Training on Galaxy: Metabarcoding

January 2024 - Webinar

FROGS Practice on command line

GABRYELLE AGOUTIN, LUCAS AUER, MARIA BERNARD, LAURENT CAUQUIL, VINCENT DARBOT, MAHENDRA MARIADASSOU, GÉRALDINE PASCAL & OLIVIER RUÉ

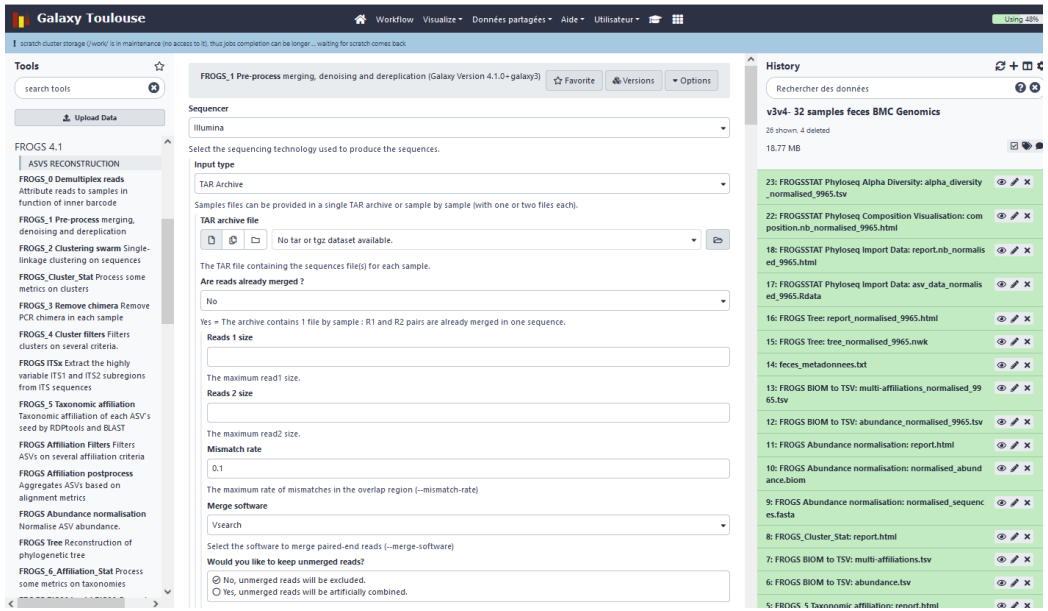
Why use FROGS on the command line?

Advantages

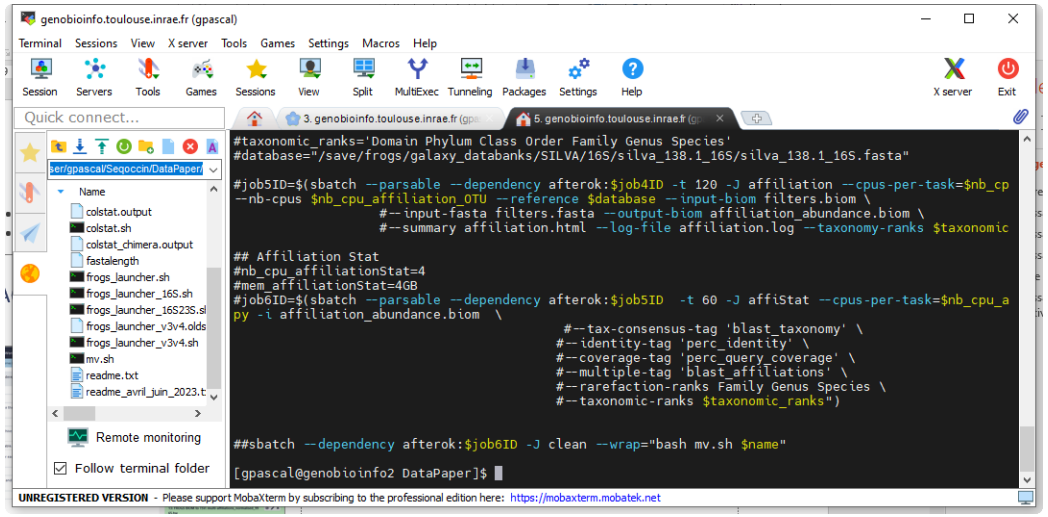
- control of the various steps
- process very large datasets
- use personal databases

FROGS: One software, 2 uses

GALAXY INTERFACE



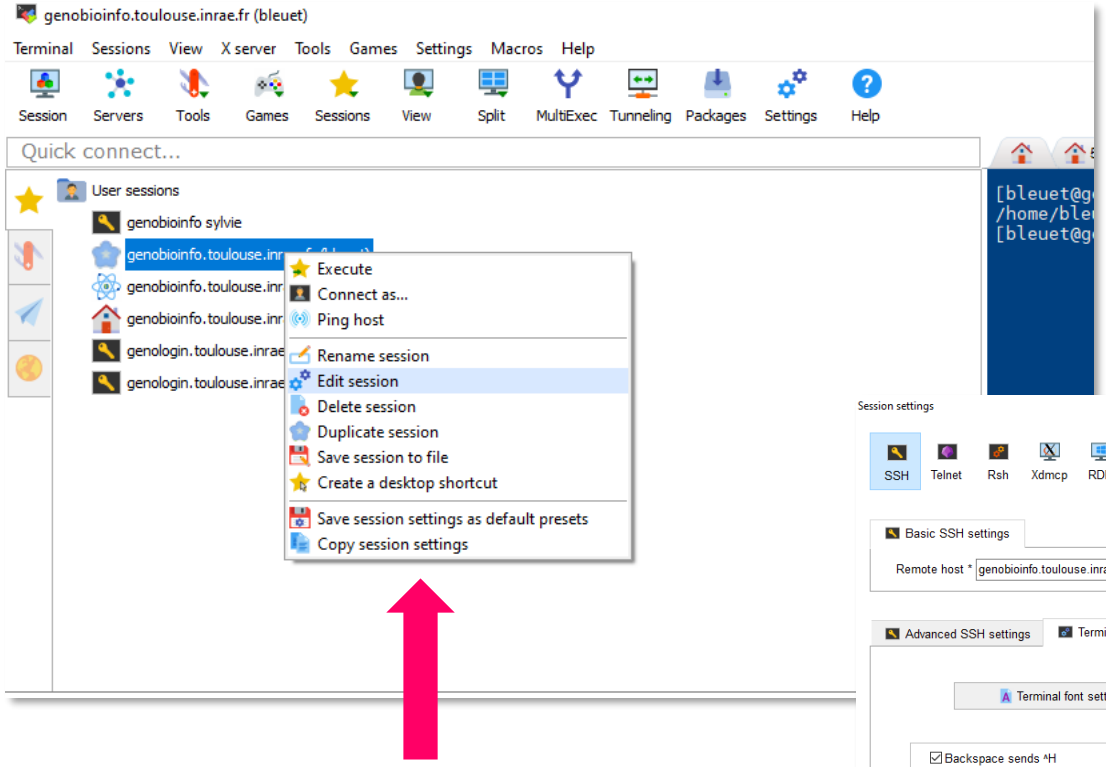
CLI: COMMAND LINE INTERFACE



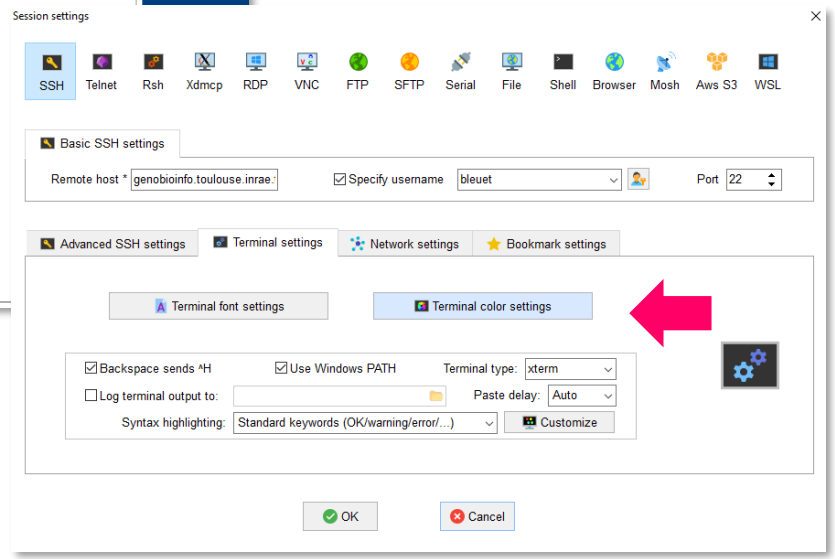
Terminal X

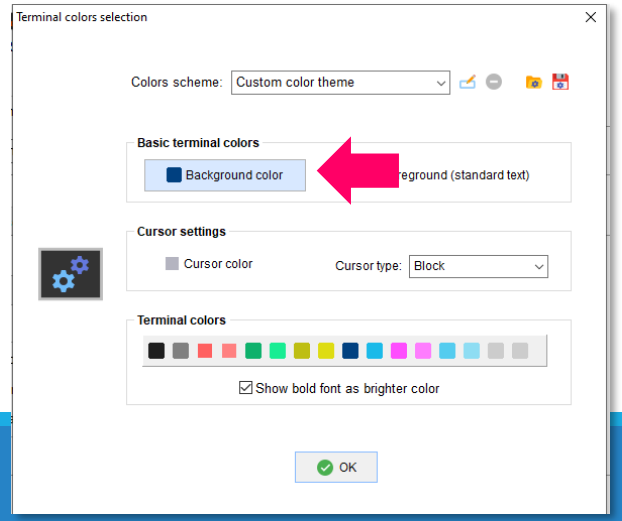
Visit of Mobaxterm

Customization

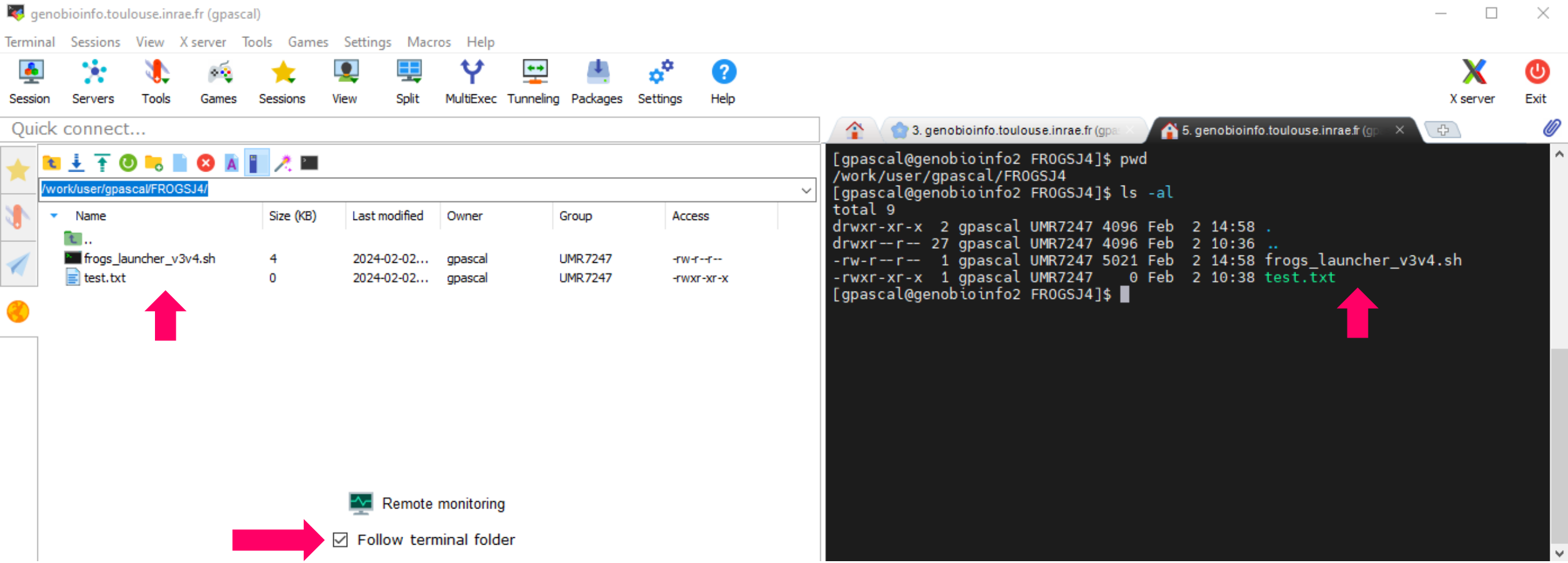


You can customize your terminal



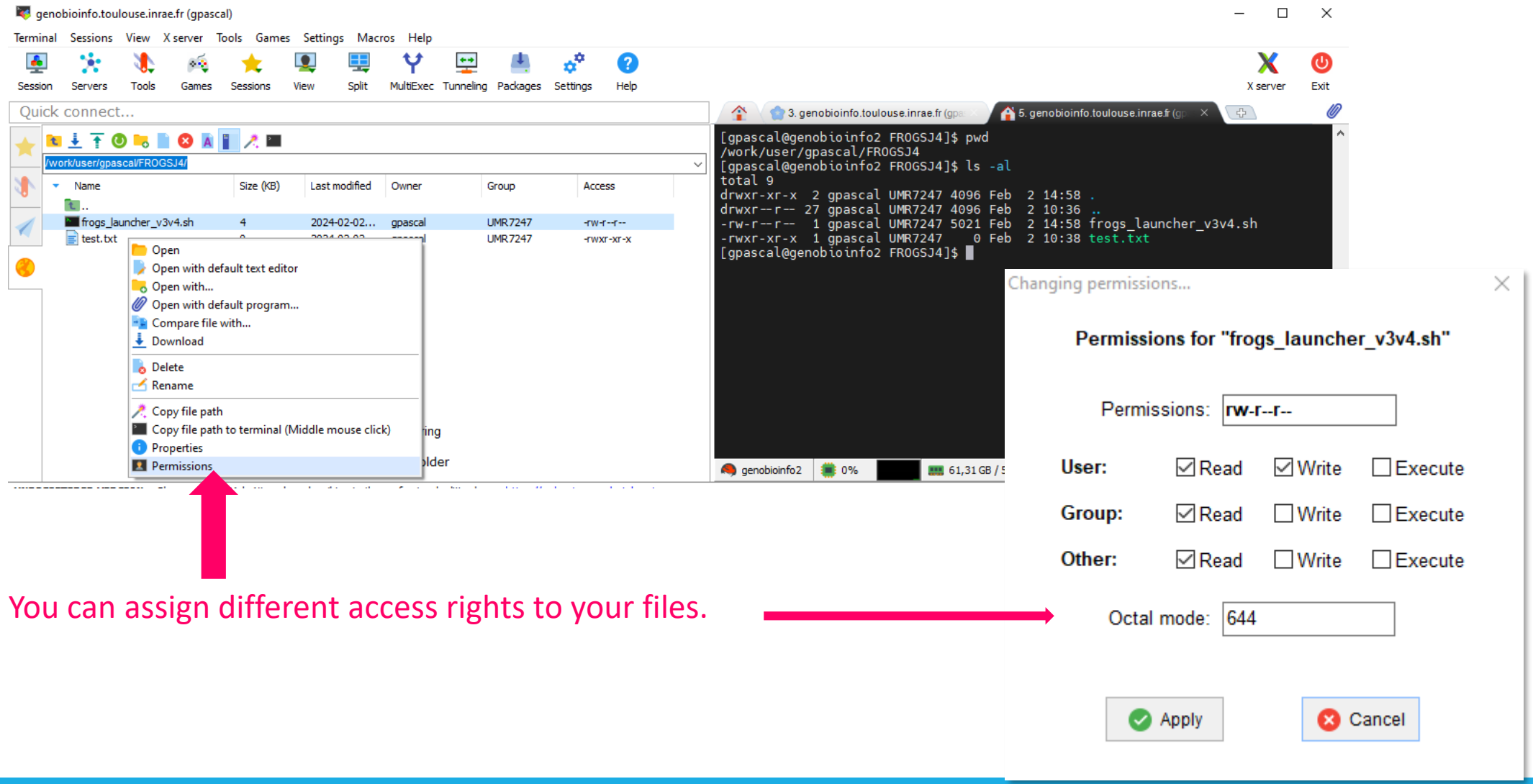


dual display



The file tree follows your path through the terminal.

Access rights



You can assign different access rights to your files.

Exercise

- As a prerequisite, we have asked you to type the command process very large datasets:

```
> chmod 751 /work/user/yourID
```

- What does it mean?

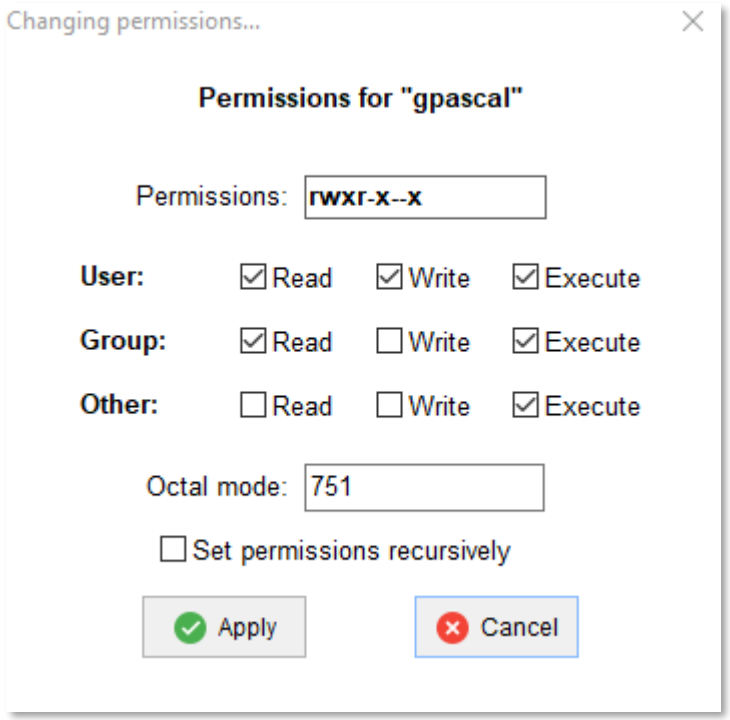
chmod:

- La commande chmod est utilisée pour modifier les permissions d'accès à un fichier ou un répertoire.
- Les permissions déterminent qui peut lire, écrire ou exécuter un fichier.
- La commande "chmod" utilise une notation numérique ou symbolique pour définir ces permissions.
- Par exemple, chmod +rwx fichier ajoute toutes les permissions à un fichier (r =read; w = write; x = execute)
- tandis que chmod 755 fichier attribue des permissions spécifiques en utilisant la notation numérique (7 (*i.e. r+w+x*)pour le propriétaire, 5 (*i.e. r+x*) pour le groupe, 5 (*i.e. r+x*) pour les autres).

Exercise

```
> chmod 751 /work/user/yourID
```

- That means:



Command lines

Principales commandes Unix/Linux

- pwd (Print Working Directory) : Affiche le répertoire actuel.
- ls (List) : Liste les fichiers et dossiers dans le répertoire actuel.
- cd (Change Directory) : Change le répertoire de travail.
- mkdir (Make Directory) : Crée un nouveau répertoire.
- cp (Copy) : Copie des fichiers ou des répertoires.
- mv (Move) : Déplace ou renomme des fichiers ou des répertoires.
- rm (Remove) : Supprime des fichiers.
- man (Manual) : Affiche le manuel ou la documentation d'une commande

Naviguer dans le Système de Fichiers

- Pour se déplacer, utilisez la commande `cd` suivie du nom du répertoire.
- Utilisez `cd ..` pour remonter d'un niveau.

Manipulation de Fichiers

- Création, Copie, Déplacement et Suppression : Utilisation des commandes touch, cp, mv, et rm pour créer, copier, déplacer et supprimer des fichiers
- Utilisation des Wildcards (*, ?) : Les wildcards permettent de spécifier des motifs lors de la manipulation de plusieurs fichiers.

Commandes Avancées

- grep (Global Regular Expression Print) : Recherche de motifs dans les fichiers.
- chmod (Change Mode) : Modification des permissions des fichiers.
- ps (Process Status) et kill : Gestion des processus.

Exercice: Navigation de base

- Affichez le répertoire de travail actuel
- Listez les fichiers et répertoires dans le répertoire actuel
- Naviguez dans un sous-répertoire nommé Formation_FROGS
- Vérifiez le changement de répertoire

Solution :

```
[bleuet@genobioinfo2 ~]$ pwd
/home/bleuet
[bleuet@genobioinfo2 ~]$ ls
save  work
[bleuet@genobioinfo2 ~]$ cd work/Formation_FROGS/
[bleuet@genobioinfo2 Formation_FROGS]$ pwd
/home/bleuet/work/Formation_FROGS
```

Exercice: copie de fichier

- Créez un nouveau répertoire sous Formation_FROGS nommé Sequences
- Vérifiez que le nouveau répertoire a été créé
- Copier les données qui se trouve ici : /work/user/gpascal/FROGSJ4/Sequences/16Sv3v4_Mock_adn_bact.tar.gz dans le dossier Sequences
- Vérifiez le contenu du répertoire Sequences

Solution :

```
[bleuet@genobioinfo2 Formation_FROGS]$ pwd
/home/bleuet/work/Formation_FROGS
[bleuet@genobioinfo2 Formation_FROGS]$ mkdir Sequences
[bleuet@genobioinfo2 Formation_FROGS]$ cp /work/user/gpascal/FROGSJ4/Sequences/16Sv3v4_Mock_adn_bact.tar.gz Sequences/
[bleuet@genobioinfo2 Formation_FROGS]$ ls Sequences/
16Sv3v4_Mock_adn_bact.tar.gz
```

Exercice: Déplacement et renommage de fichiers

- Positionnez-vous dans le répertoire `Formation_FROGS/Sequences`
- Créer un fichier nommé `toto.txt`
- Renommer le en `readme.txt`
- Déplacer-le dans le dossier `Formation_FROGS`
- Vérifier le contenu du répertoire `Formation_FROGS`

Solutions :

```
[bleuet@genobioinfo2 Formation_FROGS]$ pwd
/home/bleuet/work/Formation_FROGS
[bleuet@genobioinfo2 Formation_FROGS]$ cd Sequences/
[bleuet@genobioinfo2 Sequences]$ touch toto.txt
[bleuet@genobioinfo2 Sequences]$ ls
16Sv3v4_Mock_adn_bact.tar.gz  toto.txt
[bleuet@genobioinfo2 Sequences]$ mv toto.txt readme.txt
[bleuet@genobioinfo2 Sequences]$ ls
16Sv3v4_Mock_adn_bact.tar.gz  readme.txt
[bleuet@genobioinfo2 Sequences]$ mv readme.txt ..
[bleuet@genobioinfo2 Sequences]$ ls
16Sv3v4_Mock_adn_bact.tar.gz
[bleuet@genobioinfo2 Sequences]$ cd ..
[bleuet@genobioinfo2 Formation_FROGS]$ ls
readme.txt  Sequences  test.txt
```

1

```
[bleuet@genobioinfo2 Formation_FROGS]$ cd Sequences/
[bleuet@genobioinfo2 Sequences]$ touch toto.txt
[bleuet@genobioinfo2 Sequences]$ mv toto.txt ../readme.txt
[bleuet@genobioinfo2 Sequences]$ ls
16Sv3v4_Mock_adn_bact.tar.gz
[bleuet@genobioinfo2 Sequences]$ cd ..
[bleuet@genobioinfo2 Formation_FROGS]$ ls
readme.txt  Sequences  test.txt
```

2

La commande mv sert à renommer et déplacer un fichier

Exercice: suppression de fichier

- Positionnez-vous dans le répertoire Formation_FROGS
- Supprimer le fichier nommé test.txt
- Vérifier le contenu du répertoire Formation_FROGS

Solutions :

```
[bleuet@genobioinfo2 ~]$ cd work/Formation_FROGS/
[bleuet@genobioinfo2 Formation_FROGS]$ ls
readme.txt  Sequences  test.txt
[bleuet@genobioinfo2 Formation_FROGS]$ rm test.txt
[bleuet@genobioinfo2 Formation_FROGS]$ ls
readme.txt  Sequences
```

La commande rm est indispensable mais très dangereuse.
Elle ne previent pas avant de supprimer vos fichiers.
A manipuler avec concentration

Exercice:

- Trouver les différentes option de la commande `ls`
- Afficher tous les fichiers du répertoire `Formation_FROGS` par ordre croissant de leur date de dernière modification (les plus anciens fichiers en haut de la liste)

> man ls

1. `-l`: Cette option affiche les informations détaillées sur les fichiers et répertoires, y compris les autorisations, le nombre de liens, le propriétaire, le groupe, la taille, et la date de dernière modification.
   Exemple :

```bash
ls -l
```

2. `-a`: L'option `-a` affiche tous les fichiers, y compris ceux dont le nom commence par un point. Les fichiers dont le nom commence par un point sont généralement considérés comme des fichiers cachés.
   Exemple :

```bash
ls -a
```

3. `-h`: Cette option rend les tailles de fichiers plus lisibles en utilisant des unités comme KB, MB, GB, etc.
   Exemple :

```bash
ls -lh
```

4. `-t`: L'option `-t` trie les fichiers par date de modification, affichant les plus récents en premier.
   Exemple :

```bash
ls -lt
```

5. `-r`: L'option `-r` inverse l'ordre de la liste, affichant les fichiers en ordre décroissant.
   Exemple :

```bash
ls -lr
```

Ces options peuvent également être combinées. Par exemple, pour lister tous les fichiers, y compris les fichiers cachés, avec des informations détaillées et des tailles lisibles, vous pouvez utiliser :

```bash
ls -alh
```

Solution :

```
[bleuet@genobioinfo2 Formation_FROGS]$ ls -altr
total 2
drwxr-x--x 13 bleuet formation 4096 Feb  2 16:17 ..
-rw-r--r--  1 bleuet formation    0 Feb  2 17:39 readme.txt
drwxr-xr-x  2 bleuet formation 4096 Feb  2 17:40 Sequences
drwxr-xr-x  3 bleuet formation 4096 Feb  2 17:44 .
```

`ls -altr` liste tous les fichiers, y compris les fichiers cachés,
avec des informations détaillées, triés par date de modification (les plus récents en haut).
C'est souvent utilisé pour voir les fichiers récemment modifiés ou créés dans un répertoire.

Exercice: utilisation des wildcards

- Positionnez-vous dans le répertoire Formation_FROGS/Sequences
- Créer 3 fichiers nommés toto.txt, titi.txt et tutu.txt
- Vérifier le contenu de Formation_FROGS/Sequences
- Déplacer en 1 seule commande ces 3 fichiers dans le dossier Formation_FROGS
- Vérifier le contenu de Formation_FROGS/Sequences
- Positionnez-vous dans le répertoire Formation_FROGS
- Vérifier le contenu de Formation_FROGS
- Supprimer ces 3 fichiers et SEULEMENT ces 3 fichiers en 1 seule commande

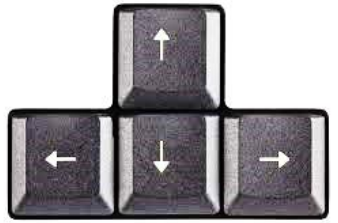
Solution :

```
[bleuet@genobioinfo2 Formation_FROGS]$ cd Sequences/
[bleuet@genobioinfo2 Sequences]$ ll
total 183808
-rw-r--r-- 1 bleuet formation 188211818 Feb  2 16:56 16Sv3v4_Mock_adn_bact.tar.gz
[bleuet@genobioinfo2 Sequences]$ touch toto.txt titi.txt tutu.txt
[bleuet@genobioinfo2 Sequences]$ ls
16Sv3v4_Mock_adn_bact.tar.gz  titi.txt  toto.txt  tutu.txt
[bleuet@genobioinfo2 Sequences]$ mv *txt ../
[bleuet@genobioinfo2 Sequences]$ ls
16Sv3v4_Mock_adn_bact.tar.gz
[bleuet@genobioinfo2 Sequences]$ cd ..
[bleuet@genobioinfo2 Formation_FROGS]$ ls
readme.txt  Sequences  titi.txt  toto.txt  tutu.txt
[bleuet@genobioinfo2 Formation_FROGS]$ rm t*.txt
[bleuet@genobioinfo2 Formation_FROGS]$ ls
readme.txt  Sequences
```

Conseils pratiques

- **Historique des Commandes :** Utilisez les touches fléchées de votre clavier pour naviguer dans l'historique des commandes.

  La commande `history` affiche une liste des commandes que vous avez tapées précédemment dans le terminal.



- **Auto-complétion avec Tab :** Appuyez sur la touche Tab pour compléter automatiquement les noms de fichiers et de répertoires.

- **Libérer le prompt** en mettant votre commande en tache de fond avec `&`
  ex: `geany &` *(lancer cet éditeur de texte sans et avec le `&`)*

FROGS command lines

Les données d’entrée

Les séquences à traiter: des questions à se poser

- Quels fichiers sont contenus dans le tar.gz ?
- Nom des fichiers ?
- Technologie de séquençage ?
- Quelle est la taille des lectures ?
- Région ciblée et longueur attendue de l'amplicon ?
- Les lectures ont-elles déjà été fusionnées ?
- Des amorces ont-elles déjà été supprimées ?
- Quelles sont les séquences des amorces ?

Quels fichiers dans le tar.gz ?

```
[bleuet@genobioinfo2 Formation_FROGS]$ cd Sequences/
[bleuet@genobioinfo2 Sequences]$ ll
total 183808
-rw-r--r-- 1 bleuet formation 188211818 Feb  2 16:56 16Sv3v4_Mock_adn_bact.tar.gz
[bleuet@genobioinfo2 Sequences]$ tar -xvf 16Sv3v4_Mock_adn_bact.tar.gz
MockADN_R1.fastq.gz
MockADN_R2.fastq.gz
Mockbact_R1.fastq.gz
Mockbact_R2.fastq.gz
```

- tar: C'est la commande de manipulation d'archives dans les systèmes de type UNIX/Linux.
- -x: Indique à tar d'extraire le contenu de l'archive.
- -v: Active le mode verbeux, c'est-à-dire qu'il affiche les fichiers qu'il extrait à mesure qu'il les traite.
- -f: Spécifie le nom de l'archive avec laquelle on veut travailler, ici 16Sv3v4_Mock_adn_bact.tar.gz

Les séquences à traiter: des questions à se poser

- Quels fichiers sont contenus dans le tar.gz ? 2 échantillons MockADN et Mockbact
- Nom des fichiers ? Du type MockADN_R1.fastq.gz -> OK
- Technologie de séquençage ? Illumina miseq
- Quelle est la taille des lectures ?
- Région ciblée et longueur attendue de l'amplicon ?
- Les lectures ont-elles déjà été fusionnées ?
- Des amorces ont-elles déjà été supprimées ?
- Quelles sont les séquences des amorces ?

Quelle est la taille des lectures ?

```
[bleuet@genobioinfo2 Sequences]$ zcat MockADN_R1.fastq.gz|awk 'NR%4==2 {print length($0); exit}'
250
```

zcat MockADN_R1.fastq.gz|awk 'NR%4==2 {print length($0); exit}'

- Le fichier de séquences est zippé, on le lit donc avec zcat
- Un séquence au format FASTQ est composée de 4 lignes
- Explication: Cette commande utilise zcat pour décompresser le fichier FASTQ et le passer à awk qui langage de programmation de traitement de texte et un utilitaire de ligne de commande dans les systèmes d'exploitation de type UNIX/Linux.

NR%4==2: Cette condition vérifie si le numéro de ligne (NR) modulo 4 est égal à 2. Cela correspond aux lignes qui contiennent les séquences dans un fichier FASTQ (les lignes 2, 6, 10, etc.).

{print length($0); exit}: Si la condition est vraie, cela imprime la longueur de la séquence de la ligne actuelle ($0 représente la ligne entière) et quitte le traitement du fichier.

Les séquences à traiter: des questions à se poser

- Quels fichiers sont contenus dans le tar.gz ? 2 échantillons MockADN et Mockbact
- Nom des fichiers ? Du type MockADN_R1.fastq.gz -> OK
- Technologie de séquençage ? Illumina miseq
- Quelle est la taille des lectures ? 250
- Région ciblée et longueur attendue de l'amplicon ? V3V4 – [300 – 490]
- Les lectures ont-elles déjà été fusionnées ? Non fichier R1 et R2
- Des amorces ont-elles déjà été supprimées ? Non
- Quelles sont les séquences des amorces ?
  5’ ACGGRAGGCAGCAG
  3’ AGGATTAGATACCCTGGTA

Panorama des outils

Les étapes FROGS : les outils

- Les noms des outils sont Galaxy sont différents des outils en python qui traitent les données.

FROGS 4.1

ASVS RECONSTRUCTION

FROGS_0 Demultiplex reads Attribute reads to samples in function of inner barcode

FROGS_1 Pre-process merging, denoising and dereplication

FROGS_2 Clustering swarm Single-linkage clustering on sequences

FROGS_Cluster_Stat Process some metrics on clusters

FROGS_3 Remove chimera Remove PCR chimera in each sample

FROGS_4 Cluster filters Filters clusters on several criteria.

FROGS ITSx Extract the highly variable ITS1 and ITS2 subregions from ITS sequences

FROGS_5 Taxonomic affiliation Taxonomic affiliation of each ASV's seed by RDPtools and BLAST

FROGS Affiliation Filters Filters ASVs on several affiliation criteria

FROGS Affiliation postprocess Aggregates ASVs based on alignment metrics

FROGS Abundance normalisation Normalise ASV abundance.

FROGS Tree Reconstruction of phylogenetic tree

FROGS_6_Affiliation_Stat Process some metrics on taxonomies

FROGS BIOM to std BIOM Converts a FROGS BIOM in fully compatible BIOM

FROGS BIOM to TSV Converts a BIOM file in TSV file

FROGS TSV_to_BIOM Converts a TSV file in a BIOM file 1

ASVS STRUCTURE AND COMPOSITION ANALYSIS

FROGSSTAT Phyloseq Import Data from 3 files: biomfile, samplefile, treefile

FROGSSTAT Phyloseq Composition Visualisation with bar plot and composition plot

FROGSSTAT Phyloseq Alpha Diversity with richness plot

FROGSSTAT Phyloseq Beta Diversity distance matrix

FROGSSTAT Phyloseq Sample Clustering of samples using different linkage methods

FROGSSTAT Phyloseq Structure Visualisation with heatmap plot and ordination plot

FROGSSTAT Phyloseq Multivariate Analysis Of Variance perform Multivariate Analysis of Variance (MANOVA)

DIFFERENTIAL ABUNDANCE ANALYSIS

FROGSSTAT DESeq2 Preprocess import a Phyloseq object and prepare it for DESeq2 differential abundance analysis

FROGSSTAT DESeq2 Visualisation to extract and visualise differentially abundant ASVs or functions

FUNCTIONNAL ABUNDANCE PREDICTIONS BASED ON MARKER GENE SEQUENCES

FROGSFUNC_1_placeseqs_and_copynumbers Places ASVs into a reference phylogenetic tree.

FROGSFUNC_2_functions Calculates functions abundances in each sample.

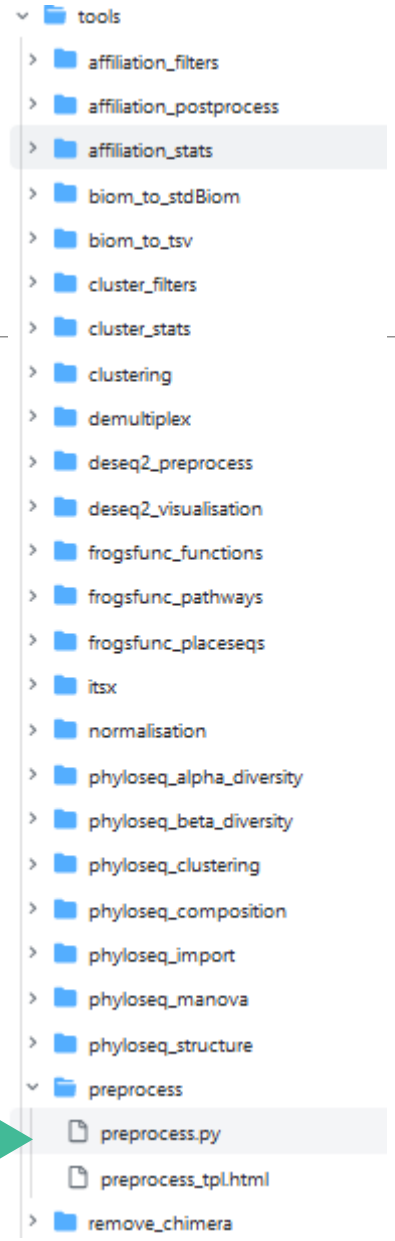
FROGSFUNC_3_pathways Calculates pathway abundances in each sample.

Les étapes FROGS : les outils

- Les noms des outils sont Galaxy sont différents des outils en python qui traitent les données.

- Tous les codes se trouvent ici https://github.com/geraldinepascal/FROGS

FROGS_1 Pre-process merging, denoising and dereplication



Chargement du logiciel dans votre environnement de travail

La commande module

- Trouver un logiciel par mot clé
- search_module FROGS

```
[bleuet@genobioinfo2 Formation_FROGS]$ search_module FROGS
bioinfo/FROGS/FROGSFUNC-v4.0.1
bioinfo/FROGS/FROGSFUNC-v4.1.0
bioinfo/FROGS/FROGS-v4.0.1
bioinfo/FROGS/FROGS-v4.1.0
```

- La dernière version de FROGS est donc disponible sous genotoul Bioinfo
- On remarque aussi que les outils FROGSFUNC sont à part et si on veut les utiliser il faudra charger ce module spécifique.

La commande module

- module show bioinfo/FROGS/FROGS-v4.1.0

```
[bleuet@genobioinfo1 ~]$ module show bioinfo/FROGS/FROGS-v4.1.0
-------------------------------------------------------------------
/tools/modulefiles/bioinfo/FROGS/FROGS-v4.1.0:

module-whatis   {loads the bioinfo/FROGS/FROGS-v4.1.0 environment}
prereq          devel/Miniconda/Miniconda3
-------------------------------------------------------------------
```

Il faudra donc lancer le module devel/Miniconda/Miniconda3 pour que FROGS fonctionne

- module show devel/Miniconda/Miniconda3

```
[bleuet@genobioinfo1 ~]$ module show devel/Miniconda/Miniconda3
-------------------------------------------------------------------
/tools/modulefiles/devel/Miniconda/Miniconda3:

module-whatis   {loads the devel/Miniconda/Miniconda3 environment}
prepend-path    PATH /usr/local/bioinfo/src/Miniconda/Miniconda3/bin/
-------------------------------------------------------------------
```

Ici pas de pré-requis au lancement de ce module

La commande module

La commande module load

- Elle permet aux utilisateurs de charger et décharger facilement des logiciels, des bibliothèques et des environnements logiciels dans leurs sessions de terminal X.
- Lorsque vous exécutez la commande module load, vous chargez un module spécifique dans votre environnement. Un module peut être une version spécifique d'un logiciel, une bibliothèque, ou un ensemble d'environnements logiciels et de variables d'environnement associées.

readme.txt: notre cahier de labo

- Nous allons écrire dans le fichier readme.txt l'ensemble des lignes de commande qui nous sera necessaire pour éxécuter les différentes de FROGS
- Ouvrir readme.txt avec geany : geany readme.txt &
- Y copier ces lignes:

```
#Note de formation

module load devel/Miniconda/Miniconda3
module load bioinfo/FROGS/FROGS-v4.1.0
```

Organisation de notre répertoire de travail

Les logs

- Les « logs » dans le contexte informatique font référence à des fichiers ou à des enregistrements qui contiennent des informations détaillées sur l'exécution d'un programme, d'un système ou d'une commande.
- Ces fichiers sont généralement utilisés pour le suivi, la détection des erreurs, le dépannage et la maintenance des systèmes et des applications.
- On va donc créer un répertoire pour les conserver:

```
[bleuet@genobioinfo2 Formation_FROGS]$ mkdir Logs
[bleuet@genobioinfo2 Formation_FROGS]$ ls
Logs   readme.txt   Sequences
```

Les fichiers de sorties

- Chaque étape de FROGS va générer des sorties.
- Nous allons les conserver dans un dossier nommé Results que l'on va créer :

```
[bleuet@genobioinfo2 Formation_FROGS]$ mkdir Results
[bleuet@genobioinfo2 Formation_FROGS]$ ls
Logs   readme.txt  Results  Sequences
```

readme.txt: notre cahier de labo

- Notons dans le readme.txt ces chemin vers ces dossiers comme variables
- Ouvrir readme.txt avec geany : geany readme.txt &
- Y copier ces lignes:

```
SEQ_DIR=Sequences
LOG_DIR=Logs
RES_DIR=Results
```

SEQ_DIR=Sequences: Définit la variable **SEQ_DIR** avec la valeur « Sequences ». « Sequences » est en fait l'adresse du répertoire où se trouve nos données à traiter.

LOG_DIR=Logs: Définit la variable LOG_DIR avec la valeur « Logs ». « Logs » est en fait l'adresse du répertoire. Les fichiers logs seront alors enregistrés dans ce répertoire.

RES_DIR=Results: Définit la variable RES_DIR avec la valeur « Results ». Cela sera utilisé pour spécifier le répertoire où les résultats du process seront enregistrés.

1er outil: preprocess.py

Les paramètres

- Pour connaitre les paramètres de preprocess.py il faut charger FROGS dans notre espace de travail :

```
[bleuet@genobioinfo2 Sequences]$ module load devel/Miniconda/Miniconda3
[bleuet@genobioinfo2 Sequences]$ module load bioinfo/FROGS/FROGS-v4.1.0
(FROGS-v4.1.0_env) [bleuet@genobioinfo2 Sequences]$
```

Le prompt a changé !

Les paramètres

- Pour connaitre les paramètres de preprocess.py, on fait appel à l'aide

```
(FROGS-v4.1.0_env) [bleuet@genobioinfo2 Sequences]$ preprocess.py --help
usage: preprocess.py [-h] [-v] {illumina,longreads,454} ...

Pre-process amplicons to use reads in diversity analysis.

positional arguments:
  {illumina,longreads,454}
    illumina            Illumina sequencers.
    longreads           longreads sequencers.
    454                 454 sequencers.

optional arguments:
  -h, --help            show this help message and exit
  -v, --version         show program's version number and exit
```

L'aide nous dit qu'il y a 3 sous rubriques à l'aide.

Nos allons choisir l'aide pour les données illumina

```
(FROGS-v4.1.0_env) [bleuet@genobioinfo1 Formation_FROGS]$ preprocess.py illumina --help
usage:
  For samples files:
    preprocess.py illumina
      --input-R1 R1_FILE [R1_FILE ...]
      --already-contiged | --input-R2 R2_FILE [R2_FILE ...] --R1-size R1_SIZE --R2-size
 ] [--merge-software {vsearch,flash,pear} [--expected-amplicon-size]] [--keep-unmerged]
      --min-amplicon-size MIN_AMPLICON_SIZE
      --max-amplicon-size MAX_AMPLICON_SIZE
      --without-primers | --five-prim-primer FIVE_PRIM_PRIMER --three-prim-primer THREE_
      [--samples-names SAMPLE_NAME [SAMPLE_NAME ...]]
      [--nb-cpus NB_CPUS] [--debug]
      [--output-dereplicated DEREPLICATED_FILE] [--output-count COUNT_FILE]
      [--summary SUMMARY_FILE] [--log-file LOG_FILE]

  For samples archive:
    preprocess.py illumina
      --input-archive ARCHIVE_FILE
      --already-contiged | --R1-size R1_SIZE --R2-size R2_SIZE [--mismatch-rate RATE ]
sh,pear} [--expected-amplicon-size]] [--keep-unmerged]
      --min-amplicon-size MIN_AMPLICON_SIZE
      --max-amplicon-size MAX_AMPLICON_SIZE
      --without-primers | --five-prim-primer FIVE_PRIM_PRIMER --three-prim-primer THREE_
      [--nb-cpus NB_CPUS] [--debug]
      [--output-dereplicated DEREPLICATED_FILE] [--output-count COUNT_FILE] [--artComb-
utput-count ART_COUNT_FILE]
      [--summary SUMMARY_FILE] [--log-file LOG_FILE]

optional arguments:
  -h, --help            show this help message and exit
  --already-contiged    The archive contains 1 file by sample : Reads 1 and
                        Reads 2 are already contiged by pair.
  --R1-size R1_SIZE     The read1 size.
  --R2-size R2_SIZE     The read2 size.
  --merge-software {vsearch,flash,pear}
                        Software used to merge paired reads [Default: vsearch]
  --mismatch-rate MISMATCH_RATE
                        Maximum mismatch rate in overlap region. [Default:
                        0.1; must be expressed as decimal, between 0 and 1]
  --quality-scale {33,64}
                        The phred base quality scale, either 33 or 64 if using
                        Vsearch as read pair merge software [Default: 33]
  --keep-unmerged       In case of uncontiged paired reads, keep unmerged, and
                        artificially combined them with 100 Ns.
  --expected-amplicon-size EXPECTED_AMPLICON_SIZE
                        The expected size for the majority of the amplicons
                        (with primers), if using Flash as read pair merge
                        software.
  --min-amplicon-size MIN_AMPLICON_SIZE
                        The minimum size for the amplicons (with primers).
  --max-amplicon-size MAX_AMPLICON_SIZE
                        The maximum size for the amplicons (with primers).
  --five-prim-primer FIVE_PRIM_PRIMER
                        The 5' primer sequence (wildcards are accepted).
  --three-prim-primer THREE_PRIM_PRIMER
                        The 3' primer sequence (wildcards are accepted).
  --without-primers     Use this option when you use custom sequencing primers
```

On tape dans le prompt:
preprocess.py illumina –help

En lisant l'aide on comprend qu'on est concerné par la partie
For sample archive

Les paramètres: comment les lire ?

```
For samples archive:
  preprocess.py illumina
    --input-archive ARCHIVE_FILE
    --already-contiged | --R1-size R1_SIZE --R2-size R2_SIZE [--mismatch-rate RATE ] [--quality-scale SCALE ] [--merge-software {vsearch,flash,pear} [--expected-amplicon-size]] [--keep-unmerged]
    --min-amplicon-size MIN_AMPLICON_SIZE
    --max-amplicon-size MAX_AMPLICON_SIZE
    --without-primers | --five-prim-primer FIVE_PRIM_PRIMER --three-prim-primer THREE_PRIM_PRIMER
    [--nb-cpus NB_CPUS] [--debug]
    [--output-dereplicated DEREPLICATED_FILE] [--output-count COUNT_FILE] [--artComb-output-dereplicated ART_DEREPLICATED_FILE] [--artComb-output-count ART_COUNT_FILE]
    [--summary SUMMARY_FILE] [--log-file LOG_FILE]
```

- 1 ligne commençant par « -- » signifie 1 choix à faire, en bleu le nom du paramètre, en blanc la variable
- `--input-archive ARCHIVE_FILE`
  La 1ère ligne, il n'y a pas de question à se poser, nous devrons écrire ce paramètre dans notre ligne de commande
- Pour la 2ème ligne on observe la présence d'un pipe « | » de séparation donc on doit choisir entre 2 options

  `--already-contiged` | `--R1-size R1_SIZE --R2-size R2_SIZE`

  On choisira pour nos données non-contiguées l'option `--R1-size R1_SIZE --R2-size R2_SIZE` ....

  A la suite, il y a des paramètres entre crochets, cela signifie que des valeurs par défaut ont été attribuées à ces paramètres et que nous pouvons les écrire dans la ligne de commande si nous souhaitons changer la valeur attribuée par défaut. Si nous n'écrivons pas dans la ligne de commande ces paramètres, la valeur par défaut sera prise alors en compte.

Les paramètres: comment les lire ?

```
For samples archive:
  preprocess.py illumina
    --input-archive ARCHIVE_FILE
    --already-contiged | --R1-size R1_SIZE --R2-size R2_SIZE [--mismatch-rate RATE ] [--quality-scale SCALE ] [--merge-software {vsearch,flash,pear} [--expected-amplicon-size]] [--keep-unmerged]
    --min-amplicon-size MIN_AMPLICON_SIZE
    --max-amplicon-size MAX_AMPLICON_SIZE
    --without-primers | --five-prim-primer FIVE_PRIM_PRIMER --three-prim-primer THREE_PRIM_PRIMER
    [--nb-cpus NB_CPUS] [--debug]
    [--output-dereplicated DEREPLICATED_FILE] [--output-count COUNT_FILE] [--artComb-output-dereplicated ART_DEREPLICATED_FILE] [--artComb-output-count ART_COUNT_FILE]
    [--summary SUMMARY_FILE] [--log-file LOG_FILE]
```

```
--merge-software {vsearch,flash,pear}
                      Software used to merge paired reads [Default: vsearch]
--mismatch-rate MISMATCH_RATE
                      Maximum mismatch rate in overlap region. [Default:
                      0.1; must be expressed as decimal, between 0 and 1]
--quality-scale {33,64}
                      The phred base quality scale, either 33 or 64 if using
                      Vsearch as read pair merge software [Default: 33]
--keep-unmerged       In case of uncontiged paired reads, keep unmerged, and
                      artificially combined them with 100 Ns.
--expected-amplicon-size EXPECTED_AMPLICON_SIZE
                      The expected size for the majority of the amplicons
                      (with primers), if using Flash as read pair merge
                      software.
```

- On choisira seulement d'écrire dans notre commande

  `--merge-software vsearch` (valeur par défaut)

  Ou

  `--merge-software pear` par exemple

Les paramètres: comment les lire ?

```
For samples archive:
  preprocess.py illumina
    --input-archive ARCHIVE_FILE
    --already-contiged | --R1-size R1_SIZE --R2-size R2_SIZE [--mismatch-rate RATE ] [--quality-scale SCALE ] [--merge-software {vsearch,flash,pear} [--expected-amplicon-size]] [--keep-unmerged]
    --min-amplicon-size MIN_AMPLICON_SIZE
    --max-amplicon-size MAX_AMPLICON_SIZE
    --without-primers | --five-prim-primer FIVE_PRIM_PRIMER --three-prim-primer THREE_PRIM_PRIMER
    [--nb-cpus NB_CPUS] [--debug]
    [--output-dereplicated DEREPLICATED_FILE] [--output-count COUNT_FILE] [--artComb-output-dereplicated ART_DEREPLICATED_FILE] [--artComb-output-count ART_COUNT_FILE]
    [--summary SUMMARY_FILE] [--log-file LOG_FILE]
```

- La 3ème et 4ème ligne sont à choix unique `--min-amplicon-size MIN_AMPLICON_SIZE` `--max-amplicon-size MAX_AMPLICON_SIZE`
- Pour la 5ème ligne on choisira car nos données ont encore leur amorces de PCR
  `--five-prim-primer FIVE_PRIM_PRIMER --three-prim-primer THREE_PRIM_PRIMER`
- Pour les derniers paramètres ils sont tous optionnels car beaucoup on des valeurs par défaut (notamment les fichiers de sortie)

```
[--nb-cpus NB_CPUS] [--debug]
[--output-dereplicated DEREPLICATED_FILE] [--output-count COUNT_FILE] [--artComb-output-dereplicated ART_DEREPLICATED_FILE] [--artComb-output-count ART_COUNT_FILE]
[--summary SUMMARY_FILE] [--log-file LOG_FILE]
```

  nous garderons : `--nb-cpu NB_CPUS`

- Certains paramètres peuvent avoir 2 systèmes d'écriture totalement équivalent : `-p NB_CPUS, --nb-cpus NB_CPUS` pour davantage de clarté nous privilégierons les paramètres explicites.

Les paramètres: comment les lire ?

```
Outputs:
  -d OUTPUT_DEREPLICATED, --output-dereplicated OUTPUT_DEREPLICATED
                        FASTA file with unique sequences. Each sequence has an
                        ID ended with the number of initial sequences
                        represented (example : ">a0101;size=10"). [Default:
                        preprocess.fasta]
  -c OUTPUT_COUNT, --output-count OUTPUT_COUNT
                        TSV file with count by sample for each unique sequence
                        (example with 3 samples : "a0101<TAB>5<TAB>8<TAB>0").
                        [Default: preprocess_counts.tsv]
  -s SUMMARY, --summary SUMMARY
                        The HTML file containing the graphs. [Default:
                        preprocess.html]
  -l LOG_FILE, --log-file LOG_FILE
                        This output file will contain several information on
                        executed commands.
```

- 4 fichiers en sortie sont demandés, nous les écrirons donc ces 4 paramètres dans la commande

La ligne de commande preprocess.py

La ligne de commande avec les valeurs de paramètres appropriées au jeu de données 16Sv3v4_Mock_adn_bact.tar.gz

```
preprocess.py illumina
--input-archive $SEQ_DIR/16Sv3v4_Mock_adn_bact.tar.gz
--R1-size 250
--R2-size 250
--merge-software vsearch
--min-amplicon-size 300
--max-amplicon-size 490
--five-prim-primer ACGGRAGGCAGCAG
--three-prim-primer AGGATTAGATACCCTGGTA
--nb-cpus 8
--output-dereplicated $RES_DIR/01-preprocess.fa
--output-count $RES_DIR/01-preprocess.count.tsv
--summary $RES_DIR/01-preprocess.html
--log-file $LOG_DIR/01-preprocess.log
```

readme.txt: notre cahier de labo

- Notons dans le readme.txt cette ligne de commande

```
preprocess.py illumina --input-archive $SEQ_DIR/16Sv3v4_Mock_adn_bact.tar.gz --R1-size 250 --R2-size 250 --merge-software vsearch --min-amplicon-size 300 --max-amplicon-size 490 --five-prim-primer ACGGRAGGCAGCAG --three-prim-primer AGGATTAGATACCCTGGTA --nb-cpus 8 --output-dereplicated $RES_DIR/01-preprocess.fa --output-count $RES_DIR/01-preprocess.count.tsv --summary $RES_DIR/01-preprocess.html --log-file $LOG_DIR/01-preprocess.log
```

On peut aussi l'écrire sur plusieurs ligne avec l'aide de « \ » comme cela:

```
preprocess.py illumina \
    --input-archive $SEQ_DIR/16Sv3v4_Mock_adn_bact.tar.gz \
    --R1-size 250 --R2-size 250 \
    --merge-software vsearch --min-amplicon-size 300 --max-amplicon-size 490 \
    --five-prim-primer ACGGRAGGCAGCAG --three-prim-primer AGGATTAGATACCCTGGTA \
    --nb-cpus 8 \
    --output-dereplicated $RES_DIR/01-preprocess.fa \
    --output-count $RES_DIR/01-preprocess.count.tsv \
    --summary $RES_DIR/01-preprocess.html \
    --log-file $LOG_DIR/01-preprocess.log
```

Lancement d’une commande sur le cluster de calcul

Ce que l’on a chacun sur genotoul:

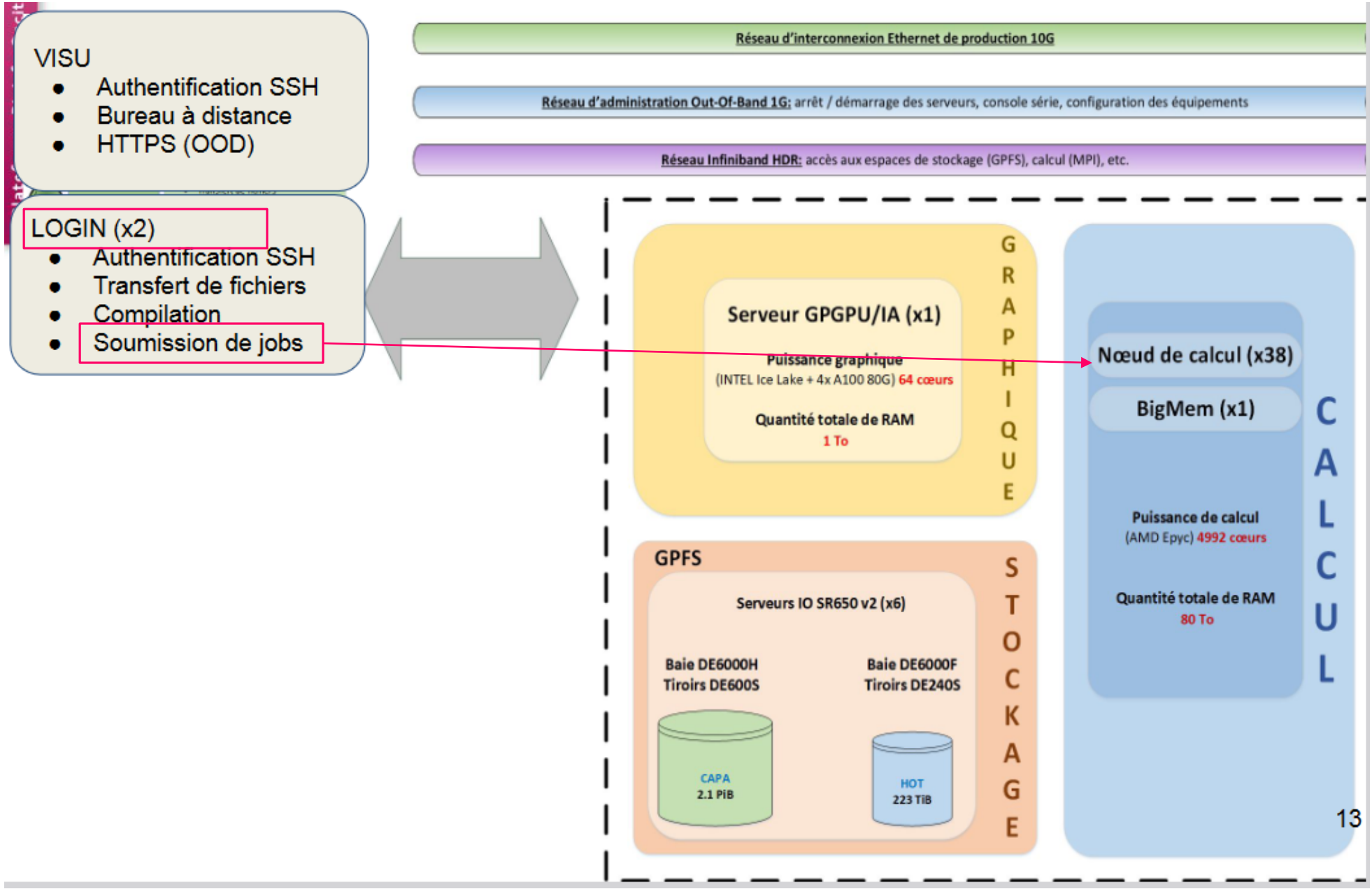
User disk spaces

- Home (10GB): configuration files only
- Save (250GB): 30-day history + replication to remote site
- Work (1TB): 7-day history (only), NO replication

Computing hours

- 100,000 h annual initial quota with possibility of extension
- 3 user groups (contributors, INRAE / OCCITANIE, others)

Donc pour avoir de la puissance de calcul il faut utiliser le cluster


VISU
Authentification SSH
Bureau à distance
HTTPS (OOD)
LOGIN (x2)
Authentification SSH
Transfert de fichiers
Compilation
Soumission de jobs
Réseau d'interconnexion Ethernet de production 10G
Réseau d'administration Out-Of-Band 1G: arrêt / démarrage des serveurs, console série, configuration des équipements
Réseau Infiniband HDR: accès aux espaces de stockage (GPFS), calcul (MPI), etc.
GRAPHIQUE
Serveur GPGPU/IA (x1)
Puissance graphique
(INTEL Ice Lake + 4x A100 80G) 64 cœurs
Quantité totale de RAM
1 To
STOCKAGE
GPFS
Serveurs IO SR650 v2 (x6)
Baie DE6000H
Tiroirs DE600S
Baie DE6000F
Tiroirs DE240S
CAPA
2.1 PiB
HOT
223 TiB
CALCUL
Nœud de calcul (x38)
BigMem (x1)
Puissance de calcul
(AMD Epyc) 4992 cœurs
Quantité totale de RAM
80 To
13

Sbatch

Cette commande est utilisée pour soumettre un script de tâche à SLURM le gestionnaire de process. Les options de cette commande:

-J, --job-name=NAME : Nom de la tâche ou du job.

-o, --output=FILE : Fichier de sortie pour la sortie standard de la tâche.

-e, --error=FILE : Fichier de sortie pour les messages d'erreur.

--cpus-per-task=NUM : Nombre de cœurs CPU à allouer par tâche.

--mem=SIZE : Mémoire à allouer à la tâche.

--time=TIME : Limite de temps pour la tâche.

--wrap="COMMAND" : Commande à exécuter en tant que tâche.

-N, --nodes=NUM : Nombre total de nœuds à utiliser

sbatch lancer pour le preprocess.py

-t 20 : On suppose que notre preprocess.py se terminera en moins de 20 minutes.

-J frogs_preprocess: Définit le nom de la tâche job comme "frogs_preprocess".

-o $LOG_DIR/%x.out: Spécifie le fichier de sortie pour la sortie standard du job. Ici, $LOG_DIR est une variable définie au début du script, représentant probablement le répertoire des logs. %x est un indicateur qui sera remplacé par le nom du job, dans ce cas, frogs_preprocess. Ainsi, la sortie standard du job sera enregistrée dans un fichier sous le dossier Logs i.e. Logs/frogs_preprocess.out.

-e $LOG_DIR/%x.err: Spécifie le fichier de sortie pour les messages d'erreur. De manière similaire à la sortie standard, les erreurs seront enregistrées dans un fichier sous le dossier Logs i.e.: Logs/frogs_preprocess.err".

--cpus-per-task=4: Alloue 4 cœurs CPU par tâche.

--mem=32G: Alloue 32 gigaoctets de mémoire par CPU.

--wrap="preprocess.py illumina ...": Spécifie la commande à exécuter. Ici, elle exécute le script Python "preprocess.py" avec plusieurs arguments pour effectuer le prétraitement de données illumina.

readme.txt: notre cahier de labo

- Notons dans le readme.txt cette ligne de commande

```
sbatch -t 20 -J frogs_preprocess -o $LOG_DIR/%x.out -e $LOG_DIR/%x.err --cpus-per-task=8 --mem=32G –wrap="preprocess.py … "
```

Combinons-là avec la commande preprocess.py précédente:

```
sbatch -t 20 -J frogs_preprocess -o $LOG_DIR/%x.out -e $LOG_DIR/%x.err --cpus-per-task=8 --mem=32G –wrap="preprocess.py illumina --input-archive $SEQ_DIR/16Sv3v4_Mock_adn_bact.tar.gz --R1-size 250 --R2-size 250 --merge-software vsearch --min-amplicon-size 300 --max-amplicon-size 490 --five-prim-primer ACGGRAGGCAGCAG --three-prim-primer AGGATTAGATACCCTGGTA --nb-cpus 8 --output-dereplicated $RES_DIR/01-preprocess.fa --output-count $RES_DIR/01-preprocess.count.tsv --summary $RES_DIR/01-preprocess.html --log-file $LOG_DIR/01-preprocess.log"
```

Commandes de control du process

- squeue display jobs and their state
  - -j <job_ID> : report more info about a particular job (wildcard is possible)
  - -u <user> : report job information for a specific user
  - ex: squeue -u bleuet
- scancel cancel a job or set of jobs
  - scancel <job_ID>
  - -u : restrict cancel to jobs owned by this user
  - -t : restrict cancel to jobs in this state
  - ex: scancel -u bleuet -t PD
- seff job_ID : affiche des statistiques sur l'utilisation des ressources par la tâche, telles que le temps CPU, la mémoire utilisée, et d'autres informations liées à la performance.

Lancement du preprocess sur le cluster de calcul

On lance le process:

On a bien lancé les 2 « module load »

```
(FROGS-v4.1.0_env) [bleuet@genobioinfo2 Formation_FROGS]$ SEQ_DIR=Sequences
(FROGS-v4.1.0_env) [bleuet@genobioinfo2 Formation_FROGS]$ LOG_DIR=Logs
(FROGS-v4.1.0_env) [bleuet@genobioinfo2 Formation_FROGS]$ RES_DIR=Results
(FROGS-v4.1.0_env) [bleuet@genobioinfo2 Formation_FROGS]$ sbatch -t 20 -J frogs_preprocess -o $LOG_DIR/%x.out -e $LOG_DIR/%x.err --cpus-per-task=8 --mem=32G --wrap="preprocess.py il
lumina --input-archive $SEQ_DIR/16Sv3v4_Mock_adn_bact.tar.gz --R1-size 250 --R2-size 250 --merge-software vsearch --min-amplicon-size 300 --max-amplicon-size 490 --five-prim-primer
ACGGRAGGCAGCAG --three-prim-primer AGGATTAGATACCCTGGTA --nb-cpus 8 --output-dereplicated $RES_DIR/01-preprocess.fa --output-count $RES_DIR/01-preprocess.count.tsv --summary $RES_DI
R/01-preprocess.html --log-file $LOG_DIR/01-preprocess.log"
Submitted batch job 4396355
(FROGS-v4.1.0_env) [bleuet@genobioinfo2 Formation_FROGS]$ squeue -u bleuet
             JOBID PARTITION     NAME     USER ST       TIME  NODES NODELIST(REASON)
           4396355     workq frogs_pr   bleuet  R       0:33      1 n023
```

On déclare nos 3 variables

On lance la commande sbatch qui lance la commande preprocess.py

Le numéro du job lancé

Pour connaitre le statut du job

Statut du job (R (running), PD (pending en attente, CG (a planté))

On lance seff pour connaitre l'efficacité du job:

```
(FROGS-v4.1.0_env) [bleuet@genobioinfo2 Formation_FROGS]$ seff 4396355
Job ID: 4396355
Cluster: genobioinfo
User/Group: bleuet/formation
State: COMPLETED (exit code 0)
Nodes: 1
Cores per node: 8
CPU Utilized: 00:01:48
CPU Efficiency: 14.84% of 00:12:08 core-walltime
Job Wall-clock time: 00:01:31
Memory Utilized: 204.10 MB
Memory Efficiency: 0.62% of 32.00 GB
```

Qu'en pensez-vous ?
Que pourrions-nous faire la prchaine fois pour ce meme type de données ?

Les checkpoints:

- Vérification des logs.
- .err : vide = OK
- .log : contient l'ensemble des lignes de commandes lancées par preprocess.py. On peut le lire avec more Logs/01-preprocess.log

```
(FROGS-v4.1.0_env) [bleuet@genobioinfo2 Formation_FROGS]$ ll
total 18
drwxr-xr-x 2 bleuet formation  4096 Feb  5 15:57 Logs
-rw-r--r-- 1 bleuet formation  1150 Feb  5 15:57 readme.txt
drwxr-xr-x 2 bleuet formation 16384 Feb  5 15:59 Results
drwxr-xr-x 2 bleuet formation  4096 Feb  2 22:34 Sequences
(FROGS-v4.1.0_env) [bleuet@genobioinfo2 Formation_FROGS]$ ll Logs/
total 16
-rw-r--r-- 1 bleuet formation 8370 Feb  5 15:59 01-preprocess.log
-rw-r--r-- 1 bleuet formation    0 Feb  5 15:56 frogs_preprocess.err
-rw-r--r-- 1 bleuet formation    0 Feb  5 15:56 frogs_preprocess.out
(FROGS-v4.1.0_env) [bleuet@genobioinfo2 Formation_FROGS]$ more Logs/01-preprocess.log
```

Lancement du clustering sur le cluster de calcul

CLUSTERING.PY

Les étape à suivre

1. Connaitre le nom du programme à lancer: https://github.com/geraldinepascal/FROGS/tree/master/tools
2. Déployer l'environnement FROGS dans son terminal (module load)
3. Lancer l'aide sur le programme pour connaitre les options (--help)
4. Écrire la commande dans son readme.txt
5. Combiner à la commande sbatch paramétrée
6. Déclarer ses variables (i.e. SEQ_DIR)
7. Lancer la commande dans son terminalX
8. Lire les logs.
9. Voir les résultats

La commande:

```
# clustering

sbatch -t 20 -J frogs_clustering -o $LOG_DIR/%x.out -e $LOG_DIR/%x.err --cpus-per-task=8 --mem=32G --wrap="clustering.py --nb-cpus 8 -d 1 --fastidious --input-fasta $RES_DIR/01-preprocess.fa --input-count $RES_DIR/01-preprocess.count.tsv --output-biom $RES_DIR/02-clustering.biom --output-fasta $RES_DIR/02-clustering.fasta  --output-compo $RES_DIR/02-clustering_compo.tsv --log-file $LOG_DIR/02-clustering.log"
```

La suite des commandes:

#clusterstat

```
sbatch -t 20 -J frogs_clustering_cluster_stat --mem=32G -o $LOG_DIR/%x.out -e $LOG_DIR/%x.err --wrap="cluster_stats.py --input-biom $RES_DIR/02-clustering.biom -o $RES_DIR/02-clustering_cluster_stat.html --log-file $LOG_DIR/02-clustering_cluster_stat.log"
```

 # remove chimera

```
sbatch -t 20 -J frogs_rm_chim -o $LOG_DIR/%x.out -e $LOG_DIR/%x.err --cpus-per-task=8 --mem=32G --wrap="remove_chimera.py --nb-cpus 8 --input-fasta $RES_DIR/02-clustering.fasta --input-biom $RES_DIR/02-clustering.biom --non-chimera $RES_DIR/03-non_chimera.fasta --out-abundance $RES_DIR/03-non_chimera.biom --summary $RES_DIR/03-remove_chimera.html --log-file $LOG_DIR/03-remove_chimera.log"
```

```
sbatch -t 20 -J frogs_rm_chim_cluster_stat --mem=32G -o $LOG_DIR/%x.out -e $LOG_DIR/%x.err --wrap="cluster_stats.py --input-biom $RES_DIR/03-non_chimera.biom -o $RES_DIR/03-non_chimera_cluster_stat.html --log-file $LOG_DIR/03-non_chimera_cluster_stat.log«
```

 # cluster filters

```
PHIX=/save/user/frogs/galaxy_databanks/phiX_genome/phi.fa
```

```
sbatch -t 20 -J frogs_filter --cpus-per-task=8 --mem=32G -o $LOG_DIR/%x.out -e $LOG_DIR/%x.err --wrap="cluster_filters.py --nb-cpus 8 --min-abundance 0.00005 --input-biom $RES_DIR/03-non_chimera.biom --input-fasta $RES_DIR/03-non_chimera.fasta --contaminant $PHIX --output-biom $RES_DIR/04-filters.biom --output-fasta $RES_DIR/04-filters.fasta --summary $RES_DIR/04-filters.html --excluded $RES_DIR/04-filters_excluded.tsv --log-file $LOG_DIR/04-filters.log"
```

```
sbatch -t 20 -J frogs_filter_cluster_stat --mem=32G -o $LOG_DIR/%x.out -e $LOG_DIR/%x.err --wrap="cluster_stats.py --input-biom $RES_DIR/04-filters.biom -o $RES_DIR/04-filters_cluster_stat.html --log-file $LOG_DIR/04-filters_cluster_stat.log"
```

La suite des commandes:

```
# affiliation

DATABQ=/save/user/frogs/galaxy_databanks/SILVA/16S/silva_138.1_16S_pintail80/silva_138.1_16S_pintail80.fasta

sbatch -t 20 -J frogs_affi --cpus-per-task=32 --mem=160G -o $LOG_DIR/%x.out -e $LOG_DIR/%x.err --
wrap="taxonomic_affiliation.py --nb-cpus 32 -m 160 --rdp --reference $DATABQ --input-biom $RES_DIR/04-filters.biom --
input-fasta $RES_DIR/04-filters.fasta --output-biom $RES_DIR/05-affiliation.biom --summary $RES_DIR/05-
affiliation.html --log-file $LOG_DIR/05-affiliation.log"

sbatch -t 20 -J frogs_blast_affiStat -o $LOG_DIR/%x.out -e $LOG_DIR/%x.err --wrap="affiliation_stats.py --
rarefaction-ranks Class Order Family Genus Species --multiple-tag blast_affiliations --tax-consensus-tag
blast_taxonomy --identity-tag perc_identity --coverage-tag perc_query_coverage -i $RES_DIR/05-affiliation.biom --
output-file $RES_DIR/05-affiliation_stat.html --log-file $LOG_DIR/05-affiliation_stat.log"

sbatch -t 20 -J frogs_rdp_affiStat -o $LOG_DIR/%x.out -e $LOG_DIR/%x.err --wrap="affiliation_stats.py --rarefaction-
ranks Class Order Family Genus Species --taxonomy-tag rdp_taxonomy --bootstrap-tag rdp_bootstrap -i $RES_DIR/05-
affiliation.biom --output-file $RES_DIR/05-affiliation_rdp_stat.html --log-file $LOG_DIR/05-affiliation_rdp_stat.log"

# tree

sbatch -t 20 -J frogs_tree --cpus-per-task=32 --mem=120G -o $LOG_DIR/%x.out -e $LOG_DIR/%x.err --wrap="tree.py --nb-
cpus 32 --input-sequences $RES_DIR/04-filters.fasta --biom-file $RES_DIR/05-affiliation.biom --out-tree $RES_DIR/06-
tree.nwk --html $RES_DIR/06-tree_summary.html --log-file $LOG_DIR/06-tree.log"

# to TSV

sbatch -t 20 -J frogs_biom_to_tsv -o $LOG_DIR/%x.out -e $LOG_DIR/%x.err --wrap="biom_to_tsv.py --input-biom
$RES_DIR/05-affiliation.biom --input-fasta $RES_DIR/04-filters.fasta --output-tsv $RES_DIR/05-affiliation.tsv --
output-multi-affi $RES_DIR/05-affiliation.multihit.tsv --log-file $LOG_DIR/05-affiliation.biom_to_tsv.log"
```

La suite des commandes:

```
# phyloseq_import
METADATA=$SEQ_DIR/le_nom_de_votre_fichier_de_metadata.tsv
sbatch -t 20 -J frogs_import_Phyloseq -o $LOG_DIR/%x.out -e
$LOG_DIR/%x.err --wrap="phyloseq_import_data.py --biomfile
$RES_DIR/05-affiliation.biom --sample $METADATA --treefile
$RES_DIR/06-tree.nwk --rdata $RES_DIR/05-affiliation.Rdata --html
$RES_DIR/phyloseq_summary.html --log-file
$LOG_DIR/phyloseq_import.log"
```

??

Lancer cette commande:

```
DATABQ=/save/user/frogs/galaxy_databanks/SILVA/16S/silva_138.1_16S_pintai
l80/silva_138.1_16S_pintail80.fasta

sbatch -t 20 -J frogs_affi --cpus-per-task=32 --mem=100G -o
$LOG_DIR/%x.out -e $LOG_DIR/%x.err --wrap="taxonomic_affiliation.py --nb-
cpus 32 -m 160 --rdp --reference $DATABQ --input-biom $RES_DIR/04-
filters.biom --input-fasta $RES_DIR/04-filters.fasta --output-biom
$RES_DIR/05-refseq-affiliation.biom --summary $RES_DIR/05-refseq-
affiliation.html --log-file $LOG_DIR/05-refseq-affiliation.log"
```

Y a-t-il un fichier .err > 0 et si oui, pourquoi? Que faut-il faire ?

??

Quelle serait la commande pour lancer un preprocess.py sur des longreads ?

Quelle serait la commande pour lancer un preprocess.py avec l'outil pear plutôt que vsearch, en gardant les lectures non contigables sur un échantillon nommé ech1 don't les sequence n'ont plus les primers ?

Où se trouve la banque de données dédiée au gène entier rRNA 16S bactérien.

??

Relancer preprocess.py sur le jeu de données

http://genoweb.toulouse.inra.fr/~formation/15_FROGS/Webinar_data/chaillou_withprimers_64renamedsamples_V1V3_10000seq_R1R2.tar.gz

Et si on faisait un pipeline ?

```
#preprocess
job1=$(sbatch -t 20 -J frogs_preprocess -o $LOG_DIR/%x.out -e $LOG_DIR/%x.err --cpus-per-
task=8 --mem=32G --wrap="preprocess.py illumina --input-archive
$SEQ_DIR/16Sv3v4_Mock_adn_bact.tar.gz --R1-size 250 --R2-size 250 --merge-software vsearch --
min-amplicon-size 300 --max-amplicon-size 490 --five-prim-primer ACGGRAGGCAGCAG --three-prim-
primer AGGATTAGATACCCTGGTA --nb-cpus 8 --output-dereplicated $RES_DIR/01-preprocess.fa --
output-count $RES_DIR/01-preprocess.count.tsv --summary $RES_DIR/01-preprocess.html --log-
file $LOG_DIR/01-preprocess.log")


#clustering
job2=$(sbatch --parsable --dependency afterok:$job1 -t 20 -J frogs_clustering -o
$LOG_DIR/%x.out -e $LOG_DIR/%x.err --cpus-per-task=8 --mem=32G  --wrap="clustering.py --nb-
cpus 8 -d 1 --fastidious --input-fasta $RES_DIR/01-preprocess.fa --input-count $RES_DIR/01-
preprocess.count.tsv --output-biom $RES_DIR/02-clustering.biom --output-fasta $RES_DIR/02-
clustering.fasta  --output-compo $RES_DIR/02-clustering_compo.tsv --log-file $LOG_DIR/02-
clustering.log")
```

Pour les utilisateurs de FROGS sous la plateforme Migale

https://documents.migale.inrae.fr/posts/tutorials/frogs16S-command-line/