Voyage initiatique vers la bioinformatique : la récompense est au bout du chemin

Alban Mancheron

VOYAGE INITIATIQUE VERS LA BIOINFORMATIQUE : LA RÉCOMPENSE EST AU BOUT DU CHEMIN

Alban MANCHERON
[Enseignant-chercheur en bioinformatique à l'université de Montpellier, linuxien depuis 1997 (convaincu depuis 1998)]

Mots-clés : EMBOSS, bioinformatique, analyse, génomique comparative, shell, SARS-CoV-2

Dans les précédents articles [1, 2], nous avions découvert quelques outils d'analyse bioinformatique et procédé à une première analyse d'une région d'intérêt du virus SARS-CoV-2. Nous allons avec ce dernier opus finir d'explorer la structure de cet organisme et illustrer encore une fois l'incidence du modèle linuxien sur les progrès dans les sciences.

Résumé des épisodes précédents **[1,2]** : à partir de la séquence de référence du virus *SARS-CoV-2* (`NC_045512.2`), nous avons détecté 191 régions potentiellement à l'origine des protéines que le virus fait synthétiser par son hôte pour pouvoir se reproduire. De ces 191 régions, sur la base de critères assez basiques, nous en avons sélectionné 36 *a priori* intéressantes. Nous avons mené quelques investigations sur la première de ces régions et montré qu'elle correspondait à une protéine connue chez d'autres coronavirus, à la base de leur capacité à se reproduire.

Nous allons aujourd'hui généraliser la mécanique de notre étude pour l'appliquer aux 35 régions restantes. Et comme annoncé dans le précédent article, si vous prenez le train en route, tout n'est pas perdu (cf. note) !

note

Les données récupérées, les résultats obtenus et les scripts développés dans le cadre de cette série d'articles sont disponibles sur la forge logicielle **GitLab** hébergée par **Framasoft** : **https://framagit.org/doccy/SARS-CoV-2-bioinfo-analysis**.

1. RAPPEL DES ÉTAPES D'ANALYSE

La première région que nous avions étudiée correspondait au 81e ORF (*Open Reading Frame* – cadre ouvert de lecture) dont la traduction sous forme de séquence d'acides aminés est dans le fichier `results/ORF/nc_045512.2_81.fasta`.

1.1 La démarche qui a marché

Cette séquence a été confrontée à la base de données `UniProtKB/Swiss-Prot(swissprot)` *via* l'outil **BLAST [3]** (nous utilisons la version pour les protéines) disponible en ligne sur le site du **NCBI [4]**. Pour cela, il a suffi de téléverser la séquence dans le formulaire, de sélectionner la base de données, d'exclure les informations relatives au *SARS-CoV-2* pour le *fairplay* (puisque nous étudions l'organisme comme s'il venait juste d'être découvert), puis de cliquer sur le bouton de soumission.

Après quelques instants, nous avons récupéré les 100 séquences (limite par défaut) les plus significativement proches et nous avons sélectionné les séquences qui s'alignaient sur plus de 80 % de notre séquence (en triant par ordre décroissant au niveau de la colonne `query cover`). Nous les avons téléchargées (menu déroulant `download`) au format **GenBank** dans le fichier `results/ORF/nc_045512.2_81_related.gb`. À partir de cet ensemble, nous avons construit un fichier contenant toutes ces séquences ainsi que notre séquence requête, puis nous avons réalisé un alignement multiple avec le programme **ClustalΩ** (en passant par son *wrapper* **emma** fourni par la suite **EMBOSS [5,6]**). Nous avons généré quelques images, notamment *via* l'outil **plotcon** de la suite EMBOSS (qui illustre le degré de conservation sur une fenêtre glissante d'un alignement) et les outils **newicktotxt** et **newicktops** du programme **NJplot** (qui permettent de montrer l'arbre de guidage généré lors de l'alignement ; si la séquence requête est éloignée des autres séquences, cela se voit). Ceci nous a permis de constater que les séquences se divisaient en deux groupes, d'un côté les séquences concernant une protéine appelée *Replicase polyprotein 1ab*, et de l'autre celles concernant une protéine appelée *Replicase polyprotein 1a*. Notre séquence requête se situant visiblement dans ce second groupe, nous avons réitéré la même analyse en ne considérant que les séquences codant pour la *Replicase polyprotein 1a*. Cette fois-ci, l'alignement produit a été bien plus concluant et nous a permis d'établir avec confiance que notre ORF pouvait être considéré *de facto* comme une CDS (*Coding Data Sequence*) associée au gène de la *Replicase polyprotein 1a*. Ceci est d'autant plus convaincant que les séquences récupérées par BLAST correspondent toutes à des coronavirus et que la *Replicase polyprotein 1a* est une sorte de super protéine qui permet aux virus qui en disposent de se répliquer.

L'automatisation de ces étapes consiste en deux parties, d'une part faire la recherche de séquences proches dans la base de données `UniProtKB/Swiss-Prot(swissprot)` avec BLAST en ligne, et d'autre part dérouler les programmes à la suite les uns des autres.

1.2 La recherche des séquences proches

La bonne nouvelle, c'est que BLAST nous autorise à soumettre plusieurs séquences en une fois. Donc finalement, la partie téléversement et paramétrage n'a pas besoin d'être automatisée.

Il faut donc créer un fichier contenant nos séquences, ce qui ne pose pas de problème majeur vu que nous disposons de toutes nos séquences individuelles dans le répertoire `results/ORF/` et qu'elles ont toutes le même

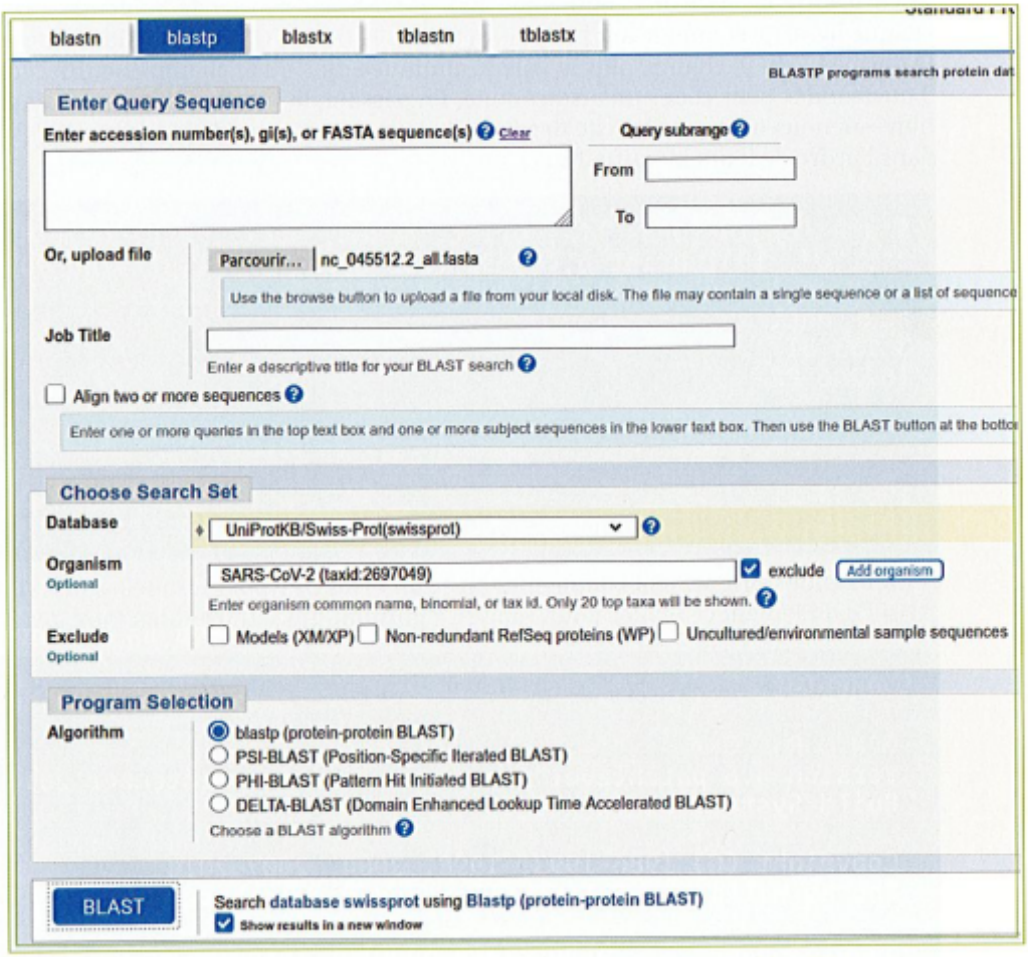

Fig. 1 : Formulaire de soumission des 36 régions d'intérêt sur le site de NCBI.

préfixe **nc_045512.2_** suivi du numéro de l'ORF suivi de l'extension **.fasta**. Notez au passage l'utilisation de l'option **--sort=version** qui permet de trier les fichiers comme s'il s'agissait de numéros de version. Ainsi le fichier **nc_045512.2_81.fasta** est bien avant le fichier **nc_045512.2_106.fasta** :

```
$ ls --sort=version results/ORF/nc_045512.2_+([0-9]).fasta | xargs cat > results/
ORF/nc_045512.2_all.fasta
$ seqcount results/ORF/nc_045512.2_all.fasta -auto -stdout
36
```

Pour la soumission, il suffit donc de téléverser ce fichier, de sélectionner la base de données **UniProtKB/Swiss-Prot(swissprot)** et d'exclure les informations relatives au *SARS-CoV-2* (cf. figure 1, page précédente).

Une fois la demande soumise, nous obtenons assez rapidement tous les résultats de nos requêtes. Pour sélectionner les résultats d'une requête, il suffit d'aller dans le menu déroulant associé à la ligne ***Results for***.

Les résultats précédés d'un astérisque sont ceux pour lesquels aucune séquence semblable n'a été retrouvée. Sur les 36 ORF, il y en a 22 qui ne donnent aucun résultat (3, 16, 24, 36, 37, 43, 58, 80, 82, 83, 94, 106, 122, 139, 160, 162, 171, 174, 188, 189, 190 et 191). Pour les 14 autres (81, 137, 159, 167, 168, 169, 172, 173, 175, 176, 179, 180, 183 et 187), il y a des résultats. Le premier est l'ORF n°81 que nous avons déjà étudié et pour lequel nous retrouvons les mêmes séquences (fort heureusement).

Commençons par étudier les ORF pour lesquels nous n'avons pas trouvé de résultats et calculons les tailles des ORF et des chaînes d'acides aminés qui en résulteraient. Pour cela, nous partons du fichier GFF que nous avions produit la dernière fois (**results/nc_045512.2_ORF_filtered.gff**) et qui n'est autre qu'un fichier tabulé contenant soit des commentaires ou des métadonnées commençant par un **#** (c'est un **dièse**, pas un *hashtag* !), soit des lignes dont le 6e champ correspond à la taille de l'ORF + 3 (pour le codon *stop* **[1]**) et le 9e champ a systématiquement – parce que nous l'avons écrit ainsi – la forme **ID=nc_045512.2_<ID>; name=ORF_<ID>** (où **<ID>** correspond au numéro de l'ORF). Ainsi, il suffit d'écrire une commande **awk** qui consiste à afficher, au démarrage, nos entêtes de colonnes ; puis pour chaque ligne ne commençant pas par un **#** dans le fichier GFF, l'identifiant obtenu en supprimant tout ce qui précède le dernier **_** du 9e champ, puis la valeur diminuée de **3** du 6e champ et enfin cette valeur divisée par **3** puisqu'il faut 3 nucléotides pour coder un acide aminé. En passant, le résultat de cette commande à la commande **sort** (avec l'option **-n**), nous obtenons la liste des ORF, de leur longueur et de la longueur des chaînes qu'ils pourraient produire, dans l'ordre de leurs identifiants :

```
$ awk 'BEGIN {OFS="\t"; print "#ORF","nucl","aa"} (/^[^#]/) { o = gensub(".*_",
"", "g", $9); print o,($6-3),(($6-3)/3) }' results/nc_045512.2_ORF_filtered.gff |
sort -n | tee results/nc_045512.2_ORF_filtered.csv
#ORF nucl aa
3 153 51
16 246 82

...

190 33 11
191 36 12
```

Nous allons simplement commenter (avec un **#**) les ORF pour lesquelles BLAST a donné un résultat et nous pouvons passer rapidement ce fichier pour analyse à **gnuplot** qui sait très bien faire des statistiques :

```
$ gnuplot

 G N U P L O T
 Version 4.6 patchlevel 6    last modified September 2014
 Build System: Linux x86_64

 Copyright (C) 1986-1993, 1998, 2004, 2007-2014
 Thomas Williams, Colin Kelley and many others

 gnuplot home:     http://www.gnuplot.info
```

```
 faq, bugs, etc:   type "help FAQ"
 immediate help:   type "help"  (plot window: hit 'h')

Terminal type set to 'x11'
gnuplot> stats "results/nc_045512.2_ORF_filtered.csv" using 1:2

* FILE:
  Records:       22
  Out of range:  0
  Invalid:       0
  Blank:         0
  Data Blocks:   1

* COLUMNS:
  Mean:        106.7273          136.0909
  Std Dev:      63.0369           67.3349
  Sum:        2348.0000         2994.0000
  Sum Sq.:  338016.0000       507204.0000

  Minimum:       3.0000 [ 0]       30.0000 [18]
  Maximum:     191.0000 [21]      261.0000 [13]
  Quartile:     43.0000           72.0000
  Median:      100.0000          153.0000
  Quartile:    171.0000          177.0000

  Linear Model: y = -0.5371 x + 193.4
  Correlation:  r = -0.5028
  Sum xy:       2.726e+05

gnuplot> stats "results/nc_045512.2_ORF_filtered.csv" using 1:3

* FILE:
  Records:       22
  Out of range:  0
  Invalid:       0
  Blank:         0
  Data Blocks:   1

* COLUMNS:
  Mean:        106.7273           45.3636
  Std Dev:      63.0369           22.4450
  Sum:        2348.0000          998.0000
  Sum Sq.:  338016.0000        56356.0000

  Minimum:       3.0000 [ 0]       10.0000 [18]
  Maximum:     191.0000 [21]       87.0000 [13]
  Quartile:     43.0000           24.0000
  Median:      100.0000           51.0000
  Quartile:    171.0000           59.0000

  Linear Model: y = -0.179 x + 64.47
  Correlation:  r = -0.5028
  Sum xy:       9.086e+04

gnuplot> exit
```

Cette rapide statistique permet de voir que les tailles des ORF non retrouvés vont de 30 nucléotides (10aa – acides aminés) à 261 nucléotides (87aa), la moitié faisant moins de 153 nucléotides (51aa) et les trois quarts faisant moins de 177 nucléotides (59aa), donc il n'y a rien de surprenant à ce que ces ORF n'aient pas été retrouvés dans la base de données **UniProtKB/Swiss-Prot(swissprot)** et il est peu probable que ceux-ci soient réellement codants pour des [parties de] protéines.

Nous pouvons donc sereinement attaquer l'analyse des ORF pour lesquels nous avons des résultats.

1.3 Le script du bourrin

Vu que nous allons essentiellement exécuter les outils que nous avions testés en ligne de commande, le **Bash** est tout indiqué ici. Nous allons partir du principe que le script, que j'ai baptisé **brutal_analysis.sh**, n'a besoin que de l'identifiant de l'ORF pour dérouler la suite des outils. Cependant, nous n'allons pas prendre trop de risques non plus et commencer par dire au script de cesser immédiatement dès qu'une erreur se produit (ligne 3). Nous allons récupérer le répertoire de travail en nous basant sur le chemin menant au script (ligne 5) et vérifier que la variable n'est pas vide par acquit de conscience (ligne 6). Nous vérifions également que nous pouvons nous rendre dans le répertoire hébergeant le script et que nous pouvons lire son contenu (lignes 8 à 15).

```
01: #!/bin/bash
02:
03: set -e
04:
05: WORKING_DIR="$(dirname "$0")"
06: test -n "${WORKING_DIR}"
07:
08: PROGNAME=$(basename "$0")
09: test -n "${PROGNAME}"
10:
11: cd "${WORKING_DIR}"
12: if test ! -f ${PROGNAME}; then
13:   echo "BUGGGG"
14:   exit 1
15: fi
```

Nous récupérons l'identifiant de l'ORF à étudier qui doit être le premier argument de notre script et nous nous assurons que cet identifiant a bel et bien été fourni en ligne de commande, sans quoi nous affichons un message rudimentaire sur l'utilisation de ce programme.

```
17: ORF_ID="$1"
18: if test -z "${ORF_ID}"; then
19:   cat <<EOF
20: usage: ${PROGNAME} <ID>
21: avec <ID> l'identifiant de l'ORF à analyser
22: EOF
23:   exit 1
24: fi
```

Nous remontons d'un niveau dans l'arborescence des répertoires (donc *a priori* dans le répertoire **analysis**), puis définissons quelques variables pour rendre le script potentiellement évolutif et éviter les coquilles typographiques qui font pleurer :

```
26: cd ../
27: orgn="nc_045512.2"
28: res_dir="results"
29: tmp_dir="${res_dir}/tmp_${ORF_ID}"
30: fich_base="${res_dir}/ORF/${orgn}"
31: fich="${fich_base}_${ORF_ID}"
32: fich_tpl="${fich}_related"
```

Ainsi, **fich_tpl** devrait valoir quelque chose comme **results/ORF/nc_045512.2_${ORF_ID}_related** et **tmp_dir** (répertoire temporaire) devrait valoir quelque chose comme **results/tmp_${ORF_ID}**.

Nous créons donc le répertoire temporaire (lignes 35 et 46).

```
34: echo "Préparation du répertoire de travail"
35: rm -rf "${tmp_dir}"
36: mkdir "${tmp_dir}"
```

Lorsque l'on télécharge les séquences sélectionnées dans BLAST, mon navigateur me propose de l'enregistrer sous le nom **sequences.gb**. Cela me convient très bien, j'enregistrerai les fichiers dans le répertoire **/tmp**. Il me suffit donc de déplacer ce fichier dans le répertoire de travail sous son nom définitif (ligne 39), puis d'extraire les séquences dans le répertoire temporaire (ligne 42), ainsi que les annotations qui vont avec (ligne 44). Cela nous permet de créer le fichier contenant toutes les séquences, y compris la séquence de l'ORF en cours d'analyse (ligne 47).

```
38: echo "Récupération des données au format genbank"
39: mv /tmp/sequence.gb ${fich_tpl}.gb
40:
41: echo "Extraction des séquences"
42: seqretsplit ${fich_tpl}.gb -feature -osdirectory2 ${tmp_dir} -osformat2 fasta -auto
43: echo "Extraction des annotations"
44: seqretsplit ${fich_tpl}.gb -feature -osdirectory2 ${tmp_dir} -osformat2 gff -auto -stdout
45:
46: echo "Construction des séquences à aligner"
47: cat ${fich}.fasta ${tmp_dir}/*fasta > ${fich_tpl}.fasta
```

Nous procédons à l'alignement des séquences (ligne 50), à la génération des représentations des arbres de guidage (lignes 53 à 57) et lançons son affichage avec **EOG** ou à défaut le programme associé aux fichiers **PNG** (ligne 58).

```
49: echo "Alignement des séquences"
50: emma ${fich_tpl}.fasta -outseq ${fich_tpl}.emma -dendoutfile ${fich_tpl}.dnd -osformat2
msf -auto
51:
52: echo "Visualisation du dendrogramme"
53: newicktotxt ${fich_tpl}.dnd
54: mv ${fich_base%.*}.txt ${fich_tpl}.txt
55: newicktops ${fich_tpl}.dnd
56: convert ${fich_base%.*}.ps ${fich_tpl}.png
57: rm -f ${fich_base%.*}.ps
58: eog ${fich_tpl}.png || xdg-open ${fich_tpl}.png
```

Nous créons le graphique de conservation de l'alignement (ligne 61) que nous visualisons directement (ligne 64) afin de pouvoir vérifier aussi que l'alignement est représentatif (que les séquences sont suffisamment semblables pour pouvoir transférer les informations).

```
60: echo "Création du graphique de conservation"
61: plotcon ${fich_tpl}.emma -graph png -goutfile ${fich_tpl}_plotcon -auto
62:
63: echo "Visualisation du graphique de conservation"
64: eog ${fich_tpl}_plotcon.1.png || xdg-open ${fich_tpl}_plotcon.1.png
```

Comme rien ne vaut malgré tout un regard sur l'alignement brut, nous l'ouvrons avec le *pager* **less**.

```
66: echo "Visualisation de l'alignement"
67: less ${fich_tpl}.emma
```

Enfin, selon la confiance que l'alignement nous inspire, il reste à copier-coller les annotations qui nous semblent pertinentes (car retrouvées dans toutes les séquences) en vue de transformer notre ORF en CDS et en gène. Pour ma part, j'utilise **joe** qui présente tous les inconvénients de **vim** et **emacs** réunis sans avoir un seul de leurs avantages (bref, j'adore !).

```
69: echo "Édition des annotations pour l'ORF ${ORF_ID}"
70: joe ${res_dir}/${orgn}.gff ${tmp_dir}/*.gff
71:
72: echo "That's all Folks\!\!\!"
```

Nous pouvons vérifier rapidement que la requête correspondant à l'ORF 81 (que nous avions traitée) donne les mêmes résultats (en sélectionnant les « bonnes » séquences). Parfait... analysons en masse.

1.4 Analyse du berserker

La mécanique est simple, pour chacune des requêtes, nous récupérons les séquences qui couvrent au moins 80 % de la requête dans le fichier `/tmp/sequence.gb` et nous appliquons le script précédent en lui passant le numéro de l'ORF en argument.

Par exemple, pour la seconde requête ayant donné un résultat (15e séquence requête, qui correspond à l'ORF n°137 ; cf. figure 2), nous retrouvons en premier les séquences provenant des mêmes organismes que précédemment (des coronavirus infectieux de la chauve-souris) et codant toutes pour la protéine *Replicase polyprotein 1ab*.

L'alignement montre clairement que la requête s'aligne seulement à la fin des autres séquences. Or pour l'ORF précédent (n°81), nous avions vu que sur ces mêmes séquences, l'alignement était partiel et portait seulement sur le début des séquences. Il ne semble pas exister de protéine *Replicase polyprotein 1b*, mais **[7]** explique que lorsque la séquence codant la *Replicase polyprotein 1a* est suivie d'un ORF valide, éventuellement avec un décalage de cadre de lecture, ce qui est le cas ici, le produit de cet ORF peut se combiner visiblement avec la chaîne d'acides aminés de la protéine *Replicase polyprotein 1a* pour former la protéine *Replicase polyprotein 1ab*. Ceci corrobore le résultat obtenu pour l'ORF n°81 que nous avions détecté comme étant codant pour la *Replicase protein 1a*.

En appliquant le script, nous réussissons à annoter tous les ORF restants sans aucune difficulté, excepté pour l'ORF n°168, où il n'y a qu'une seule séquence résultat. Ceci empêche le bon déroulement de l'étape d'alignement multiple (qui requiert de devoir aligner au moins 3 séquences), mais une astuce assez simple consiste à dupliquer le contenu du fichier `/tmp/sequence.gb` avant de lancer le script. Ceci étant, l'alignement est vraiment très partiel et il est difficile de déduire une information probante.

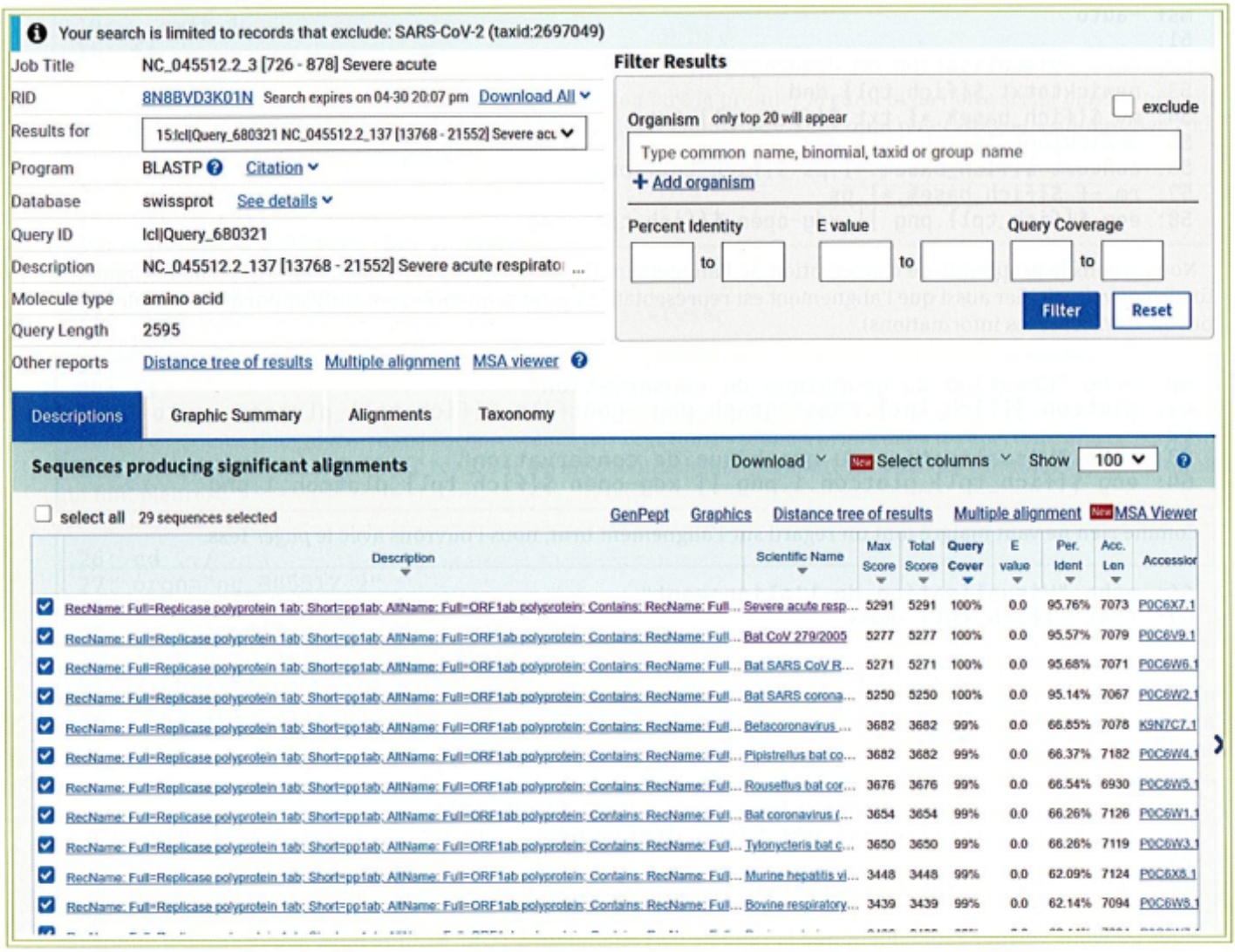

Fig. 2 : Affichage des résultats de BLAST pour la 15e requête (correspondant à l'ORF n°137) sur le site de NCBI.

Il semble évident au vu des séquences retrouvées pour chacun des ORF que la séquence étudiée est un coronavirus proche de ceux qui infectent les chauves-souris. Il reste quelques ORF pour lesquels nous n'avons pas eu d'informations.

Pour les très petits ORF (ceux qui codent pour moins de 30 acides aminés), nous pouvons nous contenter de les supprimer sans scrupule, car il y a peu de chances qu'ils codent réellement pour des protéines. Cela concerne les ORF numéros : 82 (13aa), 83 (24aa), 174 (12aa), 188 (10aa), 190 (11aa) et 191 (12aa). Nous pouvons laisser les annotations des autres ORF dans le doute.

Il est également possible (voire même judicieux) d'ajuster le début de la région 3'-UTR à partir de la dernière région codante que nous avons détectée.

Ceci nous donne le fichier final suivant (**results/nc_045512.2.gff**) :

```
##gff-version 3
##sequence-region nc_045512.2 1 29903
nc_045512.2perso region 129903 .+.ID=nc_045512.2;dbxref=taxon:2697049;name=Severe acute
respiratory syndrome coronavirus 2;Alias=SARS-CoV-2;note=Wuhan-Hu-1 (Wuhan seafood market
pneumonia virus);mol_type=genomic RNA;country=China;date=2019-12
nc_045512.2perso five_prime_UTR 1265 .+.ID=region 5' UTR
nc_045512.2perso CDS 266 13483 .+1ID=nc_045512.2_81;name=ORF_81;gene=ORF1a;parent=nc_045512.2_
pp1a;locus_tag=1a;product=Replicase polyprotein 1a;note=pp1a%3B ORF1a polyprotein
nc_045512.2perso gene 266 13483 .+.ID=nc_045512.2_pp1a;name=pp1a;gene=pp1a;locus_
tag=1a;product=Replicase polyprotein 1a;note=pp1a%3B ORF1a polyprotein
nc_045512.2perso CDS 13768 21555 .+0ID=nc_045512.2_137;name=ORF_137;gene=ORF1b;parent=
nc_045512.2_pp1ab;locus_tag=1b;product=Replicase polyprotein 1ab;note=pp1ab%3B ORF1ab
polyprotein%3BThe protein seems to be combinated with ORF1a to produce the Replicase
polyprotein 1ab with a frameshift of -1
nc_045512.2perso gene 266 21555 .+.ID=nc_045512.2_pp1ab;name=pp1ab;gene=pp1ab;locus_
tag=1ab;product=Replicase polyprotein 1ab;note=pp1ab%3B ORF1ab polyprotein%3BThe protein seems
to be combinated with ORF1a to produce the Replicase polyprotein 1ab with a frameshift of -1
nc_045512.2perso CDS 21536 25384 .+1ID=nc_045512.2_159;name=ORF_159;gene=S;parent=nc_045512.2_S;;
locus_tag=2;product=Spike glycoprotein;note=S glycoprotein%3B E2%3B Peplomer protein
nc_045512.2perso gene 21536 25384 .+.ID=nc_045512.2_S;name=S;gene=S;locus_tag=2;product=Spike
glycoprotein;note=S glycoprotein%3B E2%3B Peplomer protein
nc_045512.2perso CDS 25393 26220 .+0ID=nc_045512.2_167;name=ORF_167;gene=APA3;parent=nc_045512.2_
APA3;locus_tag=3;product=Protein 3;note=Coronavirus accessory protein 3a%3B pfam11289
nc_045512.2perso gene 25393 26220 .+.ID=nc_045512.2_APA3;name=APA3;gene=APA3;locus_
tag=3;product=Protein 3;note=Coronavirus accessory protein 3a%3B pfam11289
nc_045512.2perso CDS 26183 26281 .+1ID=nc_045512.2_168;name=ORF_168;gene=ORF3b;parent=
nc_045512.2_ORF3b;locus_tag=3b;product=Accessory protein 3b;note=ns3b%3B Accessory protein
3b%3B Non-structural protein 3b%3B Protein X2%3B seems to have only the bipartite nuclear
localization signal
nc_045512.2perso gene 26183 26281 .+.ID=nc_045512.2_ORF3b;name=ORF3b;gene=ORF3b;locus_
tag=3b;product=Accessory protein 3b;note=ns3b%3B Accessory protein 3b%3B Non-structural protein
3b%3B Protein X2%3B seems to have only the bipartite nuclear localization signal
nc_045512.2perso CDS 26245 26472 .+0ID=nc_045512.2_169;name=ORF_169;gene=E;parent=nc_045512.2_E;
locus_tag=4;product=Envelope small membrane protein;note=E protein%3B sM protein
nc_045512.2perso gene 26245 26472 .+.ID=nc_045512.2_E;name=E;gene=E;locus_tag=4;product=Envelope
small membrane protein;note=E protein%3B sM protein
nc_045512.2perso CDS 26523 27191 .+2ID=nc_045512.2_172;name=ORF_172;gene=M;parent=nc_045512.2_M;
locus_tag=5;product=Membrane protein;note=M protein%3B E1 glycoprotein%3B Matrix
glycoprotein%3B Membrane glycoprotein
nc_045512.2perso gene 26523 27191 .+.ID=nc_045512.2_M;name=VME1;gene=M;locus_
tag=5;product=Membrane protein;note=M protein%3B E1 glycoprotein%3B Matrix glycoprotein%3B
Membrane glycoprotein
nc_045512.2perso CDS 27202 27387 .+0ID=nc_045512.2_173;name=ORF_173;gene=NS6;parent=nc_045512.2_
NS6;locus_tag=6;product=Non-structural protein 6;note=ns6%3B Accessory protein 6%3B Open
reading frame 6 from SARS coronavirus%3B pfam12133
```

```
nc_045512.2perso gene 27202 27387 .+.ID=nc_045512.2_NS6;name=NS6;gene=NS6;locus_
tag=6;product=Non-structural protein 6;note=ns6%3B Accessory protein 6%3B Open reading frame 6
from SARS coronavirus%3B pfam12133
nc_045512.2perso CDS 27394 27759 .+0ID=nc_045512.2_175;name=ORF_175;gene=NS7a;parent=nc_045512.2_
NS7a;locus_tag=7a;product=Protein 7a;note=Accessory protein 7a
nc_045512.2perso gene 27394 27759 .+.ID=nc_045512.2_NS7a;name=NS7a;gene=NS7a;locus_
tag=7a;product=Protein 7a;note=Accessory protein 7a
nc_045512.2perso CDS 27756 27887 .+2ID=nc_045512.2_176;name=ORF_176;gene=NS7b;parent=nc_045512.2_
NS7b;locus_tag=7b;product=Non-structural protein 7b;note=ns7b%3B Accessory protein 7b%3B
pfam11395
nc_045512.2perso gene 27756 27887 .+.ID=nc_045512.2_NS7b;name=NS7b;gene=NS7b;locus_
tag=7b;product=Non-structural protein 7b;note=ns7b%3B Accessory protein 7b%3B pfam11395
nc_045512.2perso CDS 27894 28259 .+2ID=nc_045512.2_179;name=ORF_179;gene=NS8;parent=nc_045512.2_
NS8;locus_tag=8;product=Non-structural protein 8;note=ns8%3B Accessory protein 8%3B pfam12093
nc_045512.2perso gene 27894 28259 .+.ID=nc_045512.2_NS8;name=NS8;gene=NS8;locus_
tag=8;product=Non-structural protein 8
nc_045512.2perso CDS 28274 29533 .+1ID=nc_045512.2_187;name=ORF_187;gene=N;parent=nc_04551
2.2_N;locus_tag=9a;product=Nucleoprotein;note=Nucleocapsid protein%3B NC%3B Protein N%3B
Nucleoprotein%3B Coronavirus nucleocapsid protein%3B pfam00937
nc_045512.2perso gene 28274 29533 .+.ID=nc_045512.2_N;name=N;gene=N;locus_tag=9a;product=Nucleo
protein;note=Nucleocapsid protein%3B NC%3B Protein N%3B Nucleoprotein%3B Coronavirus
nucleocapsid protein%3B pfam00937
nc_045512.2perso CDS 28284 28577 .+2ID=nc_045512.2_180;name=ORF_180;gene=APA9b;parent=
nc_045512.2_APA9b;locus_tag=9b;product=Protein 9b;note=Accessory protein 9b%3B ORF-9b%3B SARS
lipid binding protein%3B pfam09399
nc_045512.2perso gene 28284 28577 .+.ID=nc_045512.2_APA9b;name=APA9b;gene=APA9b;locus_
tag=9b;product=Protein 9b;note=Accessory protein 9b%3B ORF-9b%3B SARS lipid binding protein%3B
pfam09399
nc_045512.2perso CDS 28734 28955 .+2ID=nc_045512.2_183;name=ORF_183;gene=ORF9c;parent=
nc_045512.2_ORF9c;locus_tag=9c;product=Uncharacterized protein 14;note=Uncharacterized protein
14%3B accessory protein ORF9c (also referred to as ORF14) from Severe acute respiratory
syndrome-associated coronavirus and related coronaviruses
nc_045512.2perso gene 28734 28955 .+.ID=nc_045512.2_ORF9c;name=ORF9c;gene=ORF9c;locus_
tag=9c;product=Uncharacterized protein 14;note=Uncharacterized protein 14%3B accessory protein
ORF9c (also referred to as ORF14) from Severe acute respiratory syndrome-associated coronavirus
and related coronaviruses
nc_045512.2filtre_ORF ORF 726 881 .+2ID=nc_045512.2_3;name=ORF_3
nc_045512.2filtre_ORF ORF 2958 3206 .+2ID=nc_045512.2_16;name=ORF_16
nc_045512.2filtre_ORF ORF 3948 4103 .+2ID=nc_045512.2_24;name=ORF_24
nc_045512.2filtre_ORF ORF 5919 6089 .+2ID=nc_045512.2_36;name=ORF_36
nc_045512.2filtre_ORF ORF 6156 6350 .+2ID=nc_045512.2_37;name=ORF_37
nc_045512.2filtre_ORF ORF 7548 7709 .+2ID=nc_045512.2_43;name=ORF_43
nc_045512.2filtre_ORF ORF 10215 10400 .+2ID=nc_045512.2_58;name=ORF_58
nc_045512.2filtre_ORF ORF 13311 13466 .+2ID=nc_045512.2_80;name=ORF_80
nc_045512.2filtre_ORF ORF 15461 15667 .+1ID=nc_045512.2_94;name=ORF_94
nc_045512.2filtre_ORF ORF 16973 17122 .+1ID=nc_045512.2_106;name=ORF_106
nc_045512.2filtre_ORF ORF 19148 19303 .+1ID=nc_045512.2_122;name=ORF_122
nc_045512.2filtre_ORF ORF 21936 22199 .+2ID=nc_045512.2_139;name=ORF_139
nc_045512.2filtre_ORF ORF 25332 25448 .+2ID=nc_045512.2_160;name=ORF_160
nc_045512.2filtre_ORF ORF 25524 25697 .+2ID=nc_045512.2_162;name=ORF_162
nc_045512.2filtre_ORF ORF 26693 26872 .+1ID=nc_045512.2_171;name=ORF_171
nc_045512.2filtre_ORF ORF 29558 29674 .+1ID=nc_045512.2_189;name=ORF_189
nc_045512.2perso three_prime_UTR 28956 29903 .+.ID=region 3' UTR
nc_045512.2perso polyA_sequence 29871 29903 .+.ID=poly-A tail
```

Il est bien évidemment possible de pousser l'analyse davantage, mais nous nous arrêterons là pour cette séquence.

2. QU'A-T-ON DÉCOUVERT ?

Produire un beau fichier GFF, c'est sympa, mais ça laisse tout de même un peu perplexe. Nous pouvons dans un premier temps essayer de le visualiser.

2.1 Le so-ASCII (*so cute*)

La suite EMBOSS propose le programme **showfeat** pour effectuer un rendu ASCII des annotations sur une séquence, mais ce programme n'accepte pas le GFF. Qu'à cela ne tienne, créons une séquence annotée au format **EMBL** :

```
$ seqret ncbi_dataset/individuals/nc_045512.2.fasta -feature -fformat gff -fopenfile
results/nc_045512.2.gff -osformat embl -auto -outseq results/nc_045512.2.embl
```

Nous découvrons un autre format, mais le contenu est relativement bien préservé, si ce n'est que les lignes commençant par **FT** sont mal formées, car elles contiennent l'identifiant de séquence avant les positions et ça, **showfeat** n'aime pas.

```
ID   NC_045512.2; SV 1; linear; unassigned DNA; STD; UNC; 29903 BP.
XX
DE   Severe acute respiratory syndrome coronavirus 2 isolate Wuhan-Hu-1,
DE   complete genome
XX
FH   Key             Location/Qualifiers
FH
FT   -               nc_045512.2:1..29903
FT                   /note="*Alias SARS-CoV-2"
FT                   /note="*dbxref: taxon:2697049"
FT                   /note="*name: Severe acute respiratory syndrome coronavirus
FT                   2"
FT                   /note="Wuhan-Hu-1 (Wuhan seafood market pneumonia virus)"
FT                   /mol_type="genomic RNA"
FT                   /country="China"
FT                   /note="*date: 2019-12"
FT   5'UTR           nc_045512.2:1..265
FT   CDS             nc_045512.2:266..13483
FT                   /note="*name: ORF_81"
FT                   /gene="ORF1a"
FT                   /note="*parent: nc_045512.2_pp1a"
FT                   /locus_tag="1a"
FT                   /product="Replicase polyprotein 1a"
FT                   /note="pp1a; ORF1a polyprotein"
FT   gene            nc_045512.2:266..13483
FT                   /note="*name: pp1a"
FT                   /gene="pp1a"
FT                   /locus_tag="1a"
FT                   /product="Replicase polyprotein 1a"
FT                   /note="pp1a; ORF1a polyprotein"
[fichier tronqué]
FT   misc_feature    nc_045512.2:26693..26872
FT                   /note="*Type SO:0000236 ORF"
FT                   /note="*name: ORF_171"
FT   misc_feature    nc_045512.2:29558..29674
FT                   /note="*Type SO:0000236 ORF"
FT                   /note="*name: ORF_189"
FT   3'UTR           nc_045512.2:28956..29903
FT   misc_feature    nc_045512.2:29871..29903
FT                   /note="*Type SO:0000610 polyA_sequence"
```

```
XX
SQ   Sequence 29903 BP; 8954 A; 5492 C; 5863 G; 9594 T; 0 other;
     ATTAAAGGTT TATACCTTCC CAGGTAACAA ACCAACCAAC TTTCGATCTC TTGTAGATCT        60
     GTTCTCTAAA CGAACTTTAA AATCTGTGTG GCTGTCACTC GGCTGCATGC TTAGTGCACT       120

...

     TTTAGTAGTG CTATCCCCAT GTGATTTTAA TAGCTTCTTA GGAGAATGAC AAAAAAAAAA     29880
     AAAAAAAAAA AAAAAAAAAA AAA                                            29903
//
```

Ceci n'est pas trop grave, un coup de **sed** et c'est fixé :

```
$ sed -i 's,^\(FT.*  \)nc_045512.2:,\1,' results/nc_045512.2.embl
$ head -n 20 results/nc_045512.2.embl
ID   NC_045512.2; SV 1; linear; unassigned DNA; STD; UNC; 29903 BP.
XX
DE   Severe acute respiratory syndrome coronavirus 2 isolate Wuhan-Hu-1,
DE   complete genome
XX
FH   Key             Location/Qualifiers
FH
FT   -               1..29903
FT                   /note="*Alias SARS-CoV-2"
FT                   /note="*dbxref: taxon:2697049"
FT                   /note="*name: Severe acute respiratory syndrome coronavirus
FT                   2"
FT                   /note="Wuhan-Hu-1 (Wuhan seafood market pneumonia virus)"
FT                   /mol_type="genomic RNA"
FT                   /country="China"
FT                   /note="*date: 2019-12"
FT   5'UTR           1..265
FT   CDS             266..13483
FT                   /note="*name: ORF_81"
FT                   /gene="ORF1a"
```

Nous pouvons à présent afficher la séquence annotée (c'est trop stylé) :

```
$ showfeat -sort type -position results/nc_045512.2.embl -auto -outfile results/
nc_045512.2.showfeat
$ cat results/nc_045512.2.showfeat
NC_045512.2
Severe acute respiratory syndrome coronavirus 2 isolate Wuhan-Hu-1, complete
genome
|==========================================================| 29903
|----------------------------------------------------------> 1-29903 -
                                                          |-> 28956-29903 3'UTR
>                                                           1-265 5'UTR
|------------------------>                                  266-13483 CDS
                          |--------------->                 13768-21555 CDS
                                          |------>          21536-25384 CDS
                                                 |->        25393-26220 CDS
                                                   >       26183-26281 CDS
                                                  |>       26245-26472 CDS
                                                   |>      26523-27191 CDS
                                                    >      27202-27387 CDS
```

```
                                                    |>      27394-27759 CDS
                                                     >      27756-27887 CDS
                                                     |>     27894-28259 CDS
                                                      |-->  28274-29533 CDS
                                                      |>    28284-28577 CDS
                                                       |>   28734-28955 CDS
|------------------------>                                 266-13483 gene
|---------------------------------------->                 266-21555 gene
                                          |------>         21536-25384 gene
                                                 |->       25393-26220 gene
                                                   >       26183-26281 gene
                                                   |>      26245-26472 gene
                                                    |>     26523-27191 gene
                                                    >      27202-27387 gene
                                                    |>     27394-27759 gene
                                                     >      27756-27887 gene
                                                     |>     27894-28259 gene
                                                      |-->  28274-29533 gene
                                                      |>    28284-28577 gene
                                                       |>   28734-28955 gene
>                                                          726-881 ORF
    |>                                                    2958-3206 ORF
      |>                                                  3948-4103 ORF
        |>                                                5919-6089 ORF
         >                                                6156-6350 ORF
           >                                              7548-7709 ORF
             >                                            10215-10400 ORF
                |>                                        13311-13466 ORF
                  >                                      15461-15667 ORF
                    >                                    16973-17122 ORF
                      >                                  19148-19303 ORF
                        >                                21936-22199 ORF
                                                 |>      25332-25448 ORF
                                                  >      25524-25697 ORF
                                                    >    26693-26872 ORF
                                                      >  29558-29674 ORF
                                                     |>  29871-29903 polyA_sequence
```

Il y a d'autres visualisations possibles par exemple avec **showseq** qui propose différents formats (ici, j'ai sélectionné l'affichage avec traduction en séquence d'acides aminés) :

```
$ showseq results/nc_045512.2.embl -format 5 -auto -outfile results/
nc_045512.2.showseq
$ cat results/nc_045512.2.showseq
NC_045512.2
Severe acute respiratory syndrome coronavirus 2 isolate Wuhan-Hu-1,
complete genome

               10        20        30        40        50        60
        ----:----|----:----|----:----|----:----|----:----|----:----|
        ATTAAAGGTTTATACCTTCCCAGGTAACAAACCAACCAACTTTCGATCTCTTGTAGATCT

        I  K  G  L  Y  L  P  R  *  Q  T  N  Q  L  S  I  S  C  R  S
         L  K  V  Y  T  F  P  G  N  K  P  T  N  F  R  S  L  V  D  L
          *  R  F  I  P  S  Q  V  T  N  Q  P  T  F  D  L  L  *  I  C
        |============================================================
        - note=" dbxref:: taxon:2697049" note=" name:: Severe acute respirato
        |============================================================
        5'UTR
```

```
                70        80        90        100       110       120
          ----:----|----:----|----:----|----:----|----:----|----:----|
          GTTCTCTAAACGAACTTTAAAATCTGTGTGGCTGTCACTCGGCTGCATGCTTAGTGCACT

          V  L  *  T  N  F  K  I  C  V  A  V  T  R  L  H  A  *  C  T
           F  S  K  R  T  L  K  S  V  W  L  S  L  G  C  M  L  S  A  L
            S  L  N  E  L  *  N  L  C  G  C  H  S  A  A  C  L  V  H  S
          ============================================================
          - note=" dbxref:: taxon:2697049" note=" name:: Severe acute respirato
          ============================================================
          5'UTR
[sortie tronquée]
                29830     29840     29850     29860     29870     29880
          ----:----|----:----|----:----|----:----|----:----|----:----|
          TTTAGTAGTGCTATCCCCATGTGATTTTAATAGCTTCTTAGGAGAATGACAAAAAAAAAA

          F  S  S  A  I  P  M  *  F  *  *  L  L  R  R  M  T  K  K  K
           L  V  V  L  S  P  C  D  F  N  S  F  L  G  E  *  Q  K  K  K
            *  *  C  Y  P  H  V  I  L  I  A  S  *  E  N  D  K  K  K  K
          ============================================================
          - note=" dbxref:: taxon:2697049" note=" name:: Severe acute respirato
          ============================================================
          3'UTR
                                                          |=========
                                                          polyA_sequence

                29890     29900
          ----:----|----:----|---
          AAAAAAAAAAAAAAAAAAAAAA

          K  K  K  K  K  K  K  X
           K  K  K  K  K  K  K  X
            K  K  K  K  K  K  K
          ======================|
          - note=" dbxref:: taxon:2697049" note=" name:: Severe acute respirato
          ======================|
          3'UTR
          ======================|
          polyA_sequence
```

2.2 Et à part ça ?

Il est toujours intéressant de regarder une véritable image. Ainsi la figure 3 a été produite au moyen d'**annotation sketch** de la suite **GenomeTools [8]**.

Ce qui est encore plus intéressant, c'est de comparer cette image avec celle produite avec le fichier GFF officiel des annotations de *SARS-CoV-2* (disponible sur le site du NCBI et dans le dépôt Git, cf. figure 4).

Nous nous apercevons que notre analyse assez simpliste est très proche de la réalité (au sens biologique du terme). Nous retrouvons en premier lieu les deux *Replicase polyprotein 1a* (ORF 81) et *Replicase polyprotein 1ab* (ORF 81 & 137) permettant au *SARS-CoV-2* de se reproduire ; la *Spike glycoprotein* (ORF 159) permettant au virus de s'accrocher aux membranes des cellules hôtes et de les pénétrer ; l'*Enveloppe small membrane protein* (ORF 169) et la *Membrane protein* (ORF172) permettant au virus de produire la membrane de sa capside (sa coque), la *Nucleoprotein* (ORF 187) qui participe à sa réplication et à sa transcription par l'hôte. Tous ces éléments sont essentiels au cycle de vie du virus (oups ! Il paraît que les virus ne sont pas vivants) et font bien évidemment l'objet de nombreuses études en vue de pouvoir lutter efficacement contre sa propagation, puisqu'ils permettent de comprendre la structure de cet organisme (cf. figure 5, page suivante).

Fig. 3 : Visualisation des annotations prédites par notre analyse pour la séquence de référence du virus SARS-CoV-2 (NC_045512.2).

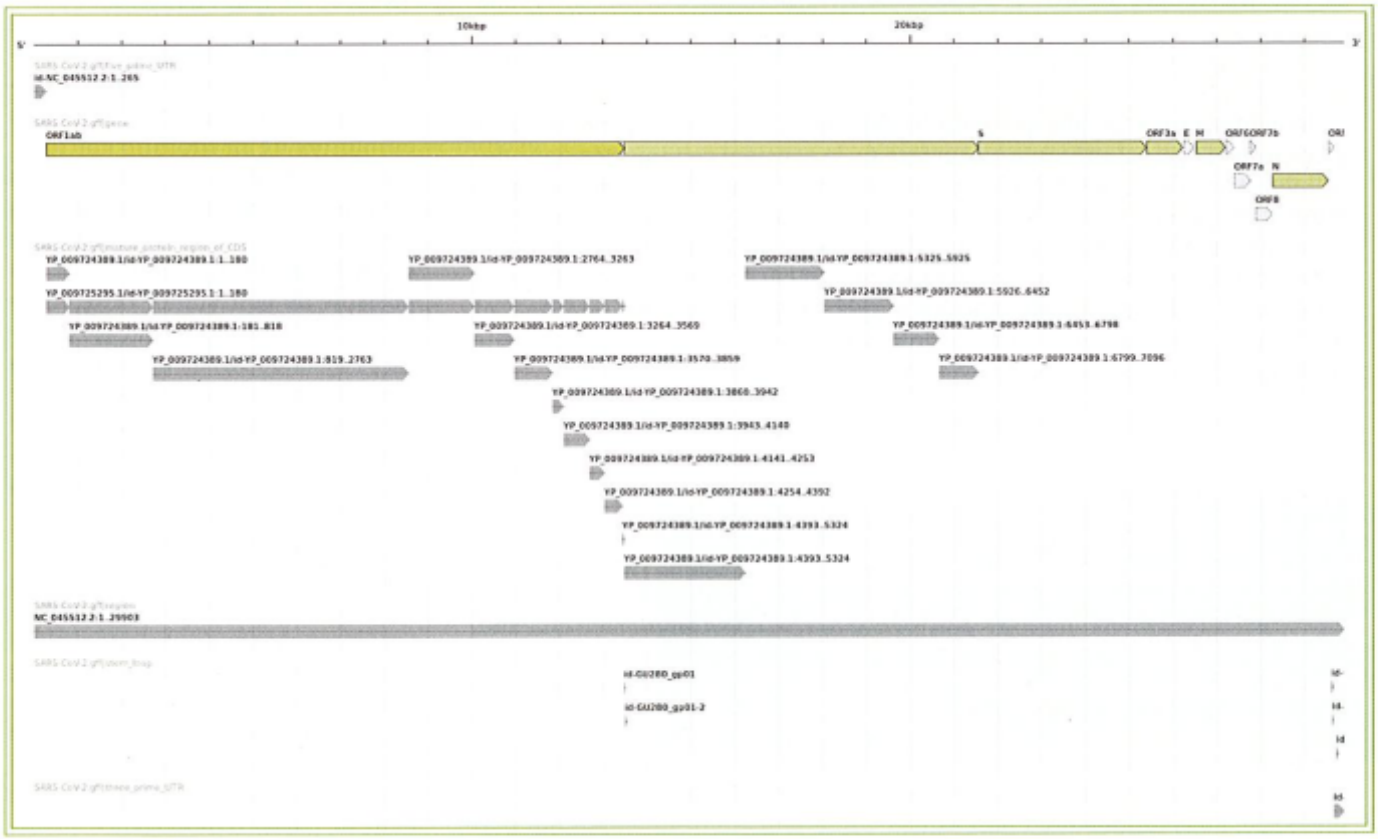

Fig. 4 : Visualisation des annotations officielles de la séquence de référence du virus SARS-CoV-2 (NC_045512.2).

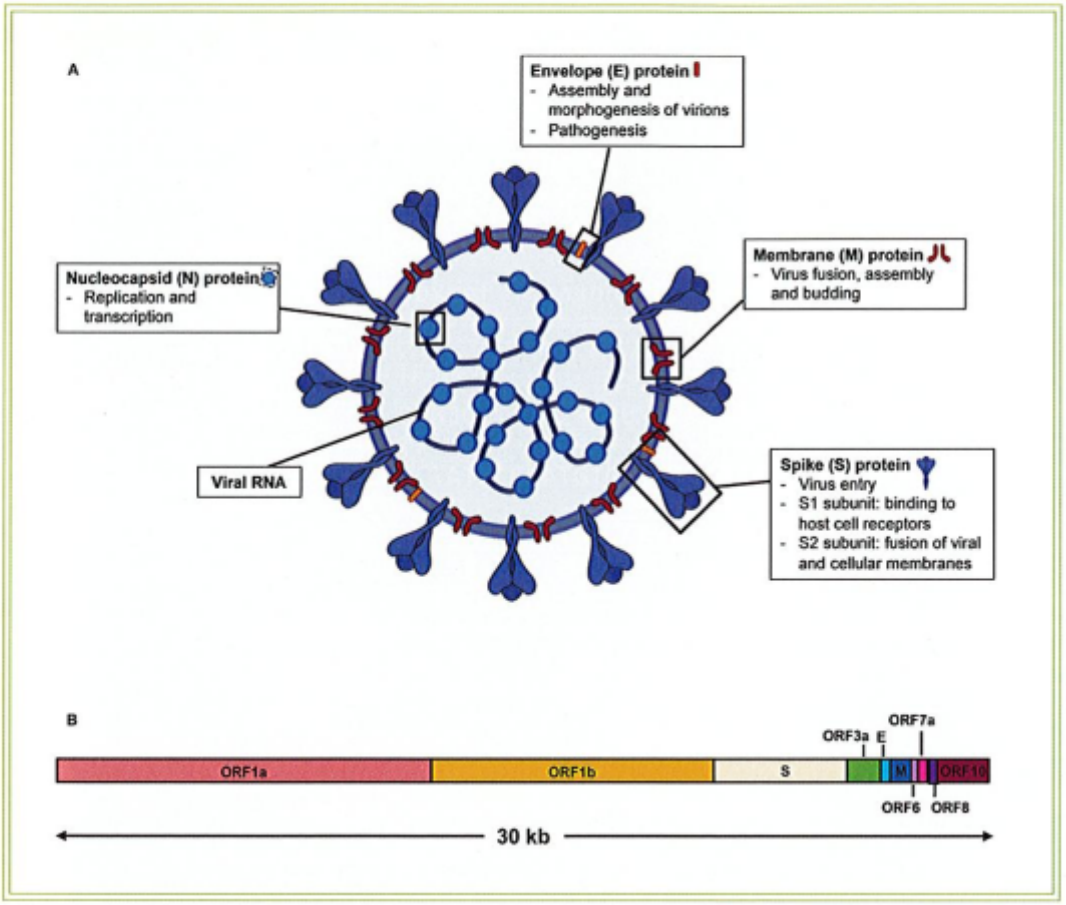

Fig. 5 : Visualisation de la structure du virus SARS-CoV-2 (NC_045512.2) extraite de [9] (sous licence CC-BY).

Imaginons par exemple que seule la séquence codante de la protéine *Spike glycoprotein* (qui est présente en surface de la capside) soit injectée dans des cellules d'un hôte potentiel. Les cellules de l'hôte vont synthétiser cette protéine et le système immunitaire pourra donc s'entraîner à la reconnaître et ainsi apprendre à lutter contre l'envahisseur fictif. Ainsi, en cas de contamination par le virus, l'hôte saura se défendre aussitôt et endiguer la prolifération du virus. C'est le principe des vaccins à ARN. Cette stratégie est robuste tant qu'il n'y a pas de mutations importantes affectant la séquence codant cette protéine (sinon il faut vacciner à nouveau avec la séquence mutée).

CONCLUSION

Vous l'aurez compris, cette analyse est très partielle et manque de rigueur. Toutefois, elle permet de montrer une fois de plus l'apport du logiciel libre (*open source*) et l'intérêt de l'ouverture des données (*open data*). Bien entendu, dans un véritable cadre d'analyse, les scripts mis en place ne sont pas suffisants (du point de vue méthodologique), pas suffisamment documentés, pas assez robustes (absence de tests), etc.

La plupart des unités de service en bioinformatique (en réalité toutes celles que je connais) travaillent exclusivement sous **GNU/Linux**. Ce n'est pas par idéologie, mais uniquement parce que c'est le seul environnement qui permet de travailler avec de bons outils en toute sérénité.

Étonnamment, à l'heure où tout le monde découvre le *Big Data* et où certains groupes tentent de s'imposer dans le secteur, en bioinformatique (où le volume de données croît exponentiellement depuis plus de 15 ans), la quasi-totalité des outils d'analyse développés le sont dans un cadre académique, avec parfois peu de moyens et de support. Ceci explique qu'une grande partie des outils sont finalement des prototypes aboutis plutôt que de réels logiciels. Et pourtant, ils permettent d'appréhender (efficacement) et de mieux comprendre le domaine du vivant.

POUR ALLER PLUS LOIN

Nous avons traité dans cette série d'articles d'analyse de séquences et de génomique comparative (deux des domaines de la bioinformatique, parmi tant d'autres). Nous nous sommes restreints à l'analyse des structures dites primaires (dans une seule dimension). Il serait tout à fait possible de poursuivre cette étude en essayant de modéliser les structures secondaires de nos séquences (les conformations locales en 3D de leurs différentes parties), puis de leurs structures tertiaires (agencement dans l'espace des structures secondaires) et quaternaires (interactions des structures entre elles). Si cette petite balade dans l'univers de la bioinformatique vous a intéressés, sachez qu'il y a de la place pour toutes et tous, que vous soyez pythoniste convaincu, développeur **C** chevronné, bayésien né, théoricien des graphes ou encore adeptes de l'apprentissage automatique (ce qui évite de parler d'intelligence, même si on la qualifie d'artificielle :D).

D'accord, il faut s'intéresser d'un peu plus près à la biologie, mais sachez que les spécialistes que j'ai rencontrés jusqu'à présent m'ont toujours transmis avec patience, bienveillance et simplicité les connaissances nécessaires à l'exercice de mon métier ; tous sans aucune exception ! Alors je vous invite très cordialement à aller plus loin. ■

RÉFÉRENCES

[1] MANCHERON A., « *Voyage initiatique vers la bioinformatique : les premiers pas* », *GNU/Linux Magazine France n°251* : **https://connect.ed-diamond.com/gnu-linux-magazine/glmf-251/voyage-initiatique-vers-la-bioinformatique-les-premiers-pas**

[2] MANCHERON A., « *Voyage initiatique vers la bioinformatique : en route pour l'aventure* », *GNU/Linux Magazine France n°253* : **https://connect.ed-diamond.com/gnu-linux-magazine/glmf-253/voyage-initiatique-vers-la-bioinformatique-en-route-pour-l-aventure**

[3] S. F. ALTSCHUL, W. GISH, W. MILLER, E. W. MYERS et D. J. LIPMAN, « *Basic Local Alignment Search Tool* », Journal of molecular biology, n°215(3), 1990, p 403 à 410.

[4] Site du *National Center for Biotechnology Information* (centre national américain) : **https://www.ncbi.nlm.nih.gov/**

[5] P. RICE, I. LONGDEN et A. BLEASBY, « *EMBOSS: The European Molecular Biology Open Software Suite* », Trends in Genetics n°16(6), 2000, p 276 à 277.

[6] Site officiel de la suite EMBOSS : **http://emboss.sourceforge.net/**

[7] J. ZIEBUHR, « *The Coronavirus Replicase - Coronavirus Replication and Reverse Genetics* », Current Topics in Microbiology and Immunology, n°287, 2005.

[8] Site officiel de la suite GenomeTools : **http://genometools.org/**

[9] C. Y.-P. LEE, R. T. P. LIN, L. RENIA et L. F. P. NG, « *Serological Approaches for COVID-19: Epidemiologic Perspective on Surveillance and Control* », Frontiers in Immunology, 2020.