Workflow

Nadia Bessoltane - INRAE
Vivien Deshaies - AP-HP

École de bioinformatique AVIESAN-IFB-INSERM 2022

Maria Bernard - INRAE
Elodie GIRARD - Institut Curie

Notion de workflow / pipeline

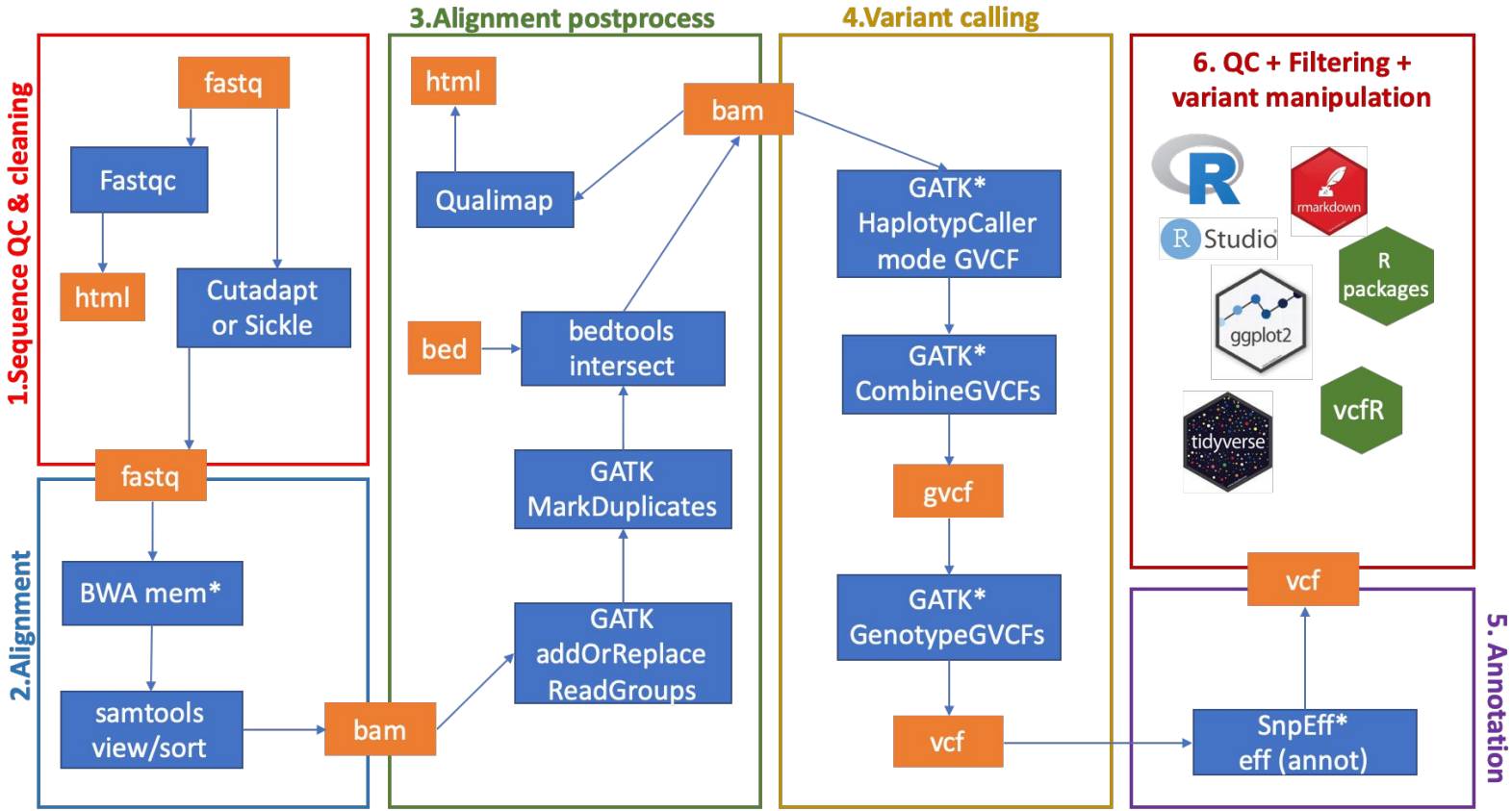
- **Workflow** : enchaînement d'étapes individuelles
- Ecriture sous forme d'un **script** en bash

→ Commence par un **"sha-bang" (#!)** qui indique l'interpréteur du script (**#!/bin/bash**)

→ Les lignes commençant par un **"#"** sont des commentaires et ne sont pas interprétées

→ Créer des **variables** pour généraliser votre script (pas spécifique à un échantillon)

Workflow de détection de small-variants



* need specific index

Exercice

Objectif : lancer le même outil (Fastqc) sur 6 échantillons Fastq différents

Nécessite :

- 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

```
$ mkdir -p  ~/tp_variant/workflow
$ cd ~/tp_variant/workflow

$ ls ~/tp_variant/fastq

$ touch fastqc.sh
```

Script1: écriture des lignes de commandes

```
#!/bin/sh

mkdir -p fastqc_res

module load fastqc/0.11.9

fastqc --outdir fastqc_res ~/tp_variant/fastq/SRR1262731_extract_R1.fq.gz


fastqc --outdir fastqc_res ~/tp_variant/fastq/SRR1262731_extract_R2.fq.gz
```

Utilisation de variable

Une variable permet d'anonymiser un script.

```
$ PRENOM="Maria"
```

'PRENOM' est le nom de la variable, 'Maria' est sa valeur
On peut ensuite utiliser une variable dans une ligne de commande

```
# la commande echo, affiche les arguments qui lui sont donnés
$ echo ${PRENOM}
```

Créez :

- une variable DATA_DIR qui prendra comme valeur le nom du dossier qui contient les fichiers fastq
- deux variables R1 et R2 qui correspondront aux noms de deux fichiers fastq R1 et R2.

Script2: anonymisation avec des variables

```
#!/bin/sh

mkdir -p fastqc_res



module load fastqc/0.11.9

fastqc --outdir fastqc_res ~/tp_variant/fastq/SRR1262731_extract_R1.fq.gz



fastqc --outdir fastqc_res ~/tp_variant/fastq/SRR1262731_extract_R2.fq.gz
```

Script2: anonymisation avec des variables

```
#!/bin/sh

mkdir -p fastqc_res
DATA_DIR="~/tp_variant/fastq"
R1=${DATA_DIR}"/SRR1262731_extract_R1.fq.gz"
R2=${DATA_DIR}"/SRR1262731_extract_R2.fq.gz"

module load fastqc/0.11.9


fastqc --outdir fastqc_res ${R1}




fastqc --outdir fastqc_res ${R2}
```

Utilisation d'une boucle

Une boucle va permettre d'itérer sur une liste de valeur pour une variable

```
$ for PRENOM in Elodie Nadia Olivier Mathieu Odile
  do
        echo ${PRENOM}
  done
```

A partir de la liste des fichiers R1 et R2 du dossier DATA_DIR, créez deux boucles (une pour les fichiers R1 et une pour les fichiers R2) pour lancer la ligne de commande fastqc sur tous les fichiers du dossier fastq.

Script3: automatisation sur plusieurs valeurs

```
#!/bin/sh

mkdir -p fastqc_res
DATA_DIR="~/tp_variant/fastq"
R1=${DATA_DIR}"/SRR1262731_extract_R1.fq.gz"
R2=${DATA_DIR}"/SRR1262731_extract_R2.fq.gz"

module load fastqc/0.11.9


fastqc --outdir fastqc_res ${R1}




fastqc --outdir fastqc_res ${R2}
```

Script3: automatisation sur plusieurs valeurs

```
#!/bin/sh

mkdir -p fastqc_res
DATA_DIR="~/tp_variant/fastq"

# charger l'outil fastq
module load fastqc/0.11.9

# lancer fastqc avce les fastq R1
for R1 in $DATA_DIR/*_R1.fq.gz
do
    fastqc --outdir fastqc_res ${R1}
done

# lancer fastqc avce les fastq R2
for R2 in $DATA_DIR/*_R2.fq.gz
do
    fastqc --outdir fastqc_res ${R2}
done
```

Lancement du workflow

Lancez votre workflow avec une commande sbatch comme nous l'avons fait jusque là.

/!\ Attention de réserver les ressources clusters dont vous avez besoin /!\

```
$ # executer le script
$ bash fastqc.sh
```

Pour aller plus loin

Conseils pour écrire un workflow à plusieurs étapes

Conseils pour écrire un workflow à plusieurs étapes

1) écrire un script qui enchaîne l'ensemble des étapes pour seul 1 échantillon:

```
$ geany mapping_calling.sh &
```

1) définir des arguments pour ce script: R1 R2 GENOME SAMPLENAME ID OUT_DIR

1) écrire un script de lancement du script mapping.sh en boucle sur les différents échantillons (boucle "for" ou copier/coller de la ligne de commande mapping_calling.sh avec des nouvelles valeurs pour les arguments)

```
$ geany launch_DNASeq.sh &
```

Définition d'arguments dans un script

Au lieu de donner une valeur à nos variables R1et R2 dans le script, on va indiquer au script d'aller les chercher dans la ligne de commande comme des options du programme.

Dans le script la valeur est remplacée par $1,$2, … $n et dans la ligne de commande on ajoute dans l'ordre les valeurs des variable 1, 2, … n.

echo.sh

```
#!/bin/sh

PRENOM='Maria' ; NOM='BERNARD'
echo ${PRENOM} ${NOM}
```

```
$ sh echo.sh
```

echo.sh

```
#!/bin/sh

PRENOM=$1; NOM=$2
echo ${PRENOM} ${NOM}
```

```
$ sh echo.sh Maria BERNARD
```

mapping_calling.sh

```
#!/bin/sh
R1=$1; R2=$2 ; GENOME=$3; NAME=$4; ID=$5;  OUT_DIR=$6

LOG_DIR=${OUT_DIR}/logs; mkdir -p ${LOG_DIR}
module load fastqc/0.11.9; module load sickle-trim/1.33; # module load bwa/0.7.17 ...

mkdir -p ${OUT_DIR}/fastqc_res
fastqc --threads 4 --outdir ${OUT_DIR}/fastqc_res ${R1} 2>&1
${LOG_DIR}/${NAME}_fastqc_R1.out
fastqc --threads 4 --outdir ${OUT_DIR}/fastqc_res ${R2} 2>&1
${LOG_DIR}/${NAME}_fastqc_R2.out

mkdir -p ${OUT_DIR}/sickle_res
sickle pe -f ${R1} -r ${R2} -t sanger -g \
   -s ${OUT_DIR}/sickle_res/${NAME}_trim_unpaired.fq.gz \
   -o ${OUT_DIR}/sickle_res/${NAME}_trim_R1.fq.gz \
   -p ${OUT_DIR}/sickle_res/${NAME}_trim_R2.fq.gz \
   > ${LOG_DIR}/${NAME}_sickle.log.txt
#... et on continue avec le mapping et les autres étapes
```

launch_DNASeq.sh (version simple)

```
#!/bin/sh
GENOME="~/tp_variant/genome/Bos_taurus.UMD3.1.dna.toplevel.6.fa"
OUT_DIR="~/tp_variant/dnaSeq_results"
DATA_DIR="~/tp_variant/fastq"
LOG_DIR=${OUT_DIR}/logs
mkdir -p ${LOG_DIR}

# sample SRR1262731
sbatch -J SRR1262731_dnaseq -o ${LOG_DIR}/SRR1262731_dnaseq.out \
    -e ${LOG_DIR}/SRR1262731_dnaseq.err --cpus-per-task=4 --mem=8G \
    --wrap="mapping_calling.sh ${DATA_DIR}/SRR1262731_extract_R1.fq.gz \
    ${DATA_DIR}/SRR1262731_extract_R2.fq.gz $GENOME SRR1262731 1 ${OUT_DIR}"

# sample SRR1205992
sbatch -J SRR1205992_dnaseq -o ${LOG_DIR}/SRR1262731_dnaseq.out \
    -e ${LOG_DIR}/SRR1205992_dnaseq.err --cpus-per-task=4 --mem=8G \
    --wrap="mapping_calling.sh ${DATA_DIR}/SRR1262731_extract_R1.fq.gz \
    ${DATA_DIR}/SRR1205992_extract_R2.fq.gz $GENOME SRR1205992 2 ${OUT_DIR}"
```

launch_DNASeq.sh (version avec boucle)

```
#!/bin/sh
GENOME="~/tp_variant/genome/Bos_taurus.UMD3.1.dna.toplevel.6.fa"
OUT_DIR="~/tp_variant/dnaSeq_results"
DATA_DIR="~/tp_variant/fastq"
LOG_DIR=${OUT_DIR}/logs
mkdir -p ${LOG_DIR}

ID=0
for R1 in ${DATA_DIR}/*_R1.fq.gz
do
let ID=${ID}+1
R2=`echo ${R1} | sed 's/_R1.fq.gz/_R2.fq.gz/'`
NAME=`basename ${R1} | sed 's/_R1.fq.gz//'`
sbatch -J ${NAME}_dnaseq -o ${LOG_DIR}/${NAME}_dnaseq.out \
    -e ${LOG_DIR}/${NAME}_dnaseq.err --cpus-per-task=4 --mem=8G \
    --wrap="mapping_calling.sh ${DATA_DIR}/${NAME}_extract_R1.fq.gz \
    ${DATA_DIR}/${NAME}_extract_R2.fq.gz $GENOME ${NAME} ${ID} ${OUT_DIR}"
done
```

Reprise du workflow : définition

[Vidéo] : **The 5 minutes IFB Core Cluster tutorial**

[Cheatsheet]
https://ifb-elixirfr.github.io/EBAII/2021/ebaiin1/DNA-seq/EBAII2021_variants.html

- **Workflow** : enchaînement d'étapes individuelles
- Ecriture sous forme d'un **script** en bash

→ Commence par un **"sha-bang" (#!)** qui indique l'interpréteur du script (**#!/bin/bash**)

→ Les lignes commençant par un "**#**" sont des commentaires et ne sont pas interprétées

→ Créer des **variables** pour généraliser votre script (pas spécifique à un échantillon)