Une introduction aux workflows

Denis Puthier, Stéphanie Le gras,
Tao Ye, Elodie Darbo, Morgane Thomas-Chollier, Rachel Legendre

École de bioinformatique AVIESAN-IFB-INSERM 2023

Outline

- **Objectif** : lancer le même outil (*fastqc*) sur 3 fichiers *fastq* différents
- **Concepts** :
  - Ecriture d'un script bash
  - Déclaration de variables pour généraliser les échantillons et les répertoires de travail
  - Réalisation d'une boucle pour lancer l'outil sur chaque échantillon
  - Premier script SLURM

- **CODE in HTML/Rmd format:**
  - https://dputhier.github.io/intro_workflow/

Démarrage d’une instance sur Jupyter

Objectif : lancer le même outil (FastQC) sur 3 échantillons Fastq différents

2325_ebaii

4 CPUs, 6 Go RAM

Les variables

- Définition: ce sont des éléments qui **associent un nom** (l'identifiant) **à une valeur**, qui sera sauvegardée dans la mémoire du système programmé.
- Il existe un certain nombre de variables systèmes pré-définies

```
$ echo $USER

$ echo $PWD

$ echo $SHELL

$ env | head # Liste les variables système
```

Les variables

- *On peut créer ses propres variables*
- *Il faudra préfixer les variables avec le caractère '$' pour y faire référence*

```
$ PROJECT=project_dputhier # Attention, pas d'espace autour de l'opérateur "="

$ echo ${PROJECT} # ou echo $PROJECT
project_dputhier

$ echo /shared/projects/${PROJECT}
/shared/projects/project_dputhier

$ WORKDIR=/shared/projects/${PROJECT}

$ echo ${WORKDIR}
```

Préparation de l'environnement de travail

```
$ mkdir -p ${WORKDIR}/tp_workflow/
$ cd ${WORKDIR}/tp_workflow/
$ mkdir data
$ cd data
$ wget https://zenodo.org/records/10081865/files/FILE1.fastq.gz
$ wget https://zenodo.org/records/10081865/files/FILE2.fastq.gz
$ wget https://zenodo.org/records/10081865/files/FILE3.fastq.gz
$ cd ..
$ touch script_01.sh
```

Contenu du fichier “script_01.sh” (à éditer)

```
#!/bin/bash

mkdir -p fastqc

module load fastqc/0.11.9


fastqc --outdir fastqc data/FILE1.fastq.gz

fastqc --outdir fastqc data/FILE2.fastq.gz

fastqc --outdir fastqc data/FILE3.fastq.gz
```

Lancement du workflow

- Lancez votre workflow dans le terminal

```
$ bash script_01.sh
```

Si besoin: `wget https://zenodo.org/records/10084017/files/script_01.sh`

script_02.sh: avec définition de la variable SAMPLE

```
#!/bin/bash

mkdir -p fastqc

module load fastqc/0.11.9

SAMPLE=FILE1
echo ">>> Processing $SAMPLE"
fastqc --outdir fastqc data/${SAMPLE}.fastq.gz

SAMPLE=FILE2
echo ">>> Processing $SAMPLE"
fastqc --outdir fastqc data/${SAMPLE}.fastq.gz

SAMPLE=FILE3
echo ">>> Processing $SAMPLE"
fastqc --outdir fastqc data/${SAMPLE}.fastq.gz
```

Lancement du workflow

- Lancez votre workflow dans le terminal

```
$ bash script_02.sh
```

Si besoin: `wget https://zenodo.org/records/10084017/files/script_02.sh`

script_03.sh: on redirige les sorties

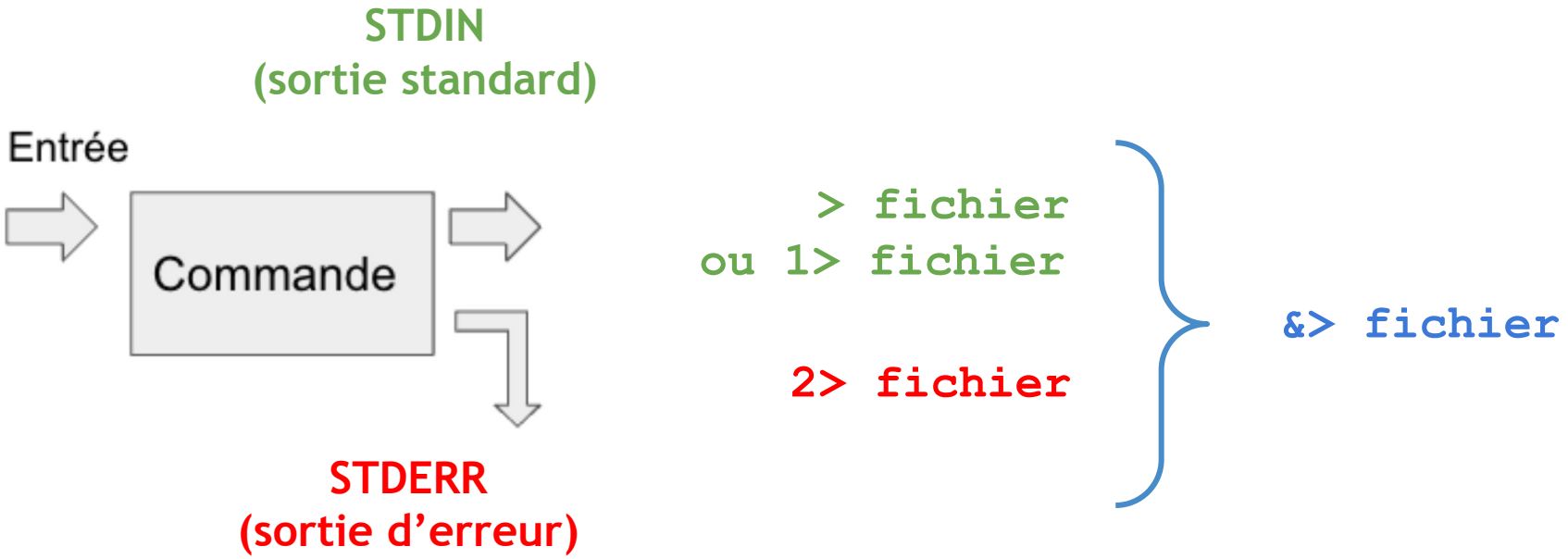
```
#!/bin/bash

mkdir -p fastqc

module load fastqc/0.11.9

SAMPLE=FILE1
echo ">>> Processing $SAMPLE"
fastqc --outdir fastqc data/${SAMPLE}.fastq.gz 2> fastqc/${SAMPLE}.log

SAMPLE=FILE2
echo ">>> Processing $SAMPLE"
fastqc --outdir fastqc data/${SAMPLE}.fastq.gz 2> fastqc/${SAMPLE}.log

SAMPLE=FILE3
echo ">>> Processing $SAMPLE"
fastqc --outdir fastqc data/${SAMPLE}.fastq.gz 2> fastqc/${SAMPLE}.log
```

Question (10 pts)

Comment rediriger le message “Analysis complete for FILE1.fastq.gz” ?

Réponse

Utiliser l'opérateur &> qui permet de rediriger STDIN (la sortie standard) et STDERR (la sortie d'erreur) dans un même fichier



Lancement du workflow

- Lancez votre workflow dans le terminal

```
$ bash script_03.sh
```

Si besoin: `wget https://zenodo.org/records/10084017/files/script_03.sh`

Utilisation des boucles ‘for’

- Une boucle permet de réaliser des tâches itératives.

```
$ for PRENOM in Rachel Stephanie Tao Maelle Denis
  do
    echo ${PRENOM}
  done
```

- On peut itérer sur une liste de fichiers

```
$ for FILE in data/*
  do
    echo ${FILE}
  done
```

- On peut itérer sur le résultat d’une commande

```
$ for READ in $(ls data/* | grep -v FILE1)
  do
    echo ${READ}
  done
```

Script_04.sh: itération sur les fichiers fastq

```
#!/bin/bash

mkdir -p fastqc

module load fastqc/0.11.9

for SAMPLE in FILE1 FILE2 FILE3
do
    echo ">>> Processing $SAMPLE"
    fastqc --outdir fastqc data/${SAMPLE}.fastq.gz &> fastqc/${SAMPLE}.log
done
```

Lancement du workflow

- Lancez votre workflow dans le terminal

```
$ bash script_04.sh
```

Si besoin: `wget https://zenodo.org/records/10084017/files/script_04.sh`

script_05.sh: itération sur le résultat d'une commande

```
#!/bin/bash

mkdir -p fastqc

module load fastqc/0.11.9

for SAMPLE in $(ls --color=none data/ | sed 's/.fastq.gz//')
do
    echo ">>> Processing $SAMPLE"
    fastqc --outdir fastqc data/${SAMPLE}.fastq.gz &> fastqc/${SAMPLE}.log
done
```

Lancement du workflow

- Lancez votre workflow dans le terminal

```
$ bash script_05.sh
```

Si besoin: `wget https://zenodo.org/records/10084017/files/script_05.sh`

Ajout des options pour SLURM

- On commence à décrire un peu plus précisément les ressources nécessaires à notre étape

```
#!/bin/bash

#SBATCH --account=$USER
#SBATCH --job-name=fastqc_test
#SBATCH --account=2325_ebaii # Modifier en fonction du projet
#SBATCH --cpus-per-task=1 # Modifier en fonction des besoins
#SBATCH --mem=4GB         # Idem

module load ...
...
```

- Pleins d'autres options utiles:
  - voir intro SLURM et https://ifb-elixirfr.gitlab.io/cluster/doc/quick-start/

script_06.sh: ajout des options SLURM

```
#!/bin/bash

#SBATCH --partition=fast
#SBATCH --job-name=my_fastqc
#SBATCH --account=2325_ebaii # Modifier en fonction du projet
#SBATCH --cpus-per-task=1 # Modifier en fonction des besoins
#SBATCH --mem=4GB         # Idem

mkdir -p fastqc

module load fastqc/0.11.9

for SAMPLE in $(ls --color=none data/ | sed 's/.fastq.gz//')
do
    echo ">>> Processing $SAMPLE"
    srun --job-name $SAMPLE fastqc --outdir fastqc data/${SAMPLE}.fastq.gz
done
```

Lancement du workflow

- Cette fois ci, on lance le script avec la commande **sbatch**

```
$ sbatch script_06.sh
```

Si besoin: `wget https://zenodo.org/records/10084017/files/script_06.sh`

- On monitore avec squeue

```
$ squeue
$ squeue -u $USER
$ sacct -u $USER
$ sacct -u $USER | tail -n 1
```

script_07.sh: parallélisation des tâches

- Au lieu de lancer chaque job l'un après l'autre, de manière séquentielle, on va les lancer en parallèle. On utilise un "job array".

```
#!/bin/bash
#SBATCH --partition=fast
#SBATCH --job-name=my_fastqc
#SBATCH --account=2325_ebaii # Modifier en fonction du projet
#SBATCH --cpus-per-task=1 # Modifier en fonction des besoins
#SBATCH --mem=4GB         # Idem
#SBATCH --array=1-3 # Modifier en fonction du nb de tâches à lancer en parallèle

mkdir -p fastqc
module load fastqc/0.11.9

# le Nième fichier de ma liste
SAMPLE=$(ls --color=none data/ | sed 's/.fastq.gz//' | \
         head -n ${SLURM_ARRAY_TASK_ID} | tail -n 1)

srun --job-name $SAMPLE fastqc --outdir fastqc data/${SAMPLE}.fastq.gz
```

Lancement du workflow

- Cette fois ci, on lance le script avec la commande **sbatch**

```
$ sbatch script_07.sh
```

Si besoin: `wget https://zenodo.org/records/10084017/files/script_07.sh`

- On monitore avec squeue

```
$ sacct -u $USER | grep RUNNING | grep FILE
```

script_08.sh: ajouter une étape

```
#!/bin/bash
#SBATCH --partition=fast
#SBATCH --job-name=my_fastqc
#SBATCH --account=2325_ebaii # Modifier en fonction du projet
#SBATCH --cpus-per-task=1 # Modifier en fonction des besoins
#SBATCH --mem=4GB         # Idem
#SBATCH --array=1-3 # Modifier en fonction du nb de tâches à lancer en parallèle

mkdir -p fastqc
module load fastqc/0.11.9
mkdir -p trimmomatic
module load trimmomatic/0.39

# le Nième fichier de ma liste
SAMPLE=$(ls --color=none data/ | sed 's/.fastq.gz//' | \
         head -n ${SLURM_ARRAY_TASK_ID} | tail -n 1)

srun --job-name FASTQC-$SAMPLE fastqc --outdir fastqc data/${SAMPLE}.fastq.gz
srun --job-name TRIM-$SAMPLE trimmomatic SE -threads 4 -phred33 \
                             data/${SAMPLE}.fastq.gz  trimmomatic/${SAMPLE}.fastq.gz \
                             SLIDINGWINDOW:4:20 MINLEN:20
```

Lancement du workflow

- Cette fois ci, on lance le script avec la commande **sbatch**

```
$ sbatch script_08.sh
```

Si besoin: `wget https://zenodo.org/records/10084017/files/script_08.sh`

- On monitore avec squeue

```
$ sacct -u $USER | grep RUNNING | grep FILE
```