate run troubleshoot

Jupyter command `jupyter-labextension` not found.
```

```
%load_ext autoreload
%autoreload 2
%matplotlib inline

from tqdm import tqdm
from typing import List, Tuple
import typing as t
import math
import torch
import numpy as np
from tqdm.auto import trange, tqdm
import matplotlib.pyplot as plt
from matplotlib.animation import FuncAnimation
import matplotlib.animation as animation
from matplotlib.backends.backend_agg import FigureCanvasAgg as
FigureCanvas
import cv2
from IPython.display import display, clear_output
import random
import pandas as pd



torch.random.manual_seed(0)

K = 5 # num arms
```

Présentation du problème

```
import torch.testing
```

```python
class ArmBernoulli:
    def __init__(self, p: float):
        """
        Vaccine treatment following a Bernoulli law (mean is p and variance is p(1-p)
        Args:
             p (float): mean parameter

        >>> torch.random.manual_seed(random_state)
        >>> arm = ArmBernoulli(0.5)
        >>> arm.sample(5)
        tensor([ True, False,  True,  True,  True])
        """
        self.immunity_rate = p

    def sample(self, n: int = 1) -> torch.Tensor:
        return torch.rand(n) < self.immunity_rate

    def __repr__(self):
        return f'<ArmBernoulli p={self.immunity_rate}'

def generate_arms(num_arms: int):
    means = torch.rand(num_arms)
    # print("means =", means)
    MAB = [ArmBernoulli(m) for m in means]
    assert MAB[0].immunity_rate == means[0]
    assert (MAB[0].sample(10) <= 1).all() and (MAB[0].sample(10) >= 0).all()
    return MAB

MAB = generate_arms(K)
```

Ce TP reprend l'exemple du médecin présenté en cours.

Q1. Créez une fonction pour trouver $\mu^{¿}$ à partir d'un `MAB`. Comment est définie la récompense R_k ? Que représente concrètement le regret dans cet exemple ?

```python
def find_mean_reward_from_best_vaccin(MAB : List[ArmBernoulli], hide_immunity_rate = False) -> float:

    N = 1000
    means : List[float]

    if hide_immunity_rate:
        means = [torch.mean(arm.sample(N), dtype=float) for arm in MAB]
    else:
        means = [arm.immunity_rate for arm in MAB]
    print("means =", means)
    mu_star : float = max(means).item()
    print("mu_star =", mu_star)
    return mu_star

def apply_vaccin(arm : ArmBernoulli) -> int:
    """Rk

    Returns 1 if the patient is immunised thanks to the vaccin and 0
if not"""
    return 1 if arm.sample(1).item() else 0

def compute_regret(MAB : List[ArmBernoulli], N : int) -> float:
    mu_star = find_mean_reward_from_best_vaccin(MAB, False)
    # rn = N * mu_star -

find_mean_reward_from_best_vaccin(MAB)
```

```
means = [tensor(0.4963), tensor(0.7682), tensor(0.0885),
tensor(0.1320), tensor(0.3074)]
mu_star = 0.7682217955589294

0.7682217955589294
```

[Ajoutez votre commentaire ici]

Que représente concrètement le regret dans cet exemple ?

[Réponse] Le regret représente le nombre de patient qui aurait pu être immunisé si le meilleur avait été utilisé à chaque fois.

Note importante : pour la suite, les tests seront faits avec 10 MAB différents ou plus pour réduire le bruit de simulation. Concrètement, on exécutera au moins 10x `generate_arms`.

I. Cas classique des bandits manchots

I.a. Solution Gloutonne

Le médecin fonctionne sur deux phases :

1. **Exploration :** N patients reçoivent une dose d'un vaccin choisi aléatoirement. Le médecin calcule le taux d'immunisation empirique :

$$\acute{R}_i = \frac{1}{T_i} \sum_{k=0}^{N-1} \chi_{v_k,i} R_k,$$

avec $T_i = \sum_{k=0}^{N-1} \chi_{v_k,i}$.

1. **Exploitation :** Le vaccin $v_i = arg\,max_j \acute{R}_j$ est utilisé pour les M patients suivants. C'est la phase de test.

Q2. Implémentez la solution gloutonne avec N = 50 et M = 500 et testez la avec 100 MAB différents (tous ont 5 vaccins). On s'intéresse à la variable aléatoire "la phase d'exploration a trouvé le bon vaccin". Quelle est l'espérance empirique de cette variable ? Et son écart-type ? Calculez de même l'espérance et l'écart-type du regret sur vos 100 simulations.

Pour rappel, le regret est défini par :

$$r_n = n\mu^{¿} - \sum_{k=0}^{n-1} R_k$$

Attention : n est le nombre total de patients, donc ici $N+M$.

```python
MABs = [generate_arms(5) for _ in range(100)]

N = 50
M = 500
log = print if False else (lambda *_, **_2 : None)
C = 2

def compute_empirical_immuniation_rate(vaccins_used : torch.Tensor, Rk : torch.Tensor, nb_vaccin : int) -> torch.Tensor:

    n_patients : int = vaccins_used.shape[0]

    vaccins_used = vaccins_used.to(torch.int)
    log("vaccins_used =", vaccins_used)

    log("Rk =", Rk)
    log("Rk shape =", Rk.shape)

    n_patient = vaccins_used.shape[0]
    assert(Rk.shape[0] == n_patient)

    # compute the number of times each vaccin has been used
    T = torch.bincount(vaccins_used, minlength=nb_vaccin)
    log("T =", T)

    # Compute the indicator function
    X = torch.zeros((n_patients, nb_vaccin))
    X[torch.arange(n_patients), vaccins_used] = 1
    log("X shape =", X.shape)
    # log("X :", X)

    Ris = torch.Tensor([1 / T[i] * (X[:,i] @ Rk) for i in range(nb_vaccin)]).flatten()
    log(Ris)

    # compute with bias
    bias = torch.sqrt(C * (math.log(n_patients) if n_patients > 0 else 0) / T)
    Ris_with_bias = Ris + bias
    log(Ris_with_bias)

    return Ris, Ris_with_bias


def exploration(MAB : List[ArmBernoulli], n_patients : int) -> tuple[torch.tensor, torch.tensor]:

    if n_patients == 0:
        return torch.Tensor(), torch.Tensor()

    nb_vaccin = len(MAB)

    log("Arms immunity rate :", [arm.immunity_rate for arm in MAB])

    # Vaccin used for all patients
    vaccins_used = torch.randint(0, nb_vaccin, (n_patients,))

    # apply vaccins
    Rk = torch.cat([MAB[vaccins_used[k]].sample(1) for k in range(n_patients)]).flatten().to(torch.float)

    return vaccins_used, Rk

def compute_regret(n_success_exploration_vaccin : int, MAB : List[ArmBernoulli], real_best_vaccin : int, n_patients : int) -> float:

    max_vaccin = (n_patients * MAB[real_best_vaccin].immunity_rate).item()
    log("max_vaccin :", max_vaccin)

    return max_vaccin - n_success_exploration_vaccin

def exploitation_no_update(MAB : List[ArmBernoulli], exploration_best_vaccin : int, n_patients : int) -> int:
    """
    Returns:
        float: Number of patients that have been immune by a vaccin
    """

    # apply vaccin in exploitation
    n_success_exploration_vaccin = MAB[exploration_best_vaccin].sample(n_patients).sum().item()
    log("exploration_vaccin :", n_success_exploration_vaccin)

    return n_success_exploration_vaccin
```

```python
def simulation_no_update(MAB : List[ArmBernoulli],
n_patients_exploration : int, n_patients_exploitation : int) ->
Tuple[bool, float]:
    """
    Returns:
        Tuple[bool, float]: Is vaccin found by exploration then TRUE |
Regret of the exploitation
    """

    vaccins_used, Rk = exploration(MAB, n_patients_exploration)

    Ris, _ =
compute_empirical_immuniation_rate(vaccins_used=vaccins_used, Rk=Rk,
nb_vaccin=len(MAB))
    exploration_best_vaccin = Ris.argmax()
    log("exploration best vaccin :", exploration_best_vaccin)

    real_best_vaccin = np.array([arm.immunity_rate for arm in
MAB]).argmax()
    log("real best vaccin :", real_best_vaccin)

    n_success_exploration_vaccin = exploitation_no_update(MAB,
exploration_best_vaccin, n_patients=n_patients_exploitation)

    regret = compute_regret(n_success_exploration_vaccin, MAB,
real_best_vaccin, n_patients_exploitation)
    log("Regret :", regret)

    return real_best_vaccin == exploration_best_vaccin, regret

def print_results(samples_exploration_found_good_vaccin :
torch.Tensor, exploitation_regrets : torch.Tensor) -> None:

    if samples_exploration_found_good_vaccin is not None:
        print("E_exploration_found_good_vaccin :",
samples_exploration_found_good_vaccin.mean())
        print("std_exploration_found_good_vaccin :",
samples_exploration_found_good_vaccin.std())

    print("E_exploitation_regrets :", exploitation_regrets.mean())
    std_exploitation_regrets = exploitation_regrets.std()
    print("std_exploitation_regrets :", std_exploitation_regrets)
    print("std_exploitation_regrets rate :", std_exploitation_regrets
/ M)

if False: # test with one simulation
    i = torch.randint(0, 100, (1, )).item()
    print("index :", i)
    simulation(MABs[i], N, M)
```

```
if True: # test with all the simulations

    l = torch.Tensor([simulation_no_update(MAB, N, M) for MAB in
tqdm(MABs)])

    samples_exploration_found_good_vaccin = l[:, 0]
    exploitation_regrets = l[:, 1]

    print_results(samples_exploration_found_good_vaccin,
exploitation_regrets)
```

{"model_id":"fb5c6440d672465db6d4cbae152ec55d","version_major":2,"vers
ion_minor":0}

```
E_exploration_found_good_vaccin : tensor(0.6600)
std_exploration_found_good_vaccin : tensor(0.4761)
E_exploitation_regrets : tensor(18.6421)
std_exploitation_regrets : tensor(37.3139)
std_exploitation_regrets rate : tensor(0.0746)
```

Q3. On propose d'améliorer l'algorithme précédant en mettant à jour les taux d'immunisation empiriques $\acute{R}_i$ pendant l'exploitation. Notez vous une amélioration du regret ? Proposez un exemple dans lequel cette mise à jour ne changera rien.

```
def exploitation_update_immune_rate(MAB : List[ArmBernoulli],
vaccins_used : torch.Tensor, Rk : torch.Tensor, n_patients : int,
real_best_vaccin : int, add_bias : bool, proba_take_best_vaccin :
float) -> Tuple[torch.Tensor, torch.Tensor, torch.Tensor]:
    """
    Returns:
        Tuple[torch.Tensor, torch.Tensor, torch.Tensor]: The immune
rate for all vaccin at each iteration, same with bias, the regret at
each iteration
    """

    sum = 0
    regrets = []
    list_Ris = []
    list_Ris_with_bias = []
    nb_vaccin = len(MAB)

    for k in range(n_patients):

        # compute immunition rates
        Ris, Ris_with_bias =
compute_empirical_immuniation_rate(vaccins_used=vaccins_used, Rk=Rk,
nb_vaccin=len(MAB))
        list_Ris.append(Ris)
        list_Ris_with_bias.append(Ris_with_bias)
        # Choose the strategy
        if torch.rand(size=(1,)).item() < proba_take_best_vaccin: #
Choose the vaccin considered as the best one
            vaccin_to_apply = (Ris_with_bias if add_bias else
Ris).argmax().to(torch.int)
            log("exploration best vaccin :", vaccin_to_apply)
        else: # choose a random vaccin
            vaccin_to_apply = torch.randint(low=0, high=nb_vaccin,
size=(1, )).item()
            log("random vaccin :", vaccin_to_apply)

        # apply vaccin
        success = MAB[vaccin_to_apply].sample(1)
        sum += success.item()

        # update vaccins_used and Rk
        vaccins_used = torch.cat((vaccins_used,
torch.Tensor([vaccin_to_apply])))
        Rk = torch.cat((Rk, success))

        # compute regret
        regret = compute_regret(sum, MAB, real_best_vaccin, k + 1)
        regrets.append(regret)

    return torch.stack(list_Ris), torch.stack(list_Ris_with_bias),
torch.Tensor(regrets)

def simulation_update_immune_rate(MAB : List[ArmBernoulli],
n_patients_exploration : int, n_patients_exploitation : int,
add_bias : bool, proba_take_best_vaccin : float) ->
Tuple[torch.Tensor, torch.Tensor, torch.Tensor]:
    """
    Returns:
        Tuple[torch.Tensor, torch.Tensor, torch.Tensor]: The immune
rate for all vaccin at each iteration, , the regret at each iteration
    """

    # exploration
    vaccins_used, Rk = exploration(MAB, n_patients_exploration)

    # get the real best vaccin
    real_best_vaccin = np.array([arm.immunity_rate for arm in
MAB]).argmax()
    log("real best vaccin :", real_best_vaccin)

    # exploitation
    return exploitation_update_immune_rate(MAB, vaccins_used, Rk,
n_patients_exploitation, real_best_vaccin, add_bias,
proba_take_best_vaccin=proba_take_best_vaccin)
```

```
# test with all the simulations
last_regrets = torch.Tensor(
    [
        simulation_update_immune_rate(MAB, add_bias=False,
proba_take_best_vaccin=1, n_patients_exploration=N,
n_patients_exploitation=M)[2][-1]
        for MAB in tqdm(MABs)
    ]
)

print_results(None, last_regrets)
```

{"model_id":"20b7d19be904412593e5f0203b96674a","version_major":2,"version_minor":0}

```
E_exploitation_regrets : tensor(5.1721)
std_exploitation_regrets : tensor(17.5161)
std_exploitation_regrets rate : tensor(0.0350)
```

[Ajoutez votre commentaire ici]

- *Notez vous une amélioration du regret ?*

Oui, nous observons une nette observation du regret.

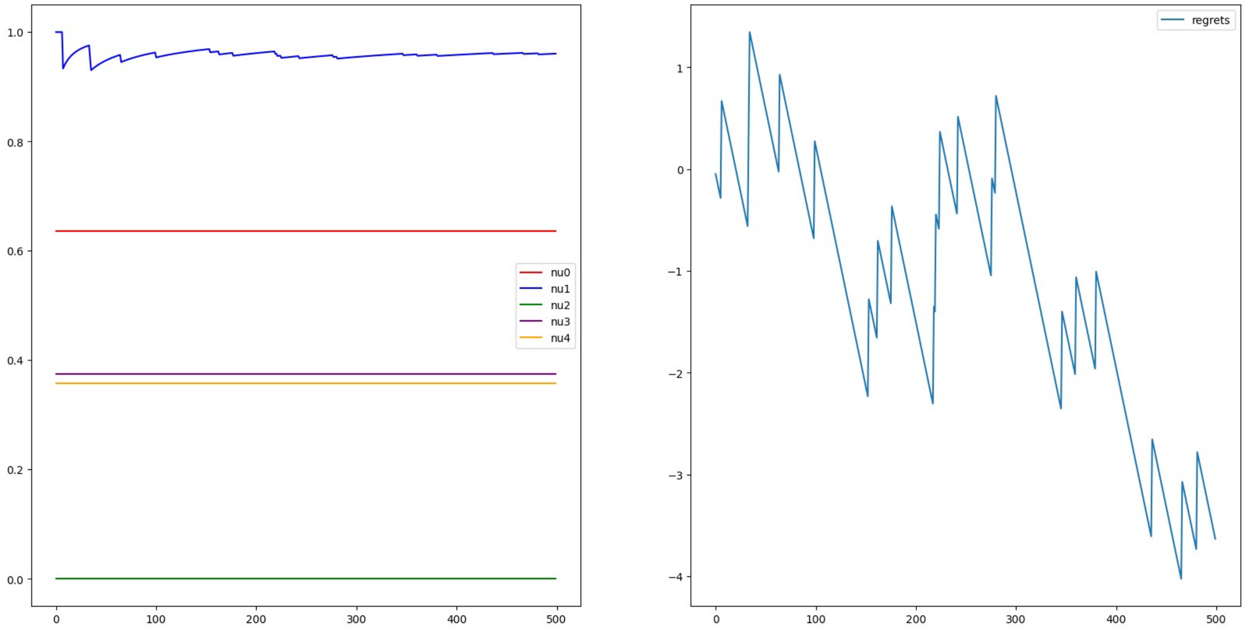
- *Proposez un exemple dans lequel cette mise à jour ne changera rien.*

Cette mise à jour ne sert à rien lorsque le meilleur vaccin a été trouvé lors de l'exploration et que mettre à jour les taux d'immunsation des vaccins ne fait pas changer le vaccin qui est considéré comme le plus performant.

Q4. Créez une figure contenant deux sous-figures : à gauche, le taux d'immunisation empirique $\acute{R}_i$ pour les 5 vaccins ; à droite, le regret r_n. La figure sera animée avec les patients : chaque frame k de l'animation représente le vaccin que l'on donne au k-ième patient.

```
def plot_a_simulation(MAB : List[ArmBernoulli], add_bias : bool =
False, proba_take_best_vaccin : float = 1, n_patients_exploration :
int = N, n_patients_exploitation : int = M) -> None:

    print("Immunity rates :", [round(arm.immunity_rate.item(), 3) for
arm in MAB])

    list_Ris, list_Ris_with_bias, regrets =
simulation_update_immune_rate(MAB, n_patients_exploration,
n_patients_exploitation, add_bias,
proba_take_best_vaccin=proba_take_best_vaccin)

    # store Tensor in a dataframe
    df_Ris = pd.DataFrame(list_Ris, columns=[f"nu{i}" for i in
range(len(MAB))])
    df_Ris_with_bias = pd.DataFrame(list_Ris_with_bias,
columns=[f"nu{i}_bias" for i in range(len(MAB))])
    df_regrets = pd.DataFrame(regrets, columns=["regrets"])

    _, axes = plt.subplots(1, 2, figsize=(20, 10))
    colors=["red", "blue", "green", "purple", "orange"] if len(MAB) ==
5 else None
    df_Ris.plot(kind="line", ax=axes[0], color=colors)
    if add_bias:
        df_Ris_with_bias.plot(kind="line", ax=axes[0], linestyle='--',
color=colors)
    df_regrets.plot(kind="line", ax=axes[1])

# The vaccin used does not changes because the best vaccin is way much
too good.
plot_a_simulation(MAB=MABs[0], add_bias=False,
proba_take_best_vaccin=1)
```

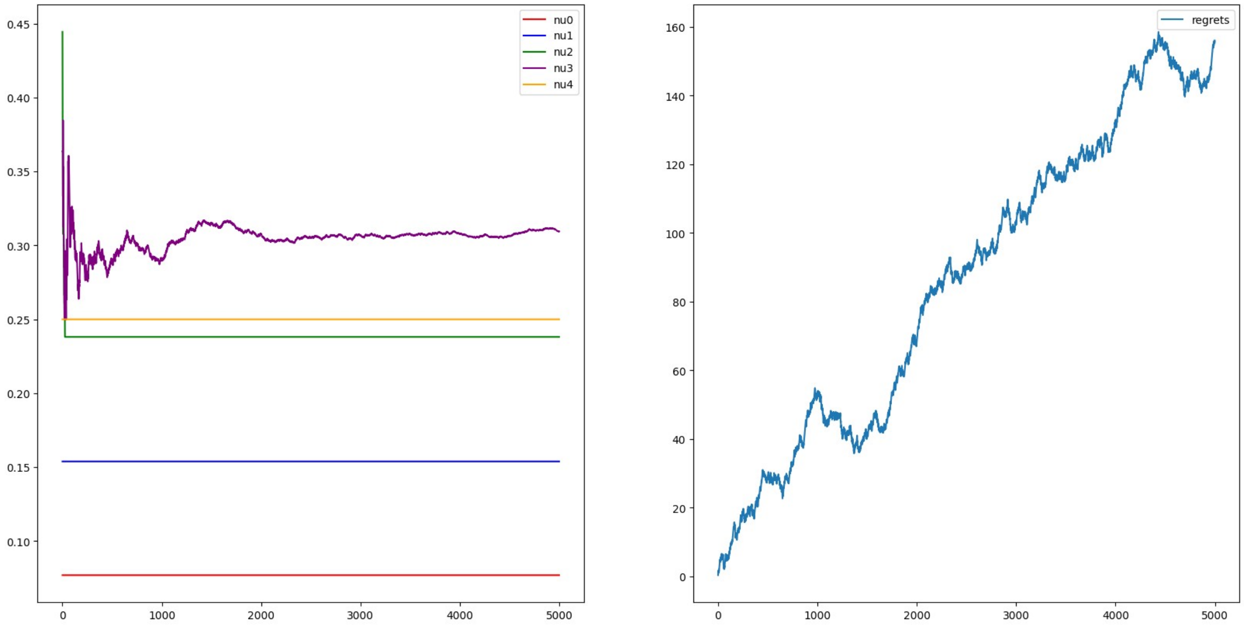
```
Immunity rates : [0.553, 0.953, 0.036, 0.185, 0.373]
```



```
def generate_custom_arms(min_rate : float, diff_rate : float, n_arms :
int = 5):
    MAB = generate_arms(n_arms)
    for k, arm in enumerate(MAB):
        arm.immunity_rate = torch.Tensor([min_rate + diff_rate * k])

    return MAB

# The immunity rates are very close so each run change a lot the
curves because 50 patients for the exploration is too low.
MAB = generate_custom_arms(0.3, 0.01)
```

```
plot_a_simulation(MAB=MAB, add_bias=False, proba_take_best_vaccin=1,
n_patients_exploitation=5000)
```

```
Immunity rates : [0.3, 0.31, 0.32, 0.33, 0.34]
```



[Ajoutez votre commentaire ici]

When the best vaccin is found, the regret does not grow too much. When it is not found, the regret grows infinitely.

Q5. On étudie maintenant l'influence de la taille du training set N. On considère que N+M est une constante, puis on fait varier N entre K et M. Calculez le regret pour ces différentes tailles du training set différents MAB et representez le regret moyen, le regret min et max (vous devriez trouver une courbe en U ou en V pour le regret moyen). Quelle est la taille optimale du training set ?

```
O = N + M
K = 5

MAB = generate_custom_arms(.3, 0.01)

list_min_max_mean = []

for nb_patiens_exploration in tqdm(range(K, O)):
    nb_patiens_exploitation = O - nb_patiens_exploration
    _, _, last_regrets = simulation_update_immune_rate(MAB,
nb_patiens_exploration, nb_patiens_exploitation, add_bias=False,
proba_take_best_vaccin=1)

    list_min_max_mean.append((last_regrets.min().item(),
last_regrets.max().item(), last_regrets.mean().item()))
```

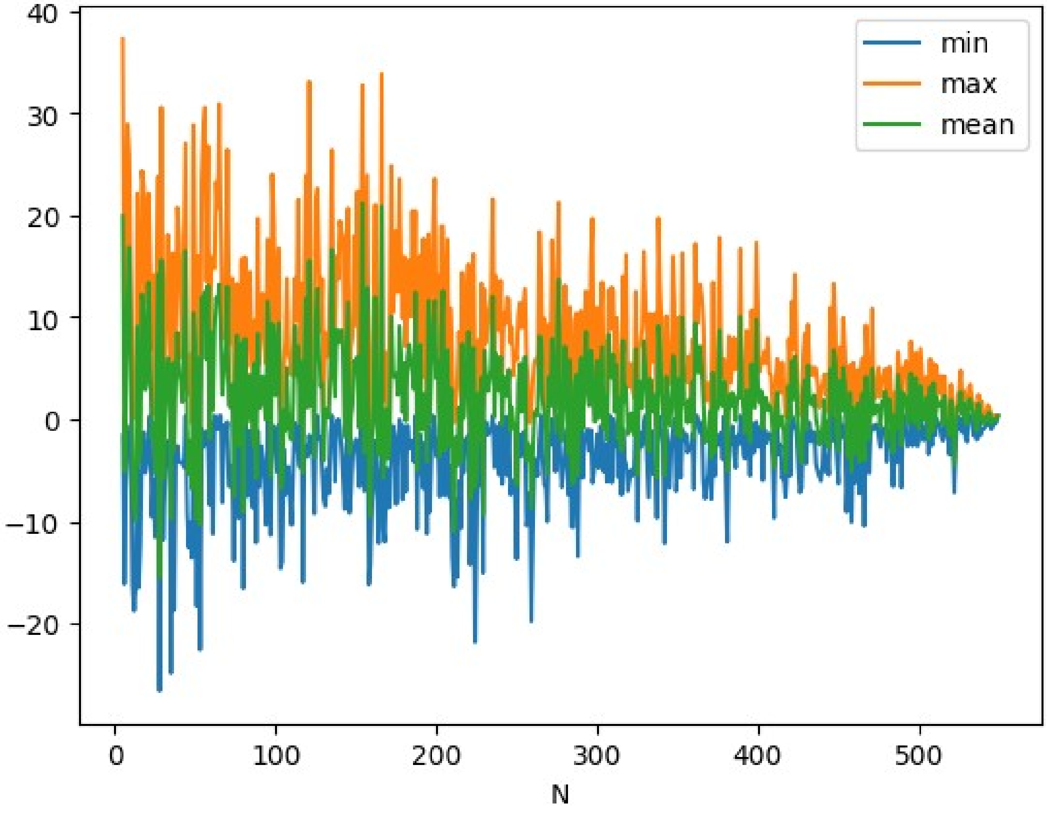
```
list_min_max_mean
```

```
{"model_id":"ee25637bc85b474eb9cfa4f4072b3a97","version_major":2,"version_minor":0}
```

```
[(-1.5999999046325684, 37.30000305175781, 19.939266204833984),
 (-16.139999389648438, 5.1399993896484375, -5.156984329223633),
 (-6.680000305175781, 12.580001831054688, 2.1706089973449707),
 (-2.520000457763672, 28.900009155273438, 11.415166854858398),
 (-2.5999999046325684, 25.94000244140625, 16.84979820251465),
 (-0.5199999809265137, 14.05999755859375, 6.6311116218566895),
 (-16.05999755859375, 0.4399986267089844, -7.728755474090576),
 (-18.759994506835938, 1.1999998092651367, -8.925760269165039),
 (-16.139999389648438, 0.039999961853027344, -9.949681282043457),
 (-3.7200002670288086, 22.1199951171875, 9.090374946594238),
 (-16.479995727539062, 3.940000534057617, -4.330466270446777),
 (-12.279998779296875, 8.82000732421875, -2.7822086811065674),
 (-1.9599990844726562, 24.360000610351562, 12.247167587280273),
 (-1.2799999713897705, 22.8800048828125, 11.222783088684082),
 (-5.220001220703125, 11.460006713867188, 2.8995113372802734),
 (-4.319999694824219, 15.199996948242188, 3.7794349193573),
 (-1.4200000762939453, 22.1199951171875, 13.402458190917969),
 (0.3400000035762787, 15.099998474121094, 6.606137275695801),
 (-9.55999755859375, 6.8600006103515625, -1.8604923486709595),
 (-5.979999542236328, 14.139999389648438, -0.46513211727142334),
 (-11.540000915527344, 0.5, -5.258094310760498),
 (-1.779998779296875, 15.160003662109375, 4.055344581604004),
 (0.019999980926513672, 23.779998779296875, 14.204283714294434),
 (-26.6199951171875, 0.6800000071525574, -15.509540557861328),
 (-2.8600006103515625, 30.540008544921875, 15.57877254486084),
 (-11.800003051757812, 1.760000228881836, -5.847306728363037),
 (-6.439998626708984, 4.5200042724609375, 0.05896049737930298),
 (-2.5799999237060547, 12.540008544921875, 4.666294097900391),
 (-2.1000003814697266, 18.080001831054688, 5.982631683349609),
 (-3.8400001525878906, 8.759994506835938, 0.9384505152702332),
 (-24.899993896484375, 1.8000001907348633, -9.780970573425293),
 (-2.779998779296875, 16.279998779296875, 5.501362800598145),
 (-18.699996948242188, 4.400001525878906, -3.0430009365081787),
 (-3.9199981689453125, 9.239997863769531, 3.352579355239868),
 (-2.5999999046325684, 20.740005493164062, 8.470529556274414),
 (-4.199999809265137, 18.05999755859375, 4.575883388519287),
 (-4.21999979019165, 11.580001831054688, 3.9750499725341797),
 (-4.2599945068359375, 6.040000915527344, 1.6855127811431885),
 (-4.700000762939453, 15.94000244140625, 6.474400043487549),
 (0.3400000035762787, 27.040008544921875, 16.512136459350586),
 (-9.240005493164062, 6.319999694824219, -0.8611871600151062),
 (-12.580001831054688, 2.440000057220459, -5.354364395141602),
 (-10.1199951171875, 6.319999694824219, -2.1291444301605225),
 (-13.580001831054688, -0.3199920654296875, -6.796772003173828),
 (-2.8600001335144043, 28.82000732421875, 10.43780517578125),
 (0.279998779296875, 13.220001220703125, 7.316000938415527),
 (-18.300003051757812, 0.3400000035762787, -9.80761432647705),
 (-3.3600006103515625, 15.979995727539062, 5.962530612945557),
 (-22.55999755859375, 3.0999999046325684, -10.301769256591797),
 (-0.9200000762939453, 23.479995727539062, 11.959758758544922),
 (-2.739999771118164, 28.339996337890625, 11.596769332885742),
 (-2.5199999809265137, 30.540008544921875, 12.481986045837402),
 (-1.1800003051757812, 13.540008544921875, 5.890751361846924),
 (-1.619999885559082, 26.779998779296875, 13.07829475402832),
 (-8.360000610351562, 2.940000534057617, -1.8732376098632812),
 (-0.2799999713897705, 15.900009155273438, 8.506734848022461),
 (-11.220001220703125, 15.0, 0.9380377531051636),
 (0.3400000035762787, 14.759994506835938, 9.068525314331055),
 (-0.6599999666213989, 23.259994506835938, 11.918933868408203),
 (-0.9800000190734863, 20.680007934570312, 11.405228614807129),
 (0.3400000035762787, 30.900009155273438, 13.199382781982422),
 (0.3400000035762787, 18.55999755859375, 7.854959487915039),
 (-8.139999389648438, 7.8600006103515625, 2.4352805614471436),
 (-0.5999984741210938, 9.900001525878906, 4.315394878387451),
 (-2.0199995040893555, 13.520004272460938, 6.715468883514404),
 (-0.2799999713897705, 26.44000244140625, 12.988751411437988),
 (-6.520000457763672, 10.040008544921875, 1.4225478172302246),
 (-6.619998931884766, 7.5200042724609375, 1.139205813407898),
 (-4.519996643066406, 13.82000732421875, 3.377401113510132),
 (-13.880001068115234, 0.7000000476837158, -7.548654079437256),
 (-2.9000000953674316, 10.779998779296875, 4.728422164916992),
 (-1.2599999904632568, 13.259994506835938, 8.209917068481445),
 (-9.699996948242188, 2.479999542236328, -2.6123881340026855),
 (-9.199996948242188, 4.459999084472656, -1.0984737873077393),
 (-0.3199999928474426, 15.699996948242188, 7.43745231628418),
 (-16.539993286132812, 1.3600000143051147, -9.1278715133667),
 (-0.3199999928474426, 15.860000610351562, 7.878678798675537),
 (-7.520000457763672, 5.099998474121094, -0.33410170674324036),
 (-7.3600006103515625, 11.720001220703125, 1.3565746545791626),
 (-1.05999755859375, 14.419998168945312, 4.349228382110596),
 (-8.060001373291016, 7.1000061035156250, 0.8780653476715088),
 (-8.699996948242188, 3.279998779296875, -2.6374993324279785),
 (-0.6599999666213989, 9.660003662109375, 5.497711658477783),
 (-12.080001831054688, 2.820000171661377, -5.75753116607666),
 (-1.1599998474121094, 19.680007934570312, 8.357789039611816),
 (-6.6199951171875, 5.340000152587891, 1.109131362121582),
 (-2.6199951171875, 8.520000457763672, 1.6270159482955933),
 (-3.8999996185302734, 12.319999694824219, 4.191572666168213),
 (-5.9799957275390625, 9.220001220703125, 1.4486223459243774),
 (-10.319992065429688, 5.479999542236328, -0.049034252762794495),
 (0.019999980926513672, 17.62000274658203, 11.555166244506836),
 (-0.6599999666213989, 17.419998168945312, 8.442511558532715),
 (-11.319992065429688, 2.2799997329711914, -5.583796501159668),
 (-0.9600000381469727, 24.0, 9.3241605758667),
 (-0.6599999666213989, 19.339996337890625, 8.682573318481445),
 (-2.3600006103515625, 7.540008544921875, 2.3455567359924316),
 (-5.159999847412109, 12.459999084472656, 4.099110126495361),
 (-0.5399999618530273, 16.779998779296875, 9.408125877380371),
 (-14.580001831054688, 1.0199999809265137, -6.750514030456543),
 (-13.94000244140625, 0.3400000035762787, -6.447218894958496),
 (-2.899993896484375, 4.840003967285156, 0.8649446964263916),
 (-7.4199981689453125, 7.600006103515625, -1.14729642868042),
 (-5.280000686645508, 13.800003051757812, 5.039819717407227),
 (-4.659999847412109, 6.340003967285156, 0.590543806552887),
 (-7.720001220703125, 5.9199981689453125, -1.6899312734603882),
 (-10.339996337890625, 4.340000152587891, -1.9913628101348877),
 (-4.579998016357422, 7.920000076293945, 1.597267508506775),
 (-0.3199999928474426, 13.800003051757812, 9.134567260742188),
 (-0.7399997711181641, 9.720001220703125, 4.343296051025391),
 (-3.259999990463257, 21.540008544921875, 7.12945032119751),
 (-4.680000305175781, 3.6200027465820312, -0.31218311190605164),
 (-0.4199981689453125, 13.279998779296875, 5.6688947677612305),
 (-15.959991455078125, 2.3400001525878906, -5.256950855255127),
 (-2.7399978637695312, 11.779998779296875, 2.913241386413574),
 (-2.5799999237060547, 23.860000610351562, 11.855314254760742),
 (-3.6399993896484375, 5.700000762939453, 1.1537216901779175),
 (-1.5799999237060547, 33.100006103515625, 15.519579887390137),
 (-3.1599960327148438, 8.839996337890625, 1.390281319618225),
 (-2.6399993896484375, 8.94000244140625, 3.570305347442627),
 (-9.239997863769531, 3.3999996185302734, -3.719858407974243),
 (-0.25999999046325684, 21.5, 10.057648658752441),
 (0.3400000035762787, 22.6199951171875, 12.794812202453613),
 (-1.940000057220459, 15.540000915527344, 6.812861442565918),
 (-1.8199996948242188, 8.639999389648438, 3.201470136642456),
 (-4.799999237060547, 13.779998779296875, 3.497720718383789),
 (-7.879997253417969, 0.7200000286102295, -4.051427841186523),
 (-8.5, 8.460006713867188, 0.9155138731002808),
 (-4.840000152587891, 6.180000305175781, 1.026651382446289),
 (-7.280000686645508, 7.120002746582031, 0.14153552055358887),
 (-1.5799999237060547, 14.759994506835938, 5.084712028503418),
 (0.3400000035762787, 26.399993896484375, 16.61879539489746),
 (-1.4599990844726562, 11.840003967285156, 4.784299850463867),
 (-6.09999942779541, 6.399993896484375, 2.1984026432037354),
 (-0.4200000762939453, 15.94000244140625, 8.787670135498047),
 (0.019999980926513672, 13.740005493164062, 7.555816650390625),
 (-0.8199996948242188, 19.419998168945312, 8.643171310424805),
 (-2.9000000953674316, 18.44000244140625, 8.75134563446045),
 (-5.659999847412109, 17.720001220703125, 6.454020023345947),
 (-8.819999694824219, 6.6999969482421875, -1.6866823434829712),
 (-2.520000457763672, 11.919998168945312, 4.478177547454834),
 (0.05999994277954102, 20.660003662109375, 11.437284469604492),
 (-9.200000762939453, 1.94000244140625, -3.3579201698303223),
 (-7.099998474121094, 7.119998931884766, 0.2159809023141861),
 (-3.2399978637695312, 8.120002746582031, 1.5871150493621826),
 (-2.8199996948242188, 17.94000244140625, 5.753965377807617),
 (-1.9800000190734863, 9.199996948242188, 4.6800007820129395),
 (-3.8999996185302734, 22.32000732421875, 6.137845039367676),
 (-6.579999923706055, 6.120002746582031, -0.8006525039672852),
 (-0.25999999046325684, 14.819999694824219, 6.848917484283447),
 (0.09999990463256836, 32.73999786376953, 21.13899040222168),
 (-7.0, 1.4000000953674316, -3.2875943183898926),
 (-4.039999961853027, 12.860000610351562, 3.566244602203369),
 (-0.940000057220459, 23.919998168945312, 12.875674247741 7),
 (-16.199996948242188, 3.90000057220459, -6.371121883392334),
 (-14.319999694824219, -0.6399999856948853, -9.469974517822266),
 (-7.8600006103515625, 4.100000381469727, -1.8248710632324219),
 (-5.939998626708984, 11.099998474121094, 1.4876614809036255),
 (-2.09999942779541, 21.0, 11.985671997070312),
 (-6.7599945068359375, 4.239999771118164, -0.3707486391067505),
 (-12.139999389648438, 2.720001220703125, -3.487201690673828),
 (-5.959999084472656, 4.659999847412109, -1.22675251960 7544),
 (-0.6599999666213989, 33.86000061035156, 20.851043701171875),
 (-10.860000610351562, 1.7000000476837158, -5.751330852508545),
 (-12.019996643066406, 4.579999923706055, -4.1727213859558105),
 (-4.779998779296875, 6.540008544921875, 1.0003682374954224),
 (-8.65999984741211, 1.040000915527343 8, -3.8378937244415283),
 (-8.580001831054688, 8.80000114440918, -1.5002630949020386),
 (-3.1999998092651367, 24.840003967285156, 10.083440780639648),
 (-0.6599999666213989, 10.5, 4.602176189422607),
 (-2.1399993896484375, 13.840003967285156, 4.16180944442749),
 (-8.299999237060547, 18.480003356933594, 4.458667755126953),
 (-2.05999755859375, 12.480003356933594, 2.402407169342041),
 (-0.019999504089355469, 23.55999755859375, 9.143004417419434),
 (-0.5199999809265137, 15.480003356933594, 6.5175275802612305),
 (-7.8600006103515625, 9.880001068115234, 0.6901355385780334),
 (-0.6599999666213989, 15.900001525878906, 5.532162666320801),
 (-7.0800018310546875, 7.1399993896484375, -0.15961992740631104),
 (-0.5999999046325684, 14.120002746582031, 5.9800004959106445),
 (-2.880000114440918, 15.599998474121094, 7.320218563079834),
 (-2.299999952316284, 10.05999755859375, 5.526612758636475),
 (-2.0999984741210938, 20.419998168945312, 4.652877330780029),
 (-4.200000762939453, 11.760002136230469, 2.959341526031494),
 (0.340000003576278 7, 20.419998168945312, 12.441983222961426),
 (-10.779998779296875, 1.0399999618530273, -5.002706527709961),
 (-1.5, 9.900001525878906, 3.323934316635132),
 (-1.4799995422363281, 16.040000915527344, 7.29500007629394 5),
 (-5.159999847412109, 4.099998474121094, -1.017269492149353),
 (-6.759998321533203, 17.68000030517578, 2.499274492263794),
 (-2.220001220703125, 5.720001220703125, 1.347395658493042),
 (-11.200000762939453, 2.3600006103515625, -4.217302322387695),
 (-0.6599999666213989, 18.139999389648438, 11.612957954406738),
 (-9.080001831054688, 6.799999237060547, -0.3081914186477661),
 (-3.6599960327148438, 10.119998931884766, 4.446290016174316),
 (0.3400000035762787, 18.459999084472656, 9.484432220458984),
 (-0.6599999666213989, 23.580001831054688, 11.612080574035645),
 (-3.5999984741210938, 9.720001220703125, 4.007143020629883),
 (-1.940000057220459, 14.120002746582031, 5.743553161621094),
 (-7.579999923706055, 0.7000000476837158, -3.641264200210571 3),
 (-0.9800000190734863, 18.900001525878906, 8.416484832763672),
 (-2.619999885559082, 18.919998168945312, 12.573816299438477),
 (-7.4799957275390625, 3.1800003051757812, -3.1278257369995117),
 (-5.260000228881836, 9.919998168945312, 2.740116596221924),
 (-0.6000003814697266, 17.62000274658203, 6.777376651763916),
 (-7.6999969482421875, 1.6000003814697266, -2.5964322090148926),
 (-4.599998474121094, 10.8800048828125, 2.990440607070923),
 (-13.479995727539062, 5.340000152587891, -4.10352897644043),
 (-16.360000610351562, -0.3199999928474426, -10.987610816955566),
 (-7.839996337890625, 1.6599998474121094, -1.7753247022628784),
 (-15.459999084472656, 3.1800003051757812, -4.554835796356201),
 (-6.079999923706055, 8.55999755859375, 1.167976975440979),
 (-10.699996948242188, 4.619999885559082, -4.247163772583008),
 (-1.880000114440918, 14.360000610351562, 7.3931145668029785),
 (-5.219999313354492, 6.260002136230469, 0.1013219803571701),
 (-2.0400009155273438, 6.700004577636719, 2.5437357425689697),
 (-2.619999885559082, 11.739997863769531, 3.941511392593384),
 (-1.559999942779541, 15.200004577636719, 8.560909271240234),
 (-14.239997863769531, 0.8000001907348633, -7.802734375),
 (-6.0199995040893555, 13.099998474121094, 3.8080496788024902),
 (-0.11999893188476562, 16.18000030517578, 7.007706642150879),
 (-21.839996337890625, 2.4600000381469727, -6.6400604248046875),
 (-9.200000762939453, 1.0199999809265137, -5.032307147979736),
 (-2.1399993896484375, 7.760002136230469, 2.5493834018707275),
 (-6.439998626708984, 1.7399978637695312, -1.9881107807159424),
 (-1.2799999713897705, 13.340003967285156, 4.096336364746094),
 (-15.040000915527344, 1.0199999809265137, -9.356573104858398),
 (-0.2799999713897705, 12.720001220703125, 6.6918745040893555),
 (-0.2999999523162842, 7.5200042724609375, 3.8106589317321777),
 (-1.5400009155273438, 9.200000762939453, 2.764591932296753),
 (-1.239999771118164, 10.760002136230469, 3.9748270511627197),
 (-0.940000057220459, 12.44000244140625, 5.022912979125977),
 (0.3400000035762787, 21.580001831054688, 12.034287452697754),
 (-4.6999969482421875, 4.859999656677246, 0.2538222670555115),
 (-0.6599999666213989, 14.05999755859375, 6.440703868865967),
 (-2.09999942779541, 11.280000686645508, 4.421538829803467),
 (-5.4199981689453125, 8.680000305175781, 2.814920425415039),
 (-0.3199999928474426, 13.599998474121094, 6.137742519378662),
 (-6.340000152587891, 4.5200042724609375, -0.5750802755355835),
 (0.220001220703125, 10.459999084472656, 4.189091682434082),
 (-3.260000228881836, 9.400001525878906, 3.3665153980255127),
 (-5.479999542236328, 11.959999084472656, 2.2618958950042725),
 (0.3400000035762787, 11.760002136230469, 4.846230030598145),
 (-7.399999618530273, 7.5200042724609375, -1.094078421592712 4),
 (-6.1399993896484375, 9.040000915527344, 1.1717497110366821),
 (-5.979999542236328, 4.299999237060547, -0.42046302556991577),
 (-7.279998779296875, 6.5800018310546875, -0.7065110802650452),
 (-13.699996948242188, 5.840000152587891, -6.426666259765625),
 (-3.119999885559082, 10.120002746582031, 3.397994041442871),
 (-1.5199999809265137, 11.400001525878906, 5.454161643981934),
 (-3.5800018310546875, 8.980003356933594, 0.734074592590332),
 (-1.4799995422363281, 10.94000244140625, 4.496757507324219),
 (-1.1399993896484375, 12.760002136230469, 6.116610527038574),
 (-10.399999618530273, 5.620002746582031, -4.081291675567627),
 (-9.700000762939453, 3.6200027465820312, -2.4056649208068848),
 (-7.180000305175781, -0.3199999928474426, -3.5735607147216797),
 (-19.779998779296875, 0.7400000095367432, -8.744878768920898),
 (-11.05999755859375, 3.5600013732910156, -3.3748269081115723),
 (-3.8400001525878906, 5.520000457763672, 0.4522496461868286),
 (-4.21999979019165, 5.9199981689453125, 0.859167218208313),
 (-5.520000457763672, 5.44000244140625, 0.38857194781303406),
 (-2.0799999237060547, 18.360000610351562, 8.129161834716797),
 (0.3400000035762787, 14.180000305175781, 7.865614891052246),
 (-3.479999542236328, 5.560001373291016, 1.8725355863571167),
 (-6.399999618530273, 9.879997253417969, 0.6722267270088196),
 (-6.739997863769531, 2.040000915527344 38, -2.308439016342163),
 (-10.0, 0.36000001430511475, -5.405195236206055),
 (-1.5, 9.300000190734863, 4.137857913970947),
 (-2.5, 4.379997253417969, 1.313262224197387 7),
 (0.019999980926513672, 17.520004272460938, 7.850863456726074),
 (-1.94000244140625, 8.360000610351562, 3.386354446411133),
 (-5.119998931884766, 2.5, -1.330289244651794 4),
 (-0.3199999928474426, 13.5, 7.000000476837158),
 (0.3400000035762787, 21.23999786376953, 13.680658340454102),
 (-4.0, 2.719999313354492, -0.5335526466369629),
 (-6.659999847412109, 1.1399993896484375, -2.141469955444336),
 (-4.840000152587891, 8.080001831054688, 2.0481185913085938),
 (-2.9600000381469727, 13.099998474121094, 7.155186176300049),
 (-0.9800000190734863, 5.5, 2.665800094604492),
 (-7.4199981689453125, 2.119999885559082, -3.3520889282226562),
 (-1.2599999904632568, 11.0, 5.79970121383667),
 (-10.540000915527344, 2.1800003051757812, -4.617518424987793),
 (-10.579998016357422, -0.6599999666213989, -6.323395252227783),
 (-1.619999885559082, 9.720001220703125, 3.409849166870117),
 (-0.6599999666213989, 10.400001525878906, 4.366692543029785),
 (-13.419998168945312, 0.7200000286102295, -5.4541215896606445),
 (-2.059999465942383, 9.05999755859375, 3.455709457397461),
 (0.019999980926513672, 10.619998931884766, 6.027693271636963),
 (-5.760002136230469, 3.560000419616699, -0.9544396996498108),
 (-2.299999952316284, 8.580001831054688, 2.9873647689819336),
 (-1.3199999332427979, 12.540000915527344, 8.560389518737793),
 (-0.9600000381469727, 6.540000915527344, 3.209531784057617),
 (-3.079998016357422, 4.680000305175781, 1.3396083116531372),
 (-1.0399999618530273, 15.680000305175781, 6.71220588684082),
 (-6.219999313354492, 19.68000030517578, 4.6938347816467285),
 (-5.260002136230469, 3.880000114440918, -0.18841221928596497),
 (-3.1599998474121094, 5.180000305175781, -0.20780827105045319),
 (-1.5, 10.94000244140625, 5.758000373840332),
 (-4.779998779296875, 4.560001373291016, 0.1064261868596077),
 (-0.6599999666213989, 9.319999694824219, 3.757420063018799),
 (-0.6399999856948853, 13.44000244140625, 7.107368469238281),
 (-5.120002746582031, 4.119998931884766, -0.9937393069267273),
 (-2.5, 7.5800018310546875, 2.1914291381835938),
 (-6.119999885559082, 2.5200042724609375, -1.9401633739471436),
 (-0.3199999928474426, 12.65999984741211, 8.274239540100098),
 (0.3400000035762787, 8.880001068115234, 5.330661773681641),
 (-6.219999313354492, 12.94000244140625, 4.090207576751709),
 (0.019999980926513672, 12.599998474121094, 6.340834140777588),
 (-2.559999942779541, 5.260002136230469, 0.7372390031814575),
 (-0.6599999666213989, 9.919998168945312, 5.306470394134521 5),
 (-5.659999847412109, 5.879997253417969, -1.9703792333602905),
 (-0.6399999856948853, 4.619999885559082, 2.112034320831299),
 (-7.4199981689453125, 1.7200000286102295, -3.058722734451294),
 (-0.940000057220459, 13.980003356933594, 7.69359016418457),
 (-4.020000457763672, 6.6399993896484375, 1.6855798959732056),
 (-4.420000076293945, 16.12000274658203, 3.1401729583740234),
 (-7.0, 3.8999996185302734, -1.4864064455032349),
 (-6.1399993896484375, 3.2000045776367188, -0.8256517052650452),
 (-4.979999542236328, 5.44000244140625, 0.5759829878807068),
 (-5.5, 2.979999542236328, -1.1796486377716064),
 (-4.880001068115234, 2.0800018310546875, -1.5263432264328003),
 (0.3400000035762787, 12.5, 6.8776116027832),
 (-9.939998626708984, 0.3400000035762787, -5.40666580200195 3),
 (-1.2799999713897705, 7.060001373291016, 3.709821939468384),
 (-2.719999313354492, 7.44000244140625, 2.192107915878296),
 (-2.5999999046325684, 8.980003356933594, 1.1172075271606445),
 (-0.2999999523162842, 16.459999084472656, 7.690227031707764),
 (0.3400000035762787, 11.779998779296875, 5.0518188476 5625),
 (-3.5199999809265137, 8.119998931884766, 2.368036985397339),
 (-7.659999847412109, 2.9200000762939453, -3.136971950531006),
 (-0.9800000190734863, 10.340003967285156, 3.972442388534546),
 (-1.7199993133544922, 5.599998474121094, 1.53351891040802),
 (-5.399999618530273, 10.080001831054688, 0.6827910542488098),
 (-4.460000038146973, 10.400001525878906, 2.4238321781158447),
 (-9.65999984741211, -0.3199999928474426, -5.732675552368164),
 (0.3400000035762787, 19.720001220703125, 9.162830352783203),
 (-0.6599999666213989, 13.420001983642578, 6.973649501800537),
 (-1.3199996948242188, 9.400001525878906, 2.760476589202881),
 (-2.359999656677246, 4.6399993896484375, 0.408134400844574),
 (-12.15999984741211, 1.7000000476837158, -5.5084614753723145),
 (-0.9600000381469727, 10.119998931884766, 4.186087131500244),
 (-3.9599990844726562, 6.159999847412109, 0.9521363377571106),
 (-6.6999969482421875, 3.479999542236328, -0.5702434778213501),
 (-2.1800003051757812, 4.479999542236328, 0.977451503276825),
 (-1.1999998092651367, 15.94000244140625, 6.226798057556152),
 (0.019999980926513672, 11.540000915527344, 6.08425760269165),
 (-7.040000915527344, 1.9200000762939453, -1.510745882987976),
 (-4.880001068115234, 2.1000003814697266, -0.3649995923042297 4),
 (-1.1999998092651367, 9.980003356933594, 4.025125980377197),
 (-5.720001220703125, 2.799999237060547, -1.7356562614440918),
 (-0.9800000190734863, 16.279998779296875, 9.939188957214355),
 (-1.9800000190734863, 4.459999084472656, 1.612449288368225),
 (-1.4399995803833008, 4.8600006103515625, 1.473846435546875),
 (-3.259998321533203, 3.6200027465820312, -0.30360791087150574),
 (-2.8600006103515625, 6.180000305175781, 1.9281867742538452),
 (-2.720001220703125, 2.1399993896484375, -0.10666629672050476),
 (-1.4399995803833008, 7.799999237060547, 3.702827930450439 5),
 (-6.840000152587891, 3.760000228881836, -1.8352627754211426),
 (0.340000003576278 7, 17.200000762939453, 9.400529861450195),
 (0.340000003576278 7, 9.319999694824219, 5.449149131774902),
 (-0.2199997901916504, 11.860000610351562, 4.59101676940918),
 (-1.3199999332427979, 13.200000762939453, 7.1717209815979),
 (-0.3199999928474426, 11.880001068115234, 4.987567901611328),
 (-6.159999847412109, 9.299999237060547, 1.7163046598434448),
 (-7.779998779296875, 3.3999996185302734, -2.277376651763916),
 (-6.5, 1.8000011444091797, -1.99439537525177),
 (-5.340000152587891, 3.5400009155273438, -1.5903862714767456),
 (-1.9199981689453125, 3.260000228881836, 0.7588892579078674),
 (-7.819999694824219, 0.5, -3.763128280639648 4),
 (-3.5399999618530273, 13.420001983642578, 4.5760674476623535),
 (-5.539999008178711, -0.6599999666213989, -3.288022518157959),
 (-1.5799980163574219, 5.719999313354492, 1.5615912675857544),
 (-2.539999008178711, 5.479999542236328, 1.080000400543213),
 (-1.6399999856948853, 17.779998779296875, 8.727012634277344),
 (-1.2999999523162842, 4.1399993896484375, 1.8343355655670166),
 (-3.9199981689453125, 5.040000915527344, 1.2879072427749634),
 (0.340000003576278 7, 5.900001525878906, 3.6727490425109863),
 (-6.880001068115234, 2.619999885559082, -1.206470131874084 5),
 (-12.020000457763672, 1.5399999618530273, -5.052662372589111),
 (-1.8400001525878906, 8.119998931884766, 2.3847622871398926),
 (-2.4200000762939453, 3.5799999237060547, 1.27856314182281 5),
 (-0.6599999666213989, 5.520000457763672, 2.600843667984009),
 (-3.299999237060547, 7.9800004959106445, 1.541212558746338),
 (-4.759998321533203, 4.880001068115234, -0.11463381350040436),
 (-1.6399999856948853, 5.159999847412109, 2.2603683471679688),
 (-2.4399986267089844, 6.300000190734863, 2.1112349033355713),
 (0.340000003576278 7, 16.740001678466797, 10.055527687072754),
 (-1.3199999332427979, 5.6399993896484375, 2.3137502670288086),
 (-3.579998016357422, 5.559999465942383, 0.7220128178596497),
 (-3.8400001525878906, 1.3400001525878906, -1.2737972736358643),
 (-0.2199997901916504, 8.700000762939453, 2.8472611904144287),
 (-2.439999580383301, 4.439998626708984, 1.0489745140075684),
 (-5.5, 0.8000011444091797, -2.176774024963379),
 (-0.3199999928474426, 10.680000305175781, 5.401947975158691),
 (-3.299999952316284, 5.659999847412109, 1.3564709424972534),
 (-1.5799999237060547, 7.340000152587891, 1.1415791511535645),
 (-0.25999999046325684, 17.31999969482422, 9.773774147033691),
 (-2.7200002670288086, 10.819999694824219, 3.370000123977661),
 (-1.4200000762939453, 4.659999847412109, 1.6342284679412842),
 (-0.6599999666213989, 5.640000343322754, 2.7759461402893066),
 (-5.979999542236328, 5.5, -0.914829671382904),
 (-0.9600000381469727, 6.6399993896484375, 2.1201372146606445),
 (-2.3199996948242188, 5.480000019073486, 2.1372416019439697),
 (-1.7200002670288086, 7.939998626708984, 0.89305579662323),
 (-1.3999996185302734, 4.520000457763672, 1.38909113407135),
 (-1.559999942779541, 4.279998779296875, 0.9085919260978699),
 (-3.8999996185302734, 1.1599998474121094, -1.0373046398162842),
 (-9.680000305175781, 0.3400000035762787, -4.744285583496094),
 (-2.0799999237060547, 4.0800018310546875, 1.1381298303604126),
 (-0.45999908447265625, 5.920001983642578, 2.622753858566284),
 (-3.4600000381469727, 3.4599990844726562, -0.7516785860061646),
 (-2.5, 2.6000003814697266, -0.2467644363641739),
 (-5.799999237060547, 5.579999923706055, 0.2607409954071045),
 (-5.439999580383301, 2.9200000762939453, -1.1694027185440063),
 (-7.6399993896484375, 0.38000011444091797, -5.250075340270996),
 (-4.619999885559082, 0.1399993896484375, -2.094545125961303 7),
 (-0.6399999856948853, 7.8600006103515625, 3.2186262607574463),
 (-5.540000915527344, 4.520000457763672, -0.03769208490848541),
 (0.3400000035762787, 11.5, 5.720154762268066),
 (-0.2199993133544922, 6.840000152587891, 1.9456251859664917),
 (-0.6599999666213989, 14.180000305175781, 6.169449329376221),
 (-1.239999771118164, 6.200000762939453, 2.764603137969 9707),
 (-4.579999923706055, 0.3400000035762787, -2.419999837875366),
 (-7.159999847412109, 2.1599998474121094, -3.6290318965911865),
 (-6.739999771118164, 0.7000000476837158, -4.334633350372 3145),
 (-1.239999771118164, 6.319999694824219, 2.9181969165802),
 (-0.3199999928474426, 8.380001068115234, 3.8391735553741455),
 (-4.3600006103515625, 2.799999237060547, -1.021666407585144),
 (-0.2999999523162842, 9.299999237060547, 5.265545845031738),
 (-0.2999999523162842, 5.279999732971191, 2.4842376708984375),
 (-2.179999828338623, 5.020000457763672, 1.5129916667938232),
 (-1.260000228881836, 4.299999237060547, 1.4072415828704834),
 (-1.2799999713897705, 5.079999923706055, 2.3895657062530518),
 (-4.42000007629394 5, 4.960000038146973, 0.014912553131580353),
 (-5.079999923706055, 3.4000015258789062, -0.7881413102149963),
 (-5.559999465942383, 1.1599998474121094, -2.495356798171997),
 (-6.039999008178711, 3.059999942779541, -1.3113510608673096),
 (-4.539999008178711, 3.3400001525878906, -1.1209088563919067),
 (-0.7600002288818359, 4.380001068115234, 1.9018352031707764),
 (-5.119999885559082, 2.3800010681152344, -2.3218514919281006),
 (-3.760000228881836, 4.380001068115234, -0.2661680281162262),
 (-5.420000076293945, 0.6800000071525574, -2.7533960342407227),
 (0.1199998855590820 3, 10.920000076293945, 4.858095645904541),
 (-2.1000003814697266, 3.0400009155273438, 0.465384781360626 2),
 (0.340000003576278 7, 13.299999237060547, 6.777087688446045),
 (-1.2999999523162842, 8.920000076293945, 5.490392208099365),
 (-4.079999923706055, 0.18000030517578125, -1.6104949712753296),
 (-6.319999694824219, 1.4000000953674316, -3.1700000762939453),
 (0.340000003576278 7, 7.079999923706055, 3.7979798316955566),
 (-1.7799997329711914, 3.520000457763672, 0.7891838550567627),
 (-0.6599999666213989, 9.920000076293945, 5.062062263488769 5),
 (-0.619999885559082, 6.960000991821289, 3.312916755676269 5),
 (-9.079999923706055, 2.7800002098083496, -2.0063157081604004),
 (-5.760000228881836, -0.2999999523162842, -3.6797869205474854),
 (-0.6399999856948853, 5.280000686645508, 2.6036562919616 7),
 (-10.059999465942383, 0.36000001430511475, -5.113913059234619),
 (-2.2799999713897705, 5.559999465942383, 1.1125276088714 6),
 (-6.280000686645508, 0.36000001430511475, -3.418889045715332),
 (-5.079999923706055, 0.8000001907348633, -1.9359546899795532),
 (-0.9200000762939453, 4.840000152587891, 1.5163638591766357),
 (-7.299999237060547, -0.2999999523162842, -4.2009196281433105),
 (-1.5999999046325684, 4.680000305175781, 1.2202327251434326),
 (-0.6599999666213989, 6.0, 2.5494120121002197),
 (-10.440000534057617, -0.2999999523162842, -4.085713863372803),
 (-0.2799999713897705, 9.220001220703125, 4.17156648635864 3),
 (-1.4799995422363281, 5.880001068115234, 1.5246343612670898),
 (-5.239999771118164, 1.0999999046325684, -1.9118516445159912),
 (-1.539999008178711, 2.9600000381469727, 0.9825001955032349),
 (-0.6399999856948853, 10.860000610351562, 4.941771984100342),
 (-0.9800000190734863, 2.520000457763672, 0.4556411802768707),
 (-0.5199999809265137, 2.020000457763672, 0.7664936780929565),
 (-0.5, 3.520000457763672, 1.3794738054275513),
 (-1.5799999237060547, 5.059999465942383, 1.2133334875106812),
 (-2.739999771118164, 2.1599998474121094, -0.7770269513130188),
 (-0.940000057220459, 3.8000011444091797, 1.7443835735321045),
 (-0.8600006103515625, 4.279999732971191, 1.4377778768539429),
 (-0.9800000190734863, 4.920000076293945, 2.747042179107666),
 (-4.199999809265137, 1.3600000143051147, -2.07285714149475 1),
 (-0.9600000381469727, 4.960000038146973, 2.015942335128784),
 (-1.3199999332427979, 5.120000839233398, 1.8476473093032837),
 (-2.6800003051757812, 2.059999942779541, -0.5295521020889282),
 (-6.559999465942383, -0.23999977111816406, -3.291818380355835),
 (-1.1999998092651367, 2.6000003814697266, 0.389230877161026),
 (-4.260000228881836, 0.9400005340576172, -0.8718748092651367),
 (0.340000003576278 7, 6.600000381469727, 4.340317726135254),
 (-0.3199999928474426, 4.579999923706055, 2.5003228187561035),
 (-6.680000305175781, 0.7200000286102295, -2.5911471843719482),
(0.3400000035762787, 4.940000534057617, 2.620000123977661),
(0.3400000035762787, 5.059999465942383, 1.7084746360778809),
(-1.2799999713897705, 5.719999313354492, 2.2196552753448486),
(-2.6599998474121094, 1.1199999885559082, -0.49087703227996826),
(-2.299999952316284, 2.3199996948242188, 0.17214298248291016),
(0.3400000035762787, 7.640000343322754, 4.483636856079102),
(-2.5999999046325684, 3.3600006103515625, 0.5907408595085144),
(-1.6999998092651367, 1.760000228881836, -0.3294338583946228),
(0.3400000035762787, 6.520000457763672, 4.067692279815674),
(-2.6399998664855957, 6.340000152587891, 1.055686354637146),
(-2.09999942779541, 0.3400000035762787, -0.9499999284744263),
(0.3400000035762787, 6.960000038146973, 3.2551021575927734),
(-0.9200000762939453, 4.619999885559082, 1.8925002813339233),
(-2.0399999618530273, 2.0399999618530273, -0.010212680324912071),
(-1.1999998092651367, 3.9600000381469727, 0.9682609438896179),
(0.019999980926513672, 4.180000305175781, 2.1311111450195312),
(-3.3999996185302734, 0.36000001430511475, -1.9181816577911377),
(-0.9600000381469727, 5.920000076293945, 3.154418706893921),
(-2.5799999237060547, -0.07999992370605469, -1.1899999380111694),
(0.3400000035762787, 5.460000038146973, 3.5302441120147705),
(-1.559999942779541, 0.7400000095367432, -0.37999993562698364),
(0.059999942779541016, 5.260000228881836, 2.2358977794647217),
(-0.3199999928474426, 3.1599998474121094, 1.3405263423919678),
(-0.6599999666213989, 2.5799999237060547, 1.3248648643493652),
(-1.6399999856948853, 4.239999771118164, 1.6788890361785889),
(-2.09999942779541, 1.0999999046325684, -0.47999992966651917),
(-0.2999999523162842, 4.220000267028809, 2.3029413223266),
(-1.5799999237060547, 0.880000114440918, -0.4321211874485016),
(-1.619999885559082, 0.3400000035762787, -0.7649999856948853),
(-3.4600000381469727, -0.2999999523162842, -1.5600001811981201),
(0.1399998664855957, 3.4000000953674316, 1.8366665840148926),
(-1.9800000190734863, 0.7800002098083496, -0.6241378784179688),
(-7.179999828338623, -0.6599999666213989, -4.570000171661377),
(-2.5799999237060547, 0.48000001907348633, -0.7955555319786072),
(0.3400000035762787, 2.119999885559082, 1.205384612083435),
(-0.9800000190734863, 2.8000001907348633, 0.8999999761581421),
(0.019999980926513672, 4.78000020980835, 2.6666665077209473),
(-1.2599999904632568, 2.0399999618530273, 0.4278261065483093),
(0.3400000035762787, 2.8000001907348633, 1.5463637113571167),
(-1.8600001335144043, 0.4200000762939453, -0.45047613978385925),
(-2.1999998092651367, 1.0199999809265137, -0.4299999177455902),
(0.3400000035762787, 3.0999999046325684, 1.9789472818374634),
(0.3400000035762787, 3.4000000953674316, 1.952222228050232),
(0.3400000035762787, 2.0799999237060547, 1.0600000619888306),
(-1.559999942779541, 1.380000114440918, -0.10999997705221176),
(0.019999980926513672, 1.4200000762939453, 0.6533333659172058),
(-1.9800000190734863, -0.2799999713897705, -1.0928571224212646),
(0.3400000035762787, 2.4200000762939453, 1.5338460206985474),
(-1.5999999046325684, 1.7000000476837158, 0.12666667997837067),
(-0.9800000190734863, 1.7400000095367432, 0.31272727251052856),
(-0.9800000190734863, 0.40000009536743164, -0.4299999177455902),
(-0.6399999856948853, 0.7200000286102295, 0.14444445073604584),
(-0.6399999856948853, 0.3400000035762787, -0.2199999839067459),
(-0.3199999928474426, 1.380000114440918, 0.5028571486473083),
(0.3400000035762787, 1.0399999618530273, 0.6899999976158142),
(-0.9800000190734863, 0.3400000035762787, -0.3799999952316284),
(-0.6599999666213989, 0.36000001430511475, -0.14999999105930328),
(-0.3199999928474426, 0.3400000035762787, 0.013333330862224102),
(-0.3199999928474426, 0.3400000035762787, 0.010000000536441803),
(0.3400000035762787, 0.3400000035762787, 0.3400000035762787)]
```

```
# convert list to dataframe
df = pd.DataFrame(list_min_max_mean, columns=["min", "max", "mean"],
index=list(range(K, 0)))

# plot
df.plot(kind="line", xlabel="N")
```

```
<Axes: xlabel='N'>
```



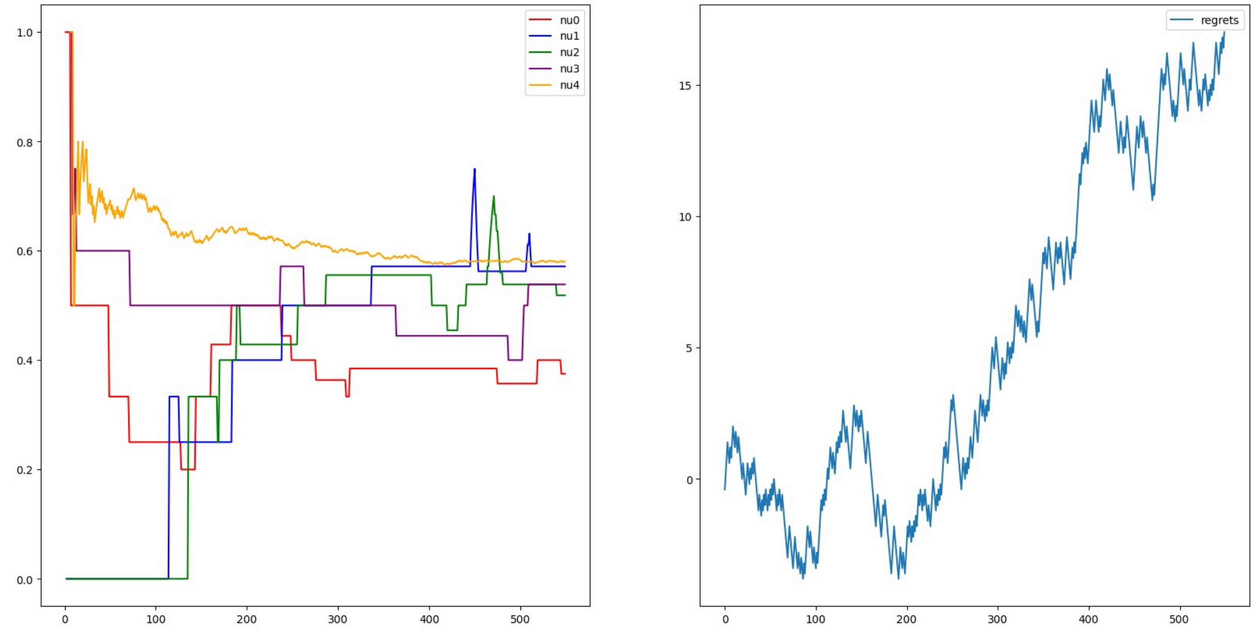
[Ajoutez votre commentaire ici]

The lower is the number of patients in exploration (N), the higher is the risk of choosing the wrong vaccin. The regret can still be low when N is low because the best vaccin can be found by "luck".

Q5. bis Nouvelle amélioration : à chaque nouveau patient, on choisit si on lui administre le meilleur vaccin avec une probabilité ϵ ou un vaccin aléatoire ($p=1-\epsilon$). Vérifiez si vous obtenez un meilleur résultat avec N = 0 ou N > 0. À votre avis, à quoi sert ϵ ?

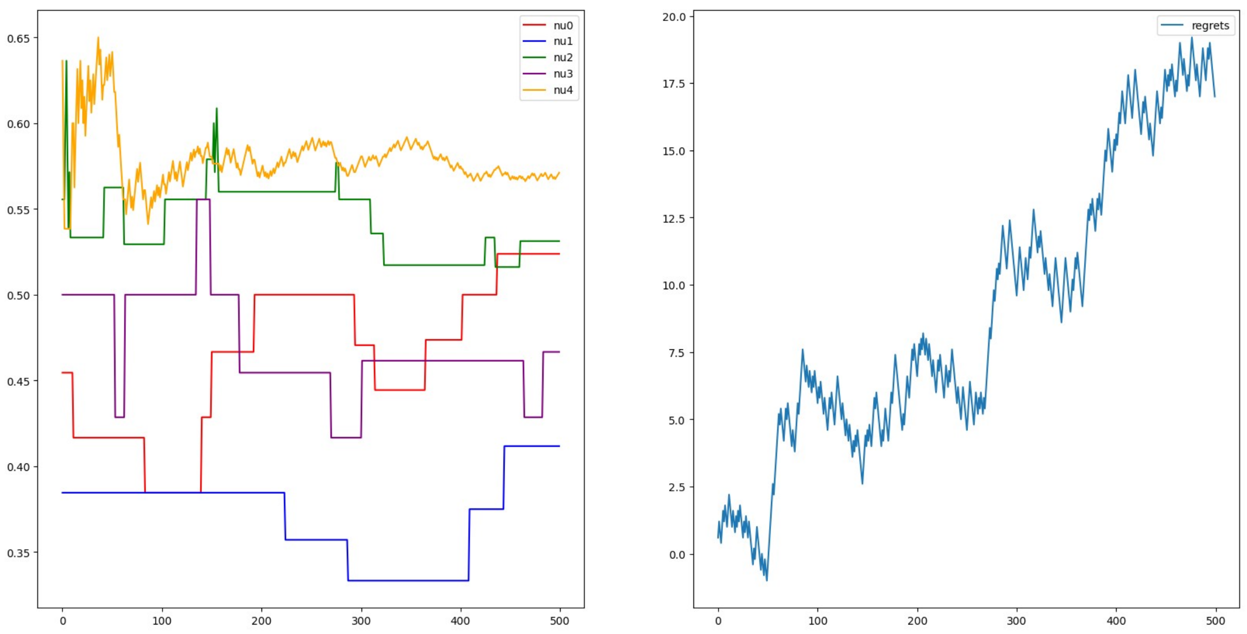
```python
MAB = generate_custom_arms(min_rate=0.4, diff_rate=0.05)

# without exploration
epsilon = 0.9
plot_a_simulation(MAB=MAB, add_bias=False,
proba_take_best_vaccin=epsilon, n_patients_exploration=0,
n_patients_exploitation=550)
```

```
Immunity rates : [0.4, 0.45, 0.5, 0.55, 0.6]
```



```python
# with exploration
epsilon = 0.9
plot_a_simulation(MAB=MAB, add_bias=False,
proba_take_best_vaccin=epsilon, n_patients_exploration=50,
n_patients_exploitation=500)
```

```
Immunity rates : [0.4, 0.45, 0.5, 0.55, 0.6]
```



```python
# regret without exploration
epsilon = 0.9
last_regrets = torch.Tensor(
    [
        simulation_update_immune_rate(MAB, add_bias=False,
proba_take_best_vaccin=epsilon, n_patients_exploration=0,
n_patients_exploitation=550)[2][-1]
        for MAB in tqdm(MABs)
    ]
)

print_results(None, last_regrets)
```

{"model_id":"414edd205d874359a9ae6018a75b0dbb","version_major":2,"version_minor":0}

```
E_exploitation_regrets : tensor(27.2443)
std_exploitation_regrets : tensor(19.6149)
std_exploitation_regrets rate : tensor(0.0392)
```

```python
# regret without exploration
epsilon = 0.9
last_regrets = torch.Tensor(
    [
        simulation_update_immune_rate(MAB, add_bias=False,
proba_take_best_vaccin=epsilon, n_patients_exploration=50,
n_patients_exploitation=500)[2][-1]
        for MAB in tqdm(MABs)
    ]
)
```

```
print_results(None, last_regrets)
```

```
{"model_id":"6ffe0d7431ce4c70b7570233069943f2","version_major":2,"version_minor":0}
```

```
E_exploitation_regrets : tensor(19.1021)
std_exploitation_regrets : tensor(10.7381)
std_exploitation_regrets rate : tensor(0.0215)
```

- Vérifiez si vous obtenez un meilleur résultat avec N = 0 ou N > 0.

The exploration phase is still useful because the regret is lower with N=50. However, the higher the number patients for the exploitation is, the lower is the difference.

- À votre avis, à quoi sert ϵ ?

ϵ is useful to eventually find the good vaccin. It avoids choosing a wrong and never change.

I.b. Borne inférieure de Lai & Robbins [Lai et Robbins, 1985]

Lai et Robbins [Lai et Robbins, 1985] considère une classe d'algorithmes π pour résoudre ce type de problèmes.

Ils ont trouvé une borne inférieure sur les récompenses cumulées en valeur asymptotique :

$$\lim_{n \to \infty} \inf_{\pi} \frac{\sum_{k=0}^{n-1} R_k}{\log n} \geq \sum_{i \text{ tel que } \mu_i < \mu^{¿}} \frac{\mu^{*} - \mu_i}{\mathrm{KL}(\mu_i, \mu^{¿})} := C(\mu)$$

avec $\mathrm{KL}(x, y) = x \log(x / y) + (1 - x) \log((1 - x)/(1 - y))$ (distance de Kullback-Leibler) et $\sum_{k=0}^{n-1} R_k$ la récompense obtenue sur n patients.

Q6. Justifiez pourquoi on peut en déduire que le regret d'un algorithme raisonnable sera au pire logarithmique.

[Ajoutez votre commentaire ici]

Q7. Tracez le regret issu de la borne de Lai & Robbins et comparez le au regret obtenu avec l'algorithme glouton.

[Ajoutez votre commentaire ici]

I.c. Upper Confidence Bounds

Cet algorithme améliore la version précédente en ajoutant un biais lié à la fréquentation de chaque vaccin :

$$\acute{\mu}_i = \hat{\mu}_i + \sqrt{\frac{C \log n}{T_i}},$$

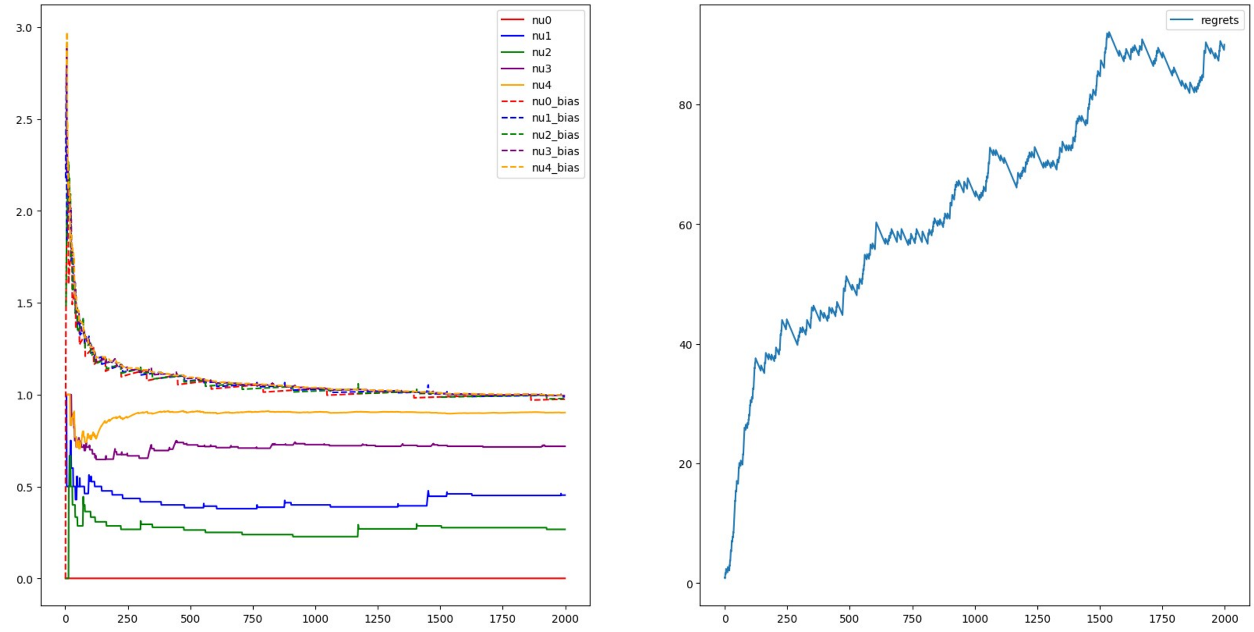
avec $C = 2$.

Q8. Implémentez la modification de cet algorithme. Observez un intérêt à conserver $N > 0$? Et $\epsilon < 1$? Expliquez pourquoi.

Dans la suite, on prendra $N = 0$ et $\epsilon = 1$.

```
MAB = generate_custom_arms(0.1, 0.2)
plot_a_simulation(MAB, n_patients_exploration=0,
n_patients_exploitation=2000, add_bias=True)
```

```
Immunity rates : [0.1, 0.3, 0.5, 0.7, 0.9]
```



[Ajoutez votre commentaire ici] There no use anymore of an exploration because even in exploitation the vaccin used change often because of the bias.

Q9. Tracez sous la forme d'une animation l'évolution des taux d'immunisation empirique (fig. de gauche) et l'évolution du regret (fig. droite). Dans la figure de gauche, vous representerez $\acute{\mu}_i$ et $\hat{\mu}_i$ pour chaque vaccin.

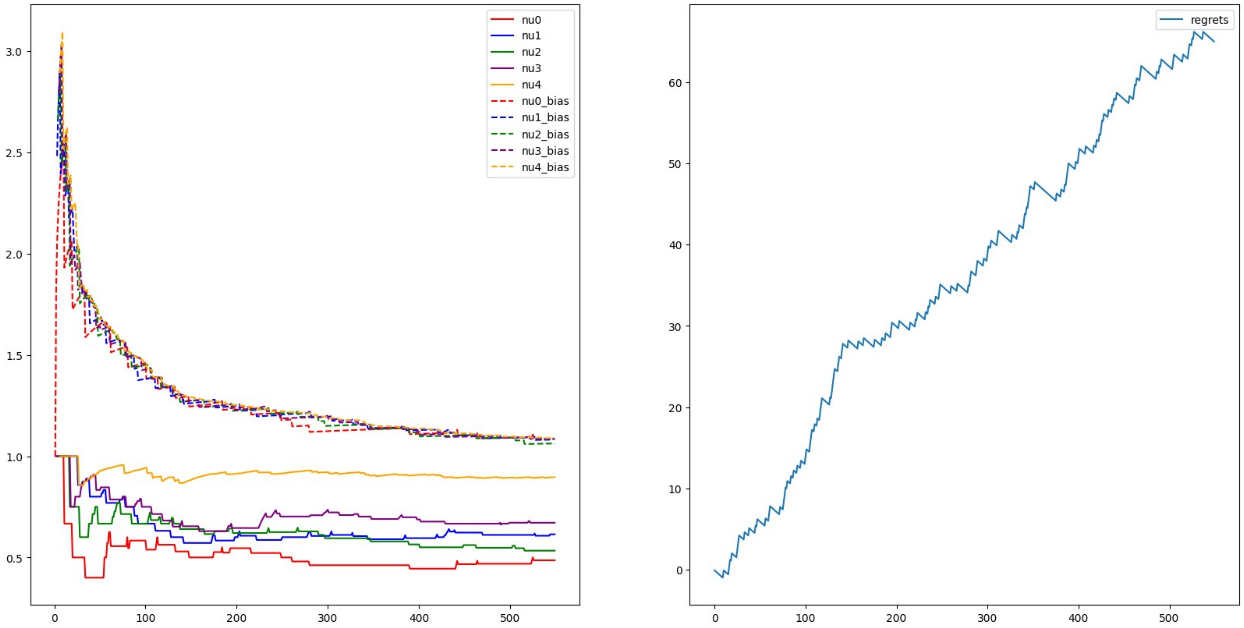
```
# See higher
```

[Ajoutez votre commentaire ici] See higher

Q10. Reprenez la question Q5 avec cette algorithme. Concluez sur l'utilité (ou l'inutilité) de la phase d'exploration. Comparez les performances d'UCB avec celles de l'algorithme glouton.

```
MAB = generate_custom_arms(0.5, 0.1)
epsilon = 0.9
plot_a_simulation(MAB, n_patients_exploration=0,
n_patients_exploitation=550, add_bias=True,
proba_take_best_vaccin=epsilon)
```

```
Immunity rates : [0.5, 0.6, 0.7, 0.8, 0.9]
```



```
# UCB without exploration with epsilon (sometimes do not take the best
vaccin but a random one)
last_regrets = torch.Tensor(
    [
        simulation_update_immune_rate(MAB, add_bias=True,
proba_take_best_vaccin=epsilon, n_patients_exploration=0,
n_patients_exploitation=550)[2][-1]
        for MAB in tqdm(MABs)
    ]
)

print_results(None, last_regrets)
```

{"model_id":"a9d3fff3ac68466b9a2cc21c1028cadf","version_major":2,"version_minor":0}

```
E_exploitation_regrets : tensor(54.2643)
std_exploitation_regrets : tensor(15.0689)
std_exploitation_regrets rate : tensor(0.0301)
```

```
# UCB with exploration
last_regrets = torch.Tensor(
    [
        simulation_update_immune_rate(MAB, add_bias=True,
proba_take_best_vaccin=epsilon, n_patients_exploration=50,
n_patients_exploitation=500)[2][-1]
        for MAB in tqdm(MABs)
    ]
)

print_results(None, last_regrets)
```

{"model_id":"8244212a1b56444d845477790aa7118e","version_major":2,"version_minor":0}

```
E_exploitation_regrets : tensor(40.0821)
std_exploitation_regrets : tensor(12.4753)
std_exploitation_regrets rate : tensor(0.0250)
```

[Ajoutez votre commentaire ici]

- Concluez sur l'utilité (ou l'inutilité) de la phase d'exploration.

The exploration phase is still useful because the regrets is lower with. I have to admit that I do not know why.

```
# Glouton with exploration
last_regrets = torch.Tensor(
    [
        simulation_update_immune_rate(MAB, add_bias=False,
proba_take_best_vaccin=epsilon, n_patients_exploration=50,
n_patients_exploitation=500)[2][-1]
        for MAB in tqdm(MABs)
    ]
)

print_results(None, last_regrets)
```

{"model_id":"89d2eb97738040458f980e49a9e1ed00","version_major":2,"version_minor":0}

```
E_exploitation_regrets : tensor(20.5821)
std_exploitation_regrets : tensor(11.6055)
std_exploitation_regrets rate : tensor(0.0232)
```

- Comparez les performances d'UCB avec celles de l'algorithme glouton.

Glouton is better. The bias might help to find the best vaccin but it might help on too few situation and the cost is to high.

Q11. Testez différentes valeurs pour C et trouvez sa valeur optimale expérimentalement.

```
for C in tqdm(range(1, 5)):
    last_regrets = torch.Tensor(
    [
        simulation_update_immune_rate(MAB, add_bias=True,
proba_take_best_vaccin=1, n_patients_exploration=50,
n_patients_exploitation=500)[2][-1]
        for MAB in tqdm(MABs)
    ]
    )

    print("C :", C)
    print_results(None, last_regrets)
```

{"model_id":"b5be28fb94dc44d6b10facdb3fec4797","version_major":2,"version_minor":0}

{"model_id":"24a1a7e2b62343a9b0c3eeba688aa4a7","version_major":2,"version_minor":0}

```
C : 1
E_exploitation_regrets : tensor(17.2821)
std_exploitation_regrets : tensor(11.2624)
std_exploitation_regrets rate : tensor(0.0225)
```

{"model_id":"795798f696344c0c9730552ac9642812","version_major":2,"version_minor":0}

```
C : 2
E_exploitation_regrets : tensor(35.3121)
std_exploitation_regrets : tensor(13.6438)
std_exploitation_regrets rate : tensor(0.0273)
```

{"model_id":"163037de8a40488b9a7e8f1e3e772d2a","version_major":2,"version_minor":0}

```
C : 3
E_exploitation_regrets : tensor(44.6021)
std_exploitation_regrets : tensor(13.6911)
std_exploitation_regrets rate : tensor(0.0274)
```

{"model_id":"86b1d3a999e94e27841281233601fac5","version_major":2,"version_minor":0}

```
C : 4
E_exploitation_regrets : tensor(52.0121)
std_exploitation_regrets : tensor(14.7403)
std_exploitation_regrets rate : tensor(0.0295)
```

[Ajoutez votre commentaire ici]

The best C is the lower one (so 0), it might have a mistake in my code to find these results.

Echantillonnage de Thomson

Cet algorithme propose de modéliser la variable aléatoire de chaque vaccin avec une loi β dont les paramètres a et b correspondent au nombre de patients que le vaccin a immunisés (resp. non immunisés).

Pour chaque patient, on tire un valeur aléatoire pour la loi β décrivant chaque vaccin, puis on choisit le vaccin avec la plus grande valeur tirée.

Q12. Implémentez cet algorithme. En testant plusieurs valeurs de N, montrez que la phase d'exploration précédente a un impact très limité. Cela veut-il dire que l'algorithme ne contient pas d'initialisation ?

[Ajoutez votre commentaire ici]

Q13. Tracez sous la forme d'une animation l'évolution des taux d'immunisation empirique (fig. de gauche) et l'évolution du regret (fig. droite). Dans la figure de gauche, vous representerez le taux d'immunisation empirique pour chaque vaccin avec un graphique en violon qui représente la loi beta associée à chaque vaccin.

[Ajoutez votre commentaire ici]

Q14. Comparez le regret avec les autres algorithmes.

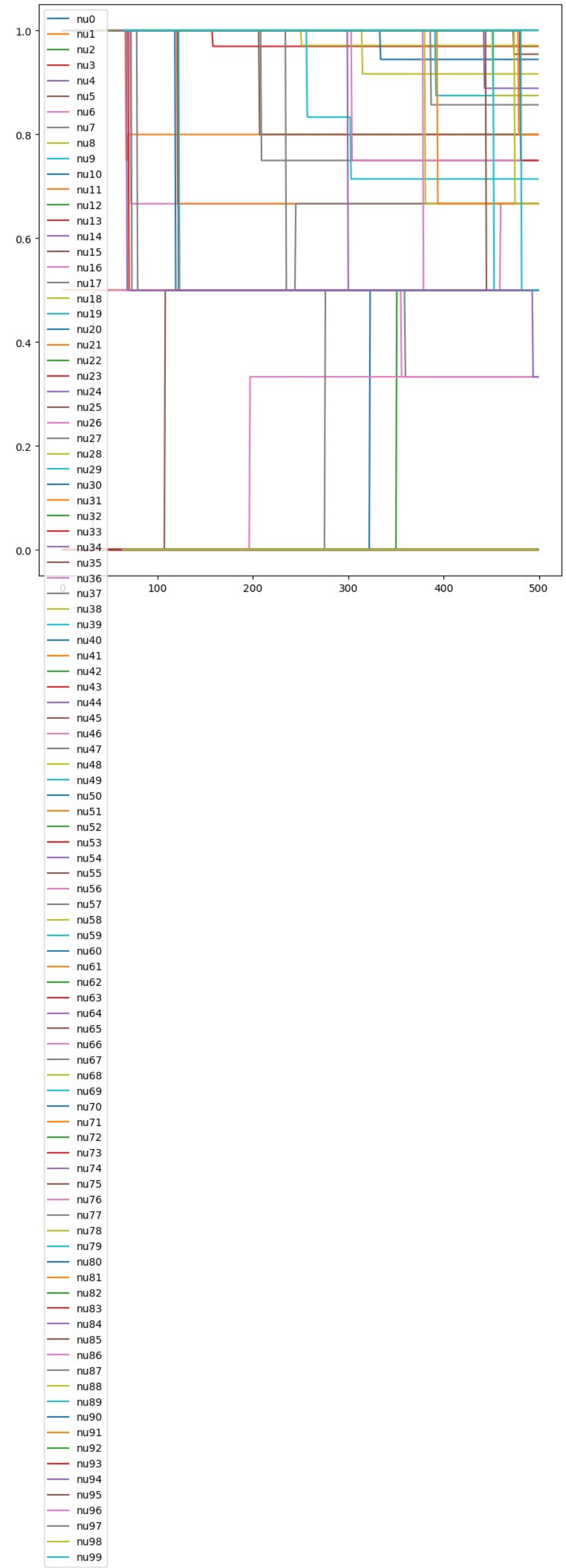
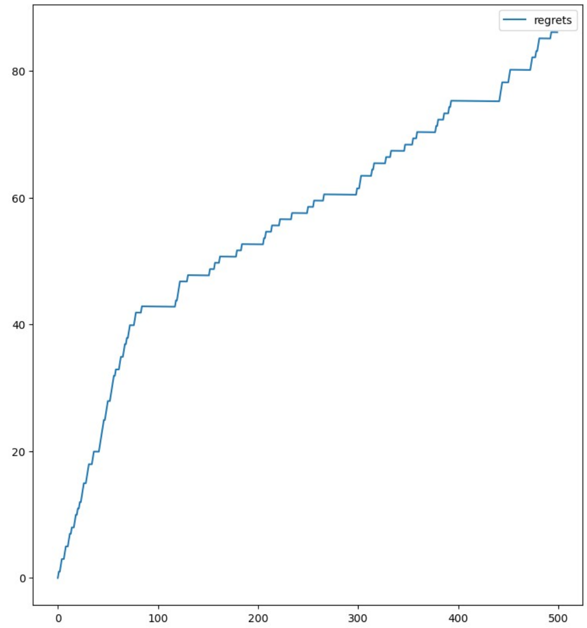
[Ajoutez votre commentaire ici]

Conclusion

Q15. Calculez le regret des algorithmes glouton, UCB & Thomson lorsqu'il y a un grand nombre de vaccins disponibles (K=100) (on prendra N=100). Faites le lien avec la malédiction de la dimension.

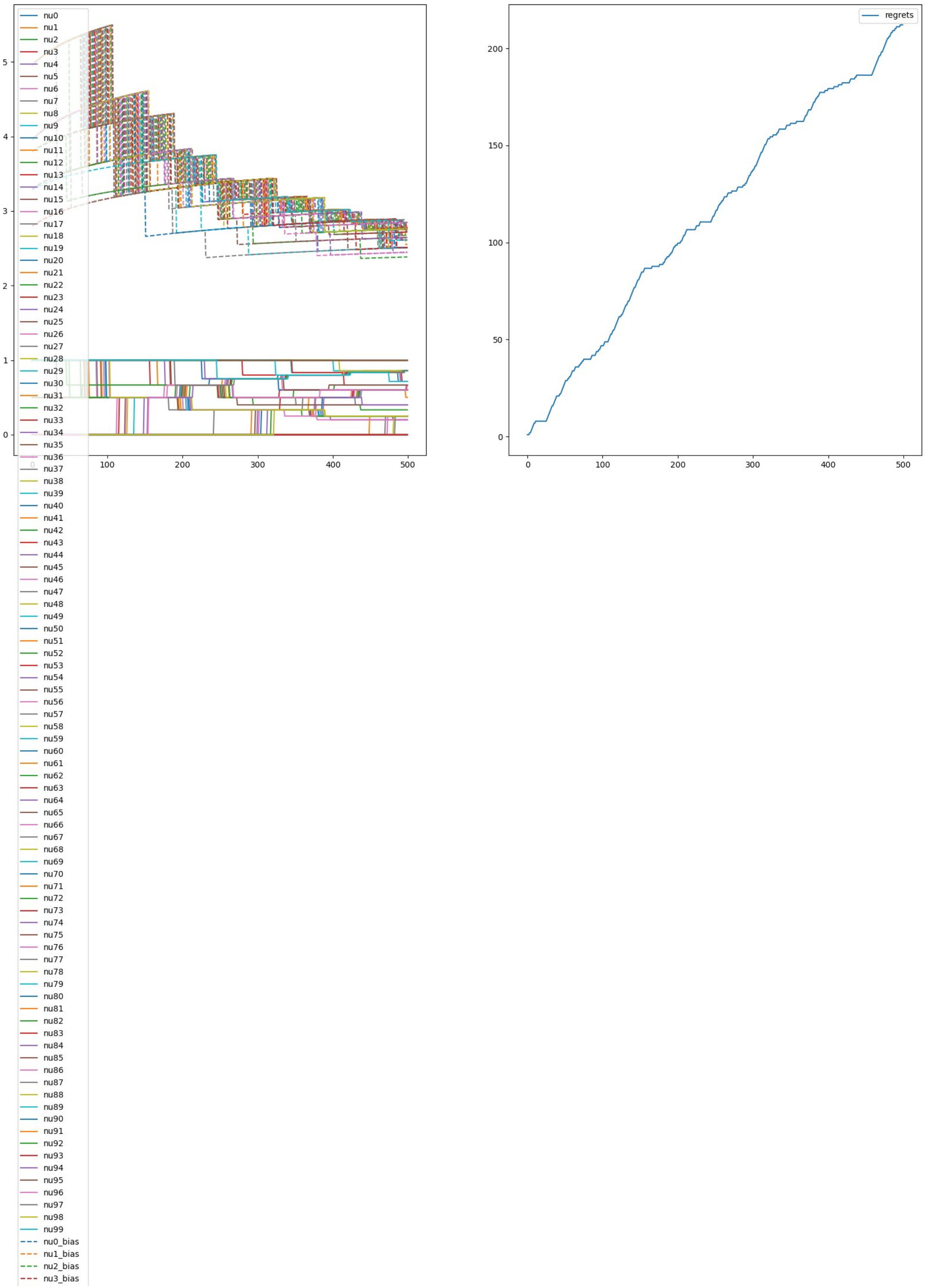
```
MABs100 = [generate_arms(100) for _ in range(100)]

# Glouton with exploration and with epsilon
plot_a_simulation(MABs100[0], n_patients_exploration=50,
n_patients_exploitation=500, add_bias=False,
proba_take_best_vaccin=0.9)
```

```
Immunity rates : [0.228, 0.86, 0.09, 0.372, 0.117, 0.543, 0.369,
0.568, 0.211, 0.997, 0.775, 0.65, 0.01, 0.737, 0.389, 0.362, 0.16,
0.851, 0.197, 0.72, 0.614, 0.704, 0.556, 0.959, 0.922, 0.667, 0.249,
0.707, 0.963, 0.845, 0.772, 0.664, 0.966, 0.207, 0.651, 0.634, 0.691,
0.221, 0.94, 0.181, 0.966, 0.998, 0.28, 0.29, 0.206, 0.15, 0.609,
0.229, 0.404, 0.19, 0.285, 0.02, 0.895, 0.053, 0.808, 0.845, 0.111,
0.794, 0.081, 0.568, 0.061, 0.316, 0.977, 0.036, 0.854, 0.136, 0.271,
0.195, 0.714, 0.75, 0.55, 0.0, 0.45, 0.264, 0.193, 0.929, 0.0, 0.116,
0.342, 0.256, 0.09, 0.849, 0.171, 0.718, 0.581, 0.732, 0.139, 0.25,
0.639, 0.677, 0.182, 0.992, 0.682, 0.017, 0.296, 0.957, 0.535, 0.919,
0.108, 0.927]
```


1.0
0.8
0.6
0.4
0.2
0.0
0
100
200
300
400
500
nu0
nu1
nu2
nu3
nu4
nu5
nu6
nu7
nu8
nu9
nu10
nu11
nu12
nu13
nu14
nu15
nu16
nu17
nu18
nu19
nu20
nu21
nu22
nu23
nu24
nu25
nu26
nu27
nu28
nu29
nu30
nu31
nu32
nu33
nu34
nu35
nu36
nu37
nu38
nu39
nu40
nu41
nu42
nu43
nu44
nu45
nu46
nu47
nu48
nu49
nu50
nu51
nu52
nu53
nu54
nu55
nu56
nu57
nu58
nu59
nu60
nu61
nu62
nu63
nu64
nu65
nu66
nu67
nu68
nu69
nu70
nu71
nu72
nu73
nu74
nu75
nu76
nu77
nu78
nu79
nu80
nu81
nu82
nu83
nu84
nu85
nu86
nu87
nu88
nu89
nu90
nu91
nu92
nu93
nu94
nu95
nu96
nu97
nu98
nu99

regrets
80
60
40
20
0
0
100
200
300
400
500

```
# UCB with exploration and with epsilon
plot_a_simulation(MABs100[0], n_patients_exploration=50,
n_patients_exploitation=500, add_bias=True,
proba_take_best_vaccin=0.9)
```

```
Immunity rates : [0.228, 0.86, 0.09, 0.372, 0.117, 0.543, 0.369,
0.568, 0.211, 0.997, 0.775, 0.65, 0.01, 0.737, 0.389, 0.362, 0.16,
0.851, 0.197, 0.72, 0.614, 0.704, 0.556, 0.959, 0.922, 0.667, 0.249,
0.707, 0.963, 0.845, 0.772, 0.664, 0.966, 0.207, 0.651, 0.634, 0.691,
0.221, 0.94, 0.181, 0.966, 0.998, 0.28, 0.29, 0.206, 0.15, 0.609,
0.229, 0.404, 0.19, 0.285, 0.02, 0.895, 0.053, 0.808, 0.845, 0.111,
0.794, 0.081, 0.568, 0.061, 0.316, 0.977, 0.036, 0.854, 0.136, 0.271,
0.195, 0.714, 0.75, 0.55, 0.0, 0.45, 0.264, 0.193, 0.929, 0.0, 0.116,
0.342, 0.256, 0.09, 0.849, 0.171, 0.718, 0.581, 0.732, 0.139, 0.25,
0.639, 0.677, 0.182, 0.992, 0.682, 0.017, 0.296, 0.957, 0.535, 0.919,
0.108, 0.927]
```



nu0
nu1
nu2
nu3
nu4
nu5
nu6
nu7
nu8
nu9
nu10
nu11
nu12
nu13
nu14
nu15
nu16
nu17
nu18
nu19
nu20
nu21
nu22
nu23
nu24
nu25
nu26
nu27
nu28
nu29
nu30
nu31
nu32
nu33
nu34
nu35
nu36
nu37
nu38
nu39
nu40
nu41
nu42
nu43
nu44
nu45
nu46
nu47
nu48
nu49
nu50
nu51
nu52
nu53
nu54
nu55
nu56
nu57
nu58
nu59
nu60
nu61
nu62
nu63
nu64
nu65
nu66
nu67
nu68
nu69
nu70
nu71
nu72
nu73
nu74
nu75
nu76
nu77
nu78
nu79
nu80
nu81
nu82
nu83
nu84
nu85
nu86
nu87
nu88
nu89
nu90
nu91
nu92
nu93
nu94
nu95
nu96
nu97
nu98
nu99
nu0_bias
nu1_bias
nu2_bias
nu3_bias
5
4
3
2
1
0
0
100
200
300
400
500
regrets
200
150
100
50
0
0
100
200
300
400
500

With a such a high dimension, the best vaccin or a good vaccin would take too much exploration too be found. Even the bias or the epsilon could not help enough to counter the dimension malediction.