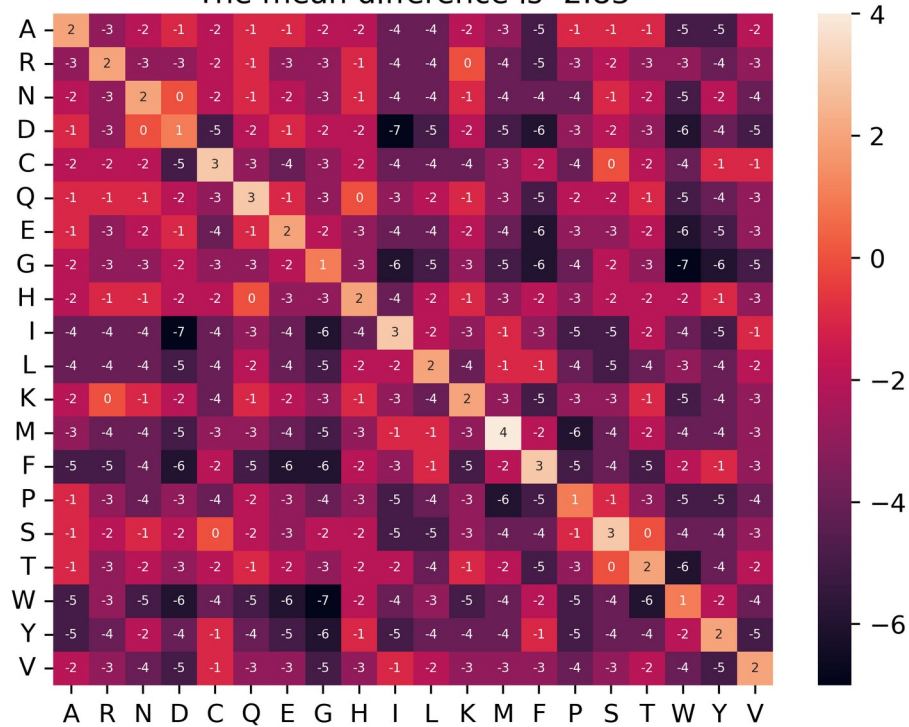


Recalcul de Blosom avec pid et clustering
corrigés

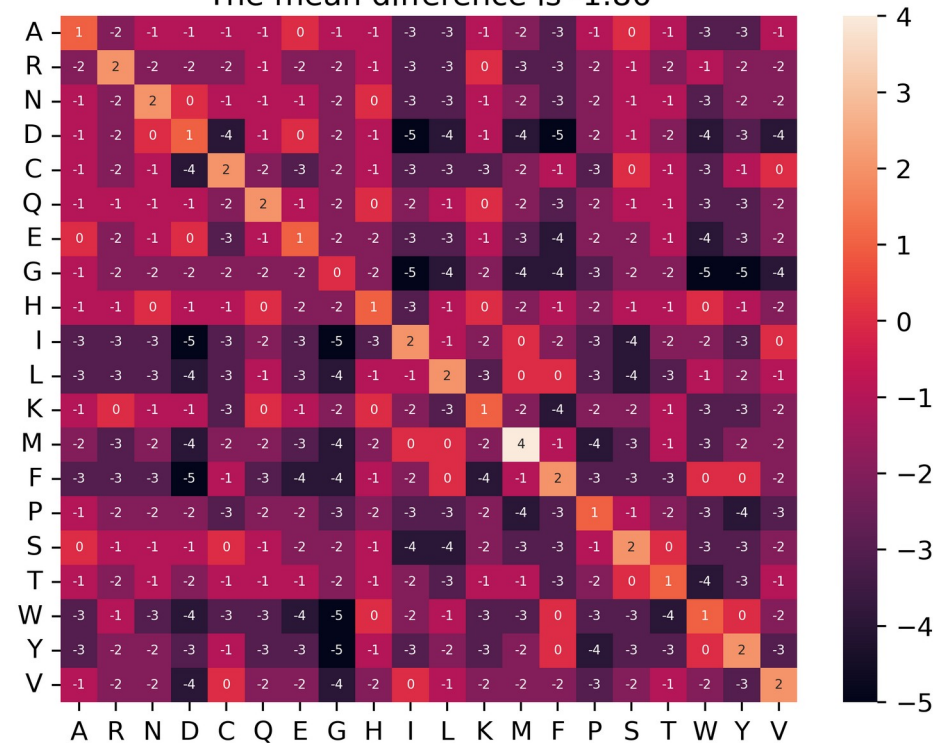
Variante blosum62 (5min)

Heatmap of the difference in Score between Blosum(Pfam_train)
and Blosum62Ref
The mean difference is -2.83



Variante blosum50 (20min)

Heatmap of the difference in Score between Blosum(Pfam_train)
and Blosum62Ref
The mean difference is -1.86

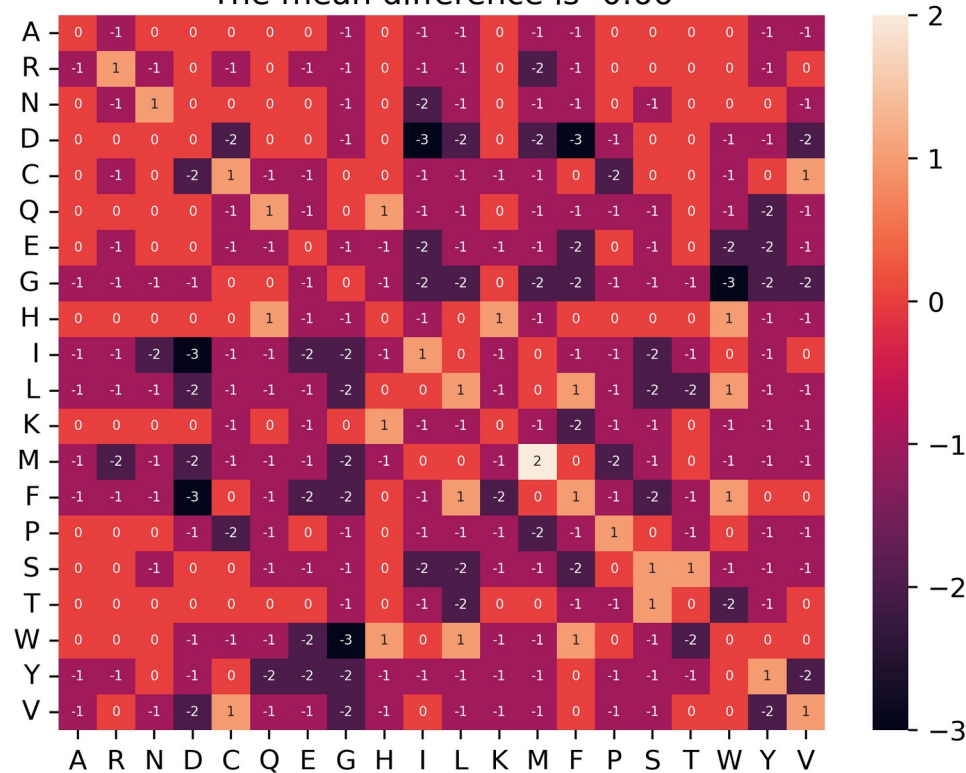


Variante blosum30 (1,5h)

Variante blosum0

Heatmap of the difference in Score between Blosum(Pfam_train)
and Blosum62Ref

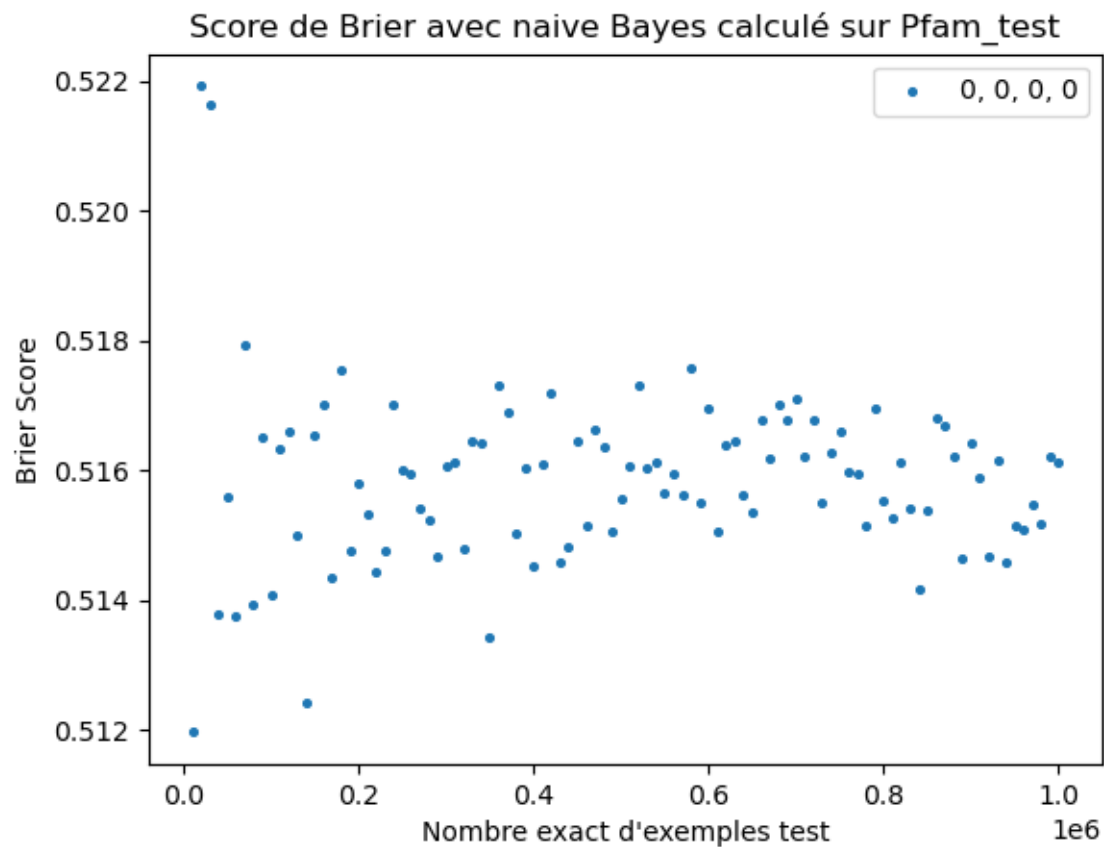
The mean difference is -0.66



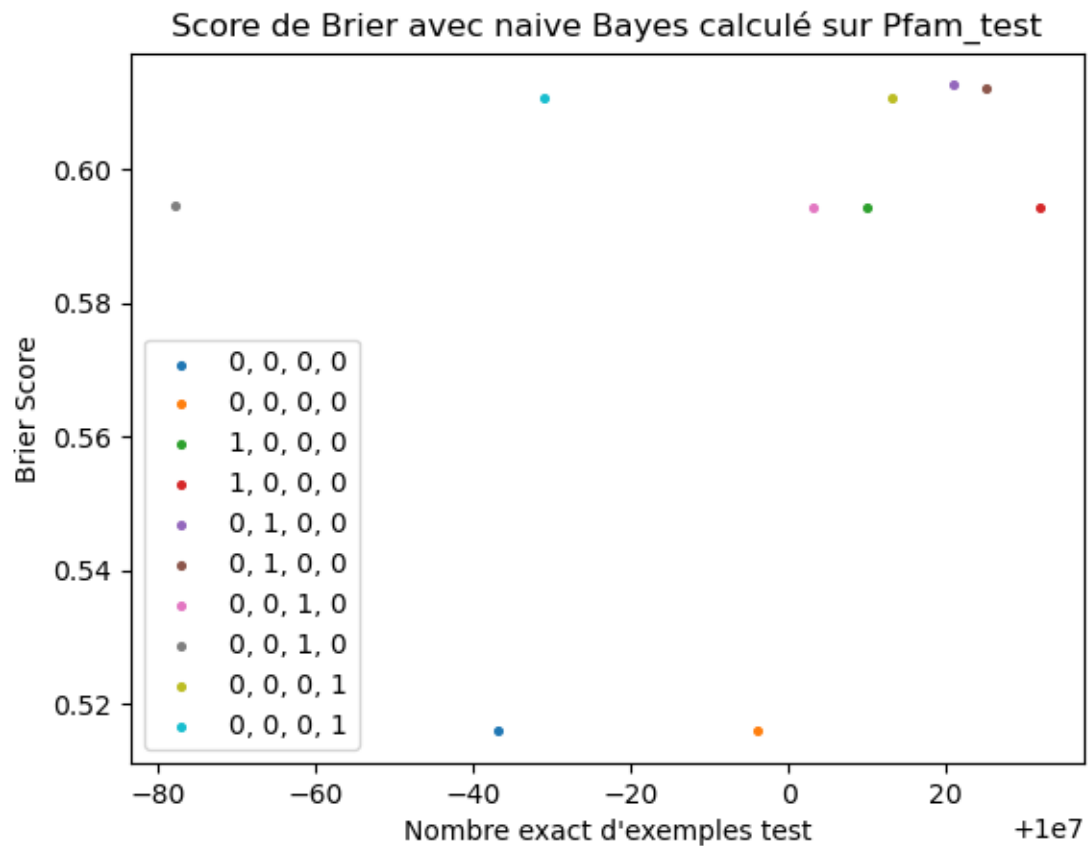
Recalcul de Scores de Brier avec Bayes Naif

(avec pid = 62, temps de calcul 13min)

Stabilisation du Score (test de 10m à 1M d'exemples)



Sur 10M d'exemples demandés



ANNEXES

Prob affichage/ veille

brier_naive_bayes_v1.py - MNHN_EvolProt - Visual Studio Code

File Edit Selection View Go Run Terminal Help

EXPLORER

- ▼ MNHN_EVOLPROT
 - selection_ex... M
 - > cluster
 - > dossier_test_score...
 - ▼ localNeighbour
 - > __pycache__
 - localNeighbourfo...
 - scriptCube_1DK.py
 - scriptCube_1DP.py
 - scriptCube_1GK.py
 - scriptCube_1GP.py
 - scriptCubeRef.py
 - scriptCubeRef.sh
 - ▼ test_prelimin...
 - brier_naive... M
 - draft_seed_select...
 - main_brier_no_co...
 - > treatment
 - > utils
 - main_blosum.py M
 - main_brier_no_con...
 - main_data_treatm...
 - main_local_neigh...
 - main_script_cube...
 - pfam_residu_desc.py
 - README.md
 - seed_non_informat...
 - .gitignore
 - Résultats_surpr... U
 - > OUTLINE
 - > TIMELINE

MNHN > test_preliminaire_voisinage_local > brier_naive_bayes_v1.py > naive_bayes_brier

```
1 import numpy as np
2 import matplotlib.pyplot as plt
3 import os
4
5 import sys
6 from pathlib import Path
7 file = Path(__file__).resolve()
8 package_root_directory_MNHN = file.parents[2]
9 root_path = file.parents[3]
10 sys.path.append(str(package_root_directory_MNHN))
11
12 import MNHN.brierNeighbour.selection_example as selection_example
13 from MNHN.utils.timer import Timer
14
15 def cube_loader(max_relative_distance, k_or_p, l_or_r, path_cube_folder):
16     """
17     max_relative_distance: indice le plus lointain dans quart de fenetre
18     k_or_p: séquence d'origine (k) ou de destination (p)
19     l_or_r: voisinage à gauche ou à droite
20     """
21     # initialisation de la liste des cubes pour un quart de fenetre contextuelle
22     list_cube_quarter_window = []
23
24     for i in range(1, max_relative_distance + 1):
25         if l_or_r == "l":
26             path_cube = f"{path_cube_folder}/proba_cond_{i},{k_or_p}.npz"
27             list_cube_quarter_window.append(np.load(path_cube, allow_pickle='TRUE').
28
29         if l_or_r == "r": # je sais qu'un else marche mais if pour le moment
30             path_cube = f"{path_cube_folder}/proba_cond_{i},{k_or_p}.npz" # à vér
31             list_cube_quarter_window.append(np.load(path_cube, allow_pickle='TRUE').
32
33     return list_cube_quarter_window
34
35
```

Ln 129, Col 50 Spaces: 4 UTF-8 LF Python 3.9.13 ('base': conda)

Temps clustering 99 % Pfam et Pfam en sortie de traitement

```
99.99490627546862, PF17407.5
---> time redundant: 2.03332 s
100.0, PF02713.17
---> time redundant: 0.02593 s
---> Compute and save non-redundant files: 59154.88105 s
nbre seed:                19 632.00
nbre seq:                  1 235 590.00
nbre position:             4 448 999.00
total character:           346 322 402.00
total character included:  192 394 396.00
mean len seq:              155.71
mean nbre seq:             62.94
---> Split data total in data A and data B: 2.01514 s
(base) pauline@abiboom:~/MNHN_EvolProt$
```